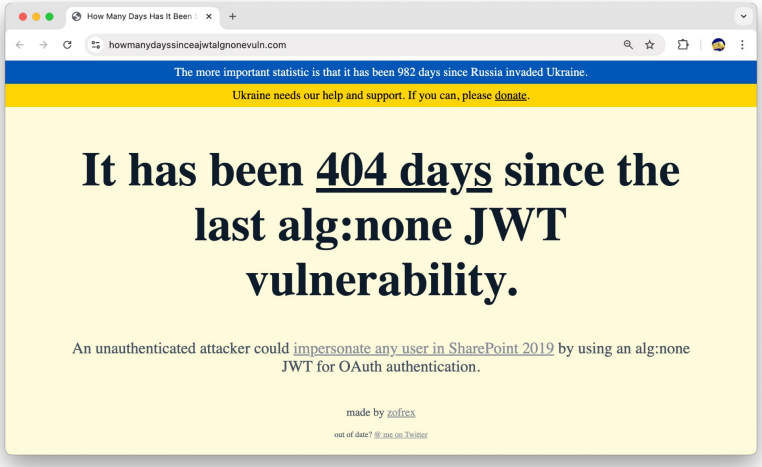
Unsigned Certificates

draft-davidben-x509-alg-none

https://www.howmanydayssinceajwtalgnonevuln.com/

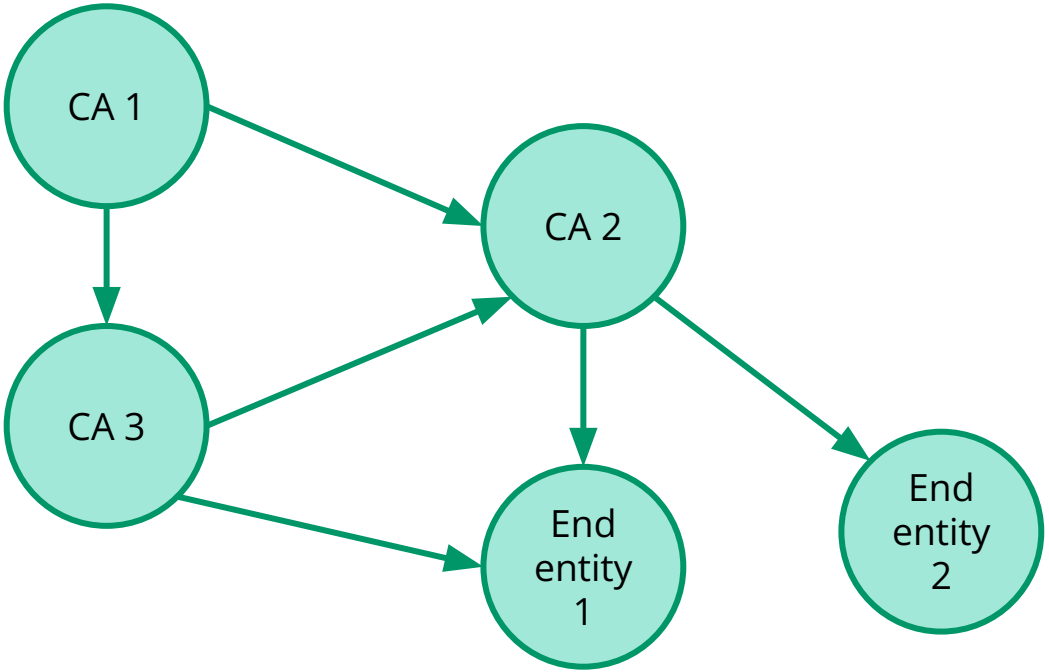


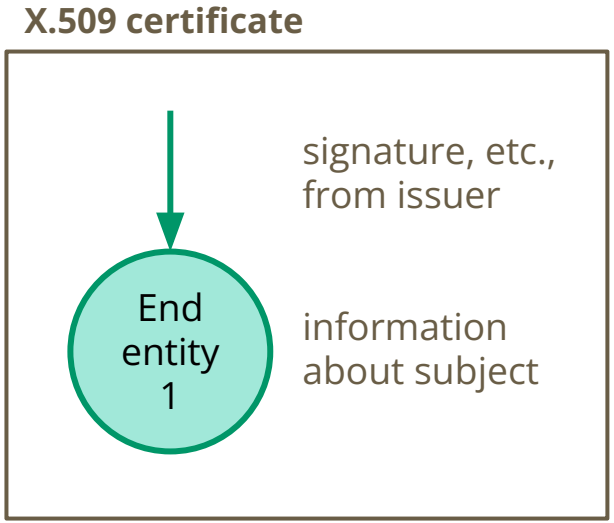
Wouldn’t it be fun to bring this to X.509?

Wait, what? Why?

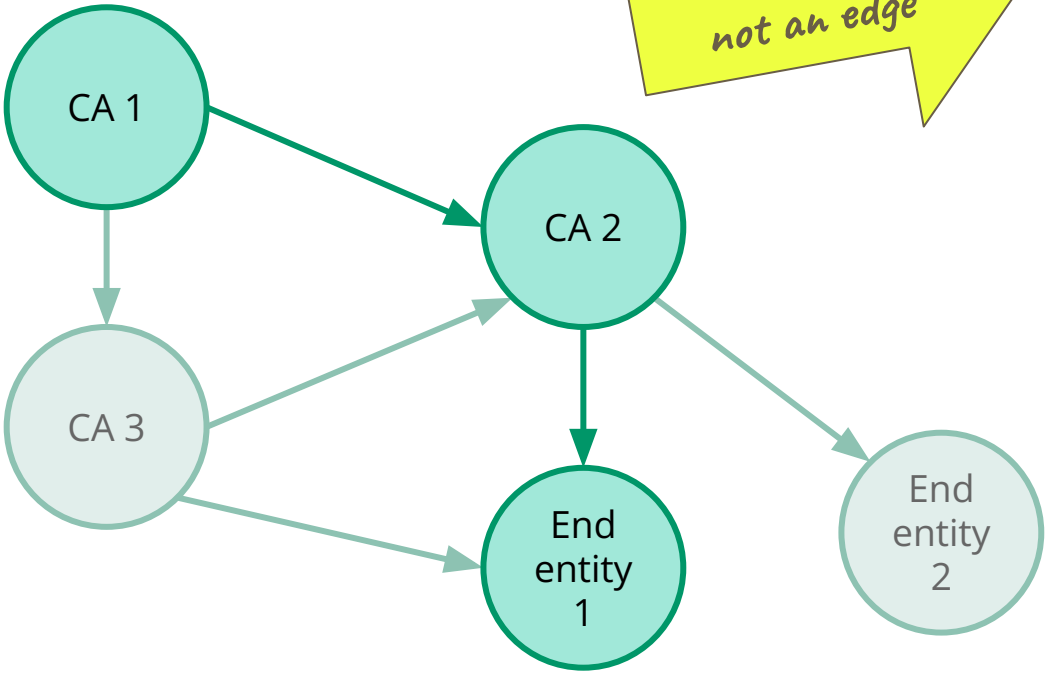
https://www.youtube.com/watch?v=cUbIkNUFs-4

Certificates are edges in the PKI graph, not nodes





Certification path

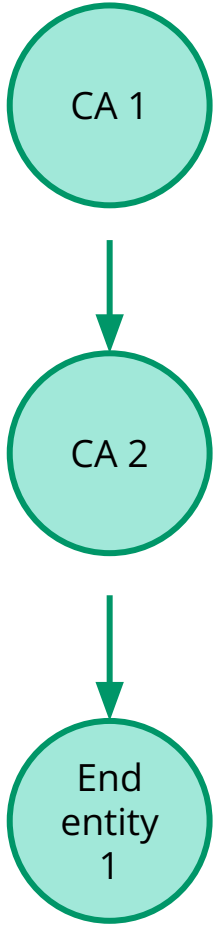


This is a node, not an edge

trust anchor (preconfigured in relying party)

intermediate certificate

end-entity certificate



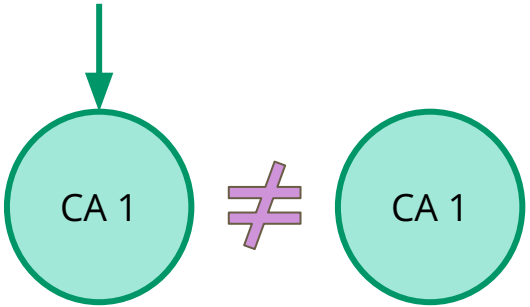
Self-signed certificates

Use cases for PKI nodes:

- Trust anchors (root CAs)
- Authenticate TLS server by fingerprint

But systems mostly take certificates (PKI edges)

Widespread solution: self-signed certificates!



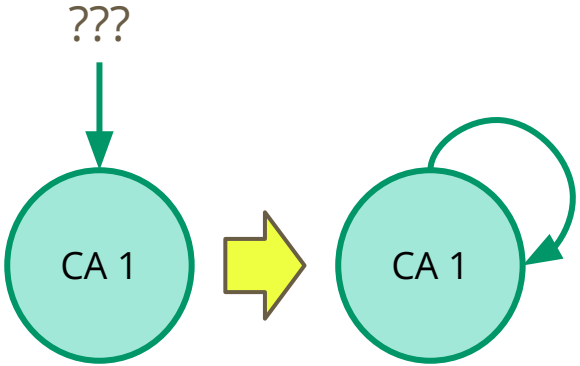
What is a self-signed certificate?

What does the signature do?

- If key trusted out-of-band, why self-attest?
- If key not trusted out-of-band, signature is irrelevant

This is an encoding hack to make PKI nodes look like PKI edges

- Self-signature has *no security value* — usually not verified at all!
- Wastes bytes with large post-quantum signatures
- Theoretical cross-protocol attack with end-entity protocol
- Not possible with KEM keys at all



Unsigned certificates

Use a less wasteful placeholder:

- New id-alg-noSignature signature algorithm
- Empty string signature

id-alg-noSignature is defined to *always be invalid*

Certificate consumers broadly *already* implement this — already expected to handle unknown signature algorithms

- If you already ignore a signature, keep ignoring it
- If you check a signature, unknown signature algorithm is rejected

Avoiding the JWT problem

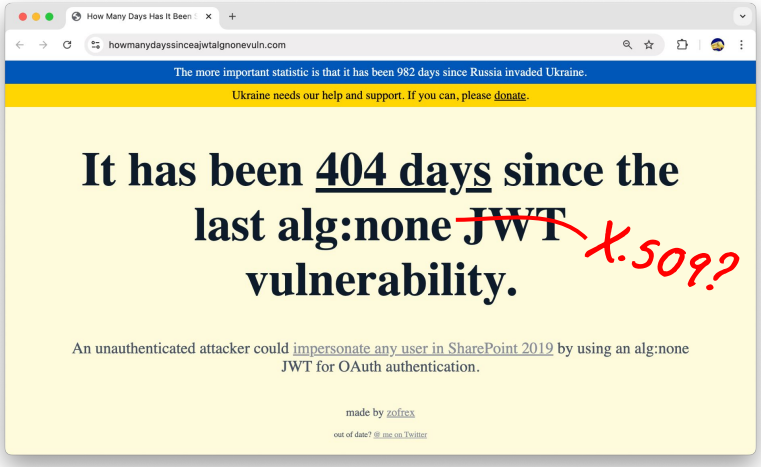
Supporting JWT alg:none means accepting them *instead of actual authentication*

- Verification is `return true`

Supporting id-alg-noSignature means *still rejecting them*

- Verification is `return false`

Only used when you *already* didn't care about the signature



4. Consuming Unsigned Certificates

X.509 signatures of type id-alg-noSignature are always invalid. This contrasts with [JWT]. When processing X.509 certificates without verifying signatures, receivers MAY accept id-alg-noSignature. When verifying X.509 signatures, receivers MUST reject id-alg-noSignature. In particular, X.509 validators MUST NOT accept id-alg-noSignature in the place of a signature in the certification path.

X.509 applications must already account for unknown signature algorithms, so applications are RECOMMENDED to satisfy these requirements by ignoring this document. An unmodified X.509 validator will not recognize id-alg-noSignature and is thus already expected to reject it in the certification path. Conversely, in contexts where an X.509 application was ignoring the self-signature, id-alg-noSignature will also be ignored, but more efficiently.

Open questions

Is this even worth doing? Can we just stop putting square pegs in round holes?

- RFC 5914: Trust Anchor Format
- RFC 7250: TLS raw public keys
- And yet everyone uses self-signed certificates…

id-alg-noSignature vs different OID?

- OID already exists… but RFC 5272 uses a redundant hash instead of empty string
- Not an *actual* conflict, but seems weird

Extend to CSRs with ignored signatures?

- Yet another square peg in a round hole
- RFC 5272 already defined this for CSRs… but there's a redundant hash in there

Next steps?