Checkpoint Verification

On the Sui network, checkpoints define the history of the blockchain. They are quite similar to the concept of blocks used by traditional blockchains like Bitcoin or Ethereum. The Sui blockchain, however, forms checkpoints after transaction execution has already happened to provide a certified history of the chain, instead of being formed before execution.

Checkpoints contain:

Both validators and Full nodes consume checkpoints to remain synchronized with the network.

For Full nodes and validators to trust a checkpoint, they must first verify it. Verification ensures that the checkpoint is a true checkpoint that the Sui validator committee created.

Checkpoint verification requires two interdependent pieces:

Assuming that the Full node (or other client) has the public keys of the validator committee that created the checkpoint, it can check the signatures on the checkpoint for validity.

Checkpoints are signed by the aggregated BLS signatures of a quorum of the committee. If the signatures are valid, the client now knows that the checkpoint was created by the validator committee, and not by some other party.

By validating checkpoints, the client can determine the make-up of the committee, because the final checkpoint of each epoch contains the validator committee (including the public keys) of the next epoch.

These pieces seem to create a circular dependency issue. The client needs to know the committee to verify checkpoints, which in turn allows it to learn what the committee is for each epoch. To solve this problem, the process is bootstrapped by starting from the genesis checkpoint, which is the earliest checkpoint in a Sui network. The genesis checkpoint contains the initial validator committee, which allows a client to verify all checkpoints in the history by using the following process:

This allows the client to eventually verify all checkpoints up to the present time.

After a client verifies a checkpoint, what can it do with that information?

As mentioned earlier, a checkpoint contains a list of transactions so a Full node, for instance, can begin fetching and executing those transactions. Because the transactions are identified by their digest (a cryptographic hash), the client can be sure that the transactions it executes have not been altered.

Additionally, the checkpoint contains the effects digests of each transaction. The effects digest is the cryptographic hash of the TransactionEffects , which is itself a structure that lists all of the inputs and outputs of a transaction. It includes the digests of all objects that were written by the transaction. This allows a Full node to verify that it has obtained the same execution results as those that the validators attested to when signing the checkpoint.

By executing checkpoints, and verifying transaction outputs, a Full node can build up the entire state of the Sui network (that is, the collection of objects in the network) and trust that every byte of every object is correct.

Checkpoint verification

For Full nodes and validators to trust a checkpoint, they must first verify it. Verification ensures that the checkpoint is a true checkpoint that the Sui validator committee created.

Checkpoint verification requires two interdependent pieces:

Assuming that the Full node (or other client) has the public keys of the validator committee that created the checkpoint, it can check the signatures on the checkpoint for validity.

Checkpoints are signed by the aggregated BLS signatures of a quorum of the committee. If the signatures are valid, the client now knows that the checkpoint was created by the validator committee, and not by some other party.

By validating checkpoints, the client can determine the make-up of the committee, because the final checkpoint of each epoch contains the validator committee (including the public keys) of the next epoch.

These pieces seem to create a circular dependency issue. The client needs to know the committee to verify checkpoints, which in turn allows it to learn what the committee is for each epoch. To solve this problem, the process is bootstrapped by starting from the

genesis checkpoint, which is the earliest checkpoint in a Sui network. The genesis checkpoint contains the initial validator committee, which allows a client to verify all checkpoints in the history by using the following process:

This allows the client to eventually verify all checkpoints up to the present time.

After a client verifies a checkpoint, what can it do with that information?

As mentioned earlier, a checkpoint contains a list of transactions so a Full node, for instance, can begin fetching and executing those transactions. Because the transactions are identified by their digest (a cryptographic hash), the client can be sure that the transactions it executes have not been altered.

Additionally, the checkpoint contains the effects digests of each transaction. The effects digest is the cryptographic hash of the TransactionEffects , which is itself a structure that lists all of the inputs and outputs of a transaction. It includes the digests of all objects that were written by the transaction. This allows a Full node to verify that it has obtained the same execution results as those that the validators attested to when signing the checkpoint.

By executing checkpoints, and verifying transaction outputs, a Full node can build up the entire state of the Sui network (that is, the collection of objects in the network) and trust that every byte of every object is correct.

What do checkpoints commit to?

After a client verifies a checkpoint, what can it do with that information?

As mentioned earlier, a checkpoint contains a list of transactions so a Full node, for instance, can begin fetching and executing those transactions. Because the transactions are identified by their digest (a cryptographic hash), the client can be sure that the transactions it executes have not been altered.

Additionally, the checkpoint contains the effects digests of each transaction. The effects digest is the cryptographic hash of the TransactionEffects , which is itself a structure that lists all of the inputs and outputs of a transaction. It includes the digests of all objects that were written by the transaction. This allows a Full node to verify that it has obtained the same execution results as those that the validators attested to when signing the checkpoint.

By executing checkpoints, and verifying transaction outputs, a Full node can build up the entire state of the Sui network (that is, the collection of objects in the network) and trust that every byte of every object is correct.