UNIVERSITY OF BRISTOL

January 2021

Faculty of Engineering

Examination for the Degrees
of
Bachelor of Engineering
Master of Engineering

COMS30021(J)
Cryptology

TIME ALLOWED:
2 Hours

This paper contains 6 questions over 11 pages.
Answer all the questions.
The maximum for this paper is 100 marks.

Other Instructions

1. **This is an open book exam.**
2. **Automated and programmable computing devices are permitted.**
3. **After completion of this exam, you will have 15 minutes to upload your submission to Blackboard.**

Preamble

This exam is composed of 6 questions, *all* of which you must answer:

- 2 regarding *symmetric cryptography*;

Question	Points
Symmetric Cryptography — MCQs	15
Hashing Passwords	20
Total:	35

- 1 at the interface between symmetric and asymmetric cryptography; and

Question	Points
Hybrid Constructions, Practical Cryptography	25
Total:	25

- 3 regarding *asymmetric cryptography*.

Question	Points
Asymmetric Cryptography — MCQs	6
Cryptography in Prime-Order Fields	12
Cryptography in Extension Fields	22
Total:	40

Use of calculators. You are free to use a computer or calculator throughout, but *must* show your working where specified. Wolfram Alpha[1] is sufficient for most of the questions in this exam (at least those that are made easier by having access to a calculator), but feel free to use other tools.

Open Book and Referencing. This is an open book exam conducted online. If you reference external material (material that we did not provide during the course of the unit), you *must* include clear references, in line with the University's academic integrity policy.

[1] https://www.wolframalpha.com/

Marking MCQs. Multiple Choice Questions (in Questions 1 and 4) may have 0 to 4 correct answers. For each question, marking starts with full marks, and a mark is removed for each incorrect classification (each invalid answer selected, and each valid answer missed), down to a minimum of 0 marks. If you believe none of the proposed answers are valid, you *must* indicate so with "None" or some other way of noting that you have seen the question and made the conscious choice of not marking any answers as valid.

Marking Scale. Partial marks will be given for answers that demonstrate general understanding but get details wrong (or forget them). In general (and where possible without fractional marks), getting 50% of the way to a full answer should net you roughly 70% of the marks. Effort beyond that will offer diminishing returns, so plan your work accordingly, and give yourself space and time to iterate on complex questions.

Q1 — Symmetric Cryptography — MCQs [15 marks]

Please recall the rules for marking MCQs stated in the preamble.

[3 marks] **1.a)** Alice and Bob share an AES key known only to them. They have never used it in the past. Alice wants to encrypt a single file to send to Bob. The file contains 2.4 Terabytes of Pokemon pictures.

A. Alice can securely send the entire file in one message using AES in CTR mode.

B. Alice can securely send the entire file using AES in ECB mode.

C. Using CTR mode does not require a nonce in this scenario, if Alice does not want to keep communicating with Bob.

D. The file is too large to send securely in a single message, regardless of which mode is used.

1.b) Your favourite encryption scheme Enc (which uses 128-bit keys) is broken! It now takes only 2^{50} encryptions, given a plaintext-ciphertext pair, to retrieve the key. The next two choices refer to this scenario.

[3 marks] i. The attack is

A. A key recovery attack.

B. A known ciphertext attack.

C. An exhaustive search.

D. A preimage attack.

[3 marks] ii. You generate two keys k_1 and k_2 and encrypt the password p to your hard drive as $c = \mathsf{Enc}_{k_2}(\mathsf{Enc}_{k_1}(p))$. Your cat walks across your keyboard and deletes both k_1 and k_2. Using the best known attack, how long (approximately) would it take to retrieve them given c, p, and access to a machine that can perform 2^{32} encryptions per second.

A. A day.

B. A week.

C. A year.

D. More than a century.

[3 marks] **1.c)** Santa keeps the naughty list in a database on a computer. The list is kept as a list of entries of the form (n, t), where n is a name, and t some additional tag. A known naughty elf has unrestricted access to the computer, and Santa believes he can use the tag t to prevent the elf from removing his name from the naughty list. How can Santa use t to prevent the naughty elf from removing his name from the database?

A. Compute t as an encryption of n with AES in CTR mode.

B. Compute t as a MAC tag for n, computed using CMAC.

C. Compute t as a digest of n, computed using SHA-256.

D. No purely cryptographic solution can be used.

[3 marks] **1.d)** Consider the set $\mathcal{B} = \{$□, ▚, ▞, ■$\}$ and the operator $+ \in \mathcal{B} \times \mathcal{B} \to \mathcal{B}$ defined by the table in Figure 1.

+	□	▚	▞	■
□	□	▚	▞	■
▚	▚	▞	■	□
▞	▞	■	□	▚
■	■	□	▚	▞

Figure 1: Definition for the + operator.

Zero or more of the following statements hold. Which?

A. $\Pr\left[b \leftarrow\$ \mathcal{B} : b \in \{$▚, ▞$\}\right] = \frac{1}{2}$.

B. $\forall b \in \mathcal{B},\ b + b =$ □.

C. $+$ with a fresh uniformly random key has perfect secrecy.

D. $\Pr\left[b_1 \leftarrow\$ \mathcal{B}; b_2 \leftarrow\$ \mathcal{B} : b_1 = b_2 =\right.$ ▞$\left.\right] = \frac{1}{4}$

Q2 — Hashing Passwords [20 marks]

This question explores the use of cryptography to protect passwords while at rest in server databases.

Consider the following system S, which is parameterized by a hash function H. When a user u registers with password p, S stores the tuple $(u, r, \mathsf{H}(p\|r))$ in its database, where r is some random *salt*. When an attempt is made to authenticate user u with a password p', S retrieves the tuple $(u, r, \mathsf{H}(p\|r))$, computes $\mathsf{H}(p'\|r)$, and successfully authenticates the user u is $\mathsf{H}(p\|r) = \mathsf{H}(p'\|r)$. The hash function H is assumed to produce uniformly sampled outputs given uniformly sampled inputs. Assume the adversary does not learn H until the start of the attack.

[2 marks] **2.a)** Assume the distribution user u draws their password p from is completely unknown to the adversary. Denote with ℓ the length of digests output by the hash function H and with n the length of p. What is the probability that the adversary successfully authenticates as user u in one attempt?

2.b) Assume now that the adversary has access to the complete database of tuples held by S. It contains entries for U users.

[2 marks] i. What is an upper-bound on the probability that the adversary authenticates as user u in one attempt?

[5 marks] ii. Assume the adversary has no knowledge of the distribution in which any of the users (not just u) sample their passwords. What can you say about the probability that the adversary authenticates as *any* user in one attempt? Consider, in particular, the effect of the random salt on the adversary's ability to search for multiple matches at once, and of the same salt being used for multiple users.

2.c) A better hash function H′ has been standardised and you want your users to benefit from it. You recompute the entire database for S, so that the tuples are now of the form $(u, r, \mathsf{H}'(\mathsf{H}(p\|r)))$.

[2 marks] i. Why would you not instead replace all entries in S's database with $(u, r, \mathsf{H}'(p\|r))$?

[2 marks] ii. Describe how authentication attempts (with a user u and a password p') are processed.

[7 marks] iii. Assume H is now broken (so that it is easy to compute preimages for any of its outputs). Show that the hash function $\mathsf{H}' \circ \mathsf{H}$ is preimage-resistant if H′ is preimage-resistant.

Q3 — Hybrid Constructions, Practical Cryptography[25 marks]

In this question, we consider constructions that combine symmetric and asymmetric cryptography. We focus on Key Encapsulation Mechanisms, and their use in constructing Hybrid Public Key Encryption.

In the rest of this question, we use a fixed cyclic group $\mathbb{G}$ of prime order q; we use g to denote some fixed generator of $\mathbb{G}$. In addition, we use a hash function H, which we will use as a Key Derivation Function; and a symmetric authenticated encryption scheme $\mathcal{E}$. All definitions below are specialised to this setting.

Key Encapsulation Mechanisms. A Key Encapsulation Mechanism (KEM) is a triple of algorithms $\mathcal{K} = (\mathsf{KGen}, \mathsf{Encap}, \mathsf{Decap})$, where:

KGen: probabilistically generates a keypair in $\mathbb{Z}_q \times \mathbb{G}$;

Encap: given a recipient's public key (in $\mathbb{G}$) and a sender's private key (in $\mathbb{Z}_q$), probabilistically generates a symmetric key (in some keyspace $\mathbb{K}$ and its encapsulation (in some set $\mathbb{E}$); and

Decap: given a recipient's secret key (in $\mathbb{G}$) and an encapsulation, recovers a symmetric key.

We do not use KGen in the following, but note for completeness that both schemes we define use standard Diffie-Hellman key generation in $\mathbb{G}$.

An KEM $\mathcal{K}$ is said to be correct if, for all $\mathsf{sk}_R, \mathsf{sk}_S \in \mathbb{Z}_q$, if $(\mathsf{k}, \mathsf{c}) \leftarrow\!\!\$\ \mathcal{K}.\mathsf{Encap}_{\mathsf{pk}_R}(\mathsf{sk}_S)$ then $\mathcal{K}.\mathsf{Decap}_{\mathsf{sk}_R}(\mathsf{c}) = \mathsf{k}$. (Where $\mathsf{pk}_R = g^{\mathsf{sk}_R}$.)

Figures 2 and 3 show two KEMs, which we study in more detail below. Note that AKEM_0 uses $\mathbb{E} = \mathbb{G}$, while AKEM_1 uses $\mathbb{E} = \mathbb{G} \times \mathbb{G}$.

AKEM_0

$\mathsf{Encap}_{\mathsf{pk}_R}(\mathsf{sk}_S)$	$\mathsf{Decap}_{\mathsf{sk}_R}(\mathsf{pk}_S)$
$\mathsf{dh} \leftarrow \mathsf{pk}_R^{\mathsf{sk}_S}$	$\mathsf{dh} \leftarrow \mathsf{pk}_S^{\mathsf{sk}_R}$
$\mathsf{k} \leftarrow \mathsf{H}(\mathsf{dh}, \mathsf{pk}_R)$	$\mathsf{k} \leftarrow \mathsf{H}(\mathsf{dh}, g^{\mathsf{sk}_R})$
return $(\mathsf{k}, g^{\mathsf{sk}_S})$	**return** k

Figure 2: The AKEM_0 KEM.

AKEM_1

$\mathsf{Encap}_{\mathsf{pk}_R}(\mathsf{sk}_S)$	$\mathsf{Decap}_{\mathsf{sk}_R}(\mathsf{pk}_E, \mathsf{pk}_S)$
$\mathsf{sk}_E \leftarrow\!\!\$\ \mathbb{Z}_q$	$\mathsf{dh} \leftarrow \mathsf{pk}_E^{\mathsf{sk}_R} \\| \mathsf{pk}_S^{\mathsf{sk}_R}$
$\mathsf{k} \leftarrow \mathsf{H}(\mathsf{pk}_R^{\mathsf{sk}_E} \\| \mathsf{pk}_R^{\mathsf{sk}_S}, \mathsf{pk}_R)$	$\mathsf{k} \leftarrow \mathsf{H}(\mathsf{dh}, g^{\mathsf{sk}_R})$
return $(\mathsf{k}, (g^{\mathsf{sk}_E}, g^{\mathsf{sk}_S}))$	**return** k

Figure 3: The AKEM_1 KEM.

Hybrid Public Key Encryption. KEMs can be used, in combination with a Data Encapsulation Mechanism (DEM), to implement Hybrid Encryption.

Here, we focus on the construction of Hybrid Public Key Encryption (HPKE) by simply using the key generated by the KEM in the Authenticated Encryption scheme $\mathcal{E}$, used as a DEM.

Given a DH keypair $(\mathsf{sk}_S, \mathsf{pk}_S)$ for the sender, a DH keypair $(\mathsf{sk}_R, \mathsf{pk}_R)$ for the recipient, some nonce n, and some message m, encryption in HPKE based on KEM $\mathcal{K}$ proceeds by:

1. using $\mathcal{K}$.Encap to generate a symmetric key k and its encapsulation e;
2. encrypting the message m with nonce n under key k with $\mathcal{E}$ into a ciphertext c; and
3. sending both c and e to the recipient.

This is described more formally in Figure 4.

$$\underline{\mathsf{Enc}^{\mathsf{n}}_{\mathsf{sk}_S,\mathsf{pk}_R}(\mathsf{m})}$$
$$(\mathsf{k},\mathsf{e}) \leftarrow\!\!\$\, \mathcal{K}.\mathsf{Encap}_{\mathsf{pk}_R}(\mathsf{sk}_S)$$
$$\mathsf{c} \leftarrow\!\!\$\, \mathcal{E}.\mathsf{Enc}^{\mathsf{n}}_{\mathsf{k}}(\mathsf{m})$$
$$\textbf{return } (\mathsf{c},\mathsf{e})$$

Figure 4: Hybrid Public Key Encryption

[3 marks] **3.a)** Show that AKEM_1 is correct as a KEM.

3.b) We like decrypting things.

[3 marks] i. Describe the decryption algorithm $\mathsf{Dec}^{\mathsf{n}}_{\mathsf{pk}_S,\mathsf{sk}_R}(\mathsf{m})$ for HPKE.

[5 marks] ii. Give a reasonable definition of correctness for HPKE, and prove your decryption algorithm correct, assuming the correctness of the underlying KEM $\mathcal{K}$ and encryption scheme $\mathcal{E}$.

3.c) We now consider the property of Perfect Forward Secrecy. Consider a scenario where Alice (with keypair $(\mathsf{sk}_A, \mathsf{pk}_A)$) and Bob (with keypair $(\mathsf{sk}_B, \mathsf{pk}_B)$) have exchanged multiple ciphertexts encrypted using HPKE with AKEM_0, which Nadia has collected. Nadia nabs Bob off the street, and tickles him until he breaks and reveals sk_B.

[3 marks] i. Argue that Nadia can retrieve *all* plaintexts from the collected ciphertexts, including those sent by Bob to Alice.

[4 marks] ii. Argue (without proving) that Nadia would have been unable to retrieve *past* plaintexts from Bob to Alice had Alice and Bob used AKEM_1 instead.

3.d) We now study a couple of implementation considerations.

[2 marks] i. Give one example of a side-channel in this exam paper's text. (There is nothing in the binary, don't waste time there.) What does it tell you about the University of Bristol's Computer Science Department's exam preparation process, or about one of your examiners' hobbies?

[5 marks] ii. Consider an implementation of HPKE that uses a theoretically secure MAC-then-Encrypt construction as its DEM component, and a secure KEM. Describe the process through which decryption must occur for the implementation to be as secure as the specification. Give details of the ordering of operation and any implementation specific measures needed.

Q4 — Asymmetric Cryptography — MCQs [6 marks]

Please recall the rules for marking MCQs stated in the preamble.

[3 marks] **4.a)** Which of the following RSA key pairs are valid? You may use a computer for this question.

A. sk, pk $= (5, 187), (3, 187)$.

B. sk, pk $= (59, 4362), (781, 4362)$.

C. sk, pk $= (916, 10379), (357, 7031)$.

D. sk, pk $= (19, 77), (24, 77)$.

[3 marks] **4.b)** For which of the following choices of

$$f(x) = x^d + c_{d-1}x^{d-1} + \cdots + c_1 x + c_0$$

with $c_i \in \mathbb{Z}/2\mathbb{Z}$ does the set

$$\{a_{d-1}x^{d-1} + \cdots + a_0 : a_i \in \mathbb{Z}/2\mathbb{Z},\ f(x) \equiv 0\}$$

define a finite field?

A. $f(x) = x^2 + x + 1$.

B. $f(x) = x^3 + x^2 + x + 1$.

C. $f(x) = x^3 + x + 1$.

D. $f(x) = x^4 + 1$.

Q5 — Cryptography in Prime-Order Fields [12 marks]

You may use a computer for this question if you wish but you must show your working. Consider the finite field $\mathbb{F}_{101}$. The element $2 \in \mathbb{F}_{101}^*$ has order 100, so generates the group $\mathbb{F}_{101}^*$ (you do not have to show this).

[4 marks] **5.a)** Using the baby-step-giant-step algorithm, compute $a \pmod{100}$ such that $2^a \equiv 53 \pmod{101}$. You may use without proof that

$$2^{-10} \equiv 65 \pmod{101}.$$

[5 marks] **5.b)** i. Solving the same problem with Pollard-ρ yields a chain

$$(G_0, b_0, c_0) \ldots (G_{22}, b_{22}, c_{22})$$

(where, for each i, we have $G_i = 2^{b_i} \cdot (2^a)^{c_i}$), for which $G_{22} = G_6$ and $G_i \neq G_j$ for all $6, 22 \neq i \neq j$. Which method was more efficient in this instance? Justify your answer.

[3 marks] ii. How does the complexity of these examples compare to the asymptotic complexities of baby-step-giant-step and Pollard-ρ?

Q6 — Cryptography in Extension Fields [22 marks]

6.a) Consider the finite field

$$\mathbb{F}_{3^2} = \{a + bx : a, b \in \mathbb{Z}/3\mathbb{Z}, x^2 + 1 \equiv 0 \pmod 3\}.$$

[3 marks] i. By 'brute-force', find an integer n for which $(1+x)^n \equiv 2 + 2x$ in $\mathbb{F}_{3^2}$.

[5 marks] ii. Name one element that generates $\mathbb{F}^*_{3^2}$ and two elements that don't generate $\mathbb{F}^*_{3^2}$. Justify your answer.

[3 marks] iii. How many choices of generator are there for $\mathbb{F}^*_{3^2}$? Justify your answer.

[4 marks] **6.b)** Consider the finite field

$$\mathbb{F}_{3^3} = \{a + bx + cx^2 : a, b, c \in \mathbb{Z}/3\mathbb{Z}, x^3 + 2x + 1 \equiv 0 \pmod 3\}.$$

The multiplicative group $\mathbb{F}^*_{3^3}$ is generated by $g = x + 1$ (you do not have to show this). We will use this setup for a Diffie-Hellman key exchange: your secret key is $\mathsf{sk}_D = 5$, and Hellman's public key is $x^2 + 2 = \mathsf{pk}_H = g^{\mathsf{sk}_H}$. Using the square-and-multiply algorithm, compute your and Hellman's shared secret key.

6.c) You should use a computer for this question. Wolfram Alpha is sufficient for what you need to do, but use any programme of your choice. Do not submit any code.

[5 marks] i. Which *two* algorithms to compute discrete logarithms would be most efficient for the finite field $\mathbb{F}_{3^{54}}$? Justify your answer.

[2 marks] ii. Are the algorithms you gave more or less efficient for solving discrete logarithms in $\mathbb{F}_{3^{53}}$? Justify your answer.