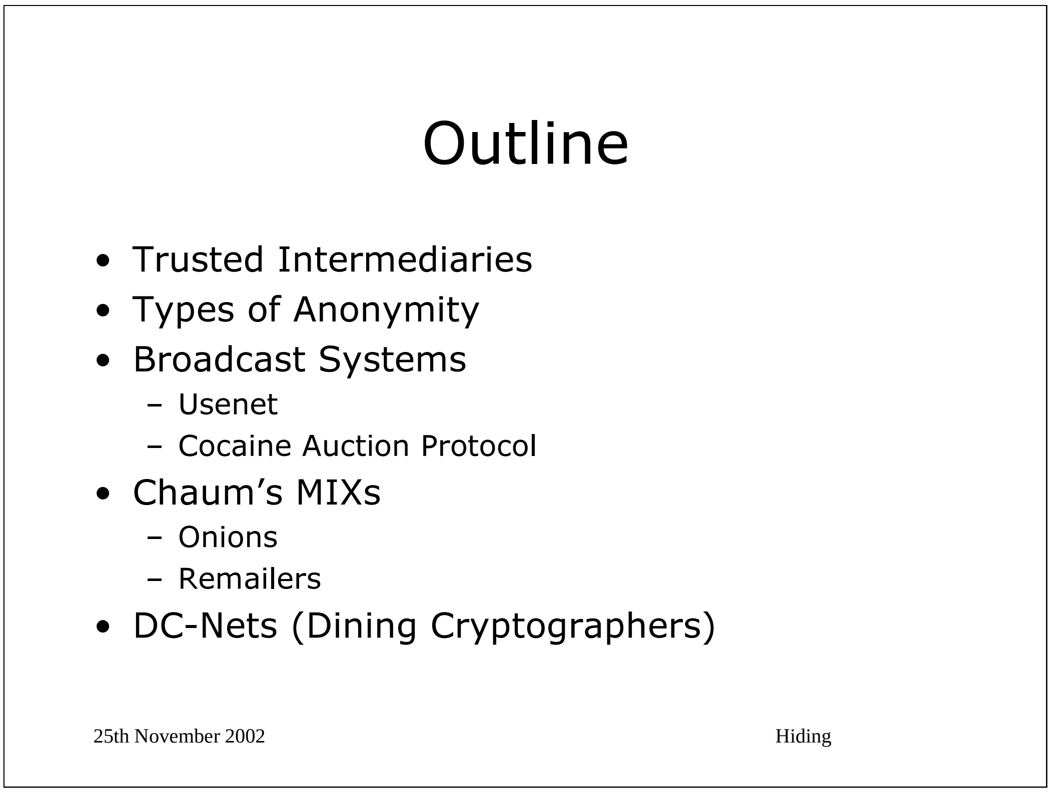
I: “Hiding”
Anonymity systems

25th November 2002

Richard Clayton

These lecture notes were specially prepared for the Cambridge University Computer Science “Additional Topics” course, Michaelmas Term 2002.

richard.clayton@cl.cam.ac.uk

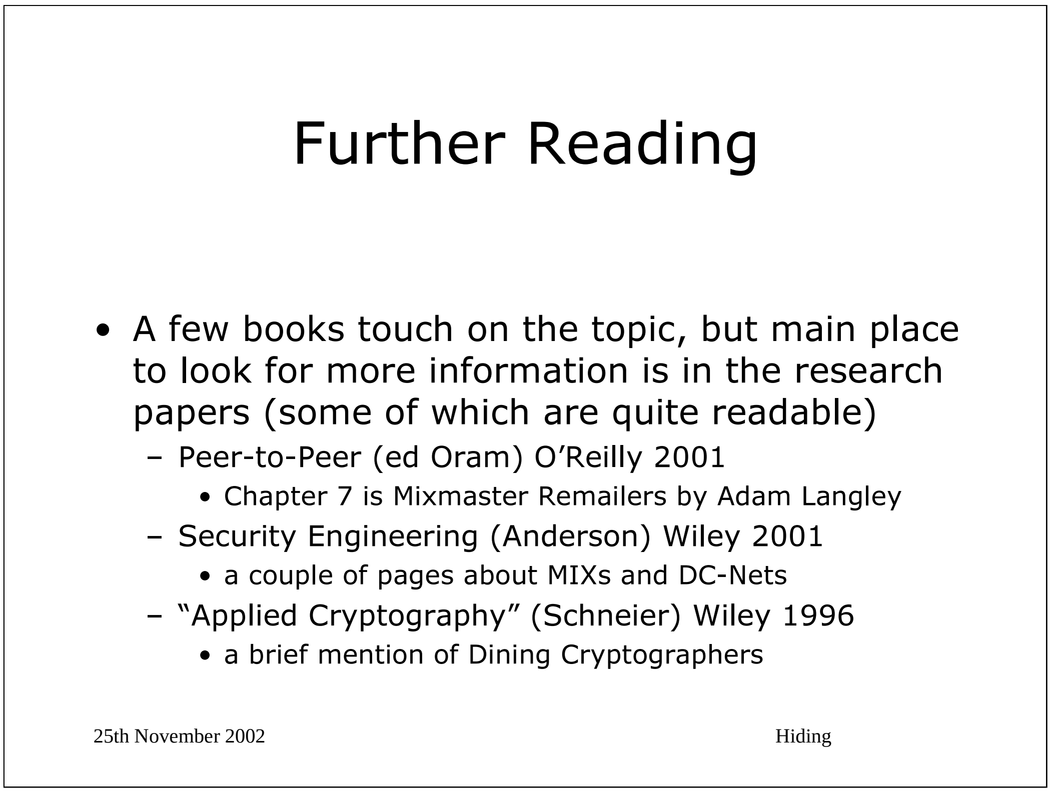


Outline

- Trusted Intermediaries
- Types of Anonymity
- Broadcast Systems
  - Usenet
  - Cocaine Auction Protocol
- Chaum's MIXs
  - Onions
  - Remailers
- DC-Nets (Dining Cryptographers)

25th November 2002 Hiding

The slides give the broad outline of the lectures and the notes ensure that the details are properly recorded, lest they be skipped over on the day. However, it is at least arguable that it will be far more interesting to take notice of what I say off-the-cuff rather than relying on this document as an accurate rendition of what the lecture was really about!



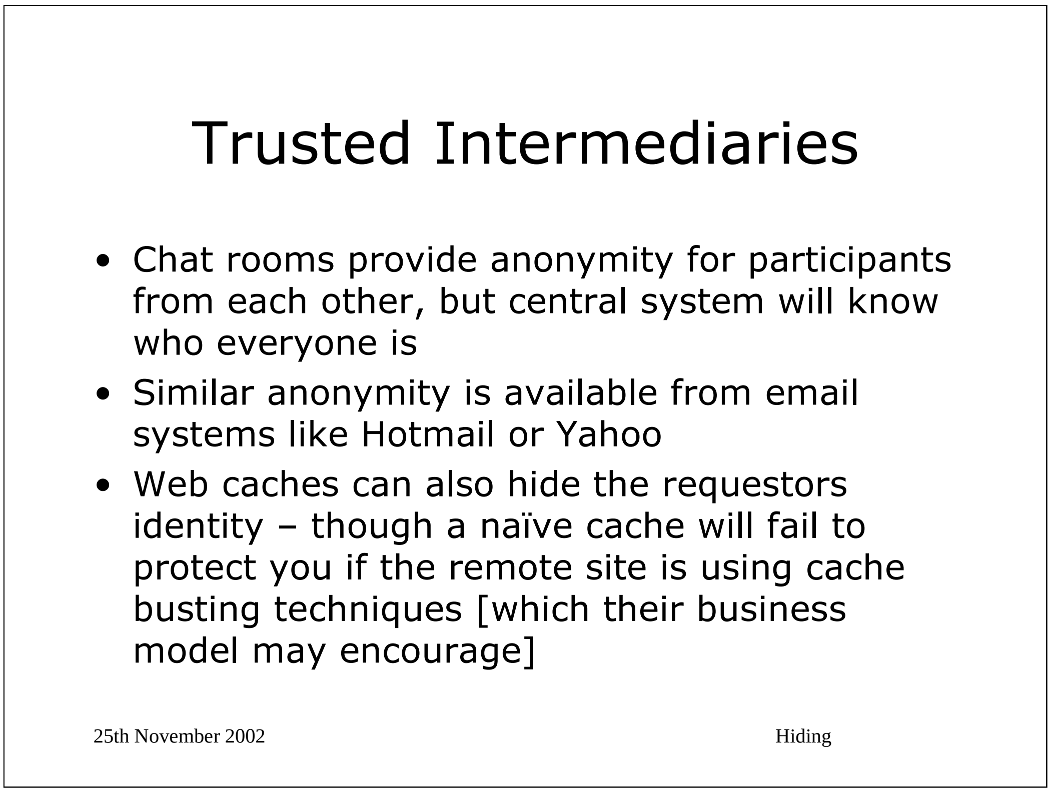
Further Reading

- A few books touch on the topic, but main place to look for more information is in the research papers (some of which are quite readable)
  - Peer-to-Peer (ed Oram) O'Reilly 2001
    - Chapter 7 is Mixmaster Remailers by Adam Langley
  - Security Engineering (Anderson) Wiley 2001
    - a couple of pages about MIXs and DC-Nets
  - "Applied Cryptography" (Schneier) Wiley 1996
    - a brief mention of Dining Cryptographers

25th November 2002 Hiding

Also, George Danezis keeps a bibliography of anonymity and pseudonymity systems papers:

http://www.cl.cam.ac.uk/~gd216/anonymity.html



Trusted Intermediaries

- Chat rooms provide anonymity for participants from each other, but central system will know who everyone is
- Similar anonymity is available from email systems like Hotmail or Yahoo
- Web caches can also hide the requestors identity – though a naïve cache will fail to protect you if the remote site is using cache busting techniques [which their business model may encourage]

25th November 2002 Hiding

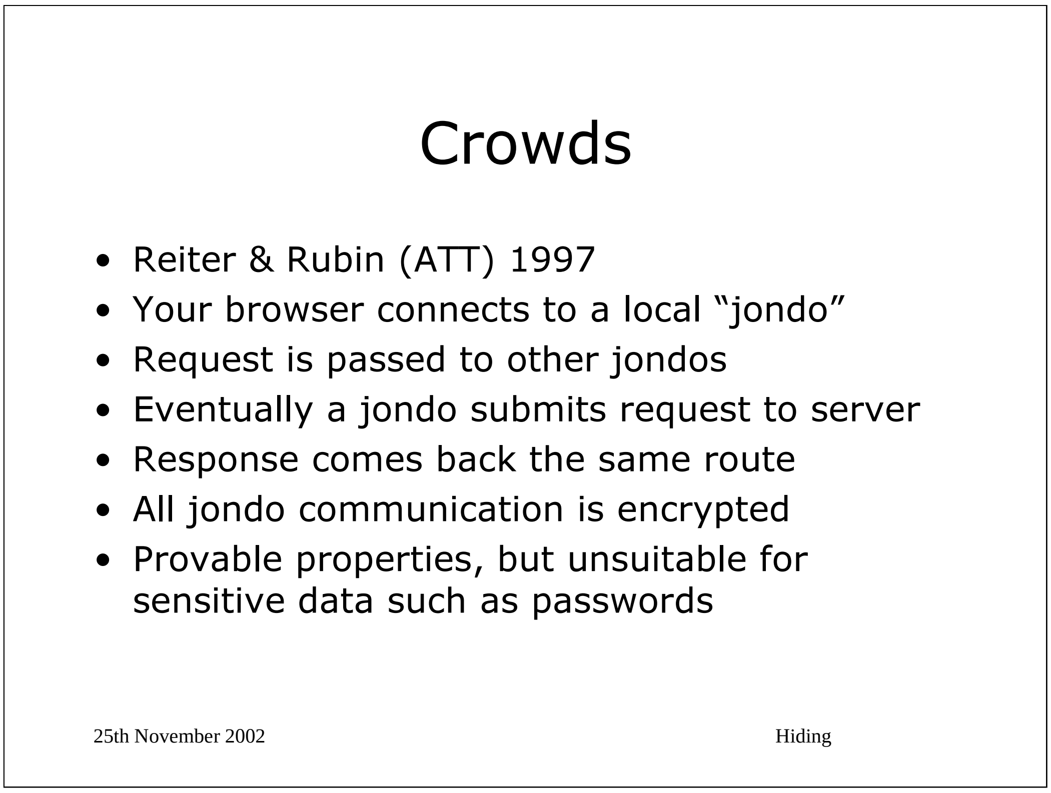
✶ Chat systems are extremely common on the Internet. Besides systems such as IRC or web-based chat on AOL, there are chat systems provided with most online games, and anyone who has played a MUD (Multi-User Dungeon) regularly will know that people come as much for the conversation as for the collecting the treasure, solving the puzzles or killing the dragons. These systems will usually provide “screen names” for the participants and their real IP address or email address will not be disclosed in public (though it will be apparent to the system operator).

✶ If you sign up for a Hotmail account (www.hotmail.com), then the sysadmins there will know where you connected from and the personal information you provided on your sign-up form. However, no-one else will be able to know who is actually behind a *aardvark512@hotmail.com* address (this may not be quite true for outgoing email because the IP address of the sender may be recorded in headers, but it’s certainly true for recipients).

✶ Caches hold local copies of remote content which can save external bandwidth and provide the information quickly. However, if the remote site has “banner ads” or is counting “hits” to demonstrate popularity then caches will reduce their income. Adding headers to say “do not cache” may not work against caches that themselves break the rules. Various tricks can be played with Java or JavaScript to create “dynamic pages” that caches cannot hold successfully, or you just give banner ad images unique names:

e.g.: http://www.clickz.com/tech/ad_tech/article.php/843731

✶ See also: http://www.cl.cam.ac.uk/~rnc1/Patterns_of_Failure.pdf



Crowds

- Reiter & Rubin (ATT) 1997
- Your browser connects to a local “jondo”
- Request is passed to other jondos
- Eventually a jondo submits request to server
- Response comes back the same route
- All jondo communication is encrypted
- Provable properties, but unsuitable for sensitive data such as passwords

25th November 2002 Hiding

✶ Crowds has its own set of web pages at:

http://www.research.att.com/projects/crowds/

there’s a technical paper, and also a slideshow and links to various articles about the system. “John Doe” is an American term for an unidentified person.

✶ Note that the system that submits a request will “carry the can” if that request is somehow dubious. The authors suggest, but do not of course guarantee, that showing the presence of the jondo will get you off the hook.

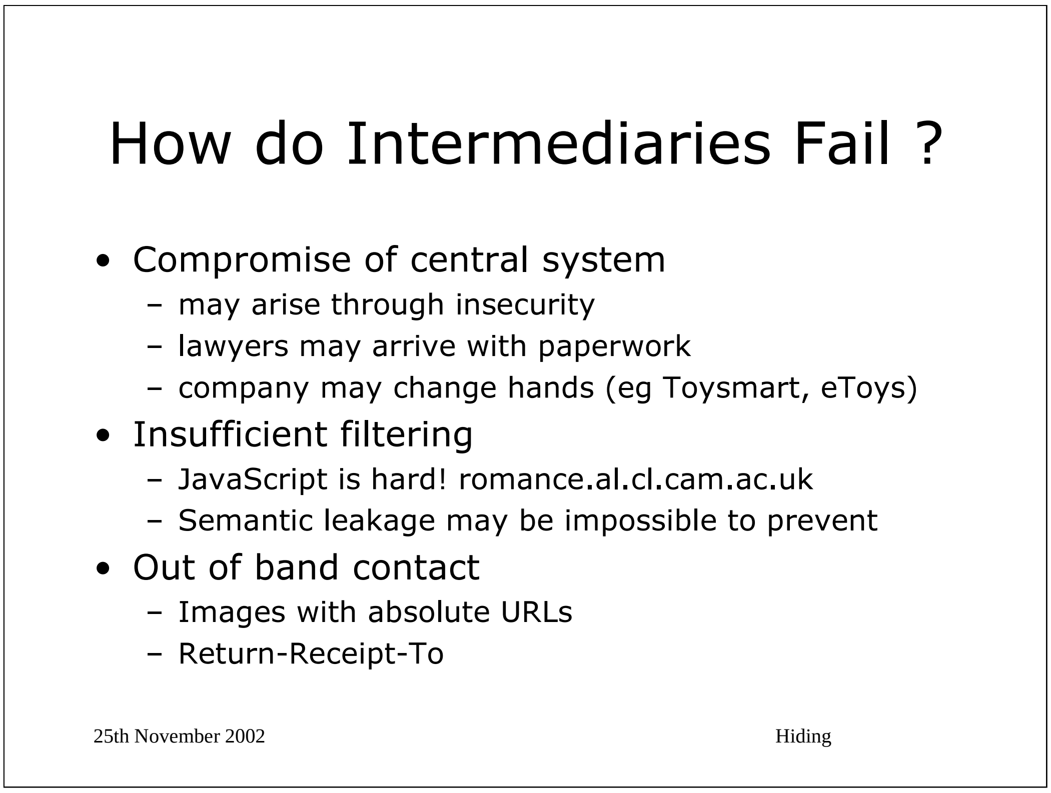
✶ Note that many URLs or transactions can carry sensitive information in the clear:

http://www.example.com/login.html?user=fred&password=secret

✶ The provability properties relate to the probability that a request will be submitted rather than passed to another jondo. If the probability is low then its less likely that the true initiator of the request is nearby.Equally, if lots of the jondos are snitches (and report to the same place) then you can calculate chance of being exposed.

✶ Some complications arise with embedded images. To avoid timing attacks (jondos at the start of the chain will see requests for the images arise very soon after page is delivered) the received page is parsed by the requestor and images are delivered along with the page.

✶ Crowds can be seen as an attack on targeted advertising (tracking individuals over multiple sites) as much as an anonymity system per se.



How do Intermediaries Fail ?

- Compromise of central system
  - may arise through insecurity
  - lawyers may arrive with paperwork
  - company may change hands (eg Toysmart, eToys)
- Insufficient filtering
  - JavaScript is hard! romance.al.cl.cam.ac.uk
  - Semantic leakage may be impossible to prevent
- Out of band contact
  - Images with absolute URLs
  - Return-Receipt-To

25th November 2002 Hiding

✶ For a longer discussion of this topic see section 4 of:

"Real World Patterns of Failure in Anonymity Systems", Richard Clayton, George Danezis and Markus Kuhn. Information Hiding Workshop, Pittsburgh 2001, in Ira S. Moskowitz (ed.): Information Hiding 2001, LNCS 2137.

online at: http://www.cl.cam.ac.uk/~rnc1/Patterns_of_Failure.pdf

✶ There are numerous examples of legal "attacks" on intermediaries. See for example:

http://www.epic.org/anonymity/aquacool_release.html

about Yahoo!

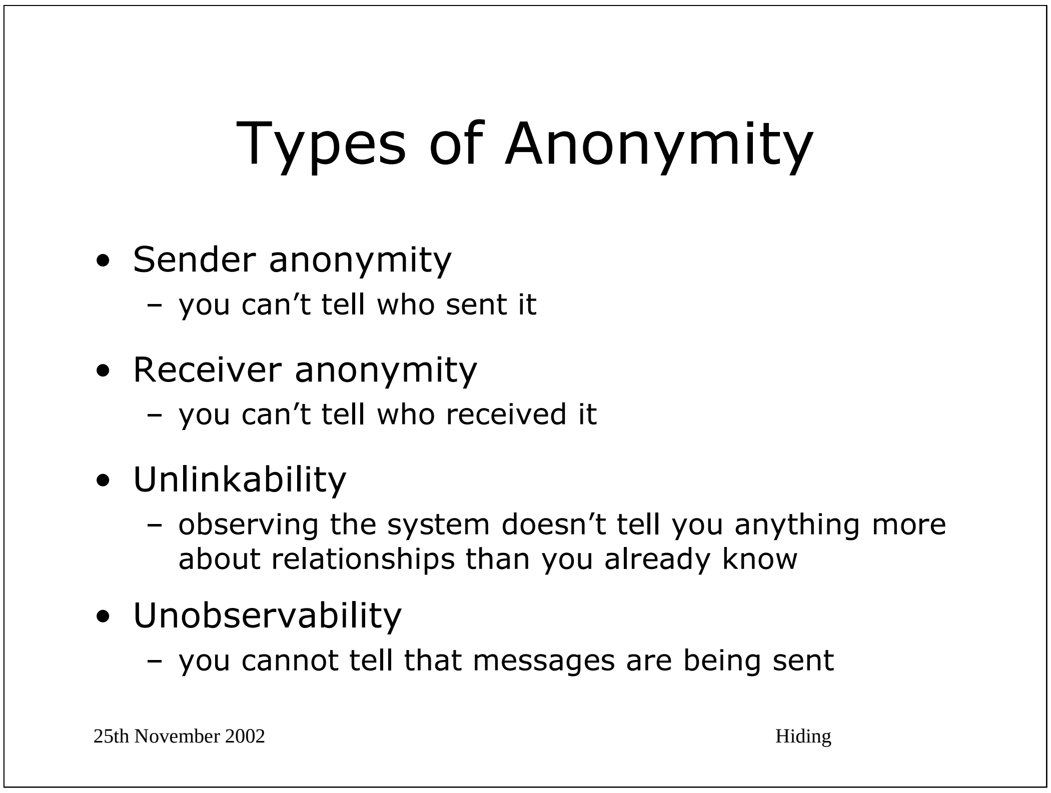
http://www.december.com/cmc/mag/1997/sep/helmers.html

about anon.penet.fi

http://www.cnn.com/2002/TECH/internet/11/05/aol.privacy.reut/

about AOL.COM

✶ For a discussion of the various attacks that worked against the Cambridge "Romance" server (romance.al.cl.cam.ac.uk) in Michaelmas 2000 then see the paper cited above (most of the simple issues are now fixed).



Types of Anonymity

- Sender anonymity
  - you can’t tell who sent it
- Receiver anonymity
  - you can’t tell who received it
- Unlinkability
  - observing the system doesn’t tell you anything more about relationships than you already know
- Unobservability
  - you cannot tell that messages are being sent

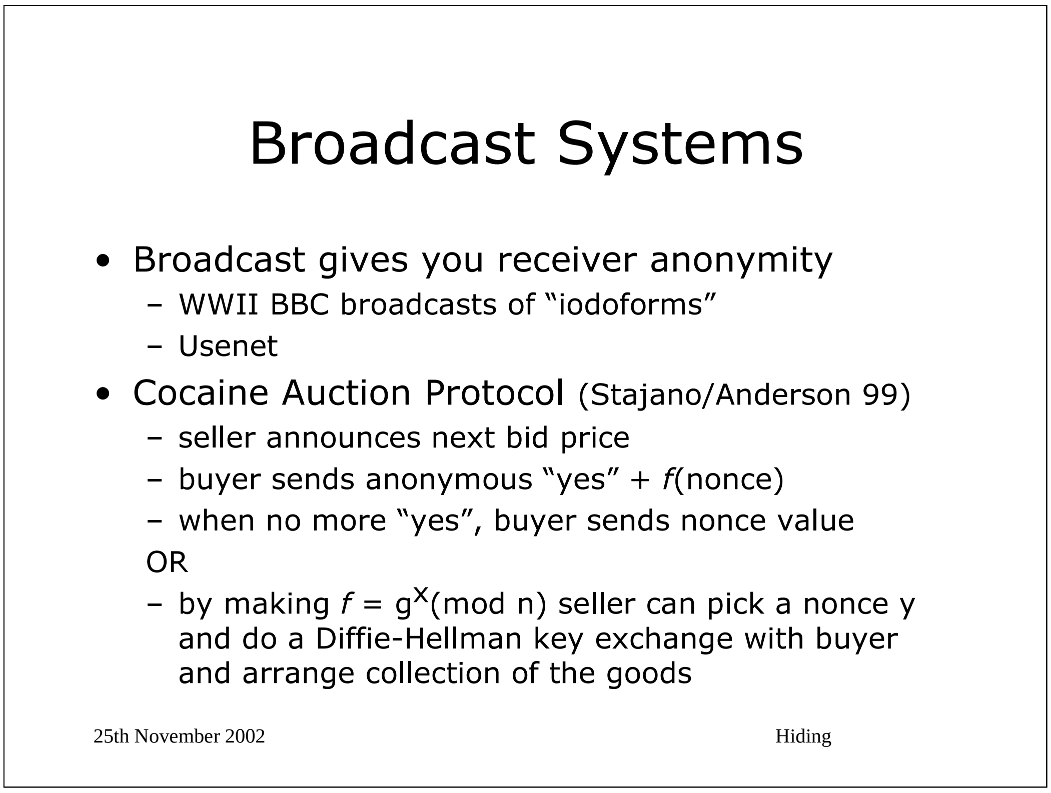
25th November 2002 Hiding

✶ For some more terms and a deeper analysis of the issues see:

“Anonymity, Unobservability and Pseudonymity – a Proposal for Terminology”, Andreas Pfitzmann and Marit Köhntopp in: Hannes Federrath (Hg.): Designing Privacy Enhancing Technologies; Proceedings Workshop on Design Issues in Anonymity and Unobservability; LNCS 2009; 2001; 1-9.

Slightly later version online as:

http://www.koehntopp.de/marit/pub/anon/Anon_Terminology.txt



Broadcast Systems

- Broadcast gives you receiver anonymity
  - WWII BBC broadcasts of "iodoforms"
  - Usenet
- Cocaine Auction Protocol (Stajano/Anderson 99)
  - seller announces next bid price
  - buyer sends anonymous "yes" + *f*(nonce)
  - when no more "yes", buyer sends nonce value

  OR
  - by making *f* = g^x(mod n) seller can pick a nonce y and do a Diffie-Hellman key exchange with buyer and arrange collection of the goods

25th November 2002 Hiding

✶ The WWII broadcasts that the BBC made on behalf of S.O.E. were called **iodoforms** "by someone with a classical education" (though apparently not a chemist since iodoform is CHI_3). Besides issuing coded commands they provided a way that agents could prove their bona fides (it would be arranged that a message of a doubter's choosing was broadcast later in the week). This proved to be of immense value in obtaining assistance from the locals.

The topic is covered in "Between Silk and Cyanide – a Codemaker's War 1941-1945", Leo Marks, HarperCollins 2000.

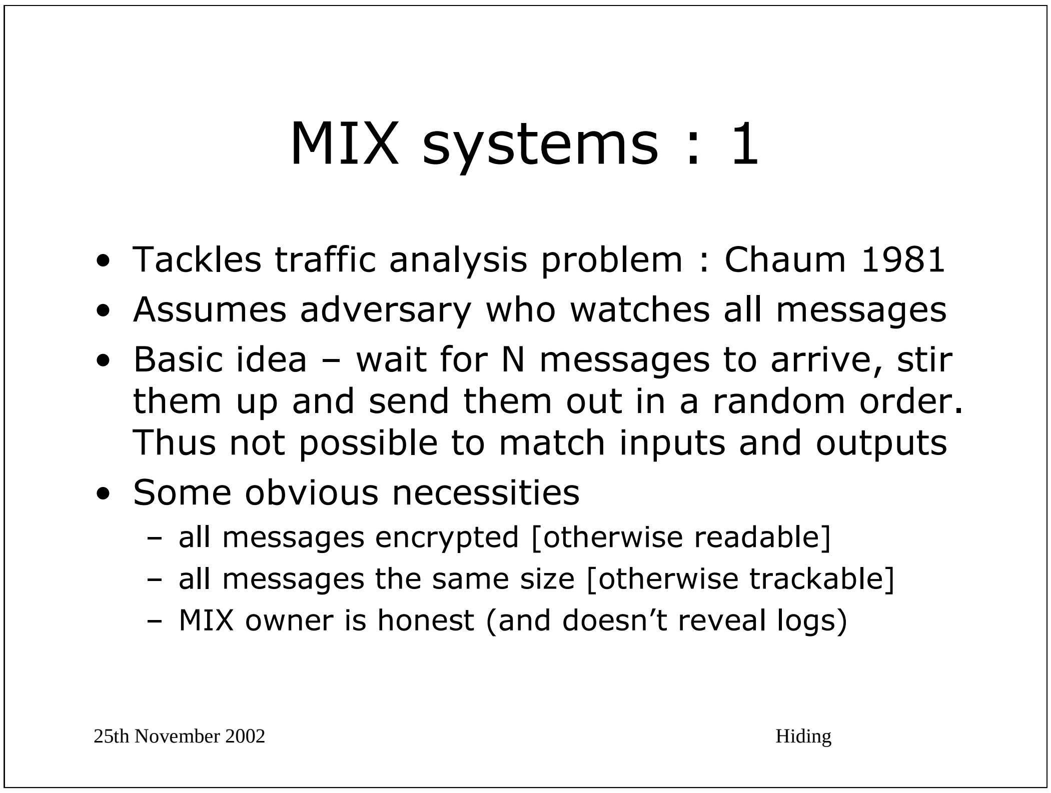
✶ The flood-fill algorithm used by Usenet is described in RFC977.

✶ "The Cocaine Auction Protocol: On The Power Of Anonymous Broadcast", Frank Stajano and Ross Anderson. Information Hiding Workshop, Dresden 1999 in A.Pfitzmann (ed), Information Hiding, 1999, LNCS 1768 pp434-447

http://www.cl.cam.ac.uk/~rja14/cocaine.pdf

The paper considers the protocol at rather more depth than is possible in the lecture. In particular it examines attacks such as what happens when the seller does not sell to the highest bidder, when the seller bids at his own auction and how to deal with "deadbeat bidders" who never show up with the money.

The paper also considers the issues surrounding broadcast as an anonymity primitive (raw broadcast not suitable for the Internet, but can be efficient on LANs or with short range wireless techniques – provided that attackers cannot use sophisticated electronics, such as direction finding kit, to monitor who is actually sending).



MIX systems : 1

- Tackles traffic analysis problem : Chaum 1981
- Assumes adversary who watches all messages
- Basic idea – wait for N messages to arrive, stir them up and send them out in a random order. Thus not possible to match inputs and outputs
- Some obvious necessities
  - all messages encrypted [otherwise readable]
  - all messages the same size [otherwise trackable]
  - MIX owner is honest (and doesn't reveal logs)

25th November 2002 Hiding

✶ Original paper is straightforward to read: "Untraceable electronic mail, return addresses, and digital pseudonyms", David Chaum, Communications of the ACM, 24(2), 1981, pp 84-88.

Online at: http://world.std.com/~franl/crypto/chaum-acm-1981.html

Also recommended is: "Mixing E-mail with BABEL", C.Gülcü & G.Tsudik, ISOC Symposium on Network and Distributed System Security, Feb *1996.*

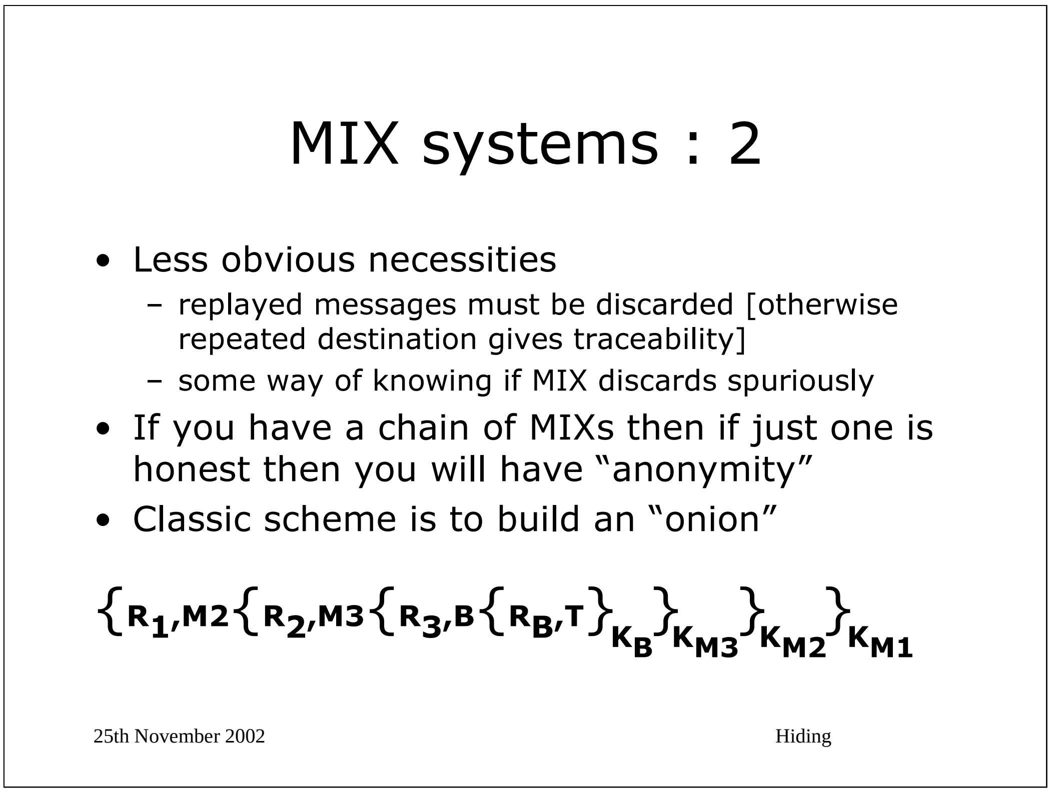
Online at: http://www.ics.uci.edu/~gts/paps/guts96.ps.gz

✶ The messages are encrypted not just to make them unreadable per se, but because otherwise it would be simple to detect which input corresponded to which output (the unlinkability property), and thereby trace the sender.

✶ If it takes a long time for N messages to arrive, then the MIX will do nothing and so messages can be delayed for a long time. Various schemes have been proposed for timed MIXs and "pool MIXs" where some messages are retained and new arrivals mixed in with them, so that the "anonymity set" is not just the most recent messages, but every message the MIX has ever received. For a modern review paper on MIX types see:

"From a Trickle to a Flood: Active Attacks on Several Mix Types", Andrei Serjantov, Roger Dingledine & Paul Syverson, in Fabien Petitcolas (Ed), Proc. 5th Workshop on Information Hiding, October 2002, LNCS (to appear)

http://www.cl.cam.ac.uk/~aas23/taxonomy.pdf



MIX systems : 2

- Less obvious necessities
  - replayed messages must be discarded [otherwise repeated destination gives traceability]
  - some way of knowing if MIX discards spuriously
- If you have a chain of MIXs then if just one is honest then you will have “anonymity”
- Classic scheme is to build an “onion”

$$\{R_1,M2\{R_2,M3\{R_3,B\{R_B,T\}_{K_B}\}_{K_{M3}}\}_{K_{M2}}\}_{K_{M1}}$$

25th November 2002 Hiding

✶ Chaum did not call the multi-mix messages “onions”. This term only became popular with the arrival of “onion routing” in the late 90s.

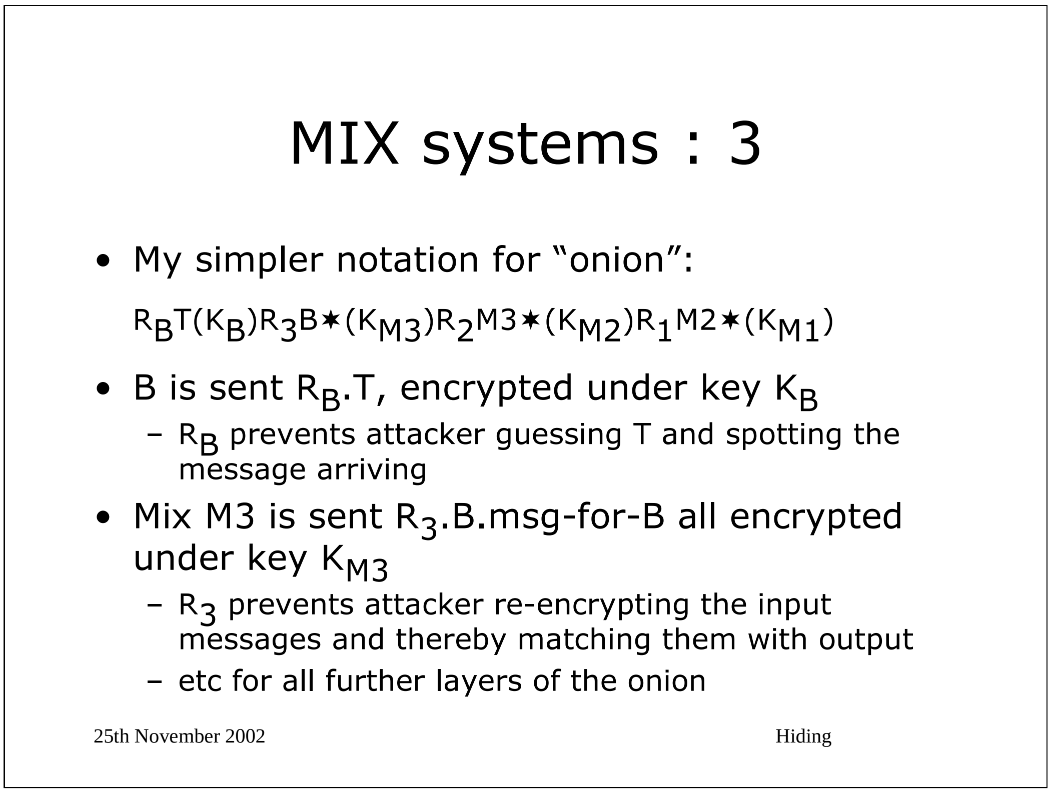
✶ Discarding duplicates can be very important. If a MIX has a batch size of 100 and, over time, distributes messages evenly to 10,000 different recipients then the probability that the next batch will contain another message to the same recipient as in this batch is about 1 in 100.

✶ Note that even if all MIXs in a chain are controlled by the NSA except for one, then the anonymity is still being achieved. One uses a chain to ensure that you aren’t trusting one particular system operator to be honest.

✶ The onion on the slide is sent to MIX M1, which will peel off a layer to reveal the next destination as M2. Then M2 will peel off the next layer and send it to M3. M3 will then discover the ultimate destination to be B. B will be able to remove the final layer of encryption and read the message text T.

The K_n values are encryption keys. It’s usual to use Public Key Encryption (such as RSA) so that messages can be encrypted to an entity’s public key and only they will hold the private key and be able to decrypt it.

The R_n values are random nonces, of which more below.



MIX systems : 3

- My simpler notation for "onion":

  $R_B T(K_B) R_3 B\ast(K_{M3}) R_2 M3\ast(K_{M2}) R_1 M2\ast(K_{M1})$
- B is sent R_B.T, encrypted under key K_B
  - R_B prevents attacker guessing T and spotting the message arriving
- Mix M3 is sent R_3.B.msg-for-B all encrypted under key K_{M3}
  - R_3 prevents attacker re-encrypting the input messages and thereby matching them with output
  - etc for all further layers of the onion

25th November 2002 Hiding

✶ There's nothing magic about the notation – it just avoids all the nested brackets! Read "(KEY)AB✶" as meaning "encrypt everything left of here using KEY. Then place A then B and then the encrypted value into a buffer".

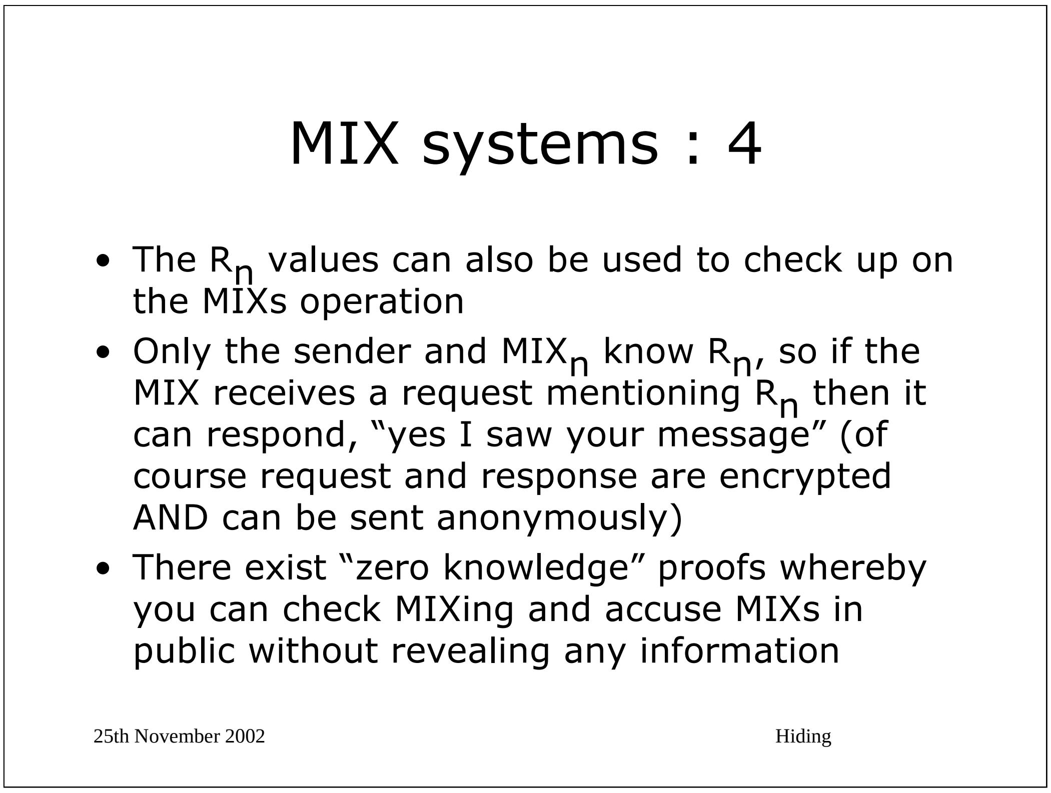
The example given is identical to the onion on the previous slide.

If you're a theoretician and don't care about practical details such as the exact order of the components of the packet sent to the next entity in the chain then you can ignore the ✶ altogether!

✶ The various "nonce" values R_1, R_2, R_3 and R_B are chosen randomly by the originator of the message.

R_B is present to ensure that the message delivered to B cannot be determined. Otherwise (in encryption schemes such as RSA) one could encrypt "flee at once, all is discovered" (or any other message) using K_B and spot its arrival.

The other values R_n are present to prevent an attacker from matching up any input and output packets.



MIX systems : 4

- The R_n values can also be used to check up on the MIXs operation
- Only the sender and MIX_n know R_n, so if the MIX receives a request mentioning R_n then it can respond, "yes I saw your message" (of course request and response are encrypted AND can be sent anonymously)
- There exist "zero knowledge" proofs whereby you can check MIXing and accuse MIXs in public without revealing any information

25th November 2002 Hiding

✶ The use of the R_n values for validation purposes is in the original Chaum paper, though implementations are rare.

✶ In practice, if a message is lost between one MIX and another then it may be very hard to determine which of them is actually at fault. This is of significant practical interest for remailers because they have turned out to be quite unreliable in practice, but it is hard to point at individual systems and blame them. For an attempt to address this see:

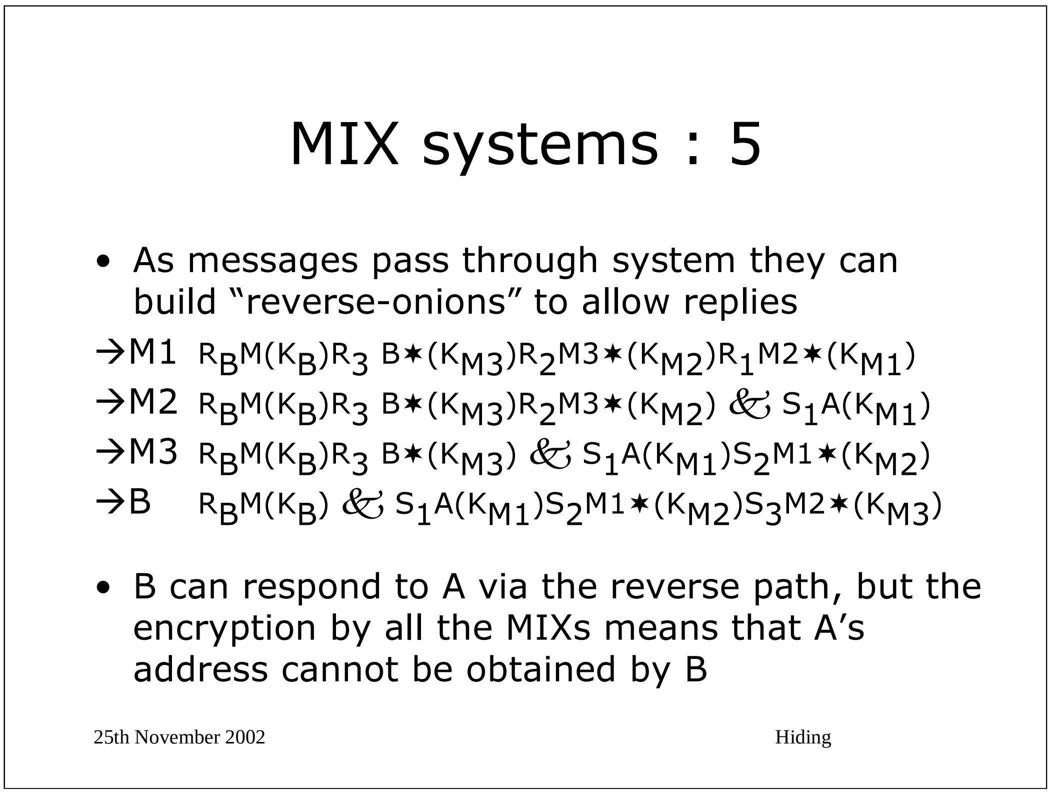
"A Reputation System to Increase MIX-net Reliability", Roger Dingledine, Michael J. Freedman, David Hopwood, and David Molnar. in Moskowitz (ed) Proceedings of the Information Hiding Workshop Pittsburgh, 2001.

✶ For zero-knowledge based MIXs (which are, in general, hard to understand (!) and hard to implement), see for example:

"Flash Mixing", Markus Jakobsson, in PODC'99, ACM, 1999, pp 83-89
Online at: http://www.rsasecurity.com/rsalabs/staff/bios/mjakobsson/flashmix/flashmix.pdf

[though later authors have found an attack on this, so some small changes are necessary in practice]

Also: "An efficient scheme for proving a shuffle", Jun Furukawa and Kazue Sako, in J. Kilian (ed): Advances in Cryptology – Proceedings CRYPTO 2001, Santa Barbara, LNCS 2139, 2001 pp 19-23



MIX systems : 5

- As messages pass through system they can build "reverse-onions" to allow replies

$\rightarrow$M1 $R_B M(K_B) R_3\ B\ast(K_{M3}) R_2 M3\ast(K_{M2}) R_1 M2\ast(K_{M1})$

$\rightarrow$M2 $R_B M(K_B) R_3\ B\ast(K_{M3}) R_2 M3\ast(K_{M2})$ & $S_1 A(K_{M1})$

$\rightarrow$M3 $R_B M(K_B) R_3\ B\ast(K_{M3})$ & $S_1 A(K_{M1}) S_2 M1\ast(K_{M2})$

$\rightarrow$B $R_B M(K_B)$ & $S_1 A(K_{M1}) S_2 M1\ast(K_{M2}) S_3 M2\ast(K_{M3})$

- B can respond to A via the reverse path, but the encryption by all the MIXs means that A's address cannot be obtained by B

25th November 2002 Hiding

✶ The way this works should be easy to see:

$$R_B M(K_B) R_3\ B\ast(K_{M3}) R_2 M3\ast(K_{M2}) R_1 M2\ast(K_{M1})$$

arrives at M1, and the encryption is removed to reveal

$$R_1 M2\ast$$

the nonce R_1 is discarded, M2 specifies the next destination and the rest of the packet ($\ast$) is the message sent to M2, viz:

$$R_B M(K_B) R_3\ B\ast(K_{M3}) R_2 M3\ast(K_{M2})$$

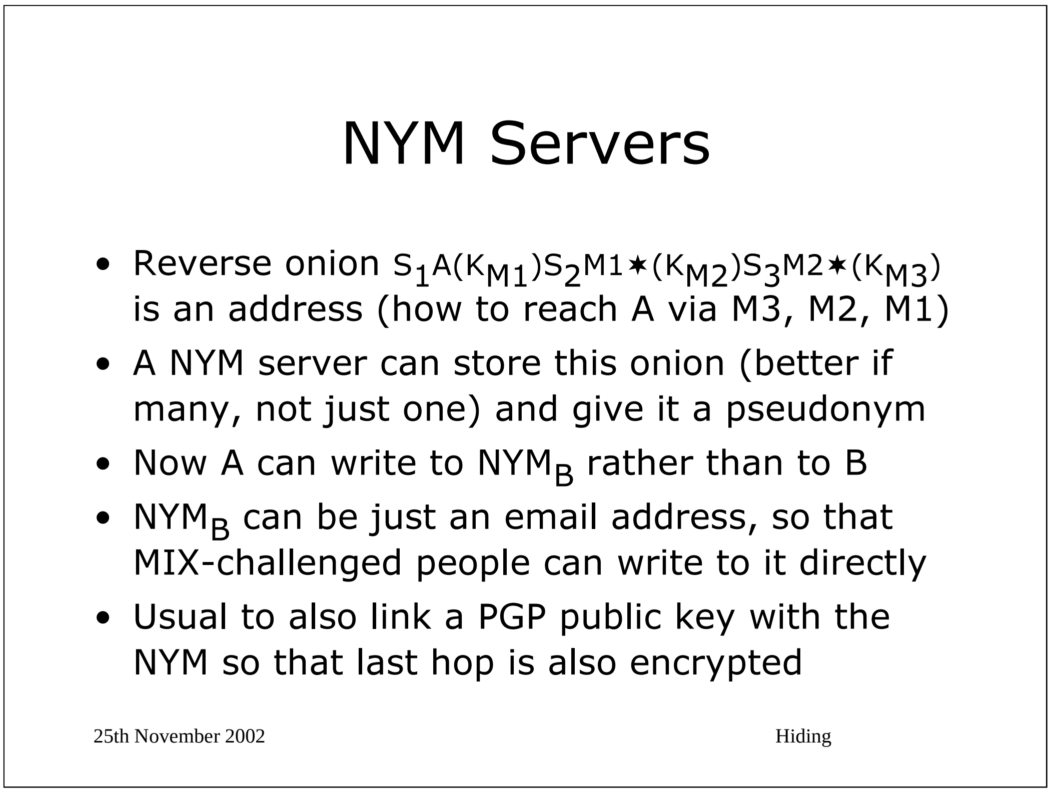
to this is appended the "reverse onion" of

$$S_1 A(K_{M1})$$

where the S_n values are more nonces. Similar actions are taken at each succeeding MIX. By taking care with padding values, it can be arranged that MIXs remain ignorant of their position on the chain.

✶ Note that by using reverse onions we have achieved "receiver anonymity" for messages back to A, but A still knows the identity of B.

✶ The reply message can be encrypted to a public key provided by A along with the outgoing message and it can be encrypted hop by hop by the MIX private keys and then decoded by A who knows which public keys to use. More neatly (and somewhat more securely), it is possible to avoid public key operations on the messages if A provides session keys within the onion to each of the MIXs in turn.



NYM Servers

- Reverse onion $S_1A(K_{M1})S_2^{M1\ast}(K_{M2})S_3^{M2\ast}(K_{M3})$ is an address (how to reach A via M3, M2, M1)
- A NYM server can store this onion (better if many, not just one) and give it a pseudonym
- Now A can write to NYM_B rather than to B
- NYM_B can be just an email address, so that MIX-challenged people can write to it directly
- Usual to also link a PGP public key with the NYM so that last hop is also encrypted

25th November 2002 Hiding

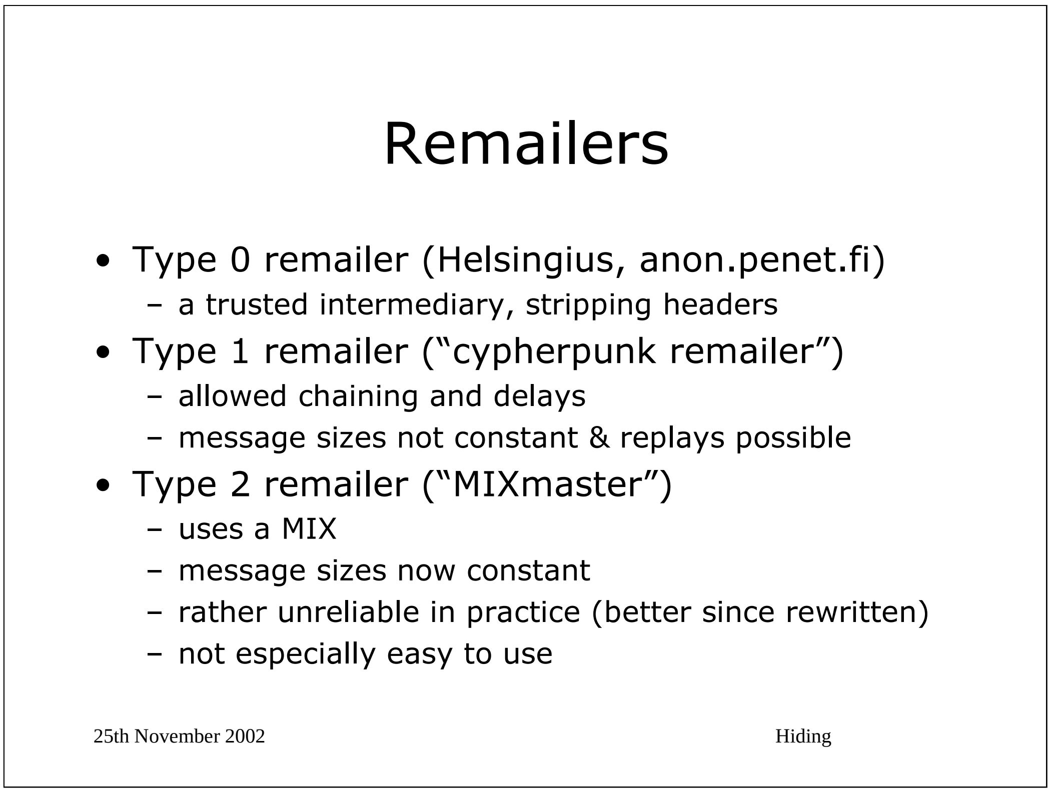
✶ With the use of a Nym server there is both sender anonymity and receiver anonymity.

✶ MIT’s Nym Server is described in:

“The design, implementation and operation of an email pseudonym server”, David Mazières and M. Frans Kaashoek in Proceedings of the 5th ACM Conference on Computer and Communications Security, 1998 available online as:

ftp://cag.lcs.mit.edu/pub/dm/papers/mazieres:pnym.ps.gz

✶ The reason for giving the NYM server multiple onions for it to use is to make it more difficult to construct probabilistic attacks based on observation of the MIX network as a whole. For a fully anonymous system the reverse onions should be seen as being for one time use only.



Remailers

- Type 0 remailer (Helsingius, anon.penet.fi)
  - a trusted intermediary, stripping headers
- Type 1 remailer (“cypherpunk remailer”)
  - allowed chaining and delays
  - message sizes not constant & replays possible
- Type 2 remailer (“MIXmaster”)
  - uses a MIX
  - message sizes now constant
  - rather unreliable in practice (better since rewritten)
  - not especially easy to use

25th November 2002 Hiding

✶ anon.penet.fi stripped all incoming headers, but recorded the “from” address. It automatically generated a pseudonym which could then be used by recipients for return email. The server would then deliver the email to the person who wrote the original message.

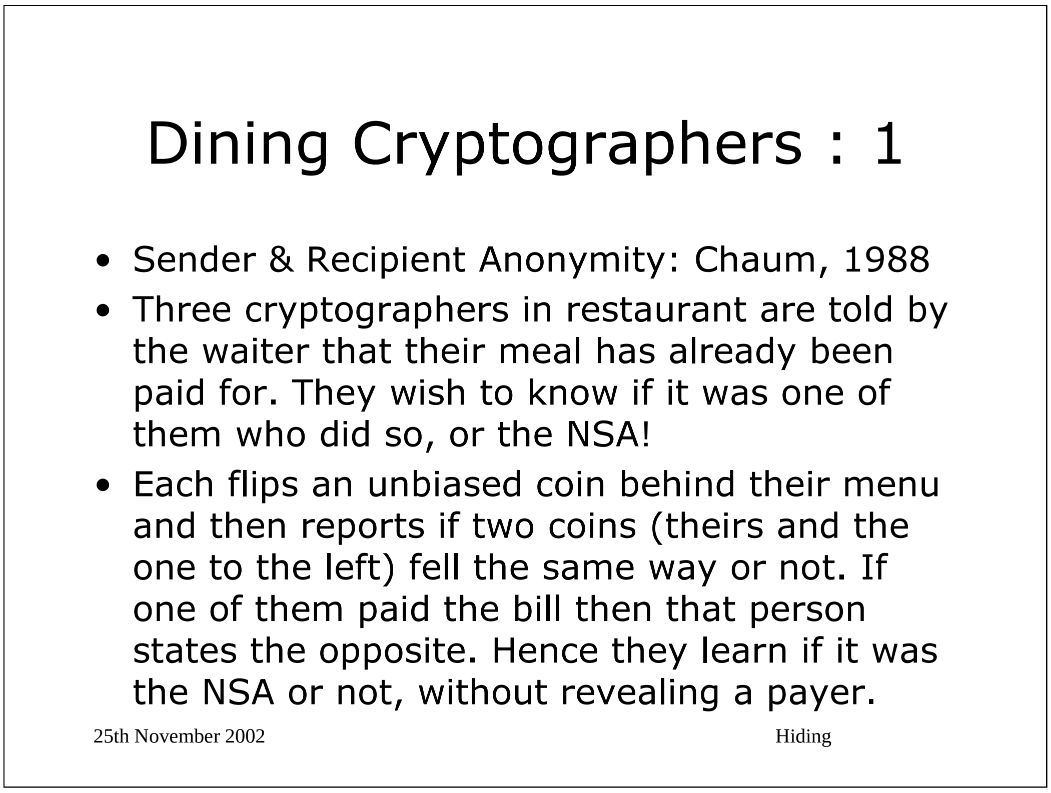
anon.penet.fi had strict limitations on message size, making it unsuitable for anything but short text messages. Attachments of pictures would be too big to be transmitted (see Friday’s lecture for the significance of this).

anon.penet.fi was shut down in 1996 when the Scientologists succeeded in a legal action to force the operator to divulge the real email address hidden behind a pseudonym.

✶ Cypherpunk remailers still exist, but are considered to be insecure against an attacker who can monitor their activities.

✶ There are about 30 MIXmaster remailers running, of which only about two thirds regularly achieve “4 nines” reliability.

✶ For a FAQ on remailers, nyms and how to use them, see:
http://www.eskimo.com/~turing/remailer/FAQ/



Dining Cryptographers : 1

- Sender & Recipient Anonymity: Chaum, 1988
- Three cryptographers in restaurant are told by the waiter that their meal has already been paid for. They wish to know if it was one of them who did so, or the NSA!
- Each flips an unbiased coin behind their menu and then reports if two coins (theirs and the one to the left) fell the same way or not. If one of them paid the bill then that person states the opposite. Hence they learn if it was the NSA or not, without revealing a payer.

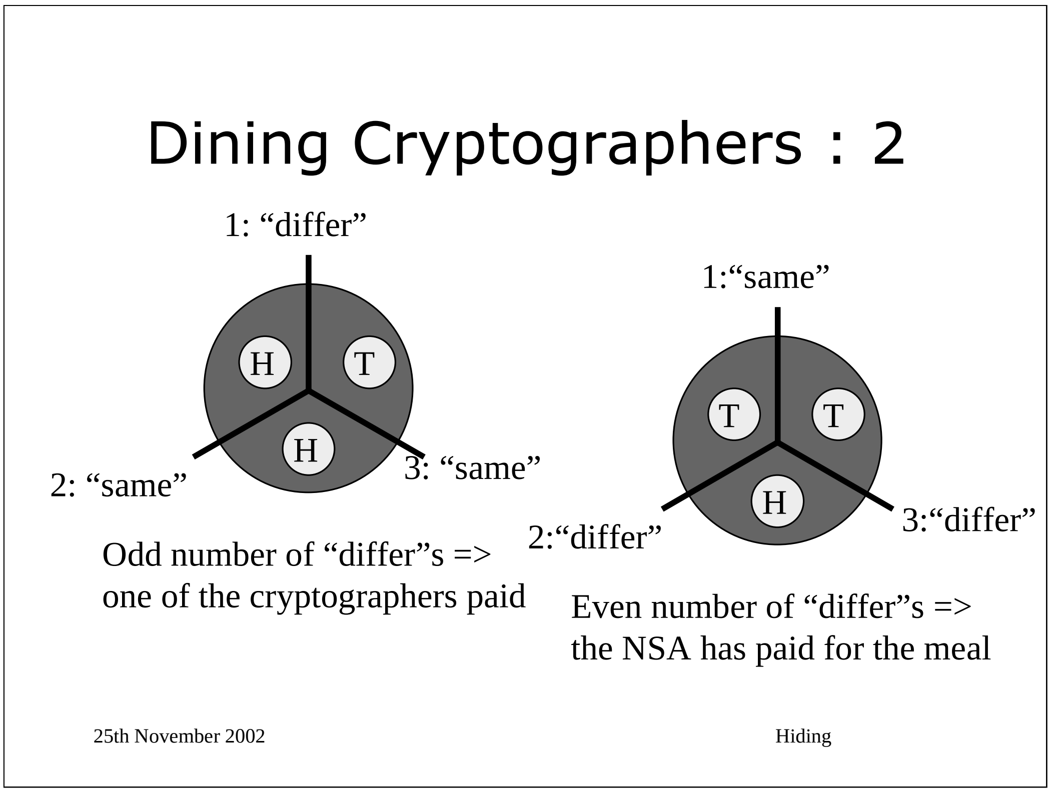
25th November 2002 Hiding

✶ The original paper starts off simply – then gets a bit more complex

"The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability", David Chaum, Journal of Cryptology, 1/1, 1988, pp 65-75. online as:

http://komarios.net/crypt/diningcr.htm

✶ The reason for Chaum choosing the US National Security Agency (NSA) as "the enemy" is because of their long term role in attempting to suppress cryptographic research in the open community. See for example, Steven Levy's book "Crypto: How the Code Rebels Beat the Government Saving Privacy in the Digital Age" Penguin, 2002



Dining Cryptographers : 2

1: “differ”
H
T
H
2: “same”
3: “same”
1:“same”
T
T
H
2:“differ”
3:“differ”

Odd number of “differ”s => one of the cryptographers paid

Even number of “differ”s => the NSA has paid for the meal

25th November 2002 Hiding

✶ There is a Java demo of the Dining Cryptographer’s Problem at:

http://komarios.net/crypt/dc-demo.htm

✶ Rather than drawing lots of restaurant tables, one can understand what is going on as follows:

Suppose that one views the coins (c1, c2, c3) as bit values 0 or 1

Reporting “same” or “different” can be viewed as reporting their sum mod 2.

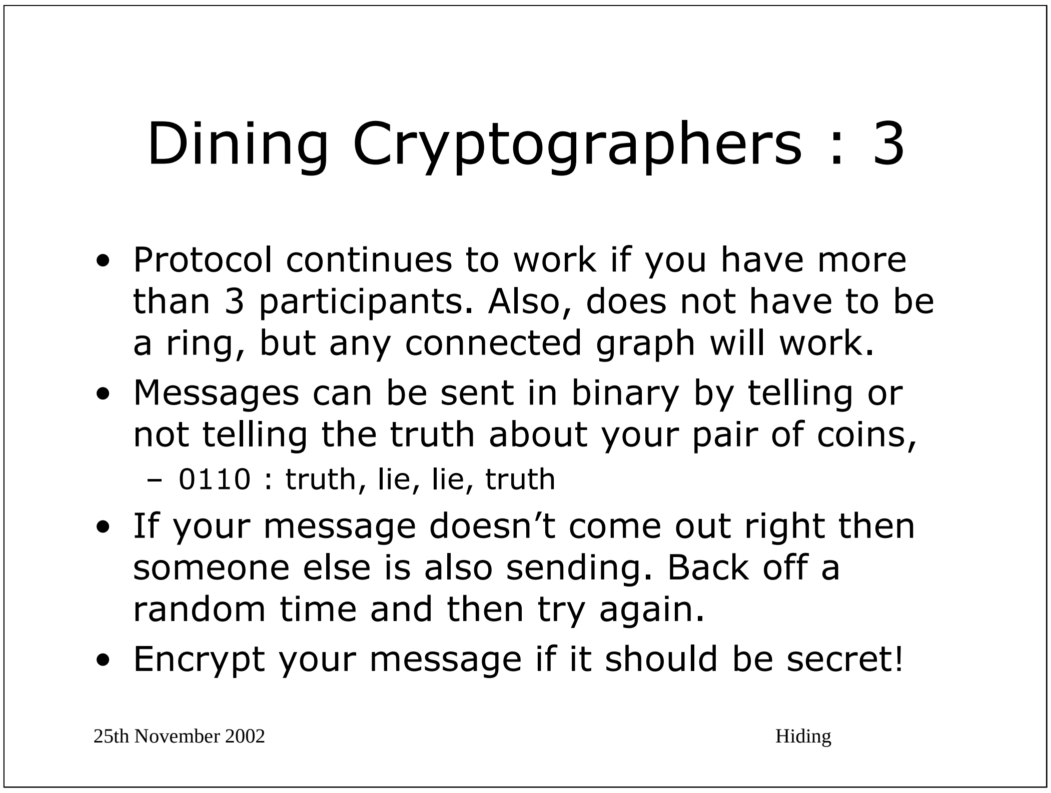
i.e.: the cryptographers report results R1, R2, R3 as follows:

$$R1 = c1 + c2 \pmod 2$$
$$R2 = c2 + c3 \pmod 2$$
$$R3 = c3 + c1 \pmod 2$$

Adding these all up, if no-one lies then one gets

$$\begin{aligned} R1 + R2 + R3 &= c1 + c2 + c2 + c3 + c3 + c1 \pmod 2 \\ &= 0 \end{aligned}$$

and if someone lies then the total will be 1 (mod 2)



Dining Cryptographers : 3

- Protocol continues to work if you have more than 3 participants. Also, does not have to be a ring, but any connected graph will work.
- Messages can be sent in binary by telling or not telling the truth about your pair of coins,
  - 0110 : truth, lie, lie, truth
- If your message doesn't come out right then someone else is also sending. Back off a random time and then try again.
- Encrypt your message if it should be secret!

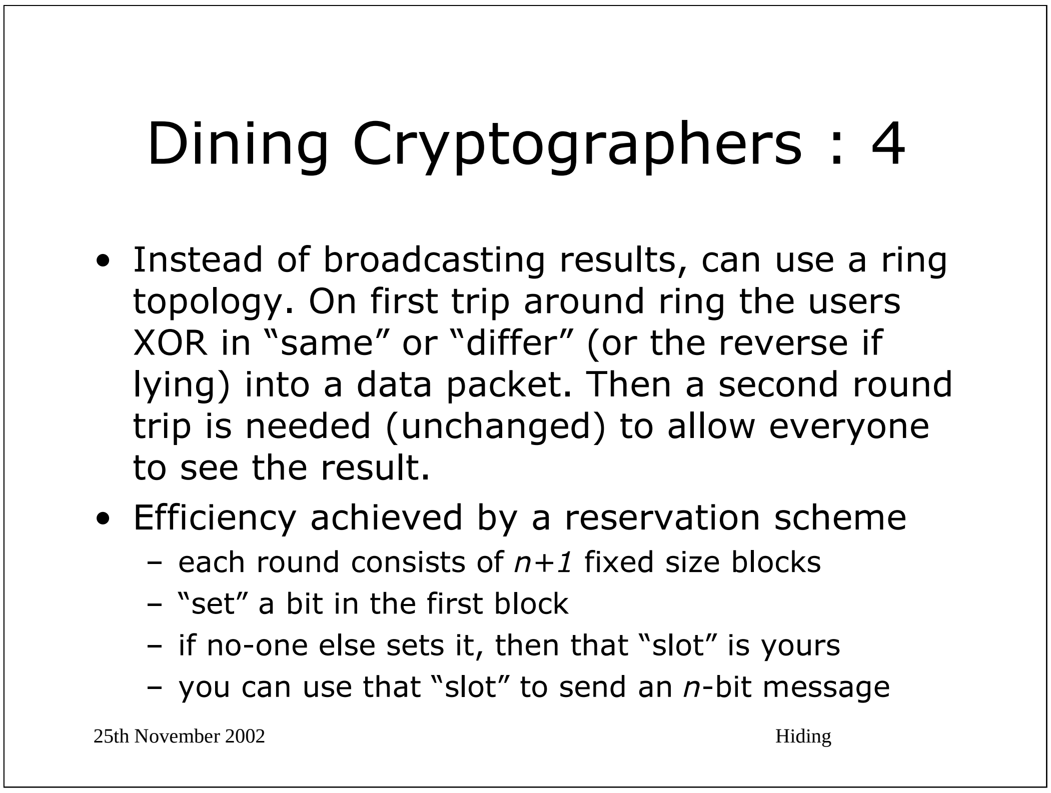
25th November 2002 Hiding

✶ It's obviously undesirable to continue to think in terms of a restaurant table, so you can replace the cryptographers with the nodes of a graph and view the edges of the graphs as representing the information sharing between them which allows them to report "same" or "different". Each node reports the sum (mod 2) of all the information that flows into that node (combined with their own node's state).

✶ One can replace the coin tosses by shared secrets, perhaps as initial values for programmable random number generators or stream ciphers. The bits that emerge are used to specify "heads" or "tails". Thus one always knows what state a neighbours "coin" is supposed to be in.

✶ Random backing off after a collision is a standard way of dealing with access to a shared resource. Ethernet uses CSMA/CD to solve a similar problem.

✶ DC-Nets are merely a transport mechanism. If you encrypt your data then you still have the key distribution problem (for symmetric crypto) or the certification problem (for asymmetric crypto) You must solve these problems before you can use encryption on your network!



Dining Cryptographers : 4

- Instead of broadcasting results, can use a ring topology. On first trip around ring the users XOR in "same" or "differ" (or the reverse if lying) into a data packet. Then a second round trip is needed (unchanged) to allow everyone to see the result.
- Efficiency achieved by a reservation scheme
  - each round consists of *n+1* fixed size blocks
  - "set" a bit in the first block
  - if no-one else sets it, then that "slot" is yours
  - you can use that "slot" to send an *n*-bit message

25th November 2002 Hiding

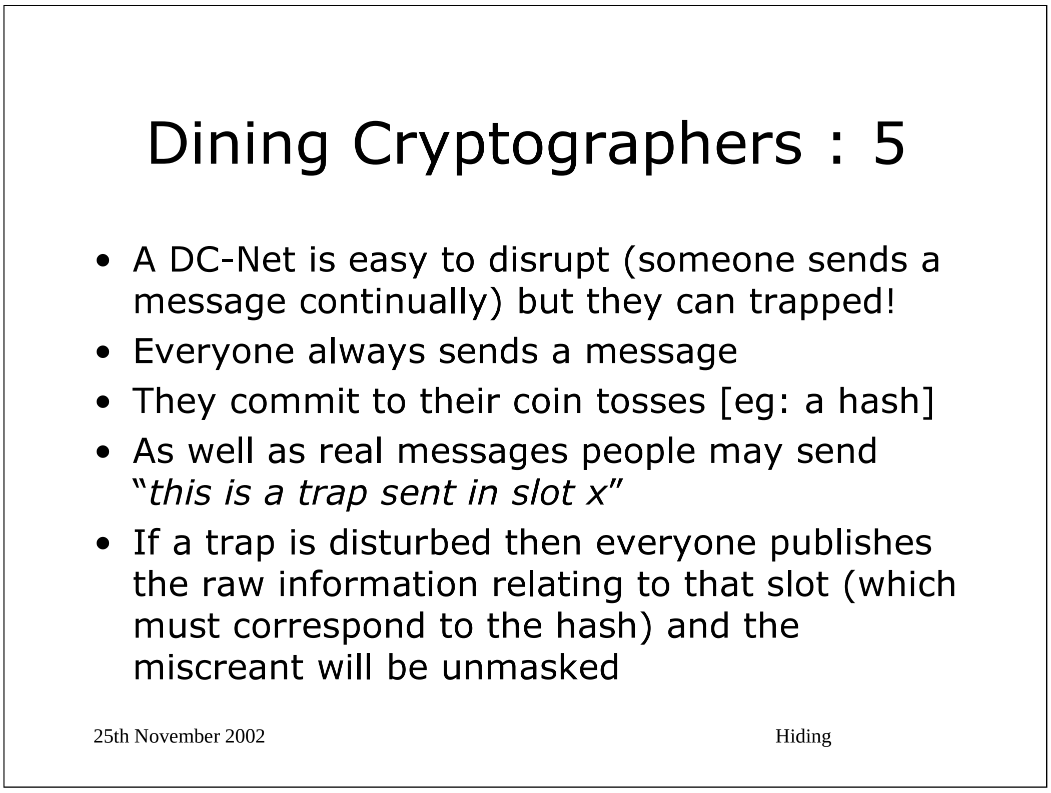
✶ It's important to note that there are two sorts of communication inherent in a DC-Net. Firstly, the keys have to be shared with neighbours. That can be done once, at the "beginning" (or after any rearrangement of which nodes are active) and is a series of **1➔1** communications. Secondly, all the participants must learn the overall state of the network for each bit of data. If they are all local to each other then a broadcast will work well. Otherwise one has to consider message passing from node to node. A simple scheme for that (but by no means the only such scheme) is to use a ring.

✶ As set out on the slide, it takes two round trips for everyone to learn the network state. Each participant will combine their random bit value with their neighbour's value. They will then invert the bit if they wish to transmit a "1".

They then XOR their value into the incoming data value and send it on to their neighbour. When the data value has been XORd into by all participants then it will contain the transmitted value (if any). This result must then be transmitted back around the ring so that every participant can learn what the overall result was.

✶ Clearly, on real networks it will make sense to handle multiple bits in parallel (perhaps 576 bytes worth or more).

✶ Since only one station can transmit at once, it makes sense to run a two phase protocol. In the first phase stations try to reserve the channel and in the second phase they use this. Since this makes clashes more likely, it makes sense to split the channel into multiple channels and allow stations to pick a random channel and then reserve and use that.



Dining Cryptographers : 5

- A DC-Net is easy to disrupt (someone sends a message continually) but they can trapped!
- Everyone always sends a message
- They commit to their coin tosses [eg: a hash]
- As well as real messages people may send "*this is a trap sent in slot x*"
- If a trap is disturbed then everyone publishes the raw information relating to that slot (which must correspond to the hash) and the miscreant will be unmasked

25th November 2002 Hiding

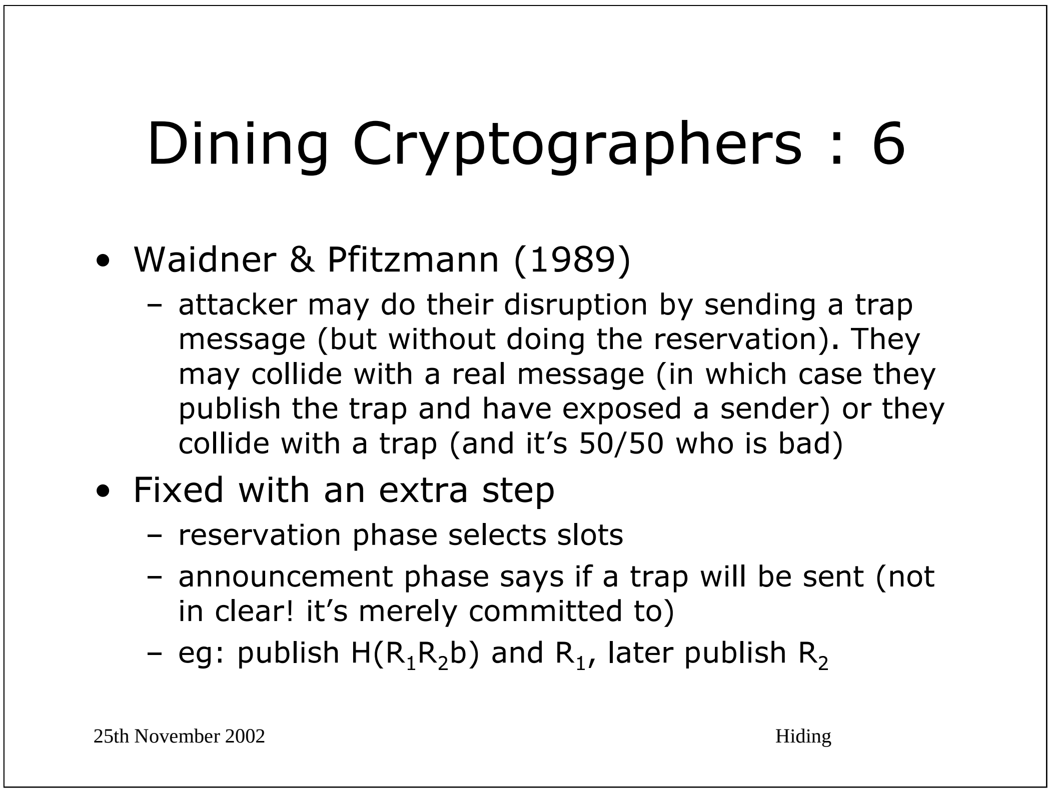
✶ Remember that the point of a DC-Net is that the sender is anonymous. If a wicked person starts sending a message at the same time as someone honest then the honest person (a) cannot tell who it is and (b) can hardly shout immediately that bad things are happening.

✶ However, if there are real messages and dummy messages in the system and they are indistinguishable then there is a real chance of the disrupter hitting a dummy message. If that happens then the disrupter can be exposed without anyone compromising the fact they were sending a real message.

[We're assuming a slot based system here, with a broadcast primitive. Everyone chooses a slot to send in at random – e.g.: n messages in n^2 slots – and then privately chooses whether to send a real message or a trap.]

✶ The commitment can work many ways, but a simple one is to work out the sequence of coin tosses that will be sent and compute and publish a cryptographic hash (SHA-1, MD5 &c) of that sequence.

✶ When a trap is disturbed everyone publishes their raw data (which they cannot lie about because of the pre-existing hash). The person who sent a message when they should not have done can be detected because they will be seen not to have followed the rules.



Dining Cryptographers : 6

- Waidner & Pfitzmann (1989)
  - attacker may do their disruption by sending a trap message (but without doing the reservation). They may collide with a real message (in which case they publish the trap and have exposed a sender) or they collide with a trap (and it's 50/50 who is bad)
- Fixed with an extra step
  - reservation phase selects slots
  - announcement phase says if a trap will be sent (not in clear! it's merely committed to)
  - eg: publish $H(R_1R_2b)$ and R_1, later publish R_2

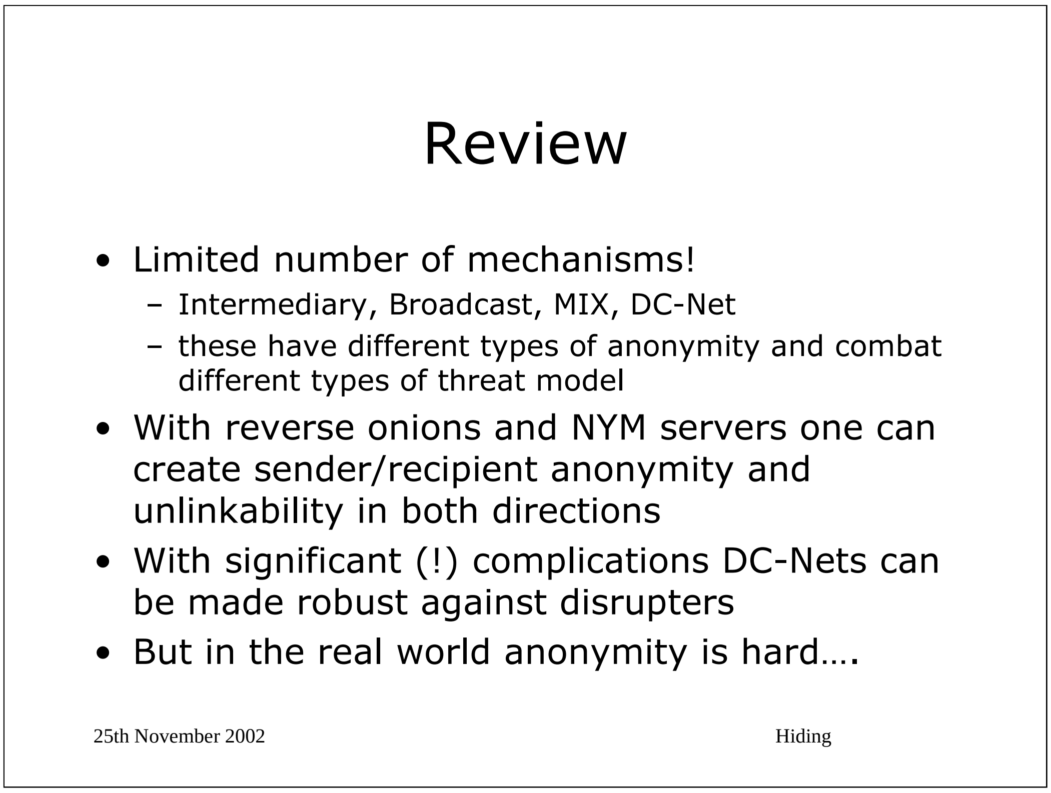
25th November 2002 Hiding

✶ Another paper that starts with a nice story – but then gets quite complex

"The Dining Cryptographers in the Disco: Unconditional sender and recipient untraceability with computationally secure serviceability", Michael Waidner and Birgit Pfitzmann. in Eurocrypt '89, LNCS 434, 1990

✶ Waidner & Pfitzmann found an attack on Chaum's original trap scheme which they then corrected with an extra step in which participants indicate whether or not they will be sending a trap.

✶ The paper is concerned with a number of other quite complex issues. For example, they're interested in ensuring that the correct type of commitment is used to ensure that attackers cannot gain information from their attacks and also in proving various guarantees even when broadcasts are no longer assumed to be reliable.



Review

- Limited number of mechanisms!
  - Intermediary, Broadcast, MIX, DC-Net
  - these have different types of anonymity and combat different types of threat model
- With reverse onions and NYM servers one can create sender/recipient anonymity and unlinkability in both directions
- With significant (!) complications DC-Nets can be made robust against disrupters
- But in the real world anonymity is hard….

25th November 2002 Hiding

✶ You can create real anonymity today – which can be of real use

anonymous helplines for victims/sufferers
whistleblowers, police informants
feedback to lecturers
refereeing of conference/journal papers
privacy – hiding from marketeers
privacy – hiding from your boss (or future boss)
privacy – hiding from your mum (spouse, or the Chief Whip)
social and political movements
criminals!

BUT

✶ The literature is full of real-world attacks on anonymity systems…

… for example, if a system is creating an onion to send through a MIX system then it might be interesting to inspect its DNS traffic and see which addresses it is looking up. This may yield B, M1, M2 and M3 directly!

✶ When you're assessing a system for its anonymity properties you have to look at the whole system – not just the specialist mechanism(s) provided by the academics.