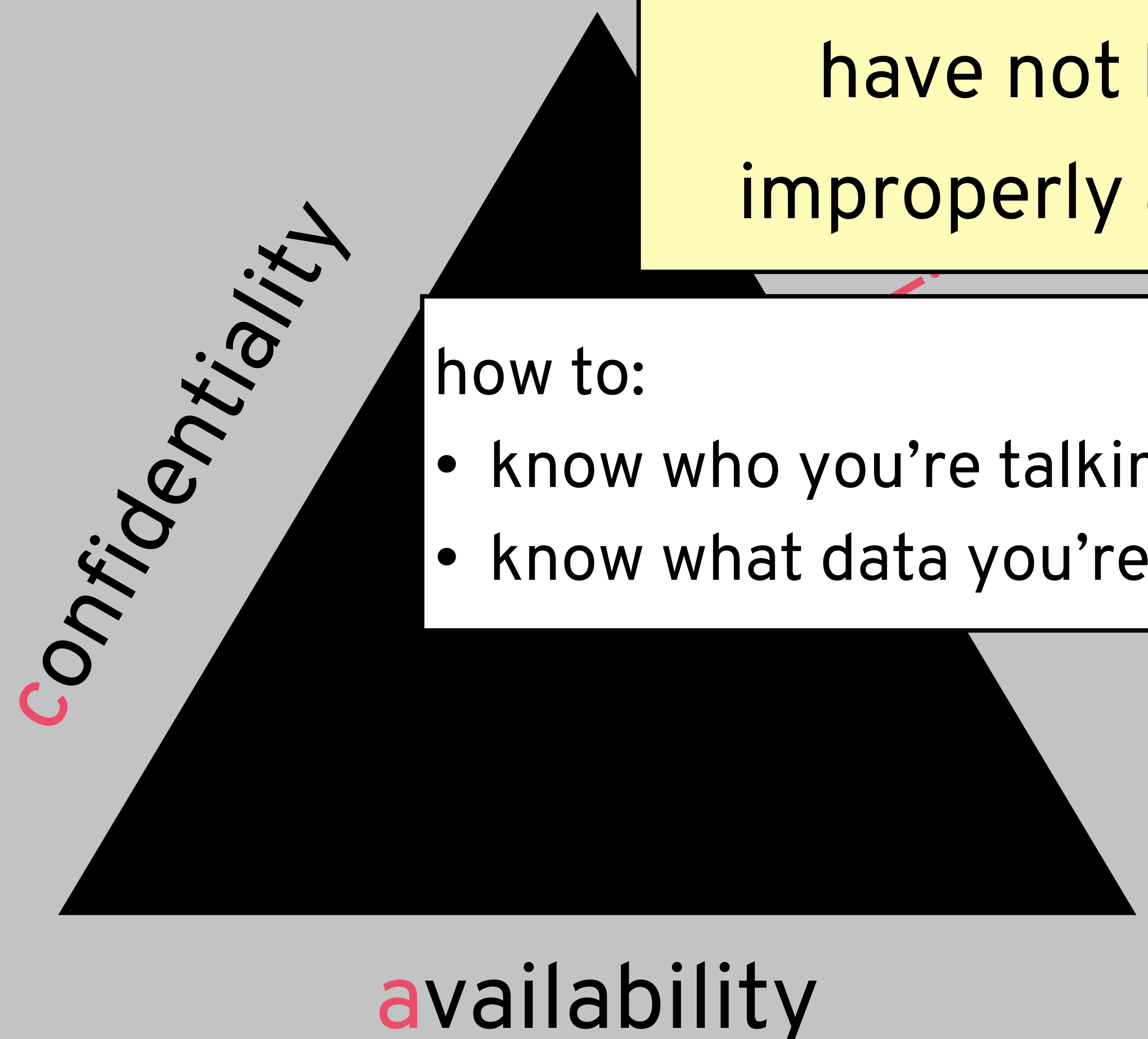
SECURITY (COMP0141): INTEGRITY

INTEGRITY

> “system and data have not been improperly altered”

how to:

- know who you’re talking to?
- know what data you’re getting?



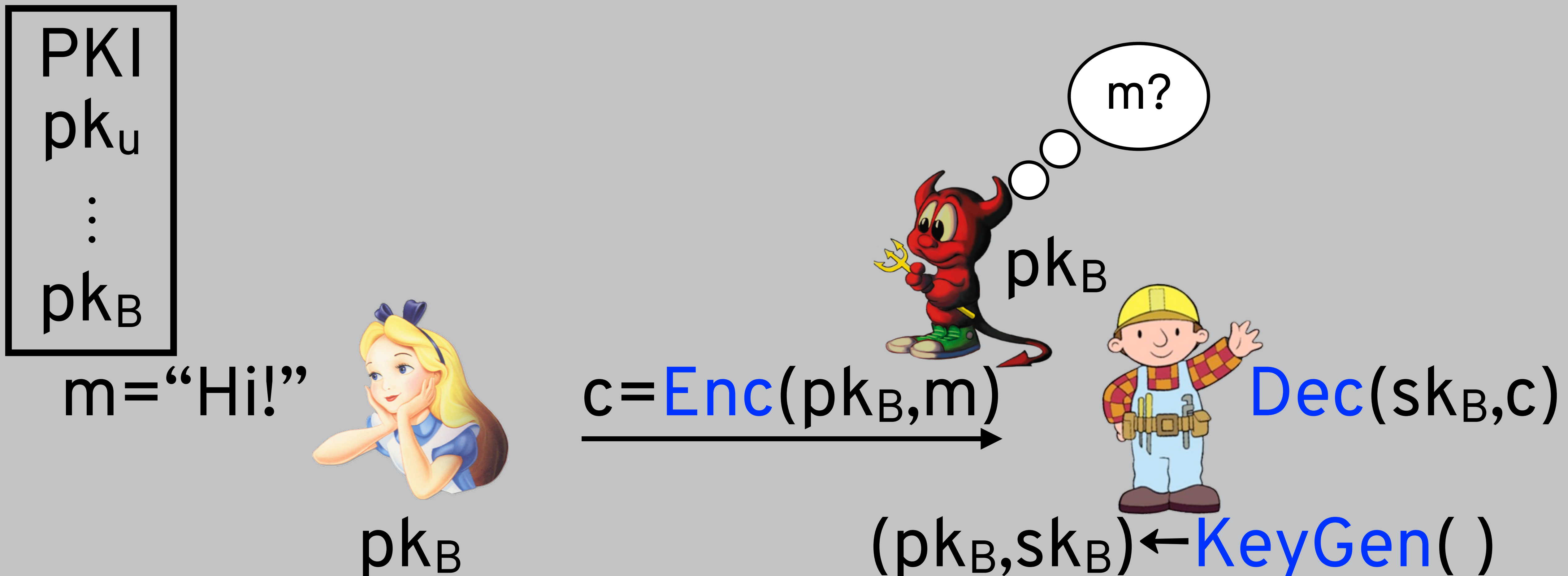
WARNING

You should never design your own cryptography!

This lecture on cryptography does not in any way qualify you to design cryptographic algorithms or protocols

Instead it’s an introduction to what you can expect from cryptography and a feeling for how these algorithms work

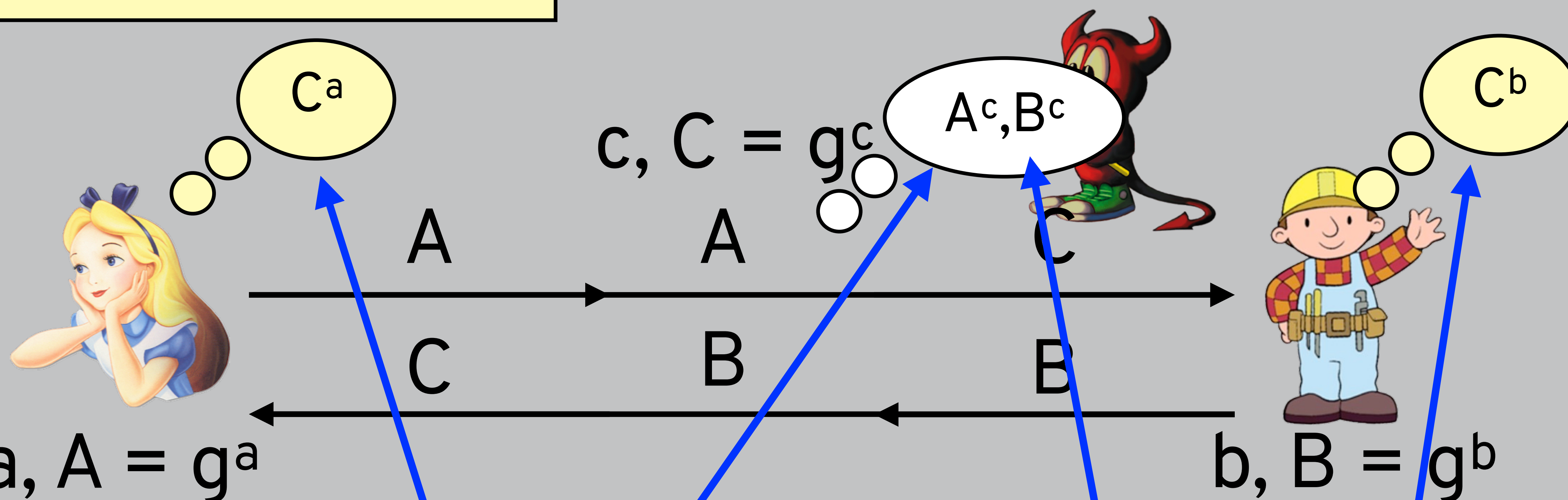
SECRET COMMUNICATION



MAN IN THE MIDDLE (MITM)

Diffie-Hellman key exchange

know who Bob is? or vice versa?



posing as Bob with Alice...

...and as Alice with Bob

for confidentiality, considered **passive** eavesdropper
for integrity, consider more **active** attacker

HOW TO PREVENT SPOOFING?

how do we do this in the physical world?

A “from me, Alice!”

B “from me, Bob!”

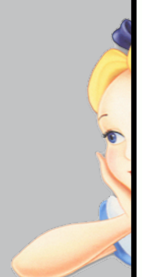
$a, A = g^a$ $b, B = g^b$

this needs to be **unforgeable**

~~A “from me, Alice!”~~

HOW TO PREVENT SPOOFING?

how do we do this in the physical world?



a, A ... $= g^b$

this needs to be **unforgeable**

~~A “from me, Alice!”~~

DIGITAL SIGNATURES

PKI
pk_u
⋮
pk_A

$m=$"Hi!"

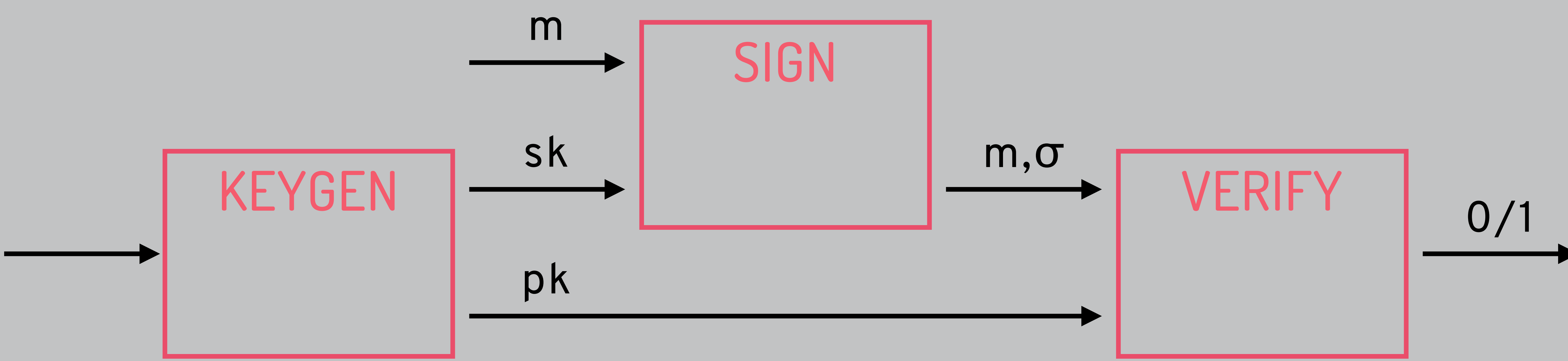
$m,\sigma=$Sign(sk_A,m) →

Verify(pk_A,σ,m)

$(pk_A,sk_A)\leftarrow$KeyGen( )

pk_A

DIGITAL SIGNATURES



Correctness: Valid signatures using valid keys will verify properly (for all k,m and (pk,sk)$\in$[KeyGen(1^k)], Verify(pk,m,Sign(sk,m)) = 1)

Unforgeability (EUF-CMA): For a given public key, an adversary can't produce new signatures that verify ((pk,sk)$\leftarrow$KeyGen(1^k), A gets pk and access to oracle Sign(m), can't output (σ,m) for m not queried to Sign)

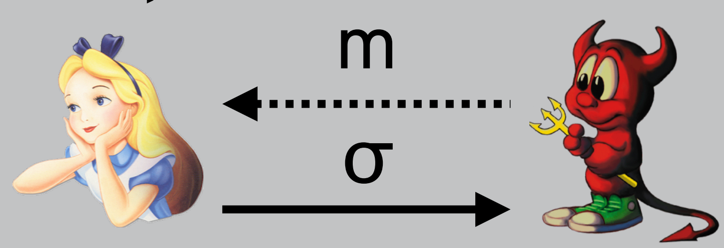
THREAT MODEL FOR SIGNATURES

Motivation:

- **Recover key**: sign all future messages
- **Forge signature**: pretend to be someone else

Capabilities:

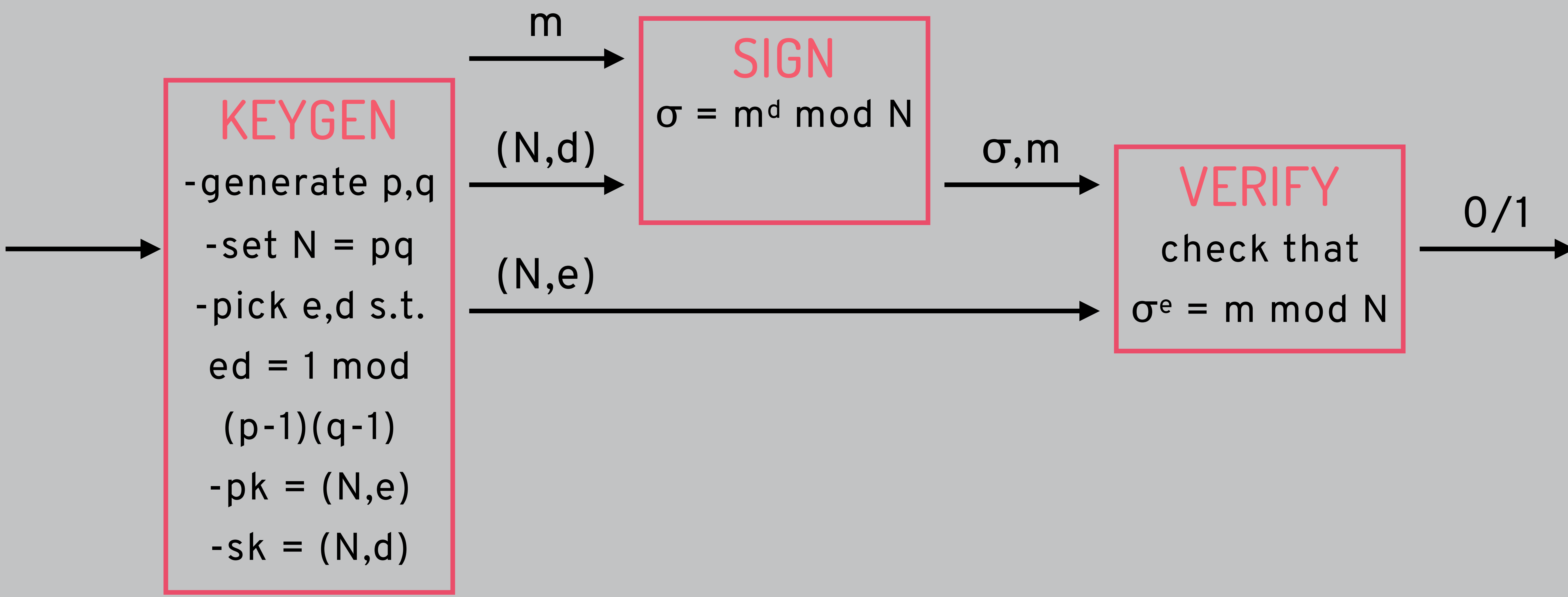
- **Known algorithm**: know scheme used to sign
- **Known signature**: (partial) information about signature



- **Chosen message**: adversary picked messages

Strongest security statement: the adversary with the strongest capabilities can't achieve even the weakest goal (EUF-CMA)

TEXTBOOK RSA SIGNATURES



CORRECTNESS OF RSA

SIGN

$\sigma = m^d \bmod N$

$\xrightarrow{c}$

VERIFY

check that

$\sigma^e = m \bmod N$

$\xrightarrow{m}$

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

$= m^{ed} \bmod N$

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

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

$ed = 1 \bmod (p-1)(q-1)$

-pk = (N,e)

-sk = (N,d)

$\xrightarrow{m}$ $\xrightarrow{(N,d)}$

SIGN

$\sigma = m^d \bmod N$

Why “textbook” RSA?

-No EUF-CMA: if adversary gets signatures σ_1 on m_1 and σ_2 on m_2 then it can create valid signature $\sigma = \sigma_1 * \sigma_2$ on $m_1 * m_2$

-This works because this function $f(m) = m^d$ is homomorphic, so $f(m_1)*f(m_2) = f(m_1*m_2)$

USING DIGITAL SIGNATURES

Diffie-Hellman key exchange

A

B

a, $A = g^a \bmod p$

$\mathbf{sk_A, pk_B}$

b, $B = g^b \bmod p$

$\mathbf{sk_B, pk_A}$

USING DIGITAL SIGNATURES

```
if Verify(pkB,σB,B)
  then sk = B^a
else abort
```

```
if Verify(pkA,σA,A)
  then sk = A^b
else abort
```

$A, \sigma_A = \text{Sign}(sk_A, A)$ →

← $B, \sigma_B = \text{Sign}(sk_B, B)$

$a,\ A = g^a \bmod p$

sk_A, pk_B

$b,\ B = g^b \bmod p$

sk_B, pk_A

USING DIGITAL SIGNATURES

if Verify(pk_B, σ_B, B)
then sk = B^a
else abort

if Verify(pk_A, σ_A, A)
then sk = A^b
else abort

a, $A = g^a \bmod p$

sk_A, pk_B

$\xrightarrow{A, \sigma_A}$

pk_A, pk_B

c, $C = g^c$

$\xrightarrow{C, \sigma}$

b, $B = g^b \bmod p$

sk_B, pk_A

by unforgeability

TRADEOFFS FOR SIGNATURES

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

what's the version of this for signatures?

MESSAGE AUTHENTICATION CODE

m="Hi!"

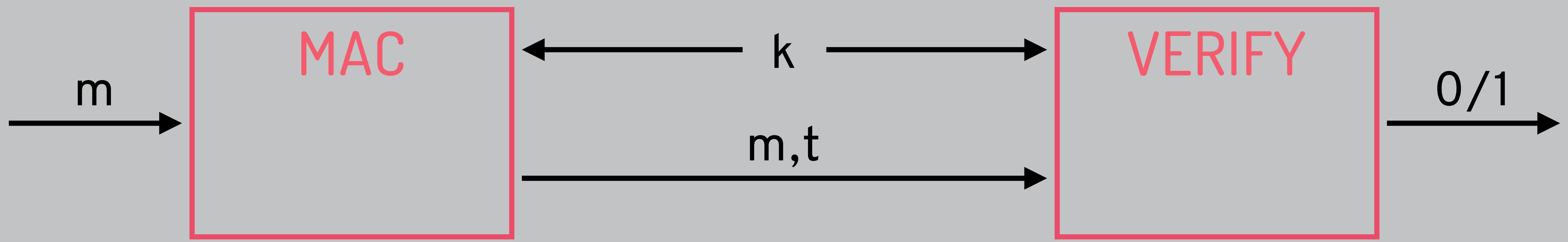
sk

m,t=MAC(sk,m) $\longrightarrow$

sk

Verify(sk,t,m)

MACS

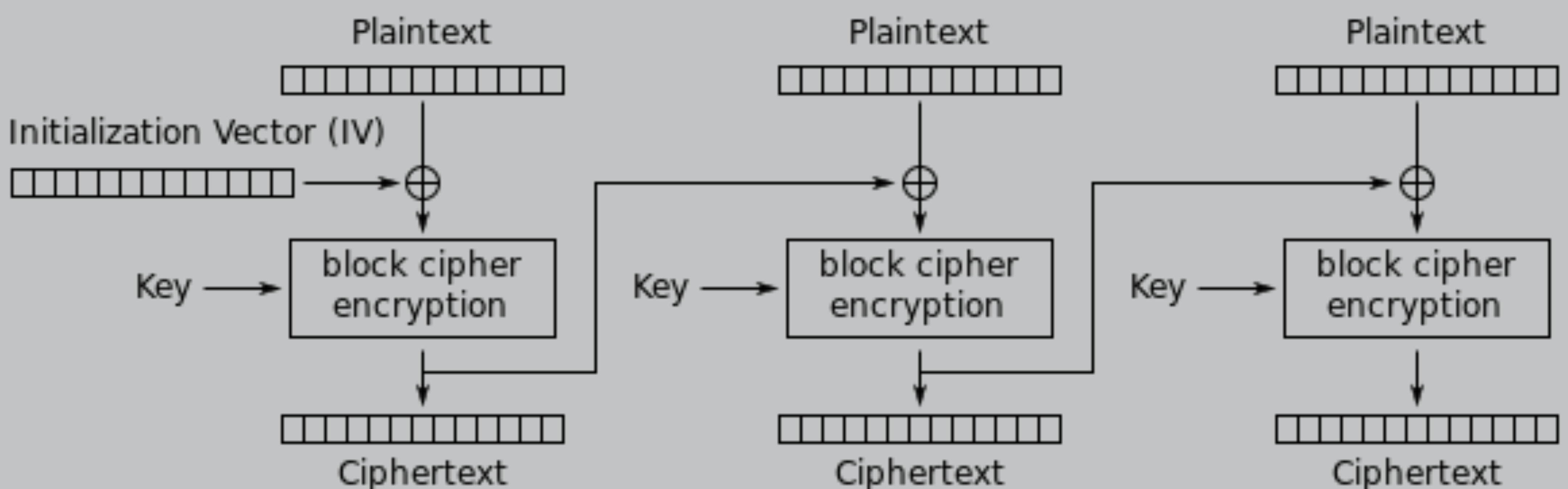


Correctness: Verify(k, m, MAC(k,m)) = 1

Unforgeability: hard to generate (m,MAC(k,m)) without knowing k

MACS FROM AES-CBC

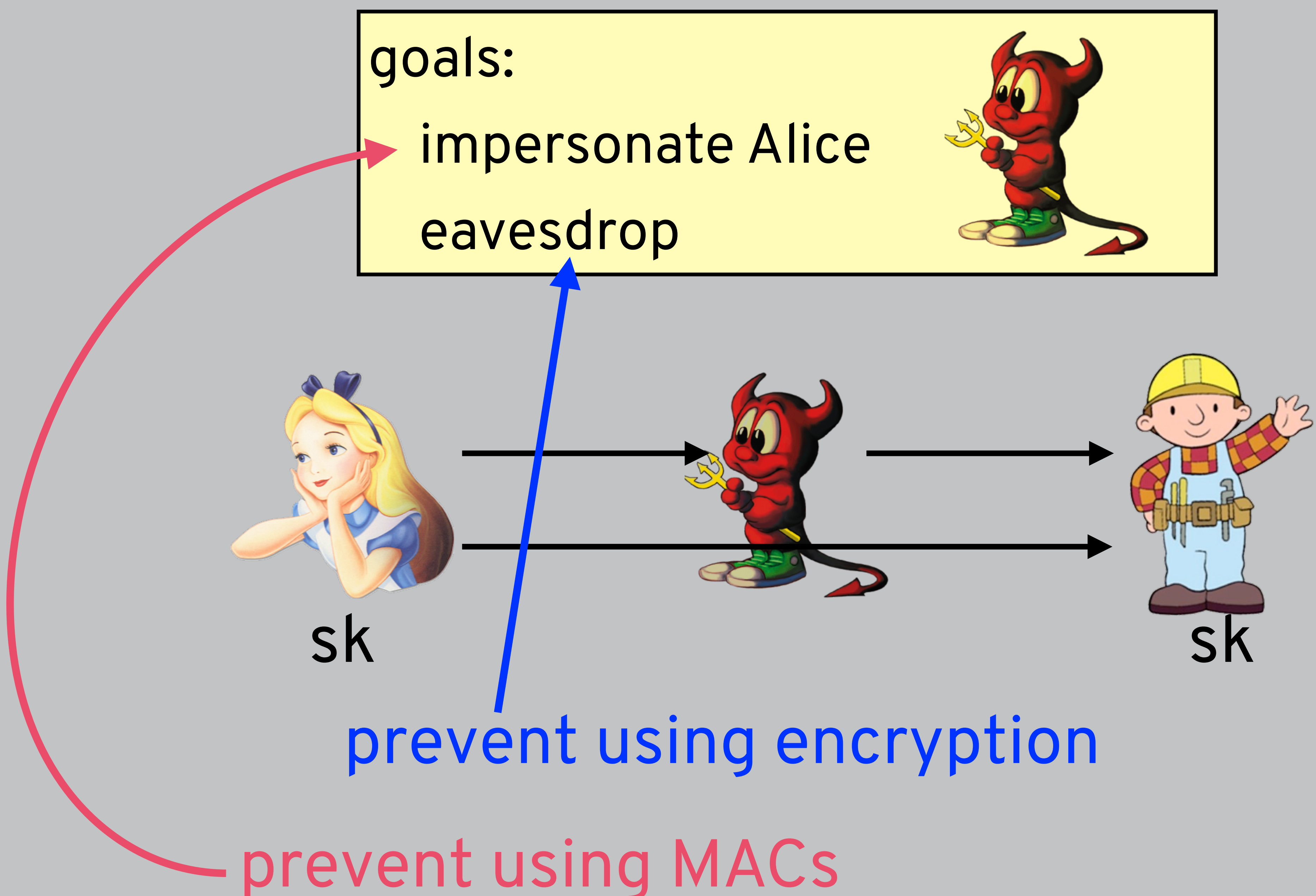
CBC (Cipher Block Chaining) mode: c_0 = IV, $c_i = \mathrm{Enc}(k, m_i \oplus c_{i-1})$



Cipher Block Chaining (CBC) mode encryption

Can use last block of this as a MAC: $\mathrm{MAC}(k, (m_1,\ldots,m_n)) = c_n$ using fixed IV for c_0, Verify(k, m, t) recomputes MAC and checks equality with t

AUTHENTICATED ENCRYPTION (AEAD)



THREAT MODEL FOR AEAD

Motivation:

- **Recover key:** learn all future plaintexts
- **Recover plaintext:** learn this specific plaintext
- **Distinguish plaintext:** learn a single bit about plaintext
- **Forge plaintext:** ciphertext decrypts to plaintext never encrypted by the sender (INT-PTXT)

Capabilities:

- **Known algorithm:** know schemes used to encrypt/MAC
- **Known ciphertext:** (partial) information about ciphertext
- **Chosen message:** adversary picked messages
- **Chosen ciphertext:** adversary picked ciphertexts

CONSTRUCTING AEAD

Encrypt-and-MAC (E&M)

m="Hi!"

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

$t=\mathrm{MAC}(sk,m)$

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

$\mathrm{Verify}(sk,t,m)$

Encrypt-then-MAC (EtM)

m="Hi!"

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

$t=\mathrm{MAC}(sk,c)$

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

$\mathrm{Verify}(sk,t,c)$

MAC-then-Encrypt (MtE)

m="Hi!"

$\mathrm{Enc}(sk,m||\mathrm{MAC}(sk,m))$

$m||t=\mathrm{Dec}(sk,c)$

$\mathrm{Verify}(sk,t,m)$

GALOIS COUNTER MODE (GCM)

Galois Counter Mode: achieving AEAD with block ciphers