E2EE & TOR

Applied Cryptography

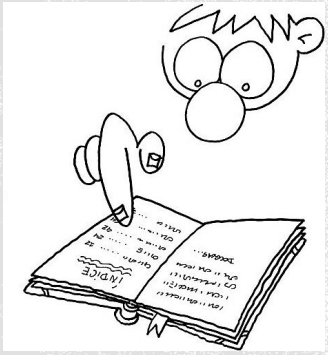
Silvio Ranise [silvio.ranise@unitn.it or *ranise@fbk.eu*]

SECURITY & TRUST

FONDAZIONE BRUNO KESSLER

CONTENTS

- End-to-End Encryption
  - PGP
    - Key management and Web of trust
- TOR
  - Tor and VPN



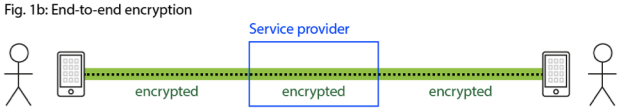
END-TO-END-ENCRYPTION (E2EE)

- The overall goal of Information Security is preserving the **integrity** and/or **confidentiality** of the data from its sending point to the receiving point in a network
- Information Security must also provide for the receiver to be certain that the sender is actually the entity that the information was received from (**authenticity**)
- In the context of networks, these properties can be achieved by using the notion of **end-to-end encryption**
- Notice that the notion of end-to-end encryption has evolved over time:
  - **Initially**: communication is never decrypted during its transport from sender to receiver
  - **After 2014**: not only the communication stays encrypted during transport but also that the provider of the communication service is not able to decrypt the communications either by having access to the private key or by having the capability to undetectably inject an adversarial public key as part of a man-in-the-middle attack

THE IMPORTANCE OF E2EE

Fig. 1a: Encryption in transit

Fig. 1b: End-to-end encryption



- In the modern usage of E2EE, typical server-based communications systems do not include end-to-end encryption
  - They can only guarantee the protection of communications between clients and servers
  - This implies that users have to trust the third parties who are running the servers with the sensitive content
- E2EE techniques in the modern sense are regarded as safer because they reduce the number of parties who might be able to interfere or break the encryption
- To understand the value of E2EE for real-world applications, an instructive reading can be found in the following blog-post:

  https://martin.kleppmann.com/2015/11/10/investigatory-powers-bill.html
  - The figure above is taken from the blog and clearly illustrates the advantages of E2EE over encryption in transit

DIGRESSION: RELEVANCE OF E2EE TO DEMOCRACY

- To understand why E2EE or better its weakening/suppression can have a negative impact on our fundamental rights and freedom, an instructive reading is the following article

  https://www.newyorker.com/tech/annals-of-technology/the-daunting-challenge-of-secure-e-mail

- Notice that recently, after the wave of terrorist attacks around the world, encryption in general and more precisely E2EE has been under pressures by governments all around the world, including the EU, as discussed for example in the following article

  https://eandt.theiet.org/content/articles/2020/11/eu-resolution-could-target-end-to-end-encryption/

SECURITY OF E2EE (1)

- It maybe still vulnerable to Man-in-the-Middle (MiTM) attacks
  - If the attackers can interfere with the key exchange protocol used
- To prevent MiTM attacks, end-point authentication is used in many E2EE solutions
  - For instance, using
    - **Certificates** (thus assuming the availability of a PKI)
    - **Web-of-trust** (we will see more on this later)
    - **Fingerprints**: short sequence of bytes used to identify a longer public key
      - Fingerprints are created by applying a cryptographic hash function to a public key
      - Since fingerprints are shorter than the keys they refer to, they can be used to simplify certain key management tasks
- Service providers may willingly or unwillingly introduce **backdoors** (= secret methods to bypass authentication and/or encryption) to their software that help subvert key negotiation or bypass encryption altogether
  - In 2013, information leaked by Edward Snowden showed that Skype had a backdoor which allowed Microsoft to hand over their users' messages to the NSA despite the fact that those messages were officially E2EE encrypted

SECURITY OF E2EE (2)

End-point security

- The E2EE paradigm does not directly address risks at the communication endpoints themselves
- Each user's computer can still be hacked to steal the owner's cryptographic key (to create a MITM attack) or simply read the recipients' decrypted messages both in real time and from log files
  - Even the most perfectly encrypted communication pipe is only as secure as the mailbox on the other end
- Major attempts to increase endpoint security have been to isolate key generation, storage and cryptographic operations to Hardware Security Modules (HSMs) such as smart cards
  - Notice that since plaintext input and output are still visible to the host system, malware can monitor conversations in real time

DESIRED PROPERTIES

- E2EE should help guarantee the following properties for applications that use the network in an intensive way (e.g., messaging applications):
  - Authentication
    - When information is received from a source, authentication means that the source is indeed as alleged in the information
    - The information was not altered along the way (integrity)
  - Confidentiality
    - The information is safe from being eavesdropped on during its transit from the sending point to the receiving point
  - Choosing the best security parameters and key management
    - The choice refers to the fact that, in general, the two endpoints of a communication link may possess different computational capabilities and, also, in general, may not have access to exactly the same set of security algorithms
    - Key management refers to providing solutions to the sort of practical problems that arise when users possess multiple public/private key pairs

CHALLENGES TO PROVIDE SECURITY PROPERTIES AT DIFFERENT LEVELS

- **Transport** layer
  - Agnostic to specific applications by adding security-related features to TCP/IP
  - Example: SSL/TLS protocol
- **Application** layer
  - Embedding security in the application itself
  - Example: **Pretty Good Privacy (PGP)**
- In both layers
  - authentication can be provided by public-key cryptography and by secure transmission of message digests or message authentication codes
  - confidentiality can be provided by symmetric key cryptography
  - when public-key cryptography is used for authentication at any layer, the key-management issues in all layers can be made complicated by the fact that, in general, users are allowed to have multiple public keys

9

PGP

INTRODUCTION

- Developed originally by Phil Zimmerman
- Now available as open source project: OpenPGP
  - https://www.openpgp.org/
- The standard is described in the RFC 4880
  - https://www.ietf.org/rfc/rfc4880.txt
- Initially used to protect **email** messages…
- … now, also used to protect the confidentiality of **data at rest**
  - For this, an alternative open source tool is used, namely GnuPG
    - https://gnupg.org/
  - Since it is relatively easy for your information to be stolen from a personal computer through malware, at the least one should keep all personal information in a GPG encrypted file

Example of GnuPG usage
- gpg --cipher-algo AES256 -c file.txt
  Enter passphrase: …
- It creates a file named file.txt.gpg
  - It is crucial to remove the original file!
- gpg file.txt.gpg
- It will decrypt the file

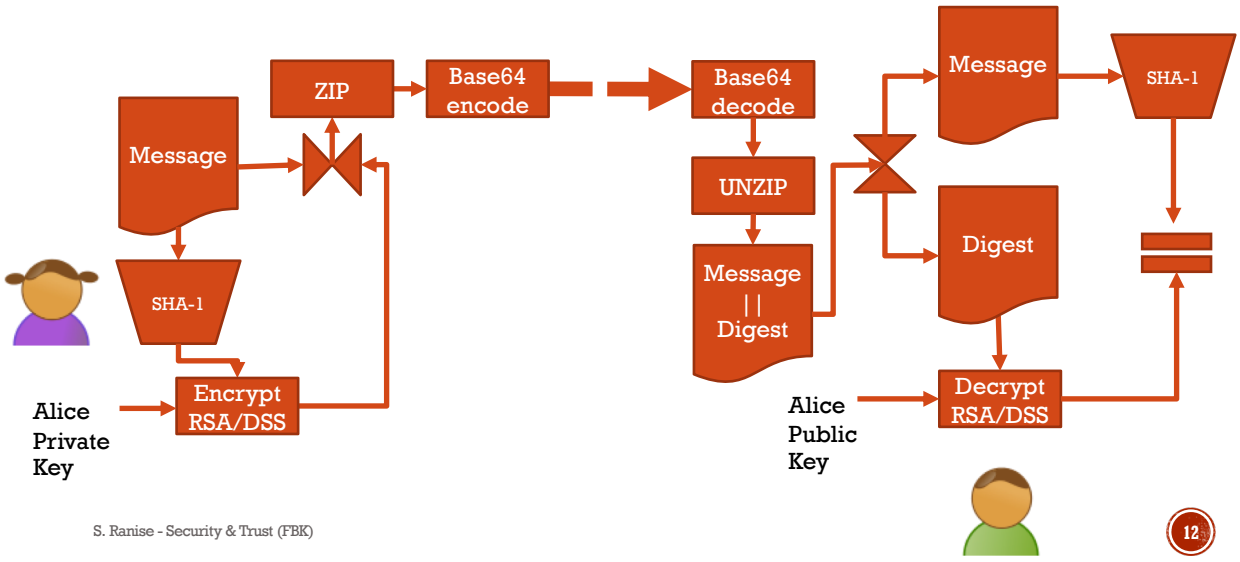
Notice that e-mail before P.G.P. was like sending an open-faced postcard through the mail
An interesting reading by the inventor of PGP: https://www.spectacle.org/795/byzim.html

PGP SERVICES: AUTHENTICATION

- Sender authentication consists of the sender attaching its digital signature to the email and the receiver verifying the signature using public-key cryptography
- The example is based on using a RSA/SHA-based digital signature
- PGP also support DSS/SHA based signatures
- *Example*
  - At the sender's end, a SHA hash function is used to create a 160-bit message digest of the outgoing email message
  - The message digest is encrypted with RSA using the sender's private key and the result prepended to the message; the composite message is transmitted to the recipient
  - The receiver uses RSA with the sender's public key to decrypt the message digest
  - The receiver compares the locally computed message digest with the received message digest

- PGP supports also detached signatures that can be sent separately to the receiver.
- These are useful when a document must be signed by multiple individuals

PGP MODE OF OPERATION: AUTHENTICATION ONLY



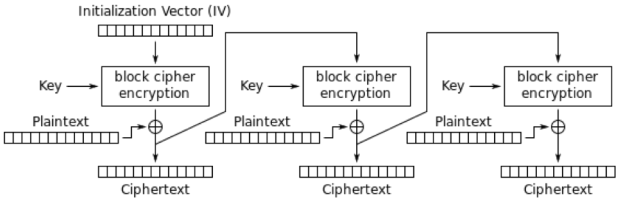
PGP SERVICES: CONFIDENTIALITY (1)

- PGP uses symmetric-key encryption for confidentiality
- 3 different block-cipher algorithms are supported for this purpose:
  - CAST-128 (default)
  - IDEA
  - 3DES

- IDEA = International Data Encryption Algorithm
- Block size = 64-bit blocks
- Key size = 128 bit keys
- It uses 8 rounds of processing on the input bit blocks, each round consisting of substitutions and permutations

- CAST-128 uses the Feistel cipher structure
- Block size = 64-bits
- Key size = between 40 and 128 bits
- Depending on the key size, the number of rounds is between 12 and 16
- How each round of processing works in CAST is different from how it works in DES
- Overall, as in DES, each round carries out a series of substitutions and permutations in the input data

PGP SERVICES: CONFIDENTIALITY (2)

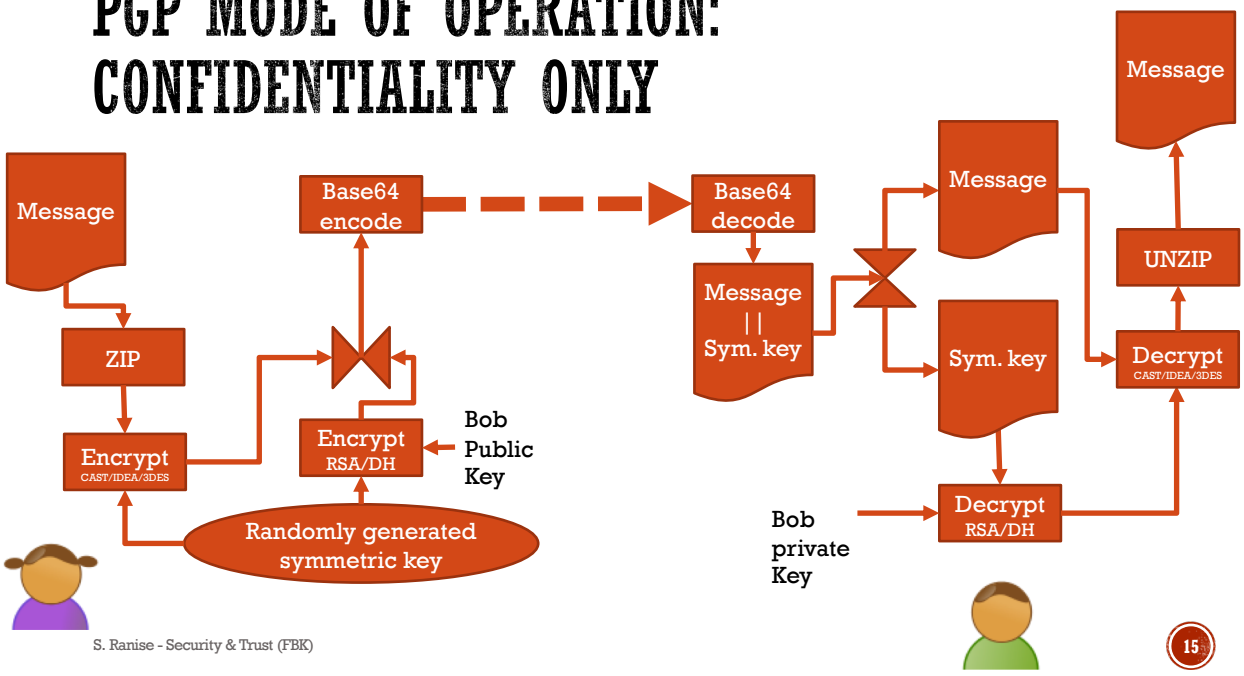


Cipher Feedback (CFB) mode encryption

Some details

- The block ciphers are used in the Cipher Feedback Mode (CFB)
- The encryption key (session key) is generated for each email message separately
- The session key is encrypted using RSA with the receiver's public key
  - Alternatively, the session key can also be established using the El Gamal algorithm
- The message sent over the network is the email message after
  - it is encrypted first with the session key and then
  - with the receiver's public key
- If confidentiality and sender-authentication are needed simultaneously, a digital signature for the message is generated using the hash code of the message plaintext and appended to the email message before it is encrypted with the session key (cf. the Authentication service described above)

PGP MODE OF OPERATION: CONFIDENTIALITY ONLY



PGP SERVICES: COMPRESSION

- By Default, PGP compresses the email message after appending the signature but before encryption
- This makes long-term storage of messages and their signatures more efficient
- This also decouples the encryption algorithm from the message verification procedures
- Compression is carried out with the ZIP algorithm

PGP SERVICES: COMPATIBILITY

- Since encryption, even when it is limited to the signature, results in arbitrary binary strings, …
- … since network message transmission is character oriented, one must represent binary data with ASCII strings
- PGP uses Base64 encoding for this purpose
- Base64 is a widely used encoding method to guarantee interoperability across different platforms and exchange binary data over networks

PGP SERVICES: SEGMENTATION

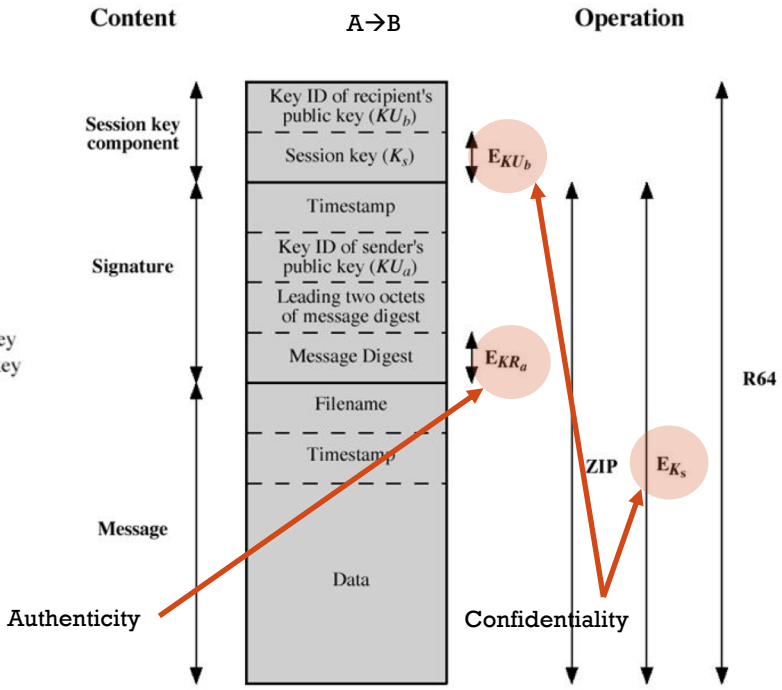
- For long email messages (generally messages containing attachments), many email systems place restrictions on how much of the message will be transmitted as a unit
- For example, some email systems segment long email messages into 50,000 byte segments and transmit each segment separately
- PGP has built-in facilities for such segmentation and re-assembly

PGP MESSAGE FORMAT

Notation:

E_{KU_b} = encryption with user b's public key
E_{KR_a} = encryption with user a's private key
E_{K_s} = encryption with session key
ZIP = Zip compression function
R64 = Radix-64 conversion function

This is Base64 encoding



20

KEY MANAGEMENT AND WEB OF TRUST

INTRODUCTION

- As seen above, public key encryption is central to PGP as it is used for both authentication and for confidentiality
  - A sender uses its private key for placing its digital signature on the outgoing message
  - A sender uses the receiver's public key for encrypting the symmetric key used for content encryption for ensuring confidentiality
- People are likely to have multiple public and private keys for a number of different practical reasons
  - For example, an individual may wish to drop an old public key, but, to allow for a smooth transition, may decide to make available both the old and the new public keys for a while
- PGP must allow for the possibility that the receiver of a message may have stored multiple public keys for a given sender
- This raises some interesting questions…

PROBLEMS WITH MULTIPLE KEYS PER USER

- **Question 1**
  - PGP uses one of the public keys made available by the recipient, how does the recipient know which public key it is?
- **Question 2**
  - The sender uses one of the multiple private keys that at its disposal for signing the message, how does the recipient know which of the corresponding public keys to use?
- Both of these problems can be solved by the sender also including the public key used….
- … this is indeed a waste in space because RSA public keys can be very large (and getting larger and larger for security as time went by)

SOLUTION

- PGP solves these problems
  - by using the notion of a relatively short key identifier (called **key ID**) and
  - requiring that every PGP agent maintain
    - its own list of paired private/public keys in what is called as a **Private Key Ring**
    - a list of the public keys for all its email correspondents in what is called as the **Public Key Ring**
- Key IDs are included in messages so that the recipient of the message knows which public key to use by using its Public Key Ring
- A Private Key Ring contains the public/private key pairs for a user
  - Private keys are stored encrypted by using a key derived from a hashed passphrase

Typically the email address

PRIVATE KEY RING

64 least significant bits of the public key

Private-Key Ring

Timestamp	Key ID*	Public Key	Encrypted Private Key	User ID*
. . .	. . .	. . .	. . .	. . .
T_i	$PU_i \bmod 2^{64}$	PU_i	$E(H(P_i), PR_i)$	User i
. . .	. . .	. . .	. . .	. . .

- The encryption algorithm asks the user to enter a passphrase
- The pass-phrase is hashed with SHA-1 to yield a 160-bit hash code
- The first 128 bits of the hash code are used as the encryption key by CAST-128 algorithm Both the passphrase and the hash code are immediately discarded

- **Timestamp:** date/time when key pair generated
- **Key ID:** least significant 64 bits of public key
- **Public key:** public-key portion of the pair
- **Private key:** private key portion of the pair
  - this field is encrypted

Public key of user *i*

Encrypt

Hash

Private key of user *i*

PUBLIC KEY RING

Multiple user identifiers can be associated with a single public key

Public-Key Ring

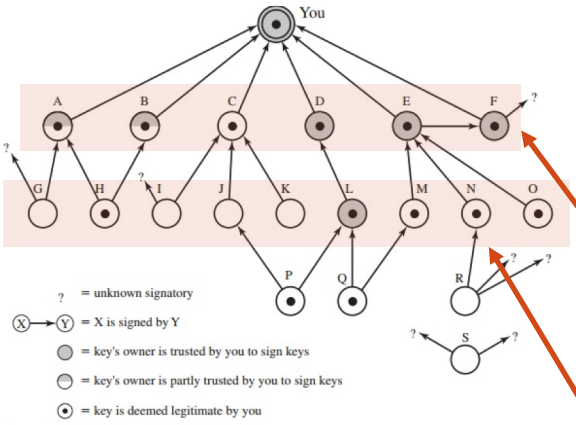
Timestamp	Key ID*	Public Key	Owner Trust	User ID*	Key Legitimacy	Signature(s)	Signature Trust(s)
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
T_i	$PU_i \bmod 2^{64}$	PU_i	$trust\_flag_i$	User i	$trust\_flag_i$		
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .

- Timestamp, Key ID, Public Key, and User ID have the same meaning as in the private key ring
- The remaining fields are used for handling trust **Owner Trust, Key Legitimacy, Signature(s), and Signature Trusts** are used to assess how much trust to place in the public keys belonging to other people

ON THE NOTION OF TRUST IN PGP

- How to measure the degree of trust is implementation dependent
  - For instance, we can have three possible values: full, partial, and none
  - A can fully trust B key if A and B have exchanged the keys manually (e.g., using USB keys)
  - A can partially trust B key if A and B have verified key IDs over the phone
- A unique feature of PGP is its own notion of a "certificate authority" for authenticating the binding between a public key and its owner
- This notion is based on PGP's **web of trust** that is a bottom-up approach to establishing trust for authentication
  - Notice that this is in contrast with the approach used in PKI that is top-down since trust can only flow downwards from the root node (that must always be trusted implicitly) to the CAs at the other nodes that descend from the root node
- In PGP's web of trust, a user's public key can be signed by any other user

AN INSTANCE OF A WEB OF TRUST



- A, B, C, D, E, F keys are signed by You
- A and B keys are partially trusted (partial shadow) to sign other entities keys
- D, E, F keys are fully trusted (complete shadow) to sign other entities keys
- C is not trusted (no shadow) to sign other entities keys
- G is signed by A and another entity not in the public key ring of You (question mark)

You acquired a number of public keys directly from their owners

The token in the circle representing an entity means that You considers legitimate the key of the entity

You acquired a number of public keys indirectly from other entities; e.g., key servers

SOME REMARKS

- In modern usage of PGP, creation of the web of trust is facilitated by the availability of free publicly available PGP Key-servers at various places around the world
  - One such server can be found at http://pgp.mit.edu/
- If you think that key management in PGP is far from being obvious and not convenient for most users except the more experienced ones, then you are not alone…
  - See, for instance, the following blog for an in-depth discussion
    https://blog.cryptographyengineering.com/2014/08/13/whats-matter-with-pgp/
- Google has released an end-to-end extension for its email service to simplify the user experience
  - See, the repository of the code at
    https://github.com/google/end-to-end

29

TOR

https://www.torproject.org/

ONION ROUTING FOR PRIVACY

- Originally sponsored by the US Naval Research Laboratory
- From 2004 to 2006 was supported by EFF
- Since 2006 it is being developed by a non-profit organization
- Motivation
  - Figure out a way to set up internet communications so that an adversary snooping on the *en route* packet traffic would not be able to analyze the packet headers for the purpose of finding out who was talking to whom
  - Gleaning information regarding the original source of the packets and their ultimate destination is referred to as the **traffic analysis attack**
  - even when protocols based on IPSec, TLS or VPN are used for establishing encrypted communication channels for the transfer of information between the web browsers and the web servers, the **packet headers are always in clear text**
- Tor is a significant attempt to address the privacy problem over the Internet

FOUR USES OF TOR

- Users of BitTorrent downloading media content generate a significant fraction of the Tor traffic

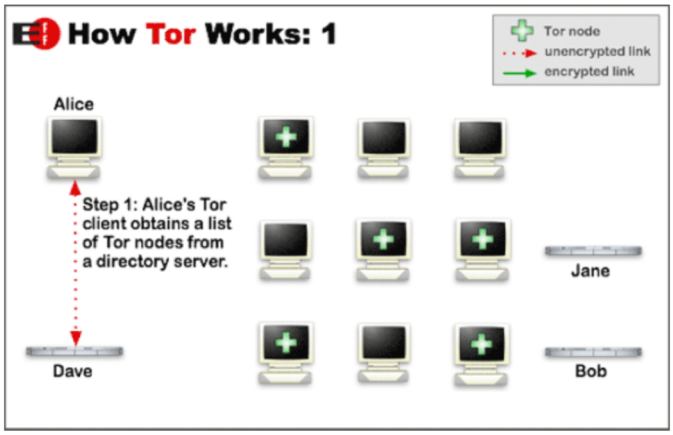
  https://hal.inria.fr/inria-00574178/document
- Tor is also popular with folks in countries where the free flow of information is restricted

  https://bitcoinmagazine.com/articles/privacy-isnt-radical-how-tor-supports-dissent-as-a-human-right
- Tor is also used by folks who want to "leak" information anonymously

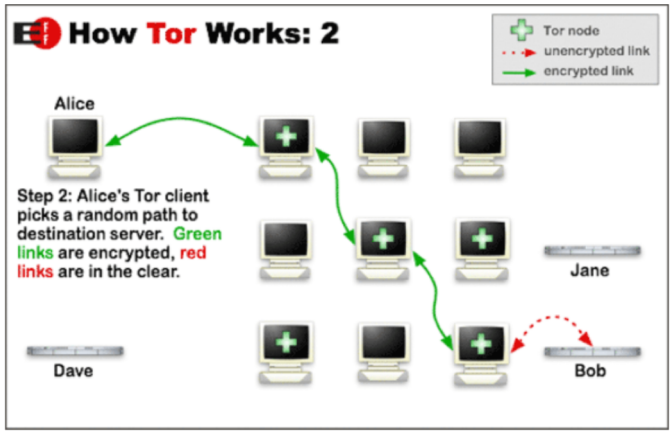
  https://wildleaks.org/
- Tor is popular for anonymous defamation

  https://reason.com/volokh/2019/05/21/court-throws-out-lawsuit-against-tor-for-providing-anonymous-routing/

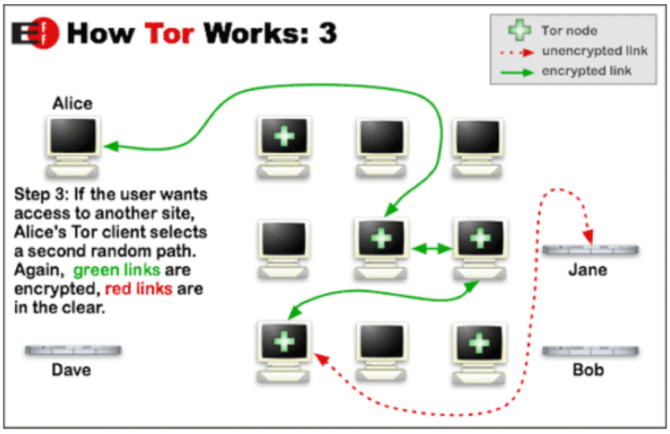
TOR: SCENARIO (1)



TOR: SCENARIO (2)



TOR: SCENARIO (3)



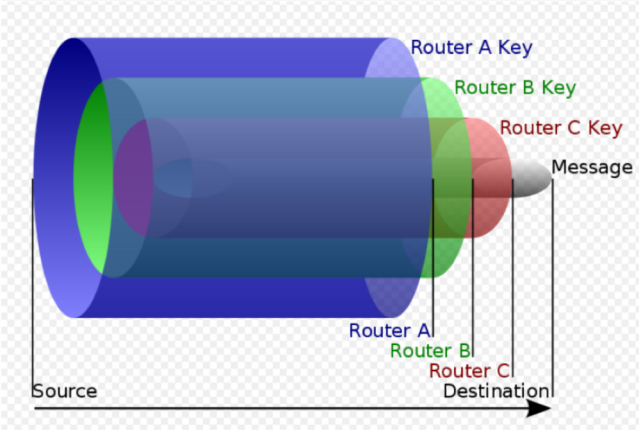
ONION ROUTING: IDEA

The source of the data sends the onion to Router A, which removes a layer of encryption to learn only where to send it next and where it came from (though it does not know if the sender is the origin or just another node). Router A sends it to Router B, which decrypts another layer to learn its next destination. Router B sends it to Router C, which removes the final layer of encryption and transmits the original message to its destination.

- Tor full specification available at

https://gitweb.torproject.org/torspec.git/tree/tor-spec.txt

- The user creates a "circuit" leading to destination
- At each hop, the node "unwraps" a layer from the packet via symmetric keys, revealing the next destination
  - Clever combination of RSA and Diffie-Hellman (including its Elliptic Curve variant) cryptographic techniques



TOR BASICS (1)

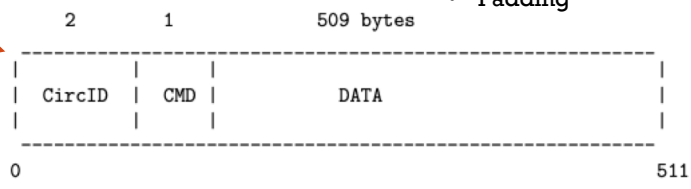
- An **overlay network** is a telecommunications **network** that is built on top of another **network** and is supported by its infrastructure
- An **overlay network** decouples **network** services from the underlying infrastructure by encapsulating one packet inside of another packet

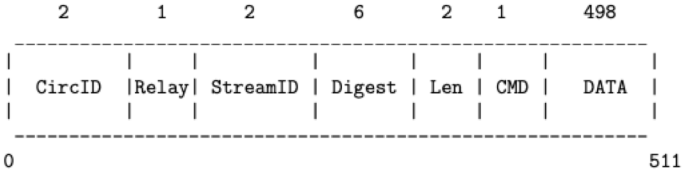
- Tor is based on the twin notions of
  - Onion Proxies (OP) and
  - Onion Routers (OR)
- A user's OP first queries a Tor directory for the IP addresses of the ORs in the **Tor overlay**
- The user then selects a subset of these ORs (typically 3) for constructing a path to the destination resource
  - The specification "onion" in OP and OR is related to the layers of encryption placed on the Tor messages such that, except for the user's OP, the routing knowledge at any single node on a path through the Tor overlay is limited to exactly two nodes, the immediately preceding node on the path and the immediately following node
- A user's OP constructs a path through the Tor overlay, that is called a **circuit**
- The two parties at the two end of a circuit may use it for an arbitrary number of TCP **streams**

- **TCP** first sets up a connection to the receiver then sends the data in segments which is carried by IP packets
- This is called **stream** because it keeps the **stream** of data between to ends during transfer

TOR BASICS (2)

- There are two types of tor packets (each consisting of 512 bytes)
  - **control** tor packets
  - **relay** tor packets





Types of control packets:
- Create
- Created
- Destroy
- Padding

Types of relay packets:
- Extend
- Extended
- Truncate
- …

TOR BASICS (3)

- Role of a control tor packet
  - Alter the relationship between the sender node and the next node on the path that receives such a packet
- Role of relay tor packet
  - As paths are constructed (and torn down) incrementally by a user's OP…
  - … the first link of the path can be constructed directly by the OP using a control tor packet…
  - … any extensions to the path are going to require that the commands for doing so be relayed to the currently last node on the path

TOR BASICS (4)

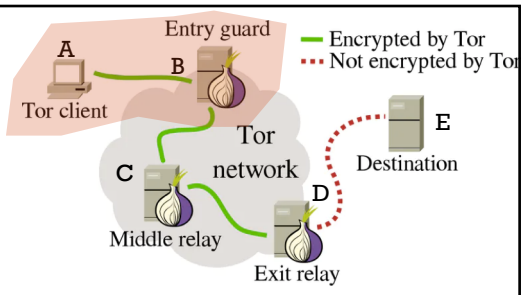
- Initially, the control and the relay tor packets work together to create an end-to-end path (i.e. a circuit) in the Tor overlay in such a way that each interior node on the path has only local knowledge of the path
- While the basic purpose of a relay tor packet is to carry the data that is exchanged between the two endpoints, that can only be done after a path is fully constructed
- During the process of path construction, the data carried by relay tor packets is for the purpose of extending the path beyond the current termination point
- Such relay tor packets generate control tor packets at the current terminal node on the path for extending the path

INCREMENTAL CREATION OF A CIRCUIT (1)

- How does a user's OP use the control tor packets to create an end-to-end circuit incrementally, one hop at a time, in the Tor overlay?

- **Assumptions**
  - every OR node has a public RSA key that it makes available to the user's OP
    - any communication sent to an OR that is encrypted with its RSA public key can only be understood by that OR
  - Diffie-Hellman (DH) keys are created on the fly between the user's OP and each of the ORs on the path chosen by the user
    - the purpose of the DH keys is that when the user's OP wants to send a message to a designated OR on the path, it is encrypted with the session key derived from the OP's DH key and that OR's DH key
  - AES is used for the symmetric-key encryption with DH session keys

INCREMENTAL CREATION OF A CIRCUIT (2)

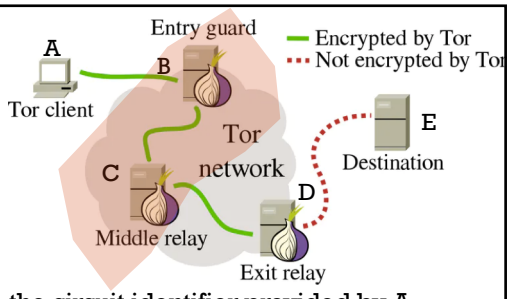


- The user's OP (A) sends a create control tor packet to the first node (B) in the path chosen by the user
  - A→B: control tor packet
  - The packet contains
    - the CircID field (circuit identifier) set to a fresh value
    - the DATA field of this packet contains A's DH key Y(A→B) that is encrypted with B's RSA public key
- B responds back to A with the created control tor packet
  - B→A: control tor packet
  - The packet contains in the DATA field B's DH Y(B→A)
- At this point, both A and B can calculate the secret session key K(AB) for their link

Remarks

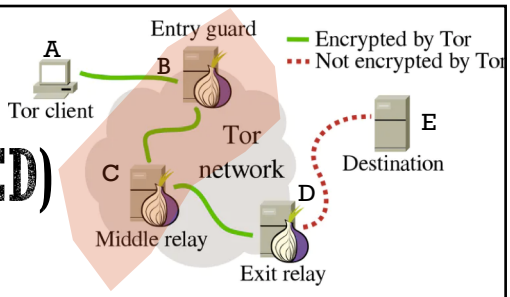
- all communications between any pair of nodes in the underlying network are over TSL
  - the public DH keys are not be visible to a packet sniffer
- The RSA public/private keys used in the transmission of the control and relay tor packets are not to be confused with the RSA public/private keys used by TLS

INCREMENTAL CREATION OF A CIRCUIT (3)



- A and B can start exchanging relay tor packets that use the circuit identifier provided by A
- To extend the circuit, A sends B a relay tor packet with the relay extend command
  - A→B: relay tor packet
  - The packet contains in the DATA field a DH key Y(A→C) that is meant specifically for the new terminal node C on the path and also includes the identity of the new node
  - To guarantee that the key Y(A→C) is not seen by B, it is encrypted with C's RSA public key
  - The DATA field in the relay extend tor packet from A to B is encrypted with the session key K(AB)
- When B receives the relay extend tor packet from A, it knows that it is the current endpoint on the path and generates a control tor packet for C
  - B→C: control tor packet
  - The packet contains a DATA field with A's DH key Y(A→C) that is meant specifically for node C and that is encrypted with C's RSA public key
  - This DATA field is encrypted with C's RSA public key

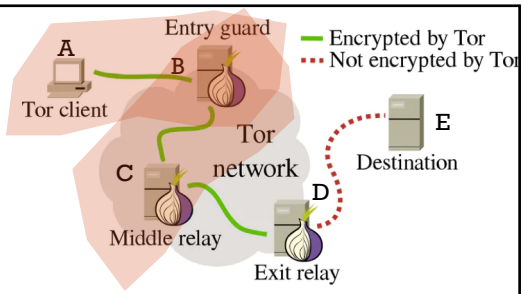
INCREMENTAL CREATION OF A CIRCUIT (3 CONTINUED)



Important observation

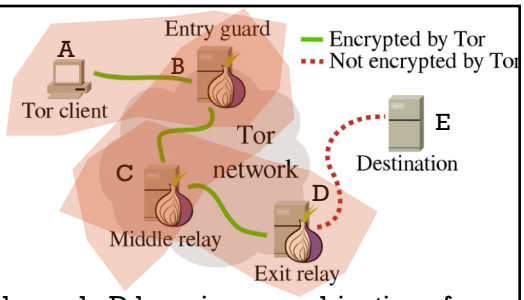
- The control packet sent by B to C uses a new randomly generated number for the circID field that becomes the identifier for the segment of the circuit between the nodes B and C
- There is no need for A to know this identifier
- Only node B knows both circID identifiers (the one for the circuit between A and B and the one for the circuit between B and C)
- This fact plays an important role in ensuring that each node on the path has only the local knowledge of the path

INCREMENTAL CREATION OF A CIRCUIT (4)



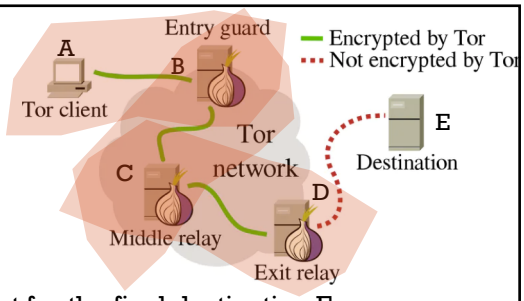
- Node C responds back to B with a created control tor packet
  - C→B: control tor packet
  - This packet contains a DATA field with C's DH key Y(C→A) meant for A
- Node B sends this acknowledgment back to A using the relay extended tor packet
  - B→A: relay extended tor packet
  - This packet contains a DATA field with the key Y(C→A)
- Now both A and C can calculate the secret session key K(AC) for any messages that A may want to send to C through B that B is not allowed to see

INCREMENTAL CREATION OF A CIRCUIT (5)



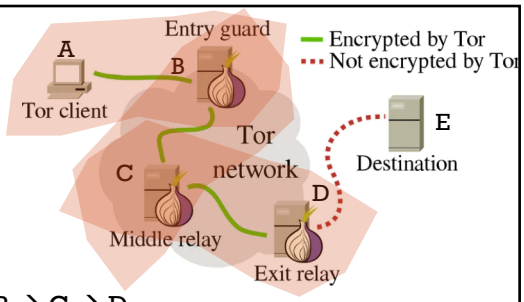
- The path may be extended in the same manner to the node D by using a combination of control and relay tor packets
- Notice that, in constructing the end-to-end circuit, there was never a need for using A's public RSA key
- As a consequence, the user A remains anonymous to all the ORs in the circuit
- But all the ORs in a circuit are known to the user A
  - This is somehow obvious since, remember, A has chosen all the nodes in the circuit

USING THE CIRCUIT (1)



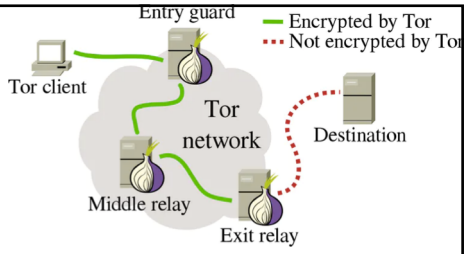
- A can start pushing data into the circuit that is meant for the final destination E
- Preliminarily, A sends a relay begin tor packet to B, from where it is forwarded to the next node on the circuit, and so on, thus creating an end-to-end stream between A and E
- The user A is allowed to create an arbitrary number of streams sharing the same circuit
- While the different TCP streams will have different streamID values in the relay tor packets that carry the stream data, they will have the same value for the circID field
  - Even though the value of this circID field will change from hop to hop in a circuit

USING THE CIRCUIT (2)



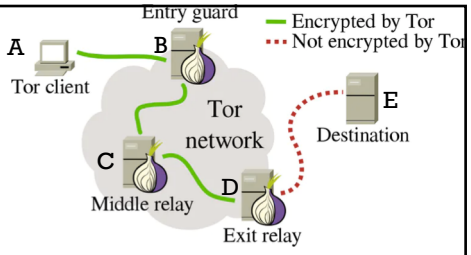
- Assume the following path in the Tor overlay: A → B → C → D
- The stream data that the user A places on the wire is encrypted with the K(AD) session key, followed by its encryption with K(AC) session key, followed by its encryption by K(AB) session key
- As these stream data bearing relay data tor packets are received by B from A, the node B uses the session key K(AB) to decrypt the top layer of encryption and forward the stream to node C in the circuit
- This process continues until the stream data reaches the final node D, from where it goes via the normal TCP transmission to the application running at the destination E

TWO IMPORTANT QUESTIONS ABOUT TOR



- **Question 1**
  - Can the exit node operator see the source IP address, meaning the IP address of node A?
- **Question 2**
  - Can the exit node operator see the data payload of the source packet?
- **Answer to question 2**
  - If node A is trying to reach an HTTPS web site, that implies the usage of encryption of the payload in the packets
  - In that case, the exit node operator obviously cannot see inside the packets that A is sending out

ANSWER TO QUESTION 1



- **Question 1**
  - Can the exit node operator see the source IP address, meaning the IP address of node A?
- In principle, the answer is no, i.e. it should not be possible for the exit node to know the source IP address
- This is so because the Tor logic that keeps A's IP address shielded from the exit node D is the same as the logic that keeps B's IP address shielded from D
- The packets that go out from D to the web server at E should only bear D's IP address in the source fields
- When D receives replies to those packets from the web server, it simply forwards them back to C
- Unfortunately, there is an attack that allows the exit node operator to identify the IP address of the source…

The details can be found at https://www.usenix.org/legacy/events/leet11/tech/full_papers/LeBlond.pdf

ONE BAD APPLE SPOILS THE BUNCH

- Paper at Usenix 2011
  - Subtitle: **Exploiting P2P Applications to Trace and Profile Tor Users**
- The authors were able to successfully reveal the source IP addresses of 10,000 hosts that used Tor for BitTorrent downloads during a period of 23 days
- Attack based on peculiarities of the BitTorrent protocol
- Being a P2P protocol, a BitTorrent client must somehow acquire a list of the peers that are the keepers of the media content that the client wishes to download and then, subsequently, join the peers
- BitTorrent gives a client three different ways to discover the peers including
  - By contacting a centralized tracker that keeps a list of all the peers currently in possession of the media content of interest to the client
  - The peer list of IP addresses that is received by a client is without encryption
  - Since this list consists of the other users of BitTorrent, by simply monitoring an exit node, it is possible to figure out the identities of the BitTorrent users

TOR IS BLOCKED IN SOME COUNTRIES (1)

Preliminary observations

- Tor has a few special servers called **Directory Authorities** that maintain a list of the IP addresses of all the currently available relays for setting up Tor circuits
- The IP addresses of all these servers are hardcoded into a Tor client
- Tor is an open-source project and all its source code is accessible
  - So, the directory authorities are easily identified
- Each Tor non-exit and exit relay sends information about itself to these Directory Authority servers once every 18 hours
- The Directory Authority servers compile this information and publish a list of all the current non-exit and exit relays once every hour
- Anyone can query a Directory Authority server and download a list of all the currently operational exit and non-exit Tor relays

TOR IS BLOCKED IN SOME COUNTRIES (2)

- Since the list of all the Tor exit and non-exit relays is publicly available, any authoritarian country can obviously block all of these IP addresses at all its major network traffic routing points and thus make Tor unusable in that country
- In addition, since anyone downloading the Tor software can turn their host into a Tor relay, what if the authoritarian country's agents own a small number of relays situated in other countries?
  - Since under ordinary circumstances relays are chosen randomly as entry points, that country would be able to track unauthorized use of Tor by its citizens
- Is there a way to overcome this problem?
  - The answer is yes and is based on the notion of **bridge…**

More information on tor bridges at
https://2019.www.torproject.org/docs/bridges.html.en

CIRCUMVENTING THE BLOCK

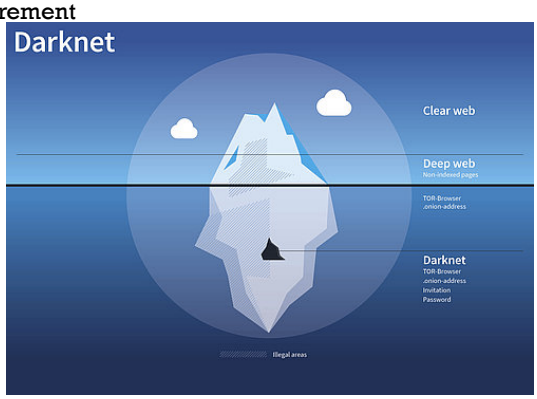
- The idea is to use an entry point which is not among those listed by the Directory Authorities
  - I.e. if a Tor client could somehow connect with an entry point in the Tor network of relays, it would then be able to construct the rest of a Tor circuit that is guaranteed to work because all the relays in the circuit are going to be in other countries and thus outside the jurisdiction of the country that is censoring Tor
- A Tor **bridge** is a third type of a relay, besides middle and exit relay, that allows for the realization of the above idea
- The only difference between a bridge and the other two types of relays is that a it does **not** publish its information to any Directory Authority
- A Tor user may, for example, turn his/her client into a bridge relay and let his/her friends know about its IP address through direct communication, such as by phone, text, or email
- Since such a relay would not broadcast its presence to a Directory Authority, it would remain unblocked until its presence is discovered

BLOCKING TOR BRIDGES

- The Great Firewall of China (GFC) now has the ability to block tor bridge relays by packet filtering at the major network traffic routing points in the country
- These packet filters, operating at network speed, use what is known as Deep Packet Inspection (DPI)
  - Roughly, DPI means that not only headers are analysed but also the data part is considered
- In the context of Tor, DPI may be based on the nature of SSL/TLS handshake used by Tor packets
- Once a packet is suspected of trying to make a connection with a bridge relay, the adversary can confirm whether or not the destination IP address is a bridge relay by sending it a packet with the purpose of initiating the construction of a circuit
- If the targeted IP address turns out to be a bridge relay, that address can subsequently be blocked

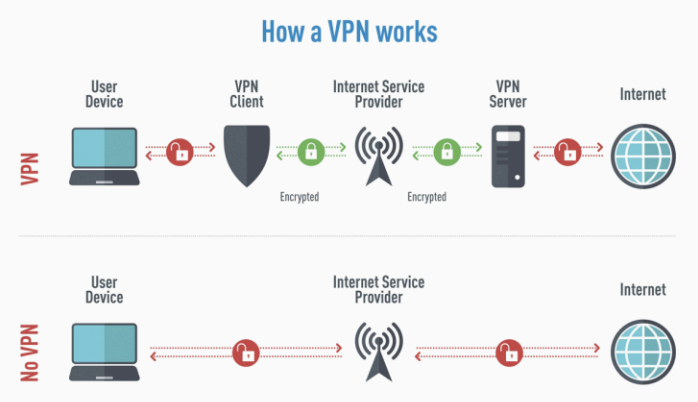
SOME FINAL REMARKS

- Usability of Tor can be improved
  - A hard-to-use system has fewer users — and because anonymity systems hide users among users, a system with fewer users provides less anonymity. Usability is thus not only a convenience: it is a ~~security~~ privacy requirement
- Tor is slow
  - A faster alternative seems to be Hornet
  - Described in this paper:
    https://arxiv.org/pdf/1507.05724v1.pdf
- Tor is the gate to the darknet



56 TOR OR VPN?

VPN (VIRTUAL PRIVATE NETWORK)



- VPNs are increasingly used because of smart working because of the pandemics
- **When you connect with a service in the internet through a VPN server, the service will only see the IP address of the VPN server, and not your actual IP address**

https://www.yellowstonecomputing.net/uploads/2/2/1/6/22165724/how-a-vpn-works-infographic-730x484_orig.png

VPN AND PRIVACY

- If someone is in a country that forbids directly connecting with a service in the internet, one might be able to access that service through a VPN server in another country and, in the process, one might be able to get past the access restriction imposed by your government
- To the extent that the destination server will not see your IP address does give you a measure of anonymity, but not to the same extent you get with Tor: the logs at the VPN proxy server would surely know one IP address
- However, using VPN in the manner described above to circumvent censorship often fails because third-party VPN servers one might use often have fixed IP addresses that can easily be blocked by the authorities simply by packet filtering at the main routing points in a country
- For alternatives that mitigate the problems above, see
  - VPN Gate (https://www.vpngate.net/en/)
  - Opera browser (https://www.opera.com/)