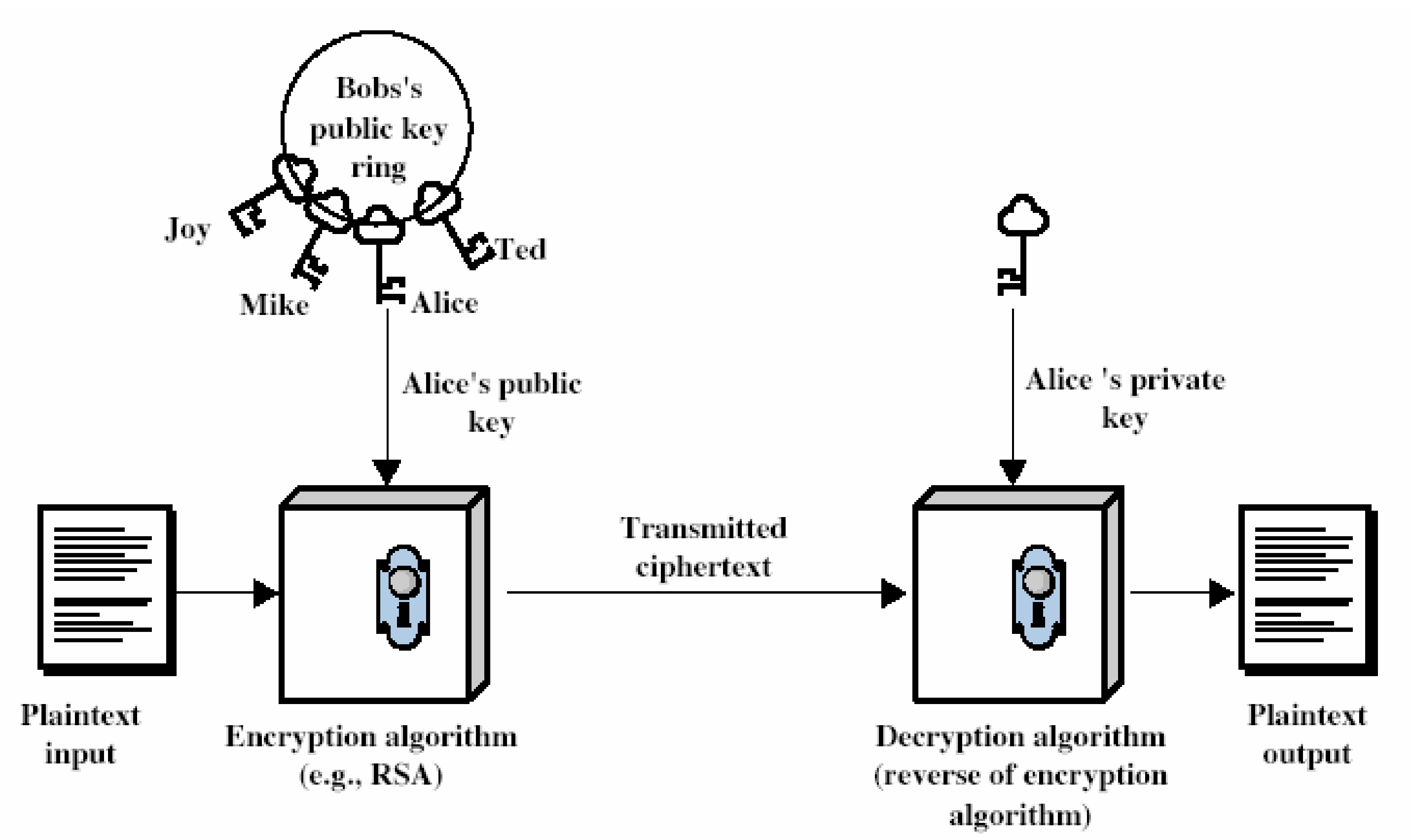
Public Key Cryptography: RSA and Lots of Number Theory

WIRELESS INFORMATION NETWORK LABORATORY

Public vs. Private-Key Cryptography

- We have just discussed traditional symmetric cryptography:
  - Uses a single key shared between sender and receiver
- Asymmetric (public key) cryptography was introduced by Diffie & Hellman and is a dramatically different approach to cryptography:
  - Two keys are used: a public and a private key
  - Alice generates a public and a private key
  - The Public Key is given to anyone who would like to securely communicate with Alice
  - Alice keeps the Private key "private" and she may decrypt messages encrypted with the public key
- Asymmetric cryptography is NOT a replacement of symmetric cryptography
  - There are many scenarios where symmetric is much better than asymmetric (such as bulk data encryption)

Public-Key Cryptography



Pros and Cons of Public Key Cryptography

- There are several advantages to public key cryptography:
  - **Asymmetry allows for easier key distribution:** We do not need a trusted third party (KDC) to distribute keys
  - **Asymmetry provides proof of origin:** The secret key is something that only one entity knows.
  - **Can be extended to form chains of trust:** Public Key Certificate frameworks provide a natural way to model trust.
- There are several disadvantages to public key cryptography:
  - **Computation Burden:** By its very nature, public key cryptography is not as fast or computationally efficient as symmetric cryptography.
  - **Communication Overhead:** PKIs typically require significant communication overhead (frequent and large messages).

Public-Key Characteristics

- Public-Key algorithms rely on two keys with the characteristics that it is:
  - computationally infeasible to find decryption key knowing only algorithm & encryption key
  - computationally easy to en/decrypt messages when the relevant (en/decrypt) key is known
  - either of the two related keys can be used for encryption, with the other used for decryption (in some schemes)

Public-Key Cryptosystems

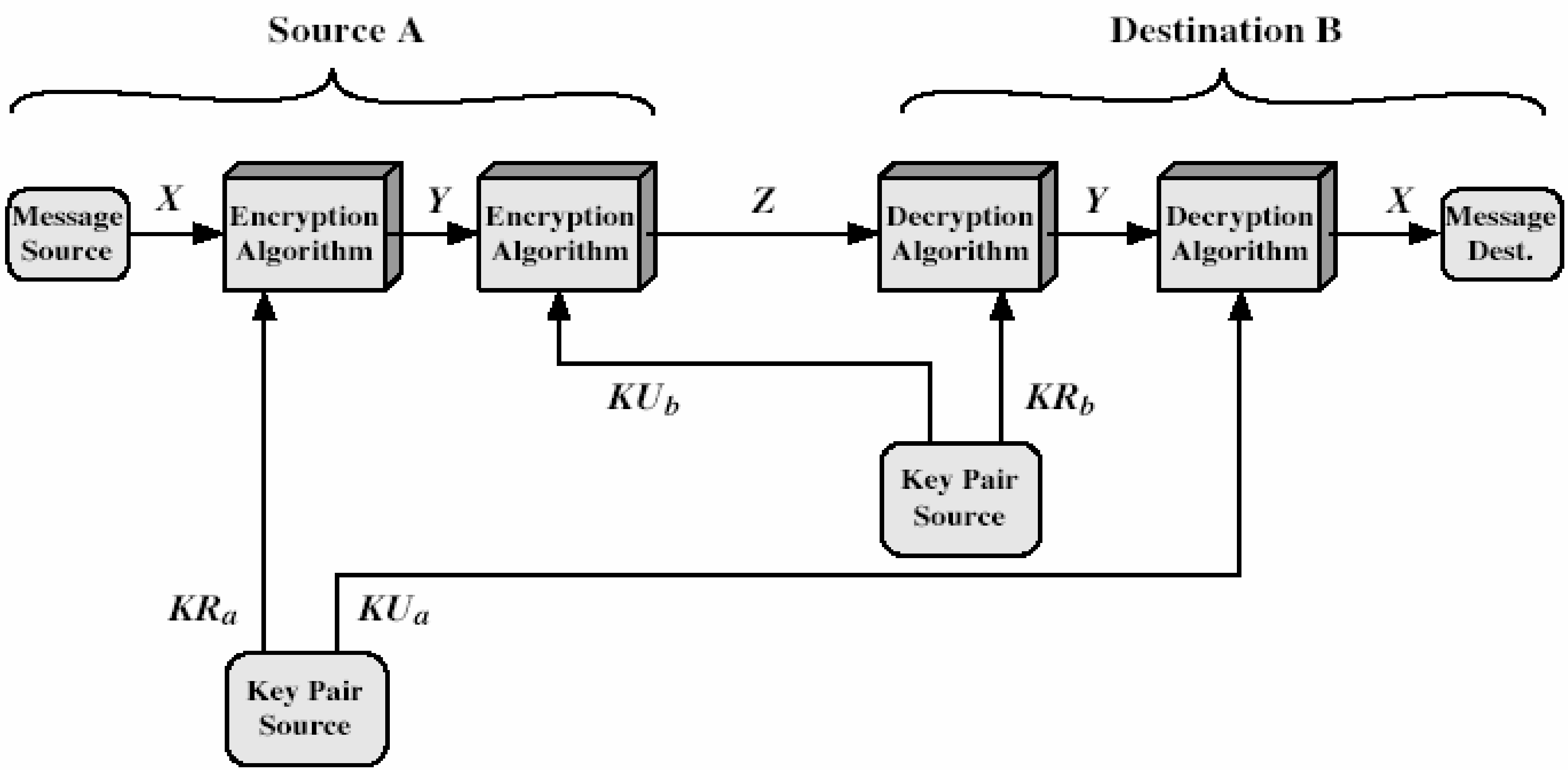


Figure 9.4 Public-Key Cryptosystem: Secrecy and Authentication

RSA

- The RSA algorithm is the most popular public key scheme and was invented by Rivest, Shamir & Adleman of MIT in 1977
- Based on exponentiation in a finite field over integers modulo a prime
  - Exponentiation takes $O((\log n)^3)$ operations (easy)
  - Exponentiation is accomplished through repeated squaring
  - Uses large integer operations
- Requires finding large primes
- The security of RSA is based on the (believed) intractability of the factoring of the product of two large primes
  - Difficulty of factoring is based upon the size of the factors
  - Factorization of RSA composite takes $O(e^{\log n \log \log n})$ operations (hard)

Setup of RSA

- Alice wishes to generate a public key and a private key

1. She first generates two large random primes p and q
2. She computes the composite n=pq, and the Euler Phi function

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

3. She chooses a random encryption exponent e such that

$$\gcd(e, \varphi(n)) = 1$$

4. She finds the decryption exponent by: $ed = 1 \mod \varphi(n)$

- Public key is {e,N}
- Private key is {d,p,q}

Encryption with RSA

- Suppose Bob wishes to encrypt a message M and send it to Alice.
- He acquires her public key {e,n}
- He computes the ciphertext:

$$c = m^e \bmod n$$

- Alice can decrypt by using her private key {d,p,q} via

$$m = c^d \bmod n$$

- Requirement: We are doing operations modulo n, so message must be smaller than m

RSA Example

1. Select primes: p=885320963 & q=238855417
2. Compute $n = pq$ =211463707796206571
3. Compute phi(n)=(p–1)(q-1)
4. Select e : gcd(e,phi(n))=1; choose e=9007
5. Determine d: de=1 mod phi(n) and d < phi(n).
   Value is d=116402471153538991

Example encryption

1. Suppose m=30120
2. Ciphertext $c=m^e \bmod n = (30120)^{9007}$=113535859035722866
3. Reconstruct plaintext: $m=c^d \bmod n$
   $= 113535859035722866^{116402471153538991}$
   = 30120

Why RSA Works

- OK… so we've covered what RSA is… now lets look at why it works, and how to make it work in practice.
- There are several key observations, most built from number theory, that we will need.
- We will cover the following:
  - Euler's Theorem
  - How to Calculate Inverses
  - Modular Exponentiation
  - Finding Primes
- We will briefly discuss the security of RSA today.
- More on the security of RSA will come in a lecture on attacks and cryptanalysis.

Euler's Theorem

- **Fermat's Little Theorem:** If p is a prime and p does not divide a, then $a^{p-1} \equiv 1 \pmod{p}$.
- Euler's Theorem is a generalization of Fermat's Little Theorem
- **Euler's Theorem:** If gcd(a, n) = 1, then $a^{\phi(n)} \equiv 1 \pmod{n}$.
- What is $\phi(n)$? It's the amount of numbers between 1 and n that relatively prime to n.
- Example: $\phi(pq) = (p-1)(q-1)$
- **Euler/Fermat Example:** Compute $2^{43210} \pmod{101}$.
- *Solution:* From Fermat's theorem, we know that $2^{100} \equiv 1 \pmod{101}$. Therefore, $2^{43210} \equiv (2^{100})^{432} 2^{10} \equiv 1^{432} 2^{10} \equiv 1024 \equiv 14 \pmod{101}$.

Why Does Euler Make RSA Work?

- **Basic Principle:** Let a, n, x, y be integers with $n \geq 1$ and gcd(a, n) = 1. If $x \equiv y \pmod{\varphi(n)}$, then $a^x \equiv a^y \pmod{n}$. In other words, if you want to work mod n, you should work mod φ(n) in the exponent.
- In RSA, we choose $ed \equiv 1 \pmod{\varphi(n)}$, so

$$m^{ed} \equiv m^1 = m \pmod{n}.$$

Or, more explicitly $ed \equiv 1 \pmod{\varphi(n)}$ means

$$ed = 1 + k\,\varphi(n)$$

So

$$m^{ed} \equiv m^{1+k\varphi(n)} = m \pmod{n}$$

WINLAB
WIRELESS INFORMATION NETWORK LABORATORY

How to Calculate Inverses for RSA?

- We need to calculate e and d such that $ed \equiv 1 \pmod{\varphi(n)}$. How do we do this?
- Step 1: Choose a random e such that $\gcd(e, \varphi(n)) = 1$.

Why?

How?

- Step 2: Now, find d.

The <u>wrong</u> way:

$$
\begin{aligned}
e \cdot 1 &\equiv e \bmod \varphi(n) \\
e \cdot 2 &\equiv 2e \bmod \varphi(n) \\
e \cdot 3 &\equiv 3e \bmod \varphi(n)
\end{aligned}
$$

and so on…

The Correct Way:
Use Extended Euclidean Algorithm!

The Plain Euclidean Algorithm

- The (plain) Euclidean Algorithm finds the gcd(a,b):
- **Example:** gcd(1180, 482)

$$1180 = 2 \cdot 482 + 216$$

$$482 = 2 \cdot 216 + 50$$

$$216 = 4 \cdot 50 + 16$$

$$50 = 3 \cdot 16 + 2$$

$$16 = 8 \cdot 2 + 0.$$

- Last non-zero remainder is the gcd.

Plain Euclidean Algorithm, pg 2.

- Formally, the Euclidean algorithm for calculating gcd(a,b):

Suppose that a>b

- Divide b into a: $a = q_1 b + r_1$
- If $r_1=0$ then $b|a$ and $\gcd(a,b)=b$

  else represent b by $b=q_2 r_1+r_2$

3. Continue in this way until remainder is zero. The gcd is last non-zero remainder.

$$
\begin{aligned}
a &= q_1 b + r_1 \\
b &= q_2 r_1 + r_2 \\
r_1 &= q_3 r_2 + r_3 \\
&\dots \\
r_{k-2} &= q_k r_{k-1} + r_k \\
r_{k-1} &= q_{k+1} r_k + 0 \\
\gcd(a,b) &= r_k
\end{aligned}
$$

Getting closer to inverses…

- We can prove the following result using the Euclidean Algorithm:
- **Theorem:** Let a and b be two integers, with at least one of a, b nonzero, and let $d = \gcd(a, b)$. Then there exist integers x, y such that $ax + by = d$. In particular, if a and b are relatively prime, then there exist integers x, y with $ax + by = 1$.
- How do we use this? Suppose we know a, b, x, y as above. Then the inverse of a (mod b) is x.
- Why?
- So, we need to find these x and y!
- Euclidean algorithm will give us x and y, if we do bookkeeping!

Showing ax+by=gcd(a,b)

- The proof of the previous theorem just involves substitution.

$r_1=a-q_1\, b$ let $x_1=1$, $y_1= -q_1$, so $r_1 = x_1\, a + y_1\, b$

Next step…

$r_2 = b - q_2\, r_1$ plug in earlier result

$r_2 = b - q_2\, (x_1\, a + y_1\, b) = -x_1 q_2\, a + (b)\,(1-y_1\, q_2)$

let $x_2 = -\, x_1 q_2$, $y_2 = (1-y_1\, q_2)$

So $r_2 = x_2\, a + y_2\, b$

Follow this process repeatedly: If $r_i = x_i\, a + y_i\, b$

Then $r_{i+1} = r_{i-1} - q_{i+1}\, r_i = x_{i-1}\, a + y_{i-1}\, b - q_{i+1}\,(x_i\, a + y_i\, b)$

$= a\,(x_{i-1} - q_{i+1}\, x_i) + b\,(y_{i-1} - q_{i+1}\, y_i)$

$= a\, x_{i+1} + b\, y_{i+1}$

Since this holds for any r_i, it holds for the last one r_k=gcd(a,b).

Extended Euclidean Algorithm

- The idea in the proof leads to the Extended Euclidean Algorithm

Input: a, b non-negative with a> b

Output: d=gcd(a,b) and x and y such that ax+by=d

```
If b=0 {
    d=a; x=1; y=0; return(d,x,y); }
x2=1; x1=0; y2=0; y1=1;
While b>0 {
    q=floor(a/b);
    r=a-q*b;
    x=x2-q*x1;
    y=y2-q*y1;
    a=b;
    b=r;
    x2=x1; x1=x;
    y2=y1; y1=y; }
d=a; x=x2; y=y2;
Return(d,x,y)
```

Implementation Detail: How to multiply fast!

- RSA needs calculations like m^e mod n. How do can we do this quickly?
- If we just do sequential multiplication, it will take forever! (Remember, n is on the order of 1000 bits!!! And so is e!!!)
- To do it effectively, we use **Repeated Squaring**:
- **Example:** Let's do 2^{1234} (mod 789)

$$
\begin{aligned}
&2^4 \equiv 4^2 \equiv 16 \\
&2^8 \equiv 16^2 \equiv 256 \\
&2^{16} \equiv 256^2 \equiv 49 \\
&2^{32} \equiv 34 \\
&2^{64} \equiv 367 \\
&2^{128} \equiv 559 \\
&2^{256} \equiv 37 \\
&2^{512} \equiv 580 \\
&2^{1024} \equiv 286.
\end{aligned}
$$

- Since 1234 = 1024 + 128 + 64 + 16 + 2 (1234 = 10011010010 in binary), thus

$$2^{1234} \equiv 286 \cdot 559 \cdot 367 \cdot 49 \cdot 4 \equiv 481 \pmod{789}.$$

Making Primes, Principles pg. 1

- **Basic Principle:** Let n be an integer and suppose there exist integers x and y with $x^2 \equiv y^2 \pmod{n}$, but $x \not\equiv \pm y \pmod{n}$. Then n is composite. Moreover, $\gcd(x - y, n)$ gives a nontrivial factor of n.

- **Proof.** Let $d = \gcd(x - y, n)$. If $d = n$ then $x \equiv y \pmod{n}$, which is assumed not to happen.

Suppose $d = 1$. We know that if $a|bc$ and $\gcd(a, b) = 1$, then $a|c$.

In our case, since n divides $x^2 - y^2 = (x-y)(x+y)$ and $d = 1$, we must have that n divides $x + y$, which contradicts the assumption that $x \not\equiv -y \pmod{n}$. Therefore, d is not $= 1$, n, so d is a nontrivial factor of n.

- **Example:** Since $12^2 \equiv 2^2 \pmod{35}$, but $12 \not\equiv \pm 2 \pmod{35}$, we know that 35 is composite. Moreover, $\gcd(12 - 2, 35) = 5$ is a nontrivial factor of 35.

Making Primes, Principles pg. 2

- We may use Fermat's Little Theorem to prove numbers are **not** prime.
- Here's the way: Suppose you have a number n and want to show it is not prime. Choose a number a, and calculate

$$a^{n-1} \pmod{n}$$

If this does not equal 1, then n cannot be prime.

Why?

- Example: Show 35 is not prime.

$$2^{34} = 2^{32}\, 2^{2} = 11 * 4 = 9 \mathrel{!=} 1 \pmod{35}$$

Hence 35 is not prime.

- But, what if $a^{n-1} \pmod{n} = 1$? This does not mean n is prime.
- Numbers n such that $a^{n-1} \pmod{n} = 1$ for a particular a are said to be *pseudoprimes base a*. "a" is said to be a *liar* for n.

Making Primes, Miller-Rabin pg. 1

- **<u>Fact</u>**: Let n be an odd prime and let $n-1=2^s r$, where r is odd. Let a be any integer such that gcd(a,n)=1. Then either $a^r \equiv 1 \pmod{n}$ or $a^{2^j r} \equiv -1 \pmod{n}$ for some $0 \le j \le s-1$.
- **<u>Definition</u>**: Let n be an odd composite with $n-1=2^s r$. Let $a \in [1, n-1]$. If either $a^r \equiv 1 \pmod{n}$ or $a^{2^j r} \equiv -1 \pmod{n}$, for some $0 \le j \le s-1$ then n is a strong <u>pseudo</u>prime base a, and a is a strong liar for n.
- **<u>Fact</u>**: If n is an odd composite integer, then at most 1/4 of the numbers a are strong liars for n.
- We can use this in a Monte-Carlo algorithm to produce "primes":
  - Test t different a's.
  - Probability of falsely identifying a prime is $\le \left(\frac{1}{4}\right)^t$

Miller-Rabin Primality Test, pg. 2

- Generate a random (odd) integer n such that $n-1 = 2^s r$

For k=1 to t **do**

Choose a random integer $2 \le a \le n-2$

Calculate $y=a^r \pmod n$

If ((y!= 1) & (y != n-1)) **then**

j=1;

While ((j <= s-1) & (y != n-1)) **do**

$y=y^2 \pmod n$

If y=1 then **Return**("Composite");

j++;

enddo

If (y != n-1) then **Return**("Composite");

endif

endfor

Return("Probably Prime");

OK, we know how to make primes… Now what?

- Not all primes are good… There are some things we should check for when choosing primes…
- Make certain (p-1) or (q-1) do not have many small factors!

Why? Else, the (p-1)-Factorization Method will make n easy to factor

- Make p and q of different lengths

Why? The following result applies…

Theorem: Suppose p and q are primes with $q < p < 2q$. Let $n=pq$, and choose e and d as in the RSA algorithm. If $d < (1/3)n^{1/4}$, then d can be calculated quickly.

- Make certain adversary doesn't know many of the digits of p or q.

Why? The following result applies…

Theorem: Let $n=pq$ have m digits. If we know the first m/4 or the last m/4 digits of p then we can efficiently factor n.

A Little on the Security of RSA

- The security of RSA is based upon the assumption that factoring the product of two large primes is **hard**.
- What if we assume factoring is impossible, then what are the logical implications?
- Most arguments go like this:
  - If factoring is hard, and XYZ is directly related to factoring, then XYZ is hard.
  - Or, say it another way… Assume XYZ is easy, then show XYZ is equivalent to factoring, which contradicts the fact that factoring is impossible!

An Example of this Principle

- Suppose Eve sees n and e (they're public!!!). We claim she can't figure out φ(n).
- **Proof:** We show that knowing n and φ(n) is equivalent to factoring, i.e. finding p and q !

p and q are the roots of $(x-p)(x-q) = x^2-(p+q)x + pq$.

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

So $(x-p)(x-q) = x^2 - (n-\varphi(n)+1)x + n$

We can solve this using quadratic formula…

$$p, q = \frac{(n-\varphi(n)+1) \pm \sqrt{(n-\varphi(n)+1)^2 - 4n}}{2}$$

So, if we could find φ(n) we would be able to factor n!!!

- This removes the **easy** way to find d by finding φ(n).

Factorization and Fermat Factorization

- Modern factorization methods involve significant mathematical machinery. However, we may use a simple factoring method to see what not to do when setting up RSA

Fermat Factorization:

Start with $n=pq$. We try to write $n=x^2 - y^2 = (x+y)(x-y)$

Can we even do this? Yes, always!

Let $p=x+y$ and $q = x-y$

$$\left.\begin{array}{l} x+y = p \\ x-y = q \end{array}\right\} \longrightarrow 2x = p+q \qquad x = (p+q)/2$$

$$\left.\begin{array}{l} -(x+y = p) \\ x - y = q \end{array}\right\} \longrightarrow -2y = q-p \qquad y=(p-q)/2$$

So this is always possible.

Now, try $n+1^2, n+2^2, n+3^2, \ldots$ until we find a square

If $n+y^2 = x^2$ then we are done! $n=x^2 - y^2$

- This method only works well when $(x+y)$ and $(x-y)$ are close! That is, when p and q are close! So, we must **not** choose p and q too close.