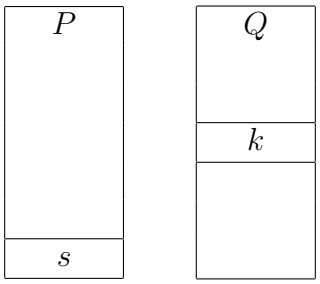
Consider two certificates P and Q where P is signed by Q. Let s be the signature in P and let k be the public key in Q.



By necessity s and k are compatible. For example, if public key k is RSA 2048 then signature s is 2048 bits in length (256 bytes). See RFC 3447 Public-Key Cryptography Standards.

A hash digest of P is contained in s. For example, the following unencrypted signature s is for RSA 2048 and hash digest SHA256. (Numerals are in hexadecimal.)

00 01 ff ··· ff 00 30 31 30 0d 06 09 60 86 48 01 65 03 04 02 01 05 00 04 20	HASH

Signature s (plaintext)

The length of HASH is 32 bytes because SHA256 is used. The 19 byte sequence starting with `30` is from "Abstract Syntax Notation One." The `ff` bytes are pad bytes that are added to make the total length of the signature 2048 bits (256 bytes). Hence there are $256 - 2 - 19 - 32 = 203$ pad bytes.

Note that the above signature is the unencrypted value of s. In the actual certificate, s is encrypted using the private key associated with k. After encryption, s is still 256 bytes long.

76 b6 97 82 0f 06 b7 48 59 02 a0 2c f4 ··· fe c3 61 25 5b 1c da 77 9a a1 63 d4 49 cd

Signature s (encrypted)

To prove that P is signed by Q, s is decrypted using Q's public key k. Then if HASH matches digest P, the signing of P by Q is proven.

Proving "P signed by Q" proves that s was encrypted using the private key associated with k. Only the owner of Q knows the private key. No one can change the contents of P without breaking the hash digest in s, and no one can change s without knowing the private key. Hence we can trust the contents of P if we trust Q.