Introduction to Information Security
14-741/18-631 Fall 2021
Unit 2: Lecture 3
UnKeyed, Other Algorithms, PKI

Limin Jia liminjia@andrew

This lecture's agenda

- **Outline**
  - One-way hash functions and hash chains
    - E.g., Lamport
    - Real-world example: MD5
  - Message Authentication Codes (MACs)
    - Real-world example: HMAC
  - Public Key Infrastructure (PKI)
- **Objective**
  - Expose you to the final pieces of the suite of core cryptographic tools we have at our disposal for secure communication protocol design
  - Allows us to piece together all elements needed

One way hash functions

- (We'll be more formal with the math soon)
- Properties of a cryptographically secure hash function
- One-wayness:

  Given y = H(x), cannot easily find x

  - Also known as pre-image resistance
- Weak collision resistance:

  Given x, cannot easily find x ≠ x' such that H(x) = H(x')

  - Also known as second pre-image resistance
- Strong collision resistance:

  Cannot easily find x, x' such that H(x) = H(x')
- Arbitrary length input
  - Can use for messages of any length to get fixed size hash

Examples of cryptographic hash functions

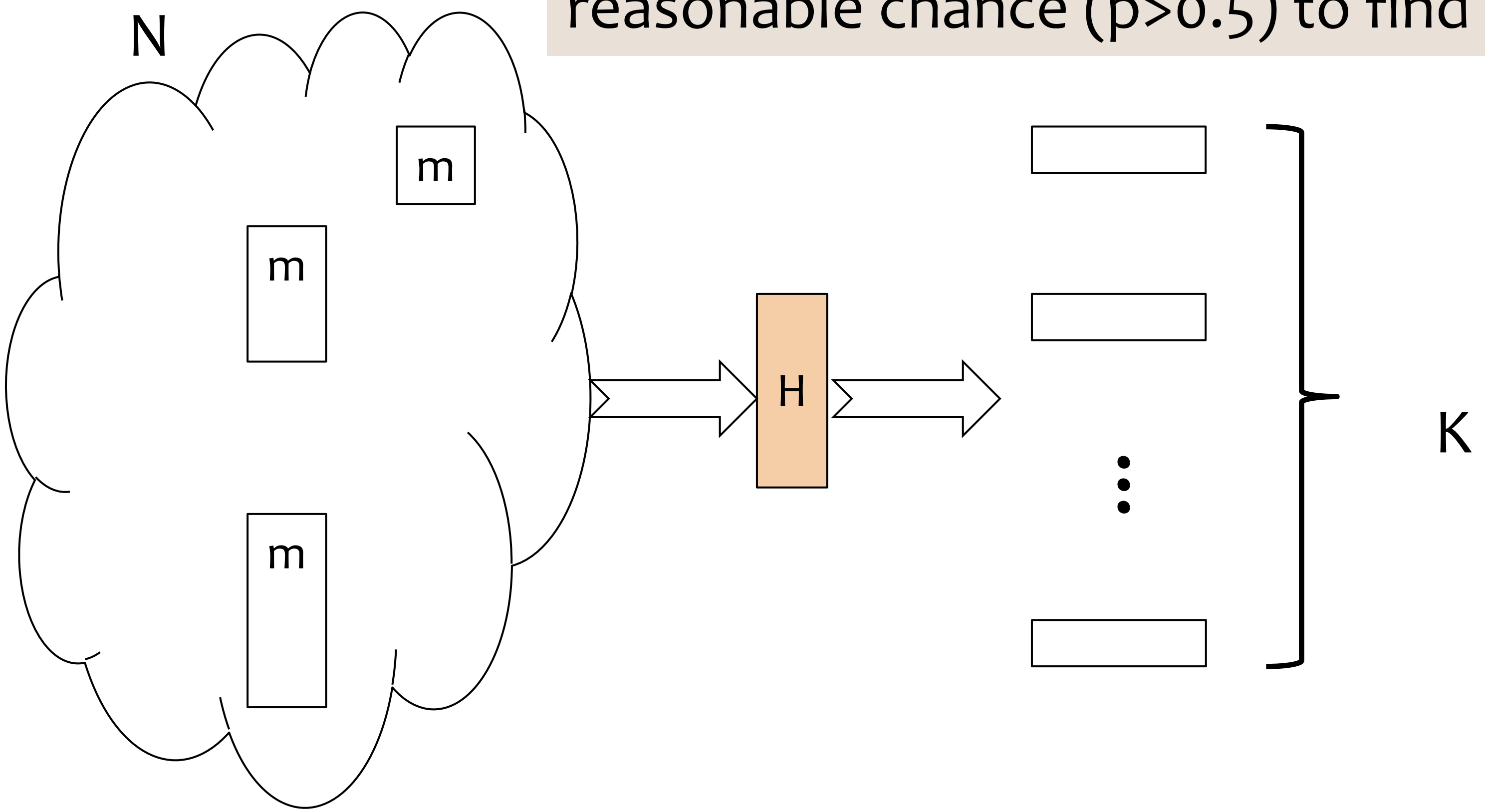
- **Obsolete:**
  - SHA-0, RIPEMD, MD4,…
    - Shown to be insecure (not collision resistant) – practical attack exists
- **Still very much deployed but insecure:**
  - MD5
    - Shown to be insecure (not collision resistant) – practical attack exists
  - SHA1
    - Shown to be insecure (not collision resistant) – attack still not very practical (but improving steadily)
- **Deployed, not broken (yet):**
  - RIPEMD-160
  - SHA2
    - Family that includes SHA-256
  - But both built on same type of construction as MD5, SHA1
    - Merkle-Damgård construction, which we'll see later
  - SHA3
  - Different construction from SHA-1, SHA-2
  - Keccak

Can we use one-way hash functions for…

- **Data integrity?**
  - Send (m, H(m))
- **Message authentication?**
  - Send (m, H(m))
- **Secrecy?**
  - Send (H(m))

How hard is it to find collision?

How large does N has to be to have a reasonable chance (p>0.5) to find a collision?

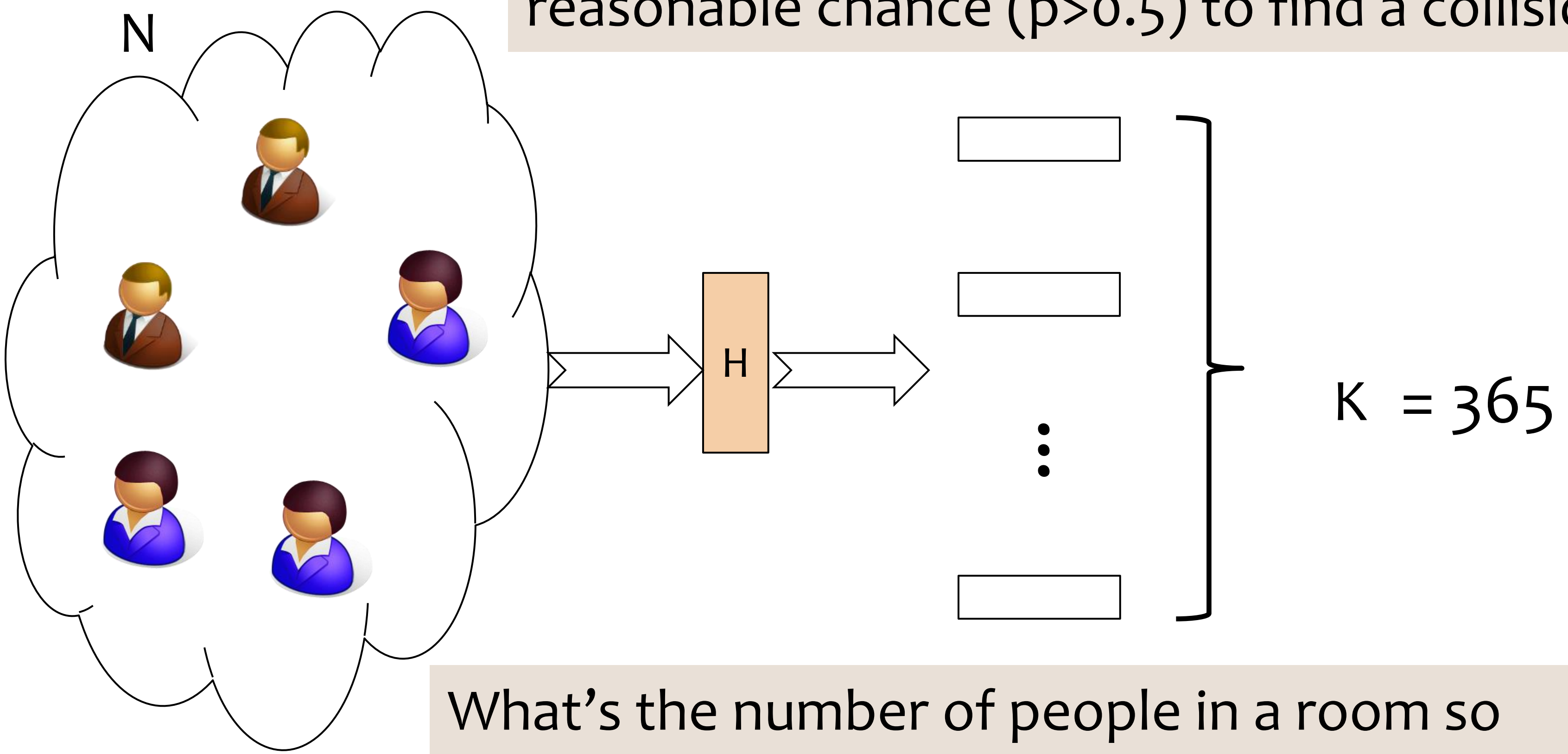


Birthday paradox

N=23 p=0.5

N=60 p = 0.99

How large does N has to be to have a reasonable chance (p>0.5) to find a collision?



What's the number of people in a room so there is a reasonable chance (p>0.5) that two people is that room share the same birthday?

Birthday paradox: proof

- **Probability that all birthdays are different**

$$\overline{p(N)} = 1 \times \frac{364}{365} \times \frac{363}{365} \times \cdots \frac{365-(N-1)}{365} = \frac{365!}{365^N (365-N)!}$$

- **Probability that at least two birthdays are identical**
- **Taylor approximation:**

$$e^{-x} \approx 1 - x \Rightarrow 1 - \frac{1}{365} \approx e^{-\frac{1}{365}}$$

- **So:**

$$p(N) = 1 - \frac{365!}{365^N (365-N)!}$$

$$p(N) \approx 1 - (e^{-1/365} e^{-2/365} \cdots e^{-(N-1)/365})$$

$$\approx 1 - e^{-N(N-1)/(2 \times 365)}$$

$$\approx 1 - e^{-N^2/(2 \times 365)}$$

How many people needed for a collision?

- **Greater than 50% iff**

$$1 - e^{-N^2/(2\times 365)} > p$$

$$\Leftrightarrow e^{-N^2/(2\times 365)} < 1 - p$$

$$\Leftrightarrow \frac{N^2}{2\times 365} > -\ln(1-p)$$

$$\Leftrightarrow N > \sqrt{2\times 365 \ln\left(\frac{1}{1-p}\right)}$$

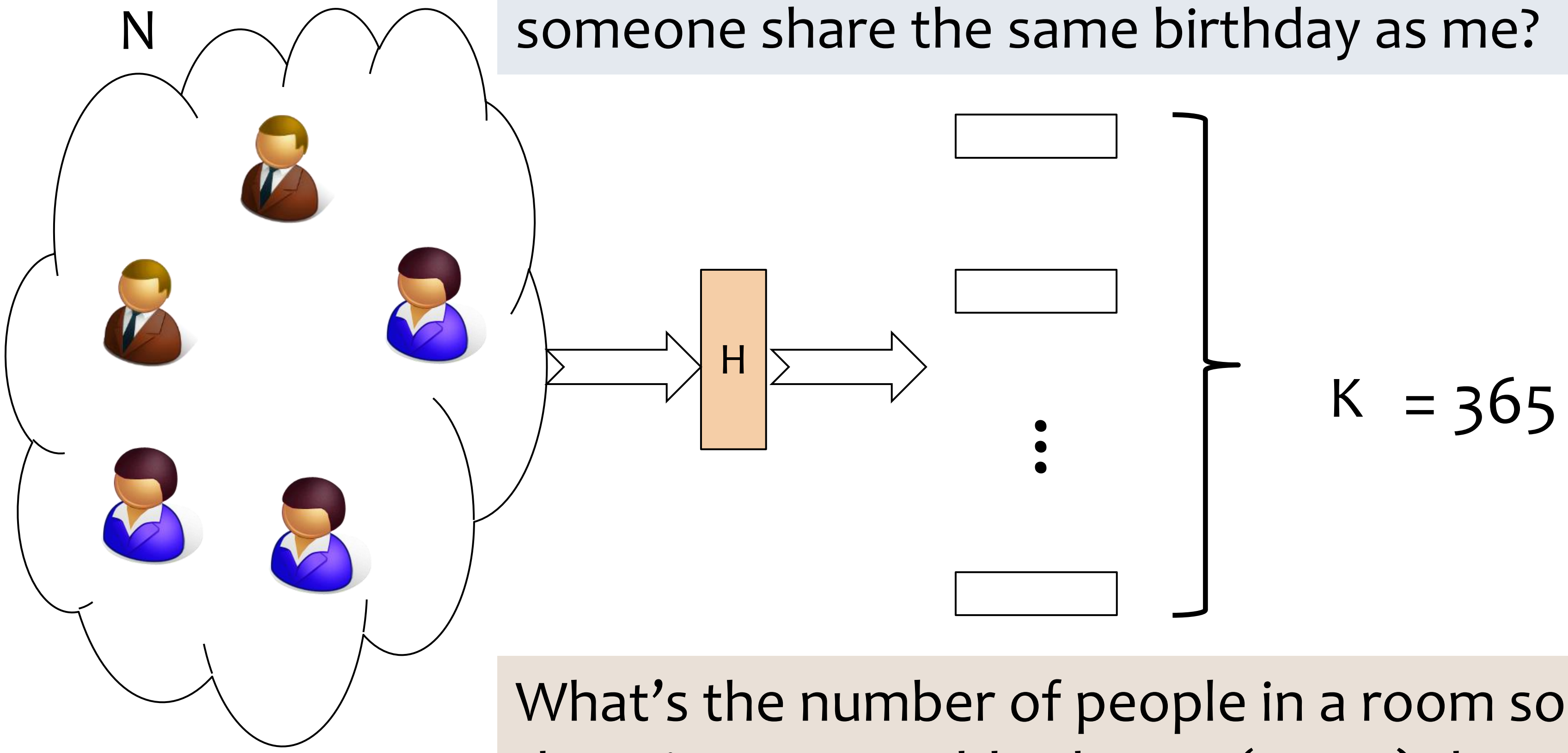
for $p = 0.5$

$$1 - e^{-N^2/(2\times 365)} > 0.5$$

$$N > 1.17\sqrt{365}$$

Why it is a paradox

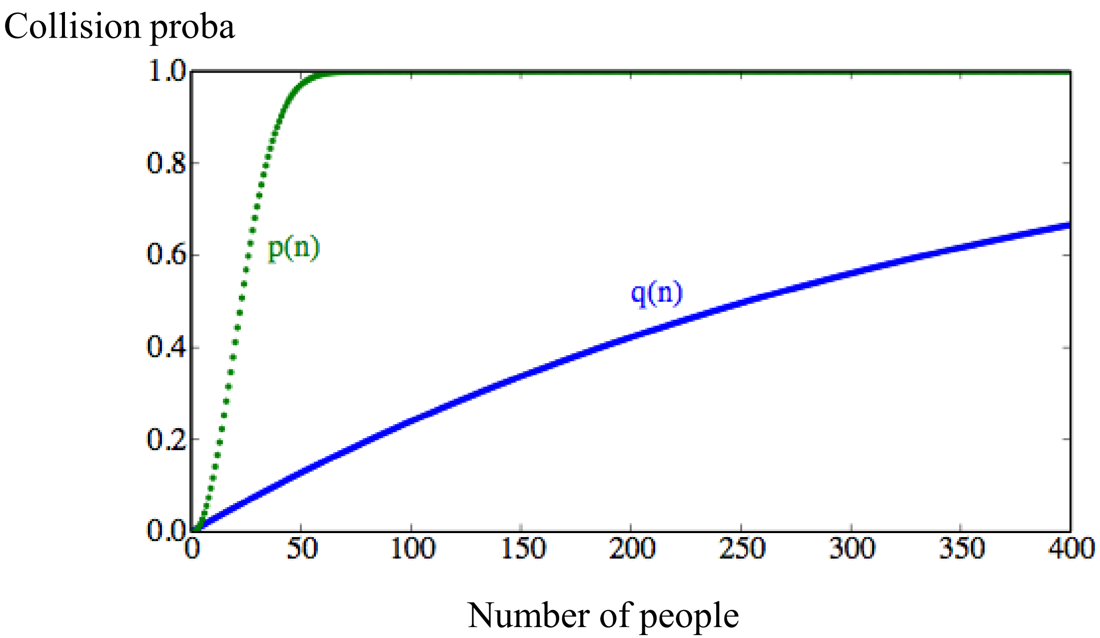
What's the number of people in a room so there is a reasonable chance (p>0.5) that someone share the same birthday as me?



What's the number of people in a room so there is a reasonable chance (p>0.5) that two people is that room share the same birthday?

Why it is a paradox

- **Two same birthdays:** $p(N) \approx 1 - e^{-N^2/(2\times 365)}$
- **Same birthday as me:** $q(n) = 1 - \left(\frac{365-1}{365}\right)^n$



Birthday paradox: consequences

- **For any hash function mapping a number *N* of inputs to a number *K* of outputs (digests), there is a pretty good probability of finding a collision if one can try a number of** $N > \sqrt{K}$ **inputs**

- **For the hash function to be secure**
  - it must be computationally infeasible to try a number of inputs to find collision
  - Make sure the length of the digest is large enough!

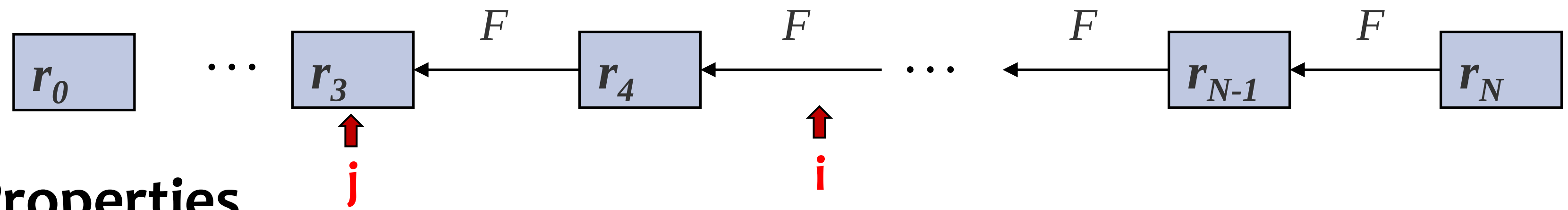
Birthday attack

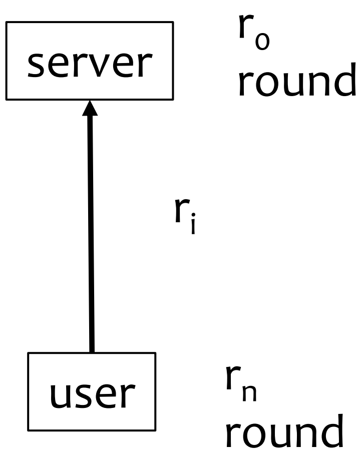
- **Attacker aims to compromise the data integrity of a letter by the victim**
- **Attack knows that the victim is using a hash function with m-bit long digest**
- **The attacker offers to draft a letter from the victim**
- **Create a large number of variations ($2^{m/2}$) on a fraudulent letter**
  - E.g. (Davies and Price, 1989),

    {This letter is | I am writing} to introduce {you to | to you} {Mr | --} Alfred { P. | -- } Barton, the {new | newly appointed}…
  - Already 32 possibilities here
- **Create a large number of cosmetic variations ($2^{m/2}$) on original letter**
  - Same technique as above
- **Try to find a collision between the two sets**
- **Present the victim with the original letter, replace it by fraudulent letter that has same hash**

One-way hash chains (Lamport)

- **Used for one time passwords**
- **Construction**
  - Pick random r_N and public one-way function F
  - $r_i=F(r_{i+1})$





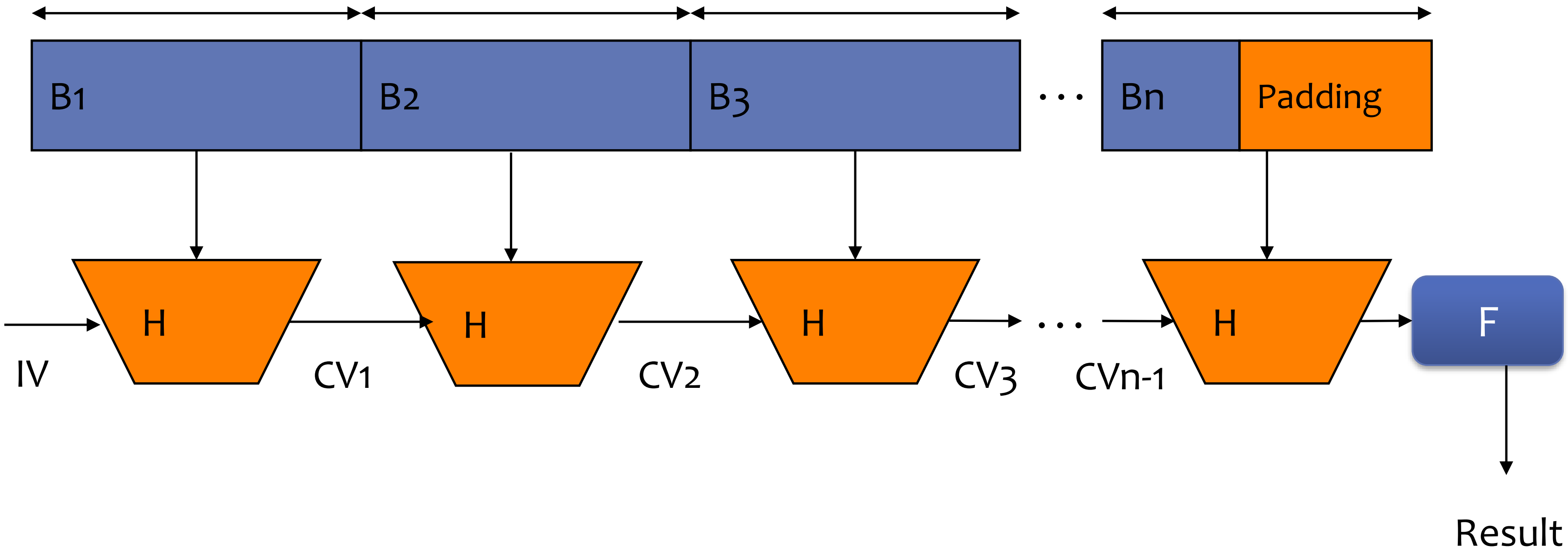
- **Properties**
  - Can't derive r_i from r_j for $j < i$
  - Efficiently authenticate r_i using r_j $(j<i)$: $r_j = F_{i-j}(r_i)$
- **Use in reverse order of construction $r_1, r_2, \ldots\ r_N$ as one-time password**
  - Secret value: r_N, public value r_0
  - Robust to missing values

MD5

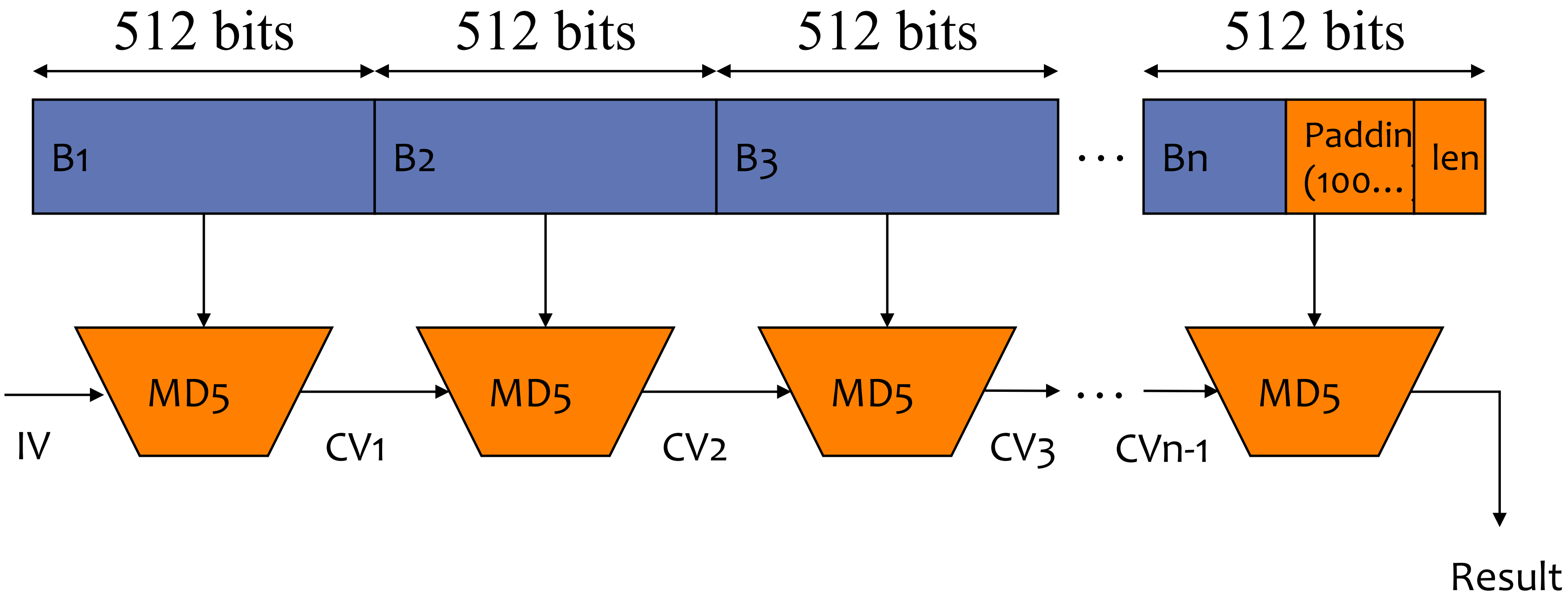
- **Designed by Ron Rivest (the R in RSA) in 1992**
- **Transforms arbitrary length input into 128-bit output**
- **MD5 is improved version of MD4**
- **Used as a basis for SHA-1 (160 bits)**
- **MD5 has been shown to be vulnerable to collisions**
  - (SHA-1 too has been shown to be broken…)

- **Based on the Merkle-Damgård construction**

Merkle-Damgård



MD5 overview



MD5 process

- **Each block Bi contains sixteen 32-bit words (512 bits)**

m_0	m_1	m_2	. . .	m_{15}

- **MD5 digest = four 32-bit words = 128 bit**

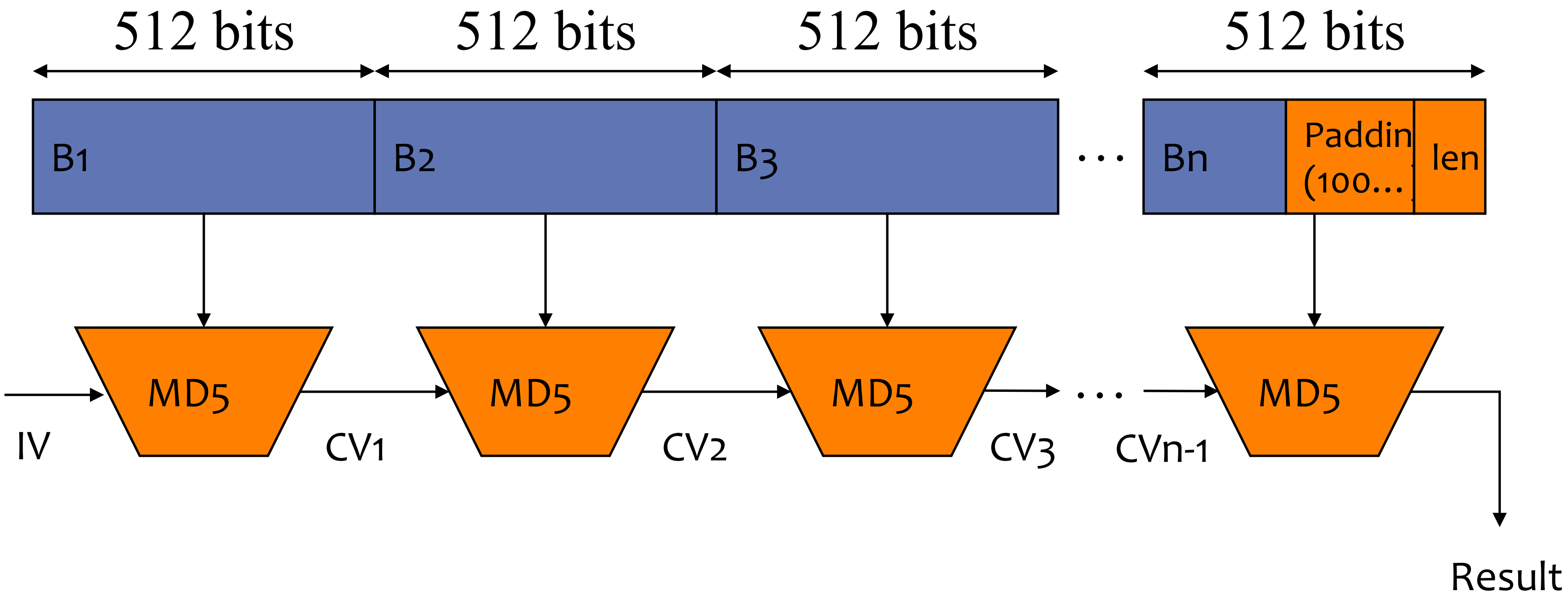
A	B	C	D

- **IV=initialization vector**

A_0	B_0	C_0	D_0

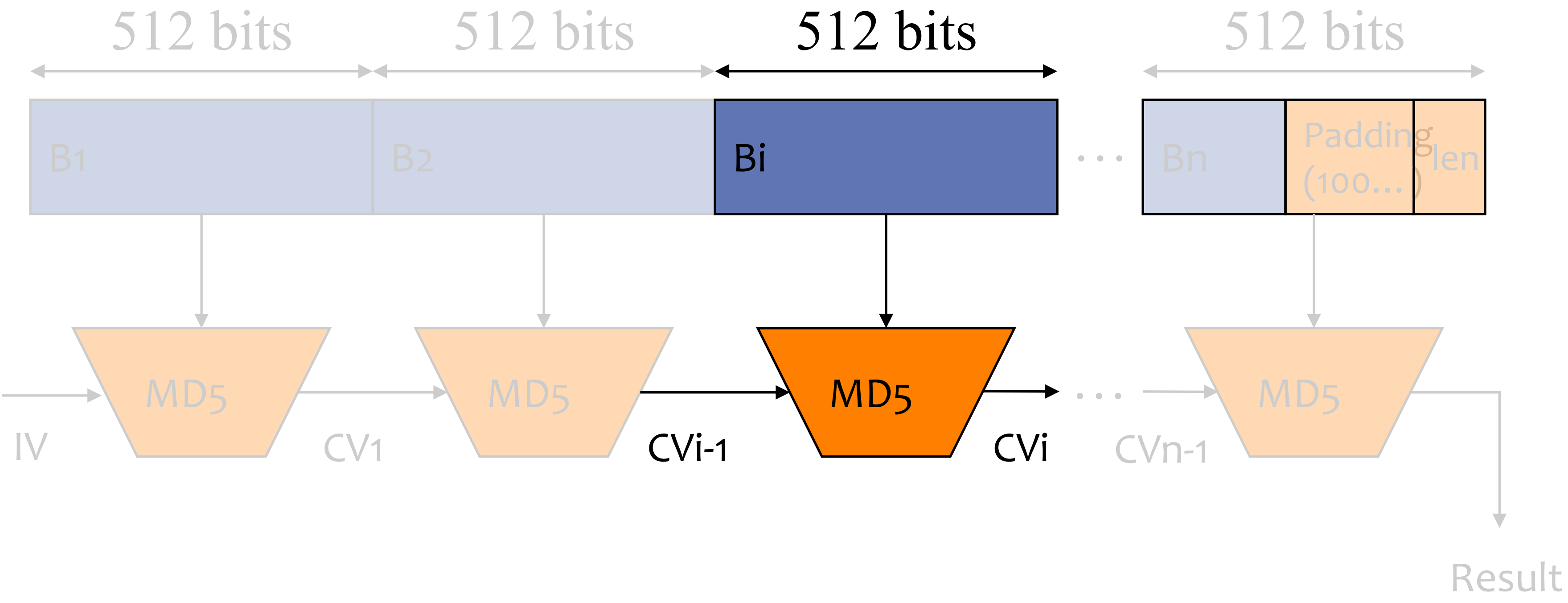
  - A0 = 01234567
  - B0 = 89ABCDEF
  - C0 = FEDCBA98
  - D0 = 76543210
- **Every stage consists of four rounds over the message block**

MD5 stage *i*

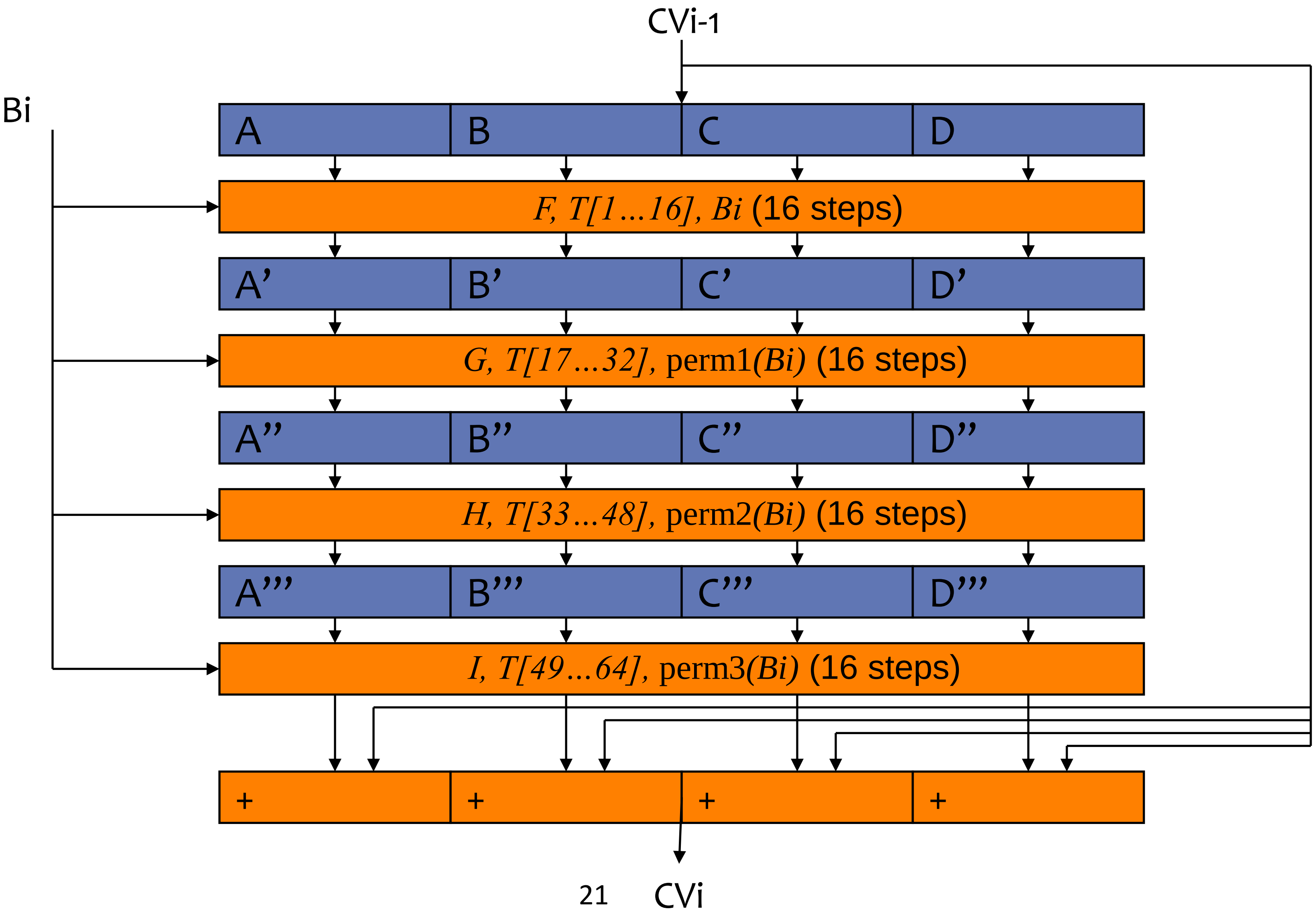


512 bits
512 bits
512 bits
512 bits
B1
B2
B3
. . .
Bn
Paddin
(100…
len
MD5
MD5
MD5
MD5
IV
CV1
CV2
CV3
. . .
CVn-1
Result

MD5 stage *i*


512 bits
512 bits
512 bits
512 bits
B1
B2
Bi
Bn
Padding (100…)
len
MD5
MD5
MD5
MD5
IV
CV1
CVi-1
CVi
CVn-1
Result

MD5 stage *i*


CVi-1
Bi
A
B
C
D
F, T[1…16], Bi (16 steps)
A’
B’
C’
D’
G, T[17…32], perm1(Bi) (16 steps)
A”
B”
C”
D”
H, T[33…48], perm2(Bi) (16 steps)
A’”
B’”
C’”
D’”
I, T[49…64], perm3(Bi) (16 steps)
+
+
+
+
CVi

MD5 main characteristics

- **Reduces arbitrary length text to a 128-bit output**
- **Uses same principle as cipher block chaining**
  - Succession of stages combining input and previous result
- **Extremely complicated in a given stage**
  - Tries to mix bits as much as possible
  - Tries to avoid collisions
  - Unfortunately, broken in 2006: two message texts with the same hash value can be found in about an hour on an IBM P960 (super multiprocessor, but still)
- **No one will ask you details about a given stage**
  - and if they do, you can tell them to go read IETF RFC 1321
- **… but you need to understand the basic principles**
  - Chaining, compression

MD5 collisions

- **Wang and Yu found a way to find M,M' and N,N' such that f(f(s,M),M') = f(f(s,N),N') for any MD5 state s**
- **Create pairs:**
  - M0, M1, …, Mi-1, **Mi**, **Mi+1**, Mi+2, …, Mn
  - M0, M1, …, Mi-1, **Ni**, **Ni+1**, Mi+2, …, Mn
- **Consider the programs:**
  - Program 1: if (data1 == data1) then { good_program } else { evil_program }
  - Program 2: if (data2 == data1) then { good_program } else { evil_program }
- **Data1 = Mi, Mi+1, Data2 = Ni, Ni+1**

Informal Definition of a MAC

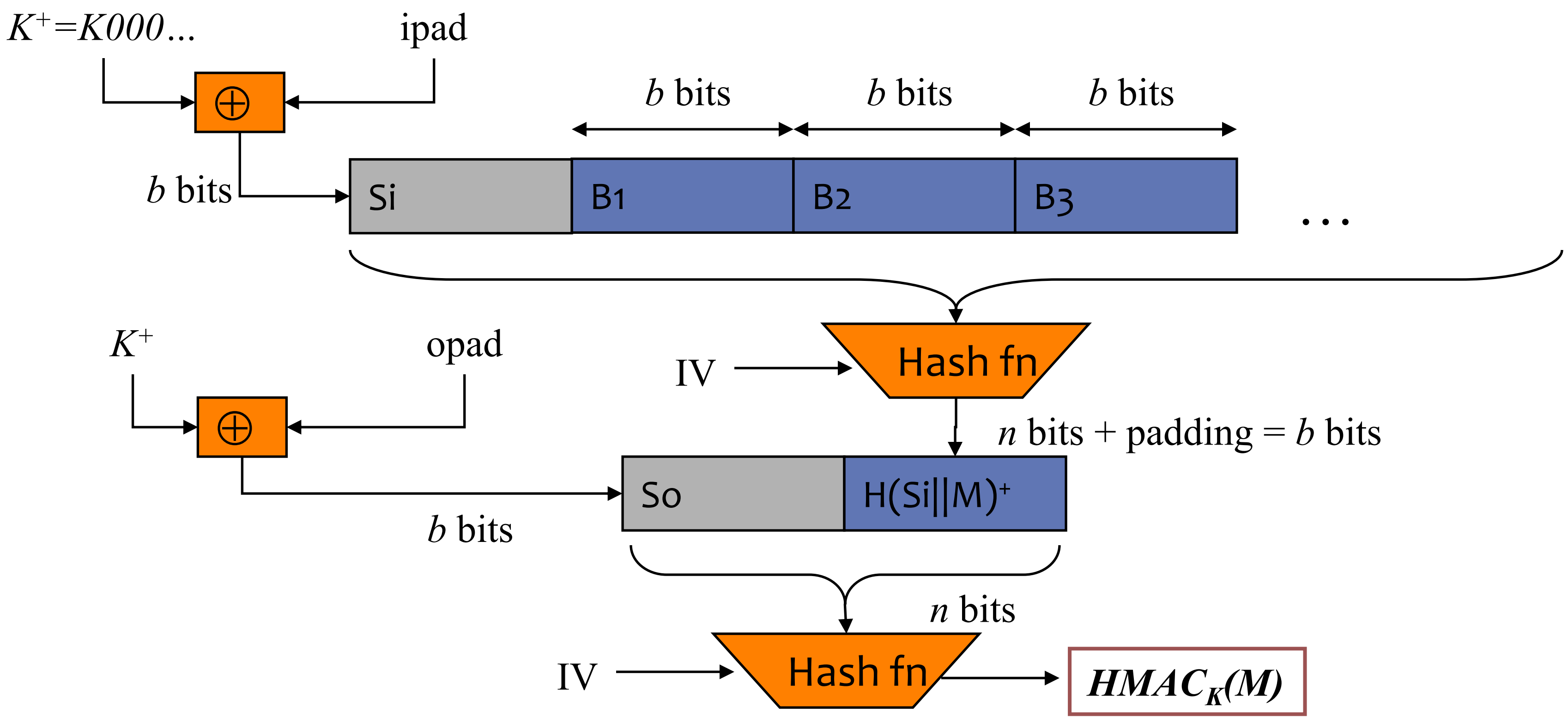
- **A message authentication code (MAC) scheme is a triple $\langle G, T, V\rangle$ of efficiently computable functions**
  - G outputs a "secret key" K
  $$K \leftarrow G(\cdot)$$
  - T takes a key K and "message" m as input, and outputs a "tag" t
  $$t \leftarrow T_K(m)$$
  - V takes a message m, tag t and key K as input, and outputs a bit b
  $$b \leftarrow V_K(m, t)$$
  - If $t \leftarrow T_K(m)$ then $V_K(m, t)$ outputs 1 ("valid")
  - Given only message/tag pairs $\{\langle m_i, T_K(m_i)\rangle\}_i$, it is computationally infeasible to compute $\langle m, t\rangle$ such that
  $$V_K(m, t) = 1$$
  for any new $m \neq m_i$

Message Authentication Codes

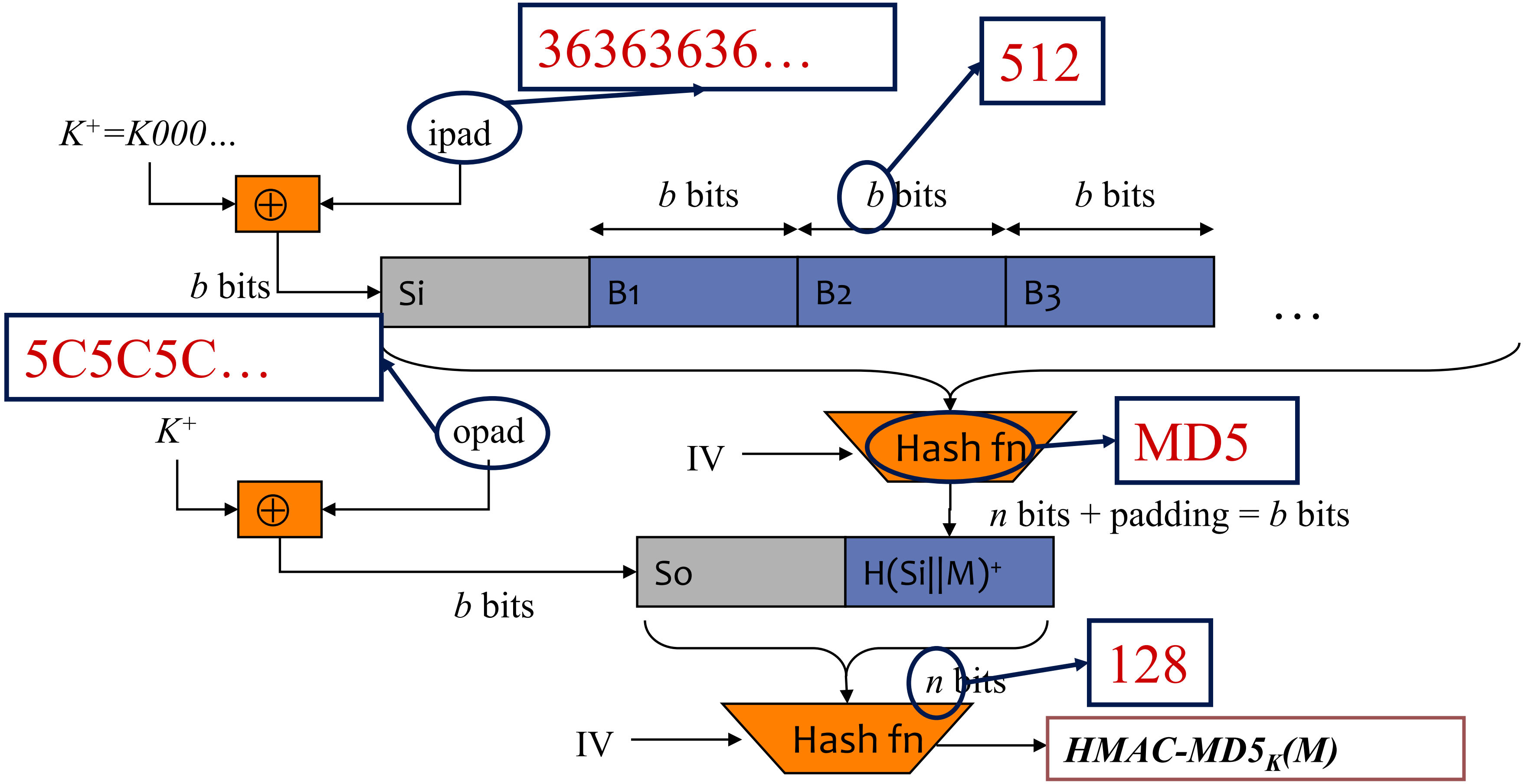
- **Sometimes abbreviated MAC**
  - Do not confuse with Media Access Carrier (e.g., Ethernet),
- **"Cryptographic checksum," i.e., keyed hash**
- **Can use symmetric block cipher or (more commonly) one-way hash function as a basis**
- **Provides authentication and integrity**
  - Send M, T(K, M)
- **Written MAC(K, M) or $MAC_K(M)$**

HMAC (Keyed hash functions)

- **MAC based on hash functions (MD5, SHA-1, RIPEMD-160, …)**
  - HMAC(K,m) = H((K⁺ ⊕ opad) || H((K⁺ ⊕ ipad) || m))



Example: HMAC-MD5



How does it all fit together?

- **We have seen many components a cryptosystem can rely on**
  - Asymmetric crypto
  - Symmetric crypto
  - One-way hash functions
  - Public key encryption scheme, digital signature scheme, MAC scheme
- **Building a cryptosystem requires understanding the possible interactions between the different components**

Example: PGP

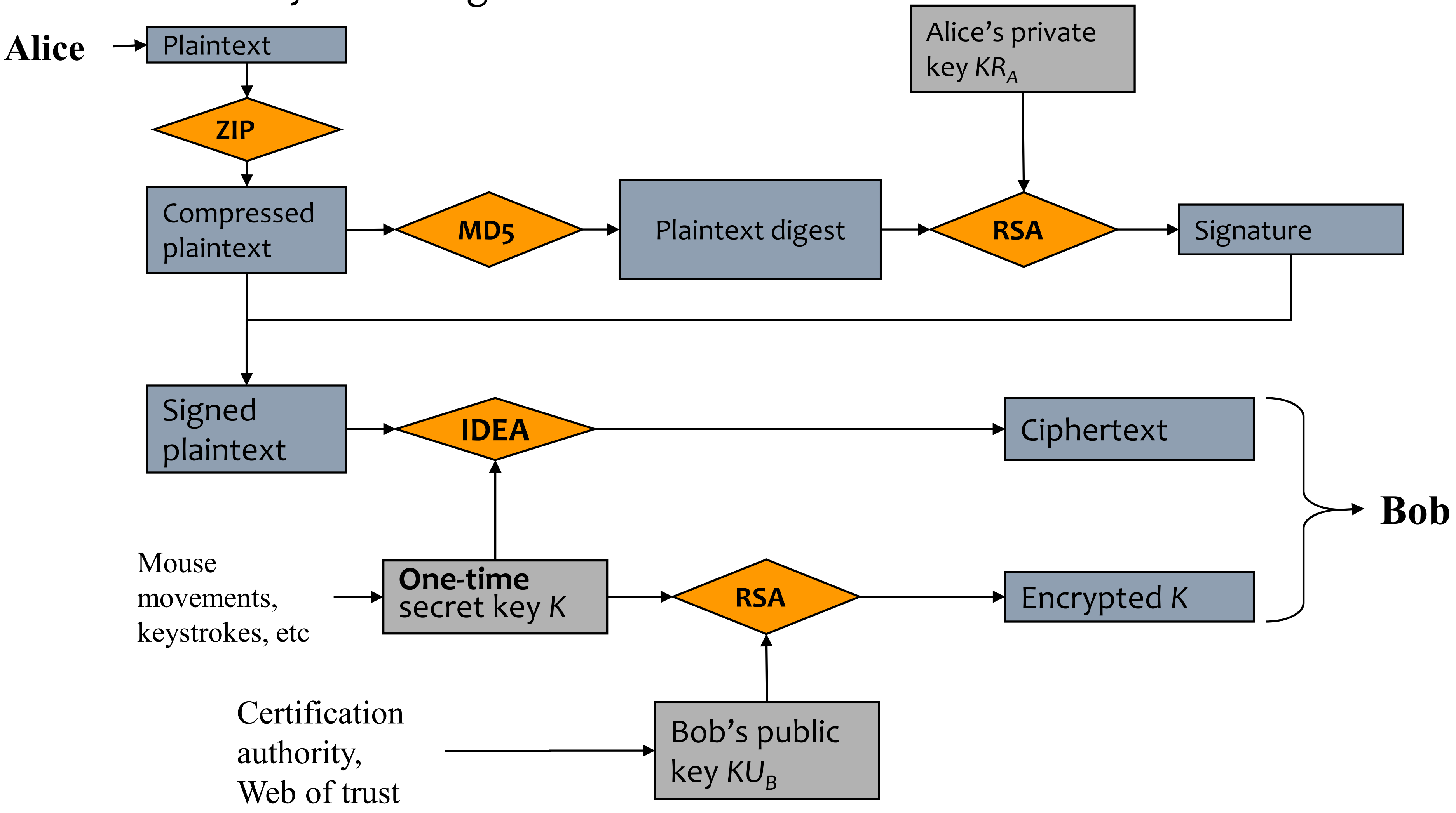
- **Pretty good privacy**
- **Used to digitally sign and/or encrypt email**
- **Author, Phil Zimmermann (MIT) was accused in the early 90s of violating US export control regulations**
  - Program was "leaked" on the Internet
  - Was target of three year criminal investigation led by the US customs
  - Never indicted

Phil Zimmermann
(source: Wikipedia)

PGP 2 overview

Scenario: Alice sends Bob an email

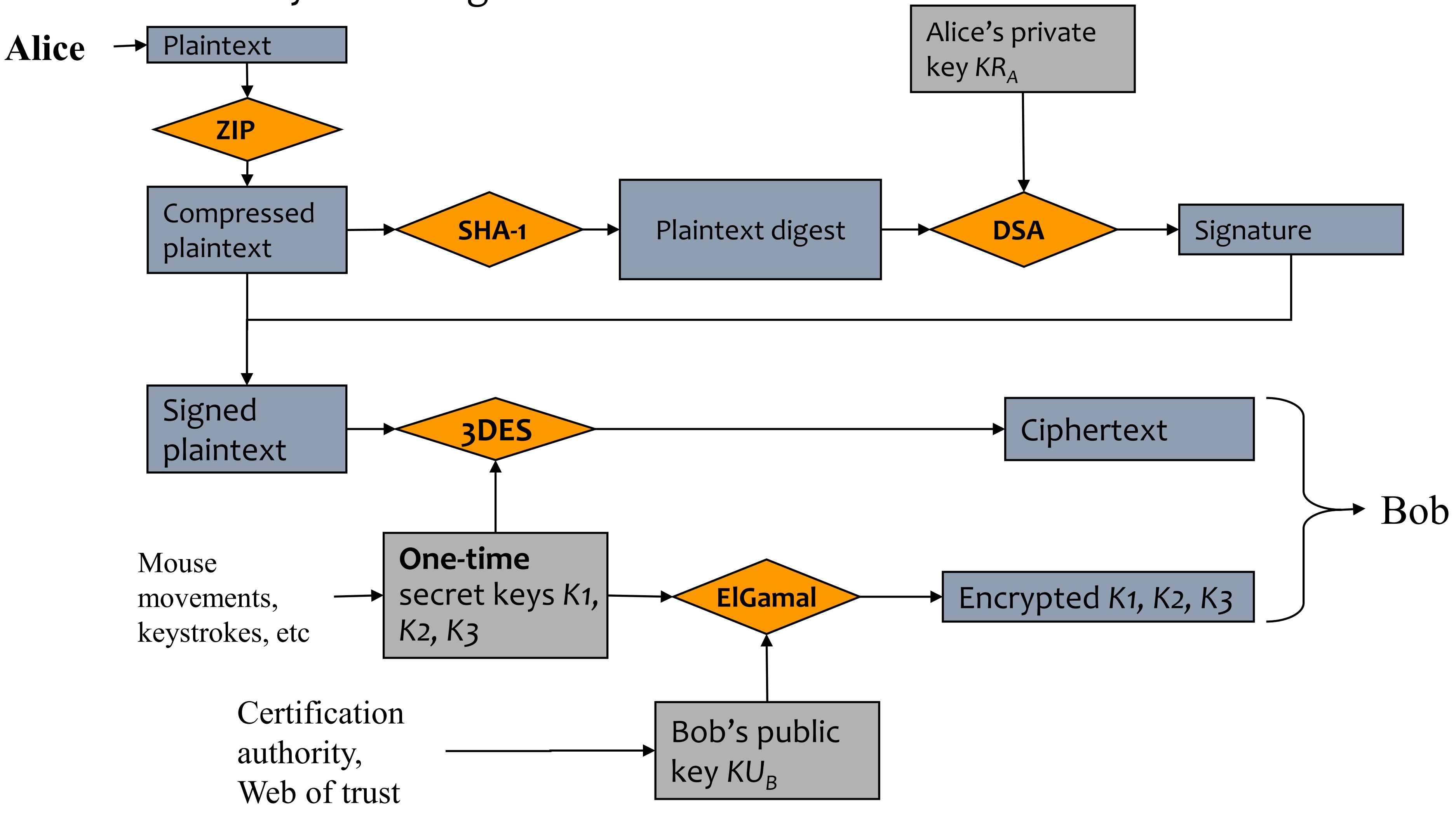
Goal: secrecy + message authentication



PGP 5 overview

Scenario: Alice sends Bob an email

Goal: secrecy + message authentication



How does Alice know K is Bob's public key?

- Need an infrastructure to help verify the authenticity of public keys (PKI)

Key components in PKI

- **CA: Certificate Authority**
  - similar to TTP (Trusted Third Party)
  - Vouch for the authenticity of public keys
  - Issuers of public-key certificates
- **A public-key certificate (or simply "certificate")**
  - binds a name to a public key
  - ensure the authenticity of a public key
  - Include: Issuer, signature, key usage, public key, valid date...

Example certificate

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            10:e6:fc:62:b7:41:8a:d5:00:5e:45:b6
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=BE, O=GlobalSign nv-sa, CN=GlobalSign Organization Validation CA - SHA256 - G2
        Validity
            Not Before: Nov 21 08:00:00 2016 GMT
            Not After : Nov 22 07:59:59 2017 GMT
        Subject: C=US, ST=California, L=San Francisco, O=Wikimedia Foundation, Inc., CN=*.wikipedia.org
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:c9:22:69:31:8a:d6:6c:ea:da:c3:7f:2c:ac:a5:
                    af:c0:02:ea:81:cb:65:b9:fd:0c:6d:46:5b:c9:1e:
                    ed:b2:ac:2a:1b:4a:ec:80:7b:e7:1a:51:e0:df:f7:
                    c7:4a:20:7b:91:4b:20:07:21:ce:cf:68:65:8c:c6:
                    9d:3b:ef:d5:c1
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Agreement
            Authority Information Access:
                CA Issuers - URI:http://secure.globalsign.com/cacert/gsorganizationvalsha2g2r1.crt
                OCSP - URI:http://ocsp2.globalsign.com/gsorganizationvalsha2g2

            X509v3 Certificate Policies:
                Policy: 1.3.6.1.4.1.4146.1.20
                  CPS: https://www.globalsign.com/repository/
                Policy: 2.23.140.1.2.2

            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 CRL Distribution Points:

                Full Name:
                  URI:http://crl.globalsign.com/gs/gsorganizationvalsha2g2.crl

            X509v3 Subject Alternative Name:
                DNS:*.wikipedia.org, DNS:*.m.mediawiki.org, DNS:*.m.wikibooks.org, DNS:*.m.wikidata.org, DNS:*.m.wikimedia.org,
      DNS:*.m.wikimediafoundation.org, DNS:*.m.wikinews.org, DNS:*.m.wikipedia.org, DNS:*.m.wikiquote.org, DNS:*.m.wikisource.org,
      DNS:*.m.wikiversity.org, DNS:*.m.wikivoyage.org, DNS:*.m.wiktionary.org, DNS:*.mediawiki.org, DNS:*.planet.wikimedia.org,
      DNS:*.wikibooks.org, DNS:*.wikidata.org, DNS:*.wikimedia.org, DNS:*.wikimediafoundation.org, DNS:*.wikinews.org, DNS:*.wikiquote.org,
      DNS:*.wikisource.org, DNS:*.wikiversity.org, DNS:*.wikivoyage.org, DNS:*.wiktionary.org, DNS:*.wmfusercontent.org,
      DNS:*.zero.wikipedia.org, DNS:mediawiki.org, DNS:w.wiki, DNS:wikibooks.org, DNS:wikidata.org, DNS:wikimedia.org,
      DNS:wikimediafoundation.org, DNS:wikinews.org, DNS:wikiquote.org, DNS:wikisource.org, DNS:wikiversity.org, DNS:wikivoyage.org,
      DNS:wiktionary.org, DNS:wmfusercontent.org, DNS:wikipedia.org
            X509v3 Extended Key Usage:
                TLS Web Server Authentication, TLS Web Client Authentication
            X509v3 Subject Key Identifier:
                28:2A:26:2A:57:8B:3B:CE:B4:D6:AB:54:EF:D7:38:21:2C:49:5C:36
            X509v3 Authority Key Identifier:
                keyid:96:DE:61:F1:BD:1C:16:29:53:1C:C0:CC:7D:3B:83:00:40:E6:1A:7C

    Signature Algorithm: sha256WithRSAEncryption
         8b:c3:ed:d1:9d:39:6f:af:40:72:bd:1e:18:5e:30:54:23:35:
```

Key components in PKI

- **CA: Certificate Authority**
  - similar to TTP (Trusted Third Party)
  - Vouch for the authenticity of public keys
  - Issuers of public-key certificates
- **A public-key certificate (or simply "certificate")**
  - binds a name to a public key
  - ensure the authenticity of a public key
  - Include: Issuer, signature, key usage, public key, valid date…
- **Trust anchor: certificates of (public keys of) entities whose trust is assumed**
- **Certificate repository: stores certificates**

Certificate chain

Trust anchor

$\{K_{CA}\}_{K_{CA}^{-1}}$

“signs”

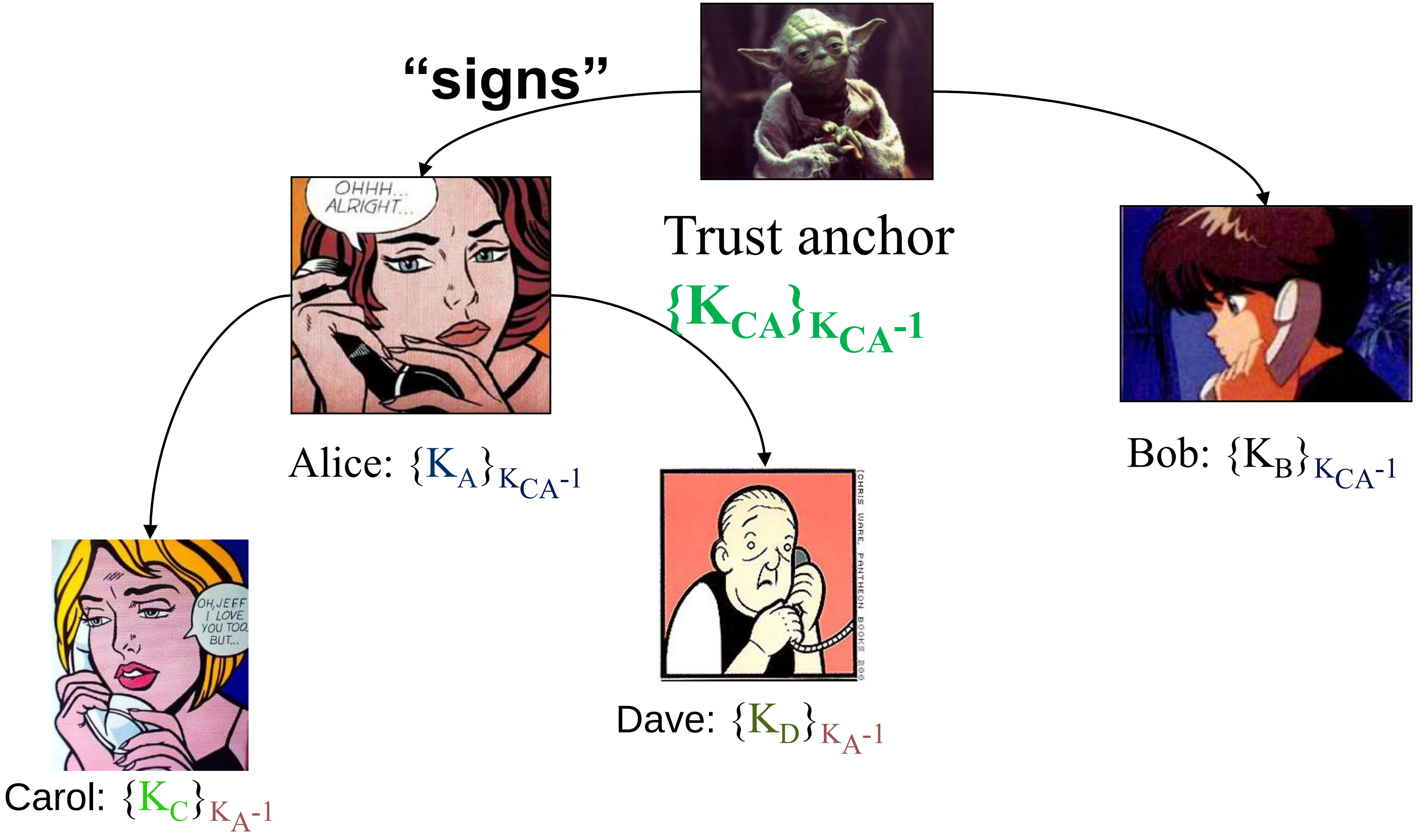
Trust anchor

$\{K_{CA}\}_{K_{CA}^{-1}}$

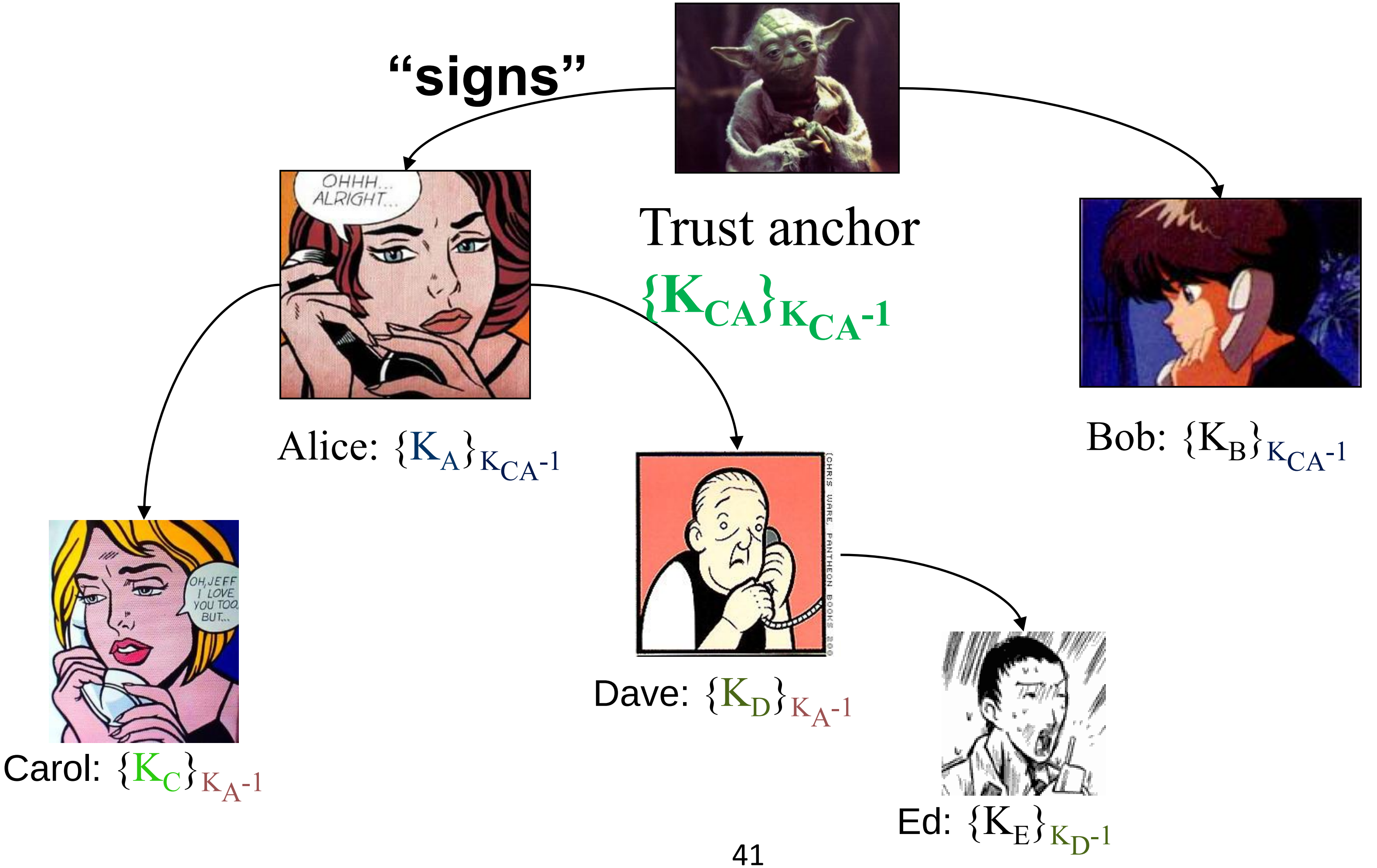
Alice: $\{K_A\}_{K_{CA}^{-1}}$

Bob: $\{K_B\}_{K_{CA}^{-1}}$

Certificate chain

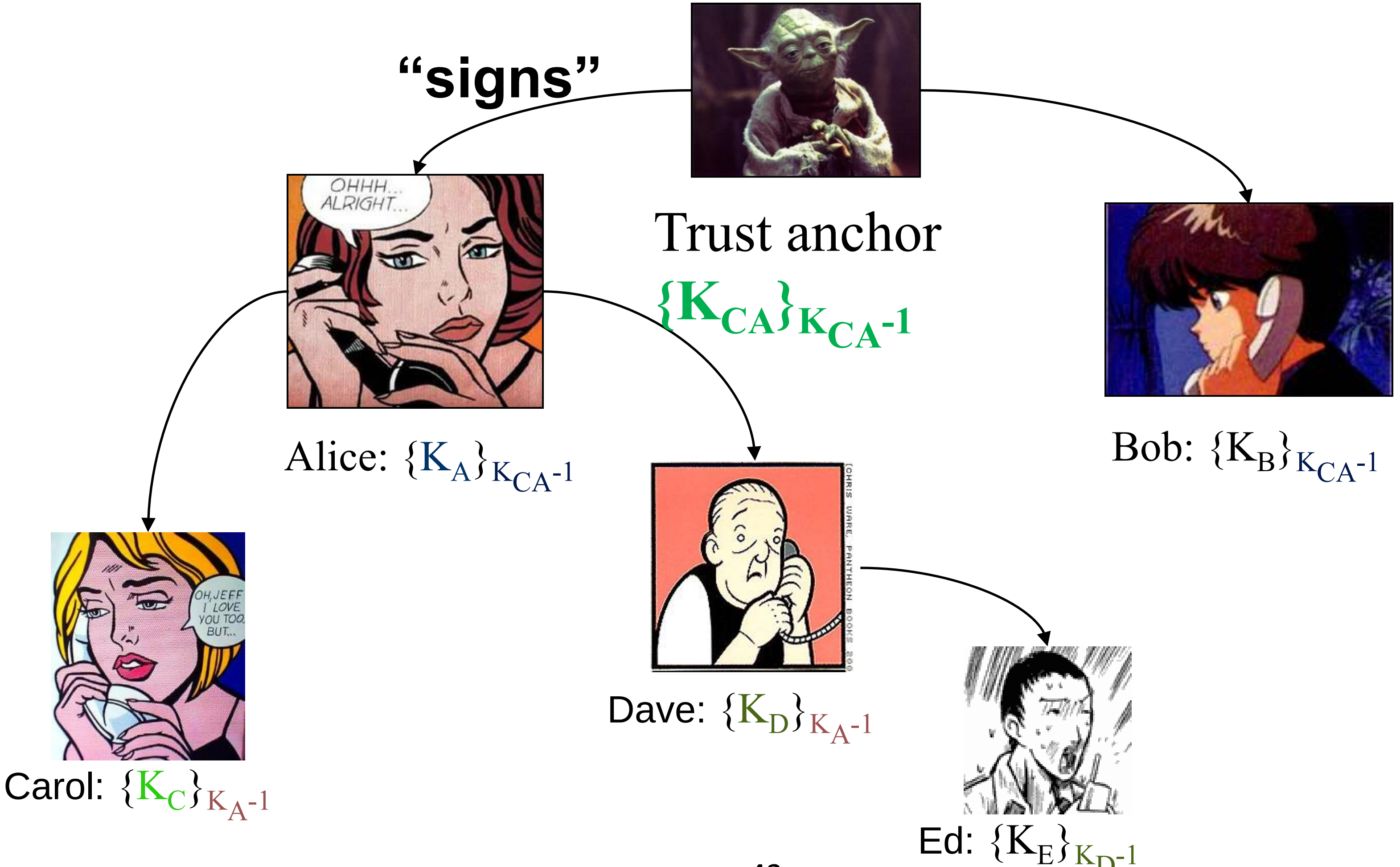


Certificate chain



Certificate chain verification

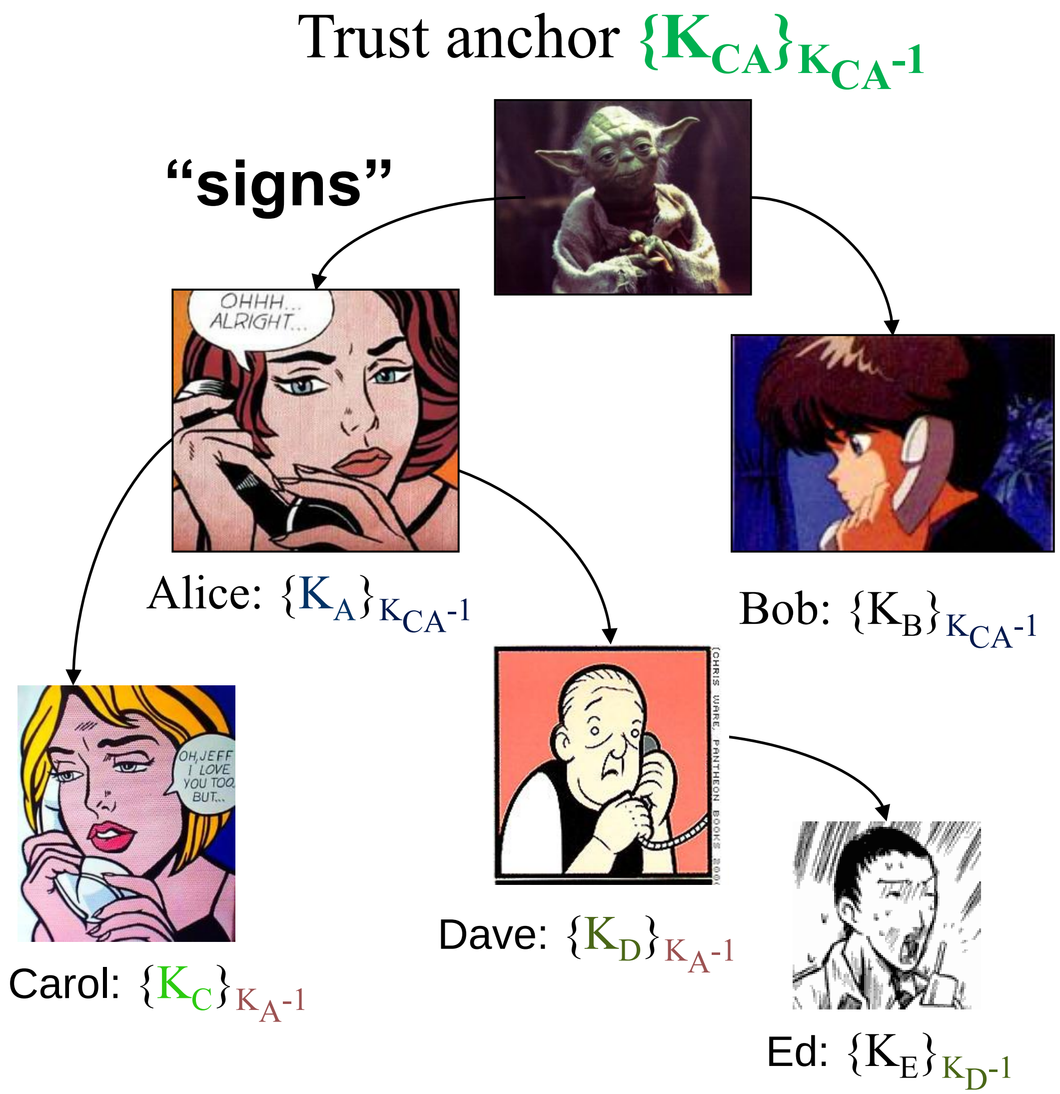
- **How can we verify that K_E is Ed's public key ?**



Certificate chain verification

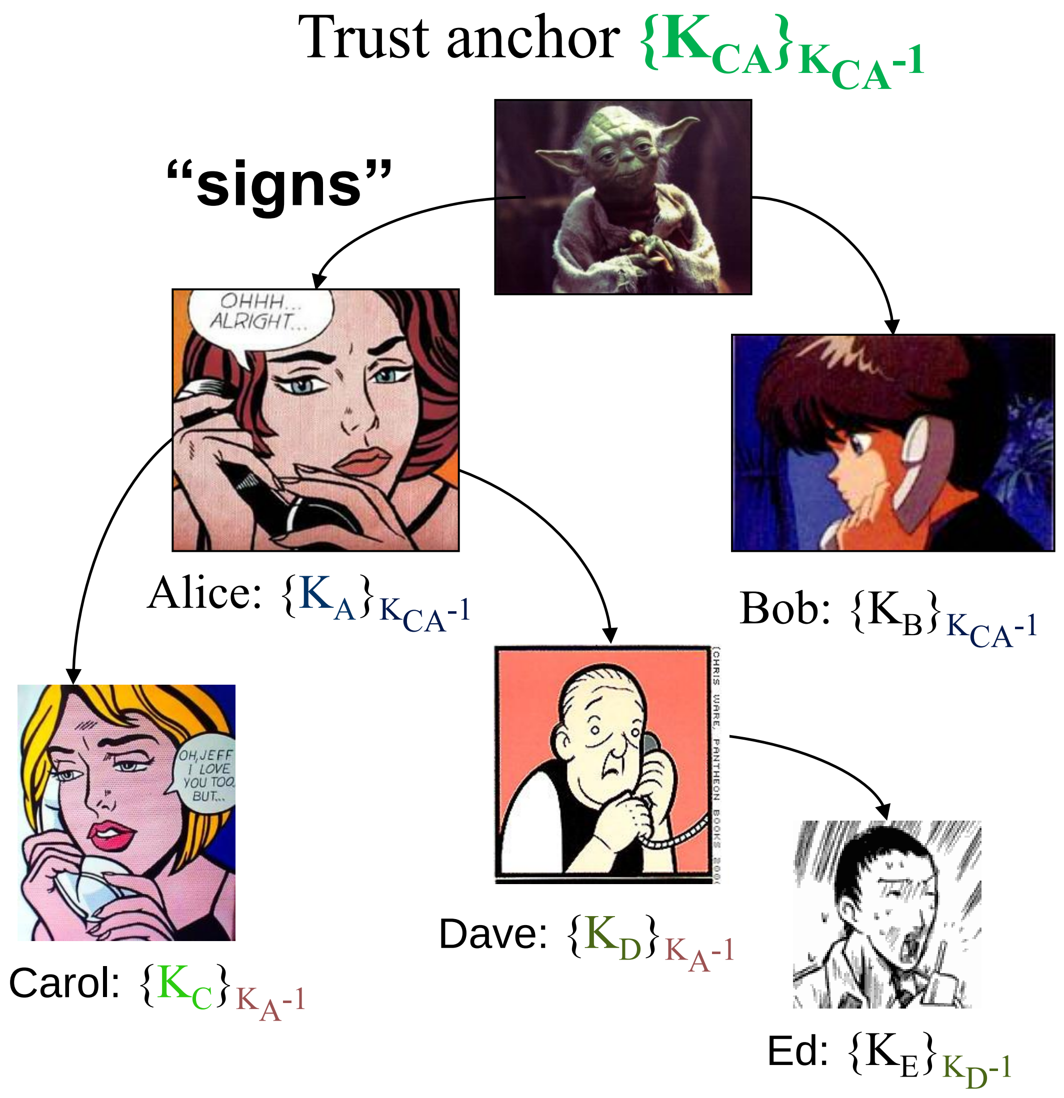
- **To verify a key K is Ed's public key, we need to:**
  1. Obtain certificate CERT for K
  2. Verify CERT using K_D
  3. Verify K_D
     1. Obtain certificate for K_D
     2. Verify certificate ….

Follow the chain and verify very certificate



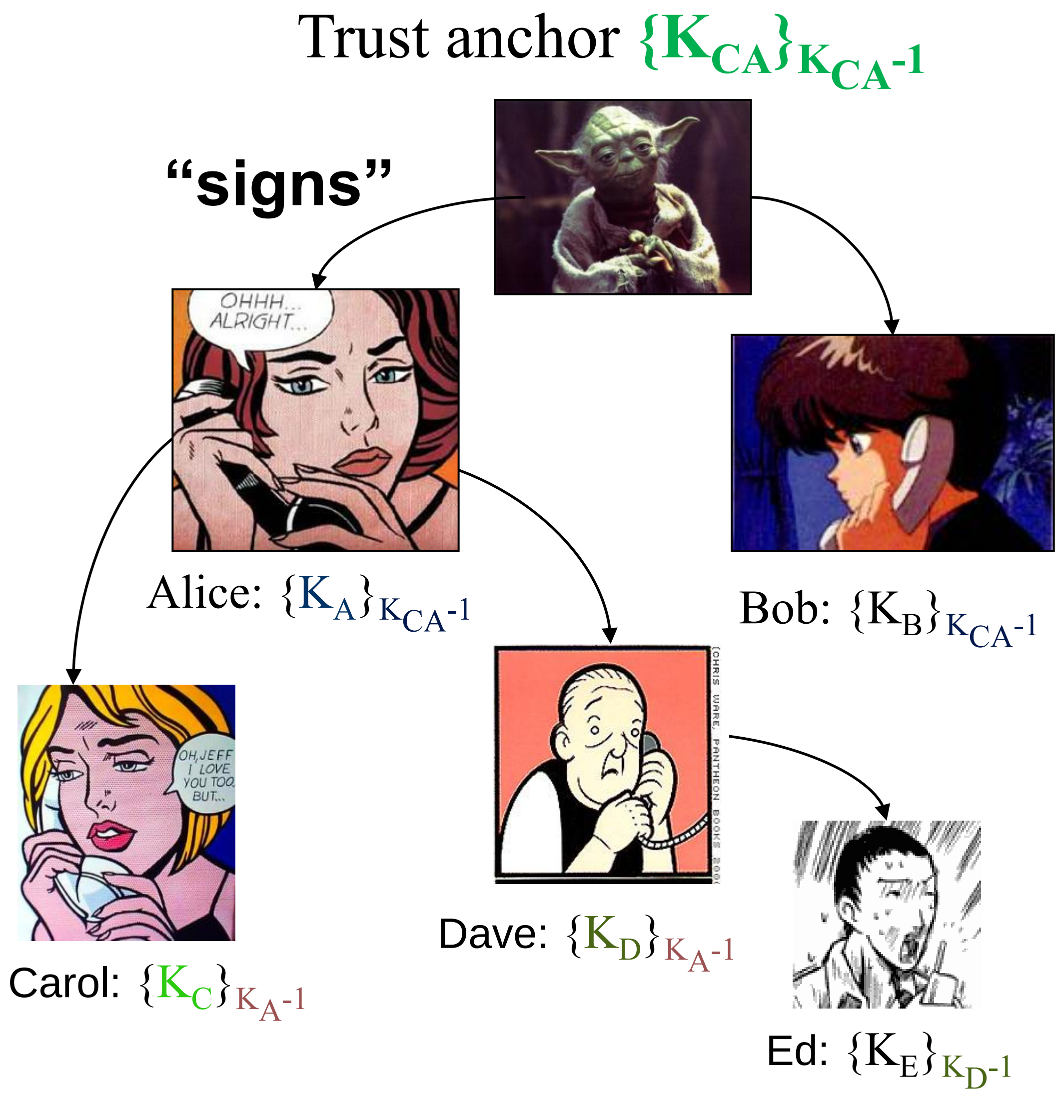
Why should we trust the verification result?

- **Assuming all users have trust in anchor K_{CA}**
- **We trust every CA in the chain**
  - trust Alice to certify Dave, and trust Alice to certify that Dave can certify others…



Application of PKI

- **How could Carol and Ed set up a secure shared session key for communication?**
  - Recall the man in the middle attack of Diffie-Hellman key exchange



PKI trust models

- **Define:**
  - Who are the trust anchors,
  - How to choose trust anchors
  - How to validate a public key

PKI trust models

- **Delegated CA**
  - CA can issue certificates to other CAs (secondary)
  - Vouch for their key and their trustworthiness as a CA
  - Delegated CA can sign certificates itself
- **Monopoly**
  - Universally trusted CA
- **Monopoly + trusted Registration Authorities (RA)**
  - RAs check identities and vouch for keys, but the CA does all actual signing
- **Oligarchy**
  - Multiple trusted anchors, users can configure

Today's PKI "hierarchy"

Verisign: K_V

CNN: $\{K_{CNN}\}_{K_V\text{-}1}$

eBay: $\{K_E\}_{K_V\text{-}1}$

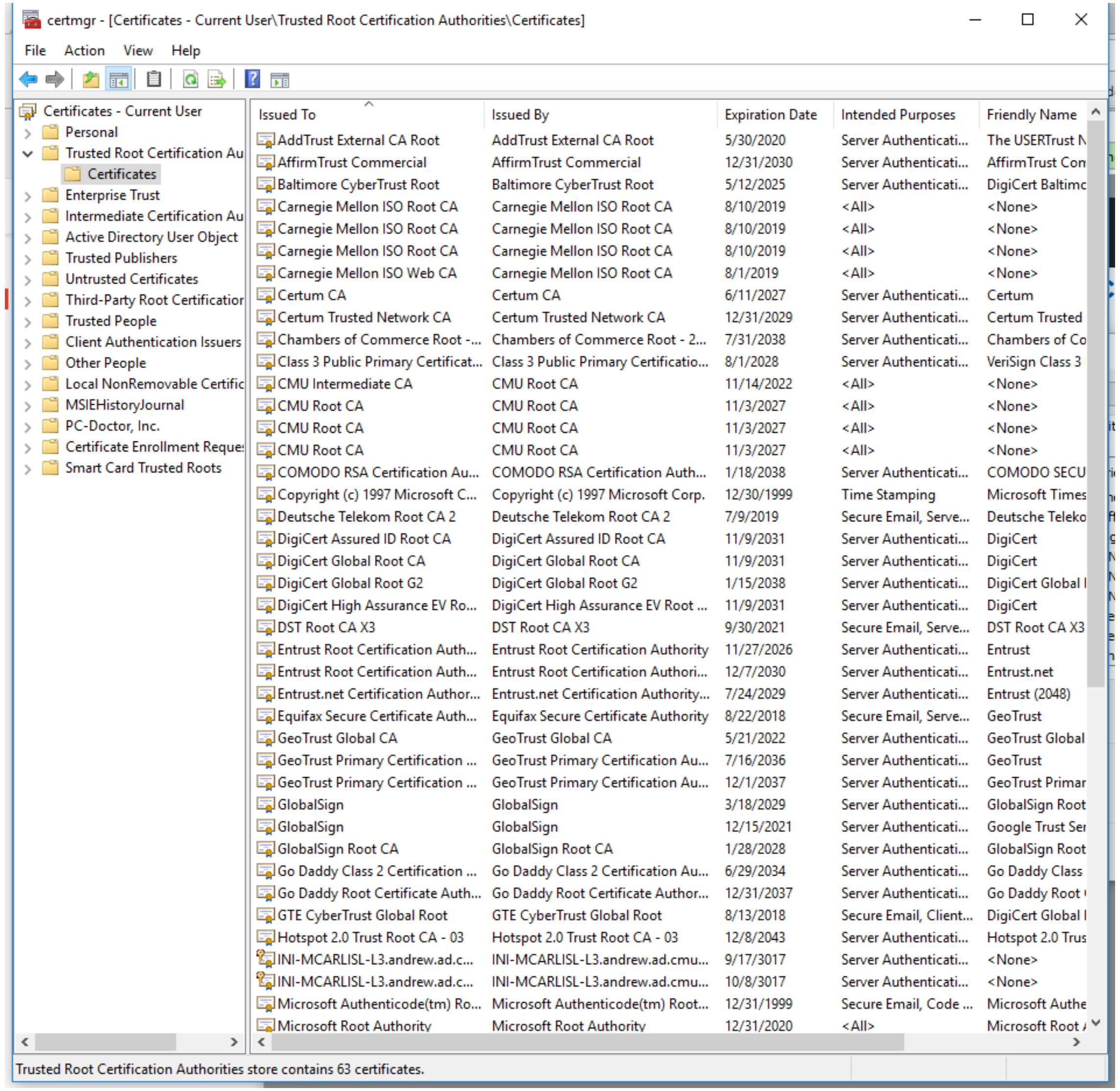
Yahoo!: $\{K_Y\}_{K_V\text{-}1}$

Comodo: K_U

Carol: $\{K_C\}_{K_U\text{-}1}$

Dave: $\{K_D\}_{K_U\text{-}1}$

Problems with today’s CA model



Issued To	Issued By	Expiration Date	Intended Purposes	Friendly Name
AddTrust External CA Root	AddTrust External CA Root	5/30/2020	Server Authenticati...	The USERTrust N
AffirmTrust Commercial	AffirmTrust Commercial	12/31/2030	Server Authenticati...	AffirmTrust Con
Baltimore CyberTrust Root	Baltimore CyberTrust Root	5/12/2025	Server Authenticati...	DigiCert Baltimc
Carnegie Mellon ISO Root CA	Carnegie Mellon ISO Root CA	8/10/2019	<All>	<None>
Carnegie Mellon ISO Root CA	Carnegie Mellon ISO Root CA	8/10/2019	<All>	<None>
Carnegie Mellon ISO Root CA	Carnegie Mellon ISO Root CA	8/10/2019	<All>	<None>
Carnegie Mellon ISO Web CA	Carnegie Mellon ISO Root CA	8/1/2019	<All>	<None>
Certum CA	Certum CA	6/11/2027	Server Authenticati...	Certum
Certum Trusted Network CA	Certum Trusted Network CA	12/31/2029	Server Authenticati...	Certum Trusted
Chambers of Commerce Root -...	Chambers of Commerce Root - 2...	7/31/2038	Server Authenticati...	Chambers of Co
Class 3 Public Primary Certificat...	Class 3 Public Primary Certificatio...	8/1/2028	Server Authenticati...	VeriSign Class 3
CMU Intermediate CA	CMU Root CA	11/14/2022	<All>	<None>
CMU Root CA	CMU Root CA	11/3/2027	<All>	<None>
CMU Root CA	CMU Root CA	11/3/2027	<All>	<None>
CMU Root CA	CMU Root CA	11/3/2027	<All>	<None>
COMODO RSA Certification Au...	COMODO RSA Certification Auth...	1/18/2038	Server Authenticati...	COMODO SECU
Copyright (c) 1997 Microsoft C...	Copyright (c) 1997 Microsoft Corp.	12/30/1999	Time Stamping	Microsoft Times
Deutsche Telekom Root CA 2	Deutsche Telekom Root CA 2	7/9/2019	Secure Email, Serve...	Deutsche Teleko
DigiCert Assured ID Root CA	DigiCert Assured ID Root CA	11/9/2031	Server Authenticati...	DigiCert
DigiCert Global Root CA	DigiCert Global Root CA	11/9/2031	Server Authenticati...	DigiCert
DigiCert Global Root G2	DigiCert Global Root G2	1/15/2038	Server Authenticati...	DigiCert Global
DigiCert High Assurance EV Ro...	DigiCert High Assurance EV Root ...	11/9/2031	Server Authenticati...	DigiCert
DST Root CA X3	DST Root CA X3	9/30/2021	Secure Email, Serve...	DST Root CA X3
Entrust Root Certification Auth...	Entrust Root Certification Authority	11/27/2026	Server Authenticati...	Entrust
Entrust Root Certification Auth...	Entrust Root Certification Authori...	12/7/2030	Server Authenticati...	Entrust.net
Entrust.net Certification Author...	Entrust.net Certification Authority...	7/24/2029	Server Authenticati...	Entrust (2048)
Equifax Secure Certificate Auth...	Equifax Secure Certificate Authority	8/22/2018	Secure Email, Serve...	GeoTrust
GeoTrust Global CA	GeoTrust Global CA	5/21/2022	Server Authenticati...	GeoTrust Global
GeoTrust Primary Certification ...	GeoTrust Primary Certification Au...	7/16/2036	Server Authenticati...	GeoTrust
GeoTrust Primary Certification ...	GeoTrust Primary Certification Au...	12/1/2037	Server Authenticati...	GeoTrust Primar
GlobalSign	GlobalSign	3/18/2029	Server Authenticati...	GlobalSign Root
GlobalSign	GlobalSign	12/15/2021	Server Authenticati...	Google Trust Ser
GlobalSign Root CA	GlobalSign Root CA	1/28/2028	Server Authenticati...	GlobalSign Root
Go Daddy Class 2 Certification ...	Go Daddy Class 2 Certification Au...	6/29/2034	Server Authenticati...	Go Daddy Class
Go Daddy Root Certificate Auth...	Go Daddy Root Certificate Author...	12/31/2037	Server Authenticati...	Go Daddy Root
GTE CyberTrust Global Root	GTE CyberTrust Global Root	8/13/2018	Secure Email, Client...	DigiCert Global
Hotspot 2.0 Trust Root CA - 03	Hotspot 2.0 Trust Root CA - 03	12/8/2043	Server Authenticati...	Hotspot 2.0 Trus
INI-MCARLISL-L3.andrew.ad.c...	INI-MCARLISL-L3.andrew.ad.cmu...	9/17/3017	Server Authenticati...	<None>
INI-MCARLISL-L3.andrew.ad.c...	INI-MCARLISL-L3.andrew.ad.cmu...	10/8/3017	Server Authenticati...	<None>
Microsoft Authenticode(tm) Ro...	Microsoft Authenticode(tm) Root...	12/31/1999	Secure Email, Code ...	Microsoft Authe
Microsoft Root Authority	Microsoft Root Authority	12/31/2020	<All>	Microsoft Root

Problems with today's CA model

- **Too many "trusted" CA's**
- **A single one of them is compromised implies a total loss of certification property**
  - E.g., Diginotar got breached
  - Person who broke into it could issue certificates that would pass as valid in any browser
    - E.g., Gmail certificates
  - Man-in-the-middle attack possible
- **Practical solution: certificate pinning**

Certificate Pinning

- **Hard Certificate Pinning**
  - Exact server certificate hard-coded into application
- **CA pinning**
  - Limited set of authorities, or a particular public key of server

Conventional PKI wisdom

- **Every entity needs a certificate**
- **Obtain cert from CA with strong protection of private key**

Certificate revocation

- **Certificate revocation is a mechanism to invalidate certificates**
  - After a private key is disclosed
  - Employee leaves corporation
- **CA periodically publishes Certificate Revocation List (CRL)**
  - Delta CRLs only contain changes
- **What is the general problem with revocation?**

Take away slide

- **One-way hash functions**
  - Unkeyed primitive
  - Useful for digital signatures, one-time passwords, …
  - Can be used to ensure integrity of the original message if collision resistant
- **Big problem: collision resistance**
  - Birthday attack (only ~$2K^{1/2}$ tries required to find collision w.h.p.)
- **MD5**
  - Chained design, resembles Cipher Block Chaining
  - Very complicated stage design to make it collision-resistant
  - … and yet, it was recently broken
- **MACs**
  - Essentially: one way hash function with a key
  - E.g., HMAC-MD5
  - Can be used to ensure integrity and authentication
- **PGP:** Example of cryptosystem that combines asymmetric, symmetric, and unkeyed primitives
- **PKI: Attempts to solve the trust problem in public keys**
  - … but mostly displaces the problem: do you trust Bob or Verisign?