Resource Centric Security to Protect Customer Energy Information in the Smart Grid

Eun-Kyu Lee, Rajit Gadh, and Mario Gerla
University of California, Los Angeles, Los Angeles, CA, USA
eklee@cs.ucla.edu, gadh@ucla.edu, gerla@cs.ucla.edu

***Abstract*—From the customer domain perspective, interoperation implies that external systems are able to control customer's energy resources as well as to read energy-related information. These two types of accesses to an energy resource affect the operation of the customer domain differently. However, most existing security mechanisms were designed at individual resource level and cannot efficiently handle such fine-grained access. To resolve the issue of fine granularity, this paper proposes a new security mechanism, *Resource Centric Security*, that leverages the concept of a filesystem Access Control List. Three privileges of read, write, and execute are defined on each energy resource, and a set of attributes is assigned to each privilege. Each external user also maintains his own set of attributes. He can access the privilege only if his attribute set matches the privilege's set. In this way, the user may receive permission to read data of a resource but not to invoke operations. We have implemented the proposed scheme on a real testbed and have run experiments. The results and following analysis discover that our scheme can provide a proper level of data protection with reasonable overhead.**

I. Introduction

Interoperation is the most important keyword in smart grid. To achieve the customer domain interoperability, National Institute of Standards and Technology (NIST) conceptually defines Energy Service Interface (ESI), a communication interface that is responsible for transmitting customer energy data at the domain boundary [9]. Standardization efforts have been developing standard ways of how to represent resource data and how to expose internal information. More specifically, new data models define energy resources and related services as objects. That is, each resource, as an object, maintains a set of values representing its status and operations actuating its behaviors. Then, the external system accesses the object via web service interfaces implemented in ESI; it controls the customer's energy resources as well as to reads energy-related information.

One interesting property in the interaction model is a clear distinction between reading data and invoking operations within an object, i.e., "fine granularity of object access". The property is especially highlighted in the smart grid context because two accesses affect the operation of the customer domain differently. For instance, leakage of customer data violates privacy policy while an abuse of control capability can harm the customer directly. Therefore, the security mechanism at the ESI must be able to understand the difference. Unfortunately, however, few security schemes under the standardization efforts address this issue. Furthermore, as more energy objects are added to the customer domain and the domain interconnects with various external systems, the interactions become extremely complicated. But, existing security schemes, relying prior knowledge of user list and database, cannot handle the complexity in an efficient way: "They do not scale".

To solve the problems, we propose a new security mechanism, *Resource Centric Security (RCSec)*, that provides an access control and data encryption. To address the fine granularity issue, RCSec leverages the concept of a filesystem Access Control List (ACL), in which each file (object) maintains an entry that predefines three classes of *user*, *group*, and *others* and determines which privileges (*read*, *write*, and *execute*) are assigned to each class. In RCSec, we do access control reversely. That is, we define three privileges within an energy resource, and then assign a set of attributes[1] to each privilege. Unlike the ACL, RCSec does not predefine the classes in advance. Instead, each accessing user must show a matched set of attributes to obtain the privilege. In this way, the user may receive permission to read data of a resource but not to invoke operations. To address scalability, we implement RCSec by exploiting Attribute Based Encryption (ABE). Given a set of attributes assigned to each resource, associated data is ABE encrypted using this set. Each external user maintains a private key consisting of his own set of attributes, and is able to decrypt the ciphertext only if his attribute set matches the resource set. Unlike the ACL, each user in RCSec is responsible for managing his own attributes. Thus, the resource (or the ESI) is free from maintaining an access control entry to perform the authorization process. This enables a security system to cope with complex interactions as well as to operate in a large-scale smart grid environment. We implement the proposed RCSec on top of our testbed of Energy Management and Control System (EMCS) and evaluate its performance in terms of overhead. Experimental results discover that the performance of the proposed scheme relies on underlying encryption algorithm.

The rest of the paper is organized as follows. Section II reviews standardization efforts that develop data and service models in the customer domain and identifies new challenges of security. Section III describes the proposed RCSec scheme consisting of encryption, privilege assignment, and authorization protocol. In Section IV, we illustrate our implementation, and the proposed scheme is evaluated with experimental results. Finally, we conclude the paper in Section V.

[1]An attribute is a property that represents the resource. For instance, an LED at an office 127 may have two attributes "LED" and "OF-127".

II. Data Model and Energy Service Interface

A. Handling Heterogeneity for Interoperability

The customer domain is a primary energy consumer, and potential inclusion of Electric Vehicle (EV) [13] and Distributed Energy Resources (DER) will enable it to store and generate power too. Thus, interoperation with the domain is essential in smart grid. However, as a customer domain system has been implemented with proprietary data and service models, a seamless interoperation still remains a non-trivial challenge [12]. To cope with the issue, several standardization efforts are under development.

First, customer data must be represented in a standard form. A simple proprietary format may be helpful for the performance of individual system. But, the customer domain consists of heterogeneous devices and networks that generate different types of data. It is impossible that an external system handles all the unstructured information. To overcome the difficulty, development of common data models have been discussed. Standardization efforts touch this issue including Facility Smart Grid Information Model (FSGIM) [1], ISO/IEC 15045, Building Automation and Control Networks (BACnet)/ISO 16484-5 [2], Open Building Information eXchange (oBIX) [5], ZigBee Smart Energy Profile (SEP) [8], and OPC Unified Architecture [6].

Second, standardized customer data must be exchanged in an interoperable manner. In an inter-domain service model, the ESI transmits data of customer energy services, e.g., energy usage measurement, remote load control, and monitoring and control of distributed generation. It is also responsible for interoperation. To this end, the ESI accepts external signals, e.g., a command message to control customer energy assets, as well as to exposes the customer energy data to external systems in a standard way. Several standardization organizations are developing the interoperable service models: Energy Interoperation (EI) [3], Energy Market Information Exchange (EMIX) [4], Open Automated Demand Response (OpenADR) [15], and Energy Services Provider Interface (ESPI) [7].

B. Observation

The standardization efforts discussed above show three interesting characteristics. First, eXtensible Markup Language (XML) is used for data representation. Data format in BACnet and oBIX is initially defined with XML, and existing KNX is also mapped to XML data format [14]. Especially, the extensible nature of the user-defined XML schema is well leveraged when mapping analog data set to digital representation. Most service models also exploit XML technology. OpenADR and EMIX define their specifications in the XML format.

Second, the way of defining customers' energy data follows an Object-Oriented (OO) design pattern. While some protocols show this property in their schema design and UML (Unified Modeling Language) representation, explicit examples can be found in oBIX and BACnet. Given predefined primitive objects such as "int" and "list", an object in oBIX is modeled with data types (values) and operations (method signatures). An object can represent a physical device directly or represent a collection of information related to a particular function.

Last, a web service technology is exploited for the inter-domain transportation of the customer data. The OpenADR specification defines two web service connections, SOAP (Simple Object Access Protocol) and REST (Representational State Transfer). EI utilizes web service technologies such as WS-Calendar and WS-Addressing for additional functionality. This sounds natural because the data is encoded in the XML format. But, it is noted that the ESI benefits the most from web services' capability of machine-to-machine communications in a distributed environment. This support is essential to the automation of the smart grid system. For instance, dynamic pricing programs requiring minimized human intervention can be achieved effectively only by automation technology.

When looking at three characteristics, one notices that they all together maximize the interoperability and automation. That is, each energy resource distinguishes its values and operations, and web services expose them by implementing three actions to the resource: Read, Write, and Invoke. These efforts are summarized as "fine granularity of data access and load controls". For instance, a user only reads energy usage of an air conditioning system, while another user may turn off the system to reduce power consumption during the on-peak period.

C. Security on the Interface

Fine granularity is a new challenge of a security mechanism at the ESI, because different actions induce different operation consequences and indicate different levels of privacy penetration. For instance, energy usage data is read to calculate bills, but the read privilege can be misused to infer private activities of the residents. The energy assets can be configured to perform an automated demand response strategy. However, if they are controlled unfavorably, the actions would have a detrimental impact on the activities. If an adversary abuses the control privilege, and sets to maximize energy loads during the peak, the reliability of smart grid would be seriously threatened. To avoid these potential problems, the owner of the energy assets wants to permit utilities to read parts of data and service providers to control contracted energy loads only, while he has a full control over his assets.

Current security schemes being considered under standardization efforts, however, cannot support the same level of granularity efficiently. The ESI or the EMCS authenticates users using its own database of ID-password sets and authorizes them with a coarse-grained rule. This way, a user would have a full control over a group of energy resources at once. However, the owner may not want the user to have excessive rights to access the resources. He would like to apply the principle of least privilege so that the user is given minimum permission that are essential to that user's work. Given the requirement of fine granularity of new data and service models in the customer domain, these schemes cannot realize the principle in an effective way. Moreover, the requirement gets greatly complicated, as more energy resources are added to the customer domain, and more external users are connected the domain. To cope with the increasing complexity of interactions, existing schemes must manage a volume of user lists and authorization rules, and develop corresponding enforcement mechanisms,

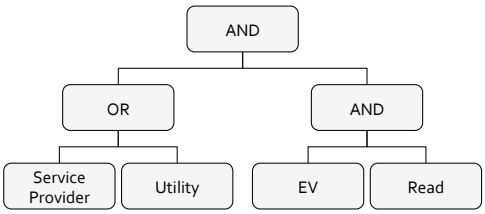

Fig. 1. An access policy tree is created when Alice (owner) encrypts data. Two users, Kevin and Sara, have own sets of attributes.

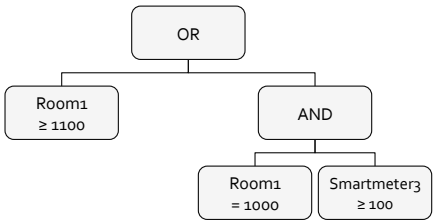

Fig. 2. Each attribute in the access policy tree is assigned privilege. Kevin, having 4 attributes, only satisfies the tree and accesses the object Smartmeter3.

which causes additional overhead to them. Thus, they cannot easily scale up in a large-scale, distributed smart grid network. To overcome the challenges, we propose a *Resource Centric Security (RCSec)* approach that takes the concept of data and service model into consideration.

III. Resource Centric Security

Based on our observation of abstraction level, the proposed RCSec realizes a fine-grained, scalable security mechanism through encryption, privilege assignment, and authorization.

A. Encryption

To achieve confidentiality, RCSec leverages the concept and implementation of Ciphertext-Policy Attribute-Based Encryption (CP-ABE) that encrypts data using user attributes [10]. CP-ABE realizes the secret sharing scheme [17] using bilinear map based Pairing-Based Cryptography (PBC). More specifically, each user is assigned a set of shares (attributes), and a data sender encrypts data using an open key and an arbitrary set of attributes. The encryptor creates an access policy tree, representing a Boolean formula defining the combination of attributes in the ciphertext. If a user presents a proper credential, i.e., any combination of his attributes satisfies the tree, he recovers the secret and is authorized to access the data. To make the tree secret, CP-ABE exploits a polynomial interpolation technique that guarantees information theoretic security. To prevent collusion attacks, an authority assigns a random number to each user whose attributes are also tied with the number.

Fig. 1 illustrates an example of an access policy tree that consists of two types of Boolean logic gates and four attributes at the leaf position. Two users, Kevin and Sara, have 5 attributes in their private keys. A decryption process begins from the leaves by matching their attributes, and each gate returns true to its parent if children satisfy the logic. If the root returns true, then the user recovers data successfully. In this way, Kevin accesses Alice's data, but Sara cannot.

B. Privilege Assignment

Each attribute represents a state of permission and does not relate to other attributes. Suppose that Kevin in Fig. 1 is permitted to read and write the thermostat data, but only to read the EV data. But, his attributes indicate his right to write the EV data. This happens mainly due to the discrepancy between the concept of attribute and an object model. To exploit the attributes in the resource centric model at the ESI, we apply a filesystem ACL that has been used in modern operating systems. In ACL, each object (e.g., a file in a Unix system) maintains own Access Control Entry (ACE), and a 3-digit code represents privilege to access the object - a user can read from, write to, or execute the object.

In RCSec, each object maps to an attribute with 3-digit privilege level. For instance, an object Smartmeter can be represented as an attribute "Smartmeter = 111". The first digit indicates permission of Read, and the following two digits indicate the rights of Write and Invoke, respectively. Thus, when a user has an attribute "Smartmeter = 100" and tries to access the object, it is permitted to read energy usage information but cannot turn on/off the device. In our implementation, the ESI encrypts an object data with attributes in which appropriate privileges are assigned. The privilege assignment allows inequality, e.g., "Smartmeter $\geq$ 100", in the access policy tree, whereas this is not used in user attributes. The inequality expression significantly simplifies the assignment rule. For instance, if a user has an attribute "Smartmeter = 111", he is still able to satisfy the inequality condition and to read data. In this way, the expression is capable of testing multiple privileged attributes at once.

The proposed assignment rule also supports hierarchical types of objects. Say, a building has more than one room, and several Smartmeters are deployed in each room. In this hierarchy, each room is also identified as an object that does not provide energy information directly. Instead, the corresponding XML document provides meta data about the object and access information to the sub objects of Smartmeters. For such resources, we assign privilege with 4-digit code. The first digit represents permission to access the object itself. An example is "Room1 = 1000". The last three digits have the same semantics to the 3-digit code, but with different scope of permission. For instance, "Room1 = 1100" implies that the user can read all the data produced in Room 1. This implies that the "Room1" attribute is more inclusive than the "Smartmeter" attribute and provides higher level of privilege. This rule makes it much easier to manage attributes. When there are 10 Smartmeters in the room, a user can use one attribute instead of ten attributes to access them. Fig. 2 depicts an access policy tree in which privilege is assigned according to the proposed rule.

C. Authorization Protocol

A user accesses an object in three ways: Read, Write, and Invoke. Authorization for the Read is performed at a user side with his own private key. Both Write and Invoke, on the other hand, occur at the customer domain upon receiving requests from the user. The ESI is not allowed to have the user's private

key that is required for the authorization[2]. Thus, we design an authorization protocol leveraging our encryption and privilege assignment.

The authorization for Write and Invoke follows almost the same procedure, and the followings describe 4 steps of procedure for the Invoke operation. We use below notations.

- u, v are two end systems. In our example, they are an user and an ESI, respectively.
- $u \rightarrow v : M$ denotes that u sends a message M to v
- $M_1|M_2$ is the concatenation of messages M_1 and M_2
- $H(M)$ is hash of M (e.g., SHA-1).
- T_x is an access policy tree containing an attribute x.
- $\{M\}_T$ is attribute based encryption with a tree *T*.
- $[M]_K$ is symmetric key encryption with a key K (e.g., AES).
- N is a random nonce value.

INVOKE REQUEST (IR). A user u generates and concatenates a request message M_{req} and a nonce N_u. M_{req} includes information about an operation that u requests v to execute. The concatenated data is then encrypted with an access policy tree T_{bruin}, where $bruin$ denotes the name of v, i.e., the ESI. Note that T_{bruin} does not imply that the tree has only one attribute. We assume that Certificate Authority (CA) assigns the $bruin$ attribute only to v, and the attribute is not forged or stolen. Thus, v is only able to satisfy the tree. The ciphertext M_{IR} is then delivered to v as follows.

$$u \rightarrow v : M_{IR},\ M_{IR} = \{N_u|M_{req}\}_{T_{bruin}}$$

AUTHORIZATION REQUEST (AR). Once v receives and decrypts M_{IR}, it obtains N_u and M_{req}. And it generates a nonce N_v. As mentioned, v does not manage users list or store any state information of external requests to achieve a lightweight and stateless distributed system. To this end, it creates and sends u a message M_{op} that stores the state information. M_{op} is not intended to expose to u but expected to return back to execute the operation later. Thus, v encrypts the message with a key K_{uv} and creates a new message M_x as below. The key is created using both v's private pseudonym PS_v and N_u. PS_v saves v's burden to remember u, while K_{uv} is still related to u through N_u.

$$M_x = [M_{op}]_{K_{uv}}$$
$$M_{op} = (N_v|M_{req}|T_i),\ K_{uv} = H(N_u) \oplus H(PS_v)$$

where $\oplus$ denotes an XOR operation. T_i represents the time when M_{op} is generated and is used to protect communication against replay attacks. v also encrypts N_v with an access policy tree T_{invoke} and creates a message, $M_y = \{N_v\}_{T_{invoke}}$. This is to challenge u if it is qualified or not. Two messages together are now encrypted with a key $(N_u + 1)$ and delivered to u as follows.

$$v \rightarrow u : M_{AR},\ M_{AR} = [M_x|M_y]_{(N_u+1)}$$

AUTHORIZATION ACK (AA). Upon receiving and decrypting M_{AR} with own nonce N_u, u obtains M_x and M_y. It cannot decrypt M_x, instead returns back to v. u recovers a nonce from M_y only if it satisfies the tree T_{invoke}. Let N'_v be the recovered nonce. u, then, collects three data, encrypts the collection with T_{bruin}, and transmits the ciphertext M_{AA} to v as follows.

$$u \rightarrow v : M_{AA},\ M_{AA} = \{M_x|(N_u + 2)|(N'_v + 2)\}_{T_{bruin}}$$

INVOKE ACK (IA). After receiving and decrypting M_{AA}, v obtains M_x, N_u, and N'_v. Using N_u, it decrypts M_x and obtains N_v, T_i, and M_{req}. Then, the authorization confirms the followings - (1) $N'_v = N_v$; and (2) $T_c - T_i \leq \Delta$, where T_c is the current time at v, and Δ denotes a timeout threshold. Once confirmed, v executes the request, M_{req}. An acknowledge message M_{ack} after the operation is encrypted and delivered to u as follows.

$$v \rightarrow u : M_{IA},\ M_{IA} = [M_{ack}]_{(N_v+3)}$$

D. Advantage

The RCSec scheme provides a few advantages. First, it supports fine granularity. An object is treated separately with distinguished attributes, and three request types for the object are given different privileges. Furthermore, these concepts are effectively implemented within the conceptual boundary of an attribute. Second, RCSec is developed based on an encryption algorithm. Thus, it inherently supports confidentiality and helps guarantee customer privacy. Last, RCSec is scalable. It does not require the ESI to maintain user information. Instead, each user manages own privilege in the form of attributes. Thus, RCSec can scale well even in a distributed system environment.

IV. PERFORMANCE EVALUATION

To validate the proposed RCSec, we implement it within our EMCS testbed. Developed on a server system running the Eeebuntu distribution, it gathers energy data periodically from various energy loads. Collected data is stored and managed in the oBIX format. The EMCS also implements the ESI that realizes HTTP-based web service communications, and RCSec runs in the ESI. We omit the analysis of underlying encryption algorithm due to space limitation. Instead, we refer [16] for interested readers.

A. Implementation

1) oBIX: Following the OO paradigm, each object in oBIX is modeled by a set of *value* objects like "str" and "bool" and a set of *op* objects that define operations with input and output objects. The object model also allows inheritance to model complicated oBIX resources by means of a contract mechanism. Realized by *is* object, it establishes the classic "is a" relationship with overriding rules. oBIX supports lower level of abstraction, which gives a huge flexibility. This benefits the customer domain in that heterogeneous data formats must be effectively represented in a common data model.

2) HTTP REST: In order to expose data to external domains, the ESI implements the web service of the oBIX specification - HTTP binding in the Representational State Transfer (REST) style. Providing a resource centric access instead of method centric one, REST utilizes a small set of

[2]In addition, RCSec pursues a pure distributed system in which the ESI never maintains any database of user list, whereas conventional authorization schemes rely on ACL stored in them for authorization.

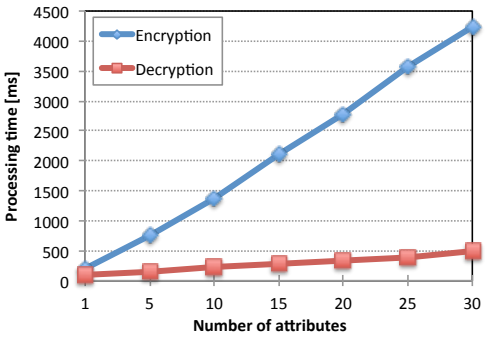

Fig. 3. Data processing time [ms] for encryption and decryption in the attribute based encryption.

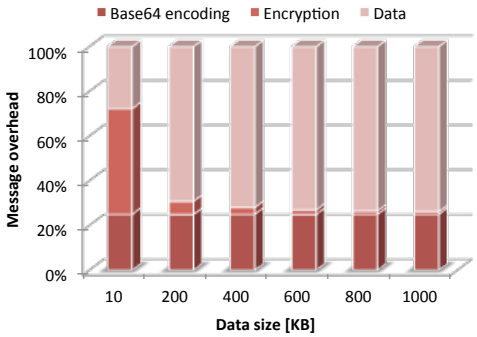

Fig. 4. Message overhead - a breakdown of a message in terms of volume.

Num. attribute (N_A)	1	5	10	15	20	25	30
Msg. overhead [KB]	2	8	17	26	35	43	52

TABLE I
MESSAGE OVERHEAD WITH VARYING NUMBERS OF ATTRIBUTES.

verbs to transfer an object's state via XML [11]. oBIX maps three oBIX requests to HTTP methods: Read with GET, Write with PUT, and Invoke with POST. Below oBIX document represents a smart plug object, "plug1" in which the energy load is controllable via two ways: Write and Invoke. To turn it on or off, an external user sends directly a PUT request targeting at the *connectLoad* object within the same URI or sends a POST request to the hyperlinked URI targeting at the operation object, *controlLoad*.

```
<obj href="http://myPAS/zigbee/plug1/">
  <str name="deviceName" val="BSPE12SOYZM43001"/>
  <bool name="connectLoad" writable="true" val="true
      "/>
  <op name="controlLoad" href="control" in="
      obix:WritePointIn" out="obix:Point"/>
  <ref name="power" href="power"/>
</obj>
```

3) Access Control and Encryption: The proposed RCSec is implemented in the ESI. In particular, it encrypts oBIX data and the authorization works with three oBIX requests, i.e., in the application layer. To implement details of RCSec, we take algorithms approved in the NISTIR 7628 guideline [18]. More specifically, we use AES-256 for the symmetric key encryption, SHA-256 for the hash function, and SHA-1 based random number generator to generate the nonce value. In addition, we leverage CP-ABE for attribute based encryption. Base64 encoding transforms the encrypted bytes to printable ASCII strings so that data is delivered in the XML form over the web. In order to minimize the overhead of encryption processing and to improve data access time, popular data is encrypted in advance and stored in cache, while MySQL based database management system manages all the data. Therefore, Read request is handled quickly with cached data. Whereas, both Write and Invoke requests require the authorization process, which introduces processing delay.

B. Experiments

Most security algorithms protect user information at the cost of additional overhead and latency. This is inevitable because stronger algorithms usually come with expensive data processing. To evaluate the proposed scheme, therefore, we examine the level of overhead in details. All the experiments are conducted by two computers that run with 2.2 GHz Intel Core 2 Duo processor and 2 GB memory.

1) Encryption and Decryption: The first experiment assesses performance of the attribute based encryption which is the most expensive portion in RCSec. In this experiment, we vary the number of attributes (N_A) used in the algorithm, and then measure processing time to encrypt and decrypt data that is 200 KB in size. As illustrated in Fig. 3, the processing time grows in proportional to N_A. Encryption is much more sensitive to N_A than decryption. When N_A=30, encryption is around 8 times slower than decryption. This is mainly attributed to difference of mathematical complexity within the encryption and decryption. On the other hand, another experiment reveals that the data size V_D barely affects the processing time although the results are omitted due to space limitation. From the results, we reason that the encryption part dominates the processing overhead of the encryption, and N_A used in encryption primarily influences the overhead.

2) Message Overhead: Having the impact of the attributes on the processing time in mind, next experiments investigate how the attributes affect the message volume. In addition to an encrypted data, a message to be transmitted contains meta information about the access policy tree and corresponding mathematical expressions. Table I shows extra volumes increased with varying numbers of attributes. We set V_D=200 KB. The result indicates that adding one attribute in encryption expands the entire message size by 1.75 KB on average. This way, the message overhead becomes 52 KB when N_A=30.

However, we note that the message overhead does not relate to V_D. That is, the overhead remains 17 KB with 10 attributes even when the algorithm encrypts data of 1000 KB. This property allows us to calculate the overall message overhead. For instance, Fig. 4 draws a breakdown of a message in terms of volume, where we vary V_D while fixing N_A=10. The figure also shows the overhead due to the Base64 encoding, which accounts for 25% all the time. Because of the fixed size of 17 KB, the encryption overhead accounts for 47.2% when V_D=10KB. As V_D increases, its portion decreases dramatically: 3.1% for 400 KB and 1.6% for 800 KB. The results again reveal noticeable influence of the attributes on the proposed security scheme.

3) Latency on Authorization: The proposed authorization protocol comprises of 4 steps (IR, AR, AA, and IA), each

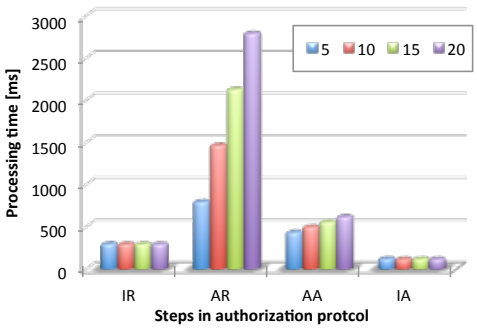

Fig. 5. Processing time of individual step in authorization along with increasing numbers of attributes.

of which involves different computational operations. This experiment investigates the performance of the protocol by showing the breakdown of the processing time of each step. In our scenario, a user (client) requests a list of energy usage data for specified period to our EMCS (server) - i.e., oBIX history service. The retrieved list data, 300 KB, is included in the acknowledge message in the IA step (M_{ack}). Four messages, M_{IR}, M_{AR}, M_{AA}, and M_{IA}, are 5 KB, 24 KB, 5 KB, and 400 KB in size, respectively (see Section III-C for notations). As the server is assumed to have unique attribute in its private key, we can minimize the number of attributes in T_{bruin} (N_{Ab}) for IR and AA. We set N_{Ab} = 2 in the experiments. N_{Ai} for T_{invoke}, on the other hand, can change along with security policies, and our experiments vary the value from 5 to 20. Note that the processing time measured includes latency to generate and parse XML data.

Fig. 5 illustrates the results showing that AR dominates the entire processing overhead, and its influence grows as N_{Ai} increases. When N_{Ai} = 20, it accounts for 73% of the whole overhead. The overhead of AR mainly comes from the attribution base encryption, i.e., M_y. When looking at the results from previous experiments together, the overhead of M_y accounts for 94.3% in AR. The overhead of AA increases slowly along with N_{Ai} - 434.6 ms when N_{Ai} = 5 and 618.6 ms when N_{Ai} = 20. Such growth is mainly attributed to the decryption of M_y. Another major portion of overhead in AA comes from encryption using T_{bruin} that generates M_{AA}. Unlike AR and AA, the overheads of IR and IA barely change. IR encrypts data using T_{bruin} whose number of attributes is 2 all the time. In IA, the algorithm involves one attributed based decryption and one symmetric encryption, which enables the processing time to keep below 120 ms. The results together conclude that reducing N_{Ai} is very critical to run RCSec in an efficient way. Note that our running system currently uses 4$\sim$8 attributes.

V. Conclusion

We have presented a novel security mechanism *Resource Centric Security* that provides fine-grained, scalable access control and encryption in the service interface of the customer domain. To support fine granularity, it leverages the concept of a filesystem access control list so that individual energy resource maintains three privileges of read, write, and execute. Instead of having three predefined classes of accessing users, RCSec dynamically assigns a set of attributes to each privilege. And, an external user can only obtain permission to each privilege by showing that his own attribute set matches the resource's set. To provide confidentiality, we implement RCSec on top of attribute based encryption that encrypts data using the assigned set of attributes. RCSec scales well and fits to distributed smart grid environment because the ESI works without any prior knowledge of user information. The experimental results and following analysis discover that RCSec can provide a proper level of abstracted data protection with reasonable overhead. Developing a compact RCSec will be an integral part of our future work.

Acknowledgment

This work has been sponsored in part by grant from the EPRI/DOE - fund 20699, Smart Grid Regional Demonstration Project.

References

[1] ASHRAE Standard Project Committee 201 (SPC 201) Facility Smart Grid Information Model (FSGIM). http://spc201.ashraepcs.org/standards.html.

[2] BACnet - A Data Communication Protocol for Building Automation and Control Networks. http://www.bacnet.org/.

[3] OASIS Energy Interoperation. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=energyinterop.

[4] OASIS Energy Market Information Exchange (EMIX) v1.0. http://docs.oasis-open.org/emix/emix/v1.0/emix-v1.0.html.

[5] oBIX - Open Building Information eXchange. http://www.obix.org/.

[6] OPC Unified Architecture. http://www.opcfoundation.org/Default.aspx/01_about/UA.asp?MID=AboutOPC.

[7] REQ.21 Energy Services Provider Interface (ESPI), North American Energy Standards Board (NAESB). http://www.naesb.org/ESPI_Standards.asp.

[8] ZigBee Smart Energy 2.0. http://www.zigbee.org/Standards/ZigBeeSmartEnergy/Version20Documents.aspx.

[9] NIST Framework and Roadmap for Smart Grid Interoperability Standards, Release 2.0. Feb. 2012.

[10] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, Oakland, USA, May 2007.

[11] R. Fielding and R. Taylor. Principled Design of the Modern Web Architecture. *ACM Transactions on Internet Technology*, 2(2):115–150, 2002.

[12] R. Gadh. Convergence for the smart grid - on the technology opportunities for future cyber-physical energy systems. In *Workshop on New Research Directions for Future Cyber-Physical Energy Systems*, June 2009.

[13] S. Mal, A. Chattopadhyay, A. Yang, and R. Gadh. Electric Vehicle Smart Charging and Vehicle-to-Grid Operation. *Int'l Journal of Parallel, Emergent and Distributed Systems*, 27(3):1–17, 2012.

[14] M. Neugschwandtner, G. Neugschwandtner, and W. Kastner. Web Services in Building Automation: Mapping KNX to oBIX. In *IEEE Conference on Industrial Informatics*, June 2007.

[15] Piette, M. Ann, G. Ghatikar, S. Kiliccote, E. Koch, D. Hennage, P. Palensky, and C. McParland. Open Automated Demand Response Communications Specification (Version 1.0). *California Energy Commission - PIER Program*, CEC5002009063, 2009.

[16] M. Pirretti, P. Traynor, P. Mcdaniel, and B. Waters. Secure Attribute-Based Systems. *Journal of Computer Security*, 18(5):799–837, 2010.

[17] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[18] Smart Grid Interoperability Panel - Cyber Security Working Group. Introduction to NISTIR 7628 Guidelines for Smart Grid Cyber Security. Sep. 2010.