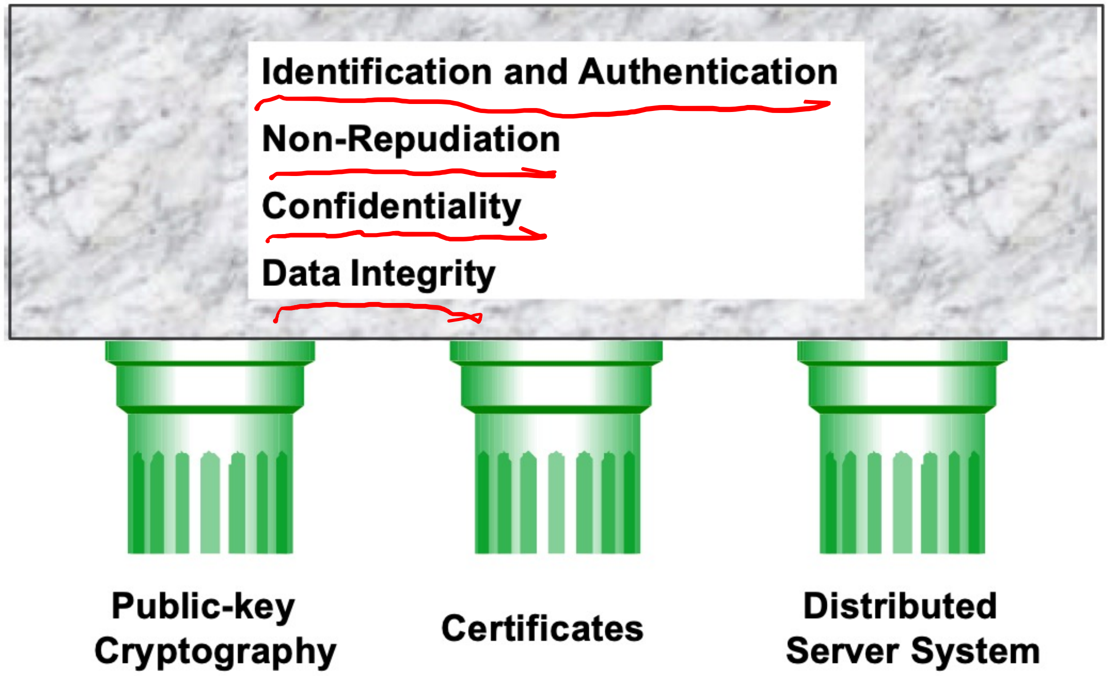
PKI and Certificates

(Section 4.5)

What is PKI?

- Use of public-key cryptography and X.509 certificates in a distributed server system to establish secure domains and trusted relationships



Identification and Authentication

Non-Repudiation

Confidentiality

Data Integrity

Public-key Cryptography

Certificates

Distributed Server System

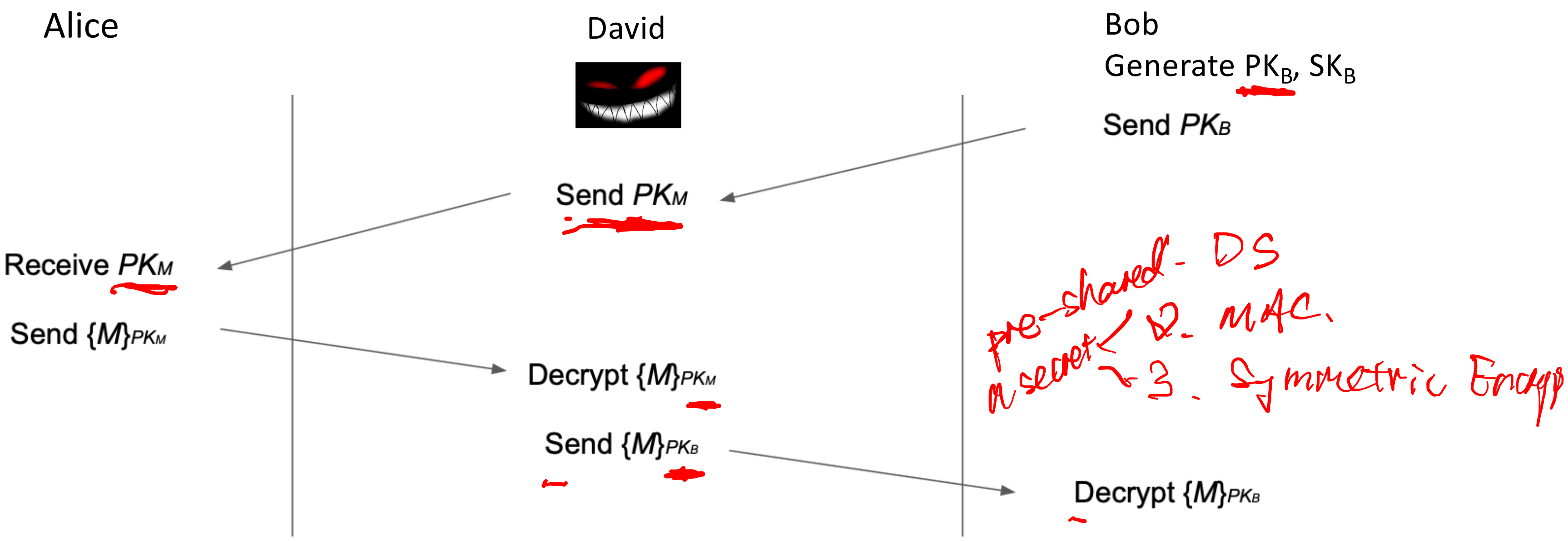
Why use public-key cryptography?

- Review: Public-key cryptography is great! We can communicate securely without a shared secret
    - Public-key encryption: Everybody encrypts with the public key, but only the owner of the private key can decrypt
    - Digital signatures: Only the owner of the private key can sign, but everybody can verify with the public key

Problem: Distributing Public Keys

- Public-key cryptography alone is not secure
- Scenario
  - Alice wants to send a message to Bob
  - Alice asks Bob for his public key
  - Bob sends his public key to Alice
  - Alice encrypts her message with Bob’s public key and sends it to Bob
- What can David do?
  - Replace Bob's public key with David’s public key
  - Now Alice has encrypted the message with David’s public key, and David can read it!

Problem: Distributing Public Keys



Alice

David

Bob

Generate PK_B, SK_B

Send PK_B

Send PK_M

Receive PK_M

Send $\{M\}_{PK_M}$

Decrypt $\{M\}_{PK_M}$

Send $\{M\}_{PK_B}$

Decrypt $\{M\}_{PK_B}$

pre-shared - DS

a secret < 2. MAC.

3. Symmetric Encryp

Man-in-the-Middle Attack

Solution: Distributing Public Keys

- Idea: Sign Bob's public key to prevent tampering
- Problem
  - If Bob signs his public key, we need his public key to verify the signature
  - But Bob's public key is what we were trying to verify in the first place!
  - Circular problem: Alice can never trust any public key she receives
- You cannot gain trust if you trust nothing. You need a root of trust!
  - **Trust anchor**: Someone that we implicitly trust
  - From our trust anchor, we can begin to trust others

CA (PK, SK)

Man-in-middle attack

Trust-on-First-Use

- **Trust-on-first-use**: The first time you communicate, trust the public key that is used and warn the user if it changes in the future
  - Used in SSH and a couple other protocols
  - Idea: Attacks aren’t frequent, so assume that you aren’t being attacked the first time communicate

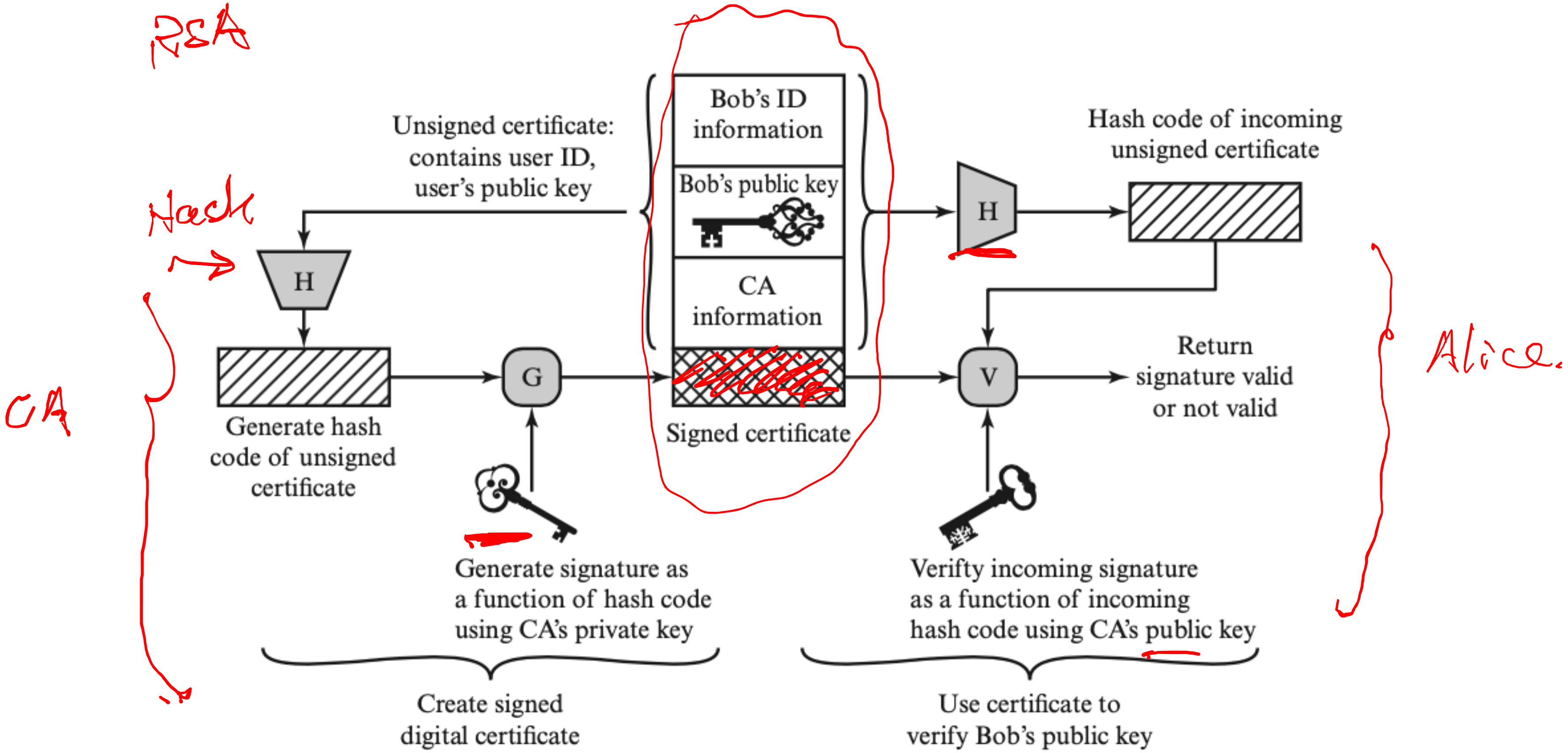
Certificates

Certificates

- **Certificate**: A signed endorsement of someone’s public key
  - A certificate contains at least two things: The **identity** of the person, and the **key** → PK
- Abbreviated notation
  - Signing with a private key *SK*: $\{\text{“Message”}\}_{SK^{-1}}$
    - Recall: A signed message must contain the message along with the signature; you can’t send the signature by itself!
- Scenario: Alice wants Bob’s public key. Alice trusts Charlie (PK_C, SK_C)

  CA
  - Charlie is our trust anchor
- If we trust PK_C, a certificate we would trust is $\{\text{“Bob’s public key is } PK_B\text{”}\}_{SK_C^{-1}}$

How do we use public-key certificate?



Unsigned certificate: contains user ID, user's public key

Bob's ID information

Bob's public key

CA information

Signed certificate

Hash code of incoming unsigned certificate

Generate hash code of unsigned certificate

Return signature valid or not valid

Generate signature as a function of hash code using CA's private key

Verifty incoming signature as a function of incoming hash code using CA's public key

Create signed digital certificate

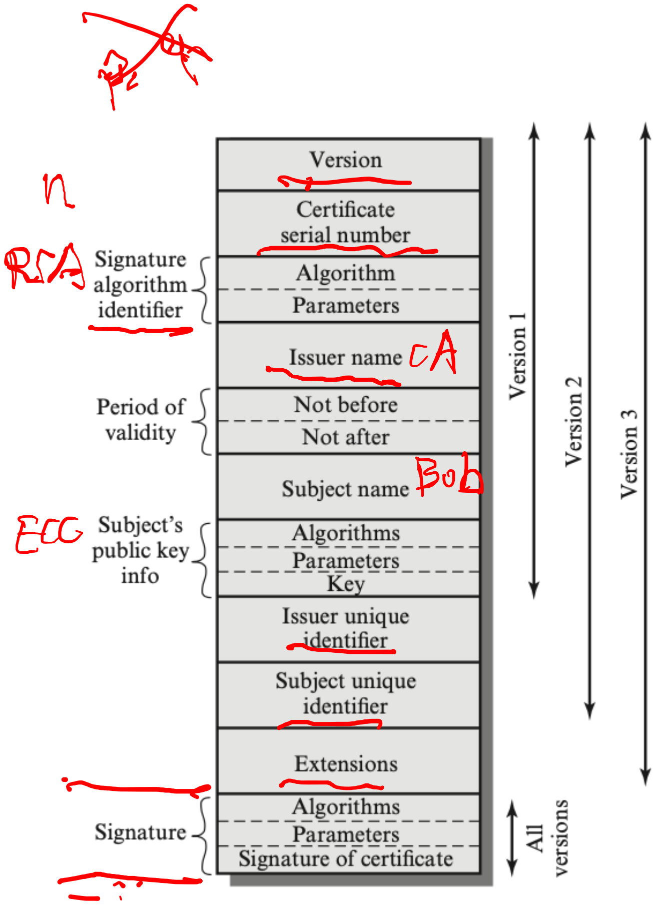
Use certificate to verify Bob's public key

X.509 Certificates

- Certificate serial # - SN
- Period validity - T^A
- Subject's public key info - Ap
- Signature signed by CA's private key
- Math notation:

$$CA \ll \underline{A} \gg \;=\; CA\,\{\underline{V, SN, AI, CA, UCA, A, UA, Ap, T^A}\}$$

subject name - Bob



n

RSA Signature algorithm identifier: Algorithm, Parameters

Version

Certificate serial number

Issuer name CA

Period of validity: Not before, Not after

Subject name Bob

ECC Subject's public key info: Algorithms, Parameters, Key

Issuer unique identifier

Subject unique identifier

Extensions

Signature: Algorithms, Parameters, Signature of certificate

readings

- Barnes, R.; Hoffman-Andrews, J.; McCarney, D.; Kasten, J. (March 2019). *Automatic Certificate Management Environment (ACME) RFC 8555*. IETF