Lecture 12 – Digital signatures, UF-CMA, RSA, PKI

TEK4500

15.11.2023

Håkon Jacobsen

hakon.jacobsen@its.uio.no

Administrative info

- If you did not pass the midterm or have not yet submitted two homework problem sets, but still **want to take the exam:** come see me (or send me an email) and we'll figure it out!

- I'll make old exams available on Canvas soon.

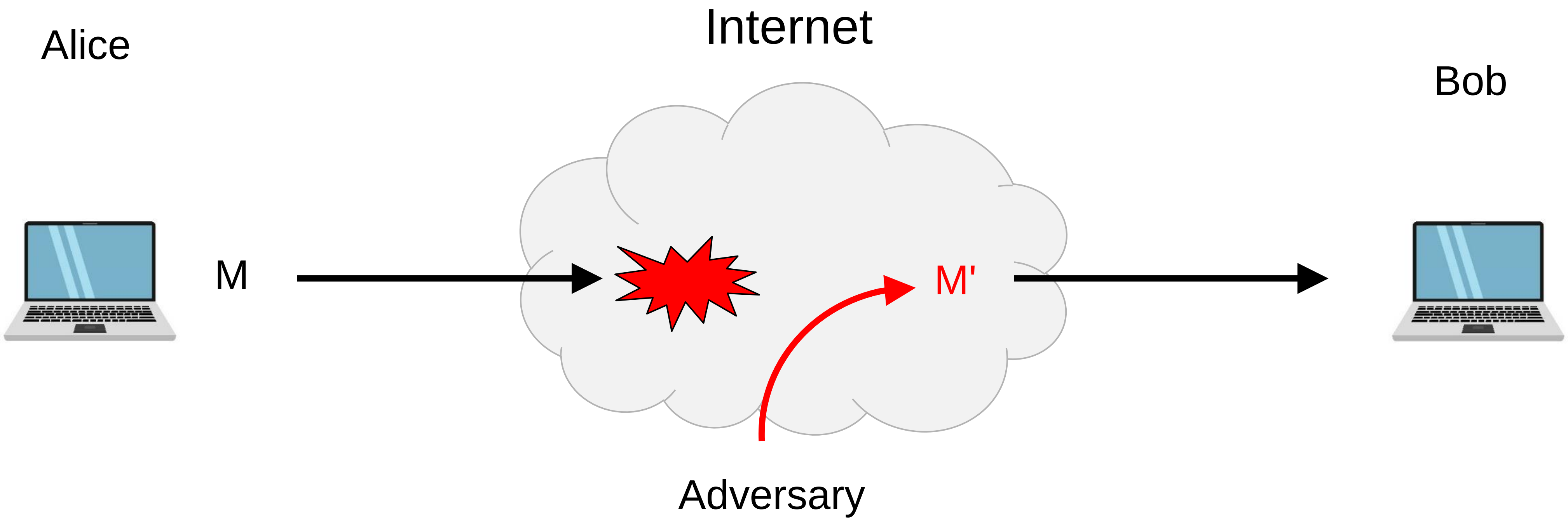
- **Remaining lectures:**
  - November 22: regular lecture (quantum computers)
  - November 29: **guest lecture!**
    - Martin Strand (researcher at FFI) will come and talk about post-quantum algorithms
  - December 6: course recap lecture + ask-me-anything session
    - If you have any specific topic you want me to repeat/treat in more detail, please let me know in advance

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)

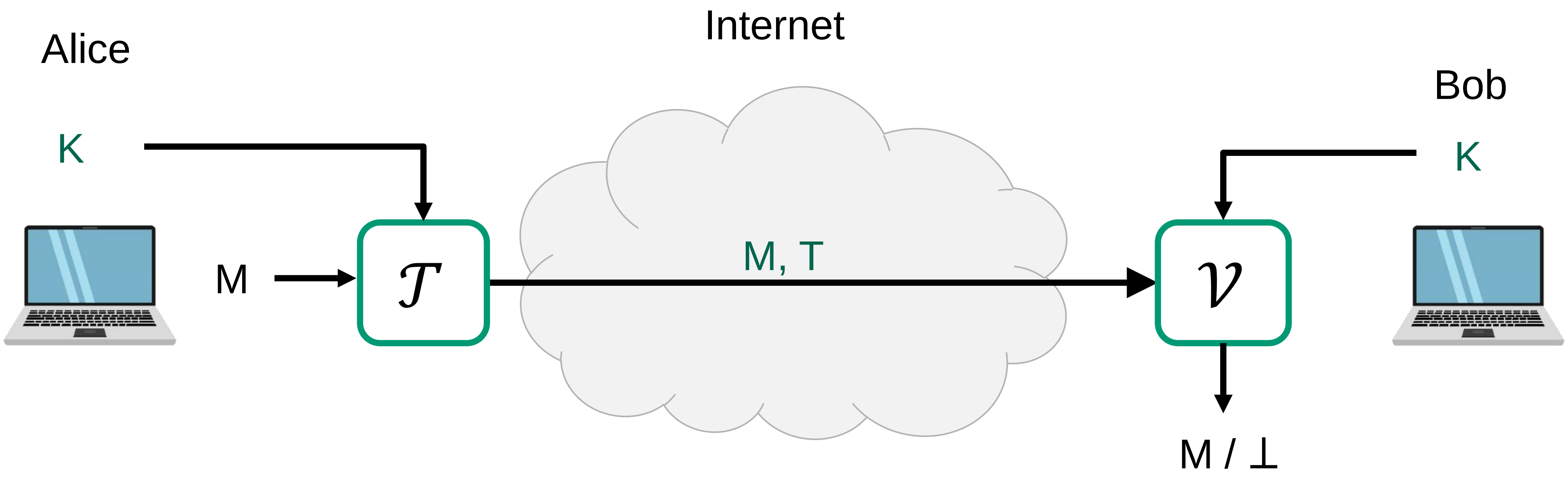
What is cryptography?



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

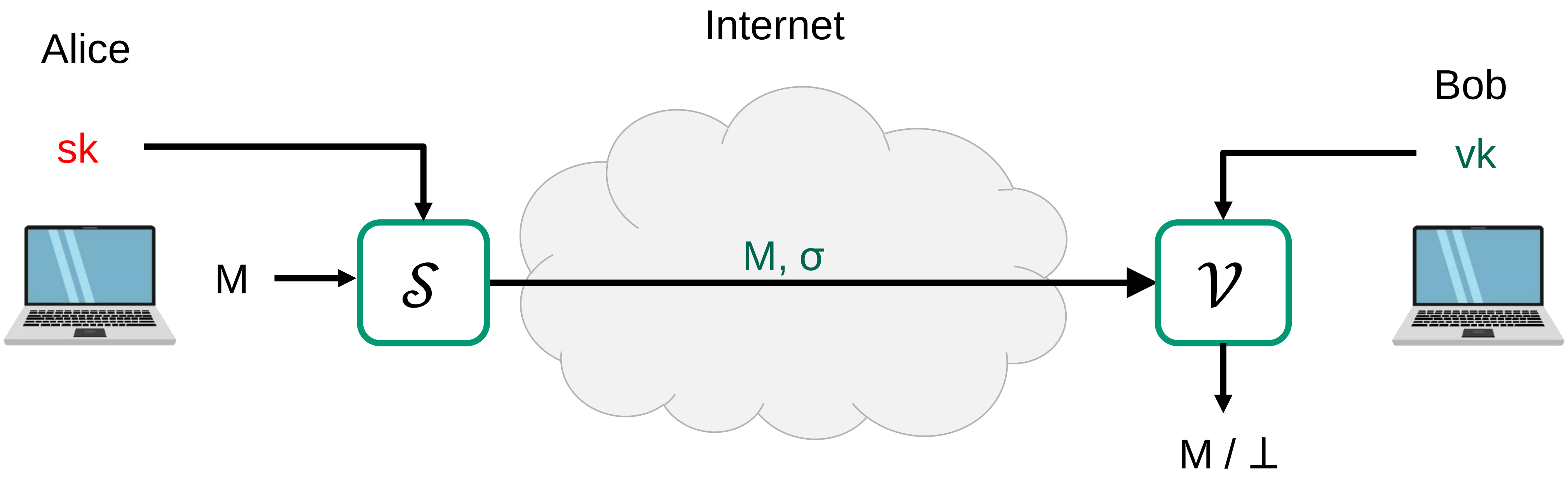


$\mathcal{T}$: tagging algorithm (public)

K : tagging / verification key (secret)

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

Digital signatures



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

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

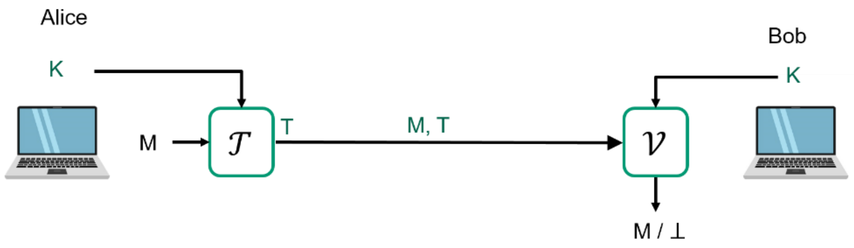
sk : signing key (secret)

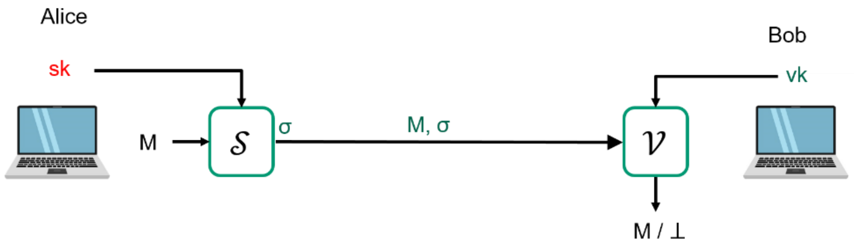
vk : verification key (public)

MACs vs. digital signatures

- MACs can only be verified by party sharing the same key
- Digital signatures can be verified by *anyone*
- **Non-repudiation:** Alice cannot deny having created σ
  - But she can deny having created T (since Bob could have done it)

MAC

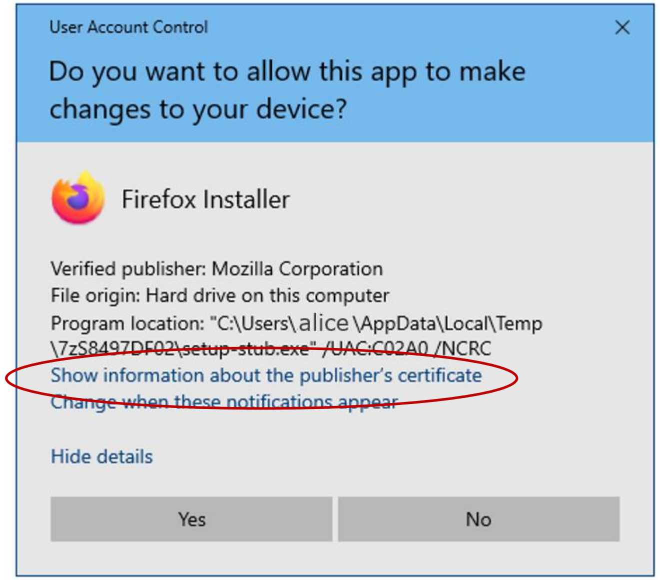


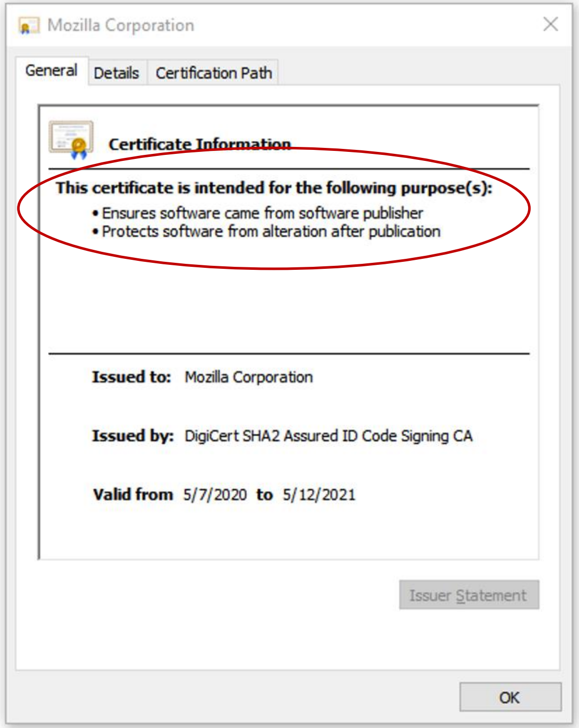


Digital signature

Applications of digital signatures

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





Digital signatures – syntax

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

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

$(sk, vk) \overset{\$}{\leftarrow} \mathrm{KeyGen}$

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

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

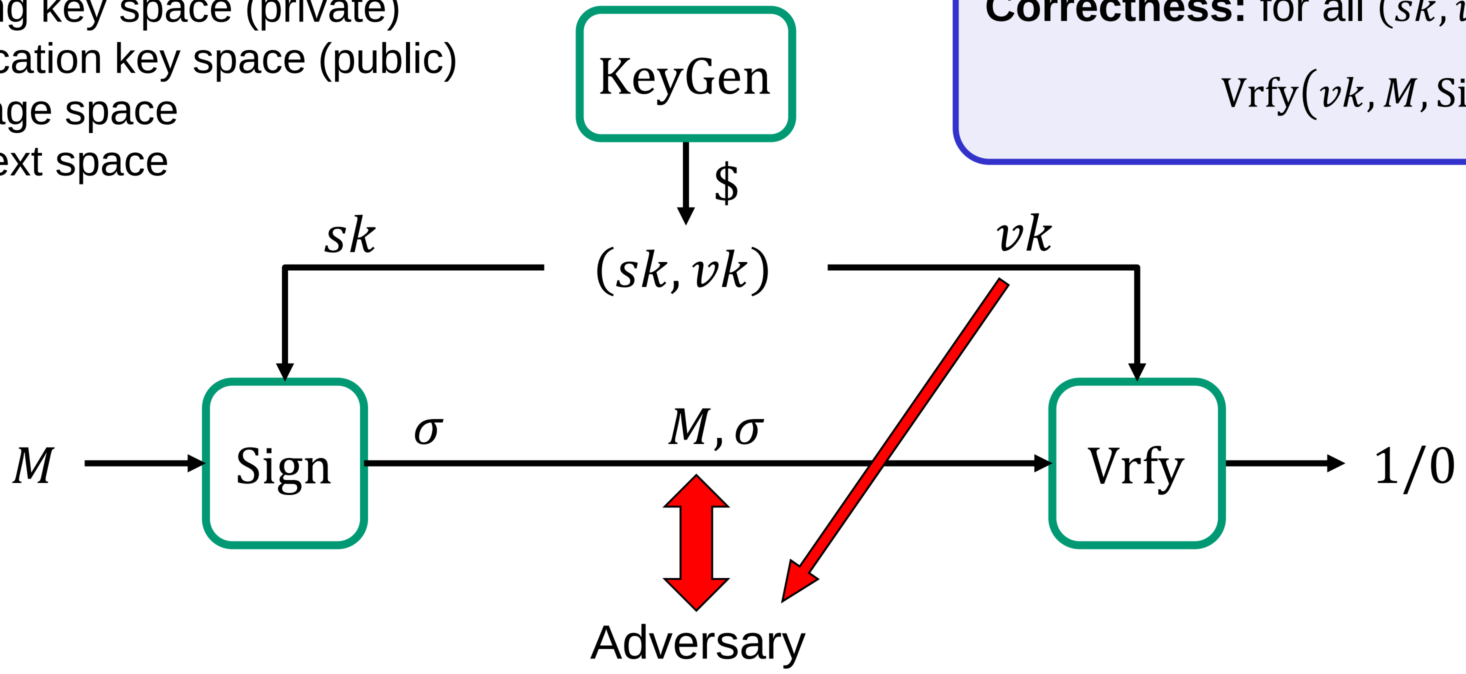
$\mathrm{Vrfy} : \mathcal{VK} \times \mathcal{M} \times \mathcal{S} \to \{0,1\}$

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

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

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

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



9

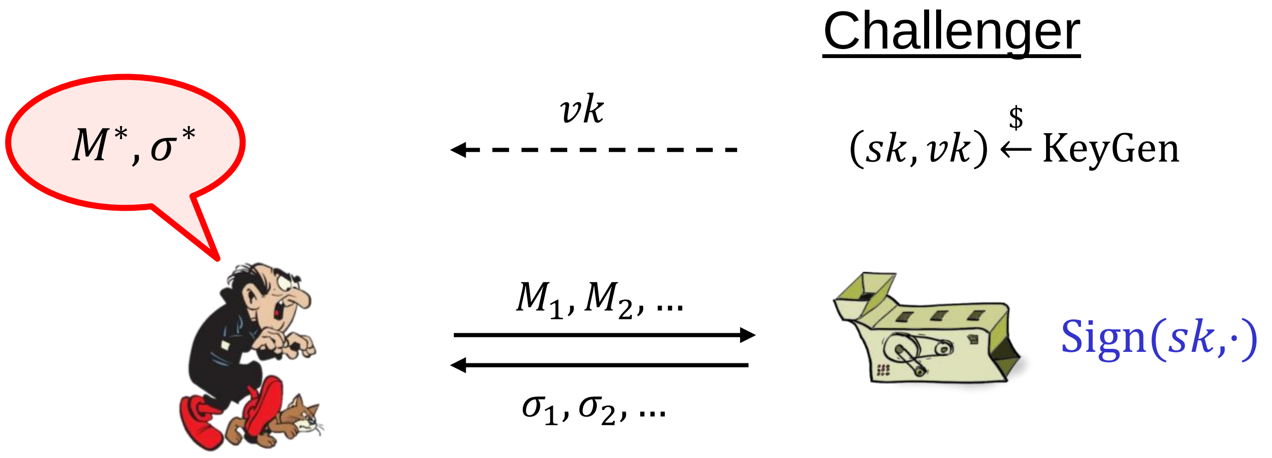
Digital signatures – security: UF-CMA

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

```
(sk, vk) <-$ Σ.KeyGen
Msgs <- [ ]                         // bookkeeping
(M*, σ*) <- A^{S(·)}(vk)
if Σ.Vrfy(vk, M*, σ*) = 1 and M* ∉ Msgs :
    return 1
else
    return 0
```

$\mathcal{S}(M)$

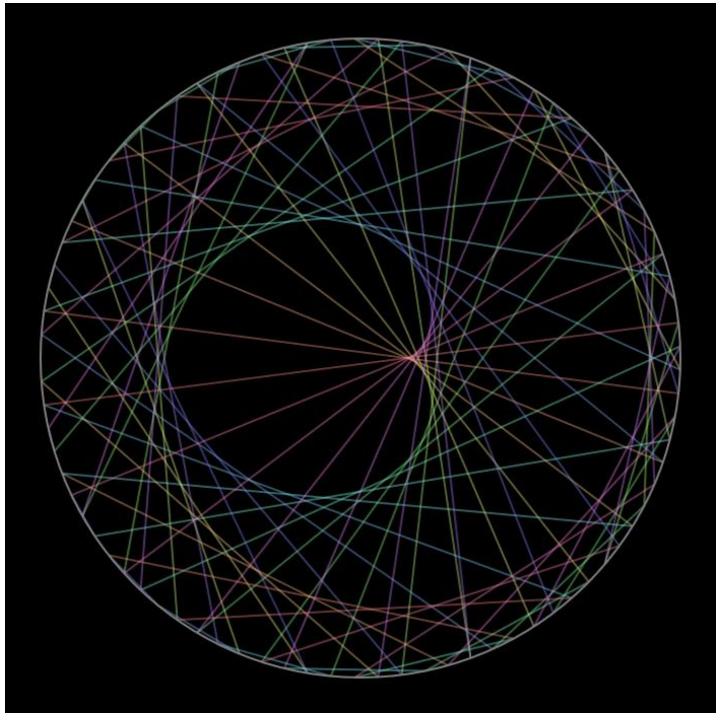
```
σ <- Σ.Sign(sk, M)
Msgs.add(M)
return σ
```



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

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

Textbook RSA signatures

$$C^{e} = M^{de} \bmod n$$

Textbook RSA signatures

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

Textbook RSA signatures

RSA.KeyGen

$p, q \xleftarrow{\$}$ two random prime numbers

$n \leftarrow p \cdot q$

$\phi(n) = (p-1)(q-1)$

choose e such that $\gcd(e, \phi(n)) = 1$

$d \leftarrow e^{-1} \bmod \phi(n)$

$sk \leftarrow (n, d)$ $pk \leftarrow (n, e)$

return (sk, pk)

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

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

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

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

$\sigma \leftarrow M^d \bmod N$
return σ

Textbook RSA signatures – (in)security

A_1

1. Output $(M, \sigma) = (1,1)$

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

$\mathbb{Z}_n^*$ is group!

A_2

1. Pick any $X \neq M$

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

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

$$=$$

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

UF-CMA Challenger

M, σ

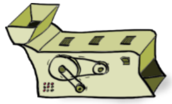
$\sigma^e = M \bmod n$

(n, e)

$(d, (e, n)) \stackrel{\$}{\leftarrow} \text{RSA.KeyGen}$

$M_1, M_2, \ldots$

$\sigma_1 = M_1^d, \sigma_2 = M_2^d, \ldots$



RSA.Sign

Textbook RSA signatures

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

RSA.KeyGen

1. $p, q \overset{\$}{\leftarrow}$ 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. $sk \leftarrow (n, d)$ $\quad pk \leftarrow (n, e)$
7. **return** (sk, pk)

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

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

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

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

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

RSA message space:

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

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

Actually want

Hashed-RSA

$$H : \{0,1\}^* \rightarrow \mathbf{Z}_n^*$$

RSA.KeyGen

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

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

```
if σ^e = H(M) mod N then
    return 1
else
    return 0
```

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

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

- $\sigma \leftarrow H(M)^d \bmod N$
- **return** σ

Hashed-RSA – security

A_1

1. ~~Output $(M, \sigma) = (1,1)$~~ Output $(X, 1)$ $\quad // H(X) = 1$

Hard to find!

A_2

1. Output $(M, \sigma) = (M, \sigma_1 \cdot \sigma_2)$

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

$$\left(H(X)^d \cdot H(Y)^d\right)^e = \underbrace{H(X) \cdot H(Y) = H(M)}_{\text{Hard to find!}} \bmod n$$

Hard to find!

UF-CMA Challenger

$(d, (e, n)) \stackrel{\$}{\leftarrow} \mathrm{RSA.KeyGen}$

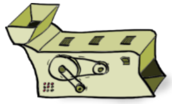
(n, e)

M, σ

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

$M_1, M_2, \ldots$

$\sigma_1 = H(M_1)^d, \ldots$



RSA. Sign

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 H is a *random oracle* then

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

Theorem: if the RSA problem is hard and H is a random oracle, then Hashed-RSA is UF-CMA secure

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

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

Discrete log based signatures

- Schnorr (see Homework 10 for details)
  - Elegant design
  - Has formal security proof (based on DLOG problem and H assumed perfect)
  - Was patented
  - One sharp edge: requires randomness during signing $\Longrightarrow$ reuse of randomness leaks private key

- (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
    - Broke all PlayStation 3's produced by Sony

Public-key infrastructure (PKI)

What are identities?

"Alice", g^{a} →

← "Bob", g^{b}

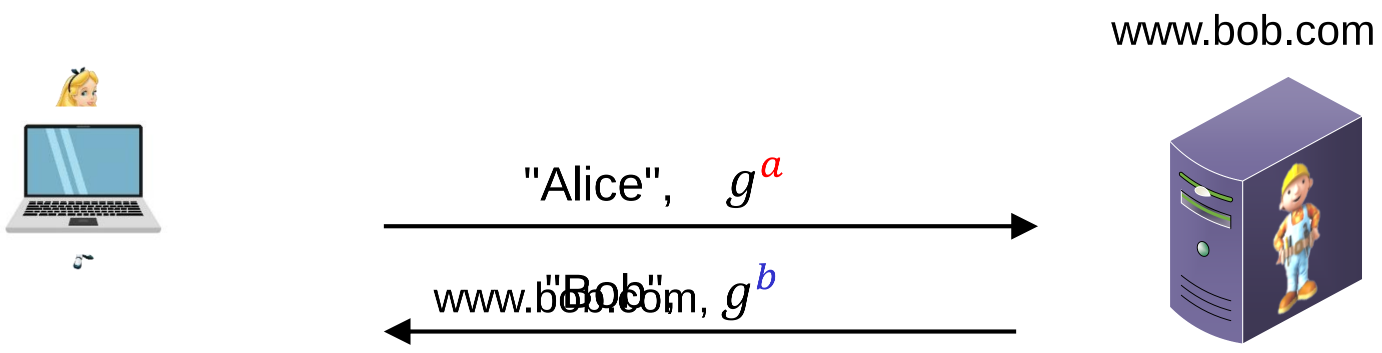
$K \leftarrow g^{ab}$ $K \leftarrow g^{ab}$

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 – 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

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

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

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

Authenticated key exchange

(sk_B, vk_B)

www.bob.com

g^{a}

www.bob.com, g^{b} $, vk_B$

σ

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

$K \leftarrow g^{ab}$

else

abort

$K \leftarrow g^{ab}$

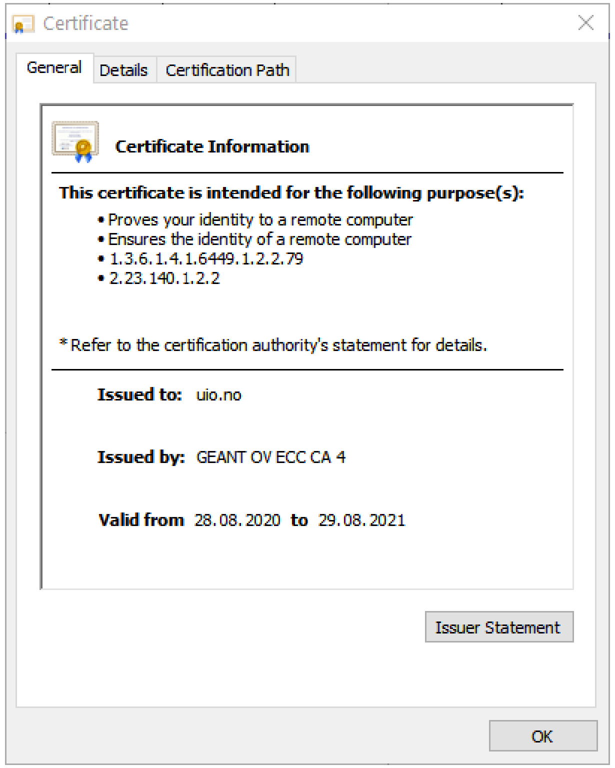
But why should we trust this vk_B? Could have been created by the adversary itself

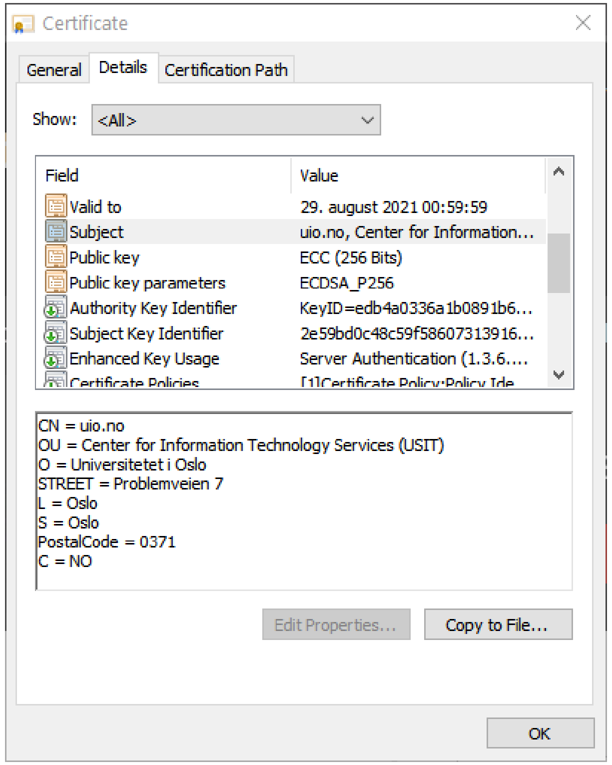
Digital certificates

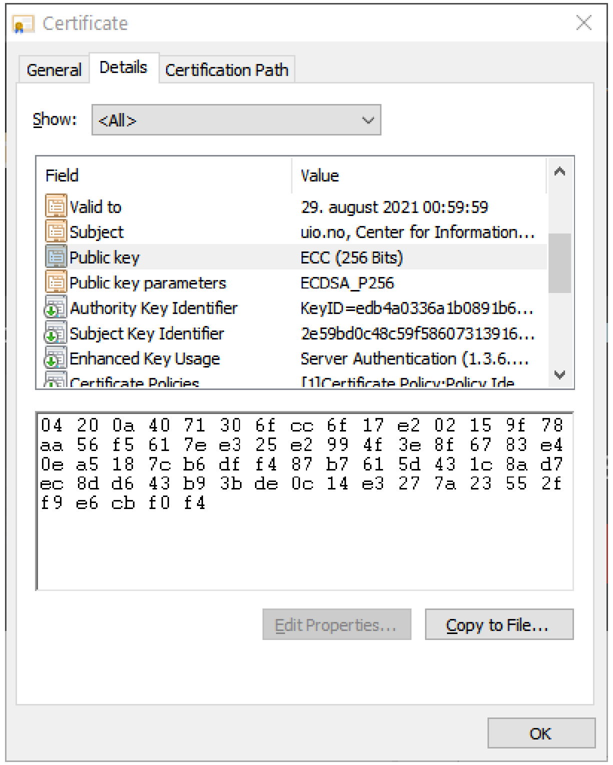
- **Digital certificate:** a way of binding a public key to an entity

- 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

vk_{CA}

(sk_B, vk_B)
www.bob.com

g^a

www.bob.com, g^b, vk_B

σ $\quad$ σ

vk_{CA}

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

and $\mathrm{Vrfy}(vk_{\mathrm{CA}}, \ldots, \sigma) = 1$

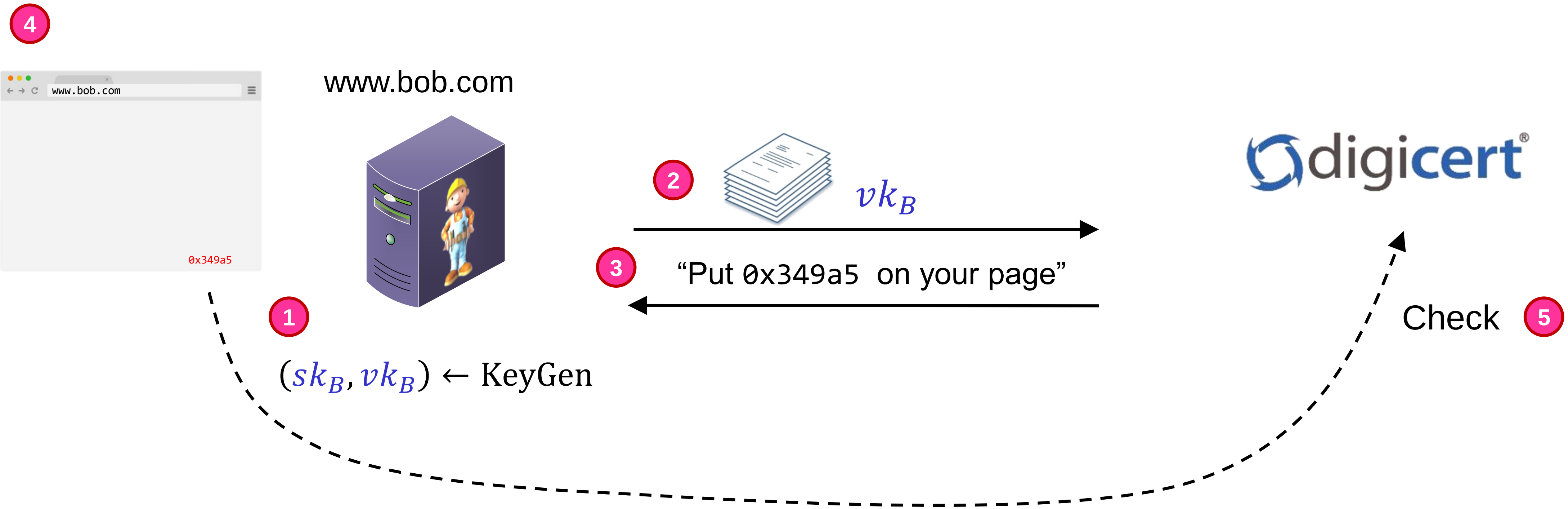
$K \leftarrow g^{ab}$

else

abort

$K \leftarrow g^{ab}$

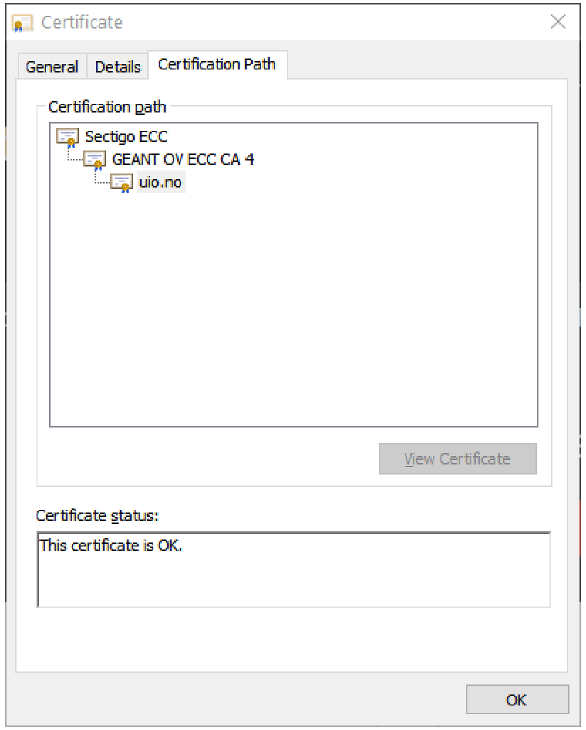
How to get a signed certificate?



Other validation methods also possible:

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

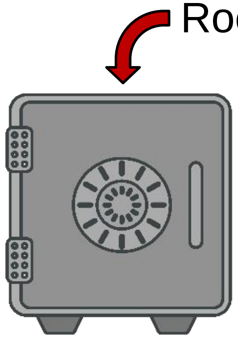
Certificate chains



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



Root CA signing key

HTTPS / TLS + PKI

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

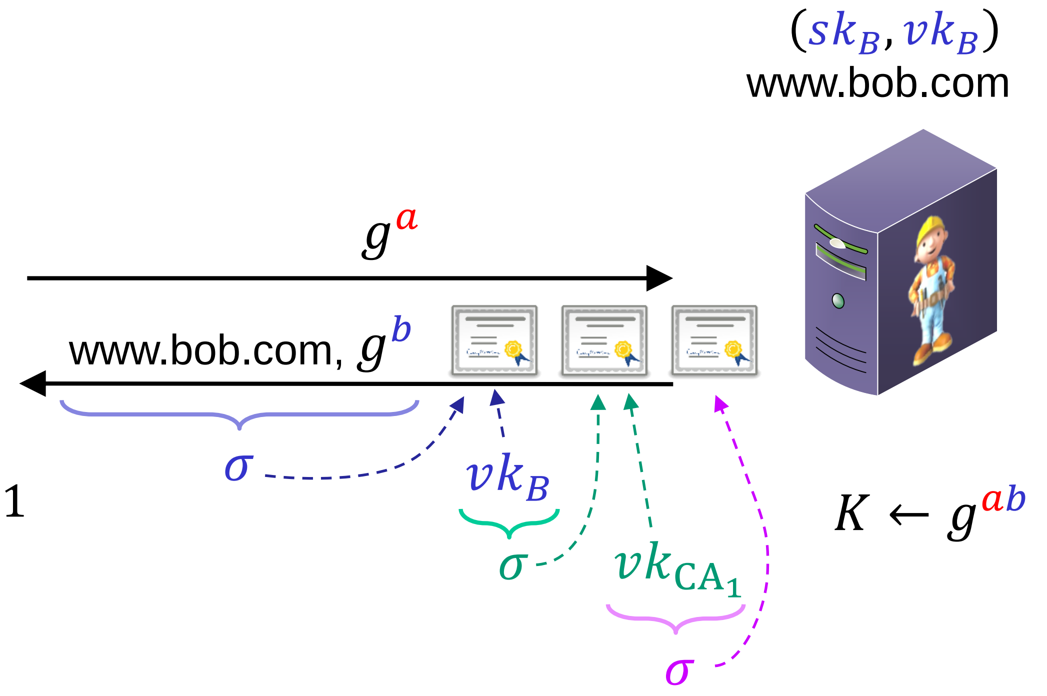
and $\mathrm{Vrfy}(vk_{\mathrm{CA}_1}, \dots, \sigma) = 1$

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

$K \leftarrow g^{ab}$

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
  - Chrome Root CA Program

- Lot's of auditing and paperwork

Let's Encrypt

- 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

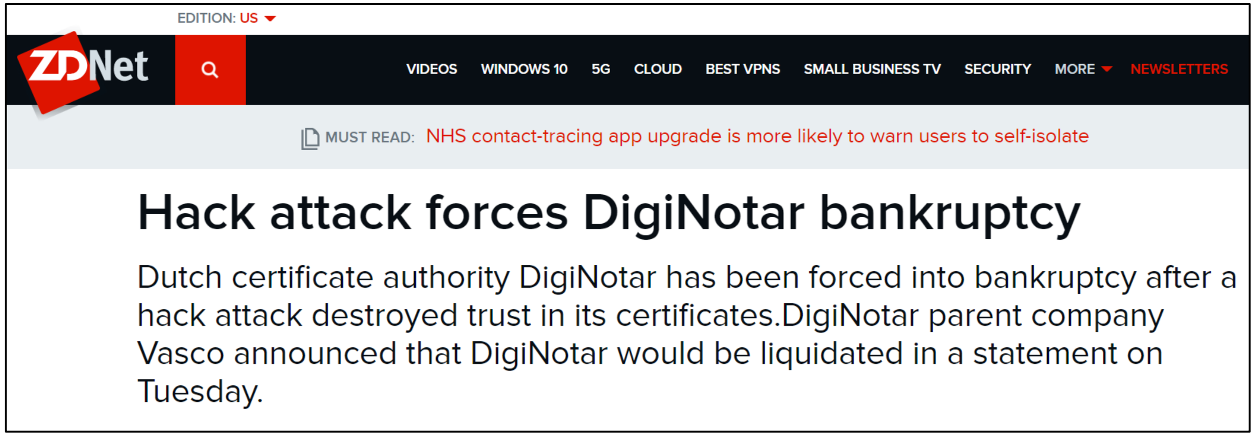

ITPro.
Business Cloud Hardware Infrastructure Security Software Technology

NEWS Home > Security

DigiNotar goes bankrupt after hack

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

by: Tom Brewster 20 Sep 2011

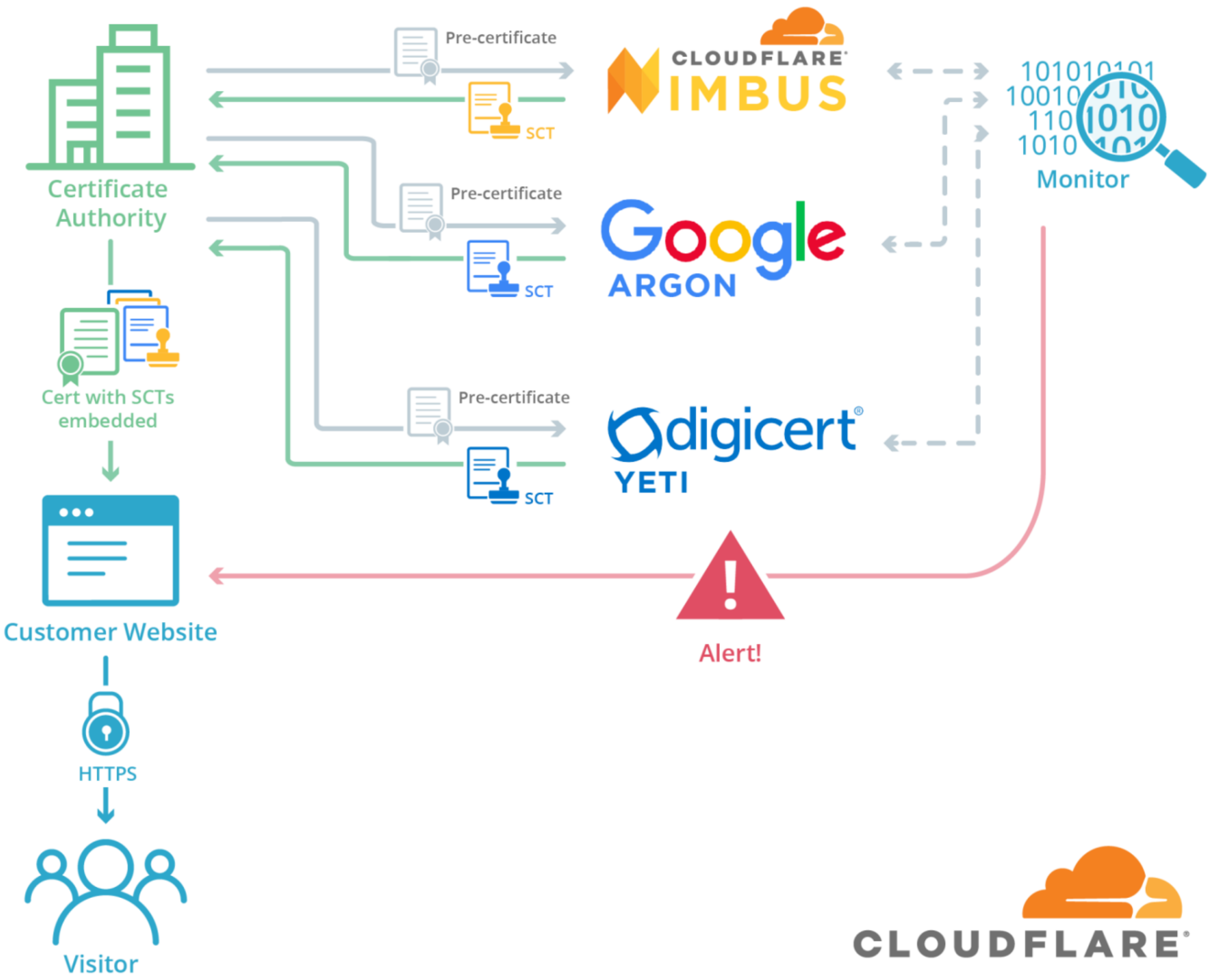

EDITION: US
ZDNet
VIDEOS WINDOWS 10 5G CLOUD BEST VPNS SMALL BUSINESS TV SECURITY MORE NEWSLETTERS

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

Hack attack forces DigiNotar bankruptcy

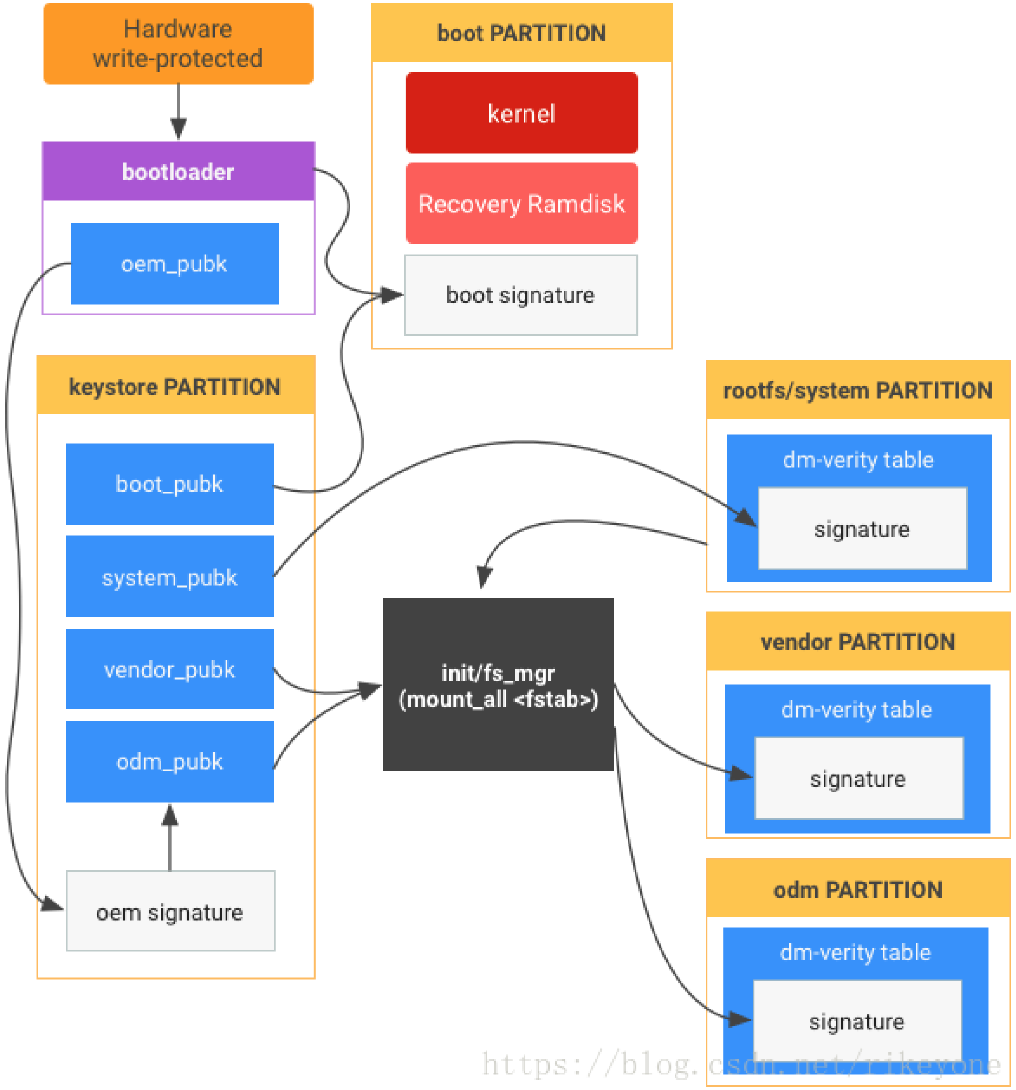
Dutch certificate authority DigiNotar has been forced into bankruptcy after a hack attack destroyed trust in its certificates.DigiNotar parent company Vasco announced that DigiNotar would be liquidated in a statement on Tuesday.

Certificate Transparency

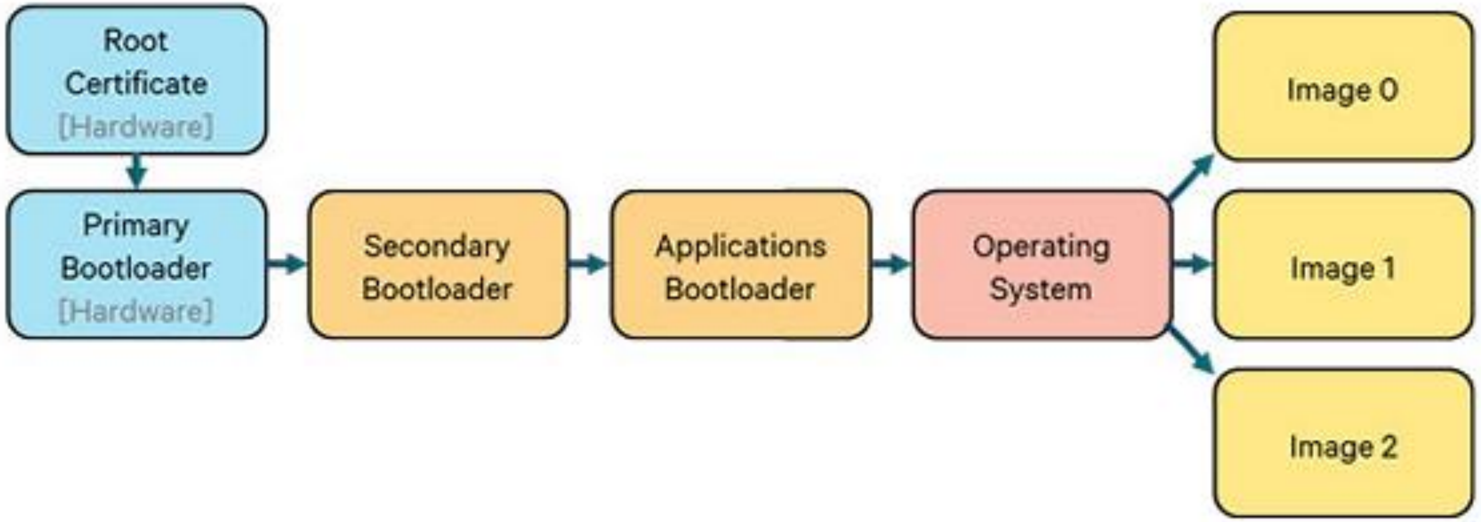


CLOUDFLARE

Android Secure Boot

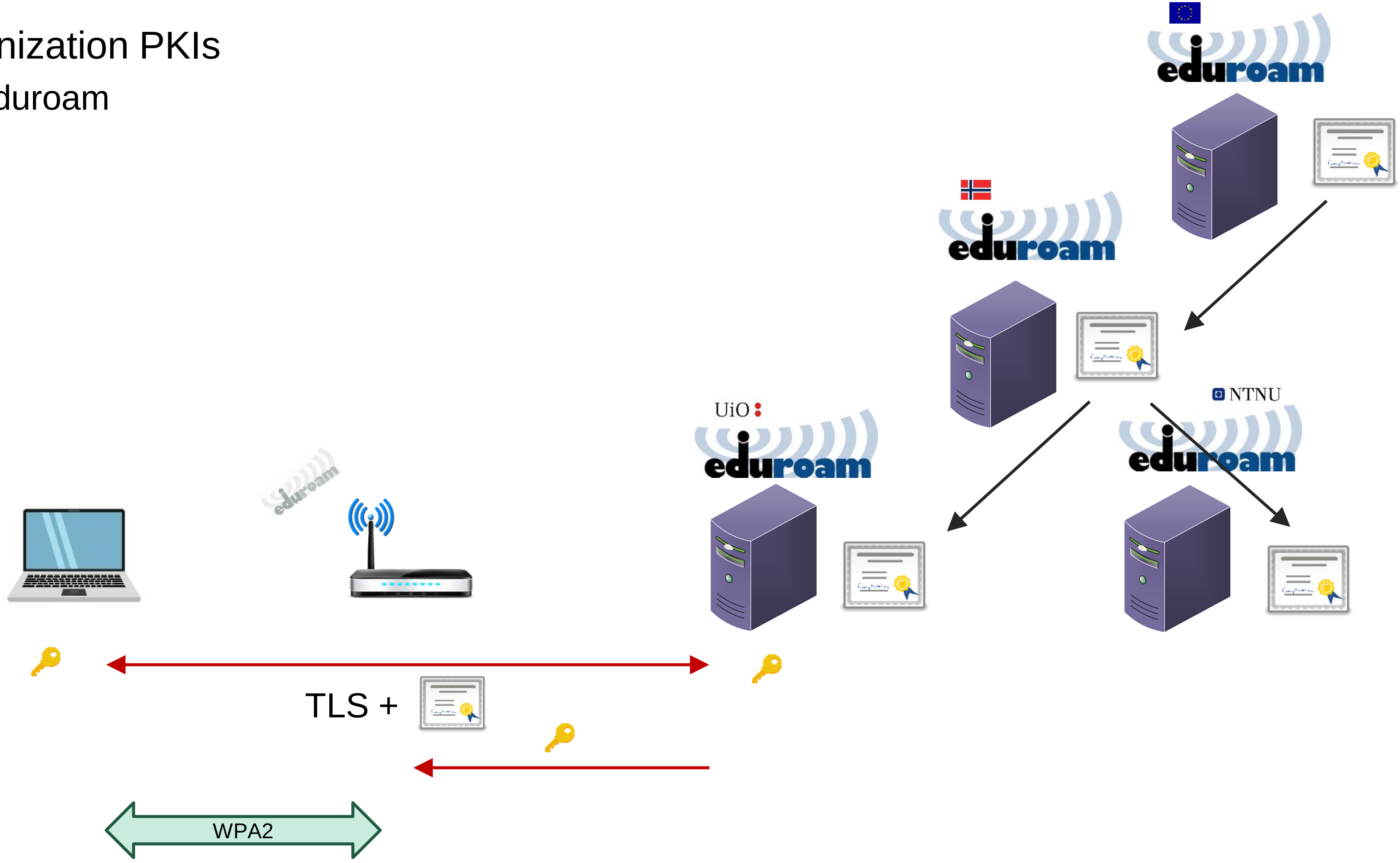


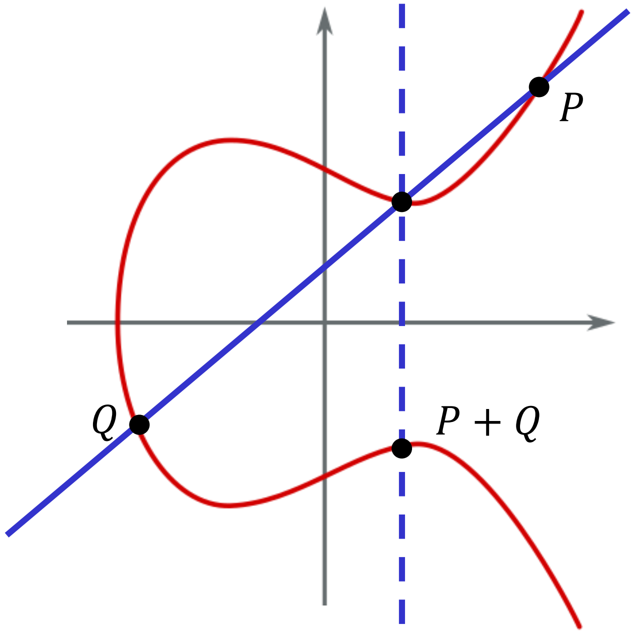
Hardware Root of Trust



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)	DH	$(\boldsymbol{Z}_p^*, \cdot)$ –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 $\text{Enc} : \mathcal{PK} \times \mathcal{M} \rightarrow \mathcal{C}$	Confidentiality: attacker should learn nothing about plaintext (except length) from ciphertexts	IND-CPA IND-CCA	ElGamal Hashed/Padded RSA Fujisaki-Okomoto-transform
Digital signatures	Create signature on variable length input $\text{Sign} : \mathcal{SK} \times \mathcal{M} \rightarrow \mathcal{S}$ $\text{Vrfy} : \mathcal{VK} \times \mathcal{M} \times \mathcal{S} \rightarrow \{1,0\}$	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

Next week

- Quantum computers