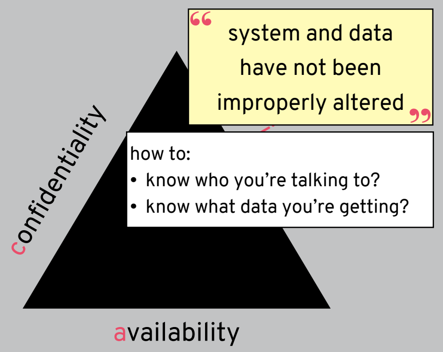
SECURITY (COMP0141): INTEGRITY

INTEGRITY



Integrity is a subtle property, can mean a number of different things

WARNING

You should never design your own cryptography!

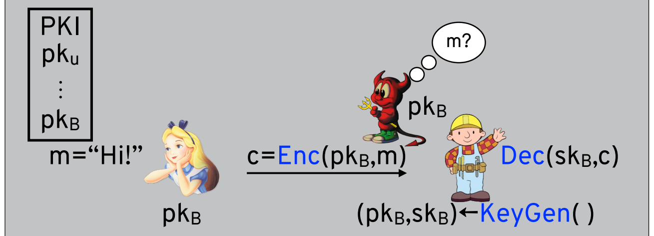
This lecture on cryptography does not in any way qualify you to design cryptographic algorithms or protocols

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

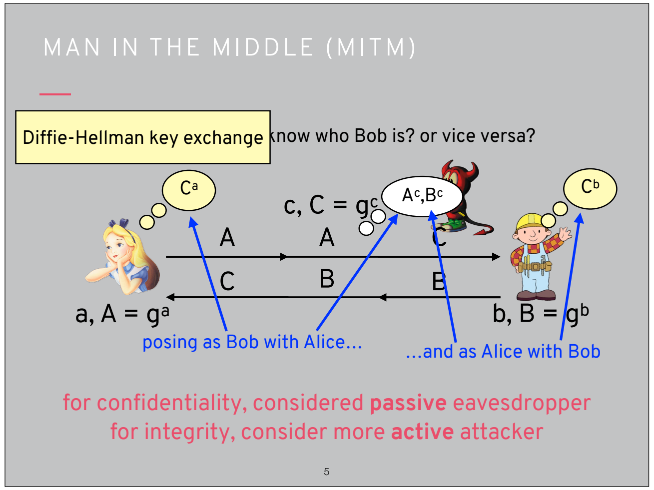
We'll be seeing more cryptographic primitives today so just remember: don't design your own crypto! Or at least don't ever deploy any crypto you designed yourself

---

SECRET COMMUNICATION

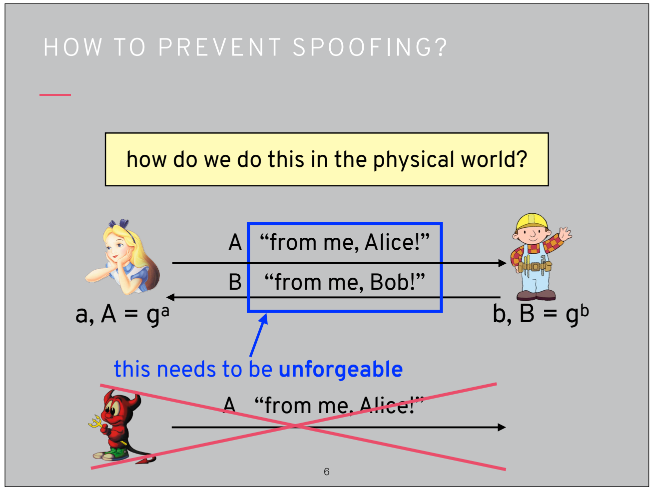


For public-key encryption to work, Alice first needs Bob's public key to encrypt messages to him. But how did she know it was Bob in the first place?

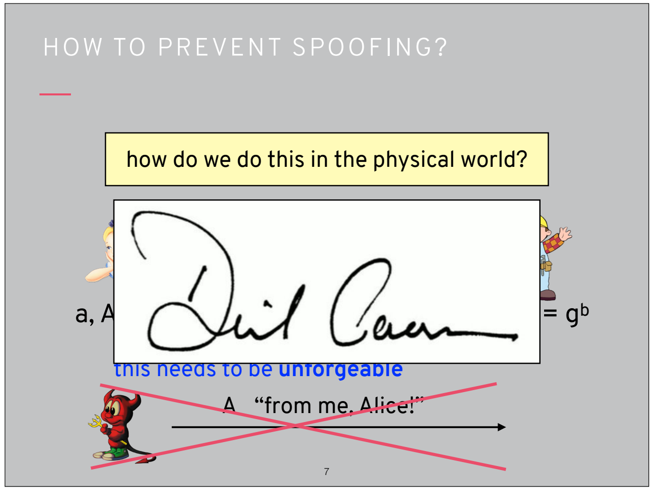


Man-in-the-middle (MitM) attacks rely on lack of integrity. Attacker sitting in the middle of the communication channel (stronger attacker than a passive eavesdropper) can intercept and alter messages to pose as Bob with Alice and as Alice with Bob.

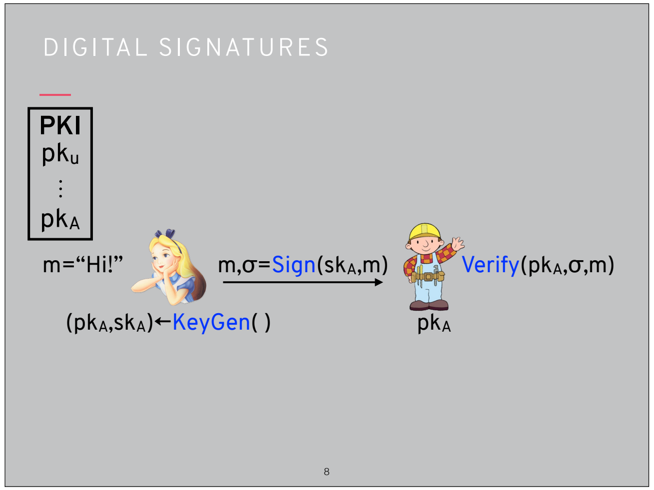
Attacker drops Alice's value A and replaces it with its own value C, then drops Bob's value B and replaces it with C.



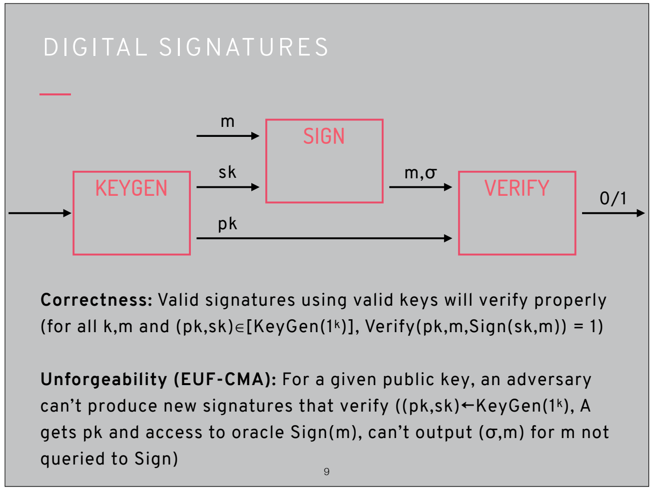
Need some way to convince people that the message is really coming from us, in a way that an attacker can't forge



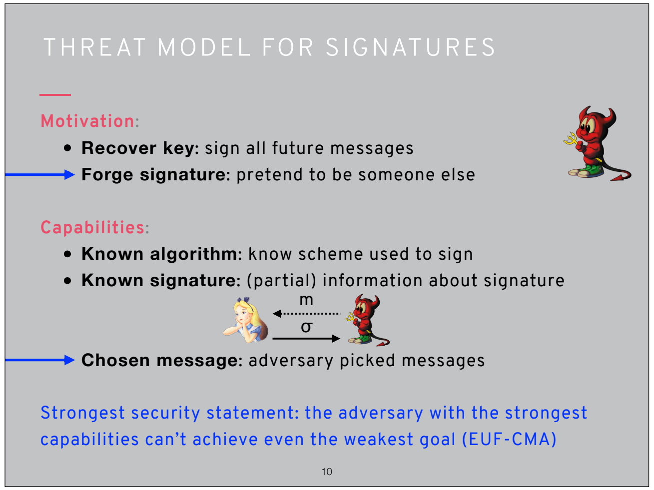
Basically need a digital analogue of physical signatures (or really something even harder to forge)



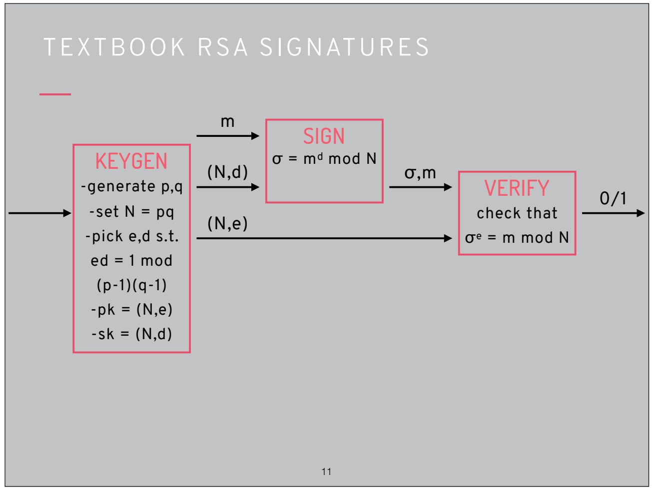
Digital signature use same idea of public and secret keys, but backwards. Now secret key is used by one person to sign (so only Alice can send messages as Alice) and public key is used by many people to verify the origin of a signature



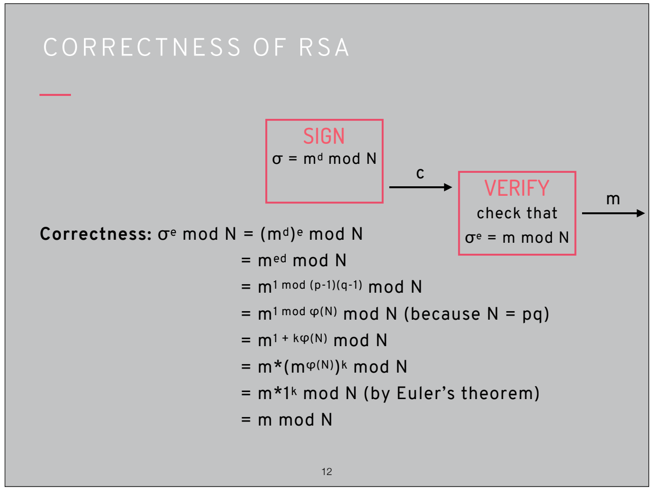
Here is the same thing shown a little more formally, along with the definitions of correctness and security, which in this case is a property called unforgeability



Again, we want to consider both the motivation and the capabilities of the attacker, and we want to say that the strongest attacker still can't achieve the weakest goal



It turns out RSA can also be used for signatures, by just swapping the way we use the public and private keys



The correctness argument is exactly the same as for encryption

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 $f(m_1)*f(m_2) = f(m_1*m_2)$

How about security? Here again this simplified version of RSA isn't very secure because we can use homomorphic property to get new signatures

USING DIGITAL SIGNATURES

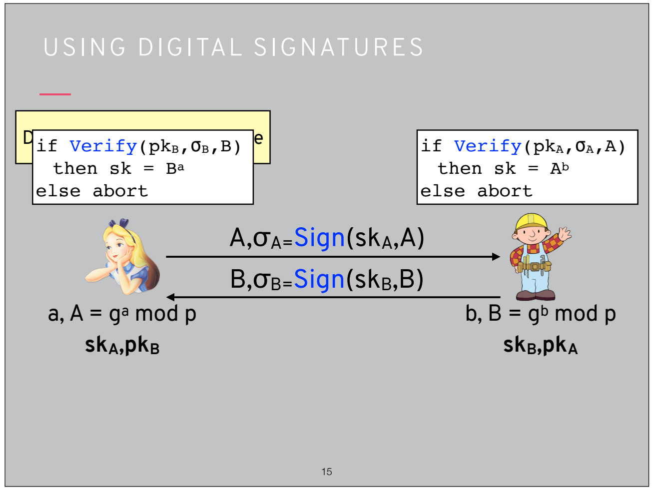
Diffie-Hellman key exchange

$\xrightarrow{A}$

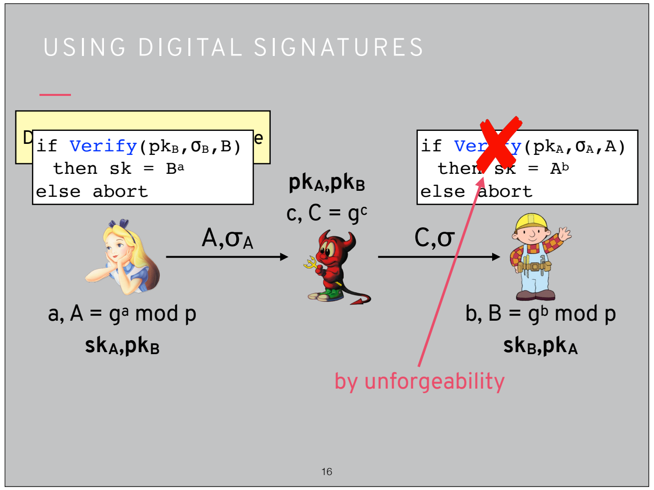
$\xleftarrow{B}$

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

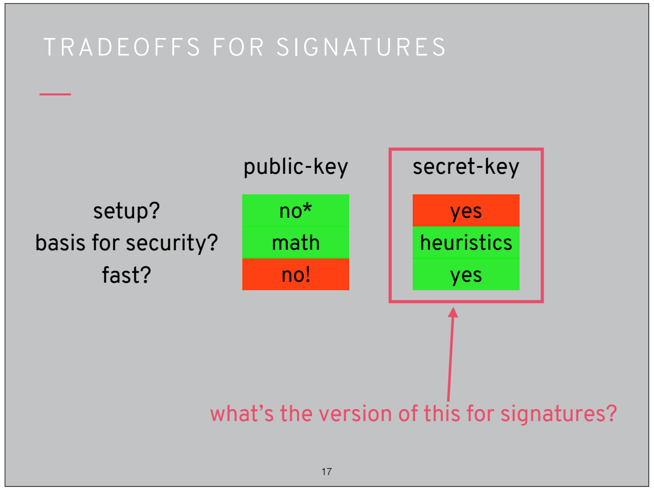
b, $B = g^b \bmod p$
sk_B, pk_A



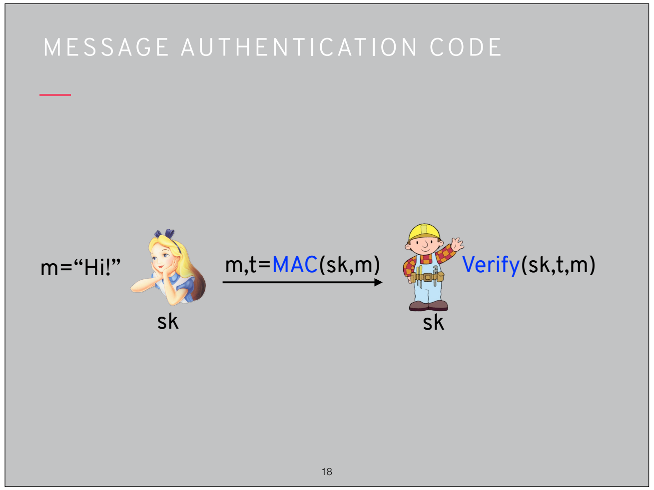
We can prevent MitM attackers using digital signatures, by having Alice and Bob sign the things they say and check that the signatures verify



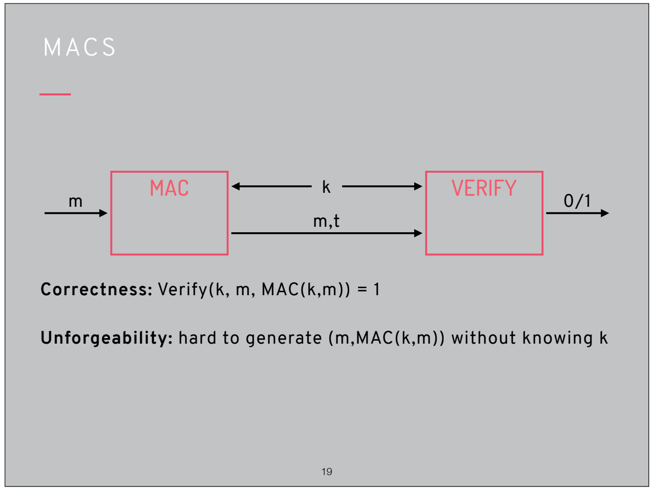
Since the attacker can't forge their signatures it can't launch the attack we saw before, since the attacker won't have and can't produce a signature on its value C



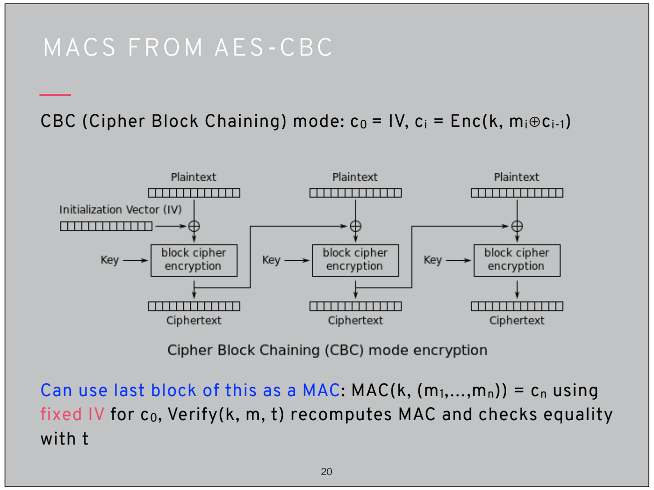
The same tradeoffs exist as they did for public-key encryption: we avoid setup but signatures are (relatively) slow and big. There is a secret-key variant called message authentication codes (MACs)



MACs are the secret-key version of signatures, just like with encryption. The value t is called the tag or the MAC

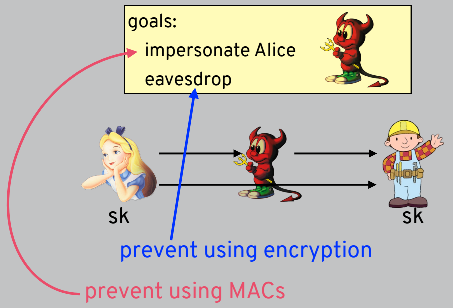


Here's the more formal view of MACs, along with their correctness and security properties. If you look at the diagram, they actually look a lot like block ciphers



This isn't a coincidence: it turns out we can build MACs from block ciphers, and in particular from AES-CBC (one of the modes of operation of AES)

AUTHENTICATED ENCRYPTION (AEAD)



If we combine with encryption, we can get both confidentiality and integrity. This is called authenticated encryption, or authenticated encryption with associated data (AEAD)

THREAT MODEL FOR AEAD

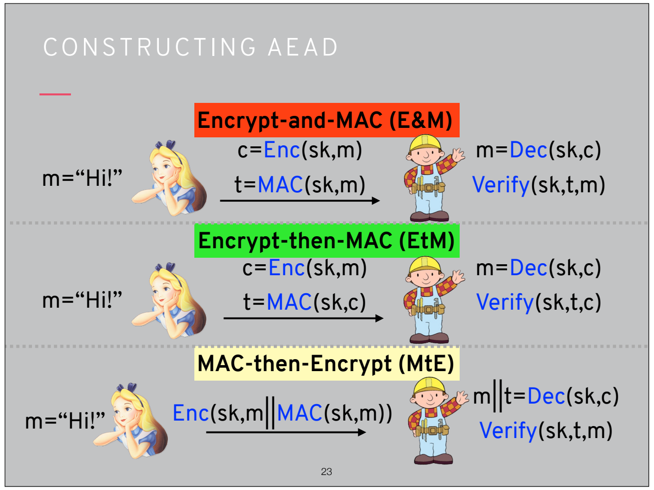
Motivation:

- **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)

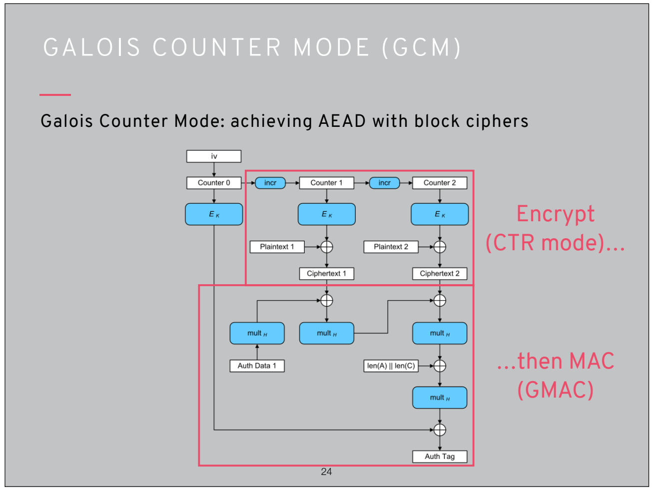
Capabilities:

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

Unsurprisingly, threat model for AEAD combines threat model for encryption with the one for MACs



It isn’t important to understand the details of these methods, but it’s worth pointing out that there are good and bad ways to combine MACs and encryption (see https://en.wikipedia.org/wiki/Authenticated_encryption for more information). Again this is an illustration of the fact that there are many subtleties in designing crypto so you should never do it yourself



AEAD is achieved with block ciphers using AES-GCM, which is an example of Encrypt-then-MAC