CSE 565 Computer Security
Spring 2019

Lecture 2: Symmetric Encryption I

Department of Computer Science and Engineering
University at Buffalo
Cryptographic Tools

- Cryptographic tools are essential in designing secure solutions and their understanding is crucial to correct usage.

- We’ll look at these types of cryptographic tools:
  - symmetric encryption
  - hash functions and message authentication codes
  - public-key encryption
  - digital signatures and certificates
  - pseudo-random number generators

- The most basic problem of cryptography:
  - ensure security of communication over insecure media
Goals of Cryptography

- **Security goals**
  - confidentiality
  - data integrity

- **Basic encryption terminology**
  - plaintext
  - ciphertext
  - cryptographic key
  - encryption
  - decryption
  - cryptanalysis
Symmetric Encryption

- **Symmetric (or secret-key) encryption** means that the same key is used both for encryption and decryption.

- The key must remain secret at both ends.

- Such algorithms are:
  - normally very fast
  - can be used as primitives in more complex cryptographic protocols
  - the key often has a short lifetime
Symmetric Encryption Formally

• More formally, a \textbf{computationally secure symmetric key encryption scheme} is defined as:
  
  – a \textbf{private-key encryption scheme} consists of polynomial-time algorithms (Gen, Enc, Dec) such that
  
  1. Gen: on input the security parameter $n$, outputs key $k$
  2. Enc: on input a key $k$ and a message $m \in \{0, 1\}^*$, outputs ciphertext $c$
  3. Dec: on input a key $k$ and ciphertext $c$, outputs plaintext $m$

  – we write $k \leftarrow \text{Gen}(1^n)$, $c \leftarrow \text{Enc}_k(m)$, and $m \leftarrow \text{Dec}_k(c)$

  • this notation means that Gen and Enc are probabilistic and Dec is deterministic
Symmetric Encryption

- The above definition allows us to encrypt messages of any length

- In practice, there are Two types of symmetric key algorithms:
  - block ciphers
    - the key has a fixed size
    - prior to encryption, the message is partitioned into blocks
    - each block is encrypted and decrypted separately
  - stream ciphers
    - the message is processed as a stream
    - pseudo-random generator is used to produce a long key stream from a short key
Attacks Against Symmetric Encryption

- Encryption and decryption algorithms are assumed to be known to the adversary

- **Types of attacks**
  - **ciphertext only attack**: adversary knows a number of ciphertexts
  - **known plaintext attack**: adversary knows some pairs of ciphertexts and corresponding plaintexts
  - **chosen plaintext attack**: adversary knows ciphertexts for messages of its choice
  - **chosen ciphertext attack**: adversary knows plaintexts for ciphertexts of its choice

- We want a general-purpose algorithm to sustain all types of attacks
Security Against Chosen-Plaintext Attacks

- In chosen-plaintext attack (CPA), adversary $A$ is allowed to ask for encryptions of messages of its choice
  - it is active and adaptive
- $A$ is given black-box access to encryption oracle and can query it on different messages
  - notation $A^O(\cdot)$ means $A$ has oracle access to algorithm $O$
- $A$ is asked to distinguish between encryptions of messages of its choice
CPA Security

- CPA indistinguishability experiment $\text{PrivK}_{\mathcal{A},\mathcal{E}}^{\text{cpa}}(n)$
  1. Random key $k$ is generated by $\text{Gen}(1^n)$
  2. $\mathcal{A}$ is given $1^n$ and ability to query $\text{Enc}_k(\cdot)$, and chooses two messages $m_0, m_1$ of the same length
  3. Random bit $b \leftarrow \{0, 1\}$ is chosen, challenge ciphertext $c \leftarrow \text{Enc}_k(m_b)$ is computed and given to $\mathcal{A}$
  4. $\mathcal{A}$ can use $\text{Enc}_k(\cdot)$ and eventually outputs bit $b'$
  5. Experiment outputs 1 if $b' = b$ ($\mathcal{A}$ wins) and 0 otherwise

- $\mathcal{E} = (\text{Gen}, \text{Enc}, \text{Dec})$ has indistinguishable encryptions under the chosen-plaintext attack (CPA-secure) if for all PPT $\mathcal{A}$

\[
\Pr[\text{PrivK}_{\mathcal{A},\mathcal{E}}^{\text{cpa}}(n) = 1] \leq \frac{1}{2} + \text{negl}(n)
\]
Block Ciphers

- The algorithm maps an $n$-bit plaintext block to an $n$-bit ciphertext block

- Most modern block ciphers are product ciphers
  - we sequentially apply more than one operation to the message

- Often a sequence of permutations and substitutions is used

- A common design for an algorithm is to proceed in iterations
  - one iteration is called a round
  - each round consists of similar operations
  - $i$th round key $k_i$ is derived from the secret key $k$ using a fixed, public algorithm
**Design Principles of Block Ciphers**

- Confusion-diffusion paradigm
  - split a block into small chunks
  - define a permutation on each chunk separately (confusion)
  - mix outputs from different chunks by rearranging bits (diffusion)
  - repeat to strengthen the result
• **Substitution-permutation networks**

  – since a permutation on a block can be specified as a lookup table, this is called **substitution**

  – instead of having substitutions defined by the key, such functions are fixed and applied to messages and keys

  – mixing algorithm is called **mixing permutation**
For this type of algorithm to be reversible, each operation needs to be invertible.
Design Principles of Block Ciphers

- Let’s denote one iteration or round by function $g$
- The initial state $s_0$ is the message $m$ itself
- In round $i$:
  - $g$’s input is round key $k_i$ and state $s_{i-1}$
  - $g$’s output is state $s_i$
- The ciphertext $c$ is the final state $s_{N_r}$, where $N_r$ is the number of rounds
- **Decryption** algorithm applies $g^{-1}$ iteratively
  - the order of round keys is reversed
  - set $s_{N_r} = c$, compute $s_{i-1} = g^{-1}(k_i, s_i)$
Another way to realize confusion-diffusion paradigm is through Feistel network:

- In Feistel network each state is divided into halves of the same length: $L_i$ and $R_i$.
- In one round:
  - $L_i = R_{i-1}$
  - $R_i = L_{i-1} \oplus f(k_i, R_{i-1})$
• Are there any advantages over the previous design?
  
  – operations no longer need to be reversible, as the inverse of the algorithm is not used!
  
  – reverse one round’s computation as $R_{i-1} = L_i$ and
  
  $L_{i-1} = R_i \oplus f(k_i, R_{i-1})$
In both types of networks, the substitution and permutation algorithms must be carefully designed

- choosing random substitution/permutation strategies leads to significantly weaker ciphers
- each bit difference in S-box input creates at least 2-bit difference in its output
- mixing permutation ensures that difference in one S-box propagates to at least 2 S-boxes in next round
• Larger key size means greater security
  – for $n$-bit keys, brute force search takes $2^n/2$ time on average

• More rounds often provide better protection
  – the number of rounds must be large enough for proper mixing

• Larger block size offers increased security
  – security of a cipher also depends on the block length
Data Encryption Standard (DES)

- In 1973 National Institute of Standards and Technology (NIST) published a solicitation for cryptosystems
- DES was developed by IBM and adopted as a standard in 1977
- It was expected to be used as a standard for 10–15 years
- Was replaced only in 2001 with AES (Advanced Encryption Standard)

- **DES characteristics:**
  - key size is 56 bits
  - block size is 64 bits
  - number of rounds is 16
DES uses Feistel network

- Feistel network is used in many block ciphers such as DES, RC5, etc.
- not used in AES
- in DES, each $L_i$ and $R_i$ is 32 bits long; $k_i$ is 48 bits long
• DES has a fixed initial permutation $IP$ prior to 16 rounds of encryption

• The inverse permutation $IP^{-1}$ is applied at the end
The $f$ function $f(k_i, R_{i-1})$

1. first expands $R_{i-1}$ from 32 to 48 bits ($k_i$ is 48 bits long)
2. XORs expanded $R_{i-1}$ with $k_i$
3. applies substitution to the result using S-boxes
4. and finally permutes the value
DES $f$ Function

32 bits $R_{i-1}$

$E$

48 bits $k_i$

48 bits

$S_1$ $\ldots$ $S_8$

6 bits

4 bits

32 bits

$P$

32 bits
• There are 8 S-boxes
  – S-boxes are the only non-linear elements in DES design
  – they are crucial for the security of the cipher

• Example: $S_1$

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 14 | 4 | 13 | 1 | 2 | 15 | 11 | 8 | 3 | 10 | 6 | 12 | 5 | 9 | 0 | 7 |
| 0  | 15 | 7  | 4  | 14 | 2  | 13 | 1  | 10 | 6  | 12 | 11 | 9  | 5  | 3  | 8 |
| 4  | 1  | 14 | 8  | 13 | 6  | 2  | 11 | 15 | 12 | 9  | 7  | 3  | 10 | 5  | 0 |
| 15 | 12 | 8  | 2  | 4  | 9  | 1  | 7  | 5  | 11 | 3  | 14 | 10 | 0  | 6  | 13 |

  – input to each S-box is 6 bits $b_1 b_2 b_3 b_4 b_5 b_6$
  – row = $b_1 b_6$, column = $b_2 b_3 b_4 b_5$
  – output is 4 bits
• More about S-boxes..
  
  – a modified version of IBM’s proposal was accepted as the standard
  
  – some of the design choices of S-boxes weren’t public, which triggered criticism
  
  – in late 1980s – early 1990s differential cryptanalysis techniques were discovered
  
  – it was then revealed that DES S-boxes were designed to prevent such attacks
  
  – such cryptanalysis techniques were known almost 20 years before they were discovered by others
- **Key computation** consists of:
  - circular shift
  - permutation
  - contraction
• Why does decryption work?
  – round function $g$ is invertible
    • compute $L_{i-1} = R_i \oplus f(k_i, L_i)$
    • compute $R_{i-1} = L_i$
  – in the beginning apply $IP$ and at the end apply $IP^{-1}$
  – round keys $k_{16}, \ldots, k_1$ and the $f$ function are computed as before
DES Weak Keys

- The master key $k$ is used to generate 16 round keys

- Some keys result in the same round key to be generated in more than one round
  - this reduces complexity of the cipher

- Solution: check for weak keys at key generation

- DES has 4 weak keys:
  - 0000000 0000000
  - 0000000 FFFFFFFF
  - FFFFFFF 0000000
  - FFFFFFF FFFFFFFF
Attacks on DES

- **Brute force attack**: try all possible $2^{56}$ keys
  - time-consuming, but no storage requirements

- **Differential cryptanalysis**: traces the difference of two messages through each round of the algorithm
  - was discovered in early 90s
  - not effective against DES

- **Linear cryptanalysis**: tries to find linear approximations to describe DES transformations
  - was discovered in 1993
  - has no practical implication
Brute Force Search Attacks on DES

- It was conjectured in 1970s that a cracker machine could be built for $20 million

- In 1990s RSA Laboratories called several DES challenges
  - Challenge II-2 was solved in 1998 by Electronic Frontier Foundation
    - a DES Cracker machine was built for less than $250,000 and found the key was in 56 hours
  - Challenge III was solved in 1999 by the DES Cracker in cooperation with a worldwide network of 100,000 computers
    - the key was found in 22 hours 15 minutes
    - http://www.distributed.net/des
Increasing Security of DES

- DES uses a 56-bit key and this raised concerns

- One proposed solution is **double DES**
  - apply DES twice by using two different keys $k_1$ and $k_2$
  - encryption $c = E_{k_2}(E_{k_1}(m))$
  - decryption $m = D_{k_1}(D_{k_2}(c))$

- The resulting key is $2 \cdot 56 = 112$ bits, so it should be more secure, right?
  - an attack called **meet-in-the-middle** discovers keys $k_1$ and $k_2$ with $2^{56}$ computation and storage
  - better, but not substantially than regular DES
Triple DES

- **Triple DES with two keys** $k_1$ and $k_2$:
  - encryption $c = E_{k_1}(D_{k_2}(E_{k_1}(m)))$
  - decryption $m = D_{k_1}(E_{k_2}(D_{k_1}(c)))$
  - key space is $2 \cdot 56 = 112$ bits

- **Triple DES with three keys** $k_1$, $k_2$, and $k_3$:
  - encryption $c = E_{k_3}(D_{k_2}(E_{k_1}(m)))$
  - decryption $m = D_{k_1}(E_{k_2}(D_{k_3}(c)))$
  - key space is $3 \cdot 56 = 168$ bits

- There is **no known practical attack** against either version

- Can be made backward compatible by setting $k_1 = k_2$ or $k_3 = k_2$
Summary of Attacks on DES

- **DES**
  - best attack: brute force search
  - $2^{55}$ work on average
  - no other requirements

- **Double DES**
  - best attack: meet-in-the-middle
  - requires 2 plaintext-ciphertext pairs
  - requires $2^{56}$ space and about $2^{56}$ work

- **Triple DES**
  - best practical attack: brute force search