A Security Architecture for the Publish/Subscribe C-DAX Middleware

Florian Heimgaertner*, Michael Hoefling*, Barbara Vieira†, Erik Poll‡ and Michael Menth*
*University of Tuebingen, Chair of Communication Networks, Tuebingen, Germany,
Email: {florian.heimgaertner,hoefling,menth}@uni-tuebingen.de
†Software Improvement Group, Amsterdam, Netherlands, Email: b.vieira@sig.eu
‡Radboud University Nijmegen, Digital Security Group, Nijmegen, Netherlands, Email: erikpoll@cs.ru.nl

Abstract—**The limited scalability, reliability, and security of today's utility communication infrastructures are main obstacles for the deployment of smart grid applications. The C-DAX project aims at providing a cyber-secure publish/subscribe middleware tailored to the needs of smart grids. C-DAX provides end-to-end security, and scalable and resilient communication among participants in a smart grid. This work presents the C-DAX security architecture, and proposes different key distribution mechanisms. Security properties are defined for control plane and data plane communication, and their underlying mechanisms are explained. The presented work is partially implemented in the C-DAX prototype and will be deployed in a field trial.**

I. Introduction

Today, smart grid (SG) refers to the next-generation electrical power grid designed to enhance the resilience of the grid to power flow disruptions, improve energy efficiency, and reduce carbon emissions. To accomplish these goals, the modern electrical power grid will incorporate a wide variety of SG applications, e.g., synchrophasor-based real-time state estimation [1], electric vehicle charging, and future retail energy transactions [2]. However, one of the main obstacles in the way of the deployment of SG applications is the limited capabilities of today's utility communication infrastructure in terms of scalability, reliability, and security.

The *Cyber-secure Data and Control Cloud for power grids* (C-DAX) project [3] develops such a cyber-secure communication middleware for smart grids, applying the publish/subscribe (pub/sub) paradigm to enable scalable, transparent, and secure end-to-end communication [4] between publishers and subscribers. Additional major advantages of C-DAX include resilient communication [5], and support for real-time applications [1].

The main contribution of this paper is the description of the C-DAX security architecture, and a presentation and discussion of its key distribution mechanisms. Parts of the security concept are already implemented in the C-DAX prototype, and will be deployed in a real-world power grid as part of a field trial.

This work is structured as follows. We review relevant aspects of the C-DAX communication architecture in Section II and present the C-DAX security architecture in Section III. In Section IV, we present methods for distributing updated symmetric keys for data plane communication and discuss their properties. We review related work in Section V and draw conclusions in Section VI.

Barbara Vieira was with the Digital Security Group of the Radboud University Nijmegen while the work was done.

II. C-DAX: A Cyber-Secure Data and Control Cloud for Power Grids

C-DAX is an FP7 project funded by the European Commission which adapts the pub/sub paradigm to the needs of power grids. It aims at developing a cyber-secure and scalable communication middleware for SGs to facilitate the flexible integration of emerging SG applications. We give a brief overview on the C-DAX architecture, its components, and basic interactions. Further details on the C-DAX architecture and its features can be found in [5] and [1].

A. Topic-Group Communication

C-DAX uses the pub/sub paradigm to decouple communication partners in space, time, and synchronization [6], [7]. Information is organized in so-called *topics*. A topic is an abstract representation of a unidirectional information channel, and is addressed using its unique name and possibly attributes, e.g., data type, location, and time. Publishers and subscribers register at a *broker* for a certain topic. Publishers send messages for that topic to the broker, which eventually forwards them to the subscribers. Data transmitted within a topic is called *topic data*.

B. Components

Figure 1 illustrates the basic structure and interactions of the C-DAX architecture. It is composed of C-DAX clients and the C-DAX cloud. SG applications use *C-DAX clients* as interface to the C-DAX cloud, which handle all C-DAX signaling transparent to the respective application. *Publishers* are C-DAX clients generating data for a specific topic. *Subscribers* are C-DAX clients interested in certain topic data. *C-DAX nodes* form the *C-DAX cloud*, and provide a specific set of functions to the cloud and clients.

Designated nodes (DNs) provide access for clients to the C-DAX cloud. They act as first point of contact and are responsible for forwarding topic data to and from the cloud, i.e., clients are pre-configured with DNs. *Data brokers (DBs)* store and forward topic data to DNs. Each topic is assigned

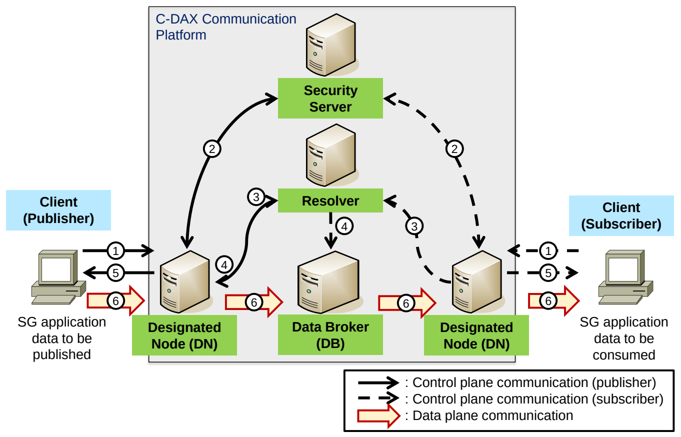
C-DAX Communication Platform
Security Server
Resolver
Client (Publisher)
Client (Subscriber)
SG application data to be published
Designated Node (DN)
Data Broker (DB)
Designated Node (DN)
SG application data to be consumed
: Control plane communication (publisher)
: Control plane communication (subscriber)
: Data plane communication

Fig. 1. The C-DAX architecture. Basic signaling steps include client join (step 1, 2, and 5), data plane configuration (step 3 and 4), and topic data transmission (step 6).

to a DB, where its publishers send topic data to. DBs store topic data for a certain time, and forward it to the topic's subscribers. The exact assignment of topics to DBs is subject to management decisions, and may be changed during runtime.

Topic names need to be mapped to DBs so that join requests can be sent to appropriate DBs that manage registrations. To that end, *resolvers (RSes)* hold topic-to-DB mappings and provide a resolution interface through which they answer mapping requests of other nodes. Security-related functionalities are provided by a *security server (SecServ)*, e.g., authentication, authorization, and key distribution.

In this paper, we use term *forwarding nodes* for DNs and DBs. The term *component* refers to both C-DAX clients and C-DAX nodes.

C. Basic Interactions

C-DAX components act jointly to provide communication services. We describe basic interactions using the examples of data publication and subscription.

1) Publication of Topic Data: The initial message exchange prior to topic data publication is shown on the left side of Figure 1; solid black lines represent publisher-specific control plane communication. When the publisher wants to publish topic data, it first sends a join message to the SecServ over its DN using the topic identifier (step 1) for publisher authentication and authorization (step 2). When the authentication and authorization of the publisher is successful, the DN forwards the join message to the RS using the topic identifier (step 3). The RS looks up its database for the topic-to-DB mapping. If such a mapping exists, the RS sends the responsible topic-to-DB mapping to the DN which installs a forwarding entry for that topic in its internal forwarding table (step 4), and the DN sends a join acknowledgment message to the publisher (step 5). The publisher starts pushing data to its DN which forwards it to the responsible DB which stores the topic data (step 6).

2) Subscription to Topic Data: Topic data retrieval works similarly. The initial message exchange prior to topic data retrieval is shown on the right side of Figure 1; dashed black lines represent subscriber-specific control plane communication. When the subscriber wants to retrieve topic data, it first sends a join message to the SecServ over its DN using the topic identifier (step 1) for subscriber authentication and authorization (step 2). When the authentication and authorization of the subscriber is successful, the DN forwards the join message to the RS using the topic identifier (step 3). At the same time, the DN installs a topic-to-client entry in its internal forwarding table, and sends a join acknowledgment message to the publisher (step 5). The RS looks up its database for the topic-to-DB mapping. If such a mapping exists, the RS forwards the join message to the responsible DB which installs a topic-to-subscriber's-DN entry in its internal forwarding table (step 4), and starts pushing topic data to all registered subscribers' DNs (step 6).

III. C-DAX Security Architecture

We now describe the security architecture of C-DAX. We first discuss the design rationale behind the security concept, introduce the basic terminology, and finally specify the required mechanisms and keys which are used to implement those properties in C-DAX.

A. Design Rationale and Terminology

Topic data transmission should be protected end-to-end because (1) only legitimate publishers may publish data for a certain topic, (2) only legitimate subscribers may receive data from a certain topic, and (3) third parties (incl. DNs, DBs, and malicious clients) must not modify or spoof topic data. The actually required security properties for the topic data transmission may vary depending on the smart grid applications, and the C-DAX middleware must be capable of supporting them.

The security architecture of C-DAX provides *topic access control*, *end-to-end integrity* and *end-to-end confidentiality* of published data, and *authentication* of clients and nodes. We describe those security features in detail below. In contrast to more innovative solutions for security in information-centric smart grid middleware presented in [4], the current C-DAX middleware uses authentication and encryption mechanisms based on standard cryptographic primitives, i.e., it can be implemented based on established and trusted security libraries. The C-DAX security architecture does not restrict the type of cryptographic primitives (i.e. symmetric or asymmetric) used to secure the communication. Nevertheless, for performance reasons we rely mainly on symmetric primitives to enforce the data plane security properties.

We write T for the set of all topics and K_t for a *topic key* associated with a topic $t \in T$. Topic keys are generated by the SecServ, and the SecServ distributes the topic keys to the respective components as part of the join response message.

B. Security Properties

1) Source Authentication: Source authentication is required for control plane messages. When processing request messages, the SecServ needs to verify the identity of the component before looking up the permissions of the requesting party in its access control list (ACL). The same requirement applies to configuration messages, where DBs need to verify, that such a messages originates from an authorized node, e.g., an RS.

Source authentication is realised using asymmetric cryptography., e.g., RSA. Each component is assigned a public/private key pair (K^+, K^-). Control plane messages are digitally signed using the private key K^-_{sender} of the respective sender. The receiver can verify the signatures using K^+_{sender}.

As usual, certificates are used to link these keys to identities. Certificates issued and signed by the SecServ provide identity information, the associated public key, and additional attributes. The additional attributes include C-DAX function information about permission to modify node configurations, e.g., for the RS function. Certificates can be attached to the signed messages. Additionally, the SecServ provides a certificate revocation list (CRL) to allow certificates to be revoked.

Because of the decoupling of publishers and subscribers, source authentication for data messages is not available in most pub/sub systems. Even though source authentication is not needed for pure topic based communication, there are smart grid applications (e.g. retail energy transactions) where messages from a publisher could lead to a binding contract. For such applications, digital signatures generated with K^-_{sender} can be used to authenticate the sender of a data plane message.

2) Topic Access Control: Topic access control is required for all topics in order to prevent unauthorized clients from publishing data. We use a shared symmetric key K_t^{auth} to implement topic access control. This key is used to compute hash message authentication codes (MACs), and is shared among authorized publishers and involved forwarding nodes for topic t. When publishing a data message msg for topic t, publishers use this key to add MAC(K_t^{auth}, msg) to the message. The forwarding nodes verify the MACs of incoming messages, and only forward messages with valid MACs; messages are discarded otherwise.

3) End-to-End Integrity: End-to-end integrity for all topics enables subscribers to verify the integrity of received topic data without having to trust intermediate forwarding nodes. The topic key K_t^{e2e} is introduced to implement end-to-end integrity in C-DAX, and is used as a shared secret to generate a MAC. In contrast to K_t^{auth}, the SecServ distributes K_t^{e2e} only to the publishers and the subscribers of topic t, i.e., the forwarding nodes do not know K_t^{e2e}. Subscribers can verify that an original message msg was not altered during forwarding when the received message contains MAC(K_t^{e2e}, msg).

4) End-to-End Confidentiality: Confidentiality means that only the intended receivers of a message can read the message content. Because control plane and data plane in C-DAX do not share the same concept of receivers, we use different mechanisms to ensure end-to-end confidentiality for control plane and data plane messages.

a) Control Plane Messages: C-DAX control plane communication consists of point-to-point messages, i.e., only the single intended receiver of a message should be able to read the message content. End-to-end confidentiality is especially important for all control plane messages containing topic keys. We use asymmetric cryptography to achieve this requirement. The SecServ uses the public key K_c^+ of a component c to encrypt the topic key K_t.

b) Data Plane Messages: Data plane communication in C-DAX is essentially many-to-many communication, i.e., only the subscribers of topic t should be able to read messages published to that topic. End-to-end confidentiality is required for transmission of personal data, e.g., smart metering data for residential buildings. Asymmetric encryption using individual public keys of the subscribers is not possible for data plane messages. As the pub/sub paradigm decouples publishers and subscribers, the publishers do not know the subscribers and their respective public keys. Therefore, we use symmetric ciphers to encrypt the payload of pub/sub data plane messages using the topic key K_t^{e2e}, e.g., AES.

As mentioned above, only publishers and subscribers receive K_t^{e2e}. DNs and DBs cannot decrypt the actual message content but can detect and discard unauthenticated messages because they possess K_t^{auth}. However, if the forwarding configuration can be manipulated, publishers are able to decrypt messages sent to the same topic by other publishers. Diversified keys for publishers can be used to prevent this. Then subscribers are still supplied with K_t^{e2e}, which now acts as a master key, while each publisher receives a unique identifier x and derived key $K_t^{e2e_x}$, derived from the master key for this value of x using some key derivation function (KDF): $K_t^{e2e_x} = \text{KDF}(K_t^{e2e}, x)$. Messages encrypted by a publisher using $K_t^{e2e_x}$ now need to include the publisher's x in the unencrypted message header. On reception of a message, subscribers derive the symmetric key for decryption and MAC verification $K_t^{e2e_x}$ using the KDF, K_t^{e2e}, and x. Publishers cannot derive the keys of other publishers without knowing the master key K_t^{e2e}, so publishers can no longer decrypt any data plane messages except their own. A similar approach is proposed in the REMP [8] protocol.

C. Application of the Security Mechanisms

We now show how the security properties are ensured by applying the security mechanisms described above. We give an example of how the mechanisms are applied during publication of confidential topic data, and provide a summary of the keys and mechanisms used in C-DAX.

Table I provides an overview of the keys used in C-DAX, the component or topic the key is associated with, and the components that know the key.

Figure 2 depicts the publication of data over the C-DAX cloud. The SecServ distributes the topic keys encrypted with the public keys of the clients and nodes. Publishers and subscribers receive both K_t^{auth} and K_t^{e2e} (step 1) while the forwarding nodes only receive K_t^{auth} (step 2). The publisher encrypts the message using K_t^{e2e} and generates one MAC using K_t^{auth} and another MAC using K_t^{e2e} (step 3). The DNs and DBs forward the message after verifying the MAC using K_t^{auth} (step 4). After receiving the message, the subscriber

TABLE I
OVERVIEW ON KEY TYPES IN C-DAX.

Name	Description	Associated With	Known By
K_c^-	Component private key	component c	only known by component c
K_c^+	Component public key	component c	may be known by all components
$K_{SecServ}^+$	SecServ public key	SecServ	must be known by all components
K_t^{auth}	Topic access control key	topic t	publishers, DNs, and DBs for topic t
K_t^{e2e}	End-to-end security key	topic t	publishers and subscribers for topic t

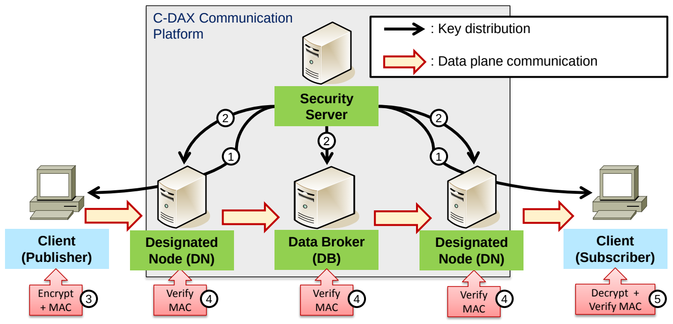

Fig. 2. C-DAX security mechanisms applied for publication of topic data.

uses K_t^{e2e} to verify the MAC and decrypt the payload (step 5).

TABLE II
C-DAX COMPONENTS AND SUPPORTED OPERATIONS.

Component	Mechanisms
SecServ	Key generation ACL lookup Signing Signature verification Asymmetric encryption Certificate revocation
DB / DN (forwarding nodes)	Signing Signature verification MAC verification Asymmetric decryption
Publisher	Signing MAC generation Symmetric encryption Asymmetric decryption
Subscriber	Signing Signature verification MAC verification Symmetric decryption Asymmetric decryption

Table II maps the C-DAX components to the mechanisms they need to support for their operation. While key generation, authorization, and asymmetric encryption is only performed by the SecServ, all components have to support signing. The components involved in data plane communications (i.e. clients and forwarding nodes) need to support asymmetric decryption of topic keys. Additionally, the forwarding nodes need to verify signatures and MACs. The publishers need to generate MACs and perform symmetric encryption. Subscribers have to verify MACs and need to do symmetric decryption. For use cases like retail energy transactions they also need to verify publisher signatures.

IV. KEY DISTRIBUTION MECHANISMS

We now describe key distribution mechanisms to securely distribute new topic keys based on the pub/sub mechanisms already provided by the C-DAX infrastructure. We first describe the requirements and prerequisites for secure distribution, propose two key distribution mechanisms, and finally discuss approaches for scheduling key updates

We use the notation of the symmetric topic key $K_t^{*,i}$, where K_t^* can be one of the keys K_t^{e2e} or K_t^{auth}, and i is the index in a chronological series of K_t^* for topic t. When a key update is performed for a topic t with the current key $K_t^{*,i}$, the updated key is denoted as $K_t^{*,i+1}$. Because updated keys need to be delivered not only to subscribers but also to publishers, we define a corresponding *key-update topic* t' for each regular topic t. The original subscribers and publishers for topic t are subscribers of topic t', and the SecServ is the only publisher for topic t'.

A. Requirements

To make sure that messages for a topic originate from legitimate publishers and can only be read by legitimate subscribers, *backward secrecy* and *forward secrecy* are required for the topic keys. We use the definitions from Steiner, Tsudik and Weidner [9] that have been adopted for the terms forward and backward secrecy in later literature [10]. *Backward secrecy* is defined as the guarantee that “old, previously used group keys must not be discovered by new group members”. *Forward secrecy* is defined as the guarantee that “new keys must remain out of reach of former group members”.

In the case of K_t^{e2e}, full forward and backward secrecy is required to prevent subscribers from decrypting messages that were not published during their subscription period. That means, the topic encryption key needs to be changed each time a subscriber joins or leaves the topic. For K_t^{auth} only forward secrecy is required because MACs generated with previous keys cannot be used for publishing data. Therefore, K_t^{auth} only needs to be changed when a publisher leaves the topic t.

To maintain forward secrecy, the new topic key $K_t^{*,i+1}$ cannot be transmitted encrypted using the old topic key $K_t^{e2e,i}$. Therefore, we must rely on asymmetric encryption to distribute the new keys and individually encrypt $K_t^{*,i+1}$ using $K_{c_1}^+ ... K_{c_n}^+$, with $\mathcal{C}_t = \{c_1, ..., c_n\}$ being the set of publishers and subscribers for topic t.

B. Distribution of Asymmetric Keys

As a prerequisite for secure key distribution in C-DAX, the component’s asymmetric key pair (K_c^+, K_c^-) and the public key $K^{+}_{SecServ}$ of the SecServ are pre-installed on each component c. Those key pairs are intended to be long-term keys, i.e., they are only changed if the original key is considered compromised or otherwise insecure. As there is no secure way to remotely install a new key on a device whose keys can no longer be trusted, manual intervention is required anyway. Therefore, we do not define automated update mechanisms for this.

C. Topic Key Update: a Push Approach

As a naïve solution, the SecServ can distribute updated keys by publishing them to t'. The distribution of the updated keys could be done in individual messages or concatenated to one large message.

This approach has two major scalability drawbacks. The first and more obvious problem is that the SecServ needs to transmit n encrypted keys through all DBs and DNs. The clients would receive multiple keys but can only decrypt one of them. To reduce this overhead at the receiver side, filters can be deployed at the DNs to reduce the number of keys delivered to the individual clients at the cost of increasing the complexity of DNs. As an alternative, separate topics could be created per client at the cost of increasing the complexity of DBs and topic management.

The second problem is that the SecServ is required to know the current subscription state of each topic to select the required set of public keys for encryption of the topic keys. Keeping the subscription state can be avoided by using the ACL as source for the set $\mathcal{C}_t$. On the other hand, this could lead to unnecessary key transmissions and would prevent wildcards from being used in the ACLs.

D. Topic Key Update: a Pull Approach

Alternatively, a pull mechanism can be used for key update notification which does not suffer from the drawbacks of the push mechanism. For this we use the topic-based pub/sub communication only to advertise the key update event, but not to publish the actual keys. The SecServ publishes a simple unencrypted notification message to topic t', and the clients are responsible for requesting a new key upon reception of the key update notification message.

The procedure of notification and subsequent key retrieval request is shown in Figure 3. The update notification is published by the SecServ to the DB and forwarded via DNs to the clients (step 1). The clients then send a signed message to the SecServ via their DN to request the new topic keys (step 2). The SecServ sends the new topic keys encrypted with the respective public key to the clients (step 3). For the sake of readability, requests of DBs and DNs to retrieve K_t^{auth} are omitted in the figure.

The key retrieval process is similar to the topic join process described in Section II.

Compared to the push approach, the pull method needs additional messages (notification and retrieval request). On the other hand, the pull method helps to avoid unnecessary key transmissions, and the SecServ does not need to keep track

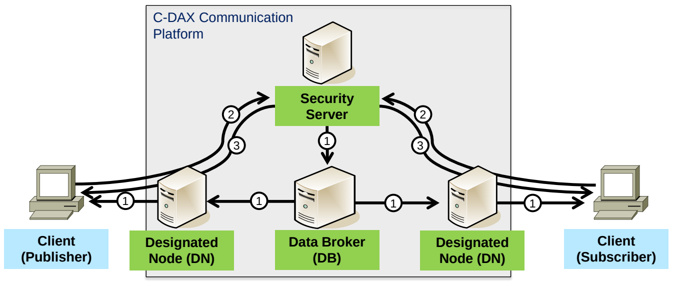

Fig. 3. Update notification and key retrieval (Pull mechanism).

of the current subscription state. However, this optimization of the SecServ comes with an increased risk of denial of service (DoS) attacks because by providing a mechanism to actively retrieve topic keys from the SecServ, clients have the opportunity to trigger expensive asymmetric encryption operations at the SecServ by sending multiple key retrieval requests. To prevent clients from overloading the SecServ with unnecessary key retrieval requests, DNs may cache the encrypted topic keys for the clients they are serving. Any subsequent key retrieval requests for the same client and topic can be handled by the DN without involving the SecServ while the key is valid. DNs can remove the old cached key should they receive an unencrypted key update notification from the SecServ, and cache the new key during the topic key retrieval process of a client.

E. Key Update Triggers

Key updates can be scheduled periodically by configuring a key lifetime and replacing it after expiration. Key updates can also be triggered by join or leave events and ACL changes. The advantage of periodical key updates is that there is no means to attack the SecServ by intentionally causing key updates. However, topics with low fluctuation in the set of publishers and subscribers benefit from event-triggered key updates because unnecessary key updates can be avoided.

F. Key Transitions

We need a mechanism for a seamless transition because the simultaneous replacement of $K_t^{*,i}$ by $K_t^{*,i+1}$ at all involved parties is impossible. Therefore, subscribers need to preserve the outdated key $K_t^{*,i}$ for a short time after the key update. If publishers switch to the new keys with a small additional delay and include an identification number like the index i of the key used into the message, the transition $K_t^{*,i} \rightarrow K_t^{*,i+1}$ can be performed without the risk of delivering messages to subscribers that are not yet or no longer in possession of the required key.

V. Related Work

The SeDAX [11] architecture uses geographical routing on a Delaunay-triangulated (DT) overlay network to forward messages to the responsible broker. The resilient end-to-end message protection framework (REMP) [8] of SeDAX uses long-term keys for each participating overlay node which are assigned during node authentication. The actual end-to-end communication between publishers and subscribers in SeDAX is protected using symmetric encryption with diversified keys based on the long-term node keys.

Nabeel et al. [12] propose a privacy-preserving context-based pub/sub system based on a modified Paillier cryptosystem and a group key management scheme. Subscriptions and content notifications are blinded in the system so that only legitimate brokers are able to match publishers to subscribers. The authors implemented their approach on the ActiveMQ pub/sub system and provide practical results showing the efficiency of their approach for a simple test setup. However, the approach may not be suitable for large data volumes and large number of publishers and subscribers because of the limited message routing efficiency as stated by the authors.

Barenghi and Pelosi [13] discuss general security and privacy challenges in smart grid infrastructures. They first analyze potential interactions of smart grid actors in the future smart grid and then address the identified challenges by giving recommendations to existing solutions. Their general recommendation is to use symmetric and asymmetric encryption, and secure hashing to protect information flows in the smart grid.

Wu and Zhou [14] propose a fault-tolerant and scalable key management system for smart grid communication based on a public key infrastructure and the Needham-Schroeder authentication protocol. Symmetric encryption is used for data communication while asymmetric encryption is used for key distribution. In contrast to our approach, this scheme generates a new session key for any single triggered message flow event, e.g., a data refresh timeout.

Long, Tipper, and Qian [15] propose a key management scheme for smart grid communications based on Iolus [16]. It is an hierarchical approach aimed at reducing the impact of triggered key updates by limiting them to the subtree where the group change has occurred.

VI. Conclusion

In this paper, we proposed a security architecture for the C-DAX middleware.. We defined the security properties *source authentication*, *topic access control*, *end-to-end integrity*, and *end-to-end confidentiality* for C-DAX and presented the mechanisms used to enforce them. We described how keys are initially distributed and how they are updated either in regular intervals or as a response to topic joins and leaves. A subset of this architecture is already implemented in the C-DAX prototype and is used in a field trial to securely exchange phasor measurement data in the Alliander LiveLab [17] smart grid test site.

Acknowledgment

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7-ICT-2011-8 under grant agreement n° 318708 (C-DAX). The authors alone are responsible for the content of this paper.

References

[1] W. K. Chai, N. Wang, K. V. Katsaros, G. Kamel, S. Melis, M. Hoefling, B. Vieira, P. Romano, S. Sarri, T. Tesfay, B. Yang, F. Heimgaertner, M. Pignati, M. Paolone, M. Menth, G. Pavlou, E. Poll, M. Mampaey, H. Bontius, and C. Develder, "An Information-Centric Communication Infrastructure for Real-Time State Estimation of Active Distribution Networks," *IEEE Transactions on Smart Grid*, to appear.

[2] M. Hoefling, F. Heimgaertner, B. Litfinski, and M. Menth, "A Perspective on the Future Retail Energy Market," in *Workshop on Demand Modeling and Quantitative Analysis of Future Generation Energy Networks and Energy Efficient Systems (FGENET)*, Mar. 2014.

[3] C-DAX Consortium, "Cyber-secure Data And Control Cloud for Power Grids," 2014. [Online]. Available: http://www.cdax.eu/

[4] B. Vieira and E. Poll, "A Security Protocol for Information-centric Networking in Smart Grids," in *ACM Workshop on Smart Energy Grid Security (SEGS)*, Nov. 2013.

[5] M. Hoefling, F. Heimgaertner, M. Menth, K. V. Katsaros, P. Romano, L. Zanni, and G. Kamel, "Enabling Resilient Smart Grid Communication over the Information-Centric C-DAX Middleware," in *ITG/GI International Conference on Networked Systems (NetSys)*, Cottbus, Germany, Mar. 2015.

[6] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The Many Faces of Publish/Subscribe," *ACM Computing Surveys*, vol. 35, no. 2, pp. 114 – 131, 2003.

[7] P. Bellavista, A. Corradi, and A. Reale, "Quality of Service in Wide Scale Publish-Subscribe Systems," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1591–1616, 2014.

[8] Y.-J. Kim, V. Kolesnikov, H. Kim, and M. Thottan, "Resilient End-to-End Message Protection for Large-scale Cyber-Physical System Communications," in *IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Nov. 2012.

[9] M. Steiner, G. Tsudik, and M. Waidner, "CLIQUES: A New Approach to Group Key Agreement," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, May 1998.

[10] Y. Kim, A. Perrig, and G. Tsudik, "Communication-Efficient Group Key Agreement," in *IFIP Conference on Information Security (IFIP/Sec'01)*, Jun. 2001.

[11] Y.-J. Kim, J. Lee, G. Atkinson, H. Kim, and M. Thottan, "SeDAX: A Scalable, Resilient, and Secure Platform for Smart Grid Communications," *IEEE JSAC*, vol. 30, no. 6, 2012.

[12] M. Nabeel, S. Appel, E. Bertino, and A. P. Buchmann, "Privacy Preserving Context Aware Publish Subscribe Systems," in *International Conference on Network and System Security (NSS)*, Jun. 2013.

[13] A. Barenghi and G. Pelosi, "Security and Privacy in Smart Grid Infrastructures," in *International Workshop on Database and Expert Systems Applications*, Aug. 2011.

[14] D. Wu and C. Zhou, "Fault-Tolerant and Scalable Key Management for Smart Grid," *IEEE Transactions on Smart Grid*, vol. 2, no. 2, pp. 375–381, Jun. 2011.

[15] X. Long, D. Tipper, and Y. Qian, "An advanced key management scheme for secure smart grid communications," in *IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Oct. 2013.

[16] S. Mittra, "Iolus: A Framework for Scalable Secure Multicasting," in *ACM SIGCOMM*, 1997.

[17] Alliander N.V., "LiveLab," 2015. [Online]. Available: https://www.alliander.com/en/innovation/our-innovations