University of London

Computing and Information Systems/Creative Computing

CO3326 Computer security

Coursework assignment 2 2023–2024

Important

Each student has been allocated a unique set of data to use for this coursework assignment. You can obtain yours using your Student Reference Number (SRN) from the following URL: foley.gold.ac.uk/cw24/api/cw2/{srn}. For example, if your SRN is 887766554, you would obtain your data from http://foley.gold.ac.uk/cw24/api/cw2/887766554. If you have difficulties obtaining your data, please email us at: intcomp@gold.ac.uk

Introduction

The use of elliptic curves in cryptography was suggested independently by Neal Koblitz and Victor S. Miller in 1985. Elliptic curve cryptography (ECC) algorithms entered wide use between 2004 to 2005, and are found in numerous popular protocols, such as Transport Layer Security (TLS) and Bitcoin. ECC allows smaller keys as compared to other public-key cyptosystems (such as RSA), which explains its popularity. Hybrid encryption schemes, which combine the convenience of a public-key cryptosystem with the efficiency of a symmetric-key cryptosystem, have also integrated ECC. This has given rise to encryption schemes such as Elliptic Curve Integrated Encryption Scheme (ECIES), which are now widely used in a variety of web applications that require secure and efficient end-to-end encryption.

This coursework assignment is designed to extend your knowledge in this area by encouraging self-study and creativity. More specifically, through a practical exercise it requires you to encrypt and decrypt messages using a simplified version of ECIES. Wikipedia is a good starting point, and the internet has plenty of information on the subject, so you will not find it difficult to read up on the following topics:

- Elliptic curve cryptography (ECC)
- Elliptic curve integrated encryption scheme (ECIES).

The coursework assignment is composed of two parts, an **exercise** and a **report**. Each part carries 50 marks (total 100 marks). For the exercise you should decrypt two given ciphertexts, and re-encrypt the retrieved plaintext with another key. The keys are given, and you should use a simplified ECIES protocol for decryption/encryption. For the report you should answer the questions laid out below. They are designed to lead you through the key considerations for the exercise.

To solve the exercise, you may find it necessary to write a program. You are welcome to use any programming language and any third-party libraries available for SHA-256 and JSON. Libraries are available for most languages, including – and not limited to – Java, C/C++, Scala, Python, JavaScript, etc. Please include key snippets of your code as an annex to your report.

You should read the coursework assignment carefully and pay particular attention to the submission requirements.

Part A – Exercise

You have been provided with the ECC key pairs – private and public – of Alice and Bob, and two encrypted messages in a format that looks like the following (this is an example for illustration): http://foley.gold.ac.uk/cw24/api/cw2/887766554.

It is in JSON format. The `srn` and `name` fields should correspond to your details and are there for marking purposes. Under the `exercise` hash you find the key pairs of Alice (under `alice`) and Bob (under `bob`). The keys consist of a `privateKey`, which is a scalar, and a `publicKey`, which is an (x, y) point on the `SECP256K1` elliptic curve over the $\mathcal{F}_p$ $(\mathbb{Z}/p\mathbb{Z})$ field.

The `SECP256K1` curve and field parameters are available here: http://foley.gold.ac.uk/cw24/api/secp256k1. `p` is the field characteristic prime, `a` and `b` are the curve coefficients (`E`: $y^2 = x^3 + ax + b$), `g` is the base point and `n` is the subgroup order. This curve is also referred to as the Bitcoin curve.

The exercise consists of two parts. For the first part you are given two encrypted messages, one sent to Alice by Bob and one sent to Bob by Alice. These are under the `messages` hash. Each message includes the `cipher` text and an `r` that is needed by the receiving party to be able to decrypt the ciphertext. Your task is to decrypt these ciphertexts. You only need to use Alice's and Bob's private keys to do this, and you will be able to retrieve the corresponding plaintexts. If your decryption and decoding is correct you will get two English dictionary words.

For the second part Alice and Bob send each other back the message they received as an acknowledgment. So, you are expected to encrypt the messages that you decrypted in the first part of the exercise using the other party's public key.

In the encryption process there is a step where the encrypting party generates a random number. At this step Bob generates $\texttt{2E2F6E5E148013}_{16}$ and Alice generates $\texttt{3C6568F12E8047}_{16}$.

The solutions for both of the exercises should be included under a `solution` hash, which should be on the same level as the `exercise` hash. The two decrypted messages for the first part should be included under an `exercise1` hash. The two encrypted messages should be included under an `exercise2` hash. Each of the messages includes the `cipher` text, the corresponding plaintext (under `text`) and an `r` that is needed by the receiving party to be able to decrypt the message.

For the sample exercise above, the correct solution is: http://foley.gold.ac.uk/cw24/CarlDavis_887766554_CO3326cw2.json.

The symmetric encryption scheme that is used – in conjunction with the elliptic curve – is the XOR operation:

$$cipher = key \oplus encode(text)$$

The `key` is the `x` coordinate of the point that the encrypting party generates as a shared secret. This will become clear during your research. The `text` needs to be encoded as a number in order to be used in the XOR operation. If you use `Java`, you can rely on the following code to encode a string as a number:

```java
public BigInteger encode(final String text){
    return new BigInteger(text.getBytes(StandardCharsets.UTF_8));
}
```

You can use the following code to decode a number as a string:

```java
public String decode(final BigInteger number) {
    final BigInteger BYTE = BigInteger.valueOf(256L);
    if (number.equals(BigInteger.ZERO)) return "";
    return decode(number.divide(BYTE))
            + ((char) number.mod(BYTE).intValue());
}
```

You can also use the following web helpers to double-check your encoding and decoding:

- http://foley.gold.ac.uk/cw24/api/encode?text=conscience&base=16
- http://foley.gold.ac.uk/cw24/api/decode?number=636f6e736369656e6365&base=16

You can experiment with different bases (e.g. 16, 10, 2) for the encoding and decoding in the web helper. The base you use when encoding makes no difference from the perspective of the encryption (i.e. the XOR operation).

NOTE

All numbers in this coursework exercise are given and expected in hexadecimal form, written as strings (i.e. between double quotes).

Part B – Report

Please answer the questions briefly and in your own words. Use diagrams where possible and explain them. Copy-pasting Wikipedia articles or verbose explanations from ChatGPT will not get you very far. Your explanations should use the elliptic curve and points you have been given for the exercise. The context of the questions is Elliptic Curve Cryptography (ECC).

Question 1

Plot the `SECP256K1` elliptic curve and explain in simple terms the group law for elliptic curves.

Question 2

Demonstrate geometric addition and scalar multiplication with arbitrary points on the curve.

Question 3

Explain in simple terms, using your own words, how elliptic curves are restricted to a finite prime field $\mathtt{F}_p$.

Question 4

Explain in simple terms, using your own words, what the subgroup order is.

Question 5

Explain in simple terms, using your own words, what the base point is and how it is chosen.

Question 6

Demonstrate, with the aid of an example, geometric addition and scalar multiplication over the prime field $\mathtt{F}_p$.

Question 7

Explain in simple terms how the ECC keys – public and private – are generated.

Question 8

With the aid of an example show the encryption process step by step.

Question 9

With the aid of an example show the decryption process step by step.

Question 10

Is the Elliptic Curve Integrated Encryption Scheme (ECIES) susceptible to chosen-plaintext or chosen-ciphertext attacks? Why? Explain briefly using your own words.

Question 11

Briefly – in one paragraph – describe the design of your code. Attach the implementation of the encryption, decryption, encoding, and decoding methods. Don't forget to acknowledge any code re-use.

Reminder: do not forget to acknowledge all sources. Make sure you acknowledge any code re-use. It is important that your submitted coursework assignment is your own individual work and, for the most part, written in your own words. You must provide appropriate in-text citation for both paraphrase and quotation, with a detailed reference section at the end of your assignment. Copying, plagiarism, unaccredited and/or wholesale reproduction of material from books or from any online source is unacceptable, and will be penalised (see: How to avoid plagiarism). You may find it helpful to look at the end of some journal or conference papers to get an idea of how to list your reference material appropriately. The Harvard Referencing Guide provides a short explanatory introduction and a checklist of examples, showing how to cite and reference material from various sources.

Submission requirements

You should upload **two** single files only. These must not be placed in a folder, zipped, etc.

The **report** should be submitted as a PDF document using the file naming conventions below: `YourName_SRN_COxxxcw#.pdf`, for example `CarlDavis_887766554_CO3326cw2.pdf`. `YourName` is your full name as it appears on your student record (check your student portal), `SRN` is your Student Reference Number, for example `887766554`, `COXXXX` is the course number, for example `CO3326`, and `cw#` is either `cw1` (coursework 1) or `cw2` (coursework 2).

The **exercise** should be submitted as a JSON file with a *strict format* and *naming scheme*. The exercise will be automatically checked by an algorithm, so pay particular attention to its format. The name of the file should be `YourName_{srn}_CO3326cw2.json`; for example, Carl Davis with SRN 887766554 would submit `CarlDavis_887766554_CO3326cw2.json`.

Note: as the JSON is evaluated by an algorithm, every quote, comma, colon, curly brace upper/lower case is crucial. Please pay attention to these, and check your JSON very carefully against the sample solution provided.

It would be a shame to lose a potential **50%** of the total marks for this coursework assignment because of a misplaced comma or a missing quote, etc. Note: there are also online tools you can use for JSON formatting and validation, for example https://jsonformatter.curiousconcept.com/, so double-check that your JSON is syntactically correct.

[END OF COURSEWORK ASSIGNMENT 2]