authentication, the HTTPS protocol is used; this protocol is also vulnerable to MITM attacks, as seen in section 3.2.5.3.

3.2.4. *Biometric authentication systems*

3.2.4.1. *Definition and principles of biometry*

Passwords and tokens may be characterized, respectively, as "what the user knows" and "what the user possesses". Biometry represents "what the user is", but also covers "what the user can do". The aim of biometry is to authenticate an individual based on one or more physical or behavioral characteristics.

A variety of biometric methods may be used in authentication protocols: fingerprint or handprint analysis, retina or iris scans, signature analysis, facial recognition, etc. The identity of a person may be verified by comparing the obtained reading with a model stored in a database consulted by the authentication server. Two different approaches may be used:

– *verification* of a nominative reading, i.e. using a single database entry with a known identifier (1:1 comparison);

– *identification* of an anonymous reading, tested for all of the samples in the database (1:*N* comparison).

In both cases, if the correspondence between the reading and the sample is judged to be sufficient, the identity of the user is considered to be verified. The degree of correspondence is evaluated by a mathematical analysis that defines an acceptance threshold above which the identity is verified. A number of organizations have worked on the standardization of biometric data; in addition to the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC), these include the National Institute of Standards and Technology (NIST), which has established the Common Biometric Exchange Formats Framework (CBEFF) standard, defining the digital format used to exchange biometric data [NIS 04].

By construction, biometric authentication can only operate in a symmetric model. The essential difference from the notion of a password lies in the personal, non-secret, and supposedly non-modifiable and non-imitable nature of the data. The fact that biometry is based on individual characteristics means that "authentication" ceases to be the most appropriate term; in reality, biometry is used for *identification*, and the individual declares *who they are* using biometric modalities. This avoids the need to memorize information, something that represents one of the major drawbacks of password authentication. Moreover, the characteristics of biometry guarantee natural association between the supplied information (the modality) and the identity of the user.

There are, however, a number of problems involved in the use of biometric authentication systems. In technical terms, biometric modalities evolve, and in some cases, can be easily imitated. In economic terms, biometric authentication systems require the use of dedicated reading equipment, which comes at a price. Finally, the use of these systems raises ethical concerns: fingerprinting is already considered as an attack on privacy in certain circles, and systems that go even further, such as smart corridors (developed by Thalès in 2008), which analyze people's walking movements in order to detect "suspicious" behavior patterns, are liable to generate considerable opposition.

3.2.4.2. *Limitations of biometric authentication systems*

In this section, we shall concentrate on the technical limitations of biometric systems, which take two forms.

Biometric modalities are an integral part of human bodies or behaviors, meaning that they are necessarily subject to evolution over time and to the hazards of everyday life – in other words, to change. A fingerprint may deteriorate, a voice may be altered by a person's state of health, and faces change over time. This means that the presented modality may be refused, even if the individual is who they claim to be. Moreover, in cases where an individual is the victim of severe physical trauma (loss of a finger or hand, detached retina, damage to the eyeball, damage to vocal cords, etc.), these identification elements cannot be replaced: unlike passwords, which can be changed as often as necessary, the affected biometric modality will need to be substituted for another same type, wherever possible. This is naturally limited, for example, to ten modalities for fingers, two for the eyes, one for the voice, handwriting, face, etc. This is a significant limitation, and certain precautions need to be taken in defining a modality acceptance threshold.

Furthermore, it is generally possible to imitate biometric modalities, although the quality of imitations and their subsequent ability to trick an identification system is variable. Although it is not possible, in practice, to modify one's physiognomy or behavior in order to resemble another person exactly, it is possible to produce an imitation using a sample retrieved by the attacker [MAT 02]. This may be indirect (fingerprint collection, voice recording, photography of a face, etc.) or direct and invasive (amputation of a finger, removal of an eye, etc.).

For this reason, biometric readers use a set of countermeasures. One of the main methods used is *liveness detection*, which aims to guarantee a genuine link between the presented modality and the individual in question. In the case of fingerprints, this method allows the detection of prosthetics or false fingers, and even amputated digits (via a measurement of blood pressure or natural perspiration, for example). A state of the art of *liveness detection* is presented in [SIN 11].

These considerations show that the level of security provided by biometric authentication depends largely on the quality of the biometric reader. The use of individual physical modalities is not sufficient in order to design a faultless authentication system, and should be combined with other authentication factors in order to produce a strong solution.

3.2.5. *The TLS protocol*

Password systems, whether static or dynamic, and biometric systems are fundamentally symmetrical, i.e. based on preliminary sharing of a secret.

TLS [RFC 08] is based on PKIs (see section 3.1.4.2), and therefore uses an asymmetric architecture. This authentication method uses the principles of asymmetric cryptography, i.e. the complementarity between a private key, never transmitted over a network, and a public key, linked to the identity of the user via an X.509 certificate. TLS is therefore different from all of the other authentication systems presented above, and is currently a fundamental protocol for network security.

3.2.5.1. *Definition and principles of the TLS protocol*

The TLS protocol, formerly known as Secure Sockets Layer (SSL), was developed by Netscape in the 1990s. The protocol is situated between the transport layer and the application layer, providing an additional security layer for the protection of application data. This is the manner in which the protocol is generally used: the TLS protocol uses an encrypted channel to add security for all application-level protocols (File Transfer Protocol (FTP), HTTP, Simple Mail Transfer Protocol (SMTP), etc.), with no interoperability constraints. It requires a reliable connection at transport level, such as the TCP protocol. By default, application-level protocols exchange data in unencrypted format, creating a number of risks; the use of a standardized, interoperable protocol offering confidentiality and data integrity is therefore highly recommended, something that goes a long way to explaining the success of TLS.

The TLS protocol allows us to establish the following security services between two entities in a network:

– *data confidentiality*, obtained by creating a secure tunnel between the client and the server, in which data are encrypted using a symmetric cryptography algorithm, such as RC4 or AES.

– *data integrity* via the calculation of a message authentication code (MAC) for each exchanged fragment of application data. This is generally carried out using a HMAC [RFC 97] algorithm, based on classic hash functions such as MD5 or SHA-1.

– *replay protection* by adding a sequence number, used in the calculation of the MAC.

– *simple* or *mutual authentication* using X.509 digital certificates.

The TLS protocol has been subject to a number of security analyses, such as [PAU 99] and [HE 05], which guarantee its robustness. Generally speaking, the only problem with mutual TLS authentication lies in the trustworthiness of PKIs, and TLS cannot be overcome by any classic forms of attack. However, it is not widely used, as it requires users to manage their own certificate. Simple authentication, where the server is authenticated to the client, is generally preferred, as the client is then able to authenticate using another method (the classic "username/password" combination, in most cases), benefitting from the secure tunnel that ensures the confidentiality and integrity of exchanges at application level. This approach remains very different from mutual authentication, where a single protocol is used to authenticate two entities to each other.

To illustrate this difference, let us consider the classic case, used by most banks, where users are authenticated using a password once the server has been authenticated and the TLS tunnel established. Eavesdropping attacks will not be possible in this case, and brute force or dictionary attacks will also be invalid, in terms of identity theft, due to the limited number of authorized authentication attempts. However, phishing attacks work well in this context: unsuspecting users will not think to check whether or not a TLS session has been established, and are therefore susceptible to provide their details to an attacker.

3.2.5.2. *Digital identity in the TLS protocol*

The digital identity model used in TLS is unlike the classic paradigms used in symmetric models and requires more detailed consideration.

Currently, TLS authentication is based almost exclusively on the use of X.509 certificates. This means that the digital identity of an individual or a machine is entirely represented by the certificate. The main (but not the sole) advantage of this approach is that a denomination (the *Common Name*, usually abbreviated to CN, for example the uniform resource locator (URL) of a server or a user name) is linked to an asymmetric public key. The entity holding the certificate therefore also holds the associated private key, which is used, without being transmitted, to prove the entity's status as the legitimate holder of the certificate.

In accordance with the PKI model, certificates are obtained on request from a recognized CA. Certificates generated by these authorities carry a digital signature, preventing third parties from modifying the contents of the certificate, which would invalidate the signature. Security is thus dependent on the solidity of the hashing function used for the signature, and weaknesses in this element may lead to collision problems between certificates, as in the case of the MD5 function [STE 07]. In the case of a server certificate, verification involves the following steps:

– verification of the validity date of the certificate;

– verification of the trustworthiness of the CA. Given that a CA may delegate certificate generation to other organizations, the verification process requires verification of the certificates of the full chain of CAs involved in certification until a recognized CA, predefined by the verifying entity, is found. For a user verifying the certificate of a Web server, for example, the root CA must feature in a list stored by the browser;

– verification of the digital signature of the certificate using the public key of the CA. This involves decrypting the digital signature using the CA's public key, and calculating the hash of the certificate using the same function used by the CA in producing the signature, typically MD5 or SHA-1. If the two values are equal, then the signature is verified.

Once these three stages have been successfully passed, a server certificate will be considered valid. However, an additional step is required to avoid vulnerability to MITM attacks, and it is surprising that this step is not explicitly included in the TLS protocol. This step involves verification of the server domain name, which is compared to the *distinguished name* (DN) of the supplied certificate.

Note that this final verification is impossible in the case of a client certificate. However, clients are required to produce a digital signature for certain data exchanged during the establishment of a TLS session, in order for the server to ensure that the user holds the private key associated with the public key concerned by the certificate.

Other PKI models may be envisaged for more specific contexts, for example limiting the influence of CAs in certificate creation [BOU 11]. However, the reference model, as summarized in this section, is currently unrivaled; the widespread success of TLS naturally leads to the use of X.509 certificates, legitimizing their mode of management.

3.2.5.3. *Limitations of the TLS protocol*

The main limitation of the TLS protocol, as we have stated, is associated with the use of PKIs. In addition to genuine questions concerning the trust placed in CAs [SOG 11], or considerations regarding complexity and the cost of establishing the necessary architecture, a problem has arisen in current Internet use whereby legitimate servers are subject to a relatively high number of authentication failures, due to the use of out-of-date certificates or simple non-recognition by browsers. This is far from ideal, and also generates a certain level of confusion for users who are frequently faced with benign errors, which they can choose to ignore; these users will then be unable to identify a genuine attack.

However, another important weakness makes TLS vulnerable to MITM attacks. This vulnerability does not concern the protocol itself, but its implementation in the HTTPS (HTTP over SSL) protocol. Attackers may prevent the victim from creating HTTPS connections with a server, i.e. by replacing all HTTPS requests with HTTP requests. To do this, the attacker launches an MITM attack, intercepting the first request sent to the secure server, benefitting from the fact that the first request is rarely in HTTPS, and often in simple HTTP, as the victim does not type "https" explicitly into the URL. In other words, the TLS session is not initiated immediately, and the aim of the attacker is to prevent this session from being established. As we see in Figure 3.7, having intercepted an HTTP request, the attacker sends an HTTPS request to the server, and receives an HTML page in response. The attacker then replaces all of the HTTPS links in the HTML, replacing them with HTTP. In this manner, the victim communicates data to the attacker in unencrypted form, while the attacker continues to exchange encrypted information with the server. From the victim's perspective, only subtle indications, for example the color of the URL bar used to show the security of exchanges in certain browsers, will be missing; this will have no impact on non-specialist users. This weakness was identified by Marlinspike [MAR 09]. The author also developed a tool known as *SSLstrip* as proof of the concept involved in this weakness.

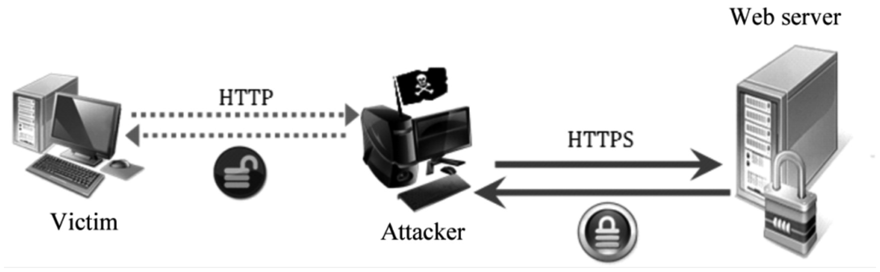

Figure 3.7. *Man-in-the-middle attack on the HTTPS protocol*

Although we need to be aware of these limitations, we should note the considerable significance of this protocol, which plays a key role in network security, notably in securing protocols at application level. When used for mutual authentication, it represents a genuinely strong solution, as long as sufficient precautions are taken to conserve the private key.

3.2.6. *The role of smart cards*

While they do not constitute an authentication system in their own right, it is useful to discuss the specific points of secure microcontrollers, such as smart cards, due to the key role they play in the design of strong authentication procedures.

The storage of secrets, whether passwords, symmetric keys or private asymmetric keys, has been mentioned on several occasions in this chapter. We have implicitly presumed that this storage is secure, without considering the meaning or the way in which this is implemented. These questions, however, are important.

One approach consists of storing these secrets in encrypted form, using a symmetric key derived from a password which the user needs to memorize. This password will be required when the user wishes to use one of their secrets. While this solution does offer secure storage, it is not particularly user-friendly. Moreover, the use of a password, even if it is not stored, is not satisfactory in the long term; malware such as keyloggers may be used to retrieve passwords used for secret encryption, making the information available to an attacker.

Smart cards offer another approach, as they constitute a trusted environment, with physical and logical countermeasures against attacks aiming to read or copy data in a fraudulent manner. If secrets, such as the private asymmetric key, are stored on a smart card, it is almost impossible for an attacker to retrieve them. The use of a PIN protects the card against use by a third party. As a physical object that must be in the possession of the legitimate user, a smart card is an excellent complement to robust authentication systems that require a trusted environment of this type for the storage of private elements. Smart cards may notably be used in addition to the TLS protocol, following the PKCS#15 standard [RSA 00], where they are used to store the private key and, sometimes, the corresponding certificate, in order to facilitate mutual authentication.