Lecture 11 – Digital signatures, UF-CMA, RSA, Schnorr, PKI

TEK4500

03.11.2021

Håkon Jacobsen

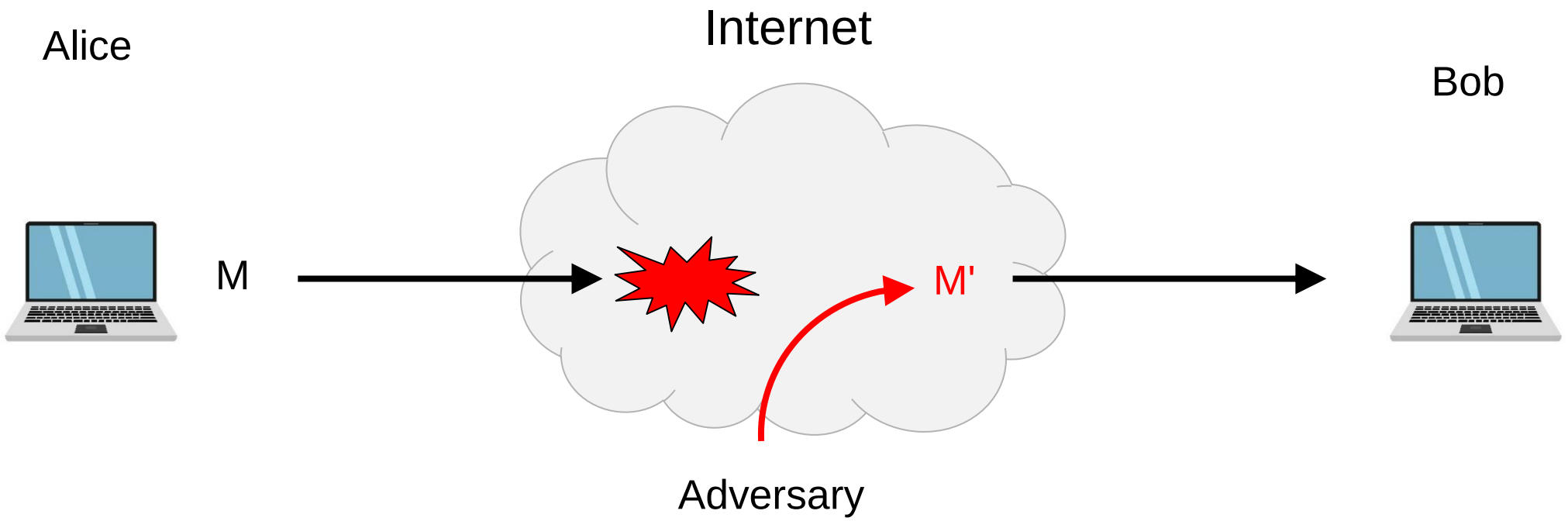
hakon.jacobsen@its.uio.no

Basic goals of cryptography

	Message privacy	**Message integrity / authentication**
Symmetric keys	Symmetric encryption	Message authentication codes (MAC)
Asymmetric keys	Asymmetric encryption (a.k.a. public-key encryption)	Digital signatures

(Key exchange)

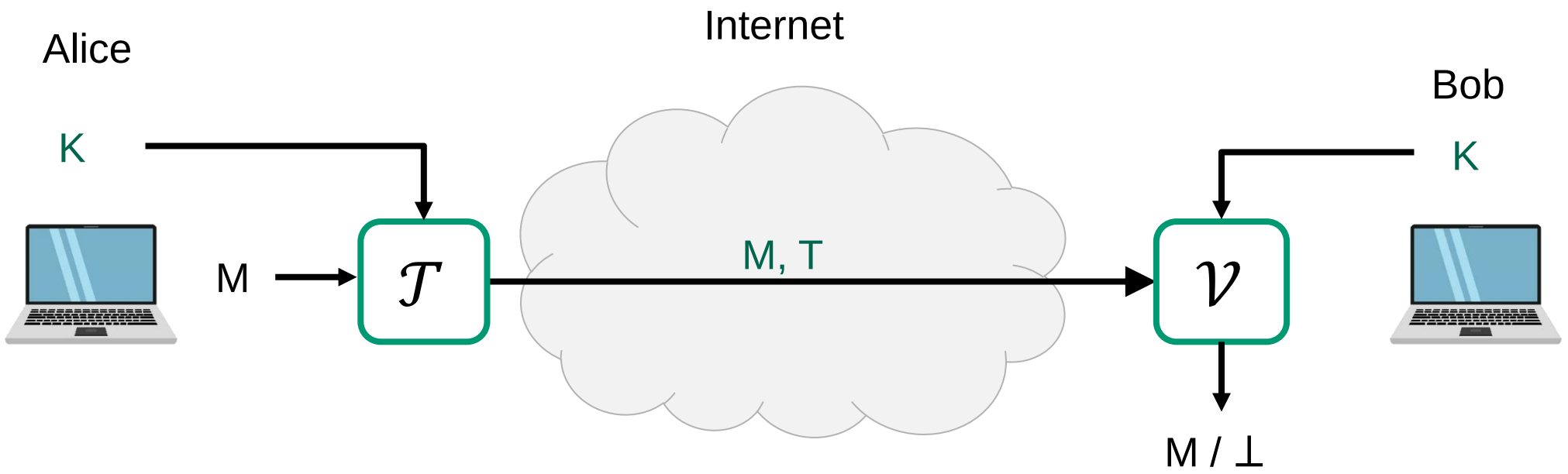
What is cryptography?



Security goals:

- **Data privacy:** adversary should not be able to read message M
- **Data integrity:** adversary should not be able to modify message M
- **Data authenticity:** message M really originated from Alice

MACs

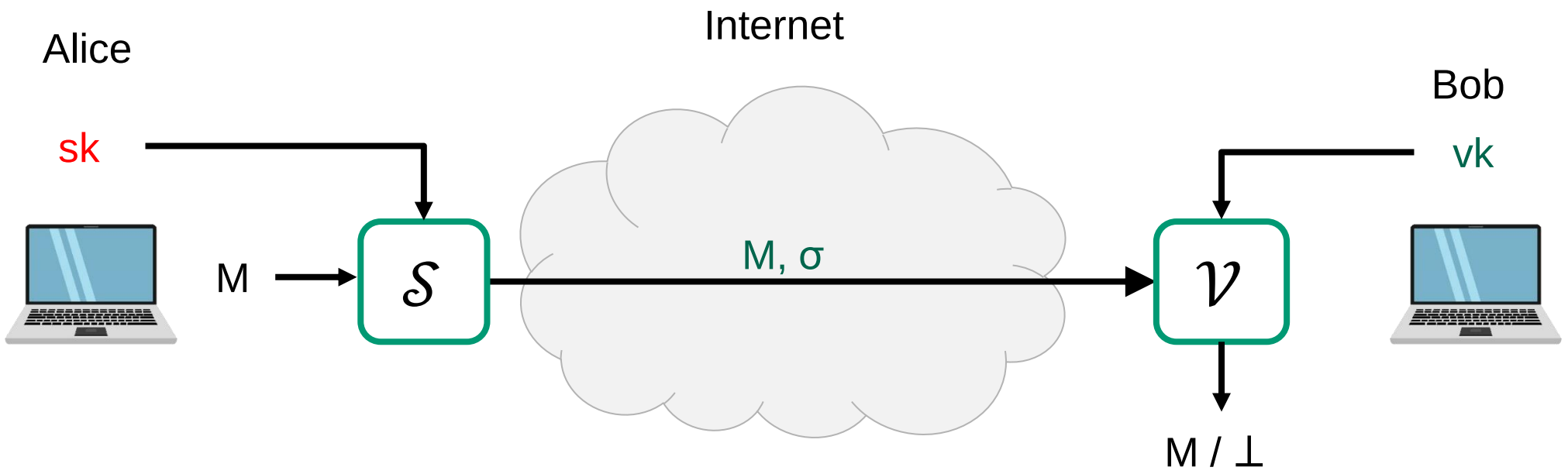


$\mathcal{T}$: tagging algorithm (public)

K : tagging / verification key (secret)

$\mathcal{V}$: verification algorithm (public)

Digital signatures



$\mathcal{T}$: tagging algorithm (public)

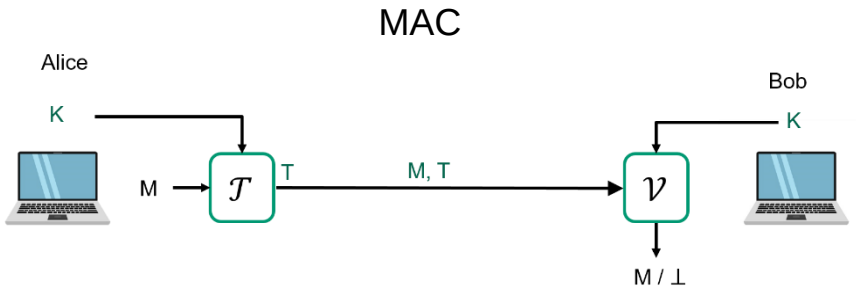
$\mathcal{V}$: verification algorithm (public)

sk : signing key (secret)

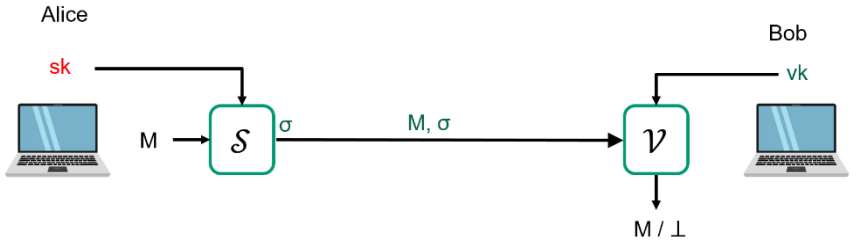
vk : verification key (public)

MACs vs. digital signatures

- MACs can only be verified by party sharing the same key



- Digital signatures can be verified by *anyone*

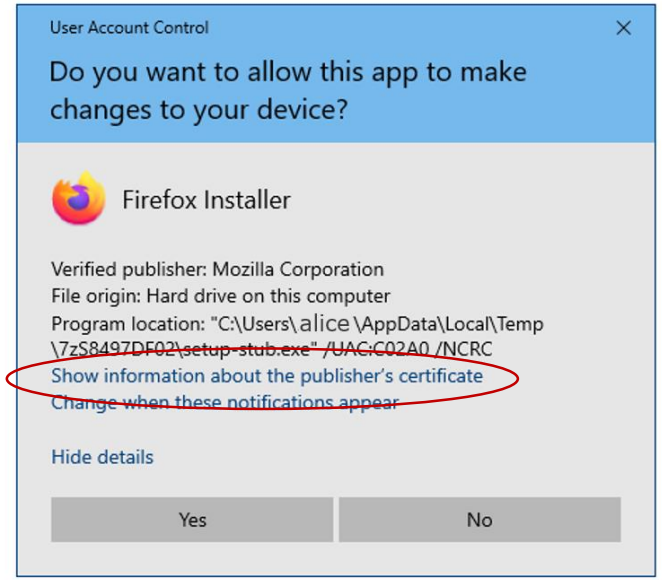


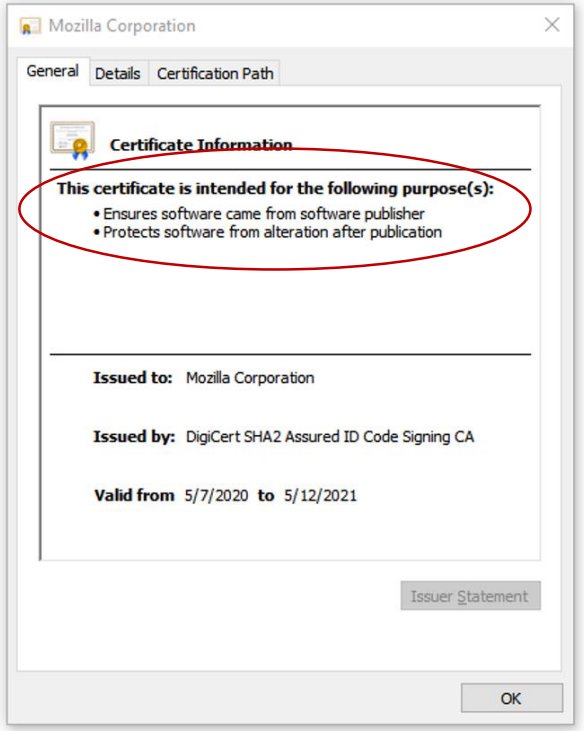
Digital signature

- **Non-repudiation:** Alice cannot deny having created σ
  - But she can deny having created T (since Bob could have done it)

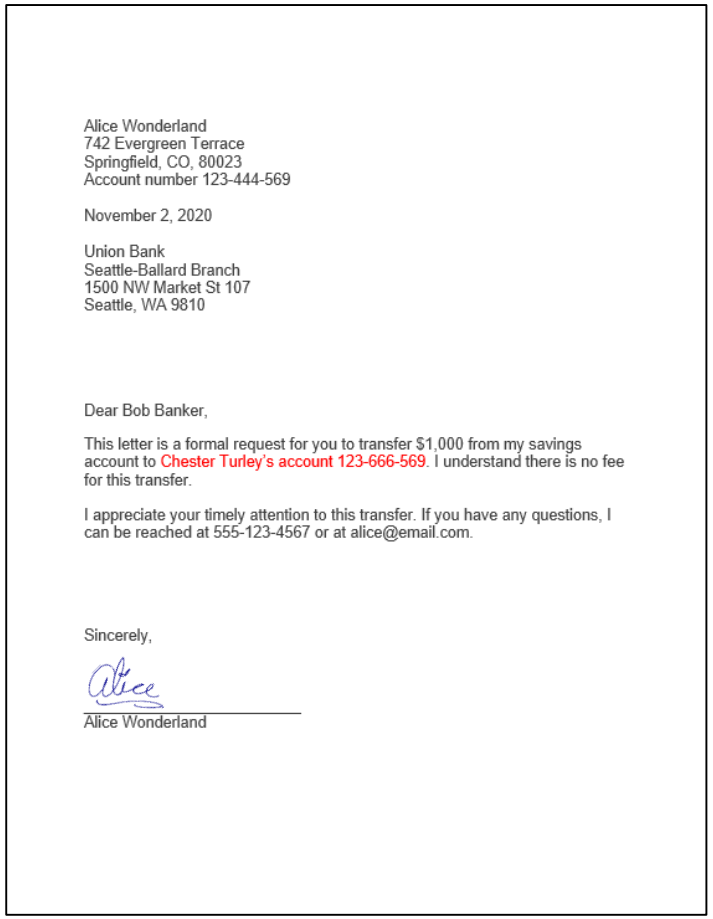
Applications of digital signatures

- Electronic document signing
- HTTPS / TLS certificates
- Software installation
- Email sender authentication
- Bitcoin





Signing electronically



Alice Wonderland
742 Evergreen Terrace
Springfield, CO, 80023
Account number 123-444-569

November 2, 2020

Union Bank
Seattle-Ballard Branch
1500 NW Market St 107
Seattle, WA 9810

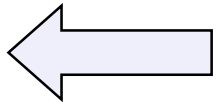
Dear Bob Banker,

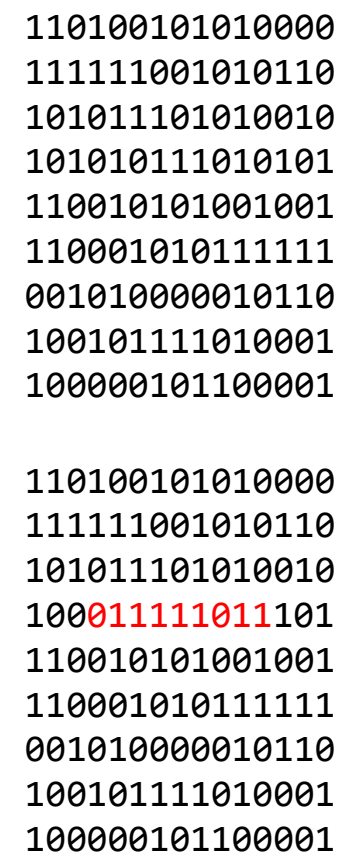
This letter is a formal request for you to transfer $1,000 from my savings account to Chester Turley's account 123-666-569. I understand there is no fee for this transfer.

I appreciate your timely attention to this transfer. If you have any questions, I can be reached at 555-123-4567 or at alice@email.com.

Sincerely,

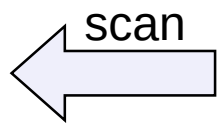
Alice Wonderland

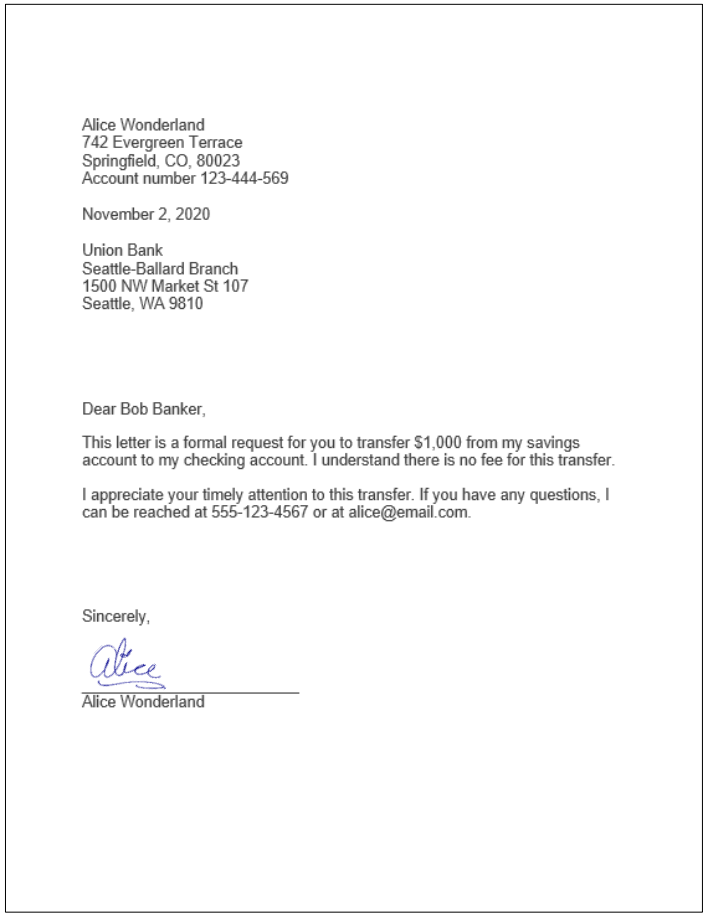




110100101010000
111111001010110
101011101010010
101010111010101
110010101001001
110001010111111
001010000010110
100101111010001
100000101100001

110100101010000
111111001010110
101011101010010
100011111011101
110010101001001
110001010111111
001010000010110
100101111010001
100000101100001





Alice Wonderland
742 Evergreen Terrace
Springfield, CO, 80023
Account number 123-444-569

November 2, 2020

Union Bank
Seattle-Ballard Branch
1500 NW Market St 107
Seattle, WA 9810

Dear Bob Banker,

This letter is a formal request for you to transfer $1,000 from my savings account to my checking account. I understand there is no fee for this transfer.

I appreciate your timely attention to this transfer. If you have any questions, I can be reached at 555-123-4567 or at alice@email.com.

Sincerely,

Alice Wonderland

Digital signatures – syntax

A **digital signature** scheme is a tuple of algorithms $\Sigma = (\text{KeyGen}, \text{Sign}, \text{Vrfy})$

$\text{KeyGen} : (\,) \to \mathcal{SK} \times \mathcal{VK}$

$(sk, vk) \xleftarrow{\$} \text{KeyGen}$

$\text{Sign} : \mathcal{SK} \times \mathcal{M} \to \mathcal{S}$

$\text{Sign}(sk, M) = \text{Sign}_{sk}(M) = \sigma$

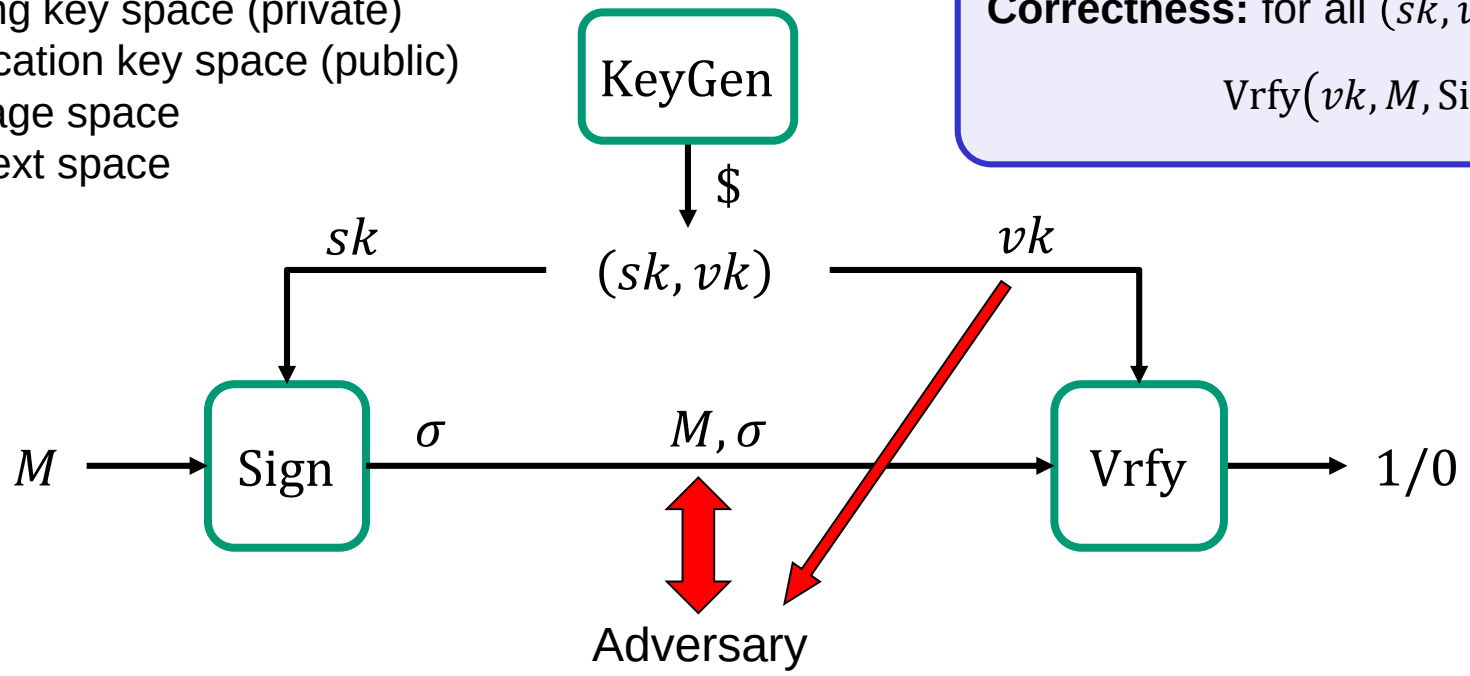
$\text{Vrfy} : \mathcal{VK} \times \mathcal{M} \times \mathcal{S} \to \{0,1\}$

$\text{Vrfy}(vk, M, \sigma) = \text{Vrfy}_{vk}(M, \sigma) = 1/0$

- $\mathcal{SK}$ – signing key space (private)
- $\mathcal{VK}$ – verification key space (public)
- $\mathcal{M}$ – message space
- $\mathcal{C}$ – ciphertext space

Correctness: for all $(sk, vk) \leftarrow \text{KeyGen}$:

$$\text{Vrfy}\big(vk, M, \text{Sign}(sk, M)\big) = 1$$



9

Digital signatures – security: UF-CMA

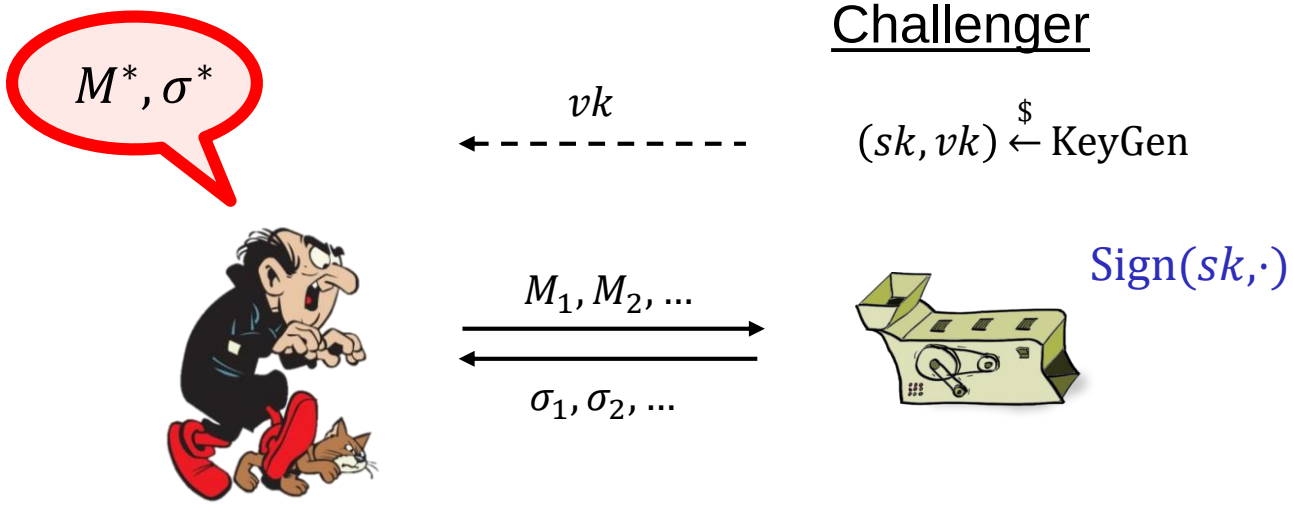
$\mathbf{Exp}_{\Sigma}^{\text{uf-cma}}(A)$

1. $(sk, vk) \xleftarrow{\$} \Sigma.\text{KeyGen}$
2. $\text{Msgs} \leftarrow [\,]$ // bookkeeping
3. $(M^*, \sigma^*) \leftarrow A^{\mathcal{S}(\cdot)}(vk)$
4. **if** $\Sigma.\text{Vrfy}(vk, M^*, \sigma^*) = 1$ and $M^* \notin \text{Msgs}$ **:**
5. **return** 1
6. **else**
7. **return** 0

$\mathcal{S}(M)$

---

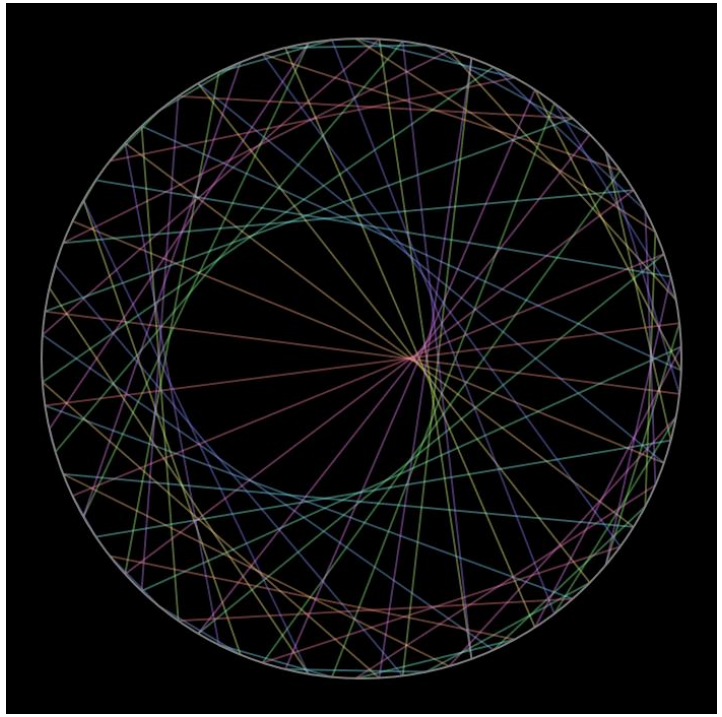
1. $\sigma \leftarrow \Sigma.\text{Sign}(sk, M)$
2. $\text{Msgs.add}(M)$
3. **return** σ



A has **forged** a signature if σ^* is valid for M^*

Definition: The **UF-CMA-advantage** of an adversary A is

$$\mathbf{Adv}_{\Sigma}^{\text{uf-cma}}(A) = \Pr\left[\mathbf{Exp}_{\Sigma}^{\text{uf-cma}}(A) \Rightarrow 1\right]$$



RSA signatures

Textbook RSA signatures

$n = p \cdot q$

$$C^e = M^e \bmod n$$

Textbook RSA signatures

$n = p \cdot q$

$$\sigma = \sigma^{ed} = M^{d} \bmod n$$

Textbook RSA signatures

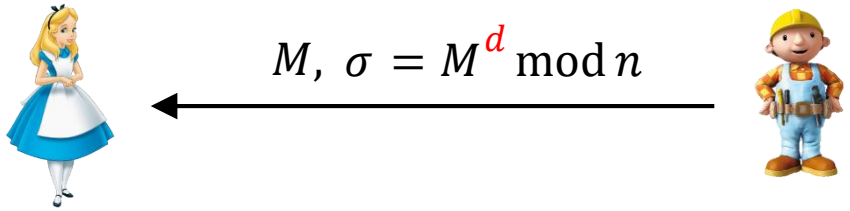
$n = p \cdot q$

RSA.KeyGen

1. $p, q \xleftarrow{\$}$ two random prime numbers
2. $n \leftarrow p \cdot q$
3. $\phi(n) = (p-1)(q-1)$
4. **choose** e such that $\gcd(e, \phi(n)) = 1$
5. $d \leftarrow e^{-1} \bmod \phi(n)$
6. $sk \leftarrow (n, d)$ $\quad pk \leftarrow (n, e)$
7. **return** (sk, pk)

RSA.Sign$\big((n, d), M\big)$

1. $\sigma \leftarrow M^d \bmod N$
2. **return** σ



RSA.Vrfy$\big((n, e), M, \sigma\big)$

1. **if** $\sigma^e = M \bmod N$ **then**
2. **return** 1
3. **else**
4. **return** 0

Textbook RSA signatures – (in)security

$A_1(n, e)$

1. $\sigma \xleftarrow{\$} \boldsymbol{Z}_n^*$

$\mathbf{Adv}_{\mathrm{RSA}}^{\text{uf-cma}}(A_1) = 1$

$\sigma^e = M \bmod n$

$A_2(n, e)$

1. Pick arbitrary $M_1 \in \boldsymbol{Z}_n^*$ (different from M)

$\mathbf{Adv}_{\mathrm{RSA}}^{\text{uf-cma}}(A_2) = 1$

$(\sigma_1 \cdot \sigma_2)^e \stackrel{?}{=} M \bmod n$

$\left(M_1^d \cdot M_2^d\right)^e = M_1^{ed} \cdot M_2^{ed} = M_1 \cdot M_2 = M_1 \cdot M \cdot M_1^{-1} = M \bmod n$

RSA.KeyGen

1. $p, q \xleftarrow{\$}$ two random prime numbers
2. $n \leftarrow p \cdot q$
3. $\phi(n) = (p-1)(q-1)$
4. **choose** e such that $\gcd(e, \phi(n)) = 1$
5. $d \leftarrow e^{-1} \bmod \phi(n)$
6. $sk \leftarrow (n, d)$ $\quad pk \leftarrow (n, e)$
7. **return** (sk, pk)

RSA.Sign$((n, d), M)$

1. $\sigma \leftarrow M^d \bmod N$
2. **return** σ

RSA.Vrfy$((n, e), M, \sigma)$

1. **if** $\sigma^e = M \bmod N$ **then**
2. $\quad$ **return** 1
3. **else**
4. $\quad$ **return** 0

Textbook RSA signatures

$H : \{0,1\}^* \to \boldsymbol{Z}_n^*$

RSA.KeyGen

1. $p, q \xleftarrow{\$}$ two random prime numbers
2. $n \leftarrow p \cdot q$
3. $\phi(n) = (p-1)(q-1)$
4. **choose** e such that $\gcd(e, \phi(n)) = 1$
5. $d \leftarrow e^{-1} \bmod \phi(n)$
6. $sk \leftarrow (n, d)$ $\quad pk \leftarrow (n, e)$
7. **return** (sk, pk)

RSA.Sign$\big((n, d), M\big)$

1. $\sigma \leftarrow M^d \bmod N$
2. **return** σ

$M,\ \sigma = M^d \bmod n$

RSA.Vrfy$\big((n, e), M, \sigma\big)$

1. **if** $\sigma^e = M \bmod N$ **then**
2. $\quad$ **return** 1
3. **else**
4. $\quad$ **return** 0

RSA message space:

$\mathcal{M} = \boldsymbol{Z}_n^*$

$\mathcal{M} = \{0,1\}^*$

Actually want

Hashed-RSA

$H : \{0,1\}^* \to \boldsymbol{Z}_n^*$

RSA.KeyGen

1. $p, q \xleftarrow{\$}$ two random prime numbers
2. $n \leftarrow p \cdot q$
3. $\phi(n) = (p-1)(q-1)$
4. **choose** e such that $\gcd(e, \phi(n)) = 1$
5. $d \leftarrow e^{-1} \bmod \phi(n)$
6. $sk \leftarrow (n, d)$ $\quad pk \leftarrow (n, e)$
7. **return** (sk, pk)

RSA.Sign$\big((n, d), M\big)$

1. $\sigma \leftarrow H(M)^d \bmod N$
2. **return** σ

$M,\ \sigma = H(M)^d \bmod n$

RSA.Vrfy$\big((n, e), M, \sigma\big)$

1. **if** $\sigma^e = H(M) \bmod N$ **then**
2. $\quad$ **return** 1
3. **else**
4. $\quad$ **return** 0

Hashed-RSA – security

$A_1(n, e)$

1. $\sigma \xleftarrow{\$} \boldsymbol{Z}_n^*$
2. $M \leftarrow \sigma^e \bmod n$
3. ~~Output (M, σ)~~ Output (X, σ) // $H(X) = M$

$\sigma^e = H(M) \bmod n$

Hard to find!

$A_2(n, e)$

1. Pick arbitrary $M_1 \in \boldsymbol{Z}_n^*$ (different from M)
2. $M_2 \leftarrow$ ~~$M \cdot M_1^{-1} \bmod n$~~ // $H(M_1) \cdot H(M_2) = H(M)$
3. Query $\sigma_1 \leftarrow \mathcal{S}(M_1)$ and $\sigma_2 \leftarrow \mathcal{S}(M_2)$
4. Output $(M, \sigma_1 \cdot \sigma_2 \bmod n)$

$(\sigma_1 \cdot \sigma_2)^e \stackrel{?}{=} H(M) \bmod n$

Hard to find!

$\left(H(M_1)^d \cdot H(M_2)^d\right)^e = H(M_1) \cdot H(M_2) \neq H(M) \bmod n$

RSA.KeyGen

1. $p, q \xleftarrow{\$}$ two random prime numbers
2. $n \leftarrow p \cdot q$
3. $\phi(n) = (p-1)(q-1)$
4. **choose** e such that $\gcd\left(e, \phi(n)\right) = 1$
5. $d \leftarrow e^{-1} \bmod \phi(n)$
6. $sk \leftarrow (n, d)$ $\quad pk \leftarrow (n, e)$
7. **return** (sk, pk)

RSA.Sign$\left((n, d), M\right)$

1. $\sigma \leftarrow H(M)^d \bmod N$
2. **return** σ

RSA.Vrfy$\left((n, e), M, \sigma\right)$

1. **if** $\sigma^e = H(M) \bmod N$ **then**
2. $\quad$ **return** 1
3. **else**
4. $\quad$ **return** 0

Hashed-RSA – security

- Factoring + RSA-problem must be hard
- What are the requirements of H?
  - Must be collision-resistant:

$$H(X) = H(Y) \implies H(X)^d = H(Y)^d = \sigma$$

  - Is this enough?
    - Unknown
    - However, if we assume that H is *perfect** then

Theorem: For *any* UF-CMA adversy A against Hashed-RSA making q sign queries, there is an algorithm B solving the RSA-problem:

$$\mathbf{Adv}^{\text{uf-cma}}_{\text{H-RSA}}(A) \leq q \cdot \mathbf{Adv}^{\text{RSA}}_{n,e}(B)$$

where H is assumed perfect*

*random oracle

RSA.KeyGen

1. $p, q \xleftarrow{\$}$ two random prime numbers
2. $n \leftarrow p \cdot q$
3. $\phi(n) = (p-1)(q-1)$
4. **choose** e such that $\gcd(e, \phi(n)) = 1$
5. $d \leftarrow e^{-1} \bmod \phi(n)$
6. $sk \leftarrow (n, d)$ $\quad$ $pk \leftarrow (n, e)$
7. **return** (sk, pk)

RSA.Sign$((n, d), M)$

1. $\sigma \leftarrow H(M)^d \bmod N$
2. **return** σ

RSA.Vrfy$((n, e), M, \sigma)$

1. **if** $\sigma^e = H(M) \bmod N$ **then**
2. $\quad$ **return** 1
3. **else**
4. $\quad$ **return** 0

$$Y = g^{x}$$

Signatures from discrete logarithms

Schnorr signatures

$(G,*) = \langle g \rangle$

$H : G \times \mathcal{M} \to \boldsymbol{Z}_p^*$

KeyGen

1. $b \xleftarrow{\$} \{1 \dots |G|\}$
2. $B \leftarrow g^b$
3. **return** $(sk = b, vk = B)$

Sign$(sk = b, M)$

1. $r \xleftarrow{\$} \{1 \dots |G|\}$
2. $R \leftarrow g^r$
3. $h \leftarrow H(R, M)$
4. $s \leftarrow r - bh \bmod p$
5. **return** $\sigma = (h, s)$

M, σ

Vrfy$\big(vk = B, M, \sigma = (h, s)\big)$

1. $R' \leftarrow g^s * B^h$
2. $h' \leftarrow H(R', M)$
3. **if** $h' = h$ **then**
4. **return** 1
5. **else**
6. **return** 0

Correctness: $\mathrm{Vrfy}\big(vk, M, \mathrm{Sign}(sk, M)\big) = 1$

$$h' = H(R', M) = H\big(g^s B^h, M\big) = H\big(g^{r-bh} g^{bh}, M\big) = H\big(g^{r-bh+bh}, M\big) = H(g^r, M) = H(R, M) = h$$

Schnorr signatures

$(G,*) = \langle g \rangle$

$H : G \times \mathcal{M} \to \boldsymbol{Z}_p^*$

KeyGen

1. $b \xleftarrow{\$} \{1 \dots |G|\}$
2. $B \leftarrow g^b$
3. **return** $(sk = b, vk = B)$

Vrfy$\big(vk = B, M, \sigma = (h,s)\big)$

1. $R' \leftarrow g^s * B^h$
2. $h' \leftarrow H(R', M)$
3. **if** $h' = h$ **then**
4. **return** 1
5. **else**
6. **return** 0

M, σ

Sign$(sk = b, M)$

1. $r \xleftarrow{\$} \{1 \dots |G|\}$
2. $R \leftarrow g^r$
3. $h \leftarrow H(R, M)$
4. $s \leftarrow r - bh \bmod p$
5. **return** $\sigma = (h,s)$

Security:

- DLOG must be hard in G
- H must be collision-resistant, one-way, etc.
- Attacker must essentially solve $g^r = g^s * g^{bh} \iff r = s + bh \iff s = r - bh$
- r must be picked new *every time*!

$\sigma = (h,s)$

$\sigma' = (h',s')$

$\overset{\text{same } r}{\Longrightarrow} \quad s - s' = (r - bh) - (r - bh') = b \cdot (h' - h) \implies b = (s - s') \cdot (h' - h)^{-1} \bmod p$

Learned private long-term key!

Discrete-log-based signatures

- Schnorr
  - Elegant design
  - Has formal security proof (based on DLOG problem and H assumed perfect)
  - Was patented

- (EC)DSA
  - Non-patented alternative
  - More complicated design than Schnorr
  - No security proof
  - Standardized by NIST (designed by NSA)
  - Very widely used
  - Same sharp edge as Schnorr
    - Reuse of r leaks long-term signing key
    - Broke all PlayStation 3's produced by Sony


CNBC

MARKETS BUSINESS INVESTING TECH POLITICS CNBC TV WATCHLIST

VIDEO GAMES

Sony: PlayStation Breach Involves 70 Million Subscribers

Chris Morris | @MorrisatLarge

Published 4:30 PM ET Tue, 26 April 2011 | Updated 5:26 PM ET Tue, 26 April 2011

Public-key infrastructure (PKI)

What are identities?

"Alice", g^a

"Bob", g^b

$K \leftarrow g^{ab}$

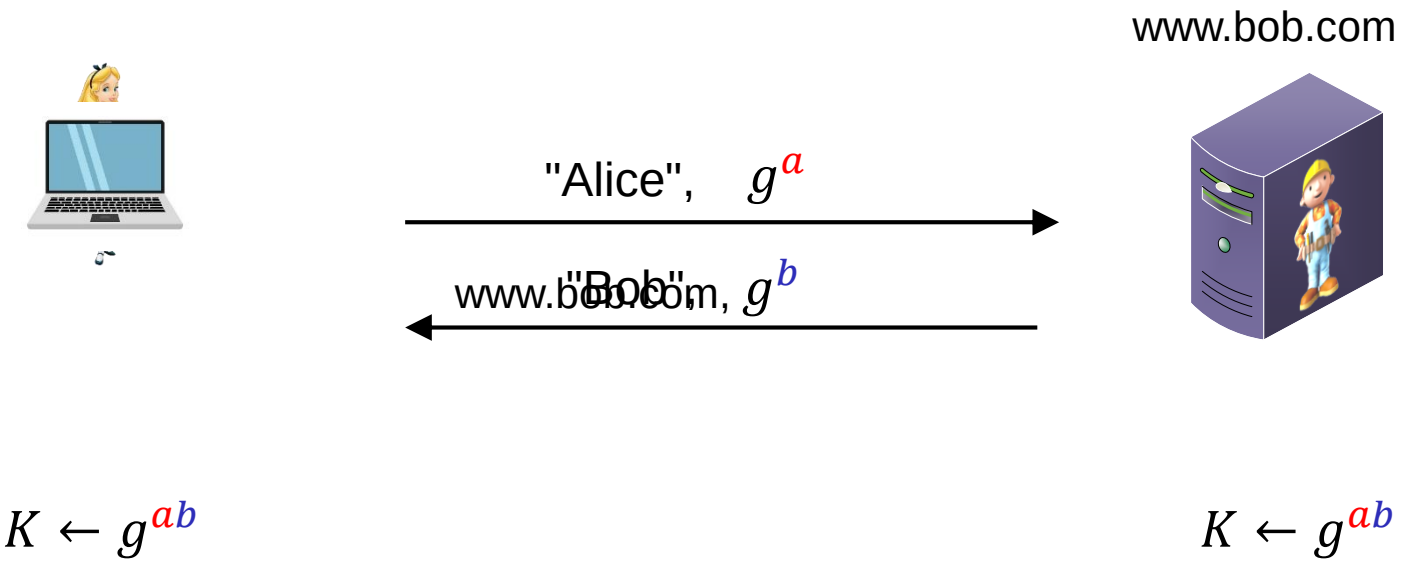
$K \leftarrow g^{ab}$

There are many Alice's and many Bob's

How do we know that g^a belongs to *this* particular Alice, and g^b to this particular Bob?

Need to **bind** public keys to entities

Identities on the internet

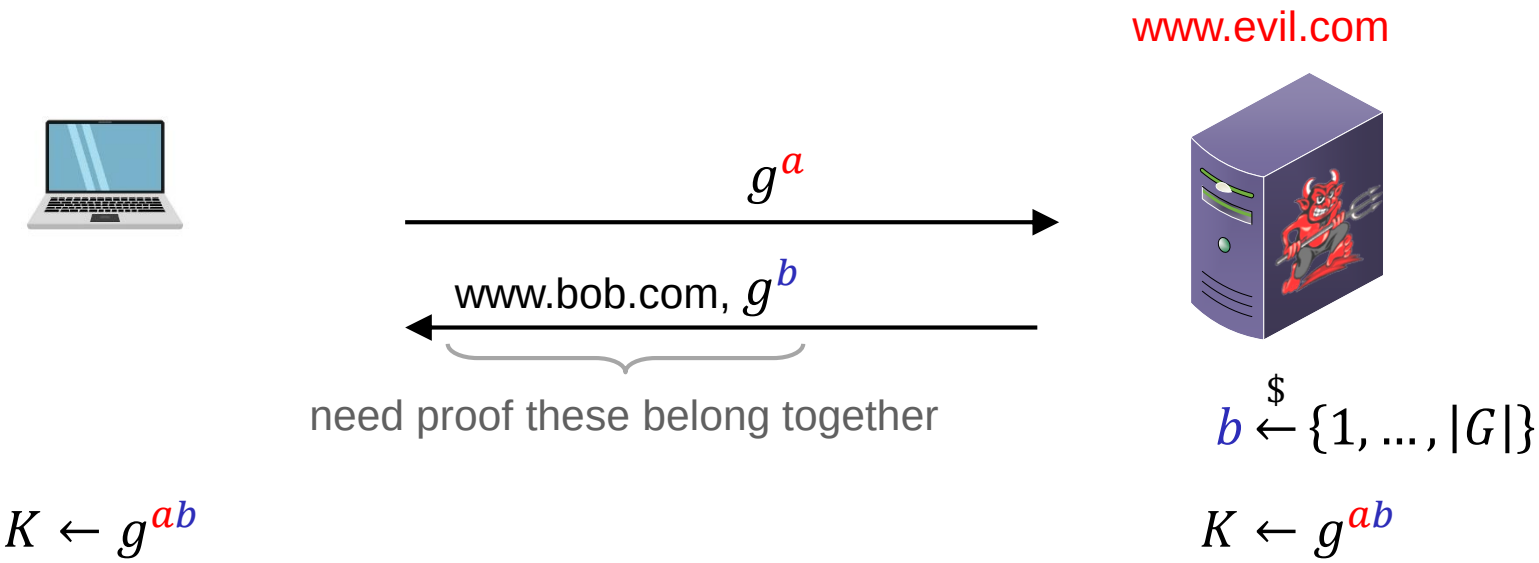


There are many Alice's and many Bob's

How do we know that g^a belongs to *this* particular Alice, and g^b to this particular Bob?

Need to **bind** public keys to entities – internet: bind public keys to **domain names**

Identities on the internet

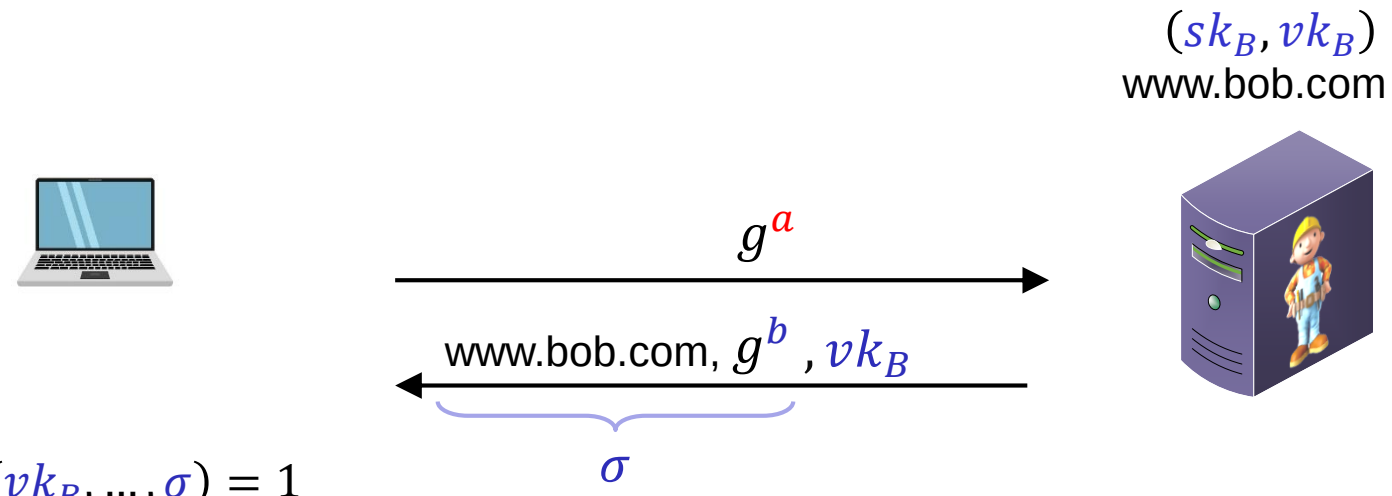


There are many Alice's and many Bob's

How do we know that g^a belongs to *this* particular Alice, and g^b to this particular Bob?

Need to **bind** public keys to entities – internet: bind public keys to **domain names**

Authenticated key exchange



if $\mathrm{Vrfy}(vk_B, \ldots, \sigma) = 1$

$K \leftarrow g^{ab}$

else

abort

$K \leftarrow g^{ab}$

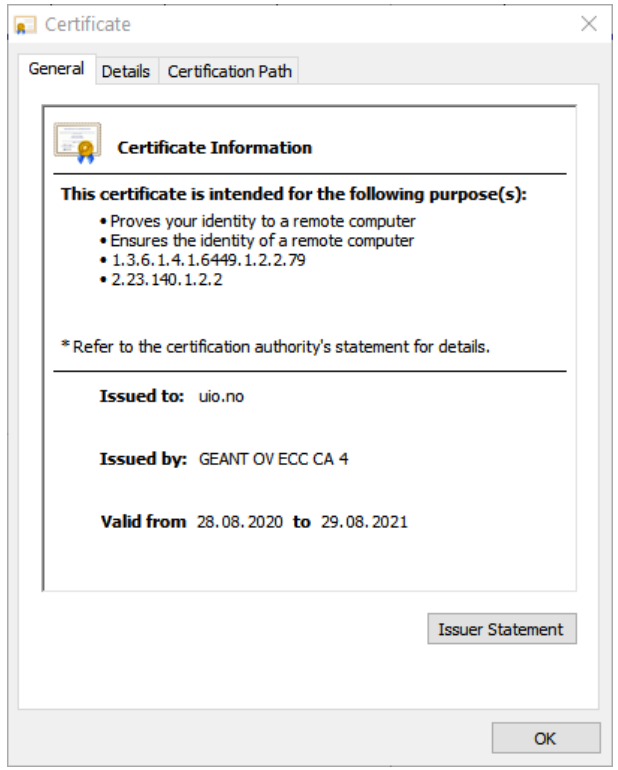
But why should we trust this vk_B? Could have been created by the adversary itself

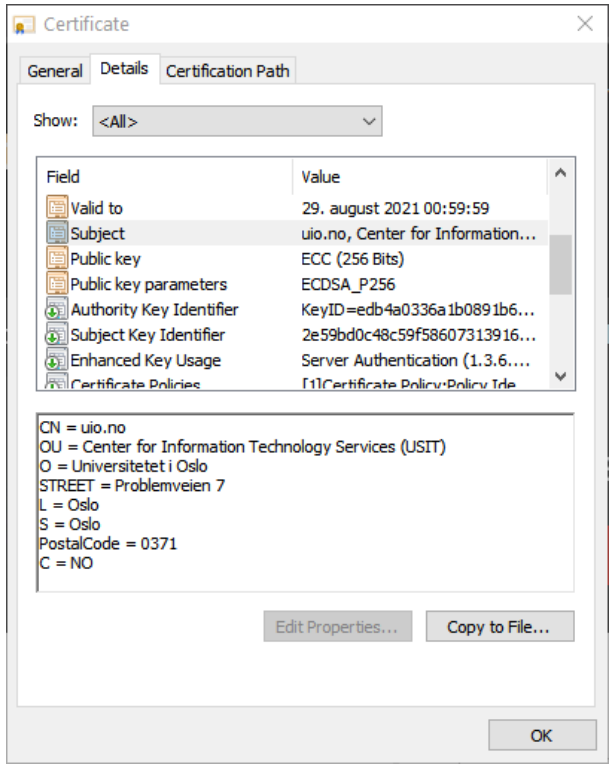
Digital certificates

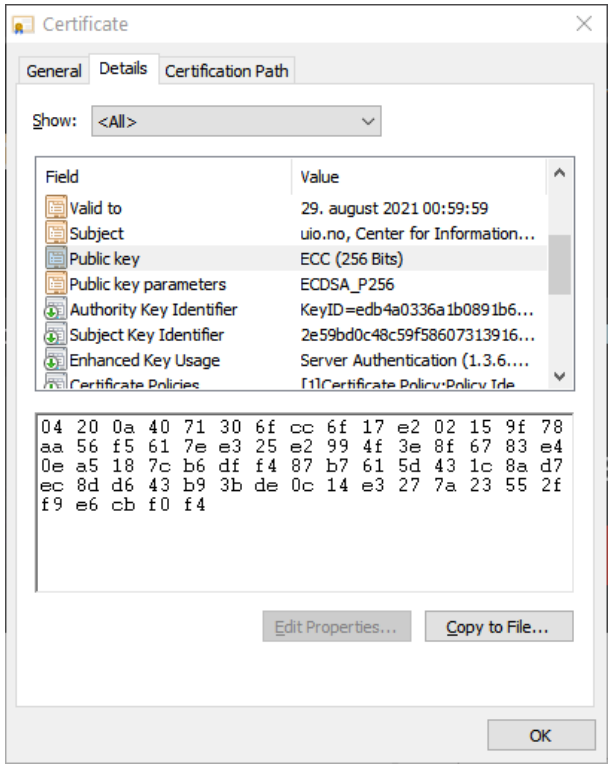
- **Digital certificate:** a way of binding a public key to an entity

- A certificate consists of:
  - The public key of the entity
  - A bunch of information identifying the entity
    - Name
    - Address
    - Occupation
    - URL
    - Email-address
    - Phone number
    - …
  - A *digital signature* on all the above by a **certificate authority (CA)**

Digital certificates







Certificate authorities (CA)

- **CA:** an issuer of digital certificates

- Acts as a trusted third-party, certifying (i.e., signing) the public keys of other entities
  - Verifies the identity of a claimed public-key owner

- The basis of a **public-key infrastructure (PKI)**

Authenticated key exchange + PKI

(sk_B, vk_B)
www.bob.com

g^a

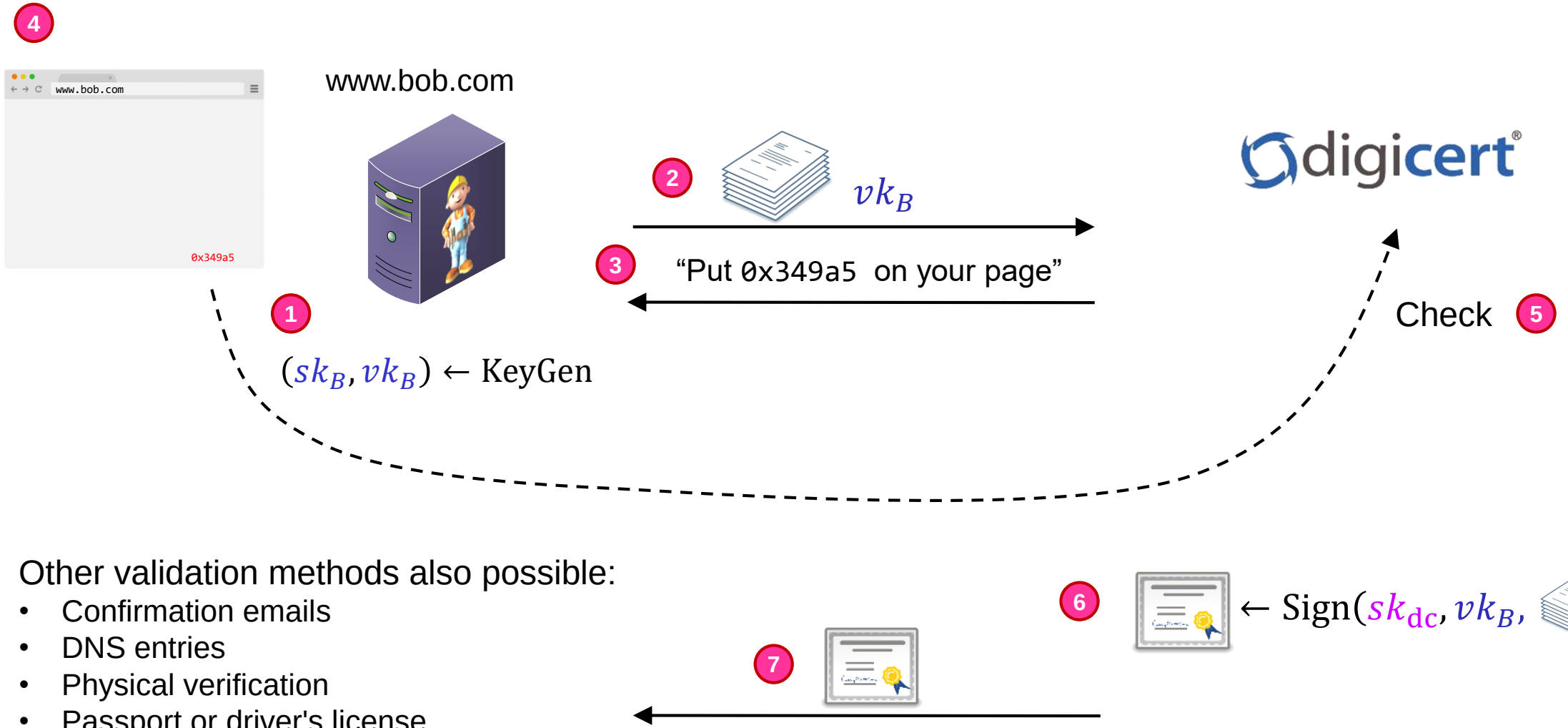
www.bob.com, g^b, vk_B

σ σ vk_{CA}

if $\text{Vrfy}(vk_B, \ldots, \sigma) = 1$
and $\text{Vrfy}(vk_{\text{CA}}, \ldots, \sigma) = 1$
$K \leftarrow g^{ab}$
else
abort

$K \leftarrow g^{ab}$

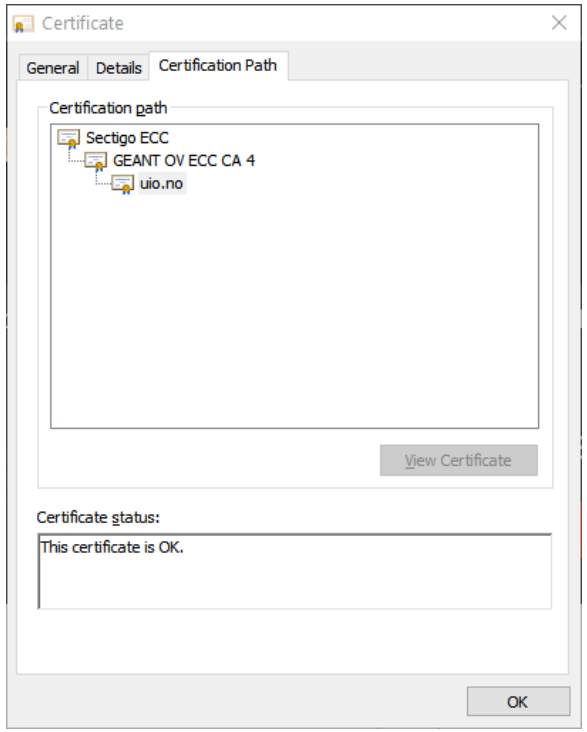
How to get a signed certificate?



Other validation methods also possible:

- Confirmation emails
- DNS entries
- Physical verification
- Passport or driver's license

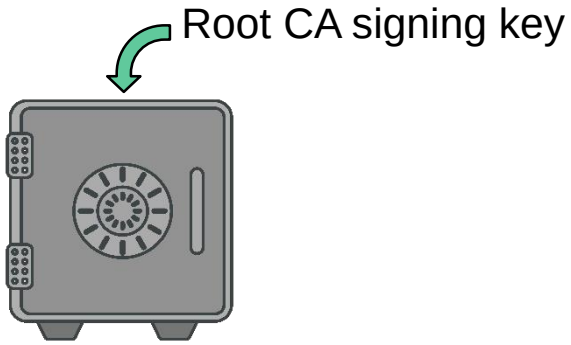
Certificate chains



Root CAs

- **Root CAs:** CAs that sign other CAs' public keys
  - \+ only a few root CAs need to be trusted by end-users
  - \+ root CAs can distribute the signing + verification load to smaller CAs
  - \- single point of failure; private key must be *very heavily* guarded

  - Root CAs for the internet: a few large multinational corporations



COMODO

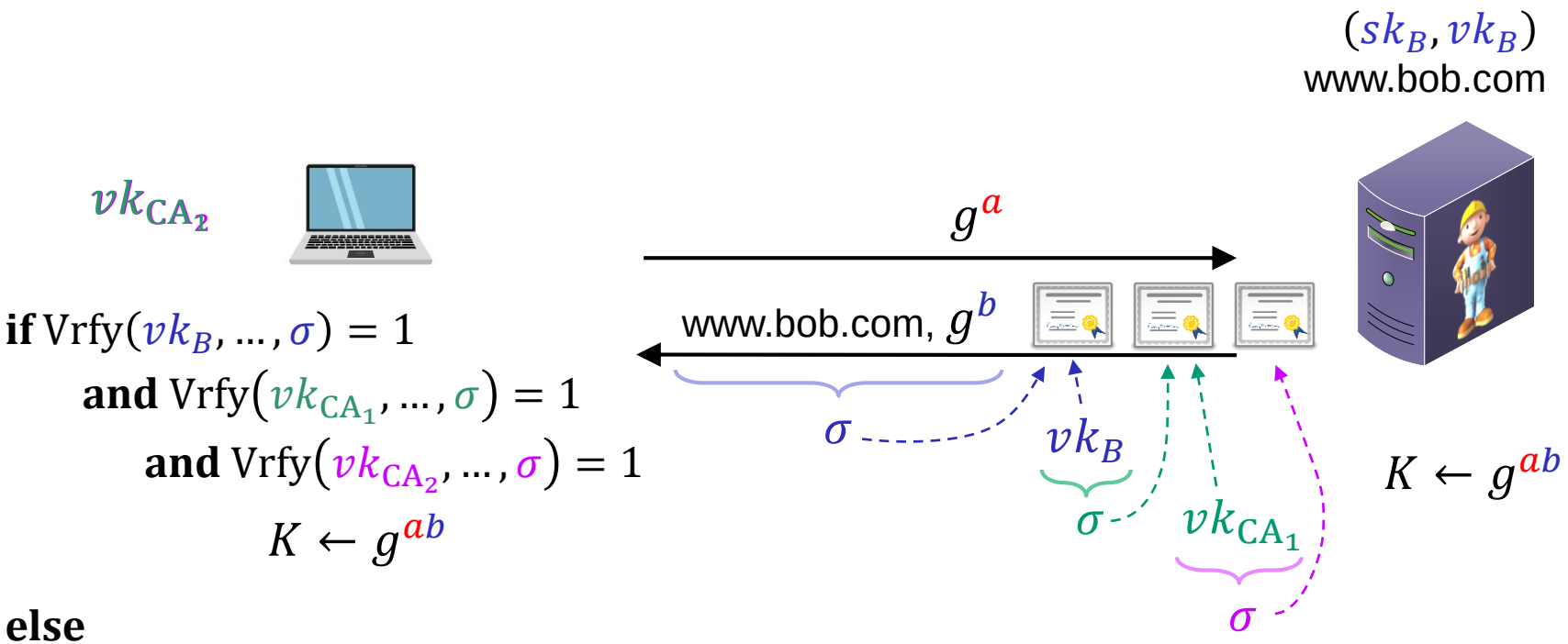
IdenTrust part of HID Global

digicert

GoDaddy

Let's Encrypt

HTTPS / TLS + PKI



How to become an internet root CA?

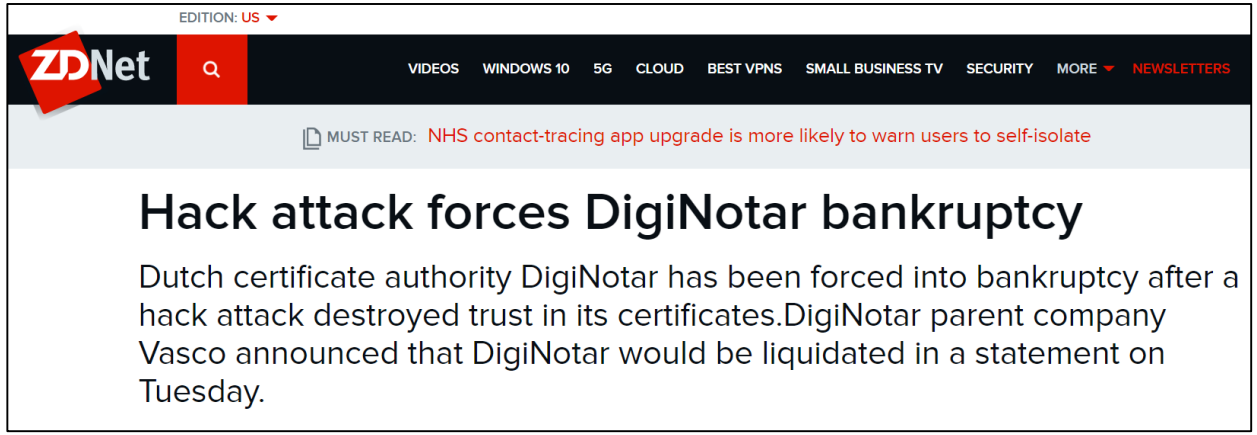
- Need to prove yourself (trust)worthy to browser and OS vendors
  - Microsoft Root Certificate Program
  - Mozilla CA Certificate Program
  - Apple Root Certificate Program
  - Chrome Root CA Program

- Lot's of auditing and paperwork

- Many formal technical and non-technical security requirements
  - CA/Browser forum
  - Baseline Requirements v1.7.3

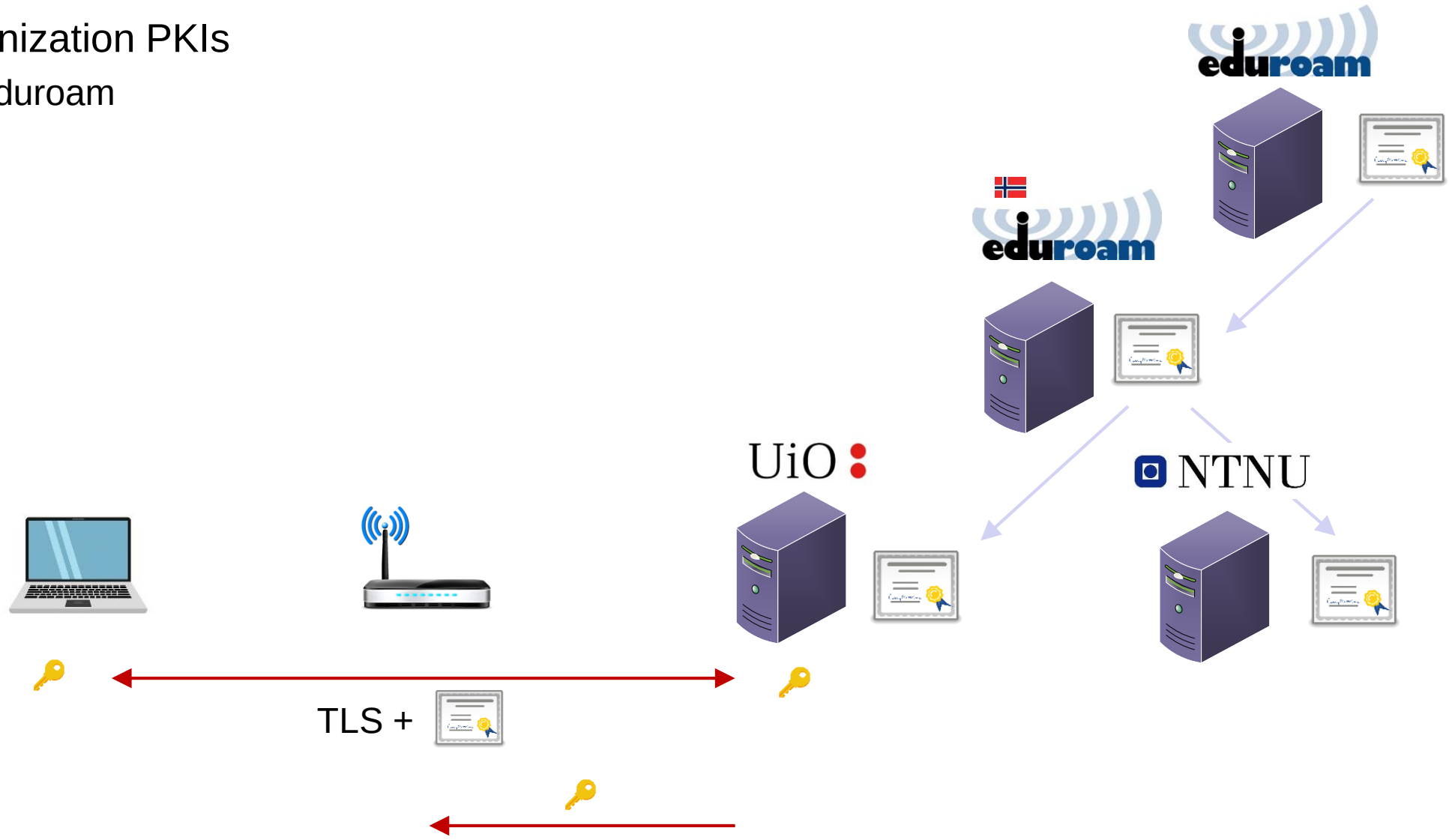
DigiNotar

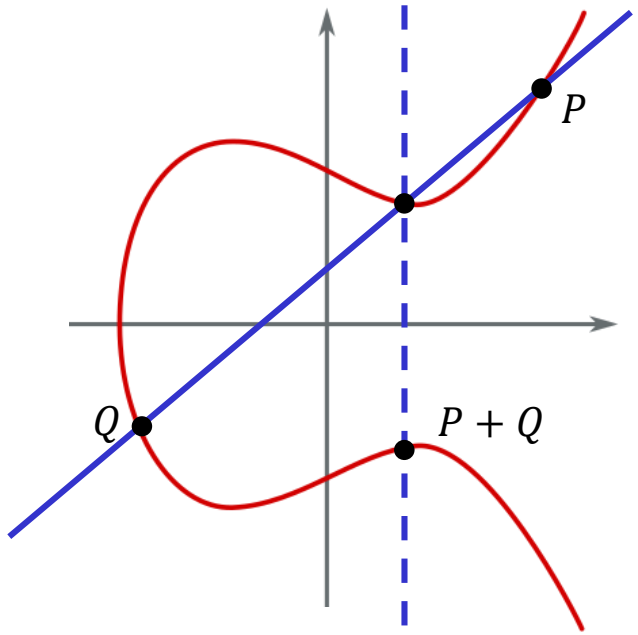
- Dutch root CA
- Lost control of their private signing key in 2011
- Fraudulent certificates issued for Gmail, Yahoo!, Mozilla, WordPress, …
- 30 000 Iranian Gmail users targeted



Other PKIs exist

- Organization PKIs
  - eduroam





$y^2 = x^3 + ax + b$

$a, b, x, y \in \boldsymbol{R}$

End of Part II
(Asymmetric crypto)

Summary of asymmetric cryptography

Primitive	Functionality + syntax	Hardness assumption / security goal	Acronym	Examples
Diffie-Hellman	Derive shared value (key) in a cyclic group $A^b = g^{ab} = B^a$	Discrete logarithm (DLOG) Diffie-Hellman (DH) Decisional Diffie-Hellman (DDH)	PRF	$(\boldsymbol{Z}_p^*,\cdot)$ −DH $(E(\boldsymbol{F}_p),+)$ −DH
RSA function	One-way trapdoor permutation	Factoring problem RSA-problem		Textbook RSA
Public-key encryption	Encrypt variable-length input $\mathrm{Enc} : \mathcal{PK} \times \mathcal{M} \to \mathcal{C}$	Confidentiality: attacker should learn nothing about plaintext (except length) from ciphertexts	IND-CPA IND-CCA	ElGamal Hashed/Padded RSA Fujisaki-Okomoto-transform
Digital signatures	Create signature on variable length input $\mathrm{Sign} : \mathcal{SK} \times \mathcal{M} \to \mathcal{S}$ $\mathrm{Vrfy} : \mathcal{VK} \times \mathcal{M} \times \mathcal{S} \to \{1,0\}$	Integrity: attacker shouldn't be able to forge messages, i.e., create new messages with valid signatures	UF-CMA	Schnorr Hashed-RSA ECDSA

Cryptographic groups	Comment	Computational problem	Best-known attack	Common sizes
$(\boldsymbol{Z}_p^*,\cdot)$	p prime $\lvert\boldsymbol{Z}_p^*\rvert = p-1$	Discrete logarithm	General number field sieve (GNFS)	$\lvert p\rvert \approx 2000-3000$ bits
Subgroups $H < (\boldsymbol{Z}_p^*,\cdot)$	$\lvert H\rvert = q$ (typically prime)	Discrete logarithm	GNFS	$\lvert q\rvert \approx 256$ bits
$(E(\boldsymbol{F}_p),+)$	p prime $\lvert E(\boldsymbol{F}_p)\rvert = q$ (typically) prime $p \neq q$	Discrete logarithm	Generic attacks: Baby-step giant-step, Pollard-rho, Pohlig-Hellman	$\lvert E(\boldsymbol{F}_p)\rvert \approx 256$ bits $\lvert p\rvert \approx 256$ bits
$(\boldsymbol{Z}_n^*,\cdot)$	n not prime $\lvert\boldsymbol{Z}_n^*\rvert = \phi(n)$	Factoring	GNFS	$\lvert n\rvert \approx 2000-4000$ bits

Next week

- Quantum computers