A comprehensive and lightweight security architecture to secure the IoT throughout the lifecycle of a device based on HIMMO

Oscar Garcia-Morchon, Ronald Rietman, Sahil Sharma, Ludo Tolhuizen, and Jose Luis Torre-Arce

Philips Group Innovation, Research, Eindhoven, The Netherlands
{oscar.garcia, ronald.rietman, sahil.sharma, ludo.tolhuizen, jose.luis.torre.arce}@philips.com

Abstract. Smart objects are devices with computational and communication capabilities connected to the Internet forming the so called Internet of Things (IoT). The IoT enables many applications, for instance outdoor lighting control, smart energy and water management, or environmental sensing in a smart city environment. Security in such scenarios remains an open challenge due to the resource-constrained nature of devices and networks or the multiple ways in which opponents can attack the system during the lifecycle of a smart object. This paper firstly reviews security and operational goals in an IoT scenario inspired in a smart city environment. Then, we present a comprehensive and lightweight security architecture to secure the IoT throughout the lifecycle of a device. Our solution relies on the lightweight HIMMO scheme as the building stone and shows how HIMMO is not only efficient resource-wise, but that it enables advanced IoT protocols and deployments. Our design and analysis show that our HIMMO-based security architecture can be easily integrated in existing communication protocols such as IEEE 802.15.4 or OMA LWM2M providing a number of advantages that existing solutions cannot provide both performance and operation-wise.

Keywords: Lightweight, Key distribution, Comprehensive Security Architecture, Internet of Things.

1 Introduction

The ubiquitous connection of devices to the Internet, the Internet of Things (IoT), will account for more than a third of the total Internet connections by 2018, according to the Cisco M2M Devices Forecast 2013–2018 [1]. These devices will be deployed in multiple scenarios including smart homes, healthcare, or smart cities. In each of these environments, multiple applications are enabled: the IoT in a smart city can mean connecting city infrastructure such as water meters, environmental sensors, or lighting infrastructure to automate and improve city flows and functionalities.

According the same Cisco Forecast, privacy and security are still two of the technical issues that remain unsolved. The reason is that solutions created for traditional computer networks do not suit the IoT performance, operational and security requirements. For instance, public-key cryptography allows any pair of devices to setup a secure channel or enables accountability. However, it is computationally expensive and requires the exchange of long keys, which has a negative impact during IoT operation. Symmetric cryptography is lightweight but it does not scale, in particular regarding key distribution and management, and this can lead to a lower security level. For instance, if a wireless network relies on a single key, then the capture of a single device will break down the whole system. Finally, IoT scenarios involve both traditional and new threats and security protocols and primitives should be adapted to address them. In particular, a security solution for the IoT should be secure and technically feasible (e.g., performance wise) not only during operation but during the whole lifecycle of a device.

In this context, this paper firstly reviews these operational and security goals in the context of the lifecycle of a smart object deployed in a smart city scenario. We then propose a comprehensive security architecture that addresses both goals during the lifecycle of a smart device. Our security architecture relies on the recently introduced and lightweight HIMMO scheme [2]. In this paper we show that HIMMO not only provides good performance, but that it also enables very attractive features from the point of view of deployment, operation, and maintenance. Our solution relies on an infrastructure of Trusted Third Parties (TTPs) for the management of the HIMMO security domains, device credentials and keying materials. This facilitates a secure manufacturing process and distribution of HIMMO keying materials. The easy integration of HIMMO with standard protocols such as LWM2M or IEEE 802.15.4 and simple extensions of these protocols allows us to ensure efficient and secure network access and device registration with a back-end server. Secure network operation is further achieved in the sense of full collusion resistance and easy and lightweight management of device credentials.

The rest of the paper is organized as follows: Section 2 describes the use cases for IoT in a smart city scenario, introduces relevant communication protocols, and analyzes operational and security goals for the IoT. Section 3 reviews the HIMMO scheme. Section 4 details the proposed security architecture according to the typical lifecycle of a smart object. Section 5 describes the implementation and evaluation of the main building blocks of our security architecture. Section 6 discusses how our architecture addresses the identified operational and security goals and compares it with related work. Section 7 concludes this paper and points out future work.

2 Background

2.1 Use cases

The IoT in the context of smart cities enables services such as outdoor lighting control, water control, smart energy networks, and environmental sensing [3]. For instance, an environmental sensing network based on the IoT can be realized by means of a wireless mesh network that enables the communication between sensors: the environmental sensors would use the mesh network to reach a border router that would further forward the messages containing the gathered measurements towards a back-end system in charge of device and data management. Here, the devices constitute one of the key enablers of these services.

However, the limited amount of resources, e.g., regarding energy or communication, has heavily limited and limits the functionalities that these systems can offer. Security is specially affected by this since most cryptographic primitives or protocols are just too heavy for many use cases so that a designer is confronted with two options: (i) using strong security, but that negatively affects the operation of the system and the way a user interacts with the system, or (ii) using weaker security, in order to not affect the expected system operation. In this context, it is important to note that devices follow a lifecycle [4] that is to be considered in order to build a security architecture. This lifecycle includes several phases: manufacturing, bootstrapping, commissioning, and operational. In each of these phases, the system should remain secure while operating as expected.

2.2 Relevant protocols

The above architecture in which a device communicates over a mesh network with a back-end system can be realized by means of multiple communication protocols. For instance, 6LoWPAN [5]/IEEE 802.15.4 [6] networks enable IP connectivity in a mesh network. Currently there are multiple standards in development such as ZigBee-NAN or Wi-Sun. Network connectivity from the border router to the back-end can be based on a cellular link. End to end communication can be based on OMA LWM2M, in which application data is exchanged between client and server by means of CoAP [7] and the end to end communication is secured by means of DTLS [8] using pre-shared keys, raw public-keys or certificates.

While protocols are available, security primitives are not optimal. For instance, IEEE 802.15.4 networks often rely on a network or system wide symmetric-key. The performance of DTLS and its cipher modes lack either flexibility or performance in many cases.

2.3 Operational and performance goals

The above scenarios have very specific operational and performance needs. A security architecture for the IoT should address all of them.

Because of the resource-constrained nature of the devices, security solutions should achieve good performance (**O-1**) regarding energy consumption, bandwidth requirements, number of round trips, memory needs, and CPU usage.

The deployment of an IoT network is usually done in several phases. It should thus be possible to add devices to a running system in a simple way (**O-2**).

Scalability is a key requirement so that a very high number of devices and back-end servers can be easily supported (**O-3**). In order to ensure this scalability, it is required to enable easy management of device's credentials and attributes in both centralized and distributed communication patterns (**O-4**).

Another important requirement refers to the easy integration with existing communication protocols and architectures (**O-5**). This facilitates adoption and allows for a smooth transition by avoiding the costs related to a change in the technology of the network.

Finally, solutions should fit the use cases not only during the operational phase but throughout all the lifecycle of a device (**O-6**). In the context of IoT this also means that the deployed solutions have to remain secure during a long period of time (10, 20, or even 30 years) (**O-7**).

2.4 Attack model and security goals

In addition to the above operational and performance goals, we consider an attack model in which the opponent can aim at disrupting the system operation at different stages of the lifecycle of a device. Next, we identify potential attacks and discuss security goals that aim to prevent them.

First of all, the attacker (either external or insider) can aim at compromising a root of trust, such as a Certification Authority (CA) in a Public Key Infrastructure (PKI). This would allow him to gain full control over the system. One example of this type of attack is the one suffered by DigiNotar [9]. Therefore, the first security goals refer to being resilient to the compromise of a root of trust (**S-1**) and ensuring that a single root of trust cannot monitor and control communication links (**S-2**). This last requirement should still be compatible with key escrow if required (**S-3**).

The next type of attack focuses on the manufacturing process in which the devices are configured with credentials and secret keys. If an opponent manages to get control on a manufacturing facility, then he can modify or copy this information. An important security goal is to facilitate a secure manufacturing process that prevents this (**S-4**).

The next type of attack can happen when the devices are being deployed. In this case, the back-end server might be exposed to fake devices and, in a similar way, fake servers might impersonate the actual back-end server to gain control over the devices. Authentication and authorization of device (**S-5**) and back-end server (**S-6**) is required to prevent this situation. Even with end to end authentication, the devices that are routing the information could still be exposed to a Denial of Service (DoS) attack, since they would not be able to identify by themselves the communicating parties. Thus, another goal is the prevention of DoS attacks during network access (**S-7**).

During operation, an attacker might aim at physically capturing devices to misuse their secret keys and credentials towards the back-end server or towards any other device in the network. Therefore, a key goal will be that the compromise of any number of devices does not affect the security of the whole system (**S-8**). Another goal to deal with this attack is to facilitate the identification and blacklisting of compromised devices (**S-9**).

Also during operation, another security goal refers to the capability of establishing a common shared key for providing further security services (**S-10**).

As the devices will be on the field for many years, a security solution should provide long term security including resilience against post-quantum attacks (**S-11**).

Other security goals are perfect forward secrecy (**S-12**), meaning that a session key derived from a set of long-term keys cannot be compromised if one of the long-term keys is compromised in future, and non-repudiation (**S-13**), e.g. allowing a metering device signing the energy consumption so that there is proof of the amount of consumed energy.

3 HIMMO

HIMMO is a Key Pre-Distribution Scheme (KPS), a concept introduced by Matsumoto and Imai in 1987 [10]. Blundo *et al.* [11] present an elegant and efficient KPS based on symmetric polynomials. However, their KPS is prone to collusion attacks: if an attacker has compromised $\alpha + 1$ nodes, where α is the degree of the polynomial in any variable, then he can crack the complete system by using simple (Lagrange) interpolation. There was no known KPS that is both efficient and not prone to efficient attacks of multiple colluding (or compromised) nodes (see [2] for further references) until recently the HIMMO scheme solved this problem. This section reviews the operation of the HIMMO scheme that enables any pair of devices in a system to directly agree on a common symmetric-key based on their identifiers and a secret key generating polynomial as introduced in [12]. Like Blundo's scheme, HIMMO is based on symmetric polynomials, but it introduces new features to make simple interpolation attacks by colluding nodes infeasible. The underlying security principles on which HIMMO relies have been analyzed in [13] and [14]. Furthermore, this section describes two protocol extensions of the HIMMO scheme as described in [2].

We use the following notation: for each integer x and positive integer M, we denote by $\langle x\rangle_M$ the unique integer $y \in \{0,1,\ldots,M-1\}$ such that $x \equiv y \bmod M$.

3.1 HIMMO operation

Like any KPS, HIMMO requires a TTP, and three phases can be distinguished in its operation [10].

In the **setup phase**, the TTP selects positive integers B, b, m and α, where $m \geq 2$. The number B is the bit length of the identifiers that will be used in the system, while b denotes the bit length of the keys that will be generated. The TTP generates the public modulus N, an odd number of length exactly $(\alpha+1)B+b$ bits (so $2^{(\alpha+1)B+b-1} < N < 2^{(\alpha+1)B+b}$). It also randomly generates m distinct secret moduli $q_1,\ldots,q_m$ of the form $q_i = N - 2^b\beta_i$, where $0 \leq \beta_i < 2^B$ and at least one of $\beta_1,\ldots,\beta_m$ is odd. Finally, the TTP generates the secret root keying material, that consists of the coefficients of m bi-variate symmetric polynomials of degree at most α in each variable. For $1 \leq i \leq m$, the i-th root keying polynomial $R^{(i)}(x,y)$ is written as

$$R^{(i)}(x,y) = \sum_{j=0}^{\alpha}\sum_{k=0}^{\alpha} R^{(i)}_{j,k} x^j y^k$$

$$\text{with } 0 \leq R^{(i)}_{j,k} = R^{(i)}_{k,j} \leq q_i - 1.$$

In the **keying material extraction phase**, the TTP provides each node ξ in the system, with $0 \leq \xi < 2^B$, the coefficients of the key generating polynomial G_ξ:

$$G_\xi(y) = \sum_{k=0}^{\alpha} G_{\xi,k} y^k \tag{1}$$

where

$$G_{\xi,k} = \langle \sum_{i=1}^{m} \langle \sum_{j=0}^{\alpha} R^{(i)}_{j,k}\xi^j \rangle_{q_i} \rangle_N. \tag{2}$$

In the **key generation phase**, a node ξ wishing to communicate with node η with $0 \leq \eta < 2^B$, computes:

$$K_{\xi,\eta} = \big\langle \langle G_\xi(\eta)\rangle_N \big\rangle_{2^b} \tag{3}$$

It can be shown that $K_{\xi,\eta}$ and $K_{\eta,\xi}$ need not be equal. However, as shown in Theorem 1 in [2], for all identifiers ξ and η with $0 \leq \xi,\eta \leq 2^B$,

$$K_{\xi,\eta} \in \{\langle K_{\eta,\xi} + jN\rangle_{2^b} \mid 0 \leq |j| \leq 2m\}$$

In order to perform key reconciliation , i.e. to make sure that ξ and η use the same key to protect their future communications, the initiator of the key generation (say node

ξ) sends to the other node, simultaneously with an encrypted message, information on $K_{\xi,\eta}$ that enables node η to select $K_{\xi,\eta}$ from the candidate set $C = \{\langle K_{\eta,\xi} + jN\rangle_{2^b} \mid 0 \leq |j| \leq 2m\}$. No additional communication thus is required for key reconciliation. The key $K_{\xi,\eta}$ will be used for securing future communication between ξ and η. As an example of helper data used for key reconciliation, node ξ sends to node η the number $\sigma_{\xi,\eta} = \langle K_{\xi,\eta}\rangle_{2^s}$, where $s = \lceil \log_2(4m+1)\rceil$. Node η can efficiently obtain the integer j such that $|j| \leq 2m$ and $K_{\xi,\eta} \equiv K_{\eta,\xi} + jN \bmod 2^b$ by using that $jN \equiv K_{\xi,\eta} - K_{\eta,\xi} \equiv \sigma_{\xi,\eta} - K_{\eta,\xi} \bmod 2^s$. As N is odd, the latter equation allows for determination of j. As $\sigma_{\xi,\eta}$ reveals the s least significant bits of $K_{\xi,\eta}$, only the $b - s$ most significant bits $K_{\xi,\eta}$, that is, the number $\lfloor 2^{-s}K_{\xi,\eta}\rfloor$, should be used as key.

3.2 Implicit certification and verification of credentials

Implicit certification and verification of credentials is further enabled on top of the basic HIMMO scheme. A node that wants to register with the system provides the TTP with its credentials, e.g., device type, manufacturing date, etc. The TTP, which can also add further information to the node's credentials such as a unique node identifier or the issue date of the keying material and its expiration date, obtains the node's identity as $\xi = H(\mathit{credentials})$, where H is a public hash function. When a first node with identity ξ wants to securely send a message M to a second node with identity η, the following steps are taken.

- Step 1: Node ξ computes a common key $K_{\xi,\eta}$ with node η. It uses the computed common key to encrypt and authenticate its credentials and message M, say $e = E_{K_{\xi,\eta}}(\mathit{credentials}|M)$.
- Step 2: Node ξ sends $(\xi, e, \sigma_{\xi,\eta})$ to node η, where $\sigma_{\xi,\eta}$ is helper data helping node η to find $K_{\xi,\eta}$.
- Step 3: Node η receives $(\xi', e', \sigma'_{\xi,\eta})$. Using $\sigma'_{\xi,\eta}$, it computes its common key $K_{\eta,\xi'}$ with ξ' to decrypt e' obtaining the message M and verifying the authenticity of the received message. Furthermore, it checks whether the *credentials'* in e' correspond with ξ', that is, it validates if $\xi' = H(\mathit{credentials'})$.

This method not only allows for direct secure communication of message M, but also for implicit certification and verification of ξ's credentials because the key generating polynomial assigned to a node is linked to its credentials by means of H. If the output size of H is long enough, e.g., 256 bits, the input (i.e., the credentials) contains a unique node identifier, and if H is a secure one-way hash function, then it is infeasible for an attacker to find any other set of credentials leading to the same identity ξ. The fact that credential verification might be prone to birthday attacks motivates the choice for the relation between identifier and key sizes, namely, $B = 2b$. In this way, the scheme provides an equivalent security level for credential verification and key generation. The capability for credential verification enables e.g. the verification of the expiration date of the credentials (and the keying material) of a node, or verification of the access roles of the sender node ξ.

3.3 Enhancing privacy by using multiple TTPs

Using multiple TTPs was introduced by Matsumoto and Imai [10] for KPS and can also be elegantly supported by HIMMO [2]. In this scheme, a number of TTPs provide a node with keying materials linked to the node's identifier during the keying material extraction phase. Upon reception, the device combines the different keying materials by adding the coefficients of the key generating polynomials modulo N. Key generation is performed as usual. This scheme enjoys two interesting properties without increasing the resource requirements of the nodes. First, privacy is enhanced since a single TTP cannot eavesdrop the communication links. In fact, all TTPs should collude to monitor the communication links. Secondly, compromising a subset of TTPs does not break the overall system.

4 Design

This section describes the proposed security architecture focusing on the security goals presented in Section 2.4 and the operational goals described in Section 2.3. In the following subsections, we detail our solution according to the lifecycle of a smart object in which we show how keying materials are managed by means of a TTP infrastructure. This TTP infrastructure can be used to enable secure manufacturing of the smart objects (Section 4.1). The HIMMO keying material enables secure network access of a smart object in a typical smart city scenario (Section 4.2). Finally, this HIMMO keying material provides a way of ensuring secure operation and credential management during normal operation (Section 4.3).

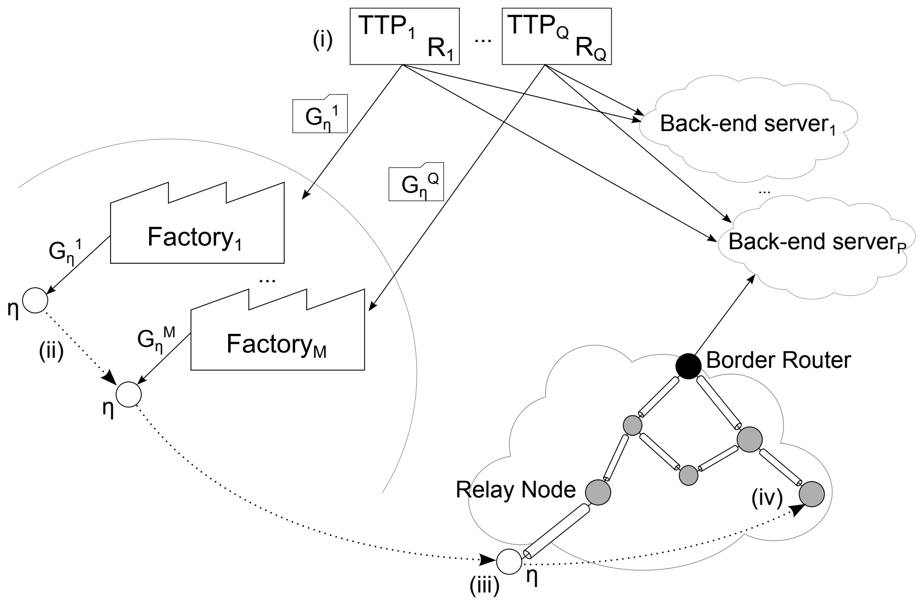

Fig. 1: Architecture overview.

Figure 1 depicts the components of our security architecture including: (i) an infrastructure of roots of trust in charge of handling the HIMMO root keying materials (Section 4.1). (ii) A number of factories producing smart objects configured with secret keys and credentials (Section 4.1). (iii) An authentication process, in which a smart object registers with a back-end system in order to get credentials for access to the network (Section 4.2). (iv) A secure operation phase in which smart objects can securely communicate with each other, using the credentials obtained in the previous step (Section 4.3).

4.1 TTP infrastructure and Smart object manufacturing

Our architecture relies on the HIMMO capability for working with multiple TTPs to address goals **S-1, S-2, S-3, S-4, and S-13**.

We consider multiple TTPs can generate and securely manage HIMMO root keying materials. The role of these TTPs is similar to today's CA infrastructure with the obvious differences in the underlying technology.

When a new set of devices needs to be manufactured, the back-end server in charge of those devices will request a subset of the TTPs to extract HIMMO keying materials linked to some device credentials. In the following we will assume that these credentials depend on the unique MAC address of each device η, e.g., $\eta =$MAC address of η. The back-end server will also determine which factories will get HIMMO keying materials from which TTPs. Next, TTP j will extract HIMMO keying material $G^j_\eta(y)$ for device η and securely send these keying materials to the corresponding manufacturing facility l. Here, each device η will receive $G^j_\eta(y)$ and update its locally aggregated keying material as $G_\eta(y) = \quad G_\eta(y) + G^j_\eta(y) \; _N$ In a similar way, the back-end server ξ can request at any time a new set of keying materials from those TTPs and obtain its aggregated keying material in a similar way.

4.2 The authentication process for network access

We use the HIMMO keying material to enable secure network access and device registration in a typical smart city scenario, addressing goals **S-5, S-6, S-7, S-8, S-9, S-10**. In this procedure, the joining node η aims to register with a back-end server after its verification and the back-end server aims to register the joining node after authenticating it. Although the server can verify the client, it is equally important that devices in the network can verify the authenticity of the joining node in order to prevent possible abuse that could lead to DoS attacks. The verification could be done at different levels in the network to thwart an attack as early as possible. The process is described below and the devices are organized in an IEEE 802.15.4 [6] mesh network, which is typical example of an IoT network.

- Step 1: A joining node η is installed and waits to hear beacons from devices in a network.
- Step 2: A neighboring node, working as relay node or a border router, regularly sends a broadcast beacon as part of the normal network traffic.
- Step 3: Upon reception of a network beacon originating, e.g., from a relay node r, node η will securely send a IEEE 802.15.4 message protected with the HIMMO key $K_{\eta,r} = \quad \langle G_\eta(r)\rangle_{N \;\; 2^b}$. The Key Identifier and Key Identifier Mode [6] can be used to indicate that HIMMO key derivation is in use.
- Step 4: The relay node will process this message protected at IEEE 802.15.4 and reply with the identity of the border router, if verification of the message sent by η is successful. $K_{\eta,r}$ could be cached by r for further communication.
- Step 5: Node η then will start a DTLS [8] handshake towards the back-end server through relay node r in which the communication at IEEE 802.15.4 is protected by $K_{\eta,r}$. Note that the relay node operates as described in [15]. Furthermore, the first DTLS message, Client Hello, includes the identity of the joining node and an authentication token created with $K_{\eta,br}$ so that the border router can verify it.
- Step 6: Node r will relay only DTLS traffic from device η. Furthermore, the border router will only forward the DTLS traffic if the verification of the authentication token is successful and the identity of the joining node is not blacklisted.

– Step 7: The back-end server gets the request from the joining node and engages in a DTLS handshake in which both joining node and back-end server mutually authenticate each other and verify their credentials. This handshake may be based on the lightweight scheme as proposed in [16]. Upon successful establishment of the DTLS session, the joining node will receive the network key, $K_{Network}$, from the server.

Upon successful completion of this protocol, the key used to secure IEEE 802.15.4 layer communication on the link between node η and any node ξ is computed as $K_{\eta,\xi}^{Network} = K_{\eta,\xi} \oplus K_{Network}$, where $\oplus$ denotes bit-wise XOR.

We note that the above protocol relies on the identity-based nature of HIMMO and its properties to mutually authenticate the joining device and back-end server. This is achieved by means of DTLS-HIMMO since HIMMO can be easily integrated in DTLS [16]. Furthermore, the above protocol uses HIMMO identities to allow for early detection and prevention of DoS attacks at relaying devices and at the border router.

4.3 Secure operation

Secure operation aims at fulfilling security goals **S-3, S-5, S-6, S-8, S-9, S-10, S-13** and all operational and performance objectives.

Firstly, we illustrate how HIMMO ensures a fully collusion resistant mesh network based on IEEE 802.15.4. IEEE 802.15.4 supports the usage of pairwise keys between devices, however, key agreement is left out of the standard. Therefore, in practice, most standards rely on a network or system-wide key so that if a single device is captured, then the whole network or system breaks down. Furthermore, compromised devices cannot be easily identified since an attacker can fake any identity so that securely updating such a network or system wide key is infeasible (in addition to being very costly from an operational point of view). In our security architecture, secure operation in a IEEE 802.15.4 network works as follows:

– Step 1: Note ξ will advertise its presence by means of unsecured broadcast beacon.
– Step 2: Node η will obtain a common HIMMO key with device η as $K_{\eta,\xi} = \langle G_\eta(\xi)\rangle_{N\ 2^b}$. Node η will also obtain helper data $\sigma_{\eta,\xi}$. If node η has joined the network, then it will use pairwise key $K_{\eta,\xi}^{Network} = K_{\eta,\xi} \oplus K_{Network}$. If node η has not joined the network yet, then it will use pairwise key $K_{\eta,\xi}$.
– Step 3: Node η sends a secure IEEE 802.15.4 message to ξ protected with the pairwise key determined in the previous step. This message contains the MAC address of the sender (η) and $\sigma_{\eta,\xi}$ in the Key Identifier field of the 802.15.4 security header.
– Step 4: Node ξ receives the secure message from η. It generates the pairwise HIMMO key with the MAC address of η as $K_{\xi,\eta} = \langle G_\xi(\eta)\rangle_{N\ 2^b}$ and recovers $K_{\eta,\xi}$ from $K_{\xi,\eta}$ and $\sigma_{\eta,\xi}$. Assuming that node ξ has already joined the network, then ξ knows $K_{Network}$ so that it can compute $K_{\eta,\xi}^{Network}$ and process the incoming message with both $K_{\eta,\xi}$ and $K_{\eta,\xi}^{Network}$. If the verification of the message with $K_{\eta,\xi}$ is successful, then node η is considered a joining device and only DTLS traffic for network access is allowed. Alternatively, the message is verified with $K_{\eta,\xi}^{Network}$ so that the device is identified as part of the network and normal routing and data traffic is enabled.

This solution enables pairwise keys between devices based on HIMMO so that the IEEE 802.15.4 network becomes fully collusion resistant to the physical capture of devices. Note that this represents a huge improvement with the state-of-the-art in which most deployments rely on a single network-wide or even system-wide key that represents a single point of failure. Furthermore, compromised devices can be identified since the HIMMO keying material is cryptographically bound to the device identifier and in this case it is chosen to be the MAC address of the device. This means that captured devices can be blacklisted. Finally, this method allows us to easily differentiate devices that have joined the network from devices that have not done so yet. This differentiation, however, could also be determined by indicating which key is used to protect the packet by using the Key Index field of the Key Identifier in the IEEE 802.15.4 Auxiliary Security Header (ASH) [6]. This would avoid verification of the packet with two keys in Step 4. The Key Index field (or a part of it) will indicate to node ξ which key, $K_{\eta,\xi}$ or $K_{\eta,\xi}^{Network}$, was used to secure the packet.

We further note that HIMMO cannot only be used at MAC layer to generate pairwise keys between neighboring devices, but it can also be used to enable secure communication between any pair of devices in the network/system at application layer by means of DTLS [16]. This is similar to the process described in previous section with the difference that the communication is not between the joining device and the back-end server, but between two arbitrary devices. As a consequence any pair of devices can securely agree on a common symmetric key used in DTLS and verify any other set of credentials. Performance benefits will become clear in the next section.

5 Implementation and evaluation

The design described in Section 4 addresses the needs of real-world environments as Smart Santander [3]. This section provides performance measures for the building blocks that comprise this architecture.

According to its different functionalities and resource requirements, we divide the HIMMO implementation into two modules: one that provides TTP capabilities and would likely be implemented in a server, and another one that provides node functionalities and would likely be implemented in a device.

The TTP module allows choosing the HIMMO public parameters described in Section 3.1. It also generates the keying materials for the network devices tailored to the word-length and architecture of each device for optimal performance. The TTP module is implemented in Java and can be easily integrated in a web service.

For the node module implementation, it is worth noting that HIMMO has been designed to enable very efficient performance in constrained devices. From Equation (3) we observe that obtaining a symmetric key just requires the evaluation of a polynomial of degree α modulo N and taking the b least significant bits. This means that only $\alpha + 1$ modular multiplications are required to compute the key. In each multiplication, the B bit identifier multiplies the $(\alpha + 1)B + b$ bit coefficient and the result is reduced modulo N. These modular operations can be implemented in a very efficient manner for appropriate choices for N, e.g., $N = 2^{(\alpha+1)B+b} - 1$. Many implementations are possible for the node module because of the heterogeneous nature of the devices. Specialized implementations will optimize performance, sacrificing portability. We have both developed a generic and portable library that can be adapted to work with different devices and also optimized the algorithm for a very resource constrained device.

In order to evaluate the performance of the HIMMO scheme, we have implemented it on three different devices: a very resource-constrained 8-bit CPU ATMEGA128L running at 8 MHz, the 32-bit NXP LPC1769 LPCXpresso Board running at 120 MHz, and an Intel i3 3120M (64-bit), at 2.50 GHz running Xubuntu 14.04. The implementations for the NXP LPC1769 and the Intel i3 3120M are based on our generic C library, including the big integer arithmetic required for key generation. The implementation for the ATMEGA128L, on the other hand, is optimized in assembler. This show both the possibilities of the generic C library and the optimization power of a specialized implementation.

For very resource constrained devices, specialized implementations are preferred: our optimized implementation for the ATMEGA128L fits in just 428 bytes of flash memory. This shows that HIMMO can fit even in very resource constrained devices or when integrated as part of the IEEE 802.15.4 logic described in the above section. We also note that the RAM consumption is linear with α since we have to keep in memory a term that is $(\alpha+1)B+b$ bits. Table 1 and Table 2 provide a brief summary of the performance of the HIMMO scheme implemented in the above CPUs.

Table 1: HIMMO performance for $B = b = 128$ as a function of α.

	α	26	34	40	50
	Keying material size (KB)	6.90	11.18	15.07	22.83
CPU time (msec)	ATMEGA128L (8-bit @ 8 MHz)	223	367	497	743
	NXP LPC1769 (32-bit @ 120 MHz)	7.46	11.82	15.74	23.48
	Intel i3 3120M (64-bit @ 2.5 GHz)	0.034	0.053	0.069	0.103

Table 2: HIMMO performance for $\alpha = 26$ as a function of $b = B$.

	$b = B$	64	128	192	256
	Keying material size (KB)	3.45	6.90	10.34	13.79
CPU time (msec)	ATMEGA128L (8-bit @ 8 MHz)	63	223	393	632
	NXP LPC1769 (32-bit @ 120 MHz)	2.55	7.46	14.93	25.16
	Intel i3 3120M (64-bit @ 2.5 GHz)	0.015	0.034	0.062	0.100

The above node module can be used to enable the functionality described in the design section. In particular, it can be easily integrated with IEEE 802.15.4 ensuring a fully collusion resistant network during operation. It can also be used to enable secure network access and registration by means of DTLS-HIMMO, achieving functionalities such as mutual authentication and verification, which today are only possible with public-key cryptography, at the speed and memory requirements of symmetric cryptography. In this context, the left part of Figure 2 shows the time elapsed to successfully complete a DTLS handshake in different modes, we can see that HIMMO with mutual verification is almost 7 times faster than mutual verification with the

ECC alternative. On the right, we can see the ratio between exchanged data and payload: for a secure interchange of 1 KB this ratio is around 40 % smaller with HIMMO compared with ECC both providing mutual authentication. More detailed performance measures, as well as a complete description of this implementation are available on [16].

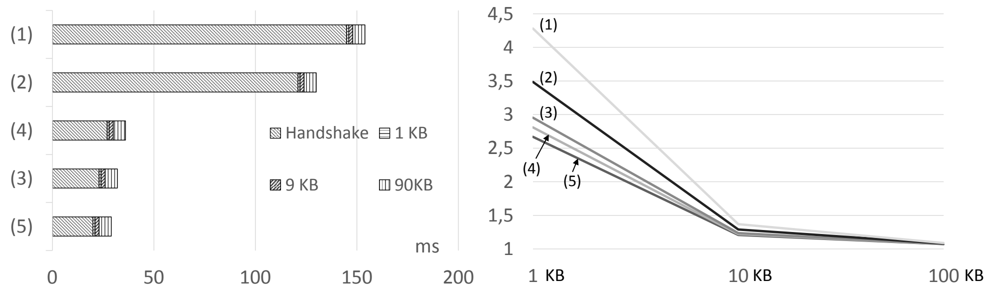

Fig. 2: (1) ECDH-ECDSA with mutual authentication, (2) ECDH-ECDSA with server authentication, (3) HIMMO with mutual verification of client's and server's credentials ($t = 5, B = 256, b = 32, \alpha = 17$), (4) HIMMO with mutual authentication ($t = 5, B = 32, b = 32, \alpha = 50$) and (6) PSK.

6 Discussion and comparison with related work

This section discuses how the proposed security architecture fulfills the goals described in Sections 2.4 and 2.3, and compares our solution with approaches based on symmetric cryptography (Pre-Shared Key (PSK)) and on asymmetric cryptography (Public Key Cryptography (PKC)).

A PSK is lightweight and very efficient but, as we can see in Table 3, it is unable to support most of the security goals identified in Section 2.4. Such a solution is not scalable. We might wish to share the same key with a large number of devices.This is the current approach in IEEE 802.15.4 as it does not require any additional infrastructure. However, this approach leads to a single point of failure —if the key gets compromised, the security of all the network will be broken— which is a really important problem in networks where not even the physical integrity of the devices can be guaranteed. This approach also does not enable for identification, authentication, or authorization of devices. If on the other hand many (pairwise) keys are distributed, this involves huge amounts of memory, makes credential management difficult, and the addition of new devices to the network becomes more complex.

PKC solves most security goals, but is much more resource-consuming than PSK or HIMMO in terms of required CPU time, memory, bandwidth, and round-trips (e.g., see fig:DTLSPerformance). In a similar way, in order to comply with security goal **S-1** —resiliency to root of trust compromise— it would be necessary to have a chain of CA signing the certificates; this will increase the certificate length for each signing, which will affect both memory usage and network overhead. As one of the main sources of energy consumption in an IoT network is the wireless radio, the battery life would also be affected. In contrast, HIMMO can achieve this property without performance penalty. The integration with OMA LWM2M would be easy, as it supports DTLS, but integration with IEEE 802.15.4 without protocol changes is not possible. Furthermore, it would imply a serious degradation in the network performance. Similarly, it could be possible to use certificates to design a joining protocol which would prevent DoS attacks, but it would result in a great increase in network overhead because of the certificate exchanges. Existing PKC solutions (e.g., all PKC-based cipher-suites in (D)TLS) would break down if a quantum computer is built.

HIMMO and PSK do no not offer perfect foward secrecy (**S-12**); the $*$ in the row for (**S-12**) means that PKC does offer this feature for some cipher suites, e.g. those based on DHE or ECDHE key exchange. Being symmetric schemes, HIMMO and PSK do not offer non-repudiation (**S-13**): a device A can sign data with the key it uses to communicate with device B, and claim that the data has been signed by B using the latter's key for communicating with A. PKC does offer non-repudiation, as each party has its own public–private key pair.

Table 3: Comparison between different security architectures. Relative performance from most efficient (*) to least (***).

Design goals	HIMMO	PKC	PSK
O-1: Performance	*	***	*
O-2: Easy device addition to a running system	✓	✓	✓
O-3: Scalable	✓	✓	-
O-4: Easy credential management	✓	✓	-
O-5: Easy integration with existing protocols	✓	-	✓
O-6: Fits device lifecycle	✓	✓	-
O-7: Long term security	✓	✓	✓
S-1: Resilient to root of trust compromise	✓	✓	-
S-2: Single root of trust cannot monitor	✓	✓	-
S-3: Key escrow	✓	-	✓
S-4: Facilitates secure manufacturing	✓	✓	-
S-5: Device authentication and authorization	✓	✓	-
S-6: Back-end authentication and authorization	✓	✓	-
S-7: Prevents DoS attacks	✓	✓	-
S-8: Fully collusion resistance	✓	✓	-
S-9: Device identification and blacklisting	✓	✓	-
S-10: Key agreement	✓	✓	✓
S-11: Post-quantum resilience	✓	-	✓
S-12: Perfect forward secrecy	-	✓*	-
S-13: Non-repudiation	-	✓	-

7 Conclusions

The IoT is an emerging area involving many connected smart devices that need to be secured during their entire lifecycle. This problem is not solved with current security solutions: asymmetric-key cryptography is computationally hungry, difficult to integrate with protocols, and not suitable for long term deployments, while the symmetric-key option is extremely efficient, but does not scale and offers limited security properties.

We have reviewed the operational and security requirements of a real-world IoT scenario and build on top of them a comprehensive and lightweight security architecture based on HIMMO. Thus, our security architecture is tailored to the requirements of IoT, its attack vectors, and resource requirements.

Our solution builds on HIMMO's excellent performance —a combined HIMMO-based key agreement and credential verification handshake can be done in a few hundred milliseconds even on very resource-constrained devices— to create a scalable, operationally friendly and secure architecture.

HIMMO can be easily integrated in modern protocols like IEEE 802.15.4 or DTLS, providing, at the relatively low costs of symmetric cryptographic solutions, features that before were only feasible with asymmetric cryptography. Some of the security features enabled by HIMMO are full collusion resistance, device and back-end authentication and verification, pairwise key agreement, support for multiple TTPs and key escrow, or protection against DoS attacks.

Because our architecture combines great security features with low resource needs and allows easy integration in modern protocols, we believe that it is a very competitive approach to secure the IoT and related emerging areas.

References

1. Robert Pepper. The Internet of Things is Now: M2M Devices Forecast 2013–2018. IIC Annual Conference, 2014, `http://www.iicom.org`.
2. O. García-Morchón D. Gómez-Pérez, J. Gutiérrez, R. Rietman, B. Schoenmakers, and L. Tolhuizen. HIMMO - A Lightweight, Fully Colluison Resistant Key-Predistribution Scheme. Cryptology ePrint Archive, Report 2014/698, 2014. `http://eprint.iacr.org/`.
3. Luis Sanchez, Jose A Galache, Veronica Gutierrez, Jose M Hernández, Jesús Bernat, Alex Gluhak, and Tomás García. Smartsantander: The meeting point between future internet research and experimentation and the smart cities. In *Future Network &amp; Mobile Summit (FutureNetw), 2011*, pages 1–8. IEEE, 2011.
4. Oscar Garcia-Morchon, Sandeep Kumar, Sye Keoh, Rene Hummen, and Rene Struik. Security considerations in the ip-based internet of things. Internet-Draft draft-garcia-core-security-06, IETF Secretariat, September 2013. `http://www.ietf.org/internet-drafts/draft-garcia-core-security-06.txt`.
5. N. Kushalnagar, G. Montenegro, and C. Schumacher. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. RFC 4919 (Informational), August 2007.
6. IEEE Computer Society. IEEE Standard for Local and metropolitan area networks - Part 15.4 2011 revision: Low-Rate Wireless Personal Area Networks (LR-WPANs), September 2011.
7. Z. Shelby, K. Hartke, and C. Bormann. The Constrained Application Protocol (CoAP). RFC 7252 (Proposed Standard), June 2014.
8. E. Rescorla and N. Modadugu. Datagram Transport Layer Security Version 1.2. RFC 6347 (Proposed Standard), January 2012.

9. JR Prins and Business Unit Cybercrime. DigiNotar Certificate Authority breach *Operation Black Tulip*, 2011.
10. T. Matsumoto and H. Imai. On the key predistribution system: a practical solution to the key distribution problem. In C. Pomerance, editor, *Advances in Cryptology – CRYPTO'87*, LNCS 293, pages 185–193. Springer, 1988.
11. C. Blundo, A. de Santis, A.Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly secure key distribution for dynamic conferences. *Information and Computation*, 146:1–23, 1998.
12. O. Garcia-Morchon, L. Tolhuizen, D. Gomez, and J. Gutierrez. Towards full collusion resistant ID-based establishment of pairwise keys. In *Extended abstracts of the third Workshop on Mathematical Cryptology (WMC 2012) and the third international conference on Symbolic Computation and Cryptography (SCC 2012).*, pages 30–36, 2012.
13. O. García-Morchón, D. Gómez-Pérez, J. Gutiérrez, R. Rietman, and L. Tolhuizen. The MMO problem. In *Proc. ISSAC'14*, pages 186–193. ACM, 2014.
14. O. García-Morchon, R. Rietman, I.E. Shparlinski, and L. Tolhuizen. Interpolation and approximation of polynomials in finite fields over a short interval from noisy values. *Experimental mathematics*, 23:241–260, 2014.
15. Sandeep Kumar, Sye Keoh, and Oscar Garcia-Morchon. DTLS Relay for Constrained Environments. Internet-Draft draft-kumar-dice-dtls-relay-02, IETF Secretariat, October 2014. `http://www.ietf.org/internet-drafts/draft-kumar-dice-dtls-relay-02.txt`.
16. Oscar Garcia-Morchon, Ronald Rietman, Sahil Sharma, Ludo Tolhuizen, and Jose Luis Torre-Arce. DTLS-HIMMO: Efficiently Securing a Post-Quantum World with a Fully-Collusion Resistant KPS. *Accepted for publication at ESORICS*, 2015. `https://eprint.iacr.org/2014/1008`.