Asynchronous Group Key Distribution on top of the CC2420 Security Mechanisms for Sensor Networks

Morten Tranberg Hansen
Department of Computer Science
Aarhus University, Denmark
mth@cs.au.dk

ABSTRACT

A sensor network is a network consisting of small, inexpensive, low-powered sensor nodes that communicate to complete a common task. Sensor nodes are characterized by having limited communication and computation capabilities, energy, and storage. They often are deployed in hostile environments creating a demand for encryption and authentication of the messages sent between them. Due to severe resource constraints on the sensor nodes, efficient key distribution schemes and secure communication protocols with low overhead are desired. In this paper we present an asynchronous group key distribution scheme with no time synchronization requirements. The scheme decreases the number of key updates by providing them on an as needed basis according to the amount of network traffic. We evaluate the CC2420 radio security mechanism and show how to use it as a basis to implement secure group communication using our proposed group key distribution scheme.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General - Security and protection

General Terms

Performance, Security

Keywords

CC2420 security, group key distribution, sensor network security

1. INTRODUCTION

Among a broad area of applications for sensor networks is environmental monitoring (such as monitoring a 70-meter tall redwood tree [24]), and structural health monitoring (such as monitoring of temperature and humidity in civil infrastructures and concrete elements [3]). The main goal of these applications are to transport monitored data from resource constrained sensor nodes back to one (or more) base station(s) (see Figure 1). These base stations are considered to be trusted and have sufficient resources for processing, analyzing, and storing data. The presence of base stations

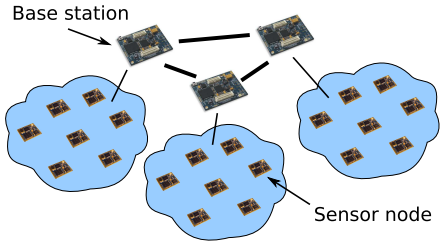


Figure 1: Network Model.

makes application development easier and less complex [6], and gives us a trusted base for enabling efficient security in the network. We define a *group* as a set of sensor nodes associated with a specific base station. We do not restrict a sensor node to only one group. We have identified five communication patterns present in each group: 1) nodes to neighbor nodes (e.g., neighborhood discovery), 2) base station to all nodes (e.g., query dissemination), 3) all nodes to base station (e.g., data collection), 4) base station to a single node (e.g., individual queries), and 5) a single node to base station (e.g., data or status notifications). Depending on the level of trust between nodes in a group an increasing number of these communication patterns can be done using secure group communication as an alternative to secure point-to-point communication. This emphasizes the need for an efficient secure group communication scheme which is the focus in this paper.

The desired security properties includes confidentiality, authentication, integrity, protection against replay attacks, and data freshness. To achieve semantic security it is important that one piece of data is not encrypted using the same key twice. To prevent this it is common to include, in the encryption process, a unique nonce that is a concatenation of a counter, the current key identifier, and the source address. To keep the nonce unique under the same key the counter cannot wrap-around. The counter can also be used as a mechanism against replay attacks and provides data freshness [17].

Schemes for pair-wise (session) key establishment [9], group key establishment [8, 16, 23], and secure communications [10, 15, 16, 17] have been proposed. In this paper we build on top of these ideas to achieve secure group communication with no time synchronization requirements and improve efficiency by showing how to utilize the hardware security mechanism of the Chipcon CC2420 IEEE 802.15.4 (CC2420) radio. The main contributions of this paper are as follows:

- An efficient asynchronous group key distribution scheme with no time synchronization requirements such that new keys can be established as needed based on the amount of traffic in the network. We do this by distributing keys from the base sta-

tion well in advance and keeping a local buffer of upcoming and recent keys on the sensor nodes.

- Analysis and evaluation of the CC2420 radios security mechanism on the TelosB mote [18] using TinyOS 2.x [13] and a demonstration of how to utilize these to improve efficiency of secure group communication.

We assume that initially a pair of symmetric keys are securely installed [11] and shared between the base station and all sensor nodes. From this key, nodes can be authenticated [9], session keys can be efficiently established [9], and secure communication from the base station to a node is possible [15]. Due to the base stations superior resources and their role as a collection point for all data, they are considered responsible for implying a desired level of intrusion detection to improve the resilience of the network. We assume the presence of a CC2420 radio [1] on all sensor nodes.

In section 2 we present our asynchronous group key distribution scheme and in section 3 we evaluate the security features of the CC2420 radio and present an efficient way to utilize these for implementing secure group communication at the link layer. In section 4 we evaluate and compare our asynchronous group key distribution to a synchronous version and estimate the overhead incurred by implementing secure communication with the CC2420 in-line security mechanisms. In section 5 we review related work and in section 6 we conclude our work.

2. ASYNCHRONOUS GROUP KEY DISTRIBUTION

To reduce the risk of cryptoanalysis key updates are necessary as it is not secure to use the same key over a long period of time. Therefore the basics for implementing secure communication is a secure key distribution scheme. In this section we propose a secure, efficient, reliable, and asynchronous group key distribution scheme that provides key updates as needed without any time synchronization requirements. Key updates are traditionally done periodically over time or re-actively whenever a node joins, leaves, or is evicted from a group. Periodical key updates have the disadvantage of not being able to adapt to a changing amount of network traffic, whereas the reactive approach can introduce an unnecessary high frequency of key updates in environments with varying amounts of traffic. We adopt a hybrid approach, with the base station as the decision maker. The base station role as a data collection point makes it a preferable place for doing intrusion detection (as was part of our assumptions) and also for deciding when to issue key updates. We realize that not all control traffic goes through the base station, but argue that this amount can be estimated from the network size and how it is configured.

Keys will be distributed from the base station to the initially not trusted sensor nodes in the group. Initially the base station authenticates the sensor nodes with their pair-wise shared secret key [9]. From this a secure channel can be established and sensor nodes can be securely configured to participate in the group key distribution scheme. When a new sensor node joins, it is authenticated and configured in the same way. When a sensor node is compromised it is excluded from the group and the remaining sensor nodes need to be re-intialized.

2.1 Key-chains

The keys distributed from the base station are based on a key-chain generated by a cryptographic one-way function. Key-chains were first adapted to sensor networks by Perrig et al. [17] and used in the μ-TESLA authenticated broadcast protocol. A key-chain of length n is generated from the last key in the chain, K_n, and a cryptographic one-way function h. The key K_n should be chosen at random and the other keys are computed as $K_i = h(K_{i+1})$. At a certain point in time each of these keys will be used as a secret group key. The base station is assumed to have sufficient resources for storing a large key-chain of keys that can be distributed to the sensor sensor nodes over time.

2.2 Key Buffer

Due to their limited resources, sensor nodes cannot locally store the entire key-chain. Instead they maintain a limited buffer of relevant keys. The buffer has three main purposes: to authenticate new keys, to discover and recover lost keys, and to be backward compatible with other sensor nodes using older keys.

To initiate the scheme the base station encrypts, authenticates, and transmits an initial group key to all sensor nodes in the group. If the key buffer size is b and this initial key is K_b the whole key buffer can be filled with keys due to the one-way property of the cryptographic function. A node's current group key is then chosen to be the key positioned in the middle of the key buffer. This key, a counter, and the nodes source address (all needed to construct the unique nonce), is then used to perform secure group communication.

To perform a key update, the base station encrypts and broadcasts the next key in the key chain under its current key. When a new key arrives at a sensor node (see Figure 2 step 1) it can be implicit authenticated by comparing the output of the public known cryptographic one-way function taken on the new key to the latest key in the buffer (see Figure 2 step 2). Due to the unreliable nature of the wireless channels an intermediate key update may have been lost. To verify a correct key after a number of lost key updates, a sensor node needs to apply the authentication procedure recursively on the new key (in Figure 2 key K_6 is lost). To limit the computational overhead of the scheme, an upper limit for recursively authentication tries will be set. When a new key is authenticated the key buffer needs to be updated accordingly (see Figure 2 step 3). The new key will be the latest key in the buffer and in case of intermediate key losses they will be restored from the properties of the cryptographic one-way function. In Figure 2 step 3 the lost key K_6 is restored as $K_6 = h(K_7)$.

When a sensor nodes key buffer has been updated its current group key needs to be adjusted accordingly. The current key will only be changed following a key update or a counter wrap-around. The latter case is due to the importance of keeping the nonce unique. This means that in case of a counter (used to construct the nonce) wrap-around, the current key has to be set to the next key in the buffer, if not; the semantic security property can be violated. To increase the chance of being able to do secure communication with

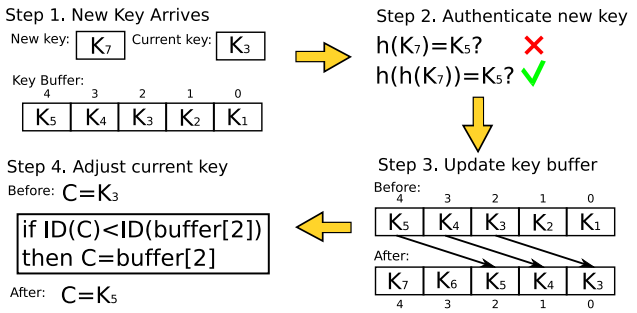


Figure 2: Key update at a sensor node.

	CC2420	SkipJack	AES
Encryption	96	460	1149
Write + Enc. + Read	2008	N/A	N/A
Write + Enc.	1105	N/A	N/A
Setup	1113	46	1844

Table 1: Performance Evaluation of the CC2420 radios stand-alone AES encryption. All units in μs.

nodes not having received the same amount of key updates, the current key should be positioned in the middle of the key buffer. So on key updates, the current key is increased according to its position in the key buffer after a number of new keys have been inserted (see Figure 2 step 4). If the current key stays in the upper half of the key buffer it is not changed. If it ends up being in the lower half of the key buffer, it will be changed to the middle key in the key buffer. On Figure 2, the current key is updated from K_3 to K_5 after the arrival of the new key K_7.

The fact that the base station can do key updates as needed, while nodes with different keys are still able to do secure group communication, makes the scheme asynchronous.

3. CC2420 GROUP COMMUNICATION

In this section we evaluate the CC2420 radio security mechanism and show how to use these as the basics for doing secure group communication with our group key distribution scheme. In our evaluation we use time as a performance measure. We assume that security is used in relation with packet transmission, so this time measure can be related to the actual energy usage, as it defines the time where the radio and the microprocessor (most CC2420 radio operations in TinyOS 2.x are synchronous) needs to be on. In other scenarios, where the CC2420 radio could be turned off instead of encrypting/decrypting data, it is likely that software implementations will be more energy efficient. This is out of the scope of this paper. All experiments are done using TinyOS 2.x [13] running on the TelosB platform [18] where performance times are measured from a 16 bit microsecond counter which is based on one of the microprocessor timers.

3.1 CC2420 Background

The CC2420 radio features the three hardware IEEE 802.15.4 in-line security suits that works on packets within the receive and transmit buffers: Counter (CTR) mode encryption, Cipher-Block Chaining MAC (CBC-MAC) authentication, and Combined Cipher Machine (CCM) authenticated encryption [1]. The security is based on an AES hardware implementation (taking a 128 bit key) which can also be used for stand-alone AES encryption on a 128 bit plaintext.

The way the CC2420 radio handles access control lists varies from the one required by the IEEE 802.15.4 specification [2]. It only has two entries but instead of having an address associated with them it has two registers specifying what key to use for transmission and reception, respectively. Each mode has an associated nonce which should be unique for all packets sent with the same key. The nonce should be set by the upper-layers before the security mechanism is used. By letting the upper-layers handle the key and nonce management the CC2420 radio circumvent a lot of the IEEE 802.15.4 flaws identified by Sastry et al. [22].

3.2 CC2420 Stand-Alone Encryption

The basis for doing security is a block cipher. Law et al. [12] benchmarked the most common software implementations of block ciphers for wireless sensor networks and found that Skipjack was the most memory efficient and Rijndael [5] the most secure. In this section we will explore the CC2420 radios stand-alone AES encryption feature working as a block cipher, and compare it to software implementations of SkipJack from TinySec [10] and a byte oriented AES implementation from Brian Gladman[1].

	CC2420	SkipJack	AES
RAM	33	42	244
ROM	1734	228	834

Table 2: Memory Evaluation of the CC2420 stand-alone AES encryption. All units in bytes.

The CC2420 radio stand-alone AES encryption takes a 128 bit key and a 128 bit plaintext and produces a 128 bit ciphertext. The encryption process is enabled by calling the SAES command strobe after writing a key to either KEY0 or KEY1 stored at RAM locations 0x100 and 0x130, setting the SECCTRL0 register to point at the key to use, and writing the plaintext to the SABUF memory space at location 0x120. We let the writing of the SECCTRL0 register and a key to the CC2420 RAM be part of the setup procedure, as they do not need to be set for each operation of the block cipher. A block cipher operation consists of writing the plaintext to the CC2420 RAM, enabling the stand-alone AES encryption, and then reading the ciphertext back again. In case of continues block cipher operations, the intermediate reads can be spared due to the CC2420 radios functionally of simultaneous write and read a buffer.

Table 1 shows the performance comparison. All values are excluding an encryption overhead of 194μs, 5μs and 2μs, and a setup overhead of 352μs, 78μs, and 78μs from respectively the CC2420, SkipJack, and AES block cipher. The overheads include acquisition of the SPI bus, creation of local variables, and function calls. Table 1 shows that the encryption operation on the CC2420 is up to ten times faster than the software implementations. The CC2420 datasheet [1] states that it can run as fast as 14μs, but we are not able to measure performance times lower than a command strobe which takes about 80μs to execute in TinyOS 2.x (the 96μs includes chip select). The bottleneck of the CC2420 is the RAM read and write. We see that it is comparable to the software implementation of AES and SkipJack under continuous block cipher operation (read/write performed as one operation). Note that a SkipJack block cipher operates on a 64 bit data string compared to a AES block cipher that operate on a 128 bit data string. The setup time for the CC2420 is dominated by the time it takes to write the key to the CC2420 RAM. It is an order of magnitude slower than the SkipJack setup time, but performs better than the AES.

Table 2 shows a memory evaluation of the block ciphers. We see that the CC2420 RAM overhead is lower than the software implementations, which can be important in some applications. The CC2420 includes several hardware presentation layers of software to interact with the CC2420 radio which increases its program memory compared to the other solution. The hardware presentation layers will only be included once, therefore the difference will be less significant if the CC2420 radio stack is also used in an application.

3.3 CC2420 In-line Security

In last section we showed that the CC2420 radios stand-alone AES security operation perform better than the software alternatives, if we neglect the RAM read and write. The CC2420 radio in-line security mechanism is performed on packets already present

[1]http://fp.gladman.plus.com/AES

in the CC2420 radios TXFIFO and RXFIFO buffers. Hence, has the potential to cause little overhead, as the read and write to these buffers is done anyway. In this section we will explore the CC2420 radio in-line security mechanism.

The encryption is enabled by calling the STXENC command strobe after configuring the SECCTRL0 and SECCTRL1 registers (see the CC2420 radios data sheet [1]), writing the nonce to TXNONCE at RAM location 0x140, and the key to either KEY0 or KEY1 at RAM locations 0x100 and 0x130, respectively. The nonce consist of the source address, a frame counter (keeping the nonce unique for each use under the same key), and a key sequence number. The decryption is enabled in a similar way, using the SRXDEC command strobe and a nonce written to RXNONCE at RAM location 0x110.

Table 3 shows a performance evaluation of the CC2420 radio's CCM, CTR, and CBC-MAC in-line security mechanisms. All values are excluding the time it takes for a SNOP command strobe to check when the security operation is complete. As before, we are not able to measure execution times lower than a single command strobe. From Table 3, we see that in accordance with the CC2420 radios data-sheet [1] that the in-line security mechanism are very fast even when compared to a single block cipher operation.

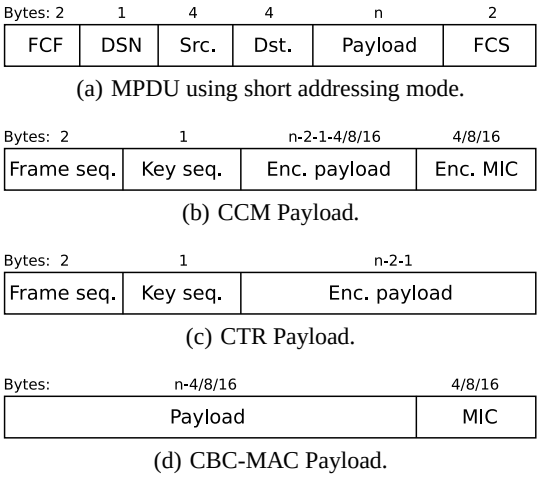
3.4 Protocol Considerations

An efficient security mechanism is not enough for an efficient secure communication scheme. In order for the receiving part to be able to decrypt a received packet, it needs to be able to reconstruct the used nonce. The IEEE 802.15.4 specification [2] proposes to prepend the frame counter and key sequence number of the nonce to the packet payload as shown in Figure 3 (b), (c), and (d). This increases the communication overhead which may not be acceptable for low power applications. In the IEEE 802.15.4 specification the data sequence number (DSN) field from the MAC Protocol Data Unit (MPDU) in Figure 3 (a), is used as a mechanism for relating an acknowledgement to its data packet. This means that it will conform to the specification as long as it changes for each subsequent packet. In order to decrease the communication overhead we propose to use the DSN for the frame counter and key sequence number. The size of the DSN is one byte, so in order for the frame counter and key sequence number to be larger we adopt the technique from Luk et al. [15] to use it for implicit synchronization of internal stored larger variables. The applicability of storing these variables comes from the fact that we implement security at the link layer; we only store them for our neighbors. Security at the link layer may increase the communication overhead but comes with the advantage of supporting secure data aggregation [19].

We let the x least significant bits of the DSN be our frame counter synchronization. This means that as long as there is less than 2^x dropped packets since the last successfully received packet from the same sender, the receiver can immediately increase its frame counter to the right value. In fact, this optimization is still useful even if 2^x or more packets are lost, since the receiver could continue to increment the counter by 2^x and reattempt decryption. Similarly, we let the $8-x$ most significant bits of the DSN be our key synchronization.

Mode \ Length	5	10	20	28
CCM	97	97	97	193
CTR	97	97	97	97
CBC-MAC	97	97	97	97

Table 3: Performance Evaluation of the in-line security mechanisms. All units in μs.



(a) MPDU using short addressing mode.

(b) CCM Payload.

(c) CTR Payload.

(d) CBC-MAC Payload.

Figure 3: The IEEE 802.15.4 MPDU and packet payloads.

4. EFFICIENCY EVALUATION

4.1 Asynchronous Key Distribution

The main contribution of our scheme is that it is asynchronous and does not require any time synchronization. This means that the base station can do key updates when they are needed instead of doing them synchronously over fixed periods of time. The base station will decide when to do key updates by estimating how much traffic is generated in the group. Not all control traffic will go through the base station, but we argue that an upper bound can be put on such traffic, and the amount should be considerable lower than the amount of data traffic that will go through the base station. The strategy for the base station is to do key updates whenever a group member is in need of one, hence a certain amount of data has been successfully sent to the base station from a specific sensor node. We assume a reliable data link layer and simulate the need for key updates over an unreliable channel, and compare our asynchronous solution to the synchronous solution proposed in Lisp [16], with different probabilities for a sensor node to send data to the base station. We implemented the core features of the schemes in TinyOS 2.x [18] and simulated key updates between two neighboring nodes in TOSSIM [14]. We assume that the active scheme is initialized beforehand and a re-initialization is to be avoided at all costs. One node is periodically, with a certain probability, deciding upon whether or not to send a data packet to the other node. The other node issues key updates according to the current distribution scheme, assuming that the first node can send a data packet at the beginning of each period.

Figure 4 shows the overhead in the form of key updates and request for key updates when increasing the chance of lost key updates (we consider a garbled update as lost). The simulation lasts 500 seconds, using a period of 1 second, a key buffer size of 5, and a new key is needed for every 5 packets sent. These values were chosen to illustrate the difference between the schemes, and may be larger in a real scenario. Results are shown for the asynchronous and the synchronous scheme when the probability of the node sending data is 10%, 50%, and 100%, respectively. The overhead introduced by the synchronous scheme is relatively stable until the probability of a message loss reaches 70%. After this, the length of the key buffer is not able to keep up for the loss of key

	Transmit	Receive
Read DSN	N/A	211
Compute key	N/A	6
Compute counter	N/A	6
Read source	N/A	254
Write key	948	948
Write nonce	911	908
Write registers	384	384
Security strobe	196	214
Overhead	2439	2931

Table 4: Computational overhead of using the CC2420 radios in-line security mechanism in CCM mode. All units are in μs.

updates, and the overhead increases. The asynchronous scheme uses half of the key buffer to handle lost key updates and the other half to be compatible with other nodes using older keys, whereas the synchronous scheme uses the whole key buffer for handling lost key updates. This means that with the same key buffer length, the asynchronous scheme is not able to handle the same amount of lost key updates, and the overhead is seen to increase when half of all key updates get lost. This could be compensated for with a larger key buffer or by re-computing older keys as needed. Re-computing an older key will increase the security overhead per packet with at least one block cipher operation taking approximately 1 ms according to our evaluation. This is at least a 25% increase compared to our computational overhead evaluation in the next sub-section.

The strength of the asynchronous scheme is when the amount of data sent decreases to 10% or 50% of its potential. In such case, the asynchronous schemes overhead is proportional to the amount of data sent whereas the synchronous key update keeps a high overhead.

Due to page restrictions, we leave out a computational overhead evaluation, but the results are similar to the ones showed in Lisp [16].

4.2 CC2420 Security Overhead

In this section we evaluate the overhead involved when implementing secure communication with the CC2420 radio in-line security mechanism. The evaluation will be based on the CCM mode as this sets an upper bound. Due to the security flaws identified by Sastry et al. [22] we do not recommend the user to use other security modes than CCM.

The security computational overhead when transmitting a packet will be less than when receiving one. This is due to the fact that the receiver has to read bytes from the RXFIFO buffer in order to determine what counter, key, and source address to include in the decryption process. Table 4 shows the computational overhead of using the CC2420 radios in-line security mechanism in CCM mode. All values are measured by an experimental study on the TelosB platform [18] using TinyOS 2.x [13] with a CC2420 packet size of 41 bytes. The transmission process includes writing the packet, key, and nonce to the CC2420 radios RAM, setting the security registers, and completing the encryption. The receive process includes reading the DSN (1 byte) and source address (2 bytes) from the CC2420 radios RAM, determining what nonce to use, writing the key and nonce to the CC2420 radios RAM, setting the security registers, and completing the decryption. We see that the computational overhead of the security is between two and three milliseconds. According to Rogaway et al. [21] a similar software implementation of CCM, will require 7 block cipher calls taking 128 bits of plaintext at a time on the same packet length. If we compare this to any block cipher from our block cipher evaluation, the in-line security mechanism will always be more than twice as fast as a similar software implementation.

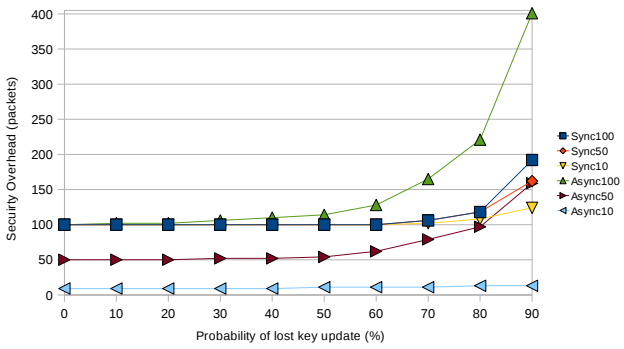


Figure 4: Asynchronous vs. Synchronous key updates.

The communication overhead of each secure packet sent is minimal as we are re-using the DSN field to synchronize the frame counter and key sequence number. If implicit synchronization is not possible, there will be a cost associated with having to do external synchronization of these values, as described by Perrig et al. [17]. We consider this case to be rare.

5. RELATED WORK

5.1 Group Key Distribution

The Lightweight Security Protocol (LiSP) [16] provides a secure group key distribution scheme that periodically renews a shared key. It achieves reliability without retransmissions, implicit authentication without incurring any additional overhead, detection and recovery from lost keys, and key refreshments without disrupting ongoing data transmission, but requires loose time synchronization. LiSP makes use of a one-way hash function to create a key-chain of keys (first introduced in sensor networks by Perrig et al. [17]). At each following time interval a new key from the key chain is used as the current key. Our asynchronous group key distribution scheme achieves the same properties without requiring any form of time synchronization.

5.2 Secure Group Communication

TinySec [10] provides a certain level of data confidentiality, authentication, and integrity. For minimal communication overhead, it provides the semantic security property by using a nonce created from its link-layer package header. This keeps the nonce short which decreases security if it repeats; making a requirement for frequent key updates. It does not support protection against replay attacks and data freshness. The authors argue that this is more efficiently handled by an upper-layer that has information about the network topology.

The MiniSec-U [15] provides data confidentiality, authentication, integrity, data freshness, and protection against replay attack. It uses implicit synchronization of a large internal stored nonce which keeps the communication overhead down while achieving a high level of security. The synchronization is done by sending the last few bits of the nonce with each package; these bits can then be used to synchronize an internally stored nonce. Due to the memory limitation of today's sensor nodes [18] it is not applicable to store a nonce for every other sensor node in an arbitrary group, so Luk et al. [15] also proposed MiniSec-B that uses a sliding-window and a bloom-filter based approach to perform memory effi-

cient protection against replay attacks when group communication is used. This requires a certain level of time synchronization and has a chance of rejecting new packets never seen before.

Both TinySec and MiniSec uses software implementations of the SkipJack [4] and the RC5 [20] block cipher as basis for their security primitives. We take advantage of the widely used CC2420 radio [1] security features and a simpler group model to achieve efficient secure group communication with protection against replay attacks without rejecting any packets not seen before.

5.3 CC2420 Security Mechanism

Healy et al. [7] has explored the stand-alone AES encryption mechanism of the CC2420 radio [1] and compared it to similar software implementations optimized for speed and memory. They show that the hardware implementation are orders of magnitude faster and more memory efficient than the similar software implementations, when the read and write to the CC2420 radios RAM are not included as part of the encryption operation. We argue that in order for the stand-alone AES encryption on the CC2420 radio to work as a block cipher at least one read or write is required per. block cipher operation. We have shown in this paper that a read/write operation takes more than twice the time of the actual encryption on the TelosB platform [18] using TinyOS 2.x [13], and hence it can not be neglected in a performance evaluation.

6. CONCLUSIONS AND FUTURE WORK

Sensor networks are often deployed in hostile environment creating a demand for secure communication. Different communication patterns require different security schemes. Based on a simple key update strategy at the base station, we showed that our proposed asynchronous key distribution scheme was able to follow varying traffic rates in a network, and hereby decrease the overhead of key updates compared to previous synchronous schemes [16]. We showed that due to slow RAM read and write operations, the CC2420 stand-alone AES encryption working as a block cipher is not beneficial in terms of speed compared to similar software implementation. This contradicts the conclusions made by Healy et al. [7] where they did not include RAM read and write in their evaluation. Instead, we found that the CC2420 radios in-line security mechanism computationally outperforms a similar software implementation and proposed a way to use this as a basis for securing group communication at the link layer, while keeping the communication overhead very low.

Future work includes a practical energy measurement study of the security mechanism used in this paper. This will show how our performance comparisons relate to the actual energy consumption, and may reveal that it is not energy efficient to use the CC2420 security mechanism if it is possible to turn off the radio instead. Our group key distribution scheme does not provide backward secrecy (previous keys can easily be disclosed from a current key). This is a general property of schemes using key chains [16, 17], and alternative approaches may be explored to cover the scenarios where backward secrecy is a requirement.

7. ACKNOWLEDGEMENTS

This work is partly sponsored by the SensoByg Project [3]. A special thanks goes to Lars Michael Kristensen, Jacob Andersen, Jesper Buus Nielsen, and Margaret Durham for proof reading, insightful comments, and valueable discussion.

8. REFERENCES

[1] Datasheet for chipcon (ti) cc2420 2.4 ghz ieee 802.15.4/zigbee rf transceiver. http://ti.com.

[2] Ieee std. 802.15.4 - 2003: Wireless medium access control (mac) and physical layer (phy) specifications for low rate wireless personal area networks (lr-wpans).

[3] The sensobyg project. http://sensobyg.dk/english.

[4] Skipjack and kea specifications. In *Federal Information Processing Standards 185*. National Institute of Standards and Technology, 1998.

[5] J. Daemen and V. Rijmen. Aes proposal: Rijndael. 1999.

[6] O. Gnawali, K.-Y. Jang, J. Paek, M. Vieira, R. Govindan, B. Greenstein, A. Joki, D. Estrin, and E. Kohler. The tenet architecture for tiered sensor networks. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 153–166, New York, NY, USA, 2006. ACM.

[7] M. Healy, T. Newe, and E. Lewis. Analysis of hardware encryption versus software encryption on wireless sensor network motes. pages 3–15, 2008.

[8] Y. Jiang, C. Lin, M. Shi, and X. Shen. Self-healing group key distribution with time-limited node revocation for wireless sensor networks. *Ad Hoc Networks*, 5(1):14–23, 2007.

[9] W.-S. Juang. Efficient user authentication and key agreement in wireless sensor networks. In *WISA*, pages 15–29, 2006.

[10] 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.

[11] C. Kuo, M. Luk, R. Negi, and A. Perrig. Message-in-a-bottle: user-friendly and secure key deployment for sensor nodes. In *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 233–246, New York, NY, USA, 2007. ACM.

[12] Y. W. Law, J. Doumen, and P. Hartel. Survey and benchmark of block ciphers for wireless sensor networks. *ACM Trans. Sen. Netw.*, 2(1):65–93, 2006.

[13] P. Levis, D. Gay, V. Handziski, J.-H. Hauer, B. Greenstein, M. Turon, J. Hui, K. Klues, C. Sharp, R. Szewczyk, J. Polastre, P. Buonadonna, L. Nachman, G. Tolle, D. Culler, and A. Wolisz. T2: A second generation os for embedded sensor networks. Technical report, 2005.

[14] P. Levis, N. Lee, M. Welsh, and D. Culler. Tossim: Accurate and scalable simulation of entire tinyos applications. In *Proceedings of the 1st international conference on Embedded networked sensor systems (SenSys '03)*. ACM, 2003.

[15] 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.

[16] T. Park and K. G. Shin. Lisp: A lightweight security protocol for wireless sensor networks. *Trans. on Embedded Computing Sys.*, 3(3):634–660, 2004.

[17] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler. Spins: security protocols for sensor networks. *Wirel. Netw.*, 8(5):521–534, 2002.

[18] J. Polastre, R. Szewczyk, and D. Culler. Telos: enabling ultra-low power wireless research. In *IPSN '05: Proceedings of the 4th international symposium on Information*

processing in sensor networks, page 48, Piscataway, NJ, USA, 2005. IEEE Press.

[19] B. Przydatek, D. Song, and A. Perrig. Sia: secure information aggregation in sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 255–265, New York, NY, USA, 2003. ACM.

[20] R. L. Rivest. The rc5 encryption algorithm. In *Fast Software Encryption*, pages 86–96. Springer Berlin / Heidelberg, 1995.

[21] P. Rogaway and D. Wagner. A critique of ccm. Comment to NIST, 2003.

[22] N. Sastry and D. Wagner. Security considerations for ieee 802.15.4 networks. In *WiSe '04: Proceedings of the 3rd ACM workshop on Wireless security*, pages 32–42, New York, NY, USA, 2004. ACM.

[23] M. Shehab, E. Bertino, and A. Ghafoor. Efficient hierarchical key generation and key diffusion for sensor networks. *Sensor and Ad Hoc Communications and Networks, 2005. IEEE SECON 2005. 2005 Second Annual IEEE Communications Society Conference on*, pages 76–84, Sept., 2005.

[24] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong. A macroscope in the redwoods. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 51–63, New York, NY, USA, 2005. ACM.