Securing the IP-based Internet of Things with HIP and DTLS

Oscar Garcia-Morchon, Sye-Loong Keoh, Sandeep S. Kumar, Pedro Moreno-Sanchez
Francisco Vidal-Meca, and Jan Henrik Ziegeldorf
Philips Research Europe, High Tech Campus 34
Eindhoven, The Netherlands
{oscar.garcia, sye.loong.keoh, sandeep.kumar, henrik.ziegeldorf}@philips.com
p.morenosanchez@um.es, francisco.vidal.meca@rwth-aachen.de

ABSTRACT

The IP-based Internet of Things (IoT) refers to the pervasive interaction of smart devices and people enabling new applications by means of new IP protocols such as 6LoWPAN and CoAP. Security is a must, and for that we need a secure architecture in which all device interactions are protected, from joining an IoT network to the secure management of keying materials. However, this is challenging because existing IP security protocols do not offer all required functionalities and typical Internet solutions do not lead to the best performance.

We propose and compare two security architectures providing secure network access, key management and secure communication. The first solution relies on a new variant of the Host Identity Protocol (HIP) based on pre-shared keys (PSK), while the second solution is based on the standard Datagram Transport Layer Security (DTLS). Our evaluation shows that although the HIP solution performs better, the currently limited usage of HIP poses severe limitations. The DTLS architecture allows for easier interaction and interoperability with the Internet, but optimizations are needed due to its performance issues.

Categories and Subject Descriptors

C.2.1 [**Computer-Communications Networks**]: Network Architecture and Design—*network communications*

General Terms

Security; Design; Management; Performance

1. INTRODUCTION

The IP-based Internet of Things (IoT) will enable smart and mobile devices equipped with sensing, acting, and wireless communication capabilities to interact and cooperate with each other in a pervasive way by means of IP connectivity. IP protocols play a key role in this vision since they allow for end-to-end connectivity using standard protocols ensuring that different smart devices can easily communicate with each other in an inexpensive way. Protocols such as IPv6, TCP and HTTP that are commonly used in traditional networks will be complemented by IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) and the Constrained Application Protocol (CoAP) currently in development in IETF. This allows smart and mobile devices used for healthcare monitoring, industrial automation and smart cities to be seamlessly connected to the Internet.

Security and privacy are mandatory requirements for the IP-based IoT in order to ensure its acceptance. The interaction between devices must be regulated during the whole device lifecycle. When a device joins an IoT network, the IoT network has to authorize its joining, such that it is provisioned and configured with the corresponding operational parameters, thus providing *network access*. During operation, devices will interact with each other requiring pairwise session keys, and for that, *key management* is needed.

These security functionalities are the focus of our research, because of two reasons: (i) no standard solution exists yet and the traditional Internet relies on too many different security protocols that are not going to fit on small devices; (ii) device mobility and system scalability are to be considered.

To overcome these issues we propose two security architectures for the IP-based IoT by adapting existing IP security protocols while relying on a single protocol. The first solution is a new variant of the Host Identity Protocol (HIP) that uses pre-shared keys (PSK) while the second solution is based on Datagram Transport Layer Security (DTLS). Beyond the handshakes themselves, we explain how to use them to allow for secure network access, flexible key management and secure communication. We use either the HIP or the DTLS handshake for network access. For key management we integrate both approaches with the Adapted Multimedia Internet KEYing (AMIKEY) protocol and use a polynomial scheme for efficient key management and generation of pairwise keys. With this, our goal is not only to secure a device during its lifecycle but also to analyze which protocol-based architecture would be the best one: the first solution is more efficient but less standard, the second one makes interaction with the Internet easier but performs worse.

The paper is organized as follows. Section 2 overviews the related work and relevant IP protocols. Section 3 details the application scenario and our design goals. In Section 4, we describe our two security architectures that are evaluated in Section 5. Section 6 concludes this paper.

WiSec'13, April 17-19, 2013, Budapest, Hungary.

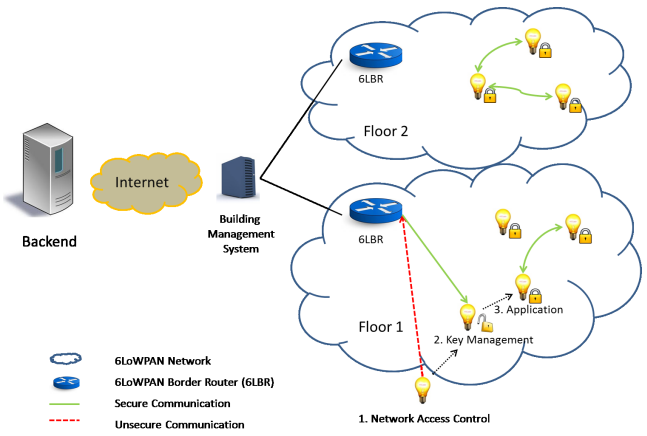


Figure 1: Building Management System Scenario

2. BACKGROUND

Research into security for wireless sensor networks has produced many results. SPINS [7] is a centralized architecture for securing uni- and multicast communication. Sizzle [12], for first time, made feasible the establishment of a secure end-to-end connection to a resource constrained device over the Internet based on HTTPS and Elliptic Curve Cryptography. Liu et al. [14] used polynomial schemes [9] to simplify key agreement in distributed sensor networks. Each node η is assigned a polynomial share $F(\eta, y)$ derived from a secret symmetric bivariate polynomial $F(x, y)$ allowing any pair of devices to establish a common key $F(\eta, \eta') = F(\eta', \eta)$. These solutions are, however, non-standard while the IoT vision requires efficient and standardized solutions. In our work, we extend standard protocols, namely HIP, DTLS, and AMIKEY, to create a secure IP-based IoT.

Host Identity Protocol (HIP) [11] introduces a cryptographic namespace of stable host identities (HIs) between the network and transport layer. Besides enabling mobility and multi-homing, the cryptographic HIs allow for a mutually authenticated Diffie-Hellman key exchange in which the generated symmetric-key usually protects an IPSec association. There are two HIP handshakes: *HIP Base EXchange (BEX)* relies heavily on public-key cryptography while *HIP Diet EXchange (DEX)* [15] defines a lightweight alternative that only relies on a static Diffie-Hellman key exchange being more suitable for constrained devices.

Datagram Transport Layer Security (DTLS) [17] is a datagram-compatible adaptation of TLS that runs on top of UDP. DTLS uses similar messages as defined in TLS including the DTLS handshake to establish a secure unicast link and the DTLS record layer to protect this link. The *DTLS handshake* supports different types of authentication mechanisms, e.g., using a pre-shared key, public-key certificates, and raw public-keys. DTLS is the mandatory standard for protection of CoAP [13] messages.

Adapted Multimedia KEYing (AMIKEY) [1] manages keying materials for securing communications within constrained networks and devices. It is based on MIKEY, a Key Management Protocol intended for use in real-time applications [10]. For this purpose, AMIKEY provides different message exchanges that may be transported directly over UDP and TCP. Essentially, they can be integrated within other protocols, such as HIP and DTLS.

3. SCENARIO AND DESIGN GOALS

Our work targets an *IoT network* running CoAP over 6LoWPAN. Devices in the *IoT network* are mobile or stationary and exhibit tight processing, memory and bandwidth constraints. The *IoT network* is connected to the public Internet through a number of 6LoWPAN border routers (6LBR). Further, we consider a centrally managed scenario in which the devices and services in each *IoT network* are managed by a *domain manager* that is located either within the *IoT network* or in the public Internet. The *domain manager* along with the *IoT network*s it manages is denoted as the *IoT domain*. Figure 1 shows an example of such a scenario – a smart building. Smart devices within the building (e.g., window blinds, lighting devices) form several IoT networks connected to a remote building management system via 6LBR.

This paper has two goals: First, describe how to secure the above scenario during the lifecycle of a device while minimizing the number of (and size of) protocols. Second, discuss what the best approach might be, one based on a more standard protocol such as DTLS or one relying on a less standard protocol combination but with better performance.

In our designs, we assume the Internet Threat Model [16] in which a malicious adversary can read and forge network traffic between devices at any point during transmission, but assume that devices themselves are secure. We consider a simplified lifecycle including three phases that currently lack a standardized (and unified) solution for IoT networks.

- *Secure Network Access* is about how to securely associate a new device to an *IoT network*. An attacker can perform network attacks during this phase, e.g., flooding or using the network for other purposes. The network can also be attacked if secure network access is not present, and thus, only devices that have been authenticated and authorized through a secure network access process should be allowed to communicate within the network. Similarly, devices that leave the IoT network should not be able to access the network with previous access parameters.

- *Key Management* is about how to handle the keys in an *IoT network* used for different purposes. Secure key derivation and management to secure interactions in the IoT domain is required, i.e., different pairwise, group, and network keys. In this way, compromising session keys or a previous derived key will not enable the attackers to obtain information about the currently used keying materials.

- *Secure Communication* is required since the adversary can eavesdrop on traffic in the IoT networkand maliciously modify it. Two communicating parties must establish a pairwise key to assure the confidentiality, integrity and authenticity of the information exchanged.

Due to the resource constrains in IoT devices, our proposed solutions are based on the assumption that a device has been configured with a PSK that is known *a priori* to the domain manager of the IoT domain it wants to join. This assumption is reasonable since a PSK could be embedded and registered during the manufacturing process of a device and the domain manager can retrieve it from a central server. Our solutions can be adapted to work with public-key cryptography when devices become more powerful.

4. DESIGN

In this section, we detail the design of our two solutions to address (i) secure network access, (ii) key management, and (iii) secure communication. The first approach relies on HIP, while the second one uses DTLS. Both of them address the three problems in three corresponding phases. In the following, we first provide a short sketch of the whole system in Section 4.1. Next, we explain each phase individually for the two proposed architectures, including their similarities and differences[1].

4.1 Overview

The first phase accounts for *secure network access*. In our architecture, the network is protected at link layer by means of a symmetric-key (L2 key), which is unknown to the joining device *a priori*. Using its link local address, the joining device authenticates itself to the domain manager of the IoT domain by means of an initial handshake (HIP or DTLS) that is based on a PSK. The PSK is assumed to have been pre-configured in the device (cf. Section 3). On success, the domain manager issues access parameters (L2 key) that would allow the joining device to access the secured IoT network and to receive a routable IPv6 address.

The second phase deals with *key management*. The joining device is provided with keying material to interact with other devices. For pairwise key generation, we make use of the polynomial scheme [9] described in Section 2. Each joining device is issued an individual share of a secret bivariate polynomial by the domain manager along with the access parameters during the network access phase. Using their individual shares, any two devices can then derive pairwise keys based on their identities in a fast and efficient way. In the HIP solution, keys are not used directly, instead they serve as root keying material in the MIKEY key derivation mechanism in order to derive fresh purpose-specific session keys for any pair of devices in the IoT network.

The final phase, *secure communication*, is achieved by protecting the exchange of CoAP messages by means of the DTLS record layer. Keys derived in the key management phase are used to protect the communication links.

4.2 Secure Network Access

We describe the details of the initial HIP and DTLS based handshakes.

4.2.1 HIP based approach

HIP-BEX or DEX as previously described in Section 2 could be used as handshakes in the secure network access phase. However, public-key cryptography is considered to be too expensive for constrained devices. We therefore propose a new variant called HIP-PSK which is more lightweight than HIP-DEX and HIP-BEX.

Figure 2 illustrates the HIP-PSK handshake. It is based on HIP-DEX but removes the ECC Diffie-Hellman method and performs a challenge-response authentication protocol [8] based on a PSK. Thus, instead of computing a static Diffie-Hellman key, the session key is derived by using CMAC [3] as

[1]The designs presented in this paper do not include the capability of secure network access in multi-hop networks and secure multicast due to the space limitation. The whole design will be presented in the future. The performance evaluation section shows the results for the prototypes including all functionalities, i.e., also multi-hop and multicast.

$$
\begin{aligned}
D \longrightarrow DM(I1) &: HIT_D, HIT_{DM}, DH_{Grouplist} \\
DM \longrightarrow D(R1) &: HIT_{DM}, HIT_D, Puzzle, DH_{Grouplist} \\
&\quad D\ solves\ Puzzle\ and\ computes\ K_s\ . \\
&\quad K_s = (PSK|puzzle|solution) \\
D \longrightarrow DM(I2) &: HIT_D, HIT_{DM}, Solution, MIC \\
&\quad DM\ checks\ Solution\ , computes\ K_s\ and \\
&\quad checks\ MIC\ correctness \\
DM \longrightarrow D(R2) &: HIT_{DM}, HIT_D, DH_{Grouplist}, MIC \\
&\quad D\ checks\ MIC\ correctness.
\end{aligned}
$$

Figure 2: HIP-PSK handshake between a joining device (D) and the domain manager (DM)

$K_s = CMAC(PSK|puzzle|solution)$ with the PSK as the secret input and the puzzle and its solution as nonces guaranteeing freshness. As in HIP-DEX, mutual authentication is achieved by generating and verifying a Message Integrity Code (MIC) on the I2 and R2 messages using CBC-MAC. HIP-PSK also preserves the DoS protection offered by HIP-DEX through the puzzle mechanism.

As in HIP-DEX, HIP-PSK identifies a device by its Host Identity Tag (HIT) computed as a truncation of its Host Identity (HI). The HI is just a random bit string. Alternatively, in the future, a public key could be used for interoperability purposes with HIP-BEX and HIP-DEX.

The details for bootstrapping a new node into the IoT network are the following: The joining device (D), performs the HIP-PSK handshake with the domain manager (DM). HIP-PSK mutually authenticates D and DM and creates a session key K_s. DM then verifies whether D is authorized to join the network. When authorized, DM uses D's HIT HIT_D to generate a polynomial share $F(HIT_D, y)$ from the secret bivariate polynomial $F(x, y)$. DM uses the session key K_s to encrypt and issue the L2 key and polynomial share to D in an additional HIP Update message. D acknowledges the receipt and can now join the secured IoT network. These two last messages are not depicted in Figure 2.

4.2.2 DTLS based approach

In this approach, the DTLS handshake protocol is used instead of HIP-PSK during the secure network access phase. Similar to the HIP approach, our design uses DTLS-PSK [5] instead of public-key based DTLS handshake, because it incurs less overhead and reduces the number of exchanged messages.

The joining device can be authenticated with the domain manager by performing the DTLS-PSK handshake relying on the link-local address. Once the device has been authenticated and authorized for the network, the established DTLS secure channel between the domain manager and the joining device is used to issue the L2 key and polynomial share to the device. At the time of the initial handshake, the joining device has not received an IP address. The domain manager thus generates a polynomial share $F(ID_{D1}, y)$ for the joining device that is bound to a given identifier ID_{D1}. This identifier will be used as the PSK hint in DTLS-PSK [5]. DTLS-PSK provides resiliency against DoS attacks through a cookie mechanism.

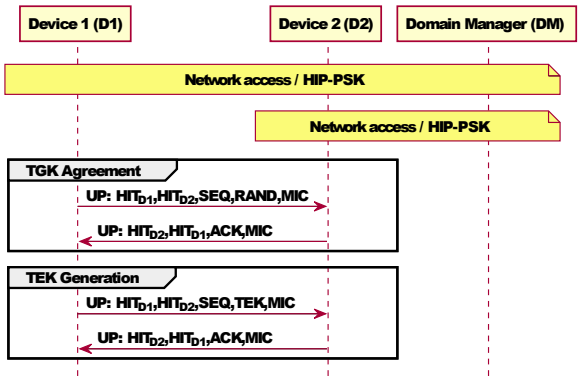


Figure 3: Management of unicast keys in HIP-based approach including TGK and TEK generation

4.3 Key Management

This section describes the details of key management in the HIP and DTLS-based approaches based on the use of the AMIKEY crypto-session bundle (CSB). A CSB is built from some root keying material (the TEK Generation Key (TGK)) and random bits (*RAND*). AMIKEY then defines a lightweight mechanism for the derivation and management of fresh purpose-specific keys, called Traffic Encryption Keys (TEKs), that are used to secure the communication links. We now explain how the root keying material, i.e. the TGK, is obtained, how the CSB is negotiated and set up, and finally how TEKs are requested and generated.

4.3.1 HIP based approach

As described in Section 4.2, each joining device has received its polynomial share linked to its HIT from the domain manager after a successful handshake during the network access phase. Our purpose is to define a method to derive keys for applications, e.g., a CoAP application on a given port. Figure 3 shows how a CSB is set up using root keying material derived from the polynomial share and how fresh TEKs are generated to secure the interactions between a pair of devices *D1* and *D2* in an IoT network.

1. *D1* and *D2* generate a pairwise key $K_{(D1,D2)}$ by using their polynomial shares and their respective identities: $K_{(D1,D2)} = F(HIT_{D1}, HIT_{D2}) = F(HIT_{D2}, HIT_{D1})$. $K_{(D1,D2)}$ is used as the TGK for the CSB.

2. *D1* and *D2* further need to agree on a random value *RAND*. *D1* provides the random value in a HIP Update message that is protected with a MIC computed with $K_{(D1,D2)}$ using CBC-MAC.

3. *D2* acknowledges the Update.

If the target application requires mutual authentication, the HIP-PSK handshake (based on $K_{(D1,D2)}$) should be executed between D1 and D2 first using the resulting session key of the HIP handshake as the TGK for the CSB. The *D1* and *D2* have now a CSB from which fresh TEKs can be derived upon request in a simple two-step AMIKEY process that we embed in the HIP Update mechanism.

4. *D1* sends a HIP Update message with the *TEK trigger parameter* to *D2*. The trigger contains the identifier associated with the new TEK, the desired length and the CSB from which to generate the key.

5. *D2* generates the new TEK using the TGK and given parameters as inputs to AES-CMAC as specified in AMIKEY/MIKEY [1, 10] and acknowledges the request.

4.3.2 DTLS based approach

As DTLS runs on the transport layer, it can be used directly to protect the applications. The polynomial shares are also used here to provide for fast pairwise key agreement between any pair of devices in the IoT domain. This pairwise key serves as the PSK in DTLS-PSK [5] enabling any two applications running on the devices to derive a session key, equivalent to the TEK in the HIP-approach. In detail:

1. Two applications running on the devices *D1* and *D2* start a DTLS-PSK handshake. They exchange their identities ID_{D1} and ID_{D2} as extensions to the first two handshake messages, the *ClientHello* and *ServerHello*.

2. Both devices then generate a pairwise key by using their polynomial shares and their respective identities: $K_{(D1,D2)} = F(ID_{D1}, ID_{D2}) = F(ID_{D2}, ID_{D1})$.

3. The derived key $K_{(D1,D2)}$ is used as the PSK to complete the DTLS-PSK handshake.

4. The result of the DTLS-PSK handshake is a session key (equivalent to a TEK) used to protect the communication link between the two applications on both devices.

4.4 Secure Communication

Once the pairwise session keys have been derived, a secure channel can be created to transport data (CoAP messages) between the devices in the IoT network. For this, we rely on the DTLS record-layer to create a secure transport layer for CoAP.

Any pair of devices in the IoT domain that wish to communicate with each other, establish a CSB and derive a fresh unicast TEK either through HIP or DTLS as described in Section 4.3. The TEK is used in the DTLS record layer (based on AES-CCM) to protect the message exchange between two applications. AES-CCM is an AES mode of operation that defines the use of AES-CBC for MAC generation with AES-CTR for encryption [4]. The CCM counter (corresponding to the DTLS epoch and sequence number fields) are initialized to 0 upon TEK establishment and used in the nonce construction in a standard way.

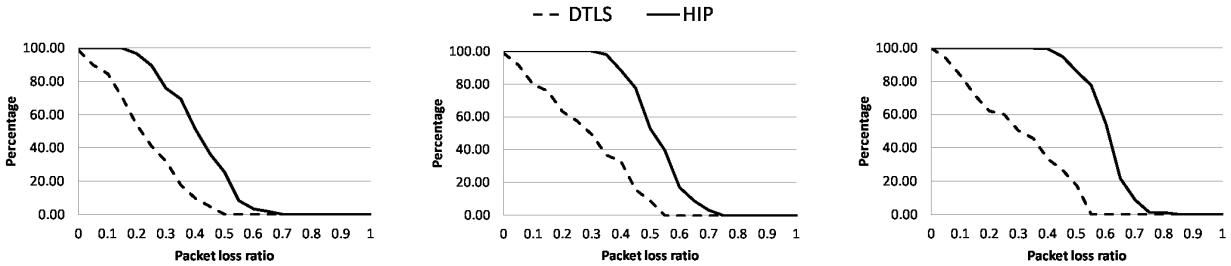
5. EVALUATION

This section describes the prototype implementation, performance and security properties of our two designs.

5.1 Prototype Implementation

Both prototypes are written in C and run as an application on Contiki OS 2.5 [6]. They have been tested in the Cooja simulator and on Redbee Econotags hardware, which features a 32-bit CPU, 128KB of ROM, 128KB of RAM, and an IEEE 802.15.4 radio with an AES hardware coprocessor.

The prototype of our DTLS-based solution uses the *tinydtls* [2] library with small changes for better performance: separate delivery instead of flight grouping of messages, redesigned retransmission mechanism, and no cookies.



(a) 15 seconds timeout (b) 30 seconds timeout (c) 45 seconds timeout

Figure 4: Comparison of the average percentage of successful handshakes (HIP-based and DTLS-based) for different packet loss ratios.

5.2 Performance

This section evaluates the memory and communication overhead of our two proposed solutions.

Table 1: Memory requirements in KB.

	HIP		DTLS	
	ROM	RAM	ROM	RAM
State Machine	5,7	1,2	8,15	1,9
Cryptography	1,7	0	3,3	1,5
Key Management	1,8	0	1	0
DTLS record layer	3,7	0,5	3,7	0,5
Total	**12.9**	**1.7**	**16.15**	**3.9**

Memory needs. Table 1 presents the ROM and RAM consumption of our two designs. The DTLS approach exhibits an overall larger memory footprint because DTLS (i) involves more messages than HIP and (ii) requires SHA2 while HIP-PSK only uses AES-CMAC. However, for key management, HIP adds slightly more overhead, as it requires the definition and handling of new parameter types whereas in DTLS the required information can simply be carried in the payload.

Table 2: Communication overhead for network access and multicast key management.

	HIP	DTLS
Number of messages	6	12
Number of roundtrips	3	4
802.15.4 headers	84 B	168 B
6LoWPAN headers	240 B	480 B
UDP headers	0 B	96 B
Application	304 B	487 B
Total	**628 B**	**1231 B**

Baseline protocol overhead. Table 2 summarizes the required number of round trips, number of messages and the total exchanged bytes for the HIP- and DTLS-based handshake carried out in ideal conditions, i.e. in a network without packet losses. The DTLS handshake is more complex, and thus it involves the exchange of more messages than in HIP. Further, DTLS runs on the transport layer, i.e. UDP, whereas HIP is carried directly over the network layer, i.e. 6LoWPAN. This directly increases the overhead due to lower layer per-packet protocol headers.

Protocol overhead with packet losses. In a network with packet losses, the solutions will perform worse because security handshakes might fail due to the lost messages. This will increase the delays and lead to a higher protocol overhead affecting how the protocols are used in real-life scenarios. Figure 4 compares the percentage of successful handshakes in both approaches as a function of different timeouts and packet loss ratios. In particular, we observe that the HIP-based approach performs significantly better than the DTLS-based approach in the presence of packet loss. Notably, the HIP-based approach is able to tolerate up to 50 % packet loss and still finish approximately four out of five handshakes within 30 seconds. Note that the distribution of polynomial shares is excluded for the sake of simplicity. Related to this is Figure 5 where we show how an increment in the packet loss ratio leads to a higher delay to perform a successful handshake. Here, the HIP-based approach exhibits a significantly lower delay, since it requires less messages, hence less retransmissions in average.

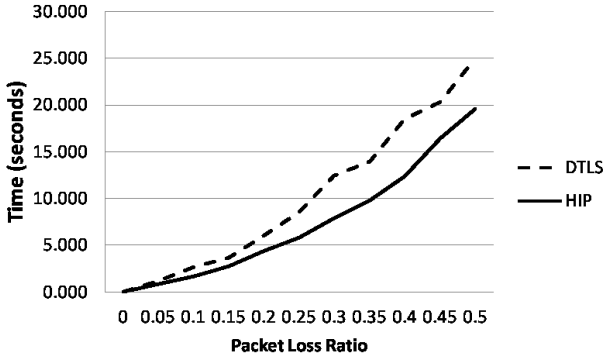


Figure 5: Times to perform network access control and multicast key management, considering only successful handshakes.

5.3 Security Analysis

Both proposed solutions exhibit similar security features. First, the new HIP-PSK handshake relies on a standard challenge-response handshake based on symmetric-keys during which also a session key is derived. The methods are

adapted from HIP-DEX, in particular the usage of CMAC for those purposes is recommended by NIST and helps to minimize resource requirements. Second, unicast communications are protected by means of the standard DTLS record layer, and thus, we offer the standard security features, namely confidentiality, authentication, and replay protection by means of counters. Third, network access is performed in a secure way by using the HIP and DTLS handshakes in each of the proposed solutions. Both HIP-PSK and DTLS-PSK provide mutual authentication between the joining device and domain manager. In both cases, corresponding authorization methods at the domain manager – after the authentication of the joining device – ensure that only valid devices access the network. Note that in our design we also include a key at MAC level whose purpose is to ensure that only authorized devices can use the wireless network. This feature is not needed in a single-hop network access case, but we have already included this here because our whole design – not presented here due to space reasons – also addresses multi-hop networks in which this is a must. Fourth, key management is in charge of securing CoAP applications. In the DTLS method, the master key derived from the polynomial share is used as master key for any application, the DTLS handshake is in charge of providing a secure link. In the HIP approach, a key derivation process based on AMIKEY and using standard key derivation functions is applied. This means that in both cases CoAP applications share master keying material and that a rather standard key derivation process is performed to separate the communication channels. In our designs we used PSK due to several reasons including: (i) our interest in comparing the protocol performance without heavier crypto; (ii) public-key crypto still remains a problem in some application areas. However, we fully acknowledge that public-key cryptography simplifies key management.

6. CONCLUSIONS AND FUTURE WORK

This work has shown how to secure the IP-based Internet of Things with focus on (i) secure network access, (ii) key management, and (iii) secure communication. We do this by means of two solutions based on standardized IP protocols, HIP and DTLS, and adapt them to the constraints of low resource devices (bandwidth, memory and CPU). The first approach relies on the less common HIP protocol further combined with the DTLS record layer and leads to good performance; the second one uses the standard DTLS showing worse performance.

Some of the notable contributions of this paper are: (i) the proposal of HIP-PSK, (ii) the usage of both HIP and DTLS for network access; (iii) the integration of AMIKEY into HIP and DTLS for key management; and (iv) the application of polynomial schemes for efficient pairwise key derivation in HIP-PSK and DTLS-PSK.

As a proof of concept, we implemented the two architectures based on HIP and DTLS over Contiki OS running on a Redbee Econotag. The evaluation shows that the HIP based mechanism is smaller in memory footprint (ROM and RAM) compared to the DTLS mechanism. Results of the communication overhead, delay and resiliency to packet loss also show that the HIP based implementation outperforms the DTLS solution.

This work shows that while the broadly used DTLS allows for better interoperability it compromises efficiency. Our less standard solution based on HIP performs better, but currently is not on the standardization roadmap. We hope that our work provides valuable protocol designs and evaluation results (with their pros and cons) which can provide the much needed direction for the standardization effort in IETF to ensure that the best solutions are adopted.

As a next step we plan to further describe how the presented security architectures, based on HIP and DTLS, can easily provide support for secure network access in multi-hop networks and for secure multicast.

7. REFERENCES

[1] R. Alexander and T. Tsao. Adapted Multimedia Internet KEYing (AMIKEY): An extension of Multimedia Internet KEYing Methods for Generic LLN Environments. Internet-draft, IETF, 2012.

[2] O. Bergmann. tinydtls - a basic dtls server template.

[3] L. Chen. Recommendation for key derivation using pseudorandom functions. SP-800-108, Computer Security Division. Information Technology Laboratory. US Department of Commerce, 2009.

[4] M. Dworkin. Recommendation for block cipher modes of operation: The ccm mode for authentication and confidentiality. SP-800-38c, NIST. Technology Administration. US Department of Commerce, 2007.

[5] P. Eronen and H. Tschofenig. Pre-Shared Key Ciphersuites for Transport Layer Security (TLS). RFC 4279 (Proposed Standard), December 2005.

[6] A. Dunkels et al. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *29th Annual IEEE International Conference on Local Computer Networks*, pages 455–462. IEEE, 2004.

[7] A. Perrig et al. Spins: security protocols for sensor networks. *Wireless Networks*, 8(5), 2002.

[8] A.J. Menezes et al. *Handbook of Applied Cryptography*. 5 edition, Aug. 2001.

[9] C. Blundo et al. Perfectly-secure key distribution for dynamic conferences. *Advances in cryptology*, 1993.

[10] J. Arkko et al. MIKEY: Multimedia Internet KEYing. RFC 3830, August 2004. Updated by RFCs 4738, 6309.

[11] R. Moskowitz et al. Host Identity Protocol Version 2 (HIPv2). Internet-draft, IETF, 2012.

[12] V. Gupta et al. Sizzle: A standards-based end-to-end security architecture for the embedded internet. *Pervasive and Mobile Computing*, 1:425–445, 2005.

[13] Z. Shelby et al. Constrained Application Protocol (CoAP). Internet-Draft draft-ietf-core-coap-12, IETF, October 2012.

[14] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, CCS '03. ACM, 2003.

[15] R. Moskowitz. HIP Diet EXchange (DEX). Internet Draft draft-moskowitz-hip-rg-dex-06, IETF, 2012.

[16] E. Rescorla and B. Korver. Guidelines for Writing RFC Text on Security Considerations. RFC 3552 (Best Current Practice), July 2003.

[17] E. Rescorla and N. Modadugu. Datagram Transport Layer Security. RFC 4347, April 2006. Obsoleted by RFC 6347, updated by RFC 5746.