Final Exam Written Solutions

Written Part

3. In Problem 2 I found one algorithm worked for (c) while a different one worked for (d). Explain why this could happen.

   Proof. In part (c) the $p-1$ algorithm didn't work, whereas in part (d) it did. This may be initially surprising but came down to the fact that one for one the prime factors p of the number from part (d), $p-1$ has small factors (which doesn't happen for the number in part (c)). On the other hand, Lenstra worked for (c) but not (d). The time complexity of Lenstra depends (loosely) on the size of the number we are factoring, so this isn't particularly surprising. That said, one could get lucky and choose a curve E and a point P such that as an element of $E(\mathbb{F}_p)$ (for the same p as above) P has small order. This means that computing nP would become impossible over $E(\mathbb{Z}/N\mathbb{Z})$ rather soon and we would find p this way. □

4. In class we suggested that the problem of factoring a number N is in some sense equivalent to being able to compute square roots modulo N. In this problem we will make this precise, for $N = pq$ a product of 2 distinct **odd** primes.

   (a) Suppose you know the factorization of N into pq. Describe an algorithm to efficiently compute whether a has a square root modulo N, and prove the correctness of your algorithm.

   Proof. By Homework 8 problem 9(b), a has a square root mod N if and only if it has a square root mod p and a square root mod q. Furthermore, appling Homework 8 Problme 9(a), a has a square root mod p if and only if it is either $0 \mod p$, or else $a^{\frac{p-1}{2}} \equiv 1 \mod p$. This precisely means that the legendre symbol $\left(\frac{a}{p}\right) \neq -1$. Similar considerations hold for q so the following algorithm works:

   (1) Input a, N and the factorization $N = pq$: An algorithm to determine if a has a square root modulon N.
   (2) If $a^{\frac{p-1}{2}} \not\equiv -1 \mod p$ and $a^{\frac{q-1}{2}} \not\equiv -1 \mod q$, return `True`.
   (3) Else return `False`.

   □

   (b) Suppose $\gcd(a, N) = 1$. Show that if a has one square root modulo N, then it exactly 4 square roots modulo N. In the case where $\gcd(a, N) \neq 1$, how many square roots might a have? Why?

   Proof. Recall that the Chinese Remainder Theorem that given 2 congruences $x \equiv b \mod p$ and $x \equiv c \mod q$, there is a unique element of $\mathbb{Z}/N\mathbb{Z}$ solving two congruences. In particular, this means that there is a bijection:

   $$\{d \in \mathbb{Z}/N\mathbb{Z}\} \leftrightarrow \{(b, c) \text{ where } b \in \mathbb{Z}/p\mathbb{Z} \text{ and } d \in \mathbb{Z}/q\mathbb{Z}\}. \tag{1}$$

   Indeed, the map from the left to the right is $d \mapsto (\overline{d}, \overline{d})$ where the coordinates are reductions modulo p and q of d respectively, and the map from right to left is using the chinese remainer to theorem as in the paragraph above.

Notice that d is a square root of a modulo N if and only if it is one modulo p and modulo q. Indeed, the forward direction is immediate, on the other hand if $d^2 \equiv a \mod p$ and $d^2 \equiv a \mod q$, then by the uniqueness part of the Chinese Remainder Theorem, $d \equiv a \mod N$. Therefore, Equation 1 restricts to a bijection:

$$\left\{ \begin{array}{c} \text{Square roots} \\ \text{of } a \text{ modulo } N. \end{array} \right\} \leftrightarrow \left\{ \begin{array}{c} \text{Pairs } (b,c) \text{ where } b \text{ is a square root} \\ \text{of } a \text{ modulo } p \text{ and } c \text{ is one modulo } q. \end{array} \right\} \tag{2}$$

We are trying to compute the size of the left hand side of Equation 2, so this bijection means it suffices to compute the size of the right side. With this in mind, suppose a has a square root mod N. Then a has a square root b modulo p and a square root c modulo q. If a is nonzero mod p it has 2 square roots mod p: b and $p-b$. And similarly if a is nonzero mod q both c and $q-c$ are square roots. Therefore if a is prime to N the set on the right in Equation 2

$$\{(b,c),(p-b,c),(b,q-c),(p-b,q-c)\}.$$

Therefore a has four square roots mod N. We briefly describe the other possibilities. Assume that we still have have a square root of a modulo N, and let $g = \gcd(a,N)$.

g=N: In this case $a \equiv 0 \mod N$ so the only square root of a is 0 itself. This corresponds to the pair $(0,0)$ on the righthandside of equation 2.

g=p: In this case $a \equiv 0 \mod p$, so it only has one square root modulo p. Therefore the righthandside of Equation 2 consists of the set $\{(0,c),(0,p-c)\}$, so that a has 2 square roots.

g=q: This case is symmetric to $g = p$, and a has 2 square roots.

We observe then that the only possibilities for the number of square roots of a are $1, 2, 4$ (if any), and that this number is controlled exactly by the value of g. Notice that a can never have exactly 3 square roots mod N. Also compare to the case where $p = q$ studied in Homework 3 Problem 7, where the only possibilities were 1 and 2. □

(c) Suppose you know the factorization of N into pq. Describe an algorithm to compute all the square roots of a modulo N if they exist. Prove the correctness of your algoritm. (You may assume you have a fast algorithm to compute square roots modulo primes.)

Proof. By Equation 2 above, all the square roots of a modulo N come from computing the square roots of a modulo p and modulo q, and *stitching them together* using the chinese remainder theorem. Therefore the following algorithm works:

(1) Input a, N and the factorization $N = pq$: An algorithm to compute all the square roots of a modulo N.
(2) Compute all the square roots b_i of a modulo p.
(3) Compute all the square roots c_j of a modulo q.
(4) Use the Chinese Remainder Theorem algorithm from Homework 5 Problem 1 to compute $d_{ij} =$ `CRT(p,q,bi,cj)`, the unique element such that $d_{ij} \equiv b_i \mod p$ and $d_{ij} \equiv c_j \mod q$.
(5) Return the complete list of d_{ij}. These are the square roots of a.

□

(d) Conversely, suppose you have an oracle that can tell you all the square roots of a modulo N if they exist. Describe a way to use constultation with this oracle to factor N. Prove your method works.

Proof. Pick some nonzero $a \in \mathbb{Z}/N\mathbb{Z}$ and suppose $\gcd(a, N) = 1$, and suppose we can compute all the square roots (r_1, r_2, r_3, r_4) of a modulo N. Then we can choose a pair of square roots such that $r_i \neq \pm r_j \mod N$. Then both $r_i - r_j, r_i + r_j$ are nonzero modulo N. But:

$$(r_i - r_j)(r_i + r_j) = r_i^2 - r_j^2 \equiv a - a \equiv 0 \mod N.$$

So $(r_i - r_j)(r_i + r_k) = kN$ for a positive number k, and it looks like we've come pretty close to finding a factorization of N. Indeed, let $g = \gcd(N, r_i - r_j)$. Then g is one of $1, p, q, N$. We can eliminate N because we already saw that $r_i - r_j$ is nonzero modulo N. On the other hand if $g = 1$, then both p and q divide $r_i + r_j$, so that $pq = N$ does. But we also saw that $r_i + r_j$ is nonzero modulo N. Therefore $g = p$ or $g = q$, and we have found a nontrivial factor of N.

This only takes care of the case where $\gcd(a, N) = 1$, but of course, otherwise it would be other p or q and we would already have our factorization. We have therefore showed that the following algorithm works:

(1) Given N and a way to compute square roots, factor N.
(2) Choose nonzero $a \in \mathbb{Z}/N\mathbb{Z}$ and compute $g = \gcd(a, N)$.
(3) If $g \neq 1$ then return the factors $(g, N/g)$.
(4) Otherwise compute the four square roots r_1, r_2, r_3, r_4 of a modulo N.
(5) Find r_i, r_j such that $r_i \neq \pm r_j$ modulo N. Then compute $g = \gcd(r_i - r_j, N)$, and return the factors $(g, N/g)$.

□

5. Let E be an elliptic curve over $\mathbb{F}_p$, and suppose that $\#E(\mathbb{F}_p) = n$. Let $P \in E(\mathbb{F}_p)$ be a point. Suppose a is an integer and that $\gcd(a, n) = 1$. Prove that P is a multiple of aP.

Proof. By Lagranges theorem, we know that $nP = \mathcal{O}$. By the extended Euclidean algorithm, there exists some u, v such that $au = 1 + vn$. Therefore we may compute:

$$u(aP) = (1 + vn)P = 1P + v(nP) = P + v\mathcal{O} = P,$$

and so P is a multiple of aP. □

6. Recall that the Elliptic Curve Diffie Hellman Problem (ECDHP) is the problem of recovring a point $nmQ \in E(\mathbb{F}_p)$, given nQ and mQ.

(a) Formulate the MV-Elgamal Problem. That is, describe the precise mathematical problem at the center of MV-Elgamal.

Proof. Fix an elliptic curve and a point $P \in E(\mathbb{F}_p)$. Given the points $nP, kP \in E(\mathbb{F}_p)$ as well as the numbers $x_{nkP}m_1, y_{nkP}m_2 \in \mathbb{F}_p^*$, recover m_1 and m_2. (where $(x_{nkP}, y_{nkP}) = nkP$). □

(b) Prove that the security MV-Elgamal Problem is equivalent to the security of the ECDHP. (Use oracles!).

Proof. First suppose you have access to a ECDHP oracle. You present the oracle nP and kP and are told the value of nkP, from which you extract the coordinates x_{nkP} and y_{nkP}. You invert these to compute

$$x_{nkP}^{-1}(x_{nkP}m_1) \equiv m_1 \mod p \quad \text{and} \quad y_{nkP}^{-1}(y_{nkP}m_2) \equiv m_2 \mod p,$$

thereby solving the MV-Elgamal problem. Conversely, suppose you are given nQ and mQ and access to an MV-Elgamal oracle. You present your MV-Elgamal oracle with nQ, mQ and any pair $c_1, c_2 \in \mathbb{F}_p^*$. The oracle presents you with $x_{nmQ}^{-1}c_1$ and $y_{nmQ}^{-1}c_2$, whereby multiplying by c_1^{-1} (respectively c_2^{-1}) and inverting you deduce $(x_{nmQ}, y_{nmQ}) = nmQ$ which solves the ECDHP. □

7. Recall that in HW4 Problem 3 we studied a public key cryptosystem that involving Alice and Bob agreeing on a public prime and exchanging a series of values in $\mathbb{F}_p^*$.

(a) Formulate a version of this cryptosystem for Elliptic curves, and prove its correctness. You may assume that you have a fast way to compute $\#E(\mathbb{F}_p)$. Be sure to make clear what is public information and what is private information. What plays the roll of the message?

Proof. We will formulate it in a table. Public information will be highlighted in purple. To begin, Even and Bob agree on the following public parameters: a prime number p, and an elliptic curve E over $\mathbb{F}_p$. The size $n = \#E(\mathbb{F}_p)$ is easily computed and can also therefore be assumed to be public. The message will be a point in this elliptic curve.

	Alice	Eve	Bob
1.	Alice has message $M \in E(\mathbb{F}_p)$		
2.	Alice chooses a with $\gcd(a, n) = 1$		
3.	Alice computes $P = aM \in E(\mathbb{F}_p)$	$\longrightarrow$	Bob recieves P
4.			Bob chooses b with $\gcd(b, n) = 1$.
5.	Alice recieves Q	$\longleftarrow$	Bob Computes $Q = bP \in E(\mathbb{F}_p)$
6.	Alice computes $a' = a^{-1} \mod n$		
7.	Alice computes $R = a'Q \in E(\mathbb{F}_p)$	$\longrightarrow$	Bob recieves R.
8.			Bob computes $b' = b^{-1} \mod n$
9.			Bob computes $S = b'R \in E(\mathbb{F}_p)$. Then $S = M$!

To prove correctness we must show that $S = M$. We first fix some notation $aa' = 1 + kn$ and $bb' = 1 + ln$. Then

$$aa'bb' = 1 + kln + (l^2 + k^2)n^2 = 1 + \lambda n$$

for some integer $\lambda = kl + (l^2 + k^2)n$. The we remind the reader that (as in problem 5), by Lagrange's theorem, $n \cdot T = \mathcal{O}$ for any $T \in E(\mathbb{F}_p)$. Then we can directly compute:

$$S = b'a'baM = (1 + \lambda n)M = M + \lambda nM,$$

as desired. □

(b) Show that a solution to the Elliptic Curve Discrete Log Problem will break this cryptosystem.

Proof. Eve intercepts P and $Q = bP$, and consults with the ECDLP oracle to recover b. She can then compute $b' = b^{-1} \mod n$ (as n is public). She then intercepts R and can compute $S = b'R$. But this is M, so Eve has discovered the message. □

(c) Show that a solution the the Elliptic Curve Diffie Hellman Problem will break this cryptosystem.

Proof. Eve intercepts $P = aM$, $Q = abM$, and $R = aba'M$. She notices that $R = a'Q$ and $P = b'Q$. To see the latter, we directly compute

$$b'Q = abb'M = a(1 + ln)M = a(M + lnM) = a(M + \mathcal{O}) = aM = P.$$

Therefore she can consult an ECDHP oracle, feeding them, $Q, a'Q$, and $b'Q$ (which we have seen are all public), and obtain $a'b'Q$. But this is $a'b'abQ = S = M$. □

(d) Discuss any other advantages or disadvantages you observe about this cryptosystem.

Proof. This is rather open ended. One thing I notice, is the fact that they have to communicate 3 times to send a message, and that Bob must be available to respond immediately to receive a message. This also allows extra opportunity for security breaches, for example, Eve could pose as Bob since the public key isn't published, and relies on Alices message to be available.

Also there is the problem of turning a plaintext (in plain language) into a point on an elliptic curve (although we study this in problem 8 from a probabilistic standpoint).

Then ther is message expansion. Suppose p is k-bits. Since $\#E(\mathbb{F}_p) \approx p$, one can only store approximately k bits of data in the curve. But we have to send 2 coordinates in $\mathbb{F}_p$ back and forth 3 times, so this is a 6:1 message expansion. This could be compressed to 3:1 using the technique of TH2 Problem 8(b), but is still greater than many other systems.

This has an important advantage over the analogous problem over $\mathbb{F}_p^*$ described in HW4. In particular, the ECDLP is more secure than the DLP. □

In the previous problem, the *message* which was sent was a chosen point on an elliptic curve. When the message is a number we can use ASCII to directly translate text to integers, but with points on an elliptic curve, the question is more subtle. MV-Elgamal got around this subtlety just using the elliptic curve to create a shared secret, and using the coordinates as keys for a symmetric cipher whose plaintext are integers (not points). In the following exercise we will describe another solution, using a probabilistic algorithm to map integers to points on the elliptic curve in a recoverable way.

8. Let E be an elliptic curve over $\mathbb{F}_p$ with equation $y^2 = x^3 + Ax + B$, where p is $2k + 1$-bits in length.

(a) The most naïve way to map a plaintext m (which can be thought of as an integer) to a point on the elliptic curve would to embed m as the x-coordinate. Explain what goes wrong with this approach.

Proof. We denote the right hand side of the equation of the curve by the function $f(x) = x^3 + Ax + B$. Give some value m, it isn't always true that $f(m) = m^3 + Am + B$ is a square. If this happens, m couldn't serve as an x-coordinate of any point in $E(\mathbb{F}_p)$. □

(b) Instead we will implement the following probabilistic algorithm, which will allow us to map a message k-bits in length to a point on the elliptic curve, in the following steps.

(1) Start with a plaintext message m, stored as an integer k-bits in length.
(2) Choose a random integer r, also k-bits in length.
(3) Compute $r||m \in \mathbb{F}_p$.
(4) Detect if $r||m$ is the x coordinate of a point in $E(\mathbb{F}_p)$. If so, compute the y coordinate, and return $P = (r||m, y)$ as your plaintext for the elliptic curve encryption. Otherwise return to step 2.

Describe exactly how Step 4 would be carried out. (You may assume that you have a fast algorithm to compute square roots modulo p if they exist). Conversely explain how to reverse the algorithm to recover the plaintext from a point.

Proof. Let $z = r||m \in \mathbb{F}_p$, and consider $f(z) = z^3 + Az + B$. We first compute $\left(\frac{f(z)}{p}\right) = f(z)^{\frac{p-1}{2}} \mod p$. If this is 1, then we compute $y = \sqrt{f(z)} \mod p$, and our message will be the point $P = (z, y)$. (Notice if $(fracf(z)p) = 0$ then we could compute a square root, but it would be 0. This would give us an order 2 point on the curve and not be very secure for communication. In general, we could check that P has relatively high order before continuing.)

To reverse this process, given a point $P = (x, y)$ we can recover the message m as the last k-bits of x. □

(c) How many values of r would expect to have to try until you have a point? Justify your answer. (Hint, this is essentially a collision algorithm, so what is the length of the list you are trying to find a collision with? You may assume $r||m$ varies 'randomly enough' as r varies.)

Proof. Let Ω be the set of choices of random values in $\mathbb{F}_p$ Define a random variable $X(\omega) = n$ if the first valid x-coordinate chosen in the occurance ω is chosen on the n'th step. We are trying to compute $E(f)$. Assume that a random choice of $z \in \mathbb{F}_p$ value has a probability ρ of being a valid x-coordinate. Then we showed that $E(X) = 1/\rho$ (for reference, November 3rd lecture on geometric distribution and November 5th lecture on expected values, or Examples 5.31 and 5.37 in [HPS]). Since we are assuming the $r||m$ behave like random choices in $\mathbb{F}_p$, it suffices to compute (or approximate) ρ. There are two ways to proceed, each with their own set of approximations. Comfortingly, both end up giving the same result.

Option 1: Let $L = \{t_1, \cdots, t_n\}$ bet the list of valid x-coordinates. We'd like to compute $\#L$, and the $\rho = \#L/p$. Notice that each valid x-coordinates give us 2 points in E (given by $\pm f(x)$), unless $f(x)$ is 0 when we get 1. This latter case rarely happens. We we can

assume there are approximately half as many x-coordinates as points in $E(\mathbb{F}_p) \setminus \mathcal{O}$ (since $\mathcal{O}$ has no x-coordinate). Since $E(\mathbb{F}_p) \approx p+1$ we have:

$$\#\{\text{valid } x\text{-coordinates}\} \approx \frac{\#E(\mathbb{F}_p) - 1}{2} \approx \frac{p}{2}.$$

Therefore $\rho \approx 1/2$, and $E(X) \approx 2$.

Option 2: Suppose f behaves sufficiently randomly, so that $f(z)$ is random if z is. Then the probability that a random z is a valid x-coordinate is the same as the probability that a random element of $\mathbb{F}_p$ is perfect square−or a quadratic residue if we'd like to eliminate cases where the y-coordinate is 0. By Homework 2 Problem 8(d) there are $(p-1)/2$ perfect square in $\mathbb{F}_p^*$, so that the number of valid x-coordinates is approximately $\frac{p-1}{2}$ (or respectively $\frac{p+1}{2}$). For large p this is approximately $p/2$. Therefore $\rho \approx 1/2$ and $E(X) \approx 2$. □

(d) Explicitly describe the steps you need to add to the algorithm from Problem 7 in order to be able to communicate in plain language. (Including applying things like `intToText`).

Proof. Call the algorithm described in part (b) `mapToPoint(m)`.

(1) Alice takes her `message` and computes `m = textToInt(message)`. She chooses a prime p with over twice as many bits as m and an elliptic curve $E/\mathbb{F}_p$, publishing both.
(2) Alice compute `P = mapToPoint(m)`. It terminates in approximately 2 steps.
(3) Alice initiates the message exchange described in problem 7.
(4) It teminates, and Bob has $P = (x, y) \in E(\mathbb{F}_p)$. Bob recoveres m as the last k bits of x.
(5) Bob recovers `message = intToText(m)`.

□

9. You are Eve, and you intercept Bob's cipher text $c = [R, c_1, c_2]$ sent to Alice. You know it was encrypted using MV-Elgamal, with a public curve E over a prime p with a base point P, and you know Alice's public key Q. You also have 1 time access to Alice's machine to decrypt MV-Elgamal ciphers. Describe a way to obtain the plaintext message Bob sent to Alice, proving the correctness of your method. **Note:** Alice's machine runs a security protocol which will not decrypt messages that have already been decrypted, so you can not just ask it to decrypt c.

Proof. Here are a couple of ways to get around Alice's basic security protocol. First one could present their decryption protocol with $[-R, c_1, c_2]$. Then it would compute $-T = (x_T, -y_T)$, and return $m_1 = x_T^{-1} c_1$ and $-m_2 = -x_T^{-1} c_2$.

Slightly less silly would be presenting the decryption protocol with $[R, kc_1, lc_2]$ for $k, l \in \mathbb{F}_p^*$. Then the decryption protocol would return km_1 and lm_2 from which m_1 and m_2 would easily be recovered. □

Bonus: Alice noticed a weakness in her machine's security protocol, and updates it to the following: If Alice's machine has already decrypted $c = [R, c_1, c_2]$, it will only decrypt ciphertexts $c' = [R', c_1', c_2']$ where $c_1' \neq c_1$, $c_2' \neq c_2$, and $R' \neq \pm R$. But you're clever: devise a 'chosen ciphertext attack' to get around this updated security protocol.

Proof. To get around these security protocols and give Alice's decryption machine something it can't distinguish as related to Bob's message, we choose some k which is coprime to $\#E(\mathbb{F}_p)$, and compute $R' = kR$. We also choose random $c_1', c_2' \in \mathbb{F}_p^*$. We then present the decryption protocol with $[R', c_1', c_2']$. It computes $T' = n_A R'$, and presents us with $(m_1', m_2') = (x_{T'}^{-1} c_1', y_{T'}^{-1} c_2')$. Since we know c_1' and c_2' we can therefore recover $T' = (x_{T'}, y_{T'})$. But then:

$$T' = n_A R' = n_A kR = k(n_A R) = kT.$$

Therefore if we compute $k^{-1} \mod \#E(\mathbb{F}_p)$ we have (as in problem 5 and 7) $k^{-1}T' = k^{-1}kT = T$. Therefore we have obtained the encryption keys (x_T, y_T), and can invert them and decrypt $(m_1, m_2) = (x_T^{-1} c_1, y_T^{-1} c_2)$. □