CMSC389R

Cryptography I

COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

homework questions

Forensics II? Due Monday.

cryptography

cryptography

- Science behind securing digital information
  - Authentication
  - Data integrity
  - Message secrecy
  - Access control

cryptography history

- one time pad (OTP) and xor
- rot13 / caeser chipher
- tattooing messages on shaved heads
- pigpen cipher
- railfence cipher
- …
- None of this is practical!

modern cryptography

- Mathematically rigorous
  - Formal definitions
  - Provability
  - Well-defined assumptions
- If someone can completely break your crypto if they know your scheme, you have failed

uh, math?

- Out of scope for this course
- Take CMSC456, MATH456, or CMSC489R/ENEE459E

Most Prevalent Cryptographic Primitives

- Hashing
  - One way function
- Symmetric key
  - Encryption, decryption, authentication
- Asymmetric (public/private) key
  - Encryption, decryption, authentication

Keywords

- Plaintext: message before any crytographic operation
- Ciphertext: output of encryption
- Hash: output of a hash algorithm

- Integrity: message has not been modified
- Authenticity: message is coming from the expected source
- Confidentiality: nobody aside from intended party can read plaintext
- Non-repudiation: can prove that sending party did send message
- Authorization: only allowing users to do what they have permission to do (only kind of related)

Hashing

- One way function
- Maps any size input to a statically sized output
- One small change in input drastically changes output
- **Very hard (impossible) to undo, instead to crack you attempt to redo**

Applications of hashing

- Instead of storing a password in plaintext, hash it and compare future hashes to the stored hash
- Fingerprint: hash a file, detect changes when fingerprint has changed
  - Can also be used to segment a file into multiple blocks and share these blocks (merkle tree)
- Cryptocurrency proof-of-work

Further notes on hashing

- Does not, on its own, ensure anything
- It is literally just a one-way function

```
md5(hello) => 5d41402abc4b2a76b9719d911017c592
md5(Hello) => 8b1a9953c4611296a827abf8c47804d7
```

- Collision-resistant: nothing should produce the same hash (but they will with a bounded hashspace and infinite message space)

Symmetric Key

- Helps ensure **integrity**, **authenticity**, **confidentiality**
  - What???
- Pre-share a secret key
- Can use this key to encrypt/decrypt **ciphertexts**, generate / decode **message authentication codes**

Applications of Symmetric Key

- Pretty straightforward
- I want to send something to someone I know (where know == share a key with)
- Encrypt it, send it their way. Nobody can read message without the key
- Perfect secrecy vs reality
  - for m in messages $P(m=m' \mid c=c') = P(m=m')$

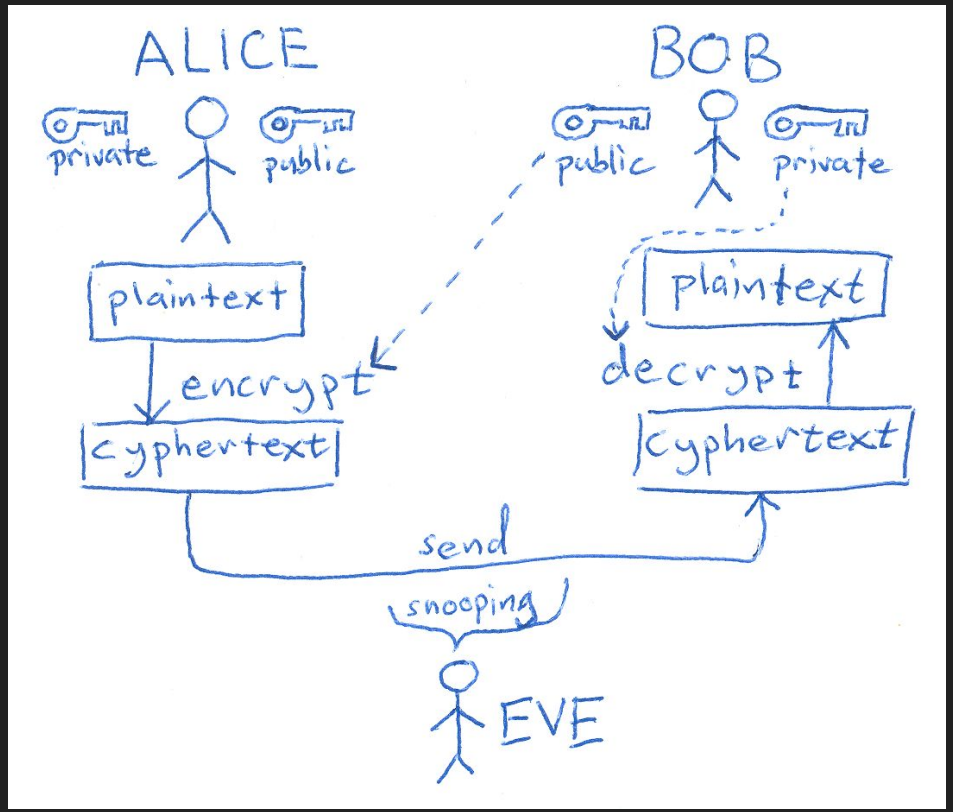
Further Notes on Symmetric Key

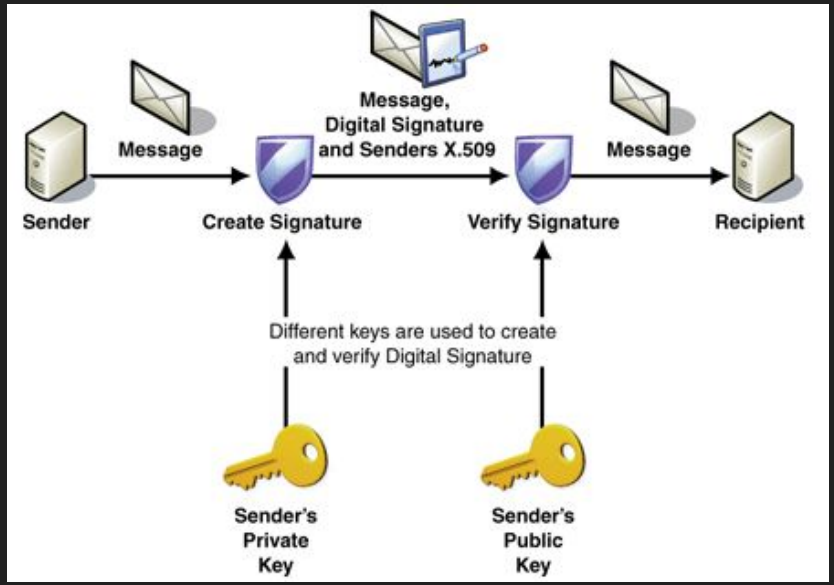
- Encoding via ASCII is not an efficient way of storing information: ciphertexts will likely contain any byte [0-255] (including null byte)
- If key is compromised, we fail entirely
- If we encrypt the same thing twice we don’t want MITM to know: we prepend plaintext with completely random initialization vector (IV) prior to sending and send that along with ciphertext

Asymmetric Key

- Helps ensure **non-repudiation**, **authenticity**, **integrity**, **confidentiality**… sometimes
- Any user wanting to participate needs a public key (visible to anyone) and a private key (which they never share)
- With someone else’s public key, you can **encrypt** messages and check a signature
- With your own private key, you can **decrypt** messages and sign a message

Asymmetric Key





Applications of Asymmetric Key

- Communicate privately with someone you've never exchanged keys with before
- Eventually, PKI (and therefore SSL infrastructure we rely on)

Further Notes on Asymmetric Key

- If our private key is leaked we are hosed
- Somewhat related to key exchange protocols such as El Gamal and Diffie Hellman
- We should (typically) conceptualize our public key as *completely public*
- Inefficient and/or impossible to encrypt large amounts of data: instead, generate a symmetric key, use it encrypt data, encrypt symmetric key with asymmetric key. **Hybrid encryption**

Is Asymmetric Magic? (yes and no)

- Inner workings are very cool! Essentially, multiply two huge primes together. Public key is the product, private key is the two primes. Works because factoring is hard™

Common Algorithms

- Hashing
  - ~~md5~~
  - ~~sha128~~
  - sha256
- Symmetric encryption
  - ~~aes128~~
  - ~~aesecb~~
  - aes256cbc

Common Algorithms

- Asymmetric key
  - RSA
  - Elliptical curve
  - Keys are growing
  - Susceptible to quantum computers, eventually (due to reliance on difficulty of factoring)
  - Quantum-resistant crypto on the rise (theoretically)

Play around with these primitives!

- Hashing
  - echo -ne “YOUR MESSAGE HERE” | <algorithm>sum
    - echo -ne “Hello” | sha256sum
- Symmetric is a bit more complicated, asymmetric even moreso...
  - https://codingbee.net/centos/openssl-demo-encrypting-decrypting-files-using-both-symmetric-and-asymmetric-encryption

Goal coming out of this class?

- Don’t need to know math, prove security, etc
- Have **working knowledge** of crypto such that you can detect vulnerabilities
- Or create workable conceptual model for a secure protocol
- E.g. write a program that allows users to write / read logs by providing a password that an unauthenticated user cannot access or modify

Key takeaways

- Hashing
  - one-to-one function, many applications ranging from storing passwords to chunking files
  - **you cannot “decrypt” a hash**, but collisions are inevitable with a bounded hash space
- Symmetric key
  - this is what most people think of for “encryption”
  - very widely used. requires a pre-shared key but this is an easy problem to solve via asymmetric and/or key exchange
  - Encryption does not guarantee integrity (unmodified payload) -- instead use **MAC**
  - **NEVER** share secret key with attacker
  - **NEVER** reuse IV (which should be randomly generated for every message)
  - Ciphertext is typically a bit larger than plaintext (typically +[1,blocksize] bytes)
- Asymmetric key
  - encryption / decryption / signing without preshared secret key… but will need to share public key(s)
  - **never** share private key
  - hybrid encryption

homework #10

has been posted.

Let us know if you have any questions!

This assignment has X parts.

It is due by 11/18 at 11:59PM.