AP/ITEC 2210 3.0 A: System Administration Fall 2023

Instructor: Jamon Camisso
ITEC 2210 Chat: mattermost.itec2210.ca
Email: jamon@yorku.ca
Website: https://eclass.yorku.ca/

Date/Time: Wednesday, 19:00-22:00
Location: Zoom / ACE 003
Office hours: Via Mattermost any time

Class 7 - Encryption, Security

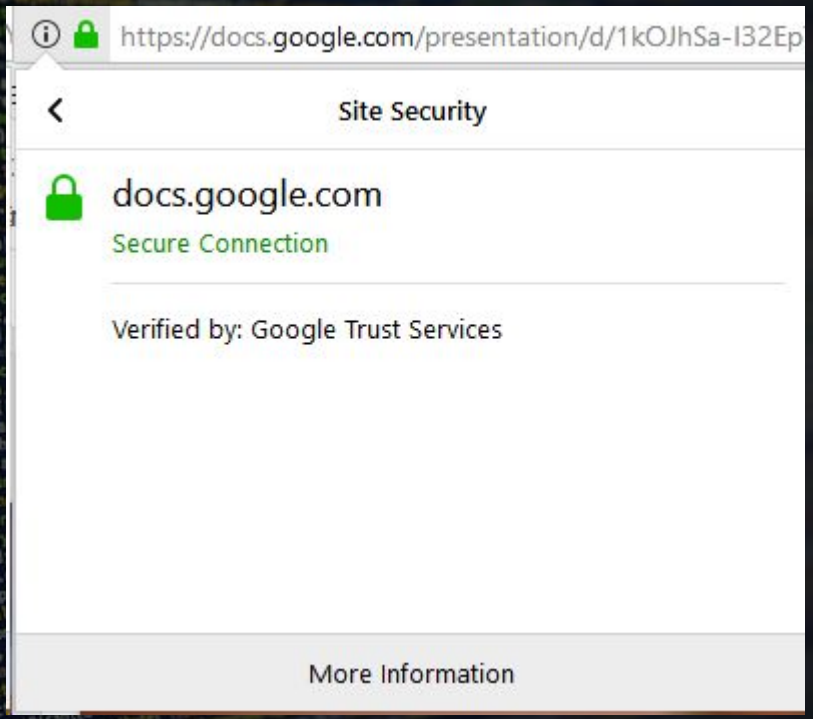
- Final exam:
  - Date - location TBD
  - In person
  - Open book
  - Cumulative

Class 7 - Encryption, Security

- Backups review:
  - The importance of offsite backups:

https://www.reuters.com/article/us-france-ovh-fire/fire-breaks-out-in-ovh-building-in-strasbourg-france-idUSKBN2B20NU

Class 7 - Encryption, Security



Class 7 - Encryption, Security

- First some notes:
  - I *will* be oversimplifying or eliding details of some algorithms or maths in this Class - good crypto is hard!
  - However, as an SA the principles are important, regardless of implementation details, and I'm confident explaining those, so here goes!
  - The nature of cryptography (and security) is it has to be nearly perfect, or it will eventually fail to a smarter or better financed adversary

Class 7 - Encryption, Security

- Cryptography principle 0
  - **Never design your own cryptography system**
  - You will get it wrong, someone will find a weakness
  - Idea is affectionately called **Schneier's Law**
  - An example analogy:
    - If you wouldn't design and strap yourself to your own rocket, you probably shouldn't design your own cryptosystem

Class 7 - Encryption, Security

- Cryptography resources
  - Menezes, A. J., C., V. O., & Vanstone, S. A. (2001). *Handbook of applied cryptography*. Boca Raton: CRC Press.
    - Available free: **https://cacr.uwaterloo.ca/hac/**
  - **https://crypto.stackexchange.com** - It is the best kind of pedantry The links to primary and secondary sources are especially helpful
  - **NIST Federal Information Processing Standards (FIPS)**
  - **Wiki.openssl.org**
  - To prove crypto is 'fun' see **Schneier Facts**

Class 7 - Encryption, Security

- Cryptography resources
  - Full course in breaking crypto systems (in order to learn about them in a practical way) here: https://cryptopals.com/

Class 7 - Encryption, Security

- **Cryptographic attacks**
  - I'm not going to cover them since there are so many, but here are some side-channel (non-cryptographic) attacks that can be more effective than finding mathematical weaknesses in a crypto system
    - Black-bag cryptanalysis - physical theft
    - Man-in-the-middle attack - eavesdropping
    - Power analysis - what it says, crazy cool
    - Replay attack - replay encrypted data, e.g. WEP, WPA Handshake
    - Rubber-hose cryptanalysis - physical or psychological torture
    - Timing analysis - using CPU timings to glean information

Class 7 - Encryption, Security

- **Cryptography basics**
  - Alice, Bob, Eve are the main actors who we'll encounter
  - Typical scenario is Alice and Bob want to communicate privately, and Eve wants to know what they're saying
  - **Plaintext** (message) is what they're communicating
  - **Ciphertext** is the encrypted version of the message

Class 7 - Encryption, Security

- **Cryptography basics**
  - **Key** is the secret that is mixed with plaintext in an encryption algorithm to produce a ciphertext
  - **Keyspace** is the size of all possible key values, e.g. $2^{\wedge 2048}$
  - **Key schedule** is an algorithm that expands a key from say, 128 bits to thousands of bits for later use in an encryption operation
  - **Block cipher** algorithms operate on chunks of data at a time
  - **Stream cipher** algorithms operate on bits of data at a time

Class 7 - Encryption, Security

What does encryption Look like?

From HAC ch1 p.13

E - encryption function
D - decryption function
m - message
c - ciphertext

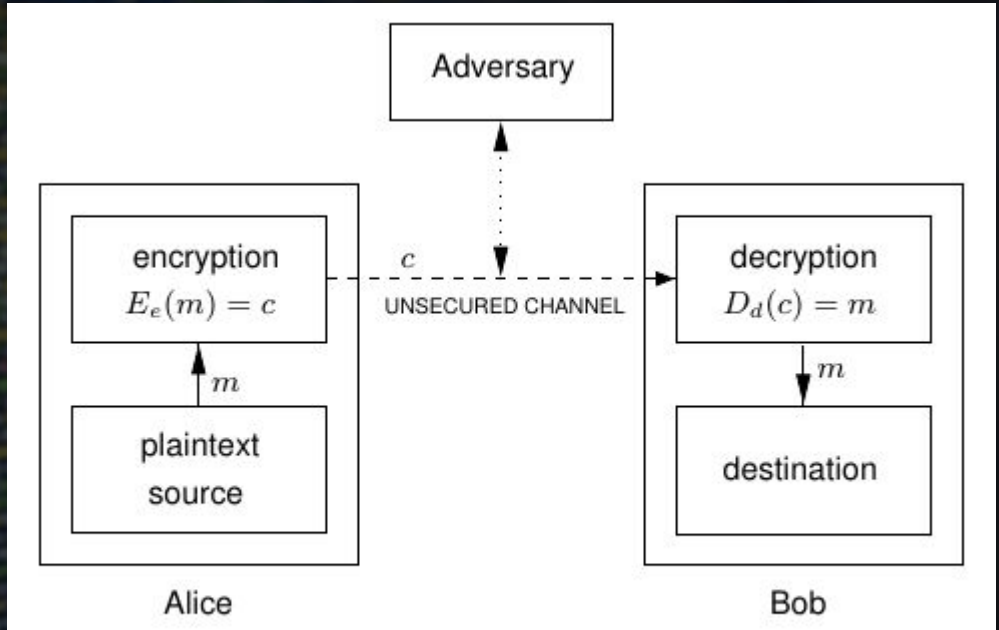


Figure 1.6: *Schematic of a two-party communication using encryption.*

Class 7 - Encryption, Security

- Cryptographic **operations**:
  - **Substitution:** A=C, B=D, C=E, D=F - Caesar cipher
  - **Transposition:** plaintext is reordered
  - Bitwise operation: bits are **XOR'ed**
  - **Round**: composition of operation(s) on a plaintext

Class 7 - Encryption, Security

- Cryptographic **operations** (from HAC, ch1, p.20):
  - **Substitution** in a round adds **confusion** to a ciphertext
    - Idea is to make the relationship between encryption key and ciphertext appear complex
  - **Transposition** adds **diffusion** to a ciphertext
    - Idea is to rearrange and spread out non-uniform bits in a message to statistical patterns, common letter combinations

Class 7 - Encryption, Security

- Cryptography basics
  - XOR:
    - 1 XOR 1 = 0
    - 1 XOR 0 = 1
    - 0 XOR 1 = 1
    - 0 XOR 0 = 0
  - Tie in to last week's RAID5/6 Class and how XOR & parity works: https://igoro.com/archive/how-raid-6-dual-parity-calculation-works

Class 7 - Encryption, Security

- Cryptographic **Hash functions**
- Map an **input** of any arbitrary length binary string to a fixed length unique binary string (**hash**) that represents the input
- Take the string `itec2210.ca` and run it through a hashing algorithm (use a terminal on your VM):
  - echo 'itec2210.ca' |md5
    b44f302d2e8f77c54455ce7b7b6f08ba
  - python -c 'print("A"*100)' |md5
    0d6aa8e34b2af058d6e4a27d7ff511dd

Class 7 - Encryption, Security

- Cryptographic **Hash functions**
- python -c 'print("A"*100)' |md5
  0d6aa8e34b2af058d6e4a27d7ff511dd
- No matter how long the input, output is always 128 bits
- Likewise with SHA1/2/3, e.g:
- python -c 'print("A"*100)' | shasum -a 256
  f76262cff1f14a9eb92d57ac0de2b8dc431f3c0fe3e88f9394140f68024066b7 -

Class 7 - Encryption, Security

- Cryptographic **Hash functions**
- Now repeat the same input to the algorithm:
- python -c 'print("A"*100)' | shasum -a 256
  f76262cff1f14a9eb92d57ac0de2b8dc431f3c0fe3e88f9394140f68024066b7 -
- python -c 'print("A"*100)' | shasum -a 256
  f76262cff1f14a9eb92d57ac0de2b8dc431f3c0fe3e88f9394140f68024066b7 -
- python -c 'print("A"*100)' | shasum -a 256
  f76262cff1f14a9eb92d57ac0de2b8dc431f3c0fe3e88f9394140f68024066b7 -

Class 7 - Encryption, Security

- Cryptographic **Hash functions**
- Notice how the output never changes?
- The same input will always produce the same set of output bits
- Enables checking things like:
  - File contents are unmodified
  - Encrypted data is intact
  - Email message signatures
  - Password entries in databases
  - TLS browser encryption

Class 7 - Encryption, Security

- Cryptographic **Hash functions**
- Map an **input** of any arbitrary length binary string to a fixed length unique binary string (**hash**) that represents the input
- It should be difficult to find two separate inputs that map to the same hash output (a **collision**) (SHA-1 bruteforce odds being $2^{\wedge 80}$, or $1.2x10^{\wedge 24}$)
- 1,200,000,000,000,000,000,000,000
- Moreover, given a hash, it should be infeasible to work backwards and find an input: algorithm is not reversible
- But wait:

Class 7 - Encryption, Security

- Cryptographic **Hash functions**
- Common hashing functions you'll encounter:
  - MD5 - **BROKEN**
  - SHA-1 - **BROKEN**
  - **SHA-2** - current standard but getting weaker against attacks
  - **SHA-3** - recently released (as of 2015), variable length digests
  - **Bcrypt** - can be made to run arbitrarily slow to prevent bruteforce
  - Hash functions have lifetimes of decades at best
  - As computer power grows, speed of finding collisions does too

Class 7 - Encryption, Security

- Cryptographic **Hash functions**
- Used extensively to create digital signatures, like SSL/TLS certificates, document signatures, email encryption
- Also used to ensure data integrity, like file transfers over untrusted networks, e.g. Debian, Ubuntu, RedHat, Fedora, software repositories
  - http://centos.mirror.iweb.ca/7/extras/x86_64/repodata/repomd.xml.asc
  - http://centos.mirror.iweb.ca/7/extras/x86_64/repodata/repomd.xml
- Note: signature is a hash of the data, which is then encrypted with the private key - later we'll see how the public key decrypts the signature

Class 7 - Encryption, Security

- **Symmetric** and **Asymmetric** encryption

- **Symmetric** - both parties share a secret encryption key
  - Can be **block** or **stream** based
  - **AES** is the current generally accepted standard for symmetric (**block**) encryption
  - **AES** encryption algorithm uses 16 byte blocks
  - Example would be a passphrase on an encrypted disk volume

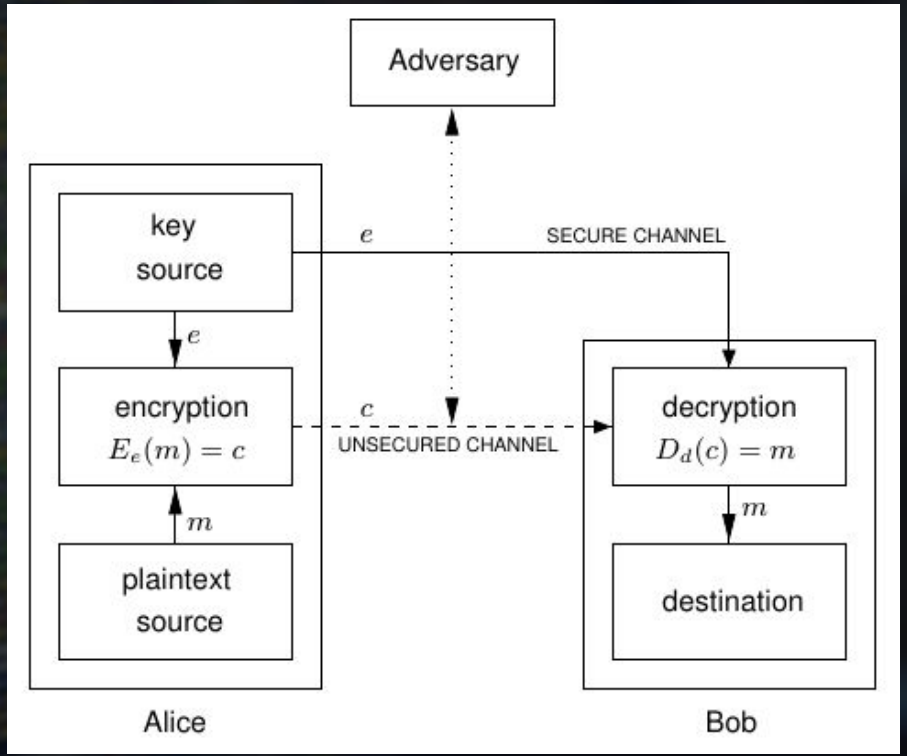
  - **RC4** is an example of a (broken) **stream** cipher
    - WEP encryption used RC4 in Initialization Vector (IV)
  - Many block ciphers can be used in stream mode

Class 7 - Encryption, Security

- **Symmetric** and **Asymmetric** encryption
- **IV aside** - visit wikipedia article, it is very helpful
  - Initialization vector or starting value is used to ensure that repeated use of a key on duplicate plaintext results in different ciphertext
  - Usually IV is mixed with key in the algorithm's state machine as part of a key schedule or keystream
  - In most block cipher modes IV will be stored with a block of ciphertext

Class 7 - Encryption, Security

- **Symmetric** encryption
- HAC ch1 p.16



Class 7 - Encryption, Security

- **Symmetric Encryption - AES**
- Designed by extensive consultation with cryptography community and standards bodies
- Intent was to replace DES (2^{56} bit key combinations, 7.2×10^{16})
- AES has:
  - **2^128** or 3.4×10^{38} possible keys (128-bit);
  - **2^192** or 6.2×10^{57} possible keys (192-bit); and
  - **2^256** or 1.1×10^{77} possible keys (256-bit)

Class 7 - Encryption, Security

- **AES**
  - “Assuming that one could build a machine that could recover a DES key in a second (i.e., try 2^{55} keys per second)
  - then it would take that machine approximately 149 thousand-billion (149 trillion) years to crack a 128-bit AES key.
  - To put that into perspective, the universe is believed to be less than 20 billion years old” (Since AES was published, WMAP determined it is likely 13.77 ± 0.059 billion years)

Class 7 - Encryption, Security

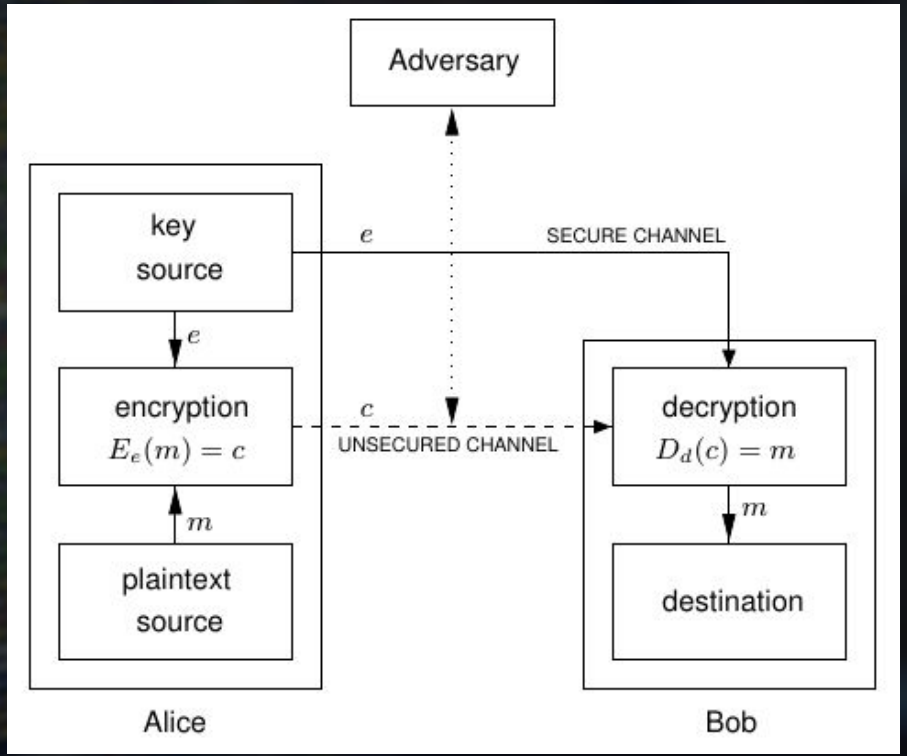
- **AES**
- So what does AES actually look like?
- https://formaestudio.com/rijndaelinspector/archivos/Rijndael_Animation_v4_eng-html5.html
- It is worth taking your time with it

Class 7 - Encryption, Security

- **Symmetric** and **Asymmetric** encryption
- **Asymmetric (AKA Public Key)**
  - Designed around a pair of large prime numbers (**key pair**) and some modular arithmetic, with one number being private, and the other public
  - The public key is used to encrypt a value, which only the private key can decrypt
  - Example would be **RSA** algorithm, used for **PGP/GPG**
  - Think of it like a locked mailbox where anyone can put a letter in, but can't take a letter out without the key

Class 7 - Encryption, Security

- **Recall symmetric** encryption
- HAC ch1 p.16



Class 7 - Encryption, Security

- **This is asymmetric** encryption
- HAC ch1 p.28

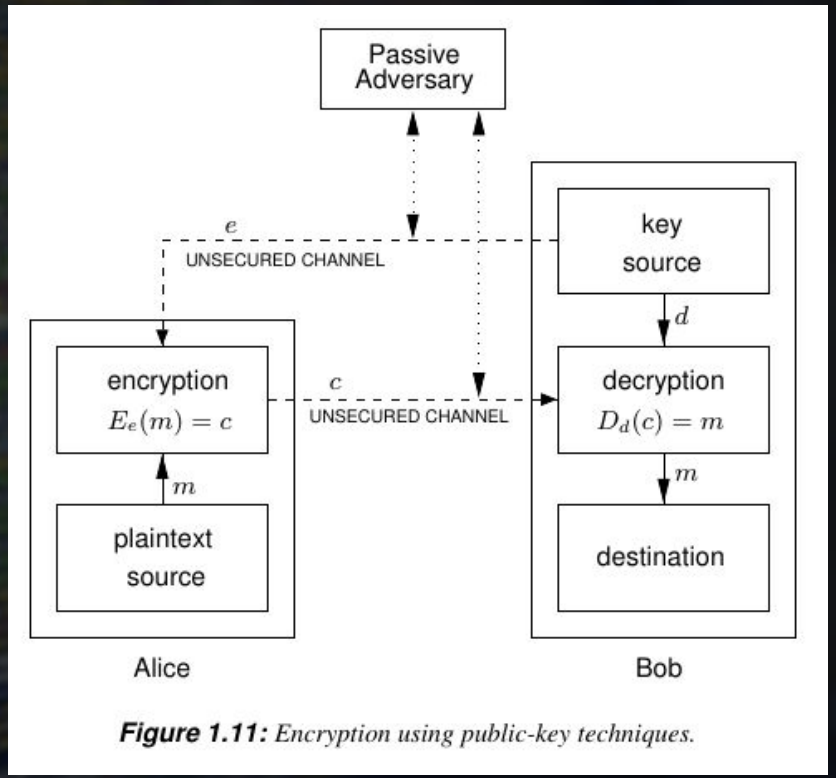
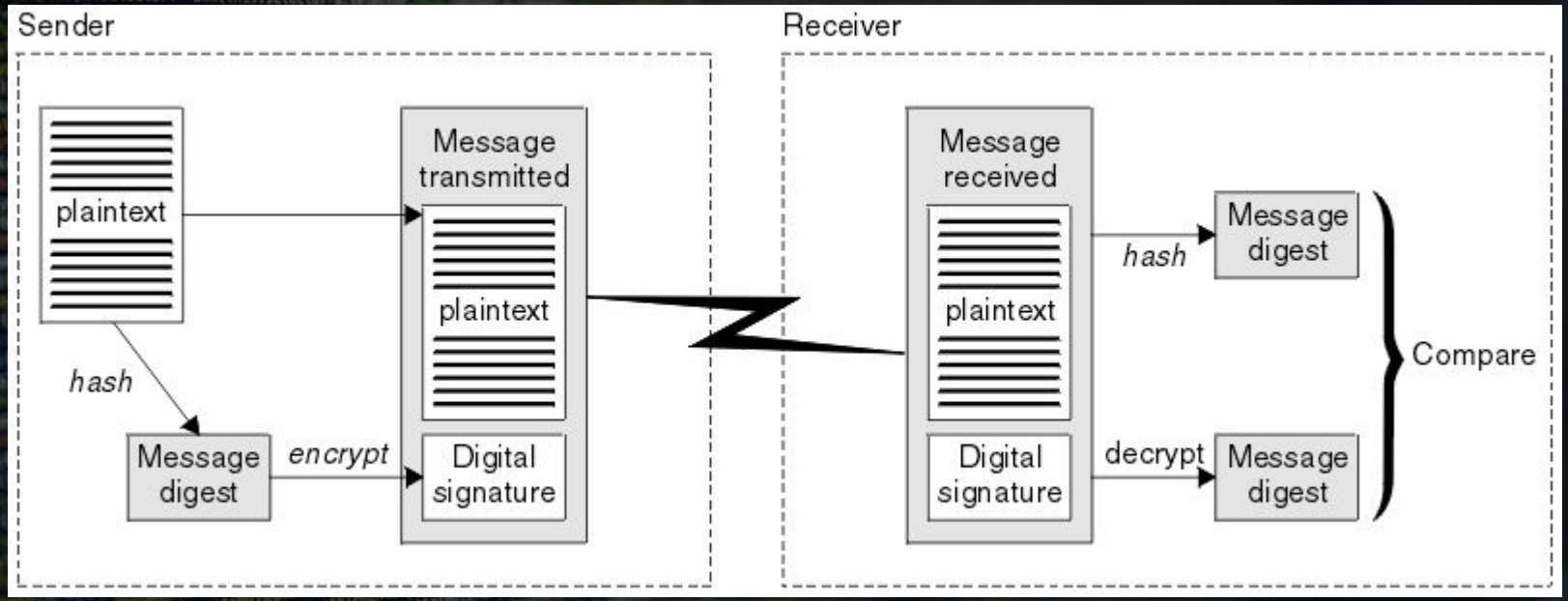


Figure 1.11: *Encryption using public-key techniques.*

Class 7 - Encryption, Security

Class 7 - Encryption, Security

- Cryptographic hash used for digital signature



https://www.ibm.com/support/knowledgecenter/SSFKSJ_7.1.0/com.ibm.mq.doc/sy10520_.htm

Class 7 - Encryption, Security

- **Asymmetric encryption**
  - Public key can encrypt a message that can only be decrypted with the private key
    - Used for secure communication
  - Private key can encrypt a message that can only be decrypted with the public key
    - Used for authentication and signatures
  - Bear with me here, the idea is to see that the following works
  - Recall modulus arithmetic: *10 mod 3 ≡ 1*

Class 7 - Encryption, Security

- **Asymmetric RSA encryption** (from HAC ch8 p.286)
- Algorithm components are:
  - 1. Generate two large random (and distinct) primes p and q, each roughly the same size
  - 2. Compute $n = pq$ and $\varphi = (p - 1)(q - 1)$
  - 3. Select a random integer e, $1 < e < \varphi$, such that $gcd(e, \varphi) = 1$ (e is coprime with phi, that is, their only common divisor is 1)
  - 4. Use the extended Euclidean algorithm (Algorithm 2.107) to compute the unique integer d, $1 < d < \varphi$, such that $ed \equiv 1 \pmod{\varphi}$
  - Otherwise expressed as $ed / \varphi = 1$
  - 5. A's public key is (n, e); A's private key is d

Class 7 - Encryption, Security

- **Asymmetric RSA encryption** (from HAC ch8 p.286)
- Example:
  - https://simple.wikipedia.org/wiki/RSA_(algorithm)

Class 7 - Encryption, Security

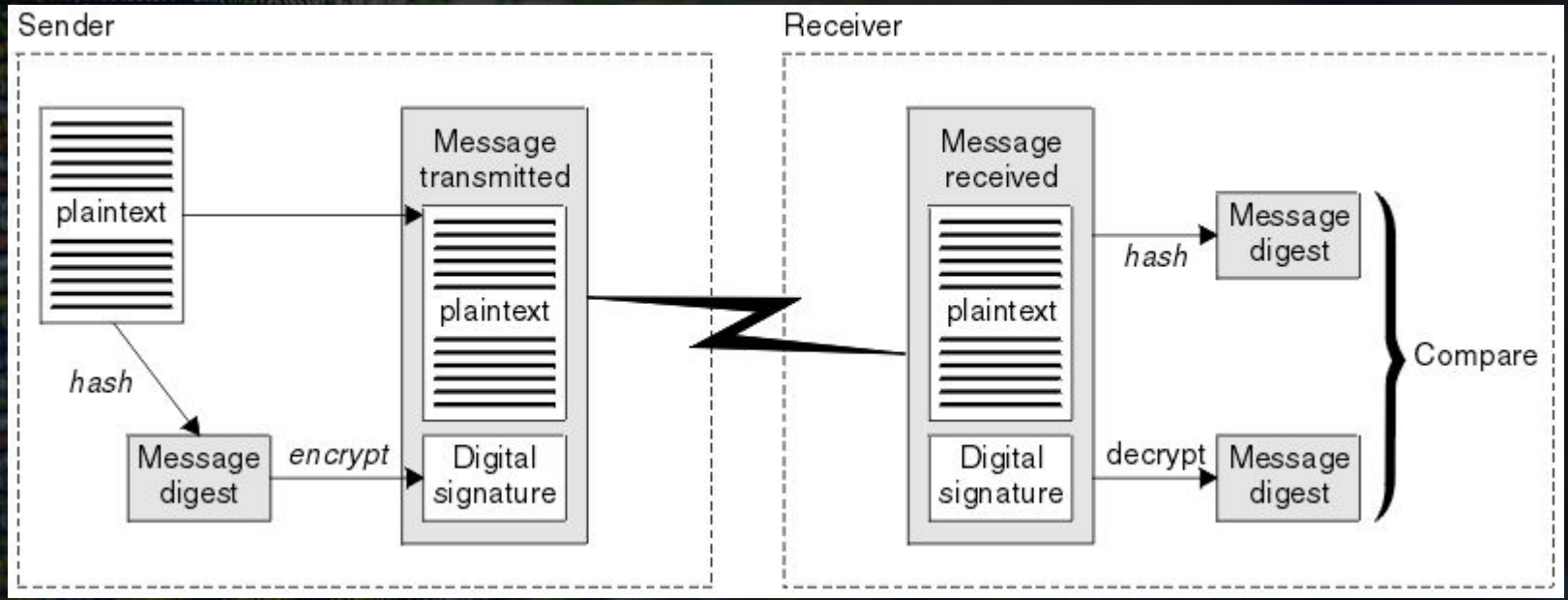
- **Asymmetric RSA encryption** (from HAC ch8)
- That *e* part
  - 3 is a good candidate because it makes small values for computation, but can lead to broken encryption if the same message is sent to multiple recipients
  - 65537 is a good value, because it is 0x10001 in hex, or 1000000000000001 in binary which makes for easy computation
  - Take a look at an OpenSSL generated RSA certificate, it is likely that it will use this value. Also ‘openssl genrsa -h’
  - Many RSA implementations choose e first, then check that phi is coprime with e

Class 7 - Encryption, Security

- **Asymmetric RSA encryption**
- Alice and Bob can now communicate securely with each other by encrypting messages with the other person's public key
- They can also sign their own messages to prove to the other person that they were the original sender, and that the encrypted message was not tampered with in transit
- Eve can listen in all she likes, and if Bob and Alice chose some large primes, Eve probably won't decrypt any messages until after the heat death of the universe

Class 7 - Encryption, Security

- Now, brief X.509/SSL/TLS certificate aside - remember this?



https://www.ibm.com/support/knowledgecenter/SSFKSJ_7.1.0/com.ibm.mq.doc/sy10520_.htm

Class 7 - Encryption, Security

- Now, brief X.509/SSL/TLS certificate aside
- **RSA** is one possible algorithm in this signing and hashing scheme
- ‘Plaintext’ is set of fields, like domain name, location, expiry dates
- As well as *e* and *n* from the RSA keypair (public key)
- A 3rd party **Certificate Authority (CA)** digitally signs the certificate with their private key, essentially validating you are who you say you are in the certificate

Class 7 - Encryption, Security

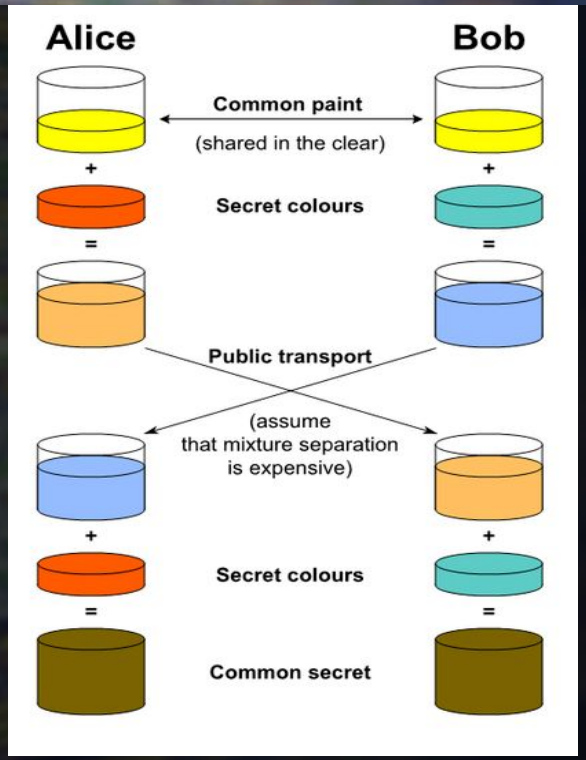
- Now, brief X.509/SSL/TLS certificate aside
- **/etc/ssl/certs** on your VM contains 3rd party CA (certificate authority) certs, tools like curl, apt rely on these for security
- Your browser and OS will also contain a vetted list of CA certificates
- You can validate a certificate that a CA has signed with their private RSA key based on the principle of reversing encryption using the public key
- Say, for example, a bank's TLS certificate, which itself has a public RSA key in it. Use the key to decrypt signatures, and encrypt messages

Class 7 - Encryption, Security

- **Asymmetric encryption**
- Ok, back to Bob and Alice
- What if Alice and Bob have never met, but want to exchange messages using a shared secret key?

Class 7 - Encryption, Security

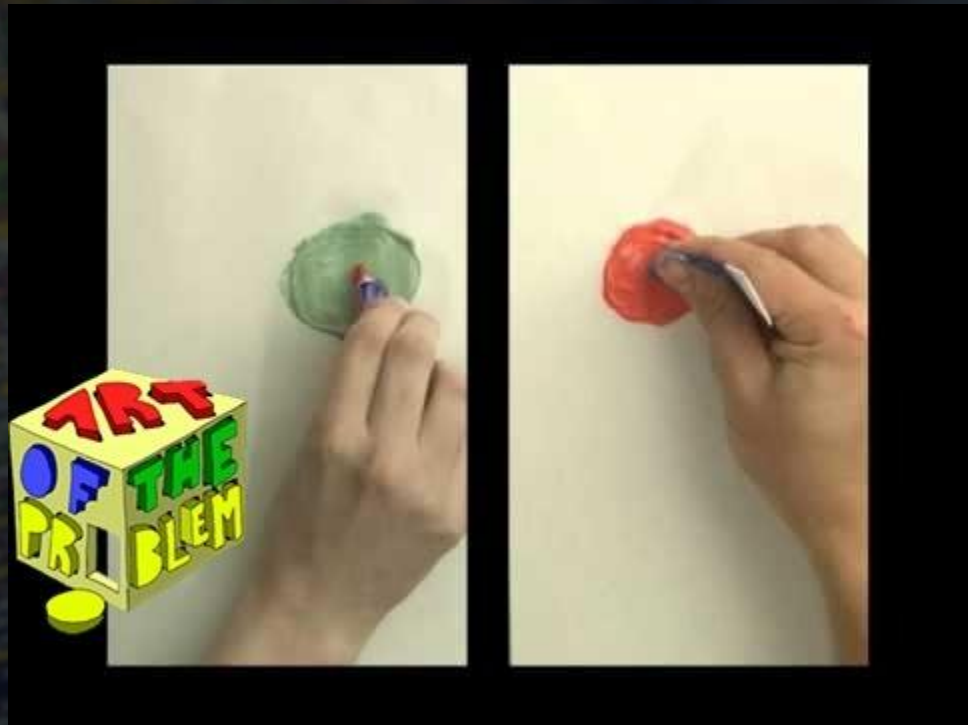
- Enter Diffie-Hellman algorithm



https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange

Class 7 - Encryption, Security

- Diffie-Hellman key exchange
- **Logjam vulnerability**
  - **See also cloudflare**



Class 7 - Encryption, Security

- **Putting it all together**
- What if two parties who have never met would like to exchange secure messages? Say a credit card transaction online
- Recall our tools so far:
  - **Hash functions** - **SHA-1**, **SHA-2**, **SHA-3**, **MD5**
  - **Symmetric ciphers** - block and stream, **AES, ChaCha20**
  - **Asymmetric encryption**
    - **RSA** algorithm
    - **DH** algorithm
  - **X509/SSL/TLS certificates** containing **CA** signed public keys

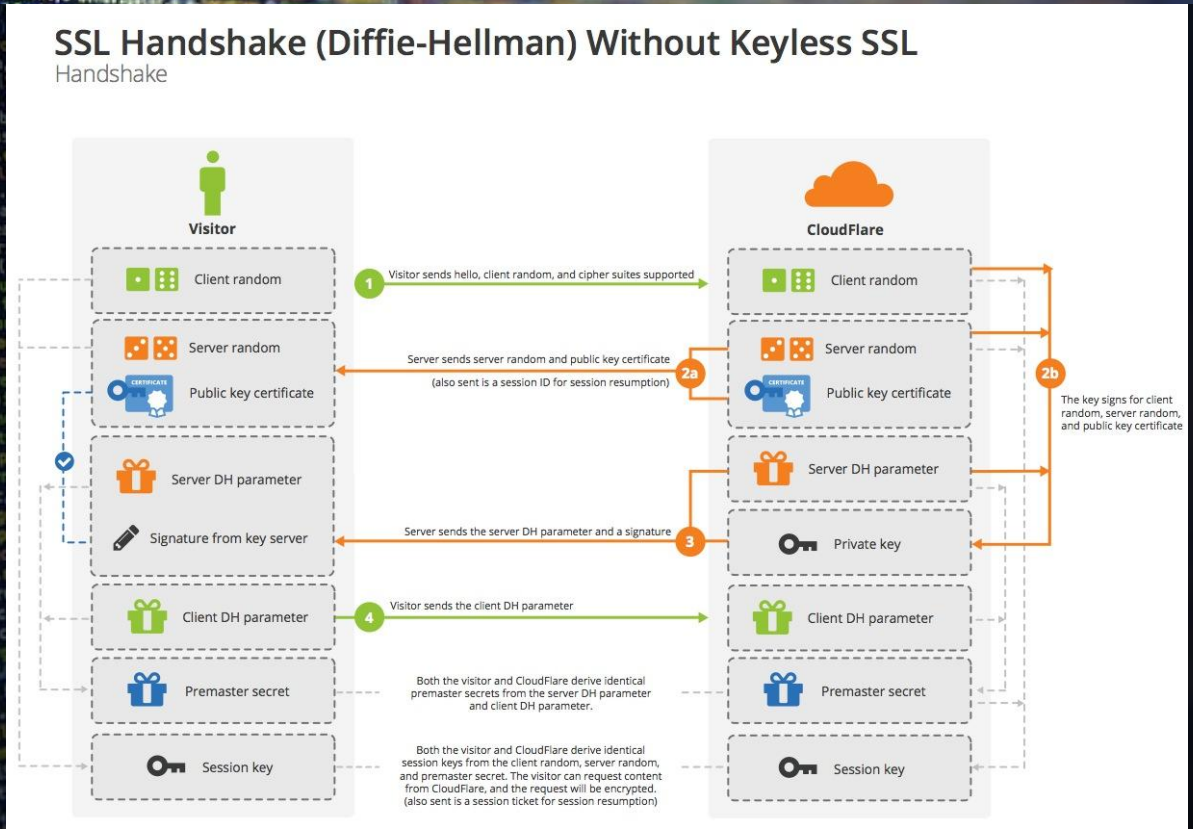
Class 7 - Encryption, Security

- TLS handshake

  “When a TLS client and server first start communicating, they agree on a protocol version, select cryptographic algorithms, optionally authenticate each other, and use public-key encryption techniques to generate shared secrets.”

- TLS parameters

Class 7 - Encryption, Security



Class 7 - Encryption, Security

- Best Explanation I’ve seen:
  - https://tls12.xargs.org/
- To be continued next week!