TTM4135 exam May 2022: Outline answers

1 Multiple choice questions

1 point for correct answer, 0.5 point penalty for incorrect answer. Explanation can say why one answer is correct, **or** why two answers are wrong for 1 point. Possible to get 0.5 point for explanation of why one answer is wrong.

1. Suppose that $3^{-1} \bmod n = 17$. Then a possible value of n is:

   (a) $n = 18$
   (b) $n = 35$
   (c) $n = 50$ ✓

   Explanation: $3 \times 17 \bmod 50 = 51 \bmod 50 = 1$ but $3 \times 17 \bmod 35 = 51 \bmod 35 = 16$ and $3 \times 17 \bmod 18 = 51 \bmod 18 = 15$.

2. Figure A (showing random-looking colours) is the result of an encryption using visual cryptography. You know that figure A is the output of a program. You do not have access to the encryption or decryption keys, but they are encoded into and you can run on as many inputs if you like. You try to find out what the unencrypted figure looked like. You are performing:

   (a) A chosen ciphertext attack
   (b) A chosen plaintext attack ✓
   (c) A ciphertext only attack

   Explanation: Access to the program allows the attacker to choose the plaintext and see the corresponding ciphertext.

3. Ykg xjl'k typo tnt hddyanien xbi dbpy kwgtkibi.P

   The ciphertext above is encrypted with the Hill cipher using trigrams (d=3).

   We decide to mount a brute force attack. How many attempts do we need at most to find the key?

   (a) 26^3
   (b) 26^2
   (c) 26^9 ✓

Explanation: Each of 9 positions can have up to 26 choices. These cannot actually be chosen randomly because the matrix needs to be invertible, but assuming that they can be chosen independently gives an upper bound.

4. Suppose that you have access to two 128-bit keys, K_1 and K_2, shared with another party. You want to use the AES block cipher to provide data confidentiality for many data blocks. Which of the following options would be most secure?

   (a) Use AES-128 double encryption: for each block first encrypt with K_1 and then encrypt the output with K_2.

   (b) Combine K_1 with K_2 by string concatenation to form K_3 and then encrypt all blocks with AES-256 using K_3. ✓

   (c) Alternate AES-128 block encryption with K_1 and K_2 as follows: encrypt block 1 using K_1, encrypt block 2 using K_2, encrypt block 3 using K_1, and so on.

   Explanation: Using AES-256 is by far the strongest since there is no known attack using less than around 2^{256} operations. Double encryption can be attacked with meet-in-the middle. Alternating keys requires only AES-128 to be broken twice, with effort not more than 2^{129} operations.

5. Suppose that an active attacker observes a ciphertext of 10 blocks which was encrypted with a block cipher using some mode of operation. The attacker also knows the exact plaintext which was encrypted. The attacker wants to change the ciphertext to ensure that a specific bit in the fifth block will be flipped after decryption. Which of the following modes of operation makes this task hard for the attacker?

   (a) ECB mode ✓

   (b) CBC mode

   (c) CTR mode

   Explanation: In CBC mode flipping a bit in one ciphertext block leads to the same bit in the following block to be flipped in the decrypted plaintext. In CTR mode flipping a bit in one ciphertext block leads to the same bit in the same block to be flipped in the decrypted plaintext. In ECB mode there is no obvious way to control how changes to the ciphertext affect the decrypted block in a specific position.

6. Why do we use *salting* when we store a password in a database?

   (a) To make online password guessing harder.

   (b) To make dictionary attacks impossible, if the database is leaked. ✓

   (c) Because passwords should never be stored in plaintext.

   Explanation: You cannot use a dictionary containing hashes of common passwords, since these hashes will not contain the salt. Online guessing it not affected by how the hashed password is stored.

7. 8911 is a Carmichael number. There are several methods to test numbers for primality. What is most likely to happen when we use them on the number 8911?

(a) The Miller-Rabin test outputs that 8911 is a prime, but the Fermat test disagrees

(b) The Fermat test outputs that 8911 is a prime, but the Miller-Rabin test disagrees ✓

(c) The Miller-Rabin test and the Fermat test both output that 8911 is not a prime

Explanation: The Miller-Rabin test never does worse than the Fermat test. Since 8911 is composite, if Miller-Rabin says "prime" then Fermat will also do so. Since 8911 is a Charmichael number, Fermat will output "prime" in most cases.

8. RSA encryption of a message M makes use of a public exponent e, a private exponent d and a modulus n. To ensure that encryption and decryption work properly it must be true that:

(a) $\gcd(M, d) = 1$

(b) $\gcd(\phi(n), d) = 1$ ✓

(c) $\gcd(M, \phi(n)) = 1$

Explanation: The private exponent d is the inverse of the public exponent e so it must be invertible mod $\phi(n)$. All values for M between 1 and $n - 1$ can be used.

9. OAEP is a coding algorithm often used together with RSA encryption. Using OAEP helps to:

(a) prevent attacks based on deterministic encryption ✓

(b) allow longer messages to be encrypted

(c) speed up decryption

Explanation: OAEP encodes the input message including a random field, thus it ensures that encryption is randomised.

10. Breaking an elliptic curve-based Diffie-Hellman instantiation with curve group size p of length 256 bits is around as hard as breaking 128-bit AES — this is currently considered secure. Why are we currently investing so much into developing new cryptographic algorithms if we can just increase the group size p to length 512 bits instead?

(a) We would have to keep doubling the group size every other year, as Moore's law dictates computers will keep speeding up as well.

(b) Elliptic curve operations are always linear in the length of p, so doubling the group size only has a small effect on brute force key search.

(c) Because our adversaries will have access to a quantum computer in the future, and with those you can break ECDH efficiently. ✓

Explanation: Shor's algorithm allows a quantum computer to efficiently solve the discrete logarithm problem, thus ensuring that an attacker can successfully attack with any reasonable sized group. Thus we need an alternative to replace algorithms relying on the difficulty of the elliptic curve discrete log problem.

11. You are designing an IoT system which will turn your house's central heating on and off based on the outside temperature. Since it's IoT, everything except the server is battery-powered, and performing computations and sending data cost a lot of power.

    You want to implement some security features — the temperature outside is not a secret, but you want to be sure the correct value is received, because otherwise an attacker could control your heating and increase your power bill. What do you use to protect the values you send from the thermometer to your server?

    (a) HMAC ✓
    (b) ECDSA
    (c) Chacha

    Explanation: Chacha does not provide integrity so does not provide what we need. Although ECDSA could be used to provide integrity of the data, HMAC is much more efficient so a better choice in this case.

12. Which of the following protocol features is shared by both the Kerberos protocol and the TLS 1.2 handshake protocol?

    (a) Forward secrecy is always provided
    (b) Knowledge of one session key does not compromise other session keys ✓
    (c) Clients need to check the validity of the communication partner's long-term key

    Explanation: Kerberos uses symmetric long-term keys shared with the server which cannot be checked by other clients. Kerberos does not provide forward secrecy. However, in both cases session key are generated independently of each other.

13. Two possible variants of the handshake protocol in TLS 1.2 are based on (i) RSA encryption and (ii) elliptic curve Diffie-Hellman (ECDH). The advantage of using the ECDH variant is:

    (a) the TLS server key exchange message is shorter
    (b) the handshake protocol is secure against quantum computers
    (c) forward secrecy for the session is provided ✓

    Explanation: The server key exchange message is empty in the RSA variant. The EC discrete log problem can be solved efficiently by quantum computers. **Note that in cases where the explanation assumed that the ECDH uses static keys, that was also given credit.**

14. Why does TLS 1.3 remove support for non-AEAD cipher suites?

    (a) Because renegotiating the cipher suite used during a TLS connection is an attack vector
    (b) This reduces the amount of handshake messages since AEAD ciphersuites are more suitable for 0-RTT
    (c) Because non-authenticated information in the header fields is a security risk ✓

Explanation: Ciphersuites are still negotiated in the handshake protocol in TLS 1.3. The amount of handshake messages is independent of using AEAD ciphersuites. In TLS 1.3 the full header is protected as additional data in the AEAD algorithm.

15. The Double Ratchet from the Signal protocol consists of two ratcheting mechanisms. Why is one ratchet based on Diffie-Hellman not sufficient?

    (a) With one ratchet we can't obtain forward secrecy when there are consecutive messages from the same party. ✓

    (b) We need two ratchets two obtain both authentication (through signatures) and secrecy (through encryption).

    (c) With one ratchet we can't support group operations in as well as in elliptic curve groups.

Explanation: There is no new Diffie-Hellman share available when two consecutive messages are in the same direction. Thus to achieve forward secrecy for those messages the key must be evolved in a different way, namely by hashing.

2 Written answer questions

1. **Autokey cipher** The Autokey cipher is a classical cipher invented in 1586. We start off by encrypting as we do with the Vigenère cipher, but instead of repeating the keyword, we use the plaintext as the key as follows.

   Given the plaintext $p_0, p_1, \ldots$ and a key consisting of characters $k_0 \ldots k_{19}$, this means we compute the ciphertext as:

   $$c_i = \begin{cases} p_i + k_i \bmod 26 & \text{for } 0 \leq i < 20 \\ p_i + p_{i-20} \bmod 26 & \text{for } i \geq 20 \end{cases}$$

   1. What is the key space for the Vigenère cipher? What is the key space for the Autokey cipher?
   2. Do you consider this cipher to be more or less secure, compared to the Vigenère cipher?
   3. What would your attack strategy be, using well-known techniques such as those from the practical assignment?

   1. The keyspaces for Vigenère and Autokey are the same: 26^{20} possibilities.
   2. Autokey is more secure, since the attacks on Vigenère are very easy given the periodicity. You essentially do 20 "ceasar cipher"-analyses on each set of ciphertext characters i, i+20, i+40, ... Attacking Autokey is more cumbersome.
   3. Assuming the text is in English and long enough, there are several possibilities we can use. Observe that the periodicity from Vigenère is gone now, so we can't use frequency analyses etc like we did in the practical.
      One possible method is as follows. Find a common word you know is probably in the plaintext (THE, LIKE, etc) and apply it to every possible word, if the resulting d-gram looks like it could be English, apply it as keystream 20 characters and see if the result also looks like it could be English. Then you can pencil it in and continue.
      You can also some sort of automated brute force which lets you know if English words appear in the plaintext and stores these possibilities underway
      Combining the two above (hand-assisted) is probably the best strategy. For all of these possibilities it holds that it is quite cumbersome to solve by hand, since changing your guess for k_1 changes the p_1-guess used to compute p_{21}, which influences p_{41} etc, so no matter what you do you want to automate (parts of) the process.

2. **Feistel ciphers** Consider a Feistel cipher with a 128-bit block size, a 128-bit key, and 16 rounds. Each round uses the Feistel construction:

   $$\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \oplus f(R_{i-1}, K_i) \end{aligned}$$

   1. Explain, with justification, what should be the length in bits of outputs from the function f.

      The output should be half the block length, so 64 bits. The length has to match with L_{i-1}.

2. What are the possible values of the length in bits of each K_i? Of the *possible* lengths, suggest what might be a reasonable choice for a practical block cipher.

   The round keys K_i are determined by the key schedule and any integer value is possible in order to make the cipher functional. Since there are 128 bits of randomness in the key, it does not make sense for the round keys to be shorter than $128/16 = 4$ bits. From what we have seen with DES it seems a reasonable choice to make the round keys smaller than the full key but larger than half a block.

3. Suppose that the function f is chosen to ignore its first input, so that we can write simply $f(R_{i-1}, K_i) = f(K_i)$. Explain why the cipher is now easy to break.

   With this choice of f, each round just adds a constant to the left and right halves. For example,

$$\begin{aligned} L_4 &= R_3 \\ &= L_2 \oplus f(K_2) \\ &= R_1 \oplus f(K_2) \\ &= L_0 \oplus f(K_0) \oplus f(K_2) \end{aligned}$$

   So the final output, $(L_{16}, R_{16}) = (L_0 \oplus C_L, R_0 \oplus C_R)$ for some constant strings C_L and C_R. These constants are easily found with a known plaintext attack.

3. Exponentiation Suppose that m is an integer of 100 bits in length (so $2^{99} \le m < 2^{100}$). For some algorithm, suppose that we need to compute the exponentiation

$$x^e \bmod m$$

for some value x of less than 100 bits in length.

1. If m is a prime number, explain why we can always assume that e is also of no more than 100 bits when making the computation.

   By Fermat's theorem, $x^e \bmod m = x^{e \bmod m-1} \bmod m$, so we can reduce e modulo $m-1$ before starting the computation.

2. If the square-and-multiply algorithm is used, what is the maximum number of multiplications needed? (You may assume that a squaring is the same cost as a multiplication.) Explain how you reach your answer.

   The number of squarings needed is determined by the length of e in bits: we must keep squaring i times until we get a value $2^{2^i} \bmod m$ which is at least half the size of e. The largest value of e so 100 bits so the largest value of i is 99. The number of multiplications between squaring is determined by the number of 1 bits in e. If all these bits are set then there are 99 multiplications. So total is 198. **Note: Although it is natural to include both squarings and multiplications in this computations, credit was also given if the squarings were omitted.**

3. Suppose now that $m = pq$ where p and q are primes of 50 bits each. If the Chinese Remainder Theorem (CRT) is applied for the computation, two exponentiations are computed. Show that the maximum number of multiplications is almost the same as the previous part of this question when the square-and-multiply algorithm is used. So why is the CRT useful?

   Now the number of multiplications will be determined by two exponents of 50 bits each, so the maximum is $2 * (49 + 49) = 196$ plus what is needed for CRT. However, these multiplications are done with a modulus which is half the size of m which requires 25% of the computations with the full modulus. So CRT still is much quicker (even with the computations for the CRT reconstruction).

4. **Corona certificates** During the fall of 2021, many European countries including Norway made use of "corona passes" or "corona certificates". In a simplified form, national authorities would sign the phrase "Person X, born Y has been vaccinated on date Z" using a government private key. This text, together with the signature, would then be turned into a QR-code which could be scanned and verified using that government's public key.

   (Picture of hierarchical certification tree omitted.)

   Eva works as a bouncer at a Trondheim night club, scanning QR-codes using the Norwegian scanning app. The app comes pre-loaded with all the public keys belonging to the Norwegian health authorities.

   a) Isak got his corona shots from the Oslo municipality. Eva scans the code and confirms that the certificate is valid. What steps does her phone perform to verify this?

   b) Even has a corona certificate from Germany. Eva scans the code and confirms that the certificate is valid. What steps does her phone perform to verify this?

   In October 2021 the private keys belonging to the Polish government were leaked, and valid QR-codes belonging to fictitious people such as "Mickey Mouse" (born 1900) and "Sponge Bob" (born 2001) started to appear online, as well as a black market for fake yet valid QR-codes.

   William buys a fake certificate on the internet for $300. The advertisement promises that the certificate will register as valid, which he verifies himself.

   c) A week after purchase, William gets his QR-code scanned by Eva and is denied access. How could Eva's phone possibly know that the code was a fake?

a. The QR-code scanned will contain m and σ. The public keys belonging to the Oslo municipality is already loaded into Eva's phone, so the phone will compute Verify(m, σ, key) and turns the screen green if this outputs true.

b. Same as above, but the phone will observe that the signature is signed with a key it does not possess. It will then either get the key from FHI if it is cached there, get the key from the EU authority if it is cached there, get it forwarded through any of these servers, or get it directly from BMG or one of its possible sub-authorities. All of these assumptions can be correct as long as they are explained well in the answer. Essential for a good answer are that 1) the certificate is verified, so the signatures are verified all the way up until the EU certificate. It fails if this somehow goes wrong. 2) only THEN we can use the key to verify the signature 3) the phone will somehow need to be sure that the key belonging to the top-level authority is authentic, for example it should be hardcoded into the phone app or FHI should somehow show this.

c. There is a certificate revocation list. The compromised key was put on this list, so now its signed certificates are not valid anymore. Discuss how the phone got to this list / who maintains it (different assumptions are possible — the polish authority can have revoked the certificated of a sub-authority, the EU can have revoked all of Poland, FHI can have put one of them on a blacklist etc). Remark that it's easy to believe the certificate expired, or that a relatively short certificate expiration date was chosen for security reasons, but most likely not a good answer. Certificates with a short expiration date are not useful within the simplified system of this exercise (unless you would get re-vaccinated every week or so). This could only work if you somehow have a database of vaccinated people that you re-generate everyone's QR-codes from on the regular.

5. **TLS Handshake** The TLS 1.2 handshake protocol allows a client and server to agree upon various parameters to be used in both the handshake and record protocols of TLS.

   1. How could a client force the server to accept the weakest ciphersuite that the server supports?

      Since the server will accept the best proposal from the client, the client could only send the weakest ciphersuite which it knows the server will accept.

   2. What will happen in the handshake protocol if the client and server do not share a ciphersuite that they both support?

      The client is allowed to make a new proposal, but in the end if they do not support a shared ciphertsuite there can be no secure connection established.

   3. What prevents an active attacker from forcing a client and server to use an older version of TLS, when there is a newer one that they both support? The client starts by offering its best version of TLS. The version number is included in all handshake messages. If the server does not support that version it should reply with a lower version which the client can accept. However, if an active attacker changes the client proposal that will fail because the finished messages include a hash of all the handshake messages and are authenticated. Another possible attacker strategy is to interrupt the communications so that some clients will automatically try a lower protocol version - this is the first step in the so-called POODLE attack. This should not really happen, but can be prevented by having clients signal that they are re-negotiating by trying a lower version (called SCSV).

6. **X3DH** The Extended Triple Diffie-Hellman protocol (X3DH) is the protocol Signal uses to initialize a conversation between two parties. Signal is a privacy-centered system, so by design anyone can create an account with Signal without uploading any proof of identity. When creating an account, you upload a public identity key (IK) and a public pre-key (SPK) to the server and hold on to the private keys that correspond to these. IK is static in the long term, while SPK is replaced with a new one every week or so.

   When Bob wants to talk to Alice, he gets Alice's public identity-key $\mathsf{IK_A}$ and a public pre-key $\mathsf{SPK_A}$ from the server. He then generates an ephemeral keypair $\mathsf{EK_B}$ and computes a shared secret SK based on four keys: the public keys $\mathsf{IK_A}$ and $\mathsf{SPK_A}$, and the private keys corresponding to $\mathsf{IK_B}$ and $\mathsf{EK_B}$.

   Since Bob sends Alice the public keys $\mathsf{IK_B}$ and $\mathsf{EK_B}$ along with his first encrypted message, and she still has the private keys corresponding to $\mathsf{IK_A}$ and $\mathsf{SPK_A}$, she can compute SK as well.

   1. What security property/properties does Bob achieve by computing a new $\mathsf{EK_B}$ every time, given that he already uses his own identity key $\mathsf{IK_B}$?

      It mitigates against replay attacks, now the initial message to Alice will look different every time even if it has been sent using the same SPK_A, IK_A and IK_B. There is also an aspect of forward secrecy: Bob can delete the secret key belonging to EK_B, as the signal ratchet will negotiate new keys for every follow-up message in the conversation.

   2. An adversary Charlie manages to take over the Signal servers. Charlie replaces $\mathsf{IK_A}$ with $\mathsf{IK_C}$ and $\mathsf{SPK_A}$ with $\mathsf{SPK_C}$, before Bob starts his conversation with Alice.

What effect does this have on the security of the protocol? How can Bob be sure that he is actually talking to Alice, instead of Charlie?

Essentially nothing has changed, since there was no verification of Alice's identity when she originally registered. If Bob wants to be sure he is talking to Alice he should verify that they both have the same SK — either by exchanging a hash of SK using some trusted, authenticated channel, or by meeting up offline and comparing the keys that are shown in the app.