PKI & X.509v3

https://youtu.be/HWSq3TJ6mxw

Public Key Infrastructure(PKI)

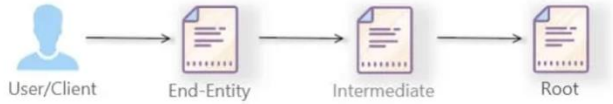
- **공개키 기반 구조(Public Key Infrastructure, PKI)**
  - **디지털 인증서**와 **공개키 암호화**를 이용하여 네트워크 상에서 안전하게 정보 교환을 할 수 있도록 지원하는 기반 및 절차들의 집합
  - **암호화된 데이터를 생성, 배포, 검증, 폐지**하고 **디지털 인증서**를 관리하기 위한 **SW, HW, 정책, 절차와 관련된 기관들을 모두 포함**하는 포괄적인 체계
  - PKI 등장 전 : 안전한 통신을 위해 사전에 대칭키를 공유하거나 신뢰를 수동으로 구축
  - PKI 등장 후 : **공개 키를 신뢰할 수 있는 방식으로 배포**하고 **사용자나 서버의 신원을 인증**할 수 있게 되어 **인터넷 상의 보안 통신** 가능
- PKI 사용 사례
  - 웹 브라우저와 서버 간의 트래픽 보호, 기업 내부망에서 사용자와 기기의 신원 확인, 이메일 및 코드 서명
- PKI는 **디지털 신뢰의 기반 인프라**로서, 공개키와 디지털 인증서 활용하여 **통신 상대의 신원을 확인**하고 **데이터를 안전하게 암호화**하여 **안전한 전자거래와 통신을 가능**하게 함

Certificate Authority

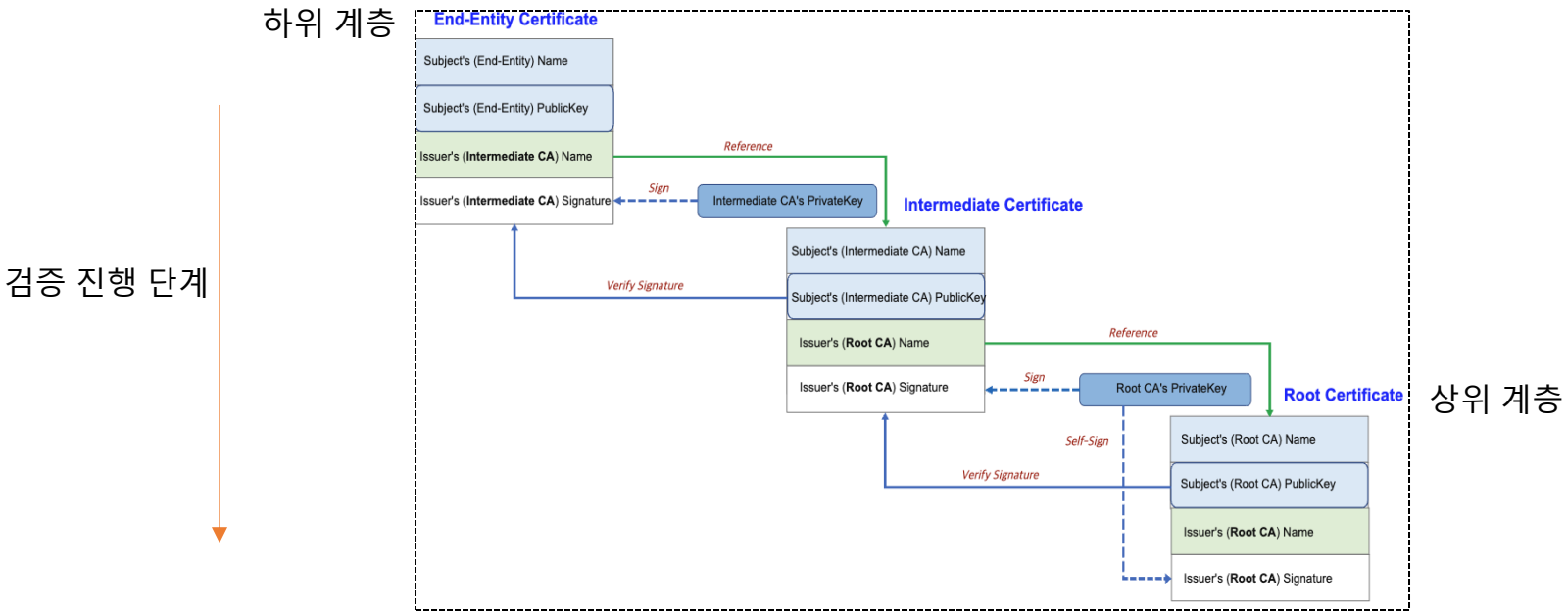
- **인증기관(Certificate Authority, CA)**
  - **PKI 체계에서 신뢰의 근간이 되는 제3자 기관**
  - 사용자, 서버, 도메인 등의 신원을 확인하고 그 신원을 공개키와 연계해 디지털 인증서로 발급하는 역할
  - **CA는 인증서를 발급할 때 '공개키-개인키'쌍을 기반으로 디지털 서명 수행**
  - 인증서 발급 단계
    1. 인증서 신청자의 신원을 여러 절차를 통해 검증
    2. 신청자의 공개 키를 포함한 디지털 인증서 발급
    - **CA는 자신의 개인키로 해당 인증서에 전자서명을 함으로써 인증서의 유효성을 담보**
    - 디지털 서명
      - CA가 발급한 인증서에 들어있는 공개키가 정말 CA가 발급했음을 증명하기 위해서 **CA는 자신의 개인키로 인증서에 서명**
      - 서명 알고리즘: **RSA, ECDSA** 등
      - 서명에 사용되는 해시 알고리즘: **SHA-256**, SHA-284, SHA-512 등
  - 위와 같이 서명된 인증서는 **CA의 공개키로 검증**될 수 있으며, 이를 통해 **해당 공개키가 검증된 신원에 속한다는 사실에 신뢰를 부여**

Chain of Trust

- PKI는 일반적으로 신뢰의 사슬(Chain of trust) 구조 사용



- Chain of Trust
  1. 최종(End-Entity)인증서(서버나 사용자 인증서)가 Root CA까지 거슬러 올라가면서 상위 인증기관의 서명을 하나씩 검증
  2. 결과적으로 시스템(브라우저/운영체제)이 신뢰하는 루트 인증서까지 도달하여 " 최종 인증서가 믿을 만하다"고 확정하는 과정



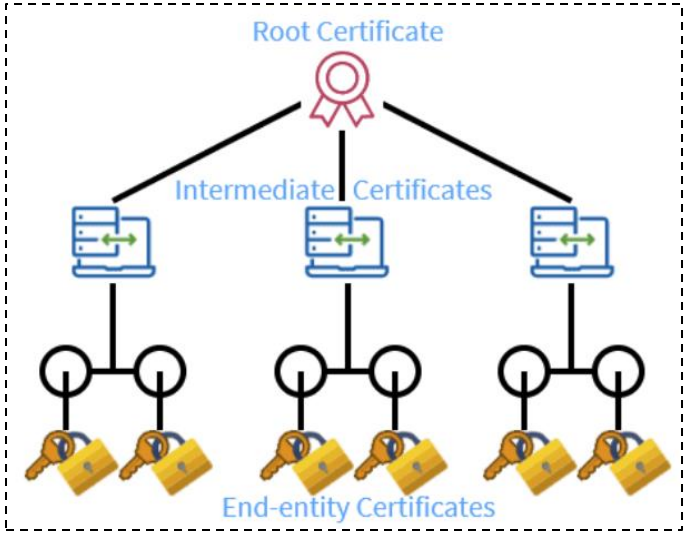
Chain of trust

https://en.wikipedia.org/wiki/Chain_of_trust
https://www.keyfactor.com/blog/certificate-chain-of-trust/

Chain of Trust 계층 구조

- **Root CA**
  - CA 체계의 최상위에 있는 인증기관
  - **자신의 개인키로 스스로 서명한 루트 인증서**를 지님
    - 운영체제나 브라우저의 신뢰 저장소(trust store)에 미리 내장되어 **모든 사용자가 기본적으로 신뢰하는 인증서**
  - **오프라인에서 엄격히 보호**
- **Intermediate CA**
  - **Root CA로부터 서명받은 중간 인증서를 보유**
  - **최종 사용자나 서버의 엔드 엔티티 인증서를 실제로 발급하는 역할**
  - 일상적인 서버/도메인 인증서 발급 업무 담당
  - 필요하면 여러 개의 Intermediate CA가 연쇄적으로 이어질 수 있음
- **End- entity Certificate(최종인증서)**
  - 실제로 웹서버나 사용자, 디바이스 등이 사용하는 인증서
  - Intermediate CA가 발급
  - 이 인증서를 사용해 HTTPS 통신 시 서버 신원 등을 증명

RootCA가 신원 확인을 거쳐 믿을 수 있다고 인정한 Intermediate CA에게 인증서 서명 권한을 위임



Chain of Trust 필요한 이유

- **보안 리스크 분산**
  - 모든 인증서를 Root CA가 직접 발급하면, Root CA의 개인키 사용 빈도가 늘어나, 유출될 위험 높아짐
  - 중간 인증 기관을 두어 Root CA는 극도로 제한된 환경에서만 사용
  - 일상적인 발급은 Intermediate CA가 담당하여 보안을 강화

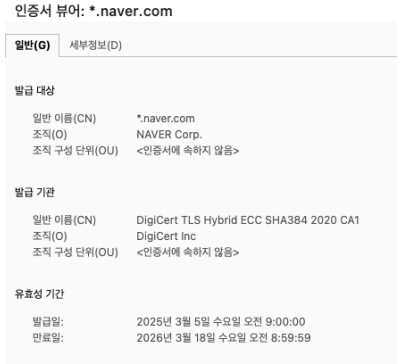
- **효율성**
  - 여러 유형의 인증서(도메인 검증, 기업 검증 등)를 전문적으로 발급하는 중간 CA를 운용하여 CA 전체의 업무 효율과 관리 체계를 개선

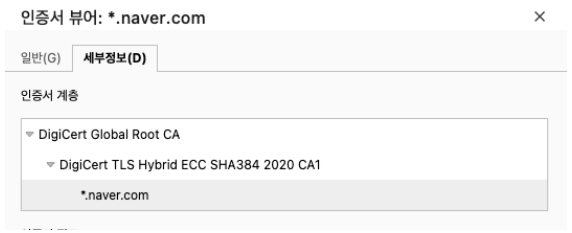
- **서명 알고리즘 적용**
  - 신뢰 체인의 각 단계에서 RSA, ECDSA 등의 공개키 알고리즘과 SHA-256 이상의 해시 함수를 사용하여 서명을 검증하기 때문에 위조 및 변조를 막고 보안을 유지

브라우저의 검증 과정 예시

- 실제로 HTTPS 사이트에 접속하면, 브라우저는 다음과 같이 **인증서 신뢰 체인**을 검증

1. 서버가 제공하는 인증서들 확인
   - 서버는 자신의 최종 인증서(End-Entity 인증서)와 함께 필요한 중간 인증서(체인 인증서)를 브라우저에 전달
2. Intermediate CA -> Root CA 순으로 검증
   - 브라우저는 '최종 인증서'가 '**Intermediate CA'의 개인키**로 올바르게 서명되었는지 검증
   - 검증 이후 'Intermediate CA' 를 다시 **'Root CA'의 공개키**로 검증
3. RootCA가 신뢰할 만한지 판단
   - 마지막으로, 검증 대상이 된 'Root CA' 인증서가 사용자 브라우저 및 운영체제의 Root 저장소에 있는 지 확인
   - 만약 ROOT 저장소에 존재한다면 최종적으로 "이 사이트는 신뢰할 수 있다"고 판단
4. 인증서의 유효 기간, 폐지 상태 등도 추가 확인
   - 단순히 서명만 검사하는 것이 아니라. 인증서가 만료되었는지, CA가 폐기하지는 않았는지(OCSP, CRL 확인)를 함께 점검



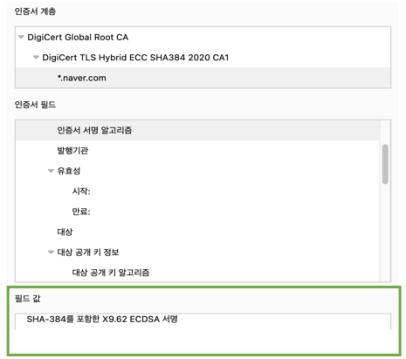


OSCP (Online Certificate Status Protocol, 온라인 인증서 상태 프로토콜) : **실시간으로 인증서의 상태**를 확인할 수 있도록 고안된 프로토콜
CRL (Certificate Revocation List, 인증서 폐지 목록) : 인증 기관은 자신이 발급한 인증서 중 더 이상 신뢰할 수 없게 된 것들을 **목록으로 만들어 서명한 후 주기적으로 배포**

https://venafi.com/blog/how-do-certificate-chains-work/

X.509

- **PKI에서 사용되는 인증서 표준 규격으로 SSL/TLS 인증서 등을 사용할 때 형식이나 구조를 정의해주는 대표적인 표준**
- X.509 인증서의 주요 요소
  - 버전(Version) : 현재 버전은 V3
  - 시리얼 번호(Serial Number)
    - 각 인증서에 부여되는 고유 식별번호
    - CA(인증기관)가 발급하는 인증서를 식별하거나 인증서 폐지 등 참조할 때 사용됨
  - 서명 알고리즘 식별자(Signature Algorithm Identifier)
    - 인증서에 서명할 때 사용된 알고리즘(**RSA with SHA-256, ECDSA with SHA-384 등**)을 표시
    - 브라우저나 OS는 식별자를 보고 적절한 방식으로 인증서 서명을 검증
  - 발행인(Issuer)
    - 인증서를 발급한 CA의 이름과 식별 정보가 담겨있음
  - 유효기간(Validity)
    - 인증서의 시작 시점과 만료 시점 정보를 담고 있으며, 유효기간은 보통 1년에서 최대 2년
  - 주체(Subject)
    - 인증서가 식별 및 보증하는 대상의 정보(**도메인 이름(CN, Common Name), 조직명(O, Organization), 개인이나 조직이 속한 국가(C, Country) 등**)가 포함
  - 주체의 공개키 정보(Subject Publick key Info)
    - **주체의 공개키와** 해당 **공개키가 사용된 알고리즘** 정보가 담겨있음
  - 확장(Extension)
    - V3 버전에서 처음 도입되었으며, Key Usage, Extended Key Usage 등이 포함
  - 서명값(Signature Value)
    - **CA가 인증서의 주요 데이터(해시)에 대해 개인키로 서명한 결과**
    - **실제 이 서명값이 유효해야 해당 인증서 신뢰할 수 있음**



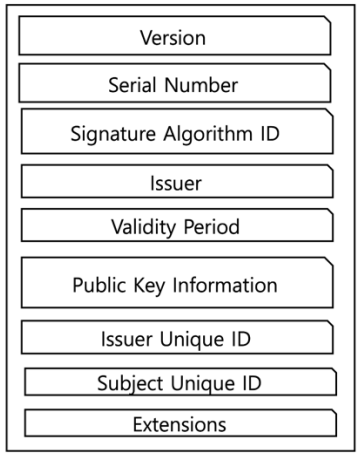
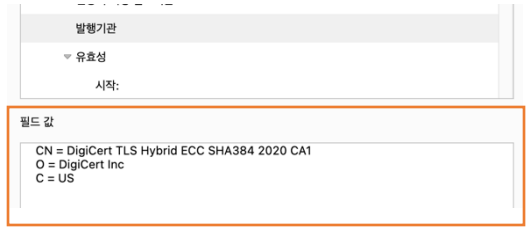


Fig. 1: Structure of X.509 Certificates.



X.509 동작 방식

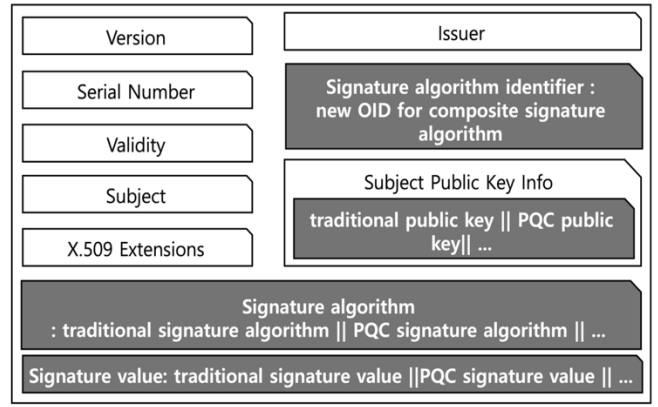
- **인증서 발급(Enrollment)**
  - 인증 대상(서버나 사용자)는 CSR(Certificate Signing Request)을 만들어 CA에 제출
    - CSR에는 **주체(Subject) 정보와 공개키 정보가 포함**되며, 인증 대상의 개인키로 서명되어 증명
  - CA는 제출받은 정보를 검증(도메인 소유 확인, 조직 검증 등)한 뒤, **X.509 형식에 맞춰 인증서 발급**
  - CA는 최종적으로 **CA의 개인키를 사용해 인증서(또는 인증서 해시)에 디지털 서명을 추가**
- **배포(Distribution)**
  - 발급받은 X.509 인증서는 서버나 사용자의 환경에 배포되어 HTTPS 설정 등에 사용됨
  - 클라이언트(브라우저)는 이 인증서를 다운받아, Chain of Trust를 통해 검증
- **검증(Validation)**
  - 클라이언트가 서버의 인증서를 받으면, **서명값을 CA의 공개키를 사용**해서 인증서가 위조되지 않았는지 확인
  - **발급한 CA인 중간 인증 기관(Intermediate CA)가 더 상위 CA(Root CA)에 의해 서명되었는지 확인 후, 최종적으로 Root CA에 도달할 수 있어야 함**
  - 인증서가 유효기간 내에 있고, **폐지 목록인 CRL에도 없으면 정상 인증서로 판단**

X.509 하이브리드 인증서

- **RSA와 ECDSA를 사용하는 X.509도 PQC 알고리즘으로 전환하는 작업이 필요함**
- **하지만 바로 PQC 알고리즘으로 전환하는 것은 현실적으로 어려움**
  - 모든 클라이언트가 PQC를 지원하기 까지 시간이 오래 걸림
  - PQC는 실무 환경에서의 **검증이 충분하지 않다**는 문제가 있음
    - **따라서, PQC를 단독으로 사용하면, 호환성 문제나 예측 불가능한 취약점이 발생할 수 있음**
- 따라서 과도기적으로 Legacy 알고리즘과 PQC 알고리즘을 사용하는 하이브리드 인증서가 고안됨
- 하이브리드 인증서
  - **하나의 인증서(X.509 포맷)에 2개 이상의 공개키와 서명 알고리즘을 포함하거나 여러 인증서를 묶어서(또는 여러 알고리즘을 동시 활용하여) 하나의 엔티티(도메인, 사용자 등)를 보증하는 방식**
  - 하나의 X.509 인증서에 Legacy 암호(RSA, ECC 등) 와 PQC 알고리즘을 동시에 포함시키는 것
  - Composite(복합) 인증서, Hybrid 인증서, Chameleon 인증서 등

X.509 하이브리드 인증서

- **Composite 인증서**
  - **하나의 X.509 인증서 안에 둘 이상의 공개키 정보(RSA/ECC + PQC)와 복합 디지털 서명을 포함하는 방식**
  - **장점**
    - **보안성 극대화:** 하나라도 안전하면 인증서 위변조 방지
    - **단일 객체 관리:** 하나의 인증서로 이중 알고리즘 운용
  - **단점**
    - **호환성 없음:** 기존 클라이언트에서 미지원 → **업데이트 필요**
    - **인증서 대형화:** 키·서명 중복으로 **크기 증가**
    - **구현 어려움:** 모든 구성서명 검증 등 **검증논리가 복잡**
  - **IETF 초안 진행:** Composite Key/Signature/KEM 초안 채택
  - **시범 구현:** OQS 등 OpenSSL 포크에서 테스트



〔그림 4〕 Schematic view of a composite certificate

Post Quantum Certificates

Post-Quantum Certificates (PQCs) are cryptographic certificates designed to be secure against potential attacks from quantum computers, which are expected to break traditional algorithms like RSA and ECC. These certificates leverage post-quantum algorithms to protect data in a future where quantum computing capabilities might threaten current cryptographic methods.

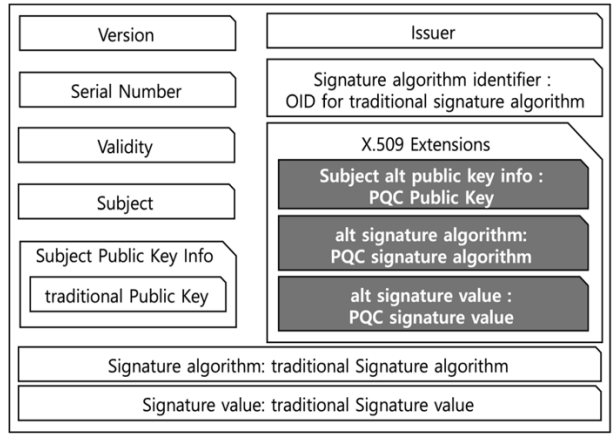
Open Quantum Safe (OQS) provides tools to integrate post-quantum algorithms into commonly used security protocols such as TLS and X.509 certificates. The project supports generating both hybrid and pure post-quantum certificates, making it possible to use combinations of traditional and post-quantum cryptography for a smoother transition during this shift.

NIST has been actively involved in selecting and standardizing post-quantum algorithms for digital signatures and key exchange. Examples of these algorithms include Dilithium for digital signatures and Kyber for key exchange, both of which have been recommended for adoption to ensure the robustness of future cryptographic solutions. In this context, QubeSec has also been working on implementing quantum-resistant certificates using these algorithms to help secure communications and sensitive data.

https://qubesec.github.io/certificates.html#:~:text=Open%20Quantum%20Safe%20,smoother%20transition%20during%20this%20shift
https://datatracker.ietf.org/doc/draft-ounsworth-pq-composite-sigs/

X.509 하이브리드 인증서

- **Hybrid 인증서**
  - **기존 인증서 필드엔 Legacy 알고리즘 사용**
  - **v3 Extensions Field에 PQC 공개키·서명 알고리즘·서명값 저장**
  - 인증서 자체는 표준 X.509 구조 유지
  - 장점
    - **완벽 호환:** 구형 시스템은 확장 무시 → 기존 방식으로 동작
    - **점진적 이행:** 하나의 인증서로 PQC 지원/미지원 모두 대응
    - **이중서명 안전:** 두 서명 중 하나가 안전하면 신뢰 유지
  - 단점
    - **구현 복잡:** 확장 필드 처리 및 이중 서명 검증 로직 추가 필요
    - **경로검증 수정:** PQC 검증 위해 PKI 소프트웨어 업그레이드 요구
    - **인증서 크기 증가:** 두 개 키/서명으로 인증서 용량 증가
  - **제품 구현:** EJBCA 등에서 지원 시작



〔그림 3〕 schematic view of a hybrid certificate

A Hybrid Certificate Authority (CA) is an X509 CA with two key pairs and two signing algorithms, where a combination of classic algorithms and PQC algorithms is used. The Alternative Signing Algorithm and the Alternative Certificate Signing Key in EJBCA are one of the PQC algorithms Dilithium or FALCON.

https://docs.keyfactor.com/ejbca/8.3.2/hybrid-ca#:~:text=A%20Hybrid%20Certificate%20Authority%20,PQC%20algorithms%20Dilithium%20or%20FALCON
https://www.ejbca.org/case/try-hybrid-certificates-with-ejbca/
https://datatracker.ietf.org/doc/draft-truskovsky-lamps-pq-hybrid-x509/

X.509 하이브리드 인증서

- **Chameleon 인증서**
  - **기본 인증서**(Base Certificate)와 **델타 인증서**(Delta Certificate) 한 쌍으로 구성
  - 기본 인증서는 기존의 RSA/ECDSA 등의 **전통적인 서명 알고리즘**으로 서명된 인증서
  - 델타 인증서는 **양자 내성(PQC) 서명 알고리즘**으로 서명된 인증서
  - **기본 인증서에는 델타 인증서를 가리키는 비필수 확장 필드(Delta Certificate Descriptor)가 포함**
    - **이 확장 필드에는 델타 인증서의 PQC 공개키, 델타 인증서의 서명 값 등이 요약되어 저장**
    - 필요 시 이를 이용해 델타 인증서를 재구성할 수 있음
  - 전통 알고리즘만 지원하는 환경에서는 Base Cert만 검증
  - PQC를 지원하는 환경에서는 Delta Cert까지 활용해 "양자 안전성"을 보장
  - **아직 표준화 진행 논의 중...**
  - 장점
    - **상황별 선택적 사용 :** 클라이언트나 서버가 **둘 중 어느 알고리즘을 쓸지**, 혹은 **둘 다 확인할지** 동적으로 결정 가능
  - 단점
    - **인증서 크기와 연산 부담 증가:** 하나의 인증서가 **둘 이상의 공개키·서명 정보를 동시에 포함**하기 때문에, 인증서 파일 사이즈가 커지고, TLS 핸드셰이크 등에서의 **검증 연산 비용**도 증가할 수 있음.
    - **구현 복잡도**

https://datatracker.ietf.org/doc/draft-bonnell-lamps-chameleon-certs/

Q & A