LG전자 BS팀

『4일차』: 오후

◈ 훈련과정명 : OS 기본

◈ 훈 련 기 간 : 2023.05.30-2[illegible]3.06.02

목차

『13과목』

5-6교시 :
File System Implementation

Syllabus

학습목표

- 이 워크샵에서는 로컬 파일 시스템 및 디렉토리 구조 구현에 대한 세부 사항 설명 할 수 있다.
- 블록 할당 및 자유 블록 알고리즘과 장단점 논의 할 수 있다.
- 파일 시스템 효율성 및 성능 문제 탐색 할 수 있다.
- 파일 시스템 장애 복구 살펴보기 할 수 있다.
- 구체적인 예로 WAFL 파일 시스템 설명 할 수 있다.

눈높이 체크

- 파일 시스템 구조를 알고 계신가요?
- 파일 시스템 작업을 알고 계신가요?
- 디렉토리 구현을 알고 계신가요?
- 할당 방법을 알고 계신가요?
- 여유 공간 관리를 알고 계신가요?
- 효율성 및 성능을 알고 계신가요?
- 회복을 알고 계신가요?

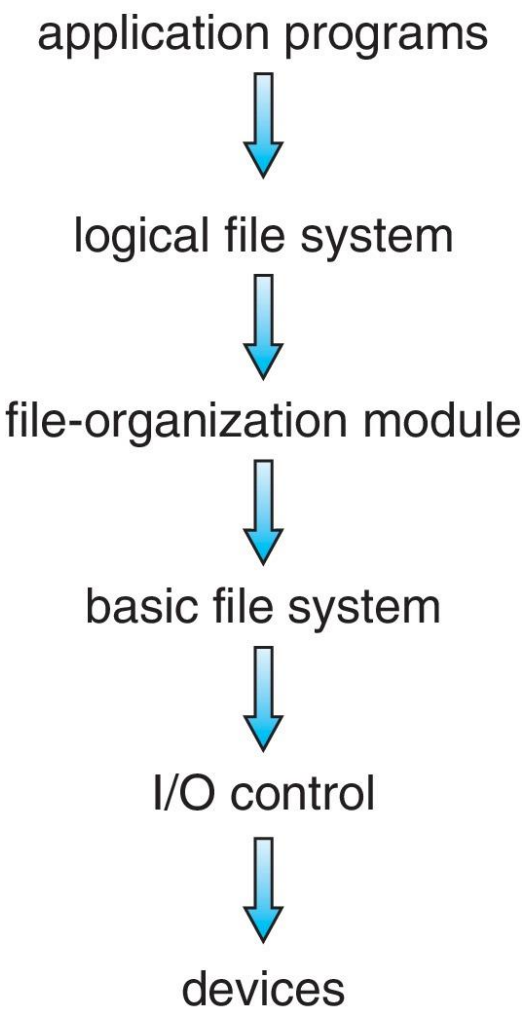
1. File-System Structure

기본 개념

- 파일 구조
  - 논리적 스토리지 유닛
  - 관련 정보 수집
- 파일 시스템은 보조 스토리지(디스크)에 상주
  - 스토리지에 대한 사용자 인터페이스 제공, 논리적 대 물리적 매핑
  - 데이터를 저장하고 위치를 쉽게 검색할 수 있도록 하여 효율적이고 편리한 디스크 액세스를 제공.
- 디스크는 제자리 재작성 및 임의 액세스를 제공.
  - 섹터 블록에서 수행되는 I/O 전송(보통 512바이트)
- 파일 제어 블록(File control block FCB) – 파일에 대한 정보로 구성된 저장 구조
- 장치 드라이버 Device driver 는 물리적 장치를 제어.
- 계층으로 구성된 파일 시스템

1. File-System Structure

Layered File System



1. File-System Structure

파일 시스템 계층

- 장치 드라이버 Device drivers 는 I/O 제어 계층에서 I/O 장치를 관리합니다.
  - 다음과 같은 명령이 주어집니다.
  - 드라이브 1, 실린더 72, 트랙 2, 섹터 10을 메모리 위치 1060으로 읽기
  - 하드웨어 컨트롤러에 저수준 하드웨어 특정 명령 출력
- "retrieve block 123"과 같은 명령이 주어진 기본 파일 시스템 Basic file system 은 장치 드라이버로 변환.
- 또한 메모리 버퍼 및 캐시 관리(할당, 해제, 교체)
  - 버퍼는 전송 중인 데이터를 보유.
  - 캐시는 자주 사용되는 데이터를 보유.
- 파일 구성 모듈 File organization module 은 파일, 논리 주소 및 물리적 블록을 이해합니다.
- 논리 블록 #을 물리적 블록 #으로 변환
- 여유 공간, 디스크 할당 관리

1. File-System Structure

파일 시스템 계층

- 논리 파일 시스템은 메타데이터 정보를 관리.
  - 파일 제어 블록(UNIX의 inode)을 유지하여 파일 이름을 파일 번호, 파일 핸들, 위치로 변환
  - 디렉토리 관리
  - 보호
- 복잡성과 중복성을 줄이는 데 유용하지만 오버헤드가 추가되고 성능이 저하될 수 있는 레이어링
- 논리적 계층은 OS 설계자에 따라 어떠한 코딩 방식으로도 구현 가능

1. File-System Structure

파일 시스템 계층

- 많은 파일 시스템, 경우에 따라 운영 체제 내에서 여러 개
- 각각 고유한 형식:
- CD-ROM은 ISO 9660.
- 유닉스에는 UFS, FFS가 있다.
- Windows에는 FAT, FAT32, NTFS는 물론 플로피, CD, DVD Blu-ray,
- Linux에는 확장 파일 시스템 ext3 및 ext4를 비롯하여 130개 이상의 유형이 있습니다. 분산 파일 시스템 등)
- 새로운 것 – ZFS, GoogleFS, Oracle ASM, FUSE

2. File-System Operations

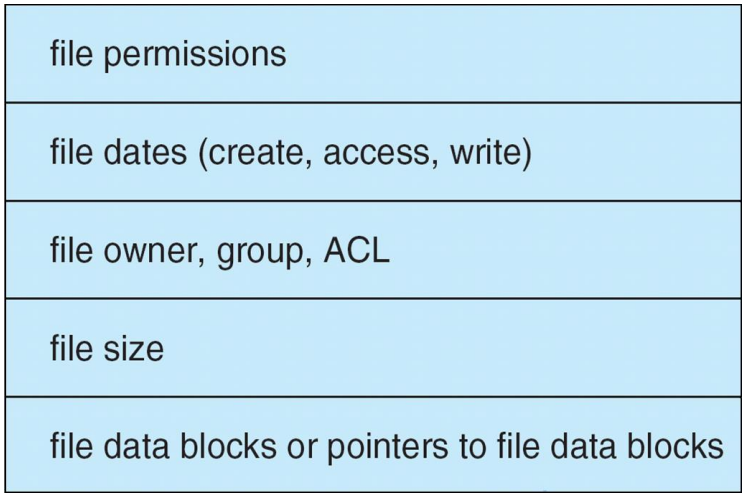
기본 개념

- API 수준에서 시스템 호출이 있지만 해당 기능을 어떻게 구현할까?
  - On-disk and in-memory structures
- 부팅 제어 블록 Boot control block 에는 시스템이 해당 볼륨에서 OS를 부팅하는 데 필요한 정보가 포함되어 있다.
  - 볼륨에 OS가 포함된 경우 필요하며 일반적으로 볼륨의 첫 번째 블록
- 볼륨 제어 블록 Volume control block (수퍼 블록, 마스터 파일 테이블 superblock, master file table)에는 볼륨 세부 정보가 포함.
  - 총 블록 수, 사용 가능한 블록 수, 블록 크기, 사용 가능한 블록 포인터 또는 배열
- 디렉토리 구조는 파일을 구성.
  - 이름 및 inode 번호, 마스터 파일 테이블

2. File-System Operations

File Control Block (FCB)

- OS는 파일에 대한 많은 세부 정보를 포함하는 파일당 FCB를 유지합니다.
- 일반적으로 inode 번호, 권한, 크기, 날짜
- 예



2. File-System Operations

메모리 내 파일 시스템 구조

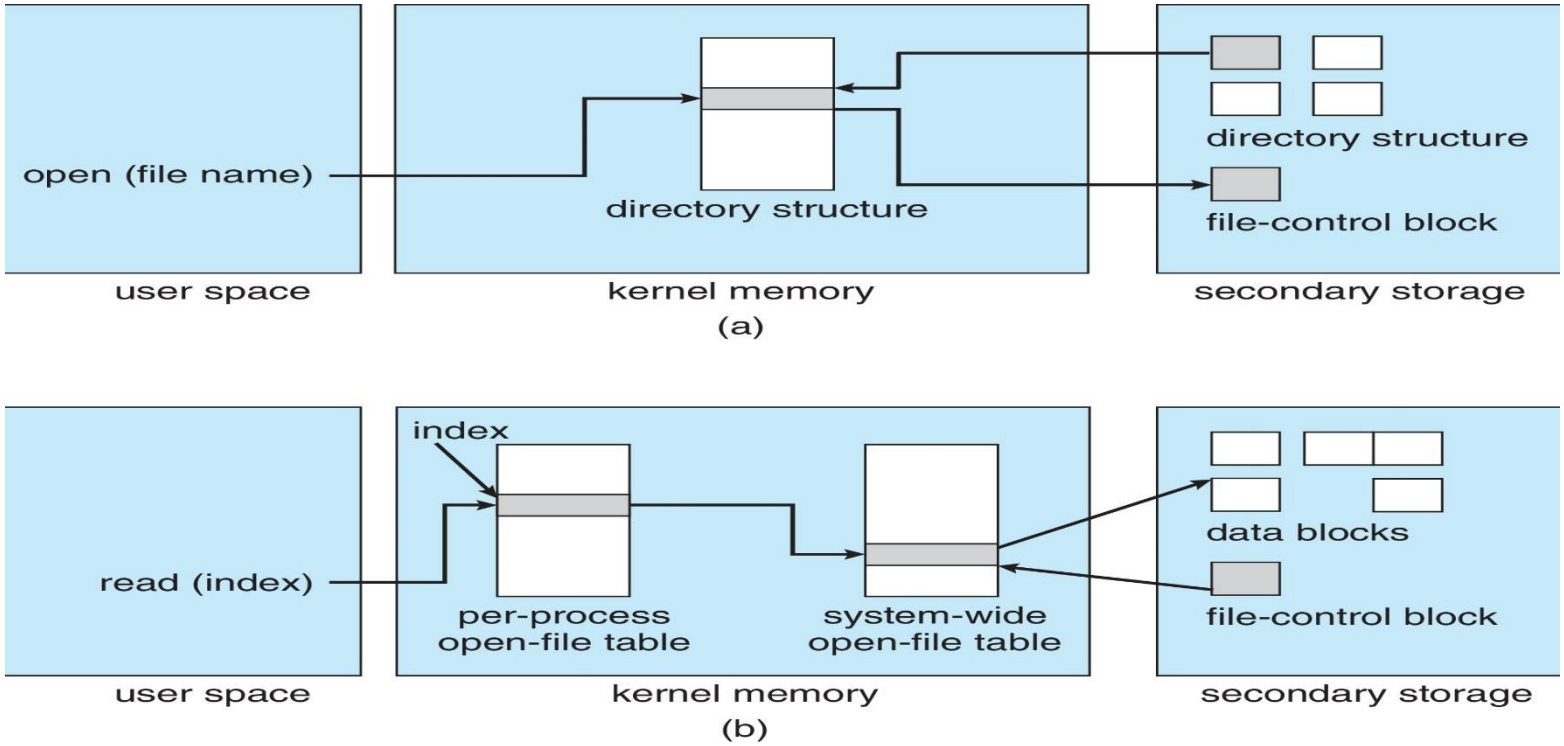
- **In-Memory File System Structures**

- 파일 시스템 마운트, 마운트 지점, 파일 시스템 유형을 저장하는 마운트 테이블 **Mount table**
- 시스템 전체의 열린 파일 테이블 **System-wide open-file table** 에는 각 파일의 **FCB** 사본 및 기타 정보가 포함됩니다.
- 프로세스별 열린 파일 테이블 **Per-process open-file table** 에는 시스템 전체 열린 파일 테이블의 적절한 항목에 대한 포인터와 기타 정보가 포함됩니다.

2. File-System Operations

메모리 내 파일 시스템 구조

- 그림 12-3(a)는 파일 열기를 나타냅니다.
- 그림 12-3(b)는 파일 읽기를 나타냅니다.



3. Directory Implementation

기본 개념

- 데이터 블록에 대한 포인터가 있는 파일 이름의 Linear list
  - 간단한 프로그래밍
  - 실행에 시간이 많이 소요됨
  - ✓ 선형 검색 시간
  - ✓ 연결된 목록을 통해 알파벳순으로 정렬하거나 B+ 트리를 사용할 수 있다.
- 해시 테이블 Hash Table - 해시 데이터 구조가 있는 선형 목록
  - 디렉토리 검색 시간 단축 Decreases
  - 충돌 Collisions – 두 파일 이름이 동일한 위치로 해시되는 상황
  - 항목이 고정 크기이거나 체인 오버플로 방법을 사용하는 경우에만 적합.

4. Allocation Methods

Allocation Methods?

- 할당 방법은 디스크 블록이 파일에 할당되는 방법을 나타낸다.

  - **Contiguous**
  - **Linked**
  - **File Allocation Table (FAT)**

4. Allocation Methods

연속 할당 방법

- 할당 방법은 디스크 블록이 파일에 할당되는 방법을 나타낸다
- 각 파일은 연속 블록 집합을 차지.
  - 대부분의 경우 최고의 성능
  - 단순성 – 시작 위치(블록 번호)와 길이(블록 수)만 필요
  - 문제는 다음과 같다.

① 파일을 위한 디스크 공간 찾기,
② 파일 크기를 알고,
③ 외부 단편화, 오프라인(다운타임) 또는 온라인 압축 필요

4. Allocation Methods

Extent-Based Systems

- 많은 최신 파일 시스템(예: Veritas File System)은 수정된 연속 할당 체계를 사용.
- 익스텐트 기반 파일 시스템은 익스텐트에 디스크 블록을 할당.
- 익스텐트는 연속적인 디스크 블록
- 파일 할당을 위해 익스텐트가 할당.
- 파일은 하나 이상의 익스텐트로 구성.

4. Allocation Methods

연결된 할당 Linked Allocation

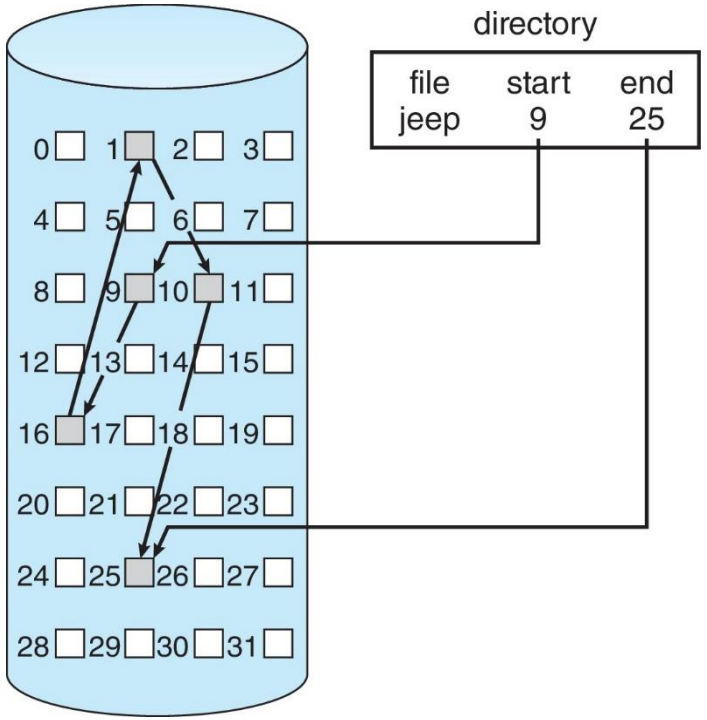
- 각 파일은 블록의 연결된 목록.
- 파일은 nil 포인터에서 끝납니다.
- 외부 조각화 없음
- 각 블록에는 다음 블록에 대한 포인터가 포함.
- 압축 없음, 외부 단편화
- 새 블록이 필요할 때 호출되는 여유 공간 관리 시스템
- 블록을 그룹으로 클러스터링하여 효율성을 향상하지만 내부 단편화를 증가시킨다.
- 신뢰성이 문제가 될 수 있음
- 블록 찾기에는 많은 I/O 및 디스크 검색이 필요할 수 있다.

4. Allocation Methods

Linked Allocation Example

- 각 파일은 디스크 블록의 연결된 목록. 블록은 디스크의 어느 위치에나 흩어져 있을 수 있다.
- 구조
- 매핑

LA/511 → Q, R

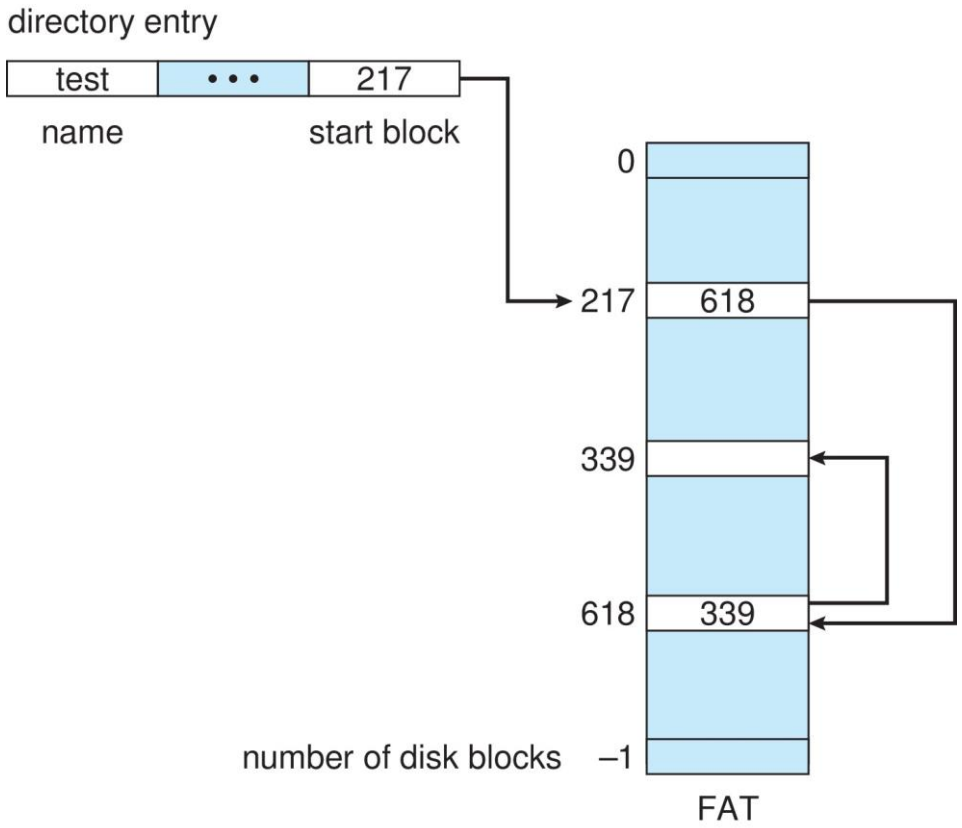


- 액세스할 블록은 파일을 나타내는 블록의 연결된 체인에 있는 Q번째 블록.
- 블록으로의 변위 = R + 1

4. Allocation Methods

FAT 할당 방법

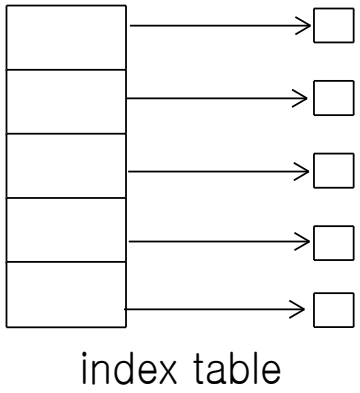
- 볼륨 시작 부분에 블록 번호로 인덱싱된 테이블이 있음
- 연결 목록과 매우 유사하지만 디스크에서 더 빠르고 캐시 가능
- 새로운 블록 할당 간단
- 파일 할당 테이블



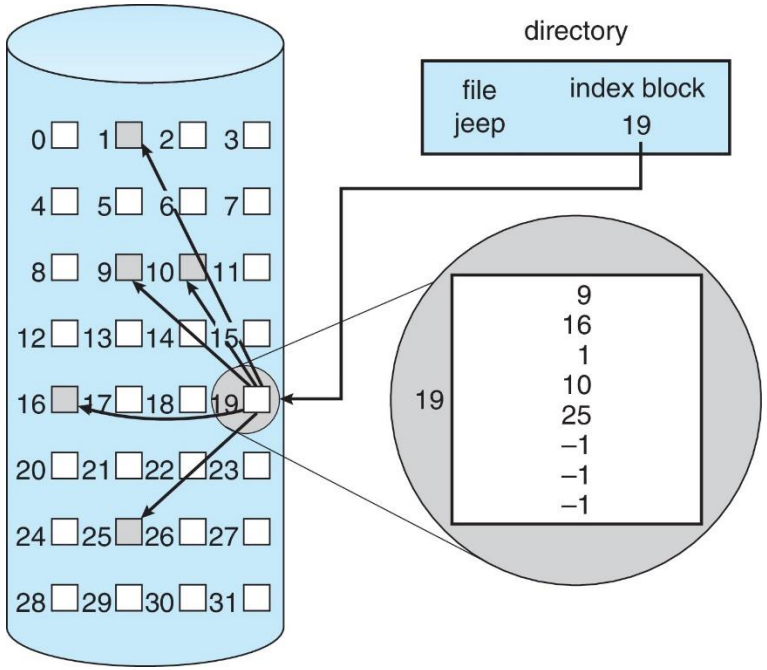
4. Allocation Methods

Indexed Allocation Method

- 각 파일에는 데이터 블록에 대한 포인터의 자체 인덱스 블록이 있습니다.
- 논리적 보기



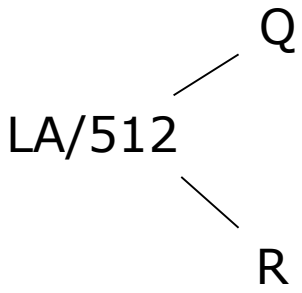
- 인덱스 할당의 예



4. Allocation Methods

Indexed Allocation Method

- 인덱스 할당 – 작은 파일
- 색인 테이블 필요
- 임의 액세스
- 외부 조각화 없이 동적 액세스하지만 인덱스 블록의 오버헤드가 있음
- 최대 크기가 256KB이고 블록 크기가 512바이트인 파일에서 논리적에서 물리적으로 매핑합니다. 인덱스 테이블에는 1개의 블록만 필요.



- 계산:
- Q = 인덱스 테이블로의 변위
- R = 블록으로 변위

『14과목』

7교시 : Security

Syllabus

학습목표

- 이 워크샵에서는 보안 위협 및 공격 논의 설명 할 수 있다.
- 암호화, 인증 및 해싱의 기본 사항 설명 할 수 있다.
- 컴퓨팅에서 암호화 사용 검토 할 수 있다.
- 보안 공격에 대한 다양한 대응책 설명 할 수 있다.

눈높이 체크

- 보안 문제를 알고 계신가요?
- 프로그램 위협을 알고 계신가요?
- 시스템 및 네트워크 위협을 알고 계신가요?
- 보안 도구로서의 암호화를 알고 계신가요?
- 사용자 인증을 알고 계신가요?
- 보안 방어 구현을 알고 계신가요?
- 시스템 및 네트워크 보호를 위한 방화벽을 알고 계신가요?
- 컴퓨터 보안 분류를 알고 계신가요?

1. 보안 문제

보안 문제란?

- 모든 상황에서 리소스를 의도한 대로 사용하고 액세스하는 경우 시스템 보안 secure
- 달성 불가능Unachievable

- 침입자(크래커) Intruders (crackers) 가 보안 위반 시도
- 위협 Threat 은 잠재적인 보안 위반.
- 공격 Attack 은 보안을 위반하려는 시도.
- 공격은 우발적이거나 악의적일 수 있다.
- 악의적인 오용보다 우발적인 오용으로부터 보호하기가 더 쉽다.

1. 보안 문제

보안 위반 범주

- 기밀 위반Breach of confidentiality
  - 무단 데이터 읽기
- 무결성 위반Breach of integrity
  - 무단 데이터 수정
- 가용성 위반Breach of availability
  - 데이터 무단 파기
- 서비스 도용Theft of service
  - 리소스 무단 사용
- 서비스 거부(DOS) Denial of service (DOS)
  - 정당한 이용방지

1. 보안 문제

보안 위반 방법

- 가장 Masquerading (위반 인증 authentication)
  - 권한을 에스컬레이션하기 위해 인증된 사용자인 척
- 재생 공격Replay attack
  - 그대로 또는 메시지 수정
- 중간자 공격Man-in-the-middle attack
  - 침입자는 데이터 흐름에 앉아 송신자에서 수신자로 또는 그 반대로 가장.
- 세션 하이재킹Session hijacking
  - 인증을 우회하기 위해 이미 설정된 세션을 가로채기
- 권한 에스컬레이션Privilege escalation
  - 사용자 또는 리소스가 가지고 있어야 하는 것 이상의 액세스 권한을 가진 일반적인 공격 유형

1. 보안 문제

보안 조치 수준

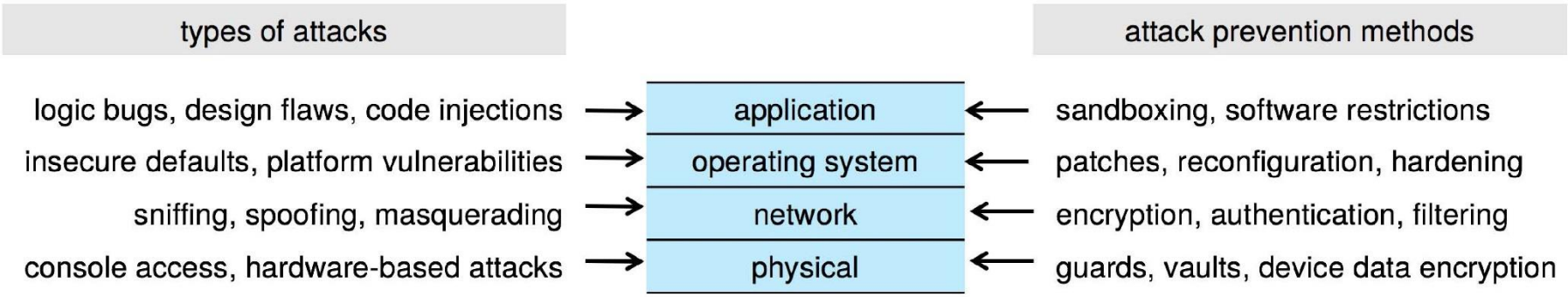
- 절대적인 보안을 유지하는 것은 불가능하지만 대부분의 침입자를 막을 수 있을 만큼 가해자에게 비용을 충분히 높인다.
- 보안은 다음과 같은 4가지 수준에서 이루어져야 효과적.
  - 물리적Physical
    - ✓ 데이터 센터, 서버, 연결된 터미널
  - 애플리케이션Application
    - ✓ 양성 또는 악성 앱은 보안 문제를 일으킬 수 있다.
  - 운영 체제
    - ✓ 보호 메커니즘, 디버깅
  - 회로망Network
    - ✓ 가로채는 통신, 중단, DOS
- 보안은 체인의 가장 약한 링크만큼 취약합니다.
- 인간도 피싱 phishing 및 사회 공학 social-engineering 공격을 통한 위험

1. 보안 문제

프로그램 위협

- **트로이 목마Trojan Horse**
  - 환경을 오용하는 코드 세그먼트
  - 사용자가 작성한 프로그램을 다른 사용자가 실행할 수 있도록 하는 메커니즘을 악용합니다.
  - 스파이웨어, 팝업 브라우저 창, 비밀 채널
  - 스파이웨어에 감염된 시스템에서 전달되는 스팸의 최대 80%
- **트랩 도어Trap Door**
  - 정상적인 보안 절차를 우회하는 특정 사용자 식별자 또는 암호
  - 컴파일러에 포함될 수 있음

4계층 보안 모델



2. 프로그램 위협

프로그램 위협

- 맬웨어 Malware - 컴퓨터를 악용, 비활성화 또는 손상시키도록 설계된 소프트웨어
- 트로이 목마 Trojan Horse -은밀하게 작동하는 프로그램
- 스파이웨어 Spyware – 추가 표시, 사용자 데이터 캡처를 위해 합법적인 소프트웨어와 함께 자주 설치되는 프로그램
- 랜섬웨어 Ransomware – 암호화를 통해 데이터를 잠그고 잠금을 해제하려면 지불을 요구.
- 기타 트랩 도어, 논리 폭탄logic boms 포함
- 모두 최소 권한 원칙을 위반하려고 합니다.

2. 프로그램 위협

버퍼 오버플로 조건이 있는 C 프로그램

- 코드 검토가 도움이 될 수 있다. 프로그래머는 서로의 코드를 검토하고 논리 흐름, 프로그래밍 결함을 찾는다.

```
#include <stdio.h>
#define BUFFER SIZE 256
int main(int argc, char *argv[])
{
        char buffer[BUFFER SIZE];
        if (argc < 2)
                return -1;
        else {
                strcpy(buffer,argv[1]);
                return 0;
        }
}
```

2. 프로그램 위협

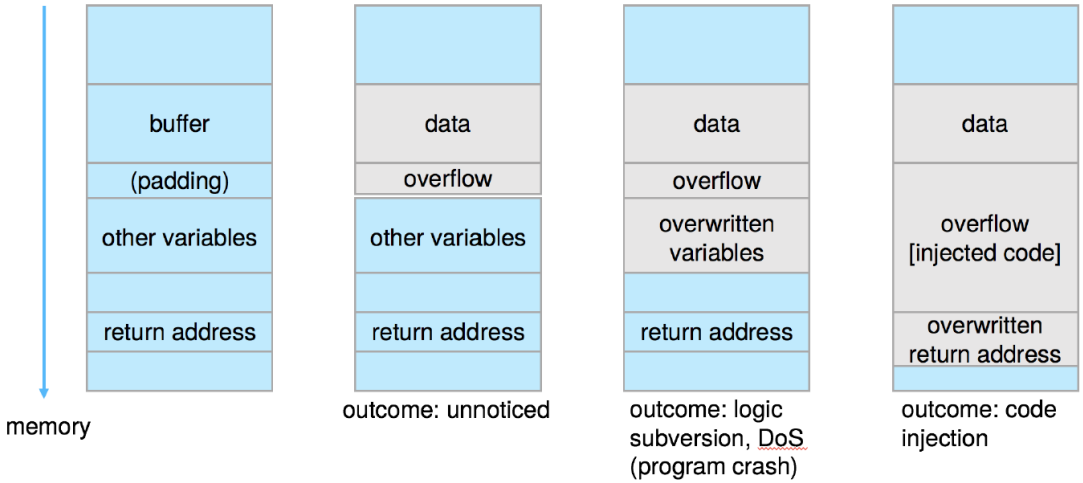
Code Injection

- 코드 주입 공격은 시스템 코드가 악성이 아니지만 실행 가능한 코드를 추가하거나 수정할 수 있는 버그가 있을 때 발생.
- 일반적으로 포인터를 통해 직접 메모리 액세스를 허용하는 C 또는 C++와 같은 저수준 언어에서 열악하거나 안전하지 않은 프로그래밍 패러다임의 결과
- 목표는 코드가 버퍼에 배치되고 공격으로 인해 실행되는 버퍼 오버플로.
- 스크립트 키디가 실행할 수 있음 – 작성되었지만 식별자를 악용하는 도구 사용

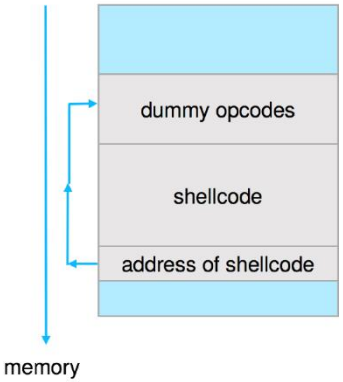
2. 프로그램 위협

Code Injection

- 코드 삽입의 결과는 다음과 같다.



- 버퍼 오버플로를 악용하기 위해 트램펄린 trampoline 을 코드 실행에 자주 사용



2. 프로그램 위협

계속되는 위협

- 공격은 여전히 흔하고 여전히 발생
- 공격은 시간이 지남에 따라 과학 실험에서 조직 범죄 도구로 이동했다.
  - 특정 기업을 대상으로
  - 스팸 및 DDOS 전달 도구로 사용할 봇넷 생성
  - 암호, 신용 카드 번호를 캡처하는 키스트로크 로거 Keystroke logger
- Windows가 대부분의 공격 대상이 되는 이유는 무엇일까?
  - 가장 흔한
  - 모두가 관리자.
  - 라이센스 Licensing
  - 유해한 것으로 간주되는 Monoculture

3. 시스템 및 네트워크 위협

기본 개념

- 일부 시스템은 기본적으로 보안이 설정되지 않고 "개방".
  - 공격 표면attack surface 감소
  - 그러나 사용하기 어렵고 관리하는 데 더 많은 지식이 필요.
- 탐지 및 방지하기 어려운 네트워크 위협
  - 약한 보호 시스템
  - 액세스의 기반이 되는 공유 비밀을 갖는 것이 더 어렵다.
  - 시스템이 인터넷에 연결되면 물리적 제한 없음
  - 또는 인터넷에 연결된 시스템이 있는 네트워크에서
  - 연결 시스템의 위치 파악조차 어려움
  - IP 주소는 지식일 뿐

3. 시스템 및 네트워크 위협

기본 개념

- 웜 Worms - 스폰 spawn 메커니즘을 사용. 독립형 프로그램
- 인터넷 웜Internet worm
- 악용된 UNIX 네트워킹 기능(원격 액세스) 및 finger 및 sendmail 프로그램의 버그
- 암호를 사용하지 않고 친근한 시스템에 액세스하기 위해 rsh가 사용하는 악용된 신뢰 관계 메커니즘
- 그래플링 훅 Grappling hook 프로그램 업로드 메인 웜 프로그램-99줄의 C 코드
- 후킹된 시스템은 메인 코드를 업로드하고 연결된 시스템을 공격하려 했다.
- 또한 암호 추측을 통해 로컬 시스템의 다른 사용자 계정에 침입을 시도했다.
- 대상 시스템이 이미 감염된 경우 7번째를 제외하고 중단.

3. 시스템 및 네트워크 위협

기본 개념

- **포트 스캔Port scanning**

- 하나 또는 IP 주소 범위의 포트 범위에 대한 자동 연결 시도
- 응답 서비스 프로토콜 탐지
- 시스템에서 실행 중인 OS 및 버전 감지
- nmap은 응답에 대해 지정된 IP 범위의 모든 포트를 스캔.
- nessus는 시스템에 적용할 수 있는 프로토콜 및 버그(및 악용) 데이터베이스를 보유하고 있습니다.
- 좀비 시스템zombie systems 에서 자주 시작됨
  - ✓ 추적 가능성을 줄이기 위해

3. 시스템 및 네트워크 위협

기본 개념

- **서비스 거부Denial of Service**
  - 유용한 작업을 수행하지 못하도록 대상 컴퓨터를 과부하
  - DDoS(Distributed Denial-of-Service)는 한 번에 여러 사이트에서 발생합니다.
  - IP 연결 핸드셰이크(SYN)의 시작을 고려하십시오.
  - OS에서 얼마나 많은 시작 연결을 처리할 수 있습니까?
  - 웹 사이트에 대한 트래픽 고려
  - 표적이 되는 것과 정말로 인기가 있는 것의 차이를 어떻게 알 수 있습니까?
  - 우발적 Accidental – 나쁜 fork() 코드를 작성하는 CS 학생
  - 의도적 Purposeful – 갈취 extortion, 처벌 punishment
- **포트 스캔Port scanning**
  - 연결을 허용하는 네트워크 포트를 찾는 자동화된 도구
  - 선과 악에 사용

4. 보안 도구로서의 암호화

기본 개념

- 사용 가능한 가장 광범위한 보안 도구
- 지정된 컴퓨터 내부에서 메시지의 소스 및 대상을 알고 보호할 수 있다.
  - ✓ OS는 프로세스 ID, 통신 포트를 생성, 관리, 보호.
- 네트워크의 메시지 소스 및 대상은 암호화 없이는 신뢰할 수 없다.
  - ✓ 로컬 네트워크 – IP 주소?
  - ✓ 승인되지 않은 호스트 추가 고려
  - ✓ WAN / 인터넷 – 진정성을 확립하는 방법
  - ✓ IP 주소를 통하지 않음

4. 보안 도구로서의 암호화

암호화 Cryptography

- 메시지의 잠재적 발신자(소스) 및/또는 수신자(목적지)를 제한하는 수단
- 비밀(키) 기반
- 활성화
  - ✓ 출처 확인
  - ✓ 특정 목적지에서만 수령
  - ✓ 발신자와 수신자 사이의 신뢰 관계

4. 보안 도구로서의 암호화

Encryption

- 가능한 메시지 수신자 집합을 제한.
- Encryption 알고리즘은 다음으로 구성.

- 키의 K 설정 keys
- M의 메시지 설정 messages
- 암호문 세트 C(encrypted 메시지) ciphertexts
- 함수 E : K → (M→C). 즉, 각 $k \in K, E_k$ 는 메시지로부터 암호문을 생성하는 함수이다.
- 모든 k에 대한 E와 E_k 는 모두 효율적으로 계산 가능한 함수여야 함.
- 함수 D : K → (C → M). 즉, 각 $k \in K, D_k$ 는 암호문으로부터 메시지를 생성하는 함수이다.
- 모든 k에 대한 D와 D_k 는 모두 효율적으로 계산 가능한 함수여야 함.

4. 보안 도구로서의 암호화

Encryption

- 암호화 알고리즘은 다음과 같은 필수 속성을 제공해야 한다. 암호문 $c \in C$ 가 주어지면 컴퓨터는 **k**를 소유하는 경우에만 $E_k(m) = c$ 가 되도록 **m**을 계산할 수 있다.
- 따라서 **k**를 보유한 컴퓨터는 암호문을 생성하는 데 사용된 평문으로 해독할 수 있지만 **k**를 보유하지 않은 컴퓨터는 암호문을 해독할 수 없습니다.
- 암호문은 일반적으로 노출되기 때문에(예: 네트워크에서 전송) 암호문에서 **k**를 도출하는 것이 불가능하다는 것이 중요합니다.

4. 보안 도구로서의 암호화

대칭 암호화 Symmetric Encryption

- 암호화 및 복호화에 사용되는 동일한 키
  - 따라서 k는 비밀로 유지되어야 합니다.
- DES는 가장 일반적으로 사용되는 대칭 블록 암호화 알고리즘(US Govt에서 생성)
  - 한 번에 데이터 블록을 암호화.
  - 키가 너무 짧아서 이제 안전하지 않은 것으로 간주됨
- 보다 안전한 것으로 간주되는 Triple-DES
  - 2개 또는 3개의 키를 사용하여 3번 사용하는 알고리즘
  - 예를 들어

$$c = E_{k3}(D_{k2}(E_{k1}(m)))$$

- 2001 NIST는 새로운 블록 암호인 AES(Advanced Encryption Standard)를 채택했다.
  - 128, 192 또는 256비트의 키는 128비트 블록에서 작동.
- RC4는 가장 일반적인 대칭형 스트림 암호이지만 취약점이 있는 것으로 알려져 있다.
  - 바이트 스트림을 암호화/해독(즉, 무선 전송).
  - 키는 의사 난수 비트 생성기에 대한 입력.
  - 무한 키스트림 생성

4. 보안 도구로서의 암호화

비대칭 암호화 Asymmetric Encryption

- 두 개의 키가 있는 각 사용자를 기반으로 하는 공개 키 암호화 Public-key encryption :
  - 공개 키 public key – 데이터를 암호화하는 데 사용되는 공개 키
  - 개인 키 private key – 데이터를 해독하는 데 사용되는 개별 사용자에게만 알려진 키
- 복호화 방식을 쉽게 파악할 수 없이 공개할 수 있는 암호화 방식이어야 함
  - 가장 일반적인 것은 RSA 블록 암호.
  - 숫자가 소수인지 여부를 테스트하는 효율적인 알고리즘
  - 숫자의 소인수를 찾는 효율적인 알고리즘은 없다.

4. 보안 도구로서의 암호화

비대칭 암호화 Asymmetric Encryption

- 공식적으로 $k_{e,N}$에서 $k_{d,N}$ 을 도출하는 것은 계산상 불가능하므로 k_e는 비밀로 유지될 필요가 없으며 널리 퍼질 수 있다.
- k_e 는 공개 키**public key**입니다.
- k_d 는 개인 키**private key**입니다.
- N은 무작위로 선택된 두 개의 큰 소수 p와 q의 곱입니다(예: p와 q는 각각 512비트임).
- 암호화 알고리즘은 $E_{ke,N}(m) = m^{k_e} \bmod N$ 이며, 여기서 $k_e\, k_e k_d \bmod (p-1)(q-1) = 1$을 만족.
- 암호 해독 알고리즘은 $D_{kd,N}(c) = c^{k_d} \bmod N$다.

4. 보안 도구로서의 암호화

비대칭 암호화 예

- 예를 들어. p = 7및 q = 13으로 만든다.
- 그런 다음 N = 7*13 = 91 및 (p−1)(q−1) = 72를 계산.
- 다음으로 상대적으로 72와 < 72인 k_e 를 선택하여 5를 산출.
- 마지막으로 $k_e k_d \bmod 72 = 1$ 이 되도록 k_d 를 계산하여 29를 산출.
- 우리의 열쇠

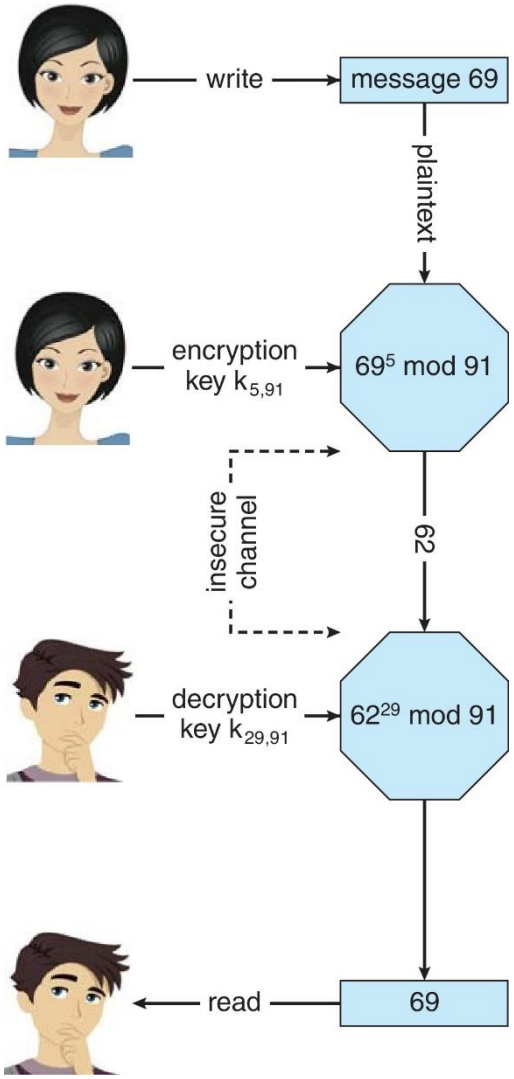
  Public key, $k_{e,N} = 5, 91$

  Private key, $k_{d,N} = 29, 91$
- 공개 키로 메시지 69를 암호화하면 암호문 62가 생성.
- 암호문은 개인키로 복호화 가능

• 공개 키는 공개 키 소유자와 통신하려는 모든 사람에게 일반 텍스트로 배포될 수 있다.

4. 보안 도구로서의 암호화

RSA 비대칭 암호화를 사용한 암호화



Public key, $k_{e,N} = 5, 91$
Private key, $k_{d,N} = 29, 91$

5. 사용자 인증

기본 개념

- **잠재적인 메시지 발신자 집합 제한**
- 암호화 보완
- 또한 수정되지 않은 메시지를 증명할 수 있다.
- **알고리즘 구성 요소**
- 키 세트 **K** keys
- 메시지 세트 **M** messages
- 인증자 세트 **A** authenticators
- 함수 $S : K \rightarrow (M \rightarrow A)$
- 즉, 각각의 $k \in K$ 에 대해 S_k 는 메시지로부터 인증자를 생성하는 함수이다.
- 모든 **k**에 대한 **S**와 S_k 는 모두 효율적으로 계산 가능한 함수여야 함.
- 함수 $V : K \rightarrow (M \times A \rightarrow \{\text{true, false}\})$. 즉, 각 $k \in K$, 에 대해 V_k 는 메시지에 대한 인증자를 검증하는 기능이다.
- 모든 **k**에 대한 **V**와 **Vk**는 모두 효율적으로 계산 가능한 함수여야 한다.

5. 사용자 인증

기본 개념

- 메시지 m에 대해 컴퓨터는 k를 소유하는 경우에만 $V_k(m, a) = \texttt{true}$ 인 인증자 $a \in A$ 를 생성할 수 있다.
- 따라서 k를 보유한 컴퓨터는 메시지에 대한 인증자를 생성할 수 있으므로 k를 보유한 다른 컴퓨터는 이를 확인할 수 있다.
- k를 가지고 있지 않은 컴퓨터는 Vk를 사용하여 확인할 수 있는 메시지에 대한 인증자를 생성할 수 없다.
- 인증자는 일반적으로 노출되기 때문에(예를 들어 인증자는 메시지 자체와 함께 네트워크에서 전송됨) 인증자에서 k를 파생시키는 것이 실현 가능하지 않아야 합니다.
- 실질적으로 $V_k(m,a) = \texttt{true}$ 이면 m이 수정되지 않았으며 메시지 전송에 k가 있음을 알 수 있다.
- 단 하나의 엔터티와 k를 공유하는 경우 메시지의 출처를 알 수 있다.

5. 사용자 인증

인증 – 해시 함수

- 인증의 근거
- m에서 고정 크기의 작은 데이터 메시지 다이제스트 message digest (해시 값) 블록을 생성.
- 해시 함수 H는 m에서 충돌에 강해야 한다.
  - $H(m) = H(m')$가 되는 $m' \neq m$ 을 찾는 것이 불가능해야 한다.
- $H(m) = H(m')$이면 $m = m'$
  - *메시지가 수정되지 않다*
- 일반적인 메시지 다이제스트 기능에는 128비트 해시를 생성하는 MD5와 160비트 해시를 출력하는 SHA-1이 포함.
- 인증자로 유용하지 않음
  - *예를 들어 $H(m)$은 메시지와 함께 보낼 수 있다.*
  - *그러나 H가 알려진 경우 누군가 m을 m'으로 수정하고 $H(m')$를 다시 계산할 수 있으며 수정은 감지되지 않는다.*
  - *따라서 $H(m)$을 인증해야 한다.*

5. 사용자 인증

인증 - MAC

- 메시지 인증 코드 message-authentication code (MAC) 인증 알고리즘에 사용되는 대칭 암호화
- 비밀 키를 사용하여 메시지에서 생성된 암호화 체크섬
- 짧은 값을 안전하게 인증할 수 있다.
- 충돌 저항성이 있는 H에 대해 H(m)을 인증하는 데 사용되는 경우 먼저 해시하여 긴 메시지를 안전하게 인증하는 방법을 얻는다.
- k는 S_k 와 V_k 를 모두 계산하는 데 필요하므로 하나를 계산할 수 있는 사람은 다른 하나를 계산할 수 있다.

5. 사용자 인증

인증 – 디지털 서명

- 비대칭 키 및 디지털 서명 알고리즘 기반
- 생성된 인증자는 디지털 서명digital signatures.
- 매우 유용함 – 누구나 메시지의 진위를 확인할 수 있음
- 디지털 서명 알고리즘에서 k_v에서 k_s 를 도출하는 것은 계산상 불가능합니다.
  - V는 단방향 함수.
  - 따라서 k_v 는 공개 키이고 k_s 는 개인 키입니다.
- RSA 디지털 서명 알고리즘 고려
  - RSA 암호화 알고리즘과 유사하지만 키 사용이 반전됨
  - 메시지의 디지털 서명 $S_{ks}(m) = H(m)^{k_s} \bmod N$
  - 키 k_s 는 다시 한 쌍 (d, N). 여기서 N은 무작위로 선택된 두 개의 큰 소수 p와 q의 곱. 검증 알고리즘은 $V_{kv}(m, a) \quad (a^{k_v} \bmod N = H(m))$
  - 여기서 k_v 는 $k_v k_s \bmod (p-1)(q-1) = 1$을 충족.

5. 사용자 인증

키 분배

- 대칭 키 전달은 큰 과제.
- 때때로 대역 외 수행out-of-band
- 비대칭 키가 확산될 수 있음 - 키 링key ring에 저장됨
- 비대칭 키 배포도 주의가 필요. 중간자 공격

디지털 인증서

- 누가 또는 무엇이 공개 키를 소유하는지 증명
- 신뢰할 수 있는 당사자가 디지털 서명한 공개 키
- 신뢰할 수 있는 당사자는 엔티티로부터 신원 증명을 받고 공개 키가 엔티티에 속함을 인증.
- 인증 기관 Certificate authority 은 신뢰할 수 있는 당사자. 공개 키는 웹 브라우저 배포에 포함되어 있.
- 키를 디지털 서명하는 등을 통해 다른 기관을 보증.