1. (0 points) READ the syllabus- Content and Policy. What is your name? Write it clearly. What is the day and time of the first midterm? Read slides on Dr. Mazurek’s lecture.

2. (25 points) Write a simple program which does the following:

   (a) INPUT: A key K, a nonce N, and a text string M
   (b) OUTPUT: Ciphertext corresponding to M encrypted under AES256-GCM (i.e. the AES algorithm with key length 256 in GCM mode) with K as the key and N as the IV.

Do this two ways and WRITE IN ENGLISH the contrast of experience: Include your code, an input of your choice, and the corresponding output. You have TWO choices:

I) Do both in PYTHON:

   (a) Cryptography library on the hw website, and
   (b) PyCrypto on the hw website

II) Do both in C (which would be harder)

   (a) C via OpenSSL on the hw website, and
   (b) libsodium on the hw website

SOLUTION TO PROBLEM TWO

Omitted

THERE ARE MORE PAGES!!!!!!!!!!!!!!!!!!!
3. (20 points) Let $N = pq$ where $p, q$ are primes. Let $m \in \{2, \ldots, N - 1\}$.

(a) (4 points) Exactly how many multiplications do you need to compute $m^{2^{16}+1}$ using repeated squaring.

(b) (4 points) Exactly how many multiplications do you need to compute $m^{2^{16}-1}$ using repeated squaring.

(c) (0 points, this is just here for information) If you did the last two problems right then $m^{2^{16}+1}$ took MUCH LESS muts than $m^{2^{16}-1}$. This is one reason why $e = 2^{16} + 1$ is so popular in RSA.

(d) (4 points) $2^{16} + 1$ is prime. Is $2^{32} + 1$ prime? If not then give its factors. (HINT- look up Fermat Primes on the web)

(e) (4 points) Why is choosing $e$ to be prime a good thing to do?

(f) (4 points) I had said in class that we do not want to pick $e$ too low. Roughly how big does $N$ have to be before picking $e = 2^{16} + 1$ is a bad thing to do. How does this $N$ compare to the number of protons in the universe? (Look up Eddington’s Number on the web)

SOLUTION TO PROBLEM TWO

a) All computations are mod $p$.

We compute:

$m^2$

$(m^2)^2 = m^4$

$(m^4)^2 = m^8$

$(m^8)^2 = m^{16}$

So to get to $m^{2^i}$ takes $i$ multiplications.

Hence $m^{2^{16}}$ takes 16 muts.

So $m^{2^{16}+1} = m^{2^{16}} \cdot m$ takes 17 muts.

b) Note that $2^{16} - 1 = 2^0 + 2^1 + \cdots + 2^{15}$.

We first compute, by repeated squaring, $m^{2^i}$ for $1 \leq i \leq 15$. That takes 15 muts.

But then we have to do
\[ m^2 \times m^2 \times \cdots \times m^{15} \]

which takes another 14 mults. Hence the total is 29.

d) \( 2^{2^5} + 1 = 641 \times 6700417 \)

e) We need \( e \) to be rel prime to \( R \). If \( e \) is prime then it is AUTOMATICALLY rel prime to \( R \).

f) If Bob sends \( m = 2 \) then this is a problem if \( m^e < N \). So we have a problem if \( 2^{65537} < N \), so \( N \sim 2^{65537} \). The number of particles in the universe is approx \( 2^{256} \) which is Much smaller.

THERE ARE MORE PAGES!!!!!!!!!!!!!!!!!!!!!
4. (25 points) (HINT — look up the Chinese Remainder Theorem.) Give an algorithm (pseudocode but more descriptive) for the following:

**Input:** $N_1, \ldots, N_L, x_1, \ldots, x_L$ where $N_1, \ldots, N_L$ are rel prime.

**Output:** An $x$ such that

\[ x \equiv x_1 \pmod{N_1} \]
\[ x \equiv x_2 \pmod{N_2} \]
\[ \vdots \]
\[ x \equiv x_L \pmod{N_L} \]

AND $0 \leq x < N_1 \cdots N_L$.

You can assume you have a program that finds inverses of numbers in mods if they exist.

Note that since all of the $N_i$ are rel prime, for all $i$ there exists a number which you can denote $M_i^{-1}$ which is the inverse of $M_i$ mod $N_i$, where $M_i = N_1 N_2 \cdots N_{i-1} N_{i+1} \cdots N_L$.

**SOLUTION TO PROBLEM FOUR**

(a) Input($N_1, \ldots, N_L, x_1, \ldots, x_L$)
(b) Let $M_i = N_1 N_2 \cdots N_{i-1} N_{i+1} \cdots N_L$.
(c) For all $1 \leq i \leq L$ find $M_i^{-1}$ which is the inverse of $M_i$ mod $N_i$
(d) Output

\[ x = x_1 M_1^{-1} M_1 + \cdots + x_L M_L^{-1} M_L \pmod{N_1 \cdots N_L} \]

We prove that this works. Look at $x$ mod $N_i$. All of the terms except the $M_i$ term drop out. The $M_i$ term is

\[ x_i M_i^{-1} M_i \equiv x_i \pmod{N_i} \]

since $M_i^{-1}$ is the inverse of $M_i$ mod $N_i$, we have just $x_i$.

**THERE ARE MORE PAGES!!!!!!!!!!!!!!!**
5. (30 points) (Read the slides on low-exponent attacks on RSA.) Before getting to the specs of the psuedocode you are to write, here is the setting.

- Zelda will do RSA with \( L \) people \( A_1, \ldots, A_L \).
- Zelda is using RSA as follows: For person \( A_i \) she uses \((e, N_i)\).
- The \( N_i \) are all relatively prime.
- \( N_1 < \cdots < N_L \).
- The parameter \( e \) – we think of it as being small but the algorithm should run even if \( e \) is not small. It may report back NO could not crack.
- We assume that Zelda sent the same message to everyone. The message is \( m \). So she send \( A_i \) the number \( m^e \mod N_i \).
- You are Eve. You already have a program that will do the Chinese Remainder Theorem. That is, you have a program that will, on input \( x_1, \ldots, x_L, N_1, \ldots, N_L \) where the \( N_i \)'s are rel prime, output \( x \) such that, for all \( 1 \leq i \leq L, x \equiv x_i \pmod{N_i} \).

NOW YOUR ASSIGNMENT:
Write pseudocode for a program such that

(a) **Input:** \( e, N_1 < \ldots < N_L \) and \( c_1, \ldots, c_L \). The \( N_i \) are rel prime. There is an \( m \) such that, for all \( 1 \leq i \leq L, c_i = m^e \pmod{N_i} \).

(b) **Output:** Either find \( m \) as in the example in class OR say that you can’t find \( m \). Prove that if \( e \leq L \) then your algorithm does find \( m \).

SOLUTION TO PROBLEM FIVE

(a) Input: \( e, N_1, \ldots, N_L \) and \( c_1, \ldots, c_L \). The \( N_i \) are rel prime. There is an \( m \) such that, for all \( 1 \leq i \leq L, c_i = m^e \pmod{N_i} \).

(b) Find (using CRT) \( x \) such that
\[
x \equiv m^e \pmod{N_1}
\]
\[
x \equiv m^e \pmod{N_2}
\]
...
\[ x \equiv m^e \pmod{N_L} \]

AND

\[ 0 \leq x < N_1 \cdots N_L. \]

(NOTE- \( x \) is an \( e \)th power mod \( N_1, N_2, \ldots, N_L \). Hence \( x \) is an \( e \)th power mod \( N_1N_2\cdots N_L \).

(c) Try to take the normal \( e \)th root of \( x \). If you succeed (and get an integer result), that is your \( m \).

By the nature of \( x \)

\[ x \equiv m^e \pmod{N_1 \cdots N_L}. \]

We are curious if the \( m^e \) calculation used wrap-around.

We know that

\[ m < N_1. \]

\[ m^2 < N_1N_2. \]

etc.

\[ m^L < N_1N_2\cdots N_L. \]

If \( e \leq L \) then we have that \( m^e < N_1 \cdots N_L \). Hence the equation did not use wrap around so \( x \equiv m^e \) means \( x = m^e \).