C&O 487/687: Assignment #4

Due date: November 17, 2022 (Thursday)

Note: Some questions use randomization to customize to you specifically. Please include your max-8-character UW user id (`ss2liang`) at the beginning of your answer so we can look up your custom solution.

1. [8 marks] **Hybrid encryption.**

   Recently, the University of Waterloo LEARN server was hacked by a group calling themselves "Really Super Awesome Association of Engineering Students" (RSA-AES), a terrorist organization bent on world domination. The first phase of their evil plan is to ensure that engineering students have better grades and hence get better co-op jobs than math students, so they hacked into the LEARN server and encrypted this CO 487 assignment in hopes of causing you to get a lower mark in this course. Luckily for you, they made some mistakes during encryption because they did not take CO 487.

   Your first task for this assignment is to decrypt the encrypted assignment PDF so that you can actually do the rest of your assignment.[1]

   I managed to capture some logs showing some of the Python code they used to encrypt the assignments, which you might find helpful in figuring out how to break their encryption. This is in the file `a4q1_attacker_log.py` on LEARN.

   You might have to do some number theory operations in Python as you try to solve this question. Here are a few functions that might be helpful:

   - `pow(x, y, z)`: computes $x^y \bmod z$
   - `factorint(x)`: attempts to factor the integer x, and if successful returns the factors and their multiplicities; see documentation at `https://docs.sympy.org/latest/modules/ntheory.html?highlight=factorint#sympy.ntheory.factor_.factorint`
   - `mod_inverse(x, z)`: computes $x^{-1} \bmod z$

   `pow` is included with Python by default, but to get `factorint` and `mod_inverse`, you have to install an additional library called `sympy` (try running `pip3 install sympy`) and then importing those functions into your program using the following source code:

   ```
   from sympy import mod_inverse
   from sympy.ntheory import factorint
   ```

   (a) [4 marks] By inspecting `a4q1_attacker_log.py`, describe 4 cryptographic mistakes made by the hackers and what you would do differently.

   (b) [3 marks] Describe the procedure you used to decrypt the file. Submit any code you write through Crowdmark, as you would a normal assignment – as a PDF or screenshot. We will be reading it, but not executing it.

   (c) [1 marks] To prove that you successfully decrypted the file, copy the following random number into your solution: 51493

   Please include your max-8-character UW user id (`ss2liang`) at the beginning of your answer so we can look up your custom solution.

[1] You can also download the decrypted version of (most of) the assignment from LEARN if you want to get started on other questions, but the decrypted version from LEARN is missing a part (worth a few marks) that can only be obtained by decrypting your own customized encrypted assignment.

2. [4 marks] **Diffie–Hellman equivalents.**

   Let p be a prime and let g be an element of large prime order q in $\mathbb{Z}_p^*$. Show that the Diffie–Hellman assumption is equivalent to the *square Diffie–Hellman assumption*, which is that: let a be chosen uniformly at random from $\mathbb{Z}_p^*$. Given g and g^a, it is computationally infeasible to determine g^{a^2}.

   To do so, you need to show two directions: DH assumption $\implies$ square DH assumption, and vice versa.

   Use the contrapositive. For example, the contrapositive of the forward direction is: if there is an efficient algorithm $\mathcal{A}$ that breaks the square DH assumption, then we can use $\mathcal{A}$ to break the DH assumption.

   Note that the reverse direction is tricky than it may first appear.

3. [4 marks] **ElGamal encryption.**

   Let G be a finite group of order q, and suppose that $g \in G$ *generates* G. This means that $G = \{g^0, g^1, \ldots, g^{q-1}\}$, i.e., every element of G can be written as g^r for some $r \in \mathbb{Z}_q$.

   The ElGamal encryption scheme uses key pairs of the form (z, g^z), where $z \in \mathbb{Z}_q$ is the private key and g^z is the public key. The encryption of a message $m \in G$ under the public key g^z is $(c_0, c_1) = (g^r, m(g^z)^r)$, where $r \in_R \mathbb{Z}_q$ is chosen randomly for each encryption.

   (a) [2 marks] Suppose that $\mathcal{O}_D$ is an efficient decryption oracle for ElGamal under arbitrary public keys. That is, given the inputs Z and (c_0, c_1), where $Z = g^z$ and $(c_0, c_1) = (g^r, m(g^z)^r)$ for any $z, r \in \mathbb{Z}_q$, the oracle $\mathcal{O}_D$ returns m.

   Recall the Diffie–Hellman problem: given g, g^x, and g^y, compute g^{xy}. Suppose that you are given $X, Y \in G$, where $X = g^x$ and $Y = g^y$. (The generator g is known to you, but x and y are not.) Describe how to efficiently compute g^{xy} using a single call to $\mathcal{O}_D$.

   You may assume that computing inverses in G is efficient.

   (b) [2 marks] Now, suppose that $\mathcal{O}_{DH}$ is an efficient Diffie–Hellman oracle. That is, given the inputs $X = g^x$ and $Y = g^y$, the oracle $\mathcal{O}_{DH}$ returns g^{xy}.

   You are given an ElGamal public key $Z = g^z$ and an ElGamal ciphertext $(c_0, c_1) = (g^r, m(g^z)^r)$. (The values X, c_0, and c_1 are known to you, but z, r, and m are not.) Describe how to efficiently compute m using a single call to $\mathcal{O}_{DH}$.

   You may assume that inversion and multiplication in G can be computed efficiently.

4. [9 marks] **Side-channel attacks.**

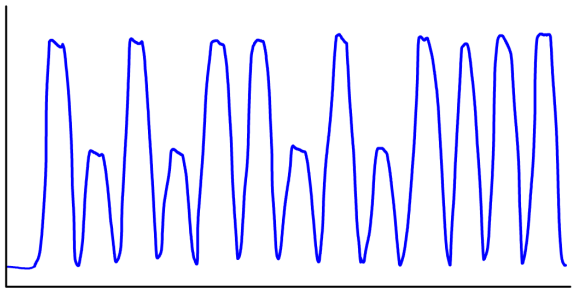
   In this problem, you will explore *side channel attacks* on elliptic curve cryptography. A side channel attack targets a flaw in an implementation of a cryptosystem which may be secure in theory.

   A *hardware security module*, or *HSM*, is a type of computer which is specifically designed to store cryptographic keys and perform cryptographic operations in a secure fashion. An HSM provides an interface whereby an authenticated user can request that operations such as hashing, signing, and key generation can be performed. Usually, an HSM will not expose secret values to users. For example, if Alice requests that an RSA key pair be generated, an HSM will securely store both the public and the private key but output only the public key. Later, Alice can decrypt messages using the stored private key by authenticating to the HSM and providing the message to be decrypted.

   For this question, suppose that Alice is using an HSM to perform elliptic curve Diffie-Hellman key exchange, with point multiplication implemented using the iterative double-and-add algorithm. (See slides 21 and 23 of Topic 3.6.) You have sneakily installed a device on the HSM's power supply which allows you to observe its power consumption.

   By observing Alice's network traffic, you deduce that at `04:08:07` UTC time, she performed an elliptic curve Diffie-Hellman key exchange with Bob.

   Suppose that you observe the following graph for the power consumption of Alice's HSM at `04:08:07`:



(a) [2 marks] Determine the most significant byte (8 bits) of Alice's private key, and explain how you obtained your answer.

(b) [3 marks] Let E be the elliptic curve $y^2 = x^3 - x + 3$ over the field $\mathbb{Z}_7$. Let $P = (2, 3)$ and $Q = (0, 6)$ be points on E. Compute the following using the formulas from class:

i. $P + P$

ii. $P + Q$

iii. $Q + Q$

(c) [3 marks] Now, perform the same calculations using

$$m = \frac{(x_P + x_Q)^2 - x_P x_Q + a}{y_P + y_Q}$$

for both addition and doubling.

(d) [1 marks] How could the HSM's implementation of double-and-add be modified to prevent the side-channel attack from part (a)? You may assume that $y_P + y_Q \neq 0$ unless $P = -Q$.

5. [5 marks] **DSA.**

Recall that in the signing step of the DSA, the signature must be regenerated in any of k, r, and s are 0.

(a) [1 marks] What problems arise if $k = 0$?

(b) [2 marks] Show how an adversary can forge a signature on any message if signatures with $r = 0$ are allowed, and said adversary intercepts a signature of the form $(0, s)$. Explain why this forgery is a valid signature.

(c) [2 marks] Show how an adversary can forge a signature on any message if signatures with $s = 0$ are allowed, and said adversary intercepts a signature of the form $(r, 0)$. Explain why this forgery is a valid signature.

Academic integrity rules

You should make an effort to solve all the problems on your own. You are also welcome to collaborate on assignments with other students in this course. However, solutions must be written up by yourself. If you do collaborate, please acknowledge your collaborators in the write-up for each problem. *If you obtain a solution with help from a book, paper, a website, or any other source, please acknowledge your source. You are not permitted to solicit help from other online bulletin boards, chat groups, newsgroups, or solutions from previous offerings of the course.*

Due date

The assignment is due via Crowdmark by 8:59:59pm on November 17, 2022. Late assignments will not be accepted.

Office hours

Office hours will take place online via the Gather.town platform; see the link on LEARN under Contents → Course Information → Office hours.

- Monday November 7 11am–12pm
- Thursday November 10 1–2pm
- Monday November 14 11am–12pm
- Tuesday November 15 11am–12pm
- Wednesday November 16 11am–12pm
- Wednesday November 16 2–3pm
- Wednesday November 16 4–5pm
- Thursday November 17 1–2pm

Changelog

- Tues. Nov. 8: assignment posted