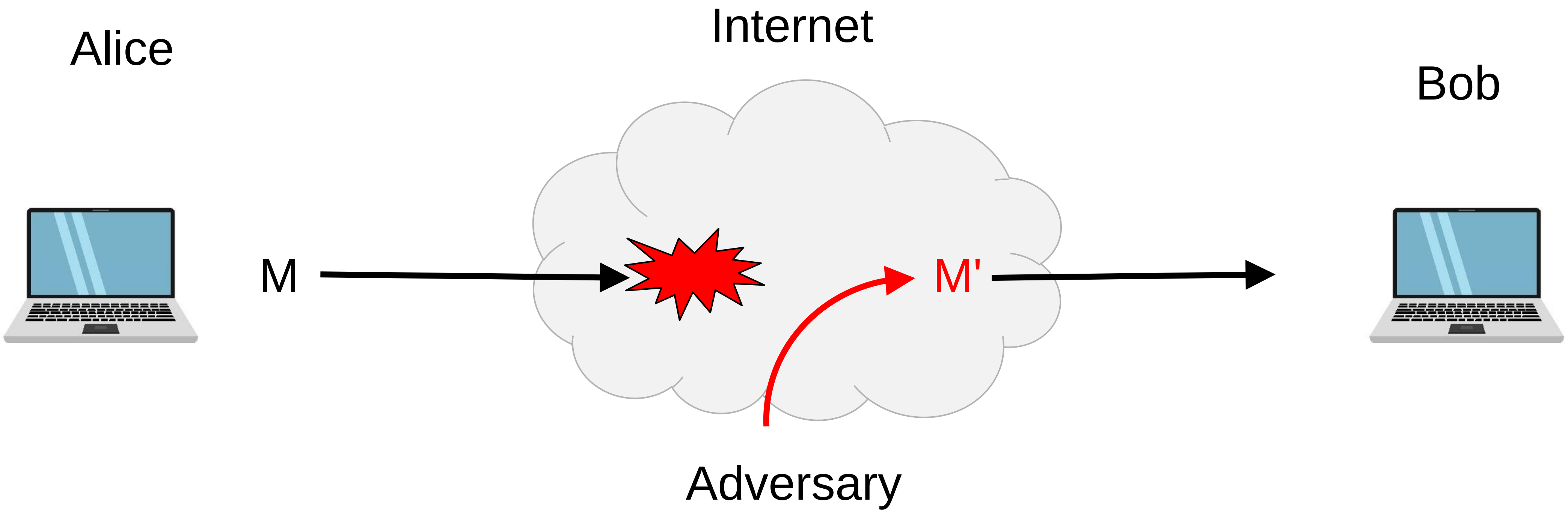
Digital signatures, UF-CMA, RSA, Schnorr, PKI

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)

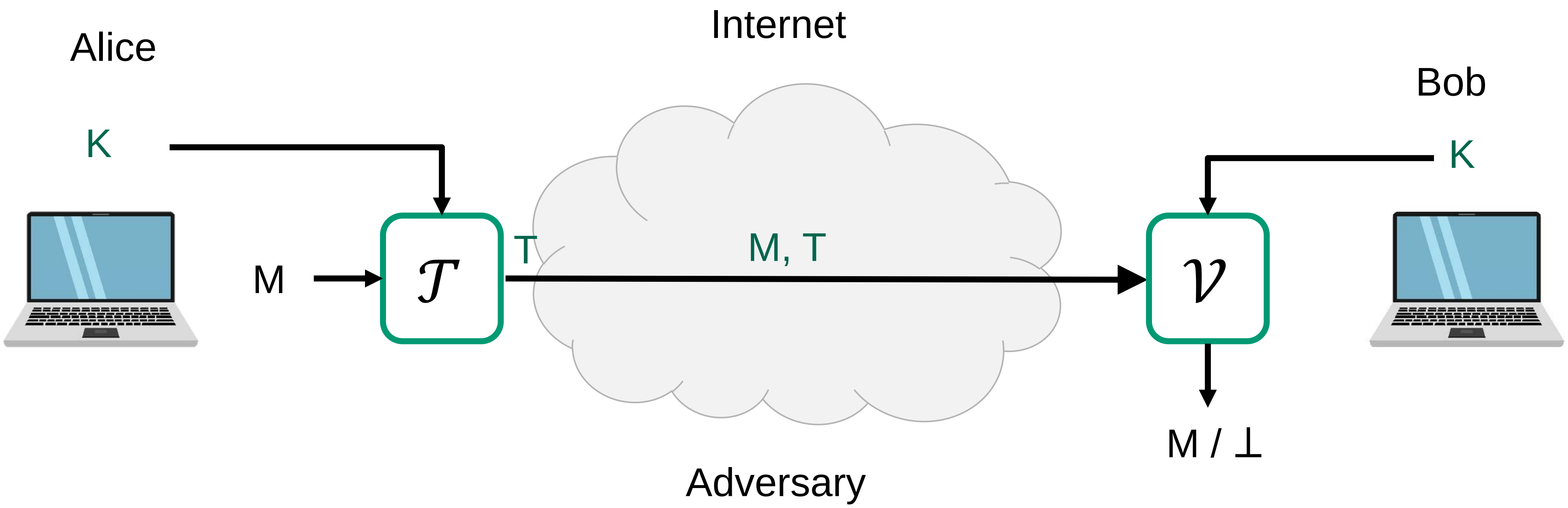
What is cryptography?



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

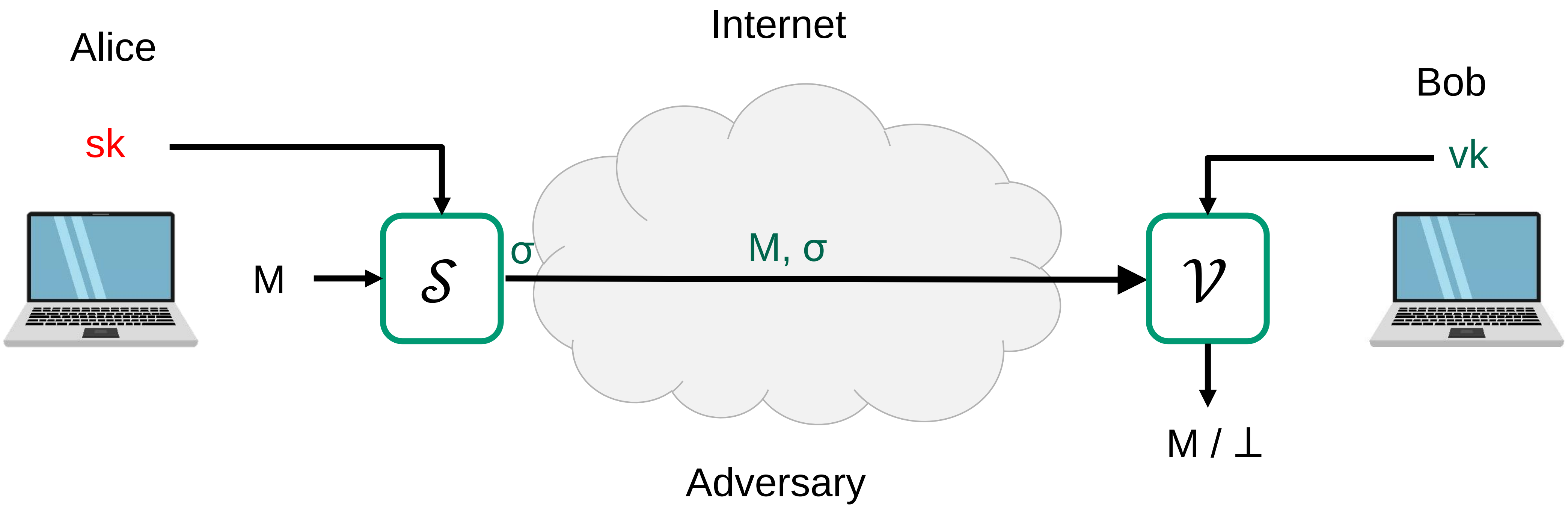


$\mathcal{T}$: tagging algorithm (public)

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

K : tagging / verification key (secret)

Achieving integrity: digital signatures



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

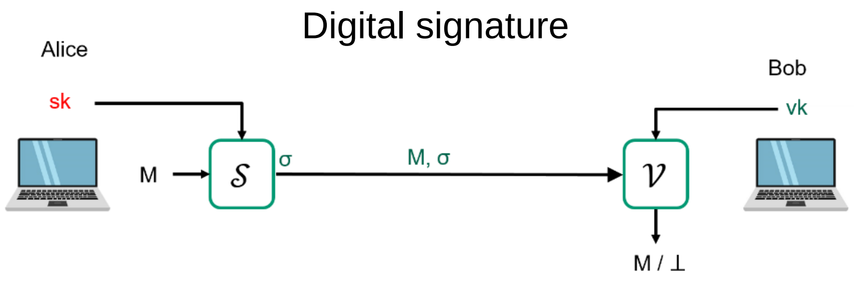
$\mathcal{V}$: verification algorithm (public)

sk : signing key (secret)

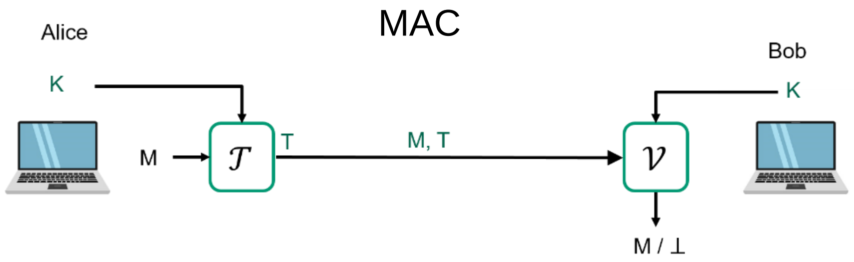
vk : verification key (public)

Digital signatures vs. MACs

- Digital signatures can be verified by *anyone*



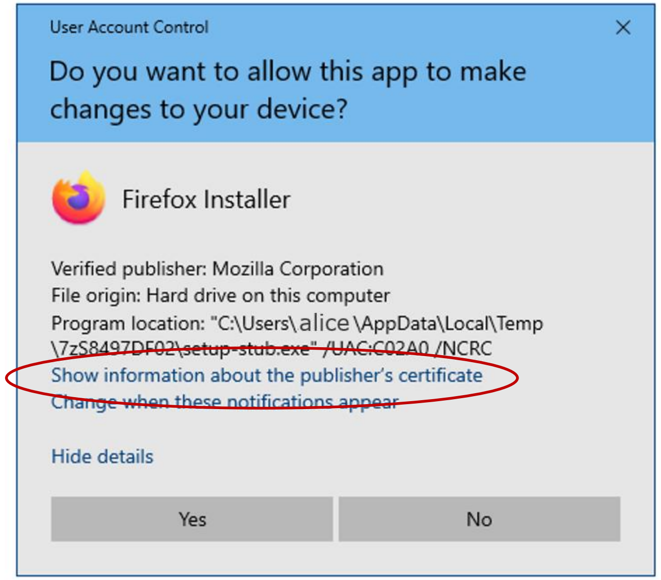
- MACs can only be verified by party sharing the same key



- **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



User Account Control

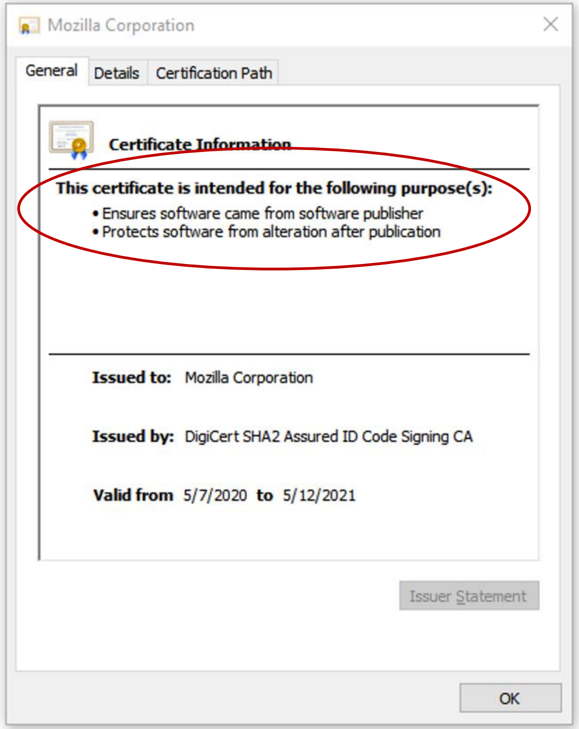
Do you want to allow this app to make changes to your device?

Firefox Installer

Verified publisher: Mozilla Corporation
File origin: Hard drive on this computer
Program location: "C:\Users\alice\AppData\Local\Temp\7zS8497DF02\setup-stub.exe" /UAC:C02A0 /NCRC

Show information about the publisher's certificate
Change when these notifications appear

Hide details



Mozilla Corporation

Certificate Information

This certificate is intended for the following purpose(s):

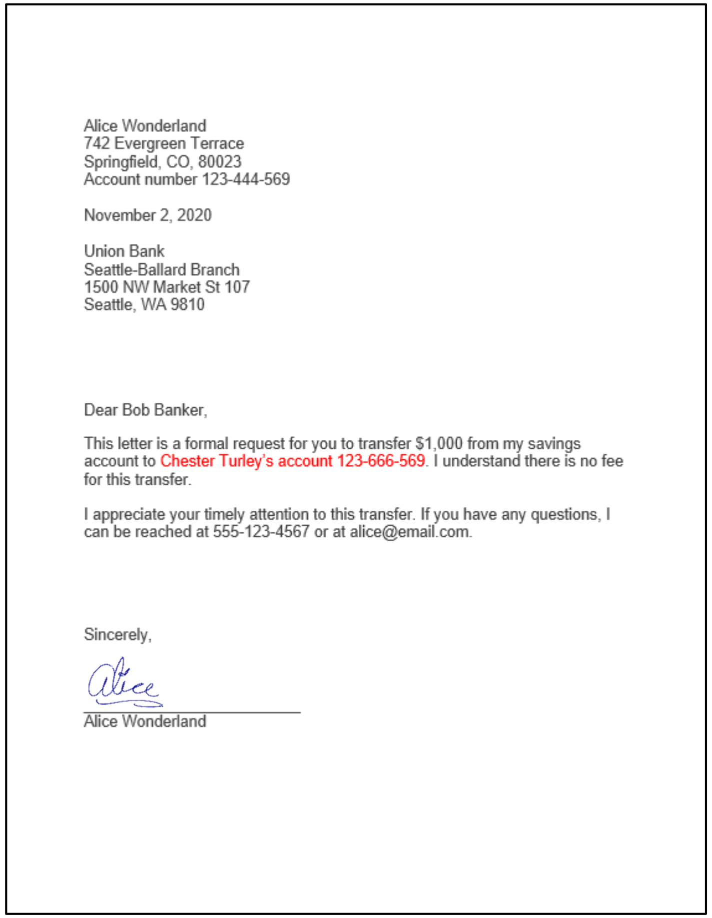
- Ensures software came from software publisher
- Protects software from alteration after publication

Issued to: Mozilla Corporation

Issued by: DigiCert SHA2 Assured ID Code Signing CA

Valid from 5/7/2020 **to** 5/12/2021

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,

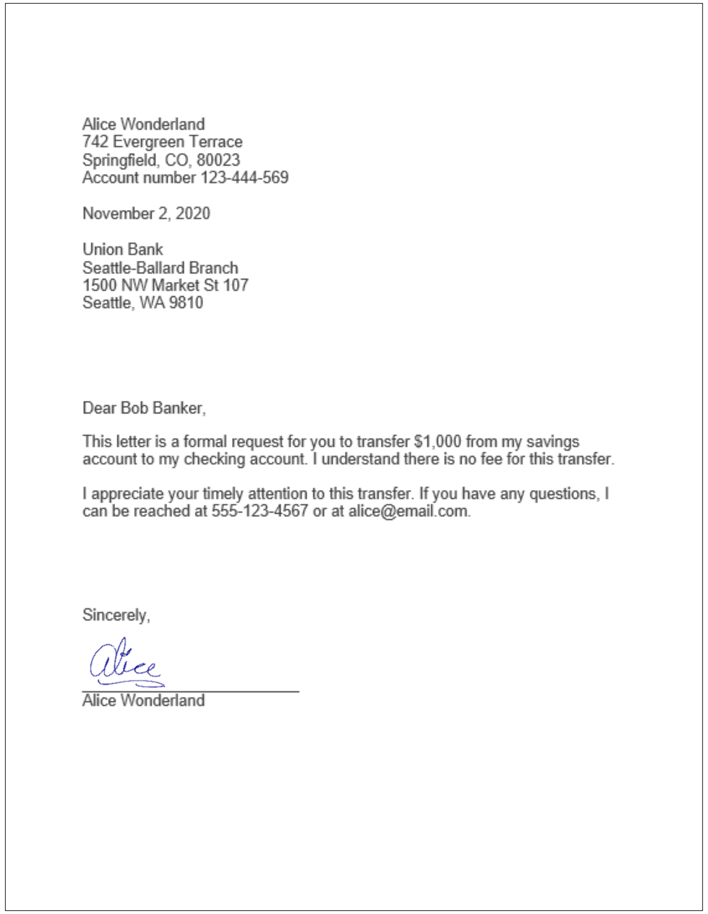
Alice

Alice Wonderland

```
110100101010000
111111001010110
101011101010010
101010111010101
110010101001001
110001010111111
001010000010110
100101111101101
100000101100001

110100101010000
111111001010110
101011101010010
100011111011101
110010101001001
110001010111111
001010000010110
100101111101101
100000101100001
```

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

Alice Wonderland

Digital signatures – syntax

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

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

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

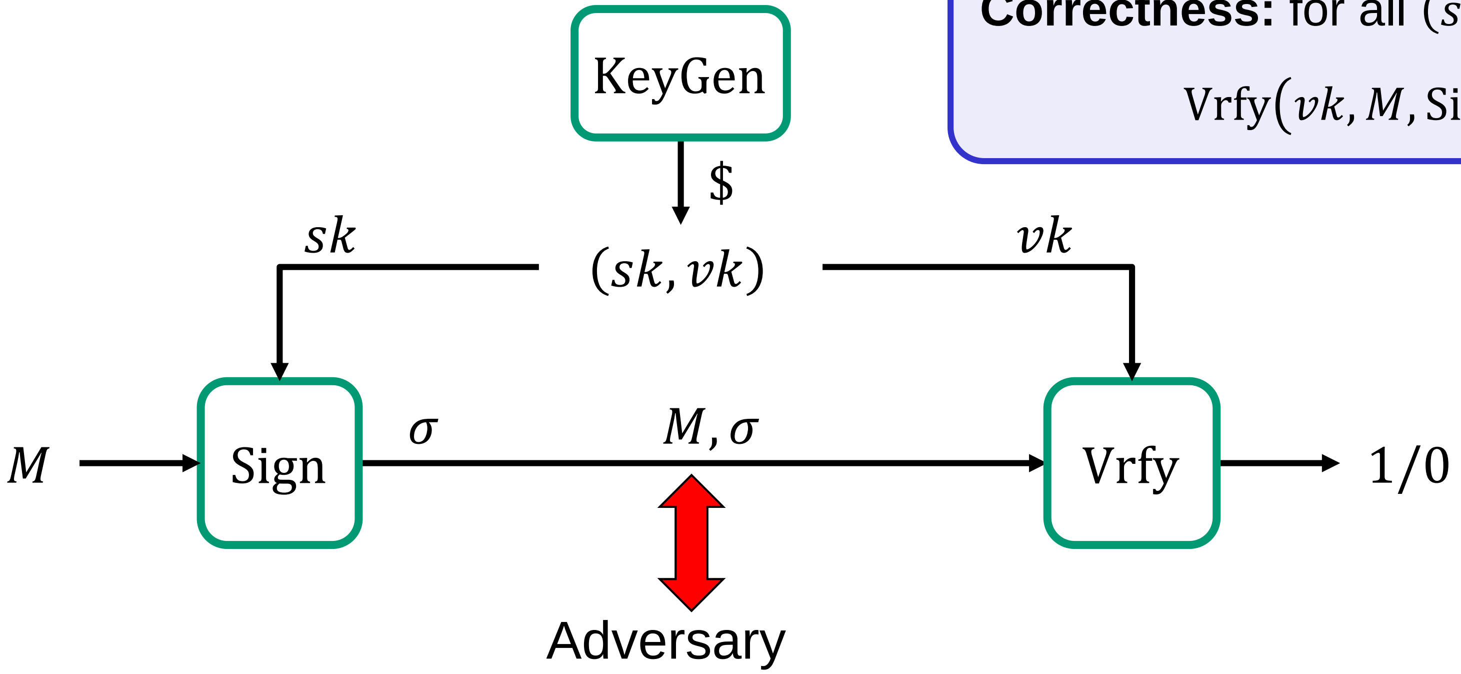
$\mathrm{Sign}(sk, M) = \mathrm{Sign}_{sk}(M) = \sigma$

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

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

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

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



KeyGen

\$

(sk, vk)

sk

vk

M

Sign

σ

M, σ

Vrfy

$1/0$

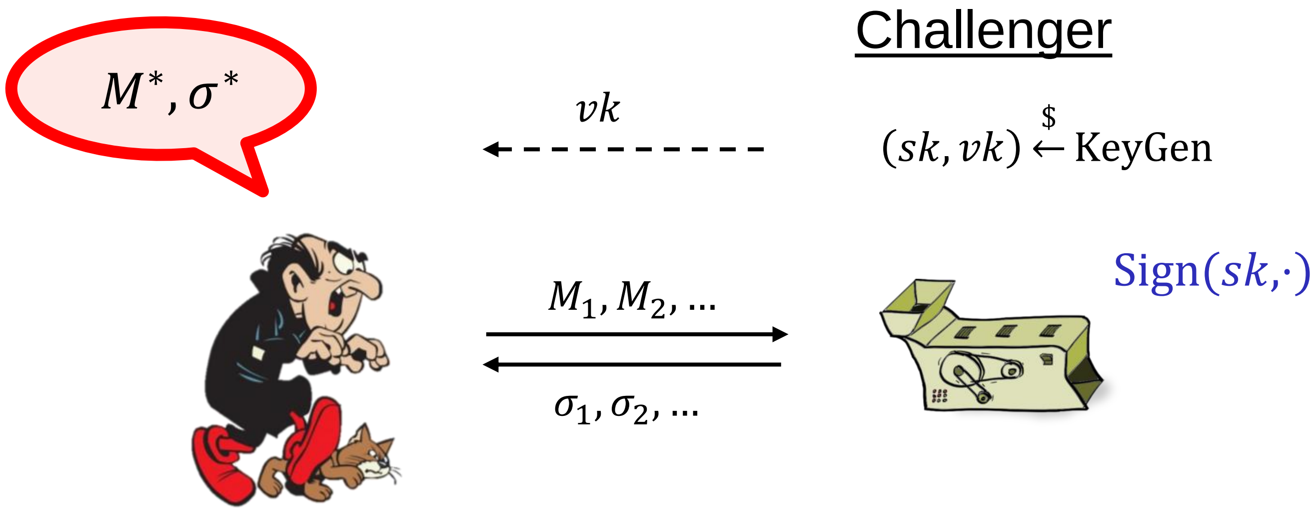
Adversary

Digital signatures – security: UF-CMA

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

$(sk, vk) \overset{\$}{\leftarrow} \Sigma.\mathrm{KeyGen}$
$S \leftarrow [\,]$
$(M^*, \sigma^*) \leftarrow A^{\mathrm{SIGN}_{sk}(\cdot)}(vk)$
if $\Sigma.\mathrm{Vrfy}(vk, M^*, \sigma^*) = 1$ and $M \notin S$ then
    return 1
else
    return 0

$\mathrm{SIGN}_{sk}(M)$
--------------------------------
$\sigma \leftarrow \Sigma.\mathrm{Sign}(sk, M)$
$S.\mathrm{add}(M)$
return $\sigma$
```



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

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

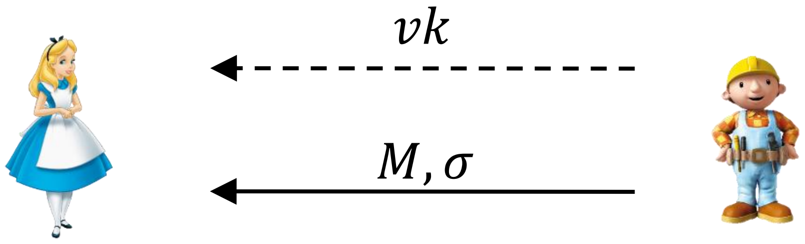
Textbook RSA signatures

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

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

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

```
if $\sigma^e = M \bmod n$ then
    return 1
else
    return 0
```



KeyGen

```
$p, q \overset{\$}{\leftarrow}$ two random prime numbers
$n \leftarrow p \cdot q$
$\phi(n) = (p-1)(q-1)$
choose $e$ such that $\gcd(e, \phi(n)) = 1$
$d \leftarrow e^{-1} \bmod \phi(n)$
$sk \leftarrow (n, d)$    $vk \leftarrow (n, e)$
return $(sk, vk)$
```

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

```
$\sigma \leftarrow M^d \bmod n$
return $\sigma$
```

Textbook RSA signatures: attacks

```
$A_1(n, e)$
    Output $(1,1)$
```

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

```
$A_2(n, e)$
    Want to forge on message $M \in \boldsymbol{Z}_n^*$
    Pick arbitrary $M_1 \in \boldsymbol{Z}_n^*$
    $M_2 \leftarrow M \cdot M_1^{-1} \bmod n$
    Query $\sigma_1 \leftarrow \text{SIGN}_{sk}(M_1)$ and $\sigma_2 \leftarrow \text{SIGN}_{sk}(M_2)$
    Output $(M, \sigma_1 \cdot \sigma_2 \bmod n)$
```

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

$$\left(M_1^d \cdot M_2^d\right)^e = M_1^{ed} \cdot M_2^{ed} \underset{\substack{\uparrow \\ \text{RSA correctness}}}{=} M_1 \cdot M_2 = M_1 \cdot M \cdot M_1^{-1} = M \bmod n$$

```
KeyGen
    $p, q \xleftarrow{\$}$ two random prime numbers
    $n \leftarrow p \cdot q$
    $\phi(n) = (p-1)(q-1)$
    choose $e$ such that $\gcd(e, \phi(n)) = 1$
    $d \leftarrow e^{-1} \bmod \phi(n)$
    $sk \leftarrow (n, d)$      $vk \leftarrow (n, e)$
    return $(sk, vk)$
```

```
Sign$(sk = (n, d), M \in \boldsymbol{Z}_n^*)$
    $\sigma \leftarrow M^d \bmod n$
    return $\sigma$
```

```
Vrfy$(vk = (n, e), M \in \boldsymbol{Z}_n^*, \sigma)$
    if $\sigma^e = M \bmod n$ then
        return 1
    else
        return 0
```

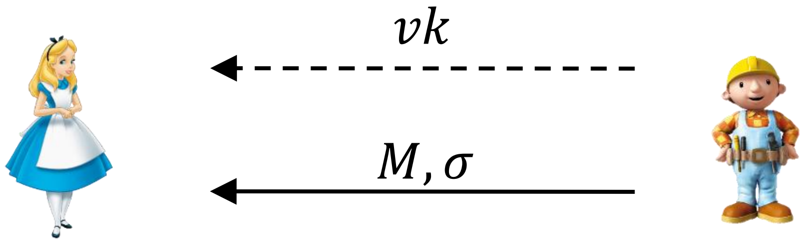
Textbook RSA signatures

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

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

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

```
if $\sigma^e = M \bmod n$ then
    return 1
else
    return 0
```



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

KeyGen

```
$p, q \overset{\$}{\leftarrow}$ two random prime numbers
$n \leftarrow p \cdot q$
$\phi(n) = (p-1)(q-1)$
choose $e$ such that $\gcd(e, \phi(n)) = 1$
$d \leftarrow e^{-1} \bmod \phi(n)$
$sk \leftarrow (n, d)$      $vk \leftarrow (n, e)$
return $(sk, vk)$
```

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

```
$\sigma \leftarrow M^d \bmod n$
return $\sigma$
```

Textbook RSA signatures

$$\mathrm{RSA.Sign}\colon\ \overbrace{\boldsymbol{Z}^{+} \times \boldsymbol{Z}^{*}_{\phi(n)}}^{\mathcal{SK}} \times \overset{\mathcal{M}}{\boldsymbol{Z}^{*}_{n}} \to \overset{\mathcal{S}}{\boldsymbol{Z}^{*}_{n}}$$

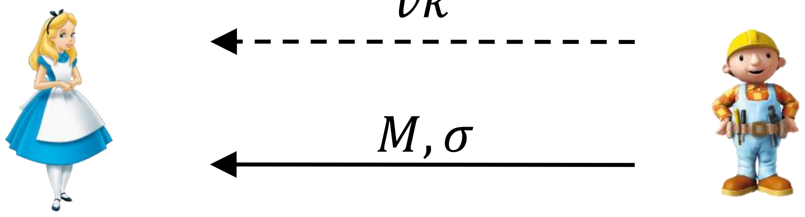
$$\mathrm{RSA.Vrfy}\colon\ \underbrace{\boldsymbol{Z}^{+} \times \boldsymbol{Z}^{*}_{\phi(n)}}_{\mathcal{PK}} \times \underset{\mathcal{M}}{\boldsymbol{Z}^{*}_{n}} \times \underset{\mathcal{S}}{\boldsymbol{Z}^{*}_{n}} \to \{1,0\}$$

KeyGen

```
$p, q \xleftarrow{\$}$ two random prime numbers
$n \leftarrow p \cdot q$
$\phi(n) = (p-1)(q-1)$
choose $e$ such that $\gcd(e, \phi(n)) = 1$
$d \leftarrow e^{-1} \bmod \phi(n)$
$sk \leftarrow (n, d)$      $vk \leftarrow (n, e)$
return $(sk, vk)$
```

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

```
if $\sigma^{e} = H(M) \bmod n$ then
    return 1
else
    return 0
```



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

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

```
$\sigma \leftarrow H(M)^{d} \bmod n$
return $\sigma$
```

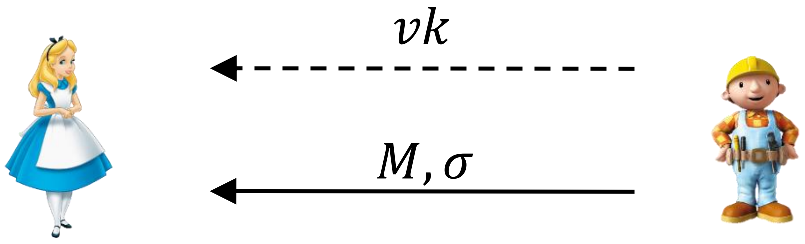
Hash-then sign paradigm

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

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

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

```
if $\sigma^{e} = H(M) \bmod n$ then
    return 1
else
    return 0
```



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

KeyGen

```
$p, q \xleftarrow{\$}$ two random prime numbers
$n \leftarrow p \cdot q$
$\phi(n) = (p-1)(q-1)$
choose $e$ such that $\gcd(e, \phi(n)) = 1$
$d \leftarrow e^{-1} \bmod \phi(n)$
$sk \leftarrow (n, d)$      $vk \leftarrow (n, e)$
return $(sk, vk)$
```

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

```
$\sigma \leftarrow H(M)^{d} \bmod n$
return $\sigma$
```

Hashed RSA signatures

$A_1(n, e)$

```
Output $(1,1)$
```

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

$A_2(n, e)$

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

Hard to find!

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

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

KeyGen

```
$p, q \overset{\$}{\leftarrow}$ two random prime numbers
$n \leftarrow p \cdot q$
$\phi(n) = (p-1)(q-1)$
choose $e$ such that $\gcd(e, \phi(n)) = 1$
$d \leftarrow e^{-1} \bmod \phi(n)$
$sk \leftarrow (n, d)$      $vk \leftarrow (n, e)$
return $(sk, vk)$
```

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

```
$\sigma \leftarrow H(M)^d \bmod n$
return $\sigma$
```

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

```
if $\sigma^e = H(M) \bmod n$ then
    return 1
else
    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

```
p, q $\overset{\$}{\leftarrow}$ two random prime numbers
$n \leftarrow p \cdot q$
$\phi(n) = (p-1)(q-1)$
choose $e$ such that $\gcd(e, \phi(n)) = 1$
$d \leftarrow e^{-1} \bmod \phi(n)$
$sk \leftarrow (n, d)$      $vk \leftarrow (n, e)$
return $(sk, vk)$
```

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

```
$\sigma \leftarrow H(M)^d \bmod n$
return $\sigma$
```

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

```
if $\sigma^e = H(M) \bmod n$ then
    return 1
else
    return 0
```

Discrete logarithm based signatures

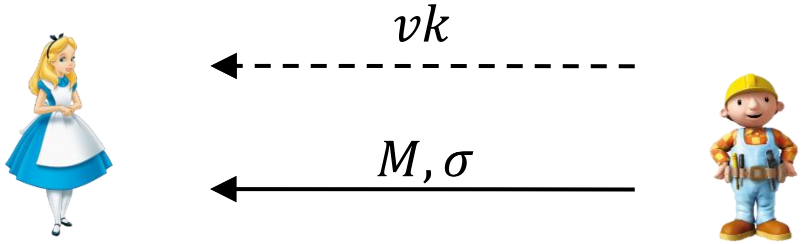
Discrete-log-based signatures: Schnorr

$$G = \langle g \rangle$$

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

KeyGen

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



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

```
$R' \leftarrow g^s B^h$
$h' \leftarrow H(R', M)$
if $h' = h$ then
    return 1
else
    return 0
```

Sign$(sk = b, M)$

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

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

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

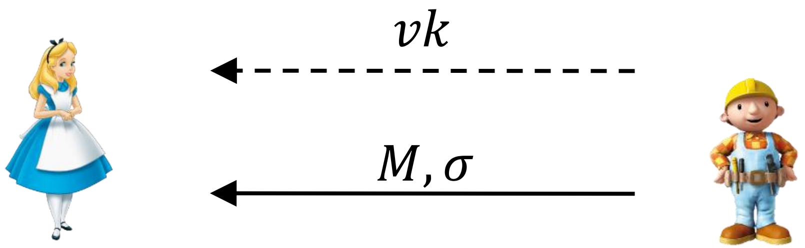
Discrete-log-based signatures: Schnorr – security

$$G = \langle g \rangle$$

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

KeyGen

```
$b \overset{\$}{\leftarrow} \{1 \ldots |G|\}$
$B \leftarrow g^b$
return $(sk = b, vk = B)$
```



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

```
$R' \leftarrow g^s B^h$
$h' \leftarrow H(R', M)$
if $h' = h$ then
    return 1
else
    return 0
```

Sign$(sk = b, M)$

```
$r \overset{\$}{\leftarrow} \{1 \ldots |G|\}$
$R \leftarrow g^r$
$h \leftarrow H(R, M)$
$s \leftarrow r - bh \bmod p$
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

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

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

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

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

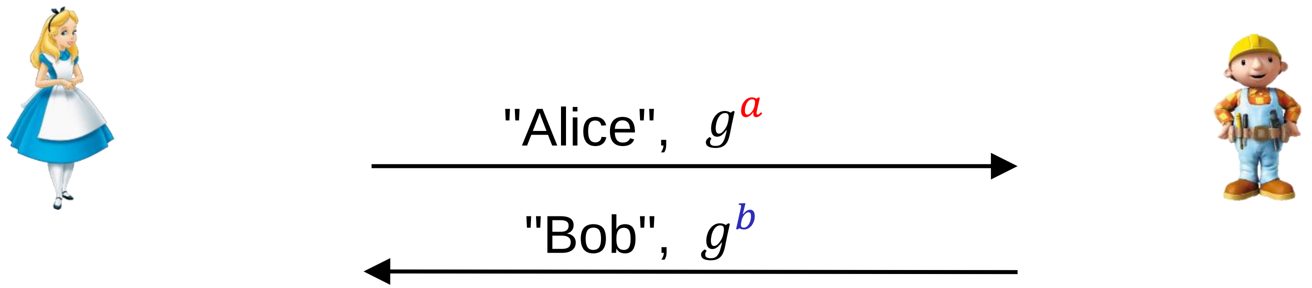
Discrete-log-based signatures: (EC)DSA

- 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}

$K \leftarrow g^{ab}$

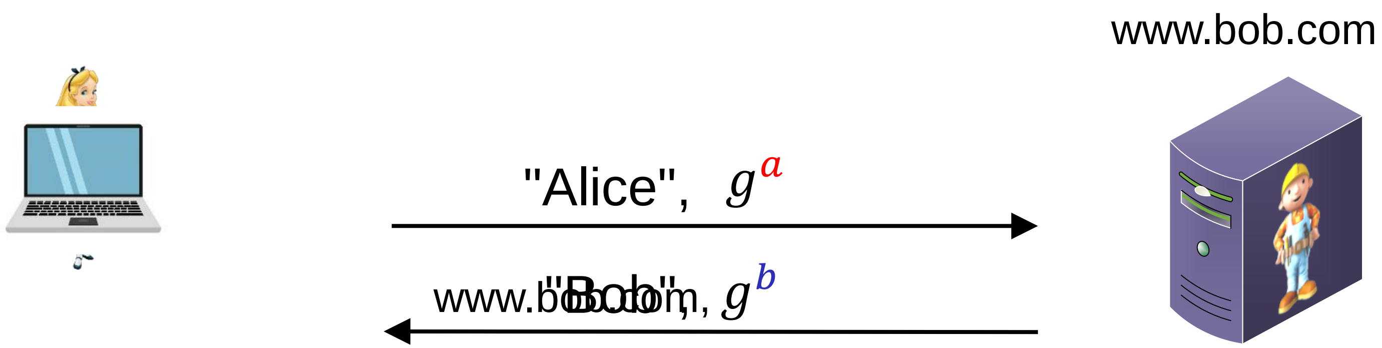
$K \leftarrow g^{ab}$

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



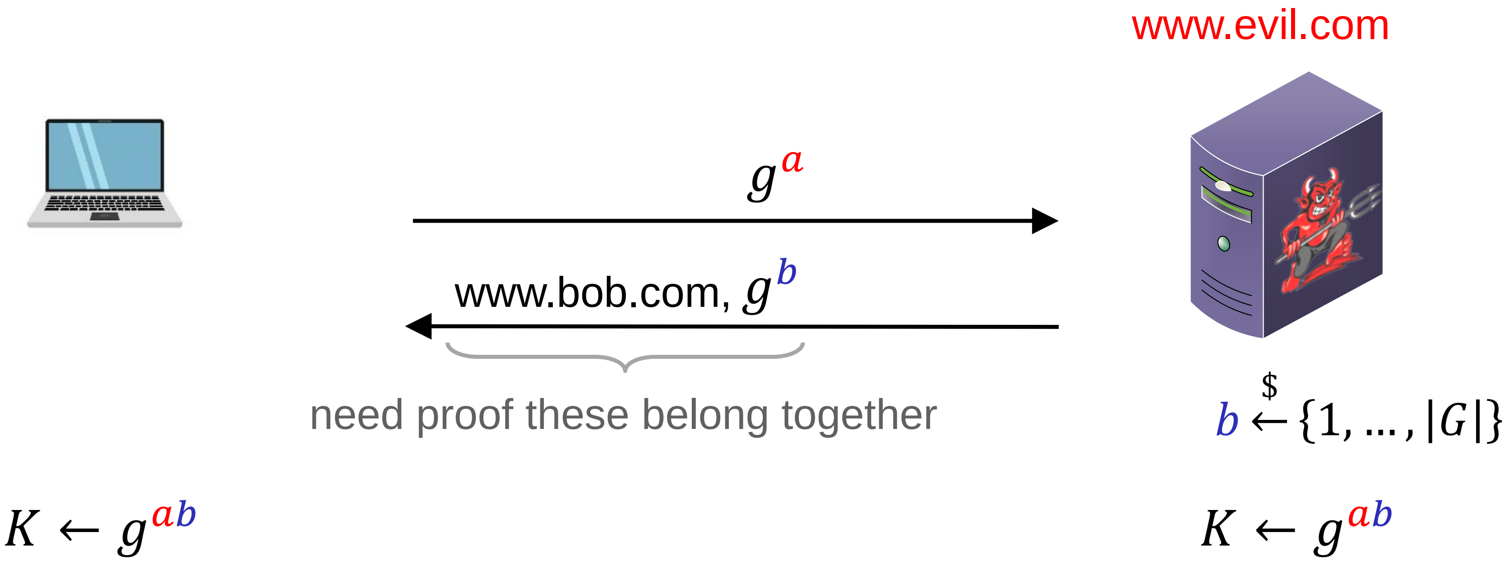
$K \leftarrow g^{ab}$ $K \leftarrow g^{ab}$

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

Identities on the internet



www.evil.com

g^a

www.bob.com, g^b

need proof these belong together

$b \xleftarrow{\$} \{1, \dots, |G|\}$

$K \leftarrow g^{ab}$

$K \leftarrow g^{ab}$

There are many Alice's and many Bob's

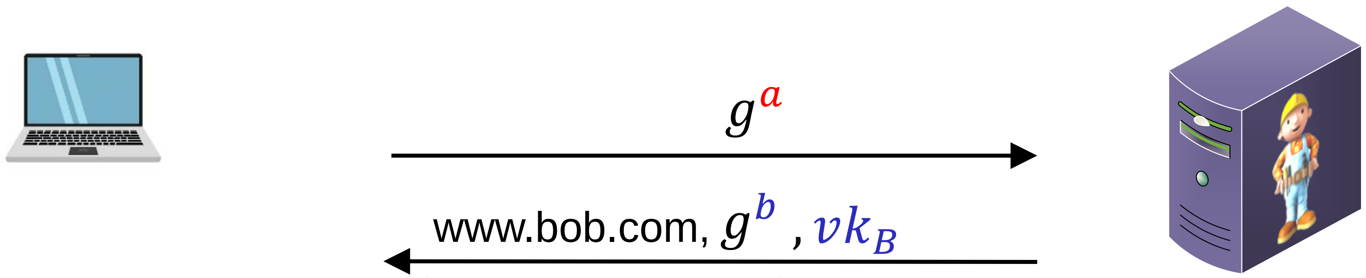
How do we know that g^a belongs to *this* particular Alice, and g^b to this particular Bob?

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

Authenticated key exchange

(sk_B, vk_B)

www.bob.com



$$g^a \longrightarrow$$

$$\longleftarrow \underbrace{\text{www.bob.com}, g^b}_{\sigma}, vk_B$$

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

$K \leftarrow g^{ab}$

else

abort

$K \leftarrow g^{ab}$

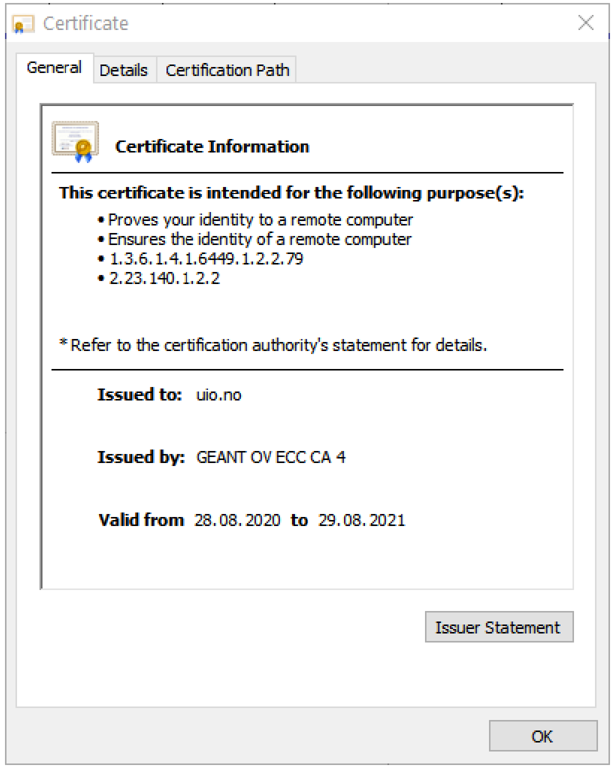
But why should we trust this vk_B? Could have been created by the adversary itself

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 Information

This certificate is intended for the following purpose(s):

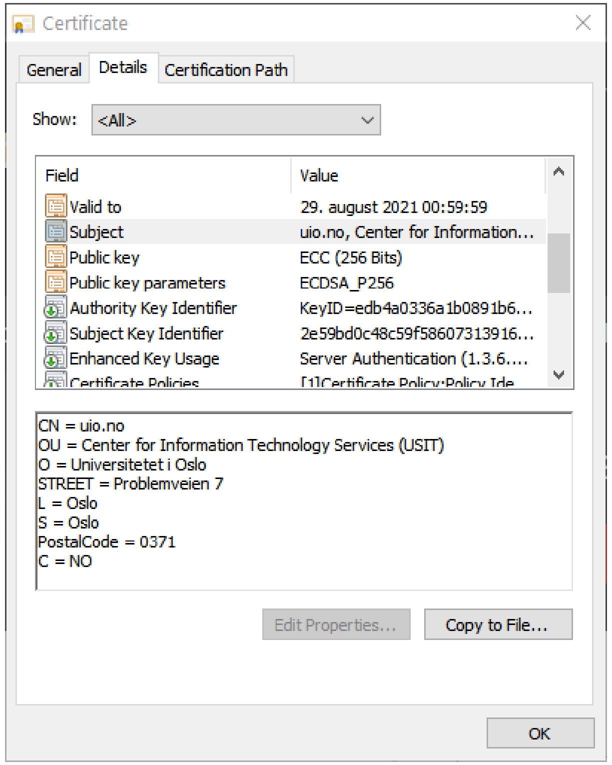
- Proves your identity to a remote computer
- Ensures the identity of a remote computer
- 1.3.6.1.4.1.6449.1.2.2.79
- 2.23.140.1.2.2

* Refer to the certification authority's statement for details.

Issued to: uio.no

Issued by: GEANT OV ECC CA 4

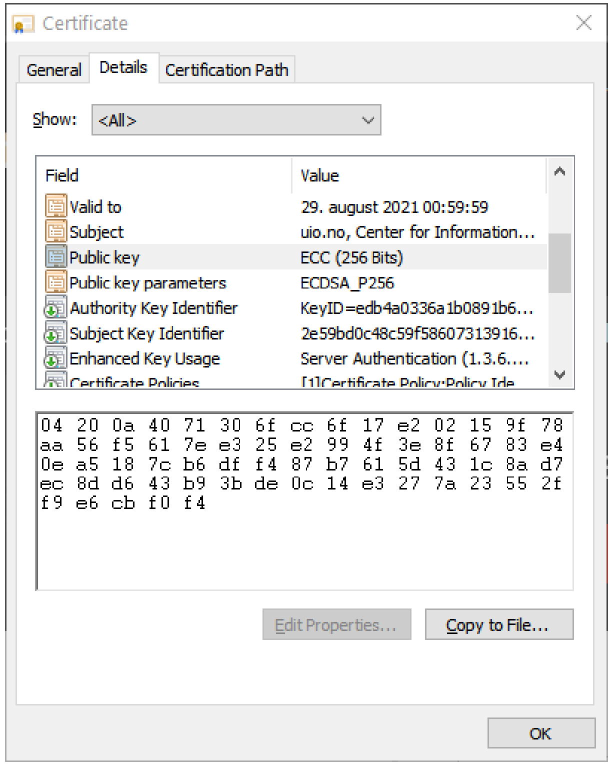
Valid from 28.08.2020 **to** 29.08.2021



Show: <All>

Field	Value
Valid to	29. august 2021 00:59:59
Subject	uio.no, Center for Information...
Public key	ECC (256 Bits)
Public key parameters	ECDSA_P256
Authority Key Identifier	KeyID=edb4a0336a1b0891b6...
Subject Key Identifier	2e59bd0c48c59f58607313916...
Enhanced Key Usage	Server Authentication (1.3.6....
Certificate Policies	[1]Certificate Policy:Policy Ide

CN = uio.no
OU = Center for Information Technology Services (USIT)
O = Universitetet i Oslo
STREET = Problemveien 7
L = Oslo
S = Oslo
PostalCode = 0371
C = NO



Show: <All>

Field	Value
Valid to	29. august 2021 00:59:59
Subject	uio.no, Center for Information...
Public key	ECC (256 Bits)
Public key parameters	ECDSA_P256
Authority Key Identifier	KeyID=edb4a0336a1b0891b6...
Subject Key Identifier	2e59bd0c48c59f58607313916...
Enhanced Key Usage	Server Authentication (1.3.6....
Certificate Policies	[1]Certificate Policy:Policy Ide

```
04 20 0a 40 71 30 6f cc 6f 17 e2 02 15 9f 78
aa 56 f5 61 7e e3 25 e2 99 4f 3e 8f 67 83 e4
0e a5 18 7c b6 df f4 87 b7 61 5d 43 1c 8a d7
ec 8d d6 43 b9 3b de 0c 14 e3 27 7a 23 55 2f
f9 e6 cb f0 f4
```

Certificate authorities (CA)

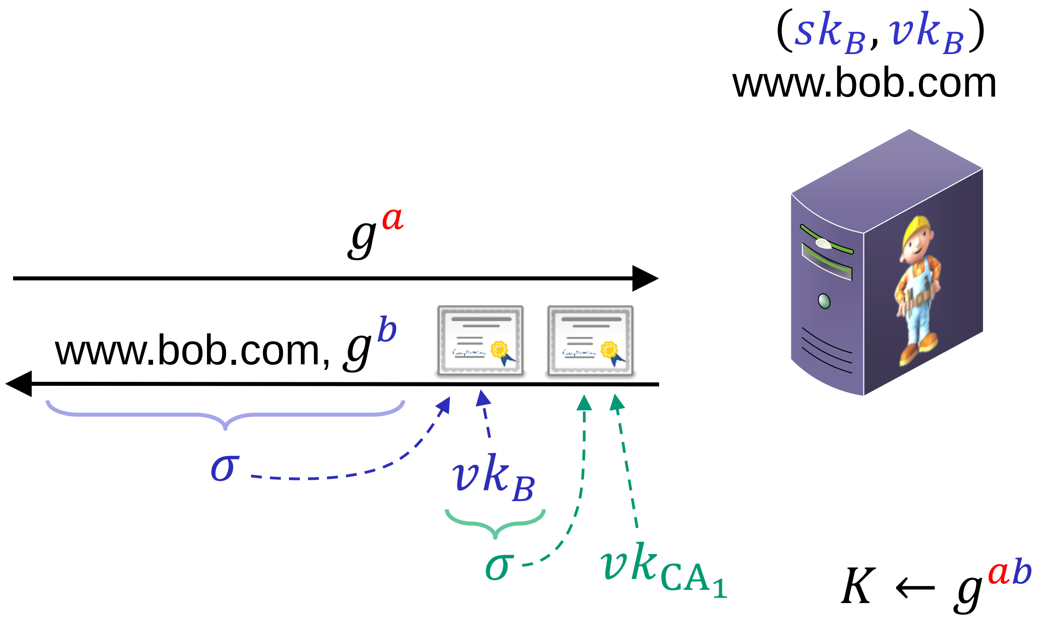
- **CA:** an issuer of digital certificates

- 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)**

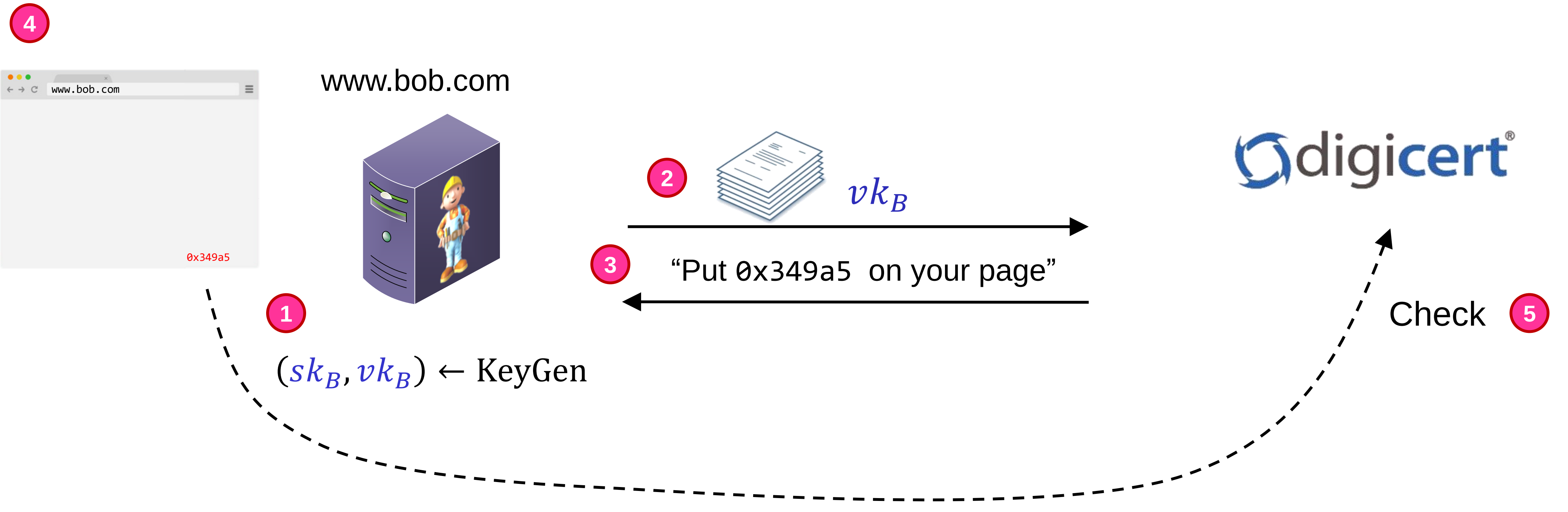
HTTPS / TLS + PKI

vk_{CA_1}



```
if Vrfy($vk_B, \dots, \sigma$) = 1
    and Vrfy($vk_{\mathrm{CA}_1}, \dots, \sigma$) = 1
        $K \leftarrow g^{ab}$
else
    abort
```

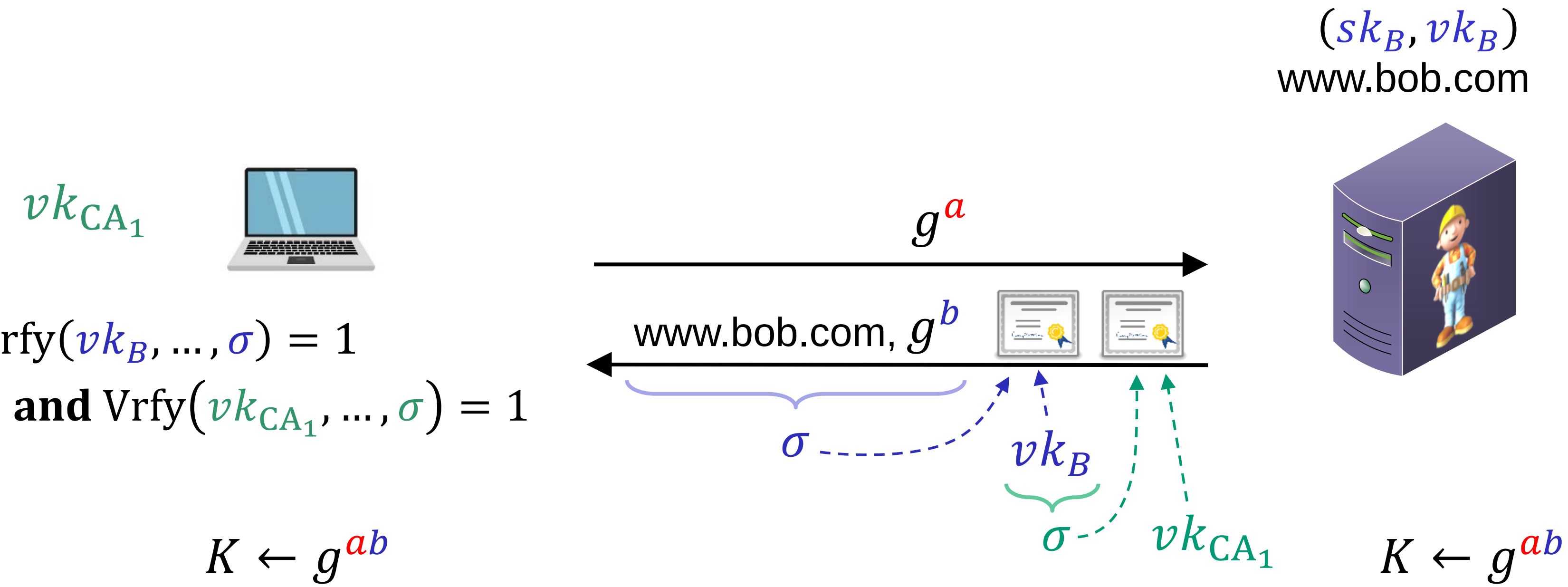
How to get a signed certificate?



Other validation methods also possible:

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

HTTPS / TLS + PKI



vk_{CA_1}

(sk_B, vk_B)
www.bob.com

g^a

www.bob.com, g^b

σ

vk_B

σ

vk_{CA_1}

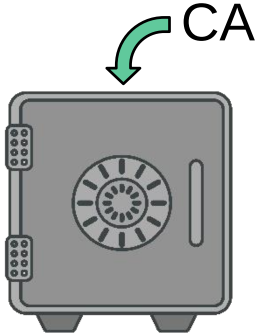
$K \leftarrow g^{ab}$

```
if Vrfy($vk_B, \dots, \sigma$) = 1
    and Vrfy($vk_{\mathrm{CA}_1}, \dots, \sigma$) = 1
        $K \leftarrow g^{ab}$
else
    abort
```

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



CA signing key

COMODO

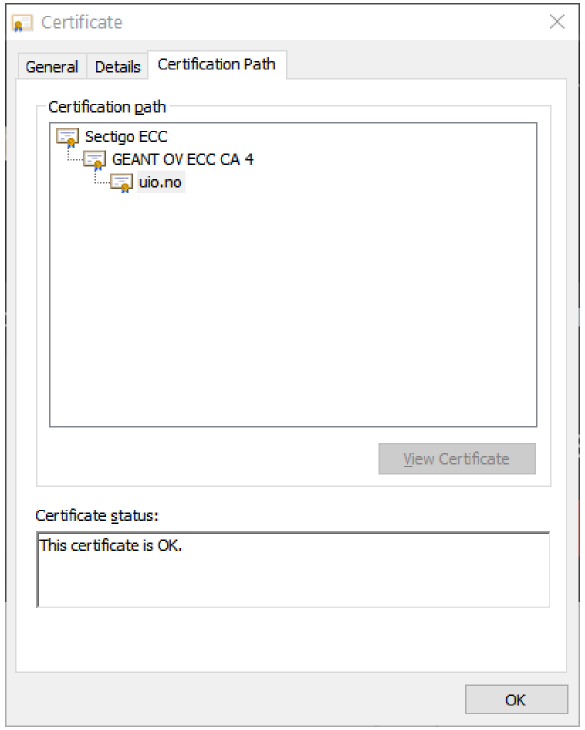
IdenTrust
part of HID Global

digicert

GoDaddy

Let's Encrypt

Certificate chains



Certification path

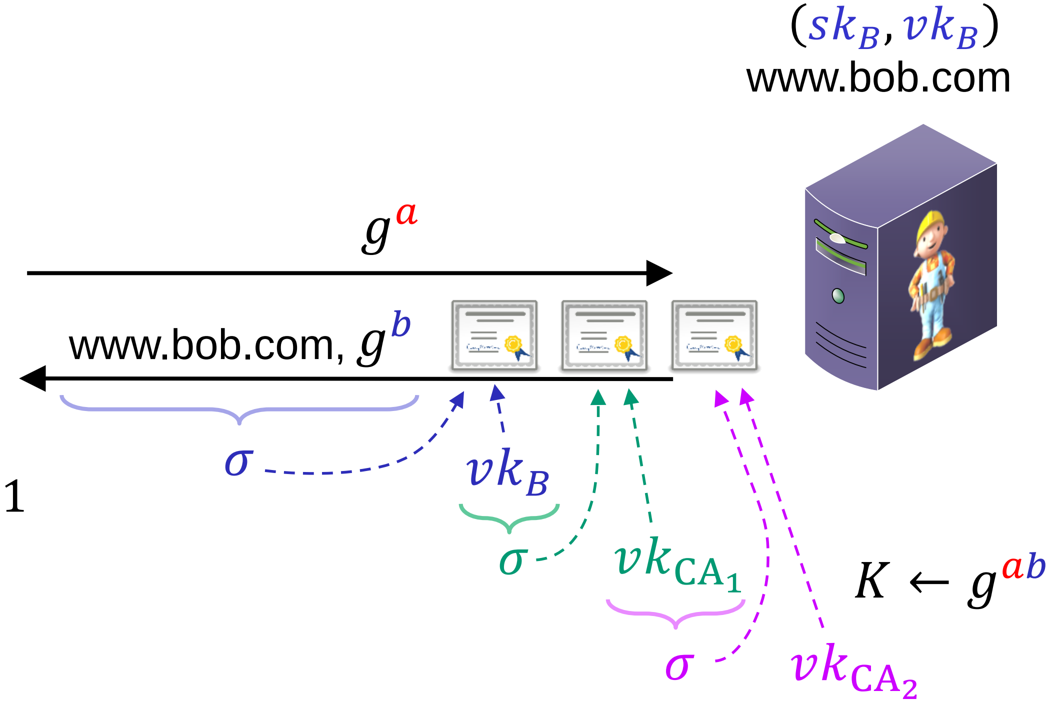
- Sectigo ECC
  - GEANT OV ECC CA 4
    - uio.no

Certificate status:

This certificate is OK.

HTTPS / TLS + PKI

```
if Vrfy($vk_B, \dots, \sigma$) = 1
    and Vrfy($vk_{\mathrm{CA}_1}, \dots, \sigma$) = 1
        and Vrfy($vk_{\mathrm{CA}_2}, \dots, \sigma$) = 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

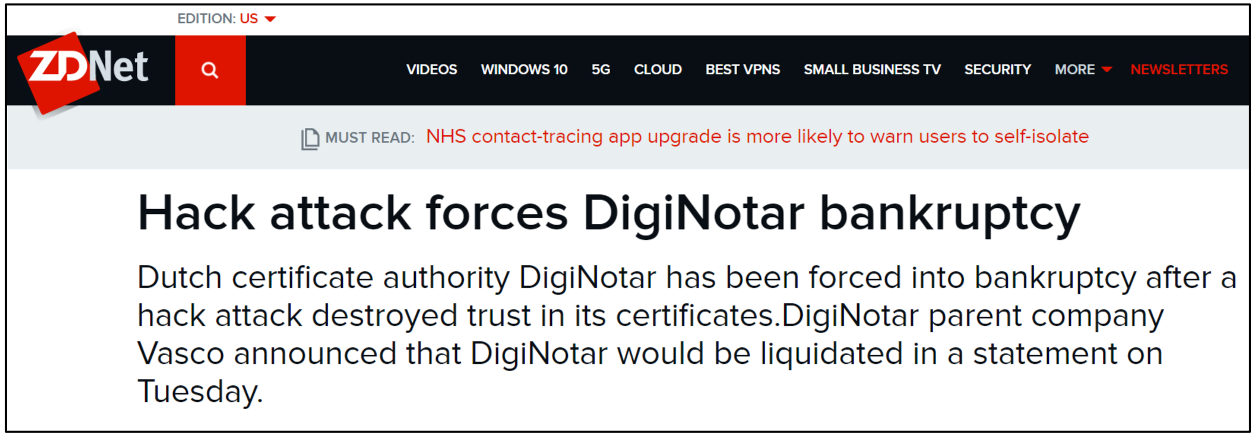
DigiNotar

- 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

DigiNotar goes bankrupt after hack

The Dutch CA goes into bankruptcy following the significant hacks claimed by ComodoHacker.

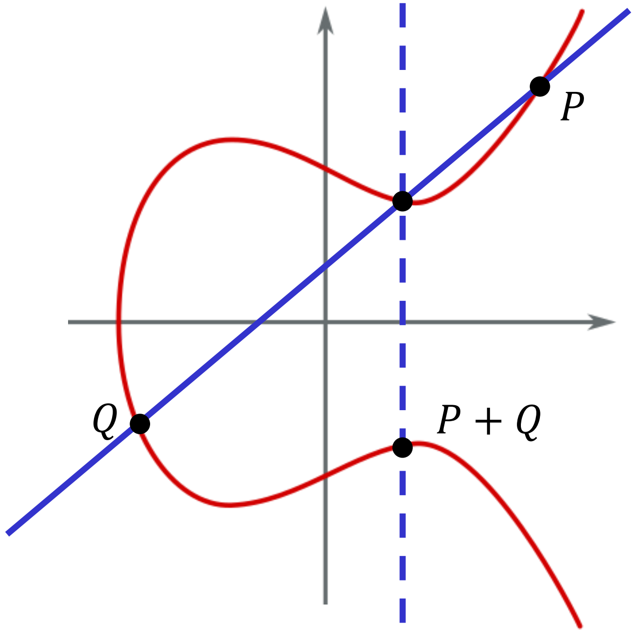
by: **Tom Brewster** 20 Sep 2011



MUST READ: NHS contact-tracing app upgrade is more likely to warn users to self-isolate

Hack attack forces DigiNotar bankruptcy

Dutch certificate authority DigiNotar has been forced into bankruptcy after a hack attack destroyed trust in its certificates.DigiNotar parent company Vasco announced that DigiNotar would be liquidated in a statement on Tuesday.



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

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

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 $\mathrm{Enc} : \mathcal{PK} \times \mathcal{M} \rightarrow \mathcal{C}$	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 $\mathrm{Sign} : \mathcal{SK} \times \mathcal{M} \rightarrow \mathcal{S}$ $\mathrm{Vrfy} : \mathcal{VK} \times \mathcal{M} \times \mathcal{S} \rightarrow \{1,0\}$	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)	$\lvert p\rvert \approx 2000\text{–}3000$ 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	$\lvert n\rvert \approx 2000\text{–}4000$ bits