Cryptography and Number Theory
Mid-Term writeup

Benji Altman

March 21, 2018

1 RSA, problems with small encryption exponents

This section will be handled backwards, in that part two of the question will be solved then part one. This is as part two is the general case of part one, so we will solve the general case then show it in action using part one as an example.

1.1 General case

Statement of Problem:

> Suppose that n people have RSA moduli $m_1, \dots m_n$ and they all have encryption exponent n. Explain how a message (an integer less than each m_i) encrypted and sent to all n people can be decrypted from the n ciphertexts without factoring the m_i. This is why small encryption exponents should not be used.

Now it can be safely assumed that all m_i are coprime. The reasoning for this is as follows. These are all just RSA moduli computed normally. If there was a significant chance that I would get two RSA moduli that share a divisor then an adversary, who is trying to crack my RSA modulus, would have a good shot if they keep generating their own RSA moduli and checking to see if either of the factors in their generated moduli are a factor in my modulus. So any reasonable assumption that RSA is secure must imply that the RSA moduli are coprime.

Now because they are coprime we may use Sunzi's theorem to compute c such that $x^n \equiv c (\text{mod } \prod_{i=1}^{n} m_i)$. Notice that $\min(\{m_i \,|\, 0 < i \leq n\})^n < \prod_{i=1}^{n} m_i$ and that $x < \min(\{m_i \,|\, 0 < i \leq n\})$, thus

$$x^n < \min(\{m_i \,|\, 0 < i \leq n\})^n < \prod_{i=1}^{n} m_i$$

so we know x^n as an integer without any modulus and thus may calculate x.

1.2 Application

Statement of Problem:

> Suppose that Bob, Claire, and Dave all have RSA public keys with encryption exponent 3. Bob's modulus is 161, Claire's is 209, and Dave's is 221. Alice sends the same message, an integer x less than 161, to Bob, Claire, and Dave. You intercept all three ciphertexts. Bob's ciphertext is 29, Claire's is 144, and Dave's is 196. That is, you know $x^3 \equiv 29 \mod 161$, $x^3 \equiv 144 \mod 209$, and $x^3 \equiv 196 \mod 221$. Solve for x without factoring any of the moduli.

Now we will employ what we stated in the last section. We first use sunzi's theorem to solve for $x^3 (\text{mod } 161 \cdot 209 \cdot 221)$. This is easiest to be done via a simple program so I'll spare the reader tedious arithmetic; the solution is $x^3 \equiv 1000000 \mod 7436429$. Now $x < 161$ so $x^3 < 161^3 < 7436429 = 161 \cdot 209 \cdot 221$, so $x^3 = 1000000$, and thus $x = 100$.

2 Linear algebra euclidean algorithm

2.1 Solutions

(a) If the i^{th} quotient is q_i, then the matrix will be

$$M_i = \begin{bmatrix} 0 & 1 \\ 1 & -q_i \end{bmatrix}$$

and it is easy to see it's determinant is $0 \cdot (-q_i) - 1 \cdot 1 = -1$.

To solve for it's inverse we concatenate the identity to M_i and row reduce M_i until it's the identity and the other side of the concatenated matrix will have the inverse. This is all rather basic linear algebra, for notational purposes, while solving fo the inverse AB will be the matrix A concatenated with B, not multiplication.

$$M_i I = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & -q_i & 1 & 0 \end{bmatrix}$$
$$I {M_i}^{-1} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & q_i \end{bmatrix}$$
$${M_i}^{-1} = \begin{bmatrix} 0 & 1 \\ 1 & q_i \end{bmatrix}$$

(b) Here the arithmetic is quite straight forward I will forgo showing it all, however note that it is made easy as if you solve $\begin{pmatrix} 0 & 1 \\ 1 & a \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 \\ 1 & b \end{pmatrix} = \begin{pmatrix} 1 & b \\ a & ab \end{pmatrix}$ then we already know $M_1 M_2$ and $M_3 M_4$ and solving for $M_1 M_2 M_3 M_4$ is simply one final multiplication, so the final solution is

$$M_1 M_2 M_3 M_4 = \begin{pmatrix} 26 & -57 \\ -83 & 182 \end{pmatrix}$$

(c) We have been given this sort of identity that $(a, b)M_1 M_2 \ldots M_r = (1, 0)$. So $M_1 M_2 \ldots M_r$ is a 2×2 matrix, thus we may write it as

$$M_1 M_2 \ldots M_r = \begin{pmatrix} \alpha & \gamma \\ \beta & \delta \end{pmatrix}$$

for some $\alpha, \beta, \gamma, \delta \in \mathbb{Z}$. It follows that

$$(a, b) \begin{pmatrix} \alpha & \gamma \\ \beta & \delta \end{pmatrix} = (1, 0)$$

so we may break apart the multiplication and see $a\alpha + b\beta = 1$. This reveals that $a\alpha \equiv 1 \mod b$ and $b\beta \equiv 1 \mod a$ so $a^{-1} \mod b \equiv \alpha$ and $b^{-1} \mod a \equiv \beta$.

3 Vigenère extensions

3.1 Large Language approach

The Friedman attack works on the fundamental idea that not all letters are equally common in texts. This leads naturally to the question, can we create a language where each letter it equally common in appearance? It would be a tremendous undertaking to create an entirely new language, where each letter is equally common, just so that we can defeat the Friedman attack. The approach I take here will instead add a sort of preprocessing step to our native language to map it into a larger language with equal distribution of letters.

Now to stay language agnostic and for simplicity, we will think of our letters as being the numbers $0, 1, \ldots, n-1$. Now each letter will have some rate of occurrence. For any letter ℓ in a language will have a probability of $p_\ell \in (0,1]$[1] of occurrence where $\sum\limits_{n=0}^{n-1} p_\ell = 1$. Now find some natural number, $n \neq 0$, such that for all letters ℓ, np_ℓ is within some specified ϵ of a positive integer, let us call this integer z_ℓ. Now we will let there be a set $\phi(\ell)$ for each letter ℓ with the requirement that $|\phi(\ell)| = z_\ell$ and for letters $a \neq b$, $\phi(a) \cap \phi(b) = \varnothing$.

Now if we have some string M with letters $M_0, M_1, \ldots$ we will then let M' consist of letters $M'_0, M'_1, \ldots$. where M'_n is randomly chosen from the set $\phi(M_n)$.

Now with a small ϵ chosen this will make it so that are letters occur with very nearly a uniform distribution in M'. We may now do a vigenère cipher on this new text. Notice that the key must be consist of letters chosen from the large language $\bigcup \phi(\ell)$.

Now depending on how ϕ is chosen then this may still be weak to modified Friedman attacks. For example if $\phi(0) = \{0, 1, \ldots, z_0 - 1\}, \phi(1) = \{z_0, z_0 + 1, \ldots, z_0 + z_1 - 1\}, \ldots$ then counting coincidences as having the same letter you said it was a near by number, then you would be able to break this. If however you choose ϕ in a more random fashion, this will not be possible.[2]

3.2 Diffusion

One of the nice things about a vigenère cipher is that it can encrypt one character at a time without knowledge of characters around it, all that is needed is it's position in the text. This lends itself nicely to parallelization. We try and throw out this advantage to be able to mix data from surrounding parts of our message based on the key for the encryption. Before we even address how to do this one might worry about letters on the edge of the message (the first and last characters), to deal with this we will consider the message to be surrounded by an infinite number of a chosen letter, so M_{-1} and $M_{|M|}$ may be used without causing any issue.

For terminology's sake let us refer to our plain-text message as M and our encrypted message as E.

One possible implementation for this diffusion would be to have $E_i = \begin{cases} M_i + K_i + M_{i-1} & K_i \text{ is even} \\ M_i + K_i + M_{i-2} & K_i \text{ is odd} \end{cases}$
Now the Friedman attack is much weaker, as in order to get a coincidence to mean something you must have i and k such that $i \equiv k(\text{mod } K.\text{Length})$, $M_i = M_k$, and depending on if K_i is even or odd $M_{i-1} = M_{k-1}$ or $M_{i-2} = M_{k-2}$. This however may still not cause a perfect uniform distribution, however with even this simple diffusion we will already have significantly decreased the chances to detect the key length.

4 Mersenne Primes

(a) If r is a prime factor of $2^q - 1$ then $2^q - 1 \equiv 0 \mod r$, thus $2^q \equiv 1 \mod r$.

(b) It follows from $2^q \equiv 1 \mod r$ that the order of 2 in modulus r must divide q. To prove this, let o be the order of 2 in mod r. This means that there is no positive integer, $k < o$ such that $2^k \equiv 1 \mod r$, then if $n \geq 1$ is not divisible by o, there must be some a, b such that $bo + a = n$ with $0 < a < o$ and thus

$$\begin{aligned} 2^n &= 2^{bo+a} \\ &= (2^o)^b \cdot 2^a \\ &\equiv 1^b \cdot 2^a \mod r \\ &\equiv 2^a \mod r \not\equiv 1 \end{aligned}$$

Now additionally we are given that q is prime, therefore q must be the order of 2 in mod r.

(c) By Fermat's little theorem we know that for any prime p, $a^{p-1} \equiv 1 \mod p$, and thus $2^{r-1} \equiv 1 \mod r$. By our proof from above we then know that $r - 1$ must be divisible by the order of 2 in mod r. We know the

[1] If a letter were to have a probability of 0 then we would simply omit it from the language, however a probability of 1 is allowed if it is the only letter in a language.

[2] A possible schema for choosing ϕ is to simply choose it randomly and publicly release your scheme, there is no need to change ϕ between usages so a common ϕ may be chosen once and used everywhere, as long as the randomizing of it's order is secure.

order to be q, so q must divide $r-1$. This means, that any prime divisor of 2^{q-1}, r, must have the property that $r-1$ is divisible by q.

d Now $127 = 2^7 - 1$, so to check for divisors we only need to check prime factors in the form $7n+1$.

- $7 \cdot 1 + 1 = 8$ is not prime
- $7 \cdot 2 + 1 = 15$ is not prime
- $7 \cdot 3 + 1 = 22$ is not prime
- $7 \cdot 4 + 1 = 29 < \sqrt{127}$, thus if 29 was a prime factor of 127 it must not be the greatest prime of 127. We know that there are no possible prime factors of 127 less than 29, thus 127 has no prime factors and must be prime.