COMP0025: Introduction to Cryptography

Coursework

Department of Computer Science
University College London

Released: November 08, 2024

Due: December 16, 2024 at 16:00 UK time

Instructions

- This assignment is part of the mandatory assessment of the *COMP0025: Introduction to Cryptography* module and will count **25%** towards your final overall mark.
- Assignment submission is due via Moodle through the TurnItIn interface on **December 16, 2024 at 16:00 UK time**. Late submissions will be accepted with deductions according to UCL's late submission policy.
- Only **PDF submissions** that are typeset with LaTeX, *e.g.*, via `https://www.overleaf.com/edu/ucl`, will be accepted. Submissions **must not include screenshots**, *e.g.*, of handwritten or drawn solutions, unless explicitly permitted. Students with disability accommodations are excluded from this requirement.
- Next to the PDF with your answers, you may be asked to hand in additional files, *e.g.*, containing source code, which should be submitted separately to the PDF. In particular, **do not** hand in your files via a zip archive. For more details, please refer to the instructions in the question text.
- This assignment is open note, open book, and open course resources. You must identify sources as accurately and fully as possible. UCL plagiarism policies will be *strictly* enforced. For more details, see `http://www.ucl.ac.uk/current-students/guidelines/plagiarism`.
- You are not allowed to consult other people (outside of course staff) on this work. Each student has to work on the assignment individually.
- Your answers will be judged in terms of their quality, the depth of understanding, and also their brevity. Explain your answers clearly, but succinctly. Partial credit may be awarded.
- The assignment has a maximum of **100 marks** allocated as follows:

	Q1	Q2	Q3	Q4	Total
Marks	25	25	25	25	100 marks

Question 1: Cryptographic Software [25 marks]

This question is meant to prepare you for interacting with security software that you may not be familiar with. In particular, you will use OpenSSL (`https://www.openssl.org/`), a widely used open source cryptography library and tool, to perform certain cryptographic operations.

In many cases the OpenSSL tool comes already pre-installed with your operating system. If that's not the case, you can download it from their website or install it through your favorite package manager. Make sure you are using at least version OpenSSL 3.0.7 from 1 November 2022.

Use OpenSSL to answer the questions below. You are expected and encouraged to read the documentation of the library, and use the help commands (`openssl help` and `man openssl`). Unless stated otherwise, copy the full commands as well as any output from those commands to the provided `q1answer.sh` file and submit it together with the PDF containing your answers to the remaining questions. Make sure that the script runs without errors when executed in the same folder as the other provided files (see below). We also expect you to cite any sources fully.

(a) Check the version of OpenSSL you are using. **[1 mark]**

(b) You are provided with two encrypted zip files: `crypto1.zip.enc` and `crypto2.zip.enc`. One of those encrypted archives contains files required for the rest of the question. The SHA1 checksum of the correct file is `8ca275a98312ab337e6c0dbacb9c4675d0043b28`. Verify the checksum of the two files and identify the correct one. **[4 marks]**

(c) The archive with the above SHA1 fingerprint was encrypted using the AES256 block cipher, with a key derived from the password `cryptorulez` using the PBKDF2 key derivation function. Decrypt the encrypted zip file that you identified previously as the correct one. **[4 marks]**

Inside the zip file (no need to provide a command for unzipping), you will find an RSA public key `public-key.pem`, an associated timestamp file `key_ts.tsr`, as well as three certificates `cert1.pem`, `cert2.pem`, and `cert3.pem`.

(d) Use OpenSSL to output 16 bytes of random data as base64 and save it to a text file. **[1 mark]**

(e) Encrypt the random text file towards the provided RSA key. Output the result in base64. **[4 marks]**

Hint: You can use the pipe operator | and the OpenSSL `base64` function after your encryption command.

(f) Use OpenSSL to read the timestamp file. **[1 mark]**

(g) When was the timestamp created? What does that imply about the key file? (Answer as a comment in the provided shell file) **[2 marks]**

(h) Verify that the timestamp belongs to the public key. **[3 marks]**

Note: The timestamp file was created using `http://timestamp.digicert.com/`. The timestamp server's certificate and any intermediate certificates are included in the timestamp, but you will need the root certificate to complete the trust chain. You can download the root certificate from: `https://www.digicert.com/kb/digicert-root-certificates.htm`. It is called `DigiCert Assured ID Root CA` and has a SHA1 fingerprint of:
`05:63:B8:63:0D:62:D7:5A:BB:C8:AB:1E:4B:DF:B5:A8:99:B2:4D:43`

(i) One of the three certificates from the archive is valid; the other two would not be accepted by a modern browser. Read the three certificates using OpenSSL. For each certificate, state if it is valid or invalid. If invalid, state the reason. For your assessments assume the release date of the coursework assignment. **[5 marks]**

Question 2: Hash Functions [25 marks]

Let H be a hash function that is based on the Merkle-Damgård mode (with MD-finalization) and that uses a Davies-Meyer compression function f parameterized with 3TDEA (which is basically 3DES and specified in NIST Special Publication (SP) 800-67 Revision 2).

(a) What are the sizes of the IV, message blocks, and fingerprint of H and why? How are the mathematical maps for H and f defined (in terms of domain(s) and codomain(s))? **[4 marks]**

(b) Describe two different attacks on H and their asymptotic complexity in terms of O notation, and argue whether they are practically feasible. **[4 marks]**

(c) How would you change H to improve its security against the previously described attacks? How does the attack complexity change? **[2 marks]**

Furthermore, consider the following questions:

(d) Let $E : \{0,1\}^\lambda \times \{0,1\}^b \to \{0,1\}^b$ be a block cipher. Assume $\lambda = b$. Consider the following compression function

$$f(x, y) = E^{-1}(x, x \oplus y \oplus 10^{b-2}1) \oplus x \ .$$

where $10^{b-2}1$ is the b-bit sized bit string that starts with bit 1 followed by $b-2$ 0 bits and ends with bit 1. Is f second-preimage-resistant? Justify your answer either by constructing an attack or by providing a security proof. **[7 marks]**

Note: In the lectures, we denoted the application of a PRF/PRP F to a message m using a secret key k as $F_k(m)$. Using E here, this would translate to $E_x(y)$.

(e) Let $F : \{0,1\}^* \to \{0,1\}^\lambda$ and $G : \{0,1\}^* \to \{0,1\}^\lambda$ be hash functions, one of which is collision-resistant, and let

$$H(x) = F(x \parallel G(x)) \parallel G(x \parallel F(x)) \ .$$

Is H collision-resistant? Justify your answer either by constructing an attack or by providing a security proof. **[8 marks]**

Question 3: Digital Signatures [25 marks]

Consider the *textbook RSA signature scheme* with $N = 99301$ and $e = 5$.

(a) What are $\phi(N)$ and d and how did you compute them? **[2 marks]**

(b) What is the message for signature $\sigma = 2024$? **[2 marks]**

(c) On the example of the textbook RSA signature scheme, describe how you can speed up the signing process using the Chinese Remainder Theorem (CRT). **[4 marks]**

(d) Given the above parameters, compute the signature for message $m = 2024$ using the CRT method. Provide all intermediate computations and results. **[6 marks]**

Consider the *Schnorr signature scheme*. Recall that a Schnorr signature σ on a message m for private key x and public key $y = g^x$ is of the form $\sigma = (s, c)$ with $c = H(r \parallel m)$ and $s = k + cx$ where $r = g^k$ and k being a random value freshly chosen by the signer. Signature σ is considered valid if $c = H(y^{-c} \cdot g^s \parallel m)$.

Assume that the signer's hardware randomness generator is faulty and provides a fresh random value only on every second call. After returning a fresh random value k on call i, it may return the value $ak + b$ for some constants a and b with a certain probability p on call $i + 1$. You can assume that the very first call to the randomness generator after its initialization always produces fresh randomness.

(e) Show that an adversary can extract the secret key x if they are able to obtain two signatures σ_0 and σ_1 on two different messages m_0 and m_1 where the random values are connected as described above and assuming the adversary knows a, b, and y. **[6 marks]**

(f) Assuming the adversary collected n consecutive signatures from the signer with the faulty randomness generator where n is even. What is the probability in terms of p that the adversary can extract the private key x? **[3 marks]**

(g) Assuming $p = 2^{-14}$, how many signatures would the adversary have to collect to extract x with a probability of at least 25%? **[2 marks]**

Question 4: Existential Unforgeability [25 marks]

Let λ denote a security parameter. Let $\mathsf{MAC}_1 = (\mathsf{Gen}_1, \mathsf{Tag}_1, \mathsf{Verify}_1)$ and $\mathsf{MAC}_2 = (\mathsf{Gen}_2, \mathsf{Tag}_2, \mathsf{Verify}_2)$ be two (deterministic) message authentication codes for which it is know that at least one of them is EUF-CMA secure with respect to λ but it is unknown which one.

(a) Build a correct and EUF-CMA secure message authentication code $\mathsf{MAC} = (\mathsf{Gen}, \mathsf{Tag}, \mathsf{Verify})$ by combining MAC_1 and MAC_2 somehow. Provide specifications for Gen, Tag, and Verify and show that MAC is correct. **[5 marks]**

Note: To build MAC, proceed in the simplest way possible. In particular, do not introduce new cryptographic primitives.

(b) Prove that your MAC construction is EUF-CMA secure with respect to λ. **[20 marks]**