On the security of security extensions for IP-based KNX networks

Aljosha Judmayer, Lukas Krammer, Wolfgang Kastner
Vienna University of Technology
Faculty of Informatics
Automation Systems Group
{ajudmayer, lkrammer, k} @auto.tuwien.ac.at

Abstract

The traditional areas of application for building automation systems (BAS) like heating, ventilation and air conditioning as well as lighting and shading are more and more extended by services requiring a more robust security infrastructure like alarm- and access control systems. Additionally, building automation networks get integrated into existing IP-based networks, or even communicate directly over the Internet. Therefore, the attack surface of BAS has increased dramatically. This requires a solid security architecture and a profound knowledge of possible attack vectors. This work reviews two security extensions for KNXnet/IP regarding their individual security properties. Thereby, it is pointed out that the current version of the draft specification, called KNXnet/IP Secure, lacks some relevant details and has certain limitations concerning the provided level of security.

1. Introduction

Since building automation networks can make use of available office networks, or even communicate with other building automation technologies over the Internet, they can no longer be seen as independent and sealed-off systems. The demand of a higher level of integration with other services and networks as well as the need of backward compatibility require a well conceived security architecture and a profound knowledge of possible attack vectors.

Therefore, two new security extensions for securing IP-based KNX networks are reviewed in this paper regarding their individual security properties. The first extension was published by the KNX association and is called *KNXnet/IP Secure* [1]. The second security extension was proposed by Granzer et al. [2] and is called *generic security concept for IP backbones*. It is shown that both proposed extensions have certain limitations concerning the provided security properties. What both extensions have in common is that they only work on IP-based backbone level. This means that all attacks on KNX line level which are outlined in Section 2 are still possible even when these extension are deployed. Since the draft specification of KNXnet/IP Secure provides more detail, the focus of this paper lies on this specification. Because the reviewed document is still a draft version, also theoretical attacks are described in this paper. These attacks are marked with the prefix *theoretical* in the headline. Theoretical attacks might cause problems in future versions of the specification, or offer unnecessary attack surface. These theoretical weaknesses are described in this paper so that they can be considered by implementers and future versions of the protocol.

At the time of writing there was no implementation of any of these two protocols available for evaluation. Therefore, software and implementation errors, like for example the correct use of a secure random number generator, are out of scope of this analysis.

The contribution of this paper is the comparison of the two proposed security extensions *KNXnet/IP Secure* [1] and the *generic security concept* [2] in terms of security. Thereby, previously unreleased weaknesses are described and discussed.

1.1. Related work

Extensive research has been conducted in the field of security in Building Automation Systems (BASes) [3, 4, 5, 2]. In [5], some well established open BASs have been analyzed regarding their individual security properties. The examined BASs are *BACnet*, *LonTalk*, *KNX* and *ZigBee*. This research implied that, with the integration of security critical services (e.g. access control mechanisms, alarm systems) into the area of building automation, certain security requirements have to be provided by BASes. The research in [5] has shown that various BASs lack fundamental security properties. Among these systems, KNX is the only one that does not utilize any form of cryptography to achieve security requirements. Therefore, the security of KNX relies heavily on so called ”security by obscurity” and physical separation which is not really applicable for KNX over RF and KNX over WLAN. Although it is known that KNX lacks fundamental security properties there are hardly any documented attack vectors or example scenarios.

1.2. Structure of this paper

Section 2 gives a general introduction to KNX. Thereby, current security issues as well as possible attack vectors are outlined. To address the current security issues in KNX, a new draft specification called *KNXnet/IP Secure* [1] was developed. This extension specifies a protocol to secure IP-based KNX traffic. The specification is described and reviewed in Section 3. In [2], a *generic security concept* for securing IP-based communication in BAS is proposed. This concept is outlined and reviewed in Section 4.

2. State of the art

This section briefly outlines the basic principles of KNX and describes some possible attack vectors in a classical KNX based building automation network which does not utilize any security extension.

2.1. KNX overview

KNX is a well-established standard for home and building automation. It emerged from the fusion of three major automation systems: European Installations Bus (EIB), European Home Systems (EHS) and BatiBus. KNX, as it is known today, was defined by the KNX Association, which is a pool of companies that founded this consortium. The KNX Association publishes the KNX Systems Specification [6], the *KNXnet/IP Secure* specification [1] and is also responsible for the certification of KNX compliant products.

The main purpose of KNX is ensuring the interoperability between products, applications and systems. This allows Sensors, Actuators and Controllers (SAC) from different vendors to work together and form a building automation network. Thereby, KNX supports several different physical layers. This paper focuses on two more widely used physical layers, namely KNX over twisted pair cables (TP1) and KNX over Ethernet (IP). KNX data that is transmitted over an IP-based network is basically encapsulated in UDP/IP datagrams. In the context of IP communication, this procedure is called KNXnet/IP.

For a detailed description of KNX please refer to the KNX Systems Specification [6] which is freely available for all members of the KNX Association.

2.2. Security in the current version of KNX

KNX supports *basic access protection* to authenticate unicast communication. This protection scheme allows to define up to 255 different access levels, where 0 is the most privileged one. Each access level can be secured by a different 4 byte password. The password as well as the exchanged data remain unencrypted while transferred over the network. Because of the lack of strong cryptography, the length of the password and the restriction to unicast communication, *authenticity*, *integrity*, *confidentiality* or *data freshness* **cannot** be guaranteed for classical KNX communication.

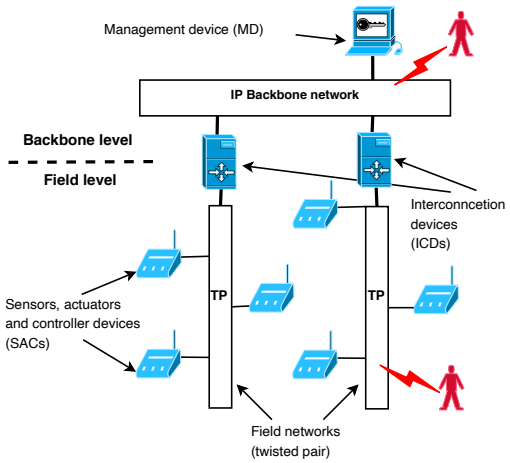

Figure 1: A two-tier KNX BAS

The KNX specification defines some security guidelines. These guidelines are based on clear network separation and "Security by Obscurity" and therefore not considered a strong security feature e.g.:

"Filtering KNXnet/IP datagrams from the network requires network analysis tools and expertise. The content of a KNXnet/IP message is not self-descriptive but requires semantic knowledge"[6]

2.3. Network attacks in the current version of KNX

Figure 1 shows the structure of a basic two-tier KNX based BAS. This installation utilizes an IP-based backbone network and twisted pair on line level. If an attacker is able to connect to a KNX line, or a KNX backbone network, there are currently **no** mechanisms available to prevent tampering. To outline a potential attack scenario, a KNX building automation network was simulated by using the Siemens GAMMA Training Kit 2 (GTK2). There are two main points where an attacker can launch network attacks: at KNX line level and at KNX backbone level.

To connect to a KNX twisted pair field network (also called *line*), an attacker needs access to the wiring of the twisted pair cabling itself or to an actuator, controller or sensor like a light switch, for example. In the test scenario the USB interface N 148/11 of the GTK2 was used. This USB interface to the KNX bus can be connected to the bus wiring by pressure contacts. At the KNX line level an attacker can use the open source software *eibd*[1] to monitor traffic, read the configuration of available devices and send arbitrary messages. Since KNX TP1 is based on a physical bus, every connected device receives every message on the very same bus/line. This eases the monitoring of transmitted data. If a password is in use, it would be transmitted in clear text. Therefore, the password can easily be captured using *eibd* on line level.

At the IP-based backbone level the attack surface is even larger, since there are already a lot of tools available that can capture and manipulate UDP/IP traffic. There-

[1] http://www.auto.tuwien.ac.at/~mkoegler/index.php/eibd

fore, an attacker can either use the *eibd* software or other tools like *tcpdump* to monitor IP backbone traffic. In our test setup, the tools *tcpdump, tcpreply* as well as the python library *scapy*[2] were used to replay and craft arbitrary packets. To capture or modify KNXnet/IP traffic in a switched network, an attacker can employ classic network based attacks like ARP cache poisoning[3] to redirect the traffic flow. Since KNXnet/IP makes extensive use of IP multicast messages, an attacker can alternatively join the appropriate IGMP[4] multicast group and thereby would receive all group communication. The latter would be less intrusive and therefore harder to detect.

2.4. Attack mitigation in the current version of KNX

Since there are no implementations of any security extension to KNX available, currently the most effective way to secure a KNX installation is a strict physical and/or logical separation of the KNX network. This separation should be as strict as possible, which prohibits the use of KNX over Power-line, RF or WLAN. For the line area, this requires the use of an elaborate TP1 cabling. Therefore, TP1 cables of a line should be placed in a way that they can not be accessed easily from another security area. For example, if KNX is used in an access control system which controls the opening mechanism of a door, special care must be taken that there are no easily accessible KNX switches, actuators, or sensors of the very same line on the outside that door. For the IP backbone area this implies complete physical separation of the backbone network, or the strict use of virtual private networks (VPN).

3. Security analysis of KNXnet/IP Secure

This section describes the general structure of the KNXnet/IP Secure protocol and highlights theoretical and practical security issues. The information on KNXnet/IP Secure provided in this section is obtained from the specification document [1]. In this document, also a detailed description of KNXnet/IP Secure is given. The specification is available for all KNX members. To get a basic understanding of KNXnet/IP Secure this paper briefly outlines its functionality.

At the time of writing, the current version of the specification was Application Note 159/13 v02 (last modified 2013.05.13). Because of the draft status of this document, there is still a lack of clarity regarding some details of the KNXnet/IP Secure protocol. If assumptions had to be made to further analyze the specification, because of unspecific or contradicting content, then these assumptions are described and further justified particularly.

Because it is important to get a profound knowledge of possible attack vectors, some attacks that might be considered theoretical are mentioned in this paper. These attack vectors are prefixed with the term *theoretical*. Although these attacks are not considered practical at the time of writing, they might pose unnecessary attack surface, are important during implementation, or gain relevance in future releases of the specification.

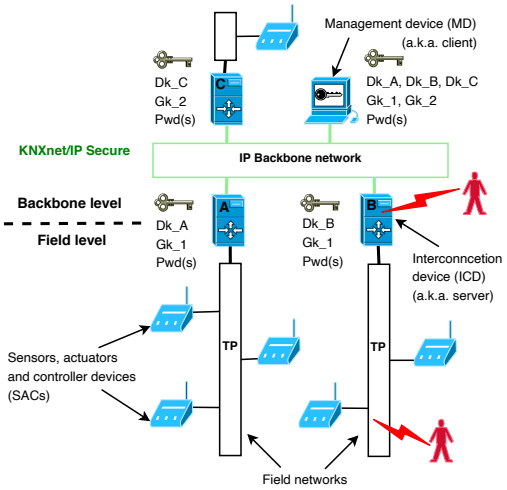

Figure 2: A two tier KNX BAS which uses KNXnet/IP Secure at the backbone level

3.1. The basic outline of KNXnet/IP Secure

KNXnet/IP Secure is a security extension for KNXnet/IP that aims to be backward compatible. This means no changes to the underlying KNX and KNXnet/IP protocol stack are required. The KNXnet/IP traffic is encapsulated in KNXnet/IP Secure wrapper frames which should provide confidentiality, integrity, freshness and authenticity. In other words, there are three logical layers encapsulated in one KNXnet/IP Secure UDP/IP datagram: *KNXnet/IP_Secure(KNXnet/IP(KNX))*.

KNXnet/IP Secure predominantly uses symmetric cryptography mechanisms. More precisely, it uses the Advanced Encryption Standard (AES) with 128 bit as a block cipher for all modes of operation required in the specification. There are two types of communication in KNXnet/IP Secure, namely *unicast communication* and *multicast communication*. Figure 2 shows a building automation network that employs KNXnet/IP Secure at the backbone level. The used secret keys (Dk, Gk, pwd) are also displayed. Note that in the context of KNXnet/IP Secure the terms "*interconnection device*", "*server*" and "*device*" are used interchangeably. The term "*client*" or "*tool*" refers to the management device which runs some management software.

Multicast communication: This type of communication should secure the traffic between members of one group. Thereby, it exclusively relies on symmetric cryptography schemes. The mode of operation used is a modified version of CCM[5]. Strictly speaking, it is a customized combination of counter mode (CTR) and

[2]https://pypi.python.org/pypi/scapy

[3]ARP stands for Address Resolution Protocol and is responsible for the mapping between link layer address and network layer address.

[4]IGMP stands for Internet Group Messaging Protocol.

[5]CCM stands for Counter with CBC-MAC and is a combined mode that defines how to use CTR in combination with CBC-MAC. CCM is

Cipher Block Chaining Message Authentication Code (CBC-MAC). There is also a special case in group communication where confidentiality is not required and only CBC-MAC is used. This communication type uses a pre-shared secret called *Group key* (Gk). This key is unique for every IP multicast group. The same group key can be found on every device that is in the same group. A device can only be in one IP multicast group at a time: *"For a device, at any given time, a single key shall be used for secure IP multicast communication."* [1]. The group key has a size of 128 bit. After a factory reset, the group key shall be zero. In order to be able to add members to a group later on, the group keys need to be present on the according management device.

Unicast communication: Unicast traffic is mainly used for configuration purposes, thus securing the communication between a management device and an interconnection device. To achieve perfect forward secrecy, KNXnet/IP Secure uses Elliptic Curve Diffie Hellman (ECDH) key exchange algorithm over NIST curve *K-283* [9, 10]. To authenticate each partner in the communication, two pre-shared secret keys are used:

- *Device authentication code* (Dk): Every device has one device authentication code. The purpose of this key is to authenticate the device to the client (e.g. management software that wants to configure a device) during unicast communication. Hereby, the size of the key is 128 bit. After a factory reset, the device authentication code is set to the factory default setup key printed on the device. In order to verify the identity of the device, the device authentication codes of the installation need to be present on the according management device.

- *Passwords* (Pwd): Every device has one or more passwords to provide different levels of access. The purpose of a password is to authenticate the client (e.g. the management software) to the device. Therefore, an array of passwords is stored on every device. The first element of this array holds the password for administrative access to the device. It is possible to define additional passwords for user level access. These passwords are stored in an array and can be identified over their individual position in the array. Every password consists of the lowest 128 bits of a SHA-256 hash of the actual password or phrase. After a factory reset, the password is set to the empty string which results in "27ae41e4649b934ca495991b7852b855". Although this value is called "password", it is basically a hash that is used as a password. It needs to be present on both ends of the communication. If not stated differently, the term "password" in this paper refers to this type of value.

specified in NIST SP 800-38C[7] and RFC 3610[8]

3.2. Impact of device compromise

The group key, the device authentication code as well as the password hashes which are used for authentication have to be stored on every KNXnet/IP device. The password hash might be the same on every device:

*"For practical reasons, a user may choose to opt for the **same password set in all devices** of this installation project."* [1].

The group key is the same for every member of the group and should have an unlimited lifetime:

*"The multicast group key shall be stored in every device belonging to the IP multicast group. It is transferred once during device commissioning using a secure unicast DeviceManagement[sic] connection. The key shall be transferred via the property PID_SECURE_IP_GROUP_KEY. The key shall have an **unlimited lifetime.**"* [1].

An attacker who is able to get physical access to one device, might be able to dump the content of the chip or the memory and read out all keys, including the 16 bytes password hash (Dk, Gk, pwd). With the gained knowledge an attacker might be able to perform the following actions:

1. An attacker might authenticate himself as a valid group member to the group of the compromised device using the group key. Every group member uses the same group key (Gk) for authentication and encryption. This generally enables every member of a group to impersonate every other member of the same group. Therefore, the property of *non-reputation* can not be achieved for group communication. In turn, this implies that if an attacker is able to compromise **one** member of the group, the whole group should be considered compromised.

2. An attacker might authenticate himself as a valid management- or user-client to any device that accepts the captured password hash (pwd). According to the specification, this might be any device in the installation. If the password hash to authenticate as a management-client is the same for every device, the attacker would be able to reconfigure every device in the installation. In other words, in such a scenario the security, of the installation has been breached.

3. An attacker might authenticate himself as a valid device to the genuine management- or user-client when a unicast connection is made to the compromised device. This can be achieved using the extracted device authentication code (Dk) of the compromised device. Since both devices rely on the same ECDH shared secret for confidentiality and integrity of the established conversation, the property of *non-repudiation* can not be fulfilled for unicast communication.

4. Since multicast communication exclusively uses a pre-shared group key, this communication cannot

fulfill the property of *forward secrecy*. Therefore, if a device of a group gets compromised, the attacker can decrypt every previously sent message that was encrypted using the extracted group key. This of course requires that the attacker has recorded the encrypted messages that have been sent in this group.

3.3. Replay of multicast communication

The group key will be used as encryption key to provide confidentiality as well as integrity in multicast communication. The group counter (GID), which is basically a 48 bit timestamp of the current system clock with a millisecond resolution, is used as a sequence identifier for a replay protection mechanism. This value must not decrease and will therefore be set to the highest value received during synchronization. For a detailed description of the synchronization mechanism refer to [1].

3.3.1 Replay after downtime

If one device is powered-down, all group messages sent by other group devices in the meanwhile can be replayed to this device if it goes online again. This is possible until it is synchronized again with the other group members. This resynchronization can be postponed by cutting the network connectivity to the rest of the group. In such a scenario, recorded messages sent in the remaining group can be replayed at will to the previously powered-down device. Despite this circumstance, the specification only requires the persistence of the GID, whereas the usage of RTC is optional: *"The group sequence counter must be monotonically increasing at least every millisecond. It shall under no circumstances be decremented ... To achieve this, the sequence counter must be* ***persisted during power-off conditions****. Even better it should be increased during power-off conditions using an RTC"* [1].

A built-in counter which is increased while the device is powered-down would prevent this attack scenario to a certain degree. However, in such a scenario the power supply must be assured for every device in the group. Devices that do not have a internal clock are still vulnerable until they are synchronized with the rest of the group.

3.3.2 Replay within the latency tolerance

The latency tolerance is defined as follows: *"If the received frame is older than the time span given in PID_SECURE_MULTICAST_LATENCY_TOLERANCE (typically a few seconds), the frame shall be discarded. This means that frames with slightly past time values shall be accepted to account for network latency."* [1]. The specification does not suggest a suitable value for the size of the tolerance window. If the latency tolerance window is defined to be relatively large, an attacker might be able to replay frames within the duration of the window. This can be the case if a device sends the group message "increase temperature by one degree" and the latency tolerance window is configured to be five seconds. An attacker might be able to replay the sent message several times within these five seconds. Thereby, he would increase the temperature by more than one degree.

The KNXnet/IP Secure specification accounts for this issue within one sentence and thereby leaves the solution to the the application layer: *"on application layer for critical applications includes an additional device specific counter that further minimizes the risk of replay attacks and reduces the need to tweak the latency tolerance on KNXnet/IP"*[1].

3.3.3 Theoretical: Replay after time manipulation

If an attacker is able to manipulate the time of **one** KNXnet/IP Secure device in a group, he will be able to set a time value that would soon exceed the 48 bit timestamp value. This would lead to an overflow or even a wrap-around of the timestamp. After a wrap-around, all previously sent group messages that have been captured by the attacker become "valid" again and can be replayed.

3.4. Counter repetition in AES-CTR

During multicast communication, it might be possible that two devices send a message at exactly the same time obviously using the same GID and therefore an identical T_0 value in AES-CTR[6]. The counter T_x for AES-CTR in KNXnet/IP Secure is composed of the 6 byte timestamp, 9 null bytes and one byte containing the counter which starts with 00 and gets incremented for every encrypted block.

$$T_0 = GID||00...00||00 \quad (1)$$

$$T_1 = GID||00...00||01 \quad (2)$$

The usage of the same counter value T_0 in combination with the same group key can break the confidentiality of group communication because of the "Two-Time-Pad problem" [11].

AES-CTR uses the counter and a secret key to generate a key-sequence which is then combined with the plain text using a bitwise exclusive-or operation. If two plain text blocks are encrypted with the same key and the same counter, an exclusive-or operation on both cipher texts yields the exclusive-or of the two plain texts:

$$c_1 = p_1 \oplus E_k(ctr) \quad (3)$$

$$c_2 = p_2 \oplus E_k(ctr) \quad (4)$$

$$c_1 \oplus c_2 = p_1 \oplus p_2 \quad (5)$$

This $p_1 \oplus p_2$ is vulnerable to frequency analysis, or if the plain text of one message is known the plain text of the other message can be calculated. This leads to the conclusion that the confidentiality of group communication can not be guaranteed if one device of the group sends a SECURE_WRAPPER frame in exactly the same millisecond as any other device of the group.

[6] AES-CTR stands for Advanced Encryption Standard in Coutner Mode of operation.

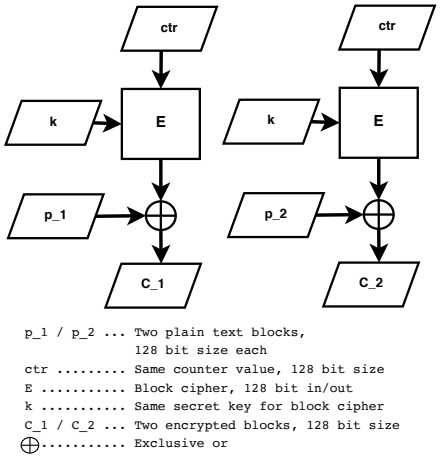

Figure 3: Identical counter values in different AES-CTR invocations

An attacker might utilize this circumstance to trigger encrypted group communication of one device in a group (e.g. a switch) at exactly the same time, when the group device he wants to attack starts his encrypted communication.

3.5. Missing Key-derivation function in ECDH

KNXnet/IP Secure does not define a key-derivation function for ECDH shared secrets which are used in unicast communication. The specification document defines the generation of the shared secret as follows:

*"The session key shall be calculated on both sides of the connection as the **first 16 octets** of the shared Elliptic-Curve Diffie-Hellman secret."* [1].

*"The **lowest 16 octes** of this number shall be used as session key for the symmetric session runtime encryption"* [1].

Despite these two contradicting explanations[7] on how to derive the secret, there is no direct reference or explanation of a key-derivation function given in the specification document. Using the derived secret directly as a key, is not considered to be "good practice".

It is advised to use a key derivation function to destroy any dependencies that may exist in the calculated shared secret. In [12], this circumstance is outlined for classical DH key exchange as follows:

"Can Alice and Bob use g^{ab} directly as their shared secret key? The answer depends on the group representation, and likely to be no. Indeed, for many cryptographic groups the length of a group element is much longer than its order p. Security of most keyed cryptographic primitives is argued under the assumption that the key is a truly random binary string of some fixed length. An element of group G cannot possibly satisfy the assumption if the size of the group is smaller than its representation." [12]

In case of K-283, the elliptic curve is defined over the finite (galois) field $\mathbb{F}_{2^{283}}$ containing 2^{283} elements. The order of the elliptic curve is $E(\mathbb{F}_{2^{283}})$ and hence the number of points on $E(\mathbb{F}_{2^{283}})$ is defined as $\#E(\mathbb{F}_{2^{283}})$. Therefore, only using data bits from points on the elliptic curve $E(\mathbb{F}_{2^{283}})$ reduces the number of possibilities, since $\#E(\mathbb{F}_{2^{283}}) < 2^{283}$. This is the case because not every element of the underlying field is a coordinate of a valid point on the elliptic curve.

[7] Depending on the interpretation of "first".

3.6. Denial-of-Service attack vectors

This section outlines attack vectors for Denial-of-Service (DoS) attacks which exploit properties of the KNXnet/IP Secure protocol for unicast connections. DoS attacks which are based on an imbalance of resources (e.g. bandwidth, CPU, RAM), or DoS attackes based on underlying protocols (e.g. ARP cache poisoning) are not in scope of this paper.

3.6.1 SECURE_CHANNEL_RESPONSE flooding

It is possible to deny the establishment of a unicast connection between a device and a client during an early stage of the secure connection setup. When *"the server runs out of free secure channels or is too busy to fulfill any additional secure channel requests, it shall send an empty SECURE_CHANNEL_RESPONSE frame with a connection index field of 00h"* [1]. This frame cannot be secured by a generated session key since no resources are available. Therefore, it can be spoofed by an attacker in order to deny a successful connection handshake.

3.6.2 SECURE_CHANNEL_REQUEST flooding

The first step of a secure connection setup is the transmission of a SECURE_CHANNEL_REQUEST including the ECDH public key parameter Q_A (called X in the specification [1]). Figure 4 shows this process. Since this first step does not involve any knowledge of secret key information, everyone can initialize an ECDH handshake and spoof the sender information. A DoS attack on a KNXnet/IP Secure device does not need to execute the third step of the secure connection setup. Therefore, an attacker can randomly choose some sender information. Due to the stateless nature of UDP, spoofing of sender information is easy for KNXnet/IP Secure connections since there are based on UDP per default.

An attacker who wants to execute a DoS attack does not need to actually calculate an ECDH public/private key pair (Q_A, d_A). Depending on the implementation of the device, the calculation of the ECDH public/private key pair (Q_B, d_B) is started right after receiving Q_A. The specification [1] does not explicitly require that the received ECDH public keys are validated. This would make it possible for an attacker to send random bit strings in the length of Q_A to initiate the computationally expensive calculation of Q_B and the shared secret k. To prevent an attacker from sending random bit strings in the length of Q_A, the device should first check if Q_A is a valid point

on the elliptic curve K-283 before it starts the computationally more expensive calculation of Q_B and the shared ECDH secret k for this connection.

If the device checks the ECDH public key Q_A before it computes Q_B and k, the attacker needs to precalculate points on the elliptic curve K-283 and initialize a SECURE_CHANNEL_REQUEST for every point he has calculated[8]. By sending $2^{16} = 65536$ SECURE_CHANNEL_REQUESTs it would be possible to occupy all secure channels of the device for two minutes.

3.7. Theoretical: unicast authentication reuse

The following sub-section describes a theoretical attack on the authentication procedure in KNXnet/IP Secure unicast communication. Figure 4 shows the authentication process for unicast connections. To secure unicast communication against replay attacks, a monotonically increasing sequence identifier called SID is used in every SECURE_WRAPPER frame.

The outlined attack might be impossible to accomplish in a real world scenario, but it shows some shortcomings in the design of the protocol. The shortcomings that are responsible for this attack vector can be summarized as follows:

- The usage of CBC-MAC with a static value $B_0 = Q$, where Q is the message size which is a fixed value for the KNX frames used at that stage of the protocol.[9]
- The combination of a static B_0 and an exclusive-or operation on the two ECDH public key values.

This offers the theoretical possibility for an attacker to execute a successful authentication without knowing the secret key used for CBC-MAC (*device authentication code*). This is theoretically possible because, the same CBC-MAC inputs (B_0*, secret key, plain text*), produce the same cipher texts.

In the context of unicast authentication, the *plain text* consists of the exclusive-or combination of the two ECDH public keys Q_A, Q_B. Since the B_0 values are the same for every message at this point (because of their equal size), identical *plain texts* produce the same CBC-MAC *cipher texts*.

The problem is, that there can be more than one exclusive-or combination of two ECDH public keys, which yields the same result and therefore the same *plain text*. An example consisting of some arbitrary values which do not represent valid ECDH public keys is depicted below:

$$Q_A = 01001_2\ Q_B = 10111_2\ Q_A \oplus Q_B = 11110_2 \quad (6)$$
$$Q_Y = 01100_2\ Q_X = 10010_2\ Q_Y \oplus Q_X = 11110_2 \quad (7)$$

[8]Choosing a point is faster than computing $Q_A = d_A * G$.

[9]The sequence identifier (SID) cannot be included into B_0 here because during the exchange of SECURE_CHANNEL_REQUEST and SECURE_CHANNEL_RESPONSE, there is no SID available. The SID is only part of a SECURE_WRAPPER frame.

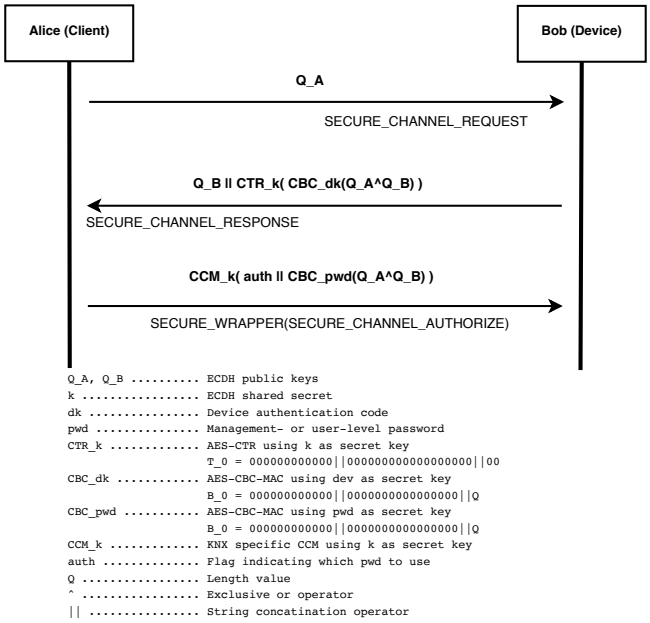

Figure 4: KNXnet/IP Secure authentication for unicast connections [1]

Let r, u, v be bit strings of 283 bits in size. For every possible value of r, there are 2^{283} possible bitwise exclusive-or combinations of u and v, where $r = u \oplus v$. Therefore, there are $2^{283} - 1$ *collisions* for r. It should be noted that not all values for u, v of these $2^{283} - 1$ combinations produce valid points on the elliptic curve K-283.

Supposingly, the KNXnet/IP protocol works as described in [1] and depicted in Figure 4, the result of $Q_A \oplus Q_B$ is used as input value for a CBC-MAC with an initialization vector of $B_0 = Q$. At that point there is no sequence identifier (SID) in the KNX frame. Such frames will always have the same length and hence the same value Q. Therefore, the same plain text will produce the same CBC-MAC. This means that an attacker which is in possession of a valid CBC-MAC $CBC_{dk}(Q_A \oplus Q_B)$ might be able to reuse this CBC-MAC in a different authentication scenario, where he does not know the key dk. This is only possible when $Q_X \oplus Q_Y = Q_A \oplus Q_B$ and thus $CBC_{dk}(Q_X \oplus Q_Y) = CBC_{dk}(Q_A \oplus Q_B)$. Such a reuse of an old CBC-MAC is possible without being able to compute the CBC-MAC.

3.7.1 Attack scenario

The described attack has two stages and aims to impersonate a KNXnet/IP Secure device (*Bob*), without knowing any secret key information. To carry out this attack, an attacker (*Mallory*) must be able to observe and/or alter all connections to *Bob*.

Stage one: KNXnet/IP Secure has a design weakness that enables an attacker to harvest valid CBC-MACs. For this purpose, *Mallory* initiates a valid device authentication procedure with *Bob* by sending Q_{Mx}. This step can be performed without knowing any secret key. When

Mallory receives a Q_{Bx} as a response, he finishes the ECDH handshake by calculating the shared ECDH secret. Then *Mallory* uses this secret to decrypt the attached AES-CTR payload and stores the valid CBC-MAC $CBC_{dk}(Q_{Mx} \oplus Q_{Bx})$, generated by *Bob*, in his database. *Mallory* also stores the values for $Q_{Mx} \oplus Q_{Bx}$ and the public private key pair (Q_{Mx}, d_{Mx}). A single database entry would consist of the following values: $\{Q_{Mx} \oplus Q_{Bx}, CBC_{dk}(Q_{Mx} \oplus Q_{Bx}), Q_{Mx}, d_{Mx}\}$, which would need approximately $37 + 16 + 37 + 36 = 126$ octets of space. This procedure is repeated several times to build up a database.

Stage two: In this stage *Mallory* tries to impersonate *Bob*. Therefore, *Mallory* needs to redirect the traffic of *Alice* that is intended for *Bob*, such that *Mallory* can listen and respond to it. If *Alice* tries to connect to *Bob*, *Mallory* receives Q_A from *Alice*. Then, *Mallory* needs to find a Q_M such that $Q_A \oplus Q_M = Q_{Mx} \oplus Q_{Bx}$, where $Q_{Mx} \oplus Q_{Bx}$ represents a previously stored value from *stage one*. While calculating Q_M is simple[10], finding the according scalar value d_M is an instance of the elliptic curve discrete logarithm problem. Therefore, *Mallory* is able to decrypt/encrypt the following communication, if and only if he has the according public private key pair (Q_M, d_M) in his database. The procedure to look up the corresponding value in the database would be the following:

Mallory takes the received Q_A, which was chosen by *Alice*, and uses exclusive-or to combine it with one stored value for $Q_{Mx} \oplus Q_{Bx}$ in his database, $Q_{candidate} = Q_A \oplus (Q_{Mx} \oplus Q_{Bx})$. The resulting value $Q_{candidate}$ is then searched for in the database. If such a value $Q_{candidate}$ is found in the database the public-private key pair $(Q_{candidate}, d_{candidate})$ can be used to impersonate *Bob*. In case no public key value in the database matches $Q_{candidate}$, *Mallory* uses the next value for $Q_{Mx+1} \oplus Q_{Bx+1}$ from the CBC-MAC database and recalculates the exclusive-or result $Q_A \oplus (Q_{Mx+1} \oplus Q_{Bx+1})$ to produce another $Q_{candidate2}$ and searches the database again for this candidate.

3.7.2 Applicability

There are $1.554135 * 10^{85}$ points on the elliptic curve K-283[9]. Storing only 1% of all points would require approximately $1.03 * 10^{73}$ terabyte of space. Moreover, not all combinations (x, y) where $x, y \in \mathbb{F}_{2^{283}}$ are valid points on the elliptic curve described by K-283[13].

Because of the amount of resources required, this is considered as a theoretical attack which might be impossible in a real world scenario. Nevertheless, their exists the theoretical possibility that a valid CBC-MAC can be stored and reused during authentication to impersonate another user/device.

[10] $Q_M = Q_A \oplus (Q_{Mx} \oplus Q_{Bx})$

3.8. Theoretical: CBC-MAC forgery

The general functionality of CBC-MAC can be found in [14] and is outlined as follows:

- The initialization vector B_0 of a CBC-MAC is combined with the first plain text block p_1 using exclusive-or, $p_1 \oplus B_0$.
- The result is fed into the encryption routine $E_k(p_1 \oplus B_0) = c_1$.
- The encrypted first block c_1 is then used for chaining. For the second block of data $E_k(p_2 \oplus c_1) = c_2$.
- The final result of the CBC-MAC calculation is the cipher text of the last block.

Due to the functionality of CBC-MAC, it is bad practice to transfer a non-static initialization vector B_0 over an insecure channel along with the protected data and its CBC-MAC. The reason for this lies in the composition of the first input to the encryption function $E_k()$. The initialization vector B_0 is combined with the first plain text block using exclusive-or before the first call of the encryption function $E_k(p_1 \oplus B_0)$. If an attacker is able to find a B'_0, p'_1 such that $p_1 \oplus B_0 = p'_1 \oplus B'_0$, then this would result in the same first CBC-MAC block $c_1 = E_k(p_1 \oplus B_0) = E_k(p'_1 \oplus B'_0)$. In this case, the attacker has found a successful collision. Some background information on this kind of attack can be found in [14, 7, 8].

In case of KNXnet/IP Secure, there is one occasion where CBC-MAC is used without further encryption, namely in the SECURE_GROUP_SYNC_* frames. These frames consist of a 6 byte header, a payload represented as an 8 byte timestamp and the *CBC-MAC* of this data. To successfully check the CBC-MAC, the receiving side needs to calculate the same CBC-MAC using the same group key as well as the same initialization vector B_0. Therefore, the receiver has to use the included timestamp as initialization vector B_0 in order to compute the correct result.

The applicability of this attack depends on the exact representation and ordering of the GID value as well as on the endianness of B_0 (little- or big endian byte order). Since the specification is not clear regarding the mentioned points, it is not certain if this attack is applicable in KNXnet/IP Secure.

4. Generic security concept for IP backbones

In [2], a *generic security concept for IP backbones* is presented. The described approach uses asymmetric cryptography to establish a symmetric key set which is used to secure the communication among the involved devices. For the asymmetric cryptography part, a Certification Authority (CA) is required. The initial setup requires the generation and distribution of certificates for each involved device, such that every device is in possession of the CA certificate, its own certificate (which is signed by the CA)

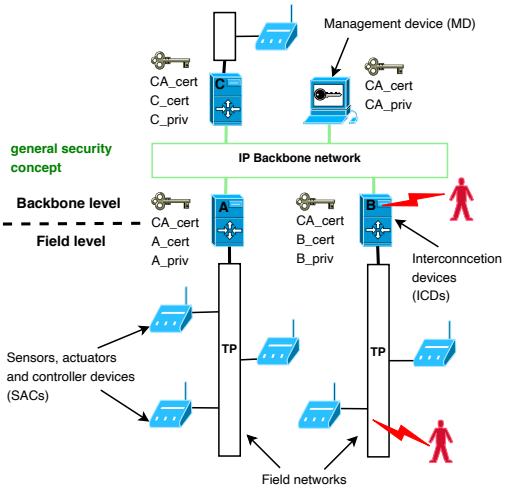

Figure 5: Distribution of certificates and keys in [2]

and the according private key part of its certificate. Figure 5 shows the distributed key material.

The symmetric key for the communication is generated by a *key set distribution protocol* which elects a coordinator that is responsible for generating and maintaining the key set. The key set contains *"a secret key SK_C for en-/decryption, a secret key SK_{MAC} for MAC calculation, and the currently valid sequence counter C"*[2]. In a *unicast scenario* the server would act as the coordinator and generate the symmetric keys after a client wants to establish a secured connection with the server. To protect the unicast key set distribution from replay and spoofing attacks, the participants sign their individual requests and make use of random nonce values. In case of unicast communication, there is no need for an elaborate election mechanism, since the server takes the role of the coordinator per definition.

In case of multicast communication, an election procedure has to be performed to determine the coordinator. This election mechanism differentiates between three cases [2]:

Device alone in a (new) group: This is the case when the device does not receive a reply from any other group member. In such a scenario, the device claims the role of the coordinator for the group. If two new devices compete on the role of the coordinator of this freshly generated group, the device *"with the lower IP address shall become coordinator"*[2].

Coordinator available: If a group coordinator is available it responds to the new device and performs the same steps as in a unicast scenario to communicate the group key set to the asking device.

Coordinator unavailable: In this scenario, the coordinator of the group is not available, because of a crash for example. In this case, the remaining devices in a group notify the new device that there is already an active key set within this group. The new device then contacts one of the members of this group to learn the current key set. Basically, this is done the same way as in a unicast connection. After this step, the new device becomes the new coordinator of this group.

4.1. Lack of forward secrecy

The proposed protocol lacks the property of forward secrecy for unicast as well as for multicast communication. To exchange key information, the election mechanism for unicast communication uses the public key of the client to encrypt the secret key which later gets used to symmetrically encrypt each message between client and server. For multicast communication, the situation is the same. If an attacker records the encrypted traffic and later gets in possession of the private key of a participant, he would be able to decrypt the key set distribution message which contains the used shared secrets for the communication. With this key set he is able to decrypt all previously stored messages that have been encrypted with this key set.

4.2. Lack of non-repudiation

Multicast communication: If an attacker is able to compromise one device of a group – for example, by having physical access to the device – he is in possession of the current key set of the group to which the device belongs to. Additionally, he is in possession of the certificate and the according private key of this device. With the current group key set the attacker can compromise the communication of the whole group by successfully spoofing any device within this group. This is possible because of the symmetric schemes used for communication (HMAC, AES-CBC). To make this kind of attack impossible, an asymmetric signature based authentication mechanism can be used instead of a symmetric HMAC based procedure. This change should be possible without deep changes to the protocol since there are already certificates available for each device.

Unicast communication: Since both communication partners share the same secret keys it can not be proven that one of the participants sent a certain message since the other one could have faked it.

4.3. Resynchronization of sequence counter left open

The proposed approach does not specify any details on the generation and synchronization of the *sequence counters* for group communication. A synchronization mechanism for multicast communication is not trivial in general. Several special cases have to be considered. For instance, if one device does not receive a group message including the current group sequence counter, this device is out-of-sync and has to synchronize itself again with the group. This procedure is not explicitly described. The only description given on the sequence number is quoted as follows and does not refer to resynchronization: *"To ensure data integrity and data freshness, HMAC [11] in combination with SHA 256 [12] and a sequence number are used."* [2].

5. Conclusion and future work

In this paper it has been shown that attacks on current KNX installation are possible and that all required tools already exist. At the time of writing, these attacks can only be mitigated by employing a strict physical separation of KNX networks. To address the security issues in KNX, two security extensions have been reviewed regarding their security properties. The Table 1 sums up the analysis and shows a comparison between the covered protocols regarding their individual security properties. Since it is difficult to sum up the individual strengths and weaknesses of the selected protocols into one of three symbols, this table gives only an informal overview.

Property	**KNX**	**KNXnet/IP Secure**	**Generic**
Authentication	**-/-**	**+/∼**	**+/+**
Authorization	**-/-**	**+/-**	**-/-**
Non-repudiation	**-/-**	**-/-**	**-/-**
Integrity	**-/-**	**+/+**	**+/+**
Freshness	**-/-**	**+/∼**	**+/∼**
Confidentiality	**-/-**	**+/∼**	**+/+**
Forward secrecy	**-/-**	**+/-**	**-/-**
Availability	**-/-**	**-/-**	**-/-**

Table 1: Overview of some provided security properties. Most of the listed security requirements for BAS have been specified in [5]. "+" stands for fulfilled, "-" stands for not fulfilled or not specified and "∼" stands for partially fulfilled. The most left value in a cell stands for unicast communication, the most right value represents multicast communication.

The results of this paper show that further evaluation and testing is required before any security extension can be widely deployed. For this purpose, possible attacks and attack vectors have to be well-defined. Since the reviewed protocol descriptions still leave space for different interpretations, this paper is solely an initial step and does not intent to be a complete analysis of both extensions. Possible attacks are not limited to cryptography and software level. Furthermore, hardware and device based attacks have to be taken into account, like the possibility of extracting secret key information from a KNX device. Further research is also required in analyzing the performance impact of cryptographic mechanisms to determine the applicability of strong cryptography in building automation.

In the future, BASs will integrate more and more services from different domains such as access control, smart metering and many others. Several publications in the field of building automation security aim at generalizing the security problems and demands of BAS and try to apply generalized concepts to solve these issues. Due to the increasing complexity of BAS and their individual requirements, more tailored security approaches could be even more promising. Since different services have different security requirements, appropriate security mechanisms and protocols have to be applied depending on the security profile of the different parts of a BAS. Because of the large number of use-cases, different resource capacities and varying security requirements, a one size fits all approach in building automation security might not be feasible.

References

[1] KNX Association, *KNXnet/IP Secure Protocol Specification - Application Note 159/13 v02*, 2013.

[2] W. Granzer, D. Lechner, F. Praus, and W. Kastner, "Securing IP Backbones in Building Automation Networks", in *Proc. 7th IEEE International Conference on Industrial Informatics (INDIN '09)*, June 2009, pp. 410–415.

[3] W. Granzer, F. Praus, and W. Kastner, "Security in Building Automation Systems", *IEEE Transactions on Industrial Electronics*, vol. 57, no. 11, pp. 3622–3630, Nov. 2010.

[4] W. Granzer, *Secure Communication in Home and Building Automation Systems*, PhD thesis, Vienna University of Technology, Institute of Computer Aided Automation, Automation Systems Group, Feb. 2010.

[5] W. Granzer and W. Kastner, "Security Analysis of Open Building Automation Systems", in *Proc. 29th International Conference on Computer Safety, Reliability and Security (SAFECOMP '10)*, 2010, pp. 303–316. SAFECOMP '10.

[6] KNX Association, *KNX System Specifications, KNX Standard, v2.0*, 2009.06.09.

[7] NIST, *NIST SP 800-38C, Special Publications - Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality*, 2007.03.

[8] Internet Engineering Task Force (IETF), D. Whiting, R. Housley and N. Ferguson, *RFC 3610 - Counter with CBC-MAC (CCM)*, 2003.09.

[9] NIST, *FIPS PUB 186-3, Federal Information Processing Standards Publication, Digital Signature Standard*, 2009.

[10] NIST, *Recommended elliptic curves for federal government use*, 1999.07.

[11] J. E. Joshua Manson, Kathryn Watkins and A. Stubblefield, *A Natural Language Approach to Automated Cryptanalysis of Two-time Pads*, 2006.

[12] I. Mironow, *Hash functions: Theory, attacks, and applications*, 2005.11.05.

[13] NIST, *NIST SP 800-56A, Special Publications - Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography*, 2007.03.

[14] FIPS PUB 113, Federal Information Processing Standards Publication, Computer Data Authentication, "http://www.itl.nist.gov/fipspubs/fip113.htm", Accessed: 2013-05-09.