OPEN LIB

OPEN LIB ALA
Engineering Guide

OPEN LIB ALA
エンジニアリングガイド

2014 年 5 月

オーエスエスブロードネット株式会社

第 0.5 版 2014 年 5 月

Printed in Japan

改版履歴

版数	改版年月日	変更内容	備考
0.1	2013/11/26	Edition5 Interrim 版。	加筆用
0.2	2013/12/27	1～3 章を執筆。	
0.3	2014/01/20	4～5 章を執筆。	開発者への改良仕様提示目的。
0.4	2014/04/20	OPEN LIB シリーズ全体の仕様整合を目的とした文言・内容の加筆修正。	改良のレビュー結果を反映。
0.5	2014/05/28	OPEN EMS シリーズのエンジニアリングガイド改版に伴う内容の整合性確保。	

参考文献

RFC5256 - The Transport Layer Security (TLS) Protocol Version 1.2
「基礎暗号学Ｉ」加藤正隆著（サイエンス社）
「現代暗号」岡本龍明・山本博資著（産業図書）
「暗号理論」伊藤正史著（ナツメ社） 他

本書の目的

本書は、OPEN LIB ALA の導入設計・運用設定・保守に必要な情報をまとめたものです。
本書は、OPEN LIB に主体的に係わるパートナー各社とエンドユーザーの技術者および、OPEN LIB に興味を持って下さった全ての方々に対する情報開示を目的に作成されています。本書のインターネット上での再配布および部分的な流用は、個人・法人を問わず自由に行えますが、弊社に著作権の帰属する情報の二次利用に際しては、弊社の著作権を明示して下さい。

本書の対象読者

本書は、OPEN LIB ALA の導入設計・運用設定に従事する SE、保守を行う CE および、システム拡張を行うプログラマを対象にしています。
以下の技術に関する知識があると、本書の理解が一層容易になります。
スマートグリッド、IEC、DLMS/COSEM、ANSI、SSL、TLS、Web サービス、UDP、TCP/IP、Linux、Java、Servlet/JSP、Apache、Tomcat、Firebird、MySQL

その他

OPEN LIB に関する技術的なご質問は、E-Mail により以下まで送信して下さい。
info@ossbn.co.jp

弊社の知的財産権に含まれない規格・技術の記述や情報の更新・バグに関し、弊社では一切の責任を負いませんのでご了承下さい。

謝辞

本製品の Edition 5 改良開発に先立ち、スマートメーターに関連する各国の技術動向・規格標準化動向、および実際の普及状況調査に全面的にご協力・ご指導を頂き、かつ多くの有用な参考資料を快くご提供頂いた、NTT データの諸兄と三重大学の内藤　克浩　先生に対し、弊社社員一同より心からの御礼を申し上げます。

目次

第1章　システム概要

1.1.　ALA の概念

OPEN LIB（OPEN LIBrary product series）は、様々な業務システムへの応用が可能な、業界標準の技術規格・仕様に基づくソフトウェア開発基盤製品シリーズです。OPEN LIB ALA（Application Level Authentication: 以降「ALA」）は、ネットワークの任意の区間をアプリケーションレベルで認証･暗号化するソフトウェアであり､任意のアプリケーション間の相互認証、共通鍵生成、定期的な共通鍵更新の仕組みを提供します。ALA は、末端の測定器・センサー等から情報を収集する際の通信（集信）と、収集情報を集計・変換＆個人情報を組み合わせた形で加工し、インターネット等の公衆網経由で公共サービスとして情報提供する際の通信（配信）の双方への応用が可能です。

ALA の適用分野は、電力・ガス・水道の遠隔検針や定置型バッテリーの状態監視、見守り・健康管理等、個人情報保護や公安上の観点から一定水準以上のセキュリティが求められる、不特定・複数のネットワークを介したアプリケーションサービスです。

本節では ALA の適用対象の一つとして、電力の遠隔検針におけるヘッドエンドとスマートメーター間集信への応用例について説明します。

スマートメーターA/C ルート通信時の POI セキュリティ問題を図 1.1(1)に示します。

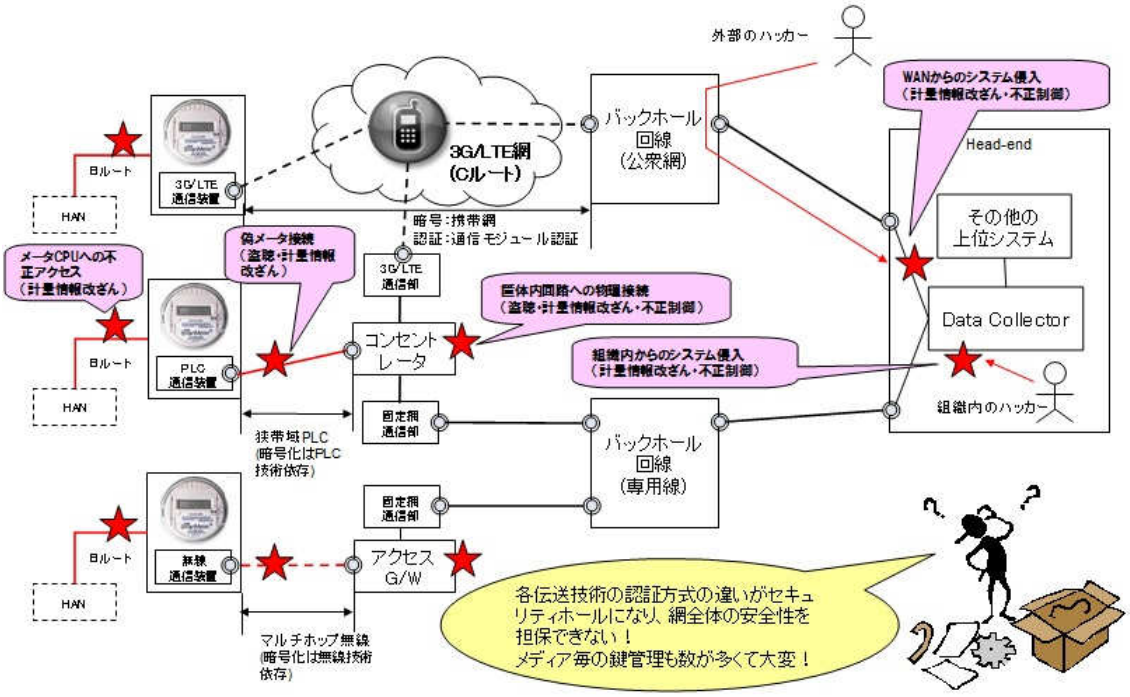

図 1.1(1)スマートメーターA/C ルート通信時の POI セキュリティ問題

文書番号：OSSBN-EGMN-13-11-004

バックホール網に 3G/LTE、アクセス網に G3-PLC を使用してスマートメーターから使用量情報を集信する場合、両網を中継するコンセントレーターがセキュリティ上の POI（相互接続点）として機能します。バックホール網の認証・暗号化には 3G/LTE キャリアのセキュリティポリシーが適用される一方で、アクセス網には G3-PLC の仕組みが適用されるため、POI であるコンセントレーターが、両者の違いを調整・仲介する役割を果たします。結果として、コンセントレーターの認証・暗号化機構上の役割が相対的に高まり、両網の暗号化鍵を集中管理するインテリジェントな設備として機能する事になります。

一方で設置数の多さから、コンセントレーター筐体の物理的な構造強化や有人定期点検業務による監視体制の強化は、コストとのトレードオフです。コストダウンの徹底により、コンセントレーター筐体の物理的な構造が脆弱となり、かつ定期点検の頻度も少ないと、開扉＆プローブ接続等の物理的な攻撃により、ハッカーからコンセントレーターを標的にした攻撃を集中的に仕掛けられ、コンセントレーターが潜在的なセキュリティホールとなるリスクが生じます。

仮にハッカーにコンセントレーターを乗っ取られた場合、各スマートメーターからの検針データ盗聴による個人の生活パターンが検知され、不在侵入窃盗や暴行・強盗等の犯罪を助長するかも知れません。ないしは、メーターへの給電遮断命令の不正発行による特定個人・法人への嫌がらせや社会騒乱等、不特定個人・法人に対するテロ的な活動を助長するかも知れません。更に、計量情報の改竄による公共財・サービスの不正利用等も懸念されます。

ALA は、対象のネットワークが複数伝送メディアの組み合わせ、ないしは複数ネットワーク事業者の組み合わせ構成で、POI 付近のセキュリティホールが懸念される上述のような場合に特に有効です。ALA により、ネットワーク区間毎に異なるセキュリティや暗号化方式の整合調整というリスクの高い手法を避け、エンド-エンドのセキュリティをアプリケーションレベルで合理的に確立できます。

ALA による POI セキュリティホール問題への対処を図 1.1(2)に示します。

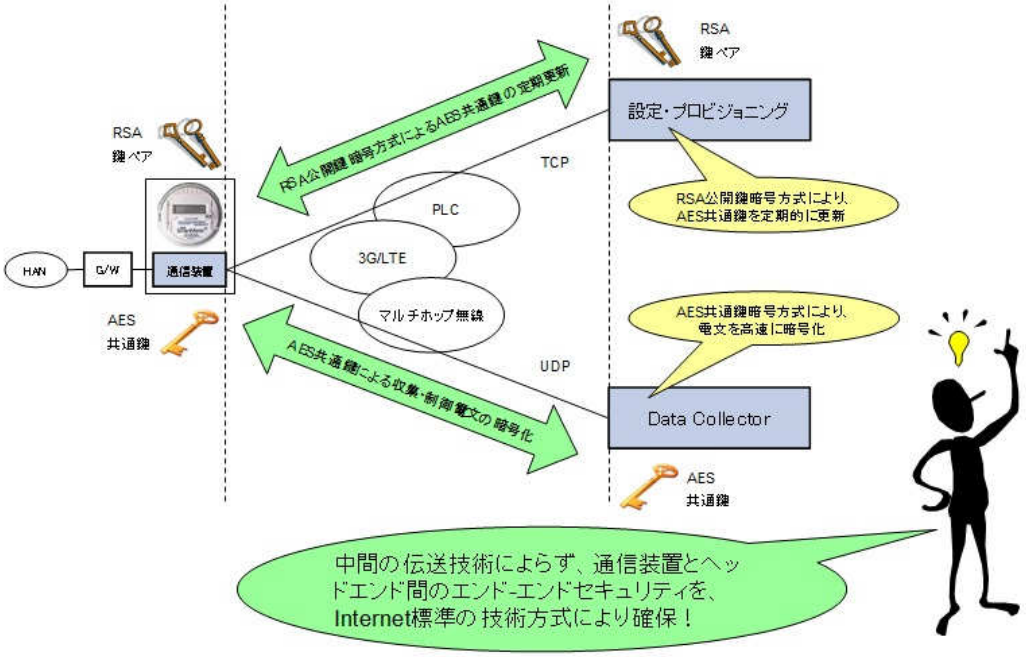

図 1.1(2) ALA による POI セキュリティホール問題への対処

文書番号：OSSBN-EGMN-13-11-004

本対処例では、PLC、3G/LTE、マルチホップ無線等の中間の伝送メディアの認証・暗号化方式によらず、スマートメーター通信装置とヘッドエンドデータコレクターのエンド～エンド間で、公開鍵暗号方式により共通鍵を定期更新し、かつ共通鍵により収集・制御電文を暗号化しています。中間の伝送技術によらず、エンド～エンド間の通信がアプリケーションレベルで認証・暗号化されるため、中間の POI を物理的に攻撃された場合にも通信の秘匿性が保証され、なりすましによる不正操作を検知・防止できます。

本節ではヘッドエンド～スマートメーター間の定期集信に限定して説明しましたが、エンド～エンド間をアプリケーションレベルで認証・暗号化する手法は、より幅広い分野・領域への適用が可能です。例えばスマートメーター関連だと、電力使用量情報の集信だけでなく、収集した電力使用量情報を「見える化」する配信サービスの認証・暗号化にも応用できます。ALA により、中間の各伝送メディアの認証・暗号化方式に依存せず、共通的な仕組みで「見える化」データを認証・暗号化できるため、例えばインターネットを介した見える化情報の配信サービス等も可能になります。

ALA には独自方式の排他的な要素技術が含まれておらず、全機能がインターネット標準の技術方式の組み合わせにより構成されています。このため ALA の適用分野・領域は、特にスマートメーター関連技術に限定されるものではなく、例えばホームセキュリティやホームオートメーション等、様々な分野・領域への応用が可能です。

認証・暗号化の技術要素は、①通信内容の暗号化（暗号強度）②暗号鍵の生成・更新（鍵配送問題）③通信対象の相互確認（認証）④通信内容の改竄検知（正真性証明）により構成されます。①の要素技術は DES や AES、②は RSA や楕円曲線暗号、③は証明書認証や PKI、④は一方向性関数やハッシュ化などであり、これらを組み合わせたインターネットの事実上の標準として、特に電子商取引の世界では、SSL/TLS が広く普及しています。

認証・暗号化技術の世界で、SSL/TLS のように既に総合的な意味での有用性が立証された技術を分解・組み替えし、新たな独自方式を構成する行為は危険です。

このため ALA では、SSL/TLS を使える限りはそのまま使う事を推奨しています。

一方で、PC 上のブラウザのようなリッチな環境と異なり、スマートメーターや各種センサー・STB 等、性能・資源が不足気味の低コスト端末の場合、単純には SSL/TLS を組み込めません。

例えば SSL/TLS の公開鍵認証・暗号化方式によるセッション毎の鍵更新には、実行環境によっては数秒～十数秒を要しますが、アプリケーションによっては秒単位の処理遅延が許されない場合があります。また、端末側からクライアント認証を実行する場合、端末側に PRNG（擬似乱数発生）機能が必要になりますが、PRNG の性能が低いと暗号文に規則性が生じ、暗号強度が下がります。更に、各端末に組み込む X.509 証明書の発行・管理主体にも考慮が必要です。というのは事業者が各端末機器の型式と流通経路･提供サービスを主体的に管理するシステムでは、証明書バージョンや認証・暗号化技術の選択等の運用上の観点で、商用 CA よりもプライベート CA の方がより柔軟であり、コスト面でも有利な場合があるからです。

ALA は上述のような場合に、独自方式への依存を極力避けながら、SSL/TLS への最低限の調整により、所望のセキュリティ性能を達成するための選択肢と代替案を事業者に提供します。

1.2. ALA のシステム構成

ALA は、X.509 証明書を発行するプライベート CA 局機能である ALA-C(Certification)と、ヘッドエンド～各端末機器間で定期的に鍵更新セッションを実行する ALA-S(8ession)から構成されます。

集信時のシステム構成例として、スマートメーターとの ALA 構成例を図 1.2 (1)に示します。

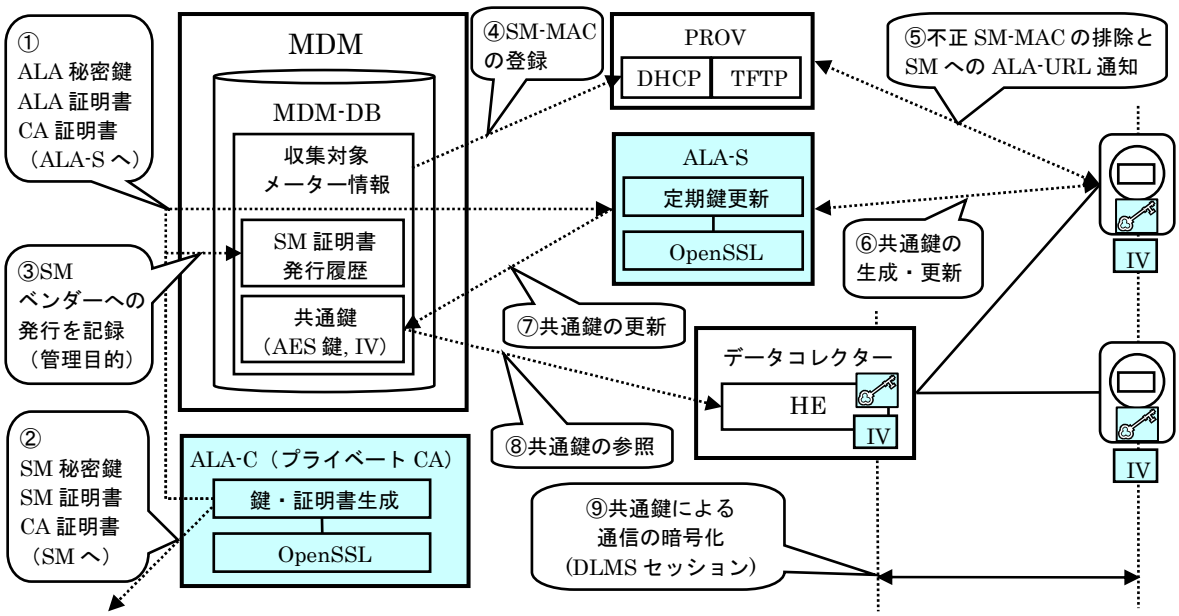

図 1.2 (1) スマートメーターとの ALA 構成例

図中の「SM」はスマートメーター、「CA」は Certificate Authority すなわち ALA-C、「証明書」は公開鍵とデジタル署名を含んだ X.509 証明書、「秘密鍵」は通信に使用しないマスタ鍵、「PROV」はプロビジョニングサーバーを表します。

①は運用に先立つ前準備であり、ALA-C の証明書作成機能により、ALA-C のルート CA 証明書、ALA-S の ALA 秘密鍵、ALA 証明書を作成し、ALA-S に登録します。

SM の追加調達時、②ALA-C は SM 秘密鍵と SM 証明書ペアを必要数分作成し、CA 証明書と合わせて SM ベンダーに鍵情報を発行して製造を依頼後、③MDM-DB に発行履歴を記録します。SM ベンダーは、各 SM 内のセキュアな不揮発記憶領域に、SM 毎に異なる SM 秘密鍵・SM 証明書と全 SM 共通の CA 証明書を焼き込みます。

SM の製造・入庫時、④SM の MAC アドレスを PROV に登録します。

SM の設置・プロビジョニング時、PROV は⑤不正 SM-MAC からの設定要求を拒否し、登録済 SM-MAC からの設定要求のみ処理します。SM は⑥PROV から取得した ALA-S の URL（又は IP アドレス）により ALS-S と通信し、X.509 証明書のデジタル署名相互認証と公開鍵暗号方式により、HE～SM 間の共通鍵（AES 鍵・IV）を生成します。この時 ALA-S は、⑦生成された共通鍵情報を MDM-DB に更新します。HE は⑧MDM-DB の共通鍵情報を参照し、⑨アプリケーションである DLMS 通信を暗号化します。

共通鍵は、ALA-S により定期的に更新されます。更新頻度は、SM の処理能力や通信経路の特性・スループット等により調整します。

公開鍵暗号方式で AES128bit 相当の暗号強度を達成するには、概ね 2048bit 以上の鍵長が必要ですが、RSA 等の公開鍵暗号方式は AES 等の共通鍵暗号方式に比較して CPU・メモリの消費が桁違いに高く、SC のように性能・資源が制限される条件下で、通信の度に公開鍵暗号方式でセッション鍵を再生成する SSL/TLS は、SC への処理負荷が高すぎます。また、通信量の増大も無視できません。

ALA-S により、負荷の重い公開鍵暗号方式の実行がバックグラウンド化され、かつ頻度が可変となるため、SC の実行環境や通信経路の特性に応じた調整が可能になります。

なお ALA では、ALA-C を使ったプライベート CA 局の構築・運用を前提としていますが、ALA-C の代わりに商用 CA 局を使う場合でも、基本的な考え方・運用ルールは同じです。

ALA-S の認証･暗号化対象となる端末機器には､OPEN LIB EA(Embedded Agent: 以降｢EA｣)を組み込みます。EA により、ALA-S に対応するセキュリティ機能を、任意の端末機器に安全・容易に実装できます。EA の詳細については、EA の関連文書を参照して下さい。

次に、配信時のシステム構成例として、見える化アプリケーション端末との ALA 構成例を図 1.2 (2)に示します。

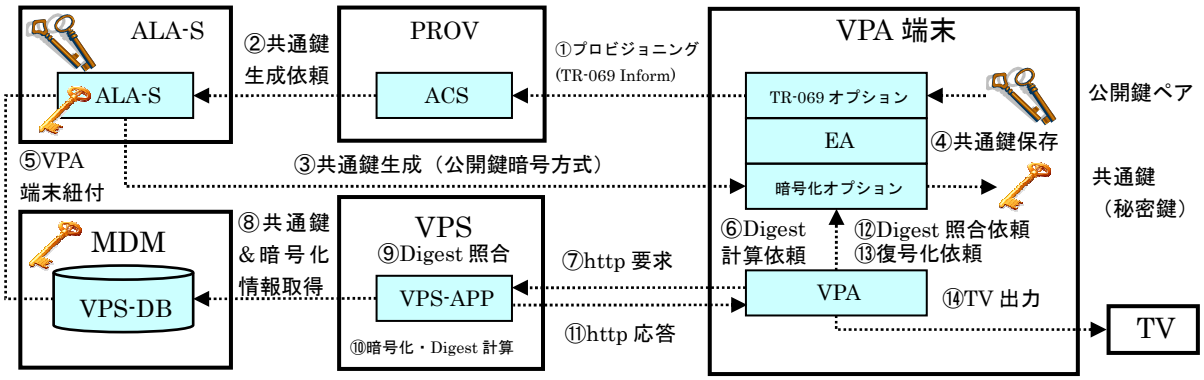

図 1.2 (2) 見える化アプリケーション端末との ALA 構成例

図中の「VPA」は OPEN EMS シリーズの見える化アプリケーション(Visualization of Power by Android)､｢VPS｣は VPA のサーバー側機能を表します。VPA, VPS の詳細については､VPA, VPS の関連文書を参照して下さい。

STB に https のクライアント認証を実装すると、SSL/TLS 公開鍵暗号方式のハンドシェイクが https セッションの度に発生します。STB の CPU が低速、又はサーバー間の通信回線が細いと、画面遷移や操作のたびに数秒単位の待ち時間が発生します。

PROV/ALA-S 構成では、VPA 端末のプロビジョニング完了時、VPA 端末内に予め組み込まれた EA と ALA-S 間で、公開鍵暗号方式により共通鍵を生成・更新し、ALA-S は自身のローカルファイルシステムに、EA は VPA 端末内の不揮発領域に共通鍵情報を保存します（①～④)。

以降の VPS～VPA 間通信は、https に比べると処理負荷が格段に軽い（数十ミリ秒単位）http に共通鍵暗号を組み合わせた方式により、全送受信データが暗号化されます。更に、共通鍵暗号を応用したログイン ID のダイジェスト化により、ログイン ID の内容が隠蔽されます。http＆共通鍵暗号＆ダイジェスト方式により、https 使用時の性能問題を回避できます。

共通鍵の再生成・更新の要否と更新周期（更新頻度）は、事業者のセキュリティポリシーや STB の性能、通信回線の速度に基づき、時間単位で ALA-S に指定します。

VPA 端末のプロビジョニングは、OTT-STB の場合は ACS/HTTP、CATV-STB の場合は DHCP/TFTP と異なりますが、①～④の流れは STB の端末種別によらず共通です。

VPA 端末の紐付操作時、ALA-S から VPS-DB に③で生成した共通鍵情報が伝播されます（⑤）。

見える化情報の要求時、VPA＆EA はダイジェストを付与し（⑥）、VPS に http 要求します（⑦）。VPS は、要求 URL のパラメーターであるログイン ID（VPA の MAC アドレス）に基づき共通鍵＆暗号化情報を VPS-DB から取得（⑧）、計算値と付与値のダイジェストを照合し（⑨）、OK なら処理を実行、出力がある場合には共通鍵暗号化、更に、応答 URL のダイジェストを計算・付与（⑩）、http 応答します（⑪）。VPA・EA はダイジェストを照合し（⑫）、OK なら情報を復号化（⑬）、TV に出力します（⑭）。ポリシーの登録・設定など、VPA から VPS に設定情報を送信時、逆の流れで VPA＆EA が暗号化＆ダイジェスト付与、暗号化された設定情報を VPS に送信、VPS がダイジェスト照合・復号化後、VPS-DB に復号化された設定情報を反映します。

上の 2 つの事例共、ALA-S は MDM ステーションに配置します。MDM ステーション概念では、MDM と PROV、ALA-S の関係は 1:1:1 です。

一方で ALA-C は、システムが大規模で複数の MDM ステーションが CE ドメインに収容される場合は CE ドメインに、システムが小規模で単一の MDM ステーションにより構成される場合は MDM ステーションに配置します。前者の場合、CE と ALA-C の関係は 1:1 です。

1.3. ALA-S の通信シーケンス

一般的な電子商取引のユースケースで、PC 上のブラウザから https を実行する場合と異なり、ALA-S のセッションシーケンスは、端末側ではなく ALA-S 側が開始します。

ALA-S のセッションシーケンス例を図 1.3 に示します。

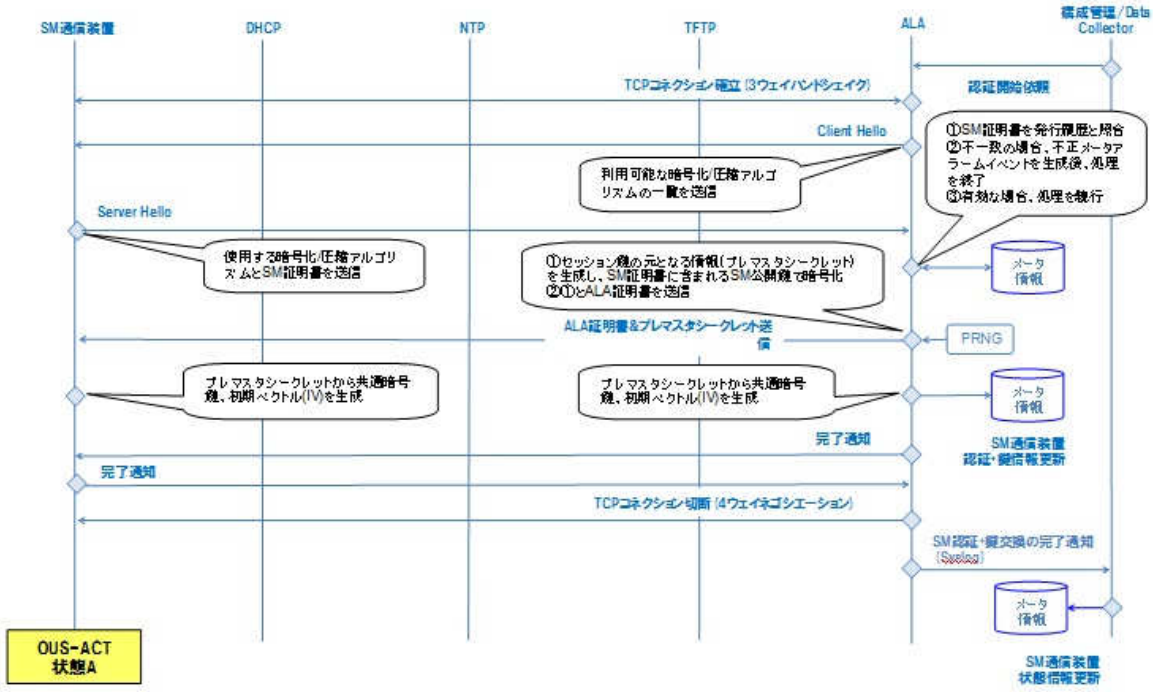

図 1.3 通信シーケンス例（スマートメーターの場合）

図中の「SM 通信装置」はスマートメーター、「ALA」は ALA-S、「構成管理/Data Collector」はプロビジョニングサーバーを表します。

構成管理からの認証開始依頼を受け、ALA が SM 通信装置に TCP コネクションを開設後、Client Hello を送信、対する SM 通信装置は ALA に Server Hello を返します。

以降は SSL/TLS 標準に沿って処理が進み、共通鍵の生成・更新後、ALA が SM 通信装置との TCP コネクションを切断、結果を構成管理に返します。共通鍵の生成・更新過程で、ALA 側の PRNG が使われている点に注意して下さい。

SSL/TLS 標準では、クライアント＆サーバー間の相互認証を行う場合、クライアント Hello を送信する側が PRNG（擬似乱数発生器）により乱数を生成する約束になっていますが、1.1 で述べた通り、PRNG の性能が低いと暗号文に規則性が生じ、暗号強度が下がります。

ALA-S 側からのセッション開始により、SSL/TLS 標準のルールに従いながら、ALA-S 実行環境の高性能な PRNG で乱数を生成できるため、所定の暗号強度を維持しながら、端末側の処理コストを低減できます。

1.4. ALA-C による鍵管理方式

「鍵配送問題」は、認証・暗号化システム設計・構築における重要事項の一つです。鍵管理方式の設計が不適切だと、人為ミス以前の構造的な問題により暗号鍵が流出する状況を招き、不正アクセスの直接的・間接的な原因となります。

ALA-C による鍵管理方式例を図 1.4 (1)に示します。

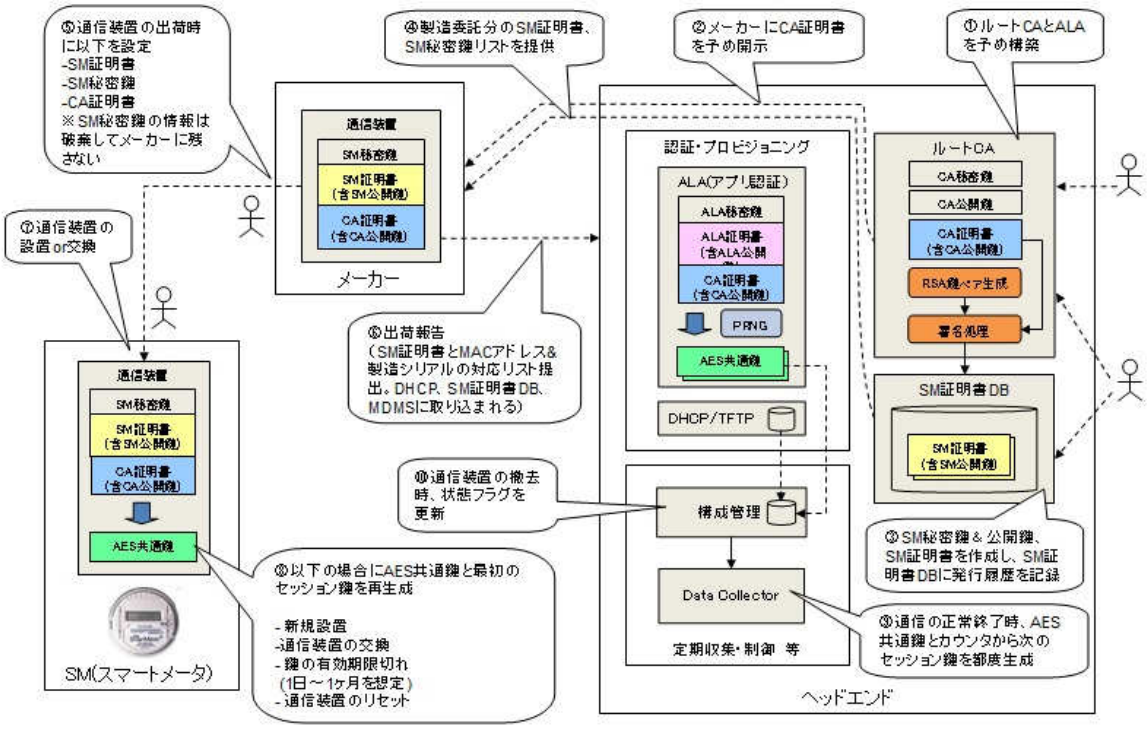

図 1.4 (1) 鍵管理方式例（スマートメーターの場合）

図中の「ルート CA」は ALA-C、「SM 証明書 DB」は MDM-DB、「構成管理」は MDM、「認証・プロビジョニング」中の「ALA（アプリ認証）」は ALA-S、「DHCP/TFTP」はプロビジョニングサーバーを表します。①～⑩の説明は、図 1.2(1)の説明と概ね重複するため省略します。

プライベート CA 局の構築を含めた証明書の作成から無効化までの包括的・主体的な鍵管理方式により、各種の不正検出と合理的なモニタリングが可能となります。

ALA による不正検出とモニタリング方式例を図 1.4 (2)に示します。

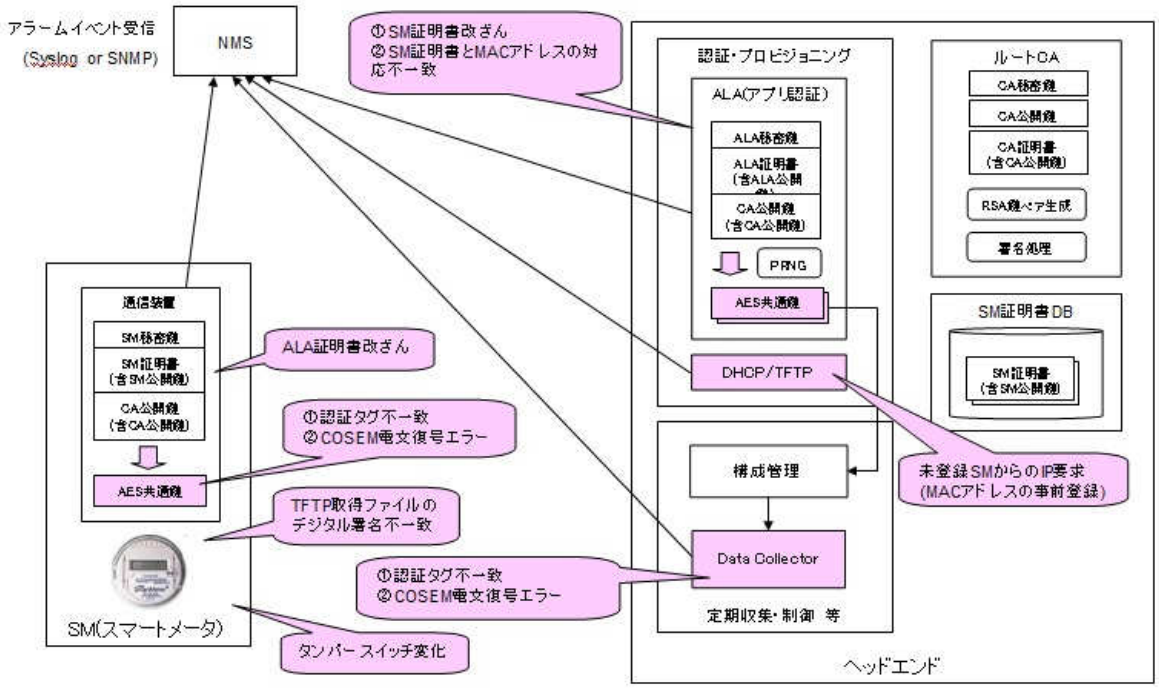

図 1.4 (2) 不正検知とモニタリング方式例（スマートメーターの場合）

図中の「NMS」は上位のネットワーク管理システムを表します。

表 1.4 (3)に、ALA により検知されるアラーム一覧を示します。

No.	内容	主体	判定根拠	タイミング	プロトコル
1	タンパースイッチ変化	メーター	センサー回路の状態変化	筐体の不正開扉時	syslog
2	ALA 証明書改ざん	メーター	CA 公開鍵により署名を復号し、ALA 公開鍵のハッシュ値と照合	プロビジョニング及び鍵の定期交換時	syslog
3	DLMS/COSEM 復号失敗	メーター	①パーサの復号エラー ②不明な OBIS/メソッドの指定	DLMS/COSEM 電文復号時	syslog
4	AES 認証タグ不一致	メーター	AES 共通鍵により受信側で認証タグを再計算・照合	DLMS/COSEM 電文受信時	syslog
5	TFTP イメージファイル署名不一致	メーター	CA 公開鍵により署名を復号し、ファイルの署名と照合	プロビジョニング時	syslog
6	メーター内部の重度障害	メーター	メーカ依存	メーカ依存	syslog
7	SM 証明書改ざん	ALA	CA 公開鍵により署名を復号し、SM 公開鍵のハッシュ値と照合	プロビジョニング及び鍵の定期交換時	snmp
8	他 SM 証明書の不正流用	ALA	SM 証明書 DB の発行履歴と照合し、MAC アドレスとの不一致又は証明書の失効を検出	プロビジョニング及び鍵の定期交換時	snmp
9	タンパースイッチ変化	メーター	センサー回路の状態変化	筐体の不正開扉時	syslog
10	ALA 証明書改ざん	メーター	CA 公開鍵により署名を復号し、ALA 公開鍵のハッシュ値と照合	プロビジョニング及び鍵の定期交換時	syslog
11	DLMS/COSEM 復号失敗	メーター	①パーサの復号エラー ②不明な OBIS/メソッドの指定	DLMS/COSEM 電文復号時	syslog

表 1.4 (3) アラーム一覧（スマートメーターの場合）

1.5. 機能一覧

ALA の機能一覧を表 1.5 に示します。

機能	分類	概要
プライベート CA 局(ALA-C)	標準	公開鍵ペアと証明書を生成します。
証明書発行履歴管理(ALA-C)		証明書の払出先と有効・失効等の状態履歴を管理します。
共有秘密鍵更新(ALA-S)		公開鍵暗号方式により共通鍵を生成します。 設定周期で共通鍵を定期的に更新します。

表 1.5 機能一覧

1.6. 動作環境

ALA-S の動作環境を表 1.6 に示します。

項目	仕様
CPU	3GHz クアッドコア・64bit
RAM	8GHz
HDD	40GB
NIC	GbE*2 ポート
OS	Linux カーネル 2.6.32 以降・64bit 版
必須ソフト	OpenSSL
必須サーバー	OPEN EMS MDM, OPEN LIB PROV
必須クライアント	OPEN LIB EA

表 1.6 動作環境

ALA の推奨動作環境は、システム規模により変わります。
本節に示す動作環境は、小規模構成による動作検証時の実環境情報です。

1.7. 注意事項

システム規模の設計に際しては､公開化暗号方式による共有秘密鍵の更新トランザクションの負荷を考慮して下さい。また、可用性要件に応じた冗長化を実施して下さい。

SM 証明書シリアル番号の照合動作は、現バージョンでは未実装です。

ALA シーケンスで交換する鍵情報は、TLS 通信確立時に生成されるものの流用ではなく、TLS 通信確立後に別途交換した鍵情報を利用します。

DLMS 通信の暗号方式には AES-CBC-128 を推奨します。AES-GCM-128 は、2013 年 2 月の動作検証当時に使用したバージョンの OpenSSL 標準コマンドにバグがあり復号不可だったため、現状では未対応・未検証です。

例外発生時のハンドリング(セキュリティ攻撃やエラー検知時の通知・リトライ処理)は、現バージョンでは未実装です。

ホームネットワーク内など、中間に NAT が介在する環境に EA を配置する場合、そのままでは ALA-S からの Client Hello が通らないケースがあります。このような場合、STUN 等、NAT 越え方式との組み合わせなどを合わせて検討して下さい。

第2章　動作原理

2.1.　認証・暗号化の要素技術

本節では、ALA を構成する認証・暗号化の各要素技術を説明します。

2.1.1.　共通鍵暗号方式

共通鍵暗号方式とは、暗号化と復号化に同じ鍵を使用する方式の総称であり、対称鍵暗号方式、秘密鍵暗号方式と呼ばれる場合もあります。

共通鍵暗号方式では、暗号化されていない平文（ひらぶん：Plane Text）に鍵（Key）と暗号化アルゴリズムを適用して変換すると、暗号文（Cipher Text）になります。暗号文に鍵と復号化アルゴリズムを適用して変換すると、元の平文に戻ります。

共通鍵暗号方式の原理を図 2.1.1 に示します。

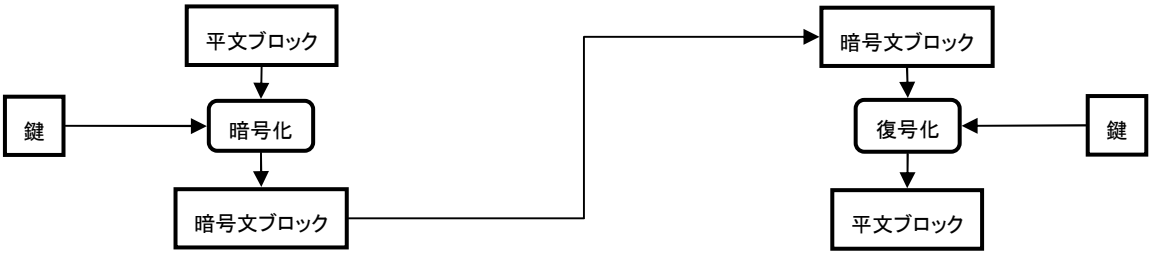

図 2.1.1 共通鍵暗号方式の原理

共通鍵暗号方式は、後述する公開鍵暗号方式に比較すると扱いが簡単で処理が高速な半面、相手先毎に固有の鍵を作成する必要があり管理が煩雑、かつ予め安全な方法で相手に鍵を渡さなければならない等の不利があり、今日では公開鍵暗号方式と組み合わせたハイブリッド方式が実用化され、広く普及しています。

共通鍵暗号方式は、ブロック暗号とストリーム暗号に大別されます。

ブロック暗号は、入力データを一定長のブロックに区切って暗号化する方式であり、一般的なデータの暗号化に広く使われています。代表的なブロック暗号を以下に示します。

DES (Data Encryption Standard)

DES は、1973 年に米国商務省標準局(NBS: National Bureau of Standard)がコンピュータデータの暗号化方式を公募し、IBM 社から提案された方式をもとに、1977 年に標準化(FIPS-77)された暗号化方式です。ブロック長 64 bit・鍵長 56bit であり、転置と換字の組合せ処理を 16 段行います。DES を 3 段に組み合わせ、暗号強度を強化したものが Triple DES です。

AES (Advanced Encryption Standard)

AES は、DES に代わる次世代の暗号標準技術として、アメリカ合衆国の標準化機関である NIST が選定・標準化した暗号化方式であり、連邦情報処理標準規格 FIPS-197 に選定されています。

ブロック・鍵長は 128/192/256bit から選択可能、鍵配送問題への対応としては、恒久的な事前共有、手作業による鍵交換、DH 鍵交換、公開鍵暗号による鍵交換の 4 方式から選択可能です。

暗号化アルゴリズムには、ベルギーの暗号学者 Joan Daemen と Vincent Rijmen によって提案された「Rijndael」 が標準方式として採用されています。

共通鍵暗号方式の事実上の標準として、最も広く普及している方式です。

IDEA (International Data Encryption Algorithm)

IDEA は、1991 年にチューリッヒ工科大学の Lai と Massey により提案された暗号化方式です。IDEA は、異なる代数群上の演算を組合せるコンセプトに基づき設計されています。

フリーの暗号化パッケージである PGP(Pretty Good Privacy)による使用が知られています。

RC5

RC5 は 1994 年に Rivest により発表された暗号化方式です。ブロック長(2w bit)・段数(r 段)・秘密鍵長(k byte)は可変であり、RC5-w/r/k の形式で指定します。

MISTY

MISTY は 1995 年に三菱電機により発表された暗号化方式です。差分解読法と線形解読法に対して安全であることが証明されており、ソフトウェア・ハードウェアの双方で高速な暗号として知られています。ブロック長 64 bit・鍵長 128bit であり、MISTY1 と MISTY2 の 2 種類があります。

Camellia

Camellia は 2000 年に NTT と三菱電機により発表された暗号化方式です。ブロック長 128bit、鍵長は 128/192/256bit の 3 通りです。

CLEFIA

CLEFIA は 2007 年にソニーにより発表された暗号化方式です。ブロック長 128bit、鍵長は 128/192/256bit の 3 通りです。軽量暗号(Lightweight Cryptography)の国際標準規格 ISO/IEC 29192 の一つとして採択され、スマートカードや RFID タグ、センサーネットワーク、医療機器などへの応用が見込まれています。

ストリーム暗号は、ビット単位またはバイト単位で逐次暗号化する方式であり、バッファリングやパディングが不要でリアルタイム処理に向いているため、音声通信のような連続的なデータの暗号化に用いられます。ストリーム暗号としては、SSL/TLS に RC4(RFC2246)がオプション扱いで採用されており、無線 LAN (WEP, WPA)などにも使用されています。

2.1.2. AES-GCM 方式

ブロック暗号を利用して、ブロック長よりも長いメッセージを暗号化するメカニズムの事を、暗号利用モード（Block cipher modes of operation）と呼びます。

最も単純な暗号利用モードである ECB モードでは、ある鍵で同一の平文を暗号化した場合、常に同一の暗号文が出力されます。このため、例えば長いメッセージのある部分が他の部分と同じかどうかが、暗号文の比較によって判断できてしまうという問題があります。このような問題を回避するために、ECB 以外にも様々な種類のモードが検討・標準化されています。

暗号利用モードには、秘匿目的のものと、認証目的のものがあります。

秘匿目的のモードとしては、ECB に CBC、OFB、CFB を加えた 4 つが、FIPS、ANSI、ISO、JIS で規格化されています。また、AES 制定時に追加された CTR も普及しています。

認証目的のモードとしては、CCM、GCM、OCB、XCBC などが普及しています。

本項では一例として、CBC よりも高効率・高性能、かつ並列処理が可能なモードである GCM (Galois/Counter Mode)を使用した AES-GCM 方式を紹介します。

AES-GCM 方式のパケット構造を図 2.1.2 (1)に示します。

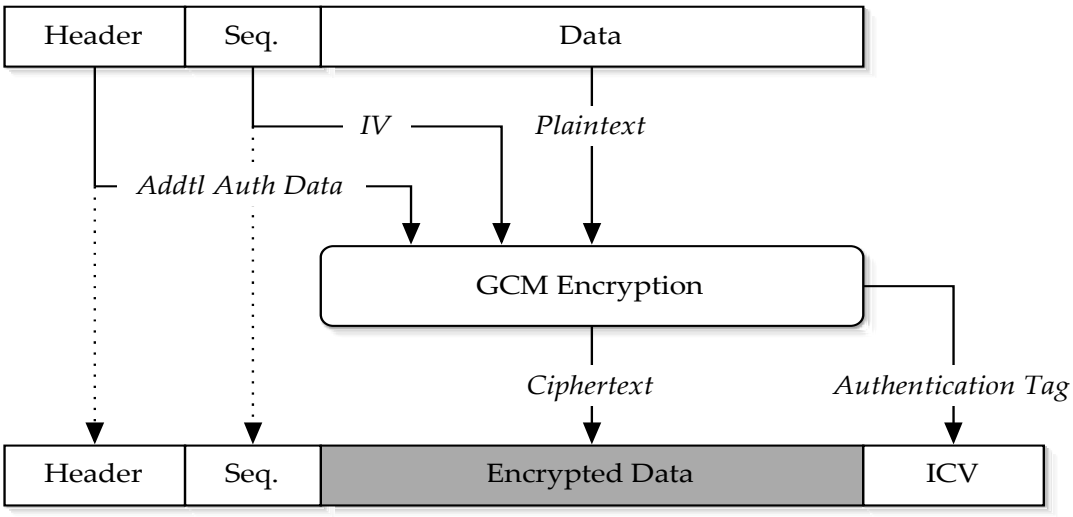

図 2.1.2 (1) AES-GCM 方式のパケット構造

AES-GCM 方式の入力は IV と平文と付加認証情報の 3 つ、出力は暗号文と認証タグ(MAC)の 2 つであり、送信内容の正真性チェックに認証タグ(MAC)を使用します。

送信側はシーケンス番号から IV を生成し、予め共有している鍵と生成した IV でデータ部の暗号化とパケット全体の検証タグを出力します。受信側はシーケンス番号から IV を生成し、予め共有している鍵と生成した IV でデータ部の復号と検証タグを用いてパケット全体を検証します。

パケット毎のシーケンス番号のインクリメント処理により、同一の鍵と IV の組み合わせの再利用を防ぐ事が可能です。

AES-GCM では、128 ビットのデータブロック単位でビット列を処理します。
この場合のカウンター初期値には、IV が使われます。
AES-GCM では、各データブロックを異なるカウンター値と共通鍵で処理します。

AES-GCM 方式の暗号処理フローを図 2.1.2 (2)に示します。

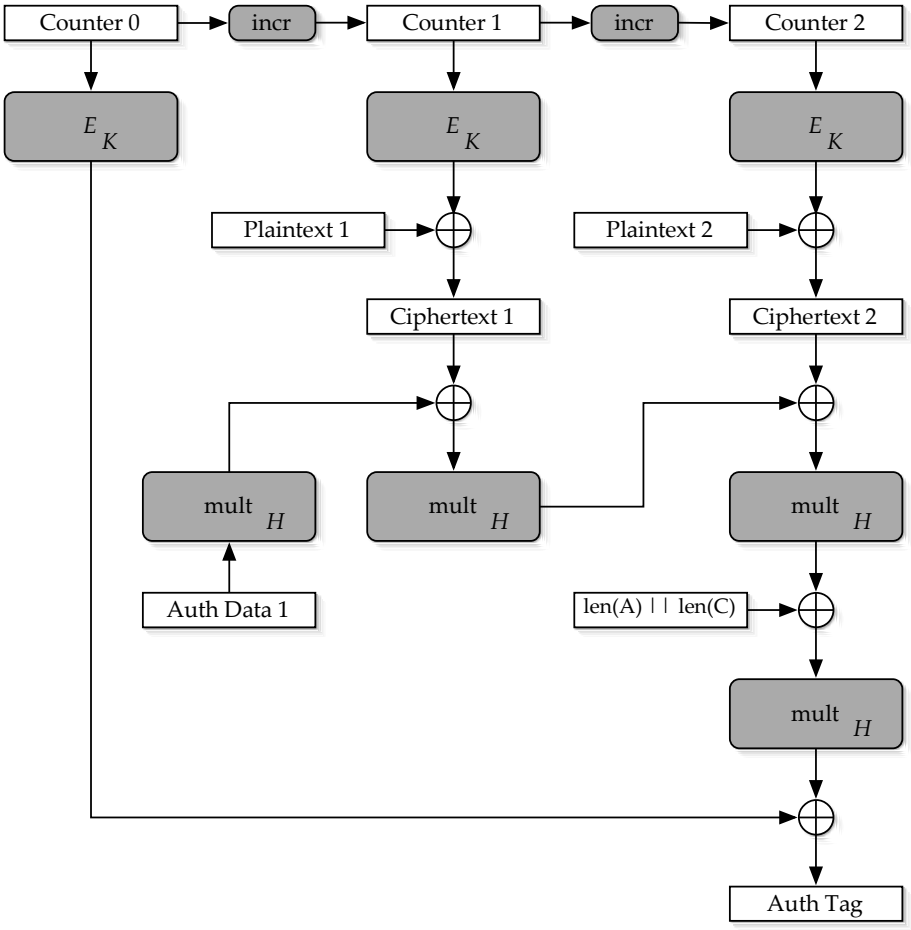

図 2.1.2 (2) AES-GCM 方式の暗号処理フロー

AES-GCM 方式の復号処理フローを図 2.1.2 (3)に示します。

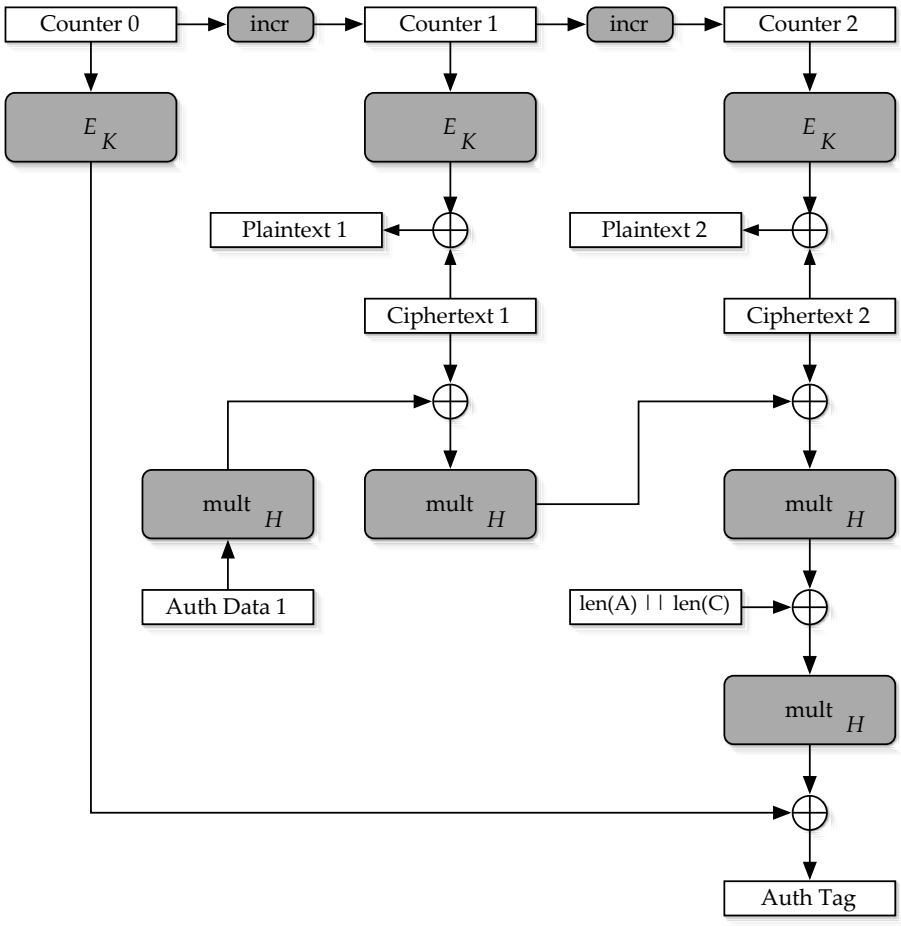

図 2.1.2 (3) AES-GCM 方式の暗号処理フロー

2.1.3. 公開鍵暗号方式

公開鍵暗号方式は、暗号化と復号化に異なる鍵を使用する非対称鍵暗号方式の一種であり、片方の鍵を相手に公開する事から「公開鍵暗号」と呼ばれています。また、暗号化と復号化が違う処理で行われるため、「非対称アルゴリズム」とも呼ばれています。
公開鍵暗号方式の手順は、以下の通りです。

(1) 受信者（送信先）が予め秘密鍵と公開鍵をペアで作成し、公開鍵を公開、秘密鍵を保管
(2) 送信者が受信者の公開鍵を入手、平文を暗号化し受信者に送信
(3) 受信者が秘密鍵により暗号文を復号化

公開鍵暗号方式の原理を図 2.1.3 に示します。

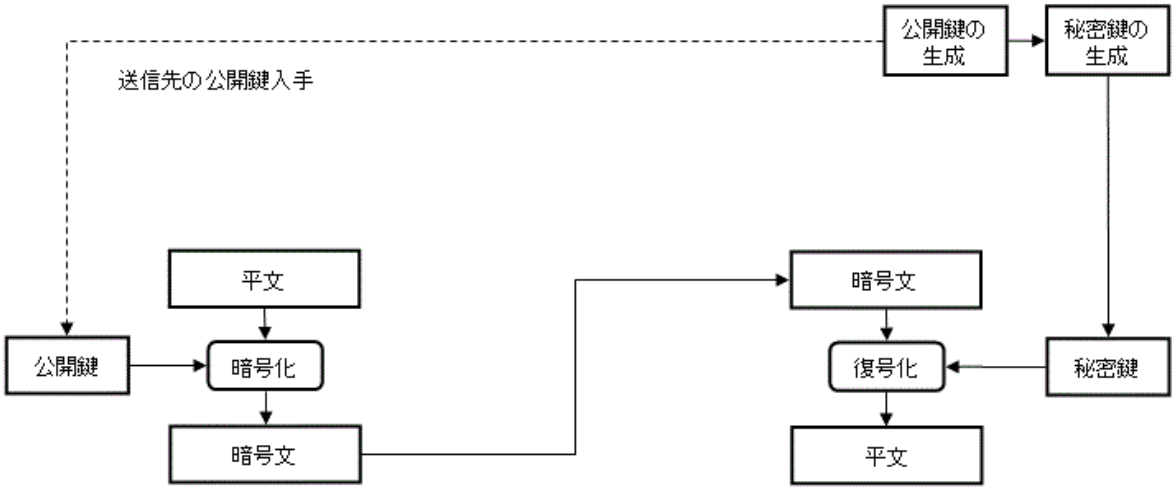

図 2.1.3 公開鍵暗号方式の原理

公開鍵暗号が有効に機能するには、以下の 2 つの条件の成立が前提となります。

- 一方の鍵から、他方の鍵を導き出すことはできない
- 一方の鍵で暗号化した暗号文は、他方の鍵でしか復号化できない

公開鍵暗号方式により、永い間暗号技術の重要問題であった安全な鍵配送を実現できます。また、一方の鍵を公開しておけば、誰にでもその鍵を使ってもらう事ができるため、多数の相手とのやり取りを行う場合でも自分の秘密鍵だけを管理すれば良く、鍵の管理もシンプルです。しかしながら、素数計算等の数学的な操作により高速処理が困難な事から、共通鍵暗号化方式と比較した場合、処理が概ね 3 桁以上重くなります。このため公開鍵暗号化方式は、共通鍵配送に限定した使用が一般的です。公開鍵方式による共通鍵生成時の通信オーバーヘッドにも注意が必要です。

公開鍵暗号方式としては、RSA が最も良く知られています。
RSA は、桁数が大きい合成数の素因数分解問題が困難である事を安全性の根拠とした公開鍵暗号の一つであり、暗号(Cipher)とデジタル署名(Digital signature)を実現できる公開鍵暗号アルゴリズムとして最初に公開されました。
RSA は 1977 年に発明され、発明者であるロナルド・リベスト（Ron Rivest）、アディ・シャミア（Adi Shamir）、レオナルド・エーデルマン（Len Adleman）の頭文字をつなげて命名されました。ディフィーとヘルマンによって発表された公開鍵暗号という新概念に対し、発明者 3 氏が秘匿や認証を実現できる具体的なアルゴリズムを与えたものです。

RSA 暗号では、鍵ペア（公開鍵と秘密鍵）を作成して公開鍵を公開します。
まず、適当な正整数 e（通常は小さな数。($65537 = 2^{16}+1$）がよく使わます）を選択します。
また、大きな 2 つの素数｛p ,q｝を生成し、それらの積 n（= pq）を求めて、｛e, n｝を平文の暗号化に使用する鍵（公開鍵）とします。
2 つの素数｛p ,q｝は、暗号文の復号に使用する鍵（秘密鍵）d の生成にも使用し（$d = e^{-1} \pmod{(p-1)(q-1)}$）、秘密に保管します。

- 暗号化（平文 m から暗号文 c を作成します）: $c = m^e \bmod n$
- 復号（暗号文 c から元の平文 m を得ます）: $m = c^d \bmod n$

ここで、暗号化（e 乗）は、｛e, n｝があれば容易に計算できるのに対して、復号（e 乗根）は、n の素因数を知らないと困難（大きい合成数の素因数分解が困難）であり、従って秘密鍵を用いずに暗号文から平文を得ることは困難である、という説明が成立します。
すなわち、数学の世界で素因数分解の高速アルゴリズムが未だに発見されていない事が、RSA 暗号の安全性の根拠です。

RSA 暗号のアルゴリズムは、1983 年 9 月 20 日にアメリカ合衆国で特許（4,405,829 号）を取得し、RSA Security 社がライセンスを独占していましたが、特許期間満了に伴い、2000 年 9 月 6 日からは誰でも自由に使用できるようになりました。
RSA で AES128bit と同水準の暗号強度を保つには､2048bit 以上の鍵長が必要とされています。

2.1.4. メッセージ認証コード

メッセージ認証コード（英: Message Authentication Code, MAC）は、メッセージを認証するための短い情報です。MAC アルゴリズムは、入力として共通鍵と認証すべき任意長のメッセージを受け取り、MAC（「タグ」とも呼ばれる）を出力します。同一の共通鍵を持つ検証者は、受信した MAC 値とメッセージから自身が計算した MAC 値の照合によりメッセージの改竄の有無を調べ、正真性を確認します。

メッセージ認証コードの原理を図 2.1.4 に示します。

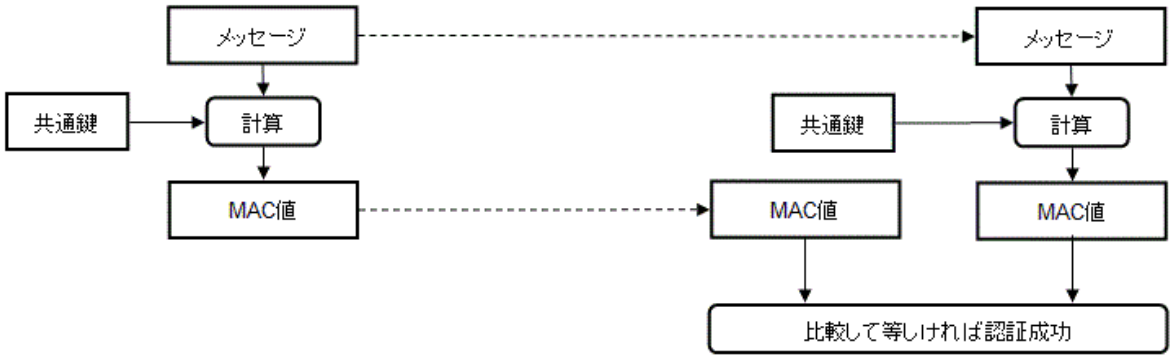

図 2.1.4 メッセージ認証コードの原理

MAC アルゴリズムは共通鍵暗号方式の応用技術であり、ハッシュ関数を使う方式（HMAC)、ブロック暗号アルゴリズムを使う方式（OMAC/CMAC、CBC-MAC、PMAC）などがあります。

SSL/TLS では、送信内容の正真性チェックに HMAC を使用しています。

HMAC は、生成アルゴリズムにハッシュ関数（SHA-1, MD-5 等）を使用した MAC です。

ハッシュ関数は、次の 4 つの特性を持つ一方向性関数です。

- 任意の桁数のデータから、一定の桁数のハッシュ値を出力可能
- 出力ハッシュ値から、元のデータを復元不可
- 元データのごく一部だけが変更された場合でも、出力ハッシュ値が大きく変化
- 違う元データから同じハッシュ値が出力される可能性が極めて稀

AES-GCM では、暗号文の正真性チェックに MAC の一種である認証タグを使用しています。

AES-GCM の場合、認証タグ長は通常、GCM 鍵長と同一です。

2.1.5. デジタル署名

デジタル署名は公開鍵暗号方式の応用技術であり、第三者認証と否認防止に有効です。

デジタル署名は、公開鍵による暗号化の逆方向処理により動作します。

デジタル署名の原理を図 2.1.5 に示します。

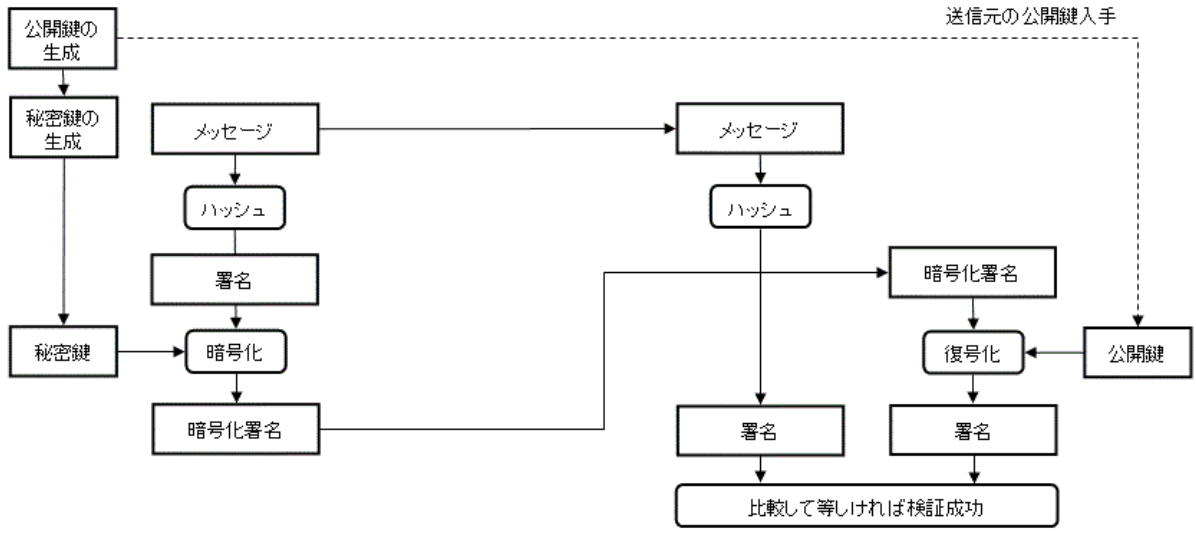

図 2.1.5 デジタル署名の原理

前項で説明したメッセージ認証コードでは、MAC 値の生成と検証に同じ鍵を使用するため（共通鍵暗号）、送信者と受信者は通信に先立ち、予め鍵を共有しておく必要があります。また、共通鍵暗号であるために、MAC によって認証されるメッセージは、偽造ではなく送信者が作成し送信したという確証、つまり否認不可性を持ちません。なぜなら、MAC 値を検証する利用者（メッセージの受信側）は、通信の相手から受け取ったメッセージではなく、受信側で捏造したメッセージについても共通鍵を使って MAC 値を生成できるからです。

公開鍵暗号を用いたデジタル署名では、メッセージの検証を公開鍵だけで行えるため、鍵の所有者はデジタル署名を作成できる秘密鍵を秘匿できます。このためデジタル署名が付与された文書は、その所有者が署名したものと確定でき、否認不可な文書を作成できます。

デジタル署名は、SSL/TLS におけるクライアント・サーバー証明書の署名技術として普及しています。署名長には通常、128bit～512bit の範囲が使われます。

2.1.6. PKI の概要

PKI (Public Key Infrastructure: 公開鍵基盤)は、公開鍵を効果的に運用するために定められた規格・仕様の総称です。PKI の主な構成要素は利用者・認証局(CA)・リポジトリの 3 つであり、基盤技術に公開鍵暗号方式（およびデジタル署名）を使用します。

PKI の原理を図 2.1.6 に示します。

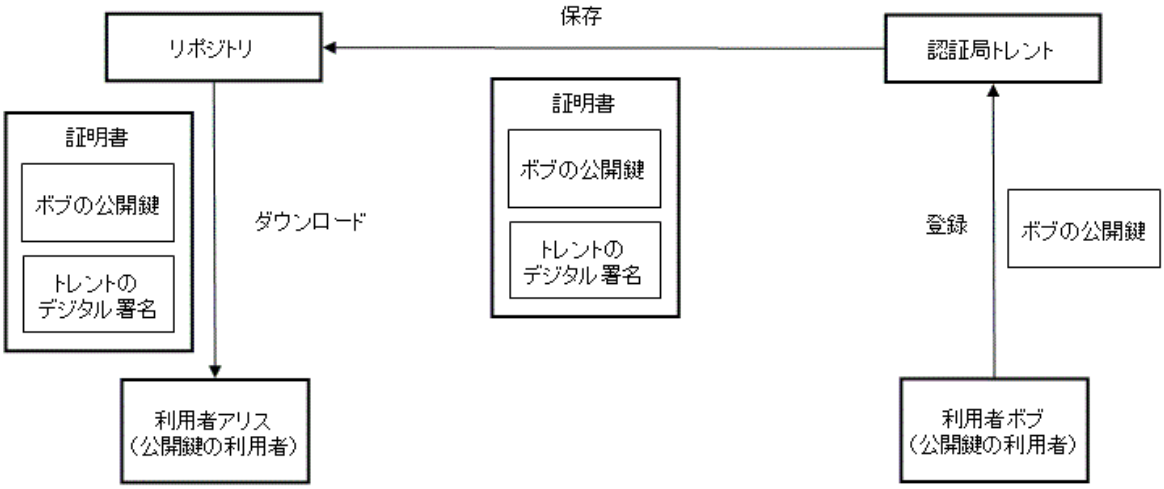

図 2.1.6 PKI の原理

ある利用者ボブが自分の秘密鍵を用いて通信文にデジタル署名を行ったとき、別の利用者アリスは、認証局トレントによって発行された利用者ボブの公開鍵証明書に収められた公開鍵を用いてボブの署名を確認できます。この仕組みによって、通信を行う人同士が事前に何らかの機密情報をやり取りする必要なく、機密性、完全性、相手の認証が行えます。

PKI のより詳細な情報については、IPA による PKI 関連技術の解説を参照して下さい。
http://www.jpa.go.jp/security/pki/

多くの企業の PKI システムは、企業のディレクトリ体系と密接な関連があり、各従業員の公開鍵が各々の個人情報（電話番号、E メールアドレス、住所、部署など）と合わせて管理されます。
主なディレクトリ技術として LDAP が普及していますが、現在最も一般的な証明書形式である X.509 は、LDAP の前身である X.500 ディレクトリスキーマに由来しています。

2.1.7. SSL/TLS の概要

SSL/TLS (Secure Socket Layer / Transport Layer Security: 以下「TLS」)は、対称暗号・メッセージ認証コード・公開鍵暗号・デジタル署名を組み合わせた暗号通信フレームワークです。
TLS は、下層の「TLS レコードプロトコル」と上層の「TLS ハンドシェイクプロトコル」により構成されます。
上層の TLS ハンドシェイクプロトコルは、以下の 4 つのサブプロトコルにより構成されます。

- ハンドシェイクプロトコル
- 暗号仕様変更プロトコル
- 警告プロトコル
- アプリケーションデータプロトコル

下層の TLS レコードプロトコルは、メッセージのフラグメント分割&圧縮、MAC 付加、ブロック暗号化、ヘッダ付加を実行します。
TLS は実効性・有用性が市場で実証された合理的なフレームワークですが、スマートメーターに応用する場合、以下の各事項への対策が必要となる点に注意が必要です。

- ブロック暗号化モードの不一致
  TLS1.0/1.1 のブロック暗号化モードは CBC ですが、例えば事業者のセキュリティポリシーにより AES-GCM への対応が必要な場合、両者が不一致となるため、何らかの対処が必要になります。2013 年 1 月現在、デファクトリファレンスの Open SSL では GCM への対応予定がない、すなわち、GCM は市場に未普及のため、メーター通信に GCM を組み込む場合、プラットフォーム別の GCM への個別対応が必要になります。

- 通信セッション毎の共通鍵生成による通信オーバーヘッド

- ターンアラウンド時間の間延び、C ルート時の通信コスト増大

TLS 標準から若干逸脱しますが、共通鍵の都度更新方式として、通信の発生しないカウンター加算&再ハッシュ方式も有効な選択肢の一つです。

TLS セッションシーケンスを図 2.1.7 に示します。

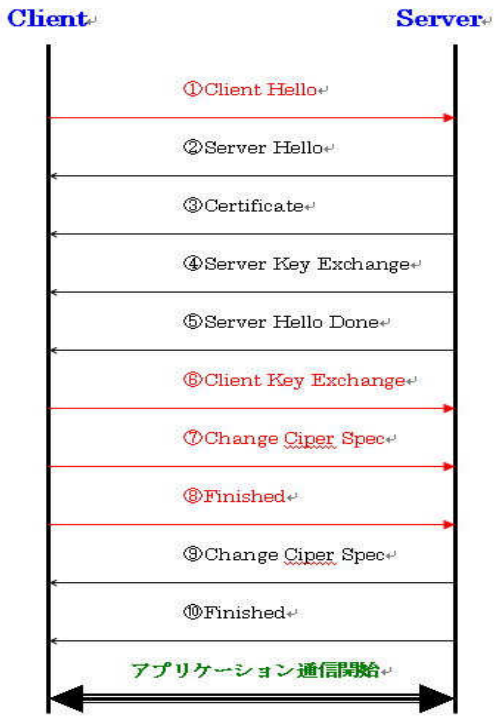

図 2.1.7 TLS セッションシーケンス

① ClientHello

クライアント（ブラウザなど）が利用可能な暗号化/圧縮アルゴリズムの一覧をサーバーに送信します。

② ServerHello

クライアントから送られてきた一覧から暗号化/圧縮アルゴリズムを決定しクライアントに通知します。

③ Certificate

サーバーのデジタル証明書をルート CA までの証明書リストを含めてクライアントに送信します。

④ ServerKeyExchange

③でデジタル証明書を送信しない場合、一時的に RSA 鍵か Diffie-Hellman 鍵を生成してクライアントに送信します。

⑤ ServerHelloDone

サーバー側の一連の処理の終了をクライアントに通知します。

⑥　ClientKeyExchange

セッション鍵の元となる情報（プリマスターシークレット）をクライアントが生成し、デジタル証明書に含まれるサーバーの公開鍵で暗号化しサーバーへ送信します。

⑦　ChangeChiperSpec

今までの情報から、ハンドシェイクプロトコルがマスターシークレットを生成し、レコードプロトコルがハンドシェイクプロトコルから受け取ったマスターシークレットからMAC鍵､暗号鍵、初期ベクトル(IV）を生成し、Change Chipher Spec プロトコルが暗号化アルゴリズムの準備完了をサーバーに通知します。

⑧　Finished

鍵交換と認証処理の完了をサーバーに通知します。

⑨　ChangeChiperSpec

サーバーが､クライアントから受信したプリプライマリーシークレットを自身の秘密鍵で複合化し､クライアント同様にマスターシークレット生成した後､MAC 鍵､暗号鍵､初期ベクトル（IV）を生成します。最後に ChangeChiperSpec が暗号化準備完了をクライアントに通知します。

⑩　Finished

鍵交換と認証処理が成功したことをクライアントに通知します。

2.2. OpenSSL の概要

OpenSSL は、SSL プロトコル・TLS プロトコルのオープンソースな実装です。

OpenSSL の中核部分は、Eric A. Young と Tim Hudson による SSLeay（1998 年 12 月に開発者が RSA Security に異動したため開発は既に終了）を基にしています。

OpenSSL の中核部分のライブラリ（C 言語で書かれている）は、基本的な暗号化関数と様々なユーティリティ関数を実装しており、複数の言語で OpenSSL ライブラリを利用できるようにするラッパーも存在します。

OpenSSL が利用可能なプラットフォームは、Unix 系（Solaris、Linux、Mac OS X、BSD 等）、OpenVMS、Windows です。

OpenSSL は以下の暗号化アルゴリズムをサポートしています。

- 暗号方式
  Blowfish、Camellia、DES、RC2、RC4、RC5、SEED、IDEA、AES
- ハッシュ関数方式
  MD5、MD2、SHA-1、SHA-2、MDC-2（英語版）
- 公開鍵暗号方式
  RSA 暗号、DSA、Diffie-Hellman 鍵共有

OpenSSL の詳細については、OpenSSL の公式サイトを参照して下さい。

ALA-C のプライベート CA 局機能と、ALA-S の共通鍵作成・更新セッション機能は、OpenSSL により実装されています。

2.3. EA の概要

EA の暗号化オプション機能は、JCE(Java Cryptography Extension)により実装されています。
JCE は、Java 言語で認証・暗号化関連の処理を開発する場合に使われる暗号化拡張機能です。
Java による暗号化には、Java のコア API である Security API と Java 暗号化拡張機能(JCE)が必要になります。JCE は Java 2 SDK v1.2.x、1.3.x では拡張機能でしたが、Java 2 SDK 1.4 以降では JCE が SDK に統合され、 標準 JCE プロバイダが予めインストール、登録されています。
JCE の詳細については、JCE の関連文書を参照して下さい。
EA の詳細については、「OPEN LIB EA エンジニアリングガイド」を参照して下さい。

第3章　データベース

3.1.　共通鍵管理フォルダ

ALA-C が作成した共通鍵情報は、ALA-S の指定フォルダに格納されます。
ALA のデフォルト（推奨）フォルダは以下の通りです。

ALA-S の共通鍵: /etc/pki/AA
スマートメーターの共通鍵: /var/pki/SM
VPA の共通鍵: /var/pki/VPA

3.2.　データベースオブジェクト

ALA 関連テーブルは、MDM の MDM-DB に格納されます。
MDM-DB の ALA 関連テーブル一覧を表 3.2(1)に示します。

№	テーブル名	レコード単位	想定最大 レコード数	格納する情報
1	CERTIFICATE_HISTORY	X.509 証明書の発行	1,200,000	X.509 証明書の発行履歴。

表 3.2(1)　MDM-DB の ALA 関連テーブル一覧

テーブルの項目定義の詳細は、「OPEN EMS MDM エンジニアリングガイド」を参照して下さい。

第4章　設計・インストール

4.1.　設計

4.1.1.　　X.509 証明書の仕様設計

ALA のデフォルトは以下の通りです。

a) 公開鍵暗号アルゴリズム：RSA
b) 公開鍵長：2048 bit
c) X.509 バージョン：v1
d) 署名アルゴリズム：sha1 with RSA Encryption
e) 署名アルゴリズム鍵長：2048 bit

ネットワーク構成やセキュリティ要件に応じ、適宜設定を変更して下さい。
公開鍵暗号アルゴリズム、署名アルゴリズムについては、第 2 章を参照して下さい。
X.509 バージョンで、v1 を選択すると非互換性の問題がある場合、v3 の選択も可能です。
公開鍵長・署名アルゴリズム鍵長は、2013 年 12 月現在、既に 1024bit が破られ、2048bit が主流となっていますが、長すぎる鍵長の選択は端末側の実行時間の間延びを招くので、必要十分な長さの設定を心掛けて下さい。

4.1.2.　　RSA 鍵ペアファイル形式

ALA デフォルトは、「.pem」(テキスト)形式です。
但し、DER(バイナリ)形式での運用も可能です。

4.2. インストール

4.2.1. ルート CA サーバー

(1) OS インストール

本項では、CentOS(RHEL 互換 OS) v6.3 を前提に説明します。なお、インストール時、OS パッケージを”Desktop”(デフォルトは”Minimal”となっています)を選択する以外は全てデフォルト設定とします。また、ファイアウォールは停止するか、適宜通信が通るように設定して下さい。

(2) OPEN SSL インストール

以下のコマンドにより OPEN SSL をインストールします。

#> yum install openssl

※2013/02/14 時点では、v1.0.1c がインストールされます。

次に OPEN SSL の動作設定ファイル(etc/pki/tls/openssl.cnf)を以下の通り編集します。

```
.
.
.
# This sets a mask for permitted string types. There are several options.
# default: PrintableString, T61String, BMPString.
# pkix   : PrintableString, BMPString (PKIX recommendation before 2004)
# utf8only: only UTF8Strings (PKIX recommendation after 2004).
# nombstr : PrintableString, T61String (no BMPStrings or UTF8Strings).
# MASK:XXXX a literal mask value.
# WARNING: ancient versions of Netscape crash on BMPStrings or UTF8Strings.
string_mask = nombstr
.
.
.
```

nombstr に修正。※デフォルトは utf8only

図 4(1) 「etc/pki/tls/openssl.cnf」の編集例

なお、上記変更を行わない場合、他のサーバーで発行した証明書等への署名時に以下のエラーが発生する場合があるため注意して下さい。

The organizationName field needed to be the same in the
CA certificate (Kanagawa) and the request (Kanagawa)

※文字コードの関係で同じ Kanagawa でも不一致と判定されてしまいます。

(3) ルート CA の登録

以下の手順で設定を行います。

```
# cd /etc/pki/tls/misc
# ./CA -newca

Making CA certificate ...
Generating a 2048 bit RSA private key
...............+++
...................+++
writing new private key to '/etc/pki/CA/private/./cakey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:JP
State or Province Name (full name) []:Kanagawa
Locality Name (eg, city) [Default City]:
Organization Name (eg, company) [Default Company Ltd]:OSS BroadNet Co., Ltd.
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:sm-ca.ossbn.co.jp
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
Using configuration from /etc/pki/tls/openssl.cnf
Enter pass phrase for /etc/pki/CA/private/./cakey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number:
            a5:f4:4a:be:ab:e6:48:e9
        Validity
            Not Before: Jan 28 05:39:03 2013 GMT
            Not After : Jan 28 05:39:03 2016 GMT
        Subject:
            countryName               = JP
            stateOrProvinceName       = Kanagawa
            organizationName          = OSS BroadNet Co., Ltd.
            commonName                = sm-ca.ossbn.co.jp
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                C6:B6:5D:39:31:BB:58:37:68:59:A1:78:OF:OD:65:42:E3:1E:2O:A5
            X509v3 Authority Key Identifier:
                keyid:C6:B6:5D:39:31:BB:58:37:68:59:A1:78:OF:OD:65:42:E3:1E:2O:A5

            X509v3 Basic Constraints:
                CA:TRUE
Certificate is to be certified until Jan 28 05:39:03 2016 GMT (1095 days)

Write out database with 1 new entries
Data Base Updated
```

OPEN SSL の標準で用意されているシェルスクリプトを実行。

秘密鍵のパスワードを入力。
今後、署名等で使用するためメモする事。

組織情報を適宜入力。尚、"Country Name"、"State or Province Name"、及び、"Organization Name"については、今後作成する他の証明書と同じである必要があり、かつ、"Common Name"は今後作成する証明書を含めユニークである必要があるため注意すること。他の項目は無指定で可。

図 4(2) ルート CA 設定手順

設定の完了時、以下のディレクトリに証明書及び秘密鍵が保管されます。

・/etc/pki/CA/cacert.pem … CA の自己署名証明書

・/etc/pki/CA/private/cakey.pem … CA の秘密鍵

CA の秘密鍵は外部に漏れないよう、厳重に注意して管理して下さい。

(4) ファイル署名用シェルスクリプトの登録

任意の場所に専用シェルスクリプト(filesign.sh)を保管します。

※chmod コマンド等で予め実行権限を与えておいて下さい。

(5) イメージファイルの署名

イメージファイルをルート CA の任意のディレクトリにコピーします。

次に署名用シェルスクリプトで署名ファイルを作成します。
署名ファイル作成手順は以下の通りです。

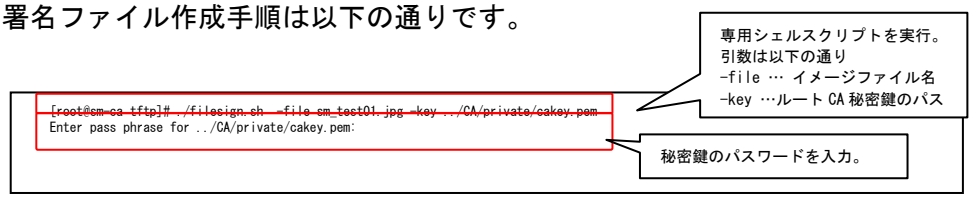

図 4(3) 署名ファイル作成手順

上記で作成した署名ファイルとイメージファイルを同梱した ZIP ファイルを作成し、
TFTP ディレクトリ(/var/lib/tftpboot)に配置します。
ZIP 圧縮コマンドは以下の通りです。

```
#> zip file.zip sm_test01.jpg
#> zip file.zip sm_test01.sig
```

イメージファイルを ZIP 圧縮

次に、署名ファイルを追加。

4.2.2. ALA サーバー

(1) OS インストール

4.2.1(1)と同じ手順を実行します。

(2) OPEN SSL インストール

SSL 通信に OPEN SSL 標準コマンドを利用するため OpenSSL をインストールします。インストール方法は、4.1(2)と同じ手順です。なお、設定ファイル等はデフォルトのまま使用します。

(3) ALA 証明書の保存先ディレクトリ(/etc/pki/AA)を以下のコマンドで作成します。

#> mkdir /etc/pki/AA

(4) ALA 証明書の作成＆登録

ルート CA で以下の手順で ALA 証明書を作成します。

```
[root@sm-ca ~]# mkdir /etc/pki/AA
[root@sm-ca ]# cd /etc/pki/AA
[root@sm-ca AA]# openssl genrsa -des3 -out aa.key 2048
Generating RSA private key, 2048 bit long modulus
.....................................................................+++
....+++
e is 65537 (0x10001)
Enter pass phrase for aa.key:
Verifying - Enter pass phrase for aa.key:

[root@sm-ca AA]# openssl req -new -key aa.key -out aa.csr
Enter pass phrase for aa.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:JP
State or Province Name (full name) []:Kanagawa
Locality Name (eg, city) [Default City]:
Organization Name (eg, company) [Default Company Ltd]:OSS BroadNet Co., Ltd.
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:aa01.ossbn.co.jp
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

[root@sm-ca AA]# openssl ca -in aa.csr -out aa.crt
Using configuration from /etc/pki/tls/openssl.cnf
Enter pass phrase for /etc/pki/CA/private/cakey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number:
            a5:f4:4a:be:ab:e6:48:ea
        Validity
            Not Before: Jan 29 05:04:57 2013 GMT
            Not After : Jan 29 05:04:57 2014 GMT
        Subject:
            countryName               = JP
            stateOrProvinceName       = Kanagawa
            organizationName          = OSS BroadNet Co., Ltd.
            commonName                = aa01.ossbn.co.jp
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                72:12:3F:6D:2E:F8:B7:5C:F2:AE:11:C1:BE:33:C5:0E:E2:04:95:C8
            X509v3 Authority Key Identifier:
                keyid:C6:B6:5D:39:31:BB:58:37:68:59:A1:78:0F:0D:65:42:E3:1E:20:A5

Certificate is to be certified until Jan 29 05:04:57 2014 GMT (365 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

ALA 証明書の一時保管ディレクトリを作

ALA の秘密鍵を生成。鍵長は 2048。最後にパスワードを入力。

上記で作成した ALA 秘密鍵から、証明書＆公開鍵を作成。この際上記で指定した秘密鍵のパスワードを入力。

組織情報を適宜入力。尚、”countryName”、”State or Province Name”、及び、”rganization Name”については、ルート CA に合わせる必要あり、かつ、”Common Name”は今後作成する証明書を含めユニークである必要があるため注意すること。他の項目は無指定で可。

上記で作成した ALA 証明書にルート CA の署名を追加。この際上記で指定したルート CA 秘密鍵とそのパスワードを入力。

図 4(4) ALA 証明書発行手順

作成した ALA 証明書(aa.crt)、秘密鍵(aa.key)、公開鍵(aa.csr)を ALA 証明書が保存されているディレクトリ(/etc/pki/AA)配下にコピーします。

(5) ALA(サンプル)シェルスクリプトの登録

所定ディレクトリ(/etc/pki/AA)に専用シェルスクリプト(tls.sh、aa.sh)を保管します。

今回は/etc/pki/AA 配下にコピーします。chmod コマンド等で予め実行権限を付加しておいて下さい。シェルスクリプトの詳細は以下の通りです。

・tls.sh　… OPEN SSL の標準コマンドを使用し、スマートメーターと TLS 通信で AES 共通鍵の交換を行います。
引数はスマートメーターの IP アドレスのみです。
本シェルスクリプトだけでも ALA セッションの一連の処理は実行されますが、標準出力で大量の情報が標準出力されるため、
通常は直接実行せずに、aa.sh を介して実行します。

・aa.sh　… ALA セッションの実行と動作ログを出力します。

第5章　運用・保守

5.1.　証明書の作成

ルート CA で、以下の手順で SM 証明書を作成します。

基本的には、第 4 章に示した ALA 証明書の作成手順と同じ流れです。

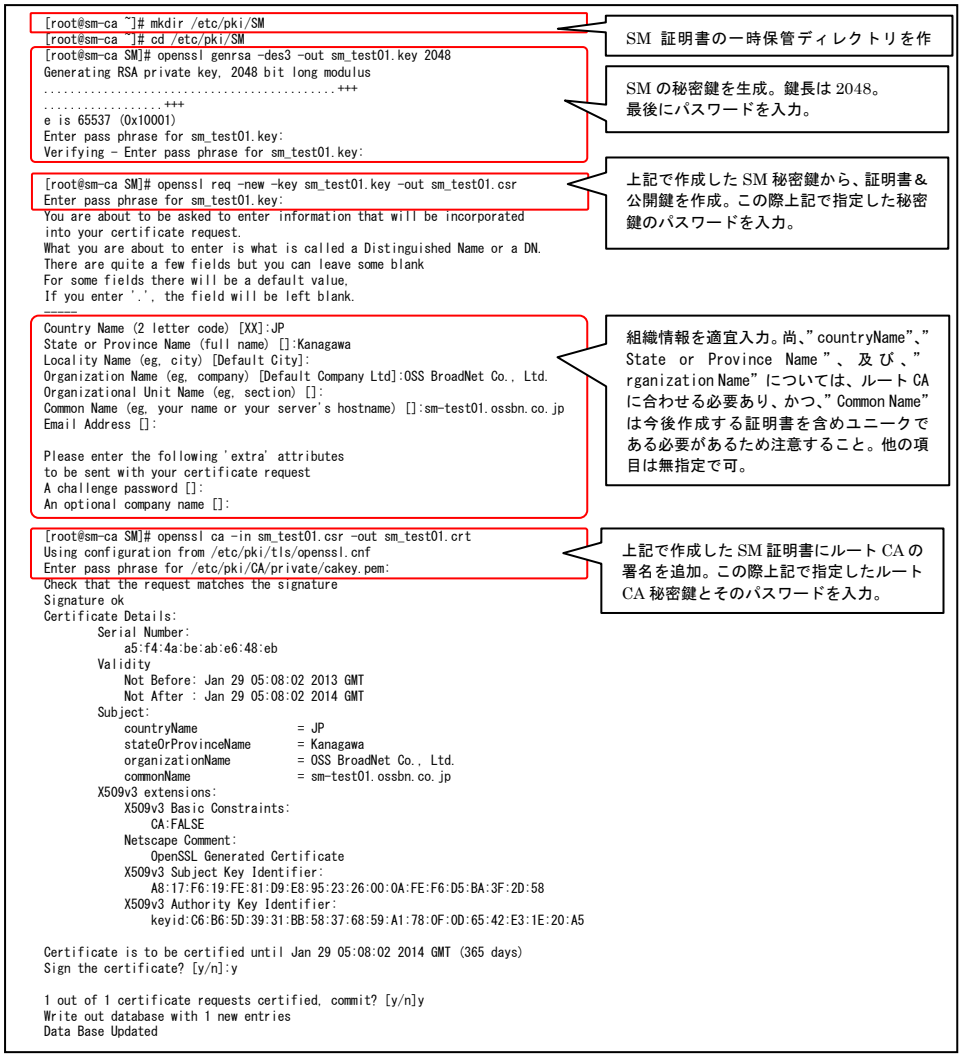

図 5.1 SM 証明書発行手順

作成した SM 証明書(sm_test01.crt)、秘密鍵(sm_test01.key)、公開鍵(sm_test01.csr)を、スマートメーター等の端末機器に組み込みます。

付録A　スマートメーターシミュレーター

1. スマートメーターシミュレーターの概要

スマートメーターシミュレーターは、OSSBN が NTT データと共同開発した DLMS/COSEM スマートメーター用アプリケーション開発用の動作検証プログラムであり、.Net framework 上で動作します。

2. セットアップ

スマートメーターシミュレーターで証明書を扱う場合、Windows に前もって証明書情報を登録しておく必要があります。方法は以下の通りです。

・ルート CA 証明書の登録

証明書マネージャ(certmgr.msc)より、ルート CA 証明書(cacert.pem)を選択しインポートします。

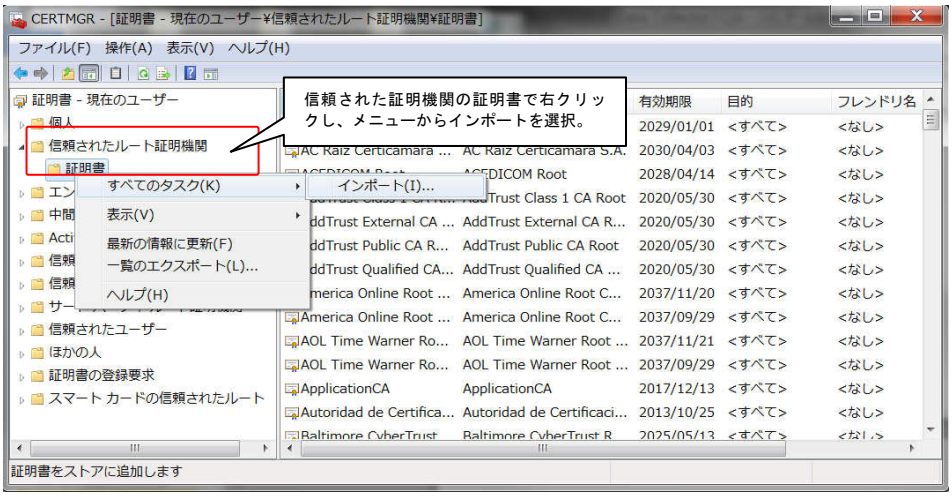

図 5.2(1)　ルート CA 証明書の登録画面

また、イメージファイルの署名確認用に、ルート CA 証明書(cacert.pem)から CA 公開鍵(pubkey.pem)を取り出し、スマートメーターシミュレータインストール先配下の tftp ディレクトリ内にコピーします。尚、CA 公開鍵は以下のコマンドで取り出すことが出来ます。

```
#> openssl x509 -in cacert.pem -pubkey > pubkey.pem
```

・

SM 証明書の登録

SM 証明書は、SM 秘密鍵を同梱した PKCS#12 形式に変更してからインポートする必要があります。以下のコマンドを実行します。

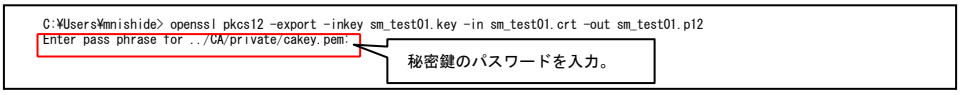

図 5.2(2) PKCS#12 形式への変換

※SM 秘密鍵(sm_test01.key)と SM 証明書(sm_test01.crt)のパスは適宜追加する事。

次に、ルート CA 証明書と同様に証明書マネージャ(certmgr.msc)より、上記で変換した PKCS#12 形式の SM 証明書を選択しインポートします。

図 5.2(3) PKCS#12 形式 SM 証明書の登録画面

3. 留意事項

スマートメーターシミュレーターは、DLMS/COSEM 以外のプロトコルには非対応です。

OPEN LIB ALA エンジニアリングガイド

2014 年 5 月 28 日 第 0.5 版 発行

著　　者　　宮副　英治、西出　誠、中武　正文
発　　行　　オーエスエスブロードネット株式会社
〒 213-0011 神奈川県川崎市高津区久本 3-5-7 新溝ノ口ビル 5F
電子メール: info@ossbn.co.jp
