CARDIFF UNIVERSITY
PRIFYSGOL CAERDYDD

MSc - Cybersecurity

CMT310: Developing Secure Systems and Applications

Hashing and Digital Signatures

Dr Neetesh Saxena
saxenan4@cardiff.ac.uk

Learning Outcomes

- Digital Signature Overview
- Hash functions
- MAC
- HMAC
- Authenticated Encryption

Digital Signature Overview

Electronic signatures

The European Community Directive on electronic signatures refers to the concept of an **electronic signature** as:

> data in electronic form attached to, or logically connected with, other electronic data and which serves as a method of authentication

What different things can you think of that might satisfy this rather vague notion of an electronic signature?

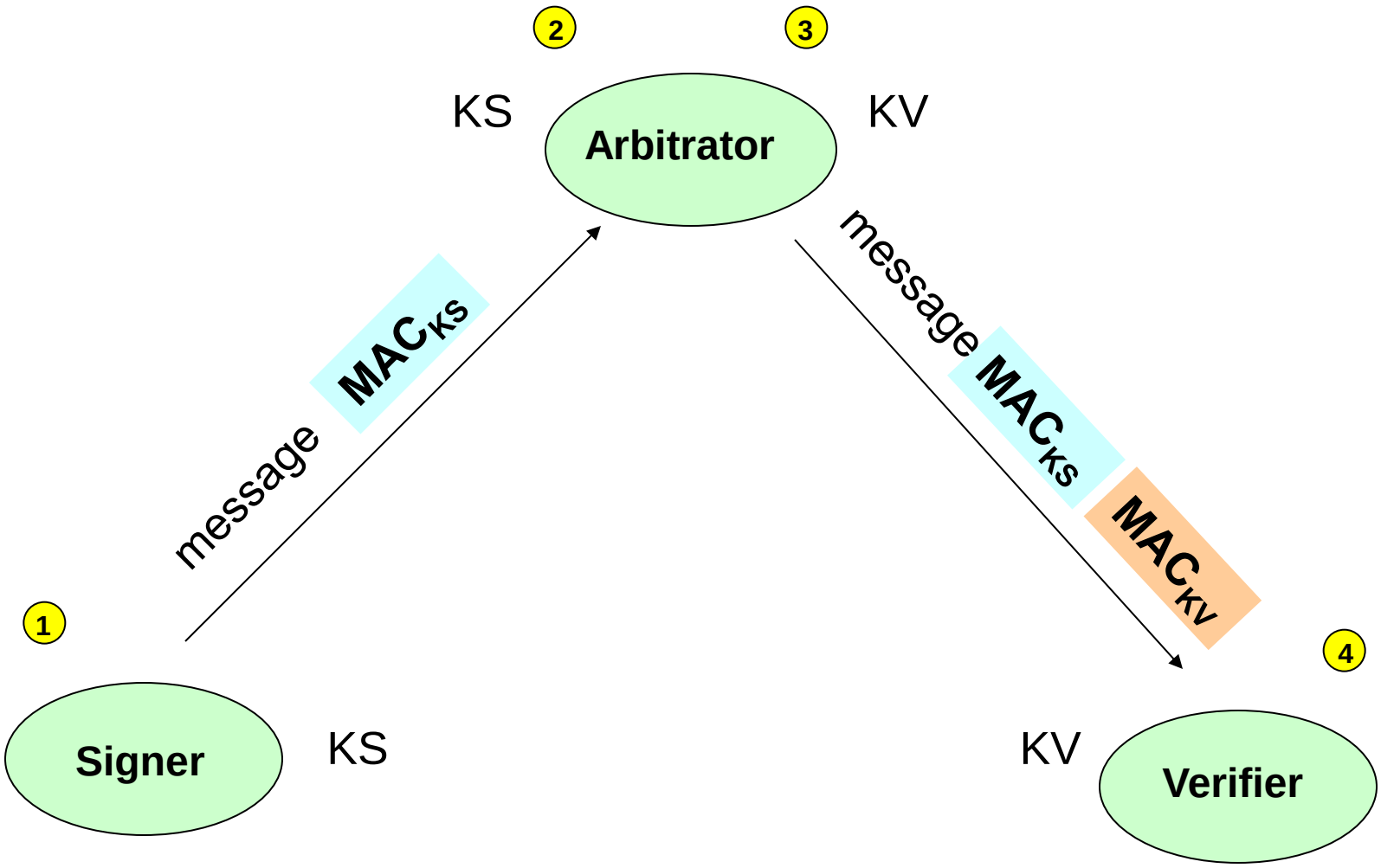
Security Requirements and Properties of DS

We define a **digital signature** on a message to be some data that have:

- Requirements:
  - Data origin authentication of the signer
  - Non-repudiation

- Properties:
  - Easy for the signer to sign a message
  - Easy for anyone to verify a message (efficient)
  - Hard for anyone to forge a digital signature
    - practically impossible for anyone who is not the legitimate signer to compute a digital signature on a message that appears to be valid.

Arbitrated Digital Signatures



Arbitrated Digital Signatures

(1) $X \rightarrow A: \ ID_X \| E(PR_x, [ID_X \| E(PU_y, E(PR_x, M))])$
(2) $A \rightarrow Y: E(PR_a, [ID_X \| E(PU_y, E(PR_x, M)) \| T])$

Public-Key Encryption, Arbiter Does Not See Message

Notations:

X=sender
Y=recipient
A=Arbiter
ID_X=ID of X

M=message
T=time stamp
PR_X=X's private key
PU_Y=Y's public key
PR_A=A's private key

Weakness: twice public-key encryptions on the message

Arbitrated Digital Signatures

1. Do arbitrated digital signatures meet the security requirements?
2. Do they have the properties that we required for a digital signature?
3. What is the main (practical) problem with implementing arbitrated signatures?

True Digital Signatures

A **true digital signature** is one that can be sent directly from the signer to the verifier.

True digital signature requirements	Public key encryption requirements
Only the holder of some secret data (private key) can sign a message	“Anyone” can encrypt a message
“Anyone” can verify that a signature is valid	Only the holder of some secret data (private key) can decrypt a message

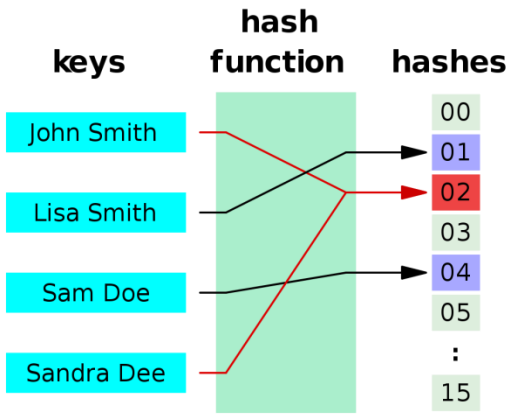
Hash Functions

Hash Functions

A **hash function** is a mathematical function that generally has the following three properties:

- **Condenses arbitrary long inputs into a fixed length output**
  - hash is much smaller than the data.
  - hash is a smaller thing that represents a larger thing - referred to as a **digest**, and the hash function as a **message digest function**.
- **Is one-way**
  - easy to compute, but very hard to recover the original data.
- **It is hard to find two inputs with the same output**
  - should be hard to find two different inputs (of any length) that when fed into the hash function result in the same hash (**collision free)**.

Hash Functions?



Consider the following two mathematical functions and explain whether they satisfy each of the properties of a hash function or not:

- Multiplying two prime numbers together
- Reducing a number modulo n

Practical Hash Functions?

There are several hash functions in common use that are believed to be secure enough for general use.

Can you name them?

Hash Functions and Data Integrity

“A hash function provides a weak notion of data integrity.”

If we had a list of MD5 hashes which contained information on all of our operating system files on our home computer you could verify the values of your files in the list and see which files have been changed or have been updated by say a virus.

BUT

If a virus replaced the system file it could also replace the MD5 values in your list with new ones and you would not be aware this had happened…

Hash Function Applications

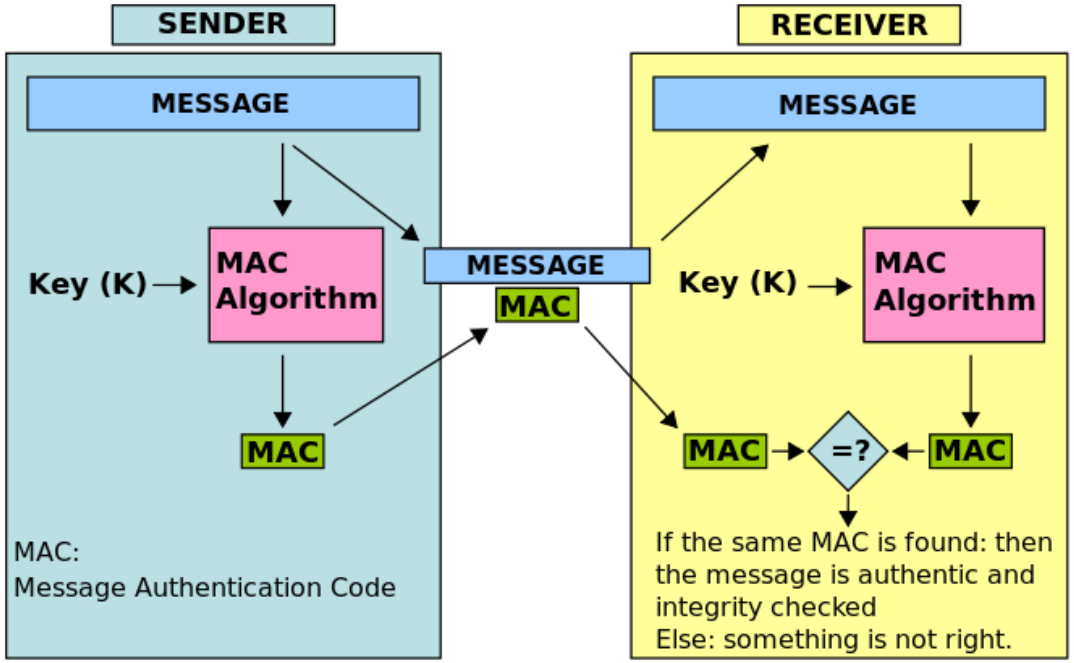
- **Digital signatures**: hash-functions are used to bind data together and make the signature process more efficient.
- **Password storage**: hash-functions are sometimes used to store highly confidential data such as passwords.
- **Cryptographic protocols**: hash-functions are often used within cryptographic protocols (including authentication protocols) to bind different data items together.
- **Hash-functions can be used as components** from which to construct other cryptographic primitives.

Is MAC a Hash Function?

- Have a fixed length output
- Rely on a symmetric key
- Provide data origin authentication (and data integrity)
- Typically constructed from block ciphers or hash functions

MAC vs Hash

- A virus can modify a file and its hash also (recalculate) → cannot detect tampering
- Virus can modify file but cannot calculate new MAC since it does not know the secret key



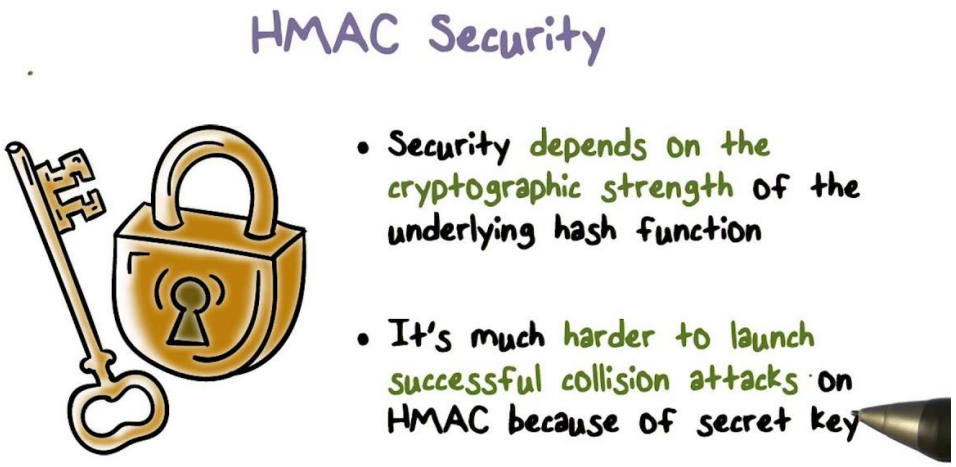
HMAC

Keyed hash of data

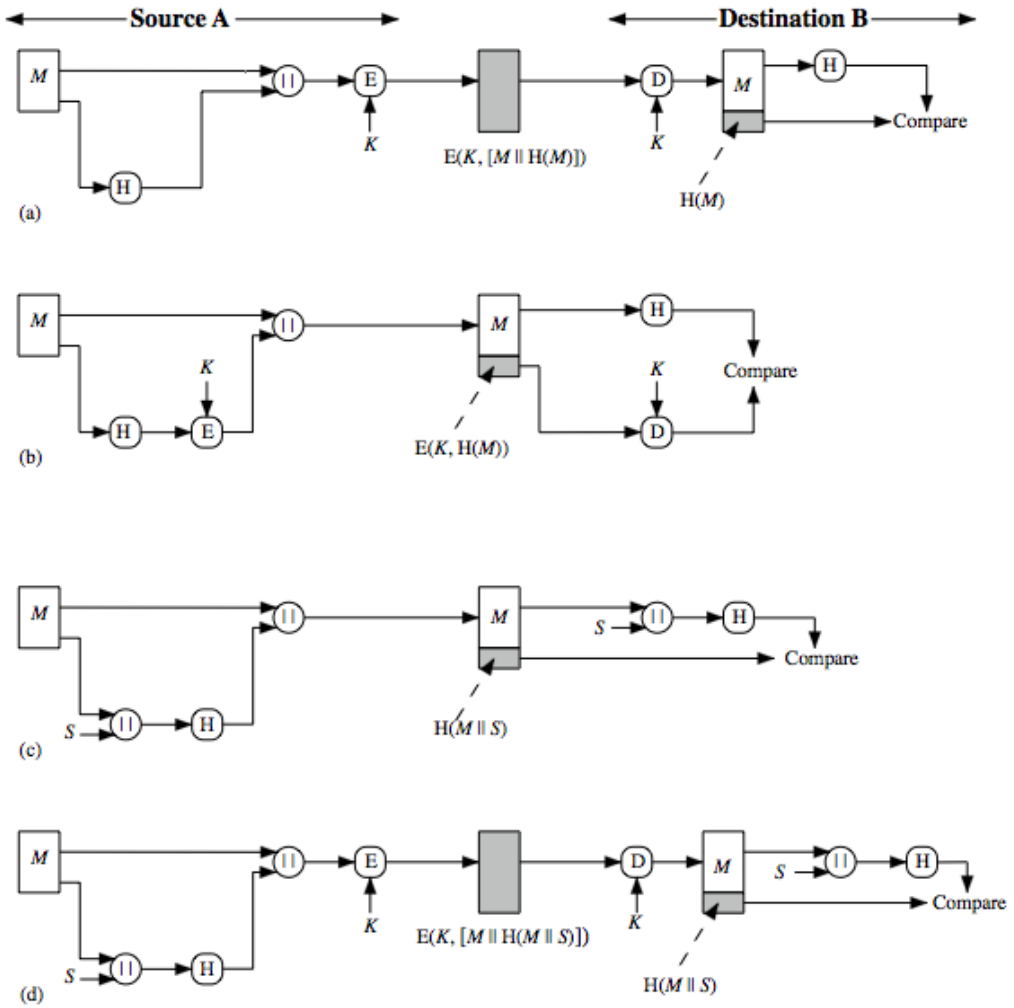
$HMAC=hash(k_2|hash(k_1|m))$

An HMAC provides collision resistance.
It also provides unforgeability.
In order to generate an HMAC, one requires a key.

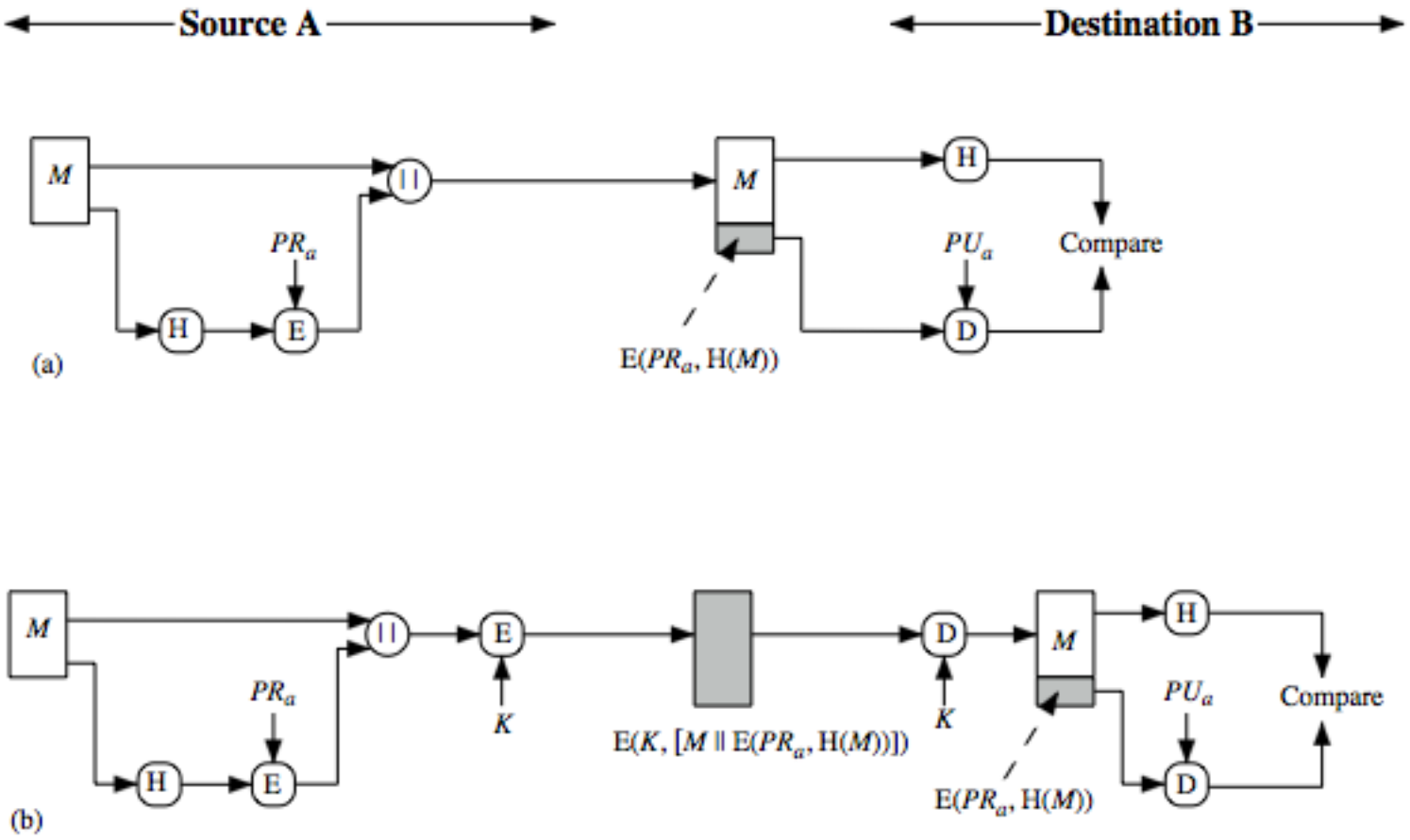
HMAC-MD5 and HMAC-SHA256 are specific MAC algorithms



Hash Functions & Encryption



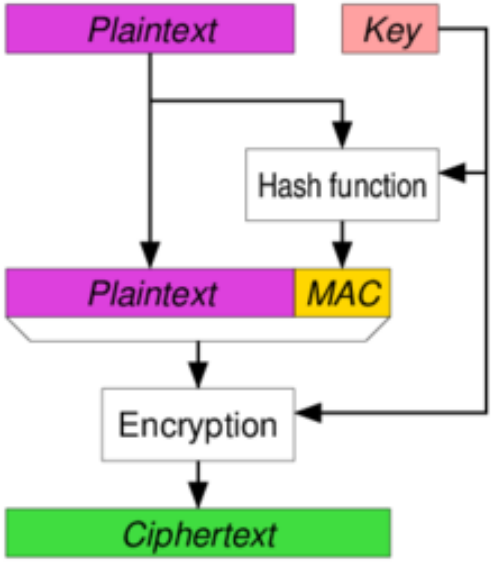
Hash Functions & Digital Signatures



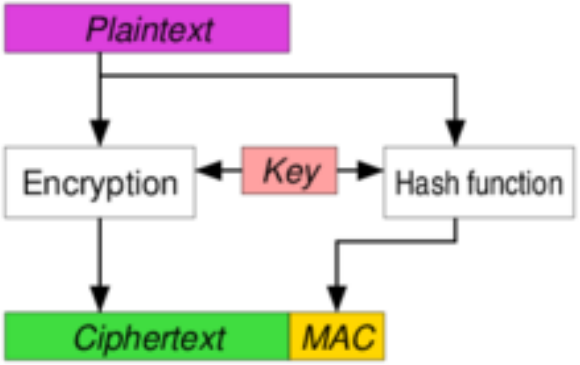
Authenticated Encryption

Confidentiality, integrity, and authenticity assurances on the data

- **MAC-then-Encrypt (MtE)**
  - A MAC is produced based on the plaintext, then the plaintext and MAC are together encrypted to produce a ciphertext.
  - Does not provide any integrity on the ciphertext
  - Used in **SSL/TLS**
  - Has not been proven to be strongly unforgeable in itself, but SSL/TLS implementation has been proven to be strongly unforgeable because of the fact that the **encoding** used alongside the MtE mechanism.
  - Despite the theoretical security, deeper analysis of SSL/TLS modelled the protection as **MAC-then-pad-then-encrypt**.

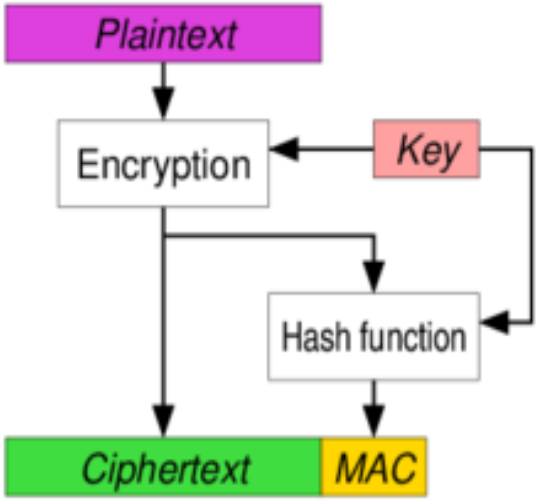


- **Encrypt-and-MAC (E&M)**
  - A MAC is produced based on the plaintext, and the plaintext is encrypted without the MAC.
  - No integrity on the ciphertext
  - Used in **SSH**.
  - MAC is taken against the plaintext. This opens the door to chosen-ciphertext attacks on the cipher.



Cont.

- **Encrypt-then-MAC (EtM)**
  - The plaintext is first encrypted, then a **MAC** is produced based on the resulting ciphertext.
  - Provides integrity of Ciphertext.
  - Used in **IPsec** (standard method according to ISO/IEC 19772:2009)
  - Highest definition of **security in AE**, but this can only be achieved when the MAC used is "strongly unforgeable".
  - **TLS and DTLS extension, SSHV2** use it.



- Associated data binds a ciphertext to the context
  - "cut-and-paste" a valid ciphertext into a different context can be detected and rejected.
  - ability to check the integrity and authenticity of associated data that is not encrypted.

- Example: encrypting the values that are inserted into a key/value database, and the record key is used as AD
  - must present the same key as AD to decrypt

Security of Hash Functions

Suppose that we sign the message **Keith owes Fred £10** by hashing it using a hash function that has a hash of just 2 bits:

there are only four possible hashes: 00, 01, 10 or 11.

Fred receives this signed message, and being a manipulative type he decides to change the message to **Keith owes Fred £100**. Of course Fred does not have Keith's signature key, so he cannot digitally sign this message. But he doesn't have to – he only has to sign the hash!

What is the probability that:

hash (**Keith owes Fred £10**) = **hash** (**Keith owes Fred £100**)?

Security of Hash Functions

Suppose a hash is 10 bits long – about 1024 hashes

1024 requests for £200

1. Pay Fred Piper £200
2. Pay F. Piper £200
3. Pay F.C. Piper two hundred pounds
4. Pay F.C. Piper two hundred pounds only
5. Pay two hundred pounds to Mr Fred Piper
6. ….

1024 request for £8000

1. Pay Fred Piper £8000
2. Pay F. Piper £8000
3. Pay F.C. Piper eight thousand pounds
4. Pay F.C. Piper eight thousand pounds only
5. Pay eight thousand pounds to Mr Fred Piper
6. ….

Since there are only 1024 different possible values of the hash, there is a very good chance that there will be at least one match…

Secure Hash Functions

Much shorter hashes than 160 bits are insecure.

Much longer hashes than 160 bits might be secure, but are not as efficient.

Now the standard is SHA256 – 256 bits.

Finding good hash functions has proven to be a significant challenge to cryptographers.