University of Oslo
INF3510 Information Security
Spring 2018
Workshop Questions & Answers

Lecture 3: Cryptography

Question 1

For which information states (storage, transmission, processing) and for which security services can cryptography be used to protect information?

Answer

Cryptography can be used to protect information when transmitted through unprotected networks or stored in places with insufficient physical or logical access control. Cryptography can provide confidentiality, integrity and authenticity (including non-repudiation) of information.

Question 2

The 4 security services i) 'data confidentiality' ii) 'data integrity' iii) 'data authentication' and iv) 'non-repudiation of origin' are related in the sense that one service often implies/provides another. Indicate for each service which of the 3 other services is/are also provided.

Answer

i. Message confidentiality through secret key encryption strictly speaking does not imply message integrity nor message authentication. However, in case the message contains significant redundancy, then it can be discovered if the message has been changed, which implies integrity and authentication.
ii. Message confidentiality through public-key encryption provides no integrity.
iii. Message integrity through secret key implies message authentication when it is assumed that the MAC (Message Authentication Code) is generated by a secret key known only to the sender and recipient. Under that assumption, message integrity and message authentication are equivalent.
iv. Message integrity implies non-repudiation of origin when it is assumed that the DigSig is generated by a private key known only to the sender. Under that assumption, message integrity and non-repudiation of sender are equivalent.
v. Message authentication with secret key implies message integrity.
vi. Non-repudiation of message origin with private key implies message authentication and message integrity.

Implies→ **Service**	Confidentiality	Integrity	Authentication	Non-repudiation
Confidentiality (secret key)	Y	N (Y)	N (Y)	N
Confidentiality (public key)	Y	N	N	N
Integrity (secret key)	N	Y	Y	N
Integrity (private key)	N	Y	Y	Y
Authentication (secret key)	N	Y	Y	N
Non-repudiation (private key)	N	Y	Y	Y

Question 3

Alice wants to send message *M* to Bob, without Eve observing it. Alice and Bob have agreed to use a symmetric cipher. Key exchange has already been done, and so they share a key *K* for a specific encryption algorithm E.

a. Outline the steps that Alice must follow for encrypting *M* and sending it to Bob.
b. Outline the steps that Bob must follow for decrypting the received ciphertext *C*.

Answer

a. Encryption:
i) Alice prepares the message *M*.
ii) Alice encrypts the message using the symmetric cipher algorithm in encryption mode E with the key *K*, to produce the ciphertext *C*, where $C = E(M,K)$.
iii) Alice transmits the ciphertext message *C* to Bob.

b. Decryption:
i) Bob receives the ciphertext *C*.
ii) Bob decrypts the ciphertext using the symmetric cipher algorithm in decryption mode D with the key *K*, to recover *M*, where $M = D(C,K)$.
iii) Bob reads/interprets the message *M*.

Question 4

Alice wants to send a message *M* with a message authentication code MAC(*M*) to Bob. Alice and Bob share a secret key *k*, and have agreed on using a specific algorithm MACfunc which takes input parameters *M* and *k* to produce MAC(*M*).

a. Outline the steps that Alice must follow for sending *M* .
b. Outline the steps that recipient Bob must follow for verifying the authenticity of *M*.
c. Explain why the MAC proves to Bob that a received message is authentic, and why Bob is **unable** to prove to a third party that the message is authentic.

Answer

a. MAC generation by Alice:
   i. Alice prepares message *M*.
   ii. Alice applies the secure algorithm MACfunc with input parameters *M* and *k* to produce $MAC(M) = MACfunc(M,k)$.
   iii. Alice transmits message *M* and MAC(*M*) to Bob, together with her unique name and specification of the MACfunc algorithm she used.
b. MAC validation by Bob:
   i. Bob receives message *M'* (denoted as *M'*, not *M*, because from Bob's point of view the message origin is still uncertain), as well as MAC(*M*).
   ii. Bob applies MACfunc on *M'* to produce $MAC(M') = MACfunc(M',k)$.
   iii. Bob checks whether MAC(*M*) =? MAC(*M'*). If TRUE, then MAC(*M*) is valid, meaning that $M' = M$. Bob therefore is convinced that Alice really is the sender of message *M*. If FALSE, then the signature MAC(*M*) is invalid, meaning that $M' \neq M$. Bob therefore does not know who created the received message *M'*. He might then decide to reject the message, or alternatively he can use it while knowing that its origin is uncertain.

c. Bob is convinced that Alice sent message *M* because only he and Alice know the secret key *k*, and because he knows that he did not send the message himself. However, Bob is unable to convince anybody else that Alice sent the message, because from the perspective of a third party it is possible that Bob could have sent the message himself.

Question 5

Alice wants to send message *M* with digital signature Sig(*M*) to Bob. They have each other's public keys, and have agreed on a specific hash function h and a signature algorithm that operates in signature mode S (equivalent to Decryption mode D) or in verification mode V (equivalent to Encryption mode E).

a. Outline the steps that Alice must follow for sending *M*.
b. Outline the steps that recipient Bob must follow for verifying the authenticity of *M*.
c. Explain why the digital signature proves to Bob that a received message is authentic, and why Bob is **able** to prove to a third party that the message is authentic.
d. Discuss the semantic interpretation of 'digitally signed', i.e. does it mean that i) Alice agrees with the content of the message, or ii) Alice sent the message without necessarily agreeing to its content ?

Answer

a. Digital signature generation by Alice:
   i. Alice prepares message *M*.
   ii. Alice applies the secure hash algorithm *h* to produce hash value h(*M*).
   iii. Alice uses her private key $K_{priv}(A)$ with the asymmetric algorithm in signature mode S to produce signature Sig(*M*) = S(h(*M*), $K_{priv}(A)$).
   iv. Alice transmits message *M* and signature Sig(*M*) to Bob, together with her unique name and specification of the hash algorithm and the asymmetric algorithm she used.

b. Digital signature validation by Bob:
   v. Bob receives message *M'* (denoted as *M'*, not *M*, because from Bob's point of view the message origin is still uncertain), as well as the signature Sig(*M*).
   vi. Bob applies the secure hash algorithm h on *M'* to produce hash value h(*M'*).
   vii. Bob recovers hash h(*M*) from signature Sig(*M*) by using Alice's public key $K_{pub}(A)$ with the asymmetric algorithm in verify mode V to produce h(*M*) = V(Sig(*M*), $K_{pub}(A)$).
   viii. Bob checks h(*M*) =? h(*M'*). If TRUE, the signature Sig(*M*) is valid, so that *M'* = *M* and Bob is convinced that Alice sent *M*. If FALSE, then Sig(*M*) is invalid, so that *M'* ≠ *M*. Bob therefore does not know who created the received message *M'*. He might then decide to reject the message, or use it knowing that its origin is uncertain.

c. The reason why Bob, and in fact anybody who knows Alice's public key $K_{pub}(A)$, is convinced that Alice sent *M* is that only Alice could have produced the corresponding digital signature Sig(*M*) because it is assumed that only she knows and possesses the private key $K_{priv}(A)$.

d. The semantic interpretation of a digital signature is *a priori* not clear. It could e.g. mean
   i. Alice agrees with the content of the message.
   ii. Alice sent the message but doesn't necessarily agree with its content

In the paper-and-pen world, interpretation (a) is normally assumed. In the digital communications world it is possible to apply interpretation (b) , where the 'message' can be some system transaction where there is no specific 'meaning'.

Question 6

Suppose that a binary additive stream cipher (such as the one time pad) has been used to encrypt an electronic funds transfer. Assuming that no other cryptographic processing is used, show that an attacker can change the amount of the funds transfer without knowing anything about the key used. (You may assume that the attacker knows the format of the plaintext message used for the funds transfer.)

Answer

The part of the message in which the amount is recorded remains in the same position in the ciphertext as in the plaintext. Therefore, the attacker can change the bits in that position which will alter the amount transferred. In general, there is no way that the attacker can know whether the alteration will increase or decrease the value of the amount. In practice the attacker may know that the amount is likely to be small and therefore only change the digits in the high value positions. This illustrates that a binary stream cipher provides no message integrity even though it can provide unconditional confidentiality if a one-time-pad is used.

Question 7

Hash functions are commonly used for checking message integrity.

a. List the four basic requirements of cryptographic hash functions
b. What's the difference between the two variants of collision resistance ?
c. Use the internet to locate a SHA-2 demonstration tool — there's an interactive one written by Geraint Luff that can be found at: http://geraintluff.github.io/sha256/
   Investigate hash function properties by computing SHA-2 hashes for the following:
   (i) Take $100 from my account
   (ii) Take $1000 from my account
   (iii) Take $100 from your account
   (iv) (You can try other hashes for both longer and shorter messages)

Answer

a. The four basic hash function properties are:
   H1: Easy to compute h(*M*).
   H1: Fixed length output for arbitrary length input
   H2: One-way - given h(*M*), it is computationally impossible to find message *M*
   H3: Collision resistant – computationally impossible to find *M* and *M'* so that h(*M*) = h(*M'*)
b. Weak: Given one specific *M,* hard to find *M'*. Strong: Pick any *M*, hard to find *M'*.
c. You should find that even though two input messages are very similar, the output hash values are completely different. Note that for any message length, whether short or long, the hash output length of SHA-2 (SHA-256) is 256 bits, or 64 hexadecimal digits.

Question 8

a. Explain why message authentication alone is insufficient as proof of message origin in general, and to settle disputes about whether messages have been sent.
b. What security service is provided by digital signatures, and explain how this service relates to message authentication.
c. In what order should the signature function and the encryption function be applied? Also explain why that order makes sense.
d. How can a sender give a plausible reason for repudiating a signed message?

Answer

a. Suppose that John sends an authenticated message to Mary. The following disputes that could arise: 1. Mary may forge a different message and claim that it came from John. Mary would simply have to create a message and append an authentication code using the key that John and Mary share. 2. John can deny sending the message. Because it is possible for Mary to forge a message, there is no way to prove that John did in fact send the message.
b. Digital signatures provide non-repudiation, which is stronger than authentication because it provides proof to third parties of message origin.

c. It is important to perform the signature function first and then an outer confidentiality function. In case of dispute, some third party must view the message and its signature. If the signature is calculated on an encrypted message, then the third party also needs access to the decryption key to read the original message. However, if the signature is the inner operation, then the recipient can store the plaintext message and its signature for later use in dispute resolution.
d. The sender can claim that the private signature key was stolen.

Question 9

The Diffie-Hellman key agreement algorithm achieves key agreement by allowing two hosts to create a shared secret.

a. Clearly explain the operation of the Diffie–Hellman key exchange protocol.
b. Clearly explain why the basic Diffie–Hellman protocol does not provide any assurance regarding which other party the protocol is run with.

Answer

a. They share a base *g* and a modulo *m*. Let *a* and *b* be the secret keys of A and B respectively. Then:
   • A → B : g^a (mod m)
   • B → B : g^b (mod m)
   • *A* computes $(g^b)^a = g^{ab}$ (mod m)
   • *B* computes $(g^a)^b = g^{ab}$ (mod m)
   A and *B* now share the symmetric key g^{ab}

b. The basic Diffie-Hellman protocol includes no authentication of the messages exchanged. Therefore, Alice has no assurance that she is running the protocol with Bob and Bob has no assurance that he is running the protocol with Alice.

Question 10

Quantum computing has the potential of making obsolete and insecure most of the standardized asymmetric cryptographic algorithms. Assume the following time periods:

- t_1: the time it takes to standardize quantum-resistant crypto (QRC) algorithms.
- t_2: the time it takes to update crypto applications with standardized QRC algorithms.
- t_3: the time that legacy non-QRC applications must remain operational and secure
- t_4: the time it takes to make large-scale quantum computers practical

a. Express in terms of inequality equations with the terms t_1, t_2, t_3 and t_4 two scenarios:
   Scenario 1 where crypto applications will remain secure
   Scenario 2 where crypto applications will become insecure
b. What is the expected time frame for standardizing quantum-resistant algorithms? See http://csrc.nist.gov/groups/ST/post-quantum-crypto/documents/pqcrypto-2016-presentation.pdf

Answer

a. Scenario 1: $t_1 + t_2 + t_2 < t_4$, Quantum computing becomes available after we have replaced all vulnerable legacy crypto algorithms with quantum resistant algorithms. This scenario is OK
   Scenario 2: $t_1 + t_2 + t_2 > t_4$, Quantum computing becomes available before we have had time to replace vulnerable legacy crypto algorithms with quantum resistant algorithms. This scenario is problematic.
b. From 2018, it will take around 2 years to standardize quantum-resistant algorithms. Hence, we can expect the standards to be ready around 2020.