

0	Encryption	Anonymity XXX
1	Key Cryptography (Solves Observability but NOT Unlinkability Problem)	Anonymity XX
2	VPN / Proxies	Anonymity X
3	Cascade of Proxies	Anonymity ✓
1	Mix Network (Solves Unlinkability Problem)	Anonymity ✓✓
2	Tor (Improved Mix Network)	Anonymity ✓✓✓

























Key Cryptography & VPN/Proxies

When sending data from one person to another, it is encrypted via public/private keys, although the source and destination are not anonymised and therefore pose the issue of linkability. As an attempt to mitigate this, proxies can be used as a "middle-man", and hence the receiver does not know who the sender is.

SENDER → **PROXY** → **RECEIVER**

- If an attacker sits between the **proxy** and **receiver**, they do not know the source address but they know the destination address.
- If an attacker sits between the **sender** and the **proxy**, they do not know the destination address but they know the sender address.
- If an attacker sits at the **proxy**, they know both the sender and receiver addresses.
- If there is an attacker between **Sender** and **Proxy**, and another attacker between **Proxy** and **Receiver**, the attacker can use statistical analysis / traffic patterns to deduce source and destination.

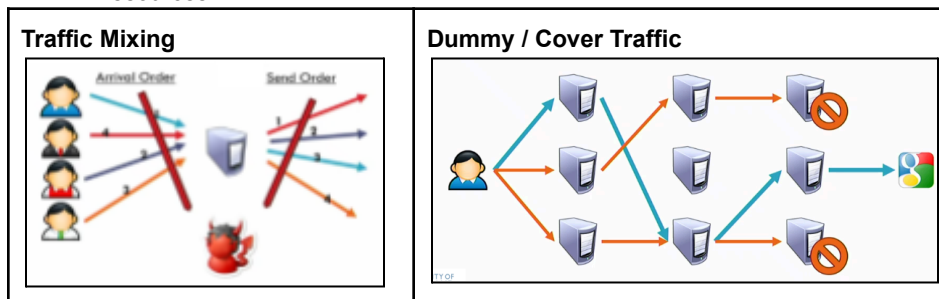
Cascade of Proxies

Mix Network			Tor		
$C = E(K_p, E(K_p, E(K_p, M)))$			$C = E(S_k, E(S_k, E(S_k, M)))$		
	$[K_p, K_p, K_p]$			$[S_k, S_k, S_k]$	
Mix 1	$\langle K_p, K_s \rangle$		Mix 1	$\langle S_k, K_p, K_s \rangle$	
Mix 2	$\langle K_p, K_s \rangle$		Mix 2	$\langle S_k, K_p, K_s \rangle$	
Mix 3	$\langle K_p, K_s \rangle$		Mix 3	$\langle S_k, K_p, K_s \rangle$	
	Message Received			Message Received	
Mix 3	$\langle K_p, K_s \rangle$		Mix 3	$\langle S_k, K_p, K_s \rangle$	
Mix 2	$\langle K_p, K_s \rangle$		Mix 2	$\langle K_p, K_s \rangle$	
Mix 1	$\langle K_p, K_s \rangle$		Mix 1	$\langle S_k, K_p, K_s \rangle$	
	$\langle K_p, K_s \rangle$ $\langle K_p, K_s \rangle$ $\langle K_p, K_s \rangle$			$\langle K_p, K_s \rangle$ $\langle K_p, K_s \rangle$ $\langle K_p, K_s \rangle$	

Mix Networks

An attempt to mitigate statistical/traffic analysis is Mix Networks. Mix Networks are a chain of anonymised proxies called 'Mixes'. Data is encrypted in layers and sent to the next mix and so on until it reaches a destination. This way, at any point between Source → Mix, Mix → Mix, Mix → destination, it will be unlikely for an attacker to identify a source and a destination.

- Each mix can artificially delay the time of sending (Latency Overhead) or randomly shuffle a set of different data packets and send them in an arbitrary order to obfuscate an attacker sitting between mixes. This requires a lot of resources
- Dummy/Cover Traffic can be used where along with the real packet, Randomly generated packets are also sent to dummy destinations to make the perception of real traffic difficult. Requires extensive computational resources



Tor - The Onion Router

Tor is a Mix Network with improvements, and instead of 'Mixes', these proxies are called 'Relays'.

1. Sender encrypts data in layers using private keys, each layer holding the address of the next relay.
2. Each relay decrypts the outer layer using a public key, revealing the next address. Hence each relay only knows the previous relay address and the following relay address.
3. At the final relay, it sends data to the receiver.
4. Receiver encrypts the reply using a public key with the relay's address
5. Each relay passes the data back by adding back their layer.
6. Sender decrypts all layers using their private keys and can view the reply.

Mix Network vs Tor

Mix Network	Tor
Use long-term static key pairs for encryption.	Uses ephemeral session keys for each session.
If a private key is compromised, past communications can be decrypted.	Provides Perfect Forward Secrecy, protecting past sessions even if keys are compromised.

[WEEK 8.2 LECTURE 22]

Tor's Perfect Forward Secrecy (PFS)

Each time there is a new "session" (layer) a new public and private key are negotiated between sender and receiver. Temporary (ephemeral) encryption keys are generated for each session. These keys are used only during that session and discarded afterward. Attackers can compromise new connections going forward, it's just the prior conversations that are safe.

Tor is not Attack-Proof

<ul style="list-style-type: none"> • Source Known • Destination Unknown 	<ul style="list-style-type: none"> • Source Unknown • Destination Unknown 	<ul style="list-style-type: none"> • Source Unknown • Destination Known 	<ul style="list-style-type: none"> • Source Known • Destination Known

Predecessor Attack

With n total relays where m of the are controlled by an attacker:

- m/n chance for an attacker to be in the entry relay.
- $(m-1) / (n-1)$ chance for an attacker to be in the exit relay.
- Hence, $(m/n)^2$ chances for an attacker to be in the circuit.

Tor Bridges

We know that there are 3 types of Tor Relays: **Entry/Guard**, **Middle**, **Exit**. Although Tor Directories are publicly available, governments can still restrict IP addresses of Tor Relays listed in these directories. Tor bridges are just unpublished proxies that are used to connect clients in censored areas to the rest of the Tor network. Bridges alone are insufficient to get around all types of censorship, hence Pluggable Transport Design is used, which disguises traffic to look like other traffic.

Block Chain

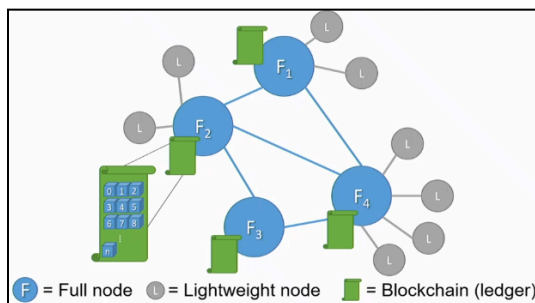
Ledger: a book or other collection of financial accounts.

Blockchain: Blocks of transactions

- PUBLIC: Open to allow anyone to participate (**permissionless blockchain**)
- PRIVATE: Closed and restricts participants (**permissioned blockchain**)

Blockchain does not work without participants (nodes)

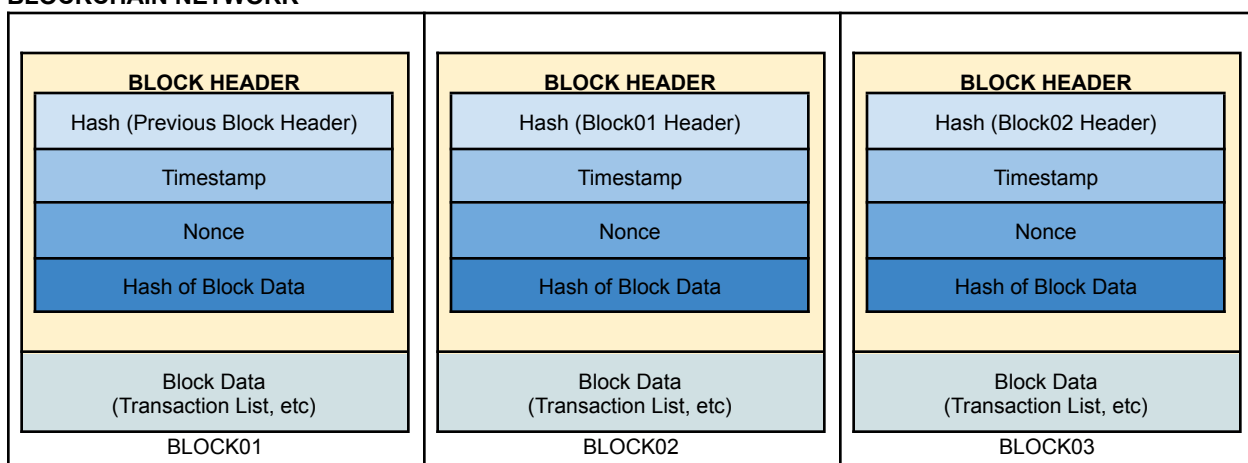
	Full Nodes	Lightweight Nodes	Publishing Nodes
What	Store a complete copy of the entire blockchain.	Keep just a small part and rely on full nodes for information.	Focus on sending and receiving transactions rather than storing the blockchain or validating it.
Why	Verify and validate transactions.	Faster and use less storage, great for mobile devices and casual users.	Facilitate communication between users and the network, helping users publish new transactions.
EG	A librarian who has every book in the library and knows where everything is.	Imagine a student who only has a few reference books but can ask the librarian (full node) for anything.	A newspaper editor who spreads news but doesn't keep a full archive of every article ever printed.



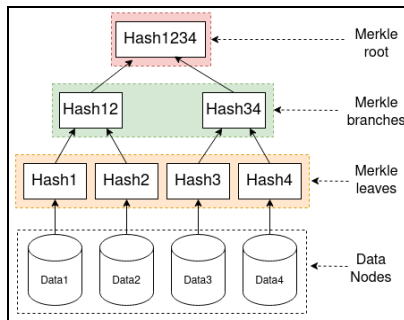
Multiple Full Nodes exist, so in the case that one Full Node goes offline or disconnects, other Full Nodes can still maintain the blockchain.

[WEEK 8.3 LECTURE 23]

BLOCKCHAIN NETWORK



- ★ **Block Data**: Actual data (Health records, or thousands of transactions of cryptocurrency)
- ★ **Timestamp**: The time a publishing node has started its block-creating task. Since blocks are created sequentially, we can deduce that the timestamp BLOCK03 > BLOCK02 > BLOCK01
- ★ **Nonce**: In blockchain, it acts as a mathematical, computationally expensive puzzle that publishing nodes solve
- ★ **Hash of Block Data**: Checks integrity of block data
- ★ **Hash of Block Header**: Provides chain functionality, a link between itself and the previous block. Hence Block02 can never be compromised as its legitimate record is stored in Block03. (Tamper resistance)



Merkle Tree: Hashes every single transaction in cryptocurrencies. This is an example of how cryptocurrencies are hashed in a block of a blockchain using the "Merkle Tree" algorithm, where each transaction (data) is hashed individually, paired and hashed again, and again until we reach a Merkle Root.

Blockchain Consensus Models (General Agreement)

- Proof of Work (PoW)
- Proof of Stake (PoS)
- Proof of Authority (PoA)
- Round Robin

Proof of Work (PoW)

Publishing nodes are incentivised to compete with other miners to publish and create new blocks. Publishing nodes may either be a Full Node or part of a mining pool.

1. New transactions are generated

Alice buys bitcoin, signs the transaction with a private key, and sends it to a full node for verification. Transactions are **NOT confirmed yet**, and can be attached with a **transaction fee** to gain priority for its confirmation.

2. Transactions are stored in mempools

Full Node verifies authenticity of transaction **and stores it in its own mempool** and broadcasts transaction to neighboring Full Nodes so miners can try and create a block, and full nodes **can remove this transaction from its mempool**.

3. Miners compete to create a block

Each miner independently groups transactions to publish valid blocks. Their incentive is a newly mined coin plus the transaction fees. Blocks are mined via solving **computationally intensive tasks**.

4. A valid block is published to the blockchain

Re-checks the Nonce and if everything is good, the blockchain is updated and continues to expand.

Miner Block Reward = newly mined coin + transaction fee

1 BTC is worth USD60,000, after every 210,000 blocks, the reward halves, hence 1 BTC will be worth USD30,000 in 4 years.

Miner Conflicts

Two miners solve a puzzle at the same time, both will broadcast their solutions to neighboring nodes, once validated and published to the blockchain, one block usually finds the chain faster and is successfully added, while the other block becomes Stale and essentially disregarded.

Block creation & Difficulty

BC tries to publish a block every 10 minutes, the difficulty is adjusted every 2016 blocks (2 weeks). So 1 hour = 6 blocks, 1 day = 144 blocks, 1 week = 1008, 2 week = 2016

BC Mathematical Challenge

Hashcash was the original puzzle, but increasing/decreasing x leading 0's by 1 doubles or halves the difficulty, a limited difficulty change, hence why BC has changed their puzzle to a 'Target Hash' where you have to find a Nonce less than x which allows fine-tuning difficulty.

EG: Current target hash is 000101 (5d) and it takes 11 minutes to mine, hence we need to decrease difficulty by 1 minute.

Traditional Hashcash Approach	New Target Hash Approach
<p>Instead of 3 leading 0's we will use 2 leading 0's. Hence 000xxx to 00xxxx</p> <p>Possible answers for 000xxx (2^3) = 8 000, 001, 010, 100, 011, 101, 110, 111</p> <p>Possible answers for 00xxxx (2^4) = 16 0000, 0001, 0011, 0111, 1111, 0010, 0101, etc</p> <p>So the number of valid answers is halved, and hence the time to create a block is also halved, not a fair adjustment.</p>	<p>Instead of the target hash being less than 000101, it needs to be less than 000110.</p> <p>Hence difficulty will be decreased as it was initially less than 5, but now has to be less than 6. Hence, time to create a block has decreased by 1.</p>

Disadvantages:

- Computationally intensive
- High energy consumption (Bitcoin uses 4x more energy than all of NZ)
- Hardware arms race

Attack Vectors

Race Attacks	<p>Mallory has 1BTC and makes two transactions:</p> <ol style="list-style-type: none"> 1. Purchases a product for 1 BTC 2. Transfers 1 BTC to another wallet <p>Neither transactions are not confirmed yet, but the merchant delivers the product and eventually it is Transaction 2 that goes through, and hence the merchant is not paid.</p> <p>Countermeasure:</p> <ul style="list-style-type: none"> - Merchant has to wait until a transaction is confirmed.
Finney Attacks	<p>Mallory has 1BTC and makes a transaction:</p> <ol style="list-style-type: none"> 1. Transfers 1 BTC to another wallet, manages to solve the Nonce and creates a valid block, but keeps the block to herself (does not broadcast) 2. Mallory then purchases something for 1 BTC and the merchant delivers the product without it being confirmed. Mallory then broadcasts the block in (1) <p>Mallory gets the product for free. Similar to Race but time is an important component, and Mallory is the attacker.</p> <p>Countermeasure:</p> <ul style="list-style-type: none"> - Merchant can wait until the transaction creates a block, and waits for 5-6 blocks after this block to ensure this transaction cannot be reversed.
51% Attack / Majority Attack	<p>Requires more than 50% of the entire hashing power, takes control of the blockchain network by outpacing other miners.</p> <ol style="list-style-type: none"> 1. Get enough miners to more than 50% the hash rate 2. Make a huge transaction and pocket the goods/services for that transaction 3. Use hash power to shadow mine starting from the block prior to your transaction 4. With 51% current hash power produce the longest chain with the most work that replaces all the blocks starting from 1 before your big transaction 5. Release this new chain which then replaces the chain everyone else was using and undoing all transactions 6. Chaos ensues and you still have the bitcoin from the large transaction plus the goods/services from spending that bitcoin

Proof of Stake (PoS)

Proof of Stake is a system where the creator of the next block is chosen based on the amount of cryptocurrency they hold. This means that the more cryptocurrency a person holds, the more likely they are to be chosen to add the next block to the blockchain. Miners from PoW compete to publish the next block but in PoS, only one validator is responsible to publish the next block, hence it is more energy efficient and does not require expensive hardware. Also, validators that misbehave such as not publishing a block within a certain time, double spend, or invalid blocks, they lose their 'Stake' assets.

Round Robin

More suited for private permissioned blockchain, where nodes know and trust the identities of publishers.

Denial of Service Attacks

3 Types of DoS

Network Bandwidth	<p>Target has a lower channel capacity connected to the internet, They have 1GB p/s but attacker has 40GB p/s. So they have no choice but to drop some incoming packets. For example, an organization hosts the Olympics with legitimate users consuming streams. DoS overwhelms this link, denying legitimate users or degrading the quality of olympic streams. For example, watching in 480p instead of 4K.</p>	<p>Exploiting service with not enough bandwidth</p> <ul style="list-style-type: none"> - Flooding ping commands - Source Address Spoofing - Backscatter Traffic
System Resources	<p>Aims to overwhelm network handling software. Packet structure triggers a bug, crashing the system, and communication becomes unavailable, this attack is an example of a "Poison Packet Attack".</p>	<p>Hiding malicious software in packets</p> <ul style="list-style-type: none"> - Syn Spoofing Attack
Application Resources	<p>Targets application resources. Attackers target specific applications such as a website or database by sending a lot of requests.</p>	<p>Flooding large numbers of requests</p> <ul style="list-style-type: none"> - Distributed DoS - SIP Flood Attacks - HTTP Flood Attack - Slowloris

DoS Flooding Attacks

- ★ **Flooding Ping Commands:** Attacker sends millions of packets to the target by exceeding their capacity. Although, sending ICMP echo requests means the source is identifiable and the target knows who the attacker is. Also, sending out a million packets means we receive a million packets back. Hence, the attacker is negatively impacted as well.
 - Attacker capacity > Target capacity
 - Potential customers of Target cannot consume their service
 - ★ **Source Address Spoofing:** Forges source address making it harder to identify a hacker and target sends replies to the Spoofed address to avoid the problem in the 'Flooding Ping Commands' attack. The spoofed address might even send back replied, further flooding the target.
 - Random Spoofing: Address chosen is random (Real or Fake)
 - Subnet Spoofing: Address is chosen randomly but within same subnet as target system
 - Fixed Spoofing: A fixed spoofed source address
 - Possible countermeasure: Identify the flow of packets by tracing way back. However this includes incorporation of multiple engineers which is time consuming, difficult to carry out, and some services delete log-history every few days
 - ★ **Backscatter Traffic:** Side effect of source address spoofing. Backscatter traffic refers to the ping replies from Spoofed IP Addresses.
 - Honeynet took a bunch of unused IP Addresses and advertised the path to these addresses, monitored these addresses and collected details on the packets sent to them. Since there were no real systems at these addresses, there should be no traffic, but there were packets directed to these unused addresses, and hence network attacks were discovered.
 - ★ **SYN Spoofing Attack:** Overflows targets the TCP connection request table. Attacker tries to fill up the request table space. Hence, legitimate users cannot fit a request into a TCP table and cannot make connections.
 - Table request capacity is usually quite small because it is developed under the assumption that the queue is emptied quickly enough.
 - 3-Way Handshake:
 - Computer A sends a SYN packet to Computer B saying "Hey, I want to SYNchronize and talk with you!"
 - Computer B then replies to Computer A with a SYN-ACK packet that says "I ACKnowledge you want to SYNchronize with me"
 - Computer A then replies with an ACK that is effectively "I ACKnowledge that you ACKnowledge that I want to SYNchronize with you"
 - Attacker generates TCP SYN packets with spoofed IP addresses. The target records in the table and sends SYN-ACK packets to spoofed addresses. If a spoofed IP has a system, the system sends RST to terminate the connection and if IP has no system, there is no response and the SYN-ACK is resent several times (Ideally what Attacker wants). The table entry is removed from the connection table eventually.
 - ★ **Distributed DoS Attack:** Attacker exploits vulnerabilities in OS or install malware on systems, using multiple systems at once. The attacker is the 'Master' and compromised systems are called 'zombies', a large collection of zombies is called a 'botnet', where zombies repeat scanning process and self-propagate.
 - ★ **SIP Flood Attacks:** Sending an invite message that establishes sessions between user agents, these invites consume considerable resources.
 - ★ **HTTP Flood Attacks:** Bombarding a web server with HTTP requests to consume considerable resources, for example, downloading a large file from the target server. This attack has a variant where bots recursively browse all HTTP links.
 - ★ **Slowloris:** Web Servers can only have a limited number of connections, attacker opens multiple connections to the target system with partial HTTP request headers, sends incomplete requests to prevent connection timeouts, attacker periodically sends partial headers. Countermeasures include: Rate limit, timeouts, and delayed binding.
- [WEEK 9.3 LECTURE 26]
- ★ **Reflection Attacks:** Use intermediary servers (reflectors) that attack on attackers' behalf. In DDoS we have zombies but in this attack, we have reflectors, they make requests where they spoof the IP as the target, make requests, and the server responds to the spoofed IP.
 - **Variant: Amplification Attack:** Generates a large number of response packets to target. Countermeasure: Block broadcast packets coming from outside the network, or limit outside ping commands.
 - **DNS Amplification Attacks:** Sends DNS requests to overwhelm target

DoS Countermeasures

Commercial solutions available:

- Intrusion detection System (IDS)
- Firewalls
- Security enhanced routers

Expensive solutions:

- Allocate excess bandwidth
- Use multiple servers to distribute load (load balancer)
- In-house vs Outsourced

Four lines of Defense:

- 1) Attack prevention and preemption (Before attack)
- 2) Attack detection and filtering (During attack)
- 3) Attack source traceback and identification (During and after attack)
- 4) Attack Reaction (After an attack)

DoS Attack Prevention

- Block spoofed source addresses on routers as close to source as possible, although this slows down routers due to IP-Checking process, hence there will be an extra delay added, even though it significantly decreases attacks using spoofed addresses.
- Filters may be used to ensure the claimed source address is the correct one
- ICMP floods or UDP floods can be throttled by rate limiting
- SYN Cookie against SYN Spoofing Attack. In the TCP request table, we can cryptographically encrypt a server's initial sequence number. When a server sends a SYN, it is sent with a cookie, and cookie is sent back to authenticate the SYN-ACK. Server does not consume extra memory but takes computational resources (cookie and encryption) and it blocks the use of some TCP extensions such as large windows.
- For amplification attacks, we can block IP directed broadcasts, blocking suspicious services and combinations, manage application attacks with some graphical puzzle (captcha), and use mirrored and replicated servers when high-performance and reliability is required.

Responding to DoS Attacks

- Identify type of attack
 - Capture and analyse packets
 - Design filters to block attack traffic upstream
 - Or identify and correct system/application bug
- Have ISP trace packet flow back to source
 - May be difficult and time consuming
 - Necessary if planning legal action
- Implement contingency plan
 - Switch to alternate backup servers
 - Commission new servers at a new site with new addresses
- Update incident response plan
 - Analyse the attack and the response for future handling