Securing IP Backbones in Building Automation Networks

Wolfgang Granzer, Daniel Lechner, Fritz Praus, Wolfgang Kastner
Vienna University of Technology, Automation Systems Group
Treitlstrasse 1-3, Vienna, Austria
Email: {w,dlechner,fpraus,k}@auto.tuwien.ac.at

***Abstract*— The use of IP networks as common backbone is becoming of increased interest in today's building automation systems (BAS). With the use of IP also new attack scenarios that threaten the overall security of BAS are introduced. Due to the absence of native security mechanisms in IP and because of its long standing and pervasive use in the IT world, many vulnerabilities exist that are well-known to attackers. To counteract these threats, this paper presents a generic concept to secure IP backbones that is tailored to the use in building automation. A main advantage of the concept is its flexibility. Due to the used protocol architecture, it is applicable to available BAS standards without the need of an adaption of existing BAS protocols. As a proof-of-concept, a prototype implementation for the KNX standard is also presented.**

I. Introduction

With the integration of security-critical services into building automation systems (BAS), a protection against malicious interference is of utmost importance. Therefore, a comprehensive security concept that incorporates security in every stage of the system's life cycle is necessary [1]. Since modern BAS are implemented as distributed systems where devices are connected to a common building automation network (BAN), a mandatory step towards an overall secure BAS is to secure the communication as part of functional security.

Today's BAS are implemented following a two-tier model. Field networks are interconnected by a common backbone [2]. At the field level, still, robust, flexible, and cost-efficient field bus technologies are used. For the backbone, IP technologies have gained wide-spread acceptance. This is due to the fact that buildings are normally equipped with IP based office networks. Since a BAS typically demands only moderate response times, it is feasible to share the medium with non control data as long as acceptable performance can be guaranteed. Furthermore, the costs for IP cabling and network interface hardware are rapidly decreasing and so even small embedded microcontrollers typically found in the building automation domain can be equipped with an Ethernet interface chip.

From the security point of view, especially the IP based backbone is prone to security attacks [3]. This is for various reasons. First, since IP as well as the underlying data link protocol (e.g., Ethernet) do not provide native mechanisms for secure communication, many well-known vulnerabilities exist which may also be exploited in BANs. Second, since IP networks may be shared with other applications (e.g., office LAN) and interconnections to foreign networks are common, gaining access to the network may be easier. Third, due to the global view at the backbone, an adversary may interfere not only with the backbone but also with the field networks that are connected to it.

Due to these reasons, especially these IP backbones have to be protected. Section II gives an overview of BAS standards that already offer IP support. Section III lists the requirements that have to be satisfied once IP backbones come into play, followed by a discussion of IP security mechanisms typically employed in the IT domain. Since none of these mechanisms can be trivially mapped to meet the demands of BAS, a generic approach to guarantee secure communication in IP backbones for BANs is introduced (cf. Section IV). Finally, a proof-of-concept implementation for the KNX standard is presented in Section V.

II. Using IP networks as backbone

Today, various building automation standards that support the use of IP networks exist. The most important (and trade-neutral) open ones are BACnet, KNX, and LonWorks [2]. This section gives a brief introduction to these standards and reviews their supported functional security mechanisms.

A. BACnet/IP

In BACnet 2008, two different possibilities to use IP in BACnet internetworks are defined. First, a tunneling scheme where BACnet messages are encapsulated into UDP/IP messages is available. This scheme can be used to interconnect different BACnet networks via IP or to provide remote access over the Internet. Second, BACnet 2008 also specifies the use of UDP/IP as native data link layer protocol (BACnet/IP). Using this scheme, an IP network can be used as native BACnet medium which may host BACnet IP only devices.

BACnet 2008 defines security services that guarantee data confidentiality, freshness, integrity as well as mutual entity authentication. Due to several security flaws [4], these services were replaced [5]. Since BACnet security services are integrated into the network layer, they are applicable to both BACnet/IP and the tunneling mechanism. The drawback of BACnet security is the use of symmetric algorithms which make the presence of an online key server mandatory. Nowadays, the choice of asymmetric algorithms based on elliptic curve cryptography (ECC) would be beneficial as these are also suitable for embedded devices and eliminate the need

978-1-4244-3760-3/09/$25.00

for a key server [6]. Moreover, the use of a single key server introduces a single point of failure. Therefore, a scheme based on multiple key servers or a mechanism to elect a new key server in case of a failure is desirable.

B. KNXnet/IP

To be able to use IP in KNX networks, KNXnet/IP has been introduced [7]. Among other services, KNXnet/IP specifies KNXnet/IP tunneling and KNXnet/IP routing. Using tunneling, a KNXnet/IP client is able to establish a point-to-point connection to a KNXnet/IP server via an IP network. KNXnet/IP routing is used to interconnect KNX network segments using IP multicast exclusively. The interconnection is provided by KNXnet/IP routers which are responsible for the en-/decapsulation of KNX packets into UDP/IP packets. However, KNXnet/IP routing does not use UDP/IP as native KNX medium – the use of KNX IP only devices is not possible. To eliminate this restriction, there are investigations underway to define IP as native medium for KNX. At the time of writing, the specification of this extension has not been finished.

Regarding security, KNX only specifies a simple access control mechanism for limiting the management access to devices. Since it is based on clear-text passwords, it is not sufficient for security-critical environments [4]. In the KNXnet/IP specification, some rudimentary security guidelines are presented that are based on isolation (e.g., firewall, KNXnet/IP only Intranet) and on "Security by Obscurity" (e.g., use of non-standard IP addresses, rely on the missing expertise of an attacker). Since preventing physical access to the network by isolation is not always possible (e.g., WLANs) and "Security by Obscurity" is a technique that (if at all) provides only temporary protection, neither the access control mechanism nor the security guidelines provide an effective protection.

C. LonWorks/IP

[8] specifies the use of IP networks as transport medium for LonWorks networks. In LonWorks/IP, LonWorks messages are encapsulated into UDP/IP or TCP/IP messages. In contrast to KNXnet/IP, IP networks can be used as native network medium for LonWorks, thus, the use of LonWorks/IP only devices is also possible. Communication between LonWorks/IP devices can be performed using multiple unicast connections. To increase the effectiveness especially in larger LonWorks/IP networks, multicast as well as a mechanism called selective forwarding can be used.

In addition to the basic LonWorks authentication mechanism which cannot be considered secure anymore [4], LonWorks/IP defines its own security mechanism. This mechanism uses MD5 together with a shared secret to guarantee data integrity and freshness. Data confidentiality is not guaranteed. However, since MD5 is not collision resistant, the used mechanism is insecure. Furthermore, mechanisms to manage and distribute the used shared secrets are missing, too.

III. Security in IP backbones

As discussed in the previous section, neither KNXnet/IP nor LonWorks/IP provide mechanisms for an effective protection against security attacks at the backbone level. The security concept of BACnet/IP is more advanced and provides a solid base for securing IP backbones. The main drawback is the need for a centralized key server. Furthermore, the security concept cannot be trivially mapped to KNXnet/IP and LonWorks/IP since the security mechanisms are incorporated into the BACnet network layer.

A. Requirements

To provide a comprehensive security concept for IP backbones which is applicable to all three standards, the following security requirements have to be satisfied:

- *Mutual entity authentication*: To avoid that an adversary impersonates legitimate devices on the backbone, involved communication partners have to authenticate each other. Only authenticated devices shall be able to securely communicate with each other.
- *Secured channel*: To securely exchange data between authenticated entities, the data has to be transmitted through a so called secured channel. A secured channel uses cryptographic and/or physical techniques to guarantee data integrity, freshness, and confidentiality [9]. While data integrity and freshness are mandatory in most BAS, providing confidentiality may be optional if the non-disclosure of the transmitted data is not a strict requirement.
- *Support for multipoint-to-multipoint connections*: In BACnet/IP, communication is usually performed using multiple unicast connections – the use of multicast is possible, but not mandatory. In LonWorks/IP, the use of both communication types has been specified. Due to reasons of efficiency, the use of multicast is recommended in larger IP networks. Finally, in KNXnet/IP routing, using multicast is mandatory. Therefore, a comprehensive security concept that supports all three standards must provide services for unicast and multicast communication.
- *Support for embedded devices*: Devices typically found in the building automation domain are embedded systems with limited resources. Since cryptographic algorithms are both processing power and memory intensive, their use must not exceed the available resources.

B. IT security mechanisms

Due to the wide-spread use of the Internet, many security mechanisms for IP networks are already available. In the following paragraphs, their suitability for securing IP backbones in BANs is discussed.

1) Internet protocol security: Due to the lack of security of the IP protocol version 4 (IPv4), a security extension called *Internet Protocol Security (IPsec)* has been defined [10]. In IPv4, the support for IPsec is optional, whereas in the new version IPv6, IPsec is mandatory. IPsec provides mutual entity

authentication and guarantees data integrity, freshness, and confidentiality. To achieve this, various cryptographic algorithms can be selected (e.g., Triple-DES, AES, HMAC-SHA1 [11], [12]). For key exchange, the *Internet Key Exchange (IKE)* protocol is used. IKE uses asymmetric algorithms like RSA, ECC, or symmetric algorithms with pre-shared secret keys, alternatively.

In principle, IPsec can be used to secure IP backbones since it is intended to secure both unicast and multicast communication. However, if there is more than one sender, the sequence numbers have to be synchronized or receivers have to track one sequence number for each sender to avoid replay attacks. In both cases, changes to the current IPsec implementation would be required. Another problem is key distribution, since it is not possible to derive a common secret key in a group without requiring major changes to existing IPsec implementations. Furthermore, due to its high complexity, IPsec is not advisable for embedded devices.

2) Transport layer security: Secure sockets layer (SSL) and its successor *transport layer security (TLS)* [13] are used to secure communication between two entities. During the initial handshake, the entities are authenticated, the used cryptographic algorithms are negotiated, and shared secret keys are exchanged. After the initial handshake, these secret keys are used to establish a secured channel between the two entities. Like in IPsec, the initial key exchange is usually done using asymmetric algorithms while the secured channel is protected using symmetric algorithms exclusively.

TLS is very flexible with respect to the use of algorithms. [14] presents a microcontroller implementation of a complete secure web server, using HTTP and TLS. In this implementation, ECC is used for the asymmetric encryption. The presented prototype implementation proves that the available resources of a 4MHz 8-bit ATmega processor are sufficient. Since TLS is a unicast protocol (asymmetric key exchange is done using unicast algorithms – there are no extensions for multiparty key exchange), TLS as well as the implementation presented in [14] can only be used to secure unicast services. Securing multicast is not supported in a native way.

3) Virtual private networks: A *virtual private network (VPN)* is a secure, logical network that is built upon a possibly insecure one [15]. A VPN is transparent to the connected devices. Usually, each device opens a secure unicast connection to a centralized server where the whole network traffic to and from the device is tunneled through. The used cryptographic algorithms for protecting the network traffic depend on the implementation. A popular VPN implementation is OpenVPN which makes use of TLS.

The main drawback of VPNs is the need for a centralized server that has to route the whole traffic. Therefore, the server has to manage one connection for each connected client. For multicast communication, this concept is clearly inappropriate. Each multicast message has to be sent to the server where it is distributed to the clients. So the server has to decrypt each multicast message once and encrypt it again several times for all other clients. This results in a high demand on bandwidth, memory (for storing the different session keys), and computational power at the server. Thus, the server represents a bottleneck and a single point of failure for the whole network. Furthermore, the required resources would exceed the capabilities of an embedded device. Therefore, a VPN solution is of limited use for embedded networks.

IV. Generic security concept for IP backbones

The analysis of available IT security mechanisms in the previous section showed that each of them has its pros and cons, but none of them fully meets all requirements defined in Section III-A. Therefore, a generic concept for securing IP backbones is presented. In the proposed protocol architecture, the security layer is located between the network layer (UDP and/or TCP) and the layer of the used BAN technology (cf. Fig. 1). The main advantage of this architecture is that the network messages of the used BAN technology remain untouched. Therefore, this concept is transparently applicable to BACnet/IP, KNXnet/IP, and LonWorks/IP.

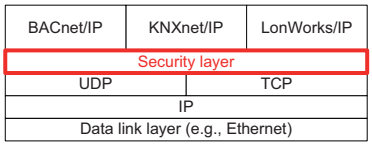


Fig. 1. Protocol architecture

To be able to securely communicate with other IP devices, each device has to establish a secure communication relation to them. This communication setup is divided into three phases: *Initial configuration*, *key set distribution*, and *secure communication*.

A. Initial configuration

Fig. 2 shows the operation sequence of the initial configuration phase. First, each device creates an ECC public/private key pair. In the presented solution, a key size of 256 bits is chosen, which is comparable to the security of AES-128 [16]. Then, the public key is transferred via a physically secured channel to the management tool which acts as certification authority (CA) – the private key remains secret to the device. The management tool creates a certificate which consists of the signed public key of the device and its signed IP address. This certificate is transmitted together with the CA's public key back to the device. With the help of this certificate, the device can authenticate itself during the key set distribution process. Furthermore, using the CA's public key, the device is also able to verify certificates of other devices.

Since the configuration phase is used to set up the initial security tokens, establishing a secured channel using cryptographic techniques is not possible. Therefore, the configuration process has to be performed in a physically secured environment. One possibility is to directly connect the device to the

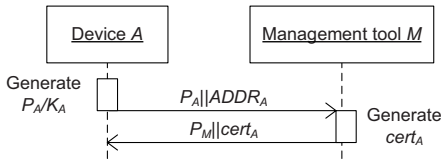


Fig. 2. Initial configuration phase[1]

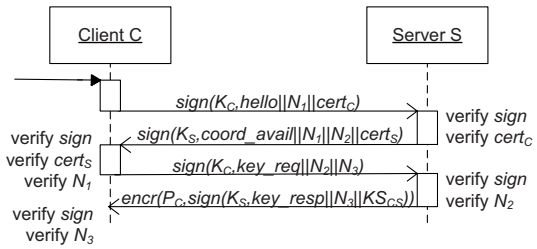


Fig. 3. Key set distribution for unicast services

management tool via a point-to-point connection (e.g., using a local EIA-232 interface).

B. Key set distribution

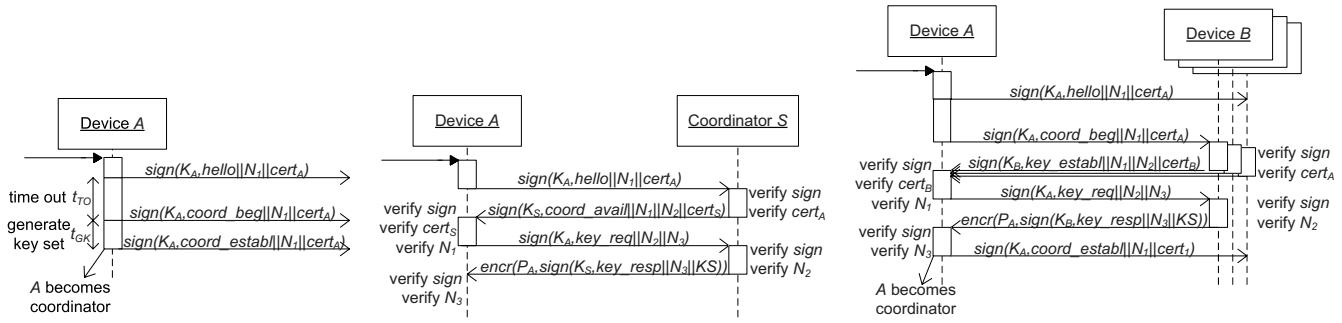
After a device has been configured accordingly, it is in possession of three security tokens: the public key of the CA, its own public/private key pair, and its certificate that proves the authenticity of its public key. Using these tokens, a device is able to start the second phase where the so called key set is distributed. A key set consists of the security tokens (e.g., shared secret keys) that are necessary to establish the secured channel in the secure communication phase.

To distribute a key set, some kind of *key set distribution protocol* is necessary. In the proposed protocol, an elected coordinator is responsible of maintaining the key set. In case of securing a unicast service (e.g., client/server connection), the key set distribution protocol works as follows: Suppose, for example, a client wants to set up a secured channel to a server (cf. Fig. 3). In case of unicast, the server acts as coordinator. To retrieve the key set from the server, the client sends a `hello`-message to the server. This message is signed with the private key of the client and contains a nonce N_1 as well as the certificate of the client. The server receives this message and verifies the signature of the message and the client certificate. If both are valid, the server immediately responds to the `hello`-message with a signed `coord_avail`-message, containing nonce N_1 of the `hello`-message and a new nonce N_2. N_1 relates the response of the server to the actual `hello`-message and avoids that someone replays the server's response to an older `hello`-message. To prove the identity of the server, the signature, the certificate, and the consistency of the nonce are verified. If they are valid, the identity of the server has been proved. 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 receives the message and proves the identity of the client by validating the signature and the consistency of N_2. If they are valid, the server responds with a `key_resp`-message, containing the number N_3 and the key set. The message itself is signed with the private key of the coordinator and encrypted with the public key of the client.

In case of multicast, a coordinator is not predefined. Rather, IP devices that are members of the multicast group must agree on a common coordinator. This works as follows: If an IP device wants to join a multicast group, it sends a `hello`-message to the multicast group. Again, this message is signed with the private key of the sender and contains a nonce N_1 and its certificate. Depending on the state of the multicast group, three different cases are distinguished:

1) Fig. 4(a): If there is no response from other IP devices within the timeout t_{TO}, the device assumes that it is the first active device within the group and takes over the coordinator role. To announce that it is the new coordinator and to prevent other devices from becoming the coordinator at the same time, it sends a `coord_beg`-message to the multicast group. After having sent this message, it generates the key set for this group. The generation process takes the time t_{GK}. After having finished the key set generation process, it sends a `coord_establ`-message. If two devices compete for the coordinator role at the same time (i.e., within t_{TO}), the one with the lower IP address shall become coordinator – after having proved the identity of the winning device, the other device shall cancel the coordinator election process. Furthermore, t_{TO} shall be greater than t_{GK}. This avoids conflicts during the key generation process (i.e., within t_{GK}), since the second device will receive a `coord_establ`-message before it initiates a `coord_beg`-message.
2) Fig. 4(b): If there is a coordinator available within the group, it responds to the `hello`-message with a signed `coord_avail`-message. The remaining steps are the same as in the unicast case (cf. Fig. 3).
3) Fig. 4(c): If there exists a coordinator that does not respond for any reason (e.g., the coordinator has crashed), the new device will try to assume the group coordinator role by sending a `coord_beg`-message (cf. case 1). The other group members will recognize this and respond with a signed `key_establ`-message, informing the new device that there is already a group key set available. This message contains the nonce N_1 of the `coord_beg`-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`-

[1]For the rest of this paper, the following notation is used: uppercases denote IP devices. $ADDR_X$ denotes the IP address, K_X the private key, P_X the public key, and $cert_X$ the certificate of device X. KS denotes a key set and SK a symmetric key which is shared among a group of devices. 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, N a number used only once (called nonce), and $||$ the concatenation operator.



(a) Device alone (b) Coordinator available (c) Coordinator unavailable

Fig. 4. Coordinator scenarios

message. After having received the key set, the new device takes over the coordinator role by sending a `coord_establ`-message.

Since it is possible that an IP device gets compromised, a revocation list is required that contains the certificates of all compromised IP devices. This revocation list is maintained by the CA i.e., the management tool and has to be distributed to all 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 CA. Since all IP devices are in possession of the public key of the CA, malicious `cert_revoke`-messages can be identified by verifying the signature.

C. Secure communication

After a key set has been distributed, 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 set. In contrast to asymmetric algorithms, a message has to be processed only once since devices of a communication relation (e.g., a single multicast group) use the same key set. A key set consists of three security tokens: a secret key SK_C for en-/decryption, a secret key SK_{MAC} for MAC calculation, and the currently valid sequence counter C. The used frame format is based on the application protocol frame format of TLS 1.2 [13].

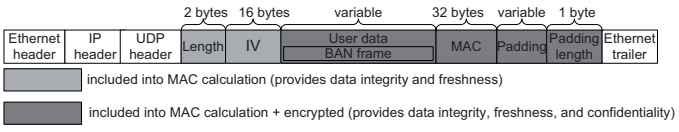


Fig. 5. Secured frame format

As shown in Fig. 5, a secured BAN frame consists of an unsecured and a secured part. The unsecured part includes the Ethernet, IP, and UDP/TCP header as well as the Ethernet trailer. The secured part consists of the $initialization\ vector(IV)$ for the en-/decryption operation, the $User\ data$ that contains the BAN frame, the MAC, the $Padding$, and the $Padding\ length$ 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 [11] in combination with SHA_256 [12] and a sequence number are used. The MAC itself is calculated as:

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

To guarantee data confidentiality, AES_128 in cipher-block chaining (CBC) mode is used. 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. To avoid an attack on CBC mode [17], a random number has to be chosen as IV.

There are two situations where the revocation and the distribution of a new key set are required. First, if a compromised device is added to the revocation list, it must be actively excluded from a communication relation. Second, due to security reasons, the current 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 devices that a new key set is available, a `new_key_avail`-message is sent by the coordinator. In case of a compromised device, every device of the multicast group has to request the new key set from the coordinator as shown in Figure 4(b). In the second case where a key set is renewed, 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.

V. Prototype implementation

To evaluate the presented security concept, a prototype implementation has been set up where the concept has been applied to KNXnet/IP (cf. Fig. 6). The network consists of

three small KNX TP 1 networks which are interconnected by an IP backbone. Three 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. A PC that is connected to a hub and running in promiscuous mode is used to sniff all packets on the IP network. In unsecured mode, it is possible to intercept and replay KNXnet/IP frames. Once the security layer is enabled, the KNXnet/IP frames are secured according to the presented scheme.

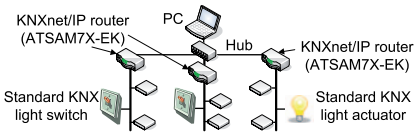


Fig. 6. Prototype network

In Fig. 7, 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 32-bit ARM7 Processor including 256 KB flash memory and 64 KB SRAM. It operates at up to 55 MHz. The CPU is connected to an Ethernet chip and to an EIA-232 interface which is used for the communication with the TP-UART.

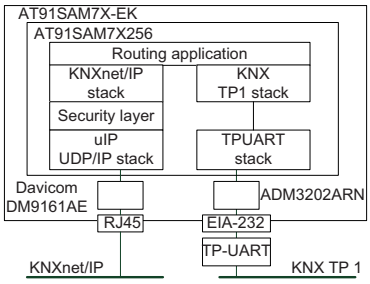


Fig. 7. Software overview

The open source library *uIP* serves as UDP/IP stack. For cryptographic calculations, a special library called *Multiprecision Integer and Rational Arithmetic C Library (MIRACL)* has been used. It provides cryptographic algorithms like AES, RSA, Diffie-Hellman, ECC over binary and prime fields, and various hashing functions. The KNXnet/IP stack and the routing application support basic KNXnet/IP routing functionality. To communicate with KNX TP 1 devices, a KNX TP 1 stack as well as a TP-UART stack are used. First measurements show that the available resources allow the use of ECC algorithms. The generation of a signature of a 140 bytes long message requires 136 ms, the verification takes 240 ms. The encryption needs 384 ms and the decryption 135 ms.

VI. Conclusion and Future Work

To protect IP backbones, a comprehensive security concept is necessary. However, available BAS standards that support the use of IP provide only rudimentary protection against security attacks. If at all, security mechanisms that are based on symmetric algorithms are used, which is due to resource constraints of the used embedded devices. The inherent drawback is the need for an online key server.

With the introduction of ECC, it has already become possible to use asymmetric algorithms even on embedded devices. In this paper, such an ECC based security concept for securing IP backbones is presented. As a next step, a formal evaluation of the proposed protocol is currently under way. Furthermore, simulation shall be used to evaluate the dynamic behavior of the protocol especially in large IP networks.

Acknowledgment

The work presented in this paper was funded by FWF (Austrian Science Foundation) under the project P19673.

References

[1] T. Novak and A. Treytl, "Functional Safety and System Security in Automation Systems – A Life Cycle Model," in *Proc. 6th IEEE Int. Conf. on Emerging Technologies and Factory Automation*, Sep. 2008, pp. 311–318.

[2] 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.

[3] A. Treytl, T. Sauter, and C. Schwaiger, "Security Measures for Industrial Fieldbus Systems - State of the Art and Solutions for IP-based Approaches," in *Proc. 5th IEEE Int. Workshop on Factory Communication Systems*, Sep. 2004, pp. 201–209.

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

[5] "BSR/ASHRAE Addendum g to ANSI/ASHRAE Standard 135-2004, BACnet- A Data Communication Protocol for Building Automation and Control Networks," Oct. 2008, third Public Review completed.

[6] A. Cilardo, L. Coppolino, N. Mazzocca, and L. Romano, "Elliptic curve cryptography engineering," *Proc. of the IEEE*, vol. 943, no. 2, pp. 395–406, 2006.

[7] "Open data communication in building automation, controls and building management – Home and building electronic systems – Part 2: KNXnet/IP Communication," EN 13321-2, 2005.

[8] "Tunneling component network protocols over Internet protocol channels," ANSI/EIA/CEA 852, 2002.

[9] W. Granzer, C. Reinisch, and W. Kastner, "Key Set Management in Networked Building Automation Systems using Multiple Key Servers," in *Proc. 7th IEEE Int. Workshop on Factory Communication Systems*, May 2008, pp. 205–214.

[10] S. Kent and K. Seo, "Security Architecture for the Internet Protocol," RFC 4301, Dec. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4301.txt

[11] "The Keyed-Hash Message Authentication Code (HMAC)," FIPS PUB 198, National Institute of Standards and Technology, 2002.

[12] "Secure hash standard," FIPS PUB 180-2, National Institute of Standards and Technology, 2002.

[13] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246, Aug. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5246.txt

[14] 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 Int. Conf. on Pervasive Computing and Communications*, Mar. 2005, pp. 247–256.

[15] B. Gleeson, A. Lin, J. Heinanen, T. Finland, G. Armitage, and A. Malis, "A Framework for IP Based Virtual Private Networks," RFC 2746, Feb. 2000. [Online]. Available: http://www.faqs.org/rfcs/rfc2764.html

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

[17] B. Moeller. (2008, Nov.) 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.