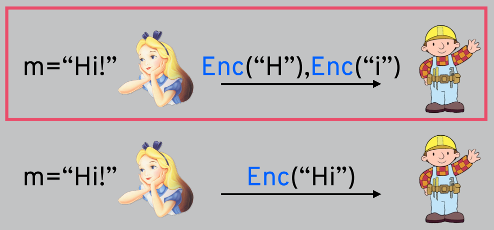
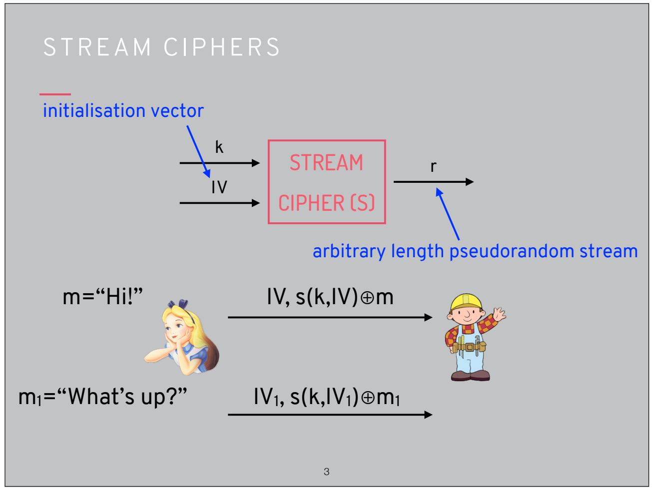
SECURITY (COMP0141): MODERN CIPHERS

MODERN CIPHERS

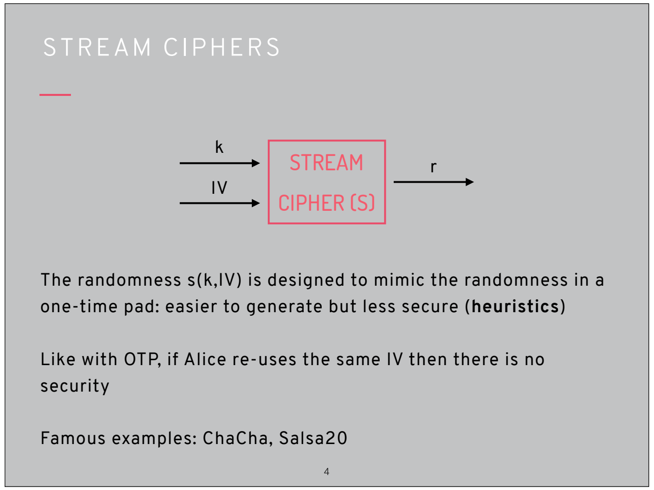
two types: **stream** ciphers and **block** ciphers



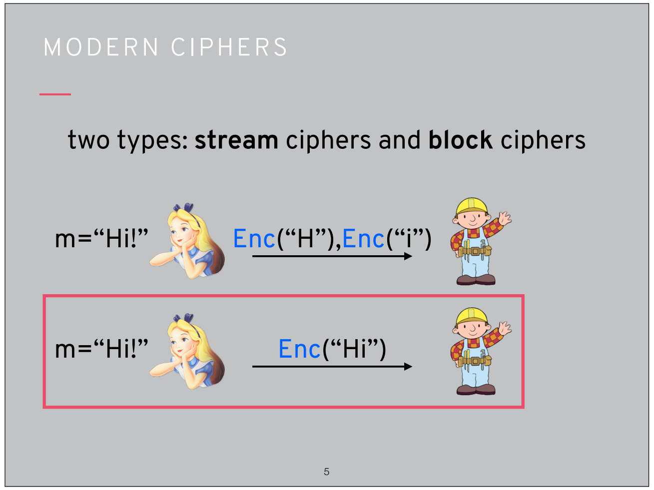
Modern ciphers can be broken into two types

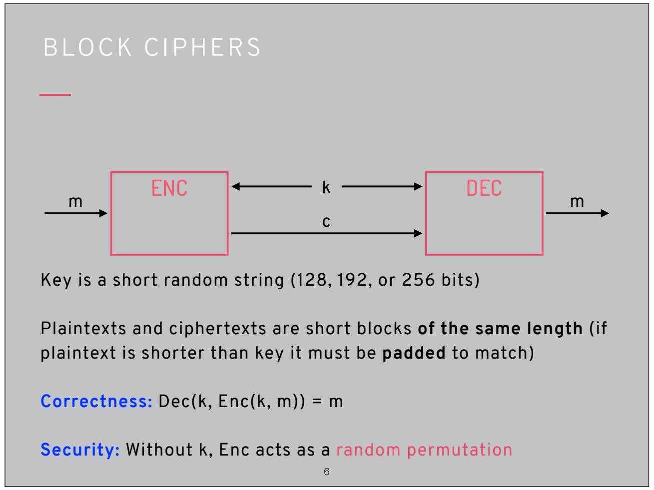


Stream ciphers are most useful in applications where the length of the message isn't known upfront. The sender then uses the stream cipher to generate a pseudorandom string that masks the message

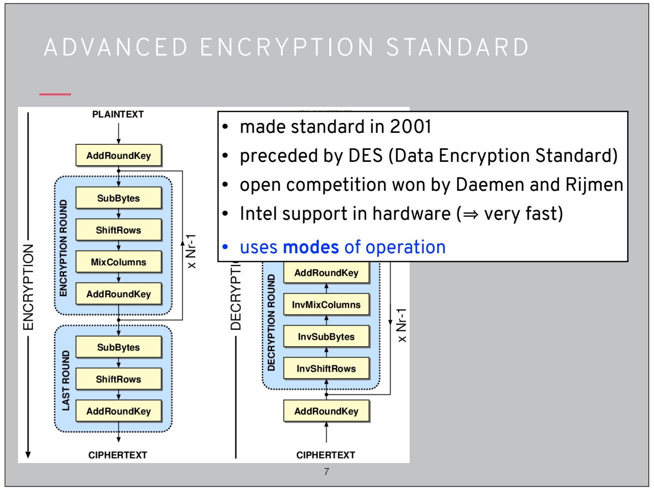


Like a OTP but cheaper and not as secure (because the randomness isn't perfect)

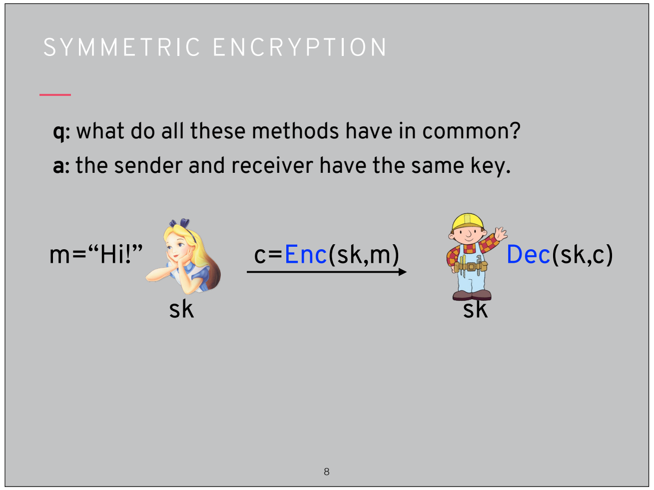




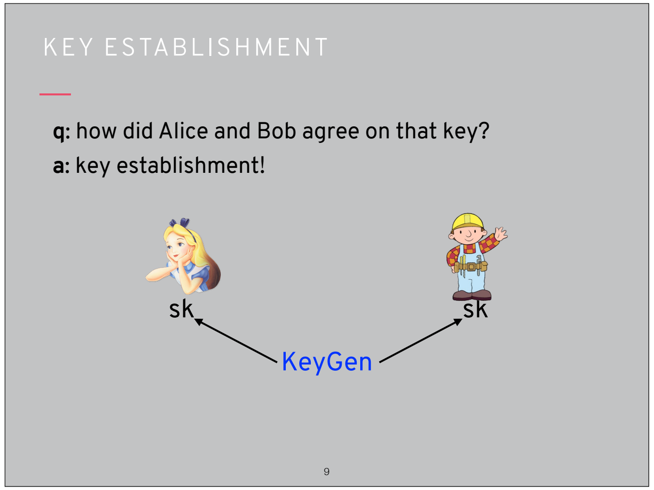
Block ciphers break plaintext into block matching the key size and satisfy two important properties: correctness and security



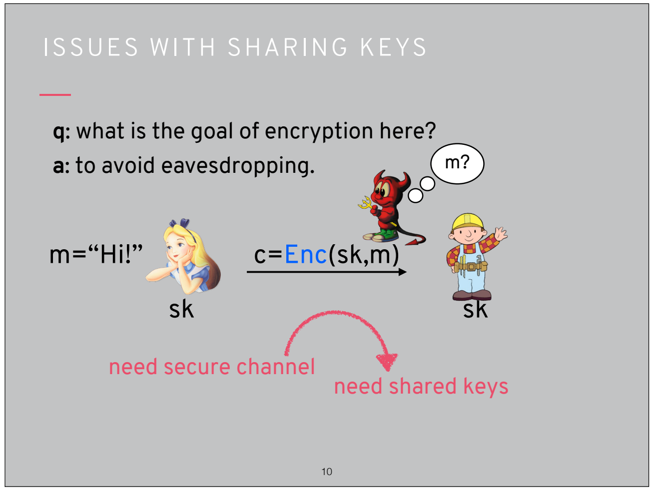
The current block cipher standard is called AES



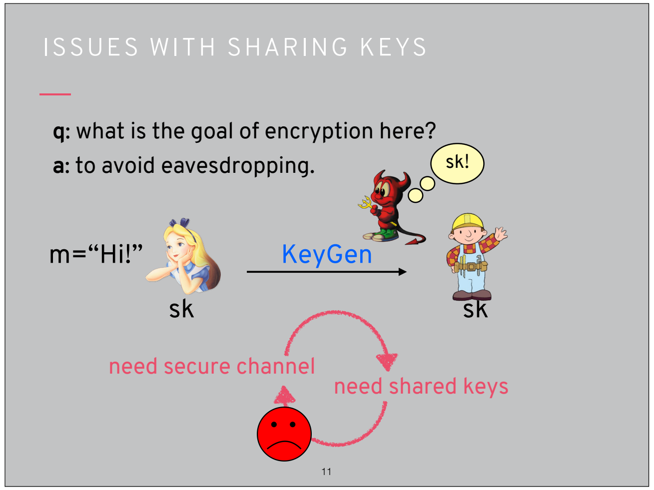
All of these are an example of symmetric encryption, since Alice and Bob have the same information (a secret key)



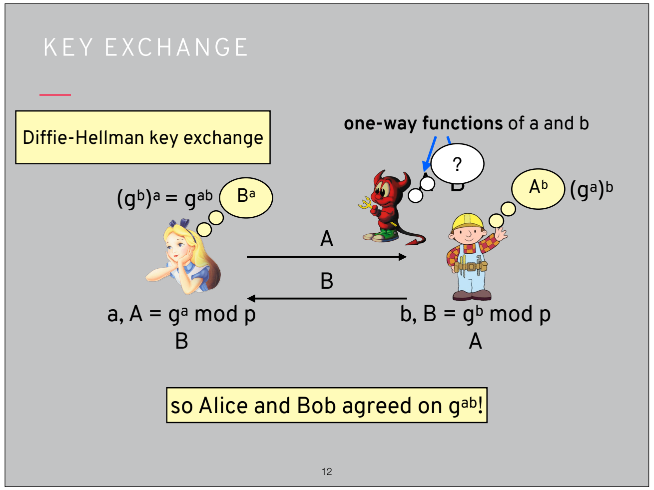
How did they agree on the key though? This is something called key establishment, or key exchange



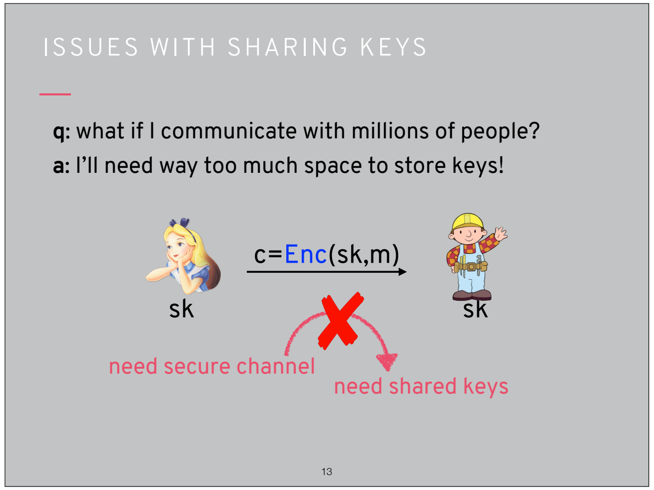
If Alice wants to send a secret message to Bob, need some way to secure the communication channel. The way we've seen that is they share a key, so we need shared keys to establish a secure channel



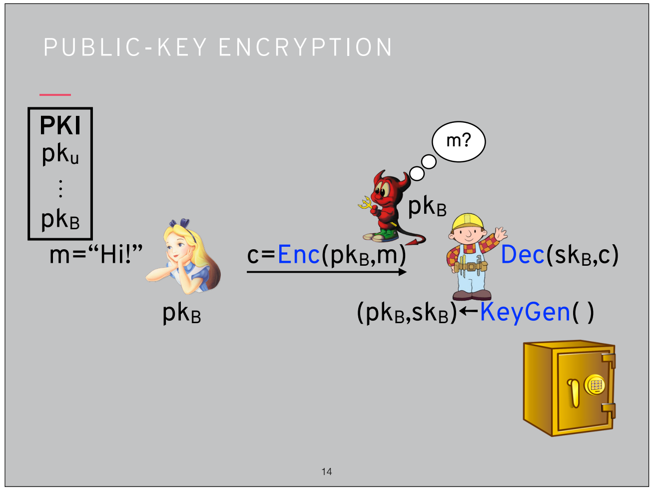
But how do we established shared keys if the adversary can hear everything we're saying? Seems like we need a secure channel!



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

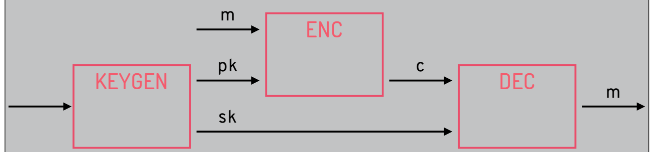


It turns out though that we don't actually need shared keys to secure channels



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: ?

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

THREAT MODEL

Motivation:

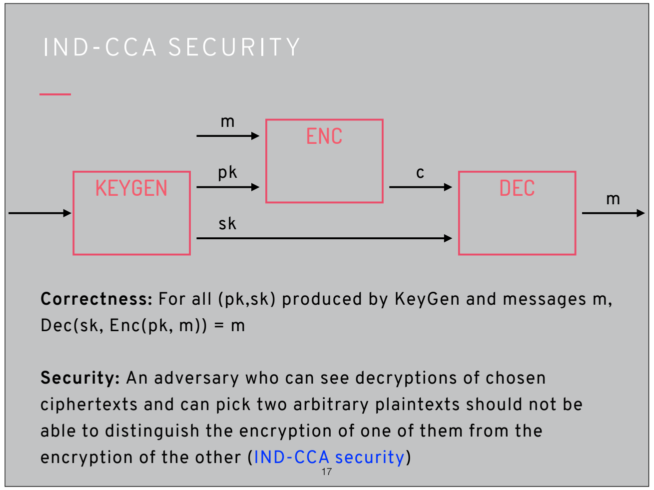
- **Recover key**: learn all future plaintexts
- **Recover plaintext**: learn this specific plaintext
- **Distinguish plaintext**: learn a single bit about plaintext

Capabilities:

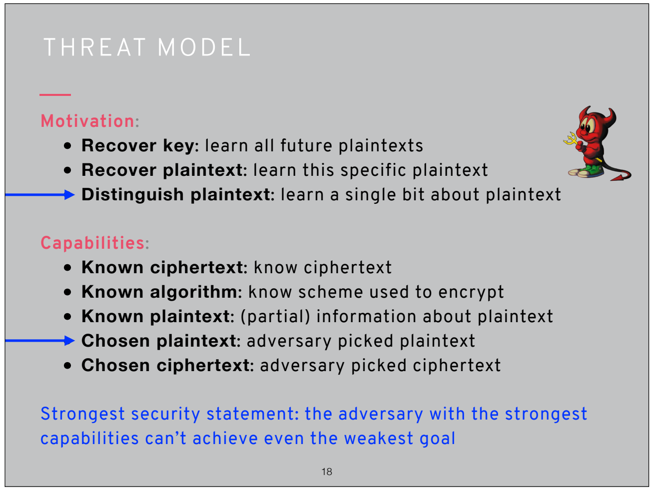
- **Known ciphertext**: know ciphertext
- **Known algorithm**: know scheme used to encrypt
- **Known plaintext**: (partial) information about plaintext
- **Chosen plaintext**: adversary picked plaintext
- **Chosen ciphertext**: adversary picked ciphertext

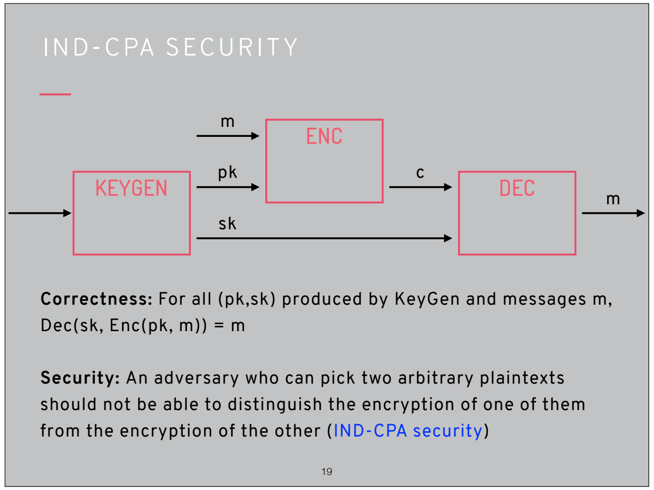
Strongest security statement: the adversary with the strongest capabilities can't achieve even the weakest goal

Remember what we said at the beginning

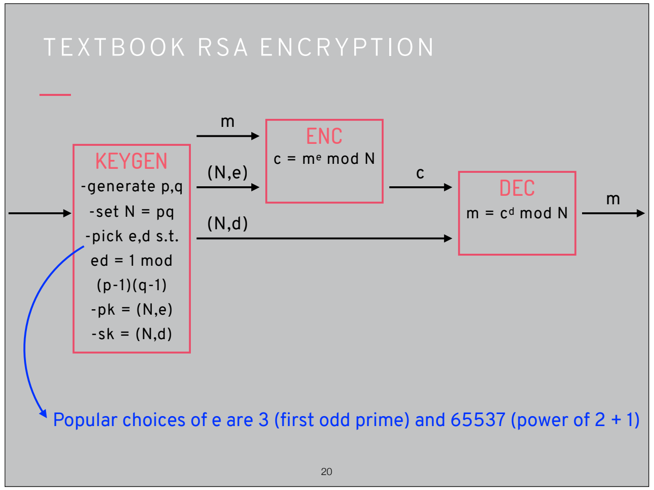


If we take the strongest capability (the adversary can pick both the plaintexts and the ciphertexts) and the weakest goal (learning even a single bit about the message) then we end up with a strong notion of security called IND-CCA (short for INDistinguishability against Chosen Ciphertext Attacks)

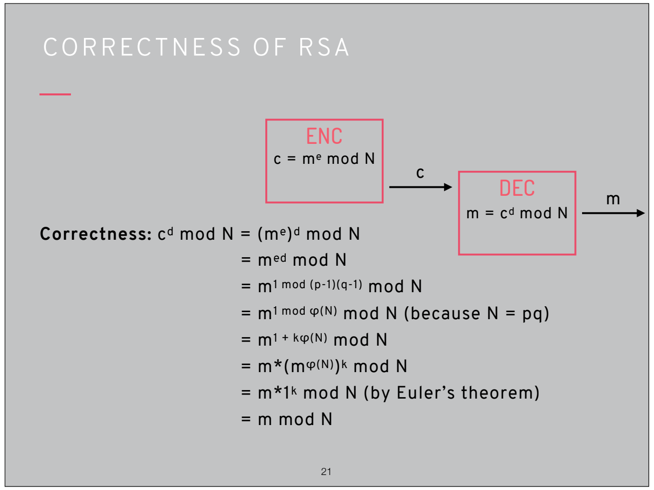




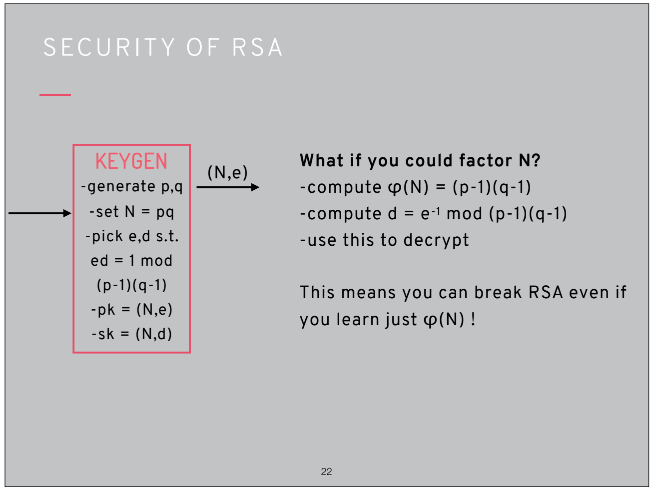
If we take the next strongest capability (the adversary can pick the plaintexts) and the weakest goal (learning even a single bit about the message) then we end up with a strong notion of security called IND-CPA (short for INDistinguishability against Chosen Plaintext Attacks)



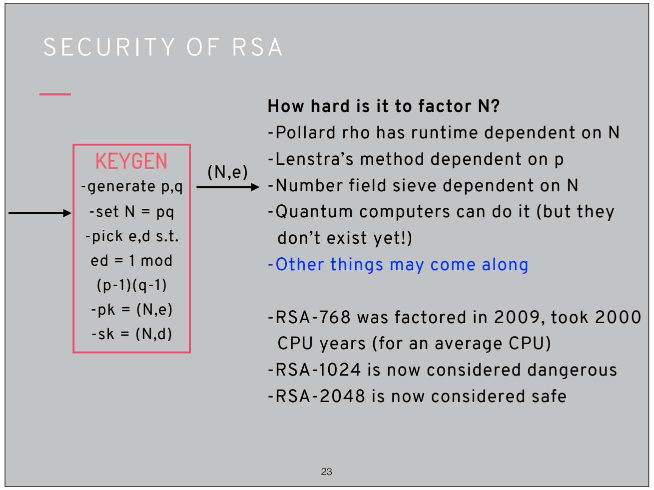
RSA is one of the most famous examples of a public-key encryption scheme. It is named after its creators: Rivest, Shamir, and Adleman



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

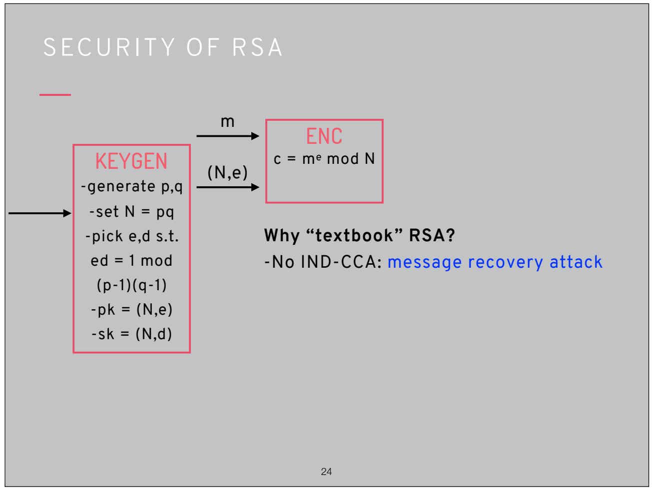


RSA becomes completely insecure if someone can factor N



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

---



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

MESSAGE RECOVERY ATTACK

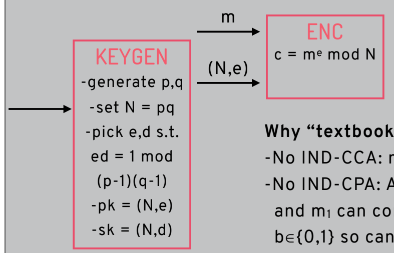
Given (N,e) and c:

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

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

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)

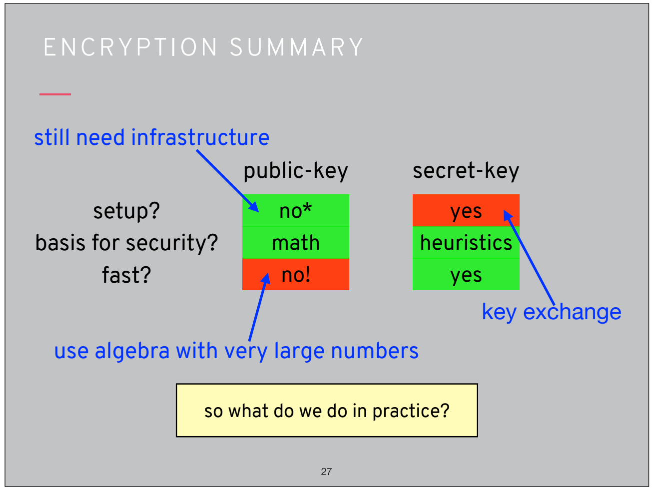
SECURITY OF RSA



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

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



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