Coursework

COMS30065 Cryptology

TB1 2022/23
Thursday November 17 — Thursday December 8 2022
Due on Thursday December 8 2022 at 13:00

Objectives and Means

We have studied security definitions and proofs for symmetric cryptography, and the mathematics underlying asymmetric cryptography. In this coursework, we will explore both of these aspects—and the other two (definitions and proof for asymmetric cryptography, and cryptanalysis for symmetric cryptography)—in a slightly more practical setting, including implementing practical attacks against badly deployed cryptography.

The breakdown of marks is as follows:

Question	Points
Asymmetric Cryptanalysis	30
Symmetric Cryptanalysis	20
Asymmetric Cryptography	35
You're Using Diffie-Hellman Wrong	15
Total:	100

General information about the Department's policy on coursework can be found in Appendix A.[1] The below refines this policy *for this unit* and does not apply to other coursework you may be taking.

[1]This is a verbatim copy of the policy, we are not to blame for any crimes against typography committed herein, such as the use of all caps for emphasis.

Instructions

Submit your answer on the Blackboard *COMS30065: Cryptology (with Coursework)* unit by *13:00 on Thursday December 8* (but aim to submit by 12:00, to account for any technical issues).

Your submission should contain at least the following two files:

- a PDF document named "`submission-xxxxxxx.pdf`" (where `xxxxxxx` is your student number) containing your answers to theoretical and dissertative questions, as well as any documentation you wish to submit for assessment;
- an archive named "`code-xxxxxxx.yyy`" (where `xxxxxxx` is your student number, and `yyy` is some file extension) containing your code solutions and any machine-readable files specified in the questions below, as well as any other machine-readable files (including code) you wish to submit for assessment, or in support of your claims.

Your code archive *must be self-contained* in that any external libraries can be obtained by following simple instructions included in the archive itself. It must fully describe all dependencies. We strongly recommend you write and test your instructions to work on the University's lab machines (accessible via SSH): we will use "compiles and runs on the lab machines" as a baseline for quality, so that anything that fails to meet this requirement may not be marked.

You may also submit additional material, where appropriate, for example as evidence of higher achievement. To have value, such material *must* be referred to from your PDF submission in the vicinity of the point or argument it supports. Additional material not referred to from your submission will not be considered for marking.

Workload

This coursework is designed to be doable to a good level of quality in much less than 60 hours overall—two of these over 15 8-hour working days correspond to 40-hour work weeks, or 9:00-17:00 5 days a week. This coursework is also designed so that spending more time on it in unhealthy ways is unlikely to yield higher marks—you will need to be able to think and write clearly to reach higher marks, not just spam your compiler with broken code until your tests go through.[2] Rest when needed, take active breaks, and alternate between your two coursework units in order to avoid tunnel vision.

[2] See the Marking Scheme in Appendix B.

Support and Collaboration

Support will be provided throughout weeks 9 to 11 in the following ways:

- synchronously (in person) on Wednesdays between 11:00 and 12:00 in MVB 3.44;
- synchronously (in person) during office hours (Fridays 11:00-12:00 in MVB 3.47 and 3.13), although we will only engage in general discussion of the material during drop-ins; and
- asynchronously through Teams (either in the Coursework channel, or in some more broadly accessible channel, not in individual chats).

The coursework is *individual*: the material (including text, code, and supporting material) you submit *must* have been written individually. However, we encourage group *discussions* about the coursework questions. The best way to be safe from potential academic integrity concerns is to take no notes or pictures of those discussions, and stop your colleagues from doing so: the only place such discussions should leave a mark is in your heads.

Do note that there are several versions of these coursework questions with subtly different values used in places. A specific version has been assigned to you, and we will expect answers to the questions set out in that version of these instructions. You can refer to the bottom left corner of each page in this document to find the version number, and to the Blackboard assignment page to find which version was allocated to you.

Auxiliary Files

This file should be accompanied by a number of auxiliary files for use in Q2:

`cbc`: An executable implementing AES256-CBC encryption under a fixed key;

`m1.txt`: An ASCII message of the utmost importance;

`n1_cbc.txt`: A hexadecimal string representing the challenge nonce;

`c1_cbc.txt`: A hexadecimal string representing the challenge ciphertext;

`n2_cbc.txt`: A hexadecimal string representing the challenge nonce;

`c2_cbc.txt`: A hexadecimal string representing the challenge ciphertext; and

`aes`: An executable implementing double AES256 enciphering under two fixed keys.

Q1 — Asymmetric Cryptanalysis [30 marks]

This question is about cryptanalysis, both applying the algorithms you have learnt to attack the Discrete Logarithm and going beyond that. You should use SageMath for this question. Please include your SageMath code in your submission.

1.a) [6 marks] This question is about Pollard-rho. Let $p = 1031$. Using SageMath, find an element $g \in \mathbb{F}_p^*$ of order 103. With your choice of generator g, use Pollard-rho to find a such that $g^a \equiv 162 \pmod{1031}$.

1.b) This question is about Pohlig-Hellman and index calculus.

i. [7 marks] Let $g = 1344 \pmod{2027}$; then g generates $\mathbb{F}_{2027}^*$ as a multiplicative group (you do not have to prove this). Use Pohlig-Hellman and index calculus to compute a such that $g^a = 478$.
Hint: When choosing your factor base, look first at the factorisation of g^i for some values of i. Note that if you've looked at g^i then often you won't learn anything new from looking at $(g^i)^j$ for small j–so think about which powers of g are useful.
Note: If you get stuck on computing a factor base, please ask for help. If you are unable to compute a factor base, you can choose to sacrifice 4 marks and ask to be given one for your instance.

ii. [4 marks] Describe an algorithm to find a factor base for a generic instance of the discrete logarithm problem in $\mathbb{F}_p$.

1.c) This question is about baby-step-giant-step and more general groups.

i. [6 marks] Let $p = 127$ and consider the set

$$S = \{\alpha + \beta\sqrt{3} : \alpha, \beta \in \mathbb{Z}/p\mathbb{Z}\} - \{0\}.$$

Prove that S is a group under multiplication.

ii. [4 marks] Use (only) the baby-step-giant-step algorithm to compute a such that $(18\sqrt{3}+125)^a = 37+12\sqrt{3}$. You may use without proof that $18\sqrt{3} + 125$ generates S.

iii. [3 marks] What would be the most efficient algorithm to solve this discrete logarithm problem? Justify your answer. How would your proposal perform for the group described in part (i) for large p (you may assume it is still a group)?

Q2 — Symmetric Cryptanalysis [20 marks]

We now consider strange and wonderful symmetric schemes, and use them and mount a few attacks.

Breaking CBC

You have been given an executable[3] `cbc`, which implements AES256-CBC under a fixed key k_{cbc}, using its first argument as a nonce and its second argument as a plaintext.

When calling it, nonces should be valid hexadecimal strings, and are truncated or zero-extended (to the right) to the appropriate length. Plaintexts are interpreted directly as bytestrings, but cannot contain null characters.

2.a) [2 marks] Produce a CBC-MAC tag for the message given in file `m1.txt` under key k_{cbc}.

2.b) [2 marks] Decrypt the ciphertext given in file `c1_cbc.txt` under key k_{cbc}, and with the nonce given in file `n1_cbc.txt`.

Describe your strategy, characterise the attack you are mounting as precisely as possible, and include the resulting plaintext (encoded as a hexadecimal string, or in ASCII) in your PDF submission, and in a machine-readable file `p1_cbc.txt` in your archive.

2.c) [6 marks] Describe and implement (in a language of your choice) a *nonce-respecting* attack that will decrypt the ciphertext given in file `c2_cbc.txt` under key k_{cbc}, and with the nonce given in file `n2_cbc.txt`. The plaintext contains only ASCII letters (uppercase and lowercase) and punctuation. It is padded using (unfortunately) PKCS padding.

Describe your strategy, characterise the attack you are mounting as precisely as possible, and include the resulting plaintext (encoded as a hexadecimal string, or in ASCII) in your PDF submission, and in a machine-readable file `p2_cbc.txt` in your archive.

Partial marks will *not* be given for attacks that are not nonce-respecting.

[3] It has been tested to run on the Linux lab machines. I can compile it for other platforms on demand, and with sufficient lead time.

Meeting Halfway

You have been given an executable[4] `aes`, which implements a double AES256 under two fixed keys k_1^{aes}, and k_2^{aes} using its argument as a plaintext. (As before, the plaintext is interpreted directly as a bytestring, but cannot contain null bytes.) By double AES256 under two fixed keys, we mean that `aes` prints to standard output the result of encrypting its input m as $\text{AES}_{k_2^{\text{aes}}}(\text{AES}_{k_1^{\text{aes}}})$.

k_1^{aes} and k_2^{aes} are 16-bit values—occupying the two rightmost bytes of a 256-bit AES key, with the other bytes zeroed out.

2.d) [10 marks] Recover k_1^{aes} and k_2^{aes}.

[4] It has been tested to run on the Linux lab machines. I can compile it for other platforms on demand, and with sufficient lead time.

Q3 — Asymmetric Cryptography [35 marks]

In this question, we consider some definitions and proofs in an RSA-like asymmetric setting. Our chosen asymmetric setting will require some rather heavy notation, which we introduce now.[5]

We consider:

- A set of public keys $\mathcal{PK}$;
- A set of private keys $\mathcal{SK}$;
- A set of keypairs $\mathcal{KP} \subseteq \mathcal{PK} \times \mathcal{SK}$;
- A domain $\mathcal{D}$; and
- A set of messages $\mathcal{M} = \{0,1\}^{\ell}$ for some $\ell > 0$.

We assume, for simplicity here, that $\mathcal{KP}$, seen as a relation, is a total and bijective function, and denote with $\mathcal{KP}^{-1}$ its inverse bijection. Given a public key $\mathsf{pk} \in \mathcal{PK}$, we consider a set $\mathcal{D}_{\mathsf{pk}} \subseteq \mathcal{D}$ of *valid* domain elements for pk.

Asymmetric Encryption

We formally define the syntax and security of asymmetric encryption schemes (in our somewhat restricted setting).

Definition 1 (Asymmetric Encryption Schemes)**.** An asymmetric encryption scheme is a triple of algorithms $\mathsf{E} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$, where:

Kg probabilistically outputs an keypair in $\mathcal{KP}$;

Enc probabilistically outputs a ciphertext in some set $\mathcal{C}$ given a public key in $\mathcal{PK}$ and a message in $\mathcal{M}$;

Dec deterministically outputs a plaintext in $\mathcal{M}$ (or an error) given a secret key in $\mathcal{SK}$ and a ciphertext in $\mathcal{C}$.

Definition 2 (CPA Security for Asymmetric Encryption)**.** Let E be an asymmetric encryption scheme. The *chosen-plaintext attack* advantage of some adversary $\mathbb{A}$ against E is defined as follows, where experiments $\mathsf{Exp}^{\mathsf{cpa\text{-}real}}_{\mathsf{E}}(\mathbb{A})$ and $\mathsf{Exp}^{\mathsf{cpa\text{-}ideal}}_{\mathsf{E}}(\mathbb{A})$ are defined as in Figure 1.

[5] The concepts are simple, but hard to express formally because the set of values our functions operate on depends on dynamically-chosen values. There is, however, nothing particularly difficult conceptually. Ask!

$\underline{\mathsf{Exp}^{\text{cpa-real}}_{\mathsf{E}}(\mathbb{A})}$
$(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Kg}$
$\mathsf{m} \xleftarrow{\$} \mathbb{A}(\mathsf{pk})$
$\mathsf{c} \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}}(\mathsf{m})$
$\widehat{\mathsf{b}} \xleftarrow{\$} \mathbb{A}(\mathsf{c})$

$\underline{\mathsf{Exp}^{\text{cpa-ideal}}_{\mathsf{E}}(\mathbb{A})}$
$(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Kg}$
$\mathsf{m} \xleftarrow{\$} \mathbb{A}(\mathsf{pk})$
$\mathsf{c} \xleftarrow{\$} \mathcal{C}_{\mathsf{pk}}$
$\widehat{\mathsf{b}} \xleftarrow{\$} \mathbb{A}(\mathsf{c})$

Figure 1: The CPA experiment

$$\mathsf{Adv}^{\text{cpa}}_{\mathsf{E}}(\mathbb{A}) = \left|\Pr\left[\mathsf{Exp}^{\text{cpa-real}}_{\mathsf{E}}(\mathbb{A}) : \widehat{\mathsf{b}}\right] - \Pr\left[\mathsf{Exp}^{\text{cpa-ideal}}_{\mathsf{E}}(\mathbb{A}) : \widehat{\mathsf{b}}\right]\right|$$

We say that E is (t, ϵ)-*CPA secure* if, for any $\mathbb{A}$ that runs in time at most t, we have $\mathsf{Adv}^{\text{cpa}}_{\mathsf{E}}(\mathbb{A}) \leq \epsilon$.

3.a) [2 marks] Show that RSA as described in lectures is not (t, ϵ)-CPA-secure for any reasonable t and ϵ.

3.b) [2 marks] Compare Definitions 1 and 2 to their symmetric equivalents. Focus in particular on discussing nonces and oracles, explaining why they do not appear in the asymmetric setting.

One-Way Trapdoor Permutations

Definition 3 (Trapdoor Permutation). A trapdoor permutation is a family $\left(f_{\mathsf{pk}}\right)_{\mathsf{pk}\in\mathcal{PK}}$ of functions from domain elements to domain elements, indexed by $\mathcal{PK}$ such that there exists a family $\left(f^{-1}_{\mathsf{sk}}\right)_{\mathsf{sk}\in\mathcal{SK}}$, indexed by $\mathcal{SK}$, and such that, for every $(\mathsf{pk}, \mathsf{sk}) \in \mathcal{KP}$ and every $\mathsf{m} \in \mathcal{D}_{\mathsf{pk}}$, we have:

- $f^{-1}_{\mathsf{sk}}(f_{\mathsf{pk}}(\mathsf{m})) = \mathsf{m}$; and
- $f_{\mathsf{pk}}(f^{-1}_{\mathsf{sk}}(\mathsf{m})) = \mathsf{m}$.

We can define a reasonable hardness assumption on trapdoor permutation, namely that it is computationally hard to compute f^{-1}_{sk} without knowing the private key sk—also known as the *trapdoor*.

$\underline{\mathsf{Exp}_{f}^{\text{owtp}}(\mathbb{A})}$
$(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathcal{KP}$
$\mathsf{m} \xleftarrow{\$} \mathcal{D}_{\mathsf{pk}}$
$\mathsf{c} \leftarrow f_{\mathsf{pk}}(\mathsf{m})$
$\widehat{\mathsf{m}} \leftarrow \mathbb{A}(\mathsf{pk}, \mathsf{c})$

Figure 2: The OWTP experiment

$\underline{\mathsf{Exp}_{\mathsf{H},f}^{\text{ees-real}}(\mathbb{A})}$
$\mathsf{k} \xleftarrow{\$} \mathcal{K}$
$(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathcal{KP}$
$\mathsf{x} \xleftarrow{\$} \mathcal{D}_{\mathsf{pk}}$
$\mathsf{y} \leftarrow \mathsf{H}_{\mathsf{k}}(\mathsf{x})$
$\widehat{\mathsf{b}} \leftarrow \mathbb{A}\left(\mathsf{k}, \mathsf{pk}, f_{\mathsf{pk}}(\mathsf{x}), \mathsf{y}\right)$

$\underline{\mathsf{Exp}_{\mathsf{H},f}^{\text{ees-ideal}}(\mathbb{A})}$
$\mathsf{k} \xleftarrow{\$} \mathcal{K}$
$(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathcal{KP}$
$\mathsf{x} \xleftarrow{\$} \mathcal{D}_{\mathsf{pk}}$
$\mathsf{y} \xleftarrow{\$} \{0,1\}^{\ell}$
$\widehat{\mathsf{b}} \leftarrow \mathbb{A}\left(\mathsf{k}, \mathsf{pk}, f_{\mathsf{pk}}(\mathsf{x}), \mathsf{y}\right)$

Figure 3: The EES experiment

Definition 4 (One-Way Trapdoor Permutation)**.** Let f be a trapdoor permutation. The *one-way* advantage of some adversary $\mathbb{A}$ against f is defined as follows, where experiment $\mathsf{Exp}_{f}^{\text{owtp}}(\mathbb{A})$ is defined as in Figure 2.

$$\mathsf{Adv}_{f}^{\text{owtp}}(\mathbb{A}) = \Pr\left[\mathsf{Exp}_{f}^{\text{owtp}}(\mathbb{A}) : \widehat{\mathsf{m}} = \mathsf{m}\right]$$

We say that f is (t, ϵ)-*OWTP secure* if, for any $\mathbb{A}$ that runs in time at most t, we have $\mathsf{Adv}_{f}^{\text{owtp}}(\mathbb{A}) \leq \epsilon$.

3.c) [2 marks] Argue, based on the easy and hard problems seen in lectures, that RSA is a reasonable candidate for a one-way trapdoor permutation. Take care to give candidate instantiations for $\mathcal{PK}$, $\mathcal{SK}$ and $\mathcal{KP}$, to define f and f^{-1}, and to prove their expected properties.

Extended Entropy-Smoothing Hash Functions

We also consider a family $(\mathsf{H}_{\mathsf{k}})_{\mathsf{k} \in \mathcal{K}}$ of hash functions from $\mathcal{D}$ to $\mathcal{M}$ indexed by some set of hash keys $\mathcal{K}$.

$\underline{\mathsf{Kg}()}$
$\mathsf{k} \xleftarrow{\$} \mathcal{K}$
$(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathcal{KP}$
return $((\mathsf{pk}, \mathsf{k}), (\mathsf{sk}, \mathsf{k}))$

$\underline{\mathsf{Enc}_{(\mathsf{pk},\mathsf{k})}(\mathsf{m})}$
$\mathsf{r} \xleftarrow{\$} \mathcal{D}_{\mathsf{pk}}$
$\mathsf{s} \xleftarrow{\$} \mathsf{H}_{\mathsf{k}}(\mathsf{r})$
return $\left(f_{\mathsf{pk}}(\mathsf{r}), \mathsf{s} \oplus \mathsf{m}\right)$

$\underline{\mathsf{Dec}_{(\mathsf{sk},\mathsf{k})}(\mathsf{h}, \mathsf{c})}$
$\mathsf{r} \leftarrow f_{\mathsf{sk}}^{-1}(\mathsf{h})$
$\mathsf{s} \xleftarrow{\$} \mathsf{H}_{\mathsf{k}}(\mathsf{r})$
return $(\mathsf{s} \oplus \mathsf{c})$

Figure 4: An asymmetric encryption scheme

We further define a novel assumption, extended entropy-smoothing, on the *joint security* of a hash function H and a trapdoor permutation f with the types specified above. Intuitively, we say that H is f-extended entropy-smoothing if no adversary can distinguish the hash of a random domain element x from a random element of $\{0,1\}^{\ell}$, *even* if it is also given the image of x by f_{pk} for a uniformly sampled key pair.

Definition 5 (Extended Entropy-Smoothing Hash Functions)**.** Let H be a a hash function family and f a trapdoor permutation as specified above. The *extended entropy-smoothing* advantage of some adversary $\mathbb{A}$ against H and f is defined as follows, where experiments $\mathsf{Exp}^{\text{ees-real}}_{\mathsf{H},f}(\mathbb{A})$ and $\mathsf{Exp}^{\text{ees-ideal}}_{\mathsf{H},f}(\mathbb{A})$ are defined as in Figure 3.

$$\mathsf{Adv}^{\text{ees}}_{\mathsf{H},f}(\mathbb{A}) = \left|\Pr\left[\mathsf{Exp}^{\text{ees-real}}_{\mathsf{H},f}(\mathbb{A}) : \widehat{\mathsf{b}}\right] - \Pr\left[\mathsf{Exp}^{\text{ees-ideal}}_{\mathsf{H},f}(\mathbb{A}) : \widehat{\mathsf{b}}\right]\right|$$

We say that H is (t, ϵ)-f*-extended entropy-smoothing* if, for any $\mathbb{A}$ that runs in time at most t, we have $\mathsf{Adv}^{\text{ees}}_{\mathsf{H},f}(\mathbb{A}) \leq \epsilon$.

3.d) [6 marks] Discuss the relation between extended entropy-smoothing and one-way trapdoor permutation. If they are related, full marks will be given for a reduction. If they are unrelated, full marks will be given for counterexamples. Partial marks will be given for reasonable *and relevant* observations on the relative hardness of the two problems.

A Simple Scheme

We now consider the asymmetric encryption scheme $\mathsf{E}_{\mathsf{H},f}$ shown in Figure 4, which is parameterised by a hash function H and a trapdoor permutation f.

3.e) [8 marks] Show that, if H is a (t, ϵ)-f-extended entropy-smoothing hash function, then $\mathsf{E}_{\mathsf{H},f}$ is (t, ϵ)-CPA secure.

You may use without proof—but by sacrificing some marks—that the image of a random domain element x by a permutation is indistinguishable from a random domain element, as long as x is not used elsewhere.

3.f) [15 marks] Using RSA as a trapdoor permutation and a cryptographic hash function of your choice, implement (in SageMath or some other language of your choice) the scheme $\mathsf{E}_{\mathsf{H},f}$.

A portion of the marks will be awarded for appropriately documenting your design and discussing any details left out of the abstract description above, but which must be fixed for implementation. A particular focus on justifying any such choices based on reasonable cryptographic security reasoning will be rewarded. Informal arguments based on direct and specific references to the unit's content will be sufficient to pass. Higher marks will be awarded for increasingly formal evidence, all the way to reduction-based proofs.

The remaining portion of the marks will be awarded for appropriately documenting your software engineering process and the quality of its outcomes. At the low end of the range, this will include basic evidence of good software practices (version control, testing). Higher marks will consider also implementation choices and their documentation, and aspects of cryptographic security relevant only to implementations (including side-channels).

Top scorers will have considered the (reasoned and justified) use of extendable output functions to support arbitrary length messages, rather than only fixed-length messages.

There is no formal split of the 15 marks between design and implementation, which will be marked as a whole, and assigned a mark following the Marking Scheme (Appendix B).

Q4 — You're Using Diffie-Hellman Wrong [15 marks]

Your former friend Eve, who you don't like ever since she fed an old sock to your cat, thinks she's still your friend. She knows you're taking Cryptology in Bristol and asks you to set her up to digitally sign messages.

You choose a prime

$$p = 340282366920938463463374607431768211537$$

and a generator

$$g = 362726788824458512630091785490142681 67$$

for $\mathbb{F}_p^*$. You then choose

$$t = 251516970657373045933706224743642467975$$

and compute another generator

$$g' = g^t = 182420262759454946430860805587228613 34$$

for $\mathbb{F}_p^*$. You suggest the parameters (p, g') as a public parameter setup for an ElGamal signature scheme. Eve immediately generates her public key

$$pk_E = 531100243319417803546184771172699586 75$$

and shares it with all her work "friends".

Take your revenge, and sign message

$$m = 302027202049471816444472291556640814 63$$

as if you were Eve.[6]

Include Eve's forged signature in your PDF submission, and as machine-readable files (or a single file) in your code archive.

In your PDF submission, explain in detail how your attack works, and how the setup choices enable it (if any of those choices are related to your attacks).

[6] It's nothing rude, we swear.

A Departmental Policy on Coursework

Deadline

The deadline for submission of all optional unit assignments is 13:00 on Thursday 8th of December (due to the fact that the University discourages Friday deadlines). Students should submit all required materials to the "Assessment, submission and feedback" section of Blackboard - it is essential that this is done on the Blackboard page related to the "With Coursework" variant of the unit.

Time commitment

You are expected to work on both of your optional unit courseworks in the 3-week coursework period as if it were a working week in a regular job - that is 5 days a week for no more than 8 hours a day. The effort spent on the assignment for each unit should be approximately equal, being roughly equivalent to 1.5 working weeks each. It is up to you how you distribute your time and workload between the two units within those constraints.

You are strongly advised NOT to try and work excessive hours during the coursework period: this is more likely to make your health worse than to make your marks better. If you need further pastoral/mental health support, please talk to your personal tutor, a senior tutor, or the university wellbeing service.

Academic Offences

Academic offences (including submission of work that is not your own, falsification of data/evidence or the use of materials without appropriate referencing) are all taken very seriously by the University. Suspected offences will be dealt with in accordance with the University's policies and procedures. If an academic offence is suspected in your work, you will be asked to attend an interview with senior members of the school, where you will be given the opportunity to defend your work. The plagiarism panel are able to apply a range of penalties, depending the severity of the offence. These include: requirement to resubmit work, capping of grades and the award of no mark for an element of assessment.

Extenuating circumstances

If the completion of your assignment has been significantly disrupted by serious health conditions, personal problems, periods of quarantine, or other similar issues, you may be able to apply for consideration of extenuating circumstances

(in accordance with the normal university policy and processes). Students should apply for consideration of extenuating circumstances as soon as possible when the problem occurs, using the following online form: https://www.bristol.ac.uk/request-extenuating-circumstances-form

You should note however that extensions of any significant length are not possible for optional unit assignments. If your application for extenuating circumstances is successful, you may be required to retake the assessment of the unit at the next available opportunity (e.g. during the summer reassessment period).

B Marking Scheme

Marking Range	Grade Descriptor
40-49	The work demonstrates basic understanding of most principles of modern cryptography, and ability to directly apply some of the techniques taught in the unit to problems that are close in nature to those seen in the unit.
50-59	The work demonstrates a good understanding of the principles, and a good ability to directly apply the techniques taught in the unit. The work also provides limited evidence that the student is able to critically evaluate the various techniques in order to select the most appropriate for some unseen problems.
60-69	The work demonstrates the student's ability to select the most appropriate technique to effectively tackle known and new problems. This may be done fully in either theoretical work, or practical implementation tasks, or partially in both.
70-79	The work demonstrates the student's ability to select and apply the most appropriate technique to effectively tackle known and new problems, including relatively open tasks. The application is carried out to a high standard of quality, and its write-up demonstrates a good grasp of theoretical and practical methodologies. In particular, some results are evaluated critically in the context of the set task.
80+	The work achieves the standard of quality of the 70-79 marking range across the entire set of skills and methods being assessed.

The coursework contains roughly

- 50 marks in small, guided questions that directly apply techniques taught as part of the unit,

- 10 marks that require a more thorough understanding of the material,
- 10 marks that are relatively direct applications of known techniques to new problems, and
- 30 marks dedicated to more open-ended problems, given without further guidance, and sometimes requiring reading well beyond the scope of the taught material.