Lecture 7

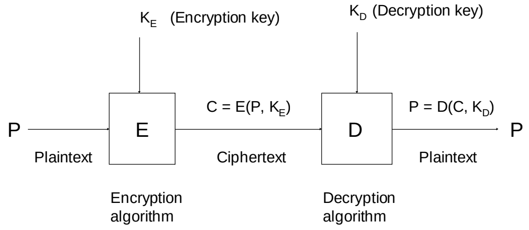
Public Key Cryptography:

Solving the age-old problems of mankind in our generation

Wholeness Statement

For thousands of years it was thought that both the sender and the receiver of an enciphered message needed to share the key. In 1976, Diffie and Hellman discovered a new type of cryptography that used different keys to encipher and decipher. Life is a field of all possibilities. Maharishi has introduced a technology that will put an end to the age-old notion that the nature of life is to suffer.

Relationship Between Plaintext and Ciphertext



Overview

1. Asymmetric cryptography uses one key to encipher and a different key to decipher.
2. The key used to encipher is public, the one used to decipher is private.
3. The RSA algorithm uses the product of two extremely large prime numbers to compute the public and private keys. It must be infeasible to factor this product, since otherwise the private key could be calculated from the public key.
4. The RSA algorithm can also encipher using the private key and decipher using the public key. This is used for digital signatures (origin integrity)
5. A cryptographic checksum is a one-way function that converts a message into a fixed length number (e.g., 128 bits). It can be used for data integrity.
6. PGP (pretty good privacy) is a program used to send secure email. It nicely integrates the use of symmetric cryptography, asymmetric cryptography, and cryptographic checksums.

Asymmetric Cryptography

- In 1976 Diffie and Hillman discovered asymmetric cryptography
- The sender and receiver of a cryptographic message use different keys
  - a public key known by everybody and
  - a secret key known only to the receiver
- Public key cryptography solves the problem of how to get the key to the recipient without the hacker eavesdropping and seeing it also

Generation of Asymmetric Keys

To send and receive private data, the receiver would first mathematically generate two keys, a public key and a private key

- Usually, key generation is automated, possibly with a user-selected password fed into the algorithm as a seed

The keys must have the following properties:

1. Anything encrypted with the public key can be decrypted with the private key
2. Anything encrypted with the public key cannot be decrypted with the public key.

Sending an encrypted message

- Given these properties, the receiver can put the public key in a public place (e.g., on a web page) for all to see (including the attacker).
  - The receiver must then keep the private key in a safe place that only he/she can access.
- The public key is used to create an encrypted message (ciphertext) which would be sent in the clear.
  - No need to worry about the attacker decrypting it because the public key (the only thing the attacker knows) cannot be used to decrypt the message.
- Upon receipt, the private key is retrieved from its safe place and used to decrypt the ciphertext.
- Thus only the receiver can read the message as long as the secret key is kept safe from the attacker
  - Thus the private key must never, ever be transmitted over a network.

Public Key Cryptosystems

Must meet the following three conditions:

1. It must be computationally easy to encrypt or decrypt a message given the appropriate key
2. It must be computationally infeasible (intractable) to derive the private key from the public key
3. It must be computationally infeasible to determine the private key from a chosen plaintext attack.

Main Point

1. Public key cryptography is a symmetry breaking technology, a symmetric key is no longer needed; instead there are two different keys used, one to encipher and one to decipher. Physics has developed the concept symmetry breaking to describe the origin of the universe and John Hagelin compares it to consciousness knowing itself.

RSA

A public-key cryptosystem used for secrecy and authentication

Prime and Relatively Prime Numbers

Definition: n is *prime* iff n is only divisible by 1 and n

Definition: n is *relatively prime* to m iff gcd(n, m) = 1

- gcd calculates the greatest common divisor
- Example: 6, 35 are relatively prime
  - factors of 6 are 1, 2, 3, 6,
  - factors of 35 are 1, 5, 7, 35, so the gcd is 1
- Example: 10, 35 are not relatively prime since their gcd is 5

How RSA Works

1. Choose two large primes, p and q (typically 1024 bits)
2. Compute N = p * q
3. Compute phi = (p-1) * (q-1)
4. Choose a number relatively prime to phi (no common divisors greater than 1) and call it e such that max(p,q)<e<phi
5. Find d<phi such that (e * d) mod phi = 1 (i.e., e*d -1 is divisible by phi)
6. Public key is (e, N), private key is (d, N)
7. If N is x bits long then the message must be broken down into blocks of x bits so that when a block is viewed as a number it is less than N.
8. Encrypt a block P as follows : P^e (mod N)
   (RSA is an exponentiation cipher)
9. Decrypt a block C as follows : C^d (mod N)

Select p and q so N is large

The number of bits in N determines the block size

Thus N should be large, otherwise the attacker might be able to rearrange ciphertext to get meaningful plaintext

- For example, if each character has to be encrypted as a separate block, then a message “ON” could be changed to “NO” by rearranging the blocks
- If we choose p and q to be 512 bit numbers, then N will be a 1024 bit number. Thus the message will be broken down into 1024/8 = 128 byte blocks.

Security of the RSA

- When RSA first came out its inventors offered a $100 reward for the first person to decrypt a message encrypted with their public key.
- It used a public key size of ~428 bits.
- It was decrypted by a group using 1600 computers over the Internet; it took 8 months.
- Recall that the public key (e, N) is available to the whole world, so attackers have the value of e and N.
- If they can find p and q such that p*q = N, then they can compute d and would have the private key.
- But if N is large enough, it is computationally infeasible to find its prime factors.
- This is why public key cryptography is secure.

Feasibility of Factoring

- The security of the RSA algorithm rests in the difficulty of factoring the product of two large prime numbers (i.e., finding p and q given only their product N)
- In October of 1992, $2^{523} – 1$ was factored into primes using a MasPar
  - an SIMD machine with 16384 processors
- Each processor could add 200,000 integers per second
- Took three weeks using the “number field sieve” algorithm
- If no new algorithms are developed for factoring numbers, then 2048-bit RSA keys are considered to be secure

Security of the RSA Public-Key Scheme

- Rivest, Shamir, and Adleman (RSA) recommend a 200-digit N, whose factorization by known techniques would require more than 10^{23} operations
- However, it has not been proven that factoring cannot be done in polynomial time (which would make factoring large numbers feasible and would render RSA insecure)

Problems with Public-Key Systems

- The main problem is that public-key systems are 1000 times slower than symmetric cryptography
  - Which security principle?
  - See PGP discussion below for a way to avoid enciphering large messages
- Another problem is how to reliably acquire the recipient's public key
  - We will go into the details in Lecture 9
  - Which security principle?

RSA Encryption and Decryption

1. the plaintext is first encoded into a sequence of integers between 0 and n-1
2. each integer, M_i, 0<= M_i <=(n-1), of the encoded plaintext is encrypted by

   $C_i = E(M_i, (n, e)) = (M_i)^e \pmod n$
3. decryption is performed using the private key (n, d)

   $M_i = D(C_i, (n, d)) = (C_i)^d \pmod n$

```
// Based on a javascript program at
   http://www.codeproject.com/jscript/JscriptRSA.asp
public class RSA
{
 // prime numbers are small enough so N needs only 8 bits.
 // This is not very secure at all, essentially
 // a glorified Caesar cipher
  int p = 7;
  int q = 29;
  public RSA()
  {
    String plainText = "sold";  // text to encrypt
    int N = p * q;
    int phi = (p-1)*(q-1);
    int e = relPrime(phi);     // (e, N) = public key
    int d = calculate_d(phi, e); // (d, N) = private key

    System.out.println("*****************");

// next divide the plaintext into 8 bit blocks
    char[] blocks = getBlocks(plainText);
```

```
// Encrypt with public key, decrypt with private key.
    for (char block : blocks)  // each block is 8 bits
    {
      int c = encrypt(e, N, block);  // using public key
      int x = decrypt(d, N, c);  // using private key
      System.out.format("%c->%d->%c\r\n", block, c, x);
      // proves it works.
    }
    System.out.println("*****************");

    // RSA has the interesting property that
    //     the private key can be used to encrypt
    // and the public key can be used to decrypt:
    for (char block : blocks)  // each block is 8 bits
    {
      int c = encrypt(d, N, block);  // using private key
      int x = decrypt(e, N, c);  // using public key
      System.out.format("%c->%d->%c\r\n", block, c, x);
      // proves it works.
    }
  }
```

```
public static void main(String[] args)
  {
    new RSA();
  }
  // Each block is 8 bits long.
  // In the real world would be at least 1024 bits.
  public char[] getBlocks(String text)
  {
      return text.toCharArray();
  }

  int power(int n, int exp)
  {
    int temp=1, i;
    for(i=1;i<=exp;i++)
      temp*=n;
    return temp;
  }
```

```
  int relPrime(int phi)
  {
    int rel=3;

    while (gcd(phi,rel)!=1)
      rel++;
    return rel;
  }

  int gcd(int a, int b)
  {
    int r;
    while (b>0)  {
      r=a%b;
      a=b;
      b=r;
    }
    return a;
  }
```

```
 int calculate_d(int phi, int e)
  {
    int x,y,x1,x2,y1,y2,temp,r,orig_phi;
    orig_phi = phi;
    x2=1;x1=0;y2=0;y1=1;
    while (e>0)   {
      temp = phi / e;
      r = phi-temp*e;
      x = x2-temp*x1;
      y = y2-temp*y1;
      phi = e;
      e = r;
      x2 = x1;
      x1 = x;
      y2 = y1;
      y1 = y;
      if (phi==1)    {
        y2 += orig_phi;
        break;
      }
    }
    return y2;
  }
```

```
 // encrypts block M using public key (e, N)
  int encrypt(int e, int N, int M)
  {
    int r,i=0,prod=1,rem_mod=0;
    while (e>0)    {
      r = e % 2;
      if (i++ == 0)
        rem_mod = M % N;
      else
        rem_mod = power(rem_mod,2) % N;
      if (r==1)
      {
        prod*=rem_mod;
        prod=prod % N;
      }
      e= e/2;
    }
    return prod;
  }
 // Decrypt ciphertext c using private key (d, N)
  int decrypt(int d, int N, int c)
  {
    int r, i = 0, prod = 1, rem_mod = 0;
    while (d>0)
    {
      r = d % 2;
      if (i++ == 0)
        rem_mod = c % N;
      else
        rem_mod = power(rem_mod,2) % N;
      if (r == 1)
      {
        prod *= rem_mod;
        prod = prod % N;
      }
      d = d / 2;
    }
    return prod;
  }
}
```

Main Point

2. Discovering the prime factors of a very large composite number is computationally infeasible and is the reason it is possible to make public one of the keys used in public key cryptography. Full appreciation of pure consciousness as the basis of life is physiologically infeasible without a pure nervous system; the TM technique is a way to remove the stresses that block appreciation of this full value of life.

Digital Signatures

Digital Signatures

- The above program demonstrated that, in RSA, the encryption and decryption functions are inverses of each other.
- So a person could encrypt with their private key.
- This, of course, would not be a wise way to encrypt a secret message since everyone has the public key and could decrypt it.
- However, it is a good way for a sender to assure the receiver that he sent the message.
- We now demonstrate how RSA can be used to provide authentication as well as secrecy

Signing Messages

Subject A could send an encrypted, signed message to B as follows:

1. A encrypts the message M with his private key to get M1
2. A encrypts M1 with B's public key to get M2
3. A sends M2 to B
4. B decrypts M2 with his private key to get M1
5. B uses A's public key to decrypt M1 to get M.

If M is intelligible, then B is confident that A sent it because only A knows the secret key used to create M1

This is origin integrity

Origin Integrity

- Note that if the message is not confidential, then it would not be necessary to encrypt it with Bob's public key.
- It wouldn't matter that the whole world could decrypt it.
- So the sender, A, would just encrypt the message with his/her private key.
- All that matters is that the receiver knows that A must have encrypted it
  - i.e., nobody else could because only A has the private key.

Cryptographic Checksums

Chapter 8.4

Cryptographic Checksums

- An established way for a receiver to verify the source integrity of a received message
- Used instead of encrypting the entire message with the sender's private key (as we did previously)
- Instead, the checksum of the message is encrypted with the sender's private key
- See the section on PGP below for an example of a system that uses this technique.

Cryptographic Checksums

Definition 8-2 A cryptographic checksum function (also called a strong hash function or a strong one-way function) h:A-->B is a function that has the following properties:

1. For any x in A, h(x) is easy to compute.
2. For any y in B, it is computationally infeasible to find x in A such that h(x)=y.
3. Given any x in A, it is computationally infeasible to find another x' in A such that x != x' and h(x')=h(x).

- Also, sometimes called a *trapdoor* function

Properties of Checksums

- Messages that differ very little are radically different
  - Even if only one bit is different
- The checksum contains fewer bits than the original message,
  - Thus more than one message must produce the same checksum
  - This is an application of the pigeonhole principle which states

    If there are n containers for n+1 objects, at least one container will hold two objects.

Digital Signatures Summary

- The document is first run through a one-way hashing algorithm that is very difficult to invert (e.g., MD5 or SHA)
- The hash typically produces a fixed-length result independent of the original document size
- The document owner applies his private key to the hashed document
  - this encrypted result is called the *signature block*
- The signature block is appended to the document and both are sent to the receiver
- When the document arrives, the receiver first computes the hash of the document as agreed upon
- Then the receiver applies the sender’s public key to the signature block, getting the hash computed by the sender
- This hash must match the hash of the document computed by the receiver

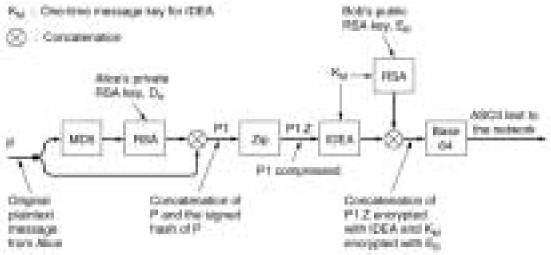
- A digital signature also makes it possible to sign e-mail messages and other digital documents such that they cannot be repudiated by the sender later

PGP

Pretty Good Privacy

PGP

- A program that demonstrates many of the cryptographic concepts that we have learned so far is PGP
- PGP is often used for signing and encrypting emails.
- Here is an overview of the process (the diagram is from "Computer Networks, fourth edition", by Andrew S. Tanenbaum):



K_M : One-time message key for IDEA
: Concatenation
Bob's public RSA key, E_B
K_M
RSA
Alice's private RSA key, D_A
P
MD5
RSA
P1
Zip
P1.Z
IDEA
Base 64
ASCII text to the network
P1 compressed
Original plaintext message from Alice
Concatenation of P and the signed hash of P
Concatenation of P1.Z encrypted with IDEA and K_M encrypted with E_B

PGP

Step by step here is what is going on:

0. Alice (A) wants to send plaintext P to Bob(B). She wants to encrypt it and digitally sign it so Bob knows it's from her.

1. Alice computes the cryptographic checksum of P using the MD5 algorithm
2. She then digitally signs the cryptographic checksum using her private key and appends the signed checksum to the end of P.
3. She zips the digitally signed P. Note that P has not been encrypted yet.
4. She creates a random session key (more on this next week) using IDEA which is a classical crypographic system (i.e., the sender and receiver use the same key).
5. She uses the random session key to encrypt the zip of the digitally signed P.

PGP

6. She encrypts the random session key using Bob's public key. In this way only Bob can decrypt it.
7. She appends the encrypted random session key to the end of the encrypted zip of P.
8. She converts the result to Base64 so that it can be transmitted over the Internet without having any bits dropped.
9. Bob receives the message and converts it from Base64.
10. He uses his private key to decrypt the random session key and then uses the random session key to decrypt the zip of P.
11. He unzips it. He now has the digitally signed P.
12. He uses Alice's public key to decrypt the digital signature.
13. He then applies that same MD5 cryptographic checksum to P and compares the result to what he got in step 12. If they are the same, then Bob is confident that Alice sent the message.

- More information on PGP is available at http://en.wikipedia.org/wiki/Pretty_Good_Privacy
- Just because cryptography is used does not mean the system is secure.
  - See http://www.cl.cam.ac.uk/~rja14/wcf.html for some examples.

Main Point

3. A cryptographic checksum function is a one-way function that "collapses" a message to a reasonably unique number. This is reminiscent of the collapse of infinity to a point described in Vedic Science. The difference is that the checksum function is one-way and hence knowledge of the message is lost. This is not the case in Vedic Science, the point is just another perspective on infinity.

CONNECTING THE PARTS OF KNOWLEDGE WITH THE WHOLENESS OF KNOWLEDGE

1. Public key cryptography uses two keys, one for enciphering and one for deciphering.

2. It is computationally infeasible to derive the private key from the public key. To do so requires factoring a very large composite number into its prime factors.

3. Transcendental Consciousness is the field of all possibilities; it breaks the boundaries relative creation.

4. Wholeness moving within itself: in Unity Consciousness one has total knowledge of the object, namely it is a manifestation of one's own Self and thus is as dear as one’s own Self.