TTM4135 Worksheet 8: TLS, IPsec, Secure Mail and Secure Messaging

Tjerand Silde and Emil August Hovd Olaisen
{tjerand.silde, emil.august.olaisen}@ntnu.no

Spring 2026

1. Review the definitions of the following concepts:
   (a) TLS ciphersuites
   (b) TLS Record protocol
   (c) TLS Handshake protocol
   (d) 0-RTT protocols;
   (e) Host-to-host, gateway-to-gateway and host-to-gateway IPsec architectures;
   (f) IPsec transport mode and tunnel mode
   (g) Domain Keys Identified Mail (DKIM)
   (h) OpenPGP
   (i) Signal protocol
2. Consider the following ciphersuite specifications for TLS. What do each of them mean? Comment on the security of each of the choices regarding: choice of algorithms; key length; forward secrecy.
   (a) TLS_RSA_WITH_RC4_128_MD5
   (b) TLS_RSA_WITH_NULL_SHA
   (c) TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
   (d) TLS_DHE_DSA_WITH_AES_256_CBC_SHA256
   (e) TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
   (f) TLS_CHACHA20_POLY1305_SHA256

Solution: The codes before the WITH statement are concerned with the public-key algorithms used for key exchange in the Handshake protocol while the codes after the WITH statement are concerned with the algorithms used for encryption and authentication in the Record protocol (and sometimes also the PRF used for session key generation).

(a) TLS_RSA_WITH_RC4_128_MD5:

- The RSA option is used for key exchange so the master secret is sent by the client encrypted using the server's RSA public key. This is still a fairly common method of key exchange in TLS and is generally believed secure but does not provide forward secrecy.
- RC4 is used for encryption with a 128-bit key. This is no longer a recommended choice since attacks have shown that RC4 is best avoided. MD5 is used to produce the MAC using the HMAC algorithm. MD5 is not a good hash function for many purposes since collisions have been found several years ago. While there are no known attacks on MD5 when used in HMAC, it is usually recommended to use a more recent hash function.

(b) TLS_RSA_WITH_NULL_SHA

- The RSA option is used as in (a).
- No encryption is used which may be justifiable in some circumstances but is potentially a major weakness. SHA-1 is used to produce the MAC using the HMAC algorithm. SHA-1 has also been broken, and recently actual collisions have been found. While there are no known attacks on SHA-1 when used in HMAC, it is usually recommended to use a more recent hash function.

(c) TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA

- The DHE code means that Diffie-Hellman with ephemeral keys is used for key exchange.
- Triple-DES in cipher block chaining mode is used for encryption in the Record layer. Triple-DES has a 168-bit key. Although the key length is probably enough, it is recommended today to avoid such block ciphers with only 64-bit block length. In addition, it is much slower than AES. SHA-1 is used for HMAC as in (b).

(d) TLS_DHE_DSA_WITH_AES_256_CBC_SHA256

- Diffie-Hellman with ephemeral keys is used with DSA signatures to authenticate. These algorithms are running in $\mathbb{Z}_p^*$ for a suitable length of prime p.
- AES in cipher block chaining mode is used for encryption in the Record layer with a 256-bit key. This key length is more than adequate for secure encryption. SHA-256 is used for HMAC. Although this length of MAC is secure enough for most security purposes it is not well matched with the encryption key length. Using instead SHA-512 would mean that finding collisions in a chosen message attack takes the same number of expected steps as brute-force key search on AES with a 256-bit key.

(e) TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

- Diffie-Hellman with ephemeral elliptic curve keys is used with ECDSA signatures to authenticate.
- AES in GCM mode is used for encryption in the Record layer with a 128-bit key. GCM mode provides authenticated encryption in one algorithm so no MAC is used. SHA-256 is used for the PRF used to derive keys from the pre-master key obtained in the handshake protocol.

(f) TLS_CHACHA20_POLY1305_SHA256

- This is a TLS 1.3 ciphersuite, which are the only ones that do not specify handshake algorithms. As for all TLS 1.3 ciphersuites, ECDHE will be used for the handshake (using an elliptic curve agreed in the extensions for the hello messages).
- As for all TLS 1.3 ciphersuites, an authenticated encryption algorithm is used for encryption on the record layer, in this case using the ChaCha/Poly1305 scheme.
- SHA-256 is used for the hash in the key derivation function used to derive the keys for use in the record layer.

3. Consider man-in-the-middle (MITM) scenarios with TLS in which a root certificate is added to the client machine.
   (a) What are scenarios in which such a MITM may be regarded as legitimate?
   (b) How might a root certificate get added to a machine in practice?
   (c) In what sense are these scenarios always bad for security, no matter how they are implemented?

Solution:

(a) In a corporate environment a company may regard it as legitimate to decrypt all company traffic at a network gateway for reasons such as checking for incoming malware and outgoing company secrets. Another scenario is that companies may believe that if they provide a service (such as advertising) this justifies decrypting and altering client traffic.

(b) In a corporate environment the company may insist on a standard operating environment for all company computers. They can then set up the root certificates to include a corporate gateway certificate. Then their corporate gateway can issue certificates for any entity which will be accepted by the client browser. Another way is that pre-loaded operating systems can have root certificates added to those in a standard browser.

(c) The above scenarios break the end-to-end security of the TLS connection. The client and server now need to trust that the intermediate server is secure and will treat their data securely.

4. This question illustrates *padding oracle attacks* on TLS. TLS uses padding on plaintexts with CBC mode encryption for block ciphers (like AES). Padding works by adding at least one byte. The padding is the representation of the number of padding bytes (padding length) preceded by that same value repeated for that number of bytes. Thus possible padding is "00" or "01 01" or "02 02 02" or .... In this question assume that the receiver always outputs an error if the padding is incorrect, and this error explicitly states that a padding error occurs.

   (a) How will this padding be checked and correctly removed by the receiver?

   (b) What happens if the last byte of the block $n-1$, C_{n-1}, of a n-block CBC ciphertext is altered? In what circumstances will the padding in the decryption of the last block, C_n, be correct? (Write an equation for the case when there is one byte of padding.)

   (c) Use the above observation to show how a padding oracle attack works to find one byte of the plaintext. How many attempts are required in order to guarantee finding that byte?

   (d) How can this attack then be extended to obtain all bytes in $P_n = D(C_n, K)$?

Solution:

(a) The receiver reads and removes the last byte and interprets it as an integer x. The receiver then removes x further bytes. For secure operation in TLS, it is essential that the receiver also checks that all removed bytes indicate the same value x.

(b) As shown in the lectures, a change in the last byte of ciphertext block C_{n-1} will result in a random recovered plaintext P_{n-1} and the same change (in terms of bits flipped) in the last byte of P_n. The padding will be correct if the change makes the last byte of P_n equal to 00.

Let us write $D(C_n, K) = X_0 \parallel X_1 \parallel \ldots \parallel X_{15}$, assuming that the encryption algorithm is AES with 16 bytes of output. If we also write $C_{n-1} = C_{n-1,0} \parallel C_{n-1,1} \parallel \ldots \parallel C_{n-1,15}$ then this padding will be accepted if

$$C_{n-1,15} \oplus X_{15} = 00$$

since in CBC mode we have $P_n = D(C_n, K) \oplus C_{n-1}$.

(c) The attacker can now try any different values for $C_{n-1,15}$ until the padding is accepted. The attacker can then assume that $C_{n-1,15} = X_{15}$. This will require at most 255 trials, or 128 on average.

(d) Once X_{15} is known, the attacker can find X_{14} by fixing $C_{n-1,15} = X_{15} \oplus 01$ and changing $C_{n-1,14}$ until the padding is accepted again. This happens when:

$$C_{n-1,14} \oplus X_{14} \parallel C_{n-1,15} \oplus X_{15} = 01 \parallel 01.$$

Thus X_{14} is found after another approximately 128 trials, and the next byte can be found looking for padding $02 \parallel 02 \parallel 02$ and so on.

5. Compare the handshake protocols in TLS 1.2 and TLS 1.3 as shown in the slides. The client is not permitted to send application data before it receives the finished message from the server.
   (a) How much longer must the client wait before sending application data in TLS 1.2 compared with TLS 1.3?
   (b) What attacks are possible in TLS 1.2 if the client could send application data in Phase 3 after its own finished message?

Solution:

(a) In TLS 1.2 the client cannot send application data until after two complete round trips, so not until part of its third message flow to the server. In TLS 1.3 the client can send application data after one round trip, in part of its second message flow. So the client only has to wait about half of the time in TLS 1.3 compare to TLS 1.2.

(b) Note that the client already has the session keys after it sends its finished message in TLS 1.2. Therefore it could use the keys to protect its application data already. However, at this point all of the communication from the server has been in plaintext and the client has no assurance that server messages have not been replayed or constructed by an attacker. In particular the attacker may have changed the chosen ciphersuite and version number to be less secure than the one preferred by the client.

6. Compare IPsec in host-to-gateway architecture with TLS. Consider the following scenarios and discuss which would be most suitable to provide security in each case.
   (a) You have two applications on your server which you want to secure with independent keys and different security services.
   (b) You want to secure a server which has a number of applications and you may want to add new applications in the future without changing the security settings.

Solution:

(a) IPsec in host-to-gateway architecture only provides security from the host to the gateway and not all the way to the application server. Therefore it is not suitable for this application. However, the two applications can be set up with different certificates and ciphersuites in TLS and thus provide a solution for this scenario.

(b) This time IPsec in host-to-gateway architecture provides exactly what is needed. It provides (the same) security for all the IP traffic which goes across the host to gateway connection and so any new applications will also be protected. However, end-to-end security is not provided for any applications in this way. Using TLS we would have to set up the protocol and certificates for the new application when it is added.

7. Three possible ways to combine encryption and MACs are:
   - encrypt first and apply the MAC to the ciphertext;
   - apply the MAC first and encrypt plaintext and MAC together;
   - apply the MAC and encrypt the plaintext separately.

Which of these is used in the TLS Record Protocol (up to version 1.2) and which is used in the ESP protocol of IPsec? Why is the third not suitable in general? (Remember that the purpose of a MAC is only to provide authentication/integrity and not confidentiality.)

Solution:

- The slides show the packet format for ESP. Here we can see that the data plus the ESP trailer are encrypted first, then the ESP header is added and the whole is authenticated with the MAC. This means that IPsec follows the encrypt-then-MAC process which is today regarded as the correct order.
- The slides shows the Record protocol format. This illustrates that the MAC is computed on all the other data before the payload and MAC are encrypted. This choice may seem natural since the MAC is directly on the data whose integrity should be protected. Unfortunately this order does not always provide the desired security even when the MAC and encryption algorithm are both secure.
- Applying MAC and encryption separately is generally not a good idea. The simple reason is that a MAC is not designed to provide confidentiality. Although we may not see any obvious way to obtain information about the plaintext from the MAC we have no guarantee that it does not leak information.

8. Explain why the lack of interaction in email delivery prevents the possibility to achieve forward secrecy for secure email. Is there a way that forward secrecy could be approximated for email?

Solution: Email uses the store-and-forward principle. The recipient is *not* assumed to be online when the email is sent, and before receiving the encrypted message we cannot assume that the recipient has contact with the sender. Since there is no interaction between endpoints, it must be possible to decrypt the message with only the long-term key. Thus, when the long-term key is compromised later, any captured encrypted mail must be decryptable by the attacker.

There are some ways that forward secrecy can be achieved to some degree.

- As mentioned in the lecture, link encryption is often used for mail while in transit. Since this link encryption uses TLS, forward secrecy can be achieved for an eavesdropper on those links. This does not help if the attacker controls one of the links.
- We can update the public encryption key and corresponding private decryption key regularly, for example on a daily basis. There are some cryptographic schemes to help do this systematically by essentially adding the date into the public key. This does not help protect today's traffic if the daily decryption key is compromised before it is updated. Also it means that old messages cannot be decrypted.
- The above impossibility argument for complete forward secrecy assumes that the long-term key is static. If the long-term key is updated every time a message is received, such that it can no longer be used to decrypt that same message, then the argument breaks down. Recently some protocols have been designed to achieve this property, but they are not very efficient.

Finally we may note that instead of email, instant messaging allows interaction which can be used to provide forward secrecy in protocols like Signal.

9. In hybrid encryption, such as used in PGP, is it better to have the public key encryption or the symmetric key encryption to be the stronger of the two?

Solution:

Breaking the long-term decryption key leads to breaking all symmetric keys due to the lack of forward secrecy. However, breaking one symmetric key breaks only one session. Therefore we should ensure that the (believed) strength of the public key encryption is at least as good as the private key encryption.

10. End-to-end security and link security are two ways of providing network security. What are some of the advantages and disadvantages of each? What protocols, or configurations, are available to provide each of these types of security in

    (a) Email

    (b) IPsec

Solution:

Note that we often think of confidentiality when talking about end-to-end security, but the arguments below are relevant also for integrity/data authentication.

With end-to-end security the endpoints (users) control their own level of security and can generate their own keys. They do not need to trust any third party for their security. However, information required to transmit the messages across the network (headers) cannot be encrypted end-to-end, since it needs to be used by network nodes. Leakage of such *metadata* can be very useful for an attacker.

With link security the header data can be protected, since the nodes typically carry traffic between many different sources and destinations. The network operators may have better capability and resources than the endpoint users to properly set up security.

(a) For email: PGP security is end-to-end, DKIM is domain to domain, and STARTTLS is link by link.

(b) For IPsec we have different architectures. Only the host-to-host architecture is end-to-end. Commonly gateway-to-gateway architecture is used.

11. The messaging protocol Signal uses pre-computed Diffie-Hellman keys to protect the *first* communicated message in any conversation. A client A pre-computes many $t_i = g^{x_i}$ values which are stored on the Signal server. When another client B starts a new conversation with A:

    - B is given a previously unused pre-computed key of A, $t_i = g^{x_i}$, and then t_i is deleted from the server;
    - B chooses an ephemeral Diffie–Hellman private key x_B;
    - B computes a message key k as a hash of $g^{x_i x_B}$;
    - B sends the first message to A protected by k and also sends g^{x_B};
    - When A receives the message, she uses x_i to recompute k and recover the first message and then deletes x_i.
    - For the next message A computes a new ephemeral Diffie–Hellman value which she combines with g^{x_B} to form the next message key.

    Since the number of new conversations that will be started is not predictable, Signal has a fallback mechanism to use the last available pre-computed key for many conversations until the supply of pre-computed keys is replenished. Signal suggests keys be replenished once a week, or once a month.

    Discuss how this process influences forward secrecy for the first message in each conversation. Include a discussion of whether the pre-computed values should be classed as long-term or ephemeral keys.

Solution: The pre-computed keys do not fit the definition of either long-term keys (which should last throughout the use of the protocol) or of ephemeral keys (which should exist only when they are used to set up one key).

If we define the pre-computed keys as *long-term*, then the definition of forward secrecy should allow an attacker to obtain these keys and then compute the key k used to protect the first message. (Due to the Diffie–Hellman ratcheting the later messages are protected by different keys and are not vulnerable.) If we define the pre-computed key to be *ephemeral* then we can say that forward secrecy is provided by the Signal protocol. This shows that when we define forward secrecy we need to be more precise about key lifetime and consider the risk of using medium-term keys which may last much longer than usual ephemeral keys.

If an attacker is able to exhaust the pre-computed keys by starting many fake conversations with A then the fallback key will be used to protect all subsequent first messages and forward secrecy is in practice lost during the window until the pre-computed keys are replenished.