# Problem Set 5–Security of RSA

CSCI3381-Cryptography

Due October 24, 2014

As usual, you can get up to 100 points on this assignment, but you get to pick and choose.

For the programming problems, you are to implement a variety of attacks on RSA. The ciphertexts and moduli are posted on the website. These attacks succeed because of poor implementations of the algorithm, due either to failure to randomly pad the plaintext prior to encryption, or problems in the key generation phase.

As we discussed in class, RSA encryption is usually used to encrypt keys for symmetric ciphers, but I do try to make the solutions a little more amusing to read. The RSA moduli used in this problem have approximately 1024 bits = 128 bytes. A single block allows texts of length at most 128 characters; considerably less for the first and last of the programming problems. So I can't give you those long literary texts you had on the preceding problems. Instead, the solutions to all but the last problem are song lyrics, brought to you by Cole Porter, David Bowie, Frank Zappa, Weird Al Yankovic, and the Beatles (I'm sorry, I'm old! I don't *know* any new songs.)

## 1 Written Problems

1. *Square roots modulo $n$; 10 points.*

*(a)* $9493^2 \bmod 11413 = 1$. Check it if you don't believe me! Why does this prove that 11413 is composite?

*(b)* Use the equation in (a) to factor 11413. It is possible to do this with a computer by trial division because the numbers are relatively small, but I want you to show the steps of a computation that can be carried out by hand, or that could be done if the numbers given in (a) were many, many times larger.

*(c)* Similarly, carry out a calculation that will find the other square roots of 1 modulo 11413. Of course, two of these square roots are 1 and 11412; find another one.

2. *Easy factorization if $p, q$ are too close together: 20 points* This problem forms the theoretical basis for one of the computer problems below. (You do not need to prove the result here in order to tackle the computer problem, you can just accept it and apply the given algorithm.)

Suppose Alice decides to use primes of more than 200 decimal digits (*i.e.,* greater than $10^{200}$) to generate her RSA modulus. She picks a prime $p$ at random, and then chooses $q$ to agree with $p$ in all but the last 100 digits. Her reasoning is that $p$ and $q$

have effectively been randomly sampled from a set of size $10^{100}$, and that it will be just as hard for an adversary to factor $N = pq$ as it would be to factor the product of two random 100-digit primes.

But Alice is in for a nasty surprise: If we have a factorization $n = pq$, then we can write the two primes as $A - x$ and $A + x$, where $A = (p + q)/2$ is the average of $p$ and $q$. Thus

$$N = pq = (A - x)(A + x) = A^2 - x^2,$$

so

$$x = \sqrt{A^2 - N}.$$

If we knew what $A$ was, then since computing exact square roots is easy, we could determine $x$ and hence the factors $p$ and $q$. But how can we determine $A$ without knowing $p$ and $q$ in the first place? Prove that in the circumstances described above, $A = \lceil \sqrt{N} \rceil$, *the smallest integer greater than* $\sqrt{N}$. Thus we can factor $N$ by solving the easy problem of computing the square root, which gives us $A$, and then $x$ and the prime factors $A \pm x$.

HINT. The 'circumstances described above' are that $p$ and $q$ differ by less than $\sqrt[4]{N}$. You need to prove that in this case

$$\sqrt{N} = \sqrt{pq} < A = \frac{p + q}{2} < \sqrt{N} + 1.$$

It's easiest to do this if you look at what you get by squaring all parts of the inequality.

# 2 Computer Problems–20 points each

Each of these problems contains an RSA ciphertext encrypted with a 1024-bit modulus, but in each case it is possible to recover the plaintext by attacking some weakness in the implementation. RSA decryption gives the plaintext as an integer, but you should convert your solution to an ASCII string; when you do this, Problems 1-5 give readable (even singable!) answers, and Problem 6 is a very short message appropriate for this class.

As you work these problems, think about which of these are vulnerable because of the absence of padding, and which because of poor practice in generating keys. Which of the attacks recover the decryption key, and which recover only the plaintext?

1. *Short message, small exponent* The ciphertext for Problem 1 was obtained by encrypting ASCII plaintext with exponent = 3 and the given modulus. Viewed as an integer, however, the plaintext message $M$ satisfies $M^3 < N$, where $N$ is the modulus. (You have to take an exact cube root.)

2. *Small exponent.* A single message was encrypted using three different RSA moduli, all with exponent $e = 3$. The three ciphertexts and moduli are given in the accompanying web page. Find the plaintext. (You have to apply the Chinese Remainder Theorem.)

3. *Common Modulus* Everyone in Alice's office is given an RSA public-private key pair with the same modulus. Alice's encryption exponent is 5, and she sees that Alicia's

encryption exponent is 3. She intercepts an encrypted message from Roberto to Alicia and is able to decrypt it, and so are you. (Alice can use her own private key to factor the modulus.)

4. *Common Factor* Alex, on the other hand, uses different moduli to generate the RSA keys used by her staff, but admits that she has a 'lucky prime' that she uses to generate all these keys. Eva uses this information, and the moduli of two of the staff members, and decrypts messages sent to each of them. Find the plaintexts. (Eva can easily factor both moduli.)

5. *Prime Factors Close Together.* Alessandra is too clever to fall into the preceding two traps. She generates an RSA key by creating two primes that agree in their 77 most significant decimal digits, but choosing the remaining 75 digits at random and taking the closest primes to the result. Cleverer Evelyn uses the public RSA information to decrypt the intercepted ciphertext. What is the plaintext? (Evelyn uses the attack described in the second written problem to factor the modulus.)

6. *Small Message.* The plaintext is a very brief ASCII text. Your are given the modulus, encryption exponent, and ciphertext. Viewed as an integer, the plaintext is more than one billion, so it would take quite a long time to try out every possibility in a slowpoke language like Python. But this integer factors into two integers each less than four million. Find the plaintext. (Use a meet-in-the-middle attack: If $c$ is the ciphertext, and $m = m_1 m_2$ the message, then $cm_1^{-e} \equiv m_2^e \pmod{n}$.)