Miscellaneous Protocols

timestamping services
- most data, including digital signatures, must be timestamped

some challenges
- once timestamped, changing even a single bit of data must be disallowed
- faking a timestamp must be impossible
- overcoming these are difficult

some solutions
- arbitrated timestamping solution
  - a trusted arbitrator is used to maintain timestamp records
  - Alice sends her document to Trent
  - Trent records the date and time, and retains copy of the document
  - Trent is used to check the timestamp of a document
  - retaining a copy of all the documents he signs may end up being very expensive for Trent

- improved arbitrated timestamping solution
  - a trusted arbitrator is used, but only the hash value of the document is timestamped
  - Alice sends hash value to Trent
  - Trent appends the date and time when the hash value was received
    - he then digitally signs the hash value and timestamp with his private key
  - Trent sends the signed hash value with timestamp back to Alice
  - this can be used to check the timestamp of a document
  - Trent does not have to retain any information

- linking timestamping protocol
  - this uses a timestamping service (or arbitrator)
    - that timestamps documents relative to what was processed before the current document
  - this ensures correct timestamping by restricting the timestamp
    - based on the relative period of when it was processed
  - useful when a document changes (e.g., an evolving document)
    - e.g., timestamp is relative to the last version of the document

- distributed timestamping protocol
  - a timestamp is generated in cooperation with multiple parties
  - Alice generates *n* cryptographic random numbers, $v_{1,} v_{2,} \cdots, v_n$
  - Alice uses these random numbers as identities
    - of the parties who will be timestamping her document
  - she sends H_n, the hash value of her document, to those individuals
  - they then timestamp the document, sign it using their private key, and send this back to her
  - basically a distributed improved arbitrated timestamping solution

subliminal channel
- a subliminal channel is a covert communication channel
- it gives a way to communicate with another person secretly while some third-party may be listening
  - the third party (ideally) has no idea!

- protocol
  - Alice generates a random (or perhaps relevant) innocuous message

- the message is signed, and a secret message is hidden within the digital signature
- Alice sends the signed message to Bob through Mallory, a patsy
  - but Mallory only sees the innocuous message
  - she has no idea of the covert message hidden in the digital signature
- actually, this is a covert channel mixed with steganography!
  - a message is hidden within the digital signature
  - and it is sent through a covert channel

coin flipping
- there are some methods to simulate coin flipping
- requirements
  - Alice must flip a coin before Bob guesses
  - Alice must not be able to re-flip after Bob guesses
  - Bob must not know the outcome before guessing

- protocol
  - Alice selects a random number, *x*, and hashes it
  - Alice sends the hash to Bob
  - Bob guesses whether *x* is odd or even, and sends his guess to Alice
  - if Bob's guess is correct, the result is heads
    - otherwise, it is tails
  - Alice sends the result and *x* to Bob, so Bob can verify
    - Bob can hash *x* if desired to confirm the originally sent hash

- this reminds me of the ice cream truck problem
  - ask me about this!

the dining cryptographers problem
- three extraordinary cryptographers gather around a table for dinner
- the waiter informs them that someone has already paid on their behalf
- the payer may be one of the three cryptographers or an outsider
- while the cryptographers respect the will of someone to pay anonymously
  - they want to find out whether the payer is one of them or an outsider

- how can this be solved?

solution

the cryptographers agree to a protocol:

each cryptographer flips a coin once with the cryptographer to their right

each cryptographer knows two coin flip results

one with their left neighbor and another with their right neighbor

if two results match (both heads or both tails), the cryptographer yells "MATCHED"

we'll call this the same thing as producing a 1

if two results do not match, the cryptographer yells "MISMATCHED"

we'll call this the same thing as producing a 0

if a cryptographer secretly paid for dinner, s/he lies and yells the opposite

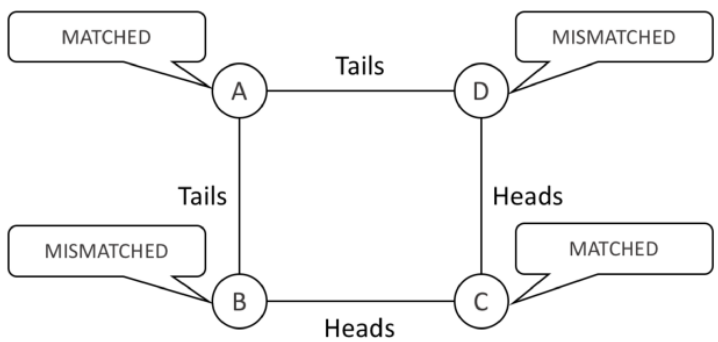
at the end, if the XOR of all the yelled results is 0, then the outsider paid

otherwise, one of the cryptographers paid

this works with any number of cryptographers

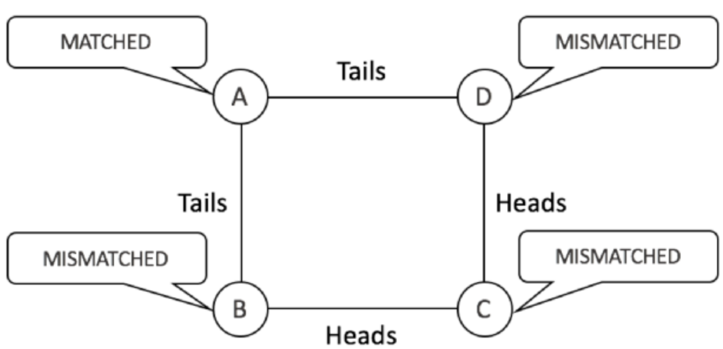
example

suppose that we have four cryptographers: A, B, C, and D (and none of them secretly paid)



$$1 \oplus 0 \oplus 1 \oplus 0 = 0$$

suppose that we have four cryptographers: A, B, C, and D (and one of them secretly paid)



$$1 \oplus 0 \oplus 0 \oplus 0 = 1$$

we are watching these cryptographers from a higher dimension

so we know that C paid for the dinner because he lied

the other cryptographers would have no idea of this, of course