Technical approaches to privacy: privacy-enhancing technologies (PETS)

b. Crypto-based solutions (Overview)

Cryptographic Mechanisms for Privacy Protection

- Anonymous communication
  - Tor
- Anonymous credentials
- Blind signatures
- Secure multiparty computation
  - Garbled circuits
  - Secret sharing
- Deterministic encryption
  - Order-preserving encryption
- Computing on encrypted data
  - Homomorphic encryption
- Oblivious RAM
- Private information retrieval
- Zero-knowledge proofs
- Etc.

Cryptographic Mechanisms for Privacy Protection

- **Anonymous communication**
  - **Tor**
- Anonymous credentials
- Blind signatures
- Secure multiparty computation
  - Garbled circuits
  - Secret sharing
- Deterministic encryption
  - Order-preserving encryption
- Computing on encrypted data
  - Homomorphic encryption
- Oblivious RAM
- Private information retrieval
- Zero-knowledge proofs
- Etc.

Definition of the rest, details in Week4

Anonymous Communication

- “The exchange of information without revealing the identity of the sender or the recipient”

    - Different from end-to-end encryption -> hides the content of the message

- Anonymous communication can be achieved with various technologies or algorithms, and the level of anonymity can vary depending on the methods

Examples: **Anonymous Messaging Apps, Anonymous Browsing (TOR, Mixnets, etc.), some forms of blockchains, etc.**

Anonymous communications - introduction

20th century perception of the Internet

"On the Internet, nobody knows you're a dog."

NY Times, July 5, 1993

21th century perception of the Internet

Credit:
Arvind
Narayanan

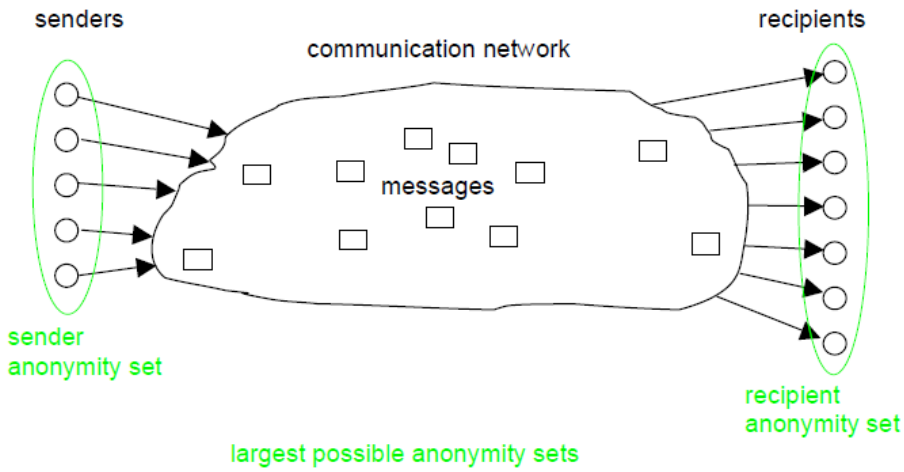
It's the Internet! Of course they know you're a dog. They also know your favorite brand of pet food and the name of the cute poodle at the park that you have a crush on!

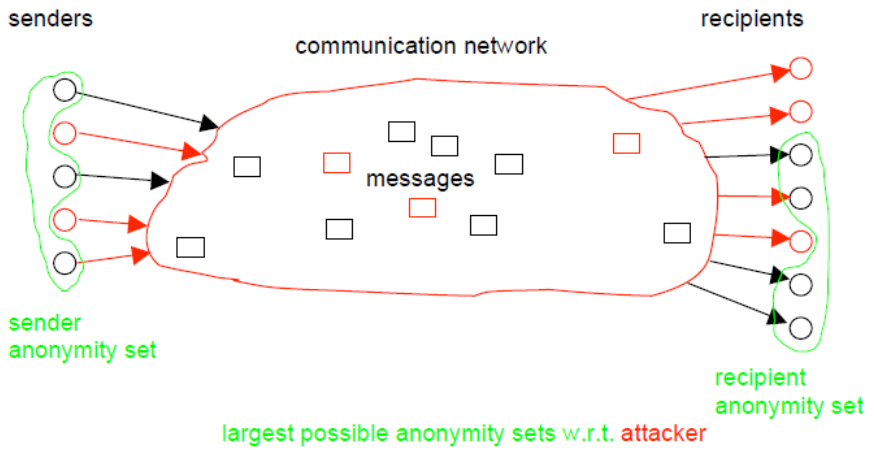
Reality

- Today's communications infrastructure is capable of identifying and recording
  - Who we are
  - What we do
  - Where we go
  - What we say
  - What we buy
  - Who are our friends/family
- Anonymity on the Internet seems to be highly desirable

Reminder - Anonymity

- Anonymity: state of being not identifiable within a set of subjects, the anonymity set
- Anonymity is stronger if:
  - larger anonymity set
  - even distribution of the sending and/or receiving





Anonymous Communications

- Infrastructure running on top of the existing Internet protocols
- Allow people to communicate without revealing their personal network IDs
  - E.g., IP addresses
- Provide a strong foundation for censorship resistance
  - Particularly well suited for people living under oppressive regimes

Who Needs It?

- Regular citizens
  - Against a stalker or censor
- Law enforcement
  - Against an investigated suspect
    - “Why is alice.fbi.gov visiting my website?”
- Companies
  - Against a competitor
    - “Hey it’s Alice, give her the ‘Alice’ version”
- Governments
  - Against untrusted ISPs

Traffic Analysis

- Ignores the content of messages
  - Payloads of the packets can be encrypted
- Tries to derive as much information as possible from only the payload and network traffic metadata
  - Source - destination
  - Packet arrival times
  - Message lengths

Adversarial Models

- Capability
  - Passive: able to monitor and record the traffic on network links
  - Active: “Passive” + ability to inject/modify/delete traffic
- Visibility
  - Partial
  - Global
- Mobility (in case visibility is partial)
  - Static
  - Adaptive
- Participation
  - External
  - Internal (has compromised one or several clients / pieces of the network infrastructure)

Anonymity Systems

- High-latency anonymity systems
  - Mixes
  - Mix networks
- Low-latency anonymity systems
  - Anonymizer.com
  - Onion routing
  - Crowds
  - Tor (anonymity network)

High-latency anonymity systems

(Only an emphasis on Mixnets)

High-Latency Anonymity Systems

- Provide strong anonymity
- Only applicable for applications that can tolerate delays of several hours or more
- Example:

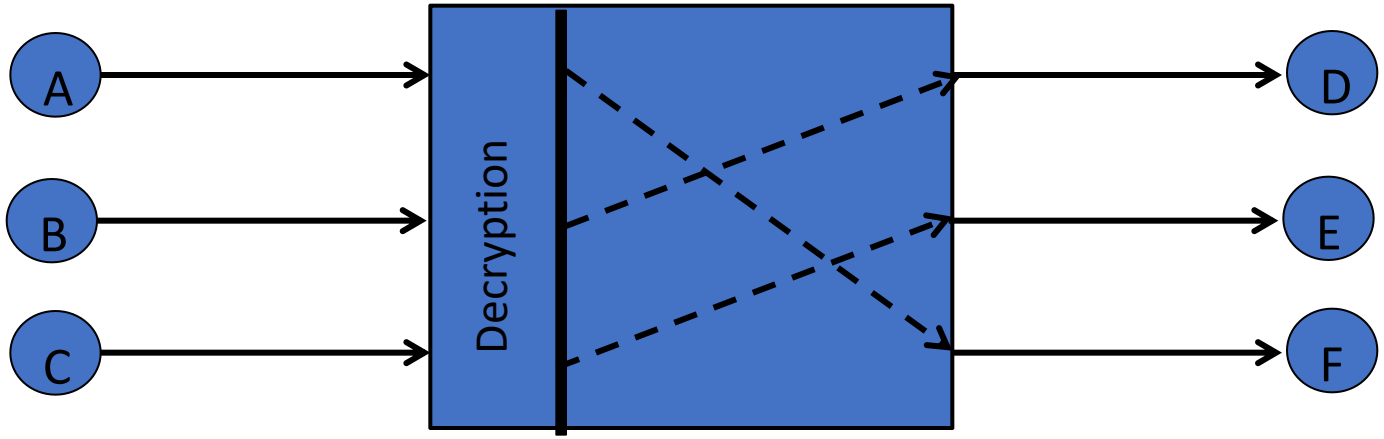
Alice anonymously sends a letter to NY Times

Bob only knows the next hop in the path and not the letter's content or its intended destination

Mixes

- Introduced by Chaum, 1981
- Idea:
  - Accepts encrypted messages as input,
  - Groups several messages together into a *batch*
  - Decrypts and forwards some or all of the messages
- Example:
  - Alice wants to anonymously send a message M to a recipient Bob *via* a single mix x
  - Alice computes $E_x(R_x, M, A_B)$ and sends to the mix
    - R_x is a string of random bits, and A_B is Bob's address
    - Mix decrypts the input and forwards M to Bob based on the mixing strategy
- An observer cannot link messages based on their arrival and departure times
  - Messages are delayed, batched, and reordered
- Sender must trust the mix

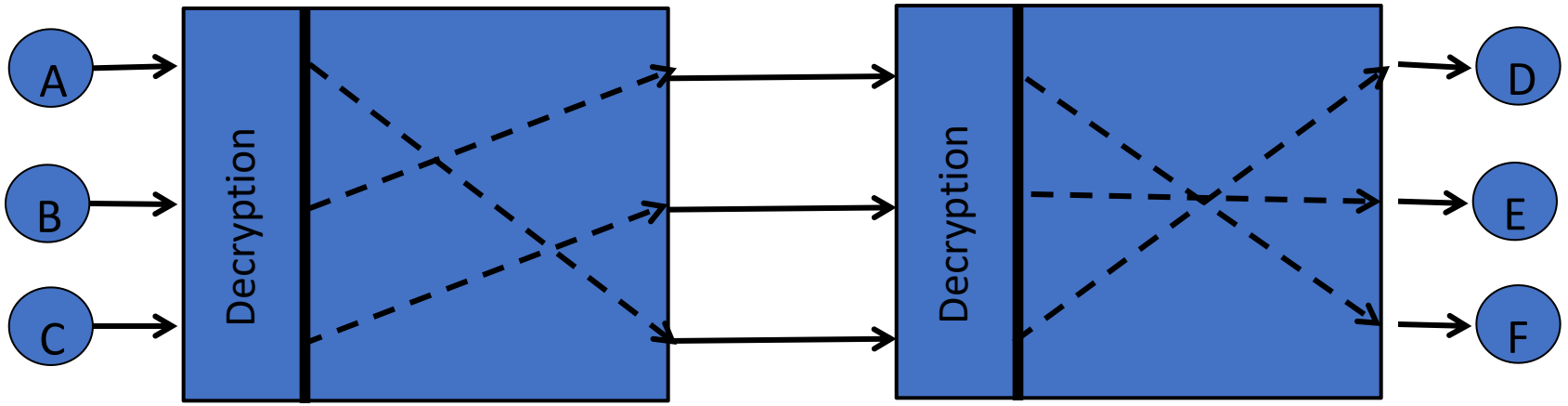
Mix: Example



Input to the mix E(message, IP_Dest)

Risk of timing attacks → the mix should introduce random delays on the packet traversal; it can also generate dummy packets to further confuse the adversary

Mix Cascading



Unless all mixes are compromised, no one can know who communicates with whom ➔ unlinkability

An identifier (pseudonym) can be used to identify a given path (in case multiple messages need to be sent)

Mix Networks

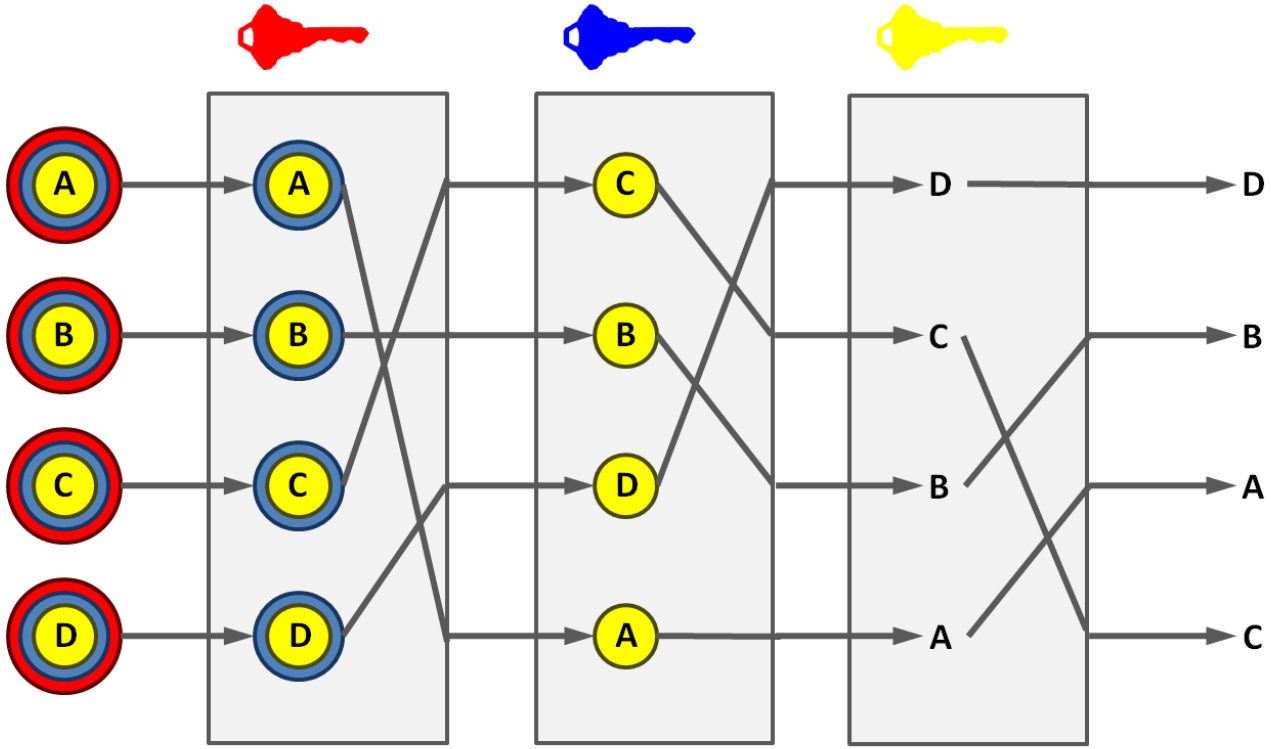


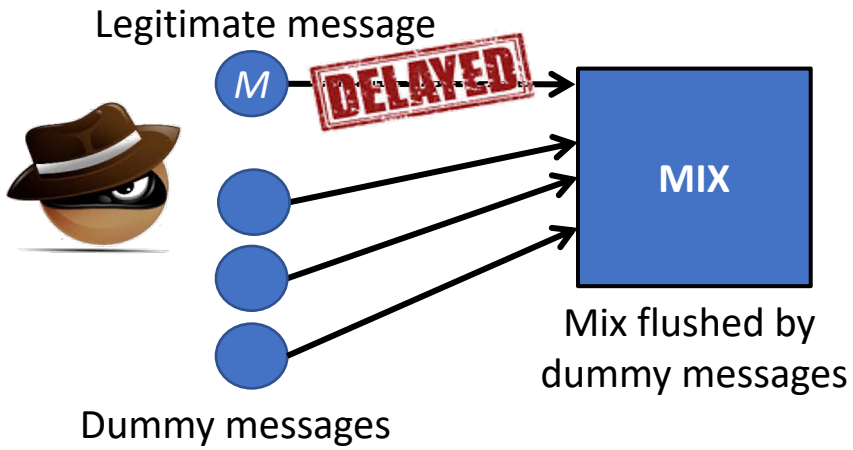
Figure: Wikipedia

- Sequence of mixes
- Each mix along a message's path knows only the hop before and after itself
- An identifier (pseudonym) can be used to identify a given path (in case multiple messages need to be sent)

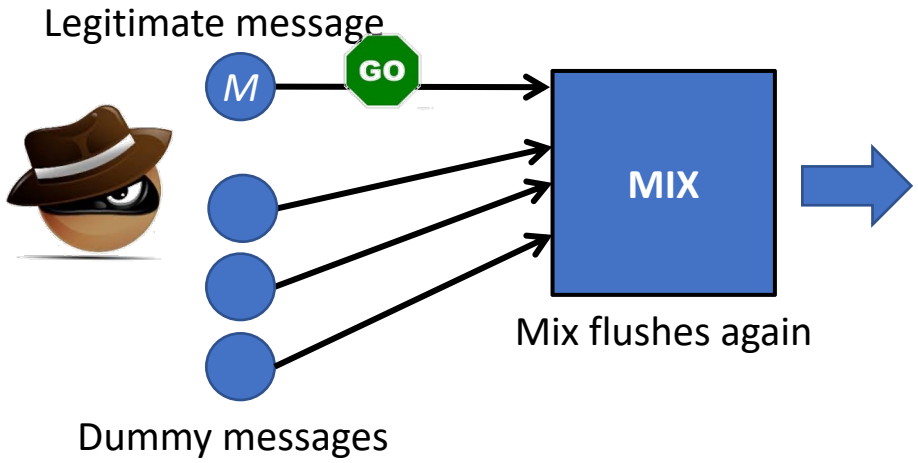
Flushing Algorithms

- Determine which messages to forward to their next destination and when to do so
- Threshold mixes
  - Collects incoming encrypted messages until it receives n messages
  - Then decrypts all of them and forwards them to their next destination in random order

(n-1) attack



Most mixes are subject to (n-1) attack



Only mix output that is not one of the dummy messages corresponds to *M*

(n-1) attack

- Also called “flooding attack”
- Attacker delays a legitimate message *M* about to enter the mix
- Generates and sends his “dummy messages” into the mix until it flushes
- Then allows *M* into the mix and sends more of his dummy messages until the mix flushes again
- The only mix output that is not one of his dummy messages thus corresponds to input *M*

Flushing Algorithms

- **Timed mixes**
  - Collects messages for a fixed length of time *t*
  - Vulnerable to an active attack similar to the (n−1) attack (called “trickle attack”)
- **Threshold and/or times mixes**
  - Collects messages until either it has received *n* messages *or* until *t* seconds have elapsed
  - Still vulnerable to (n-1) and trickle attacks
- **Pool mixes**
  - Selects a random subset of the collected messages to flush
  - Waits until it receives *n* messages in addition to the *Np* messages in the pool
  - Randomly selects *n* messages from the *n*+*Np* total messages in the mix to forward
  - Timed pool mixes operate similarly
  - *Dynamic* pool mixes send a fraction of the total messages in the pool
  - (n-1) attack becomes probabilistic

Flushing Algorithms

- **Stop-and-go mixes**
  - Instead of batching messages together, individually delay messages
  - Delay is based on exponential distribution
- **Binomial mixes**
  - At the end of a mix round lasting *t* seconds, a binomial mix flips a biased coin with a forwarding probability $p = P(n)$ for each message
  - $P(n)$ is the coin's bias when the mix contains *n* messages
- **RGB (Red Green Black) mixes**
  - Can detect the (n-1) attack
  - Genuine client messages received during a mix round are considered *black* messages
  - To estimate the number of legitimate messages received during a round, mixes send out "heartbeat" traffic, called *red* messages
    - Red traffic is "anonymously" addressed back to the mix itself
  - If the mix detects a statistically significant drop in its received red traffic, it can deduce that it is currently under attack
    - If there is an attack, red messages will delay a lot due to attacker's dummy messages
  - The mix generates additional dummy messages, called *green* traffic
    - To increase the anonymity of the legitimate, black messages output during each round

Deployed High-Latency Anonymity Systems (all dead or dying)

- anon.penet.fi
  - Simple, single-server system
  - Shut down in 1996 after lawsuit from Church of Scientology
- Cypherpunk Remailers (or Type I)
  - Loosely based on Chaum's mix design
  - Multiple, distributed servers
- Mixmaster (or Type II)
  - Improved version of Cypherpunk Remailers
  - Uses a timed dynamic pool flushing algorithm
- Mixminion (or Type III)
  - Includes directory servers for clients to learn about remailers
  - http://mixminion.net

Low-latency anonymity systems

(With an emphasis on TOR)

Low-Latency Anonymity Systems

- Flushing algorithms introduce large delays
- Need to improve performance for real-time applications (web browsing,…)
- Susceptibility to certain traffic analysis attacks increases

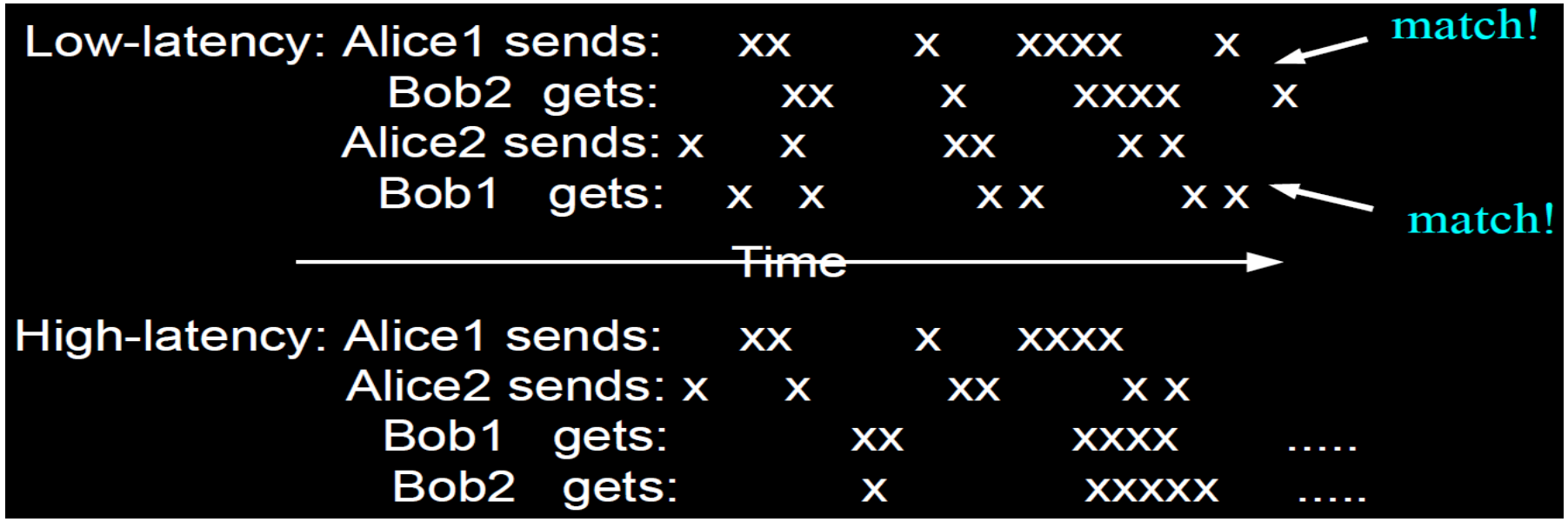


Figure: Paul Syverson, U.S. Naval Research Laboratory

anonymizer.com

- One of the first Web privacy companies (1995)
- Creates a VPN link between its servers and users' computers, creating a random IP address
- Based on the notion of *proxy*
- Forward all incoming traffic (e.g., a TCP connection) immediately without any packet reordering
- Requires clients to trust the company to not monitor or log their traffic

anonymizer.com - Example

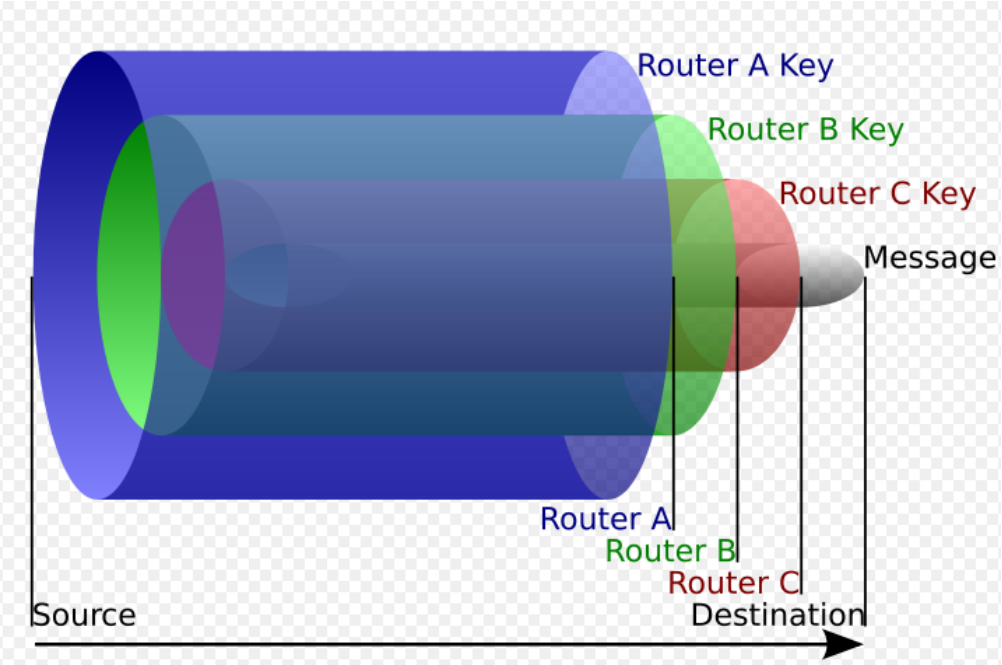
Crowds

- Designed for anonymous Web browsing
- Users in the system are represented by processes called *jondos* (a la "John Doe")
- When a user makes a Web request, his jondo randomly picks another jondo from the crowd and forwards the request
- jondo flips a coin biased with a forwarding probability $p_f > ½$
- Depending on the outcome of the coin flip:
  - the jondo either randomly selects another member of the crowd to which the request will be forwarded, or
  - it forwards the request to the intended Web server
- Reply is relayed via the reverse of the established path
- When a jondo receives a request it does not know if that jondo was the original requester or if it simply forwarded the request for another jondo

Onion Encryption and Onion Routing

- Set of servers called *onion routers* that relay traffic for clients
- Route: Source → Relay A → Relay B → Relay C → Destination
- Uses public key cryptography to establish the encrypted circuit
- Uses faster **symmetric key cryptography** to transfer the actual data
- Before transmission, the Source performs onion encryption:
- $E_{PuA}(E_{PuB}(E_{PuK_C}(E_{PuK_{Dest}}(\text{message}))))$

- Node A "peels off" the first layer of encryption, Node B the next one, etc.



Tor (The Onion Router)

- Worldwide anonymous network
- Used by journalists, whistleblowers, bloggers, law enforcement officers,…
- Main protection is against **traffic analysis** (a threat not solved by simply encrypting the messages)
- **No individual relay** ever knows the complete path that a data packet has taken
- Only the first node (Router A) knows the IP address of the sender
- A longer route provides a higher guarantee of anonymity (in practice, Tor routing uses 3 relays)

Tor - Overview

- Goal: to prevent attackers from linking communication partners, or from linking multiple communications to or from a single user
- Adversary's goal: try to link an initiator Alice with her communication partners, or try to build a profile of Alice's behavior
- Adversary model:
  - Can observe some fraction of network traffic
  - Can generate, modify, delete, or delay traffic
  - Can operate onion routers of his own
  - Can compromise some fraction of the onion routers
- Clients choose a path through the network and build a circuit
- Tor uses an incremental or *telescoping* path-building design
  - Initiator negotiates session keys with each successive hop in the circuit
  - Session keys are discarded after usage ➔ Tor users enjoy perfect forward secrecy
  - Each node in the path knows its predecessor and successor, but no other nodes in the circuit

Tor – Use Cases

- Individuals use Tor to prevent websites from tracking them
  - E-commerce site may use price discrimination based on your country or institution of origin
- Individuals use Tor when specific services are blocked by their local Internet providers
- Journalists use Tor to communicate more safely with whistleblowers
  - Edward Snowden used Tor to send information about PRISM to the Washington Post and The Guardian
- Activist groups recommend Tor as a mechanism for maintaining civil liberties online
- Law enforcement uses Tor for visiting websites without leaving government IP addresses in their web logs

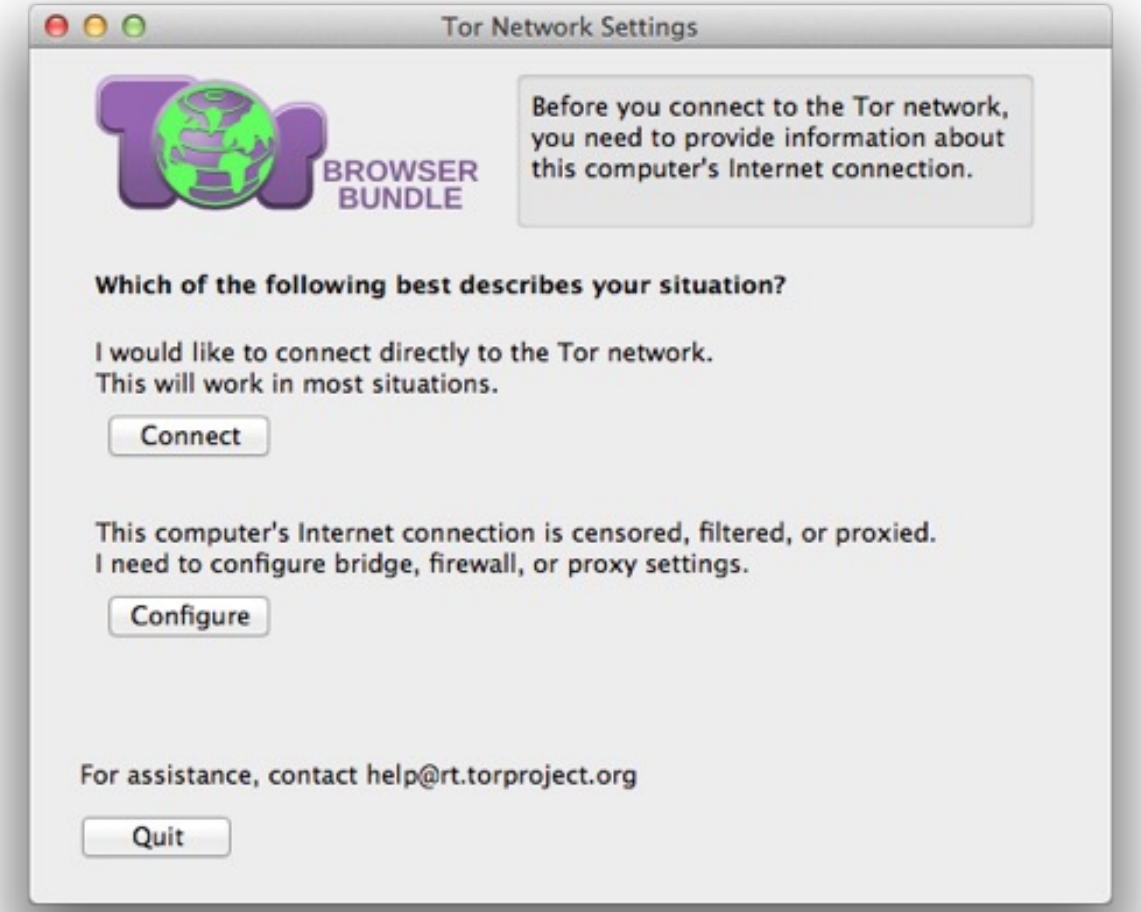
Tor – How Can you Use It?

- Tor browser: withholds some information about your computer's configuration while browsing the Web

https://www.torproject.org/projects/torbrowser.html

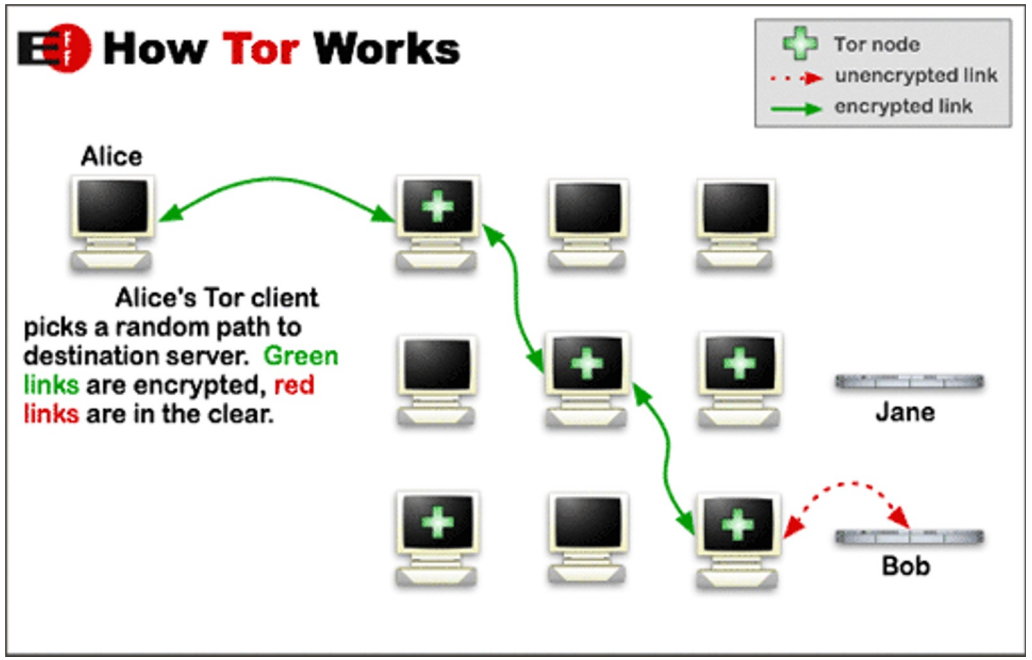
Getting in Touch with Tor

Download the Tor browser and then:



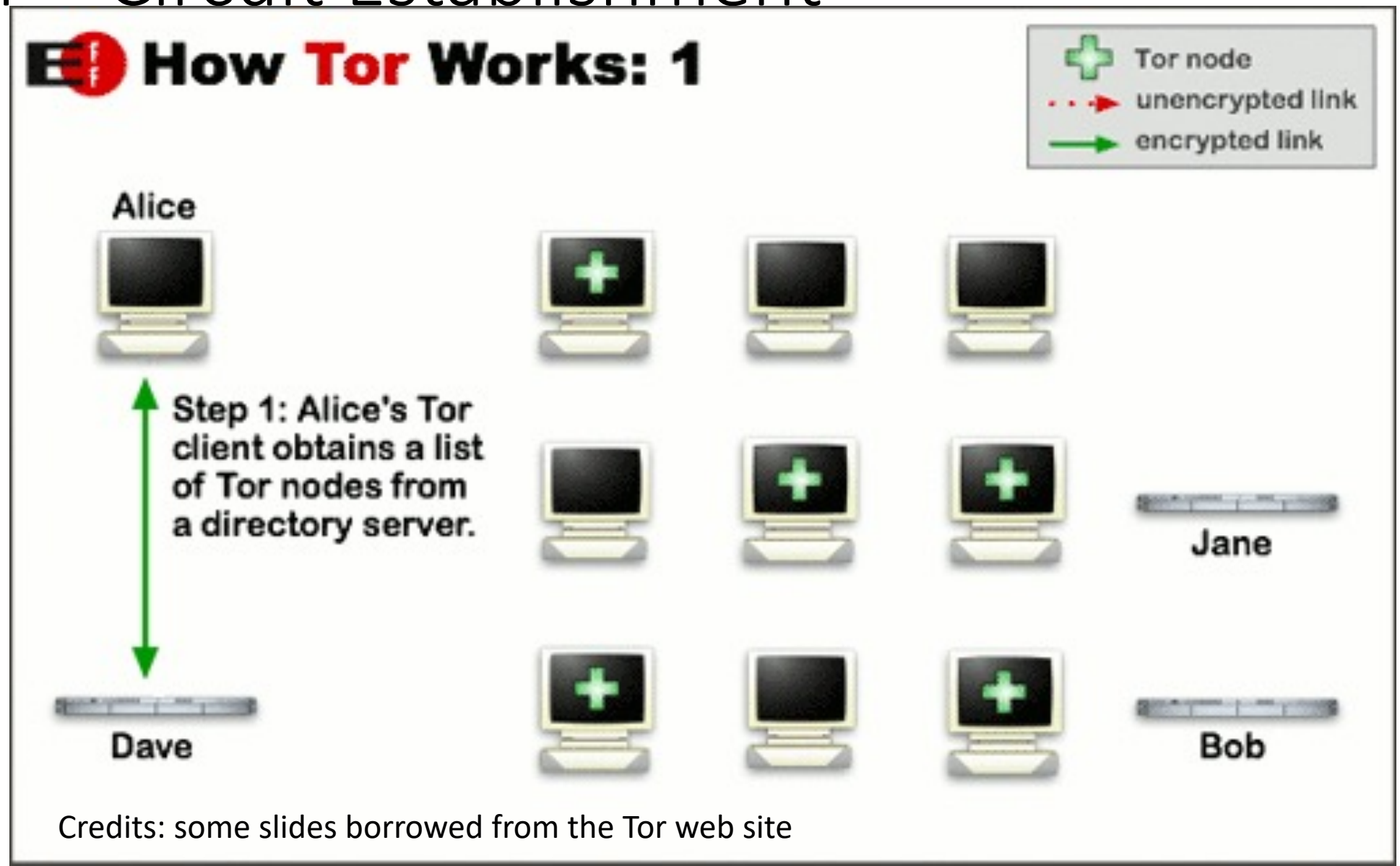
TOR

- Anonymity of participants is usually achieved by **special routing overlay networks** that hide the physical location of each node from other participants



https://www.torproject.org/about/overview.html.en

Tor – Circuit Establishment

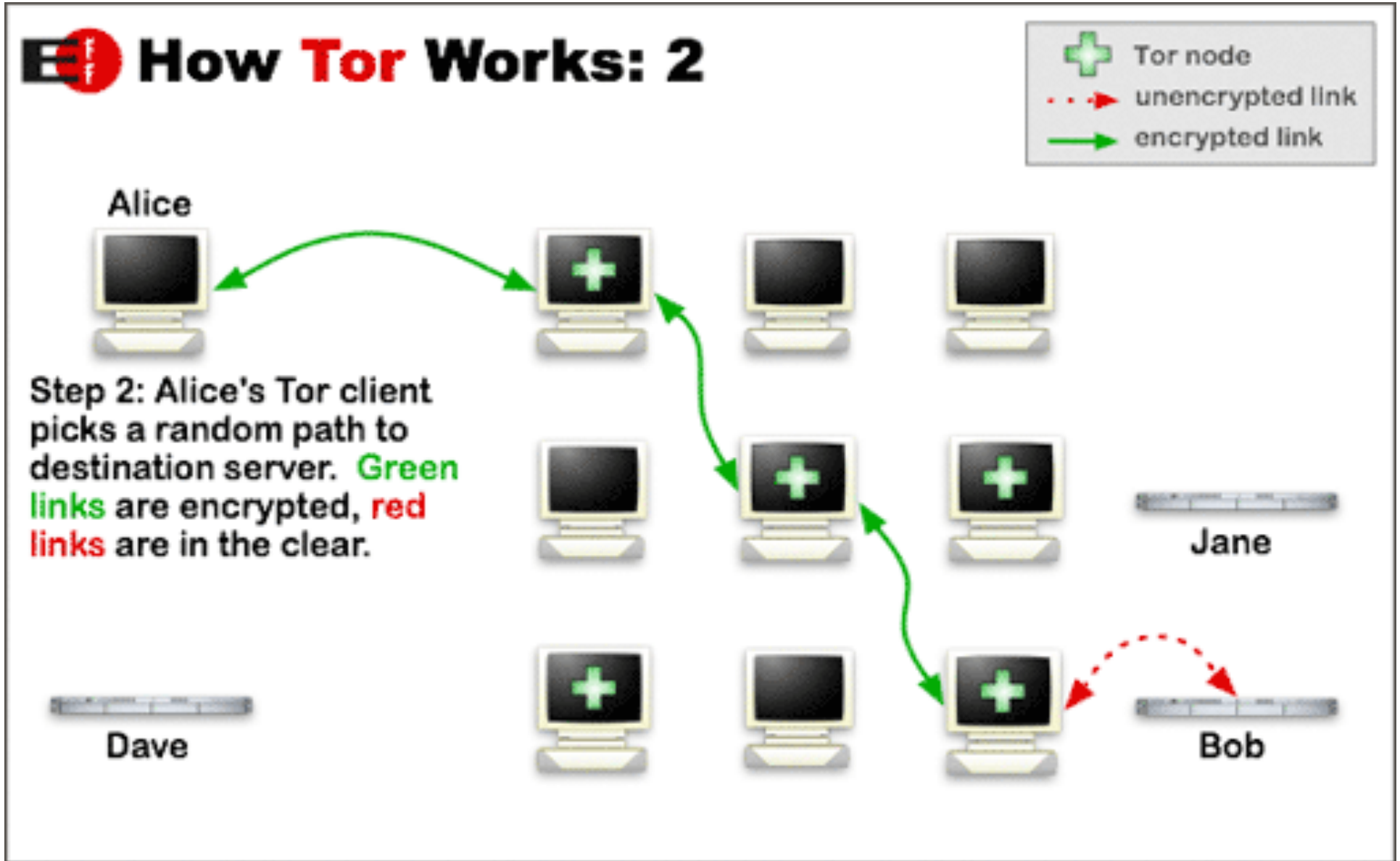


Credits: some slides borrowed from the Tor web site

Tor – Circuit Establishment

- User's software/client incrementally builds a circuit of encrypted connections through relays (onion routers) on the network
  - Each user runs local software called an *onion proxy* to fetch directories and circuits across the network
  - The circuit is extended one hop at a time
  - Entry node -> relay -> exit node
- Each relay along the way knows only the previous and the next relays
- The client negotiates a separate set of encryption keys for each hop along the circuit
  - Each node in the path uses DH key negotiation

Tor - Communication



Diffie-Hellman Key Exchange

Alice

Bob and Alice know and have the following :
$p = 23$ *(a prime number)* $g = 11$ *(a generator)*

Bob

Alice	Bob
Alice chooses a secret random number a = 6	Bob chooses a secret random number b = 5
Alice computes : $A = g^a \bmod p$ $A = 11^6 \bmod 23 = 9$	Bob computes : $B = g^b \bmod p$ $B = 11^5 \bmod 23 = 5$
Alice receives B = 5 from Bob	**Bob receives A = 9 from Alice**
Secret Key = $K = B^a \bmod p$	Secret Key = $K = A^b \bmod p$
$K = 5^6 \bmod 23 = 8$	$K = 9^5 \bmod 23 = 8$

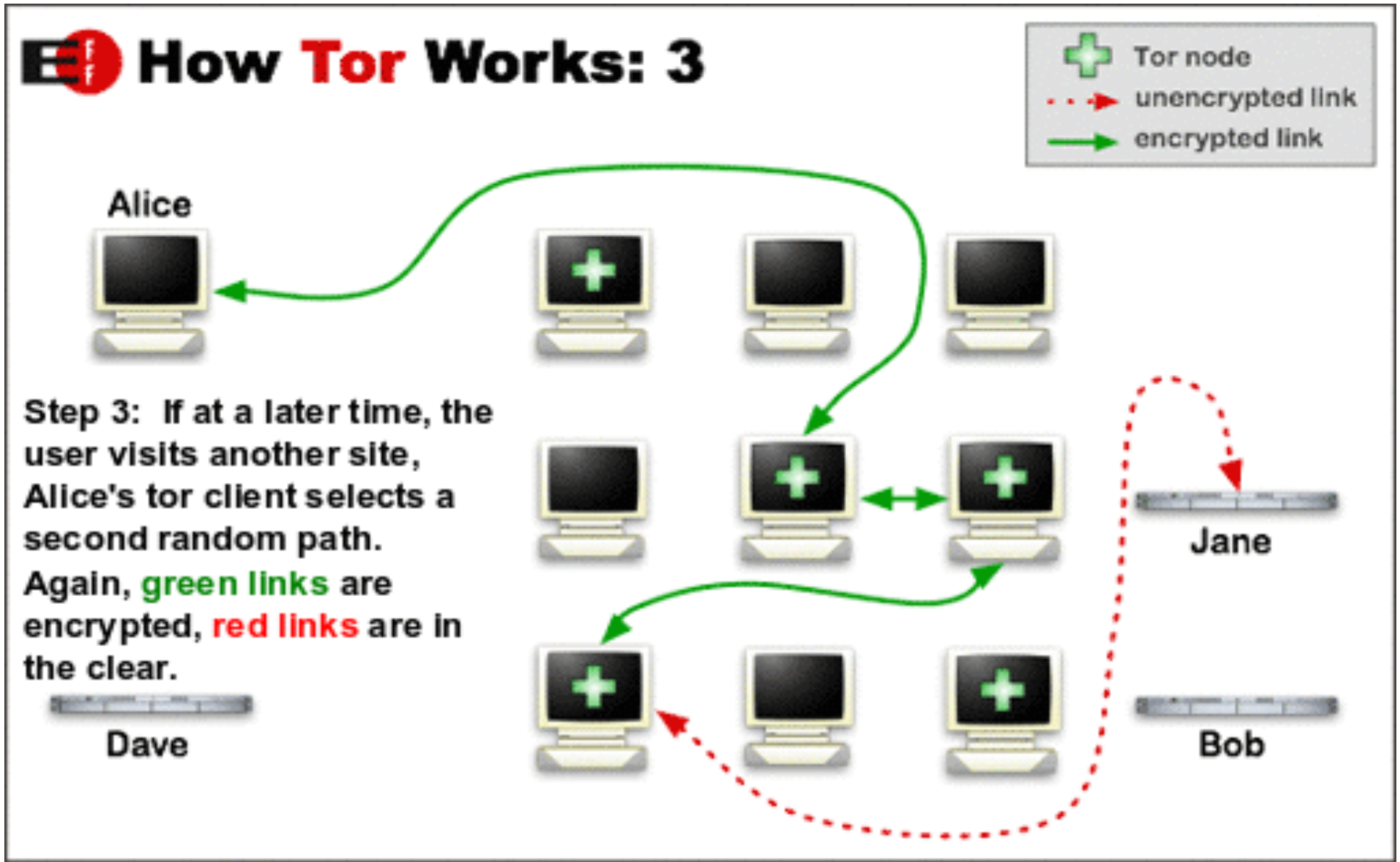
The common secret key is : 8

N.B. We could also have written : $K = g^{ab} \bmod$

Tor - Communication

- Onion proxies accept TCP streams and multiplex them across the circuits
- The onion router on the other side of the circuit connects to the requested destinations and relays data
- Neither an eavesdropper nor a compromised relay can use traffic analysis to link the connection's source and destination
- Uses the same circuit for connections that happen within the same ten minutes or so
  - Later requests are given a new circuit

Tor - New Connection



Tor – Hidden Services

- Sometimes also called “Location-Hidden Services”
- Let users publish websites and other services without revealing the location (IP address) of the site
  - Users can set up a website where people publish material without worrying about censorship
- Hidden services advertise their existence via “introduction points”
  - Know hidden service’s public key (ID) but not its IP (location)
- Using Tor “rendezvous points”, other Tor users can connect to the hidden services, each without knowing the other's network identity

Cryptographic Mechanisms for Privacy Protection

- Anonymous communication
  - Tor
- Anonymous credentials
- Blind signatures
- Secure multiparty computation
  - Garbled circuits
  - Secret sharing
- Deterministic encryption
  - Order-preserving encryption
- Computing on encrypted data
  - Homomorphic encryption
- Oblivious RAM
- Private information retrieval
- Zero-knowledge proofs
- Etc.

Anonymous Credentials (1)

- Allow users to authenticate themselves in a privacy-preserving manner
- Traditional credentials:
  - Alice obtains credentials from an organization
  - At some later point, she proves to the organization (or any other party) that she has been given appropriate credentials
- Anonymous credentials:
  - Alice can do the same without revealing who she is (only that she possesses valid credentials)
  - If she uses her credentials a second time, no one will be able to tell that the two interactions involved the same user

Anonymous Credentials (2)

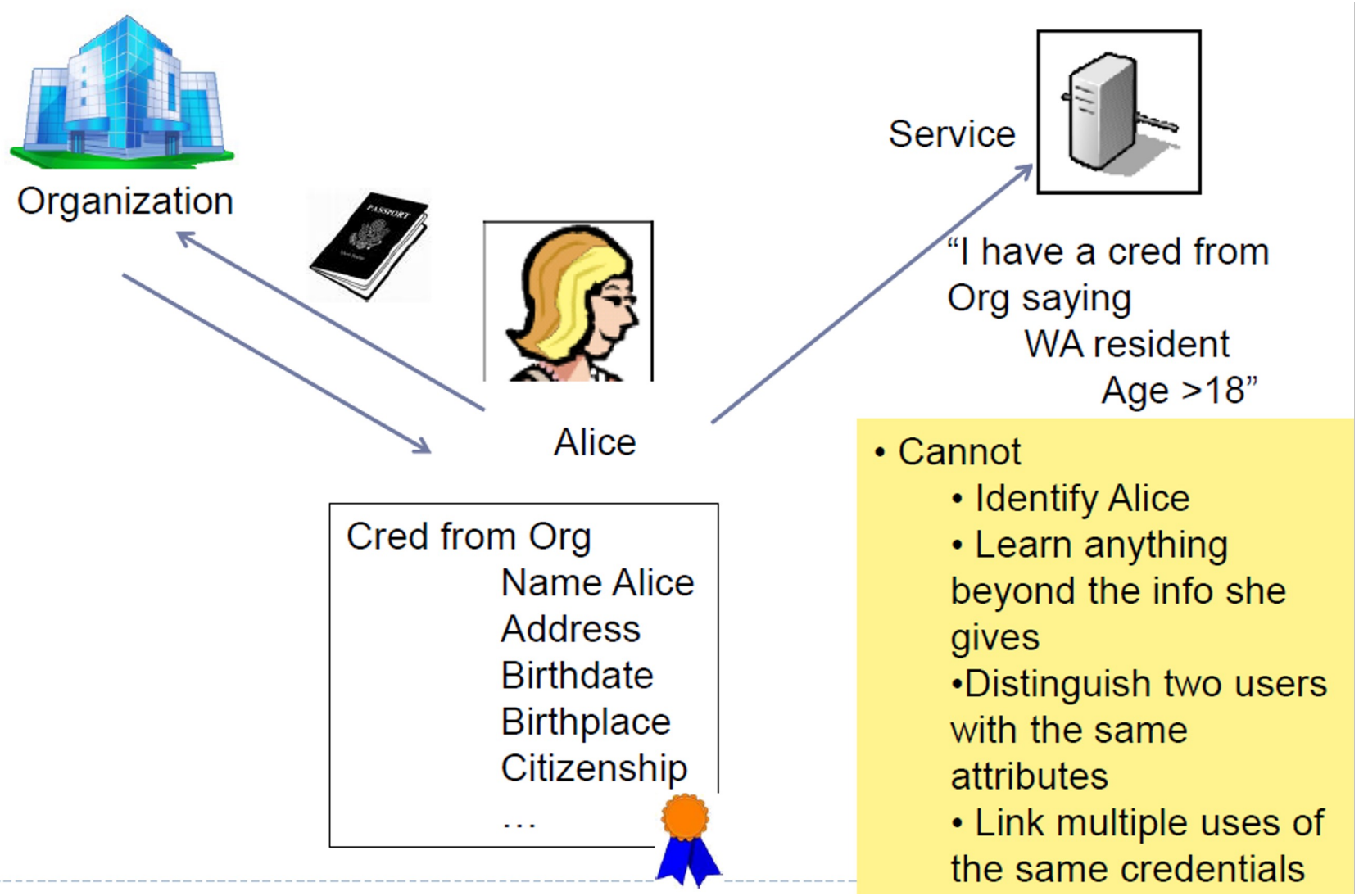


Figure: Melissa Chase – Microsoft Research

Digital Signatures

- A digital signature is used to verify the **authenticity** and **integrity** of a digital message
- It is equivalent of a handwritten signature or a stamped seal, but in digital form
- The purpose of a digital signature is to ensure that the sender of a message is who they claim to be and that the message has not been altered during transmission

Blind Signatures

- Content of a message is blinded before it is signed
- Resulting blind signature can be publicly verified against the original (unblinded) message
- Cryptographic voting systems
  - Authority checks the credentials of the voter to ensure that he is allowed to vote, and that he is not submitting more than one vote
  - Authority does not learn the voter's selections

Oblivious Transfer

- It was shown (by Kilian in 1988) that by using an implementation of oblivious transfer, and no other cryptographic primitive, it is possible to construct any secure computation protocol

Alice

Bob

Input: (m_0, m_1)

Input: $b \in \{0, 1\}$

$$\xrightarrow{(m_0, m_1)}$$

Output: nothing, $\perp$

Output: m_b

Oblivious Transfer

Alice				Bob		
Calculus	Secret	Public		Public	Secret	Calculus
Messages to be sent	m_0, m_1					
Generate RSA key pair and send public portion to Bob	d	N, e	$\Rightarrow$	N, e		Receive public key
Generate two random messages		x_0, x_1	$\Rightarrow$	x_0, x_1		Receive random messages
					k, b	Choose $b \in \{0, 1\}$ and generate random k
		v	$\Leftarrow$	$v = (x_b + k^e) \mod N$		Compute the encryption of k, blind with x_b and send to Alice
One of these will equal k, but Alice does not know which.	$k_0 = (v - x_0)^d \mod N$ $k_1 = (v - x_1)^d \mod N$					
Send both messages to Bob		$m'_0 = m_0 + k_0$ $m'_1 = m_1 + k_1$	$\Rightarrow$	m'_0, m'_1		Receive both messages
					$m_b = m'_b - k$	Bob decrypts the m'_b since he knows which x_b he selected earlier.

https://en.wikipedia.org/wiki/Oblivious_transfer

RSA key observation: $(m^e)^d \equiv m \pmod N$

Oblivious Transfer

- There are simple and efficient protocols for oblivious transfer which are secure only against semi-honest adversaries
- It is more challenging to construct oblivious transfer protocols which are secure against malicious adversaries
  - Can be achieved using zero-knowledge proofs that are used by the receiver

Secure Multiparty Computation

- Alice and Bob compute a function of their private data, without exposing anything about their data besides the result

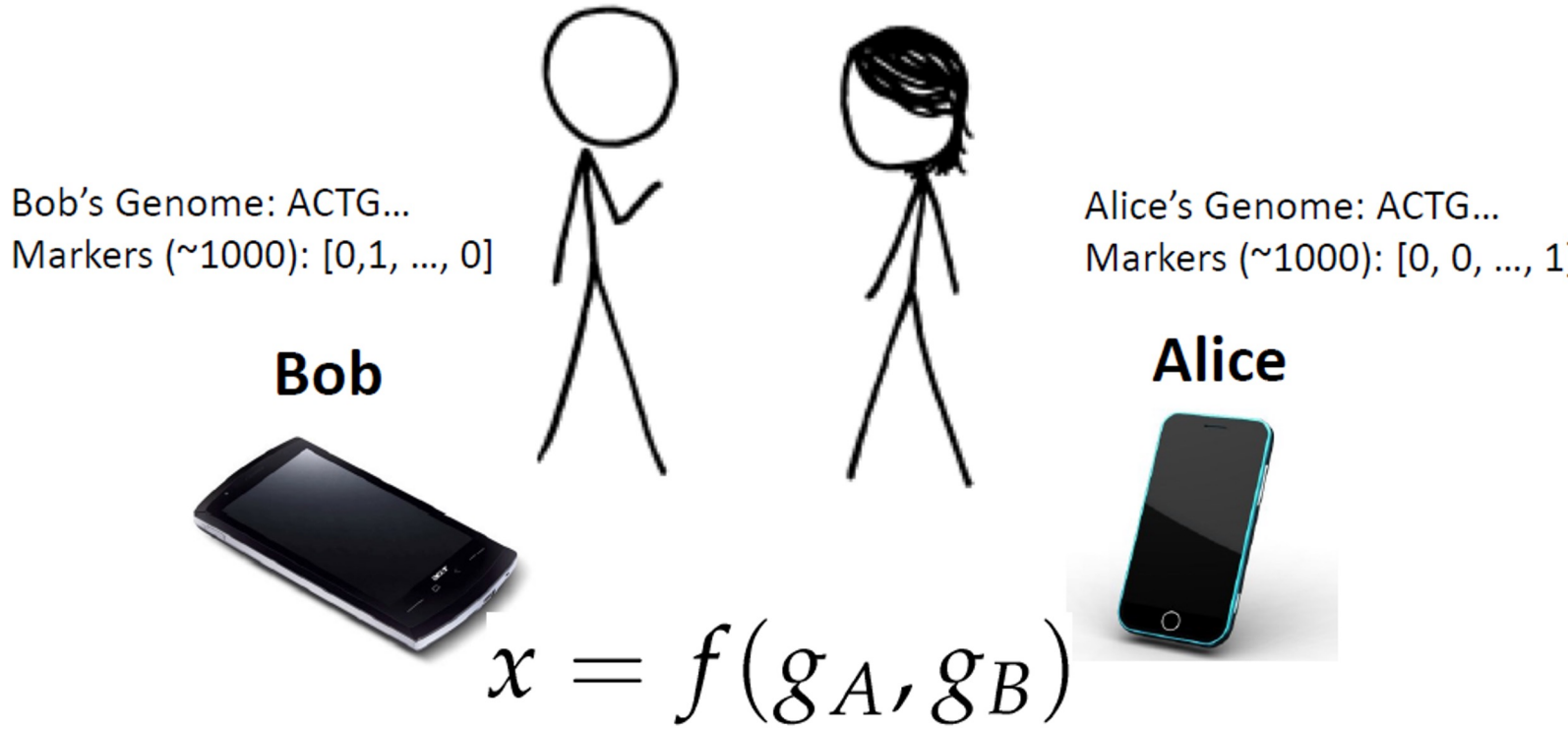
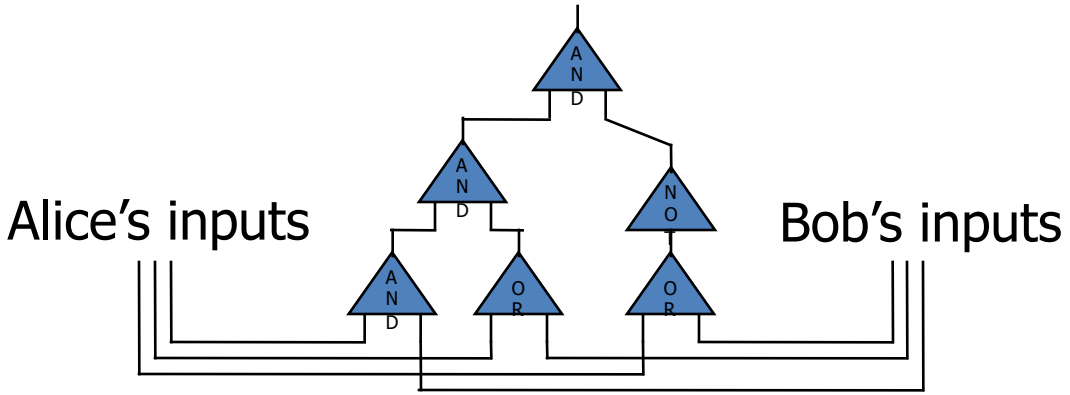


Figure: David Evans et al.

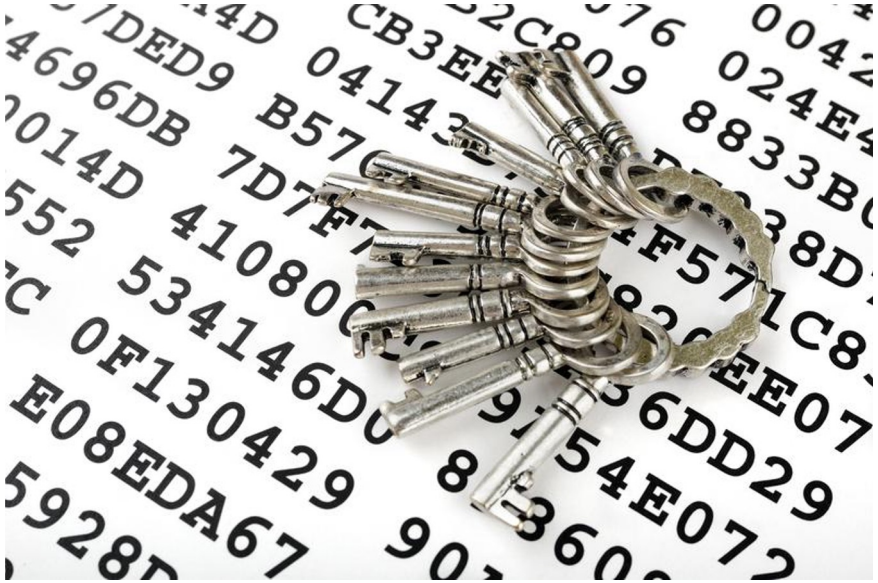
Garbled Circuits

- Bob creates a garbled circuit and sends the circuit to Alice
- Alice evaluates the circuit with her inputs and returns the result to Bob
- The result of the circuit evaluation with Alice's inputs is the output of the function Alice and Bob wish to compute
- Bob does not send his inputs to Alice, instead his inputs are encoded into the garbled circuit such that Alice cannot determine what they are

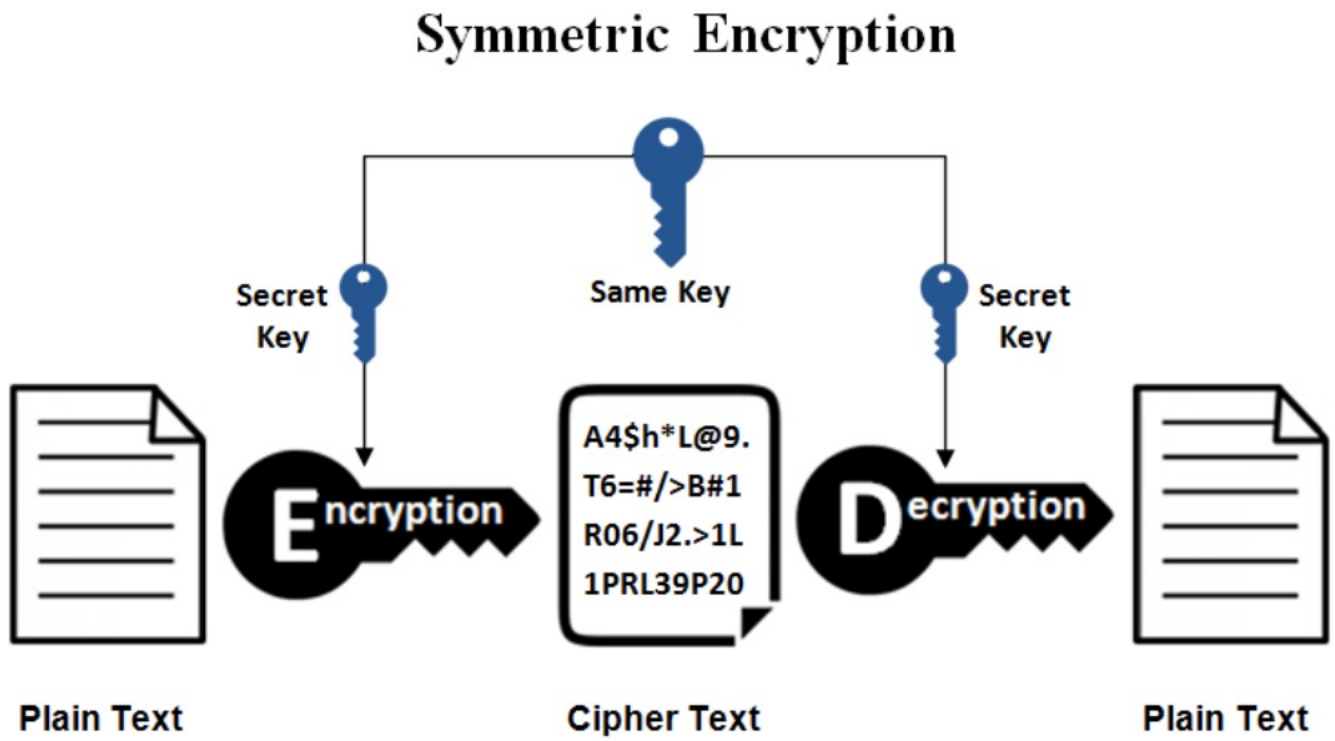


Deterministic Encryption

- Always produces the same ciphertext for a given plaintext and key
  - It is efficient in searching of encrypted data
- Order-preserving encryption
  - M>N → E(M)>E(N)
  - Leaks information to an eavesdropper, who may recognize known ciphertexts
  - Does not guarantee what is known as “semantic security” -Need for **"probabilistic encryption"**

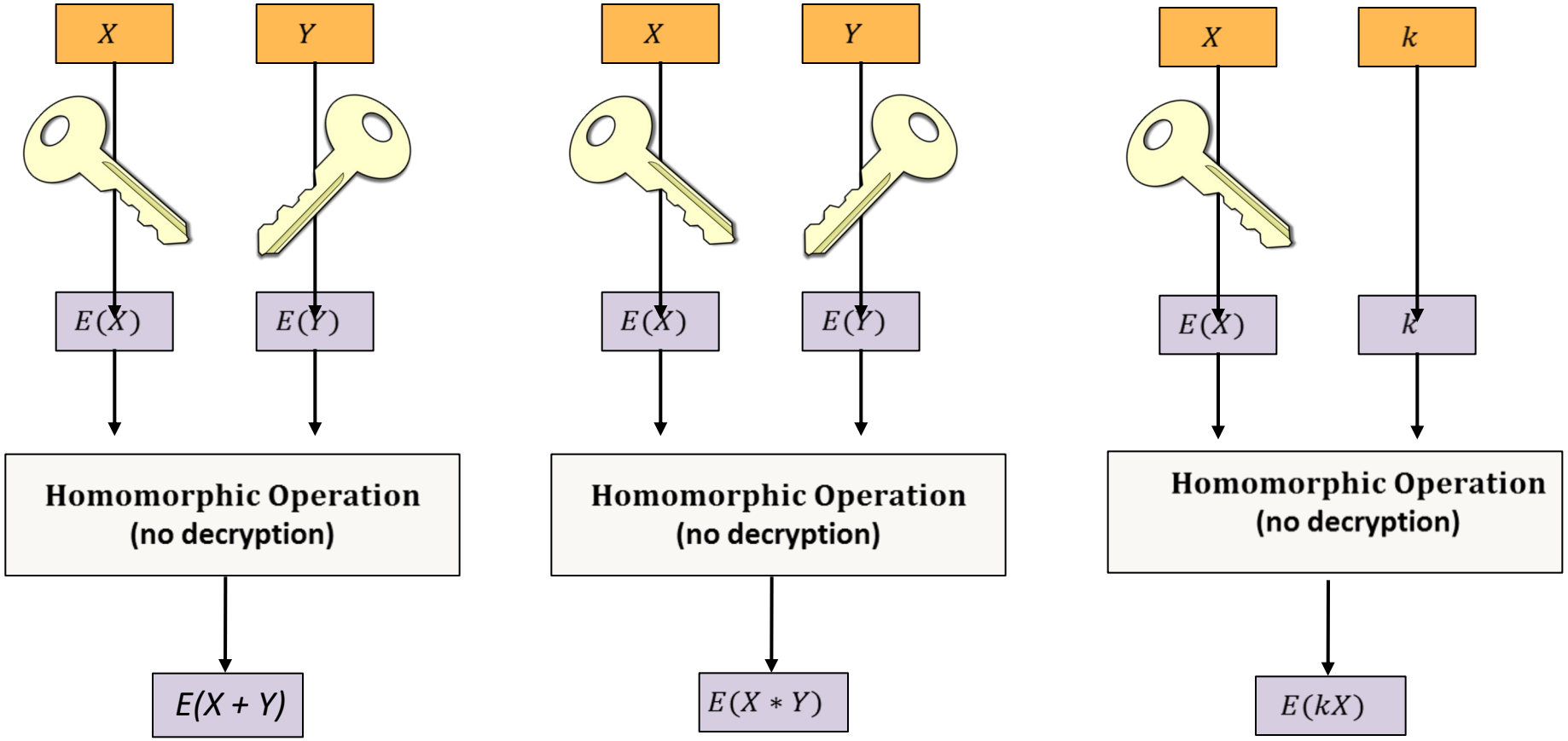


Symmetric Encryption



Homomorphic Encryption

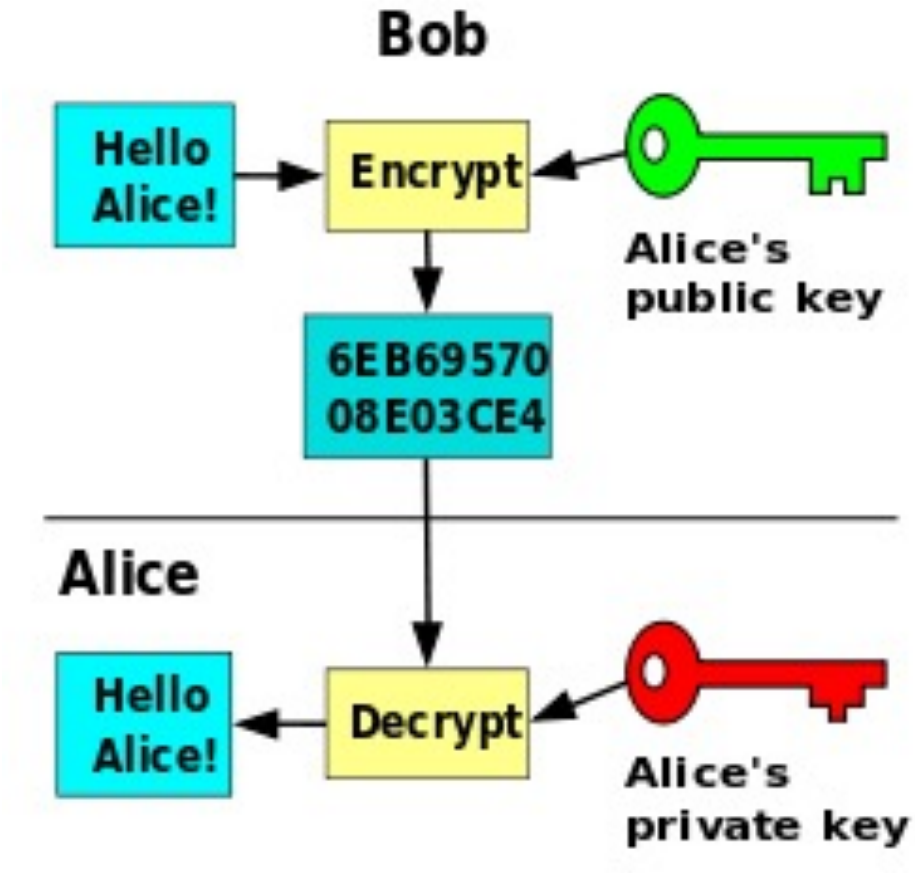
- Allows specific types of computations to be carried out on ciphertext



Benaloh, Paillier

Unpadded RSA, ElGamal

Paillier Cryptosystem



- The public key: (n, g, h = g^x)
- Secret key: x ∈ [1, n^2/2]
- Strong secret:

  Factorization of n = zy
  (z , y are safe primes)

Paillier Cryptosystem
Encryption

- To encrypt a message m ∈ Z_n
  - Select a random r ∈ [1, n/4]
  - Generate the ciphertext pair (C1,C2) such that
  - C1 = g^r mod n^2
  - C2 = h^r(1 + mn) mod n^2

  - [m]=(C1,C2)

The public key: (n, g, h = g^x)
Secret key: x ∈ [1, n^2/2]

Paillier Cryptosystem
Decryption

- The message m can be recovered from [m]=(C1,C2) as follows:
  - m = Delta(C2 /C1^x)
  - Delta(u) = [(u−1) mod n^2]/n
    - For all u ∈ {u < n^2 | u = 1 mod n}

The public key: (n, g, h = g^x)
Secret key: x ∈ [1, n^2/2]

Paillier Cryptosystem

Example

Here are some example values if you want to work through the algorithm:

Key generation

1. Pick $p = 13$ and $q = 17$. (They satisfy the condition.)
2. Compute $n = 221$.
3. Compute $\lambda = 48$.
4. Pick $g = 4886$.
5. Compute $\mu = 159$. (It exists.)

Encryption

1. Set $m_1 = 123$.
2. Pick $r_1 = 666$.
3. Compute $c_1 = 25889 \mod 221^2$.

Decryption

1. Compute $m_{\text{decrypted}} = 123 \mod 221$. (The same as m_1.)

(But beware these numbers are too small to offer any real security and my random values weren't all that random.)

https://blog.openmined.org/the-paillier-cryptosystem/

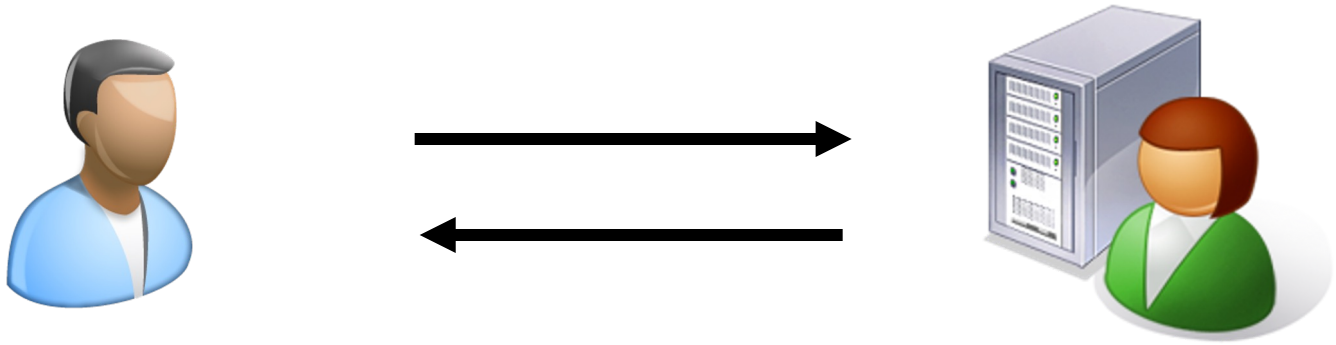
Private Set Intersection (PSI)

Protocol in which two parties jointly compute the intersection of their private input sets

There is no trusted third party!

Private Information Retrieval (PIR)

- Allows a user to retrieve an item from a server in possession of a database without revealing which item is retrieved



User U

wants to retrieve some data from D

Database D

shouldn't learn what U retrieved

IT-PIR vs. cPIR

- **Information Theoretic PIR (IT-PIR)**
  - Non-colluding *L* servers
  - Each server holds a copy of the database
  - **Perfectly secure** if some number of these servers are not colluding

- **Computational PIR (cPIR)**
  - Single database-server
  - Uses cryptographic techniques to encrypt the user's query
  - The security of cPIR relies on the security of the underling encryption
  - Privacy is ensured **only against computationally-bounded attackers**

Oblivious RAM

Same as PIR, but with R/W:

- Client outsources the storage of his data to a cloud
- Client stores only a small amount of data locally
- Client accesses (read/write) his data while hiding the identities of the items being accessed

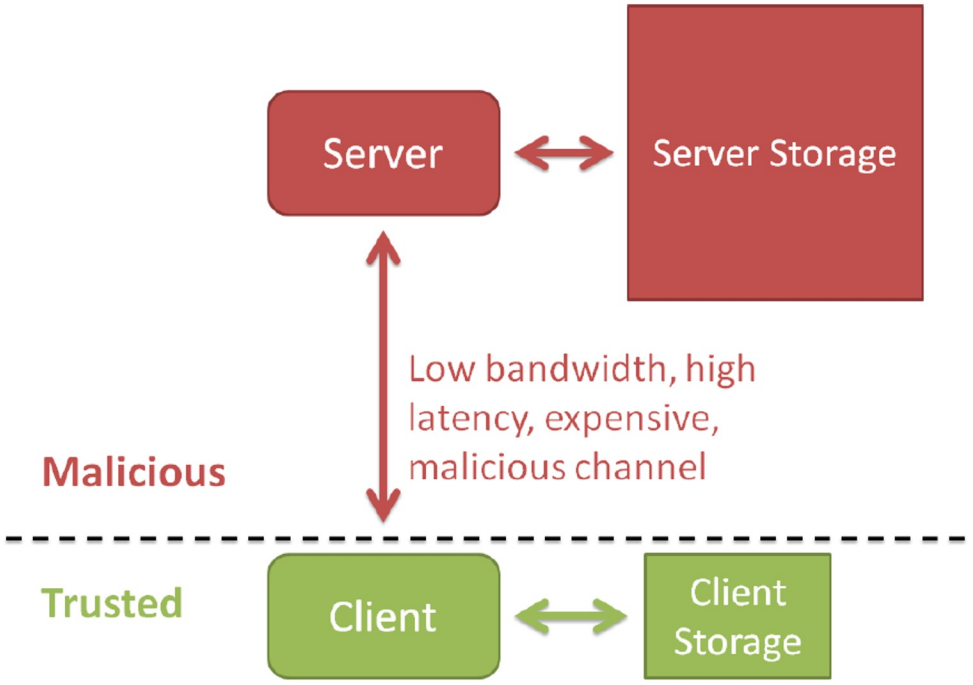
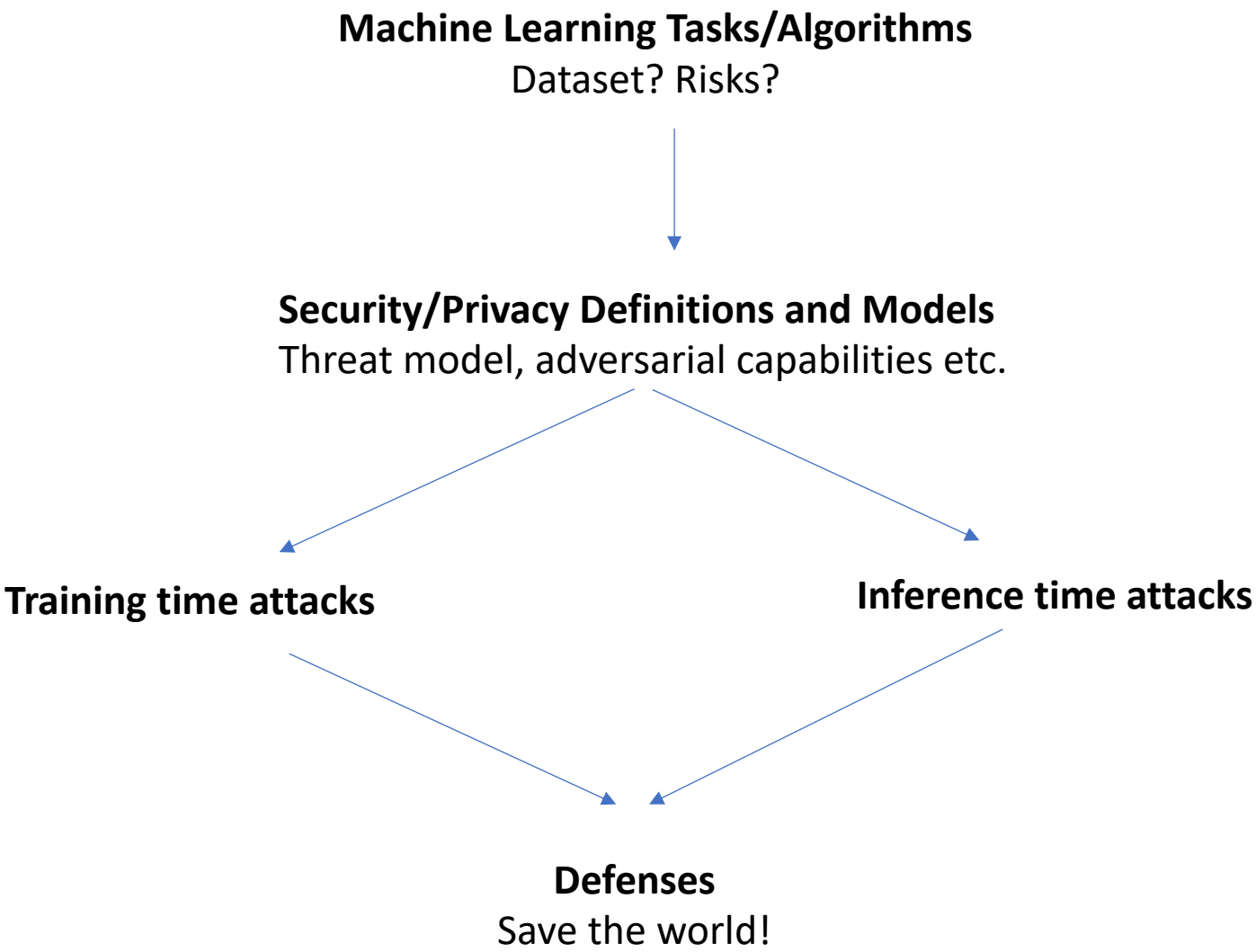


Figure: Emil Stefanov et al.

How about technical approaches to privacy in Machine Learning?


Machine Learning Tasks/Algorithms
Dataset? Risks?
Security/Privacy Definitions and Models
Threat model, adversarial capabilities etc.
Training time attacks
Inference time attacks
Defenses
Save the world!

Attacks

- Poisoning attacks
- Inference/reconstruction attacks
    - Model inversion attacks
    - Property inference attacks

- Membership Inference Attacks

Defenses

- Federated learning (not in vanilla form) with complementary mechanisms
- Differential privacy, e.g., DP-SGD algorithms
- Homomorphic encryption
- Multiparty computation
- Perturbation, randomized query responses
- Adversarial training