Network security and all iLabs

Modern cryptography for communications security
part 2

Benjamin Hof
hof@in.tum.de

Lehrstuhl für Netzarchitekturen und Netzdienste
Fakultät für Informatik
Technische Universität München

Cryptography – 16ws

Outline

Hash functions

Asymmetric setting

Using cryptography

Outline

Hash functions

Asymmetric setting

Using cryptography

Cryptographic hash functions

secret-key

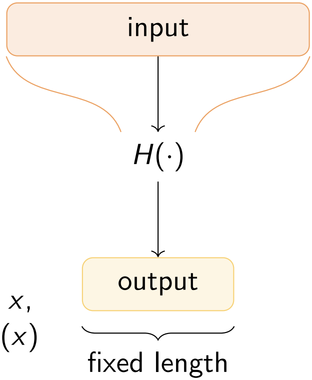
- encryption
- message authentication codes

public-key

. . .

hash functions

Hash functions


input

$H(\cdot)$

output

fixed length

- variable length input
- fixed length output

provide:

1. pre-image resistance
   given $H(x)$ with a randomly chosen x,
   infeasable to find x' s.t. $H(x') = H(x)$
   "H is one-way"
2. second pre-image resistance
   given x, infeasable to find $x' \neq x$ s.t. $H(x') = H(x)$
3. collision resistance
   infeasable to find $x \neq x'$ s.t. $H(x) = H(x')$

Example: constructing MACs from hash functions

HMAC is a popular MAC:

- opad is $0\text{x}\overline{36}$, ipad is $0\text{x}\overline{5\text{C}}$
  $tag := H(k \oplus \text{opad} \| H(k \oplus \text{ipad} \| m))$
- use SHA2-256

Used with Merkle-Damgård functions, since they allow to compute from $H(k\|m)$ the extension $H(k\|m\|tail)$.

Outline

Hash functions

Asymmetric setting

Using cryptography

The idea

We no longer have *one* shared key, but each participant has a key pair:

- a private key we give to nobody else
- a public key to be published, e. g. on a keyserver

Asymmetric cryptography

- based on mathematical problems believed to be hard
- proofs often only in the weaker random oracle model
- only authenticated channels needed for key exchange, not confidential
- less keys required
- orders of magnitude slower

Problems believed to be hard

- RSA assumption based on integer factorization
- discrete logarithm and Diffie-Hellman (DH) assumption
  - elliptic curves
  - El Gamal encryption
  - Digital Signature Standard/Algorithm

Asymmetric cryptography

symmectric

- encryption
- message authentication codes

asymmetric

- encryption
- signatures
- key exchange

hash functions

Uses

- encryption
  - encrypt with public key of key owner
  - decrypt with private key
- signatures
  - sign with private key
  - verify with public key of key owner
  - authentication with non-repudiation
- key exchange
  - protect past sessions against key compromise

Uses

- encryption
  - encrypt with public key of key owner
  - decrypt with private key
- signatures
  - sign with private key
  - verify with public key of key owner
  - authentication with non-repudiation
- key exchange
  - protect past sessions against key compromise

Encryption and signing have nothing to do with each other.

Public-key encryption scheme

1. $(pk, sk) \leftarrow Gen(1^n)$, security parameter 1^n
2. $c \leftarrow Enc_{pk}(m)$
3. $m := Dec_{sk}(c)$

We may need to map the plaintext onto the message space.

RSA primitive

Textbook RSA

0.0 $(N, p, q) \leftarrow GenModulus(1^n)$

0.1 $\phi(N) := (p-1)(q-1)$

0.2 find e: $gcd(e, \phi(N)) = 1$

0.3 $d := [e^{-1} \mod \phi(N)]$

1. public key $pk = \langle N, e \rangle$
2. private key $sk = \langle N, d \rangle$

operations:

1. public key operation on a value $y \in \mathbb{Z}_N^*$
   $z := [y^e \mod N]$
   we denote $z := RSA_{pk}(y)$
2. private key operation on a value $z \in \mathbb{Z}_N^*$
   $y := [z^d \mod N]$
   we denote $y := RSA_{sk}(z)$

RSA assumption

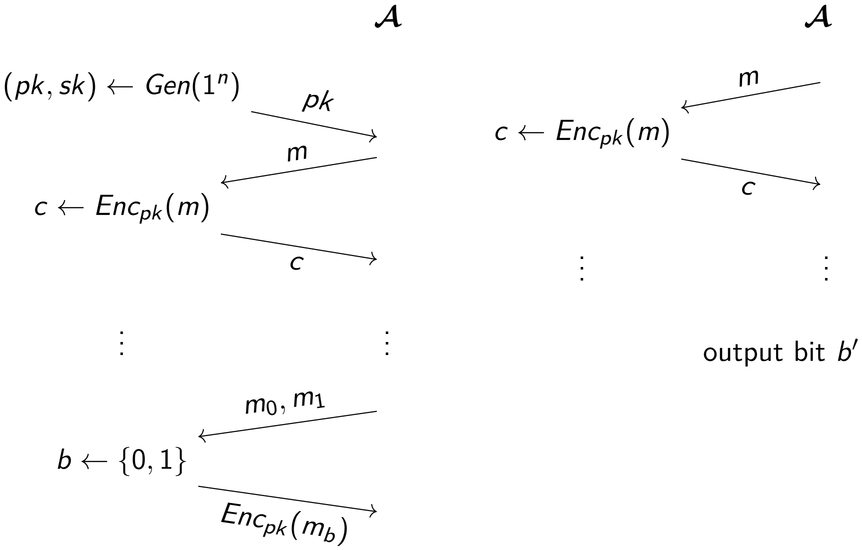
steps

1. choose uniform $x \in \mathbb{Z}_N^*$
2. $\mathcal{A}$ is given N, e, and $[x^e \bmod N]$

assumption

Infeasable to recover x.

Chosen-plaintext attack



$\mathcal{A}$

$(pk, sk) \leftarrow Gen(1^n)$

pk

m

$c \leftarrow Enc_{pk}(m)$

c

⋮ ⋮

m_0, m_1

$b \leftarrow \{0, 1\}$

$Enc_{pk}(m_b)$

$\mathcal{A}$

m

$c \leftarrow Enc_{pk}(m)$

c

⋮ ⋮

output bit b'

Security of RSA

- textbook RSA is deterministic → must be insecure against CPA

⇒ textbook RSA is not secure

- can be used to build secure encryption functions with appropriate encoding scheme

We want a construction with proof:

- use the RSA function
- breaking the construction implies ability to factor large numbers
  - "breaks RSA assumption"
  - factoring believed to be difficult (assumption!)
- secure at least against CPA

armoring ("padding") schemes needed

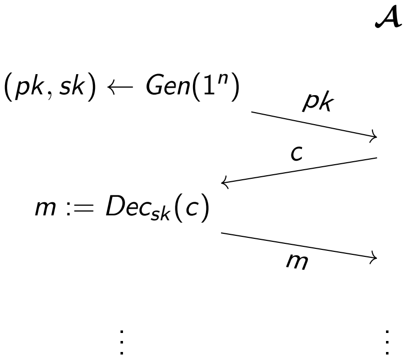
- attacks exist, but used often: PKCS #1 v1.5
- better security: PKCS #1 v2.1/v2.2 (OAEP)

Chosen-ciphertext attack

$$\mathcal{A}$$

$(pk, sk) \leftarrow Gen(1^n)$

Chosen-ciphertext attack



$\mathcal{A}$

$(pk, sk) \leftarrow Gen(1^n)$

pk

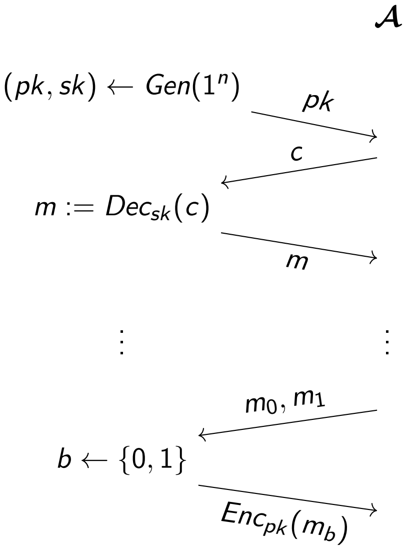
c

$m := Dec_{sk}(c)$

m

$\vdots$ $\vdots$

Chosen-ciphertext attack



$\mathcal{A}$

$(pk, sk) \leftarrow Gen(1^n)$

pk

c

$m := Dec_{sk}(c)$

m

$\vdots \qquad \vdots$

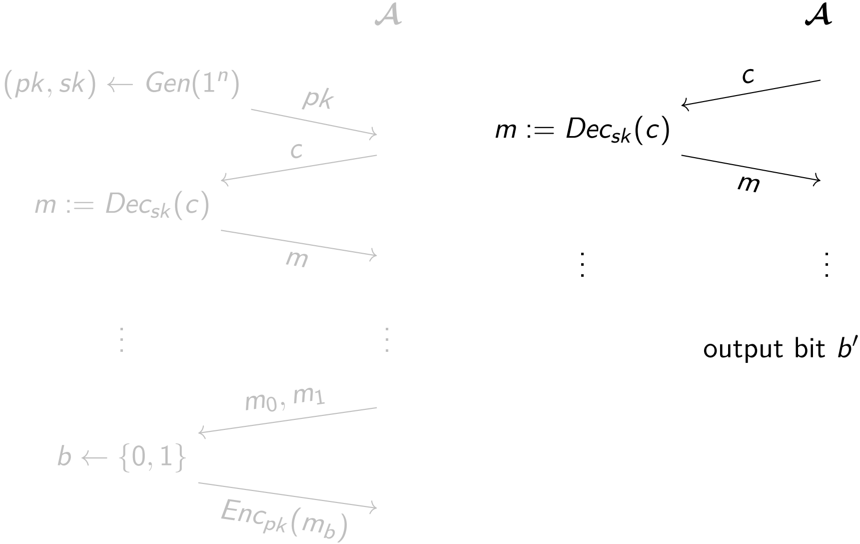
m_0, m_1

$b \leftarrow \{0, 1\}$

$Enc_{pk}(m_b)$

Adversary may not request decryption of $Enc_{pk}(m_b)$ itself.

Chosen-ciphertext attack



$\mathcal{A}$

$(pk, sk) \leftarrow Gen(1^n)$

$\xrightarrow{pk}$

$\xleftarrow{c}$

$m := Dec_{sk}(c)$

$\xrightarrow{m}$

$\vdots \qquad \vdots$

$\xleftarrow{m_0, m_1}$

$b \leftarrow \{0, 1\}$

$\xrightarrow{Enc_{pk}(m_b)}$

$\mathcal{A}$

$\xleftarrow{c}$

$m := Dec_{sk}(c)$

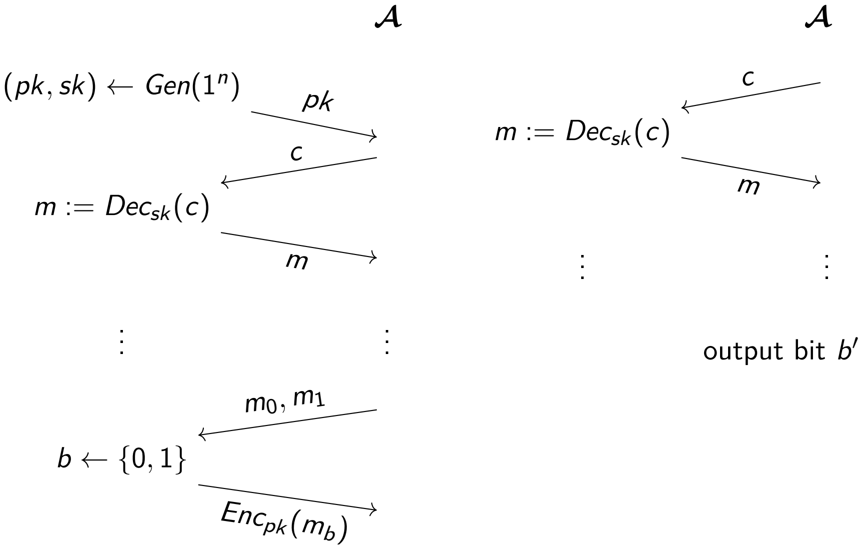
$\xrightarrow{m}$

$\vdots \qquad \vdots$

output bit b'

Adversary may not request decryption of $Enc_{pk}(m_b)$ itself.

Chosen-ciphertext attack



$\mathcal{A}$

$(pk, sk) \leftarrow Gen(1^n)$

pk

c

$m := Dec_{sk}(c)$

m

$\vdots$ $\vdots$

m_0, m_1

$b \leftarrow \{0, 1\}$

$Enc_{pk}(m_b)$

$\mathcal{A}$

c

$m := Dec_{sk}(c)$

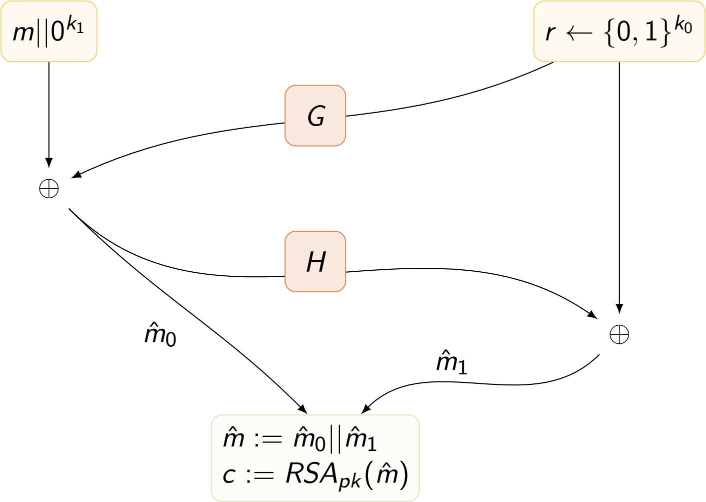
m

$\vdots$ $\vdots$

output bit b'

Adversary may not request decryption of $Enc_{pk}(m_b)$ itself.

Optimal asymmetric encryption padding



recall: $c := [\hat{m}^e \mod N]$

Discussion

A proof exists with

assumptions:

- G, H hash functions with random oracle property
- RSA assumption: RSA is one-way

result:

⇒ RSA-OAEP secure against CCA

- relaxation: negligible probability

Signature scheme

1. $(pk, sk) \leftarrow Gen(1^n)$
2. $\sigma \leftarrow Sign_{sk}(m)$
3. $b := Vrfy_{pk}(m, \sigma)$

$b = 1$ means valid, $b = 0$ invalid

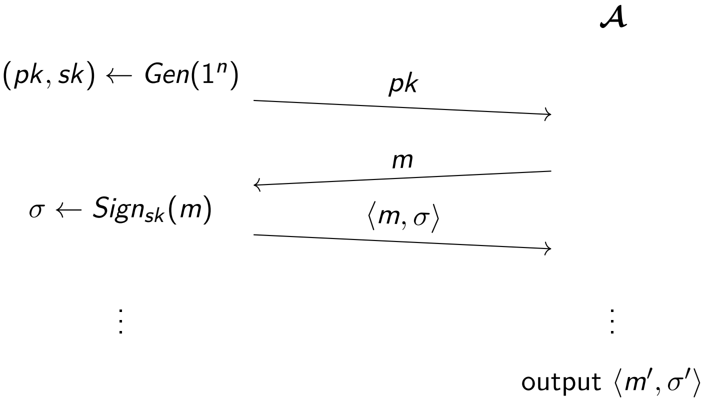
Signatures

- (often) slower than MACs
- non-repudiation
- verify OS packages

RSA signatures

- RSA not a secure signature function
- PKCS #1 v1.5
- use RSASSA-PSS (“probabilistic signature scheme”)

Adaptive chosen-message attack



$\mathcal{A}$

$(pk, sk) \leftarrow Gen(1^n)$

pk

m

$\sigma \leftarrow Sign_{sk}(m)$

$\langle m, \sigma \rangle$

$\vdots$

$\vdots$

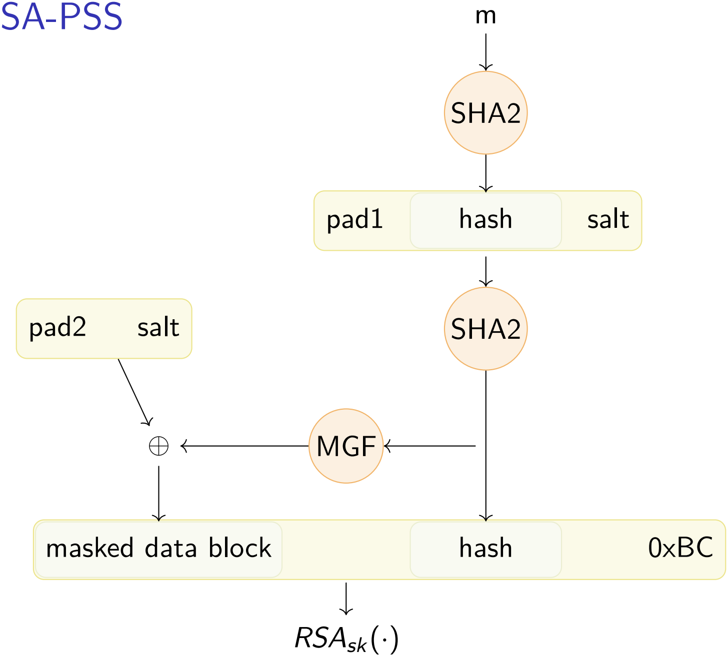
output $\langle m', \sigma' \rangle$

- let $\mathcal{Q}$ be the set of all queries m
- $\mathcal{A}$ succeeds, iff $Vrfy_{pk}(m', \sigma') = 1$ and $m' \notin \mathcal{Q}$

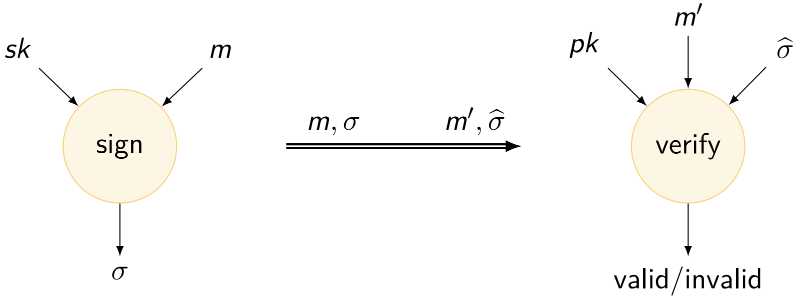
Goal

- signature function using RSA
- breaking signature function implies breaking the RSA assumption
- proof

RSASSA-PSS


m
SHA2
pad1
hash
salt
SHA2
pad2
salt
⊕
MGF
masked data block
hash
0xBC
$RSA_{sk}(\cdot)$

Overview: signatures using RSA



$Sign_{sk}(m)$:

$$\begin{aligned} em &\leftarrow PSS(m) \quad \text{// encoding} \\ \sigma &:= RSA_{sk}(em) \end{aligned}$$

$Vrfy_{pk}(m', \widehat{\sigma})$:

$$\begin{aligned} \widehat{em} &:= RSA_{pk}(\widehat{\sigma}) \\ \widehat{salt} &:= recover\text{-}PSS\text{-}salt(\widehat{em}) \\ em' &:= PSS(m', \widehat{salt}) \\ em' &\overset{?}{=} \widehat{em} \end{aligned}$$

Discussion

A proof exists with

assumptions:

- random oracle model
- RSA assumption: RSA is one-way

result:

$\Rightarrow$ RSA-PSS existentially unforgeable under adaptive chosen-message attack

- relaxation: negligible probability

Hybrid approach

Public-key cryptography

- valuable properties
- slow

Hybrid encryption

- protect shared key with public-key cryptography
- protect bulk traffic with secret-key cryptography

Example

$$k \leftarrow \{0,1\}^n$$
$$w \leftarrow \widehat{Enc_{pk}}(k)$$
$$c_0 \leftarrow Enc_k(msg_0)$$
$$c_1 \leftarrow Enc_k(msg_1)$$

transmit: $\langle w, c_0, c_1 \rangle$

Combining secret-key and public-key methods in protocols

e. g.:

handshake

- Diffie-Hellman key exchange
- signatures for entity authentication
- key derivation
- . . .

transport

- secret-key authenticated encryption
- replay protection

Perfect forward security

Assume

- long-term (identity) keys
- session keys (for protecting one connection)

Idea

- attacker captures secret-key encrypted traffic
- later: an endpoint is compromised $\rightarrow$ keys are compromised

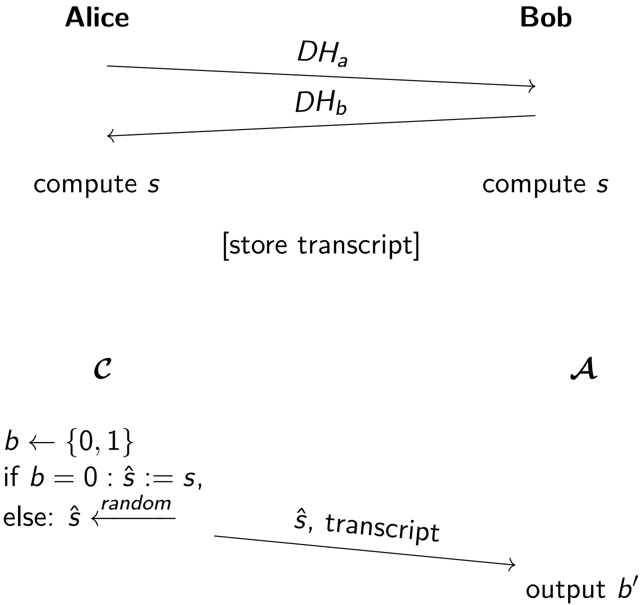
We want: security of past connections should not be broken.

Perfect forward security

protection of past sessions against:

- compromise of session key
- compromise of long-term key

Decisional Diffie-Hellman assumption



Alice $\xrightarrow{DH_a}$ **Bob**

Alice $\xleftarrow{DH_b}$ **Bob**

compute s

compute s

[store transcript]

$\mathcal{C}$

$\mathcal{A}$

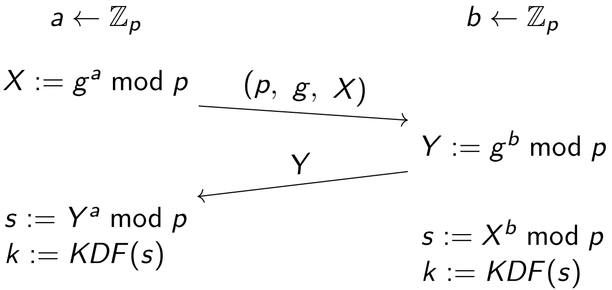
$b \leftarrow \{0, 1\}$
if $b = 0 : \hat{s} := s$,
else: $\hat{s} \xleftarrow{random}$

$\xrightarrow{\hat{s}, \text{ transcript}}$

output b'

Textbook Diffie-Hellman key exchange

- p prime
- generator g (primitive root for cyclic group of $\mathbb{Z}_p$):
  $\{g^0, g^1, g^2, \ldots\} = \{1, 2, \ldots, p-1\}$



$a \leftarrow \mathbb{Z}_p$

$b \leftarrow \mathbb{Z}_p$

$X := g^a \bmod p$

$\xrightarrow{(p,\ g,\ X)}$

$Y := g^b \bmod p$

$\xleftarrow{Y}$

$s := Y^a \bmod p$
$k := KDF(s)$

$s := X^b \bmod p$
$k := KDF(s)$

- $Y^a = g^{ba} = g^{ab} = X^b \mod p$
- insecure for certain weak values

Perfect forward security

- generate new DH key for each connection
- wipe old shared keys

Compromise of long term keys in combination with eavesdropping does not break security of past connections anymore!

Outline

Hash functions

Asymmetric setting

Using cryptography

Key size equivalents

secret-key	hash output	RSA	DLOG	EC	
128	256	3072	3072	256	near term
256	512	15360	15360	512	long term

N. Smart (editor): Algorithms, key size and parameters report, Nov. 2014, ENISA

openssl on my Skylake (E3-1270 v5, 4GHz peak), ops/s (unscientific):

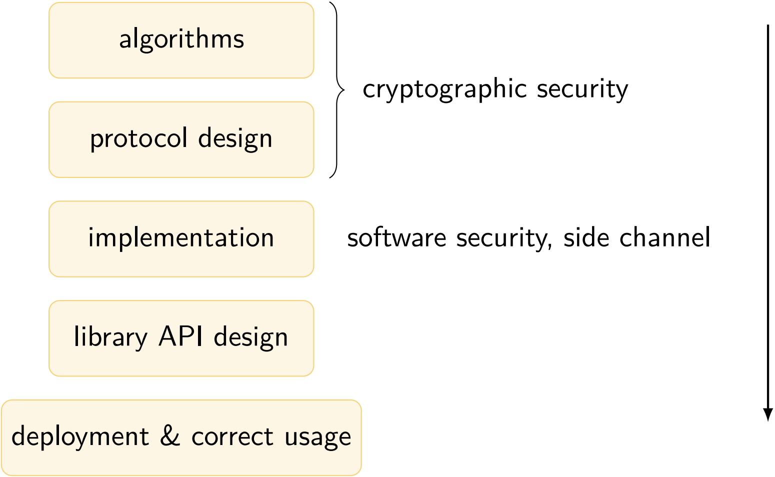
algo	signatures/s	verifications/s
ECDSAp256	33 134	14 952
RSA 2048	1 838	65 028
RSA 4096	278	18 483

Considerations

- different keys for different purposes
- algorithms from competitions: eSTREAM, PHC, AES, SHA, CAESAR
  - e. g. Salsa20, AES
- keys based on passwords: Argon2, scrypt, bcrypt, PBKDF2

In networking, timing is not “just a side channel”. Demand constant-time implementations.

What has to go right



algorithms

protocol design

cryptographic security

implementation

software security, side channel

library API design

deployment & correct usage

inspired by Matthew D. Green, Pascal Junod

Words of caution

limits

- crypto will not solve your problem
- only a small part of a secure system
- don't implement yourself

difficult to solve problems

- trust / key distribution
  - revocation
- ease of use

many requirements remaining

- replay
- timing attack
- endpoint security