Vol.2013-CDS-7 No.5
2013/5/23

IEEE1888 を用いた EMS におけるセキュリティ運用方式の検討

赤井和幸[†]　落合秀也[††]　福田富美男[†]
古川嘉識[†]　峰野博史[†††]　江崎浩[††]

ICT によるエネルギーマネジメントシステムを実現するためのプロトコルである IEEE1888 は，異なるプロトコルやセグメントのセンサやアクチュエータ機器を容易に連携可能にする．この IEEE1888 を搭載した機器やシステムは BEMS への普及が進んでいるだけでなく,HEMS や複数の BEMS および HEMS が連携する大規模な CEMS への適応も想定されている．本研究では EMS の実現時に必要となるセキュリティについて，セキュリティやマネジメントに関する拡張規格である IEEE1888.3 及び IEEE1888.1 を用いた運用方式と,リソース情報管理機構であるレジストリを活用したアクセスコントロールルールの生成・管理およびその適用方法について検証する．

Scheme for Security in EMS using IEEE1888

KAZUYUKI AKAI[†]　HIDEYA OCHIAI[††]
FUMIO FUKUDA[†]　YOSHINORI FURUKAWA[†]
HIROSHI MINENO[†††]　HIROSHI ESAKI[††]

IEEE1888 for implementing an energy management system by ICT easily connected sensors, actuators, storage and application using different protocols and network domain to each other. Therefore, IEEE1888 has been studied not only BEMS, also adaptation to HEMS and CEMS. In this study, we verify the secure EMS scheme of using IEEE1888.1 and IEEE1888.3, which is the standard for security and management and to generate and manage access control rules using the Registry.

1. 背景

1.1 研究の背景

地球温暖化問題や東日本大震災の影響による電力危機などを背景として，電力の見える化などのエネルギーマネジメントシステム（EMS）に関する取り組みが活発に行われている．具体的にはビル設備や宅内を対象とした EMS（HEMS，BEMS）や，それらを統合した大規模な EMS（CEMS）などがあり，今後は広域・大規模化とそれによる機器やシステムの連携による高度なサービスの展開が予想される．

同時に EMS の構成要素であるセンサやアクチュエータ機器の普及も進んでいる．これらの機器はそれぞれ使用用途に合ったプロトコルを用いることが多い．例えば BEMS であれば BACnet[**1**]などが普及しており，HEMS では ECHONET[**2**]や ZigBee[3]が用いられる．これらのプロトコルも広域・大規模化に対応するため TCP/IP 上での実装の研究や標準化が進んでいる．また，機器の普及に伴って多数の機器を扱ったり，HEMS などでは運用中に構成変更や機器の増減などが発生したりすることが考えられる．そのため一部のプロトコルでは同一セグメント内の機器発見や接続通知の手順が規定され，これにより EMS 内の同じプロトコルを用いる機器同士の連携や管理を容易にする．

大規模な EMS を構築するためには，既存の EMS およびそこで用いられる機器を効率よく統合することが求められる．一方 HEMS や BEMS は前述したようなプロトコルが混在するため，連携するためにはプロトコル変換などを行う必要がある．しかしながらシステムの規模が大きくなったり，ユーザの意向に従って新しいプロトコルの機器が増加したりすることが想定されるため，随時追加される機器に対向の機器やシステムが対応することは容易ではない．

1.2 IEEE1888 プロトコル

広域・大規模 EMS への適用をスコープとして標準化されたプロトコルとして IEEE1888[4][5]がある．IEEE1888 を用いた EMS は東京大学をはじめとして，多くのビル設備などへ導入が進んでいる[6][7]．IEEE1888 のアーキテクチャを図 1 に示す．

IEEE1888 は他のプロトコル機器を収容し，通信を行う「ゲートウェイ」,データを蓄積し提供する「ストレージ」，電力見える化や制御などを行う「アプリケーション（APP）」の各「コンポーネント」とそれらが通信するための手順（FETCH，WRITE，TRAP）を規定している．またコンポーネントとそれらが管理する機器やデータの情報を管理する「レジストリ（後述）」と，コンポーネントとレジストリが通信するための手順（REGISTRATION，LOOKUP）も規定している（表 1）．

[†] NTT コムウェア株式会社
NTT Comware Corporation
1-6 Nakase, Mihama-ku, Chiba-shi, Chiba, 261-0023
[††]東京大学
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8656
[†††]静岡大学
Shizuoka University
3-4-1 ,Johoku,Naka-ku,Hamamatsu-shi, Shizuoka, 432-8011

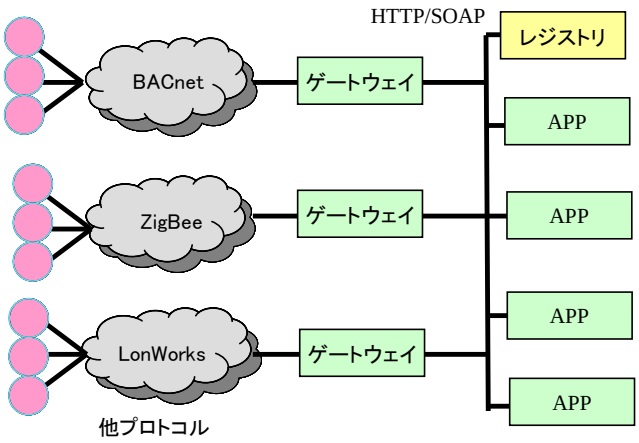


図 1. IEEE1888 アーキテクチャ概要

表 1 IEEE1888 で規定する手順

コンポーネント間通信	
FETCH	ストレージなどからデータを取得する.
WRITE	センサのデータなどを書き込む. またアクチュエータに対して操作をする.
TRAP	情報の変更などに対して指定したコンポーネントに通知するように設定する.
コンポーネント-レジストリ間通信	
REGISTRATION	コンポーネントまたはポイントの情報をレジストリに登録する.
LOOKUP	コンポーネントまたはポイントの情報を検索する.

IEEE1888 では実際に通信対象としてアクセスするコンポーネントの配下に，管理する機器やデータを示す「ポイント」という概念を持つ．これによりコンポーネント配下に複数のセンサ機器が接続されている場合や，センサデータが複数のストレージに分散して蓄積されている場合でも，一意に特定し必要な処理を行うことができる．IEEE1888 は以下の特長を持つ．

(1) SOAP による通信と他プロトコルの差異を吸収するゲートウェイ（GW）
(2) ストレージと，大量データの扱いが可能なインタフェース
(3) レジストリによるコンポーネントおよびポイントの情報管理・発見

IEEE1888 は HTTP/SOAP を用いるため既設の LAN および広域ネットワーク上で容易に構築可能であり，既存の各種プロトコルに対応した GW の実装や，小型で安価な組込みコンピュータや，複数プロトコルの混在や変更が発生する宅内向けの GW も検討が進んでいる[8][9]. またストレージのデータ蓄積により複数 APP へのデータ提供を容易化するだけでなく，GW のデータ保持量・問合せ量の削減による軽量化も期待できる．さらに IEEE1888 は他のプロトコルが規定するマルチキャストを用いた機器発見の仕組みを持たない代わりにコンポーネント情報とポイント情報の管理機能であるレジストリを規定しており，これによりセグメントに依存せずに機器やデータの発見を可能とする．レジストリは広域ネットワーク上での機器発見だけでなく[9] 複数プロトコルが混在する HEMS への適用なども検討されている[10].

2. IEEE1888 を用いたセキュアな EMS 構築の課題

既存の多くの BEMS や HEMS は専用線やローカルネットワークで運用されるため，ある程度セキュリティが担保されている．一方 IEEE1888 は TCP/IP ネットワークを用い，また既存の EMS の統合なども想定されるため，セキュアなシステム構築がより重要となる．

IEEE1888 は HTTP を用いるためセキュアな通信や認証のために TLS の利用が想定される．これは後述するセキュリティ拡張規格である IEEE1888.3 でも規定している．しかしながら TLS は基本的に通信を構成する要素が対象であり，複数のポイントを持つコンポーネントに対し，それぞれのポイントごとのアクセスコントロールを TLS だけで運用することは困難である．また HEMS や複数の EMS を統合するシステムでは構成変更や追加の頻発が想定されるため,それらが発生したときの作業の軽減も考慮する必要がある．本研究では以下の要求条件を設定する．

（1） ポイントや手順レベルのアクセスコントロール

GW やストレージは，１つのコンポーネントで複数のポイントを管理するため，ポイントや手順ごとに異なるアクセスコントロールの設定が必要となる．例えば宅内に設置する GW は, データ収集を行う APP に対してセンサのポイントへのデータ取得が可能だが，アクチュエータのポイントへの制御はできないといった事例に対して，それぞれのポイントで異なるアクセスコントロールを適用する．

（2） レジストリを含めたシステム全体で統一したアクセスコントロールリスト（ACL）の運用

IEEE1888 を用いた EMS では GW が管理するセンサのデータをストレージに蓄積し，GW とストレージの両方でデータの提供するような運用が行われる．このとき APP からセンサデータの取得を不可にする場合，そのセンサデータのポイントを保持するコンポーネントすべてに統一した ACL を設定する必要がある．またコンポーネントだけでなくレジストリについてもアクセスコントロールが必要となる．これはレジストリと同等の機能をもつ Web サービス登録・検索機構である UDDI[11]でも検討されている[12][13]. これらの機構はデータ自体の保持はしないが,機器やデータの属性情報などを保持するため，接続機器の情報が分かったり,コンポーネント URI の情報を取得されることで攻撃の起点にされたりすることが想定される．そのためレジ

ストリにアクセスするコンポーネントごとに，登録・検索が可能な範囲を制限することが望ましい．また EMS では基本的にシステム運用者がシステム全体のセキュリティを運用する．そのためシステム運用者がシステム内の全コンポーネントとレジストリの ACL の一元管理･生成ができることが必要となる．

（３） リアルタイムな ACL の生成と反映

HEMS や大規模な EMS では,システム内の機器やデータの構成変更や追加の頻発が想定される．また，システム全体のセキュリティポリシも状況に応じた変更が想定される．このような構成やポリシの変更時には ACL が随時修正され，それぞれのコンポーネントおよびレジストリがその ACL を運用できることが必要となる．

（４） 既存の IEEE1888 コンポーネントへの容易な導入

IEEE1888 のコンポーネントおよびシステムは BEMS を中心に普及が進んでいるため,それらに対して容易に適用できることが望ましい．そのため本研究では既存コンポーネントおよびレジストリに対して負担が少ない導入方式を検討する．またサーバ機能を実装しない GW やセキュリティ上の問題などからサーバの動作が制限されている機器[14]などへのセキュリティの適用も検討する．

3. IEEE1888 の拡張規格を用いたセキュアな EMS の検討

3.1 IEEE1888.3 および IEEE1888.1 の適用

IEEE1888 ではセキュリティに関する規定である IEEE1888.3 の標準化が予定されている．IEEE1888.3 が規定するアーキテクチャについて図 2 に示す．

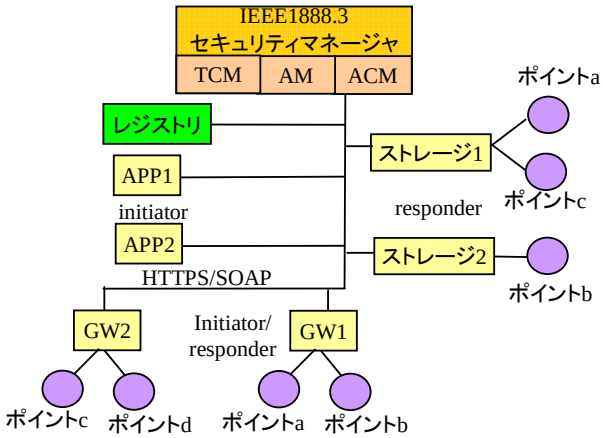


図 2 IEEE1888.3 のアーキテクチャ例

IEEE1888.3 ではコンポーネント間通信およびコンポーネントーレジストリ間通信においてポイントや手順レベルでのアクセスコントロールを行う機能（Access Control Manager:ACM）を規定している．そのため本研究ではこの規定に従った ACM を設計する．一方 IEEE1888.3 はセキュリティ導入時のガイドラインにとどまっており，具体的なインタフェースや運用方式などは規定していない．例えば ACM の設置・運用は図 2 のようなリクエストを受けるコンポーネント（responder）ごとに設置する方式と，システム内に 1 つ設置し各 responder がその ACM に問い合わせる方式が考えられる．

一方コンポーネントのマネジメントに関する規定である IEEE1888.1 も標準化が予定されている．IEEE1888.1 はシステム内に設置する中央制御装置（Manage and Control Unit:MCU）を用いてコンポーネントの死活監視や動作制御などを行うことを規定している．その MCU の 1 機能である ACL 管理機構（Resources Access Manager:RAM）には，ACL の管理および各コンポーネント（ACM）への配布と，配布時に用いるインタフェース（WRITE 手順および記述形式）が規定されている．RAM を用いた場合の ACM の運用方式について表 2 に記述する．

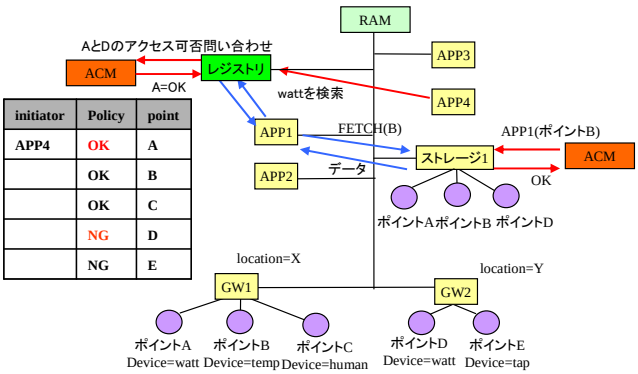
表 2 IEEE1888.3ACM の運用方式

ACMの運用方式	実装量	ACM負荷	インタフェース定義	ACL更新
システム内で１つ運用	○	×	× （responderとACM間のIF定義が必要）	○
各responderが運用	× 各responderに必要	○	○ 1888.1 （WRITE手順）	○

RAM により ACM を各 responder で運用する場合でも ACL の一元管理が可能である．また各 responder の ACM の URL の設定も不要となる．ルール配布は標準化されたポリシ記述形式 XACML[15]の利用も想定されるが[12]，WRITE 手順で必要な ACL を過不足なく記述できれば新規インタフェースが不要になるだけでなく，ACL 配信時は IEEE1888.3 が利用できるためセキュアな通信が可能とある．

3.2 ACM の検討

コンポーネントが用いる ACM は IEEE1888.3 で記述されているように，リクエストを送付するコンポーネント（initiator）からのリクエスト時に initiator が指定したポイントおよび用いた手順に対して OK または NG をレスポンスする．一方レジストリが用いる ACM はアクセス可能なコンポーネントやポイント情報の範囲をレスポンスすることが望ましい（図 3）．



initiator	Policy	point
APP4	OK	A
	OK	B
	OK	C
	NG	D
	NG	E

図 3 コンポーネントとレジストリの ACM の動作

コンポーネント間通信では，1 つのポイントにリクエストを行うと，そのポイントの情報をレスポンスする．一方 LOOKUP 手順は 1 つの検索キーで複数のポイントやコンポーネントの情報をレスポンスすることがある．そのためレスポンス時に応答可能情報と応答不可情報が混在する場合があるため，レジストリは ACM から応答可能な範囲を確認することで応答可能な情報のみをレスポンスできる．

3.3 RAM の検討

RAM は，initiator ごとにポイント及び手順分(コンポーネントであれば FETCH・WRITE・TRAP，レジストリであれば REGISTRATION，LOOKUP)の ACL を生成する．さらに本研究では以下の機能を検討する．

(1) initiator のグループ化

同じルールを適用する initiator をグループとして扱うことで ACL の生成量を減らすことが可能である．これは IEEE1888.1 や UDDI でも検討されている[12].

(2) ポイントの属性情報を用いたルール生成

システムのセキュリティポリシはコンポーネントやポイントレベルでなく，そのポイントの属性情報との関連が多いと想定される．例えば図 3 の構成では，APP によって「特定のロケーションのデータにアクセス可能」「特定のセンサデータの情報を取得可能」といったポリシが考えられる. この属性情報を元に ACL を生成することで同じ属性のポイントに対し統一した ACL が設定できると考えられる．

(3) LOOKUP 手順の ACL の補完

LOOKUP 手順は LOOKUP（component）と LOOKUP(point)があり，これらはそれぞれコンポーネント表およびポイント表（後述）を対象とする．そのためレジストリにはコンポーネント表とポイント表を対象とした ACL を生成する必要がある．このとき，あるポイント情報を特定の initiator にレスポンスしないよう設定した場合は，ポイント表を対象にした場合だけでなくコンポーネント表を対象にした場合もその情報をレスポンスしないようにする．ただしコンポーネント情報とポイント情報で異なるポリシを持つことも考えられるため個別のルール設定も可能にする．

なお，レジストリは REGISTRATION 手順もコンポーネント表とポイント表の 2 つが対象となる．コンポーネント情報の登録・更新は登録したコンポーネントのみ可能という運用が想定されるが，ポイント情報の登録・更新はコンポーネントと独立した運用が想定される. そのため initiator ごとに設定した範囲のみを登録・更新できるようにする．

3.4 RAM とレジストリの連携

IEEE1888 ではコンポーネント情報とポイント情報の更新はレジストリに随時反映されるため，その情報を活用することで随時 ACL の更新が可能となる．レジストリから以下の情報が取得可能である．

- コンポーネント URI や名前，対応メソッドなどのコンポーネントそのものの情報と，そのコンポーネントが管理しているポイントの情報（コンポーネント情報）
- ポイント ID およびその属性（ポイント情報）

コンポーネント情報により同一のポイントが複数のコンポーネントで管理される状態も把握可能である．また ACM の配布先（responder）もコンポーネント情報から判断できる．さらにレジストリを 1 つの responder とみなせば，コンポーネント情報・ポイント情報からレジストリ用の ACL の生成が可能となる．よって図 4 のように RAM は随時レジストリに情報を問い合わせ，ACL を生成・配布することで ACL のリアルタイムな生成が可能になると考えられる．

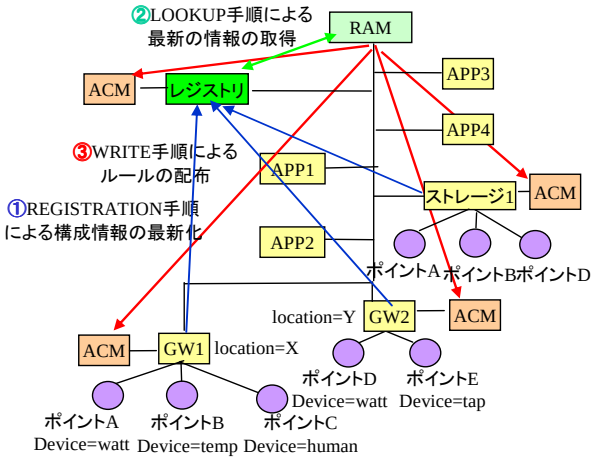


図 4 レジストリを用いた ACL の更新・配布方式

4. ACM と RAM の実装

4.1 ACM の実装

本研究では ACM を各 responder に設置する方式を採用し，コンポーネント（ストレージ）およびレジストリにそれぞれ ACM を実装し，ストレージとレジストリに受信した ACL を ACM に反映する機能を追加した. そのためレジストリにも WRITE のサーバ機能を実装した. 一方 IEEE1888.1 が規定した ACL 配布の電文構成ではレジストリの ACL を記述できない．これはレジストリが単独で複数のコンポーネントの情報を管理するためである．よって表 3 のようにレジストリへの ACL 配布電文を拡張した.

4.2 RAM の実装

RAM は LOOKUP 手順を用いてレジストリからすべてのコンポーネント情報及びポイント情報を取得し，それを用いて ACL を作成, WRITE 手順で responder に配布する機能を実装した．ACL 作成のため，本研究では検討に従い表 4 のテーブルを用意した．まず Role の定義テーブルにより initiator をグループ化し, ポイントの属性情報ルールテーブルより指定した属性のポイントと，そのポイントを管理するコンポーネントに対して統一した ACL 生成を行う. さらにコンポーネントそのものに依存するルールを設定するため，コンポーネントのルールテーブルを用意する．なお，これらのテーブルでコンポーネントだけでなくレジストリ

表 3　IEEE1888.1ACM のレジストリへの電文構成

RCM	Type	Description
acl/number_of_responders	**UInt**	**コンポーネント(responder)の数**
acl/responderW/name	**String**	**role名またはコンポーネント名**
acl/responderW/number_of_roles	Uint	The number of roles. The content is an integer equal to or greater than 0.
acl/responderW/roleX/name	String	The name of the role. Hereby X is an integer greater than 0 to number_of_roles.
acl/responderW/roleX/number_of_rules	Uint	The number of rules assigned to the roleX. The content is an integer equal to or greater than 0.
acl/responderW/roleX/ruleY/policy	String	The policy for the ruleY. The content shall be either allow or deny in ascii string. Hereby Y is an integer greater than 0 to number_of_rules.
acl/responderW/roleX/ruleY/method	String	The method name for the ruleY. The string is either query, data, or service in ASCII string.
acl/responderW/roleX/ruleY/number_of_points	Uint	The number of points for the ruleY. The content is an integer equal to or greater than 0.
acl/responderW/roleX/ruleY/pointZ	POINTID	Either of the identifier of the point, a single asterisk, or an amount of prefix of the pointed with an asterisk in the tail.

表 4　RAM が用いるテーブル

Roleの定義

role	initiator
operator	APP1
userA	APP2
userB	APP3,APP4

ポイントの属性情報ルール

name	value	role	policy	method
Location	X	userA	OK	WRITE|FETCH|LOOKUP
device	watt	userB	OK	FETCH|LOOKUP
device	temp	userB	OK	FETCH|LOOKUP

コンポーネントのルール

iniator	responder	policy	point	method
operator	ストレージ1	OK	*	WRITE|FETCH|LOOKUP
operator	レジストリ	OK	*	LOOKUP

の ACL もすべて記述可能である. 具体的にはコンポーネントのルールテーブルで, responder にレジストリを指定した場合はレジストリのポイント表に対するルールと解釈し, responder にコンポーネントを指定した場合は, Method に記述された LOOKUP, REGISTRATION 手順がコンポーネント表に対するルールと解釈する.

ルールの生成時は取得したコンポーネント情報を用い, すべての ACL を NG とする. 次にポイント情報を用いてポイントの属性情報ルールを上から順次適用し, 次にコンポーネントのルールを上から適用する.

5. 動作検証と評価

5.1 検証環境

本研究では RAM と ACM を実装したストレージ・レジストリに加えて既存のサーバ機能を実装しない GW(SNMP 通信による電力値取得及び電源の制御ができるスマートタップを接続)とセンサデータの見える化およびアクチュエータ制御を行う APP にそれぞれ IEEE1888.3 対応を実施し, IEEE1888 で規定する全コンポーネント及びレジストリを用意した. 検証に用いた構成と, 各機能に実装した手順を図 5 に示す.

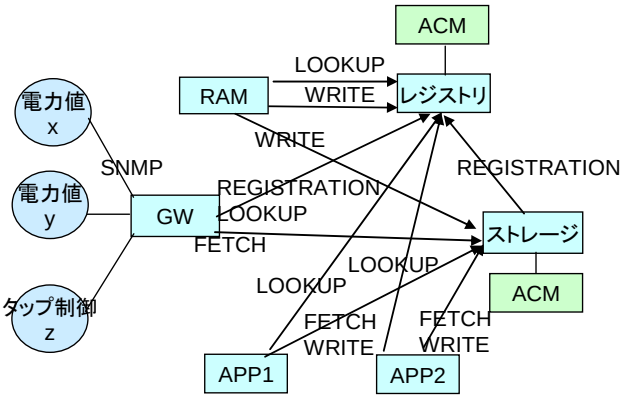


図 5 検証の構成及び実装した手順

また使用した環境は以下の通りである.

- OS: Ubuntu 10.04
- Java:OpenJDK1.6.0
- Web サーバ:Apache Tomcat 7.0
- Web サービスエンジン:Apache Axis2 1.5.6
- CPU: Core i7-640M 2.8GHz(RAM, GW), Core2 Duo P8700 2.53GHz(APP, レジストリ, ストレージ)

5.2 IEEE1888.3 と IEEE1888.1 の接続確認

図 5 の各構成要素にデジタル証明書（IEEE1888.3 の記述形式に従ったクライアントおよびサーバ証明書）を配布した. その上でコンポーネント間通信およびコンポーネントーレジストリ間通信を行い, IEEE1888.3 を用いた接続を確認した. さらに RAM から IEEE1888.3 を用いた WRITE 手順での ACL の配布を行い, IEEE1888.1 の接続も確認した.

5.3 ポイントや手順レベルのアクセスコントロール

図 5 の環境にて, ポイントごとに属性をレジストリに登録し, それぞれ異なるルールを設定し, ACL を生成・配布した. その後各 APP からレジストリ及びストレージにアクセスを行った場合, LOOKUP 手順のリクエストに対して ACL で設定した範囲のみをレジストリがレスポンスすることと, 各ポイントへの FETCH 手順のリクエストに対し, ストレージが ACL に従ってレスポンスする(電流値の応答または NG）ことを確認した.

5.4 システム全体で統一した ACL の運用

複数のコンポーネントで同一のポイントを保持する構成である図 3 の情報をレジストリに登録した上で表 4 のル

ールを用いて RAM で ACL の生成を行い，コンポーネントの ACL72 個とレジストリの ACL108 個が過不足なく生成されたことを確認した．このとき Location 及び Device の属性に基づいたルールを設定し，GW とストレージおよびレジストリ（ポイント表とコンポーネント表）で統一された ACL が生成できたことを確認した．

5.5 リアルタイムな ACL の生成と反映

図 5 の環境にて，GW の追加時に ACL の生成と配布を行い APP からアクセスを行う一連の動作を実施した（図 6）．

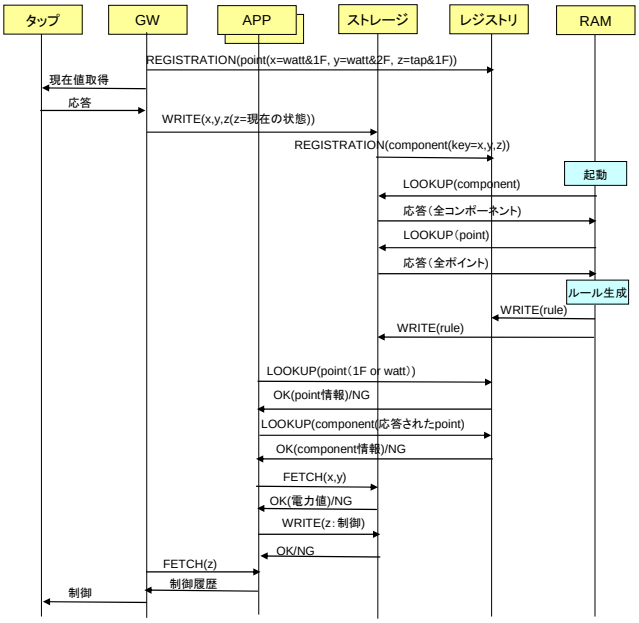


図 6 ACL 更新のシーケンス

この動作によりコンポーネントの追加時も設定した情報に従って随時 ACL が更新でき，それが各 ACM に反映できたことを確認した．また RAM のルールを変更した場合も同様にレジストリから情報を取得し，ACL を更新・配布できたことを確認した．

また ACL 生成の処理時間を計測した．属性のルールを 3 つ設定し，複数のポイントを管理するコンポーネントをレジストリに登録した．その情報を用いて ACL を生成した．このとき各コンポーネントが管理するポイント数およびコンポーネント数を変更した場合の ACL 生成時の処理時間を計測したところ図 7 となった．また各コンポーネントの管理するポイント数が同じ場合（それぞれ 4 ポイント）に，一部のポイントを複数のコンポーネントが管理するように登録した場合の ACL 生成の処理時間は図 8 となった．

5.6 既存の IEEE1888 コンポーネントへの容易な導入

図 6 にて，GW の管理するポイントの情報をレジストリに登録し，そのポイントをストレージでも管理するようにした．その後ストレージのコンポーネント情報をレジストリに登録し，RAM がその情報を用いて ACL を生成，ストレージおよびレジストリに配布した．このとき APP から GW のポイントの機器発見をした場合に，設定した ACL に従って，ストレージが発見出来ることと，発見したストレージへの FETCH 手順による電力値取得と WRITE 手順によるアクチュエータ制御についてアクセスコントロールができることを確認した．その後 GW がストレージに対して操作履歴を問い合わせ，その情報を用いてアクチュエータ制御することを確認した．これらの一連の動作により，提案したアーキテクチャを用いることでサーバ機能を実装しない GW も ACM を実装した他のコンポーネントと同様のアクセスコントロールができることを確認した．

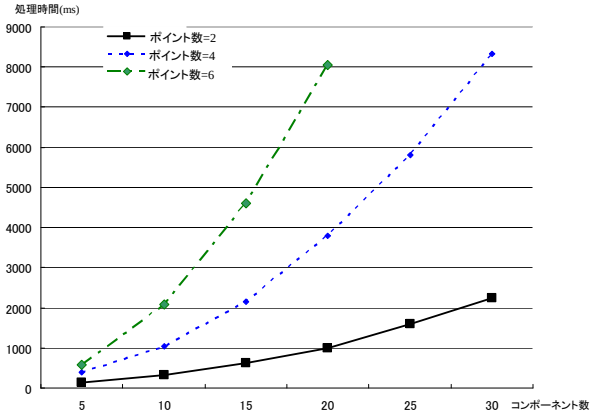


図 7 ACL 生成時間（ポイント数が異なる場合）

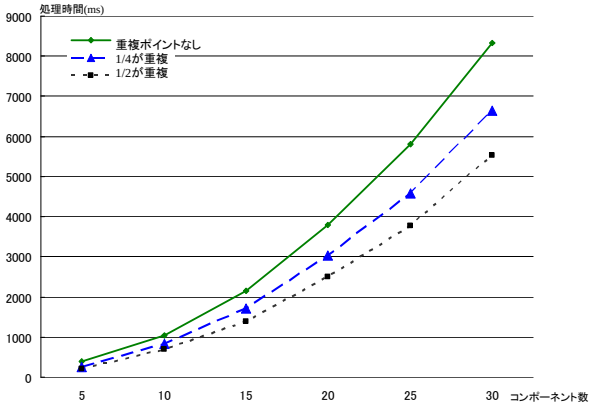


図 8 ACL 生成時間（一部ポイントを重複して管理する場合）

5.7 考察

検証により IEEE1888.3 および ACM を用いてポイントや手順レベルでのセキュアな通信とアクセスコントロールが実現できることが確認できた．さらにレジストリを含めた ACL の配布が WRITE 手順でできることを確認し，新規インタフェースの定義やそれによる実装が不要であることと，サーバ機能を実装しない GW についても追加の実装なしでアクセスコントロールが実現できることを確認した．また RAM による ACL の一元管理とレジストリを用いた ACL のリアルタイムな更新及び複数のコンポーネントへの統一したルールの設定ができることも確認した．これらの結果より提案したアーキテクチャによって要求条件を満たすことを確認できた．

本研究では RAM はユーザが任意のタイミングで実行するためコンポーネントからの REGISTRATION 手順とは非同期で動作している．そのためレジストリと RAM を同一

のサーバで運用し REGISTRATION 手順が行われたことを把握するか，コンポーネント情報の expire パラメータによって REGISTRATION 手順が行われるタイミングを推測するといった方法による改善が考えられる．

ポイントや手順レベルで ACL を設定する場合, 図 3 のような小規模な構成でも多くの ACL が生成されるが, 実際のコンポーネントはすべての手順を実装しないことが多いため不要な ACL の削減が可能である．なお，コンポーネントが実装する手順はコンポーネント情報としてレジストリへの登録・参照が可能なため，その情報を用いればよい．一方レジストリは LOOKUP 及び REGISTRATION 手順を基本的に実装しているため，不要な手順の削減はできない．本検証では各 ACM は ACL 受信時にルールをすべて再設定するため ACL をまとめて送付しているが, システムの規模が大きくなった場合には記述形式の工夫や，変更が発生した箇所のみ送付などによる ACL 電文量の削減が必要になる．例えば図 3 の場合，実際の ACL の電文量はおよそ 1200 行になるが，複数のポイントで同じポリシ（OK/NG）のものを 1 つで記述するなどの工夫によって半分以下に削減が可能である．

また処理時間を測定した結果，コンポーネントの ACL 生成数が同じ場合でもシステムの構成で生成時間が大きく異なることも確認できた．これはコンポーネントに比べてレジストリの ACL 設定がシステムの構成に大きく依存するためと考えられる．しかしながらシステムの構成に関わらず規模に従って処理時間が大幅に増大するため，大規模なシステムではレジストリからデータを取得した際に過去のデータとの差分を抽出し, その差分だけ ACL を生成するなどの改良が必要と考えられる．

6. まとめ

本研究では IEEE1888 を用いたシステムを対象とし，IEEE1888.1 と IEEE1888.3 を活用したセキュアなシステム構築とレジストリへの適用を検討した．さらにレジストリと連携した ACL 更新およびルール生成効率化や軽量 GW への適用について検証を行い，提案したアーキテクチャの有効性を示した．

一方，現在のレジストリで管理する情報は基本的に responder の情報のみであるため，現方式では initiator の追加・変更時の対応は難しい．そのため今後はレジストリの initiator の情報管理や検索方式の検討を行い，ACL 生成の効率化を検討する. また大規模 ACL 配布の削減方式やレジストリを活用したセキュリティマネジメント機構の検討を行い，既存の IEEE1888 システムへのセキュリティ導入・運用のさらなる容易化について検討を進めていく．

参考文献

1)ASHRAE BACnet，http://www.bacnet.org/
2)ECHONET CONSORTIUM，http://www.echonet.gr.jp/
3)ZigBee Alliance，http://zigbee.org/
4)IEEE 1888: http://standards.ieee.org/index.html
5)Ochiai, H., Ishiyama, M. and Momose, T. and Fujiwara, N., Ito, K., Inagaki, H., Nakagawa, ., Esaki, H., : FIAP: Facility information access protocol for data-centric building automation systems, Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on, pp.229–234 (2011).
6)Hiroshi Esaki, Hideya Ochiai, "*The Green University of Tokyo Project*", (in Japanese), IEICE Trasactions on Communications, Vol.J94-B, No.10, pp.1225-1231, October 2011
7)東大グリーン ICT プロジェクト，http://www.gutp.jp/
8)落合秀也，井上博之，"ネットワーク温度＆照度計 後編 Ethernet シールド付き Arduino にアップロードのためのライブラリを搭載", トランジスタ技術, CQ 出版社, vol.49, no.2, pp.189--195, 2012 年 2 月.
9)赤井和幸，福田富美男，福島伸夫，梁田龍治，古川嘉識: “スマートコミュニティ実現に向けた IEEE1888 による機器情報管理手法の検討”情報処理学会研究報告,2012-CDS-4(10),2012.5.
10) Hideharu Seto, Masahide Nakamura, and Shinsuke Matsumoto, ``Ubi-Regi: Service Registry for Discovering Service Resources in Ubiquitous Network," In The 13th International Conference on Information Integration and Web-based Applications & Services (iiWAS2011), pp.395-398, December 2011.
11) L. Clement, A. Hately, C. von Riegen and T. Rogers: “UDDI version 3.0.2”, UDDI Spec Technical Committee Draft,Dated 20041019 (2004). http://uddi.org/pubs/uddi v3.htm.
12) Juan Dai, Robert Steele, UDDI Access Control, In Proceedings of the Information Technology and Applications,2005
13) Adams, C. & Boeyen, S., 2002, “UDDI and WSDL Extensions for Web Services: A Security Framework”, Proceedings of the ACM workshop on XML security, Session 2, p. 30-35
14) 東日本電信電話株式会社 “フレッツ・ジョイント”, http://flets.com/joint/
15) A Brief Introduction to XACML [Online], 2003, Available:http://www.oasisopen.org/committees/download.php/2713/Brief_Introduction_to_XACML.html