On the Automatic Establishment of Security Relations for Devices

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

***Abstract*—The deployment of devices at remote or distributed locations is a typical scenario e.g. in large enterprise networks or industrial applications. In the deployment of a device identification of the device and the establishment of security relations (*trust*) between the device and other elements of the infrastructure is crucial. Usually, the process requires either to pre-configure the device or to let administrators physically access the device for configuration. Both options induce costs. For functional configurations and software distribution *zero configuration* solutions are available. One important step hereby is the establishment of trust into the individual device by the device owner. This trust establishment requires in typical schemes a large organizational involvement of the device owner resp. operator. The approach presented in this paper addresses the step of initial trust establishment.**

I. Introduction

The technical problem discussed in the following, relates to the fully automatic implementation of a unit critical for security. The unit communicates with other units or a central entity via unprotected communication channels. However, security requirements for the communication may require to establish a secure channel between the 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) or through client-specific configuration during manufacturing of the unit. A solution that achieves to establish security relations between units involving physical access to the unit is not always efficient.

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. 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 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.

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 RD.

The need for efficient and secure solutions is widely recognized. Current solutions are often based on placing public-key certificates on RD. 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 client.

The term *zero configuration* for a configuration process without physically accessing the device has been used for mainly

978-3-901882-50-0

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 and smart devices. 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.

Trust anchors and Trusted Execution Environments (TEE) allow for novel schemes of trust establishment in mobile and embedded environments. On technology broadly available is Trusted Computing (TC) as standardized by the Trusted Computing Group. The 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 remote device (RD) 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 an e.g. VPN gateway or other communication server. The components and their respective function are as follows. The **RD** 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). The **Init Server** is first contacted by the RD in order to determine the config server. The init server will most probably be provided as a service by the manufacturer of RD. It should be noted that 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 RD to the right place for initial configuration. The **Config Server** (CS) is reached via the address given to the RD by the init server. The main role of the CS is to provide to the RD 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**(EM) not directly accessible via the Internet. The actual secure channel is established between the RD and the **Terminus Server**. To do this, secrets established with the CS are used. No further contact to the init or CS is needed for actually establishing a secure channel.

IV. The Remote Device Life cycle

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

Production: An RD 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 guaranteeing protection of the key can be used. The TPM already comes with the endorsement key-pair EK_pub and EK_priv. At the end of the production, the hardware supplier attaches a mark to the RD. 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 RD. This information can be represented by QR codes. Further, the producer establishes 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.

Registration: The client stores the RD's data (e.g. represented by QR codes) in an appropriate database. The data registered have the following tasks. The fingerprint of the EK_{pub}, i.e. $H(EK_{pub})$, is used for the identification of RD. The EK_{priv} is protected by the TPM. Thus, a challenge-response protocol can be used for identification. The secret N_{device} is stored on the individual RD and only known to the customer (the owner). N_{device} is used as proof of the CS's authorization enabling it to configure the RD. A similar approach was presented in [3].

Installation: The installation phase is meant to establish a trust relationship between the RD and the customer's infrastructure. Here a central goal is establishing and securely storing the secure channel secrets on the RD The protocol for this installation consists of identification of the CS, install information on the terminus server, establish trust, and establish cryptographic keys. In the first step the RD is notified by the Init Server to which CS it should connect. The CS the actually executes the security configuration. This first creates a trust relationship between the RD 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 RD and securely saved. The following paragraphs explain this process.

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

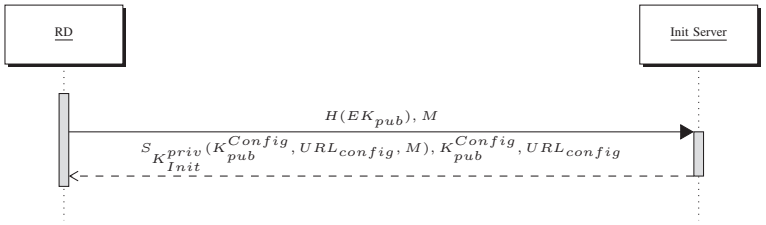

Fig. 1. Protocol for config server identification

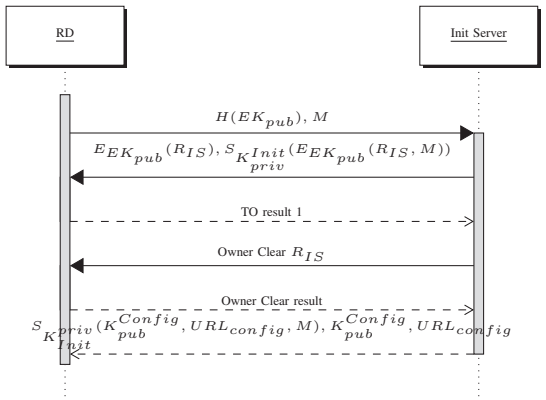

Fig. 2. Handshake for device attestation

established by the RD. The RD has an activated TPM with an added EK. The public part of the key is known by the init server and CS. The private part is saved in the TPM and cannot leave it. The EK is a unique identity for RD. K^{Init}_{pub} the init server's public key. This key also securely stored in the RD 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 CS) to the RD as a means of identifying the CSs. The init server knows the RDs and their EK_{pub}s. Thus, it can determine the corresponding CS.

The protocol shown in figure 1 describes the interaction between RD and the init server. Here, the operation $H()$ describes a hash operation like SHA-1, $S_K()$ is a digital signature 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-1) 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 RD via remote attestation.

2) Establishing Trust: The CS is now contacted by the RD. Here the following steps are carried out. (i) The RD transfers $H(EK_{pub})$. It is also possible to transfer the EK certificate. (ii) 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. (iii) The infrastructure creates a TPM owner-secret S_{owner} and saves this in the database. (iv) 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 CS is split into the actual gateway contacted by the device and an EM. The CS can associate the RM 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 RD 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^{Config}_{priv}. 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 CS. 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 RD. 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 $K^{encrypt}_{pub}$, AIK as well as the signature $S_{AIK}(K^{encrypt}_{pub})$ are then sent to the CS. To ensure that the communication is actually taking place with the preferred TPM, a handshake first takes place between the config server and the RD. In the process, the CS creates a random number R_{verify}, encrypts it with the EK_{pub} and sends this to the RD. Now the TPM_MakeIdentity order is carried out there, returning the R_{verify}, which is then sent to the CS, and proof of the TPM's identity is completed. As soon as the TPM's proven identity has been sent, the CS 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

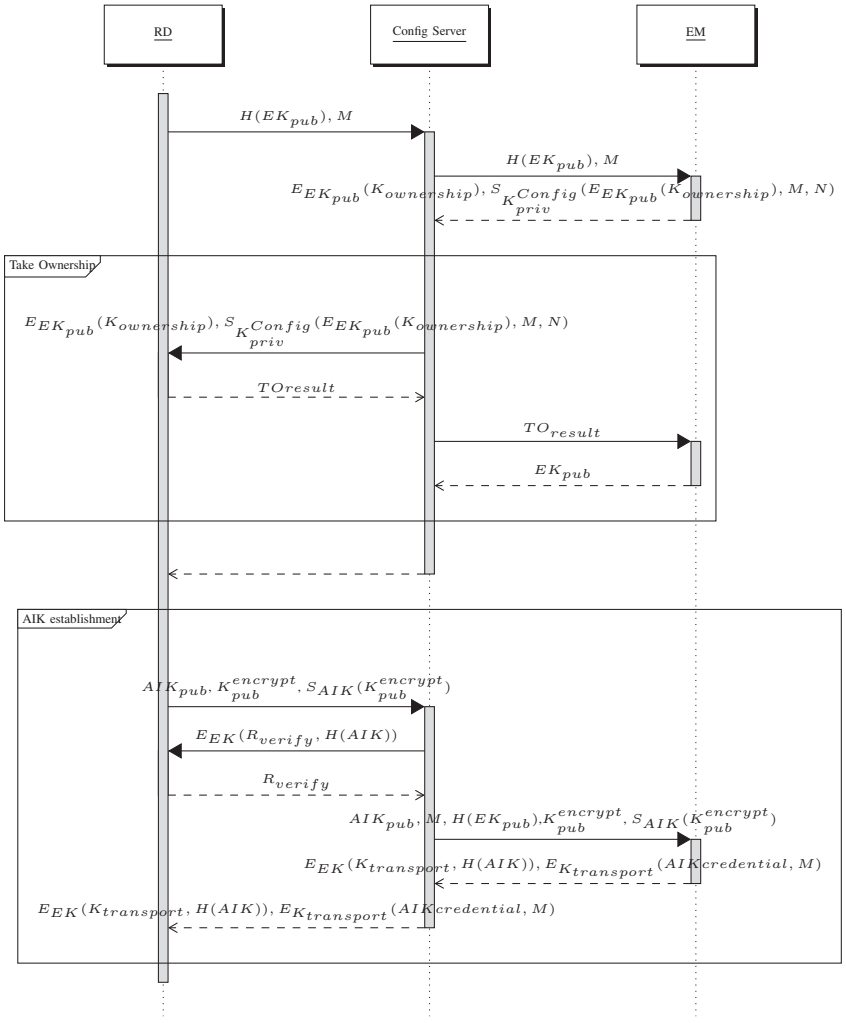

Fig. 3. Terminus trust establishment protocol

only serves to confirm the successful configuration of the RD.

3) Secret Establishment and Configuration: On the basis of the trust established, the channel secrets can be transferred to the RD. 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 RD are established between the EM and the RD. The CS is only responsible for forwarding the messages and can thus be hidden in the protocol flow. For the EM the following information is established. The EM knows the EK_{pub}**s** of the equipment that has been rolled out and with that the identities of the individual RDs. The N_{device} is sent to the RD separately for every EK. This is used to prove the EM's legitimation for establishing a secret. The EM's private key K^{Config}_{priv} with which the EM signs the answer. The EM creates the secure channel secret $K_{channel}$ and leaves this in its database. The $K^{encrypt}_{pub}$ 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 as the RD's AIK.

The protocol for sending the secret to the RD is simple. First the EM creates a key $K_{channel}$ that meets the RD's requirements. This key is stored in a database by the EM and is associated with the EK and thus with the particular RD. Then the $K_{channel}$ is encrypted for the special terminal via the EM by using the established key $K^{encrypt}_{pub}$. 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).

V. Conclusions

Roll out processes are typically the most costly step in most life cycle designs. The presented protocol and concept allows for an optimised and cost reduced process for initial deployment of devices at remote locations.

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