CS111 Spring'24 ASSIGNMENT 2

Names: Luis Barrios, Andy Payan

Problem 1:

Prove the following statement: if p and $p+2$ are twin primes and $p > 5$, then $p^3 + 3p^2 - p \equiv 23 \pmod{10}$.

Hint: The product of any k consecutive integers is divisible by k.
Alternatively, think what are possible values of p rem 2 and p rem 5.

Solution 1:

(1)

$$p^3 + 3p^2 - p \equiv 23 \pmod{10}$$
$$p^3 + 3p^2 - p \equiv 3 \pmod{10}$$
$$p^3 + 3p^2 - p - 3 \equiv 0 \pmod{10}$$

By using synthetic division, we can obtain the following terms:

$$(x-1)(x^2+4x+3) \equiv 0 \pmod{10}$$

$$(x+1)(x+3)(x-1) \equiv 0 \pmod{10}$$
$$(p+1)(p+3)(p-1) \equiv 0 \pmod{10}$$

Multiply by p and p+2 to make it consecutive.

$$(p)(p+1)(p+2)(p+3)(p-1) \equiv 0 \pmod{10}$$

(mod 10) can easily be broken up into (mod 2) and (mod 5). We now need to test the cases.

1. (mod 2)

$$\begin{aligned} p(0) &= (0)(0+1)(0+2)(0+3)(0-1) \\ &= (0)(1)(2)(3)(-1) = 0 \pmod{2}\checkmark \\ p(1) &= (1)(1+1)(1+2)(1+3)(1-1) \\ &= (1)(2)(3)(4)(0) = 0 \pmod{2}\checkmark \end{aligned}$$

Both equal to 0 due to p and p-1.

2. (mod 5)

As we already tested, we can conclude that p(0) and p(1)...

$$p(0) = 0$$

$$p(1) = 0$$

$$p(2) = (2)(2+1)(2+2)(2+3)(2-1) = 0 \pmod{5}$$
$$p(2) = (2)(3)(4)(5)(1) = 0 \pmod{5}$$
$$120 = 0 \pmod{5}$$
$$-120 \pmod{5} = 0\checkmark$$
$$p(3) = (3)(3+1)(3+2)(3+3)(3-1) = 0 \pmod{5}$$
$$p(3) = (3)(4)(5)(6)(2) = 0 \pmod{5}$$

$$720 = 0 \pmod 5$$

$$-720 \pmod 5 = 0\checkmark$$

$$p(4) = (4)(4+1)(4+2)(4+3)(4-1) = 0 \pmod 5$$

$$p(4) = (4)(5)(6)(7)(3) = 0 \pmod 5$$

$$2520 = 0 \pmod 5$$

$$-2520 \pmod 5 = 0\checkmark$$

With this, we have proved that $p^3 + 3p^2 \cdot p = 23 \pmod{10}$.

Problem 2:

Alice's RSA public key is $P = (e, n) = (7, 4891)$. Bob sends Alice the message by encoding it as follows. First he assigns numbers to characters: A is 7, B is 8, ..., Z is 32, a blank is 33, quotation marks: 34, a comma: 35, a period: 36, an apostrophe: 37. Then he uses RSA to encode each number separately.

Bob's encoded message is:

```
1619  4049  2672  1427   845   391   570  2246   849   391  2672  1855  4537  1427
 391  1427  3780  2072  2361  2072   845  1855   725  1427  2796   391  1855  3780
3780   391  4049  2672  2072  4462  2672   391  2072  2488   391  2072  2361  1689
2246  2488  2488  2072  3804  3780  1427  1545   391   725  2672  1427   845   391
4049  2672  1855   725  1427  4537  1427  4648   391  4648  1427  2361  1855  2072
 845  2488  1545   391  2672  2246  4049  1427  4537  1427  4648   391  2072  2361
1689  4648  2246  3804  1855  3804  3780  1427  1545   391  2361   849  2488   725
 391  3804  1427   391   725  2672  1427   391   725  4648   849   725  2672  4522
1619
```

Decode Bob's message. Notice that you only know Alice's public key, but don't know the private key. So you need to "break" RSA to decrypt Bob's message. For the solution, you need to provide the following:

(a) Describe step by step how you arrived at the solution: show how to find p and q, $\phi(n)$ and d.

(b) Show your work for one integer in the message (C = 4049): the expression, the decrypted integer, the character that it is mapped to.

(c) To decode the remaining numbers, you need to write a program in C++ (see below), test it in Gradescope, and append the code to HW 2, Problem 2 solutions.

(d) Give the decoded message (in integers).

(e) Give Bob's message in plaintext. What does it mean and who said it?

For part (c). Your program should :

(i) Take three integers, e, n (the public key for RSA), and m (the number of characters in the message) as input to your program. Next, input the ciphertext.

(ii) Test whether the public key is valid. If not, output a single line "Public key is not valid!" and quit the program.

(iv) If the public key is valid, decode the message.

(v) Output p and q, $\phi(n)$ and d.

(vi) On a new line, output the decoded message in integers.

(vii) On a new line, output the decoded message in English. The characters should be all uppercase. You can assume that the numbers will be assigned to characters according to the mapping above.

More information and specifications will be provided separately.

Upload your code to Gradescope to test. There will be 15-16 (open and hidden) test cases. Your score for the RSA code will be based on the score that you received in Gradescope. If you have any questions, post them on Slack.

Solution 2:
(a)

$$P = (e, n) = (7, 4891)$$

n = 4891, which is made out of 2 distinct prime numbers p = 67 and p = 73 (the only divisors other than 1 and itself). To find $\phi(n)$, you do $(p-1)(q-1)$ which makes $66 \cdot 72 = 4752$. We can verify with $gcd(e, \phi(n)) = gcd(7, 4752)$.

$$4752 = 678 \cdot 7 + 1$$

$$7 = 7 \cdot 1 + 0$$

$$gcd(4752, 7) = 1$$

Now to find d, $d = e^{-1}(\pmod{\phi(n)})$, so $d = 7^{-1}(\pmod{4752})$. We'll use the list of multiples method to obtain...

$$7 \cdot d = 4752 \cdot k + 1$$

$$7 \cdot 679 = 4752 \cdot k + 1 = 4753$$

$$d = 679$$

(b)

$$c = 4049; c^d \pmod{n} = 4049^{679} \pmod{4891}$$

$$4049^{2 \cdot 339} \cdot 4049 \pmod{4891}$$

$$16394401^{339} \cdot 4049 \pmod{4891}$$

$$4660^{338} \cdot 4660 \cdot 4049 \pmod{4891}$$

$$4660^{169 \cdot 2} \cdot 18868340 \pmod{4891}$$

$$21715600^{169} \cdot 3753 \pmod{4891}$$

$$4451^{168} \cdot 4451 \cdot 3753 \pmod{4891}$$

$$4451^{84 \cdot 2} \cdot 16704603 \pmod{4891}$$

$$19811401^{84} \cdot 1838 \pmod{4891}$$

$$2851^{2 \cdot 42} \cdot 1838 \pmod{4891}$$

$$8128201^{42} \cdot 1838 \pmod{4891}$$

$$4250^{2 \cdot 21} \cdot 1838 \pmod{4891}$$

$$18062500^{21} \cdot 1838 \pmod{4891}$$

$$\begin{gathered}
37^{20} \cdot 1838 \cdot 37 \pmod{4891} \\
37^{2 \cdot 10} \cdot 4423 \pmod{4891} \\
1369^{2 \cdot 5} \cdot 4423 \pmod{4891} \\
1874161^{5} \cdot 4423 \pmod{4891} \\
908^{4} \cdot 908 \cdot 4423 \pmod{4891} \\
908^{4} \cdot 4016084 \pmod{4891} \\
824464^{2} \cdot 573 \pmod{4891} \\
2776^{2} \cdot 573 \pmod{4891} \\
7706176 \cdot 573 \pmod{4891} \\
2851 \cdot 573 \pmod{4891} \\
1633623 \pmod{4891} = \\
29 \pmod{4891} \\
29 = W
\end{gathered}$$

(c) Code from gradescope:

```
#include <iostream>
#include <vector>
#include <cmath>

using namespace std;

bool isAcceptable(int num){
    for (int i = 2; i < num; ++i){
        if (num % i == 0){
            return false;
        }
    }
    return true;
}

//from lecture slides
int validPublicKey(int a, int b){
    if (a == b){
        return a;
    }
    if (a < b){
        swap(a, b);
    }
    return validPublicKey(a - b, b);
}

void decryptMessage(int value){
```

```
if (value == 7){
    cout << "A";
}
else if (value == 8){
    cout << "B";
}
else if (value == 9){
    cout << "C";
}
else if (value == 10){
    cout << "D";
}
else if (value == 11){
    cout << "E";
}
else if (value == 12){
    cout << "F";
}
else if (value == 13){
    cout << "G";
}
else if (value == 14){
    cout << "H";
}
else if (value == 15){
    cout << "I";
}
else if (value == 16){
    cout << "J";
}
else if (value == 17){
    cout << "K";
}
else if (value == 18){
    cout << "L";
}
else if (value == 19){
    cout << "M";
}
else if (value == 20){
    cout << "N";
}
else if (value == 21){
    cout << "O";
}
else if (value == 22){
    cout << "P";
}
else if (value == 23){
    cout << "Q";
}
```

```
    else if (value == 24){
        cout << "R";
    }
    else if (value == 25){
        cout << "S";
    }
    else if (value == 26){
        cout << "T";
    }
    else if (value == 27){
        cout << "U";
    }
    else if (value == 28){
        cout << "V";
    }
    else if (value == 29){
        cout << "W";
    }
    else if (value == 30){
        cout << "X";
    }
    else if (value == 31){
        cout << "Y";
    }
    else if (value == 32){
        cout << "Z";
    }
    else if (value == 33){
        cout << " ";
    }
    else if (value == 34){
        cout << "\"";
    }
    else if (value == 35){
        cout << ",";
    }
    else if (value == 36){
        cout << ".";
    }
    else if (value == 37){
        cout << "’";
    }
}

int main() {
    int c = 0;
    int n = 0;
    int m = 0;
    int p = 0;
    int q = 0;
    int phi = 0;
int e = 0;
int d = 1;
int exp = 1;
int multi = 1;
int k = 1;
int encryptedInteger = 0;
vector <int> encryptedMessage;

cin >> e >> n >> m;
if(e <= 0 || e >= n){
    cout << "Public key is not valid!";
    return 1;
}
for (int i = 0; i < m; ++i){
    cin >> encryptedInteger;
    encryptedMessage.push_back(encryptedInteger);
}

for (int i = 2; i <= n; ++i){
    if (n % i == 0){
        p = i;
        q = n / i;
        break;
    }
}

if (p == q || !isAcceptable(p) || !isAcceptable(q)){
    cout << "Public key is not valid!";
    return 1;
}

phi = (p-1)*(q-1);
if (phi == 0){
    cout << "Public key is not valid!";
    return 1;
}
if(validPublicKey(e, phi) != 1){
    cout << "Public key is not valid!";
    return 1;
}

for(d = 1; (e*d != phi*k+1); ++d){
    if (e*d > phi*k+1){
        ++k;
    }
}

cout << p << " " << q << " " << phi << " " << d << endl;

for(int i = 0; i < m; ++i){
    c = encryptedMessage.at(i);
```

```
        multi = 1;
        exp = d;
        while (exp != 0){
            if (exp % 2 != 0){
                multi = multi * c;
                --exp;
            }
            else {
                exp = exp / 2;
                c = pow(c, 2);
            }

            if (multi >= n){
                multi = multi % n;
            }
            if (c >= n){
                c = c % n;
            }
        }
        encryptedMessage.at(i) = multi;
    }

    for(int i = 0; i < m; ++i){
        cout << encryptedMessage.at(i) << " ";
    }

    cout << endl;

    for (int i = 0; i < m; ++i){
        decryptMessage(encryptedMessage.at(i));
    }
    return 0;
}
```

(d) encoded message in integers:

34 29 14 11 20 33 31 21 27 33 14 7 28 11 33 11 18 15 19 15 20 7 26 11 10 33 7 18 18 33 29 14 15 9 14 33 15
25 33 15 19 22 21 25 25 15 8 18 11 35 33 26 14 11 20 33 29 14 7 26 11 28 11 24 33 24 11 19 7 15 20 25 35 33
14 21 29 11 28 11 24 33 15 19 22 24 21 8 7 8 18 11 35 33 19 27 25 26 33 8 11 33 26 14 11 33 26 24 27 26 14 36 34

(e) encoded message in characters: ”WHEN YOU HAVE ELIMINATED ALL WHICH IS IMPOSSIBLE, THEN WHATEVER REMAINS, HOWEVER IMPROBABLE, MUST BE THE TRUTH.” The quote comes from Sir Arthur Conan Doyle, which is actually a famous quote from the Sherlock Holmes books. What the author meant by this quote is that when you rule out everything that is deemed impossible by deduction, whatever cases are leftover must be the truth, regardless of how improbable it is. This is a good example of Sherlock Holmes's logical thinking of when he solved his mysteries.

Problem 3:

(a) Compute $7^{1529} \pmod{51}$. Show your work.

(b) Compute $9^{-1} \pmod{19}$ by listing the multiples. Show your work.

(c) Compute $9^{-1} \pmod{19}$ using Fermat's Little Theorem. Show your work.

(d) Compute $9^{-11} \pmod{19}$ using Fermat's Little Theorem. Show your work.

(e) Find an integer x, $0 \leq x \leq 18$, that satisfies the following congruence: $9x + 13 \equiv 10 \pmod{19}$. Show your work. You should not use brute force approach.

Solution 3:

(a) $7^{1529} \pmod{51}$

$$7^{1528+1} \pmod{51}$$
$$7^{2 \cdot 764} \cdot 7 \pmod{51}$$
$$7^{2 \cdot 2 \cdot 382} \cdot 7 \pmod{51}$$
$$2401^{382} \cdot 7 \pmod{51}$$
$$4^{382} \cdot 7 \pmod{51}$$
$$4^{2 \cdot 191} \cdot 7 \pmod{51}$$
$$16^{191} \cdot 7 \pmod{51}$$
$$16^{190+1} \cdot 7 \pmod{51}$$
$$16^{190} \cdot 112 \pmod{51}$$
$$16^{190} \cdot 10 \pmod{51}$$
$$16^{2 \cdot 95} \cdot 10 \pmod{51}$$
$$256^{95} \cdot 10 \pmod{51}$$
$$1^{95} \cdot 10 \pmod{51} = 10 \pmod{51}$$
$$= 10$$

(b) $9^{-1} \pmod{19}$
so we have the formula of the form $d \cdot 9 \equiv 19k + 1$. We will now list out the multiples of 9 and 19.

multiples of 9:

$$(9, 18, 27, 36, 45, 54, 63, 72, 81, 90, 99, 108, 117, 126, 135, 144, 153)$$

multiples of 19:

$$(19, 38, 57, 76, 95, 114, 133, 152)$$

17 multiples of 9 gives us 153, while 8 multiples of 19 gives us 152. This means that when k = 8, d = 17.

$$9 \cdot (17) \equiv 19 \cdot (8) + 1$$
$$153 = 152 + 1$$
$$153 = 153$$

From this, we can say that $9^{-1} \pmod{19} = 17 \pmod{19}$, which is just 17.

(c) $9^{-1} \pmod{19}$ using Fermat's Little Theorem
19 > 2 and is also a prime number, and 9 is not a multiple of 19, so we can use Fermat's theorem.

$$9^{18} \equiv 1 \pmod{19}$$

$$9^{-1} \pmod{19} \equiv 9^{-1} \cdot 9^{18} \pmod{19}$$
$$9^{17} \pmod{19}$$
$$3^{2 \cdot 17} \pmod{19}$$
$$3^{34} \pmod{19}$$
$$3^{2 \cdot 4 \cdot 4 + 2} \pmod{19}$$
$$3^{2 \cdot 4 \cdot 4} \cdot 9 \pmod{19}$$
$$81^{4 \cdot 2} \cdot 9 \pmod{19}$$
$$5^{4 \cdot 2} \cdot 9 \pmod{19}$$
$$25^{4} \cdot 9 \pmod{19}$$
$$6^{4} \cdot 9 \pmod{19}$$
$$1296 \cdot 9 \pmod{19}$$
$$4 \cdot 9 \pmod{19}$$
$$36 \pmod{19}$$
$$17 \pmod{19} = 17$$

(d) $9^{-11} \pmod{19}$ using Fermat's Little Theorem
19 > 2 and is also a prime number, and 9 is not a multiple of 19, so we can use Fermat's theorem.

$$9^{18} = 1 \pmod{19}$$

$$9^{-11} \pmod{19} \equiv 9^{-11} \cdot 9^{18} \pmod{19}$$
$$9^{7} \pmod{19}$$
$$3^{14} \pmod{19}$$
$$3^{3 \cdot 2 \cdot 2 + 2} \pmod{19}$$
$$3^{3 \cdot 4} \cdot 9 \pmod{19}$$
$$8^{4} \cdot 9 \pmod{19}$$
$$2^{3 \cdot 4} \cdot 9 \pmod{19}$$
$$2^{6 \cdot 2} \cdot 9 \pmod{19}$$
$$64^{2} \cdot 9 \pmod{19}$$
$$7^{2} \cdot 9 \pmod{19}$$
$$49 \cdot 9 \pmod{19}$$
$$11 \cdot 9 \pmod{19}$$
$$99 \pmod{19}$$
$$4 \pmod{19} = 4$$

(e) Find an integer x, $0 \leq x \leq 18$, that satisfies the following congruence: $9x + 13 \equiv 10 \pmod{19}$.

$$9x + 13 \equiv 10 \pmod{19}$$

$$9x \equiv 10 - 13 \pmod{19} =$$

$$9x \equiv -3 \pmod{19}$$

$$9x \cdot 9^{-1} = -3 \cdot 9^{-1} \pmod{19} =$$

$$x = -3 \cdot 9^{-1} \pmod{19} =$$

$$\text{We already know from part (b) } 9^{-1} \pmod{19}$$

$$x = -3 \cdot 17 \pmod{19} =$$

$$x = -51 \pmod{19} =$$

$$-57 < -51 =$$

$$6 \pmod{19}$$

Academic integrity declaration. Academic integrity declaration. This assignment was a collaboration between Luis Barrios and Andy Payan. The only resources we had used throughout this assignment include TA office hours (with Biqian, Ezekiel, and Alice) and reviewing slides and notes given from lecture and discussion. We also used www.goodreads.com to research the given quote from Sir Arthur Conan Doyle to correctly refer to who made this message. For the code, we mainly just took a look at a website discussing vectors and how they could be used again for review. The autograder assisted in leading us to the correct path.

Submission. To submit the homework, you need to upload the pdf and cpp files to Gradescope. If you submit with a partner, you need to put two names on the assignment and submit it as a group assignment.