(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6301244号
(P6301244)
(45)発行日　平成30年3月28日(2018.3.28)　(24)登録日　平成30年3月9日(2018.3.9)

(51)Int.Cl.		F I		
H04L 9/32	(2006.01)	H04L	9/00	675A
H04L 9/08	(2006.01)	H04L	9/00	601B
G06F 21/44	(2013.01)	G06F	21/44	350

請求項の数 18　外国語出願　(全 17 頁)

(21)出願番号　特願2014-253636(P2014-253636)
(22)出願日　平成26年12月16日(2014.12.16)
(65)公開番号　特開2015-146567(P2015-146567A)
(43)公開日　平成27年8月13日(2015.8.13)
審査請求日　平成29年12月14日(2017.12.14)
(31)優先権主張番号　377/MUM/2014
(32)優先日　平成26年2月3日(2014.2.3)
(33)優先権主張国　インド(IN)

早期審査対象出願

(73)特許権者　510337621
タタ　コンサルタンシー　サービシズ　リミテッド
TATA Consultancy Services Limited
インド国　マハーラシュトラ、ムンバイ　400021、ナリマン　ポイント、ナーマル　ビルディング　9階
Nirmal Building, 9th Floor, Nariman Point, Mumbai 400021, Maharashtra, India.
(74)代理人　100083286
弁理士　三浦　邦夫

最終頁に続く

(54)【発明の名称】モノのインターネットのためのデータグラム転送における軽量認証のためのコンピュータ実施システムおよび方法

(57)【特許請求の範囲】
【請求項１】
サーバーとクライアント間のデータグラム転送に関する双方向の相互認証のためのコンピュータにより実施される、システムプロセッサを含むシステムであって、
前記システムプロセッサと協働し、第１の乱数を発生する第１の乱数発生器；
前記システムプロセッサと協働し、第２の乱数を発生する第２の乱数発生器；
前記システムプロセッサと協働し、プロビジョニング処理の間の双方向認証の前に、システムプロセッサの生成コマンド及び送信コマンドを受けて秘密鍵を生成してサーバー及びクライアントに送信する秘密鍵生成器；
前記システムプロセッサの転送コマンドを受けて、クライアントの固有ＩＤを含む第１のメッセージをクライアントからサーバーに送信するセッションイニシエータ；
前記システムプロセッサと協働して、システムプロセッサの受信コマンドを受けて前記第１のメッセージを受信し、受信したクライアントＩＤと予め格納されたクライアントを識別するためのクライアントＩＤを照合するマッチングエンジンを搭載する受信機；
固有時間制限付きセッション鍵を生成し、前記システムプロセッサの送信コマンドを受けて生成したセッション鍵を転送するセッション鍵生成器であって、前記システムプロセッサからのコマンドに応じて動作するセッション鍵タイマーを有し、該セッション鍵タイマー値の満了により、生成したセッション鍵を無効にし、新しいセッションの確立の要件を示すセッション鍵生成器；
前記システムプロセッサと協働して、前記セッション鍵を受け取り、第１の乱数発生器

によって発生された第１の乱数と前記セッション鍵生成器によって生成された前記セッション鍵を含むチャレンジコードを、システムプロセッサの発生コマンドを受けて生成し、プロセッサの送信コマンドを受けて送信するチャレンジコード生成器；
前記チャレンジコード生成器と協働して、システムプロセッサからのコマンドに応じて、前記秘密鍵生成器により秘密鍵とともに生成されたチャレンジコードを受信し、該生成されたチャレンジコードを受信して暗号化し、さらにシステムプロセッサからの送信コマンドを受けて、前記暗号化されたチャレンジコードを特定クライアントに送信する第１の暗号器；
前記システムプロセッサと協働して、前記暗号化されたチャレンジコードを受信し、さらに、システムプロセッサからのコマンドに応じて、秘密鍵生成器によって生成された秘密鍵で暗号化されたチャレンジコードを復号化し、復号化された第１の乱数及びセッション鍵を得る第１の復号器；
前記第１の復号器から受信したセッション鍵を格納するリポジトリ；
前記システムプロセッサと協働し、前記復号化された第１の乱数とセッション鍵を受信し、さらにシステムプロセッサからの送信コマンドを受けて、復号化された第１の乱数と、第２の乱数発生器によって生成され、セッション鍵で暗号化された第２の乱数を含む第２のメッセージを送信する第２暗号器；
前記システムプロセッサと協働し、前記第２のメッセージを受信し、さらにシステムプロセサからのコマンドに応じて、前記第１の乱数と第２の乱数を、セッション鍵生成器によって生成されたセッション鍵を用いて復号化する第２の復号器；
前記システムプロセッサからのコマンドに応じて、前記第２のメッセージから復号化された第１の乱数と前記第１の乱数発生器で生成された第１の乱数を比較してクライアントを認証する第１のコンパレータと認証器；
クライアントを認証した後、前記システムプロセッサからのコマンドに応じて、受信したセッション鍵生成器によって生成されたセッション鍵が含まれる第２のメッセージの中の第２の乱数を暗号化し、システムプロセッサの送信コマンドを受けて、暗号化された第２の乱数を送信する第３の暗号器；
前記システムプロセッサからの受信コマンドに応じて受信し、さらにシステムプロセッサからのコマンドに応じて、リポジトリから受信したセッション鍵で暗号化された第２の乱数を復号化する第３の復号器；及び
前記システムプロセッサからのコマンドに応じて、前記復号化した第２の乱数と前記第２の乱数発生器によって発生された第２の乱数を比較し、サーバーの認証と相互認証を達成する第２のコンパレータと認証器；を含み、
前記第１の乱数発生器で生成された第１の乱数は、第１のタイマー値を付加した第１の疑似乱数であること、
を特徴とするシステム。

【請求項２】
請求項１記載のシステムにおいて、前記第１の乱数発生器は、前記システムプロセッサからのコマンドに応じて前記第１のタイマー値を生成する第１のタイマーを有するシステム。

【請求項３】
請求項１記載のシステムにおいて、前記第２の乱数発生器は、前記システムプロセッサからのコマンドに応じて第２のタイマー値を生成する第２のタイマーを有するシステム。

【請求項４】
請求項１記載のシステムにおいて、前記第２の乱数発生器によって発生された第２の乱数は、前記第２のタイマー値を付加した第２の擬似乱数であるシステム。

【請求項５】
請求項１記載のシステムにおいて、前記秘密鍵生成器によって生成された秘密鍵は、前記セッションの開始時に生成された固有の鍵であって、進行中のセッションの間のみ有効であるシステム。

【請求項６】
請求項１記載のシステムにおいて、前記第１の乱数発生器及び第２の乱数発生器によって発生された乱数は、再現不能であり、異なるセッションによって変化するシステム。
【請求項７】
請求項１記載のシステムにおいて、前記クライアントは、前記サーバーがクライアントの実行要求のステータスに応答しないように前記サーバーと通信するシステム。
【請求項８】
請求項１記載のシステムにおいて、前記システムは、データグラムトランスポート層セキュリティ（ＤＴＬＳ）を含むトランスポート層セキュリティ方式と統合されているシステム。
【請求項９】
請求項８記載のシステムにおいて、前記システムは、制約付きデバイスのための制約付きアプリケーションプロトコル（ＣｏＡＰ）を含むアプリケーション層プロトコルを統合していて、セッションの確立は、前記データグラムトランスポート層セキュリティ（ＤＴＬＳ）におけるセッション確立のオーバーヘッドを軽減するために前記制約付きアプリケーションプロトコル（ＣｏＡＰ）に埋め込まれているシステム。
【請求項１０】
請求項１記載のシステムにおいて、セッションをリフレッシュする鍵リフレッシュタイマーを含み、該鍵リフレッシュタイマーは、鍵リフレッシュタイマー値が最大転送カウント（MAX_RETRANSMIT_COUNT）と再送のタイムアウト時間(MAX_RETRANSMISSION_TIMEOUT)との積より大きくなったとき、各セッションをリフレッシュするシステム。
【請求項１１】
サーバーとクライアント間のデータグラム転送に関する双方向の相互認証のためのコンピュータにより実施される、システム処理コマンドを含む方法であって、
第１の乱数発生器を利用して、第１の乱数を発生するステップ；
第２の乱数発生器を利用して、第２の乱数を発生するステップ；
秘密鍵生成器を利用して、システム処理コマンドに応じて秘密鍵を生成するステップ；
システム処理コマンドに応じて、双方向認証の前であって、プロビジョニング処理の間に、前記生成された秘密鍵をサーバーとクライアントに送信するステップ；
システム処理コマンドに応じて、クライアントの固有ＩＤを含む第１のメッセージを送信するステップ；
システム処理コマンドに応じて、前記第１のメッセージを受信し、受信したクライアントＩＤを予め格納されているクライアントＩＤと照合するステップ；
前記受信したクライアントＩＤに基づいてクライアントを識別するステップ；
セッション鍵生成器を利用して、セッション鍵タイマー値の満了に基づいてセッション鍵を無効にし、満了の上で新しいセッションの確立の要求を示す固有時間制限付きセッション鍵を生成するステップ；
セッション鍵を受信し、システム処理コマンドを受けて、第１の乱数発生器によって生成された第１の乱数と、前記セッション鍵生成器により生成された前記セッション鍵を含むチャレンジコードを生成するステップ；
前記チャレンジコード生成器により生成されたチャレンジコードを受信し、システム処理コマンドに応じて、前記第１の暗号化部を利用して前記秘密鍵生成器により生成された秘密鍵で前記受信したチャレンジコードを暗号化し、暗号化したチャレンジコードをシステム処理コマンドに応じて送信するステップと；
前記暗号化されたチャレンジコードを受信し、システム処理コマンドに応じて、前記第１の復号化器を利用して、前記秘密鍵生成器により生成された秘密鍵で暗号化されたチャレンジコードを復号化し、第１の乱数とセッション鍵を取得するステップ；
前記第１の復号器からセッション鍵を受信し、リポジトリに格納するステップ；
システムの処理コマンドに応じて、前記復号化された第１の乱数とセッション鍵を受信し、該復号化された第１の乱数及び前記第２の乱数発生器により発生された第２の乱数を

含むセッション鍵で暗号化された第２のメッセージを送信するステップ；
システム処理コマンドに応じて、第２のメッセージを受信し、第１の乱数と第２の乱数を、前記セッション鍵生成器により生成されたセッション鍵を用いて復号化するステップ；
システムの処理コマンドに応じて、第２のメッセージからの復号化された第１の乱数と第１の乱数発生器によって発生された第１の乱数を比較するステップ；
システム処理コマンドを受けて、前記復号化された第１の乱数が前記第１の乱数発生器によって発生された第１の乱数と一致したとき、前記クライアントを認証するステップ；
クライアントを認証した後、システムの処理コマンドに応じて、前記セッション鍵発生器によって生成されたセッション鍵とともに受信した第２のメッセージ中の第２の乱数を暗号化し、暗号化された第２の乱数を送信するステップ；
システム処理コマンドに応じて受信し、リポジトリから受信したセッション鍵で暗号化された第２の乱数を復号化するステップ；
システム処理コマンドに応じて、前記復号化された第２の乱数と前記第２の乱数発生器によって発生された第２の乱数を比較するステップ；及び
システム処理コマンドに応じて、前記復号化された第２の乱数と前記第２の乱数発生器によって発生された第２の乱数が一致するとき、相互認証を確立するためにサーバーを認証するステップ；を含み、
前記第１の乱数発生器によって発生された第１の乱数は、第１のタイマー値を付加した第１の疑似乱数であること、
を特徴とする方法。
【請求項１２】
請求項１１に記載の方法において、前記第２の乱数を発生するステップは、システム処理コマンドに応じて、第２の乱数を発生するために、第２のタイマー値の生成と該第２のタイマー値を第２の擬似乱数に付加するステップを含む方法。
【請求項１３】
請求項１１に記載の方法において、前記秘密鍵を生成するステップは、進行中のセッションの間のみ有効な、セッションの開始時に固有鍵の生成を含む方法。
【請求項１４】
請求項１１に記載の方法において、乱数を発生するステップは、システムの処理コマンドに応じて生成する数値を含み、該数値は再現不能であって異なるセッションによって変化する方法。
【請求項１５】
請求項１１に記載の方法において、クライアントは、サーバーがクライアントの実行要求のステータスに応答しないようにサーバーと通信できる方法。
【請求項１６】
請求項１１に記載の方法は、データグラムトランスポート層セキュリティ（ＤＴＬＳ）を含むトランスポート層セキュリティ方式と統合されている方法。
【請求項１７】
請求項１６に記載の方法は、制約付きデバイスのための制約付きアプリケーションプロトコル（ＣｏＡＰ）を含むアプリケーション層プロトコルと統合されていて、セッションの確立は、前記データグラムトランスポート層セキュリティ（ＤＴＬＳ）におけるセッション確立のオーバーヘッドを軽減するために前記制約付きアプリケーションプロトコル（ＣｏＡＰ）に埋め込まれている方法。
【請求項１８】
請求項１１に記載の方法は、前記鍵リフレッシュタイマーが最大転送カウント（MAX_RETRANSMIT_COUNT）と再送タイムアウト時間（MAX_RETRANSMISSION_TIMEOUT）との積より大きくなったとき、鍵リフレッシュタイマーによって各セッションをリフレッシュするステップを含む方法。
【発明の詳細な説明】

【技術分野】
【０００１】
本発明は、モノのインターネットのための認証とセキュリティに関するものである。
【背景技術】
【０００２】
本明細書において以下使用される「IoT」という表現は、モノのインターネットであって、インターネットのような構造で一意に識別可能なものを指す。
本明細書において以下使用される「M2M」という表現は、無線と有線の両方のシステムにより異種のノードを含むネットワークを介してお互いの間で通信することを可能にするマシンツーマシン通信技術を指す。
本明細書において以下使用される「ノンス（nonce)」という表現は、一度だけ使用される乱数を指す。
本明細書において以下使用される「データグラム転送」という表現は、コネクションレスのトランスポートプロトコルであって、例示的で一般的な機能は、ユーザーデータグラムプロトコル（UDP）である。
本明細書において以下使用される「プロビジョニング処理」という表現は、通信前にクライアント側とサーバー側における準備をするプロセスを指す。これは、事前共有秘密を埋め込むようなステップを含む。
本明細書において以下使用される「セッションイニシエータ」という表現は、サーバーに最初の「HELLO」メッセージを送信してセッションを開始する装置を指す。これらの定義は、当該技術分野において表現されたものに追加される。
【０００３】
IoT / M2Mは、物理的な実体を含み；アイデンティティまたは状態が、インターネットインフラを介してやりとりできる。 M2Mは、IoTの部分集合と考えることができる。M2Mにより動作するIoTの上に転写されるデータのパターンは、データトラフィックモデル及び参加ノード数の点で従来のインターネットとは異なる。M2Mは、従来のインターネット上での人対人（H2H）系の繋がりよりもはるかに多くのノードを扱っている。
【０００４】
IoT / M2Mシステムは、通常、無線および／または有線ネットワークを介して通信が許可されたセンサのような制約付きデバイスで構成されている。この無線通信ネットワークは、通常、帯域幅の観点から制約される。そのような制約下においてドメイン認証付きの、堅牢で低オーバーヘッドの安全な通信手段を配備することが課題である。従来の公開鍵暗号方式を利用した、制約付きデバイス用の強固な証明書ベースのスキームは、処理とエネルギーと帯域幅条件にコストがかかりすぎる。さらに、もしIPレイヤにおけるセキュリティ、例えばIPSecを考慮すれば、リソースの使用およびメンテナンスの点で最適とはいえない。また、TLSのようなトランスポート層セキュリティ方式は、非常に堅牢であるにもかかわらず、制約付きデバイスのためにコストがかかることがあるので、そのリソース要件には適用されない。
【０００５】
制約付きアプリケーションプロトコル（CoAP）は、インターネット上で制約付きデバイス間の双方向のコミュニケーションをRESTful（REST原則に基づいた）アーキテクチャーによって可能にする典型的なネットワークアプリケーション層プロトコルである。CoAPは、インターネットエンジニアリングタスクフォース（IETF）から主に軽量なソリューションを作成するためにユーザデータグラムプロトコル（UDP）上で動作するように設計されたIoT/ M2Mのためのセキュリティ層解法としてデータグラムトランスポート層セキュリティ（DTLS）を提案している。しかし、本格的な証明書ベースの公開鍵基盤（PKI）用のDTLSは、制約付きデバイスには最適ではない。そこで、DTLSの事前共有鍵（PSK）モードは、制約付きデバイス用の軽量代替物として定義される。このような方式は、軽量であるが堅牢性を犠牲にする。また、エンドポイントの認証を欠いている。
【０００６】

CoAPでは、DTLSは、攻撃者によって増幅攻撃を起動するために送信されたClientHelloメッセージによるサービス拒否（DoS）攻撃を軽減するためにクッキー交換技術を使用している。特に、PSKモードでは、クライアントは、事前共有鍵からプレマスターシークレットとマスターシークレットを生成し、サーバーで必須の事前共有鍵を調べるためにサーバーによって使用されるクライアント鍵交換（ClientKeyExchange）メッセージを、psk_identityを含むサーバーに送信する。しかし、プレーンテキストによるクッキー交換は堅牢ではない。また、クッキー交換メカニズムは、接続確立のためのオーバーヘッドが限られた環境のため高くつくことが分かっている。

【０００７】

したがって、IoT/ M2Mの制約下で使用可能な、認証された軽量でありながら堅牢なシステムにおいて明確なホワイトスペースがあることは明らかである。さらに、また、一般的なネットワーキング／通信システムの認証要件に対応できるシステムと方法が必要とされている。

【発明の概要】

【発明が解決しようとする課題】

【０００８】

本発明の目的は、典型的な制約付きのIoT / M2M環境でエンドポイントを相互に認証する軽量堅牢なシステムを提供することにある。

本発明のさらに別の目的は、ハンドシェイクメッセージの少ない数を使用するという点でセキュリティ方式の軽量事前共有シークレットモードを使用するシステムを提供することにある。

本発明のさらに別の目的は、一般的なネットワーキング／通信システムの認証要求に応えるために、一般的なシステムを提供することにある。

本発明のシステムのもう1つの目的は、対称暗号化を使用して、クライアントによって開始ペイロードが埋め込まれた認証及び鍵管理を可能にするシステムを提供することである。

さらに、本発明のさらに別の目的は、DTLSのようなトランスポート層セキュリティ方式と統合することができるシステムを提供し、既存DTLSスキームを強化し、交換の数を減らすことにより、より軽量化することである。

本発明のさらなる目的は、新しいヘッダオプションが付加された、制約付きデバイスのために、CoAPのようなアプリケーション層と統合することができるシステムを提供することである。

【０００９】

本発明の範囲を限定するものではない添付の図面と併せて読めば、他の目的および本発明の利点は以下の説明からより明らかになるであろう。

【課題を解決するための手段】

【００１０】

本発明は、サーバーとクライアント間のデータグラム転送（移送）の２つの相互認証のためのコンピュータにより実施されるシステムを想定している。

【００１１】

本発明は、サーバーとクライアント間のデータグラム転送に関する双方向の相互認証のためのコンピュータにより実施される、システムプロセッサを含むシステムであって、前記システムプロセッサと協働し、第１の乱数を発生する第１の乱数発生器；前記システムプロセッサと協働し、第２の乱数を発生する第２の乱数発生器；前記システムプロセッサと協働し、プロビジョニング処理の間の双方向認証の前に、システムプロセッサの生成コマンド及び送信コマンドを受けて秘密鍵を生成してサーバー及びクライアントに送信する秘密鍵生成器；前記システムプロセッサの転送コマンドを受けて、クライアントの固有ＩＤを含む第１のメッセージをクライアントからサーバーに送信するセッションイニシエータ；前記システムプロセッサと協働して、システムプロセッサの受信コマンドを受けて前記第１のメッセージを受信し、受信したクライアントＩＤと予め格納されたクライアント

を識別するためのクライアントＩＤを照合するマッチングエンジンを搭載する受信機；固有時間制限付きセッション鍵を生成し、前記システムプロセッサの送信コマンドを受けて生成したセッション鍵を転送するセッション鍵生成器であって、前記システムプロセッサからのコマンドに応じて動作するセッション鍵タイマーを有し、該セッション鍵タイマー値の満了により、生成したセッション鍵を無効にし、新しいセッションの確立の要件を示すセッション鍵生成器；前記システムプロセッサと協働して、前記セッション鍵を受け取り、第１の乱数発生器によって発生された第１の乱数と前記セッション鍵生成器によって生成された前記セッション鍵を含むチャレンジコードを、システムプロセッサの発生コマンドを受けて生成し、プロセッサの送信コマンドを受けて送信するチャレンジコード生成器；前記チャレンジコード生成器と協働して、システムプロセッサからのコマンドに応じて、前記秘密鍵生成器により秘密鍵とともに生成されたチャレンジコードを受信し、該生成されたチャレンジコードを受信して暗号化し、さらにシステムプロセッサからの送信コマンドを受けて、前記暗号化されたチャレンジコードを特定クライアントに送信する第１の暗号器；前記システムプロセッサと協働して、前記暗号化されたチャレンジコードを受信し、さらに、システムプロセッサからのコマンドに応じて、秘密鍵生成器によって生成された秘密鍵で暗号化されたチャレンジコードを復号化し、復号化された第１の乱数及びセッション鍵を得る第１の復号器；前記第１の復号器から受信したセッション鍵を格納するリポジトリ；前記システムプロセッサと協働し、前記復号化された第１の乱数とセッション鍵を受信し、さらにシステムプロセッサからの送信コマンドを受けて、復号化された第１の乱数と、第２の乱数発生器によって生成され、セッション鍵で暗号化された第２の乱数を含む第２のメッセージを送信する第２暗号器；前記システムプロセッサと協働し、前記第２のメッセージを受信し、さらにシステムプロセサからのコマンドに応じて、前記第１の乱数と第２の乱数を、セッション鍵生成器によって生成されたセッション鍵を用いて復号化する第２の復号器；前記システムプロセッサからのコマンドに応じて、前記第２のメッセージから復号化された第１の乱数と前記第１の乱数発生器で生成された第１の乱数を比較してクライアントを認証する第１のコンパレータと認証器；クライアントを認証した後、前記システムプロセッサからのコマンドに応じて、受信したセッション鍵生成器によって生成されたセッション鍵が含まれる第２のメッセージの中の第２の乱数を暗号化し、システムプロセッサの送信コマンドを受けて、暗号化された第２の乱数を送信する第３の暗号器；前記システムプロセッサからの受信コマンドに応じて受信し、さらにシステムプロセッサからのコマンドに応じて、リポジトリから受信したセッション鍵で暗号化された第２の乱数を復号化する第３の復号器；及び前記システムプロセッサからのコマンドに応じて、前記復号化した第２の乱数と前記第２の乱数発生器によって発生された第２の乱数を比較し、サーバーの認証と相互認証を達成する第２のコンパレータと認証器；を含み、前記第１の乱数発生器で生成された第１の乱数は、第１のタイマー値を付加した第１の疑似乱数であること、を特徴とする。

【００１２】

本発明の方法は、サーバーとクライアント間のデータグラム転送に関する双方向の相互認証のためのコンピュータにより実施される、システム処理コマンドを含む方法であって、第１の乱数発生器を利用して、第１の乱数を発生するステップ；第２の乱数発生器を利用して、第２の乱数を発生するステップ；秘密鍵生成器を利用して、システム処理コマンドに応じて秘密鍵を生成するステップ；システム処理コマンドに応じて、双方向認証の前であって、プロビジョニング処理の間に、前記生成された秘密鍵をサーバーとクライアントに送信するステップ；システム処理コマンドに応じて、クライアントの固有ＩＤを含む第１のメッセージを送信するステップ；システム処理コマンドに応じて、前記第１のメッセージを受信し、受信したクライアントＩＤを予め格納されているクライアントＩＤと照合するステップ；前記受信したクライアントＩＤに基づいてクライアントを識別するステップ；セッション鍵生成器を利用して、セッション鍵タイマー値の満了に基づいてセッション鍵を無効にし、満了の上で新しいセッションの確立の要求を示す固有時間制限付きセッション鍵を生成するステップ；セッション鍵を受信し、システム処理コマンドを受けて

、第１の乱数発生器によって生成された第１の乱数と、前記セッション鍵生成器により生成された前記セッション鍵を含むチャレンジコードを生成するステップ；前記チャレンジコード生成器により生成されたチャレンジコードを受信し、システム処理コマンドに応じて、前記第１の暗号化部を利用して前記秘密鍵生成器により生成された秘密鍵で前記受信したチャレンジコードを暗号化し、暗号化したチャレンジコードをシステム処理コマンドに応じて送信するステップと；前記暗号化されたチャレンジコードを受信し、システム処理コマンドに応じて、前記第１の復号化器を利用して、前記秘密鍵生成器により生成された秘密鍵で暗号化されたチャレンジコードを復号化し、第１の乱数とセッション鍵を取得するステップ；前記第１の復号器からセッション鍵を受信し、リポジトリに格納するステップ；システムの処理コマンドに応じて、前記復号化された第１の乱数とセッション鍵を受信し、該復号化された第１の乱数及び前記第２の乱数発生器により発生された第２の乱数を含むセッション鍵で暗号化された第２のメッセージを送信するステップ；システム処理コマンドに応じて、第２のメッセージを受信し、第１の乱数と第２の乱数を、前記セッション鍵生成器により生成されたセッション鍵を用いて復号化するステップ；システムの処理コマンドに応じて、第２のメッセージからの復号化された第１の乱数と第１の乱数発生器によって発生された第１の乱数を比較するステップ；システム処理コマンドを受けて、前記復号化された第１の乱数が前記第１の乱数発生器によって発生された第１の乱数と一致したとき、前記クライアントを認証するステップ；クライアントを認証した後、システムの処理コマンドに応じて、前記セッション鍵発生器によって生成されたセッション鍵とともに受信した第２のメッセージ中の第２の乱数を暗号化し、暗号化された第２の乱数を送信するステップ；システム処理コマンドに応じて受信し、リポジトリから受信したセッション鍵で暗号化された第２の乱数を復号化するステップ；システム処理コマンドに応じて、前記復号化された第２の乱数と前記第２の乱数発生器によって発生された第２の乱数を比較するステップ；及びシステム処理コマンドに応じて、前記復号化された第２の乱数と前記第２の乱数発生器によって発生された第２の乱数が一致するとき、相互認証を確立するためにサーバーを認証するステップ；を含み、前記第１の乱数発生器によって発生された第１の乱数は、第１のタイマー値を付加した第１の疑似乱数であること、を特徴とする。

【発明の効果】

【００１３】

本発明によれば、制約付きのIoT / M2M環境においてエンドポイントを相互に認証する軽量堅牢なシステムと方法を提供することができる。

【図面の簡単な説明】

【００１４】

本発明のシステムについて、添付の図面を参照して説明する。

【図１】サーバーとクライアント間の相互認証を提供するシステムの概略を示す図である。

【図２】相互認証と安全な通信を実現するためのシステムフローを示す図である。

【図３】サーバーとクライアント間のハンドシェイク中に含まれる手順を示す図である。

【図４】本発明の実施形態として、既存のアプリケーション層プロトコルに埋め込むためにCoAPメッセージ形式の導入ヘッダーオプションを示す図である。

【図５】センサデバイス（クライアント）とサーバー間の典型的な認証ハンドシェイクを示す図である。

【図６】DTLSのようなセキュリティ層と認証のための追加の層として、本発明のシステムの統合を示す図である。

【図７】事前共有鍵モード（PSK）での安全なセッション開始のための通常のDTLSハンドシェイクのタイミングを示す図である。

【図８】本発明に係る事前共有秘密により修正されたDTLSハンドシェイクの様子を示す図である。

【発明を実施するための形態】

【００１５】
本発明のシステムを、添付図面に示した実施形態に基づいて説明する。この実施形態は、本発明の範囲と領域を限定するものではない。説明は単に例を示すものであり、また本発明の好ましい実施形態及び推奨する応用を示すためのものである。
【００１６】
本明細書において、システム、さまざまな特徴及びその利点の詳細を以下のいくつかの実施形態によって説明するが、これらは本発明の範囲を限定するものではない。実施形態の説明を明確にするため、既知の構成部品や処理技術については省略する。ここで取り上げる各例は、その実行と当分野の技術者が各例を実施するための理解を容易にする目的でのみ示されるものである。よって、これらの例は実施形態の範囲を限定するものと解釈すべきではない。
【００１７】
本発明によれば、システムは、事前共有秘密を共有する二つのエンドポイント間におけるチャレンジ－レスポンス交換に基づいて、軽量かつ強固な認証方式を提供する。本発明が提案するセキュリティソリューションは、鍵管理が認証と統合された、対称鍵に基づくセキュリティメカニズムである。これは、サーバーとクライアント間のデータグラムの転送上の双方向認証を提供するIoT / M2Mに適している。
【００１８】
本発明のシステムは、低オーバーヘッドの相互認証を提供する。相互認証を達成するために、システム内のエンドポイントは、プロビジョニング処理中に事前共有鍵を使用して事前設定（provisioned）され、クライアントのデータベースは、クライアントを識別するためのサーバー側に設けられている。システムはまた、擬似乱数（PRN）モジュール及びナンス(nonces)を生成するためのタイマー（システム時間）、およびサーバー鍵生成モジュールを含む。ナンスと鍵は、安全な認証を提供する助けになる。チャレンジメッセージは、認証プロセス中にサーバーとクライアント側の両方から生成される。 AES暗号化および復号（解読）化は、クライアント側とサーバー側で使用される。
【００１９】
本発明のシステムは、さらに、PSKモードを使用してDTLSのようなトランスポート層セキュリティプロトコルに適合させることができる。適合させるための手順は、DTLSの上に暗号化されたノンスに基づいたチャレンジレスポンスを使用して認証セッションを確立し、使用するための安全なチャネルを確立することを含む。
【００２０】
添付の図面を参照すると、図1は、システムプロセッサ１０２のコマンドに基づいて、サーバー５０とクライアント２０との間で相互認証をするシステム１００の概略図である。本発明のシステム１００は、事前共有秘密を共有する２つのエンドポイント間のチャレンジ-レスポンス交換に基づく認証スキームを提案する。この事前共有秘密は、システムプロセッサ１０２からの生成コマンドに基づいて秘密鍵生成器１０によって生成される。本発明のセキュリティソリューションは鍵管理が認証と統合された、対称鍵に基づくセキュリティメカニズムである。プロビジョニング処理の間、エンドポイントは、事前共有秘密と共に構成されている。クライアント２０のセッションイニシエータ２２は、サーバー５０からクライアントに固有の識別子（ＩＤ）を使用してHELLOメッセージを送信して、セッションを開始する。サーバー５０の受信機５４は、システムプロセッサ１０２からのコマンドを受けてメッセージを受信し、最初にすべてのクライアントＩＤを格納したリポジトリ５２内から前記ＩＤを検索する。
しかし、悪意のあるクライアントによるスプーフィングを防ぐために、サーバー５０は、チャレンジコードジェネレータ６０を利用してチャレンジコードを生成する。このチャレンジコードは、セッション鍵生成部５８によって生成された固有のセッション鍵「ｋ」と第１の乱数発生器５６によって発生された乱数「nonce1」からなる。セッション鍵生成部５８は、生成するセッション鍵の有効性を判断するために適切なセッション鍵タイマー値を生成するセッション鍵タイマー（図示せず）を有する。従って、セッション鍵「ｋ」

は、セッション鍵タイマー値に基づいて無効にされる。セッション鍵タイマー値の満了は、その鍵の失効と新しいセッション鍵を使用して新しいセッションを確立するための必要性を示す。

第1の乱数発生部５６は、第１のタイマー値を生成する第１のタイマー（図示せず）を有する。第1の乱数発生器５６によって発生される「nonce1」は、この第１のタイマー値を付加した擬似乱数（PRN）である。そしてチャレンジコードは、秘密鍵生成器１０によって生成及び共有された事前共有鍵を使用して第１の暗号化部６２によって暗号化される。このチャレンジコードはクライアントに送信される。正当なクライアント２０は、チャレンジコードを、秘密鍵生成器１０によって共有された秘密鍵を利用して第１の復号化部２４を介して復号化し、さらに「nonce1」およびサーバー５０によって供給されるセッション鍵「ｋ」を取得できる。復号化されたセッション鍵「k」は、第２のレポジトリ２６に格納される。さらに、クライアント２０は、チャレンジコードに応じて、サーバー５０から受信した「nonce1」とクライアント２０において第２の乱数発生器３０が発生した「nonce2」を含む応答メッセージを生成する。第２の乱数発生部３０は、第２のタイマー値を生成する第２のタイマー（図示せず）を有する。この第２のタイマー値は、「nonce2」を形成するために、別の疑似乱数（PRN）に付加される。応答メッセージは、第１の復号器２４によって先に解読されたセッション鍵「ｋ」を用いて第２の暗号化部２８によって暗号化される。応答メッセージを受信すると、サーバー５０の第２の復号器６４は、クライアント２０からの応答を解読し、その「nonce1」が、サーバー５０が所有する（第１の乱数発生器５６からの）複製である「nonce1」と一致するか否か第１のコンパレータと認証器６６を用いて照合する。それらが一致したとき、サーバー５０は、第１のコンパレータ及び認証器６６によりそのクライアント２０を認証し、さらに、システムプロセッサ１０２からのコマンドに基づいてクライアント２０に、セッション鍵「ｋ」と第３の暗号化器６８を利用して暗号化された受信「nonce2」を含むメッセージを送信する。クライアントは、「nonce2」を、第３の復号化器３２を利用して第２のレポジトリ２６に格納されたセッション鍵「ｋ」を使用して解読する。復号化された「nonce2」は、第２のコンパレータ及び認証器３４を利用して第２の乱数発生器３０によって生成された「nonce2」と一致する。復号化された「nonce2」がクライアント２０の「nonce2」と一致したとき、サーバー５０は第２のコンパレータ及び認証器６６により認証され、相互認証が実現する。

【００２１】

ナンスと鍵は、異なるセッションを切り替えるために使用される。生成されたナンスは、タイマー（カウンタ）からのタイマー値が付加された擬似乱数（PRN）からなるので、再現不能である。これは、反射攻撃に対する耐性を提供する。

【００２２】

添付の図面を参照すると、図２は、相互認証と安全な通信を実現するためのシステムフローを示す図である。サーバーとクライアントは、プロビジョニング処理中に事前共有秘密（ｙ）で設定されている２つのエンドポイントである。プロビジョニング処理が完了した後、クライアントは、認証要求をサーバーに送信する（２００）。このセッションは、固有クライアントIDに従って「HELLO」メッセージをサーバーに送信することにより開始される。サーバーは、このメッセージを受信した後、事前構成済データベース内のクライアントIDを検索する。しかし、サーバーは、固有鍵（k）およびランダムノンス（nonce1）を含むチャレンジコードを生成し（２０２）、悪意あるクライアントのなりすまし（スプーフィング）を防止する。

チャレンジコードは、事前共有秘密（y）により暗号化され、クライアントに送信される。正当なクライアントは、チャレンジコードを事前共有秘密（Y）により復号化し（２０４）、サーバーから供給されたナンスと鍵を取得することができる。応じて、クライアントは、サーバーから受信したノンス（nonce1）と、クライアントにより生成されたノンス（nonce2）を含み、受信した固有鍵（k）を使用して暗号化された応答メッセージを作成する（２０６）。サーバーは、クライアントからの応答を復号化し、nonce1をサーバーが所有する複製であるnonce1と照合する（２０８）。２つのナンスが一致していないとき

、クライアントは認証されない（２１２）。２つのナンスが一致しているとき、クライアントは認証され、鍵の共有は完了する（２１０）。クライアントの承認後、サーバーは、ｋと連結され、ｙで暗号化されたnonce2によりクライアントチャレンジに応答する（２１４）。クライアント側では、サーバーの応答がクライアントのチャレンジを満たすとき、サーバーから受信したnonce2がクライアントが複製したnonce2と一致するか否かチェックする（２１６）。クライアントは、nonce2を自身のコピーと照合することができ、それらが一致したとき、サーバーを認証する（２１８）。ナンスが一致しなかったとき、サーバーは認証されない（２２０）。一度クライアントとサーバーの双方が認証されると、これらの間で安全なチャンネルが確立される（２２２）。

【００２３】

ナンスと鍵は、異なるセッションを切り替える認証プロセスの間に使用される。セッションは、タイマーを使用してリフレッシュされる。本発明のシステムは、各セッション中に固有の鍵と固有の128ビットのノンスを提供するために、タイマー（カウンタ）を付加した擬似乱数発生器（PRNG）を含む。ナンスは、単調増加の自然T_j（タイマー）に従うランダム性により再現不能な乱数R_j（PRN）である。R_jは、擬似乱数発生方法で生成され、T_jに包含されているので、リプレイ攻撃される可能性が極めて低いことを保証する。

$$\left\{\Pr\left(R_j|_{t=T} = R_j|_{t=T'}\right) = 1\right\} < \epsilon', \epsilon' \to 0.$$

攻撃の衝突確率は、2^{-56}程度である。

ナンスの間で予測可能な非再現性は１６ビットのT_jによって支配され、非予測可能な部分はR_jによって支配されている。

【００２４】

添付の図面を参照すると、図３は、サーバーとクライアントとの間のハンドシェイク中に含まれるステップを示す図である。それは、鍵管理アルゴリズムと軽量相互認証を示していて、D_iはクライアントを表し、Gはサーバーを表している。認証プロセスが開始される前に、秘密

$$\gamma_i = \{0,1\}^{128}$$

は、プロビジョニング処理がオフラインのとき、D_iとGの間で共有される。認証プロセスは、クライアントD_iが「HELLO、＃D_i」をサーバーＧ３００に送信するセッション開始とともに開始される（３００）。ここで、＃D_iは固有のクライアントデバイスＩＤである。一度セッションが開始されると、サーバーGは、

$$\kappa_i, nonce_1 = \{0,1\}^{128}$$

及びメッセージサイズ２５６ビットのチャレンジコード

$$AES\{\gamma_{i,}\ (\gamma_i \oplus \kappa_i \mid nonce_1)\}$$

をクライアントに送信することによって応答する（３０２）。クライアントは、チャレンジコードを復号化し、サーバーに、nonce1と追加のnonce２を含む他のチャレンジコード

$$AES\{\kappa_{i,}\ (nonce_1 \oplus \gamma_i \mid nonce_2)\}$$

を送信する（３０４）。これは、クライアントの応答及びチャレンジである。サーバー側では、サーバーは、nonce1を検証し、

$$AES\{\gamma_{i,}\ (nonce_2 \mid \kappa_i\ )\}$$

としてのk_iで暗号化されたnonce2をクライアントに送信することで応答する（３０６）

。クライアントとサーバーの両側がnonceの検証と認証を完了すると、クライアントはデータρ_iを

$$AES\{\kappa_i, (\rho_i)\}$$

としてサーバーに送信する（３０８）。

【００２５】

添付の図面を参照すると、図４および図５は、一般的なスキームの例示的な用途とセンサデバイス（クライアント）とサーバーそれぞれの間の一般的な認証ハンドシェイクなどの既存のアプリケーション層プロトコル（４００）に、本発明のシステムを埋め込んだ後の修正されたＣｏＡＰメッセージフォーマットを示している。

【００２６】

CoAPの一般的な相互作用モデルは、HTTPのクライアント／サーバーモデルに類似しており、RESTfulである。しかし、HTTPとは異なり、CoAPは、UDPのようなデータグラム指向のトランスポート上のインターチェンジと非同期で対応する。通常CoAPは、「確認可能」、「確認不能」、「承認」及び「リセット」の４種類のメッセージを含みむ。これらのメッセージは、メソッドまたはレスポンスコードに応じた要求または応答を運ぶ。

【００２７】

本発明の認証方式は、CoAPに埋め込まれたRESTfulのペイロードとして統合することができる。それは図５に見られるように、確認可能な（CON）データ転送モードとPOSTメソッドは、センサデバイス（クライアント）とサーバー間の相互認証を達成するために適用される。新しいフィールド「AUTH」は、セキュア（認証）モード４００を有効にするためにCoAPヘッダに導入される。このフィールドは、重要なオプションクラスを示す未使用のオプションに使用される。「AUTH_MSG_TYPE」という別のオプションは、認証セッションを確立するための別のメッセージを示すために、「AUTH」と一緒に導入される。

【００２８】

CoAPヘッダ内のオプションフィールドは、CoAPメッセージ内の任意の要求／応答の機能を運ぶ。本発明のために定義されたフィールドは、次の通りである。

＊　AUTH：認証／無効認証モードの有効化を示している。
　　TrueまたはFalseの値がこのフィールドに設定することができる。

＊　AUTH_MSG_TYPE：このフィールドは、「０」または「１」のいずれかであり、ここで、0 = auth_init、1 = 「response_against_challenge」である。

【００２９】

「AUTH = True」に設定することによって認証セッションが有効になったときに、ヘッダに認証処理中に交換されるすべての関連するメッセージのための「Token」の定数値を使用することによって認証セッションが保たれる。

【００３０】

図４と図５を参照すると、認証をCoAPに埋め込むために以下のステップが実行される。

＊　開始時に、センサーゲートウェイは、POSTメッセージをCONモードでサーバーの認証URIに、AUTHオプションフィールドTrue、「auth_init」例えば「０」のAUTH_MSG_TYPE値および「デバイス識別子」をペイロードとして送信する（６００）。

＊　サーバーは、ペイロードからデバイス識別子を引き出し、事前共有オプション「AUTH」を受信した後、そのデバイス識別子に関連付けられた秘密、そしてAUTH_MSG_TYPEのための「auth_init'」の値を決定する。その後、ナンス（nonce_1）と鍵（K）を生成する。サーバーは、共有秘密を使用して暗号化されたペイロードを生成する。

＊　サーバーは、新しいリソースを示す応答コードが作成されたことをクライアントに送り返す。応答におけるURIは、認証のために全体のハンドシェイクのための一時的なセッションIDを示している。無効なデバイス識別子の場合には、サーバーは、「不正」の応答コードを送信する。暗号化されたペイロードは、ピギーバックまたは別々にクライアントに送信される（６０２）。

＊　クライアントは、サーバーから受信した応答を解読し、nonce_1と「K」を得る。

そして、クライアントはナンス（nonce_2）を生成し、鍵「K」を用いて暗号化されたペイロードを生成する。それから、クライアントはこのペイロードをオプションフィールド「AUTH」とともに、「response_against_challenge」としてAUTH_MSG_TYPE値、および最後のPOSTメッセージと同じトークン値を持つPOSTメッセージを使用して送信する（６０４）。

＊　鍵リフレッシュタイマーは、セッションをリフレッシュするために維持されている。（CoAPの場合、この値は、MAX_RETRANSMIT_COUNT × MAX_RETRANSMISSION_TIMEOUT よりも大きくなければならない。）

＊　サーバーは、上記POSTのペイロードをヘッダー内の任意の値とともに「K」を用いて復号化し、受信したナンスをチェックする。サーバーは、nonceが（ステップ２で生成された）前の値と同じであると認証されたとき、リソースの変更が認証されたことを示す応答コード「Changed」を送信し、そうでなければ「認証されていない」ことを示す応答を送信する（６０６）。

【００３１】

図５を参照して使用される表記は以下のとおりである。

ψ_n :センサーゲートウェイδ_nとサーバーS との間の共有秘密

$\kappa_n|_\tau$：τ^{th}のセッションにおける、センサゲートウェイδ_nとサーバー間の鍵交換

$\langle\delta_n\rangle$：δ_nの固有のセンサデバイス／ゲートウェイ ID

$AES(.)_\kappa$：鍵κを使用して平文に AES 操作

$nonce_{\substack{i=s,\\ gw}}$：$nonce_s$=サーバー開始ナンス

$nonce_{gw}$ =センサーゲートウェイ／クライアントの開始ナンス

ω_n：センサーゲートウェイδ_nのセンサーデータ

【００３２】

認証処理が終わり、セキュアなチャネルが確立された後、クライアンがサーバーの応答に無関心であることを表示した場合など、サーバー内の一部のリソースを更新している間に、クライアントは、必要に応じて完全なオープンループモードで通信することができる。

【００３３】

本発明のシステムの一実施形態では、CoAPはNON（信頼性の無い）モードで使用され、オプションフィールド（例えば無応答）は、サーバーがリソース実行のステータスを応答する必要がないことを示すために導入される。これにより、ネットワーク上の負荷が軽減される。「NO-RESPONSE」フィールド値は、“０”または“１”であり、“０”はサーバーがステータスを応答する必要があることを示し、“１”は、サーバーが応答する必要がないことを示している。

【００３４】

添付の図面を参照すると、図６は、DTLSのようなセキュリティ層による認証のための追加の層として本発明のシステムに統合した様子を示している（５００）。

【００３５】

添付の図面を参照すると、図７は、事前共有鍵モード（PSK）によるセキュアなセッション開始のためのDTLSハンドシェイクのタイミング図を示し、図８は、本発明の方法を、安全な接続を確立する前に、派生鍵に加えて、事前共有秘密を持つ変更されたDTLSハンドシェークに帰着したDTLSフレームワークと統合した様子を示している。全てのハンドシェイクメッセージは、事前共有秘密または派生鍵Kによって暗号化されている。図７の符号“＊”は、状況依存型のメッセージを示している。図８は、図３に示すようにメッセージ交換サーバーとクライアント間のハンドシェイク時に含まれるステップ間のマッピングを示している。図7および図8を参照すると、本発明のシステムは、従来６回必要であったハ

ンドシェークを４回まで減少させている。

【００３６】

本明細書で説明した本発明によるデータグラムの転送上の軽量認証のためのコンピュータ実施システムおよび方法は、以下の技術的な向上を含むが、これらに限定されない技術的な向上も含む。

＊　セッション鍵リフレッシュタイマーを持つ、統合鍵管理と、ペイロードが埋め込まれた対称鍵ベース認証によりオーバーヘッドを減少するシステム。

＊　リソースに制約されたセンサデバイスを保護するのに最適的なシステム。

＊　交換の数を減らすことによりDTLSを軽量化するだけでなく、既存のDTLSスキームを強化するためにDTLSのようなトランスポート層セキュリティ方式と統合できるシステム。

＊　ペイロード埋め込み認証スキームとしてアプリケーションプロトコルを適応させることができるシステム。

＊　制約されたデバイスのためにCoAPのようなアプリケーション層と統合することができるシステム。

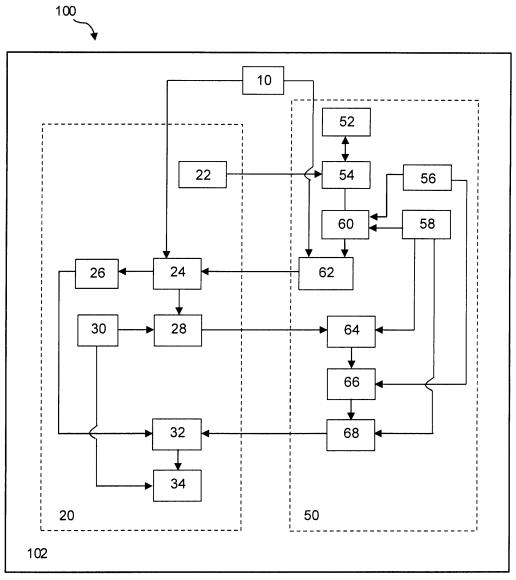
＊　認証された後のリソースの使用を最適化するために、アプリケーション層プロトコルに新しいヘッダオプションを導入することによりCoAPの開ループ通信を可能にするシステムが得られた。

＊　一般的なネットワーキング／通信システムの認証要件に応えるシステム。

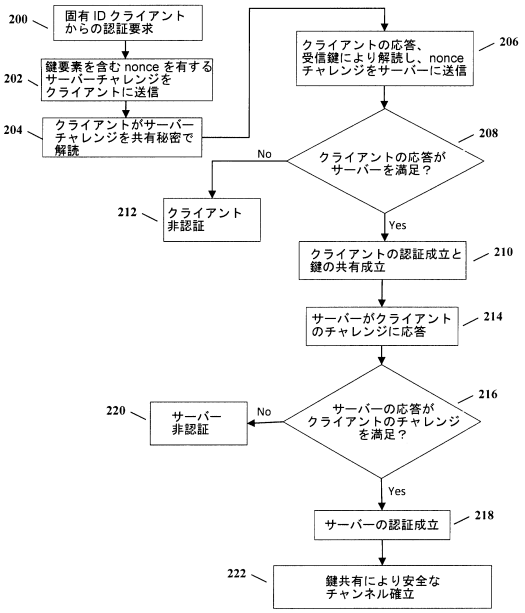
【００３７】

特定の実施形態は、現行の知識をもとに、簡単にその変更や適用を行い、基本的概念から出発することなく、即にその広い応用ができるよう、また、類推による理解を深めることのできるよう、発明の性質を全般的に示したものである。本文で使用した表現や用語は説明を目的とするもので、限定を目的としないことを理解すべきである。さらに、実施形態は、好ましい物を取り上げてはいるが、技術者は、それら実施形態に変更を加えて、実施形態の意図と適用範囲を継承しつつその応用ができることを認識できる。

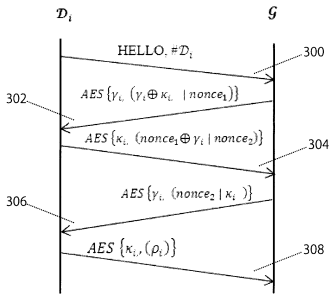
【図１】



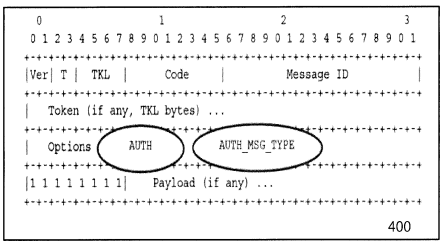
【図２】



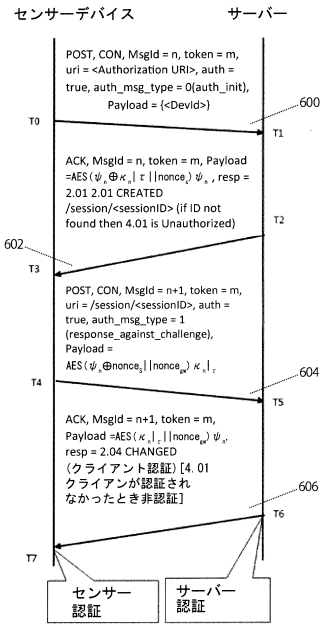
【図３】


$\mathcal{D}_i$
$\mathcal{G}$
HELLO, $\#\mathcal{D}_i$
300
302
$AES\{\gamma_{i,} (\gamma_i \oplus \kappa_{i,} \mid nonce_1)\}$
$AES\{\kappa_{i,} (nonce_1 \oplus \gamma_i \mid nonce_2)\}$
304
$AES\{\gamma_{i,} (nonce_2 \mid \kappa_i)\}$
306
$AES\{\kappa_{i,} (\rho_i)\}$
308

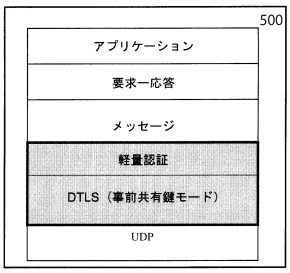
【図４】


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
|Ver| T | TKL | Code | Message ID |
| Token (if any, TKL bytes) ... |
| Options | AUTH | AUTH_MSG_TYPE |
|1 1 1 1 1 1 1 1| Payload (if any) ...
400

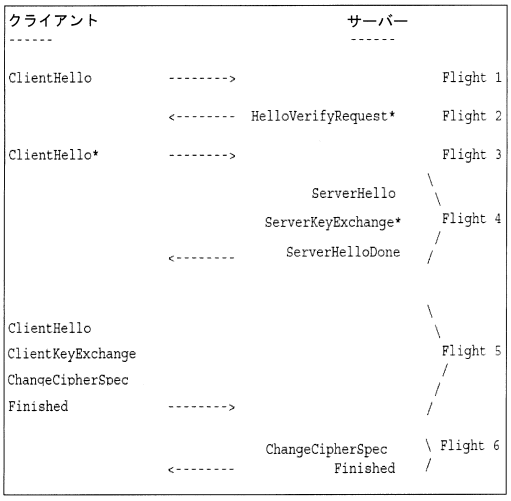
【図５】


センサーデバイス
サーバー
POST, CON, MsgId = n, token = m, uri = <Authorization URI>, auth = true, auth_msg_type = 0(auth_init), Payload = {<DevId>}
600
T0
T1
ACK, MsgId = n, token = m, Payload =AES($\psi_{s,} \oplus \kappa_{s,} \mid \tau \mid \mid nonce_{s,}$) $\psi_{s,}$, resp = 2.01 2.01 CREATED /session/<sessionID> (if ID not found then 4.01 is Unauthorized)
T2
602
T3
POST, CON, MsgId = n+1, token = m, uri = /session/<sessionID>, auth = true, auth_msg_type = 1 (response_against_challenge), Payload = AES($\psi_{s,} \oplus nonce_{s,} \mid \mid nonce_{sv,}$) $\kappa_{s,} \mid_{\tau}$
T4
604
T5
ACK, MsgId = n+1, token = m, Payload =AES($\kappa_{s,} \mid_{\tau} \mid \mid nonce_{sv,}$) $\psi_{s,}$ resp = 2.04 CHANGED (クライアント認証) [4. 01 クライアンが認証されなかったとき非認証]
606
T6
T7
センサー認証
サーバー認証

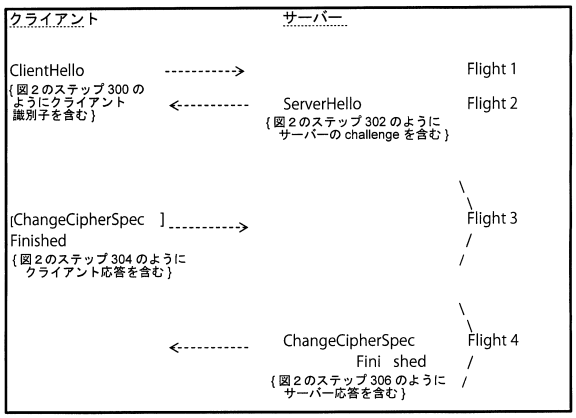
【図６】


500
アプリケーション
要求一応答
メッセージ
軽量認証
DTLS（事前共有鍵モード）
UDP

【図７】


クライアント
サーバー
ClientHello --------> Flight 1
<-------- HelloVerifyRequest* Flight 2
ClientHello* --------> Flight 3
ServerHello
ServerKeyExchange* Flight 4
<-------- ServerHelloDone
ClientHello
ClientKeyExchange Flight 5
ChangeCipherSpec
Finished --------> Flight 6
ChangeCipherSpec
<-------- Finished

【図８】


クライアント
サーバー
ClientHello
{図２のステップ 300 のようにクライアント識別子を含む}
ServerHello
{図２のステップ 302 のようにサーバーの challenge を含む}
Flight 1
Flight 2
[ChangeCipherSpec]
Finished
{図２のステップ 304 のようにクライアント応答を含む}
Flight 3
ChangeCipherSpec
Fini shed
{図２のステップ 306 のようにサーバー応答を含む}
Flight 4

フロントページの続き

(74)代理人 100166408
弁理士 三浦 邦陽
(72)発明者 アブヒジャン バッタチャリャ
インド国 ウエスト・ベンガル コルカタ ７００１５６ ラジャーハット ニュータウン エコスペース・プロット－ＩＩＦ／１２ ビルディング１Ｂ キュービクル４Ｂ－３４ タタ コンサルタンシー サービシズ
(72)発明者 ソマ バンドヨパドヒャイ
インド国 ウエスト・ベンガル コルカタ ７００１５６ ラジャーハット ニュータウン エコスペース・プロット－ＩＩＦ／１２ ビルディング１Ｂ キュービクル４Ｂ－４０ タタ コンサルタンシー サービシズ
(72)発明者 アリジット ウキル
インド国 ウエスト・ベンガル コルカタ ７００１５６ ラジャーハット ニュータウン エコスペース・プロット－ＩＩＦ／１２ ビルディング１Ｂ キュービクル４Ｂ－３３ タタ コンサルタンシー サービシズ
(72)発明者 アルパン パル
インド国 ウエスト・ベンガル コルカタ ７００１５６ ラジャーハット ニュータウン エコスペース・プロット－ＩＩＦ／１２ ビルディング１Ｂ ４階 タタ コンサルタンシー サービシズ

審査官 青木 重徳

(56)参考文献 特開２０１１－１３９４５７（ＪＰ，Ａ）
特開２００４－２４１８０２（ＪＰ，Ａ）
国際公開第２０１３／０１４６０９（WO，Ａ１）
Shahid Raza, et al., Lithe: Lightweight Secure CoAP for the Internet of Things, IEEE SENSORS JOURNAL, 米国, IEEE, ２０１３年 ８月 ７日, VOL.13, NO.10, pp.3711-3720

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｈ０４Ｌ ９／３２
Ｇ０６Ｆ ２１／４４
Ｈ０４Ｌ ９／０８
ＪＳＴＰｌｕｓ／ＪＭＥＤＰｌｕｓ／ＪＳＴ７５８０（ＪＤｒｅａｍＩＩＩ）
ＩＥＥＥ Ｘｐｌｏｒｅ