María de Fuentes J., Hernandez-Encinas L., Ribagorda A. (2018) Security Protocols for Networks and Internet: A Global Vision. In: Daimi K. (eds) Computer and Network Security Essentials. Springer, Cham.

DOI: 10.1007/978-3-319-58424-9_8

Chapter 8
Security Protocols for Networks and Internet: A Global Vision

José María de Fuentes, Luis Hernandez-Encinas, and Arturo Ribagorda

8.1 Introduction

Communication networks have evolved significantly in the last years. Since the appearance of ARPANET in the 1970s, computer networks and the Internet are at the core of modern businesses.

This trend is becoming even more acute in recent years, when a plethora of resource-constrained devices are starting to connect. This so-called Internet of Things (IoT) opens the door to advanced, ubiquitous, and personalized services [13].

The increasing need for communication also raises concerns regarding the security of the information at stake. How to determine if a given data item has arrived correctly, that is, without any alteration? How to ensure that it comes from the authorized entity? Are the data protected from unauthorized parties? These questions refer to basic protections about integrity, origin authentication, and confidentiality of the transmitted data, respectively.

In order to offer these security properties, numerous protocols have been proposed so far. In this chapter, representative examples are described in a very general way. The purpose is not to give technical insights into every part of each protocol but to understand the foundations and its main security implications. The reader is pointed to the actual reference documents for further information. Moreover, some general practical remarks are highlighted for each family of protocols.

J.M. de Fuentes (✉) • A. Ribagorda
Computer Security Lab (COSEC), Carlos III University of Madrid, Avda. Universidad 30, 28911 Leganés, Spain
e-mail: jfuentes@inf.uc3m.es; arturo@inf.uc3m.es

L. Hernandez-Encinas
Institute of Physical and Information Technologies, Spanish National Research Council (CSIC), Serrano 144, 28006 Madrid, Spain
e-mail: luis@iec.csic.es

The remainder of this chapter is organized as follows. Section 8.2 focuses on authentication protocols, with emphasis on Kerberos. Section 8.3 describes protocols for secure communication among entities, focusing on SSL/TLS and IPSec. Afterward, Sect. 8.4 introduces SSH, the best representative for secure remote communication protocols. In order to cover wireless security, Sect. 8.5 describes WEP, WPA, and WPA2 protocols. Finally, Sect. 8.6 concludes the chapter.

8.2 Authentication Protocols

Networks are composed of communicating nodes. To enable their authentication, it is necessary to clarify how this process is performed at different levels. In the link layer (layer 2 within the Open Systems Interconnection or OSI model [16]), a pair of protocols are distinguished, namely the Password Authentication Protocol (PAP) defined in RFC 1334 [12] and the Challenge Handshake Authentication Protocol (CHAP) defined in RFC 1994 [20]. Both PAP and CHAP work over the Point-to-Point Protocol (PPP) which enables direct communication between nodes. Other relevant authentication and authorization protocol is Kerberos. It works at application level to facilitate mutual authentication between clients and servers.

This section introduces the essential aspects of PAP (Sect. 8.2.1), CHAP (Sect. 8.2.2), and Kerberos (Sect. 8.2.3). Some practical remarks about these protocols are shown in Sect. 8.2.4.

8.2.1 Password Authentication Protocol (PAP)

PAP is a simple authentication mechanism similar to the use of username and password. The node which wants to be authenticated sends its name and password to the authenticator which compares both values with stored ones and authenticates accordingly. PAP is vulnerable to third parties that intercept the communication and capture the password because it travels in plain text. It is also vulnerable against trial-and-error attacks. Thus, as this is far from being a robust authentication mechanism, the use of other more robust authentication mechanisms, such as CHAP, is recommended.

8.2.2 Challenge Handshake Authentication Protocol (CHAP)

CHAP verifies node's identity periodically, ensuring that the password remains valid and that the node has not been impersonated in some way. In this protocol, usernames and passwords are encrypted.

Once the authenticator and the node which wants to be authenticated (let us refer to it as *user*) know a common secret value, the authenticator sends a challenge to the user. The latter applies a hash over the challenge and the secret value previously shared. The result of this operation is sent to the authenticator which compares this value with the stored one. If both values are identical, the authentication is performed; otherwise, the process usually finishes. The authenticator periodically sends new challenges to the user. Note that challenges include an identifier which is incremented each time, avoiding the reuse of responses, called replay attack.

8.2.3 Kerberos Protocol

Kerberos was developed to facilitate centralized and robust authentication, being able to manage thousands of users, clients, and servers [14]. It was developed by Massachusetts Institute of Technology (MIT) in 1987. The first three versions were exclusively used in MIT, but the fourth one, v.4, was open to computer companies to be included in commercial authentication systems. Finally, version 5 was adopted in 1993 by the Internet Engineering Task Force (IETF) as an Internet standard, RFC 1510, updated in 2005 [15]. Since then, it has been updated several times; the last update was in 2016 [21].

The goal of Kerberos is to provide centralized authentication between clients (acting on behalf of users) and servers, and vice versa. Applying Kerberos terminology, clients and servers are called principals. Besides, clients and servers are usually grouped into different domains called *realms*.

Broadly speaking, Kerberos uses a Key Distribution Center (KDC) which acts as a Trusted Third Party (TTP). KDC is composed of an Authentication Server (AS) and a Ticket Granting Server (TGS). These components, though different, may be in the same system. Moreover, TGS can be unique or various of them can coexist, even if there is just one realm.

In general, depicted in Fig. 8.1, Kerberos consists of three components: a client (C) acting on behalf of a user, a server (S) whose services are accessed by the client, and the KDC. A client which wants to work with a server should be authenticated first by the KDC (steps 1–2, Fig. 8.1), providing the identification of the server. Then, the KDC provides the client credentials to be used in the authentication process with the server. These credentials are transmitted encrypted with a session key. Such a key is generated by the KDC and securely transmitted to the client and the server (steps 3–4, Fig. 8.1). Indeed, session keys are distributed through *tickets*. A ticket is a certificate (which contains data to be used in the authentication) issued by the KDC and encrypted with the server's master key. This ticket is processed by the server as a means to authenticate (and authorize) the requesting user (step 5, Fig. 8.1).

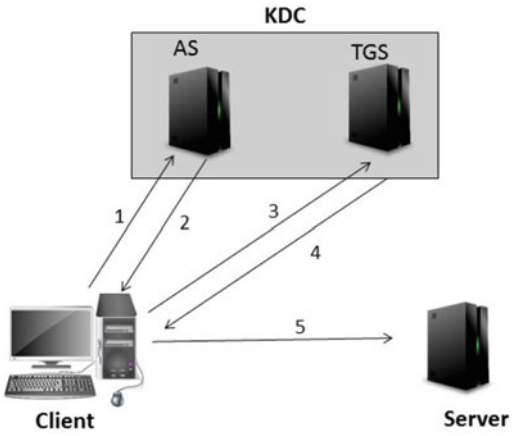

Fig. 8.1 Overview of Kerberos

8.2.4 Practical Remarks

Authentication protocols are daily used for many purposes. For example, Single Sign-On (SSO) architectures enable having a single entity in charge of authenticating the users. However, one critical remark is that the implementation of the authentication protocol can introduce vulnerabilities that are not present in the specification. For example, Microsoft Windows suffered from several Kerberos-related issues[1] that were addressed in an update of August 2016. Thus, when considering the use of a given authentication protocol, it is paramount to ensure that software components are up to date.

8.3 Secure Communication Protocols

In this section, two well-known secure communication protocols are described. In particular, Sect. 8.3.1 introduces SSL/TLS, whereas Sect. 8.3.2 describes IPSec. Practical remarks of this family of protocols are given in Sect. 8.3.3.

8.3.1 Secure Sockets Layer (SSL)

Secure Sockets Layer (SSL) was originally developed by Netscape, being SSL 3.0 (in 1996) the first stable version [6].

[1] https://support.microsoft.com/en-us/kb/3178465, (access Dec. 2016).

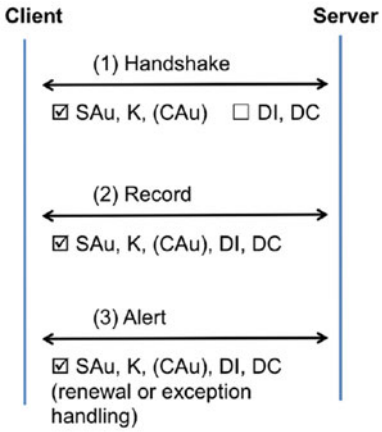

Fig. 8.2 Overview of SSL. The security provided by each subprotocol is highlighted. Key: SAu = Server Authentication, K = Agreement on key(s), (CAu) = Client Authentication (optional), DI = Data Integrity, DC = Data Confidentiality

SSL provides the following three security services: (1) data confidentiality, (2) data integrity, and (3) server authentication. Optionally, client authentication can also be requested by the server.

SSL 3.0 was attacked in 2014 using a technique referred to as POODLE.[2] As a consequence, most browsers have discontinued the support of this mechanism (e.g., Microsoft's Internet Explorer 11[3]).

An alternative to SSL is called Transport Layer Security (TLS). TLS 1.0 appeared shortly after SSL 3.0 and was indeed significantly similar. However, its publication stated that it was not meant to interoperate (by default) with SSL [3]. TLS was also the target of a variant of the said POODLE attack. Indeed, TLS is still receiving attention and as of December, 2016, its version TLS 1.3 is still under development[4] and TLS 1.2 is the one that should be used [4].

Without entering into technical insights, both SSL and TLS share a common structure in what comes to their basis. Indeed, three big subprotocols can be identified even in the most modern version of TLS. They are called *Handshake*, *Record*, and *Alert* subprotocols (Fig. 8.2). Each one is described below.

In the Handshake subprotocol (step 1, Fig. 8.2), both parties agree on the set of protocols that are going to be used. Furthermore, the server is authenticated against the client, by means of a X.509 public key certificate. After this step (and upon successful authentication), both parties agree on a shared key for the encryption of the transmitted data. Remarkably, it must be noted that the set of cryptographic protocols are negotiated through a set of rounds in which the server proposes some protocols and the client determines whether they are suitable for its resources.

[2] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-3566 (access December, 2016).

[3] https://blogs.microsoft.com/firehose/2015/04/15/april-update-for-internet-explorer-11-disables-ssl-3-0/#sm.0000x3es4m403dcm10bvx8k9qs1do (access December, 2016).

[4] https://tlswg.github.io/tls13-spec/ (access December, 2016).

Using this key and the algorithms defined in the previous phase, the Record subprotocol encrypts the actual data to be transmitted (step 2, Fig. 8.2). It also protects the message integrity using a Message Authentication Code (MAC) function.

Finally, the Alert subprotocol serves to notify when some abnormal issue takes place (step 3, Fig. 8.2). Indeed, it may serve to point out exceptions (from which the protocol may recover) or fatal, unrecoverable errors. An example of exception is when the server sends a certificate that is issued by an authority unknown to the receiver. On the other hand, fatal errors may happen, for example, when no agreement is reached in the handshake round.

8.3.2 IPSec

Internet Protocol Security (typically referred to as IPSec) is a technology for the protection of data authentication and encryption in a communication network [10]. One relevant aspect is that IPSec is not a protocol itself, but it is formed by a set of protocols, namely Internet Key Exchange (IKE), Authentication Header (AH), and Encapsulating Security Payload (ESP).

One critical remark is that IPSec operates at the network level, i.e., OSI level 3. This enables other applications and services belonging to upper layers to rely upon this technology. In the following, IKE, AH, and ESP are introduced.

8.3.2.1 IKE

Before two parties are able to exchange messages, it is necessary for them to agree on the set of protection mechanisms to be applied. This kind of agreement is called Security Association (SA) and is the rationale behind IKE [8]. In short, IKE enables setting up (and keep over time) SAs between two parties (Fig. 8.3).

IKE runs on top of the User Datagram Protocol (UDP) of the transport layer. As a practical remark, UDP does not offer any kind of reliable delivery. This means that every message may get lost without the sender noticing this issue. In order to cope with this issue, IKE is built in a challenge-response way which includes retransmission and acknowledgement mechanisms.

In an IKE run, two rounds are usually performed, namely IKE_SA_INIT and IKE_AUTH. The first one always takes place before any other round (step 1, Fig. 8.3). It enables agreeing on a shared key which is taken as a seed for two purposes—encrypting and authenticating exchanged data. IKE_SA_INIT is also applied to agree on the set of cryptographic algorithms that will be considered in the security association. This issue is also done in a challenge-response fashion, so the sender proposes a set of algorithms and the receiver either chooses one of them or returns an error if none is suitable.

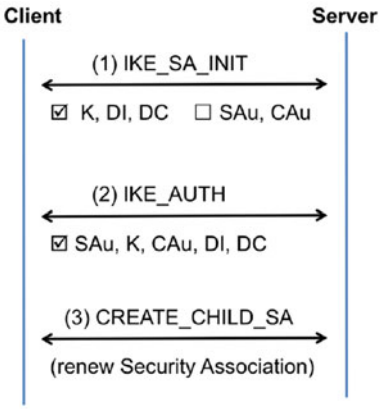

Fig. 8.3 Outline of IKE. The security achieved after each IKE round is highlighted. Key: SAu = Server Authentication, K = Agreement on key(s), (CAu) = Client Authentication (optional), DI = Data Integrity, DC = Data Confidentiality

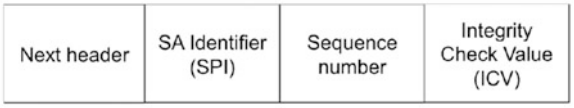

Fig. 8.4 Simplified header structure of AH

The SA itself is built in the IKE_AUTH round (step 2, Fig. 8.3). Using the key and the algorithms agreed in the previous round, both parties authenticate themselves and define the final issues of the SA. One important aspect is that the identities of the parties are encrypted using the shared key, thus ensuring the privacy of participants.

Every SA is meant to last for a given period of time. Indeed, the duration of a SA is agreed in this round. Once a SA expires, another SA comes into play. For this purpose, both parties may negotiate another SA using a CREATE_CHILD_SA round (step 3, Fig. 8.3). It must be noted that renewing SAs is beneficial from the security point of view—if a cryptographic key is used for a long time, it may become compromised in an easier way by an attacker.

8.3.2.2 Authentication Header (AH)

The main goal of the Authentication Header (AH) is to authenticate the packet content. Indeed, AH offers data integrity and sender authentication [22]. For this purpose, AH defines a header structure with four main fields (Fig. 8.4). First, an indication of the location of the next header. As AH is just one of the headers that can be included within an IPSec packet, this field points to the next in the packet to enable successful interpretation.

The next two fields are the identifier of the security association (referred to as Security Parameters Index, SPI) and the sequence number of the packet. Whereas the first field is critical to identify the security parameters to be applied, the second field enables the receiver to put packets in their correct order, no matter which

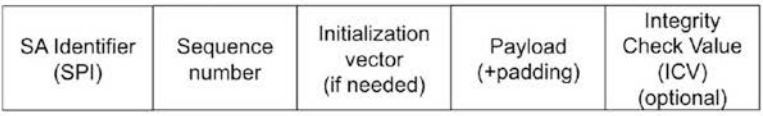

Fig. 8.5 Simplified packet structure of ESP

packet arrives first. The last field is the Integrity Check Value (ICV), which is the central element of AH header. Indeed, ICV is the element that authenticates the packet information. For this purpose, a Message Authentication Code (MAC) value is calculated, using the keys that have been determined in the security association pointed out by SPI. Remarkably, ICV is calculated over all non-mutable (or predictable) fields of the IP packet and it is mandatory in AH (as opposed to ESP in which it is optional; see Sect. 8.3.2.3) [22].

8.3.2.3 Encapsulated Security Payload (ESP)

After setting up the security association, the Encapsulated Security Payload (ESP) protocol deals with the actual protection of data [9]. For this purpose, a novel packet structure is defined (Fig. 8.5).

The first two fields are the Security Parameters Index (SPI) and the sequence number, already explained for AH (recall Sect. 8.3.2.2). The core of the packet is formed by its payload, which may have variable size. In order to avoid any third party to learn the size of the actual payload, padding is introduced. The last part of the packet structure is given by the Integrity Check Value (ICV). The ICV is calculated over all previous fields, but only if it is defined as needed within the SA in force. Otherwise, the field is omitted. It may happen, for example, when the service that is making use of IPSec already takes care of integrity, so there is no need for IPSec to check this issue as well.

8.3.2.4 Practical Setting: Tunnel vs. Transport Modes

IPSec can be configured to protect different parts of the packet. In particular, two modes are defined, namely *tunnel* and *transport* modes [10]. In tunnel mode, the whole IP packet is enclosed within another (outer) IP packet. In this way, all its elements are protected, including the header. Hence, no external entity can learn the actual identity of both participants. This header protection is not applied in transport mode, which only protects the actual payload of the IP datagram.

8.3.3 Practical Remarks

Secure communication protocols usually rely on an agreement phase between participants. As it has been shown in this section, SSL/TLS includes a round to negotiate cryptographic algorithms, whereas IPSec relies upon the concept of Security Association. Thus, it must be noted that the effective security achieved depends on two factors. On the one hand, the correctness of the software implementing the protocol. Thus, a first practical remark is that updated and well-proven cryptographic components should be applied. In order to validate that a given component is error free, recent projects such as Google's Wycheproof[5] can be considered. On the other hand, the negotiation is usually carried out without human intervention. Thus, software components must be properly configured to avoid weak settings. For example, Google Chrome can be set up to avoid obsolete cryptography.[6]

8.4 Secure Remote Communication Protocols

With the spreading of communication networks, remote management has gained momentum. In order to connect to another machine, Secure SHell (SSH) protocol is the standard alternative. This section describes the main aspects of SSH, introducing its evolution (Sect. 8.4.1) and its structure (Sect. 8.4.2). Afterward, some practical remarks are presented in Sect. 8.4.3.

8.4.1 SSH Evolution

SSH was first proposed in 1995 as a means to enable remote login in other computers.[7] This version (called SSH-1) was intended to replace other existing alternatives such as Telnet or rlogin. As compared to these technologies, SSH-1 already provided data confidentiality and integrity protection, as well as authentication of the communicants. However, several weaknesses were found in SSH-1, such as the use of weak Cyclic Redundancy Check (CRC) for integrity preservation. The design of SSH-1, as a single, monolithic protocol, was also criticized as it was not beneficial for the sake of maintainability. In order to overcome these issues, in 2006 a new version, SSH-2, was standardized by IETF (RFC 4251) [25]. There are three major improvements that motivated this evolution [2]:

[5] https://github.com/google/wycheproof (access Dec. 2016).

[6] https://www.chromium.org/Home/chromium-security/education/tls (access Dec. 2016).

[7] https://www.ssh.com/ssh/ (access December, 2016).

- **Flexibility.** In SSH-2, encryption algorithms and integrity checking functions are negotiated separately, along with their respective keys. Moreover, passwords can be changed over time. SSH-2 is also formed by three subprotocols.
- **Security.** SSH-2 features strong integrity checking. Moreover, the client can now authenticate using several means in a single SSH session. Public key certificates are now allowed for this purpose. Regarding the session key, it is now negotiated using Diffie–Hellman key exchange [5].
- **Usability.** Several sessions can be run in parallel. Moreover, host authentication is independent from the IP address, which makes SSH suitable for environments such as proxy-based networks.

8.4.2 SSH Protocol Structure

According to RFC 4251 [25], SSH-2 is formed by three main components, namely Transport Layer Protocol (TLP), Authentication Protocol (AP), and Connection Protocol (CP) (see Fig. 8.6). Each one is presented below.

TLP is the lower layer protocol, which provides with security mechanisms for server authentication, data confidentiality, and integrity (step 1, Fig. 8.6). For network bandwidth reasons, it can also provide with data compression. As SSH is typically placed in the session layer[8] (layer 5 of OSI), it leverages the transport protocols of the lower OSI layer. In particular, RFC 4251 specifies that TLP could be run on top of TCP/IP, but it could be used on top of any reliable transport protocol.

On the other hand, AP offers client (i.e., user) authentication (step 2, Fig. 8.6). AP runs on top of TLP. For client authentication, three main mechanisms are

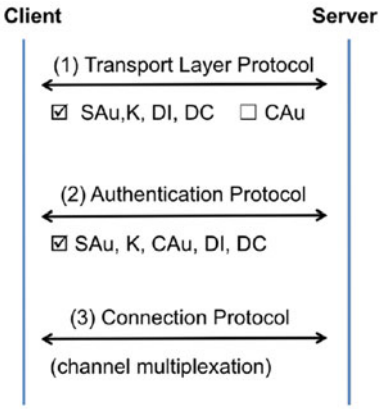

Fig. 8.6 Overview of SSH-2. The security provided by each subprotocol is highlighted. Key: SAu = Server Authentication, K = Agreement on key(s), CAu = Client Authentication, DI = Data Integrity, DC = Data Confidentiality

[8] https://www.sans.org/reading-room/whitepapers/protocols/understanding-security-osi-model-377.

allowed, namely public key authentication (using X.509 certificates), password, and host-based authentication. Only the first one is mandatorily supported for any implementation of SSH. The password-based method requires both parties to share a common secret (i.e., the password) in advance. Host-based is suitable for those sites that rely upon the host that the user is connecting from and the username within that host. As stated in RFC 4252, this form is optional and could not be suitable for high-sensitivity environments [24].

Last but not least, CP runs on top of AP and it is meant to enable channel multiplexation (step 3, Fig. 8.6). Thus, several SSH sessions can run simultaneously over a single connection. These sessions may serve to execute remote commands or to run x11-related software, that is, software programs that require graphical user interface.

8.4.3 Practical Remarks

SSH is used not only for remote communications but also for other purposes such as file transfer (Secure Copy Protocol, SCP). Thus, it is important to spread these remarks to all protocols that are based on SSH.

SSH has to be configured in the server, determining which are the considered cryptographic protocols. For example, in Ubuntu Linux systems these settings are located into the */etc/sshd/sshdconfig* file.[9] In this regard, one important aspect is to define which cryptographic algorithms are applied, avoiding weak (or vulnerable) ones. In the said file, directives *Ciphers*, *MACs*, and *KexAlgorithms* determine which encryption, MAC, and key exchange methods are allowed, respectively.

Moreover, as SSH is typically implemented through libraries or specialized software modules, it is essential to keep up to date on existing vulnerabilities. Indeed, as of December 2016 more than 360 vulnerabilities[10] can be found within the Common Vulnerabilities and Exposures (CVE) database, with some relation to SSH. It must be noted that some vulnerabilities are highly critical, even allowing unauthorized access to systems (e.g., vulnerability[11] CVE-2016-6474).

8.5 Secure Wireless Communication Protocols

Since the appearance of wireless networks, connectivity has become almost ubiquitous in developed countries and modern societies. However, security in these networks cannot be taken for granted. Thus, security protocols have been proposed

[9] https://help.ubuntu.com/community/SSH/OpenSSH/Configuring.

[10] http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=SSH (access Dec. 2016).

[11] https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-6474 (access Dec. 2016).

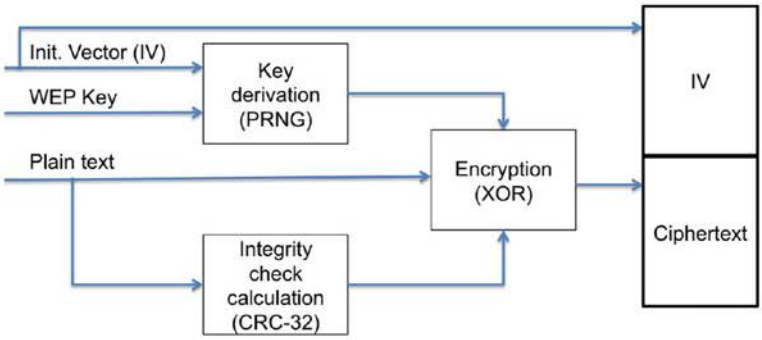

Fig. 8.7 Overview of WEP

since several decades. This section introduces the main examples of wireless security, namely WEP (Sect. 8.5.1), WPA, and WPA2 (Sect. 8.5.2). Practical considerations are introduced in Sect. 8.5.3.

8.5.1 Wired Equivalent Privacy (WEP)

WEP was included in 1997 as part of the wireless connectivity standard IEEE 802.11 [7]. This technology remained in the standard until 2004, when a revision made WEP to be superseded by WPA and WPA2 (explained below).

WEP offers data confidentiality. For this purpose, data is encrypted using algorithm RC4 [19] (Fig. 8.7). This algorithm is a stream cipher, which means that data is encrypted in a continuous manner, as opposed to block ciphers in which data is encrypted in a block-by-block basis. For this purpose, it is necessary that the encryption key comes in the form of pseudo-random sequence, which in the case of WEP is produced by a Pseudo-Random Number Generator (PRNG). Thus, PRNG is seeded with part of the WEP key, called Initialization Vector (IV).

WEP also provides data integrity. This is achieved by applying a Cyclic Redundancy Check (CRC) algorithm. In particular, CRC-32 is applied [11].

The short key length (initially, 64 bits), the lack of key renewal, as well as the election of cryptographic algorithms were the source of vulnerabilities in WEP. Furthermore, it must be noted that no explicit authentication is carried out from the access point. This facilitates launching attacks impersonating these nodes.

8.5.2 Wireless Protected Access (WPA and WPA2)

In order to overcome the limitations of WEP, in 2003 a novel protection mechanism called Wireless Protected Access (WPA) was developed [7]. The idea was to develop a novel technique that could be run directly on existing hardware.

WPA introduced Temporal Key Integrity Protocol (TKIP), a technique to improve key usage for encryption purposes [18]. In particular, TKIP enables mixing up the initialization vector with the root key, and using the result as input for the PRNG. Thanks to this action, the information available for the attacker was significantly reduced. Furthermore, TKIP features a sequence control mechanism, which is useful to counter replay attacks (i.e., attacks by repeating packets already sent).

Apart from a better key usage, WPA featured the use of an additional message authentication technique, called Michael. Thanks to Michael, should the access point receive incorrect integrity values within a period, a new session key would be applied for encryption. This is very beneficial to prevent external attackers gaining access to the network.

Despite these benefits, WPA relied upon the same cryptographic algorithms as WEP. Thus, although the attack chances were reduced, vulnerabilities were discovered as well. To address these issues, WPA2 was developed in 2004.

As opposed to WPA, WPA2 makes use of a different set of algorithms. In particular, AES-CCMP is applied. This algorithm comes from a particular instantiation of AES encryption algorithm. Remarkably, it also offers data integrity protection. Nowadays, WPA2 is resilient against the attacks that were feasible for its predecessors WEP and WPA [1].

8.5.3 Practical Remarks

Protocols for secure wireless communication can be configured in terms of the involved cryptographic algorithms. Specifically, for hardware-constrained devices it is important to carefully choose these algorithms, since there is a technical trade-off between security and performance. Thus, Potlapally et al. [17] have studied the impact of cryptographic algorithms for constrained devices. Although the study is focused on SSL, the implications are also valid for wireless protocols.

Apart from performance, cryptographic robustness is also relevant. As such, Tews and Beck [23] reported several practical attacks against WEP and WPA, along with some countermeasures. Remarkably, remediation usually involves tuning some parameters. Thus, as a practical recommendation, default settings should be revised by users to achieve the desired security level.

8.6 Conclusion

Computer networks and the Internet have greatly evolved in the last years. As a consequence, they are an integral part of any modern information technology system. In order to address their underlying security issues, a plethora of techniques have been proposed in the last decades.

In this chapter, an overview of network security-related protocols has been presented. They are focused on different areas, such as user authentication, secure communications, remote login, and wireless networks. For each protocol, a historical overview has been presented and the main features have been pointed out. The vast majority of technical issues have been left out of the discussion so that the reader gets the big picture of network security. Thus, Table 8.1 summarizes the main discussed aspects for each protocol.

Despite the amount of protocols described, many others have been intentionally left out of the scope of this chapter for space restrictions. Remarkably, other authentication technologies such as Radius or lower-level authentication protocols such as L2TP have not been addressed. However, we believe that the current overview is representative enough to show the recent evolution of these technologies.

Acknowledgements This work was supported by the MINECO grant TIN2013-46469-R (SPINY: Security and Privacy in the Internet of You), by the CAM grant S2013/ICE-3095 (CIBERDINE: Cybersecurity, Data, and Risks), which is co-funded by European Funds (FEDER), and by the MINECO grant TIN2016-79095-C2-2-R (SMOG-DEV—Security mechanisms for fog computing: advanced security for devices). Authors would like to thank the anonymous reviewers for their useful comments.

References

1. Adnan, A. H., Abdirazak, M., Sadi, A. S., Anam, T., Khan, S. Z., Rahman, M. M., et al. (2015). A comparative study of WLAN security protocols: WPA, WPA2. In *2015 International Conference on Advances in Electrical Engineering (ICAEE)* (pp. 165–169). Piscataway, NJ: IEEE.
2. Barrett, D., Silverman, R., & Byrnes, R. (2005). *SSH, the secure shell: The definitive guide* (2nd ed.). Sebastopol: O'Reilly.
3. Dierks, T., & Allen, C. (1999). *The TLS Protocol Version 1.0*. RFC 2246 (Proposed Standard). http://www.ietf.org/rfc/rfc2246.txt. Obsoleted by RFC 4346, updated by RFCs 3546, 5746, 6176, 7465, 7507, 7919.
4. Dierks, T., & Rescorla, E. (2008). *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC 5246 (Proposed Standard). http://www.ietf.org/rfc/rfc5246.txt. Updated by RFCs 5746, 5878, 6176, 7465, 7507, 7568, 7627, 7685, 7905, 7919.
5. Diffie, W., & Hellman, M. (2006). New directions in cryptography. *IEEE Transactions on Information Theory, 22*(6), 644–654.
6. Freier, A., Karlton, P., & Kocher, P. (2011). *The Secure Sockets Layer (SSL) Protocol Version 3.0*. RFC 6101 (Historic). http://www.ietf.org/rfc/rfc6101.txt.

Table 8.1 Summary of considered protocols

	Authentication			**Secure networks**			**Remote connection**		**Wireless networks**		
	PAP	**CHAP**	**Kerberos**	**IPSec**	**SSL (v. 3.0)**	**TLS (1.2)**	**SSH-1**	**SSH-2**	**WEP**	**WPA**	**WPA2**
Data confidentiality	–	Yes	Yes	Yes (ESP)	Several options. Agreed between client and server		Yes	Several symmetric algorithms (e.g., AES, 3DES, etc.)	RC4 (40 bits)	TKIP (per packet)	AES-CCMP
Data integrity	–	–	Yes	Yes (AH, ESP)	Several options. Agreed between client and server		Yes		CRC-32 (24 bits IV)	Michael	Improved MIC
Client authentication	Yes	Yes	Yes	Yes (AH, ESP)	Optional		Yes	Public key/Password/Hostbased	Yes		
Server/Access point authentication	–	Optional	Yes	Yes (IKE)	X.509 public key certificates		Yes	Public key/Certificate	Optional	Optional (EAP)[a]	Optional (WPA2-PSK)
OSI Layer	2 (Data link)		5 (Session)[b]/7 (Application)[c]	4 (Transport)	4 (Transport) and upwards		5 (Session)		2 (Data link)		

(continued)

Table 8.1 (continued)

	Authentication			**Secure networks**			**Remote connection**		**Wireless networks**		
	PAP	**CHAP**	**Kerberos**	**IPSec**	**SSL (v. 3.0)**	**TLS (1.2)**	**SSH-1**	**SSH-2**	**WEP**	**WPA**	**WPA2**
Date proposed	1992		1993	2005[d]	1996[e]	2008	1995	2006	1997	2003	2004
Date superseded/Obsoleted/Declared as insecure	2013[f]		None	None	2015[g]	None	2006	None	2003	2004	None

[a] https://www.sans.org/reading-room/whitepapers/wireless/evolution-wireless-security-80211-networks-wep-wpa-80211-standards-1109 (access Dec. 2016)
[b] https://www.sans.org/reading-room/whitepapers/protocols/understanding-security-osi-model-377 (access Feb. 2017)
[c] http://www.networksorcery.com/enp/default.htm (access Feb. 2017)
[d] https://tools.ietf.org/html/rfc4301 (access Feb. 2017)
[e] https://web.archive.org/web/19970614020952/http://home.netscape.com/newsref/std/SSL.html (access Feb. 2017)
[f] https://datatracker.ietf.org/doc/rfc1994/ (access Feb. 2017)
[g] https://tools.ietf.org/html/rfc7568 (access Feb. 2017)

7. Group, W. W. L. W. (2012). *802.11-2012 – IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*.
8. Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., & Kivinen, T. (2014). *Internet Key Exchange Protocol Version 2 (IKEv2)*. RFC 7296 (Internet Standard). http://www.ietf.org/rfc/rfc7296.txt. Updated by RFCs 7427, 7670.
9. Kent, S. (2005). *IP Encapsulating Security Payload (ESP)*. RFC 4303 (Proposed Standard). http://www.ietf.org/rfc/rfc4303.txt.
10. Kent, S., & Seo, K. (2005). *Security Architecture for the Internet Protocol*. RFC 4301 (Proposed Standard). http://www.ietf.org/rfc/rfc4301.txt. Updated by RFCs 6040, 7619.
11. Koopman, P. (2002). 32-bit cyclic redundancy codes for internet applications. In *Proceedings International Conference on Dependable Systems and Networks* (pp. 459–468).
12. Lloyd, B., & Simpson, W. (1992). *PPP Authentication Protocols*. RFC 1334 (Proposed Standard). http://www.ietf.org/rfc/rfc1334.txt. Obsoleted by RFC 1994.
13. Mattern, F., & Floerkemeier, C. (2010). Chap. From the internet of computers to the internet of things. *From active data management to event-based systems and more* (pp. 242–259). Berlin/Heidelberg: Springer. http://dl.acm.org/citation.cfm?id=1985625.1985645.
14. Neuman, B.C., & Ts'o, T. (1994). Kerberos: An authentication service for computer networks. *IEEE Communications Magazine, 32*(9), 33–38. doi:10.1109/35.312841.
15. Neuman, C., Yu, T., Hartman, S., & Raeburn, K. (2005). *The Kerberos Network Authentication Service (V5)*. RFC 4120 (Proposed Standard). http://www.ietf.org/rfc/rfc4120.txt. Updated by RFCs 4537, 5021, 5896, 6111, 6112, 6113, 6649, 6806, 7751.
16. ISO, I. (1994). IEC 7498-1: 1994 information technology-open systems interconnection-basic reference model: The basic model. ISO standard ISO/IEC, 7498-1.
17. Potlapally, N. R., Ravi, S., Raghunathan, A., & Jha, N. K. (2006). A study of the energy consumption characteristics of cryptographic algorithms and security protocols. *IEEE Transactions on Mobile Computing, 5*(2), 128–143.
18. Potter, B. (2003). Wireless security's future. *IEEE Security and Privacy, 1*(4), 68–72.
19. Rivest, R. L., & Schuldt, J. C. (2014). Spritz-a spongy rc4-like stream cipher and hash function. In *Proceedings of the Charles River Crypto Day*, Palo Alto, CA, USA (Vol. 24).
20. Simpson, W. (1996). *PPP Challenge Handshake Authentication Protocol (CHAP)*. RFC 1994 (Draft Standard). http://www.ietf.org/rfc/rfc1994.txt. Updated by RFC 2484.
21. Sorce, S., & Yu, T. (2016). *Kerberos Authorization Data Container Authenticated by Multiple Message Authentication Codes (MACs)*. RFC 7751 (Proposed Standard).
22. Stallings, W. (2002). *Cryptography and network security: Principles and practice*. Edinburgh: Pearson Education.
23. Tews, E., & Beck, M. (2009). Practical attacks against WEP and WPA. In *Proceedings of the Second ACM Conference on Wireless Network Security* (pp. 79–86). New York: ACM.
24. Ylonen, T., & Lonvick, C. (2006). *The Secure Shell (SSH) Authentication Protocol*. RFC 4252 (Proposed Standard). http://www.ietf.org/rfc/rfc4252.txt.
25. Ylonen, T., & Lonvick, C. (2006). *The Secure Shell (SSH) Protocol Architecture*. RFC 4251 (Proposed Standard). http://www.ietf.org/rfc/rfc4251.txt.