A Cross-layer Approach to Trustfulness in the Internet of Things

Antônio Augusto Fröhlich, Alexandre Massayuki Okazaki,
Rodrigo Vieira Steiner, and Peterson Oliveira
Software/Hardware Integration Lab
Federal University of Santa Catarina
PO Box 476, 88040-900 - Florianópolis, SC, Brazil
guto,alexadre,rodrigo,peterson@lisha.ufsc.br

Jean Everson Martina
Computer Security Lab
Federal University of Santa Catarina
PO Box 476, 88040-900 - Florianópolis, SC, Brazil
everson@inf.ufsc.br

***Abstract*—It is a mistake to assume that each embedded object in the Internet of Things will implement a TCP/IP stack similar to those present in contemporary operating systems. Typical requirements of ordinary things, such as low power consumption, small size, and low cost, demand innovative solutions. In this article, we describe the design, implementation, and evaluation of a trustful infrastructure for the Internet of Things based on EPOSMote. The infrastructure was built around EPOS' second generation of motes, which features an ARM processor and an IEEE 802.15.4 radio transceiver. It is presented to end users through a trustful communication protocol stack compatible with TCP/IP. Trustfulness was tackled at MAC level by extending C-MAC, EPOS native MAC protocol, with AES capabilities that were used to encrypt and authenticate IP datagrams packets. Our authentication mechanism encompasses temporal information to protect the network against replay attacks. The prototype implementation was assessed for processing, memory, and energy consumption with positive results.**

***Keywords*-Internet of Things; Cross-layer communication protocols; Trustfulness;**

I. Introduction

The idea of an Internet of Things (IoT) is quickly materializing through the adoption of RFID as a replacement for bar code along with the introduction of Near Field Communication (NFC). We are able to interface with our daily-life objects over the Internet. However, the next steps towards a global network of smart objects will drive us through several large-scale, interdisciplinary efforts. In particular, security and privacy are issues that must be consistently addressed before IoT can make its way into people's lives.

Things in IoT interact with each other and with human beings through a myriad of communication technologies, often wirelessly, and subject to interference, corruption, eavesdropping, and all kinds of attacks. Most of encryption and authentication techniques was developed for the original Internet—the Internet of People that we use today—to handle attacks can in theory be applied to the IoT. However, the microcontrollers used in smart objects will seldom be able to put up with their requirements. Furthermore, IoT will be subject to particular conditions not so often faced by today's Internet devices. *Things* will send messages that will trigger immediate reactions from the environment. Capturing and reproducing one such valid message, even if it is encrypted and signed, could lead complex systems such as roadways, factories, and even future cities to misbehave. Some *Things* will harvest energy from the environment for hours before they can say something to the world. And when they talk, one will have to decide whether or not to believe in what they say without having a chance to further discuss the subject (at least not for a couple of hours). Solutions such as transaction authentication and channel masking [6] are of little help in this context.

In this paper, we describe the design, implementation and evaluation of a trustful communication framework for the IoT conceived with these pitfalls in mind. The framework follows a cross-layer design that combines medium access control, location, timing, routing and trustfulness on highly configurable manner. It was evaluated using EPOS' second generation of motes, EPOSMoteII, which features an ARM processor and an IEEE 802.15.4 radio transceiver [13]. It is presented to end users through a trustful communication protocol compatible with TCP/IP, which per-definition ensures end-to-end reliable and ordered delivery. Trustfulness is tackled at MAC level by extending C-MAC [25], EPOS native MAC protocol, with Advanced Encryption Standard (AES) [16] capabilities that were subsequently used to encrypt and authenticate packets. EPOS Precision Time Protocol (PTP) implementation is used to enrich the authentication mechanism with temporal information.

Section II presents the design of EPOS original communication stack. EPOS trustful mechanisms are discussed separately in Section II-G. Section III introduces the new cross-layer design, in which elements of medium access control, location, timing, routing, and security are carefully merged in a single-level protocol. In Section IV we evaluate the new cross-layered protocol on the EPOSMoteII platform, followed by related works in Section V and conclusions in Section VI.

II. EPOS Original Communication Protocol Stack

EPOS original communication protocols stack features a layered architecture as suggested by the OSI model. Although developed to be energy efficient and to present low overhead, this architecture is similar to that of ordinary operating systems

traditionally used in the Internet of People. The original layered architecture is described in the following sections.

A. C-MAC

C-MAC is a highly configurable MAC protocol for Wireless Sensor Networks (WSN) realized as a framework of medium access control strategies that can be combined to produce application-specific protocols [25]. It enables application programmers to configure several communication parameters (e.g. synchronization, contention, error detection, acknowledgment, packing, etc) to adjust the protocol to the specific needs of their applications. An overview of C-MAC is rendered by the activity diagrams of Figures 1, 2, and 3.

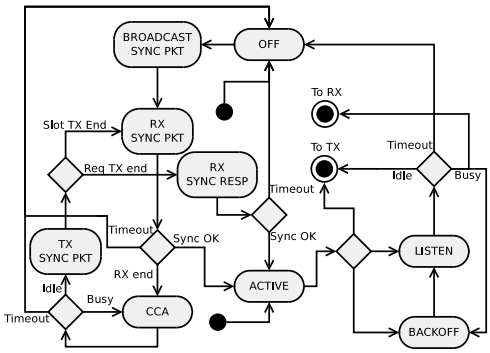

Figure 1: C-MAC synchronization activity diagram.

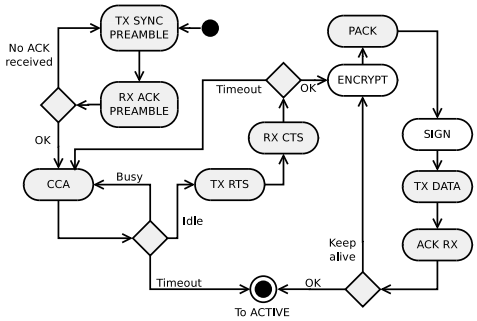

Figure 2: C-MAC transmission activity diagram.

The main configuration points of C-MAC are:

- **Physical layer:** These configuration parameters are defined by the underlying radio transceiver.
- **Synchronization:** Derived from the mechanisms used to exchange synchronization information among nodes.
- **Collision avoidance:** Configure contention mechanisms used to avoid collisions.
- **Acknowledgment:** Define if and how successful or unsuccessful packet exchanges are to be handled.
- **Error handling:** Determine which mechanisms will be used to ensure the consistency of data.
- **Security:** Determine which mechanisms will be used to ensure communication security.

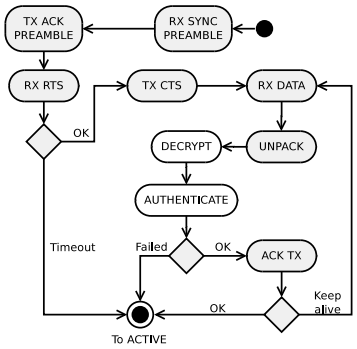

Figure 3: C-MAC reception activity diagram.

When configured to mimic preexisting MAC protocols C-MAC delivers comparable performance. This is due to the use of static metaprogramming techniques, which ensures that configurability does not come at the expense of performance or code size [25]. In this way, C-MAC's instances are fully customized at compile-time and yield extremely lean run-time MACs. C-MAC high configurability was essential to the research being presented here.

B. HECOPS

EPOS location mechanism, the Heuristic Environmental Consideration Over Positioning System [22], defines a distributed location algorithm for wireless sensor networks in which every node estimates its own position after interacting with other nodes. HECOPS establishes a ranking system to determine the reliability of each estimated position and uses heuristics to reduce the effects of measurement errors. So far, HECOPS has been applied in the realm of IoT using two main heuristics: the relationship between signal degradation and distance inferred from the Received Signal Strength Indication (RSSI) provided by C-MAC [21], and the Time Difference of Arrival (TDOA) provided by a UWB transceiver [19]. The first heuristic is depicted in Figure 4. The location of anchor nodes is determined equipping motes with GPS receivers.

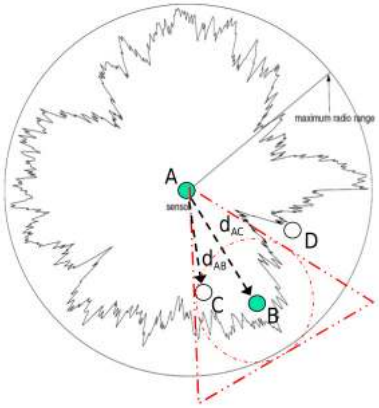

Figure 4: Overview of HECOPS.

C. PTP

Time synchronization is a mandatory OS feature for many distributed applications. EPOS timing protocol [19] delivers clock time across a wireless sensor network in conformance with the IEEE 1588 standard, the Precision Time Protocol (PTP). A node acting as a master clock extracts the base time from a GPS receiver and propagates it to slave clocks following the standard as illustrated by Figure 5. Since propagation is usually done by broadcast or zone multicast, listening nodes take advantage of protocol interactions to recalibrate whenever valid PTP messages are observed in the network. This novel kind of clock, which we named *listener*, adds to masters and slaves while saving considerable amounts of energy as discussed in section IV. EPOS PTP is able to keep a PAN synchronized with sub-millisecond precision.

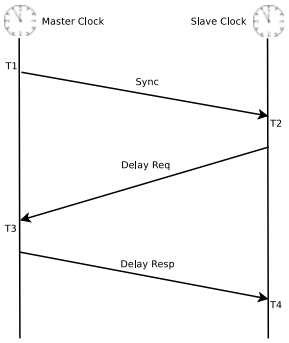

Figure 5: Overview of EPOS PTP.

D. ADHOP

Motes in a wireless sensor network usually propagate data packets in a multihop fashion and IoT devices are likely to follow a similar scheme. In this scenario, routing will strongly influence performance and energy consumption. EPOS Ant-based Dynamic Hop Optimization Protocol (ADHOP) [17], [18] was designed to address these questions. It is a self-configuring, reactive routing protocol able to handle mobile nodes and to balance energy consumption across the network. ADHOP is able to handle dynamic topologies with a combination of heuristics defined around different metrics, thus adjusting routing according with network needs. ADHOP pheromone concentration and evaporation rates are dynamically adjusted considering global information collected and disseminated by ants. A node forwarding too many packets, because it is on a strategic location, will adjust pheromone to favor other routes as soon as it realizes its resources are being drained too quickly. For instance, when energy consumption is given a higher importance by application programmers, ADHOP will adjust pheromone evaporation rates based on residual energy, eventually demoting a previously optimal route and consequently balancing energy consumption across the network.

E. TCP/IP

TCP is a key protocol for the trustful IoT platform being proposed here, that ensures ordered delivery of packets. Its acknowledgement and flow control mechanisms have been optimized to efficiently handle congestion, presumably the unique significant cause for packet loss on low-error rate networks. In the presence of higher error rates and intermittent connectivity, traditional TCP implementations continue to react to packet losses in the same way, causing a significant degradation of performance observed by peers as poor throughput and high latency [8].

The current EPOS IPv4 implementation uses TCP's window-based flow control mechanism to implement an *rendezvous* protocol and thus virtually eliminates buffer management on IoT nodes. The strategy is depicted in Figure 6. Peers announce buffer availability for a single message at a time by adjusting the window length in acknowledgement messages accordingly. Several optimizations have also been conducted to keep IP datagrams in pace with IEEE 802.15.4 127-byte MTU. Energy efficiency is sought in EPOS TCP/IP stack by incorporating the pheromone concept behind ADHOP as the IP routing metric.

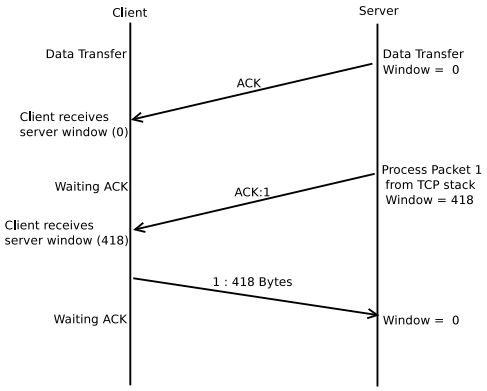

Figure 6: Overview of EPOS TCP/IP - Window 0.

F. Web Services

Many researchers and practitioners are now talking about a *Web of Things* and proposing that our daily objects will be embraced by the Web using the same protocols of the ordinary Internet. We do not believe that ordinary objects will ever implement such protocols. Nevertheless, in order to be able to design the cross-layer protocol being proposed here, we concluded that one such an implementation was necessary. We therefore implemented a small web server for EPOS, featuring the HTTP and the RESTful protocols, thus enabling our "things" to be accessed just like any other web service. Important here is to observe the enormous overhead represented by the request message depicted in Figure 7.

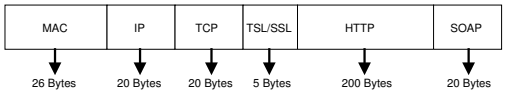

Figure 7: Message format for a RESTful web service request to an EPOSMote.

G. EPOS Strategy for Trustfulness

IoT devices will often communicate through the air using radio channels that are open for everyone to peek and poke [30]. In order to avoid undesired interference, EPOS devises a trust mechanisms that adhere the following premises:

- **Confidentiality:** the protocol must prevent unauthorized access to data. As receivers must have the right key to decrypt it. This calls for a key management strategy.
- **Authenticity:** the protocol must be able to confirm the origin of a message.
- **Integrity:** the protocol must ensure that the message was not modified on the way from sender to the recipient.

We believe that security must be handled at the lowest possible level in the system, since each additional layer of software can potentially make room for exploits. Therefore, we incorporated the proposed trustfulness mechanism into C-MAC. It was accomplished through the addition of new states to C-MAC's finite-state machine and the corresponding microcomponents to its framework. These elements were already present at Figures 2 and 3 in Section II. `ENCRYPT` is responsible for encrypting the payload. `SIGN` attaches the time-stamp, which is also encrypted, and the message authentication code to the packet. `DECRYPT` decrypts the payload, while `AUTHENTICATE` verifies if both the time-stamp and authentication code are valid.

1) Key Management: For key management, we opted for a centralized key distribution scheme. Each sensor shares a symmetric key with the gateway, which is kept in secret by both. These symmetric keys are generated following a Diffie–Hellman scheme over insecure communications channel [5]. We use EPOS' secure key bootstraping scheme as shown on Figure 8.

Our secure key bootstraping protocol enables EPOSmotes to be deployed and keyd securely at a later stage. The protocol's requirements are that the gateway knows the serial number of the nodes he will be sharing keys with, and that the nodes are able to synchronize time, thus enabling the use of an one-time password generation scheme. Both assumtions are easely achievable using our framekwork infrastructure.

We assume that most of communication in an IoT scenario will occur between devices and the gateway, but devices willing to interact with each other directly have the option to ask the gateway for a temporary group key. Alternatively, sporadic device-to-device communication can be handled by the gateway on a store-and-forward scheme.

2) Replay Attacks: To countermeasure replay attacks we added a time-stamp to C-MAC's packets. The engines generating time-stamps across the network are kept synchronized via our PTP implementation. Master clocks are usually gates and produce time seeds based on a GPS receiver or similar device.

3) Message Authentication Code: The ZigBee specification of high level protocols for IEEE 802 personal area networks defines a security architecture that is closely related to the Advanced Encryption Standard (AES) [3]. This has pushed manufacturers to include AES hardware accelerators into many IEEE 802.15.4 platforms. We therefore considered two AES-based Message Authentication Code (MAC) for EPOS: Counter with CBC-MAC (CCM), which is a generic authenticated encryption block cipher mode for AES [28]; and Poly1305-AES, a state-of-the-art computes a 16-byte authenticator of a variable-length message using a 16-byte AES key, a 16-byte additional key, and a 16-byte nonce [4].

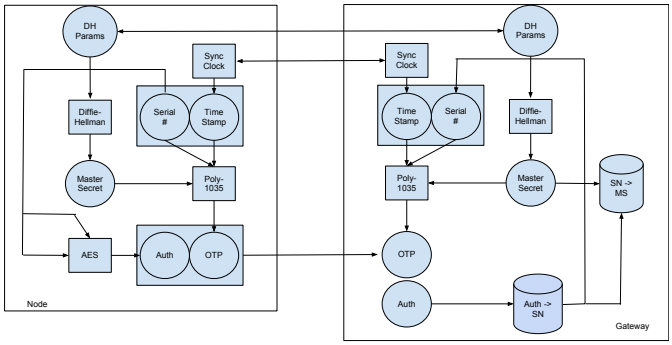

Figure 8: Overview of key management in EPOS.

4) Trusted Packets: Each packet includes the protocols headers, the application data, a time-stamp representing the current network time delivered by PTP, and the MAC produced using the AES accelerator. The application data is encrypted along with the time-stamp. Since the shared key used by AES was negotiated directly with the gateway, decrypting a valid message immediately renders the sender's identity.

III. Cross-layer Protocol

After having implemented EPOS protocol stack piece by piece over almost a decade, we observed that the traditional layered design of the contemporary communication stacks was inducing a lot of data replication on different layers. Compelled by the necessity of designing a communication protocol stack for the Internet of Things that could match its requirements of low overhead and low power, we reorganized C-MAC's microcomponent framework to combine aspects of medium access control, location, timing, routing, and trustfulness on a highly configurable cross-layer protocol.

The proposed cross-layer protocol exploits two characteristics of the original stack responsible for most of the overhead:

- **Information Sharing:** Status information about the local node and about its interactions are used across several layers. Energy availability, residual memory, transmitted and received packets, locally known address mappings are some examples of status information intensively used by several layers. Storing it separately and requesting in on the demands of a single layer is a major source of overhead. We opted for a single shared table containing all the information.
- **Implicit Learning:** The information locally handled by a node about its current status and that of its neighbors can be implicitly update For instance, acknowledge and beacon messages can carry location, timing, and routing

information almost for free. This kind of information can be updated on each received message, thus preventing message exchanges originally conceived to obtain them.

We took advantage of these observations while designing the low-overhead cross-layer protocol. We first combined several pieces of the original finite-state machines to incorporate the behavior of each layer and subsequently adjusted the format of packets to incorporate the corresponding data. The result is depicted in Figure 9.

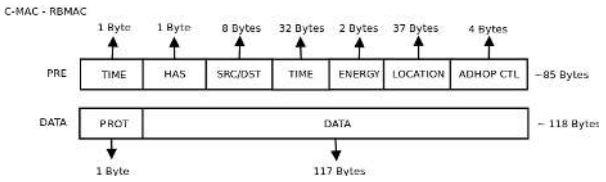

Figure 9: EPOS cross-layer protocol stack from the perspective of packet format.

IV. Evaluation

We gauged the proposed cross-layer protocol by comparing it to the original protocol stack of EPOS. The conceived experiments were carried out on EPOSMoteII atop of OpenEPOS 1.0. Whenever simulation was needed, OMNeT++ was used with a realistic model. The following sections discuss relevant details of both scenarios, EPOSMote and the simulation model.

A. EPOSMote

The EPOSMote is an open hardware project [13]. The project main objective is delivering a hardware platform to allow research on energy harvesting, biointegration, and MEMS-based sensors. The EPOSMoteII platform focus on modularization, and thus is composed by interchangeable modules for each function. Figure 10 shows the development kit which is slightly larger than a R$1 coin.

Figure 10: EPOSMoteII SDK side-by-side with a R$1 coin.

Figure 11 shows an overview of the EPOSMoteII architecture. Its hardware is designed as a layer architecture composed by a main module, a sensing module, and a power module. The main module is responsible for processing, storage, and communication. The model used in this research features a 32-bit ARM7 processor, 128kB of flash, 96kB of RAM, and an IEEE 802.15.4-compliant radio transceiver.

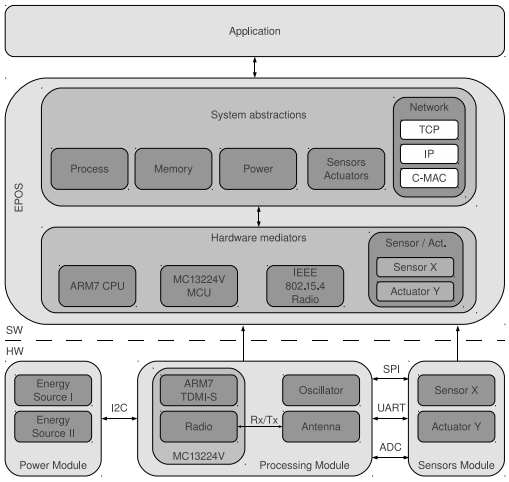

Figure 11: Architectural overview of EPOSMoteII.

B. Simulation Model

OMNeT++ simulator is an extensible, modular, component-based C++ framework for building network simulations. Table I shows the OMNeT++ simulation parameters used for the IEEE 802.15.4 simulated network. In these experiments, each simulation scenario ran for 900 seconds in an environment of high mobility that is conducive to high data loss. The simulation places nodes randomly in a squared area of $1.44\ km^2$ (edges of twelve hundred meters), and each node moves at a maximum speed of five meters per second, according to the Mass Mobility algorithm [20]. Twenty mobile source nodes generates data traffic to other twenty mobile sink nodes. The experiment explores the behavior of ADHOP varying the heuristic information and the number of nodes, ranging from twenty to two hundred. We compare it with AODV and AOER algorithms for data delivery ratio and energy consumption.

Table I: OMNeT++ Configuration

Parameter	
Simulation Time	$900\ seconds$
Number of Nodes	$20 \sim 200$
Area	$1200m\ X\ 1200m$
Mobility Model	$Mass\ Mobility$
Application Message Length	$56\ bytes$
Application Message Frequency	$0.25\ Hz$
ADHOP Header Length	$6\ bytes$
PHY Transmitter Power	$1\ mW$
PHY Sensitivity	$-85\ dBm$
PHY Thermal Noise	$-110\ dBm$
Channel Carrier Frequency	$2.4\ GHz$
Battery Voltage	$3\ V$
Battery Capacity	$2\ mAh$

C. Results

We gauged the implementation of the trustful IoT infrastructure proposed in this paper in respect to three aspects: memory consumption, encryption/decryption time, and energy consumption. For all experiments, we used GCC 4.4.4 to compile the application and the run-time support system (i.e.

EPOS). EPOSMoteII ARM processor clock was set to 24 MHz. Messages were adjusted to carry a payload of 16 bytes when encryption was activated and 7 (request) and 6 (reply) bytes otherwise. EPOSMoteII radio transceiver was adjusted to transmit at 4.5 dBm.

One node acts as a base station for the local IoT, interfacing its nodes to the ordinary Internet[1], while the other one is a sensor node. The base station broadcasts encrypted temperature requests every 10 seconds. The sensor node decrypts the request, collects the required data, and sends back a signed and encrypted reply.

In order to obtain the memory footprint of our implementation, we used the `arm-size` tool that is part of GNU Binutils. Results are shown in Table II. The *AES mediator* column represents the code needed to interact with the AES hardware accelerator in order to accomplish encryption, decryption, and authentication. *App using AES* column presents the code size of the application using the proposed trusted infrastructure and the *App without AES* column the size when using the original, plain text, TCP/IP stack. It is possible to notice that there is a difference between the value of *App using AES* and the sum of *App without AES* and *AES mediator*. This is due to the fact that not all methods from the *AES mediator* are used in *App using AES*. Mediator methods that are not effectively invoked by the client program are eliminated during compilation. This is due to the fact that besides including the mediator code the *App using AES* has to call this code, so it can be executed, whereas *App without AES* has no such calls.

Table II: Memory footprint.

Section	AES mediator	App using AES	App without AES
.text	1336 bytes	47184 bytes	45916 bytes
.data	0 bytes	217 bytes	217 bytes
.bss	10 bytes	5268 bytes	5268 bytes
TOTAL	1346 bytes	52669 bytes	51401 bytes

We used an oscilloscope to measure the time needed to encrypt, decrypt and authenticate messages in our infrastructure. A General Purpose Input/Output (GPIO) pin in EPOSMoteII is connected to the oscilloscope. We run the experiments for one minute and calculated the averages shown in Table III. Obtained values, besides confirming the efficiency of the implementation in terms of execution time, also have a positive impact in the node's battery lifetime.

Table III: Encryption/decryption/MAC check processing time.

	Encryption	Decryption	MAC Check
Time	17 μs	15 μs	12 μs

Figure 12 shows the energy consumed by both applicationsover the time. The small increase in energy consumption for *App using AES* arises from the efficient usage of the hardware accelerator.After 10 minutes executing, the difference is minimal (53.2 J with AES and 52.6 J without), and after 1 hour, the applications have consumed 319.5 J and 315.5 J, respectively, a difference of 1.25%.

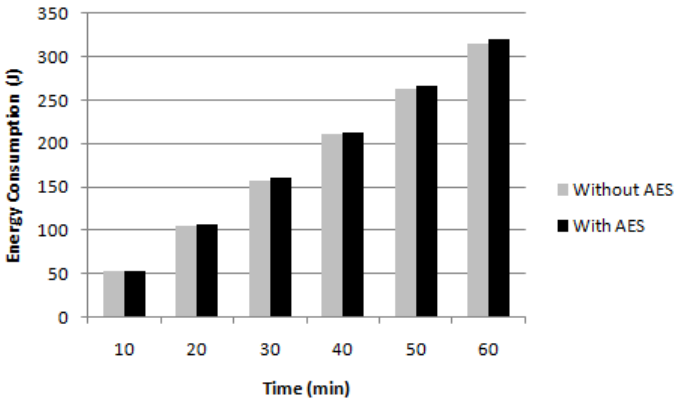

Figure 12: Energy consumption.

The energy efficiency, shown in Figure 13 in logarithmic scale, is the division of the overall energy consumption by the amount of packets successfully delivered. Energy-Aware ADHOP (EA-ADHOP) routing algorithm produces better results in terms of energy efficiency than ADHOP, AODV, and AOER. We can notice that this approach can improve energy use while reducing the energy consumption and enhancing data delivery ratio. This differs from AOER, which has an aggressive method to reduce the energy consumption [24], as shown in Figure 14. This adds to the low bit rate of IEEE 802.15.4 nodes, making the connectivity worse in higher speeds [29]. This means greater competition for the medium implying in collisions, congestions, data loss, and greater energy consumption for mobile, dense, and scalable networks, causing the depletion of energy on the routes.

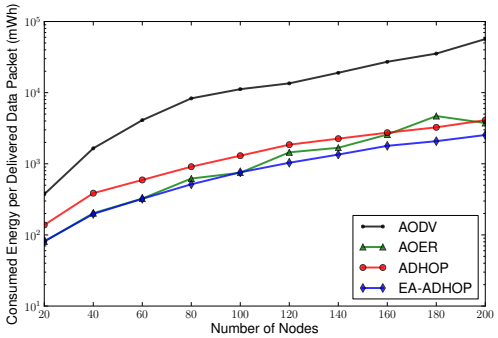

Figure 13: Energy Consumption per Delivered Data.

Another important characteristic of ADHOP is shown in Figure 15. Route Requests, Route Replies, and Route Errors are message types defined by AODV. In ADHOP, data is sent along with the ants thereby decreasing the amount of control packets in the network. Accordingly, our approach tends to produce low routing overhead for sparse networks

[1]For a larger scale experiment, the gateway would rather be configured to provide some sort of NAT service between both realms, thus alleviating the address limitation of IPv4.

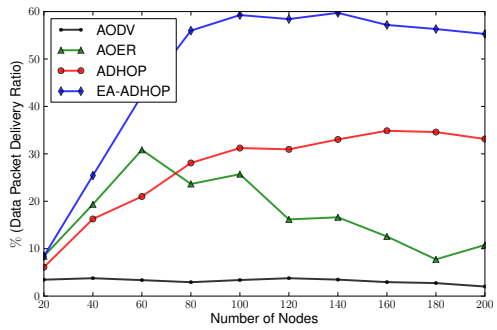

Figure 14: Delivery Ratio of Data Packets.

due to low connectivity. However, it also produces high link failures, shown in Figure 16. We can notice that ADHOP, AODV, and AOER produce better results of link failures than EA-ADHOP. Since this approach aims at energy efficiency instead of connectivity, the links between neighbor nodes tend to be more susceptible to failures.

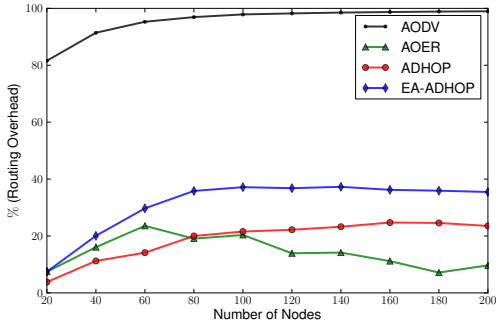

Figure 15: Overhead for the maintenance of the routing mechanism.

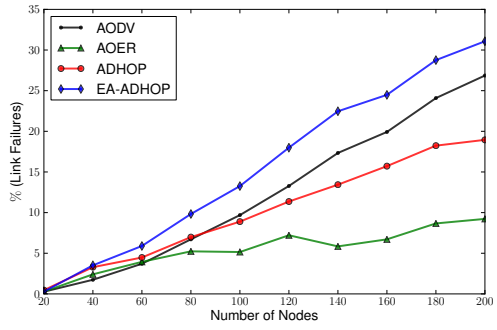

Figure 16: Link Failures

V. Related Work

TinySec [12] defines a link-layer security architecture for Wireless Sensor Networks (WSNs). TinySec supports two different security options: authenticated encryption (TinySec-AE), and authentication only (TinySec-Auth). In authenticated encryption mode, TinySec encrypts the data payload according to the Skipjack block cipher [2] and authenticates the packet with a Message Authenticity Code (MAC). In authentication only mode, TinySec authenticates the entire packet with a MAC, but the data payload is not encrypted. The inclusion of a MAC to ensure the authenticity and integrity have a cost on radio usage and, consequently, in energy consumption. This is because the hash values commonly represent a long sequence of bits. TinySec achieves low energy consumption by reducing the MAC size, hence decreasing the level of security provided. TinySec also does not attempt to protect against replay attacks, and does not discuss how to establish link-layer keys. TinySec was implemented in TinyOS and runs on Mica, Mica2, and Mica2Dot, each using Atmel processors. TinySec has 3000 lines of nesC code [7] and the implementation require 728 bytes of RAM and 7146 bytes of space.

MiniSec [15] is a secure network layer protocol for WSNs which attempts to solve the known problems of TinySec. First, it employs a block cipher mode of operation that provides both privacy and authenticity in only one pass over the message data. Second, MiniSec sends only a few bits of the Initialization Vector (IV) while retaining the security of a full-length IV per packet. In order to protect against replay attacks and reduce the radio's energy consumption, it uses synchronized counters. However, *Jinwala et al.* showed that such scheme requires costly resynchronization routines to be executed when the counters shared are desynchronized (packets delivery out-of-order) [11].

Focusing on sensor battery's useful life, *Braun and Dunkels* [27] introduces an approach to support energy efficient TCP operation in sensor networks. The concept called TCP Support for Sensor nodes (TSS) allows intermediate sensor nodes to cache TCP data segments and to perform local retransmissions TSS does not require any changes to TCP implementations at end points, and simulations show that it reduces the number of TCP data segment and acknowledgement transmissions in a wireless network. *Ganesh* [23] also introduces a mechanism which improves TCP performance, called TCP Segment Caching.

Elrahim et al. [1] proposes an energy-efficient way to implement TCP protocol in scenarios with high losses. They present a modified Congestion Control Algorithm for WSN. By increasing retransmission timeout value, they reduce the number of TCP segment transmissions that are needed to transfer a certain amount of data across a wireless sensor network with relatively high bit/packet error rates.

The size of TCP implementation also is important when developing for resource-constrained sensors. NanoTCP [10] is a protocol stack for WSNs with reduced overhead. The low memory consumption of the protocol show its suitability to resource constrained devices. But nanoTCP is a simplified version of TCP protocol, not being fully compatible. However, other implementations such as uIP and lwIP faithfully represent the TCP protocol.

Huai proposes to cut down duty cycles and decrease the energy consumption of executing the AES algorithm by run-

ning both CTR and CBC-MAC in parallel [9]. Similarly to our scheme, their design employs a hardware accelerator to offload CPU. It uses an 8-bit data path and a shared key expansion module with both AES cores, encryption and authentication. They achieved an encryption time of 71.6 ns for a payload of 17 bytes. Their parallel hardware acceleration provides better results when compared with the sequential AES hardware accelerator in the FreeScale MC13224V present in EPOSMoteII.

Furthermore, this paper analyses communication efficiency through the total delay per hop, and it shows that when the scale of the sensor networks grows, the delay has been doubled, and energy consumption has also increased accordingly.

VI. Conclusions

This paper presented a trustful infrastructure for the IoT developed within the realm of project EPOS. By trustful, we mean *reliable* and *secure*. Aspects such as people privacy and data dependability have not been considered in this paper. The proposed infrastructure is implemented around the EPOSMoteII platform and delivered to end users through a trustful communication protocol stack compatible with TCP/IP. Trustfulness for the infrastructure was achieved through a combination of mechanisms. From TCP/IP, we inherited reliable and ordered end-to-end packet delivery. C-MAC was enriched with AES-based encryption and authentication. It now also timestamps messages to prevent replay attacks. Authentication is performed using a centralized key distribution scheme in which each sensor shares a unique key with the local base station. We also developed a secure key bootstraping for key agreement between nodes and gateways. We experimentally evaluated our proposal in terms of memory consumption, encryption/authentication time, and energy consumption. The results confirm that the proposed infrastructure is able to provide confidentiality, authentication, integrity, and reliability without introducing excessive overhead to a network of things, a key step in making the Internet of Things a daily reality.

References

[1] Salwa El Ramlf Adel Gaafar A. Elrahim, Hussein A. Elsayed and Ibrahim Magdy M. Improving TCP congestion control for wireless sensor networks. In *4th Annual Caneus Fly by Wireless Workshop*, pages 1–6, Montreal, QC, Canada, June 2011.

[2] National Security Agency. Skipjack and kea algorithm specifications, may 1998.

[3] ZigBee Alliance. ZigBee Specification. online, 2007.

[4] Daniel J. Bernstein. The poly1305-aes message authentication code. In *Proceedings of Fast Software Encryption*, pages 32–49, Paris, France, Feb 2005.

[5] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[6] Xinwen Fu, Bryan Graham, Riccardo Bettati, and Wei Zhao. Active traffic analysis attacks and countermeasures. In *Proceedings of the 2003 International Conference on Computer Networks and Mobile Computing*, ICCNMC '03, pages 31–, Washington, DC, USA, 2003. IEEE Computer Society.

[7] David Gay, Philip Levis, Robert von Behren, Matt Welsh, Eric Brewer, and David Culler. The nesc language: A holistic approach to networked embedded systems. In *Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation*, PLDI '03, pages 1–11, New York, NY, USA, 2003. ACM.

[8] Elan Amir Hari Balakrishnan, Srinivasan Seshan and Randy H. Katz. Improving tcp/ip performance over wireless networks. In *Proceedings of the 1st annual international conference on Mobile computing and networking*, pages 2–11, New York, NY, USA, 1995. ACM.

[9] Lian Huai, Xuecheng Zou, Zhenglin Liu, and Yu Han. An energy-efficient aes-ccm implementation for ieee802.15.4 wireless sensor networks. *Networks Security, Wireless Communications and Trusted Computing, International Conference on*, 2:394–397, 2009.

[10] C. Jardak, E. Meshkova, J. Riihijarvi, K. Rerkrai, and P. Mahonen. Implementation and Performance Evaluation of nanoIP Protocols: Simplified Versions of TCP, UDP, HTTP and SLP for Wireless Sensor Networks. In *IEEE Wireless Communications and Networking Conference*, pages 2474–2479, March 2008.

[11] D. Jinwala, D. Patel, S. Patel, and K.S. Dasgupta. Replay protection at the link layer security in wireless sensor networks. In *Computer Science and Information Engineering, 2009 WRI World Congress on*, volume 1, pages 160 –165, 31 2009-april 2 2009.

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

[13] Software/Hardware Intergration Lab. EPOS Project.

[14] Hyeopgeon Lee, Kyounghwa Lee, and Yongtae Shin. Implementation and Performance Analysis of AES-128 CBC algorithm in WSNs. In *The 12th International Conference on Advanced Communication Technology*, pages 243–248, 2010.

[15] M. Luk, G. Mezzour, A. Perrig, and V. Gligor. Minisec: A secure sensor network communication architecture. In *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, pages 479 –488, april 2007.

[16] NIST. Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197, November 2001.

[17] Alexandre Massayuki Okazaki and Antônio Augusto Fröhlich. Ant-based Dynamic Hop Optimization Protocol: a Routing Algorithm for Mobile Wireless Sensor Networks. In *Joint Workshop of SCPA 2011 and SaCoNAS 2011 - IEEE GLOBECOM 2011*, pages 1179–1183, Huston, Texas, USA, December 2011.

[18] Alexandre Massayuki Okazaki and Antônio Augusto Fröhlich. ADHOP: an Energy Aware Routing Algorithm for Mobile Wireless Sensor Networks. In *IEEE SENSORS 2012*, Taipei, Taiwan, October 2012.

[19] Peterson Oliveira, Alexandre Massayuki Okazaki, , and Antônio Augusto Fröhlich. Sincronização de Tempo a nível de SO utilizando o protocolo IEEE1588. In *Brazilian Symposium on Computing System Engineering*, Natal, Brazil, November 2012.

[20] Charles E. Perkins and et al. Optimized smooth handoffs in mobile ip. In *IN PROCEEDINGS OF ISCC*, pages 340–346, 1999.

[21] Rafael Pereira Pires, Lucas Francisco Wanner, and Antônio Augusto Fröhlich. An Efficient Calibration Method for RSSI-based Location Algorithms. In *6th International IEEE Conference on Industrial Informatics*, pages 1183–1188, Daejeon, Korea, July 2008.

[22] Ricardo Reghelin and Antônio Augusto Fröhlich. A Decentralized Location System for Sensor Networks Using Cooperative Calibration and Heuristics. In *9th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 139–146, Torremolinos, Malaga, Spain., October 2006.

[23] R. Amutha S. Ganesh. Energy efficient transport protocol for wireless sensor networks. In *2nd IEEE International Conference on Computer Science and Information Technology*, pages 464–468, August 2009.

[24] Bing Shuang, Zhenbo Li, and Jiapin Chen. An ant-based on-demand energy route protocol for ieee 802.15.4 mesh network. *International Journal of Wireless Information Networks*, 16:225–236, 2009.

[25] Rodrigo Steiner, Tiago Rogério Mück, and Antônio Augusto Fröhlich. C-MAC: a Configurable Medium Access Control Protocol for Sensor Networks. In *9th IEEE Sensors*, pages 845–848, Waikoloa, HI, USA, November 2010.

[26] Hung-Min Sun, Shih-Ying Chang, A.B. Tello, and Yen-Hsuen Chen. An authentication scheme balancing authenticity and transmission for wireless sensor networks. In *Computer Symposium (ICS), 2010 International*, pages 222 –227, dec. 2010.

[27] Thiemo Voigt Torsten Braun and Adam Dunkels. TCP support for sensor networks. In *Fourth Annual Conference on Wireless on Demand Network Systems and Services*, pages 162–169, January 2007.

[28] D. Whiting, R. Housley, and N. Ferguson. *RFC 3610 - Counter with CBC-MAC*. online, Sep 2003.

[29] K. Zen, D. Habibi, A. Rassau, and I. Ahmad. Performance evaluation of ieee 802.15.4 for mobile sensor networks. In *5th IFIP International Conference on Wireless and Optical Communications Networks*, pages 1–5, may. 2008.

[30] Yun Zhou, Yuguang Fang, and Yanchao Zhang. Securing wireless sensor networks: A survey. *IEEE Communications Surveys & Tutorials*, 10:6–28, 2008.