Singapore, 7-9 April 2015

Secure communication in IP-based wireless sensor networks via a trusted gateway

Floris Van den Abeele, Tom Vandewinckele, Jeroen Hoebeke, Ingrid Moerman, Piet Demeester
Department of Information Technology, Ghent University – iMinds
Gaston Crommenlaan 8/201 B-9050 Ghent, Belgium
{fvdabeele,jhoebeke,imoerman,pietdm}@intec.ugent.be

***Abstract*—As the IP-integration of wireless sensor networks enables end-to-end interactions, solutions to appropriately secure these interactions with hosts on the Internet are necessary. At the same time, burdening wireless sensors with heavy security protocols should be avoided. While Datagram TLS (DTLS) strikes a good balance between these requirements, it entails a high cost for setting up communication sessions. Furthermore, not all types of communication have the same security requirements: e.g. some interactions might only require authorization and do not need confidentiality. In this paper we propose and evaluate an approach that relies on a trusted gateway to mitigate the high cost of the DTLS handshake in the WSN and to provide the flexibility necessary to support a variety of security requirements. The evaluation shows that our approach leads to considerable energy savings and latency reduction when compared to a standard DTLS use case, while requiring no changes to the end hosts themselves.**

***Keywords*—Wireless sensor networks, DTLS, IP, IoT, CoAP, 6LoWPAN, Gateway**

I. Introduction

Deploying the Internet Protocol inside WSNs has been technologically feasible for some years (e.g. via ZigBee IP and 6LoWPAN). Today however, vendors might deploy IP inside their WSN but the IP network itself is often unaccessible to the public Internet. Instead, vendors rely on their own platforms to manage all communications from and to their WSNs. While this gives vendors a large amount of control over their products, it also means that third party developers experience difficulties reusing existing sensors in new services. They are often left to integrating with vague and volatile cloud-based APIs that differ from vendor to vendor. This situation is commonly referred to as the "Intranet of Things" [1].

In order to step away from these closed ecosystems, a number of open issues for 6LoWPAN networks are identified in [2]. More specifically, the paper mentions a number of issues relating to security in 6LoWPAN WSNs. The work presented here addresses these security-related issues for the IETF low power and lossy network protocol stack [3]. In these types of WSNs, security is provided by the end-to-end transport layer protocol Datagram TLS [4]. DTLS provides communications privacy for datagram protocols (such as the Constrained Application Protocol) by enabling client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery. The protocol is based on the Transport Layer Security (TLS) protocol and provides equivalent security garantees. For the low-power and battery-operated devices commonly found in WSNs, the large communication overhead of DTLS is problematic. This is mainly due to the costly DTLS handshake. In the next section we show that the communication overhead for a single-hop network without any packet loss is an additional 11 messages spanning five extra round trips. These extra messages can be attributed almost entirely to the complex DTLS handshake. Thus, the main objective of this paper is to overcome the cost of the DTLS handshake in the WSN.

Our contributions in this paper consist of proposing, implementing and evaluating an approach that relies on a trusted gateway for mitigating the overhead of the DTLS handshake in IP-based WSNs. Furthermore, our trusted gateway concept is able to solve a number of other practical issues related to DTLS and WSNs, such as the poor scalability of PSK-based cipher suites and the loss of application-layer processing at the edge of WSNs that is inherent to end-to-end security. As a result, we hope that our work can hasten vendors to adopt the open IETF protocol stack in lieu of their closed platforms.

II. Problem statement and research goals

To give an example of the overhead incurred in setting up a DTLS session we take a look at the number of bytes that are communicated for a typical CoAP transaction on top of DTLS. For one CoAP request/response pair that would measure 72/70 bytes respectively in plain-text, DTLS triggers a message exchange of 1529 bytes spread over 13 packets that require a minimum of 5 round-trip times for being transferred. After the DTLS handshake has finished, DTLS application data messages contain a 13 byte DTLS header plus the encrypted payload (i.e. CoAP messages) that requires an additional 16 bytes for storing the authentication tag and the sequence counter for the used cipher suite. Thus, 1329 bytes were exchanged for the DTLS handshake and the close notify messages, while the two application data messages only took 200 bytes[1].

Considering wireless communication is one of the biggest energy consumers in WSNs, the cost of the handshake is problematic for employing DTLS inside WSNs. Therefor, the main research goal of this paper is to mitigate the cost of the DTLS handshake in the WSN. Rather than inventing a new

[1]Note that in this case the DTLS client grouped multiple DTLS records in one flight, whereas the server did not (mainly to avoid 6LoWPAN fragmentation while sending). Furthermore, there was no packet loss and the two close notify messages sent to close the session were counted as part of the DTLS message exchange. These numbers were obtained for 'TLS_PSK_WITH_AES_128_CCM_8', which is a cipher suite suited for constrained environments such as WSNs.

978-1-4799-8055-0/15/$31.00 ©2015 IEEE

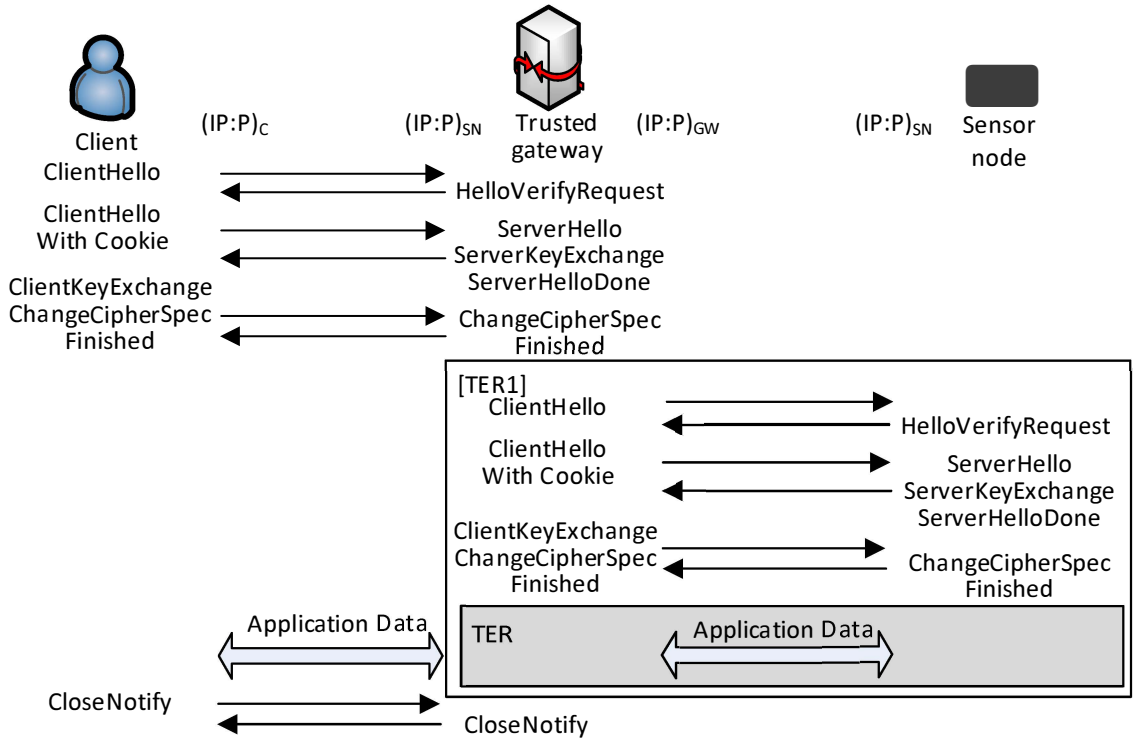


Figure 1. The trusted gateway sets up a DTLS session with the sensor node once (TER1) and will reuse this session for future DTLS clients (TER).

security protocol, our aim is to make no changes to the DTLS protocol at all. A secondary research goal is to overcome the scalability issue inherent to PSK cipher suites. While PSK cipher suites allow for very compact key exchanges (only the identity hints for the PSKs are exchanged), they are problematic when secure communication with Internet hosts is necessary as maintaining a PSK with every Internet host is impossible. When combined these two goals should lead to a solution that is readily deployable. As a result, WSNs are able to profit from the benefits provided by DTLS without suffering from its costly handshake mechanism.

III. Trusted Gateway for Mitigating the Cost of DTLS Handshakes

Considering the goals from the previous section, a number of solutions are possible. However, when changes to the endpoints' DTLS protocol are out of the question most of these solutions are no longer applicable. In the end, we have chosen for a trusted gateway approach to mitigate the cost of the handshake in the WSN. The gateway achieves this by maintaining long-lived DTLS sessions with WSN nodes and transparently terminating DTLS sessions with Internet hosts.

Figure 1 gives an overview of our termination approach. Clients $(IP:P)_C$ initiate a DTLS session with the sensor node using the actual IPv6 address of the sensor node $(IP:P)_{SN}$. If the gateway is configured to terminate sessions for the sensor node, it will intercept the ClientHello message and respond with a HelloVerifyRequest message using the sensor node's IPv6 address as a source address $(IP:P)_{SN}$. If the handshake is completed successfully, then the trusted gateway will either setup a DTLS session with the sensor (TER1, so named because it is the 1st DTLS session destined for the sensor that is terminated by the gateway) or re-use an existing session (TER) using its WSN transport address $(IP:P)_{GW}$. Once there is an active session between the gateway and sensor node, the client and sensor node are able to communicate. When the client closes its DTLS session (i.e the close notify message at the bottom of figure 1), the gateway will keep the session with the sensor node open for future re-use.

In our approach, the DTLS sessions with sensor nodes are set up once by the gateway and are reused whenever another client wishes to communicate with the sensor node via DTLS. The gateway multiplexes requests from multiple clients over the long-lived session that is maintained with the WSN node. When responses arrive from the sensor node, the gateway is also responsible for demultiplexing and making sure each response reaches its intended destination. This intercepting and terminating mode of operation is transparent to both Internet hosts and WSN nodes, i.e. no changes to the DTLS and application stack are necessary. From the point of view of the WSN node, all DTLS traffic appears to originate from the trusted gateway $(IP:P)_{GW}$. In most cases this is not a problem, however when the WSN node must know the transport address of the client then the address should be transported inside the DTLS payload (one suitable candidate could be a CoAP option). From the point of view of the client, all DTLS traffic appears to be originating from the WSN node $(IP:P)_{SN}$ as the trusted gateway operates completely transparent.

One consequence of our termination approach is that it is straightforward to employ different cipher suites for the client and the sensor node. For example, pre-shared key (PSK) cipher suites [5] are a good option for the sensor node's suite as the amount of keying material that has to be exchanged is low. For the client's cipher suite a more scalable option than PSK

suites is desirable as there are expected to be a large number of clients. Cipher suites based on public key cryptography are good candidates due to the better scalability offered by public key infrastructure in comparison to PSKs.

Another consequence of terminating DTLS sessions is that a gateway can employ a security policy. This policy can decide whether certain requests should be passed over DTLS to the sensor node or in plain text. Consider as an example the case where DTLS is only used to secure the Internet leg of the communication for non-critical communication. Figure 2 gives an overview of the different envisioned scenarios.

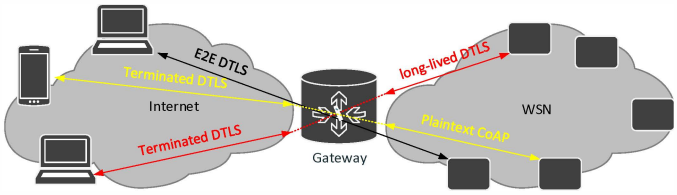


Figure 2. Different DTLS termination policies are possible at the gateway.

The resulting trusted gateway is a powerful entity that can perform (large amounts of) processing at the edge of the WSN, both on the transport layer (i.e. DTLS) but also on the application layer (thereby becoming an application proxy). This can help alleviate the load on the sensor network, which can increase its life span. Examples include access control and caching performed at the gateway for CoAP resources hosted by sensor nodes. In case of end-to-end security between client and sensor node, the gateway can not provide these services as it can not decipher the DTLS messages. Terminating the DTLS session at the gateway, makes these additional services and all their benefits possible. Another side effect is that by reducing the amount of handshakes with the WSN node, the total number of failed handshakes due to packet loss is lowered [6].

The trusted gateway approach also has a number of downsides. Firstly, it introduces another party where all communication between Internet hosts and sensor nodes passes in plain-text. Thus the risk of this party being compromised should not be overlooked. Therefor, it should be properly maintained, monitored and hardened against known attacks. On the other hand, the gateway also shields the DTLS stack on the WSN nodes from direct communication with Internet hosts, thus their exposure to potential attacks is reduced. From a management point of view, it might actually be more feasible to harden a DTLS terminating gateway (which typically runs on more accessible hardware) than an entire sensor network.

A second issue is that the gateway should be on the routing path between the Internet host and the WSN node in order to terminate the session. If the gateway is not on the routing path then an extra mechanism is necessary to intercept the DTLS traffic. A number of solutions are possible to solve this issue and these are briefly discussed in the future work section.

A potential criticism is that gateway in effect breaks the end-to-end relationship between the client and the sensor node. While this is true, the arguments from the previous paragraphs and the results in next section show that the benefits of doing so significantly outweigh the costs as long as the gateway is not compromised. In cases where the sensor node does not trust the sensor gateway, a different machine that is trusted by the sensor can be used as a trusted gateway (e.g. a reverse proxy running outside the WSN, e.g. in the cloud). We also refer to modern data centers where it is common practice to offload TLS sessions to SSL load balancers for improved scalability [7]. Similar to our approach, the machine terminating the TLS session differs from the machine responding to the HTTP request.

IV. Evaluation

In order to evaluate our termination approach we composed a realistic application scenario and ran a number of simulations for three different configurations of the gateway. In the first configuration (E2E) the gateway is left unconfigured and just acts as a normal Internet router. In this configuration the client establishes DTLS sessions with the sensor nodes directly. In the second configuration, TER1, the gateway is configured to terminate the DTLS session and setup a new DTLS session with the sensor node. This represents the case where the gateway does not have an active session with the sensor node and has to setup a new one. In the third scenario, TER, the gateway is configured to terminate the DTLS session and reuse an existing DTLS session with the sensor node inside the WSN. This is considered to be the steady state in typical operations, as a sensor node is expected to have an active DTLS session with the gateway during most of its lifetime. The E2E and TER configurations are shown in figure 2. The application scenario is also executed in plain text between the client and the sensor node. These results are labeled as PLT and serve as a lower limit for what can be achieved.

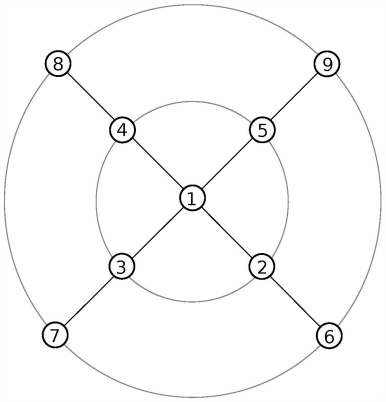
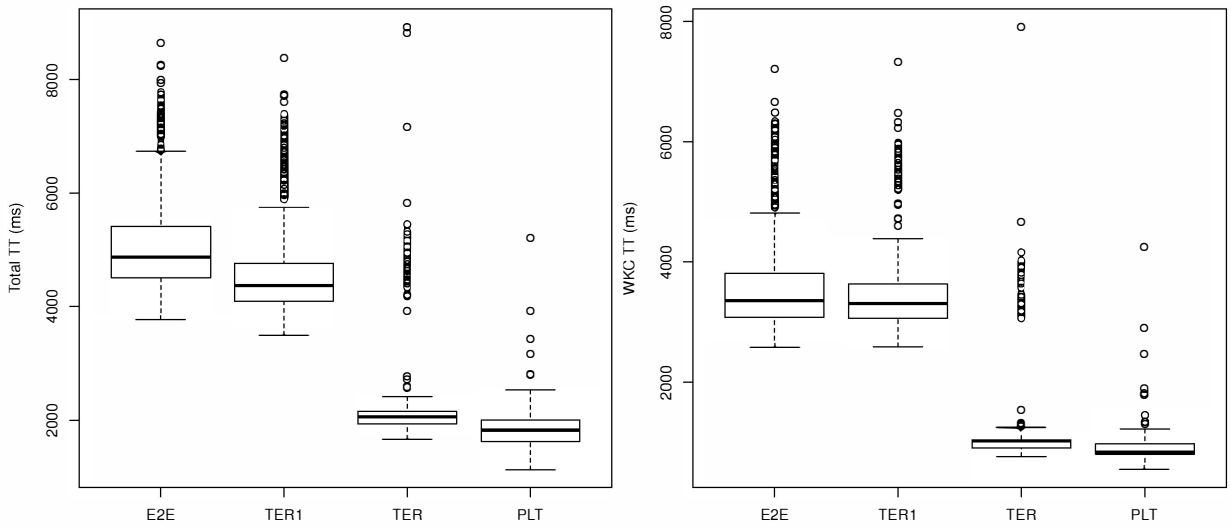


Figure 3. WSN network topology: the 9 nodes are arranged in an X pattern. The DTLS servers (6, 7, 8, 9) are 2 hops away from the border router (1).

The application scenario consists of executing three CoAP requests in sequence, i.e. executing request i+1 blocks until the response for request i is received. First, a discovery step is performed by retrieving the discovery resource ".well-known/core" from the sensor node. In our particular setup this requires two requests/responses via CoAP's block2 mechanism as the resource is larger than the MTU inside the 802.15.4 WSN. Secondly, a sensor measurement is retrieved from a "/s" resource. Finally, a request is sent to toggle an actuator via the "/a" resource. The resulting scenario represents a discover/sense/actuate cycle that is commonly found in Wireless Sensor Network use cases. Prior to every application scenario,



(a) Total transaction times (TTT)

(b) Transaction times up to the reception of .well-known/core (WKC TT)

Figure 4. Transaction times for the three configurations of the gateway (E2E, TER1, TER) and the plain-text CoAP reference case (PLT)

the client sets up a DTLS session with the DTLS server in question. After completing the scenario, the client closes the DTLS session. In case of PLT, the DTLS session is obviously not setup.

The simulated WSN consists of 9 RM090 motes, arranged in a X pattern as shown in figure 3. In the middle of the X there is a RPL border router that routes traffic to and from the WSN and that is responsible for the RPL DODAG. Of the other 8 nodes, four act as an intermediary router for the last 4 nodes that act as DTLS servers. Thus, there are four DTLS servers that are 2 hops away from the border router. The simulation was run in Cooja [8] on the RM090 hardware platform. RM090 motes contain a MSP430f5437 µC with 16 kB RAM and 256 kB ROM memory and a CC2520 802.15.4 radio, both of which are simulated in software by mspsim. For the DTLS servers inside the WSN we employed TinyDTLS configured to accept the 'TLS_PSK_WITH_AES_128_CCM_8' cipher suite with a PSK hint of 15 bytes. For all three configurations, we ran the application scenario a hundred times per DTLS server. Prior to starting our measurements, we waited 5 minutes to allow the RPL DODAG to stabilize. All results were obtained using the default CSMA MAC protocol and ContikiMAC RDC protocol available in Contiki. The trusted gateway ran on a standard x86 laptop with an Intel i5-2520M CPU and 8 GiB of RAM and was implemented as part of our CoAP++ framework in Click router [9][2].

Figure 4(a) shows the total transaction time (TTT) of setting up the DTLS session, completing the application scenario and closing the session. In the TER configuration, the DTLS session with the sensor node is already available and can be reused. As a result, the expensive DTLS handshake with the sensor node can be avoided. In this case, the TTT is composed of setting up a DTLS session between the client and the trusted gateway, completing the application scenario and closing the DTLS session between the client and the gateway. The figure shows that our approach can reduce the median TTT by more than half when compared to the E2E case. The lower TTT in the TER1 scenario when compared to the E2E configuration, is caused by the DTLS client closing the session with the gateway as opposed to the sensor node as is the case in the E2E configuration. Therefore the close notify message does not traverse the WSN (as the trusted gateway will keep the DTLS session with the WSN node open for future reuse).

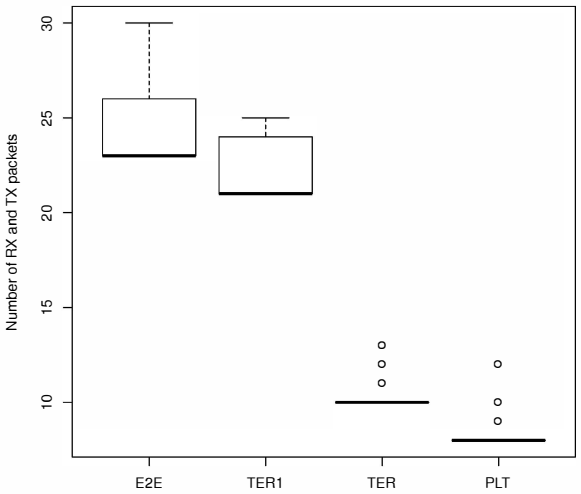


Figure 5. Number of packets received and transmitted by sensor node

This result is confirmed by figure 4(b) which shows the transaction time (TT) up to the reception of the .well-known/core resource. For the TER configuration this time is nearly equal to two times the host-to-host round trip time, as the handshake with the gateway only takes 42 ms. Because the WKC TT does not include the close notify messages, the results for E2E and TER1 are similar. Figures 4(a) and 4(b) also show that the median transaction times for the TER con-

[2]An archive file with the raw data obtained from our experiments is published at [10].

figuration are close to the lower limit of the PLT reference case. The observed difference is due to 6LoWPAN fragmentation of the larger DTLS packets and the time penalty of performing AES cryptography in software.

Figure 5 presents the sum of the number of packets that were received and transmitted by the sensor node in the form of box plots. The TER1 median is two packets smaller than E2E (21 vs 23 packets), this confirms that the close notify messages do not traverse the sensor network in the TER1 case. Also note that the TER case sends two additional packets when compared to the PLT case. This is because the combination of DTLS headers with the large discovery responses triggers 6LoWPAN fragmentation. As a result the two blocks of the discovery response are fragmented into four fragments.

In terms of energy usage, the results show that by reusing existing DTLS sessions we can achieve a median energy saving of 59.5%. In figure 6 the total energy usage of the sensor node during the application scenario is shown as a box plot on the right axis. The bar plot and its axis on the left shows the median energy usage per energy usage category. The plot shows that the relative energy savings are largest for the CPU category. This is because the SHA hash calculations (that are performed in software) during the handshake messages in case of the 'TLS_PSK_WITH_AES_128_CCM_8' cipher suite are avoided for the TER configuration. Note that when a sensor node supports a hardware coprocessor for cryptographic calculations, this difference is expected to be smaller.

The largest absolute energy savings are achieved in the radio categories. This is because the communication inherent to the DTLS handshake is not present. The difference in total energy usage between the E2E and TER1 experiments is again explained by the absence of the close notify message in the TER1 case. In effect, our approach spreads the cost of the initial DTLS handshake between the trusted gateway and the sensor nodes (TER1) over all future interactions with the sensor node (TER). When comparing to the lower limit, the TER configuration consumes additional energy for performing cryptography, sending and receiving larger packets (due to the presence of the DTLS headers) and for sending additional packets due to the 6LoWPAN fragmentation.

	RAM (kB)	ROM (kB)
DTLS server	8.1	70.8
CoAP server (reference)	6.0	48.5
Per additional DTLS session	0.596	0

Table I. CODE FOOTPRINT FOR DTLS AND COAP SERVERS AND PER ADDITIONALLY SUPPORTED DTLS SESSION

Finally, table I lists the code footprint of the DTLS server [3]. The DTLS server includes contiki's RPL implementation, the TinyDTLS server and the Erbium CoAP server. When compared to a CoAP server, the DTLS server requires an additional 22.3 kB of ROM and 2.1 kB of RAM. When more than one simultaneously-active DTLS session is required, the TinyDTLS server requires 596 bytes of RAM memory for every additional session. In our approach the DTLS server only has one active session with the gateway, so no additional RAM has to allocated to accommodate for multiple clients.

[3]The contiki source code is available at [11]. Size measurements were performed with msp430-gcc 4.7.0.

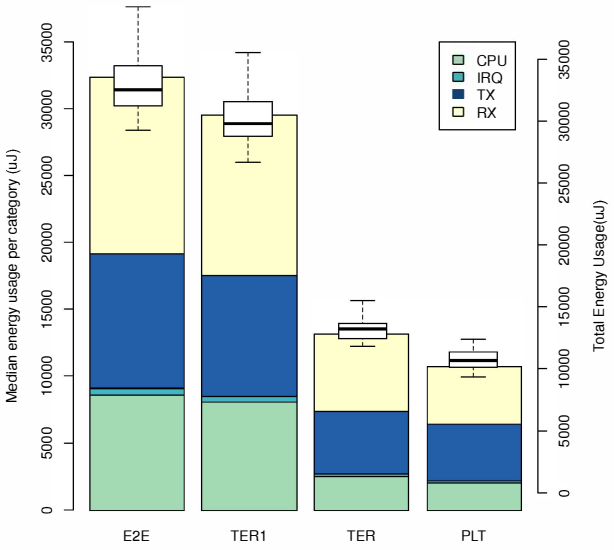


Figure 6. Median energy usage per category (left axis) and total energy usage (right axis).

Instead, supporting multiple DTLS peers is handled by the trusted gateway.

V. RELATED WORK

In terms of related work, the ongoing efforts in the DICE working group and two other relevant works from literature are discussed.

DTLS In Constrained Environments (DICE) is an IETF working group that focuses on supporting the use of DTLS transport-layer security in constrained environments. The scope includes both constrained devices and networks. Its first task is to define a DTLS profile that is suitable for Internet of Things applications and is reasonably implementable on many constrained devices. To this end, the WG has adopted a draft [12] that discusses the use of PSKs, raw public keys and certificates as well other practical issues that might arise when using DTLS. One difference from our work, is that the constrained device is considered to be the DTLS client and that they are preconfigured with the addresses of their communication servers. Our approach provides more flexibility with the constrained device as a DTLS server, as the sensors and their data are readily accessible by multiple parties over IP and are therefor not limited to the number of parties programmed into the device. Apart from multicast security, the group also intends to investigate practical issues around the DTLS handshake in constrained environments. Proposed work includes compression of DTLS messages and completing the DTLS handshakes over CoAP. These mechanisms look promising and are largely orthogonal to our work as they aim to optimize the handshake mechanism itself. Finally, the contribution of Keoh S. et al. [13] gives a clear overview on securing the Internet of Things from a standardization point of view with a focus on the IETF.

The authors of Lithe [14] propose a novel DTLS header compression scheme that aims to reduce the energy consumption by leveraging the 6LoWPAN standard. The header compression scheme significantly reduces the number of trans-

mitted bytes while its use of 6LoWPAN ensures interoperability with existing DTLS implementations. Their approach is evaluated using the open source TinyDTLS implementation on ContikiOS. The authors report significant improvements in terms of packet size, energy consumption, processing time, and response times. The presented compression scheme is complementary to our approach: i.e. both approaches can benefit from one another.

In [15], Hummen R. et al. introduce a security architecture for delegation purposes that executes the DTLS Handshake on a trusted and powerful delegation server. Afterwards, the security context is transferred securely to the constrained device by using the session resumption mechanism in DTLS. This transfer is secured by symmetric key cryptography that requires a secret preshared key between the constrained device and the delegation server. As a result, the delegation server can authenticate and authorize Internet nodes. The approach saves a considerable amount of resources on the constrained device: memory requirements decrease with 64%, calculations drop with 97% and there is 68% less traffic. While the authors' research goals are similar to ours, the followed approach is very different. There are a number of differences worth noting here. Firstly, our approach can provide similar gains by offloading the handshake to a third party without requiring any changes to the DTLS implementations running on the Internet client and the WSN node. This is a huge benefit as adopting the proposed delegation mechanism would take a considerable amount of time. Secondly, our approach also allows for more flexibility as the use of DTLS inside the WSN is configurable and therefor optional. Finally, we also consider application-layer processing at the gateway (e.g. caching) whereas the delegation server operates solely on the transport layer.

VI. Conclusion and future work

This paper has presented the use of a trusted gateway to mitigate the overhead of the DTLS handshake in IP-based WSNs. By terminating DTLS sessions with Internet hosts and multiplexing their contents over a long-lived session with the WSN node, considerable energy savings can be achieved. Through simulation of a representative application scenario we have shown that our approach can save up to 60% in energy expenditure. The time to complete this scenario is also reduced by more than half. Furthermore, our approach allows to use different cipher suites on the public Internet and the WSN. Thus, offering X.509 certificates on behalf of WSN nodes is supported as well.

In the future, we will adapt our approach for networks where the trusted gateway is not on routing path between the Internet and the WSN. In order to reduce costs the trusted gateway might be virtualized in the cloud, in this case a lightweight tunnel from the cloud to the WSN gateway might prove a feasible option. However, a solution where every sensor node has a virtual counterpart located in the cloud (hosted at an Internet address that is routed to the cloud) is also possible. Furthermore, we are in the process of adding more functionality on the application layer (such as access control and caching for CoAP resources hosted on sensor nodes) to this (potentially virtualized) trusted gateway.

Acknowledgements

The authors would like to acknowledge that part of this research was supported by the COMACOD project. The iMinds COMACOD project is cofunded by iMinds (Interdisciplinary institute for Technology) a research institute founded by the Flemish Government. Partners involved in the project are Multicap, oneAccess, Track4C, Invenso and Trimble, with project support of IWT.

References

[1] M. Zorzi, A. Gluhak, S. Lange, and A. Bassi, "From today's intranet of things to a future internet of things: a wireless-and mobility-related view," *Wireless Communications, IEEE*, vol. 17, no. 6, pp. 44–51, 2010.

[2] F. Van den Abeele, J. Hoebeke, I. Moerman, and P. Demeester, "Fine-grained management of coap interactions with constrained iot devices," in *Network Operations and Management Symposium (NOMS), 2014 IEEE*, May 2014, pp. 1–5.

[3] I. Ishaq, D. Carels, G. K. Teklemariam, J. Hoebeke, F. Van den Abeele, E. De Poorter, I. Moerman, and P. Demeester, "Ietf standardization in the field of the internet of things (iot): A survey," *Journal of Sensor and Actuator Networks*, vol. 2, no. 2, pp. 235–287, 2013.

[4] E. Rescorla and N. Modadugu, "Rfc 6347: Datagram transport layer security version 1.2," 2012. [Online]. Available: https://tools.ietf.org/html/rfc6347

[5] P. Eronen and H. Tschofenig, "Pre-shared key ciphersuites for transport layer security (tls)," RFC 4279, December, Tech. Rep., 2005.

[6] O. Garcia-Morchon, S. L. Keoh, S. Kumar, P. Moreno-Sanchez, F. Vidal-Meca, and J. H. Ziegeldorf, "Securing the IP-based internet of things with HIP and DTLS," *Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks - WiSec '13*, p. 119, 2013. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2462096.2462117

[7] P. Membrey, D. Hows, and E. Plugge, "Ssl load balancing," in *Practical Load Balancing*. Springer, 2012, pp. 175–192.

[8] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with cooja," in *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*. IEEE, 2006, pp. 641–648.

[9] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The click modular router," *ACM Trans. Comput. Syst.*, vol. 18, no. 3, pp. 263–297, Aug. 2000.

[10] F. V. den Abeele, T. Vandewinckele, J. Hoebeke, I. Moerman, and P. Demeester, "Data for Secure communication in IP-based wireless sensor networks via a trusted gateway publication," Feb. 2015. [Online]. Available: http://dx.doi.org/10.5281/zenodo.15661

[11] M. Alvira, A. Dunkels, N. Tsiftes, D. Kopf, and G. O. et al., "contiki: Ieee issnip 2015," Feb. 2015. [Online]. Available: http://dx.doi.org/10.5281/zenodo.15658

[12] E. H. Tschofenig, "A datagram transport layer security (dtls) 1.2 profile for the internet of things," 2014. [Online]. Available: https://tools.ietf.org/html/draft-ietf-dice-profile-04

[13] S. L. Keoh, S. Kumar, and H. Tschofenig, "Securing the internet of things: A standardization perspective," *Internet of Things Journal, IEEE*, vol. 1, no. 3, pp. 265–275, June 2014.

[14] S. Raza, H. Shafagh, K. Hewage, R. Hummen, and T. Voigt, "Lithe: Lightweight secure coap for the internet of things," *Sensors Journal, IEEE*, vol. 13, no. 10, pp. 3711–3720, Oct 2013.

[15] R. Hummen, H. Shafagh, S. Raza, T. Voigt, and K. Wehrle, "Delegation-based authentication and authorization for the ip-based internet of things."