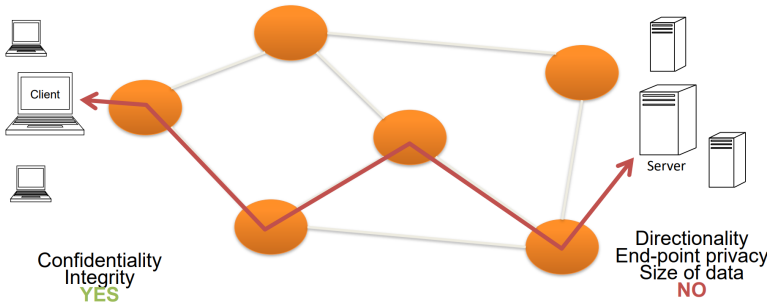
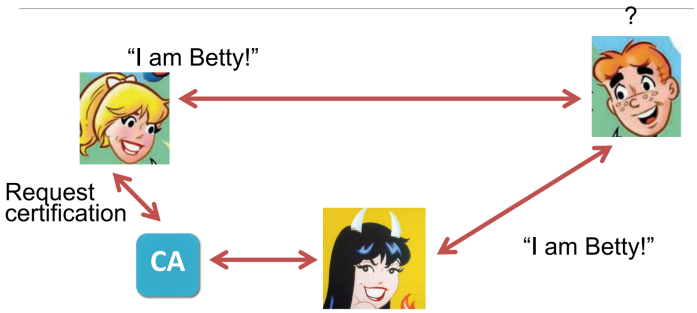
CS Revision Lecture 14, 15, 16

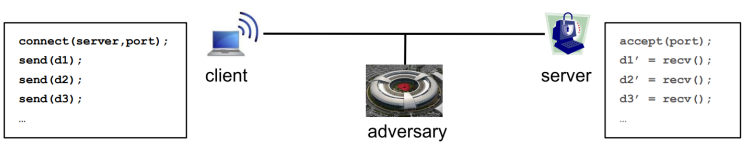
- ## Lecture 14 - SSL/TLS
  - ### Objective: Point -to-point secure channel



  - ### Authentication Problem

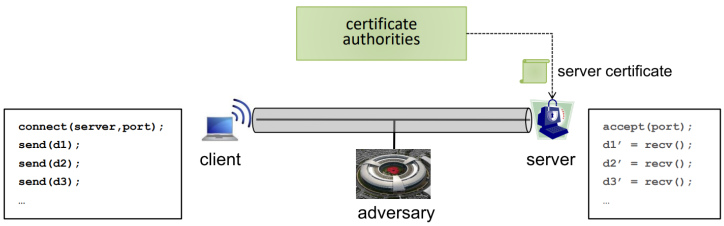


    - Who is the real Betty
      - Could recognise by certificate authority
  - ### SSL/TLS
    - Secure Sockets Layer: Developed by Netscape.
    - SSL Version 3 released in 1996.
    - Substituted by TLS 1.0 in 1999: (Transport Layer Security). Standardized by IETF. Published as RFC 2246
    - TLS 1.3. Published as RFC 8445 in 2018.
    - On top of TCP/IP, below application protocols
  - ### Secure channels for the Web
    - Threat model: Adversary fully controls the network, e.g.



- It can redirect traffic to its own server (DNS rebinding)
- It can passively read all data (IP monitoring)
- It can play an active man-in-the-middle (TCP hijacking)

- Security Goal:



- As long as the client is honest and the adversary does not know the server's private key, it cannot
  - Inject forged data into the data stream (integrity)
  - Distinguish the data stream from random bytes (confidentiality)

- Goals of SSL/TLS
  - **End-to-End Confidentiality**
    - Encrypt communication between client and server applications
  - **End-to-End Integrity**
    - Detect corruption of communication between client and server applications
  - **Required server authentication**
    - Identity of server always proved to client
  - **Optional client authentication**
    - Identity of client optionally proved to server
  - **Modular deployment**
    - Intermediate layer between application and transport layers
    - Handles encryption, integrity, and authentication on behalf of client and server applications

- Certificates
  - Public key certificate
    - Assurance by a third party that a public key is associated with an identity
    - E.g. QuoVadis certifies that the public key below is associated with University of Edinburgh web server

```
d6 23 7e f5 e7 56 2c e3 50 d7 e1 4e 98 f4 cc 97 61 b2 ae 07 b8 b8 3d 6e
02 f4 9b c1 32 e5 56 bb 78 ea c0 2e 62 84 33 27 e4 2a 83 64 9f 53 cf e7
04 92 3e 4b 6d f8 55 68 a3 40 21 ff 70 66 a6 a4 50 a8 d9 87 54 97 fe ee
5a a4 b7 99 22 57 d2 df 84 35 5b 26 8c 09 2d 98 a9 74 0f e0 d9 d3 97 1d
fd 80 8f 9c 5a e8 cc 78 0f 7b f7 95 2f 4f b4 07 cc 05 6d d3 0d 9a a4 37 fb
ef 0d a8 b9 00 6f 4b d6 3f 80 04 38 09 9a e8 6e 27 aa a4 f1 20 7e 13 57
f4 9b ca cb 7f 3c 93 4d 1f e6 55 a1 4e 7c e2 06 a5 4b 8e 20 25 5d 07 e8
98 68 1a ea 0b cc fd 53 0a 66 93 be 31 b9 75 bb aa 04 b4 8f ac 56 8b 05
4d e1 68 2e 65 04 b6 da 93 49 60 2c c7 d2 74 fa 34 da 70 8c 7d bb f2 e6
b6 df 2a 8e 48 8f 15 ae 2f a0 ad 86 07 9c 9d c9 c0 1d 8b 06 b8 b9 ba
```

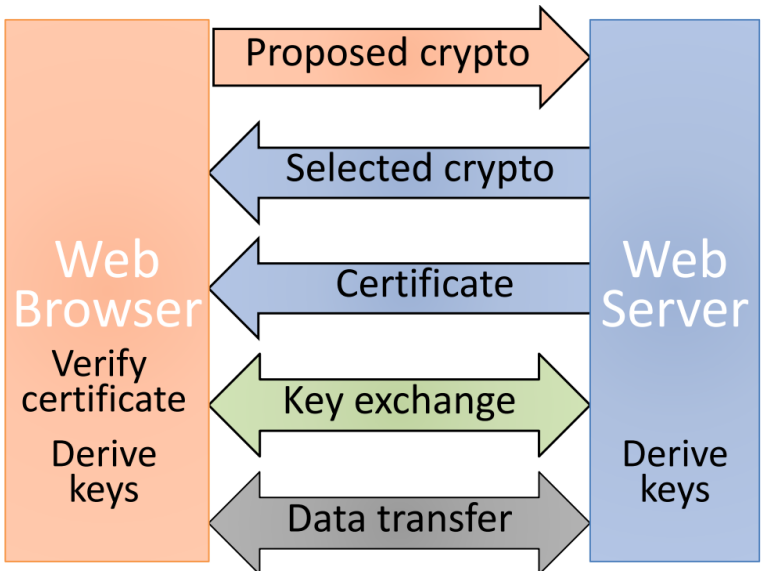
  - Certificate fields
    - **Issuer aka certificate authority** (Ex: QuoVadis)
    - **Subject** (E.g. University of Edinburgh; www.ed.ac.uk)
    - **Subject's public key parameters** (E.g. RSA2048)
    - **Subject's public key**
    - **Validity period** (E.g. 29/11/16 - 29/11/19)
    - **Signature parameters** (SHA256; PKCS #1, RSA2048)
    - **Signature**
- Chain of Trust and Revocation
  - Transitive trust
    - Trust of (public key of) issuer implies trust of (public key of) subject
    - Issuer can be subject in another certificate
    - Chain of certificates
    - Root of trust?
    - Root certificates preconfigured in operating system and browser
  - Certificate revocation
    - Mechanism to invalidate a previously issued certificate

- E.g., when private key of the subject is compromised
- **Revocation methods**
  - List of revoked certificates posted on CA's website
  - Online verification service provided by CA (OCSP stapling)

- Rogue Certificates: DigiNotar hacked in 2011
  - Google noted a DigiNotar issued certificate for google.com that wasn't on Google's own list of Google certificates
  - Used to impersonate Google mail web site and collect usernames and passwords
  - Serious impact on Dutch government IT services
  - DigiNotar went bankrupt
- TLS Building Blocks

	Confidentiality	Integrity	Authentication
Setup	Public-key based key-exchange (RSA and DH)	Public-key digital signature (e.g., RSA)	Public-key digital signature (e.g., RSA)
Data transmission	Symmetric encryption (e.g., AES in CBC mode)	Hash-based MACs (e.g., HMAC using SHA256)	

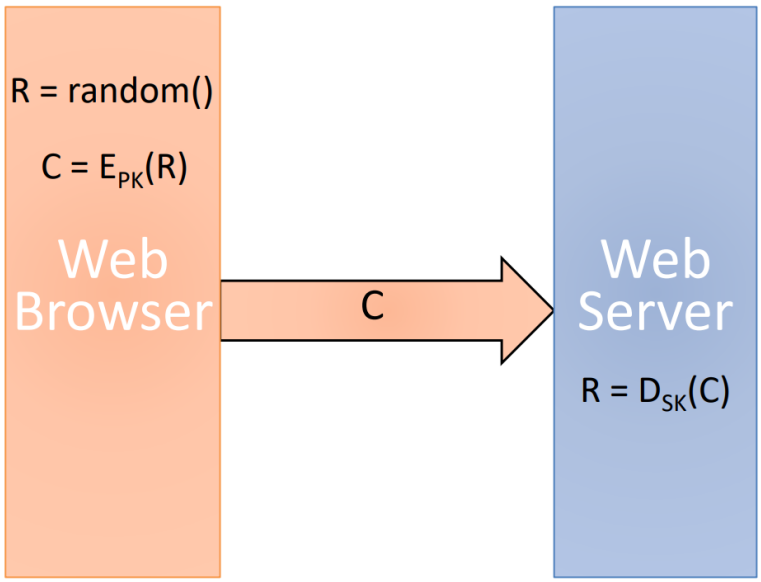
- TLS Overview(Simplified)



  - Browser sends supported crypto algorithms

- Server picks strongest algorithms it supports
- Server sends certificate (chain)
- Client verifies certificate (chain)
- Client and server agree on secret value R by exchanging messages
- Secret value R is used to derive keys for symmetric encryption and hash-based authentication of subsequent data transfer

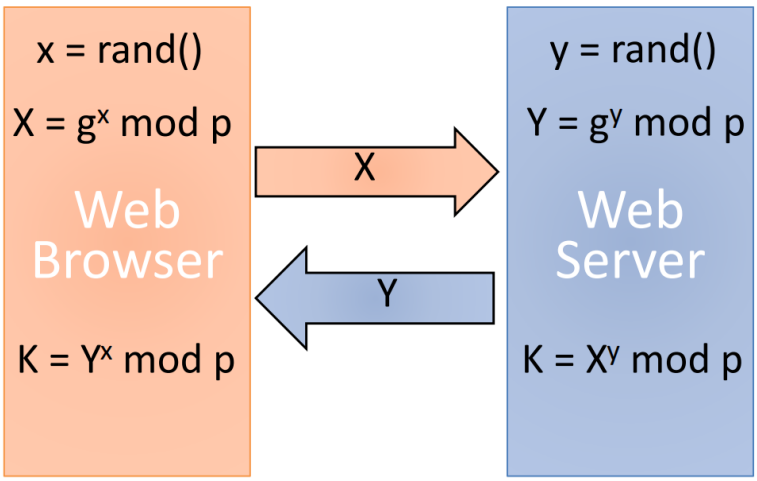
- Basic Key Exchange



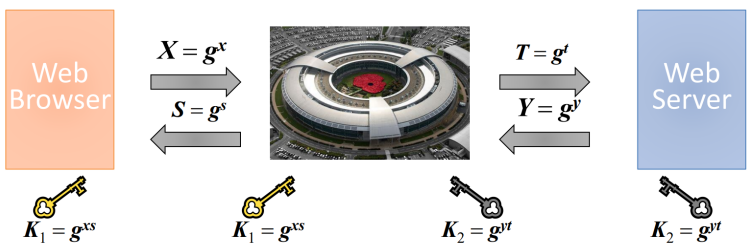
- Called RSA key exchange for historical reasons
- Client generates random secret value R
- Client encrypts R with public key, pk, of server $C = E_{pk}(R)$
- Client sends C to server
- Server decrypts C with private key, sk, of server $R = D_{sk}(C)$

- Forward Secrecy
  - Compromise of public-key pairs private keys does not break confidentiality of past messages

- TLS with basic key exchange does not provide forward secrecy
  - Attacker eavesdrop and stores communication
  - If server's private key is compromised, attacker finds secret value R in key exchange and derives encryption keys
- Achieves forward secrecy: Diffie Hellman Key Exchange



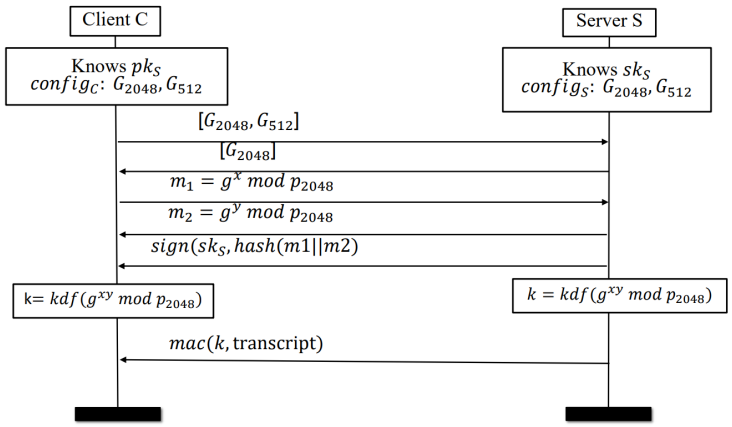
  - Public parameters: prime p and generator g of Z_p
  - Client generates random x and computes $X = g^x mod\ p$
  - Server generates random y and computes $Y = g^y mod\ p$
  - Client sends X to server
  - Server sends Y to client
  - Client and server compute $K = g^{XY} mod\ p$
- Attacker in the Middle



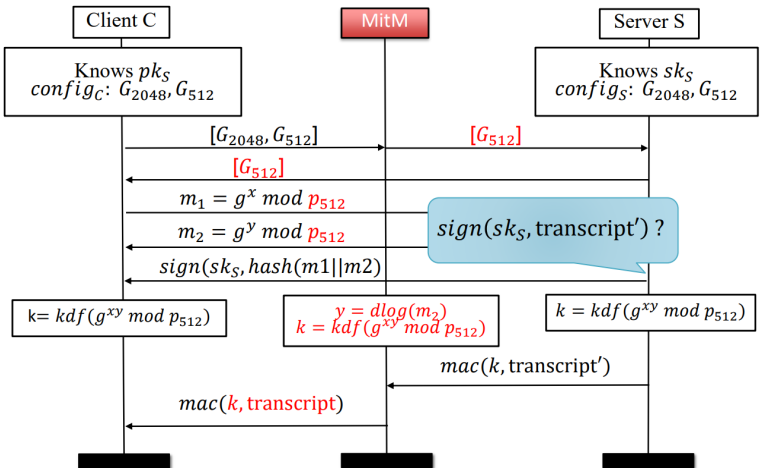
  - Solution

- Browser and server send signed $X and\ Y$
- Requires each to know the public key of the other
- One-sided authentication if only server signs

- Signed DH key exchange



- LOGJAM

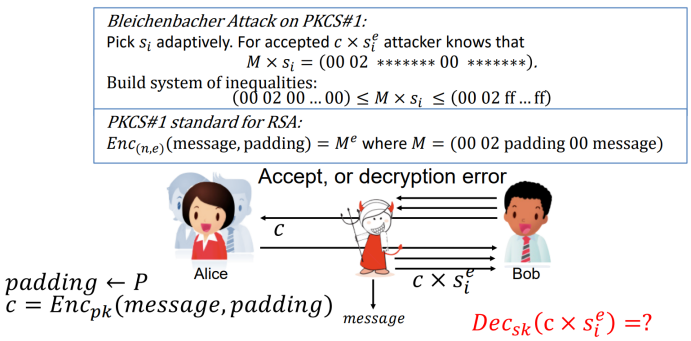


- Eve downgrade the connection
- Eve now could calculate k when using 512 bit config (G_{512} is not safe)
- When server send the transcript to client. Eve could forge the transcript since he/she knows the key. Therefore, client will not know the connection has been modified

- Back to the roots $\sqrt[e]{c}$ [RSA78]

- There was RSA
  - $E(m) = m^e \ mod\ n$
  - n = $p \cdot q$ - product of primes

- RSA is homomorphic
  - $(m^e \cdot s^e)^d = (m \cdot s)^{ed} = m \cdot s$
- Padding Attacks [B98]

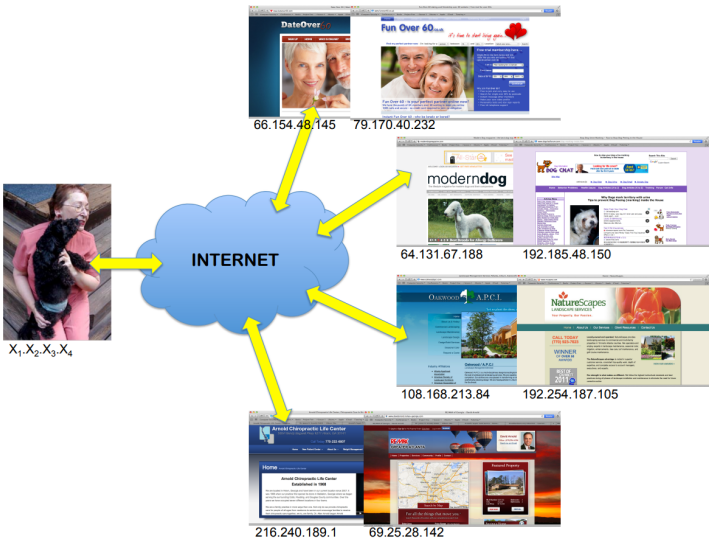


  - Message must be the format of (00 02 padding 00 message)
  - Adversary could pick the s_i. S.t. $M \times s_i = (00\ \ 02\ ******\ 00\ \ ******)$
  - If the server could send multiple times, adversary could find M by using different s_i
- What We Have Learned
  - Goals and history of the SSL/TLS protocol
  - TLS Certificates, chain of trust and revocation
  - Overview of the TLS protocol
  - Diffie-Hellman key exchange and forward secrecy
  - Logjam and Bleichenbacher attack

Lecture 15 - Anonymous communication 1

- Context
  - **The Internet is a public network**
    - Network routers see all traffic that passes through them
  - **Routing information is public**
    - IP packet headers contain source and destination of packets

- **Encryption does not hide identities**
  - encryption hides payload, but not routing information
- Routing information can reveal who you are



"With your permission, you give us more information about you, about your friends, and we can improve the quality of your searches. We don't need you to type at all. We know where you are. We know where you've been. We can more or less know what you're thinking about."

Eric Schmidt, CEO Google, 2010

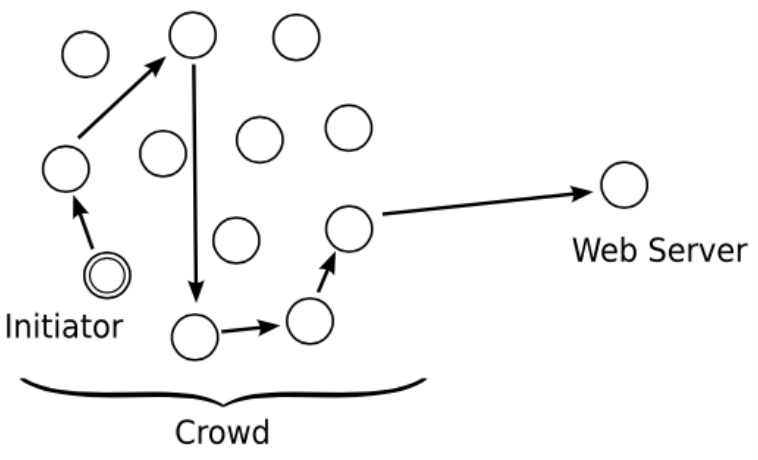
- Anonymity
  - Definition (ISO/IEC standard 15408)
    - A user may use a service or resource without disclosing the user's identity
  - $\rightarrow$ this can be achieved by hiding one's activities among others' similar activies
    - Dining cryptographers
    - Crowds

- Chaum's mix
- Onion routing

- Three-party dining cryptographers (3DC)
  - Three NSA cryptographers are having dinner. At the end of the dinner they are informed that the dinner has already been paid for. Now, either the NSA paid for the dinner, or one of the cryptographers did. They want to know if it is the NSA that paid, or one of them, without revealing the identity of the paying cryptographer.
  - 3DC protocol
    - **Phase 1**: Each pair of cryptographers flips a coin (that only they can see), where heads = 1 and tails = 0
      - each cryptographer will see two coin flips
    - **Phase 2**: Each participant publicly announces the result as follows:
      - **Did NOT pay**: the XOR of the two coin flips they observed
      - **Did pay**: the negation of the XOR of the two coin flips they observed
    - **Resolution**: If the XOR of the three announcements is
      - 0: The NSA paid
      - 1: One of them paid (but only the payer will know this.)
- Superposed sending
  - 3DC protocol generalises to any group size n > 2 (nDC)
  - Sender wants to anonymously broadcast a message m. For each bit bit, m_i of m:
    - 1. every pair of users generate a random bit (0, 1)
      - every user observes n-1 bits.
    - 2. each user (except the sender) announces (XOR of all $n-1$ observed bits)

- 3. the sender announces (XOR of all $n-1$ observed bits and m_i)
- 4. XOR of all announcements = m_i
  - every randomly generated bit occurs in this sum twice (and is cancelled by XOR)
  - m_i occurs only once

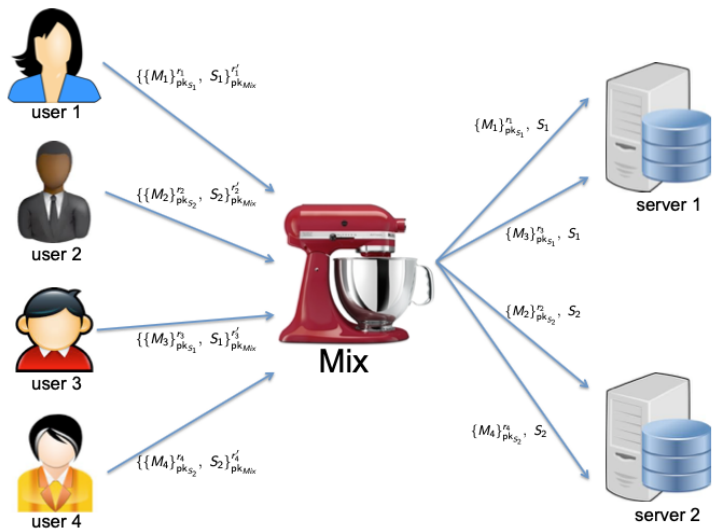
- Limitations of the DC protocol
  - The DC protocol is impractical:
    - Requires pair-wise shared secret keys (secure channels) between the participants (to share random bits)
    - Requires large amounts of randomness
    - Any one can launch a denial of service attack by either not performing the protocol properly, or by transmitting at the same time as some one else. Only one person can transmit at any given round.
- Crowds
  - Idea: randomly route the request through a crowd of users
  - a crowd is a group of m users: c out of m users may be corrupted
  - an initiator that wants to request a webpage creates a path between him and the server



- 1. the initiator selects a forwarder from the crowd and sends him his request
- 2. a forwarder delivers the request directly to the server with probability $1 - p_f$. he forwards the request to a randomly selected new forwarder from the crowd with probability p_i, the new forwarder repeats the procedure
- 3. the response from the server follows same route in opposite direction

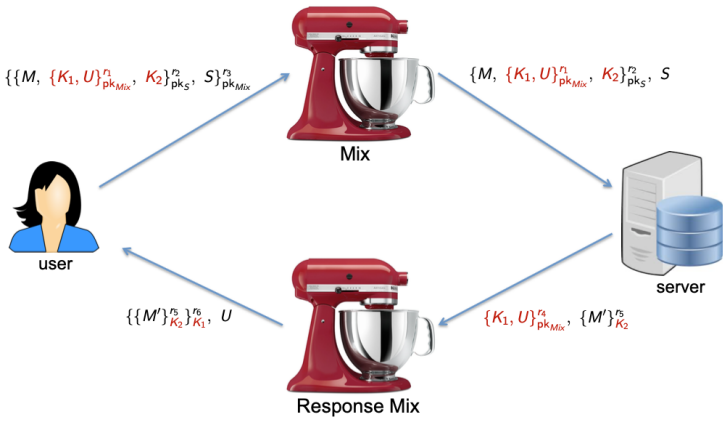
- **Crown IS NOT resistant against an attacker that sees the whole network traffic!** -- Global adversary

- Chaum's mix



- message padding and buffering to avoid time correlation attacks
- dummy messages are generated by the mixes themselves to prevent an attacker sending $n - 1$ messages to a mix with capacity n, allowing him to the link the sender of the n^{th} message with its recipient.

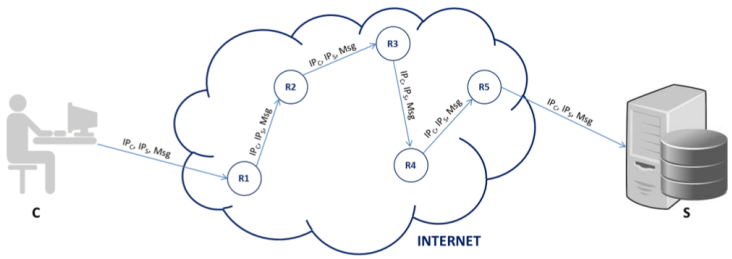
- Anonymous return addresses



- Mix cascade
  - Messages are sent through a sequence of mixes
  - some of the mixes may be corrupted
  - a single honest mix guarantees anonymity against an attacker
    - message padding
    - buffering
    - dummy messages
- Limitations of Chaum's mixnets
  - Asymmetric encryption is not efficient
  - Dummy messages are inefficient
  - Buffering is not efficient

Lecture 16 - Anonymous communication： onion routing and Tor

- Routing and privacy



  - Internet routing exposes user's privacy (meta-data like IPs)

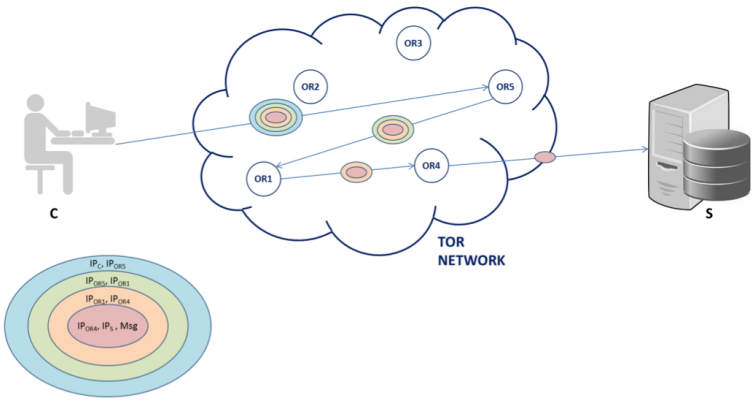
- All routers on the path between source and destination, know the origin and destination of forwarded packets
- Core internet routers are managed by governments and big corporations (so they can observe a large fraction of Internet activity)

- Today's lecture

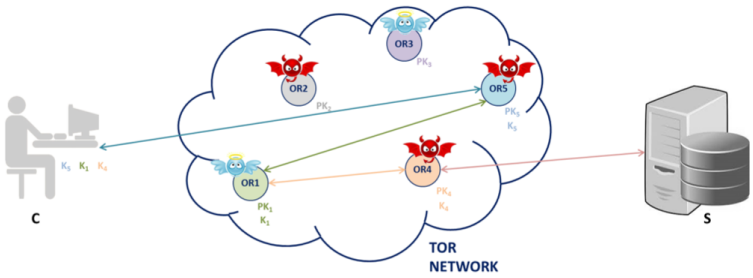
Stand up for privacy and freedom online

- Idea: Combine advantages of mixes and proxies
  - use public-key crypto only to establish circuit
  - use symmetric key crypto to exchange data
  - distribute trust like mixes
  - do not delay or batch like mixes (low-latency)
- **But does not defend against adversary that observes the whole network**

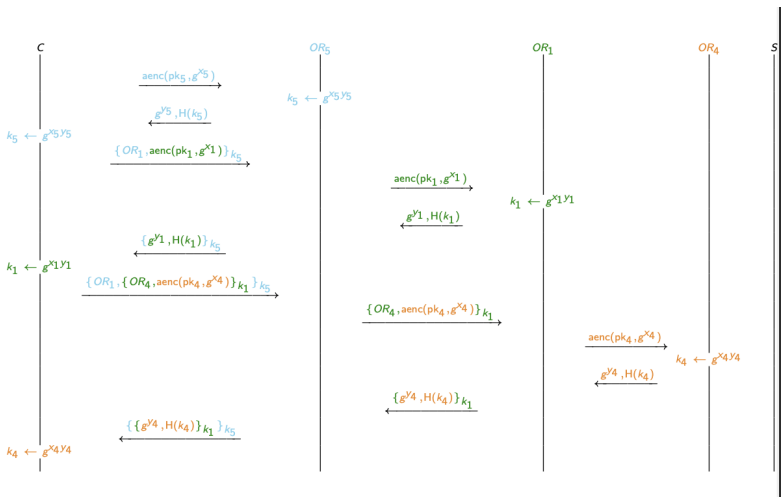
- Tor's main ingredient: the onion



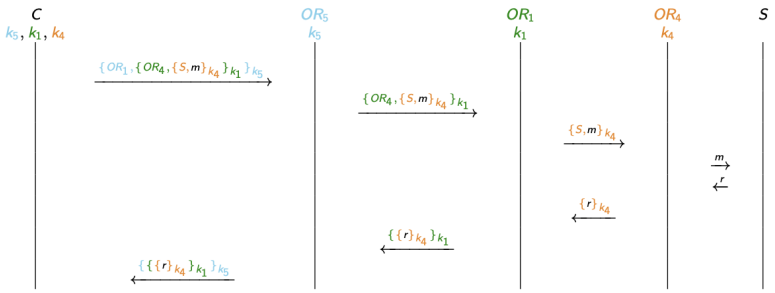
- Tor circuit setup



  - C establishes session key K_5 and circuit with Onion Router OR_5
  - C tunnels through that circuit to extend to Onion Router OR_1
  - C tunnels through that extended circuit to extend to Onion Router OR_4
  - Client applications connect and communicate of established Tor circuit
  - **A single honest Onion Router on the Tor circuit guarantees anonymity against an attacker controlling some Onion Routers**
- The (simplified) Tor message flow - circuit setup



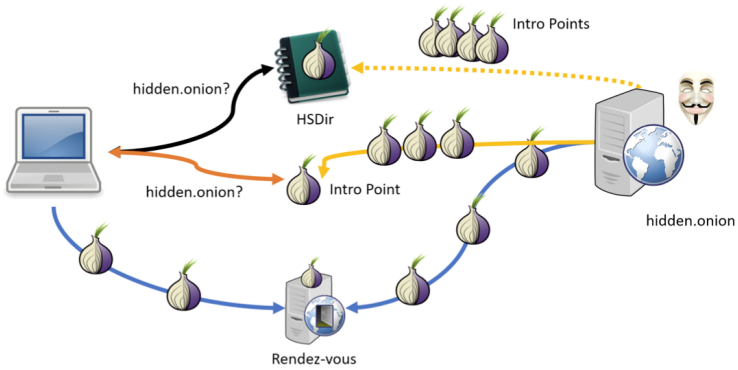
- The (simplified) Tor message flow - actual communication



- Tor only provides privacy - not confidentiality
  - Tor anonymises the origin of the traffic
  - Tor encrypts everything inside the Tor network
  - but Tor DOES NOT encrypt all traffic through the Internet
  - For confidentiality you still need to use end-to-end encryption such as SSL/TLS
- Tor take care of DNS resolution
  - Tor only anonymises TCP streams
  - But, DNS resolution is executed over UDP
  - So, DNS resolution if handled by the client browser defeats the purpose of using Tor
  - To avoid privacy breaches due to DNS resolution, the Tor browser delegates DNS resolution to the exit mode
- Avoiding censorship
  - Tor relays are listed on the public Tor directory
  - So your local ISP can observe that you are communicating with Tor nodes
  - ISPs and governments can try to block access to the Tor network by blocking Tor relays
  - Tor bridge relays are relays not listed on the public Tor directory
  - Entering the Tor network through a Tor bridge relay can prevent ISPs and governments blocking access to the Tor

network

- Limitations of Tor
  - Tor does not provide protection against end-to-end timing attacks
  - If the attacker can see both ends of the communication channel, he can correlate volume and time information on the two sides
- How do Onion Service work?



  - 1. hidden server register with HSDir with some Intro points. Later, the server could retrieve information from intro points
  - 2. The client types the hidden.onion url on the browser. The browser will check the HSDir and return the intro points to the client.
  - 3. The client access to the intro point, showing he/she interested to access to the onion service. Then let the hidden service knows that he/she is waiting at the Rendez-vous point.
  - 4. The hidden service connect to the Rendez-vous point via Tor connection.
- Conclusions
  - Presented a brief overview of several anonymity system
    - How they work

    - Their privacy guarantees
- Tor
    - How it works
    - Tradeoff between privacy and efficiency
- There is much more to anonymous communications
    - Tarzan, Bluemoon, etc.

以上内容整理于 幕布文档