TLS with PSK for Constrained Devices

Vladislav Perelman (Jacobs University, Germany)
Mehmet Ersue (Nokia Siemens Networks, Germany)

February 20, 2012

1 Introduction

In the recent years the advances in the area of Wireless Sensor Networks (WSNs) have led to extensive research of the question of securing WSNs, a challenging problem, mostly due to the constraints that have to be dealt with. Limited power supply and limited computational resources, very small RAM and Flash memory on the devices, network interfaces with low data rates and limited bandwidth, possibly hostile deployment environments – all of these restrictions are the reason why many questions are still considered to be open research topics.

Various security mechanisms have been proposed and implemented over the last several years, ranging in their layer of operation from the link layer all the way to the application layer. We believe, however, that it would be best to have as few security suites as possible, preferably only one. This would allow for easier integration and interoperability of protocols on the network. We also believe that TLS would be a good choice for such protocol, since it is wide-spread, it provides end-to-end security, runs in user space and therefore can be added to any application. In Section 3 of this paper we will look at what questions should be discussed when integrating TLS into the WSNs and porting it to the constrained devices. But before that we will give a brief overview of the most recent advances in the area of WSN security.

2 State-of-the-Art for WSN Security

TinySec [1] and ContikiSec [2] are examples of security suites that operate in the link layer. Designed for use with TinyOS and Contiki OS respectively they provide, depending on the mode of operation, either confidentiality, authenticity or both. Both mechanisms provide approximately the same level of security, by using, however, different cryptographic algorithms.

In the network layer Jorge Granjal et al. proposed a Secure Interconnection Model for WSN (SIMWSN) [3], which made use of the Authentication Header and the Encapsulating Security Protocol of IPsec to provide end-to-end security. In 2011, Shahid Raza et al. have presented their implementation of compressed IPsec for 6LoWPAN networks [4]. They developed an encoding for the AH and the ESP extension headers using the LOWPAN_NHC – compression format introduced in RFC 6282 [5].

In the transport layer Sizzle was the first implementation of the HTTPS stack (HTTP that is being used over SSL). It used MD5 and SHA1 as hashing algorithms, RC4 for bulk encryption, ECDH and ECDSA for key exchange. Sizzle demonstrated that using those protocols was feasible for constrained devices. The evaluation of Sizzle showed firstly that using ECC for public-key cryptography is by far more suited for constrained devices than RSA (which was implemented solely for the performance comparison) and secondly that performance of the secure web server on the mote is quite acceptable for infrequent communications.

SSNAIL is a lightweight SSL implementation that was developed as a security mechanism for the project called Sensor Networks for All-IP worLd (SNAIL), which had a goal of implementing an IP-WSN platform with a widespread test-bed. SNAIL sensor nodes run on two different platforms – OSAL (Operating System Abstraction Layer) [6] of TI solution and ANTS EOS [7] of RESL (Realtime an Embedded Systems Laboratory) in ICU. SSNAIL's implementation uses 30 kB of Flash memory and around 500 bytes of RAM.

In certain but quite rare cases, applications choose to introduce their own security protocols in the application layer instead of using security in the underlying layers. One example of such a protocol is the User-based Security Model (USM) of SNMPv3 [8], which is responsible for authenticating, encrypting and decrypting SNMP packets. In 2010, S. Kuryla implemented an SNMP agent under the Contiki OS [9] supporting the USM security. It provided support for HMAC-MD5-96 authentication and CFB128-AES-128 symmetric encryption operations. The USM module was by far the largest part of the implementation with respect to the statically allocated RAM – 122 bytes out of 235 bytes for the entire agent. In terms of Flash memory, the cryptography primitives, such as AES and MD5, occupied around 60% of the entire implementation – approximately 20 kB.

3 TLS for WSN

When thinking about porting TLS to the constrained environment of WSNs the first question that needs to be discussed is how the communicating sides agree on the used keys. Generally there are two options available – by the means of public-key cryptography or by using pre-shared keys. For a while public-key algorithms were considered to be too heavy-weight for the motes, however, recent developments showed that protocols like the Elliptic Curve Diffie Hellman (ECDH) [10] and the Elliptic Curve Digital Signature Algorithm (ECDSA) [11] can be made small enough to be run on constrained devices while at the same time providing reasonable security [12]. ECC-based SSL handshake has been shown to complete on average four times faster than that using RSA [13] and since keys needed for ECC functions are much smaller, less memory is used on the device. When using pre-shared keys, starting the TLS session requires much less time and computational power, however establishing the keys as well as updating the keys becomes a challenging issue. Several surveys exist that discuss various possible key management schemes in detail [14, 15, 16].

Another issue that needs to be discussed is that of trusting the used keys. Traditionally, TLS clients authenticate public keys by means of X.509 certificates, which are typically quite large, and sending them over the IEEE 802.15.4 network is a burden. The current TLS standard does not provide a way for the TLS client to tell the server that it already possesses a public key in which it trusts, leaving the TLS server no other option but to send the full certificate. P. Wouters et al. have proposed a new TLS certificate type for exchanging raw public keys for use with out-of-band authentication [17].

At Jacobs University Bremen we are currently developing a TLS implementation for the Contiki OS, that uses a pre-shared key approach. The goal is to support the TLS_PSK_WITH_AES_128_CCM_8 cipher suite, which is described in [18] and is one of the mandatory suites to implement according to the Constrained Application Protocol (CoAP) Internet Draft [19] developed in the IETF CoRE working group. Since our current goal is to make the implementation as small as possible, it does not support any of the optional messages (e.g., certificate-related messages).

We have decided that the implementation should be made available as a Contiki internal library (i.e., in the `core/net` folder), which allows it to be used in any application simply by including `tls.h`. The TLS API was made as straightforward and self-explanatory as possible – it provides functions like *TLS_Listen* and *TLS_Connect* for server and client respectively as well as read and write functions for both. When a client connects to the server, it needs to wait for the TLS handshake to take place, after which a special event is raised notifying of the handshake completion and all necessary security information is passed to the client for further use while sending and receiving data.

According to the current design, the application which includes TLS acts either as a TLS server or as a TLS client, however not as both at the same time. The pre-shared key, which is assumed to be present on the mote, should be located in the Flash memory and is accessed via Contiki's File System API. For the computing of SHA256 as well as HMAC_SHA256 an external library [20] is used, while the AES operations are done using slightly modified code from the OpenSSL project [21]. In order to reduce the amount of the used RAM, all static variables that AES requires are stored in the Flash memory.

In the next implementation step we are going to integrate it into the NETCONF-Light implementation that was developed at our university last year [22]. As recommended in Section 3.2.2. of [23], the integration in NETCONF-Light will follow the optional support of TLS Pre-Shared Key (PSK) authentication and use TLS pre-shared key cipher suites defined in [24].

During the TLS Handshake, the client indicates which key to use by including a ”PSK identity” in the TLS ClientKeyExchange message. On the server side, this PSK identity is used to lookup the key corresponding to the presented PSK identity. If the selected pre-shared keys match, then the client is authenticated and the PSK identity is used to derive the NETCONF username, which is then used as the authenticated identity of a NETCONF client.

4 Conclusion

Although implementing lightweight protocols is possible, using security protocols designed for the IP networks is not always feasible on WSN nodes for diverse reasons. Security protocols used today in the Internet were initially designed for nodes with high computational capabilities and permanent power supply, large amounts of memory and network links with sufficient bandwidth. It is a challenge for new developments to provide a lightweight and efficient implementation for constrained devices. Nevertheless, if constrained nodes already have TCP/IP connectivity then, there is a benefit to be able to communicate with Internet hosts using security solutions interoperable with the ones already deployed in the Internet. Taking into account that it is very difficult to make radical changes in the Internet, such as introducing new protocols, the only viable option for sensor networks is to adopt already existing security solutions. But the big question of how to make them as efficient as possible for WSNs, without breaking the interoperability with existing implementations, remains.

We have surveyed the state of the art research in adopting TLS for constrained WSN motes and came to the conclusion that a minimalistic implementation as proposed above is efficiently possible and still provides essential end-to-end security features. As such, we believe TLS is a good basis for a transport security protocol to be used in constrained WSN devices, since it is widely spread and enables the interoperability to existing security solutions in the Internet.

References

[1] Chris Karlof, Naveen Sastry, and David Wagner. TinySec: A Link Layer Security Architecture for Wireless Sensor Networks. In *ACM SENSYS’04*. ACM, 2004.

[2] Lander Casado and Philippas Tsigas. ContikiSec: A Secure Network Layer for Wireless Sensor Networks under the Contiki Operating System. In *Proceedings of the 14th Nordic Conference on Secure IT Systems: Identity and Privacy in the Internet Age*, pages 133–147. Springer-Verlag, 2009.

[3] J. Granjal, E. Monteiro, and J. Sá Silva. A secure interconnection model for IPv6 enabled wireless sensor networks. In *Wireless Days (WD), 2010 IFIP*, pages 1–6, oct. 2010.

[4] Shahid Raza, Simon Duquennoy, Tony Chung, Dogan Yazar, Thiemo Voigt, and Utz Roedig. Securing Communication in 6LoWPAN with Compressed IPsec. In *Proceedings of the 7th IEEE International Conference on Distributed Computing in Sensor Systems (IEEE DCOSS 2011)*, Barcelona, Spain, June 2011.

[5] J. Hui and P. Thubert. Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks. RFC 6282 (Proposed Standard), September 2011. `http://www.ietf.org/rfc/rfc6282.txt`.

[6] Texas Instruments, Inc. Z-Stack OS Abstraction Layer Application Programming Interface, 2007. document number:F8W-2003-0002.

[7] Thu-Thuy Do, Daeyoung Kim, and Tomás Sánchez López. An evolvable operating system for wireless sensor networks. *International Journal of Software Engineering and Knowledge Engineering*, 15(2):265–270, 2005.

[8] U. Blumenthal and B. Wijnen. User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3). RFC 3414 (Standard), December 2002. `http://www.ietf.org/rfc/rfc3414.txt`.

[9] Siarhei Kuryla and Jürgen Schönwälder. Evaluation of the Resource Requirements of SNMP Agents on Constrained Devices. In *Managing the Dynamics of Networks and Services*, volume 6734, pages 100–111. Springer Berlin / Heidelberg, 2011. `http://dx.doi.org/10.1007/978-3-642-21484-4_13`.

[10] ANSI X9.62. ”Elliptic Curve Key Agreement and Key Transport Protocols”. American Bankers Association, 1999.

[11] ANSI X9.63. ”The Elliptic Curve Digital Signature Algorithm”. American Bankers Association, 1999.

[12] Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid. Recommendation for key management - part 1: General (revised. In *NIST Special Publication*, 2006.

[13] Vipul Gupta et al. Sizzle: A standards-based end-to-end security architecture for the embedded internet. In *Pervasive Computing and Communications*, pages 247–256, 2005.

[14] Xiangqian Chen, Kia Makki, Kang Yen, and Niki Pissinou. Sensor network security: a survey. *IEEE Communications Surveys Tutorials*, 11(2):52–73, 2009.

[15] Yun Zhou, Yuguang Fang, and Yanchao Zhang. Securing wireless sensor networks: a survey. *Communications Surveys & Tutorials, IEEE*, 10(3):6–28, 2008.

[16] Seyit A. Camtepe and Bülent Yener. Key distribution mechanisms for wireless sensor networks: a survey. Technical report, Rensselaer Polytechnic Institute, March 2005.

[17] P. Wouters, J. Gilmore, S. Weiler, T. Kivinen, and H. Tschofenig. TLS Out-of-Band Public Key Validation draft-ietf-tls-oob-pubkey-01.txt. Expires July 10, 2012.

[18] D. McGrew and D. Bailey. AES-CCM Cipher Suites for TLS draft-mcgrew-tls-aes-ccm-02. Expires May 3, 2012.

[19] Z. Shelby, K. Hartke, C. Bortmann, and B. Frank. Constrained Application Protocol (CoAP) draft-ietf-core-coap-08. Expires May 4, 2012.

[20] Oliver Gay. Software implementation in C of FIPS 198 Keyed-Hash Message Authentication Code HMAC for SHA2. `http://www.ouah.org/ogay/hmac/`, Accessed January 2012.

[21] The OpenSSL Project. OpenSSL: The open source toolkit for SSL/TLS, April 2003. `www.openssl.org`.

[22] V. Perelman, J. Schoenwaelder, M. Ersue, and K. Watsen. Network Configuration Protocol Light (NETCONF Light) draft-schoenw-netconf-light-01. Expires July 21, 2012.

[23] M. Badra. NETCONF Over Transport Layer Security (TLS), draft-badra-netconf-rfc5539bis-00. Expires April 25, 2012.

[24] P. Eronen and H. Tschofenig. Pre-Shared Key Ciphersuites for Transport Layer Security (TLS). RFC 4279 (Proposed Standard), December 2005. `http://www.ietf.org/rfc/rfc4279.txt`.