Cryptography and Security

Questions:

- What branch of science would have informed keeping your money safe 500 years ago?
- What branch of science informs keeping your money safe today?
- Who can and cannot see your information if it is stored in the cloud at:
  - Google
  - Apple
  - Microsoft
- What is the security debate going on in the country in which Google and Apple have taken different approaches?
  - How is this tied to their business models?
- If you type in pnc.com in your browser, how can you really trust you are communicating with PNC Bank?
- At Google, data-center-to-data-center communications did not used to be encrypted. Why wouldn't they?
- If I download software or visit a web site and get a message that the certificate is invalid, what does that mean?

Goals

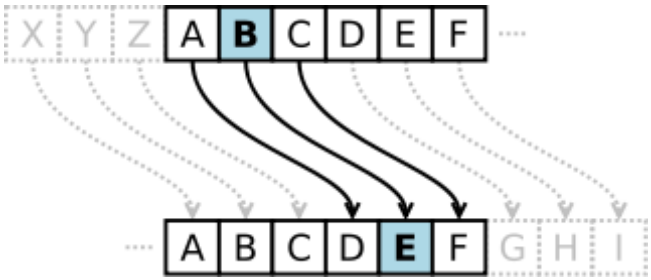
- Understand basic cryptography and security terms
- Understand security in terms of:
  - Secure web transmission
  - Authentication
    - Who are you?
  - Authorization
    - What are you allowed to do?
  - Certificates, and Digital Signatures
    - Is this document / software / transaction real?
- Have a basic understanding of the underlying theory and math behind web security.

2 kinds of cryptographic systems

- Symmetric key
  - Simple e.g. Caesar cipher
  - Most prevalent: AES
- Asymmetric key
  - Also known as Public Key (or Private / Public Key)
  - Most prevalent: RSA

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

Caesar cipher

- Secret key algorithm
  - The sender and the receiver share a secret key
- Symmetric algorithm
  - Trivially-related keys are used to encode and decode the message
    - Trivially-related: uses the same key, or keys require only a simple transformation.
    - E.g. Caesar cipher: (English) symmetric key is 26-key

Brute Force Attack

- One way to discover the plaintext is to exhaustively try every possible key.
- Is the algorithm susceptible to being broken by exhaustively trying possible keys?
- Is the Caesar cipher susceptible to brute force attack?

Brute Force Attack Countermeasures

- Use extremely large keys
  - An 8 bit key has 2^8 possible keys
    - 256
  - A 16 bit key has 2^{16} possible keys
    - 65,536
  - A 32 bit key has 2^{32} possible keys
    - 4,294,967,296
  - A 256 bit key has 2^{256} possible keys
    - $1.15792089 \times 10^{77}$

How can we "mess up" the data better?

- Cryptography is essentially trying
  - to "mess up" the original data as much as possible
  - so that someone else cannot find the original
  - but be able to get the original data back with a key
- Encryption = "mess up"

- So how can we "mess up" the data better than the Caesar cipher does?
  - The blocks of plaintext are one character.
  - There is only so much you can do to one character.
  - So how about encrypting blocks of data?

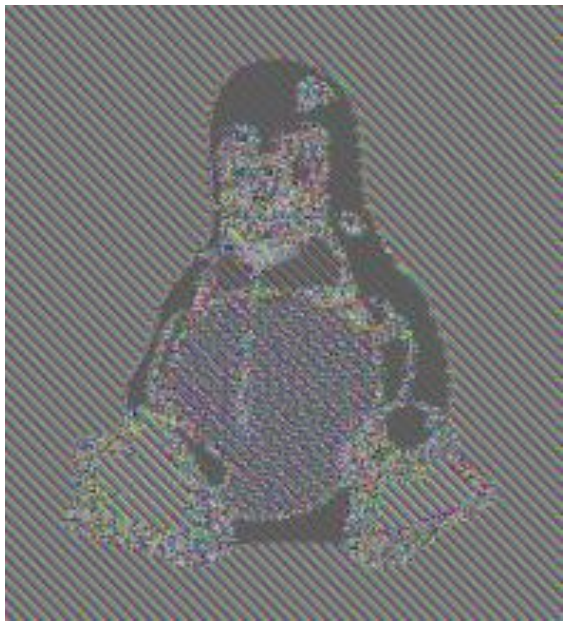
Block cipher

- Symmetric key cipher
- Works on *blocks* of text
  - E.g. 128 bit blocks
- Simple Caesar example: Two characters, Key: 1
  - cake (c=3, a=1, k=11, e=5)
  - 0011000110110101
  - 0011001010110110

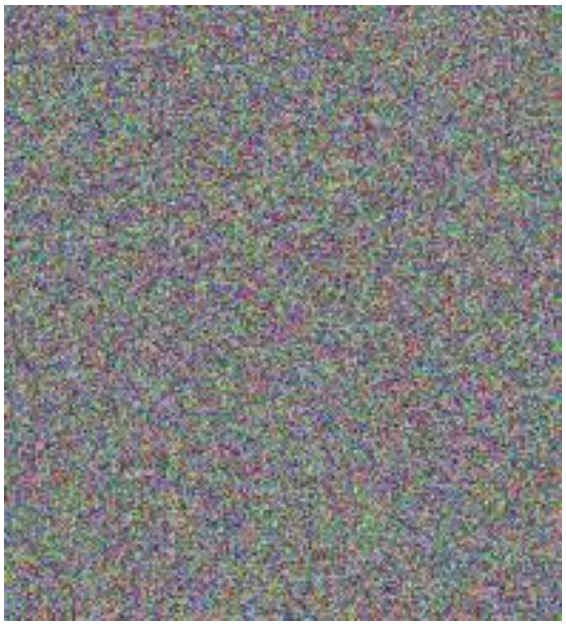
Shortcomings of block cipher

- If blocks' plaintext are identical, then their ciphertext will also be identical.
- One way to avoid repeated similarity is to mess up the current block with info from all prior blocks.
  - I.e. XOR current block with prior block, then encrypt

Original



Block Cipher



Block Cipher Chaining

Image source: http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation

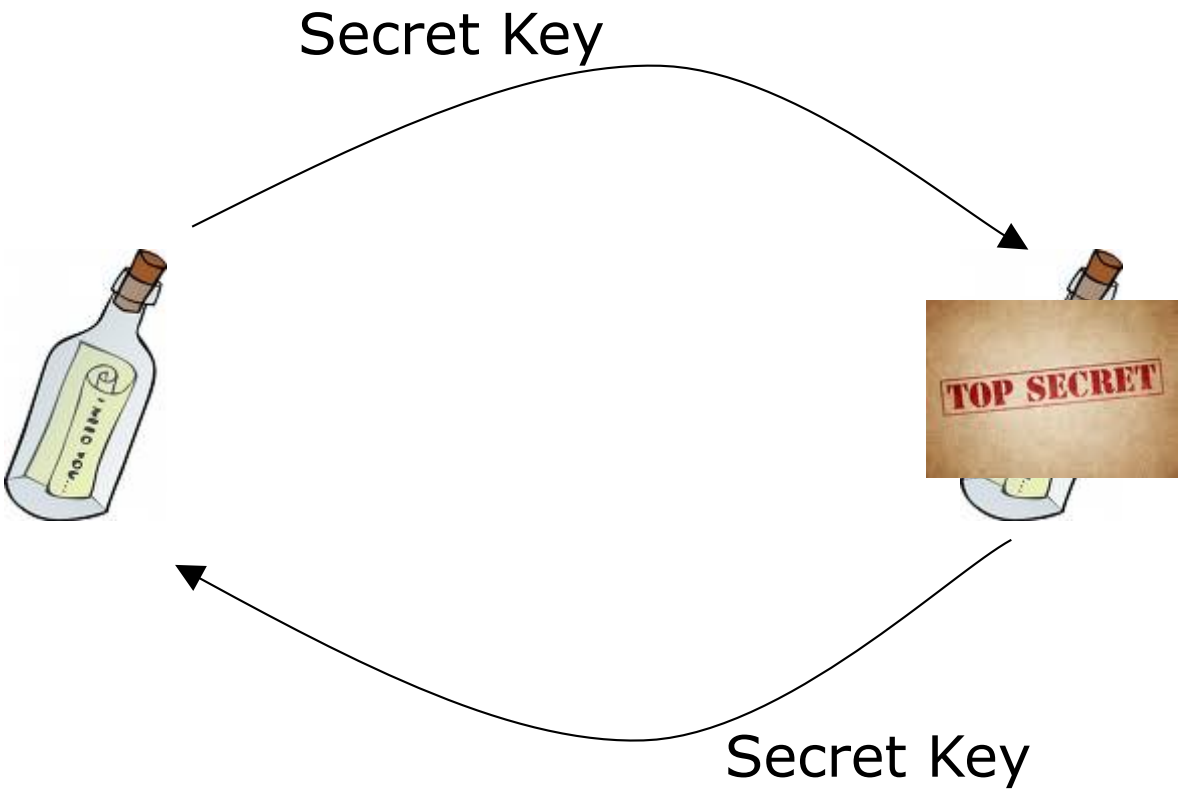
Common Symmetric Algorithms

- Advanced Encryption Standard (AES)
  - Very complex chaining block cipher
  - Adopted by US National Institute of Standards
    - Replaced the former standard: DES
      - Data Encryption Standard – 56 bit key block cipher
- To check what your browser is using:
  - In Chrome, browse to some https://… site
  - Look at the Security tab in Dev Tools

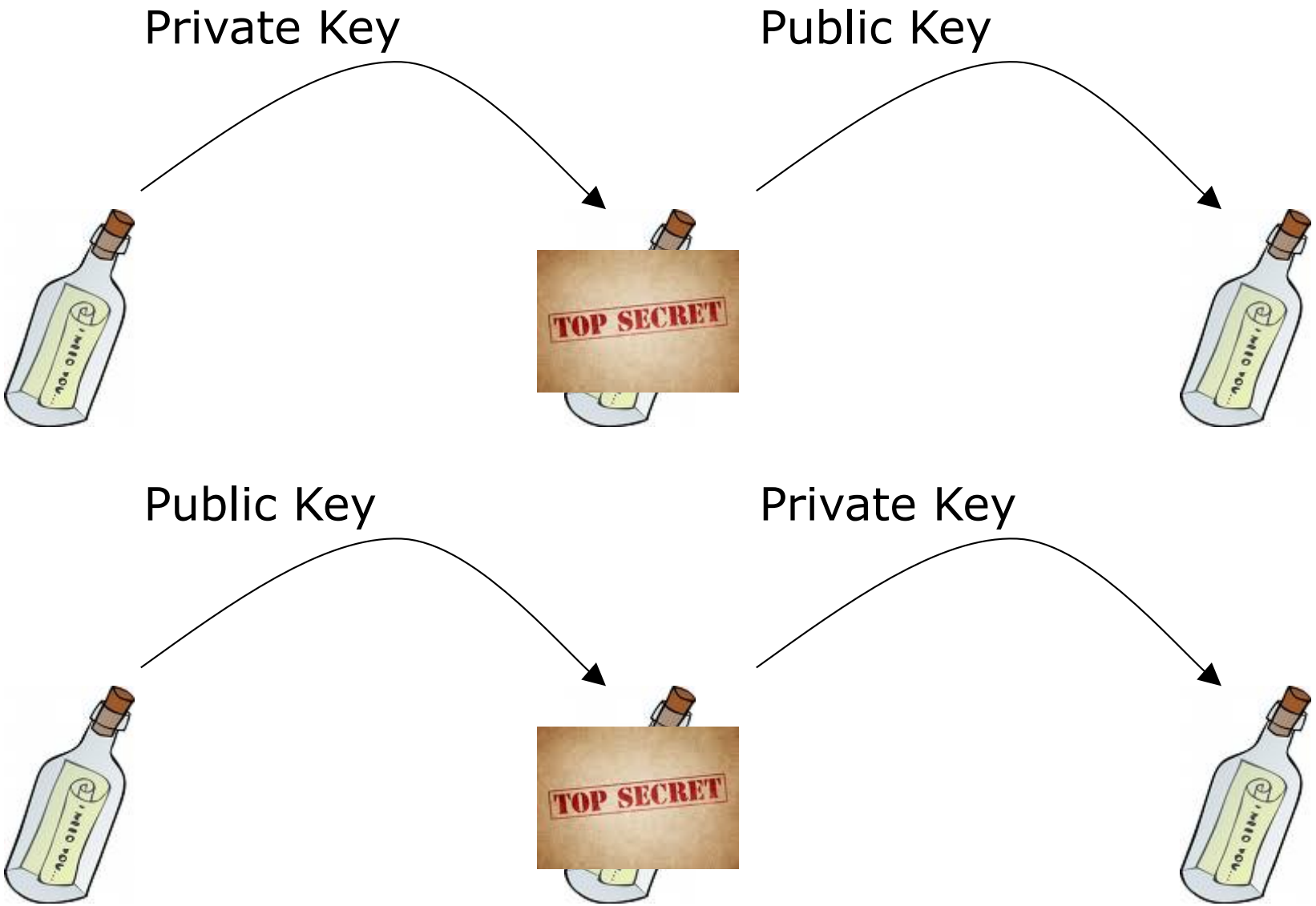
Symmetric vs Asymmetric algorithms

- 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

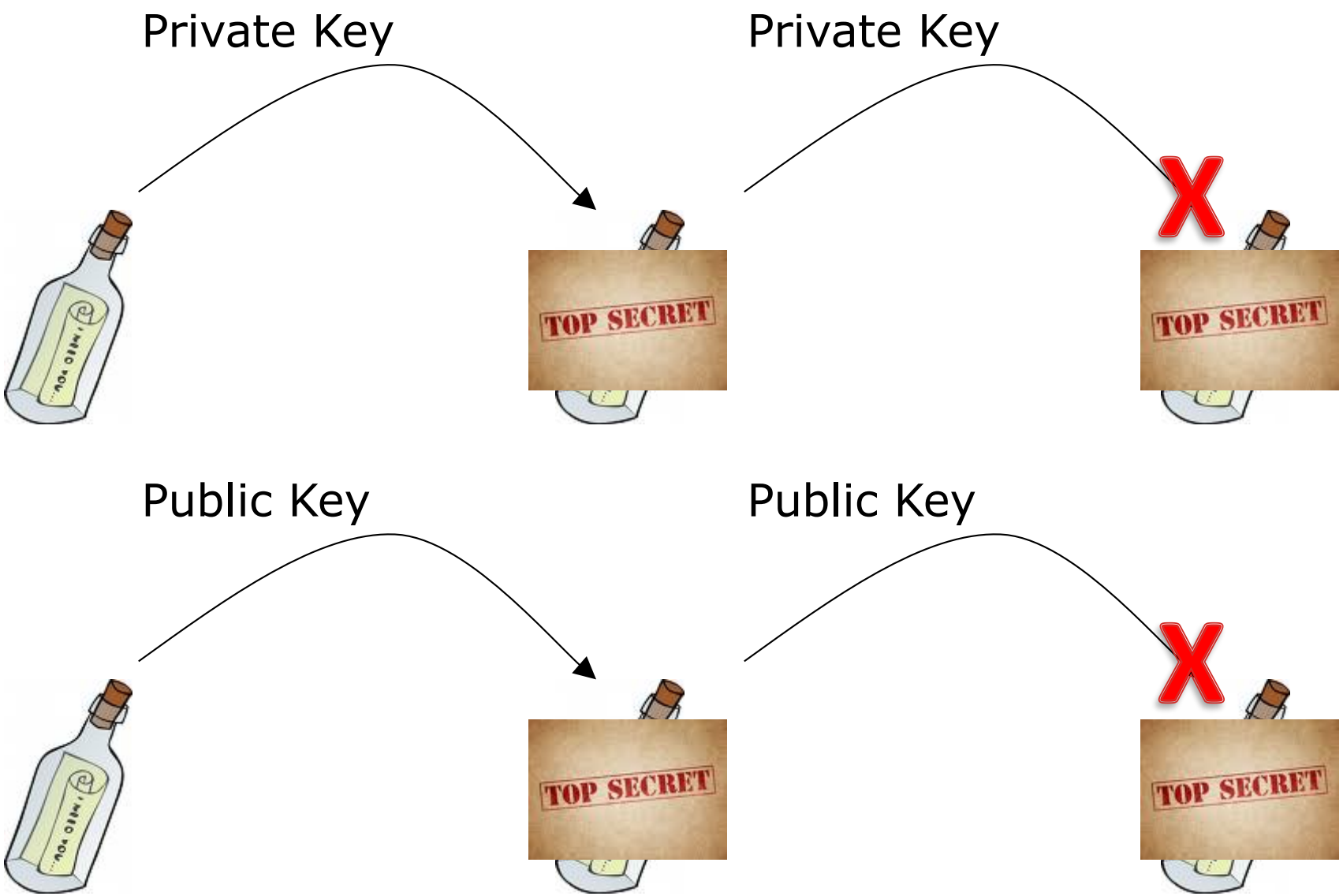
Symmetric



Asymmetric


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

Asymmetric


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

Public Key Cryptography

- Uses asymmetric keys
- Single public key
  - Pass out to the world
- 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.

RSA

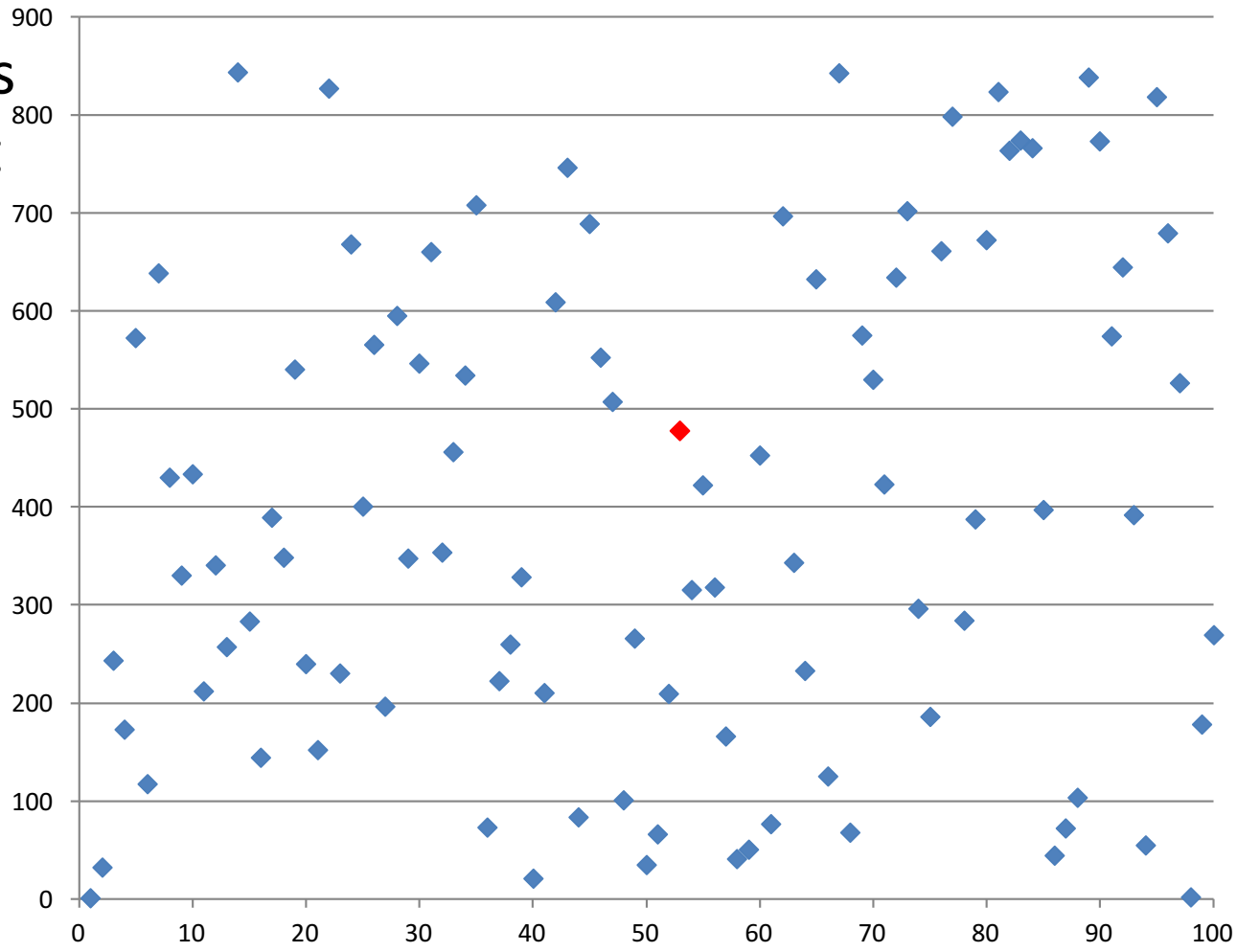
- RSA is a common public key encryption algorithm
  - Named for its authors: Rivest, Shamir, & Adleman
- Developed in 1977
- Based on the mathematics of large prime numbers
  - If you have the numbers, you can use them to encrypt messages
  - If you don't have the numbers, it is infeasible to guess them

The RSA algorithm

- There is a nice tutorial tool at:
  - https://www.cs.drexel.edu/~jpopyack/IntroCS/HW/RSAWorksheet.html
- You can practice by using prime numbers you can find at:
  - http://en.wikipedia.org/wiki/List_of_prime_numbers
- E.g. 23, 37

Plot of 1-100 encrypted with RSA key {e:5,n:851}

- RSA keys have an exponent & modulus
- You encrypt/decrypt with $y = x^5 \% 851$
- Graph is encryption of 1 to 100.
  - See a pattern?
  - Can you predict the value at 101?
- Unpredictability is its strength
- Red dot is at (53, 477)
- Corresponding key is {e:317,n:851}
  - Same exponent, different modulus
- $477^{317}\%851=53$



53 -> public key -> 477
477 -> private key -> 53

Review so far…

- What are the two types of cryptographic systems?
- Which is AES?
- Which is RSA?
- Which has a single key?
- Which has a pair of keys?
- If you have one AES key, can you guess the other?
- If you have one RSA key, can you guess the other?
- Which of the following key lengths (in bits) are susceptible to brute force attack?
  - 2, 16, 32, 256, 1024
- (T/F) If data is encrypted with a public key, then someone else can decrypt it with the public key?

Two Roles of Private and Public Keys

1. Encryption / Decryption
   - Public key
     - Used by others
     - To encrypt a message intended only for you
   - Private key
     - Used by you
     - To decrypt a message originally encrypted by your public key

2. Signing / Verification
   - Private key
     - Used to sign a document so that others can verify the source
   - Public key
     - Used to verify that a signed document was signed by you.

Cryptographic protocols

- Cryptographic protocols build on the use of cryptographic algorithms
- Two prominent protocols:
  - Transport Layer Security (TLS) (newer)
  - Secure Socket Layer (SSL) (older)
- Both are application-level protocols, that work above the transport layer (especially TCP) to provide safe end-to-end communication.
- Often folk will refer to secure communication as "over SSL" regardless of whether TLS or SSL is being used.

Secure email & web

- SMTP and IMAP (email protocols) can work above TLS or SSL to provide secure email transmission
- HTTPS is HTTP over TLS/SSL to provide secure web communication

Symmetric vs Asymmetric algorithms

- Symmetric (e.g. AES)
  - Fast
  - Difficult to distribute and and keep keys secure
- Asymmetric (e.g. RSA)
  - 100 to 1000 times slower
    - Why? Because raising each block to a ginormous exponent is slow.
  - Can allow for public keys
- TLS / SSL uses the best of both worlds:
  - Use asymmetric keys to exchange symmetric keys at the beginning of a conversation
  - Symmetric keys will have the lifespan of that conversation

TLS / SSL Protocol Handshake

- Start with RSA Public / Private keys
  - Ask Amazon for their Public key
  - Amazon replies with their Public key
- Generate a 128 bit (or bigger) random number
  - This is your "session" new AES key
  - Encrypt the new AES key with the Amazon Public Key
  - Send the encrypted key to Amazon
    - Notice you are sending an AES key encrypted with a RSA key
- Amazon receives the encrypted key
  - They (and only they) can decrypt the AES key with their RSA Private key
  - They now have the AES key you created for this session
- Amazon and your browser communicate by encrypting and decrypting all messages with the same AES 128 bit random-number key.
  - At the end of this session, both forget the AES key

Two Roles of Private and Public Keys

1. Encryption / Decryption
   - Public key
     - Used by others
     - To encrypt a message intended only for you
   - Private key
     - Used by you
     - To decrypt a message originally encrypted by your public key

2. Signing / Verification
   - Private key
     - Used to sign a document so that others can verify the source
   - Public key
     - Used to verify that a signed document was signed by you.

E.g. Verifying authenticity

- How can you guarantee to someone that a document you sent them is from you, and has not been changed?
- How can you guarantee that the software you are using came from Microsoft?
- You want to keep the document / software / image / etc. viewable and usable, but just want a scheme by which others can verify its authenticity.

Digital Signatures

- Public key encryption can be used to provide digital signatures to validate authenticity
- Digital signatures are better that real signatures, for people can alter a paper 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.
  - Or more specifically, on the *hash value* of a document.

Hash Functions

- Input data (application, document, picture, etc.)
- Outputs a large, but fixed-size number.
  - e.g. 128 bits or 160 bits
- Any intentional or accidental change in the file will change its resulting hash value.
- The file being encoded is called the “message”
- The output is called the
  - Hash value
  - Message digest
  - Or simply, digest

Properties of a good hash function

1. For any message, the hash value is easy to compute.
2. It is infeasible to create a new message that has a given hash value
   - You can't work backwards to create an imposter.
3. It is infeasible to modify a message in any way without changing its hash value
   - The most minor changes still change the hash value.
4. It is completely unlikely that two documents will have the same hash value
   - So you don't have to worry that you just happen upon another document with the same hash value

Common Hash Functions

- SHA
  - Designed by the National Security Administration (NSA)
  - SHA-1
    - 160 bit digest
      - In Feb 2017, a deliberate collision was demonstrated*
      - Breaking property #2 on the previous slide
  - SHA-2
    - A family of related hash functions
    - SHA-256 has a 256 bit digest (this is currently used in Chrome)
- MD5
  - 128 bit digest
- Both SHA-256 and MD5 are often represented as strings of hex digits

*Google announced that they had demonstrated a SHA-1 collision on 2/23/17: https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html

Demonstration of Hash Functions

- openssl dgst -sha256 invoice.txt
- Change one character
  - Compare resulting hash value

Is a hash function encryption?

- Can a hash function be used to encrypt a message?

How digital signatures work

- Take your document (email message, etc)
- Calculate a hash function on it.
  - SHA256
- 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 the sender's public key resulting in the hash value of the message.
- 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?

Signing HTTP requests

- HTTP requests to 3rd party web services often need to be signed in order to establish securely:
  - What application is making the request
  - For which user the request is being made on behalf of
  - Whether the application has the authorization to make the request on behalf of the user
  - Whether the request has been tampered with in any way in transit
- HTTP requests can be digitally signed, typically using two keys:
  - The application's API key
  - A specific users access token
  - (These both go by different names depending on the API)

Returning to TLS

- 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
- This is essentially TLS
- So, I can use this scheme to safely send Amazon.com a message with my credit card number, correct?

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?

Digital Certificates

- A Digital Certificate is a document that provides information about an organization
  - Most importantly, its public key
- And the Digital Certificate is digitally signed by some trusted party.

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
- E.g. Blackboard Digital Certificate

DigiNotar

Fraudulent Google credential found in the wild

Did counterfeit SSL cert target Iranians?

World ostracizes firm that issued bogus Google credential

DigiNotar says it was breached ... but little else

Bankruptcy report, 31 October 2011

Start

Withdraw

root certificates

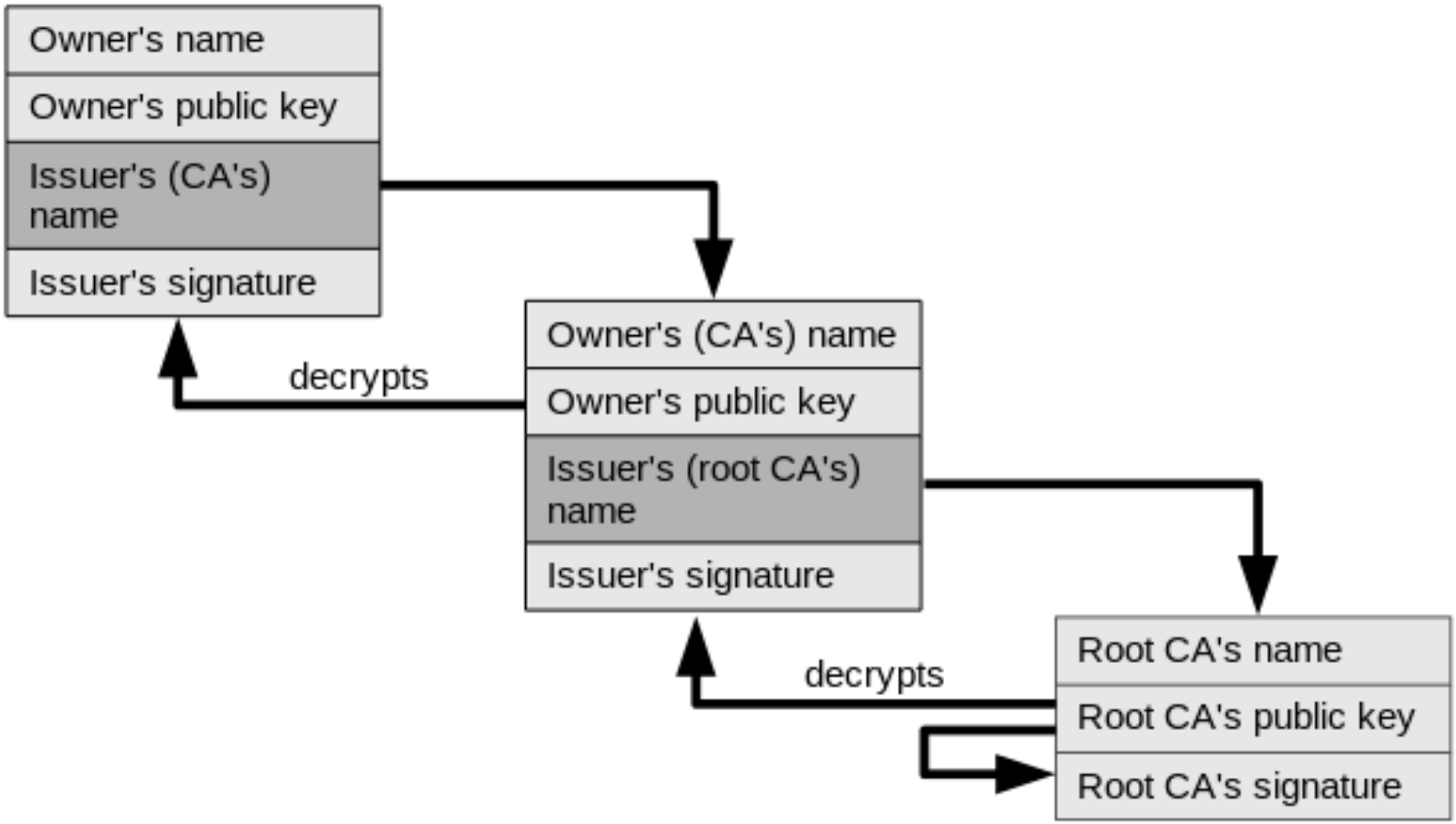
BAPI

Bankruptcy
Company DigiNotar BV
Number F 11 415
District Court of Haarlem
Date 09/20/2011

Putting it all together

- What happens when you log onto Amazon?
  - SSL / TLS handshake
    - Shared public key
      - Digital certificate to authenticate identity
        - » hash
        - » Certificate authority public key
    - Generate and share a symmetric key
    - Continue communication using Advanced Encryption Standard (AES)
  - Authenticate with name and password
    - Passed via secure SSL
    - If ok then a cookie can be tied to a session in the server that maintains that you are authenticated (at-main?)

Chain of Trust



By Yanpas - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=46369922

Review Questions

- The “padlock” on a browser means _____?
- Does the padlock means you are logged in?
- If you are logged into Amazon.com, which are true?
  - Your browser received Amazon’s public key
  - Your browser used a certificate authority to validate Amazon’s public key
  - Amazon validated your public key with a certificate authority
  - Amazon provided an encrypted hash value of the certificate authority’s digital certificate.
  - RSA might have been used between your browser and Amazon.
  - Your browser and the certificate authority are sharing a secret key
  - Your browser and Amazon are sharing a secret key.

Review Questions

- RSA is a {protocol or encryption algorithm}.
- AES is a {protocol or encryption algorithm}.
- SSL is a {protocol or encryption algorithm}.
- A hash function is an {encoding or encryption} algorithm.

Custom Authentication

- You can create your own authentication
- Require a secure channel (e.g. SSL, TLS)
- Create a login page
  - User ID and Password sent to the server
  - Store in mongoDB
    - userID as key
    - password

Storing passwords

- Storing raw passwords on a server is dangerous
  - Risk of nosey (disgruntled) employees accessing them
  - Risk if your server is broken into and file stolen
- Therefore, don't store raw passwords
  - Rather, store the hashed value of the password
    - http://nodejs.org/api/all.html#all_crypto_createhash_algorithm
  - Then with a user logs in, hash the submitted password and compare it with the stored value

More attacks…

- If your stored hashed passwords get stolen, they can still be susceptible to attacks:
  - Dictionary attack
    - Generate a large set of possible passwords (e.g. dictionary words), hash each, and test against the saved hashed password
  - Rainbow table attack
    - Create a lookup table of hash values and the original passwords for a very large set of possible passwords

Defense: add salt...

- Randomly *salt* each password
- When storing new credentials
  - Generate a new random number (the *salt*)
  - Append the number to the password string
  - Hash the new, longer string
  - Store with the user profile:
    - The hashed (password+salt)
    - The salt
- When validating credentials
  - Get the user's salt from their profile
  - Append the salt to the submitted password
  - Hash the resulting string
  - Compare the hash value to the stored hashed value
- Note:
  - If you are doing this for an actual site, research current best practices for the attacks are always advancing

Authorization

- Authorization is different than Authentication
- Authentication establishes identity
- Authorization establishes access rights
- Many sites have their own authorization schemes
- Many use the open scheme: OAUTH