Providing Tunable Security in IEEE 802.11i Enabled Networks
Providing Tunable Security in IEEE 802.11i Enabled Networks

Zoltán Faigl\textsuperscript{1}    Stefan Lindskog\textsuperscript{2}    Anna Brunstrom\textsuperscript{2}
Katalin Tóth\textsuperscript{1}

\textsuperscript{1}Department of Telecommunications
Budapest University of Technology and Economics, Hungary
\{szlaj|ktoth\}@mcl.hu

\textsuperscript{2}Department of Computer Science
Karlstad University, Sweden
\{Stefan.Lindskog|Anna.Brunstrom\}@kau.se

Abstract

The basic idea of QoS is to provide mechanisms that can offer different service levels, which are expressed through well-defined parameters that are specified at run-time on the basis of need. Bit rate, throughput, delay, jitter, and packet loss rate are all examples of common QoS parameters suggested for packet networks. These parameters are all aimed to express (and guarantee) a certain service level with respect to reliability and/or performance. In this report, we investigate how security can be treated as yet another QoS parameter through the use of tunable security services. The main idea with this work is to let users specify a trade-off between security and performance through the choice of available security configuration(s). The performance metric used is latency. The concept is illustrated using the IEEE 802.11i wireless local area networking standard.

\textbf{Keywords:} Network security, tunable security, security services, wireless networks, latency estimation, QoS, IEEE 802.11i.
Acknowledgments

This research is carried out as part of the work on the development of dynamic security services in heterogeneous wireless networks within NEWCOM. The present report is the result of a joint work between Karlstad University (KaU) in Sweden and Budapest University of Technology and Economics (BUTE) in Hungary. The work at KaU is supported by grants from the Knowledge Foundations of Sweden with TietoEnator and Ericsson as industrial partners.
### Graphical Analysis

- Graphical analysis using the ROME framework: 35
- Graphical analysis with R: 36
- Estimated latencies of the MAC data frames: 36

#### B.2 Computational Latency Calculations

- B.2.1 Number of Occurrences: 37
- B.2.2 Latencies of the Cryptographic Algorithms: 44
- B.2.3 Total latency of the security configurations: 44

#### C Ethereal Trace

#### D Detailed Description of Considered Security Configurations

- D.1 TLS_DHE_DSS with *.*: 86
  - D.1.1 Message Complexity: 86
  - D.1.2 Computational Complexity: 86
  - The length of the applied constants and messages in bytes: 86
  - Computations to generate and check the TLS handshake messages: 89
- D.2 TLS_DHE_RSA with *.*: 92
  - D.2.1 Message Complexity: 92
  - D.2.2 Computational Complexity: 92
  - The length of the applied constants and messages in bytes: 92
  - Computations to generate and check the TLS handshake messages: 95
- D.3 TLS_RSA with *.*: 98
  - D.3.1 Message Complexity: 98
  - D.3.2 Computational Complexity: 98
  - The length of the applied constants and messages in bytes: 98
  - Computations to generate and check the TLS handshake messages: 101
- D.4 TLS_DH_DSS with *.*: 105
  - D.4.1 Message Complexity: 105
  - D.4.2 Computational Complexity: 105
  - The length of the applied constants and messages in bytes: 105
  - Computations to generate and check the TLS handshake messages: 108
- D.5 TLS_DH_RSA with *.*: 110
  - D.5.1 Message Complexity: 110
  - D.5.2 Computational Complexity: 110
  - The length of the applied constants and messages in bytes: 110
  - Computations to generate and check the TLS handshake messages: 113

#### E Results of the Graphical Analysis of MAC Latency in ROME

- E.1 Graphical Results of Experiment 1: 116
  - Dependency of the MAC layer latency on the inter arrival time, the data frame length and the number of stations: 116
Dependency of the channel utilization on the inter arrival time, the data frame length, and the number of stations. ........................................ 122

Dependency of the block length on the inter arrival time, the data frame length, and the number of stations. 127

Dependency of the MAC layer latency on the channel utilization, the data frame length, and the number of stations. ........................................ 132

Dependency of the block length on the channel utilization, the data frame length, and the number of stations. 137

E.2 Graphical Results of Experiment 2 ................................. 142

Dependency of the MAC layer latency on the channel utilization and the data frame length. .................. 143

Dependency of the block length on the channel utilization and the data frame length. .................. 150

Dependency of the number of (re)sends of RTS frames on the channel utilization and the data frame length. 157

Dependency of the block length on the number of (re)sends of RTS frames and the data frame length. ........ 164

Dependency of the MAC layer latency on the waiting queue length and the number of (re)sends of RTS frames. 171

F Results of the Graphical Analysis of MAC Layer Latency in R 179

F.1 Distribution Function of the MAC Layer Latency ................. 180

F.1.1 Analysis of the MAC Layer Latency on a Logarithmic Scale 181

F.1.2 Analysis of the MAC Layer Latency on a Linear Scale ... 186
1 Introduction

The popularity of the Internet is still growing and Internet is now used for both private and commercial purposes. New networked applications are steadily emerging and some of them put new demands on the underlying network. To meet such demands, the concept of quality of service (QoS) [36] has evolved. The basic idea of QoS is to provide mechanisms that can offer different service levels, which are expressed through well-defined parameters that are specified at run-time on the basis of need. Bit rate, throughput, delay, jitter, and packet loss rate are all examples of common QoS parameters in packet networks. QoS parameters have been studied extensively and are all aimed to express (and guarantee) a certain service level with respect to reliability and/or performance.

Another increasingly important issue in computer networks is security. Traditionally, research on security has been focusing on methods that provide as much security as possible [16]. However, with the advent of many low-power computing and communication devices, it has now become desirable to trade security against other QoS parameters. The current lack of mechanisms by which system owners and users can request a specific level of security as a service in the system typically makes it impossible to offer security based on need. All users are instead offered similar services, regardless of whether it is the desired level of security, and all users are forced to bear the costs of either too much or too little protection.

By providing tunable security (TS) services, security can be treated as a QoS parameter. With such a treatment service levels can be selected on demand. The main objective of this report is to demonstrate how a security level can be selected and traded against performance in terms of latency in a wireless network setting. Trade-offs are specified through the choice of available security method(s). The concept is illustrated using the IEEE\(^1\) 802.11i standard [11], which is a security addendum to the 802.11 wireless local area networking standard [9]. In this report, our focus is on the different authentication configurations that can be selected at run-time when establishing a security association (SA) in 802.11i using open system authentication.

The remainder of the report is organized as follows. In Section 2, related work is presented. Section 3 provides an overview of the security services provided in 802.11i. The application scenario considered in this report is presented in Section 4. Possible security configurations are given in Section 5. The latencies of the respective configurations are estimated in Section 6, while Section 7 focuses on how to combine security and performance requirements. Finally, Section 8 concludes the report. In addition, Appendix A contains the abbreviations used in this work, and the remaining appendices describe the details of our analysis. Namely, Appendix B presents the steps of our calculations. Appendix C contains the trace of an EAP-TLS SA establishment process. Appendix D describes the considered security configurations and gathers the computational requirements and message flows. Appendix E contains simulation results for the estimation of 802.11 MAC layer latency at different network loads. Appendix F gives a graphical analysis of the MAC layer latency distribution function based on the

---

\(^1\)From now on, when referring to an IEEE standard the abbreviation “IEEE” is omitted. Hence, when referring to the standard IEEE 802.11 only 802.11 is written.
network load and the frame size.

2 Related Work

TS services can essentially be implemented in two fundamentally different ways: selective protection or algorithm selection [18]. Selective protection means that only parts of the data are protected. Selective protection has been suggested for both video [15, 21] and image data [25]. It has seen commercial use in Sony’s Passage technology [32] aimed for digital CATV networks. Selective encryption methods that use as a content-independent representation of the selectively encrypted data has been proposed [17, 19]. In addition, support for selective protection has also been integrated directly into several multimedia applications, e.g., Nautilus [22] and Speak Freely [33].

In this technical report we, however, focus on TS based on algorithm selection. A recent example of the use of algorithm selection is presented in [37]. The impacts of run-time security parameter changes when using the IPSec [14] protocol in a virtual private network (VPN) setting is investigated. Another example of the use of algorithm selection for IPSec can be found in [34]. Algorithm selection is also used in [13] to provide an energy efficient TS service for limited wireless devices. Algorithm selection is also used in the generalized quality of protection framework proposed by Ong et al. [23]. They propose, among other things, that an encryption service should be specified using the following parameters:

\[
\langle \text{content type, interval of security, encryption algorithm, encryption key length, encryption block length} \rangle
\]

The strength of the encryption, expressed through the last three parameters, depends on the type of content and on the time interval during which the data must be kept secure. Hager [8] has studied methods for selecting an appropriate security protocol for specific wireless network applications in order to improve the efficiency of security mechanisms. A decision model based on analytic hierarchy process is proposed. A combination of selective protection and algorithm selection is also possible, as used for instance in Authenticast [29].

The work presented in this report aims to offer a mechanism that allows trade-offs between security and performance during SA establishment in IEEE 802.11i. The proposed service provides TS through algorithm selection and the main idea is to let users specify trade-offs between security and performance through a simple high-level interface. The selection of an appropriate security configuration considers both user preferences and environmental characteristics.

3 Overview of 802.11i Security Services

802.11 is an evolving family of specifications for wireless local area networks (WLANs) developed by IEEE. 802.11i is an enhancement to 802.11 that offers additional security for WLAN applications by making use of other standards like 802.1X [10] and the extensible authentication protocol (EAP) [1]. The different standards use different terminologies. In this report the terminology of 802.11i is
Table 1: Terminology used in 802.11i, 802.1X, and EAP respectively.

<table>
<thead>
<tr>
<th>802.11i</th>
<th>802.1X</th>
<th>EAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>STA</td>
<td>Supplicant</td>
<td>EAP Peer</td>
</tr>
<tr>
<td>AP</td>
<td>Authenticator</td>
<td>Network Access Server (NAS)</td>
</tr>
<tr>
<td>AS</td>
<td>Authentication Server</td>
<td>EAP Server</td>
</tr>
</tbody>
</table>

used for the access point, the wireless device of the client and the authentication server, referred to as AP, STA, and AS, respectively. An AP is an entity that has station functionality and provides access to the distribution service through the wireless medium for associated stations, whereas an AS is an entity that verifies the identity of a station. Typically, either the remote authentication dial in user service (RADIUS) [28] or Diameter [3] is used as AS. Table 1 provides a summary and mapping of essential terms used within the different standards.

3.1 Security Services

The 802.11i standard provides a number of security features which all provide a set of configuration options. First of all, the standard defines two different methods used for SA establishment. Both operate at the link layer. The least secure variant is based on preshared keys. The other variant is based on open system authentication, which uses the 802.1X standard. 802.1X provides a framework for port-based network access control. In the 802.1X standard, EAP is used for authentication. EAP supports multiple authentication methods, e.g., EAP-MD5\(^2\), EAP-TLS\(^3\), and EAP-TTLS\(^4\). EAP-TLS offers mutual authentication between a STA and an AP, and has today become the de-facto standard for enterprise WLAN authentication and is therefore considered in this report.

3.2 Security Associations

The 802.11i protocol consists of three important steps that belong to the SA establishment, see Figure 1. When a SA has been successfully established a secure data tunnel exists and is ready for communication, which is illustrated in step 4 in the figure.

3.2.1 Step 1: Probing

The probing phase means the establishment of the communicating parties (STA and AP) and also the capability announcement of the AP. After this step, the two parties are ready for authentication, assuming that the STA has a common set of capabilities with the AP.

\(^2\)MD5 is an abbreviation for message digest algorithm number 5.

\(^3\)TLS is an abbreviation for transport layer security.

\(^4\)TTLS is an abbreviation for tunneled TLS.
3.2.2 Step 2: Authentication

The authentication phase is the most important SA establishment step from a security point of view. The message flow sent between the STA, the AP, and the AS during this step is shown in Figure 2. In case of open system authentication with 802.1x, the authentication process is running between the STA and the AS, where the AS can be configured to use EAP-TLS. The AP is transparent during the authentication process. If the authentication succeeds, the AP will receive a so-called pairwise master key (PMK) from the AS to be used for key establishment. The STA, on the other hand, calculates PMK itself based on data transferred in messages between the STA and AS.

When using EAP-TLS, the security level of both the SA establishment and the authentication is influenced by the choice of ciphersuite [6] within the TLS session, notably, the chosen type of key exchange and signature algorithm. An in-depth discussion on security configurations is given in Section 5.
3.2.3 Step 3: Key Establishment

In step 3, a 4-way handshake is performed between the STA and the AP to compute a pairwise transient key (PTK) based on the PMK and one nonce from each side. The PTK is divided into three parts and the part called temporal key (TK) is used for data encryption in step 4. The key hierarchy used in 802.11i is illustrated in Figure 3.

![Key hierarchy diagram]

Figure 3: Key hierarchy.

3.2.4 Step 4: Data Transfer

After a successful authentication of the parties, the 802.1X controlled port opens up for user data transport in the AP. A secure tunnel in the wireless link at link layer is ready for communication. The TK is used for encrypting the data with WEP, TKIP, or CCMP.

4 Application Scenario

In order to demonstrate the concept of a TS service a simple application scenario has been defined. In spite of its simplicity, it still resembles common configuration cases used in the present and in the near future. Note, however, that the scenario can later be developed by either removing our simplifications or extending it to other configuration settings.

---

5When WEP is used together with open system authentication in 802.11i it is sometimes referred to as dynamic WEP.
In the scenario, which is illustrated in Figure 4, we deal with the SA establishment using 802.11i security services in an 802.11 WLAN access network connected to a high capacity wired network. In the demonstration of our concept, we focus on how the different parameters in the authentication phase can be chosen and be traded off against performance based on user preferences and environmental characteristics.

As illustrated in Figure 4, users connect to the network using a STA, which can be a device of any type having an 802.11 WLAN network interface with support for 802.11i. A RADIUS server acts as AS and EAP-TLS is used for network access control, which implies that the RADIUS server also acts as a TLS server. In addition, a data server (DS) is considered in the application scenario, which may reside elsewhere in the wired network. The DS server provides data to the STAs in the WLAN.

The simplifications of the application scenario are threefold. First, the DS does not need to establish secure tunnels with neither the AP nor with the STA. This can be the case when the physical protection of the wired network is enough for the data transfer. In this case, the DS trusts the AS in enforcing control over the authorization of the network access of users. It also assumes that a secure tunnel will be established between the AP and STA.

Second, in the latency estimation of the SA establishment, we consider that the bottleneck for the network latency calculation is the wireless link. As a conse-
sequence, the network latency of the wired part is treated to be null and having no considerable effect on the whole latency of a service. Hence, our network latency estimation in Section 6 will consider only the latency estimation on the wireless link. This simplification could represent the case when the wired network has, e.g., only a few Gigabit Ethernet links and switches and the wireless link has 2 or 11 MB/s data rate and where the background traffic on the wired network does not much influence the wireless throughput.

Third, in our scenario the TS service is only influenced by the user preferences at the STA and not by the preferences at the DS side. This is, however, an extension that later can be considered.

5 Security Configurations

In open system authentication mode in 802.11i, the EAP protocol is used for authentication. EAP is essentially an authentication framework that supports multiple authentication methods. EAP is furthermore a lock-step protocol that only supports a single packet in flight. In an 802.11 network environment, EAP typically runs directly over the data link layer, without requiring IP [26]. The specific authentication method studied in this report is TLS. TLS is a message oriented protocol that is normally used on top of TCP [27]. It is a highly flexible and standardized protocol [6] that provides a rich set of security configurations through so-called ciphersuites.

5.1 SA Establishment using TLS

When establishing a SA in TLS, three types of control messages are exchanged between the STA and the AS: handshake, changecipherspec, and alert messages. A sequence of handshake messages are used for negotiation of security parameters. Changecipherspec messages are used to inform the other side to begin using the newly agreed security specification. Alert messages are used to signal special events and errors in both directions. Figure 5 illustrates the message flow during SA establishment in TLS.

As depicted in Figure 5, the STA initiates the SA establishment by sending a client_hello message containing protocol version and a list of preferred ciphersuites. The AS will reply with a server_hello message which includes the preferred ciphersuite followed with certificate, server_key_exchange, certificate_request, and server_done_hello messages. Note that these five messages can be sent in either the same EAP packet if they fit, or be divided into multiple EAP packets. In cases of multiple packets, each packet will be sent one by one and for each packet sent the AS must wait for a response before sending the next one. The STA replies with certificate, client_key_exchange, certificate_verify, change_cipher_spec, and finished messages, which can be sent in one or more EAP packets. Finally, the AS will respond with change_cipher_spec and finished messages. After the STA receives the finished message from the AS, the AP receives the PMK from the AS. Note that the AP only acts as a mediator between the STA and the AS during the EAP-TLS SA
establishment. The SA between the STA and AP will be created after the key establishment phase (see paragraph 3.2.3).

5.2 Ciphersuites in TLS

TLS supports a number of key exchange protocols including Diffie-Hellman (DH) and RSA. In TLS version 1.0, two different hash algorithms were defined: MD5 and SHA-1. Two stream ciphers (RC4-40 and RC4-128 where 40 and 128 specifies the key length) and five block ciphers (IDEA, RC2-40, DES40, DES, and 3DES) for bulk data encryption and decryption were also specified. All in all, 27 ciphersuite definitions were initially defined. In TLS 1.0, it is specified that all systems implementing TLS must at least support the following ciphersuite: TLS\_DHE\_DSS\_WITH\_3DES\_EDE\_CBC\_SHA. This ciphersuites specifies that the ephemeral DH (DHE) key exchange protocol, the digital signature standard (DSS) certificate, the triple DES (3DES) bulk encryption algorithm that provides encryption, decryption, and encryption (EDE) using three different keys in cipher block chaining (CBC) mode, and, finally, the secure hash algorithm (SHA) number 1 are used. When TLS was standardized it was designed to be extensible with respect to ciphersuites. For example, the addition of the AES ciphersuites is described in [5].

In common for all the above mentioned ciphersuites are that they are based
on public key certificates for authentication. Alternatives to these methods are ciphersuites based on Kerberos [20] and ciphersuites based on pre-shared keys [7].

### 5.3 Considered Ciphersuites

The most commonly used ciphersuites are based on public key certificates and we therefore restrict our attention in this report to such ciphersuites. We have also excluded all the export restricted ciphersuites, due to the fact that they are not used any more. In addition, the 802.11i standard requires mutual authentication in open system authentication mode, which excludes all ciphersuites that use anonymous DH (DH$_{anon}$) key exchange.

Remember also that TLS in this case is used only for authentication and key exchange. This implies that the choice of bulk encryption algorithm and hash algorithm will have no direct effect, since these algorithms will not influence the SA establishment. Hence, TLS$_{DHE}$DSS$_{WITH}$AES$_{256}$CBC$_{SHA}$ is equal to TLS$_{DHE}$DSS$_{WITH}$RC4$_{128}$SHA when TLS is used for SA establishment only. However, in some TLS implementations the bulk encryption algorithm will indirectly influence the key length of the signature algorithm [4]. This is due to the fact that there is no mechanism to allow users to specify a minimal key length. This implicit relation should of course be taken into consideration when constructing a production system, but for simplicity we do not here assume that the symmetric key length will affect the key length of the signature algorithm.

A ciphersuite specifies the signature algorithm used at the AS side for public key certificates. For each ciphersuite, the client side may have the possibility to influence the choice of signature algorithm for public key certificates. The client could have either a RSA or DSS signed certificate. They could be issued for RSA or DSS signature certification, or for certification of the fixed DH parameters of the client. Table 2 lists the security configurations considered in the rest of this report. For each security configuration, an identification number has been assigned for later references in this report. A '*' in the table denotes that the choice of this type of algorithm has no effect on the SA establishment.

<table>
<thead>
<tr>
<th>ID</th>
<th>Ciphersuite</th>
<th>Client certificate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a</td>
<td>TLS$<em>{DHE}$DSS$</em>{WITH}$<em>.</em></td>
<td>DSS signature</td>
</tr>
<tr>
<td>1b</td>
<td>TLS$<em>{DHE}$DSS$</em>{WITH}$<em>.</em></td>
<td>RSA signature</td>
</tr>
<tr>
<td>2a</td>
<td>TLS$<em>{DHE}$RSA$</em>{WITH}$<em>.</em></td>
<td>DSS signature</td>
</tr>
<tr>
<td>2b</td>
<td>TLS$<em>{DHE}$RSA$</em>{WITH}$<em>.</em></td>
<td>RSA signature</td>
</tr>
<tr>
<td>3a</td>
<td>TLS$_{RSA}$WITH*.*</td>
<td>DSS signature</td>
</tr>
<tr>
<td>3b</td>
<td>TLS$_{RSA}$WITH*.*</td>
<td>RSA signature</td>
</tr>
<tr>
<td>4a</td>
<td>TLS$<em>{DH}$DSS$</em>{WITH}$<em>.</em></td>
<td>DSS fixed DH</td>
</tr>
<tr>
<td>4b</td>
<td>TLS$<em>{DH}$DSS$</em>{WITH}$<em>.</em></td>
<td>RSA fixed DH</td>
</tr>
<tr>
<td>5a</td>
<td>TLS$<em>{DH}$RSA$</em>{WITH}$<em>.</em></td>
<td>DSS fixed DH</td>
</tr>
<tr>
<td>5b</td>
<td>TLS$<em>{DH}$RSA$</em>{WITH}$<em>.</em></td>
<td>RSA fixed DH</td>
</tr>
</tbody>
</table>
### 5.4 Ranking of Security Configurations

The choice of a specific ciphersuite will affect the provided level of security. AES with a key length of 128 bits is, for example, less secure than AES with a key length of 192 bits, which in turn is less secure than AES with a key length of 256. The same arguing can be done on key exchange protocols, signature algorithms, and hash algorithms. Note that it is not always possible to fully order different security algorithms with respect to the provided level of security. When combining different types of algorithms, like in the TLS ciphersuites, the task of ranking them becomes even harder. The ranking of TLS ciphersuites with respect to security strengths that are used in this report is based on guidelines from the national institute of standards and technology (NIST) [4]. The TLS security configurations listed in Table 2 are ordered in descending order of security strength according to this guideline.

### 6 Latency Estimation

The choice of security configuration will also affect other system parameters. It is common, but not always the case, that a more secure algorithm will result in increased latency, reduced throughput, and increased energy consumption. This is for example true, when comparing AES with key lengths of 128, 192, and 256 with each other, since additional encryption rounds are needed when the key length increases. The performance parameter focused on in this report is latency. Latency consists of two components: network latency and computational latency. The estimation of these components and the resulting total latency is considered below. This section involves the main results of our estimations, while the detailed steps of and the background for our analysis can be found in the appendices.

#### 6.1 Network Latency

Network latency captures the delay caused by the different medium access mechanisms of the interfaces of the network nodes to transmit the packets, and the waiting time of packets in queues. In the application scenario, we stated that the network latency is assumed to be nearly zero at the wired part of the network. The main part of the network latency is due to the behavior of the medium access protocol applied at the wireless link. Consequently the network latency for the considered security configurations can be estimated based on the message complexity of the security configurations, i.e., the number of data frames to be sent, and an estimate of the media access control (MAC) layer latency of the wireless link.

##### 6.1.1 Message Complexity

The message complexity caused by the different security configurations can be calculated by the breakdown of the EAP-TLS communication to the data link layer. In reality, during SA establishment, a number of TLS handshake, change-cipherspec, and alert messages are sent. The messages are grouped into records.
and sent in four phases by the TLS record layer. The TLS record messages are transferred in a lock-step fashion due to the EAP protocol in the sub-layer. The EAP-TLS fragment length is upper-bounded (e.g., the default max EAP-TLS fragment value is 1024 bytes in case of FreeRADIUS EAP-TLS server, which is used in this study).

The radio link layer level frames contain headers from the 802.1X, the logical link control (LLC) and the 802.11 protocols. Figure 6 shows the message complexities for the considered security configurations on the level of 802.11 frames. Note that certificates for DSS signatures or containing public DH parameters are typically 10–15% longer than the ones for RSA signatures. In the calculations of the lengths of the TLS certificate handshake messages, we assume that the certificate hierarchy is two-tier. The total number of frames sent on the channel is therefore 23 in cases of fixed DH ciphersuites (i.e., security configurations 4a, 4b, 5a, and 5b in Table 2), and 25 in the other cases. The total number of bytes sent varies between 5.7 and 6.7 KB.

6.1.2 MAC Layer Latency Estimation

The dependencies between latency and the parameters of the 802.11 MAC distributed coordination function (DCF) protocol has been examined by many researchers [2, 38]. However, the practical applicability of these results for MAC layer latency estimation is limited because the input parameters, such as the number of stations in the network, are often difficult to measure at the client side. Moreover, the dependencies are often expressed in general terms and make many restrictions.

In our work, we use simulation to estimate the MAC layer latency where MAC layer latency is defined as the delay between the packet arrival at the MAC layer and the receipt of the last bit of an ACK frame for the successful transmission of the corresponding data frame, or the drop of the data frame. Thus the MAC layer latency captures the waiting time of the data frame in the network interface queue as well as the MAC service time including one or a limited number of trials to send the data frame.

We analyzed the MAC layer latency with a simulation model written in the OMNeT++ simulator [35]. The experiment data were analyzed graphically using the ROME framework [30]. The graphical analysis showed that the current channel utilization—or vice-versa, the percent of time when the channel is idle—and the current packet length to transmit are good candidates to estimate the mean of MAC layer latency for a data frame. The channel utilization also predicts the expected variance of the MAC layer latency. The channel utilization can be conveniently measured at the client by inspecting the values of the network allocation vector (NAV) [30].

Figure 7 shows the dependency of the mean MAC layer latency as a function of the data frame length and the percentage of time the channel is in idle state. The parameters of the DCF protocol were taken from the 802.11b standard [12],

---

6DCF is the primary medium access mechanism supported by IEEE 802.11 networks.
7In the 802.11 standard, the NAV value is maintained in all stations and signals the virtual carrier busy states.
Figure 6: Message complexity.

Security configuration (total bytes sent)

Number of IEEE 802.11 frames on the radio link grouped by length in byte
the model applied the RTS/CTS access scheme.

![Figure 7: MAC layer latency vs. the proportion of time of the channel in idle state and the current data frame length.](image)

6.1.3 Combining Message Complexity and MAC Layer Latency

We can now calculate an estimation of the network latency for each security configuration. We defined three levels of network load. The channel load is called “moderate” at less than 45% of utilization, “high” for 45–70% and “saturated” above 70%. Above 70% of channel utilization the length of the waiting queues and the number of dropped packets have an increasing trend, and the MAC layer latency of a data frame may jump to the order of $10^{-3} - 10^{-2}$ seconds from the order of $10^{-1} - 10^0$ seconds. The variance of the MAC layer latency also increases significantly. In practice, this load could happen only if many stations (more than 10) would flood the network with small packets. In case of TCP based protocols this could not happen because TCP connections would regulate down their data rate.

For the estimation of the network latency, we divided our MAC layer latency results into sets based on the network load levels and the transmitted data frame length. We defined 20 equivalent frame length intervals from 0 to 1536 bytes. For each group a statistical summary of the MAC layer latency was calculated including mean, max, and min values as well as 1st, 2nd, and 3rd quantiles.

Table 3 presents the results for the estimated network latencies for the considered security configurations. The calculations were based on the mean values of the MAC layer latency sets. The order of magnitude of the estimated means of the network latency was almost the same ($10^{-2}$ seconds) except for saturated networks where it is in the order of $10^0$ seconds. There are only small differences between the security configurations because their message complexities are almost
Table 3: Estimate of network latencies [sec].

<table>
<thead>
<tr>
<th>ID</th>
<th>Moderate</th>
<th>High</th>
<th>Saturated</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a</td>
<td>1.71E-02</td>
<td>3.92E-02</td>
<td>1.85E+00</td>
</tr>
<tr>
<td>1b</td>
<td>1.71E-02</td>
<td>3.93E-02</td>
<td>1.86E+00</td>
</tr>
<tr>
<td>2a</td>
<td>1.67E-02</td>
<td>3.89E-02</td>
<td>1.87E+00</td>
</tr>
<tr>
<td>2b</td>
<td>1.67E-02</td>
<td>3.90E-02</td>
<td>1.87E+00</td>
</tr>
<tr>
<td>3a</td>
<td>1.68E-02</td>
<td>3.92E-02</td>
<td>1.88E+00</td>
</tr>
<tr>
<td>3b</td>
<td>1.65E-02</td>
<td>3.88E-02</td>
<td>1.88E+00</td>
</tr>
<tr>
<td>4a</td>
<td>1.55E-02</td>
<td>3.61E-02</td>
<td>1.71E+00</td>
</tr>
<tr>
<td>4b</td>
<td>1.56E-02</td>
<td>3.60E-02</td>
<td>1.71E+00</td>
</tr>
<tr>
<td>5a</td>
<td>1.55E-02</td>
<td>3.58E-02</td>
<td>1.73E+00</td>
</tr>
<tr>
<td>5b</td>
<td>1.55E-02</td>
<td>3.58E-02</td>
<td>1.72E+00</td>
</tr>
</tbody>
</table>

the same. The fixed DH ciphersuites are faster by 7–8% than the others.

6.2 Computational Latency

Computational latency is here defined as the delay caused by the processing of cryptographic algorithms in the nodes of the network. The delay caused by other computational processes is assumed constant and relatively small. The estimation of computational latency is a nontrivial problem [31]. However, if we know the mean computational latency of a specific cryptographic algorithm on a given data size, then we can calculate the total computational latency of each security configuration by counting the number of data blocks on which a given type of cryptographic primitive is applied. We must, however, take into consideration if some cryptographic processes run in parallel.

6.2.1 Computational Complexity

The various cryptographic algorithms used in EAP-TLS are built up from MD5 and SHA-1 hash functions, and RSA and DSS signature generation and verification algorithms. Figure 8 and Figure 9 present the number of executions of the applied cryptographic algorithms for each considered security configuration. The hashing algorithms were supposed to run on 64-byte blocks. The calculations of DH public parameters and secrets are not considered in our calculations because the DH public parameters can be pre-generated. The only real-time computation associated with them is the computation of the pre-master secret based on these values.

6.2.2 Latency of Cryptographic Algorithms

Next, we need to estimate the computational latency of the cryptographic algorithms for the given block sizes used in the considered security configurations. The computational latency mainly depends on the hardware characteristics of the
device such as CPU frequency and amount of RAM memory. Moreover, the implementation of the algorithms may also influence their speed. For the estimation we measured the computational latency of the applied cryptographic primitives on “low-end” and “high-end” devices. The low-end device was a 600 MHz Genuine Intel Celeron processor computer with 64 MB RAM, the high-end device was a Pentium (R) 4 CPU, 3GHz, 512 MB RAM computer. The speed test application in OpenSSL cryptography toolkit [24] was used for the measurements.

Table 4 summarizes the computational latency measurement results of the cryptographic primitives. The measurements were executed 30 times; and the mean values and 95% confidence intervals were calculated. Note that the confidence intervals were all within 1% of their respective mean values.

6.2.3 Combining Computational Complexity and Latency of Cryptographic Algorithms

By the combination of the computational complexities of the security configurations and the latency of the cryptographic primitives, we can estimate the order of magnitude of the computational latency during the SA establishment. Table 5 summarizes the estimated computational latencies.

6.3 Combining Network and Computational Latency

The estimated total latency is the sum of the estimated network and computational latencies. Table 6 and Table 7 show the total estimated latencies for each security configuration.
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Low-end device</th>
<th>High-end device</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHA-1 (64)</td>
<td>6.31E-08</td>
<td>6.31E-08</td>
</tr>
<tr>
<td>MD5 (64)</td>
<td>7.81E-08</td>
<td>7.82E-08</td>
</tr>
<tr>
<td>RSA sign</td>
<td>1.15E-02</td>
<td>1.15E-02</td>
</tr>
<tr>
<td>RSA verify</td>
<td>6.08E-04</td>
<td>6.07E-04</td>
</tr>
<tr>
<td>DSA sign</td>
<td>6.68E-03</td>
<td>6.67E-03</td>
</tr>
<tr>
<td>DSA verify</td>
<td>1.12E-02</td>
<td>1.12E-02</td>
</tr>
<tr>
<td>RSA encrypt</td>
<td>6.38E-08</td>
<td>6.38E-08</td>
</tr>
<tr>
<td>RSA decrypt</td>
<td>6.06E-04</td>
<td>6.07E-04</td>
</tr>
</tbody>
</table>

Table 4: Computational latencies of the cryptographic primitives [sec].
The relative latency costs of the respective security configurations in a given scenario are easily established by examining the relevant column in the table. Considering only moderate and high network loads, the highest total latency is achieved by the TLS_DHE_DSS_WITH_*_* ciphersuites with a DSS signed certificate at the client side (1a). In all environmental situations, the lowest latency is achieved by the TLS_DH_RSA_WITH_*_* with RSA signed client certificate security configuration (5b). In general, the fixed DH security configurations (4a, 4b, 5a, and 5b) show good latency performance.

We can see in the table that the choice of DSS or RSA certificates at the server and the client side significantly influences the expected latency of the session establishment. When using DSS signed certificates, signing and verifying are more computationally demanding than when using RSA. The certificate lengths are also higher in case of DSS.

By looking at the rows in Table 6 and 7, the dependency from the network load can be seen for low-end and high-end cases. In case of a highly loaded network we could expect a 50–100% increase, compared to moderate load, for the expected mean of the total latency. The proportion of the computational latency in the total latency is much more important in case of low-end devices and at moderate network loads. Therefore, the relative differences between the security configurations are also larger in these cases.
Table 5: Estimate of computational latencies [sec].

<table>
<thead>
<tr>
<th>ID</th>
<th>Low-end device</th>
<th>High-end device</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a</td>
<td>3.99E-02</td>
<td>1.62E-02</td>
</tr>
<tr>
<td>1b</td>
<td>2.71E-02</td>
<td>1.06E-02</td>
</tr>
<tr>
<td>2a</td>
<td>2.74E-02</td>
<td>1.11E-02</td>
</tr>
<tr>
<td>2b</td>
<td>1.46E-02</td>
<td>5.54E-03</td>
</tr>
<tr>
<td>3a</td>
<td>3.96E-02</td>
<td>1.57E-02</td>
</tr>
<tr>
<td>3b</td>
<td>2.68E-02</td>
<td>1.02E-02</td>
</tr>
<tr>
<td>4a</td>
<td>2.74E-02</td>
<td>1.11E-02</td>
</tr>
<tr>
<td>4b</td>
<td>1.49E-02</td>
<td>6.02E-03</td>
</tr>
<tr>
<td>5a</td>
<td>1.49E-02</td>
<td>6.02E-03</td>
</tr>
<tr>
<td>5b</td>
<td>2.48E-03</td>
<td>9.20E-04</td>
</tr>
</tbody>
</table>

7 Combining Security and Performance Requirements

To be able to select the most appropriate security configuration in a given situation both environmental characteristics and user preferences are needed. The collection of environmental characteristics in relation to latency was discussed in the previous section. User preferences are entered by the user at run-time. A simple graphical user interface (GUI) that provides five different choices is depicted in Figure 10. By moving the slider in the GUI, a user selects the desired trade-off between security and performance.

Based on environmental characteristics ($E$), user preferences ($U$), and a set of security configurations ($S$), a TS service is defined through a mapping function\footnote{We refer to this mapping function as the tunable security (TS) function.} that selects the security configuration to use in a given situation: $E \times U \rightarrow S$. In this case, the set $S$ contains the security configurations identified in Section 5. $E$ is characterized by type of equipment (low-end or high-end device) and network load (moderate, high, and saturated). $U$ has five options corresponding to what parameter (security or performance) was indicated as more important by the user: security only (SO), security most (SM), balanced (BA), performance most (PM), and performance only (PO). Table 8 illustrates a suitable mapping from $E$ and $U$ to $S$ in this particular case.

For user preferences of SO and PO the mapping function selects the most secure security configuration and the security configuration with the lowest latency in each configuration, respectively. When SM is selected the three most secure security configurations are considered (1a–2a) and the one with best latency performance among them selected. The mapping for BA considers the six most secure security configurations (1a–3a) and selects the security configuration...
Table 6: Total estimated latencies [sec] and relative performance cost of security configurations for low-end devices.

<table>
<thead>
<tr>
<th>S</th>
<th>Low-end device</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Moderate</td>
<td>High</td>
<td>Saturated</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1a</td>
<td>5.69E-02</td>
<td>100,00%</td>
<td>7.91E-02</td>
<td>100,00%</td>
<td>1.89E+00</td>
<td>98,77%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1b</td>
<td>4.41E-02</td>
<td>77,54%</td>
<td>6.64E-02</td>
<td>83,94%</td>
<td>1.88E+00</td>
<td>98,19%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2a</td>
<td>4.41E-02</td>
<td>77,53%</td>
<td>6.63E-02</td>
<td>83,88%</td>
<td>1.89E+00</td>
<td>98,81%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2b</td>
<td>3.14E-02</td>
<td>55,07%</td>
<td>5.36E-02</td>
<td>67,82%</td>
<td>1.88E+00</td>
<td>98,24%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3a</td>
<td>5.63E-02</td>
<td>98,95%</td>
<td>7.87E-02</td>
<td>99,58%</td>
<td>1.92E+00</td>
<td>100,00%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3b</td>
<td>4.32E-02</td>
<td>75,96%</td>
<td>6.56E-02</td>
<td>82,95%</td>
<td>1.90E+00</td>
<td>99,23%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4a</td>
<td>4.29E-02</td>
<td>75,30%</td>
<td>6.34E-02</td>
<td>80,24%</td>
<td>1.74E+00</td>
<td>90,66%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4b</td>
<td>3.05E-02</td>
<td>53,55%</td>
<td>5.10E-02</td>
<td>64,46%</td>
<td>1.72E+00</td>
<td>89,75%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5a</td>
<td>3.04E-02</td>
<td>53,38%</td>
<td>5.07E-02</td>
<td>64,19%</td>
<td>1.74E+00</td>
<td>90,77%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5b</td>
<td>1.80E-02</td>
<td>31,63%</td>
<td>3.83E-02</td>
<td>48,40%</td>
<td>1.72E+00</td>
<td>89,86%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7: Total estimated latencies [sec] and relative performance cost of security configurations for high-end devices.

<p>| S   | High-end device |          |          |          |          |          |          |          |          |          |          |          |          |          |          |          |
|-----|-----------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
|     |                 | Moderate | High     | Saturated|          |          |          |          |          |          |          |          |          |          |          |          |
| 1a  | 3.33E-02        | 100,00%  | 5.54E-02 | 100,00%  | 1.87E+00 | 98,76%   |          |          |          |          |          |          |          |          |          |
| 1b  | 2.77E-02        | 83,31%   | 4.99E-02 | 90,14%   | 1.87E+00 | 98,56%   |          |          |          |          |          |          |          |          |          |
| 2a  | 2.78E-02        | 83,84%   | 5.00E-02 | 90,27%   | 1.88E+00 | 99,19%   |          |          |          |          |          |          |          |          |          |
| 2b  | 2.23E-02        | 66,95%   | 4.45E-02 | 80,41%   | 1.87E+00 | 99,00%   |          |          |          |          |          |          |          |          |          |
| 3a  | 3.25E-02        | 97,72%   | 5.40E-02 | 99,11%   | 1.89E+00 | 100,00%  |          |          |          |          |          |          |          |          |          |
| 3b  | 2.66E-02        | 80,12%   | 4.90E-02 | 88,43%   | 1.89E+00 | 99,60%   |          |          |          |          |          |          |          |          |          |
| 4a  | 2.66E-02        | 80,02%   | 4.72E-02 | 85,19%   | 1.72E+00 | 90,94%   |          |          |          |          |          |          |          |          |          |
| 4b  | 2.16E-02        | 64,90%   | 4.20E-02 | 75,93%   | 1.71E+00 | 90,41%   |          |          |          |          |          |          |          |          |          |
| 5a  | 2.15E-02        | 64,61%   | 4.18E-02 | 75,55%   | 1.73E+00 | 91,44%   |          |          |          |          |          |          |          |          |          |
| 5b  | 1.65E-02        | 49,48%   | 3.67E-02 | 66,29%   | 1.72E+00 | 90,91%   |          |          |          |          |          |          |          |          |          |</p>
<table>
<thead>
<tr>
<th>User preferences (U)</th>
<th>Performance only</th>
<th>Performance most</th>
<th>Balanced</th>
<th>Security most</th>
<th>Security only</th>
<th>Low-end device</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Moderate</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>High</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Saturated</td>
</tr>
<tr>
<td>Performance only</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Moderate</td>
</tr>
<tr>
<td>Performance most</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>High</td>
</tr>
<tr>
<td>Balanced</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Saturated</td>
</tr>
<tr>
<td>Security most</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Moderate</td>
</tr>
<tr>
<td>Security only</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>High</td>
</tr>
<tr>
<td>Low-end device</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Saturated</td>
</tr>
</tbody>
</table>
with lowest latency for a given configuration. For PM the mapping considers the three security configurations with lowest latency and selects the most secure amongst them. In case of equal performance for two security configurations the most secure one is selected. Note that the mapping rules used in our example are quite simplistic and that the mapping in Table 8 only represents one feasible TS function.

8 Summary and Future Work

In this report we have illustrated how a TS service can be defined for the establishment of a security association in 802.11i. The service selects an appropriate security configuration considering the trade-off between security and latency. In order to perform an intelligent trade-off the latency of the considered security configurations must be estimated and the ordering of the security configurations in terms of security established. The latency estimation is influenced by environmental parameters such as network load and the type of device used. Finally, the preferences of the user are also considered. The presented TS function is by no means the only feasible mapping, but it illustrates how the TS service can be constructed. The design of TS services is essential to be able to treat security as a QoS dimension in future wireless networks.

In the future, a general TS service model should be established which could serve as a tool to design TS services, and analyze existing security services from the point of view of the possible trade-offs between security and performance. In
a general model the tuner entity (e.g., the user) may prefer to define not only the network latency constraints, but also other performance related parameters (e.g., energy consumption, jitter, and usability) or the combination of them. The security level could also be defined in many ways, not only by the security level of the SA establishment. For example, confidence level of the session, integrity level, authenticity, or more abstract parameters as resistance to denial of service (DoS) or other attacks could also be the preferred security constraints defined by the tuner. More research work should be done to estimate network latency of different random access protocols, such as the DCF protocol used in this work. The DCF protocol has similar behavior to positive feedback controlled systems, where a phase shifting, i.e., a sudden degradation of the effective throughput of the channel can be caused by the increase of the amount of repeated and collided frames. This behavior should be analyzed more precisely and the estimation method could be extended to other random access protocols.

References


# Abbreviations

Abbreviations used in this report are summarized in alphabetical order below.

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3DES</td>
<td>Triple DES</td>
</tr>
<tr>
<td>AAA</td>
<td>Authentication, authorization, and accounting</td>
</tr>
<tr>
<td>ACK</td>
<td>Acknowledgement</td>
</tr>
<tr>
<td>AES</td>
<td>Advanced encryption standard</td>
</tr>
<tr>
<td>AP</td>
<td>Access point</td>
</tr>
<tr>
<td>AS</td>
<td>Authentication server</td>
</tr>
<tr>
<td>BUTE</td>
<td>Budapest University of Technology and Economics</td>
</tr>
<tr>
<td>CATV</td>
<td>Cable antenna television</td>
</tr>
<tr>
<td>CBC</td>
<td>Cipher block chaining</td>
</tr>
<tr>
<td>CCMP</td>
<td>Counter-mode/CBC MAC protocol</td>
</tr>
<tr>
<td>CPU</td>
<td>Central processing unit</td>
</tr>
<tr>
<td>CTS</td>
<td>Clear to send</td>
</tr>
<tr>
<td>CW</td>
<td>Contention window</td>
</tr>
<tr>
<td>DCF</td>
<td>Distributed coordination function</td>
</tr>
<tr>
<td>DES</td>
<td>Data encryption standard</td>
</tr>
<tr>
<td>DH</td>
<td>Diffie-Hellman</td>
</tr>
<tr>
<td>DHE</td>
<td>Ephemeral DH</td>
</tr>
<tr>
<td>DoS</td>
<td>Denial of service</td>
</tr>
<tr>
<td>DS</td>
<td>Data server</td>
</tr>
<tr>
<td>DSS</td>
<td>Digital signature standard</td>
</tr>
<tr>
<td>EAP</td>
<td>Extensible authentication protocol</td>
</tr>
<tr>
<td>EDE</td>
<td>Encryption, decryption, and encryption</td>
</tr>
<tr>
<td>EIFS</td>
<td>Extended interframe space</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical user interface</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronics Engineers</td>
</tr>
<tr>
<td>IP</td>
<td>Internet protocol</td>
</tr>
<tr>
<td>IPSec</td>
<td>IP security</td>
</tr>
<tr>
<td>KaU</td>
<td>Karlstad University</td>
</tr>
<tr>
<td>KCK</td>
<td>Key confirmation key</td>
</tr>
<tr>
<td>KEK</td>
<td>Key encryption key</td>
</tr>
<tr>
<td>LAN</td>
<td>Local area network</td>
</tr>
<tr>
<td>LLC</td>
<td>Logical link control</td>
</tr>
<tr>
<td>MAC</td>
<td>Message authentication code or media access control</td>
</tr>
<tr>
<td>MD5</td>
<td>Message digest algorithm number 5</td>
</tr>
<tr>
<td>MK</td>
<td>Master key</td>
</tr>
<tr>
<td>NAS</td>
<td>Network access server</td>
</tr>
<tr>
<td>NAV</td>
<td>Network allocation vector</td>
</tr>
<tr>
<td>NEWCOM</td>
<td>Network of Excellence in Wireless Communication</td>
</tr>
<tr>
<td>Acronym</td>
<td>Abbreviation</td>
</tr>
<tr>
<td>---------</td>
<td>--------------</td>
</tr>
<tr>
<td>NIST</td>
<td>National Institute of Standards and Technology</td>
</tr>
<tr>
<td>OMNeT++</td>
<td>Objective modular network testbed in C++</td>
</tr>
<tr>
<td>PMK</td>
<td>Pairwise master key</td>
</tr>
<tr>
<td>PTK</td>
<td>Pairwise transient key</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of service</td>
</tr>
<tr>
<td>RADIUS</td>
<td>Remote authentication dial in user service</td>
</tr>
<tr>
<td>RAM</td>
<td>Random access memory</td>
</tr>
<tr>
<td>ROME</td>
<td>Root based object oriented midas extension</td>
</tr>
<tr>
<td>RSA</td>
<td>Rivest, Shamir and Adleman</td>
</tr>
<tr>
<td>RTS</td>
<td>Request to send</td>
</tr>
<tr>
<td>SA</td>
<td>Security association</td>
</tr>
<tr>
<td>SHA</td>
<td>Secure hash algorithm</td>
</tr>
<tr>
<td>SHA-1</td>
<td>SHA version 1</td>
</tr>
<tr>
<td>SIFS</td>
<td>Short interframe space</td>
</tr>
<tr>
<td>STA</td>
<td>Station</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission control protocol</td>
</tr>
<tr>
<td>TK</td>
<td>Temporal key</td>
</tr>
<tr>
<td>TKIP</td>
<td>Temporal key integrity protocol</td>
</tr>
<tr>
<td>TLS</td>
<td>Transport layer security</td>
</tr>
<tr>
<td>TTLS</td>
<td>Tunneled TLS</td>
</tr>
<tr>
<td>TS</td>
<td>Tunable security</td>
</tr>
<tr>
<td>VPN</td>
<td>Virtual private network</td>
</tr>
<tr>
<td>WEP</td>
<td>Wireless equivalent privacy</td>
</tr>
<tr>
<td>WLAN</td>
<td>Wireless LAN</td>
</tr>
</tbody>
</table>
B Latency Estimations

This appendix contains the detailed description of the latency estimation used in this work. In order to estimate the costs of the choices for different security configurations in terms of latency, the computational and the network latency of each SA establishment configuration have been analyzed.

The functioning of the different security configurations is analysed based on the descriptions of the standards of the TLS working group. For verification one security configuration is also analyzed through an Ethereal trace, see Appendix C.

Our approach is to factorize the computational and network latencies and to analyze the dependencies of the values of the components based on the circumstance factors. In case of network latency estimation, the components are the times for successful transmission of given sized data frames, through the 802.11b MAC layer. The circumstance factor that influences the latency is the network load. In case of computational latency, the components are the processing times of the different cryptographic algorithms — such as hashing with given block size, signing and signature verification algorithms — on different type of devices meant as circumstance factor. To get the total estimated latencies, the sum of the values of the components multiplied by the number of their occurrences are calculated.

To define the number of occurrences of the components, a deep analysis of the TLS standards and the Ethereal trace (see Appendix C) is necessary. In order to reduce the space of the set of circumstances, we defined six clusters: three classes of network load levels multiplied by two classes of device types. Note, that our methodology can easily be adapted to other security frameworks or protocols, which use similar components and circumstance factors.

B.1 Network Latency Calculations

B.1.1 Number of Occurrences

The calculation of the number of occurrences of the 802.11 data frames with different lengths is done as follows. The Ethereal trace in Appendix C and the TLS standards have been analyzed. Based on them, the expected or real TLS message lengths used in the different SA establishments have been calculated. The calculation is made on several assumptions:

- The certificate hierarchy is two-tier. When an entity sends its signing or fixed DH certificate, it is always followed by the signing certificate of the root CA.

- If an entity has a DSS signing certificate, the certificate of the root CA is also a DSS signing certificate. This is also true for RSA signing certificates.

- The lengths of the certificates were based on real certificates, generated with the OpenSSL package. Table 9 contains the certificate lengths used in the calculations.

Table 10, Table 11 and Table 12 contain the lengths of the TLS messages in bytes for each considered security configuration. The values include also the TLS
Table 9: Certificate lengths.

<table>
<thead>
<tr>
<th>Certificate type</th>
<th>Length [bytes]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed DH certificate with DSS signature</td>
<td>871</td>
</tr>
<tr>
<td>Fixed DH certificate with RSA signature</td>
<td>871</td>
</tr>
<tr>
<td>DSS signing certificate of the root CA</td>
<td>1086</td>
</tr>
<tr>
<td>RSA signing certificate of the root CA</td>
<td>895</td>
</tr>
<tr>
<td>DSS signing certificate of the server</td>
<td>856</td>
</tr>
<tr>
<td>RSA signing certificate of the server</td>
<td>665</td>
</tr>
<tr>
<td>DSS signing certificate of the client</td>
<td>871</td>
</tr>
<tr>
<td>RSA signing certificate of the client</td>
<td>680</td>
</tr>
</tbody>
</table>

record layer headers (5 bytes). In case of TLS handshake messages, a 4 bytes long header defines the type and length of the payload.

The next step is the projection of the TLS control message exchange to the MAC layer level communication. The number of occurrences of the 802.11 data frames with different lengths are calculated. Section 6.1.1 describes the behavior of the TLS record layer and the EAP protocol. Two important parameters of the calculation should be mentioned here. The maximum EAP-TLS fragment size is by default set to 1024 bytes in the FreeRADIUS EAP-TLS server, which is used in our study. This determines the fragmentation of the TLS records in the EAP request frames. The EAP fragment size of the client was not assigned by us. Instead, our Windows client, which sent EAP responses with 1486 bytes long EAP-TLS fragments at the longest, set this value.

Table 13 summarizes the 802.11 data frame lengths of each security configuration. The numbers in the message type field refer to the TLS message types, and their meanings can be found, e.g., based on Table 10.

The message complexity diagram (Figure 6) presented Subsection 6.1.1 summarizes the relations between the 802.11 MAC layer level communication and each considered security configuration.
Table 10: TLS message lengths for security configurations based on $\text{TLS}_\text{DHE}_{\text{*}*}$.

<table>
<thead>
<tr>
<th>No.</th>
<th>TLS message</th>
<th>1a</th>
<th>1b</th>
<th>2a</th>
<th>2b</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>client hello</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>Only one cipher suite name is in the cipher suite list</td>
</tr>
<tr>
<td>2</td>
<td>server hello</td>
<td>79</td>
<td>79</td>
<td>79</td>
<td>79</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>certificate</td>
<td>1942</td>
<td>1942</td>
<td>1578</td>
<td>1578</td>
<td>1a/b: 2x DSS Sign. Cert. 2a/b: 2x RSA Sign. Cert.</td>
</tr>
<tr>
<td>4</td>
<td>server key exchange</td>
<td>274</td>
<td>274</td>
<td>274</td>
<td>274</td>
<td>Diffie-Hellman parameters of the server</td>
</tr>
<tr>
<td>5</td>
<td>certificate request</td>
<td>158</td>
<td>158</td>
<td>158</td>
<td>158</td>
<td>Four types of certificates are accepted by the server:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Fixed-DH with RSA or DSS signature, DSS or RSA Sign. Cert.</td>
</tr>
<tr>
<td>6</td>
<td>done</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>This message is under the same TLS Record header as the certificate request message. (5)</td>
</tr>
<tr>
<td>8</td>
<td>client key exchange</td>
<td>75</td>
<td>75</td>
<td>75</td>
<td>75</td>
<td>Diffie-Hellman parameters of the client</td>
</tr>
<tr>
<td>9</td>
<td>certificate verify</td>
<td>29</td>
<td>45</td>
<td>29</td>
<td>45</td>
<td>a: DSS signature, b: RSA signature of the client</td>
</tr>
<tr>
<td>10</td>
<td>change cipher spec</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>finished</td>
<td>53</td>
<td>53</td>
<td>53</td>
<td>53</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>change cipher spec</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>finished</td>
<td>53</td>
<td>53</td>
<td>53</td>
<td>53</td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>TLS message</td>
<td>Length (B)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>-------------</td>
<td>------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>client hello</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>server hello</td>
<td>79</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>certificate request</td>
<td>1578</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>server key exchange</td>
<td>301</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>certificate verify</td>
<td>1970, 1588</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>client key exchange</td>
<td>48</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>certificate verify</td>
<td>29, 45</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>change cipher spec</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>finished</td>
<td>53</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>change cipher spec</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Notes:
- RSA RSA
- DSS, RSA
- DSS
- RSA

Table 11: TLS message lengths for security configurations based on TLS RSA.*
Table 12: TLS message lengths for security configurations based on TLS-DH*.*.

<table>
<thead>
<tr>
<th>No.</th>
<th>TLS message</th>
<th>4a</th>
<th>4b</th>
<th>5a</th>
<th>5b</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>client hello</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>Only one cipher suite name is in the cipher suite list.</td>
</tr>
<tr>
<td>2</td>
<td>server hello</td>
<td>79</td>
<td>79</td>
<td>79</td>
<td>79</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>certificate</td>
<td>1957</td>
<td>1957</td>
<td>1766</td>
<td>1766</td>
<td>Fixed-DH Cert. signed with DSS (4) or RSA (5) + DSS (4) or RSA (5) Sign. Cert. of the root CA.</td>
</tr>
<tr>
<td>4</td>
<td>server key exchange</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>not sent</td>
</tr>
<tr>
<td>5</td>
<td>certificate request</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>not sent</td>
</tr>
<tr>
<td>6</td>
<td>done</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>TLS Record header included</td>
</tr>
<tr>
<td>7</td>
<td>certificate</td>
<td>1957</td>
<td>1766</td>
<td>1957</td>
<td>1766</td>
<td>Fixed-DH Cert. signed with DSS (a) or RSA (b) + DSS (a) or RSA (a) Sign. Cert. of the root CA.</td>
</tr>
<tr>
<td>8</td>
<td>client key exchange</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>empty message, only the header</td>
</tr>
<tr>
<td>9</td>
<td>certificate verify</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>not sent</td>
</tr>
<tr>
<td>10</td>
<td>change cipher spec</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>finished</td>
<td>53</td>
<td>53</td>
<td>53</td>
<td>53</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>change cipher spec</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>finished</td>
<td>53</td>
<td>53</td>
<td>53</td>
<td>53</td>
<td></td>
</tr>
</tbody>
</table>
Table 13: 802.11 data frame lengths in the considered security configurations.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Message Type</th>
<th>1a</th>
<th>1b</th>
<th>2a</th>
<th>2b</th>
<th>3a</th>
<th>3b</th>
<th>4a</th>
<th>4b</th>
<th>5a</th>
<th>5b</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authen. Request</td>
<td>126</td>
<td>126</td>
<td>126</td>
<td>126</td>
<td>126</td>
<td>126</td>
<td>126</td>
<td>126</td>
<td>126</td>
<td>126</td>
<td>126</td>
</tr>
<tr>
<td>Authen. Response</td>
<td>33</td>
<td>33</td>
<td>33</td>
<td>33</td>
<td>33</td>
<td>33</td>
<td>33</td>
<td>33</td>
<td>33</td>
<td>33</td>
<td>33</td>
</tr>
<tr>
<td>Association Request</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Association Response</td>
<td>68</td>
<td>68</td>
<td>68</td>
<td>68</td>
<td>68</td>
<td>68</td>
<td>68</td>
<td>68</td>
<td>68</td>
<td>68</td>
<td>68</td>
</tr>
<tr>
<td>Reassoc. Request</td>
<td>69</td>
<td>69</td>
<td>69</td>
<td>69</td>
<td>69</td>
<td>69</td>
<td>69</td>
<td>69</td>
<td>69</td>
<td>69</td>
<td>69</td>
</tr>
<tr>
<td>Reassoc. Response</td>
<td>96</td>
<td>96</td>
<td>96</td>
<td>96</td>
<td>96</td>
<td>96</td>
<td>96</td>
<td>96</td>
<td>96</td>
<td>96</td>
<td>96</td>
</tr>
<tr>
<td>Pad</td>
<td>78</td>
<td>78</td>
<td>78</td>
<td>78</td>
<td>78</td>
<td>78</td>
<td>78</td>
<td>78</td>
<td>78</td>
<td>78</td>
<td>78</td>
</tr>
<tr>
<td>Value Segment</td>
<td>78</td>
<td>78</td>
<td>78</td>
<td>78</td>
<td>78</td>
<td>78</td>
<td>78</td>
<td>78</td>
<td>78</td>
<td>78</td>
<td>78</td>
</tr>
<tr>
<td>802.1x/EAPOL</td>
<td>78</td>
<td>78</td>
<td>78</td>
<td>78</td>
<td>78</td>
<td>78</td>
<td>78</td>
<td>78</td>
<td>78</td>
<td>78</td>
<td>78</td>
</tr>
<tr>
<td>802.11 Management</td>
<td>455</td>
<td>455</td>
<td>455</td>
<td>455</td>
<td>455</td>
<td>455</td>
<td>455</td>
<td>455</td>
<td>455</td>
<td>455</td>
<td>455</td>
</tr>
<tr>
<td>802.11 Association</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>802.11 Reassociation</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>802.11 Authentication</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>802.11 Request</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>802.11 Response</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>802.11 Type</td>
<td>49</td>
<td>49</td>
<td>49</td>
<td>49</td>
<td>49</td>
<td>49</td>
<td>49</td>
<td>49</td>
<td>49</td>
<td>49</td>
<td>49</td>
</tr>
<tr>
<td>802.11 Authent.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>802.11 Authent. Response</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>802.11 Assoc.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>802.11 Assoc. Response</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>802.11 Reassoc.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>802.11 Reassoc. Response</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>802.11 802.11 Type</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>802.11 802.11 Authent.</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>802.11 802.11 Authent. Response</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>802.11 802.11 Assoc.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>802.11 802.11 Assoc. Response</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>802.11 802.11 Reassoc.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>802.11 802.11 Reassoc. Response</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>802.11 802.11 Type</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>802.11 802.11 Authent.</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>802.11 802.11 Authent. Response</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>802.11 802.11 Assoc.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>802.11 802.11 Assoc. Response</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>802.11 802.11 Reassoc.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>802.11 802.11 Reassoc. Response</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Key1: 802.1x/EAP
Key2: 802.1x/EAPOL
Key3: 802.1x/EAP
Key4: 802.11
B.1.2 MAC Layer Latency Estimation

After getting the number of occurrences of the 802.11 data frames with a given length, the next step consisted of the calculation of their latencies in function of the circumstances. The analysis of the MAC layer latency was done with the aid of an OMNeT++ simulation written by us.

Features of the simulation model. The simulation functions are very similar to the default 802.11 DCF mechanism. It does not implement exactly every parameter written in the 802.11 standard but its behavior is very similar to it. The main implemented features are as follows:

- DCF access protocol with RTS/CTS access scheme in infrastructured network.
- Slotted medium access, i.e., every station has a centrally synchronized clock.
- Exponential backoff procedure after failure. This means that the contention window size is incremented for a station if transmission failure occurs. It is used as described in the standard.
- A simple way to set the radio coverage range for each station.
- Collisions are handled as follows:
  - If all of the stations are in the same radio coverage range, then collision can happen due to the fact that the backoff timer of two or more stations reach 0 at the same time slot.
  - If we define separate radio coverage ranges, stations in different ranges can not detect each other’s transmission. This causes anomalies in their virtual carrier sense states. One or more stations in other ranges may disturb and cause collision in a given range (hidden station problem). E.g., if STA1 is in the radio coverage range set A, STA2 is in B and the AP is in both sets, than STA1 and STA2 can not detect each others transmission and can cause collisions at the proximity of the AP.

Input and output parameters of the simulation model. The input parameters influence the behavior of the simulation model. The main input parameters used in our measurements were the following:

- Number of stations.
- Data unit inter arrival time from the upper layer in the MAC layer of a station. It is given with a distribution function, which is always set to exponential distribution.
- The data frame length transmitted by a station. This is set either to a constant value or is given with a uniform distribution function in our measurements.
The values of the output parameters are influenced by the input parameter settings. The main output parameters are as follows:

- **Scalars**

  **Channel utilization** Percentage of time of the channel in:
  - idle state
  - busy state (in case of successful transmission)
  - collided state (in case of collision of two or more transmissions)

- **Vectors**

  **MAC layer latency** The delay between the income time of the data packet in the MAC layer of the station and the time of the reception of the last bit of the ACK frame for the successful transmission of the data frame, or the delay between the income time of the data unit in the MAC layer and the time of drop of the packet because of the incapability to send it after a number of retries (retryLimit).

  **Block length** This is the delay between the transmission of the first RTS frame and the reception of the last bit of the ACK frame or the drop of the packet. It does not include waiting time in the transmission queue of the station and the interval until the transmission of the first RTS frame.

  **Data length** This is the length of the 802.11 data frame to be sent. Its value depends on the input function giving the distribution of the data frame lengths.

  **Number of (re)sends of RTS frames** It gives the number of trials of RTS transmission until a data unit is sent successfully or it is dropped.

  **Queue length at income time** The MAC layer is modeled as a waiting queue and a server. This is the length of the waiting queue when a data unit arrives in the MAC layer from the upper layer.

  **Queue length at the end of transmission** The length of the waiting queue when the data frame is sent successfully or dropped.

All of the before mentioned output vectors are measured for each transmitted data frame during the runs of the simulation. The simulation model has also other input parameters which make difference among the different versions of the 802.11 protocol. These parameters are set as specified for the 802.11b protocol (see Table 14) [12].

**Simulation experiments.** The aim of our experiments is to generate measurement data and to define which parameters that could serve to predict the MAC layer latency of a 802.11 data frame with a given length. Graphical and statistical analysis is performed on the data generated in the experiments. The experiments consisted of running the simulation with many combinations of the input parameters and gathering the output values into MySQL tables. The MySQL database
Table 14: The 802.11b parameters of the simulation model.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slot time</td>
<td>20 μs</td>
</tr>
<tr>
<td>SIFS</td>
<td>10 μs</td>
</tr>
<tr>
<td>DIFS</td>
<td>40 μs</td>
</tr>
<tr>
<td>EIFS</td>
<td>354 μs</td>
</tr>
<tr>
<td>RTS</td>
<td>160 bit</td>
</tr>
<tr>
<td>CTS</td>
<td>112 bit</td>
</tr>
<tr>
<td>ACK</td>
<td>112 bit</td>
</tr>
<tr>
<td>RetryLimit (short and long)</td>
<td>7</td>
</tr>
<tr>
<td>RTS Threshold</td>
<td>0 bit</td>
</tr>
<tr>
<td>CWmin</td>
<td>31</td>
</tr>
<tr>
<td>CWmax</td>
<td>1024</td>
</tr>
</tbody>
</table>

The engine allowed us to rearrange the measurement data and analyze with graphical and statistical tools the statistical dependencies among the different output and input values. Two experiments have been conducted. In experiment 1, the dependency of the MAC layer latency (and other output parameters) is analyzed on the main input parameters. We ran the simulation with all permutations of the input parameters given in Table 15.

In experiment 2, the dependency of the MAC layer latency (and other output parameters) is analyzed on the channel utilization and the data frame length. The simulation is executed with all permutations of the input parameters given in Table 16.

In both experiments all the stations are within the same radio coverage set. In experiment 1, all the stations behave uniformly. The output parameters of one given station have been analyzed. In experiment 2, we configured the simulation in a way to generate different circumstance cases for the channel load. We measured the output parameters of a selected station which behaved differently from the others. The remaining stations behaved uniformly, causing different background traffic conditions. In both experiments, one run of the simulation ended at the event where either 1000 data frames had been sent by the observed station, or the waiting queue length of the observed station reached 1500 packets.

**Graphical analysis using the ROME framework.** To get some ideas on the dependencies, dependencies between different input and output parameters were analyzed using diagrams. A practical problem was the limitation of the dimensions which can be plotted out at the same time. We decided to use the ROME framework simply because it supports the generation of interactive histograms, which are able to manage more than three dimensions. This means that the de-
Table 15: Input parameters for experiment 1.

<table>
<thead>
<tr>
<th>Input parameter name</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of stations</td>
<td>{2, 4, 6, 8, 10, 12, 14, 16, 18, 20}</td>
</tr>
<tr>
<td>Inter arrival time of data units in the stations [sec]</td>
<td>exponential(mean = {0.05, 0.01, 0.015, 0.02, 0.025, 0.03, 0.035, 0.04, 0.045, 0.05, 0.055, 0.06, 0.065, 0.07, 0.075, 0.08, 0.085, 0.09, 0.095, 0.1, 0.2, 0.3, 0.4, 0.5})</td>
</tr>
<tr>
<td>Data frame lengths [bit]</td>
<td>{512, 1024, 1536, 2048, 2560, 3072, 3584, 4096, 5120, 5632, 6144, 6656, 7168, 7680, 8192, 8704, 9216, 9728, 10240, 10752, 11264, 11776, 12288}</td>
</tr>
</tbody>
</table>

Graphical analysis with R. The graphical analysis of the 802.11b DCF simulation model showed us that the current channel utilization—or vice versa, the percent of time when the channel is idle—and the current data frame length to transmit are good candidates to estimate the mean and variance of the MAC layer latency. Consequently, as a next step we analyzed more precisely the shape of the distribution function of the MAC layer latency with the R statistical analyzer. The results of the graphical analysis with R can be found in Appendix F.

Estimated latencies of the MAC data frames. The components of the network latency calculation are the MAC layer latencies of the data frames with given lengths, sent during the SA establishment. The circumstance factor is the channel load. In order to reduce the number of components, we defined 20 equivalent intervals for the frame lengths between [0,1536] bytes and classified the transmitted frames into these groups. To reduce the number of circumstance factor values three intervals of channel utilization were defined: moderate (0–45%), high (45–70%) and saturated ( > 70%). These classes were defined after the observation of the results of the experiments with simulation (see Appendix E and F).

To get the values of the components in each circumstance case and for each considered security configuration, the MAC layer latency in every cluster — in-
Table 16: Input parameters for experiment 2.

<table>
<thead>
<tr>
<th><strong>Input parameter name</strong></th>
<th><strong>Values</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of stations</td>
<td>{2, 4, 8, 12, 16, 20}</td>
</tr>
</tbody>
</table>
| Inter arrival times in the stations               | exponential(mean = \{0.005, 0.01, \\
| in the background [sec]                           | 0.03, 0.05, 0.1\}) |
| Data frame length of the stations in the background [bit] | \{512, 1024, 2048, 3072, 4096, 5120, 6144, \\
| in the background [sec]                           | 7168, 8192, 9216, 10240, 11264, 12288\} |
| Inter arrival times of the observed station [sec] | exponential([0.01, 0.03, 0.05, 0.1]) |
| Data frame lengths of the observed station [sec]  | uniform(min = 256, max = 12288) |

dexed by the data frame length group and channel load class — was analyzed. For each observed cluster a statistical summary of the MAC layer latency was calculated, including mean(), max(), min() values, as well as 1st, 2nd, and 3rd quartiles. As a result, we got the latency values of the components for the calculation of the total network latency. Table 17, Table 18 and Table 19 summarize these values for each network load cases.

The total network latencies of the considered security configurations are calculated based on the mean values in Table 17, Table 18, and Table 19. Table 3 in Subsection 6.1.3 presents the result of the calculation.

B.2 Computational Latency Calculations

The computational latency is factorized by the different cryptographic algorithms as components. To calculate the total computational latency, the number of executions of each type of algorithm was needed. The circumstance factor in this case was the type of the device, i.e., low-end or high-end.

B.2.1 Number of Occurrences

Table 20 presents the main computational steps related to each TLS message, while Table 21 shows how many bytes of data the different cryptographic algorithms process. This is important to know for the hashing algorithms (MD5 and SHA-1) since they are operating on 64-byte blocks.

Table 22 contains the final summary of the number of executions of each cryptographic algorithm.

In Subsection 6.2.1, Figure 8 and Figure 9 present the computational complexities of the considered methods based on this table.
Table 1: Estimated MAC layer latencies of the data frames at moderate network load.

<table>
<thead>
<tr>
<th>Length (bytes)</th>
<th>FirstQ</th>
<th>SecondQ</th>
<th>MIN</th>
<th>Mean</th>
<th>MAX</th>
<th>10%</th>
<th>90%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.39E-04</td>
<td>3.56E-04</td>
<td>4.24E-04</td>
<td>5.42E-04</td>
<td>6.47E-04</td>
<td>4.08E-03</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1.39E-04</td>
<td>3.56E-04</td>
<td>4.24E-04</td>
<td>5.42E-04</td>
<td>6.47E-04</td>
<td>4.08E-03</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1.67E-04</td>
<td>2.65E-04</td>
<td>4.57E-04</td>
<td>5.37E-04</td>
<td>6.78E-04</td>
<td>4.96E-03</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1.67E-04</td>
<td>2.65E-04</td>
<td>4.57E-04</td>
<td>5.37E-04</td>
<td>6.78E-04</td>
<td>4.96E-03</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1.67E-04</td>
<td>2.65E-04</td>
<td>4.57E-04</td>
<td>5.37E-04</td>
<td>6.78E-04</td>
<td>4.96E-03</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1.67E-04</td>
<td>2.65E-04</td>
<td>4.57E-04</td>
<td>5.37E-04</td>
<td>6.78E-04</td>
<td>4.96E-03</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1.67E-04</td>
<td>2.65E-04</td>
<td>4.57E-04</td>
<td>5.37E-04</td>
<td>6.78E-04</td>
<td>4.96E-03</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1.67E-04</td>
<td>2.65E-04</td>
<td>4.57E-04</td>
<td>5.37E-04</td>
<td>6.78E-04</td>
<td>4.96E-03</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1.67E-04</td>
<td>2.65E-04</td>
<td>4.57E-04</td>
<td>5.37E-04</td>
<td>6.78E-04</td>
<td>4.96E-03</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.67E-04</td>
<td>2.65E-04</td>
<td>4.57E-04</td>
<td>5.37E-04</td>
<td>6.78E-04</td>
<td>4.96E-03</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1.67E-04</td>
<td>2.65E-04</td>
<td>4.57E-04</td>
<td>5.37E-04</td>
<td>6.78E-04</td>
<td>4.96E-03</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>2.41E-04</td>
<td>4.15E-04</td>
<td>6.34E-04</td>
<td>6.84E-04</td>
<td>8.38E-04</td>
<td>6.27E-03</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>2.41E-04</td>
<td>4.15E-04</td>
<td>6.34E-04</td>
<td>6.84E-04</td>
<td>8.38E-04</td>
<td>6.27E-03</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>2.76E-04</td>
<td>5.77E-04</td>
<td>7.29E-04</td>
<td>7.34E-04</td>
<td>8.06E-04</td>
<td>4.12E-03</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>3.60E-04</td>
<td>6.29E-04</td>
<td>7.57E-04</td>
<td>9.67E-04</td>
<td>1.30E-03</td>
<td>7.02E-03</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>4.25E-04</td>
<td>6.48E-04</td>
<td>8.71E-04</td>
<td>8.78E-04</td>
<td>9.05E-04</td>
<td>5.78E-03</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>4.25E-04</td>
<td>6.48E-04</td>
<td>8.71E-04</td>
<td>8.78E-04</td>
<td>9.05E-04</td>
<td>5.78E-03</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>5.07E-04</td>
<td>6.22E-04</td>
<td>6.88E-04</td>
<td>8.09E-04</td>
<td>8.44E-04</td>
<td>6.27E-03</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>5.07E-04</td>
<td>6.22E-04</td>
<td>6.88E-04</td>
<td>8.09E-04</td>
<td>8.44E-04</td>
<td>6.27E-03</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>5.07E-04</td>
<td>6.22E-04</td>
<td>6.88E-04</td>
<td>8.09E-04</td>
<td>8.44E-04</td>
<td>6.27E-03</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>5.07E-04</td>
<td>6.22E-04</td>
<td>6.88E-04</td>
<td>8.09E-04</td>
<td>8.44E-04</td>
<td>6.27E-03</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>5.07E-04</td>
<td>6.22E-04</td>
<td>6.88E-04</td>
<td>8.09E-04</td>
<td>8.44E-04</td>
<td>6.27E-03</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>5.07E-04</td>
<td>6.22E-04</td>
<td>6.88E-04</td>
<td>8.09E-04</td>
<td>8.44E-04</td>
<td>6.27E-03</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>5.07E-04</td>
<td>6.22E-04</td>
<td>6.88E-04</td>
<td>8.09E-04</td>
<td>8.44E-04</td>
<td>6.27E-03</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>5.07E-04</td>
<td>6.22E-04</td>
<td>6.88E-04</td>
<td>8.09E-04</td>
<td>8.44E-04</td>
<td>6.27E-03</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>5.07E-04</td>
<td>6.22E-04</td>
<td>6.88E-04</td>
<td>8.09E-04</td>
<td>8.44E-04</td>
<td>6.27E-03</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>5.07E-04</td>
<td>6.22E-04</td>
<td>6.88E-04</td>
<td>8.09E-04</td>
<td>8.44E-04</td>
<td>6.27E-03</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>5.07E-04</td>
<td>6.22E-04</td>
<td>6.88E-04</td>
<td>8.09E-04</td>
<td>8.44E-04</td>
<td>6.27E-03</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>5.07E-04</td>
<td>6.22E-04</td>
<td>6.88E-04</td>
<td>8.09E-04</td>
<td>8.44E-04</td>
<td>6.27E-03</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>5.07E-04</td>
<td>6.22E-04</td>
<td>6.88E-04</td>
<td>8.09E-04</td>
<td>8.44E-04</td>
<td>6.27E-03</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Estimated MAC layer latencies of the data frames at moderate network load.
Table 18: Estimated MAC layer latencies of the data frames at high network load.

<table>
<thead>
<tr>
<th>Length (bytes)</th>
<th>High load (45% – 70%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MIN</td>
</tr>
<tr>
<td>1</td>
<td>1.18E-03</td>
</tr>
<tr>
<td>2</td>
<td>1.18E-03</td>
</tr>
<tr>
<td>3</td>
<td>1.18E-03</td>
</tr>
<tr>
<td>4</td>
<td>1.18E-03</td>
</tr>
<tr>
<td>5</td>
<td>1.18E-03</td>
</tr>
<tr>
<td>6</td>
<td>1.18E-03</td>
</tr>
<tr>
<td>7</td>
<td>1.18E-03</td>
</tr>
<tr>
<td>8</td>
<td>1.18E-03</td>
</tr>
<tr>
<td>9</td>
<td>1.18E-03</td>
</tr>
<tr>
<td>10</td>
<td>1.18E-03</td>
</tr>
<tr>
<td>11</td>
<td>1.18E-03</td>
</tr>
<tr>
<td>12</td>
<td>1.18E-03</td>
</tr>
<tr>
<td>13</td>
<td>1.18E-03</td>
</tr>
<tr>
<td>14</td>
<td>1.18E-03</td>
</tr>
<tr>
<td>15</td>
<td>1.18E-03</td>
</tr>
<tr>
<td>16</td>
<td>1.18E-03</td>
</tr>
<tr>
<td>17</td>
<td>1.18E-03</td>
</tr>
<tr>
<td>18</td>
<td>1.18E-03</td>
</tr>
<tr>
<td>19</td>
<td>1.18E-03</td>
</tr>
<tr>
<td>20</td>
<td>1.18E-03</td>
</tr>
<tr>
<td>21</td>
<td>1.18E-03</td>
</tr>
<tr>
<td>22</td>
<td>1.18E-03</td>
</tr>
<tr>
<td>23</td>
<td>1.18E-03</td>
</tr>
<tr>
<td>24</td>
<td>1.18E-03</td>
</tr>
</tbody>
</table>
Table 19: Estimated MAC layer latencies of the data frames at saturated network load.

<table>
<thead>
<tr>
<th>Length (bytes)</th>
<th>Saturated (70% - 100%)</th>
<th>MIN</th>
<th>P90</th>
<th>Mean</th>
<th>P99</th>
<th>MAX</th>
<th>Length</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32</td>
<td>6</td>
<td>929</td>
<td>999</td>
<td>122</td>
<td>131</td>
<td>36</td>
<td>116</td>
</tr>
<tr>
<td>2</td>
<td>64</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>3</td>
<td>96</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>4</td>
<td>128</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>5</td>
<td>160</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>6</td>
<td>192</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>7</td>
<td>224</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>8</td>
<td>256</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>9</td>
<td>288</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>10</td>
<td>320</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>11</td>
<td>352</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>12</td>
<td>384</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>13</td>
<td>416</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>14</td>
<td>448</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>15</td>
<td>480</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>16</td>
<td>512</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>17</td>
<td>544</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>18</td>
<td>576</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>19</td>
<td>608</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>20</td>
<td>640</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>21</td>
<td>672</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>22</td>
<td>704</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>23</td>
<td>736</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>24</td>
<td>768</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>25</td>
<td>800</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>26</td>
<td>832</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>27</td>
<td>864</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>28</td>
<td>896</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>29</td>
<td>928</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>30</td>
<td>960</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>31</td>
<td>992</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
<tr>
<td>32</td>
<td>1024</td>
<td>6</td>
<td>108</td>
<td>214</td>
<td>116</td>
<td>126</td>
<td>36</td>
<td>128</td>
</tr>
</tbody>
</table>

Table 19: Estimated MAC layer latencies of the data frames at saturated network load.
Table 20: The main computational steps related to each TLS message.

<table>
<thead>
<tr>
<th>TLS msg.</th>
<th>1a</th>
<th>1b</th>
<th>2a</th>
<th>2b</th>
<th>3a</th>
<th>3b</th>
<th>4a</th>
<th>4b</th>
<th>5a</th>
<th>5b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>2 DSS-1</td>
<td>2 DSS-1</td>
<td>2 RSA-1</td>
<td>2 RSA-1</td>
<td>2 RSA-1</td>
<td>2 RSA-1</td>
<td>2 DSS-1</td>
<td>2 DSS-1</td>
<td>2 RSA-1</td>
<td>2 RSA-1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2 RSA-1</td>
<td>2 RSA-1</td>
<td>2 RSA-1</td>
<td>2 RSA-1</td>
<td>2 RSA-1</td>
<td>2 RSA-1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>2 DSS-1</td>
<td>2 RSA-1</td>
<td>2 DSS-1</td>
<td>2 RSA-1</td>
<td>2 RSA-1</td>
<td>2 RSA-1</td>
<td>2 DSS-1</td>
<td>2 RSA-1</td>
<td>2 RSA-1</td>
<td>2 RSA-1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2 RSA-1</td>
<td>2 RSA-1</td>
<td>2 RSA-1</td>
<td>2 RSA-1</td>
<td>2 RSA-1</td>
<td>2 RSA-1</td>
<td>2 RSA-1</td>
</tr>
<tr>
<td>9</td>
<td>2 RSA, 2 DSS</td>
<td>2 RSA, 2 DSS</td>
<td>2 RSA, 2 DSS</td>
<td>2 RSA, 2 DSS</td>
<td>2 RSA, 2 DSS</td>
<td>2 RSA, 2 DSS</td>
<td>2 RSA, 2 DSS</td>
<td>2 RSA, 2 DSS</td>
<td>2 RSA, 2 DSS</td>
<td>2 RSA, 2 DSS</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>6 MD5, 6 SHA</td>
<td>6 MD5, 6 SHA</td>
<td>6 MD5, 6 SHA</td>
<td>6 MD5, 6 SHA</td>
<td>6 MD5, 6 SHA</td>
<td>6 MD5, 6 SHA</td>
<td>6 MD5, 6 SHA</td>
<td>6 MD5, 6 SHA</td>
<td>6 MD5, 6 SHA</td>
<td>6 MD5, 6 SHA</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>6 MD5, 6 SHA</td>
<td>6 MD5, 6 SHA</td>
<td>6 MD5, 6 SHA</td>
<td>6 MD5, 6 SHA</td>
<td>6 MD5, 6 SHA</td>
<td>6 MD5, 6 SHA</td>
<td>6 MD5, 6 SHA</td>
<td>6 MD5, 6 SHA</td>
<td>6 MD5, 6 SHA</td>
<td>6 MD5, 6 SHA</td>
</tr>
</tbody>
</table>
Table 2: The lengths of data processed by the cryptographic primitives.
### Table 22: Computational complexity of the considered security configurations.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>RSA sign.</th>
<th>RSA verif.</th>
<th>DSA sign.</th>
<th>DSA verif.</th>
<th>SHA-1 encrypt.</th>
<th>SHA-1 decrypt.</th>
<th>MD5 encrypt.</th>
<th>MD5 decrypt.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a</td>
<td></td>
<td>1</td>
<td></td>
<td>5</td>
<td>306</td>
<td>512</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1b</td>
<td>1</td>
<td>3</td>
<td></td>
<td>2</td>
<td>439</td>
<td>445</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2a</td>
<td></td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>409</td>
<td>447</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2b</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>392</td>
<td>382</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3a</td>
<td></td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>317</td>
<td>455</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3b</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>442</td>
<td>392</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4a</td>
<td>4</td>
<td>274</td>
<td>2</td>
<td>2</td>
<td>274</td>
<td>283</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4b</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>200</td>
<td>293</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5a</td>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>290</td>
<td>293</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5b</td>
<td></td>
<td>4</td>
<td></td>
<td>1</td>
<td>306</td>
<td>250</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 23: Proportion of the computational latency in the total latency.

<table>
<thead>
<tr>
<th>SC</th>
<th>Low-end device</th>
<th></th>
<th>High-end device</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Moderate</td>
<td>High</td>
<td>Saturated</td>
<td>Moderate</td>
</tr>
<tr>
<td>1a</td>
<td>70.04%</td>
<td>50.44%</td>
<td>2.11%</td>
<td>48.70%</td>
</tr>
<tr>
<td>1b</td>
<td>61.34%</td>
<td>40.81%</td>
<td>1.44%</td>
<td>38.39%</td>
</tr>
<tr>
<td>2a</td>
<td>62.13%</td>
<td>41.35%</td>
<td>1.45%</td>
<td>39.90%</td>
</tr>
<tr>
<td>2b</td>
<td>46.65%</td>
<td>27.28%</td>
<td>0.78%</td>
<td>24.87%</td>
</tr>
<tr>
<td>3a</td>
<td>70.21%</td>
<td>50.24%</td>
<td>2.06%</td>
<td>48.35%</td>
</tr>
<tr>
<td>3b</td>
<td>61.87%</td>
<td>40.80%</td>
<td>1.41%</td>
<td>38.10%</td>
</tr>
<tr>
<td>4a</td>
<td>63.86%</td>
<td>43.16%</td>
<td>1.58%</td>
<td>41.78%</td>
</tr>
<tr>
<td>4b</td>
<td>48.96%</td>
<td>29.29%</td>
<td>0.87%</td>
<td>27.89%</td>
</tr>
<tr>
<td>5a</td>
<td>49.12%</td>
<td>29.42%</td>
<td>0.86%</td>
<td>28.01%</td>
</tr>
<tr>
<td>5b</td>
<td>13.75%</td>
<td>6.47%</td>
<td>0.14%</td>
<td>5.59%</td>
</tr>
</tbody>
</table>

B.2.2 Latencies of the Cryptographic Algorithms

The processing times of the cryptographic algorithms were measured on the two types of devices, described in Subsection 6.2.2. The measurements were performed with the OpenSSL speed test application. The speed test application does not perform MD5 hashing before RSA signing or verification, neither SHA-1 before DSS signing or verification. As a consequence, in case of signing or signature verification, we also calculated with MD5 hashings in case of RSA signing or verification, and SHA-1 hashings in case of DSS signing or verifications. The addition can be seen by comparing Table 20 and Table 21.

B.2.3 Total latency of the security configurations

Section 7 describes the results for the mean total latencies of the methods and their relative performance cost (see Table 8). In Table 23, as an additional information, the proportion of the computational latencies compared to the total latencies are shown.
C Ethereal Trace

The traffic of the security association establishment with TLS_DHE_RSA_with_*-* configuration with a RSA signing certificate at the client side was monitored on the radio channel between the STA and the AP. Table 24 and Table 25 describe the data frames sent on the radio link in time sequence order. The remaining part of this section contains the trace of the communication.
<table>
<thead>
<tr>
<th>No.</th>
<th>Source</th>
<th>Destination</th>
<th>Proto</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3comEuro</td>
<td>Cisco</td>
<td>802.11</td>
<td>Probe Request</td>
</tr>
<tr>
<td></td>
<td>d7:73:a5</td>
<td>8a:c9:b0</td>
<td>EAP</td>
<td>SN=22, FN=0, SSID: tsunami</td>
</tr>
<tr>
<td>2</td>
<td>Cisco</td>
<td>3comEuro</td>
<td>802.11</td>
<td>Probe Response</td>
</tr>
<tr>
<td></td>
<td>8a:c9:b0</td>
<td>d7:73:a5</td>
<td>EAP</td>
<td>SN=1027, FN=0, BI=100, SSID: tsunami, Name: ap</td>
</tr>
<tr>
<td>3</td>
<td>3comEuro</td>
<td>Cisco</td>
<td>802.11</td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>d7:73:a5</td>
<td>8a:c9:b0</td>
<td>EAP</td>
<td>SN=23, FN=0</td>
</tr>
<tr>
<td>4</td>
<td>Cisco</td>
<td>3comEuro</td>
<td>802.11</td>
<td>Authentication</td>
</tr>
<tr>
<td></td>
<td>8a:c9:b0</td>
<td>d7:73:a5</td>
<td>EAP</td>
<td>SN=1028, FN=0</td>
</tr>
<tr>
<td>5</td>
<td>3comEuro</td>
<td>Cisco</td>
<td>802.11</td>
<td>Association Request</td>
</tr>
<tr>
<td></td>
<td>d7:73:a5</td>
<td>8a:c9:b0</td>
<td>EAP</td>
<td>SN=24, FN=0, SSID: tsunami, Name: ap</td>
</tr>
<tr>
<td>6</td>
<td>Cisco</td>
<td>3comEuro</td>
<td>802.11</td>
<td>Association Response</td>
</tr>
<tr>
<td></td>
<td>8a:c9:b0</td>
<td>d7:73:a5</td>
<td>EAP</td>
<td>SN=1029, FN=0, Name: ap</td>
</tr>
<tr>
<td>7</td>
<td>3comEuro</td>
<td>Cisco</td>
<td>802.11</td>
<td>EAP Request, Identity [RFC3748]</td>
</tr>
<tr>
<td></td>
<td>d7:73:a5</td>
<td>8a:c9:b0</td>
<td>EAP</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Cisco</td>
<td>3comEuro</td>
<td>802.11</td>
<td>EAP Response, Identity [RFC3748]</td>
</tr>
<tr>
<td></td>
<td>8a:c9:b0</td>
<td>d7:73:a5</td>
<td>EAP</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>3comEuro</td>
<td>Cisco</td>
<td>802.11</td>
<td>EAP Request, EAP-TLS [RFC2716] [Aboba]</td>
</tr>
<tr>
<td></td>
<td>d7:73:a5</td>
<td>8a:c9:b0</td>
<td>EAP</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Cisco</td>
<td>3comEuro</td>
<td>802.11</td>
<td>TLS Client Hello</td>
</tr>
<tr>
<td></td>
<td>8a:c9:b0</td>
<td>d7:73:a5</td>
<td>TLS</td>
<td>Certificate, Client Key Exchange, Certificate Request, Server Hello Done</td>
</tr>
<tr>
<td>11</td>
<td>3comEuro</td>
<td>Cisco</td>
<td>802.11</td>
<td>TLS Server Hello, Certificate, Server Key Exchange, Certificate Request, Server Hello Done</td>
</tr>
<tr>
<td></td>
<td>d7:73:a5</td>
<td>8a:c9:b0</td>
<td>TLS</td>
<td>Certificate, Client Key Exchange, Certificate Request, Server Hello Done</td>
</tr>
<tr>
<td>12</td>
<td>Cisco</td>
<td>3comEuro</td>
<td>802.11</td>
<td>TLS Server Hello, Certificate, Server Key Exchange, Certificate Request, Server Hello Done</td>
</tr>
<tr>
<td></td>
<td>8a:c9:b0</td>
<td>d7:73:a5</td>
<td>TLS</td>
<td>Certificate, Client Key Exchange, Certificate Request, Server Hello Done</td>
</tr>
<tr>
<td>13</td>
<td>3comEuro</td>
<td>Cisco</td>
<td>802.11</td>
<td>TLS Certificate, Client Key Exchange, Certificate Request, Server Hello Done</td>
</tr>
<tr>
<td></td>
<td>d7:73:a5</td>
<td>8a:c9:b0</td>
<td>TLS</td>
<td>Certificate, Client Key Exchange, Certificate Request, Server Hello Done</td>
</tr>
<tr>
<td>14</td>
<td>Cisco</td>
<td>3comEuro</td>
<td>802.11</td>
<td>TLS Certificate, Client Key Exchange, Certificate Request, Server Hello Done</td>
</tr>
<tr>
<td></td>
<td>8a:c9:b0</td>
<td>d7:73:a5</td>
<td>TLS</td>
<td>Certificate, Client Key Exchange, Certificate Request, Server Hello Done</td>
</tr>
<tr>
<td>15</td>
<td>3comEuro</td>
<td>Cisco</td>
<td>802.11</td>
<td>TLS Certificate, Client Key_exchange, Certificate Request, Server Hello Done</td>
</tr>
<tr>
<td></td>
<td>d7:73:a5</td>
<td>8a:c9:b0</td>
<td>TLS</td>
<td>Certificate, Client Key Exchange, Certificate Request, Server Hello Done</td>
</tr>
<tr>
<td>16</td>
<td>Cisco</td>
<td>3comEuro</td>
<td>802.11</td>
<td>TLS Certificate, Client Key Exchange, Certificate Request, Server Hello Done</td>
</tr>
<tr>
<td></td>
<td>8a:c9:b0</td>
<td>d7:73:a5</td>
<td>TLS</td>
<td>Certificate, Client Key Exchange, Certificate Request, Server Hello Done</td>
</tr>
<tr>
<td>17</td>
<td>3comEuro</td>
<td>Cisco</td>
<td>802.11</td>
<td>TLS Certificate, Client Key Exchange, Certificate Request, Server Hello Done</td>
</tr>
<tr>
<td></td>
<td>d7:73:a5</td>
<td>8a:c9:b0</td>
<td>TLS</td>
<td>Certificate, Client Key Exchange, Certificate Request, Server Hello Done</td>
</tr>
<tr>
<td>18</td>
<td>Cisco</td>
<td>3comEuro</td>
<td>802.11</td>
<td>TLS Certificate, Client Key Exchange, Certificate Request, Server Hello Done</td>
</tr>
<tr>
<td></td>
<td>8a:c9:b0</td>
<td>d7:73:a5</td>
<td>TLS</td>
<td>Certificate, Client Key Exchange, Certificate Request, Server Hello Done</td>
</tr>
<tr>
<td>No.</td>
<td>Source</td>
<td>Destination</td>
<td>Protocol</td>
<td>Info</td>
</tr>
<tr>
<td>-----</td>
<td>-----------------</td>
<td>---------------------</td>
<td>----------</td>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>19</td>
<td>Cisco.Sec610</td>
<td>3com.Enroll.77-73-95</td>
<td>TLS</td>
<td>Change Cipher Spec., Encrypted Handshake Message</td>
</tr>
<tr>
<td>20</td>
<td>3com.Enroll.77-73-95</td>
<td>Cisco.Sec610</td>
<td>EAP</td>
<td>Response, EAP-TLS [RFC2716] [Aboba]</td>
</tr>
<tr>
<td>21</td>
<td>Cisco.Sec610</td>
<td>3com.Enroll.77-73-95</td>
<td>EAP</td>
<td>Success</td>
</tr>
<tr>
<td>22</td>
<td>Cisco.Sec610</td>
<td>3com.Enroll.77-73-95</td>
<td>EAPOL</td>
<td>Key</td>
</tr>
<tr>
<td>23</td>
<td>3com.Enroll.77-73-95</td>
<td>Cisco.Sec610</td>
<td>EAPOL</td>
<td>Key</td>
</tr>
<tr>
<td>24</td>
<td>Cisco.Sec610</td>
<td>3com.Enroll.77-73-95</td>
<td>EAPOL</td>
<td>Key</td>
</tr>
<tr>
<td>25</td>
<td>3com.Enroll.77-73-95</td>
<td>Cisco.Sec610</td>
<td>EAPOL</td>
<td>Key</td>
</tr>
<tr>
<td>No.</td>
<td>Time</td>
<td>Source</td>
<td>Destination</td>
<td>Protocol Info</td>
</tr>
<tr>
<td>-----</td>
<td>------------</td>
<td>-----------------</td>
<td>-------------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>1 0:00:000</td>
<td>xcom_e4773a5</td>
<td>Cisco_e8c9b60</td>
<td>Probe Request Probe Request, SN: 32, FN: 0, SSID: &quot;tsunami&quot;</td>
</tr>
</tbody>
</table>

Frame 1 (40 bytes on wire, 49 bytes captured)  
Arrival Time: Oct 18, 2003 15:47:42.315980000  
Time delta from previous packet: 0.000000000 seconds  
Time since reference or first frame: 0.000000000 seconds  
Frame Number: 1  
Packet Length: 40 bytes  
Capture Length: 40 bytes  
Protocols in frame: wlan

IEEE 802.11  
Type/Subtype: Probe Request (4)  
Frame Control: 802040 (Normal)  
Version: 0  
Type: Management frame (0)  
Subtype: 4  
Flags: 0x0d  
SSID status: Not leaving DS or network is operating in AD-HOC mode (To DS: 0 From DS: 0) (0x00)  
... 0.: More Fragments: This is the last fragment  
... 5.: Retry: Frame is not being retransmitted  
... 0.: PWR MGT: STA will stay up  
... 5.: More Data: No data buffered  
... 0.: Protected flag: Data is not protected  
... 0.: Order flag: Not strictly ordered  
Duration: 314  
Destination address:Cisco_e8c9b60 (00:12:43:8a:b0)  
Source address:xcom_e4773a5 (00:0e:6a:73:a5)  
RSS Id:Cisco_e8c9b60 (00:12:43:8a:b0)  
Fragment number: 0  
Sequence number: 22

IEEE 802.11 wireless LAN management frame  
Tagged parameters (26 bytes)  
SSID parameter set: "tsunami"  
Tag Number: 0 (SSID parameter set)  
Tag length: 7  
Tag interpretation: tsunami  
Supported Rates: 1.0 2.0 5.5 11.0 6.0 12.0 24.0 36.0  
Tag Number: 1 (Supported Rates)  
Tag length: 8  
Tag interpretation: Supported rates: 1.0 2.0 5.5 11.0 6.0 12.0 24.0 36.0  
Extended Supported Rates: 9.0 10.0 18.0 36.0  
Tag Number: 50 (Extended Supported Rates)  
Tag length: 4  
Tag interpretation: Supported rates: 9.0 10.0 18.0 36.0

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol Info</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2 0:00:006</td>
<td>Cisco_e8c9b60</td>
<td>xcom_e4773a5</td>
<td>Probe Response Probe Response, SN: 1027, FN: 0x110, SSID: &quot;tsunami&quot;, Name: &quot;ap&quot;</td>
</tr>
</tbody>
</table>

Frame 2 (185 bytes on wire, 185 bytes captured)  
Time delta from previous packet: 0.000000000 seconds  
Time since reference or first frame: 0.000000000 seconds  
Frame Number: 2  
Packet Length: 185 bytes  
Capture Length: 185 bytes  
Protocols in frame: wlan

IEEE 802.11  
Type/Subtype: Probe Request (0)  
Frame Control: 802050 (Normal)  
Version: 0  
Type: Management frame (0)  
Subtype: 5  
Flags: 0x0d  
SSID status: Not leaving DS or network is operating in AD-HOC mode (To DS: 0 From DS: 0) (0x00)  
... 0.: More Fragments: This is the last fragment  
... 5.: Retry: Frame is not being retransmitted  
... 0.: PWR MGT: STA will stay up  
... 0.: More Data: No data buffered  
... 0.: Protected flag: Data is not protected  
... 0.: Order flag: Not strictly ordered  
Duration: 314  
Destination address:xcom_e4773a5 (00:0e:6a:73:a5)  
Source address:Cisco_e8c9b60 (00:12:43:8a:b0)  
RSS Id:Cisco_e8c9b60 (00:12:43:8a:b0)  
Fragment number: 0  
Sequence number: 1027

IEEE 802.11 wireless LAN management frame  
Fixed parameters (12 bytes)  
Timestamp: 0x000000000000005a  
Beacon Interval: 8120000 [Seconds]  
Capability Information: 0x03f  
... 0.: ESS capabilities: Transmitter is an AP  
... 0.: IBSS status: Transmitter belongs to an IBSS  
... 8.: 0.: CPP participation capabilities: No point coordinator at AP (0x0000)  
... 0.: Privacy: AP/STA can support WEP
Providing Tunable Security in IEEE 802.11i Enabled Networks

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>170</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Frame 3 (30 bytes on wire, 30 bytes captured)
Arrival Time: Oct 18, 2005 15:47:42.318808000
Time delta from previous packet: 0.000822000 seconds
Time since reference or first frame: 0.002918000 seconds
Frame Number: 3

Packet Length: 30 bytes
Capture Length: 30 bytes
Protocols in frame: wlan
IEEE 802.11
Type/Subtype: Authentication (11)
Frame Control: 0x0080 (Normal)
Version: 0
Type: Management frame (0)
Subtype: 11
Flags: 0x0

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol Info</th>
<th>Time since reference or first frame</th>
<th>Arrival Time</th>
<th>Duration</th>
<th>Sequence number</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.003477</td>
<td>Cisco</td>
<td>3comEuro</td>
<td>Authentication Authentication, SSID: &quot;tsunami&quot;, Name: &quot;&quot;</td>
<td>0.000559 seconds</td>
<td>Oct 18, 2005 15:47:42 13997000</td>
<td>314</td>
<td>23</td>
</tr>
</tbody>
</table>

Frame 4 (30 bytes on wire, 30 bytes captured)
Time since reference or first frame: 0.003477000 seconds
Frame Number: 4
Packet Length: 30 bytes
Capture Length: 30 bytes
Protocols in frame: wlan
IEEE 802.11 wireless LAN management frame
Fixed parameters (6 bytes)
Authentication Algorithm: Open System (0)
Authentication Sequence: 0
Status code: Successful (0x0000)

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol Info</th>
<th>Time since reference or first frame</th>
<th>Arrival Time</th>
<th>Duration</th>
<th>Sequence number</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.005006</td>
<td>3comEuro</td>
<td>Cisco</td>
<td>Association Request Association Request, SN:24,PN:0, SSID: &quot;tsunami&quot;, Name: &quot;&quot;</td>
<td>0.003477000 seconds</td>
<td>Oct 18, 2005 15:47:42 13997000</td>
<td>314</td>
<td>23</td>
</tr>
</tbody>
</table>

Frame 5 (126 bytes on wire, 126 bytes captured)
Time since reference or first frame: 0.003477000 seconds
Frame Number: 5
Packet Length: 126 bytes
Capture Length: 126 bytes
Protocols in frame: wlan
IEEE 802.11 wireless LAN management frame
Fixed parameters (6 bytes)
Authentication Algorithm: Open System (0)
Authentication Sequence: 0
Status code: Successful (0x0000)

IEEE 802.11 wireless LAN management frame
Fixed parameters (6 bytes)
Authentication Algorithm: Open System (0)
Authentication Sequence: 0
Status code: Successful (0x0000)
Providing Tunable Security in IEEE 802.11i Enabled Networks

0. … = Protected flag: Data is not protected
2. … = Order flag: Not strictly ordered
Duration: 314
Destination address: Cisco-a8c9b6 (00:12:43:ca:9b:60)
Source address: b0d7:577a:5 (00:12:43:ca:9b:60)

250
BSS Id: Cisco-a8c9b6 (00:12:43:ca:9b:60)
Fragment number: 24

IEEE 802.11 wireless LAN management frame

Fixed parameters (4 bytes)
Capability Information: 0x0411
  ……: 1 = ESS capabilities: Transmitter is an AP
  ……: 0 = ESS status: Transmitter belongs to a BSS
  ……: 0 = CPP participation capabilities: No point coordinator at AP (0xFFFF)
  ……: 1 = Privacy: AP/STA can support WEP

Tagged parameters (80 bytes)
SSID parameter set: "tsunami"
  Tag Number: 6 (SSID parameter set)
  Tag length: 7
  Tag interpretation: tsunami

Supported Rates: 1, 2, 5.5, 11, 18, 24, 36, 48, 54, 6, 9, 12, 18, 24, 36, 48, 54, 72, 90, 108, 144 Mb/Sec

RSN Information
  Tag Number: 48 (RSN Information)
  Tag length: 28
  Tag interpretation: RSN IE, version 1
  Tag interpretation: Multicast cipher suite: AES (CCM)
  Tag interpretation: # of unicast cipher suites: 3
  Tag interpretation: 3 = authentication suite: 1
  Tag interpretation: 3 = key management suite: 1
  Tag interpretation: 3 = pre-authentication: Transmitter does not support pre-authentication
  Tag interpretation: 3 = RSN No Pairwise capabilities: Transmitter can support WEP default key or simultaneously with Pairwise key
  Tag interpretation: 4 = replay counters per PTKSA/STKSA/STKSA-SA (0x0002)
  Tag interpretation: 4 = replay counters per PTKSA/STKSA/STKSA-SA (0x0002)

Cisco Unknown 1 + Device Name
  Tag Number: 113 (Cisco Unknown 1 + Device Name)
  Tag length: 30
  Tag interpretation: Unknown + Name: Reserved tag number: Tag 48 Len 10

Vendor Specific Aironet
  Tag Number: 221 (Vendor Specific)
  Tag length: 5
  Vendor: Aironet
  Tag interpretation: Not interpreted

Vendor Specific Aironet
  Tag Number: 221 (Vendor Specific)
  Tag length: 5
  Vendor: Aironet
  Tag interpretation: Not interpreted

No. Time Source Destination Protocol Info
$ 0.007751 Cisco-a8c9b6 Suefawd-577a:5 Association Response Association

Frame 0 (80 bytes on wire, 80 bytes captured)
Arrival Time Oct 16 2005 10:47:42.327841000
Time delta from previous packet: 0.007751000 seconds
Time since reference or first frame: 0.007751000 seconds
Frame Number: 8
Packet Length: 80 bytes
Capture Length: 80 bytes
Protocols in frame: wlan
IEEE 802.11 wireless LAN management frame

Fixed parameters (8 bytes)
- Capability Information: 0x0411
- ...0 = ESS capabilities: Transmitter is an AP
- ...0 = ESS status: Transmitter belongs to a BSS
- ...0 = CFP participation capabilities: No point coordinator at AP (8:000)
- ...0 = Privacy: AP/STA can support WEP
- ...0 = Short Preamble: Short preamble not allowed
- ...0 = PCCC: PCCC modulation not allowed
- ...0 = Channel Agility: Channel agility not in use

Tagged parameters (58 bytes)
- Supported Rates: 1.0(M) 2.0(M) 5.5(M) 9.0 11.0(M)
- Tag Number: 1 (Supported Rates)
- Tag length: 4
- Tag interpretation: Supported rates: 1.0(M) 2.0(M) 5.5(M) 9.0 11.0(M)
- Cisco Unknown 1 + Device Name
- Cisco Unknown 1 + Device Name
- Tag length: 10
- Tag interpretation: Unknown + Name: ap
- Vendor Specific: Cisco
- Vendor Specific: Cisco
- Vendor Specific: Cisco
- Tag length: 3
- Tag interpretation: Not interpreted

Frame 7 (78 bytes on wire, 78 bytes captured)
Arrival Time: Oct 18, 2005 15:47:42.324265000

IEEE 802.11 Frame
- Frame Control: 0x0203 (Normal)
- Version: 0
- Type: Data frame (2)
- Flags: 0x02
- Duration: 370
  - 0.000624000 seconds

IEEE 802.11 wireless LAN management frame

Tag interpretation:
- Vendor Specific: Cisco
- Vendor Specific: Cisco
- Vendor Specific: Cisco
- Tag length: 3
- Tag interpretation: Not interpreted

No. Time Source Destination Protocol Info
420 7 0.000375 Cisco_xcvr9510 omniSec_47.73a5 EAP Request, Identity [RFC3748]

Frame 7 (78 bytes on wire, 78 bytes captured)
Arrival Time: Oct 18, 2005 15:47:42.324265000

IEEE 802.11 Frame
- Frame Control: 0x0203 (Normal)
- Version: 0
- Type: Data frame (2)
- Flags: 0x02
- Duration: 370
  - 0.000624000 seconds

IEEE 802.11 wireless LAN management frame

Tag interpretation:
- Vendor Specific: Cisco
- Vendor Specific: Cisco
- Vendor Specific: Cisco
- Tag length: 3
- Tag interpretation: Not interpreted
Providing Tunable Security in IEEE 802.11i Enabled Networks

460

**802.1X Authentication**

<table>
<thead>
<tr>
<th>No.</th>
<th>Time (0x03)</th>
<th>Source Destination Protocol Info</th>
<th>EAP Response, Identity [RFC3748]</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>5:31:3080</td>
<td>C0:84:9a:47:73:a5 Cisco_aac3b0</td>
<td>EAP Request, EAP–TLS [RFC2716]</td>
</tr>
</tbody>
</table>

Frame # 8 (78 bytes on wire, 78 bytes captured)
Arrival Time: Oct 18, 2005 15:47:42.638800000
time delta from previous packet: 0.004978000 seconds
Time since reference or first frame: 0.313018000 seconds
Packet Length: 78 bytes
Capture Length: 78 bytes
Protocols in frame: wlan, caps, swap, dlsn

IEEE 802.11

Type/Subtype: Data (32)
Frame Control: 0x1208 (Normal)
Version: 0
Type: Data frame (2)
Subtype: 0

Flags: 0x1

**IEEE 802.11 Authentication**

<table>
<thead>
<tr>
<th>No.</th>
<th>Time (0x03)</th>
<th>Source Destination Protocol Info</th>
<th>EAP Response, Identity [RFC3748]</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>0:31:3080</td>
<td>C0:84:9a:47:73:a5 Cisco_aac3b0</td>
<td>EAP Request, EAP–TLS [RFC2716]</td>
</tr>
</tbody>
</table>

Frame # 9 (78 bytes on wire, 78 bytes captured)
Arrival Time: Oct 18, 2005 15:47:42.638800000
time delta from previous packet: 0.004978000 seconds
Time since reference or first frame: 0.313018000 seconds
Packet Length: 78 bytes
Capture Length: 78 bytes
Protocols in frame: wlan, caps, swap, dlsn

IEEE 802.11

Type/Subtype: Data (32)
Frame Control: 0x1208 (Normal)
Version: 0
Type: Data frame (2)
Subtype: 0

Flags: 0x2

**IEEE 802.11 Authentication**

<table>
<thead>
<tr>
<th>No.</th>
<th>Time (0x03)</th>
<th>Source Destination Protocol Info</th>
<th>EAP Response, Identity [RFC3748]</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0:31:3080</td>
<td>C0:84:9a:47:73:a5 Cisco_aac3b0</td>
<td>EAP Request, EAP–TLS [RFC2716]</td>
</tr>
</tbody>
</table>

Frame # 10 (78 bytes on wire, 78 bytes captured)
Arrival Time: Oct 18, 2005 15:47:42.638800000
time delta from previous packet: 0.004978000 seconds
Time since reference or first frame: 0.313018000 seconds
Packet Length: 78 bytes
Capture Length: 78 bytes
Protocols in frame: wlan, caps, swap, dlsn

IEEE 802.11

Type/Subtype: Data (32)
Frame Control: 0x1208 (Normal)
Version: 0
Type: Data frame (2)
Subtype: 0

Flags: 0x2
550
... 5. ..... = More Data: No data buffered
... 6. ..... = Protected flag: Data is not protected
... 6. ..... = Order flag: Not strictly ordered
Duration: 215
Destination address: 00:12:43:8a:b0:1d
BSSID: 00:12:43:8a:b0:1d
Frame length: 950
Fragment number: 0
Sequence number: 1024
Logical–Link Control

560
DSAP: SNAP (dot)
IG Bit: Individual
SSAP: SNAP (dot)
CR Bit: Compressed
Control field: U. func/UI (0x80)
000 00. = Command: Unnumbered Information (0x80)
11 = Frame type: Unnumbered frame (0x80)
Organization Code: Encapsulated Ethernet (0x0800)
Type: 802.1X Authentication (0x888e)
802.1X Authentication

570
Version: 1
Type: EAP Packet (0)
Length: 6
Extensible Authentication Protocol
Code: Request (1)
Id: 2
Length: 6
Type: EAP–TLS [RFC2716] [Aboba] (13)
Flags(S29): Start

580
No. Time Source Destination Protocol Info
10 04:07:00 3comEuro_87:73:s5 Cisco_8ac9:b0 TLS Client Hello

Frame ID: 142 bytes on wire, 142 bytes captured
Arrival Time: Oct 18, 2005 15:47:42.723390000
Time delta from previous packet: 0.0000540000 seconds
Time since reference or first frame: 0.4075000000 seconds
Frame Number: 10
Packet Length: 142 bytes
Capture Length: 142 bytes
Protocols found inside: wlan

IEEE 802.11
Type/Subtype: Data (32)
Frame Control: 0x0108 (Normal)
Version: 0
Type: Data frame (2)
Subtype: 0
Flags: 0x8

600
... 5. ..... = More Fragments: This is the last fragment
... 6. ..... = Retry: Frame is not being retransmitted
... 6. ..... = PWR MGT: STA will stay up
... 6. ..... = More Data: No data buffered
... 6. ..... = Protected flag: Data is not protected
... 6. ..... = Order flag: Not strictly ordered
Duration: 214
BSSID: 00:12:43:8a:b0:1d
Source address: 00:12:43:8a:b0:1d
Fragment number: 0
Sequence number: 26
Logical–Link Control

610
Destination address: Cisco_8ac9:b0 (00:12:43:8a:b0:1d)
Frame length: 950
Fragment number: 0
Sequence number: 1024
Logical–Link Control

802.1X Authentication
Version: 1
Type: EAP Packet (0)
Length: 106
Extensible Authentication Protocol
Code: Response (2)
Id: 2

630
Type: EAP–TLS [RFC2716] [Aboba] (13)
Flags(S29): Length
Length: 96
Secure Socket Layer
TLS Record Layer: Handshake Protocol: Client Hello
Content Type: Handshake (22)
Version: TLS 1.0 (0x0001)
Length: 81
Handshake Protocol: Client Hello

640
Handshake Type: Client Hello (1)
Length: 87
Providing Tunable Security in IEEE 802.11i Enabled Networks

No. Time Source Destination Protocol Info
11 0.42197 Cisco_xa:9.b0 Sx509sat 373:1x5 TLS Server Hello, Certificate, Server Key Exchange, Certificate Request, Server Hello Done

Frame 11 (1270 bytes on wire, 1270 bytes captured)
Arrival Time: Oct 16, 2005 15:47:42.741870000
Time delta from previous packet: 0.017697000 seconds
Time since reference or first frame: 0.425197000 seconds
Packet Number: 11
Packet Length: 1270 bytes
Capture Length: 1270 bytes

Protocols in frame (framed): wlan80211cap, pcap
Frame from DS to a STA via AP (To STA) 0 From DS: 1 (0x02)
... 03: MORE FRAGMENTS: This is the last fragment
... 00: EAP Packet (0x02)
... 03: MORE DATA: No data buffered
... 00: ORDER FLAG: Not strictly ordered
Duration: 213
Destination address: Sx509sat:773:1x5 (0x5c:fe:a7:73:1x5)
Source address: Cisco_xa:9.b0 (0x12:43:8a:9b:0)
Fragment number: 0
Sequence number: 1036

Logical--Link Control
DSAP: SNAP (0xa)
ISAP: Individual
SSAP: SNAP (0xa)
CR: Dot Command
Control field: U: func=UI (0x05)
300. 6: Command: Unnumbered Information (0x06)
... 11: Frame type: Unnumbered frame (0x00)
Organization Code: Encapsulated Ethernet (0x000000)
Type: 16 Authentication (0x006)
802.1X Authentication
Version: 1
Type: EAP (0x01)
Length: 0
Extensible Authentication Protocol
Code: Request (1)
Id: 3
Length: 1034
Type: EAP (0x01)
Length: 0
Extensible Authentication Protocol
Code: Request (1)
Id: 3
Length: 1034
Frame 11, payload: 0–103 (1024 bytes)
Frame 13, payload: 1024–2047 (1024 bytes)
Frame 15, payload: 2048–2089 (42 bytes)

Secure Socket Layer

TLS Record Layer: Handshake Protocol: Server Hello

Content Type: Handshake (12)
Version: TLS 1.0 (0x0301)
Length: 74
Handshake Protocol: Server Hello
Handshake Type: Server Hello (2)
Length: 70
Version: TLS 1.0 (0x0301)
Random:bytes Oct 18, 2005 16:07:07.000000000
Session ID: Length 32
Cipher Suite: TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x0003)

Compression Method: null (0)

TLS Record Layer: Handshake Protocol: Certificate
Content Type: Handshake (22)
Version: TLS 1.0 (0x0301)
Length: 2573
Handshake Protocol: Certificate
Handshake Type: Certificate (11)
Length: 1589
Certificate Length: 1588
Certificates (1588 bytes)
Certificate: 03020100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
IASString: szlaj@mcl.hu

validity
notBefore: utcTime (0)
notAfter: utcTime (0)
utctime: 0101235921Z

subject: rdnSequence (8)

subjectPublicKeyInfo (7 items)

subjectKeyIdentifier (4 items)

subjectKeyIdentifier (7 items)

authorityKeyIdentifier (2 items)

extensions (5 items)

directoryName: rdnSequence (8)
Frame 12 (78 bytes on wire, 78 bytes captured)
Arrival Time: Oct 18, 2005 15:47:42.743684000
Time delta from reference or first frame: 0.002597000 seconds
Packet Length: 78 bytes
Capture Length: 78 bytes

IEEE 802.11 Type/Data: EAPOL
Frame Control: 0x028E (Normal)
Version: 0
Type: Data frame (2)
Subtype: 0
Flags: 0d
DS status: Frame from STA to DS via an AP (To DS: 1 Prom DS: 0) (0x01)
... 5: More Fragments: This is the last fragment
... 8: More: Frame is not being retransmitted
... 9: PWR MGT: STA will stay up
... 6: Protected flag: Data is not protected
... 7: Order flag: Not strictly ordered

Duration: 316
BSS Id: Cisco_x8c96b0 (00:12:43:x8c96b0)
Source address: fae8:c777:a5 (00:0e:6a:73:a5)
Destination address: Cisco_x8c96b0 (00:12:43:x8c96b0)
Fragment number: 0
Sequence number: 27

Logical-Link Control
DSAP: SNAP (0xa)
DSAP: SNAP (0xa)

CR Bit: Command
Control field: U, func=U (0x03)
### 802.1X Authentication

**Version:** 1  
**Type:** EAP Packet (0)  
**Length:** 6

#### Extensible Authentication Protocol

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1120</td>
<td>0.503526</td>
<td>Cisco 0xa:9b:00</td>
<td>x509sat 47:73:a5</td>
<td>TLS Server Hello, Certificate, Server Key Exchange, Certificate Request, Server Hello Done</td>
</tr>
</tbody>
</table>

Frame 13 (1070 bytes on wire, 1070 byte captured)  
Arrival Time: Oct 18, 2005 15:47:43.41941000  
Time delta from previous packet: 0.075732000 seconds  
Time since reference or first frame: 0.503526000 seconds  
Frame Number: 13

Capture Length: 1070 bytes

Protocols in frame [truncated]:  
- 802.1X Authentication
- EAP-TLS [RFC2716] [Ahobas] (13)

Flags (0x00):

- 0x00 = Command: Unnumbered Information
- 0x01 = Frame type: Unnumbered frame

Organization Code: Encapsulated Ethernet (0x0000)

#### EAP-TLS

**Version:** 1  
**Type:** EAP Packet (0)  
**Length:** 1024

#### Extensible Authentication Protocol

**Version:** 1  
**Type:** EAP Packet (0)  
**Length:** 1024

**Extensible Authentication Protocol**

**Version:** 1  
**Type:** EAP Packet (0)  
**Length:** 1024

#### EAP-TLS

**Version:** 1  
**Type:** EAP Packet (0)  
**Length:** 1024

#### Extensible Authentication Protocol

**Version:** 1  
**Type:** EAP Packet (0)  
**Length:** 1024

#### EAP-TLS

**Version:** 1  
**Type:** EAP Packet (0)  
**Length:** 1024

#### Extensible Authentication Protocol

**Version:** 1  
**Type:** EAP Packet (0)  
**Length:** 1024

#### EAP-TLS

**Version:** 1  
**Type:** EAP Packet (0)  
**Length:** 1024

#### Extensible Authentication Protocol

**Version:** 1  
**Type:** EAP Packet (0)  
**Length:** 1024

#### EAP-TLS

**Version:** 1  
**Type:** EAP Packet (0)  
**Length:** 1024

#### Extensible Authentication Protocol

**Version:** 1  
**Type:** EAP Packet (0)  
**Length:** 1024

#### EAP-TLS

**Version:** 1  
**Type:** EAP Packet (0)  
**Length:** 1024

#### Extensible Authentication Protocol

**Version:** 1  
**Type:** EAP Packet (0)  
**Length:** 1024

#### EAP-TLS

**Version:** 1  
**Type:** EAP Packet (0)  
**Length:** 1024

#### Extensible Authentication Protocol

**Version:** 1  
**Type:** EAP Packet (0)  
**Length:** 1024

#### EAP-TLS

**Version:** 1  
**Type:** EAP Packet (0)  
**Length:** 1024

#### Extensible Authentication Protocol

**Version:** 1  
**Type:** EAP Packet (0)  
**Length:** 1024

#### EAP-TLS

**Version:** 1  
**Type:** EAP Packet (0)  
**Length:** 1024

#### Extensible Authentication Protocol

**Version:** 1  
**Type:** EAP Packet (0)  
**Length:** 1024

#### EAP-TLS

**Version:** 1  
**Type:** EAP Packet (0)  
**Length:** 1024

#### Extensible Authentication Protocol

**Version:** 1  
**Type:** EAP Packet (0)  
**Length:** 1024
Providing Tunable Security in IEEE 802.11i Enabled Networks

TLS Record Layer Handshake Protocol: Certificate

Content Type: Handshake (22)
Version: TLS 1.0 (0x0301)
Length: 1200

Handshake Type: Certificate (21)
Length: 1250
Certificates (1585 bytes)
Certificate Length: 685
Certificate: B9E5F480EB5002D0B0A4D7107F2684C4463E604D1: (0x0282.113540.1.1:postmaster@localhost.id-at-commonName=testserver.id-at-organizationalUnitName=MCL.id-at-organizationName=BME-HT.id-at-stateOrProvinceName=Budapest.id-at-countryName=HU)
Algorithm: rdnSequence (9)
sigAlg: md5WithRSAEncryption (2)
length: 1250
serialNumber: 0x05273207829204444
Algorithm: id-dsaWithRSAEncryption (3)

Handshake Protocol: Handshake
Length: 560

Handshake Type: Certificate (21)
Length: 1200
Certificates (1585 bytes)
Certificate Length: 685
Certificate: B9E5F480EB5002D0B0A4D7107F2684C4463E604D1: (0x0282.113540.1.1:postmaster@localhost.id-at-commonName=testserver.id-at-organizationalUnitName=MCL.id-at-organizationName=BME-HT.id-at-stateOrProvinceName=Budapest.id-at-countryName=HU)
Algorithm: rdnSequence (9)
sigAlg: md5WithRSAEncryption (2)
length: 1250
serialNumber: 0x05273207829204444
Algorithm: id-dsaWithRSAEncryption (3)

Handshake Type: Certificate (21)
Length: 1200
Certificates (1585 bytes)
Certificate Length: 685
Certificate: B9E5F480EB5002D0B0A4D7107F2684C4463E604D1: (0x0282.113540.1.1:postmaster@localhost.id-at-commonName=testserver.id-at-organizationalUnitName=MCL.id-at-organizationName=BME-HT.id-at-stateOrProvinceName=Budapest.id-at-countryName=HU)
Algorithm: rdnSequence (9)
sigAlg: md5WithRSAEncryption (2)
length: 1250
serialNumber: 0x05273207829204444
Algorithm: id-dsaWithRSAEncryption (3)

Certificate Length: 685
Certificate: B9E5F480EB5002D0B0A4D7107F2684C4463E604D1: (0x0282.113540.1.1:postmaster@localhost.id-at-commonName=testserver.id-at-organizationalUnitName=MCL.id-at-organizationName=BME-HT.id-at-stateOrProvinceName=Budapest.id-at-countryName=HU)
Algorithm: rdnSequence (9)
sigAlg: md5WithRSAEncryption (2)
length: 1250
serialNumber: 0x05273207829204444
Algorithm: id-dsaWithRSAEncryption (3)

Certificate Length: 685
Certificate: B9E5F480EB5002D0B0A4D7107F2684C4463E604D1: (0x0282.113540.1.1:postmaster@localhost.id-at-commonName=testserver.id-at-organizationalUnitName=MCL.id-at-organizationName=BME-HT.id-at-stateOrProvinceName=Budapest.id-at-countryName=HU)
Algorithm: rdnSequence (9)
sigAlg: md5WithRSAEncryption (2)
length: 1250
serialNumber: 0x05273207829204444
Algorithm: id-dsaWithRSAEncryption (3)

Certificate Length: 685
Certificate: B9E5F480EB5002D0B0A4D7107F2684C4463E604D1: (0x0282.113540.1.1:postmaster@localhost.id-at-commonName=testserver.id-at-organizationalUnitName=MCL.id-at-organizationName=BME-HT.id-at-stateOrProvinceName=Budapest.id-at-countryName=HU)
Algorithm: rdnSequence (9)
sigAlg: md5WithRSAEncryption (2)
length: 1250
serialNumber: 0x05273207829204444
Algorithm: id-dsaWithRSAEncryption (3)

Certificate Length: 685
Certificate: B9E5F480EB5002D0B0A4D7107F2684C4463E604D1: (0x0282.113540.1.1:postmaster@localhost.id-at-commonName=testserver.id-at-organizationalUnitName=MCL.id-at-organizationName=BME-HT.id-at-stateOrProvinceName=Budapest.id-at-countryName=HU)
Algorithm: rdnSequence (9)
sigAlg: md5WithRSAEncryption (2)
length: 1250
serialNumber: 0x05273207829204444
Algorithm: id-dsaWithRSAEncryption (3)
DirectoryString: printableString (1) printableString: texterver
Item: 1 item (iso:2.840.113549.1.1:postmaster@localhost)
Item: (iso:2.840.113549.1.1:postmaster@localhost)
id: 1.2.840.113549.1.1 (iso:2.840.113549.1.1)

BER: Directory for OID:1.2.840.113549.1.1 not implemented. Contact IAI developers if you want this supported.
IA5String: postmaster@localhost

subjectPublicKeyInfo
algorithm: id=rsaEncryption
Algorithm Id: 1.2.840.113549.1.1 (id=rsaEncryption)
Padding: 0

subjectPublicKey:
30820802100006092A8648CE040002010002010000320100060C2A000A028201016A

extensions: 1 item
Item: (id=ce-extension)
Extension Id: 2.5.28.37 (id=ce-extension)
KeyUsage: 1 item
1.3.6.1.5.5.7.1 (id=serverAuth)
AlgorithmIdentifier: (id=MD5WithRSAEncryption)
Algorithm Id: 1.2.840.113549.1.1 (id=MD5WithRSAEncryption)
Padding: 0

encrypted: EBF9F7B401503B97734ED68DC338E0B933333A1D020F

Certificate Length: 955

Certificate:
302608020100020100000500553000043001020205001E07064044430D0609...

serialNumber: 060772107029404409
signature (md5WithRSAEncryption)
Algorithm Id: 1.2.840.113549.1.1 (id=md5WithRSAEncryption)
issuer: rdnSequence

rdnSequence: 7 items (iso:2.840.113549.1.1:szlaj@bme.hu)
1. at=commonName=Faigl Zoltan
2. at=organizationalUnitName=MCL
3. at=organizationName=BMW:=HT
4. at=countryName=HU
5. at=countryName=HU
6. at=countryName=HU
7. at=countryName=HU

Item: 1 item (id=countryName)
Item: 1 item (id=stateOrProvinceName)
Item: 1 item (id=localityName)
Item: 1 item (id=organizationName)
Item: 1 item (id=organizationalUnitName)
Item: 1 item (id=organizationalUnitName)
Item: 1 item (id=organizationalUnitName)
Item: 1 item (id=organizationalUnitName)

validity
notBefore: utcTime (0)
notAfter: utcTime (0)

subject: rdnSequence

rdnSequence: 7 items (iso:2.840.113549.1.1:szlaj@bme.hu)
1. at=commonName=Faigl Zoltan
2. at=organizationalUnitName=MCL
3. at=organizationName=BMW:=HT
4. at=countryName=HU
5. at=stateOrProvinceName=Budapest
6. at=countryName=HU
7. at=countryName=HU

Item: 1 item (id=countryName)
Item: 1 item (id=stateOrProvinceName)
Item: 1 item (id=countryName)

DirectoryString: printableString (1) printableString: texterver
Item: 1 item (iso:2.840.113549.1.1:postmaster@localhost)
Item: (iso:2.840.113549.1.1:postmaster@localhost)
id: 1.2.840.113549.1.1 (iso:2.840.113549.1.1)

BER: Directory for OID:1.2.840.113549.1.1 not implemented. Contact IAI developers if you want this supported.
IA5String: postmaster@localhost

subjectPublicKeyInfo
algorithm: id=rsaEncryption
Algorithm Id: 1.2.840.113549.1.1 (id=rsaEncryption)
Padding: 0

subjectPublicKey:
30820802100006092A8648CE040002010002010000320100060C2A000A028201016A

extensions: 1 item
Item: (id=ce-extension)
Extension Id: 2.5.28.37 (id=ce-extension)
KeyUsage: 1 item
1.3.6.1.5.5.7.1 (id=serverAuth)
AlgorithmIdentifier: (id=MD5WithRSAEncryption)
Algorithm Id: 1.2.840.113549.1.1 (id=MD5WithRSAEncryption)
Padding: 0

encrypted: EBF9F7B401503B97734ED68DC338E0B933333A1D020F

Certificate Length: 955

Certificate:
302608020100020100000500553000043001020205001E07064044430D0609...

serialNumber: 060772107029404409
signature (md5WithRSAEncryption)
Algorithm Id: 1.2.840.113549.1.1 (id=md5WithRSAEncryption)
issuer: rdnSequence

rdnSequence: 7 items (iso:2.840.113549.1.1:szlaj@bme.hu)
1. at=commonName=Faigl Zoltan
2. at=organizationalUnitName=MCL
3. at=organizationName=BMW:=HT
4. at=countryName=HU
5. at=countryName=HU
6. at=countryName=HU
7. at=countryName=HU

Item: 1 item (id=countryName)
Item: 1 item (id=stateOrProvinceName)
Item: 1 item (id=localityName)
Item: 1 item (id=organizationName)
Item: 1 item (id=organizationalUnitName)
Item: 1 item (id=organizationalUnitName)
Item: 1 item (id=organizationalUnitName)
Item: 1 item (id=organizationalUnitName)

validity
notBefore: utcTime (0)
notAfter: utcTime (0)

subject: rdnSequence

rdnSequence: 7 items (iso:2.840.113549.1.1:szlaj@bme.hu)
1. at=commonName=Faigl Zoltan
2. at=organizationalUnitName=MCL
3. at=organizationName=BMW:=HT
4. at=countryName=HU
5. at=stateOrProvinceName=Budapest
6. at=countryName=HU
7. at=countryName=HU

Item: 1 item (id=countryName)
Item: 1 item (id=stateOrProvinceName)
Item: 1 item (id=countryName)

DirectoryString: printableString (1)
<table>
<thead>
<tr>
<th>Item</th>
<th>1430</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>subjectPublicKey</strong></td>
<td>1440</td>
</tr>
<tr>
<td>Item</td>
<td>1445</td>
</tr>
<tr>
<td>Item</td>
<td>1460</td>
</tr>
</tbody>
</table>

**Providing Tunable Security in IEEE 802.11i Enabled Networks**

63
Frame 14 (78 bytes on wire, 78 bytes captured)

Time delta from previous packet: 0.003234000 seconds

Frame Number: 14
Packet Length: 78 bytes
Capture Length: 78 bytes
Protocols in frame: wlan

IEEE 802.11
Type/Subtype: Data (32)
Frame Control: 0x00 (Normal)
Version: 0
Type: Data frame (2)
Subtype: 0
Flag: No

DL status: Frame from STA to DS via an AP (To DS: 1 From DS: 0) (0x01)
... 0... More Fragments: This is the last fragment
... 0... Retry: Frame is not being retransmitted
... 0... More Data: No data buffered
... 0... Protected flag: Data is not protected
... 0... Order flag: Not strictly ordered

Duration: 0.506760000 seconds

IEEE 802.1X Authentication
Type: EAP Packet (0)
Length: 0
Extensible Authentication Protocol
Code: Response (2)
Id: 4
Length: 0
Type: EAP−TLS [RFC2716] [Aboba] (13)
Flags(0x00)
Frame 15 (88 bytes on wire, 88 bytes captured)
Arrival Time: Oct 16, 2005 15:47:42.631740000

Time delta from previous packet: 0.000600000 seconds
Time since reference or first frame: 0.535583200 seconds
Frame Number: 15
Packet Length: 88 bytes
Capture Length: 88 bytes

Protocols in frame (from left to right): wlan1/lecapol/soapke-pkcs -1x5e98aet x509sat x509aet x509sat x509aet x509sat x509sat x509aet

IEEE 802.11

Table 15.80

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>0.535588</td>
<td>Cisc0x-8a-9b-0</td>
<td>x509sat,47-73-5</td>
<td>TLS: Server Hello, Certificates, Server Key Exchange, Certificate Request, Server Hello Done</td>
</tr>
</tbody>
</table>

1570

Version: 0
Type: Data frame (2)
Subtype: 0
Flag(s): 0x02
...DS status: Frame from DS to a STA via AP/To DS: 0 From DS: 1 (0x02)
...F: 0x0 More Fragments: This is the last fragment
...B: 0x0 Retransmit: Frame is not being retransmitted
...0x0 More Data: STA will stay up
...0x0 More Data: No data buffered
...0x0 Protected flag: Data is not protected
...0x0 Order flag: Not strictly ordered

Duration: 213
Destination address: 30x509sat,47-73-5 (00:0e:6a:73:01)
BSSID: Cisc0x-8a-9b-0 (00:12:43:8a:9b:0)
Source address: Cisc0x-8a-9b-0 (00:12:43:8a:9b:0)

Fragment number: 0
Sequence number: 1029

Logical-Link Control

DSAP: SNAP (80a)
SSAP: SNAP (80a)
CR Bit: Command
Control field: U, fcs=UI (0x00)
000 00: 0x0 Command: Unnumbered Information (0x09)
...11: Frame type Unnumbered frame (0x02)
Organization Code: Encapsulated Ethernet (0x000000)

Type: 802.1X Authentication (0x0868)

802.1X Authentication

Version: 1

Type: EAP Packet (0)
Length: 52

Extensible Authentication Protocol

Code: Request (1)
Id: 5
Length: 52
Type: EAP-TLS [RFC2716] [Aboba] (13)
Flags(80a): Length
Length: 209

EAP-TLS Fragments (2090 bytes): #11(1024), #12(1024), #13(42)

Frame: 11, payload: 0x1023 (1024 bytes)
Frame: 12, payload: 1024–2047 (1024 bytes)
Frame: 13, payload: 2048–3071 (42 bytes)

Secure Socket Layer

TLS Record Layer Handshake Protocol: Server Hello
Content Type: Handshake (22)
Version: TLS 1.0 (0x30)
Length: 74

Handshake Protocol: Server Hello
Handshake Protocol: Server Hello (2)

Length: 70
Version: TLS 1.0 (0x30)
Random gen_unix_time: Oct 18, 2005 16:07:07.000000000
Random bytes
Session ID Length: 32
Session ID (32 bytes)
Cipher Suite: TLS-DHE_RSA_WITH_AES_256_CBC_SHA (0x0019)
Compression Method: null (0)

TLS Record Layer Handshake Protocol: Certificate
Content Type: Handshake (22)
Version: TLS 1.0 (0x30)
Length: 1573

Handshake Protocol: Certificate
Handshake Protocol: Certificate (11)

Length: 1569
Certificates Length: 1566
Certificates (1566 bytes)
Certificate Length: 685
Certificate: 38301FADA3D3C52322320900A67107063C444630D068
...() (0x2.040.11294.1.1.2: postmaster@localhost, id = at - commonName: testserver, id = at - organizationalUnitName: MCI, id = at - organizationName: RIEEEH, id = at - stateOrProvinceName: Budapest, id = signedCertificate
...version: v3 (2)
serialNumber : 00da7210ff2ad0048d7f
signature (md5WithRSAEncryption)
Algorithm id: 1.2.840.113549.1.4 (md5WithRSAEncryption)
inner: rdSequence (8)
  rdSequence: 7 items (iso.2.5.4.11.1.1 := szlaj@nucl.hu, id := at := commonName = Faigl Zoltan, id := at := organizationalUnitName = MCL, id := at := organizationName = BME-HT, id := at := localityName = Budapest, id := at := stateOrProvinceName = Budapest, id := at := countryName = HU)
Item: 1 item (id := at := countryName = HU)
  id := 2.5.4.8 (id := at := countryName)
  CountryName: HU
Item: 1 item (id := at := stateOrProvinceName = Budapest)
  id := 2.5.4.8 (id := at := stateOrProvinceName)
  DirectoryString: printableString (8)
  printableString: Budapest
Item: 1 item (id := at := localityName = Budapest)
  id := 2.5.4.7 (id := at := localityName)
  DirectoryString: printableString (4)
  printableString: Budapest
Item: 1 item (id := at := organizationName = BME-HT)
  id := 2.5.4.11 (id := at := organizationName)
  DirectoryString: printableString (4)
  printableString: BME-HT
Item: 1 item (id := at := organizationalUnitName = MCL)
  id := 2.5.4.11 (id := at := organizationalUnitName)
  DirectoryString: printableString (4)
  printableString: MCL
Item: 1 item (id := at := commonName = Faigl Zoltan)
  id := 2.5.4.3 (id := at := commonName)
  DirectoryString: printableString (4)
  printableString: Faigl Zoltan
Item: 1 item (iso.2.5.4.11.1.1 := szlaj@nucl.hu)
  id := 1.2.840.113549.1.1.1.1 (iso.2.5.4.11.1.1)
  BER: Dissector for OID 1.2.840.113549.1.1 not implemented: Contact Ethereal developers if you want this supported.
  IASString: szlaj@nucl.hu
validity notBefore: utcTime (0)
utcTime: 051023143137
notAfter: utcTime (0)
utcTime: 051023143137
subject: rdSequence (8)
  rdSequence: 6 items (iso.2.5.4.11.1.1 := postmaster@LocalHost, id := at := commonName = testserver, id := at := organizationalUnitName = MCL, id := at := organizationName = BME-HT, id := at := stateOrProvinceName = Budapest, id := at := countryName = HU)
Item: 1 item (id := at := countryName = HU)
  id := 2.5.4.8 (id := at := countryName)
  CountryName: HU
Item: 1 item (id := at := stateOrProvinceName = Budapest)
  id := 2.5.4.8 (id := at := stateOrProvinceName)
  DirectoryString: printableString (4)
  printableString: Budapest
Item: 1 item (id := at := organizationName = BME-HT)
  id := 2.5.4.11 (id := at := organizationName)
  DirectoryString: printableString (4)
  printableString: BME-HT
Item: 1 item (id := at := organizationalUnitName = MCL)
  id := 2.5.4.11 (id := at := organizationalUnitName)
  DirectoryString: printableString (4)
  printableString: MCL
Item: 1 item (id := at := commonName = testserver)
  id := 2.5.4.3 (id := at := commonName)
  DirectoryString: printableString (4)
  printableString: testserver
Item: 1 item (iso.2.5.4.11.1.1 := postmaster@LocalHost)
  id := 1.2.840.113549.1.1.1.1 (iso.2.5.4.11.1.1)
  BER: Dissector for OID 1.2.840.113549.1.1 not implemented: Contact Ethereal developers if you want this supported.
  IASString: postmaster@localhost
subjectPublicKeyInfo
  algorithm (rsaEncryption)
  Algorithm Id: 1.2.840.113549.1.1 (rsaEncryption)
  PublicKeyInfo: 5
  subjectPublicKey: 30819d020103490009d0c07414a5e9177fd467...
  extensions: 1 item
    id := 2.5.29 := KeyUsage
    Item: 1 item (id := 2.5.29.1 := digitalSignature)
    Item: 1 item (id := 2.5.29.2 := nonRepudiation)
    Item: 1 item (id := 2.5.29.3 := keyEncipherment)
    Item: 1 item (id := 2.5.29.4 := dataEncipherment)
    Item: 1 item (id := 2.5.29.5 := keyAgreement)
    Item: 1 item (id := 2.5.29.6 := keyCertSign)
    Item: 1 item (id := 2.5.29.7 := cRLSign)
    Item: 1 item (id := 2.5.29.8 := encipherOnly)
    Item: 1 item (id := 2.5.29.9 := decipherOnly)

...
1840

subjectPublicKeyInfo

Item 1 item (id-at-commonName:Faigl Zoltán)

Id: 25.8.3 (id-at-commonName)

directoryString : printableString

Padding: 9
subjectPublicKey: Faigl Zoltán

Item 1 item (idx:2.840.113549.1.9.1licated)

Id: 1.2.840.113549.1.9.1 (idx:2.840.113549.1.9.1licated)

DER: Dissector for OID:1.2.840.113549.1.9.1 not implemented. Contact Ethereal developers if you want this supported

1.2.840.113549.1.9.1->_1.2.840.113549.1.9.1licated

1850

subjectPublicKeyInfo

Algorithm: id:2.840.113549.1.11 [realEncryption]

Padding: 9

subjectPublicKey: Faigl Zoltán

Item 1 item (idx:2.840.113549.1.9.1licated)

Id: 1.2.840.113549.1.9.1 (idx:2.840.113549.1.9.1licated)

extension: 1 items

1860

subjectKeyIdentifier

Extension Id: 25.6.25 (id-ce-SubjectKeyId)

AuthorityKeyId: 7888A4DAA5E4684E164BEC0B1E834305E29012E9

authorityCertIssuer: 1 item

directoryName: rdnSequence

directoryName: dir:sequence: 7 items

id-at-organizationalUnitName: MCL

id.at-stateOrProvinceName: Budapest

id-at-countryName: HU

id-at-organizationName: BME–HT

id-at-localityName: Budapest

Id: 1.2.840.113549.1.9.1licated

countryName: HU

1870

subjectPublicKeyInfo

Item 1 item (idx:2.840.113549.1.9.1licated)

Id: 25.8.4 (id-at-countryName)

directoryString : printableString

Padding: 9

subjectPublicKey: Faigl Zoltán

Item 1 item (idx:2.840.113549.1.9.1licated)

Id: 1.2.840.113549.1.9.1 (idx:2.840.113549.1.9.1licated)

DER: Dissector for OID:1.2.840.113549.1.9.1 not implemented. Contact Ethereal developers if you want this supported

1.2.840.113549.1.9.1->_1.2.840.113549.1.9.1licated

1890

subjectPublicKeyInfo

Item 1 item (idx:2.840.113549.1.9.1licated)

Id: 1.2.840.113549.1.9.1 (idx:2.840.113549.1.9.1licated)

DER: Dissector for OID:1.2.840.113549.1.9.1 not implemented. Contact Ethereal developers if you want this supported

1.2.840.113549.1.9.1->_1.2.840.113549.1.9.1licated

authorityCertSerialNumber : 00a72707063464449

Item 1 item (idx:2.840.113549.1.9.1icated)

Extension Id: 25.6.25 (id-ce-BasicConstraints)

BasicConstraints: Syntax: A: True

algorithmIdentifier: (md5WithRSAEncryption)

Algorithm Id: 1.2.840.113549.1.14 [md5WithRSAEncryption]

Padding: 8

encrypted: 5CDD55D7BD4C49B1A0D2574399366E655B1B63AF41C64...
### 1930
Content Type: Handshake (22)
- **Version**: TLS 1.0 (0x30)
- **Length**: 104
  - **Handshake Type**: Certificate Request
  - **Handshake Length**: 146
  - **Certificate types count**: 4
  - **Certificate types (4 types)**:
    1. Certificate type RSA Fixed DH (3)
    2. Certificate type DSS Fixed DH (4)
    3. Certificate type RSA Sig (5)
    4. Certificate type DSS Sig (2)
  - **Distinguished Names Length**: 159
  - **Distinguished Names (138 bytes)**
  - **Handshake Protocol**: Server Hello Done
- **Handshake Type**: Server Hello Done (14)
- **Length**: 0

### 1940
- **No.**
- **Time**
- **Source**
- **Destination**
- **Protocol Info**
- **IEEE 802.11**
- **Type/Subtype**: Data (0)
- **Frame Control**: 0x00 (Normal)

### 1950
- **Version**: 0
- **Type**: Data frame (2)
- **Subtype**: 0
- **Flags**: 81
  - **DS status**: Frame from STA to DS via an AP (To DS: 1 From DS: 0) (0x01)
  - **.Δ. = More Fragments**: This is the last fragment
  - **.Ω. = More Data**: No data buffered
  - **.Ω. = Protected flag**: Data is not protected
- **Duration**: 314
- **BSSID**: 00:12:43:8a:73:a5
- **Source address**: 3comEuro (00:12:43:8a:73:a5)
- **Destination address**: Cisco (00:12:43:8a:73:a5)
- **Fragment number**: 0
- **Sequence number**: 28

#### Logical Link Control
- **DSAP**: SNAP (80)
- **SSAP**: SNAP (80)
- **CR Bit**: Individual
- **Control field**: U:func:13 (0x05)
  - **00. 00. = Command**: Unnumbered Information (0x00)
  - **.1. = Frame type**: Unnumbered Frame (0x00)
- **Organization Code**: Encapsulated Ethernet (0x000000)

#### Type: 802.11X Authentication (0x8086)

#### 802.1X Authentication
- **Version**: 1
- **Type**: EAP Packet (0)

### 2000
- **Length**: 1496
- **Extensible Authentication Protocol**: Code: Response (2)
  - **Id**: 5
  - **Length**: 1496
- **Type**: EAP-TLS (RFC2716) [Ahobal] (12)
- **Flag (0x03)**: Length More

### 2010
- **Length**: 1580
- **TLS Record Layer**: Handshake Protocol Certificate
- **Content Type**: Handshake (22)
- **Version**: TLS 1.0 (0x30)
- **Length**: 1580
- **Certificates Length**: 1581
- **Certificates**: 1581
- **Certificate Length**: 680
Algorithm Id: 1.2.840.113549.1.9.1
issuer: rdnSequence (6)
  rdnSequence: 7 items (iso.2.5.4.11): Faigl Zoltan
  id-at:commonName=Faigl Zoltan
  id-at:organizationUnitName=MCL
  id-at:organizationName=BME–HT
  id-at:stateOrProvinceName=Budapest
  id-at:countryName=HU
Item: 1 item [id-at:countryName=HU]
  Id: 2.5.4.8 (id-at:countryName)
  CountryName: HU
2030
Item: 1 item [id-at:stateOrProvinceName=Budapest]
  Id: 2.5.4.8 (id-at:stateOrProvinceName)
  DirectoryString: printableString (1)
  printableString: Budapest
2040
Item: 1 item [id-at:localityName=Budapest]
  Id: 2.5.4.7 (id-at:localityName)
  DirectoryString: printableString (1)
  printableString: Budapest
2050
Item: 1 item [id-at:organizationName=BME–HT]
  Id: 2.5.4.10 (id-at:organizationName)
  DirectoryString: printableString (1)
  printableString: BME–HT
2060
Item: 1 item [id-at:organizationalUnitName=MCL]
  Id: 2.5.4.11 (id-at:organizationalUnitName)
  DirectoryString: printableString (1)
  printableString: MCL
2070
Item: 1 item [id-at:countryName=HU]
  Id: 2.5.4.8 (id-at:countryName)
  CountryName: HU
2080
Item: 1 item [id-at:stateOrProvinceName=Hungary]
  Id: 2.5.4.8 (id-at:stateOrProvinceName)
  DirectoryString: printableString (1)
  printableString: Hungary
2090
Item: 1 item [id-at:localityName=Budapest]
  Id: 2.5.4.7 (id-at:localityName)
  DirectoryString: printableString (1)
  printableString: Budapest
2100
Item: 1 item [id-at:organizationName=BME–HT]
  Id: 2.5.4.10 (id-at:organizationName)
  DirectoryString: printableString (1)
  printableString: BME–HT
2110
Item: 1 item [id-at:organizationalUnitName=MCL]
  Id: 2.5.4.11 (id-at:organizationalUnitName)
  DirectoryString: printableString (1)
  printableString: MCL
subjectPublicKeyInfo
  algorithm: (0xa9) 1.2.840.113549.1.1.1 (rsaEncryption)
  Padding: 0
  subjectPublicKey: 02802A33AFAF7E315969BDA4ECC84A48D3F85383E...

extension: 1
  item (id=2.5.4.7.4) (id=extKeyUsage) Extension Id: 2.5.21.97 (id=cc-extKeyUsage)
  KeyPurposeIds: 1
    Item: 1.3.6.1.5.5.7.3.1 (id-kp-clientAuth)

algorithmIdentifier (md5WithRSAEncryption)
Algorithm Id: 1.2.840.113549.1.1.4 (md5WithRSAEncryption)

Padding: 0
encrypted: 29246EDE85433757BF18DE9502C5A0558E698CB5B484...
Certificate Length: 895
Certificate: B3E492443492322C20D258E7A12078D4C44930D6D93.

issuer: rdnSequence: 7
  if (0.2.5.4.11.1) (ld=countryName=RU)
    at (1)
      id-at-countryName: Budapest
    at (2)
      id-at-countryName: HU
  at (3)
    id-at-countryName: Budapest
  at (4)
    id-at-countryName: HU
  at (5)
    id-at-countryName: Budapest
  at (6)
    id-at-countryName: Budapest
  at (7)
    id-at-countryName: Budapest

subject: rdnSequence: 7
  if (0.2.5.4.11.1) (ld=countryName=RU)
    at (1)
      id-at-countryName: Budapest
    at (2)
      id-at-countryName: HU
  at (3)
    id-at-countryName: Budapest
  at (4)
    id-at-countryName: Budapest
  at (5)
    id-at-countryName: Budapest
  at (6)
    id-at-countryName: Budapest
  at (7)
    id-at-countryName: Budapest

BER: Director for OID 1.2.840.113549.1.9.1 not implemented. Contact Ethereal developers if you want this supported
IA5String: client@domain.com

subjectPublicPkey
  algorithm: (0xa9) 1.2.840.113549.1.1.1 (rsaEncryption)
  Padding: 0
  subjectPublicKey: 02802A33AFAF7E315969BDA4ECC84A48D3F85383E...

extension: 1
  item (id=2.5.4.7.4) (id=extKeyUsage) Extension Id: 2.5.21.97 (id=cc-extKeyUsage)
  KeyPurposeIds: 1
    Item: 1.3.6.1.5.5.7.3.1 (id-kp-clientAuth)

algorithmIdentifier (md5WithRSAEncryption)
Algorithm Id: 1.2.840.113549.1.1.4 (md5WithRSAEncryption)

Padding: 0
encrypted: 29246EDE85433757BF18DE9502C5A0558E698CB5B484...
Certificate Length: 895
Certificate: B3E492443492322C20D258E7A12078D4C44930D6D93.

issuer: rdnSequence: 7
  if (0.2.5.4.11.1) (ld=countryName=RU)
    at (1)
      id-at-countryName: Budapest
    at (2)
      id-at-countryName: HU
  at (3)
    id-at-countryName: Budapest
  at (4)
    id-at-countryName: Budapest
  at (5)
    id-at-countryName: Budapest
  at (6)
    id-at-countryName: Budapest
  at (7)
    id-at-countryName: Budapest

subject: rdnSequence: 7
  if (0.2.5.4.11.1) (ld=countryName=RU)
    at (1)
      id-at-countryName: Budapest
    at (2)
      id-at-countryName: HU
  at (3)
    id-at-countryName: Budapest
  at (4)
    id-at-countryName: Budapest
  at (5)
    id-at-countryName: Budapest
  at (6)
    id-at-countryName: Budapest
  at (7)
    id-at-countryName: Budapest

BER: Director for OID 1.2.840.113549.1.9.1 not implemented. Contact Ethereal developers if you want this supported
IA5String: client@domain.com

subjectPublicPkey
  algorithm: (0xa9) 1.2.840.113549.1.1.1 (rsaEncryption)
  Padding: 0
  subjectPublicKey: 02802A33AFAF7E315969BDA4ECC84A48D3F85383E...

extension: 1
  item (id=2.5.4.7.4) (id=extKeyUsage) Extension Id: 2.5.21.97 (id=cc-extKeyUsage)
  KeyPurposeIds: 1
    Item: 1.3.6.1.5.5.7.3.1 (id-kp-clientAuth)

algorithmIdentifier (md5WithRSAEncryption)
Algorithm Id: 1.2.840.113549.1.1.4 (md5WithRSAEncryption)

Padding: 0
encrypted: 29246EDE85433757BF18DE9502C5A0558E698CB5B484...
Certificate Length: 895
Certificate: B3E492443492322C20D258E7A12078D4C44930D6D93.

issuer: rdnSequence: 7
  if (0.2.5.4.11.1) (ld=countryName=RU)
    at (1)
      id-at-countryName: Budapest
    at (2)
      id-at-countryName: HU
  at (3)
    id-at-countryName: Budapest
  at (4)
    id-at-countryName: Budapest
  at (5)
    id-at-countryName: Budapest
  at (6)
    id-at-countryName: Budapest
  at (7)
    id-at-countryName: Budapest

subject: rdnSequence: 7
  if (0.2.5.4.11.1) (ld=countryName=RU)
    at (1)
      id-at-countryName: Budapest
    at (2)
      id-at-countryName: HU
  at (3)
    id-at-countryName: Budapest
  at (4)
    id-at-countryName: Budapest
  at (5)
    id-at-countryName: Budapest
  at (6)
    id-at-countryName: Budapest
  at (7)
    id-at-countryName: Budapest

BER: Director for OID 1.2.840.113549.1.9.1 not implemented. Contact Ethereal developers if you want this supported
IA5String: client@domain.com

subjectPublicPkey
  algorithm: (0xa9) 1.2.840.113549.1.1.1 (rsaEncryption)
  Padding: 0
  subjectPublicKey: 02802A33AFAF7E315969BDA4ECC84A48D3F85383E...

extension: 1
  item (id=2.5.4.7.4) (id=extKeyUsage) Extension Id: 2.5.21.97 (id=cc-extKeyUsage)
  KeyPurposeIds: 1
    Item: 1.3.6.1.5.5.7.3.1 (id-kp-clientAuth)

algorithmIdentifier (md5WithRSAEncryption)
Algorithm Id: 1.2.840.113549.1.1.4 (md5WithRSAEncryption)

Padding: 0
encrypted: 29246EDE85433757BF18DE9502C5A0558E698CB5B484...
Certificate Length: 895
Certificate: B3E492443492322C20D258E7A12078D4C44930D6D93.
subjectPublicKeyInfo
algorithm (rsaEncryption)
Algorithm Id: 1.2.840.113549.1.1.1 (rsaEncryption)
Padding: 0
subjectPublicKey: 30818D208210996331406CFE56E2D77C9E4F33E710D...
extensions: 3 items
Item (id=ce-subjectKeyIdentifier)
  Extension Id: 23.29.14 (id=ce-subjectKeyIdentifier)
  SubjectKeyId: 78B8AB4AE54B84A4CE62B18A15D28512E9
Item (id=ce-authorityKeyIdentifier)
  Extension Id: 23.29.35 (id=ce-authorityKeyIdentifier)
  AuthorityKeyIdentifier:
    keyIdentifier: 78B8AB4AE54B84A4CE62B18A15D28512E9
    authorityCertIssuer: 1 items
      Item (id=countryName:HU)
        Item (id=countryName:HU)
          Item (id=countryName:HU)
            Item (id=countryName:HU)
              Item (id=countryName:HU)
                Item (id=countryName:HU)
                  Item (id=countryName:HU)
                    Item (id=countryName:HU)
                      Item (id=countryName:HU)
                        Item (id=countryName:HU)
                          Item (id=countryName:HU)
                            Item (id=countryName:HU)
                              Item (id=countryName:HU)
                                Item (id=countryName:HU)
                                  Item (id=countryName:HU)
                                    Item (id=countryName:HU)
                                      Item (id=countryName:HU)
                                        Item (id=countryName:HU)
                                          Item (id=countryName:HU)
                                            Item (id=countryName:HU)
                                              Item (id=countryName:HU)
                                                Item (id=countryName:HU)
                                                  Item (id=countryName:HU)
                                                    Item (id=countryName:HU)
                                                      Item (id=countryName:HU)
                                                        Item (id=countryName:HU)
                                                          Item (id=countryName:HU)
                                                            Item (id=countryName:HU)
                                                              Item (id=countryName:HU)
                                                                Item (id=countryName:HU)
                                                                  Item (id=countryName:HU)
                                                                    Item (id=countryName:HU)
                                                                      Item (id=countryName:HU)
                                                                        Item (id=countryName:HU)
                                                                          Item (id=countryName:HU)
                                                                            Item (id=countryName:HU)
                                                                             Item (id=countryName:HU)
                                                                              Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
                                                                                Item (id=countryName:HU)
Provisioning Tunable Security in IEEE 802.11i Enabled Networks

algorithmIdentifier (md5WithRSAEncryption)
Algorithm Id: 1.2.16869.3.1.2 (md5WithRSAEncryption)
Padding: 0
encrypted: 0C3D30494DDOB0EE8587BD4FA41C84...

2300
TLS Record Layer Handshake Protocol Client Key Exchange
Content Type: Handshake (22)
Version: TLS 1.0 (0x0301)
Length: 70
Handshake Protocol: Client Key Exchange
Handshake Type: Client Key Exchange (16)
Length: 86

TLS Record Layer Handshake Protocol Certificate Verify
Content Type: Handshake (22)
Version: TLS 1.0 (0x0301)
Length: 130
Handshake Protocol: Certificate Verify
Handshake Type: Certificate Verify (15)
Length: 1

TLS Record Layer Change Cipher Spec Protocol Change Cipher Spec
Content Type: Change Cipher Spec (20)
Version: TLS 1.0 (0x0301)
Length: 1

Change Cipher Spec Message
TLS Record Layer Handshake Protocol Encrypted Handshake Message
Content Type: Handshake (22)
Version: TLS 1.0 (0x0301)
Length: 48
Handshake Protocol: Encrypted Handshake Message

No. Time Source Destination Protocol Info
2330 17 071120 Cisco_aac9:b0 3comEuro_i7:73:a5 EAP Request, EAP-TLS [RFC2716] [Aboba]

Frame 17 (78 bytes on wire, 78 bytes captured)
Arrival Time: Oct 18, 2005 15:47:43.087100000
Time delta from previous packet: 0.007131000 seconds
Time since reference or first frame: 0.771210000 seconds
Frame Number: 17
Packet Length: 78 bytes
Capture Length: 78 bytes

Protocols in frame: wlanEap
eas
IEEE 802.11
Type/Subtype: Data (22)
Frame Control: 802.20 (Normal)
Version: 0
Type: Data frame (3)
Subtype: 0
Flag: 0

DS status: Frame from DS to a STA via AP(To DS: 0 From DS: 1) (0x02)
.... 0: = More Fragments: This is the last fragement
.... 0: = Retry: Frame is not being retransmitted
.... 0: = PWR MGT: STA will stay up
.... 0: = No Data: No data buffered
.... 0: = Protected flag: Data is not protected
.... 0: = Order flag: Not strictly ordered

Duration: 213
Destination address: 3comEuro_i7:73:a5 (00:0e:6a:i7:73:a5)
Source address: Cisco_aac9:b0 (00:12:43:aac9:b0)
Fragment number: 0

2340
Sequence number: 1043
Logical-Link Control
DSAP: SNAP (0xaa)
ISAP: SNAP (0xaa)

IEEE 802.11
Type/EAP Packet (0)
Length: 6
Extensible Authentication Protocol
Code: Request (1)
Id: 6
Length: 6
Type: EAP-TLS [RFC2716] [Aboba] (13)
Flag(0x0)

2370
Type: 802.1X Authentication (0x0806)
802.1X Authentication
Version: 1
Type: EAP Packet (0)
Length: 6
Extensible Authentication Protocol
Code: Request (1)
Id: 6
Length: 6
Type: EAP-TLS [RFC2716] [Aboba] (13)
Flag(0x0)

No. Time Source Destination Protocol Info
2380 18 077926 3comEuro_i7:73:a5 Cisco_aac9:b0 TLS Certificate, Client Key Exchange
Certificate Verify, Change Cipher Spec, Encrypted Handshake Message

Frame 18 (422 bytes on wire, 422 bytes captured)
| Time delta from previous packet: 0.005716000 seconds |
| Time since reference of first frame: 0.779260000 seconds |
| Frame Number: 18 |
| Packet Length: 422 bytes |
| Capture Length: 422 bytes |
| Protocol in frame [truncated]: wlan:

```
Considering that the text contains data related to an IEEE 802.11 packet, here is a detailed explanation:

**IEEE 802.11**

| Type/Subtype: Data (2) |
| Frame Control: Bsdip (Normal) |
| Version: 6 |
| Type: Data frame (2) |
| Subtype: 8 |
| Flags: 3x1 |
| DS status: Frame from STA to DS via an AP (To DS: 1 From DS: 0) (0x01) |
| .... 8...: More Fragments: This is the last fragment |
| .... 0...: Retry: Frame is not being retransmitted |
| .... 6...: PWR MGT: STA will stay up |
| .... S...: More Data: No data buffered |
| .... 8...: Protected flag: Data is not protected |
| .... 0...: Order flag: Not strictly ordered |

**Duration:** 0.005716000 seconds |

**Source address:** 00:12:43:14:90.80 |

**Destination address:** 00:12:43:14:90.80 |

**Fragment number:** 0 |

**Sequence number:** 30 |

**Logical-Link Control**

**DSAP-SNAP (0x91)**

| IG Bit: Individual |
| DSAP: SNAP (0x91) |
| CR Bit: Command |

**Control field:** U: func=UI (0x09) |

**00 02:** Command: Unnumbered Information (0x00) |

**31:** Frame type: Unnumbered frame (0x31) |

**Organization Code:** Encapsulated Ethernet (0x0001) |

**802.1X Authentication**

| Type: EAP Packet (0) |
| Length: 388 |

**EAP-TLS (RFC2716) [Alcohx] (13) |

**Flags:** (0x00) |

**EAP-TLS Fragments (100 bytes): #1(1400), #2(380) |

**Frame 16:** payload: 0-1465 (1448 bytes) |

**Frame 18:** payload: 1468-1865 (398 bytes) |

**Secure Socket Layer**

**TLS Record Layer:** Handshake Protocol: Certificate |

**Content Type:** Handshake (22) |

**Version:** TLS 1.2 (0x001) |

**Length:** 3598 |

**Handshake Protocol:** Certificate |

**Handshake Type:** Certificate (11) |

**Length:** 1594 |

**Certificates Length:** 1593 |

**Certificates (181 bytes):** |

**Certificate Length:** 65535 |

**Certificate:** 392D202DDC012D12D0261B2710B8C0464A30D0859 |

**serialNumber:** 0x007103004b444444 |

**signature (md5WithRSAEncryption):** |

**Algorithm Id:** 2.2.840.113549.1.1.11 (md5WithRSAEncryption) |

**issuer:** rdsnSequence (8) |

**rdsnSequence:** 7 items (0x2.2.840.113549.1.1.11=md5WithRSAEncryption) |

**id-at-commonName=FiGZ Zoltán, id-at-organizationName=MCI, id-at-organizationName=BME-HT, id-at-localityName=Budapest, id-at-countryName=HU** |

**Item:** 1 item (id-at-countryName=HU) |

**Id:** 2.5.4.1 (id-at-countryName) |

**CountryName:** HU |

**Item:** 1 item (id-at-stateOrProvinceName=Budapest) |

**Id:** 2.5.4.7 (id-at-stateOrProvinceName) |

**DirectoryString:** printableString (1) |

**printableString:** Budapest |

| Item:** 1 item (id-at-localityName=Budapest) |

**Id:** 2.5.4.7 (id-at-localityName) |

**DirectoryString:** printableString (1) |

**printableString:** Budapest |

**Item:** 1 item (id-at-organizationName=BME-HT) |
Providing Tunable Security in IEEE 802.11i Enabled Networks

(ISO) szlaj.2.840.113549.1.9.1 = (algorithmIdentifier
signedCertificate = (subject
subjectPublicKeyInfo
extensions
issuer
Item: 1.2.840.113549.1.9.1 (iso: 2.840.113549.1.9.1)
SER: Director for OID: 2.840.113549.1.9.1 not implemented. Contact Ethereal developers if you want this supported.
IA5String: szlaj@mcl.hu)

subject: rdnSequence (0)
rdnSequence: 7 items (iso: 2.840.113549.1.9.1: client@email.hu,
id: at: commonName: client smith
id: at: organizationalUnitName: MCL,
id: at: organizationName: BME – HT
id: at: localityName: Budapest,
id: at: stateOrProvinceName: Hungary
id: at: countryName: HU)
Item: 1 item (id: at: countryName=HU)
Id: 25.46 (id: at: countryName)
CountryName: HU
Item: 1 item (id: at: stateOrProvinceName=Hungary)
Id: 25.48 (id: at: stateOrProvinceName)
printableString: Budapest
Item: 1 item (id: at: localityName=Budapest)
Id: 25.47 (id: at: localityName)
printableString: Budapest
Item: 1 item (id: at: organizationName=client)
Id: 25.43 (id: at: organizationName)
printableString: client@email.hu

subjectPublicKeyInfo
subjectPublicKey: 0BD2B02084373370B9E29529CA06580F038CB584...
extension: 1 item
Item (id: ce=extKeyUsage)
Extension: 25.29 (id: ce=extKeyUsage)
KeyPurposeIDs: 1 item
Item: 1.3.15.3.7.2.1 (id: kp-clientAuth)

subjectPublicKeyInfo
subjectPublicKey: 0BD2B02084373370B9E29529CA06580F038CB584...
extension: 1 item
Item (id: ce=extKeyUsage)
Extension: 25.29 (id: ce=extKeyUsage)
KeyPurposeIDs: 1 item
Item: 1.3.15.3.7.2.1 (id: kp-clientAuth)

subjectPublicKeyInfo
subjectPublicKey: 0BD2B02084373370B9E29529CA06580F038CB584...
extension: 1 item
Item (id: ce=extKeyUsage)
Extension: 25.29 (id: ce=extKeyUsage)
KeyPurposeIDs: 1 item
Item: 1.3.15.3.7.2.1 (id: kp-clientAuth)
subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

76 Zoltán Faigl, Stefan Lindskog, Anna Brunstrom, and Katalin Tóth

extensions 3 items

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString

2590

DirectoryString

2580

subjectPublicKeyInfo

2650

Id

2630

2620

DirectoryString
Algorithm Id

Item (id=ce-subjectKeyIdentifier)

Extension Id: 25:29:14 (id=ce-subjectKeyIdentifier)

SubjectKeyIdentifier: 79B8A4A54EAB4AC6D0BD31AE3D342B512E9

Item (id=ce-authorityKeyIdentifier)

Extension Id: 25:29:33 (id=ce-authorityKeyIdentifier)

AuthorityKeyIdentifier:

keyIdentifier: 79B8A4A54EAB4AC6D0BD31AE3D342B512E9

authorityCertIssuer: 1 item

directoryName: rdnSequence [0]

rdnSequence: 7 items

- szlaj@nucl.hu
  - id=commonName: Faigl Zoltan
  - id=organizationName: MCL
  - id=organizationalUnitName: BME-HT
  - id=countryName: HU

Item: 1 item

id=countryName: HU

Item: 1 item

id=stateOrProvinceName: Budapest

Item: 1 item

id=localityName: Budapest

Item: 1 item

id=organizationName: BMW-HT

Item: 1 item

id=organizationalUnitName: BMW-HT

Item: 1 item

id=organizationalUnitName: MCL

Item: 1 item

id=commonName: Faigl Zoltan

Item: 1 item

id=organizationalUnitName: MCL

Item: 1 item

id=organizationalUnitName: MCL

Item: 1 item

id=localityName: Budapest

Item: 1 item

id=countryName: HU

Item: 1 item

id=stateOrProvinceName: Budapest

Item: 1 item

id=localityName: Budapest

Item: 1 item

id=organizationName: BMW-HT

Item: 1 item

id=organizationalUnitName: BMW-HT

Item: 1 item

id=organizationalUnitName: MCL

Item: 1 item

id=commonName: Faigl Zoltan

 TLS Record Layer: Handshake Protocol: Client Key Exchange

Content Type: Handshake (22)

Version: TLS 1.0 (0x03)

Length: 70

Handshake Protocol: Client Key Exchange

Handshake Type: Client Key Exchange (16)

TLS Record Layer: Handshake Protocol: Certificate Verify

Content Type: Handshake (22)

Version: TLS 1.0 (0x03)

Length: 130

Handshake Protocol: Certificate Verify

Handshake Type: Certificate Verify (15)

TLS Record Layer: Change Cipher Spec Protocol: Change Cipher Spec

Content Type: Change Cipher Spec (20)

Version: TLS 1.0 (0x03)

Length: 1

Change Cipher Spec Message

TLS Record Layer: Handshake Protocol: Encrypted Handshake Message

Content Type: Handshake (22)

Version: TLS 1.0 (0x03)

Length: 40

Handshake Protocol: Encrypted Handshake Message
<table>
<thead>
<tr>
<th>Frame 19 (125 bytes on wire, 124 bytes captured)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrival Time: Oct 18, 2005 15:47:43.140000000</td>
</tr>
<tr>
<td>Time delta from previous packet: 0.000000000 seconds</td>
</tr>
<tr>
<td>Time since reference or first frame: 0.830915000 seconds</td>
</tr>
<tr>
<td>Frame Number: 19</td>
</tr>
<tr>
<td>Packet Length: 124 bytes</td>
</tr>
<tr>
<td>Capture Length: 124 bytes</td>
</tr>
<tr>
<td>Protocol in frame: wlan</td>
</tr>
<tr>
<td>IEEE 802.11</td>
</tr>
<tr>
<td>Type/Subtype: Data (32)</td>
</tr>
<tr>
<td>Frame Control: 0x0208 (Normal)</td>
</tr>
<tr>
<td>Version: 9</td>
</tr>
<tr>
<td>Type: Data frame (2)</td>
</tr>
<tr>
<td>Subtype: 5</td>
</tr>
<tr>
<td>Flags: 0x02</td>
</tr>
<tr>
<td>DS status: Frame from DS to a STA via AP (To DS: 0 From DS: 1) (0x02)</td>
</tr>
<tr>
<td>... 0.. = More Fragments: This is the last fragment</td>
</tr>
<tr>
<td>... 0.. = Retry: Frame is not being retransmitted</td>
</tr>
<tr>
<td>... 0.. = PWR MGT: STA will stay up</td>
</tr>
<tr>
<td>... 0.. = More Data: No data buffered</td>
</tr>
<tr>
<td>... 0 = Protected flag: Data is not protected</td>
</tr>
<tr>
<td>... 0 = Order flag: Not strictly ordered</td>
</tr>
<tr>
<td>Duration: 0.830915 seconds</td>
</tr>
<tr>
<td>Destination address: 0x7f00000000000000000000000000000000000000</td>
</tr>
<tr>
<td>Source address: 0x7f00000000000000000000000000000000000000</td>
</tr>
<tr>
<td>Fragment number: 5</td>
</tr>
<tr>
<td>Sequence number: 0x0ff044</td>
</tr>
<tr>
<td>Logical-Link Control</td>
</tr>
<tr>
<td>DLAP: SNAP (0x00)</td>
</tr>
<tr>
<td>IG Bit: Individual</td>
</tr>
<tr>
<td>SNAP: SNAP (0x00)</td>
</tr>
<tr>
<td>CR Bit: Command</td>
</tr>
<tr>
<td>Control field: U: Func/UI (0x01)</td>
</tr>
<tr>
<td>00: 00: 0 = Command: Unnumbered Information (0x00)</td>
</tr>
<tr>
<td>... 11 = Frame type: Unnumbered frame (0x03)</td>
</tr>
<tr>
<td>2800 Organization Code: Encapsulated Ethernet (0x00000000)</td>
</tr>
<tr>
<td>Type: 0x0211 Authentication (0x0000)</td>
</tr>
<tr>
<td>0x0211 Authentication</td>
</tr>
<tr>
<td>Version: 1</td>
</tr>
<tr>
<td>Type: EAP Packet (0)</td>
</tr>
<tr>
<td>Length: 68</td>
</tr>
<tr>
<td>Extensible Authentication Protocol</td>
</tr>
<tr>
<td>Code: Request (1)</td>
</tr>
<tr>
<td>Id: 7</td>
</tr>
<tr>
<td>Length: 69</td>
</tr>
<tr>
<td>Type: EAP−TLS [RFC2716] [Aboba] (13)</td>
</tr>
<tr>
<td>Flags(0x00):</td>
</tr>
<tr>
<td>Length: 18</td>
</tr>
<tr>
<td>Secure Socket Layer</td>
</tr>
<tr>
<td>TLS Record Layer: Change Cipher Spec Protocol: Change Cipher Spec</td>
</tr>
<tr>
<td>Content Type: Change Cipher Spec (20)</td>
</tr>
<tr>
<td>Version: TLS 1.1 (0x0401)</td>
</tr>
<tr>
<td>Length: 18</td>
</tr>
<tr>
<td>Change Cipher Spec Message</td>
</tr>
<tr>
<td>TLS Record Layer: Handshake Protocol: Encrypted Handshake Message</td>
</tr>
<tr>
<td>2820 Content Type: Handshake (22)</td>
</tr>
<tr>
<td>Version: TLS 1.1 (0x0401)</td>
</tr>
<tr>
<td>Length: 48</td>
</tr>
<tr>
<td>Handshake Protocol: Encrypted Handshake Message</td>
</tr>
</tbody>
</table>

| Frame 20 (18 bytes on wire, 18 bytes captured) |
| Arrival Time: Oct 18, 2005 15:47:43.156805000 |
| Time delta from previous packet: 0.000000000 seconds |
| Time since reference or first frame: 0.840656000 seconds |
| Frame Number: 20 |
| Packet Length: 18 bytes |
| Capture Length: 18 bytes |
| Protocol in frame: wlan|eapi|eapool|eapol|eapol|
| IEEE 802.11 |
| Type/Subtype: Data (32) |
| Frame Control: 0x0208 (Normal) |
| Version: 9 |
| Type: Data frame (2) |
| Subtype: 5 |
| Flags: 0x01 |
| DS status: Frame from STA to DS via an AP (To DS: 1 From DS: 0) (0x01) |
| ... 0.. = More Fragments: This is the last fragment |
| ... 0.. = Retry: Frame is not being retransmitted |
| ... 0.. = PWR MGT: STA will stay up |
| ... 0.. = More Data: No data buffered |
| ... 0 = Protected flag: Data is not protected |
| ... 0 = Order flag: Not strictly ordered |
| Duration: 0.840656 seconds |
| Destination address: 0x7f00000000000000000000000000000000000000 |
| Source address: 0x7f00000000000000000000000000000000000000 |
| Fragment number: 5 |
| Sequence number: 0x0ff044 |
| Logical-Link Control |
| DLAP: SNAP (0x00) |
| IG Bit: Individual |
| SNAP: SNAP (0x00) |
| CR Bit: Command |
| Control field: U: Func/UI (0x01) |
| 00: 00: 0 = Command: Unnumbered Information (0x00) |
| ... 11 = Frame type: Unnumbered frame (0x03) |
Providing Tunable Security in IEEE 802.11i Enabled Networks

...3 ... More Data: No data buffered
...3 ... Protected flag: Data is not protected
...3 ... Order flag: Not strictly ordered
Duration: 134

BSSID: Cisc0-8ac:9:b0 (00:12:43:8a:b0)
Source address: 3comEuro_c77:3:a5 (00:0e:6a:77:3:a5)
Destination address: Cisc0-8ac:9:b0 (00:12:43:8a:b0)
Fragment number: 0
Sequence number: 31
Logical-Link Control

DSAP: SNAP (0x0)
IG Bit: Individual
SSID: SNAP (0x0)
CR Bit: Command
Control field: U, f unc=UI (0x05)
...H1: Frame type: Unnumbered frame (0x03)
Organization Code: Encapsulated Ethernet (0x888E)
Type: 802.1X Authentication (0x8880)

EAPOL Authentication

Version: 1
Type: EAP Packet (0)
Length: 4
Extensible Authentication Protocol
Code: Response (2)
Id: 7
Length: 6
Type: EAP->TLS [RFC2716] (Asha) (13)
Flags(0x0)

No. Time Source Destination Protocol Info
21 0.000061 Cisc0-8ac:9:b0 3comEuro_c77:3:a5 EAP Success

Frame 21 (78 bytes on wire, 78 bytes captured)
Arrival Time: Oct 18, 2005 15:47:43.200851000
Time delta from previous packet: 0.044305000 seconds
Time since reference or first frame: 0.044305000 seconds
Frame Number: 21

Packet Length: 78 bytes
Capture Length: 78 bytes
Protocols in frame: wlan80211, eapol

IEEE 802.11

Type/Subtype: Data (32)
Frame Control: 0x22 (Normal)
Version: 0
Type: Data frame (2)
Subtype: 0
Flags: 0x2

...DS status: Frame from DS to a STA via AP/(To DS: 0 From DS: 1) (0x02)
...: More Fragments: This is the last fragment
...: Retry: Frame is not being retransmitted
...: PWR MGT: STA will stay up
...: More Data: No data buffered
...: Protected flag: Data is not protected
...: Order flag: Not strictly ordered
Duration: 213

Destination address: 3comEuro_c77:3:a5 (00:0e:6a:77:3:a5)
BSSID: Cisc0-8ac:9:b0 (00:12:43:8a:b0)
Fragment number: 0
Sequence number: 1046
Logical-Link Control

DSAP: SNAP (0x0)
IG Bit: Individual
SSID: SNAP (0x0)
CR Bit: Command
Control field: U, f unc=UI (0x05)
...H1: Frame type: Unnumbered frame (0x03)
Organization Code: Encapsulated Ethernet (0x888E)
Type: 802.1X Authentication (0x8880)

EAPOL Authentication

Version: 1
Type: EAP Packet (0)
Length: 3
Extensible Authentication Protocol
Code: Success (3)
Id: 7
Length: 4

No. Time Source Destination Protocol Info
22 0.000552 Cisc0-8ac:9:b0 3comEuro_c77:3:a5 EAPOL Key

Frame 22 (153 bytes on wire, 153 bytes captured)
Arrival Time: Oct 18, 2005 15:47:43.205452000
Time delta from previous packet: 0.000501000 seconds
Time since reference or first frame: 0.000501000 seconds
Frame Number: 22
Packet Length: 153 bytes
Capture Length: 153 bytes
Protocol in frame: wlan80211
IEEE 802.11
Type/Subtype: Data (32)
Frame Control: 0x2028 (Normal)
Version: 0
Type: Data frame (2)
Subtype: 5
Flag: 0x2
DS status: Frame from DS to STA via AP (To DS: 0 From DS: 1) (0x02)
... 2... = More Fragments: This is the last fragment
... 6... = Retry: Frame is not being retransmitted
... 8... = PWR MGT: STA will stay up
... 5... = More Data: No data buffered
0... 0... = Protected flag: Data is not protected
0... 0... = Order flag: Not strictly ordered
Duration: 313

Destination address: 00:12:43:8a:b0:c9:03:e5
Physical address: 0x888e (IEEE 802.11B SSAP)
SSID: Cisco (00:12:43:8a:b0:c9:03:e5)
Vendor: Cisco
SSID: 802.11 Authentication: EAPOL RSN key

Key Information: 6008
... 00... = Key Descriptor Version: AES–CBC–MAC for MIC and HMAC–SHA1 for encryption (2)
... 1... = Key Type: Pairwise key
... 00... = Key Index: 0
... 0... = Install flag: Not set
... 1... = Key Ack flag: Set
... 0... = Key MIC flag: Not set
... 0... = Secure flag: Not set
... 0... = Error flag: Not set
... 0... = Request flag: Not set
... 0... = Encrypted Key Data flag: Not set

No. Time Source Destination Protocol Info
23 0.0009512 Cisco 8a:b0:c9:03:e5 EAPOL Key

Frame 23 (153 bytes on wire, 153 bytes decoded)
Arrival Time: 2005-10-16 15:47:43.221801958
Time delta from previous packet: 0.905911000 seconds
Time since reference or first frame: 0.000000000 seconds
Frame Number: 23
Packet Length: 153 bytes
Capture Length: 153 bytes
Protocol in frame: wlan80211
IEEE 802.11
Type/Subtype: Data (32)
Frame Control: 0x2028 (Normal)
Version: 0
Type: Data frame (2)
Subtype: 5
Flag: 0x2
DS status: Frame from STA to DS via an AP (To DS: 1 From DS: 0) (0x01)
... 2... = More Fragments: This is the last fragment
... 6... = Retry: Frame is not being retransmitted
... 8... = PWR MGT: STA will stay up
... 5... = More Data: No data buffered

Duration: 314
BSSID: Cisco 8a:b0:c9:03:e5 (00:12:43:8a:b0:c9:03:e5)
Frame 24 (187 bytes on wire, 187 bytes captured)
Arrival Time: Oct 18 15:47:43.223803000
Time delta from previous packet: 0.907948000 seconds
Frame Number: 24
Packet Length: 187 bytes
Capture Length: 187 bytes
Protocols in frame: wlan因地
IEEE 802.11
Type/Subtype: Data (32)
Frame Control: 802.3 (Normal)
Version: 0
Subtype: 0
Flags: 0
 DS status: Frame from DS to a STA via AP(To DS: 0 From DS: 1) (0x02)
... 0: More Fragments This is the last fragment
... 0: Retry Frame is not being retransmitted
... 0: PWR MGT STA will stay up
... 0: More Data No data buffered
... 0: Order flag Not strictly ordered
Duration: 215
Destination address: 30eE6F4773a5 (00:0e:6a:73:a5)
BSS Id: Cisco_e8a9b0 (00:12:48:e8:a9b0)
Source address: Cisco_e8a9b0 (00:12:48:e8:a9b0)
Fragment number: 0
Sequence number: 0
Logical–Link Control
Source: 82 Zoltán Faigl, Stefan Lindskog, Anna Brunstrom, and Katalin Tóth

IEEE 802.11 Authentication

Version: 2
Type: Key (3)
Length: 151

Descriptor Type: EAPOL RSN key (3)

Key Information: 0030

…… …………………………………………. = Key Descr. Version: AES−CRC−MAC for MIC and HMAC−SHA1 for encryption (2)
…… …………………………………………. = Key Type: Pairwise key
…… …………………………………………. = Key Index: 0
…… …………………………………………. = Install flag: Set
…… …………………………………………. = Key Ack flag: Set
…… …………………………………………. = Error flag: Not set
…… …………………………………………. = Request flag: Not set
…… …………………………………………. = Encrypted Key Data flag: Set

Key Length: 16

Header Counter: 2

Nonce: FF(FED1ER4AF7930EC92F26B21C7FEEAF9DC81B657C31...

Key IV: 00000000000000000000000000000000

WPA Key ID: 0000000000000000

WPA Key MIC: 28A174983E4801958D61C3F3046

WPA Key Length: 56

WPA Key: 44141454396F0D6BC09F656D831DC4E6DF0392161823C2...

IEEE 802.11

Type/Subtype: Data (32)

Frame Control: 0x0108 (Normal)

Version: 0

Type: Data frame (2)

Subtype: 0

Flags: 0x1

DS status: Frame from STA to DS via an AP (To DS: 1 From DS: 0) (0x01)

…0… = More Fragments: This is the last fragment

…0… = Retry: Frame is not being retransmitted

…0… = PWR MGT: STA will stay up

…0… = More Data: No data buffered

…0… = Protected flag: Data is not protected

…0… = Order flag: Not strictly ordered

Duration: 0x14

RSSI Id: Cisco_eac9@bc (0x1014:13:9:bc)

Source address: 3COM:GM:a7:71:a5 (0x00:0e:6a:73:c9:00)

Destination address: Cisco_eac9@bc (0x1014:13:9:bc)

Fragment number: 0

Sequence number: 33

Logical−Link Control

DSAP: SNAP (80)

IG Bit: Individual

DSAP: SNAP (80)

CR Bit: Command

Control field: U: func=UI (0x03)

…… …………………………………………. = Command: Unnumbered Information (0x00)
…… …………………………………………. = Frame type: Unnumbered frame (0x00)

Organization Code: Encapsulated Ethernet (0x000000)

Type: 802.1X Authentication (0x080e)

IEEE 802.11 Authentication

Version: 1

Type: Key (3)

Length: 151

Descriptor Type: EAPOL RSN key (3)

Key Information: 0030a

…… …………………………………………. = Key Descr. Version: AES−CRC−MAC for MIC and HMAC−SHA1 for encryption (2)
…… …………………………………………. = Key Type: Pairwise key
…… …………………………………………. = Key Index: 0
…… …………………………………………. = Install flag: Not set
…… …………………………………………. = Key Ack flag: Not set
…… …………………………………………. = Key MIC flag: Set
…… …………………………………………. = Secure flag Set

…0… = Error flag: Not set

…… …………………………………………. = Encrypted Key Data flag: Set

Key Length: 16

Key ID: 0000000000000000

WPA Key MIC: 28A174983E4801958D61C3F3046

WPA Key Length: 56

WPA Key: 44141454396F0D6BC09F656D831DC4E6DF0392161823C2...
Providing Tunable Security in IEEE 802.11i Enabled Networks

No.  Time  Source  Destination  Protocol Info
26  0.914627  3comEuro, d7:73:a5  Cisco, c9:b0  Data  Data, SN=34, FN=0

Frame 26 (116 bytes on wire, 116 bytes captured)
Arrival Time: Oct 18, 2005 15:47:43.230517000
Time delta from previous packet: 0.001085000 seconds
Time since reference or first frame: 0.914627000 seconds
Frame Number: 26
Packet Length: 116 bytes
Capture Length: 116 bytes
Protocols in frame: wlan, data
IEEE 802.11
Type/Subtype: Data (32)
Frame Control: 0x4108 (Normal)
Version: 0
Type: Data frame (2)
Subtype: 0
Flags: 0x41
DS status: Frame from STA to DS via AP (To DS: 0 From DS: 1) (0x01)
... 0. ... More Fragments: This is the last fragment
... 0. ... Retry: Frame is not being retransmitted
... 0. ... PWR MGT: STA will stay up
... 0. ... More Data: No data buffered
... 0. ... Protected flag: Data is protected
... 0. ... Order flag: Not strictly ordered
Duration: 213
BSS ID: Cisco, c9:b0 (00:12:43:8a:c9:b0)
Source address: 3comEuro, d7:73:a5 (00:0e:6a:d7:73:a5)
Destination address: Cisco, c9:b0 (00:12:43:8a:c9:b0)
Fragment number: 0
Sequence number: 34
TKIP/CCMP parameters
CCMP Ext. Initialization Vector: 0x0000000000000000
Key Index: 0

No.  Time  Source  Destination  Protocol Info
27  0.915758  Cisco, c9:b0  3comEuro, d7:73:a5  Data  Data, SN=1050, FN=0

Frame 27 (116 bytes on wire, 116 bytes captured)
Arrival Time: Oct 18, 2005 15:47:43.231648000
Time delta from previous packet: 0.001131000 seconds
Time since reference or first frame: 0.915758000 seconds
Frame Number: 27
Packet Length: 116 bytes
Capture Length: 116 bytes
Protocols in frame: wlan, data
IEEE 802.11
Type/Subtype: Data (32)
Frame Control: 0x4208 (Normal)
Version: 0
Type: Data frame (2)
Subtype: 0
Flags: 0x42
DS status: Frame from DS to a STA via AP (To DS: 0 From DS: 1) (0x02)
... 0. ... More Fragments: This is the last fragment
... 0. ... Retry: Frame is not being retransmitted
... 0. ... PWR MGT: STA will stay up
... 0. ... More Data: No data buffered
... 0. ... Protected flag: Data is protected
... 0. ... Order flag: Not strictly ordered
Duration: 213
BSS ID: Cisco, c9:b0 (00:12:43:8a:c9:b0)
Source address: 3comEuro, d7:73:a5 (00:0e:6a:d7:73:a5)
Destination address: Cisco, c9:b0 (00:12:43:8a:c9:b0)
Fragment number: 0
Sequence number: 1050
TKIP/CCMP parameters
CCMP Ext. Initialization Vector: 0x0000000000000000
Key Index: 0
Data (84 bytes)

```
0000 91 6a db 90 1e 50 e7 94 40 34 44 48 .P. . .P.. ..
0010 02 4e 21 c5 14 c8 a3 42 50 3b 26 a5 d1 57 db 09 .N!. . .BP;&. .W. .
0020 d6 70 84 b5 5a b9 13 39 26 45 a6 a3 24 66 91: be .p. .Z. .9+E. .-. .
0030 02 fa bf 25 a7 d1 00 53 8f 1b 07 40 f4 16 7b c3 .X..S...{.
0040 97 8e 91 3b a8 1a 42 31 27 47 3a 77 53 62 37 7b .17U:qs{.
0050 cf 3d 5a be <Z.
```
D Detailed Description of Considered Security Configurations

In this appendix, detailed descriptions of the considered ciphersuites (see Table 2 in Section 6) with respect to message and computational complexity are presented.
D.1 TLS_DHE_DSS_with_*_*

TLS_DHE_DSS_with_*_* is the most secure ciphersuite considered in this report. It uses DHE key exchange and the DSS algorithm for signing the server certificate. A client may either use a client certificate signed by the DSS algorithm (1a) (notation used in Table 2 in Section 6) or the RSA algorithm (1b). Note that 1a is more secure than 1b.

D.1.1 Message Complexity

Figure 11 and Figure 12 present the message flow and the data frame lengths in bytes, transmitted on the 802.11 radio channel, for the TLS_DHE_DSS_with_*_* ciphersuite. The numbers at the message names refer to the TLS message types, and their meaning can be found, e.g., in Table 10. The numbers after the message names are the size of the frames in bytes.

D.1.2 Computational Complexity

A detailed description of the cryptographic computations performed in relation to each TLS message is given in this section.

The length of the applied constants and messages in bytes

- Client_random: 32
- Server_random: 32
- Premaster secret: \(g^{xy} mod p\) where the length of \(p\) is \(128–1024\) bytes. 128 bytes long \(p\) is used in the calculation.
- Master secret: 48
  1. SHA-1(113+80) → 20
  2. SHA-1(114+80) → 20
  3. SHA-1(115+80) → 20
  4. MD5(68+80) → 16
  5. MD5(68+80) → 16
  6. MD5(68+80) → 16
  7. 4∥5∥6 → 48
- Handshake1: 4555 (1a) / 4170 (1b)
  - ClientHello: 50
  - ServerHello: 79
  - Certificate: 1942
  - ServerKeyExchange: 274
  - CertificateRequest: 158
Figure 11: Message flow of the SA establishment with the TLS_DHE_DSS_with_*_* security configuration.
Figure 12: Message flow of the SA establishment with the TLS_DHE_DSS_with_**_** security configuration. (Continuation from Figure 11.)

- ServerHelloDone: 4
- Certificate: 1970 (1a) / 1588 (1b)
- ClientKeyExchange: 75

- Handshake2: 4590 (1a) / 4221 (1b)
  - Handshake1: 4555 (1a) / 4170 (1b)
  - CertificateVerify: 29 (1a) / 45 (1b)
  - ChangeCipherSpec: 6

- Handshake3: 4649 (1a) / 4280 (1b)
  - Handshake2: 4590 (1a) / 4221 (1b)
Providing Tunable Security in IEEE 802.11i Enabled Networks

- Finished (Client → Server): 53
- ChangeCipherSpec: 6

- Master_secret_left, Master_secret_right: 24

- Label: CLIENT FINISHED or SERVER FINISHED: 15

- Seed1: 36
  1. MD5(handshake2) = MD5(4590 (1a) / 4221 (1b)) → 16
  2. SHA-1(handshake2) = SHA-1(4590 (1a) / 4221 (1b)) → 20
  3. 1∥2 → 36

- Seed2: 36
  1. MD5(handshake3) = MD5(4649 (1a) / 4280 (1b)) → 16
  2. SHA-1(handshake3) = SHA-1(4649 (1a) / 4280 (1b)) → 20
  3. 1∥2 → 36

- Length of certificates:
  - ServerCert: 856
  - RootCACert of the server: 1086
  - RootCACert of the client: 1086 (1a) / 895 (1b)
  - ClientCert: 871 (1a) / 680 (1b)

**Computations to generate and check the TLS handshake messages** The numbers express the sizes in bytes of the data that are either the input or the output of the specific algorithms.

- **ServerKeyExchange** Generate p, g and x and calculate \((g^x \mod p)\) DH public value of the server.

- **Certificate messages** The client checks the certificate of the server and the certificate of the root CA after the reception of the certificate handshake message from the server.

  (1a)
  - MD5(ServCert)=MD5(856)
  - DSS-1(20)
  - MD5(RootCACert of the server)=MD5(1086)
  - DSS-1(20)

  (1b)
  - MD5(ServCert)=MD5(856)
  - DSS-1(20)
  - MD5(RootCACert of the server)=MD5(1086)
• DSS-1(20)

The server checks the certificate of the client and the certificate of the root CA after the reception of the certificate message from the client.

(1a) • SHA-1(ClientCert)=SHA-1(871)
• DSS-1(20)
• SHA-1(RootCACert of the client)=SHA-1(1086)
• DSS-1(20)

(1b) • MD5(ClientCert)=MD5(680)
• RSA-1(16)
• MD5(RootCACert of the client)=MD5(895)
• RSA-1(16)

ClientKeyExchange The client generates its one-time DH parameters:

• p
• g (group value that must be the same as the server uses)
• y
• the public DH-value (g^y mod p)

The client sends the public DH-value to the server. Furthermore, it calculates the pre-master secret with the Diffie-Hellman key derivation algorithm.

CertificateVerify Signature from the client:

(1a) • SHA-1(Handshake1) = SHA-1(4555) → 20
• DSS(20) → 36

(1b) • MD5(Handshake1) = MD5(4170) → 16
• SHA-1(Handshake1) = SHA-1(4170) → 20
• RSA(16 || 20) = RSA(36) → 36

Signature verification at server:

(1a) • SHA-1(Handshake1) = SHA-1(4555) → 20
• DSS-1(20)

(1b) • MD5(Handshake1) = MD5(4170) → 16
• SHA-1(Handshake1) = SHA-1(4170) → 20
• RSA-1(16 || 20) = RSA-1(36)
Providing Tunable Security in IEEE 802.11i Enabled Networks

**Finished from client to server**  The client calculates the finished message and the server checks it, as a consequence the following calculation is made twice, once at each side.

- **Seed1 generation (2 steps)**
  - $A_1 = \text{MD5}(\text{master\_secret\_left} + \text{label} \parallel \text{seed1}) = \text{MD5}(75) \rightarrow 16$
  - $P_{\text{MD5}}(\text{master\_secret\_left} + \text{label} \parallel \text{seed1}) = \text{MD5}(\text{master\_secret\_left} + \text{label} \parallel A_1 \parallel \text{seed1}) = \text{MD5}(91) \rightarrow 16$
  - $A_1 = \text{SHA-1}(\text{master\_secret\_right} + \text{label} \parallel \text{seed1}) = \text{SHA-1}(75) \rightarrow 20$
  - $P_{\text{SHA-1}}(\text{master\_secret\_left} + \text{label} \parallel \text{seed1}) = \text{SHA-1}(\text{master\_secret\_right} + \text{label} \parallel A_1 \parallel \text{seed1}) = \text{SHA-1}(95) \rightarrow 20$
  - $P_{\text{MD5}}(\text{master\_secret\_left} + \text{label} \parallel \text{seed1}) + \text{mod} P_{\text{SHA-1}}(\text{master\_secret\_left} + \text{label} \parallel \text{seed1})$. (This step is not computationally complex, it is simply the XOR of the two variables.)

**Finished from server to client**  The server calculates the finished message and sends it to the client. Client checks it, as a consequence the following calculation is made twice, once at each side.

- **Seed2 generation (2 steps)**
  - $A_1 = \text{MD5}(\text{master\_secret\_left} + \text{label} \parallel \text{seed2}) = \text{MD5}(75) \rightarrow 16$
  - $P_{\text{MD5}}(\text{master\_secret\_left} + \text{label} \parallel \text{seed2}) = \text{MD5}(\text{master\_secret\_left} + \text{label} \parallel A_1 \parallel \text{seed2}) = \text{MD5}(91) \rightarrow 16$
  - $A_1 = \text{SHA-1}(\text{master\_secret\_right} + \text{label} \parallel \text{seed2}) = \text{SHA-1}(75) \rightarrow 20$
  - $P_{\text{SHA-1}}(\text{master\_secret\_left} + \text{label} \parallel \text{seed2}) = \text{SHA-1}(\text{master\_secret\_right} + \text{label} \parallel A_1 \parallel \text{seed2}) = \text{SHA-1}(95) \rightarrow 20$
  - $P_{\text{MD5}}(\text{master\_secret\_left} + \text{label} \parallel \text{seed2}) + \text{mod} P_{\text{SHA-1}}(\text{master\_secret\_left} + \text{label} \parallel \text{seed2})$. (This step is not computationally complex, since XOR is used.)
D.2 TLS_DHE_RSA_with_**

TLS_DHE_RSA_with_** is the second most secure ciphersuite considered. It also uses DHE key exchange, but with the RSA algorithm for signing the server certificate instead of DSS. Again a client may either use a client certificate signed by DSS (2a) (notation used in Table 2 in Section 6) or RSA (2b), and 2a is stronger with respect to security than 2b.

D.2.1 Message Complexity

Figure 13 and Figure 14 present the message flow and the data frame lengths in bytes, transmitted on the 802.11 radio channel, for the TLS_DHE_RSA_with_** ciphersuite. The numbers at the message names refer to the TLS message types, and their meaning can be found, e.g., in Table 10. The numbers after the message names are the size of the frames in bytes.

D.2.2 Computational Complexity

A detailed description of the cryptographic computations performed in relation to each TLS message is given in this section.

The length of the applied constants and messages in bytes

- Client_random: 32
- Server_random: 32
- Premaster secret: $g^{xy} \mod p$ where the length of $p$ is $\sim 128–1024$ bytes. 128 bytes long $p$ is used in the calculation.
- Master secret: 48
  1. SHA-1(113+80) $\rightarrow$ 20
  2. SHA-1(114+80) $\rightarrow$ 20
  3. SHA-1(115+80) $\rightarrow$ 20
  4. MD5(68+80) $\rightarrow$ 16
  5. MD5(68+80) $\rightarrow$ 16
  6. MD5(68+80) $\rightarrow$ 16
  7. $4∥5∥6$ $\rightarrow$ 48
- Handshake1: 4191 (2a) / 3806 (2b)
  - ClientHello: 50
  - ServerHello: 79
  - Certificate: 1578
  - ServerKeyExchange: 274
  - CertificateRequest: 158
Figure 13: Message flow of the SA establishment with the TLS,DHE,RSA with * * security configuration.
94 Zoltán Faigl, Stefan Lindskog, Anna Brunstrom, and Katalin Tóth

STA AP AS 7, 8, 9, 10, 11 7, 8, 9, 10, 11
Access Challenge

EAP Request: 78

7, 8, 9, 10, 11: 693 (2a) / 327 (2b)

12, 13: 105

EAP TLS Response: 78

Access Request

EAP Success: 78

Access Accept: E{PMK}

Key1: 153

Key2: 153

Key3: 187

Key4: 131

Data session

Figure 14: Message flow of the SA establishment with the TLS_DHE_RSA_with_*_* security configuration. (Continuation from Figure 13.)

- ServerHelloDone: 4
- Certificate: 1970 (2a) / 1588 (2b)
- ClientKeyExchange: 75

- Handshake2: 4226 (2a) / 3857 (2b)
- Handshake1: 4191 (2a) / 3806 (2b)
- CertificateVerify: 29 (2a) / 45 (2b)
- ChangeCipherSpec: 6
• Handshake3: 4285 (2a) / 3916 (2b)
  - Handshake2: 4226 (2a) / 3857 (2b)
  - Finished (Client → Server): 53
  - ChangeCipherSpec: 6
• Master_secret_left, Master_secret_right: 24
• Label: CLIENT FINISHED or SERVER FINISHED: 15
• Seed1: 36
  1. MD5(handshake2) = MD5(4226 (2a) / 3857 (2b)) → 16
  2. SHA-1(handshake2) = SHA-1(4226 (2a) / 3857 (2b)) → 20
  3. 1||2 → 36
• Seed2: 36
  1. MD5(handshake3) = MD5(4285 (2a) / 3916 (2b)) → 16
  2. SHA-1(handshake3) = SHA-1(4285 (2a) / 3916 (2b)) → 20
  3. 1||2 → 36
• Length of certificates:
  - ServerCert: 665
  - RootCACert of the server: 895
  - RootCACert of the client: 1086 (2a) / 895 (2b)
  - ClientCert: 871 (2a) / 680 (2b)

**Computations to generate and check the TLS handshake messages**  The numbers express the sizes in bytes of the data that are either the input or the output of the specific algorithms.

**ServerKeyExchange**  Generate p, g and x and calculate \((g^x \mod p)\) DH public value of the server.

**Certificate messages**  The client checks the certificate of the server and the certificate of the root CA after the reception of the certificate handshake message from the server.

(2a)
- MD5(ServCert)=MD5(665)
- RSA-1(16)
- MD5(RootCACert of the server)=MD5(895)
- RSA-1(16)

(2b)
- MD5(ServCert)=MD5(665)
The server checks the certificate of the client and the certificate of the root CA after the reception of the certificate message from the client.

(2a)  
- SHA-1(ClientCert)=SHA-1(871)  
- DSS-1(20)  
- SHA-1(RootCACert of the client)=SHA-1(1086)  
- DSS-1(20)

(2b)  
- MD5(ClientCert)=MD5(680)  
- RSA-1(16)  
- MD5(RootCACert of the client)=MD5(895)  
- RSA-1(16)

**ClientKeyExchange**  The client generates its one-time DH parameters:

- **p**
- **g** (group value that must be the same as the server uses)
- **y**
- the public DH-value \((g^y \mod p)\)

The client sends the public DH-value to the server. Furthermore, it calculates the pre-master secret with the Diffie-Hellman key derivation algorithm.

**CertificateVerify**  Signature from the client:

(2a)  
- SHA-1(Handshake1) = SHA-1(4191 (2a)) → 20  
- DSS-1( 20 ) → 36

(2b)  
- MD5(Handshake1) = MD5(3806 (2b)) → 16  
- SHA-1(Handshake1) = SHA-1(3806 (2b)) → 20  
- RSA(16 || 20) = RSA(36) → 36

Signature verification at server:

(2a)  
- SHA-1(Handshake1) = SHA-1(4191 (2a)) → 20  
- DSS-1( 20 )

(2b)  
- MD5(Handshake1) = MD5(3806 (2b)) → 16  
- SHA-1(Handshake1) = SHA-1(3806 (2b)) → 20  
- RSA-1(16 || 20) = RSA-1(36)
Finished from client to server  The client calculates the finished message and the server checks it, as a consequence the following calculation is made twice, once at each side.

- **Seed1 generation (2 steps)**
  - \( A_1 = \text{MD5(master\_secret\_left + label} \parallel \text{seed1)} = \text{MD5(75)} \rightarrow 16 \)
  - \( P_{\text{MD5(master\_secret\_left + label} \parallel \text{seed1)} = \text{MD5(master\_secret\_left + label} \parallel A_1 \parallel \text{seed1)} = \text{MD5(91)} \rightarrow 16 \)
  - \( A_1 = \text{SHA-1(master\_secret\_right + label} \parallel \text{seed1)} = \text{SHA-1(75)} \rightarrow 20 \)
  - \( P_{\text{SHA-1(master\_secret\_left + label} \parallel \text{seed1)} = \text{SHA-1(master\_secret\_right + label} \parallel A_1 \parallel \text{seed1)} = \text{SHA-1(95)} \rightarrow 20 \)
  - \( P_{\text{MD5(master\_secret\_left + label} \parallel \text{seed1}} + \text{mod} P_{\text{SHA-1(master\_secret\_left + label} \parallel \text{seed1)}} \). (This step is not computationally complex, it is simply the XOR of the two variables.)

Finished from server to client  The server calculates the finished message and sends it to the client. Client checks it, as a consequence the following calculation is made twice, once at each side.

- **Seed2 generation (2 steps)**
  - \( A_1 = \text{MD5(master\_secret\_left + label} \parallel \text{seed2)} = \text{MD5(75)} \rightarrow 16 \)
  - \( P_{\text{MD5(master\_secret\_left + label} \parallel \text{seed2)} = \text{MD5(master\_secret\_left + label} \parallel A_1 \parallel \text{seed2)} = \text{MD5(91)} \rightarrow 16 \)
  - \( A_1 = \text{SHA-1(master\_secret\_right + label} \parallel \text{seed2)} = \text{SHA-1(75)} \rightarrow 20 \)
  - \( P_{\text{SHA-1(master\_secret\_left + label} \parallel \text{seed2)} = \text{SHA-1(master\_secret\_right + label} \parallel A_1 \parallel \text{seed2)} = \text{SHA-1(95)} \rightarrow 20 \)
  - \( P_{\text{MD5(master\_secret\_left + label} \parallel \text{seed2}} + \text{mod} P_{\text{SHA-1(master\_secret\_left + label} \parallel \text{seed2})} \). (This step is not computationally complex, since XOR is used.)
D.3 TLS_RSA_with_*_*

The third considered ciphersuite is TLS_RSA_with_*_*, which uses RSA for both key exchange and signing of server certificates. The client may, however, use either DSS (3a) or RSA (3b) for signing. 3a is more secure than 3b.

D.3.1 Message Complexity

Figure 15 and Figure 16 present the message flow and the data frame lengths in bytes, transmitted on the 802.11 radio channel, for the TLS_RSA_with_*_* ciphersuite. The numbers at the message names refer to the TLS message types, and their meaning can be found, e.g., in Table 10. The numbers after the message names are the size of the frames in bytes.

D.3.2 Computational Complexity

A detailed description of the cryptographic computations performed in relation to each TLS message is given in this section.

The length of the applied constants and messages in bytes

- Client_random: 32
- Server_random: 32
- Premaster secret: 48 (client generates)
- Master secret: 48
  
  1. SHA-1(113) $\rightarrow$ 20
  2. SHA-1(114) $\rightarrow$ 20
  3. SHA-1(115) $\rightarrow$ 20
  4. MD5(68) $\rightarrow$ 16
  5. MD5(68) $\rightarrow$ 16
  6. MD5(68) $\rightarrow$ 16
  7. $4\parallel 5\parallel 6$ $\rightarrow$ 48

- Handshake1: 4186 (3a) / 3804 (3b)
  - ClientHello: 50
  - ServerHello: 79
  - Certificate: 1578
  - ServerKeyExchange: 301
  - CertificateRequest: 156
  - ServerHelloDone: 4
  - Certificate: 1970 (3a) / 1588 (3b)
Figure 15: Message flow of the SA establishment with the TLS_RSA_with_*_* security configuration.
Figure 16: Message flow of the SA establishment with the TLS_RSA_with_\*\_\* security configuration. (Continuation from Figure 15.)

- Client Key Exchange: 48
- Handshake2: 4221 (3a) / 3855 (3b)
  - Handshake1: 4186 (3a) / 3804 (3b)
  - CertificateVerify: 29 (3a) / 45 (3b)
  - ChangeCipherSpec: 6
- Handshake3: 4280 (3a) / 3914 (3b)
  - Handshake2: 4221 (3a) / 3855 (3b)
  - Finished (Client → Server): 53
  - ChangeCipherSpec: 6

- Master_secret_left, Master_secret_right: 24

- Label: CLIENT FINISHED or SERVER FINISHED: 15

- Seed1: 36
  1. MD5(handshake2) = MD5(4221 (3a) / 3855 (3b) ) → 16
  2. SHA-1(handshake2) = SHA-1(4221 (3a) / 3855 (3b) ) → 20
  3. 1∥2 → 36

- Seed2: 36
  1. MD5(handshake3) = MD5(4280 (3a) / 3914 (3b)) → 16
  2. SHA-1(handshake3) = SHA-1(4280 (3a) / 3914 (3b)) → 20
  3. 1∥2 → 36

- Length of certificates:
  - ServerCert: 665
  - RootCACert of the server: 895
  - RootCACert of the client: 1086 (3a) / 895 (3b)
  - ClientCert: 871 (3a) / 680 (3b)

**Computations to generate and check the TLS handshake messages**  The numbers express the sizes in bytes of the data that are either the input or the output of the specific algorithms.

**ServerKeyExchange**  In the ServerKeyExchange message, an exponent and modulus are sent from the server serving as a public RSA encryption key, signed with RSA by the server using MD5 and SHA-1 hash algorithms.

- MD5(Client_Random || Server_Random || exponent || modulus) = MD5(32+32+128+128) → 16
- SHA-1(Client_Random || Server_Random || exponent || modulus) = SHA-1(32+32+128+128) → 20
- RSA(16 || 20) = RSA (36) → 36 (sign)
Certificate messages  There are two certificate messages sent during the TLS handshake protocol, one from the server and another from the client.

The client checks the certificate of the server and the certificate of the root CA after the reception of the certificate handshake message from the server. Since the server uses RSA signed certificates, the computations at the client side are the same for case (3a) and (3b).

(3a)  
- MD5(ServCert)=MD5(665)
- RSA-1(16)
- MD5(RootCACert of the server)=MD5(895)
- RSA-1(16)

(3b)  
- MD5(ServCert)=MD5(665)
- RSA-1(16)
- MD5(RootCACert of the server)=MD5(895)
- RSA-1(16)

The server checks the certificate of the client and the certificate of the root CA after the reception of the certificate message from the client.

(3a)  
- SHA-1(ClientCert)=SHA-1(871)
- DSS-1(20)
- SHA-1(RootCACert of the client)=SHA-1(1086)
- DSS-1(20)

(3b)  
- MD5(ClientCert)=MD5(680)
- RSA-1(16)
- MD5(RootCACert of the client)=MD5(895)
- RSA-1(16)

ClientKeyExchange  The client generates a 48 bytes long pre-master secret and encrypts it with RSA encryption using the RSA public key of the server (from the ServerKeyExchange):

- RSA encryption: 48

Then the server side needs to decrypt the pre-master secret:

- RSA-1 decryption: 48
Certificate Verify  Signature from the client:

(3a)  • SHA-1(Handshake1) = SHA-1(4186) → 20
      • DSS(20) → 36
(3b)  • MD5(Handshake1) = MD5(3804) → 16
      • SHA-1(Handshake1) = SHA-1(3804) → 20
      • RSA(16 || 20) = RSA(36) → 36

Signature verification at the server:

(3a)  • SHA-1(Handshake1) = SHA-1(4187) → 20
      • DSS-1(20)
(3b)  • MD5(Handshake1) = MD5(3804) → 16
      • SHA-1(Handshake1) = SHA-1(3804) → 20
      • RSA-1(16 || 20) = RSA-1(36)

Finished from client to server  The client calculates the finished message and the server checks it, as a consequence the following calculation is made twice, once at each side.

• Seed1 generation (2 steps)
  • A1 = MD5(master_secret_left + label || seed1) = MD5(75) → 16
  • P_MD5(master_secret_left + label || seed1) = MD5(master_secret_left + label || A1 || seed1) = MD5(91) → 16
  • A1 = SHA-1(master_secret_right + label || seed1) = SHA-1(75) → 20
  • P_SHA-1(master_secret_left + label || seed1) = SHA-1(master_secret_right + label || A1 || seed1) = SHA-1(95) → 20
  • P_MD5(master_secret_left + label || seed1) + mod P_SHA-1(master_secret_left + label || seed1). (This step is not computationally complex, it is simply the XOR of the two variables.)

Finished from server to client  The server calculates the finished message and sends it to the client. Client checks it, as a consequence the following calculation is made twice, once at each side.

• Seed2 generation (2 steps)
  • A1 = MD5(master_secret_left + label || seed2) = MD5(75) → 16
  • P_MD5(master_secret_left + label || seed2) = MD5(master_secret_left + label || A1 || seed2) = MD5(91) → 16
  • A1 = SHA-1(master_secret_right + label || seed2) = SHA-1(75) → 20
• $P_{SHA-1}(master\_secret\_left + label \parallel seed2) = SHA-1(master\_secret\_right + label \parallel A1 \parallel seed2) = SHA-1(95) \rightarrow 20$

• $P_{MD5}(master\_secret\_left + label \parallel seed2) + mod P_{SHA-1}(master\_secret\_left + label \parallel seed2)$. (This step is not computationally complex, since XOR is used.)
D.4 TLS_DH_DSS_with_*_*

This ciphersuite uses fixed DH for key exchange and DSS signed server certificates. A client can use either DSS (4a) or RSA (4b) signed certificates, where 4a is more secure than 4b.

D.4.1 Message Complexity

Figure 17 and Figure 18 present the message flow and the data frame lengths in bytes, transmitted on the 802.11 radio channel, for the TLS_DH_DSS_with_*_* ciphersuite. The numbers at the message names refer to the TLS message types, and their meaning can be found, e.g., in Table 10. The numbers after the message names are the size of the frames in bytes.

D.4.2 Computational Complexity

A detailed description of the cryptographic computations performed in relation to each TLS message is given in this section.

The length of the applied constants and messages in bytes

- Client_random: 32
- Server_random: 32
- Premaster secret: $g^{xy} \mod p$ where the length of $p$ is $128–1024$ bytes. 128 bytes long $p$ is used in the calculation.
- Master secret: 48
  1. SHA-1(113+80) → 20
  2. SHA-1(114+80) → 20
  3. SHA-1(115+80) → 20
  4. MD5(68+80) → 16
  5. MD5(68+80) → 16
  6. MD5(68+80) → 16
  7. 4∥5∥6 → 48
- Handshake1: 4061 (4a) / 3870 (4b)
  - ClientHello: 50
  - ServerHello: 79
  - Certificate: 1957 (4a) / 1957 (4a)
  - ServerHelloDone: 9
  - Certificate: 1957 (4a) / 1766(4b)
  - ClientKeyExchange: 9
Figure 17: Message flow of the SA establishment with the TLS_DH_DSS_with_*_* security configuration.
Figure 18: Message flow of the SA establishment with the TLS_DH_DSS_with_*_* security configuration. (Continuation from Figure 17.)

- Handshake2: 4067 (4a) / 3876(4b)
  - Handshake1: 4061 (4a) / 3870 (4b)
  - ChangeCipherSpec: 6
- Handshake3: 4126 (4a) / 3935 (4b)
  - Handshake2: 4067 (4a) / 3876(4b)
  - Finished (Client → Server): 53
  - ChangeCipherSpec: 6

- Master_secret_left, Master_secret_right: 24
• Label: CLIENT FINISHED or SERVER FINISHED: 15

• Seed1: 36

  1. MD5(handshake2) = MD5(4067 (4a) / 3876(4b)) → 16
  2. SHA-1(handshake2) = SHA-1(4067 (4a) / 3876(4b)) → 20
  3. 1∥2 → 36

• Seed2: 36

  1. MD5(handshake3) = MD5(4126 (4a) / 3935 (4b)) → 16
  2. SHA-1(handshake3) = SHA-1(4126 (4a) / 3935 (4b)) → 20
  3. 1∥2 → 36

• Length of certificates:
  – Fixed-DH certificate of the server: 871 (similar to DSS signing certificate)
  – Signing certificate of the root CA of the server: 1086
  – Fixed-DH certificate of the client: 871
  – Signing certificate of the root CA of the client: 1086 (4a) / 895 (4b)

Computations to generate and check the TLS handshake messages
The numbers express the sizes in bytes of the data that are either the input or the output of the specific algorithms.

Certificate messages
The server and the client exchange their fixed DH certificates and the signing certificate of the root CA who has signed the fixed DH certificate. They have to check the certificates of the other party. The client checks the certificate of the server and the certificate of the root CA after the reception of the certificate handshake message from the server.

(4a) • SHA1(DHServCert)=SHA1(871)
     • DSS-1(20)
     • SHA1(RootCACert of the server)=SHA1(1086)
     • DSS-1(20)

(4b) • MD5(DHServCert)=MD5(871)
     • DSS-1(20)
     • MD5(RootCACert of the server)=MD5(1086)
     • DSS-1(20)

The server checks the certificate of the client and the certificate of the root CA after the reception of the certificate message from the client.

(4a) • SHA-1(DHClientCert)=SHA-1(871)
• DSS-1(20)
• SHA-1(RootCACert of the client)=SHA-1(1086)
• DSS-1(20)

\[ \text{(4b)} \]
• MD5(DHClientCert)=MD5(871)
• RSA-1(16)
• MD5(RootCACert of the client)=MD5(895)
• RSA-1(16)

**Finished from client to server** The client calculates the finished message and the server checks it, as a consequence the following calculation is made twice, once at each side.

• Seed1 generation (2 steps)
  • \( A_1 = \text{MD5(master_secret_left} + \text{label} \parallel \text{seed1}) = \text{MD5(75)} \rightarrow 16 \)
  • \( P_{\text{MD5}}(\text{master_secret_left} + \text{label} \parallel \text{seed1}) = \text{MD5(master_secret_left} + \text{label} \parallel A_1 \parallel \text{seed1}) = \text{MD5(91)} \rightarrow 16 \)
  • \( A_1 = \text{SHA-1(master_secret_right} + \text{label} \parallel \text{seed1}) = \text{SHA-1(75)} \rightarrow 20 \)
  • \( P_{\text{SHA-1}}(\text{master_secret_left} + \text{label} \parallel \text{seed1}) = \text{SHA-1(master_secret_right} + \text{label} \parallel A_1 \parallel \text{seed1}) = \text{SHA-1(95)} \rightarrow 20 \)
  • \( P_{\text{MD5}}(\text{master_secret_left} + \text{label} \parallel \text{seed1}) + \text{mod} P_{\text{SHA-1}}(\text{master_secret_left} + \text{label} \parallel \text{seed1}). \) (This step is not computationally complex, it is simply the XOR of the two variables.)

**Finished from server to client** The server calculates the finished message and sends it to the client. Client checks it, as a consequence the following calculation is made twice, once at each side.

• Seed2 generation (2 steps)
  • \( A_1 = \text{MD5(master_secret_left} + \text{label} \parallel \text{seed2}) = \text{MD5(75)} \rightarrow 16 \)
  • \( P_{\text{MD5}}(\text{master_secret_left} + \text{label} \parallel \text{seed2}) = \text{MD5(master_secret_left} + \text{label} \parallel A_1 \parallel \text{seed2}) = \text{MD5(91)} \rightarrow 16 \)
  • \( A_1 = \text{SHA-1(master_secret_right} + \text{label} \parallel \text{seed2}) = \text{SHA-1(75)} \rightarrow 20 \)
  • \( P_{\text{SHA-1}}(\text{master_secret_left} + \text{label} \parallel \text{seed2}) = \text{SHA-1(master_secret_right} + \text{label} \parallel A_1 \parallel \text{seed2}) = \text{SHA-1(95)} \rightarrow 20 \)
  • \( P_{\text{MD5}}(\text{master_secret_left} + \text{label} \parallel \text{seed2}) + \text{mod} P_{\text{SHA-1}}(\text{master_secret_left} + \text{label} \parallel \text{seed2}). \) (This step is not computationally complex, since XOR is used.)
D.5 TLS_DH_RSA_with_*_*

TLS_DH_RSA_with_*_* is the least secure ciphersuite considered in this report. It uses fixed DH for key exchange and RSA signed server certificates. A client can use either DSS (5a) or RSA (5b) signed certificates, where 5a is more secure than 5b.

D.5.1 Message Complexity

Figure 19 and Figure 20 present the message flow and the data frame lengths in bytes, transmitted on the 802.11 radio channel, for the TLS_DH_DSS_with_*_* ciphersuite. The numbers at the message names refer to the TLS message types, and their meaning can be found, e.g., in Table 10. The numbers after the message names are the size of the frames in bytes.

D.5.2 Computational Complexity

A detailed description of the cryptographic computations performed in relation to each TLS message is given in this section.

The length of the applied constants and messages in bytes

- Client_random: 32
- Server_random: 32
- Premaster secret: $g^xy \mod p$ where the length of $p$ is $\sim 128$–1024 bytes. 128 bytes long $p$ is used in the calculation.
- Master secret: 48
  1. SHA-1(113+80) → 20
  2. SHA-1(114+80) → 20
  3. SHA-1(115+80) → 20
  4. MD5(68+80) → 16
  5. MD5(68+80) → 16
  6. MD5(68+80) → 16
  7. 4∥5∥6 → 48
- Handshake1: 3870 (5a) / 3679 (5b)
  - ClientHello: 50
  - ServerHello: 79
  - Certificate: 1766 (5a) / 1766(5a)
  - ServerHelloDone: 9
  - Certificate: 1957 (5a) / 1766(5b)
  - ClientKeyExchange: 9
Figure 19: Message flow of the SA establishment with the TLS_DH_RSA_with_*. security configuration.
Figure 20: Message flow of the SA establishment with the TLS_DH_RSA_with_#_#_#_# security configuration. (Continuation from Figure 19.)

- Handshake2: 3876 (5a) / 3685(5b)
  - Handshake1: 3870 (5a) / 3679 (5b)
  - ChangeCipherSpec: 6
- Handshake3: 3935 (5a) / 3744 (5b)
  - Handshake2: 3876 (5a) / 3685 (5b)
  - Finished (Client → Server): 53
  - ChangeCipherSpec: 6
- Master_secret_left, Master_secret_right: 24
- Label: CLIENT FINISHED or SERVER FINISHED: 15
- Seed1: 36
  1. MD5(handshake2) = MD5(3876 (5a) / 3685(5b)) → 16
2. SHA-1(handshake2) = SHA-1(3876 (5a) / 3685(5b) ) → 20
3. 1 || 2 → 36

- Seed2: 36
  1. MD5(handshake3) = MD5(3935 (5a) / 3744 (5b)) → 16
  2. SHA-1(handshake3) = SHA-1(3935 (5a) / 3744 (5b)) → 20
  3. 1 || 2 → 36

- Length of certificates:
  - Fixed-DH certificate of the server: 871 (similar to DSS signing certificate)
  - Signing certificate of the root CA of the server: 895
  - Fixed-DH certificate of the client: 871
  - Signing certificate of the root CA: 1086 (5a) / 895 (5b)

**Computations to generate and check the TLS handshake messages**

**Certificate messages** The server and the client exchange their fixed DH certificates and the signing certificate of the root CA who has signed the fixed DH certificate. They have to check the certificates of the other party. The client checks the certificate of the server and the certificate of the root CA after the reception of the certificate handshake message from the server.

(5a)
- MD5(DHServCert) = MD5(871)
  - RSA-1(16)
  - MD5(RootCACert of the server) = MD5(895)
  - RSA-1(16)

(5b)
- MD5(DHServCert) = MD5(871)
  - RSA-1(16)
  - MD5(RootCACert of the server) = MD5(895)
  - RSA-1(16)

The server checks the certificate of the client and the certificate of the root CA after the reception of the certificate message from the client.

(5a)
- SHA-1(DHClientCert) = SHA-1(871)
  - DSS-1(20)
  - SHA-1(RootCACert of the client) = SHA-1(1086)
  - DSS-1(20)

(5b)
- MD5(DHClientCert) = MD5(871)
  - RSA-1(16)
  - MD5(RootCACert of the client) = MD5(895)
  - RSA-1(16)
Finished from client to server  The client calculates the finished message and the server checks it, as a consequence the following calculation is made twice, once at each side.

- Seed1 generation (2 steps)
  
  - $A_1 = \text{MD5(master}\_\text{secret}\_\text{left} + \text{label} \parallel \text{seed1}) = \text{MD5(75)} \rightarrow 16$
  
  - $P\_\text{MD5(master}\_\text{secret}\_\text{left} + \text{label} \parallel \text{A1} \parallel \text{seed1}) = \text{MD5(91)} \rightarrow 16$
  
- $A_1 = \text{SHA-1(master}\_\text{secret}\_\text{right} + \text{label} \parallel \text{seed1}) = \text{SHA-1(75)} \rightarrow 20$

- $P\_\text{SHA-1(master}\_\text{secret}\_\text{right} + \text{label} \parallel A_1 \parallel \text{seed1}) = \text{SHA-1(95)} \rightarrow 20$

- $P\_\text{MD5(master}\_\text{secret}\_\text{left} + \text{label} \parallel \text{seed1}) + \text{mod P\_SHA-1(master}\_\text{secret}\_\text{left} + \text{label} \parallel \text{seed1})$. (This step is not computationally complex, it is simply the XOR of the two variables.)

Finished from server to client  The server calculates the finished message and sends it to the client. Client checks it, as a consequence the following calculation is made twice, once at each side.

- Seed2 generation (2 steps)

  - $A_1 = \text{MD5(master}\_\text{secret}\_\text{left} + \text{label} \parallel \text{seed2}) = \text{MD5(75)} \rightarrow 16$

  - $P\_\text{MD5(master}\_\text{secret}\_\text{left} + \text{label} \parallel \text{A1} \parallel \text{seed2}) = \text{MD5(91)} \rightarrow 16$

- $A_1 = \text{SHA-1(master}\_\text{secret}\_\text{right} + \text{label} \parallel \text{seed2}) = \text{SHA-1(75)} \rightarrow 20$

- $P\_\text{SHA-1(master}\_\text{secret}\_\text{right} + \text{label} \parallel A_1 \parallel \text{seed2}) = \text{SHA-1(95)} \rightarrow 20$

- $P\_\text{MD5(master}\_\text{secret}\_\text{left} + \text{label} \parallel \text{seed2}) + \text{mod P\_SHA-1(master}\_\text{secret}\_\text{left} + \text{label} \parallel \text{seed2})$. (This step is not computationally complex, since XOR is used.)
E Results of the graphical analysis of MAC latency in ROME
E.1 Graphical Results of Experiment 1

The aim of the first experiment was to analyze the main output parameters based on a very simple traffic source model. In one simulation run, the stations send data frames with constant length and with exponential inter arrival time. One run is determined by three input parameters: the number of stations in the network, and the traffic source model, i.e., the mean of the exponential inter arrival time of the data units in the MAC layer of the stations and the value of the constant data frame length. The inter arrival time and the number of stations are responsible for the "density" of the packet transmission, and the data frame length value influences the length of the channel occupation.

The following groups of plots show how the alteration of the input parameters influence the mean and the variance of the output parameters (axis Z). In most of the plots that will be presented, axis X and Y show the mean of the inter arrival time and the data length, and each plot represent different fixed value for the number of stations in the network. As a consequence, one column of a histogram represent one given run of the simulation. In each group of plots, the first histogram is a specific one, because it aggregates all of the runs of the experiment, which means, that it is an average of all of the simulation runs with different number of stations. The first histogram below always shows the general tendency of the dependency of the output parameter on the mean inter arrival time and the data frame length, for any number of stations in the network.

To simplify the description of the plots, we use a notation of $Z \sim X, Y, V$ in the captions of the figures, which means that the given histogram shows the dependency of the mean of the Z value as a function of the parameters X, Y, and V. On the left histogram we always use a logarithmic scale, while on the right histogram, we use a linear scale. Moreover, the right histogram shows also the variance of the Z values, by displaying not a simple point but a vertical line, which length is showing the variance of Z value at that point of X, Y and V. Note, that in Experiment 1 the variance of Z is very small thus we typically just see points on the right diagram. In this case, the two diagrams differ only in the facts that the left shows the mean Z value on a logarithmic scale and the right one shows it the mean value of Z and its near zero variance on a linear scale.

In the caption of the figures, the mean of the inter arrival times of the data units with exponential distribution is abbreviated with iaTime, the data frame length is noted by dfLength, the number of stations is abbreviated with ns. In all cases, one station acts as the receiver of all the other stations, and it does not transmit data. As a consequence, e.g., $ns = 2$ means a situation where only one station utilizes the channel, in case of $ns = 20$, 19 stations transmit frames.

Dependency of the MAC layer latency on the inter arrival time, the data frame length and the number of stations. The following group of histograms try to give response for the basic question of how the MAC layer latency depends of the traffic intensity. By observing the plots, we can see that the MAC layer latency shows a linear dependency on the data frame length in the order of magnitude of $10^{-4} - 10^{-3}$. In case of few stations (2, 4) the transmission
is whatever intensive, the MAC layer latency shows only a dependency on the
data frame length. Note, that the dips in the plot mean that we did not have
measurements in that domain. With the increase of the number of stations, we
can see that below a certain inter arrival time (0.002 seconds), the MAC layer
latency increases to the order of $10^{-2} - 1$. The more the number of stations is,
the less data length values cause this increase. When the MAC layer latency is
in the order of $10^{-2} - 1$, the dependency on the data frame length is negligible.
The variance of the MAC layer latency is near zero because every station uses the
constant data length, as described in Table 15. This is also true for higher values
of $ns$. 
Figure 21: MAC layer latency $\sim iaTime, dfLength$ (aggregated "number of stations").

Figure 22: MAC layer latency $\sim iaTime, dfLength, ns = 2$.

Figure 23: MAC layer latency $\sim iaTime, dfLength, ns = 4$. 
Figure 24: MAC layer latency $\sim iaTime, dfLength, ns = 6$.

Figure 25: MAC layer latency $\sim iaTime, dfLength, ns = 8$.

Figure 26: MAC layer latency $\sim iaTime, dfLength, ns = 10$. 
Figure 27: MAC layer latency $\sim iaTime, dfLength, ns = 12.$

Figure 28: MAC layer latency $\sim iaTime, dfLength, ns = 14.$

Figure 29: MAC layer latency $\sim iaTime, dfLength, ns = 16.$
Figure 30: MAC layer latency $\sim iaTime, dfLength, ns = 18$.

Figure 31: MAC layer latency $\sim iaTime, dfLength, ns = 20$. 
Dependency of the channel utilization on the inter arrival time, the data frame length, and the number of stations. The following group of histograms shows the dependency of the channel utilization on the basic input parameters. The channel utilization is the percent of time when the channel is in busy or collided state in a given simulation run. The first histogram aggregates the results of all the experiments with different number of stations. It shows a general average and variance of the channel utilization on the inter arrival time and data frame length. In case of the other histograms, each column represents one given run. The histograms show very well, that the channel utilization increases suddenly, with a phase shifting, as the intensity of the traffic grows. Moreover, the utilization can never be greater than 90%.

![Histograms showing channel utilization dependency](image)

Figure 32: Channel utilization ~ iaTime, dfLength (aggregated “number of stations”.)
Providing Tunable Security in IEEE 802.11i Enabled Networks

Figure 33: Channel utilization $\sim iaTime, dfLength, ns = 2$.  

Figure 34: Channel utilization $\sim iaTime, dfLength, ns = 4$.  

Figure 35: Channel utilization $\sim iaTime, dfLength, ns = 6$.  

Figure 36: Channel utilization $\sim \text{iaTime}, df\text{Length}, ns = 8$.

Figure 37: Channel utilization $\sim \text{iaTime}, df\text{Length}, ns = 10$.

Figure 38: Channel utilization $\sim \text{iaTime}, df\text{Length}, ns = 12$. 
Figure 39: Channel utilization $\sim iaTime, dfLength, ns = 14$.

Figure 40: Channel utilization $\sim iaTime, dfLength, ns = 16$.

Figure 41: Channel utilization $\sim iaTime, dfLength, ns = 18$. 
Figure 42: Channel utilization $\sim iaTime, dfLength, ns = 20$. 
Dependency of the block length on the inter arrival time, the data frame length, and the number of stations. The following group of histograms shows the dependency of the block length in function of the three input parameters. The block length measures how much time the service takes to successfully send the data frame on the channel. The block length is lesser than the MAC layer latency by the waiting time in the transmission queue of the station and the first time interval until the transmission of the first RTS frame for that data unit\(^9\). The results show that the block length has the same tendency as the MAC layer latency. In cases where the traffic is normal, i.e., not intensive, the block length plus a constant value describes well the MAC layer latency. This probably means that there is an almost empty waiting queue with constant length in the transmitters. The first histogram is an aggregation of the results of the remaining histograms.

\[\text{Mean blocklength [sec]}\]

\[\text{Variance of blocklength [sec]}\]

Figure 43: block length \(\sim\) iaTime, dfLength (aggregated "number of stations").

\(^9\)In the future, it will be practical to also include the interval until the first sending of RTS in the definition of the block length.
Figure 44: block length $\sim iaTime, dfLength, ns = 2$. 

Figure 45: block length $\sim iaTime, dfLength, ns = 4$. 

Figure 46: block length $\sim iaTime, dfLength, ns = 6$. 
Figure 47: block length $\sim iaTime, dfLength, ns = 8$.

Figure 48: block length $\sim iaTime, dfLength, ns = 10$.

Figure 49: block length $\sim iaTime, dfLength, ns = 12$. 

mean interarrival time [sec] 0 0.05 0.1 0.15 0.2 0.25 0.3 0.35 0.4 0.45 0.5

data frame length [bit] 0 2000 4000 6000 8000 10000 12000

Variance of blocklength [sec] 0 0.002 0.004 0.006 0.008 0.01 0.012 0.014 0.016 0.018 0.02 0.022
<table>
<thead>
<tr>
<th>mean interarrival time [sec]</th>
<th>data frame length [bit]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.05</td>
<td>2000</td>
</tr>
<tr>
<td>0.1</td>
<td>4000</td>
</tr>
<tr>
<td>0.15</td>
<td>6000</td>
</tr>
<tr>
<td>0.2</td>
<td>8000</td>
</tr>
<tr>
<td>0.25</td>
<td>10000</td>
</tr>
<tr>
<td>0.3</td>
<td>12000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mean blocklength [sec]</th>
<th>Variance of blocklength [sec]</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0.002</td>
</tr>
<tr>
<td>-3</td>
<td>0.004</td>
</tr>
<tr>
<td>10</td>
<td>0.006</td>
</tr>
<tr>
<td>-2</td>
<td>0.008</td>
</tr>
<tr>
<td>10</td>
<td>0.01</td>
</tr>
<tr>
<td>-2</td>
<td>0.012</td>
</tr>
<tr>
<td>10</td>
<td>0.014</td>
</tr>
<tr>
<td>-2</td>
<td>0.016</td>
</tr>
<tr>
<td>10</td>
<td>0.018</td>
</tr>
<tr>
<td>-2</td>
<td>0.02</td>
</tr>
<tr>
<td>10</td>
<td>0.022</td>
</tr>
</tbody>
</table>

Figure 50: block length $\sim \text{iaTime, dfLength, ns} = 14$.  

Figure 51: block length $\sim \text{iaTime, dfLength, ns} = 16$.  

Figure 52: block length $\sim \text{iaTime, dfLength, ns} = 18$.  
Figure 53: block length $\sim$ iaTime, dfLength, ns = 20.
Dependency of the MAC layer latency on the channel utilization, the data frame length, and the number of stations. In the following subsection, the results of experiment 1 are analyzed from another perspective, notably how the output parameters, in this case the MAC layer latency, depend on the data frame length and another output parameter, which is easy to measure at the side of the stations, i.e., the channel utilization. The channel utilization is expressed by the percentage of time when the channel is in idle state and abbreviated by chIdle in the caption of the figures. In this case, it is not true that one column of a histogram represent one given run of the simulation. The histograms of this group show that few stations can not reach more than a certain level of channel utilization, even if they transmit intensively. The tendencies of the plots show how the value of the data frame length (the same constant value for all of the stations) cause higher channel utilizations, and how the number of participating stations influences the level of channel utilization. The plots also reflect the weakness of the first experiment, since we do not know what would happen with short frames in high channel utilization, because high channel utilization was reached only when we applied longer data frames. This highlights the aim of the second experiment (see Appendix E.2), where we made independent the channel utilization from the data frame length by separating the behavior of the observed station from the other stations, and letting it send in every simulation run data frames with any length.

Figure 54: MAC layer latency $\sim \text{chIdle, dfLength}$ (aggregated "number of stations").
Figure 55: MAC layer latency $\sim \text{chIdle, dfLength, ns} = 2$.

Figure 56: MAC layer latency $\sim \text{chIdle, dfLength, ns} = 4$.

Figure 57: MAC layer latency $\sim \text{chIdle, dfLength, ns} = 6$. 
Figure 58: MAC layer latency $\sim \text{chIdle, dfLength}, ns = 8$.

Figure 59: MAC layer latency $\sim \text{chIdle, dfLength}, ns = 10$.

Figure 60: MAC layer latency $\sim \text{chIdle, dfLength}, ns = 12$. 
Figure 61: MAC layer latency $\sim ch\text{Idle}, df\text{Length, } ns = 14.$

Figure 62: MAC layer latency $\sim ch\text{Idle}, df\text{Length, } ns = 16.$

Figure 63: MAC layer latency $\sim ch\text{Idle}, df\text{Length, } ns = 18.$
Figure 64: MAC layer latency $\sim chIdle, dfLength, ns = 20$. 
Dependency of the block length on the channel utilization, the data frame length, and the number of stations. The following group of histograms show the dependency of the block length on the channel utilization and the data frame length. The conclusions are very similar to those of the previous group of histograms, with the difference that now we analyze the block length instead of the MAC layer latency. The histograms show which channel utilizations could be reached in experiment 1 by the different runs with given constant data frame length settings, and also present the block length for the different cases. This result again highlights the need for a new experiment (see the results of experiment 2 in Appendix E.2) which separates the channel utilization from the data frame length.

Figure 65: block length $\sim ch\text{Idle}, df\text{Length}$ (aggregated "number of stations").
Figure 66: block length $\sim chIdle, dfLength, ns = 2$.

Figure 67: block length $\sim chIdle, dfLength, ns = 4$.

Figure 68: block length $\sim chIdle, dfLength, ns = 6$. 

\[ \text{channel idle \[%\]} \]
\[ \text{data frame length [bit]} \]
\[ \text{Mean blocklength [sec]} \]
\[ \text{Variance blocklength [sec]} \]
Figure 69: block length \( \sim ch\text{Idle}, df\text{Length}, ns = 8 \).

Figure 70: block length \( \sim ch\text{Idle}, df\text{Length}, ns = 10 \).

Figure 71: block length \( \sim ch\text{Idle}, df\text{Length}, ns = 12 \).
Figure 72: block length $\sim chIdle, dfLength, ns = 14$.

Figure 73: block length $\sim chIdle, dfLength, ns = 16$.

Figure 74: block length $\sim chIdle, dfLength, ns = 18$. 
Figure 75: block length $\sim chIdle, dfLength, ns = 20$. 
E.2 Graphical Results of Experiment 2

In the second experiment, we were curious about the dependency of the MAC layer latency, and other output parameters mainly in function of the channel utilization and data frame length. The following groups of histograms present the different dependencies of the measurement data of the second experiment. In experiment 2, the columns of histograms are not representing a given set of simulation runs as was the case in experiment 1. This is explained by the fact that the same channel utilization value can be reached by several configurations of the simulation. Generally, most of the runs were causing moderate and high network load, i.e., the most of the results are filled in the histograms under moderate or high channel load values. In this experiment we have one observed station, which transmits in every run data frames with uniformly distributed length within $[256, 12258]$ bits. The background traffic was generated by the remaining stations. The traffic source model applied for the background stations was similar to that of experiment 1, i.e., generation of constant length data frames with exponential inter arrival time in a given run. In this Appendix, several dependencies are analyzed graphically with a group of histograms. In one group, the first histogram is always an aggregated result of all of the runs of the experiment. Then every three histogram forms a subset of the experiment, where a given number of stations was used. For example the first three histograms present the case where two stations were in the network, the next three plots show the case where four stations constituted the network, etc. The number of stations is noted in the caption of the figures with ns. (The plots having the same ns are drawn on the same page.) It is important to note, that one station acts always as a receiver of all the traffic, so it doesn’t generate traffic. Another station is the observed station with distinct behavior. The remainders serve as the background traffic generators. It can be seen, that when $ns = 2$, then all the traffic is generated by the observed station because there are no background stations. With the increase of the number of stations, the background stations constitute a more and more dominating group, having an important effect on the whole traffic and load of the network. The three histograms on one page, having the same ns value, differs in the traffic source configuration of the observed station. The mean of the inter arrival time of the data units is 0.05 seconds for the first, 0.03 seconds for the second and 0.01 seconds for the third histogram on each page. The distribution function of the inter arrival time is exponential in all three cases. The axis “data frame length [bit]” always show the data frame lengths of the observed station, which is uniformly distributed in all runs of the experiment, and, as a consequence, the channel utilization is relatively independent of this value.

To simplify the description of the plots, we use the notation of $Z \sim X, Y, V, W$ in the captions of the figures, meaning that the given histogram represents the dependency of the mean value of $Z$ at the left side and the variance of the $Z$ value at the right side in function of the parameters $X, Y, V,$ and $W$, where $X$ and $Y$ are the two dimensions of the histograms. $X$ and $Y$ are usually the data frame length of the observed station, abbreviated with dfLength, the percentage of time when the channel is in idle state, noted as chIdle. $V$ and $W$ are always the number of stations, ns, and the inter arrival time of the observed station, noted with iaTime.
Dependency of the MAC layer latency on the channel utilization and the data frame length. The following group of histograms shows the dependency of the MAC layer latency on the channel utilization and the data frame length. Channel utilization is expressed with its negative, the channel in idle state. The graphical results show that in the course of an increasing channel load there is a phase shifting where the channel utilization suddenly increases, leaving out a small interval. In this case, the MAC layer latency also increases by one order of magnitude at least. The mathematical description of this effect should be analyzed more deeply in the future.

Figure 76: MAC layer latency $\sim chIdle, dfLength$ (aggregated for all ns and iaTime values).
Figure 77: MAC layer latency $\sim chIdle, dfLength, ns = 2, iaTime = 0.05$.

Figure 78: MAC layer latency $\sim chIdle, dfLength, ns = 2, iaTime = 0.03$.

Figure 79: MAC layer latency $\sim chIdle, dfLength, ns = 2, iaTime = 0.01$. 
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>-4</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-3</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-2</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>3</td>
</tr>
</tbody>
</table>

Figure 80: MAC layer latency $\sim chIdle, dfLength, ns = 4, iaTime = 0.05$.

Figure 81: MAC layer latency $\sim chIdle, dfLength, ns = 4, iaTime = 0.03$.

Figure 82: MAC layer latency $\sim chIdle, dfLength, ns = 4, iaTime = 0.01$. 
Figure 83: MAC layer latency $\sim$ chIdle, dfLength, ns = 8, iaTime = 0.05.

Figure 84: MAC layer latency $\sim$ chIdle, dfLength, ns = 8, iaTime = 0.03.

Figure 85: MAC layer latency $\sim$ chIdle, dfLength, ns = 8, iaTime = 0.01.
Figure 86: MAC layer latency $\sim$ chIdle, dfLength, ns = 12, iaTime = 0.05.

Figure 87: MAC layer latency $\sim$ chIdle, dfLength, ns = 12, iaTime = 0.03.

Figure 88: MAC layer latency $\sim$ chIdle, dfLength, ns = 12, iaTime = 0.01.
Figure 89: MAC layer latency $\sim chIdle, dfLength, ns = 16, iaTime = 0.05$.

Figure 90: MAC layer latency $\sim chIdle, dfLength, ns = 16, iaTime = 0.03$.

Figure 91: MAC layer latency $\sim chIdle, dfLength, ns = 16, iaTime = 0.01$. 
Figure 92: MAC layer latency $\sim ch Idle, df Length, ns = 20, ia Time = 0.05$.

Figure 93: MAC layer latency $\sim ch Idle, df Length, ns = 20, ia Time = 0.03$.

Figure 94: MAC layer latency $\sim ch Idle, df Length, ns = 20, ia Time = 0.01$. 
Dependency of the block length on the channel utilization and the data frame length. The following group of histograms show the dependency of the block length on the channel utilization and data frame length. The same statements can be said for the block length as for the MAC layer latency. At lower channel utilizations a linear dependency of the block length can be seen in function of the data frame length. As the channel load increases a phase shifting can be reached, where the channel load increases suddenly. This can be caused by the fact that the collisions and the repeated transmissions occupies the channel and the effective throughput decreases suddenly. When is this phase shifting reached? This is a question that should be analyzed in the future.

Figure 95: block length $\sim$ chIdle, dfLength (aggregated for all ns and iaTime values).
Figure 96: block length \( \sim ch\text{Idle}, df\text{Length}, ns = 2, ia\text{Time} = 0.05 \).

Figure 97: block length \( \sim ch\text{Idle}, df\text{Length}, ns = 2, ia\text{Time} = 0.03 \).

Figure 98: block length \( \sim ch\text{Idle}, df\text{Length}, ns = 2, ia\text{Time} = 0.01 \).
Figure 99: block length $\sim ch\text{Idle}, df\text{Length}, ns = 4, ia\text{Time} = 0.05$.

Figure 100: block length $\sim ch\text{Idle}, df\text{Length}, ns = 4, ia\text{Time} = 0.03$.

Figure 101: block length $\sim ch\text{Idle}, df\text{Length}, ns = 4, ia\text{Time} = 0.01$. 
Figure 102: block length $\sim chIdle, dfLength, ns = 8, iaTime = 0.05$.

Figure 103: block length $\sim chIdle, dfLength, ns = 8, iaTime = 0.03$.

Figure 104: block length $\sim chIdle, dfLength, ns = 8, iaTime = 0.01$. 
<table>
<thead>
<tr>
<th>Mean blocklength [sec]</th>
<th>Variance of blocklength [sec]</th>
</tr>
</thead>
<tbody>
<tr>
<td>-5 10</td>
<td>0</td>
</tr>
<tr>
<td>-4 10</td>
<td>0.005</td>
</tr>
<tr>
<td>-3 10</td>
<td>0.01</td>
</tr>
<tr>
<td>-2 10</td>
<td>0.015</td>
</tr>
<tr>
<td>-1 10</td>
<td>0.02</td>
</tr>
<tr>
<td>0 10</td>
<td>0.025</td>
</tr>
<tr>
<td>1 10</td>
<td>0.03</td>
</tr>
<tr>
<td>2 10</td>
<td>0.035</td>
</tr>
<tr>
<td>3 10</td>
<td>0.04</td>
</tr>
<tr>
<td>4 10</td>
<td>0.045</td>
</tr>
<tr>
<td>5 10</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Figure 105: block length $\sim chIdle, dfLength, ns = 12, iaTime = 0.05$.

Figure 106: block length $\sim chIdle, dfLength, ns = 12, iaTime = 0.03$.

Figure 107: block length $\sim chIdle, dfLength, ns = 12, iaTime = 0.01$. 
Figure 108: block length $\sim chIdle, dfLength, ns = 16, iaTime = 0.05$.

Figure 109: block length $\sim chIdle, dfLength, ns = 16, iaTime = 0.03$.

Figure 110: block length $\sim chIdle, dfLength, ns = 16, iaTime = 0.01$. 


Figure 111: block length $\sim chIdle, dfLength, ns = 20, iaTime = 0.05$.

Figure 112: block length $\sim chIdle, dfLength, ns = 20, iaTime = 0.03$.

Figure 113: block length $\sim chIdle, dfLength, ns = 20, iaTime = 0.01$. 
Dependency of the number of (re)sends of RTS frames on the channel utilization and the data frame length. The next group of histograms show the dependency between the number of (re)transmissions (called (re)sends in the following) of RTS frames for the same data frame (noted by resendRTS), as a function of the channel utilization and the data frame length. It can be seen that there is no considerable dependence between resendRTS and the data frame length. The number of (re)sends depends strongly on the channel utilization. The mean of resendRTS increases suddenly and its variance becomes considerable after a phase shifting in the channel utilization.

Figure 114: ResendRTS ∼ chIdle, dfLength (aggregated for all ns and iaTime values).
Figure 115: ResendRTS \(\sim\) chIdle, dfLength, ns = 2, iaTime = 0.05.

Figure 116: ResendRTS \(\sim\) chIdle, dfLength, ns = 2, iaTime = 0.03.

Figure 117: ResendRTS \(\sim\) chIdle, dfLength, ns = 2, iaTime = 0.01.
Figure 118: ResendRTS $\sim chIdle, dfLength, ns = 4, iaTime = 0.05$.

Figure 119: ResendRTS $\sim chIdle, dfLength, ns = 4, iaTime = 0.03$.

Figure 120: ResendRTS $\sim chIdle, dfLength, ns = 4, iaTime = 0.01$. 
Figure 121: ResendRTS $\sim$ chIdle, dfLength, ns = 8, iaTime = 0.05.

Figure 122: ResendRTS $\sim$ chIdle, dfLength, ns = 8, iaTime = 0.03.

Figure 123: ResendRTS $\sim$ chIdle, dfLength, ns = 8, iaTime = 0.01.
Figure 124: ResendRTS $\sim$ $chIdle, dfLength, ns = 12, iaTime = 0.05$.

Figure 125: ResendRTS $\sim$ $chIdle, dfLength, ns = 12, iaTime = 0.03$.

Figure 126: ResendRTS $\sim$ $chIdle, dfLength, ns = 12, iaTime = 0.01$. 
Figure 127: ResendRTS $\sim$ chIdle, dfLength, $ns = 16, iaTime = 0.05$.

Figure 128: ResendRTS $\sim$ chIdle, dfLength, $ns = 16, iaTime = 0.03$.

Figure 129: ResendRTS $\sim$ chIdle, dfLength, $ns = 16, iaTime = 0.01$. 
Figure 130: $\text{ResendRTS} \sim \text{chIdle}, \text{dfLength}, ns = 20, \text{iaTime} = 0.05$.

Figure 131: $\text{ResendRTS} \sim \text{chIdle}, \text{dfLength}, ns = 20, \text{iaTime} = 0.03$.

Figure 132: $\text{ResendRTS} \sim \text{chIdle}, \text{dfLength}, ns = 20, \text{iaTime} = 0.01$. 
Dependency of the block length on the number of (re)sends of RTS frames and the data frame length. The following groups of histograms analyze dependencies which are thought to contain some linearity. In this group we analyzed, what is the dependency between the block length and the data length and the number of (re)sends of RTS frames. A block consists of several trials to reserve the channel with RTS frames, which can be sent only if the backoff counter of the transmitter reaches zero. There is the risk, that the transmitted RTS frame will collide with other frames on the channel which cause the retransmission of a next RTS frame. The time interval between the intent to send an RTS frame, and its realization may not be a linear factor of the block length, but the number of retrials to send RTS frames is a linear increasing factor in the block length. The data frame length influences the block length because after several trials of sending RTS frames, when a CTS frame is successfully received from the other station, the transmitter sends the data frame, which length influences naturally the block length. The histograms show that when there are at least one retrial of RTS sending, the dependency of the block length becomes infinitesimal from the data frame length. In case of one RTS sending, the block length is a linear function of the data frame length.

Figure 133: block length $\sim$ resend$_{RTS}$.dfLength (aggregated for all ns and iaTime values).
Figure 134: block length $\sim \text{resendRTS, dfLength, ns} = 2, \text{iaTime} = 0.05$.

Figure 135: block length $\sim \text{resendRTS, dfLength, ns} = 2, \text{iaTime} = 0.03$.

Figure 136: block length $\sim \text{resendRTS, dfLength, ns} = 2, \text{iaTime} = 0.01$. 
Figure 137: block length $\sim$ resendRTS, dfLength, ns = 4, iaTime = 0.05.

Figure 138: block length $\sim$ resendRTS, dfLength, ns = 4, iaTime = 0.03.

Figure 139: block length $\sim$ resendRTS, dfLength, ns = 4, iaTime = 0.01.
Figure 140: block length \(\sim resendRTS, dfLength, ns = 8, iaTime = 0.05.\)

Figure 141: block length \(\sim resendRTS, dfLength, ns = 8, iaTime = 0.03.\)

Figure 142: block length \(\sim resendRTS, dfLength, ns = 8, iaTime = 0.01.\)
Figure 143: block length $\sim$ resendRTS, dfLength, ns = 12, iaTime = 0.05.

Figure 144: block length $\sim$ resendRTS, dfLength, ns = 12, iaTime = 0.03.

Figure 145: block length $\sim$ resendRTS, dfLength, ns = 12, iaTime = 0.01.
Figure 146: block length \( \sim \) resend\(RTS\), \(dfLength, ns = 16, iaTime = 0.05\).

Figure 147: block length \( \sim \) resend\(RTS\), \(dfLength, ns = 16, iaTime = 0.03\).

Figure 148: block length \( \sim \) resend\(RTS\), \(dfLength, ns = 16, iaTime = 0.01\).
Figure 149: block length $\sim$ resendRTS, dfLength, ns = 20, iaTime = 0.05.

Figure 150: block length $\sim$ resendRTS, dfLength, ns = 20, iaTime = 0.03.

Figure 151: block length $\sim$ resendRTS, dfLength, ns = 20, iaTime = 0.01.
Dependency of the MAC layer latency on the waiting queue length and the number of (re)sends of RTS frames. The waiting queue length is measured when the data unit arrives in the MAC layer from the upper layer. Finally, this group of histograms analyzes another dependency thought to have some linearity inside. These histograms show the dependency of the MAC layer latency on the waiting queue length (noted as qLength in the captions of the figures)—measured at the moment of data unit arrival into the MAC layer from the upper layer of the transmitter—and the number of (re)sends of RTS frame(s) for the same data frame (noted as resendRTS). The first three plots present the aggregated results in the same histogram with different zoom-level and projection. The effect of the waiting queue length seems to be very important for the total MAC layer latency, if the waiting queue exists. The waiting queue length showed an increasing tendency in many cases, even if it was not so fast. The histograms show that state of run where the endSimulation() function was called, i.e., the observed station sent 1000 data frames successfully or its waiting queue reached a length of 1500 packets. When a waiting queue exists, it can be a linear increasing factor for the MAC layer latency, but the variation of MAC layer latency is very high, i.e., can be measured in seconds.
Figure 152: MAC layer latency $\sim q\text{Length, resendRTS}$ (aggregated for all ns and iaTime values).

Figure 153: MAC layer latency $\sim q\text{Length, resendRTS}$ (aggregated for all ns and iaTime values).

Figure 154: MAC layer latency $\sim q\text{Length, resendRTS}$ (aggregated for all ns and iaTime values). Projection on qLength.
Figure 155: MAC layer latency $\sim q\text{Length}, resendRTS, ns = 2, iaTime = 0.05$.

Figure 156: MAC layer latency $\sim q\text{Length}, resendRTS, ns = 2, iaTime = 0.03$.

Figure 157: MAC layer latency $\sim q\text{Length}, resendRTS, ns = 2, iaTime = 0.01$. 
Figure 158: MAC layer latency $\sim q\text{Length}, \text{resendRTS}, ns = 4, \text{iaTime} = 0.05$.

Figure 159: MAC layer latency $\sim q\text{Length}, \text{resendRTS}, ns = 4, \text{iaTime} = 0.03$.

Figure 160: MAC layer latency $\sim q\text{Length}, \text{resendRTS}, ns = 4, \text{iaTime} = 0.01$. 
Figure 161: MAC layer latency $\sim qLength, resendRTS, ns = 8, iaTime = 0.05$.

Figure 162: MAC layer latency $\sim qLength, resendRTS, ns = 8, iaTime = 0.03$.

Figure 163: MAC layer latency $\sim qLength, resendRTS, ns = 8, iaTime = 0.01$. 
Figure 164: MAC layer latency $\sim q\text{Length}, \text{resendRTS}, ns = 12, ia\text{Time} = 0.05$.

Figure 165: MAC layer latency $\sim q\text{Length}, \text{resendRTS}, ns = 12, ia\text{Time} = 0.03$.

Figure 166: MAC layer latency $\sim q\text{Length}, \text{resendRTS}, ns = 12, ia\text{Time} = 0.01$. 
Figure 167: MAC layer latency $\sim q_{Length}, resendRTS, ns = 16, iaTime = 0.05$.

Figure 168: MAC layer latency $\sim q_{Length}, resendRTS, ns = 16, iaTime = 0.03$.

Figure 169: MAC layer latency $\sim q_{Length}, resendRTS, ns = 16, iaTime = 0.01$. 
Figure 170: MAC layer latency $\sim q\text{Length}, \text{resendRTS}, \text{ns} = 20, \text{iaTime} = 0.05$.

Figure 171: MAC layer latency $\sim q\text{Length}, \text{resendRTS}, \text{ns} = 20, \text{iaTime} = 0.03$.

Figure 172: MAC layer latency $\sim q\text{Length}, \text{resendRTS}, \text{ns} = 20, \text{iaTime} = 0.01$. 
F  Results of the Graphical Analysis of MAC Layer Latency in R
F.1 Distribution Function of the MAC Layer Latency

In this appendix, the results of the graphical analysis with R related to the analysis of the distribution of the MAC layer latency are presented. The figures were generated based on the output data of experiment 2. The input parameters (channel idle and data frame length) are segmented, and each segment has a group id. The MAC layer latency values are classified into clusters based on these groups. The MAC layer latency distributions are plotted out for each cluster using boxplots. A boxplot is a way of summarizing a set of data measured on an interval scale. It is often used in exploratory data analysis. It is a type of graph which is used to show the shape of the distribution, its central value, and variability. The picture produced consists of the most extreme values in the data set (maximum and minimum values), the lower and upper quartiles (the upper and downer sides of the box), and the median (a horizontal wide line in the box). The narrowness of the boxplot reflects the number of the sampled data.

Axis Y shows the MAC layer latency and axis X represents the proportion of the channel in idle state. The channel idle percentage values were segmented into ten equivalent groups (noted as chIdleGroup in the caption of the figures) where each group has a number in sequential order. \( X = 10 \) means that the channel is idle in \( 90 - 100\% \) of the time, in case of \( X = 9 \) the channel is idle in \( 80 - 90\% \) and so on until \( X = 1 \) where the channel is idle in \( 0 - 10\% \). The data frame length values were divided into ten equivalent intervals from 0 to 1536 bytes, and for each segment a boxplot of the MAC layer latency was generated. The data frame length group is noted by dfLengthGroup in the captions of the figures. The mean length value of the actual length group is highlighted in bytes also in the boxplots.

The boxplots showed us that the channel utilization can never reach more than \( 90\% \) due to the behavior of the 802.11b DCF protocol. The minimum of the MAC layer latency time could probably be easily predicted with a linear prediction model using the data frame length as the input factor. The distribution of the MAC layer latency is skew and becomes more and more skewer by the increase of channel load. The variation of the MAC layer latency increases by a tendency more and less like exponential, with the increase of the channel utilization. In case of higher channel utilizations (more then 45%) its upper part becomes extremely wide, i.e., there is a high variation in the latency. The mean MAC layer latency increases slower than exponentially with the increase of channel utilization.
F.1.1 Analysis of the MAC Layer Latency on a Logarithmic Scale

Figure 173: MAC layer latency $\sim chIdleGroup, dfLengthGroup = 1$. Logarithmic.

Figure 174: MAC layer latency $\sim chIdleGroup, dfLengthGroup = 2$. Logarithmic.

Figure 175: MAC layer latency $\sim chIdleGroup, dfLengthGroup = 3$. Logarithmic.

Figure 176: MAC layer latency $\sim chIdleGroup, dfLengthGroup = 4$. Logarithmic.
Figure 177: MAC layer latency $\sim chIdleGroup, dfLengthGroup = 5$. Logarithmic.

Figure 178: MAC layer latency $\sim chIdleGroup, dfLengthGroup = 6$. Logarithmic.

Figure 179: MAC layer latency $\sim chIdleGroup, dfLengthGroup = 7$. Logarithmic.

Figure 180: MAC layer latency $\sim chIdleGroup, dfLengthGroup = 8$. Logarithmic.
Figure 181: MAC layer latency $\sim chIdleGroup, dfLengthGroup = 9$. Logarithmic.

Figure 182: MAC layer latency $\sim chIdleGroup, dfLengthGroup = 10$. Logarithmic.

Figure 183: MAC layer latency $\sim chIdleGroup, dfLengthGroup = 1$. Logarithmic. Zoomed-in.

Figure 184: MAC layer latency $\sim chIdleGroup, dfLengthGroup = 2$. Logarithmic. Zoomed-in.
Figure 185: MAC layer latency $\sim chIdleGroup, dfLengthGroup = 3$. Logarithmic. Zoomed-in.

Figure 186: MAC layer latency $\sim chIdleGroup, dfLengthGroup = 4$. Logarithmic. Zoomed-in.

Figure 187: MAC layer latency $\sim chIdleGroup, dfLengthGroup = 5$. Logarithmic. Zoomed-in.

Figure 188: MAC layer latency $\sim chIdleGroup, dfLengthGroup = 6$. Logarithmic. Zoomed-in.
Figure 189: MAC layer latency $\sim ch_{IdleGroup}, df_{LengthGroup} = 7$. Logarithmic. Zoomed-in.

Figure 190: MAC layer latency $\sim ch_{IdleGroup}, df_{LengthGroup} = 8$. Logarithmic. Zoomed-in.

Figure 191: MAC layer latency $\sim ch_{IdleGroup}, df_{LengthGroup} = 9$. Logarithmic. Zoomed-in.

Figure 192: MAC layer latency $\sim ch_{IdleGroup}, df_{LengthGroup} = 10$. Logarithmic. Zoomed-in.
F.1.2 Analysis of the MAC Layer Latency on a Linear Scale

Figure 193: MAC layer latency $\sim chIdleGroup, dfLengthGroup = 1$. Linear.

Figure 194: MAC layer latency $\sim chIdleGroup, dfLengthGroup = 2$. Linear.

Figure 195: MAC layer latency $\sim chIdleGroup, dfLengthGroup = 3$. Linear.

Figure 196: MAC layer latency $\sim chIdleGroup, dfLengthGroup = 4$. Linear.
Figure 197: MAC layer latency $\sim ch\text{IdleGroup}, df\text{LengthGroup} = 5$. Linear.

Figure 198: MAC layer latency $\sim ch\text{IdleGroup}, df\text{LengthGroup} = 6$. Linear.

Figure 199: MAC layer latency $\sim ch\text{IdleGroup}, df\text{LengthGroup} = 7$. Linear.

Figure 200: MAC layer latency $\sim ch\text{IdleGroup}, df\text{LengthGroup} = 8$. Linear.
Figure 201: MAC layer latency $\sim chIdleGroup, dfLengthGroup = 9$. Linear.

Figure 202: MAC layer latency $\sim chIdleGroup, dfLengthGroup = 10$. Linear.

Figure 203: MAC layer latency $\sim chIdleGroup, dfLengthGroup = 1$. Linear. Zoomed-in.

Figure 204: MAC layer latency $\sim chIdleGroup, dfLengthGroup = 2$. Linear. Zoomed-in.
Figure 205: MAC layer latency \( \sim ch\text{IdleGroup}, df\text{LengthGroup} = 3 \). Linear. Zoomed-in.

Figure 206: MAC layer latency \( \sim ch\text{IdleGroup}, df\text{LengthGroup} = 4 \). Linear. Zoomed-in.

Figure 207: MAC layer latency \( \sim ch\text{IdleGroup}, df\text{LengthGroup} = 5 \). Linear. Zoomed-in.

Figure 208: MAC layer latency \( \sim ch\text{IdleGroup}, df\text{LengthGroup} = 6 \). Linear. Zoomed-in.
Figure 209: MAC layer latency $\sim chIdleGroup, dfLengthGroup = 7$. Linear. Zoomed-in.

Figure 210: MAC layer latency $\sim chIdleGroup, dfLengthGroup = 8$. Linear. Zoomed-in.

Figure 211: MAC layer latency $\sim chIdleGroup, dfLengthGroup = 9$. Linear. Zoomed-in.

Figure 212: MAC layer latency $\sim chIdleGroup, dfLengthGroup = 10$. Linear. Zoomed-in.
Providing Tunable Security in IEEE 802.11i Enabled Networks

The basic idea of QoS is to provide mechanisms that can offer different service levels, which are expressed through well-defined parameters that are specified at run-time on the basis of need. Bit rate, throughput, delay, jitter, and packet loss rate are all examples of common QoS parameters suggested for packet networks. These parameters are all aimed to express (and guarantee) a certain service level with respect to reliability and/or performance. In this report, we investigate how security can be treated as yet another QoS parameter through the use of tunable security services. The main idea with this work is to let users specify a trade-off between security and performance through the choice of available security configuration(s). The performance metric used is latency. The concept is illustrated using the IEEE 802.11i wireless local area networking standard.