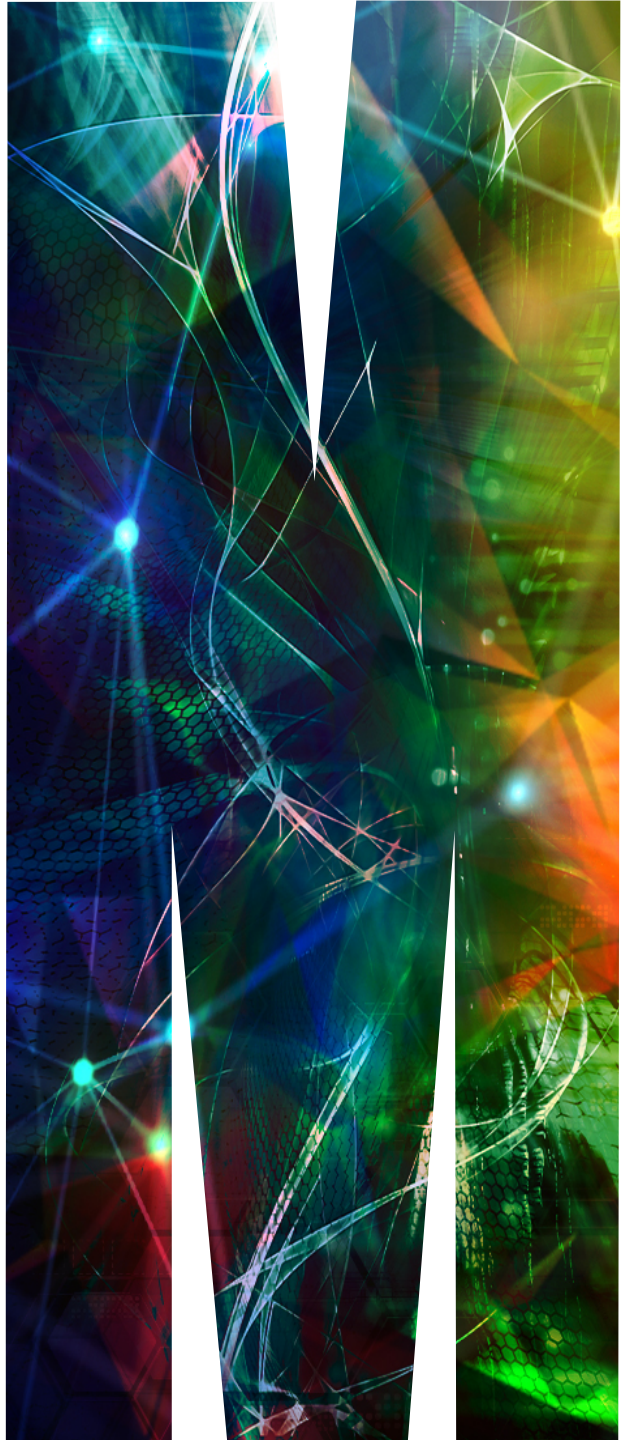
MONASH University

FIT5214: Blockchain

Lecture 10: Algorand



Lecturer: Rafael Dowsley

rafael.dowsley@monash.edu

https://dowsley.net

Unit Structure

- Lecture 1: Introduction to Blockchain
- Lecture 2: Bitcoin
- Lecture 3: Ethereum and Smart Contracts
- Lecture 4: Proof-of-Work (PoW)
- Lecture 5: Attacks on Blockchains
- Lecture 6: Class Test/Alternatives to PoW
- Lecture 7: Proof-of-Stake (PoS)
- Lecture 8: Privacy
- Lecture 9: Byzantine Agreement
- Lecture 10: Algorand
- Lecture 11: Blockchain Network
- Lecture 12: Payment Channels

Unit Structure

- Lecture 1: Introduction to Blockchain
- Lecture 2: Bitcoin
- Lecture 3: Ethereum and Smart Contracts
- Lecture 4: Proof-of-Work (PoW)
- Lecture 5: Attacks on Blockchains
- Lecture 6: Class Test/Alternatives to PoW
- Lecture 7: Proof-of-Stake (PoS)
- Lecture 8: Privacy
- Lecture 9: Byzantine Agreement
- Lecture 10: Algorand
- Lecture 11: Blockchain Network
- Lecture 12: Payment Channels

Recap: Proof-of-Stake (PoS)

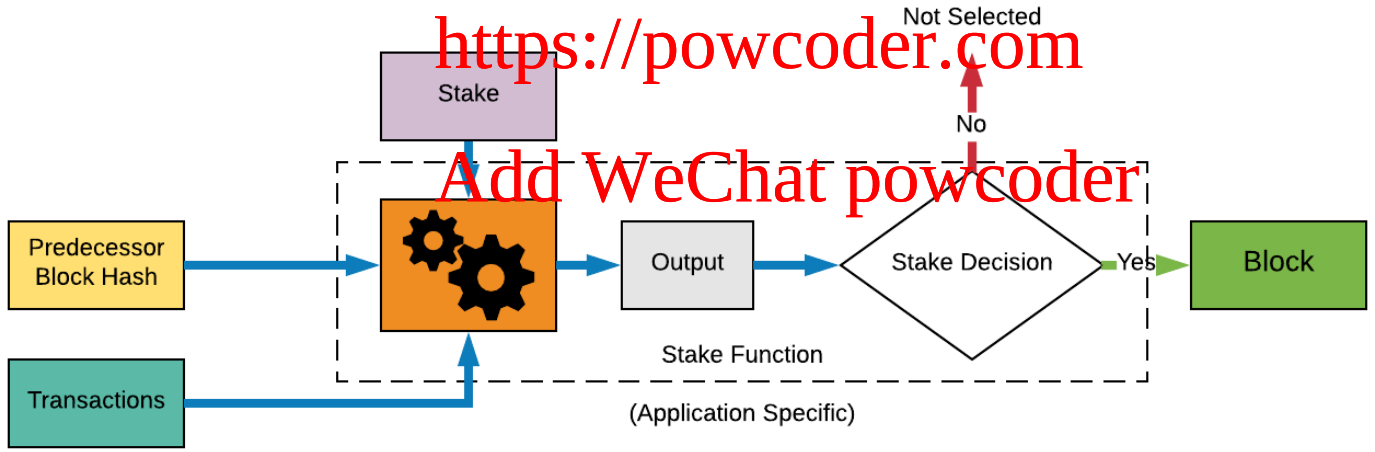
- PoW has the following disadvantages:
  - Control of the network is very centralised in a few mining pools.
  - A huge amount of energy is wasted computing useless hash outputs.
  - Migration of hash power to do 51% attacks in platforms with smaller amounts of total hash power.

- To solve some of PoW weakness, PoS uses the number of controlled coins (instead of hashing power) in order to determine the voting power.

Recap: Proof-of-Stake (PoS)

There are different types of PoS, depending on the "things" put at stake:

- Proof-of-Lock/Deposit
- Balance-based
- …



Nakamoto-style consensus: e.g., Ouroboros
BFT-style consensus: e.g., Algorand

Recap: Ouroboros Praos PoS

Nakamoto-style consensus dividing the time into slots. For each slot, select leader to create the block corresponding to that slot.

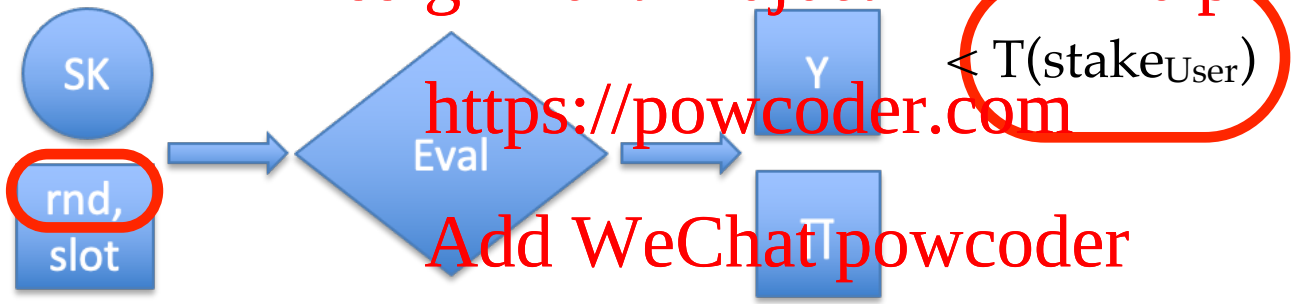
Lottery: the slot leader is randomly selected with the users probabilities of being selected proportional to their stake.

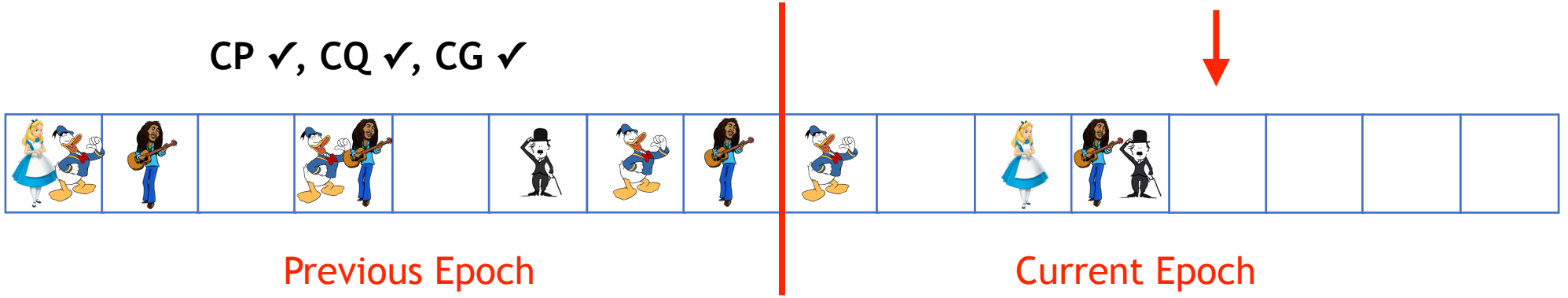
More than 50% of the coins should be controlled by users that behave honestly.

Recap: Ouroboros Praos PoS

The protocols works in epochs. Each epoch contains a fixed number of slots.

The parties need to agree on the random nonce rnd and stake distribution that is used to select leaders in the current epoch.





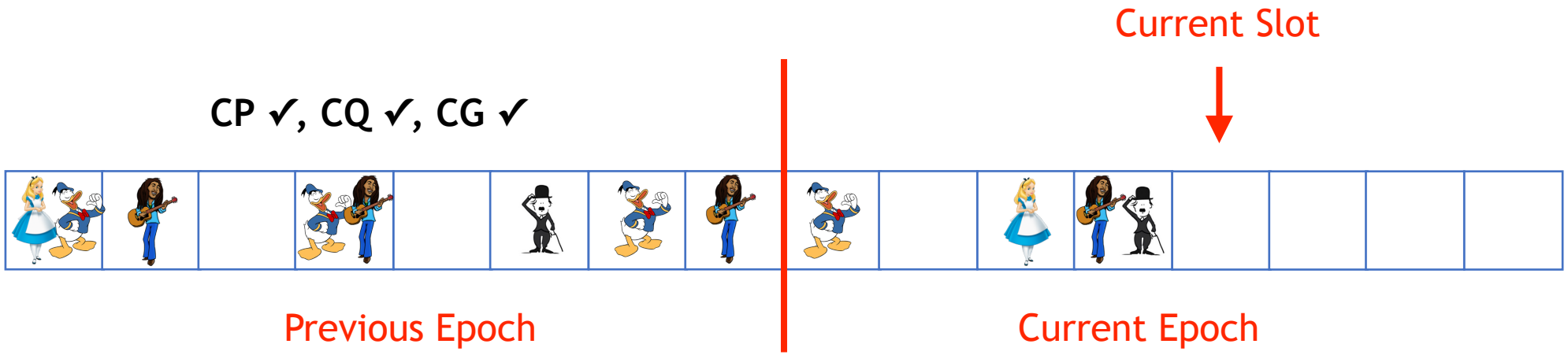
Recap: Ouroboros Praos PoS

The protocols works in epochs. Each epoch contains a fixed number of slots.

The parties need to agree on the random nonce rnd and stake distribution that is used to select leaders in the current epoch.

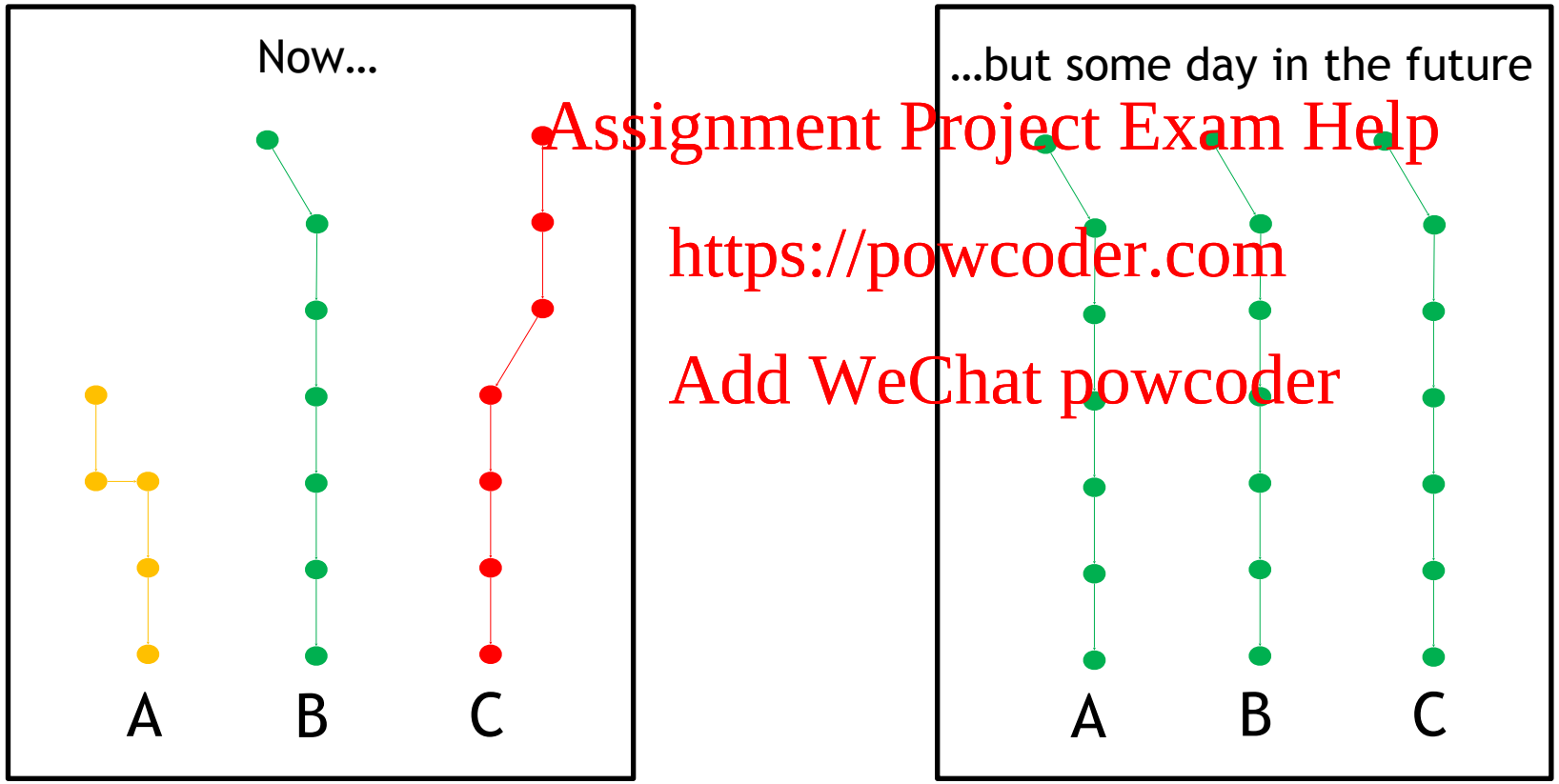
Agreement on the stake distribution is done in the previous epoch and fixed for the current epoch.

Similarly, an agreement on the new random nonce is achieved on the previous epoch by hashing verifiable random values from the chain to obtain the new nonce. And this nonce is affected by honest blocks.



Recap: Eventual Consistency

Eventually, all the nodes will agree on the same blockchain

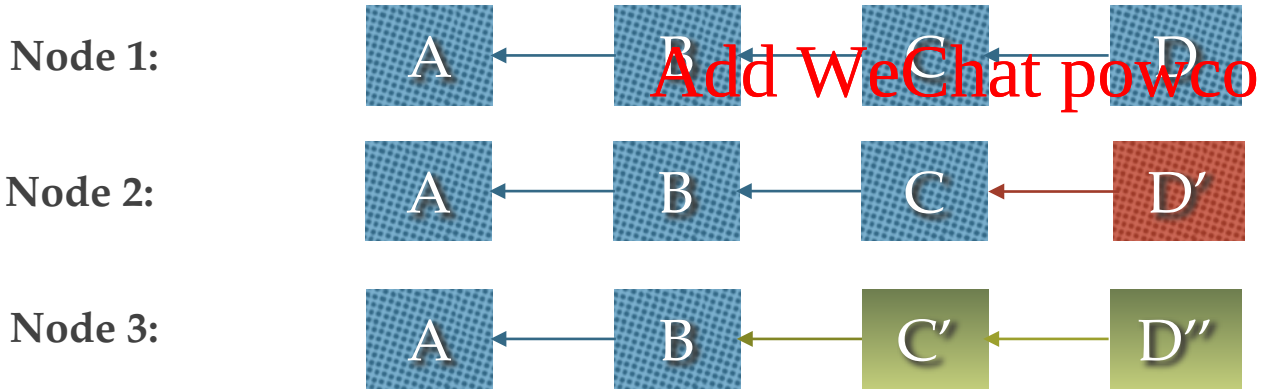


Recap: T-Consistency

T-consistency **is used to quantify the quality of the Nakamoto-style consensus — after cutting down the last T blocks, all nodes should agree on the rest of the blockchain.**

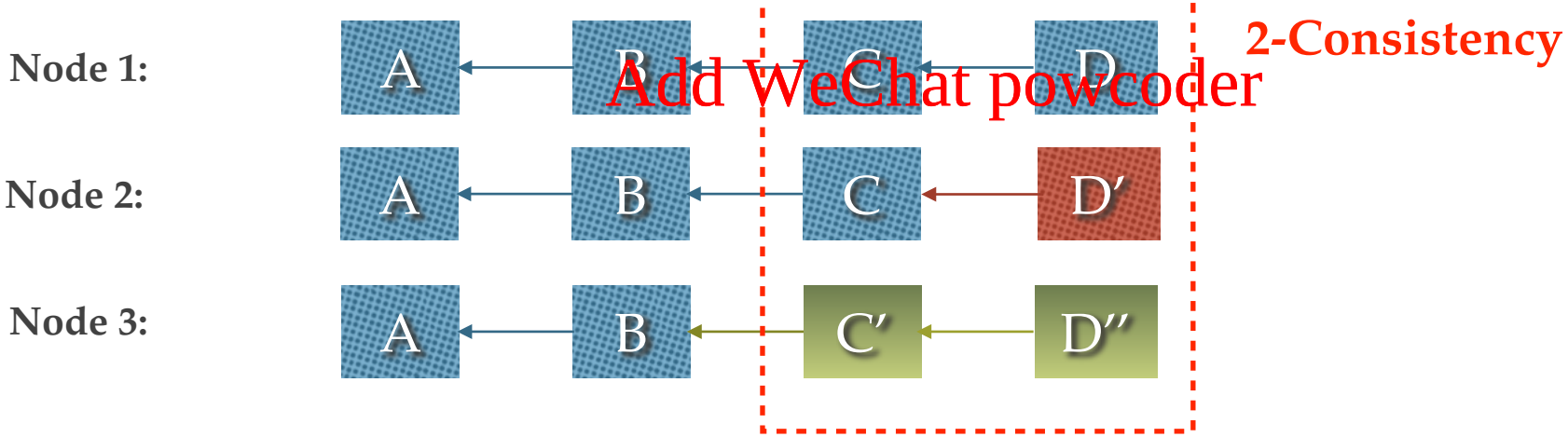
Recap: T-Consistency

T-consistency is used to quantify the quality of the Nakamoto-style consensus — after cutting down the last T blocks, all nodes should agree on the rest of the blockchain.



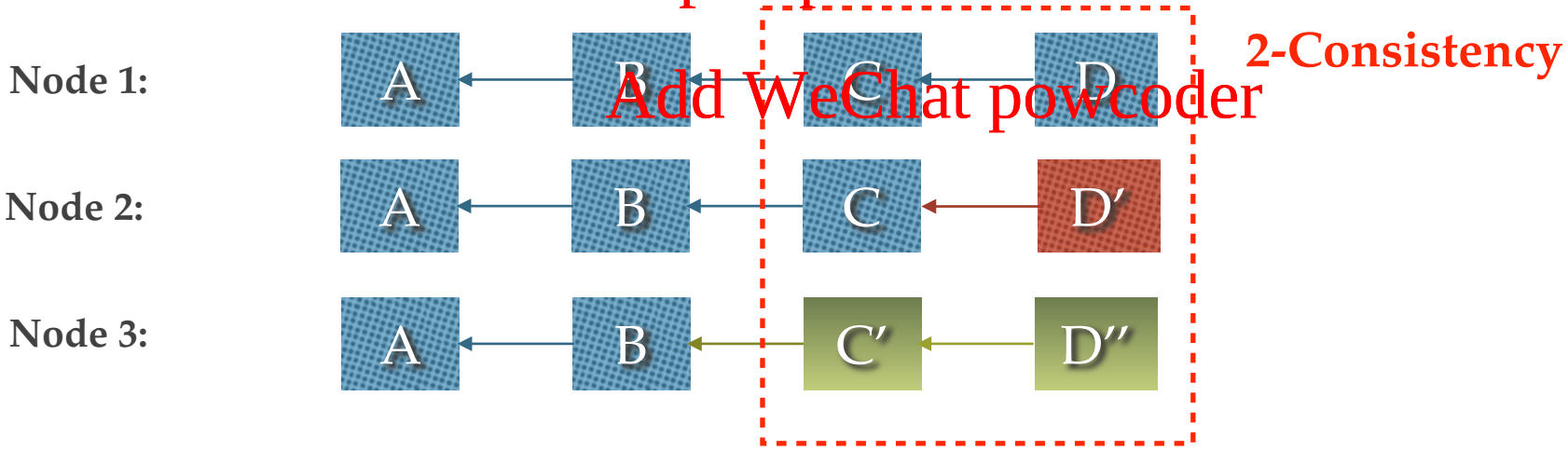
Recap: T-Consistency

T-consistency is used to quantify the quality of the Nakamoto-style consensus — after cutting down the last T blocks, all nodes should agree on the rest of the blockchain.



Recap: T-Consistency

T-consistency is used to quantify the quality of the Nakamoto-style consensus — after cutting down the last T blocks, all nodes should agree on the rest of the blockchain.



The best consistency is 0-consistency, which can be provided by traditional consensus protocols

Ouroboros Praos PoS

- As other blockchain solutions that use Nakamoto-consensus, Ouroboros Praos only guarantees eventual consistency.

Ouroboros Praos PoS

- As other blockchain solutions that use Nakamoto-consensus, Ouroboros Praos only guarantees eventual consistency.

- Therefore, in order to have a high degree of confidence that a transaction will not be reverted once inserted in a block, a user needs to wait until many blocks are added on top of the block containing the transaction.

Ouroboros Praos PoS

- As other blockchain solutions that use Nakamoto-consensus, Ouroboros Praos only guarantees eventual consistency.
- Therefore, in order to have a high degree of confidence that a transaction will not be reverted once inserted in a block, a user needs to wait until many blocks are added on top of the block containing the transaction.
- *But this is bad for the usability of the system in many realistic situations.*

Ouroboros Praos PoS

- As other blockchain solutions that use Nakamoto-consensus, Ouroboros Praos only guarantees eventual consistency.

- Therefore, in order to have a high degree of confidence that a transaction will not be reverted once inserted in a block, a user needs to wait until many blocks are added on top of the block containing the transaction.

- *But this is bad for the usability of the system in many realistic situations.*

- **Ideally, we would like to guarantee that a transaction would not be reverted as soon as the block containing it is accepted as part of the blockchain.**

Recap: consensus

Definition 1. (Consensus.) There are n nodes, of which at most f can be malicious, i.e., at least $n - f$ nodes are correct. For all $i \in [1, n]$, node i starts with an input value v_i. The nodes must decide for one of those values, satisfying the following properties:

- **Agreement:** All correct nodes decide for the same value.

- **Termination:** All correct nodes terminate in finite time.
- **Validity:** The decision value must be the input value of a node.

Recap: safety and liveness

- The **safety** of the global state is ensured by the consensus
  - The entire *valid* transaction history can be *agreed by all* correct nodes. Agreed values cannot be changed later on. (Agreement & Validity)
- The **liveness** of the system is ensured by the consensus
  - All nodes can *terminate* their process and reach a conclusion. (Termination)

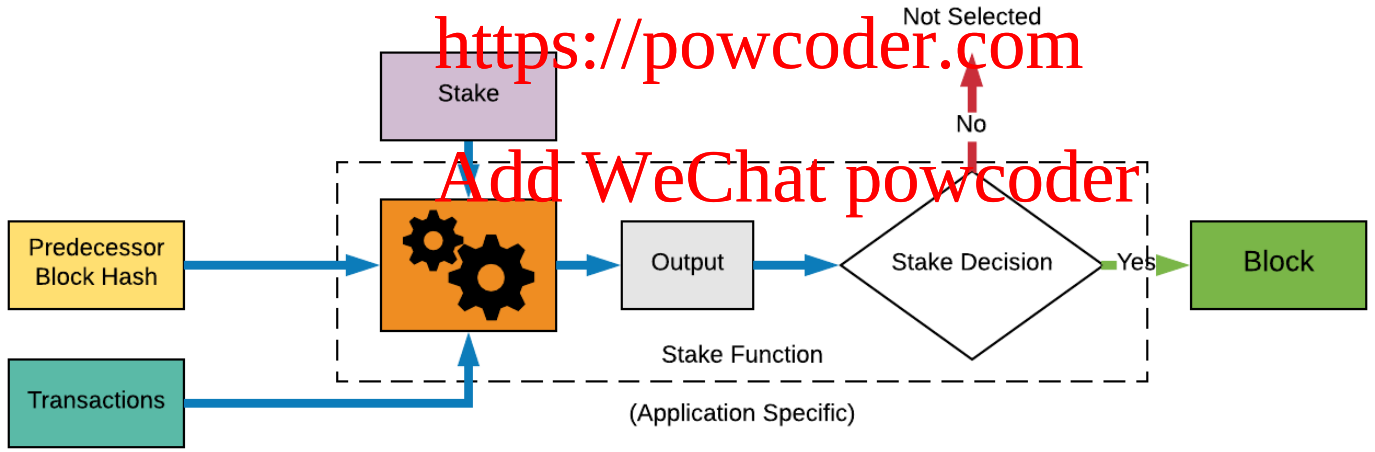
Recap: Byzantine agreement (BA)

Byzantine agreement (BA) systems, also known as Byzantine fault tolerant (BFT) systems, provide solutions to the Byzantine generals problem.

Recap: Proof-of-Stake (PoS)

There are different types of PoS, depending on the "things" put at stake:

- Proof-of-Lock/Deposit
- Balance-based
- …



Nakamoto-style consensus: e.g., Ouroboros
BFT-style consensus: e.g., Algorand

Algorand

- Idea: Combine PoS with BFT-style consensus in order to be able to confirm transactions faster.

Algorand

- **Idea: Combine PoS with BFT-style consensus in order to be able to confirm transactions faster.**

- BFT solutions can only tolerate less than 1/3 of the participants being malicious.

Algorand

- **Idea: Combine PoS with BFT-style consensus in order to be able to confirm transactions faster.**

- BFT solutions can only tolerate less than 1/3 of the participants being malicious.

- **This translates into Algorand only being able to tolerate less than 1/3 of the coins being controlled by the adversary.**

Scalability

- Running a BFT protocol among millions of users would not be practical.

Scalability

- Running a BFT protocol among millions of users would not be practical.

- **Consensus by committee: use a representative committee (with thousands of votes) for reaching consensus.**

Scalability

- Running a BFT protocol among millions of users would not be practical.

- **Consensus by committee: use a representative committee (with thousands of votes) for reaching consensus.**

- Users are randomly selected to participate in the committee, with their expected number of votes in the committee proportional to the amount of coins that they control.

Scalability

- Running a BFT protocol among millions of users would not be practical.
- **Consensus by committee: use a representative committee (with thousands of votes) for reaching consensus.**
- Users are randomly selected to participate in the committee, with their expected number of votes in the committee proportional to the amount of coins that they control.
- Challenge: relying on a committee creates the possibility of targeted attacks against the chosen committee members.

BA⋆

- New Byzantine Agreement protocol called BA⋆ that scales to many users.

BA⋆

- New Byzantine Agreement protocol called BA⋆ that scales to many users.

- BA⋆ is executed to reach consensus among participants on the next set of transactions.

BA⋆

- New Byzantine Agreement protocol called BA⋆ that scales to many users.
- BA⋆ is executed to reach consensus among participants on the next set of transactions.
- Reach consensus on a new block with low latency and without the possibility of forks.

BA⋆

- Scalability: mechanism based on VRFs allows users to privately and non-interactively check whether they are selected to participate in BA⋆ and to include a proof of their selection in their network messages.

BA⋆

- Scalability: mechanism based on VRFs allows users to privately and non-interactively check whether they are selected to participate in BA⋆ and to include a proof of their selection in their network messages.

- **Player Replaceability:** users do not keep any private state except private keys, which allows Algorand to replace participants immediately after they send a message. This mitigates targeted attacks on chosen participants after their identity is revealed.

BA⋆

- Scalability: mechanism based on VRFs allows users to privately and non-interactively check whether they are selected to participate in BA⋆ and to include a proof of their selection in their network messages.

- **Player Replaceability:** users do not keep any private state except private keys, which allows Algorand to replace participants immediately after they send a message. This mitigates targeted attacks on chosen participants after their identity is revealed.

- Once a committee member sends his message (exposing his identity to an adversary), the committee member becomes irrelevant to BA⋆.

BA⋆

- Scalability: mechanism based on VRFs allows users to privately and non-interactively check whether they are selected to participate in BA⋆ and to include a proof of their selection in their network messages.

- **Player Replaceability:** users do not keep any private state except private keys, which allows Algorand to replace participants immediately after they send a message. This mitigates targeted attacks on chosen participants after their identity is revealed.

- Once a committee member sends his message (exposing his identity to an adversary), the committee member becomes irrelevant to BA⋆.

- Having no private state makes all users equally capable of participating. Elect new committee members for each step of BA⋆.

Model

- To achieve liveness, Algorand makes the “strong synchrony” assumption that most honest users (e.g., 95%) can send messages that will be received by most other honest users (e.g., 95%) within a known time bound.

Model

- To achieve liveness, Algorand makes the “strong synchrony” assumption that most honest users (e.g., 95%) can send messages that will be received by most other honest users (e.g., 95%) within a known time bound.

- The adversary can control the network of a few honest users, but does not allow the adversary to manipulate the network at a large scale, and does not allow network partitions.

Model

- To achieve liveness, Algorand makes the “strong synchrony” assumption that most honest users (e.g., 95%) can send messages that will be received by most other honest users (e.g., 95%) within a known time bound.

- The adversary can control the network of a few honest users, but does not allow the adversary to manipulate the network at a large scale, and does not allow network partitions.

- Algorand achieves safety with a “weak synchrony” assumption: the network can be asynchronous (i.e., entirely controlled by the adversary) for a long but bounded period of time (e.g., at most 1 day or 1 week). After an asynchrony period, the network must be strongly synchronous for a reasonably long period again.

Model

- To achieve liveness, Algorand makes the “strong synchrony” assumption that most honest users (e.g., 95%) can send messages that will be received by most other honest users (e.g., 95%) within a known time bound.

- The adversary can control the network of a few honest users, but does not allow the adversary to manipulate the network at a large scale, and does not allow network partitions.

- Algorand achieves safety with a “weak synchrony” assumption: the network can be asynchronous (i.e., entirely controlled by the adversary) for a long but bounded period of time (e.g., at most 1 day or 1 week). After an asynchrony period, the network must be strongly synchronous for a reasonably long period again.

- Loosely synchronised clocks across all users in order to recover liveness after weak synchrony.

Algorand vs Ouroboros

- Let f denote the fraction of the total coins that is controlled by the adversary.

Algorand vs Ouroboros

- Let f denote the fraction of the total coins that is controlled by the adversary.

- Ouroboros is secure against richer adversaries that control any fraction $f < 1/2$. Algorand would already not work for $1/3 \leq f < 1/2$.

Algorand vs Ouroboros

- Let f denote the fraction of the total coins that is controlled by the adversary.
- Ouroboros is secure against richer adversaries that control any fraction $f < 1/2$. Algorand would already not work for $1/3 \leq f < 1/2$.
- The transactions in Algorand can be confirmed much faster due to the properties of BFT protocols.

Algorand vs Ouroboros

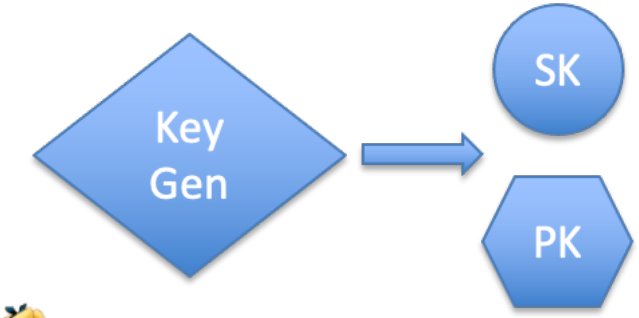
- Let f denote the fraction of the total coins that is controlled by the adversary.
- Ouroboros is secure against richer adversaries that control any fraction $f < 1/2$. Algorand would already not work for $1/3 \leq f < 1/2$.
- The transactions in Algorand can be confirmed much faster due to the properties of BFT protocols.
- Both Algorand and Ouroboros use VRFs as an essential building block.

Recap: Verifiable Random Function (VRF)

- High-level idea: A PRF that can be publicly verified.

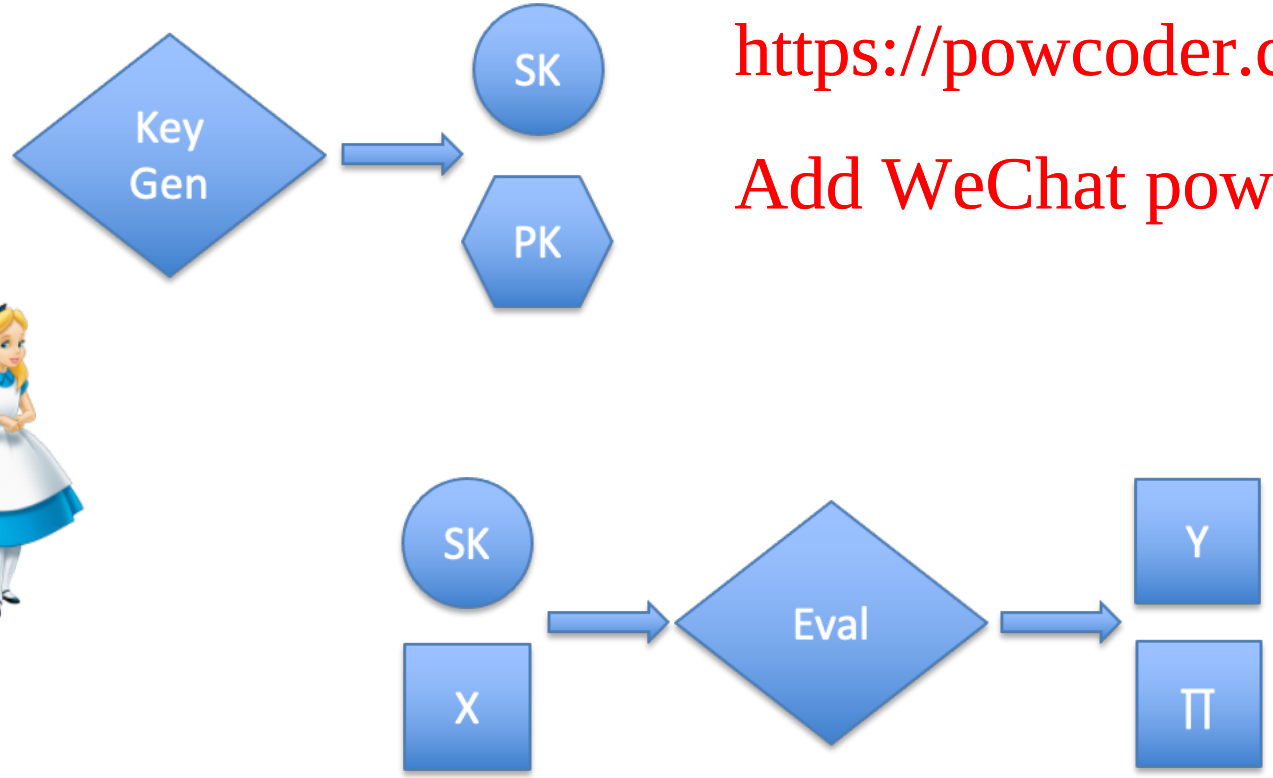
Recap: Verifiable Random Function (VRF)

- High-level idea: A PRF that can be publicly verified.



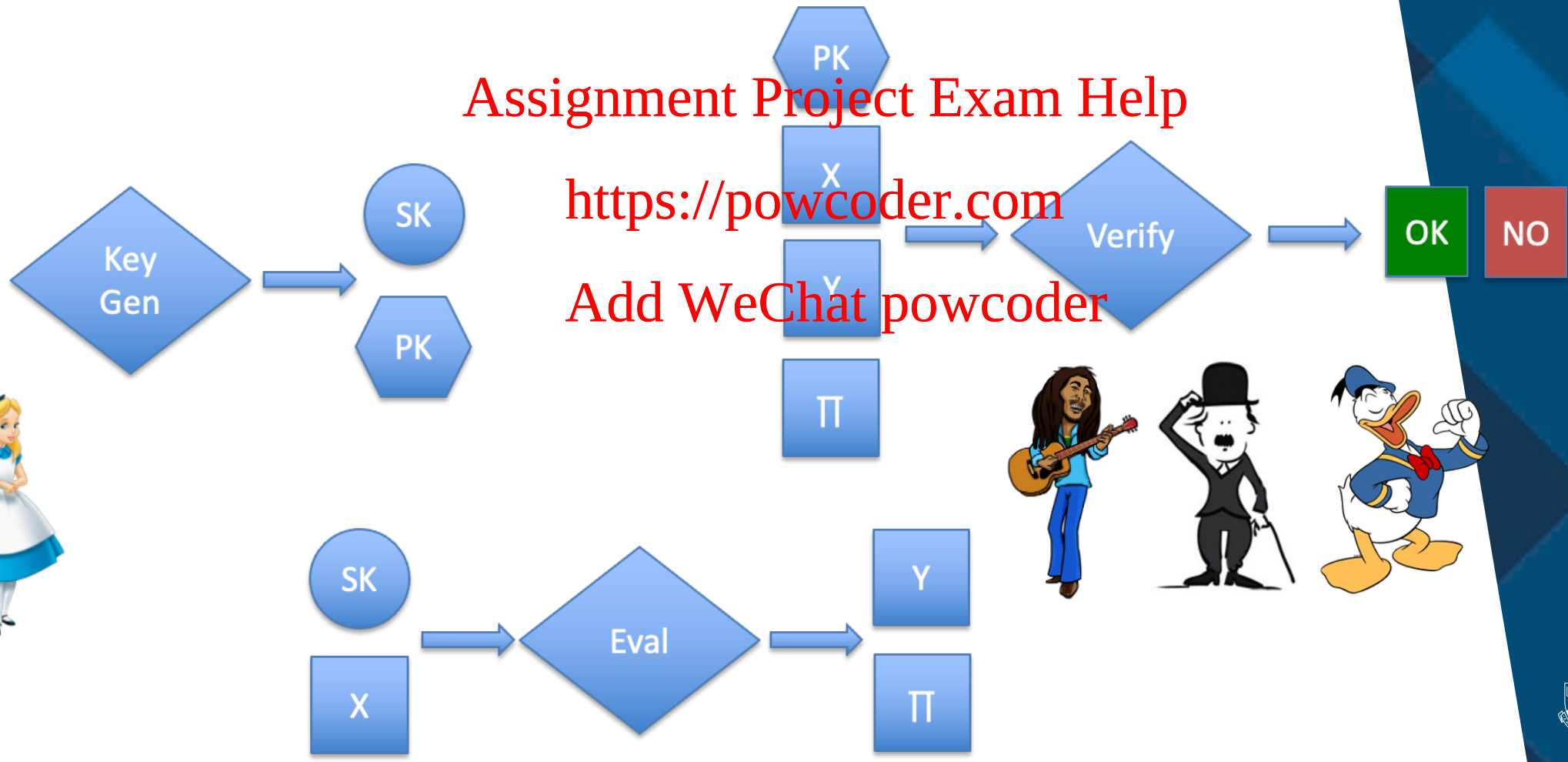
Recap: Verifiable Random Function (VRF)

- High-level idea: A PRF that can be publicly verified.



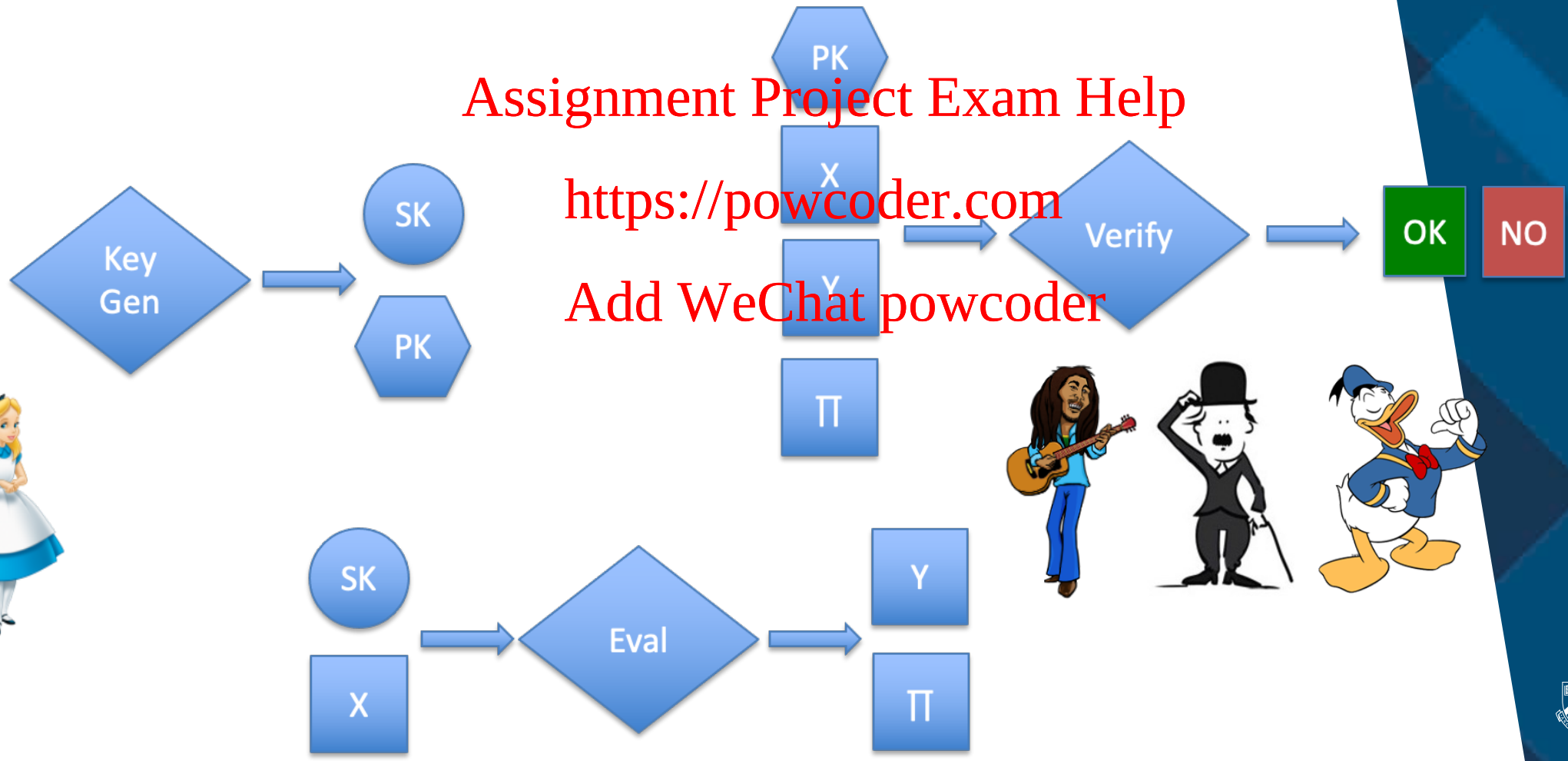
Recap: Verifiable Random Function (VRF)

- High-level idea: A PRF that can be publicly verified.



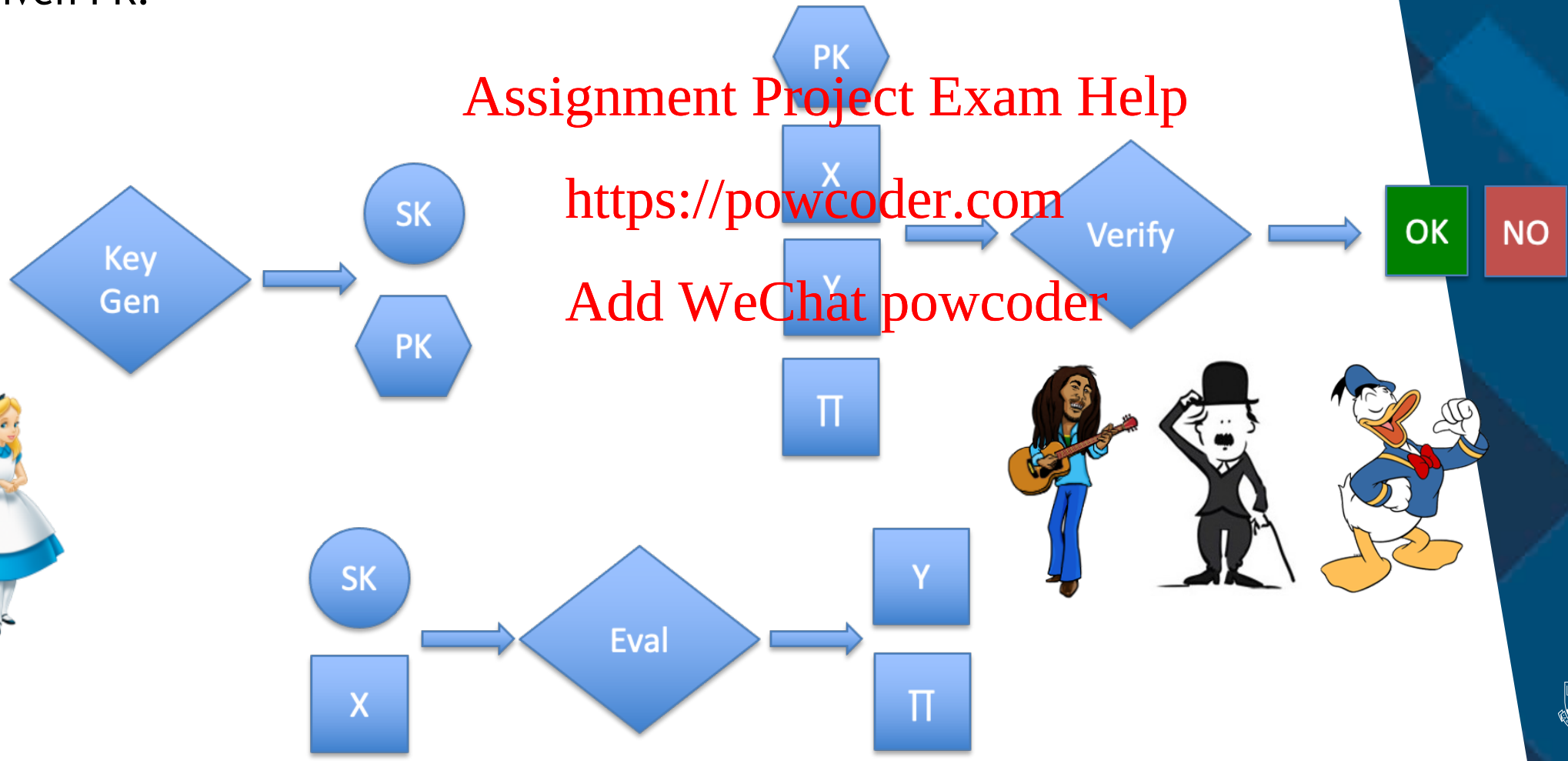
Recap: Verifiable Random Function (VRF)

- Security properties of VRF: pseudorandomness, provability and uniqueness.



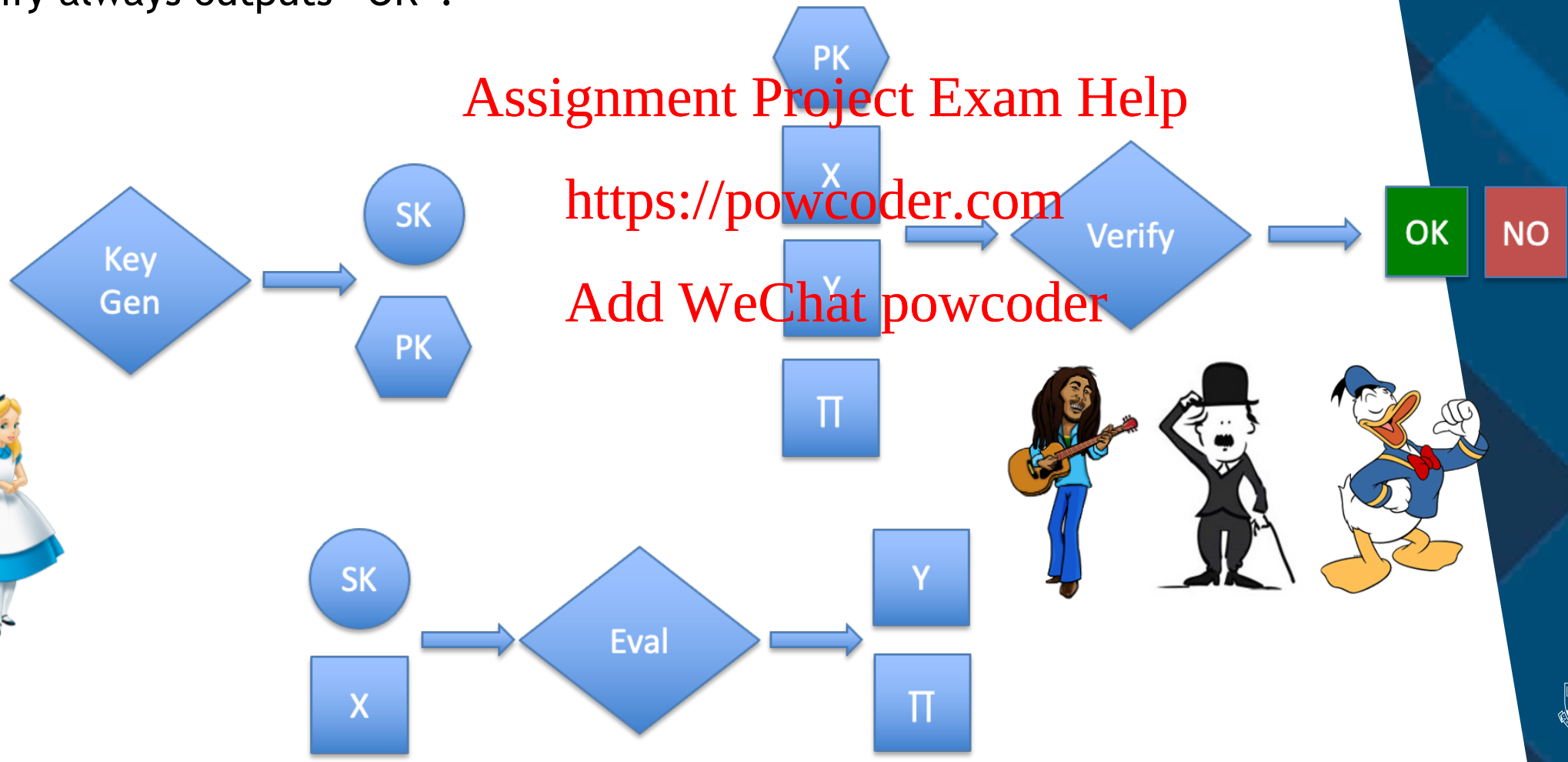
Recap: Verifiable Random Function (VRF)

- Pseudorandomness: the output Y is pseudorandom, even when the adversary is given PK.



Recap: Verifiable Random Function (VRF)

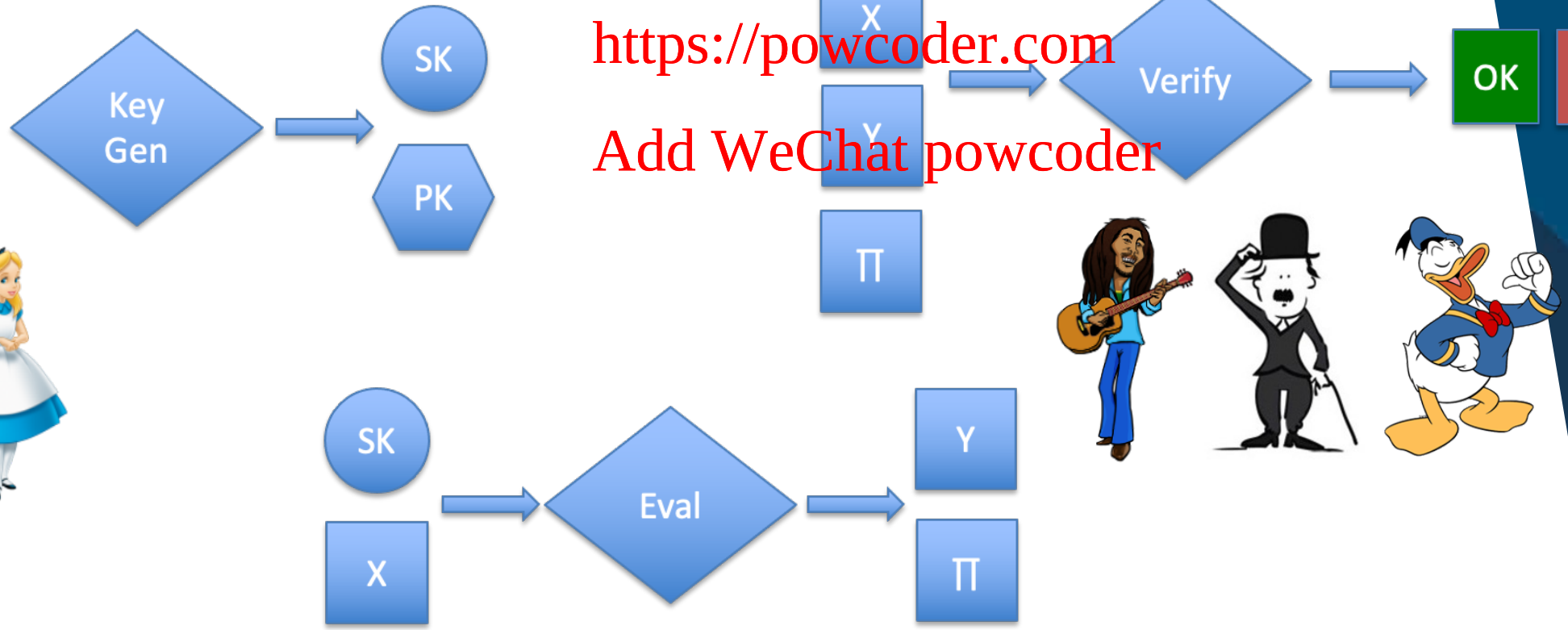
- Provability: For honestly generated keys, input X, output Y and proof ∏, Verify always outputs “OK”.



MONASH University

Recap: Verifiable Random Function (VRF)

- Uniqueness: No tuple (PK, X, Y_0, Π_0, Y_1, Π_1) such that Verify(PK, X, Y_0,Π_0)=OK, Verify(PK, X, Y_1, Π_1)=OK and $Y_0 \neq Y_1$.



VRF in Algorand

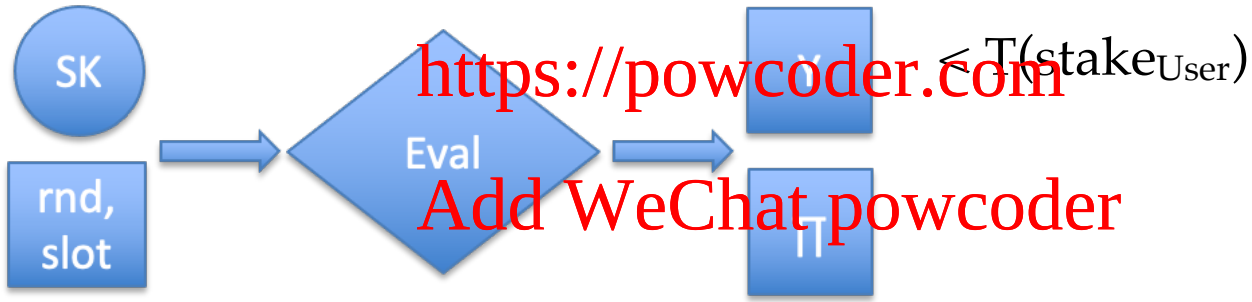
- Used in a local, private lottery for election of block proposers and committee members.

VRF in Algorand

- Used in a local, private lottery for election of block proposers and committee members.
- VRF should be secure even in the case of maliciously generated keys.

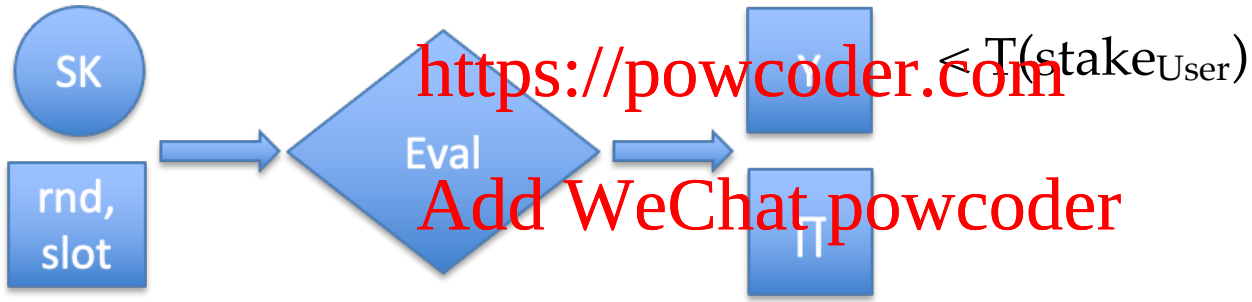
VRF in Algorand

- Used in a local, private lottery for election of block proposers and committee members.
- VRF should be secure even in the case of maliciously generated keys.



VRF in Algorand

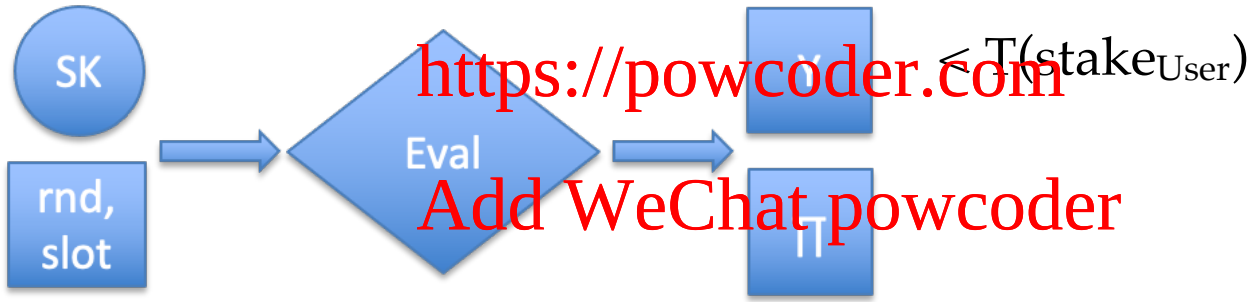
- Used in a local, private lottery for election of block proposers and committee members.
- VRF should be secure even in the case of maliciously generated keys.



- **Using a private lottery helps mitigating adaptive attacks.**

VRF in Algorand

- Used in a local, private lottery for election of block proposers and committee members.
- VRF should be secure even in the case of maliciously generated keys.



- **Using a private lottery helps mitigating adaptive attacks.**
- When a user claims to be a block proposer/committee member, anyone can publicly verify if that claim is legitimate or not.

BA⋆ Consensus

- Executes in steps, communicating using a gossip protocol, and produces a new agreed-upon block. BA⋆ can produce two kinds of consensus: final consensus and tentative consensus.

BA⋆ Consensus

- Executes in steps, communicating using a gossip protocol, and produces a new agreed-upon block. BA⋆ can produce two kinds of consensus: final consensus and tentative consensus.

- If one user reaches final consensus, then any other user that reaches final or tentative consensus in the same round must agree on the same block value (regardless of whether the strong synchrony assumption held). This ensures safety as all future transactions will be chained to this final block.

BA⋆ Consensus

- Executes in steps, communicating using a gossip protocol, and produces a new agreed-upon block. BA⋆ can produce two kinds of consensus: final consensus and tentative consensus.

- If one user reaches final consensus, then any other user that reaches final or tentative consensus in the same round must agree on the same block value (regardless of whether the strong synchrony assumption held). This ensures safety as all future transactions will be chained to this final block.

- A transaction is confirmed when the transaction's block (or any of its successor blocks) reaches final consensus.

Block Proposal

- All users execute cryptographic sortition (using the VRF) to determine if they are selected to propose a block in a given round.

Block Proposal

- All users execute cryptographic sortition (using the VRF) to determine if they are selected to propose a block in a given round.

- Each selected user has a priority, which can be compared between users, and a proof of the chosen user's priority.

Block Proposal

- All users execute cryptographic sortition (using the VRF) to determine if they are selected to propose a block in a given round.
- Each selected user has a priority, which can be compared between users, and a proof of the chosen user's priority.
- Since sortition is random, there may be multiple users selected to propose a block, and the priority determines which block everyone should adopt.

Block Proposal

- All users execute cryptographic sortition (using the VRF) to determine if they are selected to propose a block in a given round.
- Each selected user has a priority, which can be compared between users, and a proof of the chosen user's priority.
- Since sortition is random, there may be multiple users selected to propose a block, and the priority determines which block everyone should adopt.
- Each user initialises BA* with the highest-priority block that they received.

BA⋆ Steps

- BA⋆ executes in repeated steps. Each step begins with sortition, where all users check whether they have been selected as committee members in that step.

BA⋆ Steps

- BA⋆ executes in repeated steps. Each step begins with sortition, where all users check whether they have been selected as committee members in that step.

- Committee members broadcast a message for that step (which includes their proof of selection).

BA⋆ Steps

- BA⋆ executes in repeated steps. Each step begins with sortition, where all users check whether they have been selected as committee members in that step.
- Committee members broadcast a message for that step (which includes their proof of selection).
- Steps repeat until, in some step of BA⋆, enough users in the committee reach consensus.

BA⋆ Steps

- BA⋆ executes in repeated steps. Each step begins with sortition, where all users check whether they have been selected as committee members in that step.
- Committee members broadcast a message for that step (which includes their proof of selection).
- Steps repeat until, in some step of BA⋆, enough users in the committee reach consensus.
- Steps are not synchronised across users; each user checks for selection as soon as he observes the previous step had ended.

BA⋆ Phases

- In the first phase, BA⋆ reduces the problem of agreeing on a block to agreement on one of two options (either a specific proposed block hash, or the hash of an empty block).

BA⋆ Phases

- In the first phase, BA⋆ reduces the problem of agreeing on a block to agreement on one of two options (either a specific proposed block hash, or the hash of an empty block).

- In the second phase, BA⋆ reaches agreement on one of these options.

BA⋆ Phases

- In the first phase, BA⋆ reduces the problem of agreeing on a block to agreement on one of two options (either a specific proposed block hash, or the hash of an empty block).
- In the second phase, BA⋆ reaches agreement on one of these options.
- The first phase always takes two steps. The number of steps on the second phase varies.

Votes

- Votes for hashes of blocks, instead of entire block contents (save bandwidth).

Votes

- Votes for hashes of blocks, instead of entire block contents (save bandwidth).

- In each step, every committee member casts a vote/votes for some value, and all users count the votes.

Votes

- Votes for hashes of blocks, instead of entire block contents (save bandwidth).
- In each step, every committee member casts a vote/votes for some value, and all users count the votes.
- Users that receive more than a threshold of votes for some value will vote for that value in the next step (if selected as a committee member).

Votes

- Votes for hashes of blocks, instead of entire block contents (save bandwidth).
- In each step, every committee member casts a vote/votes for some value, and all users count the votes.
- Users that receive more than a threshold of votes for some value will vote for that value in the next step (if selected as a committee member).
- In the common case when the network is strongly synchronous and the highest-priority block proposer was honest, most votes will be for the same proposed block hash.

Votes

- Votes for hashes of blocks, instead of entire block contents (save bandwidth).
- In each step, every committee member casts a vote/votes for some value, and all users count the votes.
- Users that receive more than a threshold of votes for some value will vote for that value in the next step (if selected as a committee member).
- In the common case when the network is strongly synchronous and the highest-priority block proposer was honest, most votes will be for the same proposed block hash.
- If the users do not receive enough votes for any value, they time out, and their choice of vote for the next step is determined by the step number.

Votes

- A user U may receive votes that push the votes observed by it past the threshold for consensus. But as other users might still had timed out in that step, the user U keeps voting for the consensus value in the next 3 steps.

Votes

- A user U may receive votes that push the votes observed by it past the threshold for consensus. But as other users might still had timed out in that step, the user U keeps voting for the consensus value in the next 3 steps.
- This ensures consensus in a strongly synchronous network.

Votes

- A user U may receive votes that push the votes observed by it past the threshold for consensus. But as other users might still had timed out in that step, the user U keeps voting for the consensus value in the next 3 steps.

- This ensures consensus in a strongly synchronous network.

- **Tentative consensus: If the network is not strongly synchronous (e.g., there is a partition)/proposer is malicious, some users may return consensus on the hash of a proposed block, while others return the hash of the empty block.**

Votes

- A user U may receive votes that push the votes observed by it past the threshold for consensus. But as other users might still had timed out in that step, the user U keeps voting for the consensus value in the next 3 steps.

- This ensures consensus in a strongly synchronous network.

- **Tentative consensus: If the network is not strongly synchronous (e.g., there is a partition)/proposer is malicious, some users may return consensus on the hash of a proposed block, while others return the hash of the empty block.**

- **Final consensus: a consensus on a value V is final only if it reached the very first step of the second phase, and if enough users observed this consensus being reached.**

Getting Unstuck

- If the honest users are split into two groups neither of which is large enough to gather enough votes on their own (but large enough together with the adversary's votes), they can get stuck with one group voting for the hash of a proposed block, and another for the empty hash (by being manipulated by the adversary).

Getting Unstuck

- If the honest users are split into two groups neither of which is large enough to gather enough votes on their own (but large enough together with the adversary's votes), they can get stuck with one group voting for the hash of a proposed block, and another for the empty hash (by being manipulated by the adversary).

- Avoid this attack by pushing, in certain steps of BA, towards accepting one of the options based on a random "common coin" (i.e., a binary value that is predominantly the same for all users). Although this may sound circular, the users need not reach formal consensus on this common coin.

Getting Unstuck

- If the honest users are split into two groups neither of which is large enough to gather enough votes on their own (but large enough together with the adversary's votes), they can get stuck with one group voting for the hash of a proposed block, and another for the empty hash (by being manipulated by the adversary).

- Avoid this attack by pushing, in certain steps of BA, towards accepting one of the options based on a random "common coin" (i.e., a binary value that is predominantly the same for all users). Although this may sound circular, the users need not reach formal consensus on this common coin.

- As long as enough users observe the same coin (and it is unknown to the attacker in advance of the step), consensus is reached in next iteration with probability 1/2.

BA⋆ Efficiency

- When the network is strongly synchronous, BA⋆ guarantees that if all honest users start with the same initial block (i.e., the highest priority block proposer was honest), then BA⋆ establishes final consensus over that block and terminates precisely in 4 interactive steps between users.

BA⋆ Efficiency

- When the network is strongly synchronous, BA⋆ guarantees that if all honest users start with the same initial block (i.e., the highest priority block proposer was honest), then BA⋆ establishes final consensus over that block and terminates precisely in 4 interactive steps between users.

- Under the same network conditions, and in the worst case of a particularly lucky adversary, all honest users reach consensus on the next block within expected 13 steps.

Reading

Reading material:
Algorand: Scaling Byzantine Agreements for Cryptocurrencies

Further information about Algorand:
https://www.algorand.com/technology/white-papers

Next Week

- Blockchain Network