RSA, POLYNOMAILS, SECRET SHARING, ERASURE ERRORS 4

COMPUTER SCIENCE MENTORS 70

September 25-29, 2017

1 RSA

1.1 Questions

1. **How does RSA work?**

   a. Alice wants to send Bob a message $m = 5$ using his public key ($n = 26$, $e = 11$). What cipher text E(m) will Alice send?

Solution:

$$\begin{aligned}
5^1 &= 5 \mod 26 \\
5^2 &= 5 \mod 26 \\
&= -1 \mod 26 \\
5^4 &= (-1)^2 \mod 26 \\
&= 1 \mod 26 \\
5^8 &= 1 \mod 26 \\
5^{11} &= 5^8 * 5^2 * 5^1 \mod 26 \\
&= 1 * -1 * 5 \mod 26 \\
&= -5 \mod 26 \\
&= 21 \mod 26
\end{aligned}$$

b. What is the value of d (Bobs private key) in this scheme? Note that traditional RSA schemes use much larger prime numbers, so its harder to break n down into its prime factors than it is in this problem.

Solution: $n = 26 \rightarrow$ because $26 = pq$ and $p \neq a * q$ for all a within integers, $p = 13$, $q = 2$

$$d = e^{-1} \mod (13-1)(2-1)$$

$$d = 11^{-1} \mod 12$$

$$d = 11$$

2. In RSA, if Alice wants to send a confidential message to Bob, she uses Bobs public key to encode it. Then Bob uses his private key to decode the message. Suppose that Bob chose $N = 77$. And then Bob chose $e = 3$ so his public key is (3, 77). And then Bob chose $d = 26$ so his private key is (26, 77).
   Will this work for encoding and decoding messages? If not, where did Bob first go wrong in the above sequence of steps and what is the consequence of that error? If it does work, then show that it works.

   > **Solution:** e should be co-prime to $(p-1)(q-1)$.
   > $e = 3$ is not co-prime to $(7-1)(11-1) = 60$, so this is incorrect, since therefore e does not have an inverse $\mod 60$.

3. **Coin tosses over text messages**
   You and one of your friends want to get your hands on the new gadget thats coming out. One of you has to wait in line overnight so that you have a chance to get the gadgets while they last. In order to decide who this person should be, you both agree to toss a coin. But you wont meet each other until the day of the actual sale and you have to settle this coin toss over text messages (using your old gadgets). Obviously neither of you trusts the other person to simply do the coin toss and report the results.

   How can you use RSA to help fix the problem?

   > **Solution:** If there was a way for me to make my choice (i.e. toss the coin) without revealing to my friend what the result was before s/he makes her/his decision, then we would be in good shape. RSA enables us to do just that. One can commit to a choice without revealing what that choice really is. So here is how we proceed:
   >
   > 1. One issue we need to consider is that if "heads" and "tails" are the only things we're encrypting and decrypting, then given some public key and an encrypted message it's easy to simply encrypt "heads" and encrypt "tails" and check which one matches the received encrypted message.
   > 2. I select a public key (N,e) and a private key d. I toss a coin. If I get "heads", I choose some random word that begins with an H. If I get "tails", I choose some random word that begins with a T. I do this so that my friend can't "reverse engineer" the encrypted message to figure out what my result was. Instead of sending the result to my friend, I first encrypt my word using the public key (N,e). Then I send my friend the public key along with the encrypted message.
   > 3. My friend is supposedly (read the next part for why the word supposedly is used) unable to see what the result of the coin toss was and therefore cannot

cheat. So s/he makes her/his choice (what HEADS and TAILS mean) and sends it to me, using the same technique (some word starting with H for heads and T for tails).

4. Once I have successfully received the result, I reveal the result of the coin toss by sending my friend my word in plain text (i.e. with no encryption). My friend can now verify that I have not cheated (i.e. I have not changed the result) by encrypting the result using the public key I have given her/him and making sure it was the same as the encrypted message I send her/him. Note that RSA encryption and decryption are both bijections, therefore if I know the encrypted version of two messages are the same, then those two messages must be the same.

Note that I cannot cheat here, because I commit to the result of the coin toss before I know my friends choice. Commitment is a very useful primitive (used in many places in cryptography) that enables a party to convincingly commit to a choice without revealing it until they choose to reveal it. The party should not be able to change their mind after the commitment which is what the scheme guarantees.

2 Secret Sharing

2.1 Questions

1. Suppose the Oral Exam questions are created by 2 TAs and 3 Readers. The answers are all encrypted and we know that:
   (a) Together, both TAs should be able to access the answers
   (b) All 3 Readers can also access the answers
   (c) One TA and one Reader should also be able to do the same

   Design a secret sharing scheme to make this work.

   Solution: Use a 2 degree polynomial which requires at least 3 shares to recover the polynomial. Generate a total of 7 shares, give each Reader a share, and each TA 2 shares. Then, all possible combinations will have at least 3 shares to recover the answer key. Basically the point of this problem is to assign different weights to different classes of people. If we give one share to everyone, then 2 Readers can also recover the secret and the scheme is broken.

2. An officer stored an important letter in her safe. In case she is killed in battle, she decides to share the password with her troops. Everyone knows there are 3 spies among the troops, but no one knows who they are except for the three spies themselves.The 3 spies can coordinate with each other and they will either lie and make people not able to open the safe, or will open the safe themselves if they can. Therefore, the officer would like a scheme to share the password that satisfies the following conditions:

   1. When M of them get together, they are guaranteed to be able to open the safe even if they have spies among them.
   2. The 3 spies must not be able to open the safe all by themselves.

   Please help the officer to design a scheme to share her password. What is the scheme? What is the smallest M? Show your work and argue why your scheme works and any smaller M couldnt work.

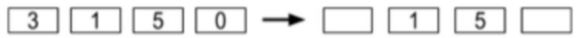
Solution: The key insight is to realize that both polynomial-based secret-sharing and polynomial-based error correction work on the basis of evaluating an underlying polynomial at many points and then trying to recover that polynomial. Hence they can be easily combined. Suppose the password is s. The officer can construct a polynomial $P(x)$ such that $s = P(0)$ and share $(i, P(i))$ to the i-th person in her troops. Then the problem is: what should the degree of $P(x)$ be and what is the smallest M? First, the degree of polynomial d should not be less than 3. It is because when $d < 3$, the 3 spies can decide the polynomial $P(x)$ uniquely. Thus, n will be at least 4 symbols. Lets choose a polynomial $P(x)$ of degree 3 such that $s = P(0)$. We now view the 3 spies as 3 general errors. Then the smallest $M = 10$ since n is at least 4 symbols and we have $k = 3$ general errors, leading us to a codeword of $4 + 2 \cdot 3 = 10$ symbols (or people in our case). Even though the 3 spies are among the 10 people and try to lie on their numbers, the 10 people can still be able to correct the $k = 3$ general errors by the Berlekamp-Welch algorithm and find the correct $P(x)$.

Alternative solution: Another valid approach is making $P(x)$ of degree $M1$ and adding 6 public points to deal with 3 general errors from the spies. In other words, in addition to their own point $(i, P(i))$, everyone also knows the values of 6 more points, $(t+1, P(t+1)), (t+2, P(t+2)), \ldots, (t+6, P(t+6))$, where t is the number of the troops. The spies have access to total of $3 + 6 = 9$ points so the degree $M - 1$ must be at least 9 to prevent the spies from opening the safe by themselves. Therefore, the minimum M is 10.

3 Erasure Errors

3.1 Introduction

We want to send n packets and we know that k packets could get lost.



How many more points does Alice need to send to account for k possible errors? __

Solution: k

What degree will the resulting polynomial be? ___

Solution: $n-1$

How large should q be if Alice is sending n packets with k erasure errors, where each packet has b bits?

Solution: Modulus should be larger than $n+k$ and larger than 2^b and be prime

What would happen if Alice instead send $n+k-1$? Why will Bob be unable to recover the message?

Solution: Bob will receive $n-1$ distinct points and needs to reconstruct a polynomial of degree $n-1$. By Fact #3 this is impossible. There are q polynomials of at most degree $n-1$ in $GF(q)$ that go through the $n-1$ points that Alice sent.

3.2 Questions

1. Suppose $A=1$, $B=2$, $C=3$, $D=4$, and $E=5$. Assume we want to send a message of length 3. Recover the lost part of the message, or explain why it can not be done.

1. C_AA

   Solution: $P(0) = 3, P(2) = 1, P(3) = 1$. Once we interpolate the polynomial over $\mod 7$, as E is 5, we get $5x^2 + 3x + 3$. Now, once we evaluate this at 1, we get 4. So, in the end, its CDAA.

2. CE_ _

   Solution: Impossible. In order to get the original degree 2 polynomial, we need at least $3 > 2$ points.

2. Suppose we want to send n packets, and we know $p = 20\%$ of the packets will be erased. How many extra packets should we send? What happens if p increases (say to 90%)?

   Solution: We want to have $(1-p)*(n+k) = n$, where k is the number of additional packets we send. Solving for k, we get $\frac{n}{1-p} - n$. When p is large, we have to send many times the number of original packets. (fraction packets not erased)*(how many packets are sent) = (number packets in original message)

4 Polynomials

4.1 Introduction

1. There is a unique polynomial of degree $n - 1$ such that $P(i) = m_i$ for each packet $m_1, \ldots, m_n$
2. To account for errors we send $c_1 = P(1), \ldots, c_{n+j} = P(n + j)$
3. If polynomial $P(x)$ has degree $n - 1$ then we can uniquely reconstruct it from any n distinct points.
4. If a polynomial $P(x)$ has degree $n - 1$ then it can be uniquely described by its n coefficients

4.2 Questions

1. Define the sequence of polynomials by $P_0(x) = x+12$, $P_1(x) = x^2-5x+5$ and $P_n(x) = xP_{n-2}(x) - P_{n-1}(x)$. (For instance, $P_2(x) = 17x - 5$ and $P_3(x) = x^3 - 5x^2 - 12x + 5$.)

(a) Show that $P_n(7) \equiv 0 \pmod{19}$ for every $n \in N$.

Solution:

(a) Prove using strong induction.

Base Case There are two base cases because each polynomial is defined in terms of the two previous ones except for P_0 and P_1.

$$P_0(7) \equiv 7 + 12 \equiv 19 \equiv 0 \pmod{19}$$
$$P_1(7) \equiv 7^2 - 5 \cdot 7 + 5 \equiv 49 - 35 + 5 \equiv 19 \equiv 0 \pmod{19}$$

Inductive Hypothesis Assume $P_n(7) \equiv 0 \pmod{19}$ for every $n \leq k$.

Inductive Step Using the definition of P_{k+1}, we have that

$$\begin{aligned} P_{k+1}(7) &\equiv xP_{k-1}(7) - P_k(7) \pmod{19} \\ &\equiv x \cdot 0 - 0 \pmod{19} \\ &\equiv 0 \pmod{19} \end{aligned}$$

Therefore, $P_n(7) \equiv 0 \pmod{19}$ for all natural numbers n.

(b) Show that, for every prime q, if $P_{2013}(x) \not\equiv 0 \pmod{q}$, then $P_{2013}(x)$ has at most 2013 roots modulo q.

Solution: This question asks to prove that, for all prime numbers q, if $P_{2013}(x)$ is a non-zero polynomial $\pmod{q}$, then $P_{2013}(x)$ has at most 2013 roots $\pmod{q}$.

The proof of Property 1 of polynomials (a polynomial of degree d can have at most d roots) still works in the finite field $GF(q)$. Therefore we need only show that P_{2013} has degree at most 2013. We prove that $\deg(P_n) \leq n$ for $n > 1$ by strong induction.

Base cases There are 4:

$$\begin{aligned} \deg(P_0) &= \deg(x + 12) = 1 \\ \deg(P_1) &= \deg(x^2 - 5x + 5) = 2 \\ \deg(P_2) &= \deg(xP_0(x) - P_1(x)) \leq 2 \\ \deg(P_3) &= \deg(xP_1(x) - P_2(x)) \leq 3 \end{aligned}$$

Inductive Hypothesis Assume $\deg(P_n) \leq n$ for all $2 \leq n \leq k$.

Inductive Step Then

$$
\begin{aligned}
\deg(P_{k+1}(x))& \\
&\leq \max\{\deg(xP_{k-1}(x)), \deg(P_k(x))\} \\
&= \max\{1 + \deg(P_{k-1}(x)), \deg(P_k(x))\} \\
&\leq \max\{1 + k - 1, k\} \\
&\leq k \\
&\leq k + 1
\end{aligned}
$$

5 Secret Sharing

5.1 Questions

1. Suppose the Oral Exam questions are created by 2 TAs and 3 Readers. The answers are all encrypted and we know that:
   (a) Together, both TAs should be able to access the answers
   (b) All 3 Readers can also access the answers
   (c) One TA and one Reader should also be able to do the same

   Design a secret sharing scheme to make this work.

   Solution: Use a 2 degree polynomial which requires at least 3 shares to recover the polynomial. Generate a total of 7 shares, give each Reader a share, and each TA 2 shares. Then, all possible combinations will have at least 3 shares to recover the answer key. Basically the point of this problem is to assign different weights to different classes of people. If we give one share to everyone, then 2 Readers can also recover the secret and the scheme is broken.

2. The United Nations (for the purposes of this question) consists of n countries, each having k representatives. A vault in the United Nations can be opened with a secret combination s. The vault should only be opened in one of two situations. First, it can be opened if all n countries in the UN help. Second, it can be opened if at least m countries get together with the Secretary General of the UN.

   Propose a scheme that gives private information to the Secretary General and n countries so that s can only be recovered under either one of the two specified conditions.

   Solution: Have two schemes, one for the first condition and one for the second.

   For the first condition: just one polynomial of degree n - 1 or less would do, where each country gets one point. The polynomial evaluated at 0 would give the secret.

   For the second condition: one polynomial is created of degree m - 1 and a point is given to each country. Another polynomial of degree 1 is created, where one point is given to the secretary general and the second point can be constructed from the first polynomial if m or more of the countries come together. With these two points, we have a unique 1-degree polynomial, which could give the secret evaluated at 0.

6 Erasure Errors

6.1 Introduction

We want to send n packets and we know that k packets could get lost.

$$\boxed{3}\ \boxed{1}\ \boxed{5}\ \boxed{0} \longrightarrow \boxed{}\ \boxed{1}\ \boxed{5}\ \boxed{}$$

How many more points does Alice need to send to account for k possible errors? __

Solution: k

What degree will the resulting polynomial be? ___

Solution: $n - 1$

How large should q be if Alice is sending n packets with k erasure errors, where each packet has b bits?

Solution: Modulus should be larger than $n + k$ and larger than 2^b and be prime

What would happen if Alice instead send $n+k-1$? Why will Bob be unable to recover the message?

Solution: Bob will receive $n - 1$ distinct points and needs to reconstruct a polynomial of degree $n - 1$. By Fact #3 this is impossible. There are q polynomials of at most degree $n - 1$ in $GF(q)$ that go through the $n - 1$ points that Alice sent.

6.2 Questions

1. Suppose $A = 1$, $B = 2$, $C = 3$, $D = 4$, and $E = 5$. Assume we want to send a message of length 3. Recover the lost part of the message, or explain why it can not be done.

1. C_AA

> **Solution:** $P(0) = 3, P(2) = 1, P(3) = 1$. Once we interpolate the polynomial over $\mod 7$, as E is 5, we get $5x^2 + 3x + 3$. Now, once we evaluate this at 1, we get 4. So, in the end, its CDAA.

2. CE_ _

> **Solution:** Impossible. In order to get the original degree 2 polynomial, we need at least $3 > 2$ points.

2. Suppose we want to send n packets, and we know $p = 20\%$ of the packets will be erased. How many extra packets should we send? What happens if p increases (say to 90%)?

> **Solution:** We want to have $(1-p)*(n+k) = n$, where k is the number of additional packets we send. Solving for k, we get $\frac{n}{1-p} - n$. When p is large, we have to send many times the number of original packets. (fraction packets not erased)*(how many packets are sent) = (number packets in original message)