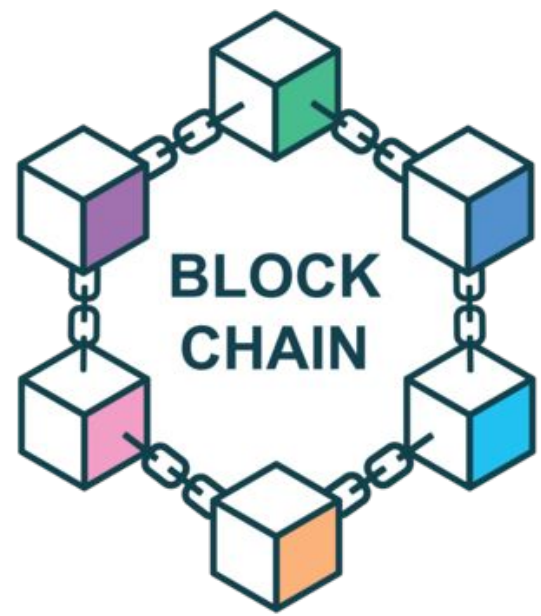


Blockchain Security ISSUES

UNIT 1

Name: Divesh Jadhvani

Goal : Understanding the security issues in blockchain



Pre-Lecture Example: The Virtual Store and the Fake Shopper

Scenario Overview:

- **Virtual Store:** A digital store where people use **CryptoCoins** to buy products.
- **Blockchain:** A secure digital ledger that records transactions and prevents manipulation.

The Problem:

- **Fake Shopper:** Alice tries to spend the same **CryptoCoins** twice.
 - Buys a **T-shirt** and then tries to buy **shoes** with the same CryptoCoins.

Blockchain Solution:

- **Blockchain** prevents double-spending by checking the transaction history.
 - Ensures that once CryptoCoins are spent, they cannot be used again.

Attack Example:

- **51% Attack:** Hackers take control of the ledger and change past transactions, allowing double-spending.



Agenda

Introduction to Blockchain Security

Common Blockchain Attacks

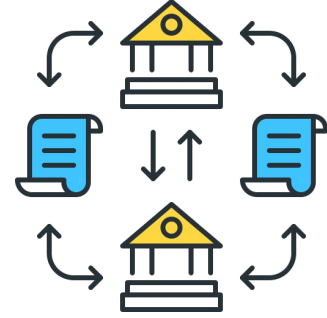
- 51% Attack
- Sybil Attack
- Eclipse Attack
- Double-Spending Attack

Case Studies

Defense Mechanisms



Importance of Security in Blockchain



Why is security important in blockchain?

- Blockchain is decentralized: no central authority (like a bank) to oversee transactions.
- Security prevents fraud, theft, and manipulation of valuable assets like cryptocurrencies.
- Without security, transactions are at risk, making blockchain unreliable.

Real-life example:

- Think of blockchain like a digital locker for valuable items. If the locker isn't secure, thieves can steal your valuables.

Blockchain Security: Not 100% Foolproof

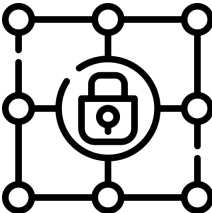


How blockchain is secure, but not 100% foolproof

- Blockchain uses cryptography to secure transactions and prevent tampering.
- Still, hackers may find ways to exploit weaknesses in the system, like bugs or flaws in software.

Real-life analogy:

- Imagine a vault with a strong lock (blockchain's security), but if a thief knows the lock's code or has the key (software vulnerability), they can break in.



Examples of Blockchain Security Incidents

Mt. Gox Hack (2014): 850,000 Bitcoins stolen from an exchange.

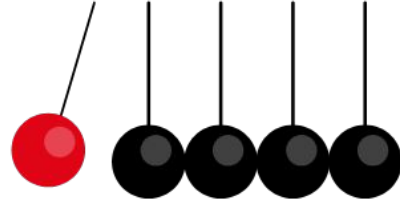
Poly Network Hack (2021): \$600 million stolen but returned by the hacker.

Lesson:

- Even though the blockchain itself is secure, the platforms and exchanges built on top of it can have weak spots.



51% Attack



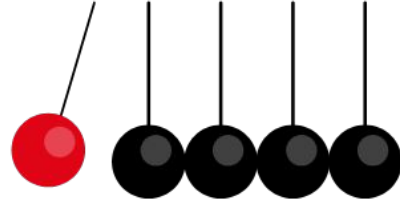
What is a 51% Attack?

- A group of miners controls more than 50% of the blockchain's mining power.
- They can manipulate the blockchain by reversing transactions and double-spending coins.

Real-life example:

- It's like one person having most of the votes in an election. They can change the results to their favor by controlling the majority.

How 51% Attack Works

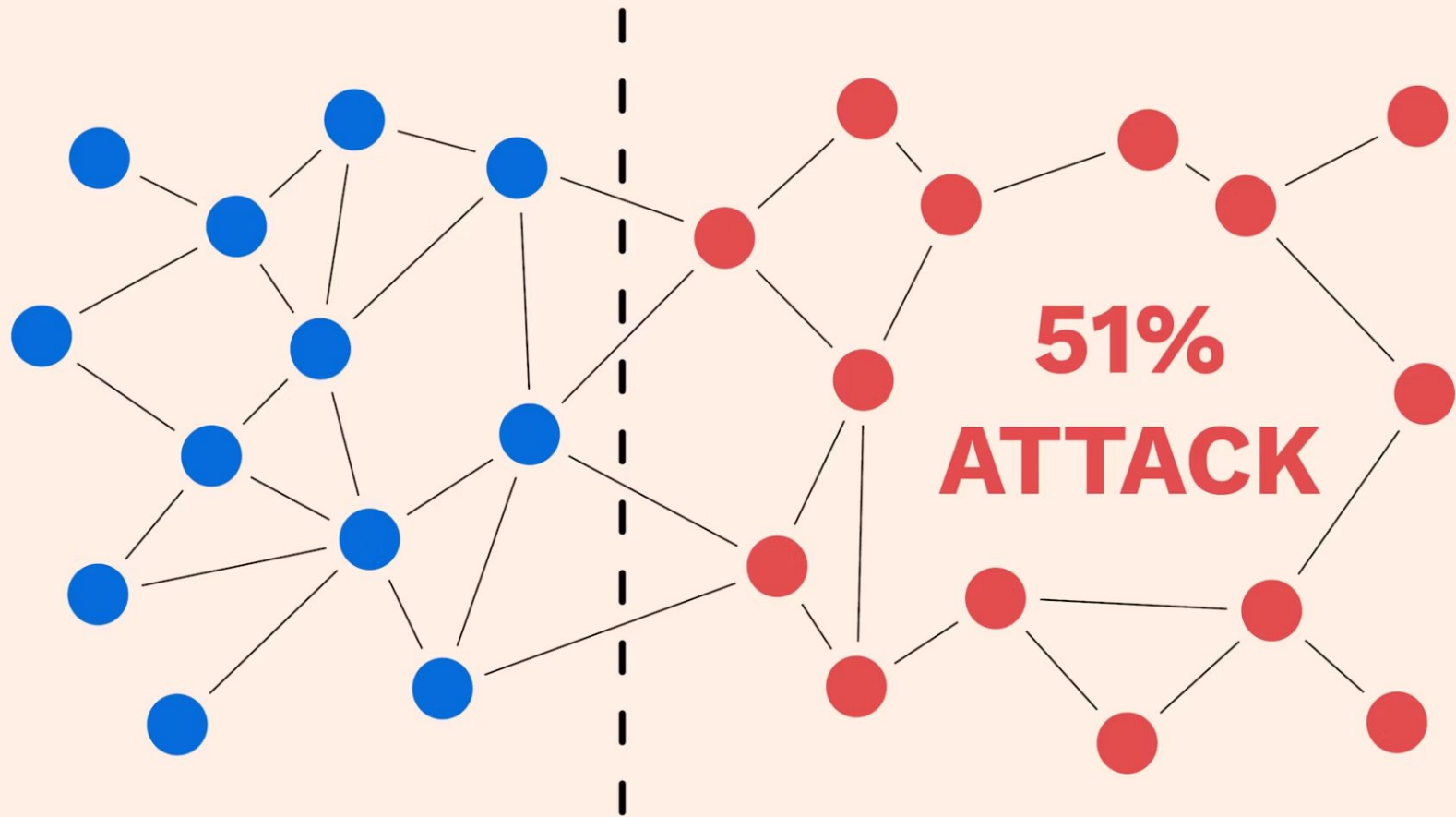


The Attack Process:

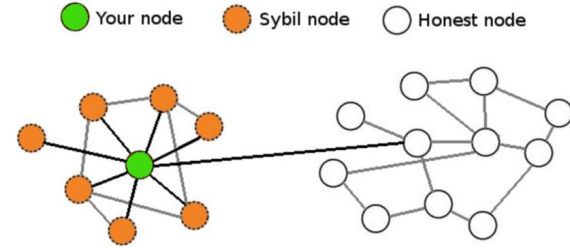
- The attacker gains majority mining power.
- They alter transactions (reversing them) and can double-spend cryptocurrency.
- This undermines the integrity of the blockchain.

Real-life analogy:

- Imagine one student in a group project controlling most of the decisions. They can change past decisions to benefit themselves.



Sybil Attack

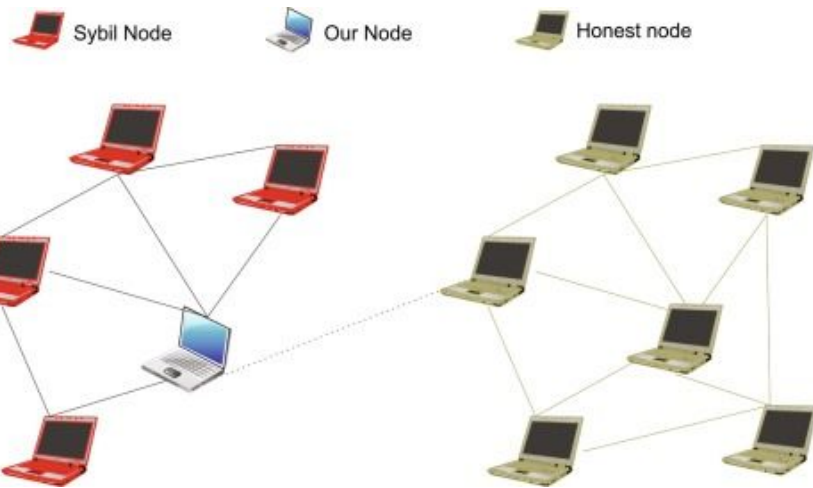


What is a Sybil Attack?

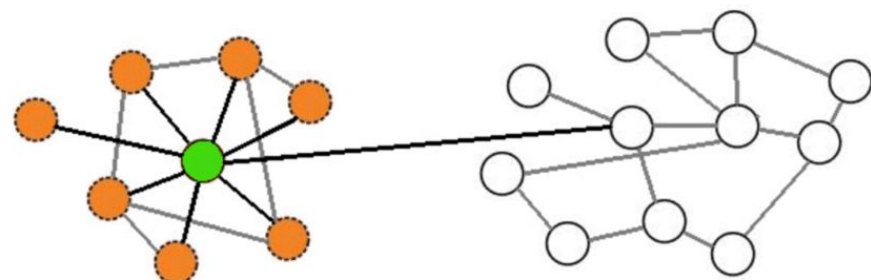
- A single attacker creates multiple fake identities (or nodes) to manipulate the blockchain network.
- This allows them to take control and disrupt the system by flooding the network with fake data.

Real-life example:

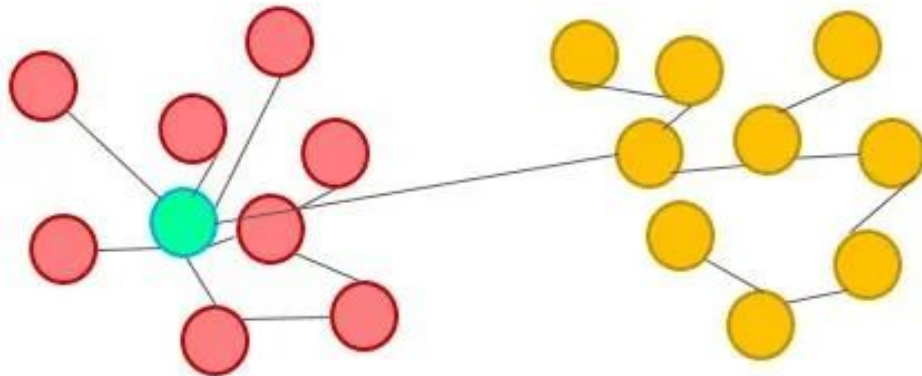
- In an online voting system, one person creates hundreds of fake accounts to cast multiple votes, unfairly deciding the result.



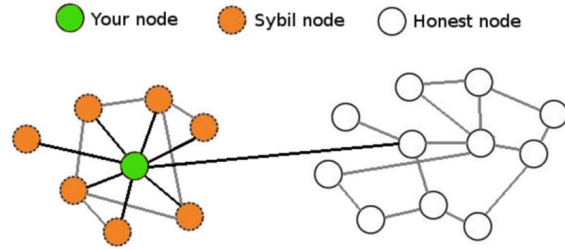
● Your node ● Sybil node ○ Honest node



● Sybil node ● Your node ● Honest node



How Sybil Attack Works



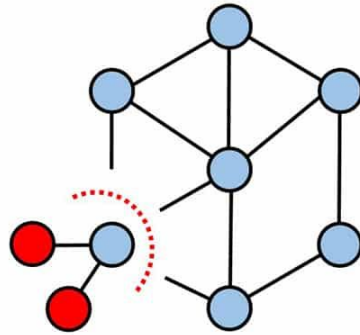
The Attack Process:

- Fake nodes (created by the attacker) overwhelm the blockchain.
- These fake identities can interfere with verifying transactions and lead to manipulation of the system.

Real-life analogy:

- Imagine a school election where one student makes multiple fake accounts and votes several times, changing the result.

Eclipse Attack



What is an Eclipse Attack?

- An attacker isolates a specific node from the rest of the network.
- This isolated node is fed false or manipulated information, which can lead to bad decisions or wrong data being used.

Real-life example:

- It's like blocking a person from receiving messages in a group chat. The attacker can send fake messages, and the isolated person won't know.

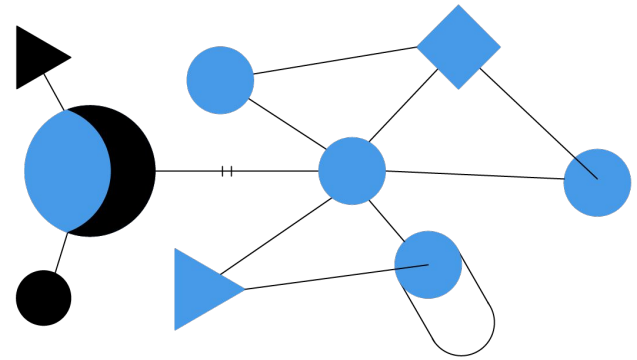
How Eclipse Attack Works

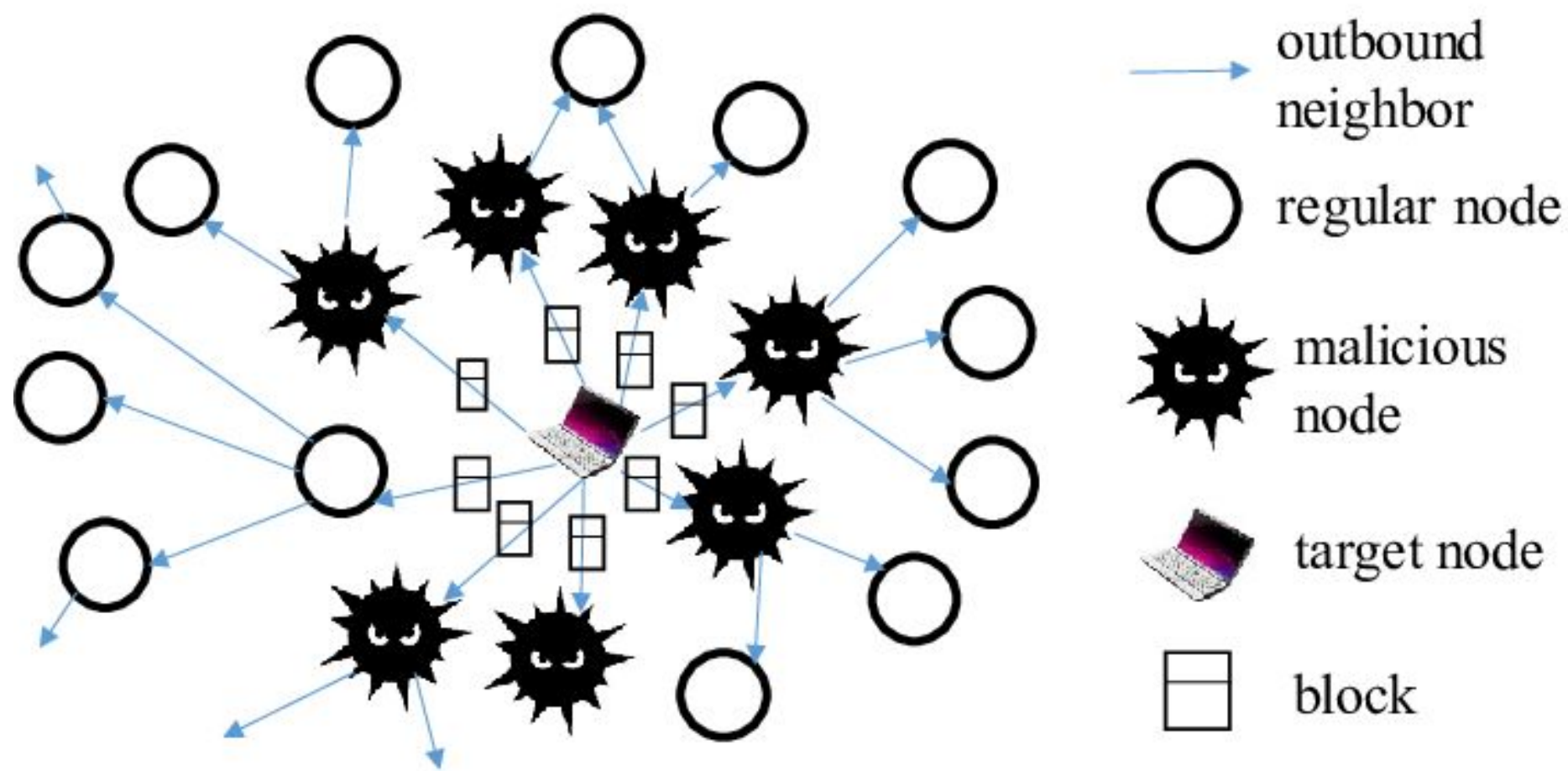
The Attack Process:

- Attacker isolates the target node.
- The isolated node is now only receiving false data from the attacker.
- This can result in incorrect actions being taken by the isolated node.

Real-life analogy:

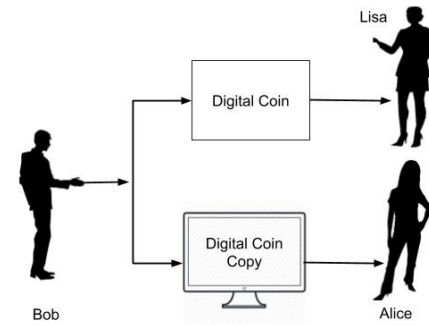
- Think of being in a meeting and one person is isolated from the rest of the group, receiving only one-sided information. They might make decisions based on incomplete or wrong data.





Double-Spending Attack

What is a Double-Spending Attack?



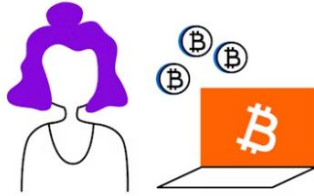
- An attacker tries to spend the same cryptocurrency more than once.
- Blockchain prevents this by keeping a public ledger and confirming every transaction.

Real-life example:

- It's like trying to buy two items at different stores with the same \$10 bill. Blockchain ensures you can only spend it once

What is Double Spending

and why is it such a problem?



Alice

Without exception, all Bitcoin transactions are included in a block of transactions. Each block has a timestamp with encoded information that makes it more difficult to manipulate the blockchain.



Katy

Double spending is a type of deceit where the same money is promised to two parties but only delivered to one.



John

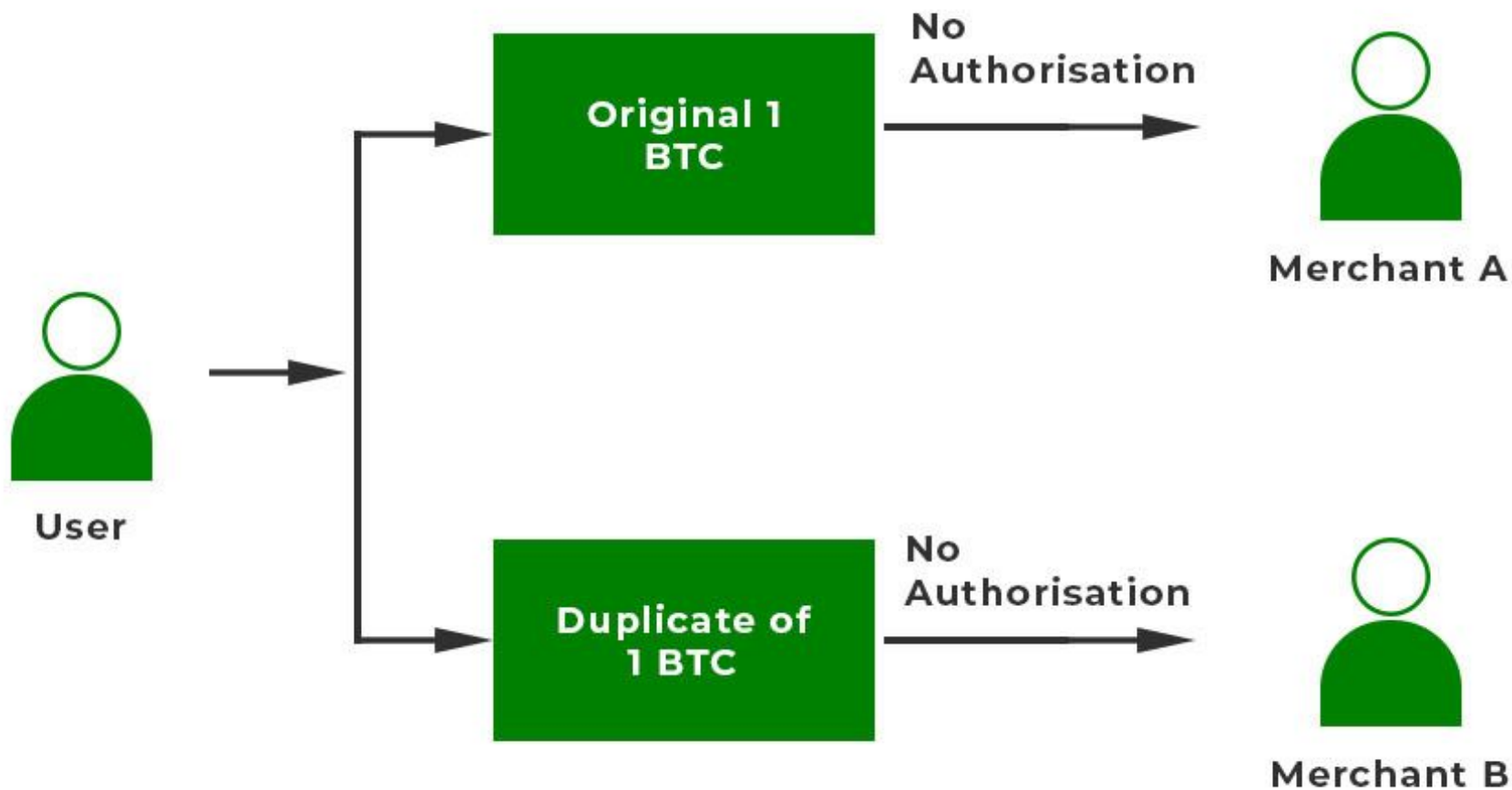


The mechanism of the blockchain ensures that the party spending the bitcoins is the real owner.



Bob

The technology behind Bitcoin ensures that the party who spends the bitcoins is the real owner by only processing verified transactions.



Case Studies of Blockchain Attacks



Bitcoin Gold 51% Attack (2018):

- Attackers controlled mining power, reversed transactions, and stole funds.
- Resulted in millions of dollars in losses.

Ethereum Classic 51% Attack (2020):

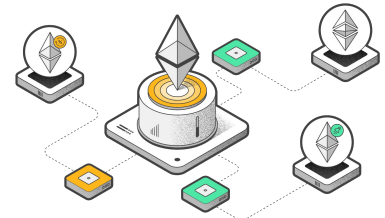
- Attackers used majority control to double-spend and manipulate the blockchain.
- \$5.6 million stolen.

Lesson Learned:

- Even though the blockchain itself is secure, networks with lower mining power are more vulnerable to 51% attacks.

Defense Mechanisms Against Attacks

PROOF OF WORK



Proof of Work (PoW):

- Miners solve complex puzzles to add blocks to the chain.
- PoW makes it very difficult for attackers to control the network without immense computing power.

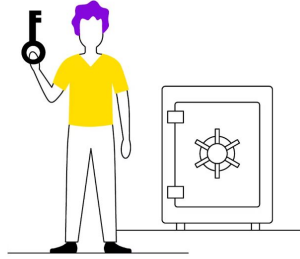
Proof of Stake (PoS):

- Validators are selected based on the amount of cryptocurrency they own and are "staking."
- If they act dishonestly, they lose their stake (coins), which makes attacks costly.

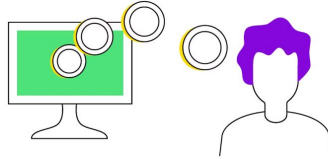
Real-life analogy:

- PoW is like a race where participants solve math problems. The harder the problem, the harder it is to cheat.
- PoS is like a voting system where those who have more at stake have more influence. If they cheat, they lose their stake.

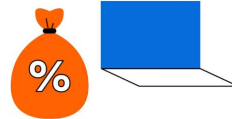
Proof of Stake



In a PoS network, the creator of a block is chosen according to certain criteria, such as wealth

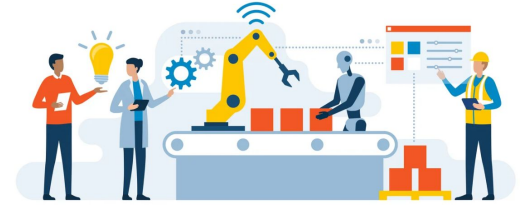


Stakers receive transaction fees but there is no block reward



Cryptocurrencies mined based on PoS require less computational power and are thus more cost-effective

Recap



Blockchain Security is important to keep transactions safe and prevent fraud.

Common Attacks we discussed:

- **51% Attack:** Hackers control the majority of the network.
- **Sybil Attack:** Fake users flood the system.
- **Eclipse Attack:** Isolating nodes with false data.
- **Double-Spending:** Trying to spend the same cryptocurrency twice.

Defense Mechanisms help stop these attacks and keep blockchain safe.

Quiz 1 – Double-Spending Concept



Question 1: What is a Double-Spending Attack?

- A) Spending money twice by using the same funds
- B) Spending money once on two different products
- C) Spending money in two different stores with different wallets
- D) All of the above

Quiz 2 – 51% Attack Understanding



Question 2: What does a 51% Attack involve?

- A) Attacker controls over 50% of the blockchain's mining power.
- B) Attacker creates fake accounts on the network.
- C) Attacker steals coins without changing the ledger.
- D) Attacker adds new users to the blockchain.

Quiz 3 – Blockchain Security Mechanisms



Question 3: Which of the following is NOT a way blockchain prevents attacks?

- A) Double-Spending Prevention
- B) Secure Ledger Recording
- C) Unlimited mining power for everyone
- D) Consensus Mechanisms (Proof of Work, Proof of Stake)

ANSWERS



Correct Answer: **A)** Spending money twice by using the same funds.

Correct Answer: **A)** Attacker controls over 50% of the blockchain's mining power.

Correct Answer: **C)** Unlimited mining power for everyone.

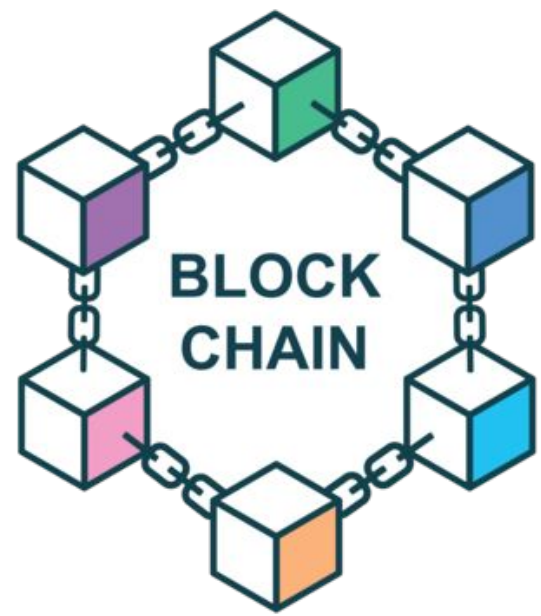
Thank You

Blockchain Security ISSUES

UNIT 1 - Day 2

Name: Divesh Jadhvani

Goal : Understanding the security issues in blockchain



The Untrusted Piggy Bank 🐷💰 -A Simple Example of Smart Contract Vulnerabilities

1. You and friends create a **shared piggy bank** where anyone can **deposit and withdraw** money.
2. Everything works fine... until Alex finds a trick.

The Trick (A Security Flaw):

1. Alex **requests ₹100 withdrawal**.
2. Before the system updates his balance, he **quickly requests another ₹100**.
3. The system hasn't recorded the first withdrawal, so **Alex keeps withdrawing repeatedly**.
4. **Result?** The piggy bank is **drained** unfairly! 💸

⚠️ What Went Wrong?

- The system **didn't update the balance before allowing another withdrawal**.
- This is similar to a **Reentrancy Attack** in smart contracts.

🎯 Lesson for Today:

- **Smart contracts must be written carefully** to avoid such loopholes.
- Today, we'll learn how these issues happen and how to prevent them. 🚀

Agenda

Smart Contract Security & Solidity Issues - <https://etherscan.io/>

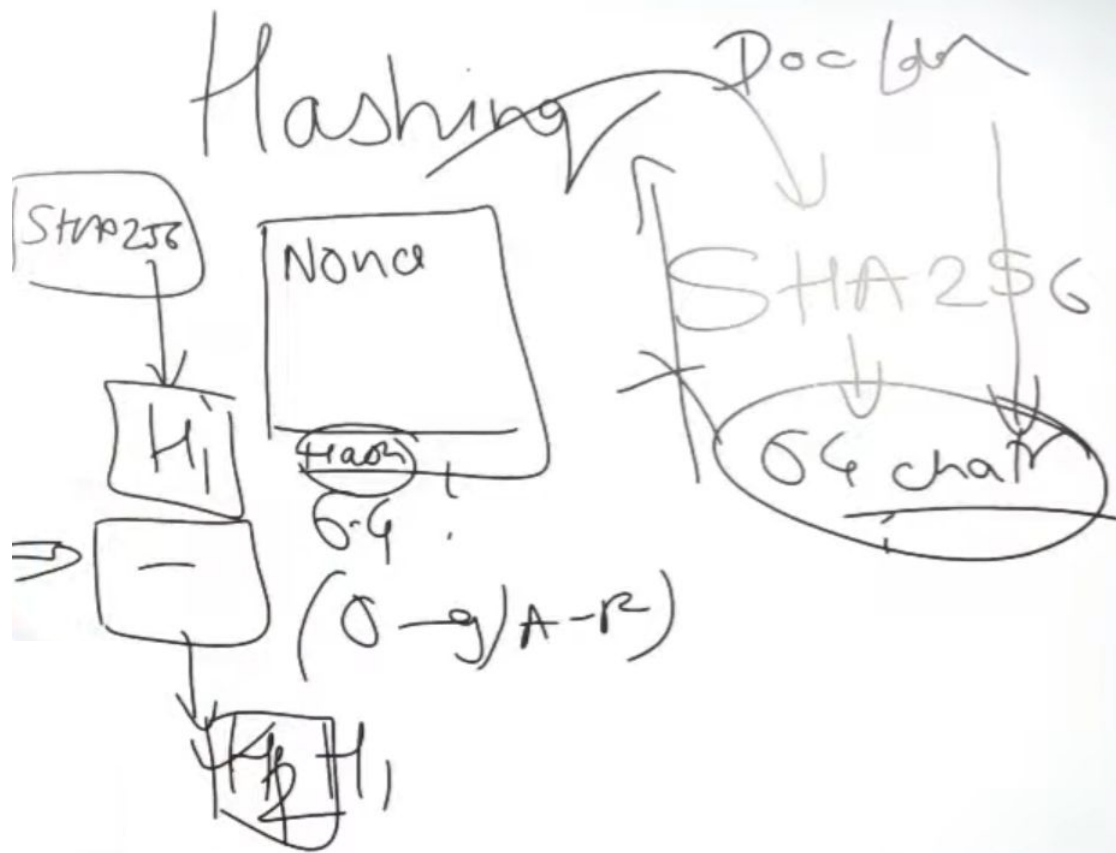
What is Solidity? <https://remix.ethereum.org/>

Common Solidity Attacks

- Reentrancy Attack
- Integer Overflow & Underflow
- Denial of Service (DoS) Attack
- Default Visibility Issues
- Randomness Issues

Case Studies





Solidity

PL

High Level

Statically
Typed

Smart
Contracts

.sol

Remix

Christain
Reitwiesner

C++
Python
JS

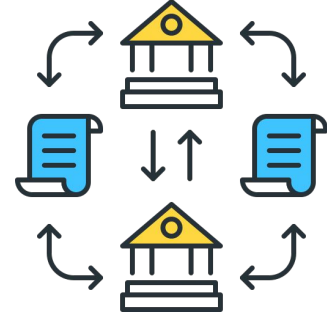
Ethereum

Gavin Wood

Case
sensitive

C curly bracket
language

Smart Contract Security & Solidity Issues



- What is a smart contract?
- Why are smart contracts vulnerable?
- Common security risks in Solidity

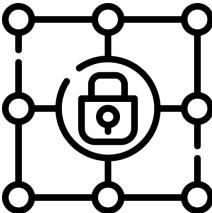
Example: A vending machine that automatically releases a product when money is inserted (like a smart contract). If the machine has a bug, people could take extra products without paying.



What is Solidity?

- Solidity is a programming language for writing smart contracts on Ethereum.
- It allows developers to create decentralized applications (DApps).
- Used for automated transactions without middlemen.

Example: Like writing instructions for a self-operating ATM that sends money only if the correct PIN is entered.



Why are Smart Contracts Vulnerable?

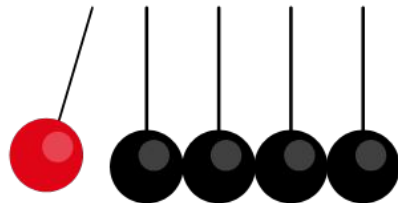
- Smart contracts are **immutable** (cannot be changed once deployed).
- Any mistake in the code remains forever.
- They handle money, making them a target for hackers.
- Lack of proper testing can lead to security issues.

OR

- **Code is Public** – Since blockchain transactions are open, anyone can inspect smart contracts and look for weaknesses.
- **No Updates** – If a smart contract has a flaw, it cannot be fixed unless a new contract is deployed.
- **Hacker Exploits** – Malicious users can manipulate smart contracts to steal funds.
- **Human Errors** – A single coding mistake can cause massive financial losses.

Example: A bank that automatically transfers salaries but a bug causes double payments.

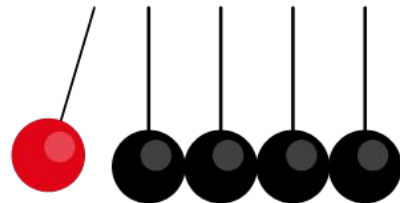
Common Solidity Security Issues



- Reentrancy Attack
- Integer Overflow & Underflow
- Denial of Service (DoS) Attack
- Default Visibility Issues
- Randomness Issues

Example: A poorly coded lottery smart contract that lets someone withdraw the prize multiple times.

Reentrancy Attack (With Example)



- Attackers exploit the way a contract sends funds before updating its balance.
- Example: A contract allows a user to withdraw multiple times before balance updates.
- **Example:** A store allows refunds but does not check if the item was returned, leading to multiple refunds for one item.

Solution: Use **Checks-Effects-Interactions** pattern and **Reentrancy Guard**.

Integer Overflow & Underflow

- Happens when a number exceeds its maximum or minimum limit.
- Example: A smart contract with uint8 (0-255) overflows back to 0.

Example: A car's speedometer going past 999 and resetting to 0.

Solution: Use **SafeMath library** to prevent miscalculations.

Example Scenario (Bank Account Trick):

- 1 You have **₹50,000** in your bank.
- 2 Instead of depositing ₹10,000, you **hack the system and deposit -10,000**.
- 3 If the system **doesn't check for negative numbers**, it might **think you deposited 9,99,99,99,999** (a huge positive number).
- 4 Now, the system believes you have **infinite money** and lets you withdraw **as much as you want**.

		Signed		Unsigned	
Type	Storage (Bytes)	Min	Max	Min	Max
TINYINT	1	-128	127	0	255
SMALLINT	2	-32768	32767	0	65535
MEDIUMINT	3	-8388608	8388607	0	16777215
INT	4	-2147483648	2147483647	0	4294967295
BIGINT	8	-2^{63}	$2^{63} - 1$	0	$2^{64} - 1$

Denial of Service (DoS) Attack


- Attackers block the execution of smart contracts.
- Example: A contract with a loop that never ends.

Example: A website that crashes because too many fake requests are sent at once.

Solution: **avoid loops** in critical functions.

Default Visibility Issues

- Functions in Solidity are **public by default**, making them easy to exploit.
- Example: A hacker calls a function that was not meant to be accessed externally.
- **Example:** A bank's admin panel is left open to the public, letting anyone transfer money.

```
contract Vault {  
  
    address public owner;  
  
    function changeOwner(address _newOwner) public {  
  
        owner = _newOwner; //  Should be restricted to the original owner!  
  
    }  
  
}
```


Randomness Issues in Smart Contracts

- Smart contracts cannot generate **true random numbers**.
- Example: Predictable randomness in gambling contracts leads to manipulation.

Example: A lucky draw where the winner can predict the result in advance.

Solution: Use **Oracles** like Chainlink for secure randomness.

Consensus

- mechanisms/
protocols/ (backbone)
Algorithms
- Valid or not
- Malicious behaviour
or Attacks
or hacking
- POW, POS, POA, POB, POET
- Differences exists 

POW

Miners

- 1) Block Reward
- 2) Requires heavy equipment investmest
- 3) Special equipment Units
- 4) Stakes Ki Jyada Baat Nahi hoti

Solve the puzzle

EC ↑

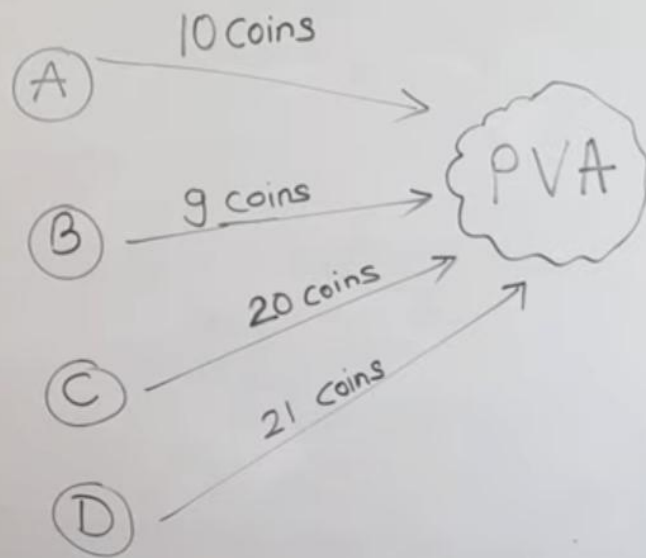
Computational Power

POS

- 1) Validators
- 2) Transaction Fees
- 3) NO Such requirement
- 4) Standard equipment Units
- 5) Sirf Stakes Ki hi baat hoti hai
- 6) No need
- 7) EC ↓
- 8) Cryptocurrencies

POB

- Burn the Coins
- Burn ↑ ↑ add the block
- Investment made through Coin burns
- Less P & E Consumption
- LTC



B ↑ A ↓ V ↑

POA

→ POW + POS

→ Miners + Validators

→ 1) Miners mine a new block using CP.

2) Group of validators are picked.

3) These validators then validate the block

4) After getting a valid tag it gets added to the blockchain

→ Attacks

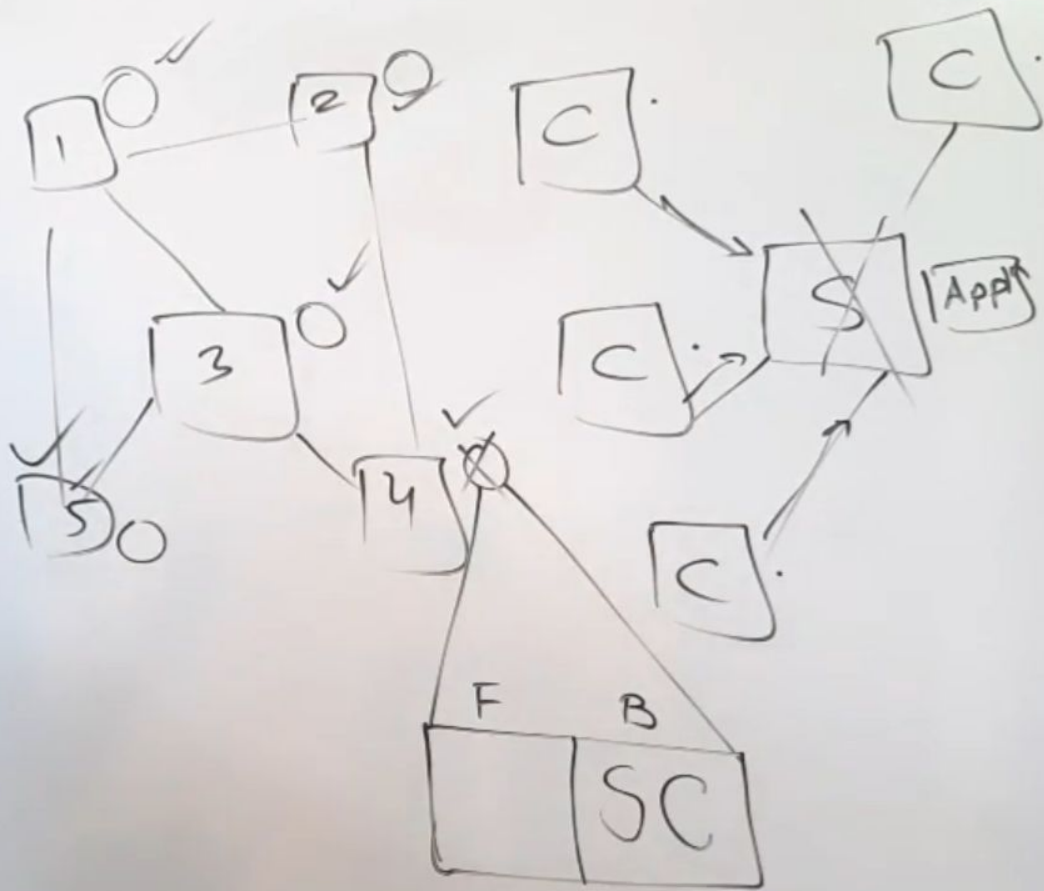
NewBlock: Header info
+
Reward
Addr

Decred

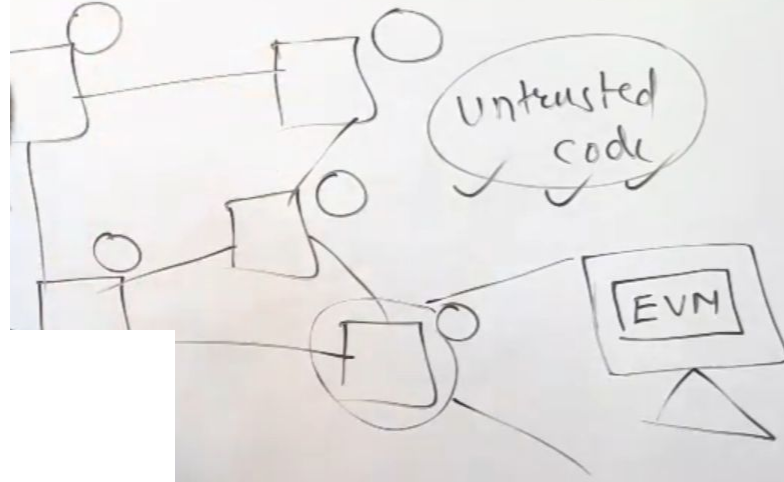
POET

- Permissioned Blockchain
- Intel
- Equal chance to win the Reward
- Waiting time

Dapps



EVM



Real-Life Smart Contract Hacks

- Bancor Vulnerability (2018) – \$23M loss (Due to visibility & reentrancy issues).
- Fomo3D Exploit (2018) – Ethereum Locked (Randomness issue exploited).
- PancakeSwap & Cream Finance DNS Hijacking (2021) – Phishing Attack.
- **Example:** A bug in an online banking app allowing hackers to withdraw more money than they had.

How to Secure Smart Contracts?



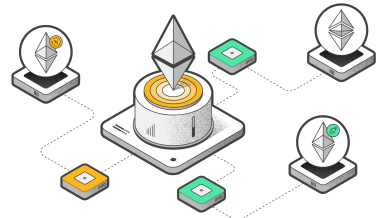
- Use **SafeMath** for calculations.
- Implement **Reentrancy Guard** to prevent multiple withdrawals.
- Limit loops & expensive operations to prevent **DoS attacks**.
- Set correct function **visibility** to avoid unauthorized access.

Use **external Oracles** for randomness.

Example: Just like a bank has security checks, smart contracts need proper coding and safeguards to prevent fraud.

PROOF OF WORK

Recap & Key Takeaways



- Smart contracts are **powerful but vulnerable**.
- Common security issues include **reentrancy, overflow, and DoS**.
- Real-world attacks have led to millions in losses.

Security best practices help **prevent vulnerabilities**.

Example: Just like locking doors and setting up alarms for security, smart contracts need protective coding.