NATIONAL UNIVERSITY OF SINGAPORE

CS2107 – Introduction to Information Security

(AY2020/21 Semester 1)

Mid-Term Quiz

Date: 2 Oct 2020 Time: 10:15 - 11:35AM

STUDENT NUMBER :	A								

NAME :

INSTRUCTIONS TO CANDIDATES

1. This question paper consists of **NINETEEN (19)** questions in **THREE (3)** parts; and comprises **EIGHT (8)** printed pages, including this page.
2. Fill in your Student Number and Name above with a pen.
3. This mid-term quiz has **30 marks**, and is worth **15%** of your final mark.
4. Answer **ALL** questions.
5. You may use pen or pencil to write your answers, but please erase cleanly, and write legibly. Marks may be deducted for illegible handwriting.
6. Write your answers on this **question paper**.
7. This is an **OPEN BOOK** assessment.
8. You are allowed to use **NUS APPROVED CALCULATORS**. Yet, you should be able to work out the answers without using a calculator.

Part A (5 marks): Multiple Choice Questions

Instructions: Choose the ***best answer***, and circle/cross the corresponding letter choice below. No mark is deducted for wrong answers.

A1. In Microsoft's STRIDE security threat model, two types of threat are *tampering with data* and *spoofing of user identity*. Which security requirements are compromised by them, respectively?

a) Confidentiality and Integrity
b) Integrity and Confidentiality
c) Integrity and Authenticity
d) Authenticity and Integrity
e) Availability and Non-repudiation

A2. Which of the following statements about classical ciphers is *incorrect*?

a) Substitution cipher has a smaller key space size than Shift cipher's
b) Shift cipher is a monoalphabetic cipher
c) Vigenere Cipher is not a monoalphabetic cipher
d) One-time pad's ciphertext leaks no additional information at all about the corresponding plaintext
e) Permutation cipher is insecure in both known-plaintext and ciphertext-only attacks

A3. Suppose a fingerprint recognition system (FRS) is deployed to protect the door access to a company's lounge (with sofas and newspapers inside). Access to this lounge should be *more accepting* as it is considered a common area for all staff members. Nonetheless, *only* staff members can access it. Which policy should be enforced on the lounge door's FRS?

a) Its threshold should be set to EER
b) Its threshold should be set to 1
c) Its threshold should be set to 0
d) Its rejected genuine matches over all attempted genuine matches must be low
e) Its successful false matches over all attempted false matches must be low

(**Additional Note**:

First of all, you should see that option d is a correct answer.

The above-mentioned constraint basically says that the threshold cannot be 0 (i.e. anyone can go in). However, it does not mandate that there must really be a low successful false match rate when it is not attainable (more about this below).

Of course, ideally, the objectives should both d and e. However, setting the threshold is a trade-off between achieving d and e. These two conflicting objectives are diagrammatically shown on Slide our 67. Given the constraint, we should then prioritize a low FNMR (option d), but not with the threshold set to 0. We can still afford a slightly high FMR, i.e. successful false match rate, as no sensitive asset items are in the lounge as described in the problem. Part A on MCQs asks you to select the best answer. Hence, given the problem constraints and reasoning above, the available option d is clearly the one and should be selected.)

A4. The success of a Padding Oracle attack by an attacker as discussed in the class relies on the following conditions, *except*:

a) The use of CBC mode-of-operation in encrypting a target plaintext
b) The attacker's accessibility to a padding oracle, which works as a decryption oracle
c) The oracle's ability in recognising that the ciphertexts sent by the attacker do not contain a proper padding
d) The attacker's ability in supplying modified IVs or ciphertext portions in its queries to the oracle
e) The attacker's ability to receive an error message from the oracle, if the padding of a plaintext recovered by the oracle is incorrect

A5. In RSA, which task below is computationally *difficult*? (Note on the notation used: n is the RSA modulus; p and q are both the modulus' prime factors):

a) Given p and q, compute n
b) Given n and p, compute q
c) Given n and q, compute p
d) Given p and q, compute $\varphi(n)$
e) Given n, compute $\varphi(n)$

Part B (10 marks): Security Terminology

Instructions:

The next ten questions (B1 to B10) give security-related descriptions, which are taken from various articles/writings on the Internet. Below is a list of security terms. Fill in the blanks in the next ten questions with the ***most appropriate*** terms from the list. Put only one choice per blank. You may ignore any grammatical rules on plural forms. Note that it is possible for some choices to appear more than once in your answers in this part.

Cryptography Objects:
Block cipher
Stream cipher
Initial Value (IV)
Pseudo random sequence
One-time pad
Symmetric key
Public key
Private key
Signature
Certificate
Self-signed certificate
Certification Authority
Certification path
Hash

Cryptography Notions:
Symmetric Key Cryptography
Public Key Cryptography
RSA scheme
Public Key Infrastructure
Kerckhoffs' principle
One way

Attacks:
Denial of Service
Man-in-the-middle
Chosen-plaintext
Known-plaintext
Frequency analysis
Brute-force

Miscellaneous:
2FA
Covert channel
Bring-your-own-device
Botnet
Worm

MAC	Side-channel
Authenticated encryption	Phishing
Nonce	Skimming
Mode-of-operation	Birthday

B1. $\boxed{\text{Kerckhoffs's principle}}$ says that one ought to design systems under the assumption that the enemy will immediately gain full familiarity with them.

B2. $\boxed{\text{Mode-of-operation}}$ describes how to repeatedly apply a cipher's single-block operation to securely transform amounts of data larger than a block.

B3. To randomize the encryption and hence to produce distinct ciphertexts even if the same plaintext is encrypted multiple times, several modes use a block of bits known as $\boxed{\textit{Initial Value (IV)} \mid \text{Nonce}}$.

B4. A/an $\boxed{\text{MAC}}$, also known as a tag, is a short piece of information derived from a secret key which is used to authenticate a message,
i.e. to confirm that the message came from the stated sender and has not been changed.

B5. $\boxed{\text{Stream cipher}}$ can be viewed as approximating the action of a proven unbreakable cipher, the one-time pad.

B6. Nuke attack is a/an $\boxed{\text{Denial of Service}}$ attack, which makes use of fragmented or invalid ICMP packets sent to a target computer in order to slow down the computer until it comes to a complete stop.

B7. A/an $\boxed{\textit{Skimming} \mid \text{Phishing}}$ attack is the theft of personal information having used in an otherwise a normal transaction.

B8. A/An $\boxed{\text{Certification Authority}}$ acts as a third party trusted both by the subject (owner) of a certificate and by the parties relying upon the certificate.

B9. In a/an $\boxed{\text{Man-in-the-middle}}$ attack, the attacker secretly relays and possibly alters the communications between two parties who believe that they are directly communicating with each other.

B10. A valid $\boxed{\text{Signature}}$ of a message, which is generated using asymmetric cryptography, gives a recipient very strong reason to believe that the message was created by a known sender, and that the message was not altered in transit.

Part C (15 marks): Structured Questions

Instructions: Write your answers in the spaces provided. For questions that require calculations, please **show your workings sufficiently**. Giving correct answers without any workings shown will give you *partial* marks only!

C1. *Lucky Length for Encryption* (3 marks)

Bob likes the number 88, which he views as his lucky number. He wants to use an **88-bit** key for a secret-key based encryption that he develops.

a) (1 mark) What is the **key space size** of Bob's encryption scheme?

The key space size is: $\mathbf{2^{88}}$

b) (2 marks) Suppose it takes 1,024 clock cycles to test whether an 88-bit encryption key is correct, when given an 88-bit plaintext and its corresponding ciphertext. How long does it take to **exhaustively check** all the keys using a 4GHz ***dual***-core processor?
(***Hint***: For simplicity, you can take 1 year ≈ 2^{25} seconds.
Also note that: 1K = 2^{10}, 1M = 2^{20}, 1G = 2^{30}, 1T = 2^{40}.)

Notice that a 4GHz dual-core processor has:
$2^1 . 2^2 . 2^{30} = 2^{33}$ cycles per second.

From the problem description, testing 1 key takes 1,024 = 2^{10} cycles.
In 1 second, the processor can thus check $2^{33} / 2^{10} = 2^{23}$ keys.

To check all 2^{88} keys, the processor needs $2^{88} / 2^{23} = \mathbf{2^{65}}$ **seconds**.
Since 1 year $\approx 2^{25}$ seconds, the total time needed is therefore:
$2^{65} / 2^{25} \approx 2^{40} \approx$ **1 Tera years**.

C2. *Lucky Length for Hash Function* (4 marks)

Since Bob likes the number 88, he wants to design a hash function that produces an **88-bit digest**. Alice, however, warns that the hash function provides a much weaker actual bit security. Alice wants to show the feasibility of creating a **collision** on Bob's hash function.

a) (2 marks) In generating message digests, how many randomly-generated messages should Alice produce so that, with a probability of more than 0.5, she has **two messages with the same digest?** Show your working clearly and succinctly.
Note: You can use some approximation that can help simplify your calculation.

Let's apply the *standard* birthday attack formula.
From the problem description, we know that T = 2^{88}.
To have a probability more than 0.5 that a collision occurs,
we just need to find *M* that satisfies the inequaility condition: $M > 1.17 \,.\, \sqrt{T}$.

We can just easily set *M* to: $2 \,.\, 2^{88/2} = 2^1 \,.\, 2^{44} = \mathbf{2^{45}}$.

b) (2 marks) Suppose it takes 512 clock cycles to generate a digest using Bob's hash function. How long will a **cluster of 1,024 servers**, each with a ***quad***-core 4Ghz processor, take to produce a collision based on your answer in Part a of this question? You can express your answer in seconds or hours.
(***Hint***: For simplicity, you can take 1 year ≈ 2^{25} seconds, 1 hour ≈ 2^{12} seconds.
Also note that: 1K = 2^{10}, 1M = 2^{20}, 1G = 2^{30}, 1T = 2^{40}.)

Given 1,024 servers, each with a quad-core processor, we thus have:
$1024 \,.\, 4 = 2^{10} \,.\, 2^2 = 2^{12}$ processors.

Notice that a 4GHz (single-core) processor has $2^2 \,.\, 2^{30} = 2^{32}$ cycles per second.
In total, the cluster of servers thus have $2^{12} \,.\, 2^{32} = 2^{44}$ cycles per second.

From the problem description, generating 1 digest takes $512 = 2^9$ cycles.
In 1 second, the cluster of servers can thus generate $2^{44} / 2^9 = 2^{35}$ digests.

To generate all 2^{45} digests (from answer to Part a), the cluster of servers need:
$2^{45} / 2^{35} =$ **2^{10} seconds**.
Since 1 hour $\approx 2^{12}$ seconds, the total time needed is therefore:
$2^{10} / 2^{12} \approx 2^{-2} \approx$ **1/4 hour**.

C3. *RSA: Numbers of RSA, and O$P3$* (5 marks)

a) (3 marks) To test your understanding of how (the classroom) RSA works, do answer the three given questions (i-iii) below.
Suppose the two prime factors used are p = 11 and q = 17.
(i) What is the RSA modulus n?
(iii) What is $\varphi(n)$?
(iii) Suppose e=3, which among the following possible values should be d:
23, 67, 107, or 121?

(i) The RSA modulus $n = p \,.\, q$ = 11 , 17 = 187

(ii) $\varphi(n) = (p\text{-}1)(q\text{-}1)$ = (11-1)(17-1) = 10 . 16 = 160

(iii) d is 107 since 3 . 107 = 1 (mod 160)

b) (2 marks) Alice wants to use the classroom RSA scheme to **message Bob** without any PKCS paddings or authenticity-related measures incorporated. She encrypts a number m that represents the amount of money Bob needs to transfer.
Her generated **ciphertext** is: $c = m^e$ (mod n), with (n, e) as Bob's RSA public key.

Mallory is Alice's friend, who is not so friendly towards Bob. Mallory is a **man-in-the-middle**, who can intercept c from Alice, modify it into c', and then send c' to Bob. Now, Mallory wants to triple m, i.e. turn m into $3 \times m$, in c', but without knowing m. What should Mallory set c' to? Explain briefly why your c' should work.
Note: The number m is an integer, and for simplicity in this case is $1 < m < 10{,}000$.

Mallory can set c' for Bob to: $\mathbf{3^e \,.\, m^e}$ **(mod *n*)**, which is basically $(3 \,.\, m)^e$ (mod n).

When Bob recovers the plaintext, it will be:
$(3 \,.\, m)^{e^d}$ = $(3.\, m)^{ed}$ = $3 \,.\, m$ (mod n).

C4. *Attackable OTP?* (3 marks)

Bob really likes **One-Time Pad** (OTP), which does achieve perfect security. Bob thinks that he should be able to use OTP by itself for a secure message communication.

Suppose Bob's OTP keys are random and always fresh as required. His plaintexts, however, always start with "`From: Bob`" string, and this is known by Mallory. Mallory is a **man-in-the-middle**, who can intercept Bob's ciphertexts, modify them, and then relay the modified ciphertexts to the respective receivers.

Suppose now Mallory wants to modify all Bob's OTP ciphertexts so that, when decrypted by their respective receivers using correct keys, the recovered plaintexts start with "`From: Bot`" instead. What should Mallory turn **each OTP ciphertext** from Bob into? Explain briefly why your attack works.

(***Note***: Suppose the two relevant characters are encoded using their following ASCII-based binary strings: '`b`' → 0110 0010, '`t`' → 0111 0100.)

Let's consider that the ciphertext (*c*) and key (*k*) byte strings below are **zero**-based. To launch her attack, Mallory can just replace **the 8-th byte** of Bob's OTP ciphertexts, *c*[8], with ***c'*[8] = *c*[8] ⊕ b ⊕ t**. In other words, Mallory can just **XOR** *c*[8] with: `b` ⊕ `t` = 0110 0010 ⊕ 0111 0100 = **0001 0110**.

To see how this attack works, let's consider the decryption done by a receiver. Notice that the receiver of an authentic ciphertext *c* from Bob will perform the following using its key *k*: *c*[8] ⊕ *k*[8] = `b`. As such, ***k*[8]** = *c*[8] ⊕ `b`. Therefore, when the receiver decrypts *c'* containing the modified *c'*[8], the recovered ***m'*[8]** is: *c'*[8] ⊕ *k*[8] = *c*[8] ⊕ `b` ⊕ `t` ⊕ *c*[8] ⊕ `b` = `t` as intended.

Alternatively, notice that XOR-ing a bit with 1 does flip the bit, i.e. 1→0 and 0→1. Performing ***c'*[8] = *c*[8] ⊕ 0001 0110** above is thus equivalent to flipping the 3 bits of *c*[8] at index 1, 2 and 4.
As mentioned in the lecture, flipping a particular bit of an OTP ciphertext will flip the corresponding decrypted bit. Hence, you can also see that, by performing the 3-bit flipping, the '`b`' will become '`t`' in the recovered plaintext, i.e. *m'*[8] = `t`.

Below is also another alternative answer. First, notice that Mallory is in the known-plaintext attack scenario w.r.t. the first 9 bytes of the messages.
As such, given a ciphertext, Mallory can easily determine the first 9 bytes of the key being used by Bob: ***k*[0..8] = *c*[0..8] ⊕ "`From: Bob`"**.
Given *k*[0..8], Mallory can then derive the 9-byte prefix of the modified ciphertext as follows: ***c'*[0..8] = *k*[0..8] ⊕ "`From: Bot`"**.
In fact, the operation ***c'*[8] = *c*[8] ⊕ b ⊕ t** above essentially first recover ***k*[8] = *c*[8] ⊕ b**, and then set ***c'*[8] = *k*[8] ⊕ t** that will give *m'*[8] = `t` as required.

**~~~ END OF PAPER ~~~**