

- [6 points] Note (2024): the notation  $\log_5[13]_{23}$  wasn't used in our course, but it is explained in the parentheses. Use Shanks's "babystep-giantstep" algorithm to compute  $\log_5[13]_{23}$  (that is, find an integer  $x$  such that  $5^x \equiv 13 \pmod{23}$ ). Clearly label the two lists that you create and the common element between them. A multiplication table modulo 23 is provided at the back of the exam packet, for convenience.
- [6 points] Let  $p = 53$ ,  $q = 13$ ,  $g = 10$  be parameters for DSA (these satisfy the conditions in table 4.3). Suppose that Samantha has chosen the private signing key  $a = 7$ . Using  $k = 2$  as the ephemeral key, compute a DSA signature for the document  $D = 3$ . (Note: you do not need to calculate the public key  $A$  in order to solve this problem.)
- Integers  $p$  and  $q$  are both primes, exactly 42 bits in length. The numbers  $p - 1$  and  $q - 1$  factor into primes as follows.

$$\begin{aligned} p - 1 &= 2 \cdot 29 \cdot 353 \cdot 433 \cdot 601 \cdot 821 \\ q - 1 &= 2 \cdot 2199023249261 \end{aligned}$$

You may assume, without proof, that 2 is a primitive root modulo  $p$  and modulo  $q$ .

- [2 points] Explain briefly why discrete logarithms modulo  $p$  can be computed much more rapidly than discrete logarithms modulo  $q$  (be specific about which algorithms are involved; you do not need to describe the algorithms in detail).
- [4 points] Let  $N = pq$ . Suppose that Eve attempts to factor  $N$  by calling the following function (this is similar to the code provided on Problem Set 7, except that the initial value of  $a$  is chosen to be  $a = 2$ , rather than chosen at random, and it does not bother to check whether or not  $a$  is a unit initially).

```
def pollardWith2(N):
    a = 2
    j = 2
    while fractions.gcd(a-1,N) == 1:
        a = pow(a,j,N)
        j += 1
    return fractions.gcd(a-1,N)
```

What will be the return value of this function when called on  $N = pq$ ? How many times will the `while` loop iterate before returning this value?

- [3 points] Prove that if  $p$  is a prime number, and  $a$  is an integer such that  $a^2 \equiv 1 \pmod{p}$ , then either  $a \equiv 1 \pmod{p}$  or  $a \equiv -1 \pmod{p}$ .
  - [3 points] Suppose that  $p$  is a prime number,  $p - 1 = 2^k q$  for  $q$  an odd integer, and  $a$  is an integer with  $1 \leq a \leq N - 1$ . Deduce from part (a) that either  $a^q \equiv 1 \pmod{p}$  or one of the numbers  $a^q, a^{2q}, a^{4q}, \dots, a^{2^{k-1}q}$  is congruent to  $-1$  modulo  $p$ .
- Suppose that  $p, g$  are public parameters for Elgamal signatures (you may assume that  $g$  is a primitive root modulo  $p$ ), and that Samantha's public verification key is  $A$ . Samantha publishes a valid signature  $(S_1, S_2)$  for a document  $D$ , and Eve observes that  $S_1$  is exactly equal

to  $g$ . This might occur if Samantha is not choosing her ephemeral key sufficiently randomly.

- (a) [4 points] Assuming that  $\gcd(g, p - 1) = 1$ , write a function `extract(p, g, A, S1, S2, D)` that extracts Samantha's private signing key  $a$  from this information. You may assume that you have already implemented a function `ext_euclid(a, b)`, which returns a list  $[u, v, g]$  such that  $g = \gcd(a, b)$  and  $au + bv = g$ . Your code does not need to check that  $S_1 = g$ , or that  $\gcd(g, p - 1) = 1$ ; assume that it will only receive input meeting these conditions. Your code should be efficient enough to finish in a matter of seconds if all the arguments are 1024 bits long or shorter.
- (b) [2 points] Describe briefly how you would modify your code to work in the more general situation where  $\gcd(g, p - 1)$  is relatively small, but may not be equal to 1. You do not need to write a second program; just clearly describe the steps that you would take.

*Reference information. You may detach this sheet for easier use.*

**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	0
1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
2	0	2	4	6	8	10	12	14	16	18	20	22	1	3	5	7	9	11	13	15	17	19	21
3	0	3	6	9	12	15	18	21	1	4	7	10	13	16	19	22	2	5	8	11	14	17	20
4	0	4	8	12	16	20	1	5	9	13	17	21	2	6	10	14	18	22	3	7	11	15	19
5	0	5	10	15	20	2	7	12	17	22	4	9	14	19	1	6	11	16	21	3	8	13	18
6	0	6	12	18	1	7	13	19	2	8	14	20	3	9	15	21	4	10	16	22	5	11	17
7	0	7	14	21	5	12	19	3	10	17	1	8	15	22	6	13	20	4	11	18	2	9	16
8	0	8	16	1	9	17	2	10	18	3	11	19	4	12	20	5	13	21	6	14	22	7	15
9	0	9	18	4	13	22	8	17	3	12	21	7	16	2	11	20	6	15	1	10	19	5	14
10	0	10	20	7	17	4	14	1	11	21	8	18	5	15	2	12	22	9	19	6	16	3	13
11	0	11	22	10	21	9	20	8	19	7	18	6	17	5	16	4	15	3	14	2	13	1	12
12	0	12	1	13	2	14	3	15	4	16	5	17	6	18	7	19	8	20	9	21	10	22	11
13	0	13	3	16	6	19	9	22	12	2	15	5	18	8	21	11	1	14	4	17	7	20	10
14	0	14	5	19	10	1	15	6	20	11	2	16	7	21	12	3	17	8	22	13	4	18	9
15	0	15	7	22	14	6	21	13	5	20	12	4	19	11	3	18	10	2	17	9	1	16	8
16	0	16	9	2	18	11	4	20	13	6	22	15	8	1	17	10	3	19	12	5	21	14	7
17	0	17	11	5	22	16	10	4	21	15	9	3	20	14	8	2	19	13	7	1	18	12	6
18	0	18	13	8	3	21	16	11	6	1	19	14	9	4	22	17	12	7	2	20	15	10	5
19	0	19	15	11	7	3	22	18	14	10	6	2	21	17	13	9	5	1	20	16	12	8	4
20	0	20	17	14	11	8	5	2	22	19	16	13	10	7	4	1	21	18	15	12	9	6	3
21	0	21	19	17	15	13	11	9	7	5	3	1	22	20	18	16	14	12	10	8	6	4	2
22	0	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

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 $\xrightarrow{\hspace{2cm}}$ $A$ $B \xleftarrow{\hspace{2cm}}$ Bob sends $B$ to Alice	
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)