Security for KNXnet/IP

Daniel Lechner Wolfgang Granzer Wolfgang Kastner *

Automation Systems Group
Institute of Automation
Vienna University of Technology

Treitlstraße 1-3, A-1040 Vienna, Austria
{dlechner,w,k} @ auto.tuwien.ac.at

With the integration of security-critical applications into KNX installations, a comprehensive security concept is mandatory. Most important, transmitted data have to be secured. Since IP networks as transport medium for KNX communication become of increasing importance, an approach to secure KNXnet/IP is presented in this paper. The main features of this approach are the suitability for embedded microcontrollers and the support for secure unicast and multicast communication.

1 Introduction

The core application area of KNX is environmental control with the traditional building services lighting/shading as well as heating, ventilation, and air conditioning (HVAC). Up to now, other building service types such as safety-critical (e.g., fire alarm systems) and security-critical services (e.g., access control, intrusion alarm systems) have been only realized by dedicated stand-alone systems.

Today, a tighter integration of these formerly separated systems is desirable. However, to integrate security-critical applications into KNX installations, the underlying control network has to be protected against malicious interference. An important step towards a secure KNX system is to secure the communication i.e., all exchanged data within the control network.

KNX networks are typically implemented following a two-tier model [1]. Field networks are home for sensors, actuators, and controllers that interact with the environment and perform measurement and control tasks. These field networks are interconnected by a common backbone where management nodes (e.g., operator workstation, logging server) that require a global view of the entire KNX network are located. Figure 1 shows a typical example of such a topology.

At the field level, robustness and flexibility is most important. Therefore, the KNX network media Twisted-Pair (TP) with all its benefits (e.g., free topology, small network stack footprint) as well as KNX Powerline and KNX RF are well-established. At the backbone level, it is more

*This work was funded by FWF (Österreichischer Fonds zur Förderung der Wissenschaftlichen Forschung; Austrian Science Foundation) under the project P19673.

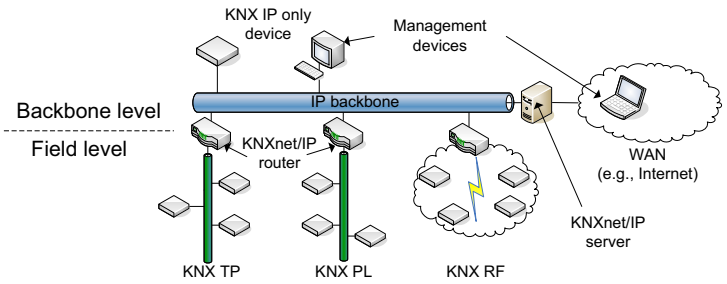


Figure 1: KNX networks

common to use high performance network media. Today, a trend towards the use of IP based networks as backbone can be observed. The main reason is that due to widespread use of IP based networks (especially Ethernet based networks) in the IT world, the cost for cabling and hardware for network interface controllers are rapidly decreasing. Therefore, it becomes economically feasable to extend embedded microcontrollers with a dedicated Ethernet interface chip. Another reason for the popularity of IP based backbones is that many buildings already provide an existing IP infrastructure (e.g., office LAN) which can be shared with the building automation network.

Due to these reasons, KNXnet/IP was introduced. KNXnet/IP provides different mechanisms to encapsulate KNX network messages into IP messages. In addition to these simple encapsulation mechanisms, there are investigations underway that use IP directly as a native KNX medium [2].

With the integration of security-critical applications, the communication within the KNX networks has to be protected against security attacks. Especially the IP backbone is prone to security attacks due to the widespread use of the IP protocol. Unfortunately, the IP protocol itself as well as the underlying network medium (e.g., Ethernet) do not natively provide any security mechanisms. Therefore, in this paper, an approach to secure KNXnet/IP networks is presented. In Section 2, a brief introduction of KNXnet/IP and the available services is given. In Section 3, the security aspects of KNXnet/IP are discussed. Then, an overview of available security mechanisms that may be suitable to secure KNXnet/IP networks is presented (cf. Section 4). Since none of these available mechanisms satisfies all requirements of secure KNXnet/IP networks, a new approach to guarantee secure communication in KNXnet/IP networks is introduced (cf. Section 5). Finally, a proof-of-concept implementation is presented in Section 6.

2 KNXnet/IP

To be able to use IP networks in EIB installations, *EIBnet/IP* was introduced. EIBnet/IP specifies mechanisms to use EIB on top of the IP protocol. With the definition of the KNX specification, EIBnet/IP is now called *KNXnet/IP*.

The KNXnet/IP specification is divided into 8 parts where the parts 2 – 8 specify different services:

- *Core services:* Besides the definition of basic KNXnet/IP, this part includes the specifica-

tion of services for KNXnet/IP device discovery and connection management.
- *Device management:* This part includes services for remote management of KNXnet/IP servers (e.g., read and write of KNXnet/IP specific properties).
- *Tunneling:* KNXnet/IP clients can use the tunneling service to establish a management connection to a KNXnet/IP server.
- *Routing:* KNXnet/IP routers use the KNXnet/IP routing service to forward the network messages between local KNX network segments over the IP network.
- *Remote logging:* This part defines services for offline-monitoring of KNX networks.
- *Remote configuration and diagnosis:* Here, configuration and diagnosis services are defined.
- *Object Server:* In this part, services to access a KNXnet/IP object server are defined. A KNXnet/IP client can use these services to access the communication objects on the KNXnet/IP object server.

In Figure 2, different opportunities to use an IP network in a KNX installation are shown. A KNXnet/IP network consists of *KNXnet/IP servers* and *KNXnet/IP clients*. While a KNXnet/IP server implements the server part of the KNXnet/IP protocol and communicates with other KNXnet/IP servers and/or clients, a KNXnet/IP client implements the KNXnet/IP client protocol and is only able to communicate with KNXnet/IP servers. In the upper left part of the figure, a typical two-tier KNX control network is presented where an IP backbone interconnects various local KNX network segments. At the network boundaries, KNXnet/IP routers are located. In the upper right part, two KNX installations are interconnected through a wide area network (WAN) like the Internet. Again, KNXnet/IP routers are used to provide an interconnection of these two KNX installations. In both cases, a routing protocol is necessary. To be able to exchange KNX messages between the different local KNX network segments, each KNXnet/IP router has to forward the KNX messages to the KNXnet/IP routers located at the same IP network. This is done by using the so called KNXnet/IP routing protocol which is based on multicast.

In the lower part of the figure, two scenarios where a KNXnet/IP client establishes a connection to a KNXnet/IP server are shown. On the left hand side, a management client (e.g., operator workstation) opens a management connection to a KNXnet/IP server to perform configuration and/or maintenance tasks. To establish this unicast connection, the KNXnet/IP tunneling protocol can be used. On the right hand side, a web application accesses a KNXnet/IP object server to retrieve process data (i.e., KNX group objects). The way to access the group objects depends on the protocol that the KNXnet/IP object server supports. This can be, for example, the KNXnet/IP object access protocol or any other proprietary application layer protocol (e.g., an HTTP based protocol like oBIX [3]).

In the current specification of KNXnet/IP, the use of IP as native transport medium is not foreseen – the IP network is only used as simple transport medium. However, there are investigations underway to define IP as native medium for KNX. For more details on the use of IP as native KNX medium refer to [2].

3 Security in KNXnet/IP

As mentioned in Section 1, an important step towards a secure KNX network is to protect the backbone against malicious interference. Especially IP based backbones are prone to attacks. This is for two reasons: First, due to the widespread use of IP networks, IP based protocols

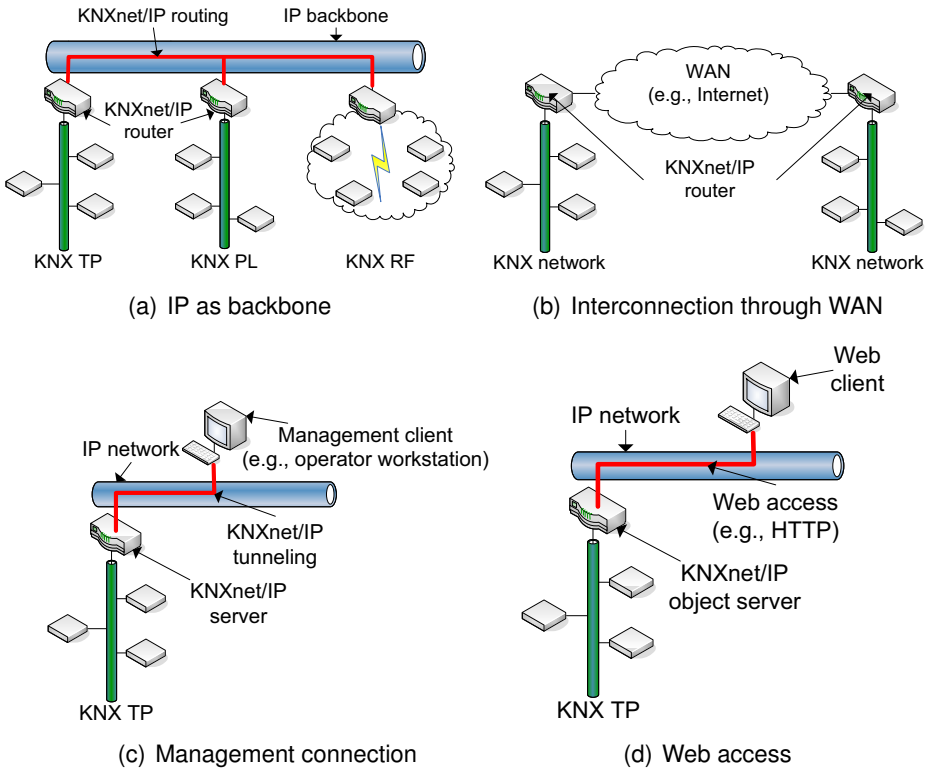


Figure 2: Using IP networks in KNX installations

along with their design flaws and security vulnerabilities are well-known and so, many IP based security attacks exist. Second, the physical access to IP based backbones can be gained more easily since they are often shared with the office IP LAN. Furthermore, it is common to support remote management access (e.g., from the Internet through a web gateway) which can be misused to gain access to the system.

To fully protect the communication within a KNXnet/IP network, different security mechanisms have to be integrated into the KNXnet/IP protocol. Regarding the demands on a secure KNXnet/IP network, the following objectives can be identified:

- *Mutual entity authentication*: To prevent an impersonation of legitimate KNXnet/IP devices, the involved KNXnet/IP servers and clients must prove their identity before they can securely communicate with each other.
- *Secured channel*: To protect the transmitted data that is exchanged between the authenticated communication partners, a secured channel has to be provided. Such a channel uses physical and/or cryptographic techniques to guarantee different security objectives [4]. Depending on the security requirements of the application, these are data integrity, freshness, and/or confidentiality [5].
- *Support for unicast and multicast*: As mentioned above, KNXnet/IP specifies different communication services. The main goal of a security extension for KNXnet/IP is to fully support all of these KNXnet/IP services. While most are based on unicast communication, some services like KNXnet/IP routing and the KNXnet/IP discovery service are based on multicast. Therefore, a security extension has to be used that provides support for unicast and multicast communication.

- *Low processing power and memory consumption*: Providing security using physical techniques (e.g., immuring the network cables) is not always easy to achieve. In such a case, cryptographic techniques have to be used. For reasons of cost efficiency, embedded devices with limited processing power and memory are commonly used in KNX installations. Since cryptographic techniques are computationally intensive, their use must not exceed the available device resources.

Currently, KNXnet/IP does not specify security mechanisms that fulfill the demands of secure communication. In Chapter 1 of the KNXnet/IP specification, various security threats are briefly discussed and some rudimentary countermeasures are presented. Most of these countermeasures are based on an isolation of the KNXnet/IP network (e.g., firewall, KNXnet/IP only Intranet) and on "Security by Obscurity" (e.g., use non-standard multicast address, rely on the missing expertise of an attacker). However, since preventing physical access to the network by isolation is not always possible (e.g., in office LANs, WLANs) and "Security by Obscurity" is a technique that only temporarily provides protection (if at all), these suggested countermeasures cannot fully satisfy the requirements mentioned above.

4 Available IP Security Mechanisms

Due to the widespread use of the Internet, security has been a major research field in the IT world for years. Therefore, many different security mechanisms for IP based networks are available. Each of them is suitable for a certain application field. In this section, the most important IT mechanisms are analyzed with respect to their suitability for KNXnet/IP.

4.1 Internet Protocol Security (IPSec)

Internet Protocol Security (IPsec) [6] is an extension to the IP protocol. IPSec is a part of IPv6 but since IPv4 is still the predominantly used network protocol in the IT world, it has been ported to extend IPv4.

IPSec ensures mutual entity authentication and provides data integrity, freshness, and confidentiality. IPsec defines different cryptographic algorithms for encryption and signature generation. For example, DES, Triple-DES, or AES can be used as encryption algorithm and HMAC-SHA1 for Message Authentication Code (MAC) calculation. For key exchange, the *Internet Key Exchange (IKE)* protocol is used. IKE uses the Diffie-Hellman algorithm to derive the encryption and signature generation keys for IPsec [7]. For IKE, public key (RSA or elliptic curve) or symmetric algorithms can be used.

IPSec is intended to secure both unicast and multicast communication. However, there are some problems when multicast is used in IPsec. If there is more than one sender, the sequence numbers (to avoid replay attacks) have to be synchronized between the senders. Another possibility would be that the receivers have to track one sequence number for each sender which would require a change of the current security association (SA) implementation. Another problem is the key distribution. Since IKE uses the Diffie-Hellman algorithm, it is not easy to derive a common secret key in a group without big adaptions to existing implementations. A solution would be the use of an explicit key server which would result in too much overhead for small networks. Furthermore, this key server would represent a single point of failure.

In principle, IPsec can be used as security extension in KNXnet/IP networks. However, due to the high complexity of the IPSec specification, the demands regarding device resources and the mentioned problems with multicast, IPSec is not advisable for networks containing embedded devices. If more powerful devices are available, IPSec can be used to secure KNXnet/IP tunneling or other KNXnet/IP unicast services. For KNXnet/IP routing, special implementations of IPSec would be required.

4.2 SSL/TLS

Secure Sockets Layer (SSL) and its successor *Transport Layer Security (TLS)* [8] is a protocol developed for securing communication between two parties. During the initial handshake, the used cryptographic algorithms are negotiated, secret keys for the secured channel are exchanged, and entities are authenticated. After this initial handshake, a secured channel between the two communication parties is established. Like IPSec, the initial key exchange during the handshake is usually done using public key algorithms. The secured channel uses symmetric algorithms for guaranteeing data integrity, freshness, and confidentiality.

TLS offers similar algorithms to IPSec but operates on a higher level in the ISO/OSI layer model. It encrypts data between the transport layer (TCP or UDP) and the application layer. Besides the classical utilization of TLS for securing HTTP traffic, there are some techniques to set up VPNs by encrypting whole IP packets.

TLS is very flexible with respect to the use of algorithms. So it is possible to implement secured unicast connections on embedded devices, too. [9] shows an implementation of a complete secure web server, using HTTP and TLS. In this implementation, the asymmetric encryption part of TLS is done with the help of elliptic curve cryptography (ECC) [10]. The web server runs on very constrained devices like a 4MHz 8-bit ATmega processor.

Since TLS is a unicast protocol (asymmetric key exchange is done using unicast algorithms – there are no extensions for multiparty key exchange), it can only be used to secure KNXnet/IP unicast services. It is not usable for multicast environments like KNXnet/IP routing.

4.3 VPN

A *Virtual Private Network (VPN)* is a secure, logical network that is built upon a possible insecure network called host network [11]. A VPN is transparent to the connected devices. Usually, each device opens a secure connection to a centralized server where the whole network traffic to and from the device is tunneled through. A popular VPN implementation is OpenVPN [12].

OpenVPN provides several methods to ensure authentication: a pre-shared symmetric key, a username and password combination, a TLS certificate, or a combination of these methods. While IPSec manipulates the network protocol, OpenVPN encapsulates the encrypted VPN packets into untouched TCP or UDP packets of the host network. However, the VPN packets themselves are secured down to IP layer (Layer 3) in routing mode or down to Ethernet (layer 2) in bridging mode.

As already mentioned, a centralized server controls the connections and each client connects via a secure unicast connection to the server. This means that the server has to keep track of several connections with different keys. The whole traffic within this network is routed through these tunneling connections to the server (cf. Figure 3). Multicast messages are sent to the

server and distributed there. So the server has to decrypt each multicast message once and encrypt it again several times for all other parties. This results in a high demand on bandwidth, memory (for storing the different session keys), and computational power at the server.

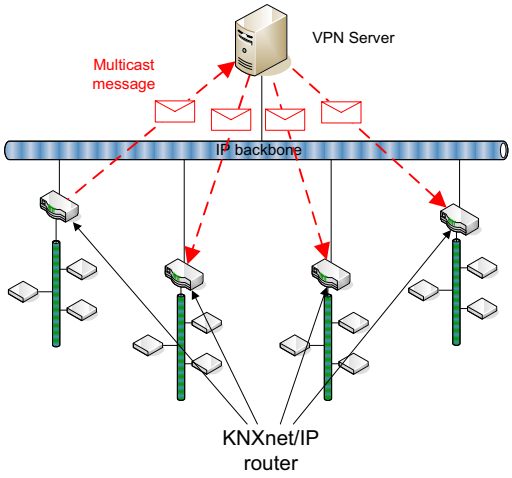


Figure 3: VPN multicast

The main drawback of VPNs is the need for a centralized server. As mentioned above, the whole network traffic is routed through the server. Therefore, the server represents a bottleneck and single point of failure for the whole network especially if multicast is used like in KNXnet/IP routing. Furthermore, the required resources at the server will exceed the capabilities of an embedded device and therefore more powerful devices would have to be used as VPN servers. So, a VPN solution is of limited use for KNXnet/IP.

4.4 Summary

The analysis of proven IT security mechanisms showed that each one has its advantages and disadvantages, but none of them fully meets the requirements of KNXnet/IP. While IPSec is only of limited use for embedded devices and has some problems with multicast, TLS completely lacks multicast support. VPN implementations like OpenVPN have a bad performance in large networks due to conceptual reasons. Table 1 shows the results of the comparison of the analyzed IT security mechanisms.

	Entity authentication	Secured channel	Multicast	Resource consumption
IPSec	+	+	∼[1]	–
SSL/TLS	+	+	–	∼[2]
OpenVPN	+	+	–[3]	–[3]

Table 1: Security protocols

[1] See notes in Section 4.1.
[2] If special encryption-mechanisms like ECC are used.
[3] In case of many clients.

5 Solution

In [13], a security extension called EIBsec for KNX TP 1 networks has been presented. With little adaption, EIBsec can also be applied to other KNX network media. However, EIBsec is tailored to the characteristics of KNX TP 1 (e.g., limited length of standard KNX frames) and dedicated to secure KNX frames. Therefore, the KNXnet/IP specific part of KNXnet/IP frames (e.g., KNXnet/IP header) as well as services that are only available for KNXnet/IP (e.g., KNXnet/IP device discovery) cannot be secured using EIBsec.

Due to these reasons, a new security extension to protect KNXnet/IP communication is presented in this paper. The basic protocol architecture of this extension is shown in Figure 4. The security layer is located between the network layer (UDP and/or TCP) and the KNXnet/IP layer. The main advantage of this architecture is that a change of the KNXnet/IP frame format is not necessary.

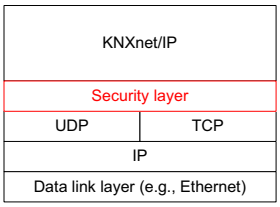


Figure 4: Protocol architecture

To be able to securely communicate with other KNXnet/IP devices, a KNXnet/IP device has to establish a secure communication relation to the other communication parties. This communication setup is divided into three phases: *configuration*, *key set distribution*, and *secure communication*. In the configuration phase, the ETS acts as certification authority (CA) and signs the public keys of the KNXnet/IP devices to create certificates. These certificates together with the public key of the ETS are distributed to the KNXnet/IP devices. In the key set distribution phase, the KNXnet/IP devices can authenticate themselves and agree on a so called key set. This key set is used in the third phase for setting up the secured channel.

5.1 Configuration using ETS

Figure 5[4] shows the operation sequence of the configuration phase. During configuration, each KNXnet/IP device creates its own key pair for asymmetric encryption. While the private key remains secret to the device, the public key is transferred via a physically secured channel to the ETS. The ETS acts as certification authority (CA). It signs the received public key to create a certificate and transmits the certificate together with its own public key back to the KNXnet/IP device. The certificate consists of the signed public key of the KNXnet/IP device and its IP address. With the help of these certificates, the KNXnet/IP devices can authenticate themselves during the key set distribution process.

[4] For the rest of this paper, the following notation is used: K denotes a private key, P a public key, KS a key set, SK a symmetric key, N a number used only once (called nonce), and $cert$ a certificate. Furthermore, $sign(X, text)$ denotes the signature calculation of $text$ using the secret key X, $encr(X, text)$ the encryption of $text$ using the secret key X, $secured(X, text)$ a secured transmission of $text$ using key set X, and $||$ the concatenation operator.

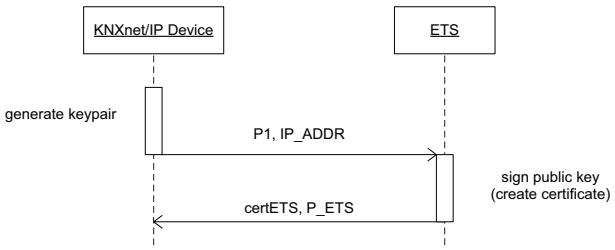


Figure 5: Configuration phase

Since the configuration phase is used to set up the initial security primitives (i.e., the public key of the CA and its certificate), establishing a secured channel using cryptographic techniques is not possible during the configuration phase. Therefore, the configuration process has to be performed in a physically secured environment. One possibility is to directly connect the KNXnet/IP device to the ETS via a point-to-point connection (e.g., using a local EIA-232 interface).

5.2 Key set distribution

After a KNXnet/IP device has been configured accordingly, it is in possession of three security primitives: the public key of the CA (i.e., the public key of the installation's ETS), its own public/private key pair, and its certificate that proves the authenticity of the public key. Using these primitives, a KNXnet/IP device is able to start the second phase where the key sets for the secure communication phase are distributed.

To distribute these key sets, some kind of *key set distribution protocol* is necessary. As mentioned in Section 2, KNXnet/IP specifies unicast and multicast services. Therefore, a key set distribution protocol is required that supports a distribution not only between two devices (for KNXnet/IP tunneling) but also among a multicast group (for KNXnet/IP routing).

In the proposed key set distribution protocol, the key sets for secure communication are maintained by a coordinator. In the case of a KNXnet/IP unicast service, the key set distribution protocol works as follows: Suppose, for example, a KNXnet/IP client wants to set up a secured channel to a KNXnet/IP server (cf. Figure 6). In the case of unicast, the KNXnet/IP server acts as coordinator. To retrieve the key set from the KNXnet/IP server, the KNXnet/IP client sends a `hello`-message to the KNXnet/IP server. This message is signed with the public key of the client and contains a nonce N_1 as well as the certificate of the client. The KNXnet/IP server receives this message and verifies the signature and the client certificate. If both are valid, the identity of the client has been proven and so the server immediately responds to the `hello`-message with a signed `coord_avail`-message, containing nonce N_1 of the `hello`-message and a nonce N_2. N_1 relates the response of the KNXnet/IP server to the actual `hello`-message and avoids that someone replays the server's response to an older `hello`-message [14]. To prove the identity of the server, the signature and the certificate are verified. If both are valid, the identity of the server has also been proven and so mutual entity authentication has been guaranteed. Afterwards, the client requests the key set from the server by sending a signed `key_req`-message – containing the nonce N_2 which relates this message to the `coord_avail`-message. Again, to avoid replay attacks in the next step, a newly created

nonce N_3 is included in the message. The coordinator responds with a `key_resp`-message, containing the number N_3 and the key set itself. The message itself is signed with the private key of the coordinator and encrypted with the public key of the KNXnet/IP client.

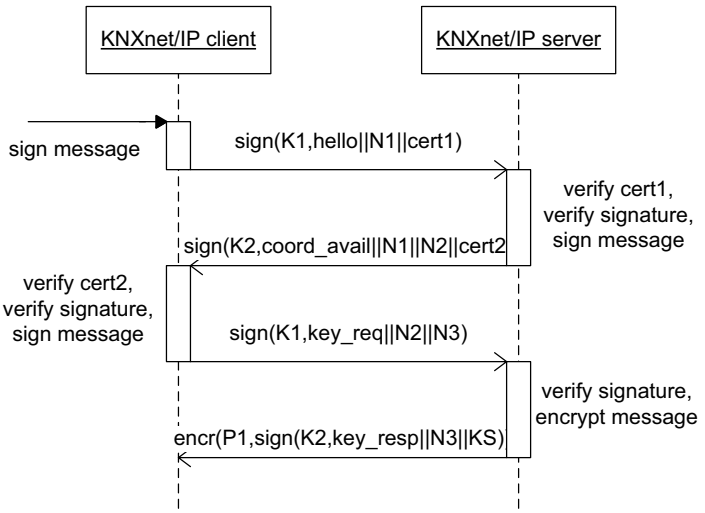


Figure 6: Key set distribution for unicast services

In the case of multicast services, the role of the coordinator is not predefined. Therefore, the KNXnet/IP devices that are members of the same multicast group[5] must agree on a common coordinator. This works as follows: If a KNXnet/IP device wants to join a multicast group (e.g., KNXnet/IP router after power up), it sends a `hello`-message to the multicast group. Again, this message is signed with the public key of the sender and contains a nonce N_1 as well as the certificate. Depending on the state of the multicast group, four different cases are distinguished:

1. Figure 7(a): If there is no response from other KNXnet/IP devices within the timeout t_{TO}, the sender of the `hello`-message assumes that it is the first active device within the group and so it becomes the coordinator. To announce that it assumes the coordinator role and to avoid that a second device tries to become coordinator at the same time, it will send a `coord_beg`-message. After having sent this announce message, it generates the key set for this group. The generation process takes the time t_{GK}. When the coordinator has finished the key set generation process, it sends out the message `coord_establ`. Note, that the timeout t_{TO} shall be greater than the key generation time t_{GK} to avoid conflicts with other devices that try to become coordinator at the same time (cf. Case 4).
2. Figure 7(b): If there is a coordinator available in the group, it immediately responds to the `hello`-message with a signed `coord_avail`-message, containing the nonce N_1 of the `hello`-message and a new nonce N_2. Again, N_1 relates the response of the coordinator to the actual `hello`-message and avoids that someone replays the coordinator's response to an older `hello`-message. After having received the response from the coordinator, the sender of the `hello`-message is able to request the group key set from the coordinator by sending a signed `key_req`-message. This message contains the nonce N_2 which relates this message to the `coord_avail`-message. Again, to avoid replay attacks in the next step, a newly created nonce N_3 is included in the message. The coor-

[5]For the rest of this paper, the term "multicast group" denotes a IP multicast group and not a KNX multicast group.

dinator respond with the encrypted `key_resp`-message, containing N_3 and the key set itself.

3. Figure 7(c): If there is a coordinator but it does not respond for any reason (e.g., the coordinator crashes), the new device takes over the role of the group's coordinator by sending a `coord_begin`-message (cf. Case 1). If there are other group members, they immediately respond with a signed `key_establ`-message, informing the new coordinator that there is already a group key set available. This message contains the nonce N_1 of the `coord_begin`-message and a new nonce N_2. The new device chooses one of the responding group members for authentication and obtains the key set from it by sending a `key_req`-message. After having received the key set, the new device takes over the coordinator role by sending a `coord_establ`-message.
4. Figure 7(d): If two devices are sending the `hello`-message at the same time (i.e., within the time interval $t_{TO} + t_{GK}$) and there is no coordinator available yet, the one with the smaller IP address will continue by sending a signed `won`-message, containing the nonce of the other device. This ensures, that the device owns the private key associated to the certificate. The loser acknowledges the correct message with a `ack`-message letting the winner continue establishing the coordinator role like in Case 1 by sending a `coord_beg`-message. If the device with the smaller IP address cannot send the correct message (cf. Figure 7(e)), the device with the higher IP address sends the `won`-message after a timeout. This can happen due to network problems or if the device is a malicious one which replayed a `hello`-message and does not own the private key for the sent certificate. If the winning device does not receive a (valid) acknowledgment, the winning device will repeat the `won`-message three times. If a valid acknowledgment is still missing, the winning device assumes that a competing device is a malicious one and so it starts establishing the coordinator role by sending a `coord_begin`-message.

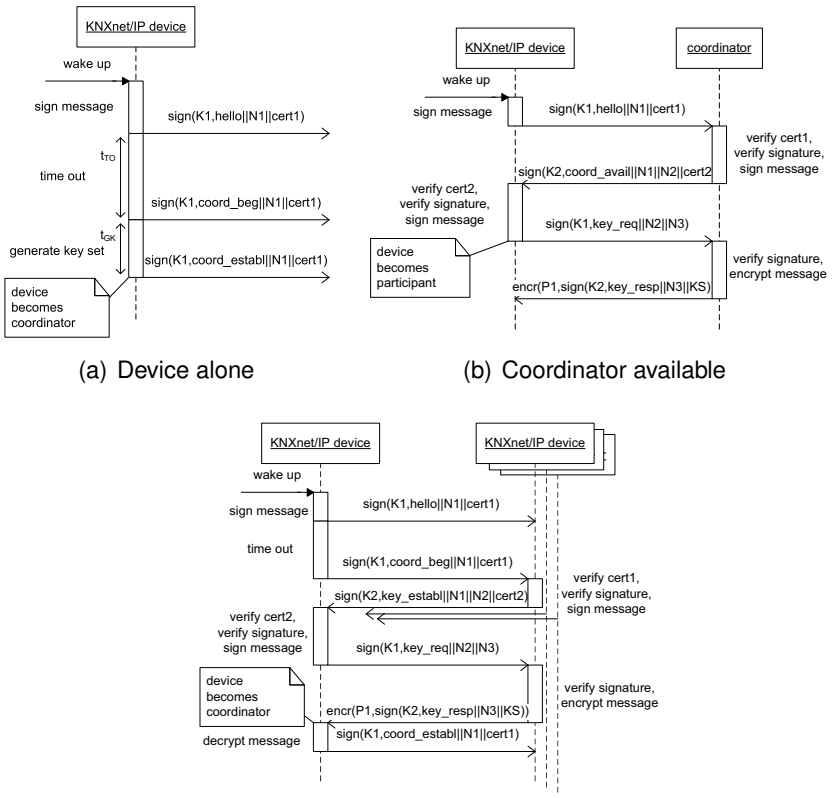
When the coordinator is established, each KNXnet/IP device can send a `key_req`-message to receive the group key set in the `key_resp`-message from the coordinator.

5.2.1 Certificate revocation

Since it is possible that a KNXnet/IP device gets compromised, a revocation list is required that contains the certificates of all compromised KNXnet/IP devices. This revocation list is maintained by the ETS and has to be distributed to all KNXnet/IP devices when a new certificate is added to the list. This is done by sending a `cert_revoke`-message that is signed with the private key of the ETS. Since all KNXnet/IP devices are in possession of the public key of the ETS, malicious `cert_revoke`-messages can be identified by verifying the signature.

5.3 Secure communication

After a key set has been distributed during the second phase, the secured channel between the communication parties can be established. To guarantee the security objectives of the secured channel, symmetric algorithms are used exclusively. Besides the advantage of higher processing speed, all multicast parties have to share only a single key. In contrast to asymmetric algorithms, a message has to be processed only once since KNXnet/IP devices of a communication relation (e.g., a single multicast group) use the same key set.



(a) Device alone (b) Coordinator available

(c) Coordinator dead

(d) Conflicting devices (e) Conflicting devices

Figure 7: Coordinator scenarios

A key set consists of 3 security primitives: a secret key SK_C for en-/decryption, a secret key SK_{MAC} for MAC calculation, and the current valid sequence counter C. The used frame format is based on the application protocol frame format of TLS 1.2 [8] (cf. Figure 8). As shown in this figure, a secured KNXnet/IP frame consists of an unsecured part and a secured part. The unsecured part includes of the Ethernet, IP, and UDP/TCP header as well as the Ethernet trailer. The secured part consists of the $initializationvector(IV)$ for the en-/decryption operation, the $Userdata$ that contains the KNXnet/IP frame, the MAC, the $Padding$, and the $Paddinglength$ as well as the total length of the secured part (i.e., sum of IV, $User\ data$, MAC, $Padding$, and $Padding\ length$). To ensure data integrity and data freshness, HMAC [15] in combination with

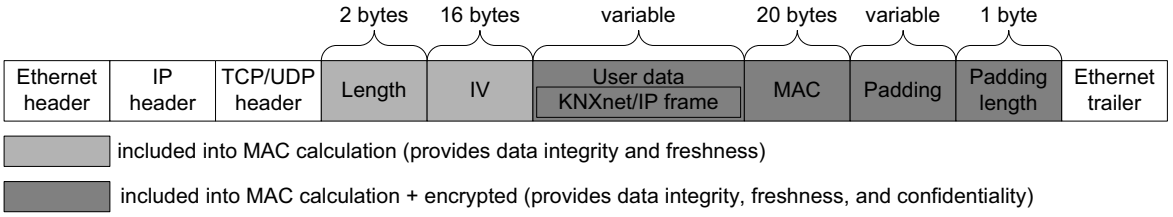


Figure 8: Secured frame format

SHA_256 [16] and a sequence number is used. HMAC provides an integrity check based on a secret key. The creation of the HMAC-tag takes 2 bit wise exclusive-OR operations (XOR) and two hashing operations:

$$HMAC(SK_{MAC}, text) = SHA\_256((SK_{MAC} \oplus opad)||SHA\_256\left((SK_{MAC} \oplus ipad)||text\right))$$

where SK_{MAC} denotes the shared secret key for MAC calculation. The secret key SK_{MAC} is XOR'ed with the constant values $ipad = 36_h$ and concatenated with the message text. After applying the hash function H (SHA-256 in our case), the result will be appended to the key K XOR'ed with the constant $opad = 5C_h$. The MAC itself is now calculated as

$$MAC = HMAC(SK_{MAC}, C||Length||IV||User\ data||Padding||Padding\ length)$$

where C denotes the current sequence counter and $||$ the concatenation operation.

To guarantee data confidentiality, AES_128 in cipher-block chaining (CBC) mode is used. The main advantages of AES are best performance and patent free implementation. The encryption process works as follows:

$$CIPHERTEXT = AES\_128\_CBC(SK_C, IV, User\_data||MAC||Padding||Padding\ length)$$

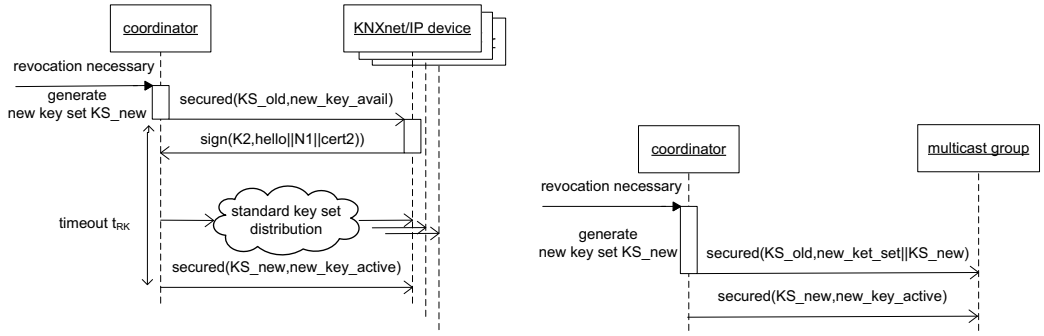
The AES_128_CBC encryption process takes three parameters as input: the shared secret key SK_C, the plain text (i.e., $User\_data||MAC||Padding||Padding\ length$), and IV. IV is for randomization of the first CBC operation. To avoid an attack on CBC mode [17], a random number has to be chosen as IV.

5.3.1 Key set revocation

There are two situations where the revocation of a key set and the distribution of a new one are required: First, it may be possible that a group member has to be actively excluded from

a communication relation. This may be necessary if a compromised device is added to the revocation list (cf. Section 5.2.1). Second, due to security reasons, the current used key set shall be recreated periodically to reduce the number of key set uses.

Therefore, the coordinator is able to create a new key set. To inform the other communication parties that a new key set is available, a `new_key_avail`-message is sent by the coordinator. In case of a compromised device (cf. Figure 9(a)), every device of the multicast group has to request the new key set from the coordinator as shown in Figure 7(b). Figure 9(b), on the other hand, shows the second case where a key set is renewed to limit the amount of key set uses. In this case, the new key set can be encrypted with the old one and transmitted over the already established secured channel. After a configurable timeout t_{RK}, the coordinator initiates the switch over to the new key set by sending the `new_key_active`-message, making the old key set obsolete.



(a) Revocation in case of a compromised device (b) Revocation in case of out-dated key set

Figure 9: Revocation scenarios

6 Prototype implementation

In the testing environment, a prototype KNX network has been set up (cf. Figure 10). This network consists of two small KNX TP 1 network segments which are interconnected by an IP backbone. Two AT91SAM7X-EK evaluation boards act as KNXnet/IP routers. While the onboard Ethernet controller provides the interface to the IP backbone, a TP-UART [18] is used for access to the KNX TP 1 network segment. For demonstration purposes, a standard KNX light switch as well as a standard KNX light actuator are connected to the TP 1 networks.

Connected to the hub, a PC running in promiscuous mode is sniffing all packets in the IP network. With the help of Wireshark [19] and the KNXnet/IP plugin [20], IP traffic can be displayed and analyzed. In unsecured mode, it is possible to intercept the clear text version of the KNXnet/IP frames. Furthermore, it is possible to replay intercepted KNXnet/IP frames to fool the KNX installation. When the security layer is enabled, the KNXnet/IP frames are secured and cannot be analyzed on the PC anymore. So, replay attacks are no longer possible.

6.1 Software architecture

In Figure 11, the basic hardware layout as well as the software architecture of the secure KNXnet/IP router is shown. The boards are equipped with an AT91SAM7X256 chip which is a

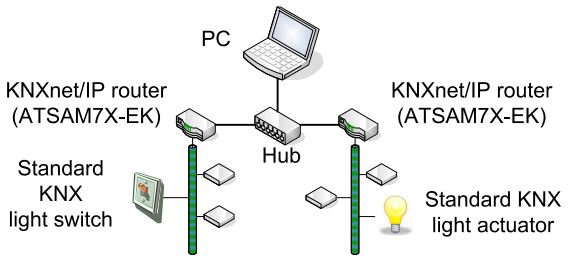


Figure 10: Prototype network

32-bit RISC Processor (ARM7TDMI) including 256 KB internal flash memory and 64 KB internal single-cycle access SRAM. It operates at up to 55 MHz. Via the Media Independent Interface (MII), the CPU is connected to a Davicom DM9161AE Ethernet chip. An ADM3202ARN chip drives the EIA-232 interface of the board which is used for the communication with the TP-UART.

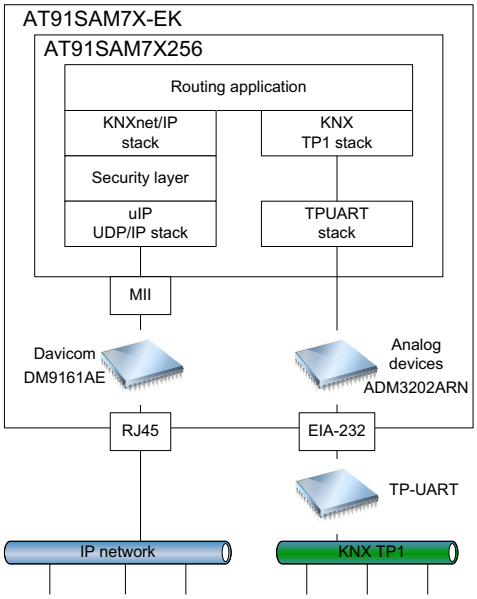


Figure 11: Software overview

The security layer of the software implementation is located between the UDP/IP stack and the KNXnet/IP stack. It is transparent to the KNXnet/IP stack and the layers above.

The open source library `uIP` serves as UDP/IP stack. It has been developed at the Swedish Institute of Computer Science [21]. `uIP` features very small code size and low memory usage, hence it is suitable for the use in embedded microcontrollers.

For cryptographic calculations, many arithmetic operations on large numbers have to be performed. Since native C cannot handle such large numbers, a special library has to be used. `MIRACL (Multiprecision Integer and Rational Arithmetic C Library)` supports all of the required functions. Furthermore, it provides many different cryptographic algorithms like AES, RSA, Diffie-Hellman, ECC over binary and prime fields, and hashing functions. Op-

timizations for different processor architectures are written in inline assembly for time-critical routines to enhance speed and exploit special features of some processors. The library also offers some parameters to optimize the memory consumption. The `MIRACL` library is developed by Shamus Software Ltd. [22].

The KNXnet/IP stack and the routing application support basic KNXnet/IP routing functionality. Extended functionality like the lost message handling and other services like KNXnet/IP tunneling are currently not implemented. Since the implementation and operation is completely independent from the security layer, the task of the security layer is unaffected by the reduced range of functions of the layers above. To communicate with KNX TP 1 devices, a KNX TP1 stack as well as a TP-UART stack are used. These stacks have been developed at the Vienna University of Technology [23].

7 Conclusion and future work

In this paper, a security extension for KNXnet/IP has been presented. This extension can be used to secure unicast and multicast KNXnet/IP services.

In large networks, the coordinator that distributes the shared key within the multicast group, will be a bottleneck and slow down the startup process. To solve this problem, a dedicated, more powerful key server that can handle more than one public key session simultaneously can be used. There is no need to change the implementation of the security layer in the KNXnet/IP devices since the authentication and key distribution sequence remain the same.

Another possible solution to gain performance during the key set distribution is to use a hierarchical structure of the KNXnet/IP devices. If organized in a tree, each KNXnet/IP device which has already received the key set can send it to its children. Figure 12 illustrates the key set distribution in a tree with eight KNXnet/IP devices. In the first round the coordinator sends the key set to one participant. In the second round, two key sets can be delivered, in the third round three devices will receive the key set, and so on. In this case with eight devices, it takes three rounds to spread the key sets, while the original implementation needs seven rounds.

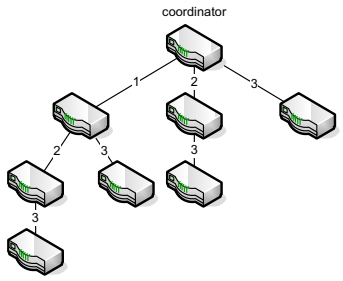


Figure 12: Key set propagation tree

Other improvements concern the implementation of the KNXnet/IP stack and the routing application. As already mentioned in Section 6, the current implementation provides functionality for KNXnet/IP routing. These software modules could be enhanced to support other KNXnet/IP services like tunneling and device management.

References

[1] W. Kastner, G. Neugschwandtner, S. Soucek, and H. M. Newman, "Communication Systems for Building Automation and Control", *Proc. of the IEEE*, vol. 93, no. 6, pp. 1178–1203, 2005.

[2] "KNX Journal", Jan. 2008, pages 2, 11–16.

[3] M. Neugschwandtner, G. Neugschwandtner, and W. Kastner, "Interoperable Web Services for Building Automation – Integrating KNX", in *Proc. Konnex Scientific Conference*, Nov. 2006.

[4] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 5th edition, 2001.

[5] W. Granzer, W. Kastner, G. Neugschwandtner, and F. Praus, "Security in Networked Building Automation Systems", in *Proc. 6th IEEE International Workshop on Factory Communication Systems*, June 2006, pp. 283–292.

[6] S. Kent and K. Seo, "RFC 4301: Security Architecture for the Internet Protocol", Dec. 2005.

[7] D. Harkins and D. Carrel, "RFC 2409: The Internet Key Exchange (IKE)", Nov. 1998, Status: PROPOSED STANDARD.

[8] T. Dierks and E. Rescorla, "RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2", Aug. 2008.

[9] V. Gupta, M. Millard, S. Fung, Y. Zhu, N. Gura, H. Eberle, and S. C. Shantz, "Sizzle: A Standards-Based End-to-End Security Architecture for the Embedded Internet", in *Proc. of the 3rd IEEE International Conference on Pervasive Computing and Communications*, 2005, pp. 247–256, Washington, DC, USA. IEEE Computer Society.

[10] D. R. Hankerson, S. A. Vanstone, and A. J. Menezes, *Guide to Elliptic Curve Cryptography*, Springer, 2004.

[11] B. Gleeson, A. Lin, J. Heinanen, T. Finland, G. Armitage, and A. Malis, "RFC 2746: A Framework for IP Based Virtual Private Networks", Feb. 2000.

[12] (Nov. 2008), OpenVPN [Online]. Available: http://openvpn.net

[13] W. Granzer, G. Neugschwandtner, and W. Kastner, "EIBsec: A Security Extension to KNX/EIB", in *Proc. Konnex Scientific Conference*, Nov. 2006.

[14] A. S. Tanenbaum and M. van Steen, *Distributed Systems: Principles and Paradigms*, Prentice Hall, 2002.

[15] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104 (Informational), Feb. 1997.

[16] "Secure Hash Standard", FIPS PUB 180-2, National Institute of Standards and Technology, Aug. 2002.

[17] B. Moeller, (Nov. 2008), Security of CBC Ciphersuites in SSL/TLS: Problems and Countermeasures [Online]. Available: http://www.openssl.org/ bodo/tls-cbc.txt

[18] Siemens, "Technical Data EIB-TP-UART-IC", 2001, version D.

[19] G. Combs, (Nov. 2008), Wireshark Website [Online]. Available: http://www.wireshark.org

[20] H. Weillechner, (Nov. 2008), Wireshark KNXnet/IP plugin [Online]. Available: https://www.auto.tuwien.ac.at/a-lab/wireshark-plugin-for-knxnet/ip.html

[21] A. Dunkels, (Oct. 2008), uIP Website [Online]. Available: http://www.sics.se/~adam/uip

[22] (Oct. 2008), Shamus Software Ltd. [Online]. Available: http://www.shamus.ie

[23] W. Granzer and W. Kastner, "BACnet over KNX", in *Proc. Konnex Scientific Conference*, Nov. 2007.

Introducing Security and Authentication in KNX

Salvatore Cavalieri and Giovanni Cutuli

University of Catania, Faculty of Engineering
Department of Computer Science and Telecommunications Engineering
Viale A.Doria, 6 – 95125 Catania (Italy)
E-mails: Salvatore.Cavalieri@diit.unict.it, Giovanni.Cutuli@diit.unict.it
Tel.: +39 095 738 2362, Fax:+39 095 738 2397

***Abstract* – Information flow exchanged in Home and Building Automation environments may be critical, as it may regard commands to actuators and/or private and secret information. The need to protect critical data exchanged has led to introduce secure transmission inside communication systems used for home/building automation applications. The paper deals with security and authentication inside the KNX, which, at this moment, doesn't foresee any security and authentication mechanisms. A security and authentication solution will be presented and assessed comparing it with other approaches known in literature.**

I. INTRODUCTION

Home and building automation is nowadays featured by a great choice of applications covering several areas, among which control, surveillance and entertainment. The information flow exchanged in some of these environments may be critical, as it may regard commands to actuators and/or private and secret information.

The need to protect critical data exchanged between some devices belonging to home and building environments has led to the need to introduce secure transmission inside communication systems. Secure transmission is generally realised by security and authentication mechanisms; security protects all the data during transmission while authentication guarantees procedures to install permitted devices.

The paper deals with security and authentication inside the KNX, which already met the requirements of both European Cenelec standards EN 50090 as well as EN 13321-1 and it has been recently approved as an international standard (ISO/IEC 14543-3), confirming its relevance in the home and building automation field [1][2][3].

At this moment, KNX standard doesn't foresee security and authentication mechanisms. In literature, some proposals improving secure transmission in KNX are present. Among them, one of the most remarkable is that presented in [4], called EIBsec. EIBsec foresees the presence of a complex device called ACU (Advanced Coupler Unit), which consists of a normal KNX coupler unit plus a key server unit. Cooperation among several ACUs realises security and authentication all over the network.

In this paper, the authors present another security and authentication solution; it is fully compliant with KNX standard, as it realises security and authentication through exchanges of information based on already existing services at KNX application layer and on KNX application objects.

The paper is organised as it follows. After a brief overview of the KNX standard, the proposed solution will be described in the details, focusing on its implementation inside KNX standard. Then, results of the validation of the procedures adopted to introduce security and authentication in KNX will be presented. Final remarks will focus on comparison of the proposal with the EIBsec solution known in literature, pointing out the advantages offered.

II. KNX OVERVIEW

KNX stack foresees the physical [5], data link [6], network [7], transport [8] and application layers [9]; session and presentation layer are empty and transparent to the neighbour layers.

Several physical media have been foreseen in the KNX standard: powerline [10], Radio Frequency [11] and twisted pair [12].

KNX Data Link Layer [6] includes the medium access control (based on the CSMA protocol defined in EN50065-1) and the logic link control. According to the medium access control, each devices can start medium access procedure only if the bus has been free for a period of time value, chosen randomly and uniformly distributed between 85 msec and 115 msec. If within 35 ms from the end of the message transmission an ack has not been received, transmission is considered failed; in this case, the transmission can be tried again until a maximum of 2 times. If the bus is always busy or the number of maximum retry has been reached, the transmission is considered failed and this event is communicated to the upper layers.

Network layer in KNX standard [7] is compliant with the definition of the ISO/OSI model network layer (ISO7498).

KNX Transport layer [8] provides data transmission over four different communication modes: point-to-point connection-less (multicast), point-to-all-points connection-less (broadcast), point-to-point connection-less and point-to-point connection-oriented.

According to KNX Application layer [9], particular objects, called datapoints, allow communication between devices; they are abstract objects giving access to internal objects (Application Interface Objects) in a device.

Datapoints can store any type of information and can be writeable and readable by a generic device. Services available at application layer can be divided in 4 groups: Multicast communication services (used for group communications), Broadcast communication services, Point-to-point connectionless communication services (used for communications between two nodes without establishing connection before starting transmissions), Point-to-point connection-oriented communication services (used for the communication between two nodes after having established connection).

KNX services at application layer used in this proposal are point-to-point connectionless services; in particular, read and write property services, called A_PropertyValue_Read and A_Property Value_Write, have been used.

III. Security and Authentication

In this section a brief overview about the main features of the existing approaches to realise security of data transmission and authentication of the devices involved in the data exchange is given.

A. Data transmission security

Cryptography is the science of writing in secret code and is an ancient art. In data and telecommunications, cryptography is necessary when communicating over any untrusted medium.

Modern cryptography techniques can be divided in Symmetric and Asymmetric cryptography.

In symmetric-key cryptography both the sender and receiver share the same secret called "key", used for encryption and decryption of each information exchanged between them. One of the most popular and secure algorithms used in symmetric cryptography is AES (Advanced Encryption Standard) [13]. AES is a fast algorithm both in hardware and software implementation and can be implemented even into a limited-resource microcontroller. The main disadvantage of symmetric ciphers is relevant to the key management, as each pair of communication devices must share a different key, so if the network has N devices, each device needs (N-1) keys to communicate and to exchange data with the other devices. Using symmetric cryptography there will be $N*(N-1)$ key in the network. Furthermore, using symmetric cryptography it is difficult to establish a secret key between two devices through an insecure channel.

In 1976, Whitfield Diffie and Martin Hellman proposed the concept of public-key cryptography also known as asymmetric cryptography [14]. In asymmetric cryptography two different keys are used. The keys are not equal but they are mathematically related; they are called public key and private key. Public key is known to everyone that wants to encrypt any message to be sent; it is computationally infeasible calculate private key knowing only the public key. The public key can be distributed to everybody but the private key must remain secret. The public key is used to encrypt messages and only the private key can decrypt them, that's why the private key must remain secret. Diffie and Hellman proposed a key-exchange protocol to allow devices to exchange the keys through an insecure channel. The Diffie-Hellman protocol uses the "Multiplicative group of integers modulo p". In the following a description of the Diffie-Hellman protocol is given by an example. The example will be relevant to a communication network, where a "device" is a communication node that wants to access the communication creating a secure channel; it interacts with another communication node, called "controller" which is in charge to create the secure transmission.

The controller chooses three numbers called "g", "p" and "a"; for example it chooses g = 4, p = 11 and a = 3. Number "a" must be kept secret. The controller sends to the device "g", "p" and "$A = (g^a) \bmod p$" (so it sends g = 4, p = 11 and A = 9). The device receiving the data set "g", "p" and "A", chooses its secret parameter "b" (for example b = 7) and sends back to the controller the number $B = (g^b) \bmod p$ (so it sends back B = 5, in the example). The device computes its temporary key, K, equal to $(A^b) \bmod p$; at the same time, the controller computes its temporary key given by $(B^a) \bmod p$. Both the controller and the device now have the same temporary key because $A^b = B^a$, in fact $A = g^a$ and $B = g^b$ so $A^b = (g^a)^b = (g^b)^a$. The key for this example is K = 4. As already seen, the only secret parameters are "a", "b", $g^{a \cdot b}$, $g^{b \cdot a}$. Of course, much larger values of "a", "b" and "p" are needed to make this example secure. Many studies showed that it is computationally infeasible calculate "a" given only "g", "p" and $g^a \bmod p$ (these parameters are sent in clear through the network) only if "p" is a prime number at least 300 digits long and "a" and "b" are at least 100 digits long. This problem is known as "the discrete logarithm problem".

B. Authentication

The authentication allows evaluating the authenticity of something or someone. In computer science the challenge authentication algorithms are widely used; basically they foresee that a device challenges another device that will be authenticated only if the response to the challenge is correct [15]. Challenge-based approach will be used in the proposal presented in the paper, as it will be shown.

IV. An Overview of the Proposal

The proposal foresees the presence of a special device called controller, inside KNX network; it coordinates all the activities about security and authentication procedures here proposed. Controller may be included in the BCU (Bus Coupler Unit), foresees by the KNX standard to couple different KNX segment [16].

According to the proposal here presented, the secure and authenticated access in KNX is gained through a three-step process.

During the first step, a device asks the controller to establish a temporary key using the Diffie-Hellman protocol [14], as described before. This key is useful for keeping secured all the next communications between controller and device. Once the temporary key has been established, the authentication step occurs; the controller challenges the device, which will be authenticated if and only if it can correctly evaluate the challenge just received. If the device is not able to send back a valid response it means that the device has not been correctly configured or it is a device that wants to get access for attacking the network performing a passive listening of the data transmitted on the bus (an example attack is the Man In The Middle Attack). If the authentication process successfully succeeds, the network controller can send the real network key to the device, so realising the last step of the proposal.

Once the device has the network key, it will encrypt/decrypt outcoming/incoming data using Advanced Encryption Standard (AES) [13]. If needed, the key may be periodically changed by the controller that communicates to the devices that the key should be regenerated; this means that each device has to restart the three-step procedure described above.

According to the proposal, each device (even a Man In The Middle device attacker) can get the temporary key, but it cannot get the network key if the authentication protocol fails. In this case it will not be able to access the network because the controller will not give the network key to it.

The proposed solution foresees a distributed deny of service detection because every time a new device is added to the network and gets the network key from the controller, this last inserts the device into a particular list (Secured Node List-SNL), which is distributed to the other authenticated devices. Doing so every device can drop data coming from a device that is not included in the last SNL received.

The authentication algorithm is based on some operations that the device should be able to perform once it has received the challenge by the controller. There are lots of challenge methods that can be used to authenticate devices, but in this proposal a simple challenge method is used, in order to light implementation on a μC; in any case, simplicity in this phase doesn't introduce criticism as the challenge step is already secured by the temporary key owned by the controller and the device. It is highly improbable that two devices has the same temporary key because every single device that will try to connect to the controller will obtain different A, g, p values randomly generated by the controller. It has been assumed that the controller sends a string of bit, made up by X bits for the operator plus N bits for the mask bit; the N value coincides with the length of the temporary key, while the X value depends of the number of operators used. For example assuming to use only four operators (Or, And, Xor and Nand), a value of 2 for X is given; in this case the 2-bits information code may be:

- 0 0 : perform OR (bit by bit)
- 0 1 : perform AND (bit by bit)
- 1 0 : perform XOR (bit by bit)
- 1 1 : perform NAND (bit by bit)

The operator coded by the X bits, is applied by the challenged device to the N mask bits and the temporary key. The result achieved is sent back to the controller as the result of the challenge; it will evaluated by the controller, as said before.

In the following a very simple example of the authentication procedure is given. Considering the 2-bit code seen before, it's assumed that the controller sends "11" plus mask bit, made up by the sequence 010. It's also assumed that the temporary key is 011. If the device is correctly configured, it will compute 011 NAND 010 obtaining 101, so it will send 101 to the controller. The controller compares the value sent by the device and the correct value. In the case of a valid response, the controller will send the network key to the device.

V. Realisation of Security and Authentication Inside KNX

The security and authentication procedures defined in the proposal have been fully integrated inside the KNX standard; in particular, KNX point-to-point connectionless application layer services and KNX datapoints have been used to realise the exchange of information between controller and devices to achieve security and authentication. In addition suitable data structures have been defined and implemented inside both the controller and device in order to implement the proposal here presented.

The following KNX Application Datapoints have been defined for a generic device:

- Diffie-Hellman g, p, A (DHgpA): this is the container for the receiving g, p and A parameters involved in the Diffie-Hellman key exchange process;
- Challenge Object (CO): this is the container of the challenge code sent by the controller once the temporary key exchange process is done;
- Network Key (NK): this object contains the network key sent by the controller once the authentication process succeeded;
- Restart Object (RO): if the controller writes this object, the device must restart all the process described in order to exchange a new network key created by the controller after a certain amount of time is elapsed.
- Secured Node List (SNL): this is an address list of the authenticated devices.

The application objects defined for the controller are:

- Activated Device List (ADL): when a new device is connected to the network it sends its own address to the controller, put in this object;
- Diffie-Hellman B (DHB): this is the container of the B parameter received by the device during the temporary key exchange process;

- Challenge Response (CR): this object will contain the response sent by the device that has previously asked for the authentication.

Each communication step is done using the already existing services at application layer. In particular in this proposal the A_PropertyValue_Write and the A_PropertyValue_Read application layer are used in order to allow the communication between device and controller.

In the following, the authors will deeply describe the protocol which implements the security and authentication proposal, pointing out the use of KNX datapoints previously introduced.

A. Security and Authentication Procedure

The procedure here proposed requires that at installation/configuration phase, the controller address must be set in each device to be connected to the network.

The request procedure for a temporary key is activated once the device sends its own address to the controller, writing the ADL objects inside the controller.

The controller generates A, g and p parameters and writes them into the DHgpA object in the device.

On the basis of these three parameters, the device computes the B parameters, writing it into the DHB object maintained by the controller.

From now on, both the device and the controller have the temporary key and they will use it to encrypt and decrypt the messages.

After this step, the controller writes into the CO object a code encrypted using the temporary key; this realises the challenge the device must perform in order to get authenticated.

The device will write the challenge response in the CR object of the controller so it can evaluate the response just received.

If the device sends back a valid response, the controller writes the network key into the NK object of the device. There aren't any problems in storing the network key into that object because it has been encrypted using the temporary key previously exchanged, and nobody else can know it as. From now on, the communication will be encrypted and the data send will be secured.

If a device sends back a wrong challenge response, it has been foreseen that the device may try to be authenticated for a certain number of times (fixed during configuration of the controller). Once the maximum number of attempts has been reached, the controller adds the device into an internal black list, refusing every attempts to be authenticated made by the device in the future.

The controller updates the Secured Node List (SNL) in each device, each time a new device has been authenticated. This list is used by each device before starting communication with another one, in order to verify if the other device is able to support secure communication, using cryptography. The SNL is present in each secured device so if the controller will stop working, encrypted communication is possible because every device knows the list of secured devices.

In order to improve the security, it's possible to foresee that periodically the controller change the network key, as already said. This is done, setting a value into the RO object in all devices. In this case, each device will restart the whole process in order to get the new network key.

VI. Implementation and Validation

A particular effort has been put in the verification of the feasibility of implementation and in the protocol validation of the proposed approach. Implementation has been realized on a very common microcontroller, while validation has been done using Estelle [17].

A. Implementation

A study has been carried on in order to verify the feasibility of the implementation of the proposed solution using the hardware/software resources generally required by a KNX device. This feasibility study seemed to be necessary as the Diffie-Hellman algorithm is asymmetric and involves many arithmetic and logic operations, requiring not trivial computing resources.

As explained in several papers presented in literature (see for example [18]), one of the controller used to implement KNX device belongs to Texas Instruments MSP430 family. In particular we considered the MSP430F2274 microcontroller equipped with 32 Kbytes of flash memory, 1 Kbytes of RAM and an operation frequency of 16 MHz.

The analysis carried on by the authors allowed to state the feasibility of the implementation of the proposed security and authentication approach. In particular no criticisms have been pointed out during execution of the Diffie-Hellman code inside the μC considered in the study.

B.Validation

Validation has been carried on in order to verify correctness of the security and authentication procedure here presented, taking into account two different points of view, as explained in the following paragraphs.

In the previous section, the security and authentication procedure has been described in a very simple way; indeed, the procedure is based on a state machine, whose formal description has been avoided in this paper in order to light the overview of the security and authentication procedure. On account of what said, procedure validation means verification of the correctness of the state machine which it has been based on; this means to perform a reachability analysis (i.e. verify that each state is reached for the exact inputs/actions foreseen in the procedure), and to be sure that that no deadlock may occur (i.e. the system remains blocked in a state).

The aim of the proposal presented in the paper was, as said more than once, that to introduce security and authentication inside KNX, in order to block and to keep out from

communication all the devices not authorised. For this reason, the other goal of the validation was that to verify the real capability of the proposal to achieve its aim.

Validation has been realised taking into account the two goals just described, through modelling and simulation, using Estelle [17], which is particularly suitable to model state machine. The Estelle Development Toolset [19], has been used to perform both Estelle-based modelling and simulation. Using EDT has been possible to simulate the exchange of information between communication devices and the controller, keeping under control every step of the proposal here presented.

A KNX network made up by 10 communication nodes and 1 controller has been modelled. The KNX communication stack and services and protocols involved in the proposal have been modelled in Estelle for each communication node.

As one of the goals of the validation was that to verify the capability of the proposal to block communication devices not authorised, the following roles have been modelled inside each communication node:

- Secure device. The activities modelled for this kind of communication node are relevant to the typical send/receive activities performed in a KNX network. This kind of device is successful authenticated and secured by the controller.
- Attacker device. The role modelled is relevant to a device that wants to access the network to get information and to change data depending on the attack they want to perform. Two types of attack have been simulated:
  - Denial Of Service (DoS). In this scenario, attacker devices, once received the temporary key from the controller, try to send authorisation requests keeping busy the controller in order to block authorisations to other devices.
  - Man in the middle (MiM). The attacker devices sniff the communication medium, catch each frame sent and change information here contained, on the fly. Information written by the attacker devices is relevant to the data field and depends on the kind of attack that has been performed.

Simulation carried on allowed to verify that each state of the state machine defined by the authors is reachable and that no deadlocks occur.

Further simulation revealed that secure devices obtain network access given by the controller while the attacker devices have been kept out after a certain amount of failed attempts. In fact, all the devices obtained the temporary key for securing initial step of the proposed protocol, but the attacker devices couldn't be able to correctly answer to the challenge sent by the controller, not receiving the network key.

In particular, considering the DoS attack, the attacker devices that tried to unsuccessfully get network key more than the pre-configured maximum number of times, are temporary inserted into the black list maintained by the controller. In this way, the DoS attack becomes ineffective.

Considering the MiM attack, it is blocked from the beginning. As said more than once, everything is sent on the network is secured by the temporary key; for this reason the attacker cannot understand the information contained in each frame it sniffs on the bus. The consequence is that when it changes the data field, it cannot calculate a new CRC for the data it wants to overwrite, making the new frame produced will be discarded by the receiving node as wrong.

VII. Final Remarks

The current version of the KNX standard does not foresee services to guarantee secure transmission and to perform device authentication. Everybody can listen the network in order to catch traffic because data are sent in clear. It would be very dangerous if an unauthorized device (like an attacker) could read/write some important data on the bus especially if the data are relevant to some security control in the home/building.

The paper has presented a way to introduce data security and authentication mechanisms. Basically, the proposal features a secure data exchange realised through the well-known Diffie-Hellman algorithm.

The proposal is fully compliant with the current KNX standard as it uses the KNX services available at the application layer and the relevant communication objects, in order to realise all the exchange of information aimed to secure the communication.

As said in the introduction, nowadays literature presents other proposals concerning security and authentication in KNX. Among them, the proposal which seems the most important, due to the huge number of citations, is that presented in [4] and called EIBsec.

The approach here presented features several advantages if compared with EIBsec; in particular, comparing the two proposals, the following advantages may be pointed out.

The solution here presented doesn't need ad-hoc services, because some services present at application layer are used to support the exchange of information between controller and devices, as already said before; this means that no extension to KNX standard is needed, as done in the EIBsec proposal. Another difference between the two proposals is about the configuration procedure; in the EIBsec the management procedures is unfeasible using the ETS tool [20], on account of the presence of the extension of the KNX services. Our proposal continues to maintain the original ETS tool, due to the full compliance with the KNX services.

Another difference is that in our proposal there is no initial secret key distribution problem. As explained in [4], EIBsec foresees a preliminary procedure related to a “secure initial key distribution” between a Server Key (contained in the ACU controller) and the devices. This distribution is realised using ad-hoc services (i.e. not foreseen in the KNX standard) and is basically in clear, so each key transmitted may be

intercepted; for this reason EIBsec proposes ad-hoc procedures to make a secure key exchange. Possible solutions proposed inside EIBsec, are the use of point-to-point connections to upload keys into the devices, or upload into the EEPROM of the device. In the approach here proposed, an exchange of information between the controller and devices is foreseen in order each device achieves the initial secure key; the information flow is realised using standard application layer services and is related to the Diffie-Hellman algorithm, consisting of the three parameters called "g", "p" and "A". This exchange of information is intrinsically secure, i.e. cannot be affected by attack, as can be verified in [14].

So concluding, both the EIBsec and our proposal foresee the use of an initial key distribution. In EIBsec, distribution of this key has been realised by ad-hoc management system and ad-hoc services; this distribution is basically in clear, but secure procedures are suggested in order to make key distribution secure. On the other hand, in the approach here proposed, distribution of the initial key is intrinsically secure and uses KNX services.

REFERENCES

[1] Cenelec EN 50090, Home and Building Electronic System (HBES), 2005.

[2] Cenelec EN 13321-1, Open Data Communication in Building Automation, Controls and Building Management, HBES, 2006.

[3] ISO/IEC 1454-3, Home Electronic System (HSE) Architecture, 2006.

[4] Wolfgang Granzer, Georg Neugschwandtner, and Wolfgang Kastner, "EIBsec: A Security Extension to KNX/EIB", Konnex Scientific Conference, November 2006.

[5] KNX Association, KNX Standard, System specifications: *Physical Layer General*, Chapter 3/3/1, 2001.

[6] KNX Association, KNX Standard, System specifications: *Data Link Layer General*, Chapter 3/3/2, 2001.

[7] KNX Association, KNX Standard, System specifications: *Network Layer*, Chapter 3/3/3, 2001.

[8] KNX Association, KNX Standard, System specifications: *Transport Layer*, Chapter 3/3/4, 2001.

[9] KNX Association, KNX Standard, System specifications: *Application Layer*, Chapter 3/3/7, 2001.

[10] KNX Association, KNX Standard, System specifications: *Communication media Powerline*, Chapter 3/2/4, 2001.

[11] KNX Association, KNX Standard, System specifications: *KNX Radio Frequency,* Volume 3, Supplement 22, 2003.

[12] KNX Association, KNX Standard, System specifications: *Communication media Twisted pair*, Chapter 3/2/2, 2001.

[13] Tariq Jamil, *The Rijndael algorithm*, Potentials, IEEE, April-May 2004, Volume: 23, Issue: 2, pp. 36- 38, ISSN: 0278-6648, INSPEC Accession Number: 7965773.

[14] W. Diffie and M. E. Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory, vol. IT-22, pp.644-654, Nov. 1976.

[15] H. M. Sun, *An efficient remote user authentication scheme using smart card*, IEEE Trans. On Consumer Electronics, Vol.46, No. 4, pp. 958 – 960, Nov. 2000.

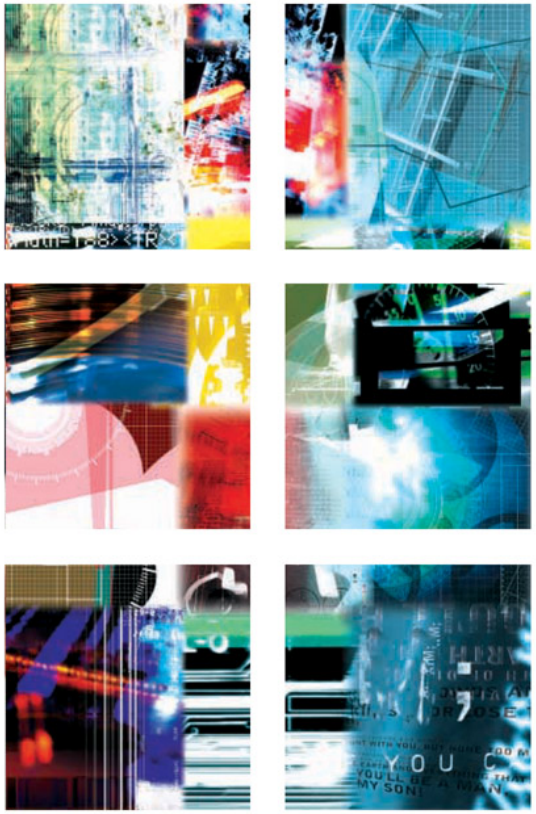
[16] KNX Association, KNX Standard, *Basic and System Components/Devices – Minimum requirements – standardized solutions - Tests*, Chapter 9/4/1, 2003.

[17] ISO/TC97/SC21/WG1/DIS9074, *Estelle – A formal description* technique *based on an extended state transition model*, 1987.

[18] Dr. T. Weinzierl, *Implementation of the KNX landscape*, KNX Scientific Conference 2006.

[19] http://www-lor.int-evry.fr/edt/

[20] http://www.knx.org/knx-tools/ets/description/



www.knx.org

Secure Communication

This document is a presentation of the ongoing working Task Force Secure Communication.

All contents are subject to change

Outline

- Introduction
- Scope of Secure communication
- General rules
- Cryptographic assets
- Secure frames
- Resources
- Configuration
- Key generation & exchange
- Runtime demonstration

Introduction

- **Task Secure Communication is still working and all contents of this presentation are subject to changes.**

- **Secure communication provides a solution for**
  - Unsafe transmission
    - Wireless link / RF media
    - Power line
  - Secured applications
    - Shutter & door control
    - Anti-Intrusion
    - Metering
    - …….

Scope of secure communication

- **Media**
  - Radio Frequency
  - TP1
  - Power line
- **Configuration**
  - Push-button
  - Controller Mode, LTE
  - ETS standard mode
- **Product types**
  - RF : unidirectional & bidirectional
  - TP : must be compatible with existing couplers
    (TP uart limit to 64 bytes telegrams)

Scope of secure communication

- **Confidentiality**
  - Information should only be available to those who are authorized (ex : HVAC status = holidays may indicate a absence of people in the house)
- **Authentication & integrity**
  - Data origin authentication : the receiver is sure of the identity of the sender
  - Transmitter and receiver are sure of the identity of the other party
  - preventing an unauthorized party from altering data.
- **Data-freshness**
  - It guarantees that the data processed by en entity is valid at the current point in time. Thus it prevents from message injection and replaying attacks, by an unauthorized entity.

General rules

Rule 1

- The secured communication feature is related to the runtime (Group Communication) and the management (P2P Communication)

Rule 2

- The need of a secure communication is a requirement from the receiver

Recommendation 1

- Confidentiality → product requirement
- Authentication → in input datapoints of the receiver

Cryptographic algorithm

- **Independence of algorithms**
  - The secure communication is expected to permit the selection of different symmetrical cryptographic algorithms without affecting the other parts of its implementation

- **Preliminary specification**
  - AES 128 : Symmetrical algorithm, 10^{28} MIPS years to attack, Protection life time : beyond 2031
  - Free sources in C for AES 128 have been implemented on a MSP430F2370 at 4MHz.
    - (http://www.progressive-coding.com/tutorial.php?id=0)
    - 16 bytes encryption → 38 ms
    - 16 bytes decryption → 38 ms

Cryptographic algorithm

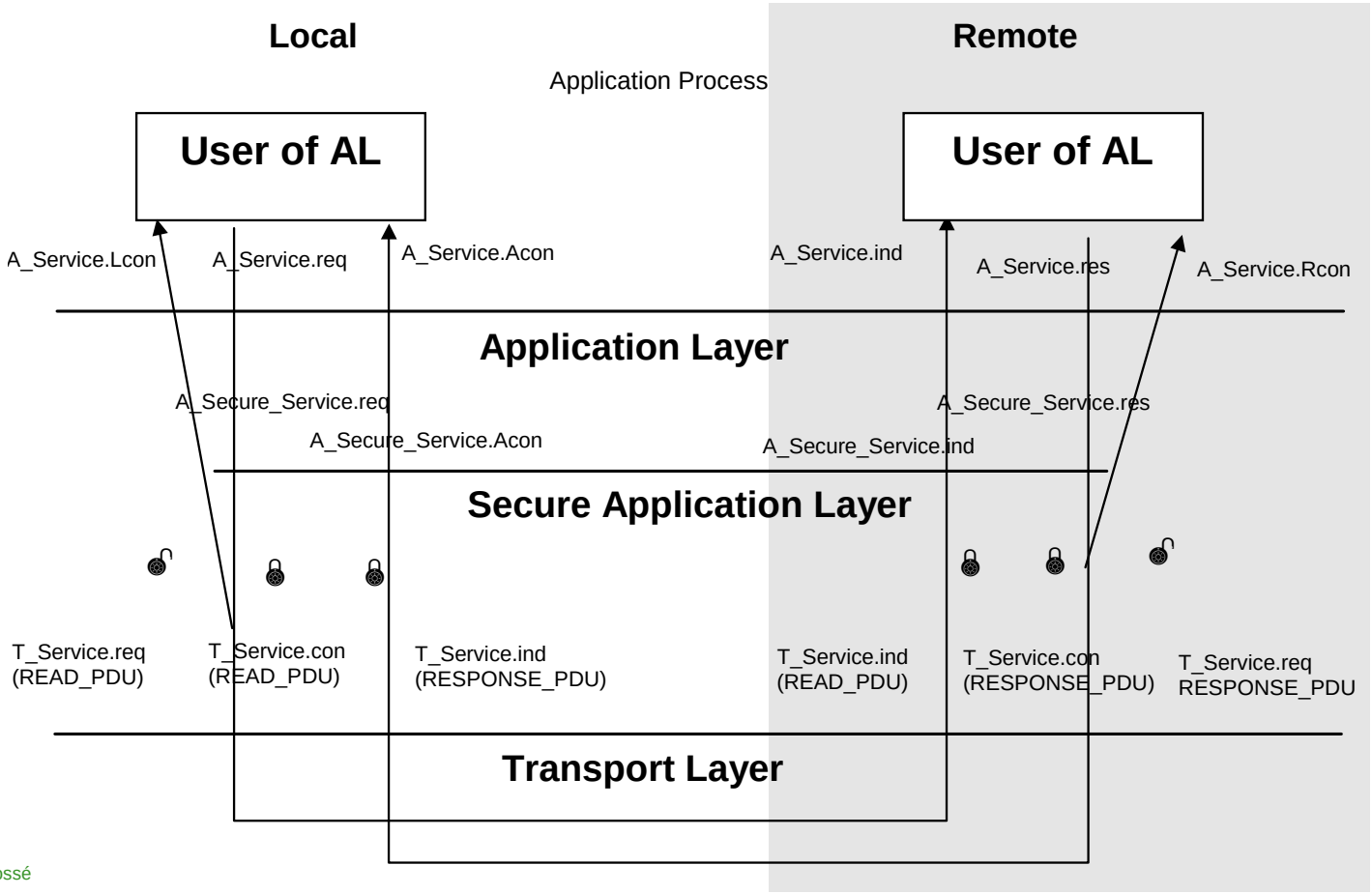
- **Prelilminary specification (2)**
  - A commercialized solution mentions for AES-128 for MSP430 (asm + C interfaces) mentions :
    - Encryption : 5342 cycles (thus 1,4 ms at 4 MHz)
    - Decryption : 8802 cycles (thus 2,2 ms at 4 MHz)
    - http://jce.iaik.tugraz.at/sic/products/crypto_software_for_microcontrollers/texas_instruments_msp430_microcontrollers
- **Sequence number (counter)**
  -

size random value (bits)	128	64	48	32	16
counter values possibilites	3.40282E+38	1.84467E+19	2.8147E+14	4294967296	65536
max Knx messages sent / sec	89	89	89	89	89
years number	1.21239E+29	6572386712	100286.662	1.53025303	2.335E-05

  - Minimum counter number size : 32 bits

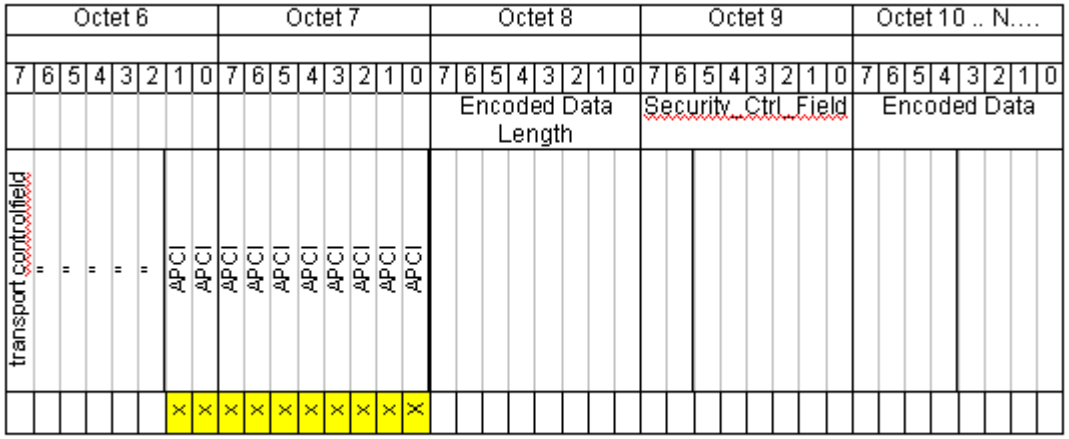
Secure frame

- A_Secure_Application service



Secure frame

- The secure application service is applied by the Application Layer to transmit a secure telegram on the bus. When the remote application layer receives this service, it will check the security according to the configuration of SKI (Security Key Information), and reject or transmit internally to AL according to the security checking.



Format of the secure A_Secure_Application-PDU

- The security_ctrl_field indicates Confidentiality, Authentication, algorithm & operation modes, padding methods that are used to secure the “original” Data.

Secure frame

- **Extended frame format**
  - The data length necessary for Secure telegram requires the use of long frames. For RF medium, all frames are capable of long data. For TP, extended frame format must be used
- **Communication mode**
  - The “security key info” used by the A_Secure_Data_Service are depending on the communication mode (multicast, broadcast, point-to-point).
    - P2P : Individual source address is checked in SKI table
    - Multicast : Datapoint Number is checked in SKI table
    - Broadcast : Individual source address is checked in SKI table
- **Secure Mode switching**
  - PID_Secure_Mode function property disable and enable the Security mode. If Secure_mode is enabled, then the Function Property service must be checked for security before changing the mode. This fix one record in the SKI table : (TSAP type= 00, individual address of the Tool, Key of the Tool).

Secure frame

- **Security control field**

b7	b6	b5	b4	b3	b2	b1	b0
algorithm				Padding method		Authentication	Confidentiality

Figure 4 : Security Control Filed byte

- Algorithm : AES-128 with ECB mode
  (= each 16-byte data block is encrypted separately)
- Padding method : random byte padding

Secure frame

- **Encoded data (confidentiality only)**
  - Encoded_Data_Length = length(Decoded_Data + Padding_Data)
  - Encoded Data = **Encryption** (
    Security_Control_Field.algorithm,
    Decoded_Data + Padding Data,
    Confidentiality Key)

Decoded Data				
2 bytes	4 bytes	Variable length	Length indicated in padding length	2 bytes
Padding length	**Counter**	**initial APCI + Initial data**	**Padded bits/bytes**	**CRC defined with initial data**

Figure 5 : Decoded Data

Secure frame

- **Encoded data (authentication only)**
  - Encoded_Data_Length = length(APDU + Padding_data) + signature size
  - Encoded Data = APDU + Padding_Data + **Signature** (Decoded_Data, Authentication Key)

Decoded Data		
2 bytes	4 bytes	APDU length
Padding length	**Counter**	APDU

Secure frame

- **Encoded data (encryption & authentication)**
  - Encoded_Data_Length = length(APDU + Padding_data(signature)) + signature size + length(Padding_Data(encryption))
  - Encoded Data = Encryption (Security_Control_Field.algorithm, **APDU + Padding_Data(signature) + Signature (Decoded_Data) + Padding_data(encryption), Authentication Key)**

Decoded Data1		
2 bytes	4 bytes	APDU length
Padding length	**Counter**	APDU

Decoded Data2				
2 bytes	Variable length	Signature size	Length indicated in padding length	2 bytes
Padding length	**APDU**	**Signature** (Decoded_Data1)	**Padded bits/bytes**	**CRC defined with initial data**

Resources

- **Security Key Info**
  - Two parts
    - Configurations information

TSAP (2 bytes)	Serial Number (6 bytes)	Reserved (4 bits)	TSAP Type (2 bits)	Confidentiality requirement (1 bit)	Authentication requirement (1 bit)	Confidentiality key (size :16 bytes)	Authentication key (size :16 bytes)
		0000					

    - Runtime information (must save and restored in cased of power down)

Counter against replay attacks (4 bytes)	Hack attacks counter (1 byte)

Resources

- **Counter**
  - Increment : +1
  - Acceptance windows : 20
  - Out of windows => increment hack attack counter in S.K.I
  - Resynchronization : automatically
    - If some failed frame occur (ex: out of the acceptance window), the same number of good frames is requested in order that the receiver accepts a telegram. (Good frames means that the counter is incremented by the increment, but can be outside the window)

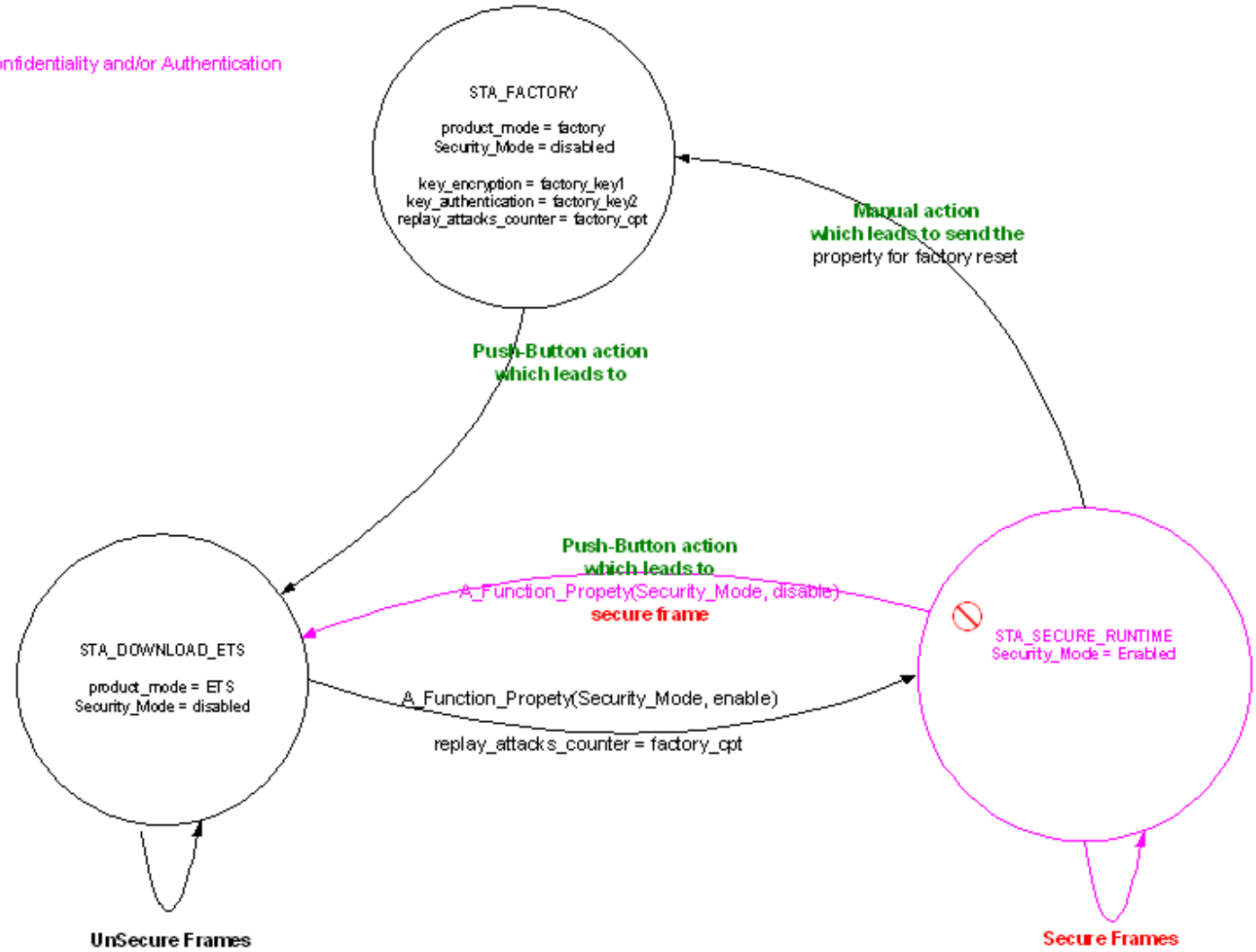
Configuration

- **General configuration rule**
  - All situations require a press on a self-confident product.
  - It’s a mean to prevent from unwanted configuration (burglars).

Configuration

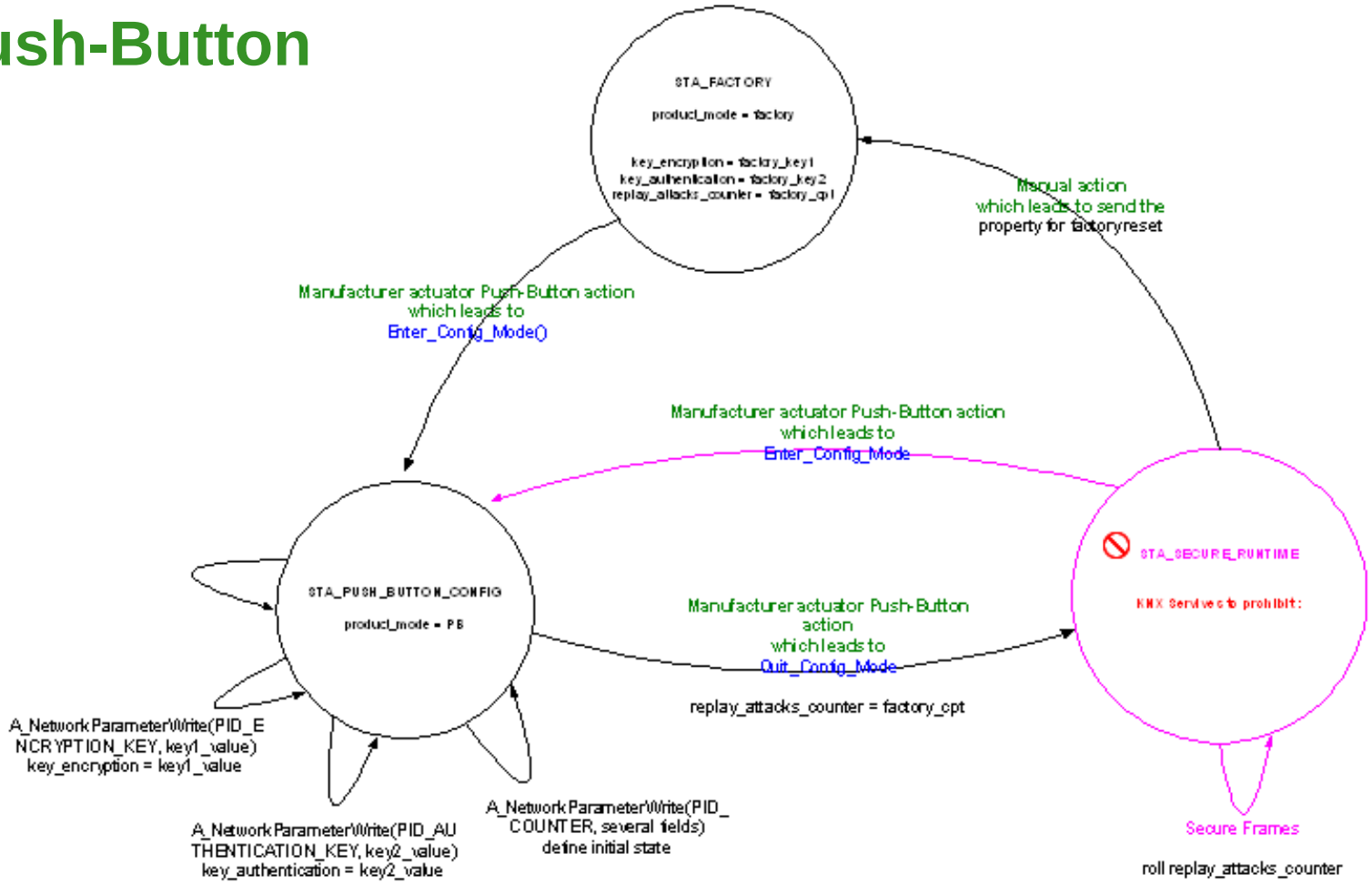
- **ETS**

Secure frame : frame for confidentiality and/or Authentication



Configuration

- **Push-Button**



Runtime demonstration

- 1 secured RF push-button and 1 receiver
- 1 unsecured RF push-button and 1 receiver
- Hager USB RF sniffer
- Key are hard coded and security mode is no switch able

- To show :
  - Timing aspects (between secured and unsecured)
  - Only Software evolution on existing devices
    - Msp430 based with Semtech RF chipset

Runtime demonstration

- Unsecured RF PB telegrams

1	00 09 06 70 16 D4	5FF	8	81		3	6	31/10/2008	14:34:35
2	00 09 06 70 16 D4	5FF	8	81		3	6	31/10/2008	14:34:35
3	00 09 06 70 16 D4	5FF	8	80		3	6	31/10/2008	14:34:37
4	00 09 06 70 16 D4	5FF	8	80		3	6	31/10/2008	14:34:37

ON
ON
OFF
OFF

2 telegrams because PB is unidirectional)

2 telegrams because PB is unidirectional)

- Secured RF PB telegrams

1	00 09 12 34 56 78	5FF	8	3FE	10 05 7F 80 25 FC 29 20 D5 0D 5A 08 B0 D1 F3 A0 43 FA	2	0	31/10/200
2	00 09 12 34 56 78	5FF	8	3FE	10 05 0D 69 69 86 6C AD 28 BC 92 42 B1 CC 15 C5 F2 83	2	0	31/10/200
3	00 09 12 34 56 78	5FF	8	3FE	10 05 F9 83 54 F5 05 9B 86 43 5C 98 69 FF D6 E9 C3 68	2	0	31/10/200
4	00 09 12 34 56 78	5FF	8	3FE	10 05 9D 40 F7 E4 46 5B 8C 61 DB 74 52 85 90 8C 36 FA	2	0	31/10/200

ON
OFF
ON
OFF

Encoded Length data = 16

Algorithm = 0 AES128 ECB
Confidentiality ON, Padding = Byte random

APCI Secure = testing value

Conclusion

- **Specification still under work in Task Force Secure Communication**
- **Key exchange and generation to be defined**
- **Solution possible only with software but also compatible with Hardware chips**
- **Same for all medium (requires serial number on TP1)**
- **Not so far from ISO/IEC FDIS 24767-2**

  (Internal security services : Secure Communication Protocol for Middleware)