University of London

Computing and Information Systems/Creative Computing

CO3326 Computer security

Coursework assignment 1 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/cw1/{srn}. For example, if your SRN is 887766554, you would obtain your data from http://foley.gold.ac.uk/cw24/api/cw1/887766554. If you have difficulties obtaining your data, please email us at: intcomp@gold.ac.uk

Introduction

Dual_EC_DRBG (Dual Elliptic Curve Deterministic Random Bit Generator) is an algorithm that was presented as a cryptographically secure pseudorandom number generator (CSPRNG) using methods in elliptic curve cryptography. Despite wide public criticism, including the public identification of the possibility that the National Security Agency (NSA) put a backdoor into a recommended implementation, it was for seven years one of four CSPRNGs standardised in NIST SP 800-90A (as originally published in June 2006, until it was withdrawn in 2014).

Concerns about the integrity of Dual_EC_DRBG had been raised by the cryptographic community even before the Snowden revelations in 2013. After the disclosures by Edward Snowden, it became evident that the NSA had actively worked to promote the adoption of Dual_EC_DRBG as a standard, despite its known weaknesses and suspicions about a potential backdoor.

In the aftermath of the scandal, there were calls for the removal of Dual_EC_DRBG from cryptographic standards, and many organisations and developers moved to alternative algorithms with better security assurances. The incident highlighted the importance of open and transparent processes in the development of cryptographic standards and the potential risks associated with relying on algorithms that may have been influenced by intelligence agencies for undisclosed purposes.

Nevertheless, the idea of designing a cryptographically secure pseudorandom number generator based on elliptic curve cryptography and two fixed points remains salient. This coursework assignment is designed to extend your knowledge in this area by encouraging self-study and creativity. More specifically, you will create your own Dual_EC_DRBG protocol and build in a backdoor in a similar way the NSA was alleged to have done. A short overview of the Dual_EC_DRBG is provided below. In addition, 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)
- Cryptographically secure pseudorandom number generator (CSPRNG)
- Dual_EC_DRBG.

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 generate a sequence of five pseudorandom numbers starting from an initial state, and then extrapolate from two intercepted pseudorandom numbers the subsequent numbers, having built in a backdoor to the protocol. 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.

Dual_EC_DRBG overview

The notation used in this coursework is consistent with what you find in most sources online. The algorithm uses a single integer s as state. Whenever a new random number is requested, this integer is updated. The k-th state is given by:

$$s_k = g_P(s_{k-1})$$

The returned random integer r is a function of the state. The k-th random number is:

$$r_k = g_Q(s_k)$$

The function $g_P(x)$ depends on the fixed elliptic curve point P; $g_Q(x)$ is similar except that it uses the point Q. The points P and Q stay constant for a particular implementation of the algorithm.

The algorithm allows for different constants, variable output length and other customisation. For this coursework assignment we use the constants from curve `SECP256K1` (also referred to as the Bitcoin curve). The algorithm operates over a finite field $\mathcal{F}_{prime}$ ($\mathbb{Z}/prime\mathbb{Z}$) where *prime* is a prime number. The state, the seed and the random numbers are all elements of this field. The field size is:

$\texttt{fffffffffffffffffffffffffffffffffffffffffffffffffffffffefffffc2f}_{16}$

An elliptic curve over $\mathcal{F}_{prime}$ is defined as `E`: $y^2 = x^3 + ax + b$. As we are using the `SECP256K1` curve in this coursework, we have $a = 0$ and $b = 7$. The set of all points on the curve is defined as $E(\mathcal{F}_{prime})$. Two of these points are P and Q: $P, Q \in E(\mathcal{F}_{prime})$.

A function to extract the x-coordinate is used, which 'converts' from elliptic curve points to elements of the field:

$$X(x, y) = x$$

The least significant 16 bits of the output integers are truncated before being output with a simple bitwise right shift operation:

$$t(x) = x \gg 16$$

The functions g_P and g_Q are defined as:

$$g_P(x) = X(x \cdot P)$$
$$g_Q(x) = t(X(x \cdot Q))$$

where $x \cdot P$ (or $x \cdot Q$) is scalar multiplication by x of point P (or Q) on the elliptic curve over $\mathcal{F}_{prime}$.

The generator is seeded with a *nonce* (number used once) value from the field $\mathcal{F}_{prime}$:

$$s_1 = g_P(nonce)$$

The k-th state and random number are:

$$s_k = g_P(s_{k-1})$$
$$r_k = g_Q(s_k)$$

The sequence of random numbers are:

$$r_1, r_2, \ldots$$

Part A – Exercise

You have been provided with the parameters of the elliptic curve and the prime finite field that the algorithm operates on, together with the two fixed points – `P` and `Q` – on the curve that the pseudorandom number generator relies upon, in a format that looks like the following (this is an example for illustration): http://foley.gold.ac.uk/cw24/api/cw1/887766554.

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 a `dualEcDrbg` hash, which defines the elliptic curve coefficients `a`, `b`, and the field characteristic `prime`. The curve (hence `a`, `b` and `prime`) is the same for all students, but the points `P` and `Q` are different for each student. Therefore, while `a`, `b` and `prime` are the same, your `P` and `Q` are different from the points provided in the example.

The exercise consists of two parts. For the first part you are given a `nonce` (unique to you) that initialises the state, and the exercise requires you to generate the first five pseudorandom numbers (r_1, r_2, r_3, r_4, and r_5) according to your Dual_EC_DRBG protocol. For the second part you are given a sequence of two consecutive `intercepted` pseudorandom numbers that your Dual_EC_DRBG protocol generates, and the `factor` that you built in as a backdoor into your protocol, as the NSA allegedly did. You are expected to find the next three 'random' numbers.

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 sequence of the five generated pseudorandom numbers starting from `nonce` should be included under the `exercise1` hash, and the sequence of all five pseudorandom numbers following on from – and including – the two `intercepted` pseudorandom numbers should be included under the `exercise2` hash. Both `exercise1` and `exercise2` expect arrays of hexadecimal numbers. For the sample exercise above, the correct solution is: http://foley.gold.ac.uk/cw24/CarlDavis_887766554_CO3326cw1.json.

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 cryptographically secure pseudorandom number generation using elliptic curve cryptography and Dual_EC_DRBG.

Question 1

Explain in your own words the elliptic curve discrete logarithm problem (ECDLP).

Question 2

Consider the elliptic curve given by the equation `E`: $y^2 = x^3 - 3x + 2$ over $\mathbb{R}$. Is this a singular curve? Explain why.

Question 3

Explain why singular elliptic curves are bad for a discrete logarithm based cryptosystem.

Question 4

Consider the elliptic curve `E`: $y^2 = x^3 + 7$ over the field $\mathcal{F}_{43}$.

i. For which values of $x \in \mathcal{F}_{43}$ is $x^3 + 7$ a perfect square in $\mathcal{F}_{43}$?
ii. Demonstrate the addition $(13, 22) \oplus (21, 25)$ in the group of the curve.
iii. Demonstrate the scalar multiplication $17 \cdot (13, 21)$.
iv. Demonstrate the scalar multiplication $31 \cdot (12, 12)$. What is the significance of this?
v. List all points on `E`, i.e. list $E(\mathcal{F}_{43})$.
vi. What is the order of the curve? What is the cofactor? Explain why.

Question 5

Other than the alleged backdoor, what has been the other criticism of the Dual_EC_DRBG protocol as originally put forward by NIST?

Question 6

Explain how you can reverse the truncation to fin x, and how you can work out whether x comes from a legitimate point on the curve in the field. Is this enough to crack the protocol without the backdoor? Explain why.

Question 7

Explain what the `factor` you were given actually is and how it can be used as a backdoor.

Question 8

Explain step-by-step how you compute the next pseudorandom number using the backdoor and the two consecutive intercepted numbers you were given.

Question 9

What is the implication of being able to guess what the next pseudorandom number generated by the protocol?

Question 10

Briefly – in one paragraph – describe the design of your code. Attach the implementation of the scalar multiplication and the method that solves the next pseudorandom number given the two preceding ones. 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 (this should not be included in the word count). 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_CO3326cw1.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}_CO3326cw1.json`; for example, Carl Davis with SRN 887766554 would submit `CarlDavis_887766554_CO3326cw1.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 1]