Cryptography

- Much of computer security is about keeping secrets
- One method of doing so is to make it hard for others to read the secrets
- While (usually) making it simple for authorized parties to read them
- That's what cryptography is all about

What Is Encryption?

- Encryption is the process of hiding information in plain sight
- Transform the secret data into something else
- Even if the attacker can see the transformed data, he can't understand the underlying secret
- Usually, someone you want to understand it can

Cryptography Terminology

- Typically described in terms of sending a message
  - Though it's used for many other purposes
- The sender is *S*
- The receiver is *R*
- *Encryption* is the process of making message unreadable/unalterable by anyone but *R*
- *Decryption* is the process of making the encrypted message readable by *R*
- A system performing these transformations is a *cryptosystem*
  - Rules for transformation sometimes called a *cipher*

Plaintext and Ciphertext

- *Plaintext* is the original form of the message (often referred to as *P*)

Transfer $100 to my savings account

- *Ciphertext* is the encrypted form of the message (often referred to as *C*)

Sqzmredq #099 sn lx rzuhmfr zbbntms

Cryptographic Keys

- Most cryptographic algorithms use a *key* to perform encryption and decryption
  - Referred to as *K*
- The key is a secret
- Without the key, decryption is hard
- With the key, decryption is easy
- Reduces the secrecy problem from your (long) message to the (short) key
  - But there's still a secret

More Terminology

- The encryption algorithm is referred to as *E()*
- $C = E(K,P)$
- The decryption algorithm is referred to as *D()*
- The decryption algorithm also has a key
- The combination of the two algorithms are often called a *cryptosystem*

Symmetric and Asymmetric Cryptosystems

- Symmetric cryptosystems use the same keys for E and D :

  $P = D(K, C)$

  – Expanding, $P = D(K, E(K,P))$
- Asymmetric cryptosystems use different keys for E and D:

  $C = E(K_E,P)$

  $P = D(K_D,C)$

  – Expanding, $P = D(K_D , E(K_E ,P))$

Desirable Characteristics of Keyed Cryptosystems

- If you change only the key, a given plaintext encrypts to a different ciphertext
- Same applies to decryption
- Changes in the key ideally should cause unpredictable changes in the ciphertext
- Decryption should be hard without knowing the key
- The less a given key is used, the better (in security terms)

Cryptography and Operating Systems

- What does cryptography have to offer operating systems?
- Which hard security problems in operating systems can we solve with cryptography?
- Where doesn’t it help?

Cryptography and Secrecy

- Pretty obvious
- Only those knowing the proper keys can decrypt the message
  - Thus preserving secrecy
- Used cleverly, it can provide other forms of secrecy
- Clear where we'd use this for distributed systems
- Where does it make sense in a single machine?

Cryptography and Authentication

- How can I prove to you that I created a piece of data?
- What if I give you the data in encrypted form?
  - Using a key only you and I know
- Then only you or I could have created it
  - Unless one of us told someone else the key . . .
  - Or one of us is trying to screw the other

Cryptography and Integrity

- Changing one bit of a piece of ciphertext completely garbles it
  - For many forms of cryptography
- If a checksum is part of encrypted data, that's detectable
- If you don't need secrecy, can get the same effect
  - By encrypting only the checksum

Symmetric Cryptosystems

- $C = E(K,P)$
- $P = D(K,C)$
- *E()* and *D()* are not necessarily the same operations

Advantages of Symmetric Cryptosystems

- \+ Encryption and authentication performed in a single operation
- \+ Well-known (and trusted) ones perform much faster than asymmetric key systems
- \+ No centralized authority required
  - Though key servers help a lot

Disadvantages of Symmetric Cryptosystems

- Encryption and authentication performed in a single operation
  - Makes signature more difficult
- Non-repudiation hard without servers
- Key distribution can be a problem
- Scaling
  - Especially for Internet use

Some Popular Symmetric Ciphers

- The Data Encryption Standard
  - The old US encryption standard
  - Still fairly widely used, due to legacy
  - Kind of weak by modern standards
- The Advanced Encryption Standard
  - The current US encryption standard
  - Probably the most widely used cipher
- Blowfish
- There are many, many others

Symmetric Ciphers and Brute Force Attacks

- If your symmetric cipher has no flaws, how can attackers crack it?
- *Brute force* – try every possible key until one works
- The cost of brute force attacks depends on key length
  - Assuming random choice of key
  - For N possible keys, attack must try N/2 keys, on average, before finding the right one

How Long Are the Keys?

- DES used 56 bit keys
  - Brute force attacks on that require a lot of time and resources
  - But they are demonstrably possible
  - Attackers can thus crack DES, if they really care
- AES uses either 128 bit or 256 bit keys
  - Even the shorter key length is beyond the powers of brute force today
  - 2^{127} decryption attempts is still a lot, by any standard

Asymmetric Cryptosystems

- Often called *public key cryptography*
  - Or PK, for short
- The encrypter and decrypter have different keys
  - $C = E(K_E,P)$
  - $P = D(K_D,C)$
- Often works the other way, too
  - $C' = E(K_D,P)$
  - $P = D(K_E,C')$

Using Public Key Cryptography

- Keys are created in pairs
- One key is kept secret by the owner
- The other is made public to the world
  - Hence the name
- If you want to send an encrypted message to someone, encrypt with his public key
  - Only he has private key to decrypt

Authentication With Public Keys

- If I want to “sign” a message, encrypt it with my private key
- Only I know private key, so no one else could create that message
- Everyone knows my public key, so everyone can check my claim directly
- Much better than with symmetric crypto
  - The receiver could not have created the message
  - Only the sender could have

PK Key Management

- To communicate via shared key cryptography, key must be distributed
  - In trusted fashion
- To communicate via public key cryptography, need to find out each other's public key
  - "Simply publish public keys"
- Not really that simple, for most cases

Issues With PK Key Distribution

- Security of public key cryptography depends on using the right public key
- If I am fooled into using wrong one, that key's owner reads my message
- Need high assurance that a given key belongs to a particular person
  - Either a *key distribution infrastructure*
  - Or use of *certificates*
- Both are problematic, at high scale and in the real world

The Nature of PK Algorithms

- Usually based on some problem in mathematics
  - Like factoring extremely large numbers
- Security less dependent on brute force
- More on the complexity of the underlying problem

Choosing Keys for Asymmetric Ciphers

- For symmetric ciphers, the key can be any random number of the right size
  - You can't do that for asymmetric ciphers
- Only some public/private key pairs "work"
  - Generally, finding a usable pair takes a fair amount of time
  - E.g., for RSA you perform operations on 100-200 digit prime numbers to get keys
- You thus tend to use one public/private key pair for a long time
  - Issues of PK key distribution and typical usage also suggest long lifetimes for these keys

Example Public Key Ciphers

- RSA
  - The most popular public key algorithm
  - Used on pretty much everyone's computer, nowadays
- Elliptic curve cryptography
  - Not as widely used as RSA
  - Tends to have better performance
  - Not as widely used or studied

Security of PK Systems

- Based on solving the underlying problem
  - E.g., for RSA, factoring large numbers
- In 2009, a 768 bit RSA key was successfully factored
- Research on integer factorization suggests keys up to 2048 bits may be insecure
  - In 2013, Google went from 1024 to 2048 bit keys
- Size will keep increasing
- The longer the key, the more expensive the encryption and decryption

Combined Use of Symmetric and Asymmetric Cryptography

- Very common to use both in a single session
- Asymmetric cryptography essentially used to "bootstrap" symmetric crypto
- Use RSA (or another PK algorithm) to authenticate and establish a *session key*
- Use DES or AES with session key for the rest of the transmission

For Example,

Alice wants to share K_S only with Bob

Bob wants to be sure it's Alice's key

Only Bob can decrypt it

Only Alice could have created it

Alice

K_{EA} K_{DA}

K_{DB}

Bob

K_{EB} K_{DB}

K_{DA}

K_S

$C=E(K_S,K_{DB})$

$M=E(C,K_{EA})$

$K_S=D(C,K_{EB})$

$C=D(M,K_{DA})$

Secure Hash Algorithms

- A method of protecting data from modification
- Doesn't actually prevent modification
- But gives strong evidence that modification did or didn't occur
- Typically used with other cryptographic techniques
  - Like *digital signatures*, a method of using cryptography to sign something

Idea Behind Secure Hashes

- Apply a one-way cryptographic function to data in question
- Producing a much shorter result
- Save the cryptographic hash
  - Or, for messages, send it with the message
- To check for tampering, repeat the function on the data and compare to the hash value
- If attacker can get at the hash, often you also encrypt it