Network security

Modern cryptography for communications security

Benjamin Hof
hof@in.tum.de

Lehrstuhl für Netzarchitekturen und Netzdienste
Fakultät für Informatik
Technische Universität München

Cryptography part 2 – 15ws

Outline

Hash functions and private-key cryptography

Public-key setting

Outline

Hash functions and private-key cryptography

Public-key setting

Cryptographic hash functions

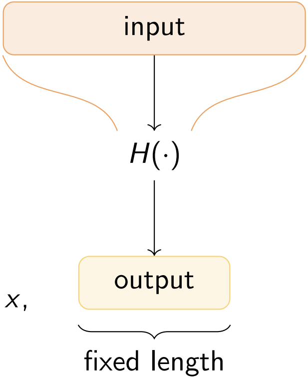
private-key

- encryption
- message authentication codes
- *hash functions*

public-key

...

Hash functions



- variable length input
- fixed length output

provide:

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

Birthday problem

question one

- number of people in a room required
- s. t. P[same birthday as you] ≥ 0.5:

$$1 - (\frac{364}{365}^n) \geq 0.5$$

$\geq$ 253 people necessary.

question two

- number of people in a room required
- s. t. P[at least two people with same birthday] ≥ 0.5

$\approx$ $const \cdot \sqrt{365} \approx 23$.

Birthday problem

question one

- number of people in a room required
- s. t. P[same birthday as you] ≥ 0.5:

$$1 - (\frac{364}{365}^n) \geq 0.5$$

$\geq$ 253 people necessary. Second pre-image

question two

- number of people in a room required
- s. t. P[at least two people with same birthday] ≥ 0.5

$\approx$ $const \cdot \sqrt{365} \approx 23$. Collision

Birthday problem (cont'd)

- collision resitance is the strongest property
  - implies pre-image resistance and second pre-image resistance
- usually broken broken first: MD5, SHA1
- hash function with output size of 128 bit: $\leq 2^{128}$ possible outputs
- finding collisions: $\sqrt{2^{128}} = 2^{64}$
- minimum output size: 256

HMAC

A popular MAC:

- opad is $0x\overline{36}$, ipad is $0x\overline{5C}$
  $tag := H(k \oplus \text{opad} \| H(k \oplus \text{ipad} \| m))$
- use SHA2-512, truncate tag to 256 bits

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

Combining confidentiality and authentication

- encrypt-then-authenticate:
  $c \leftarrow Enc_{k_1}(m), t \leftarrow Mac_{k_2}(c)$
  transmit: $\langle c, t \rangle$
  This is generally secure.
- authenticated encryption
  Also a good choice.
  e. g. offset codebook (OCB), Galois counter mode (GCM)

Recap: private-key cryptography

- attacker power: probabilistic polynomial time
- confidentiality defined as IND-CPA:
  encryption, e. g. AES-CTR$
- message authentication defined as existentially unforgeable under adaptive chosen-message attack:
  message authentication codes, e. g. HMAC-SHA2
- authenticated encryption modes

Outline

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

Public-key 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 private
- less keys required
- orders of magnitude slower

Problems believed to be hard

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

Public-key cryptography

private-key

- encryption
- message authentication codes
- hash functions

public-key

- encryption
- signatures
- key exchange

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

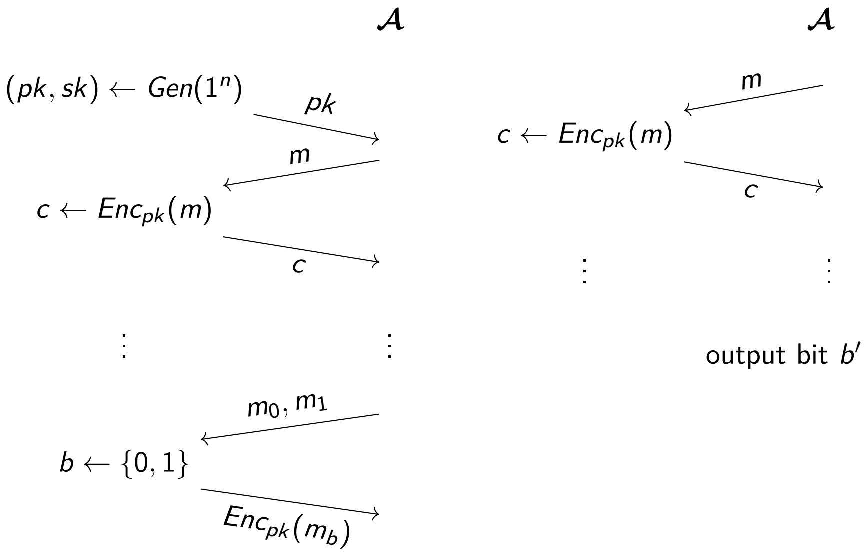
steps

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



Security of RSA

- textbook RSA is deterministic $\rightarrow$ must be insecure against CPA

$\Rightarrow$ 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 belived to be difficult (assumption!)
- secure at least against CPA

armoring ("padding") schemes needed

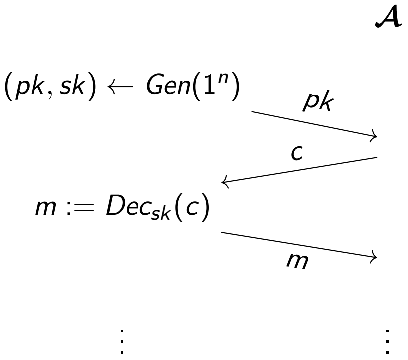
- attacks exist, but used often: PKCS #1 v1.5
- better security: PKCS #1 v2.1/v2.2 (OAEP)

Chosen-ciphertext attack

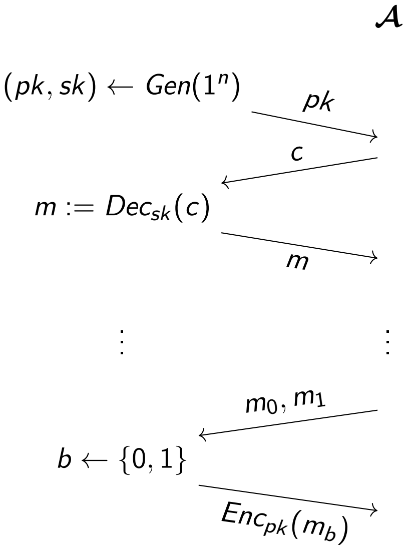
$\mathcal{A}$

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

Chosen-ciphertext attack

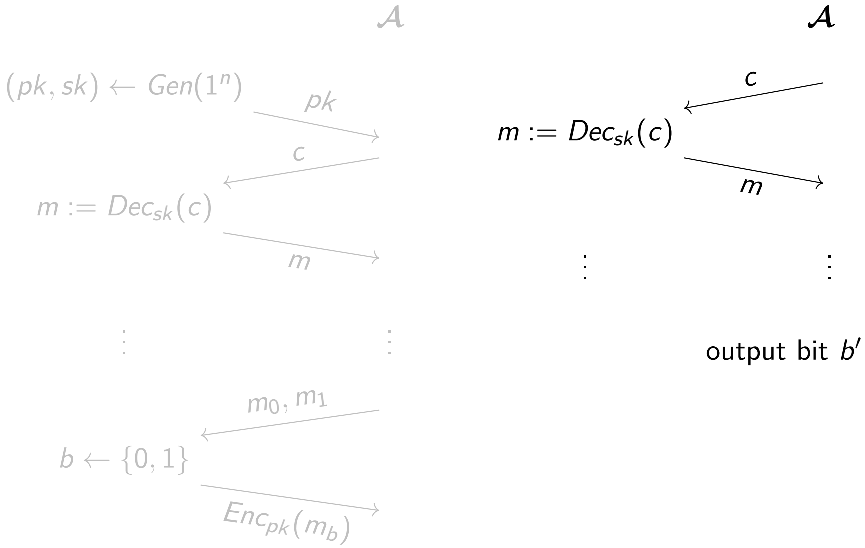


Chosen-ciphertext attack



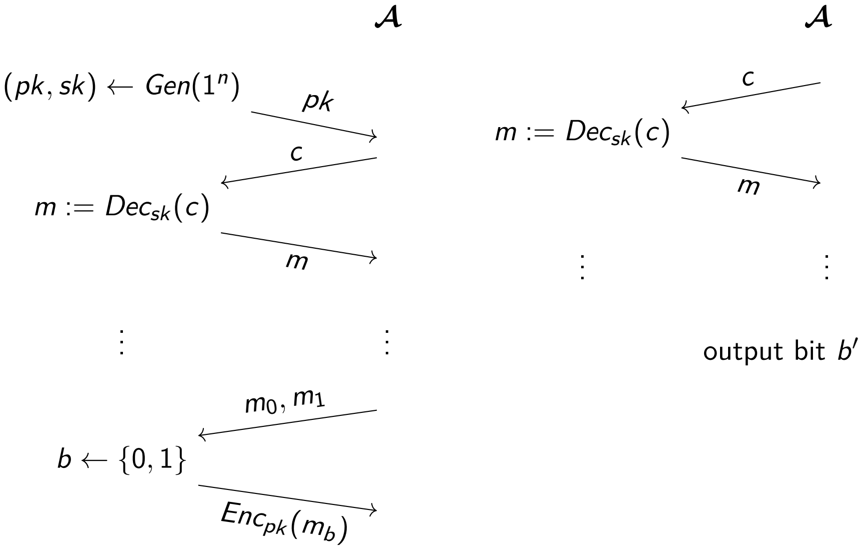
Adversary may not request decryption of $Enc_{pk}(m_b)$ itself.

Chosen-ciphertext attack



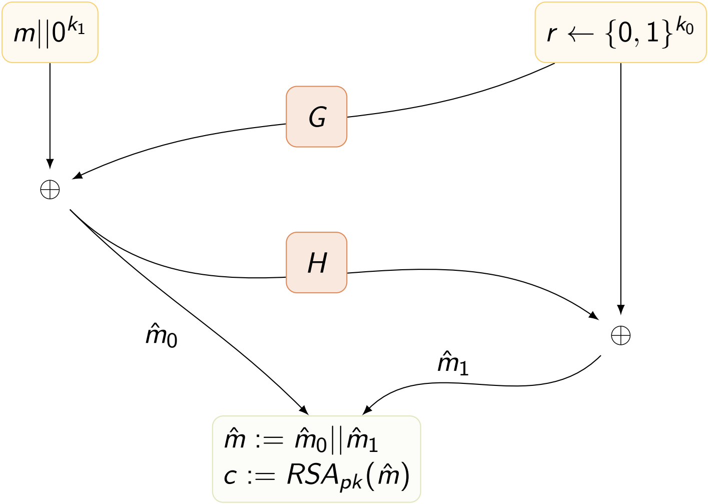
Adversary may not request decryption of $Enc_{pk}(m_b)$ itself.

Chosen-ciphertext attack



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:

$\Rightarrow$ RSA-OAEP secure against CCA

- 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

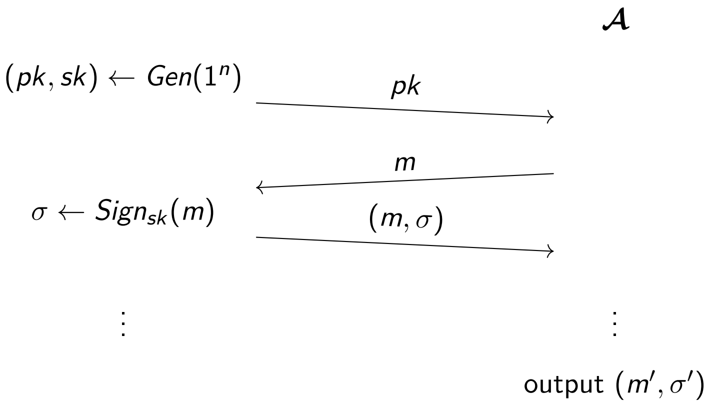
Signatures

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

RSA signatures

- RSA not a secure signature function
- PKCS #1 v1.5
- use RSASSA-PSS

Adaptive chosen-message attack

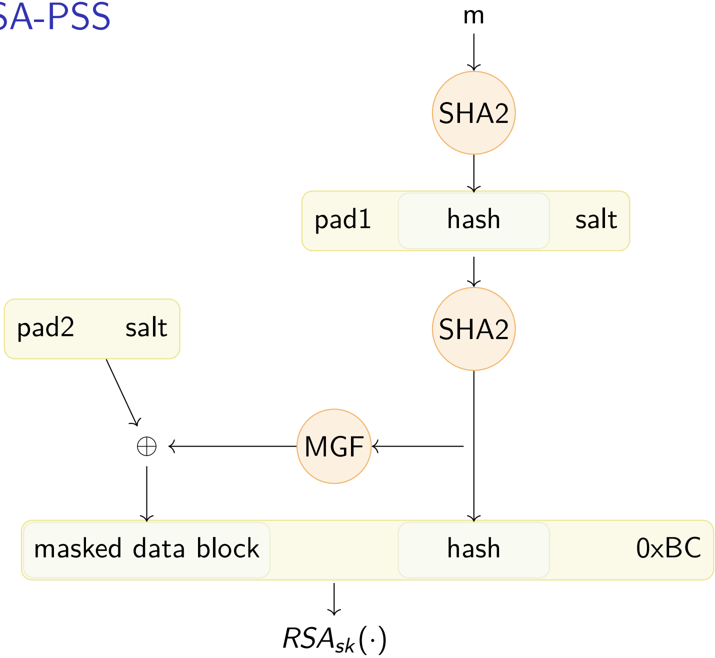


- let $\mathcal{Q}$ be the set of all queries m
- $\mathcal{A}$ succeeds, iff $Vrfy_{pk}(m', \sigma') = 1$ and $m' \notin \mathcal{Q}$

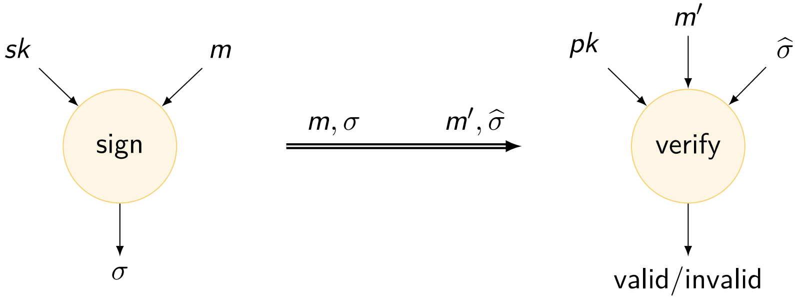
Goal

- 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



$Sign_{sk}(m)$:

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

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

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

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

- negligible probability

Combining signatures and encryption

Goal: S sends message m to R, assuring:

- secrecy
- message came from S

encrypt-then-authenticate

- $\langle S, c, Sign_{sk_S}(c) \rangle$
- attacker A executes CCA: $\langle A, c, Sign_{sk_A}(c) \rangle$

Combining signatures and encryption

Goal: S sends message m to R, assuring:

- secrecy
- message came from S

encrypt-then-authenticate

- $\langle S, c, Sign_{sk_S}(c) \rangle$
- attacker A executes CCA: $\langle A, c, Sign_{sk_A}(c) \rangle$ successful attack

Signcryption cont'd

authenticate-then-encrypt

- $\sigma \leftarrow Sign_{sk_S}(m)$
- $\langle S, Enc_{ek_R}(m||\sigma)\rangle$
- Malicious R to R': $\langle S, Enc_{ek_{R'}}(m||\sigma)\rangle$

Signcryption cont'd

authenticate-then-encrypt

- $\sigma \leftarrow Sign_{sk_S}(m)$
- $\langle S, Enc_{ek_R}(m||\sigma)\rangle$
- Malicious R to R': $\langle S, Enc_{ek_{R'}}(m||\sigma)\rangle$ successful attack

solution for AtE

- compute $\sigma \leftarrow Sign_{sk_S}(m||R)$

Perfect forward security

Assume

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

Idea

- attacker captures private-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 **Bob**

DH_a →

← DH_b

compute s compute s

[store transcript]

$\mathcal{C}$ $\mathcal{A}$

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

$\hat{s}$, transcript →

output b'

Textbook Diffie-Hellman key exchange

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

$a \leftarrow \mathbb{Z}_p$		$b \leftarrow \mathbb{Z}_p$
$X := g^a \bmod p$	$(p,\ g,\ X)$ $\longrightarrow$	
	$\longleftarrow$ Y	$Y := g^b \bmod p$
$s := Y^a \bmod p$ $k := KDF(s)$		$s := X^b \bmod p$ $k := KDF(s)$

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

Elliptic curve Diffie-Hellman key exchange: X25519

- $p = 2^{255} - 19$
- $E(F_p \times F_p)$
- $E : y^2 = x^3 + 486662x^2 + x$

$a \leftarrow \{0,1\}^{255}$ | $b \leftarrow \{0,1\}^{255}$

$A := aG$ → A →

$B := bG$

← B ←

$s := aB$ | $s := bA$

$k := KDF(s_x)$ | $k := KDF(s_x)$

(Other ECDH cryptosystems will need additional verification steps.)

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!

Hybrid approach

Public-key cryptography

- valuable properties
- slow

Hybrid encryption

- protect shared key with public-key cryptography
- protect bulk traffic with private-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 private-key and public-key methods in protocols

e. g.:

handshake

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

transport

- private-key authenticated encryption
- replay protection

Key size equivalents

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

ENISA report, Nov. 2014

openssl on my E5-1630, ops/s (very unscientific):

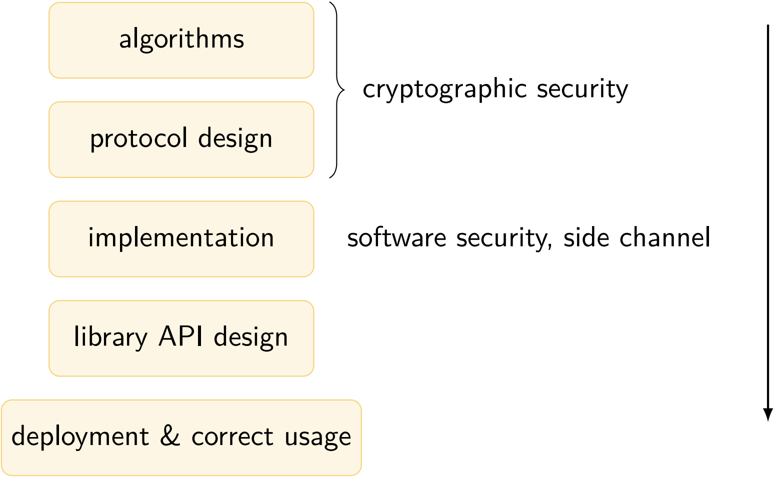
- 175 sig RSA4096
- 1773 sig RSA2048
- 10990 vrfy ECDSAp256

Considerations

- different keys for different purposes
- algorithms from competitions: eSTREAM, PHC, AES, SHA, CAESAR
  - e. g. Salsa20, AES
- keysizes: ENISA, ECRYPT2, Suite B, keylength.com
  - e. g. ECRYPT2: RSA keys $\geq$ 3248 bit
- 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



insipired 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