1 웹의 탄생

1 월드와이드웹(WWW)

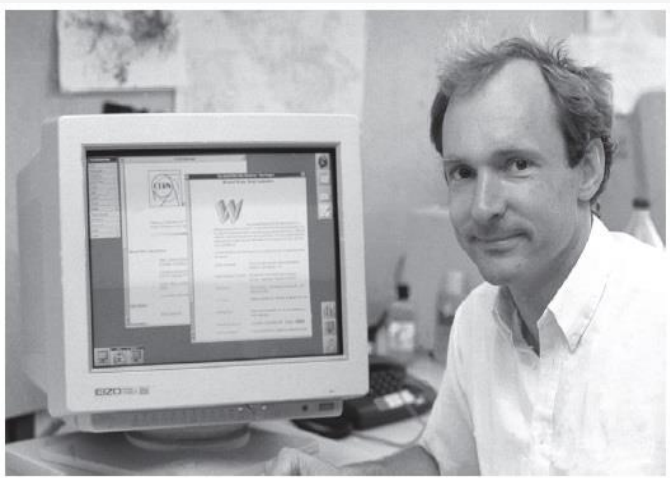
- 인터넷에 연결된 컴퓨터들이 하이퍼텍스트 형식으로 표현된 다양한 정보를 효과적으로 이용할 수 있도록 구성한 전 세계적인 시스템(HTML, HTTP)
- 간단히 웹이라고 부름(Internet vs. Web)

2 웹의 탄생

- 1989년 3월 13일, 유럽입자물리연구소(CERN)에 근무하던 소프트웨어 공학자 팀 버너스 리가 과학자들 사이에 쉽게 정보를 주고받기 위한 목적으로 정보 관리 제안서를 발표 (최초의 인터넷 기반 하이퍼텍스트 프로젝트)
- 이후 1990년에 하이퍼텍스트 브라우저와 편집기가 개발되고 URL, HTTP, HTML이 차례대로 설계됨(URI)

2 웹의 탄생

- 1991년 8월 팀 버너스 리는
  월드와이드웹의 개념을 포함한 사이트를 일반인에게
  최초로 공개하고, 로열티를 포기



팀 버너스 리

※ 출처 : 인터넷 해킹과 보안, 김경곤,
한빛아카데미, 2017

3 초창기 웹

- 단순한 텍스트와 링크 위주

하이퍼텍스트

- 글자에 링크를 걸어놓고 클릭하면 다른 화면이 나타나는 것(비순차적)

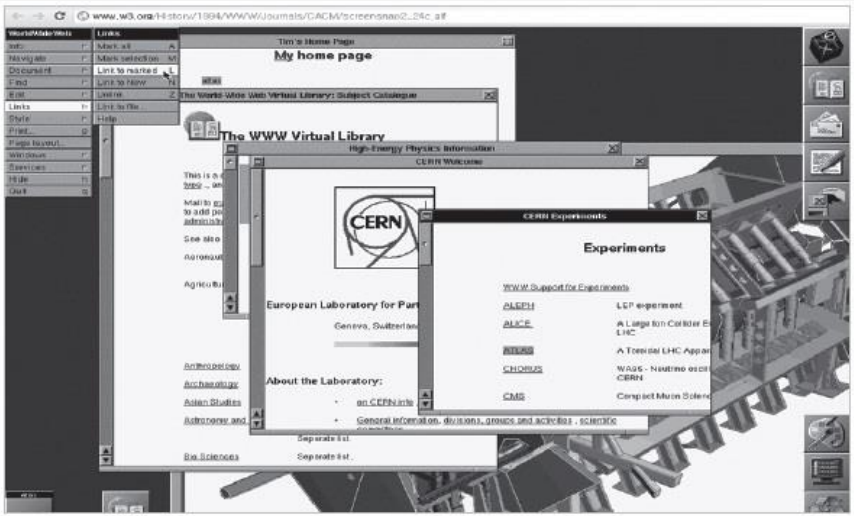
하이퍼링크

- 한 페이지에서 다른 페이지의 문서로 쉽게 이동

웹 서핑, 웹 브라우징

- 하이퍼링크를 따라 이동하는 것

3 초창기 웹



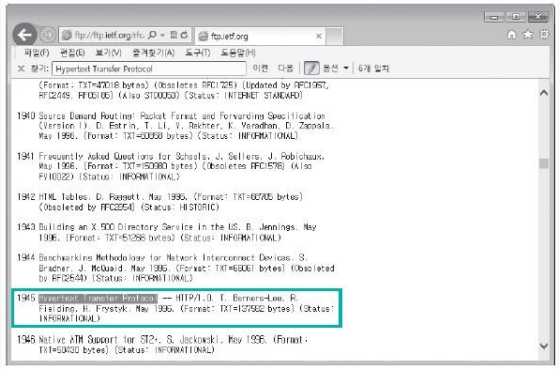
※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

팀 버너스 리가
만든 초창기
브라우저 화면

2 HTTP의 기본 개념

1 Request

- HTTP(Hypertext Transfer Protocol)
  - 인터넷에서 가장 많이 사용하는 프로토콜
    (팀 버너스 리가 웹을 만들면서 개발)
  - 문서 간의 상호 연결을 통해 다양한 텍스트, 그래픽,
    애니메이션을 화면에 보여주고 사운드를 재생

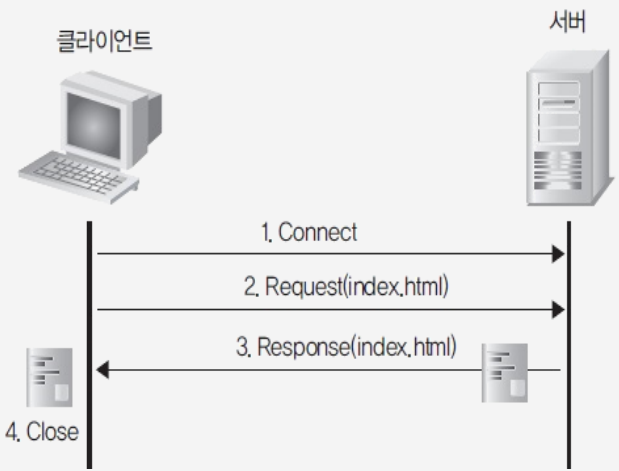


RFC-INDEX에서
HTTP 1.0 번호

※ 출처 : 인터넷 해킹과 보안, 김경곤,
한빛아카데미, 2017

1 Request

- HTTP(Hypertext Transfer Protocol)
  - 0.9 버전의 HTTP는 서버에서 단순히 읽기 기능만 지원



HTTP 0.9를 이용하여 서버에 연결하기

※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

1 Request

- HTTP(Hypertext Transfer Protocol)
  - 현재 웹에서 주로 사용하는 HTTP는 1.0과 1.1 버전 (메소드?)

HTTP 1.0	HTTP 1.1
1996년 5월에 완성되었으며, 메소드는 GET, HEAD, POST 방식만 지원	2001년에 공식 발표되어 메소드는 OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, CONNECT 방식 지원

VS

1 Request

Request

```
GET / HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Accept-Encoding: gzip, deflate
Cookie: HSID=AaxlkKoV2snlEi6UQ; SSID=AAYVu_evC0Tiu3aVc;
APISID=JEWp0eojRTLFtYKJ/ACgEWh_0mL8Li_-fl;
SAPISID=kabRBO-uT0ebDcfc/A2byDX--FwN649tHw
Host: www.google.comConnection: Keep-Alive
Accept-Language: ko-KR
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)
```

1 Request

- Request
  - 첫 번째 줄 : GET / HTTP /1.1
  - HTTP 전송 방법 : 웹 서버로부터 자료를 가져오는 기능을 하는 GET을 많이 사용(GET 메소드는 별도의 메시지 보디를 필요로 하지 않음)(vs. POST)
  - 요청된 URL
    : 웹 서버에 있는 자료를 요청할 때 사용되는 경로 (vs. URI)
  - HTTP 버전
    : 인터넷에서 가장 일반적으로 사용되는 HTTP 버전은 1.0과 1.1(대부분의 브라우저는 초기값으로 1.1 사용)(1.0 vs. 1.1)

1 Request

- 웹 해킹과 관련된 요소
  - 서버가 클라이언트에 전송한 인자값에 추가 정보를 보낼 때 사용

```
Cookie: HSID=AaxlkKoV2snlEi6UQ; SSID=AAYVu_evC0Tiu3aVc;
APISID= JEWp0eojRTLFtYKJ/ACgEWh_0mL8Li_-fl;
SAPISID=kabRBO-uT0ebDcfc/A2byDX--FwN649tHw
```

  - URL 주소에 나타난 호스트명을 자세하게 나타내기 위해 사용

```
Host: www.google.com
```

1 Request

- 웹 해킹과 관련된 요소
  - 브라우저나 기타 클라이언트의 소프트웨어 정보를 보여줌

```
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1;Trident/5.0)
```

  - GET 방식은 요청 데이터에 대한 인수를 URL을 통해 웹 브라우저로 전송(링크 주소만 알아도 연결된 페이지의 내용 확인 가능)(URI)

```
http://www.hanb.co.kr/edu/view_detail.html?hi_id=363
```

1 Request

- POST
  - HTTP의 보디(body) 영역에 소켓을 이용하여 데이터 전송
  - URL을 통해 인수값을 전송하지 않기 때문에 다른 사람이 링크를 통해 해당 페이지를 볼 수 없음
  - 보내려는 인자값이 URL을 통해 노출되지 않아 보안 측면에서 GET 방식보다 안전

1 Request

- 일반적인 게시판에서의 GET 방식과 POST 방식의 사용
  - 목록이나 글을 보는 화면에는 접근 자유도를 부여하기 위해 GET 방식 사용
  - 글을 저장 · 수정 · 삭제하는 작업을 할 때는 보안을 위해 POST 방식 사용

1 Request

POST 메소드의 예(DoS)

```
POST / HTTP/1.0
Accept: */*X-CI: 126323033
X-AT: OVERNET
X-GO: 1;KR;842;9530
X-DM: www.google.co.kr
X-SP: 762
Host: dr1.webhancer.com
Content-Length: 0
Pragma: no-cache
Connection: Close
```

1 Request

- Request 패킷의 메소드
  - HEAD
    - 서버 쪽 데이터를 검색하고 요청하는 데 사용
  - OPTIONS
    - 자원에 대한 요구- 응답 관계에서 관련된 선택 사항에 대한 정보를 요청할 때 사용
  - PUT
    - 메시지에 포함되어 있는 데이터를 지정한 URI 장소에 지정된 이름으로 저장

1 Request

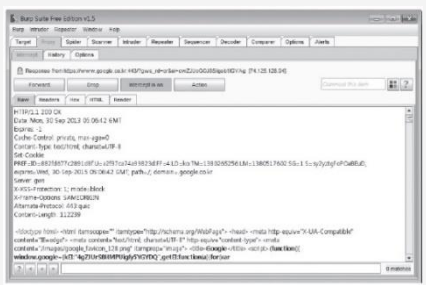
- Request 패킷의 메소드
  - DELETE
    - URI에 지정되어 있는 자원을 서버에서 지울 수 있게 함(지움)
  - TRACE
    - 요구 메시지의 최종 수신처까지 루프백 검사용으로 사용

2 Response

- 클라이언트가 보낸 Request의 응답 패킷으로 형식이 간단함
- Response 패킷에 담긴 주요 내용은 서버에서 쓰이는 프로토콜 버전, HTTP 상태 코드(200 OK) 등이며, 전달할 데이터의 형식, 데이터 길이 등과 같은 추가 정보가 포함되어 있음(해킹)

※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

Response
응답 화면



2 Response

HTTP 상태 코드

일반적인 상태 코드

상태 코드	함축적 의미	설명
100번대	정보 전송	임시 응답을 나타내는 것은 Status-Line과 선택적인 헤더로 이루어져 있고 빈 줄로 끝을 맺는다. HTTP 1.0까지는 계열에 대한 어떤 정의도 이루어지지 않았기 때문에 시험용 외에는 서버 쪽의 추가 응답이 없다.
200번대	성공	클라이언트의 요청이 성공적으로 수신되어 처리되었음을 의미한다.
300번대	리다이렉션	클라이언트의 요구 사항을 처리하려면 다른 곳에 있는 자원이 필요하다는 것을 의미한다.
400번대	클라이언트 측 에러	클라이언트가 서버에 보내는 요구 메시지를 완전히 처리하지 못한 경우처럼 클라이언트 측에서 오류가 발생한 것을 의미한다.
500번대	서버 측 에러	서버 자체에서 생긴 오류 상황이나 클라이언트의 요구 사항을 제대로 처리할 수 없을 때 발생한다.

※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

2 Response

HTTP 세부적인 상태 코드

상세한 상태 코드

상태 코드	의미	상태 코드	의미
100	Continue	404	Not Found
101	Switching Protocols	405	Method Not Allowed
200	OK	406	Not Acceptable
201	Created	407	Proxy Authentication Required
202	Accepted	408	Request Time-Out
203	Not-Authorized Information	409	Conflict
204	No Content	410	Gone
205	Reset Content	411	Length Required
206	Partial Content	412	Precondition Failed

※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

2 Response

HTTP 세부적인 상태 코드

상세한 상태 코드

300	Multiple Choices	413	Request Entity Too Large
301	Moved Permanently	414	Request URI Too Large
302	Moved Temporarily	415	Unsupported Media Type
303	See Other	500	Internal Server Error
304	Not Modified	501	Not Implemented
305	Use Proxy	502	Bad Gateway
400	Bad Request	503	Service Unavailable
401	Unauthorized	504	Gateway Time-Out
402	Payment Required	505	HTTP Version not Supported
403	Forbidden		

※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

2 Response

- HTTP 세부적인 상태 코드 설명
  - 200 OK
    - 클라이언트의 요청이 성공했다는 것을 나타냄
  - 201 Created
    - 클라이언트의 PUT 요청이 성공적이라는 것을 나타냄
  - 301 Moved Permanently
    - 브라우저의 요청을 다른 URL로 항시 전달

2 Response

- HTTP 세부적인 상태 코드 설명
  - 302 Moved Temporarily
    - 브라우저의 요청을 임시 URL로 바꾸고 Location 헤더에 임시로 변경한 URL의 정보를 적음(클라이언트가 다음에 같은 요청을 하면 기존 URL로 돌아감)
  - 304 Not Modified
    - 브라우저가 서버에 요청한 자료에 대해 서버는 클라이언트 내에 복사된 캐시를 사용하면 된다는 것을 의미(DoS)

2 Response

- HTTP 세부적인 상태 코드 설명
  - 400 Bad Request
    - 클라이언트가 서버에 잘못된 요청을 했다는 것을 나타냄
  - 401 Unauthorized
    - 서버가 클라이언트의 요청에 대해 HTTP 인증 확인을 요구
  - 403 Forbidden
    - 클라이언트의 요청에 대해 접근을 차단

2 Response

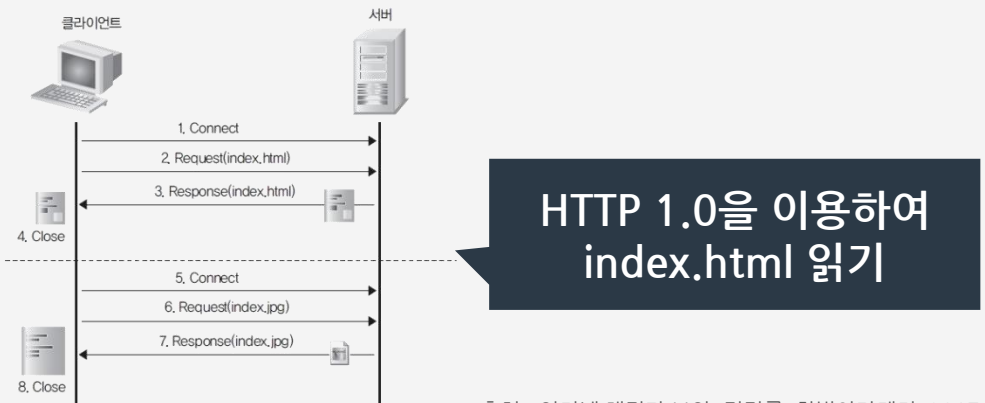
- HTTP 세부적인 상태 코드 설명
  - 404 Not Found
    - 클라이언트가 서버에 요청한 자료가 존재하지 않음
  - 405 Method Not Allowed
    - 클라이언트가 요청에 이용한 메소드는 해당 URL에 지원이 불가능함

2 Response

- HTTP 세부적인 상태 코드 설명
  - 413 Request Entity Too Large
    - 클라이언트가 요청한 보디가 서버에서 처리하기에는 너무 큼
  - 500 Internal Server Error
    - 서버가 클라이언트의 요청을 실행할 수 없을 때 500 상태 코드가 발생(SQL 인젝션 취약점이 존재하는지 확인할 때 유용)(SQL Injection)

2 Response

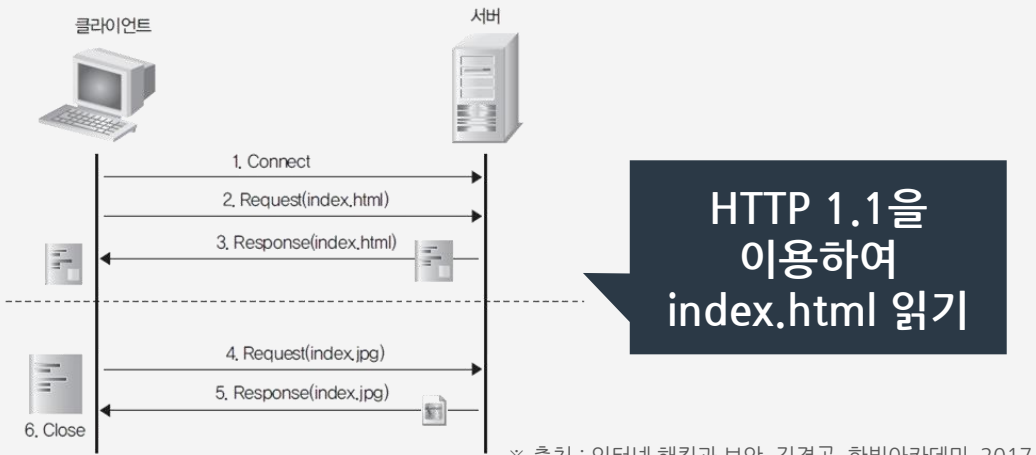
- HTTP 1.0
  - 문서에 몇 개의 그림이 있든 상관없이 텍스트가 저장된 HTML 문서를 먼저 전송 받은 후 연결을 끊고 다시 연결하여 그림을 전송 받음



※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

2 Response

- HTTP 1.1
  - 연결 요청이 계속 들어오면 HTML 문서를 받은 후 바로 그림 파일을 요청



※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

3 웹 애플리케이션 기술

1 서버 측 기능

- 초기의 웹 서버는 단순한 정적 페이지 제공
- 현재는 입력한 값에 따라 다양한 결과를 화면에 보여주는 동적 기능 제공(정적 vs. 동적 페이지)

1 서버 측 기능

- 웹 애플리케이션이 이용하는 서버 측 기능
  - ASP, JSP, PHP, VBScript,
    펄과 같은 서버 측 스크립트 언어
  - 아파치, IIS, 넷스케이프 엔터프라이즈와 같은
    웹 서버(아파치 vs. IIS)
  - Microsoft SQL Server, 오라클, 사이베이스,
    MySQL과 같은 데이터베이스(DBMS vs. 파일)

1 웹 애플리케이션 기술

- 서버 측 스크립트 언어(Server Side Script)
  - 클라이언트가 요청한 데이터를 서버 측에서 처리하여 원하는 결과를 돌려주기 위해 사용하는 언어
  - 윈도우 계열 기반은 주로 ASP, 웹 애플리케이션 플랫폼에는 주로 JSP 사용

1 서버 측 기능

웹 서버

- 일반적으로 많이 사용하는 웹 서버는 아파치와 IIS

개발사	웹 서버 수(2013년 1월 기준)	점유율(%)
아파치	348,119,032	55.26
마이크로소프트	105,619,177	16.93
nginx	79,640,472	12.64
구글	22,574,858	3.58
기타		11.59

개발사별 웹 서버 점유율

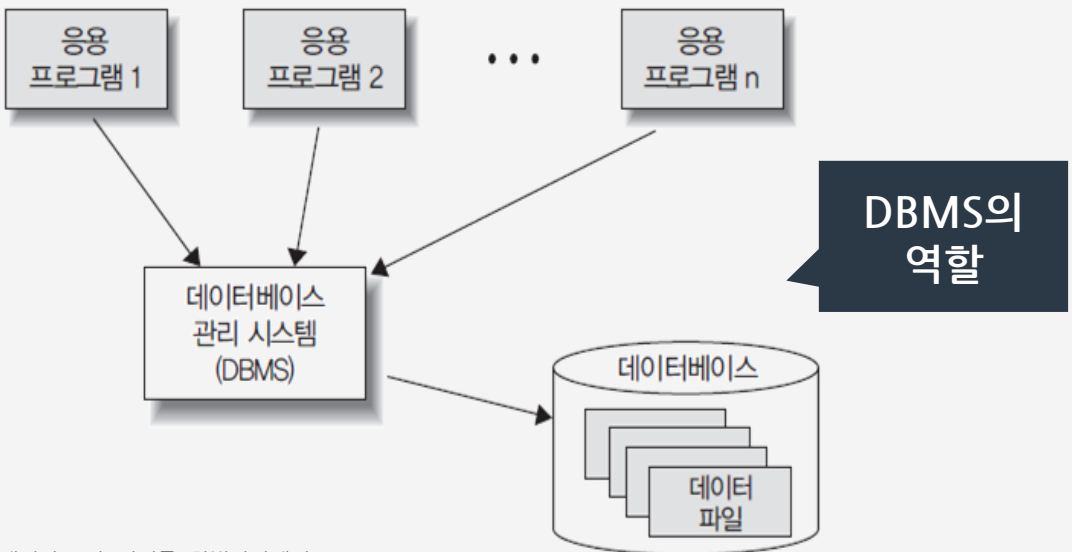
※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

1 서버 측 기능

- 데이터베이스
  - 데이터베이스 관리 시스템(DBMS)
    - 데이터베이스를 관리하는 소프트웨어
  - DBMS에는 오라클, DB2, Microsoft SQL Server, 사이베이스, MySQL 등이 있음
  - DBMS를 사용하면 데이터베이스를 만들고 데이터를 입력 · 변경 · 검색할 수 있음(파일 시스템)

1 서버 측 기능

데이터베이스



※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

2 클라이언트 측 기능

- HTML
  - 1980년 유럽입자물리연구소(CERN)의 팀 버너스 리가 HTML의 원형인 인콰이어를 제안
  - 1991년 말 팀 버너스 리가 인터넷에서 문서를 'HTML 태그'라고 부르면서 시작(tag)
  - HTML은 2000년부터 국제표준(ISO/IEC 15445:2000)이 됨
  - 2010년 8월 HTML5 Working Draft가 공개
  - 2014년 10월 28일 HTML5가 표준으로 확정 (시멘텍, 서버푸시, drag&drop)

2 클라이언트 측 기능

- 자바스크립트
  - 객체 기반의 스크립트 프로그래밍 언어
  - 넷스케이프커뮤니케이션의 운영자인 브렌던 아이크가 모카라는 이름으로 처음 개발
  - 라이브스크립트(LiveScript)라는 이름을 거쳐 자바스크립트가 됨
  - 성능 문제로 인해 서버 측에서 처리하지 않는 부분을 클라이언트 측에서 처리할 수 있도록 할 때 주로 사용(client side script)

4 암호개요

1 암호에서 사용하는 이름

- 앨리스와 밥(Alice and Bob)
  - 일반적으로 앨리스는 메시지를 전송하고 밥이 수신하는 모델에서 사용
  - 비대칭 암호 시스템인 RSA를 만든 사람 중의 하나인 Ron Rivest가 1978년에 처음으로 사용

- 이브(Eve)
  - 영어로 도청자(eavesdropper)는 소극적인 공격자를 의미
  - 이브는 앨리스와 밥 사이에 이루어지는 통신을 도청하기는 하지만 통신 중인 메시지를 수정하지는 못함

1 암호에서 사용하는 이름

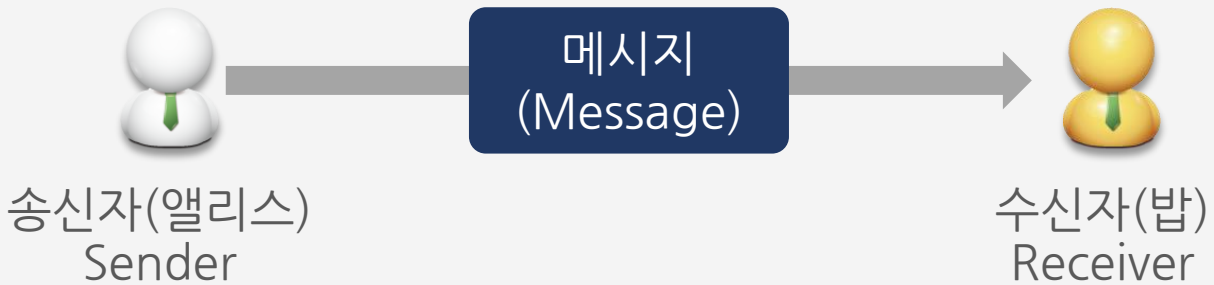
- 맬로리(Mallory)
  - 영어로 악의를 가진(malicious) 공격자를 의미
  - 이브와는 다르게 맬로리는 메시지를 수정하고, 자신의 메시지로 대체하여 이전의 메시지를 재전송할 수 있는 능력을 가지고 있음

- 트렌트(Trent)
  - 영어로 신뢰할 수 있는 중재자(trusted arbitrator)이며, 중립적인 위치에 있는 제3자
  - 사용되는 프로토콜에 따라 그 역할이 달라짐

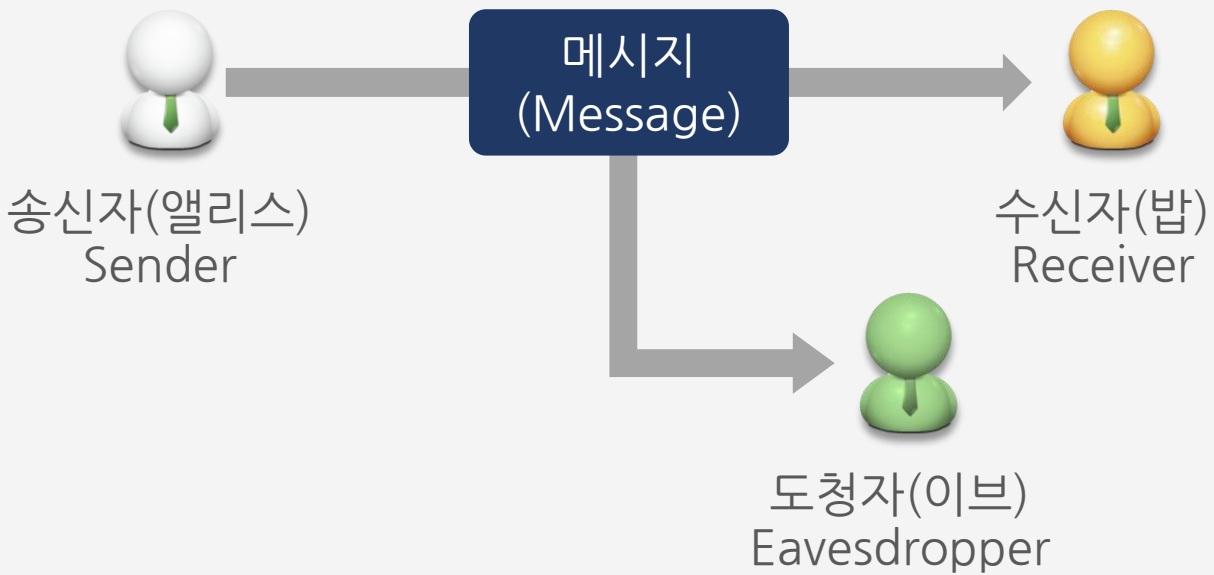
1 암호에서 사용하는 이름

- 빅터(Victor)
  - 영어명은 verifier이며 Pat이나 Peggy라는 이름을 사용하기도 함
  - 의도된 거래나 통신이 실제로 발생했다는 것을 검증할 때 등장

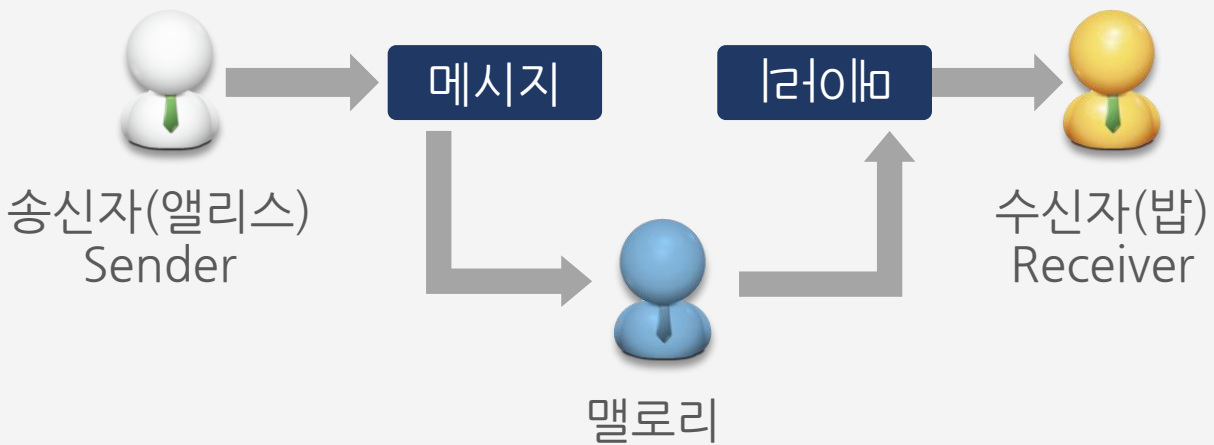
2 송신자 · 수신자 · 도청자



2 송신자 · 수신자 · 도청자


메시지
(Message)
송신자(앨리스)
Sender
수신자(밥)
Receiver
도청자(이브)
Eavesdropper

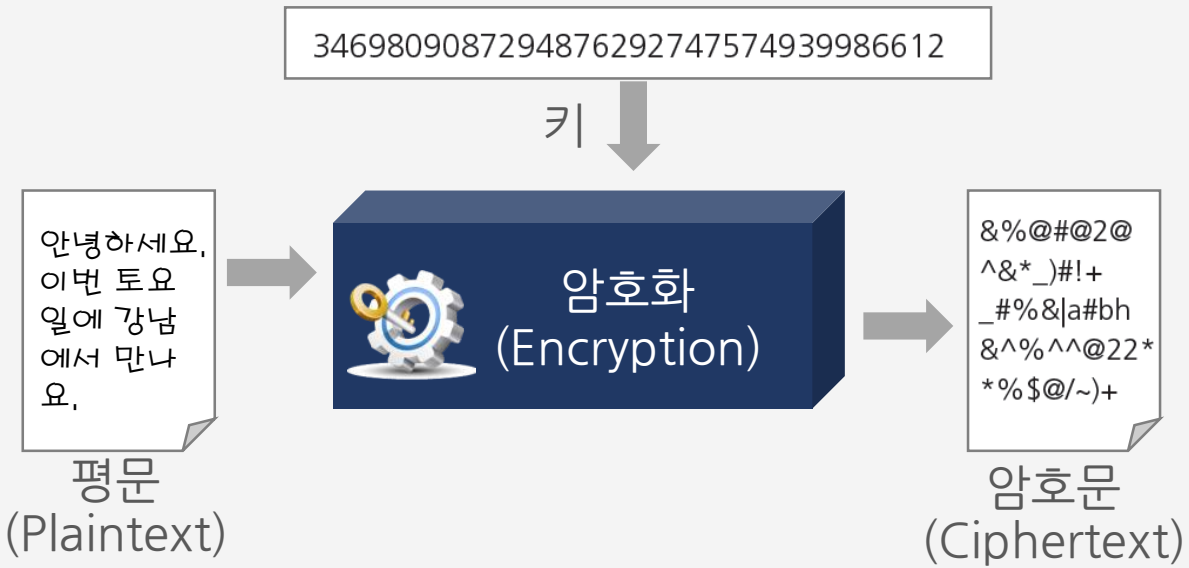
2 송신자 · 수신자 · 도청자



3 암호화와 복호화

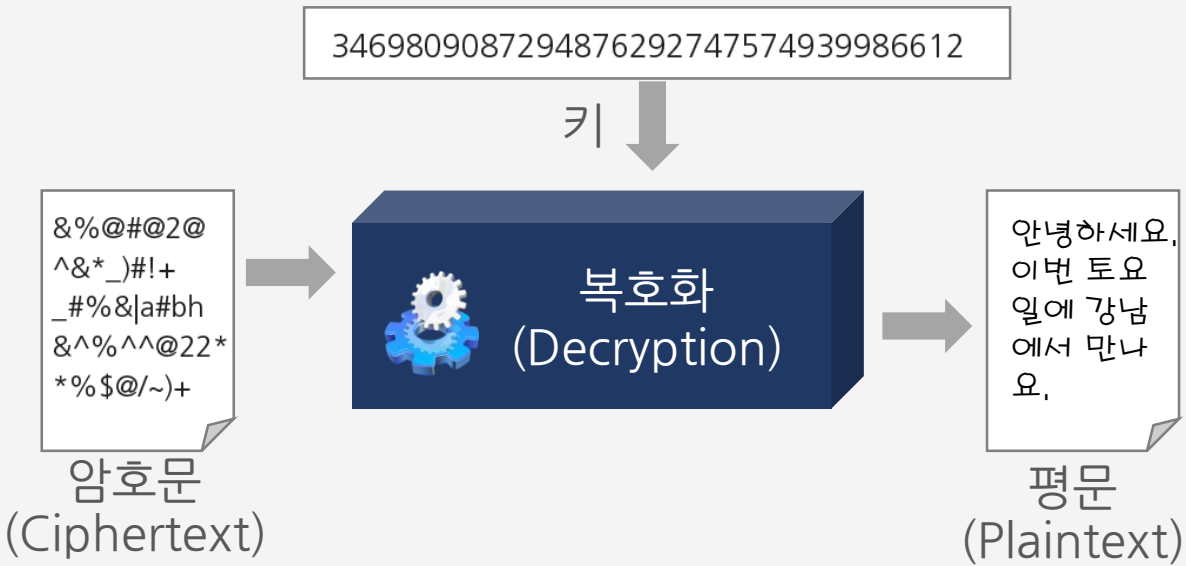
- 평문(plaintext)
  - 암호화하기 전의 메시지
- 암호문(ciphertext)
  - 암호화한 후의 메시지
- 암호기술
  - 중간에서 도청자가 암호문을 가로채어 갖게 된다고 하더라고 특정 비밀값을 모른다면 암호문을 평문으로 복호화 할 수 없도록 하는 기술

3 암호화와 복호화



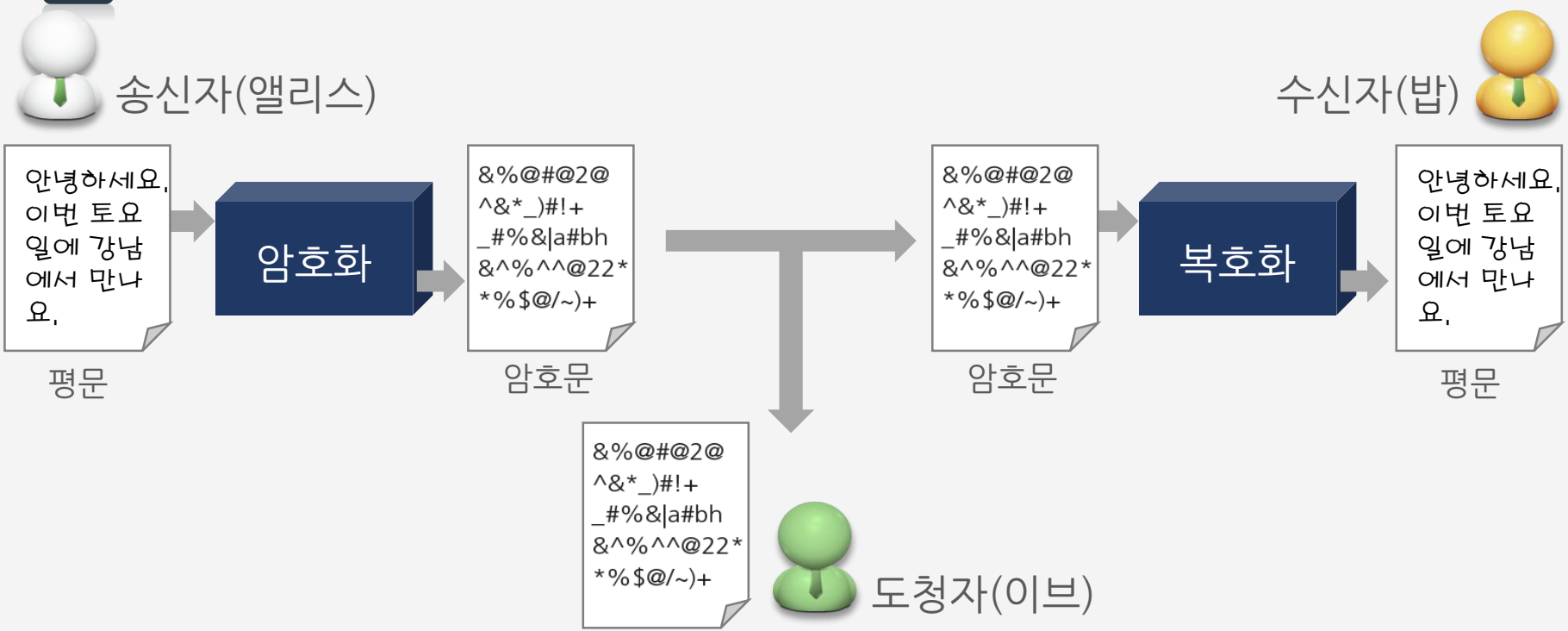
〈암호화 과정〉

3 암호화와 복호화



〈복호화 과정〉

4 암호문


송신자(앨리스)
수신자(밥)
안녕하세요, 이번 토요일에 강남에서 만나요.
암호화
&%@#@2@^&*_)#!+_#%&|a#bh&^%^^@22**%$@/~)+
복호화
평문
암호문
암호문
평문
&%@#@2@^&*_)#!+_#%&|a#bh&^%^^@22**%$@/~)+
도청자(이브)

5 암호의 기밀성

- 메일의 기밀성(confidentiality, 또는 비밀성)
  - 앨리스와 밥은 암호(cryptography) 기술을 사용해서 메일의 내용을 비밀로 유지

6 해독

- 복호화
  - 정당한 수신자가 암호문을 평문으로 바꾸는 것
- 암호 해독(cryptanalysis)
  - 수신자 이외의 사람이 암호문으로부터 평문을 복원하려고 시도하는 것
- 암호 해독자(cryptanalyst)
  - 암호 해독을 하는 사람
  - 나쁜 의도를 가진 자
  - 암호 연구자

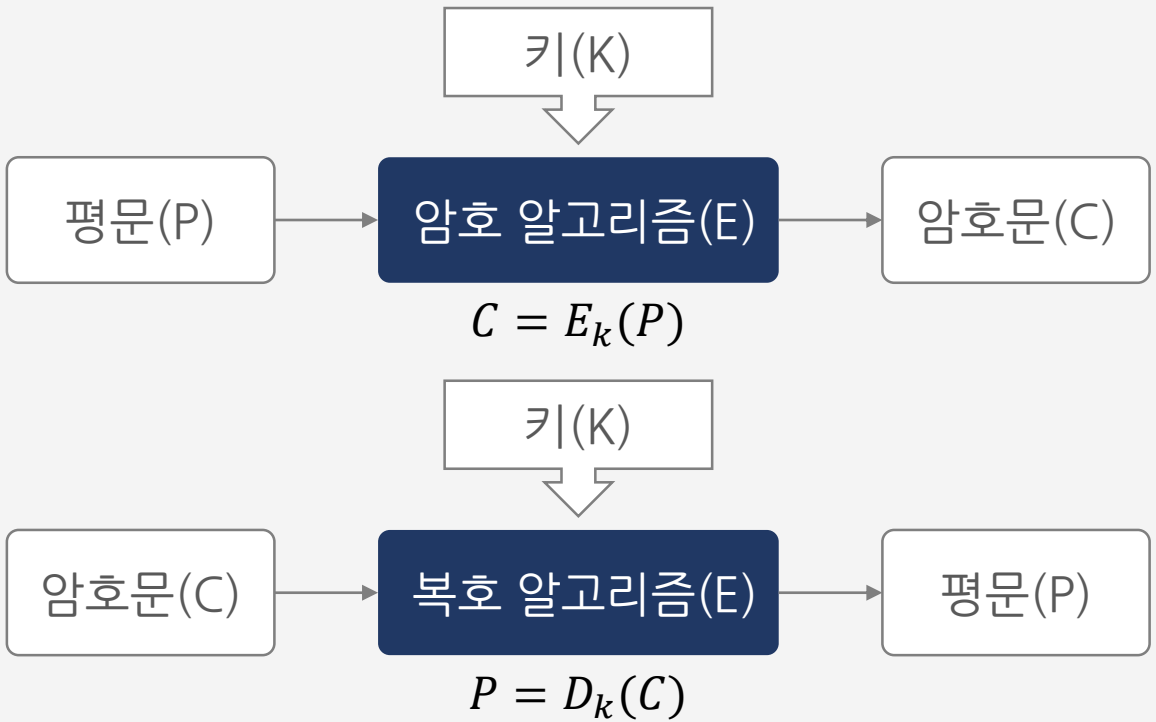
7 암호 시스템의 요소

- 평문(plaintext)
- 암호문(ciphertext)
- 암호 알고리즘(encryption algorithm)
- 복호 알고리즘(decryption algorithm)
- 키(key)

8 암호 시스템의 기호 표현

- $C = E_k(P)$
  - 평문 P를 키 K로 암호화하여(E) 암호문 C를 얻음
- $P = D_k(C)$
  - 암호문 C를 키 K로 복호화하여(D) 평문 P를 얻음
- 다른 표현
  - $C = E_k(P) = E(K, P)$
  - $P = D_k(C) = D(K, P)$

9 암호화와 복호화 알고리즘



9 암호화와 복호화 알고리즘

- 암호화 알고리즘
  - 평문을 암호문으로 만드는 절차
- 복호화 알고리즘
  - 암호문을 평문으로 만드는 절차
- 암호 알고리즘
  - 암호화와 복호화 알고리즘을 합한 알고리즘

9 암호화와 복호화 알고리즘

- 키
  - 암호 알고리즘의 키는 다음과 같은 매우 긴 숫자

203554728568477650354673080689430768

  - 2진화된 숫자로 변경하여 사용
  - 암호 키의 안전: 매우 중요!