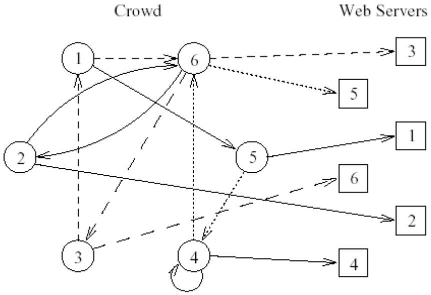
Anonymity and Pseudonymity

Anonymity, further approaches

M.Reiter, A. Rubin, 1998, Crowds: anonymity for Web Transactions

- Based on the idea “blending into a crowd”, that is hiding one’s actions within the actions of many others
- To execute a web transaction a user first joins a “crowd” of other users;
- Then the user’s request to a web server is passed to a random member of the crowd;
- That member can either submit the request to the server, or forward it to another randomly chosen member of the crowd and so on.

Crowds



Paths in a crowd. Picture by M.Reiter and A. Rubin

Privacy protection by the crowd

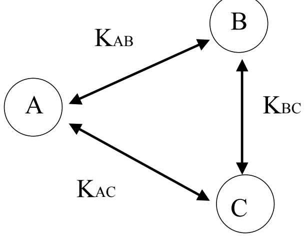
- When the request submitted to the end server, it is submitted by a **random** member of a crowd, so identity of an initiator is hidden (“in the crowd”) from an external observer
- Members of the crowd cannot identify initiator as well, they “just passing requests”

Crowds vs anonymizers and mixes

- Unlike an anonymizer crowds provide no single point, where an attacker can compromise anonymity of all users
- Crowds does not provide anonymity against a global adversary able to oversee all communications. In contrast, mix-networks protect anonymity in that case.
- Crowds admit very efficient implementations in comparison with mixes: no encryption/decryption operations, no inflation of message lengths.

DC-networks

D.Chaum, 1988: **D**ining **C**ryptographer networks



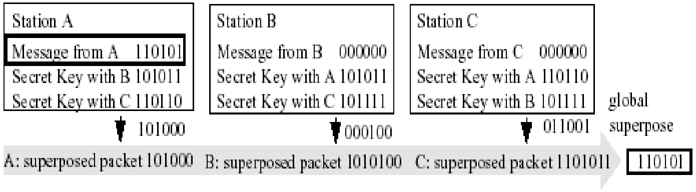
• At the preliminary stage between some pairs of nodes (at the picture between all) secret keys (sequences of bits) are exchanged

DC-networks

- To send a message M (sequence of bits), a node, say A, *broadcasts* the value **$(M +_2 K_{AB} +_2 K_{AC})$**, i.e. superposition of the message and all keys of A, here **$+_2$** stands for bitwise addition modulo 2 (or XOR operation)
- All other nodes broadcast superpositions of all their keys. So, B broadcasts **$(K_{AB}+K_{BC})$** and C broadcasts **$(K_{AC}+K_{BC})$**
- All nodes then superpose all received messages and get **$(M+_2K_{AB}+_2K_{AC}+_2K_{AB}+_2K_{BC}+_2K_{AC}+_2K_{BC}) = M$**

  (the initial message !!!)

DC-network



Station A
Message from A 110101
Secret Key with B 101011
Secret Key with C 110110

Station B
Message from B 000000
Secret Key with A 101011
Secret Key with C 101111

Station C
Message from C 000000
Secret Key with A 110110
Secret Key with B 101111

101000
000100
011001

A: superposed packet 101000 B: superposed packet 1010100 C: superposed packet 1101011

global superpose
110101

A message sent by A in the DC-network.
Picture by A.Pfitzmann

Anonymity by DC-networks

- DC-networks provide for *sender* anonymity because an adversary is unable to decide whether the packets he may observe contain a message or not;
- DC-networks can be used in combination with other mechanisms, such as mix-networks to enhance anonymity
- A major drawback is that DC-Networks require the preliminary stage exchanging the secret keys between participants
- Every round of communication requires a new set of keys
- Every node needs to participate every time a message is broadcasted => high load on the nodes => impractical in large networks

Broadcast and receiver anonymity

- Broadcast itself is a way to protect anonymity of a receiver: sender just broadcasts a message to some group of users, including intended receiver;
- Of course, as such it does not protect the content of communications;
- Better way: a sender broadcasts a message, encrypted in a way that **only** intended receiver can decrypt;
- It can be done by **public-key encryption** (we will discuss later on in the course)

Unlinkable anonymity

- Highest degree of anonymity (of communications, or transactions) is **unlinkable anonymity**
- Communications (transactions) provide unlinkable anonymity if
  - they do not reveal any information about identity of participants, and
  - There is no way to establish that a participant is the same participant that performed some other transaction
- Example: cash transactions

Linkable anonymity

- Unlinkable anonymity, often is too strong requirement
- Linkable anonymity appears when
  - No information about true identity of participants is revealed, but
  - Different transactions made by the same participant can be linked together
- Example: transactions by pre-paid telephone cards
- Transactions with linkable anonymity protect user privacy and at the same time allow to collect some (aggregate) information about users

Linkable anonymity and pseudonymity

- When transactions or communications provide with linkable anonymity, then we are dealing essentially with **pseudonyms**, which are
  - identifiers, linked to the true names (identities). In the phone card example, the number on the card can serve as the pseudonym of the card holder
- If the link between true names and identifiers (pseudonyms) is persistent and unforgeable, that is only a particular user (group of users) can use a pseudonym, we call such a property **persistent pseudonymity**, or just **pseudonymity**

Pseudonymity and Reputation

- Pseudonymity can protect privacy (no true identity revealed) and at the same time
- Allows to use pseudonyms to build **(digital) reputation** of an user (participant)
- Example:
  - in online auction user using particular pseudonym can be known as the trustable partner, who sells goods of good quality, etc
- Persistence of pseudonyms is important here

Communications using pseudonymes

Email pseudonym server **nym.alias.net**

- Allows anyone create an email pseudonym (alias, nym) without revealing his identity
- Nym appears as an ordinary email address to the rest of the world
- Nym.alias.net uses the anonymous remailer network as a mix-net, i.e. it forwards mail received for a nym through a sequence of independent remailers

Persistence

- Persistence of pseudonymity in nym.alias.net is achieved by using **public key** encryption
- Server can ensure the user is the same if it is able to decrypt the user signature by the key it has on file for the user
- A pair of suitable keys is established during the registration procedure
- We will return to the details later in the course