Tiny 3-TLS: A Trust Delegation Protocol for Wireless Sensor Networks

Sepideh Fouladgar, Bastien Mainaud, Khaled Masmoudi, and Hossam Afifi

Institut National des Télécommunications, 9 rue Charles Fourier,
91011 Evry Cedex, France
{sepideh.fouladgar,bastien.mainaud,khaled.masmoudi,
hossam.afifi}@int-evry.fr

Abstract. Adapting security protocols to wireless sensor networks architectures is a challenging research field because of their specific constraints. Actually, sensors are computationally weak devices, unable to perform heavy cryptographic operations like classical asymmetric algorithms (RSA, Diffie-Hellman). In this paper, we introduce Tiny 3-TLS, an extension and adaptation of TLS handshake sub-protocol that allows establishing secure communications between sensing nodes and remote monitoring terminals. Our protocol aims at guaranteeing the integrity and confidentiality of communications between sensors and distant terminals, after having established mutual authentication between the two parties. In order to achieve these security goals without putting too much burden on sensing devices, Tiny 3-TLS rely on an intermediate node, the sink node. Depending on the trustworthiness of this sink node and on the applications, we propose two versions of our proposition. Besides, we provide a formal validation of the protocol's security goals achievement and an evaluation of its computation and delay performances.

1 Introduction

Wireless sensor networks (WSN) have been generating much interest in the last few years. The need for efficient security is more and more appealed since the most of sensor applications require wireless communications for flexible deployment purposes. Besides, the accuracy and the integrity of the conveyed data may be of very high importance. For instance, WSN may be deployed in e-health applications, including medical monitoring and remedies administration, which are critical as the patients' lives are at stake. Most WSN architectures rely on a central node that collects and processes the data. It is generally an entity located outside the WSN, and is linked to it through an interconnecting architecture (usually the Internet), either an ad hoc or an infrastructure-based network. The interconnection between WSN and Internet is made by means of a gateway acting as a radio base station: the sink. This gateway modifies the message from the WSN in order to be compliant to Internet, usually IP-based, protocols. The complete operation of this gateway is not detailed in this document.The need for security is a request of WSN community. On the one hand,

L. Buttyan, V. Gligor, and D. Westhoff (Eds.): ESAS 2006, LNCS 4357, pp. 32–42, 2006.

the major constraint is the weak resources of such nodes in terms of memory, processing capacity and power consumption. That's why it is not possible to use classical cryptographic algorithms and security protocols in networking architectures that include WSNs. On the other hand, TLS (Transport Layer Security protocol) [6] has become the de facto secure application-level tunneling protocol; in order to adapt it in the context of wireless sensor networks, we propose a solution which enables establishing TLS tunnels between a node of the WSN, and a monitoring remote device. The negotiation may rely on the sink node to perform as much cryptographic operations as possible. The remainder of this paper will unfold as follows: the next section will focus on the existing tunneling technologies and the security mechanisms for WSN. Next, we describe our solution to support traditional sensor networks security mechanisms, followed by an evaluation of the computation time of our protocol over Avrora platform [1] and a formal validation of the protocol from a security point of view using an automatic protocol analyzer. Finally we conclude and provide some possible extensions to our work.

2 Related Work

TLS, is an application-independant set of protocols that enables encryption, authentication and integrity for data exchanged between a client and a server. TLS consists of many subprotocols, among which Handshake protocol. This latter allows a client and a server to negotiate a cyphersuite, authenticate each other and obtain a shared master key, usually using public key algorithms. Once the shared master key established, the two parties derive symmetric keys and use symmetric algorithms for fast encryption and authentication of application data. Thus, TLS Handshake protocol uses public key technology to support symmetric key management. Although many prior security proposals for sensor networks considered that sensor constraints were incompatible with public key cryptography, many more recent work showed that public key technology can also be deployed in the realm of sensor networks. Watro et al.[3] conceived a security scheme, Tiny PK, based on public key technology, for providing authentication and key exchange between an external party and a sensor network. The fact is that TinyPK is based on a precautionous implementation of RSA cryptosystem albeit Elliptic Curve Cryptography (ECC) appears as an alternative to RSA for resource constrained devices. Indeed, ECC can offer equivalent security for fairly smaller keys. Moreover, TinyPK uses checksums to insure message integrity during its key management protocol, whereas checksums have shown to be poor and easily misled integrity mechanisms. An end-to-end security architecture for low power devices (Sizzle), which lies on ECC, has been implemented by Gupta et al.[4]. It allows embedding a secure web server in low power devices for monitoring and control purposes. Sizzle architecture is composed of a control station located somewhere in the internet, the sensors being controlled and a gateway that serves as a bridge between these two elements. The gateway connects to the internet using an ethernet-like high-speed link and connects to the sensors via a

lower-speed wireless link such as 802.15.4 [13]. In Sizzle, the gateway does not perform any cryptographic operation as all data stays encrypted when crossing the gateway. It just transmits the messages between the control station and the sensors. As a consequence, all the burden of cryptographic operations is on the sensors. Moreover the gateway does not authenticate itself neither to the control station nor to the sensors; and these, are not authenticated either. Therefore, man in the middle attacks are easily achievable.

3 Tiny 3-TLS

3.1 Trust Model and Security Goals

The goal of our solution is to provide an end-to-end secure communication between a remote device and a wireless sensor network . Tiny 3-TLS achieves the following security functionalities:

- *injective agreement on a shared session key between a remote terminal and the WSN, possibly through the help of the gateway,*
- *mutual authentication between the gateway and the remote node*

These goals are validated using an automatic protocol analyser, as described in section 5.

Even though classical TLS Handshake achieves these goals, it is not adapted as is to our context. We use concepts from IEEE 802.1X standard [15] trust model to build trust between the WSN, the gateway and the remote node (see figure 1). In this scheme, the WSN acts as the authenticator (the resource), the remote node as the supplicant and the gateway as the authentication server.

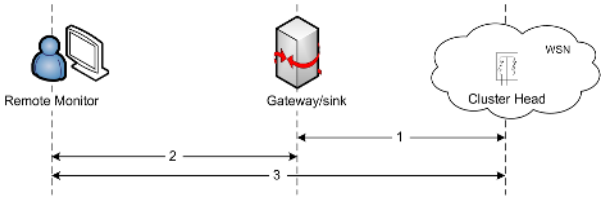

Fig. 1. Trust establishment sequence

Figure 1 shows the pre-establishment of trust between the security gateway and the group of sensors in (1). Once trust is established between the gateway and the remote node (2), it is transitiveley achieved between the remote node and the WSN (3).

3.2 Problem Statement

Partially Trusted Versus Fully Trusted Gateway:
Tiny 3-TLS adapts TLS handshake sub-protocol in order to have secure communications between a remote client terminal and a sensor network. To balance

sensors' low computational capabilities, the Tiny 3-TLS architecture is based on a third party [5], the security gateway (GW) that assists the sensors for cryptographic computations.

Henceforth, we consider two cases:

- In the first case, the security gateway is partially trusted by the sensors and will only help the two parties to authenticate each other. Loosely speaking, by partially trusted, we mean that the gateway introduces the group of sensors to the distant terminal and reciprocally this latter to the sensors, but will not interfere further in the sensors/terminal relationship. In fact, this mutual authentication will help establishing a shared secret, unknown to the gateway, between the two parties, allowing the data exchanged between the terminal and the sensors to remain encrypted when crossing the gateway. Thus, at the end of Tiny 3-TLS handshake, we will have a secure end to end tunnel between the two entities.
- In the other case, the security gateway is fully trusted. That is to say, the gateway will not only help the authentication between the two parties but will also possess the shared secret between both entities.

Use Cases:

Among project MAGNET-Beyond [14] scenarios, one could consider *MAGNET.Care*. In this scenario, a patient is connected to the external world through his Body Area Network (BAN), which includes a set of sensors reporting health data like the current temperature and blood pressure to a coordinator/receiver, acting as a cluster head, which in turn sends reports to a remote monitoring device. The coordinator is connected to the external world by means of a gateway. Whenever the remote monitor polls the BAN for data, the gateway acts as a reverse proxy, authenticates the monitoring device and then grants access.

One possible application of the partially trusted gateway scenario could be that of an attending medical practitioner who wants to monitor his patient's

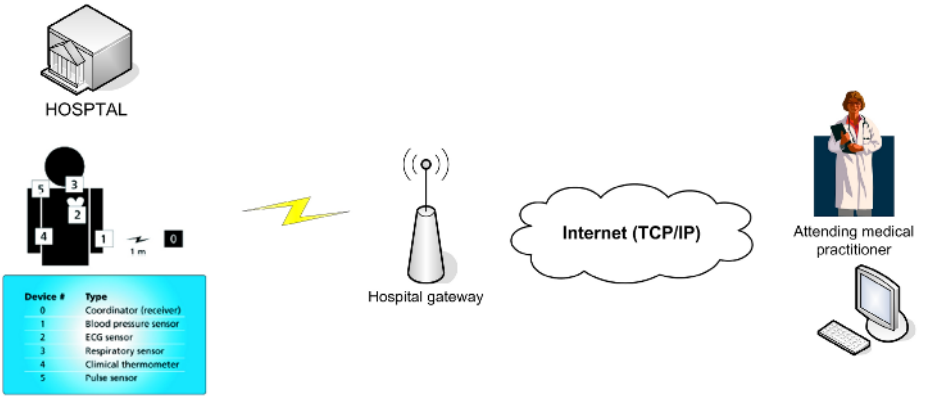

Fig. 2. A partially trusted gateway use case

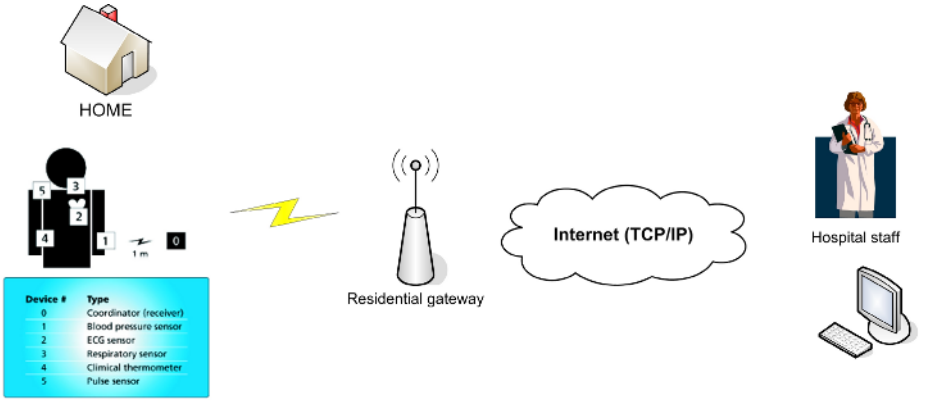

Fig. 3. A fully trusted gateway use case

health condition while he is at the hospital. He can connect his laptop to the hospital gateway (Security Gateway) and query the sensors connected to the patient. Then the gateway authenticates the laptop to the patients sensors. However, if the gateway is not fully trusted, it won't see the information exchanged between the physician laptop and the sensors, for evident privacy reasons. This use case is shown in figure 2.

On the other hand, if the patient is at home and, the physicians at the hospital want to retrieve some health data (see figure 3), the monitoring device authenticates to the patient's residential gateway. The reverse proxy in this case is part of the personal network of the patient and therefore should be fully trusted.

Protocol Assumptions and Statements:
Tiny 3-TLS handshake involves three parties among which a sensor network. When the gateway is partially trusted, in order to establish a shared secret master key which is unknown to the gateway between the sensors and the distant terminal, Tiny 3-TLS uses ECDH (Elliptic Curve Diffie-Hellman) key agreement protocol [7, 8]. In fact, asymmetric cryptography is necessary in order to distribute safely among the targeted entities, the shared secret, even if the gateway is watching. As precised above, ECC (Elliptic Curve Cryptography) offers asymmetric cryptography with considerably lower computational burden and smaller key sizes than traditional asymmetric cryptosystems and thus, is fully convenient to a protocol involving a sensor network. On the other hand, when the gateway is totally trusted, the classical TLS key agreement is used, since the shared master key no more needs to be hidden from the security gateway.

In this paper, we have considered that the handshake is done between the client terminal and a cluster head sensor. Once the master key (shared secret between the two entities) is derived, the cluster head sensor will broadcast the keys in a secure manner to other sensors. This broadcast is out of the scope of this paper.

We assume that the security gateway and the sensor network share a symmetric key K that is used to encrypt any message between both entities. We will use in the table 1 syntax to describe Tiny 3-TLS.

Table 1. Figure 1 and 2 syntax

$BigAlice, GW, TinyBob$	Principals
K	Symmetric Key shared between the gateway and TinyBob
PK_x	Public key of principal x
ID_x	Identifier of principal x
$Cert_x$	Certificate of principal x
H(.)	Hash function
$ECDH_x$	Elliptic curve Diffie-Hellman public values of principal x
PK_x^{-1}	Private key of principal x
$\{M\}\_K$	M encrypted with key K
N_x	Nonce generated by principal x
P_x	Ciphersuite offer by principal x
$x\|y$	x concatenated to y
PMS	Pre-Master Secret
M	Concatenation of all previously exchanged messages between BigAlice and the gateway

3.3 Case 1: The Security Gateway Is Partially Trusted

In this case, the security gateway GW supports the remote terminal (BigAlice) and the sensor network (TinyBob) in sharing a secret, though without possessing it. First of all, BigAlice sends a Client Hello message that contains its identifier, the SessionID, a ciphersuite offer and a nonce (1). This message is encrypted with K symmetric key and forwarded by GW to TinyBob (2). This latter replies with a Server Hello message including its identifier, the SessionID, a nonce, a ciphersuite counteroffer and its ECDH public values (3). GW keeps the ECDH values for itself and transmits to BigAlice a Server Hello message containing the SessionID and TinyBob's identifier, nonce and cyphersuite counteroffer. It also conveys its own certificate and a certificate request to BigAlice (4). Hence, BigAlice responds with its certificate, its ECDH public values and a newly generated nonce (gateway authentication nonce, $N'_{BigAlice}$), both encrypted with GW public key (recovered from GW certificate) and a signature of its ECDH public values, TinyBob's nonce and identifier (5). The gateway authenticates BigAlice and recovers its ECDH public values and the "gateway authentication nonce". It ciphers this latter and TinyBob's ECDH public elements, with BigAlice public key and transmits the ciphertext to BigAlice (6). The fact that the gateway could decipher the "gateway authentication nonce" and send it back to BigAlice, authenticates the gateway. Likewise, it sends to TinyBob, BigAlice's ECDH public values and a hash of all previously exchanged messages between the gateway and BigAlice, all being encrypted by K (7). Finally, TinyBob and BigAlice can communicate directly and exchange "Finished" messages (8, 9) where

$$Finished = H(R, M) \; where \; R = PRF(DHK, N_{BigAlice}, N_{TinyBob}),$$
$$DHK \; being \; ECDH \; agreed \; key.$$

BigAlice sends its "Finished" message to TinyBob encrypted with BigAliceMasterKey. Likewise, TinyBob sends its "Finished" message to BigAlice encrypted with TinyBobMasterKey.

$$BigAliceMasterKey = KeyGen(ID_{BigAlice}, N_{BigAlice}, N_{TinyBob}, R)$$
$$TinyBobMasterKey = KeyGen(ID_{TinyBob}, N_{BigAlice}, N_{TinyBob}, R) \text{ (figure 4).}$$

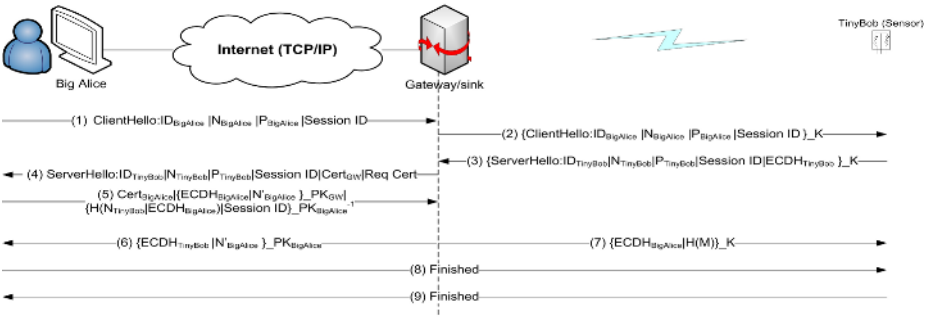

Fig. 4. The gateway is partially trusted

3.4 Case 2: The Security Gateway Is Fully Trusted

In this case, all the communication between TinyBob and BigAlice can be seen in the clear by the gateway. Client and Server Hello messages are identical to previous case (messages 1-4), except for the third message which do not contain TinyBob's public Diffie-Hellman elements. Once Client and Server Hello messages exchanged, BigAlice generates a symmetric pre-master secret (PMS). It responds to the gateway with its certificate, PMS and TinyBob's nonce encrypted with GW public key (recovered from GW certificate) and a signature of PMS, TinyBob's nonce and identifier. Big Alice adds a "Finished" message (5) where :

$$Finished = H(R, H(M)) \; where \; R = PRF(PMS, N_{BigAlice}, N_{TinyBob})$$

Once the "Finished" message received from BigAlice, the gateway generates a nonce, a Client-read-key and a Client-write-key:

$$Client - write - key = KeyGen(ID_{BigAlice}, N_{BigAlice}, N_{TinyBob}, R)$$
$$Client - read - key = KeyGen(ID_{TinyBob}, N_{BigAlice}, N_{TinyBob}, R)$$

Then, the gateway encrypts these three elements with K and transmits the cyphertext to TinyBob (6). TinyBob decrypts the cyphertext and sends back a hash of its identifier and the nonce generated by the gateway (7). Finally the gateway sends BigAlice a "Finished" message (8) (figure 5).

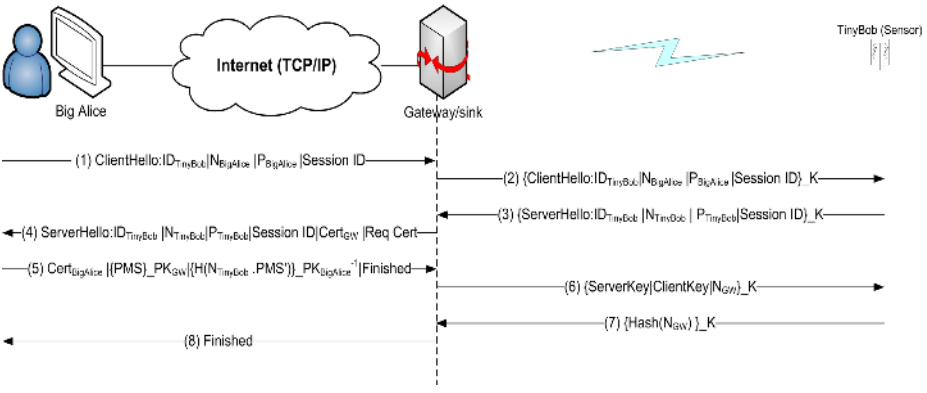

Fig. 5. The gateway is fully trusted

4 Performance Evaluation of Tiny 3-TLS

We have emulated our protocol from the sensor side by means of Avrora for the analysis of the execution time. Avrora (Beta 1.6 version released in July 2005) is an AVR (Advanced Virtual Risc) Simulation and Analysis Framework developped by the UCLA Compilers Group. It provides some information (time consumption, sleep period, energy consumption) about the application downloaded on a sensor. It emulates the code processing on a MICA2 sensor. MICA2 uses a 7,37 Mhz single processor board (MPR2400CA) with 128 KB of EEPROM for instructions and 4 KB for data. The application is written in Nesc for TinyOS 1.x.

In this paper we aimed at limiting the number of computations made by the sensors, specially public key cryptography. In the case where a gateway is fully or at least partially trusted (i.e trusted for authentication), this latter can help decreasing the burden on sensors. Indeed, there are few cases where the gateway is unknown or publicly available. Sizzle [4] is an end-to-end security architecture for low power devices. This solution is the closest of the objectives of our architecture. There are few differences between the both solutions and the following table illustrates the advantages and the drawbacks, in terms of computation, of our two use cases (Partially and Fully trusted) compared to [4] proposition.

Table 2. Comparison of Partially and Fully trusted use cases with Sizzle[4]

Tiny 3-TLS version	Partially trusted	Fully trusted
Tiny 3-TLS additional operations	- 1 symmetric encryption - 1 symmetric decryption	- 1 symmetric encryption - 1 symmetric decryption
Sizzle [4] additional operations	- 1 signature verification	- 1 Diffie-Hellman shared key generation - 1 signature verification - "Finished" messages calculations

In comparison with "Sizzle", both versions of Tiny 3-TLS protocol perform one additional symmetric encryption and one additional decryption operation while performing less asymmetric cryptographic computations.

In the Partially Trusted Gateway use case, symmetric decryption of the Client Hello message sent by the gateway to the sensor (message 2) and the encryption of the sensor response (message 3) last 44,8 ms and 156,8 ms, respectively. A signature verification, even in an optimized implementation will always last longer by at least two orders of magnitude. This confirms the fact that Partially Trusted Gateway scenario is less time and energy expensive than Sizzle. In a more obvious fashion, the Fully Trusted Gateway use case is also advantageous in terms of time and energy consumption. Indeed, both symmetric encryption and decryption of Client Hello messages last 44,8 ms. Besides, the most costly operation, that is key agreement, is delegated to the gateway.

5 Security Analysis and Formal Validation

AVISPA Security Analyser:
In order to analyze the security of Tiny 3-TLS, we used Automatic Validation of Internet Protocols and Application (AVISPA) tool, a security protocol analyzer [2]. AVISPA uses a High Level Protocol Specification Language (HLPSL) [9] to describe security protocols and specifying which security goals are achieved by a given one. HLPSL is an expressive and straightforward language, based on the work of Lamport on Temporal Logic of Actions [12]. Communication channels are represented by the variables carrying different properties of a particular environment. We have used OFMC [11] tool since it provides support for specific algebraic properties, in our case the exponential operator used for Diffie-Hellman key agreement in the first case. We correctly compiled our HLPSL model and validated Tiny 3-TLS. The output of the analyzer is provided in table 3.

The Attacker Model:
We have used Dolev-Yao intruder model [10] in which all communications with the intruder are synchronous. In other words, the intruder is in full knowledge of all messages to and from the honest participants. In this model, the attacker can not lead physical attacks against legitimate entities, however, it can participate to the protocol, generating its own messages, replaying old messages and eavesdropping on communications between the different entities. Though, we test our protocol against replay, identity theft, information leakage and man in the middle attacks.

In both cases, the output shows that the security goals are reached after the validation process and that the protocol is safe (that is no attack threatening the specified goals was found). These goals are 1) Mutual strong authentication between BigAlice and GW,and 2) secrecy of Client-write-key and Client-read-key.

Table 3. OFMC output of AVISPA security analyzer

Partially trusted	Fully trusted
% Version of 2005/06/07	% Version of 2005/06/07
SUMMARY	SUMMARY
SAFE	SAFE
DETAILS	DETAILS
BOUNDED_NUMBER_OF_SESSIONS	BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL	PROTOCOL
GOAL	GOAL
as_specified	as_specified
BACKEND	BACKEND
OFMC	OFMC
COMMENTS	COMMENTS
STATISTICS	STATISTICS
parseTime: 0.00s	parseTime: 0.00s
searchTime: 25.21s	searchTime: 4.87s
visitedNodes: 5206 nodes	visitedNodes: 1499 nodes
depth: 13 plies	depth: 11 plies

6 Conclusion and Future Work

In this paper, we proposed Tiny 3-TLS, an extension to TLS handshake that helps establishing end-to-end tunnels between nodes in a wireless sensor network and an external remote terminal. Contrary to other propositions, we rely on the sink node as an intermediate for trust establishment, since it is a fundamental entity in any network architecture that includes sensors.

Depending on the trust model of the sink node, we designed two versions of the protocol with the objective of relieving as far as possible the low-capacity node, that is the sensor, from the burden of costly cryptographic operations and the transmission of their results. Another design challenge was to introduce as few new messages as possible. The resulting protocol, in both versions, does not introduce any change in TLS handshake implementation from the client side.

Finally, we formally validated the new protocol using an automatic protocol analyzer, AVISPA. We are currently implementing the whole protocol and we will consider in future work the dissemination of the generated keys to other sensors of the cluster and network in order to support one-to-many communication security based on TLS.

Acknowledgment

This paper describes work undertaken in the context of the research project IST-MAGNET. MAGNET Beyond is a continuation of the MAGNET research project (www.ist-magnet.org). MAGNET Beyond is a worldwide R&D project within Mobile and Wireless Systems and Platforms Beyond 3G. It will introduce new technologies, systems, and applications that are at the same time user-centric and secure. MAGNET Beyond will develop user-centric business model concepts for secure Personal Networks in multi-network, multi-device, and multi-user environments. It has 30 partners from 15 countries, among these highly influential Industrial Partners, Universities, Research Centers, and SMEs.

References

1. http://compilers.cs.ucla.edu/avrora/, the Avrora project homepage.
2. http://www.avispa-project.org/, the AVISPA project homepage.
3. Watro, R., Kong, D., Cuti, S., Gardiner, C., Lynn, C., Kruus, P.: *TinyPK: Securing Sensor Networks with Public Key Technology.* In ACM Workshop on Security of Ad Hoc and Sensor Networks, October 2004.
4. Gupta, V.,Millard, M., Fung, S., Zhu, Y., Gura, N., Eberle, H.,Shantz, S.C.: *Sizzle: A Standards-based end-to-end Security Architecture for the Embedded Internet.* In Third IEEE International Conference on Pervasive Computing and Communications, March 2005.
5. Masmoudi, K., Hussein, M., Afifi, H., Seret, D.: *Tri-party TLS Adaptation for Trust Delegation in Home Networks.* In IEEE International Conference on Security and Privacy for Emerging Areas in Communication Networks, September 2005.
6. Dierks, T., Rescorla, E.: *The Transport Layer Security (TLS) Protocol - Version 1.1.* IETF RFC 4346, April 2006.
7. Koblitz, N.: *Elliptic Curve Cryptosystems.* Mathematics of Computation, 48:203-209, 1987.
8. Miller, V.: *Uses of Elliptic Curves in Cryptography*, In Advances in Cryptology, CRYPTO'85, LNCS 218, Springer-Verlag, pp. 417-462, 1985.
9. Chevalier, Y. et al.: *A High-Level Protocol Specification Language for Industrial Security-Sensitive Protocols: www.avispa-project.org*
10. Dolev, D. and Yao, A.:*On the Security of Public-Key Protocols.* IEEE Transactions on Information Theory, 2(29), 1983.
11. Basin, D., Modersheim, S. and Viganno, L.: *OFMC: A Symbolic Model-Checker for Security Protocols.* International Journal of Information Security, 2004.
12. Lamport, L.: *The temporal logic of actions.* ACM Transactions on Programming Languages and Systems, 16(3):872923, May 1994.
13. *Wireless medium access control and physical layer specifications for low-rate wireless personal area networks.* IEEE Standard, 802.15.4-2003, May 2003. ISBN 0-7381-3677-5
14. http://www.ist-magnet.org, IST MAGNET-Beyond project homepage.
15. IEEE Std. 802.1X-2004, Standards for Local and Metropolitan Area Networks: Port Based Network Acces Control.