C&O 487/687: Assignment #4

Due date: November 14, 2024 (Thursday)

If you are requesting an extension or accommodation because of a verification of illness form, self-declared student absence, or AccessAbility Services accommodation, see the "Missed assignment / extension policy and forms" instructions on the CO 487/687 LEARN site.

Throughout this assignment, you make use of the following facts about number theory:

- Fact 1: If $a, b \in \mathbb{Z}_p^*$ and p is prime, then ab is a square modulo p if and only if either a and b are both squares modulo p, or a and b are both non-squares modulo p.
- Fact 2: If p is prime, then exactly half the elements in $\mathbb{Z}_p^*$ are squares.
- Fact 3: There exists an efficient algorithm for determining if an element a has a square root modulo p.
- Fact 4: There exists an efficient algorithm for computing square roots modulo p, if it exists.

1. [9 marks] **Hybrid encryption.**

   Recently, the University of Waterloo LEARN server was hacked by the Really Super Awesome Association of Engineering Students (RSA-AES), who are continuing their maniacal plot for world domination. The next 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 $x^y \bmod z$
   - `isprime(x)`: determines whether the integer x is prime; see `https://docs.sympy.org/latest/modules/ntheory.html#sympy.ntheory.primetest.isprime`
   - `nextprime(x)`: returns the next prime number larger than x; see `https://docs.sympy.org/latest/modules/ntheory.html#sympy.ntheory.generate.nextprime`
   - `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#sympy.ntheory.factor_.factorint`
   - `mod_inverse(x, z)`: computes $x^{-1} \bmod z$
   - `decimal.getcontext().prec = 100; decimal.Decimal(x).sqrt()`: compute high-precision square roots of real numbers

   `pow` is included with Python by default. To get `decimal`, you have to put `import decimal` at the top of the program. To get the rest of the functions, 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:

[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.

```
from sympy import mod_inverse
from sympy.ntheory import isprime, nextprime
```

Note that all of the required packages are installed on the UW math Jupyter notebook server `https://jupyter.math.uwaterloo.ca`, but you need to make sure you use the "Python 3 extra" kernel.

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

(b) [4 marks] Describe the procedure you used to decrypt the file. In your description, focus on the main cryptographic break you found. 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: 18658

This question uses randomization to customize to you specifically. Please include your max-8-character UW user id (`j394zhao`) at the beginning of your answer so we can look up your custom solution.

2. [9 marks] **Diffie–Hellman and discrete logarithms**

Let G be a cyclic group of prime order q, with generator g; this information is publicly known. Consider the following three computational hardness assumptions:

- **Decisional Diffie–Hellman (DDH):** Given either (g^a, g^b, g^{ab}) or (g^a, g^b, g^c), where a, b, and c are random, distinguish whether you were given a triple of the first form or a triple of the second form.
- **Computational Diffie–Hellman (CDH):** Given g^a and g^b, where a and b are random, compute g^{ab}.
- **Discrete Logarithm Problem (DLP):** Given g^a, where a is random, compute a.

(a) [2 marks] Show that, if the DDH problem is hard, then the CDH problem must also be hard.
Hint: Prove the contrapositive. Assume that there exists an algorithm, $\mathcal{A}$, which can successfully solve CDH, and write an algorithm, $\mathcal{B}$, that has blackbox access to $\mathcal{A}$, which can solve the DDH problem.

(b) [2 marks] Show that, if the CDH problem is hard, then the discrete logarithm problem must also be hard.
Hint: Prove the contrapositive. Assume that there exists an algorithm, $\mathcal{A}$, which can successfully solve DLP, and write an algorithm, $\mathcal{B}$, that has blackbox access to $\mathcal{A}$, which can solve the CDH problem.

(c) [5 marks] It is not the case that DDH is hard in every group. Suppose our adversary has an oracle, O, which tells them whether a given input is a square modulo p (where p is prime). For instance, if $p = 7$, then $O(2) = true$ since $3^2 \equiv 2 \mod 7$, but $O(3) = false$, since 3 is not a square modulo 7. Write an adversary $\mathcal{A}$, with access to O,[2] which can solve DDH in the group $\mathbb{Z}_p^*$ (for prime p). *Hint: Your adversary does not have to succeed with probability 1; it is enough that they succeed with probability $\frac{1}{2} + x$, where x is non-negligible.*

[2]Giving the adversary such an oracle is not as outlandish as it may seem. The Legendre symbol is an efficiently computable function from elementary number theory which returns 1 if the input is a square modulo p, -1 if the input is not a square modulo p, and 0 if the input is divisible by p. So, in the real world, every adversary has access to O!

3. [4 marks] **Backdoored PRG**

   A *pseudorandom generator* (PRG) is a deterministic function that expands a short random seed into a longer random looking sequence. We will construct a pseudorandom generator from a deterministic function f that takes as input a state s_{i-1} and produces a new state s_i and outputs $t_i \in S$ for some set S.

   To produce a long pseudorandom sequence of ℓ numbers from set S, start with an initial seed s_0, iterate the function as many times as needed by computing

   $$(s_i, t_i) \leftarrow f(s_{i-1})$$

   for $i = 1, \ldots, \ell$, and output the final output $t_1, \ldots, t_\ell \in S^\ell$

   We will design a PRG based off of discrete logarithms. The following information is public:

   - A prime number p, an element $g \in \mathbb{Z}_p^*$ of large prime order q, and the the order q.
   - Another element h in the subgroup generated by g (in other words, $h = g^u \bmod p$ for some u; but u is *not* given.

   Our PRG is based on the following function f, defined as follows:

   $$f(s_{i-1}) = (s_i, t_i)$$

   where

   $$r_i = g^{s_{i-1}} \qquad s_i = g^{r_i} \qquad t_i = h^{r_i}$$

   (a) [1 marks] For parameters $p = 167$, $g = 21$, $q = 83$, and $h = 6$, and seed $s_0 = 123$, produce the output for $\ell = 3$ (produce the values t_1, t_2, t_3).

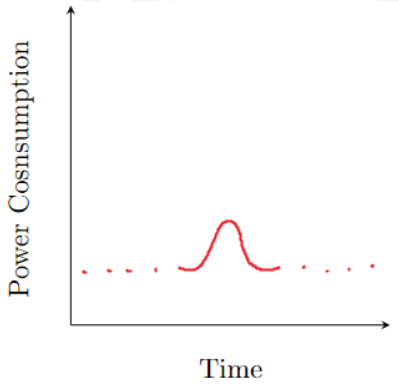
   (b) [2 marks] It is possible to show that, for well-chosen parameters, this PRG is secure under the assumption that the Diffie–Hellman problem is hard. If, however, the adversary knows the value u such that $h = g^u$, called a backdoor, the PRG is not secure. Suppose the value of t_1 had been published. Explain how, for arbitrary parameters meeting the conditions described, an adversary could use u and t_1 to find s_1, and use this to predict t_2. Justify your response. Verify your attack works for the above examples; you may make use of the backdoor I chose while generating this question: namely, that $6 \equiv 21^{39} \bmod 167$.

   (c) [1 marks] Suggest a countermeasure to the attack of part (b).

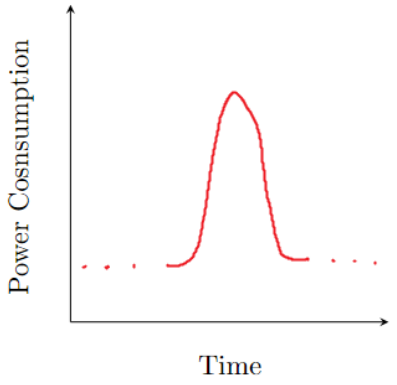
4. [9 marks] **Side channel attack on elliptic curve cryptography**

   Eve and Mallory, the leaders of the Engineering Committee for Digital Hijinks (ECDH), are up to no good. Luckily, while proctoring one of Eve's midterms, a clever CO 487 TA managed to sneakily attach a (very small and undetectable) device to Eve's phone that measures power consumption. By observing Eve's network traffic, the TAs were able to deduce that on Friday, November 1, 2024, at 07:13:41 UTC time, she performed an elliptic curve Diffie–Hellman key exchange with Mallory.

   While the power consumption measurement tool is impressive, it can only do so much. Modular addition, modular subtraction, and modular reduction don't require enough power to produce any meaningful output on the device; however, modular multiplication and the computation of modular inverses do. On a graph of power consumption versus time, multiplication and modular inverse operations look roughly as follows:

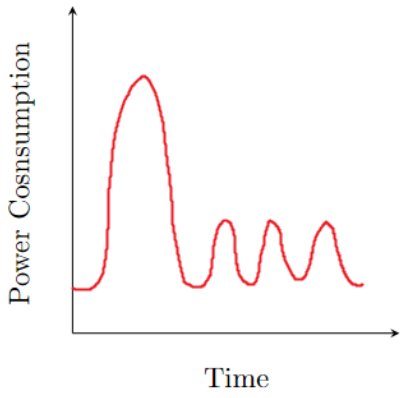


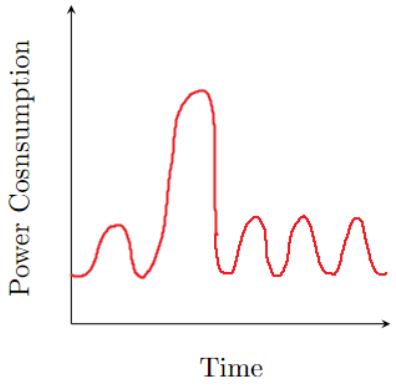
Modular multiplication power usage



Modular inversion power usage

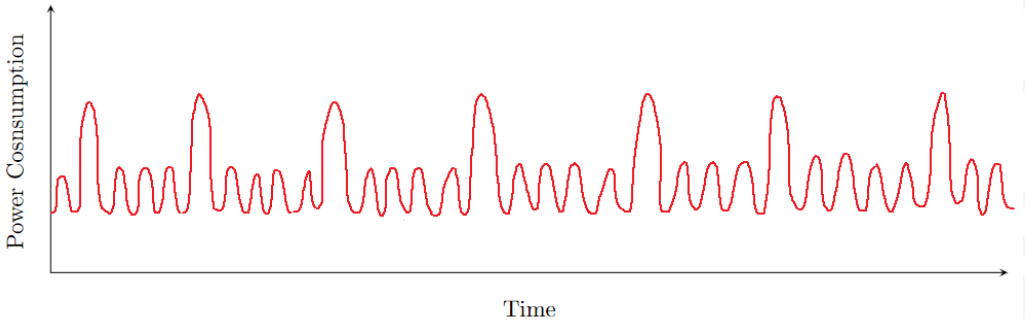
(a) Let P and Q be distinct points on an elliptic curve, and consider the following two power consumption graphs:





Of the graphs above, one is the result of computing $P + Q$ and the other is the result of computing $P + P$. Which is which? Justify your reasoning.

(b) Here is the beginning of the power consumption graph that the TAs obtained from Eve's phone on Friday, November 1, 2024, at 07:13:41 UTC time:



All operations captured in the graph were performed before sending the first message of the elliptic curve Diffie–Hellman key exchange to Mallory. Based on the operations performed, determine the first (i.e. most significant) 4 bits of Eve's secret key. Justify your reasoning.

(c) Let E be the elliptic curve $y^2 = x^3 + 3x + 2$ over $\mathbb{Z}_{11}$. Let $P = (2, 4)$ be a point on E. Compute $9P$. Show your work. You may use computational tools as needed.

(d) Suggest a modification to the algorithm used to compute scalar multiplication of elliptic curve points that would prevent the side channel attack that you carried out in part (b).
Hint: In some cases, your algorithm might perform more operations than the one you learned in class.

5. [6 marks] **Learning with errors**

Here is simplified version of the LWE encryption scheme.

- The parameters n and q are positive integers, q a prime.
- The secret key (s, e) is a pair of random vectors with components in $\{-2, -1, 0, 1, 2\}$. The vectors s and e both have n components.
- The public key is a pair (A, b), where A is a uniformly random square matrix of $\mathbb{Z}_q^{n \times n}$ and $b = As + e$.

Note that the matrix A has a very high probability of being invertible when sampled uniformly at random; for the rest of this question, you can assume that A is invertible (modulo q). You can also assume that $A + I$ is invertible (modulo q) with high probability.

The encryption algorithm is the following. Given a plaintext $m \in \{0, 1\}$:

i. Sample a random vector $s' \in_R \{-2, -1, 0, 1, 2\}^n$.
ii. Sample a random vector $e' \in_R \{-2, -1, 0, 1, 2\}^n$.
iii. Sample a random value $e'' \in_R \{-2, -1, 0, 1, 2\}$.
iv. Compute $c_1 = s'A + e' \bmod q$.
v. Compute $c_2 = s'b + e'' + \lfloor \frac{q}{2} \rfloor m \bmod q$.
vi. Return the ciphertext (c_1, c_2).

(a) In this part, let $n = 4$ and $q = 3329$. The following public key was generated:

$$A = \begin{pmatrix} 1855 & 3164 & 2694 & 2409 \\ 1038 & 2187 & 922 & 2111 \\ 3116 & 772 & 2743 & 217 \\ 1613 & 842 & 3260 & 256 \end{pmatrix} \quad , \quad b = (674, 2069, 399, 2489)$$

and the corresponding secret key is

$$s = (-1, 1, 1, 0).$$

The message $m = 1$ was encrypted to

$$c_1 = (220, 1211, 2479, 1812) \quad , \quad c_2 = 1799.$$

Write out the steps showing the decryption of (c_1, c_2) using s to verify that it successfully recovers $m = 1$. You may use computational tools as needed.

As the name "learning with errors" suggests, this scheme comes with two error terms e, e'. It is natural to ask if we really need these terms. The goal of rest of this question is to prove that we indeed need them.

(b) Prove that if, instead of being random, e was always zero, the scheme would not be secure.
(c) Prove that if, instead of being random, e' was always zero, the scheme would not be secure.
(d) What if s what sampled as required, but we fixed e to be equal to s? Would it be secure?
(e) What if s' what sampled as required, but we fixed e' to be equal to s'? Would it be secure?

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 11:59:59pm on November 14, 2024. Late assignments will not be accepted.

Changelog

- Wed. Nov. 6: Added some useful number theory facts at the top of the assignment.