CS 6324: Information Security
Symmetric & Public-Key Cryptography

Dr. Junia Valente

Department of Computer Science
The University of Texas at Dallas

(contains material from: “Security Engineering” by Ross Anderson, and Dr. Alvaro Cardenas’ Info Sec slides)
Announcement:

- Dr. Alvaro Cardenas will cover next class!
- He will talk about message integrity and cover various concepts including: hash functions, message authentication codes (MACs), public-key infrastructure, etc.
Recap from last class:

- Intro to cryptography & symmetric-key cryptography
- Talked about various ciphers (and why they aren’t secure)
  - Stream ciphers: Caesar cipher, Shift cipher, Vigenère cipher, etc.
  - Block cipher: Playfair
- Talked about the properties of crypto algorithms & keys (e.g., key is the only part kept secret, crypto algorithm details are public)

**Goal:** Changing one input bit (of the plaintext) should--on average--cause half of the output bits (the ciphertext) to change.
Outline

- **Symmetric Encryption**
- Public-Key Cryptography
Symmetric encryption:

- **E, D**: ciphers
- **k**: secret key (e.g., 128-bit)
- **m, c**: plaintext, ciphertext
- **n**: nonce (e.g., randomness, initialization vector aka IV)

Encryption algorithm is publicly known – so, never use a proprietary cipher!
Single vs. Multi Use Key:

**Single use key** (one time key):
- Key is only used to encrypt one message
  
  e.g., encrypted email: new key generated for every email

**Multi use key** (many time key):
- Key used to encrypt multiple messages
  
  e.g., SSL: same key used to encrypt many packets
- Need to use a nonce (unique or random)
A **perfectly** secure encryption system is one that:

Even if an adversary runs for an **unbounded** amount of time, they **learn nothing** about the plaintext from the ciphertext.

- Except the length

**Adversary is unable to determine any information at all about the plaintext**
Perfect Secrecy:

As we saw, classic ciphers fail to satisfy the definition of perfect secrecy. We talk next about one-time pad (single use key) which satisfies this property.
One-Time Pad (OTP): (Single use key)

This encryption scheme requires the communicating parties to share a one-time pre-agreed key of the same size (or longer) than the plaintext message.

Described by Frank Miller (1882), re-invented by Gilbert S. Vernam (1917)

Instead of alphabets, we focus on binary messages in modern crypto.

Shannon (1949): OTP is “secure” against ciphertext-only attacks.
One-Time Pad (OTP): (Single use key)

- Select a uniformly random key $K \leftarrow \{0, 1\}^l$ with the **same size** of the plaintext message $M$, i.e., $|M| = l$.

- **encryption step**: $C = M \oplus K$ (XOR)
- **decryption step**: $M = C \oplus K$

![Diagram and table illustration of OTP process]
One-Time Pad in Practice:

Still used in some diplomatic and intelligence communications
- But too expensive for most applications!
- Why?
  - Long key length
  - Can only be used once!

Q: is OTP an example of a *stream cipher* or *block cipher*?
Computational Secrecy:

We need to relax the property of “perfect secrecy” with a “computation secrecy” property instead:

- Instead of requiring total secrecy against unbounded adversaries, require secrecy against bounded adversaries except with some small probability

- Given enough time and resources, the encryption can be broken
Modern Primitives Overview:

1) Stream ciphers: Random Generators (aka keystream generator)

2) Block ciphers: Random Permutations

Note: they are primitives to build encryption algorithms, but by themselves they are not secure!
Problem: recall that for One-Time Pad, the key is as long as the message
Solution: pseudo random key – stream ciphers

Basically: Pseudo Random Number Generators (PRNG)
(How they work? Given a seed, they produce a sequence that looks random)

\[ C \leftarrow PRNG(K) \oplus M \]

Example of stream ciphers: RC4, SEAL
Not as secure as block ciphers
- Do not use the **key** as a **seed** to pseudorandom number generator

- Use an additional random number (aka **nonce**) as part of the seed to the pseudorandom number generator

\[ C \leftarrow PRNG(K) \oplus M \]

- Use a hash function: \( H(n, k) = seed \)

*(We’ll talk about hash functions next lecture)*
Block Ciphers:

Block ciphers are the crypto workhorse

Encrypt a large block!
Basically a keyed random permutation $F_k(m)$

**Canonical examples:**

1. **Triple-DES:** $n = 64$ bits, $k = 168$ bits
2. **AES:** $n = 128$ bits, $k = 128, 192, 256$ bits

Here we are using “$n$” as block size
1. Data Encryption Standard (DES):

- Developed at IBM (early 1970s), some input from NSA
- Widely used in banking, government, and embedded applications

- Worked well in practice (but brute-force attacks now)
  - Efficient to encrypt & decrypt
  - Not provably secure

- Improvements:
  - Triple DES, AES (aka Rijndael)
  - In practice, no compelling reason to use anything but AES
2. Advanced Encryption Standard (AES):

- Developed by Vincent Rijmen and Joan Daemen
- AES is the most popular block cipher used
- It has a block size of 128-bits, and can use a key of 128, 192, or 256 bits in length.

- It is based on a design principle known as a substitution and permutation network:
  - The function produced by a substitution permutation iteration will be indistinguishable from a random function
In practice, how you use an encryption algorithm is often more important than which one you pick.

An important factor is the “mode of operation”, which specifies how a block cipher with a fixed block cipher (e.g., DES, AES) can be extended to process messages of arbitrary length.

There are several standard modes of operation; and choosing the right one, is an important factor in using a block cipher securely!
Modes of Operation: Electronic Code Book (ECB)

Problem: Identical blocks encrypt identically, i.e., if \( m_1 = m_2 \) then \( c_1 = c_2 \)
- Known-Plaintext Attack (KPA): attacker has samples of plaintext & ciphertext
- Chosen-Plaintext Attack (CPA): attacker chooses plaintext to encrypt
- Chosen-Ciphertext Attacks (CCA): attacker chooses ciphertexts to decrypt

In each case the objective of the attacker may be to either deduce the answer to a query he hasn’t already made (a forgery attack), or to perform a key recovery attack.

KPA < CPA < CCA -- more secure

For having at least CPA security (common best practices), encryption algorithm must be randomized
If we use ECB mode to encrypt redundant data, the patterns will show through, letting an opponent deduce information about the plaintext.

Note: Recall the example on slide 24 (lecture 3)? The network traffic was encrypted with AES using ECB mode; that’s why it leaked patterns.
**Advantages:** Identical blocks encrypt differently

Last ciphertext block depends on entire input

**Disadvantage:** Encryption cannot be done in parallel
Modes of Encryption:

- **CBC** is not secure if adversary can predict IV
- **CBC** needs to use a random nonce for IV

There are other modes:

- **CTR** mode is also secure and can be parallelized
- Even better, **Galois/Counter Mode (GCM)** mode provides encryption and authentication (we’ll get back to this after we talk about message integrity)
- One-time pad is the only cipher that can give **perfect security**
  - But the key can only be used once to achieve perfect security

- In practice, we repeat the use of the same key

- As a result, we have more advanced adversaries:
  - An eavesdropper is a **passive attacker** (ciphertext-only attack)
  - **Active attackers**: chosen-plaintext attack (CPA) & chosen-ciphertext attack (CCA)

- There are two main ciphers in practice:
  - Stream ciphers
  - Block ciphers

**Common best practices:**
- Use AES (a block cipher) in a randomized mode of operation (e.g., GCM mode)
Drawbacks of Symmetric Encryption:

- Alice and Bob need to share their secret key in advance.
  - They can meet in person and share a DVD with the key?

- Need to know in advance the parties with whom you will communicate.

- Difficult to distribute and manage keys in a large organization.

- If you want to communicate with N parties, you need N keys.
Outline

- Symmetric Encryption

- **PUBLIC-KEY CRYPTOGRAPHY**
Physical-World Analogy:

- Secret-Key Encryption
  Alice and Bob both have access to a locker with a single lock, and both have the key.

- Public-Key Encryption
  Alice distributes identical locks to anyone who wants to send her a message, but only Alice has the key to open the locks that anyone can use.
- Public-key cryptography (PKC) refers to a cryptographic system requiring two separate keys: one which is secret and one which is public.

- Although they are different, the two parts of the key-pair are mathematically linked.

- One key locks (encrypts) the plaintext, and the other unlocks (decrypts) the ciphertext.
Public-key Cryptography:

We use the **public key** to encrypt the plaintext, and the **private key** to decrypt the ciphertext.
Public-key Cryptography:

Alice shares her public key to anyone who wants to communicate with her.

Bob uses Alice’s public key to encrypt a message; now only Alice (who has the secret part of her key-pair, i.e., the secret key) can decrypt the message.
Algorithms in Public-Key Encryption:

Key-generation algorithm:
Randomized algorithm that outputs a key-pair: 
\((pk, sk)\),

where \(pk\) is the public key; and \(sk\) the secret key

Encryption algorithm:
\(c \leftarrow E_{pk}(m)\)

Decryption algorithm:
\(m = D_{sk}(c)\)
Public-Key Idea (1976):

- Diffie and Hellman (1976) “New Directions in Cryptography”
- First ideas of public-key cryptography and digital signatures
- Diffie and Hellman won the Turing award in 2015
- Ralph Merkle had similar ideas around the same time
- James Ellis proposed public-key crypto in 1970 in a classified paper. It was only made public in 1997.
Examples:

- **RSA** – relies on the assumption that factoring the product of two large primes is almost impossible

- **El Gamal** – relies on the assumption that taking a discrete logarithm is almost impossible

- **Elliptic Curves** – relies on the assumption that taking the logarithm in elliptic curves is almost impossible
- Public-key encryption is slow (symmetric encryption is orders of magnitude more efficient).

In practice, public-key encryption is used only to **distribute a symmetric key** between participants.

How?

1. With the public key, encrypt the symmetric key
2. Encrypt the rest of the message with the symmetric key
Drawbacks of Public-Key Encryption:

- Another problem: how do we know a key belongs to Bob? (and not to an attacker?)
  - Someone else can publish their public key and claim to be Bob!
  - A man-in-the-middle can intercept the distribution of a public key from Bob to Alice and change it to $P_{k_a}$ (the attacker’s public key)

- One solution: Public-Key Infrastructure (PKI)
  - Trusted root authority (e.g., VeriSign)
Next time:

- We’ll talk about Message Integrity…