Efficiently securing data on a wireless sensor network

Efficiently securing data on a wireless sensor network

M Healy[1], T Newe and E Lewis

Optical Fibre Sensors Research Centre, Department of Electronic and Computer Engineering, University of Limerick, Limerick, Ireland

michael.healy@ul.ie

Abstract. Due to the sensitive nature of the data many wireless sensor networks are tasked to collect security of this data is an important concern. The best way to secure this data is to encrypt it using a secure encryption algorithm before it is transmitted over the air ways. However due to the constrained nature of the resources available on sensor nodes the cost, both in terms of power consumption and speed, of any software based encryption procedure can often out weigh the risks of the transmission being intercepted. We present a solution to reduce this cost of employing encryption by taking advantage of a resource already available on many sensor nodes; this resource being the encryption module available on the Chipcon CC2420 transceiver chip.

1. Introduction

Technological advancements in recent years have enabled the development of tiny, cheap, disposable and self contained battery powered computers, known as sensor nodes or "motes", which can accept input from an attached sensor, process this input and transmit the results wirelessly to some interested device(s). When a number of these nodes work together, conceivably up to hundreds of thousands, a Wireless Sensor Network (WSN) is formed.

These wireless sensor networks have the potential to allow a level of integration between computers and the physical world that, to date, has been impossible. The uses for such networks is almost limitless and include such diverse applications as a counter sniper system for urban warfare [1] tracking the path of a forest fire [2], determining the structural stability of a building after an earthquake [3], or tracking people or objects inside a building [4], etc.

Advances in wireless communication have been a major factor in allowing the development of large networks of sensors. However, as stated in the IEEE 802.15.4 standard specification [5], the wireless connectivity of the sensors is not so much a feature of the sensors but rather an application enabler (unlike the majority of wireless applications currently available). This is the case because wired sensor networks on the required scale would be very costly to build and maintain and also very costly to install, making them impractical.

Despite making such sensor networks possible, the wireless nature of the sensors presents a number of problems to the developer. Chief among these is the problem of security. Many WSNs are designed to collect data which is sensitive in some way, such as information about a person's health, confidential data about a company's manufacturing process, troop locations on a battlefield, etc. Without adequate security this data can easily be read and/or modified by an attacker. Due to the

[1] To whom any correspondence should be addressed.

wireless nature of the sensor networks detecting and preventing eavesdropping and modification of the data is greatly complicated. Also the constrained nature of resources on the wireless sensor nodes means that security architectures used for traditional wireless networks are not viable.

As power supply and RAM are normally the most constrained resources on a node, a memory and power efficient cryptographic algorithm is a very important part of any proposed security architecture for a WSN. Traditionally RC5 [6] and Skipjack [7] are considered to be the most suitable algorithms for WSNs [8, 9] due to their speed and memory usage but as shown by Law et al [10] AES [11] or MISTY1 [12] can be more suitable, depending on the situation.

A solution to reduce power and memory costs, as well as to greatly increase speed is to use a hardware implementation of the block cipher being used. As AES is currently the most widely used secure block cipher there is a larger range of commercially available implementations than any other algorithm. However none of the current range of wireless sensor nodes has such a chip already included and adding one can prove difficult and very time consuming, especially for large deployments.

Another solution is available. As can be seen by table 1 the majority of the currently available sensor nodes, both those commercially available and those created by research institutions, use the same IEEE 802.15.4 compliant transceiver, the Chipcon CC2420 [13]. The CC2420 features a number of security operations, including AES encryption. We test the performance of using this hardware encryption on the CC2420 compared with a software implementation of the AES algorithm and present our results here.

2. Background

In this section we describe our target hardware, the operating system we use and explain why AES is an appropriate choice for use in wireless sensor networks.

2.1. Sensor Node Hardware

The sensor nodes that work together to form a WSN are composed of four sub-systems; a computing sub-system, a communication sub-system, a power subsystem and a sensing sub-system.

The computing sub-system consists of a processor and memory, which includes program memory, RAM and possibly non-volatile data memory. An important aspect of processors in sensor nodes is different operational modes, usually Active, Idle and Sleep. This is important so as to preserve power as much as possible without impeding the operation of the processor when it is required.

The communication sub-system is required to enable the sensor nodes to communicate with each other and with a base station. Generally the communication sub-system is a short range radio but the use of infrared communication, ultrasound and inductive fields has also been explored. Most currently available sensor nodes use a radio chip which conforms to the IEEE 802.15.4 standard, but as can be seen from table 1 some nodes use Bluetooth as an alternative.

The power sub-system consists of a battery which supplies power to the sensor node. Due to the long term unattended operation of nodes in a WSN the developer must ensure every aspect of the network such as communication algorithms, localization algorithms, sensing devices, etc., must be as efficient as possible in their power usage. A power generator may also be included to recharge the battery onsite. Photovoltaic, motion/vibration and thermoelectric energy conversion are all possible sources of power, depending on the location of the node [14].

Sensor transducers translate physical phenomena to electrical signals. Therefore the sensing sub-system of the node is its link to the outside world. Some nodes have sensors built in, but many do not, instead providing suitable ports allowing a variety of sensors to be attached for more versatility.

2.1.1. Chipcon CC2420 [13]. As already mentioned the CC2420 transceiver chip from Chipcon is currently the most popular radio chip on wireless sensor nodes. The CC2420 works in the 2.4 Ghz band, is IEEE 802.15.4 compliant, ZigBee ready, low cost, and designed for low-voltage and low-power wireless applications. These features, along with a host of other features make the CC2420

Table 1 Selection of currently available wireless sensor nodes

Platform	MCU	RAM	Program Memory	Non-volatile data memory	Radio Chip
BTnode3	ATMega 128	64 KB	128 KB	180 KB	CC1000/ ZV4002 Bluetooth
Cricket	ATMega 128	4 KB	128 KB	512 KB	CC100
Imote2	Intel PXA271	256 KB	32 MB	0	CC2420
MICA2	ATMega 128	4 KB	128 KB	512 KB	CC1000
MICAz	ATMega 128	4 KB	128 KB	512 KB	CC2420
Shimmer	TI MSP 430	10 KB	48 KB	Up to 2 GB	CC2420 and WML-C46 Bluetooth
TelosA	TI MSP 430	2 KB	60 KB	512 KB	CC2420
TelosB	TI MSP 430	10 KB	48 KB	1 MB	CC2420
Tmote Sky	TI MSP 430	10 KB	48 KB	1 MB	CC2420
XYZ	ARM 7	32KB	256 KB	256 KB	CC2420

ideal for use on sensor nodes.

The CC2420 feature that we are most interested in is its hardware security operations. The CC2420 is capable of performing IEEE 802.15.4 MAC security operations, including counter (CTR) mode encryption and decryption, CBC-MAC authentication and CCM encryption plus authentication. The CC2420 also offers plain stand-alone encryption of 128 bit blocks. Each of these security functions are based on AES encryption using 128 bit keys.

One limitation of the CC2420 security wise is that it does not offer AES decryption. However the impact of this limitation is negated by the fact that for many applications any decryptions are performed by the base station which can be a much more powerful device and/or has less limitation on its power supply. Another option is to use a block cipher mode of operation which does not require a decryption function, such as Cipher Feed Back (CFB) mode, Output Feed Back (OFB) mode, or Counter (CTR) mode. A point to note is that the successor to the CC2420, the currently unreleased Chipcon CC2430 [15] does offer AES decryption, and as soon as this chip is released it is likely to start replacing the CC2420 on new sensor node designs.

2.2. TinyOS

As can be seen from table 1 the specifications of the target hardware for our system are varied. The variety of processors, radio chips as well as the varying amount of RAM and program memory means the software written for each node would need to be significantly different. However a number of operating systems already exist to solve this problem. We chose to use the TinyOS [16] operating system as our development platform, mainly because it currently supports the largest range of hardware and has good power management [17].

TinyOS was the first operating system specifically designed for WSNs and as a result it still has the largest user base and is the standard by which the other operating systems are judged. TinyOS was initially developed at the University of California, Berkeley but is now being developed by a consortium and is open source, making it easy for developers to customize it as required.

TinyOS is not an operating system in the traditional sense; rather it is a programming framework which contains a set of components that allows an application specific OS to be constructed for each particular need, consisting of selected system components and custom components. The principal goal of this component based architecture is to allow application designers to build components that can easily work together to produce a complete, concurrent system but still performs extensive compile time checks.

2.3. AES/Rijndael

Rijndael [11] was chosen as the Advanced Encryption Standard (AES) by the National Institute of Standards and Technology of the United States in 2001 to replace the existing Data Encryption Standard (DES). It is also one of the ciphers recommended for use by Japans CRYPTREC and by the New European Schemes for Signature, Integrity and Encryption (NESSIE) consortium.

Rijndael is designed to work with three different key lengths of 128 bits, 192 bits or 256 bits depending on security requirements. A key length of 128 bits is considered more than sufficient for

Table 2 Program memory and RAM usage (bytes)

	MICAz		Tmote SKY	
	ROM	**RAM**	**ROM**	**RAM**
CC2420_encrypt	10058	437	11872	407
software_encrypt_ref	11980	2844	13206	2811
software_encrypt	12720	1915	13980	1883
software_encrypt_minRAM	12748	625	14200	887

Table 3 Time to generate lookup tables

MICAz	**Tmote SKY**
7.88137 ms	13.29879 ms

securing wireless sensor networks because the length of time any collected data needs to remain confidential will have long since expired before an exhaustive key search on this length key can be performed using currently available technology. However for increased security, at a cost of memory and speed, one of the longer key lengths can also easily be employed.

One of the criteria used to choose AES was the cipher's efficiency and performance on a variety of platforms, from 8-bit smart cards to high end processors. For this reason Rijndael should be well suited for use on wireless sensor nodes, despite their limited processing power.

Rijndael has come under intense scrutiny both during the review process before it was chosen as AES and also since then. As of April 2007 no feasible attacks against the algorithm have been published and so it is still considered secure.

3. Implementation

We implemented an application in TinyOS which employs the CC2420 hardware encryption features in order to encrypt 16 bytes of data and then send this data to a listening base station.

We implemented this application for two popular sensor node platforms, Crossbow's MICAz and Moteiv's Tmote SKY. The only difference between the two versions of the application is that for the MICAz version we had to implement the function to read from the RAM on the CC2420 chip ourselves whereas for the Tmote SKY this function was already made available by TinyOS.

Using the stand alone encryption of the CC2420 is relatively straight forward. First off the two Security Control registers, SECCTRL0 and SECCTRL1, which control the security operations of the CC2420, need to be set. As we are using key 0 as our stand-alone key we can set both of these registers to 0. We then write the 128 bit key and the plaintext to the appropriate RAM locations on the CC2420; from location 0x100 and 0x120 respectively. The SAES command strobe is then sent in order to begin the encryption operation. We can tell when the encryption operation is completed by reading the ENC_BUSY status bit, which tells us whether the encryption module is busy or not. We read the CC2420's status bits by sending the SNOP command strobe. The final step is to read back the cipher text. As the encryption module overwrites the plaintext with the ciphertext this is simply a case of reading 16 bytes from the location the plaintext was originally written to, i.e. from location 0x120. It is worth noting that the CC2420 RAM write operation also outputs the data currently at the location being written to, so that a new plaintext can be written at the same time as reading out the previous ciphertext in order to streamline encrypting data longer than 128 bits.

It is also worth noting that when using the other security operations supported by the CC2420, i.e. the IEEE 802.15.4 MAC security modes, the encryption happens after the message is buffered in the TXFIFO buffer, and the decryption happens before the message is read out of the RXFIFO buffer. This means that the encrypted message does not have to be read from the CC2420 at any stage, which would only have to be sent back to the transceiver for another operation anyway, so using these modes is even simpler and more transparent for the application developer than using the stand-alone encryption mode.

4. Results

In the discussion and tables below the TinyOS implementation of using the CC2420 chip to perform AES encryption will be referred to CC2420_encrypt. In order to test the performance of using the CC2420 to do the encryption we needed something to compare it to. Towards this end we created some software implementations of Rijndael for the sensor nodes. Our first implementation, software_encrypt_ref, is as simple an implementation as possible based on the Rijndael AES proposal

Table 4 Time to run setup procedure

	MICAz	Tmote SKY
CC2420_encrypt	294.003 µs	2.06034 ms
software_encrypt_ref	652.15 µs	940.83 µs
software_encrypt	565.37 µs	752.27 µs
software_encrypt_minRAM	574.85 µs	934.83 µs

Table 5 Time to encrypt 16 bytes

	MICAz	Tmote SKY
CC2420_encrypt	29.8310 µs	449.203 µs
software_encrypt_ref	1.49578 ms	1.94108 ms
software_encrypt	1.45753 ms	1.87064 ms
software_encrypt_minRAM	1.54772 ms	1.94374 ms

[11]. This version has no optimization and all the lookup tables are generated at runtime when the program initializes. We made some simple optimizations to this code, including using pre-computed lookup tables, to produce our second version: software_encrypt. As the RAM usage of these two versions of the code was high we created another version which stored the required lookup tables in program memory. We will refer to this version as software_encrypt_minRAM. None of these software implementations of AES are as efficient as they could possibly be, but their run times and memory usage is within a few per cent of what a highly optimized version is likely to be.

Table 2 shows the amount of program memory and RAM each implementation uses on each platform. As can be clearly seen the CC2420_encrypt version is the most efficient in terms of RAM usage. This version also uses less program memory than the other three versions, the difference not seeming as significant because the base TinyOS code is still the majority user of the code space. This table also shows that, for the most part, the Tmote SKY makes more efficient use of RAM than the MICAz, but is less efficient when it comes to program memory. Ideally this situation should be reversed because, as can be seen from table 1, the Tmote SKY has significantly more RAM available to it that the MICAz and significantly less program memory.

The software_encrypt_ref version generates the lookup tables used in order to save program memory. This technique can also be used to save RAM. However as we saved these generated tables in static global arrays, they only needed to be generated once so we did not reap the RAM saving benefits. The time it takes to generate these tables is shown in table 3. In order to create a software implementation of AES which optimizes both RAM and program memory usage these tables would need to be generated every time they were required.

Tables 4 and 5 show the amount of time it takes to execute various parts of code on the two platforms. Table 4 shows the amount of time each implementation takes running the setup procedure before encryption can commence. For the CC2420_encrypt version this involves setting the security registers and loading the key and plaintext into the CC2420's RAM and the time taken to read the resulting ciphertext back is also included. For the other three versions the setup procedure involves expanding the key being used into each round key. Table 5 shows the amount of time it takes to actually perform the encryption routine on 16 bytes of data. As expected performing the encryption is much faster for the CC2420_encrypt version. However the Tmote SKY is nearly 15 times slower than the MICAz. Also, as seen in table 4, CC2420_encrypt's setup procedure on the Tmote SKY is very slow compared to that of the MICAz's. These timing differences cannot be adequately explained by the processor and clock differences between the two platforms and so we suspect that the difference is due to TinyOS's platform specific routines that read from and write to the RAM on the CC2420.

Tables 3 and 4 also show that the RAM optimized software implementations are slower than the unoptimized versions on both platforms. This increase in processing time is due to the extra overhead of reading from the lookup tables placed in program memory over that of reading the same table in RAM.

5. Conclusions

We have shown that using hardware encryption instead of software based encryption is beneficial in terms of speed and memory usage for wireless sensor networks. In particular we have shown that using a resource already existing on many wireless sensor nodes, the Chipcon CC2420 transceiver chip, can significantly reduce the cost of securing data on a sensor network.

Acknowledgments
The authors wish to thank the following for their financial support:

- SFI Research Frontiers Programme grant number 05/RFP/CMS0071
- The Embark Initiative and Intel, who fund this research through the Irish Research Council for Science, Engineering and Technology (IRCSET) postgraduate Research Scholarship Scheme.

References

[1] Lédeczi Á, Nádas A, Völgyesi P, Balogh G, Kusy B, Sallai J, Pap G, Dóra S, Molnár K, Maróti M and Simon G 2005 Countersniper system for urban warfare *ACM Trans. on Sensor Networks* **1**(2) 157-177

[2] Fok C-L, Roman G-C and Lu C 2005 Mobile agent middleware for sensor networks: an application case study *Proc. 4th Int. Conf. on Information Processing in Sensor Networks* (Los Angeles, California, USA, 25-27 April 2005) (IEEE) pp 382-387

[3] Schmid T, Dubois-Ferrière H and Vetterli M 2005 SensorScope: Experiences with a wireless building monitoring sensor network *Proc Workshop on Real-World Wireless Sensor Networks* (Stockholm, Sweden, June 2005)

[4] Want R, Hopper A, Falcão V and Gibbons J 1992 The active badge location system *ACM Trans. on Information Systems* **10**(1) 91-102

[5] 2003 *802.15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)* (New York: IEEE Standards Association)

[6] Rivest R 1995 The RC5 encryption algorithm *Proc of the 1994 Leuven Workshop on Fast Software Encryption* (Springer-Verlag) pp 86-96

[7] NIST 1998 *Skipjack and KEA Algorithm Specifications Version 2.0* NIST

[8] Karlof C. Sastry N and Wagner D 2004 TinySec: a link layer security architecture for wireless sensor networks *Proc 2nd Int. Conf. on Embedded Networked Sensor Systems* (Baltimore, MD, USA) (ACM Press) pp 162-175

[9] Vitaletti A and Palombizio G 2006 Rijndael for sensor networks: is speed the main issue? *Proc 2nd Workshop on Cryptography for Ad Hoc Networks* (Venice, Italy, July 2006)

[10] Law Y W, Doumen J and Hartel P 2006 Survey and benchmark of block ciphers for wireless sensor networks *ACM Trans. on Sensor Networks* **2**(1) 65-93

[11] Daemen J and Rijmen V 1999 *AES Proposal: Rijndael*

[12] Matsui M 1997 New Block Encryption Algorithm MISTY *Proc. Fast Software Encryption, 4th Int. Workshop* (LNCS vol. 1267) ed Biham E (Springer-Verlag) pp 54-68

[13] Chipcon 2004 CC2420 datasheet [online], available: http://www.chipcon.com/files/CC2420_Data_Sheet_1_3.pdf [accessed 9 Feb 2007]

[14] Roundy S, Wright P K and Rabaey J M 2003 *Energy Scavenging for Wireless Sensor Networks with Special Focus on Vibrations* (Springer)

[15] Chipcon 2007 CC2430 datasheet [online], available: http://www.chipcon.com/files/CC2430_Data_Sheet_rev2p01.pdf [accessed 14 May 2007]

[16] Hill J, Szewczyk R, Woo A, Hollar S, Culler D E and Pister K S J 2000 System architecture directions for networked sensors *Proc 9th Int. Conf. on Architectural Support for Programming Languates and Operating Systems* (Cambridge, MA, USA, November 12-15) (New York: ACM Press) pp 93-104

[17] Healy M, Newe T and Lewis E 2007 Power management in operating systems for wireless sensor nodes *Proc IEEE Sensors Applications Symposium* (San Diego, CA, USA, February 6-8)

[18] Sun K, Ning P and Wang C 2006 TinySeRSync: secure and resilient time synchronization in wireless sensor networks *Proc. 13th ACM Conf. on Computer and Communications Security* (Alexandria, Virginia, USA, Oct 30 - Nov 3)(New York: ACM Press) pp 264-277