Towards A Trust Computing Architecture for RPL in Cyber Physical Systems

Sebastian Seeber*, Anuj Sehgal†, Björn Stelte*, Gabi Dreo Rodosek*, Jürgen Schönwälder†
*Universität der Bundeswehr München, Faculty of Computer Science, 85577 Neubiberg, Germany
Email: {sebastian.seeber, bjoern.stelte, gabi.dreo}@unibw.de
†Computer Science, Jacobs University Bremen, Campus Ring 1, 28759 Bremen, Germany
Email: {s.anuj, j.schoenwaelder}@jacobs-university.de

***Abstract*—Cyber Physical Systems (CPSs) are widely expected to be formed of networked resource constrained devices. To suit the constraints of such networks, the IETF developed the RPL routing protocol for Low-power and Lossy Networks (LLNs). Security in CPSs is important for maintaining the integrity and privacy of data, while also improving network resiliency to attacks. Even though RPL provides support for integrity and confidentiality of messages, details regarding key management and signatures are not covered. Since complexity and size is a core concern in LLNs, off-loading the security features to a Trusted Platform Module (TPM) can make it possible to include sophisticated security provisions in an RPL implementation. This paper presents how it would be possible to use the security mechanisms of a TPM in order to secure the communication in an RPL network.**

I. Introduction

Many Cyber Physical Systems (CPSs) are expected to be built using embedded devices with limited computing resources and low-power low-data-rate wireless radios [1]. To avoid proprietary systems leading to non-interoperable devices, the IETF developed the RPL routing protocol to address routing in Low-power Lossy Networks (LLNs) in a standard way [2]. In fact, RPL is also the default routing protocol for LLN IP networking.

Since RPL is a likely candidate for use in CPSs, concerns regarding exposure of data, system compromise due to attacks and identity assurance for authorization extend to it as well. While the RPL standard does provide information on ensuring integrity and confidentiality of messages, implementation of this is beyond scope. This makes it necessary to develop suitable security for LLNs using RPL. Cryptographic algorithms are known to occupy the most memory and take many CPU cycles, thereby greatly affecting the overall performance of a resource constrained device (RCD) [3]. As such, it might be better to off-load most of the security features to a co-processor in order to minimize resource usage. Using a Trusted Platform Module (TPM) [4] as a co-processor on the RCD is one approach, which provides the added advantage of tamper-free data storage and implied trust in certain situations.

In this paper we discuss how a TPM can be used in order to secure communication in an RPL network. Section 2 provides an overview of other approaches for securing LLN networks, including other TPM based methods. This is followed by an overview of the RPL protocol and its secure modes in Section 3. An overview of the trusted computing architecture is provided in Section 4. This is used to outline the proposed TPM based trust establishment and key exchange mechanism used to secure RPL in Section 5. Some preliminary results are presented in Section 6, followed by future work in Section 7. Concluding remarks are made in Section 8.

II. Related Work

The use of TLS/DTLS for securing application layer communications has been previously investigated. Some approaches perform all the cryptographic work on the LLN's main processor and assume usage of pre-shared keys, since key exchange mechanisms are deemed invalid options as a result of the limited bandwidth and processing power [3]. A TPM based approach for DTLS leverages the TPM's hardware accelerated RSA encryption and tamper-proof key storage [5].

There are also some other studies that use a TPM for asymmetric encryption and key exchange in wireless sensor networks [6], [7]. However, none of these approaches establish trustworthiness of a node before exchanging keying material via RSA, and neither do they provide for a method to securely change keys. They also do not protect against differential cryptanalysis. Our TPM based approach aims to solve these problems as well.

III. The RPL Protocol

RPL was developed by the IETF RoLL working group for the use in LLNs [2]. Running RPL in a network leads to the formation of a Destination Oriented Directed Acyclic Graph (DODAG) with a root. RPL supports multiple objective functions for goal based topology optimization, e.g. saving energy and reducing number of hops. As such, each network can have multiple DODAGs, each with its own objective function and root.

Formation and maintenance of the RPL DODAG is carried out using control messages. RPL provides for the following modes of operation, two of which are secure and may lead to encryption of the control messages. *(1) Unsecure*: In this mode RPL control messages are sent without security mechanisms. Link-layer or other security techniques are recommended for use in this case. *(2) Preinstalled:* This mode describes the scenario when nodes joining an RPL instance have a key available to them beforehand. These symmetric keys are used to

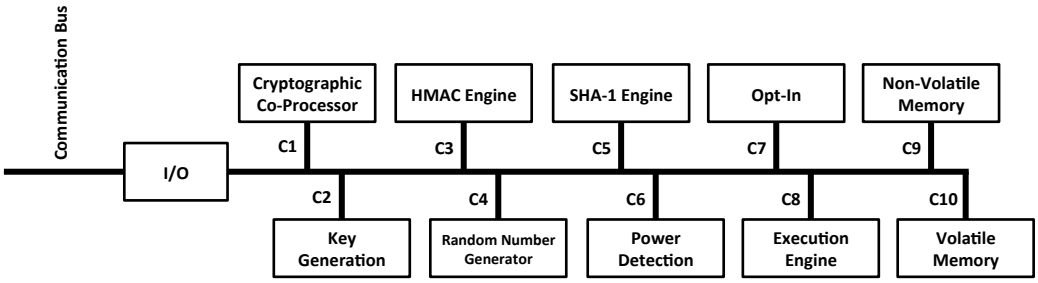

Figure 1: The conceptual architecture of a TPM [4].

generate and handle secure RPL messages. *(3) Authenticated:* In this mode nodes are required to obtain the key from an authentication authority. Once obtained, the symmetric key can be used to generate and process secure RPL messages. The management procedures for this key are beyond the scope of the RPL standard.

While the *Preinstalled* mode of RPL assumes that keys are available to the nodes at network start-up, it does not allow for replacing this key in situations when a malicious node may learn the key via differential cryptanalysis. To avoid such attacks, ideally the symmetric key should be changed periodically. This creates the need for a key exchange mechanism. A trust establishment method to verify the identity of the key supplier and client-node also becomes necessary in this situation.

On the other hand the *Authenticated* mode requires keys to be provided by authentication authorities, and as such, requires a key exchange mechanism and trust establishment method as well. Besides using symmetric keys for encryption of data in packets, the RPL standard also requires secure packets to carry an RSA signature in them.

Securing data packets is beyond the scope of our current work since DTLS is recommended as the datagram security protocol of choice for LLNs [8]. The goal of our work is to secure RPL by encrypting control messages and providing node authentication.

IV. Trusted Computing Architecture

The TPM is a hardware component that securely stores digital keys, certificates and passwords [4]. TPMs are designed to protect key operations and other security tasks that would otherwise be performed on unprotected interfaces, in unprotected communications. Only plain text data of the system that can be used without violating security or privacy is accessible to exterior actions. A conceptual overview of the TPM architecture is provided in Figure 1.

The cryptographic co-processor (C1) implements cryptographic operations within the TPM. The TPM provides cryptographic operations, such as asymmetric key generation and encryption/decryption (RSA), hashing (SHA-1) and random number generation. Other relevant TPM features include the key generation component (C2) to create RSA key pairs and symmetric keys. The generating function is a protected capability and the private keys are held in shielded locations.

The HMAC Engine (C3) provides two pieces of information to the TPM; proof of knowledge of the authentication and authorization data, and proof that the request arriving is authorized and has no modifications made to the command in transit. The SHA-1 (C5) hash capability is primarily used internally by the TPM, as it is a trusted implementation of a hash algorithm.

Besides these, a TPM also provides secure storage of keys in its memory via write-only and read-only access. Each TPM also has a public-private RSA key-pair created, uniquely identifying it, during manufacture. These keys and the write-only memory features of the TPM can be useful in secure key exchanges and trust establishment.

V. Proposed Approach

Securing an RPL network requires key generation, storage and exchange. However, before providing or receiving keying material, a node's trustworthiness must also be established. Lastly, RPL requires messages to be signed using RSA. Using a TPM to assist in solving these issues provides not only a level of implied trust, since keying material stored in write-only memory cannot be compromised by an attacker but also assists in reducing the cryptographic processing load on RCDs.

A TPM may also be used to solve the message signature issue as TPMs are capable of generating RSA signatures. The only drawback is that the current TPM specification only allows for SHA-1 hashes to be used in the signatures, while RPL requires SHA-256 hashes. However, the SHA-256 hash is expected to be available in the upcoming version of the specification. In the meantime, we propose using SHA-1 for testing purposes.

As such, we propose developing a RCD node that includes a TPM chip. Adding a TPM chip, such as the Atmel AT97SC3204T does not increase the physical footprint significantly. The cost of nodes will also not increase by much since these TPMs can be purchased for close to $1. Lastly, integration of TPMs with existing RCDs using popular microcontroller families (AVRs, MSP430, etc.) is also not complicated since it uses a standard IIC bus, which is available on nearly all modern microcontrollers.

A TPM solves the message signing, key generation and storage problems, a separate method for establishing trust between nodes and key exchanges needs to be designed. A TPM can assist with these as well.

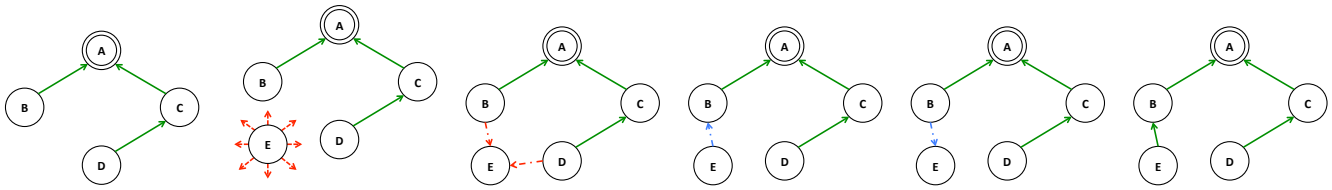

Figure 2: The procedure for a new node to prove trust in order to obtain a Group Key, which is used to join a secure RPL network. (a) A secure RPL DODAG. (b) Node *E* broadcasts *GKRequest* to obtain Group Key. (c) Nodes *B* and *D* respond with *GKChallenge* message to prove their trustworthiness. (d) Node *E* responds with *GKChallengeResponse* to prove it's trustworthiness. (e) Node *B* responds with *GKResponse* to provide Group Key to node *E*. (f) Node *E* uses Group Key to join secure RPL DODAG.

A. Trust Establishment and Key Exchange

As a pre-requisite for our approach, the TPM of each node must have a set of up to 5 symmetric-keys stored in the write-only section of it's memory. This set of keys should be identical on all nodes. These keys will be used in the trust establishment and identity verification procedure and are as such referred to as Identity-Establishment Keys (IKs). A set of 5 keys is used, instead of a larger number, since the TPM provides only limited storage capabilities. In general, using larger number of keys would improve security, as it will become clear from the node join procedure. Of course, another option would be to use a key of larger bit-size, however, TPMs only support key sizes up to 2048-bit, and diversifying keying information reduces the chances of successful differential cryptanalysis.

The TPM's asymmetric key-pair is used by our approach in order to securely pass the symmetric Group Key (GK) used by the RPL network. This GK is stored in the TPM, in order to ensure its security. The overall procedure a node must follow to establish trust and then obtain the Group Key is outlined below.

1) Node Join Procedure: Consider an RPL DODAG as shown in Figure 2a. When a new node wishes to join an existing secure RPL network, it must first obtain the appropriate GK, so that it can decrypt the contents of the RPL control messages. Rather than using a centralized key authority, as is suggested by the RPL standard, we propose that a new node can obtain this key from any node in its neighborhood, which is participating in a DODAG.

As such, the node wishing to join the RPL network broadcasts a *GKRequest* message, as node *E* is shown doing in Figure 2b. This message contains the node's public key ($PubKey_E$) and a randomly generated number ($RandNum_E$). It is important to note that both, $PubKey_E$ and $RandNum_E$, are provided by the TPM.

Nodes within the local neighborhood, i.e. *B* and *D*, participating in an RPL DODAG encrypt $RandNum_E$ using a randomly chosen IK stored on the TPM, leading to $EncRandNum_{E_{IK_n}}$. The nodes now respond with a *GKChallenge* message, Figure 2c, which is encrypted using $PubKey_E$, containing $EncRandNum_{E_{IK_n}}$ and another random number generated locally, $RandNum_B$ or $RandNum_D$. In order to prevent sleep derivation attacks, we limit the number of join requests processed by nodes to five per minute.

Upon receiving the response, node *E* decrypts the message using $PrivKey_E$. It then proceeds to decrypt $EncRandNum_{E_{IK_n}}$ with the IKs, until the expected random number result is obtained. If none of the IKs lead to a valid decryption, the node which sent this response is deemed untrustworthy and further communication with it is ignored. On the other hand if a valid decryption is obtained, the node wishing to join now trusts the node to provide a valid GK. Supposing a valid decryption was obtained only through the response from node *B*, node *E* must now prove its trustworthiness before receiving the key. A similar procedure is followed, wherein node *E* responds with a *GKChallengeResponse* message, Figure 2d, containing $EncRandNum_{B_{IK_n}}$. A valid decryption of this proves trustworthiness of node *E*.

Using this procedure, mutual trust of both nodes is established. Since the IKs are stored within the write-only section of the TPM, these keys cannot be easily compromised and provide for a trustworthy encryption source. Also, using the TPM's random number generator provides a dependable source of this data. Normally, challenge-response methods might be subject to differential cryptanalysis attacks that can decipher the encryption key used by gathering enough samples of encrypted messages. However, randomly switching between IKs ensures the lack of a discernible pattern, thereby denying the attacker successful cryptanalysis.

At this point, node *B* sends a *GKResponse* message, Figure 2e, encrypted with $PubKey_E$, containing the group key of the RPL network. Upon receiving the group key, node *E* stores it within the TPM to prevent tampering. Node *E* can now process all the secure RPL control messages (DIO, DIS and DAO), thereby joining a network normally.

The initial startup procedure would be the same as well, except that only a root node would exist and the GK would be obtained from it. The TPM will be used to generate a GK.

2) Key Update Procedure: Since an attacker may decipher the GK using differential cryptanalysis, it is important to periodically change the GK. It may also be necessary to change the GK on-demand. As such, a dependable key change

Table I: Performance comparison between AVR only and TPM based approach for *1-bit* of data.

	AVR		AVR + TPM	
Operation	***Time***	***Energy***	***Time***	***Energy***
RSA Encrypt	279.70 ms	34960 μJ	*18.24 ms*	*3063 μJ*
RSA Decrypt	6.16 ms	770 μJ	*4.42 ms*	*735 μJ*

mechanism is also important.

The procedure is started by the root, by sending a *GKUpdateStart* message to its children. Assuming the network shown in Figure 2f, this message is sent by node *A* to nodes *B* and *C*. It contains $PubKey_A$ and $RandNum_A$. The children generate $EncRandNum_{A_{IK_n}}$ by using a randomly chosen IK. $EncRandNum_{A_{IK_n}}$ is sent to the parent, along with $RandNum_{(B|C)}$ in the *GKUpdateChallenge* message. If the parent is able to obtain an appropriate decryption of $EncRandNum_{A_{IK_n}}$ by using one of the IKs, then it may trust the child with a new GK. As such, it responds with the *GKUpdate* message that contains the new GK and $EncRandNum_{(B|C)}$, encrypted with $PrivKey_A$.

If $EncRandNum_{(B|C)}$ can be decrypted correctly using an IK by the child, trustworthiness of the parent is established and the new GK can be stored in the TPM. The child node must then repeat this procedure for any children it further has, till such time all nodes have received the new GK.

This approach leverages the security provided by a TPM to ensure the trustworthiness of the keys used establishing trust and encrypting the GKs. Not only can this be used to provide keys in the *Authenticated* secure mode of RPL, but it can also be used to periodically rotate keys in the *Preinstalled* mode.

VI. Preliminary Results

Since our proposed hardware node is not yet ready, we have used Atmel AVR Simulator[1] in order to obtain some preliminary results. Using the ATMega128 microcontroller running at 4 MHz as the target, the AVRCryptoLib[2] was used in order to perform RSA encryption and decryption using a 64-bit key stored in flash. To obtain an approximation of improvements that a TPM might produce, the AVRCryptoLib was replaced with a call to a library virtualizing TPM operations[3]. For energy calculations, the AVR was assumed running at 5V consuming 25mA and a TPM chip at 3.3V and 50.4mA[4].

It is clear from the results presented in Table I that using a TPM can provide significant energy and processing time savings. Also, it is worth noting that the RSA algorithm occupies about 1 KB in flash, at the 64-bit key size, which can also be saved by using a TPM to handle RSA operations. Implementation on hardware is likely to produce results that are somewhat different since these values do not take into account time spent in writing and retrieving values from physical registers. But these simulated results provide a good approximation and it is safe to say that energy savings of up to 90% and similar processing time reductions can be obtained by using a TPM. Similar gains can be expected while using the TPM's RSA signature and SHA-1 hash features.

As larger key sizes are used, up to 2048-bit for RSA, the advantage of off-loading the cryptography tasks to a TPM presents even more advantages, since such key sizes would cause nearly all of the node's memory to be used.

[1] http://avr.sourceforge.net

[2] http://www.das-labor.org/wiki/AVR-Crypto-Lib/

[3] http://tpm-emulator.berlios.de

[4] These values are based on the ATMega128 and AT97SC3204T datasheets.

VII. Future Work

The TPM based approach outlined here needs to be thoroughly evaluated and compared to other approaches. Simulation results already show the in-principle gains of using a TPM, but performance analysis on real hardware is needed. Furthermore, it is important to measure network lifetimes since CPSs may be battery powered, effect of network size on key dissemination, packet overhead, latencies, processing times, disconnects during key changes and effect of different cryptographic attacks.

VIII. Conclusion

The RPL standard allows secure modes, but keying details are left out. Our approach is to use a TPM, which provides accurate and tamper-proof data in insecure environments, while ensuring integrity and authenticity of received messages. We designed a trust establishment and key exchange mechanism around the implied trust a TPM offers, to provide keys for secure RPL modes. Unlike other approaches, this ensures that nodes only provide keys to and use those supplied by trustworthy nodes. Using a TPM on RCDs reduces the processing load on the main processor. With our approach, dissemination of misleading routing information, which affects the availability of the whole network, can be effectively prevented by using a TPM.

Acknowledgment

This work was partly funded by Flamingo, a Network of Excellence project (ICT-318488) supported by the European Commission under its Seventh Framework Program.

References

[1] F.-J. Wu, Y.-F. Kao, and Y.-C. Tseng, "From Wireless Sensor Networks towards Cyber Physical Systems," *Pervasive and Mobile Computing*, vol. 7, no. 4, pp. 397–413, Aug 2011.

[2] T. Winter and P. Thubert, "RPL: IPv6 Routing Protocol for Low power and Lossy Networks," *IETF RFC 6550*, Mar 2012.

[3] A. Sehgal, V. Perelman, S. Kuryla, and J. Schönwälder, "Management of Resource Constrained Devices in the Internet of Things," *IEEE Communications Magazine*, vol. 50, no. 12, pp. 144–149, 2012.

[4] S. L. Kinney, *Trusted Platform Module Basics: Using TPM in Embedded Systems (Embedded Technology)*. Newnes, August 2006.

[5] T. Kothmayr, W. Hu, C. Schmitt, M. Bruenig, and G. Carle, "Securing the internet of things with DTLS," in *9th ACM Conference on Embedded Networked Sensor Systems*, Seattle, WA, Nov 2011.

[6] W. Hu, P. Corke, W. C. Shih, and L. Overs, "secFleck: A Public Key Technology Platform for Wireless Sensor Networks," in *6th European Conference on Wireless Sensor Networks*, Cork, Ireland, Feb 2009.

[7] W. Hu, H. Tan, P. Corke, W. C. Shih, and S. Jha, "Toward trusted wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 7, no. 1, pp. 1–25, Aug 2010.

[8] Z. Shelby, K. Hartke, and C. Bormann, "Constrained Application Protocol (CoAP)," *IETF Internet Draft <draft-ietf-core-coap-18>*, Jun 2013.