Smart Card Based Security for Fieldbus Systems

Christian Schwaiger
Austria Card
Vienna, Austria
Email: christian.schwaiger@austriacard.at

Albert Treytl
Institute of Computer Technology
Technical University of Vienna
Vienna, Austria
Email: treytl@ict.tuwien.ac.at

Abstract— **The provision of security services in fieldbus systems will gain in importance in the coming years. A need for such services is easiest seen in fieldbus systems for building automation as they give ample opportunities for mischievous adversaries to tamper with. Without loss of generality we discuss different ways to integrate security services in fieldbus systems after pointing out possible security deficiencies in three widely used fieldbus systems for building automation. Exemplarily, we show how to integrate security services to the LonWorks system. To do so we introduce a secure smart card and discuss the security services that were implemented, based on some simple but realistic assumptions about a possible security policy.**

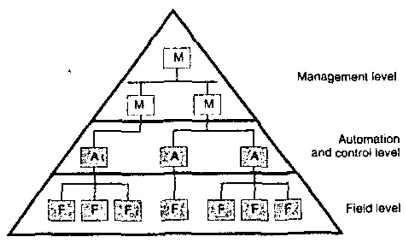

Fig. 1. CIM-pyramid model

I. Introduction

When fieldbus systems were first conceived in industrial automation in the 1970ies the idea of full vertical integration of plant communication which led to the CIM-pyramid model as shown in figure 1 was developed. While this model was (and is) sound the technology (as well as the theory) to realize the underlying ideas were immature and forbiddingly expensive. Only in the late 1990ies with the ubiquitiousness of IP-based networks in LANs and the Internet this idea became feasible and began to attract more attention. As the Internet is a public network and remote installation, monitoring and control of a fieldbus network is a most attractive application, and taking into account the quite miserable reputation of the Internet and its underlying IP protocol stack in regards of security, the last years brought security topics to the minds of researchers. But still, the awareness of most researchers as well as that of industry needs to be strengthened. Another reason for the neglect of security issues in fieldbus systems in the past stems from the fact that they were often used within closed environments (e. g. in industry automation) and security threats from the inside were not considered, though it is known today that many violations of security policies come from the inside. The best we can say is that security was only an add-on in fieldbus design. Today this point of view slowly changes partly because of the above reasons and partly because fieldbus systems were also introduced to areas like building automation where many parties may have (and need to have) access to such a system.

The three big security goals that can be formulated are confidentiality, meaning that only authorized entities must be able to read confidential data, integrity, stating that no unauthorized entity must be able to change data without being detected and availability, mandating that data is on-hand when it is needed, or CIA for short. While availability is a well-known problem - even when no malicious attacks have to be considered - and while integrity is one of the big goals of fieldbus protocol design - alas only in a context where no active attackers need to be taken care of - confidentiality is a feature as of yet never implemented (at least in the open) in common fieldbus systems, with the notable exception of BACnet (see section II-C). In research we find various (but still not too many) works aimed to secure access to a fieldbus over the Internet via a gateway approach as e. g. in [1] while security on the fieldbus itself has rather been of no interest until now with only some exceptions such as [2].

Regarding the fact that - according to some possible security policies - security services such as encryption to protect confidentiality or message authentication codes (MACs) or other means to protect integrity as well as authentication services to identify an entity and associate it with access rights, have a need to be included on the fieldbus level itself, and considering today's hardware basis which is badly suited to carry out cryptographic operations we propose the utilization of smart cards which provide the needed functionality that can be integrated in fieldbus nodes. At the same time smart cards increase the level of realizable security because of their design that is aimed at making them tamper proof. While the following discusses mainly fieldbus systems for building automation, the application of smart cards or other security tokens is not constricted to such systems alone but could also be adopted to any other fieldbus system.

The rest of the paper is organized as follows: Section II discusses security means already implemented in fieldbusses for building automation, section III shows how to implement security measures in the LonWorks fieldbus by using smart cards, section IV discusses why specific security measures

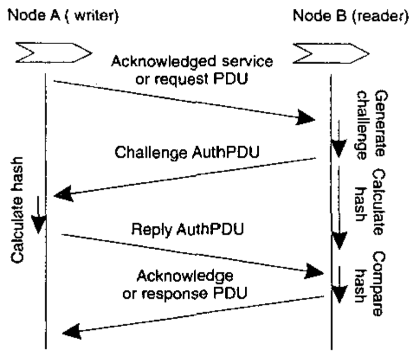

Fig. 2. LonWorks authentication protocol

where taken as well as some alternatives and finally section V summarizes the work, points out possible application areas of interest and talks about possible future work items.

II. Today's Security Means in Fieldbus Systems for Building Automation

This section will give a description of the security means currently available for three common fieldbusses for home and building automation.

A. *LonWorks*

Echelons fieldbus system LonWorks [3] offers only authentication services, which are implemented at the level of the transport- and session layers. There are no encryption services for data confidentiality and access control is only handled on a per node basis.

The authentication protocol is based on the following four step challenge-response mechanism to verify the identity of a writing node (cf. figure 2):

1) After receiving a PDU (protocol data unit) with a set authentication bit the receiving node (challenger, Node B) starts with generating a 64 bit random number, which is sent back to the challengee (Node A) using a special authentication PDU.
2) The challenger then immediately starts computing a one way transformation on the created random number and the data of the received message using its private authentication key.
3) The challengee uses the same transformation with its authentication key to transform the received random number and the contents of the original message and returns the resulting 64 bit hash value to the challenger.
4) Finally, the challenger compares the results of both transformations. The result of this comparison is passed to the layer which invoked the authentication service.

The authentication mechanism is handled by the authentication server, which is implemented within the transaction control sublayer of layer 4 in every LonWorks node. It offers the three basic services Initiate_Challenge(), for sending the random number, Reply() for transmitting the transformation and Process_Reply() to pass the results (fail, pass) to the invoking layers - either the transport or the session layer. In all cases the PDU is passed to the application layer for processing, along with notification as to whether the authentication failed or succeeded. It is up to the application layer to honor the result. For communication with network variables, which is the most common case for interoperable communications in LonWorks, both nodes must activate authentication in the network variable description and the node which should be authenticated must initiate the authentication process actively by setting the authentication bit to allow correct updating or polling with authentication [4]. This is a serious security flaw, because neither read or write of network variables can be protected unilateral.

The actual challenge and response messages are transmitted by special authentication protocol data units (AuthPDUs) and are only available for acknowledged unicast and multicast services. Especially, if more than one node is addressed the implemented challenge-response algorithm will cause a huge overhead, because every receiver must answers with its own random number.

The cryptographic algorithm used for the one-way transformation is not publicly available from Echelon and normally built into the neuron chip firmware. The key data of the algorithm are:

- 48 bit authentication key, which is stored in a read-protected section of the domain table and set by network management
- 64 bit random number
- 64 bit hash value from random number and message data

Because the design details of the cryptographic algorithm and the random number generator are not publicly available, their strengths and weaknesses cannot be analyzed and investigated from a cryptographic view point. Therefore, the algorithms have to be thought of as being weak. Additionally, today a key length of 48 bit is not assumed to be strong enough .

Two further weaknesses are key distribution and authentication within groups of nodes. Initial key distribution is a general problem, because it requires a secure channel or out-of-band communication. The recommended practice is to connect only one node directly to the management tool on a secure network during initial key transmission. Consecutive key updates can then be done by transmitting an increment to the encryption key via the network management message *update key* over the network.

Another weakness is that only one authentication key[1] can be stored in a node. If a group of nodes uses authenticated services, all must share the same key. Hence the flaw in having only one key available is that a node can only join one group and that the identity of a particular node cannot be proved anymore.

Access control for communication objects like network variables and explicit messages or network management com-

[1]If a neuron chip is member of two domains, each domain can have a separate authentication key, but these keys cannot be used mutually

mands can be controlled by the authentication mechanism with the above mentioned limitations. To protect the application code from manipulation the write/read protection can be enabled by a preprocessor command, but still vital areas (see III-B) like the SNVT[2] structures and the application data area may be read. Write access is limited to the configuration structure.
Comprising the LonWorks authentication service is more designed to prevent simple record and playback of commands rather than to implement a (strong) authentication of sending nodes. Also the access service provides no reliable protection of confidential data within a node.

B. *European Installation Bus*

The European Installation Bus (EIB) [5] implements security concepts only on a rudimentary scale. The application layer offers password based services for access control to the node's memory and object properties. There are no further authentication or encryption services available.
For access control in a node there are 255 different levels[3], which can be assigned to EIB objects or memory pages. For each of these levels a unique 4 byte password - in EIB it is also referred to as key and handled as a 32 bit unsigned integer - can be assigned. The value 0xFFFFFFFF is reserved for the empty, not valid password.
If a node wants to access a protected area in a remote node it must use the A_Authorize service primitive (see figure 3) to negotiate the access rights. The procedure includes the following steps:

1) A A_Authorize_Request-PDU is sent by Node A to transmit the key for the desired access level. The access level itself cannot be declared directly.
2) The receiving node (Node B) will search its access table for the key and retrieve the highest access level or 0xFF if the key is incorrect. The retrieved access level is then stored as current access level at the receiving node. The result is also transmitted to Node A.
3) After this negotiation the access level is set for the whole duration of the connection or until a new negotiation request. Access to read/write operations will only be granted if the current access level is less than the access level of the requested object.

The different access levels can be assigned to object properties and memory pages and consists of a 4 bit read access level and a second 4 bit write access level. Both levels can be set either using a range between 0x0 for highest access rights and 0xF for free access or to the following four levels:

- AccRFree - free access
- AccRApp - access to application parameters
- AccRLoad - access to address and association table
- AccRSys - access to application programmes

[2]Standard network variable types are defined by LonMark to define format and meaning of network variables (www.lonmark.org). These SNVTs are organized in the LonMark SNVT masterlist.
[3]0x00 is the highest whereas 0xFF is the lowest security level.

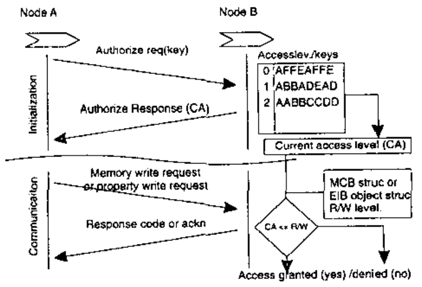

Fig. 3. EIB access control

Modifying and changing of keys is done using the A_SetKey-service primitive. If such a A_Key_Write-PDU is received the key is only updated when the current access level is higher than the access level specified in the PDU. If the modification was authorized correctly and executed successfully the access level is returned in an A_Key_Response-PDU, otherwise 0xFF is returned.
The big security flaw in the EIB access control is that all PDUs and keys are transmitted in plain text over the medium and can be simple recorded using a protocol analyser. Moreover, there are no means to protect against replay attacks and the access control applies only to the protection of the management operations. This means that normal operations of the application layer on the other hand are not protected ([5], [6]).

C. *BACnet*

To allow the use of secure network communication BACnet offers services to provide peer entity, data origin and operator authentication, as well as data confidentiality and integrity [7]. The fundamental elements of the used security architecture are a central trusted key server and the symmetric Data Encryption Standard (DES) algorithm.
In the BACnet security architecture the three parties *keyserver*, *client* and *server* can be identified as shown in figure 4. The basic services of the BACnet security architecture are the RequestKey service, the authentication service and message encryption.
The most basic service is RequestKey to obtain a session key. This process is initiated by a device (device A) by sending a RequestKey request to the keyserver. The payload of the sent APDU includes the device address as well as the device identifier of Device A and Device B. To protect this request the APDU is encrypted with the private[4] key of device A, which is only known to device A and the key server. By successfully decrypting the payload the key server verifies device A and returns a 56 bit session key to device A. The session key and the address of device A are also transmitted to device B. This is done for each node using the AddListElement-request, which is secured by encrypting the message with the respective nodes private key. After the session key is

[4]In BACnet secret keys that identify a device are called private keys.

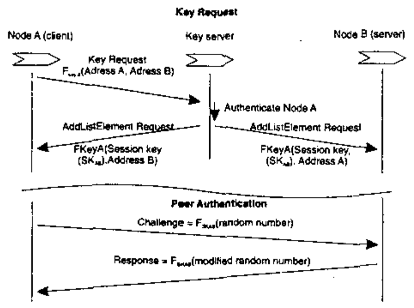
Key Request
Node A (client)
Key server
Node B (server)
Key Request
F_{KeyA}(Adress A, Adress B)
Authenticate Node A
AddListElement Request
AddListElement Request
FKeyA(Session key (SK_{AB}).Address B)
FKeyA(Session key, (SK_{AB}), Address A)
Peer Authentication
Challenge = F_{SKAB}(random number)
Response = F_{SKAB}(modified random number)

Fig. 4. BACnet device authentication

obtained message encryption or peer authentication can be started with the *authenticate* primitive. Message encryption is done by encrypting the payload using the DES algorithm. The fixed part of the APDU and the lower parts of the protocol control information are not encrypted. The one way peer authentication mechanism (figure 4) on the other hand involves a three step procedure similar to the LonWorks authentication mechanism:

1) Device A generates a random challenge number and transmits it to device B using the authenticate primitive encrypted by the session key.
2) Device B decrypts the message and modifies the random number by bitwise inversion of the most and least significant two bits and returns the number encrypted.
3) Device A test the modified number to authenticate Device B.

Further security services of BACnet are Message Execution Authentication, Message Initiation Authentication, Data Confidentiality and Operator Authentication: Message Execution and Initiation Authentication extends the authentication mechanism by including the ID of the message into the authenticate primitive. Proceeding this way the reception (execution) respectively the sending (initiation) of a message by a certain device can be proven. For operator authentication again the authenticate primitive is used and extended by the optional fields user name and password. The transmitted user name and password can either be verified locally or forwarded to the key server for verification. If the user name and password match, the modified random number is used to transmit a replay-resistant positive result. On password mismatch an error message is sent. To transmit data in a confidential manner the authentication mechanism together with the Message Initiation Authentication can be used to enable or disable encryption of the data payload. Using the optional Start_Enciphered_Session flag of the authenticate primitive will start encipherment and the modified random number is used as unique reply. In order to avoid spoofing of the message Message Initiation Authentication shall be used to verify the senders identity.

The initial generation and distribution of the private keys is a crucial parameter for the overall security of the BACnet security architecture, because the whole system relies on the non disclosure of the private keys. Especially the keyserver, which holds a copy of all private keys and passwords used in the network, must be protected. The BACnet standard [7] does not specify this area and leaves the actual implementation open. Also the use of the session key is not regulated, usually two keys should be used for authentication purposes and encrypting data.

Concerning future developments in the security part of BACnet there are discussions to replace the DES algorithm by other means. Additionally, in connection with BACnet/IP - a protocol to transmit BACnet PDUs over IP networks - Kerberos [8] is a candidate to replace the BACnet security server.

TABLE I
SECURITY MEANS OF ANALYSED FIELDBUSSES

Service	LonWorks	EIB	BACnet
Encryption	n/a	n/a	DES
Integrity	challenge-resp. mechanism	n/a	DES
Authenitication	challenge-resp. mechanism	plaintext password based access control	challenge-resp. mechanism using DES

D. Summary

With the exception of BACnet the security architecture is implemented very rudimentary and *not suitable* for any application at all (cf. table I). Especially, the unencrypted transmission over the publicly accessible network as well as the limited access control and authentication should be reconsidered critically. Another big problem is the production of cryptographically secure random numbers that are needed for different authentication services or for the generation of keys in BACnet. While nothing is known about the generation of such numbers in LonWorks and EIB the BACnet standard leaves the realization of any security measures totally up to the implementor. While generation of random number needs either a source of true randomness which is definitely not available in common nodes other secure mechanisms often need more computing power than found in the hardware of typical nodes.

III. INTEGRATION OF SMART CARDS IN LONWORKS FIELDBUS NODES

A. General Concerns of Smart Card Integration

The examined fieldbus systems were all designed with respect to optimize network performance and for LonWorks and EIB to handle small data packages efficiently. Due to this reasons in the trade off between performance and security the security part was not emphasized. To enhance the security of the fieldbus system additional security features can be integrated at two levels:

1) Integration within the fieldbus protocol layers.

2) Integration on top of the fieldbus protocol layers.

The first possibility of integration within the different layers of the fieldbus protocols offers the possibility to tackle potential risks at the layer they occur on. The great disadvantage of this approach on the other hand is the fact, that most fieldbus protocols are not designed for security rather they are designed as light weight efficient communication protocols. Incorporating security will call for partially massive changes in the communication protocol. Especially on hardware based implementations like LonWorks or EIB these changes can hardly be done even for academic research.

The second approach would be to integrate an additional security layer outside the fieldbus layers. This can be either done on top of the application layer or by introducing (sub-)layers, when the existing protocols do not define these layers. E. g. for the original CAN-system only layer one and two are defined and therefore security could be integrated on layer 4 to achieve an end-to-end security.

In our approach we chose the second way to integrate smart card based security services for an end-to-end communication, because the LonWorks protocol is fully implemented in hard- and firmware within the neuron chip and there are no possibilities to change the protocol stack within the neuron chip. Another argument for our decision, although more economical, is that such a system still fully complys with the standards although interoperability is confined by defining special user defined structures.

B. Integration of a Smart Card and a LonWorks Neuron Chip

Our intention for introducing smart card based security to LonWorks were driven by the following aspects:

- Increase the efficiency compared to the native authentication procedure.
- Increase security by encryption to the data transmission.
- Replace unknown algorithms by publicly known secure algorithms.
- Allow a node to be within more than one security domain.
- Support also unacknowledged services.

Our approach was in the first step to set up a flexible hardware platform for teaching and research work, that is comparable to LonWorks modules used in building automation today and to implement basic security features on top of this system in the second step.

Fieldbus nodes in general and LonWorks nodes in particular are not well protected to be qualified for storing data in a secure way. With the appropriate management tool and physical access to the node and/or the network every property of a node from simple configuration properties to the whole application can be read and often be changed too (cf. section II-A). In opposite to industrial automation where physical access is often limited, in building automation the network and the nodes are "publicly" accessible. Therefore, some kind of security token must be introduced, which on the one hand can securely store secrets (e. g. keys) and on the other hand execute security procedures (e. g. cryptographic algorithms) to prevent security breaches due to application manipulation.

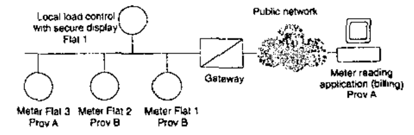

Fig. 5. Demonstration application setup

The first step in joining these two worlds was the implementation of a smart card interface for a neuron chip, which is the standard LonWorks node. Although all interfaces to a smart card are well defined [9] it is not possible to integrate any of the ISO 7816 protocols by software in the neuron chip due to limitations in memory and computing power. Hence an appropriate smart card reader has to be selected, which can be interfaced without exceeding the computing power, memory and IO facilities of the neuron chip. The results of this research were disheartening, because for a great number of readers the interface description was not available or the implementation of the interface was too complex, because many manufactures tend to reduce hardware and do as much protocol handling as possible in software, which is no problem for PC-based systems. Finally, the Cream 135C smart card reader with a 9600 bit/s RS232 interface from SC ITEC[5] was chosen, because it offers an extremely light-weight protocol that is suitable for simple, single threaded micro-controllers. It uses a full custom ASIC to implement the ISO 7816 protocols.

On the LonWorks side we decided to use standard neuron chips, to implement our concepts with technology that is already installed in the field. Commercially available are two different models the neuron chip 3120, with up to 2kbyte of RAM and EEPROM and the 3150 neuron chip with up to 64 kbyte of RAM or EEPROM. After investigating several modules from different manufacturers in the end the decision was in favour of the LTM-10A Module from Echelon, which is equipped with the standard neuron chip 3150, 32 kbyte Flash and 32 kbyte RAM and a FTT-10 transceiver. This decision was primarily made with regard to a more versatile evaluation platform and easier programming. On the smaller 3120-modules, the 2k EEPROM memory inhibits more complex programmes.

Our setup demonstrates secure transmission of confidential data within a LonWorks network. As a possible scenario we chose remote meter reading in an apartment building (figure 5) including the meters, some local display unit and a remote meter reading. The billing system is at the moment - to allow for an easier setup - connected directly to the fieldbus. This causes no restrictions because a gateway normally has to transmit the meter values transparently to the billing system. Figure 6 shows the secure meter node including a S0-power meter input, a digital output for a main power switch to turn off the power supply remotely, the smart card reader and a RS232 connection to a PC for debugging purposes.

[5]SC ITEC GmbH, Paderborn, Germany, www.sc-itec.de

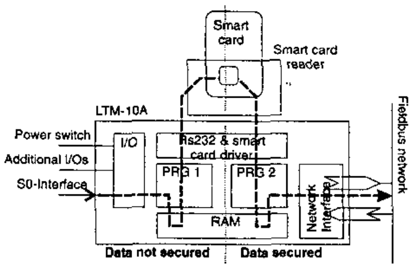

Fig. 6. Metering setup

Securing of transmitted data includes the following steps:

1) Starting at the process IOs the raw input data of the S0 interface are first transformed to an actual power value, which is defined in LonWorks by standard LonWorks C-structure SNVT_elec_kwh.
2) The data is transmitted to the smart card.
3) The data is secured using the smart card's en- and decryption and MAC calculation capabilities.
4) The secured data are passed to the network protocol stack of LonWorks and transmitted to the recipients node.

The reverse scheme is used at the recipient or if output data should be changed. Section IV gives details and rationale about the used security primitives.

In LonWorks data can be transmitted in two different ways

- either by exlicit messages, which allow to transmit binary data up to 32 (228) bytes[6]
- or by standard network variables, that also describe the content of the transmitted data.

LonWorks does not support any packet segmentation and reassembly mechanism (SAR). If such algorithms are required they must be implemented at the application layer, but this will on the one hand degrade the performance of transmission and on the other hand cost valuable programme space in the already limited neuron chips. Interoperable communication in LonWorks is based on the use of network variables, because together with a SNVT definition the format and meaning of data are settled. Hence our implementation also uses network variables to transmit the data. Because in the SNVT master list no SNVTs are defined to transmit encrypted data, the SNVT_str_asc, which is normally used to transmit short text strings and status messages of up to 31 ASCII-encoded characters, was selected as a transparent container to transmit the encrypted data. This is no limitation for the processed 2 byte energy value and should be fine with most other data to be transmitted. In case of longer messages an additional mechanisms that chains two or more secured message parts together would be needed.

Using the available security mechanisms of the integrated smart card the following security goals are reached:

[6]32 bytes is the interoperable size, which can be received by all nodes. The maximum limit of 228 bytes requires nodes with extended network buffers.

- Transmission efficiency is increased, because only one message is needed to transmit data confidentially and authenticated[7].
- Along with acknowledged services, unacknowledged services and any kind of broadcast and multicast services can be used.
- Data can be transmitted encrypted.
- The unknown algorithm of LonWorks is replaced by a secure algorithm.
- Data can be authenticated and encrypted on the basis of network variables. Hence a node can authenticate and encrypt data for independent connections[8].

One still not fully tackled matter is the fact that the node memory is still not protected and data could be read before they are encrypted (figure 6 left side). In our first approach we concentrated on transport security only (cf. section IV). Possible solutions for preventing the above attack will most likely need additional hardware and will not allow processing of data within a LonWorks node [10].

IV. Security Measures & Implementation

A. Assumed Security Policy

Because the discussion of security issues is pointless without an underlying security policy that identifies valuable assets as well as how they need to be protected we state our basic assumptions that influenced our choice of security mechanisms.

Considering a typical LonWorks installation in building automation we assume that one organizational entity is responsible for installing, configuring, maintaining and controlling the network. The assets are mainly the nodes that need to be secured because they contain or process sensitive information, such as e. g. proprietary programmes in the first or alarm data in the second case. This distinction has to be drawn in the beginning of the development cycle. Finally, we assume that the transmission lines, e. g. cable or wireless transmission, are subject to eavesdropping as well as to modification and insertion of data by a malicious adversary, while the nodes themselves are not accessible to an adversary. To simplify the assumptions we determined that a secured node offers and enforces integrity as well as encryption services, though one could argue that in some circumstances integrity alone would suffice. Authentication is needed for transmitted data, only. There is no need for a provision of non-repudiation services.

B. Symmetric vs. Asymmetric Security Algorithms

Today we distinguish between cryptographic primitives that are based either on a shared key between communication parties and such where each party that participates in a communication process has a key pair consisting of a private key that must be kept secret and a public key that may be distributed freely. The former variants are called symmetric

[7]For building automation control systems the reduction of network load is the main advantage - delays introduced by the smart card communication can be neglected in normal operation and will be a topic for further research.

[8]The number is only limited by the memory available on the smart card.

because the same key is used on both sides and the latter are called asymmetric because the public key is used to encrypt data and the private key is used to decrypt data [9]. Their major difference is that in asymmetric cryptography an infeasible amount of computational effort is necessary to derive the private key off the public key while in symmetric cryptography this is not the case. It is easy to see that asymmetric primitives offer an advantage if two communicating parties do not know each other in advance because by transmitting a public key an adversary gains no knowledge of the private key [10]. In contrast using symmetric cryptography we would need to securely set up approx. n^2 shared keys between n potential communication partners which works for small groups but is not very flexible for large groups and infeasible for networks like the Internet where parties do not even know each other in advance. On the other hand symmetric cryptography is far superior if it comes to computational speed which is the reason why protocols such as TLS/SSL [8] , which is widely used to secure Internet communications, use a hybrid approach whereby asymmetric algorithms are used to exchange a symmetric session key that is used to secure bulk data. Both, symmetric and asymmetric primitives provide building blocks to implement integrity, authentication and encryption mechanisms that can be integrated in security protocols.

C. *Used Security Mechanisms*

In our scenario as identified above we have one centrally managed installation where all nodes are known at installation time. While it is possible to build groups of nodes such as e. g. fire alarms there is no pressing necessity to do so. Anyhow, it is easy to see that symmetric mechanisms are sufficient whereby a symmetric key is shared between all nodes that need to be secured. In case where nodes have been grouped in what one could call *security domains* each such domain would share a distinct key between its member nodes. If nodes are replaced or added the responsibility to install the needed keys (either for the whole network or for the group the node will belong to) is with the organizational entity. This mandates that provisions to securely store secret keys have to be foreseen. At the moment in our demonstrator this is done by physical exchanging smart cards. While this is a reasonable way for initilization, key update should be done over the network using additional administration keys. While this is true for encryption it also holds for integrity mechanisms because - as stated above - there is no need to include any non-repudiation services that would make the inclusion of digital signatures, which are based on asymmetric cryptography, necessary. In the following sections when we talk about keys needed to provide integrity and encryption services it has to be made clear that those keys must be distinct, i. e. we *need* one dedicated key for each service which must not be used to provide any other service.

1) Encryption: For symmetric encryption a variety of block ciphers that are believed to be secure are known, most notably the *Digital Encryption Standard* (DES) [11] and the *Advanced Encryption Standard* (AES) [12] which is based on and a subset of the Rijndael cipher which was designated as the successor of the DES in 2000. Both algorithms are iterated ciphers with the AES being a Substitution-Permutation Network (SPN), and the DES being a Feistel network which is a special case of the former. While no design weaknesses could be found in DES the replacement with the AES was necessary because of the small key size of DES which is only 56 bits. Strengthening DES by enlarging the key space as done in Triple-DES (also called 3-DES) where one DES operation is carried out three times for each block with different keys substantially slows down en-/decryption. Additionally, the block size of 64 bits is too small by today's standards although attacks on ciphers with small block size are not very practical. On the contrary, AES has a block size of 128 bits and a variable key size of 128, 192 or 256 bits, which will be sufficient for some time to come, according to [13], under the assumption that no short-cut attack against AES will be found. An additional feature of AES is its versatility - it is comparatively fast on a wide range of platforms ranging from 8 bit micro-processors to high-end computation platforms.

For the integration in LonWorks nodes the major goal was to choose a widely available encryption algorithm that has no (known) security weaknesses. For presentation purposes simple DES could have been chosen as it gives better performance than 3-DES and is widely available in most smart-card platforms. Anyhow, for real production systems the limited key length must be taken into account, therefore 3-DES was chosen. Another requirement for a production system is the provision of a schedule for regular key changes. They are themselves dependent on the amount of data en-/decrypted with a single key, which is again dependent on the number of nodes that use the same key i. e. if the nodes have been sub-divided into different security domains with distinct keys, or not. Regarding the operation of en-/decryption which takes 64 bits at one time it was decided to use CBC mode [14] (cipher block chaining) which xors the output of one block with the next input block before encryption of the next input block. To fill up blocks that are not a multiple of 64 bits mandatory ISO-80[11] padding was chosen. which means that even if the last block to process has 64 bits padding has to be done, which prevents ambiguities at the receiving end. An alternative, to avoid padding, would have been to use counter mode which on the down-side would mandate an additional secret and shared bit string, the counter.

2) Integrity: While in the discussion of security the importance of encryption is usually focused on, from our point of view integrity is of bigger importance in any control network. Despite a common misconception that was upheld until the 1970ies and that is still rampant today encryption is

[9]For digital signatures the private key is used to sign and the public key is used to verify a signature.

[10]The problem of getting an authentic public key is usually solved using certificates which will not be discussed further here.

[11]With this padding mechanism the first padding byte consists of 0x80 followed by 0x00 until the end of the block.

not adequate to provide for the integrity of data [16]. This also constitutes our main critique about the security services available in BACnet, where we assume that usually only one key will be used between two communication partners. While data integrity is built into all modern fieldbus protocols - which strengthens our assessment about its importance above - the provided methods like e. g. CRC codes are not designed to help against (real-time) manipulation by an active adversary. Therefore, we chose to include a message authentication code (MAC) in our design that provides data authentication, resistant against unauthorized manipulation.

Today we basically find two wide-spread methods to generate a MAC using symmetric keys:

- Encryption functions used in cipher block chaining mode[12]
- Keyed cryptographic hash functions

Using an encryption primitive such as those discussed above, usually the last block of the CBC encryption is taken and output after applying an appropriate output transformation, such as truncating the last block and outputting only the n leftmost bits or applying a final encryption operation with a different key. A secure hash function, such as SHA-1 or RIPEMD-160, which takes an arbitrary input and transforms it to a fixed-size output has the property that it is *preimage resistant*, *2nd preimage resistant* and *collision free*. Together with a secret key various constructions exist to obtain a MAC, most notably the wide-spread HMAC construction. See [15], for an in-depth treatment of secure hash functions, MACs and their construction.

In choosing a MAC - as well as in choosing an encryption function - the highest priority is security. Considering the availability of different algorithms it is noticeable that today's hash functions tend to be faster than DES (and much faster than 3-DES) while giving comparable security. Therefore, and because of educational reasons it was decided to implement the HMAC algorithm using SHA-1 as the hash function which is also denoted as HMAC-SHA-1.

D. Smart Cards

The implementation of either of the previously mentioned security algorithms in fieldbus nodes is no trivial tasks considering that - to drive down node costs - the used hardware is stripped down as much as possible in regards of processing power and memory. Together with the insight that maybe not all nodes need to enforce the security policy decided upon for the fieldbus, the implementation of security services using a security token seems to be reasonable. While security token is a neutral notion the only (widely) available token that is small enough, comparatively cheap, well standardized and also very tamper-proof is a smart card. Additionally, smart cards usually consume very little power, though in some modern systems active counter-measures against passive intruders might lead to higher power consumption when cryptographic operations are performed. [17] contains extensive information about this technology.

[12]This is also the only reason why we mention encryption mechanisms prior to integrity mechanisms.

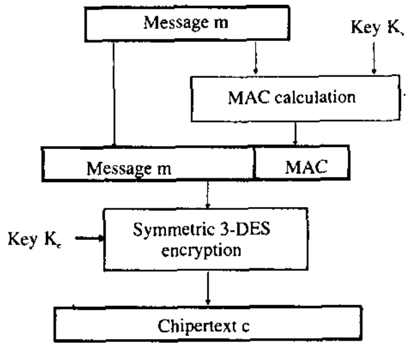

Fig. 7. MAC calculation and encryption of data

Since the beginnings of chip card technology efforts to internationally standardize this technology have left to the ISO 7816 series of standards [9] that regulate not only the physical and electrical characteristics of a smart card but also standardize two communication protocols T=0 (used in GSM systems) and T=1 (used in 3rd generation mobile communication systems such as UMTS and in banking standards) of which at least one but today even more often both are readily available on modern smart cards. Smart cards themselves are a subgroup of chip cards and consist of an own CPU (e. g. 8051 derivates), ROM (typically up to 128 kbyte) and EEPROM (typically up to 64 kbyte) for the OS and persistent data as well as some RAM (typically up to 1 kbyte)[13]. With growing sophistication smart cards also contain coprocessors that may speed up cryptographic operations. Furthermore, they usually contain circuitry designed to prevent tampering with the hardware such as e. g. light sensors, that protect secret data like keying material from intruders. This trait makes them especially useful if we consider the fact that nodes that need to protect data might be accessible to an intruder.

For the implementation of the above mentioned encryption and integrity services Giesecke&Devrient Java Cards were used[14] because of their Java Card 2.1 capabilities which allow for a rapid prototyping of applications in a subset of the popular Java programming language. Apart from providing a DES implementation as well as the SHA-1 algorithm needed to construct the HMAC the programming environment allows to simulate the written programmes in a GUI making the whole development process well suited for educational purposes. For the reasons described in section IV-C.2 the available asymmetric RSA algorithm was not needed. Figure 7 shows how a value is processed by the card before the authenticated and encrypted value is transmitted over the network.

[13]Other CPUs, increasingly 32 bit processors as well as more memory than stated may be available in the newest generation.

[14]The Sm@art Cafe professional toolkit 1.1 was kindly provided by Giesecke&Devrient.

V. Conclusion

With the sharing of communication resources, remote control and the amenability of networks in building automation increasing the need for integrity, confidentiality and authenticity of data becomes more important and will develop into a serious problem if appropriate provisions are not put in place in time. Using a LonWorks fieldbus system we showed exemplarily how to implement security systems using a smart card. Possible application areas for the described security services might be, but are not limited to energy metering, security and safety systems, home automation and remote control of critical infrastructures.

The presented method could be extended to other fieldbus systems or in the case of BACnet to implement the specified security services in a secure manner. While our solution is viable additional work must be invested in such topics as secure key distribution and key update as well as other administrative tasks; integration of such services in existing systems might require substantial effort. Moreover, physical protection of nodes might become a topic in surroundings where an adversary otherwise has the possibility to physically access a node. A related problem is that of fabrication or manipulation of data before it reaches a node.

Technical problems that could arise in the attempt to achieve fieldbus systems that can be configured according to particular security needs can be solved. Solutions may lead to a new fieldbus system that supports the conceived security services - which from our perspective is a worthwhile goal if efforts are well-funded and supported on a broad international basis. Organizational problems might pose an ever bigger problem. First, people need to become aware that the fundamental basis of any secure solution is a specific security policy that is tailored to the situation at hand. To aid in this process examples of "typical" security policies for different fieldbus configurations as identified in [2] could be of much help. Another important point to emphasize is that security risks are not only introduced from the outside which still today leads many people to conclude that a well-configured firewall is all that is needed. This is a flawed assumption that could lead to dire consequences.

References

[1] Christian Schwaiger, Thilo Sauter, *A secure architecture for fieldbus/Internet gateways*, Proceedings of 8th IEEE International Conference on Emerging Technologies and Factory Automation, pp. 279-285, 2001.

[2] Christian Schwaiger, Thilo Sauter, *Security strategies for field area networks*, Industrial Electronics Society, IEEE 2002 28th Annual Conference of the IEEE, pp: 2915-2920, 2002.

[3] Electronic Industry Alliance, *EIA/CEA-709.1 Control Network Protocol Specification*, Rev. B,2002.

[4] Echelon, *Neuron C Programmer's Guide*, Rev. 6, 2002.

[5] European Installation Bus Association, *EIB Handbook Series Volume 3: System Specifications*, Release 3.0, 1999.

[6] Sauter, Kastner, Dietrich, *EIB Installation Bus System* , Publicis, 2001.

[7] American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc., *ANSI/ASHRAE Standard 135-2001 BACnet - Data Communication Protocol for Building Automation and Control Networks*, 2001.

[8] William Stallings,*Cryptography and Network Security: Principles and Practice*, 3rd edition ,Prentice Hall,2003.

[9] International Standardization Organization, *ISO 7816, Identification cards Integrated circuit(s) cards with contacts Part 1-10*, 1996-2002.

[10] Peter Palensky, Thilo Sauter, *Security Considerations for FAN-Internet connections*, Proceedings of the IEEE International Workshop on Factory Communication Systems, Porto 2000, pp. 27-35, 2000.

[11] National Institute of Standards and Technology (NIST), *FIPS 46-3, Data Encryption Standard (DES), 1999.*

[12] National Institute of Standards and Technology (NIST), *FIPS 197, Advanced Encryption Standard (AES)*, 2001.

[13] Arjen K. Lenstra, Eric R. Verheul, *Selecting Cryptographic Key Sizes*, Journal of Cryptology, vol. 14, nr. 4, Springer-Verlag, pp. 255-293, 2001.

[14] National Institute of Standards and Technology (NIST), *SP 800-38A 2001 ED, Recommendation for Block Cipher Modes of Operation*, 2001.

[15] Douglas Stinson, *Cryptography: Theory and Practice*,2nd edition, Chapman & Hall/CRC, 2002.

[16] Gustavus J. Simmons, ed., *Contemporary Cryptology : The Science of Information Integrity*, IEEE Press, 1992.

[17] Wolfgang Rankl, Wolfgang Effing, *Smart Card Handbook*, 2nd edition, John Wiley & Sons, 2000.