Secure Deployment of SmartGrid Equipment

Nicolai Kuntze and Carsten Rudolph
{nicolai.kuntze|carsten.rudolph}@sit.fraunhofer.de
Fraunhofer Institute for Secure Information Technology - SIT
Darmstadt, Germany

***Abstract*—Introducing the SmartGrid goes in hand with the deployment of a large number of devices and with the establishment of a rather conmplex communication infrastructure. SmartGrid equipment like SmartMeters, other remote sensors, control equipment, or additional sub-stations, all have different security requirements. Common to all of them is that each device needs to be securely identified in the infrastructure and that proper security management needs to be in place.**

To allow for remote management, security functionality has to be incorporated in the design of a device. Furthermore, unique identities and establishment of cryptographic keys for secure communication requires an individual initial configuration of each device. Currently, this initial configuration is a manual operation executed during deployment, often by copying configuration data from memory devices such as USB tokens or SD cards.

This paper introduces a novel and secure approach to support efficient roll-out processes in the domain of SmartGrid equipment providing for a faster and cost efficient scheme using already available and standardized security technologies. The main goal is to totally remove any direct physical interaction with the device during deployment and without any client-specific preconfiguration.

I. Introduction

SmartGrid concepts introduce new IT equipment and new inter-connectivity of this equipment. Examples for units with critical security requirements include SmartMeter gateways or control units of devices for "micro cogeneration" [9] (microCHP) as part of distributed energy resource (DER). These units communicate with other units or central entities via unprotected communication channels. However, obvious security requirements for the communication require to establish secure channels between different units. Establishing security relations usually requires to store individual secrets (i.e. cryptographic keys) on the device. This is either done by direct access through administrators at the client side (perhaps in a specially protected environment), during installation/deployment e.g. via USB tokens, or through client-specific configuration during manufacturing of the unit. All these solutions require physical access by skilled personnel. Such a solution that achieves to establish security relations between units involving physical access to the unit is not always efficient. One example is the deployment of smart meters where this would mean that each smart meter would either need to be individually configured before installation or the (probably not highly qualified) person installing the smart meter needs to execute configuration steps on the meter.

The approach described in this paper should provide a more efficient process for the establishment of security relations. No individual configuration is necessary for the device. Instead, the complete deployment can take place at the location of intended use. Further, no customer data needs to be located in the unit before the automatic initialization. Thus, enabling a direct trust relationship between the unit and the customer's other components only requires off-line registration of the device but no physical interaction with the device.

The following document describes a process for the implementation of such equipment within a machine to machine scenario. In this case, the exemplary goal for the demonstration of the concept is showing the equipment can establish a secure channel to a central unit. Customer secrets that are necessary for this secure channel are transferred onto the equipment during the automatic initialization at the place of deployment and not via a process involving physical interaction, for example by way of a USB stick for the installation. The concept can be used for various machine to machine communication scenarios, for instance a remote measurement device (SmartMeter). So far, in M2M scenarios direct configuration of the units by administrators is the usual approach. To do this, customer-specific secrets enabling a later communication are introduced to the units prior to the actual deployment. This action does not allow for a remote implementation and is thus clearly more expensive. One additional advantage of the presented approach is that it provides a secure identification of a device. Current solutions rely on a combination of insecure hardware identification (e.g. using MAC addresses) and cryptographic keys or other credentials protected by software.

In the following text, the process is explained by way of an example using a hardware trust anchor, the Trusted Platform Module (TPM) and TPM-specific implementation variants are shown. It should be noted that the presented approach can also be implemented with other hardware security solutions (e.g. TrustZone or other secure elements) or without hardware-based security. Nevertheless, the security level of the implementation will depend on the hardware trust anchor and on how it is integrated into the device.

As an example for an embedded system to be configured, this publication uses a secure channel to a SmartMeter Gatewayas the example application. The paper presents a short overview on remote security configuration and trust establishment in section II. An architecture to allow for a secure trust establishment in shown in section III. The underlying life cycle for an embedded device is detailed in section IV. Within the life cycle description the relevant protocols and interactions of the components are included. The publication concludes in section V with a short summary and outlook on application domains.

II. Security Configuration and Trust Establishment

Secure channels are based on cryptographic keys (asymmetric or symmetric) for both sides. Pre-established shared secrets or certified keys are common approaches for channel establishment.The owner needs to establish this data on the device.

The need for efficient and secure solutions is widely recognized. Current solutions are often based on placing public-key certificates on the device (e.g. as proposed by Shemyak [14]). A securely stored certificate can be used as a basis for establishing security relations and to build secure channels. However, there need to be different certificates for each client deploying such devices. Thus, individual client-specific certificates need to be brought onto the device either by the manufacturer or by the client.

Secure identification of devices was proposed for Virtual Private Networks by George and Maunier [8]. VPN applications are very similar to the problem stated. However, in this scheme the main goal is to increase security by combining hardware-based device identification with user identification (e.g. by using smart-cards). Efficient device management is not part of this scheme. Nevertheless, the proposed combination of device identification and user identification should be considered very useful to increase security and can also be applied on top of the protocol introduced in the following sections.

The term *zero configuration* for a configuration process without physically accessing the device has been used for mainly functional issues. Recent work on secure zero configuration has concentrated on ad-hoc communication scenarios and on the question how IP address establishment, DNS, etc. can be secured in such scenarios [1]. Current work in automated trust establishment has be done in the area of network attached devices in the context of Internet gateways [13] and smart devices [12]. The approaches are based on external information like pre-deployed keys in the first case or observation, recommendation and reputation in the latter one. Even as these schemes provide the required ability to configure the involved devices on the basis of the external data, these schemes require some initial trust establishment by the device owner. This interaction creates an administrative overhead and cost on the deployment.

Trust anchors and Trusted Execution Environments (TEE) [3] allow for novel schemes of trust establishment in mobile and embedded environments [11]. On technology broadly available is Trusted Computing (TC) as standardized by the Trusted Computing Group. The Trusted Platform Module (TPM) is hereby the core anchor for trust. This chip incorporates strong asymmetric key cryptography, cryptographic hash functions and a random number generator. Additionally each TPM has a unique key pair whose private key is securely stored on the chip and only the TPM itself can use it for e.g. signing or encryption.

Practical solutions for the problem of a real zero-touch solution for the establishment of security relations have so far not emerged. The protocol presented in the following sections uses a TPM to for a hardware-based pragmatic and cost-efficient way for zero-touch security establishment.

III. Architectural Overview

A SmartMeter device should access an infrastructure via a secure channel (e.g. SSL/TLS) in order to access various services. The relevant components in this scenario are an init server, a configuration server as well as the secure channel terminus in the form of a e.g. VPN gateway or other communication server. The components and their respective function from the perspective of the SmartMeter are presented below. connections.

- **SmartMeter:** The SmartMeter is the center of attention and must be configured accordingly before it is actually put into use. The configuration data consist of the terminus address, the special channel settings as well as the required channel secrets (i.e. cryptographic keys and credentials).
- **Init Server:** The init server is first contacted by the SmartMeter in order to determine the config server. The init server will most probably be provided as a service by the manufacturer of the SmartMeter. It should be noted the in the protocol, the init server will not provide or store any cryptographic keys (except maybe its own private key) or distribute any security-relevant information. Its role is mainly to point the SmartMeter to the right place for initial configuration.
- **Config Server:** The config server is reached via the address given to the SmartMeter by the init server. The main role of the config server is to provide to the SmartMeter the special secure channel configuration data as well as the secrets clearly specific to the equipment and necessary for establishing the secure channel in relation to the terminus server. Actual management and storage of secrets is done by another component, namely the **Endpoint Manager** not directly accesible via the Internet.
- **Terminus Server:** The actual secure channel is established between the SmartMeter and the terminus server. To do this, secrets established with the config server are used. No further contact to the init or config server is needed for actually establishing a secure channel.

IV. The SmartMeter Life cycle

The SmartMeter's life cycle is divided into the following phases: production, customer registration, installation via the user, operation, maintenance, and decommissioning.

A. Production

A SmartMeter needs to use a securely stored asymmetric keys for communication and signatures. As a technical realization of this requirement, a hardware-protected trust anchor in the form of a TPM [15] guaranteeing protection of the key can be used. The TPM already comes with the endoresement key-pair EK_pub and EK_priv. as well as with an optional EK certificate.At the end of the production, the hardware

supplier attaches a mark to the SmartMeter. This mark contains the equipment's serial number. The fingerprint of EK_{pub} is selected and used later in the registration as well as in the installation. Furthermore, an equipment-specific secret N_{device} is created, delivered separately from the equipment, and stored securely on the SmartMeter. This information can be represented by QR codes [6]. Further, the producer establihes on the device a known URL of the init server as well as the certificates of the init servers. This information is the same for all produced devices and not customer-specific.

B. Registration

The client (e.g. a utility) stores the device's data (e.g. represented by QR codes) in an appropriate database. The data registered have the following tasks:

- EK_{pub}: The fingerprint of the EK_{pub}, i.e. $H(EK_{pub})$, is used for the identification of the device. The EK_{priv} is protected by the TPM. Thus, a challenge-response protocol can be used for identification.
- N_{device}:This secret is stored on the individual SmartMeter and only known to the customer (the owner). N_{device} is used as proof of the config server's authorization enabling it to configure the SmartMeter. A similar approach was presented in [5].

C. Installation

The installation phase is meant to establish a trust relationship between the SmartMeter and the customer's infrastructure. Here a central goal is establishing and securely storing the secure channel secrets on the SmartMeter The protocol for this installation consists of identification of the config server, install inforamtion on the terminus server , establish trust, and establish cryptographic keys. In the first step the SmartMeter is notified by the Init Server to which Config server it should connect. The Config server the actually executes the security configuration. This first creates a trust relationship between the SmartMeter and the Terminus server. This trust is based on the identity of the device as well as its attested software configuration. Once the trust has been established, the channel secrets are transferred onto the SmartMeter and securely saved. The following paragraphs explain this process in more detail.

1) Config Server Identification: The newly delivered and registered SmartMeter is not configured at this point. The device only needs network access and it retrieves information on the config server from the init server. For this, the SmartMeter contacts the URL_{known} and presents $h(EK_{pub})$. The answer consists of the config server's URL. The init server is operated by the producer or service provider who knows the config server allocations to devices. Then, the SmartMeter starts establishing trust with the config server. The protocol for registering the device is in figure 1. Here the following knowledge has been established by the SmartMeter:

- EK: The SmartMeter has an activated TPM with an added endorsement key. The public part of the key is known by the init server and config server. The private part is saved in the TPM and cannot leave it. The EKis a unique identity for the device.

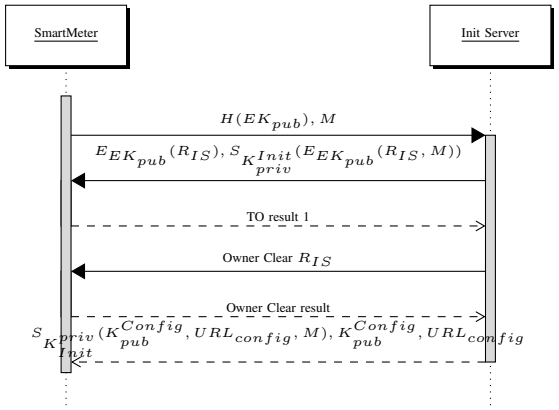

Fig. 2. Handshake for device attestation

- K^{Init}_{pub}: the init server's public key. This key also securely stored in the SmartMeter is optional. However, if this key is not known a signature on the init server's information cannot be verified. An attack on the init server's channel is thus possible.

The init server URL_{known} delivers the address URL_{config} as well as other data (e.g. K^{Config}_{pub}, the public key of the config server) to the SmartMeter as a means of identifying the config servers. The init server knows the SmartMeters and their EK_{pub}s. Thus, it can determine the corresponding config server.

The protocol shown in figure 1describes the interaction between SmartMeter and the init server. Here, the operation $H()$ describes a hash operation like SHA-1 [4], $S_K()$ is a digital signature [7] used by key K. To prove the TPM's authenticity, a TPM_TakeOwnership (Operation $TO()$) can be performed and a corresponding handshake (shown in Figure IV-C1) implemented. It is important to note that the result of the final OwnerClear must also be communicated to the init server. m is a message to the init server as well as a random number that is returned in the answer as proof of the answer's newness. This process can also be used by the init server to check the status of the SmartMeter via remote attestation.

2) Establishing Trust: The config server is now contacted by the SmartMeter. Here the following steps are carried out:

- The SmartMeter transfers $H(EK_{pub})$. It is also possible to transfer the EK certificate.
- The infrastructure checks the certificate and whether a suitable $h(EK_{pub}) = EK_{print}$ was registered. In addition to this, N_{device} is taken from the database.
- The infrastructure creates a TPM owner-secret S_{owner} and saves this in the database.
- An establishment of an OIAP session with the TPM in which TPM_TakeOwnership is carried out using the S_{owner}. The TPM_TakeOwnership's return value can be used to securely identify the TPM.

The result of these steps is a TPM with activated ownership by the customer. All secrets required are only created and stored by the customer. Building on this, an attestation identity key (AIK) [2] must be established. In Figure 3 the protocol for establishing trust is depicted. Here, the role of the config server

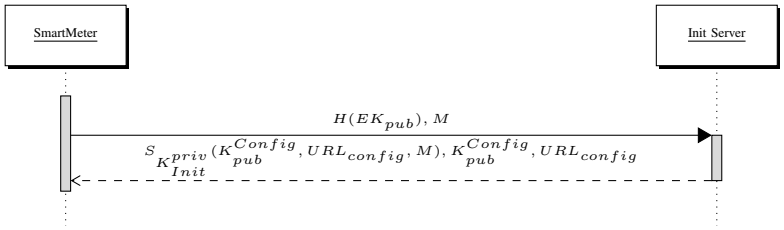

Fig. 1. Protocol for config server identification

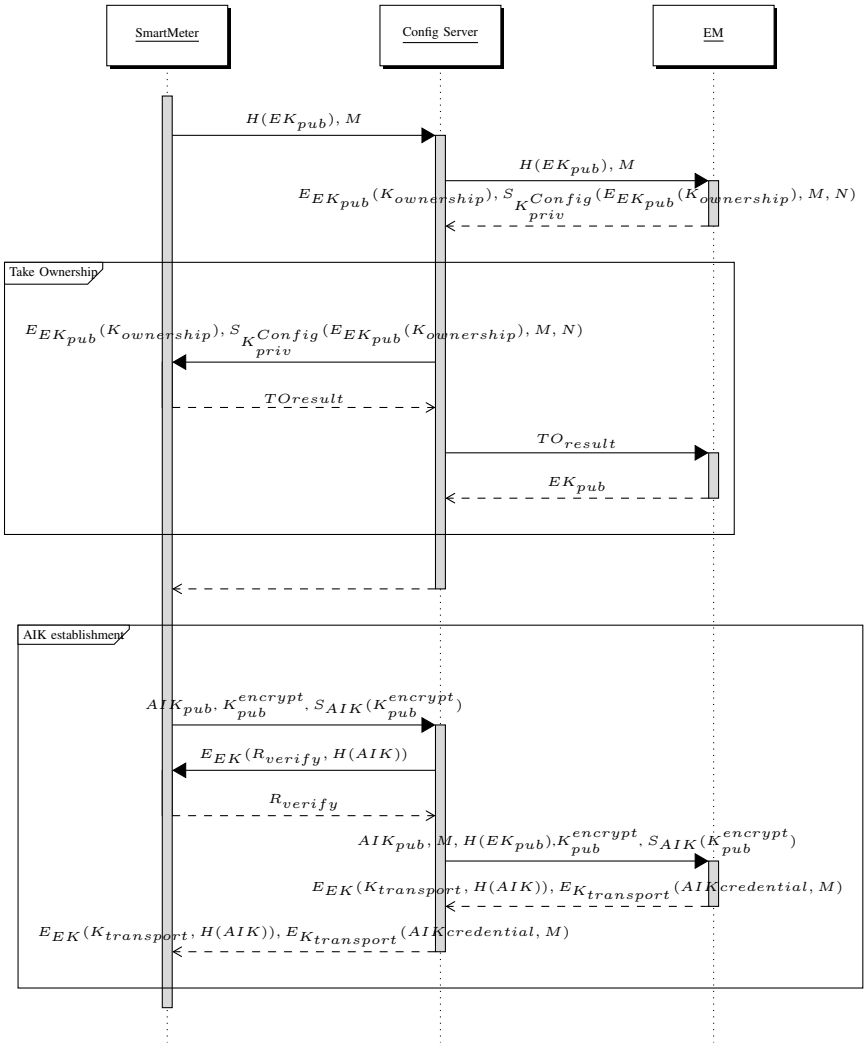

Fig. 3. Terminus trust establishment protocol

is splittet into the actual gateway contacted by the device and an endpoint manager EM. The config server can associate the SmartMeters supplied with their $H(EK_{pub})$s. In doing so, non-legitimized requests can be recognized.

The protocol is divided into the *take ownership* phase and the AIK registration phase. As a part of the preparation, the SmartMeter transfers $h(EK_{pub})$. This key is then used for identifying the equipment. Random number M identifies the session and prevents the replay of messages.

The TakeOwnership process is initiated by transmitting $H(EK_{pub})$ to EM by CS. Due to this request, the EM finds the EK_{pub} in its database and creates a secret $K_{ownership}$ for the TakeOwnership Process. $K_{ownership}$ is then encrypted using EK_{pub}. Furthermore, a signature is created via the encrypted $K_{ownership}$ as well as M and N_{device} using K_{priv}^{Config}. The encrypted $K_{ownership}$ as well as its signature are transferred M and N are checked so that a re-occurrence is not possible and the EM has proven ownership of N. After the signature has been verified, a TPM_TakeOwnership is carried out using $E_{EK_{pub}}(K_{ownership})$. The result of the operation is then sent back to the EM via the config server. The EM checks this result. Thereby it is confirmed that the correct TPM was communicated. The secret $K_{ownership}$ can now become persistent. Finally, the establishment of the AIK begins with the creation

of an AIK on the SmartMeter. Since it is necessary to send a key to the TPM when establishing the secret, a $K^{encrypt}$ is created in the TPM and verified by the AIK. The $K_{pub}^{encrypt}$, AIK as well as the signature $S_{AIK}(K_{pub}^{encrypt})$ are then sent to the config server. To ensure that the communication is actually taking place with the preferred TPM, a handshake first takes place between the config server and the SmartMeter. In the process, the config server creates a random number R_{verify}, encrypts it with the EK_{pub} and sends this to the SmartMeter. Now the TPM_MakeIdentity order is carried out there, returning the R_{verify}, which is then sent to the config server, and proof of the TPM's identity is completed. As soon as the TPM's proven identity has been sent, the config server sends AIK_{pub},M and $H(EK_{pub})$ to the EM, which in turn calculates a corresponding answer package and stores the AIK_{pub} in the database. This answer packet only serves to confirm the successful configuration of the SmartMeter.

3) Secret Establishment and Configuration: On the basis of the trust established, the channel secrets can be transferred to the SmartMeter. To do this, the TPM_Bind ability is used. This ability enables to *bind* data to a specific TPM. The data can only be decrypted by this particular TPM. The previously established AIK is used to certify a key for the bind procedure. The secrets for the use of the SmartMeter are established between the EM and the SmartMeter. The config server is only responsible for forwarding the messages and can thus be hidden in the protocol flow. For the EM the following information is established:

- EK The EM knows the EK_{pub}s of the equipment (for example the SmartMeters) that has been rolled out and with that the identities of the individual SmartMeters.
- N_{device} The N_{device} is sent to the SmartMeter separately for every EK. This is used to prove the EM's legitimation for establishing a secret.
- K_{priv}^{Config} The EM's private key with which the EM signs the answer.
- $K_{channel}$ The EM creates the secure channel secret and leaves this in its database.
- $K_{pub}^{encrypt}$ The $K_{pub}^{encrypt}$ was created by the TPM and enables the EM to encrypt data so that this data can only be released by the particular TPM.
- AIK The SmartMeter's AIK

The protocol for sending the secret to the SmartMeter is simple. First the EM creates a key $K_{channel}$ that meets the SmartMeter's requirements. This key is stored in a database by the EM and is associated with the EK and thus with the particular SmartMeter. Then the $K_{channel}$ is encrypted for the special terminal via the EM by using the established key $K_{pub}^{encrypt}$. A $H(M|N)$ is included in the encrypted data package. It proves that the EM has the N_{device} and that the answer is connected with the original request (identified using M).

The life-cycle is completed with a protocol for decommissioning that uses TPM_RevokeOwnership to clear all secrets.

V. Conclusions

Roll out processes are typically the most costly step in most life cycle designs. Each physical interaction with a particular device requires is more costly then remote operations. The presented scheme allows for a swift deployment without a logistics process for the security aspects aka the USB stick. This remote configuration and its security is rooted in hardware based security already available to the market and deployed in standard PC and server equipment. This guarantees for a cost optimal adaptation for the SmartGrid environment approaching the management support.

On the basis of this scheme various other use cases can be supported as for example the re-assignment of the individual device to a new owner without the need to send a technician. In this publication the chosen example was the SmartMeter but the proposed configuration approach can be utilised also in other infrastructure components like switches, routers, gateways or other sensors.

Future research needs to define the integration of the configuration approach presented into existing and future energy network protocols and processes as well as the support of the technology within the individual SmartMeter architecture.

References

[1] Leung A. and Mitchell C.J. Towards secure zero configuration. In *Proceedings of Western European Workshop on Research in Cryptography (WeWoRC 2005)*, 2005.

[2] E. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 132–145. ACM, 2004.

[3] V. Costan, L. Sarmenta, M. Van Dijk, and S. Devadas. The trusted execution module: Commodity general-purpose trusted computing. *Smart Card Research and Advanced Applications*, pages 133–148, 2008.

[4] D. Eastlake and P. Jones. Us secure hash algorithm 1 (sha1), 2001.

[5] Stephen Hanna. Configuring Security Parameters in Small Devices, July 2002.

[6] H.W. Liu and Y. Yan. Recognition and decoding of qr code. *Jisuanji Gongcheng yu Sheji(Computer Engineering and Design)*, 26(6):1560–1562, 2005.

[7] R. Merkle. A certified digital signature. In *Advances in Cryptology-CRYPTO89 Proceedings*, pages 218–238. Springer, 1990.

[8] George P. and Gérald M. Combining user and platform trust properties to enhance vpn client authentication. In *Security and Management'05*, pages 297–303, 2005.

[9] AD Peacock and M. Newborough. Impact of micro-CHP systems on domestic sector CO2 emissions. *Applied Thermal Engineering*, 25(17-18):2653–2676, 2005.

[10] R. Sailer, X. Zhang, T. Jaeger, and L. Van Doorn. Design and implementation of a tcg-based integrity measurement architecture. In *Proceedings of the 13th conference on USENIX Security Symposium*, volume 13, pages 16–16, 2004.

[11] A.U. Schmidt, N. Kuntze, and M. Kasper. On the deployment of mobile trusted modules. In *Wireless Communications and Networking Conference, 2008. WCNC 2008. IEEE*, pages 3169–3174. IEEE, 2008.

[12] J.M. Seigneur, C.D. Jensen, S. Farrell, E. Gray, and Y. Chen. Towards security auto-configuration for smart appliances. In *Proceedings of the Smart Objects Conference*, volume 2003, 2003.

[13] K. Selén. Upnp security in internet gateway devices. In *TKK T-110.5190 Seminar on Internetworking*, 2006.

[14] K. Shemyak. Secure distribution of the device identity in mobile access network. In *Security and Privacy in Mobile Information and Communication Systems*, Lecture Notes of the Institute for Computer Sciences, pages 117–126, 2010.

[15] N.M. Smith and D.W. Grawrock. Trusted platform module apparatus, systems, and methods, March 30 2005. US Patent App. 11/094,840.