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

TEK4500

03.11.2020

Håkon Jacobsen

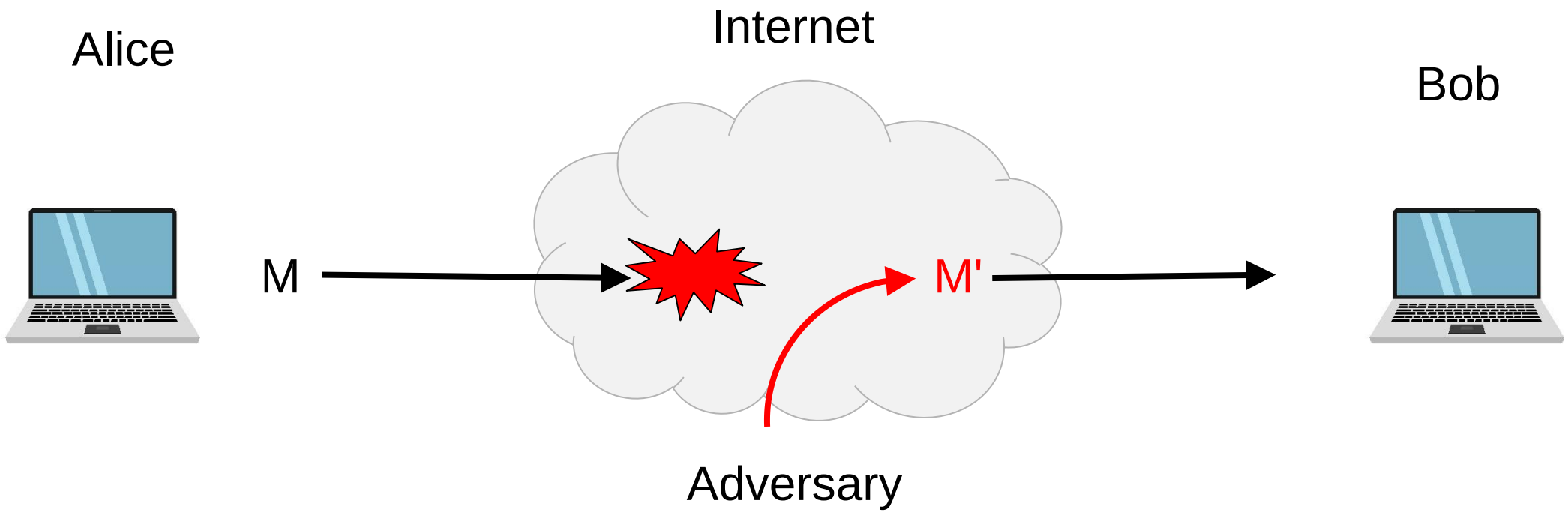
hakon.jacobsen@its.uio.no

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)

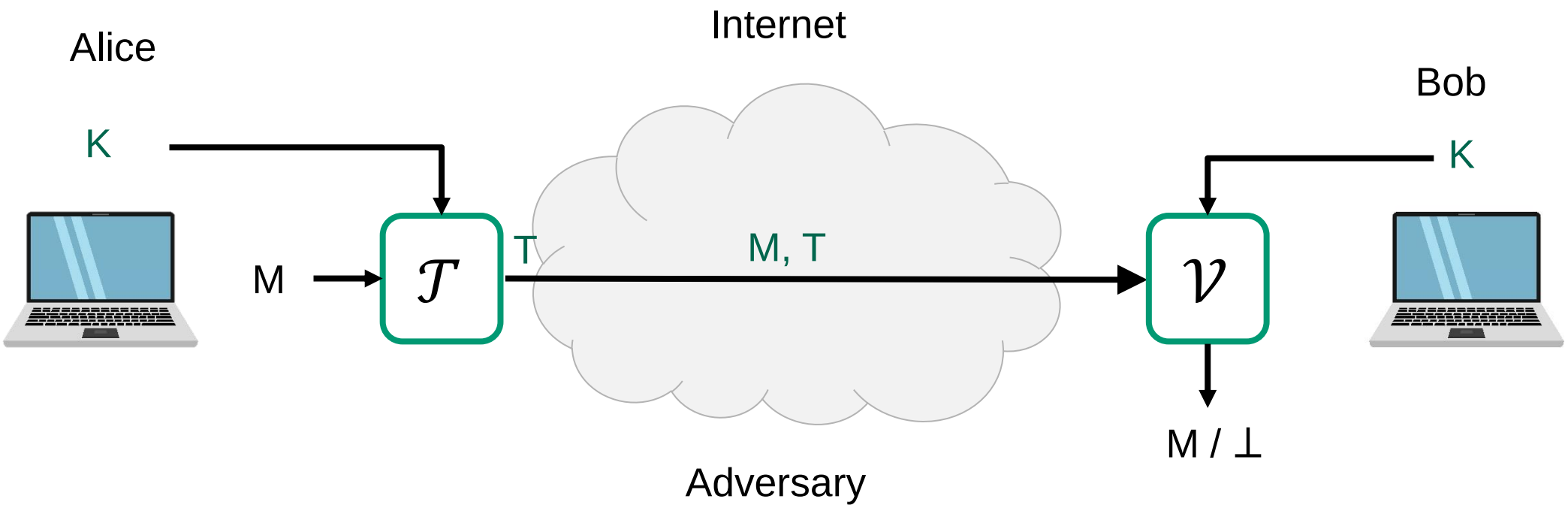
What is cryptography?



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

Achieving integrity: MACs

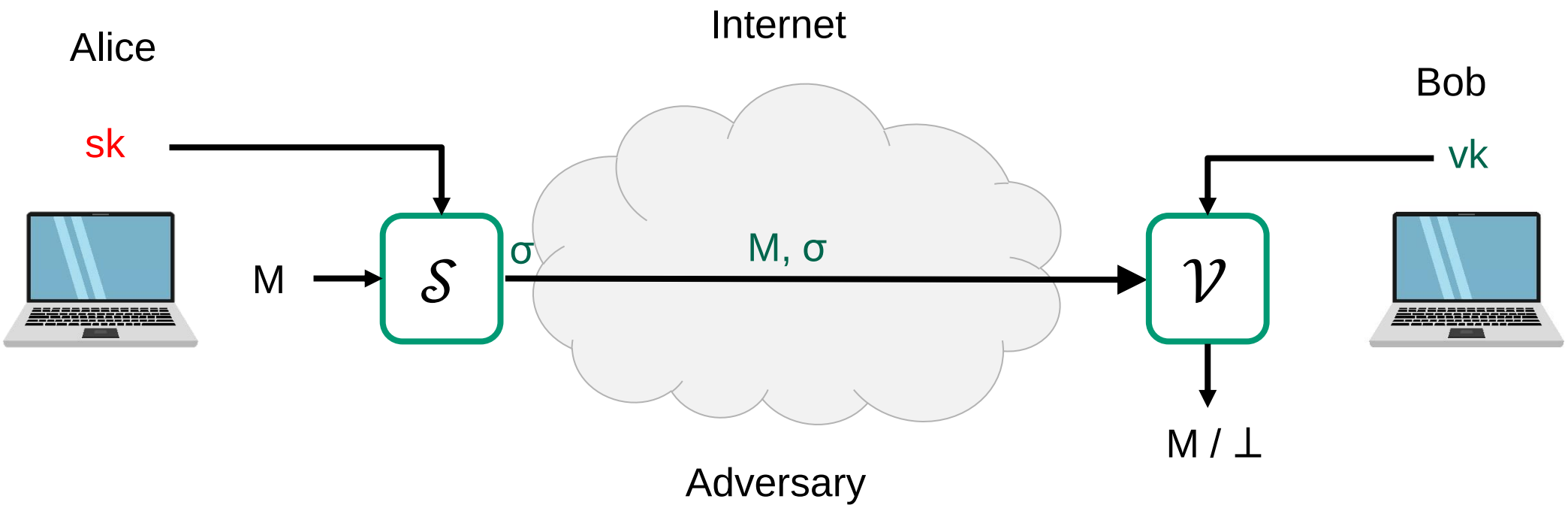


$\mathcal{T}$: tagging algorithm (public)

K : tagging / verification key (secret)

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

Achieving integrity: digital signatures



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

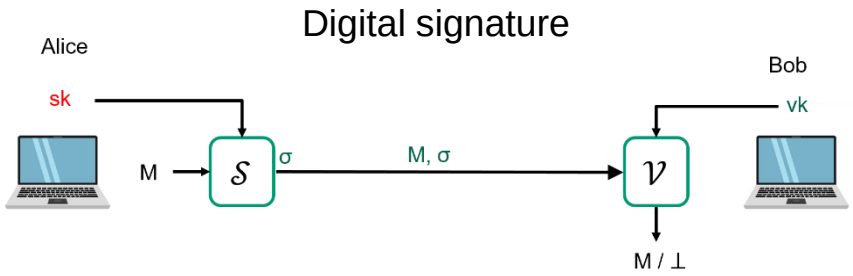
sk : signing key (secret)

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

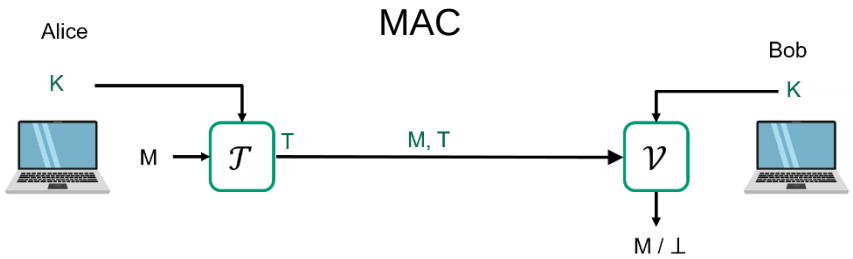
vk : verification key (public)

Digital signatures vs. MACs

- Digital signatures can be verified by *anyone*



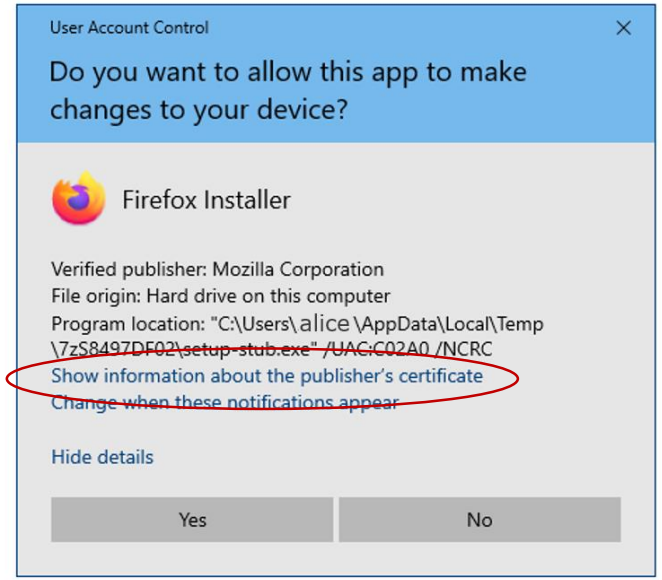
- MACs can only be verified by party sharing the same key

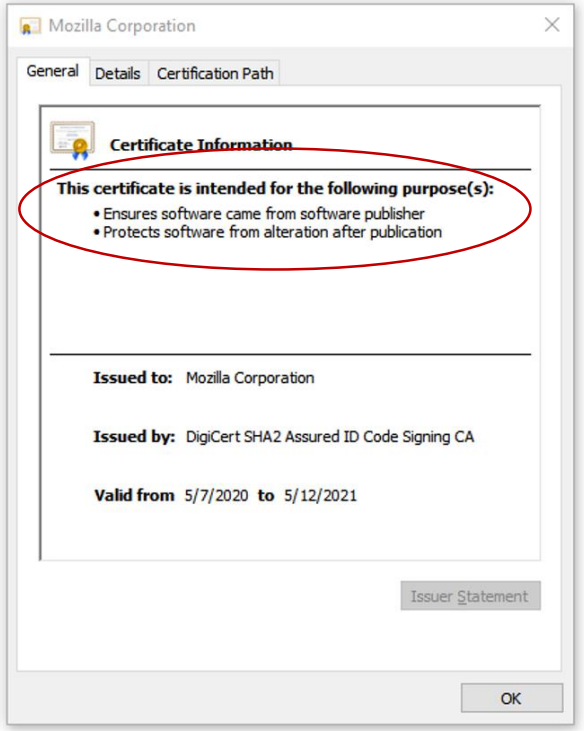


- **Non-repudiation:** Alice cannot deny having created σ
  - But she can deny having created T (since Bob could have done it)

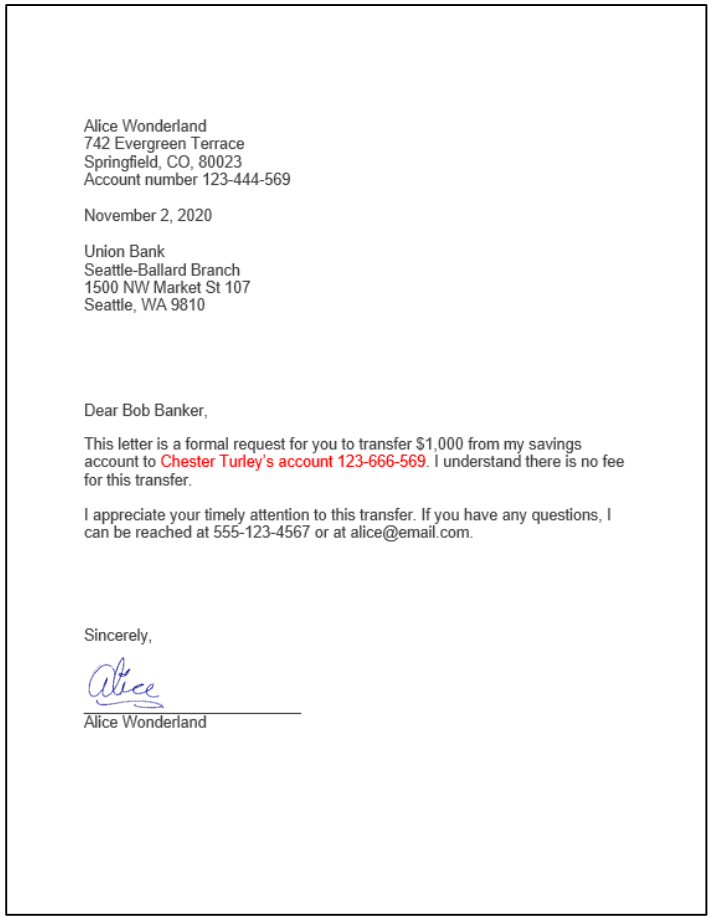
Applications of digital signatures

- 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

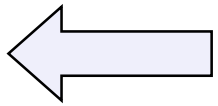
Dear Bob Banker,

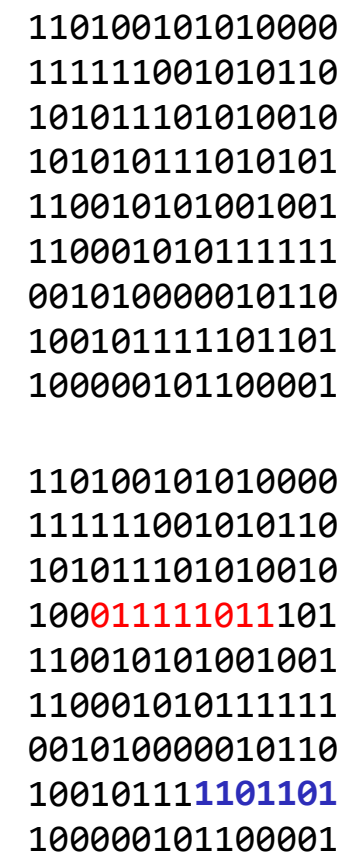
This letter is a formal request for you to transfer $1,000 from my savings account to Chester Turley's account 123-666-569. I understand there is no fee for this transfer.

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,

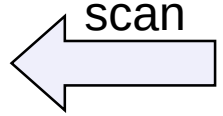
Alice Wonderland

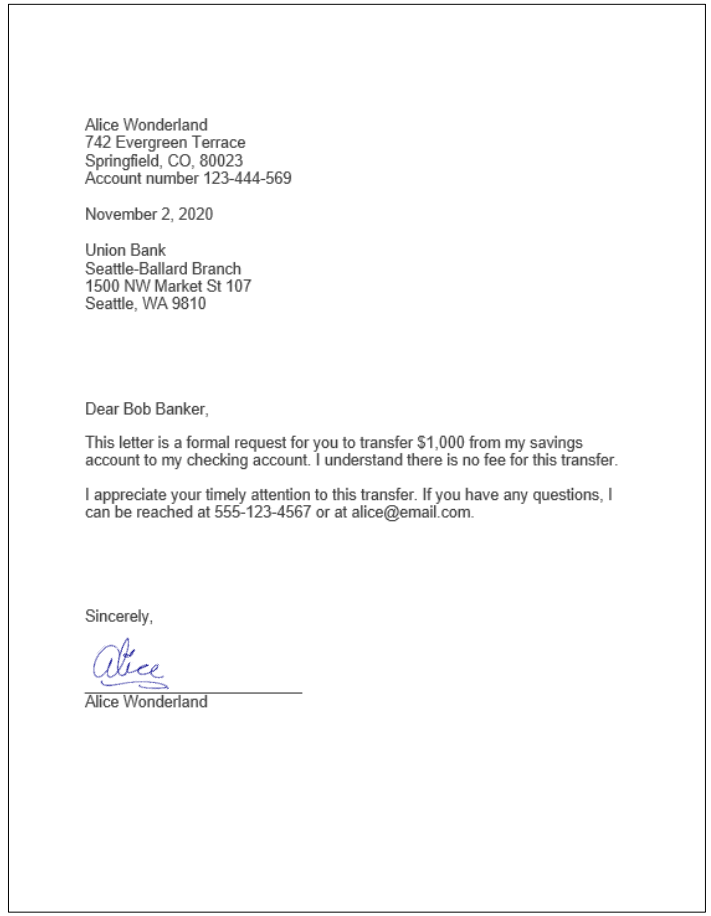




```
1101001010100001111111001010110101011101010010101010111010101100100100111000101011111100010100001011010111011010000101100001
```

scan





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 $\Sigma = (\mathrm{KeyGen}, \mathrm{Sign}, \mathrm{Vrfy})$

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

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

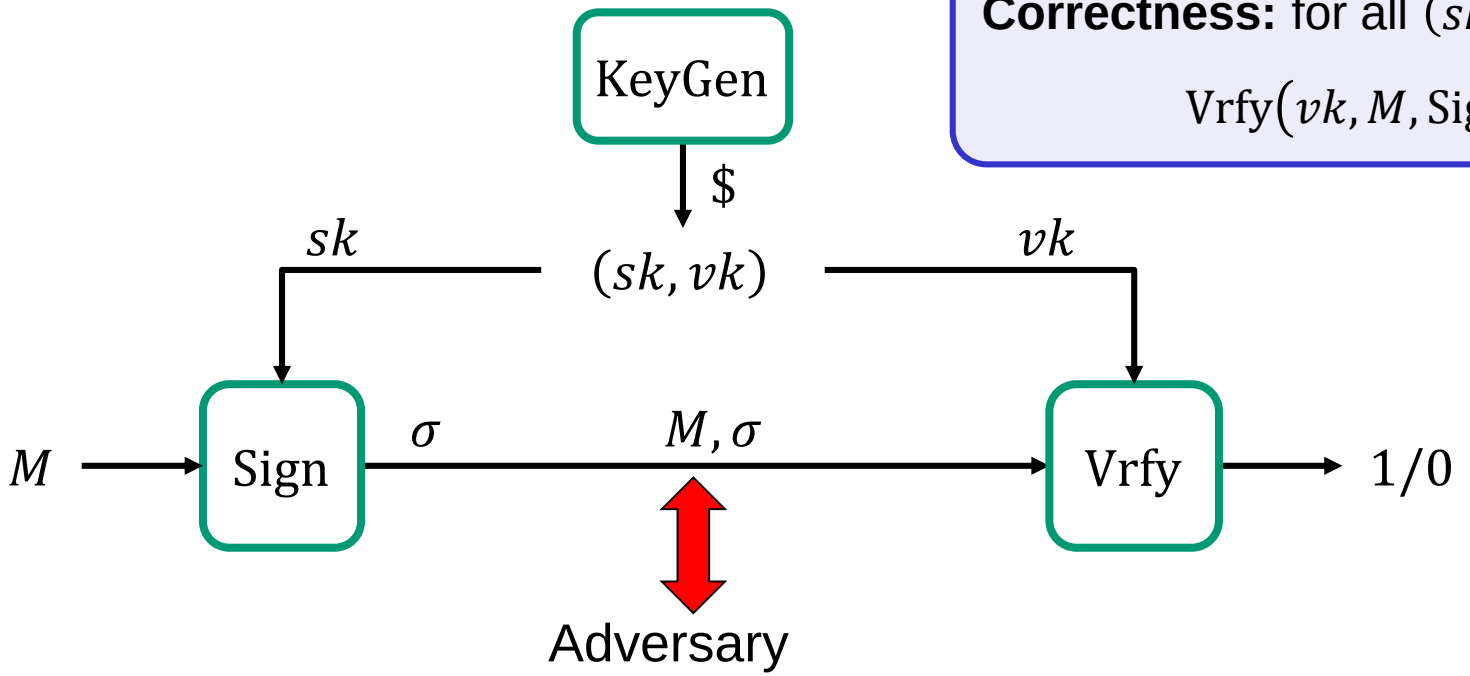
$\mathrm{Sign}(sk, M) = \mathrm{Sign}_{sk}(M) = \sigma$

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

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

Correctness: for all $(sk, pk) \leftarrow \mathrm{KeyGen}$:

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



9

Digital signatures – security: UF-CMA

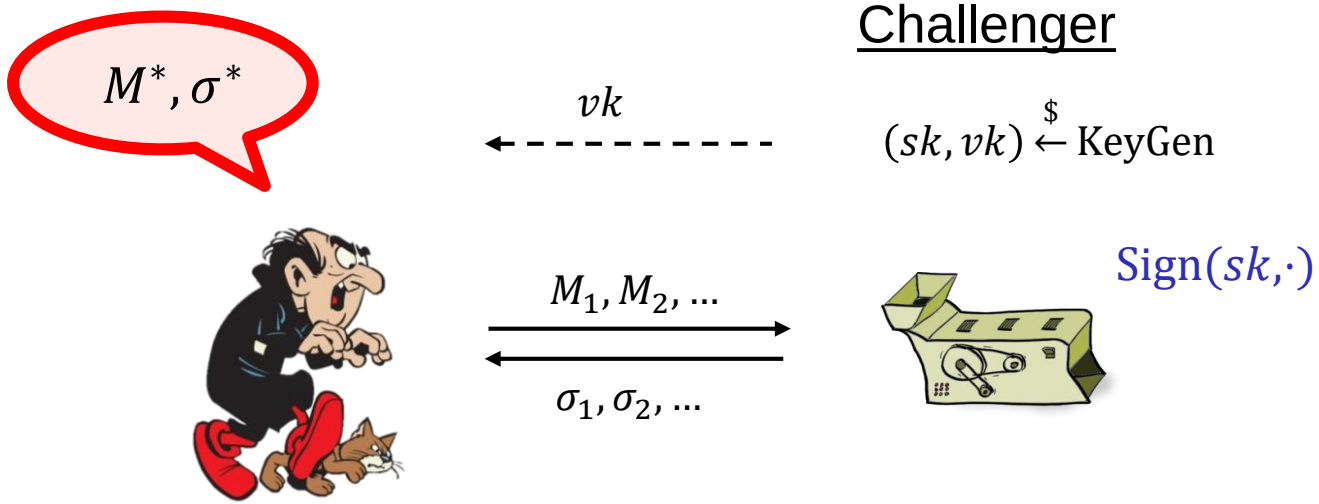
$\mathbf{Exp}_{\Sigma}^{\text{uf-cma}}(A)$

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

$\text{SIGN}_{sk}(M)$

---

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



If σ^* is a valid signature for M^* then the adversary has **forged** a signature

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

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

Textbook RSA signatures

$\mathcal{SK}$ $\mathcal{M}$ $\mathcal{S}$

RSA. Sign: $\underbrace{\boldsymbol{Z}^{+} \times \boldsymbol{Z}^{*}_{\phi(n)}}_{\mathcal{SK}} \times \boldsymbol{Z}^{*}_{n} \rightarrow \boldsymbol{Z}^{*}_{n}$

RSA. Vrfy: $\underbrace{\boldsymbol{Z}^{+} \times \boldsymbol{Z}^{*}_{\phi(n)}}_{\mathcal{PK}} \times \underbrace{\boldsymbol{Z}^{*}_{n}}_{\mathcal{M}} \times \underbrace{\boldsymbol{Z}^{*}_{n}}_{\mathcal{S}} \rightarrow \{1,0\}$

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. $sk \leftarrow (n, d)$ $\quad vk \leftarrow (n, e)$
7. **return** (sk, vk)

Sign$(sk = (n, d), M \in \boldsymbol{Z}^{*}_{n})$

1. $\sigma \leftarrow M^{d} \bmod n$
2. **return** σ

Vrfy$(vk = (n, e), M \in \boldsymbol{Z}^{*}_{n}, \sigma)$

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

vk

M, σ

Textbook RSA signatures: attacks

$A_1(n, e)$

1. Output (1,1)

$1^e \stackrel{?}{=} 1 \bmod n$ Yes! $\Rightarrow \mathbf{Adv}^{\text{uf-cma}}_{\text{RSA}}(A_1) = 1$

$A_2(n, e)$

1. Want to forge on message $M \in \boldsymbol{Z}_n^*$
2. Pick arbitrary $M_1 \in \boldsymbol{Z}_n^*$
3. $M_2 \leftarrow M \cdot M_1^{-1} \bmod n$
4. Query $\sigma_1 \leftarrow \text{SIGN}_{sk}(M_1)$ and $\sigma_2 \leftarrow \text{SIGN}_{sk}(M_2)$
5. Output $(M, \sigma_1 \cdot \sigma_2 \bmod n)$

$(\sigma_1 \cdot \sigma_2)^e \stackrel{?}{=} M \bmod n$ Yes! $\Rightarrow \mathbf{Adv}^{\text{uf-cma}}_{\text{RSA}}(A_2) = 1$

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

↑ RSA correctness

KeyGen

1. $p, q \stackrel{\$}{\leftarrow}$ 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. $sk \leftarrow (n, d)$ $vk \leftarrow (n, e)$
7. **return** (sk, vk)

Sign$(sk = (n, d), M \in \boldsymbol{Z}_n^*)$

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

Vrfy$(vk = (n, e), M \in \boldsymbol{Z}_n^*, \sigma)$

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

Textbook RSA signatures

$$\text{RSA.Sign: } \overbrace{\mathbb{Z}^+ \times \mathbb{Z}^*_{\phi(n)}}^{\mathcal{SK}} \times \mathbb{Z}^*_n \to \mathbb{Z}^*_n \quad (\mathcal{M} = \mathbb{Z}^*_n,\ \mathcal{S} = \mathbb{Z}^*_n)$$

$$\text{RSA.Vrfy: } \underbrace{\mathbb{Z}^+ \times \mathbb{Z}^*_{\phi(n)}}_{\mathcal{PK}} \times \mathbb{Z}^*_n \times \mathbb{Z}^*_n \to \{1,0\} \quad (\mathcal{M} = \mathbb{Z}^*_n,\ \mathcal{S} = \mathbb{Z}^*_n)$$

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. $sk \leftarrow (n, d)$ $\quad vk \leftarrow (n, e)$
7. **return** (sk, vk)

Sign$(sk = (n, d), M \in \mathbb{Z}^*_n)$

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

Vrfy$(vk = (n, e), M \in \mathbb{Z}^*_n, \sigma)$

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

vk ⟵ - - - - - - - - - - - - -

M, σ ⟵ ——————

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

Textbook RSA signatures

$$\text{RSA.Sign: } \overbrace{\boldsymbol{Z}^{+} \times \boldsymbol{Z}^{*}_{\phi(n)}}^{\mathcal{SK}} \times \underbrace{\boldsymbol{Z}^{*}_{n}}_{\mathcal{M}} \rightarrow \underbrace{\boldsymbol{Z}^{*}_{n}}_{\mathcal{S}}$$

$$\text{RSA.Vrfy: } \underbrace{\boldsymbol{Z}^{+} \times \boldsymbol{Z}^{*}_{\phi(n)}}_{\mathcal{PK}} \times \underbrace{\boldsymbol{Z}^{*}_{n}}_{\mathcal{M}} \times \underbrace{\boldsymbol{Z}^{*}_{n}}_{\mathcal{S}} \rightarrow \{1,0\}$$

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. $sk \leftarrow (n, d)$ $\quad vk \leftarrow (n, e)$
7. **return** (sk, vk)

Sign$(sk = (n, d), M \in \boldsymbol{Z}^{*}_{n})$

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

vk

M, σ

Vrfy$(vk = (n, e), M \in \boldsymbol{Z}^{*}_{n}, \sigma)$

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

$$H : \{0,1\}^{*} \rightarrow \boldsymbol{Z}^{*}_{n}$$

Hash-then sign paradigm

$$\text{RSA.Sign: } \overbrace{\boldsymbol{Z}^{+} \times \boldsymbol{Z}^{*}_{\phi(n)}}^{\mathcal{SK}} \times \{0,1\}^{*} \to \boldsymbol{Z}^{*}_{n} \qquad \mathcal{M} \quad \mathcal{S}$$

$$\text{RSA.Vrfy: } \underbrace{\boldsymbol{Z}^{+} \times \boldsymbol{Z}^{*}_{\phi(n)}}_{\mathcal{PK}} \times \{0,1\}^{*} \times \boldsymbol{Z}^{*}_{n} \to \{1,0\} \qquad \mathcal{M} \quad \mathcal{S}$$

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. $sk \leftarrow (n, d)$ $\quad vk \leftarrow (n, e)$
7. **return** (sk, vk)

Sign$(sk = (n, d), M \in \boldsymbol{Z}^{*}_{n})$

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

Vrfy$(vk = (n, e), M \in \boldsymbol{Z}^{*}_{n}, \sigma)$

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

vk

M, σ

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

Hashed RSA signatures

$A_1(n, e)$

1. Output (1,1)

$1^e \stackrel{?}{=} H(1) \bmod n$ No! $\Rightarrow \mathbf{Adv}^{\text{uf-cma}}_{\text{RSA}}(A_1) \approx 0$

$A_2(n, e)$

1. Want to forge on message $M \in \{0,1\}^*$
2. Find $M_1, M_2 \in \{0,1\}^*$ such that $H(M) = H(M_1) \cdot H(M_2) \bmod n$
3. Query $\sigma_1 \leftarrow \text{SIGN}_{sk}(M_1)$ and $\sigma_2 \leftarrow \text{SIGN}_{sk}(M_2)$
4. Output $(M, \sigma_1 \cdot \sigma_2 \bmod n)$

Hard to find!

$(\sigma_1 \cdot \sigma_2)^e \stackrel{?}{=} H(M) \bmod n$ No! $\Rightarrow \mathbf{Adv}^{\text{uf-cma}}_{\text{RSA}}(A_2) \approx 0$

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

KeyGen

1. $p, q \stackrel{\$}{\leftarrow}$ 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. $sk \leftarrow (n, d)$ $vk \leftarrow (n, e)$
7. **return** (sk, vk)

Sign$(sk = (n, d), M \in \boldsymbol{Z}_n^*)$

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

Vrfy$(vk = (n, e), M \in \boldsymbol{Z}_n^*, \sigma)$

1. **if** $\sigma^e = H(M) \bmod n$ **then**
2. **return** 1
3. **else**
4. **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 $\mathrm{SIGN}_{sk}(\cdot)$ queries, there is an algorithm B solving the RSA-problem:

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

where H is assumed perfect

*Perfect = random oracle

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. $sk \leftarrow (n, d)$ $\quad$ $vk \leftarrow (n, e)$
7. **return** (sk, vk)

Sign$(sk = (n, d), M \in \boldsymbol{Z}_n^*)$

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

Vrfy$(vk = (n, e), M \in \boldsymbol{Z}_n^*, \sigma)$

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

Discrete logarithm based signatures

Discrete-log-based signatures: Schnorr

$$G = \langle g \rangle$$

$$H : G \times \{0,1\}^* \to \boldsymbol{Z}_p^*$$

KeyGen

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

vk

M, σ

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

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

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

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

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

Discrete-log-based signatures: Schnorr – security

$$G = \langle g \rangle$$

$$H : G \times \{0,1\}^* \to \boldsymbol{Z}_p^*$$

KeyGen

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

vk

M, σ

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

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

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.
- r must be picked new *every time*!
- Attacker must essentially solve

$$g^r = g^s B^h \iff r = s + bh \iff s = r - bh$$

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

$$\Rightarrow \quad s - s' = (r - bh) - (r - bh') = (h' - h) \cdot b$$

$$\Rightarrow b = (s - s') \cdot (h' - h)^{-1} \bmod p$$

Discrete-log-based signatures: (EC)DSA

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

- (EC)DSA
  - Non-patented alternative
  - Derived from ElGamal-based signature scheme
  - More complicated design than Schnorr
  - No security proof
  - Standardized by NIST
  - Very widely used
  - Same r-reuse problem as Schnorr: leaks long-term signing key
    - Broke all Playstations 3's produced by Sony

Public-key infrastructure (PKI)

What are identities?

"Alice", g^a

"Bob", g^b

$K \leftarrow g^{ab}$

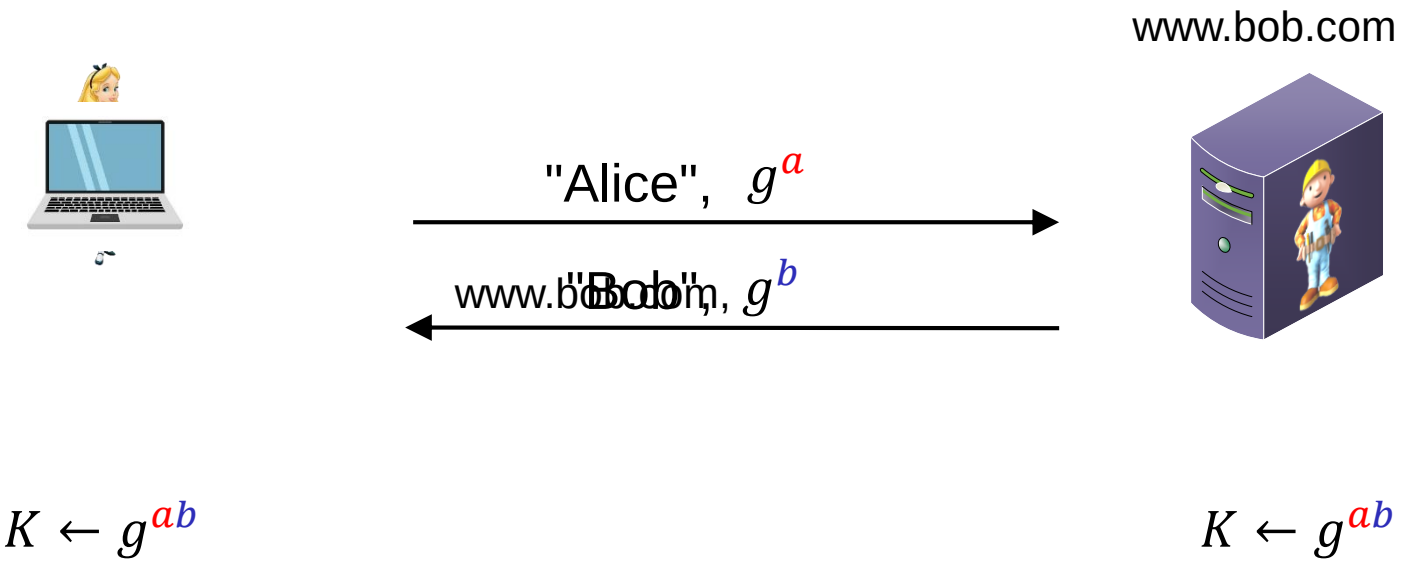
$K \leftarrow g^{ab}$

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

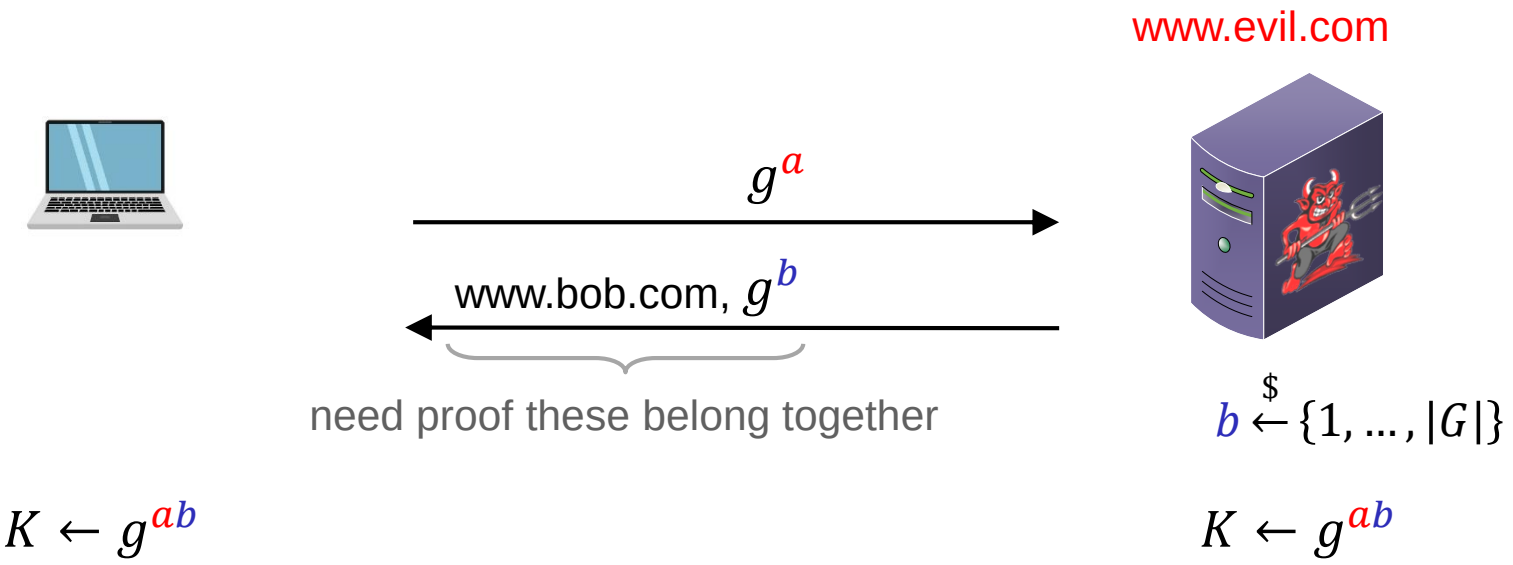


There are many Alice's and many Bob's

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 – on the internet: bind public keys to **domain names**

Identities on the internet

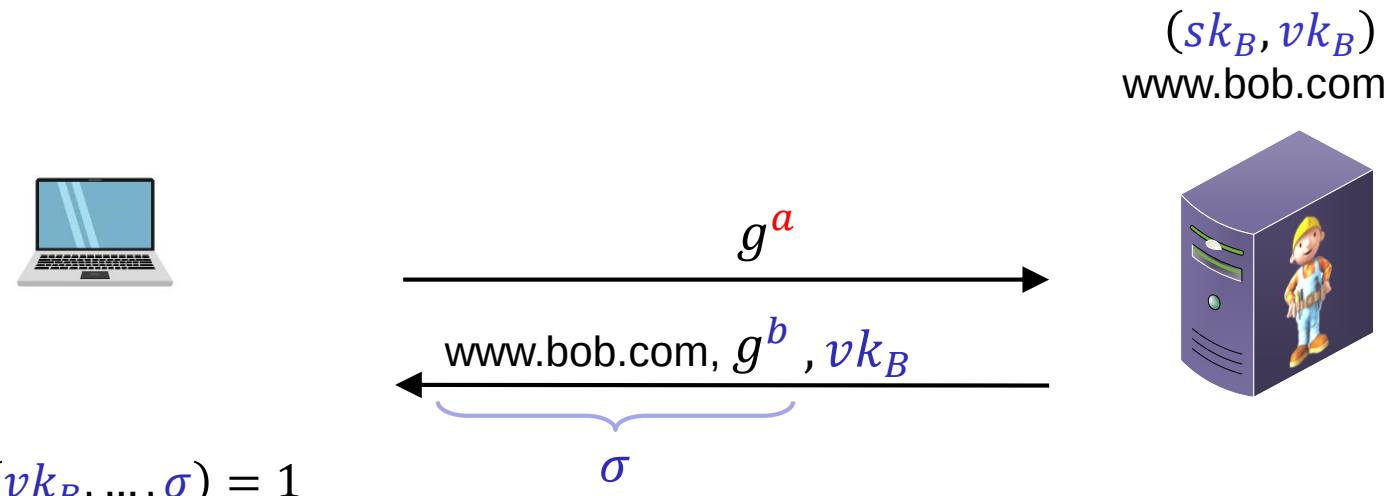


There are many Alice's and many Bob's

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 – on the internet: bind public keys to **domain names**

Authenticated key exchange



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

$K \leftarrow g^{ab}$

else

abort

$K \leftarrow g^{ab}$

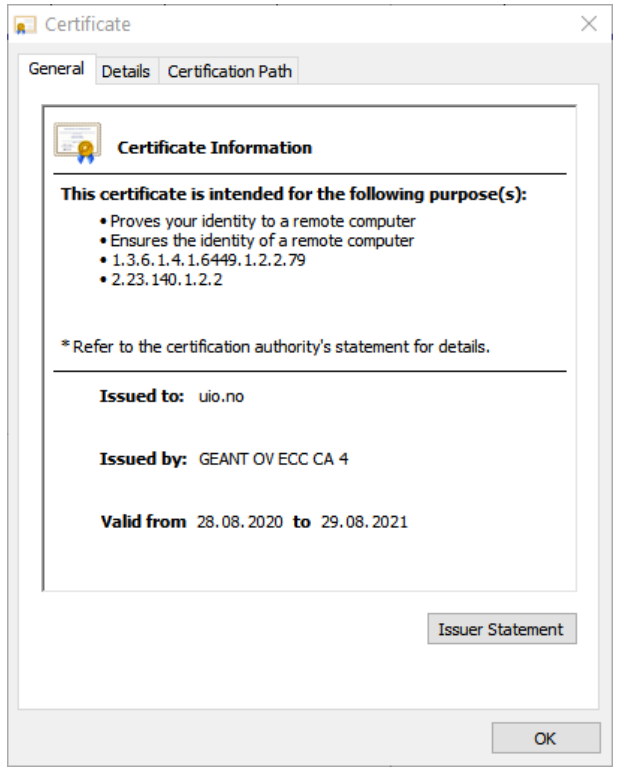
But why should we trust this vk_B? Could have been created by the adversary itself

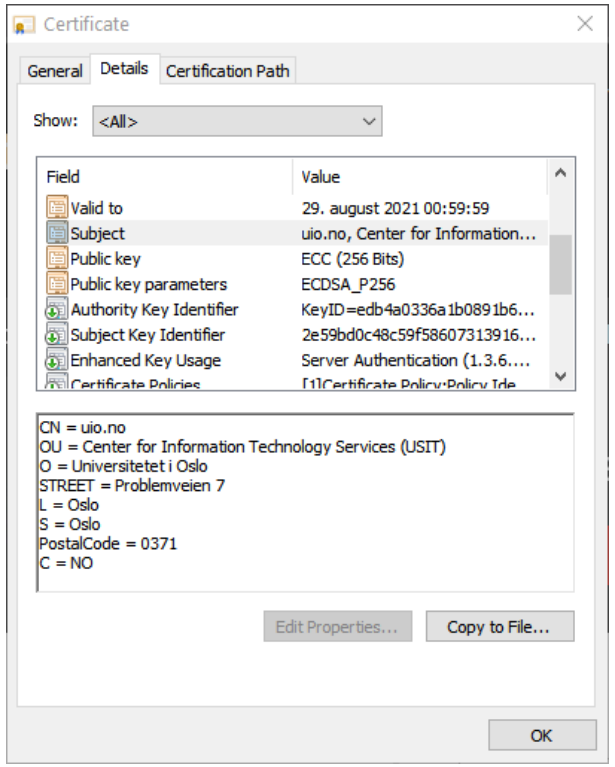
Digital certificates

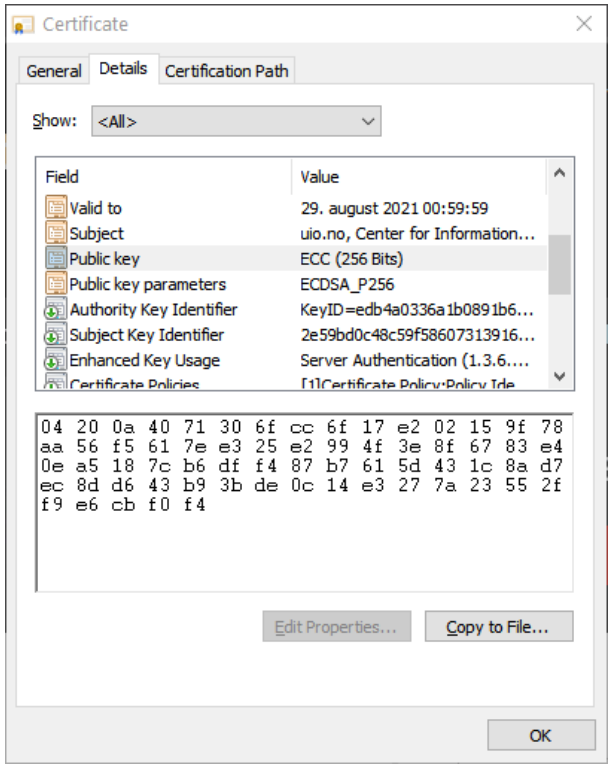
- **Digital certificate:** a way of binding a public key to an entity

- 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







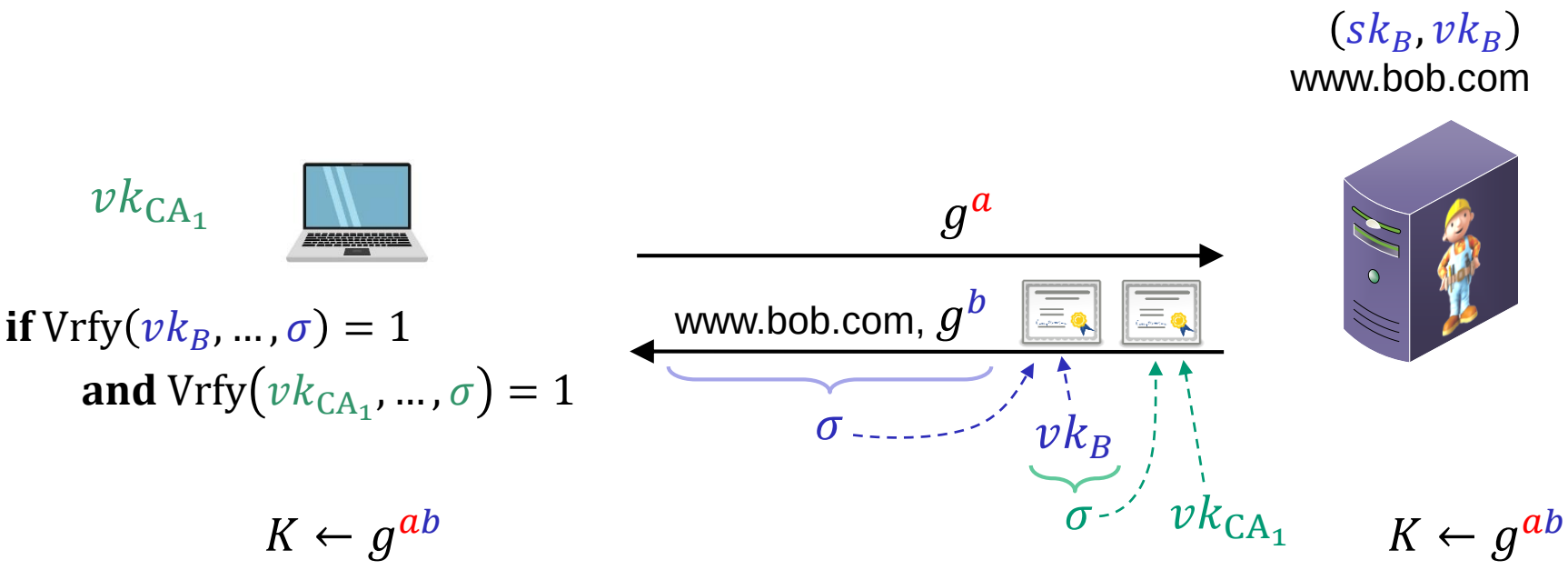
Certificate authorities (CA)

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

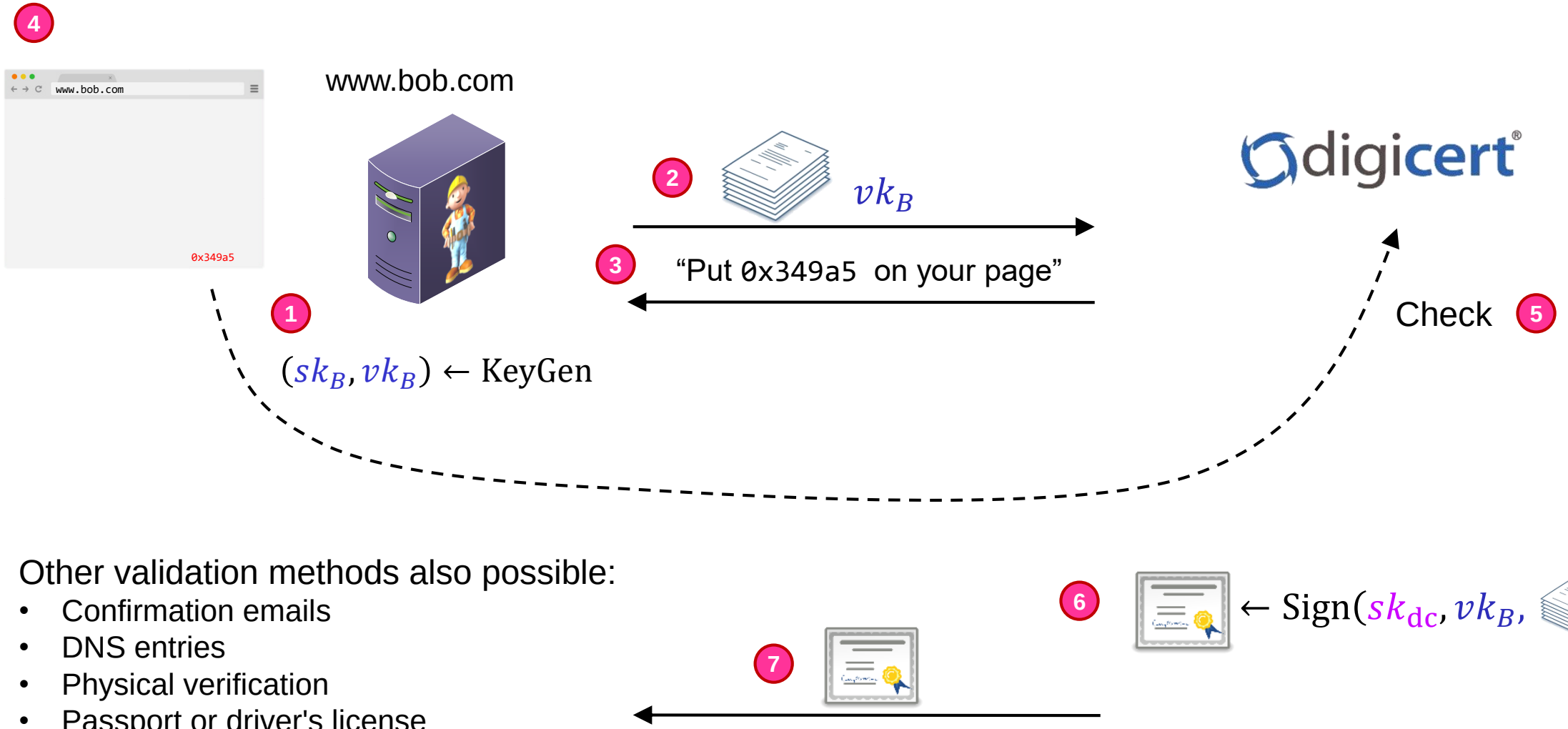
- Acts as a trusted third-party, certifying (i.e., signing) the public keys of other entities
  - Verifies the identity of a claimed public-key owner

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

HTTPS / TLS + PKI



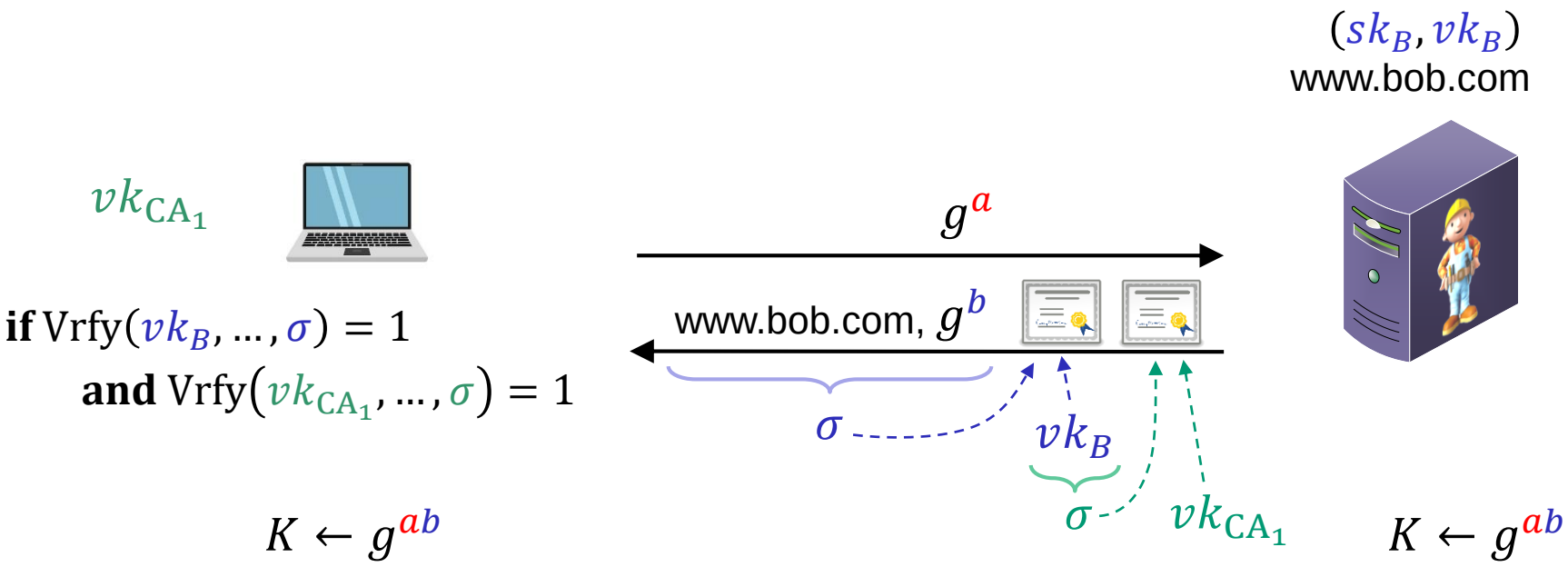
How to get a signed certificate?



Other validation methods also possible:

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

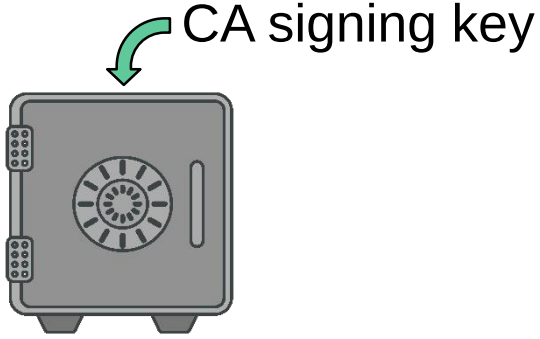
HTTPS / TLS + PKI



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

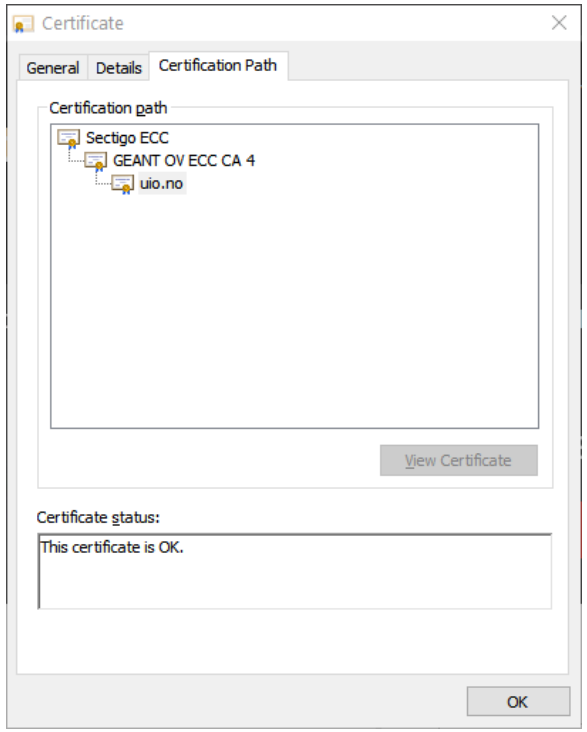
IdenTrust part of HID Global

digicert

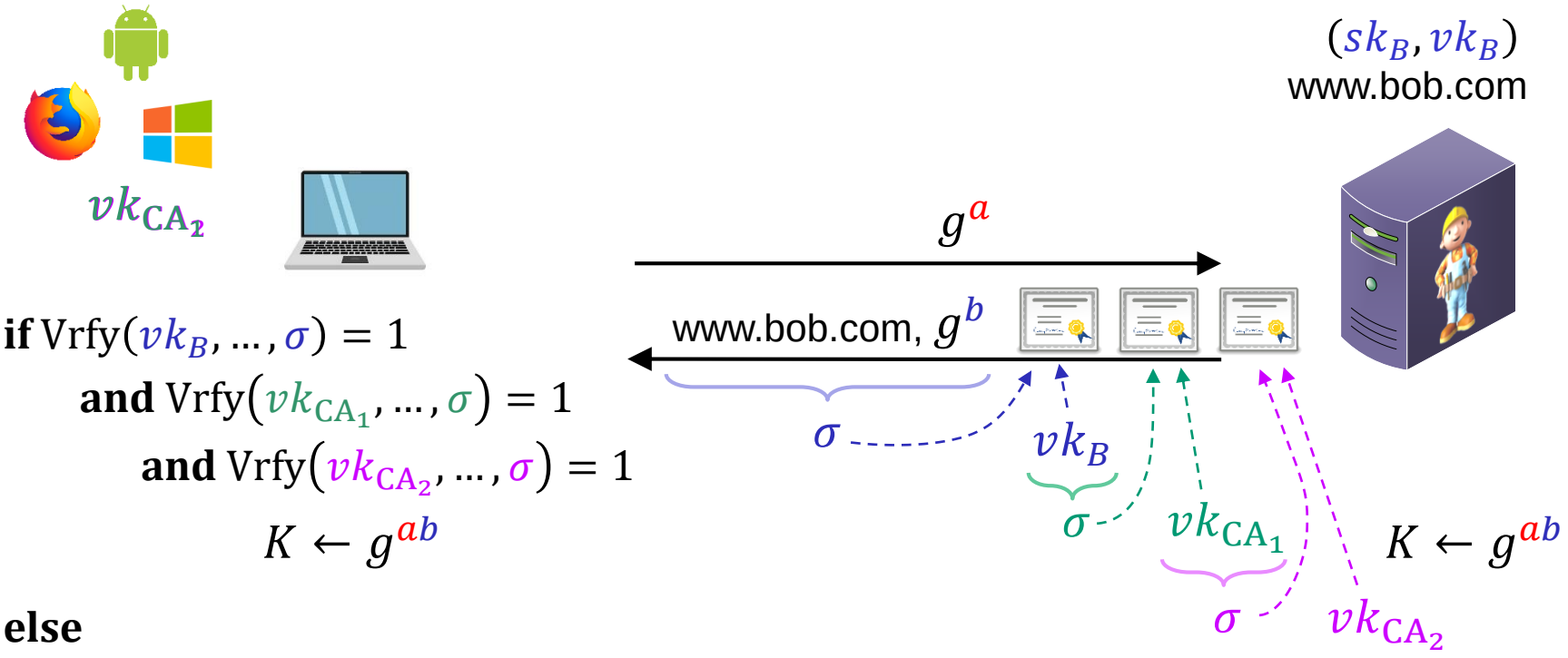
GoDaddy

Let's Encrypt

Certificate chains



HTTPS / TLS + PKI

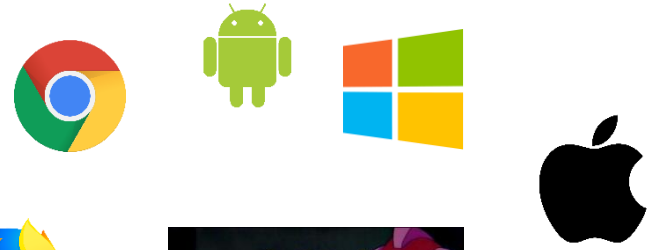


if $\text{Vrfy}(vk_B, \ldots, \sigma) = 1$
  and $\text{Vrfy}(vk_{CA_1}, \ldots, \sigma) = 1$
    and $\text{Vrfy}(vk_{CA_2}, \ldots, \sigma) = 1$
      $K \leftarrow g^{ab}$

else
  abort

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



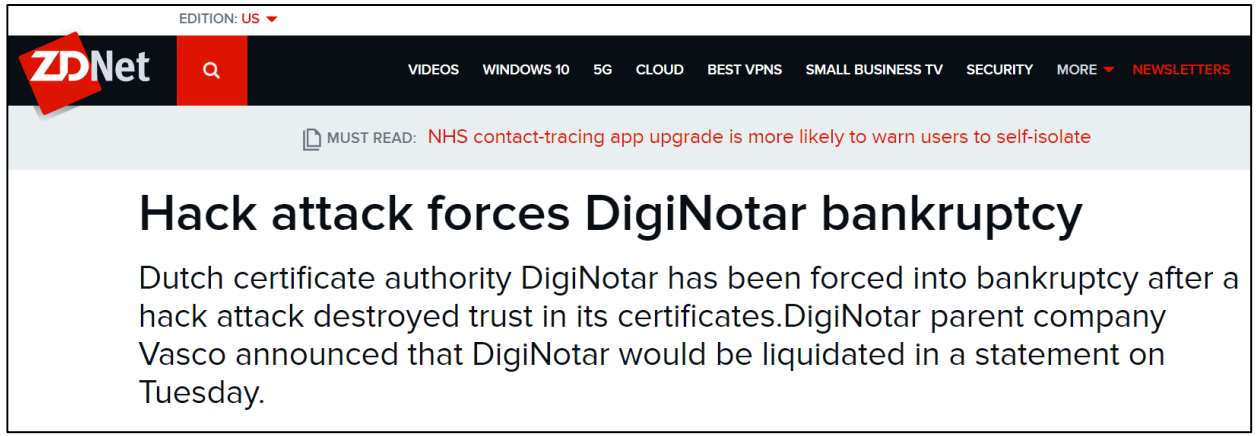
- Lot's of auditing and paperwork

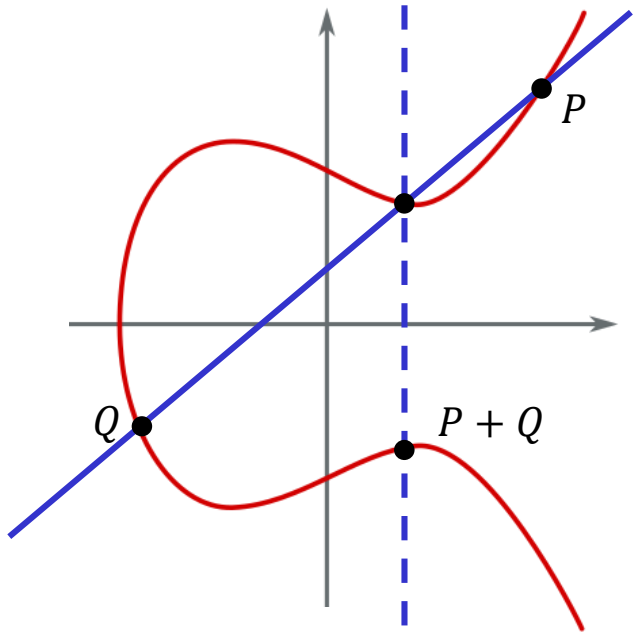
- Many formal technical and non-technical security requirements
  - CA/Browser forum
  - Baseline Requirements v1.7.3

Let's Encrypt

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





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

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

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 function/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 Padded RSA Fujisaki-Okomoto-transform
Digital signatures	Produce 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