

Note (2024): this exam is from an older version of this course, and is a bit longer and more difficult than our exam is likely to be.

1. [10 points] Consider the elliptic curve $Y^2 = X^3 + X - 1$ over $\mathbb{Z}/5\mathbb{Z}$.
 - (a) Determine the number of points on this curve (including the point \mathcal{O}).
 - (b) Determine the order of the point $P = (1, 1)$.
2. [10 points] Explain briefly why each of the following choices is made in DSA. Be specific about which mathematical facts would make the algorithm either incorrect or insecure otherwise.
 - (a) The number q is a *prime* number.
 - (b) The numbers p, q satisfy $p \equiv 1 \pmod{q}$.
 - (c) The number k is selected *at random*.

3. [10 points] Alice's RSA public key has modulus N . Bob cannot remember whether her encrypting exponent is 16 or 27. In a well-meaning but very foolish blunder, he decides to encrypt his message m with *both* possible encrypting exponents, creating c_1 (using $e = 16$) and c_2 (using $e = 27$). Bob uses the correct modulus N in both cases. He then sends both c_1 and c_2 to Alice, with an explanation of what happened. Eve intercepts c_1 and c_2 , as well as the information of which exponent was used to create which ciphertext.

Express m in terms of c_1 and c_2 using arithmetic modulo N . This will show that Eve can learn the plaintext m without much effort.

4. [10 points] The following function definition is meant to calculate the sum of two points P, Q on the elliptic curve $Y^2 = X^3 + AX + B$ over $\mathbb{Z}/p\mathbb{Z}$, but it contains a flaw. Explain the case in which the code will not work properly, and how to fix it.

Assumptions: each point (P, Q , or the return value) is either a pair (x, y) of two integers with $0 \leq x, y < p$, or the number 0 (for the point \mathcal{O}). You may assume that both P and Q do in fact lie on the curve defined by A and B . Also assume that `inv_mod(a,m)` is a correctly implemented function that returns the inverse of a modulo m whenever a is a unit modulo m , but which results in an error if a is not a unit modulo m .

```
def add(P,Q,A,B,p):
    if P==0: return Q
    if Q==0: return P
    if P[0] == Q[0] and P[1] != Q[1]: return 0
    if P[0] != Q[0]:
        rise = (P[1] - Q[1]) % p
        run = (P[0] - Q[0]) % p
    else:
        rise = (3*P[0]*P[0] + A) % p
        run = (2*P[1]) % p
    slope = (rise*inv_mod(run,p)) % p
    y_int = (P[1] - P[0]*slope) % p
```

```

x = (slope*slope - P[0]-Q[0]) % p
y = (-(slope*x + y_int)) % p
return (x,y)

```

5. [10 points] Write a function `pickg(p,q)` with the following behavior: if p, q are both prime numbers, then the return value must be either a number a between 1 and $p - 1$ inclusive with order q modulo p , or the number -1 if no such integer a exists. Your function may be randomized. For full points the (expected value of the) number of arithmetic operations performed by the function must be $\mathcal{O}(\log p)$.
6. [10 points] Suppose that Samantha is using ECDSA parameters with $q = 7$. She has published two valid signatures: $(2, 3)$ for the document $d = 4$, and $(2, 6)$ for the document $d' = 5$. Eve learns that she used the same random element e to produce both signatures. Determine Samantha's secret signing key, s .

Note. I am withholding the information of Samantha's public key and the system parameters for this problem, since the numbers are small enough that a brute force solution would be possible. In reality, of course, Eve would know all of this, but q would also be large enough that brute force would not be feasible.

7. [10 points] Suppose that Eve has intercepted a ciphertext from Bob to Alice. In addition, she knows by other means that the plaintext is one of only 1000 possibilities (for example, it might specify a landmark where Alice and Bob will meet, written in a predictable format and chosen from a short list of options). As usual, Eve knows Alice's public key, but not her private key.
 - (a) Suppose that the cryptosystem being used is RSA. Explain how Eve can very quickly identify for certain which of the 1000 candidates is the true plaintext.
 - (b) Suppose that the cryptosystem being used is Menezes-Vanstone (table 6.13). Describe a procedure Eve could use that, *with very high probability*, will pick out the correct plaintext from the list. (More formally: your procedure should have the property that if the 999 false plaintexts were chosen uniformly at random, then the probability of choosing one of them should be negligible.)
8. [10 points] **Note (2024): This problem concerns a cryptosystem we did not discuss this semester.** The NTru procedure (table 7.4) stipulates that p and q should be chosen such that $\gcd(p, q) = 1$. Suppose that parameters are chosen that do not obey this rule, and instead $p \mid q$. In this case, the system is completely insecure. Write a function that Eve could use to can break it.

Specifically: write a function `extract(e,N,p,q,d,h)` that efficiently extracts the plaintext \mathbf{m} from any cipher text \mathbf{e} , given only the public key and system parameters, and assuming that p divides q . The arguments \mathbf{e} and \mathbf{h} will be given as lists of N integers. The coefficients in your answer should be either centerlifted modulo p or reduced modulo p in the typical way.

9. [10 points] Suppose that P, Q are two points on an elliptic curve over $\mathbb{Z}/9719\mathbb{Z}$ (the number $p = 9719$ is prime). The order of the elliptic curve is a prime number q , and neither P nor Q

is \mathcal{O} . Alice has constructed the following two lists of points.

$$\begin{aligned} &[\mathcal{O}, P, 2P, \dots, 99P] \\ &[Q, Q \oplus 100P, Q \oplus 200P, \dots, Q \oplus 9900P] \end{aligned}$$

Prove that there must exist a common element between these two lists, and describe how finding this common element can be used to find an integer n such that $Q = nP$.

10. [10 points] Note (2024): This problem concerns a cryptosystem we did not discuss this semester. Suppose that the NTru cryptosystem (Table 7.4) is modified in the following ways.

- The single integer d in the parameters is replaced with three integers d_1, d_2, d_3 such that $d_1 > d_2 > d_3$. The requirement that $q > (6d + 1)p$ is removed.
- When Alice chooses \mathbf{f} , she chooses it from $\mathcal{T}(d_1 + 1, d_1)$.
- When Alice chooses \mathbf{g} , she chooses it from $\mathcal{T}(d_2, d_2)$.
- When Bob chooses \mathbf{r} , he chooses it from $\mathcal{T}(d_3, d_3)$.

Derive an inequality of the form “ $q > \dots$ ” (to replace $q > (6d + 1)p$ from the original version) in terms of d_1, d_2, d_3 (not all three of which must necessarily be used) and the other public parameters, such that decryption is guaranteed to succeed as long as this inequality holds.

11. [15 points] Samantha and Victor agree to the following digital signature scheme. The public parameters and key creation are identical to those of ECDSA. The verification procedure is different: to decide whether (s_1, s_2) is a valid signature for a document d , Victor computes

$$\begin{aligned} w_1 &\equiv s_1^{-1}d \pmod{q} \\ w_2 &\equiv s_1^{-1}s_2 \pmod{q}, \end{aligned}$$

then he check to see whether or not

$$x(w_1G \oplus w_2V) \% q = s_1.$$

If so, he regards (s_1, s_2) as a valid signature for d .

- (a) Describe a signing procedure that Samantha can follow to produce a valid signature on a given document d . The procedure should be randomized in such a way that it will generate different signatures if executed repeatedly on the same document.
 - (b) Describe a forgery procedure that Eve can follow to create a signature (s_1, s_2) and a document d such that (s_1, s_2) is a valid signature for d under this scheme. Note that Eve does not need to be able to choose d in advance. The procedure should be randomized in such a way that it can generate many different forgeries (on many different documents).
12. [15 points] Note (2024): this problem concerns a factoring procedure we did not discuss this semester. Suppose that n is an odd integer such that exactly $\frac{1}{32}$ of all units modulo n are squares (i.e. are congruent to some integer square modulo n). Alice wishes to factor n . Suppose that Alice chooses m distinct elements a_1, a_2, \dots, a_m of $\{1, 2, \dots, \frac{n-1}{2}\}$ at random.

- (a) Suppose that Alice discovers that $a_i^2 \equiv a_j^2 \pmod{n}$ for some $i \neq j$. Write a function `factor(n, ai, aj)` which returns a proper factor (i.e. a factor besides 1 or n) of n given the values a_i and a_j whose squares are congruent. For full credit, your function should perform no more than $\mathcal{O}(\log n)$ arithmetic operations.
- (b) Assuming that all m of these elements a_i are (distinct) *units* modulo n , prove that the probability that $a_i^2 \equiv a_j^2 \pmod{n}$ for *some* $i \neq j$ is at least $1 - e^{-32\binom{m}{2}/\phi(n)}$. You may assume without proof that $e^{-x} \geq 1 - x$ for all real numbers x . You may also assume that the values $a_i^2 \pmod{n}$ is equally likely to be any of the squares modulo n .
- (c) Suppose that the assumption in part (b) fails, and in fact one of the a_i is *not* a unit modulo n . This is a feature, not a bug: describe how Alice can quickly find a proper factor of n in this case, before she even looks for any collisions.

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	
<p style="text-align: center;"> Alice sends A to Bob $\xrightarrow{\hspace{2cm}}$ A B $\xleftarrow{\hspace{2cm}}$ Bob sends B to Alice </p>	
Further private computations	
Alice	Bob
Compute the number $B^a \pmod{p}$. The shared secret value is $B^a \equiv (g^b)^a \equiv g^{ab} \equiv (g^a)^b \equiv A^b \pmod{p}$.	Compute the number $A^b \pmod{p}$.

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 $1 \leq a \leq p - 1$. 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 $S^e \pmod{N}$ 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 $1 \leq a \leq p - 1$. Compute $A = g^a \pmod{p}$. Publish the verification key A .	
Signing	
Choose document $D \pmod{p}$. 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 $A^{S_1} S_2^{S_2} \pmod{p}$. Verify that it is equal to $g^D \pmod{p}$.

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 $1 \leq a \leq q - 1$. Compute $A = g^a \pmod{p}$. Publish the verification key A .	
Signing	
Choose document $D \pmod{q}$. Choose random element $1 < k < q$. Compute signature $S_1 \equiv (g^k \pmod{p}) \pmod{q}$ 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 $(g^{V_1} A^{V_2} \pmod{p}) \pmod{q} = S_1$.

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 . Computes the point $Q_A = n_AP$.	Chooses a secret integer n_B . Computes the point $Q_B = n_BP$.
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_AQ_B . The shared secret value is $n_AQ_B = n_A(n_BP) = n_B(n_AP) = n_BQ_A$.	Computes the point n_BQ_A .

Table 6.5: Diffie–Hellman key exchange using elliptic curves

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, $s_1 = x(eG) \bmod q$ 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 $v_2 \equiv s_1s_2^{-1} \pmod{q}$. Compute $v_1G + v_2V \in E(\mathbb{F}_p)$ and verify that $x(v_1G + v_2V) \bmod q = s_1$.	

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

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 $E(\mathbb{F}_p)$.	
Alice	Bob
Key Creation	
Chooses a secret multiplier n_A . Computes $Q_A = n_AP$. Publishes the public key Q_A .	
Encryption	
Chooses plaintext values m_1 and m_2 modulo p . Chooses a random number k . Computes $R = kP$. Computes $S = kQ_A$ and writes it as $S = (x_S, y_S)$. Sets $c_1 \equiv x_S m_1 \pmod{p}$ and $c_2 \equiv y_S m_2 \pmod{p}$. Sends ciphertext (R, c_1, c_2) to Alice.	
Decryption	
Computes $T = n_AR$ and writes it as $T = (x_T, y_T)$. 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 $m'_2 = m_2$.	

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

Table 7.1: A congruential public key cryptosystem

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 $f \in \mathcal{T}(d + 1, d)$ that is invertible in R_q and R_p . Choose private $g \in \mathcal{T}(d, d)$. Compute F_q , the inverse of f in R_q . Compute F_p , the inverse of f in R_p . Publish the public key $h = F_q \star g$.	
Encryption	
Choose plaintext $m \in R_p$. Choose a random $r \in \mathcal{T}(d, d)$. Use Alice's public key h to compute $e \equiv pr \star h + m \pmod{q}$. Send ciphertext e to Alice.	
Decryption	
Compute $f \star e \equiv pg \star r + f \star m \pmod{q}$. Center-lift to $a \in R$ and compute $m \equiv F_p \star a \pmod{p}$.	

Table 7.4: NTRUEncrypt: the NTRU public key cryptosystem