Coursework

COMS30065 Cryptology

TB1 2021-2022
Monday November 22 — Friday December 10 2021
Deadline: Friday December 10 2021 — 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 symmetric and asymmetric cryptography.

The breakdown of marks is as follows:

Question	Points
Asymmetric Proofs	10
Asymmetric Cryptanalysis	30
Symmetric Cryptanalysis	25
Implementing Modes of Operation	20
RSA Decrypting without Secret Keys	15
Total:	100

Instructions

Submit your answer on the Blackboard *COMS30065: Cryptology (with Coursework)* unit by *13:00 on Friday December 10* (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 examination;
- 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 examination, 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 will 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 in less than 60 hours overall—two of these over three weeks 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.[1] Rest when needed, take active breaks, and alternate between your two coursework units in order to avoid tunnel vision.

Support and Collaboration

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

- synchronously (through Teams and in person) on Tuesdays between 16:00 and 17:00;
- asynchronously through Teams.

[1] See the Marking Scheme in Appendix A.

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 5 different 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 Q3.d, files:

  `ctr`: An executable implementing AES256-CTR encryption under a fixed key;

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

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

- For Q3.d, files:

  `cfb`: An executable implementing AES256-CFB encryption under a fixed key;

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

  `c_cfb.txt`: A hexadecimal string representing the challenge ciphertext.

- For Q5, files:

  `n1.txt`-`n5.txt`: Decimal strings representing the public moduli N_1, ..., N_5; and

  `c1.txt`-`c5.txt`: Decimal strings representing the ciphertext c_1, ..., c_5. Only c_1 is relevant to you personally.

Q1 — Asymmetric Proofs [10 marks]

In this question, consider a fixed cyclic group $\mathbb{G}$ whose order is some prime q, and a generator g of $\mathbb{G}$. All definitions shown below are specialised to this setup.

In practice, very few cryptographic schemes have security that relies directly on the hardness of the Discrete Logarithm Problem. A common assumption used in security proofs is that the Decisional Diffie-Hellman problem—shown below—is hard.

$\mathsf{DDH}^{\text{real}}_{\mathcal{D}}()$
$a \leftarrow\!\$\, \mathbb{Z}/q\mathbb{Z}$
$b \leftarrow\!\$\, \mathbb{Z}/q\mathbb{Z}$
$z \leftarrow ab$
$r \leftarrow\!\$\, \mathcal{D}(g^a, g^b, g^z)$

$\mathsf{DDH}^{\text{ideal}}_{\mathcal{D}}()$
$a \leftarrow\!\$\, \mathbb{Z}/q\mathbb{Z}$
$b \leftarrow\!\$\, \mathbb{Z}/q\mathbb{Z}$
$z \leftarrow\!\$\, \mathbb{Z}/q\mathbb{Z}$
$r \leftarrow\!\$\, \mathcal{D}(g^a, g^b, g^z)$

$$\mathsf{Adv}^{\text{ddh}}(\mathcal{D}) = \Pr\left[\mathsf{DDH}^{\text{real}}_{\mathcal{D}}() : r\right] - \Pr\left[\mathsf{DDH}^{\text{ideal}}_{\mathcal{D}}() : r\right]$$

Figure 1: The Decisional Diffie-Hellman problem

More formally, we say that the DDH problem is *hard in a cyclic group* $\mathbb{G}$ if, all efficient distinguishers $\mathcal{D}$ have a small advantage $\mathsf{Adv}^{\text{ddh}}(\mathcal{D})$ in distinguishing real DDH tuples from random ones.

You Find a Strange Cipher

Consider the following enciphering scheme $\mathcal{N}$ over group $\mathbb{G}$:

KGen(): sample x uniformly at random in $\mathbb{Z}/q\mathbb{Z}$, and return g^x;

$\mathsf{Enc}_k(m)$: return $k \cdot m$.

1.c) [2 marks] Prove that $\mathcal{N}$ has perfect secrecy.

You Build a Public Key Encryption Scheme

We turn the enciphering scheme $\mathcal{N}$ into a public key encryption scheme. A public key encryption scheme is a triple of algorithms $\mathcal{E} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$,

where KGen generates *key pairs*, composed of a public key and a secret key, Enc encrypts a message using a public key, and Dec decrypts ciphertexts using a secret key.

Let $\mathcal{E}$ be the following public key encryption scheme:

KGen(): sample sk uniformly at random in $\mathbb{Z}/q\mathbb{Z}$, compute pk $\leftarrow g^{\mathsf{sk}}$; return $(\mathsf{pk}, \mathsf{sk})$;

$\mathsf{Enc}_{\mathsf{pk}}(m)$: sample r uniformly at random in $\mathbb{Z}/q\mathbb{Z}$, return the pair $(g^r, \mathsf{pk}^r \cdot m)$.

$\mathsf{Dec}_{\mathsf{sk}}(c_1, c_2)$: return $c_2 \cdot (c_1^{\mathsf{sk}})^{-1}$.

We want to show that $\mathcal{E}$ is secure against chosen plaintext attacks. We first need to define what that means for public key encryption. We do so using the experiments and advantage definition below. $\mathcal{C}$ is the set of ciphertexts.

$\mathsf{CPA}^{real}_{\mathcal{E},\mathcal{A}}()$	$\mathsf{CPA}^{ideal}_{\mathcal{E},\mathcal{A}}()$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}()$	$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}()$
$m^* \leftarrow\!\!\$\ \mathcal{A}(\mathsf{pk})$	$m^* \leftarrow\!\!\$\ \mathcal{A}(\mathsf{pk})$
$c^* \leftarrow\!\!\$\ \mathsf{Enc}_{\mathsf{sk}}(m^*)$	$c^* \leftarrow\!\!\$\ \mathcal{C}$
$b \leftarrow\!\!\$\ \mathcal{A}(c)$	$b \leftarrow\!\!\$\ \mathcal{A}(c)$

$$\mathsf{Adv}^{\mathrm{cpa}}_{\mathcal{E}}(\mathcal{A}) = \Pr\left[\mathsf{CPA}^{real}_{\mathcal{A}}() : b\right] - \Pr\left[\mathsf{CPA}^{ideal}_{\mathcal{A}}() : b\right]$$

Figure 2: CPA Experiment for Public Key Encryption

1.b) [8 marks] Prove that the CPA security of $\mathcal{E}$ follows from the hardness of DDH in $\mathbb{G}$.

Q2 — 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, an open source computer algebra library based on Python that you can download at `www.sagemath.org`.

2.a) [5 marks] Using SageMath to aid you, use the baby-step-giant-step algorithm to compute a such that $362^a = 712 \pmod{829}$. Please include your SageMath computations in your submission.

2.b) This question is about Pollard-rho.

i. [4 marks] Find a generator g for $\mathbb{F}_{31}^*$ and explain how you checked that your choice of g is a generator.

ii. [3 marks] With your choice of generator g, attempt to use Pollard-rho to find a such that $g^a \equiv 2 \pmod{31}$. If Pollard-rho fails, explain why. If you were unable to do part (i), you may use $g = 12$.

iii. [4 marks] Write a SageMath script to find all the values of $g^a \pmod{31}$ for which Pollard-rho fails.

iv. [3 marks] Discuss whether or not Pollard-rho is likely to fail for a real-life cryptographic instance.

2.c) This question is about index calculus.

i. [7 marks] Let $g = 486 \pmod{563}$; then g generates $\mathbb{F}_{563}^*$ as a multiplicative group (you do not have to prove this). Use index calculus to compute $a \pmod{562}$ such that $g^a = 177$.
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 $\mathbb{F}_p$.

Q3 — Symmetric Cryptanalysis [25 marks]

We now consider and mount a few attacks on practical schemes deployed badly.

You Invert A Hash Function

Consider a hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^{256}$. In this question, assume it behaves like a truly random function.

3.a) [4 marks] You are given $n \in \{n \in \mathbb{N} \mid n \leq 256\}$, and a 256-bit string $y \in \{0,1\}^n$, chosen uniformly at random. You are given q attempts to find $x \in \{0,1\}^*$ such that $\mathsf{H}(x) \in \{z \| y \mid z \in \{0,1\}^{256-n}\}$. What is your probability of succeeding?

Justify your answer, which should be expressed as a function of n and q.

3.b) [4 marks] Based on the above, come up with a strategy to find a bitstring $x \in \{0,1\}^*$ such that the last 7 characters of $\mathsf{H}(x)$, when encoded in hexadecimal, are your student number. Describe your strategy.

Include in your PDF submission a hexadecimal encoding of such a bitstring x where H is SHA-256. Also include this string in a file `preimage.txt` in your archive.

3.c) [2 marks] Reflect on the performance of your attack (in terms of number of guesses) based on the analysis above.

You Misuse Encryption Schemes

You have been given two executables.[2]

`ctr` implements AES256-CTR under a fixed key k_{ctr}, using its first argument as a nonce and its second argument as a plaintext.

`cfb` implements AES256-CFB under a (different) fixed key k_{cfb}, using its first argument as a nonce and its second argument as a plaintext.

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 a null character.

[2] These have been tested to run on the Linux lab machines. I can compile them for other platforms on demand, and with sufficient lead time.

3.d) [2 marks] Decrypt ciphertext c_{ctr} (also given in file `c_ctr.txt`) under key k_{ctr}, and nonce n_{ctr} (also given in file `n_ctr.txt`).

$n_{\text{ctr}} =$ `821f5b2a281be84e`

$c_{\text{ctr}} =$ `70d298039885d52da7fd0696921f51ddd84879ab`
`0638f4020536a17ca68d43d65a4564b4ccc86578`
`b5c9e78c617ade716960fdefc70e5a9cdf`

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 `p_ctr.txt` in your archive.

3.e) [8 marks] Decrypt ciphertext c_{cfb} (also given in file `c_cfb.txt`) under key k_{ctr}, and nonce n_{cfb} (also given in file `n_cfb.txt`).

$n_{\text{cfb}} =$ `749c5b51a5785234`

$c_{\text{cfb}} =$ `e5d8e3a634c710e792a3c65c1ad61e88dc0febbe`
`5f5b07a2c3efae20bcacd3e0a76166fd28c4ca17`
`dcbc1b6610c79c0944`

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 `p_cfb.txt` in your archive.

3.f) [5 marks] In a language of your choice, implement a program that automates the attack from Part 3.e. Submit your code in your code archive, including a few tests.

Q4 — Implementing Modes of Operation [20 marks]

Design a mode of operation that realizes secure nonce-based authenticated encryption from a secure blockcipher with n_k-bit keys and ℓ-bit blocks, and a secure MAC n_k-bit keys, and n-bit tags. The key for the encryption scheme should be no larger than n_k bits.

Justify your design and its security using knowledge and skills from the unit. You may assume that the MAC is a PRF.

Implement your design, instantiating its blockcipher with AES256, and its MAC with HMAC-SHA256. The implementation can be written in a language of your choice. You can use existing code and libraries for AES256 and HMAC-SHA256, but all other code components *must* be original.

Marking Criteria. The design will be evaluated for security and simplicity, as well as the quality of the evidence provided for its security. Informal arguments based on direct and specific references to the unit's content will be sufficient to pass. Higher marks will be given for increasingly formal evidence.

The implementation will be evaluated for correctness, simplicity and security, as well as the quality of the evidence provided for correctness and security. 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.

There is no formal split of the 20 marks between design and implementation, which will be marked as a whole, and assigned a mark following the Marking Scheme (Appendix A).

$\mathsf{KGen}((\mathsf{ID}_i)_{1\leq i\leq n}, s)$

foreach i **in** $\{1 \ldots n\}$

$p_i \leftarrow \mathsf{generate\_prime}(s)$

$q_i \leftarrow \mathsf{generate\_prime}(s \| \mathsf{ID}_i)$

$N_i \leftarrow p_i \cdot q_i$

$\mathsf{pk}_i \leftarrow (65537, N_i)$

$\mathsf{sk}_i \leftarrow \left(65537^{-1} \pmod{N_i}, N_i\right)$

return $(\mathsf{pk}_i, \mathsf{sk}_i)_{1\leq i\leq n}$

Figure 3: Our key generation process, which generates n keypairs $((\mathsf{pk}_i, \mathsf{sk}_i))_{1\leq i\leq n}$ for users with IDs $(\mathsf{ID}_i)_{1\leq i\leq n}$, from a single random seed s using a pseudorandom generator generate_prime.

Q5 — RSA Decrypting without Secret Keys [15 marks]

In this question, you break RSA-8192.[3] In the following, RSA is used without any padding, and *exactly* as described in the lectures. You may wish to rely on Sage to tackle this question.

We have generated keypairs for all of you, using a process described in pseudocode in Figure 3. The moduli $\{N_i\}_{1\leq i\leq 5}$ are in the attached files `n1.txt`, `n2.txt`, `n3.txt`, `n4.txt` and `n5.txt`.

Your public key is $(65537, N_1)$. (Your public modulus is in file `n1.txt`.)

Decrypt the ciphertext given (as a decimal number) in file `c1.txt`, which has been encrypted under your public key. The plaintext is an ASCII string, but other forms will be accepted. Include the recovered plaintext in your PDF submission and in a machine-readable file `ptxt1.txt` in your archive.

Explain your attack strategy, and how you conducted it in practie, in a short report.

Marking Criteria. At the bottom end of the range, it will be sufficient to retrieve the plaintext. At the top end of the range, the report will be of sufficient quality to clearly communicate: where the flaw lies, why and how the attack works, and briefly discuss how it could be avoided without too much change.

[3] This is stronger than you'll ever find in practice, so this serves to demonstrate that using very strong cryptography badly doesn't help. And also that there's always a way to avoid having to redo the work when you accidentally generate primes that are twice the expected length.

A 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.