Advanced Web Security

Anonymity

Anonymity on Internet

- Some degree of anonymity from using pseudonyms
- However, anonymity is always limited by IP address
  - TCP will reveal your IP address
  - IP address together with ISP cooperation → Anonymity is broken
  - We do not assume that the ISP will keep our identity secret
    - In any case, we are not anonymous to the ISP
- Assumption: If we know IP address, we know identity

Usage of Anonymity

Pros

- Discussion of sensitive and personal issues
- Information searches
- Freedom of speech in intolerant environments
- Polls/Surveys
- Counceling for drug or alcohol abuse
- Criminal victims
- Snitches and rats

Cons

- Accusations
- Propaganda
- Spreading of copyright protected material
- Other illegal acts

We just focus on the technology

Anonymous communication properties

- **Sender anonymity** – The identity of the party sending the message is hidden
- **Receiver anonymity** – The identity of the (actual) receiver of the message is hidden
- **Unlinkability of sender and receiver** – Two parties can not be identified as communicating with each other
- **Sender activity** – the fact that a sender is sending something
- **Receiver activity** – the fact that the receiver is receiving something
- **Sender content** – the sender sent a particular content
- **Receiver content** – the receiver received a particular content

Traffic analysis problem

- The problem of keeping confidential who is talking to whom and when they talk
- Can also be used to identify which type of data is sent even if data is encrypted

- Related problem: traffic confirmation problem – the problem of deciding if two given peers are communicating at a certain time
  - Very difficult to protect against

Anonymity

- You can only be anonymous within a set of other users
- More people using the system → more anonymity is possible
- Anonymity set – The set of people in which you are anonymous
  - Large set can provide higher anonymity than small

Anonymity Solutions

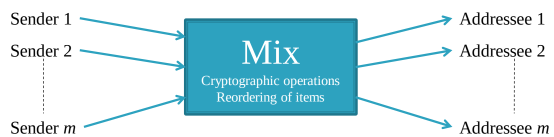
- Two main categories
  - High-latency designs → There is a significant delay between sending and receiving a message
    - Mixmaster
    - Babel
    - Mixminion
  - Low-latency designs → Aims at minimizing the time between sending and receiving a message
    - Tor
    - Java anon proxy
    - Freedom

The Global Passive Adversary (GPA)

- An adversary that can observe every node in the network
- A very strong adversary
- Typically
  - Low-latency designs cannot protect against the GPA
  - High-latency designs aims at protecting against the GPA

Anonymity

- First anonymity system:
  - D. Chaum, 1981
- Allowed *anonymous encrypted* email (hiding the address of the sender)
- Uses *public key cryptography*
- Based on a *Mix*



Assumptions

- Security assumption
  - It is not possible to find the correspondence between plaintext and ciphertext
- Power of adversary
  - Anyone can learn origin and destination of any and all messages in the underlying communication system
  - Anyone may inject, modify or remove messages
- Goals:
  1. The receiver should not be able to identify the sender
  2. An observer should not be able to link sender and receiver
  3. An eavesdropper should not be able to read the communication

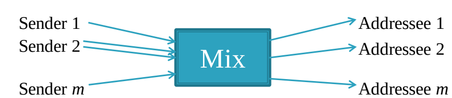
The Mix

- Purpose: Hide the correspondence between input and output items
- Computer that process each item
  - Can perform cryptographic operations
  - Reorders incoming items
- High latency – many inputs must be collected before output is produced to avoid timing attacks
- Make sure no item is processed more than once
  - Otherwise sender can be identified

Duplicating an Item

- If one item is duplicated at both input and output we know the correspondence between that sender and addressee

Sender 2 sent message to Addressee 1!



Duplicate message deleted by Mix

Sending a Message

- Prepare a message M to send to A:
  1. Add randomness R_0 to message in order to avoid simple guessing attacks
  2. Encrypt message using A's public key
  3. Add address of receiver together with randomness R_1
  4. Encrypt with public key of Mix
- Process data in Mix
  1. Decrypt data using private key of Mix
  2. Remove R_1
  3. Identify receiver A and forward encrypted message to A
- Read message at Receiver
  1. Decrypt message with receiver's private key

$$K_1(R_1, K_a(R_0, M), A) \longrightarrow \boxed{\text{Mix}} \longrightarrow K_a(R_0, M), A$$

Return Address

- Allow addressee to return message to sender without revealing return address
- Include untraceable return address

$$K_1(R_1, A_x), K_x$$

A_x – Return address used by addressee
K_x - Temporary public key used by addressee
R_1 - Random string also used as key by Mix

- Message returned to sender and processed by Mix

$$A_x, R_1(K_x(R_0, M)) \longleftarrow \boxed{\text{Mix}} \longleftarrow K_1(R_1, A_x), K_x(R_0, M)$$

Using Several Mixes

- Use cascade of *n* mixes – only one needs to be honest
- Prepare message for *n* Mixes

$$K_n(R_n, K_{n-1}(R_{n-1}, \ldots, K_2(R_2, K_1(R_1, K_a(R_0, M), A_y))\cdots))$$

- Message exiting first Mix

$$K_{n-1}(R_{n-1}, \ldots, K_2(R_2, K_1(R_1, K_a(R_0, M), A_y))\cdots)$$

- Message exiting last Mix

$$K_a(R_0, M), A_y$$

Sender *x* $\longrightarrow \boxed{\text{Mix}} \longrightarrow \boxed{\text{Mix}} \longrightarrow \text{- - - - -} \longrightarrow \boxed{\text{Mix}} \longrightarrow$ Addressee *y*

First Mix K_n Last Mix K_1

Several Mixes with Return Address

- Untracable return address prepared as

$$K_1(R_1, K_2(R_2, \ldots, K_{n-1}(R_{n-1}, K_n(R_n, A_x))\cdots))$$

- Returned together with message

$$K_x(R_0, M)$$

- Output of first mix

$$K_2(R_2, \ldots, K_{n-1}(R_{n-1}, K_n(R_n, A_x))\cdots), R_1(K_x(R_0, M))$$

- Output of last Mix

$$A_x, R_n(R_{n-1}(R_{n-2}\ldots R_2(R_1(K_x(R_0, M)))\cdots))$$

Sender *x* $\longleftarrow \boxed{\text{Mix}} \longleftarrow \boxed{\text{Mix}} \longleftarrow \text{- - - - -} \longleftarrow \boxed{\text{Mix}} \longleftarrow$ Addressee *y*

Last Mix K_n First Mix K_1

Attacks on Mixes

- Message size
  - Problem: If the size of the messages are different then it is easy to correlate inputs and outputs
  - Solution: Pad all packets to the same size
- Replay attack
  - Problem: Replaying an old message will send it to the same place as before
  - Solution: Save a fingerprint of ALL messages, or use time stamps
- N–1 attack
  - Problem: If an adversary controls all messages going into the Mix except the target message, then it is easy to see where the target message is going
  - Solution: Make sure all packets come from different senders (Will at least make this attack more difficult)

The Long-Term Intersection Attack

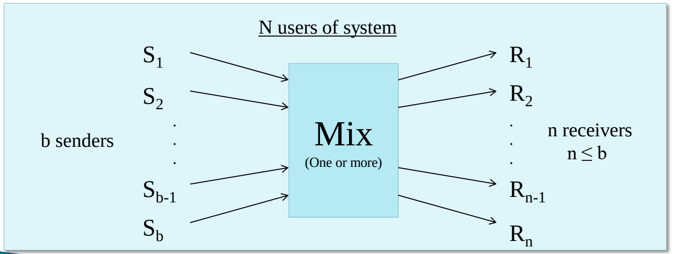
- Generic attack, works even when Mixes are perfect
- Assumption
  - Adversary can observe all messages entering and leaving the mix network
  - Adversary has good chance of guessing when a message entering the network is likely to leave
- Assume messages can be linked to same sender
  - Mailing list or forum pseudonym
  - Consecutive messages in conversation
- Store a list of possible senders at time i, i+1,i+2,....
- Intersect lists → right sender will be left in list
- Problem: The anonymity set is different in each time instance

Possible Countermeasures

- Remove linkability
  - Wait for later batches
  - Sometimes difficult or impossible
- Send dummy traffic from as many users as possible
  - Will increase the anonymity sets
  - Not possible with offline users

Disclosure Attack

- Kedogan, Agrawal and Penz, 2002
- Generic attack, similar to the intersection attack
- Assume all senders are different
- Alice has *m* communication partners, reveal these!



Disclosure Attack

Learning phase

- Attacker records all receivers used when Alice is sender at time t in set R_t
- This is done until we have m mutually disjoint sets $R_1,\ldots R_m$

$$\forall i, i \neq j, R_i \cap R_j = \emptyset$$

- *Now we know that each set includes a communication partner of Alice*

Disclosure Attack

Excluding phase

- Take new set R
  - If
  $$R \cap R_i \neq \emptyset \text{ and } R \cap R_j = \emptyset \quad \forall j \neq i$$
  - Then replace R_i by $R_i \cap R$
- Decrease set size until all m sets contain only one recipient → these m recipients are Alice's communication partners
- Conclusion: A Mix is insecure if $m \leq \lfloor N/n \rfloor$

General Problem

- High-latency design might be ok for email
- For interactive use such as HTTP, low latency is required

Onion Routing

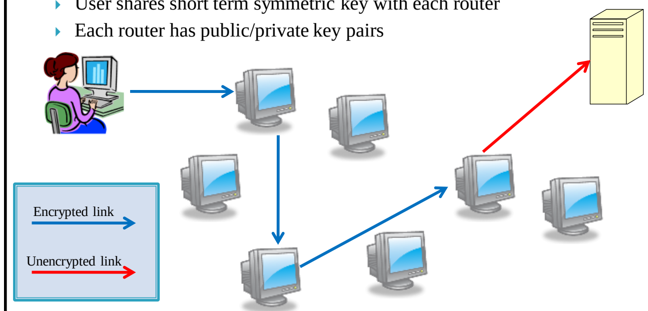
- Basically a set of real-time Chaum Mixes
  - One Mix is not enough in real-time since timing attacks will be ”easy”
  - Anonymity set is necessarily small
  - On the other hand, real-time allows for e.g., HTTP based traffic, voice over IP, etc
- Improving protection
  - Many Mixes
  - Large volumes of traffic
  - Synthetic traffic
  - All packets arriving within some small fixed time interval is mixed

Tor – The Onion Router

- Dingledine, Mathewson and Syverson, 2004
- Originally sponsored by US Naval Research Laboratory
- Now funded by Tor Project, a non-profit organization
- Low latency – appropriate for real-time traffic like HTTP
- Because of the low latency requirement we do not want each ”Mix” to decrypt the message using a private key
  - Symmetric cryptography is much faster

Tor Network

- User connects to remote server through several onion routers
- Each (blue) link is encrypted with TLS
- User shares short term symmetric key with each router
- Each router has public/private key pairs



Data Transmitted in Cells

- Each cell is 512 bytes
- Two types of cells
  - Control cells – always interpreted by receiving node
    - Padding
    - Create, Created
    - Destroy
  - Relay cells – Interpreted by last cell, relayed by other cells
    - Relay begin
    - Relay data
    - Relay extend
    - ...

Control Cells and Relay Cells

Bytes	
2	CircID
1	CMD
509	DATA

Control cell

Bytes	
2	CircID
1	CMD
1	Relay CMD
2	Recognized
2	StreamID
4	Digest
2	Length
498	DATA

Relay cell

- CircID – Hold a 2-byte ID for the connection to the next router
- CMD – command to determine what to do
- Recognized – used to know if packet should be relayed or interpreted
- StreamID – ID for this stream
- Digest – Integrity check
- Length – bytes in DATA that is real data

Negotiating Symmetric Key with First Node

- Diffie-Hellman key exchange

Alice — OR$_1$

$$\text{ID}_1, \text{Create}, K_{OR_1}(g^{x_1}) \mod p \longrightarrow$$

$$K_1 = (g^{x_1})^{y_1} \mod p$$

$$\longleftarrow \text{ID}_1, \text{Created}, g^{y_1} \mod p, H(K_1)$$

$$K_1 = (g^{y_1})^{x_1} \mod p$$

- Unilateral entity authentication – Alice knows she is handshaking with OR$_1$, but OR$_1$ does not know who is opening the circuit.
- Unilateral key authentication – Alice knows that only OR$_1$ knows the shared key K$_1$.

K() – Public key encryption (asymm)
K{} – Secret key encryption (symm)

Negotiating Symmetric Key with Second Node

- Contact second node through first node

Alice — OR$_1$ — OR$_2$

$$\text{ID}_1, \text{Relay}, K_1\{\text{Extend}, OR_2, K_{OR_2}(g^{x_2})\} \longrightarrow$$

$$\text{ID}_2, \text{Create}, K_{OR_2}(g^{x_2}) \longrightarrow$$

$$\longleftarrow \text{ID}_2, \text{Created}, g^{y_2}, H(K_2)$$

$$\longleftarrow \text{ID}_1, \text{Relay}, K_1\{\text{Extended}, g^{y_2}, H(K_2)\}$$

- OR$_1$ keeps track of mapping between ID$_1$ and ID$_2$
- Alice do not need any info about ID$_2$

K() – Public key encryption (asymm)
K{} – Secret key encryption (symm)

Constructing Circuit

- OR only knows about node before and after.
  - ID is used to identify where to send packets
  - Table is kept to identify where incoming packets are going
- User changes to new circuit periodically (about once per 10 minutes)
- Setup requires public key operations
  - Takes time
- Users construct circuits preemptively

Relaying Data Through Nodes

Alice — OR$_1$ — OR$_2$ — Website

$$\text{ID}_1, \text{Relay}, K_1\{K_2\{\text{Begin}, website\}\} \longrightarrow$$

$$\text{ID}_2, \text{Relay}, K_2\{\text{Begin}, website\} \longrightarrow$$

$$\longleftrightarrow \textit{TCP Handshake}$$

$$\longleftarrow \text{ID}_2, \text{Relay}, K_2\{Connected\}$$

$$\longleftarrow \text{ID}_1, \text{Relay}, K_1\{K_2\{Connected\}\}$$

$$\text{ID}_1, \text{Relay}, K_1\{K_2\{Data, \text{"..."}\}\} \longrightarrow$$

$$\text{ID}_2, \text{Relay}, K_2\{Data, \text{"..."}\} \longrightarrow$$

$$\text{"..."} \longrightarrow$$

128 bit AES in counter mode is used

K() – Public key encryption (asymm)
K{} – Secret key encryption (symm)

Determine when to Relay and when to Interpret

2	CircID
1	CMD
1	Relay CMD
2	**Recognized**
2	StreamID
4	**Digest**
2	Length
498	DATA

Relay cell

- Control cells are always interpreted
- Relay cells are interpreted by exit node
- Recognized field set to zero and digest calculated with SHA-1 by initiator before encrypted with $K_n,\ldots,K_2,K_1$
- After decryption at OR_i these 6 bytes are checked. If OK, then interpret, otherwise relay
- Probablity of failure: 2^{-48}
- Leaky pipe circuit – Allow exit at different ORs in same circuit

Tor Security

- Tor does not claim to protect against
  - A Global Passive Adversary – someone who can observe all network links
  - Traffic confirmation attacks
- Almost no mixing because of real time
- Is the Global Passive Adversary realistic?
  - Maybe not, but an attacker does not really have to observe everything.
  - A real world observer can instead be adaptive – select where to observe based on prior information
- Users running their own router has higher anonymity

Perfect Forward Secrecy

- If private key of a router is compromised, the session key can still not be derived
  - Property of Diffie-Hellman Key exchange

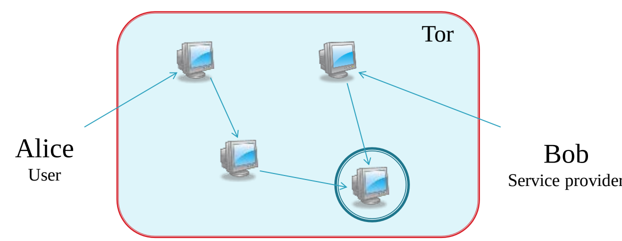
$$\xrightarrow{\mathrm{ID}_1, \mathrm{Create}, \boxed{K_{OR_1}}(g^{x_1}) \mod p}$$

$$\xleftarrow{\mathrm{ID}_1, \mathrm{Created}, g^{y_1} \mod p, H(K_1)}$$

- Implication: If encrypted traffic is recorded, and session keys are destroyed, it can never be decrypted even if long-term keys are stolen
- Compare to the case when the client e.g., generates a session key and encrypts it with router's public key → no perfect forward secrecy

Hidden Services

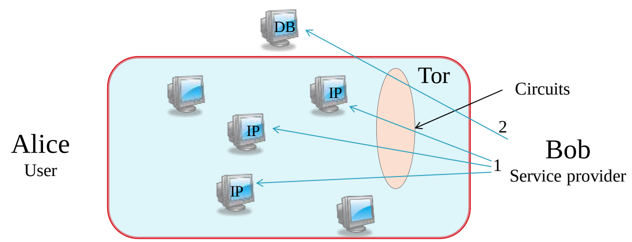
- Anonymous services (hidden services), e.g., anonymous webserver
- How can we provide a service through a web server without revealing our IP?



Using Rendezvous Points

Bob initializes service

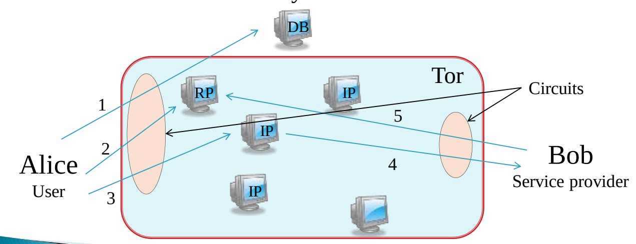
- Bobs service identified by his public key
- Bobs chooses a set of introduction points (IP)
- Bob builds circuit to each introduction point
  - ”Public key” can be found at IP_A, IP_B, IP_C, ... is written in database



Using Rendezvous Points

Alice wants to use Bobs service

- Alice finds IPs to Bob in database
- Alice chooses OR as rendezvous point (RP) and connects to it
- Alice connects to one of Bobs IPs and tells it about her RP
- Introduction point forwards message to Bob and Bob can connect to the RP chosen by Alice



Difference Between Anonymity and Encryption

- Tor does not provide end-to-end encryption
- Owner of exit node decrypts traffic and forwards it in plain text
- Identifying information in data stream can still reveal your identity
  - If you log in to an email account
  - If you pay with your credit card
  - If you send instant messages
- Sept 2007, 100 email accounts from embassies and government agencies posted on internet
  - Tor exit node was used to capture the info
  - Probable chain of events: Stolen accounts → Use Tor to hide who you are → End node sees the traffic