Evaluating Energy-Efficiency of Hardware-based Security Mechanisms

Christian Haas, Stephan Munz, Joachim Wilke, Anton Hergenröder
Institute of Telematics
Karlsruhe Institute of Technology, Karlsruhe, Germany
Email: christian.haas@kit.edu, smunz@wareconsult.com, joachim.wilke@kit.edu, anton.hergenroeder@kit.edu

***Abstract*—Security in Wireless Sensor Networks (WSNs) is an omnipresent topic. In many application scenarios, like the surveillance of critical areas or infrastructures, security mechanisms have to be used to build reliable and secure applications. Up to now, most of the used cryptographic algorithms have been implemented in software despite the resource constraints in terms of processing power, memory and energy. In the past few years, the usage of special hardware accelerated security modules has been proposed as a viable alternative to software implementations. However, the energy-efficiency has not yet been evaluated in-depth. In this paper, we analyze the VaultIC420 security module and present an evaluation of its energy-efficiency. We compare the performance and energy-efficiency of the hardware module to common software implementations like TinyECC. For the energy measurements, we use IRIS sensor nodes in the SANDbed testbed at the Karlsruhe Institute of Technology. Our evaluation shows, that the VaultIC420 can save up to 76% of energy using different MAC layer protocols. It also shows, that the current draw of the VaultIC420 requires a duty-cycling mechanisms to achieve any savings compared to the software implementation.**

***Keywords*-Security; Energy-Efficiency; Testbed**

I. Introduction

Over the last several years, Wireless Sensor Networks (WSNs) have been proposed for a vast number of applications. In most scenarios, secure communication and robust operation are among the most important application requirements. The surveillance of critical infrastructures or critical areas like borders or industrial complexes is one of the most named and analyzed scenarios. Here, the need for secure communication and robust operation is widely accepted. Many security protocols and cryptographic mechanisms to that end have been proposed, but they have rarely been implemented or evaluated in realistic applications or environments.

A typical wireless sensor node is a very resource constraint system, e.g. the sensor nodes' energy budget and processing power is very limited. Thus the computational expensive cryptographic mechanisms have to be used with care. This is especially true for the more costly asymmetric cryptographic algorithms. Up to now, most research has been done on efficiently implementing security protocols or mechanisms in software. Recently, some researchers proposed the usage of hardware security modules like a Trusted Platform Modules (TPMs) or similar hardware [1], [6]. One key motivation for using hardware security modules is reducing the energy consumption for the cryptographic operations and therefore improving the energy-efficiency of the overall application. Also, the usage of asymmetric cryptographic algorithms with larger exponents (e.g. RSA) now seems to be feasible when using hardware modules. Unfortunately, the question whether these modules can actually improve the energy-efficiency hasn't been analyzed in-depth. One reason is that the measurement of energy consumption in a sensor network is very challenging. To gain reliable measurements, special purpose measurement hardware has to be used. With such devices, one is capable of performing high resolutive measurements of the energy consumption on real nodes running a real sensor network application.

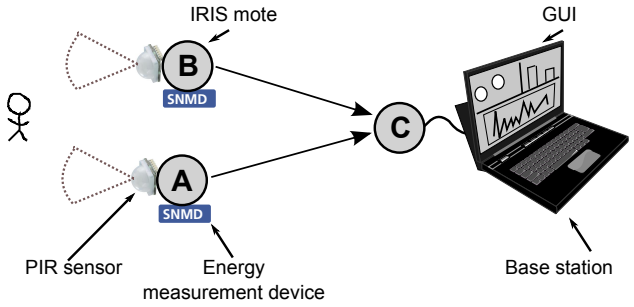

Figure 1. Monitoring scenario

In this work, we will evaluate the energy-efficiency of hardware-based security mechanisms within a monitoring scenario. For our experiments, we use a monitoring application already shown as a demo in [6]. It is capable of using cryptographic mechanisms implemented in software and in hardware. In the demonstrator, we presented the general approach for measuring and comparing the energy consumption of ECDSA within a monitoring scenario. In our demo, we used the SANDbed testbed to gain reliable and accurate measurements of the energy consumption. In SANDbed, each sensor node is equipped with a *Sensor Node Management Device* (SNMDs) [3].

The basic monitoring scenario is shown in Figure 1: Two IRIS sensor nodes are equipped with a passive infrared sensor (PIR) to detect motion. Both nodes are each connected to a SNMD. If motion is detected, a PIR-event message is cryptographically secured and sent to the base station (node C).

We will show, that hardware-based mechanisms can in-

978-1-4244-9529-0/13/$31.00 ©2013 IEEE

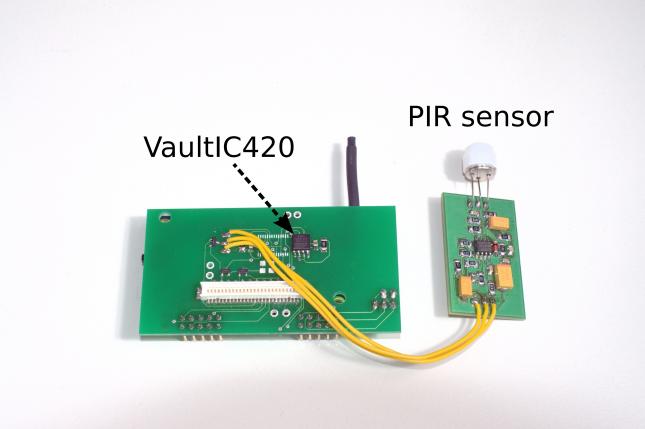

Figure 2. New sensor board with VaultIC420 and PIR sensor

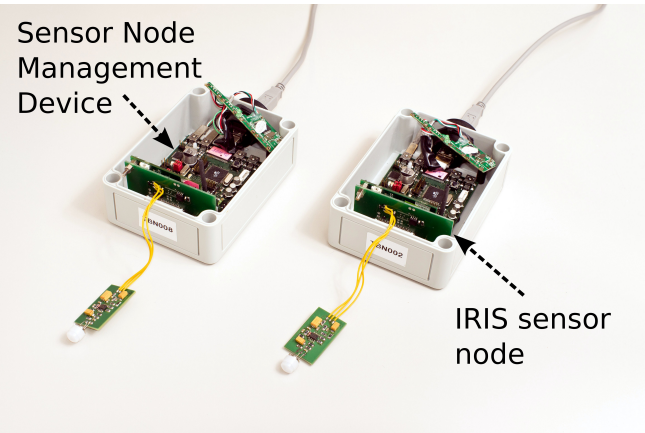

Figure 3. SANDbed Hardware used in the demonstrator and in our evaluation

deed improve the energy-efficiency of the overall application but only for certain algorithms and if the hardware modules are duty-cycled.

II. Related work

Up to now, most research has been done on efficiently implementing well known security protocols or mechanisms in software. Yet, there are very few publicly available implementation that have been evaluated with regard to energy-efficiency. Examples are the TinyECC library for elliptic curve cryptography [2] or the RELIC library [13]. Nevertheless it is widely accepted, that improving the energy-efficiency of cryptographic algorithms will make the implementation of secure applications, like monitoring applications for critical infrastructures, more feasible [5].

Some security protocols like the Rich Uncle protocol [12] are built upon the requirement for nodes with higher processing power or unlimited energy resources. More recent work has proposed the usage of hardware-based security mechanisms like a trusted platform module (TPM) or similar hardware [1] for that purpose. With these hardware modules, commonly known and used algorithms like RSA can now be used more efficiently on typical sensor nodes. One further motivation for using hardware-based security modules is limiting the energy consumption for the cryptographic operations. The authors implemented a TPM hardware module within their sensor node platform. They argue, that the hardware-based security module makes public key cryptography with RSA feasible compared to a software only implementation. However, within their work they only computed the overall energy consumption of the hardware module based upon previous measurements with an oscilloscope. While this approach can be used to quickly get some energy usage data the approach is also error prone and not very accurate. This is especially true when more complex applications have to be compared. While the authors claim that the costs for using the TPM module are quite high, they don't present a duty-cycling of the TPM module. The overhead for powering the TPM up or down while preserving the TPM's state and key material is not presented.

Based upon this approach with TPM modules, there are several implementations of different security protocols. Krauß et al. implemented a secure code update protocol with attestation, T-CUP [11]. Yet, they didn't perform an evaluation of the actual energy consumption. Another approach for attestation with a TPM module was published in [10]. Here, the authors evaluate the overall energy consumption of the protocol by using the execution and the current draw of different operation states of the TPM. Again it is not stated how the measurement of the current draw was performed, duty-cycling of the TPM is not considered. Kothmayr et al. presented a study on the usage of a TPM module for implementing DTLS over 6LoWPAN [9]. They showed, that they could successfully implement a standard conform DTLS handshake. There is no comparison of the energy-efficiency with a software implementation.

III. Evaluation tools and scenario

Our evaluation scenario can be seen in Figure 1. For our evaluation, we use the implementation of a small monitoring system for critical areas that is able to detect any movement in the critical area as described in [6]. We use two IRIS sensor nodes (nodes A and B) with passive infrared sensors (PIR). Whenever one of the PIR sensor detects some motion, a so called PIR-event is triggered which is then transmitted to a basestation (node C). All of the nodes have an attached hardware security module which can be activated and deactivated depending on the evaluation scenario. As a hardware security module, we use the VaultIC420 from INSIDE Secure which is also attached to the extension board that can be seen in Figure 2. To perform energy measurements, we employ two *Sensor Node Management Devices* (SNMDs), see Figure 3. Our application is written in TinyOS, a special operating system for WSNs. During our evaluation we use different MAC-layer protocols, IEEE

802.15.4 as implemented in TinyOS and TinyOS Low Power Listening [8] as the de facto standard duty-cycling MAC-layer protocol. In the demonstrator, we presented the general evaluation scenario for comparing the energy consumption of different protocols. As an example, we compared the energy consumption of ECDSA signature generation and verification in a monitoring scenario. In this work, we are analyzing the energy consumption of different cryptographic algorithms and services like encryption, message digests and digital signatures. Moreover, we are also analyzing the additional overhead for duty-cycling the hardware module as well as a more detailed comparison to software based implementations.

A. Energy Measurement

For a distributed measurement of energy we use Sensor Node Management Devices (SNMD) [3] which are deployed at the SANDbed testbed at the Karlsruhe Institute of Technology. Attached to each SNMD is a standard IRIS sensor node and a sensor board with the VaultIC420 3. SNMDs are capable of high resolutive voltage and current measurements. The SNMD provides sampling frequencies of up to 500kHz with an average measurement error below 1% [7]. Measurements are sent to *Management Nodes* over USB to store the data for a later evaluation. No instrumentation of the application code is necessary and measurements do not interfere with the sensor node's normal mode of operation. The measurement samples taken with distinct SNMDs in SANDbed are not time synchronized, but the sensor nodes and therefore the measurements can be started simultaneously. Measurements in SANDbed are thus side effect free with regard to communication overhead and operation of the measured application.

Furthermore, the SNMDs enable the sensor nodes to control the measurements. Therefore the SNMD passively probes three specific pins of the expansion interface on the IRIS nodes. These pins are used by the IRIS platform to signal the state of the sensor node with LEDs. The SNMD can then detect and log any change on this pins and accordingly all state changes of the three LEDs. Consequently the sensor node can signal its state to the SNMD. The SNMD is currently able to react on this signals with three actions: Start and stop of a measurement or just logging the event as a marker in the measurement log. The start and stop actions enable measuring the energy consumption of just some certain section of the application, which is interesting for the evaluation. On the other hand, setting markers in measurement logs enables by far more detailed insights in the application flow. For example, the energy consumption of different parts of the application can be evaluated after the experiment in retrospect. Signaling the node state in this way is almost side-effect free because the SNMD detects the pin changes very quickly, thus the LED have to be powered on for approx. $12\mu s$ to be detected as a signal. This causes a very small energy consumption overhead of about $1.2\mu J$ which does not affect the measurement result and can only be measured with sampling rates above $90kHz$.

In our evaluation the markers are used to mark the two most significant sections in the application. These are the calculation of the cryptographic algorithms and the transmission of the message. This technique enables us to analyze the energy consumption of every single cryptographic calculation and more important to distinguish between the energy consumed for cryptographic calculations and wireless communication.

B. Cryptographic Algorithms

In our evaluation we use and compare different cryptographic algorithms. The VaultIC420 has the capability to compute a wide range of symmetric and asymmetric cryptographic algorithms, as well as different hash functions. As symmetric algorithms, the VaultIC420 supports the DES/3DES standard and AES up to a key size of 256 bit. Moreover, the hardware module supports asymmetric cryptography with RSA, DSA and ECC with multiple key sizes. As cryptographic services, the VaultIC420 offers Public key pair generation, digital signatures, encryption/decryption, message digests, key wrapping/unwrapping, HOTP one-time password generation and a true random number generator. The VaultIC420 also offers some software features, like a secure file system, rights management (e.g., administrator, approved user, Non-approved User) and a FIPS 140-2 identity-based authentication using pass-word (Secure Channel Protocol v2 and v3). The VaultIC420 is connected to the IRIS node via the I^2C bus (two wire interface) with INSIDE's proprietary communication protocol. We therefore implemented a standard conform I^2C driver in TinyOS as well as INSIDE's communication protocol. The I^2C bus has a line speed of 100 kbit/s.

In our evaluation, we will focus on the most popular algorithms that have been proposed for WSNs, AES encryption/decryption, message digests with HMAC-SHA1 and digital signatures with ECDSA. For all these cryptographic algorithms and services, we incorporated software implementations in our application. For AES, we use the AES implementation that can be found in the TinyOS contrib repository. For the computation of SHA1 and TinyECC, we used TinyECC [2]. We implemented HMAC on top of the TinyECC SHA1 implementation. While we are aware that these implementations are not the most optimized implementations, all the code is publicly available and therefore often used in many applications or publications.

IV. Evaluation

In our evaluation, we are first comparing the energy consumption of the different cryptographic algorithms directly. In a second step, we are going to analyze the energy-efficiency of the VaultIC420 in a more realistic scenario.

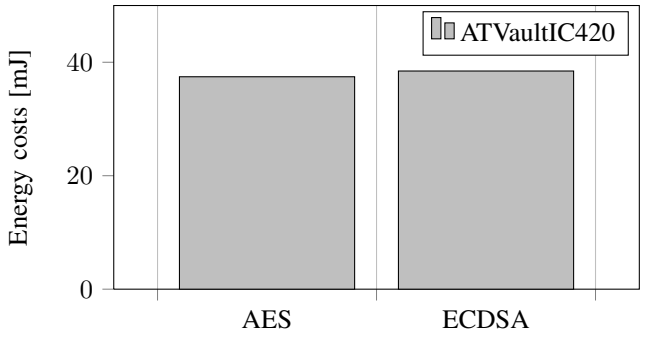

Figure 4. Initialization costs of the VaultIC420

In our evaluation scenario, we are also analyzing the costs for transmitting the PIR-event as well as the costs for duty-cycling the hardware. As MAC-layer protocols, we use IEEE 802.15.4 as well as TinyOS Low Power Listening (LPL).

Before the VaultIC420 can be used, it has to be enabled and initialized. In our tests, we used a guard time of 500ms after the start of the initialization before we actually used the VaultIC420 for any cryptographic computation. While the value of 500ms might be reduced after optimizing the TinyOS I^2C driver, it guaranteed a stable and reliable usage of the hardware in our experiment.

Figure 4 shows the initialization costs for AES and ECDSA. For AES, the initialization costs are 37.44mJ, for ECDSA slightly higher with 38.45mJ. In our evaluation, all the symmetric algorithms as well as the algorithms based on hash functions showed the same initialization costs as AES. In the following comparison of the cryptographic algorithms, these initialization costs are not included, we just compare the energy consumption for the computation of the digests. For all the comparisons, we used two test vectors of 16 Byte and 96 Byte. For all tests, we verified that both software and hardware digest were actually equal and therefore the algorithms were working correctly.

The first algorithms we compared was AES. We compared the energy costs of encryption and decryption of the two test vectors, see Figure 5. In our tests, the software based encryption and decryption was way more energy-efficient than the hardware-based encryption and decryption. For the 16 Byte test vector, the software implementation consumed around 10 times less energy(1.267mJ vs. 0.123mJ) for encryption and 9 times less energy for decryption (1.266mJ vs. 0.142mJ). For the 96 Byte test vector, the software based implementation consumed 6 times less energy for encryption and 4 times less energy for decryption respectively.

Next, we compared the energy consumption of SHA1 and the HMAC-SHA1 message digests. Again, we used the same test vectors as in the previous comparison. Figure 6 shows the energy consumption the computation of the SHA1 message digests. Again, the software based implementation was more energy efficient. The software implementation consumed approx. half as much energy as the hardware-based approach, 0.681mJ and 1.183mJ for the 16 Byte vector, 1.363mJ and 2.142mJ for the 96 Byte vector.

The energy consumption of the generation and verification of HMAC-SHA1 message digests is shown in Figure IV. For the first time in our evaluation, the VaultIC420 now outperforms the software implementation in terms of energy consumption. For both test vectors, the software implementation now is almost twice as expensive as the usage of the VaultIC420. In this test case, the additional overhead for duty-cycling the hardware module and transmitting the data over the I^2C bus combined with the relatively short computation outperforms the costs for the computation in software.

As asymmetric cryptographic algorithm, we evaluated the ECDSA algorithm. Again, we used the same two test vectors to compare the energy consumption of signature generation and verification. Figure 8 shows the energy consumption within our measurements. In contrast to the previous measurements, the usage of the VaultIC420 is now the more energy-efficient approach. For both test vectors, the generation of a 192-bit signature is 16 times more costly with the software implementation (e.g. for the 16 Byte vector 13.97mJ vs 212.44mJ). The verification of a signature is only 12 times more costly with TinyECC.

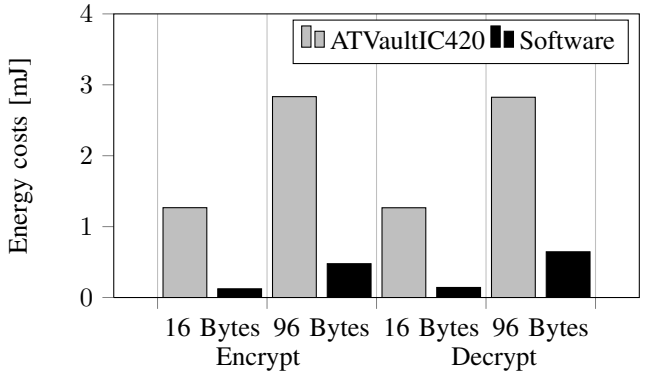

Figure 5. Energy costs of AES encryption/decryption

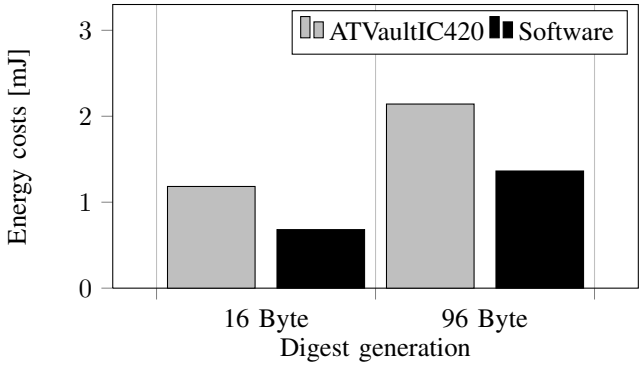

Figure 6. Energy costs of SHA1 digest generation

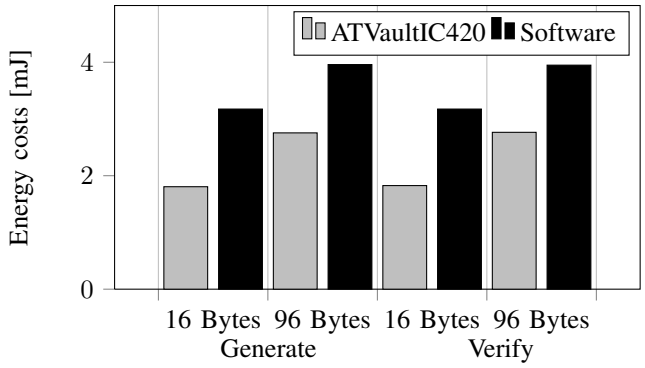

Figure 7. Energy costs of HMAC-SHA1 digest generation/verification

Up to know, we compared the energy consumption of several cryptographic algorithms directly without accounting for the initialization costs. We could show, that both SHA1 as well as AES could be used more energy-efficient with a software implementation. For the VaultIC420, the relatively slow communication over the I^2C interface as well as the additional overhead in TinyOS make the hardware based approach more costly. Due to longer computations within HMAC-SHA1 and ECDSA, the usage of the VaultIC420 now becomes more energy-efficient.

A. Real World Application

So far, we only compared the cryptographic algorithms in isolation. As this approach is not well suited for a realistic evaluation, we now compare both the software implementation and the VaultIC420 in our monitoring scenario. We compared the energy-efficiency of the generation of a message digest/signature of the PIR-event, sending the message to the base station, receiving the message at the base station and then verifying the message signature. We compared both HMAC-SHA1 and ECDSA, as these algorithms seem to be the most used algorithms within e.g. DTLS. As MAC-layer protocols, we performed our evaluation with 802.15.4 and TinyOS Low-Power-Listening.

The results of 15 measurements per configuration are shown in Figure 9. In each run, we measured the energy consumption of the signature/digest generation, the message transmission for the sender and the signature/digest verification, message reception for the base station. After the message was transmitted, we powered the VaultIC420 down. In this test, we also measured the energy consumption of the initialization of the hardware. When using ECDSA signatures, it can be clearly seen that the VaultIC420 was way more energy-efficient than the software implementation with TinyECC for both MAC layer protocols. In fact, the VaultIC is 4 times more energy-efficient (e.g. 827.17mJ vs. 217.33mJ with 802.15.4). Obviously, the energy consumption with TinyOS LPL is lower than with plain 802.15.5, as the radio transmitter is powered off if not used. The results for HMAC-SHA1 are quite different. Here, the software implementation is still more energy-efficient than the VaultIC420. With 802.15.4, the software implementation is 5 times more energy-efficient, with LPL 12 times. This is quite surprising, as in the direct comparison earlier HMAC-SHA1 was cheaper in hardware. We found that this effect was caused due to the additional duty-cycling of the VaultIC420, the chip was active for a longer time and only shut down after the message was transmitted successfully. As the VaultIC420 has a current draw of approx. 21mA, this caused additional energy costs.

B. Memory Usage

To conclude our evaluation, table I shows the memory usage after compiling the monitoring application with our software and hardware implementation. The VaultIC420 uses 16970 Byte of ROM and 2177 Byte of RAM. The software implementation is almost 3 times bigger with regard to both RAM and ROM. Note that an IRIS node has only 8 kByte of RAM, but additionally to the static memory of our application the stack also has to fit within the 8 kByte of RAM (e.g. for TinyECC up to 800 Byte). It can be clearly seen that with the hardware implementation it would be possible to implement a bigger application whereas with with the software implementation one would have to be quite

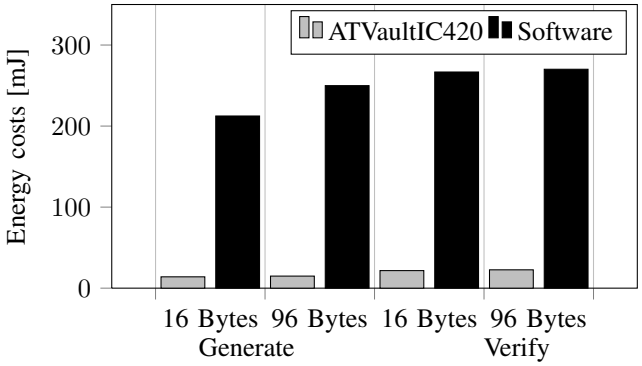

Figure 8. Energy costs of ECDSA signature generation/verification

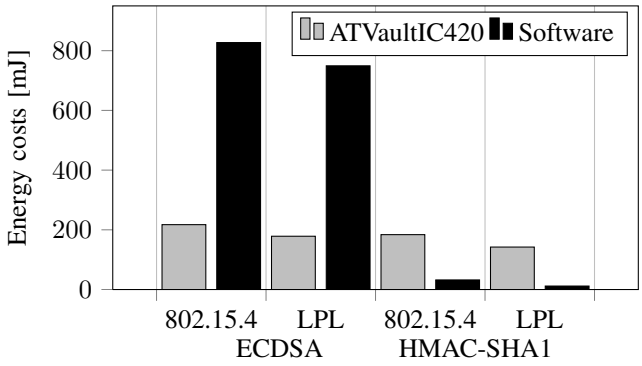

Figure 9. Energy costs of ECDSA signatures and HMAC-SHA1 digests

Implementation	ROM	RAM
VaultIC420	16970 Byte	2177 Byte
Software	43426 Byte	5917 Byte

Table I
MEMORY USAGE OF THE SOFTWARE AND HARDWARE IMPLEMENTATION

careful with the RAM usage.

C. Discussion

In our evaluation, we compared the energy consumption of different cryptographic algorithms provided by the VaultIC420 and by publicly available software implementations. We could show, that the VaultIC420 can in fact improve the overall energy-efficiency of an application. But in contrast to earlier expectations, this is only the case for when the computations in software are long enough to compensate for the relatively high energy costs for the VaultIC420 and its duty-cycling. Without duty-cycling, the VaultIC420 is not applicable in real-world applications due to its relatively high current draw of 21mA. As a conclusion, one has to be careful about choosing a hardware module when designing a secure application in WSNs. The very slow I^2C bus and the very high initialization time combined with the relatively high current draw of the hardware module have to be taken into account. Also, the used MAC protocol and the communication pattern has to be chosen very carefully. For symmetric key operations, it might be more feasible to use a software based approach if there is enough memory on the wireless sensor node. For the hardware modules, it might also be important to use or analyze possible power saving mechanisms, especially if an application uses a lot of periodic traffic and a little higher latency is not a problem.

V. SUMMARY

Up to our knowledge, we are the first to evaluate the energy-efficiency of hardware based security mechanisms in a real-world scenario. We showed how these hardware modules can save energy, especially when using asymmetric cryptographic operations. With the VaultIC420 it is possible to save up to 76% of energy compared to a software based implementation when using ECDSA as signature algorithm. In contrast to earlier expectations, the hardware module is not able to outperform our software implementation for symmetric algorithms like AES or HMAC-SHA1. Here, the overhead for using the I^2C bus as well as the relatively high current draw makes the usage of the VaultIC420 more expensive. We also presented a generic approach for evaluating the energy-efficiency of WSN protocols and algorithms. Our approach can easily be extended any other real-world experiments and evaluations. Future work will include an evaluation using more duty-cycling MAC layer protocols and a study of possible queuing and power saving mechanisms to improve the VaultIC420 duty-cycling.

REFERENCES

[1] Hu, Wen and Corke, Peter and Shih, Wen Chan and Overs, Leslie, *secFleck: A Public Key Technology Platform for Wireless Sensor Networks*, Proceedings of the 6th European Conference on Wireless Sensor Networks (EWSN), 2009.

[2] An Liu and Peng Ning, *TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks*, Proc. of the 7th Conference on Information Processing in Sensor Networks, 2008.

[3] A. Hergenröder and J. Wilke and D. Meier, *Distributed Energy Measurements in WSN Testbeds with a Sensor Node Management Device (SNMD)*, Workshop Proc. of the 23th International Conference on Architecture of Computing Systems, 2010.

[4] ATVaultIC420 security module, http://www.insidesecure.com/.

[5] C. Haas, D. Krüger, D. Pfisterer, S. Fischer, D. Dudek, A. Kuntz, M. Zitterbart and P. Rothenpieler, *FleGSens - secure area monitoring using wireless sensor networks*, Proceedings of the 4th Safety and Security Systems in Europe, 2009

[6] C. Haas, A. Hergenröder, J. Wilke, T. Wiskot and M. Niedermann, *Demo: Evaluating Energy-Efficiency of Hardware-based Security Mechanisms*, 9th European Conference on Wireless Sensor Networks, 2012

[7] A. Hergenröder and J. Horneber, *Facing Challenges in Evaluation of WSN Energy Efficiency with Distributed Energy Measurements*, Multihop Wireless Network Testbeds and Experiments Workshop, IEEE Computer Society, 2011

[8] D. Moss and P. Levis, *BoX-MACs: Exploiting Physical and Link Layer Boundaries in Low-Power Networking*, Computer Systems Laboratory Stanford University, 2008.

[9] T. Kothmayr, W. Hu, C. Schmitt and G. Carle, *Securing the internet of things with DTLS*, In Proceedings of the 9th Conference on Embedded Networked Sensor Systems, 2011.

[10] H. Tan, W. Hu and S. Jha, *A TPM-enabled remote attestation protocol (TRAP) in wireless sensor networks*, In Proceedings of the 6th Workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks, 2011.

[11] S. Wagner, C. Krauß and Claudia Eckert, *T-CUP: A TPM-based Code Update Protocol Enabling Attestations for Sensor Networks*, 7th International ICST Conference on Security and Privacy in Communication Networks (SecureComm), 2011.

[12] D. W. Carman and P. S. Kruus and B. J. Matt, *Constraints and approaches for distributed sensor network security*, NAI Labs, Security Research Division, 2000.

[13] D. F. Aranha and C. P. L. Gouvêa, *RELIC is an Efficient LIbrary for Cryptography*, http://code.google.com/p/relic-toolkit/

[14] S. Pelissier, T.V. Prabhakar, H.S Jamadagni, R. VenkateshaPrasad and I. Niemegeers, *Providing security in energy harvesting sensor networks*, IEEE Consumer Communications and Networking Conference (CCNC), 2011.