Lecture 9: The RSA Public-Key Cryptosystem, I

William Stein

Math 124 Harvard University **Fall 2001**

Key Ideas:

- Creating an RSA public key
- Encrypting and decrypting messages
- Breaking RSA and factoring

Contents

1 How RSA works

1.1 One-way Functions

The *fundamental idea* behind RSA is to try to construct a "one-way function", i.e., an "encryption" function

$$E : X \to X$$

such that it is easy for Nikita, say, to compute E^{-1}, but very hard for anybody else to compute E^{-1}.

1.2 How Nikita Makes an RSA Public Key

Here is how Nikita makes a one-way function E:

1. Nikita picks two large primes p and q, and lets $n = pq$.

2. It is easy for Nikita to then compute

$$\varphi(n) = \varphi(p) \cdot \varphi(q) = (p-1) \cdot (q-1).$$

3. Nikita next chooses a "random" integer e with

$$1 < e < \varphi(n) \text{ and } \gcd(e, \varphi(n)) = 1.$$

4. Finally, Nikita uses the algorithm from Lecture 7 to find a solution d to the equation

$$ex \equiv 1 \pmod{\varphi(n)}.$$

The Encoding Function:
Nikita defines a function $E : \mathbb{Z}/n\mathbb{Z} \to \mathbb{Z}/n\mathbb{Z}$

$$E(x) = x^e.$$

(Recall that $\mathbb{Z}/n\mathbb{Z} = \{0, 1, \ldots, n-1\}$ with addition and multiplication modulo n.) Then anybody can compute E fairly quickly using the repeated-squaring algorithm from Lecture 7.

Nikita's **public key** is the pair of integers (n, e), which is just enough information for people to easily compute E. Nikita knows a number d such that $ed \equiv 1 \pmod{\varphi(n)}$, so, as we will see below, she can quickly compute E^{-1}.

Now Michael or even The Collective can send Nikita a message whenever they want, even if Nikita is asleep. They look up how to compute E and compute E(their message).

1.3 Sending Nikita an Encrypted Message

Encode your message as a sequence of numbers modulo n (see Section 2):

$$m_1, \ldots, m_r \in \mathbb{Z}/n\mathbb{Z}.$$

Send

$$E(m_1), \ldots, E(m_r)$$

to Nikita. (Recall that $E(m) = m^e$.)

1.4 How Nikita Decrypts a Message

When Nikita receives an $E(m_i)$, she finds m_i as follows:

$$m_i = E^{-1}(E(m_i)) = E(m_i)^d = (m_i^e)^d = m_i.$$

The following proposition proves that the last equality holds.

Proposition 1.1. *Let n be a square-free integer and let $d, e \in \mathbb{N}$ such that $p-1 \mid de-1$ for each prime $p \mid n$. Then $a^{de} \equiv a \pmod{n}$ for all $a \in \mathbb{Z}$.*

Proof. Since $n \mid a^{de} - a$ if and only if $p \mid a^{de} - a$ for each prime divisor of p, it suffices to prove that $a^{de} \equiv a \pmod{p}$ for each prime divisor p of n. If $\gcd(a,p) \neq 0$, then $a \equiv 0 \pmod{p}$, so $a^{de} \equiv a \pmod{p}$. If $\gcd(a,p) = 1$, then Fermat's Little Theorem asserts that $a^{p-1} \equiv 1 \pmod{p}$. Since $p-1 \mid de-1$, we have $a^{de-1} \equiv 1 \pmod{p}$ as well. Multiplying both sides by a shows that $a^{de} \equiv a \pmod{p}$. $\square$

2 Encoding a Phrase in a Number

Think of a sequence of letters and spaces as a number in base 27. Let a single-space correspond to 0, the letter A to 1, B to 2, ..., Z to 26. Thus, e.g., "HARVARD" denotes a number written in base 27. The corresponding number written in decimal is 1808939906:

$$\text{HARVARD} \quad \leftrightarrow \quad 8+27\cdot 1+27^2\cdot 18+27^3\cdot 22+27^4\cdot 1+27^5\cdot 18+27^6\cdot 4 = 1808939906$$

To recover the digits of the number, repeatedly divide by 27:

$$\begin{array}{lcrcrl} 1808939906 & = & 66997774\cdot 27 & + & 8 & \text{H} \\ 66997774 & = & 2481399\cdot 27 & + & 1 & \text{A} \end{array}$$

and so on.

2.1 How Many Letters Can a Number "Hold"?

If $27^k < n$, then k letters can be encoded in a number $< n$. Put another way,

$$k < \log(n)/\log(27) = \log_{27}(n).$$

3 Examples

3.1 A Small Example

So the arithmetic is easy to follow, we use small primes p and q and encrypt the single letter "X".

1. Choose p and q: Let $p = 17$, $q = 19$, so $n = pq = 323$.

2. Compute $\varphi(n)$:

   $$\varphi(n) = \varphi(p \cdot q) = \varphi(p) \cdot \varphi(q) = (p-1)(q-1) = pq - p - q + 1 = 323 - 17 - 19 + 1 = 288.$$

3. Randomly choose an $e \in \mathbb{Z}/323\mathbb{Z}$: We choose $e = 95$.

4. Solve
   $$95x \equiv 1 \pmod{288}.$$
   Using the GCD algorithm, we find that $d = 191$ solves the equation.

The public key is $(323, 95)$. So $E : \mathbb{Z}/323\mathbb{Z} \to \mathbb{Z}/323\mathbb{Z}$ is defined by

$$E(x) = x^{95}.$$

Next, we encrypt the letter "X". It is encoded as the number 24, since X is the 24th letter of the alphabet. We have

$$E(24) = 24^{95} = 294 \in \mathbb{Z}/323\mathbb{Z}.$$

To decrypt, we compute E^{-1}:

$$E^{-1}(294) = 294^{191} = 24 \in \mathbb{Z}/323\mathbb{Z}.$$

3.2 A Bigger Example in PARI

```
? p=nextprime(random(10^30))
%3 = 738873402423833494183027176953
? q=nextprime(random(10^25))
%4 = 3787776806865662882378273
? n=p*q
%5 = 2798687536910915970127263606347911460948554197853542169
? e=random(n)
%6 = 1483959194866204179348536010284716655442139024915720699
? phin=(p-1)*(q-1)
%7 = 2798687536910915970127262867470721260308194351943986944
? while(gcd(e,phin)!=1,e=e+1)
? e
%8 = 1483959194866204179348536010284716655442139024915720699
? d = lift(Mod(e,phin)^(-1));
%9 = 2113367928496305469541348387088632973457802358781610803
? (e*d)%phin
%10 = 1
? log(n)/log(27)
%11 = 38.03851667699197952338510248
```

We can encode single blocks of up to 38 letters. Let's encode "HARVARD":

```
? m=8+27*1+27^2*18+27^3*22+27^4*1+27^5*18+27^6*4
%12 = 1808939906
? E(x)=lift(Mod(x,n)^e)
? D(x)=lift(Mod(x,n)^d)
? secret_message = E(m)
%14 = 625425724974078486559370130768554070421628674916144724
? D(secret_message)
%15 = 1808939906
```

The following complete PARI program automates the whole process, though it is a little clumsy. Call this file `rsa.gp`. It uses { and } so that functions can be extended over more than one line.

```
/* rsa.gp ------------------------------------------------ */
{alphabet=[" ","A","B","C","D","E","F","G","H","I","J","K","L","M",
           "N","O","P","Q","R","S","T","U","V","W","X","Y","Z"];
}
{letter_to_number(l,
     n)=
     for(n=1,27,if(alphabet[n]==l,return(n-1)));
     error("invalid input.")
}
{number_to_message(n,
     s="")=
     while(n>0, s = concat(s,alphabet[n%27+1]); n = n \ 27);
     return(s)
}
{message_to_number(w,
     i,n=0)=
     for(i=1,length(w), n = n + 27^(i-1)*letter_to_number(w[i]));
     return(n);
}
{make_rsa_key(len,
     p,q,n,e,d)=
     p = nextprime(random(10^(len/2)));
     q = nextprime(random(10^(len/2+3)));
     n = p*q; phin = (p-1)*(q-1);
     e = random(phin);
     while(gcd(e,phin)!=1,e=e+1);
     d = lift(Mod(e,phin)^(-1));
     return([n,e,d]);
}
encrypt(message, n, e) = lift(Mod(message_to_number(message),n)^e);
decrypt(secret, n, d)  = number_to_message(lift(Mod(secret,n)^d));
/* rsa.gp ------------------------------------------------ */
```

Here is an example that uses the above little program.

```
? \r rsa
? setrand(1)   \\ default random number seed is 1!
? rsa=make_rsa_key(20)   \\ returns [n, e, d]
%2 = [89050154117716728145939, 33735260657253161660951,
      49244741969289756040079]
? n = rsa[1]; e = rsa[2]; d = rsa[3];
? public_key = [n,e]
%3 = [89050154117716728145939, 33735260657253161660951]
? msg = ["H", "A", "R", "V", "A", "R", "D"];   \\ clumsy!!!
? secret = encrypt(msg,n,e)
%36 = 75524965161901413275866
? decrypt(secret, n, d)
%37 = "HARVARD"
```

4 A Connection Between Breaking RSA and Factoring Integers

Nikita's public key is (n, e). If we compute the factorization of $n = pq$, then we can compute $\varphi(n)$ and hence deduce her secret decoder number d.
It is no easier to $\varphi(n)$ than to factor n:
Suppose $n = pq$. Given $\varphi(n)$, it is very easy to compute p and q. We have

$$\varphi(n) = (p-1)(q-1) = pq - (p+q) + 1,$$

so we know $pq = n$ and $p + q = n + 1 - \varphi(n)$. Thus we know the polynomial

$$x^2 - (p+q)x + pq = (x-p)(x-q)$$

whose roots are p and q.

There is also a more complicated "probabilistic algorithm" to find p and q given the secret decoding number d. I might describe it in the next lecture.