CS 458 — Module 5: Internet Application Security and Privacy

1 Cryptography

- Recall Kerchhoffs' Principle/Shannon's Maxim — *assume the enemy knows the system*.
- We assume that Eve:
  - May know the algorithm
  - May know part of the plaintext
  - May have many plaintext/ciphertext pairs
  - May have access to an encryption/decryption oracle

1.1 Secret-key Encryption

- AKA symmetric encryption.
- Requires using a key to encrypt and decrypt the message.
- A theoretically perfect cryptosystem is the OTP; but this is very hard to use in practice since any key reuse ruins it, and must be truly random (pseudorandomness is not enough).
- SKES relies on mathematical problems that are hard to do in the opposite direction. This may not hold true with quantum computing for some currently-used options!
- Two main types — stream and block ciphers.
- A stream cipher takes in a keystream, XORs with the PT, and gets the CT.
- Can be very fast, and useful if you need to send a lot of data securely, but can be tricky to use correctly.
- To avoid key re-use, we may use IVs/salts to help make the key different.
- Block ciphers operate by taking part of the plaintext and spitting out the corresponding ciphertext.
- What happens when the PT is larger than one block? We have *modes of operation* for this:
  - ECB uses the same keystream for each block — but this degrades into a shitty substitution cipher!
  - Other examples are CBC, CTR, GCM, etc, that starts with an IV.
- One problem with SKES is how to share keys?

1.2 Public-key Encryption

- Allows Alice to send a secret message to Bob without a pre-arranged shared secret!
- Each user has a public key and a private key.
- These use a lot of math to allow for Alice to encrypt via Bob's public key and Bob to decrypt via his private key.
- One downside with PKES is that a key size must be *much* larger to get equivalent security to, for example, an AES SKES key.
- Another is that with larger keys, means more time. This is slow to practically use for some use cases!
- We may instead use *hybrid* cryptography. This uses PKES to encrypt smaller messages, like a key for a SKES system, then encrypt the messages via SKES!