Public Key Parameter Rules

W. E. Burr

4 December 1996

NIST

What are PK Parameters

- Constants used in public key computation
  - publicly known
  - carefully chosen
  - same parameters can be used with many keys & certificates
  - but need not necessarily be the same for all certificates and keys

Three DSS Parameters

- p, a prime modulus
  - $2^{L-1} < p < 2^{L}$ for $512 \le L \le 1024$ and L a multiple of 64
- q, a prime divisor of p-1
  - $2^{159} < q < 2^{160}$
- g, which has order q, mod p
  - $g = h^{(p-1)/q} \bmod p$, where h is any integer with $1 < h$, $p-1$, such that $h^{(p-1)/q} \bmod p > 1$

Three DSS Parameters

- Are large numbers
  - p is 512 to 1024 bits
  - q is 160 bits
  - g is 512 to 1024 bits
- total of 1184 to 2208 bits
- Substantial storage & bandwidth cost to replicate in every certificate

Why Do We Need the Rules

- Increase security
  - prevent parameter substitution attacks
- Improve interoperability
  - avoid different assumptions
- Improve performance
  - inheritance can save a lot of bandwidth
    - parameters are not repeated in every certificate

Parameters and X.509

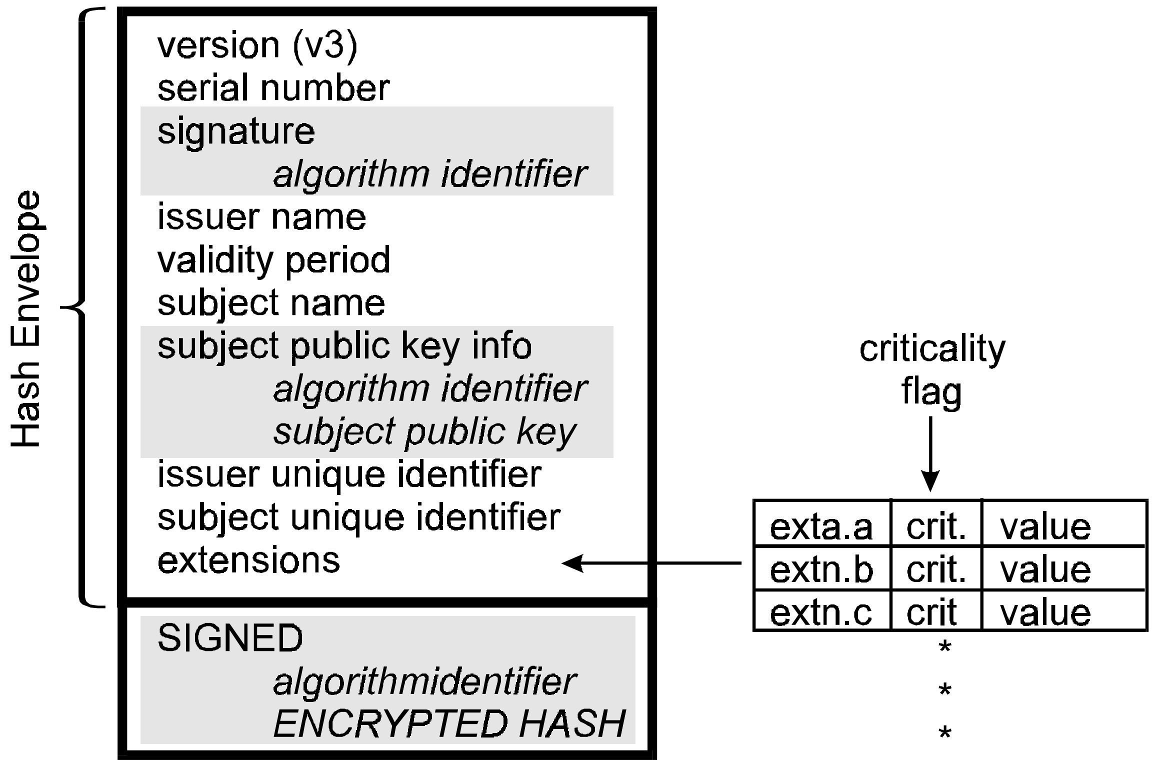
- X.509 standard is confusing
  - three places in certificate where syntax permits parameters to be stated
    - only one of these is “secure”
    - parameter substitution attack may be possible if certificate using system gets parameters from wrong place

AlgorithmIdentifier

- AlgorithmIdentifier syntax defines the algorithm and states parameters
- There are three occurrences of AlgorithmIdentifier in an X.509 certificate

```
AlgorithmIdentifier      ::=      SEQUENCE{
    algorithm            ALGORITHM.&id({SupportedAlgorithms}),
    parameters           ALGORITHM.&Type ({SupportedAlgorithms}
                         { @algorithm}) OPTIONAL }
```

X.509 v3 Certificate



The $64 Question

- Which of the three do we use to validate a digital signature?
- X.509 text doesn't state this clearly and directly
- The answer matters
  - see Chokhani paper
    - http://www.cygnacom.com/docfiles/dsaflaw.zip

The Wrong Answers

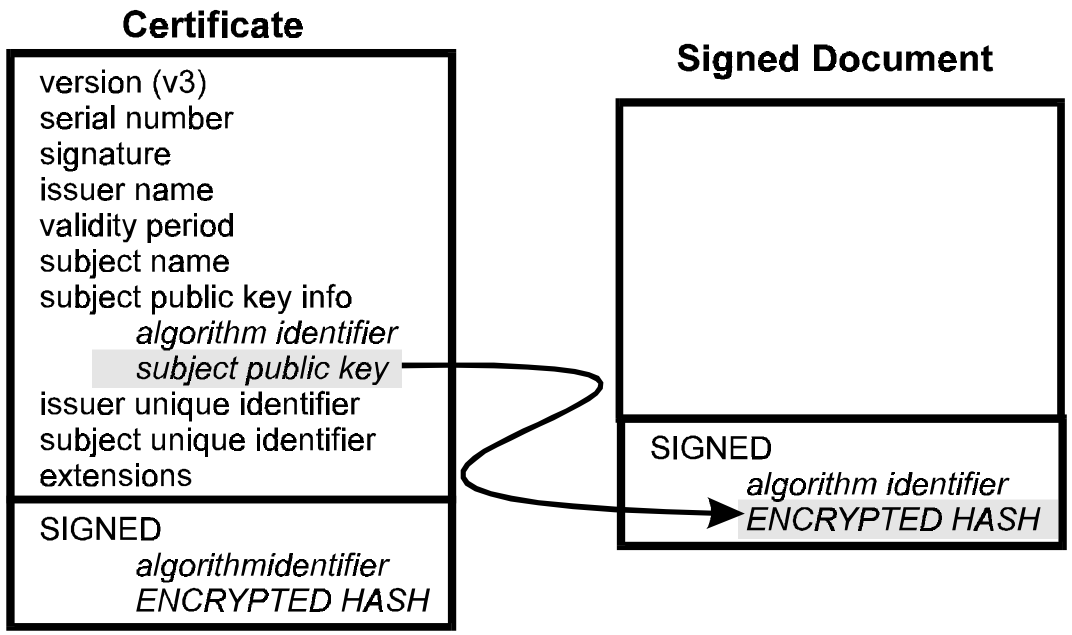
- Can’t use parameters in SIGNED itself
  - not within the protected envelope & circular
- Can’t use signature in a certificate to validate that same certificate
  - circular

The Right Answer

- We get the parameters needed to validate a signature from the same place we get the public key used to validate that signature: the subjectPublicKeyInfo field of the *signer's* certificate.

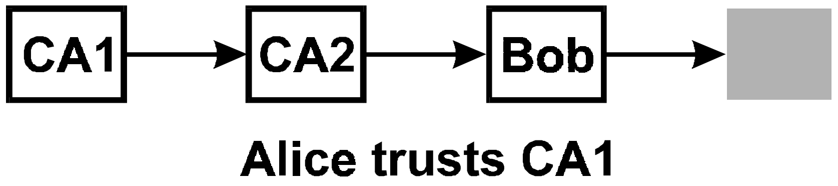
Signature Validation

- Public key used to validate a signature comes from the signer's certificate



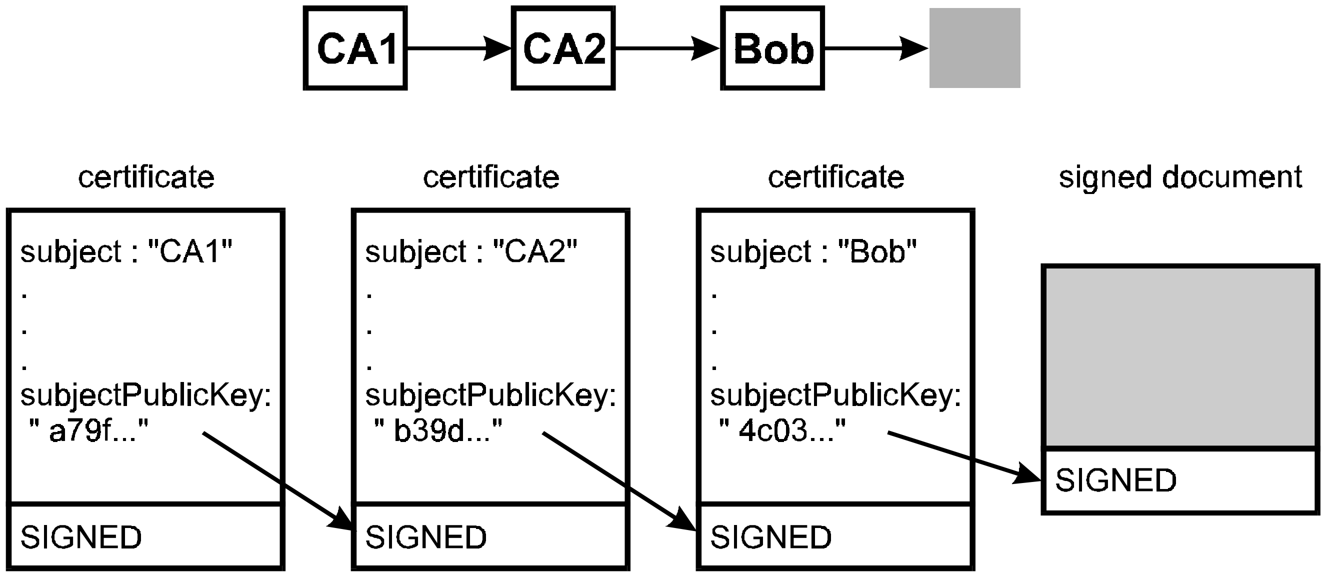
Certification Path

- Alice can verify Bob’s signature by verifying a chain of certificates starting from one issued by a Certification Authority (CA) she trusts (and whose public key she knows)



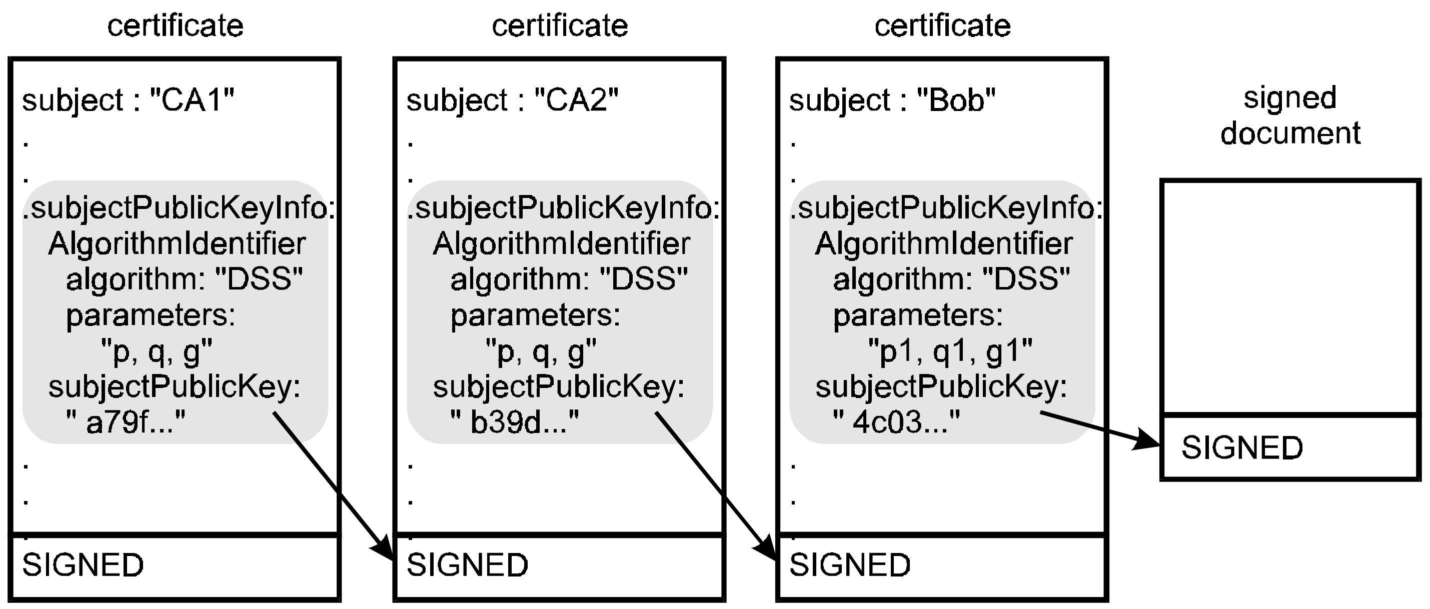
Certification Path

- A somewhat more mechanical view



Certification Path

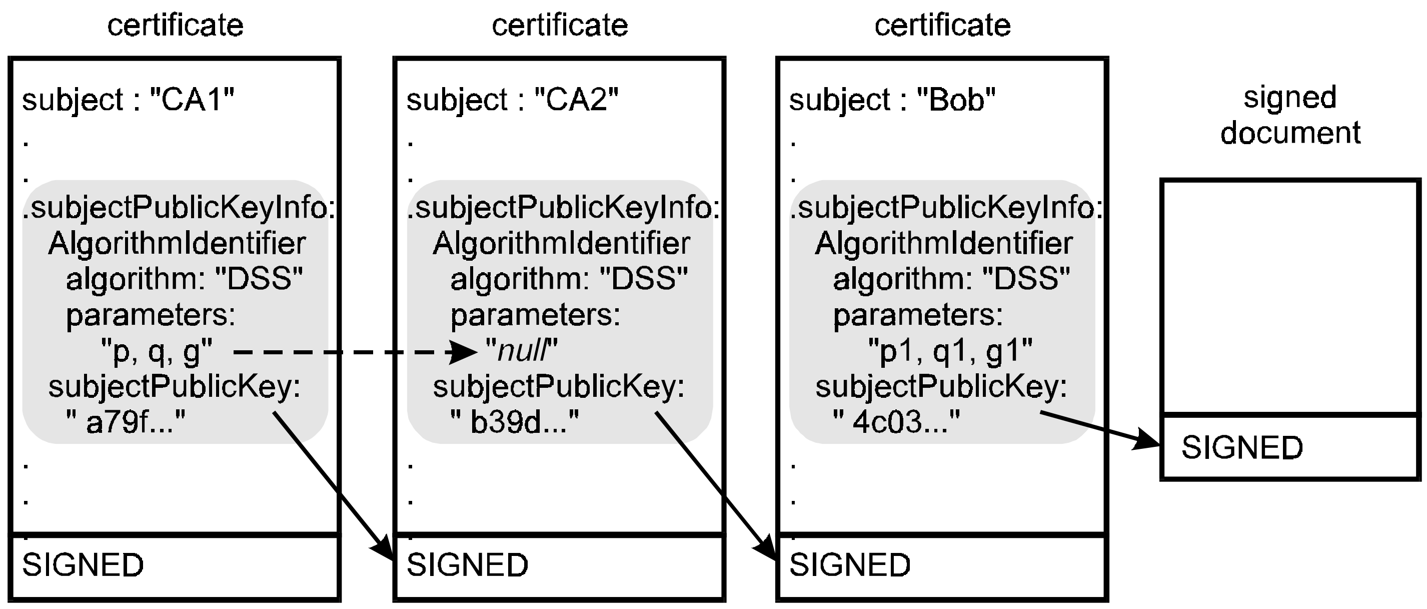
- A more detailed mechanical view



Parameter Inheritance

- What if parameters field in certificate subjectPublicKeyInfo field is null?
- Proposed answer: Inherit parameters from those used in the previous stage of the certification path
  - X.509 is silent on this subject

Parameter Inheritance



Mixed Algorithms

- CA's and subject's algorithm may differ
  - can use RSA to sign a certificate with a DSS public key
- Change of algorithm blocks parameter inheritance
  - if algorithm in subjectPublicKeyInfo field is different than algorithm used to sign certificate, parameters must be explicitly stated

Where are We Going?

- Getting parameter rules included in
  - PKIX
  - MISSI specifications
  - ISO TC 68 Draft
  - MISPC
  - X9.57 footnote
    - too late to change normative text