SECURITY (COMP0141): MODERN CIPHERS

MODERN CIPHERS

two types: **stream** ciphers and **block** ciphers

m="Hi!"  Enc("H"),Enc("i") 

m="Hi!"  Enc("Hi") 

STREAM CIPHERS

initialisation vector

k

IV

STREAM CIPHER (S)

r

arbitrary length pseudorandom stream

m="Hi!"

$IV, s(k,IV) \oplus m$

m_1="What's up?"

$IV_1, s(k,IV_1) \oplus m_1$

STREAM CIPHERS

The randomness s(k,IV) is designed to mimic the randomness in a one-time pad: easier to generate but less secure (**heuristics**)

Like with OTP, if Alice re-uses the same IV then there is no security

Famous examples: ChaCha, Salsa20

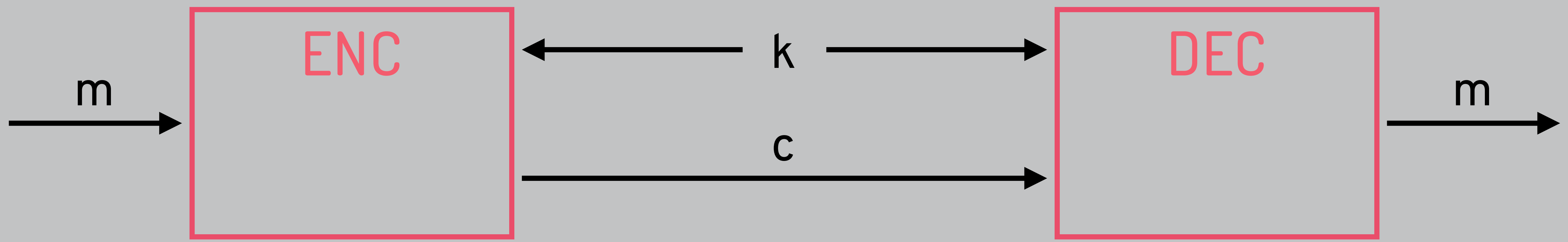
MODERN CIPHERS

two types: **stream** ciphers and **block** ciphers

m="Hi!"  Enc("H"),Enc("i") $\longrightarrow$ 

m="Hi!"  Enc("Hi") $\longrightarrow$ 

BLOCK CIPHERS



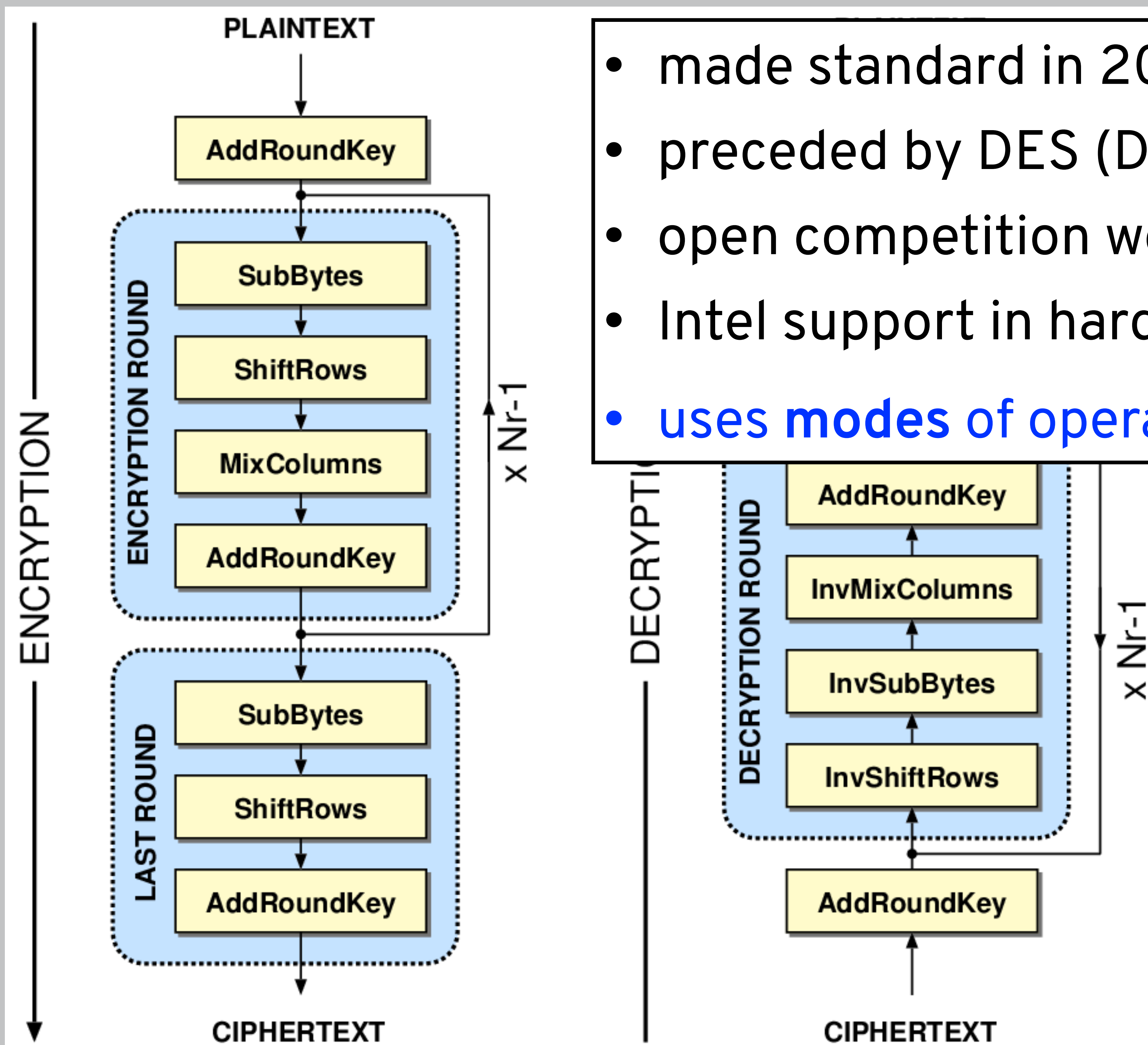
Key is a short random string (128, 192, or 256 bits)

Plaintexts and ciphertexts are short blocks **of the same length** (if plaintext is shorter than key it must be **padded** to match)

Correctness: Dec(k, Enc(k, m)) = m

Security: Without k, Enc acts as a random permutation

ADVANCED ENCRYPTION STANDARD



- made standard in 2001
- preceded by DES (Data Encryption Standard)
- open competition won by Daemen and Rijmen
- Intel support in hardware (⇒ very fast)
- uses **modes** of operation

SYMMETRIC ENCRYPTION

q: what do all these methods have in common?

a: the sender and receiver have the same key.

m=“Hi!”

sk

$\xrightarrow{c=\mathrm{Enc}(sk,m)}$

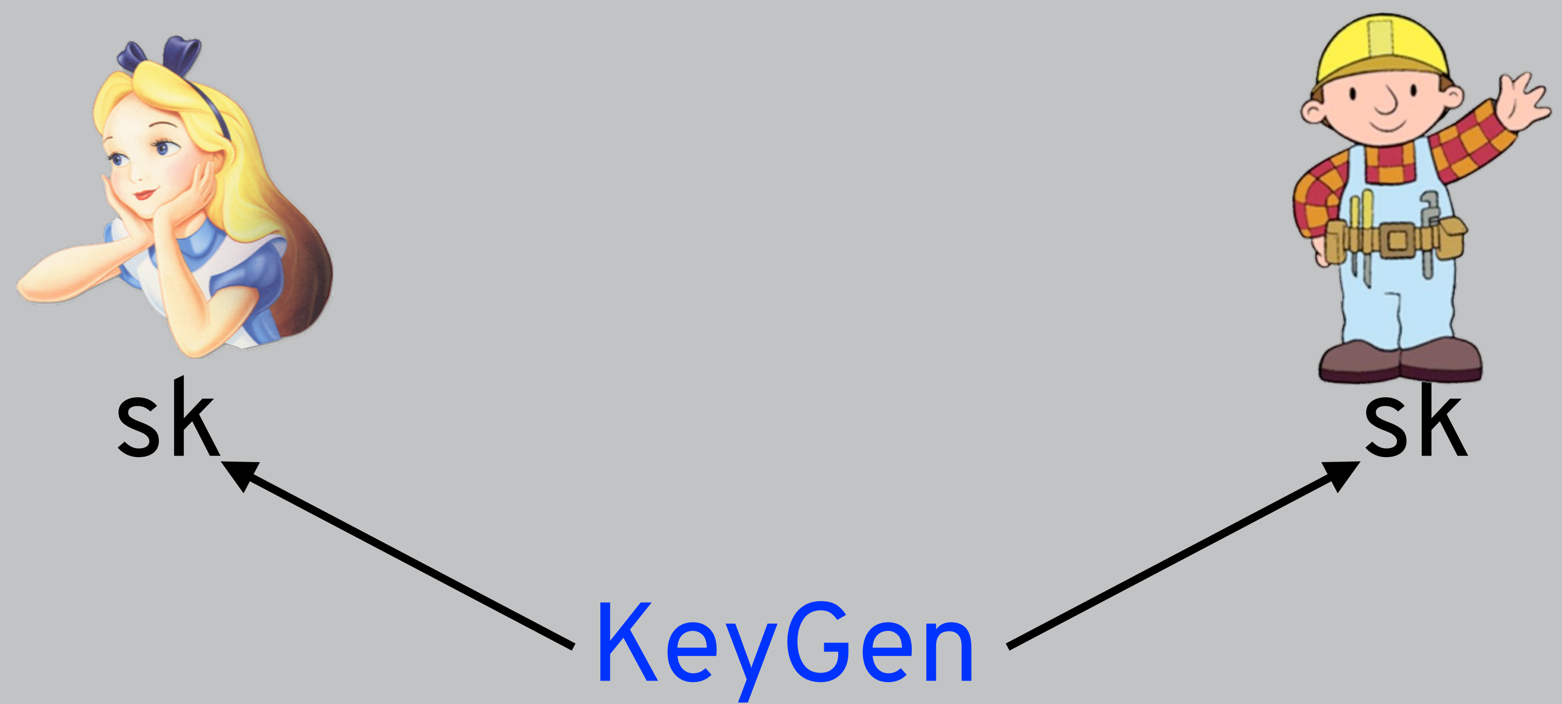
Dec(sk,c)

sk

KEY ESTABLISHMENT

q: how did Alice and Bob agree on that key?

a: key establishment!



ISSUES WITH SHARING KEYS

q: what is the goal of encryption here?

a: to avoid eavesdropping.

m="Hi!"

sk

$c=\mathrm{Enc}(sk,m)$

sk

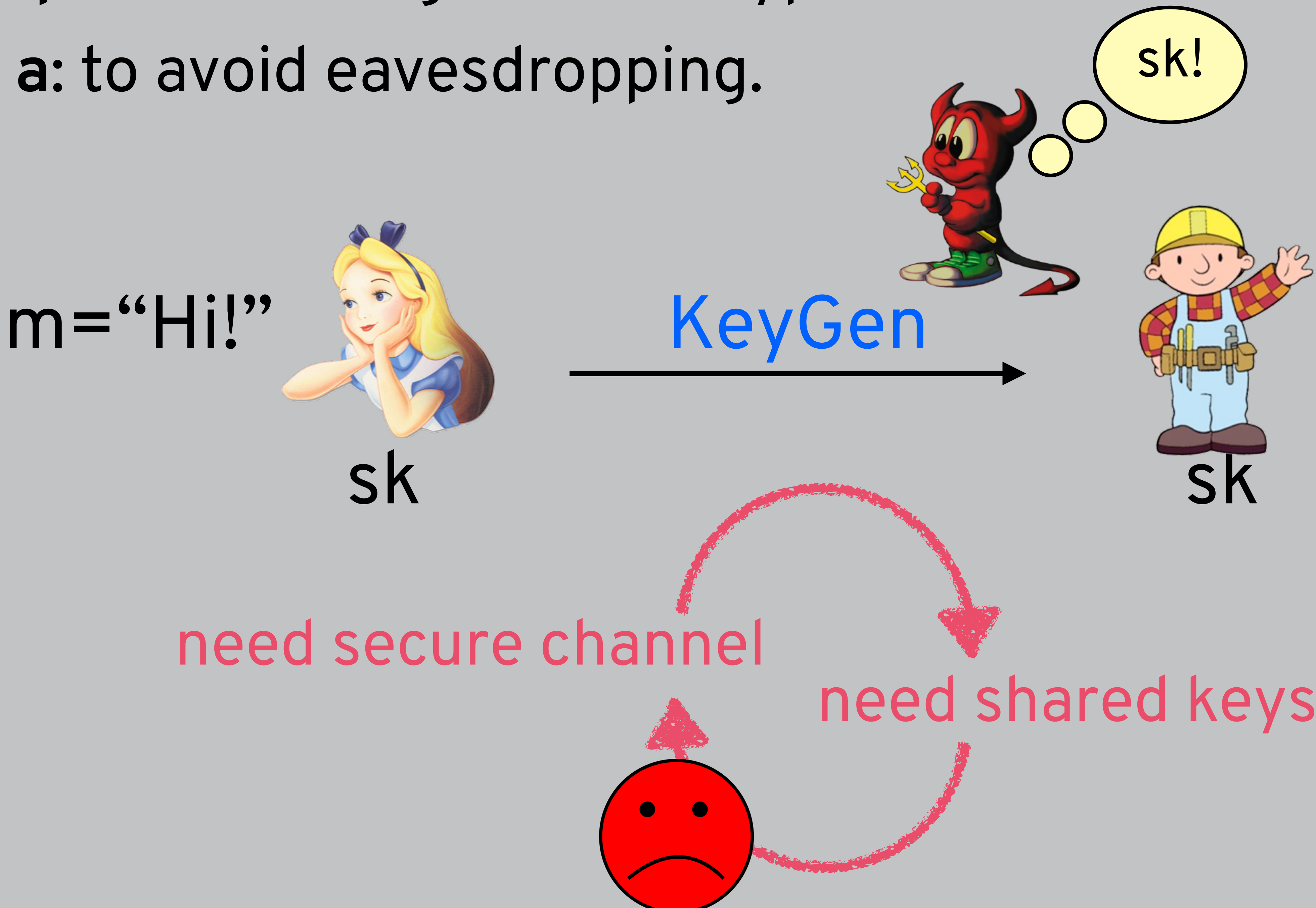
need secure channel

need shared keys

ISSUES WITH SHARING KEYS

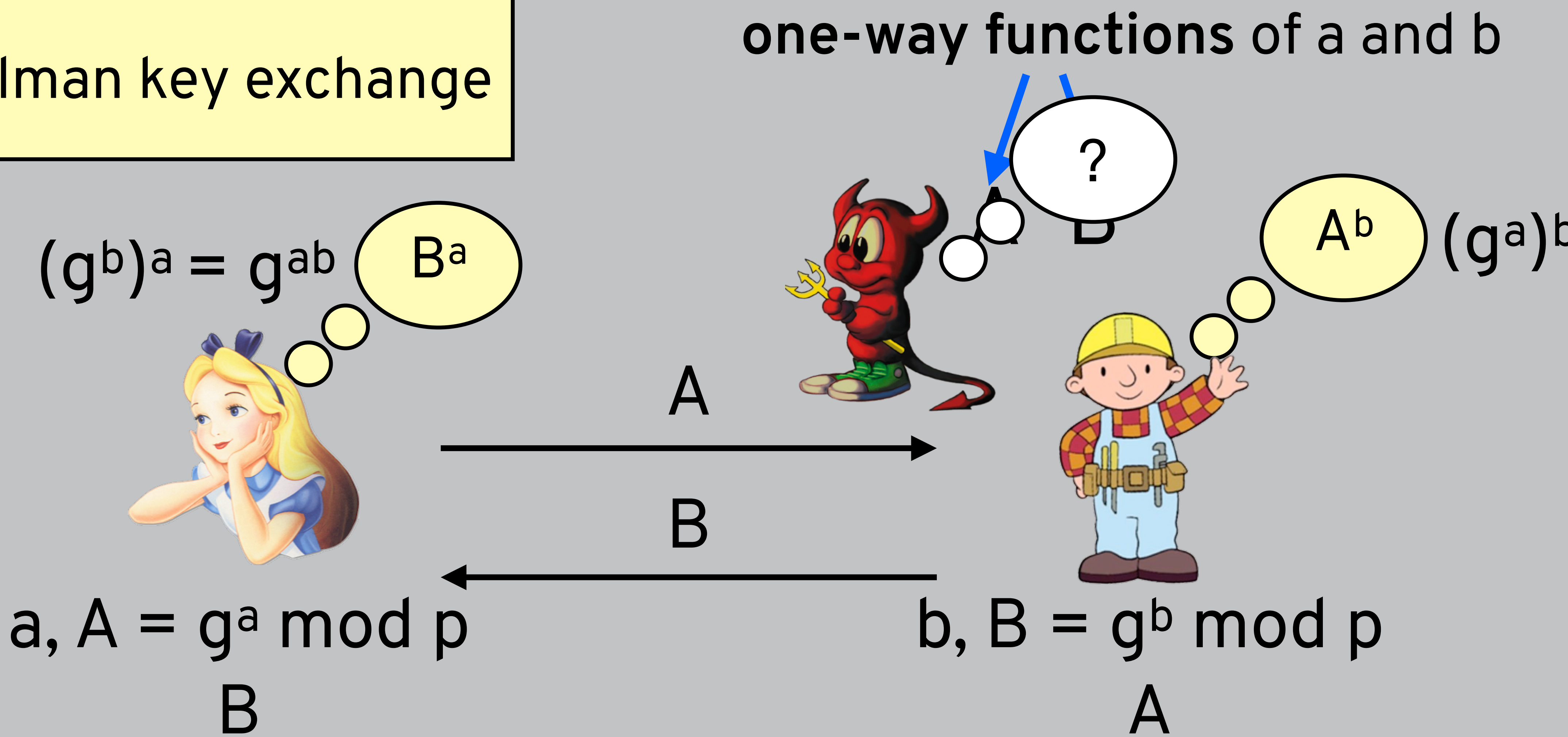
q: what is the goal of encryption here?

a: to avoid eavesdropping.



KEY EXCHANGE

Diffie-Hellman key exchange

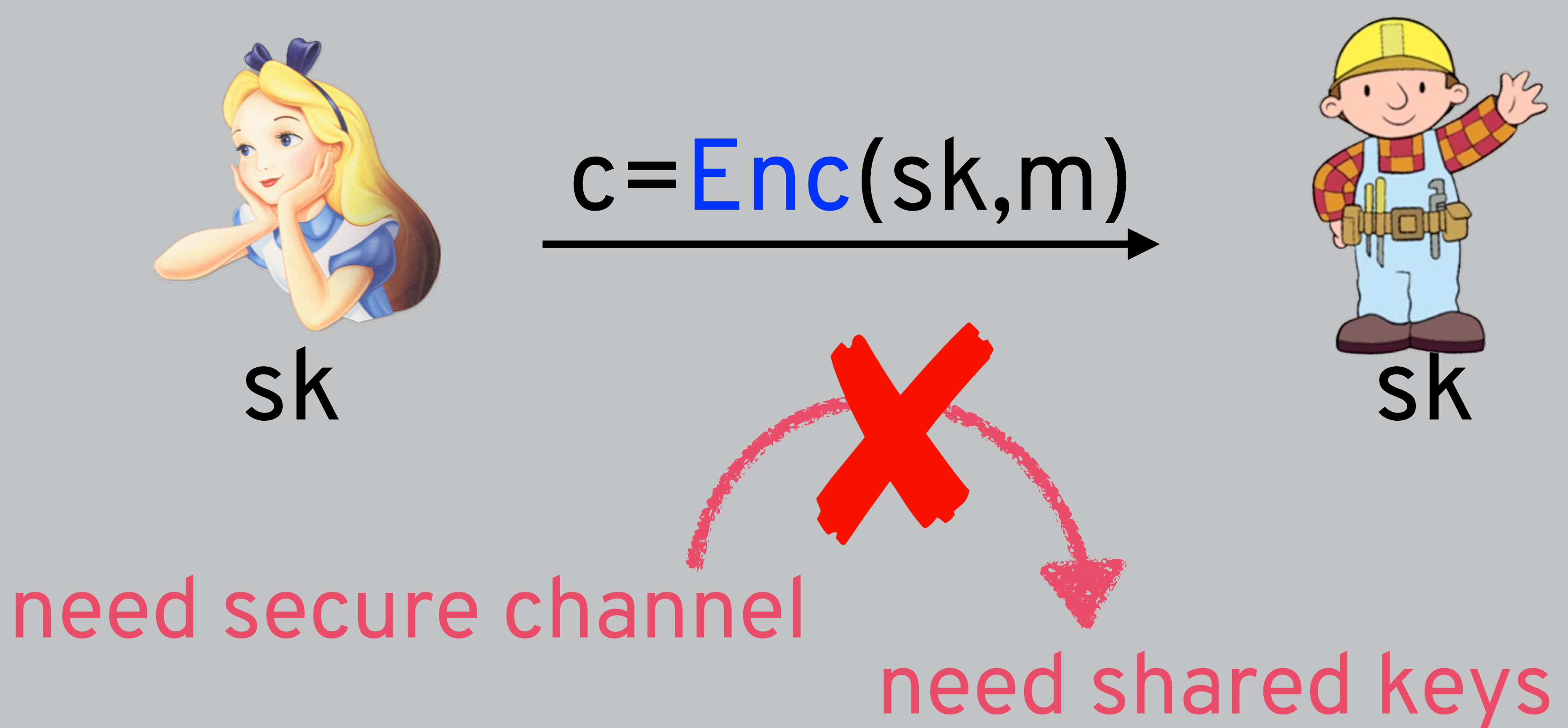


so Alice and Bob agreed on g^{ab}!

ISSUES WITH SHARING KEYS

q: what if I communicate with millions of people?

a: I'll need way too much space to store keys!



PUBLIC-KEY ENCRYPTION

PKI

pk_u

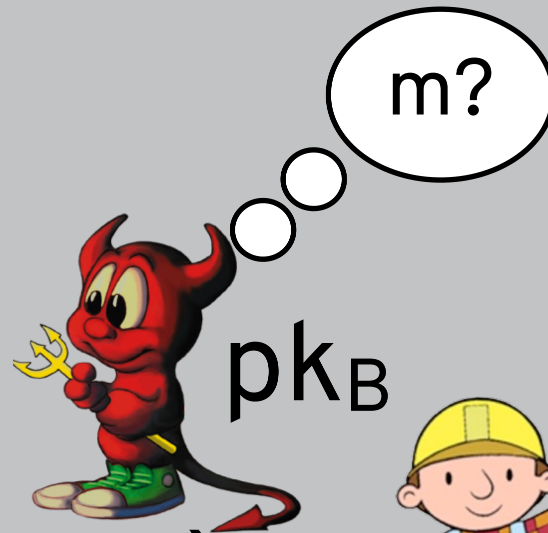
$\vdots$

pk_B

$m=$"Hi!"

pk_B

$c=\mathsf{Enc}(pk_B, m)$ $\longrightarrow$

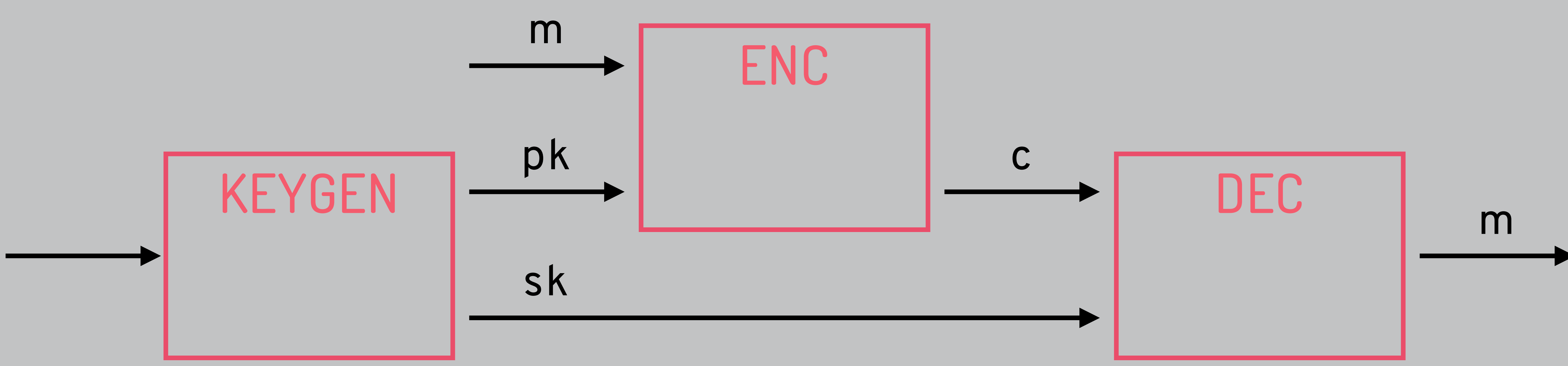


pk_B

$\mathsf{Dec}(sk_B, c)$

$(pk_B, sk_B) \leftarrow \mathsf{KeyGen}(\,)$

PUBLIC-KEY ENCRYPTION



Correctness: For all (pk,sk) produced by KeyGen and messages m, Dec(sk, Enc(pk, m)) = m

Security: ?

THREAT MODEL

Motivation:

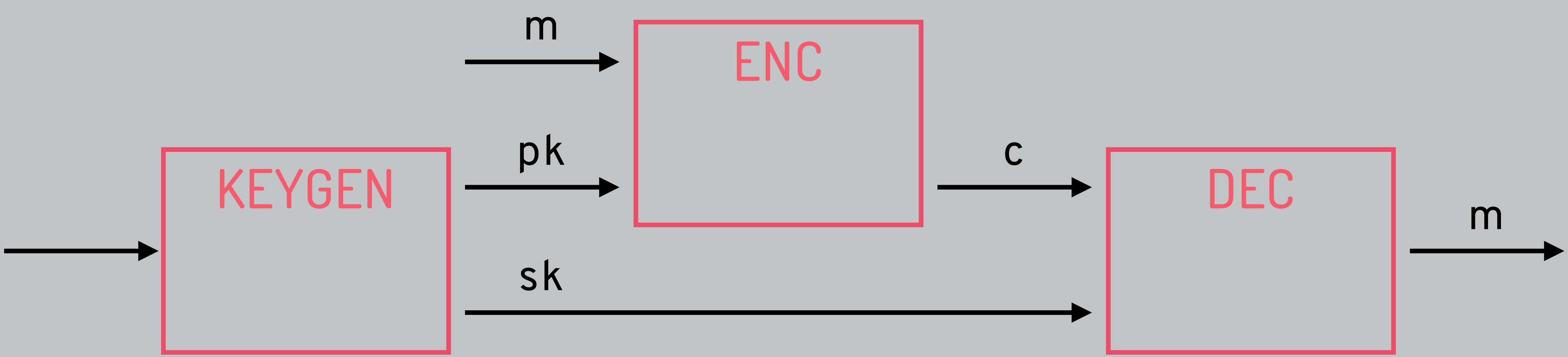
- **Recover key:** learn all future plaintexts
- **Recover plaintext:** learn this specific plaintext
- **Distinguish plaintext:** learn a single bit about plaintext

Capabilities:

- **Known ciphertext:** know ciphertext
- **Known algorithm:** know scheme used to encrypt
- **Known plaintext:** (partial) information about plaintext
- **Chosen plaintext:** adversary picked plaintext
- **Chosen ciphertext:** adversary picked ciphertext

Strongest security statement: the adversary with the strongest capabilities can’t achieve even the weakest goal

IND-CCA SECURITY



Correctness: For all (pk,sk) produced by KeyGen and messages m, Dec(sk, Enc(pk, m)) = m

Security: An adversary who can see decryptions of chosen ciphertexts and can pick two arbitrary plaintexts should not be able to distinguish the encryption of one of them from the encryption of the other (IND-CCA security)

THREAT MODEL

Motivation:

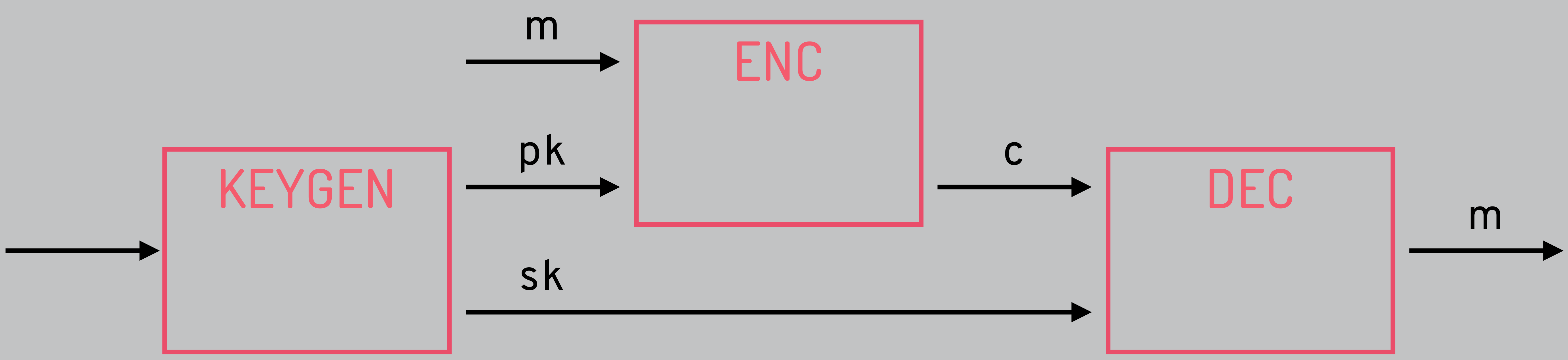
- **Recover key**: learn all future plaintexts
- **Recover plaintext**: learn this specific plaintext
- **Distinguish plaintext**: learn a single bit about plaintext

Capabilities:

- **Known ciphertext**: know ciphertext
- **Known algorithm**: know scheme used to encrypt
- **Known plaintext**: (partial) information about plaintext
- **Chosen plaintext**: adversary picked plaintext
- **Chosen ciphertext**: adversary picked ciphertext

Strongest security statement: the adversary with the strongest capabilities can't achieve even the weakest goal

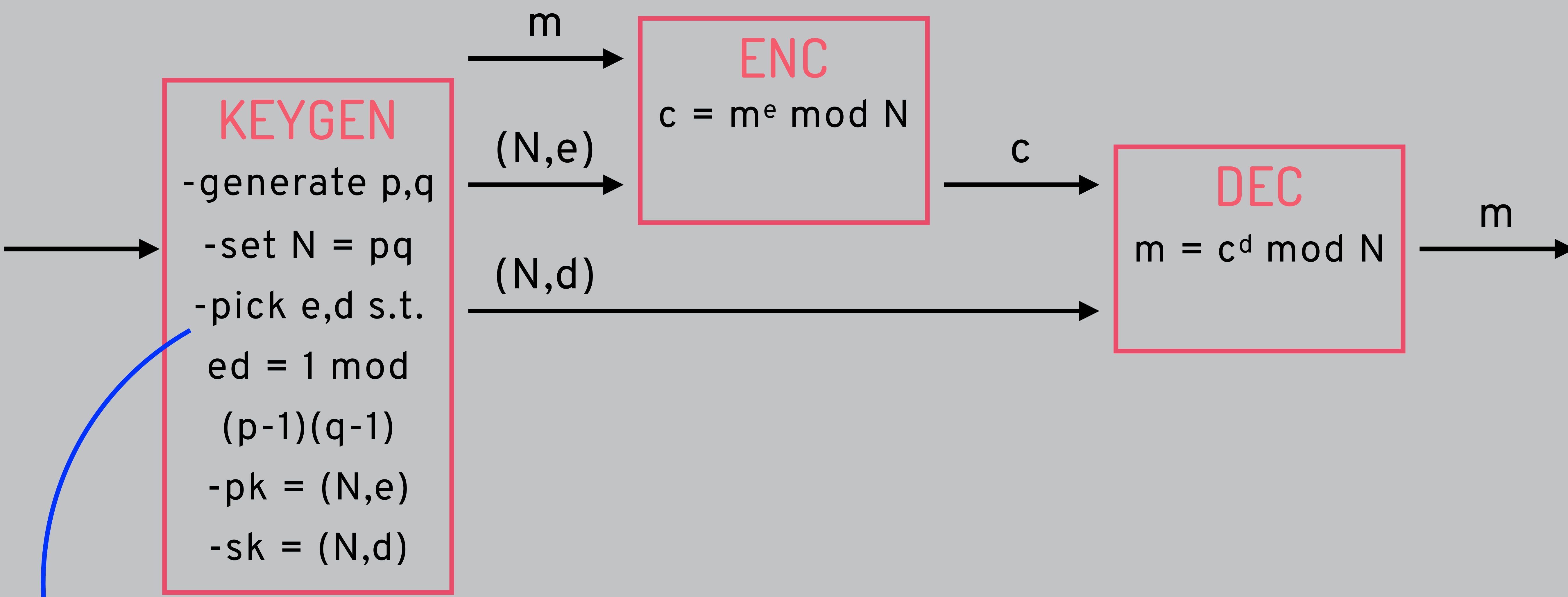
IND-CPA SECURITY



Correctness: For all (pk,sk) produced by KeyGen and messages m, Dec(sk, Enc(pk, m)) = m

Security: An adversary who can pick two arbitrary plaintexts should not be able to distinguish the encryption of one of them from the encryption of the other (IND-CPA security)

TEXTBOOK RSA ENCRYPTION



Popular choices of e are 3 (first odd prime) and 65537 (power of 2 + 1)

CORRECTNESS OF RSA

ENC

$c = m^e \bmod N$

$\xrightarrow{c}$

DEC

$m = c^d \bmod N$

$\xrightarrow{m}$

Correctness: $c^d \bmod N = (m^e)^d \bmod N$

$= m^{ed} \bmod N$

$= m^{1 \bmod (p-1)(q-1)} \bmod N$

$= m^{1 \bmod \varphi(N)} \bmod N$ (because $N = pq$)

$= m^{1 + k\varphi(N)} \bmod N$

$= m*(m^{\varphi(N)})^k \bmod N$

$= m*1^k \bmod N$ (by Euler's theorem)

$= m \bmod N$

SECURITY OF RSA

KEYGEN

-generate p,q

-set N = pq

-pick e,d s.t. ed = 1 mod (p-1)(q-1)

-pk = (N,e)

-sk = (N,d)

$\xrightarrow{(N,e)}$

What if you could factor N?

-compute $\varphi(N) = (p-1)(q-1)$

-compute $d = e^{-1} \bmod (p-1)(q-1)$

-use this to decrypt

This means you can break RSA even if you learn just $\varphi(N)$!

SECURITY OF RSA

KEYGEN

-generate p,q

-set N = pq

-pick e,d s.t.

ed = 1 mod

(p-1)(q-1)

-pk = (N,e)

-sk = (N,d)

(N,e)

How hard is it to factor N?

-Pollard rho has runtime dependent on N

-Lenstra's method dependent on p

-Number field sieve dependent on N

-Quantum computers can do it (but they don't exist yet!)

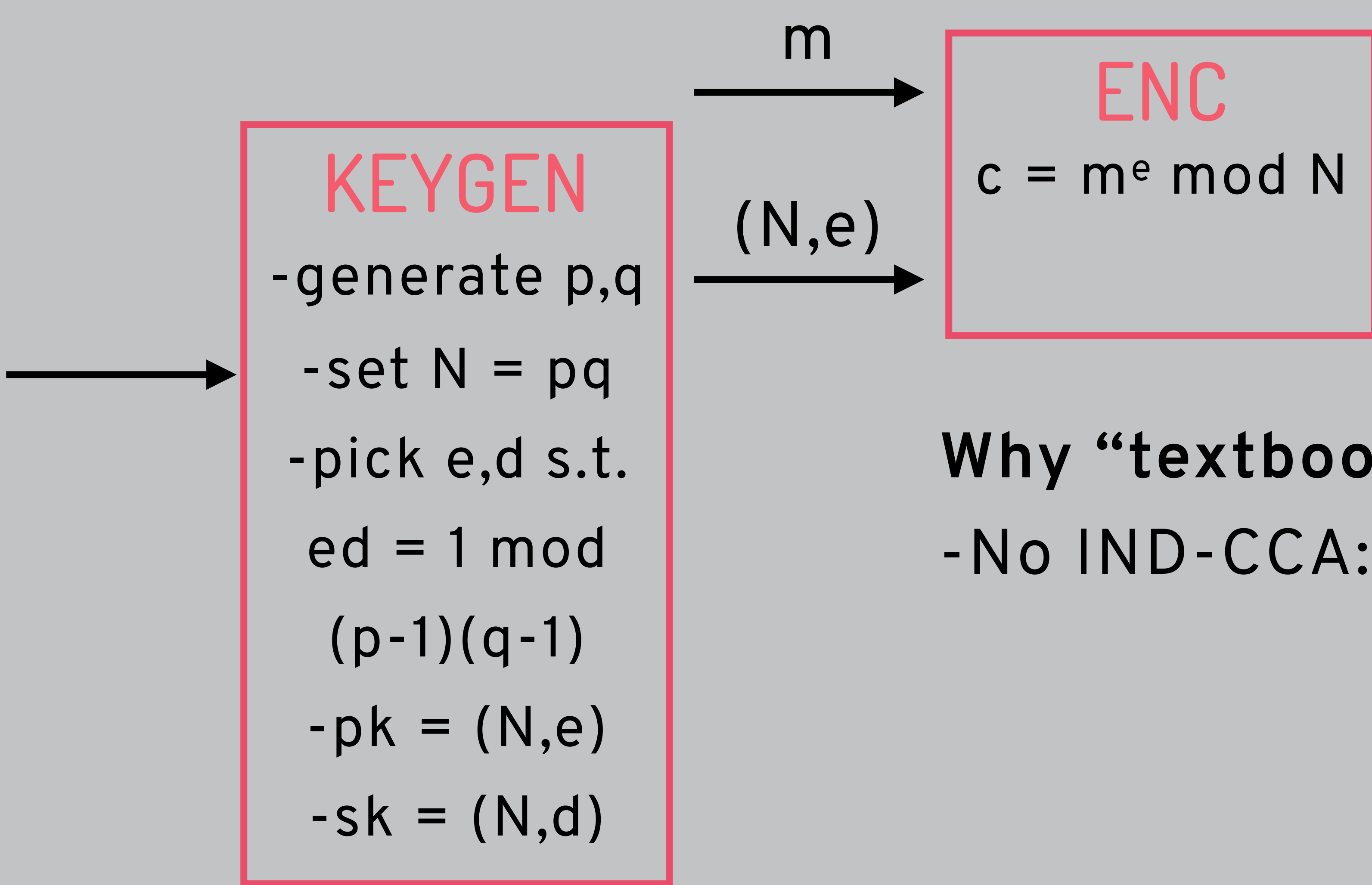
-Other things may come along

-RSA-768 was factored in 2009, took 2000 CPU years (for an average CPU)

-RSA-1024 is now considered dangerous

-RSA-2048 is now considered safe

SECURITY OF RSA



Why "textbook" RSA?

-No IND-CCA: message recovery attack

MESSAGE RECOVERY ATTACK

Given (N,e) and c:

- Compute $c_r = c*r^e \bmod N$

- Get decryption m_r of c_r (chosen ciphertext)

- Compute $m_r*r^{-1} \bmod N = (c*r^e)^d*r^{-1} \bmod N$

$$\begin{aligned}
&= c^d*r^{ed}*r^{-1} \bmod N\\
&= (m^e)^d*r^{ed}*r^{-1} \bmod N\\
&= m^{ed}*r*r^{-1} \bmod N\\
&= m \bmod N
\end{aligned}$$

SECURITY OF RSA

m

ENC

$c = m^e \bmod N$

(N,e)

KEYGEN

- generate p,q
- set $N = pq$
- pick e,d s.t. $ed = 1 \bmod (p-1)(q-1)$
- pk = (N,e)
- sk = (N,d)

Why "textbook" RSA?

- No IND-CCA: message recovery attack
- No IND-CPA: Adversary who can pick m_0 and m_1 can compute $(m_b)^e \bmod N$ for $b \in \{0,1\}$ so can clearly distinguish
- In particular this is because Enc is completely **deterministic** (no randomness to hide value of m)
- In practice we use RSA-OAEP

ENCRYPTION SUMMARY

still need infrastructure

	public-key	secret-key
setup?	no*	yes
basis for security?	math	heuristics
fast?	no!	yes

key exchange

use algebra with very large numbers

so what do we do in practice?