Name: Gautham Gali

VT ID: 906577777

VT email: ggali14@vt.edu

1. In the Proof-of-Stake mechanism, there are “Nothing at Stake” and “Long Range” attacks.

A. Describe these attacks in detail.

B. Why do these attacks not exist in the Proof-of-Work?

Ans.

Nothing at Stake:

In proof-of-stake consensus systems, validators have a financial incentive to mine on each blockchain split that occurs. This phenomenon, known as "nothing-at-stake," disrupts consensus and may increase the system's susceptibility to assaults.
A distributed consensus process known as proof-of-stake, or PoS, is predicated on the idea that miners, or validators, in a network have an ownership stake in it. This provides them a financial incentive to act honorably in order to preserve the value of both the network and their stake in it.
It is not computationally expensive for validators to append additional blocks to the end of a proof-of-work blockchain, in contrast to proof-of-work based systems.

Proof of stake (PoS) is not a novel approach to consensus, nor is Ethereum the first system to attempt it. PoS was first deployed by Peercoin in 2013, and shortly after, other projects (including PIVX, Reddcoin, etc.) adopted their own variations of the technology. These initiatives aimed to preserve the high degrees of security and decentralization necessary for a cryptocurrency network while removing the issues related to energy-intensive mining through the use of proof-of-work (PoS).
Although the PoS community was excited about their new consensus technique, critics quickly pointed out two possible security vulnerabilities with PoS: the nothing at risk dilemma and the long range attack problem.
A sizable portion of the cryptocurrency community wasn't persuaded when PoS initially emerged that token ownership alone would be sufficient to deter unethical activity. The nothing at stake dilemma was one of their main worries.

The nothing at stake theory postulates that, in the initial iterations of PoS, each validator will continue to add to each other's forks.
Validators are required to accomplish this for two main reasons.

Firstly, validating transactions across several forks doesn't cost a validator anything, unlike proof of work (PoW). Because you no longer require proof-of-work (PoW) to construct a block, it is computationally inexpensive to build on every fork.

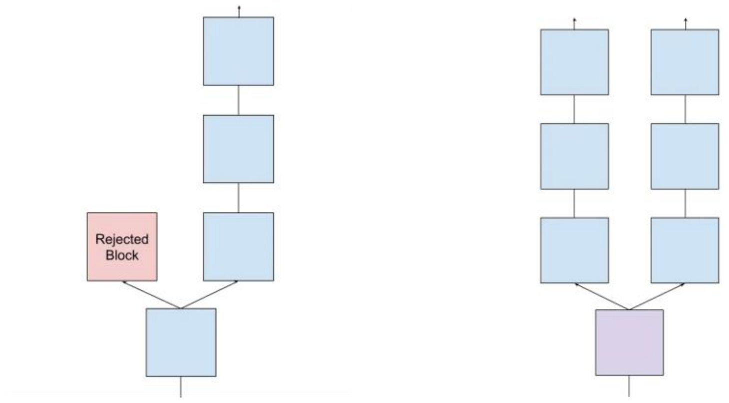
Second, because it is believed that building on every fork will benefit them financially, validators are expected to do so. Validators will get transaction fees on the victorious fork if they mine on

Name: Gautham Gali

VT ID: 906577777

VT email: ggali14@vt.edu

both (or more) chains. Validators will at the very least break consensus and increase the network's susceptibility to double spend attacks if they bet in each fork.



On the left, we an ideal situation. Sometimes forks occur, but they are resolved quickly. On the right, every validator is building on both forks. this hypothetical problem as the nothing at stake problem.

The motivation to mine several chains at once is absent from PoW. A miner's odds of mining a block are not increased if they decide to divide their hash power (computer power) across two chains.

Strict Security Assumptions
It is important to highlight the fact that the nothing at stake theory takes extremely strict security assumptions into account. Here is a list of the security assumptions:

1. It assumes that a validator will seek profit whenever there is an opportunity to do so, even if it is at the expense of the security or quality of the network. This is a standard (and smart) security assumption in the crypto world.
2. It assumes that 0 validators will act altruistically (aka no validators will mine on only 1 chain at a time out of the goodness of their hearts).
3. It assumes that validators went out of their way to either modify their validation software or download software that someone else modified. Standard validation software will not come with the ability to mine on all forks. This is because standard software comes with an internal logic for choosing a “true” fork when a fork occurs.

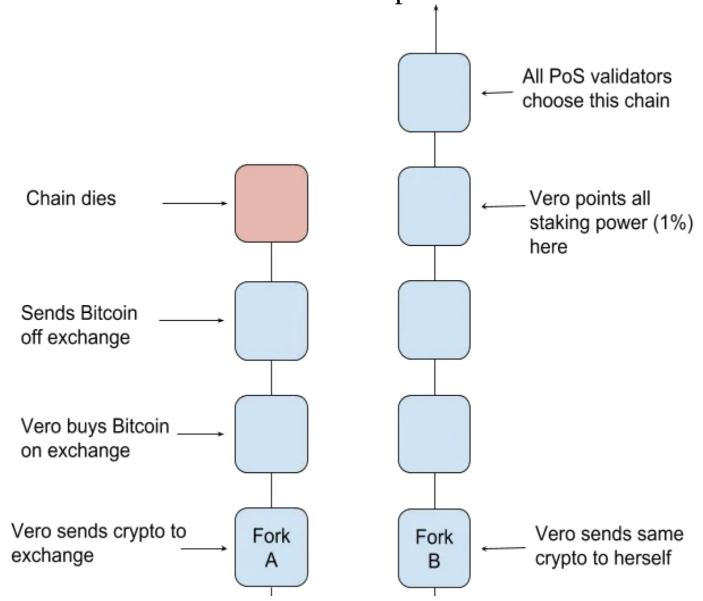
The fear of the nothing at stake theory is not just that it might delay or complicate consensus, the primary fear is that it will enable cheaper attacks compared to proof of work. In PoW, an attacker needs 51% of the total hash power to attack the network, but in PoS it is suspected that it costs as little as purchasing 1% of the stake in the network.

Name: Gautham Gali

VT ID: 906577777

VT email: ggali14@vt.edu

Here what the 1% attack is suspected to look like.


All PoS validators choose this chain
Chain dies
Vero points all staking power (1%) here
Sends Bitcoin off exchange
Vero buys Bitcoin on exchange
Vero sends crypto to exchange
Fork A
Fork B
Vero sends same crypto to herself

First, let's assume that a fork has occurred and all validators are actively building on both forks. Veronica, our attacker, is interested in executing a double spend attack. She sends her crypto to an exchange in one fork (fork A) and to a public key she controls in the other fork (Fork B). After enough time passes, the exchange accepts her deposit because it is recognizing fork A. Veronica then buys some Bitcoin with her deposit and quickly takes that Bitcoin off the exchange.

Now this is how she gets away with it.

Now she points her 1% staking power to fork B. Eventually she is selected to validate a block and builds only on Fork B. Being that everyone was building on both and there wasn't a clear "longest chain," the entire network now converges on (aka chooses) Fork B.

Veronica has now successfully stolen Bitcoin off an exchange. The crypto she sent to the exchange has returned to a public key in her control and now she has some extra Bitcoin too.

A more realistic attack scenario

The previous attack scenario was dependent on the assumption that every validator would build on every fork when a fork took place. This is how our attacker was able to double spend with only a 1% stake in the network. In a more realistic scenario, it is safe to assume that there will be honest validators that refuse to build on forked chains.

Name: Gautham Gali

VT ID: 906577777

VT email: ggali14@vt.edu

If Veronica wanted to double spend in this scenario, she would either have to buy more stake in the network or bribe other validators to help her in the attack.

Long Range attacks:

One of the greatest threats against Proof of Stake protocols is Long Range attacks. Due to the existence of Weak Subjectivity and Costless Simulation, these attacks are more dangerous than in Proof of Work protocols

In short, a Long Range attack is a scenario where an adversary creates a branch on the blockchain starting from the Genesis block and overtakes the main chain. This branch may contain different transactions and blocks and is also referred to as Alternative History or History Revision attack.

Weak Subjectivity:

This term is used to describe a problem that affects new nodes of a blockchain network and nodes who are brought online after a significant amount of time being offline. Online nodes are not affected by weak subjectivity.

When a new node is added to the network, it is always provided with the genesis block. This is the only block that all nodes accept as the first one. Along with the genesis block, the node will be presented with all of the currently published branches of that blockchain. Unfortunately, the node will not be able to tell right away which branch is the main chain.

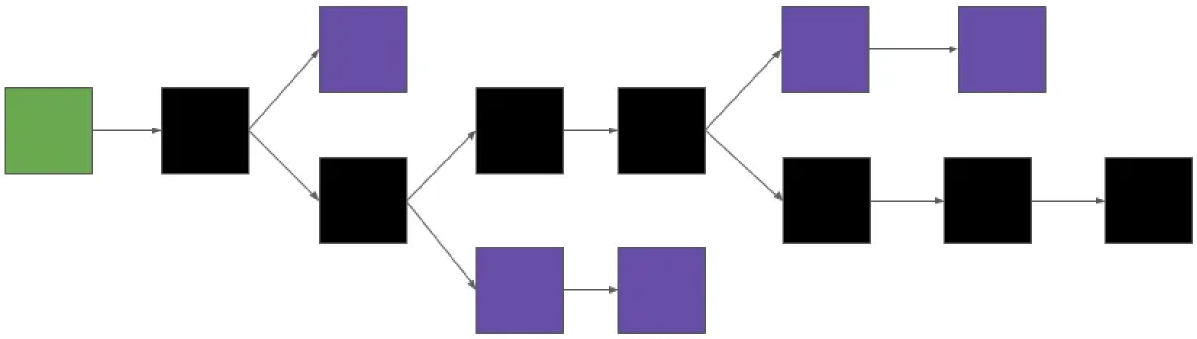
The same applies to nodes that have been offline for a long period of time e.g. a few months. At some point in time, these nodes knew which branch was the main-chain but after all that time being offline, they no longer do.

Online nodes follow and monitor the blockchain in real-time. They cannot be duped into accepting a different branch as the main chain unless a branch legitimately becomes the main chain.

Name: Gautham Gali

VT ID: 906577777

VT email: ggali14@vt.edu



Sample blockchain, the green block is the genesis block, purple denotes the orphaned/branches and the black blocks the main chain

As you can see, the above blockchain snapshot has many different branches. Some of them are longer than others. The main chain could be any of the branches that are presented.

The first rule we're going to talk about in this article is the longest chain rule. According to this rule, the main chain will always be the branch with the greatest number of blocks. In Proof of Work blockchains where physical resources are needed to produce a block, the longest chain rule is a great way to identify how much work has been invested in a branch.

Long Range Attacks

There are three different types of Long Range attacks to this date. Most of the publications usually mix the first two cases which are the Simple and Posterior Corruption, or they just accept Posterior Corruption as the only case of these attacks. Stake bleeding is fairly new research, published in 2018.

1. Simple
2. Posterior Corruption
3. Stake Bleeding

Starting off with the simplest case of Long Range attacks, we will build up to more complex scenarios. In our examples, we have a validator pool with 3 validators, Bob, Alice, and Malory. For the sake of simplicity, all of them own the same portion of the system's stake, 33,3%.

Key Detail: Information about validators and their stake is located in the Genesis block.

Simple

The first case refers to a naive implementation of the Proof of Stake protocol. In this scenario, nodes do not check block timestamps.

In a normal cycle of the PoS protocol, every validator will get the chance to validate blocks.

Name: Gautham Gali

VT ID: 906577777

VT email: ggali14@vt.edu

Snapshot of the example blockchain, each validator has an equal chance of getting elected

Malory decides to perform a Long Range attack and creates an alternative branch of the blockchain. Malory goes back to the genesis block, forks the blockchain and starts minting her branch.

Since the validator information is located inside the Genesis block, Malory will not be able to produce blocks faster than she would in the main-chain. Malory produces blocks with the same rate. With all those conditions, the only way for Malory to advance her branch and overtake the main-chain will be to produce blocks ahead of time.

In Malory's branch the chances of being elected are the same as on the other branch. "Parenthesis" blocks are forfeited blocks. The current length of Malory's branch is 2 blocks

Triple dots denote multiple forfeited blocks. In order for Malory to compete with the main-chain, she will have to compute blocks ahead of time. In the above blockchain snapshot both branches have 5 blocksMalory will have to forge timestamps, and since she is the sole active stakeholder in that branch it is possible to do so. In implementations where nodes do not take into consideration timestamps, both branches will be valid and nodes won't be able to spot Malory's trick.

B. Absence of These Attacks in Proof-of-Work

These attacks are largely non-existent in Proof-of-Work systems due to the following reasons:

1. High Cost of Simultaneous Mining: In PoW, the significant computational power and electricity required to solve cryptographic puzzles (mining) means that miners have a high cost for each block they attempt to produce. Therefore, miners are financially disincentivized to waste resources on older chains or on multiple chains simultaneously as seen in the "Nothing at Stake" attack in PoS. Miners are compelled to follow the protocol's rules and mine on the longest valid chain, as their potential reward is tied to their investment in computational resources.

2. Impracticality of Rewriting History: Rewriting a long segment of the blockchain in a PoW system would require an enormous amount of computational power to redo the work of the entire altered segment up to the present. This makes a Long Range attack highly impractical and expensive because the attacker would need to control more than 50% of the entire network's hash rate to achieve such a feat, ensuring a level of security proportional to the amount of work (and real-world resources) required to support the chain.

---

2. One of the main goals of bitcoin is to achieve anonymity in digital transaction.

A. Describe the main techniques that Bitcoin used towards enabling anonymity.

B. Unfortunately, bitcoin is far from being completely anonymous. Describe how bitcoin transactions can be deanonymized. How many ways the attacker can exploit to do so?

Name: Gautham Gali

VT ID: 906577777

VT email: ggali14@vt.edu

Bitcoin employs several techniques to enhance user anonymity, making it difficult (though not impossible) to trace transactions back to individuals. Here are the main techniques used:

1. Pseudonymous Addresses

Each Bitcoin user has the ability to create many public addresses, which are essentially accounts that aren't connected to them personally. The random strings of digits and characters that make up these addresses are produced by the users' cryptographic keys. They are not genuinely anonymous; instead, they are regarded as pseudonymous as they do not include identifiable information. By utilizing a different address for every transaction, users may improve their privacy and make it more difficult to connect all of their transactions to a single person.

2. Decentralized Network

As a decentralized peer-to-peer network, Bitcoin lacks a central location for the processing or storing of transaction data. Because there is no central database of transaction details or personal information that can be accessed, as there is in traditional banking systems, this decentralization naturally makes it more difficult to locate specific users.

3. No Mandatory Identity Verification

Users of Bitcoin are not required to submit to identity verification in order to interact with the network. Anyone without personal information may download a wallet and begin transacting. All users benefit from an additional degree of anonymity due to the lack of required identity.

4. Mixing Services

Mixing services, often referred to as tumblers, are used to obscure the Bitcoin transaction traces in order to promote transactional anonymity. Through separate addresses, a mixing service returns coins that may be recognizable or "tainted" to users by combining them with other currencies. Because this procedure severs the direct connections between arriving and departing transactions, it may become much more challenging to determine the origin of the cash.

5. Network Layer Privacy

The user's IP address, which could normally be used to connect transactions to a person's location and maybe their identity, can be hidden when using technologies like Tor in conjunction with Bitcoin transactions. The anonymous connection between Bitcoin nodes via Tor makes traceability much more difficult.

B. 1. Clustering of Addresses

Name: Gautham Gali

VT ID: 906577777

VT email: ggali14@vt.edu

Research by Meiklejohn et al. (2013) in the paper "A Fistful of Bitcoins: Characterizing Payments Among Men with No Names" demonstrated how clustering of addresses could be used to identify entities controlling multiple addresses. This technique is based on the assumption that all inputs to a transaction are controlled by the same entity. By analyzing the blockchain's public ledger and grouping together addresses that appear as inputs in the same transactions, researchers and attackers alike can cluster addresses that are likely controlled by the same wallet. This can potentially expose the transaction history and balance of a user if even one address in the cluster is de-anonymized.

## 2. Idiosyncratic Features of Wallet Software

Different Bitcoin wallet software has unique behaviors that can inadvertently aid in transaction tracking and user identification:

Change Addresses: Wallets typically generate new addresses for transaction change. This is a privacy feature designed to prevent reuse of addresses. However, attackers can exploit this by distinguishing which outputs are change addresses. Since change addresses are typically new addresses that have never appeared on the blockchain before, if an output address in a transaction is new, it is likely a change address. Non-change outputs, which are addresses used to send Bitcoin to another party, might have appeared previously on the blockchain.

Linking Change to Inputs: Once an attacker identifies which output is the change, they can reasonably infer that the other outputs are payments and that the inputs (previous transactions feeding into the transaction) are controlled by the same user as the change address. This helps to link addresses and transactions over time to a single user or wallet, increasing the ability to track their activities.

## 3. Transaction Graph Analysis

Transaction graph analysis is a method that examines the relationships between transactions and addresses on the application layer of the Bitcoin protocol. This involves more than just looking at the inputs and outputs of transactions; it also analyzes the timing of transactions and the amounts transferred to infer which addresses might be controlled by the same entities. This analysis can reveal common spending patterns and may link seemingly unrelated addresses to a single entity or individual.

## 4. Network Layer Analysis

A different approach, highlighted by Dan Kaminsky during his 2011 Black Hat talk, focuses on the network layer of Bitcoin transactions. Kaminsky pointed out that "the first node to inform you of a transaction is probably the source of it." This suggests a method of analysis where one could set up

Name: Gautham Gali

VT ID: 906577777

VT email: ggali14@vt.edu

nodes throughout the Bitcoin network to monitor where transactions originate. If a specific node consistently broadcasts transactions before any other nodes, it is likely that the transactions are being created by the wallet associated with that node. This technique relies on analyzing network traffic to make educated guesses about the origins of transactions, potentially revealing the physical nodes or network endpoints where transactions are initiated.

---

3. Suppose Bob would like to receive a donation for his project.

So, he is planning to put his bitcoin addresses on a public donation forum along with his personal website. However, since all the users will make donations to one of these addresses, it is likely that all the donations can be linkable and reveal Bob's identity (due to his website).

To address this privacy issue, Bob has to generate a so-called stealth address that permits any sender to always derive new address per transaction and only Bob can know the corresponding private key.

A. Using a public key crypto technique that you are familiar with to design a simple scheme to generate stealth address securely.

B. Based on your stealth address design, explain how Bob can determine which transactions in the blockchain are directed to him. What is the cost of doing so?

A.

This system allows donors to send Bitcoin to an address that appears unique each time, yet all funds go to Bob's wallet without revealing his identity or linking transactions to him. Here is a simple scheme using public key cryptography to generate stealth addresses securely:

Step 1: Bob's Initial Setup

Bob needs to generate a pair of keys for his stealth address setup:

Bob's main public key (P): Bob generates a public/private key pair (P, p). He publishes P on his website and donation forums, not the actual Bitcoin address.

Bob's main private key (p): This private key is kept secure and never shared.

Step 2: Donor's Address Generation

When a donor decides to send Bitcoin to Bob, they follow these steps:

Generate a Random Number (r): The donor generates a random number that will be used only once.

Name: Gautham Gali

VT ID: 906577777

VT email: ggali14@vt.edu

Calculate a New Public Key (Q): The donor uses Bob's public key (P) and their random number (r) to generate a unique public key for the transaction. This can be done using Elliptic Curve Cryptography (ECC) as follows:

Calculate $R = r * G$, where G is the generator point of the elliptic curve.

Calculate $Q = R + P$.

Q is now a unique public key derived from Bob's public key and the donor's nonce.

Generate the Bitcoin Address: The donor converts Q into a standard Bitcoin address using the normal conversion process (apply SHA-256 hashing, then RIPEMD-160 hashing, and finally base58check to create the address).

Step 3: Communicating R

The donor needs to securely transmit R (or its hash) to Bob, typically included in the transaction's metadata or a separate message. This allows Bob to reconstruct the unique address Q without revealing r.

Step 4: Bob Receives the Donation

To access the funds sent to the unique address, Bob performs the following:

Obtain R from the Transaction: Bob retrieves R from the blockchain or the accompanying message.

Calculate Q: Bob uses his private key p and the received R to calculate Q:

$Q = R + p * G$.

Since $R = r * G$ and $P = p * G$, then $Q = r * G + p * G = (r + p) * G$, which matches the donor's calculation.

Access the Funds: With Q known, Bob can derive the private key associated with that particular transaction's address and access the funds sent to it.

Security and Privacy

Unlinkability: Each transaction uses a different Bitcoin address, which makes it impossible to link them together just by looking at the blockchain.

Security: Only Bob can calculate and access the private key corresponding to each unique address because only he knows his private key p.

Simplicity: The method is based on well-established cryptographic principles, particularly those used in ECC and Bitcoin's address generation algorithms.

This approach allows Bob to receive donations anonymously on public forums without exposing his transaction history or balance, ensuring privacy and security in his fundraising efforts.

Name: Gautham Gali

VT ID: 906577777

VT email: ggali14@vt.edu

B.

Bob can identify transactions directed towards him by utilizing his private key and the public metadata embedded in each transaction. This process requires some computation, which is the primary cost associated with using stealth addresses. Here's how Bob can determine which transactions are meant for him and the associated costs:

Determining Transactions Directed to Bob

Scan the Blockchain: Bob continuously monitors the blockchain for transactions that include the public part of the nonce (R) in their metadata. Since R is essential for deriving the unique address for each transaction, it must be made publicly available by the donor within the transaction or its accompanying data.

Reconstruct the Public Key (Q): For each transaction that includes an R, Bob performs the following calculations:

Calculate Q: Bob computes $Q = R + p * G$, where p is his private key and G is the generator point used in the elliptic curve employed by Bitcoin. This computation yields the public key corresponding to the address to which the donor sent the funds.

Verify the Address: Bob transforms the derived public key Q into a Bitcoin address using the same method Bitcoin uses (SHA-256, followed by RIPEMD-160, and then Base58Check encoding). He then checks if this address matches the receiving address of the transaction.

Claim the Funds: Once Bob verifies that the address derived from Q matches the recipient address in the transaction, he knows the transaction was intended for him, and he can proceed to claim the funds.

Cost of Using Stealth Addresses

Computational Cost: The primary cost of using stealth addresses is computational. Each transaction directed to Bob requires:

One elliptic curve point addition ($R + p * G$).

The standard Bitcoin address generation process (SHA-256, RIPEMD-160, and Base58Check encoding).

Given that elliptic curve operations are relatively fast but not trivial in terms of computational resources, the cost increases with the number of transactions Bob needs to verify. This can become significant if there are many small transactions.

Storage and Bandwidth Cost: Bob needs to store his private key and maintain a system that can continually scan the blockchain and process transactions. This involves storage for the blockchain and bandwidth to stay updated with the latest blocks, which can be substantial given the size and growth rate of the blockchain.

Name: Gautham Gali

VT ID: 906577777

VT email: ggali14@vt.edu

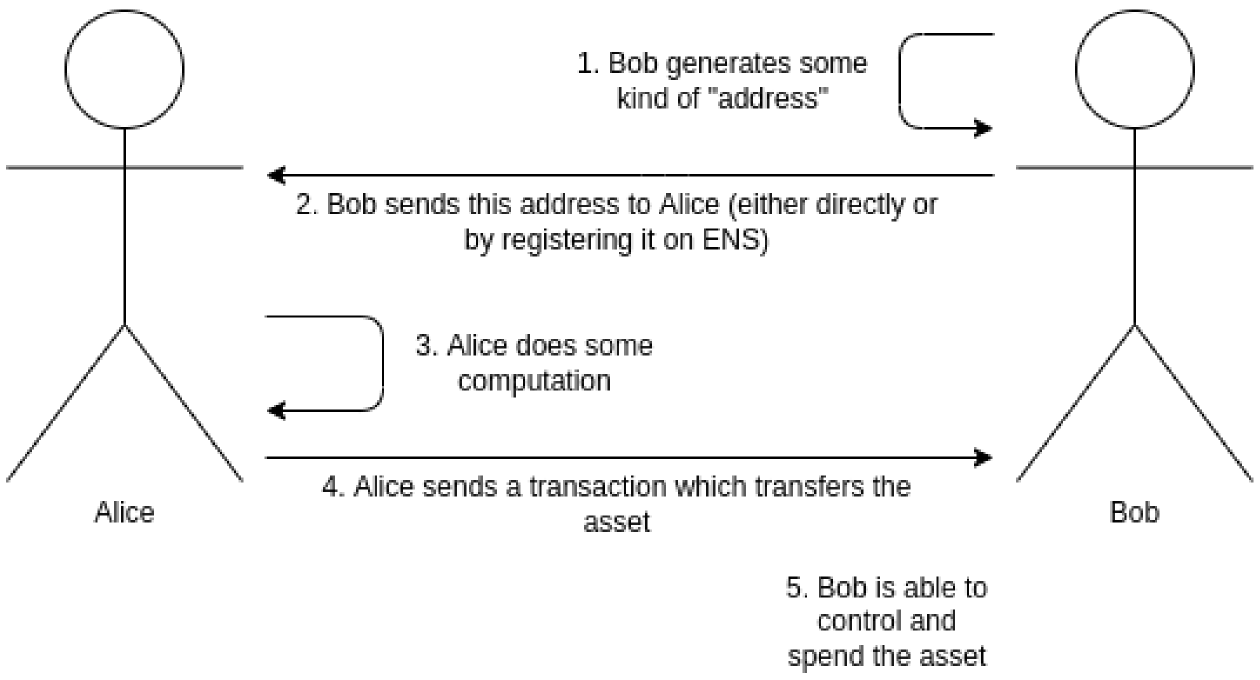
Privacy and Security Cost: While stealth addresses greatly enhance privacy by making transactions unlinkable directly on the blockchain, they slightly increase the risk if Bob's private key (p) is ever compromised. The attacker would then potentially be able to derive and claim future transactions directed to Bob using the same method he employs.

Ease of Use: There is an inherent cost in terms of ease of use for both Bob and his donors. Donors must be able to generate and correctly embed R in their transactions, which requires additional tools or modifications to standard Bitcoin wallets. Bob must also educate his donors on how to correctly use stealth addresses to ensure the transactions are processed correctly.

Creating stealth addresses on Ethereum

Suppose Alice wants to send some ETH to Bob privately. It is impossible to hide the transfer, but concealing Bob's identity is more feasible. Either party to this transaction can create a stealth address that the recipient (Bob) will control. For Alice, this requires a special public key and a key only the sender knows.

Bob does not have to generate a new address for each transaction, and he does not have to interact with Alice at all if he registers his meta address on ENS — Ethereum's Domain Name System. Here is how such stealth addresses work.


1. Bob generates some kind of "address"
2. Bob sends this address to Alice (either directly or by registering it on ENS)
3. Alice does some computation
4. Alice sends a transaction which transfers the asset
Alice
Bob
5. Bob is able to control and spend the asset

1. Bob generates a *root spending key* and a *stealth meta address* for his Ethereum name (e.g., bob.eth). He passes the meta address to Alice or records it in ENS.

Name: Gautham Gali

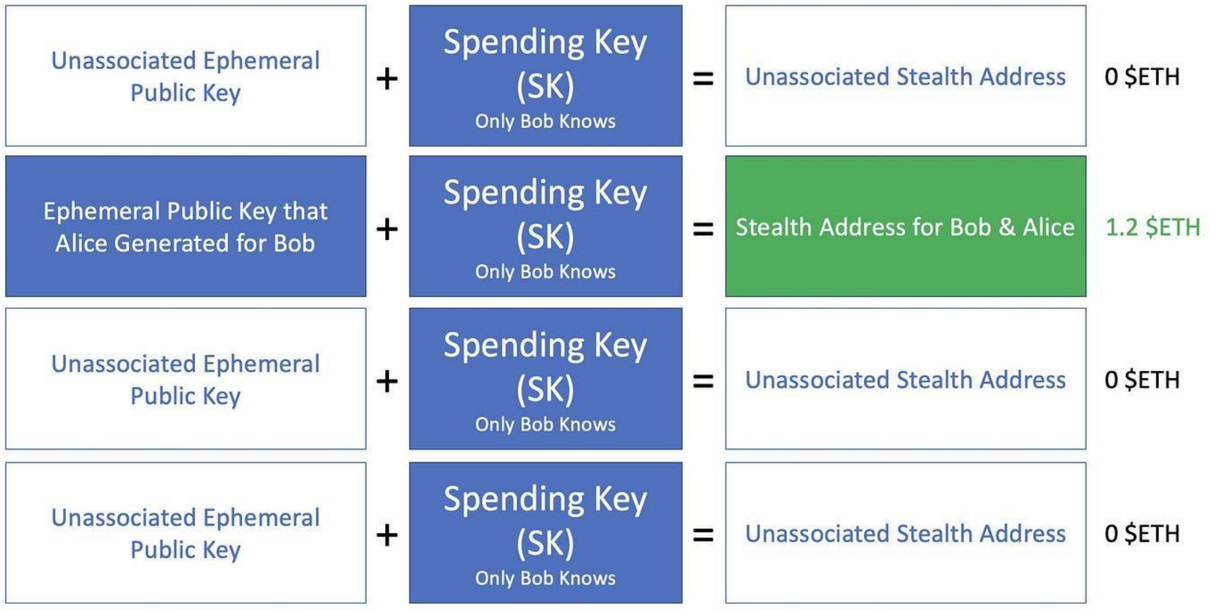
VT ID: 906577777

VT email: ggali14@vt.edu

2. Knowing the meta address (or after looking it up in ENS), Alice creates Bob's stealth address via a computation involving a single-use *ephemeral key* only she knows.

3. Alice transfers ETH to Bob's stealth address. In the process, Alice generates and publishes an *ephemeral public key* – an on-chain piece of cryptographic data.

4. Bob scans the registry of all ephemeral public keys to discover the recipient address. In the process, each ephemeral public key is combined with his root spending key to generate a stealth address and check if it holds any assets. Once a match is found, Bob generates and memorizes the spending key for that address.


Unassociated Ephemeral Public Key
+
Spending Key (SK) Only Bob Knows
=
Unassociated Stealth Address
0 $ETH
Ephemeral Public Key that Alice Generated for Bob
+
Spending Key (SK) Only Bob Knows
=
Stealth Address for Bob & Alice
1.2 $ETH
Unassociated Ephemeral Public Key
+
Spending Key (SK) Only Bob Knows
=
Unassociated Stealth Address
0 $ETH
Unassociated Ephemeral Public Key
+
Spending Key (SK) Only Bob Knows
=
Unassociated Stealth Address
0 $ETH

One of the obstacles to this implementation is the Ethereum gas fees. When digital assets, whether fungible or non-fungible, land in a stealth address, it is otherwise empty. The owner cannot transfer what he received elsewhere. As their stealth address only contains what the sender transferred, they have no ETH to pay for the network gas.

Sending ETH from the main wallet is possible but undesirable. Such a transaction would create a publicly visible link, defeating the purpose of the stealth address. Buterin mentions two solutions – a type of zero-knowledge proofs (ZK-SNARKs) and specialized transaction aggregators. The first option is prohibitively expensive – "hundreds of thousands of gas just for a single transfer."

Aggregators (aka "searchers") let you buy a set of "tickets" at once and use them to pay for transactions on-chain. To transfer assets from a stealth address, one would submit an encoded pre-paid ticket to the aggregator. The latter would include their transaction in a bundle repeatedly until it was accepted in a block. This method does not involve additional fees and has few trust and regulatory concerns due to its purpose.

Name: Gautham Gali

VT ID: 906577777

VT email: ggali14@vt.edu

Spending and viewing keys

Thanks to the elliptic curve technology, users do not have to manage everything with one root spending key. Instead, they may use a spending key and a viewing key, with the latter showing its owner's stealth addresses without enabling spending.

More efficient scanning

Adding a view tag to each ephemeral public key makes scanning their sets easier. This view tag may be minuscule – just one byte of the shared secret. According to Buterin's estimates, this would speed up Bob's calculations to create and check the full address to 1/256 of the original time. Computing the shared secret would only require one elliptic curve multiplication for each ephemeral public key.

---

4. Off-chain storage was introduced to address the (cost) problem of storing large amount of data on the chain. With programmable blockchain, it is possible to perform computation beyond data storage.

A. If the computation is too heavy, would it be possible to move the computation off-the-chain as storage? If not, why? If yes, describe the main techniques to enable off-chain computation and what should be stored on blockchain afterwards?

off-chain storage solutions and programmable blockchains such as those that support smart contracts (like Ethereum) significantly extend the capabilities and efficiency of blockchain applications by addressing the limitations associated with storing large amounts of data directly on-chain and performing complex computations. Here’s how these technologies contribute to the blockchain ecosystem:

Off-Chain Storage

Off-chain storage refers to the practice of storing data outside the blockchain but in a way that is securely linked to the blockchain. This method addresses several critical issues:

Cost Reduction: Storing data on-chain can be prohibitively expensive, especially on blockchains like Ethereum where the cost of transactions (gas fees) can vary significantly. Off-chain storage allows for large amounts of data to be stored at a much lower cost.

Scalability: By moving data off-chain, the blockchain is relieved of the burden of processing large volumes of data. This helps in scaling the network and improving transaction speeds.

Privacy: Off-chain storage can enhance privacy since sensitive data doesn’t need to be publicly recorded on the blockchain. Access to off-chain data can be controlled and managed to ensure privacy and security.

Name: Gautham Gali

VT ID: 906577777

VT email: ggali14@vt.edu

Flexibility: Off-chain solutions provide flexibility in managing data. For example, data can be stored in traditional databases, file storage systems, or even other decentralized storage networks like IPFS (InterPlanetary File System), depending on the application's requirements.

Programmable Blockchains

Programmable blockchains that allow for the execution of smart contracts enable complex computations to be performed in a decentralized manner. Smart contracts are self-executing contracts with the terms of the agreement directly written into lines of code. Here’s how programmable blockchains are enhancing capabilities:

Automated Transactions: Smart contracts automate transactions and other blockchain operations, reducing the need for intermediaries and increasing efficiency.

Extended Functionality: Beyond simple transactions, smart contracts enable a range of decentralized applications (dApps) to be built on blockchain platforms. These can include decentralized finance (DeFi) applications, supply chain management tools, digital identity systems, and much more.

Conditional Transactions: Smart contracts allow for transactions that are executed only when certain predefined conditions are met. This can be used for escrow services, automatic payments upon completion of services, and other conditional operations.

Interoperability: Some programmable blockchains are designed to support interoperability with other blockchains, enabling a seamless exchange of information and value across different networks.

Enhanced Security: Although smart contracts are susceptible to bugs, the decentralized nature of blockchain provides a higher level of security compared to traditional computing systems. However, the immutable nature of blockchain means that once a smart contract is deployed, it cannot be altered, emphasizing the need for rigorous testing and security audits.

Integration of Off-Chain Storage and Programmable Blockchains

Integrating off-chain storage with programmable blockchains often involves using oracles or other middleware solutions that allow smart contracts to interact securely with off-chain data. This integration enables complex applications that require extensive data storage and sophisticated computations without overburdening the blockchain or compromising on speed and efficiency.

Together, off-chain storage and programmable blockchains represent a powerful combination that addresses many of the inherent limitations of earlier blockchain technologies, paving the way for broader adoption and more innovative applications in various sectors.

Name: Gautham Gali

VT ID: 906577777

VT email: ggali14@vt.edu

A.

Yes, it is indeed possible and often practical to move heavy computations off the blockchain to overcome the limitations associated with on-chain executions, such as high costs, lower speeds, and scalability issues. Off-chain computation involves processing data and executing complex algorithms outside the blockchain, while still ensuring the results are verifiable and tamper-proof when brought back onto the blockchain. This approach is facilitated by several techniques:

**Main Techniques for Off-Chain Computation**

1. **Layer 2 Scaling Solutions**: These are protocols built on top of the blockchain to increase its processing capacity. Examples include state channels, sidechains, and rollups. Each has unique mechanisms for handling transactions off-chain while still securing them with the underlying blockchain's security model.
    - **State Channels**: Allow two parties to conduct multiple transactions in a private channel and only submit the initial and final states to the blockchain. This is useful for applications like micropayments and interactive games.
    - **Sidechains**: Independent blockchains that run parallel to the main blockchain and can have different parameters for block size or consensus methods. They link back to the main chain allowing for asset and data transfer.
    - **Rollups**: Execute transactions outside the main Ethereum chain (Layer 1) but post transaction data on Layer 1. They enhance scalability by processing many off-chain transactions as a single one on-chain.
2. **Trusted Execution Environments (TEEs)**: These are secure areas within processors that provide a level of assurance regarding the integrity and confidentiality of the operations performed within them. Examples include Intel’s SGX (Software Guard Extensions). TEEs can perform computations and then provide proofs that the computation was executed correctly, which can be verified on-chain.
3. **Zero-Knowledge Proofs (ZKP)**: These are cryptographic proofs that allow one party to prove to another that a given statement is true, without revealing any additional information apart from the fact that the statement is true. ZKPs can be used for off-chain computations where the proof of computation’s correctness is submitted on-chain, rather than the data itself or the detailed execution path.
4. **InterPlanetary File System (IPFS) and Other Decentralized Storage Solutions**: While primarily used for storage, systems like IPFS can work alongside other decentralized computing resources to manage data used in off-chain computations.

**What Should Be Stored on Blockchain After Off-Chain Computation?**

After off-chain computation, certain elements need to be stored on-chain to ensure integrity, transparency, and verifiability:

Name: Gautham Gali

VT ID: 906577777

VT email: ggali14@vt.edu

1. **Final State or Result**: The outcomes of the computations, especially if they affect state changes on the blockchain (e.g., balances, ownership records).
2. **Proofs of Computation**: This includes cryptographic proofs such as those provided by zero-knowledge proofs or hashes that verify the correctness and completeness of the off-chain computations.
3. **Commitments**: Before starting off-chain computation, parties may submit commitments to the blockchain. These are hashes of the data or the intended computation, which can later be revealed and verified against the commitments.
4. **Transaction Receipts or Metadata**: These provide a record of off-chain activities relevant to on-chain transactions and can be used for audit or conflict resolution.

---

5. In class, we have studied the Byzantine Broadcast (BB) problem, where a single node (the sender) has a private input and the goal is to broadcast that input to everyone else. For this problem, you can assume that the private input is either 0 or 1.

A closely related problem is Byzantine Agreement (BA). In this problem, each node i € {1,2,..., n} has its own private bit bi € {0, 1}. Up to f out of n nodes can be byzantine, meaning they can behave arbitrarily. A deterministic BA protocol must satisfy the following two properties:

• Agreement: the protocol always terminates with all honest nodes outputting the same bit.

• Validity: if all honest nodes have the same private input b € {0, 1}, then the protocol terminates with all such nodes outputting b.

A. Show that a deterministic BA protocol can only exists when f ≤ n/2.

B. Given f < n/2, prove that there exists a deterministic BB protocol satisfying validity and agreement (as defined for the BB problem) if and only if there exists a deterministic BA protocol satisfying validity and agreement (as defined for the BA problem).

To understand why a deterministic Byzantine Agreement (BA) protocol can only exist when $f \leq n/2$, it's essential to dive into the definitions and implications of the Agreement and Validity conditions within the constraints of up to $f$ Byzantine (arbitrary behaving) nodes among $n$ total nodes.

**Agreement and Validity Conditions**

1. **Agreement**: All honest nodes must agree on the same output bit.
2. **Validity**: If all honest nodes start with the same initial bit $b$, then they all must output $b$.

**Analysis for $f > n/2$**

Name: Gautham Gali

VT ID: 906577777

VT email: ggali14@vt.edu

When the number of Byzantine nodes $f$ is greater than $n/2$, the Byzantine nodes can form a majority in the network. This majority allows them to potentially influence the network in several detrimental ways:

- **Dominating Communication**: They can refuse to pass along messages or alter the messages being sent between honest nodes. This can create partitions in the network where honest nodes receive different information.
- **Creating Conflicting Views**: Byzantine nodes can send conflicting information to different honest nodes. For instance, they could send some honest nodes a bit of 0 and others a bit of 1.
- **Overriding the Validity Condition**: Even if all honest nodes start with the same bit $b$, Byzantine nodes can introduce the opposite bit effectively enough that some honest nodes may end up agreeing on an incorrect bit, violating the validity condition.

**Why $f \leq n/2$ is Required**

With $f \leq n/2$, honest nodes always form a majority, or at least half, in the network. This majority is crucial because:

- **Majority Voting**: Honest nodes can potentially override the Byzantine nodes by sheer numbers, assuming they can communicate effectively and verify the integrity of their communication (which often requires additional assumptions like signatures or a reliable broadcast mechanism).
- **Reaching Consensus**: Since honest nodes hold a majority, any protocol that relies on a majority vote (such as majority-based Byzantine Fault Tolerance protocols) will correctly reflect the initial bit $b$ of the honest nodes, assuming they all started with the same $b$.
- **Preventing Splits**: Byzantine nodes lack the numbers to create splits in the network that prevent consensus among honest nodes.

Let's consider an example to illustrate why a deterministic Byzantine Agreement (BA) protocol requires that $f \leq n/2$ and demonstrate what happens when $f > n/2$.

**Example Setup**

Consider a network of $n$=5 nodes where up to $f$ nodes can behave Byzantine. Let's first examine the situation where $f$=2 (which satisfies $f \leq n/2$) and then see what happens when $f$=3 (which violates $f \leq n/2$).

**Case 1: $f$=2 (Satisfying $f \leq n/2$)**

Name: Gautham Gali

VT ID: 906577777

VT email: ggali14@vt.edu

- **Nodes: A, B, C are honest, D and E are Byzantine.**
- **Initial Bits: Suppose all honest nodes have the same initial bit $b$=0.**

**Protocol Execution:**

1. **Round 1**: Each node sends its bit to every other node.
    - Honest nodes A, B, C send 0.
    - Byzantine nodes D and E send 1 to some nodes and 0 to others, specifically:
        - D sends 0 to A and B, and 1 to C.
        - E sends 1 to A and B, and 0 to C.
2. **Receiving Phase**:
    - A receives: 0 (from B, C), 1 (from D, E)
    - B receives: 0 (from A, C), 1 (from D, E)
    - C receives: 0 (from A, B), 1 (from D, E)
3. **Majority Decision**:
    - All honest nodes see that the majority of received bits (including from other honest nodes) are 0, thus they all agree on 0.

Here, despite the Byzantine behavior, the honest majority ensures the correct bit is agreed upon, fulfilling both the Agreement and Validity conditions.

**Case 2: $f$=3 (Violating $f≤n/2$)**

- **Nodes: A and B are honest, C, D, and E are Byzantine.**
- **Initial Bits: Suppose honest nodes have the same initial bit $b$=0.**

**Protocol Execution:**

1. **Round 1**: Each node sends its bit to every other node.
    - Honest nodes A and B send 0.
    - Byzantine nodes C, D, and E send 1 to all nodes, or even worse, send mixed signals:
        - C sends 1 to all.
        - D sends 0 to A and 1 to B.

Name: Gautham Gali

VT ID: 906577777

VT email: ggali14@vt.edu

- E sends 1 to A and 0 to B.

2. **Receiving Phase**:
    - A receives: 0 (from B), 1 (from C, D, E) and possibly another 0 from D (manipulated).
    - B receives: 0 (from A), 1 (from C, D, E) and possibly another 0 from E (manipulated).
3. **Majority Decision**:
    - A and B may see a majority of 1s (especially if manipulation is not aligned or more sophisticated lies are told).

Here, the Byzantine nodes have enough presence to distort the communication, potentially leading to a situation where the honest nodes cannot reliably agree on the correct initial bit or even agree on the same output bit, violating both the Agreement and Validity conditions.

This example clearly demonstrates that when $f > n/2$, Byzantine nodes can effectively disrupt the protocol by either preventing a unanimous agreement or causing honest nodes to agree on incorrect outputs. This proves the necessity for $f \leq n/2$ in any deterministic Byzantine Agreement protocol to ensure correct and consistent outcomes despite Byzantine faults.

B.

**Definitions and Requirements**

**Byzantine Broadcast (BB):**

- **Validity**: If the sender is honest and its input bit is $b$, then all honest nodes output $b$.
- **Agreement**: All honest nodes output the same bit.

**Byzantine Agreement (BA):**

- **Validity**: If all honest nodes have the same input bit $b$, then all honest nodes output $b$.
- **Agreement**: All honest nodes output the same bit.

**Proving Equivalence**

**From BB to BA:**

1. **Assume BB Protocol Exists**: Suppose there is a deterministic BB protocol that satisfies the conditions of validity and agreement.

Name: Gautham Gali

VT ID: 906577777

VT email: ggali14@vt.edu

2. **Construct a BA Protocol**: Use the BB protocol to construct a BA protocol. The idea is to have each node propose its bit using the BB protocol, taking turns as the sender.
3. **Execution**:
    - Each node $i$ acts as a sender in a round of the BB protocol and broadcasts its bit $bi$ to all other nodes.
    - Each node collects the broadcasted bits from all rounds. Since $f<n/2$ , the majority of these bits will be from honest nodes.
4. **Decision**:
    - Each node computes the majority bit of the received values. If most honest nodes started with the same bit, then by the validity of the BB protocol, the majority of outputs in their respective rounds will be that bit, ensuring the BA protocol's validity.
    - By the BB protocol's agreement, all honest nodes receive the same set of bits in each round and hence compute the same majority, ensuring the BA protocol's agreement.

**From BA to BB:**

1. **Assume BA Protocol Exists**: Suppose there is a deterministic BA protocol that meets the conditions of validity and agreement.
2. **Construct a BB Protocol**: Use the BA protocol as a BB protocol by selecting a designated sender. This node broadcasts its bit to all nodes as the initial step of the BA protocol.
3. **Execution**:
    - The sender node sends its bit $b$ to all nodes, who then use this bit as their initial input for the BA protocol.
4. **Protocol Flow**:
    - All nodes run the BA protocol with the sender's bit as their input.
5. **Outcome**:
    - By the validity of the BA protocol, if the sender is honest and all honest nodes start with its bit $b$, then all honest nodes output $b$, satisfying the BB protocol's validity.
    - The BA protocol's agreement ensures that all honest nodes output the same bit, satisfying the BB protocol's agreement.

Name: Gautham Gali

VT ID: 906577777

VT email: ggali14@vt.edu

To illustrate the equivalence between a deterministic Byzantine Broadcast (BB) protocol and a deterministic Byzantine Agreement (BA) protocol under the condition $f<n/2$, let's consider a small network example with 5 nodes where up to 2 nodes can be Byzantine (i.e., $f=2, n=5$).

**Example Network Setup**

- **Nodes**: $A, B, C, D, E$
- **Assumption**: Nodes $D$ and $E$ can be Byzantine.

**Constructing BA from BB**

**Assume a BB Protocol Exists**:

- **Validity**: If the sender (say, $A$) is honest and its input bit is 0, then all honest nodes output 0.
- **Agreement**: All honest nodes output the same bit.

**Steps to Construct BA**:

1. **Broadcast Phase**:
    - Each node uses the B protocol to broadcast its bit to all other nodes.
    - Let's assume initial bits are $A=0, B=0, C=1$ (honest nodes) and $D$, $E$ (Byzantine) can send any bits.
2. **Collection Phase**:
    - Each node collects bits from all others. Assume $D$ and $E$ broadcast 1 consistently for confusion.
3. **Decision Phase**:
    - Each node applies a majority rule: $A, B, C$ count more 0's than 1's from honest nodes, thus decide on 0.

**Outcome**:

- **Agreement**: All honest nodes decide 0, regardless of Byzantine influence.
- **Validity**: The majority decision reflects the initial agreement among honest nodes.

**Constructing BB from BA**

**Assume a BA Protocol Exists**:

- **Validity**: If all honest nodes input the same bit $b$, then all honest nodes output $b$.
- **Agreement**: All honest nodes output the same bit.

Name: Gautham Gali

VT ID: 906577777

VT email: ggali14@vt.edu

**Steps to Construct BB**:

1. **Designated Sender**:
    - $A$ is the designated sender and decides to broadcast 0.
2. **Broadcast Phase**:
    - $A$ sends its bit 0 to all nodes as their initial input for the BA protocol.
3. **Execution of BA**:
    - All nodes execute the BA protocol using 0 as their starting bit.
    - $D$ and $E$ may try to influence the protocol but can only send their bits.
4. **Outcome**:
    - **Validity**: Since $AA$ is honest and all honest nodes used $AA$'s bit 0 as input, they all output 0.
    - **Agreement**: All honest nodes output 0.