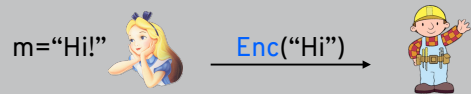
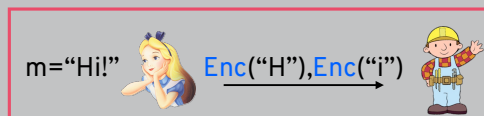


# SECURITY (COMP0141): MODERN CIPHERS



## MODERN CIPHERS

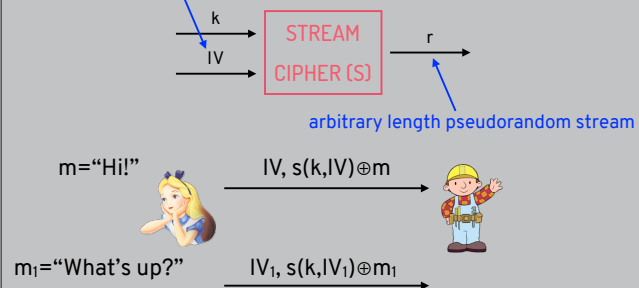
two types: **stream** ciphers and **block** ciphers



Modern ciphers can be broken into two types

## STREAM CIPHERS

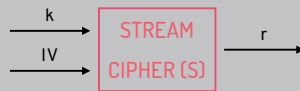
initialisation vector



3

Stream ciphers are most useful in applications where the length of the message isn't known upfront. The sender then uses the stream cipher to generate a pseudorandom string that masks the message

## 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

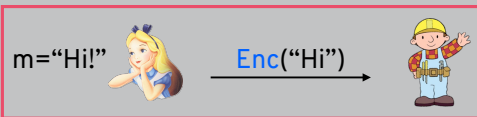
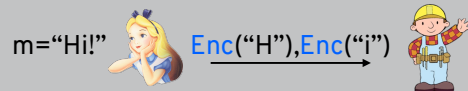
Famous examples: ChaCha, Salsa20

4

Like a OTP but cheaper and not as secure (because the randomness isn't perfect)

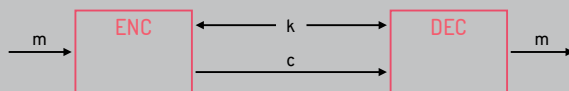
## MODERN CIPHERS

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



5

## BLOCK CIPHERS



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

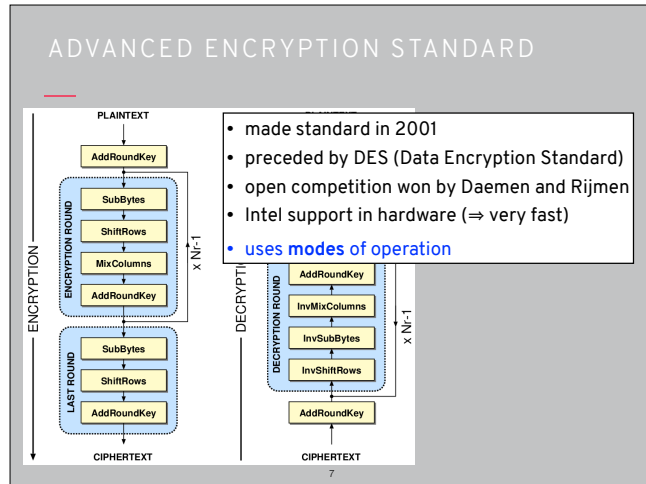
Plaintexts and ciphertexts are short blocks **of the same length** (if plaintext is shorter than key it must be **padded** to match)

**Correctness:**  $\text{Dec}(k, \text{Enc}(k, m)) = m$

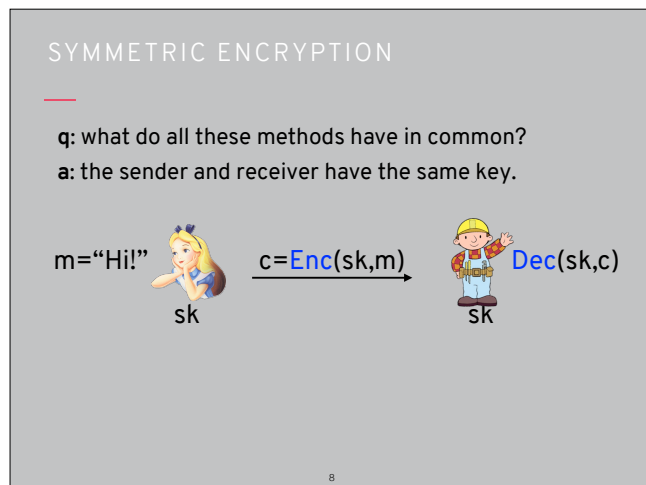
**Security:** Without k, Enc acts as a **random permutation**

6

Block ciphers break plaintext into block matching the key size and satisfy two important properties: correctness and security



The current block cipher standard is called AES

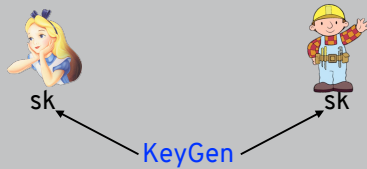


All of these are an example of symmetric encryption, since Alice and Bob have the same information (a secret key)

## KEY ESTABLISHMENT

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

a: key establishment!



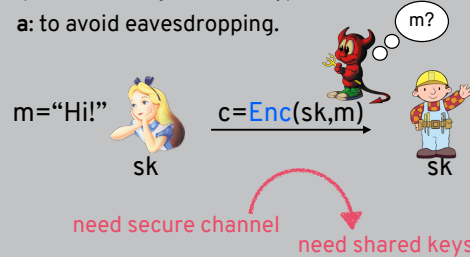
9

How did they agree on the key though? This is something called key establishment, or key exchange

## ISSUES WITH SHARING KEYS

q: what is the goal of encryption here?

a: to avoid eavesdropping.



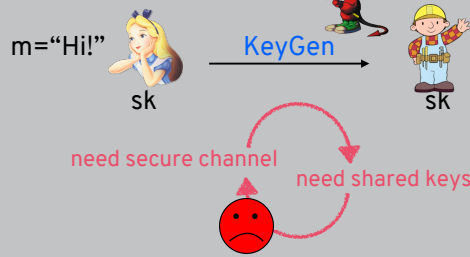
10

If Alice wants to send a secret message to Bob, need some way to secure the communication channel. The way we've seen that is they share a key, so we need shared keys to establish a secure channel

## ISSUES WITH SHARING KEYS

q: what is the goal of encryption here?

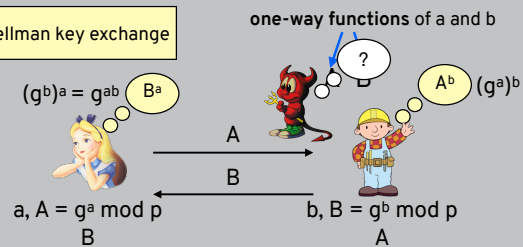
a: to avoid eavesdropping.



But how do we established shared keys if the adversary can hear everything we're saying? Seems like we need a secure channel!

## KEY EXCHANGE

Diffie-Hellman key exchange



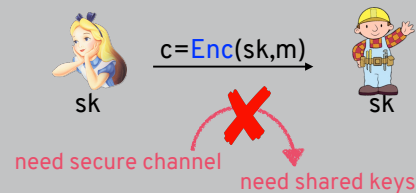
so Alice and Bob agreed on  $g^{ab}$ !

Using this, we can establish keys using Diffie-Hellman key exchange. This is the topic that really kickstarted the modern era of cryptography

## ISSUES WITH SHARING KEYS

q: what if I communicate with millions of people?

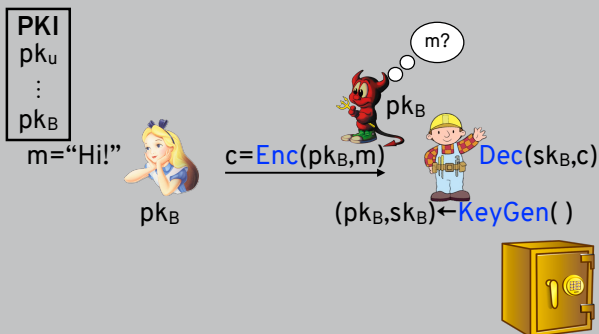
a: I'll need way too much space to store keys!



13

It turns out though that we don't actually need shared keys to secure channels

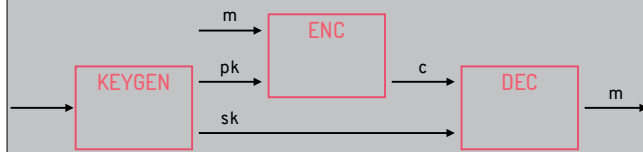
## PUBLIC-KEY ENCRYPTION



14

Can use public-key encryption instead, in which Alice (or anyone) can use a public key to encrypt messages to Bob, and then Bob can use a corresponding secret key to decrypt them. Can think of it like a safe: anyone can lock it by closing the door, but only people with the key can open it

## PUBLIC-KEY ENCRYPTION



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

**Security:** ?

15

A little more formally, here's what PKE looks like. What's the right security notion though?

## 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

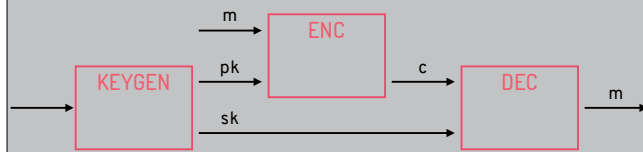
Strongest security statement: the adversary with the strongest capabilities can't achieve even the weakest goal

16

Remember what we said at the beginning



## 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](#))

17

If we take the strongest capability (the adversary can pick both the plaintexts and the ciphertexts) and the weakest goal (learning even a single bit about the message) then we end up with a strong notion of security called IND-CCA (short for INDistinguishability against Chosen Ciphertext Attacks)

## THREAT MODEL

### Motivation:

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



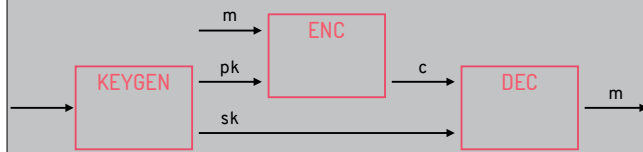
### Capabilities:

- **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

18

## IND-CPA SECURITY



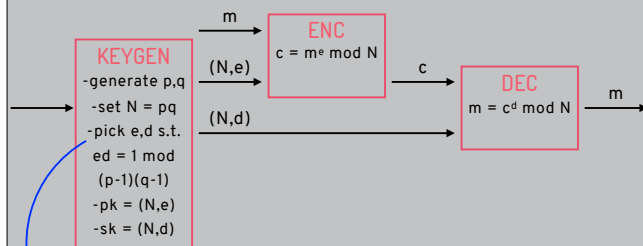
**Correctness:** For all  $(pk, sk)$  produced by KeyGen and messages  $m$ ,  $Dec(sk, Enc(pk, m)) = m$

**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**)

19

If we take the next strongest capability (the adversary can pick the plaintexts) and the weakest goal (learning even a single bit about the message) then we end up with a strong notion of security called IND-CPA (short for INDistinguishability against Chosen Plaintext Attacks)

## TEXTBOOK RSA ENCRYPTION

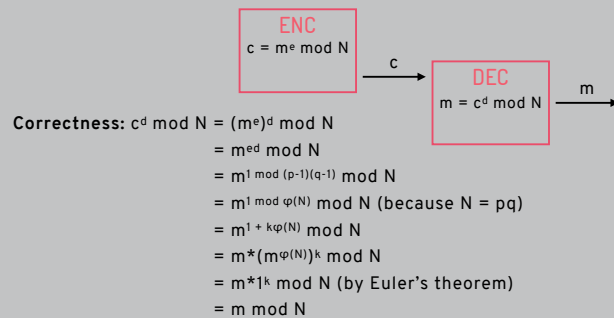


Popular choices of  $e$  are 3 (first odd prime) and 65537 (power of 2 + 1)

20

RSA is one of the most famous examples of a public-key encryption scheme. It is named after its creators: Rivest, Shamir, and Adleman

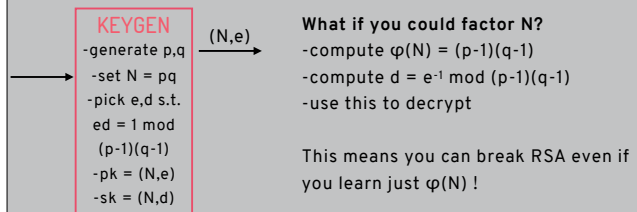
## CORRECTNESS OF RSA



21

Its correctness follows from some of the mathematical properties and theorems that we saw earlier

## SECURITY OF RSA



22

RSA becomes completely insecure if someone can factor  $N$

## 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 =  $(N, d)$

(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

23

Luckily this is a very well-studied problem and seems to be hard for now, although this may of course change in the future. For now the recommended modulus size is 2048 bits

## 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 =  $(N, d)$

m

(N,e)

### ENC

$$c = m^e \bmod N$$

### Why "textbook" RSA?

- No IND-CCA: [message recovery attack](#)

24

The version we've seen is not the real version used in practice, because it's not actually very secure

## MESSAGE RECOVERY ATTACK

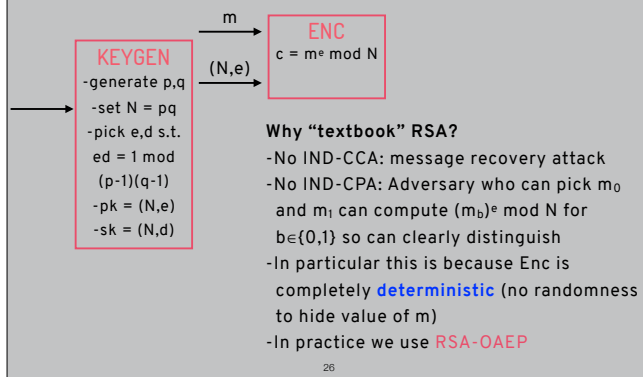
Given  $(N, e)$  and  $c$ :

- Compute  $c_r = c * r^e \bmod N$
- Get decryption  $m_r$  of  $c_r$  (chosen ciphertext)
- Compute  $m_r * r^{-1} \bmod N = (c * r^e)^{d * r^{-1}} \bmod N$ 
  - $= c^{d * r^{ed * r^{-1}}} \bmod N$
  - $= (m^e)^{d * r^{ed * r^{-1}}} \bmod N$
  - $= m^{ed * r * r^{-1}} \bmod N$
  - $= m \bmod N$

25

For example it is possible to carry out a message recovery attack if an attacker has the ability to get decryptions of arbitrary ciphertexts (as in IND-CCA)

## SECURITY OF RSA



26

Even IND-CPA isn't achieved because encryption is deterministic so it's easy for an attacker to distinguish two ciphertexts (can just encrypt them himself). In practice we use something called OAEP (Optimistic Asymmetric Encryption Padding) to prevent these attacks

## 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?

Again, see tradeoffs between public-key and secret-key encryption (requiring setup vs. being really fast)