Sensornetworks: Integration into IPv6 based networks and security on layer 3

Andreas Scheibleger
Betreuer: Corinna Schmitt
Seminar Sensorknoten: Betrieb, Netze und Anwendungen SS2010
Lehrstuhl Netzarchitekturen und Netzdienste, Lehrstuhl Betriebssysteme und Systemarchitektur
Fakultät für Informatik, Technische Universität München
Email: scheible@in.tum.de

ABSTRACT
Sensor networks are becomming more and more important. They are no more only used internally in companies but get gradually connected to the internet. As there are that much sensor nodes it is sensible to use the IPv6 protocol, since it provides a 128 bit address space. This technique however needs some adaptation on the link layer as well as on the network layer. Therefore there arise the same security risks which need to be solved as on typical stationary devices. Solving the integration and adding security on the network layer is going to be discussed in this paper.

Keywords
IPv6, 6LoWPAN, MiniSec, Security, IEEE 802.15.4

1. INTRODUCTION
Sensor nodes get more and more connected to already established conventional networks. For that to work there was the need to develop methods, which allow the usage of standard protocols on devices which have limited memory and processor resources. In favor of that the "IPv6 over Low power Wireless Personal Area Networks" (6LoWPAN [9]) architecture was developed. This architecture describes a mechanism to encapsulate the data and compress the headers in order to send and receive IPv6 packets over a IEEE802.15.4 [5] network. In the area of sensor networks the usual security criterias, namely "confidentiality", "integrity" and "accountability" need to be guaranteed as well as the criteria of "freshness". In addition there is the need for low energy consumption. For this purpose the "MiniSec" [8] protocol got developed. Those two topics are going to be covered in the following.

2. CLASSIFICATION
This document refers to security on layer 3 whereas layer 3 in the OSI-model is meant. To make it clear the model is shown in fig.1. By that one can see that it is not about how packets are sent or received, but the type or content of the packet is of interest. The network layer, loosely speaking, is responsible for accepting packets destined for the device and for forwarding packets if the current device is used as an intermediate node, i.e. routing.
The first part of this document deals with integration of sensor nodes into an IPv6 network. IPv6 (Internet Protocol version 6) is settled as well on this layer, as is the actually more widely used IPv4.

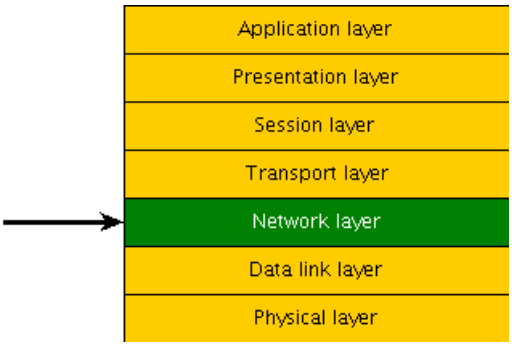

Figure 1: OSI-model

3. 6LOWPAN
3.1 Motivation
By extending LoWPAN or 802.15.4 [5] based networks so that it is possible to transmit encapsulated IPv6 [4] packets, one can make a fully into the IP network integrated device out of a simple sensor node. The problem is that the maximum packet size for 802.15.4 is limited on the physical layer (see fig.1) to 127 bytes. The maximum overhead for the frame and for the security on the link layer amount 25 and 21 bytes, respectively. With a default size of 40 bytes for the IPv6 header and 8 bytes for the UDP header, there only remain 33 bytes for the payload. Thus it becomes clear that a procedure must be introduced to shrink the large IPv6 header as well as the UDP header in order to transmit the packet over a LoWPAN network. In addition this modified packet needs to be encapsulated into the existing LoWPAN packet. For that an adaptation layer is inserted at first which is needed for mesh-routing, fragmentation and message identification. In connection it will be shown how in accordance to RFC4944 [9] the IPv6 header can be compressed in the extreme case from 40 down to 2 bytes and the UDP header from 8 down to 4 bytes. Further how stateless address autoconfiguration is done in order for every node being reachable via IPv6.

3.2 Adaptation header
Fig.2 shows a full adaptation header including the fields for mesh-routing as well as the fields for fragmentation. Those are optional and are only used if necessary to avoid unnecessary overhead.

Figure 2: full adaptation header

Mesh type: This type field with the corresponding header field should only be inserted if the connection is not a direct one, but needs mesh-routing. The mesh-type starts as shown in fig.3 with the bit sequence 10 followed by two bits (V, F) which indicate the adress type. Subsequently a 4 bit hops-left counter which is decreased at every hop is appended. If this counter reaches a value of 0 the packet should be dropped. V specifies the address type of the originator

Figure 3: Mesh type and header

address and should be set to 0 if it is a IEEE extended 64-bit address or otherwise to 1 in case of a short 16-bit address. F is used the same way for the destination address.

fragment type: If the whole payload fits into a single 802.15.4 frame, this type field and its corresponding header field should be left away. In case of needed fragmentation the header of the first fragment should be built according to fig.4, the header of the subsequent fragments according to fig.5. The datagram size is 11 bits long and should be the

Figure 4: First fragment

Figure 5: Subsequent fragments

same for all correlated fragments indicating the size of the whole packet. The tag is 16 bits wide and should be the same for all correlated fragments, either. It should be reset to zero if it reaches the maximum possible value. The 8 bit wide offset is present with all fragments but the first one. The offset for the first fragment is implicitly set to 0.

Dispatch type: This field is mandatory and starts as shown in fig.6 with the bit sequence 01 follwed by a 6 bit wide selector which indicates the header type of the subsequent header. Thereby the values for the selector are defined as listed in table 1.

Figure 6: Dispatch type and header

00 xxxxxx	NALP	The following data is not part of the LoWPAN encapsulation
01 000001	IPv6	An uncrompressed IPv6 header is next
01 000010	LOWPAN_HC1	A LOWPAN_HC1 compressed header is next
01 ...	reserved	
01 010000	LOWPAN_BC0	A LOWPAN_BC0 header for mesh broadcasting/multicasting is next
01 ...	reserved	
01 111111	ESC	Another 8 bit selector is following in order to have values greater than 63

Table 1: Selector values

3.3 Address autoconfiguration

The interface identifier for an IEEE 802.15.4 interface cannot directly be used as interface identifier for IPv6 as this one is 2 bytes longer. After optaining such a compatible interface identifier (see 3.3.1) we can generate a IPv6 link local address (see 3.3.2).

3.3.1 Interface identifier

IEEE 802.15.4 devices may have an IEEE EUI-64 address or a short 16 bit address. If it is a short 16 bit address then one has to derive a IEEE EUI-64 bit address out of this very short address. This is achieved by firstly generating a 32 bit prefix which is built out of the concatenation of the 16 bit PAN ID (Personal area network Identifier) and a sequence of 16 zero bits. This prefix is prepended to the 16 bit short address which gives a 48 bit pseudo interface identifier. This is then equally long as a default built in ethernet interface identifier. In accordance with RFC2464 [3] this 48 bit long address is then enlarged to an EUI-64 address as described in the following. Let the 48 bit address be the one shown in fig.7. To form the EUI-64 interface identifier

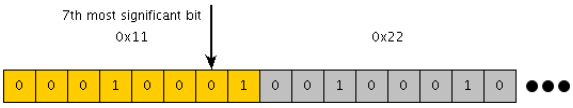

Figure 7: original interface identifier (11:22:33:44:55:66)

needed, the first three bytes are used as company identifier and are copied into the EUI-64 identifier. The next two bytes are statically set to FFFE hexadecimal. Afterwards the last three bytes are appended to the EUI-64 identifier. The last thing to be done is to set the Universal/Local bit to 1 as this derived EUI-64 identifier is in general not universally unique. This bit is the seventh most significant bit. Applying this we get the following:

13:22:33:FF:FE:44:55:66

Now that we either have a EUI-64 built in address or a derived one we can go further an build the interface identifier

out of it. This is simply achieved by complementing the Universal/Local bit. This needs to be done as global uniqueness in EUI-64 is indicated by a 0 and global uniqueness in the IPv6 interface identifier is signified by a 1. To go on with the example above we end up with the following IPv6 interface identifier:

11:22:33:FF:FE:44:55:66

3.3.2 IPv6 link local address

Now having obtained the interface identifier as described in 3.3.1 we can form the IPv6 link local address out of it to the prefix FE80::/64 which identifies an IPv6 link local address. It is built as shown in fig.8 by appending the 64 bit interface identifier to the already mentioned IPv6 link local prefix FE80::/64.

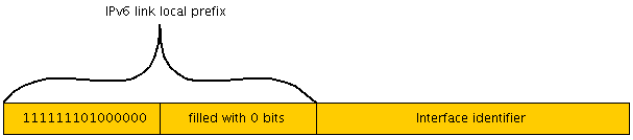

Figure 8: IPv6 link local address

3.4 Header compression

Given the assumption that the LoWPAN devices are in the same network they share some information that is not needed to be transmitted, so these informations can be left away in order to compress the IPv6 header. So the following fields can be left away or considered to be compressable:

- Version: is automatically set to IPv6
- both IPv6 source and destination addresses are considered to be link local and in addition can be derived from the link layer interface identifiers.
- packet length can be inferred either by the datagram_size in the fragmentation header or by the frame length on the link layer.
- traffic class and flow label are considered to be zero
- the next header is UDP, ICMP or TCP

So the only field that needs to be transmitted as a whole is the hop limit. This is of course the best case scenario. For example by changing the link layer interface identifier by software for security reasons the source and destination addresses cannot be derived that easily. Packets following the assumption to be compressible can be compressed via the LOWPAN_HC1 format by using a dispatch-type of LOWPAN_HC1 followed by a "HC1 encoding" field as shown in fig.9 and additionally the uncrompressible header fields. Thereby the encoding field is defined as described in tables 2,3,4 and 5, where PI indicates that the prefix is carried uncompressed, PC stands for the prefix being compressed that means it is a link local prefix, II stands for the interface identifier being carried uncompressed and IC stands for the

Figure 9: LOWPAN_HC1 encoding

00	PI, II
01	PI, IC
10	PC, II
11	PC, IC

Table 2: HC1 encoding: IPv6 source (bits 0 and 1) and destination (bits 1 and 2) address

0	not compressed
1	both are zero

Table 3: HC1 encoding: Traffic class and flow label (bit 4)

00	not compressed
01	UDP
10	ICMP
11	TCP

Table 4: HC1 encoding: next header (bits 5 and 6)

0	no more header compression
1	HC2 encoding is following the HC1 encoding header

Table 5: HC1 encoding: HC2 encoding (bit 7)

interface identifier being elided as it is derivable from the link layer interface identifier. Analysing this one can see that in the ideal case having a HC1 encoding of "11111011" leads to an IPv6 header of as low as 2 bytes as 1 byte is used for the encoding field and the other one is the hop limit. All other fields are given implicitly.

By setting anything but 00 in the two next header bits as shown in table 4 and in addition setting the HC2 encoding bit to 1 as shown in table 5 it is possible to compress the next header as well. RFC4944 [9] describes a HC2 encoding for UDP packets called HC_UDP. This encoding mechanism allows to compress the source as well as the destination port and the length field. This modified header is of the form shown in fig.10, where the bits 3 through 7 are reserved for future use. The compression can be achieved according to tables 6 and 7. If a port is compressed, it is compressed to 4 bits and the full 16 bits port number is calculated by adding this short port to the fixed port 61616.

Figure 10: HC_UDP encoding

3.5 Integration

So far we have seen how in general IPv6 packets can be encapsulated into 802.15.4 packets (3.2). Then we have seen how those IPv6 packets can be compressed for the sensor nodes to be able to transmit them. But there is a last point

0	not compressed
1	compressed

Table 6: HC_UDP encoding: source (bit 0) and destination (bit 1) port

0	not compressed
1	compressed. Can be calculated from the IPv6 payload length

Table 7: HC_UDP encoding: length (bit 2)

that needs to be achieved and that is the one of integrating the sensor nodes into an existing IPv6 network. For that to be done we need a topology similar to the one shown in fig.11 where the sensor nodes can directly communicate with one another and if they want to open a connection with a full IPv6 station they have to route their packets over an edge router which translates those 802.15.4 packets into real IPv6 packets and vice versa.

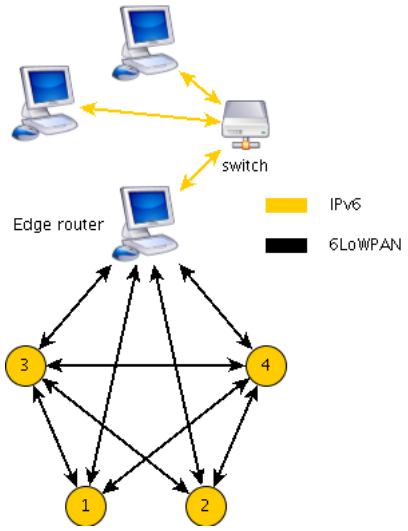

Figure 11: 6LowPAN network integrated into IPv6 network

4. MINISEC

4.1 Basics

MiniSec was developed to address and guarantee the needed security criterias "confidentiality", "authenticity" and "replay-protection" as well as freshness and low energy consumption. Thereby the two modes of communication must be addressed seperately as the replay protection cannot be guaranteed the same way. Namely this is unicast and broadcast which get addressed by MiniSec-U as described in 4.2.1 and MiniSec-B as described in 4.2.2, respectively. Authenticity is warranted in this protocol by the use of OCB (Offset Codebook) [12] as block cipher mode. Same is used to ensure confidentiality. Replay protection depends on the communicationmode, as already stated. As initialisation vector MiniSec uses a counter which is different depending on the mode of communication as well.

The key distribution system is not specified with MiniSec so any system can be applied to fit the individual needs for security.

4.1.1 OCB

In order to describe OCB [7] we need to define some notation first. The notations used are listed in table 8. It must be noted that MiniSec-U uses a directed key, i.e. there is a key for the communication from A to B as well as for the communication from B to A. In contrast Minisec-B uses one network-wide key per sender.

M	the message to be encrypted
H	an optional header that is correlated with M
E	a block cipher (e.g. AES)
K	the key to be used for the communication
N	a nonce
$S \ll 1$	Left-shift S by 1 bit, eliminating the first bit and appending a "0"
λ	$E_K(N)$ is the encryption of N with the cipher E and key K
ζ	$E_K(0)$
2S	$S \ll 1$ if S starts with a 0, otherwise $(S \ll 1) \oplus 0x000...087$
3S	$2S \oplus S$
5S	$(2(2S)) \oplus S$
S0*	S with appended 0s, such that the desired length is achieved
S10*	S with an appended 1 followed by 0s such that the desired length is achieved
len	the binary representation of the length represented with the desired amount of bits

Table 8: OCB notation

In the beginning M is divided into equally sized blocks while the last one may be of shorter size. The block size is chosen such that the block cipher used is capable to process the blocks. Additionally an arbitrary nonce N of the same size is required. The encryption works as shown in fig.12. The XOR of Pad and M_n means that only the first len(M_n) Bits of Pad are used. The checksum is calculated as follows: $M1 \oplus M2 \oplus ... \oplus M_{n-1} \oplus (C_m 0^* \oplus Pad)$.

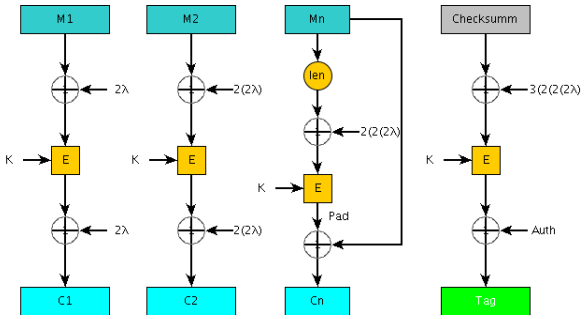

Figure 12: OCB main scheme

Auth can be neglected if no additional header H was handed over as in this case Auth only consists of 0 bits. Otherwise Auth is calculated according to fig.13. For that the header is divided into equally sized blocks while the last may again be

of shorter size. If H_n is as big as the other blocks, then ζ_{End} is equal to $3(2(2(2(5\zeta))))$. Otherwise H_n is padded with a 1 bit and as many 0 bits as needed (H_n10^*) and ζ_{End} is equal to $5(2(2(2(5\zeta))))$.

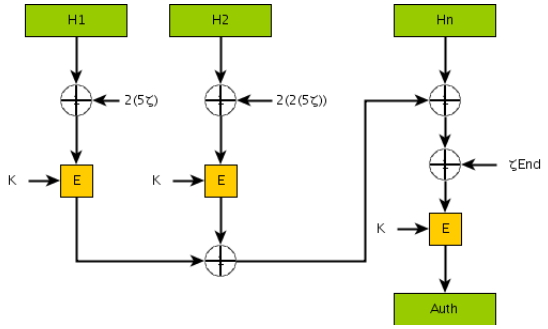

Figure 13: OCB auth calculation

The outcome then is the tupel containing the cipher-text C and the authentication tag "Tag". In order to prove authenticity and integrity the receiver only has to apply the inverse OCB on the cipher-text and calculate the tag on its own. The message was mangled or injected, if the received and calculated tags do not match.
The advantage directly arising for sensor networks is that authenticity and encryption are achieved in one step and therefore less computational resources are needed. In addition no unnecessary bits are transmitted as a shorter block is not enlarged to fit a given block size.

4.2 The protocol

4.2.1 MiniSec-U

MiniSec-U is the variant used if the connection is a unicast one. At this a message is sent by exactly one sender and is destined for exactly one receiver. For this connection a key $K_{A,B}$ is used for messages sent from A to B and a different key $K_{B,A}$ for the opposite direction. For each key a monotonous increasing synchronized counter $C_{A,B}$ or $C_{B,A}$ is needed in addition. This counters are used as initialisation vectors between the two parties A and B. This vector is not transmitted as a whole with every message, but only the last Δ bits as shown in fig.14. Thus no unnecessary in-

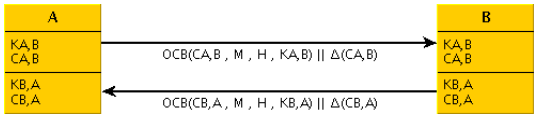

Figure 14: MiniSec-U communication scheme

formations about the initialisation vector are transmitted. This is of benefit for sensor networks since transmission of data is the most energy consuming operation. Each of the parties increases the internally stored counter with each sent or received message. The receiver in addition can adjust its counter based on the last Δ bits received with the message, if the counter is no more in sync. If too many messages were lost in transmission and so simply adjusting according to the last few bits does not work, it is still possible to adjust the counter by gradually increasing the counter by 2^{Δ}. If this does not work after repeated attempts, resynchronisation will be needed which is quite energy consuming.

4.2.2 MiniSec-B

This variant is used if broadcast messages need to be transmitted. It is clear that in a broadcast environment it is not possible to use the already exchanged unicast keys. Therefore a network-wide key is used for every principal that wants to send a broadcast message. This key is used to encrypt the message with OCB. As it is not possible for low memory reasons that every node holds a counter for every possible sender, a different approach must be used. Nevertheless it is possible to guarantee replay protection by combining two Methods. These are the sliding-windows and bloom-filters [2].

With the sliding-windows approach a finite time t_e is defined which is used to determine the length of a so called epoch. Afterwards the time is divided into epochs of length t_e. By a loose time synchronisation between the nodes the current epoch E is known to every participant. t_e is chosen such that it equals twice the time of the maximum possible time synchronisation error Δ_S plus the maximum network latency Δ_L ($t_e = 2 * \Delta_S + \Delta_L$). By using the current epoch number as initialisation vector for OCB, packets from past epochs could automatically be detected as replays. But since by the time synchronisation error and network latency a correctly sent message can reach the receiver in a wrong epoch, two epochs will be put into consideration. As shown in fig.15 based on the time of arrival either the previous or the following epoch will be considered in addition. Let t be the

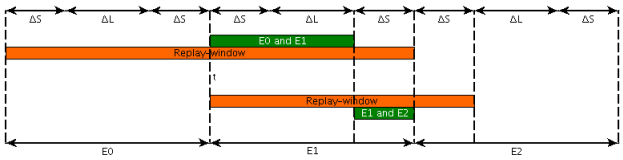

Figure 15: Sliding-window approach and Replay-attack window

time at the beginning of epoch E1 as shown in fig.15. Then epochs E0 and E1 will be considered if and only if the arrival time t_A is between t and $t + \Delta_S + \Delta_L$. Epochs E1 and E2 are considered accordingly if and only if t_A is between the rest of the current epoch, i.e. between $t + \Delta_S + \Delta_L$ and $t + 2 * \Delta_S + \Delta_L$.
To validate that this division is correct, we consider the least possible time and the highest possible time the message could be sent such that it arrives in the respective time window.

Case 1: Packet arrives such that E0 and E1 are put into consideration. In this case the least possible time s_{min} is $t - \Delta_S - \Delta_L$ lying in E0. The maximum possible time s_{max} is $t + \Delta_S + \Delta_L + \Delta_S$ lying in E1. Hence this part is correct.

Case 2: Packet arrives such that E1 and E2 are put into consideration. In this case the least possible time s_{min} is $t + \Delta_S + \Delta_L - \Delta_S - \Delta_L = t$ and lyes therefore in E1. The

maximum sent time s_{max} is $t + 2 * \Delta_S + \Delta_L + \Delta_S$ lying in E2. Hence this part is correct either.

This shows directly that packets captured at the beginning of an epoch can be replayed during this whole epoch and in addition up to $\Delta_S + \Delta_L$ in the following epoch.
In order to guarantee replay protection during this time window, the sender holds an internal counter which is increased with every sent packet. The counter is reset at the end of every epoch and can therefore be very short. The advantage of a short counter is that it can be transmitted as a whole and the receivers do not have to save the state of the counter themselves. In addition bloom-filters are used.
Bloom-filters are probabilistic datastructures which can be used to check whether an element is present or not and that in a very memeory and processor efficient way. The only two possible operations are adding elements and checking whether an element is present. Deleting is not possible. When checking for the presence of an element false-positives are possible false-negatives in contrast are not. For this to achieve an array is used which is filled initially with 0 bits. By adding an element this element gets mapped by different hash-functions to different indices which are then set to 1. In order to check whether an element exists one has to calculate all the hash-functions and check whether all fields in the array that the indices refer to are set to 1. If even one such array field is set to 0 it is sure that the given element was not yet added. This procedure is shown schematically in fig.16. The initialisation vector is constructed by the concatenation

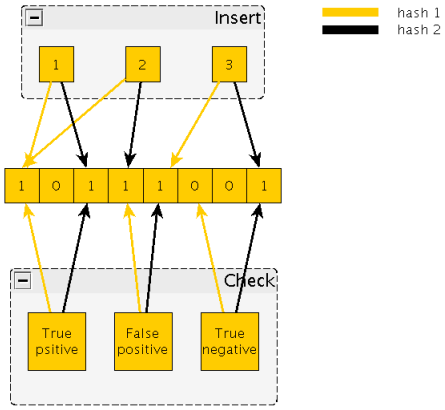

Figure 16: Bloom-Filter

of the senders ID, the counter of the sender and the current epoch number. Since the epoch number is never reset this vector will never be reused. The receiver has to manage two separate bloom-filter in which the received packets are inserted. As key for the insertion the concatenation of the senders counter and the senders ID is used. One has to use two separate bloom-filters since as shown in fig.15 two epochs are checked at every time. Further the counter of the sender is reset at the end of every epoch and therefore an assignment to the epoch is necessary. It would be possible to add the epoch number to the key when inserting into the bloom-filter, but by that many elements would be added to the filter very quickly. Hence the false-positive rate would increase steadily. By resetting the bloom-filter after its validity and reusing it for a new epoch the problem of an increasing false-positive rate is elided.
One of the two filters always belongs to the current epoch and the other one belongs either to the previous or the following epoch. Being within $\Delta_S + \Delta_L$ from the beginning of the epoch the second filter will be used for the previous epoch. When leaving this time slot the filter is reset and used for the following epoch.
As a valid message must belong to any of the considered epochs and the counter as well as the ID of the sender is sent in plain text, one can determine the correct epoch by decrypting the message. To accomplish this one simply has to decrypt the ciphertext with both epoch numbers one after another and check whether the decryption succeeded. Afterwards it can be checked whether the message already exists in the corresponding bloom-filter. If this check returns true it is most likely that the message was replayed and can therefore be dropped. Otherwise it is for sure that it is new and can be accepted and inserted into the bloom-filter.

4.3 Security analysis

As stated in [14] in order to analyse security it is at least necessary to consider the criterias confidentiality, authenticity and in the case of sensor networks freshness. In the following im going to split authenticity further into accountability and integrity.

Confidentiality: Confidentiality is given in both MiniSec-U as well as MiniSec-B by applying encryption utilising OCB (4.1.1). This procedure is as safe as the underlying block cipher that is used. So by using a cipher that has no yet known security threats this criteria can be considered as proofed. Semantic security is given in case of MiniSec-U by using a counter as IV for every direction. In case of MiniSec-B this is achieved by using the epoch number and a shorter counter.

Integrity: Integrity is directly given in both variants by the tag. Here the security directly depends on the length of the tag. The probability to guess a correct tag with a length of 32 bit is 2^{-32} and this can be considered as inveasible.

Accountability: MiniSec-U directly provides accountability by using a seperate key for every node and direction. Without the knowledge of this key it is inveasible to calculate a correct tag and can therefore not masquerade as another node. MiniSec-B in contrast uses a network wide key that every receiver knows. By that every node connected to the network that has received this key can masquerade as the sender since all other parameters like the counter are known as well. This could be remedied by not allowing to use a network wide key a second time.

Freshness: In the variant of MiniSec-U monotonous increasing counters are used as initialisation vectors which are kept internally by both parties. Hence the receiver can check at any time whether the packet is fresh or replayed. In MiniSec-B it would be possible to replay packets during a certain time window if only sliding-windows were applied.

By additionally utilising bloom-filters these packets can be detected. Replayed packets from old epochs are detected by the sliding-windows and replayed packets that would be accepted by the sliding-windows are detected by the bloom-filters. Hence freshness is guaranteed with both variants.

4.4 Comparision to similar protocols

4.4.1 TinySec

As stated in [6] TinySec uses CBC (Ciffer Block Chaining) as block cipher mode and in addition for authentication a CBC-MAC. The great disadvantage with that is that in order to calculate the authentication code the whole encryption algorithm has to be recalculated a second time. One cannot simply use the same key for encryption and authentication as this would lead to great security threats [1]. In this case it would be possible to change the ciphertext without the possibility to notice the modification. Further CBC-MAC uses an initialisation vector 0 and therefore the first encryption round cannot be reused. TinySec uses a 8 bytes long initialisation vector which is used as a counter as MiniSec does in order to prevent a quick repetition. However this initialisation vector is appended to every packet which means an overhead of 8 bytes minus a few bits in comparison to MiniSec. The last block is not enlarged to a given block size but kept as short as possible by using "ciphertext stealing" [13]. TinySec uses a network wide key even for unicast communications. This arises the advantage that no key-exchange algorithms have to be performed neither in unicast nor in broadcast communications. This is achieved for the cost of lower security as accountability can not be guaranteed at any degree. Further, none of the receivers saves anything about the received packets or the state of the last initialisation vector. So freshness can't be guaranteed as well.

4.4.2 SNEP & μTESLA

Another possible protocol combination is SNEP (Sensor network encryption protocol) [11] for unicast connections and μTESLA [11] for broadcast authentication.

SNEP is a procedure which guarantees confidentiality, authenticity and freshness for unicast connections. Confidentiality is achieved by using symmetric cryptography as MiniSec does, but SNEP only uses one key per communication pair. As initialisation vector there is a counter for each direction which is managed by both parties like it is managed in MiniSec. In comparison to MiniSec this counter is not at all transmitted with the packets and thus reduces the energy consumption. The drawback is that in lossy networks the counter cannot be resynchronized without the influence of the other party and the resynchronization consumes much energy. By using a non repeating counter as initialisation vector, freshness is given out of the box as no packet can be replayed. Authenticity is achieved by a separate MAC. This is as already discussed in 4.4.1 a drawback in so far as the whole message needs to be taken into account another time.

In order to authenticate broadcast messages one can use a simpler variant of TESLA [10] called μTESLA. This is a procedure which can be used for authentication but not for encryption. It is again necessary that the parties are loosely time synchronized in order to divide the time in parts as it is done in MiniSec-B (4.2.2). In each of these parts a separate key K_i is used. For this purpose the sender generates a key K_n and calculates based on that subsequent keys K_{n-i} by using a one-way-function F as shown in fig.17. Based on this function every node can calculate the key K_{m-i} out of K_m but never K_{m+1}. By negotiating a key K_0 between the sender and all the receivers, every receiver can proofe the authenticity of a key with a higher index. The sender

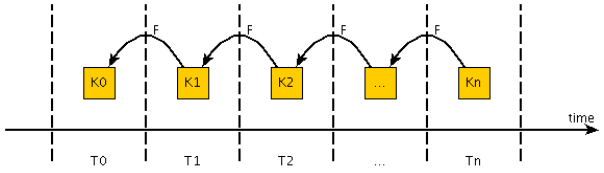

Figure 17: μTESLA representation

authenticates in the time slot T_i every packet with the key K_i and the receiver buffers the packets in the first instance in memory. After a certain defined amount of time the sender discloses the key K_i whereupon the receiver can check the authenticity of this key by subsequently utilising the one-way-function F, whereas the following must apply: $K_0 = F^i(K_i)$. Is this the case all the packets buffered for the time slot T_i can be authenticated with the Key K_i and this key can now be set to be the new trusted key so that the applied one-way-function chain is kept as short as possible. The advantage in comparison to MiniSec is that no other node can masquerade as the sender as keys with a higher index cannot be calculated and old keys are no more valid after disclosing it. The drawback however is that this can nearly not be used for sensor nodes to send broadcast messages, but only to receive them. This is because the full key chain cannot be saved in the limited memory of such nodes and recalculation of the keys all the time is very computationally expensive.

4.4.3 Common security protocols

Common security protocols have the big advantage that they are widely used and are therefore tested by a wide variety of users and got analysed by security experts. The drawback however is that they are not built to be used in an environment with limited resources. Common security protocols often implement very complex algorithms to achieve a very high rate of security. However this is not possible on sensor nodes as they are very limited in memory and computing power. Further most of the protocols rely on TCP which needs many packets to be received and sent which consumes much of the limited energy. Asymmetric cryptography would be very secure but considering for example a 2048 bit key would impose that every node saves the public keys of its neighbours and this is nearly impossible because of the very limited memory. In addition assymmetric encryption algorithms are way more computationally expensive than symmetric ones. Another criteria is as discussed in [6] that in conventional networks most connections are end-to-end ones and the intermediate nodes (e.g. routers) do not check the packets but simply forward them as transmission does not matter. In sensor networks however data transmission is the most energy consuming factor and therefore such a procedure should not be chosen.

5. CONCLUSION

As we have seen 6LoWPAN is fairly well suited for the integration of 802.15.4 capable devices into existing IPv6 networks. The specification not only allows these devices to be integrated but in addition defines procedures that are optimized for the usage in environments with very limited resources and therefore can be used without any doubt even with very cheap and simple sensor nodes.

MiniSec is, as well as 6LoWPAN is, highly optimized for the use with sensor nodes and networks. This protocol tries to eliminate all the overhead that is not absolutely needed to guarantee the security requirements. But it does not simply leave away every conceivable overhead but tries to find the golden mean between letting informations away and sending that much that no additional negotiation is needed. Nevertheless at least the unicast method fully assures the security requirements in wireless networks. The broadcast variant ensures the main criterias namely confidentiality, integrity and freshness as well, but not accountability. So this protocol is quite well suited for sensor networks particularly as it is not vulnerable to any known attack except brute-forcing, which is always a possible attack.

6. REFERENCES

[1] M. Bellare, J. K. T, and P. Rogaway. The security of the cipher block chaining message authentication code, 2001.

[2] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13:422–426, 1970.

[3] M. Crawford. RFC2464 transmission of ipv6 packets over ethernet networks. http://www.faqs.org/rfcs/rfc2464.html, December 1998.

[4] S. Deering and R. Hinden. RFC2460 internet protocol, version 6 (ipv6) specification. http://tools.ietf.org/html/rfc2460, December 1998.

[5] IEEE Computer Society. Ieee std. 802.15.4-2003, December 2003.

[6] C. Karlof, N. Sastry, and D. Wagner. Tinysec: a link layer security architecture for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 162–175, New York, NY, USA, 2004. ACM.

[7] T. Krovetz and P. Rogaway. The ocb authenticated-encryption algorithm. http://www.cs.ucdavis.edu/ rogaway/papers/draft-krovetz-ocb-00.txt, March 2005.

[8] M. Luk, G. Mezzour, A. Perrig, and V. Gligor. Minisec: a secure sensor network communication architecture. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 479–488, New York, NY, USA, 2007. ACM.

[9] G. Montenegro, J. Hui, and D. Culler. RFC4944 transmission of ipv6 packets over ieee 802.15.4 networks. http://tools.ietf.org/html/rfc4944, September 2007.

[10] A. Perrig, R. Canetti, J. D. Tygar, and D. Song. The tesla broadcast authentication protocol. *RSA CryptoBytes*, 5:2002, 2002.

[11] A. Perrig, R. Szewczyk, J. Tygar, V. Wen, and D. E. Culler. Spins: Security protocols for sensor networks. *Wireless Networks*, 8(5):521–534, September 2002.

[12] P. Rogaway, M. Bellare, and J. Black. Ocb: A block-cipher mode of operation for efficient authenticated encryption. *ACM Trans. Inf. Syst. Secur.*, 6(3):365–403, 2003.

[13] B. Schneier. Applied cryptographie, second edition. John Wiley and Sons, 1996.

[14] L. Tobarra, D. Cazorla, F. Cuartero, and G. Diaz. Analysis of security protocol minisec for wireless sensor networks. In *Proceedings of the IV Congreso Iberoamericano de Seguridad Informatica (CIBSI 2007)*, pages 1–13, November 2007.