Grant Agreement Number: **248113/O70**

Project acronym: **IoTSec**

AMI Security Analysis

D4.3.1

Due delivery date: M18

Actual delivery date: M35

Organization name of lead participant for this deliverable:

NR

Dissemination level		
PU	Public	**X**
RE	Restricted to a group specified by the consortium	
CO	Confidential, only for members of the consortium	

Deliverable number:	D 4.3.1
Deliverable responsible:	NR
Work package:	WP4
Editor(s):	Sigurd Eskeland and Åsmund Skomedal

Author(s)	
Name	**Organisation**
Sigurd Eskeland	NR
Åsmund Skomedal	NR

Document Revision History			
Version	**Date**	**Modifications Introduced**	
		Modification Reason	**Modified by**
1.0	27/9-2018	First version of the document	

1 ABSTRACT

This task addresses a security evaluation of the Kamstrup AMS that was carried out in conjunction with the IoTSec project. This white paper contains a high-level overview of the Kamstrup Omnia suite, scope and method considerations that pertains to a security protocol analysis of the same AMS. Relevant security properties and security requirements are discussed.

I. TABLE OF CONTENTS

II. TABLE OF FIGURES AND TABLES

2 INTRODUCTION

The IoTSec project is a research project funded by the Norwegian Research council. One of the activities in this project is to contribute towards the establishment of relevant services that are to be offered by, or channelled via, the Smart Grid Security Centre (SGSC) in Halden, Norway. This centre plans to offer competence and knowledge to smart grid projects and activities in both the private and public sector. Distribution system operators (DSO), advanced metering systems (AMS) and advanced meter infrastructure (AMI) providers are potential clients of the centre.

The purposes of this activity in IoTSec project are as follows:

- White paper containing an approach and method for a security analysis of an existing AMS [This document: Open]
- Establish a security model of relevant parts of an existing AMS [NR domain: 'Strengt Fortrolig']

The secondary goals of this activity are to:

- Using the proposed evaluation approach, evaluate the security of relevant parts of an existing AMS [NR domain: 'Strengt Fortrolig']
- Collect relevant metrics from the evaluation process [Project domain: Confidential]
- If appropriate propose security measures for the case study [NR domain: 'Strengt Fortrolig']

This case study is in part based on documentation that NR has received from Norgesnett under a strict confidentiality agreement, and which is classified according to NR's security classification policy as 'Strengt Fortrolig' ('Secret'). Part of this project activity is to establish a security model of an existing AMS, which provides a proper and natural case for employing the proposed evaluation approach.

AMS infrastructures are subject to a wide range of potential threats that pertain at the various parts and areas of such an infrastructure. This project activity is confined to AMS communication security, in particular the security protocol[1] of an existing AMS.

To design a security system requires that relevant security threats are identified. Pertaining security requirements need to be formulated, and security measures implemented and tested in order for that system to be secure with respect to the identified threats.

Likewise, to carry out a security analysis on a model requires that relevant security threats are identified and then security requirements that provide protection against those threats need to be identified. Alternatively, the analysis could start with identifying a set of relevant security requirements. The analysis needs to show whether each security requirement is assured by the system. If so, then the system is 'secure'.

[1] A "protocol" specifies an exact sequence and order of messages that is to be exchanged between two (or more) entities of a communication system. A security protocol is used to obtain secure communication in insecure networks, and it is a communication protocol that performs a security-related function and applies cryptographic methods in order to achieve the desired security.

The proposed evaluation method requires a model of the security protocol to be established. Security protocols utilize cryptographic mechanisms, and in a model like this, such mechanisms are regarded as primitives that have ideal security properties. For example, resistance to "known ciphertext attacks" relates to symmetric encryption algorithms and assures that given a set of ciphertexts it is not possible to find the plaintexts or the secret key that was used for computing those ciphertexts. Therefore, this type of analysis does not cover considerations of concrete implementations of cryptographic algorithms, such as types of hash functions or key lengths. Hence, details of concrete cryptographic algorithms are open for implementation considerations.

2.1 Scope of the evaluation

'Security' is a broad concept, which involves everything from software vulnerabilities to protecting data and information. The scope of this report includes those aspects that are related to protecting data - in particular data in transmission. The part of a communication system that has security measures for protecting data in transmission is at the design level called a security protocol or cryptographic protocol. These are protocols, which usually have a single security purpose such as secure key establishment or user authentication. Application protocols with security measures is a communication design that has multiple security functions and are more comprehensive than security protocols. The scope of this project activity is confined to an AMS security protocol.

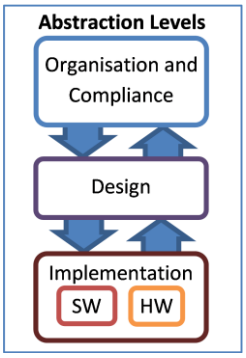

Figure 1. System abstraction levels

As shown in Figure 1, there are at least three levels of abstraction that are intuitively separable. These levels of abstraction are:

1) Security requirements. High-level textual requirements (organisation level).
2) Design. Design-level requirements corresponding to the high-level requirements. Logical and structural design of security measures
3) Implementation. Concrete code and operational systems

To reduce ambiguities and encourage consistency throughout the system development process, it will be necessary to 'translate' high-level (textual) security requirements to a model's level of abstraction. To prepare for this analysis method we need to identify the high-level security requirements, and then refine and translate them in terms of design level requirements.

A generic AMS is assumed to have four types of entities:

i) A key production environment (KPE)
ii) A centralized head-end system (HES) at the front end at the DSO
iii) Concentrators that forward measurement data and instructions between the HES and smart meters
iv) Smart meters

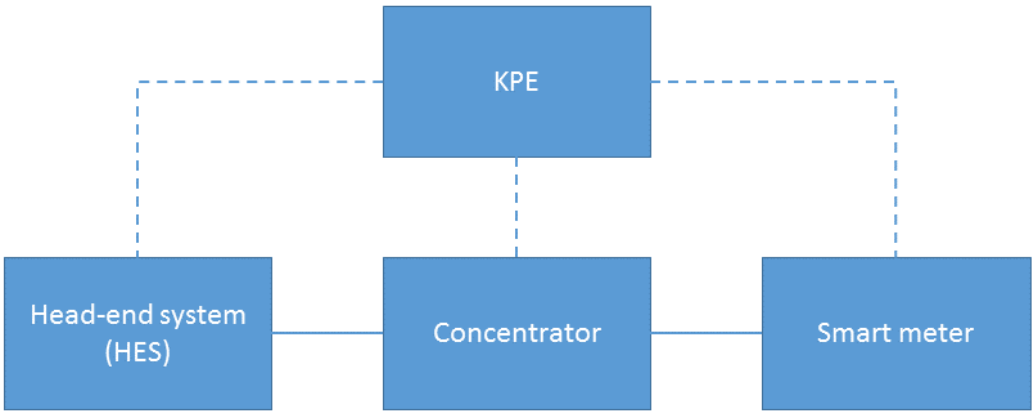

Figure 2. AMS modelling overview

The overall security requirements at AMS system level are:

- New smart meters and concentrators shall be authenticated and authorised system members
- Messages containing data and instructions to and from meters and concentrators shall resist
  - Breach of confidentiality
  - Breach of integrity
  - Masquerade/spoofing attacks

The scope of this evaluation is limited to the meters' connectivity with the DSO. The applications of the homeowners that are connected via the Home Area Network (HAN) interface are outside the scope of this analysis.

2.2 AMS security analysis

The overall security analysis process is conducted in part with the Prosa toolset. However, analysing security protocols involve a number of aspects that present intricate subtleties that may be hard to identify for a given security protocol. Although there exist tools that can carry out automatic analysis, they are confined to small cryptographic protocols, which means that such tools are not fit comprehensive application protocols with security measures, such as an AMS system protocol. In this case study we have a complex security design that Prosa may be suitable to model.

The Prosa toolset provides a semi-automated security analysis of such systems. The cornerstone of the security analysis method is to establish a precise model of those parts of the system that procure secure communication.

The analysis process contains five activities as shown below.

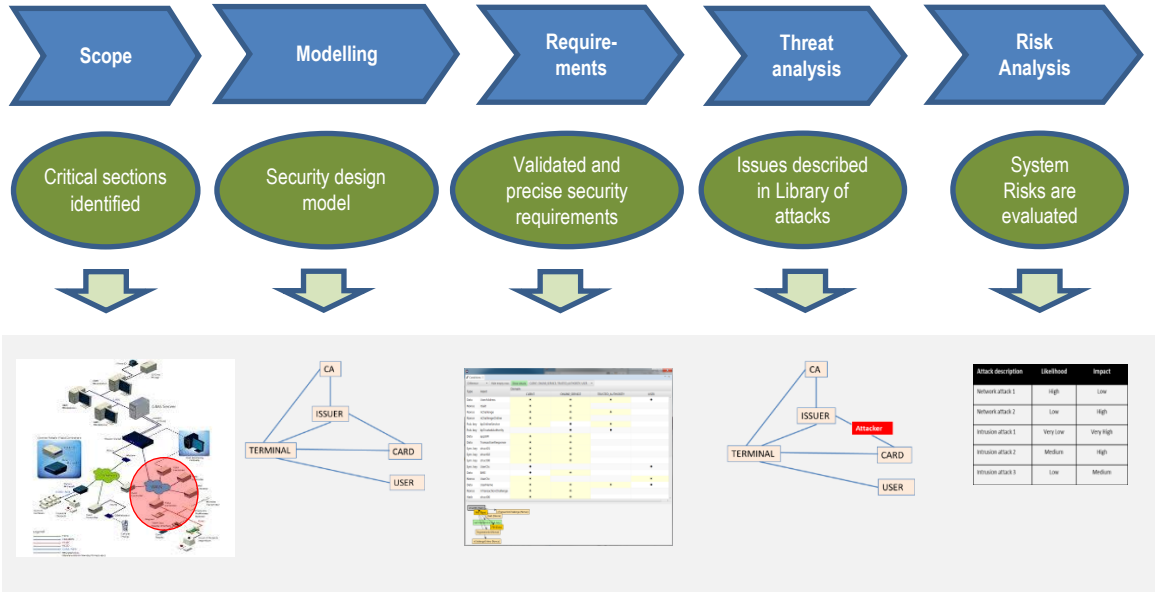

Figure 3. Overall design and analysis process

Scope. The purpose of this initial activity is to identify the part of the system to be analysed, which is, in particular, the part of a system that involves and establishes secure communication.

Modelling. This activity results in producing a model of the protocol design. A model includes pertaining entities[2]; messages containing plaintexts, ciphertexts, hashes, or other types of outputs from cryptographic operations; cryptographic elements such as symmetric keys, asymmetric keys (public/private key pairs), nonces, timestamps, and so on; and the relationship (or accessibility) between each entity and cryptographic data elements. Prosa has its own modelling language (which include syntax and semantics) and by which security models are formulated.

A range of visualisation tools renders various aspects of a particular security design in different 'views'. A frequently used view is the 'interaction view', which is a flexible version of traditional message sequence diagrams. Another is the 'protection view', which is a multi-layout version of a traditional key hierarchy.

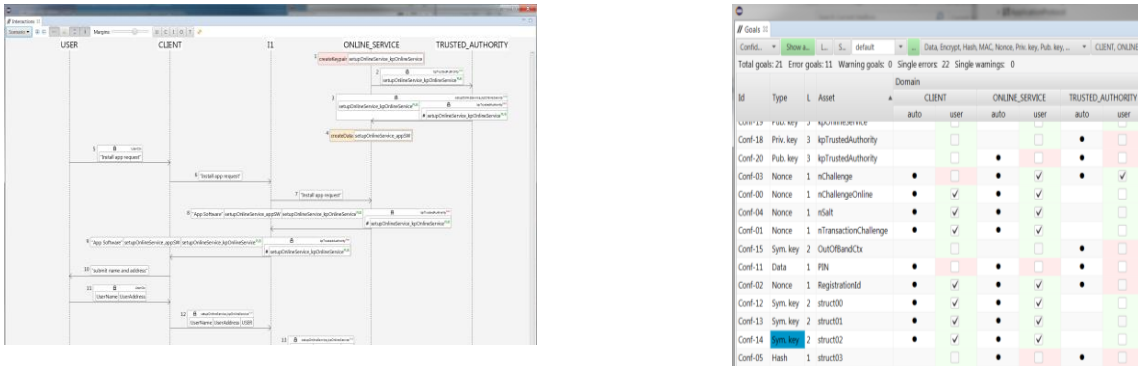

Figure 4. Message sequence diagram and security requirements specification

[2] In Prosa, an 'entity' is referred to as an 'agent'.

Security-model consistency check. In a design process, the designer must initially identify relevant security threats, whereupon he or she then needs to put forward high-level security requirements. The designer must then translate the high-level security requirements into the model's level of abstraction; by making a design whose security measures are in agreement with those threats and requirements. The overall goal is that the security model provides protection against the given threats.

To prepare for this analysis method, we therefore need to identify relevant high-level security requirements and refine them, and have them translated in terms of design-level requirements that comply to pertaining cryptographic data elements (such as symmetric keys, asymmetric keys (public/private key pairs), nonces, timestamps, and so on), and their relations to pertaining entities.

The translation into model-level requirements is a manual task. For confidentiality and integrity, model-level security requirements are specified as relationships between pairs of entities and cryptographic data elements. This is formulated in Prosa by means of a requirement table. For each data element that is subject to a requirement one 'ticks' which entities that are allowed to process, i.e., to access a given data element with regard to confidentiality and integrity.

At the model level, confidentiality is therefore expressed as rules that indicate what specific entities that are allowed to obtain the value of specific cryptographic data element at the end of the protocol. This is somewhat synonymous to 'read' access and is in a 'positivistic' manner, meaning that no entity should obtain that element value other than those entities that are specified in the requirement table. At the overall high level, the security evaluation should thus consider whether confidentiality protection is assured or not.

Integrity in the Prosa context means that a given entity is the originator of a cryptographic data element, and in that regard 'wrote' that element. An element in the requirement table that is 'checked off' as integrity protected, is a statement that there is an intension that this element should have assurance against adversarial modification during the protocol run. The analysis must show whether this holds or not. At the overall high level, the security evaluation should consider whether integrity protection is assured or not in the model.

The consistency check performed by the tool will then automatically compare the specified requirements with the actual properties of the security design. The discrepancies are then presented with color-coded fields in tables. The tables can be tailored and filtered with respect to the types of data elements that are displayed.

Threat analysis. After the 'baseline' security model has been established one investigates how the security design handles various threat events. Here, the security is analysed with regard to whether the requirements are preserved or not and in conjunction with threats. These are typically from one of the following categories:

- Passive attacks (eavesdropping attacks)
- Active attacks (man-in-the-middle attacks)
- Compromised entities

Parts of the threat analysis must be carried out manually. Those parts of the threat analysis that are done by using Prosa, are realized by designing concrete attack scenarios or 'attack models' for each potential threat event that is considered. Each attack model can be analysed with the Prosa tool during the consistency check, thus showing detailed properties of (partially) compromised system that includes the attacking entities. Hence, a collection of attack models is created and this 'catalogue of threats' can then be used as input to a subsequent risk assessment.

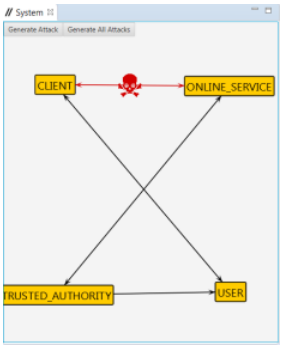

Figure 5. System view

Risk assessment. This is a normal qualitative risk assessment with likelihood/impact considerations for each threat in the threats catalogue. An important part of this process is to concretise the risk policy by setting security risk acceptance criteria, i.e., defining which risks are acceptable and which are not. Principles for risk treatment should be established, taking into account that not all risks can be treated with additional cryptographic measures, but that manual procedures should be considered. A traditional risk level matrix of 5-by-5 categories of likelihood and impact are normally used in the assessment of each threat event.

RISK LEVEL	Level of Impact				
Likelihood (event occurs)	Very low	Low	Moderate	High	Very High
Very high	Very low	Low	Moderate	High	Very High
High	Very low	Low	Moderate	High	Very High
Moderate	Very low	Low	Moderate	Moderate *)	High
Low	Very low	Low	Low	Low **)	Moderate *)
Very low	Very low	Very low	Very low	Low **)	Low **)

Figure 6. Security risk assessment table example

3 OMNIA

3.1 Omnia suite

The smart meters offered by Kamstrup are part of the Omnia suite. The system consists of four main components:

- Omnipower: Includes smart meters for various purposes such as electricity
- Omnicon: Concentrators
- Omnisoft: System software
- Omniserve: Includes products and for service and email and telephone support

The concentrators are part of the infrastructure that brings data back and forth between the head-end system (HES) and the smart meters. The communication is normally wireless, but can be configured for various communication infrastructures. The concentrators and smart meters support radio mesh network and point-to-point configurations. The former is the “default” and latter is used in areas with low population densities, whereof smart meters are normally equipped with modems for the 2G/4G mobile phone network.

Omnisoft includes software such as the UtiliDriver head-end software, VisionAir meter data management (MDM), Network Manager, and the Key Management Service (KMS) programs.

The UtiliDriver head-end software handles all communication technologies and meter types, and receives the meter data such as readings, events and alarms. When new meters are installed, the pertaining concentrators inform the UtiliDriver about the new meter.

VisionAir is a meter data management (MDM) system software that handles management of smart meters, and storage of meter data. It works with other external business system software such as billing and customer management. It consists of a centralized database; a server that interfaces between the mentioned VisionAir database and Omnisoft UtiliDriver, web services for integration with external systems such as billing and customer management; and a file export service.

Network Manager is part of the UtiliDriver software that has two main purposes: 1) It manages the communication networks and the concentrators in the network. It has the capability to show in a geographic map the location of the smart meters (i.e., the “usage points”) and divide them into groups and subgroups. This means that GPS coordinates are loaded into the Network Manager at installation time. 2) The Network Manager helps to manage and monitor the network and shows performances of the connectivity of the wide-area-network (WAN). It carries out meter/concentrator firmware updates, and adds and removes meters.

The Omnisoft Key Management Service (KMS) software is a facility for management and secure storage (in a secure database) of encryption keys that are shared between a smart meter and the central system of the electricity operator, i.e., the HES. Smart meters contain unique cryptographic keys that are preinstalled during the production.

When smart meters are installed at customers during the deployment phase, the HES needs to obtain the same keys in order to communicate securely later on with those exact meters by means of encryption. In the deployment phase, the KMS creates a cryptographic certificate representing a HES, and which is then supplied to the key production environment (KPE) at the Kamstrup production facility. The KPE encrypts relevant smart meter keys using the public key of the HES certificate to ensure that the encryption keys are securely transferred to the HES.

The KMS also generates new keys for key updates.

3.2 Security in the Omnia suite

According to Section 6.8 in the Kamstrup document "Omnia 3.1 Technical Description", the Omnia suite supports the following security measures and requirements:

- End-to-end encryption between meter and head-end system
- Unique encryption keys for each meter
- Data protection by confidentiality, integrity, availability and privacy
- Device authentication
- Replay protection
- Key replacement
- Advanced roles and rights scheme
- Audit logging
- Secure workflows and processes

Note that these security requirements are generic and at a high level, and that is translated to the pertaining level of the model, to specific entities and data elements.

3.3 Case study

The case study is performed on the AMS solution operated by Norgesnett, previously Fredrikstad Energi Nett (FEN). Kamstrup is the AMS provider used by Norgesnett, and via Norgesnett we have been put in contact with Kamstrup.

This case study is in part based on documentation that NR has received from Norgesnett under a strict confidentiality agreement, and which is classified according to NR's security classification policy as 'Strengt Fortrolig' ('Secret'). In particular, Kamstrup has provided some classified and detailed information about their AMS solution, which was necessary to build the security model. This white paper is not classified and is publishable by approval from Norgesnett and Kamstrup, and contains therefore only selected and aggregated information about the particular solution. This includes the following characteristics of the system model and the analysis process:

i) The number of entities
ii) The number of messages
iii) The number of distinct keys
iv) The number of levels in the key hierarchy
v) Considered threats
vi) Security requirements (data confidentiality, integrity, entity authentication)
vii) The principles behind the applied security measures

This type of high-level reporting is common in situations where detailed architectures and design does not have a classification that allows open distribution.

Table 1 shows some security protocol characteristics of the Kamstrup AMS with regard to the scope of the assessment in this project. The considered entities are key production environment (KPE), head-end system (HES), concentrator and smart meter. Each meter is associated a specific concentrator, while each concentrator is associated the HES. All meter/HES communication goes through a concentrator.

It should be noted that the actual number of keys in Kamstrup AMS implementations do not reflect the information in Table 1, which refers to the model. The implementation contains a number of keys that are assigned to the instructions/operations, so that different operations are assigned unique keys. Since the same type of cryptographic measure is used for all such instructions, it is sufficient that this is represented with one generic key in the model. Likewise, just one generic meter, one concentrator and one HES are modelled. In practical realizations, the number of meters and concentrators is of course arbitrary.

Protocol aspect	Number
Entities	4
Security messages	28
Symmetric keys	9
Key pairs	1
Counters	7
Nonces	0
Data elements	2
Phases	3

Table 1. Characteristics of the AMS model

Symmetric keys are shared pairwise: meter/concentrator, concentrator/HES and meter/HES. There is one key pair (a public key and a corresponding private key) modelled representing the HES, as discussed in Section 3.1. Three phases are modelled: A production phase, a deployment phase and an operations phase. All together, the model includes 28 distinct messages.

4 AMS SECURITY EVALUATION

A security threat refers to a potential violation of a security requirement. Threats are imposed by entities that we refer to as adversaries. An attack is when the adversary carries out a malicious act against a system.

In a security design process, relevant threats need to be identified first. The threats dictates what security requirements that must be formulated in order to counter those identified threats. The security requirements then dictates necessary security measures of the security design.

In this section, we present relevant threats and pertaining security requirements. The security considered in this report pertains to communicational threats realized by adversaries having eavesdropping capabilities or capabilities to modify communication by modifying, adding or removing messages. Cryptographic mechanisms are security measures to protect against such threats.

We have not considered threat scenarios that apply directly to the software/hardware/firmware implementation levels, e.g. including implementation issues such as software/hardware/firmware vulnerability exploitations. Nor does it include security issues pertaining directly to the network level, such as denial of service attacks. This should not be confused with communication security with regard to assuring confidentiality and integrity.

4.1 Types of adversaries

The users of security protocols are initially issued long-term keys. A legitimate user is referred to as an insider. A user or an insider that act according to the protocol is honest. However, a malicious insider is an internal adversary that legitimately possesses long-term user keys, and may therefore have an advantage compared to a malicious outsider that does not possess such keys.

Although an internal and external adversary may have the same goal, it could be reasonable to assume that an insider may have greater capability to succeed with an attack than an external adversary.

A *passive* adversary is a malicious outsider has the capability to monitor the communication, and capture and record messages that are sent over the communication link. By keeping a history of previously exchanged messages, such an adversary would look for ways to attack the protocol. In a wireless network, this is a reasonable assumption.

An *active* adversary malicious outsider that is capable to control all or one or more communication lines of the network by replacing, modifying, deleting or inserting messages that are exchanged by the protocol, including replaying old messages. Such an adversary is sometimes referred to as a Dolev-Yao-attacker. The goal of the adversary could, for example, be to enforce establishment of an old key in regard to one or more victim users, or to enforce establishment of a key that is computable by the adversary.

4.2 Threats

In an electronic communication setting, the participants send and receive messages that consist of bit sequences. Messages sent over public computer networks or broadcasted over wireless network can be captured or eavesdropped. Eavesdropping is a passive attack. Typically, the motivation for

such attacks is to obtain some secret data, i.e., confidentiality breach. An entity that controls parts of the network and are capable to modify messages is called an active adversary.

Table 2 lists relevant types of attacks on communication, and maps to respective passive and active adversaries. Note that the motivation of active attacks (integrity breeches) may be to obtain confidential information. Thus, integrity breaches may lead to confidentiality breaches.

Type of attacks	Adversary	Security breaches
Eavesdropping	Passive	Confidentiality breach
Modification attacks	Active	Integrity breach
Replay attacks	Active	Integrity or authentication breach
Reflection attacks	Active	Integrity or authentication breach
Spoofing attacks	Active	Authentication breach
Key compromise	Active	Confidentiality integrity or authentication breach
Denial of service	-	-

Table 2. Protocol attacks and security breaches

Modification attacks is a general category of active attacks. Replay, preplay, and reflection attacks are types of active attacks. A replay attack is when an adversary sends messages that he has recorded from a previous session. Reflection attacks denote when an adversary sends messages back to the principal who sent them.

“Masquerading” is when an adversary imposes as someone else, like a legitimate user. Normally, users are authenticated by means of a secret (a shared symmetric key or a private key), so in authenticated protocols the adversary needs to obtain the long-term secret of the victim. “Key compromise” is when such a secret has been compromised. In two-party cryptographic protocols, a usual assumption is that long-term keys are shared in advance and are not compromised. However, there may in turn be protocol designs where a secret is transferred and that secret could be compromised, which may lead to key compromise.

“Key compromise impersonation” is a specialized type of attacks where an adversary A has obtained user B’s long-term secret, and attempts to masquerade as user C to user A. Such attacks are relevant to protocols with the forward secrecy property.

Notice that denial-of-service-attacks are not relevant here, since such attacks are pertinent to the network level, and not to cryptography.

4.3 Security requirements

Without security measures there is no assurance that received bits on a communication link are authentic, meaning that there is no certainty about the claimed originator (machine or person) or certainty about whether or not they have been intentionally modified during transmission by some adversary. An adversary posing as somebody else could be the real originator of communicated data (masquerading or spoofing attack) or modified them while in transit. Security protocols seek to solve such potential security threats by confidentiality protection and entity authentication. The following lists relevant security requirements:

1. Confidentiality
2. Integrity
3. Data origin authentication (message authentication)

4. Entity authentication
5. (Forward secrecy)

Confidentiality is the assurance that data cannot be viewed by an unauthorised user. This is sometimes referred to as secrecy. It could be pointed out that confidentiality and secrecy should not be confused with privacy, since privacy is linked to protecting persons while confidentiality is related to protecting data. In computer networks, confidentiality measures include in particular cryptographic algorithms and protocols.

Data integrity is the assurance that data are has not been altered and that their consistency is preserved. It is commonly needed to detect bit errors in data communication. Integrity can be realized by hashes and checksums, which adds redundancy of the given data. Note that such mechanisms alone *do not assure integrity in conjunction with an adversary*, since the adversary that may modify data content could easily compute a hash digest that corresponds to the modified data. By replacing the existing hash with the new one, the attack is therefore not detectable.

Data origin authentication or message authentication is the assurance that a message has not been modified (data integrity) and that the receiving party can verify the source of the message. This property is realized by means of message authentication codes (MACs), but can also be realized by authenticated encryption (AE) or digital signatures. Such security measures can provide integrity protection in presence of an adversary due to the use of secret cryptographic keys. Since the adversary does not know the actual keys, he or she is prevented from computing correct authentication codes that correspond to the modified data.

Since secret keys are required to compute MACs, a MAC is a cryptographic binding of the pertinent data and its originator (represented by that key). MAC-authenticated data can only be verified by the recipient in conjunction with the pertaining MAC if the recipient possesses the exact same secret key.

Message authentication does not prevent data modification, but provides detection for whether data has been altered or not. See Definitions 9.76, 9.77 in Handbook of Applied Cryptography, Alfred Menezes et al. Similar to this is non-repudiation (non-deniability), which refers to that the author or sender cannot deny that it was the originator with regard to any party. In contrast, data origin authentication only assures this with regard to the recipient holding the same shared key used for computing the MAC. Non-repudiation is typically provided by digital signatures and is realized by means of asymmetric cryptography.

Entity authentication is a highly important security property in security protocols, and is the assurance that a given entity is involved and currently active in a communication session. In other words, this property gives one participant in the protocol assurance of “liveness” of another participant.

In security protocols where entity authentication is properly implemented and assured, active attacks such as replay attacks are prevented.

Entity authentication is realized by nonces, counters or timestamps. Considering nonces first, this normally involves cryptographic challenge/response-mechanisms, where a user initially selects a random nonce that it sends to the other user. This nonce represents a “challenge”. The second user encrypts the received nonce along with other information such as his identity, and sends this back. The first user checks to see if the encrypted response agree with the challenge. If the nonce was unique (not reused) and no one else holds the encryption key, then the first user has assurance that the second user is authentic, since only this one would be able to compute the cryptographic response.

Entity authentication can be realized by counters in conjunction with encryption. Counter values may not themselves be confidential, but the difficulty for an adversary is the computation of the cryptographically correct ciphertext containing the counter value. This requires that both parties keep track of the last counter values. Timestamps require that the clocks of both parties are synchronized.

Note that there is a transitive relationship between entity authentication, message authentication and integrity. If entity authentication is assured, then message authentication is assured, which again assures integrity. So integrity is assured if entity authentication is assured.

Services	Data integrity protection	Data origin authentication	Non-repudiation	Entity authentication
Hash function	No	No	No	
Public key encryption	No	No	No	Yes*
Symmetric key encryption	Yes	Yes	No	Yes*
MAC	Yes	Yes	No	Yes*
Digital signature	Yes	Yes	Yes	Yes*

Table 3. Security properties that may be provided by cryptographic primitives alone

Table 3 shows a mapping between cryptographic primitive types and basic security properties. The mapping indicates what security properties that may be realized by isolated use of a given function. Notice that such usage assumes the presence of an adversary. For instance, hash functions do not alone provide integrity protection in the presence of an adversary since that adversary may alter the input data and compute the corresponding hash value.

Note that entity authentication requires representations of freshness (nonces, counters, timestamps). To obtain entity authentication requires a cryptographic mechanism that relies on secrets (secret symmetric key or private key) in conjunction with nonces, counters, or timestamps. The asterisk in Table 3 indicates this dependency.

Forward secrecy refers to when long-term key(s) have been compromised. Forward secrecy is achieved if the adversary is not able to obtain session keys of previous sessions using the compromised long-term key(s). Depending on the context, this property may be desirable, but is not critical.

4.4 Modelling and security evaluation in Prosa

The criteria for validation of the method should be as objective as possible, but here it is a challenge that the number of 'samples' is only one. In order to verify the applicability of the Prosa based method the following assumptions and criteria are considered.

The criteria for the overall evaluation of this modelling and analysis methods are the following:

1) Entities (i.e., users, communication points), cryptographic data types and messages are defined
2) Associated requirements are expressed
3) Security primitives deployed in the system are modelled
4) AMS phases (production, deployment, operative) are modelled
5) The key management (key update) operations can be modelled

6) Relevant threats are pertaining security requirements are identified
7) Associated risks are assessed

Results from executing the case study can be summarized as follows:

1) All entity types, cryptographic data types and messages have been modelled in Prosa.
2) Part of the requirements have been expressed in Prosa
3) All security primitives deployed in the system have been modelled
4) AMS phases (production, deployment, operative) have been modelled
5) Key management in the deployment context have been modelled. Key management in the operation phase have been partly modelled due to time constraints.
6) The degree of protection against a passive adversary has been investigated. Protection an active adversary has been partly investigated due to time constraints.

As noted, a passive adversary represents a threat against confidentiality breach. An active adversary represents threats against integrity or authentication breaches.

During these activities, we will collect metrics on the complexity of the actual model, the threats and their assessment.

Design input. As input to this security design process will take the written requirements from several sources and filter out those that are relevant for the design of the cryptographic and related mechanisms deployed in the system. As this case study is based on an existing system we must also model the security measures of the system as faithfully as possible from the given documentation of the actual implementation.

Design Analysis output. The results from this analysis are as described above at two levels. Primarily the results are about the methodology and how it can be applied and adapted to analysis IoT systems and smart grid metering infrastructures in particular.

5 CONCLUSION

This white paper is part of an IoTSec project task and it addresses an approach and method for a security analysis of an existing AMS, in particular with regard to the security protocol design. Relevant threats, security properties and requirements have been formulated and discussed. The main tasks has been to i) establish a model of the security protocols of the Kamstrup AMS, ii) identify relevant threats and security requirements and iii) to carry out a security evaluation to see if the system is secure with regard to the identified security requirements.

Modelling and parts of the evaluation was carried out with the Prosa security tool. All entity types, cryptographic data, messages, security primitives and protocol phases have been modelled. Most of the requirements and key management operations have been modelled, but not all due to time constraints. With these minor limitations the analysis covered the degree of protection against a passive adversary. The degree of protection against active adversaries was investigated in part due to time constraints. The actual security model and the findings from the evaluation are classified and not open documents.

6 APPENDIX

6.1 Types of security protocols

Confidentiality and entity authentication are typical requirements for secure communication. It could be noted that in order to obtain confidentiality, authentication must be assured. This is because if user Alice believes he is communicating with user Bob, which in reality is an adversary posing as Bob, then Alice will reveal to the adversary information that was intended for Bob only. Hence, the confidentiality requirement is broken.

Confidentiality is obtained by encryption. Encryption should not be carried out by long-term keys in order to avoid repetition of identical ciphertexts and so-called "key wearing". If the same plaintext is repetitively encrypted using the same key, then ciphertexts computed by deterministic cryptographic algorithms would be identical. Using session keys prevent key wear and identification of ciphertexts containing identical plaintexts.

Note the distinction between long-term user keys and session keys. A long-term user keys represents the user in a security protocol, and is necessary for user authentication. Protocols providing only user authentication are known as *authentication protocols*. If confidentiality is a security requirement, then establishment of a session key must be carried out while user authentication assured. Protocols providing establishment of session keys are known as *key establishment protocols*.

Two important aspects of key establishment protocols are *key freshness* and *key authentication*:

1) Session keys should be fresh, new and unique for each session. A cryptokey must be unpredictable, meaning it must be random and have high entropy. Session keys may be generated by one user, and then securely transferred to the other user by means of encryption, or they could be derived from secretly shared cryptographic material, such as exchanged nonces, which in that case must be encrypted; or long-term keys in conjunction with a one-way function.
2) A user is authenticated when the other user is assured about the first user's identity, and that he or she currently active in the protocol (liveness).

A third property that is relevant is *key confirmation*, which is assurance to the participants that they share the same session key.

As a side note, asymmetric encryption algorithms are slow and symmetric algorithms are fast. By using an asymmetric algorithm to encrypt a symmetric key, the performance is considerably improved.

6.2 Sketch and guidelines of protocol analysis

The following items characterize the design of a cryptographic protocol:

- entities (or users)
- the employed cryptographic primitives
- cryptographic keys (long-term and short-term keys)
- nonces, counters, timestamps
- message sequences

In this report we will not discuss public key-oriented cryptography due to limited scope.

6.2.1 Motives for using cryptographic primitives

Cryptographic primitives include mainly cryptographic one-way (hash) functions, asymmetric and symmetric cryptographic algorithms. Overall purposes of cryptographic primitives are to:

1) Impose verifiable and secure bindings between exchanged data items by means of cryptographically secure transformations.
2) Cryptographic transformations are based on computational difficulties.

Hash functions are non-linear transformations and have no intrinsic secrets. However, a hash value represents a difficulty of obtaining the corresponding input value if it has high entropy, i.e., it is random or is not predictable. The input size is arbitrary and the output lengths are typically 128 or 256 bits. Uses of hash functions may vary, and are used to in authentication schemes and to compute compact representations of larger data strings.

Symmetric cryptographic algorithms are publicized and have no intrinsic secrets. The difficulty to recover plaintexts from ciphertext is based on the secret encryption key.

6.2.2 Cryptographic binding

Due to the one-way properties of hash-functions and symmetric cryptographic algorithms, they can be used to cryptographically bind two or more data items that may or may not include secret keys. Given a hash function f, its input may consist of two data items so that the output is given by

$$\text{digest} = f(\text{data1},\text{data2})$$

Hence, the digest is a representation of both input items that cryptographically binds both together. If an input is a secret key and a non-secret data item;

$$\text{digest} = f(\text{key},\text{data})$$

the digest represents a binding of the key and data item that can only be verified by those possessing the secret key.

6.2.3 Liveness, freshness, and user authentication

To realize user authentication, note the following assumptions:

1. A symmetric key is only shared by two specific users, e.g., Alice and Bob.
2. Verifiable liveness: Alice authenticates Bob by verifying the liveness of Bob in conjunction with the secret shared key.

Liveness is the assurance that somebody is currently active in the protocol, and is established by means of something that procure uniqueness: Nonces, timestamps or counters. This can be obtained by a challenge/response-mechanism that includes a cryptographic function (hash function or a symmetric cryptographic algorithm), and that outputs a “user authenticity token”. Let’s say Alice wants to authenticate Bob. An important assumption is that Alice and Bob in advance share a (long-term) secret key. Alice sends a challenge to Bob, and expects a response (i.e., a “user authenticity token”) that only Bob should be able to compute. Bob’s liveness at the other end is authentic if Bob’s response to Alice is computed using a cryptographic function, whose inputs are:

1) A specific nonce that he receives from Alice in conjunction with the shared symmetric key.
2) A timestamp in conjunction with the shared symmetric key
3) A counter value in conjunction with the shared symmetric key

Item 1 constitutes a description of a cryptographic challenge/response mechanism. Nonces are randomly generated, and have thus high entropy.

- Definition: High entropy means that if a bit sequence has high entropy, then it is hard to predict the input value of the cryptographic function that caused this value. Random nonces have high entropy since it is not possible to predict randomness.

A random nonce that is generated by Alice cannot be predicted by Bob that receives that nonce. Alice and Bob share a randomly selected secret key (with high entropy). Bob supplies the nonce and the shared key into cryptographic function (a hash function or a symmetric cryptographic algorithm), and sends the result to Alice. Since Alice holds the secret key, only she can check if the "user authenticity token" carries the original challenged nonce.

Due to the secrecy of the shared key, an adversary that may know the nonce value is prevented from computing valid responses. Therefore, the output of the cryptographic function is not predictable, and serves as a "user authenticity token" that assures Alice that Bob on the other end, i.e., the liveness and authenticity of Bob.

Regarding Items 2 and 3, timestamps and counters have low entropy and are therefore predictable, but provide uniqueness as input to the cryptographic function due to their incremental nature. Timestamps and counters assure uniqueness as long as long as it is verified that counters are not reused. Items 2 and 3 do not require explicit challenge messages, since the timestamp or the counter implicitly constitutes the challenge.

Since Alice and Bob share a secret key with high entropy, then the output of the cryptographic function is not predictable. Therefore, it serves as a "user authenticity token" that establishes to the recipient the liveness and authenticity of the sender.

In summary, the measures given in Items 1,2,3 prevent an adversary from:

- Computing valid "user authenticity tokens" due to the secrecy of the shared keys - even if low entropy counter/timestamp values are known to the adversary.
- Replaying former messages due to the uniqueness of the counter/timestamp values.

6.2.4 Session key establishment: Key transfer, key contribution, key derivation

There are several types of session key establishment:

1) Key transfer: The session key is computed by one party and securely transferred to the other party.
2) Key contribution: The session is contributorily established by both parties, and is derived from high entropy random nonces that are securely exchanged by the same parties.
3) Key derivation: The session key is securely derived from a shared secret in conjunction with some material of uniqueness (i.e., a timestamp or a counter).

Notice that key transfer (Item 1) and key contribution (Item 2) have the commonality that both involve transmissions of messages. According to Item 1, securely transfer a session key, and according to Item 2, securely exchange of nonces whereof the session key is derived from.

The principal distinction is that in Item 1, one party generates the session key, while in Item 2 the session key is derived from contributions from both parties. Item 1 has the risk that the key generating party may reuse former key. Item 2 provides better assurance that the session key is unique. Even if Alice reuses former key material, or an adversary posing as Alice reuses former key material by replaying former messages, the session key will be unique as long as Bob provides fresh key material.

Key derivation (Item 3) requires no explicit messages containing key material, since each party keep track of the current counter value. Note the analogue use of cryptographic functions (hash function or a symmetric cryptographic algorithm) for the purposes of user authentication (previous subsection) by establishment of "user authenticity tokens":

$$\text{user\_authenticity\_token} = f(\text{shared\_key},\text{uniqueness})$$

and key derivation (Item 3):

$$\text{session\_key} = f(\text{shared\_key},\text{uniqueness})$$

where "uniqueness" is a nonce, timestamp, or counter value. A usage distinction is the authenticity token is not confidential, and would be sent over the communication channel, while the derived session key is confidential, and would be subsequently used as input to a cryptographic function.