- The exam is *double-sided*. Make sure to read both sides of each page.
- The time limit is three hours.
- No calculators are permitted.
- You are permitted one page of notes, front and back.
- The textbook’s summary tables for the systems we have studied are provided at the back. There is also a multiplication table modulo 23. You may detach these sheets for easier reference.
- For any problem asking you to write a program, you may write in a language of your choice or in pseudocode, as long as your answer is sufficiently specific to tell the runtime of the program.

<table>
<thead>
<tr>
<th></th>
<th>/12</th>
<th></th>
<th></th>
<th>/7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>/7</td>
<td>4</td>
<td>/7</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>/7</td>
<td>6</td>
<td>/7</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>/7</td>
<td>8</td>
<td>/7</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>/7</td>
<td>10</td>
<td>/7</td>
<td></td>
</tr>
<tr>
<td>Σ</td>
<td></td>
<td></td>
<td></td>
<td>/75</td>
</tr>
</tbody>
</table>
(1) Briefly explain why each of the following choices is made in the cryptosystems we have studied (e.g. give a reason why it is necessary for the rest of the algorithm to work, why it makes a specific attack more difficult, or why it makes a computation more efficient).

(a) The element \( g \) in Diffie-Hellman (table 2.2) is chosen to have “large prime order.”

The order should be large so that collision algorithms (e.g. BSGS) are ineffective.

The order being prime ensures that the Pohlig-Hellman DLP algorithm cannot be used.

(b) In Elgamal encryption (table 2.3) and digital signatures (table 4.2), the number \( k \) is chosen randomly each time a document is encrypted or signed.

Repeating the same key, or using a closely related key, may allow Eve to solve for one plaintext in terms of another.

In the signature scheme, related keys may allow Eve to solve for the private (signing) key.

(c) The encryption exponent \( e \) in RSA (table 3.1) satisfies \( \gcd(e, (p-1)(q-1)) = 1 \).

Alice needs to find an inverse of \( e \) modulo \( (p-1)(q-1) \) (a decryption exponent); this won’t exist unless

\[ \gcd(e, (p-1)(q-1)) = 1. \]

(d) The two primes \( p, q \) is DSA (table 4.3) satisfy \( p \equiv 1 \pmod{q} \).

\( q \) must divide \( p-1 \), otherwise elements \( g \) of order \( q \) cannot exist.

Parts (e-h) on reverse side.
(e) The prime $p$ in ECDSA (table 6.7) can be chosen much smaller than the prime $p$ in DSA (table 4.3).

In DSA, $p$ must be large enough to detect DLP algorithm for $(\mathbb{Z}/p)^*$, e.g. the number field sieve.

For the ECDLP, the best alg. require $O(\sqrt{\text{ord}(g)})$ steps, allowing $p$ to be much smaller.

(f) The primes $p$ and $q$ in ECDSA are roughly the same size (same number of bits in length).

Hase's thm. states that $|E(\mathbb{F}_p)| \approx p^{(p+1)/2}$ when $\text{ord}(g) \approx p$,
and $q$ can be taken equal to $|E(\mathbb{F}_p)|$.

(g) In the congruential cryptosystem (table 7.1), the plaintext $m$ is chosen less than $\sqrt{q/4}$, rather than less than $\sqrt{q/2}$ like the numbers $f$, $g$ and $r$.

This bound, along with $q > \sqrt{q/4}$, ensures that $m < q$, so that learning $m \mod q$ at the end is enough to learn $m$.

(h) In NTRU (table 7.4), the element $f \in R$ is chosen from the set $T(d+1, d)$ rather than from the set $T(d, d)$ like the elements $g$ and $r$. (Recall that the notation $T(d_1, d_2)$ denotes the set of polynomials in $R$ with $d_1$ coefficients equal to 1, $d_2$ coefficients equal to $-1$, and all other coefficients equal to 0.)

Elements of $T(d,d)$ are never invertible in $R_p$ or $R_q$ (the sum of the coeffs. is 0); $f$ needs to have an inverse in both rings, for the rest of the computations.

(12 points)
(2) Find the smallest positive integer \( n \) such that all three of the following congruences hold.

\[
\begin{align*}
n &\equiv 3 \pmod{5} \\
n &\equiv 7 \pmod{8} \\
n &\equiv 0 \pmod{9}
\end{align*}
\]

\[
\exists h: \quad n = 3 + 5h
\]

\[
\Rightarrow \quad 3 + 5h \equiv 7 \pmod{8} \quad \Rightarrow \quad 5h \equiv 4 \pmod{8}
\]

Note \( 5^{-1} \equiv 5 \pmod{8} \) \( \text{(since 25 \equiv 1)} \), hence

\[
h \equiv 5 \cdot 4 \equiv 4 \pmod{8}; \quad \exists k \text{ s.t. } h = 4 + 8k.
\]

\[
\Rightarrow \quad n = 3 + 5(4 + 8k) = 23 + 40k \quad \text{(i.e. } n \equiv 23 \pmod{40}) \).
\]

\[
\Rightarrow \quad 23 + 40k \equiv 0 \pmod{9} \quad \Rightarrow \quad 5 + 4k \equiv 0 \pmod{9}
\]

\[
\Rightarrow \quad 4k \equiv -5 \equiv 4 \pmod{9}
\]

\[
\Rightarrow \quad k \equiv 1 \pmod{9} \quad \text{(4 invertible \pmod{9})}
\]

\[
\text{i.e. } \exists l \text{ s.t. } k = 1 + 9l
\]

\[
\Rightarrow \quad n = 23 + 40(1 + 9l) = 63 + 360l
\]

\[
\text{i.e. } \quad n \equiv 63 \pmod{360}.
\]

The smallest such \( n \) is \( n = 63 \).

More space for work on reverse side.

(7 points)
Additional space for problem 2.
(3) Let $p$ be a prime number, and $E$ be the elliptic curve over $\mathbb{F}_p$ described by $Y^2 \equiv X^3 + AX + B \pmod{p}$, where $A$ and $B$ are constants.

(a) Prove that given any integer $x$ with $0 \leq x < p$, there are at most two integers $y$ with $0 \leq y < p$ such that $(x, y) \in E(\mathbb{F}_p)$.

Any such $y$ must satisfy

$$y^2 \equiv x^3 + AX + B \pmod{p}.$$  

So if $y_1, y_2$ are two such, then $y_1^2 \equiv y_2^2 \pmod{p}$.

ie.  $y_1^2 - y_2^2 = (y_1 + y_2)(y_1 - y_2) \equiv 0 \pmod{p}$

ie.  $p | (y_1 + y_2)(y_1 - y_2)$.

Since $p$ is prime, either $p | (y_1 + y_2)$ or $p | (y_1 - y_2)$

ie. either $y_1 \equiv -y_2 \pmod{p}$ or $y_1 \equiv y_2 \pmod{p}$.

So given one such $y$, the only other possibility

is $y_2 = p - y_1$.

(3 points)

(b) Under what circumstances is there exactly one point on the elliptic curve with $X$-coordinate equal to $x$?

If $x^3 + AX + B \equiv 0 \pmod{p}$, then $y \equiv 0$ is the only matching $y$-coordinate, since $y \equiv -y \pmod{p}$ in this case (and only this case).

( the other possibility is $p=2$, which is usually ruled out when working with elliptic curves in this form ).

(1 point)
(c) Prove that if \( P, Q \in E(\mathbb{F}_p) \) are two points on the elliptic curve with the same \( X \)-coordinate, and \( n \) is any integer, then either \( n \cdot P \) and \( n \cdot Q \) are both equal to the point \( O \) at infinity, or both have the same \( X \)-coordinate.

If \( P = (x, y) \), then \( Q \) must be (either equal to \( P, 0) \)

\[
Q = (x, -y) \quad \text{(from part (a)} \]

\[
= \Theta P \quad \text{(inverse in group structure).} \]

If \( P = Q \), the result is clear, so assume \( P = \Theta Q \).

Then \( \ n \cdot P = (-n) \cdot Q = \Theta n \cdot Q \).

So either \( n \cdot P = O \) & \( n \cdot Q = \Theta O = O \),

or \( n \cdot P = (x, y) \) & \( n \cdot Q = \Theta (x', y') \)

\[
= (x', -y'), \]

which has the same \( x \)-coordinate.

(3 points)
(4) Write a function `decipher(c, p, q, e)`, and any necessary helper functions, to decipher messages encrypted with RSA. The input consists of the ciphertext $c$, the secret primes $p, q$, and the encryption exponent $e$ (notation as in table 3.1).

You should implement any helper functions you use that are not built into Python, or the standard programming language of your choice. You may assume that a fast modular exponentiation function `pow(a, b, m)` (returning $a^{b\mod m}$) is built-in (as it is in Python).

```python
def inverse(a, m):
    # based on extended eucl alg.
    pre = 0, m
    cur = 1, a
    # pre, cur will be of form v, q for an equation
    while cur[1] > 0:
        k = pre[1] / cur[1]
        nxt = pre[0] - k * cur[0], pre[1] - k * cur[1]
        pre, cur = cur, nxt
    v, q = pre
    # q = gcd(q, m)
    if q != 1:
        return None
    return v % m

def decipher(c, p, q, e):
    d = inverse(e, (p-1)*(q-1))
    return pow(c, e, p*q)
```

More space for work on reverse side. 

(7 points)
Additional space for problem 4.
(5) Suppose that Alice and Bob are using NTRU with parameters \((N, q, p, d) = (5, 23, 3, 1)\) (notation as in table 7.4). Alice’s public key is
\[
h = 21 + 14x + 13x^2 + 4x^3 + 17x^4.
\]
Bob wishes to encipher the message
\[
m = 1 + x + x^2 - x^4.
\]
Find a valid ciphertext \(e\) that Bob might compute to send this message. (There are many possible answers; you only need to give one.)

Note that a multiplication table for \(\mathbb{Z}/23\) is provided at the back of the exam packet, which may be useful in your computations.

\[
e = m + 3(21 + 14x + 13x^2 + 4x^3 + 17x^4) \ast r \quad \text{when } r \in \mathbb{T}(1,1).
\]

\text{eg. we can select } r = 1 - x. \text{ In this case:}
\[
h \ast r = 21 + 14x + 13x^2 + 4x^3 + 17x^4
-21 - 14x^2 - 13x^3 - 4x^4 - 17
\equiv 4 + 16x + 22x^2 + 14x^3 + 13x^4
\]
using chart,
\[
p \cdot h \ast r = 12 + 2x + 20x^2 + 19x^3 + 16x^4
\]
hence
\[
e = 13 + 3x + 21x^2 + 19x^3 + 15x^4
\]
(or, in centered lifted form, \(-10 + 3x - 2x^2 - 4x^3 - 8x^4\))

// there are 20 elements of \(\mathbb{T}(1,1)\), so 19 other answers possible.
Additional space for problem 5.
(6) Let $p$ be a prime number, and $a$ an integer with $1 \leq a \leq p - 1$.

(a) Define the order of $a$ modulo $p$.

$$\text{ord}[a]_p = \text{minimum positive integer } e \text{ s.t. } a^e \equiv 1 \text{ mod } p.$$  

Equivalently, the period of the sequence $\{a^e \mod p : e \in \mathbb{Z}\}$, or the number of distinct numbers in this sequence.

(2 points)

(b) Define what it means for $a$ to be a primitive root modulo $p$.

$$a \text{ is a primitive root } := \text{ord}[a]_p = p - 1.$$  

Equivalently, all nonzero $[b]_p \in \mathbb{Z}/p\mathbb{Z}$ are powers of $[a]_p$ (so discrete logarithms are well-defined).

(2 points)

(c) Let $p = 7$. For each choice of $a$ from 1 to 6 inclusive, determine the order of $a$, and identify whether or not it is a primitive root.

<table>
<thead>
<tr>
<th>$a$</th>
<th>powers of $a \mod 7$</th>
<th>order</th>
<th>prim. root?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1, 1, 1, ...</td>
<td>1</td>
<td>no</td>
</tr>
<tr>
<td>2</td>
<td>2, 4, 1, ...</td>
<td>3</td>
<td>no</td>
</tr>
<tr>
<td>3</td>
<td>3, 2, 6, 4, 5, 1, ...</td>
<td>6</td>
<td>yes</td>
</tr>
<tr>
<td>4</td>
<td>4, 2, 1, ...</td>
<td>3</td>
<td>no</td>
</tr>
<tr>
<td>5</td>
<td>5, 4, 6, 2, 3, 1, ...</td>
<td>6</td>
<td>yes</td>
</tr>
<tr>
<td>6</td>
<td>6, 1, ...</td>
<td>2</td>
<td>no</td>
</tr>
</tbody>
</table>

More space for work on reverse side.  (3 points)
Additional space for problem 6.
Each day, Alice and Bob perform Elliptic Curve Diffie-Hellman key exchange (notation as in table 6.5) to establish an encryption key for the day. Each day they use the same public parameters: the prime $p = 23$, curve $Y^2 \equiv X^3 + 2X + 6 \pmod{23}$, and the point $P = (1,3)$.

On Monday, Alice and Bob exchange the values

$$Q_A = (18,20) \quad Q_B = (4,3)$$

and establish the shared secret $S = (19,7)$. Due to careless data management, Eve manages to learn all three of these values.

On Tuesday, Alice and Bob exchange the values

$$Q_A' = (5,16) \quad Q_B' = (18,3)$$

and establish the shared secret $S'$, which Eve is not able to intercept. However, Eve does notice that, due to poor random number generation by both Alice and Bob, these values are related to Monday’s values by the equations

$$Q_A' = Q_A \oplus P \quad Q_B' = 2 \cdot Q_B.$$

Use this information to determine the new shared secret $S'$. There is a multiplication table for $\mathbb{Z}/23$ at the back of the exam packet that may be useful in your computations. For partial credit you may express your answer in terms of the given points and elliptic curve operations; for full credit you should calculate the coordinates explicitly.

$$\begin{align*}
S' &= \lambda \cdot n_A \cdot n_B \cdot P = \lambda \cdot n_A \cdot n_B \cdot (Q_B) = n_A \cdot (2 \cdot Q_B) \\
&= 2 \cdot (n_A \cdot Q_B) = 2 \cdot (n_A \cdot n_B \cdot P) = 2 \cdot n_B \cdot (Q_A) = 2 n_B \cdot (Q_A \oplus P) \\
&= 2(n_B \cdot Q_A) \oplus (2 \cdot n_B) P = 2 \cdot S \oplus 2 \cdot Q_B.
\end{align*}$$

We could compute this either as $2 \cdot S \oplus Q_B = 2 \cdot (19,7) \oplus (18,3)$, or as $2 \cdot (S \oplus Q_B) = 2 \cdot ((19,7) \oplus (4,3))$.

Here’s how to do the first option: (chartured for all multiplication)

1) $(19,7) \oplus (19,7)$

$\lambda = (3 \cdot 19 + 2 \cdot (2,7))^{-1}$

$\equiv 4 \cdot 5 \equiv 20$

$X_3 = 20 \cdot 19 \cdot 19$

$\equiv 17 \pmod{23}$

$Y_3 = -7 - 20 \cdot (17-19)$

$\equiv -13 \equiv 10$

$\Rightarrow (19,7) = (17,10)$

2) $(17,10) \oplus (18,3)$

$\lambda = \frac{1}{(10-7) \cdot (17-18)}$

$\equiv 16 \pmod{23}$

$X_7 = 16 \cdot 17 - 18$

$\equiv 14 \pmod{23}$

$Y_7 = -16 + 15 \cdot (17-18)$

$\equiv -8 \equiv 15 \pmod{23}$

$\Rightarrow (17,10) \oplus (18,3) = (14,15)$

So the new shared secret is

$$S' = (14,15)$$

Alternatively, one can compute

$$(19,7) \oplus (4,3) = (6,21)$$

and

$$2 \cdot (6,21) = (14,15).$$

More space for work on reverse side.

(7 points)
Additional space for problem 7.
(8) (a) Estimate the number of 512-bit prime numbers (that is, prime numbers between $2^{511}$ and $2^{512} - 1$ inclusive). Your answer will be marked correct if it within a factor of 10 of the correct figure, and may be expressed in terms of standard mathematical functions (exponentials, logarithms, etc.).

Prime number theorem: roughly one in $\ln(2^{512}) = 512 \ln 2$

512-bit numbers are prime.

So the number of 512-bit primes is roughly $\frac{2^{511}}{512 \ln 2}$.

(2 points)

(b) Assume that you have implemented a function `is_prime(n)` that efficiently determines whether or not $n$ is prime, and returns either True or False. Write a function `safe_prime()` that returns a 512-bit prime number $p$ such that the number $p - 1$ has at least one prime factor that is at least 256 bits long.

```python
import random
def make_prime(bits):
    while True:
        p = random.randrange(2**bits, 2**(bits+1))
        if is_prime(p): return p

def safe_prime():
    q = make_prime(256) # make a factor of p-1.
    # will set p = k*q + 1 for some k
    min_k = (2**511)/q + 1
    max_k = (2**512)/q
    while True:
        k = random.randrange(min_k, max_k)
        p = k*q + 1
        if is_prime(p):
            return p
```

(5 points)
Additional space for problem 8.
(9) Consider the following variation on the NTRU cryptosystem. In advance, Alice and Bob agree to the following public parameters.
\[ N = 503, \quad q = 257, \quad p = 3 \]

Privately, Alice chooses three polynomials at random, from the following sets. She keeps these polynomials secret; they constitute her private key.
\[ f \in \mathcal{T}(101, 100), \quad g_1 \in \mathcal{T}(31, 30), \quad g_2 \in \mathcal{T}(10, 10) \]
(Recall that \( \mathcal{T}(d, e) \) denotes the subset of the ring \( R = \mathbb{Z}[X]/(X^N - 1) \), where elements are represented as a list of \( N \) coefficients, consisting of polynomials with exactly \( d \) coefficients equal to 1, \( e \) coefficients equal to \(-1\), and the rest of the coefficients equal to 0.)

Alice ensures that \( f \) is invertible modulo \( q \) (otherwise she chooses a new value), with inverse \( F_q \in R_q \). She then computes the following two elements of \( R_q \). She distributes these values; they constitute her public key.
\[ h_1 = F_q \ast g_1 \pmod{q}, \quad h_2 = F_q \ast g_2 \pmod{q} \]

To send messages, Bob chooses a plaintext \( m \in R_p \), chooses a random ephemeral key \( r \in \mathcal{T}(10, 10) \), and computes a ciphertext \( e \in R_q \) as follows:
\[ e \equiv h_1 \ast c_l(p)(m) + ph_2 \ast r \pmod{q}. \]
(Here \( c_l(p) \) denotes centerlifting from \( R_q \) to \( R \); in the case \( p = 3 \) this gives a polynomials with all coefficients equal to \(-1, 0, \) or \( 1\).)

(a) Describe a procedure that Alice can use to recover the plaintext \( m \) from the ciphertext \( e \). You may need to make an additional assumption about an element being invertible in a ring.

First compute \( f \ast e \pmod{q} \); this is \( \equiv q_i \ast c_l(p)(m) + pq_i \ast r \pmod{q} \) since \( f \ast F_q \ast g_i \equiv q_i \pmod{q} \).

Centerlift this to obtain a polynomial \( a = c_l(f \ast e) \in R \).

Compute \( (g_i^{-1}) \ast a \pmod{p} \), where \( g_i^{-1} \) is the inverse of \( g_i \) in \( R_p \) (i.e. modulo \( p \)).

This will be the plaintext \( m \).

Part (b) on reverse side.
(b) Prove that the method you describe in part (a) will succeed, given the specific parameters specified above.

We know that

\[ a \equiv q_1 \cdot m + q_3 \cdot q_2 \cdot n \mod 257. \]

As long as the RHS has all coeffs. between \(-257/2 \& 257/2\), it will be its own cent uplift; since a is centered, that will ensure

\[ a = q_1 \cdot m + q_3 \cdot q_2 \cdot n \quad \text{(exact equality, in } R) \]

and in turn

\[ a \equiv q_1 \cdot m \mod 3 \& (q_1^{-1}) \cdot a \equiv m \mod 3, \]

so this recovers the plaintext.

So it suffices to show that \(| q_1 \cdot m + q_3 \cdot q_2 \cdot n |_\infty \leq \frac{257}{2} = 128.5 \).

From a lemma in class, \( q_1 \in \mathcal{T}(31, 30) \& | m |_{\infty} \leq 1 \) implies

\[ |q_1 \cdot m|_\infty \leq (31+30) \cdot 1 = 61 \]

\& \( q_2 \in \mathcal{T}(10, 10) \), \(|n|_{\infty} \leq 1 \) implies

\[ |q_3 \cdot q_2 \cdot n|_\infty \leq (10+10) \cdot 1 = 20 \]

\[ \Rightarrow |13 \cdot q_2 \cdot n|_\infty \leq 3 \cdot 20 = 60. \]

By the triangle inequality,

\[ |q_1 \cdot m + q_3 \cdot q_2 \cdot n|_\infty \]

\[ \leq |q_1 \cdot m| + 3 \cdot |q_2 \cdot n|_\infty \]

\[ \leq 61 + 60 = 121 < \frac{257}{2}. \]

So indeed \( a = q_1 \cdot m + q_3 \cdot q_2 \cdot n \),

and decryption always succeeds. (4 points)
(10) Suppose that $p$ and $q$ are prime numbers, $E$ is an elliptic curve over $\mathbb{F}_p$, and $G \in E(\mathbb{F}_p)$ is a point of order $q$.

Samantha and Victor are making use of the following signature scheme, similar to ECDSA. Samantha has a secret signing key $s$ ($1 < s < q - 1$), and a verification key $V = s \cdot G$, which is public information. A signature consists of a pair $(s_1, s_2)$ of integers, both between 0 and $q - 1$ inclusive, and a document consists of an integer $d$ from 1 to $q - 1$ inclusive. Victor will consider a signature $(s_1, s_2)$ valid for the document $d$ if the following equation holds:

$$x((d^{-1}s_1) \cdot V \oplus (d^{-1}s_2) \cdot G) \% q = s_1$$

Here $d^{-1}$ denotes the inverse modulo $q$, and $x(P)$ denotes the $x$-coordinate of a point $P$ on $E(\mathbb{F}_p)$.

(a) Suppose that Samantha wishes to sign a document $d$, and she begins by choosing a random ephemeral key $e$, and computing $s_1 = x(e \cdot G) \% q$. Explain a method she can use to compute a value $s_2$ such that $(s_1, s_2)$ will be a valid signature for $d$.

It's enough to ensure that $(d^{-1}s_1) \cdot V \oplus (d^{-1}s_2) \cdot G = e \cdot G$

ie. $d^{-1}s_1 \cdot s + d^{-1}s_2 \equiv e \mod q$ (since $\text{ord} G = q$)

ie. $s_1 \cdot s + s_2 \equiv d \cdot e \mod q$.

So Samantha can compute $s_2$ as

$$s_2 \equiv d \cdot e - s_1 \cdot s \mod q.$$
(b) Suppose that Eve wishes to forge a valid signature for this system. As in the “blind forgery” methods we’ve discussed in class, she will not be able to choose the document \(d\) in advance. Instead, she begins by choosing two integers \(i\) and \(j\) at random from 1 to \(q - 1\) inclusive, and computes 
\[ s_1 = x (i \cdot G \oplus j \cdot V)^\% q. \]
Explain a method Eve can use to compute a value of \(s_2\) and a value of \(d\), so that \((s_1, s_2)\) will be a valid signature for the document \(d\) (even though \(d\) will likely appear to be gibberish).

It's enough to ensure that
\[
(d^{-1} s_1) V \oplus (d^{-1} s_2) G = i \cdot G \oplus j \cdot V.
\]

For then, it suffices to ensure that
\[
d^{-1} s_1 \equiv j \mod q \\
&
d^{-1} s_2 \equiv i \mod q.
\]

So Eve can compute
\[
d \equiv j^{-1} \cdot s_1 \mod q \quad \text{(note that here we need } j \neq 0 \mod q \text{)}
\]
\[
\&
S_2 \equiv d \cdot i \mod q 
\equiv i \cdot j^{-1} s_1 \mod q.
\]

(3 points)

(c) Explain briefly how Samantha and Victor could modify this signature scheme using a hash function, in order to make Eve’s method in (b) infeasible.

If \(h\) is a secure hash function, (with output in \([0, q - 1]\)), they can use \(h(d)\) instead of \(d\) in the vent. eqn.

Eve’s attack in (b) is now useless since she would have to invert the hash function to get a doc. \(d\) hashing to the value \(j^{-1} s_1 \mod q\).

(1 point)
"Bonus" (to keep me happy during grading, not for real points): fill in cryptography-related (or totally unrelated) dialog for this comic.
Additional space for work.
A trusted party chooses and publishes a (large) prime \( p \) and an integer \( g \) having large prime order in \( \mathbb{F}_p^* \).

### Private computations

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose a secret integer ( a ).</td>
<td>Choose a secret integer ( b ).</td>
</tr>
<tr>
<td>Compute ( A = g^a \mod p ).</td>
<td>Compute ( B = g^b \mod (mod p) ).</td>
</tr>
</tbody>
</table>

### Public exchange of values

- Alice sends \( A \) to Bob
- Bob sends \( B \) to Alice

### Further private computations

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute the number ( B^a \mod p ).</td>
<td>Compute the number ( A^b \mod p ).</td>
</tr>
<tr>
<td>The shared secret value is ( B^a \equiv (g^a)^a \equiv g^{ab} \equiv (g^a)^b \equiv A^b \mod p ).</td>
<td></td>
</tr>
</tbody>
</table>

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.

#### Key creation

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose private key ( 1 \leq a \leq p-1 ).</td>
<td>Compute ( A = g^a \mod (mod p) ).</td>
</tr>
<tr>
<td>Publish the public key ( A ).</td>
<td>Publish the public key ( A ).</td>
</tr>
</tbody>
</table>

#### Encryption

Choose plaintext \( m \).
Choose random element \( k \).
Use Alice’s public key \( A \) to compute \( c_1 = g^k \mod p \) and \( c_2 = m \cdot A^k \mod (mod p) \).
Send ciphertext \((c_1,c_2) \) to Alice.

#### Decryption

\[ \left( c_2 \right)^{a-1} \cdot c_1 \mod p \]
This quantity is equal to \( m \).

Table 2.3: Elgamal key creation, encryption, and decryption

### Public parameter creation

A trusted party chooses and publishes a large prime \( p \) and an element \( g \) modulo \( p \) of large (prime) order.

#### Key creation

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose secret signing key ( 1 \leq a \leq p-1 ).</td>
<td>Compute ( A = g^a \mod (mod p) ).</td>
</tr>
<tr>
<td>Publish the verification key ( A ).</td>
<td>Publish the verification key ( A ).</td>
</tr>
</tbody>
</table>

#### Signing

Choose document \( D \mod p \).
Choose random element \( 1 < k < p \) satisfying \( \gcd(k,p-1) = 1 \).
Compute signature \( S_1 = g^k \mod p \) and \( S_2 = (D-aS_1)k^{-1} \mod (p-1) \).

#### Verification

\[ A^{S_1} S_2^p \mod p \]
Verify that it is equal to \( g^D \mod p \).

Table 3.1: RSA key creation, encryption, and decryption

### Public parameter creation

A trusted party chooses and publishes large primes \( p \) and \( q \) satisfying \( p \equiv 1 \mod q \) and an element \( g \) of order \( q \) modulo \( p \).

#### Key creation

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose secret signing key ( 1 \leq a \leq q-1 ).</td>
<td>Compute ( A = g^a \mod (mod p) ).</td>
</tr>
<tr>
<td>Publish the verification key ( A ).</td>
<td>Publish the verification key ( A ).</td>
</tr>
</tbody>
</table>

#### Signing

Choose document \( D \mod q \).
Choose random element \( 1 < k < q \).
Compute signature \( S_1 = g^k \mod q \) and \( S_2 = (D-aS_1)k^{-1} \mod q \).

#### Verification

\[ V_1 = D S_2^{-1} \mod q \]
\[ V_2 = S_1 S_2^{-1} \mod q \].
Verify that \( (g^V_1 A^{V_2} \mod p) \mod q = S_1 \).

Table 4.2: The Elgamal digital signature algorithm

### Public parameter creation

A trusted party chooses and publishes a large prime \( p \) and \( q \).

#### Key creation

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose secret primes ( p ) and ( q ).</td>
<td>Choose verification exponent ( e ) with ( \gcd(e, (p-1)(q-1)) = 1 ).</td>
</tr>
<tr>
<td>Publish ( N = pq ) and ( e ).</td>
<td>Publish ( N = pq ) and ( e ).</td>
</tr>
</tbody>
</table>

#### Signing

Compute \( d \) satisfying \( de \equiv 1 \mod (p-1)(q-1) \).
Sign document \( D \) by computing \( S \equiv D^d \mod (mod N) \).

#### Verification

Compute \( S^e \mod N \) and verify that it is equal to \( D \).

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

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chooses a secret integer $n_A$.</td>
<td>Chooses a secret integer $n_B$.</td>
</tr>
<tr>
<td>Computes the point $Q_A = n_A P$.</td>
<td>Computes the point $Q_B = n_B P$.</td>
</tr>
</tbody>
</table>

Public exchange of values

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sends $Q_A$ to Bob</td>
<td>Sends $Q_B$ to Alice</td>
</tr>
</tbody>
</table>

Further private computations

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computes the point $n_A Q_B$.</td>
<td>Computes the point $n_B Q_A$.</td>
</tr>
<tr>
<td>The shared secret value is $n_A Q_B = n_A (n_B P) = n_B (n_A P) = n_B Q_A$.</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.5: Diffie–Hellman key exchange using elliptic curves

<table>
<thead>
<tr>
<th>Public parameter creation</th>
</tr>
</thead>
<tbody>
<tr>
<td>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 $q$.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Samantha</th>
<th>Victor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose secret signing key $1 &lt; s &lt; q - 1$.</td>
<td>Compute $V = sG$ in $E(\mathbb{F}_p)$.</td>
</tr>
<tr>
<td>Publish the verification key $V$.</td>
<td></td>
</tr>
</tbody>
</table>

Signing

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose document $d$ mod $q$.</td>
<td>Choose random element $e$ mod $q$.</td>
</tr>
</tbody>
</table>
| Compute $eG$ in $E(\mathbb{F}_p)$ and then, $s_1 = x(eG) \pmod{q}$ and $s_2 = (d + s_1)e^{-1} \pmod{q}$. | Compute $V$.
| Publish the signature $(s_1, s_2)$. |

Verification

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute $v_1 = ds_2^{-1} \pmod{q}$ and $v_2 = s_1 s_2^{-1} \pmod{q}$.</td>
<td>Compute $v_1 G + v_2 V$ in $E(\mathbb{F}_p)$ and verify that $x(v_1 G + v_2 V) \pmod{q} = s_1$.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Public Parameter Creation</th>
</tr>
</thead>
<tbody>
<tr>
<td>A trusted party chooses and publishes a (large) prime $p$, an elliptic curve $E$ over $\mathbb{F}_p$, and a point $P$ in $E(\mathbb{F}_p)$.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chooses a secret multiplier $n_A$.</td>
<td>Computes $Q_A = n_A P$.</td>
</tr>
<tr>
<td>Publishes the public key $Q_A$.</td>
<td></td>
</tr>
</tbody>
</table>

Encryption

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chooses plaintext values $m_1$ and $m_2$ modulo $p$.</td>
<td>Computes $R = k P$.</td>
</tr>
<tr>
<td>Chooses a random number $k$.</td>
<td>Computes $S = kQ_A$ and writes it as $S = (x_s, y_s)$.</td>
</tr>
<tr>
<td>Computes $c_1 \equiv x_s m_1 \pmod{p}$ and $c_2 \equiv y_s m_2 \pmod{p}$.</td>
<td>Sends ciphertext $(R, c_1, c_2)$ to Alice.</td>
</tr>
</tbody>
</table>

Decryption

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computes $T = n_AR$ and writes it as $T = (x_T, y_T)$.</td>
<td>Computes $T = n_AR$ and writes it as $T = (x_T, y_T)$.</td>
</tr>
<tr>
<td>Sets $m_1' \equiv x_T^{-1} c_1 \pmod{p}$ and $m_2' \equiv y_T^{-1} c_2 \pmod{p}$. Then $m_1 = m_1'$ and $m_2 = m_2'$.</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.13: Menezes–Vanstone variant of ElGamal (Exercises 6.17, 6.18)

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 $\gcd(f, g) = 1$. 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 $e \equiv fe \pmod{q}$ with $0 < a < q$. Compute $b \equiv f^{-1} a \pmod{q}$ with $0 < b < q$. Then $b$ is the plaintext $m$. |

Table 7.1: A congruential public key cryptosystem

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key creation</td>
<td></td>
</tr>
<tr>
<td>Choose private $f \in \mathbb{T}(d+1, d)$ that is invertible in $R_q$ and $R_p$. Choose private $g \in \mathbb{T}(d, d)$. Compute $F_q$, the inverse of $f$ in $R_q$. Compute $F_p$, the inverse of $f$ in $R_p$. Publish the public key $h = F_q * g$.</td>
<td></td>
</tr>
</tbody>
</table>

**Encryption**
Choose plaintext $m \in R_p$. Choose a random $r \in \mathbb{T}(d, d)$. Use Alice’s public key $h$ to compute $e \equiv pr + m \pmod{q}$. Send ciphertext $e$ to Alice. |

**Decryption**
Compute $f * e \equiv pg + m \pmod{q}$ (central lift from $R_p$ to $R$). Center-lift to $a \in R$ and compute $m \equiv F_p * a \pmod{p}$. |

Table 7.4: NTRUEncrypt: the NTRU public key cryptosystem

**Relevant definitions:** (in NTRU)
$\mathbb{T}(d, d) = \{ \text{elements of } R \text{ with exactly } d \text{ coefficients equal to } 1 \text{ and the rest equal to } 0 \}$. |

$R_q = (\mathbb{Z}/q)[x]/(x^n - 1)$
## Multiplication table modulo 23

|   | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 1 | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| 2 | 2  | 4  | 6  | 8  | 10 | 12 | 14 | 16 | 18 | 20 | 22 | 1  | 3  | 5  | 7  | 9  | 11 | 13 | 15 | 17 | 19 | 21 |
| 3 | 3  | 6  | 9  | 12 | 15 | 18 | 21 | 4  | 7  | 10 | 13 | 16 | 19 | 22 | 2  | 5  | 8  | 11 | 14 | 17 | 20 | 3  |
| 4 | 4  | 8  | 12 | 16 | 20 | 1  | 5  | 9  | 13 | 17 | 21 | 2  | 6  | 10 | 14 | 18 | 22 | 3  | 7  | 11 | 15 | 19 |
| 5 | 5  | 10 | 15 | 20 | 2  | 7  | 12 | 17 | 22 | 4  | 9  | 14 | 19 | 6  | 11 | 16 | 21 | 3  | 8  | 13 | 18 | 1 |
| 6 | 6  | 12 | 18 | 1  | 7  | 13 | 19 | 2  | 8  | 14 | 20 | 3  | 9  | 15 | 21 | 4  | 10 | 16 | 22 | 5  | 11 | 17 |
| 7 | 7  | 14 | 21 | 5  | 12 | 19 | 3  | 10 | 17 | 1  | 8  | 15 | 22 | 6  | 13 | 20 | 4  | 11 | 18 | 2  | 9  | 16 |
| 8 | 8  | 16 | 1  | 9  | 17 | 2  | 10 | 18 | 3  | 11 | 19 | 4  | 12 | 20 | 5  | 13 | 21 | 6  | 14 | 22 | 7  | 15 |
| 9 | 9  | 18 | 4  | 13 | 22 | 8  | 17 | 3  | 12 | 21 | 7  | 16 | 2  | 11 | 20 | 6  | 15 | 1  | 10 | 19 | 5  | 14 |
| 10| 10 | 20 | 7  | 17 | 4  | 14 | 1  | 11 | 21 | 8  | 18 | 5  | 15 | 2  | 12 | 22 | 9  | 19 | 6  | 16 | 3  | 13 |
| 11| 11 | 22 | 10 | 21 | 9  | 20 | 8  | 19 | 7  | 18 | 6  | 17 | 5  | 16 | 4  | 15 | 3  | 14 | 2  | 13 | 1  | 12 |
| 12| 12 | 1  | 13 | 2  | 14 | 3  | 15 | 4  | 16 | 5  | 17 | 6  | 18 | 7  | 19 | 8  | 20 | 9  | 21 | 10 | 22 | 11 |
| 13| 13 | 3  | 16 | 6  | 19 | 9  | 22 | 12 | 5  | 15 | 8  | 21 | 11 | 1  | 14 | 4  | 17 | 1  | 14 | 7  | 17 | 10 |
| 14| 14 | 5  | 19 | 10 | 1  | 15 | 6  | 20 | 11 | 2  | 16 | 7  | 21 | 12 | 3  | 17 | 8  | 22 | 13 | 4  | 18 | 9  |
| 15| 15 | 7  | 22 | 14 | 6  | 21 | 13 | 5  | 20 | 12 | 4  | 19 | 11 | 3  | 18 | 10 | 2  | 17 | 9  | 1  | 16 | 8  |
| 16| 16 | 9  | 2  | 18 | 11 | 4  | 20 | 13 | 6  | 22 | 15 | 8  | 1  | 17 | 10 | 3  | 19 | 12 | 5  | 21 | 14 | 7  |
| 17| 17 | 11 | 5  | 22 | 16 | 10 | 4  | 21 | 15 | 9  | 3  | 20 | 14 | 8  | 2  | 19 | 13 | 7  | 1  | 18 | 12 | 6  |
| 18| 18 | 13 | 8  | 3  | 21 | 16 | 11 | 6  | 1  | 19 | 14 | 9  | 4  | 22 | 17 | 12 | 7  | 2  | 20 | 15 | 10 | 5  |
| 19| 19 | 15 | 11 | 7  | 3  | 22 | 18 | 14 | 10 | 6  | 2  | 21 | 17 | 13 | 9  | 5  | 1  | 20 | 16 | 12 | 8  | 4  |
| 20| 20 | 17 | 14 | 11 | 8  | 5  | 2  | 22 | 19 | 16 | 13 | 10 | 7  | 4  | 1  | 21 | 18 | 15 | 12 | 9  | 6  | 3  |
| 21| 21 | 19 | 17 | 15 | 13 | 11 | 9  | 7  | 5  | 3  | 1  | 22 | 20 | 18 | 16 | 14 | 12 | 10 | 8  | 6  | 4  | 2  |
| 22| 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  |
This page intentionally left blank.