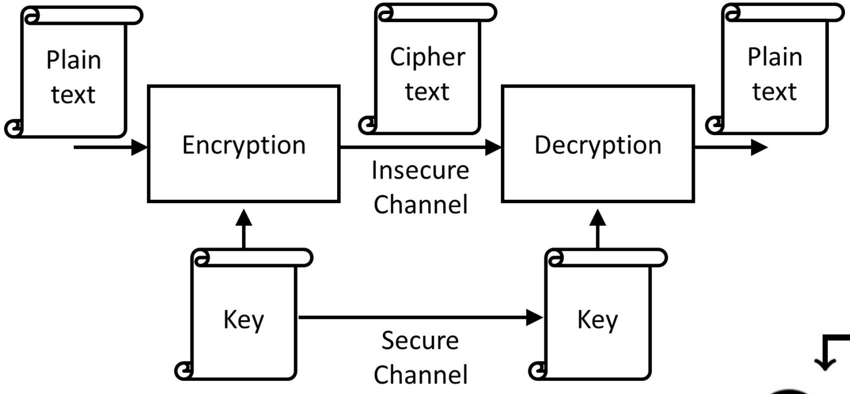
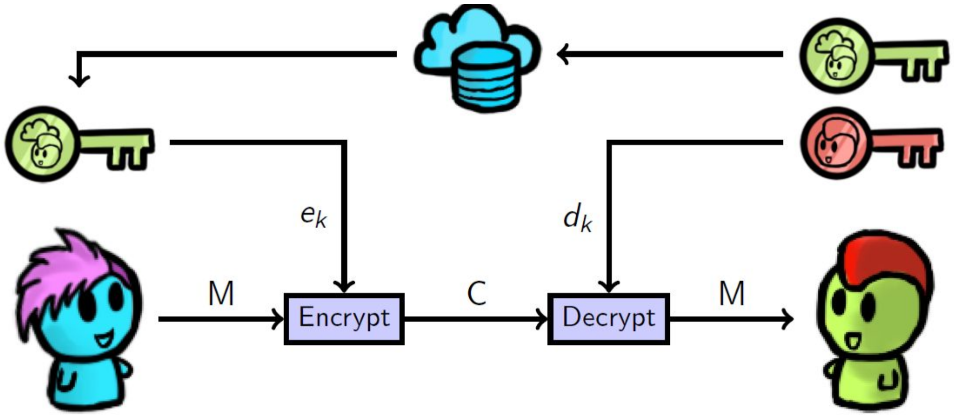
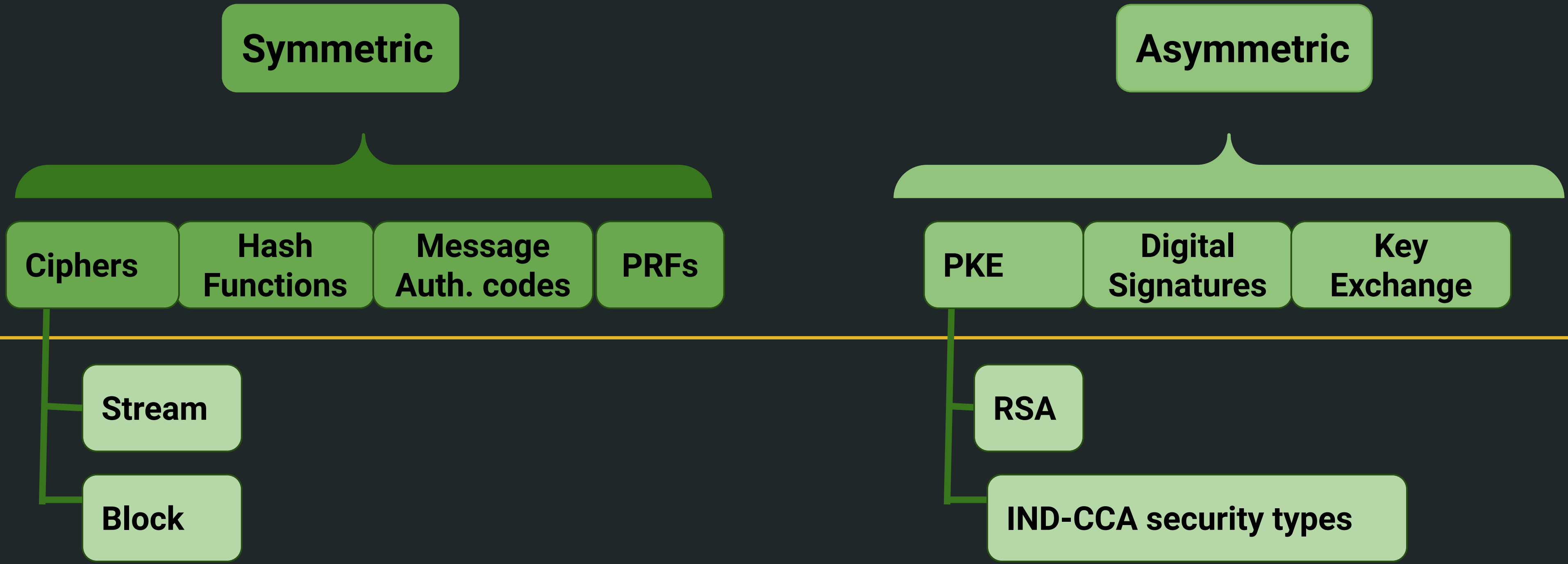
CS489/689
Privacy, Cryptography, Network and Data Security

Winter 2023, Tuesday/Thursday 8:30-9:50am

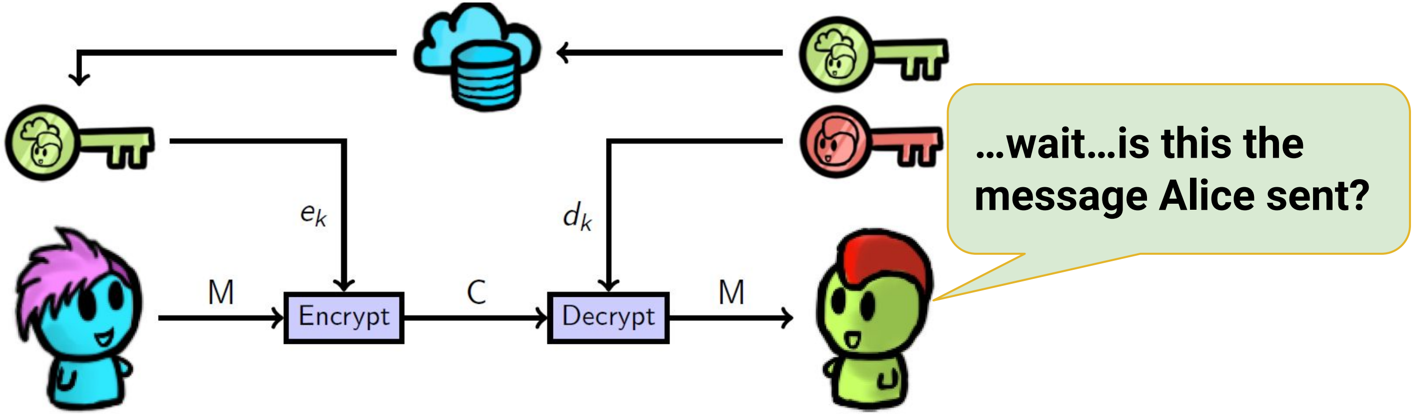
Block/Stream Ciphers, Public Key Cryptography…



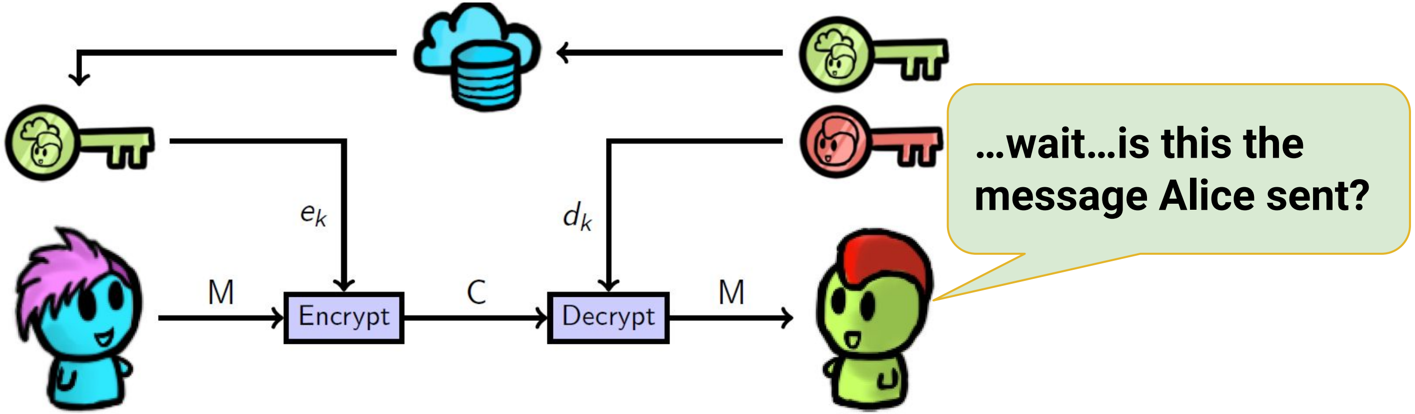



Symmetric
Asymmetric
Ciphers
Hash Functions
Message Auth. codes
PRFs
PKE
Digital Signatures
Key Exchange
Stream
Block
RSA
IND-CCA security types

Detect? Messages Changed in Transit



Detect? Messages Changed in Transit

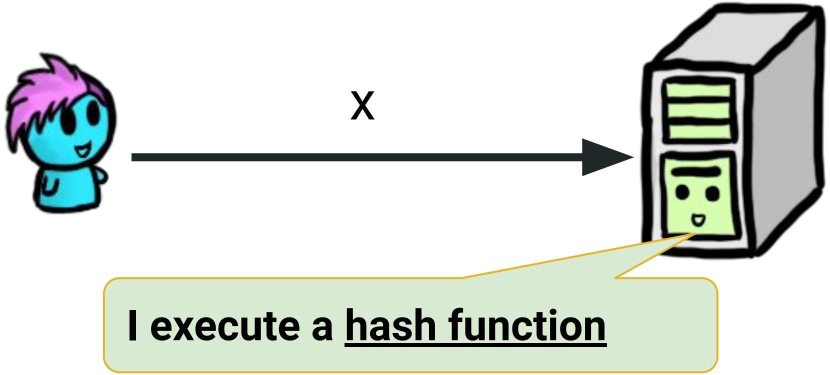


Checksums, appended so Bob can verify it

Not. Good. Enough.

Goal: Make it hard for Mallory to find a second message with the same checksum as the "real" one

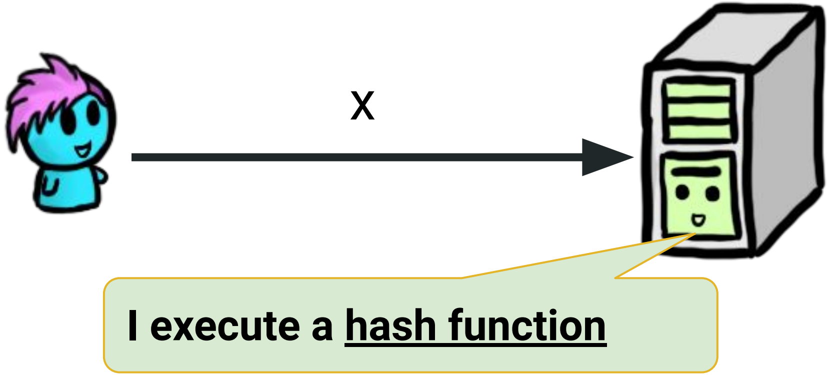
Towards Integrity: Cryptographic Hash Functions



Common examples:

- MD5, SHA-1, SHA-2, SHA-3 (aka Keccak after 2012)

Towards Integrity: Cryptographic Hash Functions

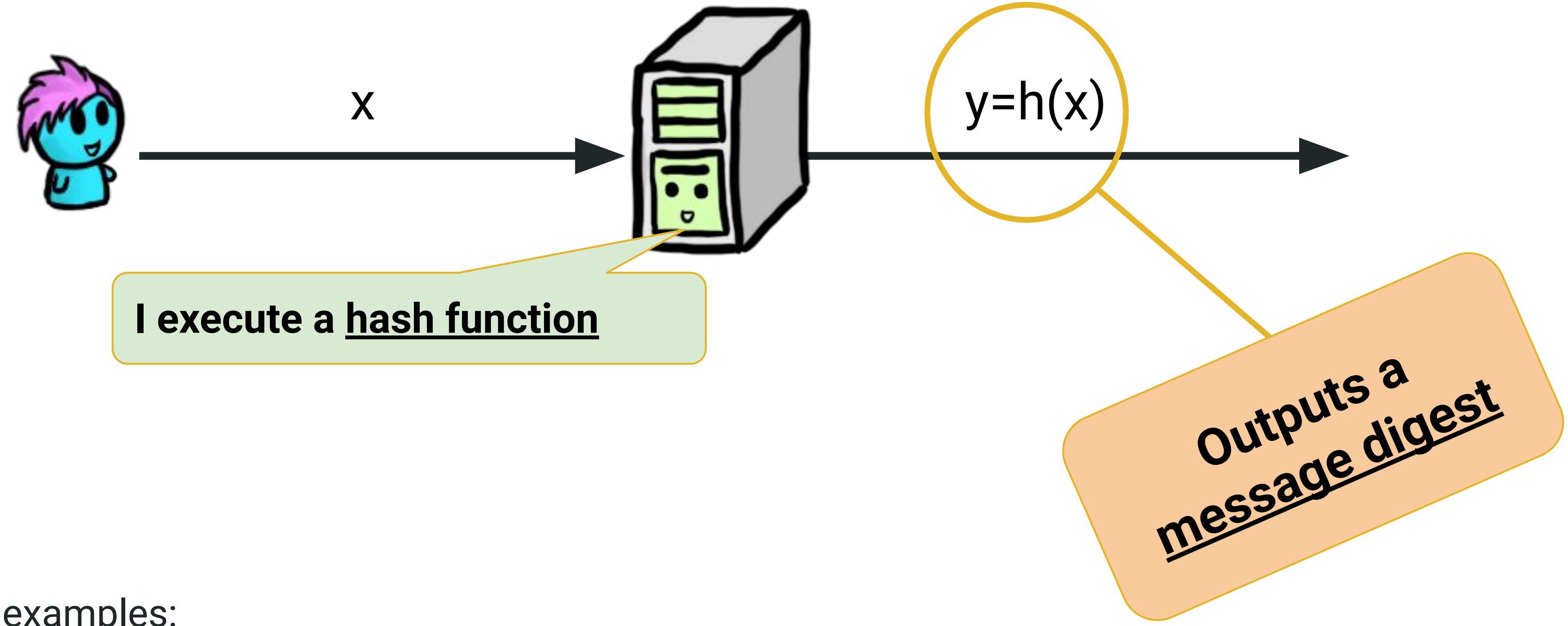


Takes an arbitrary length string, and computes a fixed length string.

Common examples:

- MD5, SHA-1, SHA-2, SHA-3 (aka Keccak after 2012)

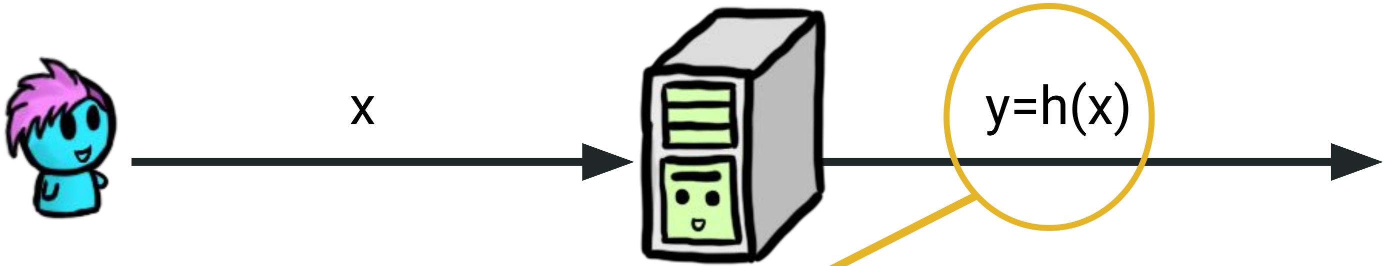
Towards Integrity: Cryptographic Hash Functions



Common examples:

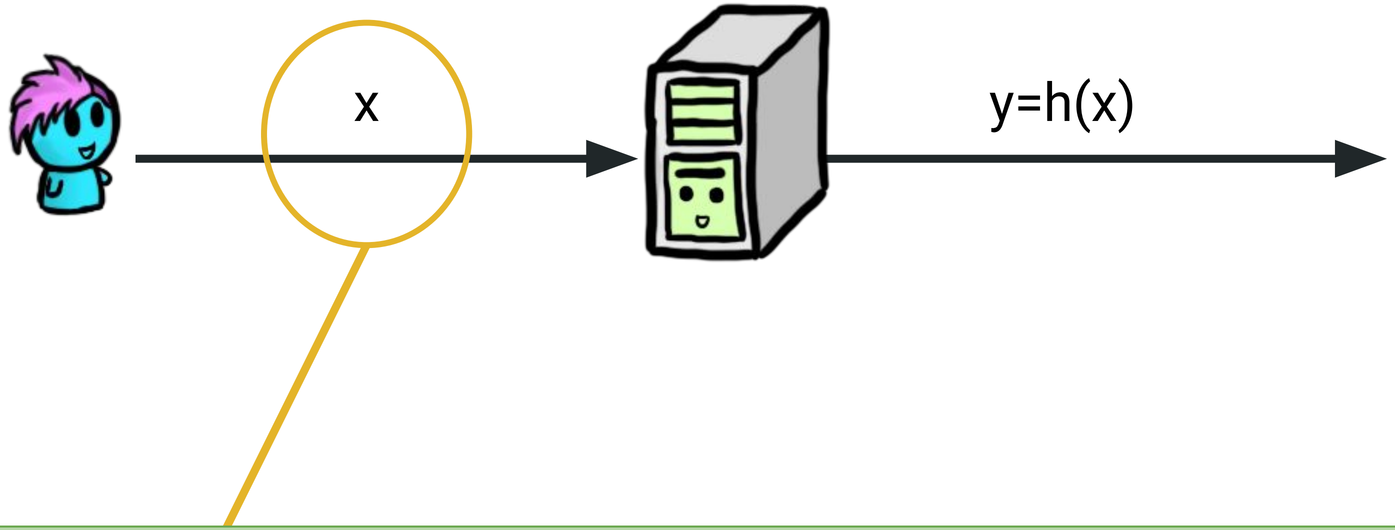
- MD5, SHA-1, SHA-2, SHA-3 (aka Keccak after 2012)

Properties: Preimage-Resistance



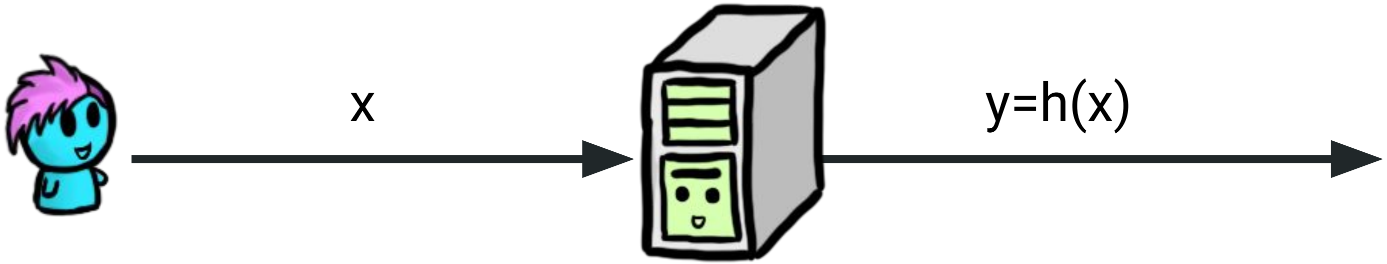
Goal: Given y, “hard” to find x such that h(x) = y

Properties: Second Preimage-Resistance



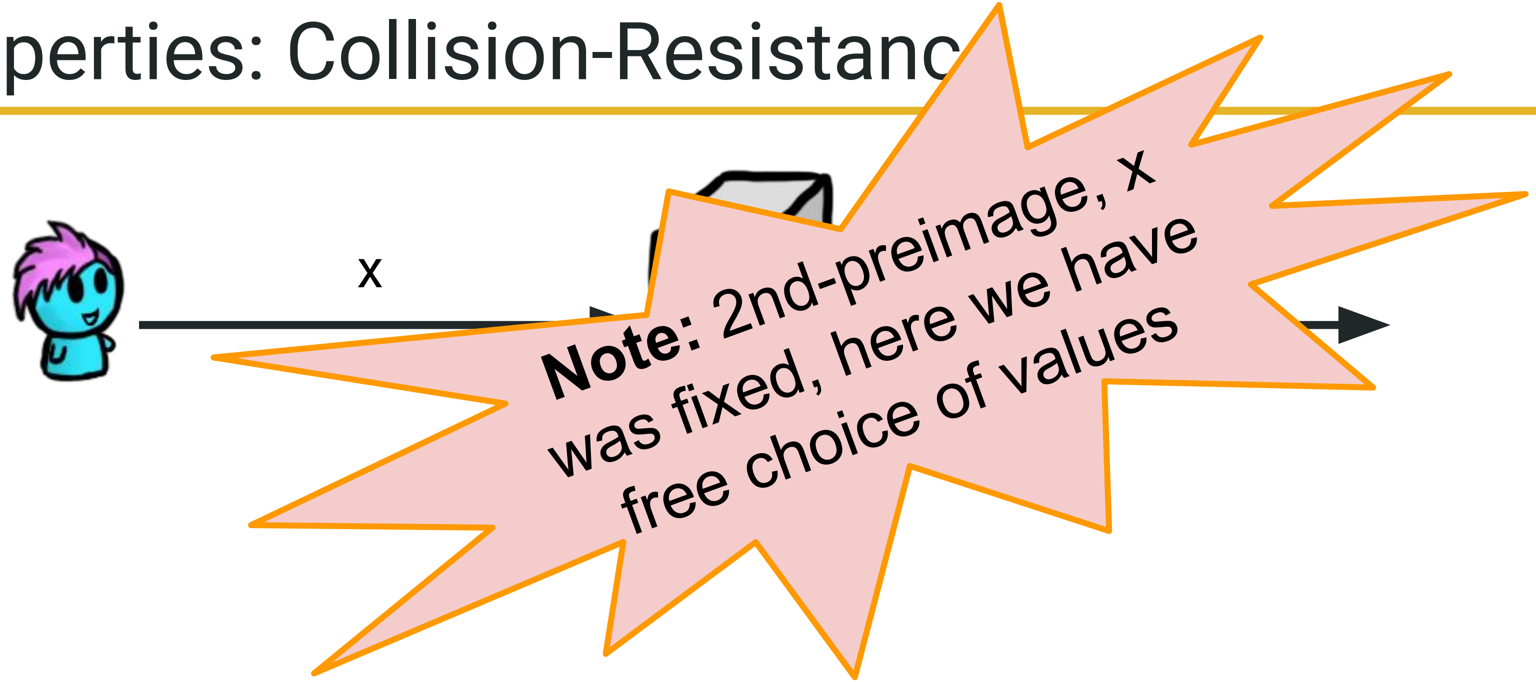
Goal: Given x, “hard” to find x’ <> x such that h(x) = h(x’)

Properties: Collision-Resistance



Goal: It's hard to find any two distinct x, x' such that h(x) = h(x')

Properties: Collision-Resistanc



Goal: It's hard to find any two distinct x, x' such that h(x) = h(x')

Making it too hard to break these properties?

- SHA-1: takes 2^{160} work to find a preimage or second image
- SHA-1: takes 2^{80} to find a collision using brute-force search

Making it too hard to break these properties?

- SHA-1: takes 2^{160} work to find a preimage or second image
- SHA-1: takes 2^{80} to find a collision using brute-force search

There are faster ways to find collisions in SHA-1 or MD5

Collisions and the Birthday Paradox

Collisions are easier due to the birthday paradox

Collisions and the Birthday Paradox

Collisions are easier due to the birthday paradox

What's the probability two of us have the same birthday?

Collisions and the Birthday Paradox

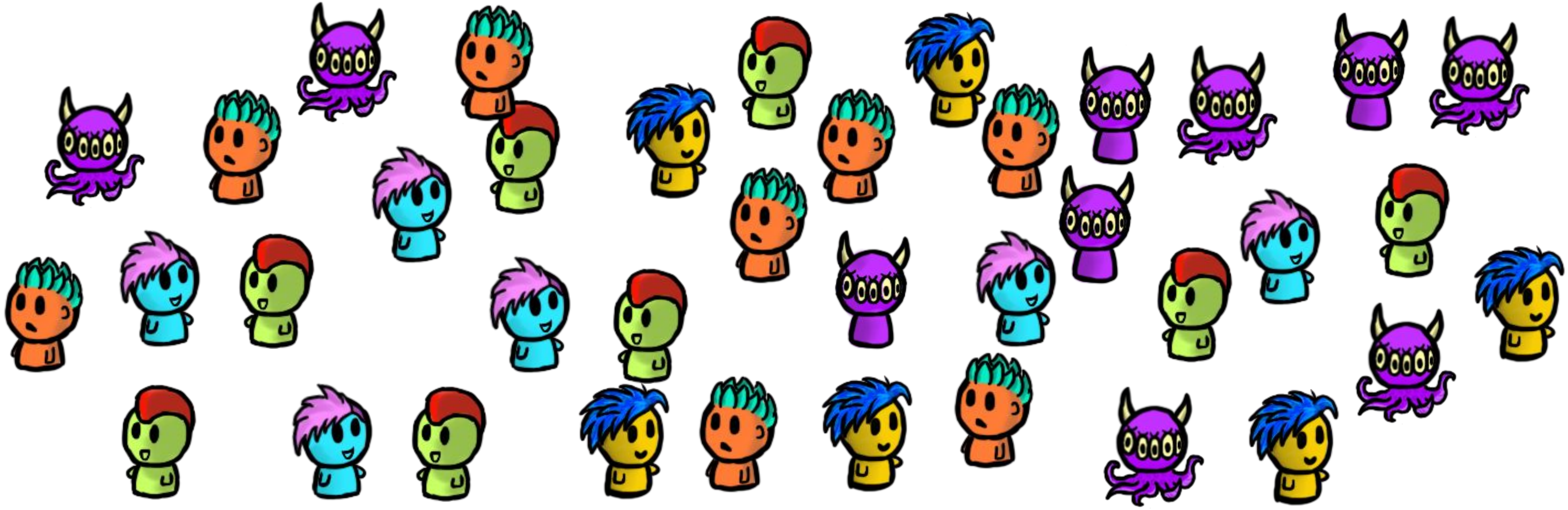
Collisions are easier due to the birthday paradox

What’s the probability two of us have the same birthday?

There’s 23 of us, so larger than 50%!!

Collisions and the Birthday Paradox

Collisions are easier due to the birthday paradox



Collisions and the Birthday Paradox

Collisions are easier due to the birthday paradox

Collisions and the Birthday Paradox

Collisions are easier due to the birthday paradox

Collisions and the Birthday Paradox

Collisions are easier due to the birth... pa...

There's ...

Not the end of our problems

How about a bad example? (Integrity over Conf.)

Q: What can Mallory do to send the message she wants (change it)?

A: Just change it…Mallory can compute the new hash herself.

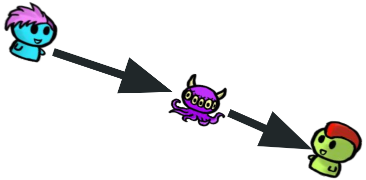
How about a less bad example? (Integrity & Conf.)

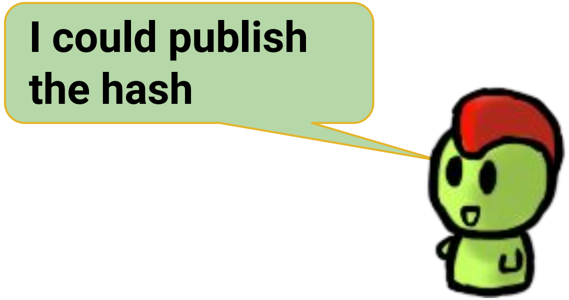
Q: What can Mallory do to send the message she wants (change it)?

A: Still. Just change it.

Limitations for Cryptographic Hash Functions

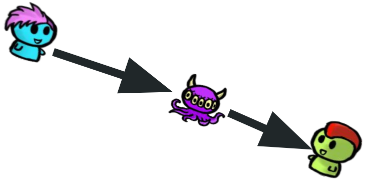
- Integrity guarantees only when there is a **secure** way of sending/storing the message digest





Limitations for Cryptographic Hash Functions

- Integrity guarantees only when there is a **<u>secure</u>** way of sending/storing the message digest

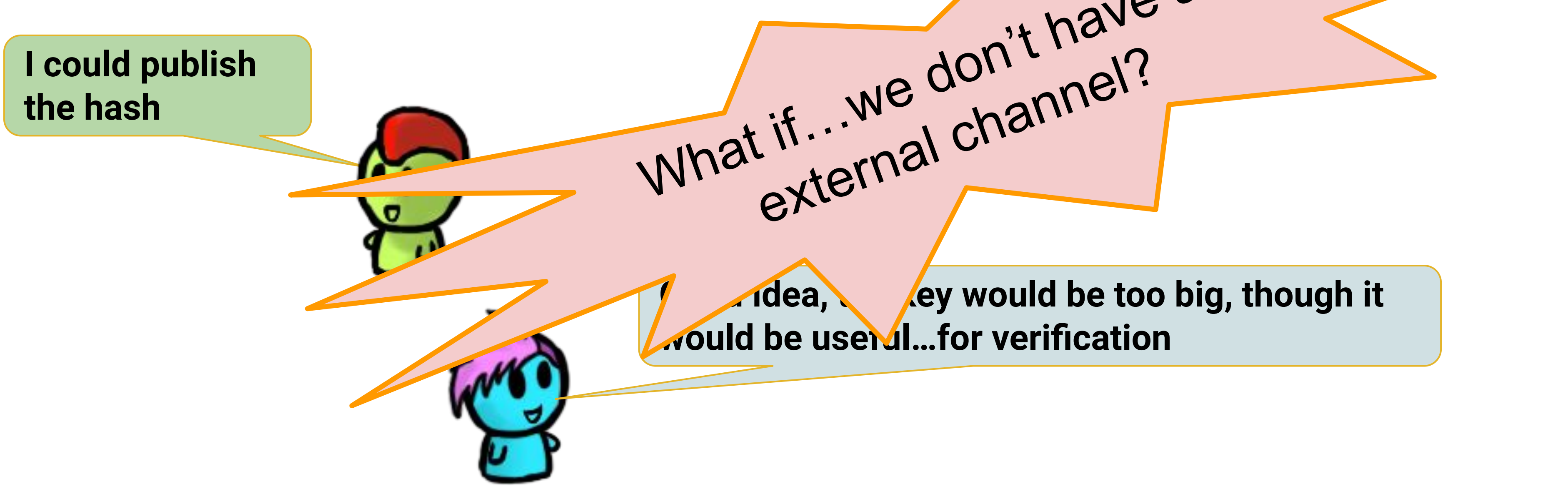


I could publish the hash

Good idea, the key would be too big, though it would be useful…for verification

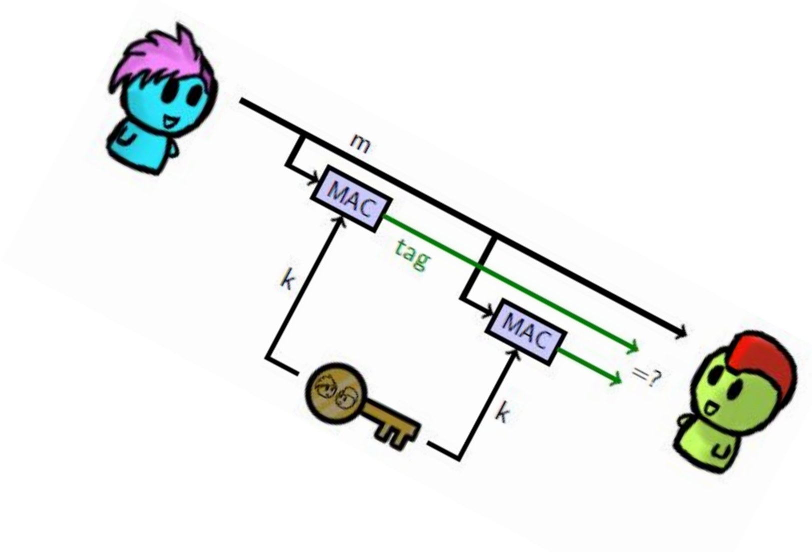
Limitations for Cryptographic Hash Functions

- Integrity guarantees only when there is a ecure way of sending/storing the message d



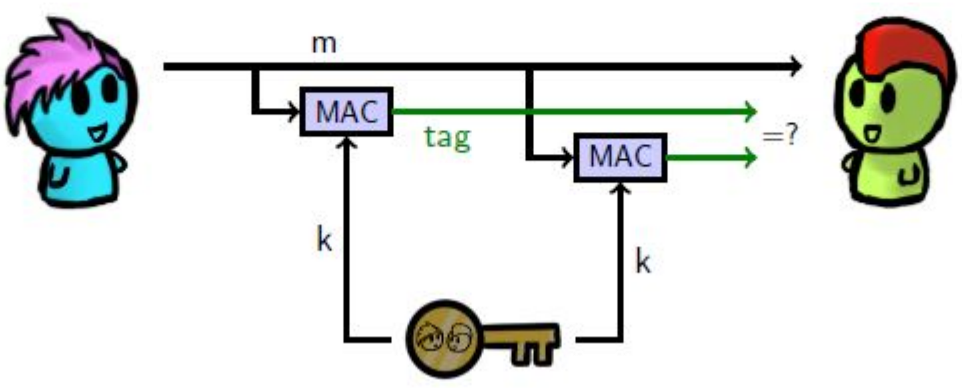
Authentication and Hash Functions

- Use “keyed hash functions”
- Requires the key to generate or check the hash value (tag)



Called: Message authentication codes (MACs)

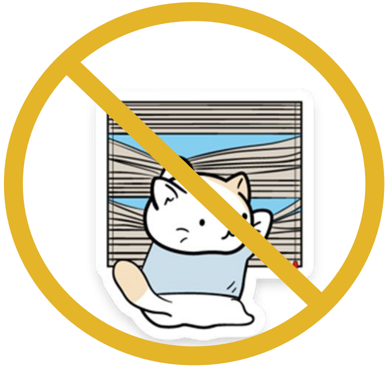
Message Authentication Codes (MACs)



I don't have the key to generate or check the values…

Use "keyed hash functions"
e.g., SHA-1-HMAC, SHA-256-HMAC, CBC-MAC

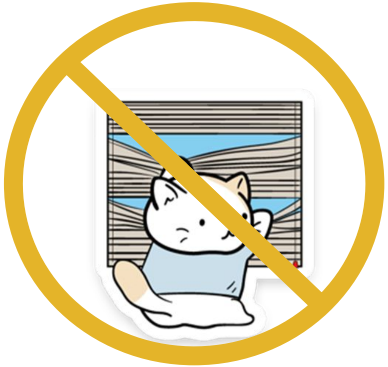
Combine Ciphers and MACs



Confidentiality

Integrity

Combine Ciphers and MACs



Confidentiality

Integrity

Practical systems need both

But how to combine them?

- MAC-then-Encrypt versus
- Encrypt-and-MAC versus
- Encrypt-then-MAC

But how to combine them?

- MAC-then-Encrypt versus
- Encrypt-and-MAC versus
- Encrypt-then-MAC

Ideally, there is an authenticated encryption mode that combines them…but…

Examples that do:

- GCM, CCM, or OCB

Make it work?

- Alice and Bob have a secret key **k** for a cryptosystem
- Also, a secret key **K’** for their MAC

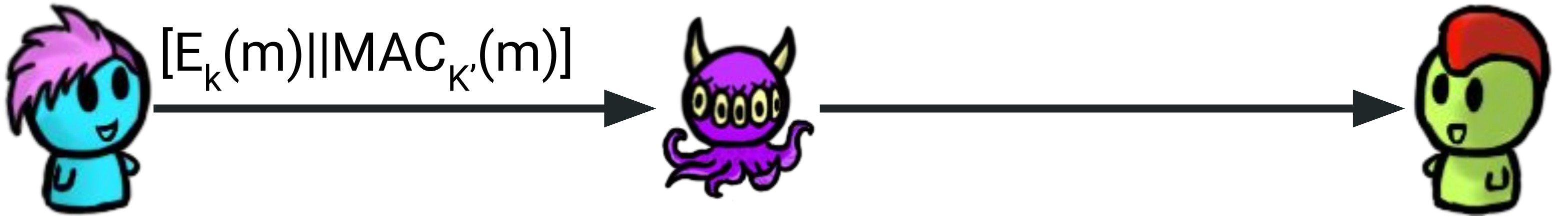
Consider: How can Alice build a message for Bob in the following three scenarios.

MAC-then-Encrypt

- Alice and Bob have a secret key **k** for a cryptosystem and a secret key **K'** for their MAC
- Compute the MAC on the message, then encrypt the message and MAC together, and send that ciphertext.

Encrypt-and-MAC:

- Alice and Bob have a secret key **k** for a cryptosystem and a secret key **K’** for their MAC
- Compute the MAC on the message, the encryption of the message, and send both.



Encrypt-then-MAC:

- Alice and Bob have a secret key **k** for a cryptosystem and a secret key **K'** for their MAC
- Encrypt the message, compute the MAC on the encryption, send encrypted message and MAC

Which order is correct?

Usually: we want the receiver to verify the MAC first!

Q: Which should be recommended then?

$E_k(m||MAC_{K'}(m))$ vs. $E_k(m)||MAC_{K'}(m)$ vs. $E_k(m)||MAC_{K'}(m)$

Which order is correct?

Usually: we want the receiver to verify the MAC first!

Q: Which should be recommended then?

$E_k(m||MAC_{K'}(m))$ vs. $E_k(m)||MAC_{K'}(m)$ vs. $E_k(m)||MAC_{K'}(E_k(m))$

Recommended: Encrypt-then-MAC, $E_k(m)||MAC_{K'}(E_k(m))$

Which order is correct?

Usually: we want the receiver to verify the MAC first!

Q: Which should be recommended then?

$E_k(m||MAC_{K'}(m))$ vs. $E_k(m)||MAC_{K'}(m)$ vs. $E_k(m)||MAC_{K'}(E_k(m))$

Recommended: Encrypt-then-MAC, $E_k(m)||MAC_{K'}(E_k(m))$

Encrypt-then-Mac and the Doom Principle

Act.

Q: What are possible problems that can arise from the other orderings?

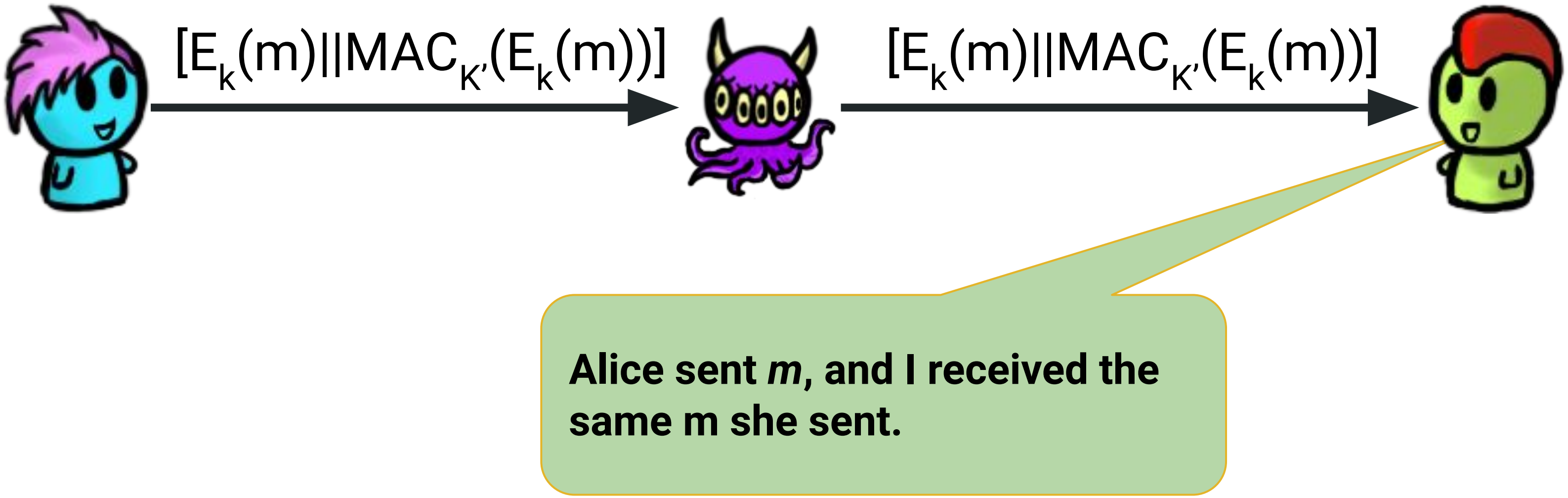
A: Identify an (one) attack for each of $E_k(m||MAC_{K'}(m))$ and $E_k(m)||MAC_{K'}(m)$ Explain the attack at a high level (3-6 sentences or bullet points probably needed)

Hints:

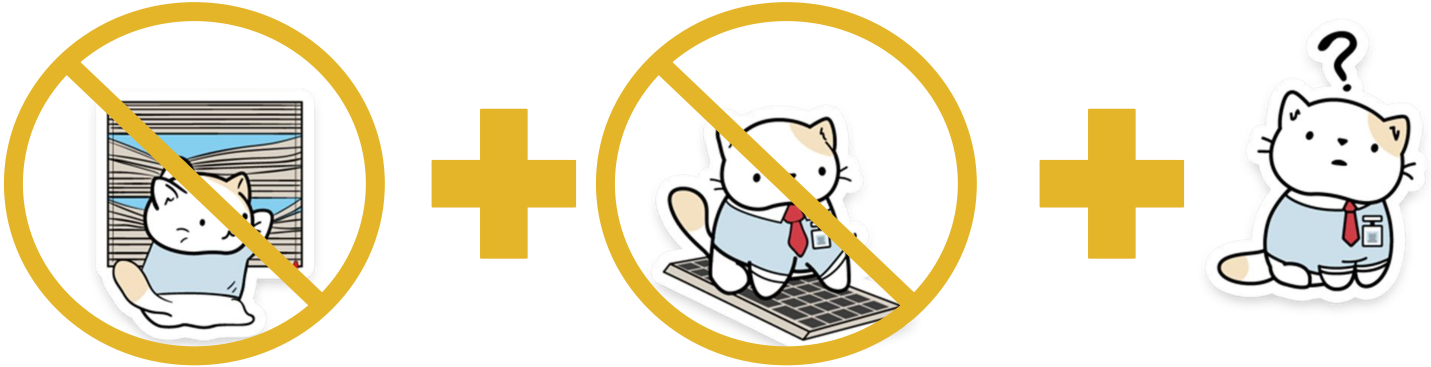
- Properties of cryptosystems we have covered (good and bad)
- https://moxie.org/2011/12/13/the-cryptographic-doom-principle.html

More properties that matter?

Repudiation

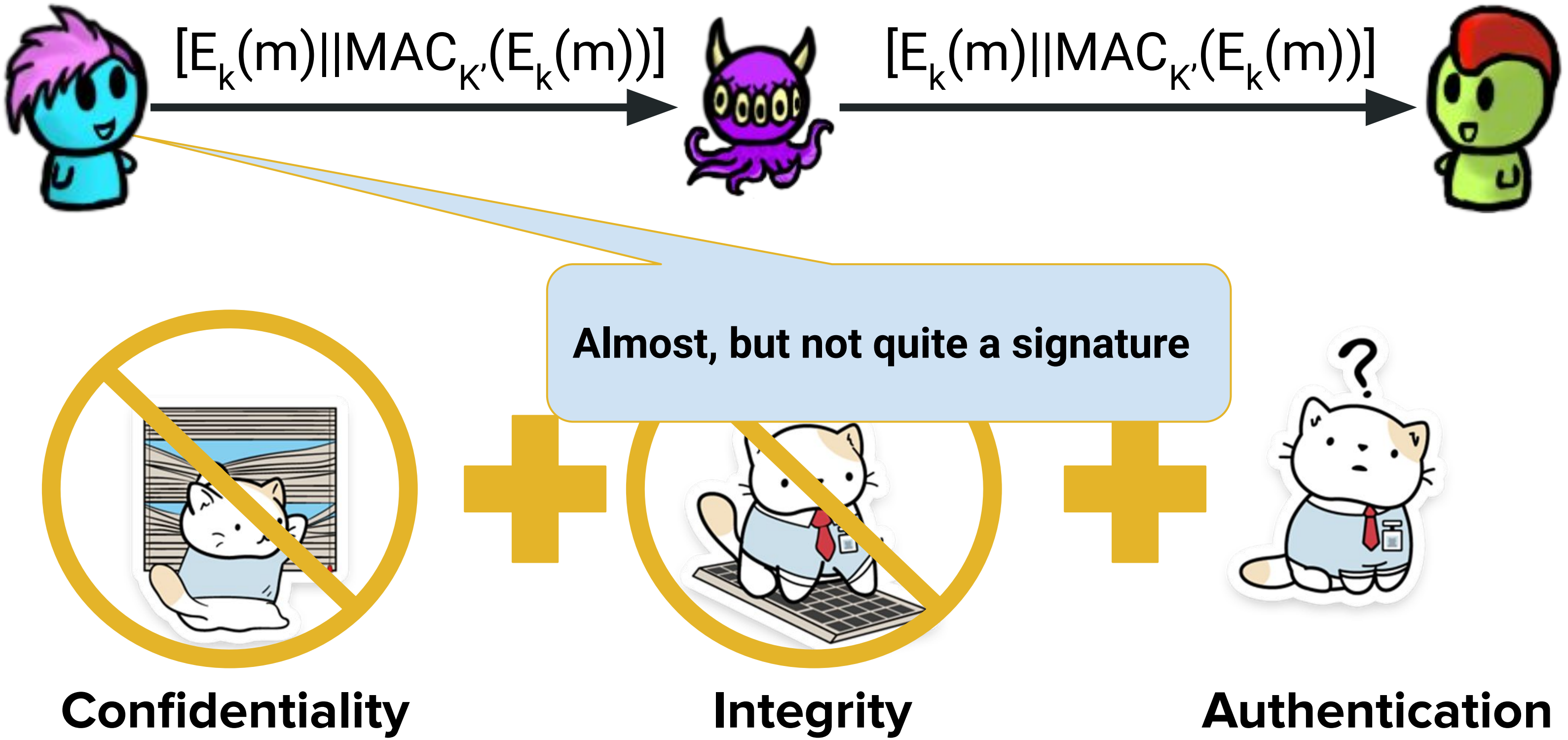


Repudiation



Confidentiality **Integrity** **Authentication**

Repudiation



Repudiation

Almost, but not quite a signature…So…you’re saying Bob can’t prove Alice sent m?

Repudiation

Almost, but not quite a signature…So…you're saying Bob can't prove Alice sent m?

Q: Why can't Bob prove it?

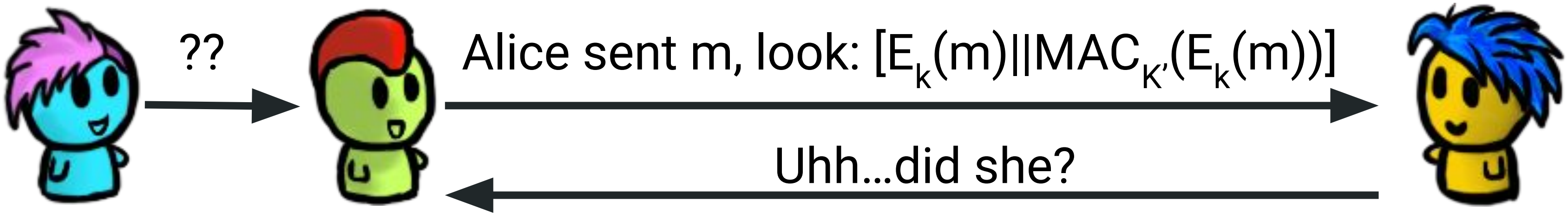
Repudiation

Almost, but not quite a signature…So…you're saying Bob can't prove Alice sent m?

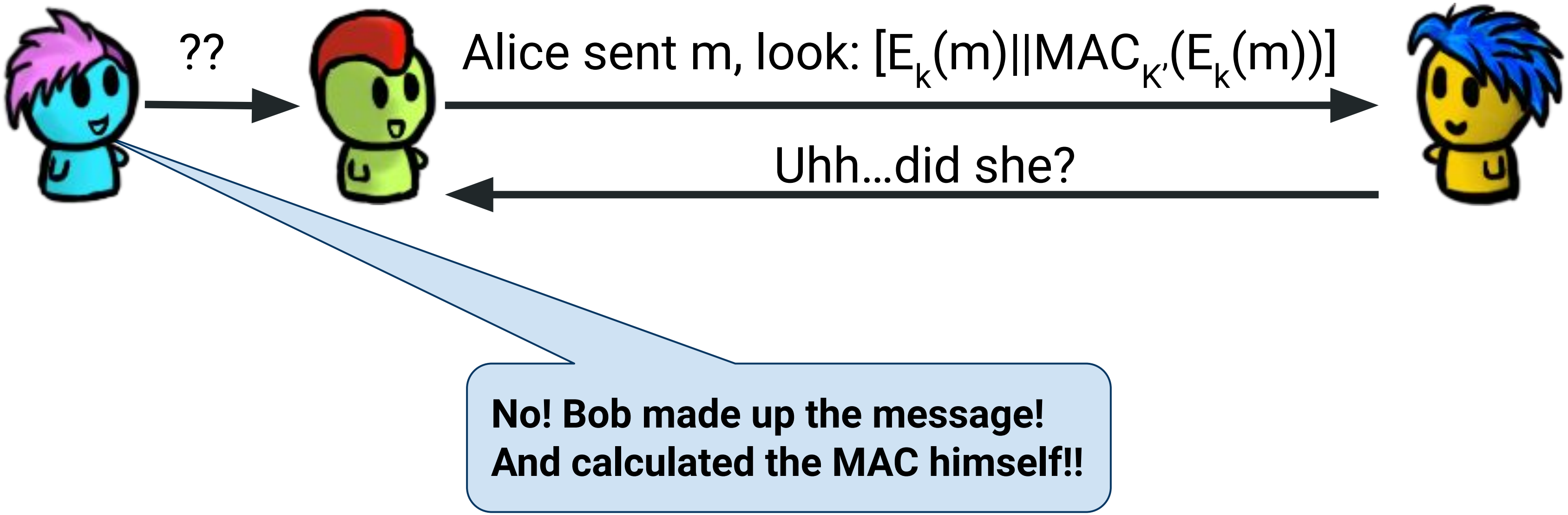
Q: Why can't Bob prove it?

A: Either Alice or Bob could create any message and MAC combo…also Carol doesn't know the secret keys.

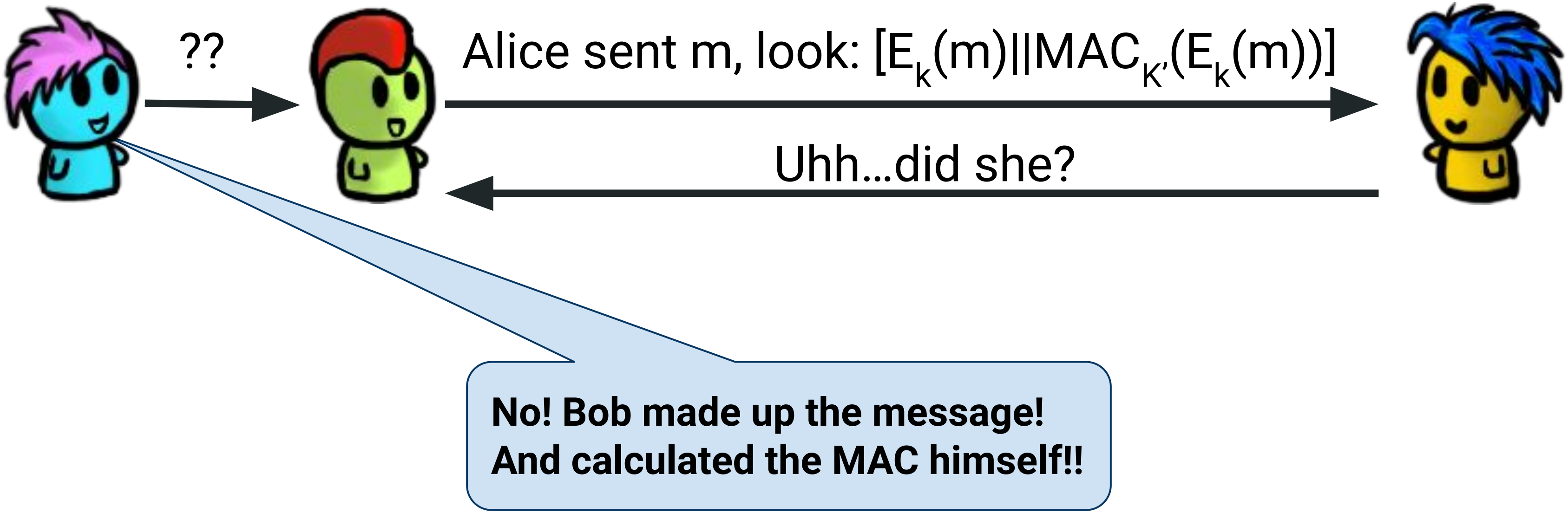
Implications? Repudiation Con't



Implications? Repudiation Con't

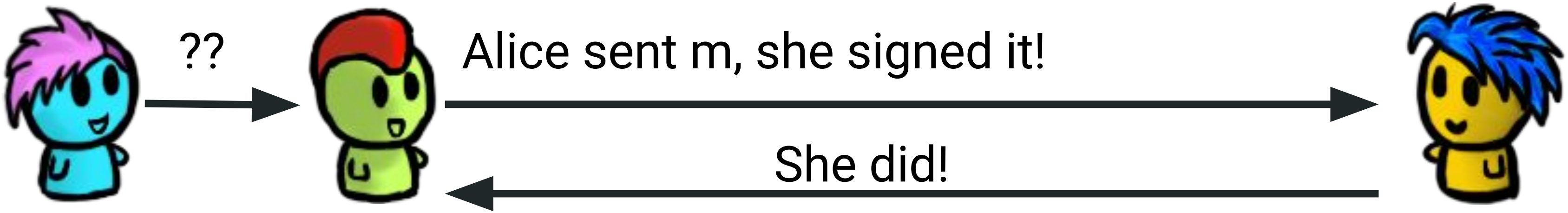


Implications? Repudiation Con’t

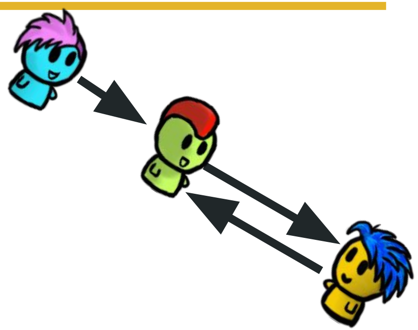


Repudiation Property: For some applications this property is good…others less good (private convos, ecommerce…).

Digital Signatures - For When Repudiation is Bad



Properties and Goals from Digital Signatures

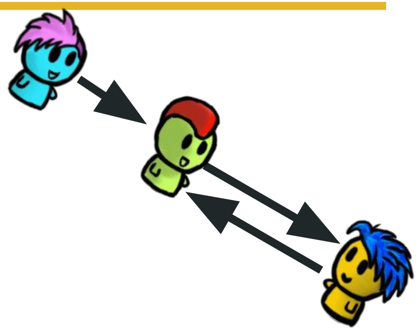


If Bob receives a message with Alice's digital signature then it should mean:

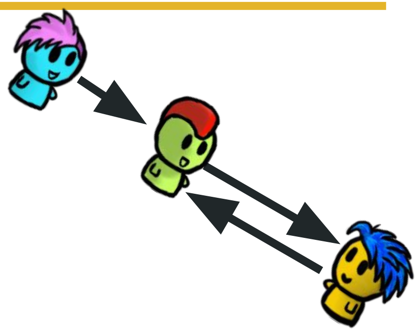
- Alice sent it (not ), this is like a MAC
- The message has not been altered after sending, MAC
- The above two properties should be **provable** to a third party, this property is not like a MAC

Properties and Goals from Digital Signatures

If Bob receives a message with Alice's digital signature then it should mean:



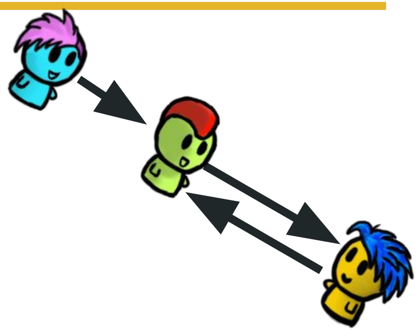
Properties and Goals from Digital Signatures



If Bob receives a message with Alice's digital signature then it should mean:

- Alice sent it (not ), this is like a MAC

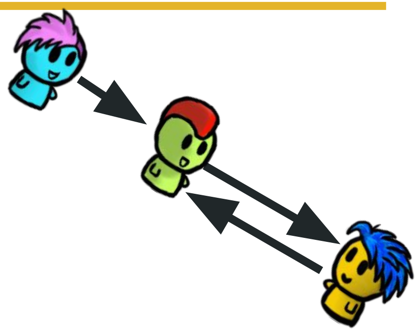
Properties and Goals from Digital Signatures



If Bob receives a message with Alice's digital signature then it should mean:

- Alice sent it (not ), this is like a MAC
- The message has not been altered after sending, MAC

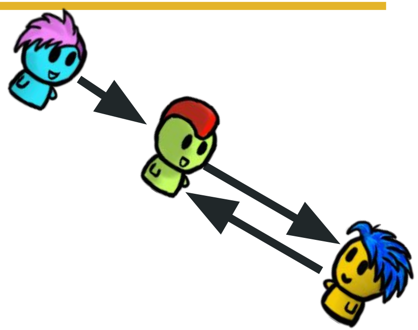
Properties and Goals from Digital Signatures



If Bob receives a message with Alice's digital signature then it should mean:

- Alice sent it (not ), this is like a MAC
- The message has not been altered after sending, MAC
- The above two properties should be **provable** to a third party, this property is not like a MAC

Properties and Goals from Digital Signatures



If Bob receives a message with Alice's digital signature then it should mean:

- Alice sent it (not ), this is like a MAC
- The message has not been altered after sending, MAC
- The above two properties should be **provable** to a third party, this property is not like a MAC

Achievable? Use techniques similar to public-key crypto (last class)

Making Digital Signatures

1. Two keys again 

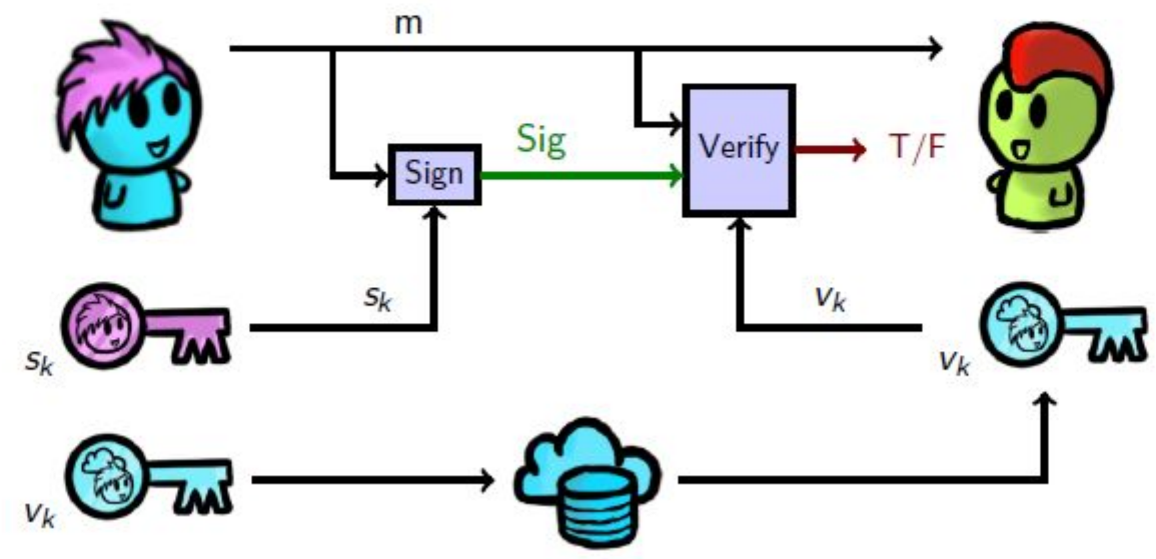
2. Everyone gets the verification key 

3. Alice signs with private signing key 

4. Bob verifies using verification key 

5. If it verifies correctly, success, valid signature

DIgital Signatures at a Glance

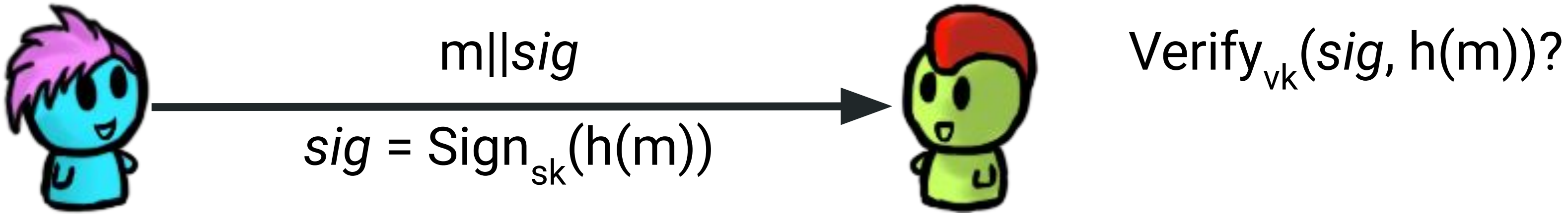


Faster Signatures, aka More Hybrids

- Signing large messages, slow
- However, a hash is much smaller than the message…

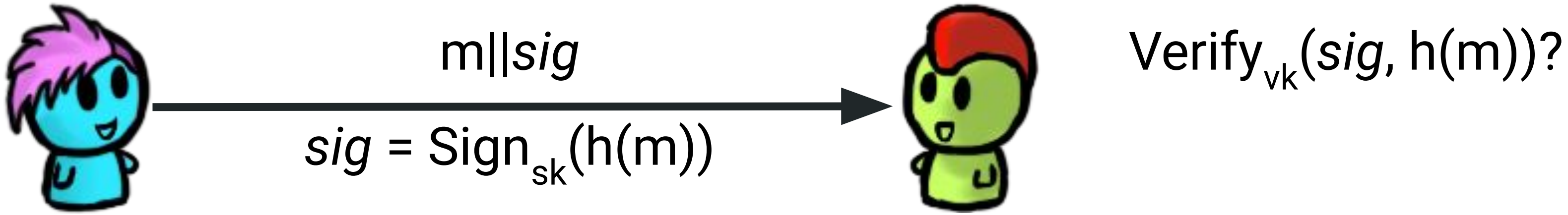
Faster Signatures, aka More Hybrids

- Signing large messages, slow
- However, a hash is much smaller than the message…



Faster Signatures, aka More Hybrids

- Signing large messages, slow
- However, a hash is much smaller than the message…



- Finally, authenticity and confidentiality are separate, you need to include both if you want to achieve both

The Key Management Problem

Q: How can Alice and Bob be sure they're talking to each other?

The Key Management Problem

Q: How can Alice and Bob be sure they're talking to each other?

A: By having each other's verification key!

The Key Management Problem

Q: How can Alice and Bob be sure they're talking to each other?

A: By having each other's verification key!

Q: But how do they get the keys…

A: Know it personally (manual keying e.g., SSH) or trust a friend (web of trust e.g, PGP)

The Key Management Problem…Solutions?

Q: But how do they get the keys…

A: Know it personally (manual keying e.g., SSH)

A: Trust a friend (web of trust e.g, PGP)

A: Trust some third party to tell them (CAs, e.g., TLS/SSL)

Tuesday: More Cryptography…