Distributed Systems

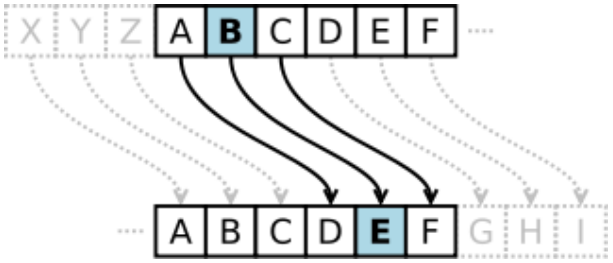
Security and Cryptography Concepts

Assumptions and Guidelines

- Interfaces are exposed
  - How?
- Networks are insecure
  - How?
- Algorithms are available to attackers.
  - We assume they understand RSA, DES, etc.
  - Why not keep them secret?
- Attackers may have have large resources.
  - Why?
- Limit the lifetime and scope of secrets.
- Minimize the trusted base.

Julius Caesar (shift) cipher

- Pick a *key* (a number)
- Shift the letters of the *plaintext* by the key to create the *ciphertext*.
- E.g.
  - Plaintext: Yellow cake
  - Key: 3
  - Ciphertext: Bhoorz fdnh



Source: http://en.wikipedia.org/wiki/File:Caesar3.svg

Basic Crypto Terminology:

- Is this *secret-key* or *public-key* algorithm?
- Is this a *symmetric* or *asymmetric* algorithm?
- Is it vulnerable to *brute-force attack?*
- Is it a *block-cipher*?

Block cipher

How could you make it a *block-cipher*?

- E.g. Two characters, Key: 1
- cake
- 0011000110110011
- 0011001010110100

Block cipher chaining

How could you use *block-cipher chaining*?

- Key: 1
- cake
- 0011000110110101
- 01000101 - xor
- 01000110 - encrypt
- 0100011101101 - xor
- 0100011101110 - encrypt

Block cipher

How could you use *block-cipher chaining*?

- Key: 1
- cake
- 0011000110110101
- 01000101 - xor
- 01000110 - encrypt
- 010001101101 - xor
- 010001101110 - encrypt
- Do the last letter

Block cipher

How could you use *block-cipher chaining*?

- Key: 1
- cake
- 0011000110110101
- 01000101 - xor
- 01000110 - encrypt
- 0100011011O1 - xor
- 0100011011O0 - encrypt

- 0100011011101011 - xor
- 0100011011101100 - encrypt

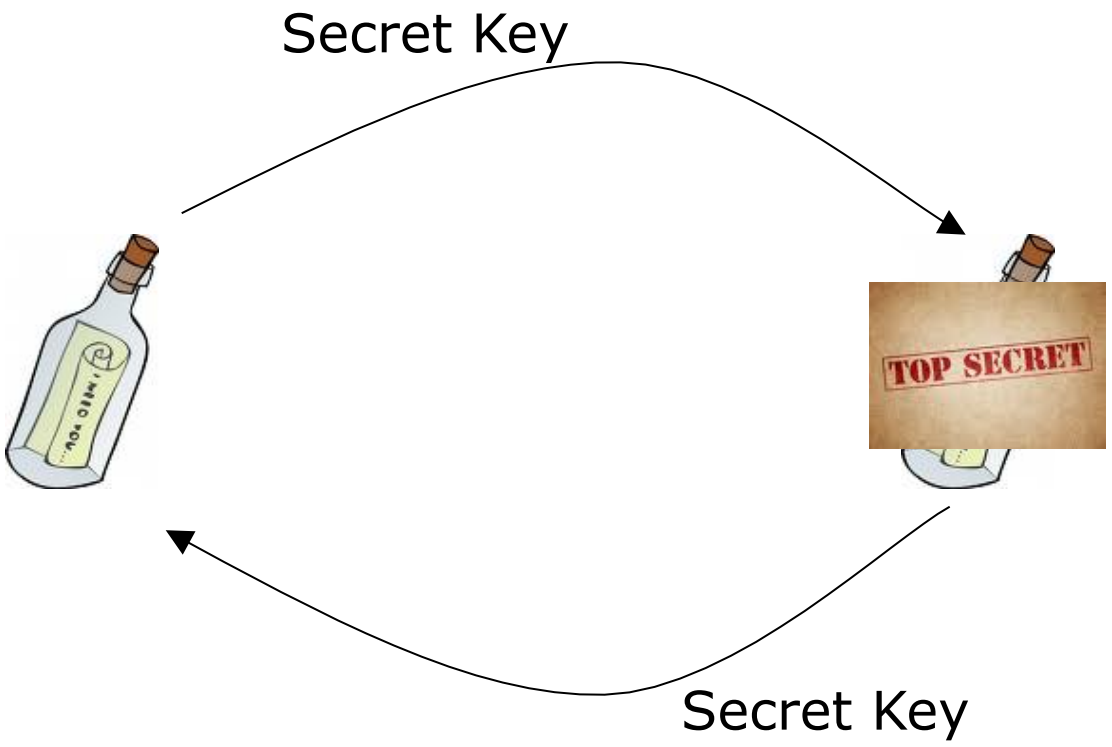
Advanced Encryption Standard

- Very complex chaining block cipher
- Symmetric algorithm
- Adopted by US National Institute of Standards
  - Replaced DES
- It is what most browsers use:
  - https://www.fortify.net/cgi/ssl_2.pl

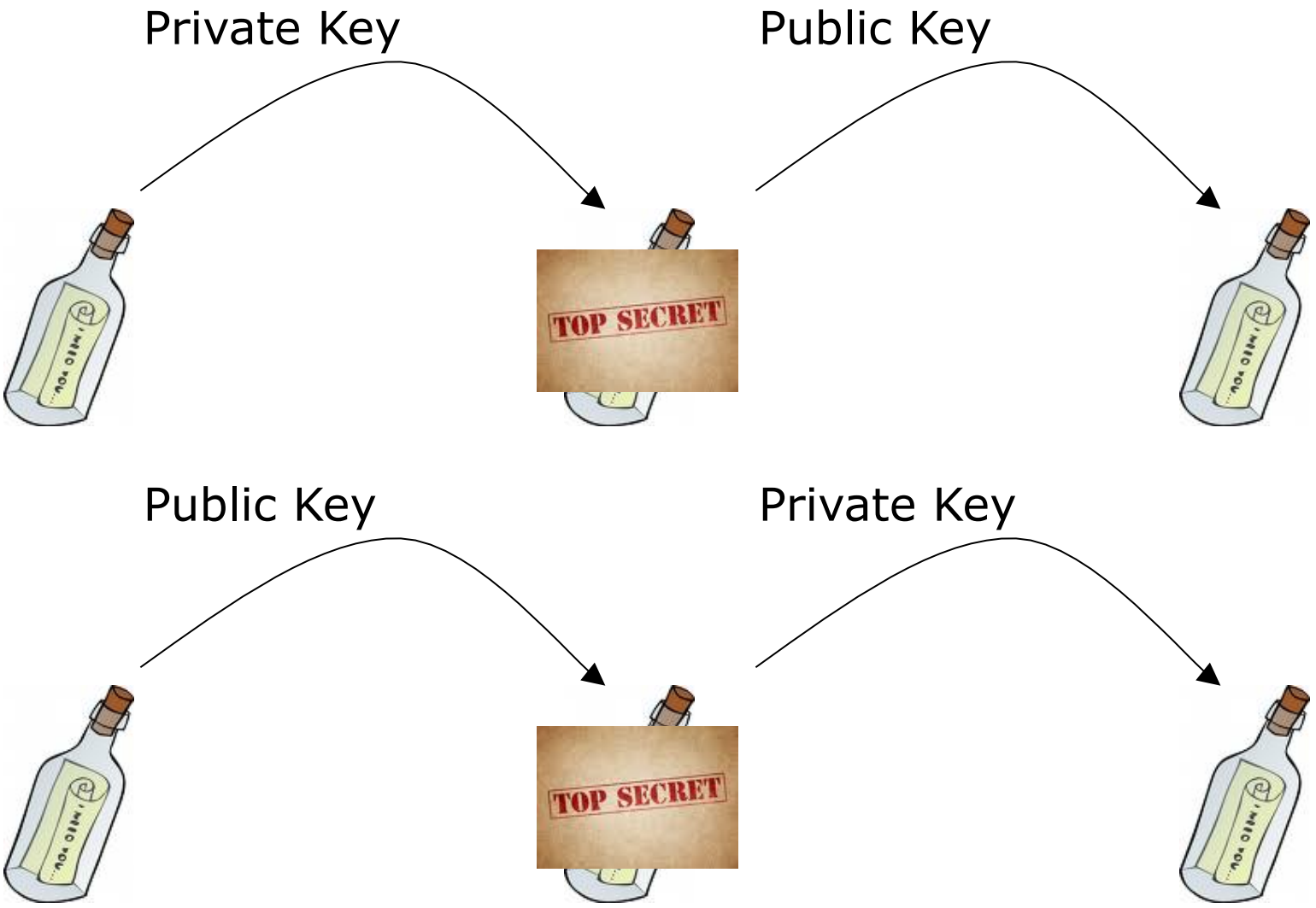
Asymmetric algorithms

- Symmetric algorithms require Alice and Bob to share a secret (the key).
  - Both can encrypt and decrypt messages
- Asymmetric algorithms allow for ***not*** sharing a secret.
  - Alice can encode a message for Bob
  - She ***cannot*** decode messages already encoded for Bob

Symmetric



Asymmetric


Private Key
Public Key
TOP SECRET
Public Key
Private Key
TOP SECRET

Public Key

- Uses asymmetric keys
- Single public key
  - Let the world see
- Single private key
  - You keep secret
- Public can:
  - Encrypt a message for you using the public key
  - Decrypt a message encrypted with your private key using your public key.

What doesn't work

- No one can decrypt a message encrypted with your public key with your public key
  - Only with the private key
- You can't decrypt a message that you encrypted with your private key with your private key
  - Only with your public key
  - (You should not be needing to do this)

Symmetric vs Asymmetric algorithms

- Symmetric
  - Fast
  - Difficult to distribute keys and and keep them secure
- Asymmetric
  - 100 to 1000 times slower
  - Can allow for public keys
- Best of both worlds, which we will discuss more later:
  - Use asymmetric keys to exchange symmetric keys at the beginning of a conversation
  - Symmetric keys will have the lifespan of that conversation
  - E.g.: SSL, PGP

Scenario

- I want President Obama to be able to send me confidential email messages:
  - I publish my public key on my web site
  - He encrypts his email with my public key
  - Sends it to me
    - What can Eve see?
  - Only I can decrypt his email with my private key

But who sent it?

- How do I know the message came from President Obama?
- How do I know it wasn't that sneaky Mallory, pretending to be the President?
- How can I be sure that the message is coming from the President?

Solution

- President Obama puts his public key on his web site
- He encrypts his message with his private key
- He encrypts that encrypted message with my public key
- He sends it to me.
- What does Eve see?
- What do I do?
- How do I know it is from President Obama?
- If he encrypted it with his private key, why did he need to use my public key?
- Could Mallory fake it?

RSA

- RSA is a commonly used for public key encryption

The RSA Public-Key Cryptosystem

- Developed by Rivest, Shamir, and Aldeman in 1977

The RSA Public-Key Cryptosystem

1. Select at random two large prime numbers p and q.
   - These numbers would normally be about 500 digits
2. Compute n by the equation $n = pq$.
3. Compute $\phi(n) = (p - 1)(q - 1)$
4. Select a small odd integer e that is relatively prime to $\phi(n)$

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

- Called Euler's totient function
  - (not important here)
- What's interesting is its range:
  - Equals the number of positive integers less than or equal to n that are coprime to n
  - E.g. $\phi(15) = 8$
    - {1, 2, 4, 7, 8, 11, 13, 14}
  - So when using big prime numbers, it will also be very big.

The RSA Public-Key Cryptosystem

1. Select at random two large prime numbers p and q.
   - These numbers would normally be about 500 digits
2. Compute n by the equation $n = pq$.
3. Compute $\phi(n) = (p - 1)(q - 1)$
4. Select a small odd integer e that is relatively prime to $\phi(n)$

Why small odd integer?

- The public key will be
  - {that integer, n}
- Your private key will be
  - {another integer, n}
- The range of your private integer is dependent on $\phi(n)$, which is quite large
  - So you can choose 3
  - But can be some risk from very small integers
  - Commonly use 65537 (a prime)

The RSA Public-Key Cryptosystem

1. Select at random two large prime numbers p and q.
   - These numbers would normally be about 500 digits
2. Compute n by the equation $n = pq$.
3. Compute $\phi(n) = (p - 1)(q - 1)$
4. Select a small odd integer e that is relatively prime to $\phi(n)$

The RSA Public-Key Cryptosystem

5. Compute d as the multiplicative inverse of e modulo $\phi(n)$.
   - A theorem in number theory asserts that d exists and is uniquely defined.
6. Publish the pair P = (e,n) as the RSA public key.
7. Keep secret the pair S = (d,n) as the RSA secret key.

The RSA Public-Key Cryptosystem

8. To encrypt a message M compute
   $C = M^e \pmod{n}$

9. To decrypt a message C compute
   $M = C^d \pmod{n}$

Key Selection Phase

1. Select at random two large prime numbers p and q. These numbers would normally be about 500 digits in length.

   $p = 3 \quad q = 11$

2. Compute n by the equation n = pq.

   $n = 33$

3. Compute $\phi(n) = (p - 1)(q - 1)$

   $\phi(n) = (2) * (10) = 20$

Key Selection Phase

$p = 3 \quad q = 11 \quad n = 33 \quad \phi(n) = 20$

4. Select a small odd integer e that is relatively prime to $\phi(n)$

   $e = 3$

Key Selection Phase

$p = 3 \quad q = 11 \quad n = 33 \quad \phi(n) = 20 \quad e = 3$

5. Compute d as the multiplicative inverse of e, modulo $\phi(n)$.

We need a d so that $ed \bmod \phi(n) = 1$

Let's try 1.
$3 * 1 \bmod 20 = 3 \bmod 20 = 3$. Nope.

Key Selection Phase

$p = 3 \quad q = 11 \quad n = 33 \quad \phi(n) = 20 \quad e = 3$

We need a d so that $ed \bmod \phi = 1$

Let's try 2.

$3 * 2 \bmod 20 = 6 \bmod 20 = 6$. Nope.

Lets search using a spreadsheet...

$3 * 7 \bmod 20 = 21 \bmod 20 = 1$. We found it!

Key Selection Phase

$p = 3 \quad q = 11 \quad n = 33 \quad \phi(n) = 20 \quad e = 3 \quad d = 7$

6. Publish the pair P = (e,n) as the RSA public key.

"Hey everyone, my key pair is 3 and 33"

7. Keep secret the pair S = (d,n) as the RSA secret key.

"I'm not telling anyone about 7 and 33!!"

Message encoding phase

Bob's public keys are e = 3 n = 33

Alice wants to send the letter 'd' to Bob.
Suppose that we have a public code where
'a' = 0 'b' = 1 'c' = 2 'd' = 3 and so on…

Alice's software knows that

8. To encrypt a message M compute
   $C = M^e \pmod n$ $= 3^3 \bmod 33 = 27 \bmod 33 = 27$

Message decoding phase

Bob's private keys are d = 7 n = 33

Bob receives a 27 from Alice:

9. To decrypt a message C compute

$M = C^d \pmod{n}$

$= 27^7 \bmod 33 = 10460353203 \bmod 33$

= 3 (which is 'd')

(Note: Google can help with this big math.)

Eve

- Eve sees the message going by: “27”
- On its way to Bob with public key {3, 33}
- Can Eve do anything about it?

Eve

- Eve sees the message going by: “27”
- On its way to Bob with public key {3, 33}
- Can Eve do anything about it?
- Easy to factor 33
  - 11 and 3
  - With these, can calculate d
- That is why you pick a very big p & q to get a very big n.

Simple Exercise

- Working in pairs, communicate with another team
- Team A & B: Agree on a symmetric key {0-9}
- Team A:
  - Encrypt a simple numeric message
    - E.g. 12
    - Encrypt each digit separately.
  - Pass the message to Team B.
  - Decrypt the message, get the number

Advanced Exercise

- Work again in pairs
- Create asymmetric and symmetric keys
  - Create a private and public key
    - Use small primes!
  - Pick a new Caesar cipher key
- Secretly communicate the Caesar cipher key
  - Encrypt a message containing a new Caesar cipher key with the private key
    - I.e. The message itself is a small number key
  - Pass the message and public key to another team.
  - The other team decrypts the message, gets the Caesar cipher key
- Pass a message using the Caesar cipher key
    - E.g. “Help”

RSA

1. Select at random two small (typically large) prime numbers p and q
2. Compute n = pq.
3. Compute $\phi(n) = (p-1)(q-1)$
4. Select a small odd integer e that is relatively prime to $\phi(n)$
5. Compute d as the multiplicative inverse of e modulo $\phi(n)$
   - $ed \bmod \phi(n) = 1$
   - Or in Excel, iterate with =MOD((A2*B2),C2)
6. Publish the pair P = (e,n) as the RSA public key.
7. Keep secret the pair S = (d,n) as the RSA secret key.

- Public Key Encrypt/Decrypt message M: $C = M^e \pmod n$
- Private Key Encrypt/Decrypt message C: $M = C^d \pmod n$

What was this exercise?

- Used public and private keys
- To share a symmetric key
- Used the symmetric key to communicate.
- This is what you are doing in Project 4
- With a bit more complexity, this is essentially SSL

  …almost

How do I know this email came from you?

- Public key encryption can be used to provide digital signatures
- Even better that real signatures, for people can alter a document once you have signed it.
- With digital signatures, if the *document* is changed, then the signature becomes invalid.
- This is because the signature is a number based on the content of the document.

How digital signatures work

- Take your document (email message, etc)
- Calculate a hash function on it.
  - SHA1
  - MD5
- Encrypt the resulting hash value with your private key.
- The encrypted hash value is the *digital signature.*
- Send it …

How digital signatures work

- … The recipient
- Receives the document (email message, etc) and the digital signature.
- Decrypts the signature with _____ resulting in _____
- Calculate a hash of the document & compare it with the sender's hash value
- Should they be equal? Why?

How digital signatures work

- What does it mean if the hash values are not equal?
- Could Mallory change the document?
- Can Mallory change the document without changing the hash value?
- Can Eve read the document (email, etc.)?
  - What can I do about that?

Exercise

- In your pairs, since you already have public and private keys, how could you simulate digital signatures?
- Take some time to create a workable scheme, even if it has some (huge) shortcomings.

Returning to SSL

- We saw we could use:
  - Asymmetric keys (RSA)
  - to share a symmetric key (AES)
  - Then pass messages back and forth efficiently using the symmetric key
- So, I can use this scheme to send Amazon.com a message with my credit card number

BUT

- Do I really know who I've been negotiating with?
- Is it really Amazon.com?
  - Or Mallory?
- They sent me their public key to use, so if I knew that this was **really** Amazon.com's public key, then I could trust I'm working with the real Amazon.com, for only Amazon.com has the corresponding private key.
- What if someone I **trust** confirmed that this was Amazon.com's public key?
- How could I trust them?

Digital Certificates

- Issued by trusted entities
  - Company IT Department (internally)
  - VeriSign
  - Thawte
  - Lets of others
- Typically contains
  - Owner's name
  - Owner's public key
  - Expiration date
  - Name of certificate issuer
  - Serial number
  - **Issuer's digital signature**

In our class exercise

- How could we simulate digital certificates?