Amherst College
Department of Mathematics and Statistics

MATH 252 | FINAL EXAM | SPRING 2019

NAME: Solutions

Read This First!

- Keep cell phones off and out of sight.
- Do not talk during the exam.
- You are allowed one page of notes, front and back.
- You may use a calculator, but **you are expected to use only the four arithmetic functions**, in order to be fair to students with a four-function calculator. Clearly write the calculations you have done on the page.
- You may use any of the blank pages to continue answers if you run out of space. Please clearly indicate on the problem's original page if you do so, so that I know to look for it.
- In order to receive full credit on a problem, solution methods must be complete, logical and understandable.

Grading - For Instructor Use Only

Question:	1	2	3	4	5	6	7	8	Total
Points:	9	7	7	8	8	7	7	7	60
Score:									

This page intentionally left blank. You may use it for scratchwork or to continue answers to any question (note clearly on the original page if you do so).

1. [9 points] Samantha has published the following RSA public key: her modulus is $N = 299$ and her verification key is $e = 5$ (see the summary table at the back of the exam packet for notation). Victor receives the following three documents and signatures. Determine which signatures are valid, and which are invalid.

   (a) Document $D = 90$, signature $S = 155$.

$$S^2 = 155^2 = 24025 \equiv 24025 - 80 \cdot 299$$
$$\equiv 105 \bmod 299$$
$$S^4 \equiv 105^2 = 11025 \equiv 11025 - 36 \cdot 299$$
$$\equiv 261 \bmod 299$$
$$S^5 = S^4 \cdot S \equiv 261 \cdot 155 = 40455 \equiv 40455 - 135 \cdot 299$$
$$\equiv 90 \bmod 299$$

$\Rightarrow S^5 \equiv D \bmod N$, so the signature is <u>valid</u>.

   (b) Document $D = 153$, signature $S = 50$.

$$S^2 = 50^2 = 2500 \equiv 2500 - 8 \cdot 299$$
$$\equiv 108 \bmod 299$$
$$S^4 \equiv 108^2 = 11664 \equiv 11664 - 39 \cdot 299$$
$$\equiv 3 \bmod N$$
$$S^5 \equiv 3 \cdot 50 \equiv 150 \bmod 299$$

$\Rightarrow S^5 \not\equiv D \bmod N$, so the signature is <u>invalid</u>.

   (c) Document $D = 238$, signature $S = 101$.

$$S^2 = 101^2 = 10201 \equiv 10201 - 34 \cdot 299$$
$$\equiv 35 \bmod 299$$
$$S^4 \equiv 35^2 = 1225 \equiv 1225 - 4 \cdot 299$$
$$\equiv 29 \bmod N$$
$$S^5 \equiv 29 \cdot 101 = 2929 \equiv 2929 - 9 \cdot 299$$
$$\equiv 238 \bmod 299$$

$\Rightarrow S^5 \equiv D \bmod N$, so the signature is <u>valid</u>.

This page intentionally left blank. You may use it for scratchwork or to continue answers to any question (note clearly on the original page if you do so).

2. [7 points] Alice is implementing some code to perform elliptic curve Diffie-Hellman key exchange (see the summary table at the back of the exam packet). So far, she has written a working implementation of a function ecAdd(P, Q, A, B, p), which accepts two points P, Q on an elliptic curve over $\mathbb{F}_p$ defined by the congruence $Y^2 \equiv X^3 + AX + B \pmod{p}$, and returns $P \oplus Q$.

Write a function ecdh(P, QB, A, B, p) that takes the public parameters and Bob's point Q_B, and returns both the point Q_A that Alice should send to Bob and the shared secret S. You should fully implement any helper function you need, except functions that are built-in to Python and the ecAdd function. For full points, your function should only need to call ecAdd $\mathcal{O}(\log p)$ times (you do not need to prove that this is true, however). A less efficient implementation will receive partial credit.

```
#helper function: fast-multiplication ("double & add")
def ecMult(n, P, A,B, p):
    Q = O                    # Note: assumes n≥0.
    while n>0:
        if n%2 == 1:
            Q = ecAdd(Q,P, A,B,p)
        n //= 2
        P = ecAdd(P,P, A,B,p)
    return Q


#main Diffie-Hellman code
import random
def ecdh(P, QB, A, B, p):
    nA = random.randint(2, p)    # since order of P is ≈ p, this
                                 # upper bound is a decent choice.
    QA = ecMult(nA, P, A,B,p)
    S = ecMult(nA, QB, A,B,p)
    return QA, S
```

This page intentionally left blank. You may use it for scratchwork or to continue answers to any question (note clearly on the original page if you do so).

3. [7 points] Suppose that Alice and Bob perform Diffie-Hellman key exchange two days in a row. The public parameters p, g are the same on both days (see the summary table at the back of the packet for notation). On the first day, Alice and Bob exchange numbers A and B to establish a shared secret S. On the second day, Alice and Bob exchange numbers A' and B' and establish shared secret S'.

Eve intercepts the numbers A, B, A', and B', as usual. She notices that Alice and Bob are not generating their random numbers very well, and the following simple relationships hold between A and A', and between B and B'.

$$\begin{aligned} A' &\equiv A^2 \pmod{p} \\ B' &\equiv g^7 B \pmod{p} \end{aligned}$$

Show that if Eve manages to learn the first shared secret S, then she can quickly compute the second shared secret S' as well. Describe as specifically as possible how she could compute it from the information she knows.

Observe that if the secret numbers are a, b, a', b', then:

$$\begin{aligned} S' &\equiv g^{a'b'} \equiv (g^{a'})^{b'} \equiv (A')^{b'} \mod p \\ &\equiv (A^2)^{b'} \equiv (g^{2a})^{b'} \equiv (g^{b'})^{2a} \equiv (B')^{2a} \mod p \\ &\equiv (g^7 B)^{2a} \equiv g^{14a} \cdot B^{2a} \equiv (g^a)^{14} (B^a)^2 \mod p \\ &\equiv \underline{A^{14} \cdot S^2 \mod p} \end{aligned}$$

(another point of view: $a' \equiv 2a \mod (\text{order of } g)$ & $b' \equiv b+7 \mod (\text{order of } g)$,
so $g^{a'b'} \equiv g^{2a(b+7)} \equiv g^{2ab+14a} \equiv (g^{ab})^2 \cdot (g^a)^{14} \equiv S^2 \cdot A^{14} \mod p$)

So Eve could, if she learns S, compute $A^{14} S^2 \mod p$ to also learn S'.

This page intentionally left blank. You may use it for scratchwork or to continue answers to any question (note clearly on the original page if you do so).

4. [8 points] Suppose that p, q are two distinct primes, and $N = pq$. Suppose that a is an integer such that $a \equiv 1 \pmod{p}$.

(a) Prove that if $a \equiv 1 \pmod{q}$ as well, then in fact $a \equiv 1 \pmod{N}$.

$a \equiv 1 \bmod q \Rightarrow \exists k \in \mathbb{Z}$ st. $a - 1 = kq$.

Since $a \equiv 1 \bmod p$, $p \mid kq$.

Since $p \neq q$ & q is prime, ~~$p \nmid q$~~ $\gcd(p,q) = 1$,

so $p \mid k$ by Euclids' lemma. Hence $\exists l \in \mathbb{Z}$

st. $k = lp$.

Thus $a - 1 = lpq$, ie. $pq \mid (a-1)$,

ie. $a \equiv 1 \bmod pq$, as desired.

(b) Prove conversely that if $a \equiv 1 \pmod{N}$, then $a \equiv 1 \pmod{q}$.

$a - 1 \equiv 0 \bmod N \Rightarrow N \mid (a-1)$

$\Rightarrow pq \mid (a-1)$

$\Rightarrow q \mid (a-1)$

$\Rightarrow a \equiv 1 \bmod q$, as desired.

This page intentionally left blank. You may use it for scratchwork or to continue answers to any question (note clearly on the original page if you do so).

5. [8 points] Define $p = 1213, q = 1129$, and $N = pq$. Both p and q are primes (you don't need to prove this), and $p-1, q-1$ have the following prime factorizations.

$$\begin{aligned} p-1 &= 2^2 \cdot 3 \cdot 101 \\ q-1 &= 2^3 \cdot 3 \cdot 47 \end{aligned}$$

Suppose that a is an integer that is a primitive root modulo p and also a primitive root modulo q.

(a) Determine the minimum positive integer n such that

$$\gcd(a^{n!} - 1, N) = p,$$

or prove that no such integer exists.

No such n exists.

Proof: Suppose for contradiction that $\gcd(a^{n!}-1, N) = p$. Then $p \mid (a^{n!}-1)$, ie. $a^{n!} \equiv 1 \bmod p$. Since a is a prim. root, this means $(p-1) \mid n!$, so $101 \mid n!$. Since 101 is prime, Euclids lemma implies that 101 divides one of $1, 2, \cdots, n$. hence $n \geq 101$. But this means that all of $8, 3, 47$ are among the numbers $1, 2, \cdots, n$. so $(q-1) \mid n!$ as well. & $a^{n!} \equiv 1 \bmod q$. By problem 4, $a^{n!} \equiv 1 \bmod N$, so $N \mid (a^{n!}-1)$, hence $\gcd(a^{n!}-1, N) = N$, not p. This is a contradiction.

(b) Determine the minimum positive integer n such that

$$\gcd(a^{n!} - 1, N) = q,$$

or prove that no such integer exists.

$n = 47$ is the smallest such n. This is because

1) no smaller n works, since $n < 47 \Rightarrow q-1 \nmid n! \Rightarrow a^{n!} \not\equiv 1 \bmod q$ (see the reasoning in (a));

2) $a^{47!} \equiv 1 \bmod q$ since $q-1 \mid 47!$ (all of $8, 3, 47$ occur in $1 \cdot 2 \cdot 3 \cdots 47$)

3) $a^{47!} \not\equiv 1 \bmod p$, as explained in (a).

So $a^{47!}-1$ is divis. by q but not p, & thus $\gcd(a^{47!}-1, N) = q$.

This page intentionally left blank. You may use it for scratchwork or to continue answers to any question (note clearly on the original page if you do so).

6. [7 points] Alice and Bob are using the NTRU cryptosystem, with the following public parameters.

$$N = 7 \qquad p = 3 \qquad q = 41 \qquad d = 2$$

Alice's private information and public key are as follows.

$$\begin{aligned}
\mathbf{f} &= 1 + X + X^3 - X^4 - X^6 \\
\mathbf{g} &= 1 - X + X^2 - X^6 \\
\mathbf{F}_q &= -3 + 12X + 19X^2 - 5X^3 - 2X^4 + 8X^5 + 13X^6 \\
\mathbf{F}_p &= X^2 + X^3 - X^4 \\
\mathbf{h} &= -20 + 9X + 9X^2 - 10X^3 + 14X^4 - 8X^5 + 6X^6
\end{aligned}$$

Bob wishes to send Alice a plaintext $\mathbf{m}$, which he encrypts to the following ciphertext.

$$\mathbf{e} = 20 - 5X + 9X^3 + 11X^4 - 2X^5 + 12X^6$$

Alice begins the decryption process by computing the following convolution product.

$$\mathbf{f} \star \mathbf{e} = 39 + 2X + 6X^3 + 38X^4 + 2X^5 + 40X^6$$

Complete the decryption process and determine the plaintext $\mathbf{m}$. Express your answer as a polynomial that has been centerlifted modulo $p = 3$.

$a = cl_q(f \star e) = -2 + 2X + 6X^3 - 3X^4 + 2X^5 - X^6 \quad (q = 41)$

$\Rightarrow \quad a \equiv 1 - X - X^5 - X^6 \mod p \quad (p = 3)$

$\Rightarrow \quad F_p \star a \equiv (X^2 + X^3 - X^4) \star (1 - X - X^5 - X^6) \mod 3$

$\equiv (X^2 - X^3 - X^0 - X^1)$

$+ (X^3 - X^4 - X^1 - X^2)$

$- (X^4 - X^5 - X^2 - X^3) \quad \mod 3$

$\equiv -1 - 2X + X^2 + X^3 - 2X^4 + X^5 \mod 3$

$\equiv -1 + X + X^2 + X^3 + X^4 + X^5 \mod 3$ (centerlifted)

$\Rightarrow \quad \boxed{m = -1 + X + X^2 + X^3 + X^4 + X^5}$

This page intentionally left blank. You may use it for scratchwork or to continue answers to any question (note clearly on the original page if you do so).

7. [7 points] Samantha is using DSA signatures, with public parameters p, q, g and public verification key A (see the summary table at the back of the exam packet for notation). She publishes two documents D and D' with valid DSA signature (S_1, S_2) and (S_1', S_2') (respectively). Unfortunately, she has made a mistake, and used the same ephemeral key k for both signatures.

   (a) How might Eve notice that Samantha has used the same ephemeral key twice, given the published information?

She could observe that $\boxed{S_1 = S_1'}$.

Strictly speaking, this implies only that $g^k \% p \% q = g^{k'} \% p \% q$.

But this coincidence is extremely unlikely unless

$$g^k \equiv g^{k'} \bmod p \text{ (there are only } q \text{ powers of } g \bmod p\text{)},$$

which is equivalent to $k \equiv k' \bmod q$.

   (b) Write a function `stealKey` that Eve could use to compute Samantha's secret signing key a from the published information. You may assume that Eve has already implemented a function `modInv` to compute modular inverses. You may also make the following assumptions: $S_2 \not\equiv S_2' \pmod q$ and $S_1 \not\equiv 0 \pmod q$.

Eve knows that

$$S_2 \equiv k^{-1}(D + aS_1) \bmod q \qquad (1)$$
$$\& \quad S_2' \equiv k^{-1}(D' + aS_1) \bmod q \qquad (2)$$

or, equivalently,

$$kS_2 \equiv D + aS_1 \bmod q \qquad (3)$$
$$\& \quad kS_2' \equiv D' + aS_1 \bmod q. \qquad (4)$$

One way to solve this is to first solve for k: (in 1st congruence)

$$k \equiv S_2^{-1}D + aS_2^{-1}S_1 \bmod q$$

then substitute this into the second:

$$(S_2^{-1}D + aS_2^{-1}S_1)S_2' \equiv D' + aS_1 \bmod q \qquad (5)$$

$$\Rightarrow \quad (D + aS_1)S_2' \equiv (D' + aS_1)S_2 \bmod q \qquad (6)$$

[Technical point: we've technically assumed $S_2 \not\equiv 0 \bmod q$ in this logic, which might not be so. We can avoid this in hindsight by subtracting $S_2' \cdot$(line 3) from $S_2 \cdot$(line 4) to obtain line (6) directly.]

This page intentionally left blank. You may use it for scratchwork or to continue answers to any question (note clearly on the original page if you do so).

Doing some algebra:

$$a[S_1 S_2' - S_1 S_2] \equiv D' S_2 - D S_2' \mod q$$

$$a \cdot S_1 (S_2' - S_2) \equiv D' S_2 - D S_2' \mod q$$

$$\Rightarrow \quad \underline{a \equiv S_1^{-1} (S_2' - S_2)^{-1} (D' S_2 - D S_2') \mod q}$$

(since we assumed $S_1 \not\equiv 0 \mod q$ & $S_2 \not\equiv S_2' \mod q$, & q is prime, both these inverses exist).

Eve can use this congruence to steal the key.
(I write S_2, S_2', etc. for readability; in python you could write S21, S22, etc. instead)

```
def stealKey(S1, S2, D, S1', S2', D', q):
    factor1 = modInv(S1, q)
    factor2 = modInv(S2'-S2, q)
    factor3 = (D' * S2 - D * S2') % q
    return factor1 * factor2 * factor3 % q
```

Alternate sol'n:
(solve for k first)
Subtract cong. (4) from (3):

$$k(S_2 - S_2') \equiv D - D' \mod q$$

$$\Rightarrow \quad k \equiv (D - D')(S_2 - S_2')^{-1} \mod q$$

Then:

$$a \equiv (k S_2 - D) \cdot S_1^{-1} \mod q$$

```
def stealKey(...):
    k = (D-D') * modInv(S2-S2', q)
    a = (k*S2-D) * modInv(S1, q) % q
    return a
```

8. [7 points] Alice and Bob are using a cryptosystem similar to NTRU, described as follows.

Parameters: $N = 107$, $p = 3$, $q = 331$, $d = 20$. (Note in particular that the inequality $q > (6d+1)p$ from NTRU does not hold, so you should not assume it in your argument).

Key creation: Alice chooses two private elements $\mathbf{f}, \mathbf{g} \in \mathcal{T}(d+1, d)$. You may assume that *both* are invertible in both R_p and R_q. Alice computes the inverse $\mathbf{F}_q$ in R_q, and publishes a public key $\mathbf{h} \equiv \mathbf{F}_q \star \mathbf{g} \pmod q$.

Encryption: Bob's plaintext is a *ternary* polynomial $\mathbf{m} \in R$. Bob chooses a random (ephemeral) polynomial $\mathbf{r}$ that is also ternary (but not necessarily having any specific number of +1's and −1's), and uses Alice's public key to compute a ciphertext $\mathbf{e} \equiv \mathbf{h} \star \mathbf{m} + p\mathbf{r} \pmod q$.

(Recall that a ternary polynomial is a polynomial with all coefficients −1, 0, or 1; equivalently, a polynomial with $|\mathbf{m}|_\infty \le 1$).

In this problem, you will work out a decryption procedure for this system.

(a) In decryption, Alice begins by computing $\mathbf{f} \star \mathbf{e}$ and centerlifiting it $\pmod q$ to a polynomial $\mathbf{a}$. In other words (using our notation from class), $\mathbf{a} = \mathrm{cl}_q(\mathbf{f} \star \mathbf{e})$. Prove that $\mathbf{a}$ is *exactly equal* (not just congruent!) to $\mathbf{g} \star \mathbf{m} + p\, \mathbf{f} \star \mathbf{r}$.
Be sure to refer to the specific parameter values stated above. You should carefully state any lemmas from class that you use in your proof, but you do not need to prove them from scratch.

First observe that it's congruent mod q, since

$$f \star e \equiv f \star h \star m + p f \star r \mod q$$
$$\equiv f \star F_q \star g \star m + p f \star r \mod q$$
$$\equiv 1 \star g \star m + p f \star r \mod q$$
$$(\text{since } f \star F_q \equiv 1 \mod q)$$
$$\equiv g \star m + p f \star r \mod q.$$

Now, recall two facts proved in class:

(1) $\forall a, b \in R$, $|a+b|_\infty \le |a|_\infty + |b|_\infty$

(2) If $a \in \mathcal{T}(d,e)$ & $b \in R$, then $|a \star b|_\infty \le (d+e) \cdot |b|_\infty$.

~~Thus~~

(continued on reverse)

Additional space for part (a).

Therefore:

- $|g \star m + p\, f \star r|_\infty \leq |g \star m|_\infty + |p\, f \star r|_\infty$, (fact (1))

$|g \star m|_\infty \leq (d+d+1) \cdot |m|_\infty \leq d+d+1 = 2d+1$,
(fact (2), w/ $g \in \mathcal{T}(d, d+1)$ & $|m|_\infty \leq 1$)

& $|f \star r|_\infty \leq (d+d+1) \cdot |r|_\infty \leq 2d+1$
(fact (2) again, w/ $f \in \mathcal{T}(d, d+1)$ & $|r|_\infty \leq 1$)

$$\Rightarrow |g \star m + p\, f \star r| \leq (2d+1) + p(2d+1)$$
$$= (p+1)(2d+1)$$
$$= (3+1) \cdot (2 \cdot 20 + 1) = 4 \cdot 41 = \underline{164}.$$

Now, $\frac{q}{2} = \frac{331}{2} = 165.5 > |g \star m + p\, f \star r|_\infty$. This means that $\underline{g \star m + p\, f \star r \text{ is already centerlifted}}$, so it is equal to any centerlifted polynomial that is congruent to it modulo q.

Therefore indeed

$$\underline{cl_q(f \star e) = g \star m + p\, f \star r},$$

as desired.

(b) Explain the last step of the decryption process: once Alice has computed $\mathbf{a}$, how could she compute the original plaintext $\mathbf{m}$?

Alice knows that $a = g \star m + pf \star r \equiv g \star m \bmod p$.

Alice can compute $G_p \star a$ (where G_p is the inverse of g in R_p), which must satisfy

$$G_p \star a \equiv G_p \star g \star m \mod p$$
$$\equiv 1 \star m \mod p$$
$$\equiv m \mod p.$$

To find m, she can centerlift this mod p. In short,

$$m = cl_p(G_p \star a).$$

This page intentionally left blank. You may use it for scratchwork or to continue answers to any question (note clearly on the original page if you do so).

This page intentionally left blank. You may use it for scratchwork or to continue answers to any question (note clearly on the original page if you do so).

This page intentionally left blank. You may use it for scratchwork or to continue answers to any question (note clearly on the original page if you do so).

Reference tables from textbook:

Public parameter creation	
A trusted party chooses and publishes a (large) prime p and an integer g having large prime order in $\mathbb{F}_p^*$.	
Private computations	
Alice	**Bob**
Choose a secret integer a. Compute $A \equiv g^a \pmod p$.	Choose a secret integer b. Compute $B \equiv g^b \pmod p$.
Public exchange of values	
Alice sends A to Bob $\longrightarrow$	A
B	$\longleftarrow$ Bob sends B to Alice
Further private computations	
Alice	**Bob**
Compute the number $B^a \pmod p$.	Compute the number $A^b \pmod p$.
The shared secret value is $B^a \equiv (g^b)^a \equiv g^{ab} \equiv (g^a)^b \equiv A^b \pmod p$.	

Table 2.2: Diffie–Hellman key exchange

Public parameter creation	
A trusted party chooses and publishes a large prime p and an element g modulo p of large (prime) order.	
Alice	**Bob**
Key creation	
Choose private key $1 \le a \le p-1$. Compute $A = g^a \pmod p$. Publish the public key A.	
Encryption	
	Choose plaintext m. Choose random element k. Use Alice's public key A to compute $c_1 = g^k \pmod p$ and $c_2 = mA^k \pmod p$. Send ciphertext (c_1, c_2) to Alice.
Decryption	
Compute $(c_1^a)^{-1} \cdot c_2 \pmod p$. This quantity is equal to m.	

Table 2.3: Elgamal key creation, encryption, and decryption

Bob	Alice
Key creation	
Choose secret primes p and q. Choose encryption exponent e with $\gcd(e, (p-1)(q-1)) = 1$. Publish $N = pq$ and e.	
Encryption	
	Choose plaintext m. Use Bob's public key (N, e) to compute $c \equiv m^e \pmod N$. Send ciphertext c to Bob.
Decryption	
Compute d satisfying $ed \equiv 1 \pmod{(p-1)(q-1)}$. Compute $m' \equiv c^d \pmod N$. Then m' equals the plaintext m.	

Table 3.1: RSA key creation, encryption, and decryption

Samantha	Victor
Key creation	
Choose secret primes p and q. Choose verification exponent e with $\gcd(e, (p-1)(q-1)) = 1$. Publish $N = pq$ and e.	
Signing	
Compute d satisfying $de \equiv 1 \pmod{(p-1)(q-1)}$. Sign document D by computing $S \equiv D^d \pmod N$.	
Verification	
	Compute $S^e \bmod N$ and verify that it is equal to D.

Table 4.1: RSA digital signatures

Public parameter creation	
A trusted party chooses and publishes a large prime p and primitive root g modulo p.	
Samantha	**Victor**
Key creation	
Choose secret signing key $1 \le a \le p-1$. Compute $A = g^a \pmod p$. Publish the verification key A.	
Signing	
Choose document $D \bmod p$. Choose random element $1 < k < p$ satisfying $\gcd(k, p-1) = 1$. Compute signature $S_1 \equiv g^k \pmod p$ and $S_2 \equiv (D - aS_1)k^{-1} \pmod{p-1}$.	
Verification	
	Compute $A^{S_1} S_1^{S_2} \bmod p$. Verify that it is equal to $g^D \bmod p$.

Table 4.2: The Elgamal digital signature algorithm

Public parameter creation	
A trusted party chooses and publishes large primes p and q satisfying $p \equiv 1 \pmod q$ and an element g of order q modulo p.	
Samantha	**Victor**
Key creation	
Choose secret signing key $1 \le a \le q-1$. Compute $A = g^a \pmod p$. Publish the verification key A.	
Signing	
Choose document $D \bmod q$. Choose random element $1 < k < q$. Compute signature $S_1 \equiv (g^k \bmod p) \bmod q$ and $S_2 \equiv (D + aS_1)k^{-1} \pmod q$.	
Verification	
	Compute $V_1 \equiv DS_2^{-1} \pmod q$ and $V_2 \equiv S_1 S_2^{-1} \pmod q$. Verify that $(g^{V_1} A^{V_2} \bmod p) \bmod q = S_1$.

Table 4.3: The digital signature algorithm (DSA)

Public parameter creation	
A trusted party chooses and publishes a (large) prime p, an elliptic curve E over $\mathbb{F}_p$, and a point P in $E(\mathbb{F}_p)$.	
Private computations	
Alice	**Bob**
Chooses a secret integer n_A.	Chooses a secret integer n_B.
Computes the point $Q_A = n_A P$.	Computes the point $Q_B = n_B P$.
Public exchange of values	
Alice sends Q_A to Bob $\longrightarrow$	Q_A
Q_B $\longleftarrow$	Bob sends Q_B to Alice
Further private computations	
Alice	**Bob**
Computes the point $n_A Q_B$.	Computes the point $n_B Q_A$.
The shared secret value is $n_A Q_B = n_A(n_B P) = n_B(n_A P) = n_B Q_A$.	

Table 6.5: Diffie–Hellman key exchange using elliptic curves

Alice	Bob
Key Creation	
Choose a large integer modulus q. Choose secret integers f and g with $f < \sqrt{q/2}$, $\sqrt{q/4} < g < \sqrt{q/2}$, and $\gcd(f, qg) = 1$. Compute $h \equiv f^{-1}g \pmod q$. Publish the public key (q, h).	
Encryption	
	Choose plaintext m with $m < \sqrt{q/4}$. Use Alice's public key (q, h) to compute $e \equiv rh + m \pmod q$. Send ciphertext e to Alice.
Decryption	
Compute $a \equiv fe \pmod q$ with $0 < a < q$. Compute $b \equiv f^{-1}a \pmod g$ with $0 < b < g$. Then b is the plaintext m.	

Table 7.1: A congruential public key cryptosystem

Addendum to Table 7.1: The random element r (in "Encryption") should be chosen such that $r < \sqrt{q/2}$ as well.

Public parameter creation	
A trusted party chooses a finite field $\mathbb{F}_p$, an elliptic curve $E/\mathbb{F}_p$, and a point $G \in E(\mathbb{F}_p)$ of large prime order q.	
Samantha	**Victor**
Key creation	
Choose secret signing key $1 < s < q - 1$. Compute $V = sG \in E(\mathbb{F}_p)$. Publish the verification key V.	
Signing	
Choose document $d \bmod q$. Choose random element $e \bmod q$. Compute $eG \in E(\mathbb{F}_p)$ and then, $s_1 \equiv x(eG) \bmod q$ and $s_2 \equiv (d + ss_1)e^{-1} \pmod q$. Publish the signature (s_1, s_2).	
Verification	
	Compute $v_1 \equiv ds_2^{-1} \pmod q$ and $v_2 \equiv s_1 s_2^{-1} \pmod q$. Compute $v_1 G + v_2 V \in E(\mathbb{F}_p)$ and verify that $x(v_1 G + v_2 V) \bmod q = s_1$.

Table 6.7: The elliptic curve digital signature algorithm (ECDSA)

Public parameter creation	
A trusted party chooses public parameters (N, p, q, d) with N and p prime, $\gcd(p, q) = \gcd(N, q) = 1$, and $q > (6d + 1)p$.	
Alice	**Bob**
Key creation	
Choose private $\boldsymbol{f} \in \mathcal{T}(d + 1, d)$ that is invertible in R_q and R_p. Choose private $\boldsymbol{g} \in \mathcal{T}(d, d)$. Compute $\boldsymbol{F}_q$, the inverse of $\boldsymbol{f}$ in R_q. Compute $\boldsymbol{F}_p$, the inverse of $\boldsymbol{f}$ in R_p. Publish the public key $\boldsymbol{h} = \boldsymbol{F}_q \star \boldsymbol{g}$.	
Encryption	
	Choose plaintext $\boldsymbol{m} \in R_p$. Choose a random $\boldsymbol{r} \in \mathcal{T}(d, d)$. Use Alice's public key $\boldsymbol{h}$ to compute $\mathbf{e} \equiv p\boldsymbol{r} \star \boldsymbol{h} + \boldsymbol{m} \pmod q$. Send ciphertext $\boldsymbol{e}$ to Alice.
Decryption	
Compute $\boldsymbol{f} \star \boldsymbol{e} \equiv p\boldsymbol{g} \star \boldsymbol{r} + \boldsymbol{f} \star \boldsymbol{m} \pmod q$. Center-lift to $\boldsymbol{a} \in R$ and compute $\boldsymbol{m} \equiv \boldsymbol{F}_p \star \boldsymbol{a} \pmod p$.	

Table 7.4: NTRUEncryt: the NTRU public key cryptosystem

Addendum to Table 7.4:

- In "Encryption," you should assume that m is *centerlifted* modulo q.
- Recall: the notation $\mathcal{T}(d_1, d_2)$ denotes the set of all polynomials in R with exactly d_1 $+1$'s, d_2 -1's, and all other coefficients 0.