암호화 알고리즘

데이터를 안전하게 보호하기 위해 사용되는 수학적 기법으로, 기밀성, 무결성, 인증 등의 보안 목적을 달성하기 위한 것

대칭 키 암호화 (Symmetric-key encryption)

하나의 키로 암호화하고, 같은 키로 데이터를 복호화하는 방식.

- 암호화와 복호화에 동일한 키를 사용하므로 처리속도가 빠름.
- 주로 대용량 데이터의 암호화에 사용됨.
- 키 관리가 중요하며, 안전한 키 분배가 필요.

대표적인 알고리즘: AES (Advanced Encryption Standard), DES (Data Encryption Standard), 3DES (Triple DES)

비대칭 키 암호화 (Asymmetric-key encryption)

공개 키와 개인 키라는 두개의 키를 사용하여 데이터를 암호화하고, 서로 다른 키로 데이터를 복호화 하는 방식.

- 데이터 송수신자 간의 안전한 통신을 지원.
- 공개 키는 누구나 알 수 있지만, 개인 키는 소유자만 알고 있어야 함.
- 암호화 속도가 대칭 키보다 느림.

대표적인 알고리즘: RSA (Rivest-Shamir-Adleman), ECC (Elliptic Curve Cryptography), DSA (Digital Signature Algorithm)

해시 함수 (Hash function)

임의의 길이의 데이터를 고정된 길이의 해시 값으로 변환하는 함수.

- 동일한 데이터는 항상 동일한 해시 값이 생성됨.
- 무결성 검사, 메시지 무결성 확인, 패스워드 저장 등에 사용됨.
- 해시 충돌(두개 이상의 다른 입력 값이 같은 해시 값을 생성하는 현상)에 대한

안전성이 중요함.

대표적인 알고리즘: SHA-256, SHA-3, MD5 (보안 취약점으로 현재는 사용이 권장되지 않음)

전자 서명 (Digital Signature)

데이터의 무결성을 보장하고, 데이터의 송신자를 인증하기 위해 사용됨.

개인 키로 생성된 서명은 공개 키로만 확인할 수 있음.

비밀성과 무결성을 동시에 보장함.

전자 문서, 전자 거래에서 사용됨.

대표적인 알고리즘: RSA, DSA, ECDSA (Elliptic Curve Digital Signature Algorithm)

전송 계층 보안 (TLS, Transport Layer Security)

네트워크 통신에서 데이터의 기밀성과 무결성을 보호하는 프로토콜.

대칭 키와 비대칭 키 암호화를 혼합하여 사용함.

HTTPS 등으로 웹 통신에서 많이 사용됨.

인증 및 통신 보안을 제공함

선택 기준

보안 강도: 알고리즘의 보안 수준이 얼마나 뛰어난 지 평가함.

성능: 암호화와 복호화의 속도 및 리소스 사용량을 고려함.

키 관리: 키의 생성, 분배, 저장, 회수 등을 관리하는데 필요한 복잡성을 평가함.

표준화: 산업 표준과 보안 인증을 받은 알고리즘을 선택하는 것이 좋음.

regex 정규표현식

정규표현식(Regular Expression/Regex)은 특정한 패턴을 기술하기 위해 사용되는 문자열.

주로 텍스트 검색, 문자열 처리에서 활용되며, 특정 규칙을 따르는 문자열을 찾거나 변환하는 작업에 유용함.

거의 모든 프로그래밍 언어와 텍스트 편집기에서 지원되며, 강력한 문자열 처리도구임.

기본 구성 요소

리터럴(literal): 일반적인 문자나 숫자와 정확히 일치하는 부분을 나타냄.

Ex) "abc"는 "abc"라는 문자열을 의미.

메타문자(meta-characters): 특별한 의미를 가진 문자들로, 패턴을 기술하는데 사용

'.'(마침표): 어떤 한 문자를 나타냄

'^'(캐럿): 문자열의 시작을 나타냄

'$'(달러 기호): 문자열의 끝을 나타냄

'*','+','?': 각각 바로 앞에 있는 요소를 0회 이상, 1회 이상, 0회 또는 1회 등장시키는 수량자를 나타냄.

'[]': 문자 집합을 나타내며, 대괄호 안에 들어가는 문자 중 하나와 일치함.

'()': 그룹화를 통해 일치하는 부분을 하나의 그룹으로 묶을 수 있음.

'|'(파이프): OR 연산자로, 여러 패턴 중 하나와 일치하는지 검사할 때 사용됨.

'₩': 이스케이프 문자로, 특수 문자의 원래 의미를 무효화하거나, 메타문자를 리터럴로 사용할 때 사용됨.

사용 예시

문자열 검색과 대체: 특정 패턴을 찾거나 다른 문자열로 대체할 수 있음.

데이터 유효성 검사: 이메일, 전화번호, 주민등록번호 등의 형식 검사에 사용됨.

텍스트 처리 및 추출: 특정 문자열에서 원하는 부분을 추출하거나 변경할 수 있음.

로그 파일 분석: 로그 파일에서 특정 패턴을 추출하거나 필터링할 때 유용함.

이메일 주소 패턴: `^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+₩.[a-zA-Z]{2,}$`

^ : 문자열의 시작

[a-zA-Z0-9._%+-]+ : 이메일 주소의 로컬파트로 사용할 수 있는 문자들을 허용.

하나이상의 문자가 올 수 있음.

@ : @

[a-zA-Z0-9.-]+ : 이메일 주소의 도메인 이름 파트로 사용할 수 있는 문자들을 허용.

₩. : 도메인 이름과 최상위 도메인 사이의 점을 나타냄. ₩를 사용해 일반 점표시.

[a-zA-Z]{2,} : 최소 두 개 이상의 문자로 이루어진 최상위 도메인을 나타냄

$: 문자열의 끝을 나타냄.

전화번호 패턴: ` ₩d{3}-₩d{3,4}-₩d{4} ` ex) 010-1234-5678

₩d{3} : 세 자리 숫자를 나타냄

- : 하이픈.

₩d{3,4} : 세 자리 또는 네 자리 숫자를 나타냄.

₩d{4} : 네 자리 숫자를 나타냄.

URL 프로토콜 및 도메인 추출: `^(https?://)?([a-zA-Z0-9-]+₩.)+[a-zA-Z]{2,}(/[^₩s]*)?$`

^ : 문자열의 시작.

(https?://)? : http 또는 https 프로토콜을 나타내며, 있을 수도 있고 없을 수도 있음.

[a-zA-Z0-9-]+ : 도메인 이름의 첫 번째 레벨 부분을 나타냄.

₩. : 도메인 이름과 도메인의 다음 레벨 사이의 점을 나타냄.

([a-zA-Z0-9-]+₩.)+ : 하위 도메인들을 나타냄

[a-zA-Z]{2,} : 최소 두개 이상의 문자로 이루어진 최상위 도메인을 나타냄.

(/[^₩s]*)? : 선택적으로 URL 경로를 포함할 수 있음.

$: 문자열의 끝을 나타냄

HTML 태그에서 내용 추출: `<p>(.*?)</p>`

(.*?) : 임의의 문자열(태그를 포함하지 않는)을 의미.

? 는 가능한 가장 적게 일치하도록 함.