1. Consider the ElGamal cryptosystem with \( b \equiv a^d \pmod{p} \), where

\[
\begin{align*}
  p &= 135322025527948169565801712526686305334500182031517376903466749501913666265021 \\
  a &= 2 \\
  b &= 591459223934937555743855628734052287057862941531907443642747371024854341326 \\
  d &= 6063188541214496266260822945492895830629904429359680619578799138331318053027 \\
\end{align*}
\]

(a) Encode the plaintext \( m \) using the random key \( k \), where

\[
\begin{align*}
  m &= 12019220724030803241420011419070417030402140308130619070818151100081319042319 \\
  k &= 4603799665325142105471559684242857506990992745520804628041593957550121772525 \\
\end{align*}
\]

(b) The ciphertext pair \((r, t)\) with

\[
\begin{align*}
  r &= 127743390919365237786815860974455787013568722004943642833623781336575732260330 \\
  t &= 2006881048960152692934933387815464953301540648076799420273209802552375424096 \\
\end{align*}
\]

corresponds to a plaintext message encoded in the standard way (\(a = 00\), \(b = 01\), \(\ldots\), \(z = 25\)): find the plaintext.

2. Use the baby-step giant-step algorithm to compute the discrete logarithm \( \log_a b \pmod{p} \), where

(a) \( a = 5 \), \( b = 208 \), \( p = 277 \).

(b) \( a = 5 \), \( b = 181 \), \( p = 383 \).

3. Use the Pohlig-Hellman algorithm to compute the discrete logarithm \( \log_a b \pmod{p} \), where \( a = 3 \), \( b = 11059101 \), \( p = 58564001 \).

4. You run a quadratic-sieving algorithm to compute discrete logarithms to the primitive root base \( a = 10 \) modulo

\[
p = 39119362024094229361385833
\]

but your computer runs out of memory just before it finishes the first step. Instead, it leaves you the five congruences

\[
\begin{align*}
  a^1 &\equiv 2 \cdot 5 \\
  a^{583746707701017028596572} &\equiv 7 \cdot 11^2 \\
  a^{2401453523712100525373399} &\equiv 3 \cdot 11^5 \\
  a^{3253693761724484113713347} &\equiv 7 \cdot 11 \\
  a^{4563955537604250268550839} &\equiv 5 \cdot 7^2
\end{align*}
\]

all of which are modulo \( p \).

(a) Find \( \log_a 2 \), \( \log_a 3 \), \( \log_a 5 \), \( \log_a 7 \), and \( \log_a 11 \).

(b) You then run the second part of the quadratic-sieving algorithm to compute the discrete logarithm of \( n = 23883233477604142285940924 \) to the base \( a \) modulo \( p \), but your computer again runs out of memory: it only returns the congruence

\[
n \cdot 10^{34923038423694} \equiv 2^{11} 3^7 7^8 11^{11} \pmod{p}
\]

Find the discrete logarithm \( \log_a(n) \).
5. (Non-Collaboration Problem) Eve reads about the baby-step giant-step algorithm and decides to adapt it to create an attack on RSA: she knows Bob’s public key \((N, e)\) and has a plaintext-ciphertext pair \((m, c)\) and wants to find a decryption exponent \(d\). She chooses an integer \(M\) such that \(M^2 \geq N\) and then computes two lists: the values \(c^x \pmod N\) for all \(0 \leq x \leq M - 1\) and the values \(mc^{-My} \pmod N\) for all \(0 \leq y \leq M - 1\).

(a) Explain why Eve is always guaranteed to find a match between the two lists, and how she can use a match to find an exponent \(d\) such that \(c \equiv m^d \pmod N\).

(b) Is the value \(d\) from part (a) always guaranteed to be an actual decryption exponent for all ciphertexts? [Hint: What if \(m = c = 1\)?]

(c) Explain why this attack will not be a very useful practical attack on RSA. [Hint: How long does it take to compute each list?]

6. For a hash function \(H\) to be cryptographically useful, it needs to be (i) easy to evaluate, (ii) hard to invert, and (iii) hard to find collisions. (We say \(H\) is hard to invert if for any \(y\), it is difficult to find an \(x\) for which \(H(x) = y\), and we say it is hard to find collisions if it is difficult to find \(x_1\) and \(x_2\) for which \(H(x_1) = H(x_2)\).)

(a) Suppose \(N = pq\) is the product of two large primes; recall from our discussion of Rabin encryption that it is hard to compute square roots modulo \(N\). Which of the properties (i)-(iii) are satisfied for the function \(H(x) = x^2 \pmod N\) if \(x\) is restricted to being between \(0\) and \(N - 1\)? What if \(x\) can be arbitrarily large?

(b) Suppose \(p\) is a fixed large prime for which it is hard to compute discrete logarithms and \(a\) is a primitive root modulo \(p\). Which of the properties (i)-(iii) are satisfied for the function \(H(x) = a^x \pmod p\) if \(x\) is restricted to being between \(0\) and \(p - 2\)? What if \(x\) can be arbitrarily large?