DSL

UCSB

Database and Distributed Computing Fundamentals of Blockchains

DSL at UCSB

Sujaya Maiyya, Victor Zakhary, Divyakant Agrawal, Amr El Abbadi

Traditional Banking Systems

DSL at UCSB

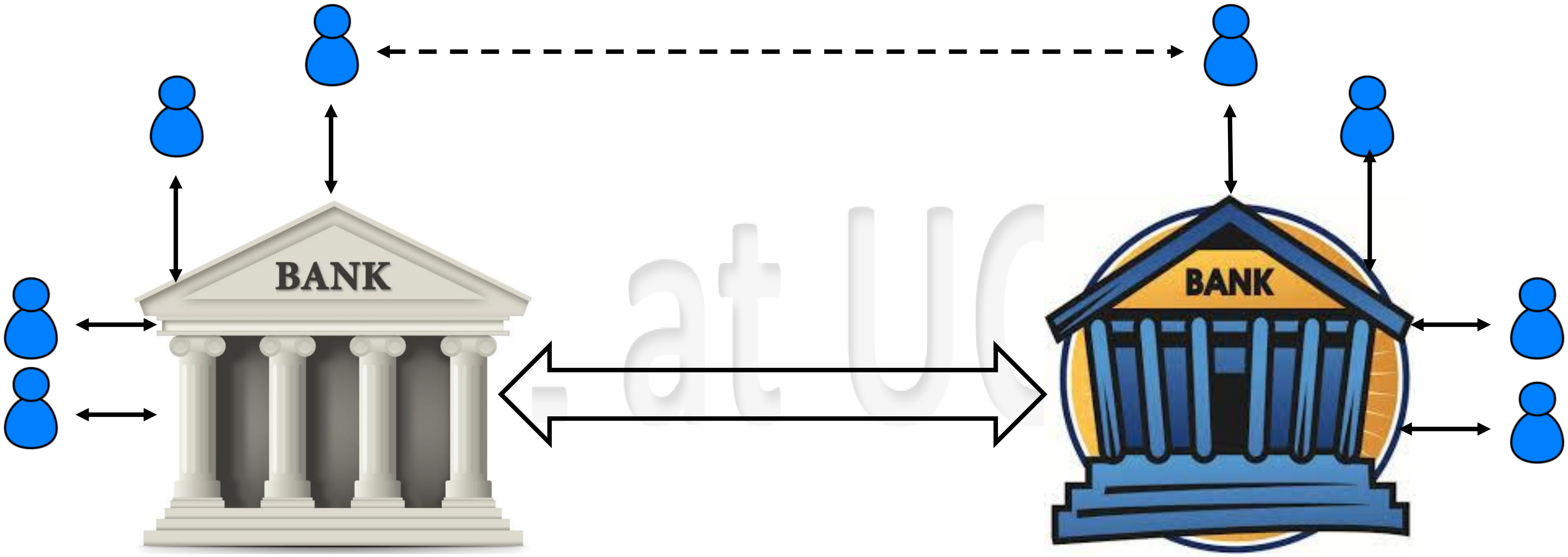
Traditional Banking Systems

Traditional Banking Systems

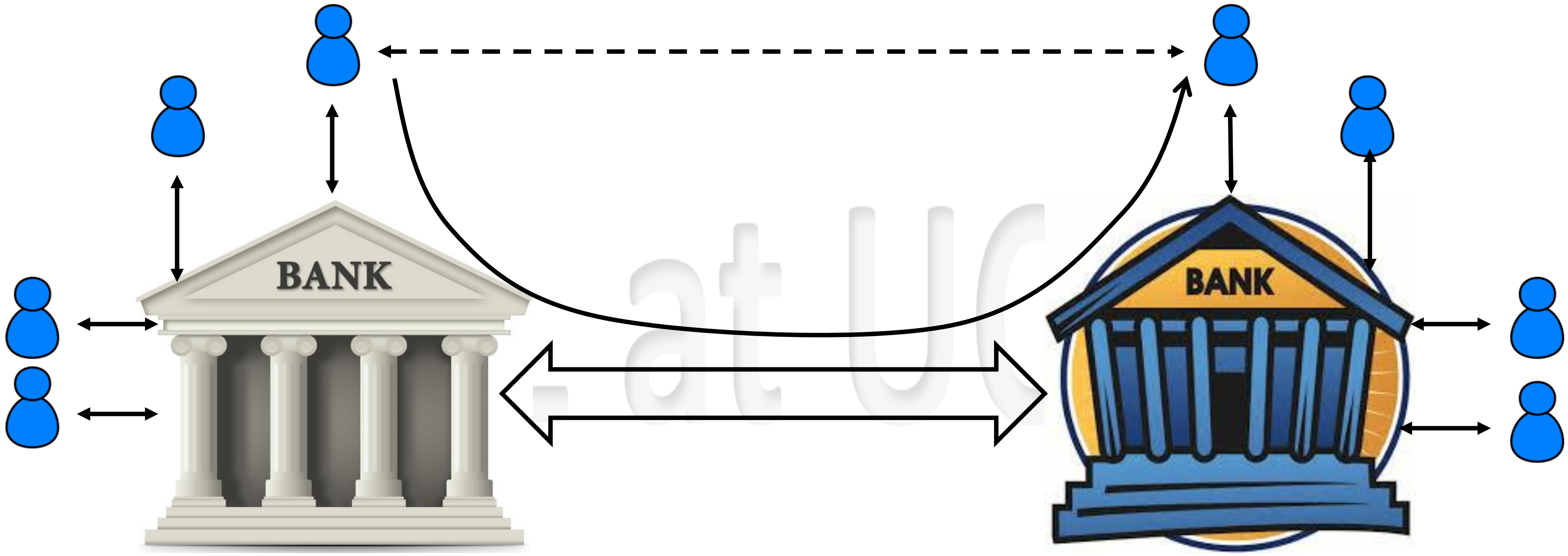
Traditional Banking Systems

Traditional Banking Systems

Traditional Banking Systems



Traditional Banking Systems



Traditional Banking Systems

- From Database and Distributed Computing Perspective

DSL at UCSB

Traditional Banking Systems

- From Database and Distributed Computing Perspective
- Identities and Signatures

DSL at UCSB

Traditional Banking Systems

- From Database and Distributed Computing Perspective
- Identities and Signatures
  - You are your signature [ID, username and password]

DSL at UCSB

Traditional Banking Systems

- From Database and Distributed Computing Perspective
- Identities and Signatures
  - You are your signature [ID, username and password]
- Ledger

DSL at UCSB

Traditional Banking Systems

- From Database and Distributed Computing Perspective
- Identities and Signatures
  - You are your signature [ID, username and password]
- Ledger
  - The balance of each identity (saved in a DB)

DSL at UCSB

Traditional Banking Systems

- From Database and Distributed Computing Perspective
- Identities and Signatures
  - You are your signature [ID, username and password]
- Ledger
  - The balance of each identity (saved in a DB)
- Transactions

DSL at UCSB

Traditional Banking Systems

- From Database and Distributed Computing Perspective
- Identities and Signatures
  - You are your signature [ID, username and password]
- Ledger
  - The balance of each identity (saved in a DB)
- Transactions
  - Move money from one identity to another

DSL at UCSB

Traditional Banking Systems

- From Database and Distributed Computing Perspective
- Identities and Signatures
  - You are your signature [ID, username and password]
- Ledger
  - The balance of each identity (saved in a DB)
- Transactions
  - Move money from one identity to another
  - Concurrency control to serialize transactions (prevent double spending)

DSL at UCSB

Traditional Banking Systems

- From Database and Distributed Computing Perspective
- Identities and Signatures
  - You are your signature [ID, username and password]
- Ledger
  - The balance of each identity (saved in a DB)
- Transactions
  - Move money from one identity to another
  - Concurrency control to serialize transactions (prevent double spending)
  - Typically backed by a transactions log

DSL at UCSB

Traditional Banking Systems

- From Database and Distributed Computing Perspective
- Identities and Signatures
  - You are your signature [ID, username and password]
- Ledger
  - The balance of each identity (saved in a DB)
- Transactions
  - Move money from one identity to another
  - Concurrency control to serialize transactions (prevent double spending)
  - Typically backed by a transactions log
    - Log is persistent

DSL at UCSB

Traditional Banking Systems

- From Database and Distributed Computing Perspective
- Identities and Signatures
  - You are your signature [ID, username and password]
- Ledger
  - The balance of each identity (saved in a DB)
- Transactions
  - Move money from one identity to another
  - Concurrency control to serialize transactions (prevent double spending)
  - Typically backed by a transactions log
    - Log is persistent
    - Log is immutable and tamper-free (end-users trust this)

DSL at UCSB

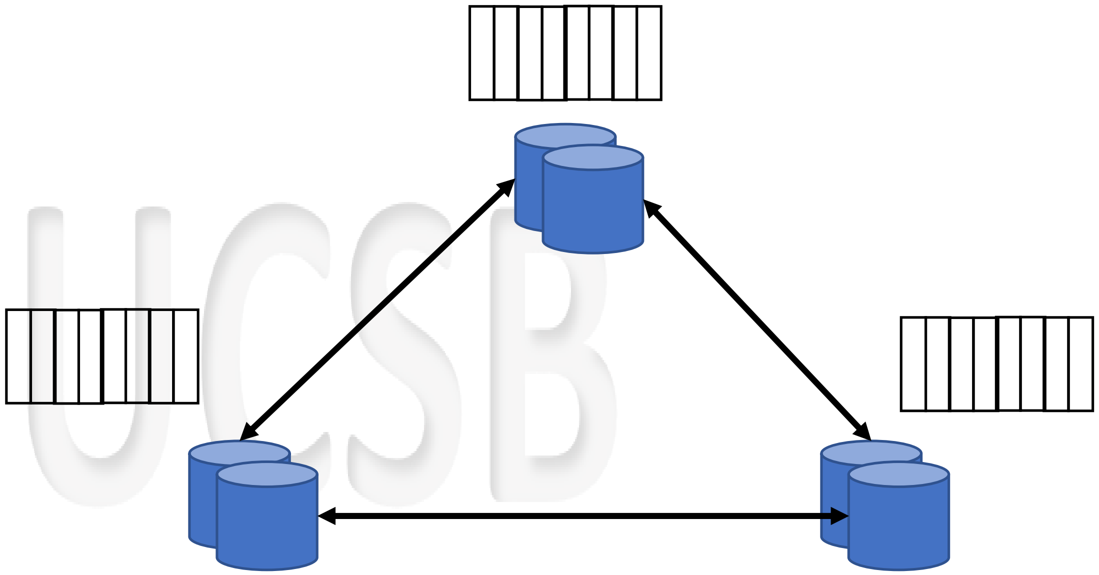
Traditional Banking Systems

DSL at UCSB

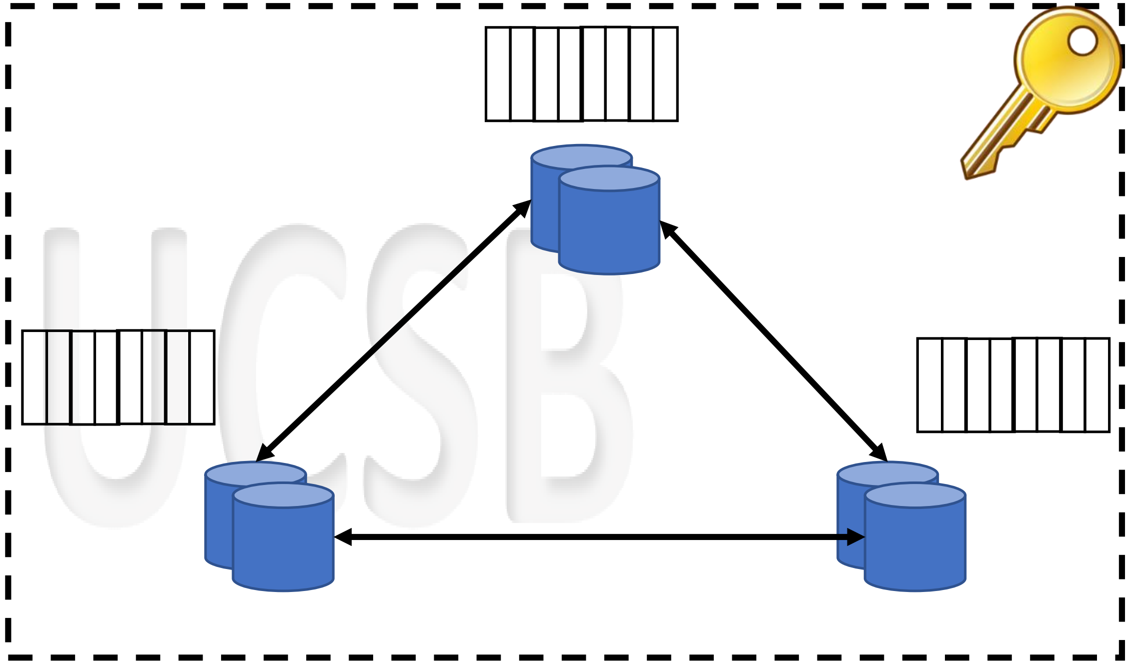
Traditional Banking Systems

... at UCSB

Traditional Banking Systems

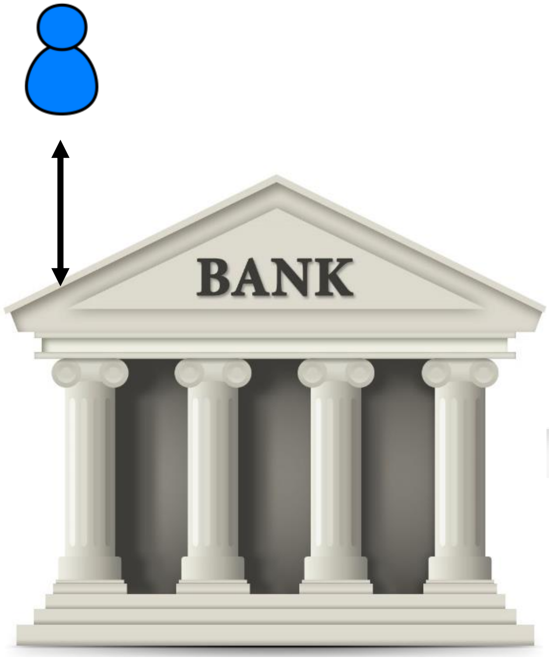


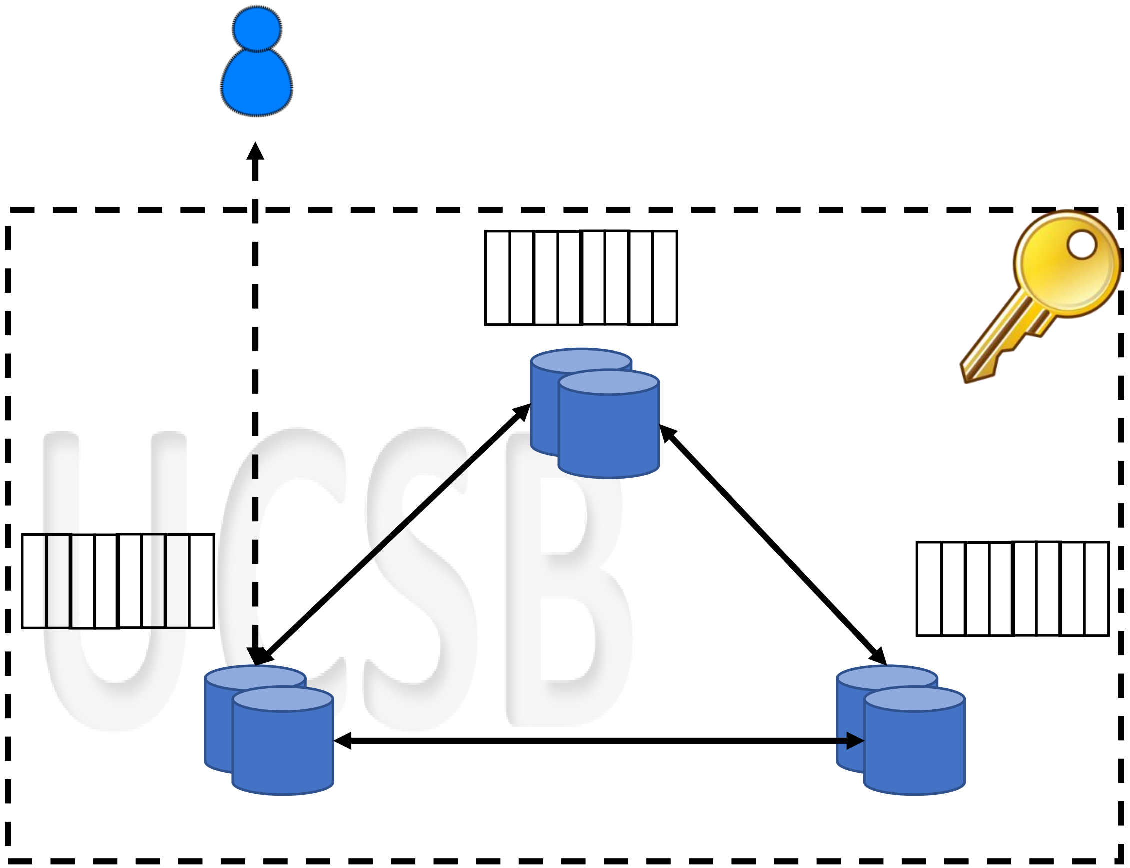
Traditional Banking Systems



- at UCSB

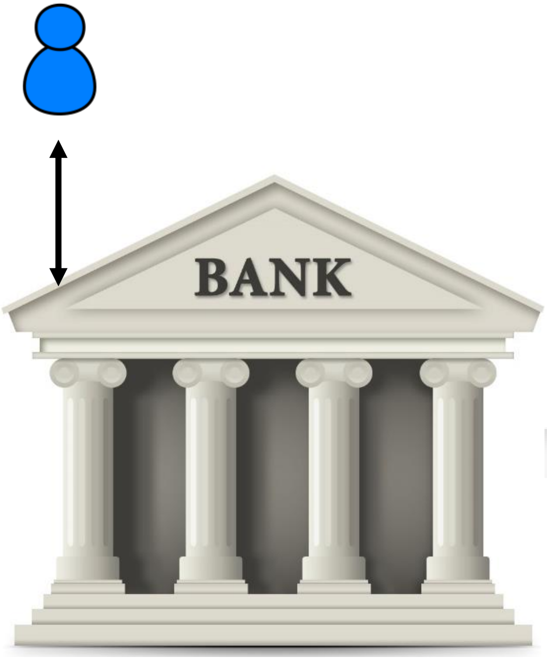
Traditional Banking Systems

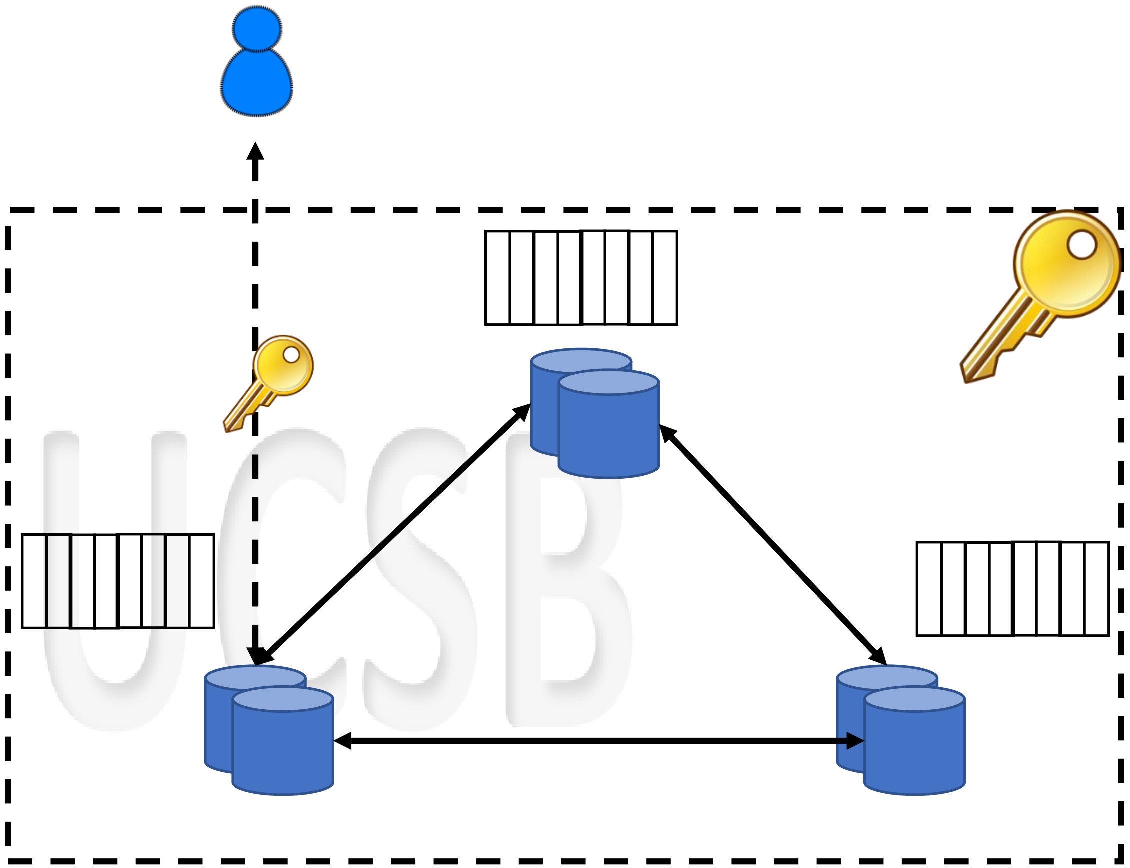




at UCSB

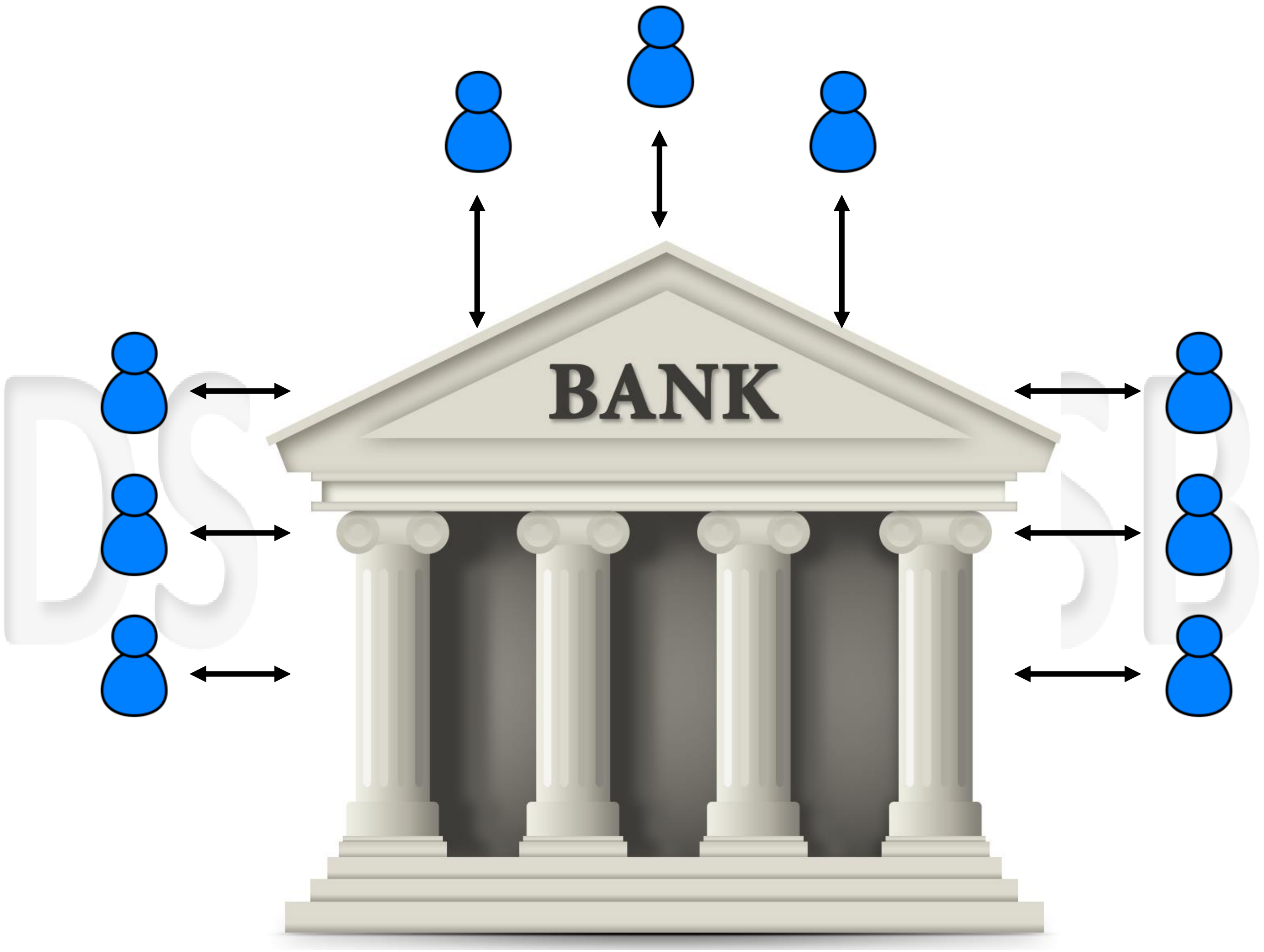
Traditional Banking Systems



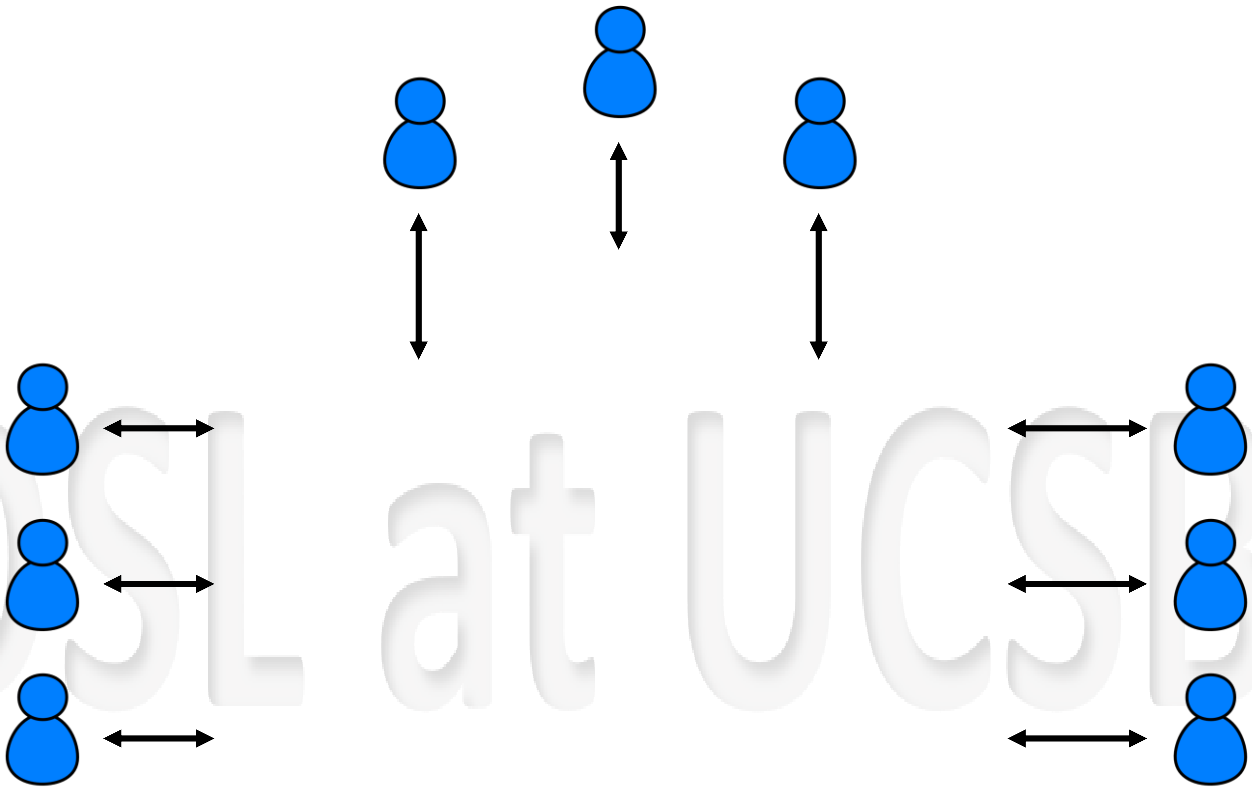


- at UCSB

Bitcoin

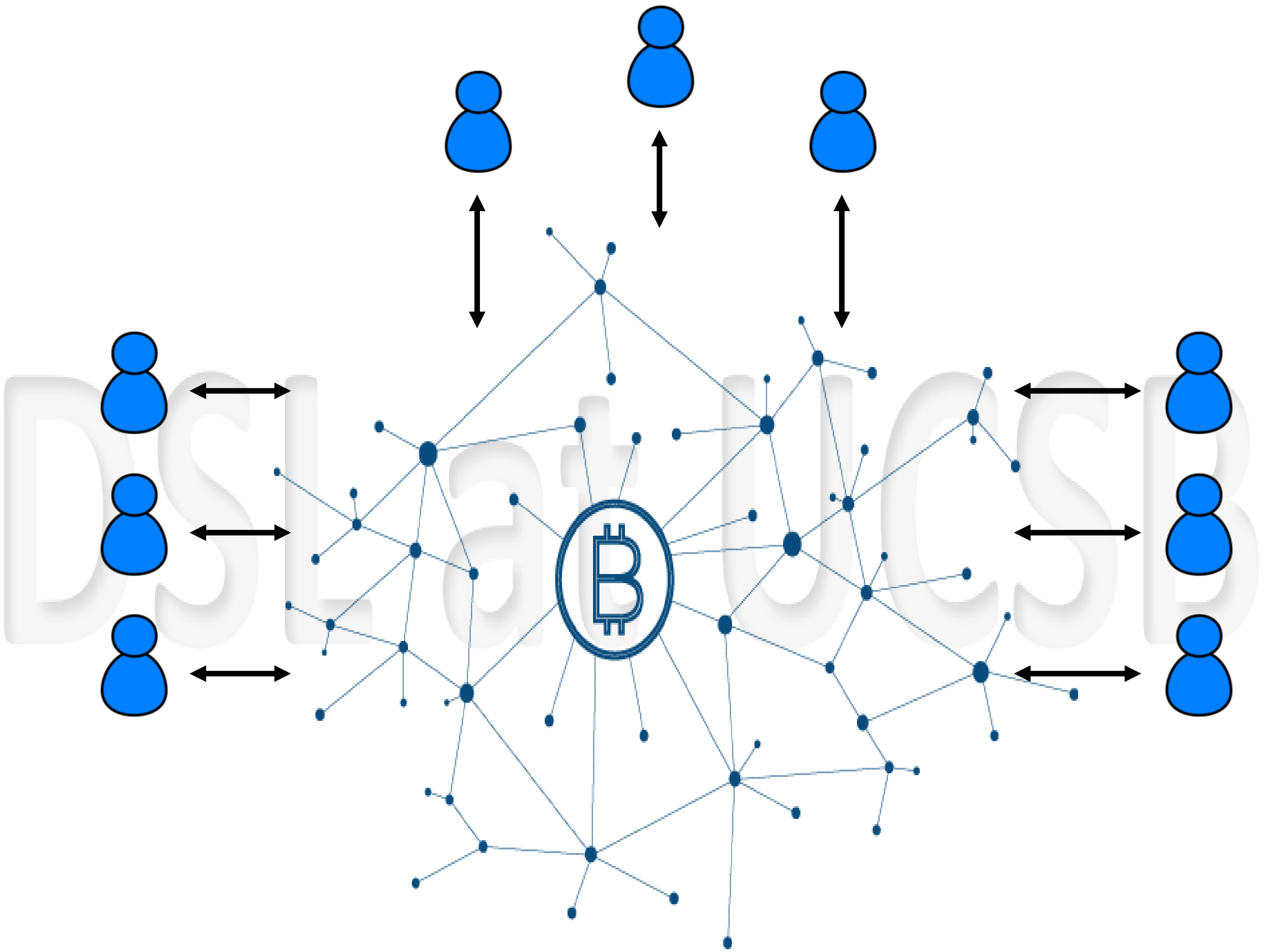


Bitcoin



DSL at UCSB

Bitcoin



Bitcoin: A Peer-to-Peer Electronic Cash System

- From Database and Distributed Computing Perspective
- Identities and Signatures
  - Public/Private key pair
- Ledger
  - The balance of each identity (saved in the blockchain)
- Transactions
  - Move bitcoins from one identity to another
  - Concurrency control to serialize transactions (Mining and PoW)
  - Typically backed by a transactions log (blockchain)
    - Log is persistent (replicated across the network nodes)
    - Log is immutable and tamper-free (PoW and Hash pointers)

DSL at UCSB

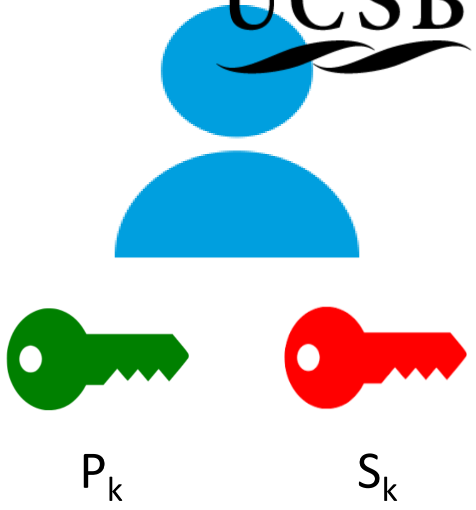
Digital Signatures

UCSB

DSL at UCSB

Digital Signatures

- P_k, S_k ← Keygen(keysize)



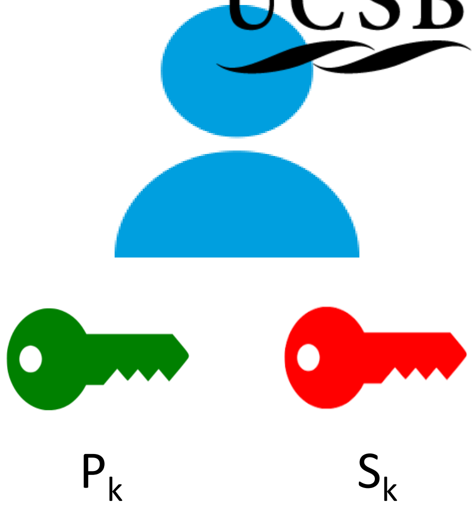
DSL at UCSB

DSL

UCSB

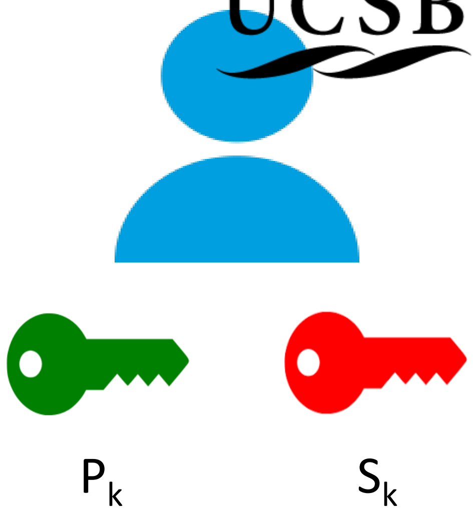
Digital Signatures

- P_k, $S_k \leftarrow$ Keygen(keysize)
- Your P_k is your identity (username, e-mail address)



DSL at UCSB

Digital Signatures

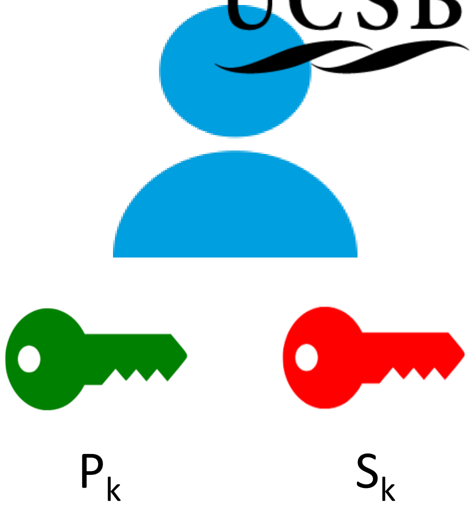


- P_k, S_k ← Keygen(keysize)
- Your P_k is your identity (username, e-mail address)
- Your S_k is your signature (password)
- P_k is made public and used to verify documents signed by S_k
- S_k is private

DSL at UCSB

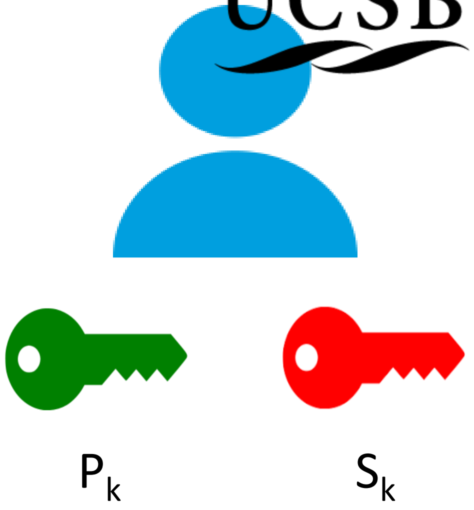
Digital Signatures

- P_k is made public and used to verify documents signed by S_k
- S_k is private

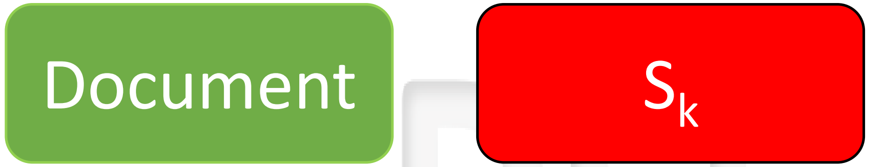


DSL at UCSB

Digital Signatures



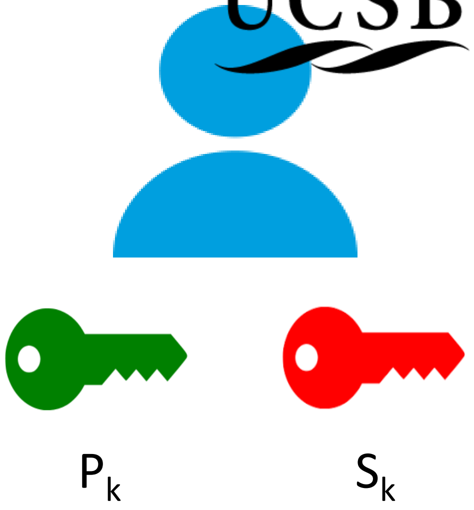
- P_k is made public and used to verify documents signed by S_k
- S_k is private



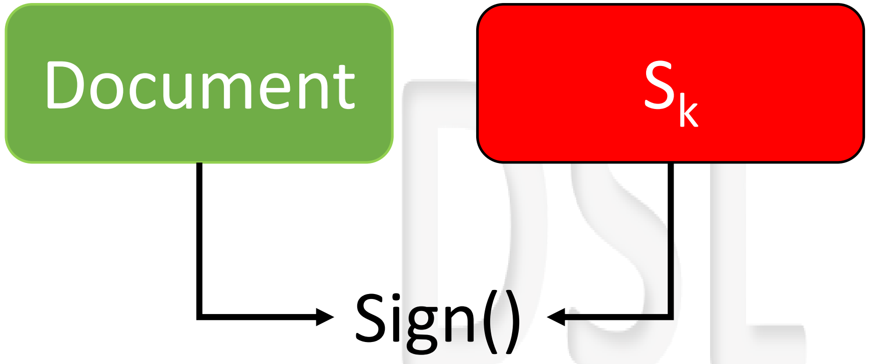
DSL at UCSB

DSL

Digital Signatures



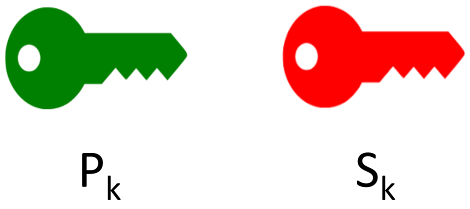
- P_k is made public and used to verify documents signed by S_k
- S_k is private

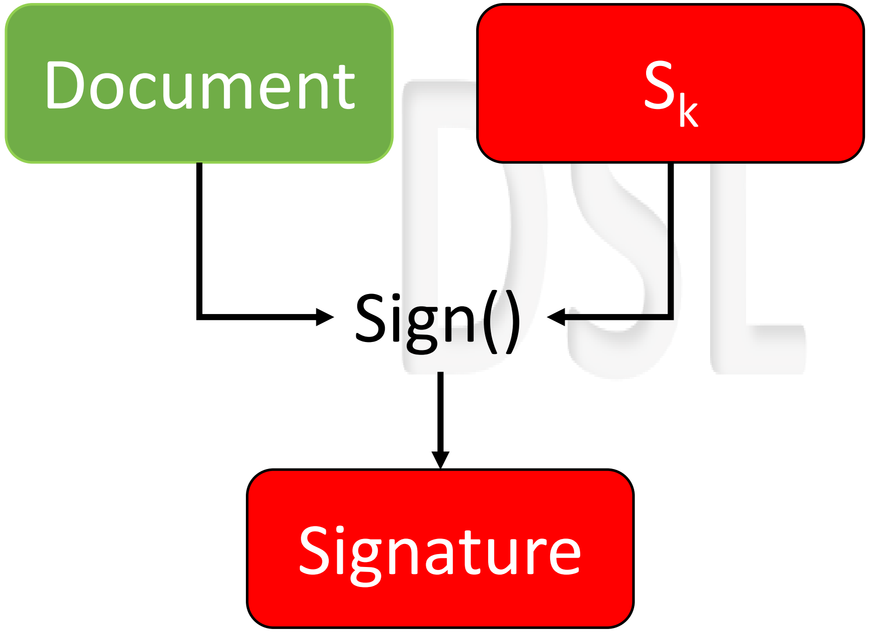


DSL at UCSB

Digital Signatures

- P_k is made public and used to verify documents signed by S_k
- S_k is private

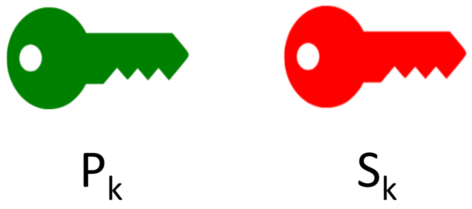




DSL at UCSB

Digital Signatures

- P_k is made public and used to verify documents signed by S_k
- S_k is private

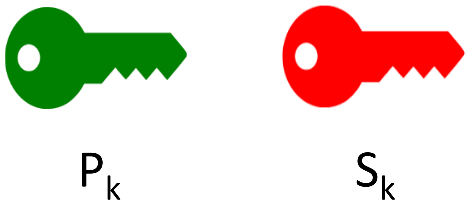


DSL at UCSB

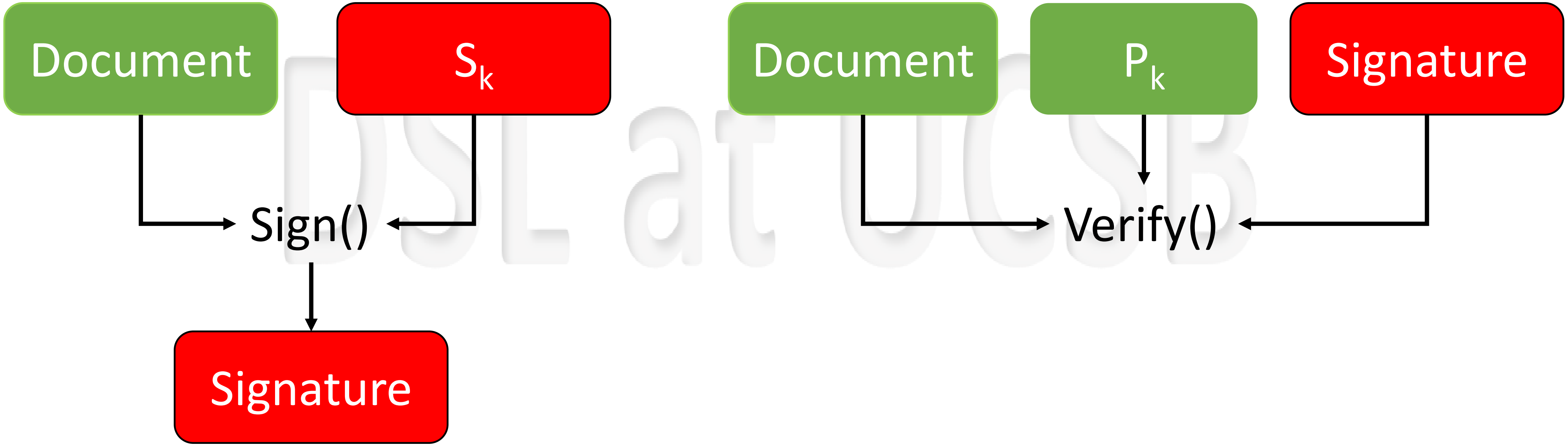
DSL

UCSB

Digital Signatures



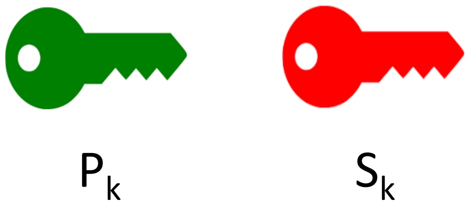
- P_k is made public and used to verify documents signed by S_k
- S_k is private

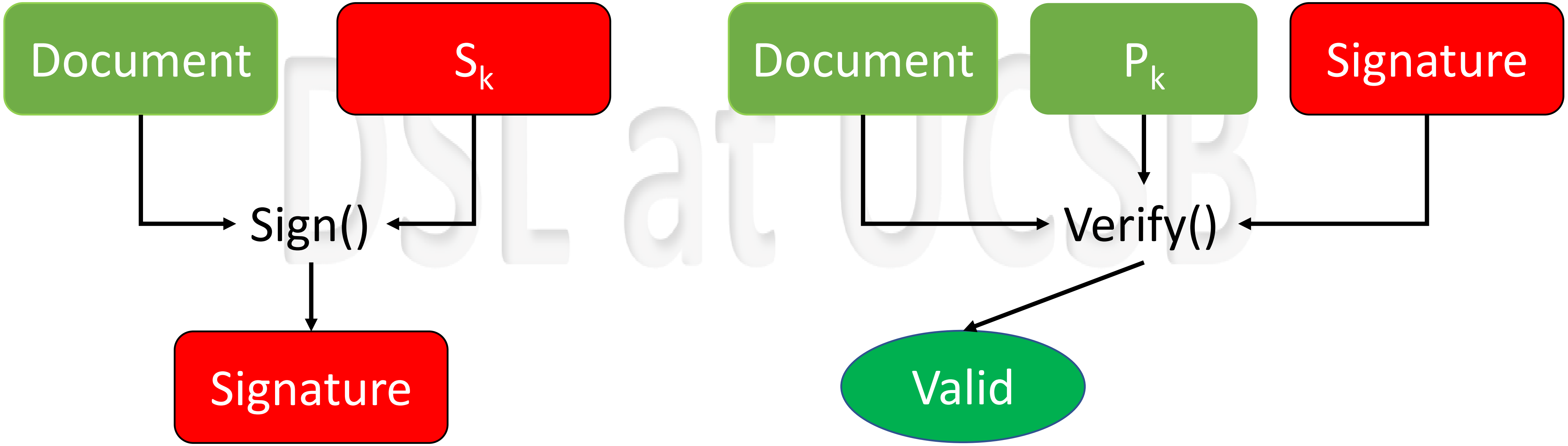


DSL at UCSB

Digital Signatures

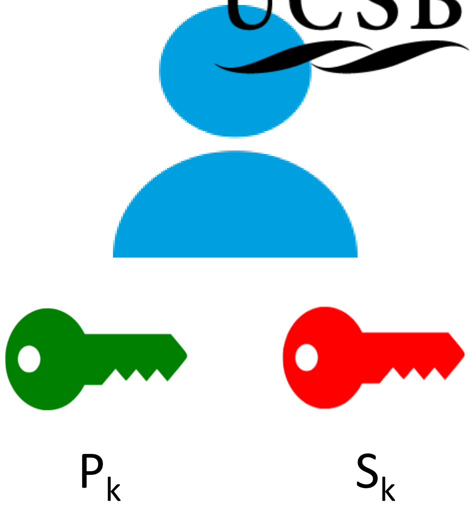
- P_k is made public and used to verify documents signed by S_k
- S_k is private



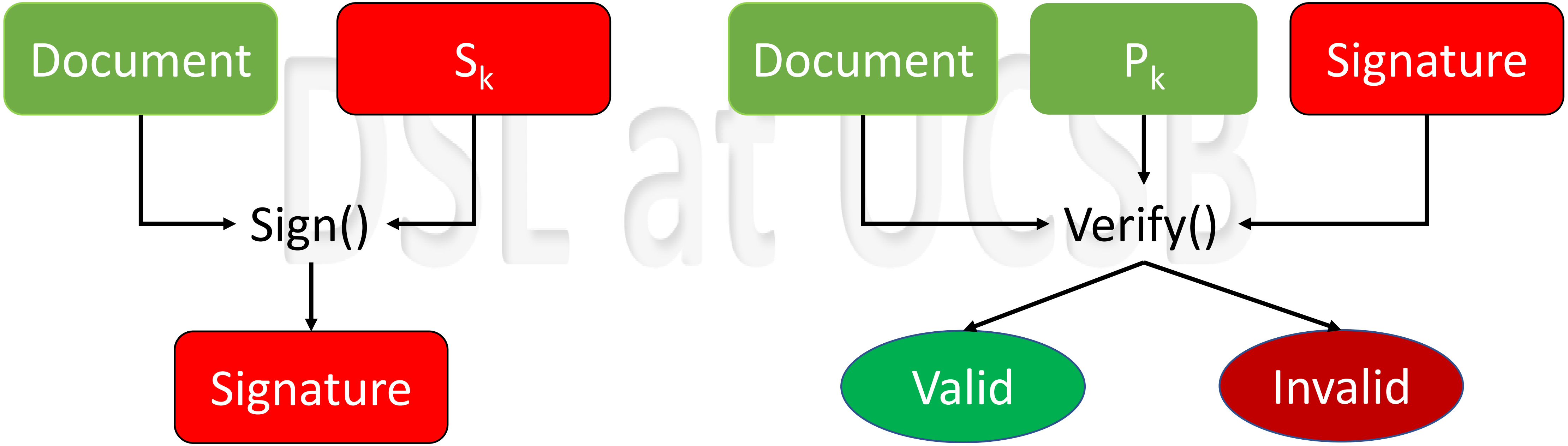


DSL at UCSB

Digital Signatures

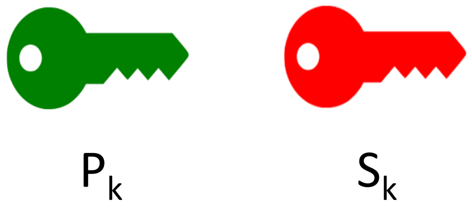


- P_k is made public and used to verify documents signed by S_k
- S_k is private

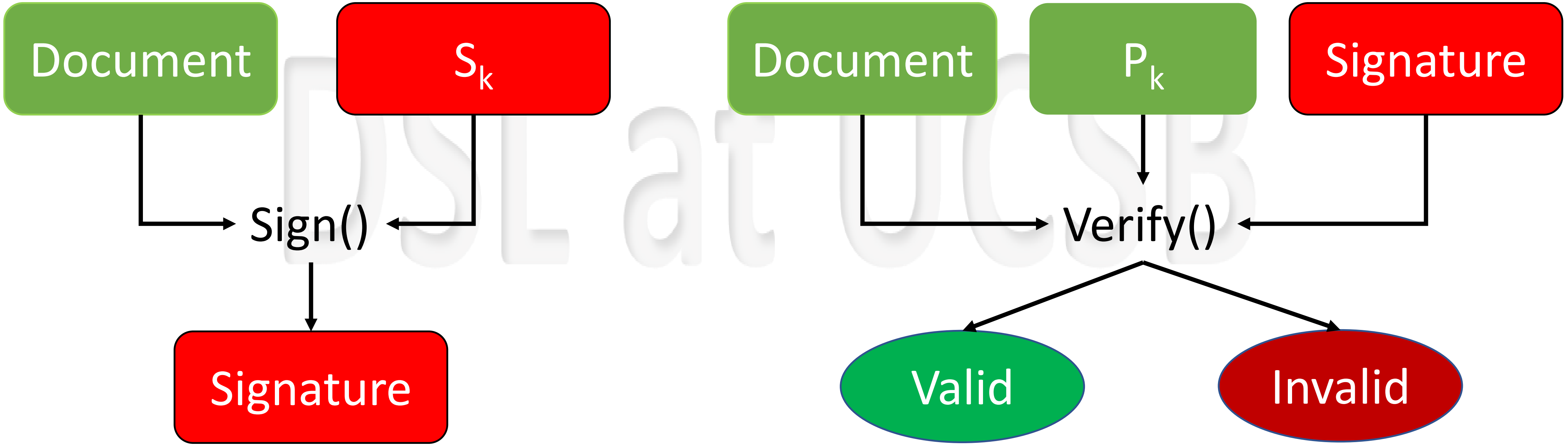


DSL at UCSB

Digital Signatures



- P_k is made public and used to verify documents signed by S_k
- S_k is private



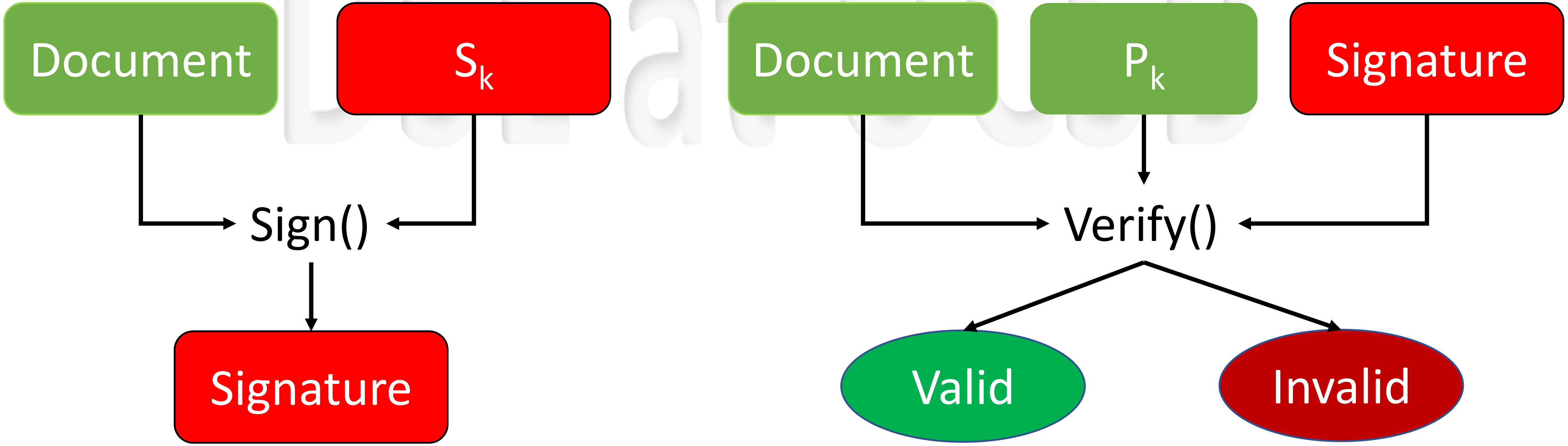
Used for Authentication not privacy

DSL at UCSB

Digital Signatures

- Unique to the signed document
- Mathematically hard to forge
- Mathematically easy to verify

DSL at UCSB



Digital Signatures and Bitcoin

- A bitcoin is a chain of digital signatures
  - Coin owners digitally sign their coins to transfer them to other recipients

DSL at UCSB

Digital Signatures and Bitcoin

- A bitcoin is a chain of digital signatures
  - Coin owners digitally sign their coins to transfer them to other recipients
  - Alice wants to move a bitcoin to Bob

DSL at UCSB

Digital Signatures and Bitcoin

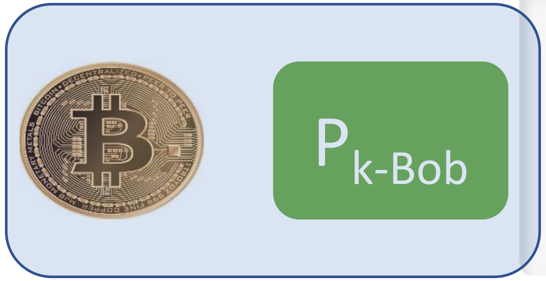
- A bitcoin is a chain of digital signatures
  - Coin owners digitally sign their coins to transfer them to other recipients
  - Alice wants to move a bitcoin to Bob

$P_{k\text{-}Bob}$

DSL at UCSB

Digital Signatures and Bitcoin

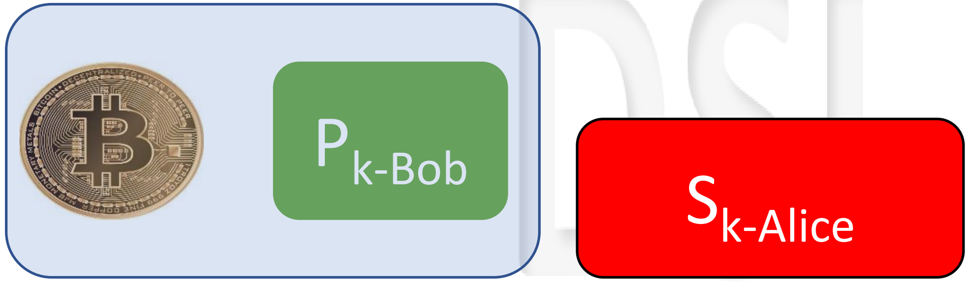
- A bitcoin is a chain of digital signatures
  - Coin owners digitally sign their coins to transfer them to other recipients
  - Alice wants to move a bitcoin to Bob



DSL at UCSB

Digital Signatures and Bitcoin

- A bitcoin is a chain of digital signatures
  - Coin owners digitally sign their coins to transfer them to other recipients
  - Alice wants to move a bitcoin to Bob

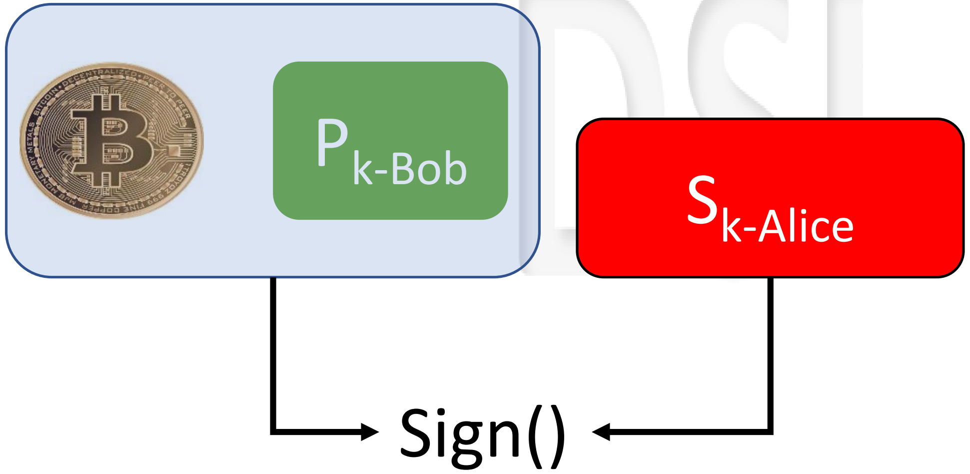


DSL at UCSB

Digital Signatures and Bitcoin

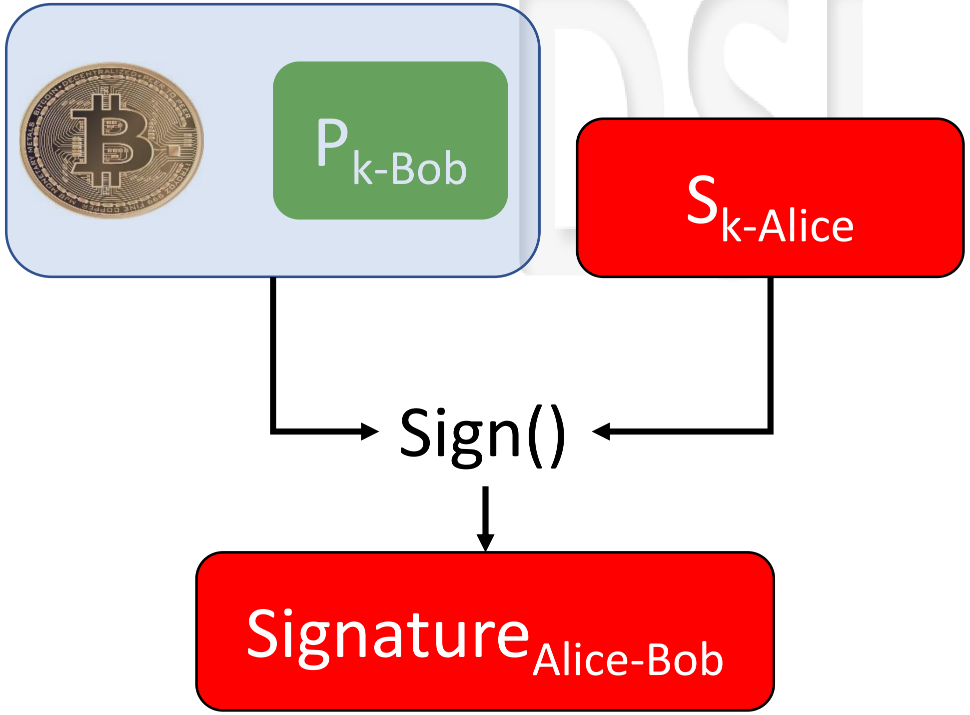
- A bitcoin is a chain of digital signatures
  - Coin owners digitally sign their coins to transfer them to other recipients
  - Alice wants to move a bitcoin to Bob

DSL at UCSB



Digital Signatures and Bitcoin

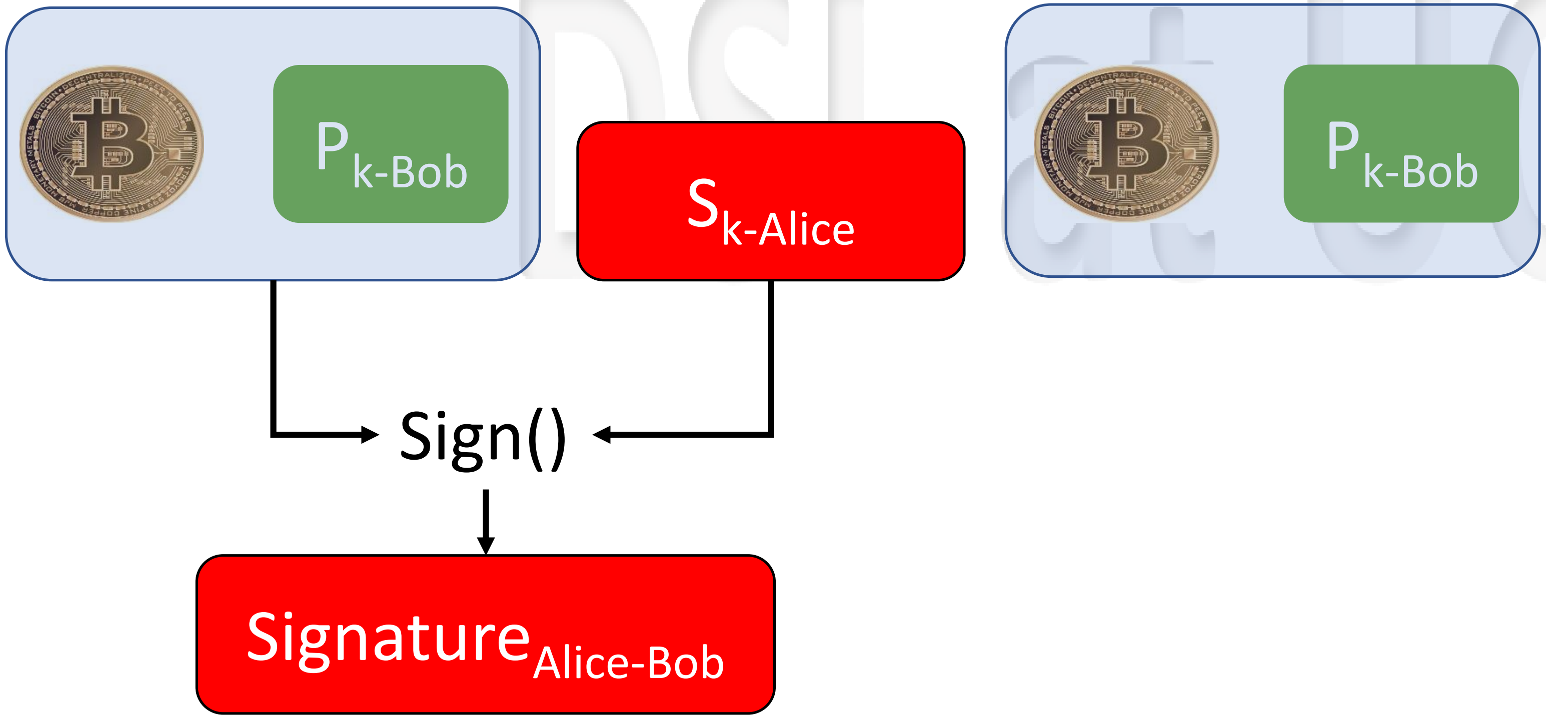
- A bitcoin is a chain of digital signatures
  - Coin owners digitally sign their coins to transfer them to other recipients
  - Alice wants to move a bitcoin to Bob



DSL at UCSB

Digital Signatures and Bitcoin

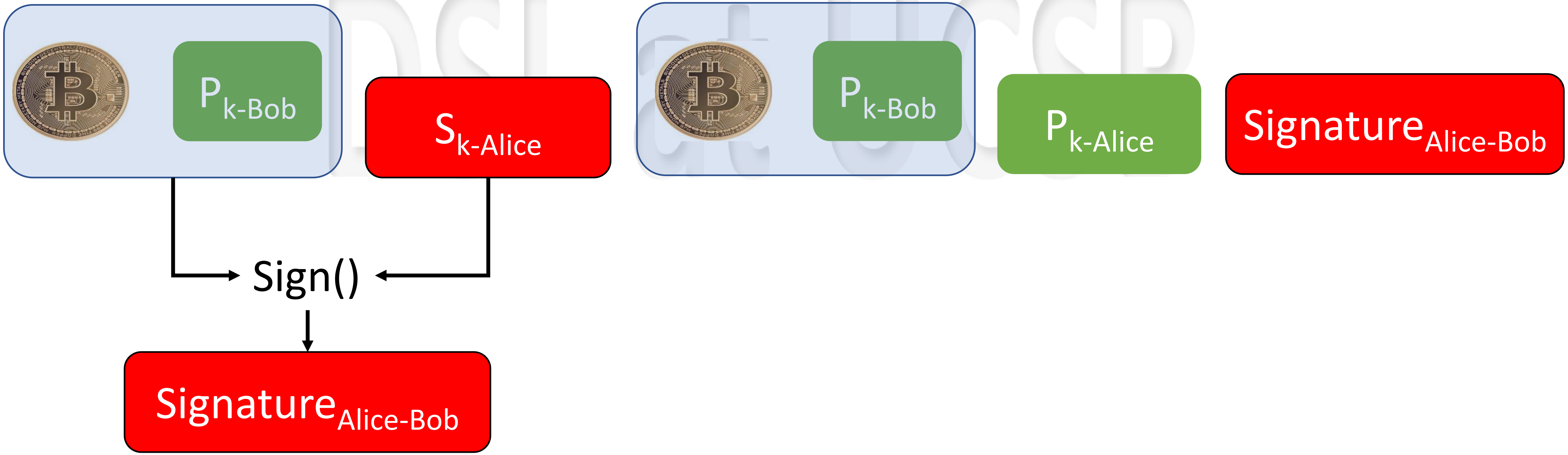
- A bitcoin is a chain of digital signatures
  - Coin owners digitally sign their coins to transfer them to other recipients
  - Alice wants to move a bitcoin to Bob



DSL UCSB

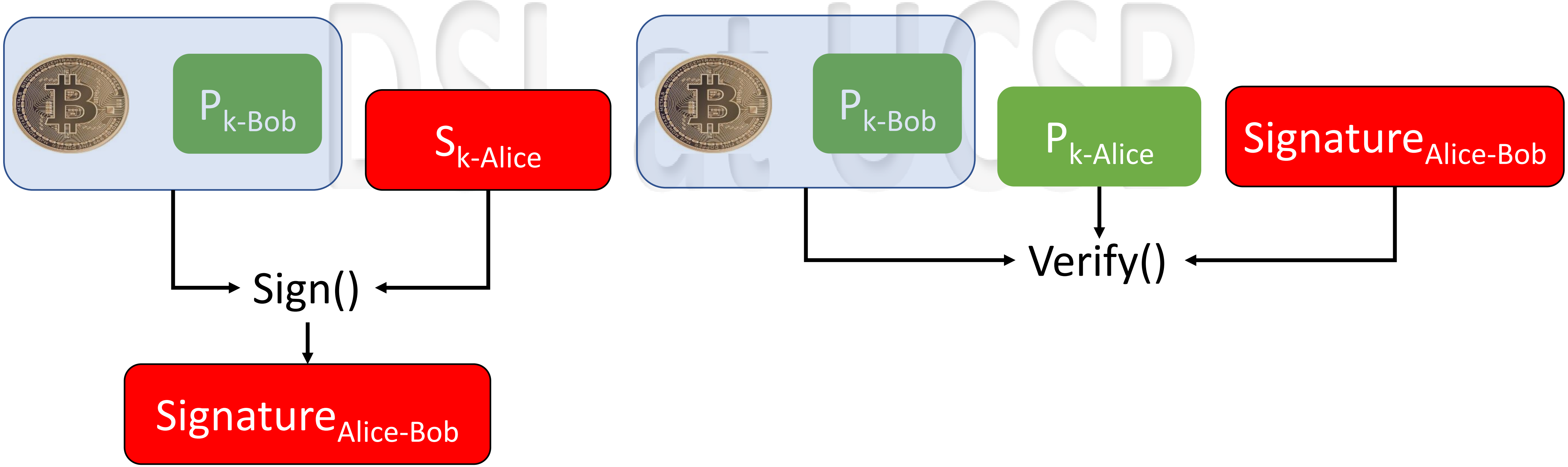
Digital Signatures and Bitcoin

- A bitcoin is a chain of digital signatures
  - Coin owners digitally sign their coins to transfer them to other recipients
  - Alice wants to move a bitcoin to Bob



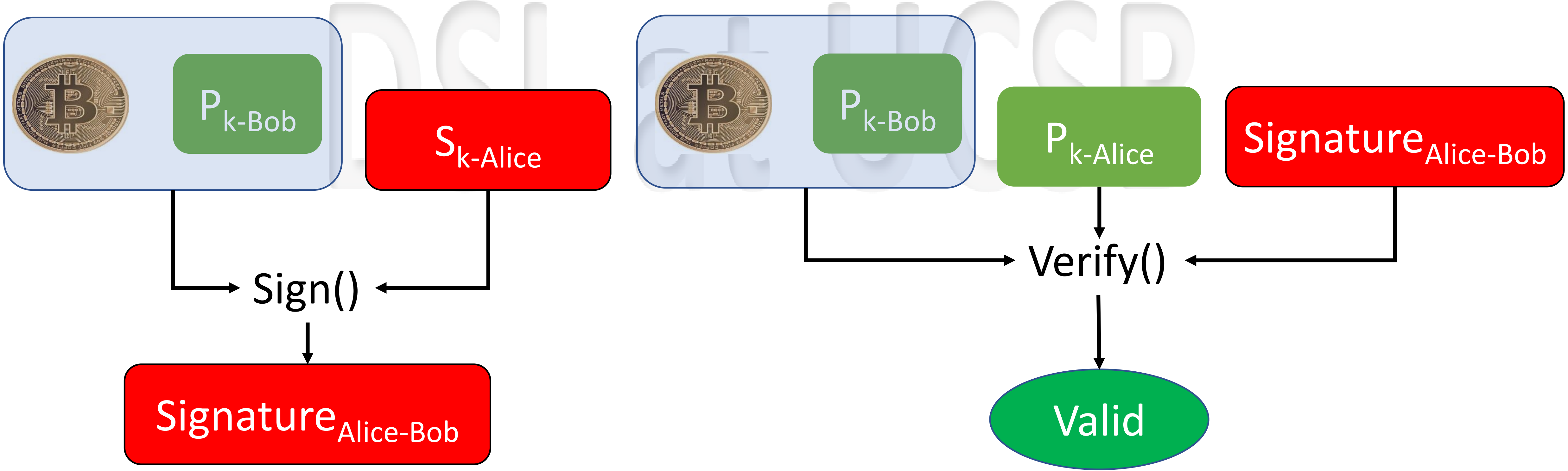
Digital Signatures and Bitcoin

- A bitcoin is a chain of digital signatures
  - Coin owners digitally sign their coins to transfer them to other recipients
  - Alice wants to move a bitcoin to Bob



Digital Signatures and Bitcoin

- A bitcoin is a chain of digital signatures
  - Coin owners digitally sign their coins to transfer them to other recipients
  - Alice wants to move a bitcoin to Bob



Digital Signatures and Bitcoin

- Now what if Bob wants to move his coins to Diana

DSL at UCSB

Digital Signatures and Bitcoin

- Now what if Bob wants to move his coins to Diana

Signature$_{\text{Alice-Bob}}$

DSL at UCSB

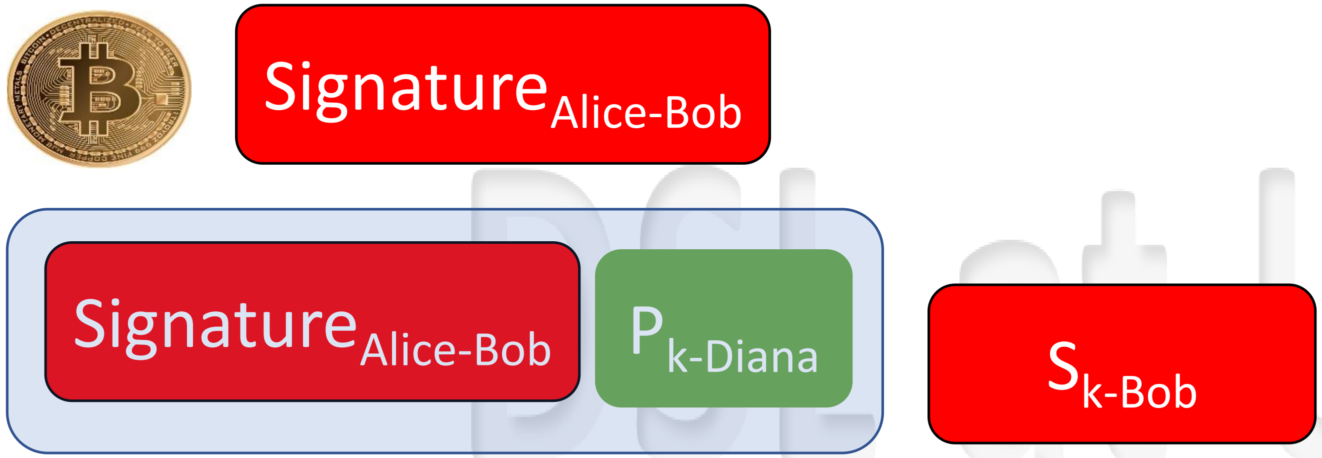
Digital Signatures and Bitcoin

- Now what if Bob wants to move his coins to Diana

DSL at UCSB

Digital Signatures and Bitcoin

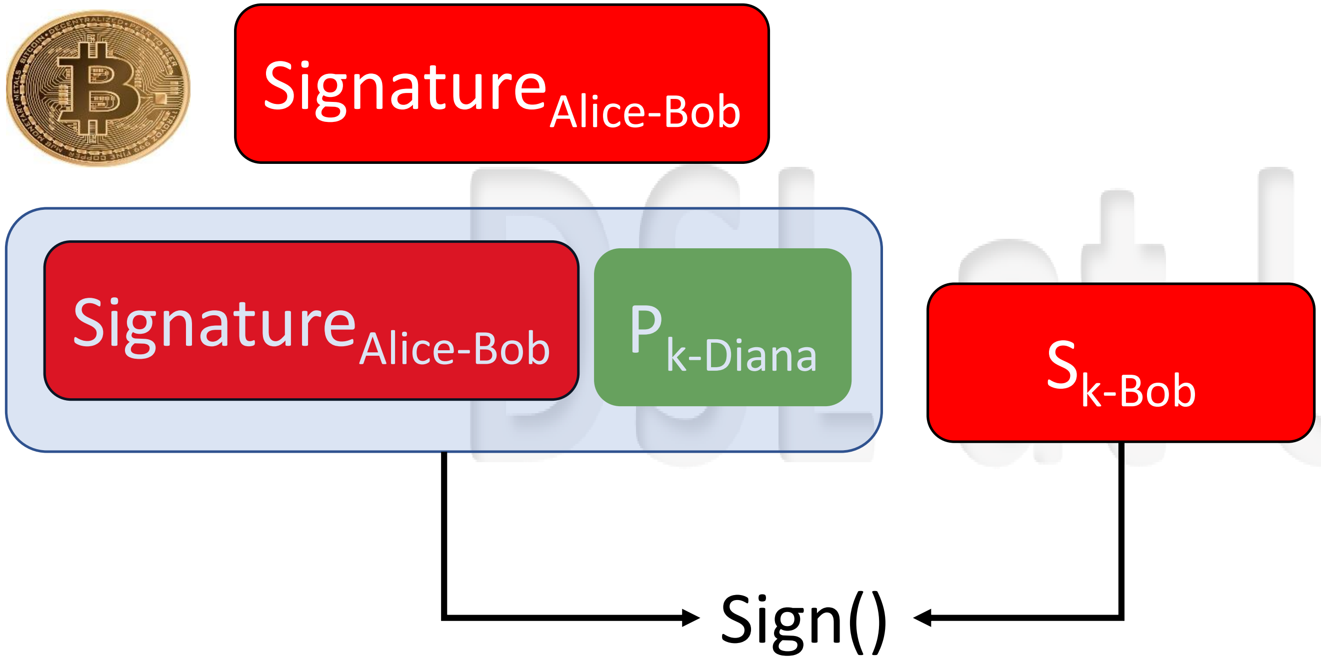
- Now what if Bob wants to move his coins to Diana



DSL at UCSB

Digital Signatures and Bitcoin

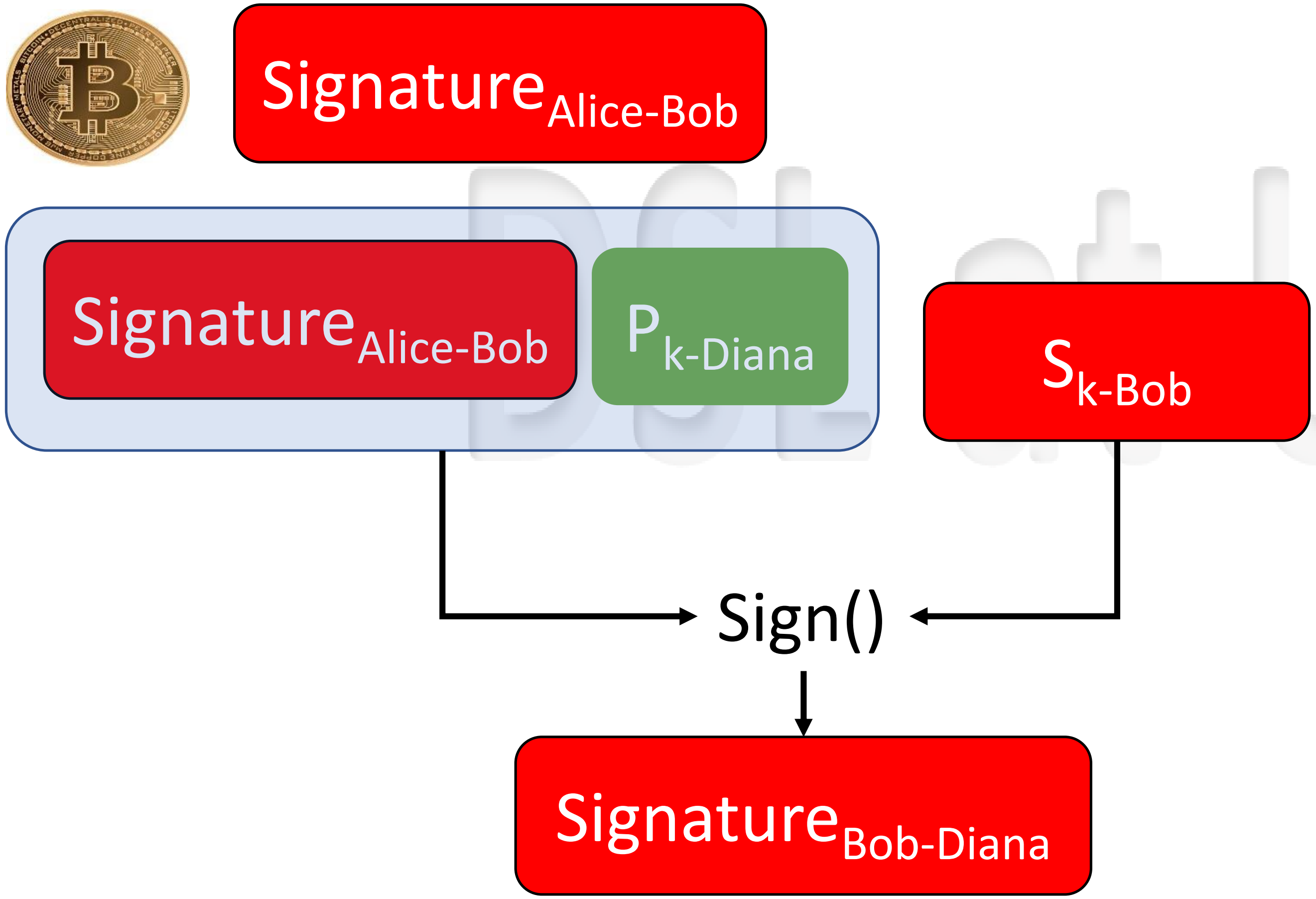
- Now what if Bob wants to move his coins to Diana



DSL at UCSB

Digital Signatures and Bitcoin

- Now what if Bob wants to move his coins to Diana



DSL at UCSB

A Bitcoin Big Picture

DSL at UCSB

A Bitcoin Big Picture

Signature$_{\text{...-Alice}}$

DSL at UCSB

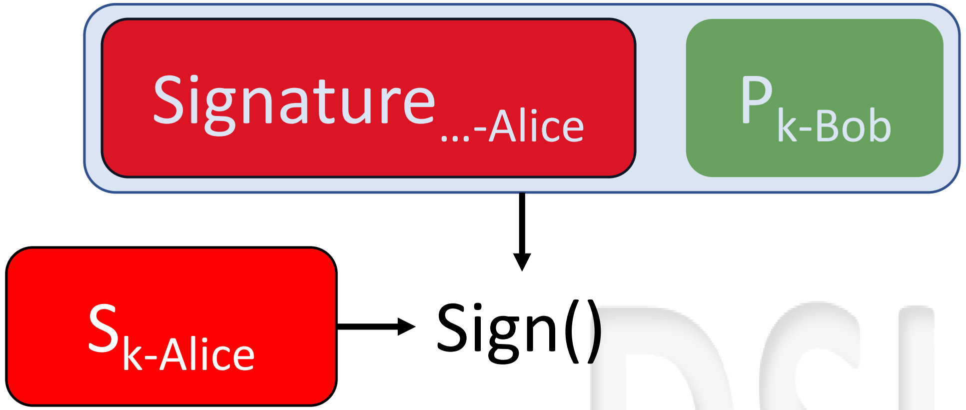
A Bitcoin Big Picture

$\text{Signature}_{\text{...-Alice}}$

$P_{\text{k-Bob}}$

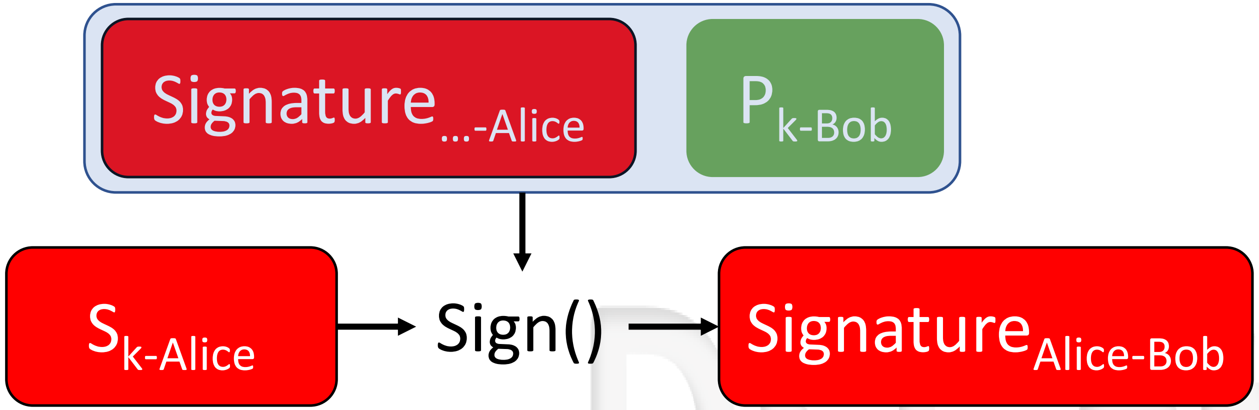
DSL at UCSB

A Bitcoin Big Picture



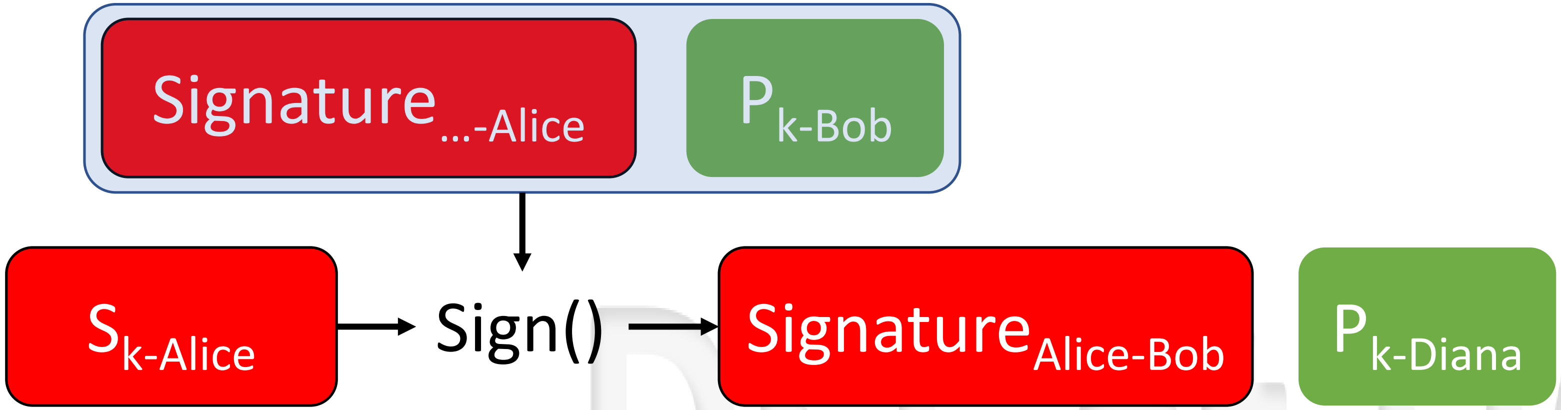
DSL at UCSB

A Bitcoin Big Picture



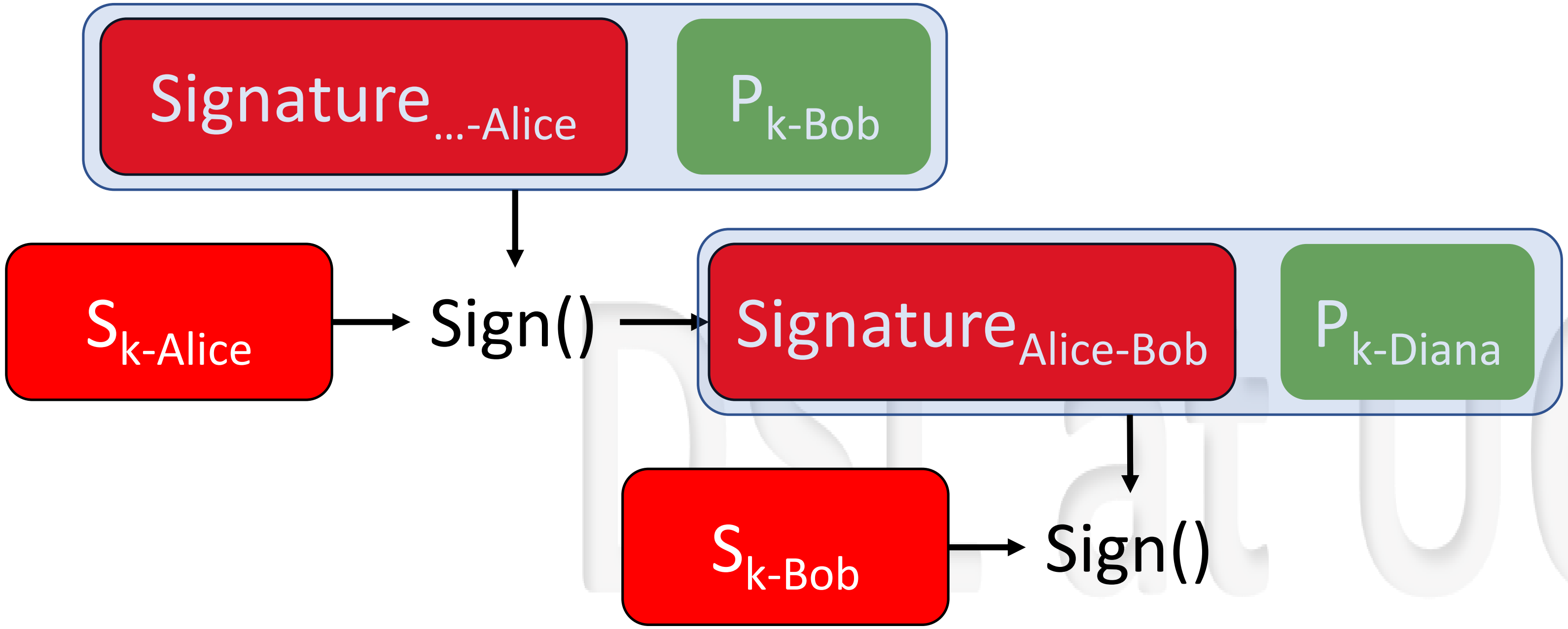
DSL at UCSB

A Bitcoin Big Picture



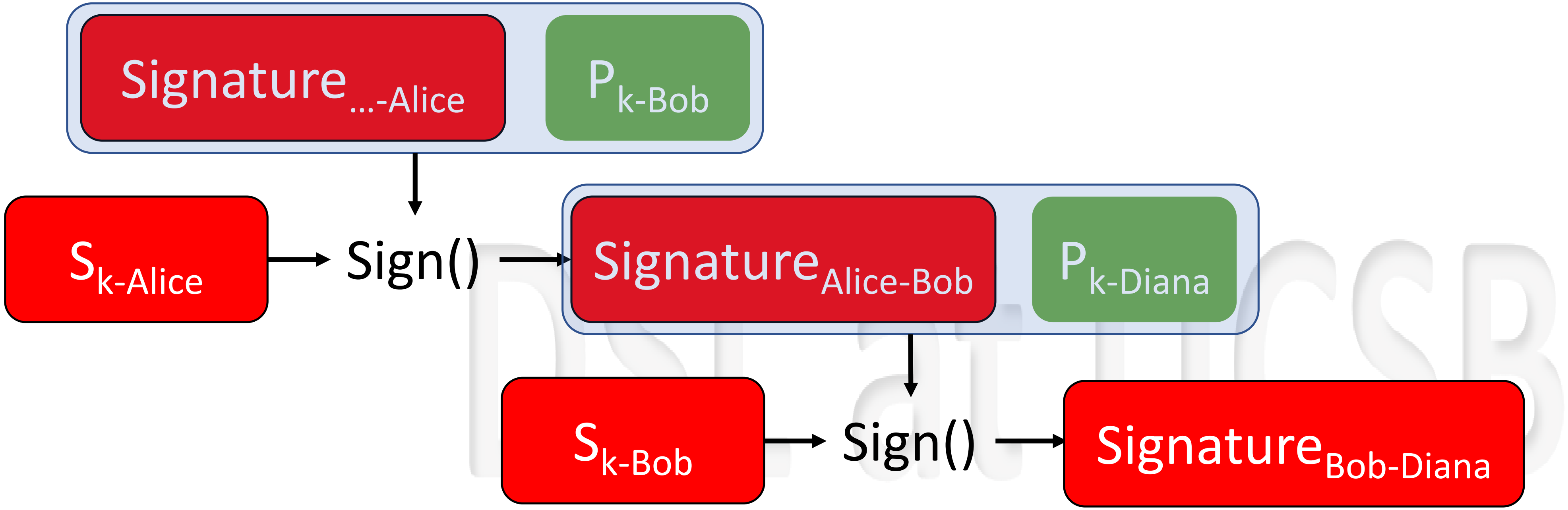
DSL at UCSB

A Bitcoin Big Picture



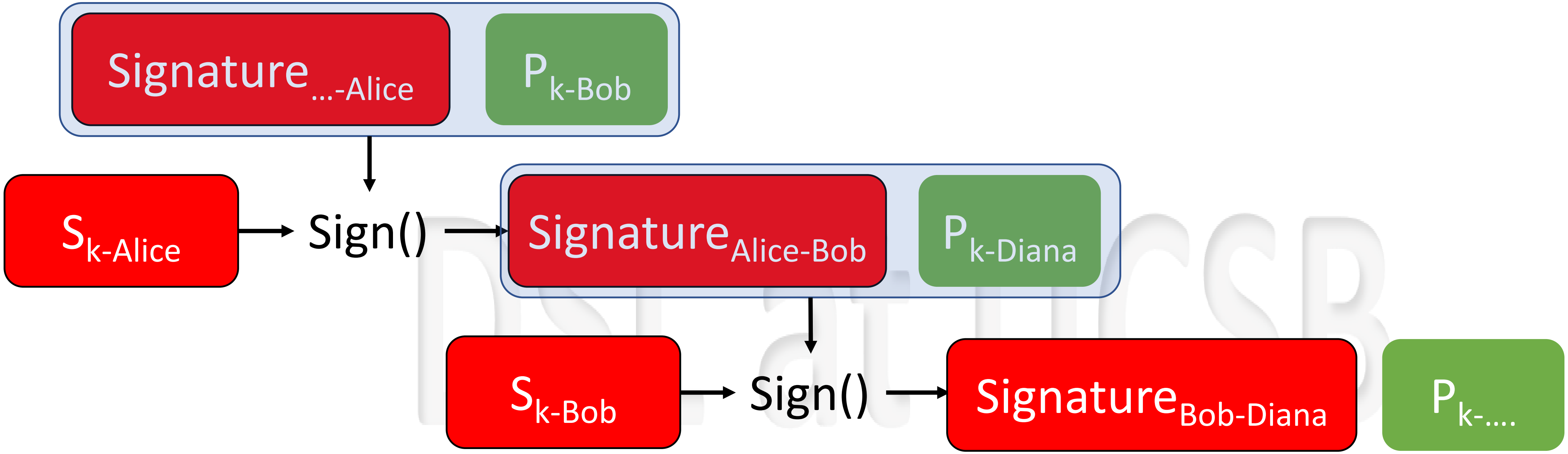
DSL at UCSB

A Bitcoin Big Picture



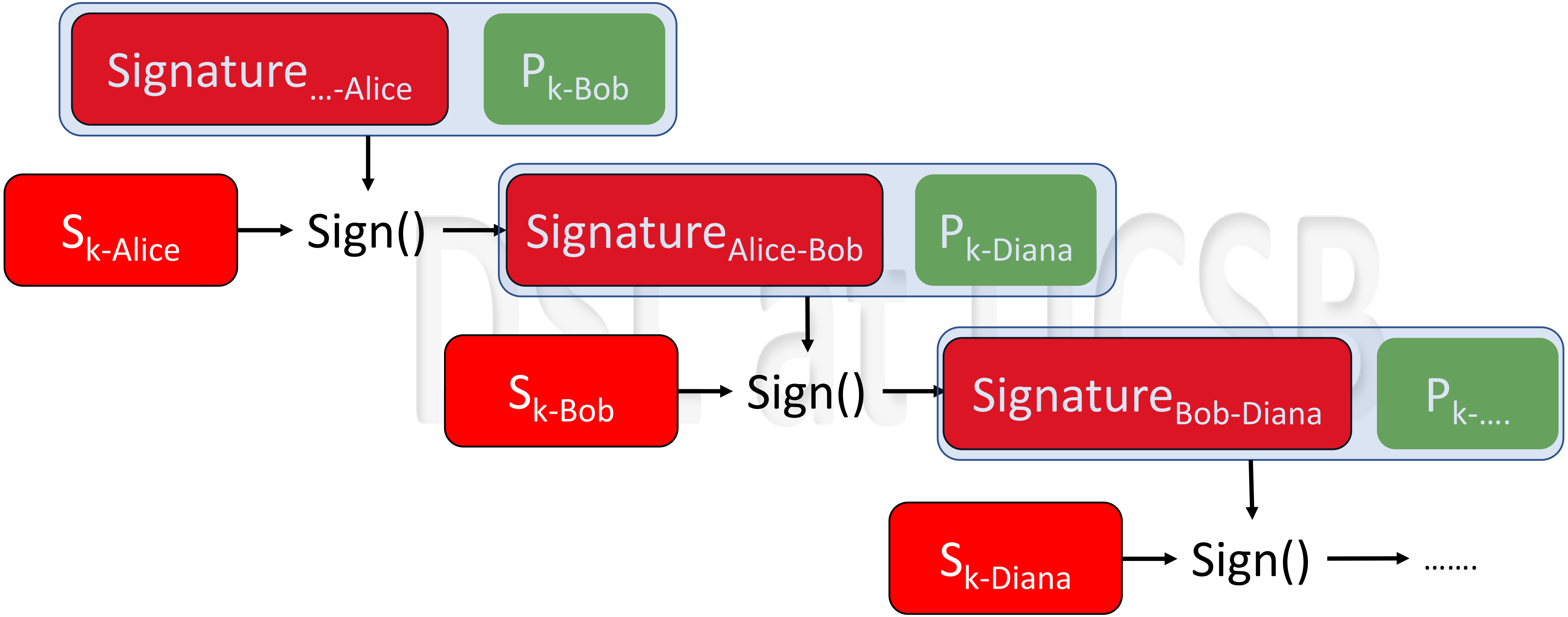
DSL at UCSB

A Bitcoin Big Picture

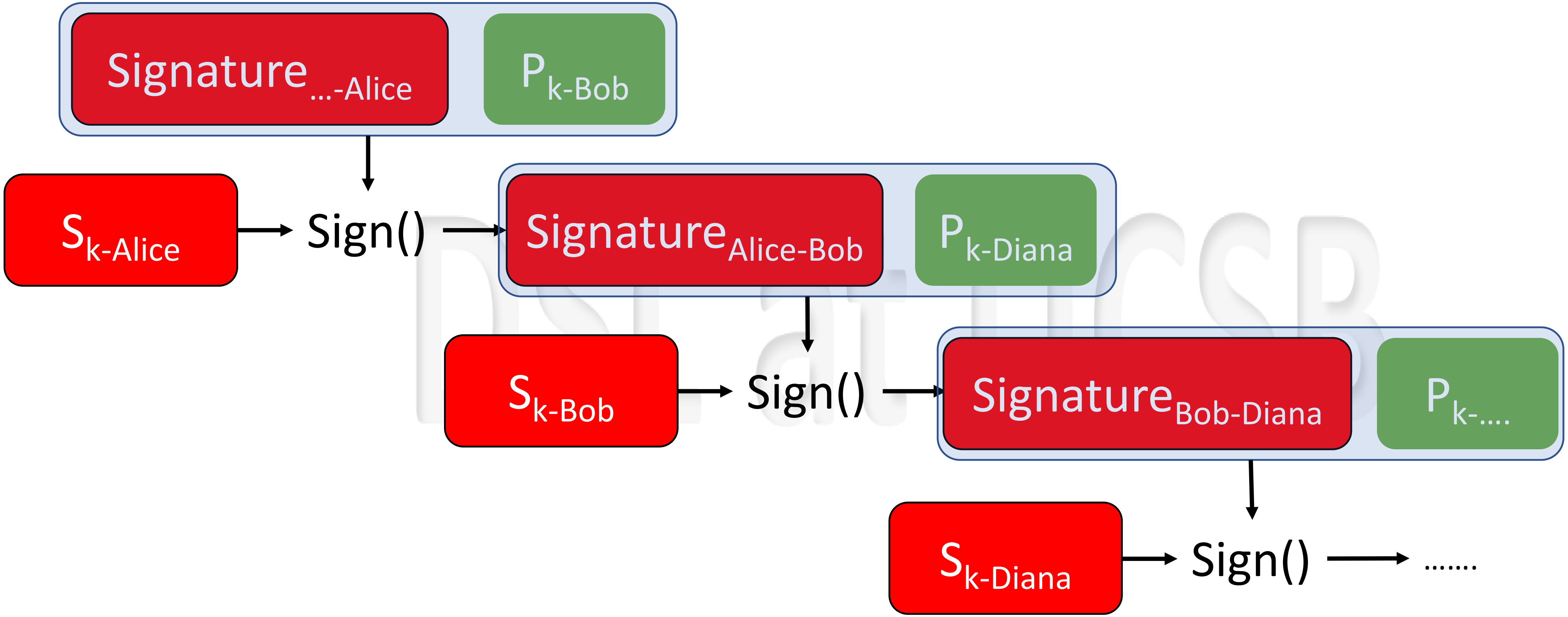


DSL at UCSB

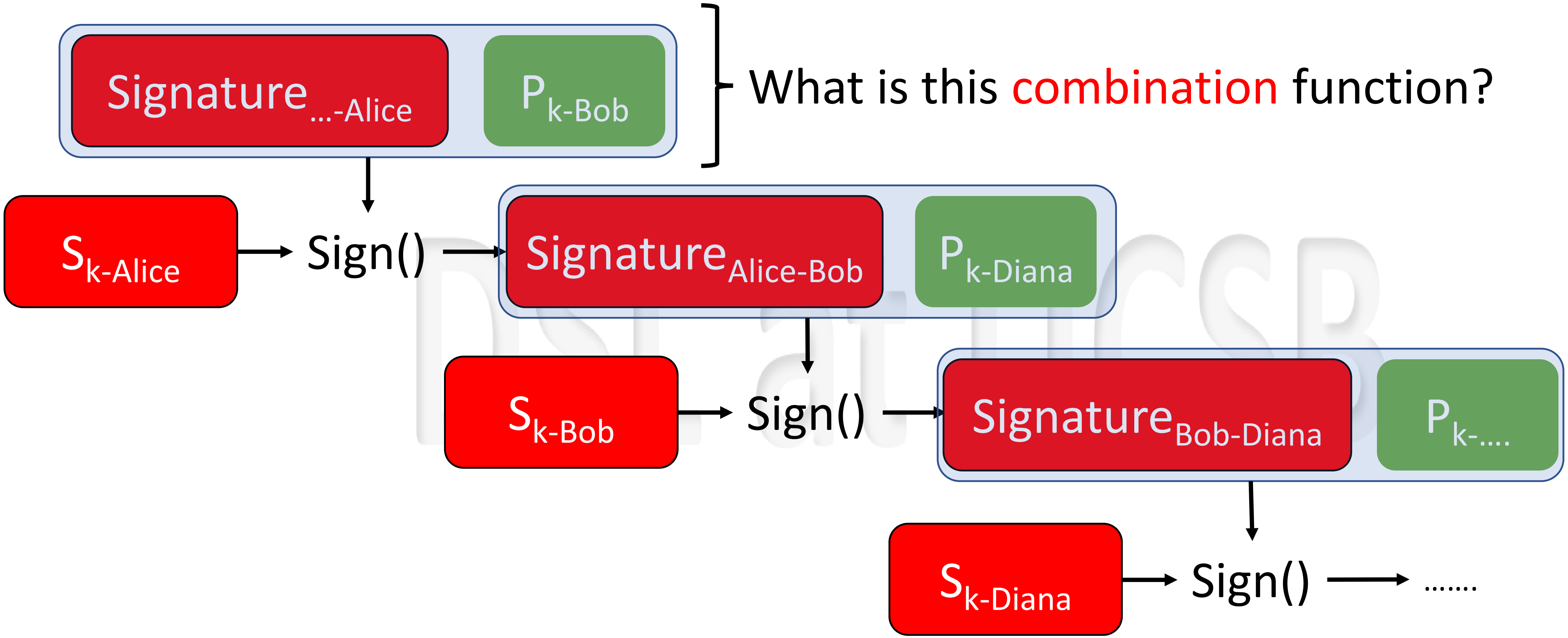
A Bitcoin Big Picture



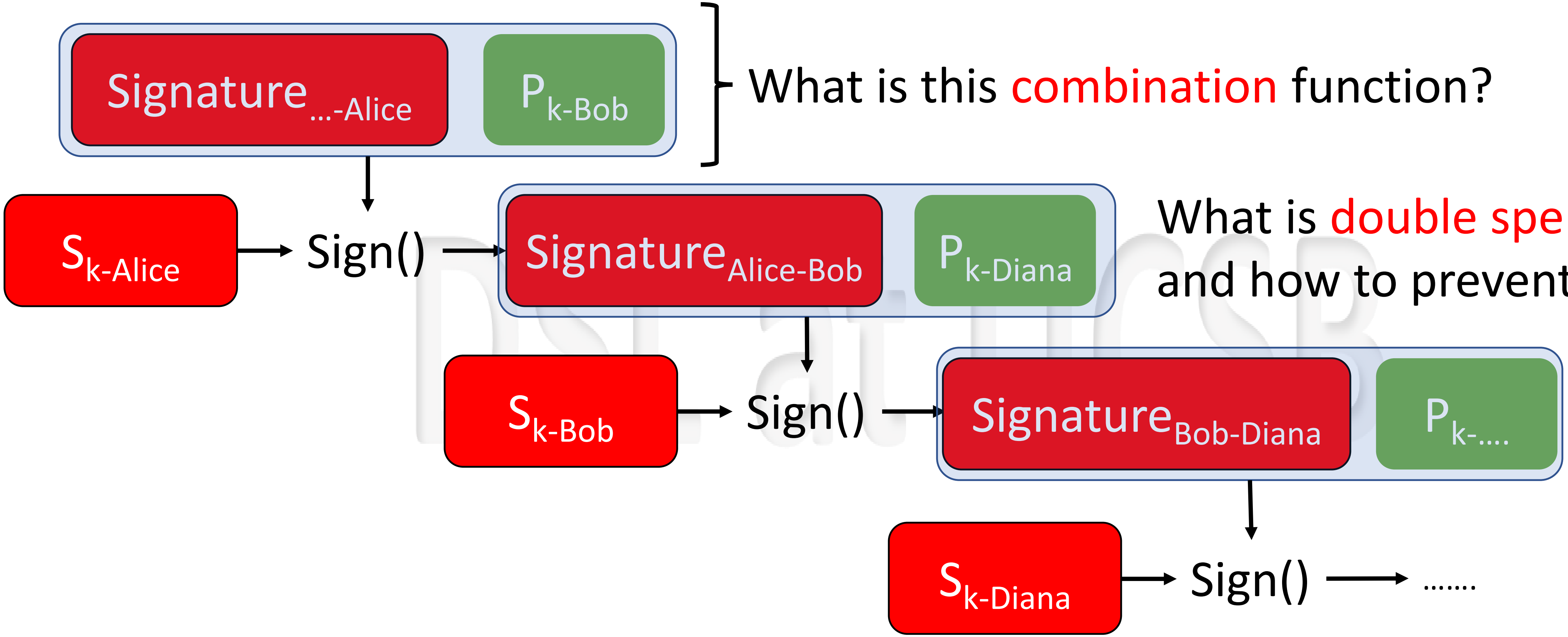
What About's?


Signature ...-Alice
Pk-Bob
Sk-Alice
Sign()
Signature Alice-Bob
Pk-Diana
Sk-Bob
Sign()
Signature Bob-Diana
Pk-....
Sk-Diana
Sign()
....

What About's?


Signature ...-Alice
Pk-Bob
What is this combination function?
Sk-Alice
Sign()
Signature Alice-Bob
Pk-Diana
Sk-Bob
Sign()
Signature Bob-Diana
Pk-....
Sk-Diana
Sign()
....

What About's?

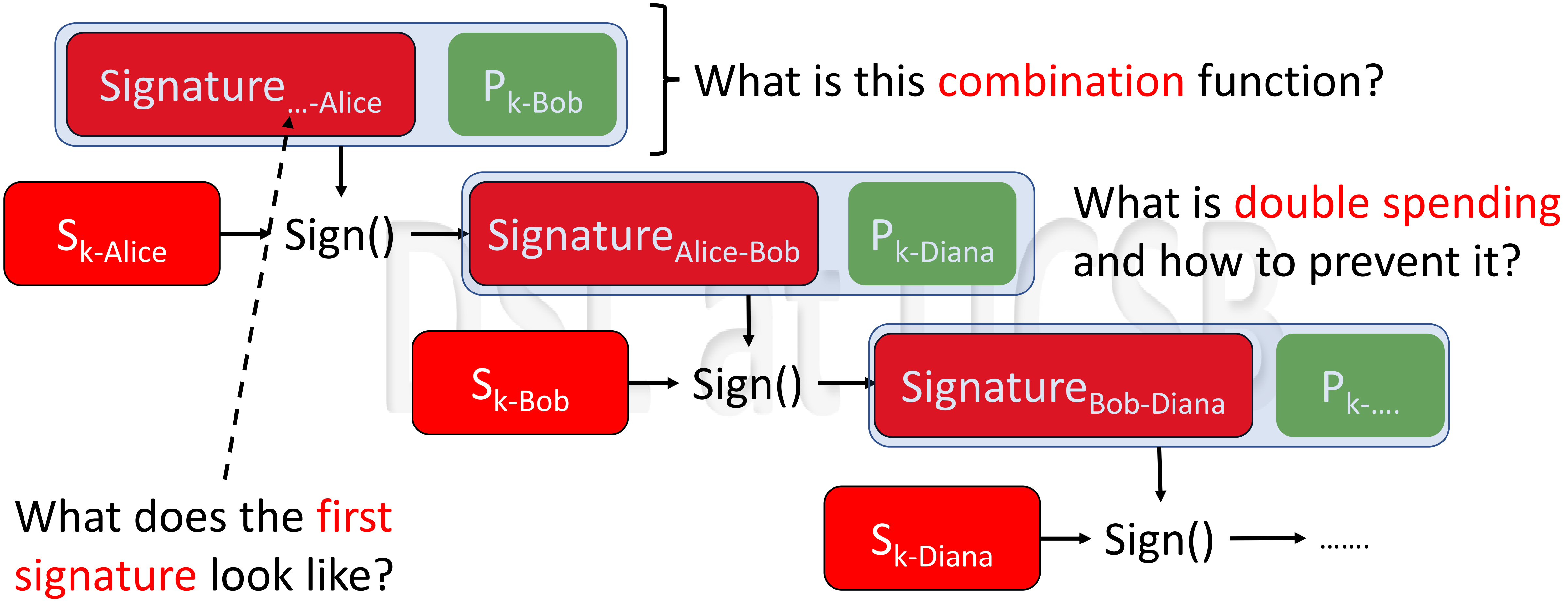


What is this combination function?

What is double spending and how to prevent it?

DSL at UCSB

What About's?



DSL at UCSB

Hashing H(x)

$\text{Signature}_{\text{Alice-Bob}}$ $P_{\text{k-Diana}}$

DSL at UCSB

Hashing H(x)

$\text{Signature}_{\text{Alice-Bob}}$ $P_{\text{k-Diana}}$

- Signatures and public keys are combined using Hashing

DSL at UCSB

Hashing H(x)

$Signature_{Alice\text{-}Bob}$ $P_{k\text{-}Diana}$

- Signatures and public keys are combined using Hashing
- Takes **any** string x **of any length** as input
- **Fixed** output size (e.g., 256 bits)

DSL at UCSB

Hashing H(x)

$\text{Signature}_{\text{Alice-Bob}}$ $P_{\text{k-Diana}}$

- Signatures and public keys are combined using Hashing
- Takes **any** string x **of any length** as input
- **Fixed** output size (e.g., 256 bits)
- Efficiently computable.
- **Satisfies:**
  - Collision Free: no two x, y s.t. H(x) = H(y)
    - Message digest.
  - Hiding: Given H(x) infeasible to find x (one-way hash function)
    - Commitment: commit to a value and reveal later
  - Puzzle Friendly: Given a random puzzle ID and a target **set** Y it is hard to find x such that: H(ID | x) ε Y

DSL at UCSB

Bitcoin uses SHA-256

$\text{Signature}_{\text{Alice-Bob}}$	$P_{\text{k-Diana}}$

DSL at UCSB

Bitcoin uses SHA-256

$Signature_{Alice-Bob}$ $P_{k-Diana}$

SHA256($Signature_{Alice-Bob}$ || $P_{k-Diana}$) =
256-bit (32-byte) unique string

DSL at UCSB

Bitcoin uses SHA-256

$\text{Signature}_{\text{Alice-Bob}}$ $P_{\text{k-Diana}}$

SHA256($\text{Signature}_{\text{Alice-Bob}}$ || $P_{\text{k-Diana}}$) =

256-bit (32-byte) unique string

DSL at UCSB

Bitcoin uses SHA-256

$\text{Signature}_{\text{Alice-Bob}}$ $P_{\text{k-Diana}}$

SHA256($\text{Signature}_{\text{Alice-Bob}}$ || $P_{\text{k-Diana}}$) =

256-bit (32-byte) unique string

SHA256(abc) =
ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad

DSL at UCSB

Bitcoin uses SHA-256

$\text{Signature}_{\text{Alice-Bob}}$ $P_{\text{k-Diana}}$

SHA256($\text{Signature}_{\text{Alice-Bob}}$ || $P_{\text{k-Diana}}$) =

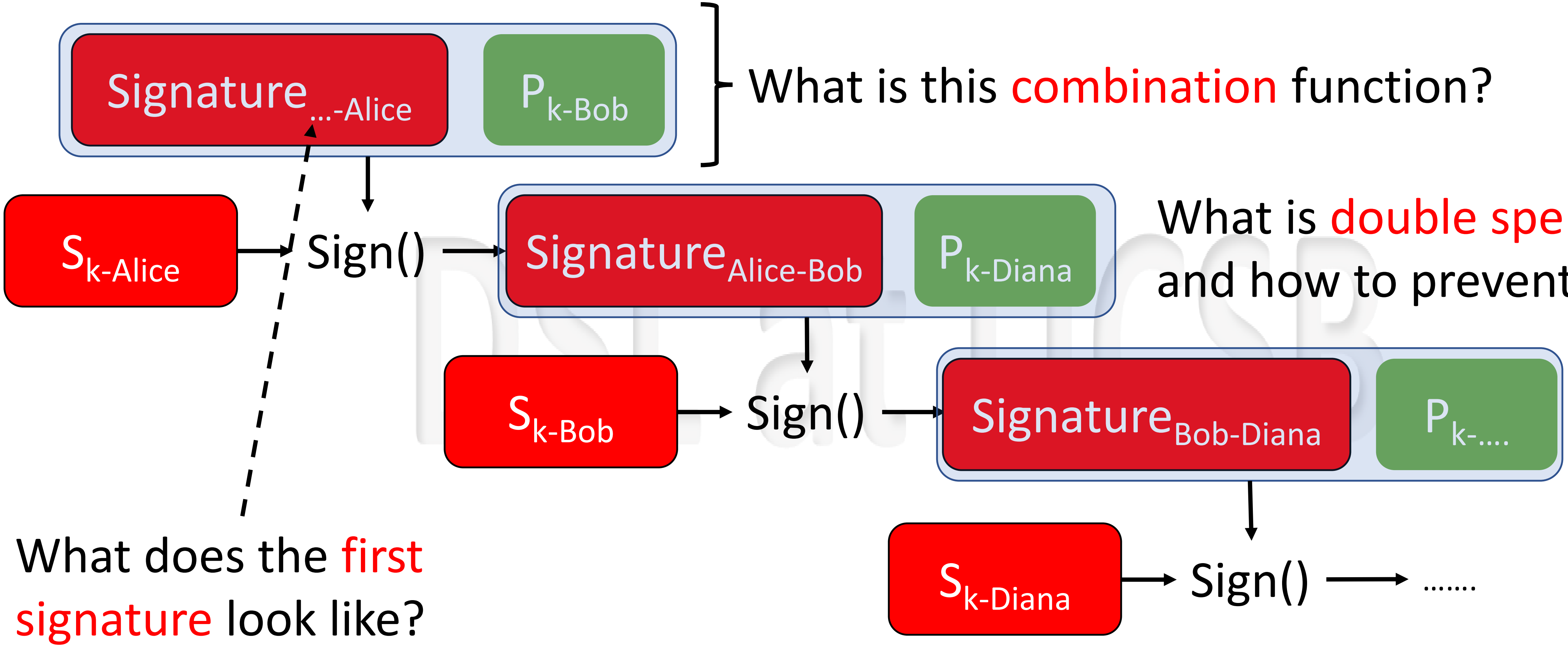
256-bit (32-byte) unique string

DSL at UCSB

SHA256(abc) =
ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad

SHA256(abC) =
0a2432a1e349d8fdb9bfca91bba9e9f2836990fe937193d84deef26c6f3b8f76

What About's?



What is this combination function?

What is double spending and how to prevent it?

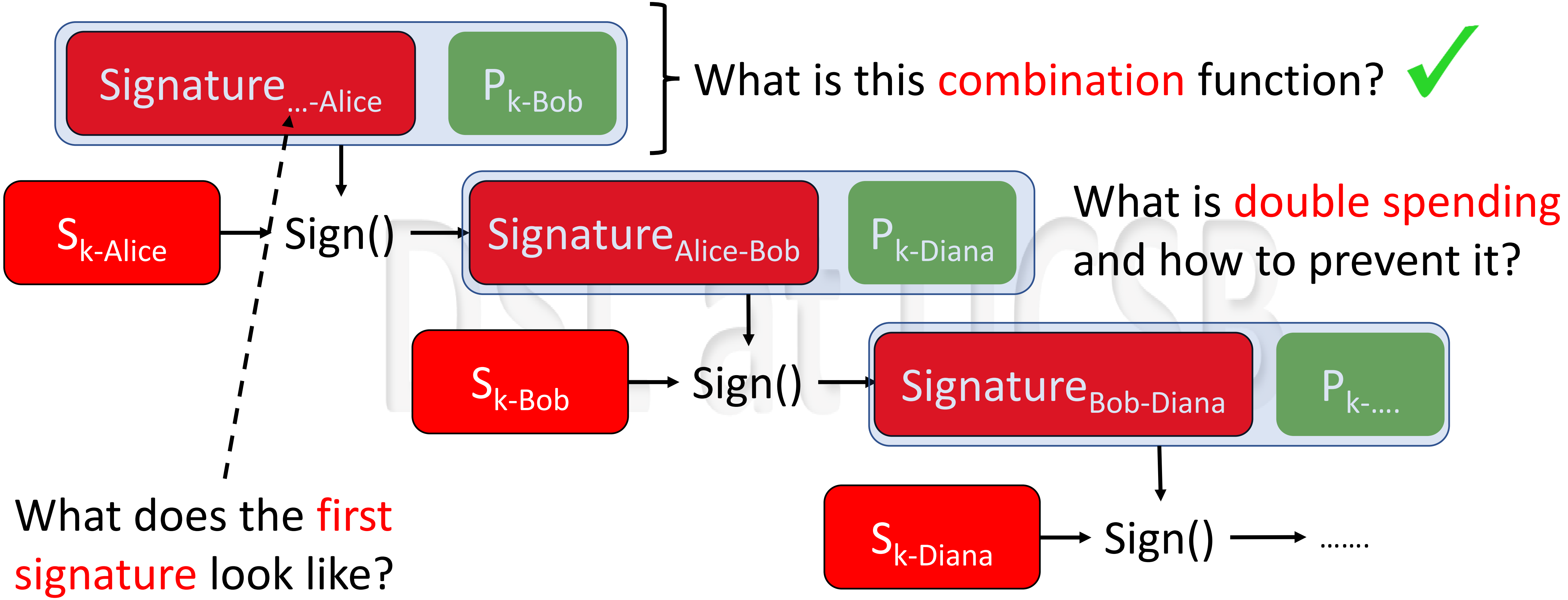
What does the first signature look like?

DSL at UCSB

DSL

UCSB

What About's?



DSL at UCSB

Double Spending

- Spending the same digital cash asset more than once
- Impossible to do in physical cash
- Prevented in traditional banking systems through concurrency control

DSL at UCSB

Double Spending

- Spending the same digital cash asset more than once
- Impossible to do in physical cash
- Prevented in traditional banking systems through concurrency control

$\text{Signature}_{\text{Alice-Bob}}$

DSL at UCSB

Double Spending

- Spending the same digital cash asset more than once
- Impossible to do in physical cash
- Prevented in traditional banking systems through concurrency control

DSL at UCSB

$\text{Signature}_{\text{Alice-Bob}}$

$\text{Signature}_{\text{Alice-Bob}}$

Double Spending

- Spending the same digital cash asset more than once
- Impossible to do in physical cash
- Prevented in traditional banking systems through concurrency control

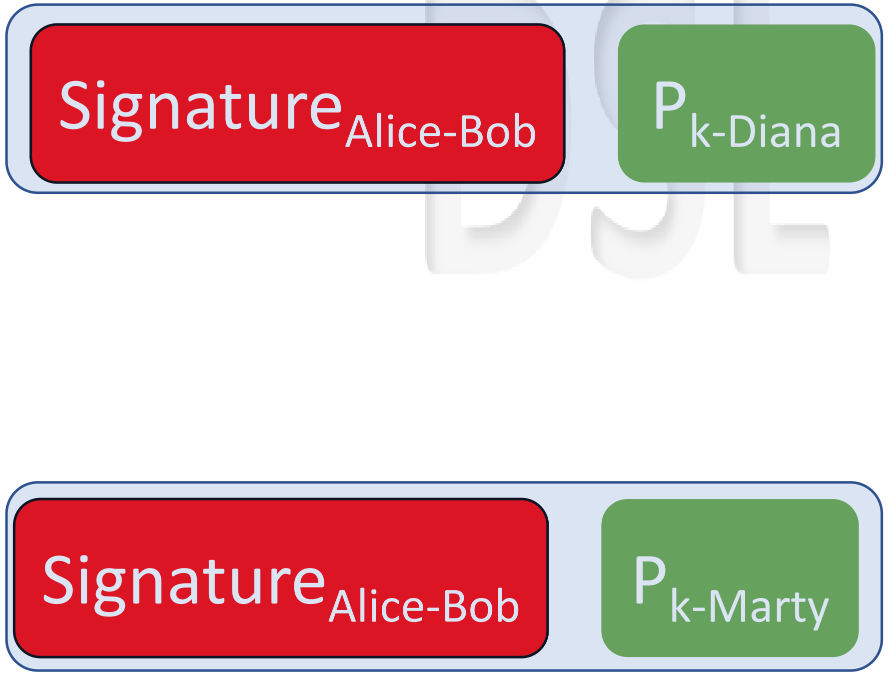
DSL at UCSB

$\text{Signature}_{\text{Alice-Bob}}$ $P_{\text{k-Diana}}$

$\text{Signature}_{\text{Alice-Bob}}$ $P_{\text{k-Marty}}$

Double Spending

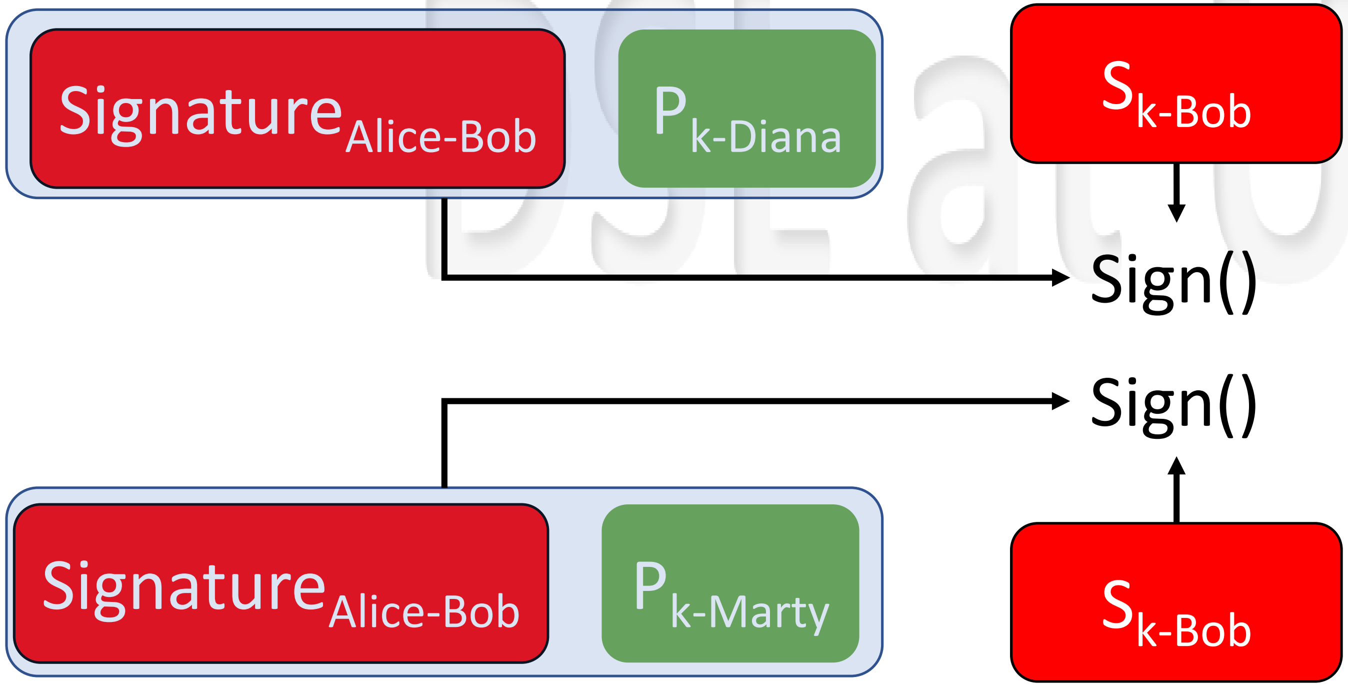
- Spending the same digital cash asset more than once
- Impossible to do in physical cash
- Prevented in traditional banking systems through concurrency control



DSL at UCSB

Double Spending

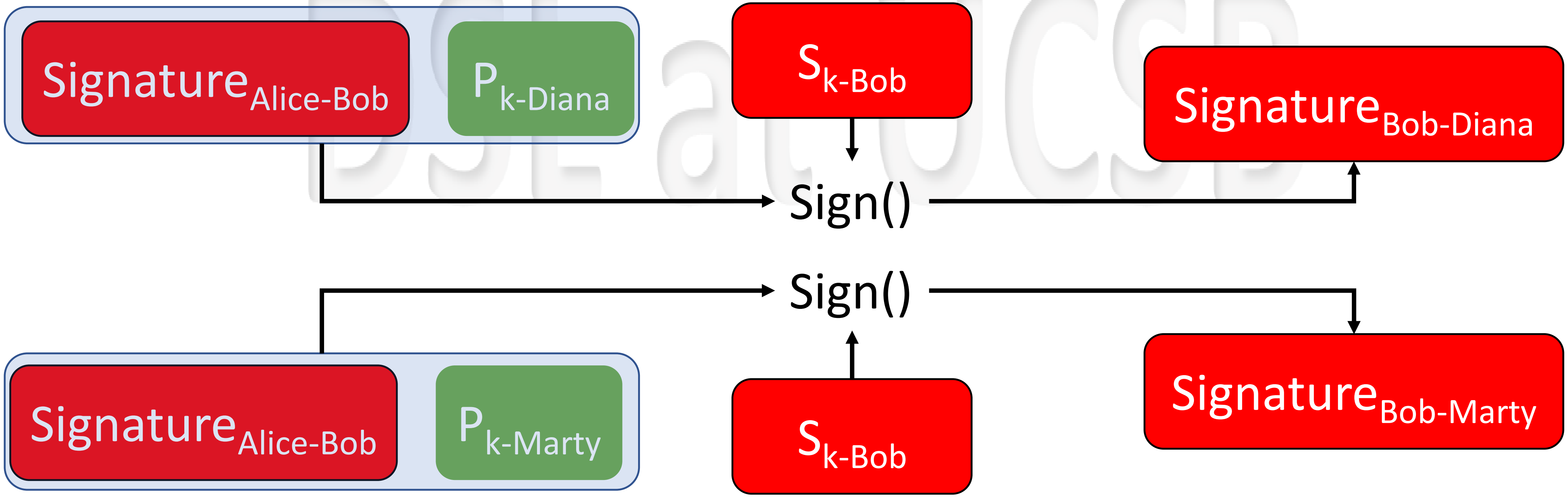
- Spending the same digital cash asset more than once
- Impossible to do in physical cash
- Prevented in traditional banking systems through concurrency control



DSL at UCSB

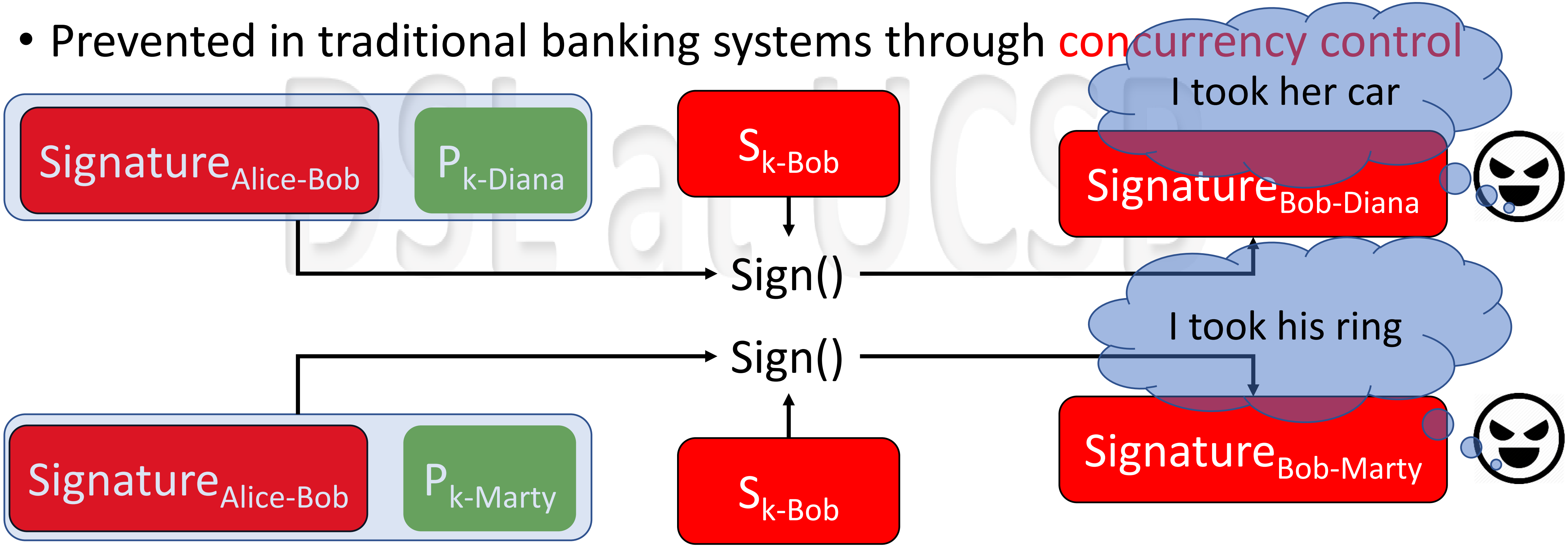
Double Spending

- Spending the same digital cash asset more than once
- Impossible to do in physical cash
- Prevented in traditional banking systems through concurrency control



Double Spending

- Spending the same digital cash asset more than once
- Impossible to do in physical cash
- Prevented in traditional banking systems through concurrency control



Double Spending Prevention

- Centralized

DSL at UCSB

Double Spending Prevention

- Centralized
  - Transactions on coins go through a trusted 3rd party (Trent)

DSL at UCSB

Double Spending Prevention

- Centralized
  - Transactions on coins go through a trusted 3rd party (Trent)

50 BTC

$\text{Signature}_{\text{Trent-Bob}}$

DSL at UCSB

Double Spending Prevention

- Centralized
  - Transactions on coins go through a trusted 3rd party (Trent)

50 BTC

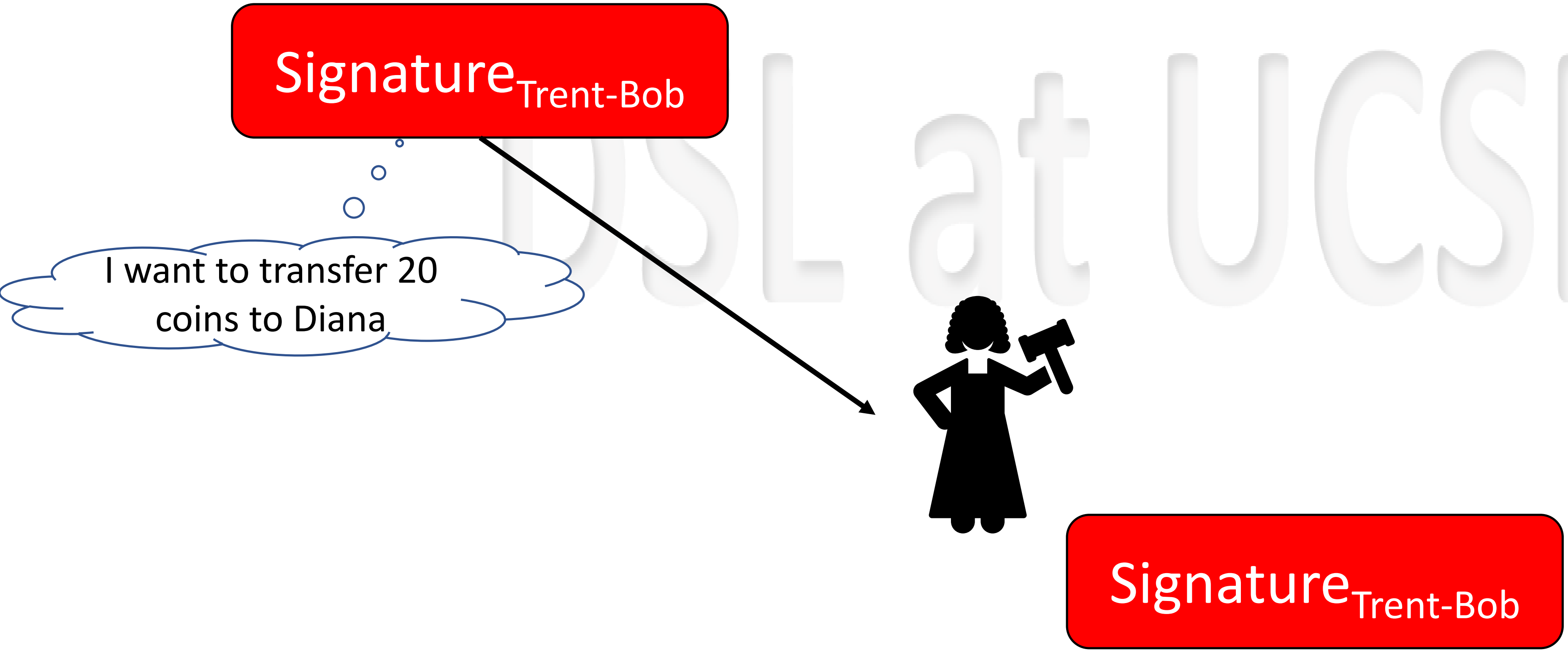
$\text{Signature}_{\text{Trent-Bob}}$

I want to transfer 20 coins to Diana

DSL at UCSB

Double Spending Prevention

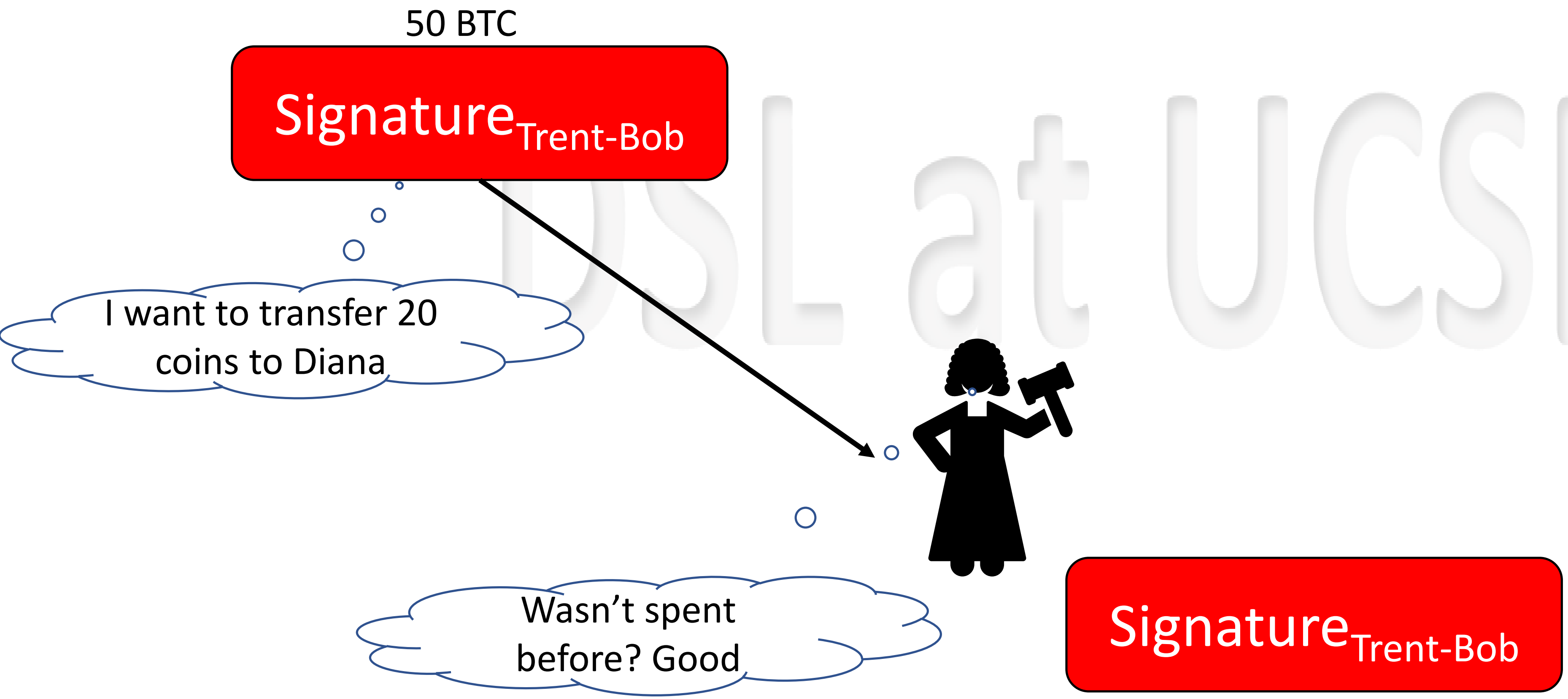
- Centralized
  - Transactions on coins go through a trusted 3rd party (Trent)



DSL at UCSB

Double Spending Prevention

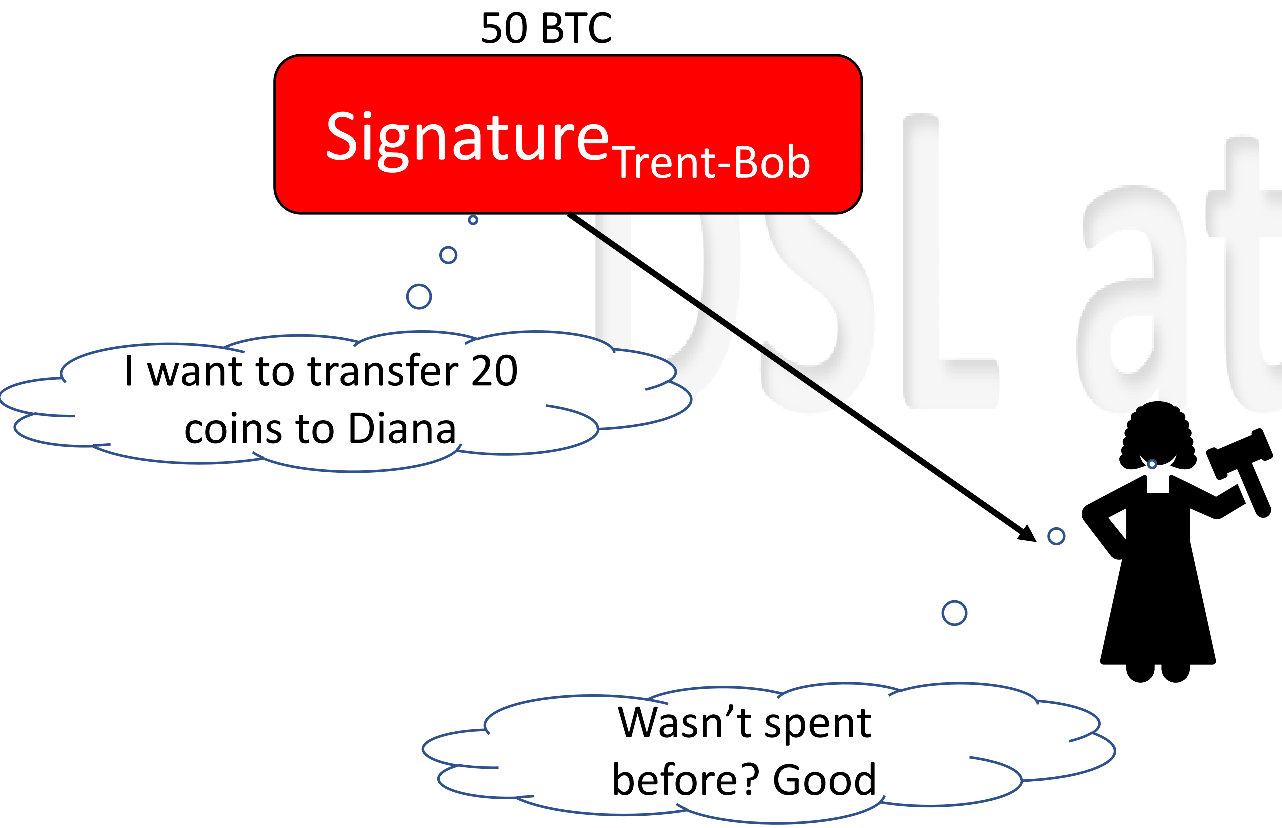
- Centralized
  - Transactions on coins go through a trusted 3rd party (Trent)



DSL at UCSB

Double Spending Prevention

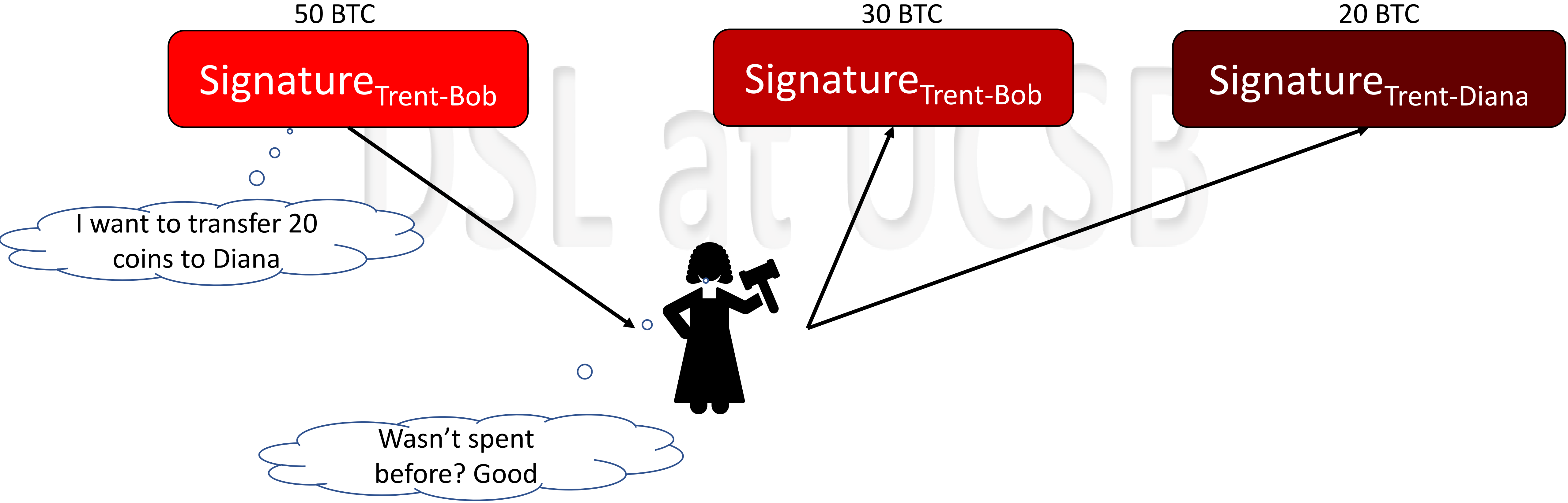
- Centralized
  - Transactions on coins go through a trusted 3rd party (Trent)



DSL at UCSB

Double Spending Prevention

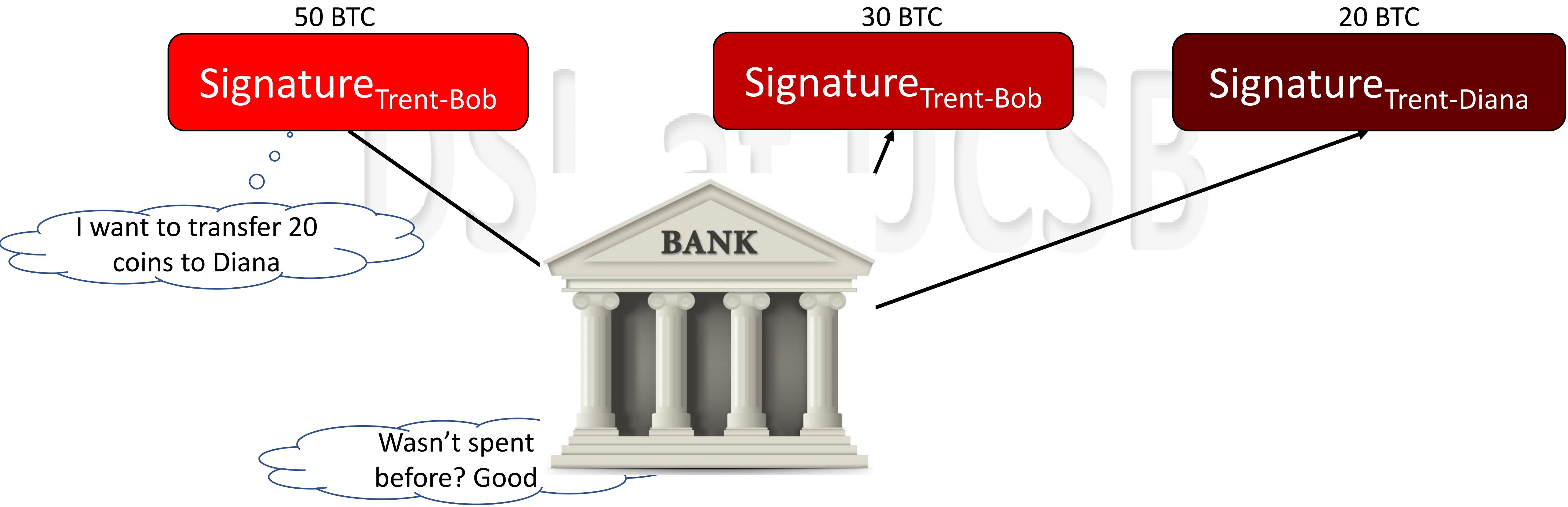
- Centralized
  - Transactions on coins go through a trusted 3rd party (Trent)



DSL at UCSB

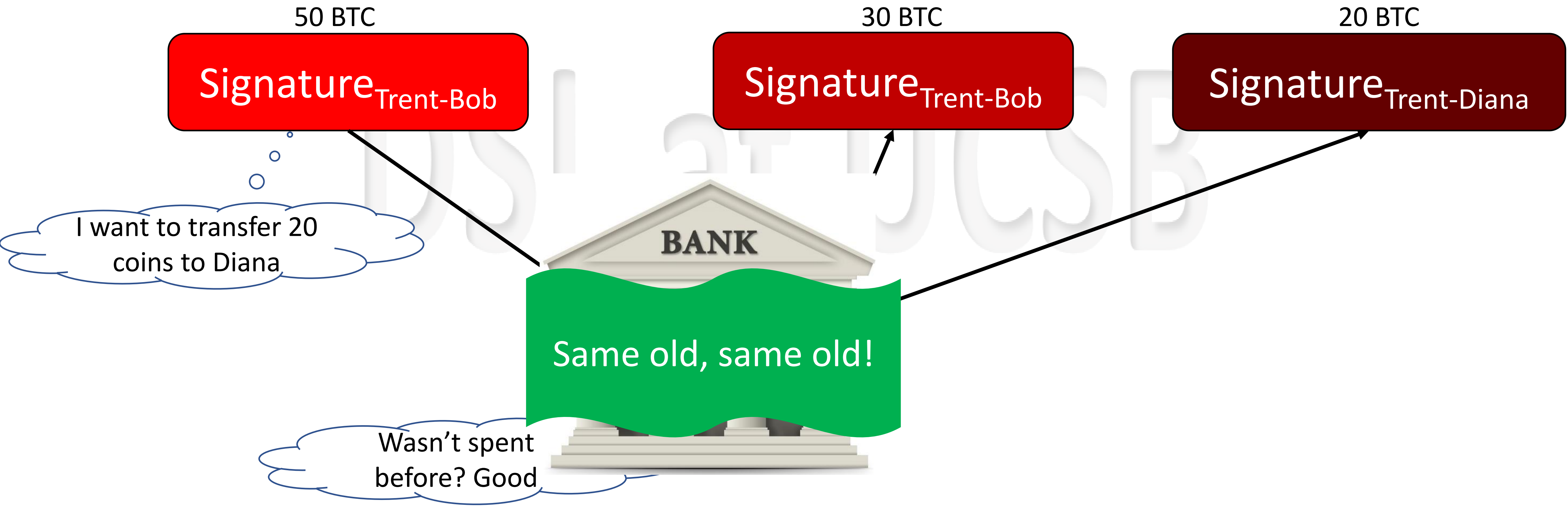
Double Spending Prevention

- Centralized
  - Transactions on coins go through a trusted 3rd party (Trent)



Double Spending Prevention

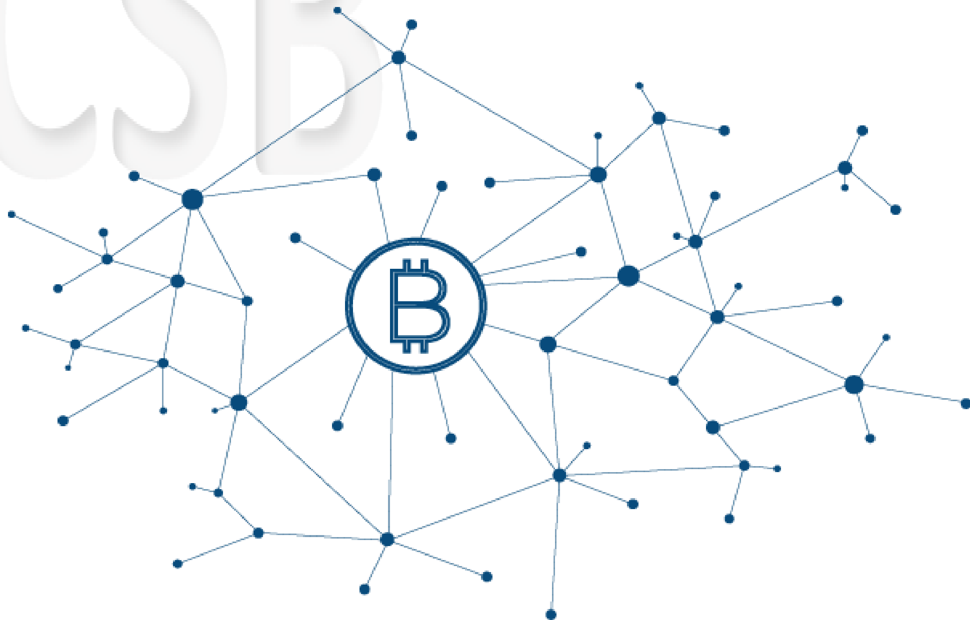
- Centralized
  - Transactions on coins go through a trusted 3rd party (Trent)



Double Spending Prevention

- Decentralized

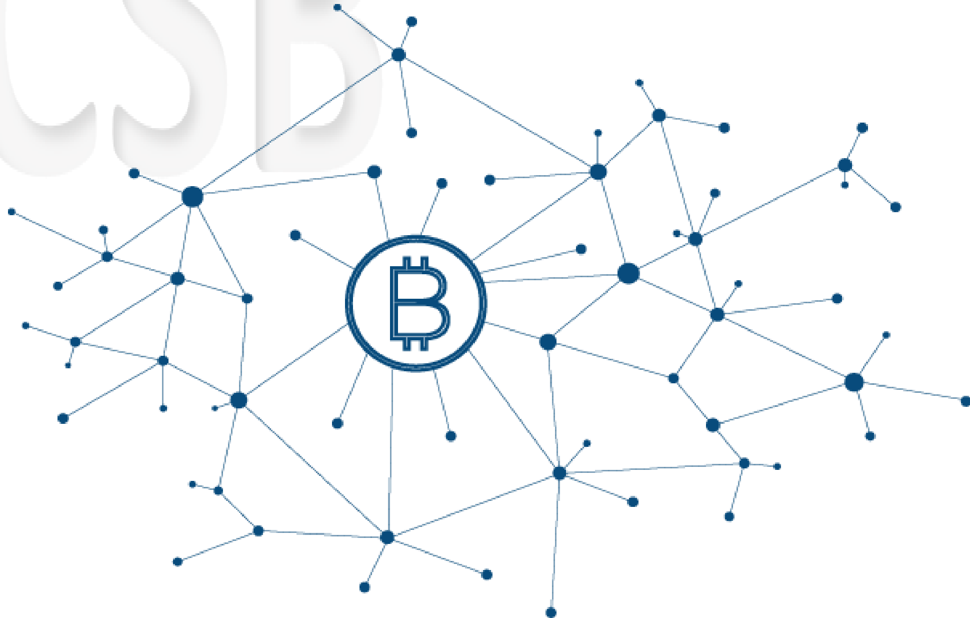
DSL at UCSB



Double Spending Prevention

- Decentralized
  - A network of nodes maintains a ledger

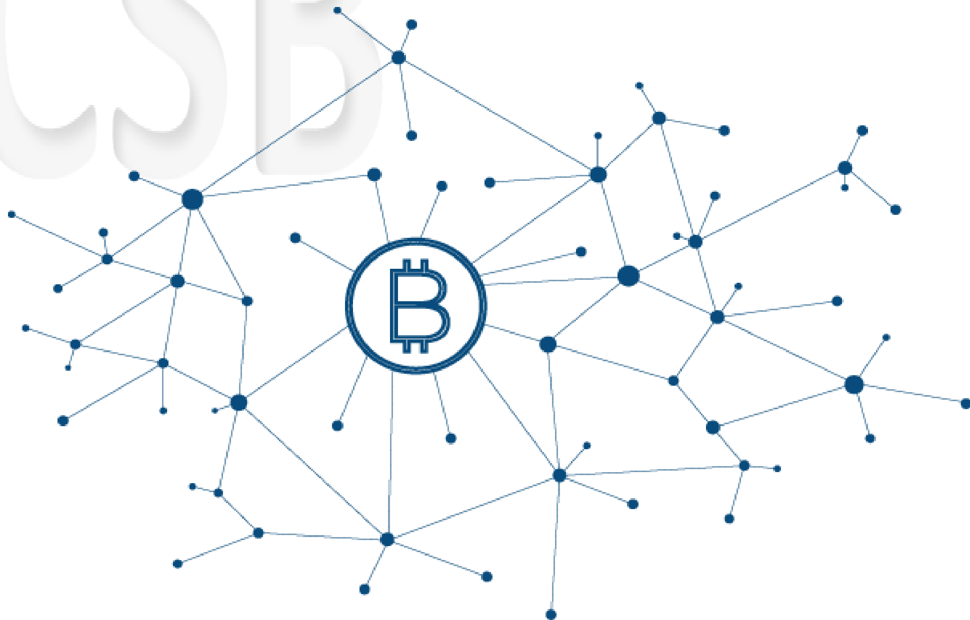
DSL at UCSB



Double Spending Prevention

- Decentralized
  - A network of nodes maintains a ledger
  - Network nodes work to agree on transactions order
    - Serializing transactions on every coin prevents double spending

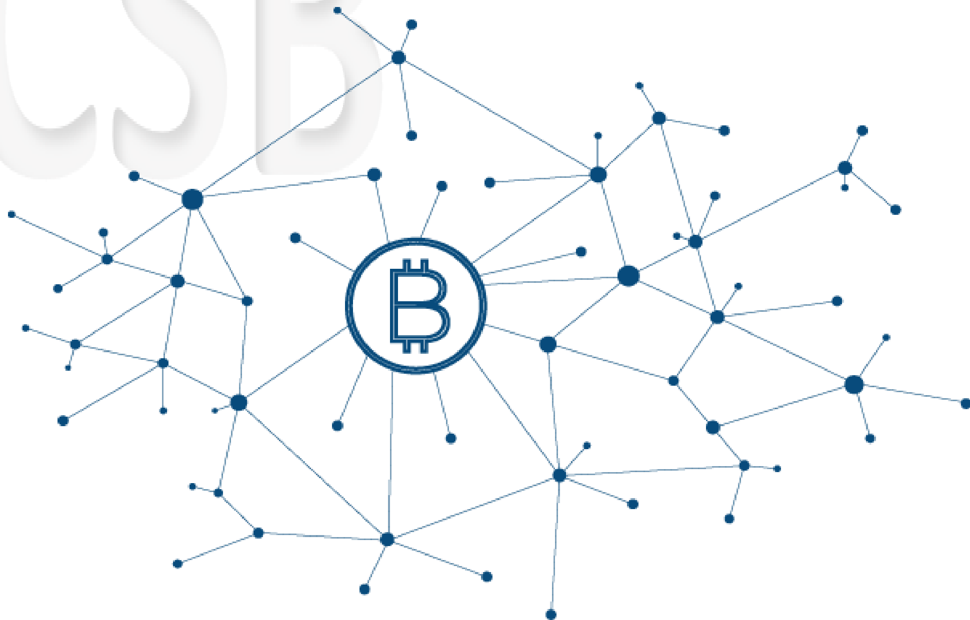
DSL at UCSB



Double Spending Prevention

- Decentralized
  - A network of nodes maintains a ledger
  - Network nodes work to agree on transactions order
    - Serializing transactions on every coin prevents double spending
  - What is the ledger?

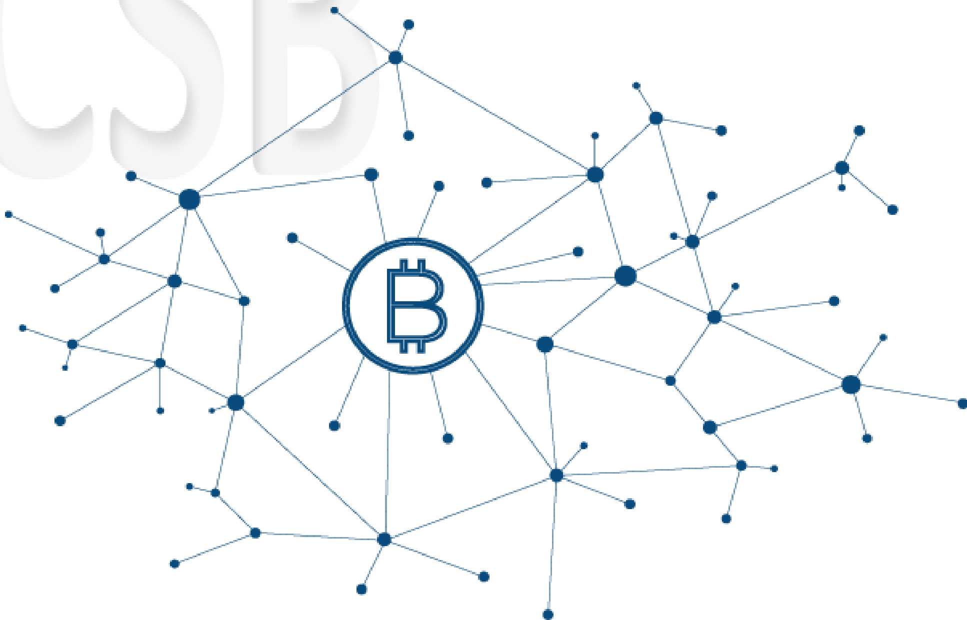
DSL at UCSB



Double Spending Prevention

- Decentralized
  - A network of nodes maintains a ledger
  - Network nodes work to agree on transactions order
    - Serializing transactions on every coin prevents double spending
  - What is the ledger?
  - How to agree on transaction order?

DSL at UCSB



Double Spending Prevention

- Decentralized
  - A network of nodes maintains a ledger
  - Network nodes work to agree on transactions order
    - Serializing transactions on every coin prevents double spending
  - What is the ledger?
  - How to agree on transaction order?
  - What incentives network nodes to maintain the ledger?

DSL at UCSB

What is the Ledger?

DSL at UCSB

What is the Ledger?

- Blockchain

DSL at UCSB

What is the Ledger?

- Blockchain

DSL at UCSB

What is the Ledger?

- Blockchain 

- Transactions are grouped into blocks

DSL at UCSB

What is the Ledger?

- Blockchain 

- Transactions are grouped into blocks
  - Blocks are chained to each other through pointers (Hence blockchain)

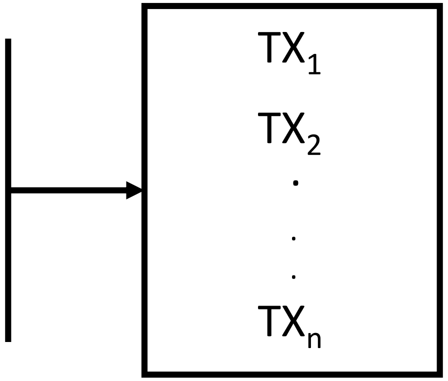
DSL at UCSB

What is the Ledger?

- Blockchain

- Transactions are grouped into blocks
  - Blocks are chained to each other through pointers (Hence blockchain)

DSL at UCSB

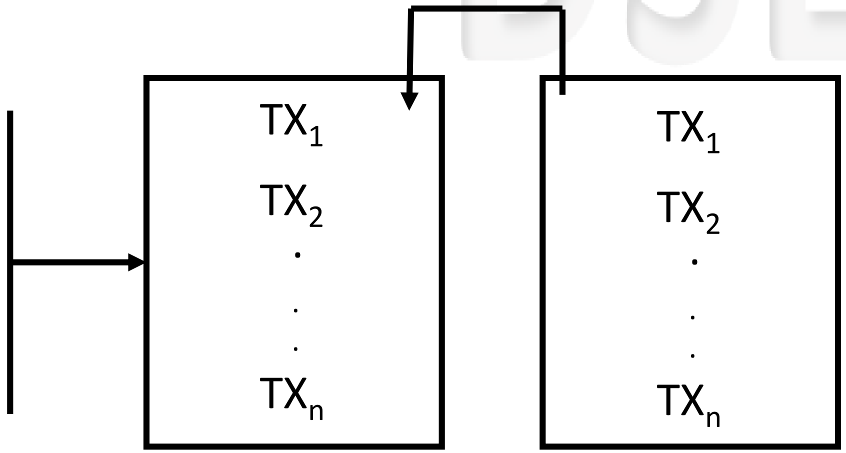


What is the Ledger?

- Blockchain

- Transactions are grouped into blocks
    - Blocks are chained to each other through pointers (Hence blockchain)

DSL at UCSB



DSL

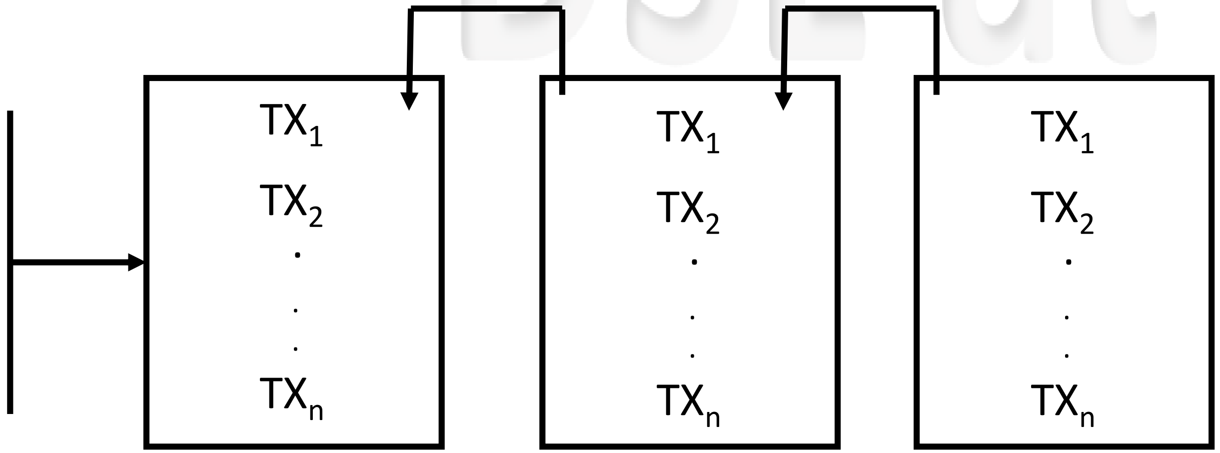
UCSB

What is the Ledger?

- Blockchain

- Transactions are grouped into blocks
  - Blocks are chained to each other through pointers (Hence blockchain)

DSL at UCSB

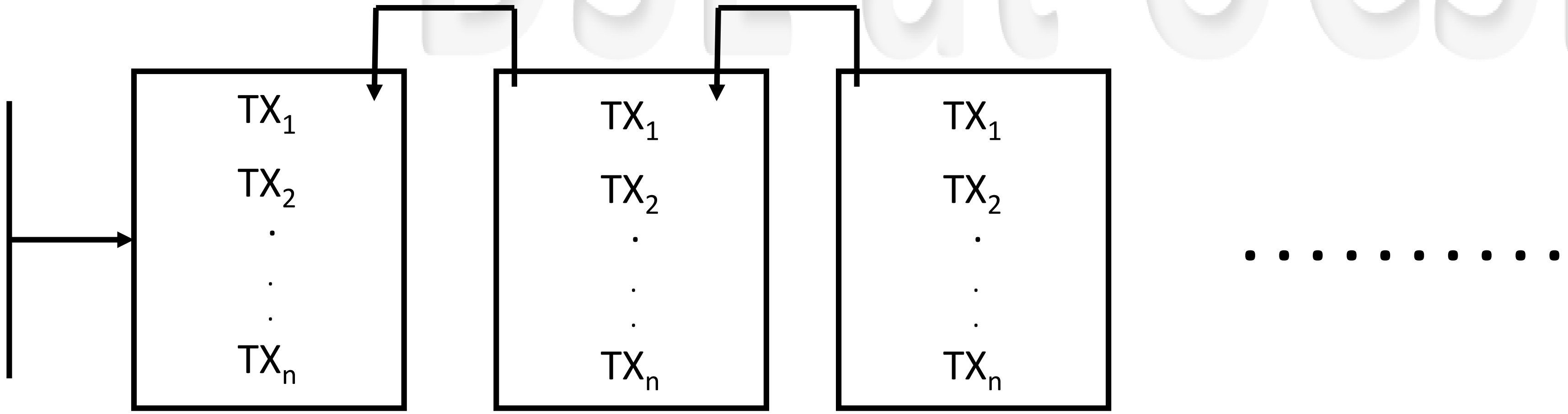


What is the Ledger?

- Blockchain

- Transactions are grouped into blocks
  - Blocks are chained to each other through pointers (Hence blockchain)

DSL at UCSB

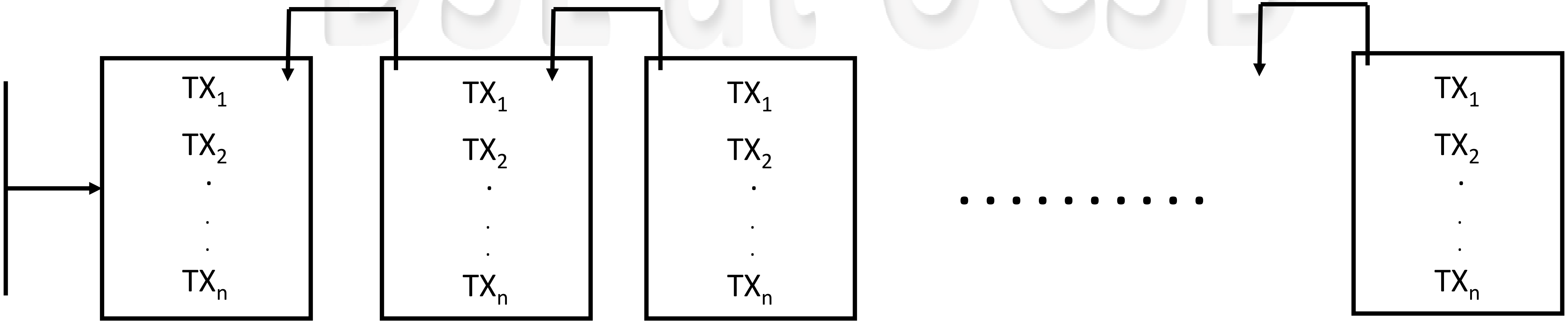


What is the Ledger?

- Blockchain 

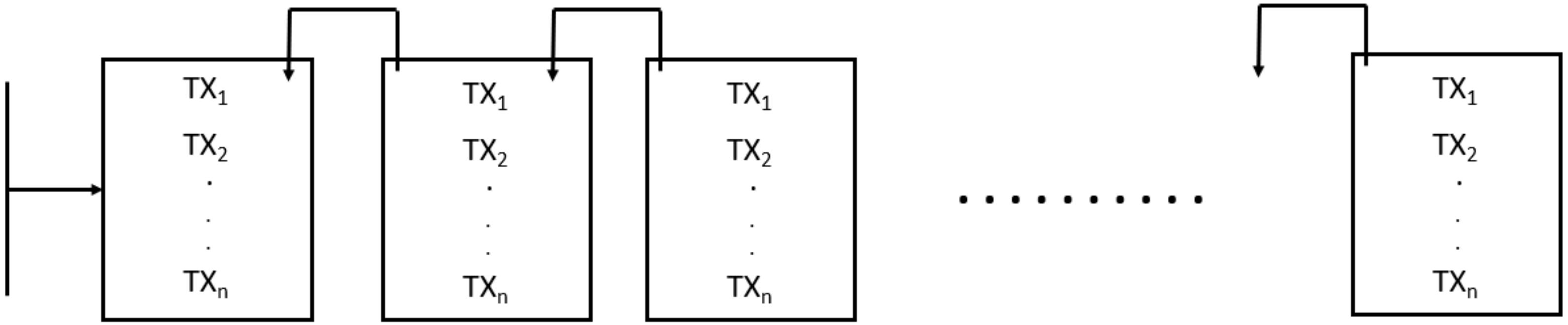
- Transactions are grouped into blocks
  - Blocks are chained to each other through pointers (Hence blockchain)

DSL at UCSB



The Ledger's What About's?

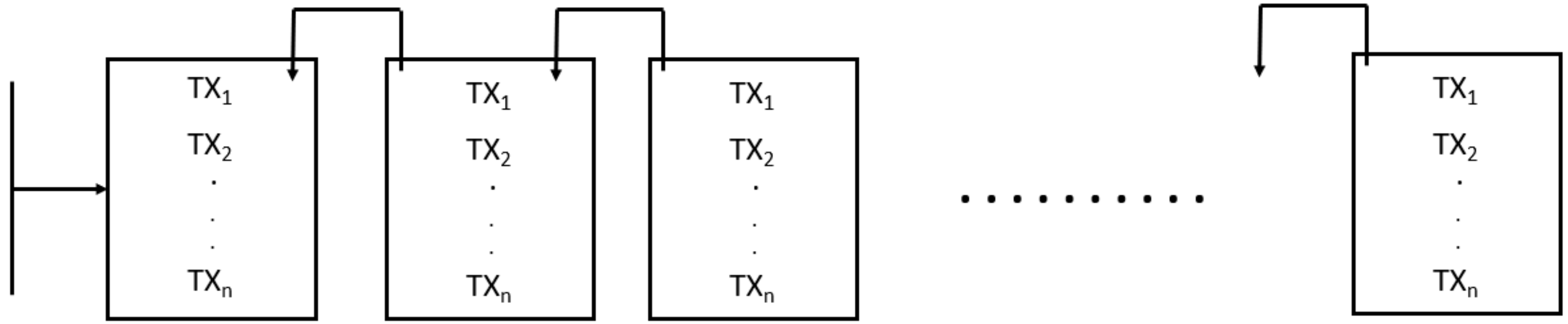
DSL at UCSB



The Ledger’s What About's?

- Where is the ledger stored?

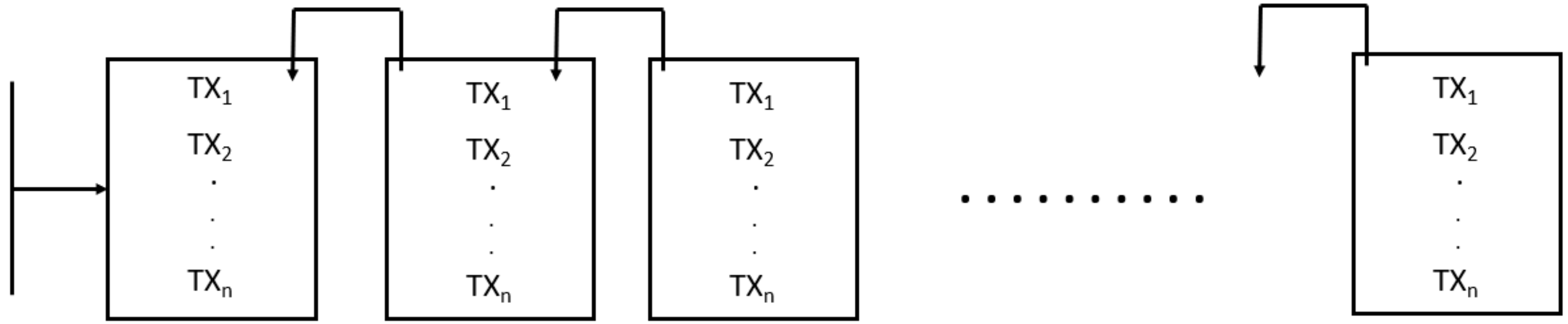
DSL at UCSB



The Ledger’s What About's?

- Where is the ledger stored?
  - Each network node maintains its copy of the ledger

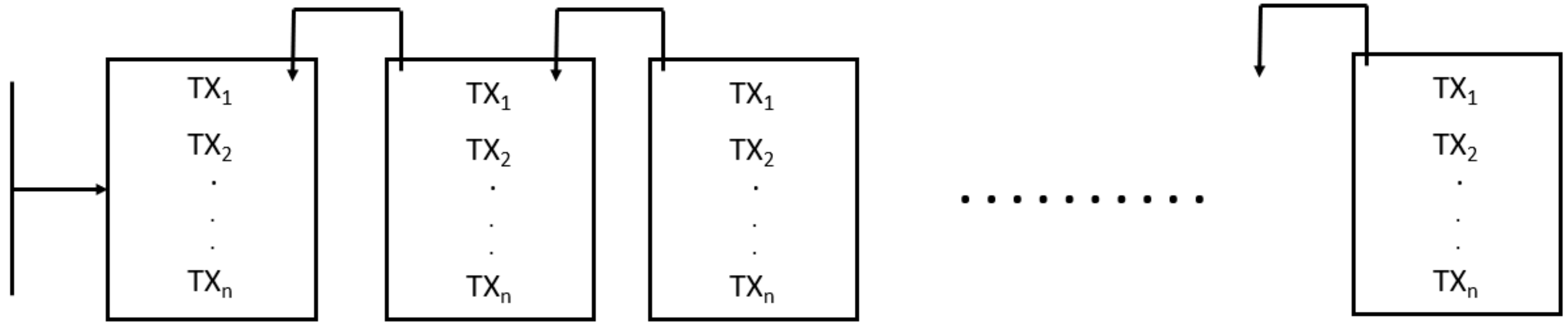
DSL at UCSB



The Ledger’s What About's?

- Where is the ledger stored?
  - Each network node maintains its copy of the ledger
- How is the ledger tamper-free?

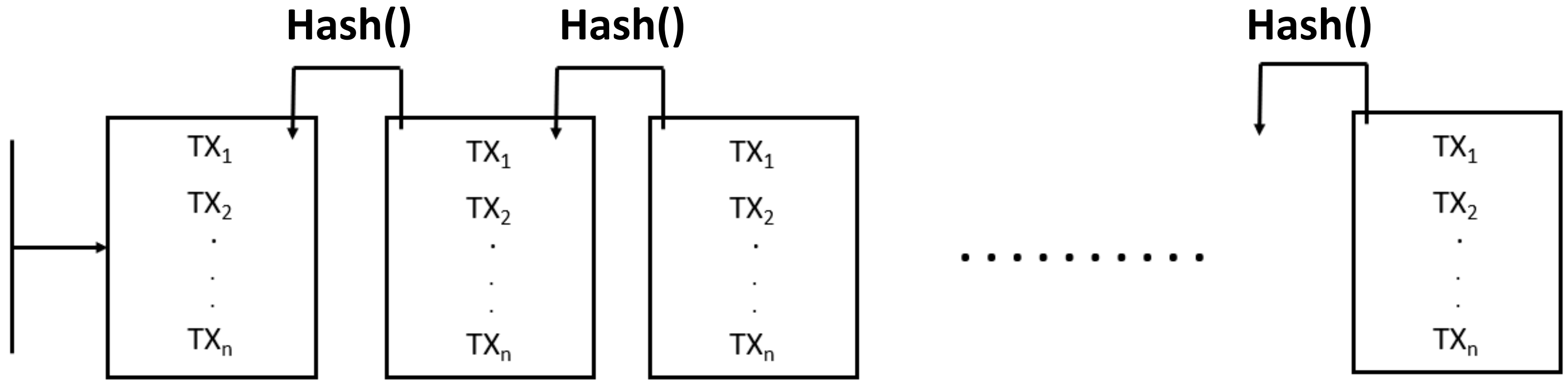
DSL at UCSB



The Ledger's What About's?

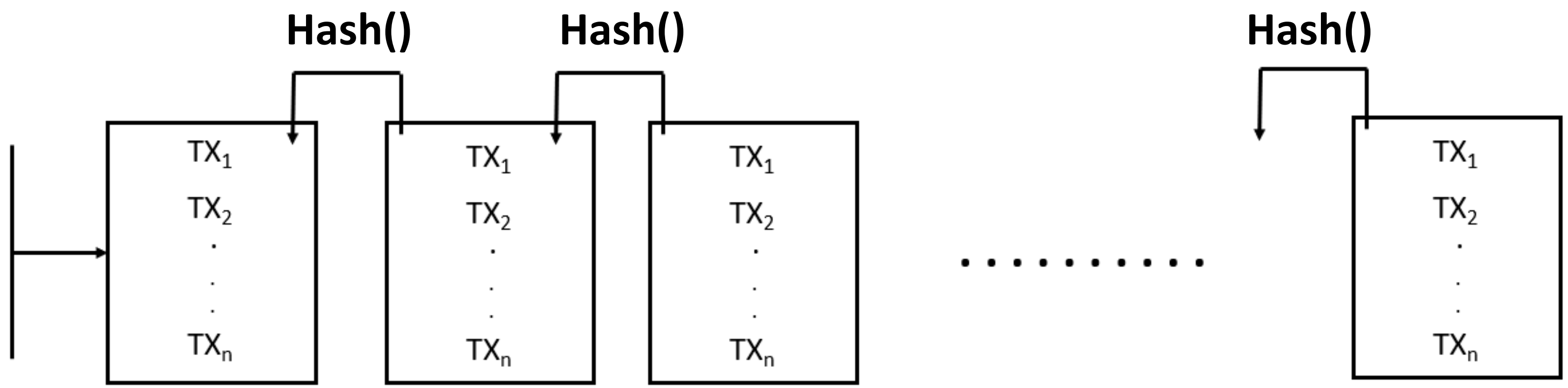
- Where is the ledger stored?
  - Each network node maintains its copy of the ledger
- How is the ledger tamper-free?
  1. Blocks are connected through **hash-pointers**

DSL at UCSB



The Ledger's What About's?

- Where is the ledger stored?
  - Each network node maintains its copy of the ledger
- How is the ledger tamper-free?
  1. Blocks are connected through **hash-pointers**
     - Each block contains the hash of the previous block
     - This hash gives each block its location in the blockchain
     - Tampering with the content of any block can easily be detected (**is this enough? NO**)

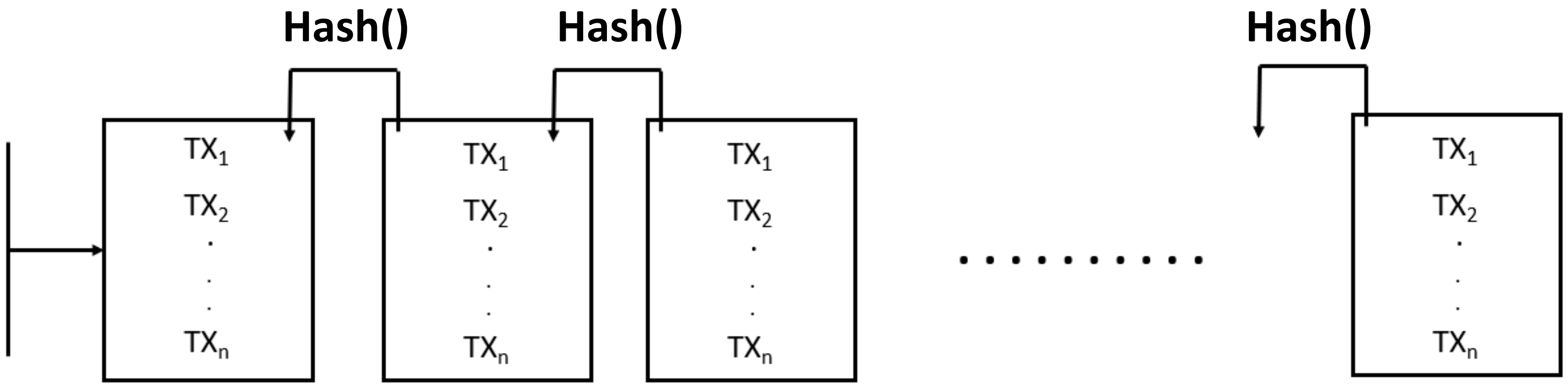


DSL at UCSB

DSL

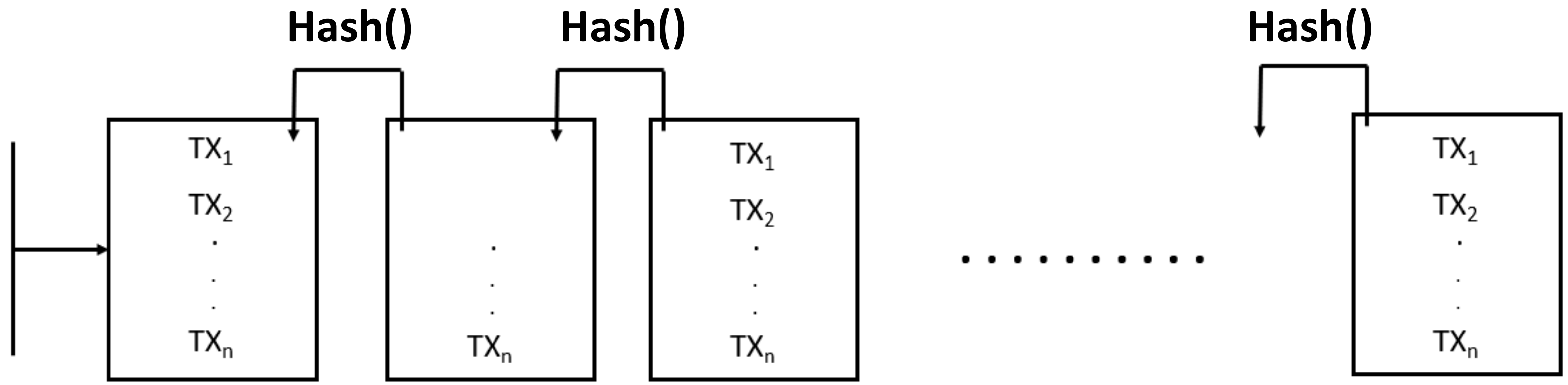
UCSB

Tampering with the Ledger



DSL at UCSB

Tampering with the Ledger

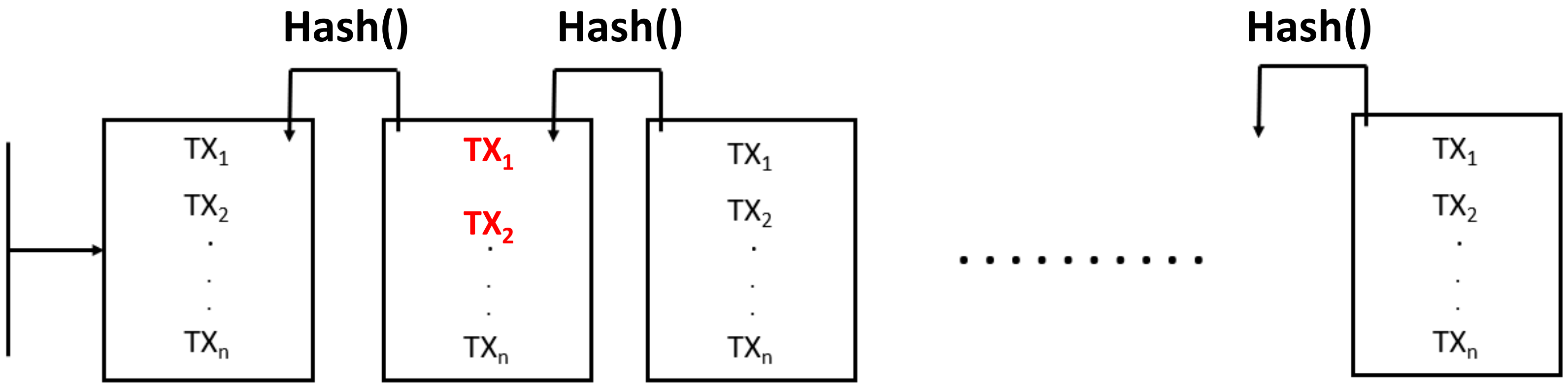


DSL at UCSB

DSL

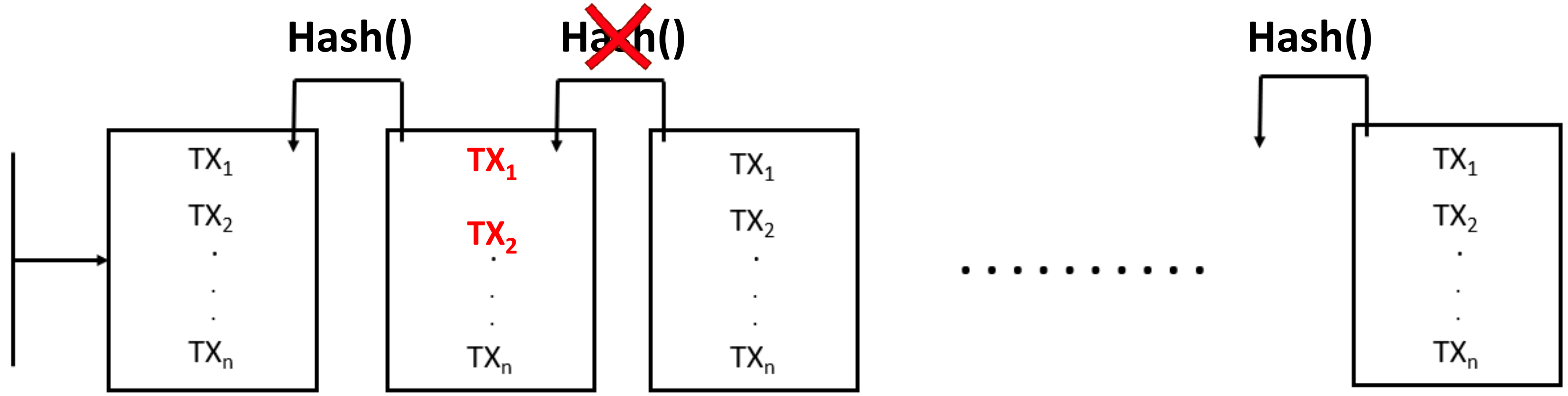
UCSB

Tampering with the Ledger



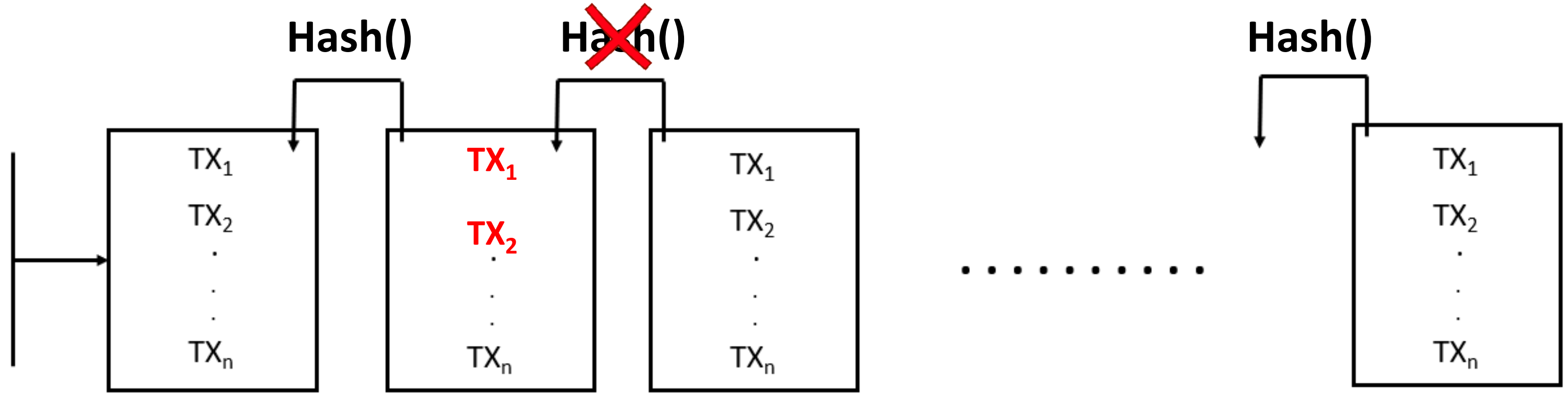
DSL at UCSB

Tampering with the Ledger



DSL at UCSB

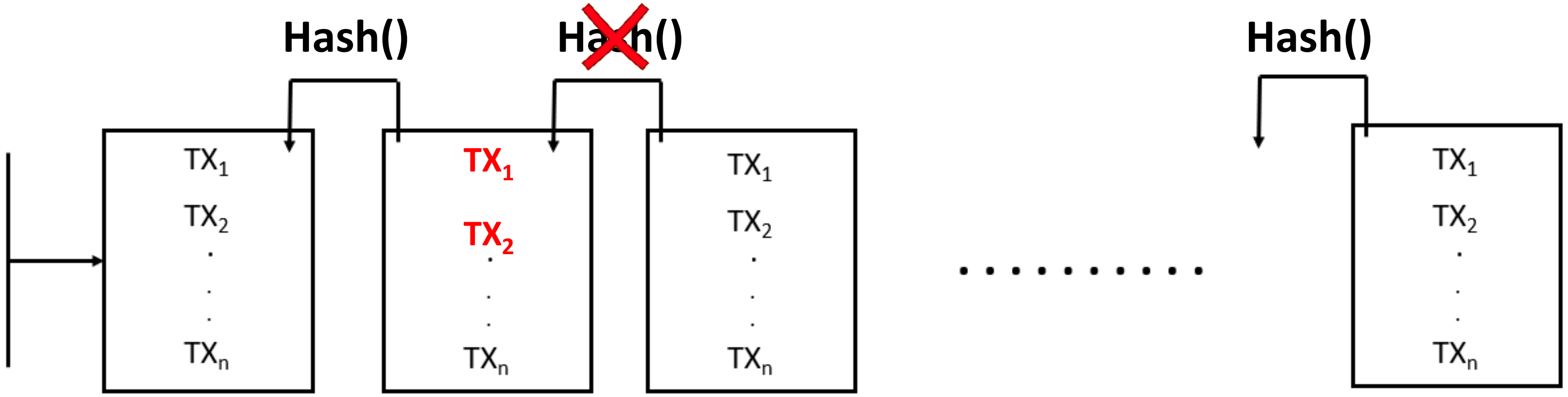
Tampering with the Ledger



Inconsistent Blockchain

DSL at UCSB

Tampering with the Ledger

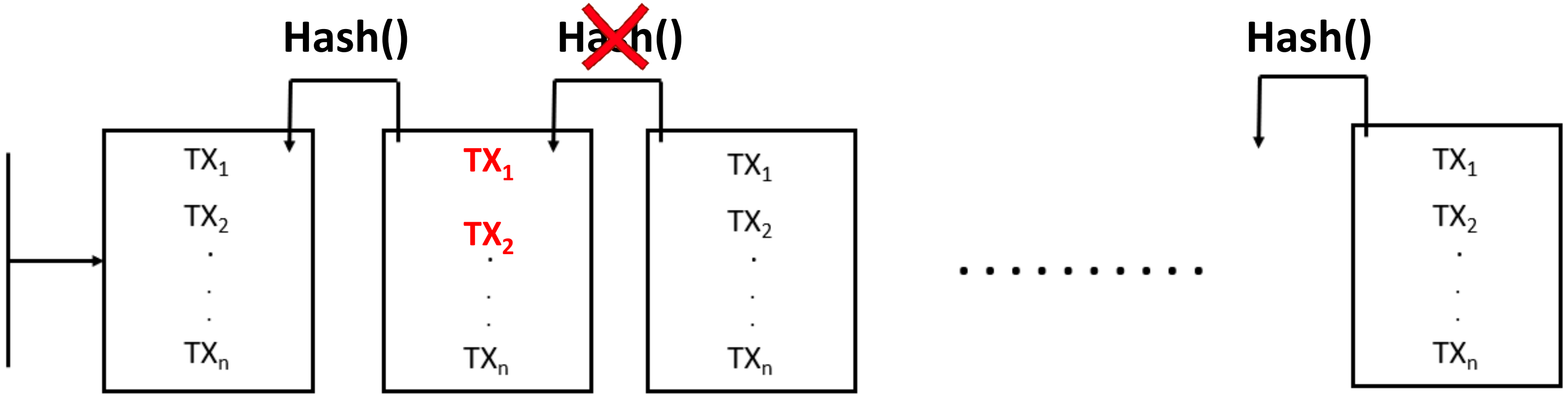


Inconsistent Blockchain

However,

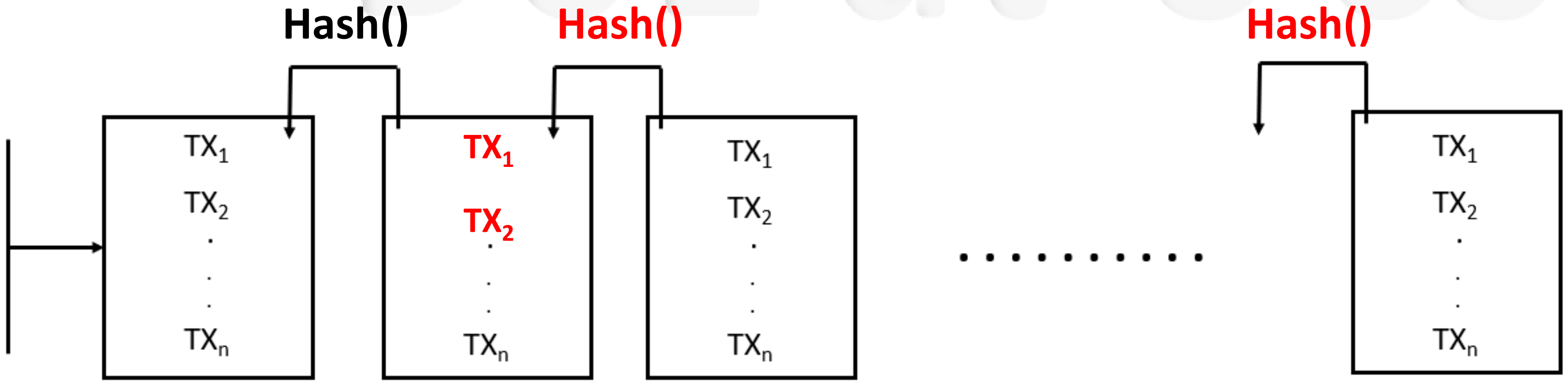
DSL at UCSB

Tampering with the Ledger



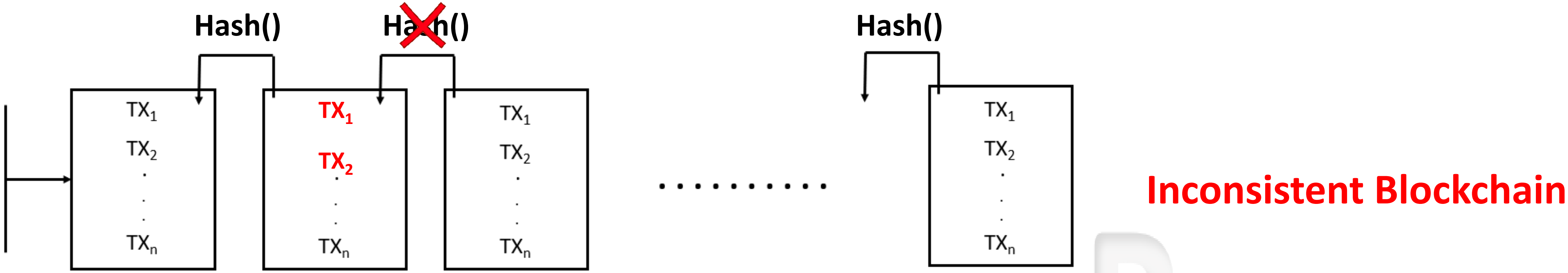
Inconsistent Blockchain

However,



DSL at UCSB

Tampering with the Ledger



However,

DSL at UCSB

The Ledger's What About's?

- How is the ledger tamper-free?
  1. Blocks are connected through **hash-pointers**
     - Each block contains the hash of the previous block
     - This hash gives each block its location in the blockchain
     - Tampering the content of any block can easily be detected (**is this enough? NO**)

DSL at UCSB

The Ledger's What About's?

- How is the ledger tamper-free?
  1. Blocks are connected through **hash-pointers**
     - Each block contains the hash of the previous block
     - This hash gives each block its location in the blockchain
     - Tampering the content of any block can easily be detected (**is this enough? NO**)
  2. Replacing a consistent blockchain with another tampered consistent block chain should be **made very hard**, How?

DSL at UCSB

Network Nodes Big Picture

Network Nodes Big Picture

Network Nodes Big Picture

Making Progress

DSL at UCSB

Making Progress

- The ledger is fully replicated to all network nodes

DSL at UCSB

Making Progress

- The ledger is fully replicated to all network nodes
- To make progress:

DSL at UCSB

Making Progress

- The ledger is fully replicated to all network nodes
- To make progress:
  - Network nodes group new transactions into a block

DSL at UCSB

Making Progress

- The ledger is fully replicated to all network nodes
- To make progress:
  - Network nodes group new transactions into a block
  - Blocks are fixed in size (1MB)

DSL at UCSB

Making Progress

- The ledger is fully replicated to all network nodes
- To make progress:
  - Network nodes group new transactions into a block
  - Blocks are fixed in size (1MB)
  - Network nodes **validate** new transactions to make sure that:

DSL at UCSB

Making Progress

- The ledger is fully replicated to all network nodes
- To make progress:
  - Network nodes group new transactions into a block
  - Blocks are fixed in size (1MB)
  - Network nodes **validate** new transactions to make sure that:
    - Transactions on the new block do not conflict with **each other**
    - Transactions on the new block do not conflict with **previous blocks transactions**

DSL at UCSB

Making Progress

- The ledger is fully replicated to all network nodes
- To make progress:
  - Network nodes group new transactions into a block
  - Blocks are fixed in size (1MB)
  - Network nodes **validate** new transactions to make sure that:
    - Transactions on the new block do not conflict with **each other**
    - Transactions on the new block do not conflict with **previous blocks transactions**
  - Network nodes need to agree on the next block to be added to the blockchain

DSL at UCSB

Making Progress

- The ledger is fully replicated to all network nodes
- To make progress:
  - Network nodes group new transactions into a block
  - Blocks are fixed in size (1MB)
  - Network nodes **validate** new transactions to make sure that:
    - Transactions on the new block do not conflict with **each other**
    - Transactions on the new block do not conflict with **previous blocks transactions**
  - Network nodes need to agree on the next block to be added to the blockchain

Consensus

DSL at UCSB

Consensus

- Types of systems: synchronous and asynchronous

DSL at UCSB

Consensus

- Types of systems: synchronous and asynchronous
- Problem statement: given n processes and one leader:
  - Agreement: all correct processes agree on the same value
  - Validity: If initiator does not fail, all correct processes agree on its value

DSL at UCSB

Consensus

- Types of systems: synchronous and asynchronous
- Problem statement: given n processes and one leader:
  - Agreement: all correct processes agree on the same value
  - Validity: If initiator does not fail, all correct processes agree on its value
- Types of failure:
  - Crash
  - Malicious (or Byzantine)

DSL at UCSB

Consensus

- Types of systems: synchronous and asynchronous
- Problem statement: given n processes and one leader:
  - Agreement: all correct processes agree on the same value
  - Validity: If initiator does not fail, all correct processes agree on its value
- Types of failure:
  - Crash
  - Malicious (or Byzantine)
- Important Impossibility Results:

DSL at UCSB

Consensus

- Types of systems: synchronous and asynchronous
- Problem statement: given n processes and one leader:
  - Agreement: all correct processes agree on the same value
  - Validity: If initiator does not fail, all correct processes agree on its value
- Types of failure:
  - Crash
  - Malicious (or Byzantine)
- Important Impossibility Results:
  - **FLP,** in **asynchronous** systems:
    - With even 1 crash failure, termination isn't guaranteed (no liveness)

DSL at UCSB

Consensus

- Types of systems: synchronous and asynchronous
- Problem statement: given n processes and one leader:
  - Agreement: all correct processes agree on the same value
  - Validity: If initiator does not fail, all correct processes agree on its value
- Types of failure:
  - Crash
  - Malicious (or Byzantine)
- Important Impossibility Results:
  - **FLP,** in **asynchronous** systems:
    - With even 1 crash failure, termination isn't guaranteed (no liveness)
  - **Synchronous** systems:
    - Termination is guaranteed if number of failed malicious processes (f) is at most 1/3 n

DSL at UCSB

(Multi-) Paxos

DSL at UCSB

(Multi-) Paxos

- Paxos is a consensus algorithm
  - Processes want to agree on a value (e.g., the next block to be added to the chain)

DSL at UCSB

(Multi-) Paxos

- Paxos is a consensus algorithm
  - Processes want to agree on a value (e.g., the next block to be added to the chain)
- Paxos is currently used to manage local data in global-scale systems
  - Spanner [OSDI’12, SIGMOD’17], Megastore [CIDR’11], etc

DSL at UCSB

(Multi-) Paxos

- Paxos is a consensus algorithm
  - Processes want to agree on a value (e.g., the next block to be added to the chain)
- Paxos is currently used to manage local data in global-scale systems
  - Spanner [OSDI’12, SIGMOD’17], Megastore [CIDR’11], etc
- Multi-Paxos, simplified:

DSL at UCSB

(Multi-) Paxos

- Paxos is a consensus algorithm
  - Processes want to agree on a value (e.g., the next block to be added to the chain)
- Paxos is currently used to manage local data in global-scale systems
  - Spanner [OSDI’12, SIGMOD’17], Megastore [CIDR’11], etc
- Multi-Paxos, simplified:
  - Initially, a leader is elected by a majority quorum

DSL at UCSB

(Multi-) Paxos

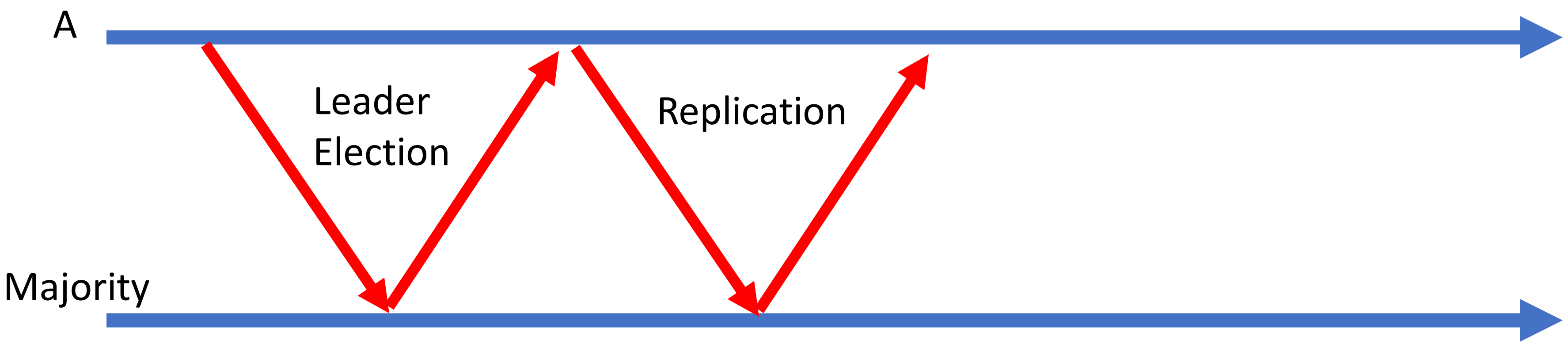
- Paxos is a consensus algorithm
  - Processes want to agree on a value (e.g., the next block to be added to the chain)
- Paxos is currently used to manage local data in global-scale systems
  - Spanner [OSDI’12, SIGMOD’17], Megastore [CIDR’11], etc
- Multi-Paxos, simplified:
  - Initially, a leader is elected by a majority quorum
  - **Replication:** Leader replicates new updates to a majority quorum

DSL at UCSB

(Multi-) Paxos

- Paxos is a consensus algorithm
  - Processes want to agree on a value (e.g., the next block to be added to the chain)
- Paxos is currently used to manage local data in global-scale systems
  - Spanner [OSDI'12, SIGMOD'17], Megastore [CIDR'11], etc
- Multi-Paxos, simplified:
  - Initially, a leader is elected by a majority quorum
  - **Replication:** Leader replicates new updates to a majority quorum

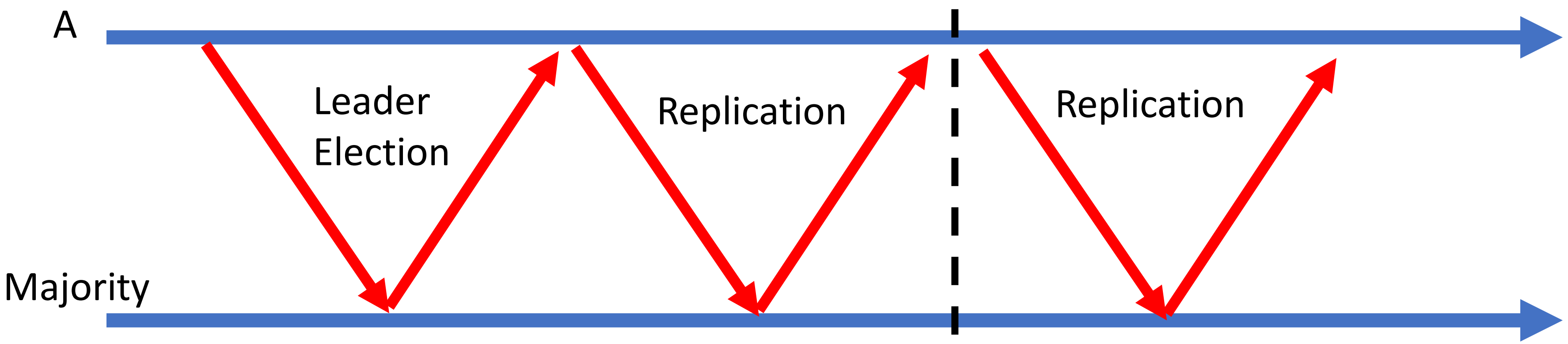
DSL at UCSB



(Multi-) Paxos

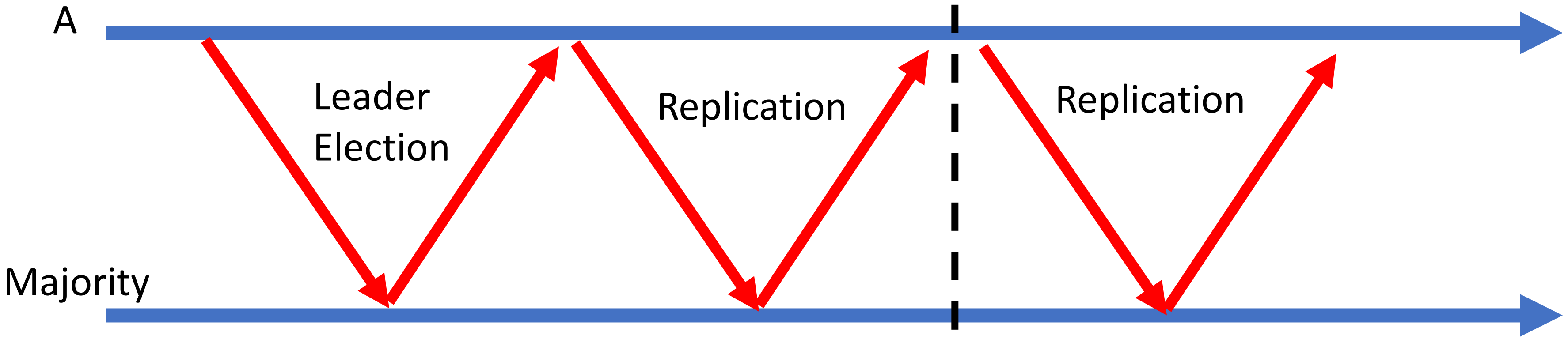
- Paxos is a consensus algorithm
  - Processes want to agree on a value (e.g., the next block to be added to the chain)
- Paxos is currently used to manage local data in global-scale systems
  - Spanner [OSDI'12, SIGMOD'17], Megastore [CIDR'11], etc
- Multi-Paxos, simplified:
  - Initially, a leader is elected by a majority quorum
  - **Replication:** Leader replicates new updates to a majority quorum

DSL at UCSB



(Multi-) Paxos

- Paxos is a consensus algorithm
  - Processes want to agree on a value (e.g., the next block to be added to the chain)
- Paxos is currently used to manage local data in global-scale systems
  - Spanner [OSDI'12, SIGMOD'17], Megastore [CIDR'11], etc
- Multi-Paxos, simplified:
  - Initially, a leader is elected by a majority quorum
  - **Replication:** Leader replicates new updates to a majority quorum
  - **Leader Election:** If the leader fails, a new leader is elected



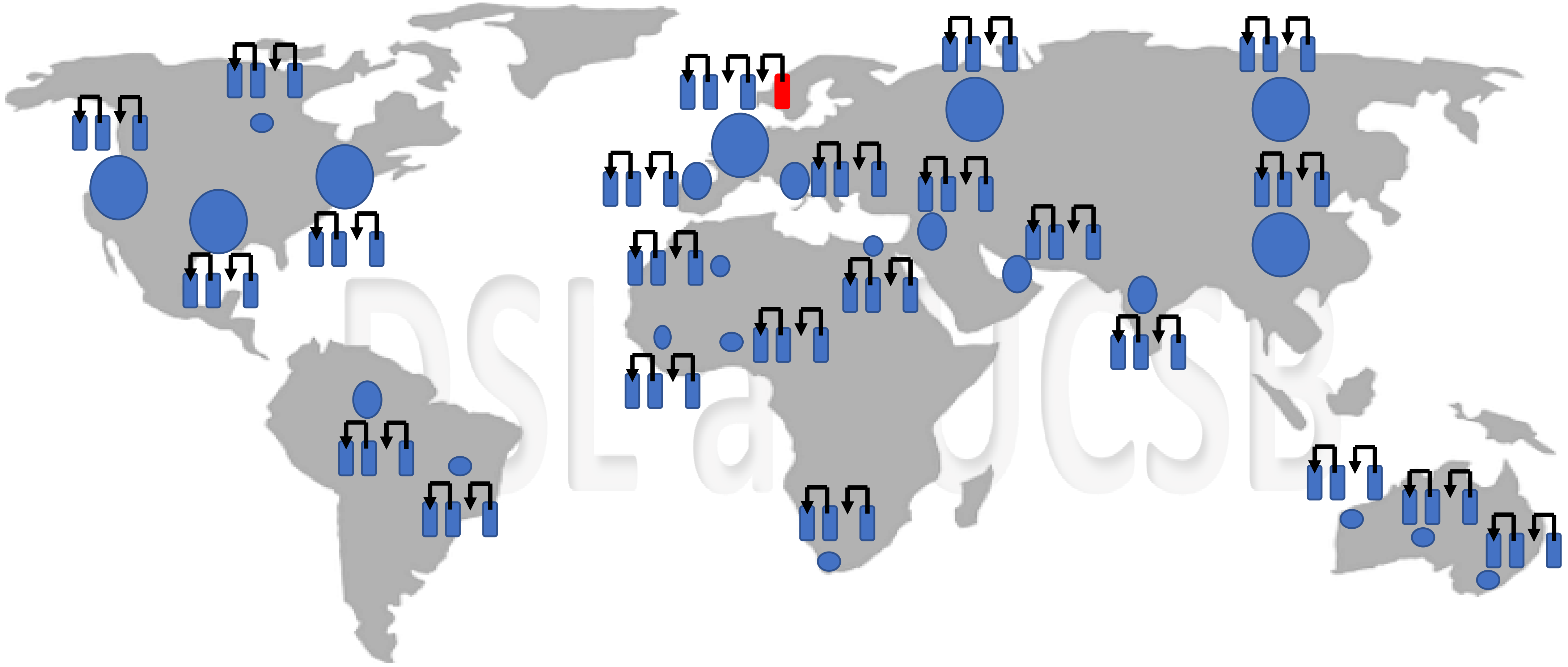
DSL at UCSB

Can Network Nodes Use Paxos?

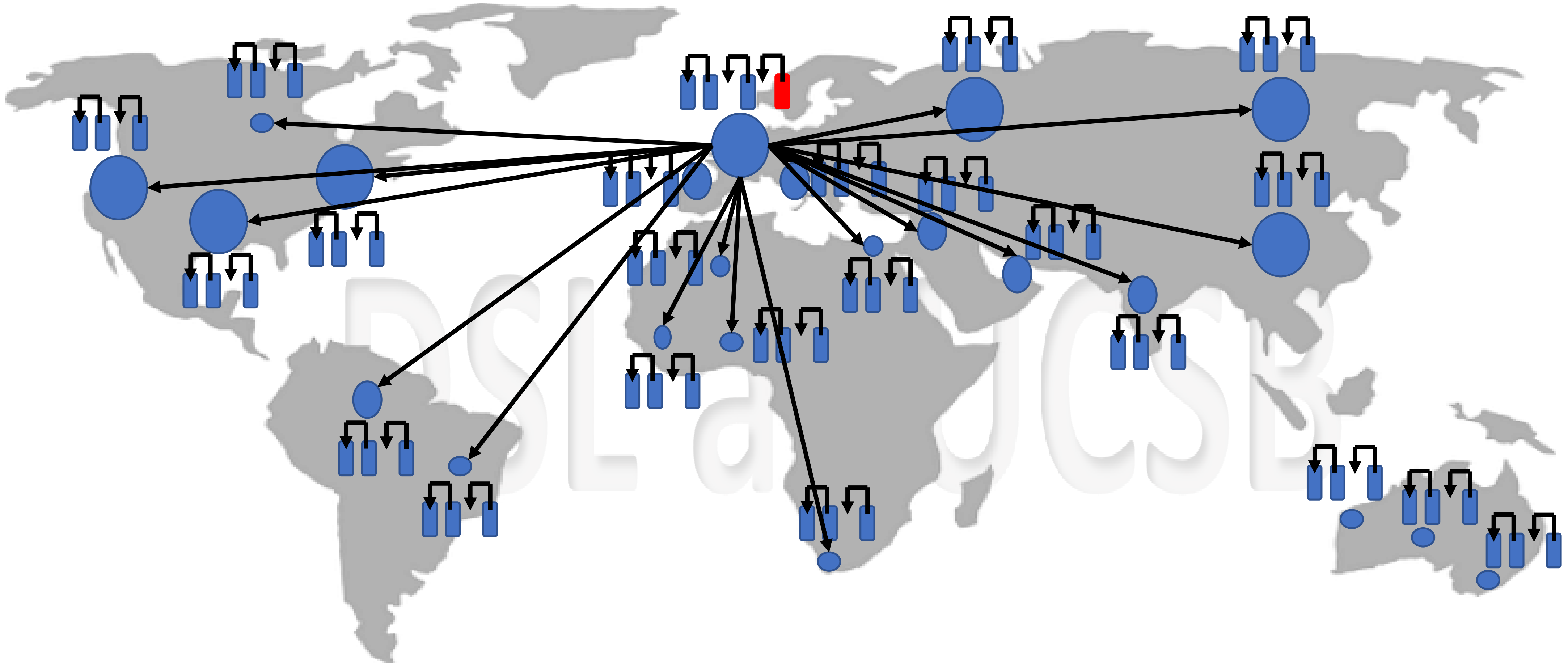
DSL

UCSB

Can Network Nodes Use Paxos?



Can Network Nodes Use Paxos?



Paxos Consensus

DSL at UCSB

Paxos Consensus

- All **participants** should be known **a priori**

DSL at UCSB

Paxos Consensus

- All **participants** should be known **a priori**
  - **Permissioned** vs **Permissionless** settings

DSL at UCSB

Paxos Consensus

- All **participants** should be known **a priori**
  - **Permissioned** vs **Permissionless** settings
  - **Permissionless** setting:
    - Network nodes freely join or leave the network at anytime

DSL at UCSB

Paxos Consensus

- All **participants** should be known **a priori**
  - **Permissioned** vs **Permissionless** settings
  - **Permissionless** setting:
    - Network nodes freely join or leave the network at anytime
- Tolerates only **Crash** failures

DSL at UCSB

Paxos Consensus

- All **participants** should be known **a priori**
  - **Permissioned** vs **Permissionless** settings
  - **Permissionless** setting:
    - Network nodes freely join or leave the network at anytime
- Tolerates only **Crash** failures
  - However, network nodes can be **Malicious**

DSL at UCSB

Paxos Consensus

- All **participants** should be known **a priori**
  - **Permissioned** vs **Permissionless** settings
  - **Permissionless** setting:
    - Network nodes freely join or leave the network at anytime
- Tolerates only **Crash** failures
  - However, network nodes can be **Malicious**
  - To make progress, at least **1/2** of the participants should be **alive**
  - Progress is not guaranteed (FLP impossibility)

DSL at UCSB

DSL UCSB

Paxos Consensus

- All **participants** should be known **a priori**
  - **Permissioned** vs **Permissionless** settings
  - **Permissionless** setting:
    - Network nodes freely join or leave the network at anytime
- Tolerates only **Crash** failures
  - However, network nodes can be **Malicious**
  - To make progress, at least **1/2** of the participants should be **alive**
  - Progress is not guaranteed (FLP impossibility)
- Also, Paxos has high network overhead

DSL at UCSB

DSL

UCSB

Practical Byzantine Fault Tolerance (PBFT)

DSL at UCSB

Practical Byzantine Fault Tolerance (PBFT)

- Goal: Implement a deterministic replication service with **arbitrary malicious faults** in an asynchronous environment

DSL at UCSB

Practical Byzantine Fault Tolerance (PBFT)

- Goal: Implement a deterministic replication service with **arbitrary malicious faults** in an asynchronous environment
- No assumptions about faulty behavior
- No bounds on delays

DSL at UCSB

Practical Byzantine Fault Tolerance (PBFT)

- Goal: Implement a deterministic replication service with **arbitrary malicious faults** in an asynchronous environment
- No assumptions about faulty behavior
- No bounds on delays
- Provides **safety** in asynchronous system and assume eventual time bounds for **liveness**

DSL at UCSB

Practical Byzantine Fault Tolerance (PBFT)

- Goal: Implement a deterministic replication service with **arbitrary malicious faults** in an asynchronous environment
- No assumptions about faulty behavior
- No bounds on delays
- Provides **safety** in asynchronous system and assume eventual time bounds for **liveness**
- Assumptions:

DSL at UCSB

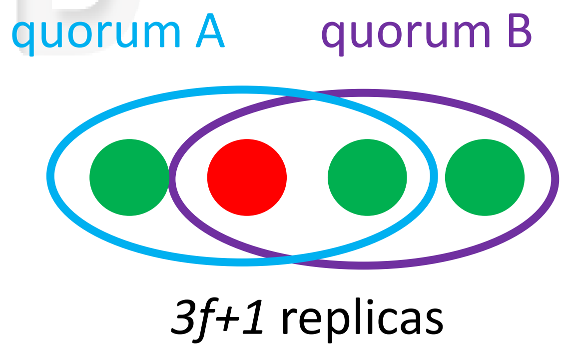
Practical Byzantine Fault Tolerance (PBFT)

- Goal: Implement a deterministic replication service with **arbitrary malicious faults** in an asynchronous environment
- No assumptions about faulty behavior
- No bounds on delays
- Provides **safety** in asynchronous system and assume eventual time bounds for **liveness**
- Assumptions:
  - 3f+1 replicas to tolerate f Byzantine faults (optimal)

DSL at UCSB

Practical Byzantine Fault Tolerance (PBFT)

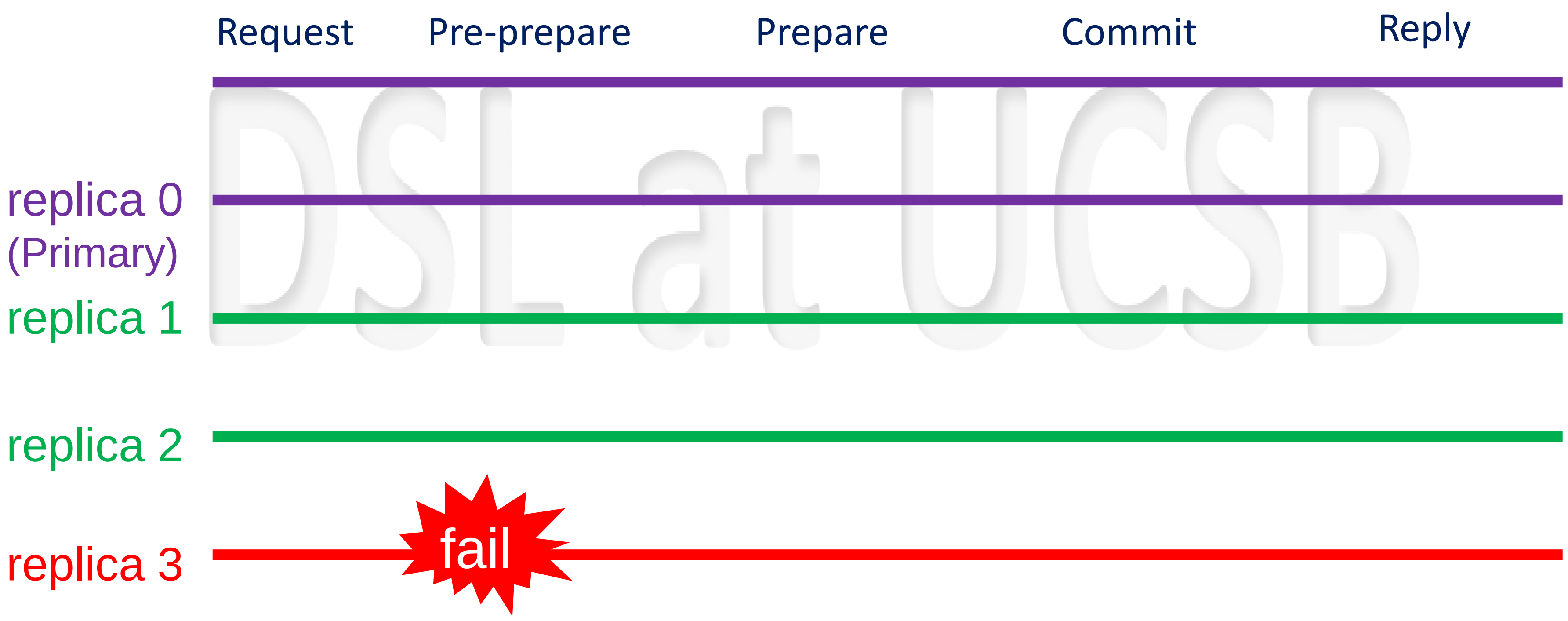
- Goal: Implement a deterministic replication service with **arbitrary malicious faults** in an asynchronous environment
- No assumptions about faulty behavior
- No bounds on delays
- Provides **safety** in asynchronous system and assume eventual time bounds for **liveness**
- Assumptions:
  - 3f+1 replicas to tolerate f Byzantine faults (optimal)
    - quorums have at least 2f+1 replicas
    - quorums intersect in f+1, hence have at least one correct replica
  - Strong cryptography
  - Only for liveness: eventual time bounds



DSL at UCSB

Algorithm

The algorithm has three main phases: (1) *pre-prepare* picks order of requests (2) *prepare* ensures order within views, (3) *commit* ensures order across views

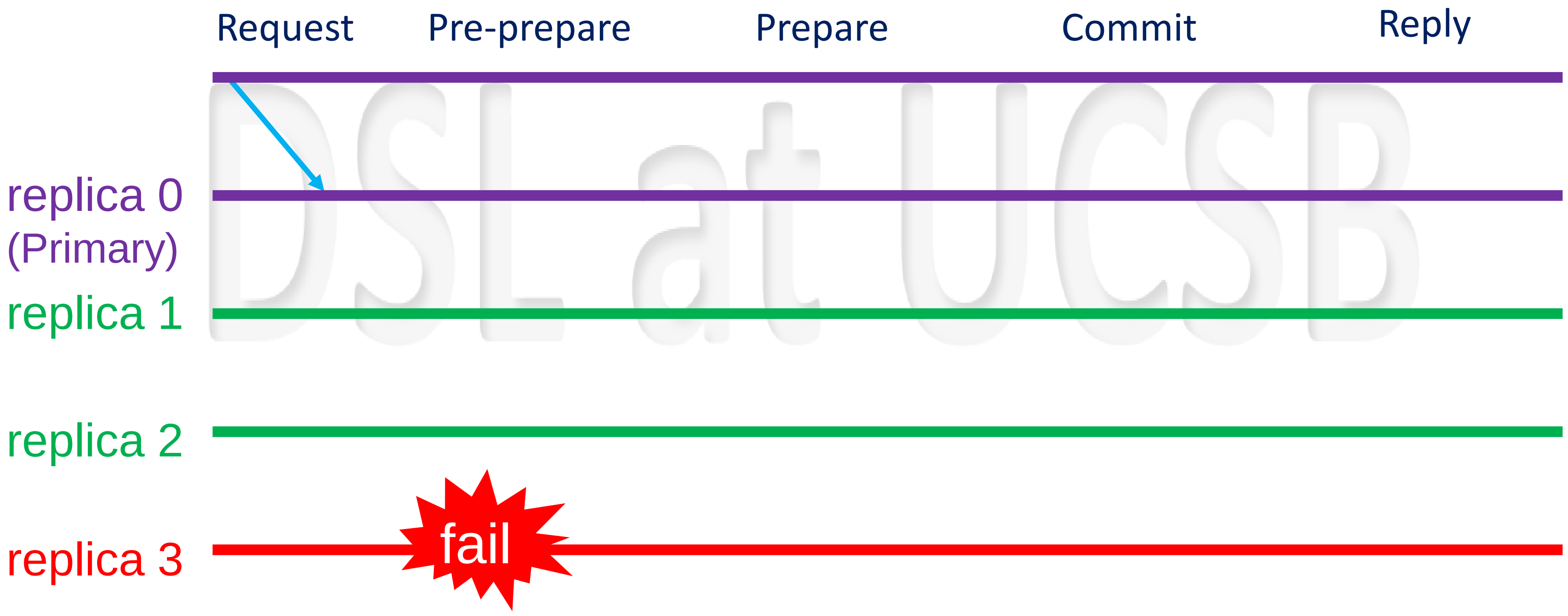


DSL at UCSB

Algorithm

The algorithm has three main phases: (1) *pre-prepare* picks order of requests (2) *prepare* ensures order within views, (3) *commit* ensures order across views

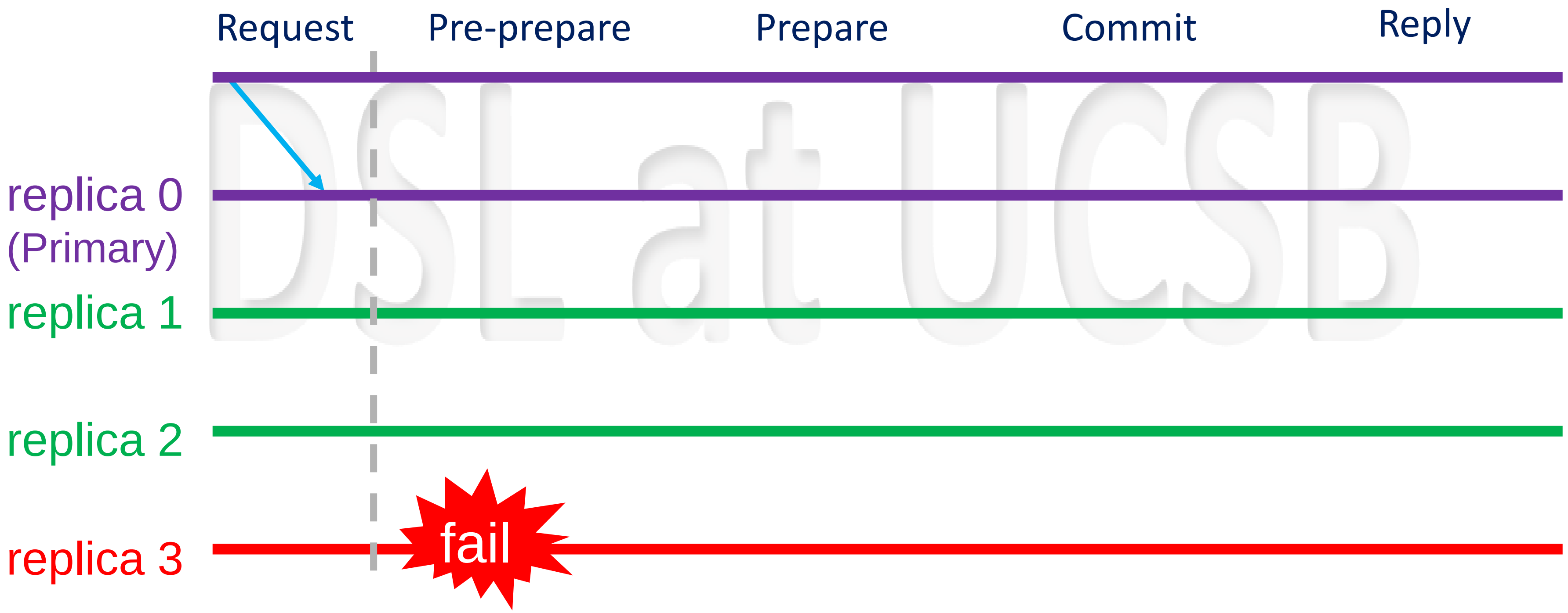
(1) A client sends a request for a service to the primary



DSL at UCSB

Algorithm

The algorithm has three main phases: (1) *pre-prepare* picks order of requests (2) *prepare* ensures order within views, (3) *commit* ensures order across views

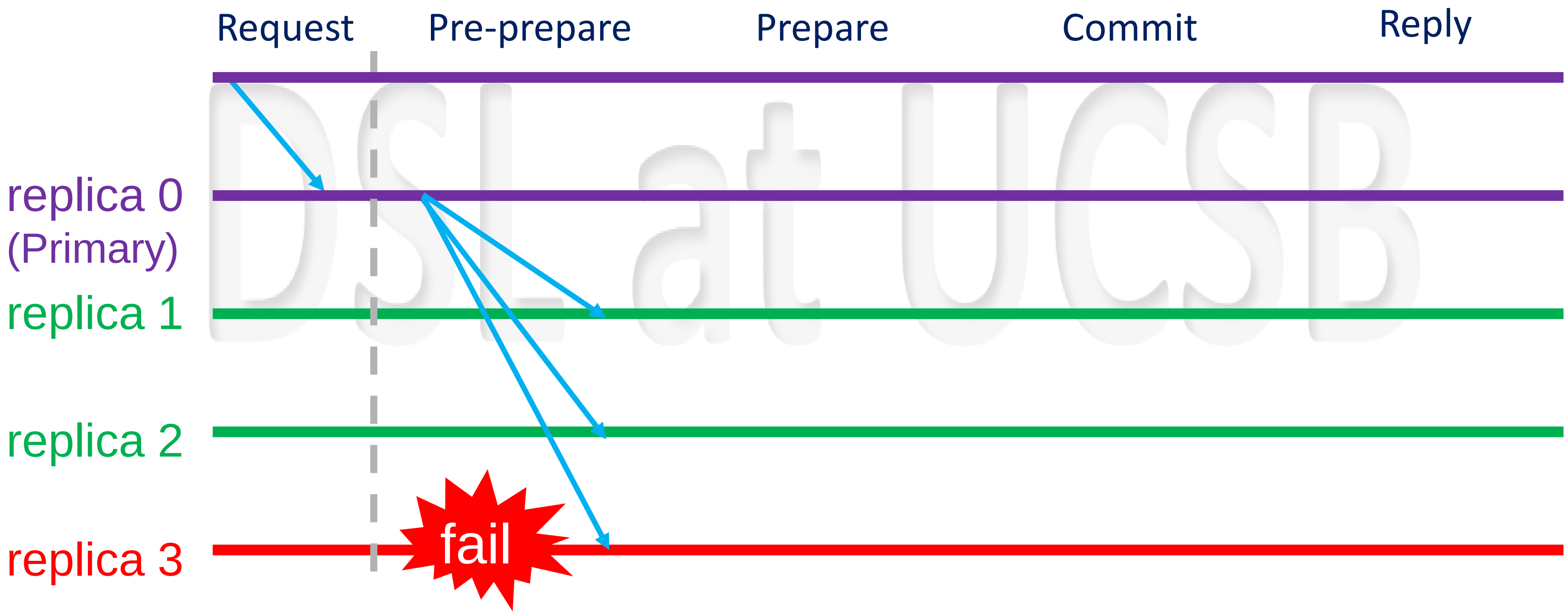


DSL at UCSB

Algorithm

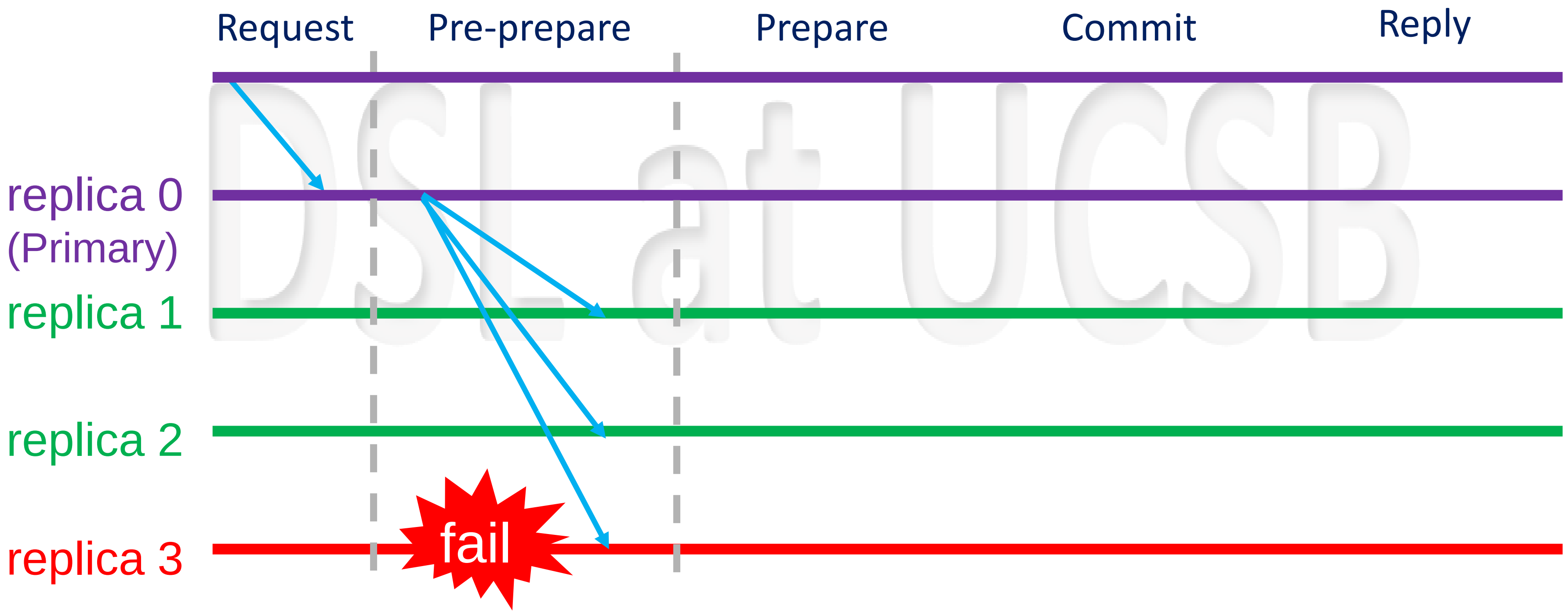
The algorithm has three main phases: (1) *pre-prepare* picks order of requests (2) *prepare* ensures order within views, (3) *commit* ensures order across views

(2) The primary multicasts the request to the backups



Algorithm

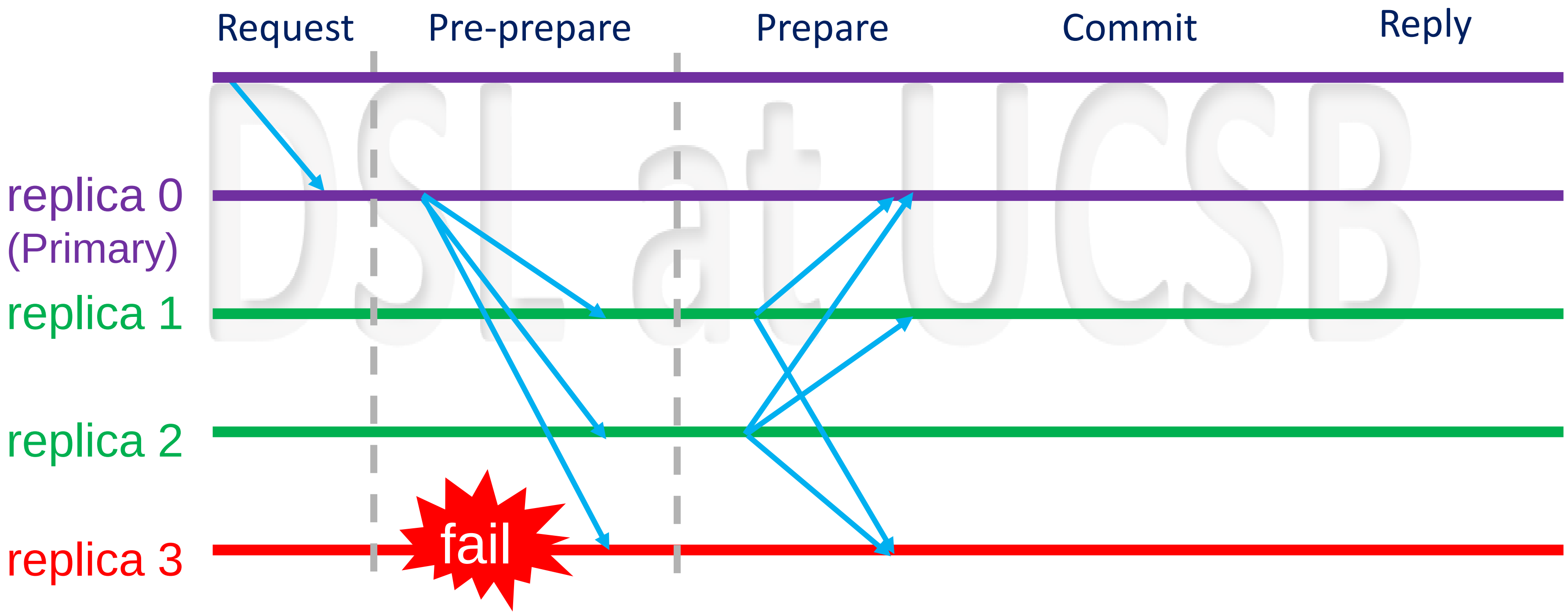
The algorithm has three main phases: (1) *pre-prepare* picks order of requests (2) *prepare* ensures order within views, (3) *commit* ensures order across views



Algorithm

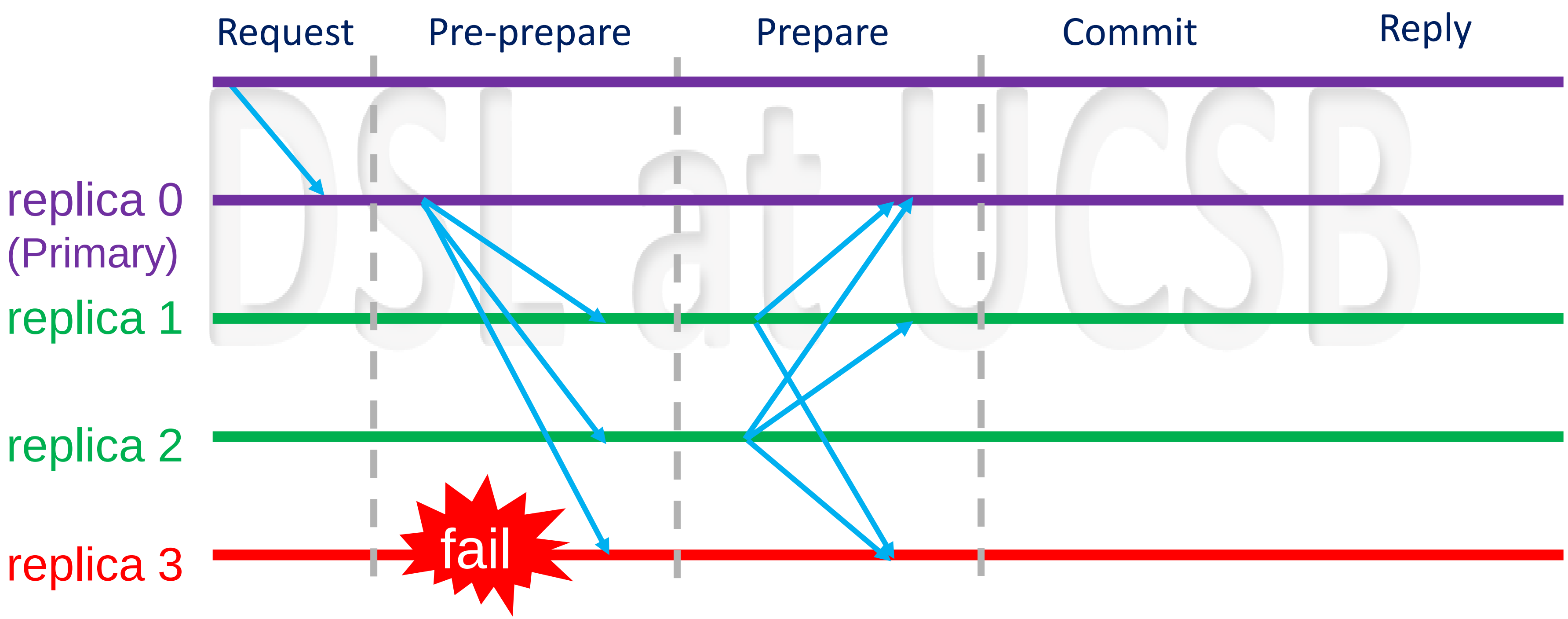
The algorithm has three main phases: (1) *pre-prepare* picks order of requests (2) *prepare* ensures order within views, (3) *commit* ensures order across views

(3) Backups multicast PREPARE message



Algorithm

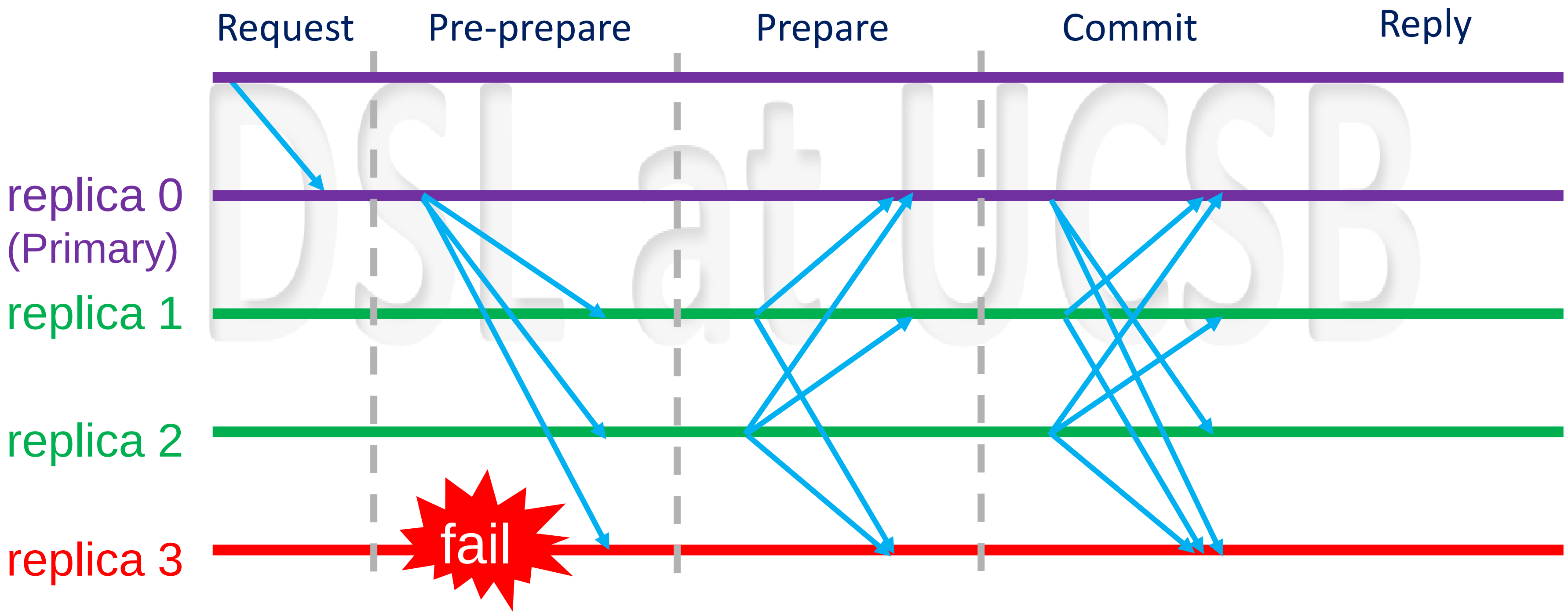
The algorithm has three main phases: (1) *pre-prepare* picks order of requests (2) *prepare* ensures order within views, (3) *commit* ensures order across views



Algorithm

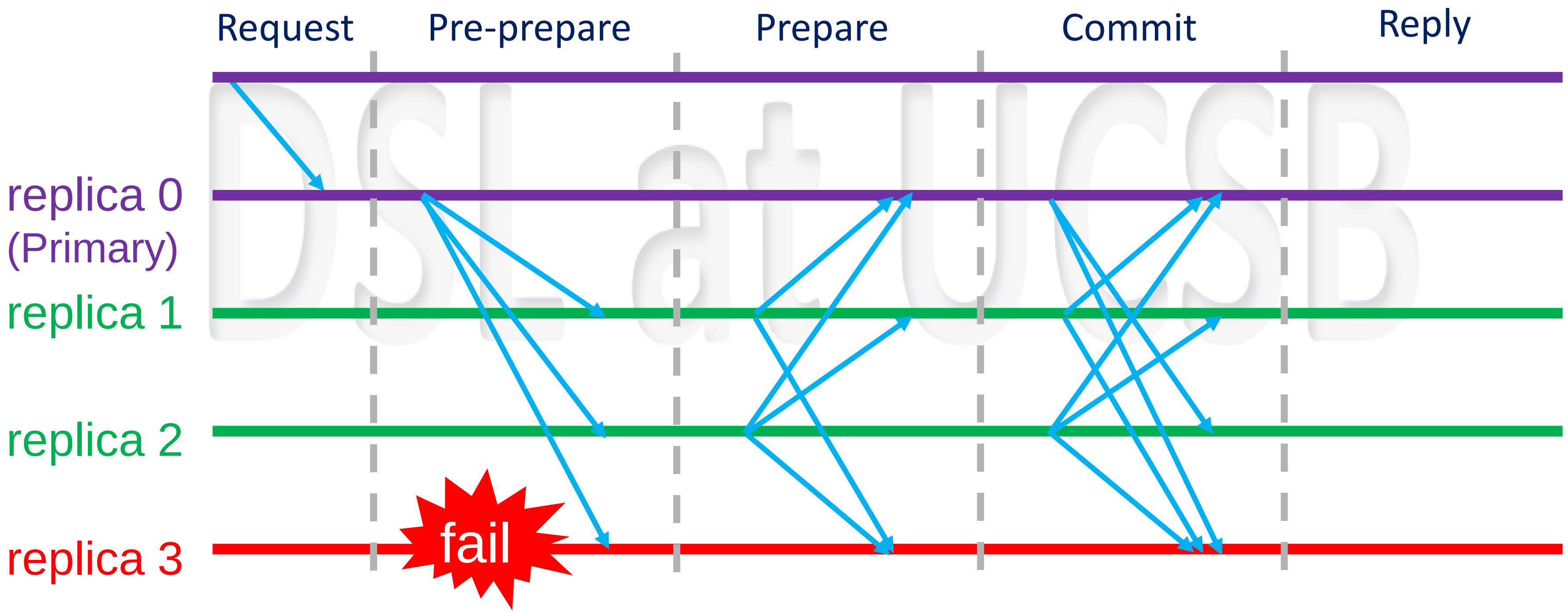
The algorithm has three main phases: (1) *pre-prepare* picks order of requests (2) *prepare* ensures order within views, (3) *commit* ensures order across views

(4) If a replica receives at least 2f matching PREPARE message, multicasts a COMMIT message



Algorithm

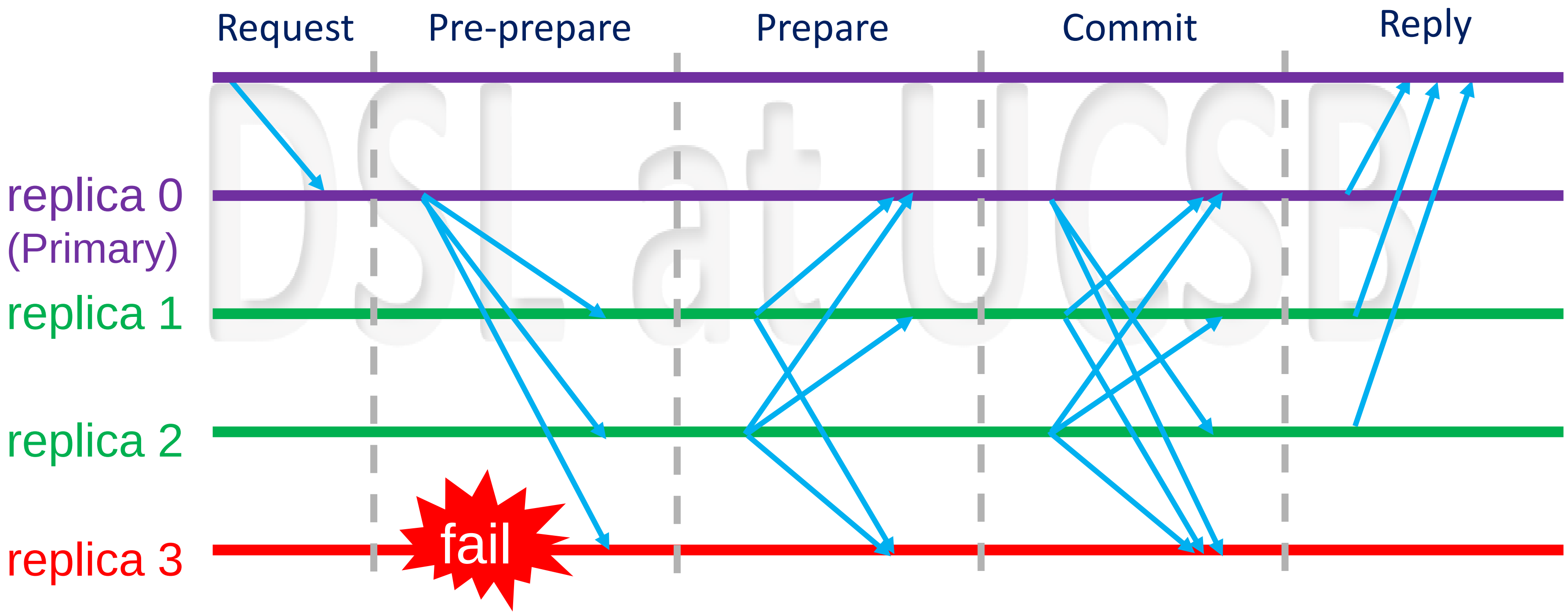
The algorithm has three main phases: (1) *pre-prepare* picks order of requests (2) *prepare* ensures order within views, (3) *commit* ensures order across views



Algorithm

The algorithm has three main phases: (1) *pre-prepare* picks order of requests (2) *prepare* ensures order within views, (3) *commit* ensures order across views

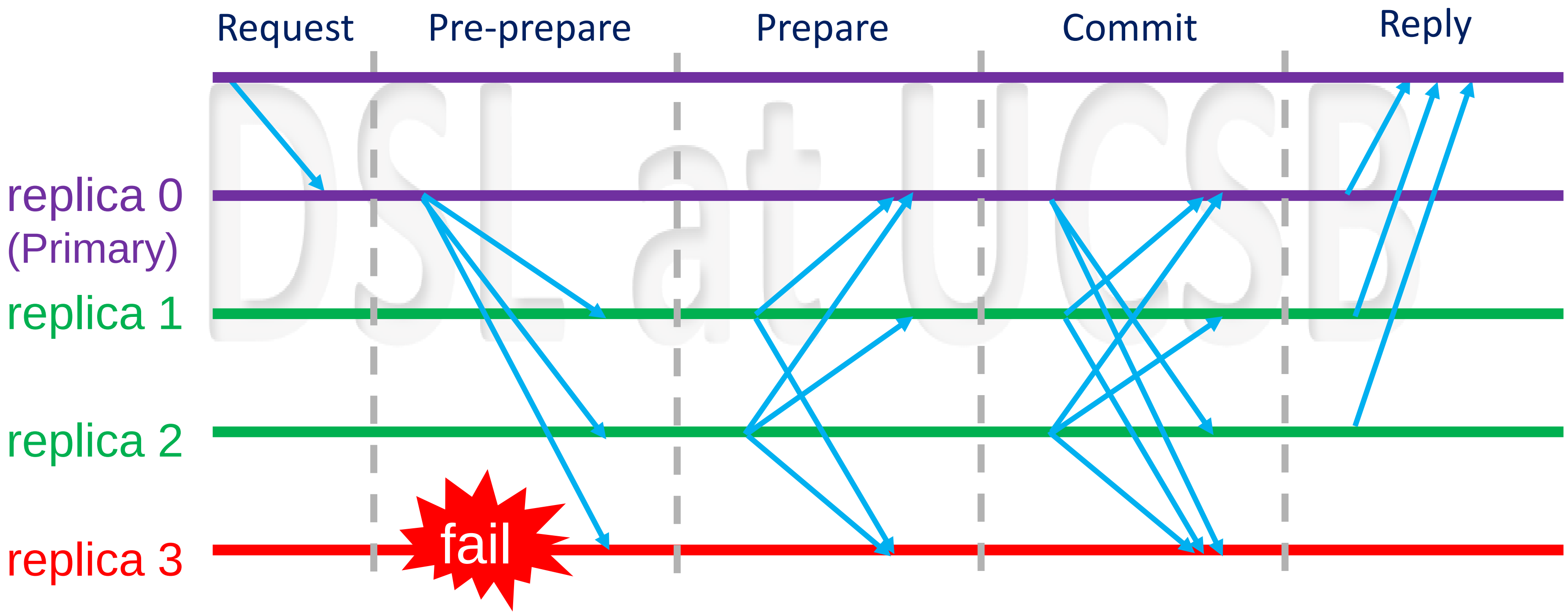
(5) If a replica receives at least 2f COMMIT messages, reply the result to the client



Algorithm

The algorithm has three main phases: (1) *pre-prepare* picks order of requests (2) *prepare* ensures order within views, (3) *commit* ensures order across views

(6) The client waits for **f+1** replies from different replicas with the same result

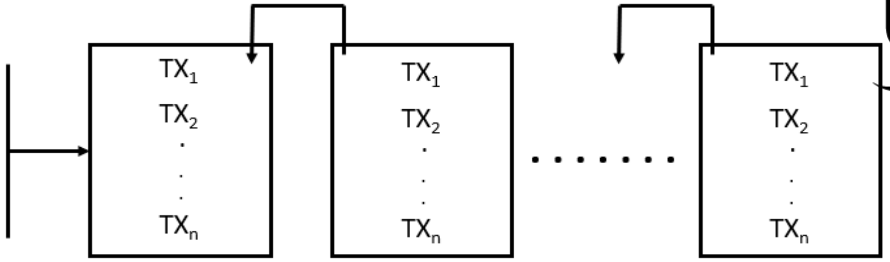


PBFT Consensus

- Tolerates Byzantine (Malicious) failures
  - To make progress, at least **2/3** of the participants should be **correct**
  - Progress is not guaranteed (FLP impossibility)
- However, PBFT is **Permissioned**
  - All participants should be known **a priori**
- Also, PBFT has high network overhead $O(N^2)$ [number of messages]
  - Every node multi-casts their responses to every other node

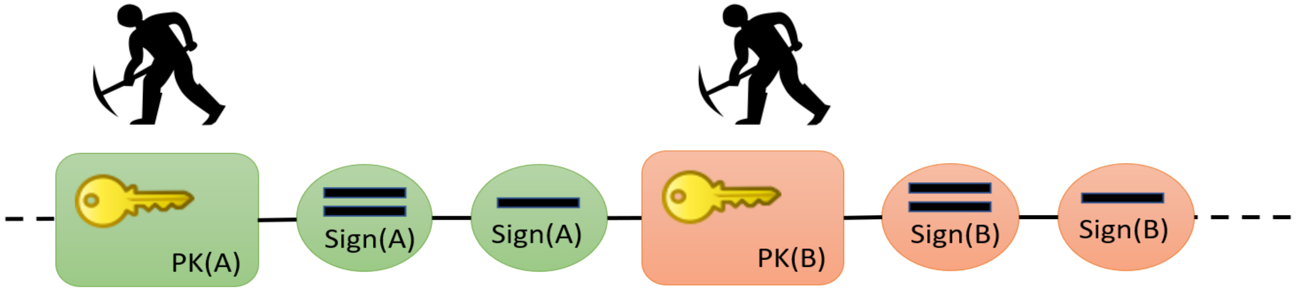
DSL at UCSB

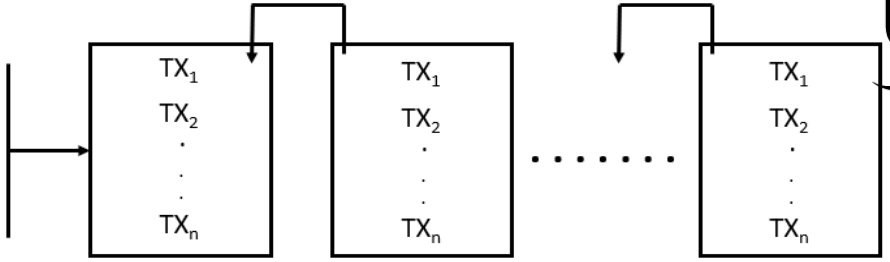
DSL at UCSB



DSL at UCSB

DSL

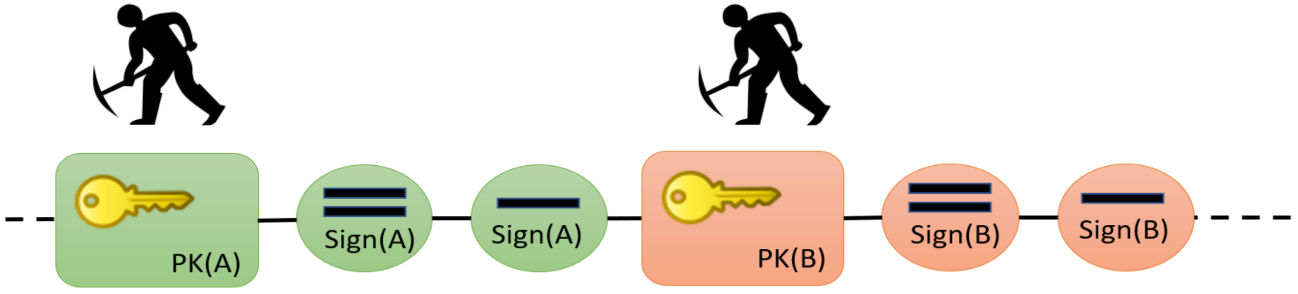


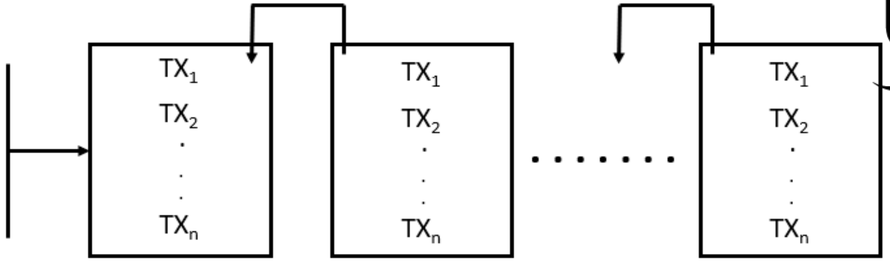


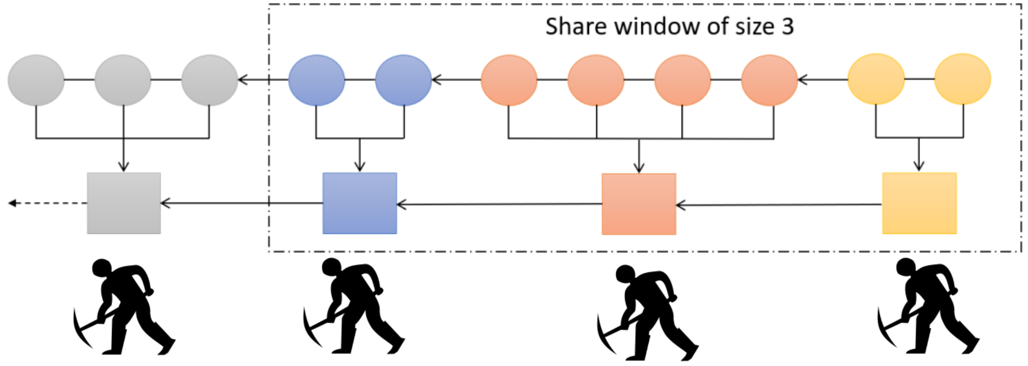
UCSB

DSL at UCSB

DSL



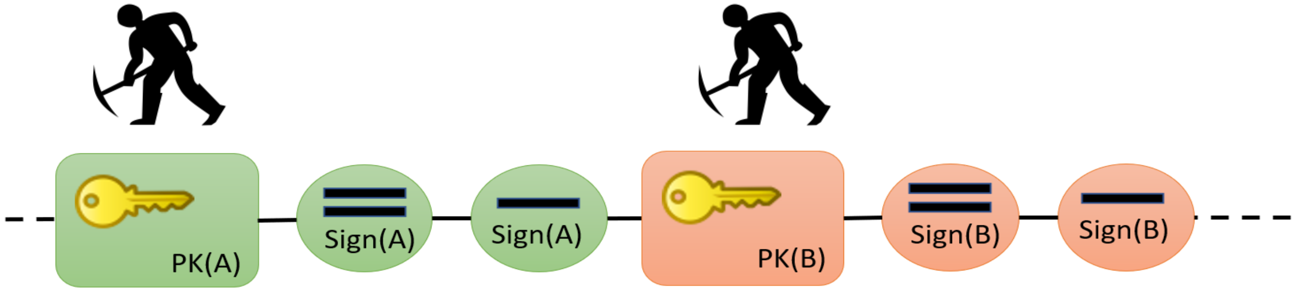


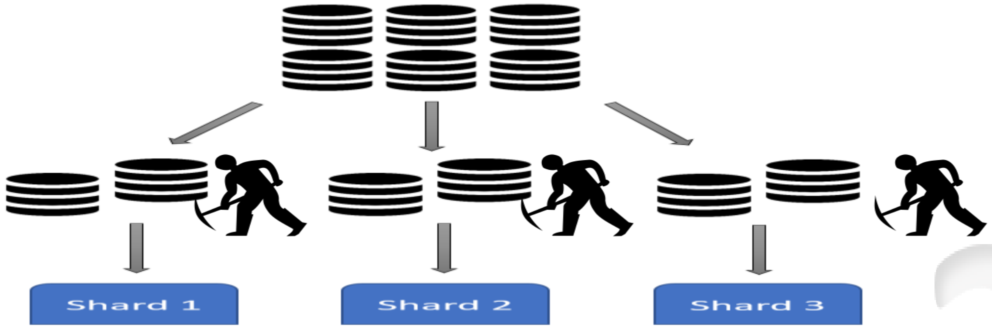


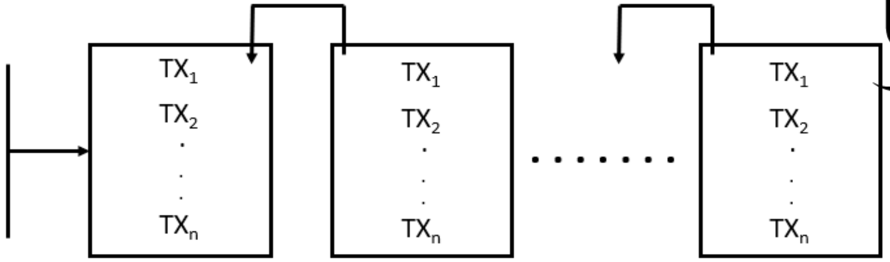
DSL at UCSB

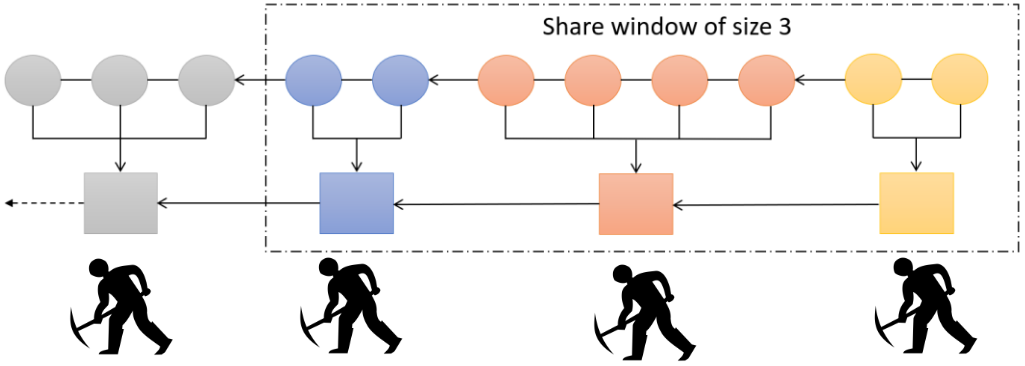
DSL

UCSB





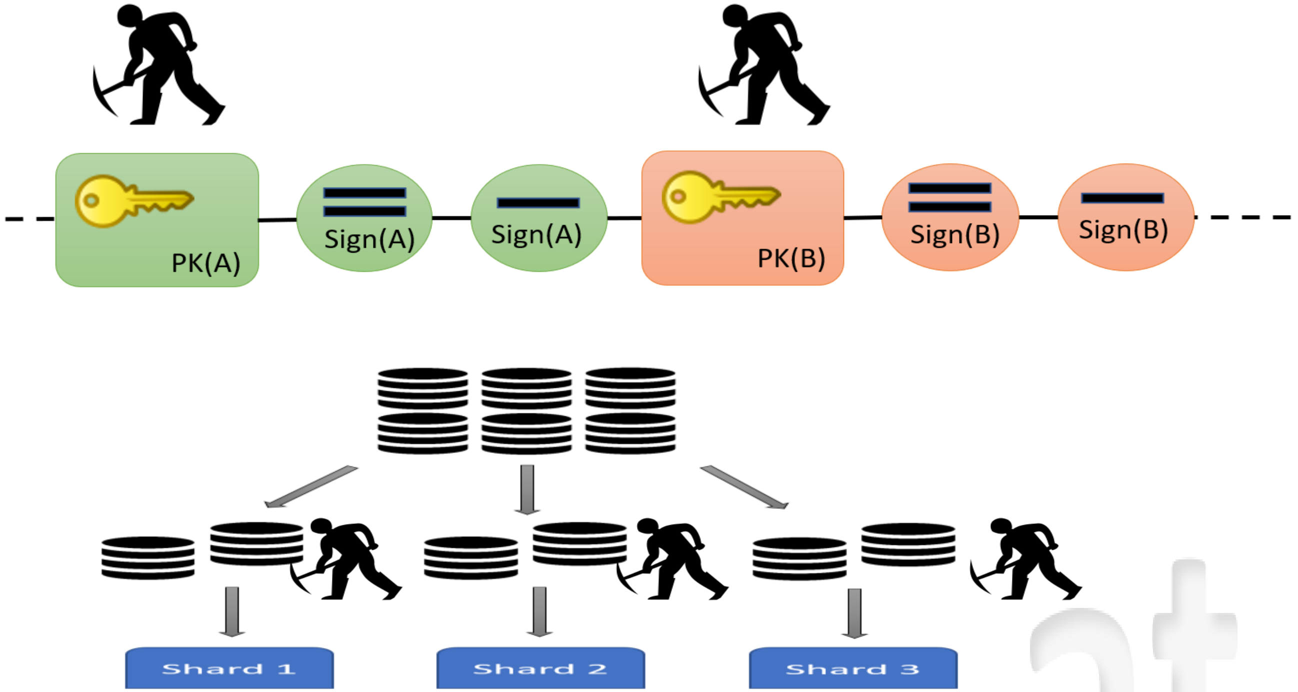


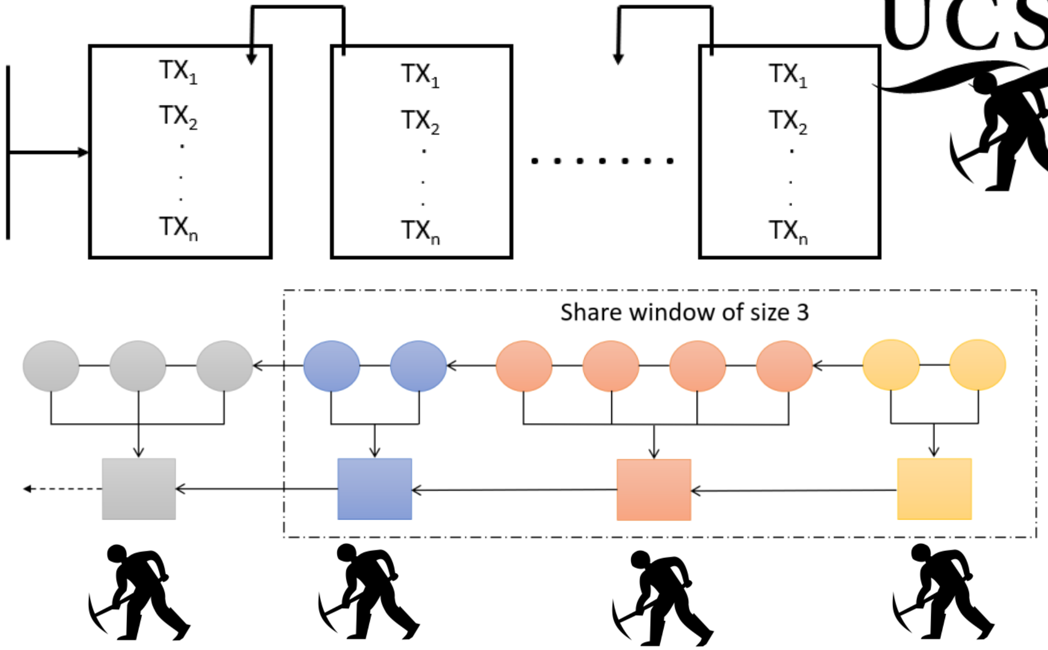


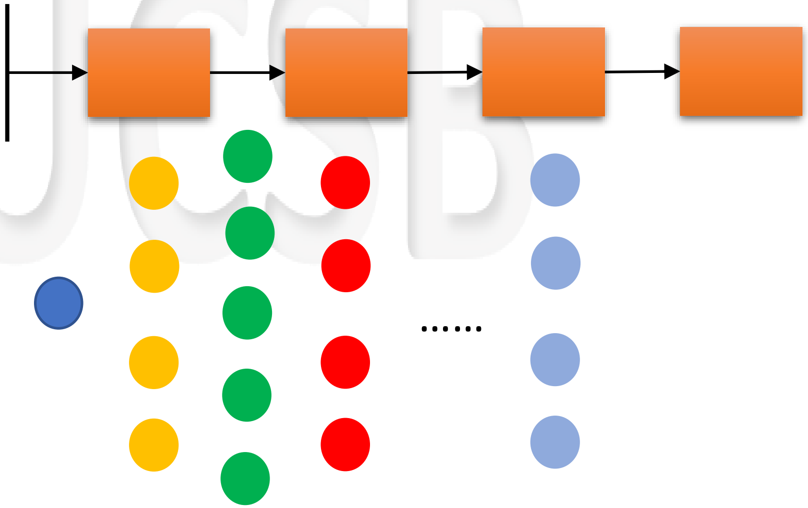
DSL at UCSB

DSL

UCSB

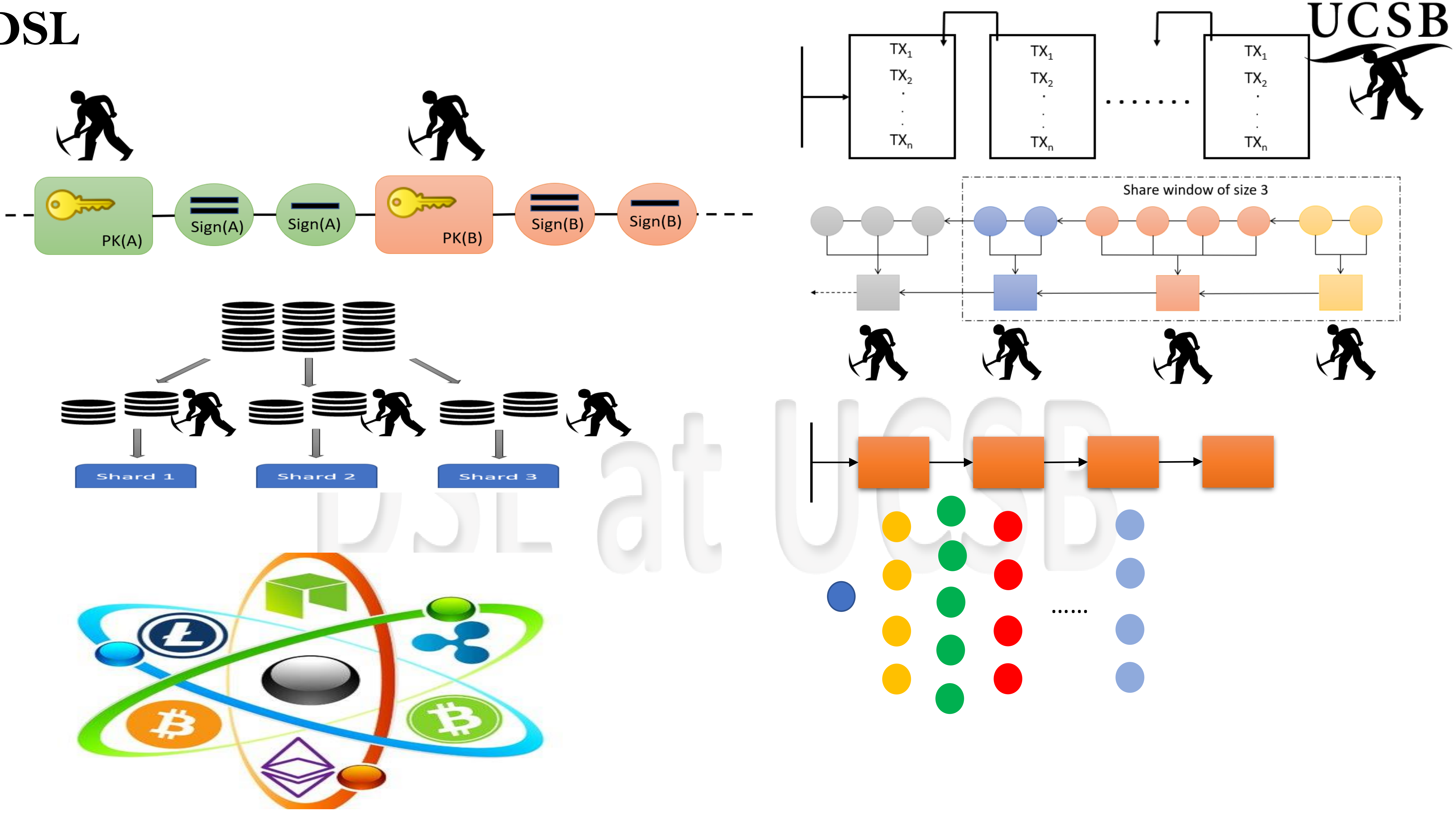




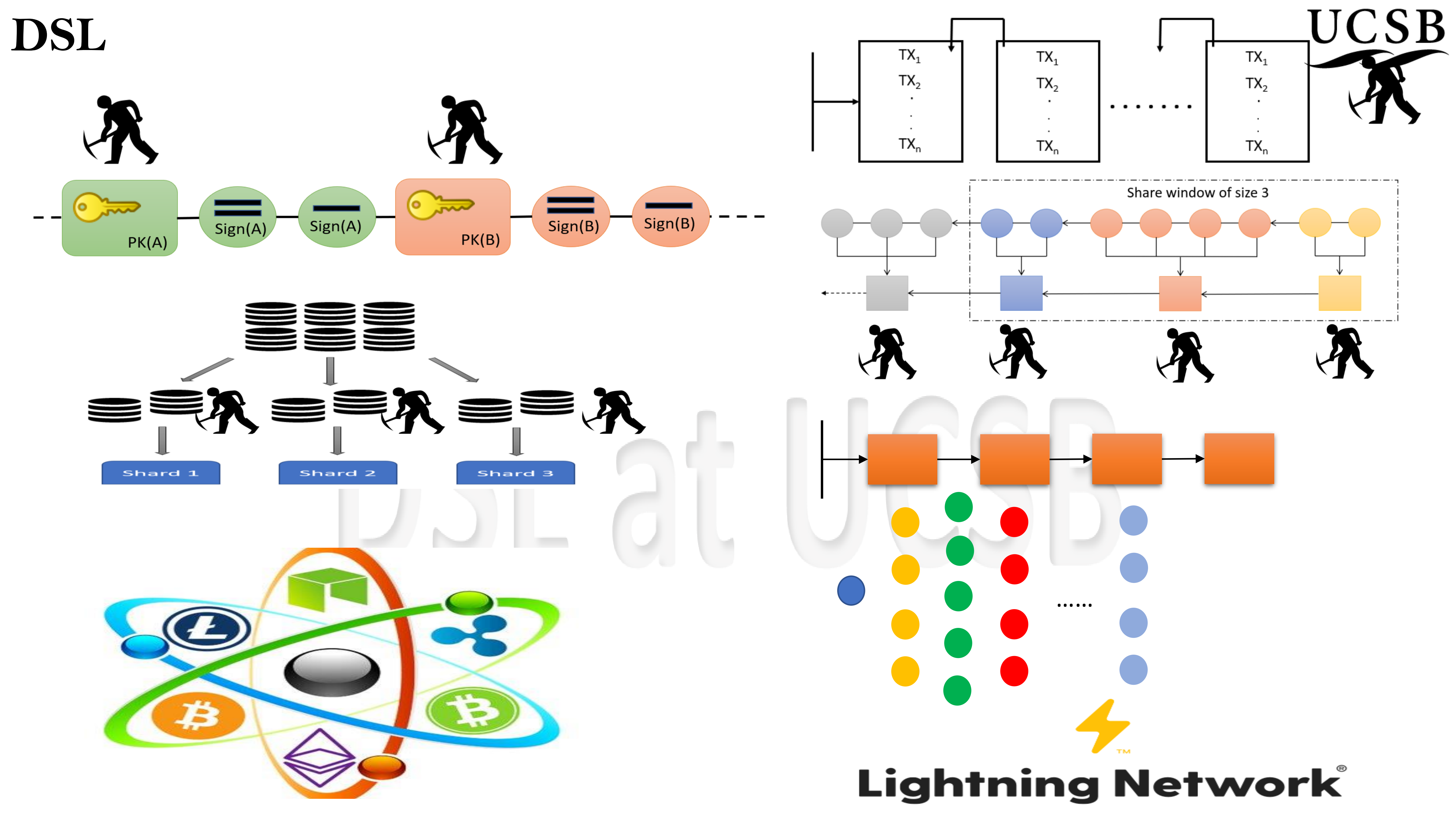


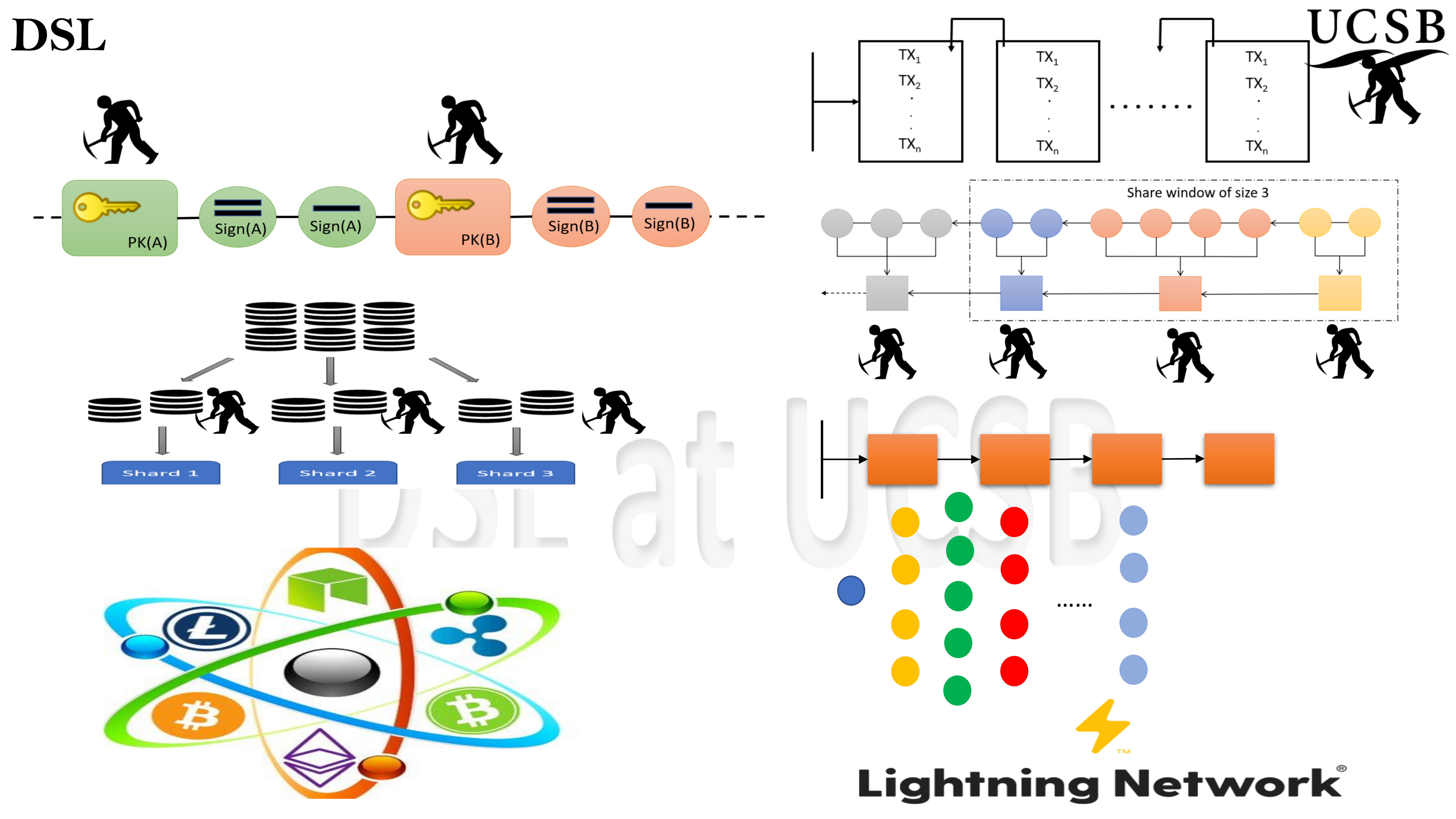
DSL at UCSB

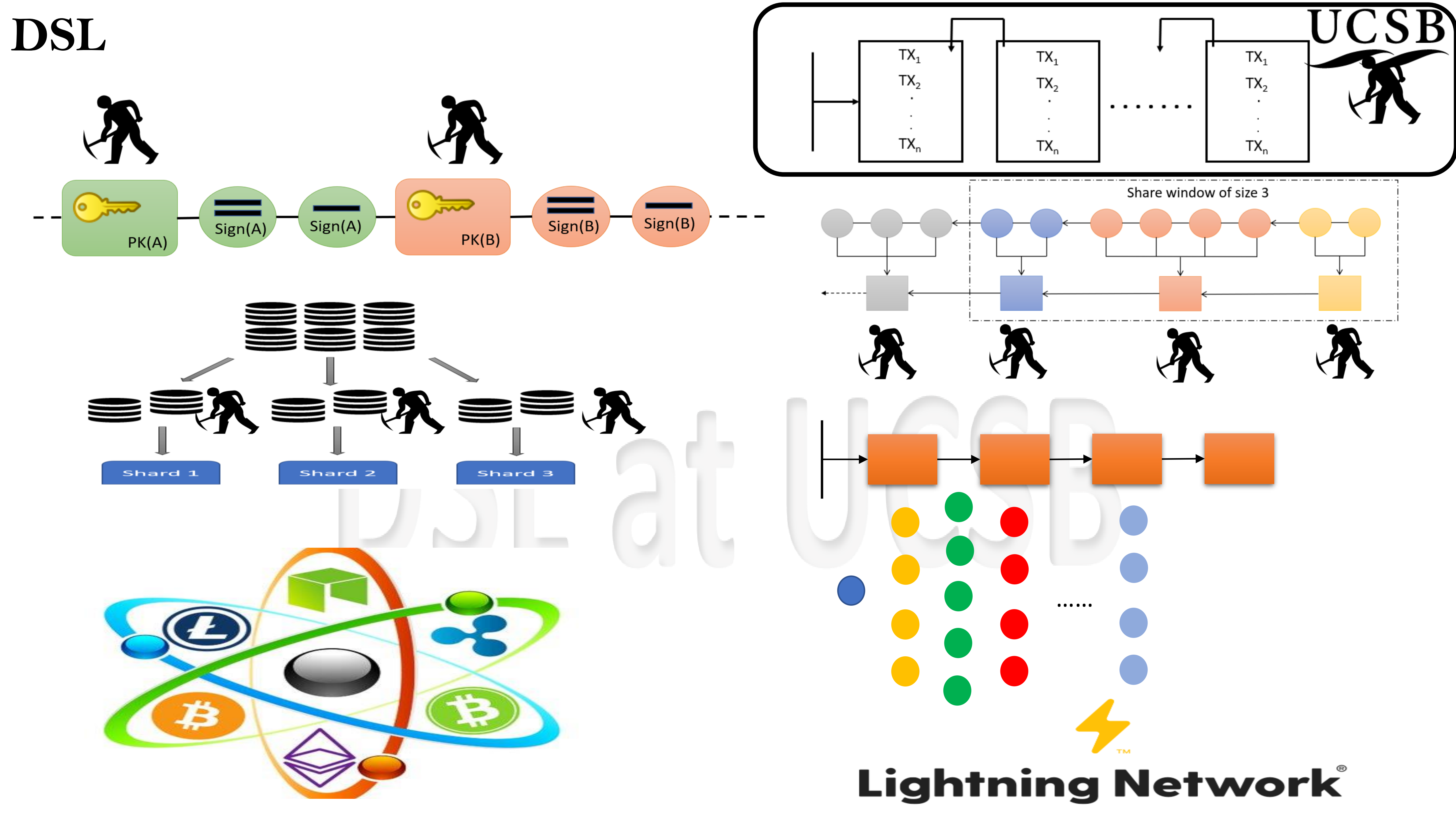
DSL



DSL at UCSB


DSL
UCSB
TX1
TX2
TXn
PK(A)
Sign(A)
Sign(A)
PK(B)
Sign(B)
Sign(B)
Share window of size 3
Shard 1
Shard 2
Shard 3
DSL at UCSB
Lightning Network


DSL
UCSB
PK(A)
Sign(A)
Sign(A)
PK(B)
Sign(B)
Sign(B)
TX1
TX2
TXn
Share window of size 3
Shard 1
Shard 2
Shard 3
DSL at UCSB
Lightning Network


DSL
UCSB
TX1
TX2
TXn
TX1
TX2
TXn
TX1
TX2
TXn
Share window of size 3
PK(A)
Sign(A)
Sign(A)
PK(B)
Sign(B)
Sign(B)
Shard 1
Shard 2
Shard 3
DSL at UCSB
Lightning Network

Nakamoto’s Consensus

- Intuitively, network nodes race to solve a puzzle
- This puzzle is computationally expensive
- Once a network node finds (mines) a solution:
  - It adds its block of transactions to the blockchain
  - It multi-casts the solution to other network nodes
  - Other network nodes accept and verify the solution

DSL at UCSB

Mining Details

DSL at UCSB

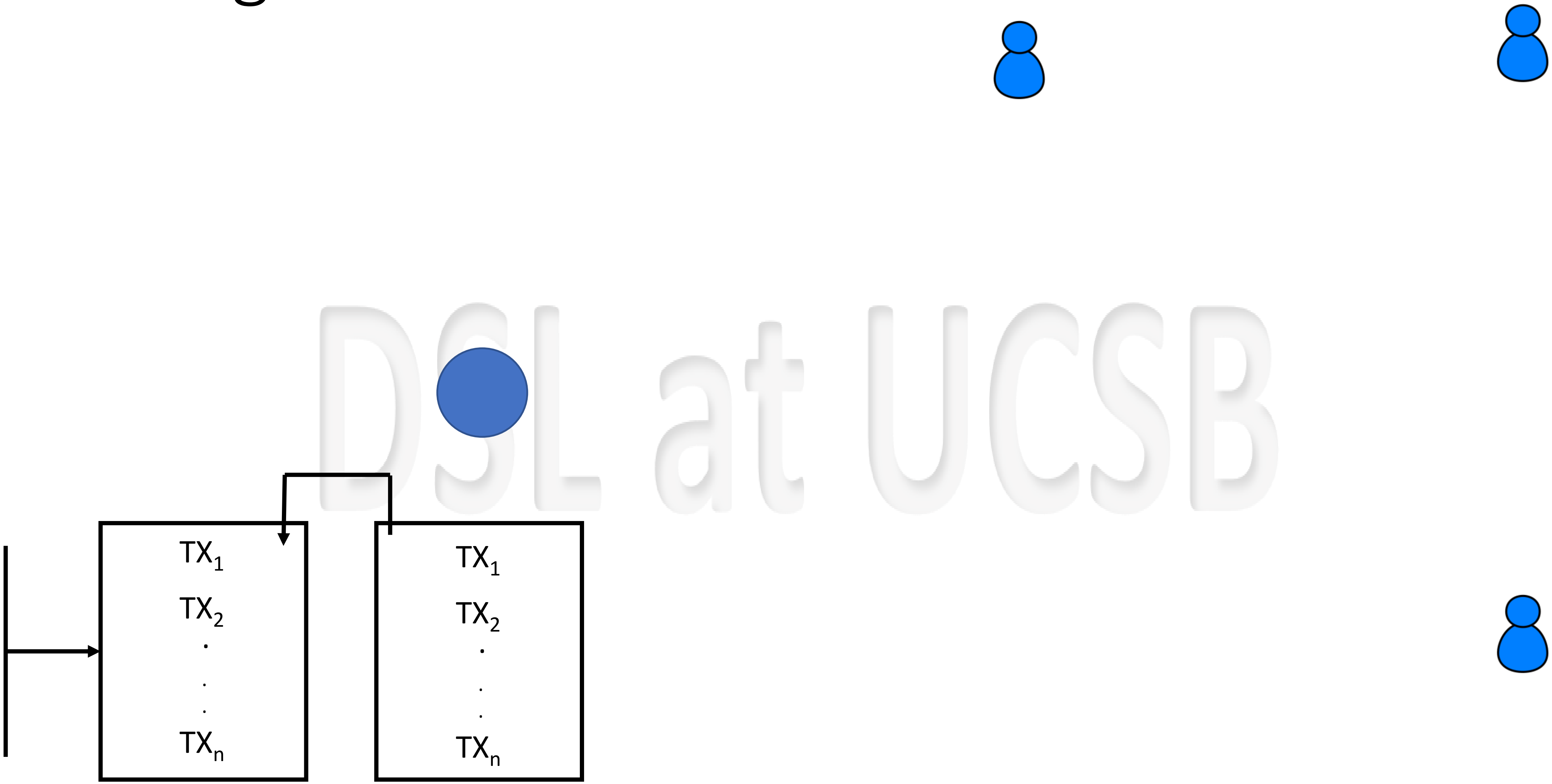
Mining Details

DSL at UCSB

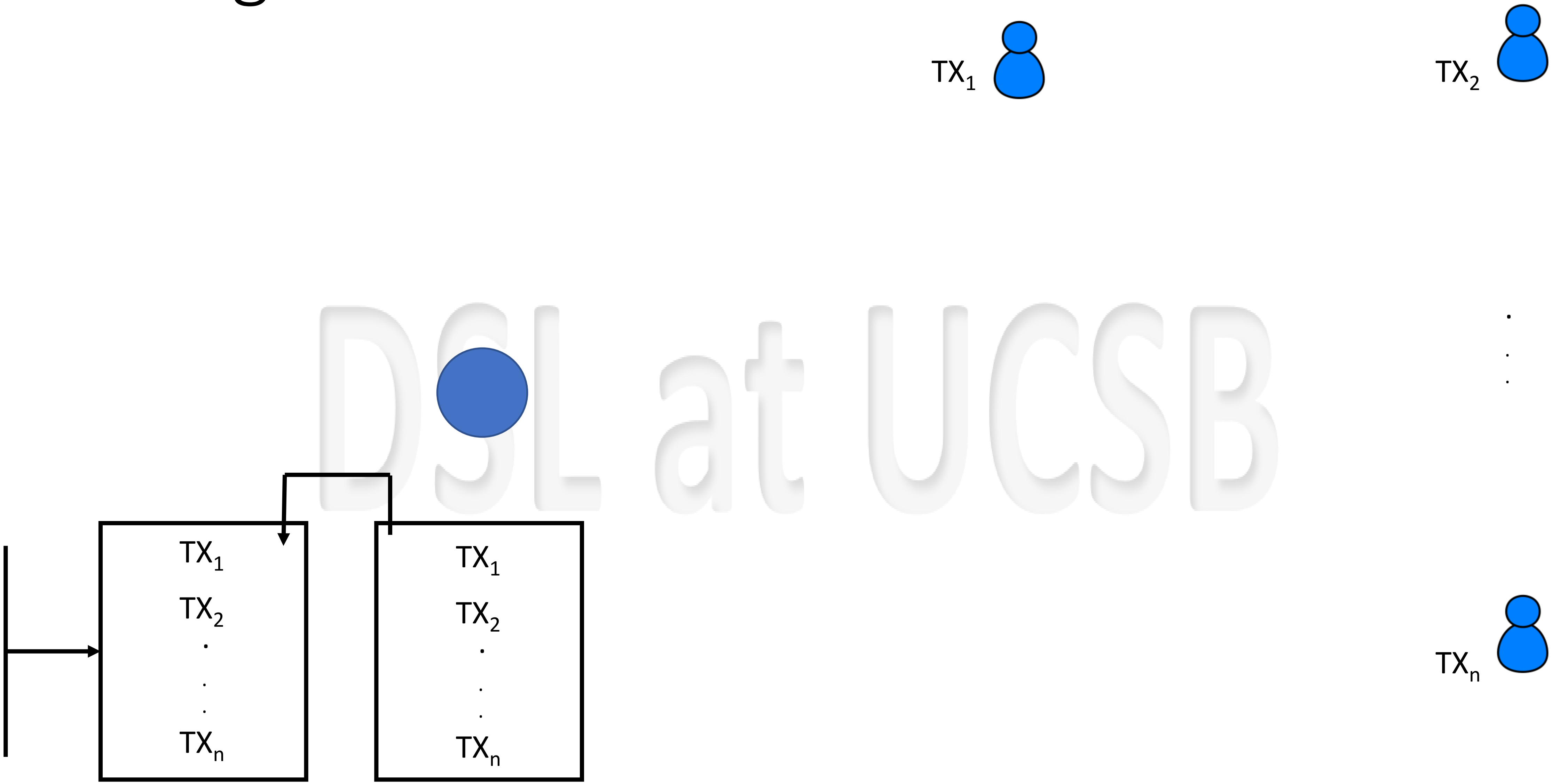
DSL

Mining Details

DSL at UCSB

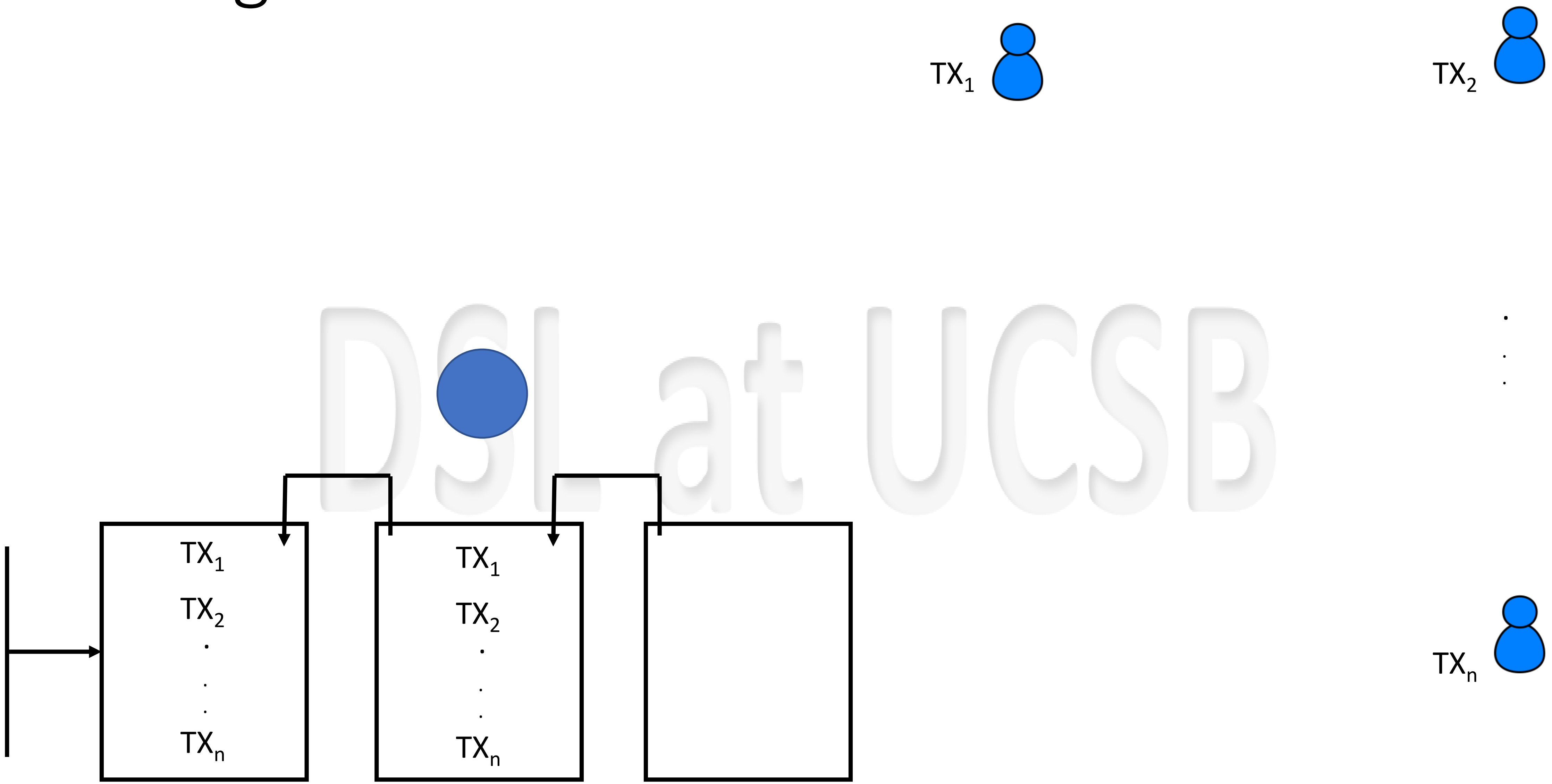


Mining Details



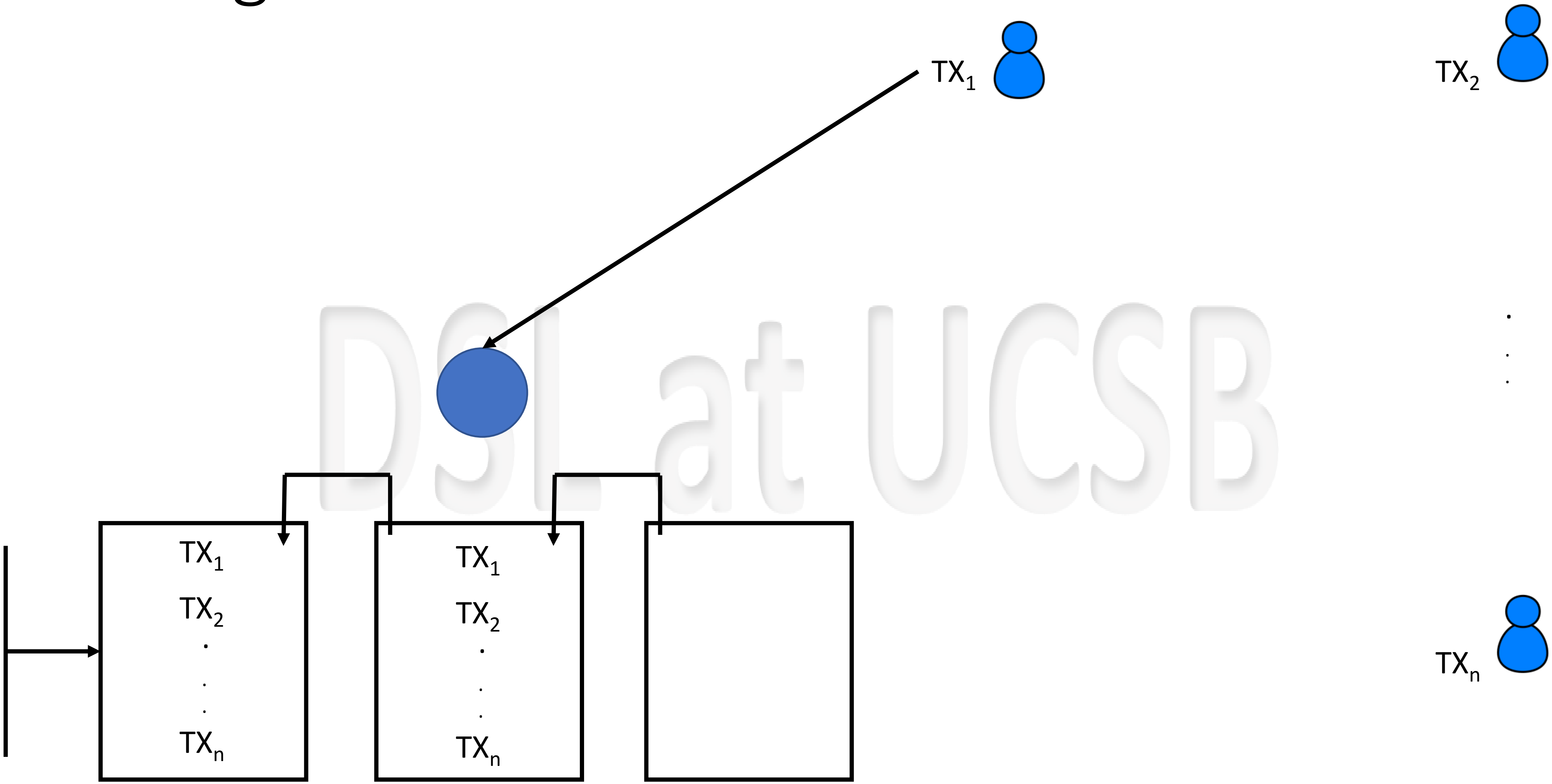
DSL at UCSB

Mining Details



DSL at UCSB

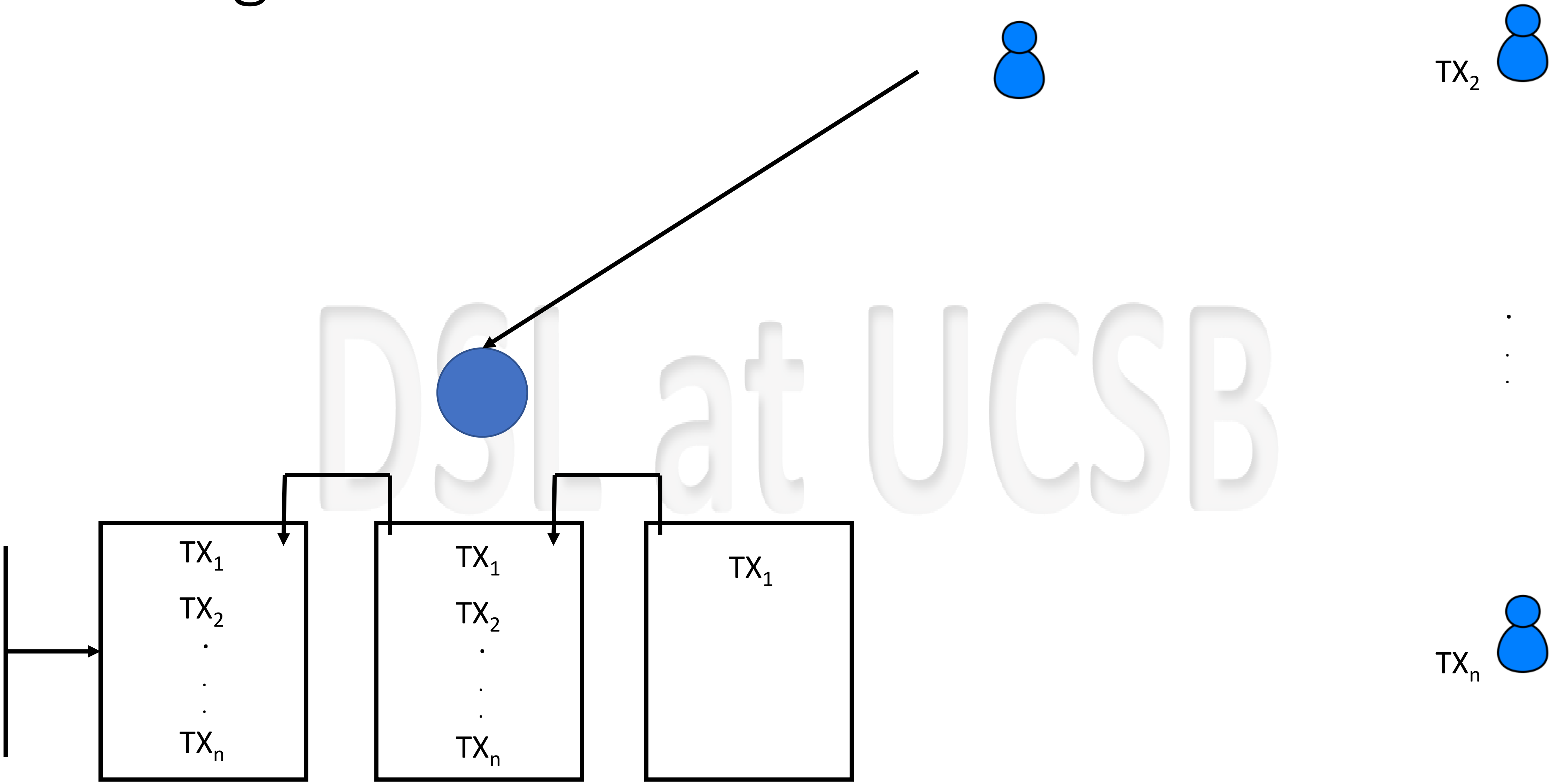
Mining Details



DSL at UCSB

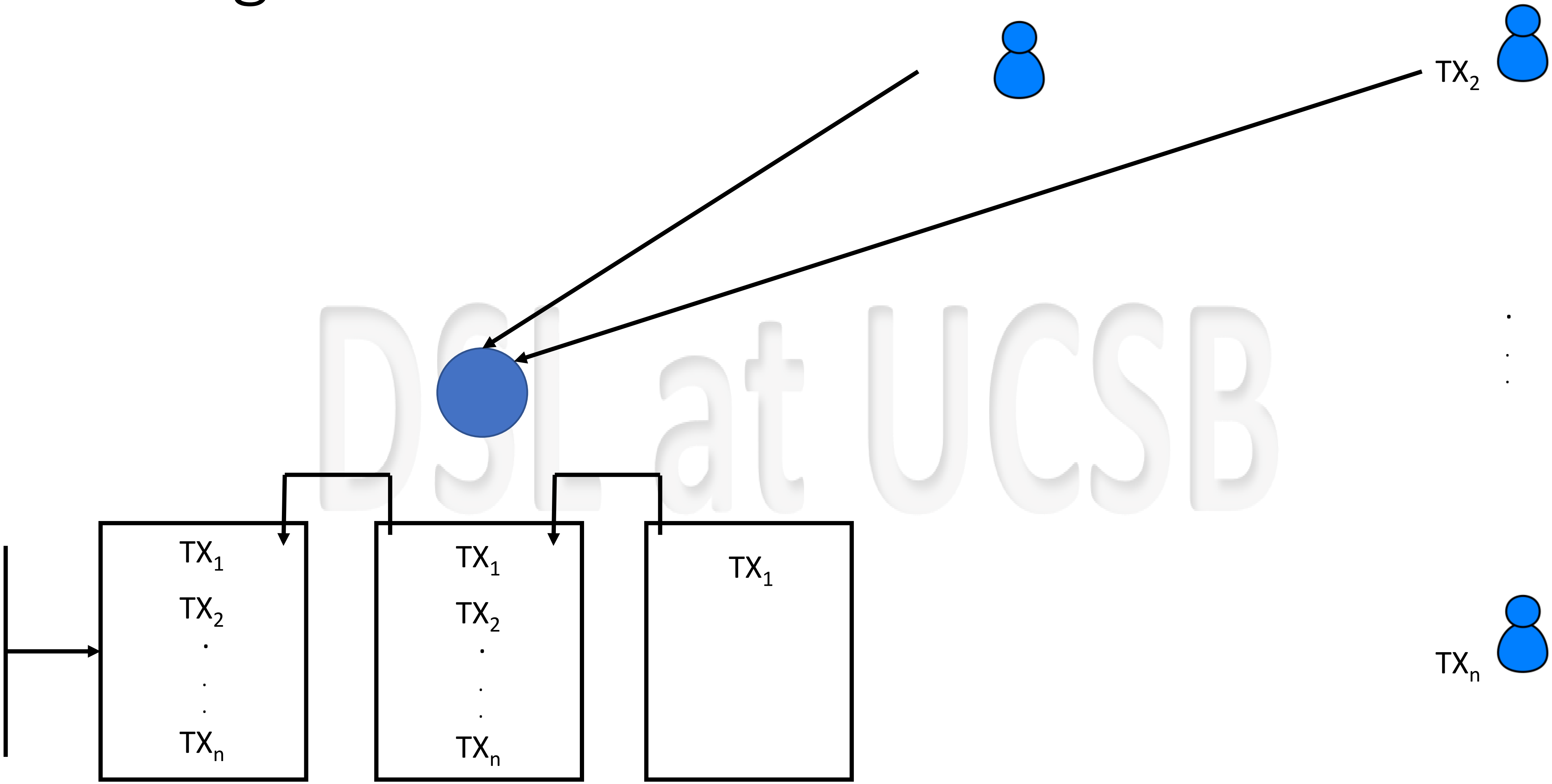
DSL UCSB

Mining Details



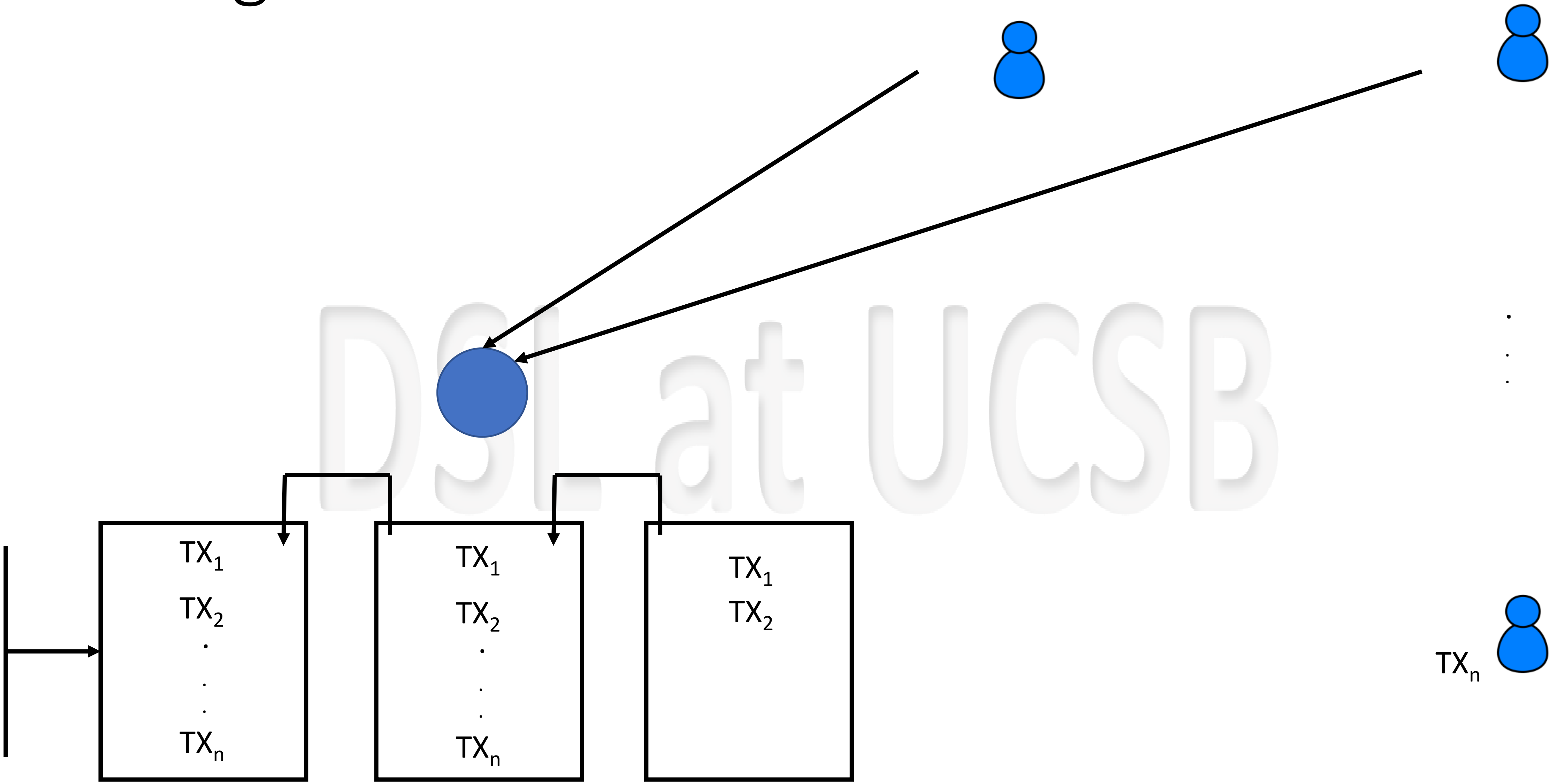
DSL at UCSB

Mining Details



DSL at UCSB

Mining Details

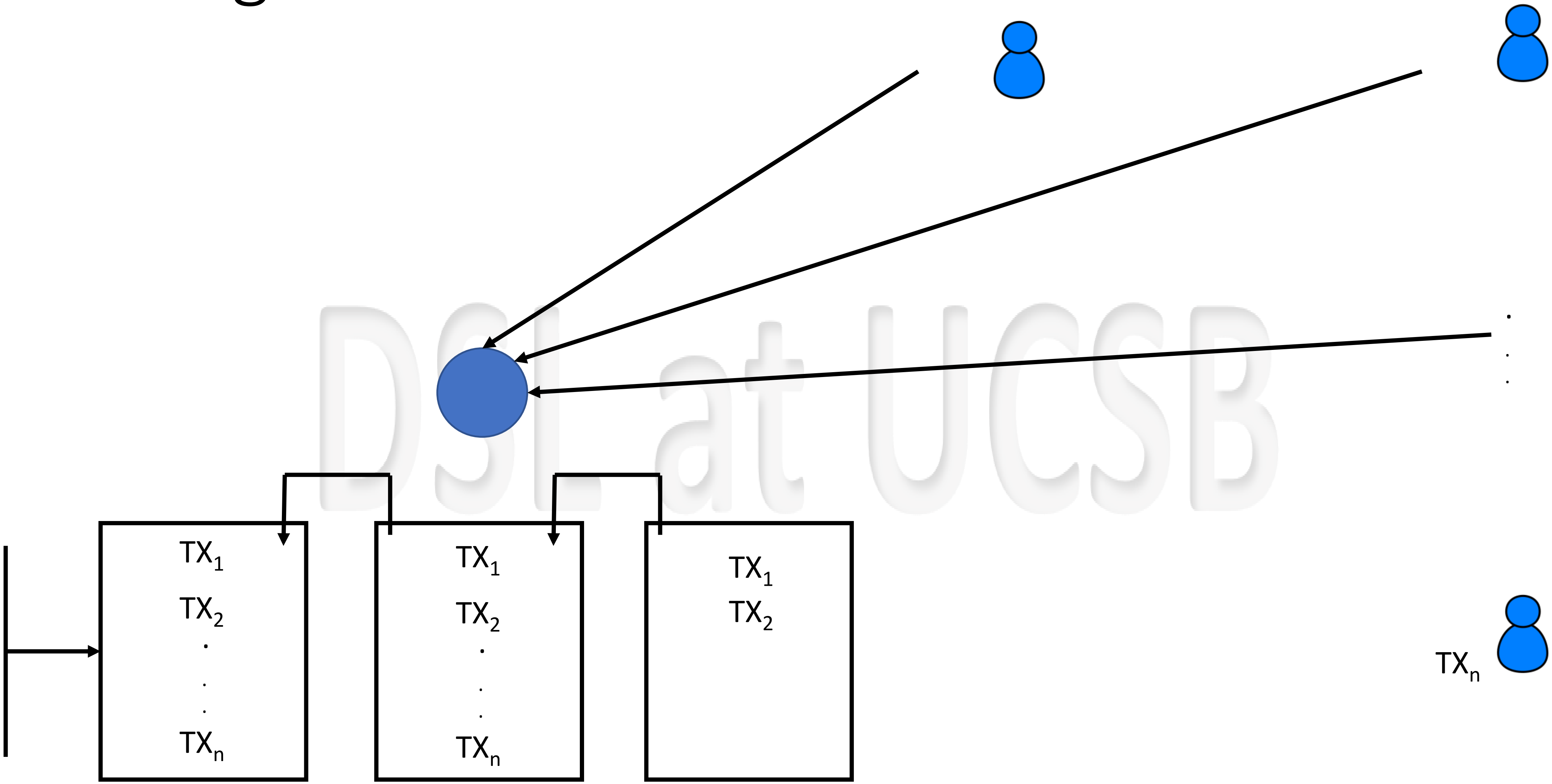


DSL at UCSB

DSL

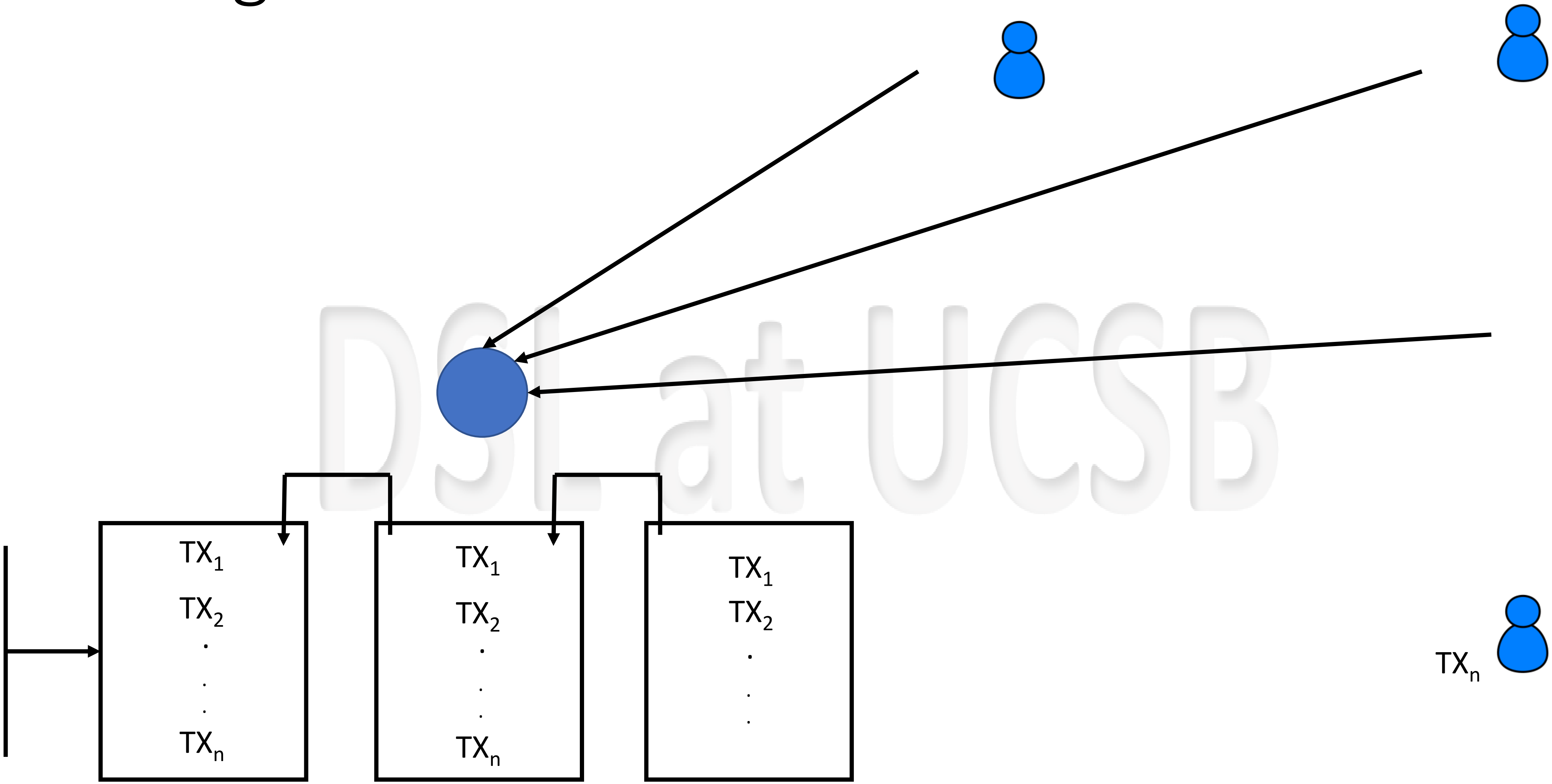
UCSB

Mining Details



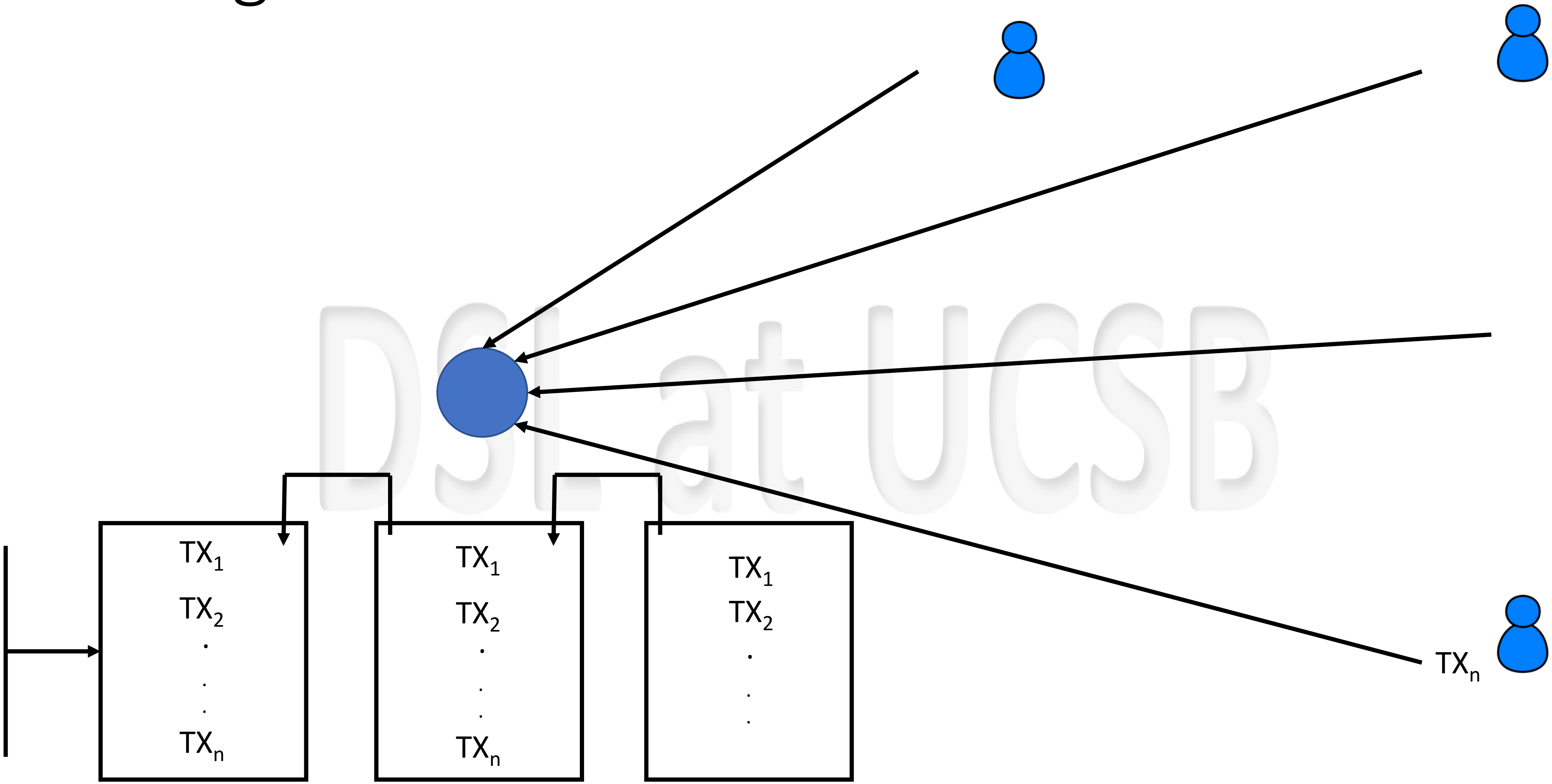
DSL at UCSB

Mining Details



DSL at UCSB

Mining Details

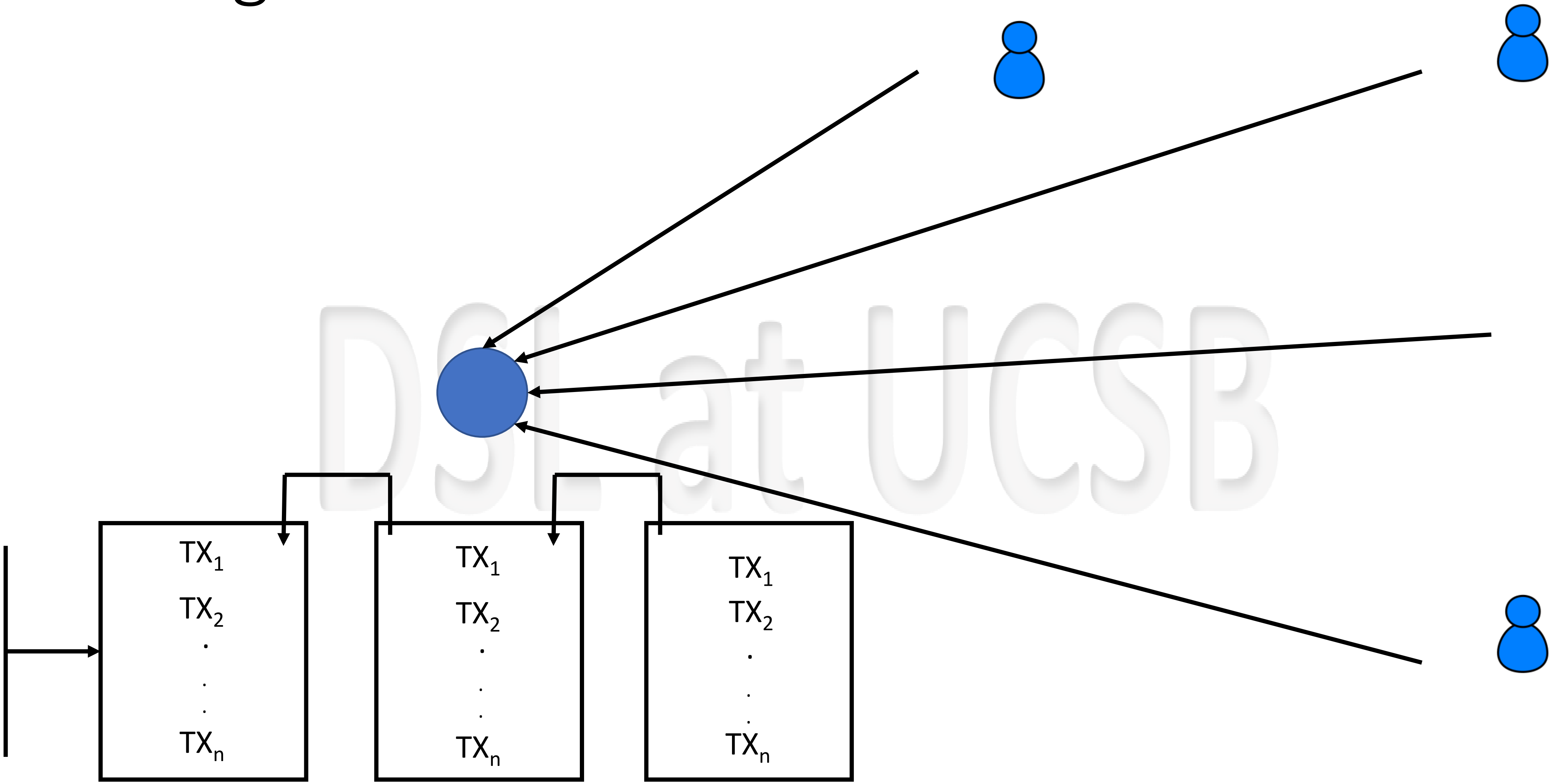


DSL at UCSB

DSL

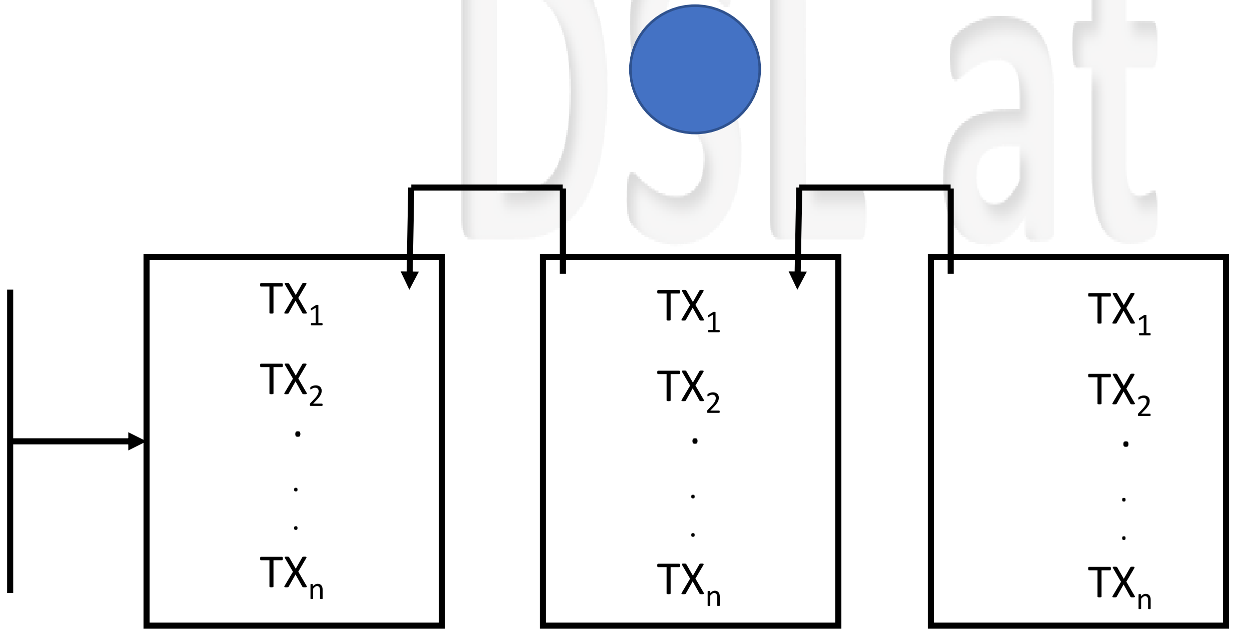
UCSB

Mining Details



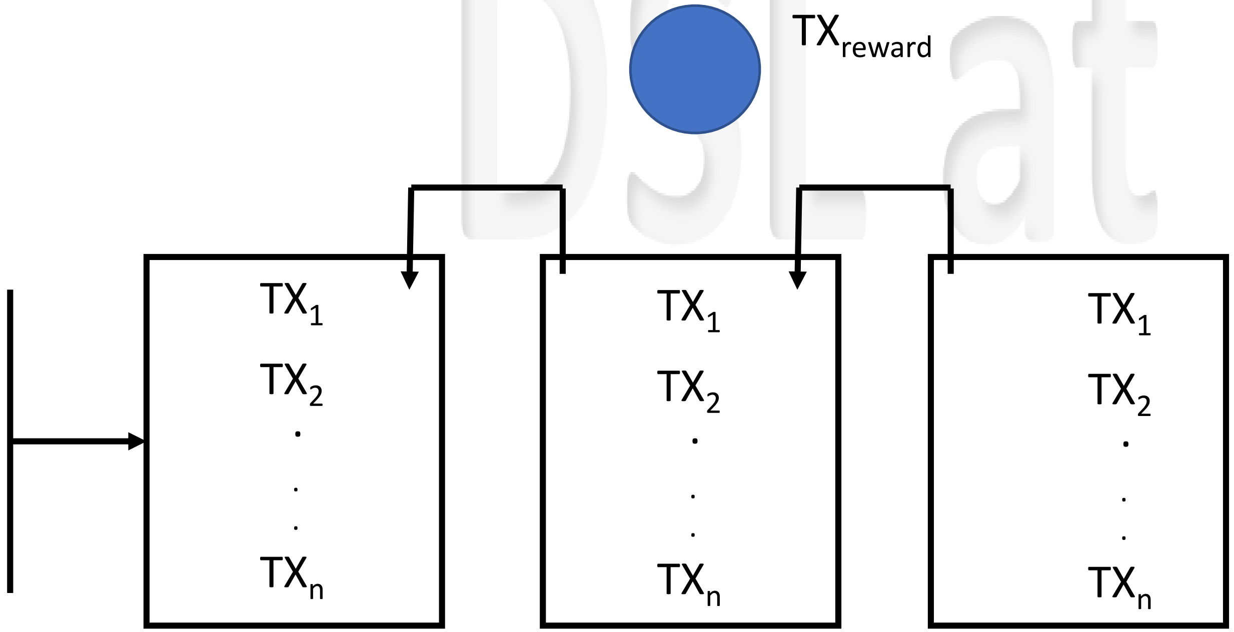
DSL at UCSB

Mining Details



DSL at UCSB

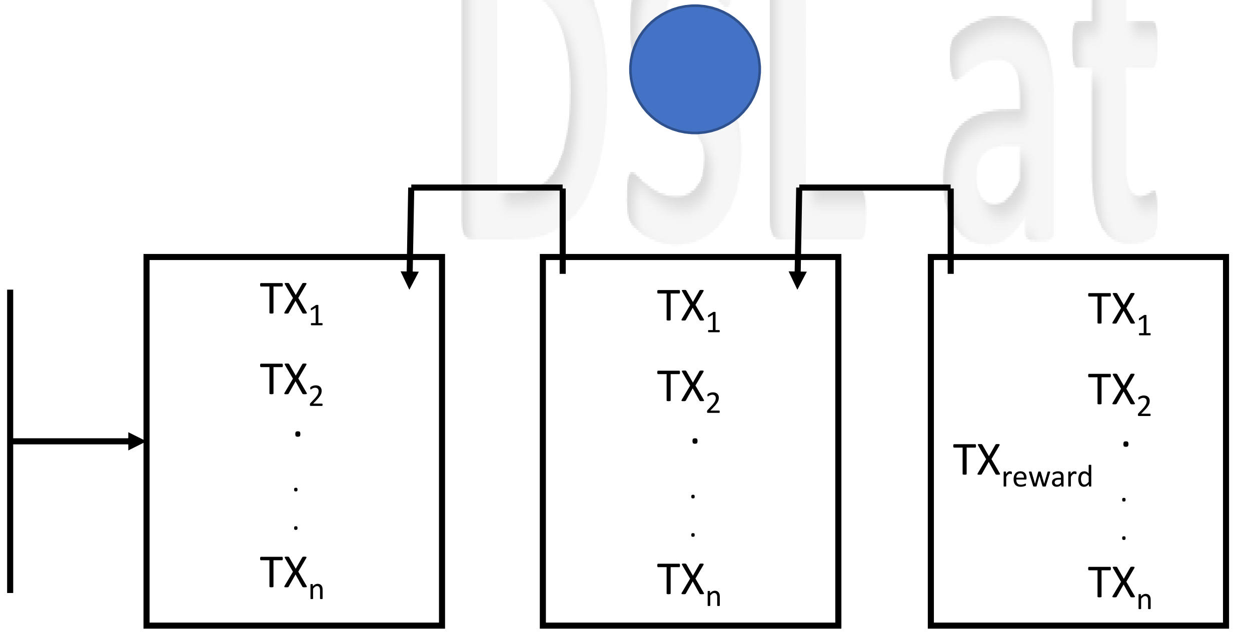
Mining Details



DSL at UCSB

Mining Details

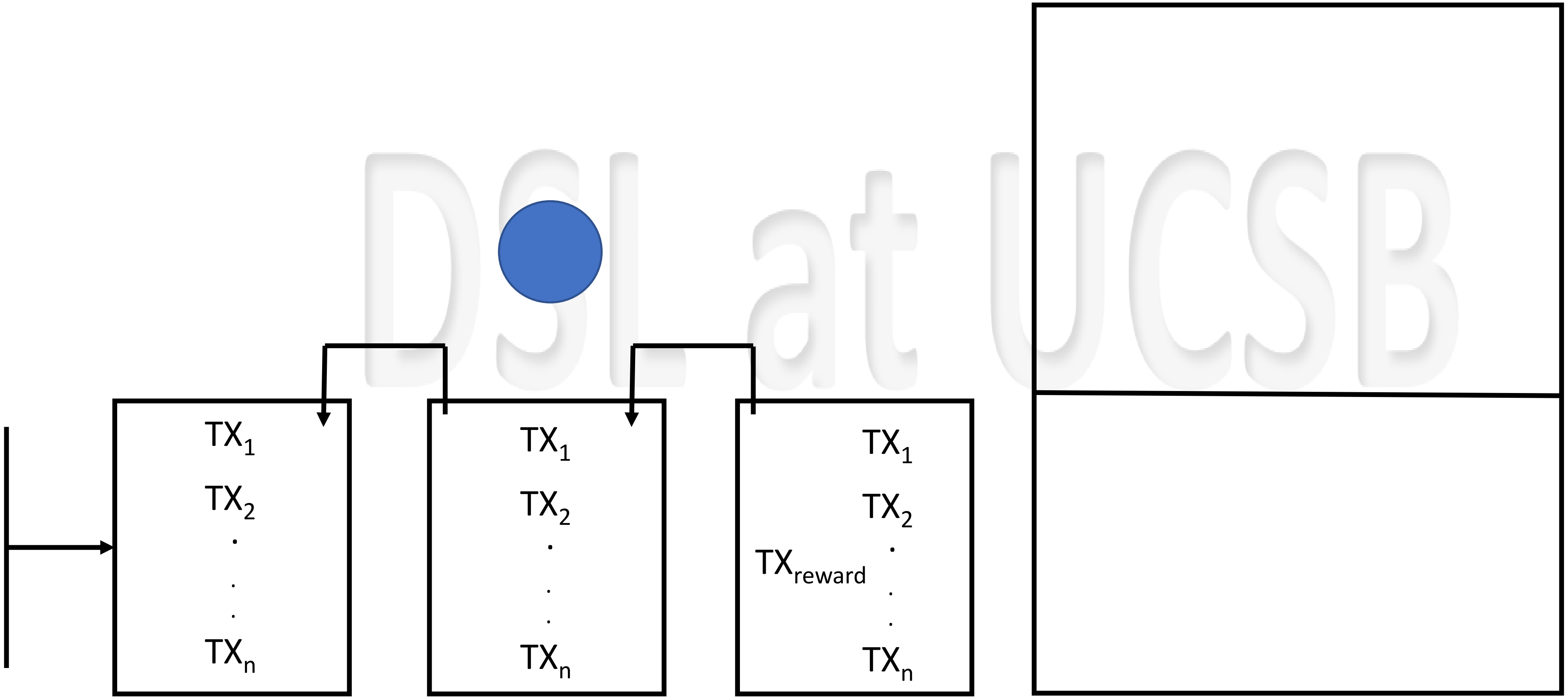
DSL at UCSB



DSL

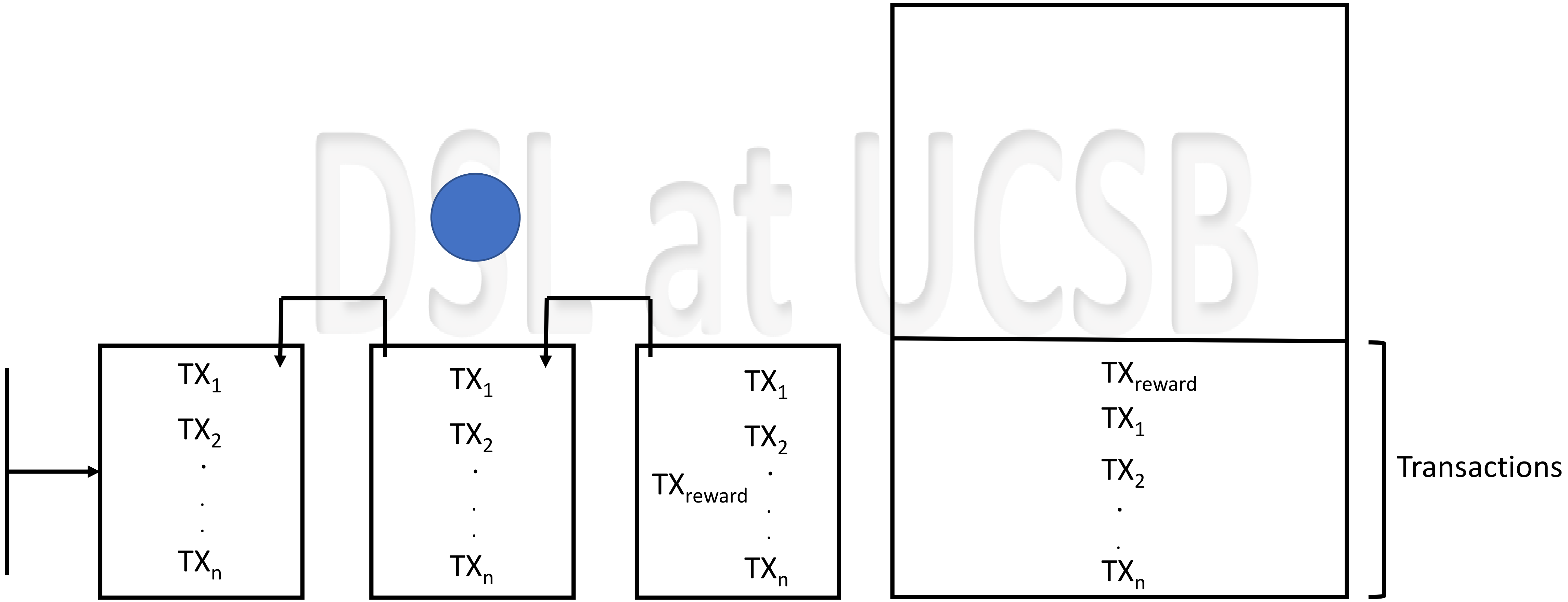
UCSB

Mining Details



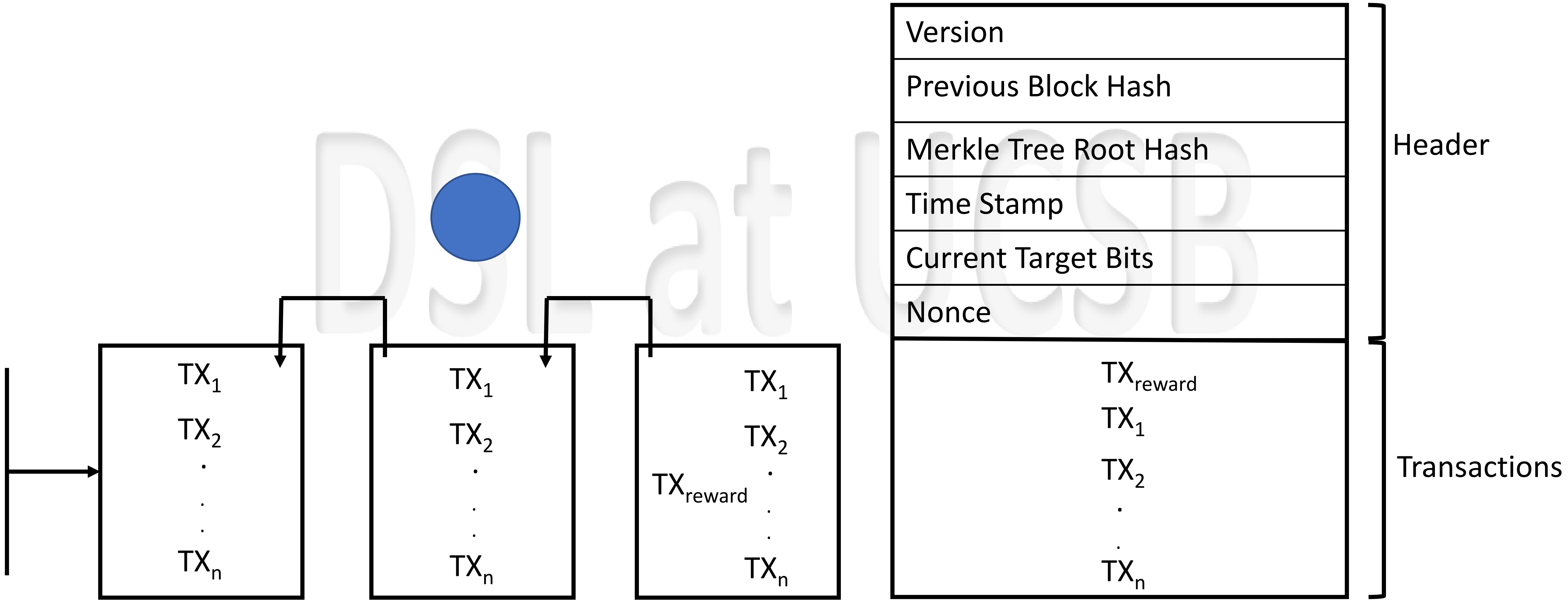
DSL at UCSB

Mining Details



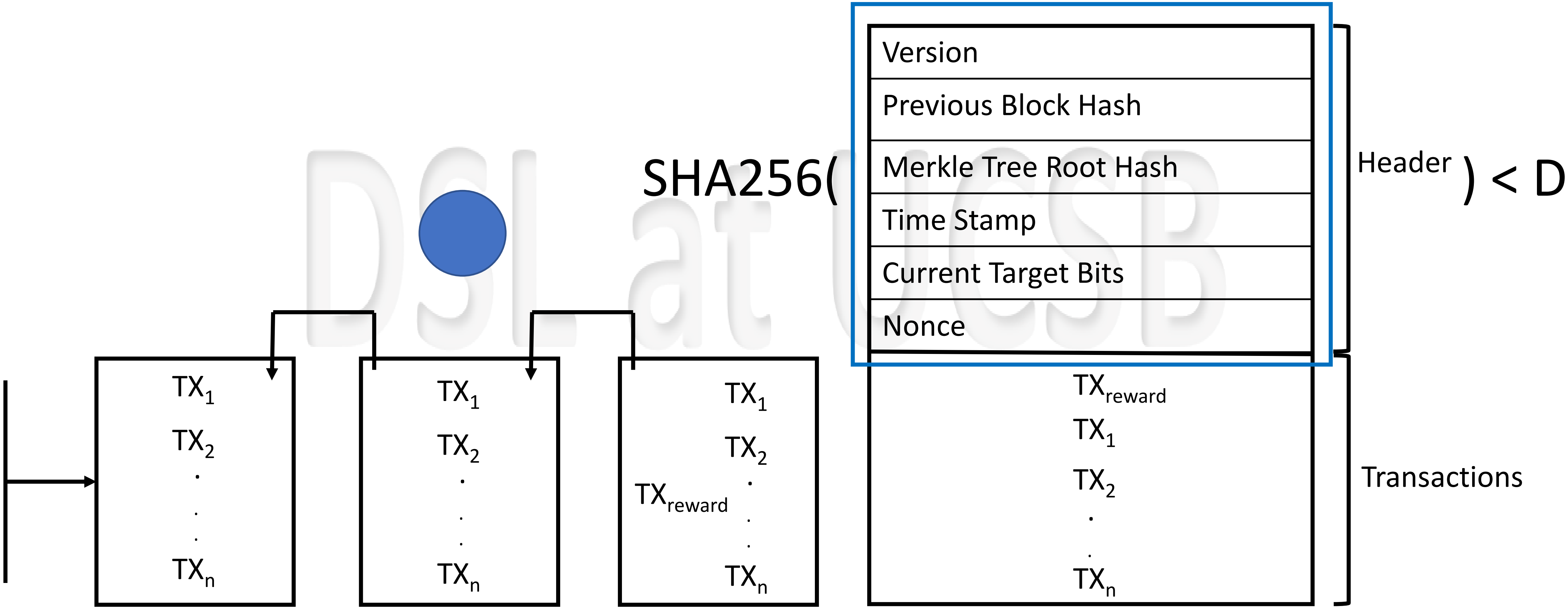
DSL at UCSB

Mining Details



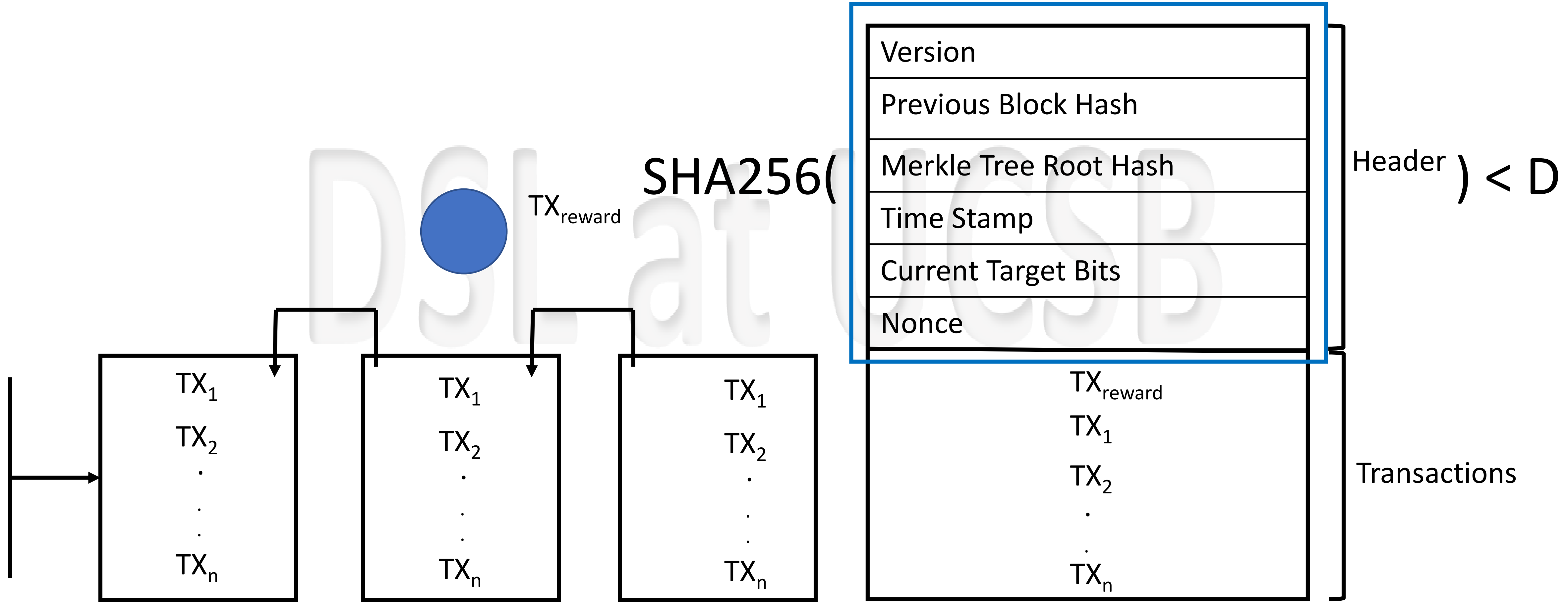
DSL at UCSB

Mining Details



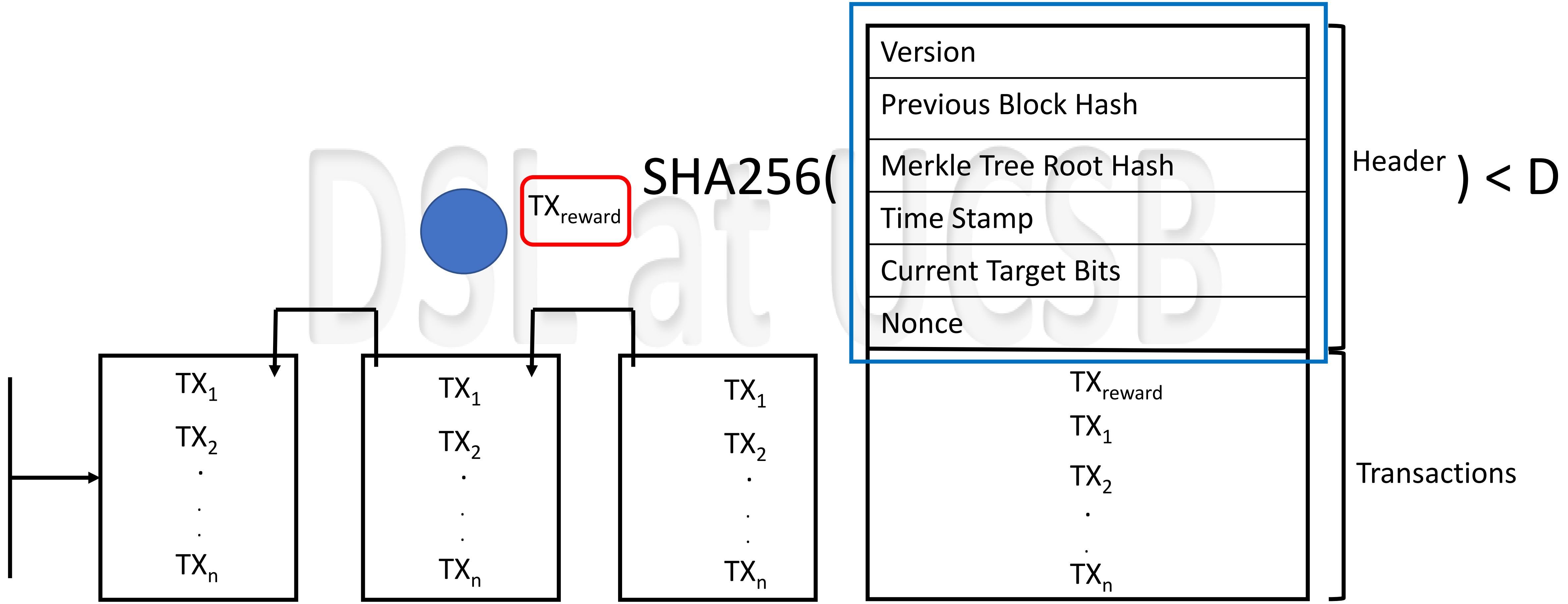
DSL at UCSB

Mining Details



DSL at UCSB

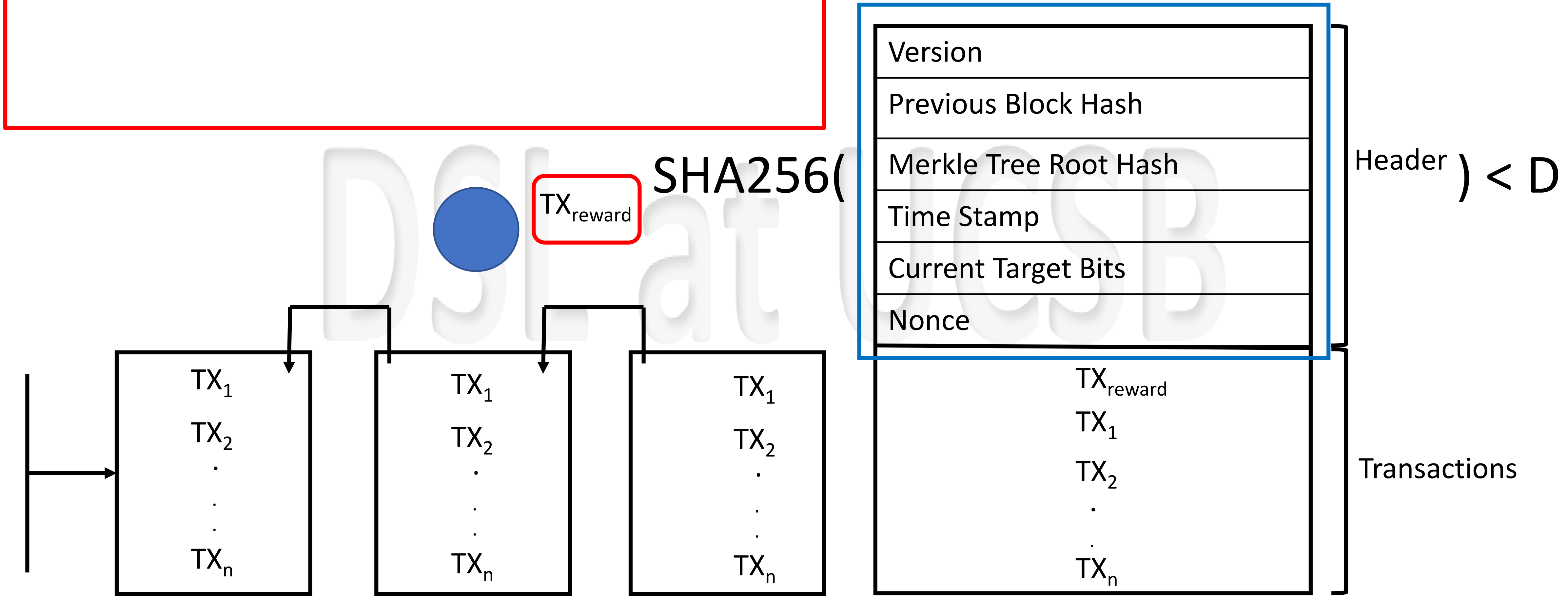
Mining Details



DSL at UCSB

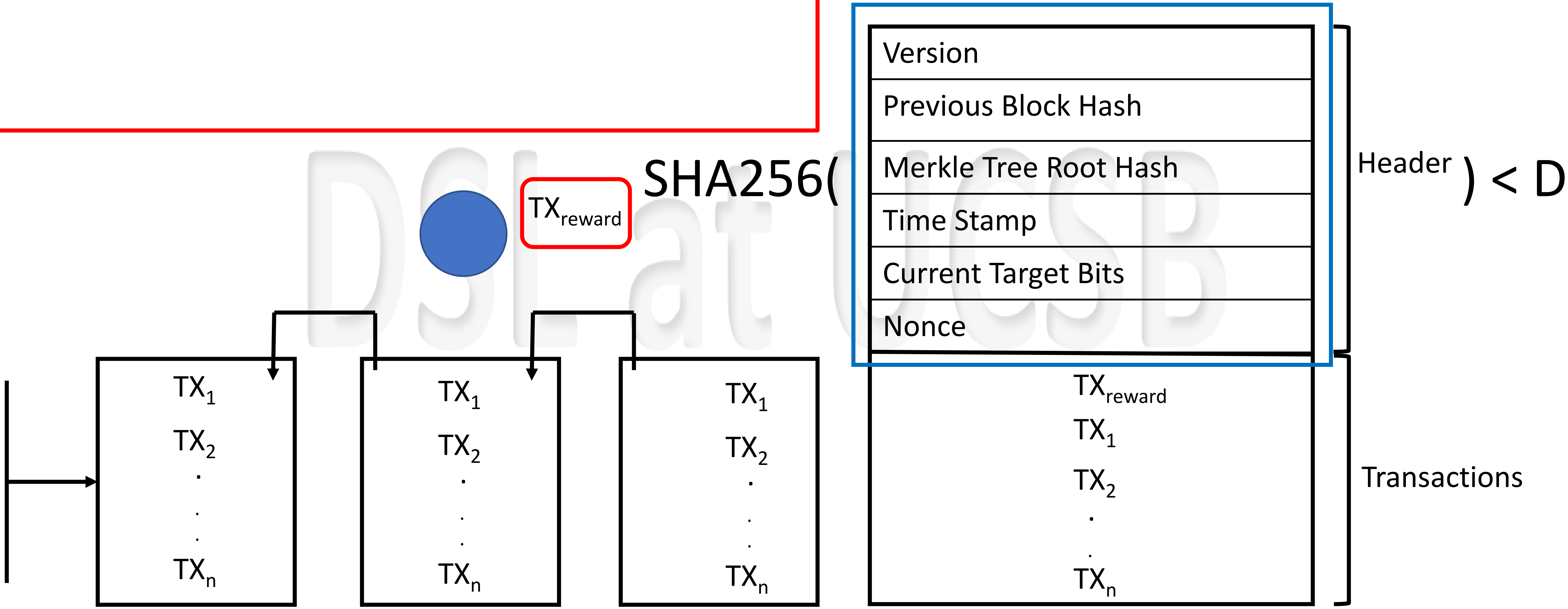
Mining Details

- TX_{reward} is self signed (also called coinbase transaction)



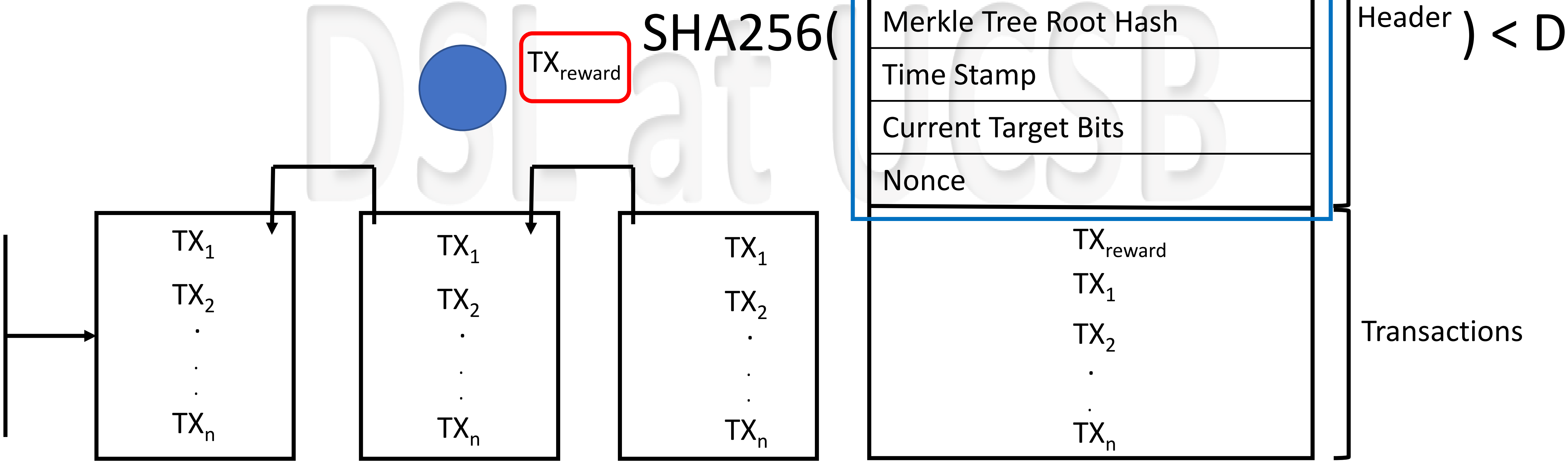
Mining Details

- TX_{reward} is self signed (also called coinbase transaction)
- TX_{reward} is bitcoin’s way to create new coins



Mining Details

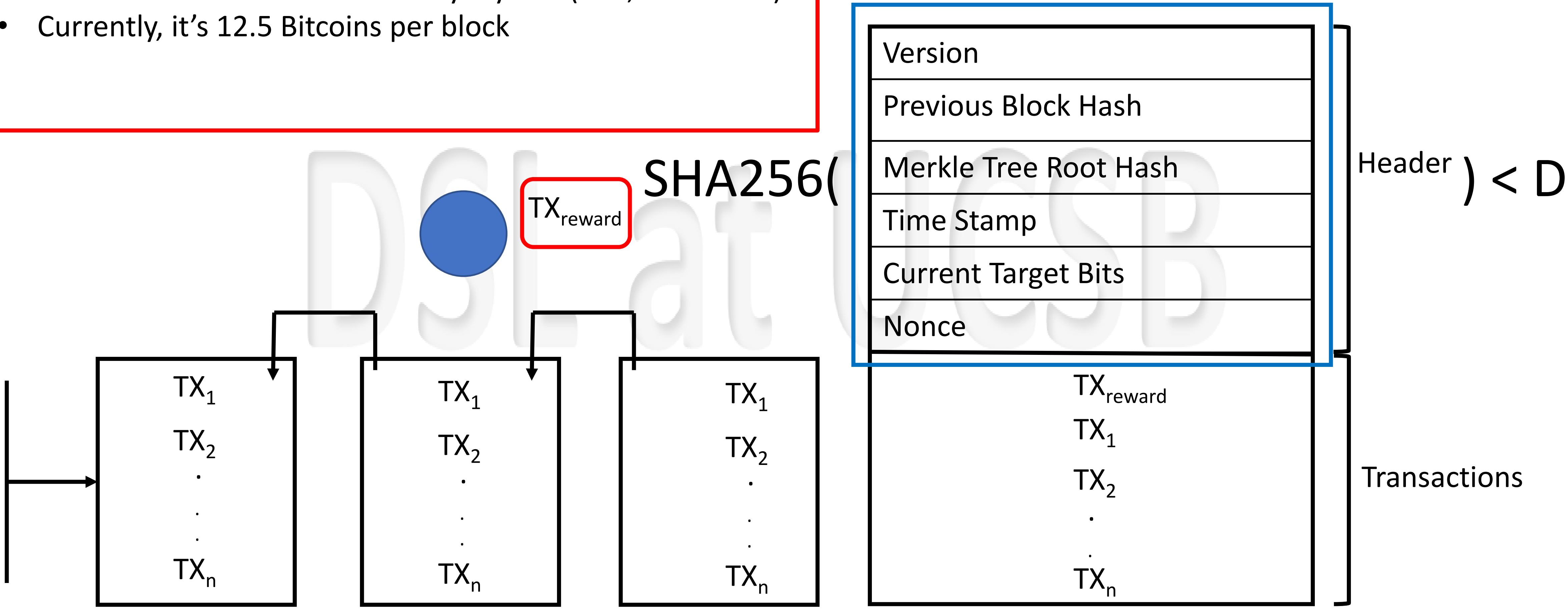
- TX_{reward} is self signed (also called coinbase transaction)
- TX_{reward} is bitcoin's way to create new coins
- The reward value is halved every 4 years (210,000 blocks)



DSL at UCSB

Mining Details

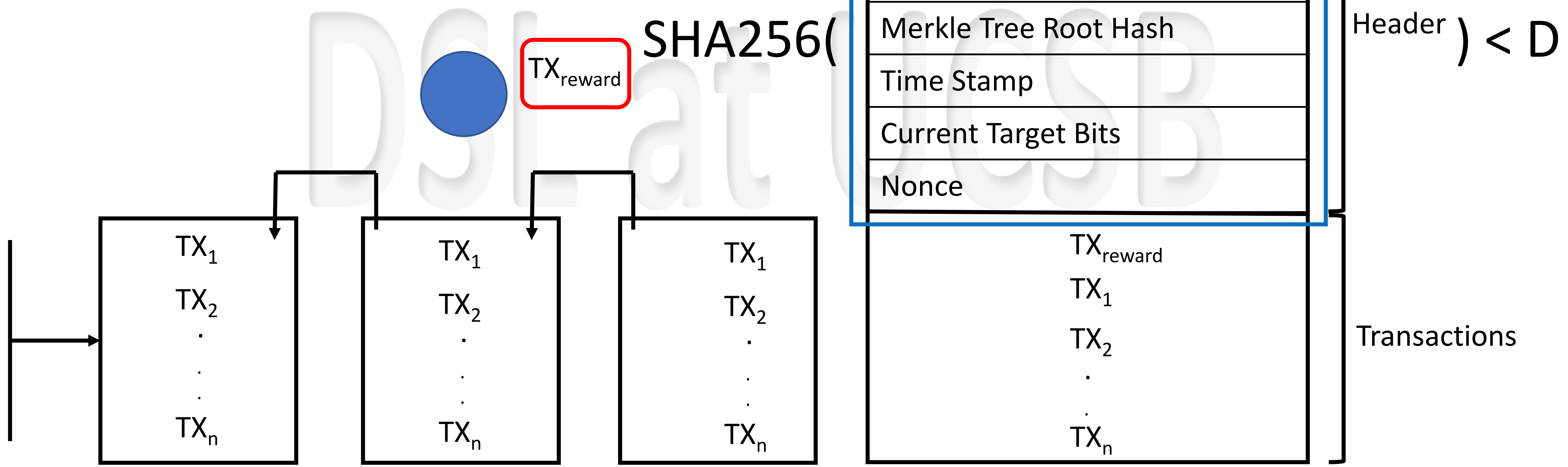
- TX_{reward} is self signed (also called coinbase transaction)
- TX_{reward} is bitcoin's way to create new coins
- The reward value is halved every 4 years (210,000 blocks)
- Currently, it's 12.5 Bitcoins per block



DSL at UCSB

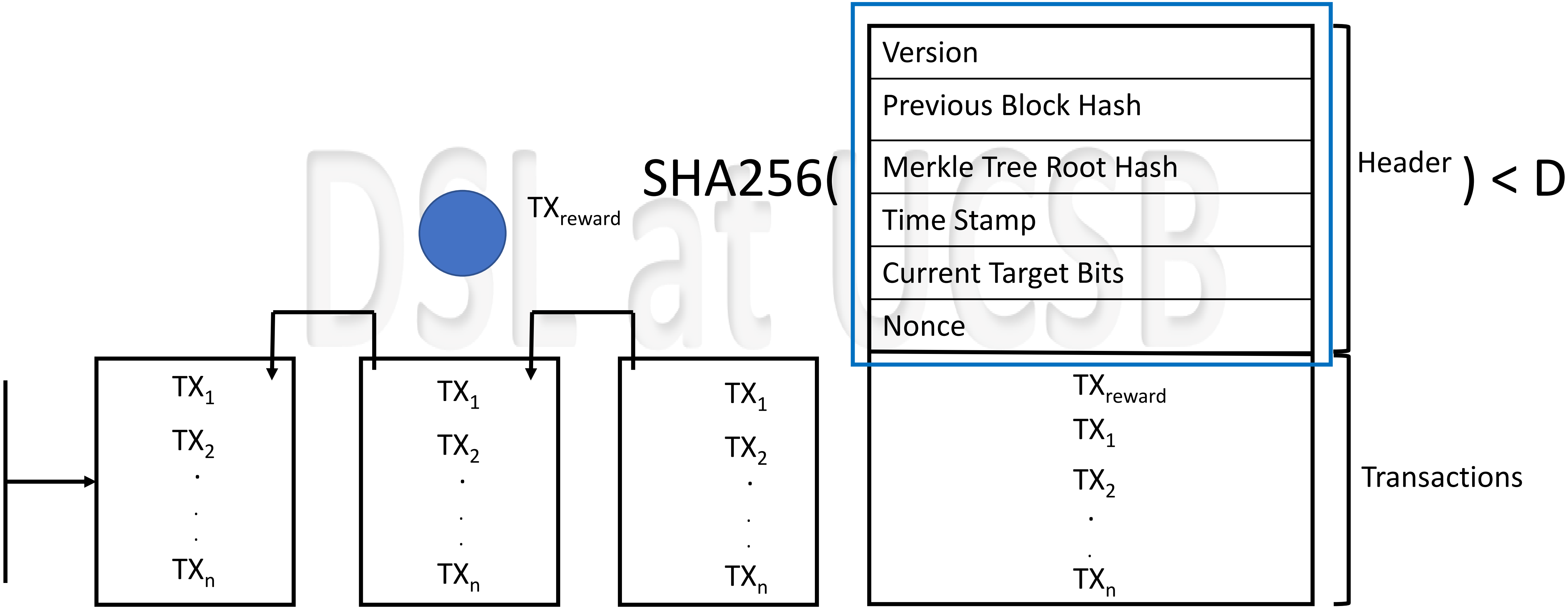
Mining Details

- TX_{reward} is self signed (also called coinbase transaction)
- TX_{reward} is bitcoin's way to create new coins
- The reward value is halved every 4 years (210,000 blocks)
- Currently, it's 12.5 Bitcoins per block
- Incentives network nodes to mine



DSL at UCSB

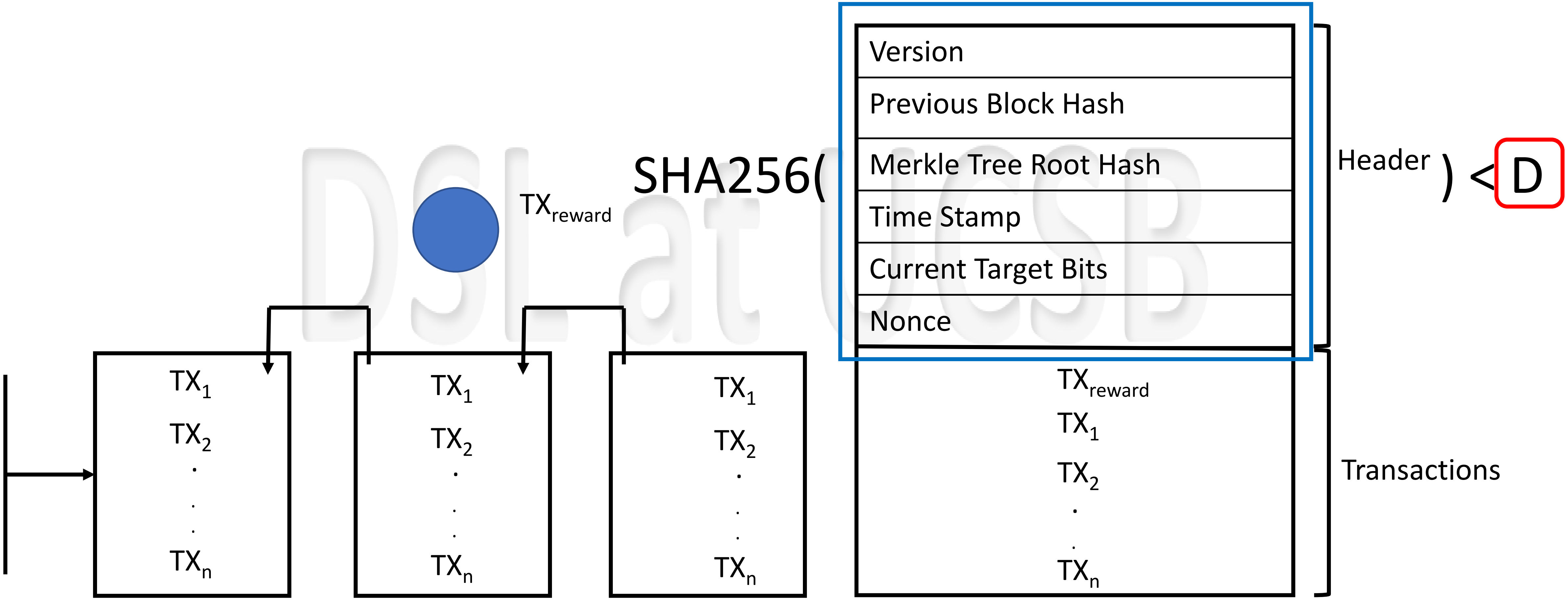
Mining Details



DSL

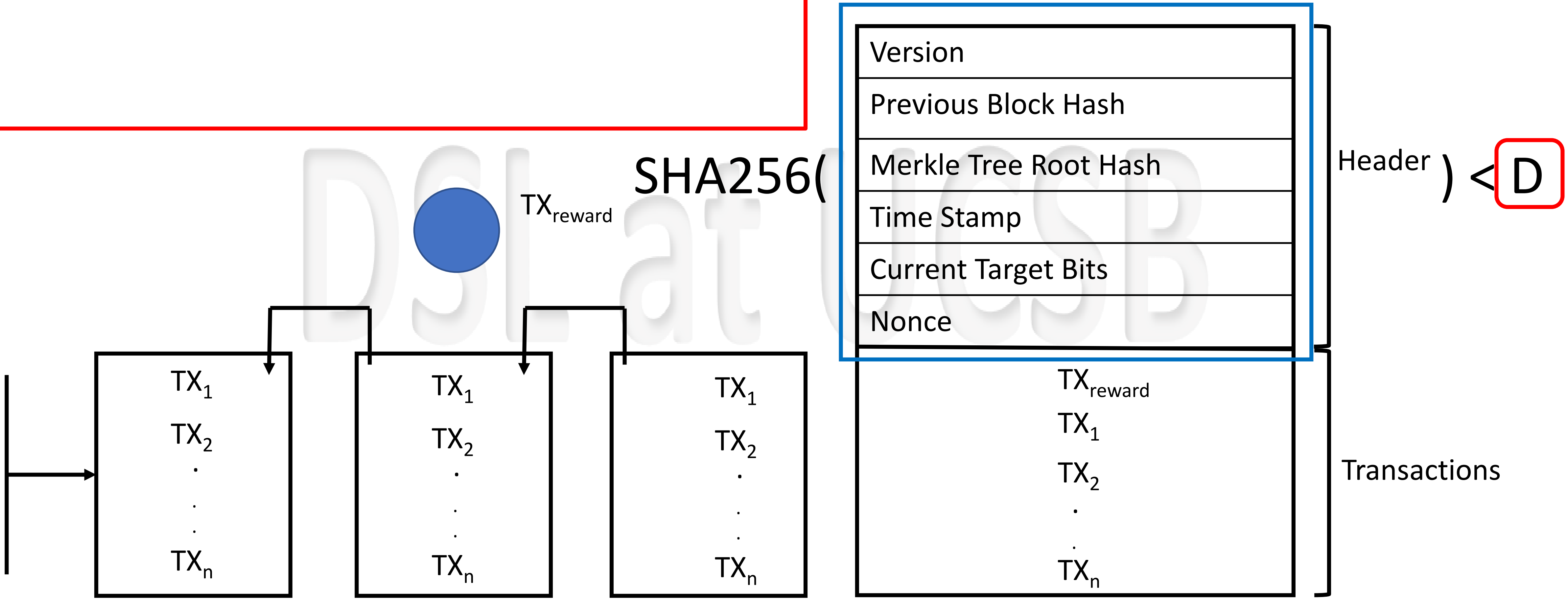
UCSB

Mining Details



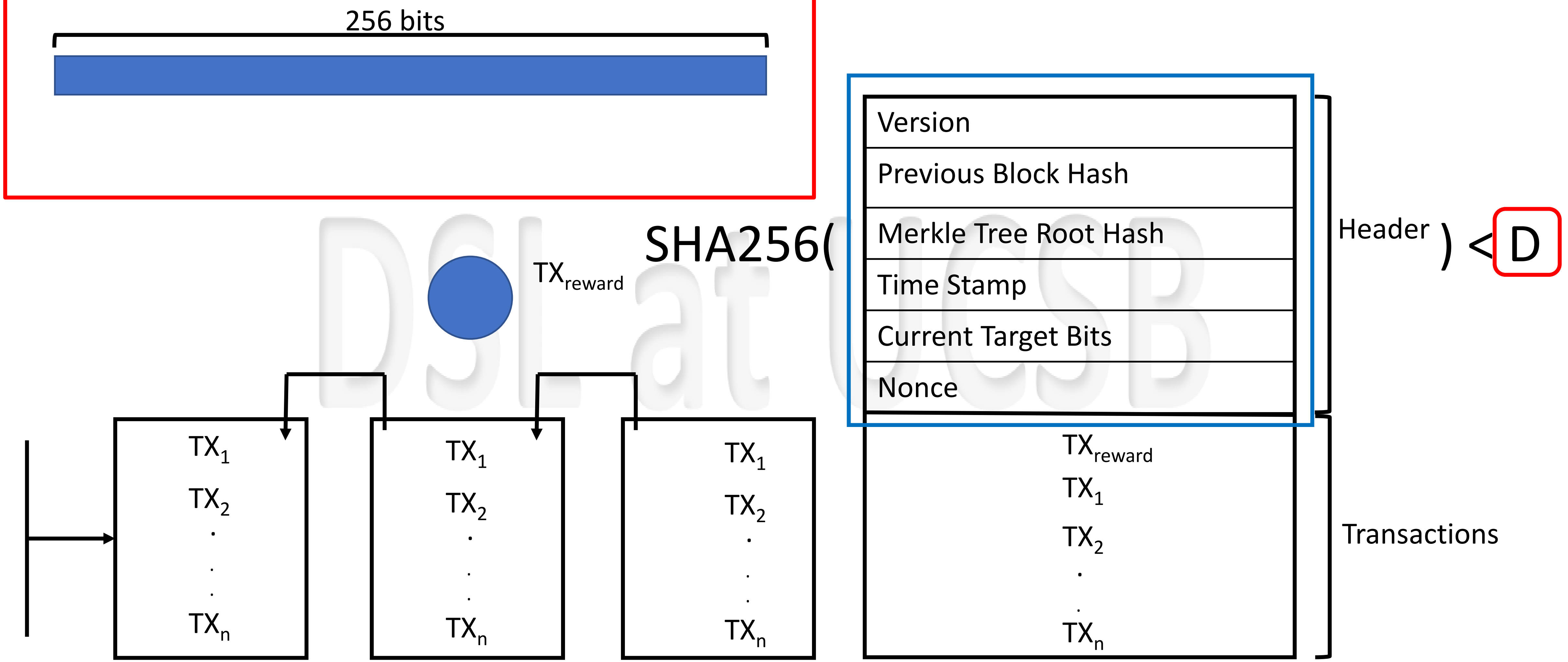
Mining Details

- D: dynamically adjusted difficulty



Mining Details

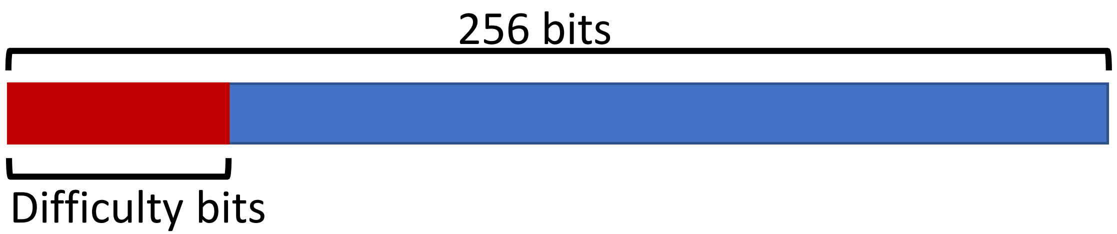
- D: dynamically adjusted difficulty

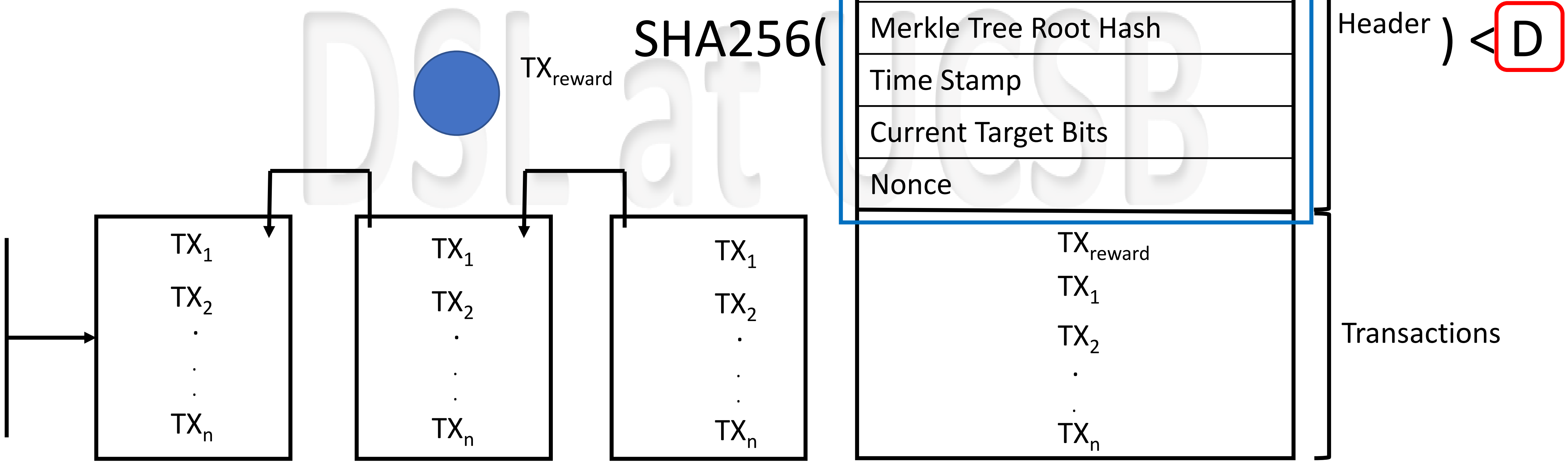


DSL at UCSB

Mining Details

- D: dynamically adjusted difficulty

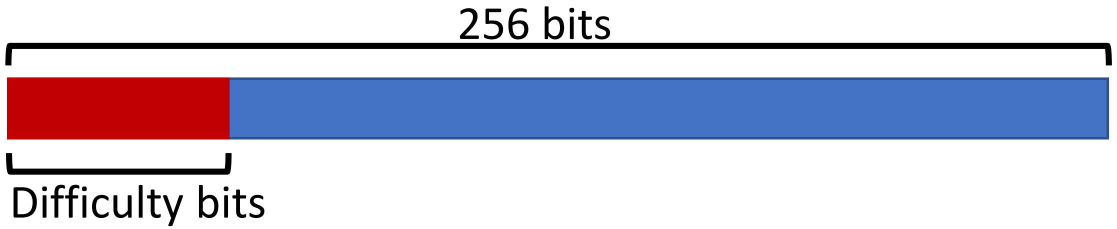




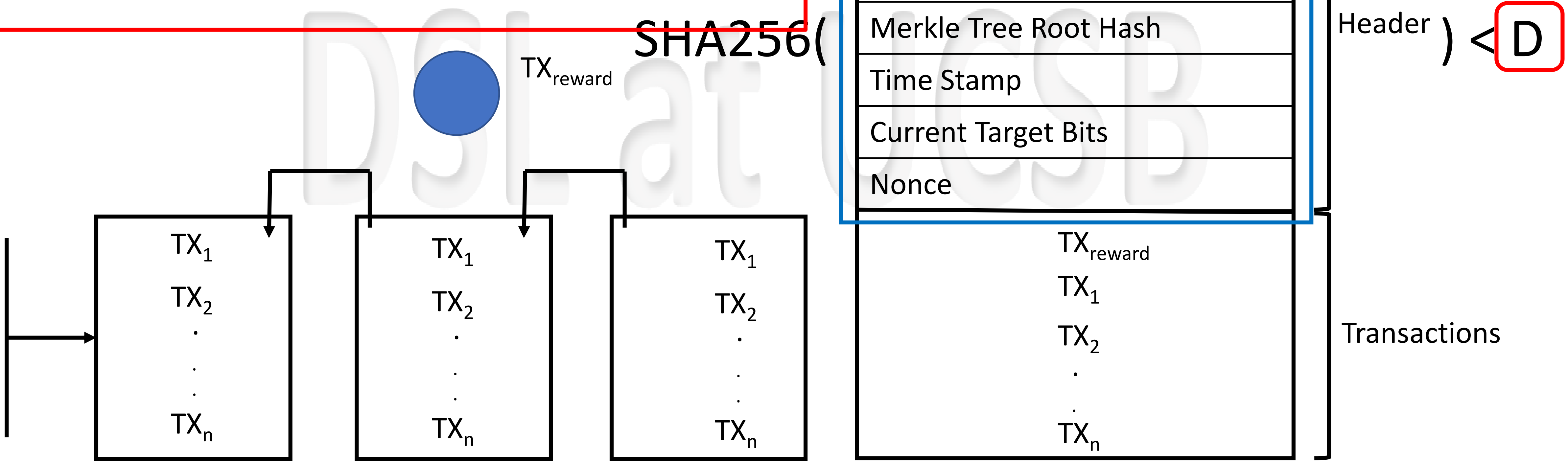
DSL at UCSB

Mining Details

- D: dynamically adjusted difficulty



- Difficulty is adjusted every 2016 blocks (almost 2 weeks)



DSL at UCSB

Difficulty

DSL at UCSB

DSL

UCSB

Difficulty

- Adjust difficulty every 2016 blocks

DSL at UCSB

Difficulty

- Adjust difficulty every 2016 blocks
- Expected 20160 mins to mine (10 mins per block)

DSL at UCSB

Difficulty

- Adjust difficulty every 2016 blocks
- Expected 20160 mins to mine (10 mins per block)
- Actual time = timestamp of block 2016 – time stamp of block 1

DSL at UCSB

Difficulty

- Adjust difficulty every 2016 blocks
- Expected 20160 mins to mine (10 mins per block)
- Actual time = timestamp of block 2016 – time stamp of block 1
- New_difficulty = old_difficulty * expected/actual

DSL at UCSB

Difficulty

- Adjust difficulty every 2016 blocks
- Expected 20160 mins to mine (10 mins per block)
- Actual time = timestamp of block 2016 – time stamp of block 1
- New_difficulty = old_difficulty * expected/actual
- Difficulty decreases if actual > expected, otherwise, increases

DSL at UCSB

DSL

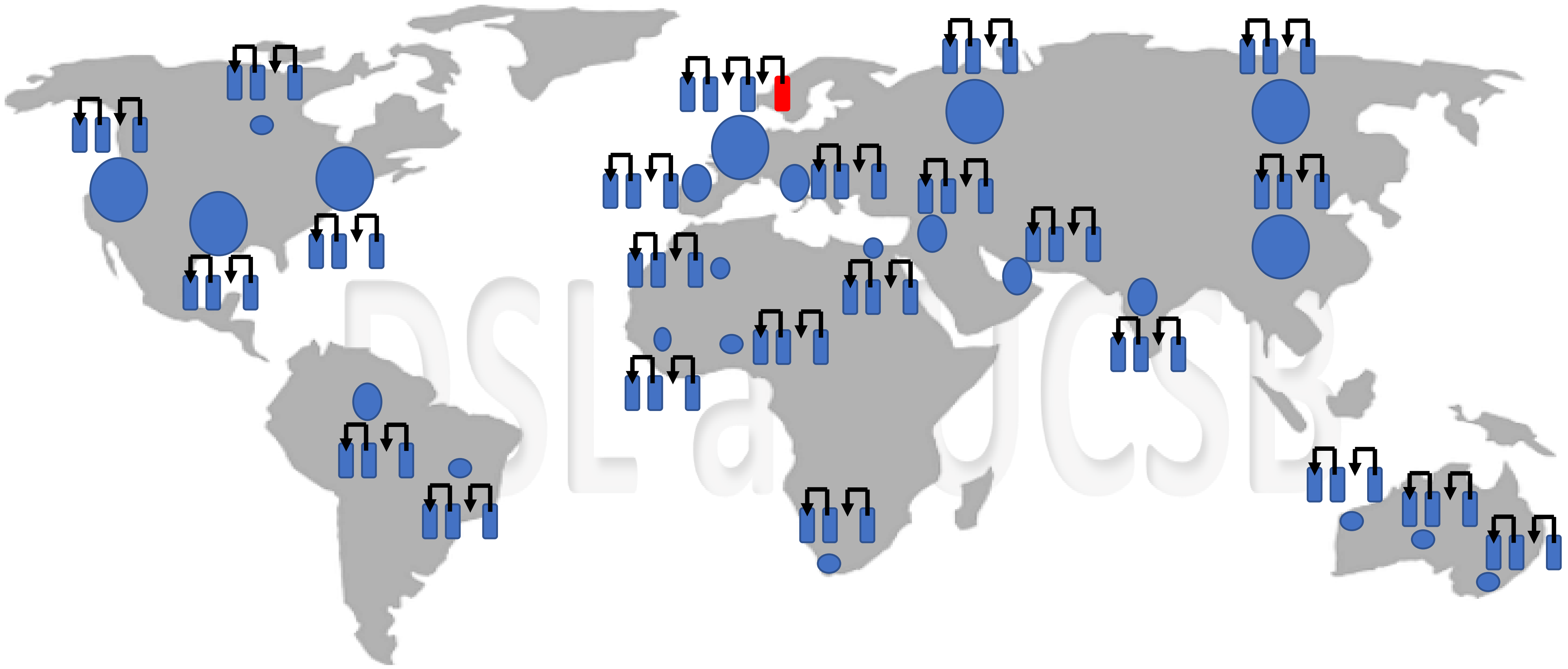
UCSB

Mining Big Picture

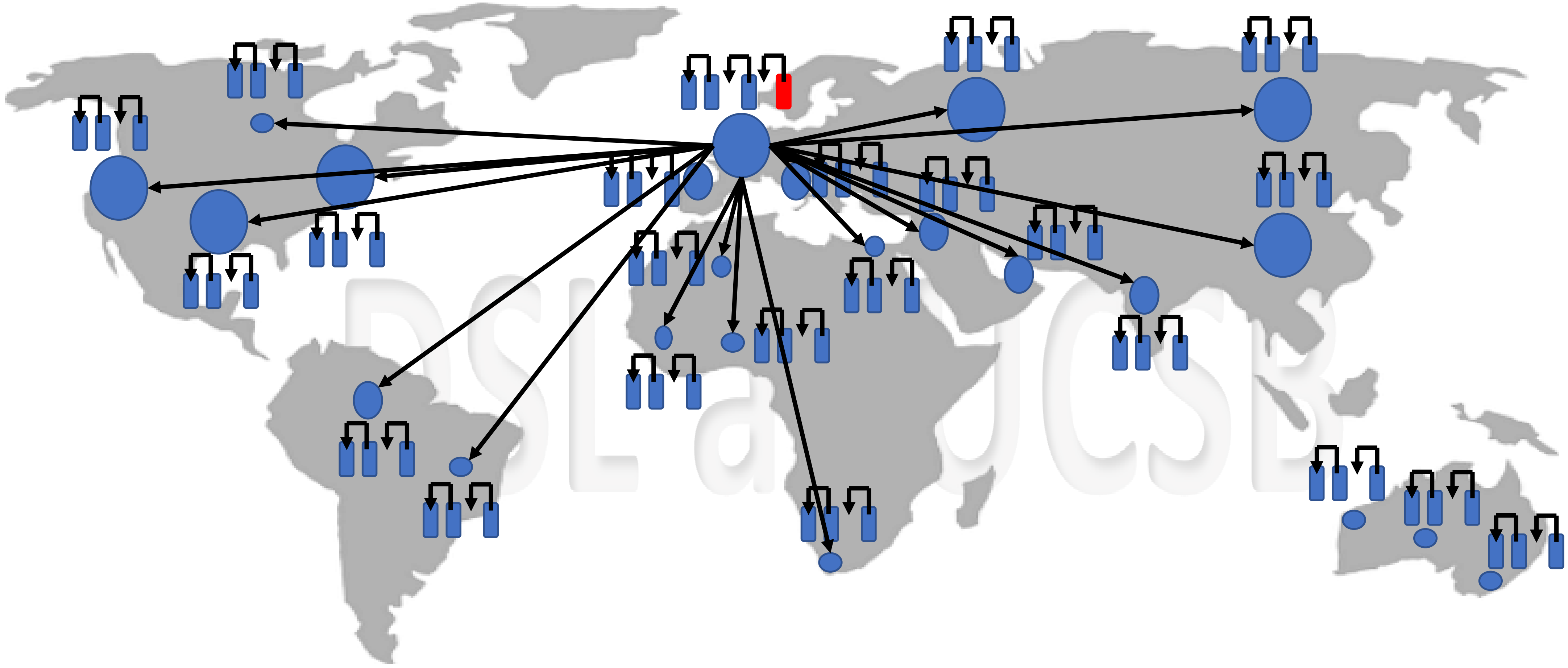
DSL

UCSB

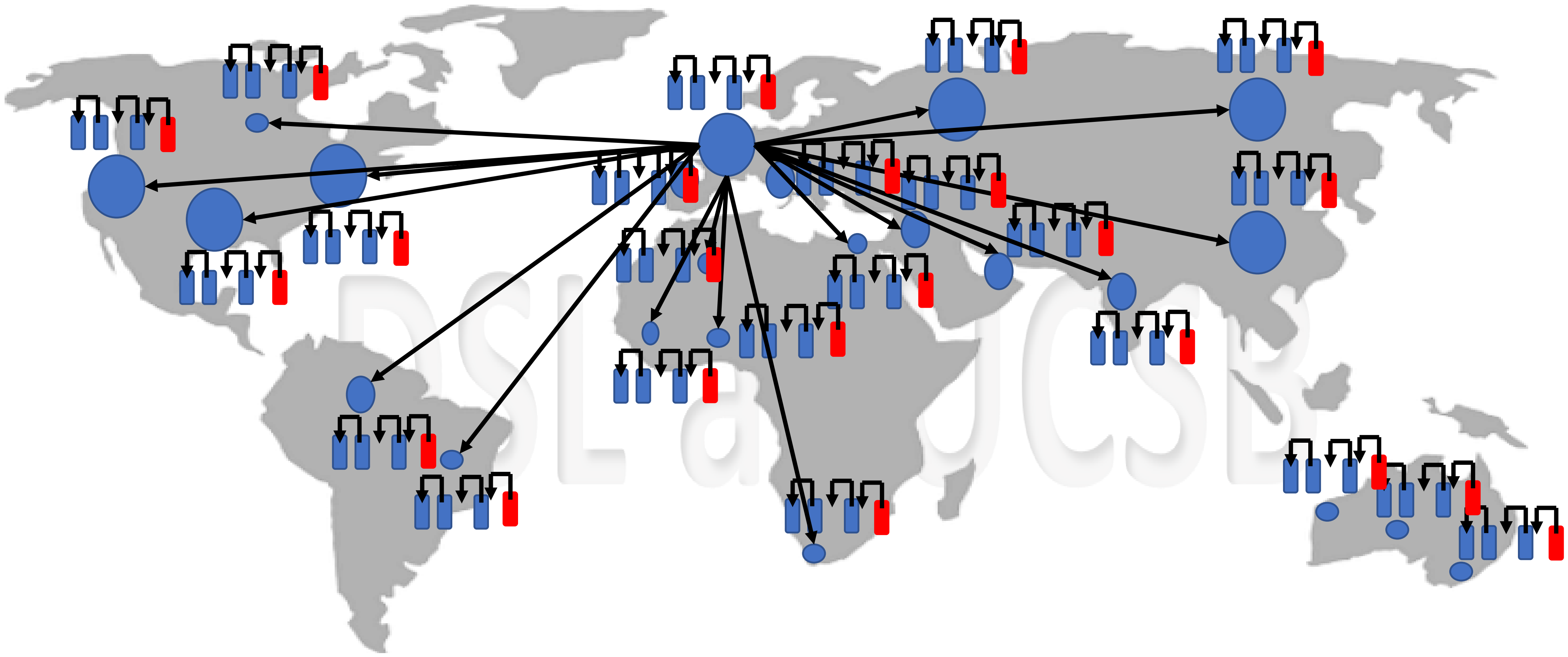
Mining Big Picture


DSL @ UCSB

Mining Big Picture



Mining Big Picture



Mining Details

- Find a nonce that results in SHA256(block) < Difficulty

DSL at UCSB

Mining Details

- Find a nonce that results in SHA256(block) < Difficulty
- The solution space is a set. Once a solution is found, a block is mined

DSL at UCSB

Mining Details

- Find a nonce that results in SHA256(block) < Difficulty
- The solution space is a set. Once a solution is found, a block is mined
- Easily verified by network nodes

DSL at UCSB

Mining Details

- Find a nonce that results in SHA256(block) < Difficulty
- The solution space is a set. Once a solution is found, a block is mined
- Easily verified by network nodes
- Cannot be precomputed
  - Depends on current block transactions and previous blocks

DSL at UCSB

Mining Details

- Find a nonce that results in SHA256(block) < Difficulty
- The solution space is a set. Once a solution is found, a block is mined
- Easily verified by network nodes
- Cannot be precomputed
  - Depends on current block transactions and previous blocks
- Cannot be stolen
  - Reward Transaction is signed to the public key of the miner

DSL at UCSB

Mining Details

- Find a nonce that results in SHA256(block) < Difficulty
- The solution space is a set. Once a solution is found, a block is mined
- Easily verified by network nodes
- Cannot be precomputed
  - Depends on current block transactions and previous blocks
- Cannot be stolen
  - Reward Transaction is signed to the public key of the miner
- Network nodes accept the first found block:
  - The problem is difficult, there is no guaranteed bound to find another block

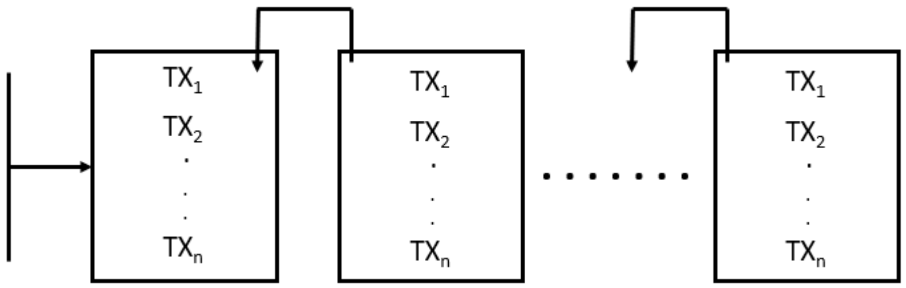
DSL at UCSB

Mining Details

- Find a nonce that results in SHA256(block) < Difficulty
- The solution space is a set. Once a solution is found, a block is mined
- Easily verified by network nodes
- Cannot be precomputed
  - Depends on current block transactions and previous blocks
- Cannot be stolen
  - Reward Transaction is signed to the public key of the miner
- Network nodes accept the first found block:
  - The problem is difficult, there is no guaranteed bound to find another block
- What happens when 2 nodes concurrently mine a block? Fork

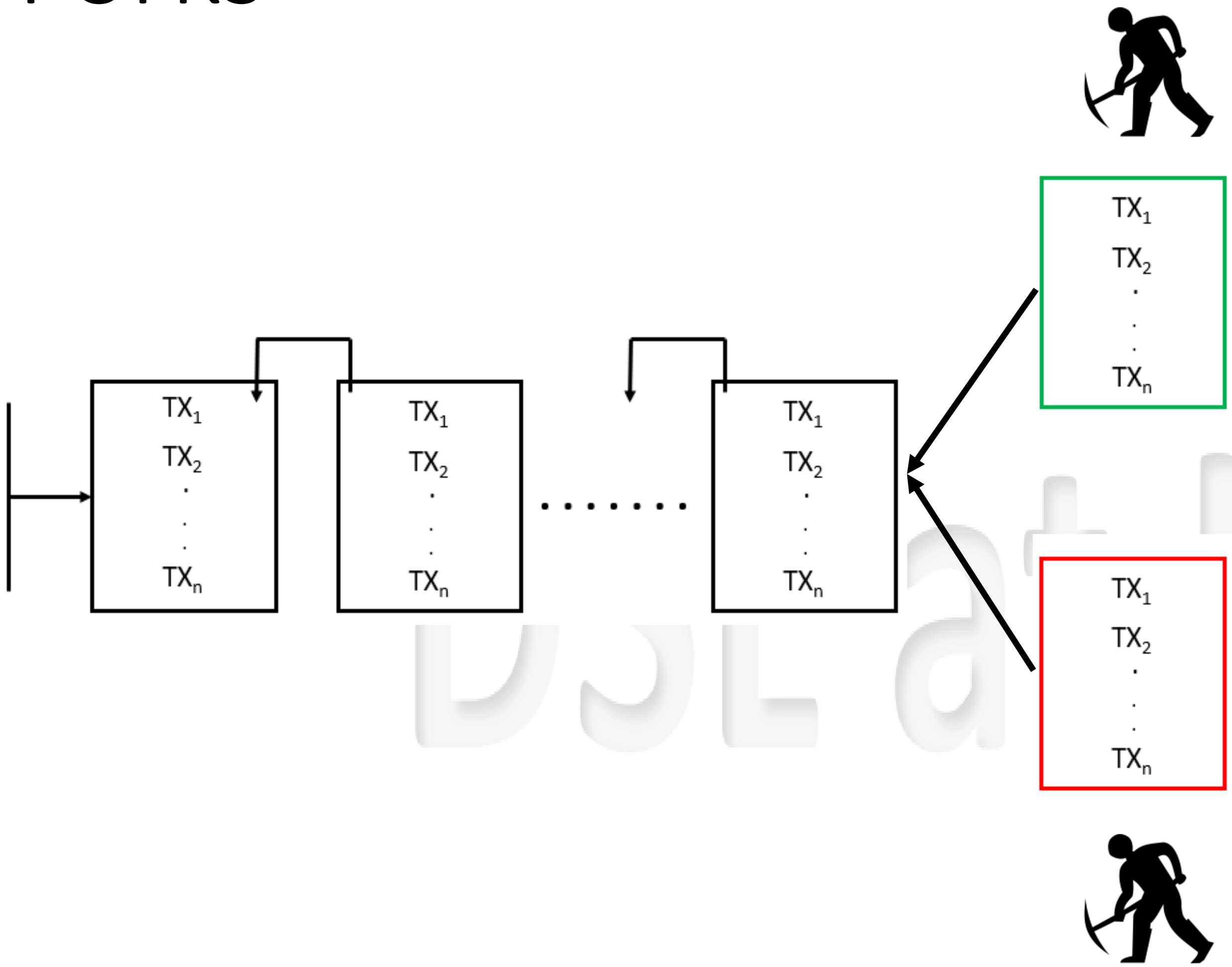
DSL at UCSB

Forks



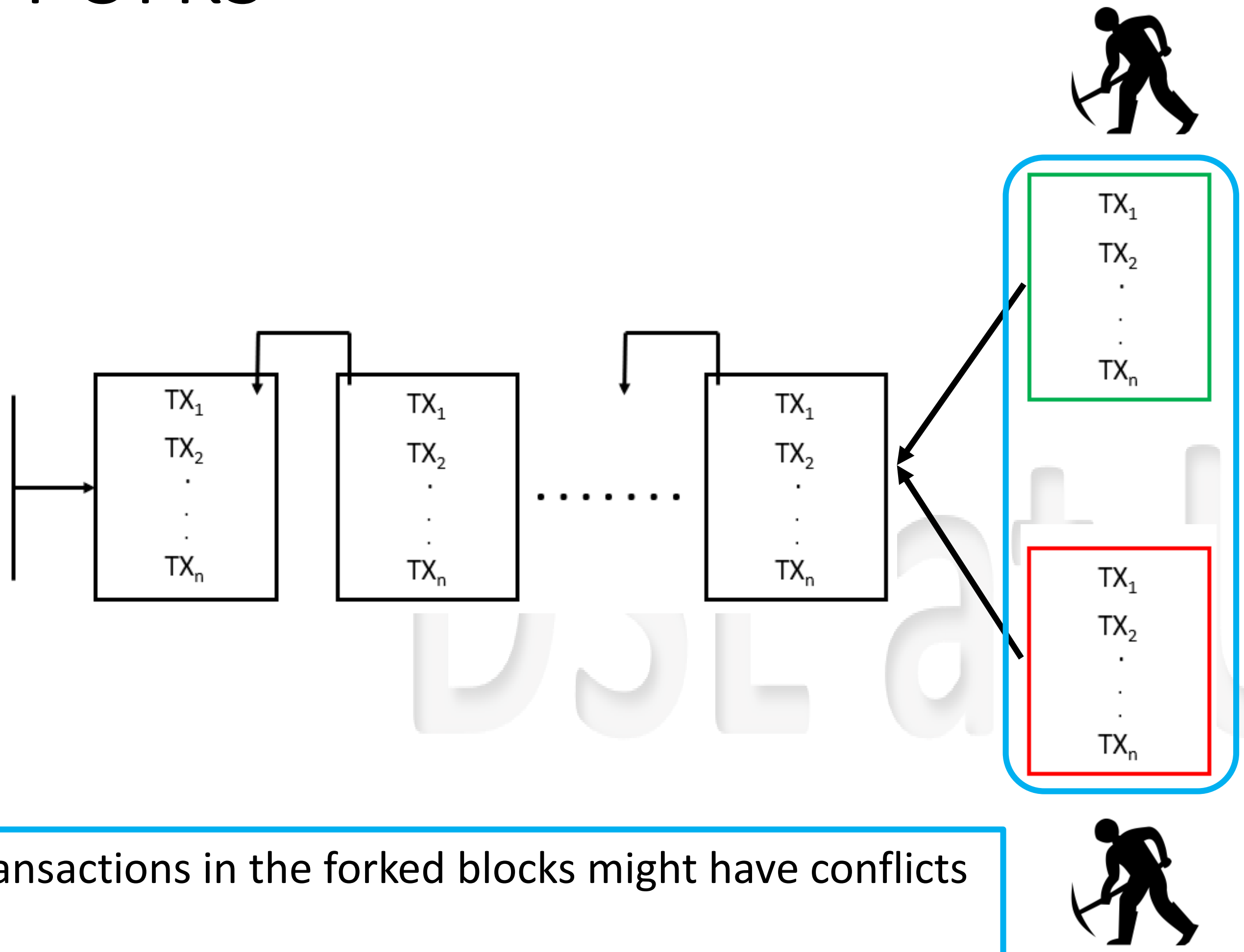
DSL at UCSB

Forks



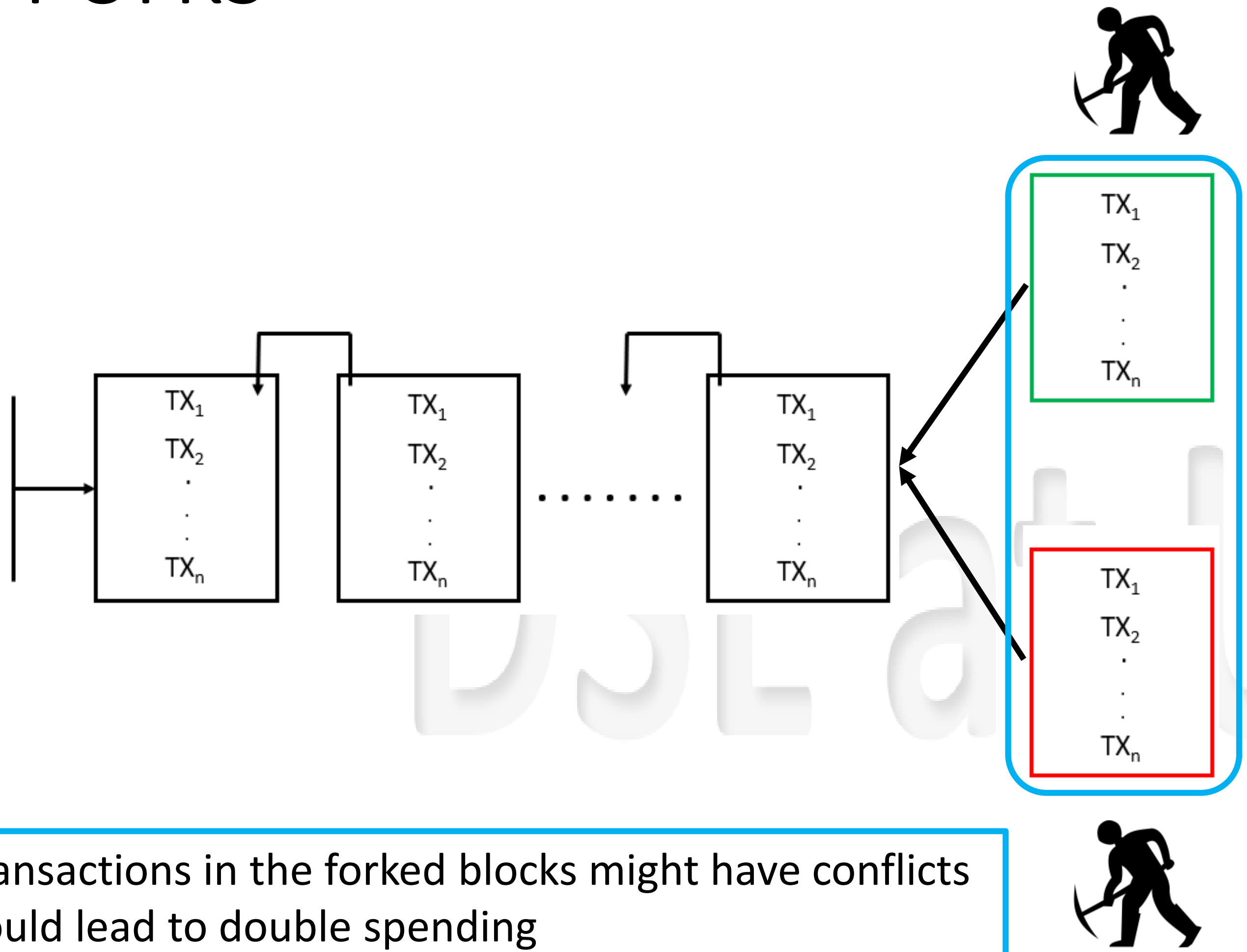
DSL at UCSB

Forks



- Transactions in the forked blocks might have conflicts

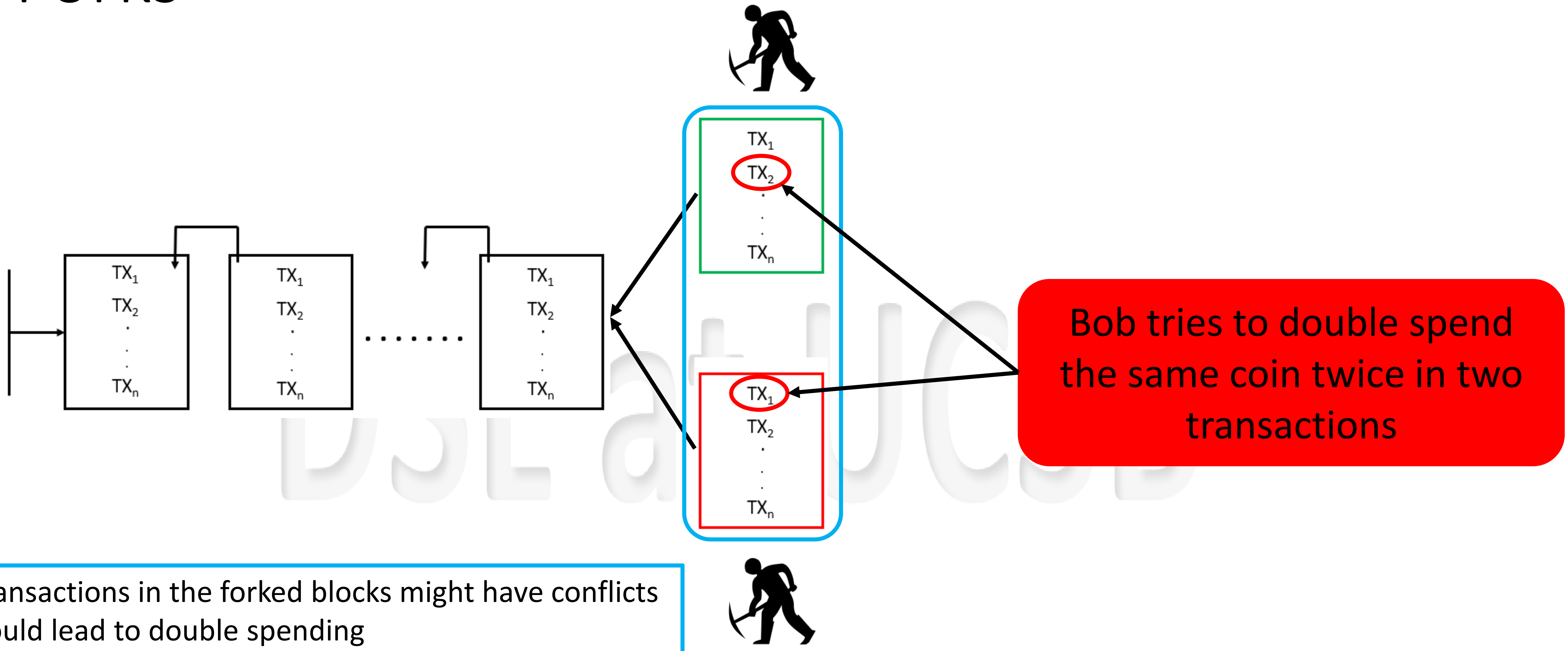
Forks



- Transactions in the forked blocks might have conflicts
- Could lead to double spending

DSL at UCSB

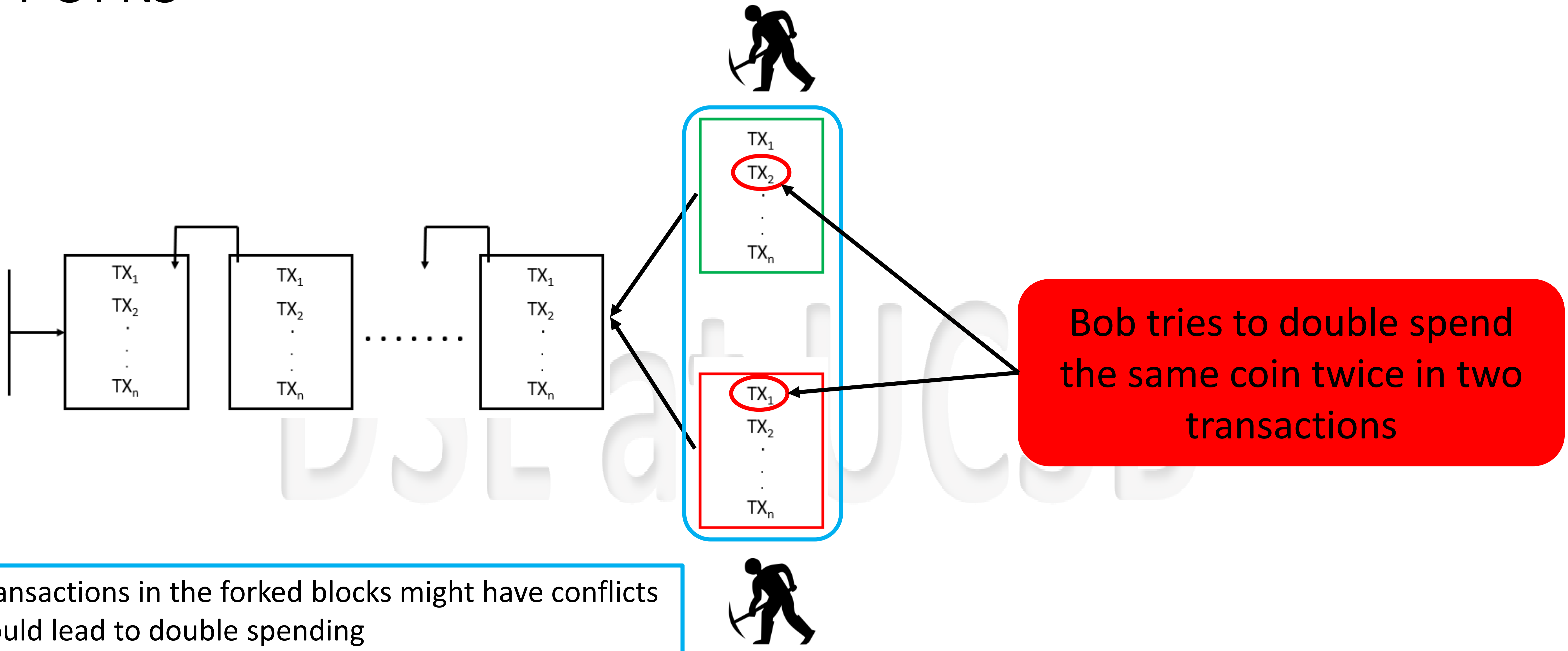
Forks



- Transactions in the forked blocks might have conflicts
- Could lead to double spending

DSL at UCSB

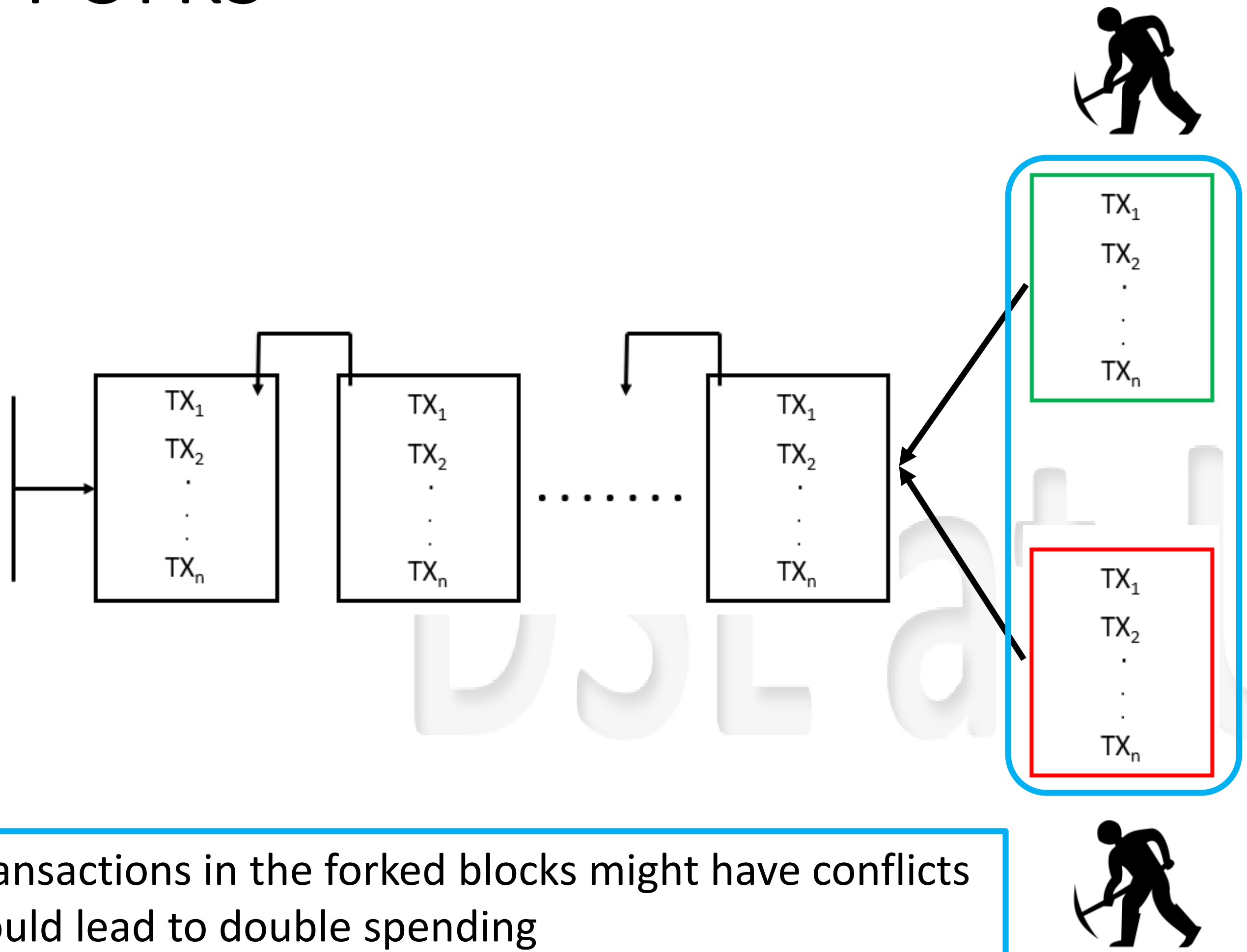
Forks



- Transactions in the forked blocks might have conflicts
- Could lead to double spending
- Forks have to be eliminated

DSL at UCSB

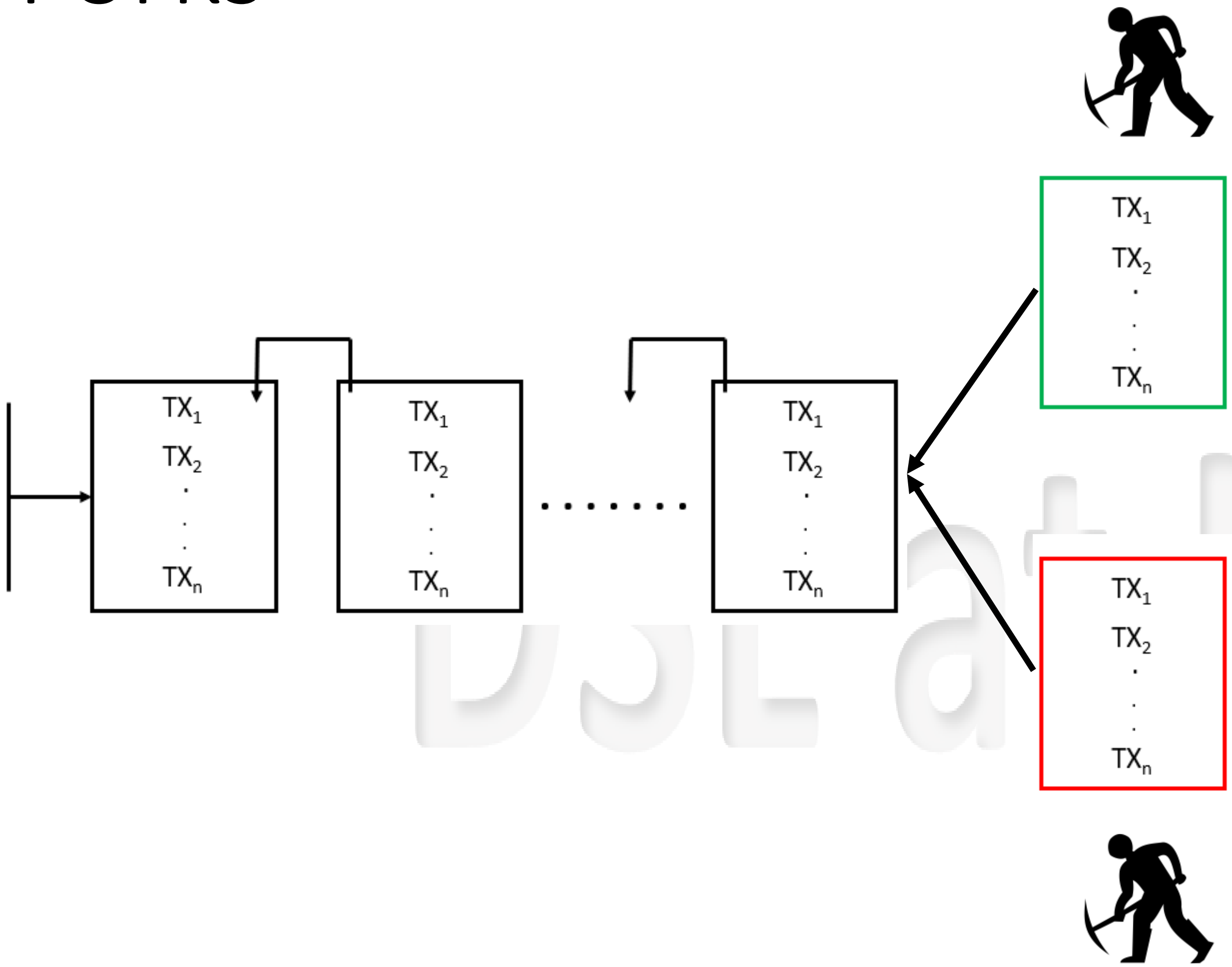
Forks



- Transactions in the forked blocks might have conflicts
- Could lead to double spending
- Forks have to be eliminated

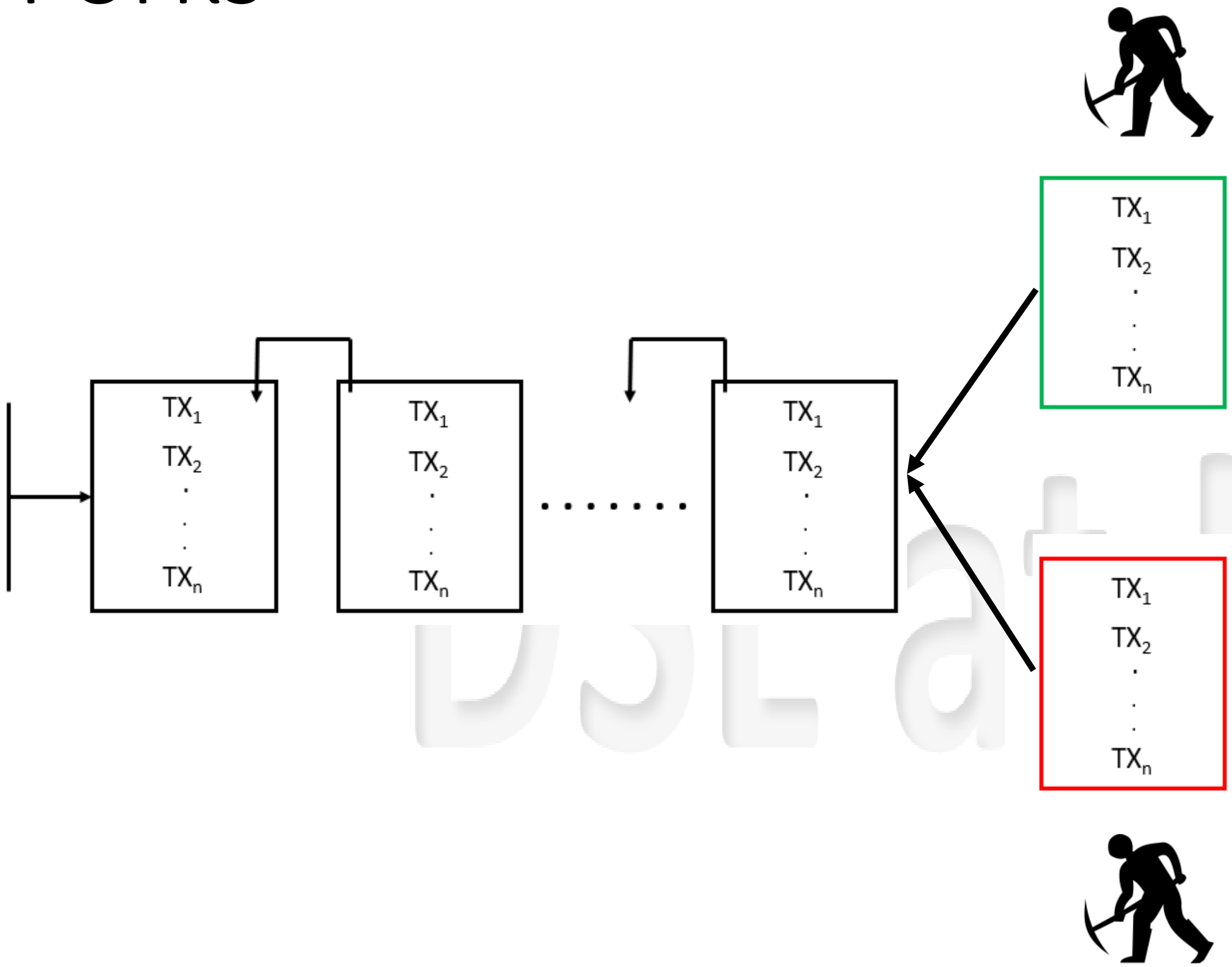
DSL at UCSB

Forks



DSL at UCSB

Forks

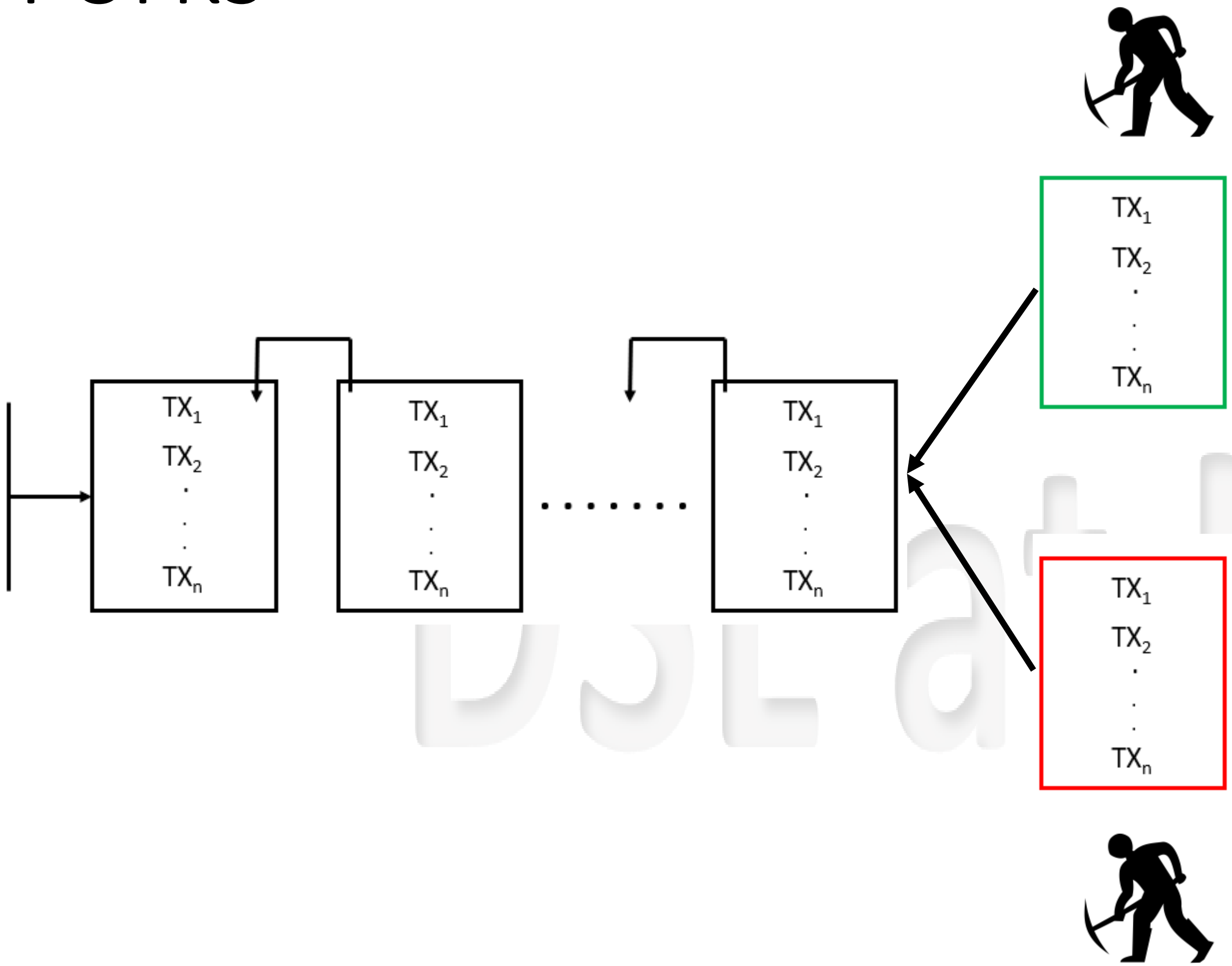


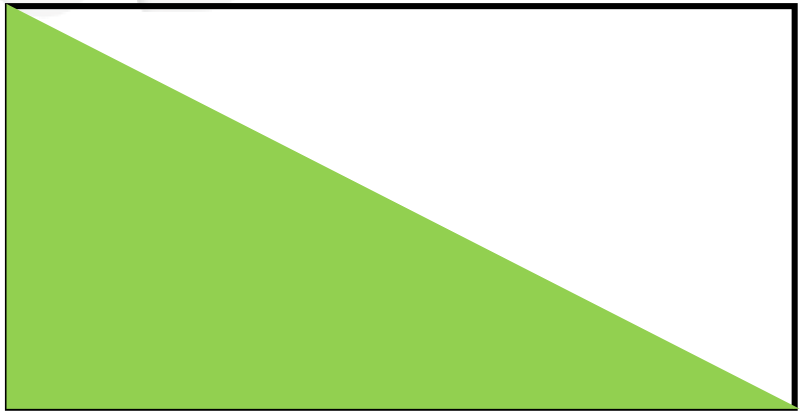
DSL at UCSB

DSL

UCSB

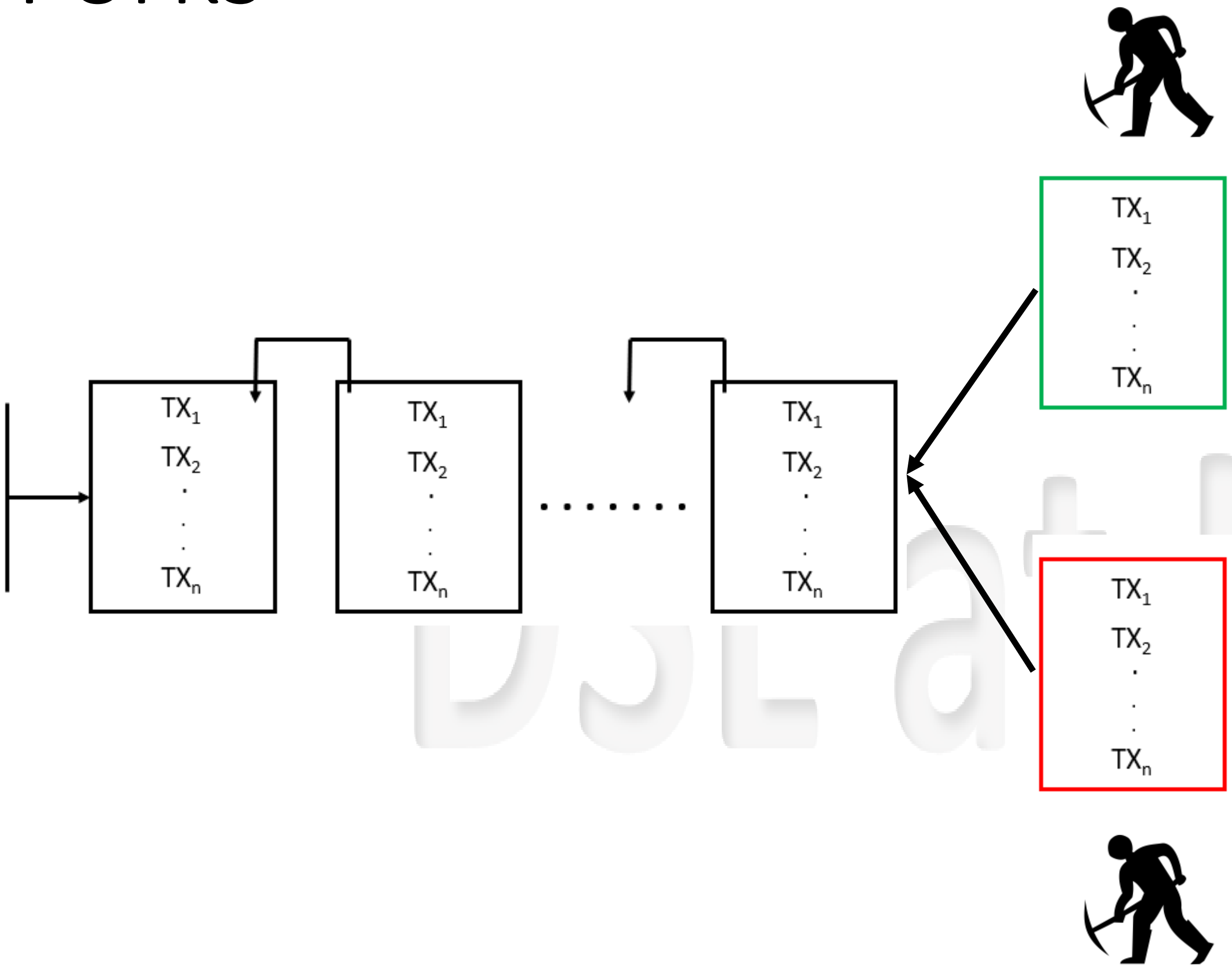
Forks



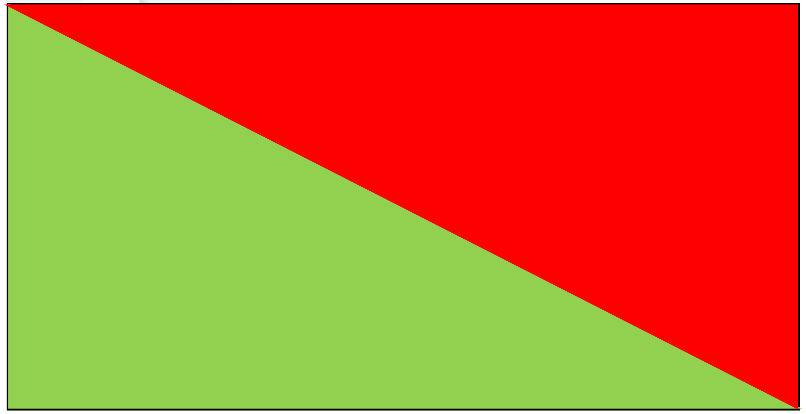


DSL at UCSB

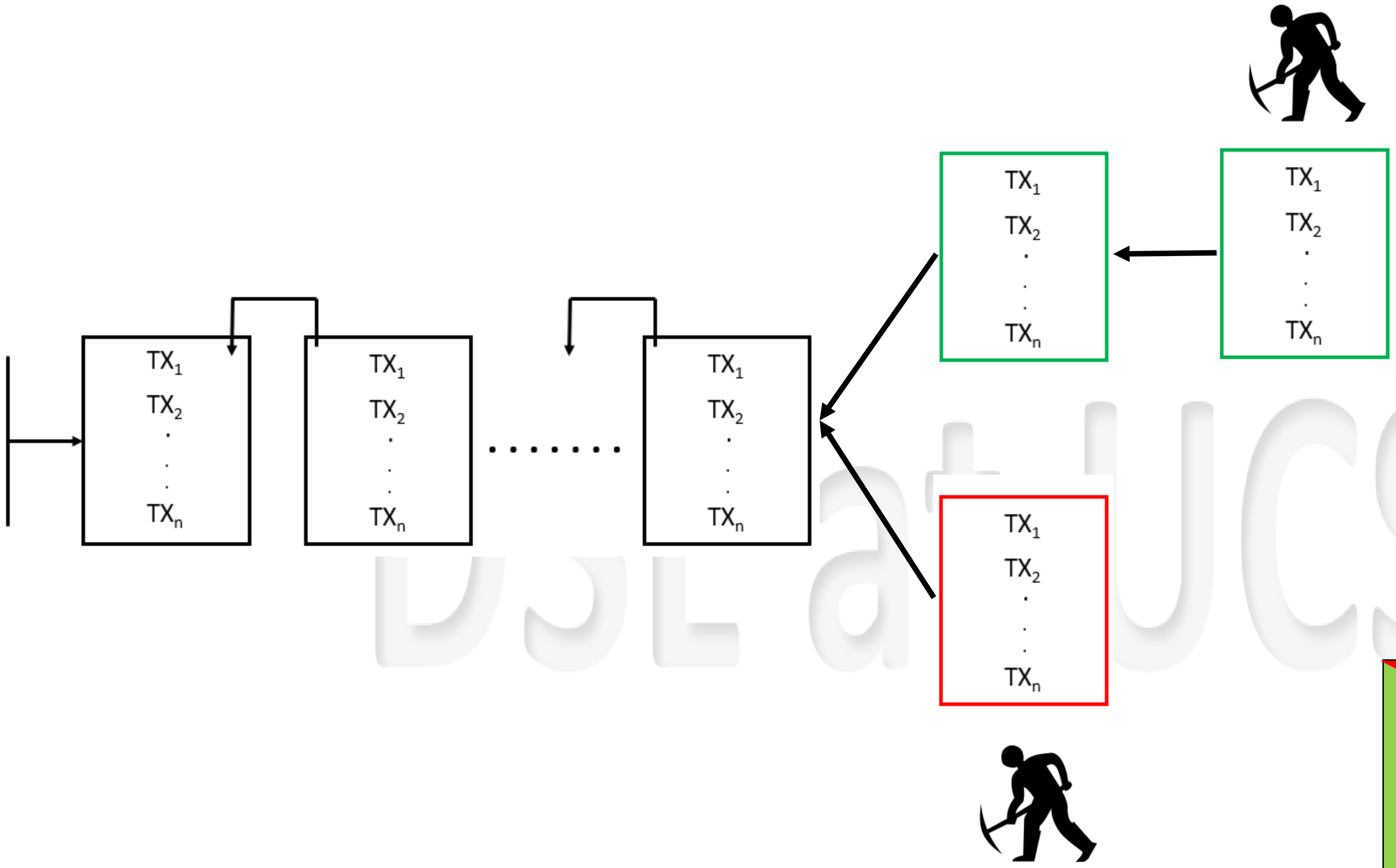
Forks

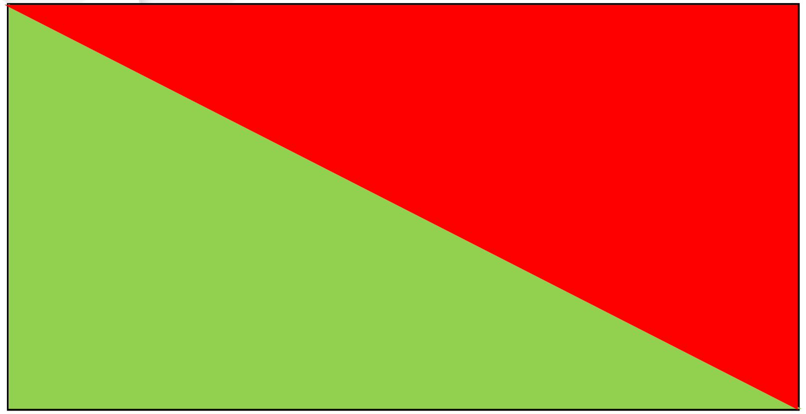


DSL at UCSB



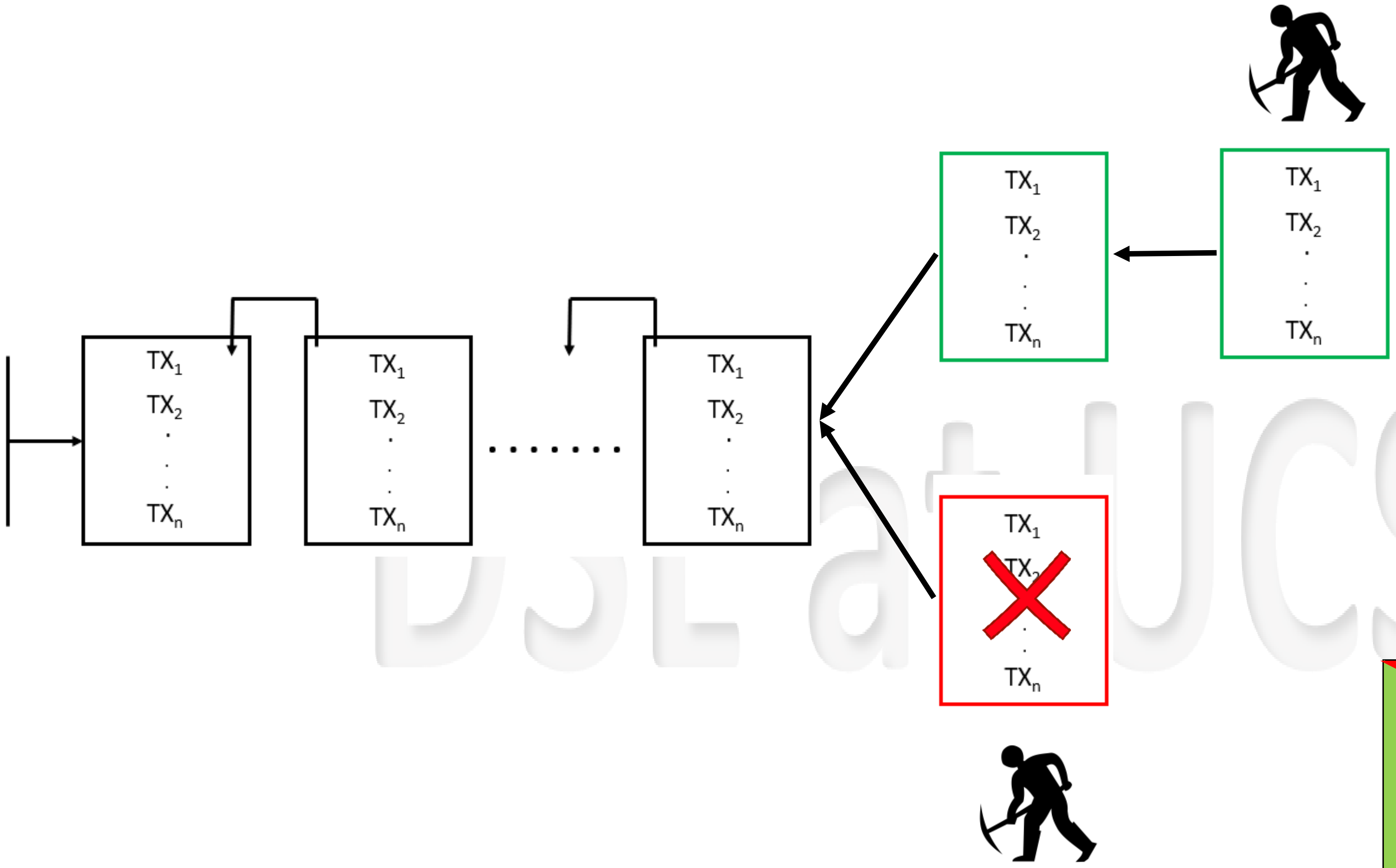
Forks





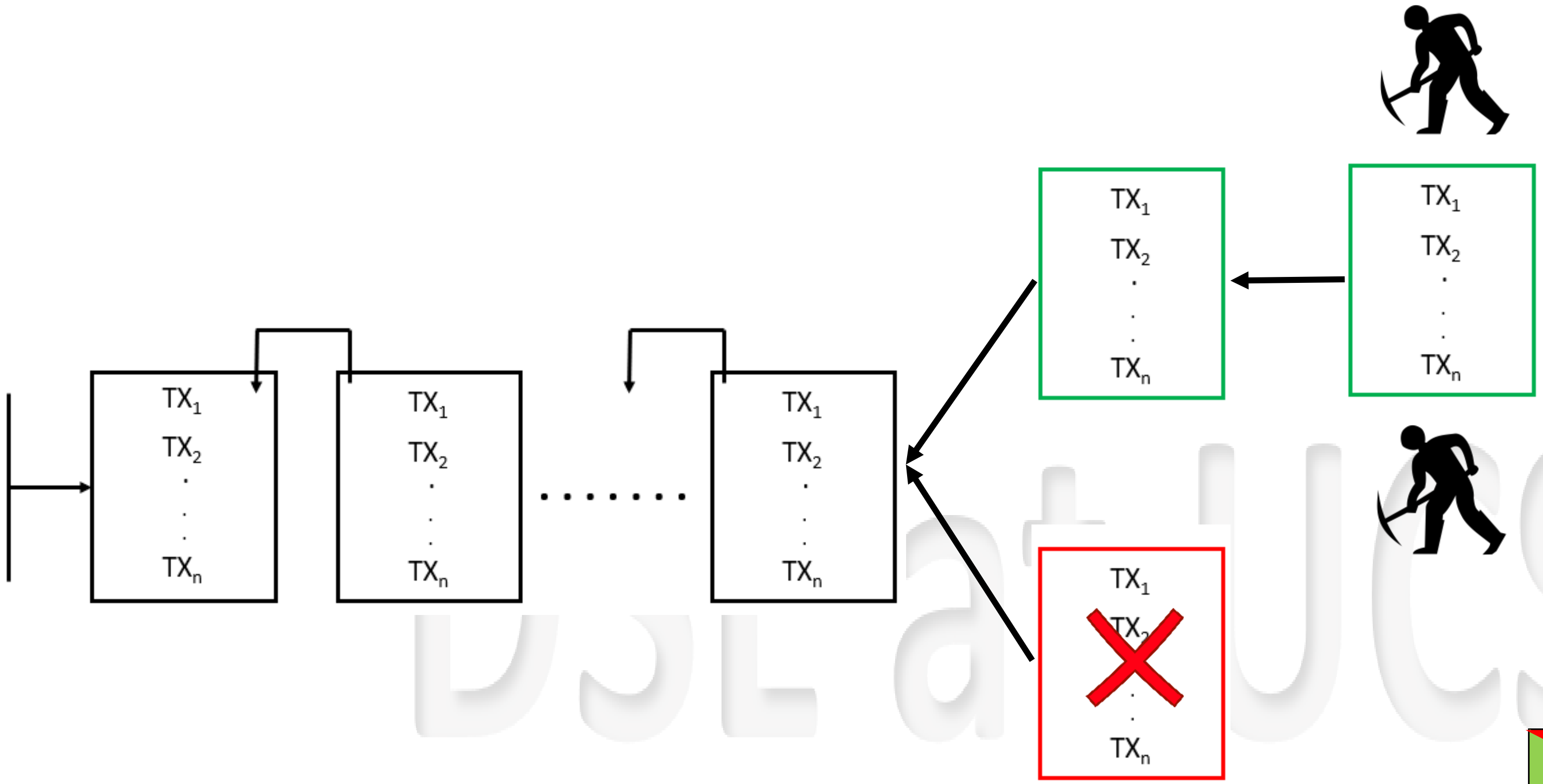
DSL at UCSB

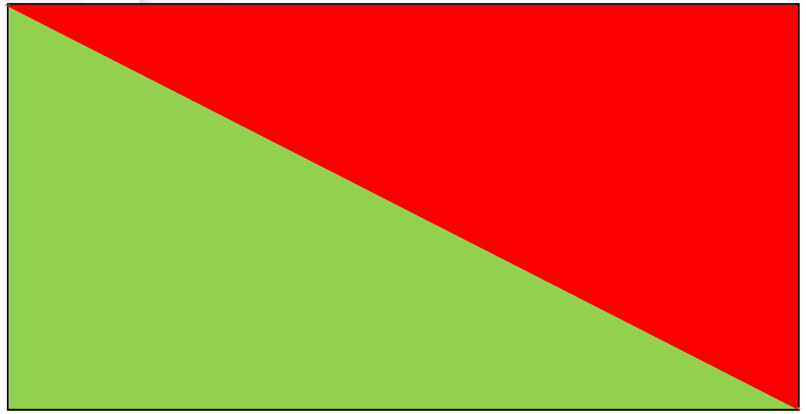
Forks



DSL at UCSB

Forks

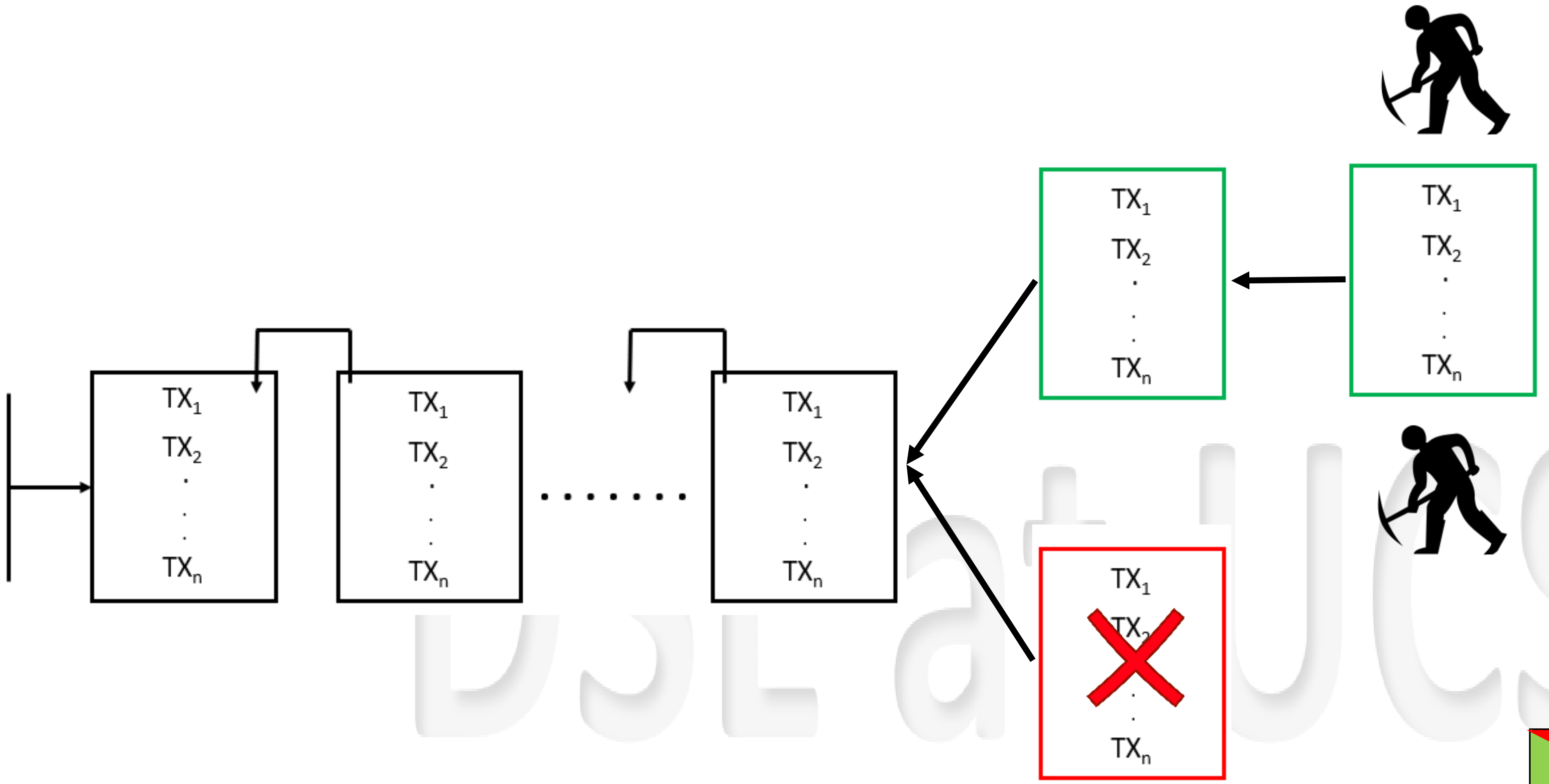




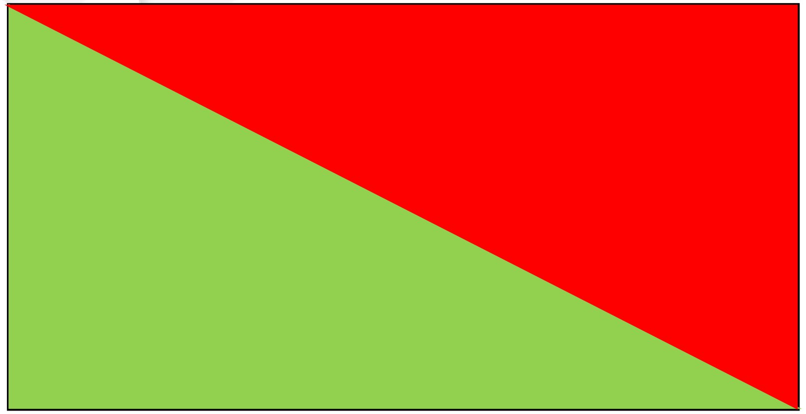
- Miners join the longest chain to resolve forks

DSL at UCSB

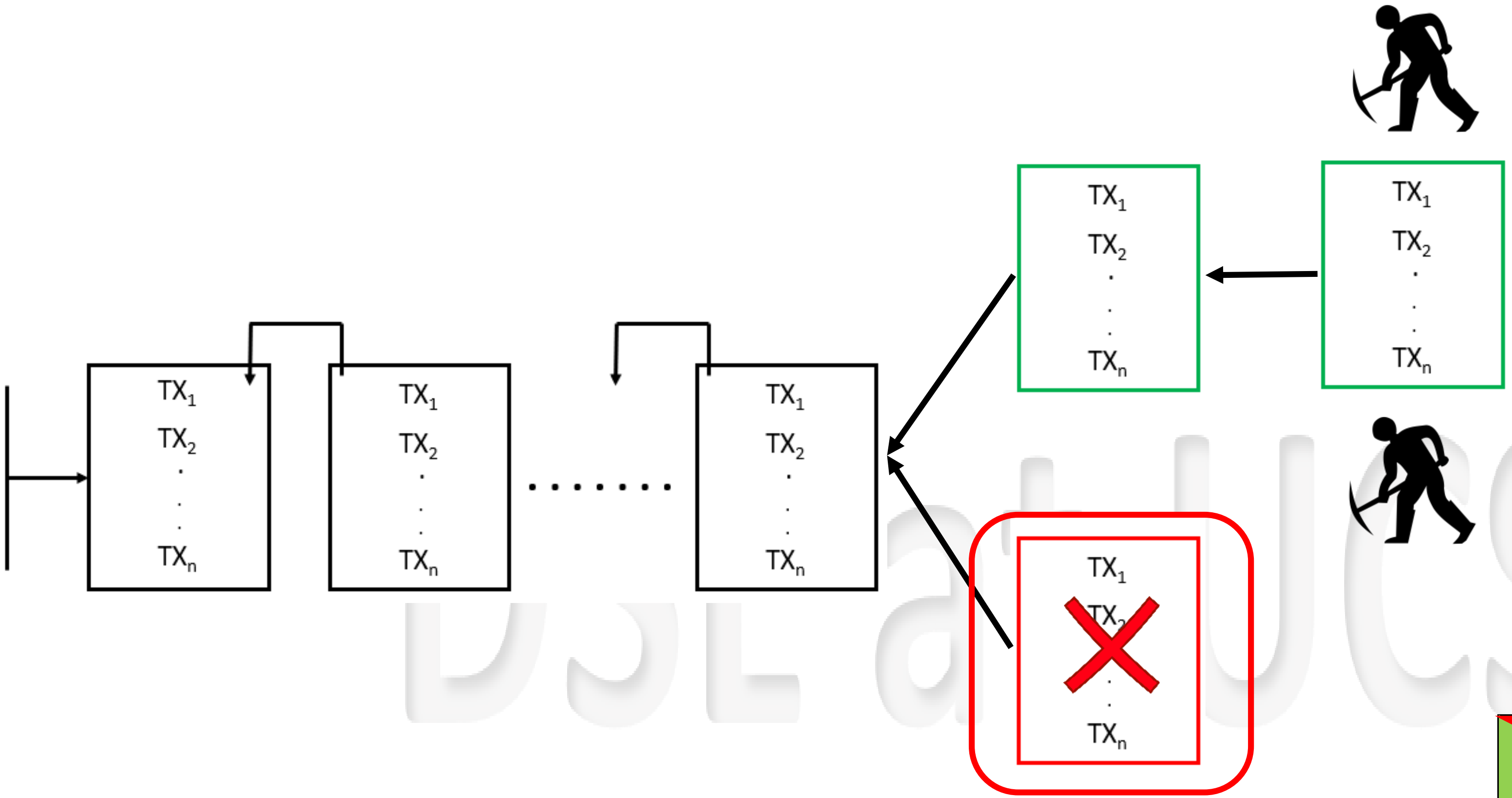
Forks



DSL at UCSB

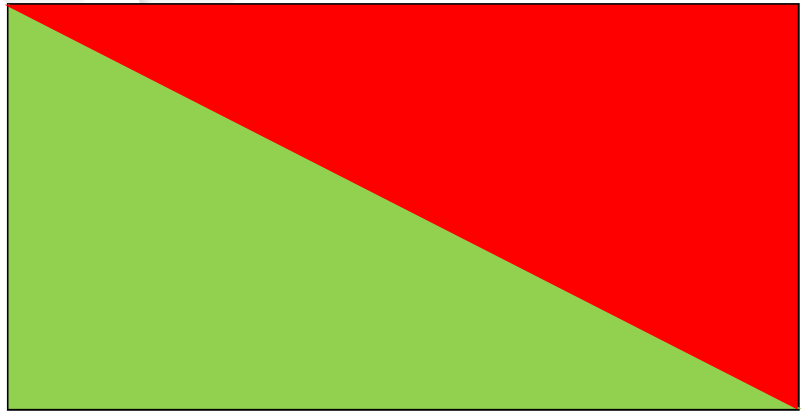


Forks

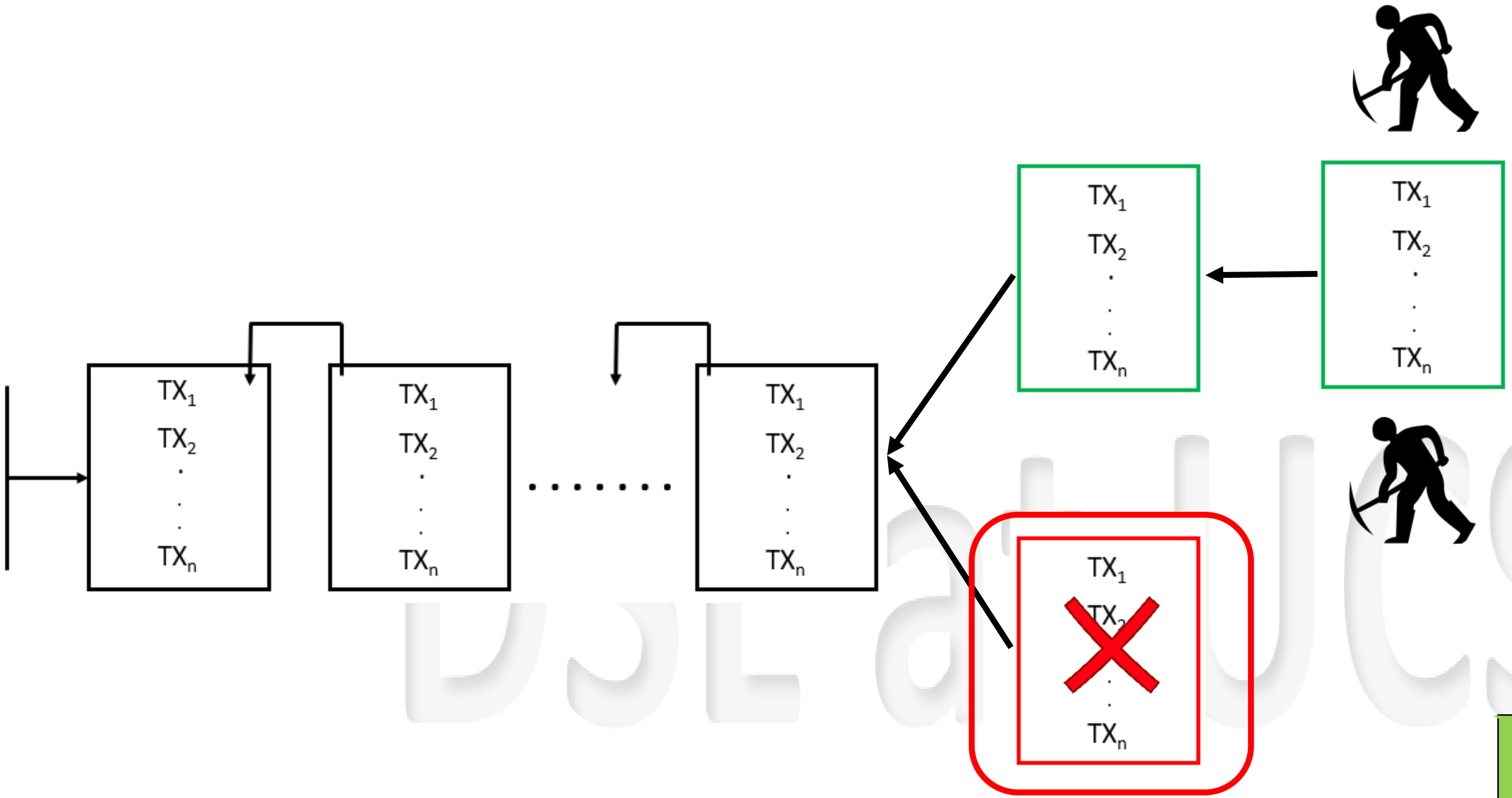


DSL at UCSB

- Transactions in this block have to be resubmitted



Forks



DSL at UCSB

- Transactions in this block have to be resubmitted

51% Attack

- If 51% of the computation (hash) power are malicious:
  - They can cooperate to fork the chain at any block
- Can lead to double spending

DSL at UCSB

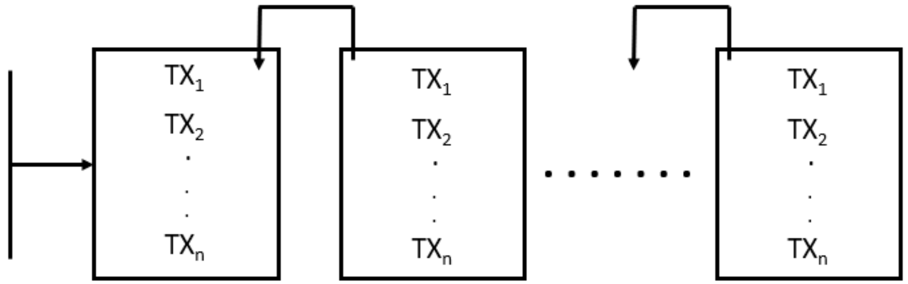
DSL

UCSB

51% Attack

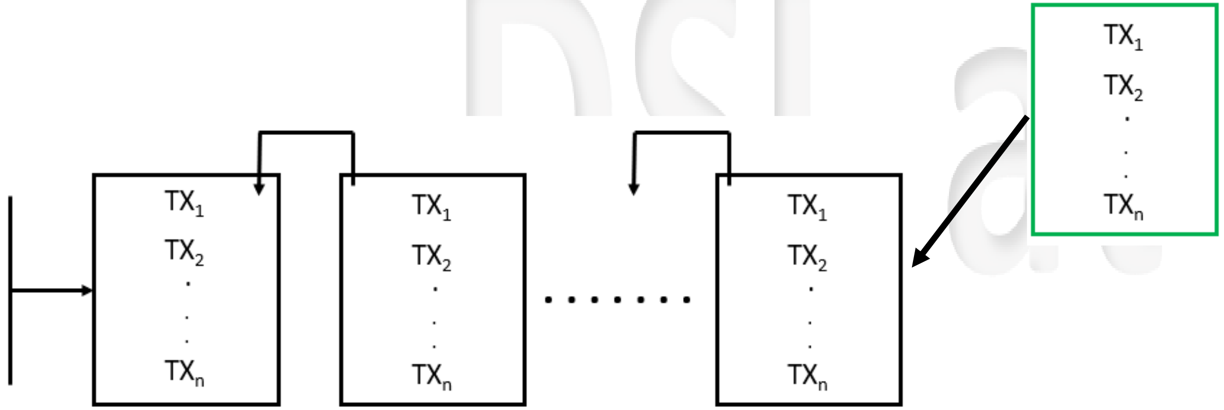
- If 51% of the computation (hash) power are malicious:
  - They can cooperate to fork the chain at any block
- Can lead to double spending

DSL at UCSB



51% Attack

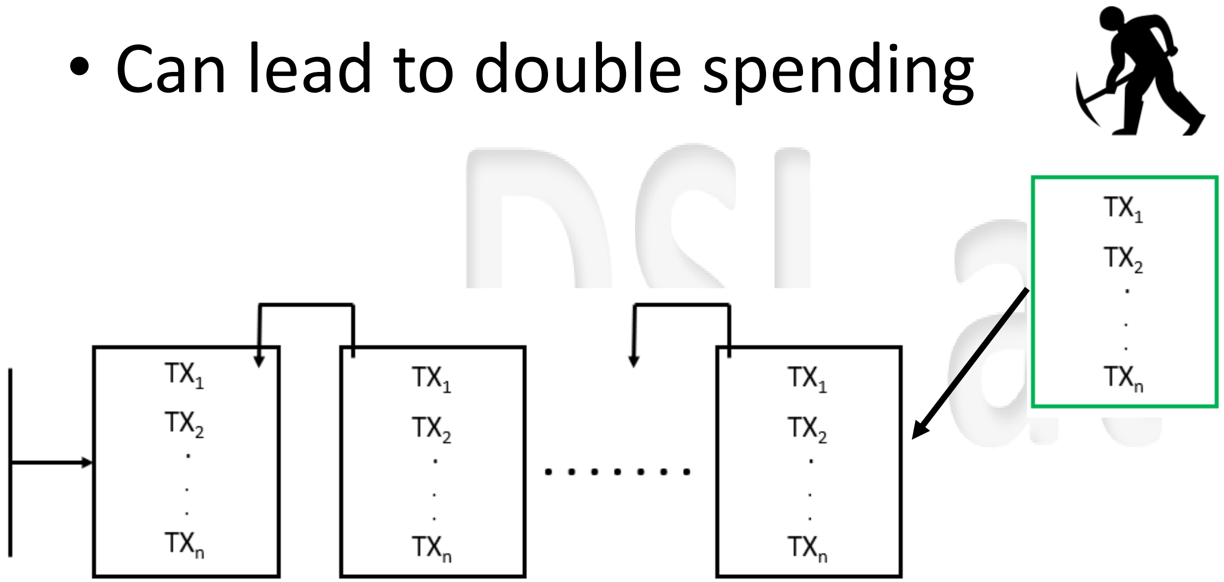
- If 51% of the computation (hash) power are malicious:
  - They can cooperate to fork the chain at any block
- Can lead to double spending



DSL at UCSB

51% Attack

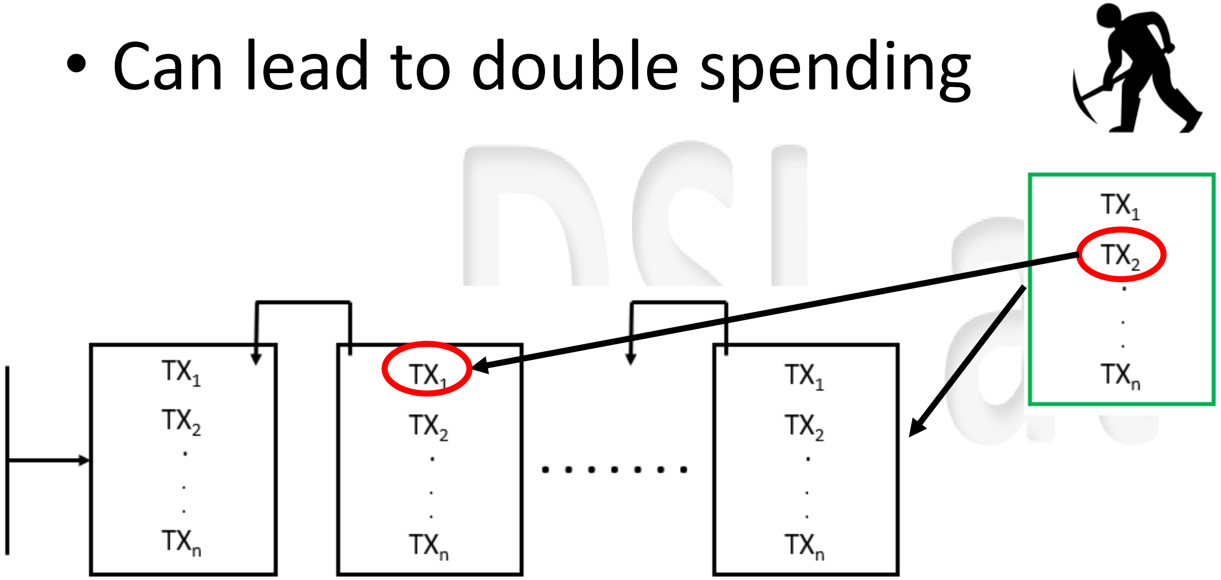
- If 51% of the computation (hash) power are malicious:
  - They can cooperate to fork the chain at any block
- Can lead to double spending



DSL at UCSB

51% Attack

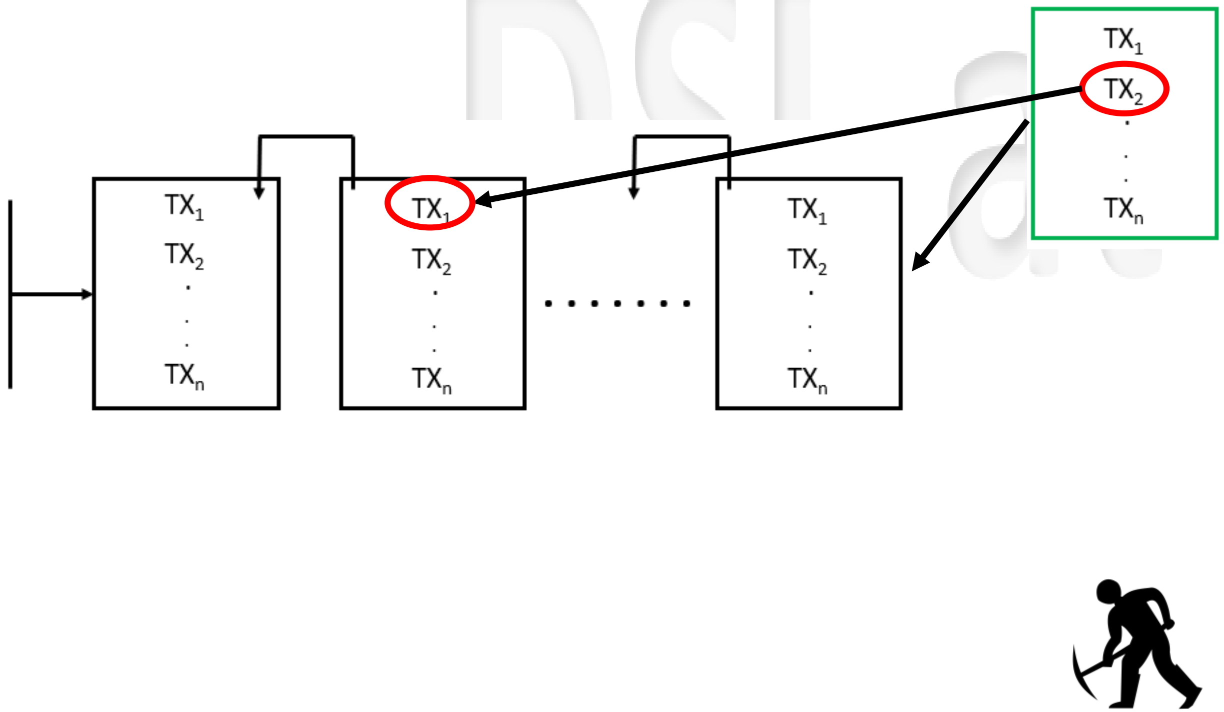
- If 51% of the computation (hash) power are malicious:
  - They can cooperate to fork the chain at any block
- Can lead to double spending



DSL at UCSB

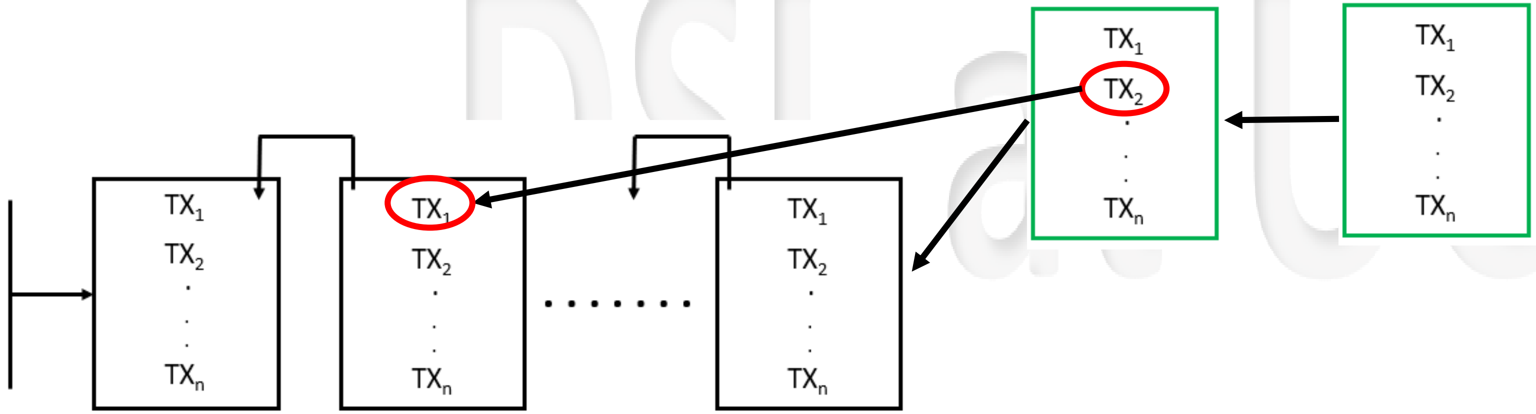
51% Attack

- If 51% of the computation (hash) power are malicious:
  - They can cooperate to fork the chain at any block
- Can lead to double spending



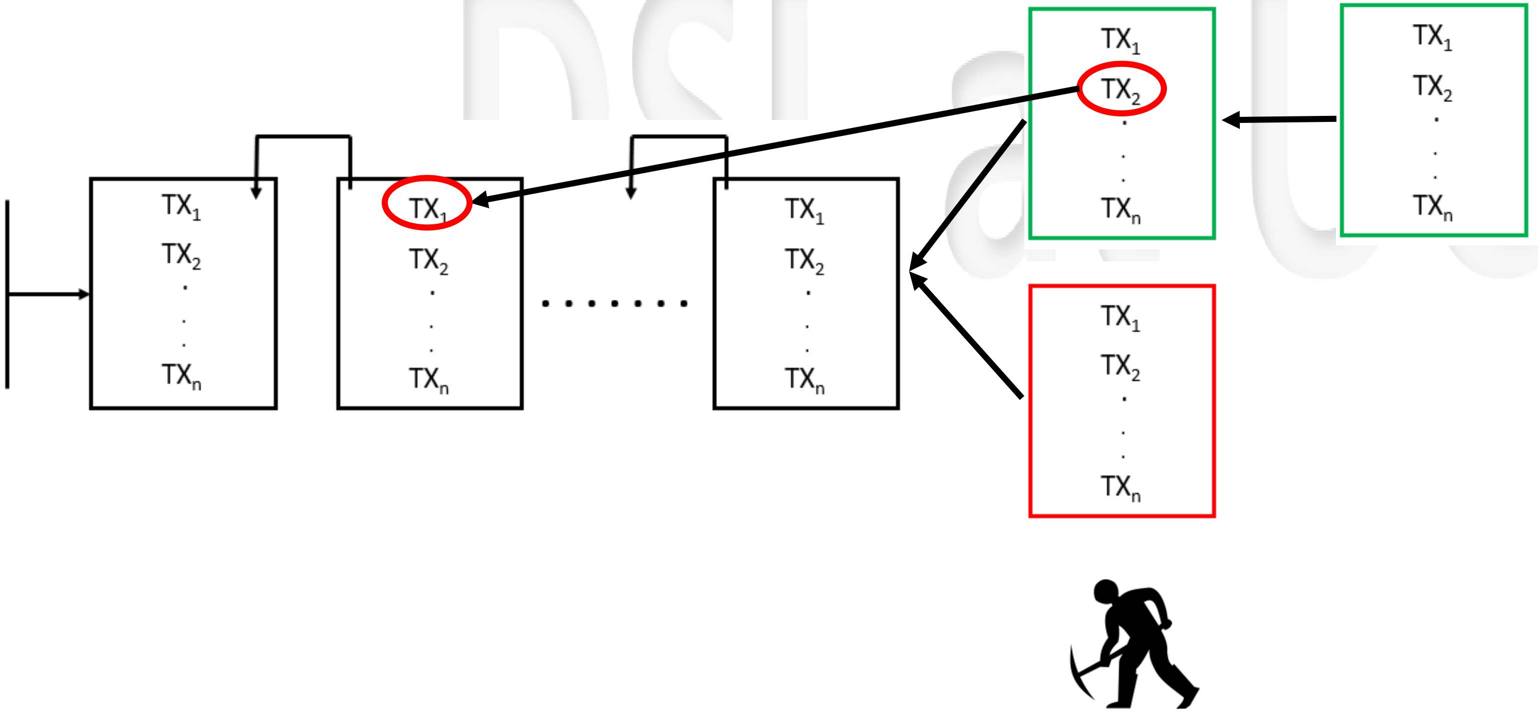
51% Attack

- If 51% of the computation (hash) power are malicious:
  - They can cooperate to fork the chain at any block
- Can lead to double spending



51% Attack

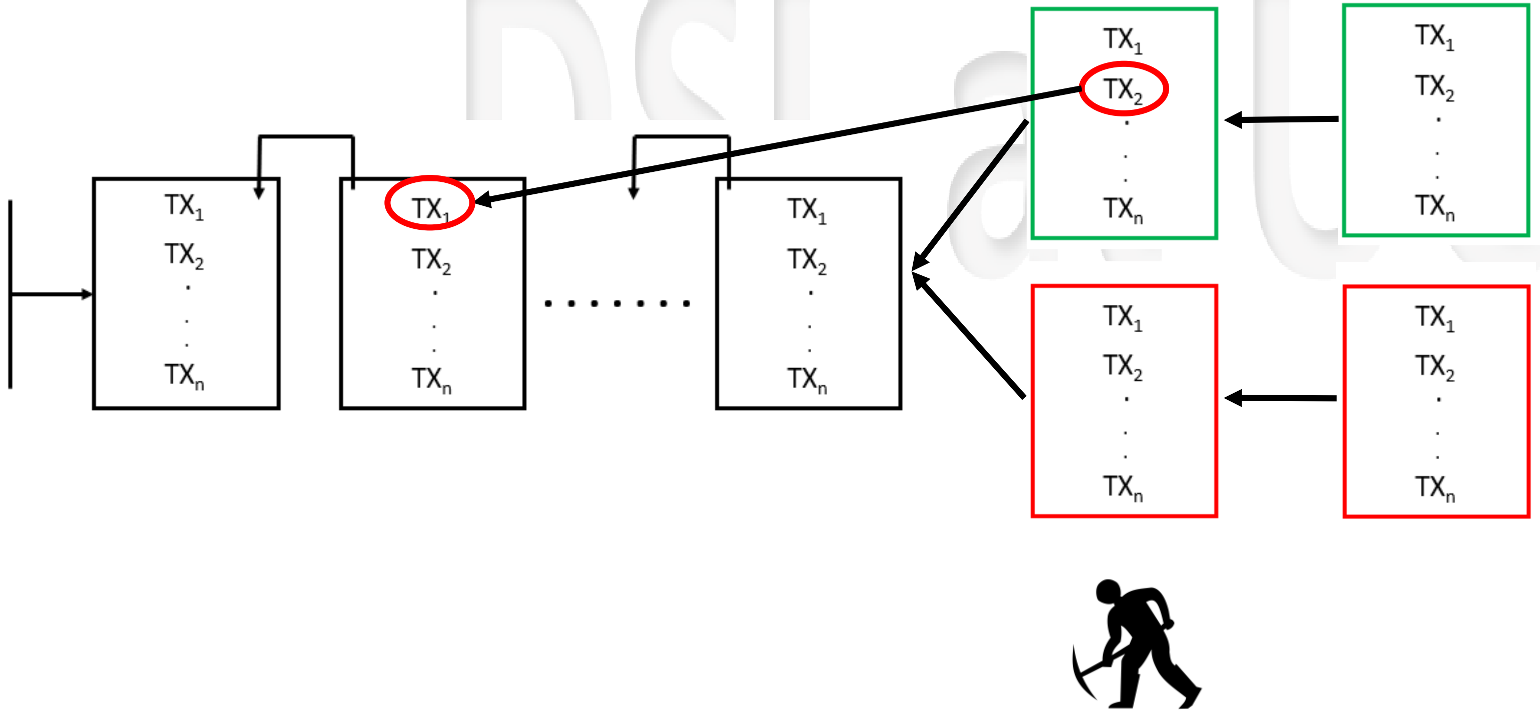
- If 51% of the computation (hash) power are malicious:
  - They can cooperate to fork the chain at any block
- Can lead to double spending



DSL at UCSB

51% Attack

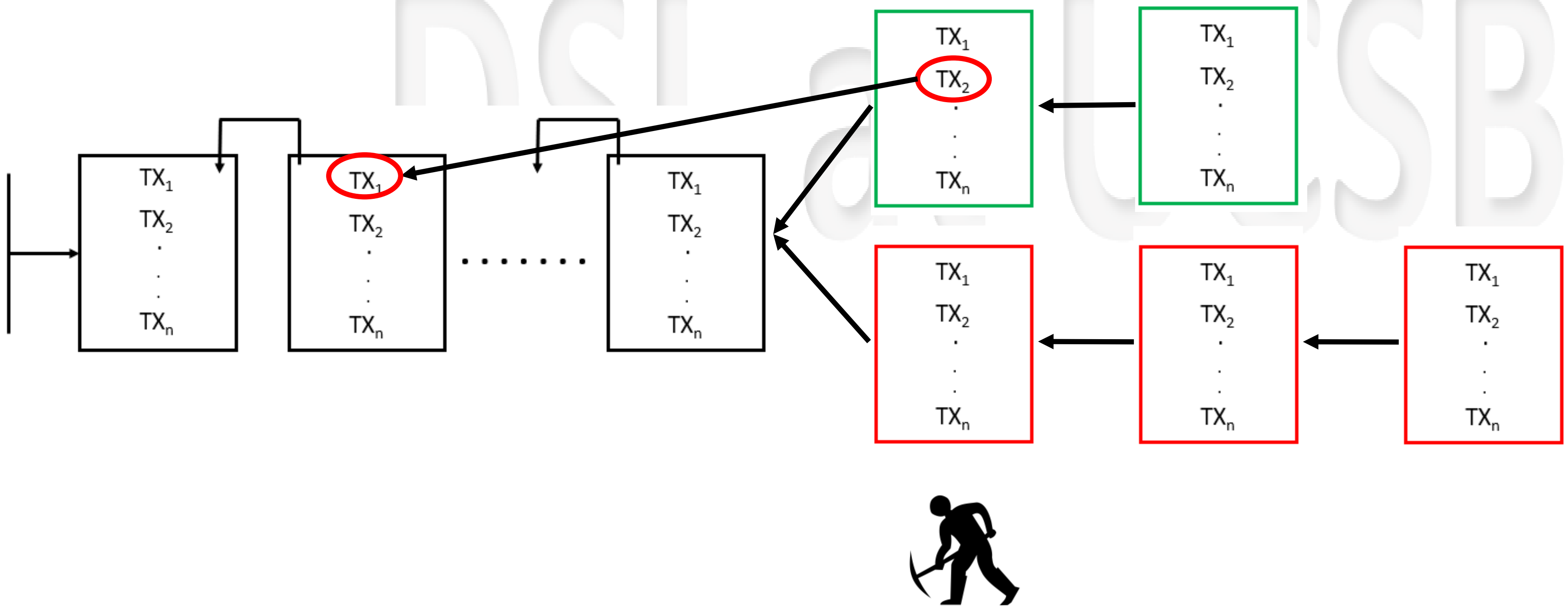
- If 51% of the computation (hash) power are malicious:
  - They can cooperate to fork the chain at any block
- Can lead to double spending



DSL at UCSB

51% Attack

- If 51% of the computation (hash) power are malicious:
  - They can cooperate to fork the chain at any block
- Can lead to double spending

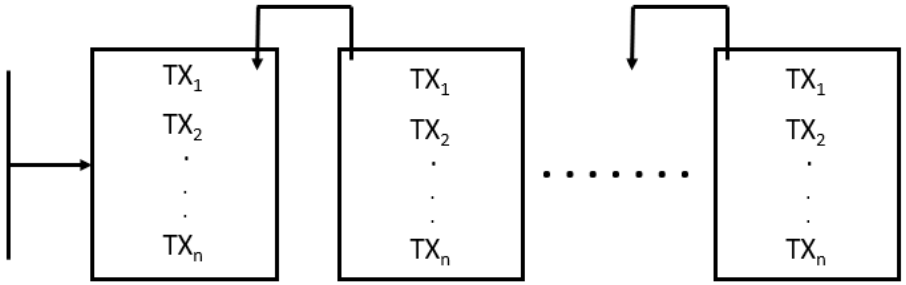


Selfish Mining

DSL at UCSB

Eyal, Ittay, and Emin Gün Sirer. "Majority is not enough: Bitcoin mining is vulnerable." *Communications of the ACM* 61.7 (2018): 95-102.

Selfish Mining

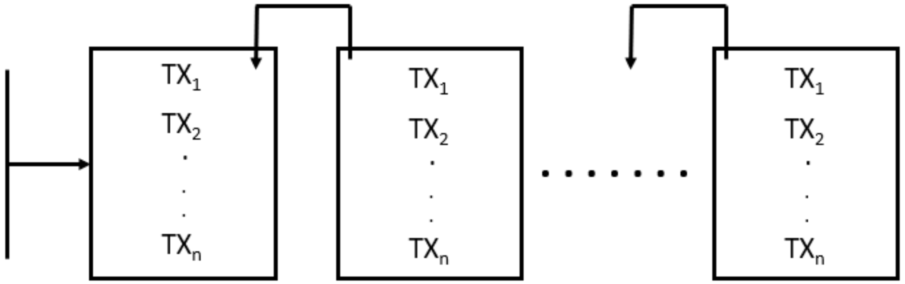


DSL at UCSB

Eyal, Ittay, and Emin Gün Sirer. "Majority is not enough: Bitcoin mining is vulnerable." *Communications of the ACM* 61.7 (2018): 95-102.

Selfish Mining

Honest Miner

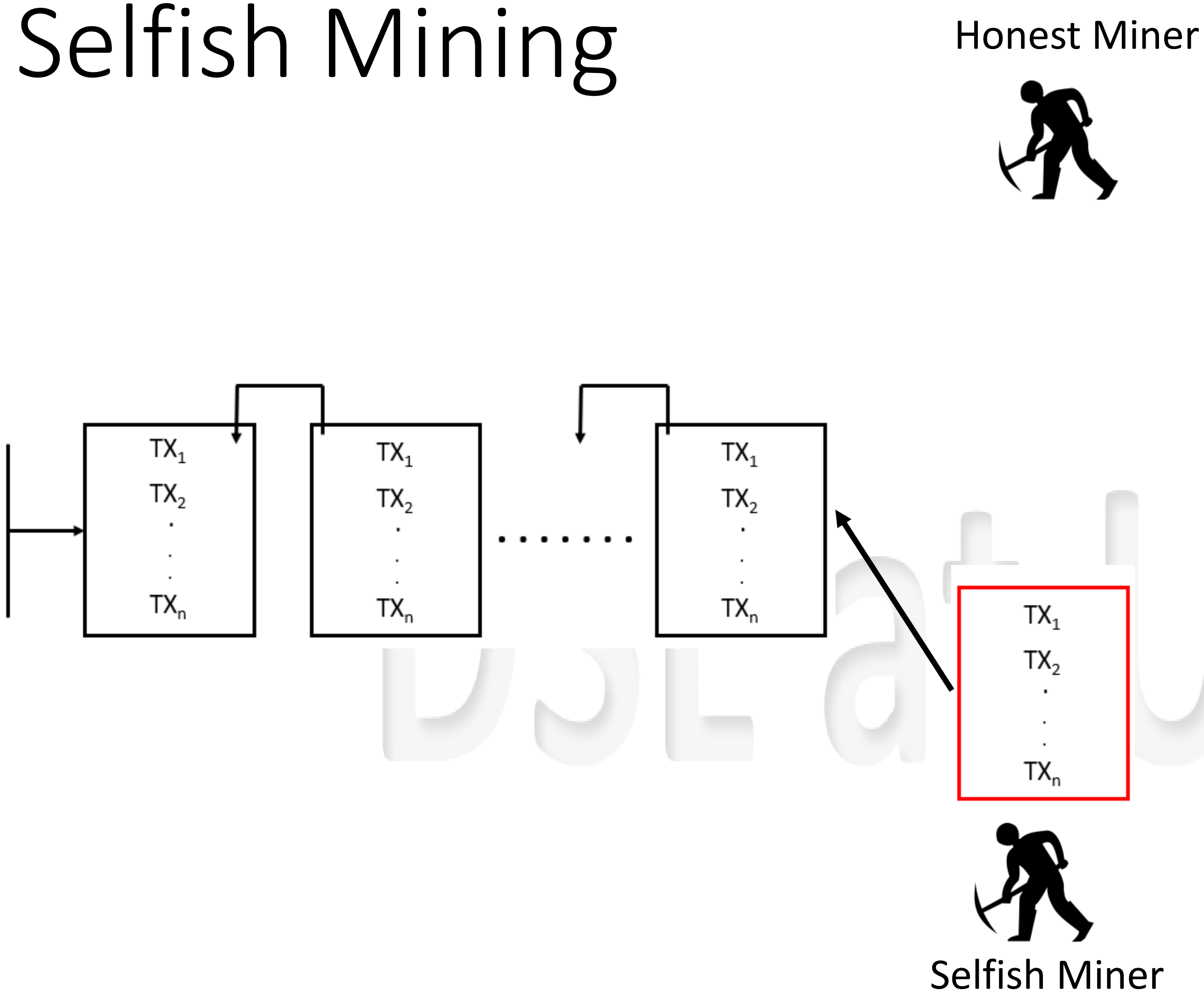


DSL at UCSB

Selfish Miner

Eyal, Ittay, and Emin Gün Sirer. "Majority is not enough: Bitcoin mining is vulnerable." *Communications of the ACM* 61.7 (2018): 95-102.

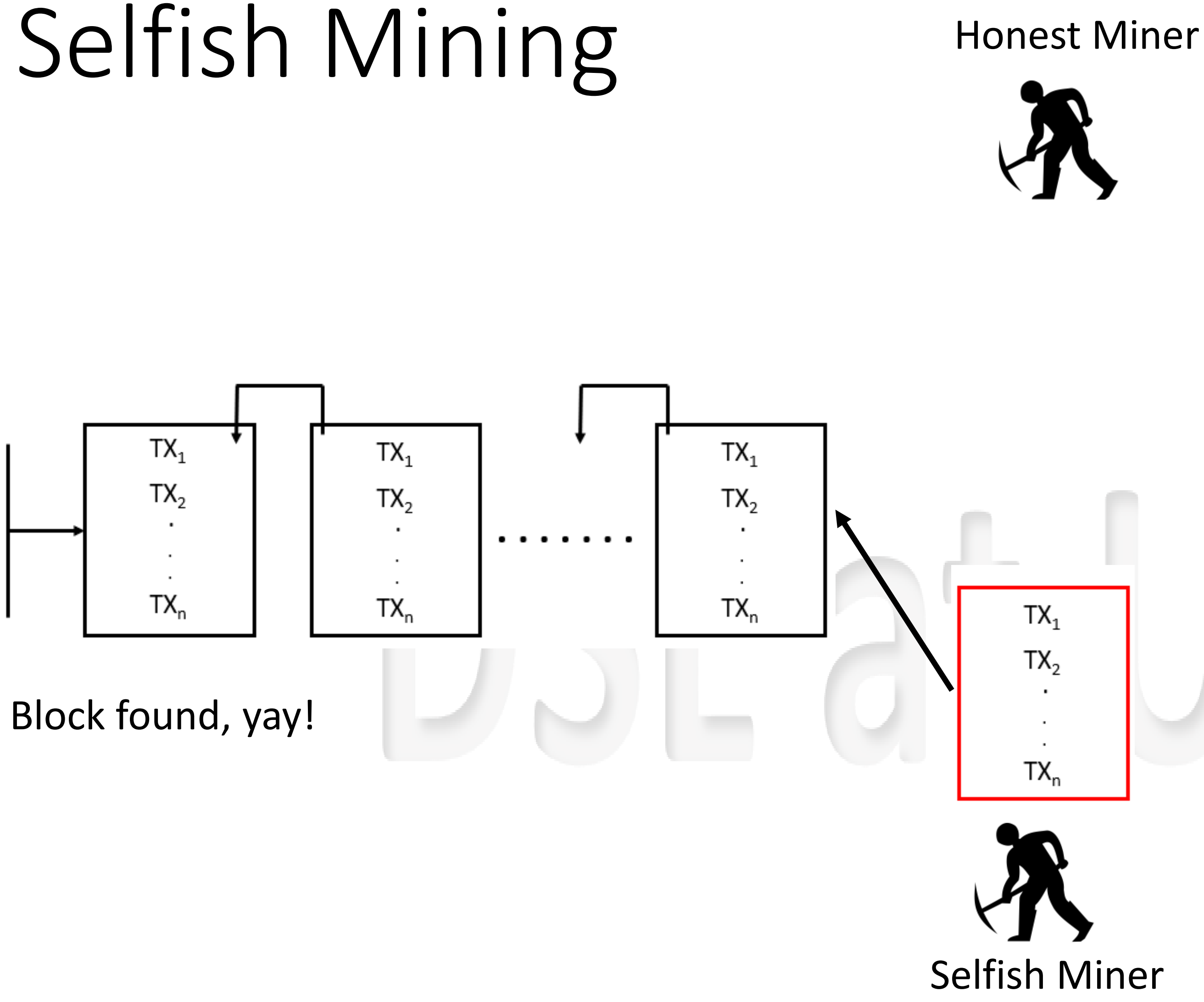
Selfish Mining



DSL at UCSB

Eyal, Ittay, and Emin Gün Sirer. "Majority is not enough: Bitcoin mining is vulnerable." *Communications of the ACM* 61.7 (2018): 95-102.

Selfish Mining

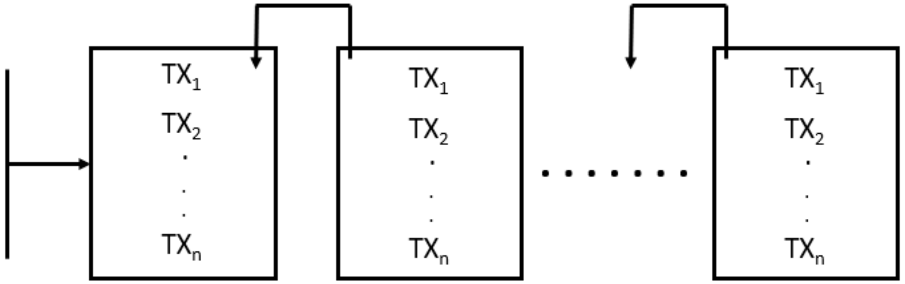


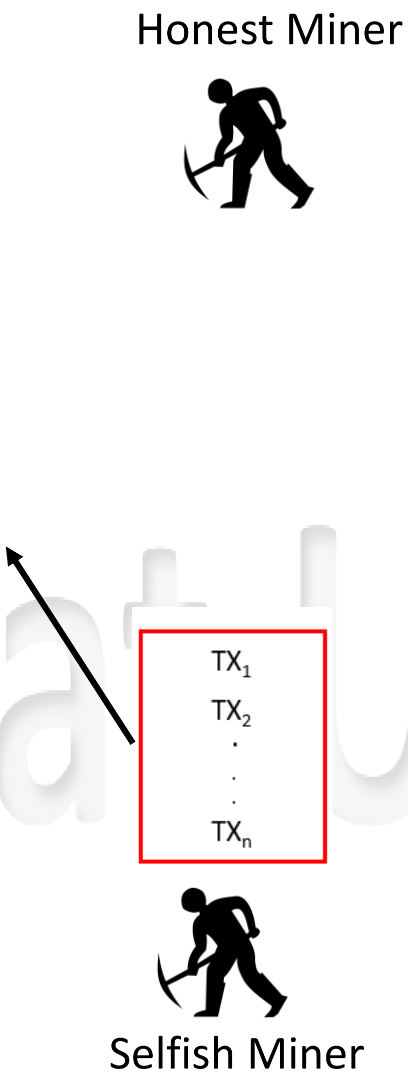
- Block found, yay!

DSL at UCSB

Eyal, Ittay, and Emin Gün Sirer. "Majority is not enough: Bitcoin mining is vulnerable." *Communications of the ACM* 61.7 (2018): 95-102.

Selfish Mining



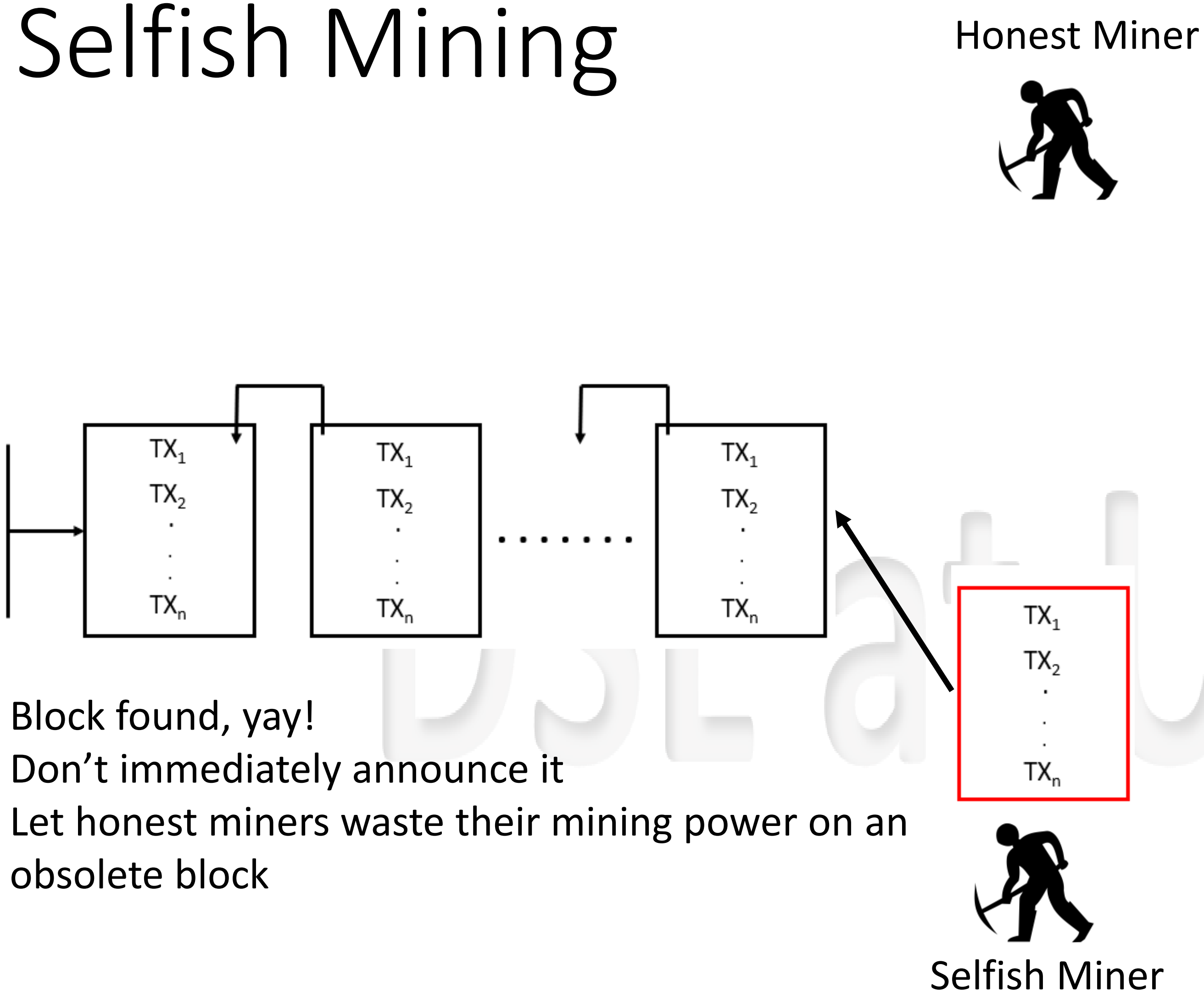


- Block found, yay!
- Don’t immediately announce it

DSL at UCSB

Eyal, Ittay, and Emin Gün Sirer. "Majority is not enough: Bitcoin mining is vulnerable." *Communications of the ACM* 61.7 (2018): 95-102.

Selfish Mining

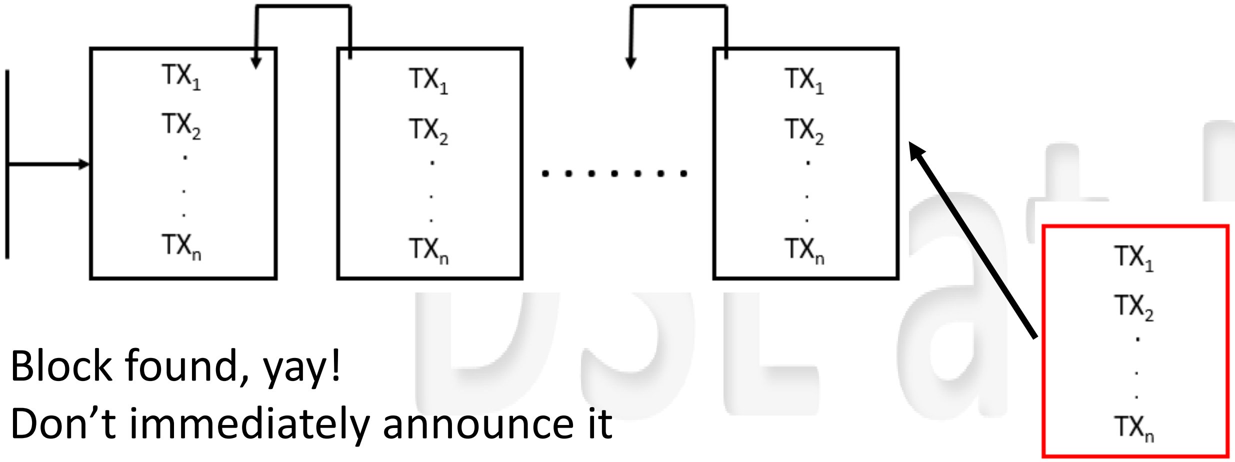


- Block found, yay!
- Don't immediately announce it
- Let honest miners waste their mining power on an obsolete block

DSL at UCSB

Eyal, Ittay, and Emin Gün Sirer. "Majority is not enough: Bitcoin mining is vulnerable." *Communications of the ACM* 61.7 (2018): 95-102.

Selfish Mining

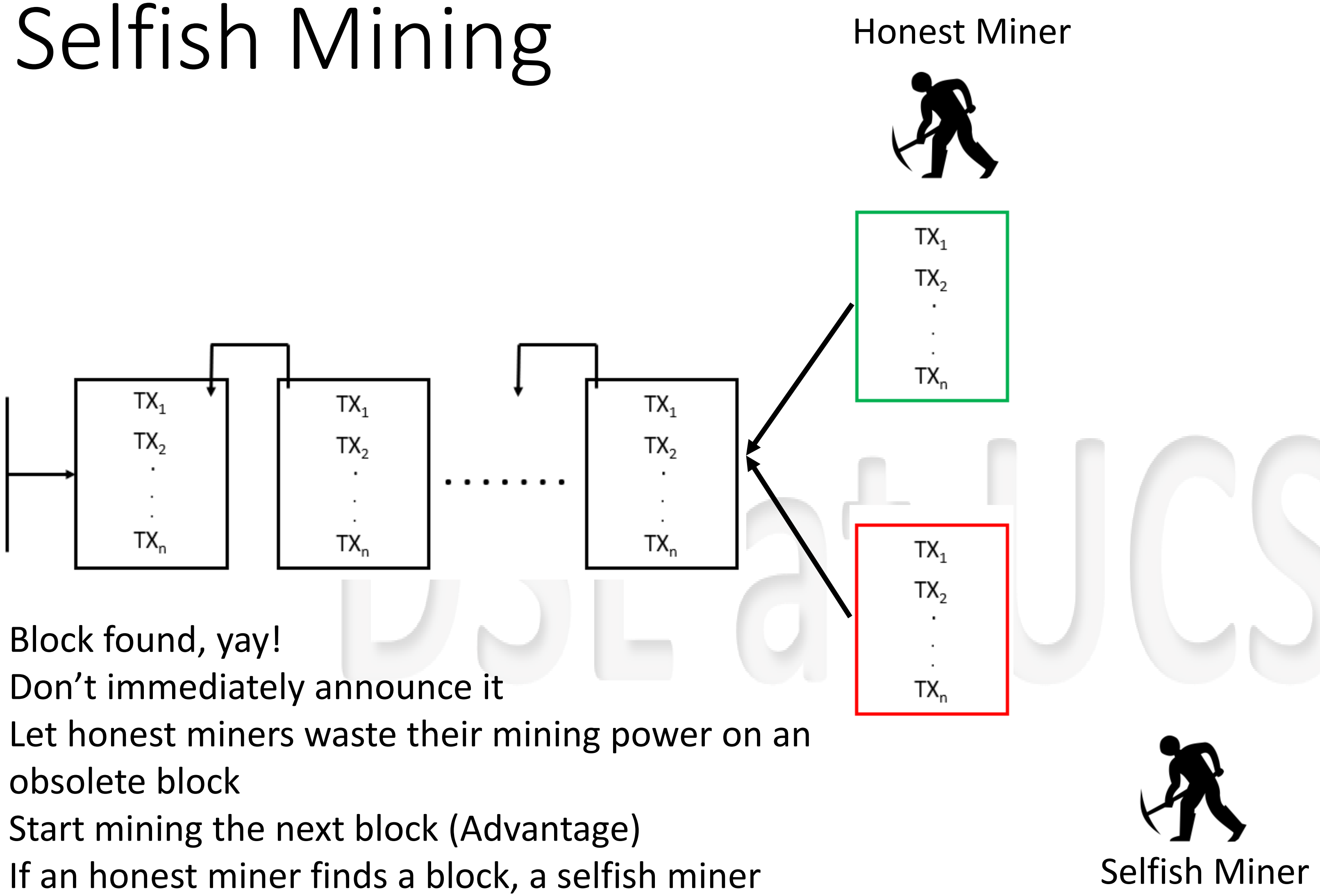


- Block found, yay!
- Don’t immediately announce it
- Let honest miners waste their mining power on an obsolete block
- Start mining the next block (Advantage)

DSL at UCSB

Eyal, Ittay, and Emin Gün Sirer. "Majority is not enough: Bitcoin mining is vulnerable." *Communications of the ACM* 61.7 (2018): 95-102.

Selfish Mining

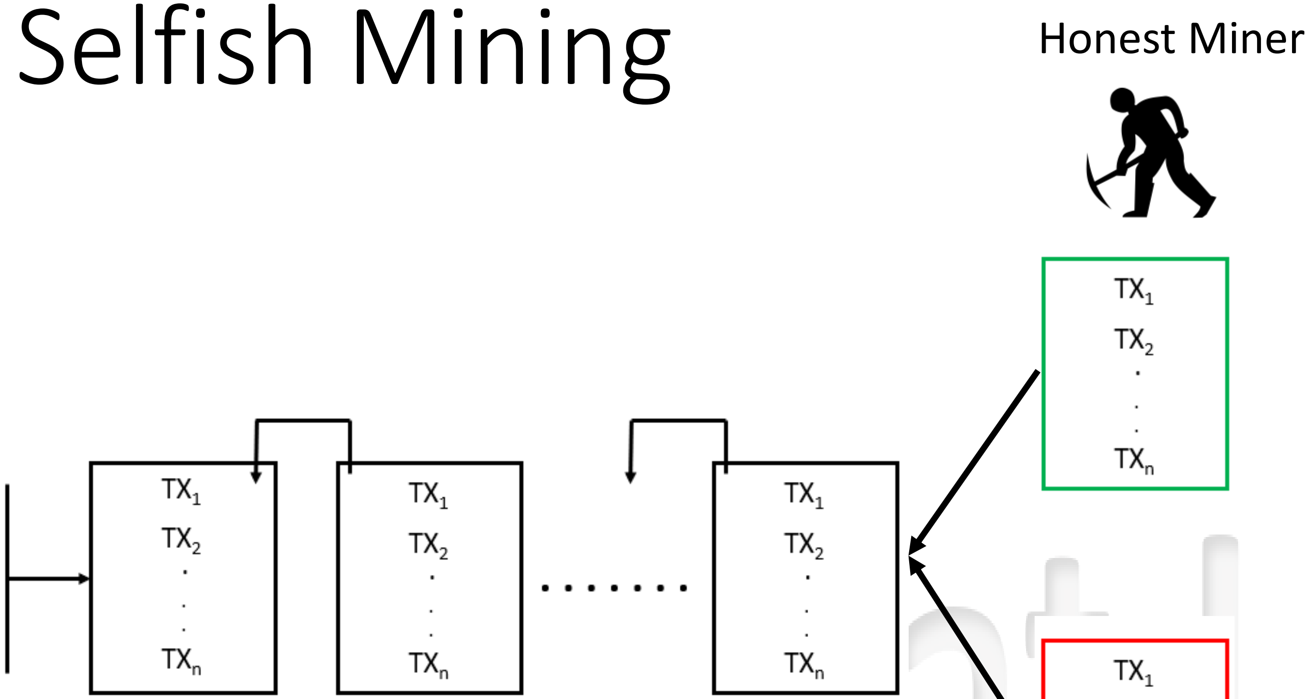


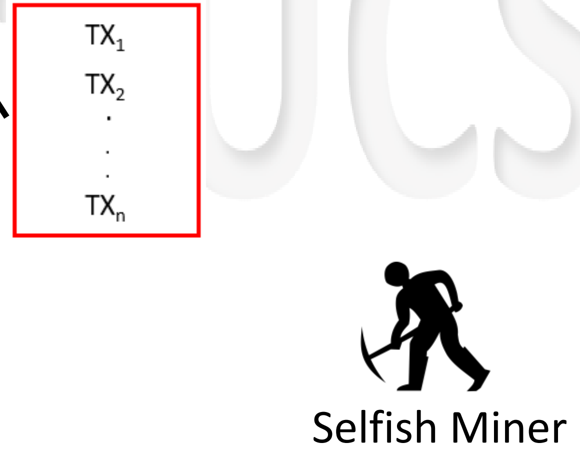
- Block found, yay!
- Don’t immediately announce it
- Let honest miners waste their mining power on an obsolete block
- Start mining the next block (Advantage)
- If an honest miner finds a block, a selfish miner immediately announces their found block

DSL at UCSB

Eyal, Ittay, and Emin Gün Sirer. "Majority is not enough: Bitcoin mining is vulnerable." *Communications of the ACM* 61.7 (2018): 95-102.

Selfish Mining



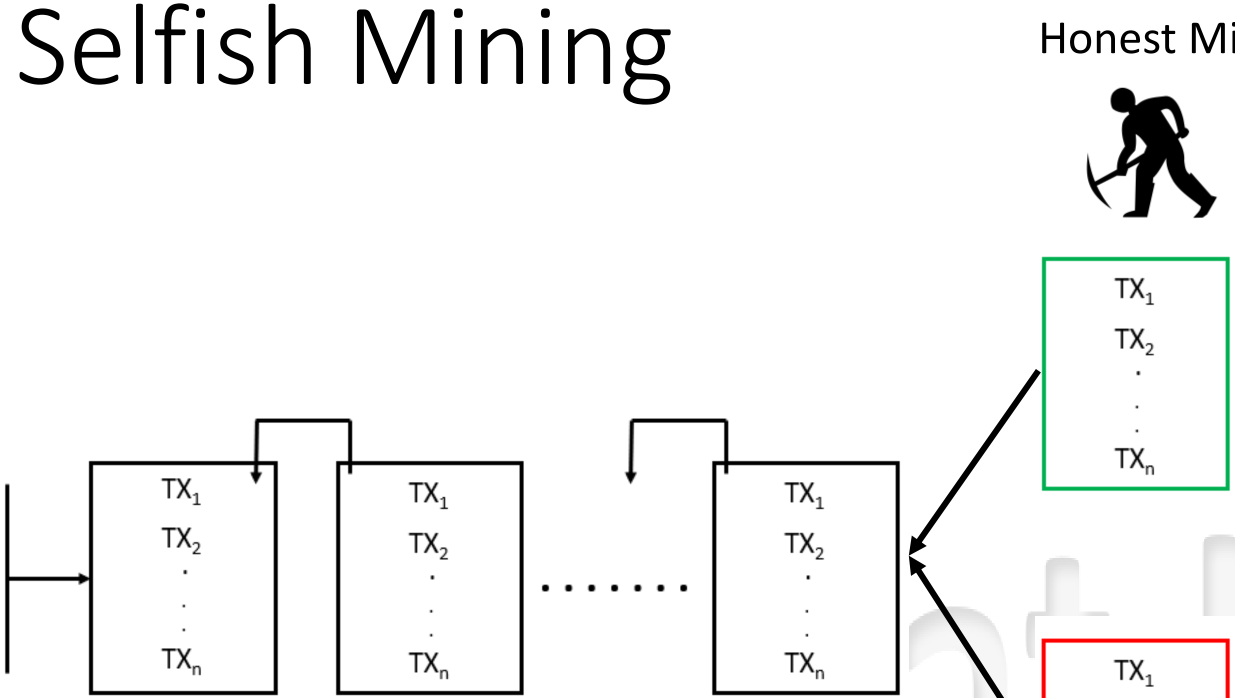


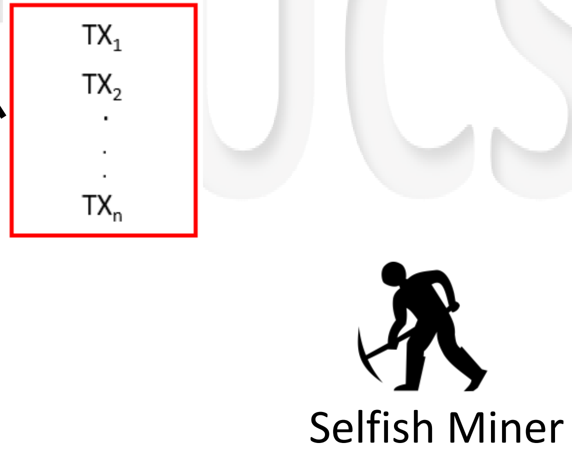
- Block found, yay!
- Don't immediately announce it
- Let honest miners waste their mining power on an obsolete block
- Start mining the next block (Advantage)
- If an honest miner finds a block, a selfish miner immediately announces their found block
- This splits the power of honest miners

Eyal, Ittay, and Emin Gün Sirer. "Majority is not enough: Bitcoin mining is vulnerable." *Communications of the ACM* 61.7 (2018): 95-102.

DSL at UCSB

Selfish Mining



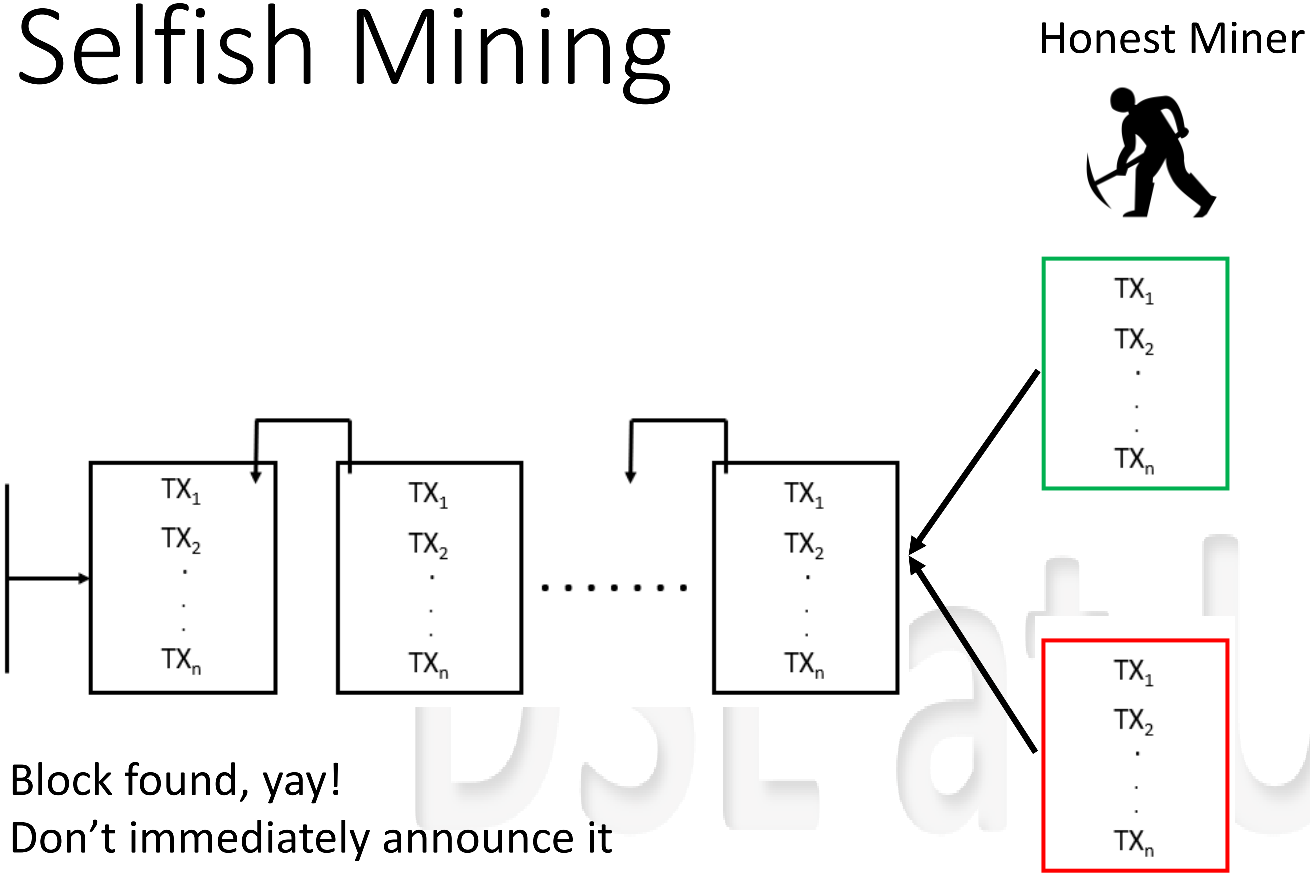


- Block found, yay!
- Don’t immediately announce it
- Let honest miners waste their mining power on an obsolete block
- Start mining the next block (Advantage)
- If an honest miner finds a block, a selfish miner immediately announces their found block
- This splits the power of honest miners

DSL at UCSB

Eyal, Ittay, and Emin Gün Sirer. "Majority is not enough: Bitcoin mining is vulnerable." *Communications of the ACM* 61.7 (2018): 95-102.

Selfish Mining

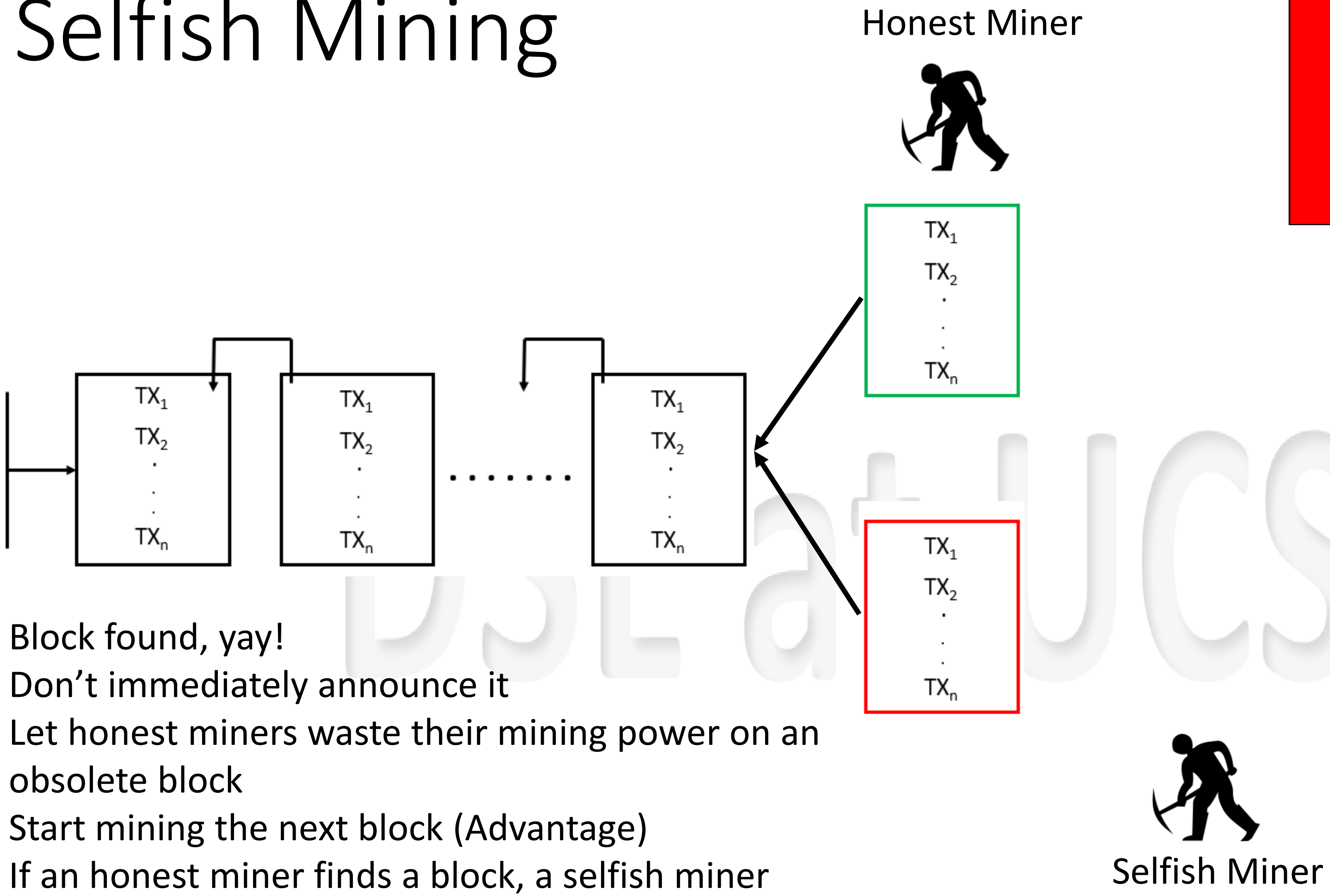


- Block found, yay!
- Don’t immediately announce it
- Let honest miners waste their mining power on an obsolete block
- Start mining the next block (Advantage)
- If an honest miner finds a block, a selfish miner immediately announces their found block
- This splits the power of honest miners

DSL at UCSB

Eyal, Ittay, and Emin Gün Sirer. "Majority is not enough: Bitcoin mining is vulnerable." *Communications of the ACM* 61.7 (2018): 95-102.

Selfish Mining

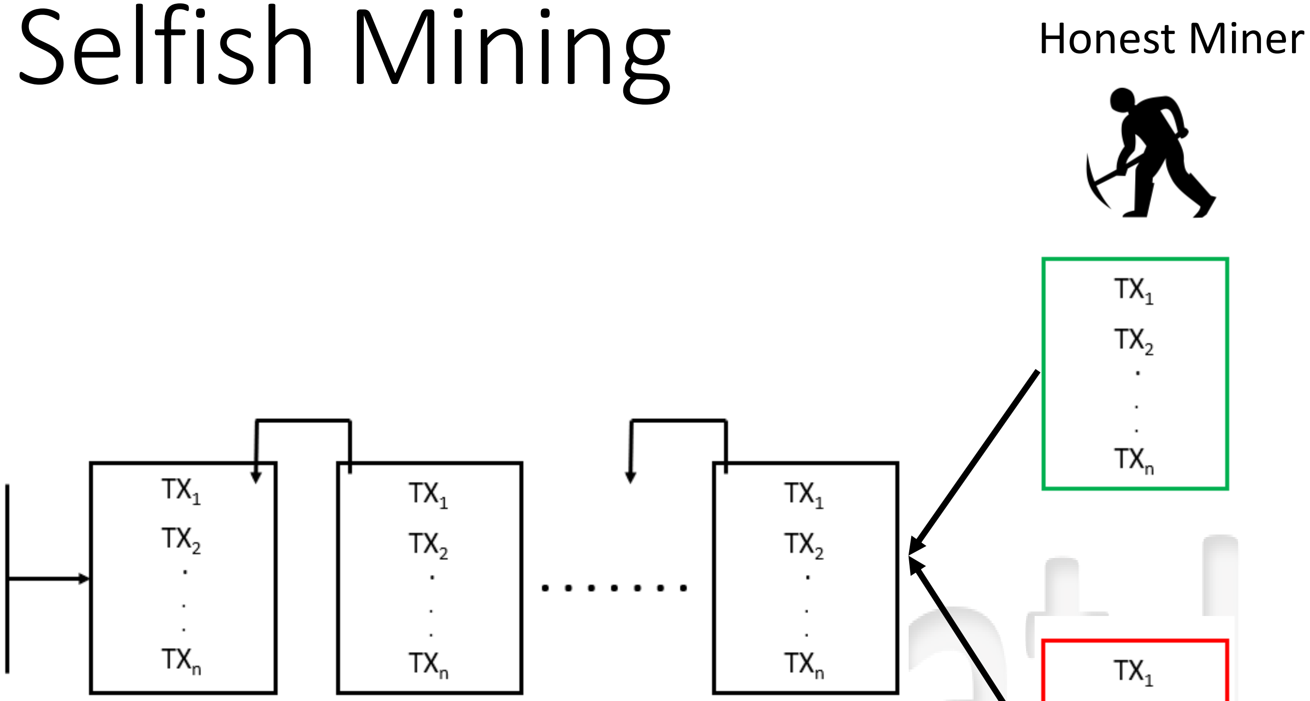


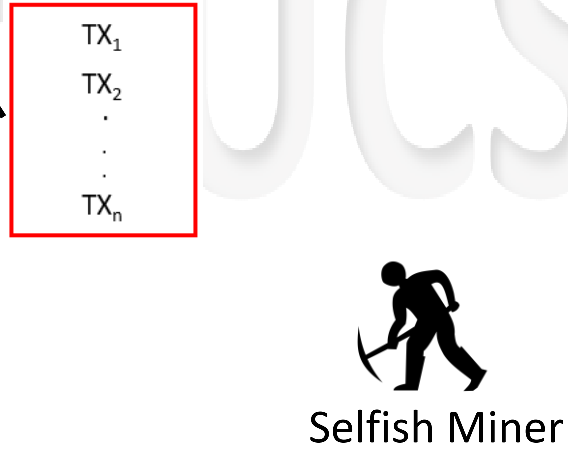
- Block found, yay!
- Don’t immediately announce it
- Let honest miners waste their mining power on an obsolete block
- Start mining the next block (Advantage)
- If an honest miner finds a block, a selfish miner immediately announces their found block
- This splits the power of honest miners

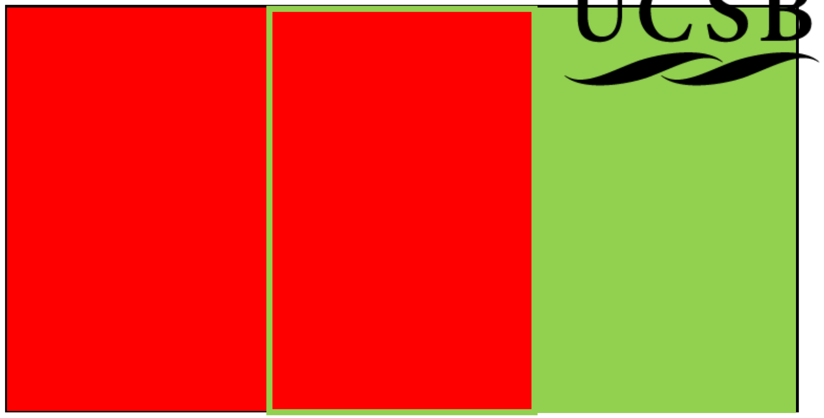
DSL at UCSB

Eyal, Ittay, and Emin Gün Sirer. "Majority is not enough: Bitcoin mining is vulnerable." *Communications of the ACM* 61.7 (2018): 95-102.

Selfish Mining





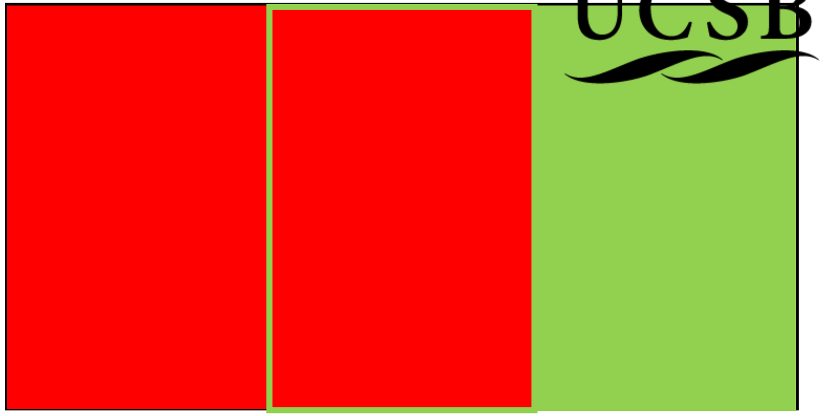


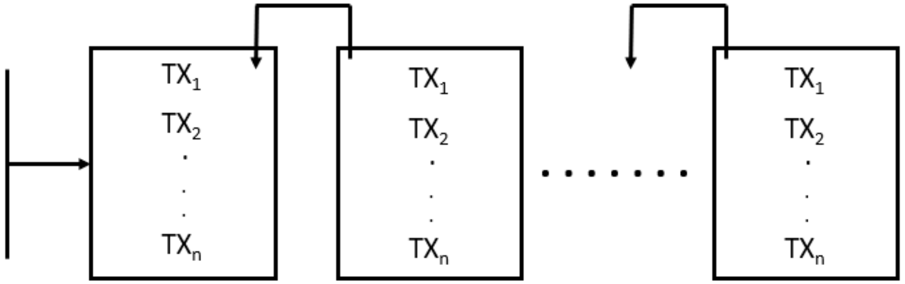
- Block found, yay!
- Don’t immediately announce it
- Let honest miners waste their mining power on an obsolete block
- Start mining the next block (Advantage)
- If an honest miner finds a block, a selfish miner immediately announces their found block
- This splits the power of honest miners

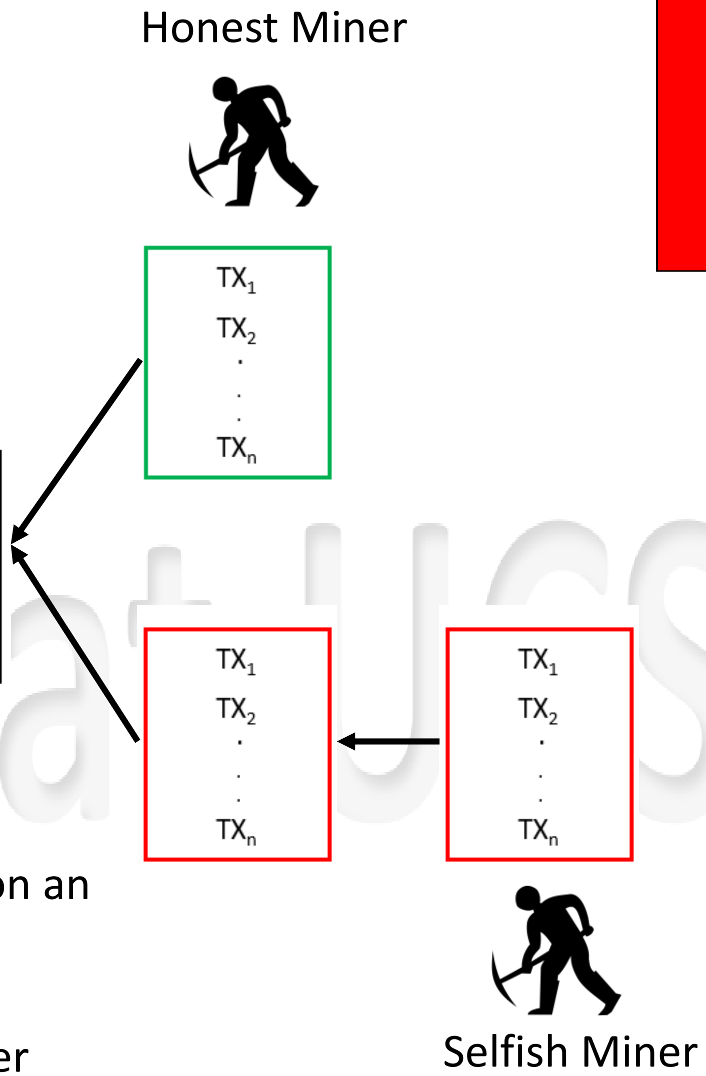
Eyal, Ittay, and Emin Gün Sirer. "Majority is not enough: Bitcoin mining is vulnerable." *Communications of the ACM* 61.7 (2018): 95-102.

DSL at UCSB

Selfish Mining





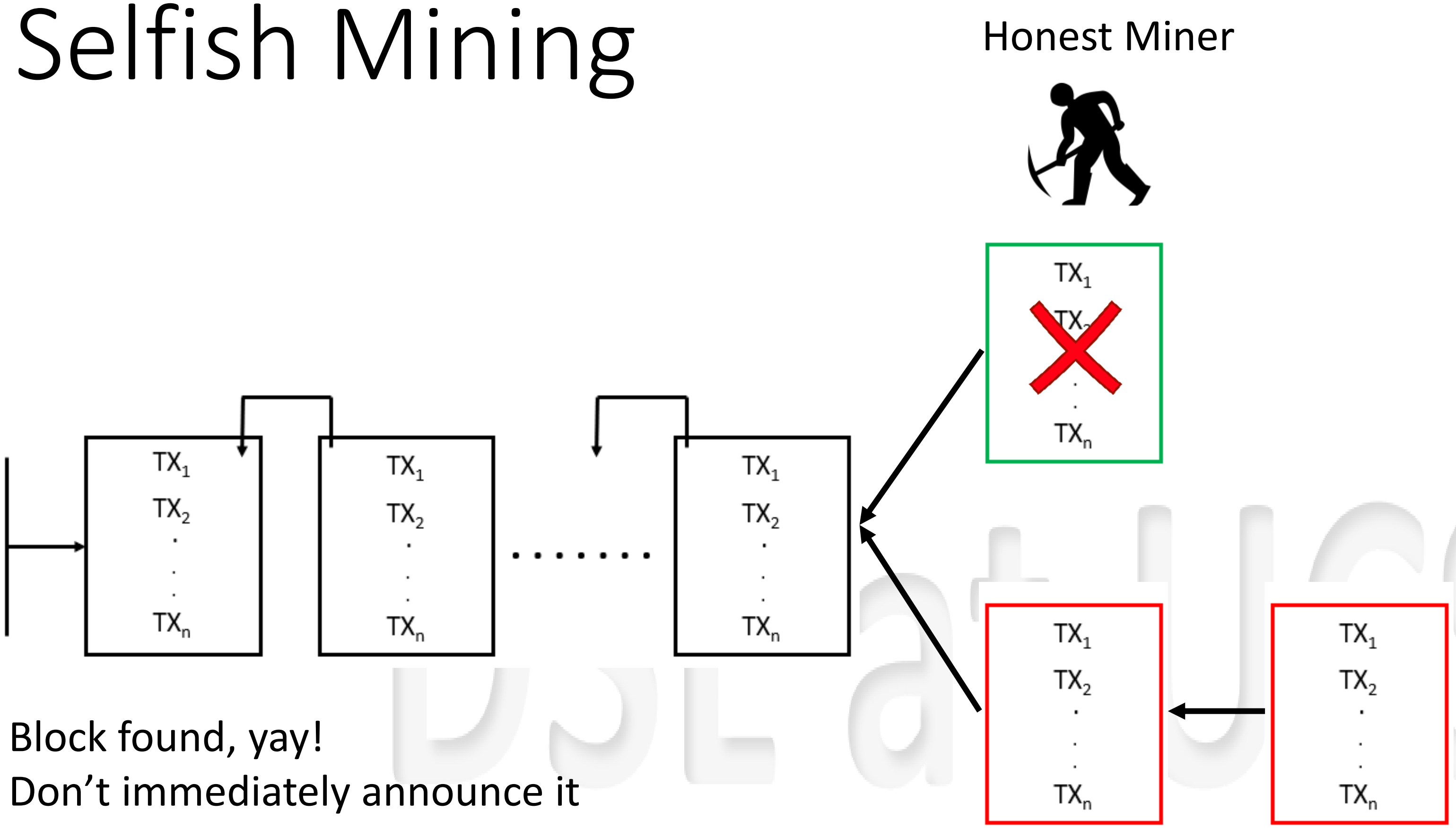


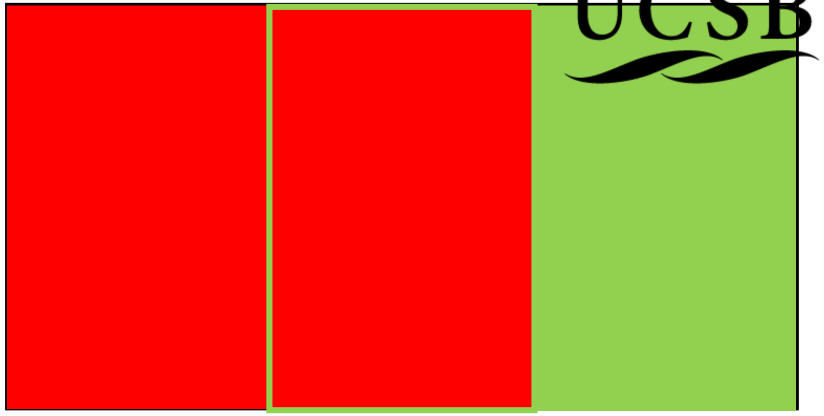
- Block found, yay!
- Don’t immediately announce it
- Let honest miners waste their mining power on an obsolete block
- Start mining the next block (Advantage)
- If an honest miner finds a block, a selfish miner immediately announces their found block
- This splits the power of honest miners

DSL at UCSB

Eyal, Ittay, and Emin Gün Sirer. "Majority is not enough: Bitcoin mining is vulnerable." *Communications of the ACM* 61.7 (2018): 95-102.

Selfish Mining



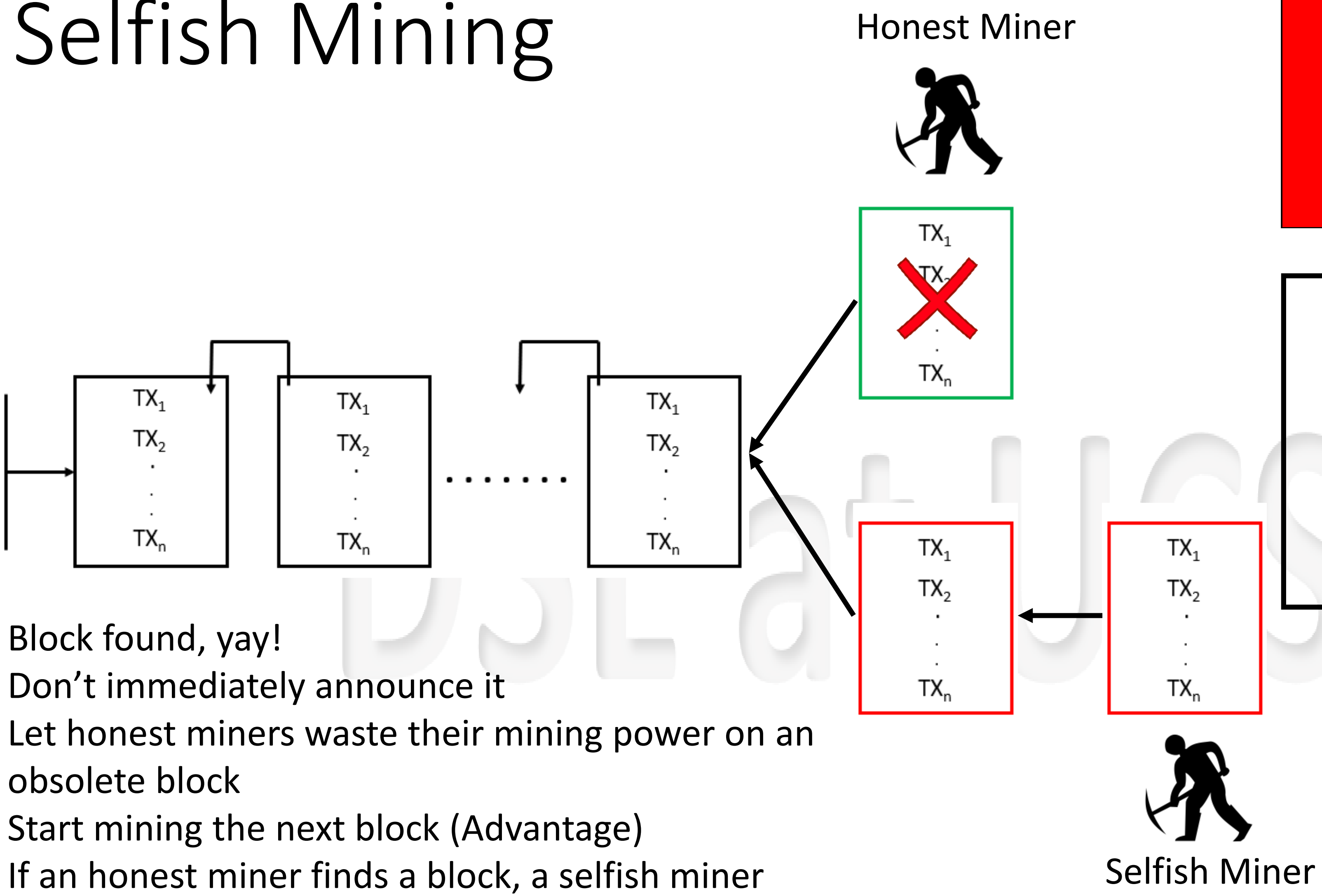


- Block found, yay!
- Don’t immediately announce it
- Let honest miners waste their mining power on an obsolete block
- Start mining the next block (Advantage)
- If an honest miner finds a block, a selfish miner immediately announces their found block
- This splits the power of honest miners

DSL at UCSB

Eyal, Ittay, and Emin Gün Sirer. "Majority is not enough: Bitcoin mining is vulnerable." *Communications of the ACM* 61.7 (2018): 95-102.

Selfish Mining



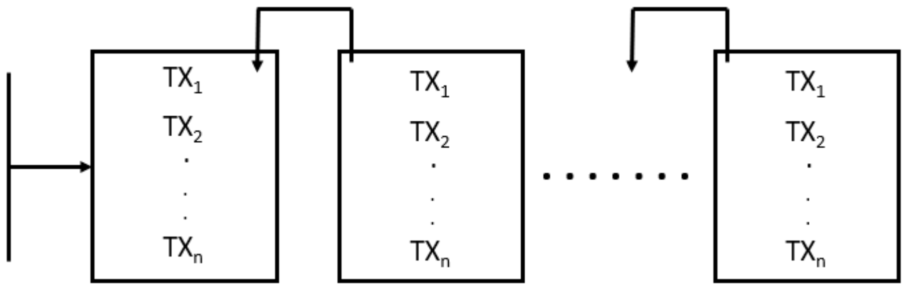
- Block found, yay!
- Don’t immediately announce it
- Let honest miners waste their mining power on an obsolete block
- Start mining the next block (Advantage)
- If an honest miner finds a block, a selfish miner immediately announces their found block
- This splits the power of honest miners

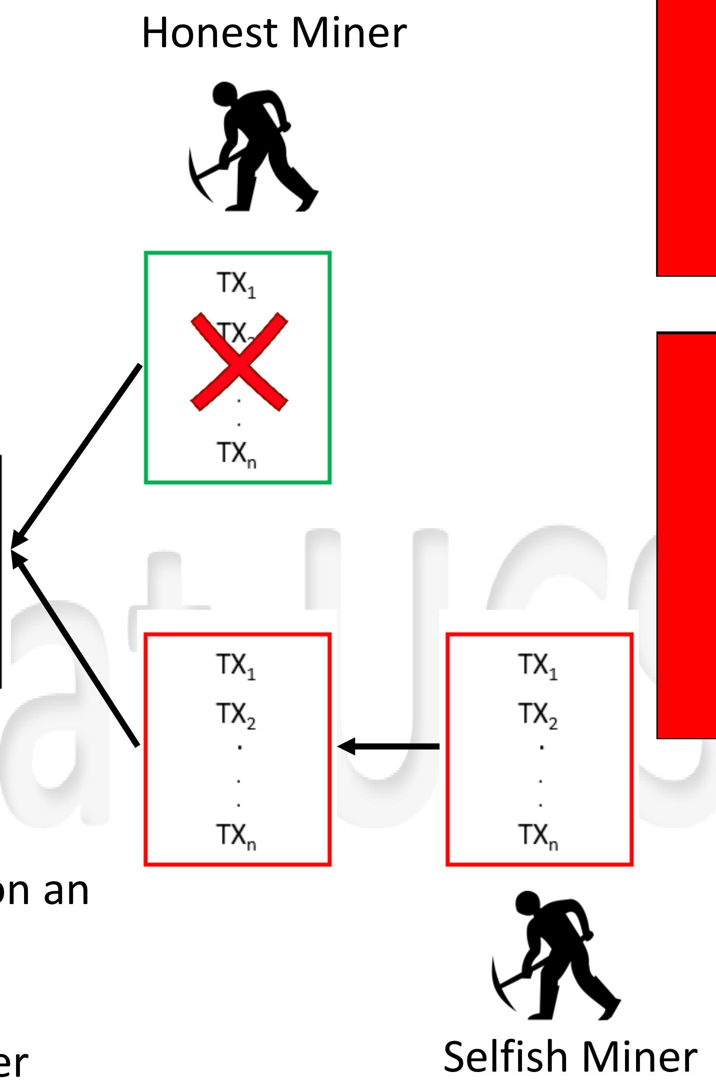
DSL at UCSB

Eyal, Ittay, and Emin Gün Sirer. "Majority is not enough: Bitcoin mining is vulnerable." *Communications of the ACM* 61.7 (2018): 95-102.

DSL

Selfish Mining

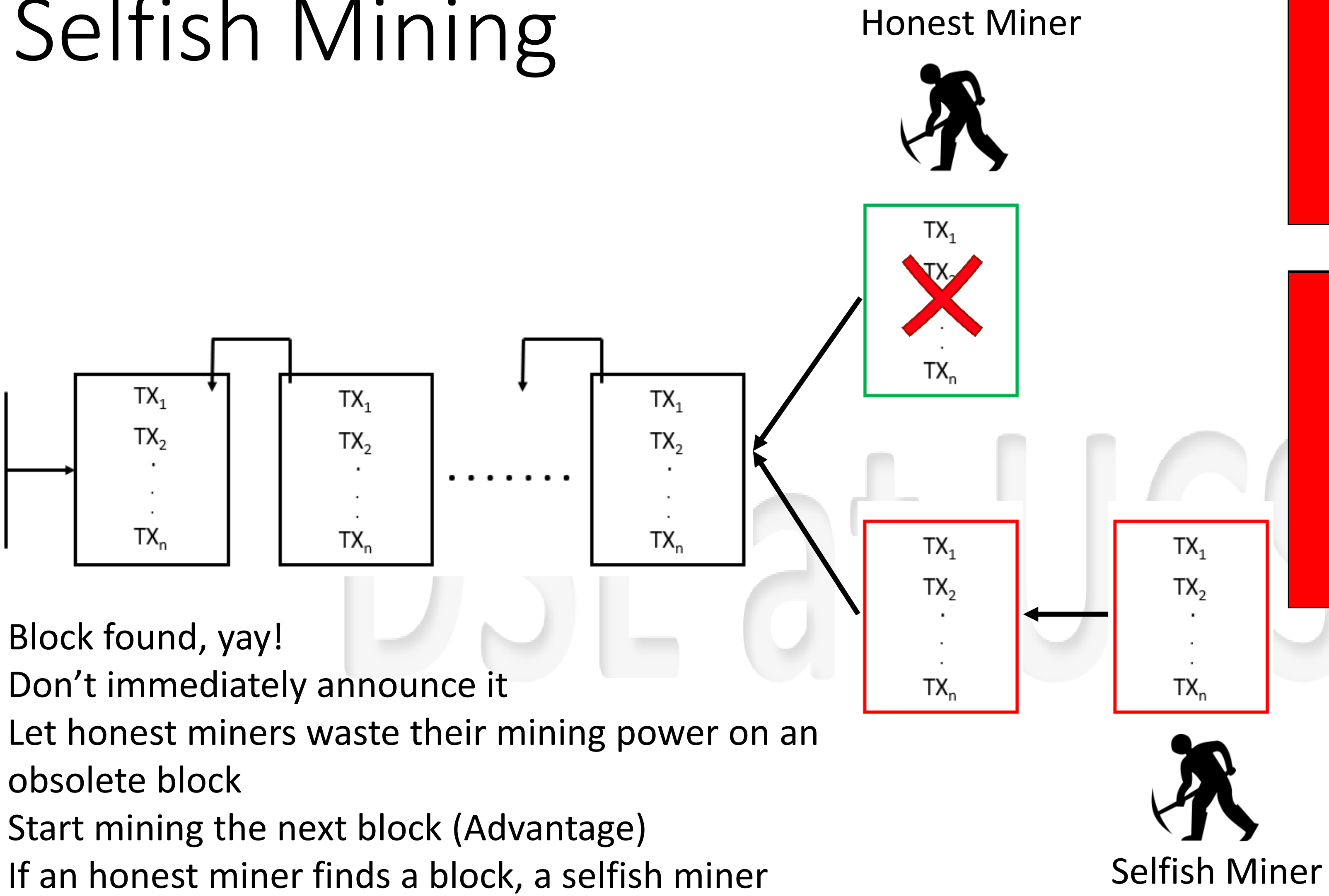




- Block found, yay!
- Don’t immediately announce it
- Let honest miners waste their mining power on an obsolete block
- Start mining the next block (Advantage)
- If an honest miner finds a block, a selfish miner immediately announces their found block
- This splits the power of honest miners

Eyal, Ittay, and Emin Gün Sirer. "Majority is not enough: Bitcoin mining is vulnerable." *Communications of the ACM* 61.7 (2018): 95-102.

Selfish Mining



- Block found, yay!
- Don’t immediately announce it
- Let honest miners waste their mining power on an obsolete block
- Start mining the next block (Advantage)
- If an honest miner finds a block, a selfish miner immediately announces their found block
- This splits the power of honest miners

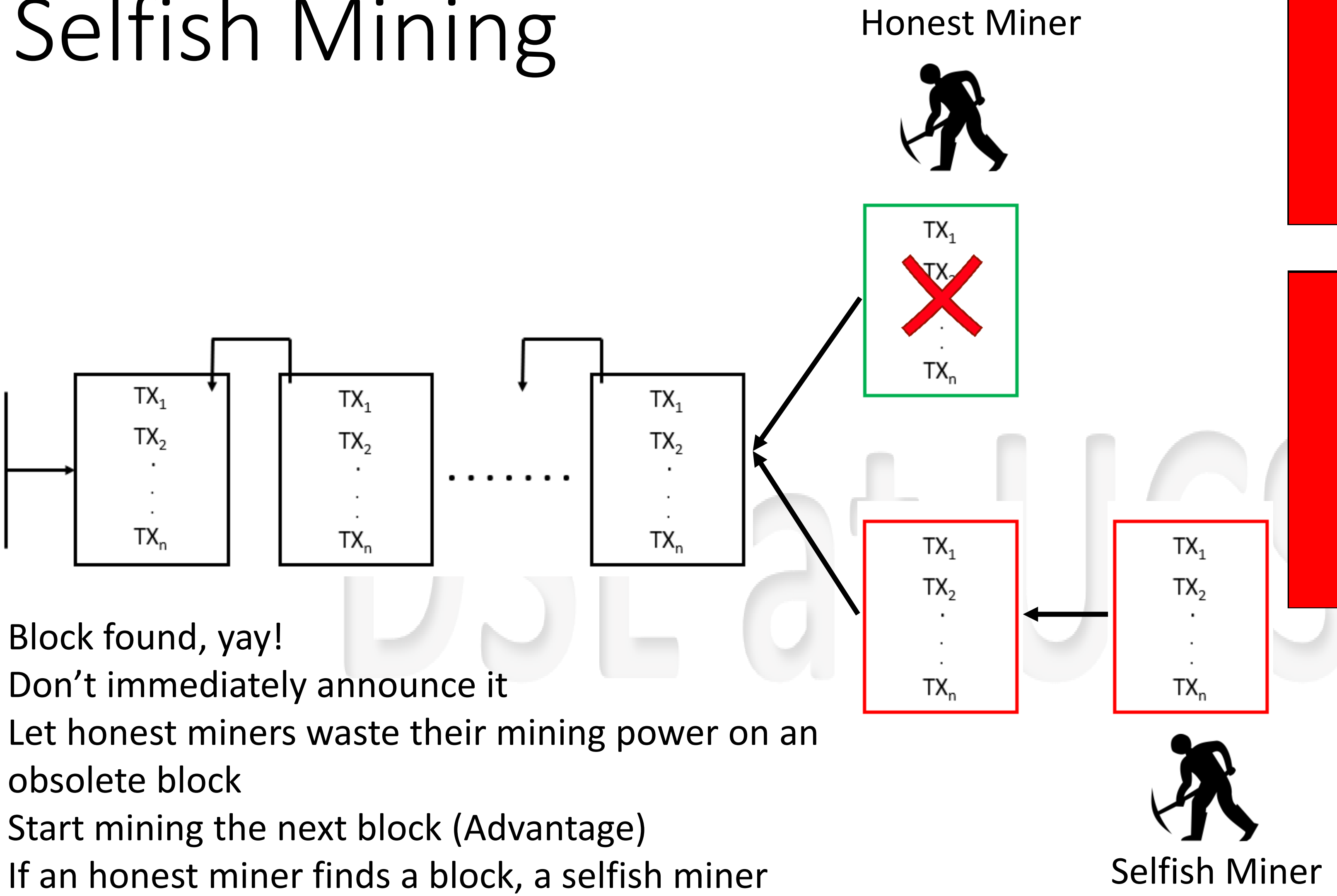
DSL at UCSB

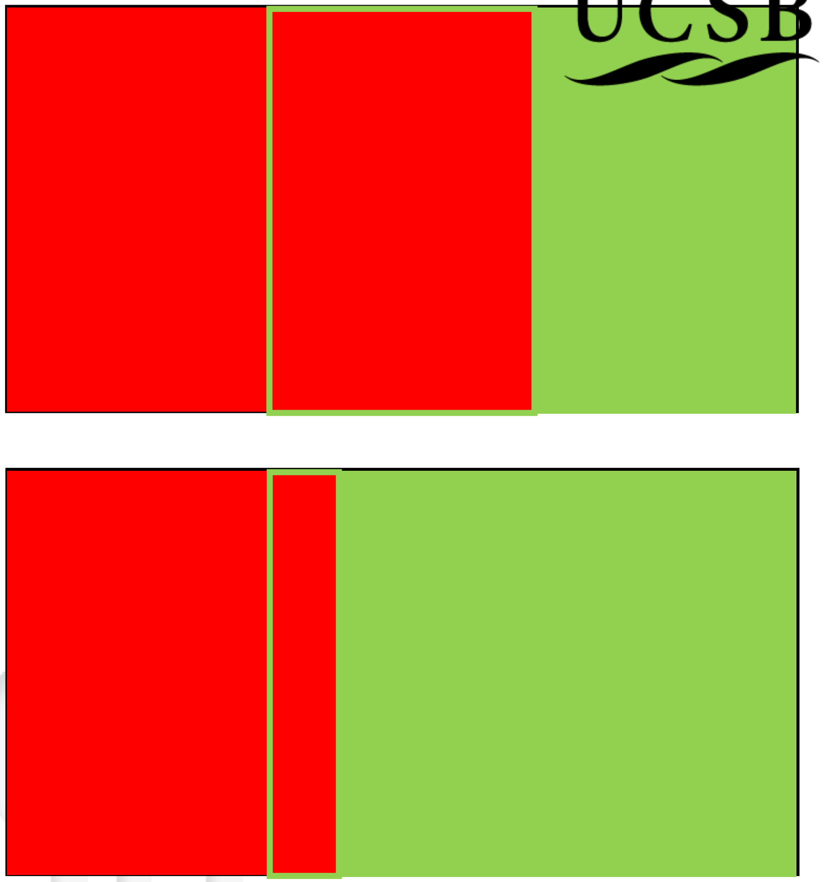
Eyal, Ittay, and Emin Gün Sirer. "Majority is not enough: Bitcoin mining is vulnerable." *Communications of the ACM* 61.7 (2018): 95-102.

DSL

UCSB

Selfish Mining





- Block found, yay!
- Don’t immediately announce it
- Let honest miners waste their mining power on an obsolete block
- Start mining the next block (Advantage)
- If an honest miner finds a block, a selfish miner immediately announces their found block
- This splits the power of honest miners

DSL at UCSB

Eyal, Ittay, and Emin Gün Sirer. "Majority is not enough: Bitcoin mining is vulnerable." *Communications of the ACM* 61.7 (2018): 95-102.

Limitations of Bitcoin

DSL at UCSB

Limitations of Bitcoin

- High transaction-confirmation **latency**

DSL at UCSB

Limitations of Bitcoin

- High transaction-confirmation **latency**
- **Probabilistic** consistency guarantees

DSL at UCSB

Limitations of Bitcoin

- High transaction-confirmation **latency**
- **Probabilistic** consistency guarantees
- Very **low TPS** (Transactions per second) - average of **3 to 7 TPS**

DSL at UCSB

Limitations of Bitcoin

- High transaction-confirmation **latency**
- **Probabilistic** consistency guarantees
- Very **low TPS** (Transactions per second) - average of **3 to 7 TPS**
- New block added every **10 minutes**.

DSL at UCSB

How to scale Bitcoin?

DSL at UCSB

How to scale Bitcoin?

- Two obvious options for increasing Bitcoin's transaction throughput:

DSL at UCSB

How to scale Bitcoin?

- Two obvious options for increasing Bitcoin's transaction throughput: **increase** the size of **blocks**, or **decrease** the block **interval**

DSL at UCSB

How to scale Bitcoin?

- Two obvious options for increasing Bitcoin's transaction throughput: **increase** the size of **blocks**, or **decrease** the block **interval**

- Why they don't work?

DSL at UCSB

How to scale Bitcoin?

- Two obvious options for increasing Bitcoin's transaction throughput: **increase** the size of **blocks**, or **decrease** the block **interval**

- Why they don't work?

  - **Decreases fairness** - giving large miners an advantage

DSL at UCSB

How to scale Bitcoin?

- Two obvious options for increasing Bitcoin's transaction throughput:

  increase the size of **blocks**, or **decrease** the block **interval**

- Why they don't work?
  - **Decreases fairness** - giving large miners an advantage
  - Requires more storage space and verification time

DSL at UCSB

How to scale Bitcoin?

- Two obvious options for increasing Bitcoin's transaction throughput: **increase** the size of **blocks**, or **decrease** the block **interval**

- Why they don't work?
  - **Decreases fairness** - giving large miners an advantage
  - Requires more storage space and verification time
  - Leads to higher number of **forks**

DSL at UCSB

Bitcoin Alternatives

DSL at UCSB

Overview

DSL at UCSB

Overview

- All solutions want to increase txn throughput by reducing consensus amongst all nodes to smaller set of nodes

DSL at UCSB

Overview

- All solutions want to increase txn throughput by reducing consensus amongst all nodes to smaller set of nodes

Mine once, publish txns many times → **BitcoinNG**

DSL at UCSB

Overview

- All solutions want to increase txn throughput by reducing consensus amongst all nodes to smaller set of nodes

Mine once, publish txns many times → **BitcoinNG**

Form a committee to vouch for new block → **ByzCoin**

DSL at UCSB

Overview

- All solutions want to increase txn throughput by reducing consensus amongst all nodes to smaller set of nodes

Mine once, publish txns many times → **BitcoinNG**

Form a committee to vouch for new block → **ByzCoin**

Shard txns across different committees → **Elastico**

DSL at UCSB

Overview

- All solutions want to increase txn throughput by reducing consensus amongst all nodes to smaller set of nodes

Mine once, publish txns many times	BitcoinNG
Form a committee to vouch for new block	ByzCoin
Shard txns across different committees	Elastico
Using committees with Proof-of-stake	Algorand

DSL at UCSB

SOLUTION 1

Mine once, publish txns many times	BitcoinNG
Form a committee to vouch for new block	ByzCoin
Shard txns across different committees	Elastico
Using committees with Proof-of-stake	Algorand

DSL at UCSB

Bitcoin NG (Next Generation)

DSL at UCSB

Bitcoin NG (Next Generation)

- Bitcoin is **retrospective**: a block encases transactions from preceding 10 minutes.

DSL at UCSB

Bitcoin NG (Next Generation)

- Bitcoin is **retrospective**: a block encases transactions from preceding 10 minutes.
- Bitcoin NG is **forward-looking**: elect a leader every 10 minutes and the leader vets for future transactions as they occur.

DSL at UCSB

Bitcoin NG (Next Generation)

- Bitcoin is **retrospective**: a block encases transactions from preceding 10 minutes.

- Bitcoin NG is **forward-looking**: elect a leader every 10 minutes and the leader vets for future transactions as they occur.

DSL at UCSB

Eyal, Ittay, et al. "Bitcoin-NG: A Scalable Blockchain Protocol." *NSDI*. 2016.

Bitcoin NG: Keyblocks and Microblocks

DSL at UCSB

Bitcoin NG: Keyblocks and Microblocks

Observation: In Bitcoin, blocks provide two purpose:
consensus and
txn verification

DSL at UCSB

Bitcoin NG: Keyblocks and Microblocks

Observation: In Bitcoin, blocks provide two purpose:
consensus and
txn verification

DSL at UCSB

Bitcoin NG: Keyblocks and Microblocks

Observation: In Bitcoin, blocks provide two purpose: **consensus** and **txn verification**

Keyblocks: Used for Leader Election and created using Proof-of-work

DSL UCSB

Bitcoin NG: Keyblocks and Microblocks

Observation: In Bitcoin, blocks provide two purpose: **consensus** and **txn verification**

Keyblocks:
Used for Leader Election and created using Proof-of-work

Bitcoin NG: Keyblocks and Microblocks

Observation: In Bitcoin, blocks provide two purpose: **consensus** and **txn verification**

→

Keyblocks: Used for Leader Election and created using Proof-of-work

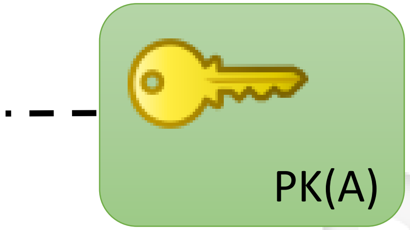
→

Microblocks: Contains txns and is generated by the epoch leader, signed by leader's private key

Keyblocks and Microblocks

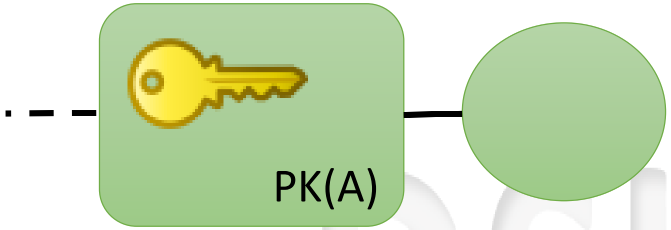
DSL at UCSB

Keyblocks and Microblocks



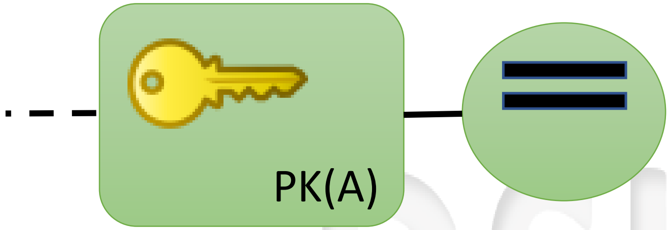
DSL at UCSB

Keyblocks and Microblocks



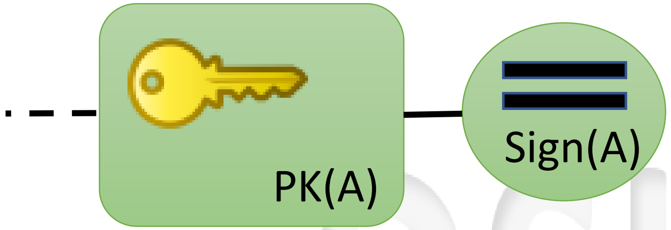
DSL at UCSB

Keyblocks and Microblocks



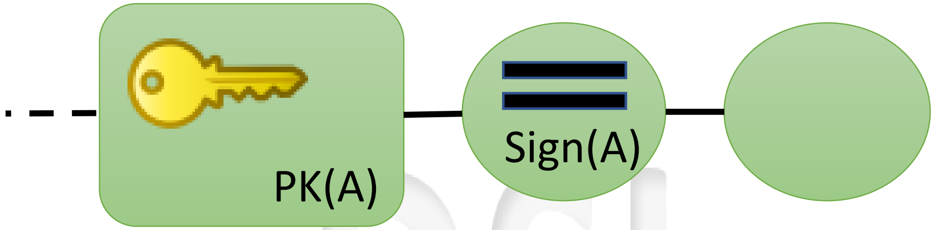
DSL at UCSB

Keyblocks and Microblocks



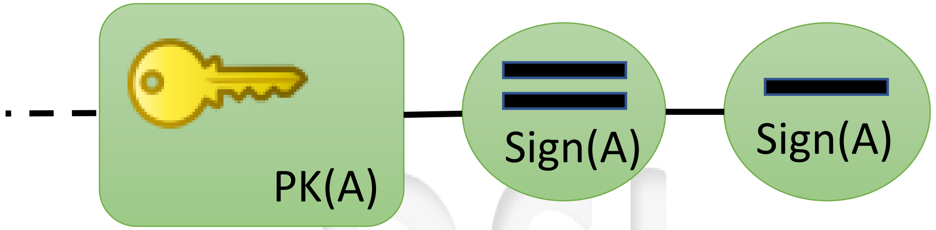
DSL at UCSB

Keyblocks and Microblocks



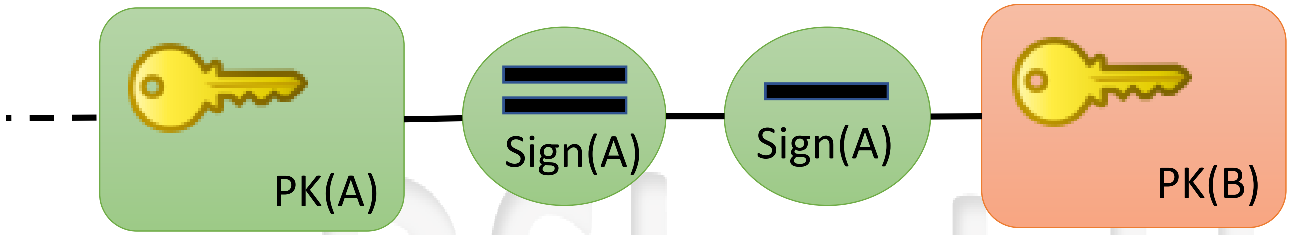
DSL at UCSB

Keyblocks and Microblocks



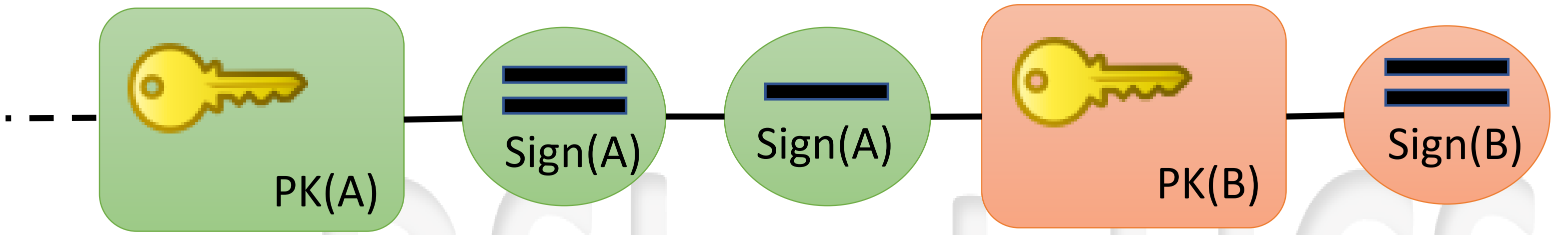
DSL at UCSB

Keyblocks and Microblocks



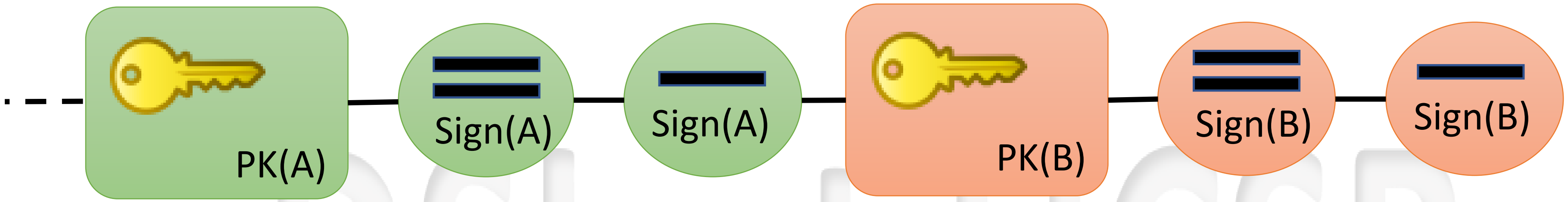
DSL at UCSB

Keyblocks and Microblocks



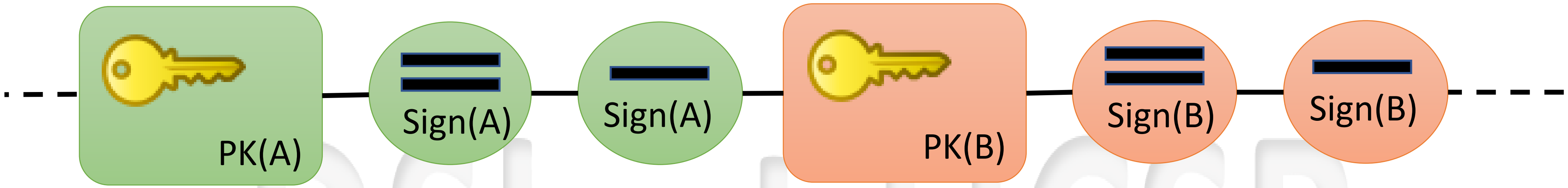
DSL at UCSB

Keyblocks and Microblocks



DSL at UCSB

Keyblocks and Microblocks



DSL at UCSB

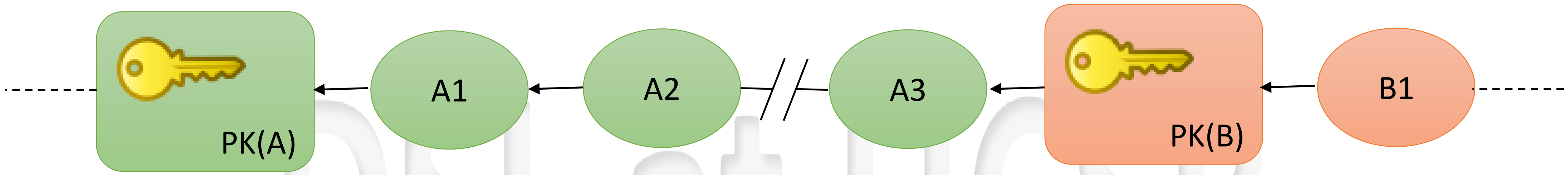
Remuneration

DSL at UCSB

DSL

UCSB

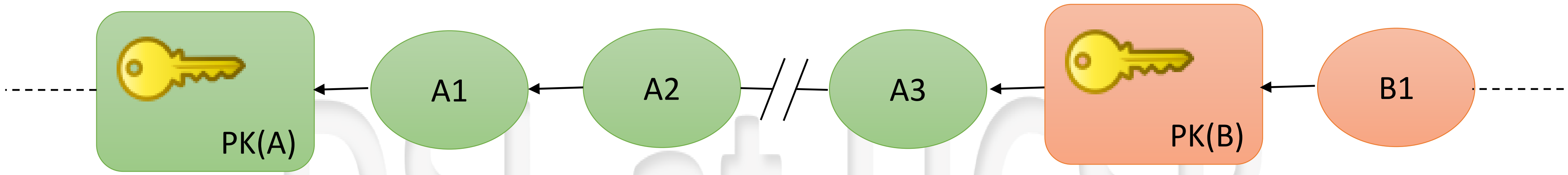
Remuneration



DSL at UCSB

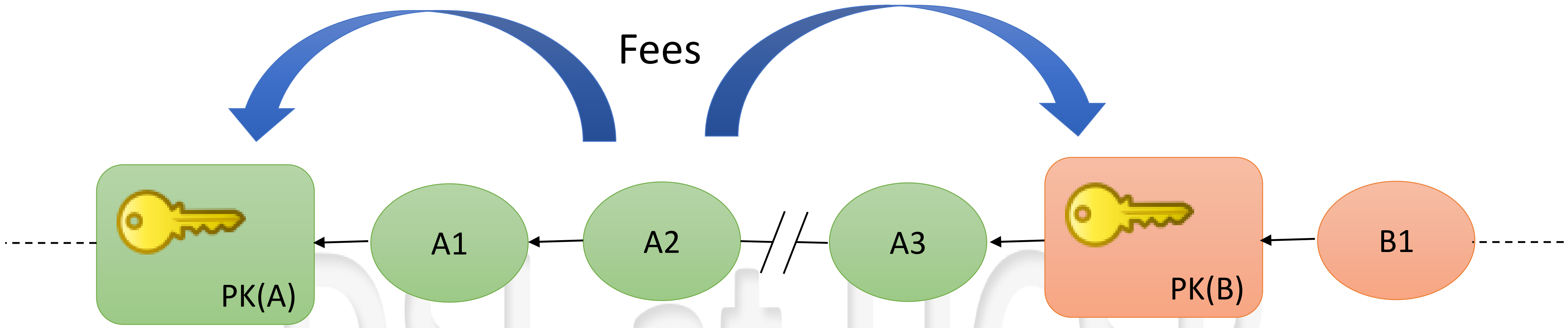
Remuneration

Fees



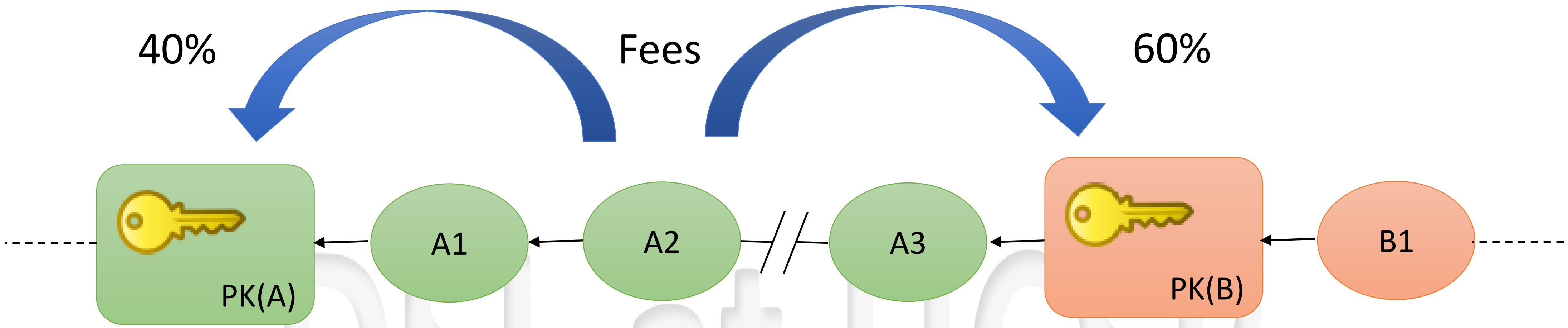
DSL at UCSB

Remuneration



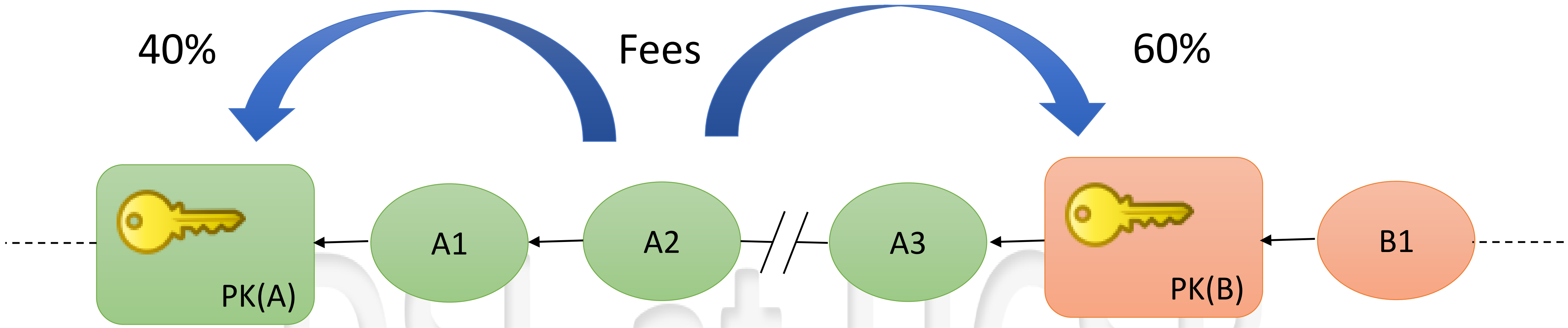
DSL at UCSB

Remuneration



DSL at UCSB

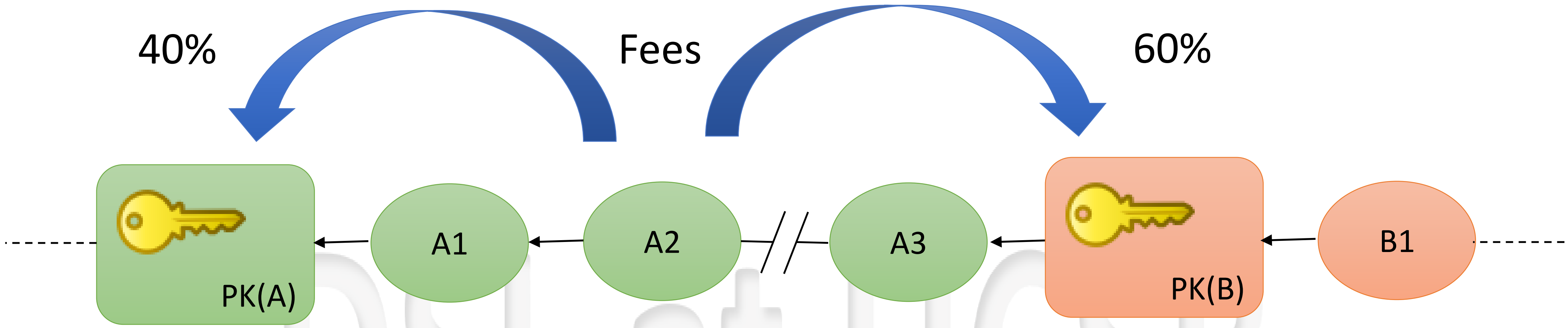
Remuneration



DSL at UCSB

- Encourages next leader to mine on **top** of the **latest** microblock

Remuneration



DSL at UCSB

- Encourages next leader to mine on **top** of the **latest** microblock
- Current leader should be motivated to add more microblocks instead of '**hiding**' them

Forks in BitcoinNG

DSL at UCSB

Forks in BitcoinNG

- Since microblocks generated **cheaply** and **quickly** by the leader

DSL at UCSB

Forks in BitcoinNG

- Since microblocks generated **cheaply** and **quickly** by the leader

➔ leads to **forks** on most leader switches causing **double spending**

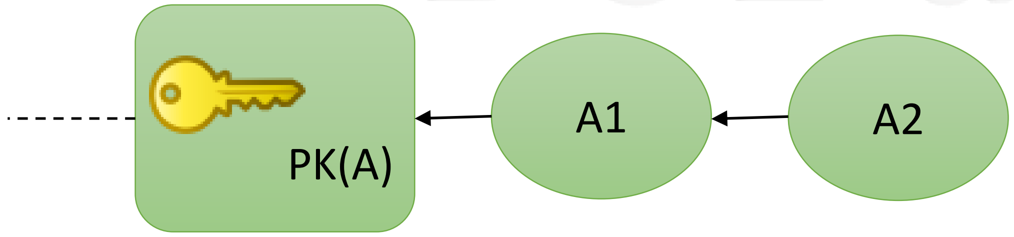
DSL at UCSB

Forks in BitcoinNG

- Since microblocks generated **cheaply** and **quickly** by the leader

➔ leads to **forks** on most leader switches causing **double spending**

DSL at UCSB



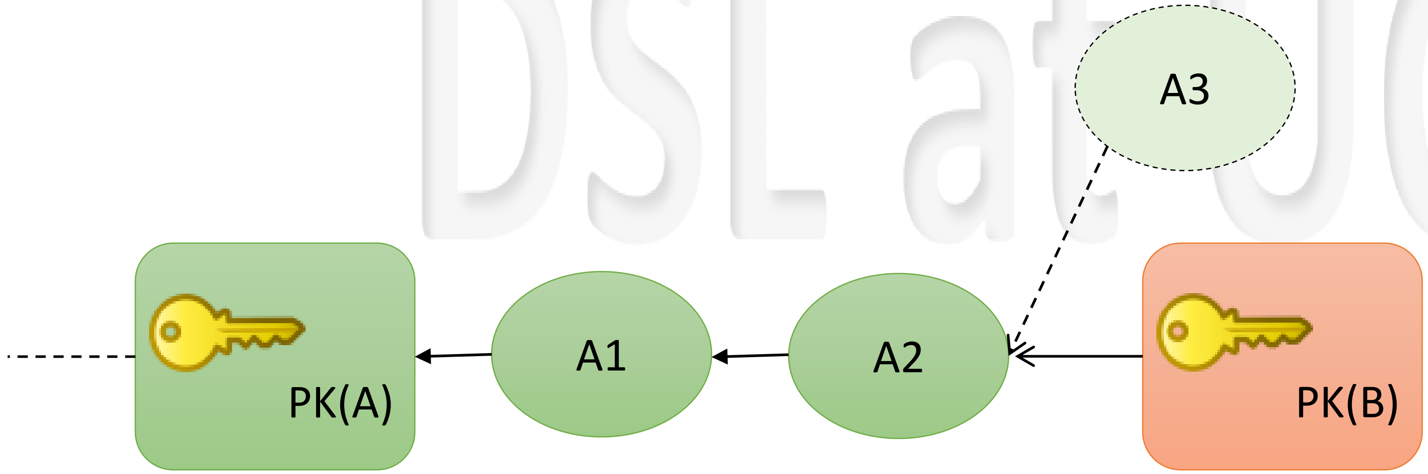
DSL

UCSB

Forks in BitcoinNG

- Since microblocks generated **cheaply** and **quickly** by the leader

➔ leads to **forks** on most leader switches causing **double spending**



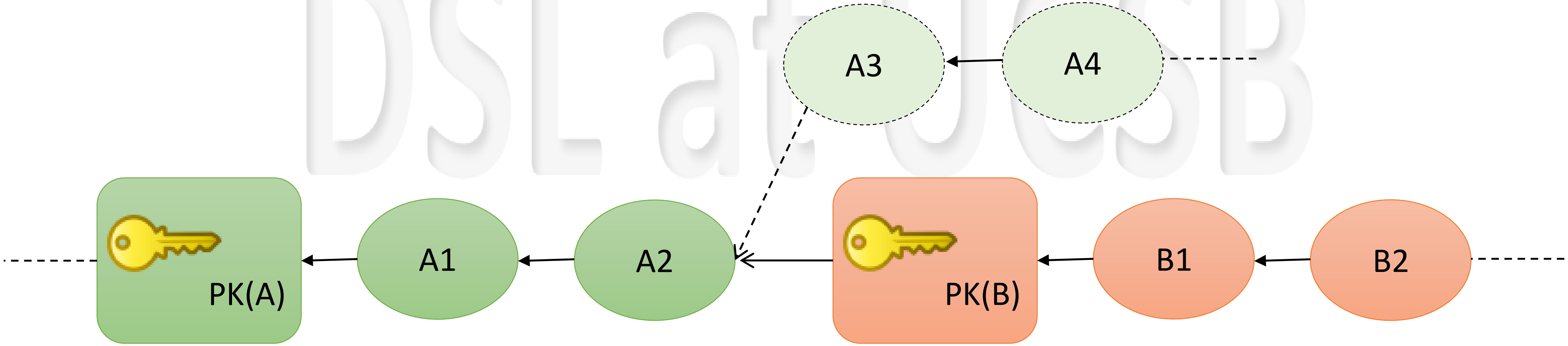
DSL at UCSB

Forks in BitcoinNG

- Since microblocks generated **cheaply** and **quickly** by the leader

➔ leads to **forks** on most leader switches causing **double spending**

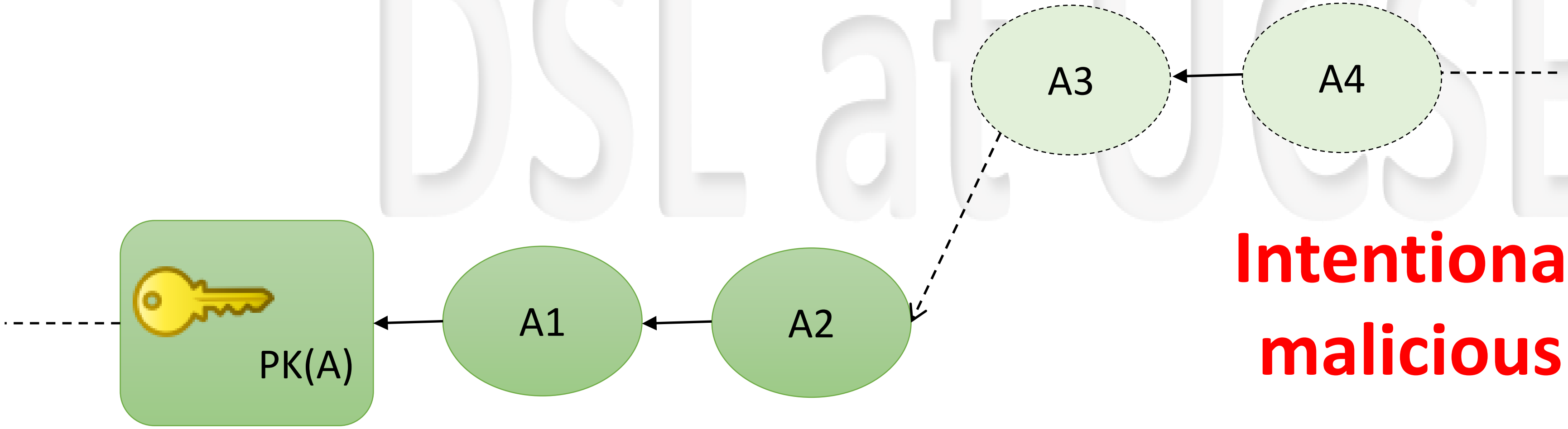
DSL at UCSB



Forks in BitcoinNG

- Since microblocks generated **cheaply** and **quickly** by the leader

➔ leads to **forks** on most leader switches causing **double spending**



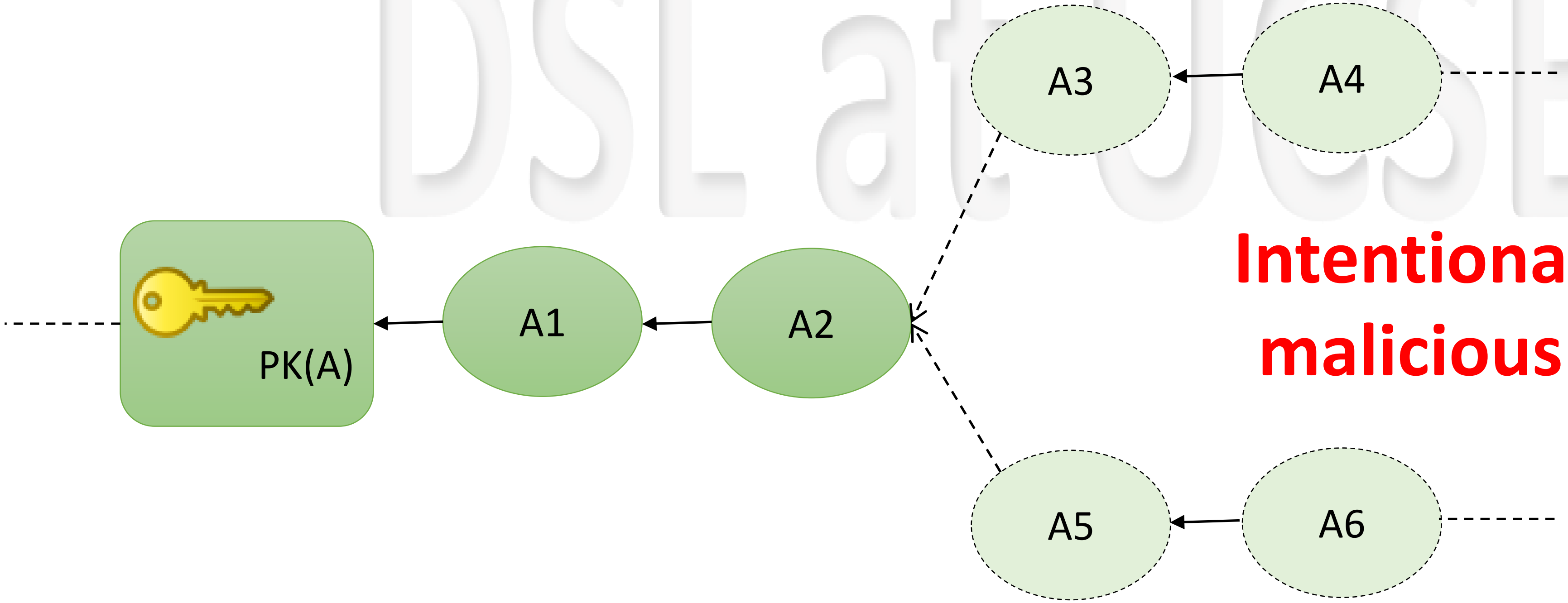
Intentional forks by malicious leader!!

Forks in BitcoinNG

- Since microblocks generated **cheaply** and **quickly** by the leader

➔ leads to **forks** on most leader switches causing **double spending**

DSL at UCSB



Intentional forks by malicious leader!!

Bitcoin-NG review

DSL at UCSB

Bitcoin-NG review

- Does **not** provide **strong consistency** guarantees

DSL at UCSB

Bitcoin-NG review

- Does **not** provide **strong consistency** guarantees
- Does **not** eliminate **selfish mining** by a malicious leader

DSL at UCSB

Bitcoin-NG review

- Does **not** provide **strong consistency** guarantees
- Does **not** eliminate **selfish mining** by a malicious leader
- Still has delay in commitment

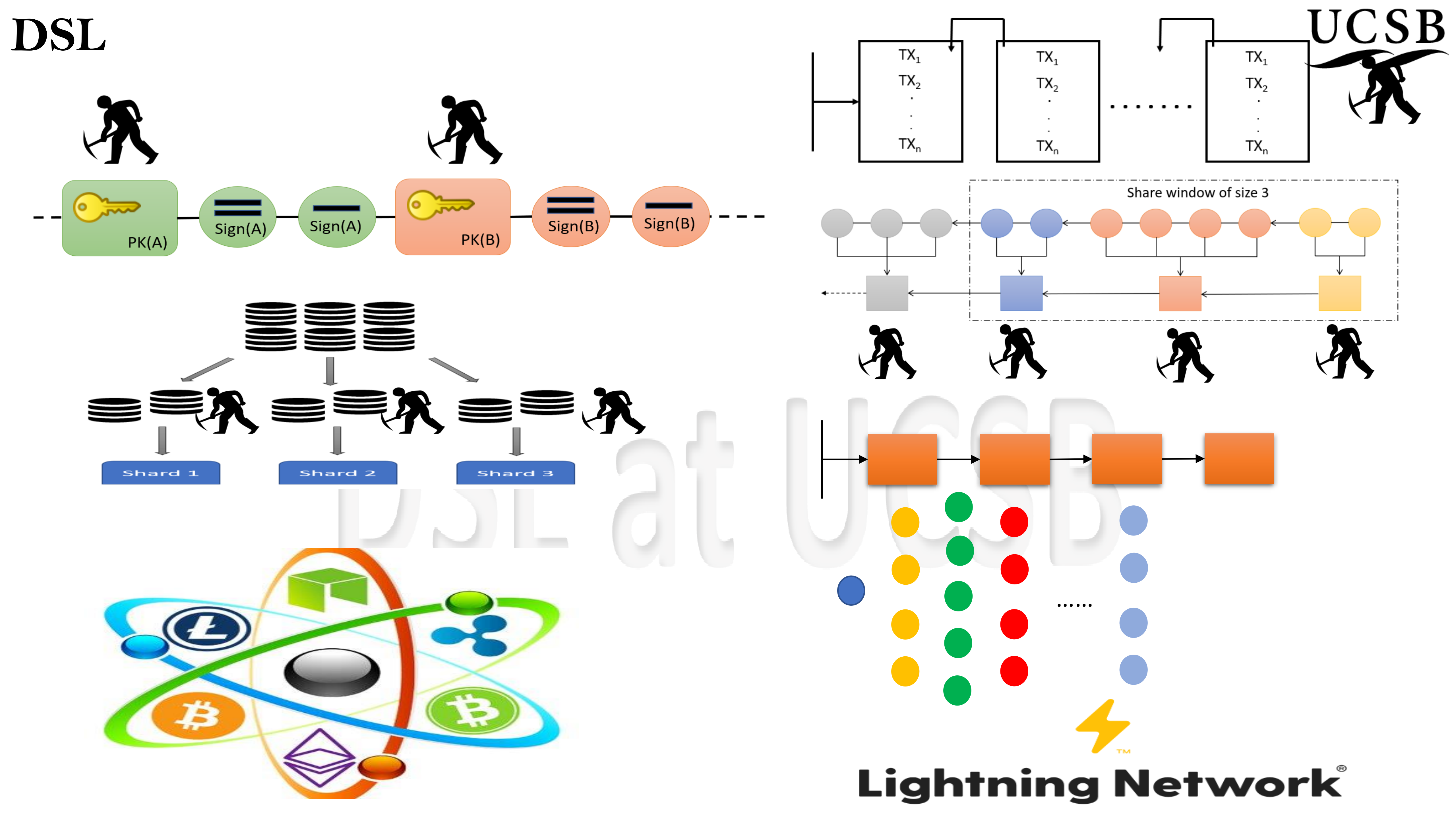
DSL at UCSB

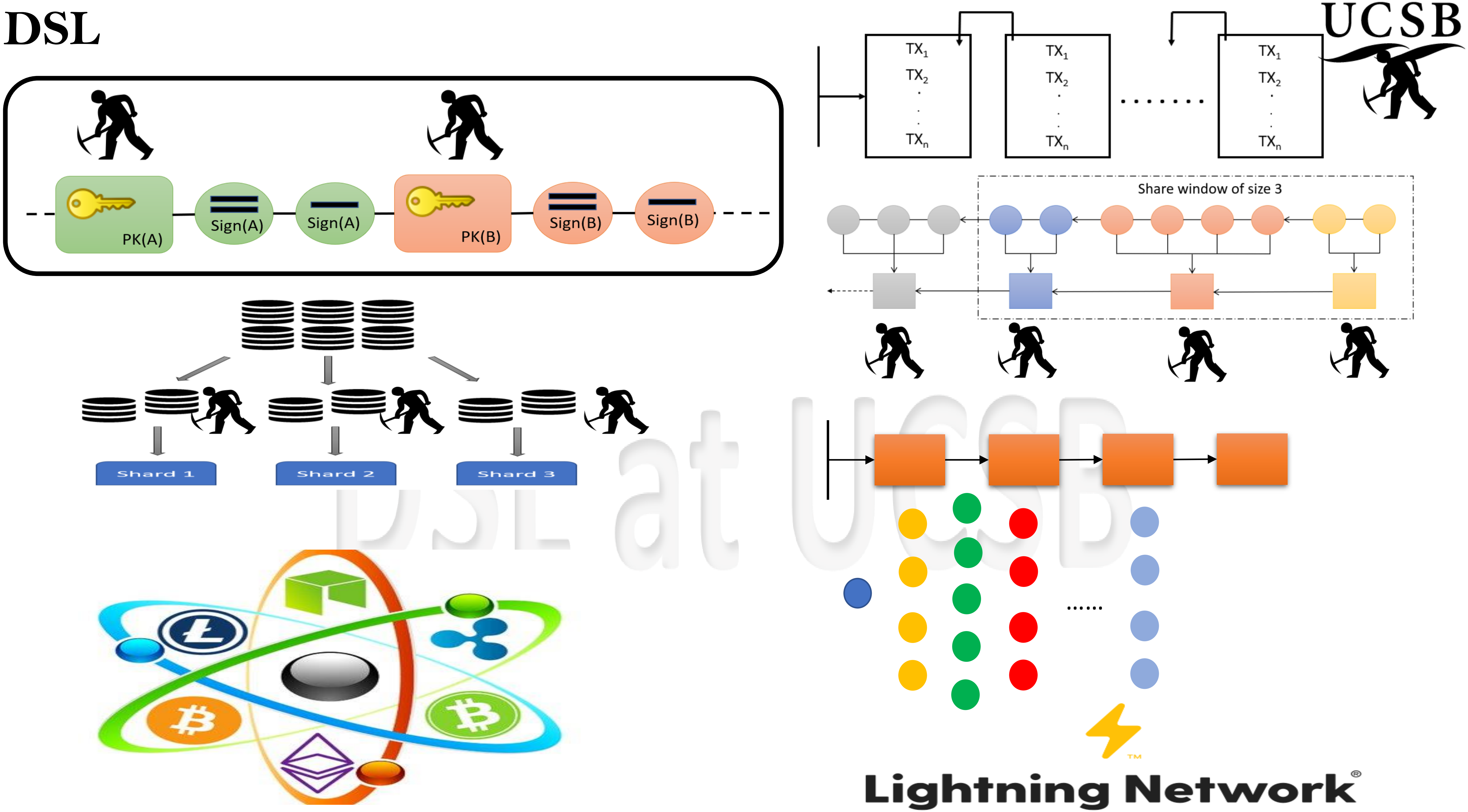
Bitcoin-NG review

- Does **not** provide **strong consistency** guarantees
- Does **not** eliminate **selfish mining** by a malicious leader
- Still has delay in commitment

- But provides key insight in **increasing throughput** and **reducing latency** due to block separation

DSL at UCSB


DSL
UCSB
TX_1
TX_2
TX_n
TX_1
TX_2
TX_n
TX_1
TX_2
TX_n
PK(A)
Sign(A)
Sign(A)
PK(B)
Sign(B)
Sign(B)
Share window of size 3
Shard 1
Shard 2
Shard 3
DSL at UCSB
Lightning Network


DSL
UCSB
PK(A)
Sign(A)
Sign(A)
PK(B)
Sign(B)
Sign(B)
TX1
TX2
TXn
Share window of size 3
Shard 1
Shard 2
Shard 3
DSL at UCSB
Lightning Network®

SOLUTION 2

Mine once, publish txns many times — BitcoinNG

Form a committee to vouch for new block — ByzCoin

Shard txns across different committees — Elastico

Using committees with Proof-of-stake — Algorand

DSL at UCSB

DSL

ByzCoin

Enhancing Bitcoin Security & Performance With Strong Consistency via Collective Signing

UCSB

DSL at UCSB

DSL

UCSB

ByzCoin

Enhancing Bitcoin Security & Performance With Strong Consistency via Collective Signing

To commit Bitcoin transactions irreversibly(strong consistency) within seconds

DSL at UCSB

ByzCoin

Enhancing Bitcoin Security & Performance With Strong Consistency via Collective Signing

To commit Bitcoin transactions irreversibly(strong consistency) within seconds

ByzCoin = Practical Byzantine Fault Tolerance + Collective Signing

ByzCoin

Enhancing Bitcoin Security & Performance With Strong Consistency via Collective Signing

To commit Bitcoin transactions irreversibly(strong consistency) within seconds

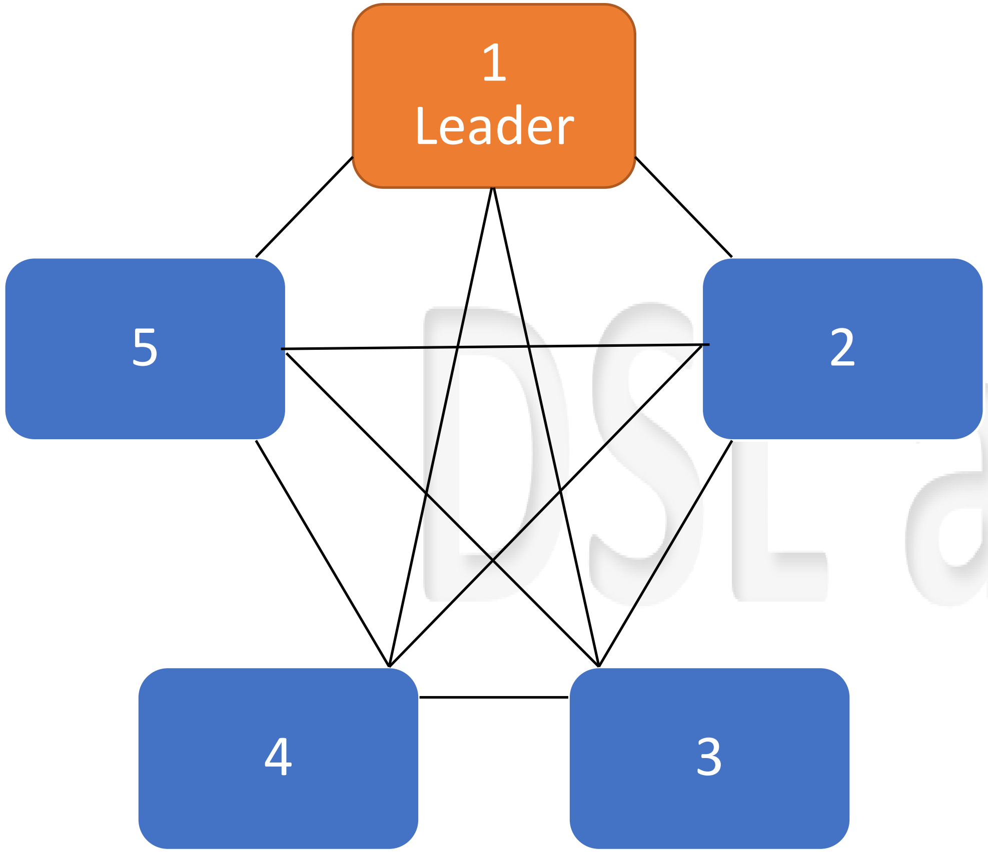
ByzCoin = Practical Byzantine Fault Tolerance + Collective Signing

Kogias, Eleftherios Kokoris, et al. "Enhancing bitcoin security and performance with strong consistency via collective signing." *25th USENIX Security Symposium (USENIX Security 16)*. 2016.

Strawman Design: PBFTCoin

DSL at UCSB

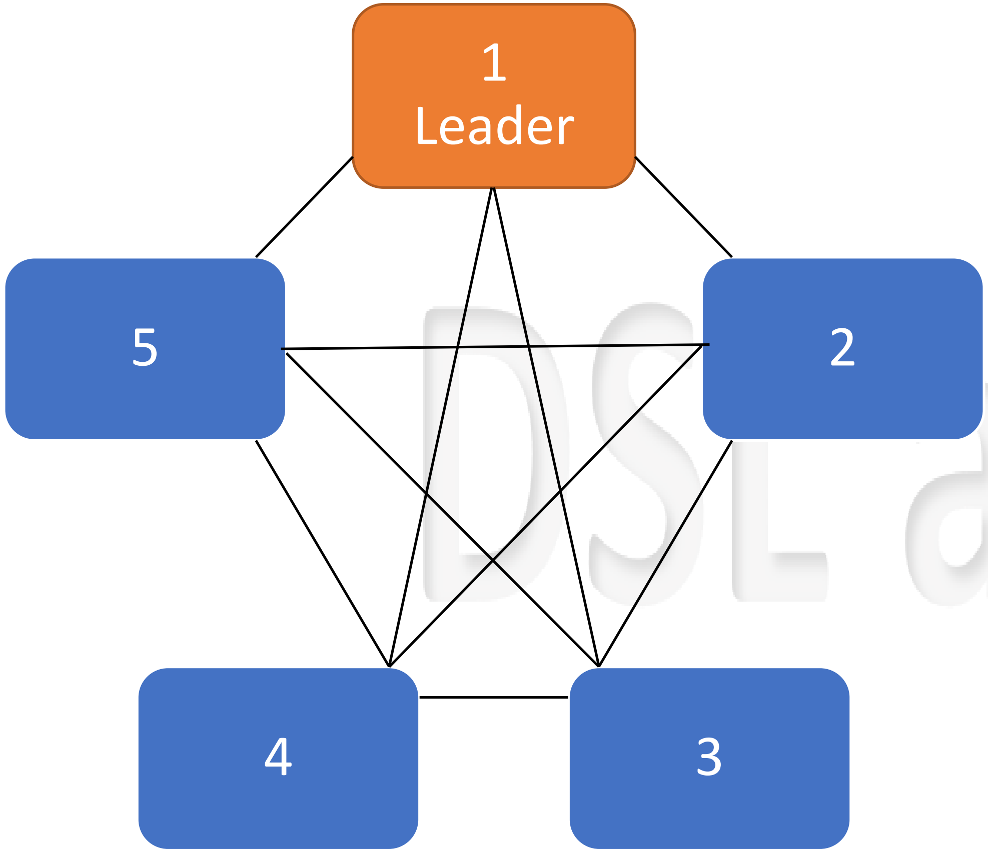
Strawman Design: PBFTCoin



- Naïve, unrealistic but simple: PBFT + Bitcoin

DSL at UCSB

Strawman Design: PBFTCoin



- Naïve, unrealistic but simple: PBFT + Bitcoin

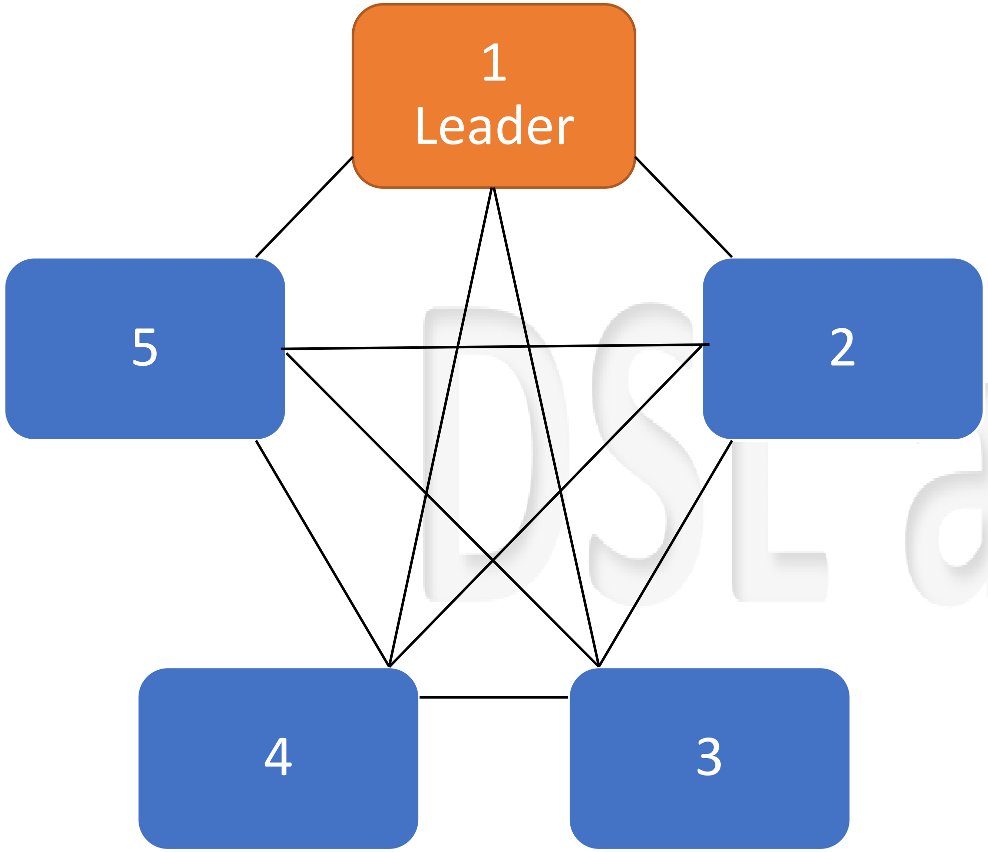
- **TRUSTEES**: 3*f*+1 replicas, at max *f* faulty

DSL at UCSB

DSL

UCSB

Strawman Design: PBFTCoin



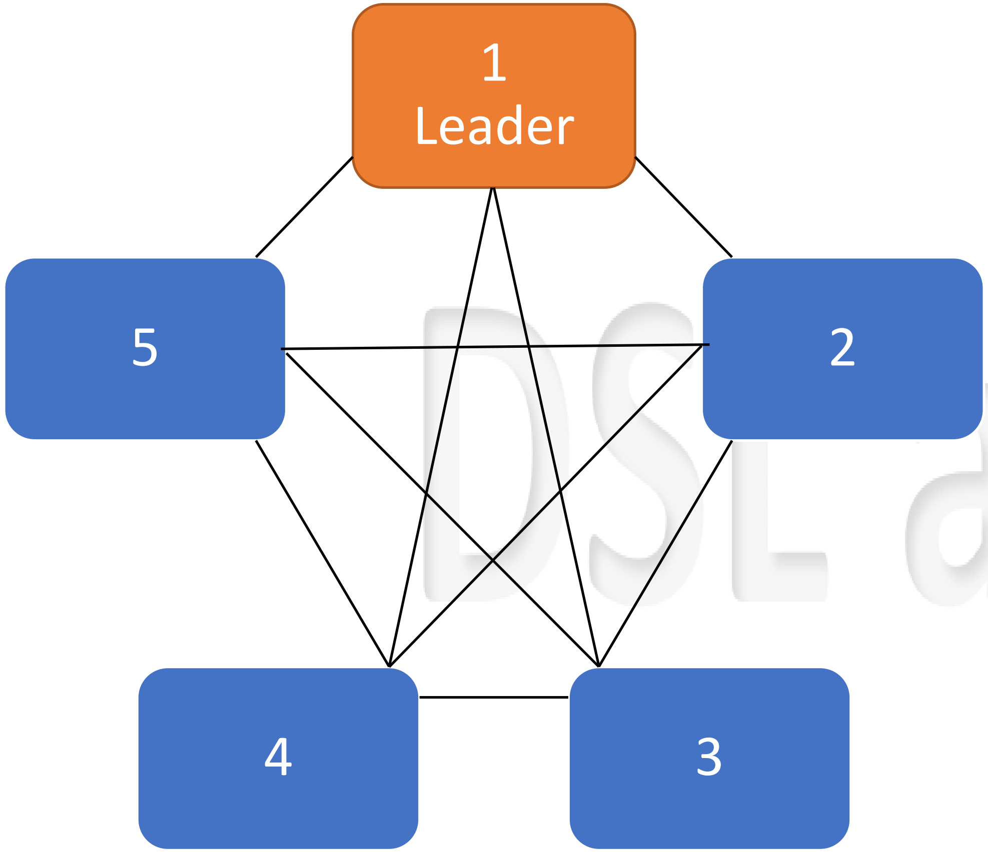
- Naïve, unrealistic but simple: PBFT + Bitcoin

- **TRUSTEES**: 3*f*+1 replicas, at max *f* faulty

- Trustees run PBFT to decide next block

DSL at UCSB

Strawman Design: PBFTCoin



- Naïve, unrealistic but simple: PBFT + Bitcoin

- **TRUSTEES**: 3*f*+1 replicas, at max *f* faulty

- Trustees run PBFT to decide next block

- COMMUNICATION COMPLEXITY : **O(n^2)**

DSL at UCSB

Using PBFT for Bitcoin's open membership

DSL at UCSB

Using PBFT for Bitcoin's open membership

Step 1: Opening the Consensus Group

DSL at UCSB

Using PBFT for Bitcoin's open membership

Step 1: Opening the Consensus Group

- Fixed size dynamically changing sliding **SHARE window**

DSL at UCSB

Using PBFT for Bitcoin's open membership

Step 1: Opening the Consensus Group

- Fixed size dynamically changing sliding **SHARE window**

- **Incentive** = new block's transaction fee split by consensus group

DSL at UCSB

Using PBFT for Bitcoin's open membership
Step 1: Opening the Consensus Group

- Fixed size dynamically changing sliding **SHARE window**
- **Incentive** = new block’s transaction fee split by consensus group
- **Voting power of miner** = No. of blocks the miner has successfully mined in the current window

DSL at UCSB

Using PBFT for Bitcoin's open membership Step 1: Opening the Consensus Group

- Fixed size dynamically changing sliding **SHARE window**
- **Incentive** = new block's transaction fee split by consensus group
- **Voting power of miner** = No. of blocks the miner has successfully mined in the current window
- Last miner is leader. Leader proposes the block

DSL at UCSB

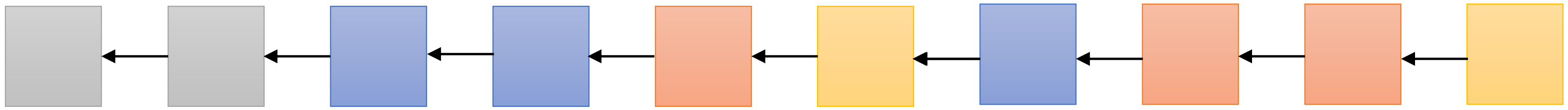
Step 1. ByzCoin's blockchain

DSL at UCSB

DSL

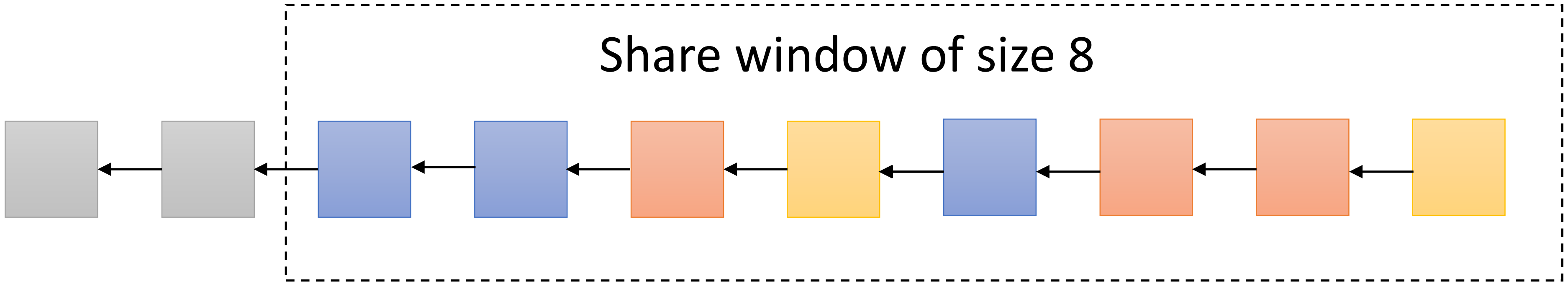
UCSB

Step 1. ByzCoin's blockchain



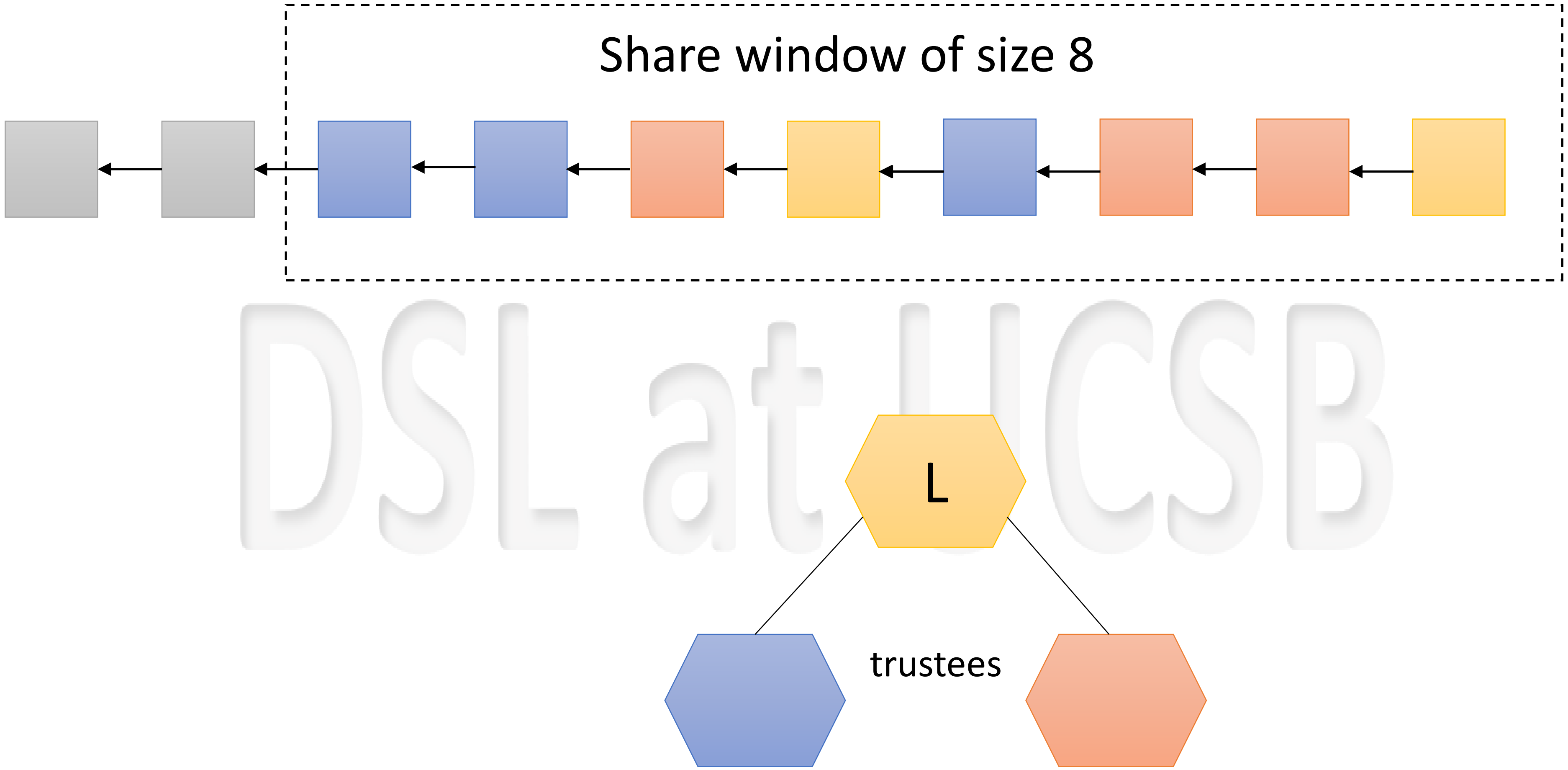
DSL at UCSB

Step 1. ByzCoin's blockchain



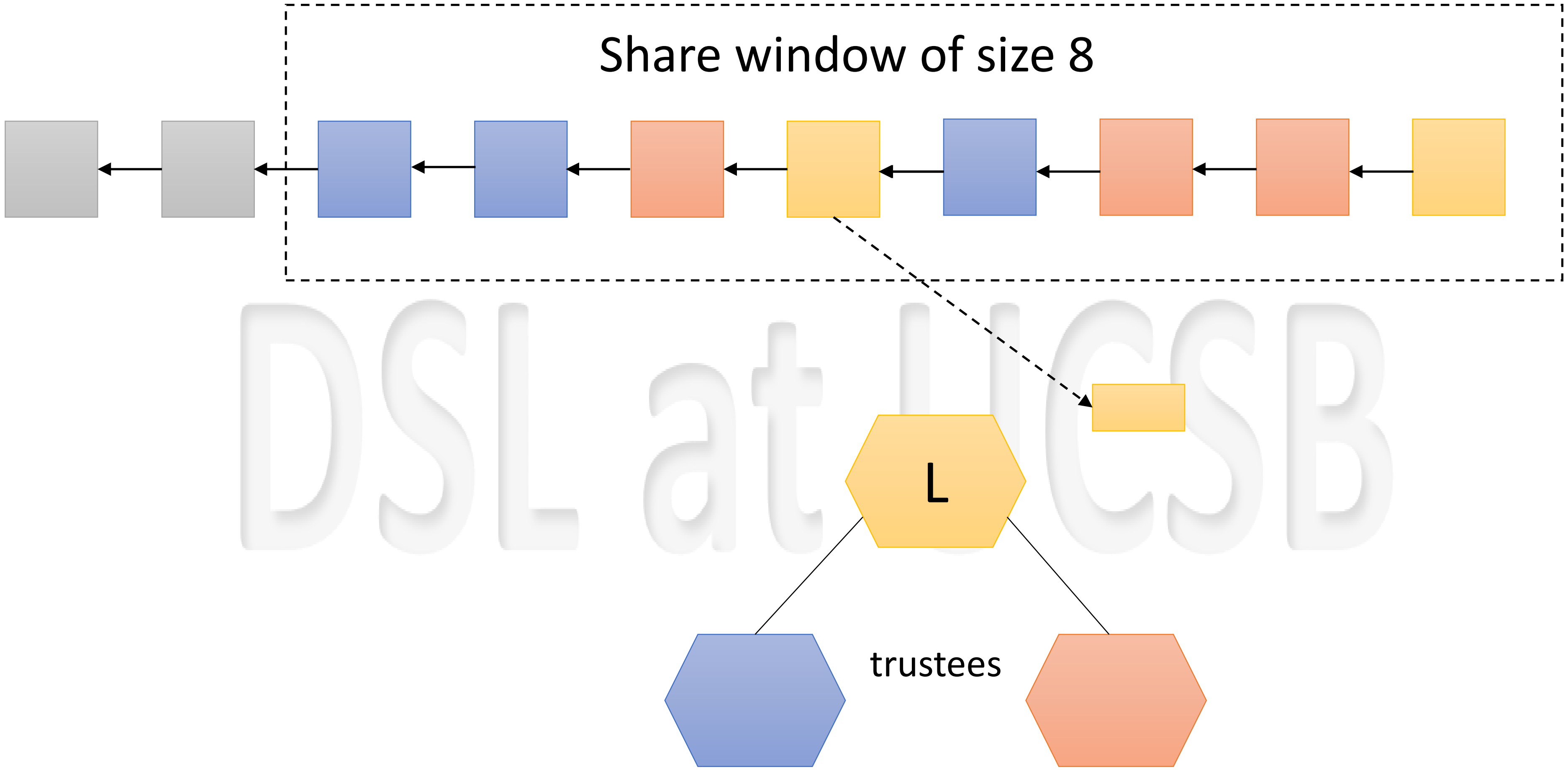
DSL at UCSB

Step 1. ByzCoin's blockchain



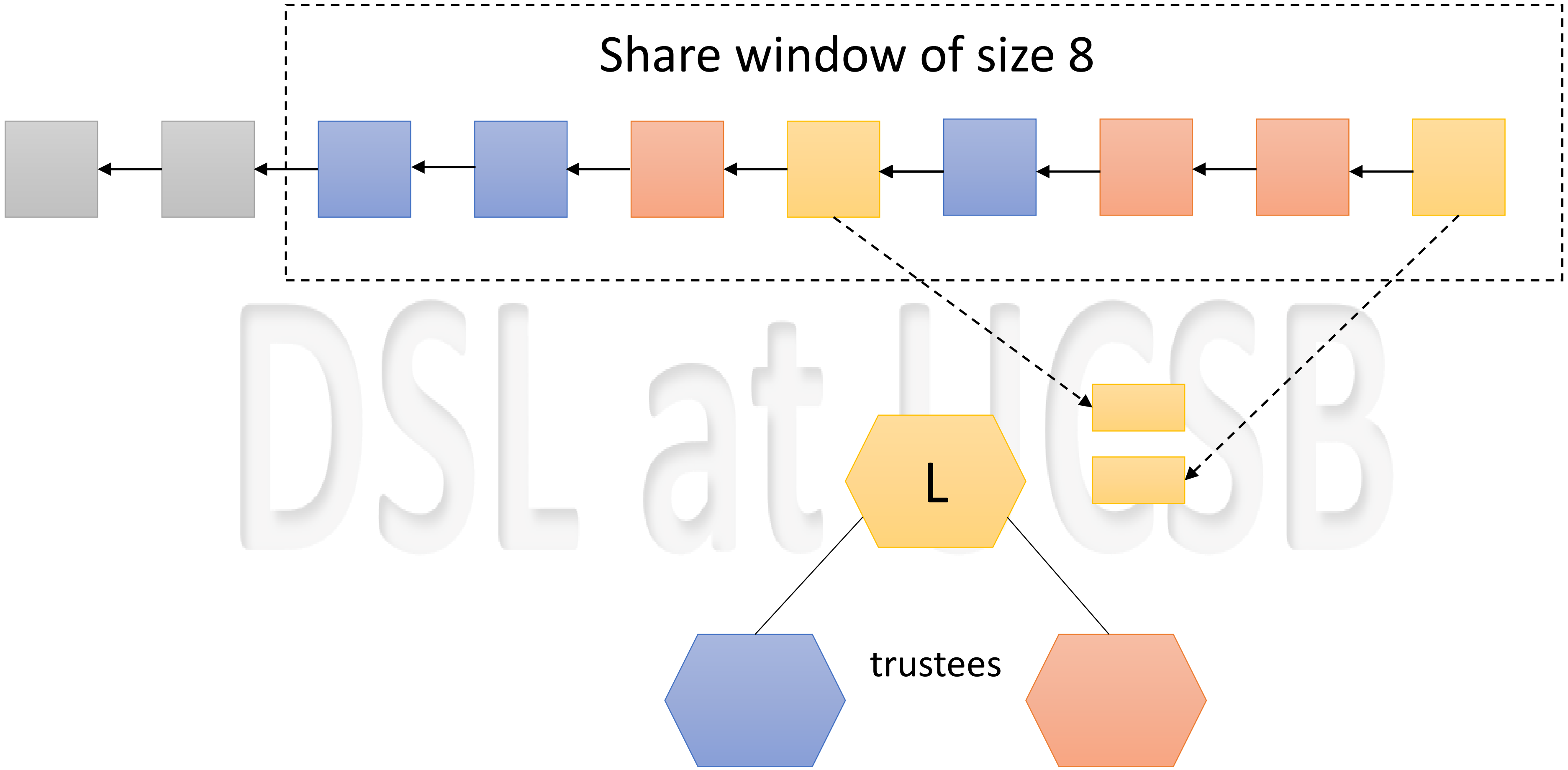
DSL at UCSB

Step 1. ByzCoin's blockchain



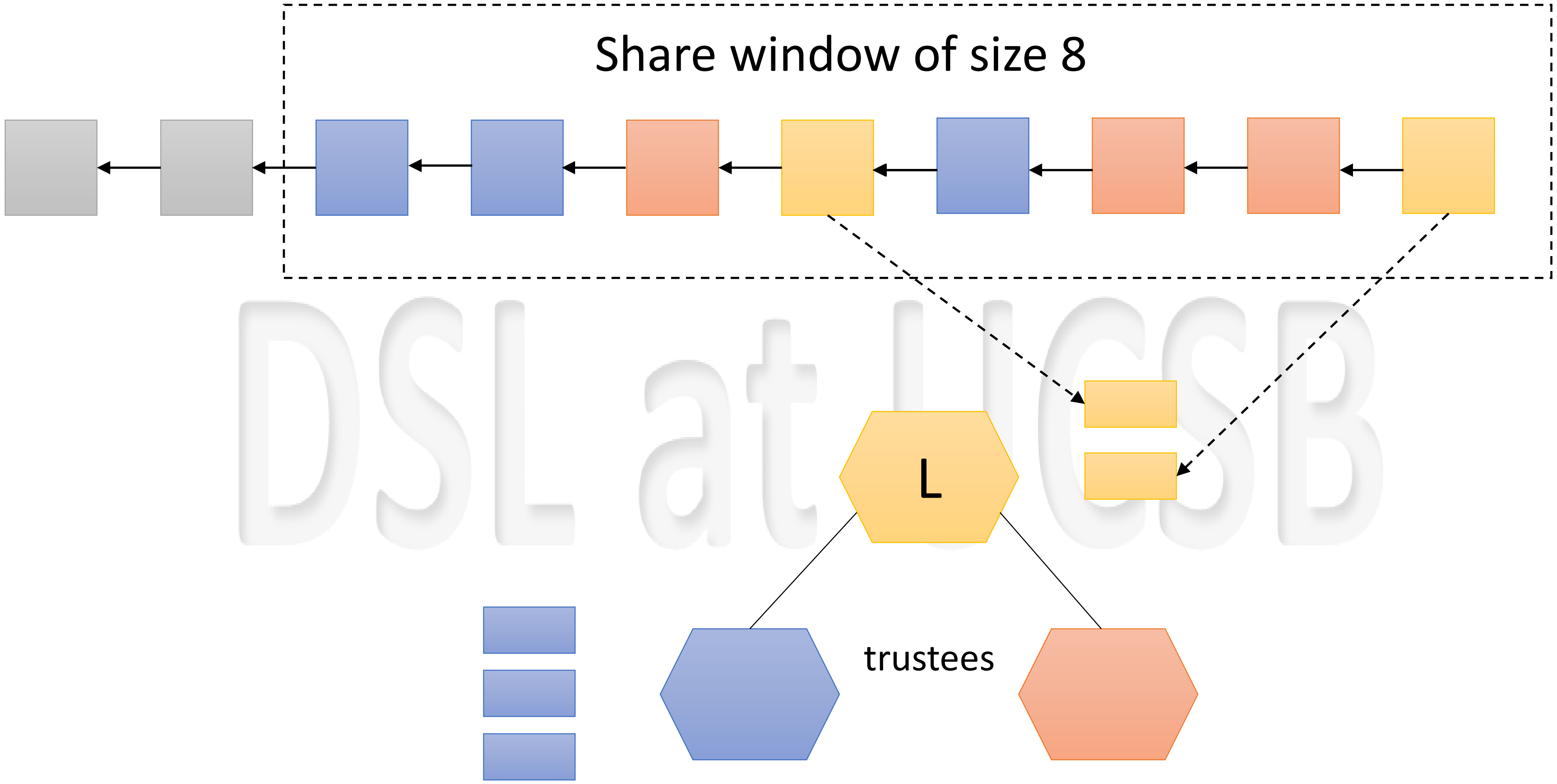
DSL at UCSB

Step 1. ByzCoin's blockchain



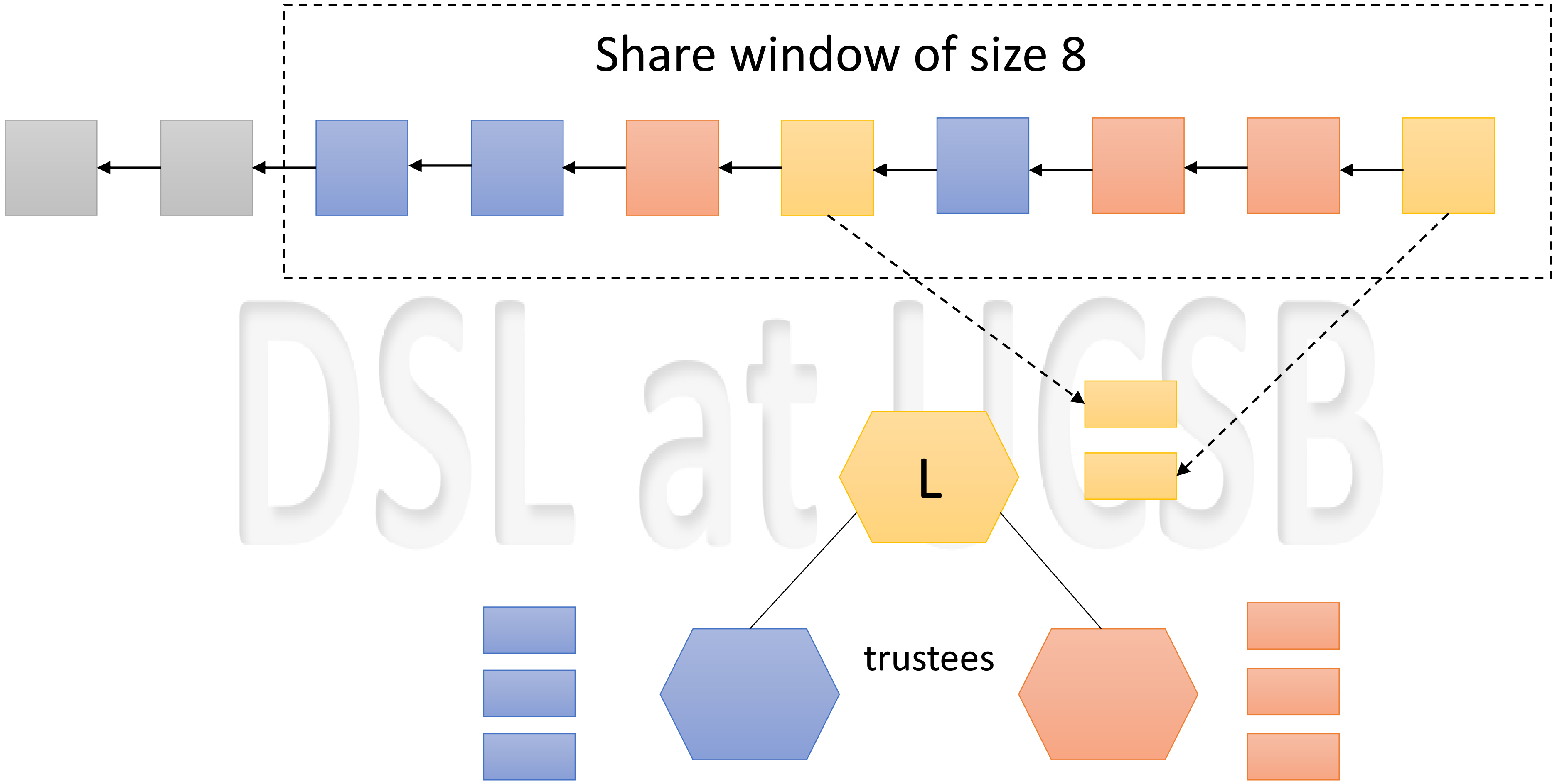
DSL at UCSB

Step 1. ByzCoin's blockchain



DSL at UCSB

Step 1. ByzCoin's blockchain



DSL at UCSB

Step 2: Decoupling Txn Verification from Leader Election

DSL at UCSB

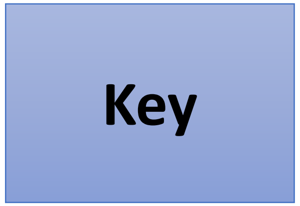
Step 2: Decoupling Txn Verification from Leader Election

- 2 different kinds of blocks:

DSL at UCSB

Step 2: Decoupling Txn Verification from Leader Election

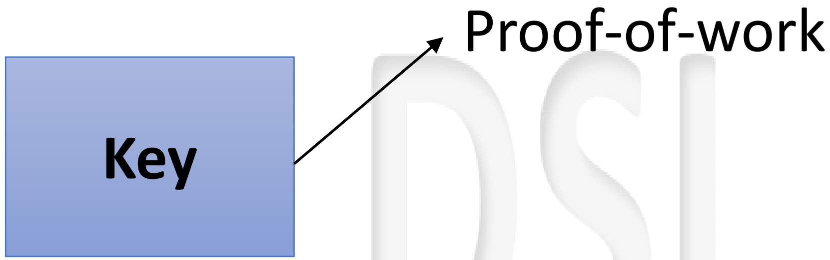
- 2 different kinds of blocks:



DSL at UCSB

Step 2: Decoupling Txn Verification from Leader Election

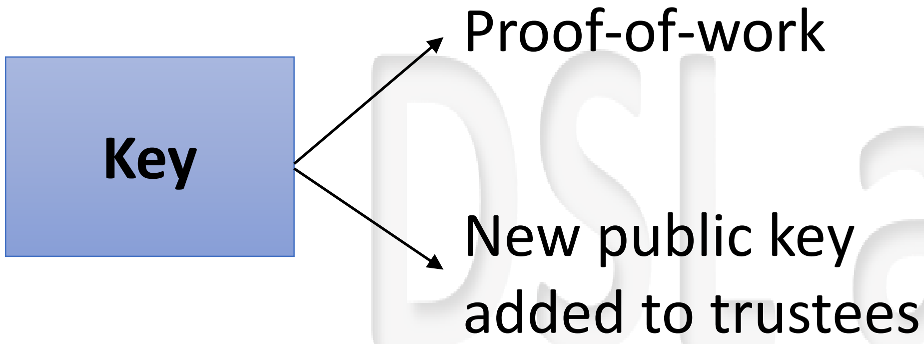
- 2 different kinds of blocks:



DSL at UCSB

Step 2: Decoupling Txn Verification from Leader Election

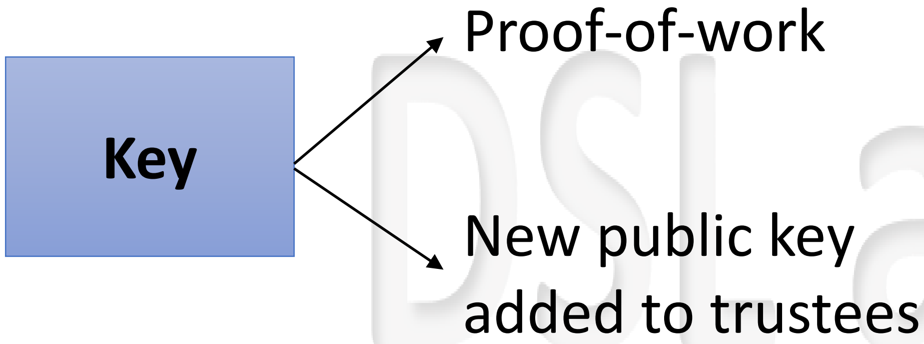
- 2 different kinds of blocks:



DSL at UCSB

Step 2: Decoupling Txn Verification from Leader Election

- 2 different kinds of blocks:

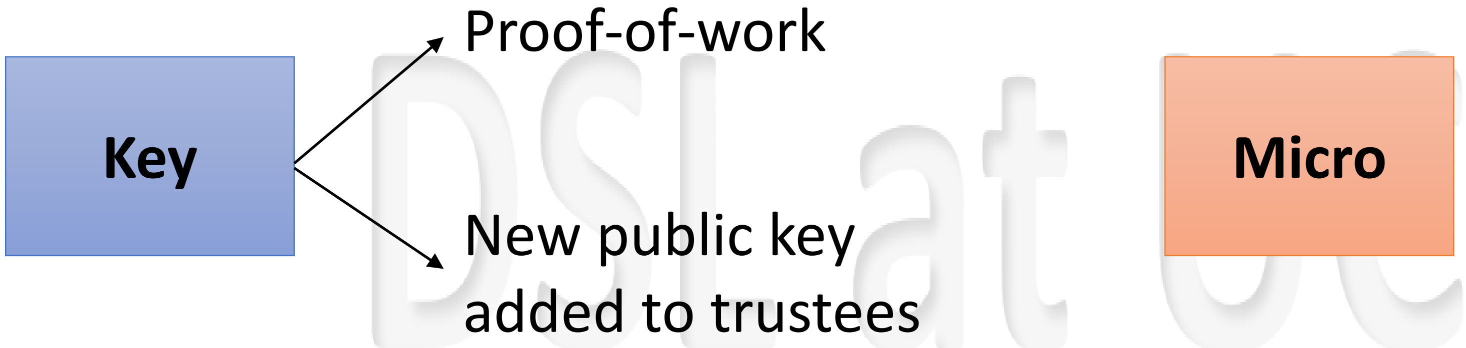


- Key blocks are created by mining PoW

DSL at UCSB

Step 2: Decoupling Txn Verification from Leader Election

- 2 different kinds of blocks:



DSL at UCSB

- Key blocks are created by mining PoW

Step 2: Decoupling Txn Verification from Leader Election

- 2 different kinds of blocks:

DSL at UCSB

- Key blocks are created by mining PoW

Step 2: Decoupling Txn Verification from Leader Election

- 2 different kinds of blocks:

- Key blocks are created by mining PoW

DSL at UCSB

Step 2: Decoupling Txn Verification from Leader Election

- 2 different kinds of blocks:

DSL at UCSB

- Key blocks are created by mining PoW
- PBFT is used to obtain consensus on Micro blocks

Step 2: Decoupling Txn Verification from Leader Election

- 2 different kinds of blocks:

DSL at UCSB

- Key blocks are created by mining PoW
- PBFT is used to obtain consensus on Micro blocks
- To avoid race condition, separate keyblock chain from microblock chain

Signing microblocks

DSL at UCSB

Signing microblocks

- Every microblock should be signed by a majority of current trustees

DSL at UCSB

Signing microblocks

- Every microblock should be signed by a majority of current trustees
- Byzcoin adapts a leader-based approach – **Collective Signing**

DSL at UCSB

Signing microblocks

- Every microblock should be signed by a majority of current trustees
- Byzcoin adapts a leader-based approach – **Collective Signing**
- Leader requests that statements be **publicly validated** and **co-signed** by decentralized group of witnesses

DSL at UCSB

Signing microblocks

- Every microblock should be signed by a majority of current trustees
- Byzcoin adapts a leader-based approach – **Collective Signing**
- Leader requests that statements be **publicly validated** and **co-signed** by decentralized group of witnesses
- Optimize **Schnorr** multi-signatures using **communication trees**

DSL at UCSB

Signing microblocks

- Every microblock should be signed by a majority of current trustees
- Byzcoin adapts a leader-based approach – **Collective Signing**
- Leader requests that statements be **publicly validated** and **co-signed** by decentralized group of witnesses
- Optimize **Schnorr** multi-signatures using **communication trees**
- Communication complexity: **O(N)**

DSL at UCSB

Signing microblocks

- Every microblock should be signed by a majority of current trustees
- Byzcoin adapts a leader-based approach – **Collective Signing**
- Leader requests that statements be **publicly validated** and **co-signed** by decentralized group of witnesses
- Optimize **Schnorr** multi-signatures using **communication trees**
- Communication complexity: **O(N)**

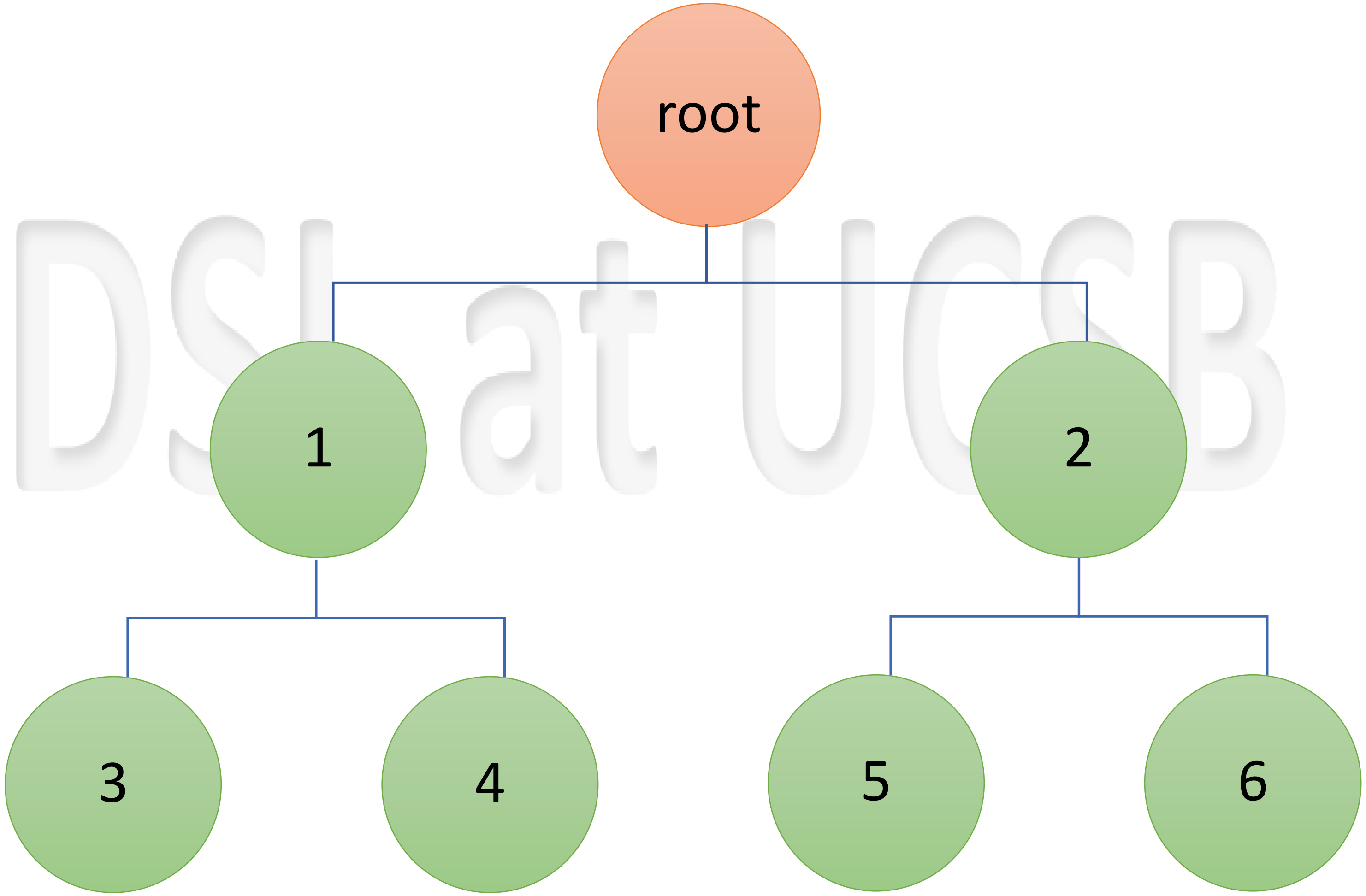
DSL at UCSB

SYTA, E., TAMAS, I., VISHER, D., WOLINSKY, D. I., L., GAILLY, N., KHOFFI, I., AND FORD, B. Keeping Authorities "Honest or Bust" with Decentralized Witness Cosigning. In 37th IEEE Symposium on Security and Privacy (May 2016).

Step 3: Scaling PBFT using Collective Signing

DSL at UCSB

Step 3: Scaling PBFT using Collective Signing

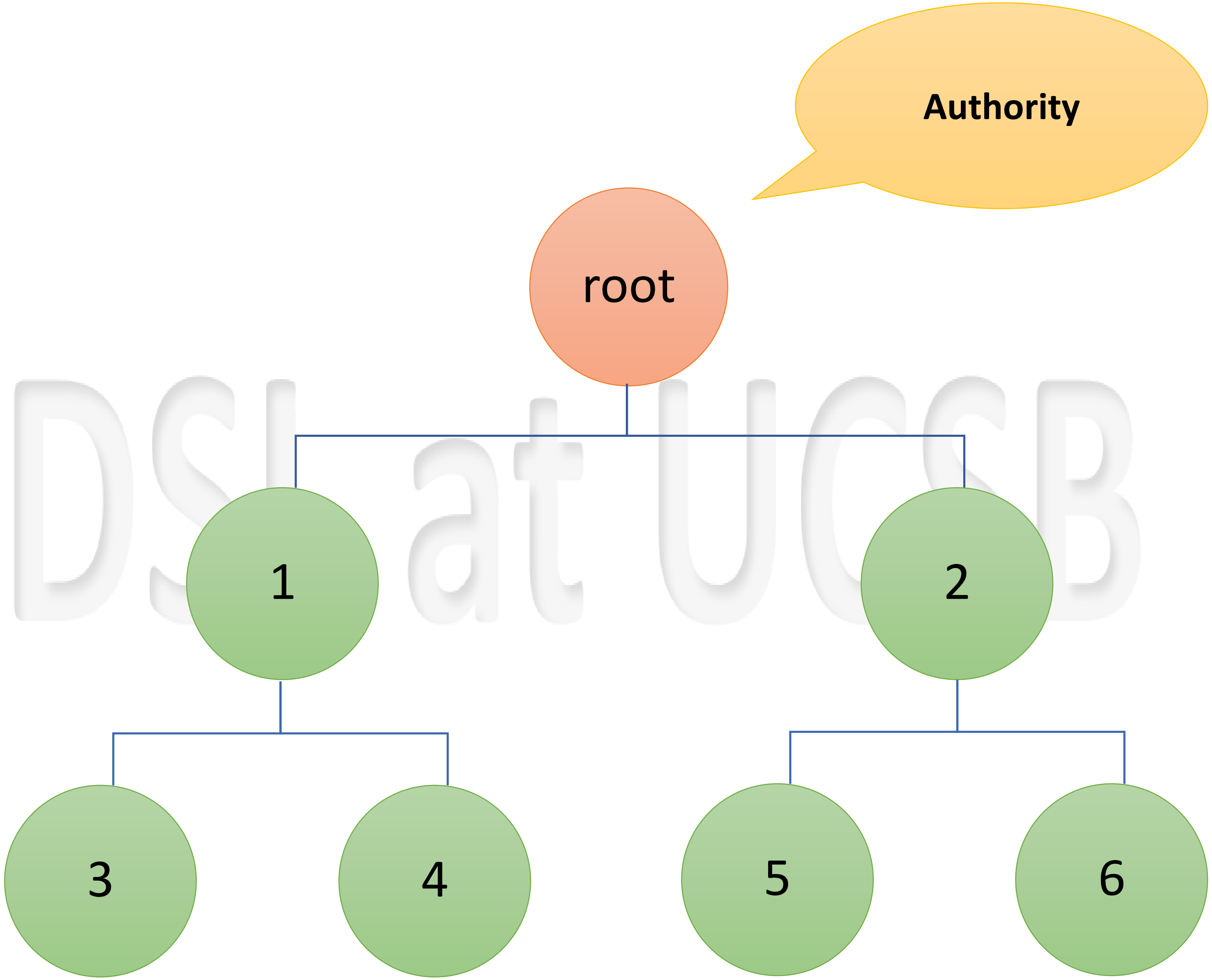


DSL at UCSB

DSL

Step 3: Scaling PBFT using Collective Signing

UCSB

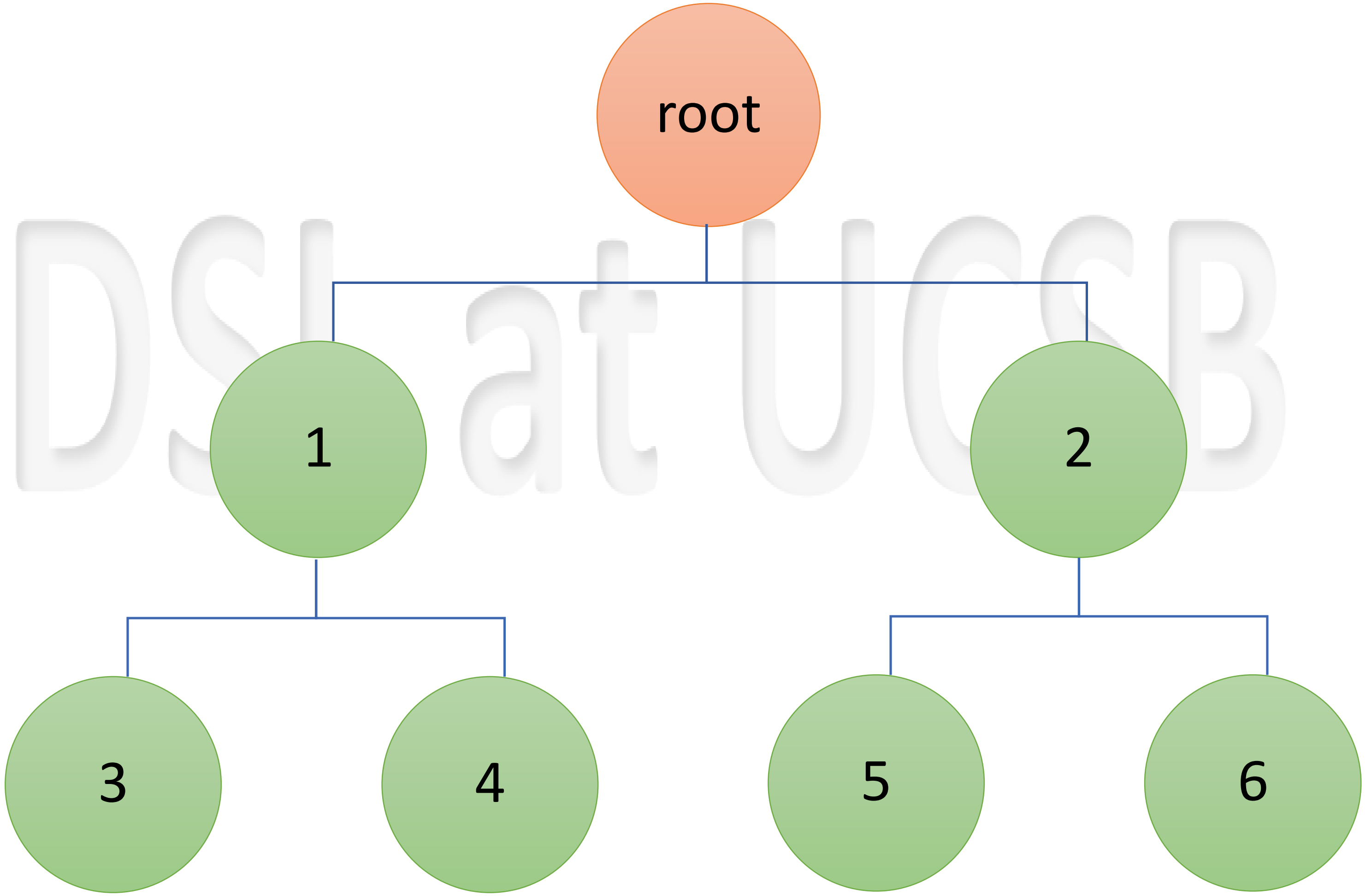


DSL at UCSB

DSL

Step 3: Scaling PBFT using Collective Signing

UCSB

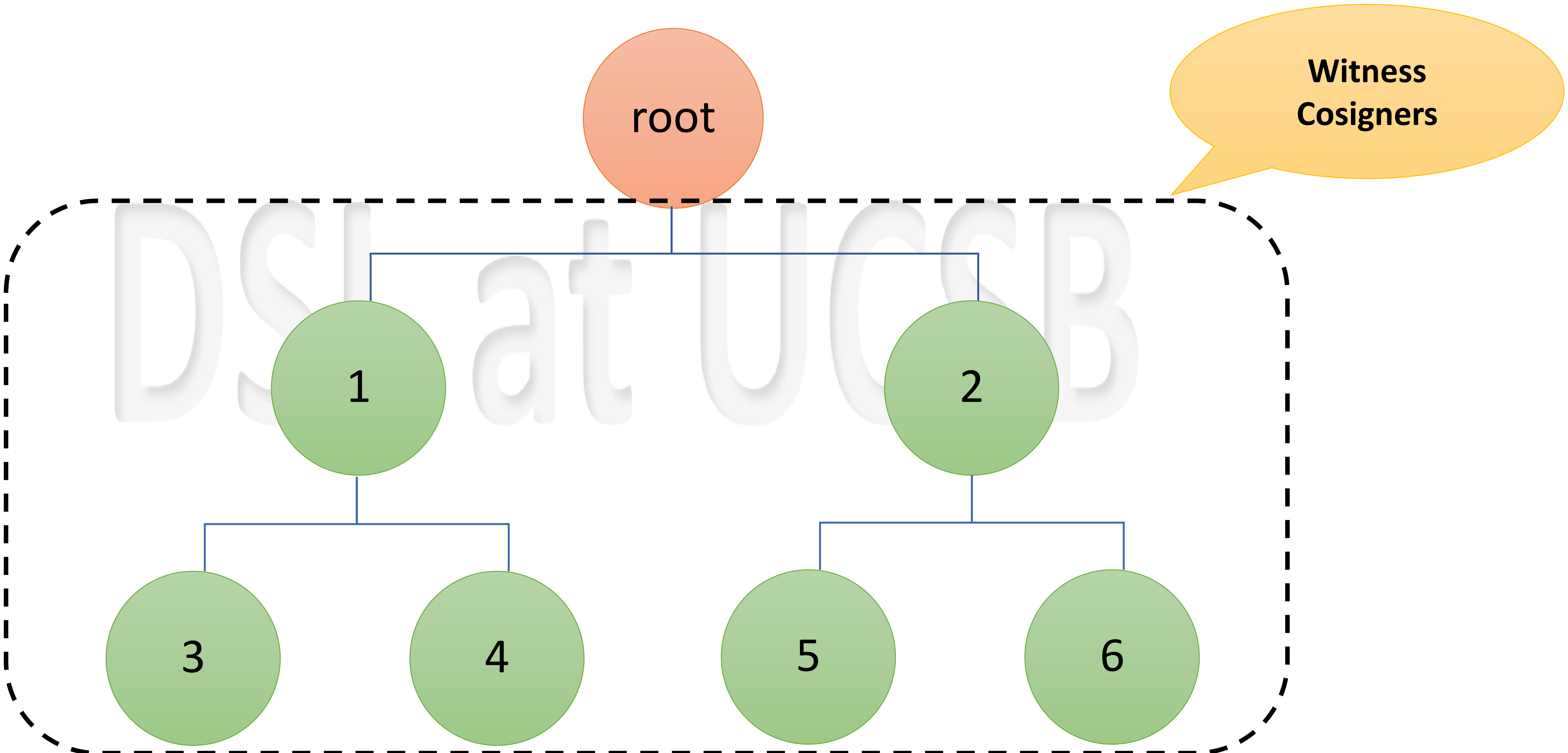


DSL at UCSB

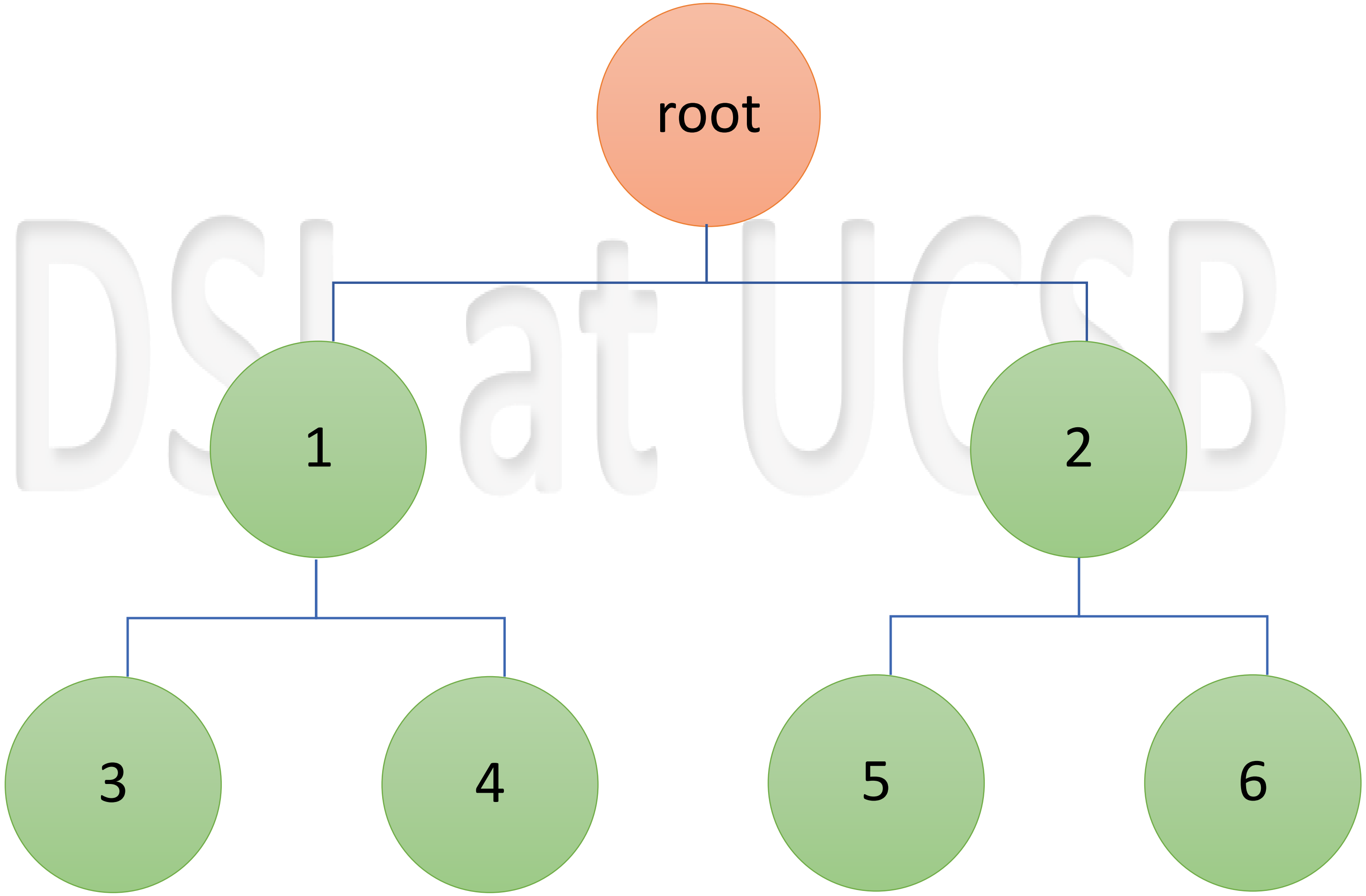
DSL

Step 3: Scaling PBFT using Collective Signing

UCSB



Step 3: Scaling PBFT using Collective Signing

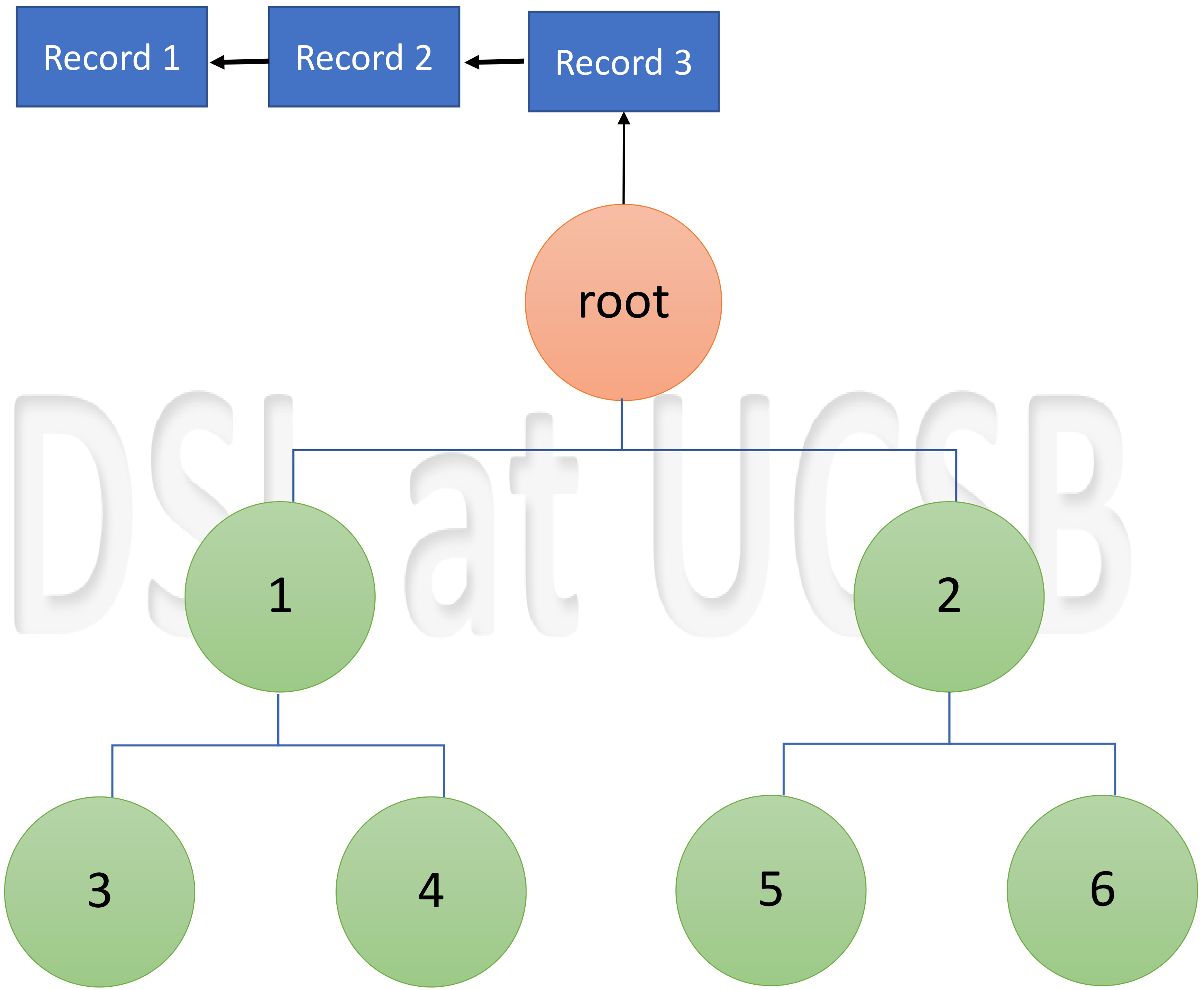


DSL at UCSB

DSL

Step 3: Scaling PBFT using Collective Signing

UCSB

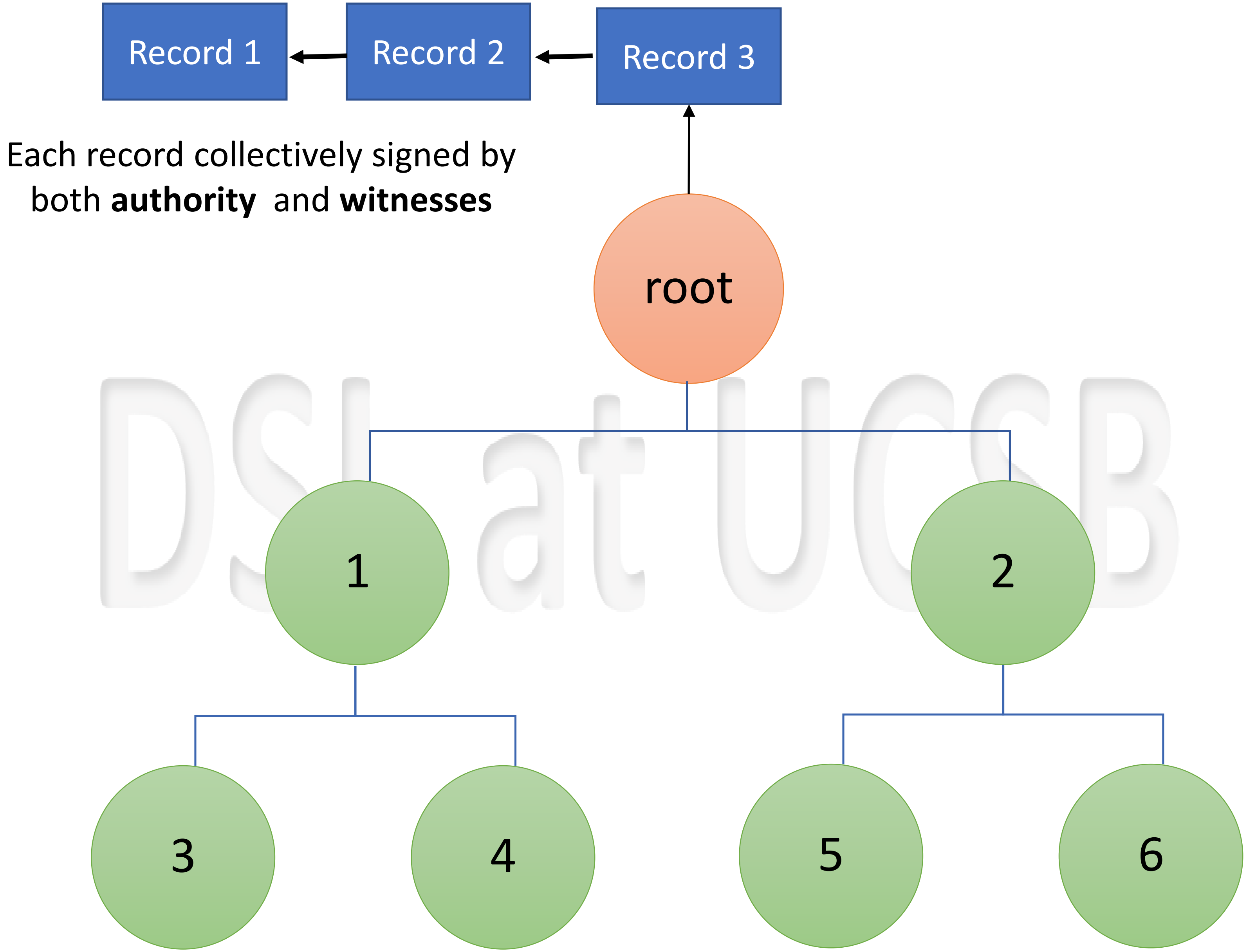


DSL at UCSB

DSL

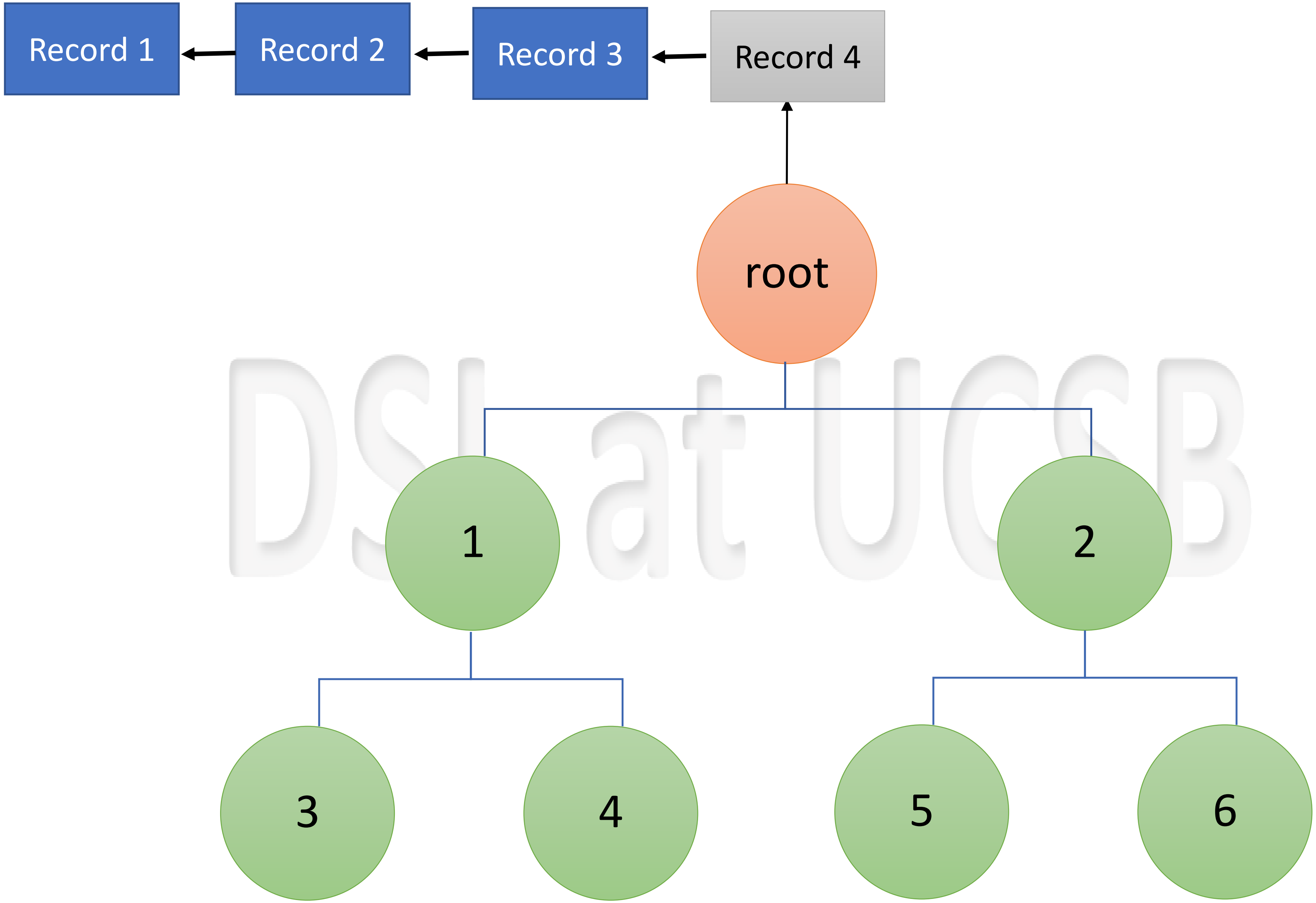
Step 3: Scaling PBFT using Collective Signing

UCSB

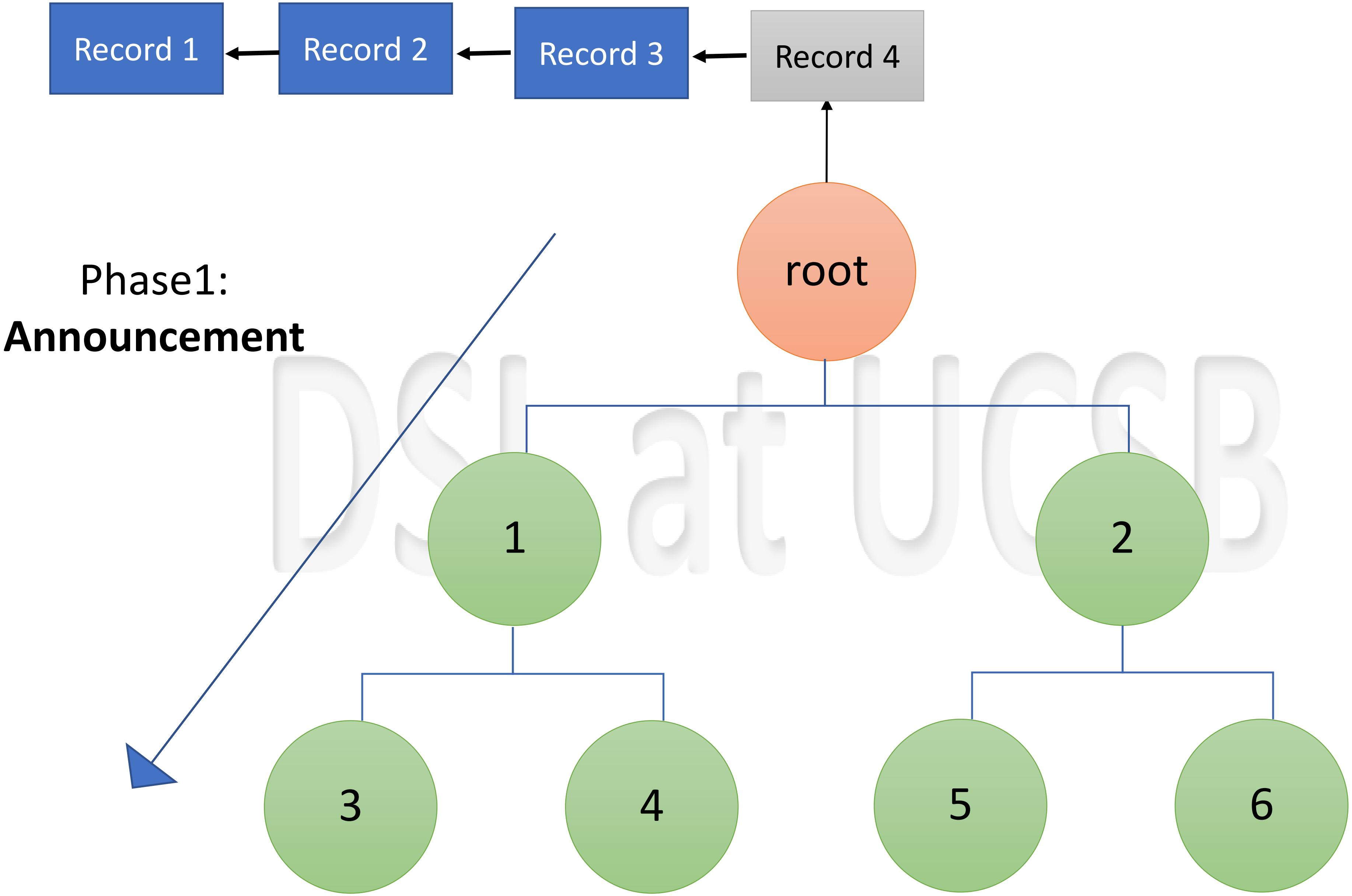


DSL at UCSB

CoSi – Collective Signing



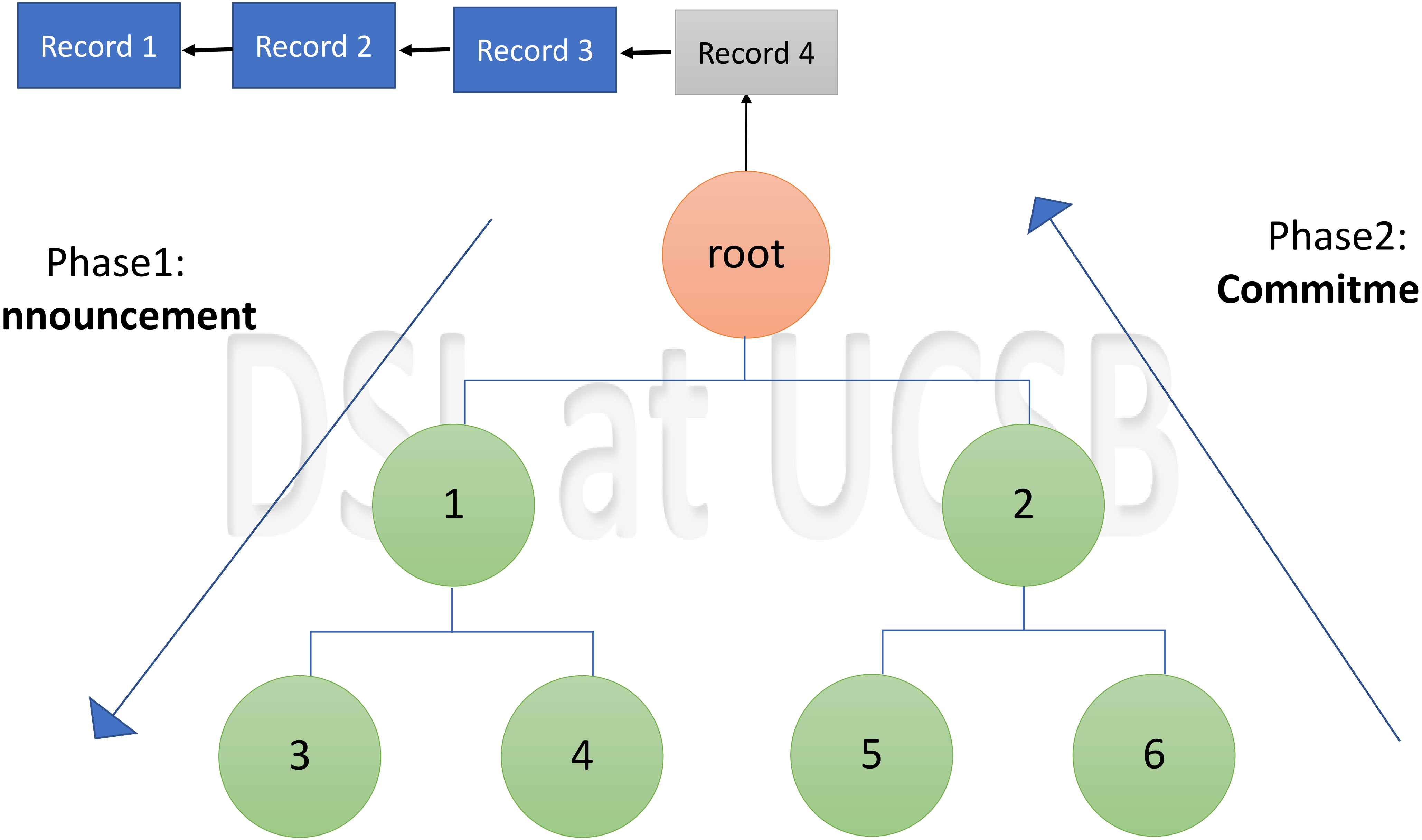
CoSi – Collective Signing



DSL

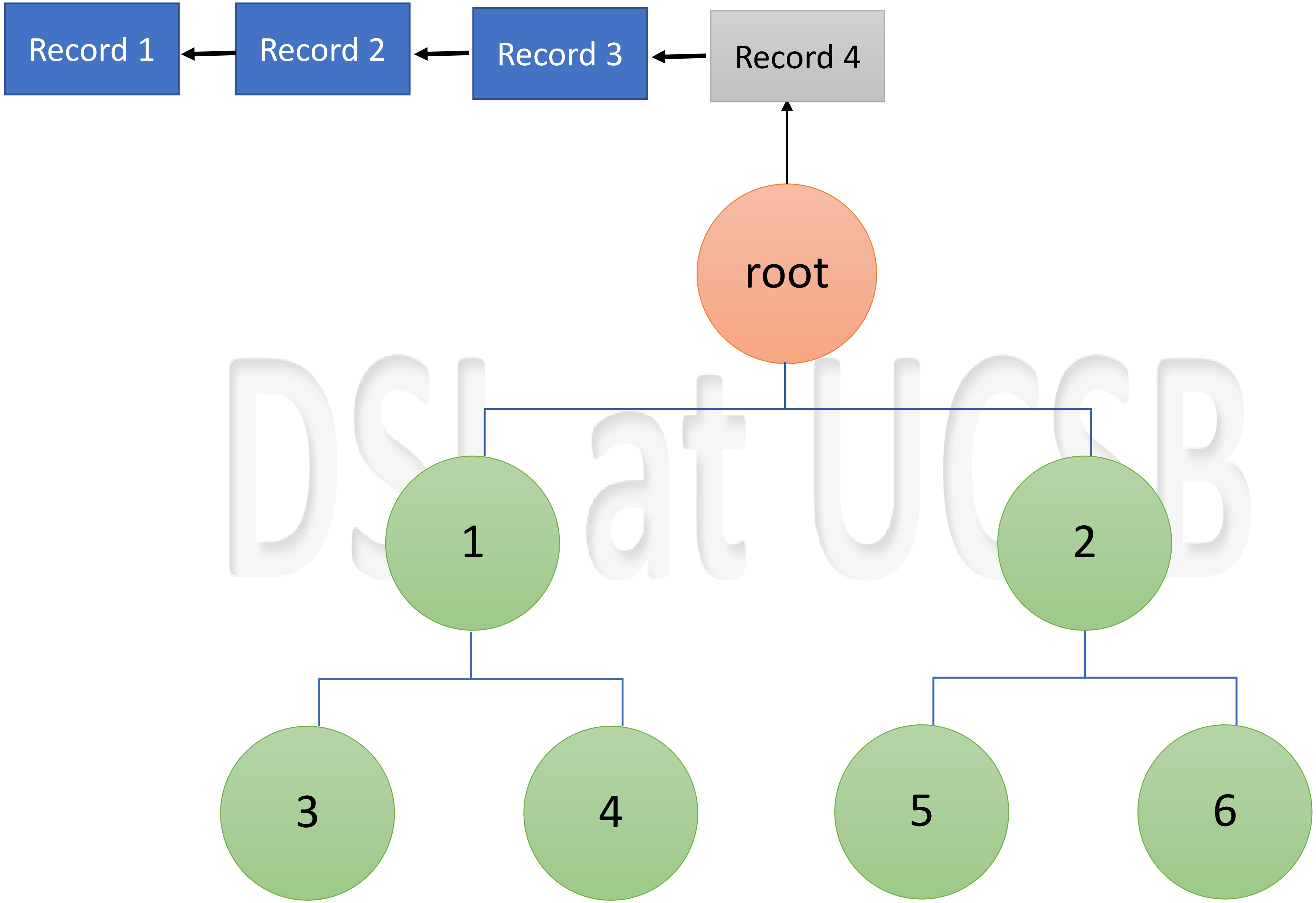
CoSi – Collective Signing

UCSB



DSL at UCSB

CoSi – Collective Signing

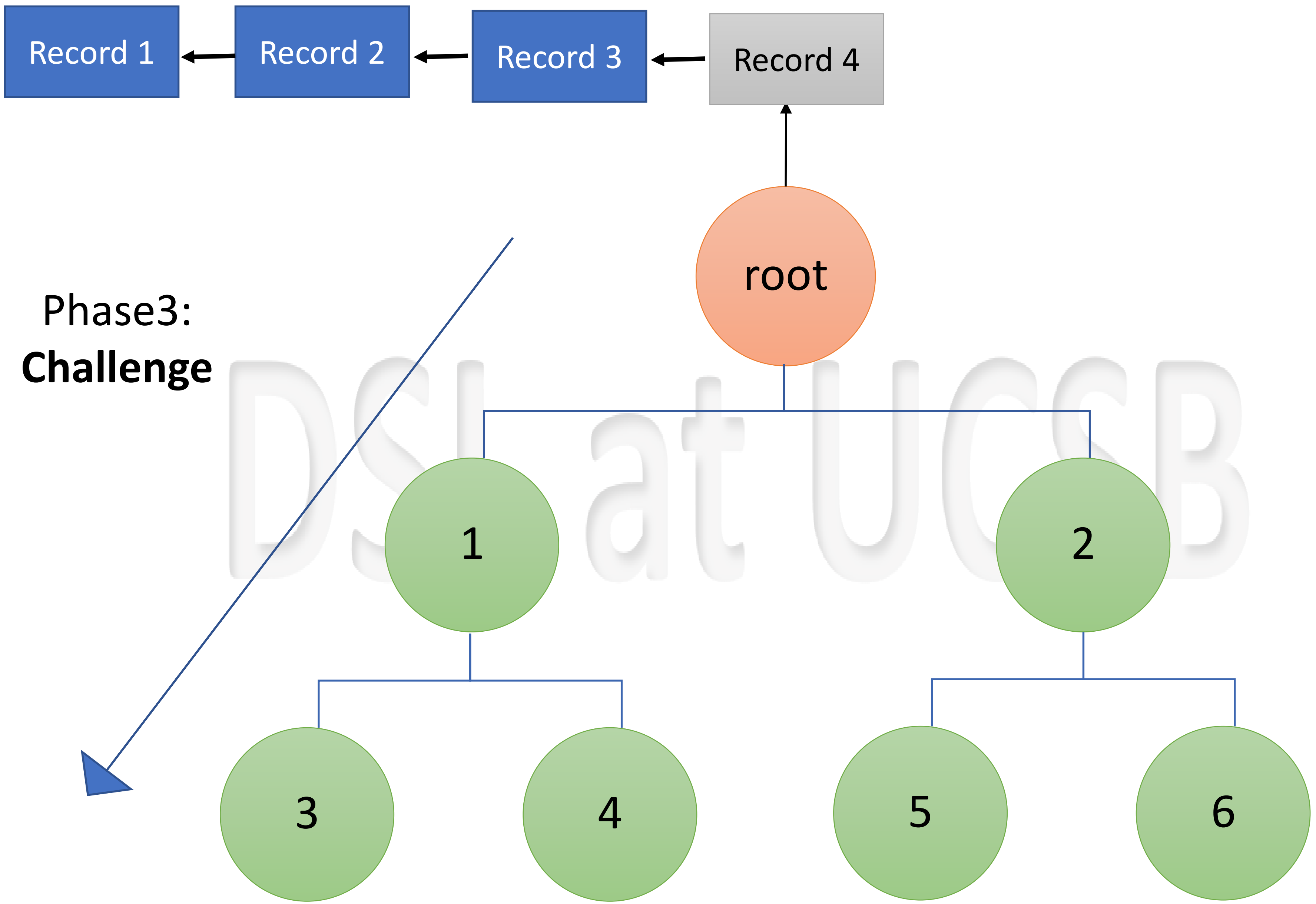


DSL at UCSB

DSL

CoSi – Collective Signing

UCSB

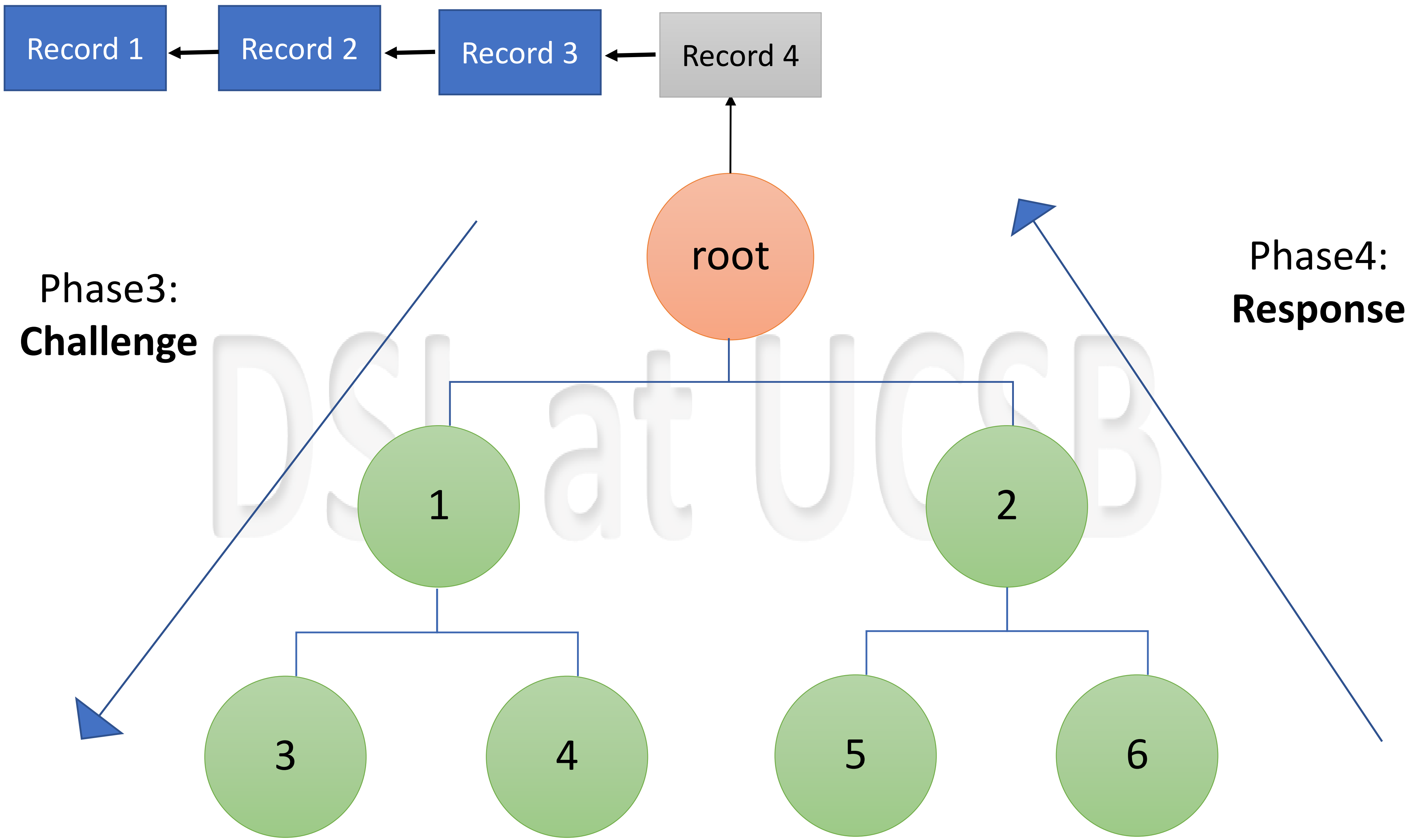


DSL at UCSB

DSL

CoSi – Collective Signing

UCSB



DSL at UCSB

Step 4: Using CoSi to achieve PBFT

DSL at UCSB

Step 4: Using CoSi to achieve PBFT

Announcement

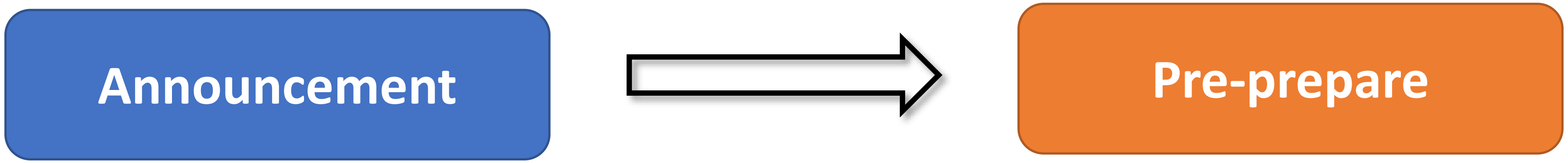
DSL at UCSB

Step 4: Using CoSi to achieve PBFT

Announcement →

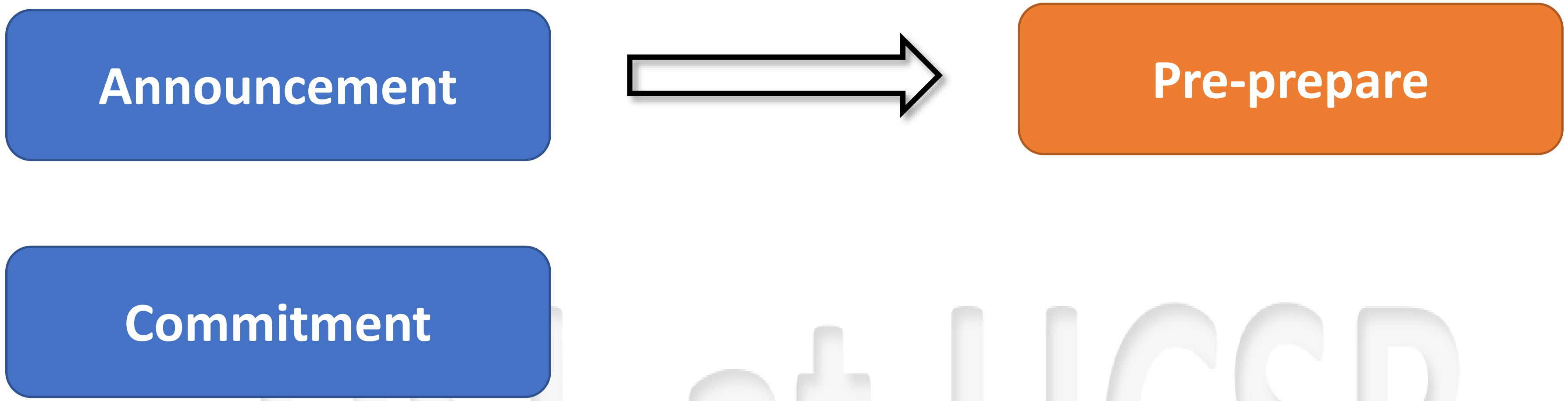
DSL at UCSB

Step 4: Using CoSi to achieve PBFT



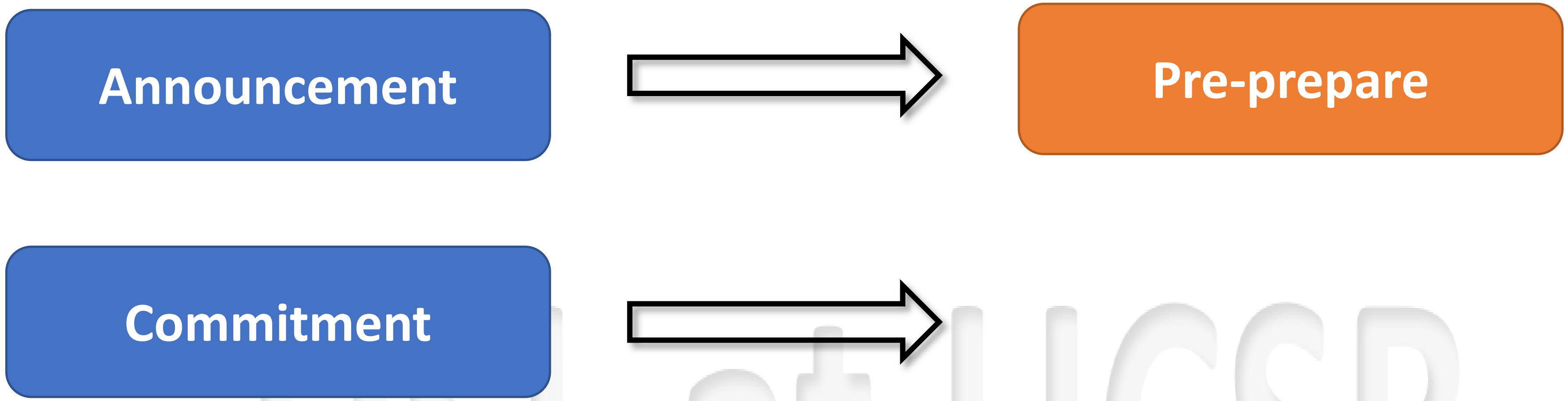
DSL at UCSB

Step 4: Using CoSi to achieve PBFT



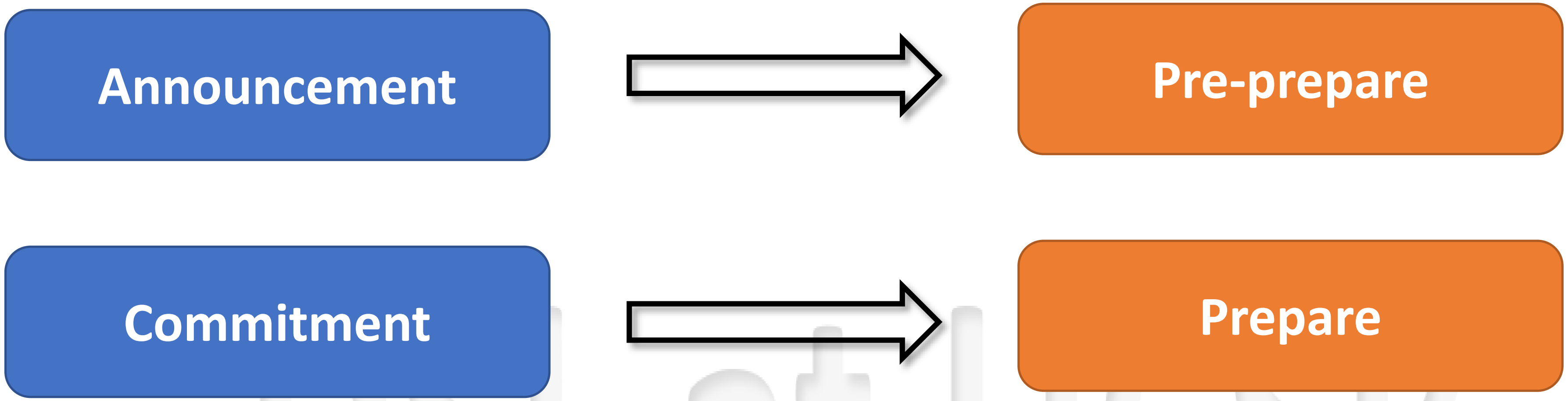
DSL at UCSB

Step 4: Using CoSi to achieve PBFT



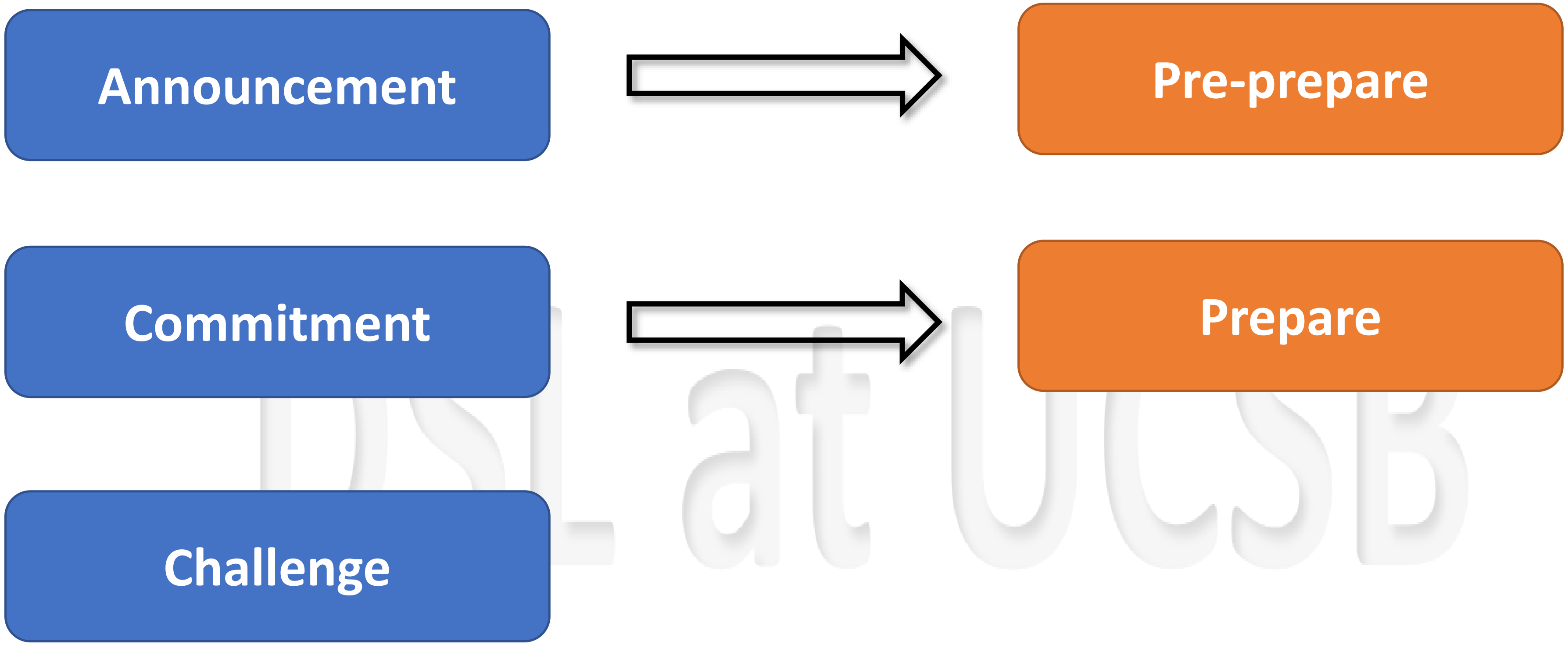
DSL at UCSB

Step 4: Using CoSi to achieve PBFT



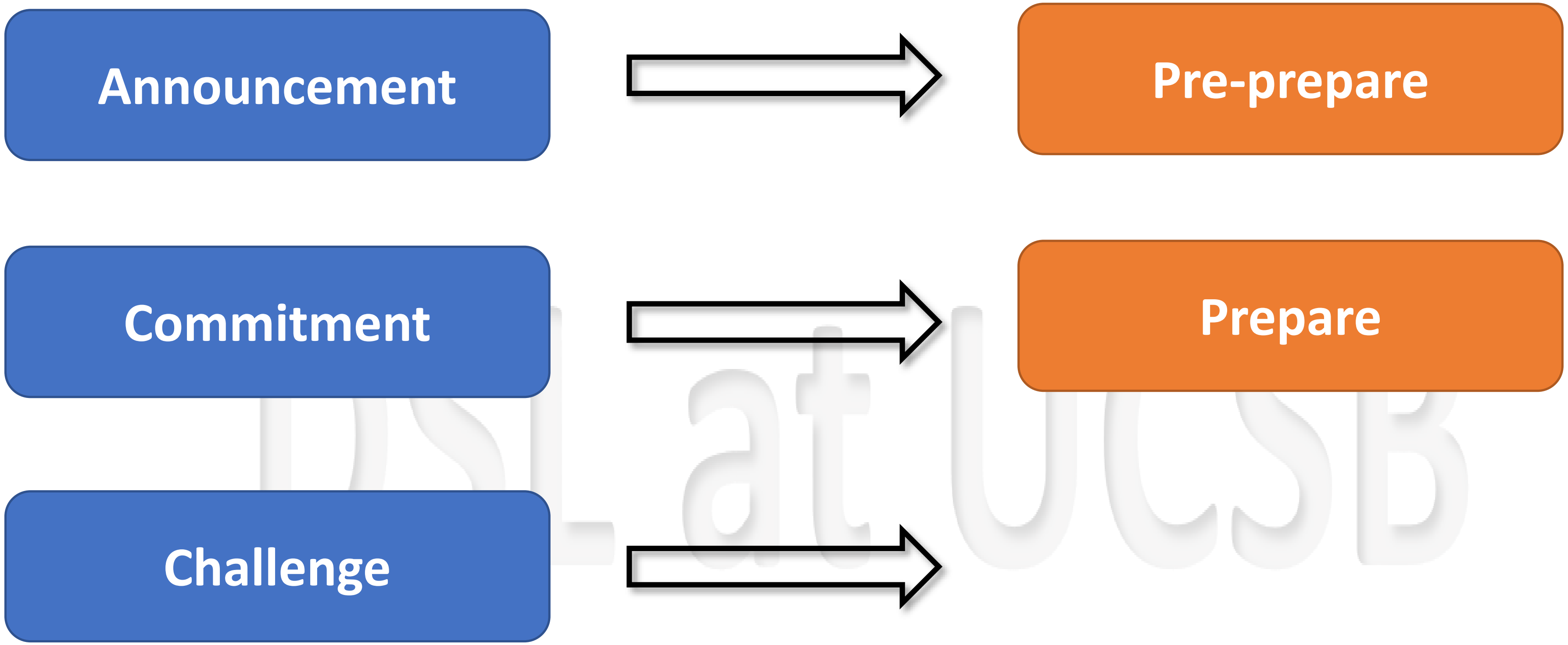
DSL at UCSB

Step 4: Using CoSi to achieve PBFT



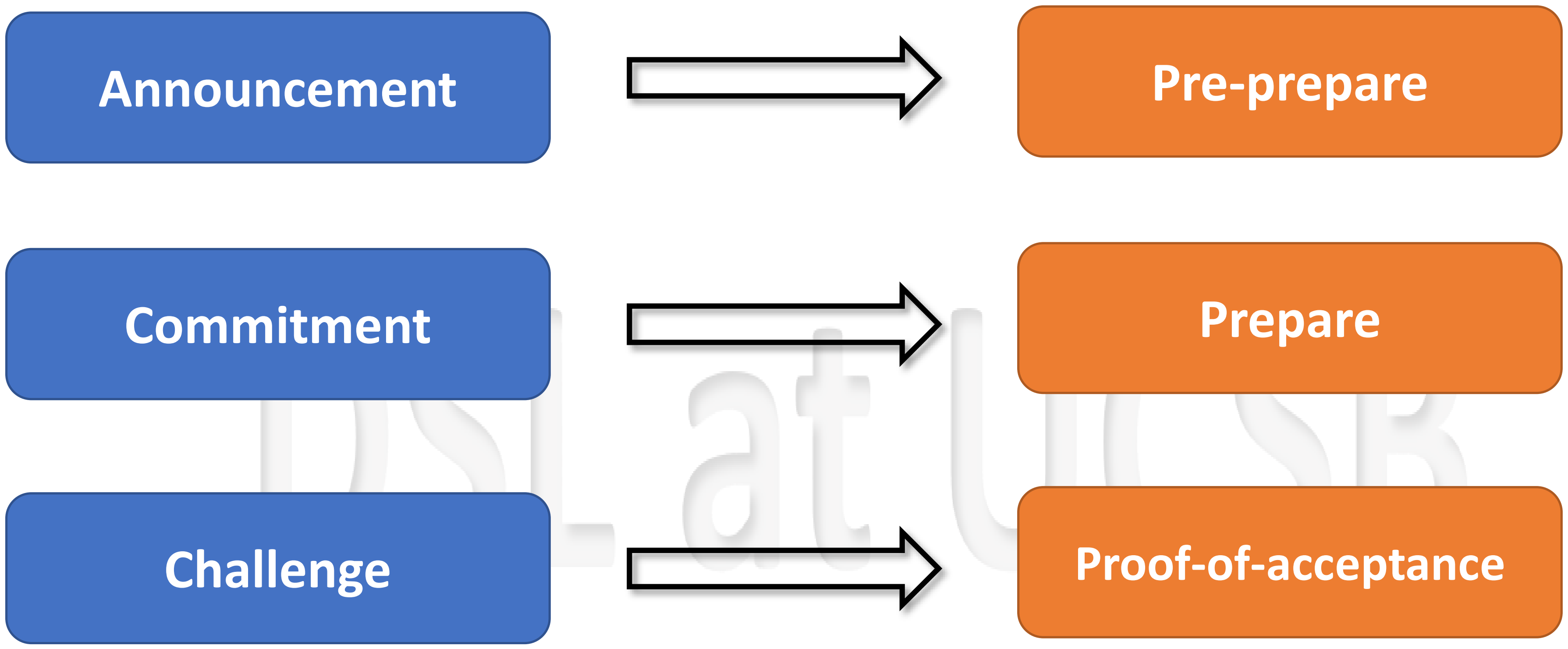
DSL at UCSB

Step 4: Using CoSi to achieve PBFT



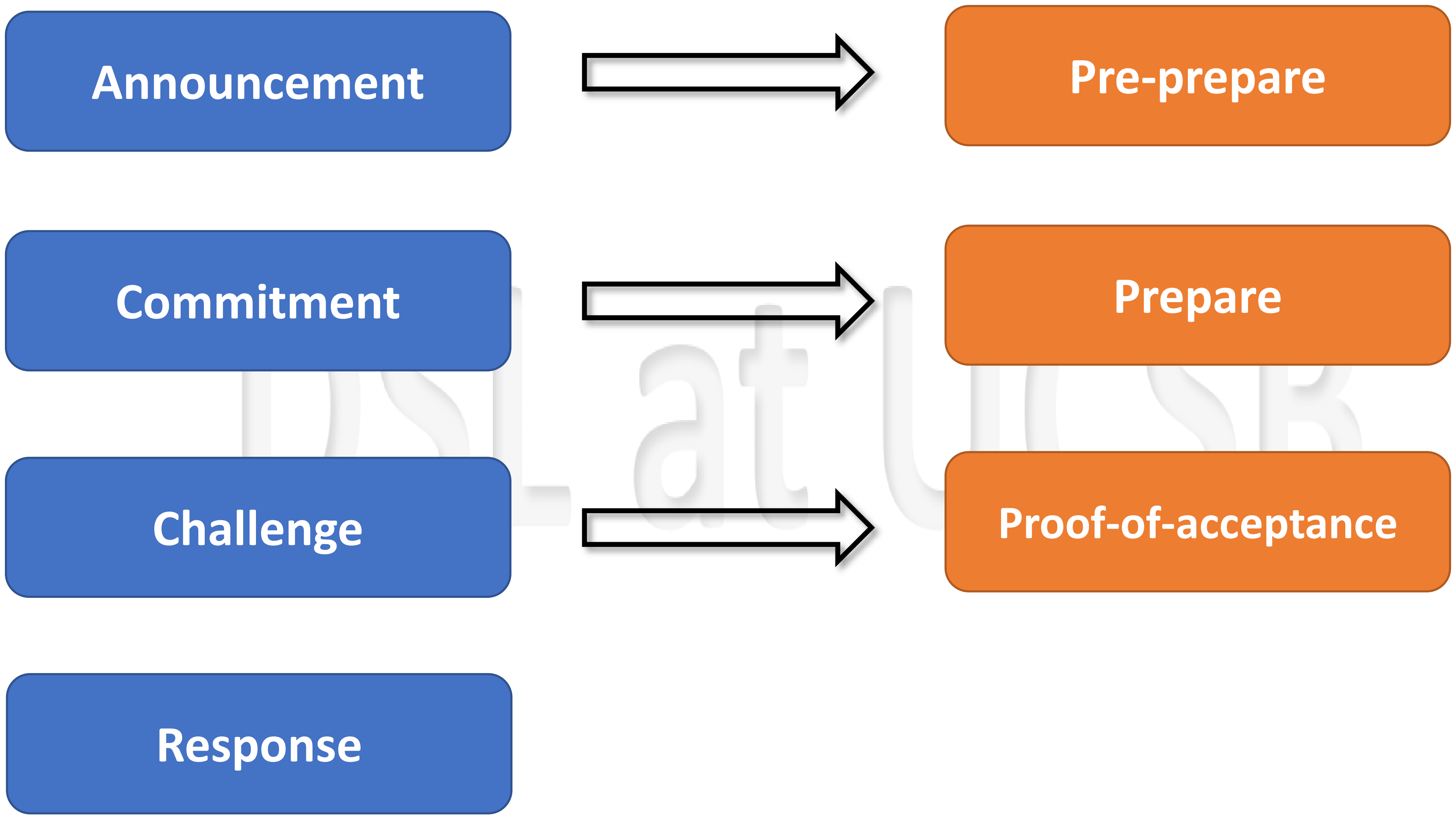
DSL at UCSB

Step 4: Using CoSi to achieve PBFT



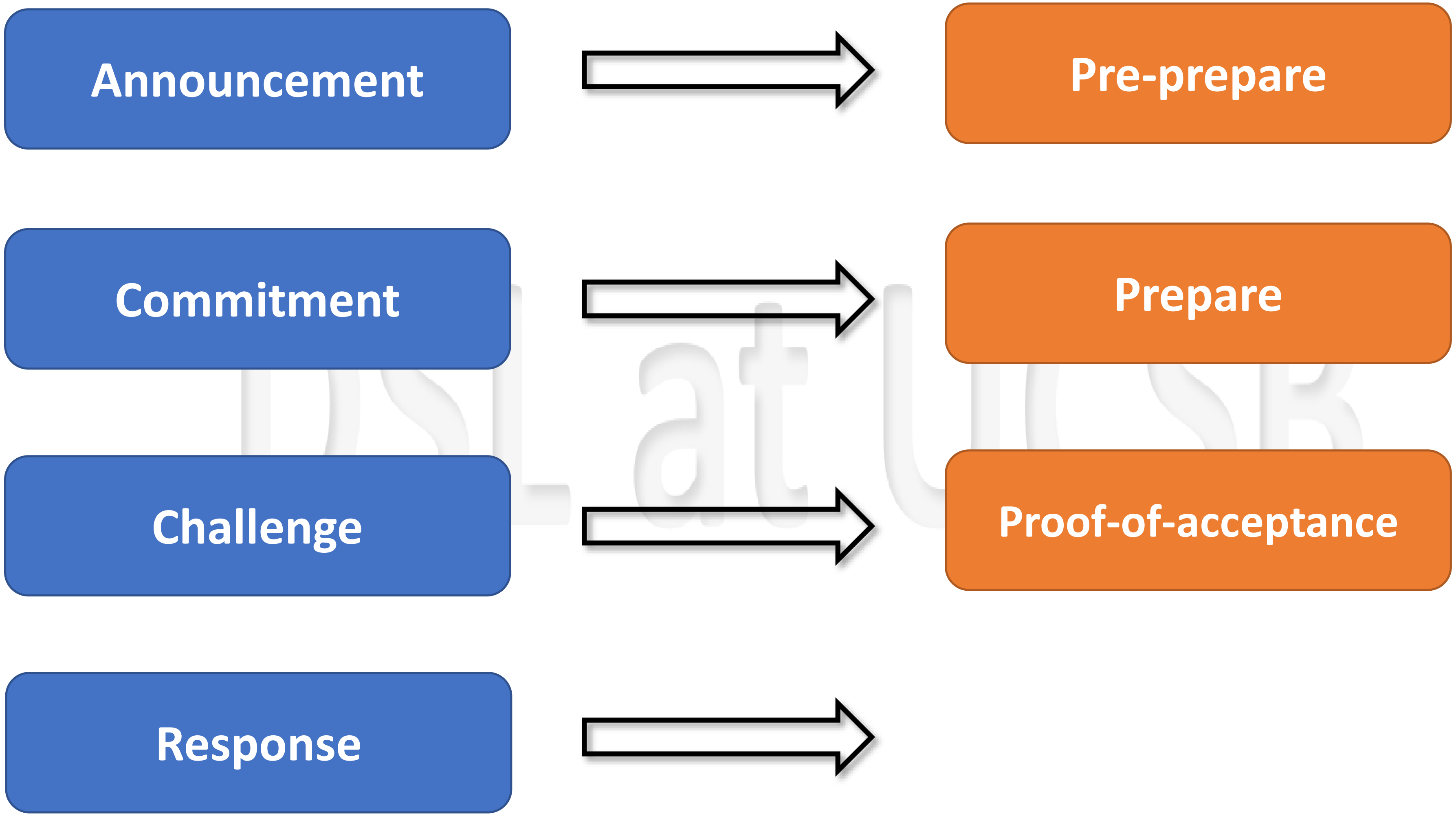
DSL at UCSB

Step 4: Using CoSi to achieve PBFT



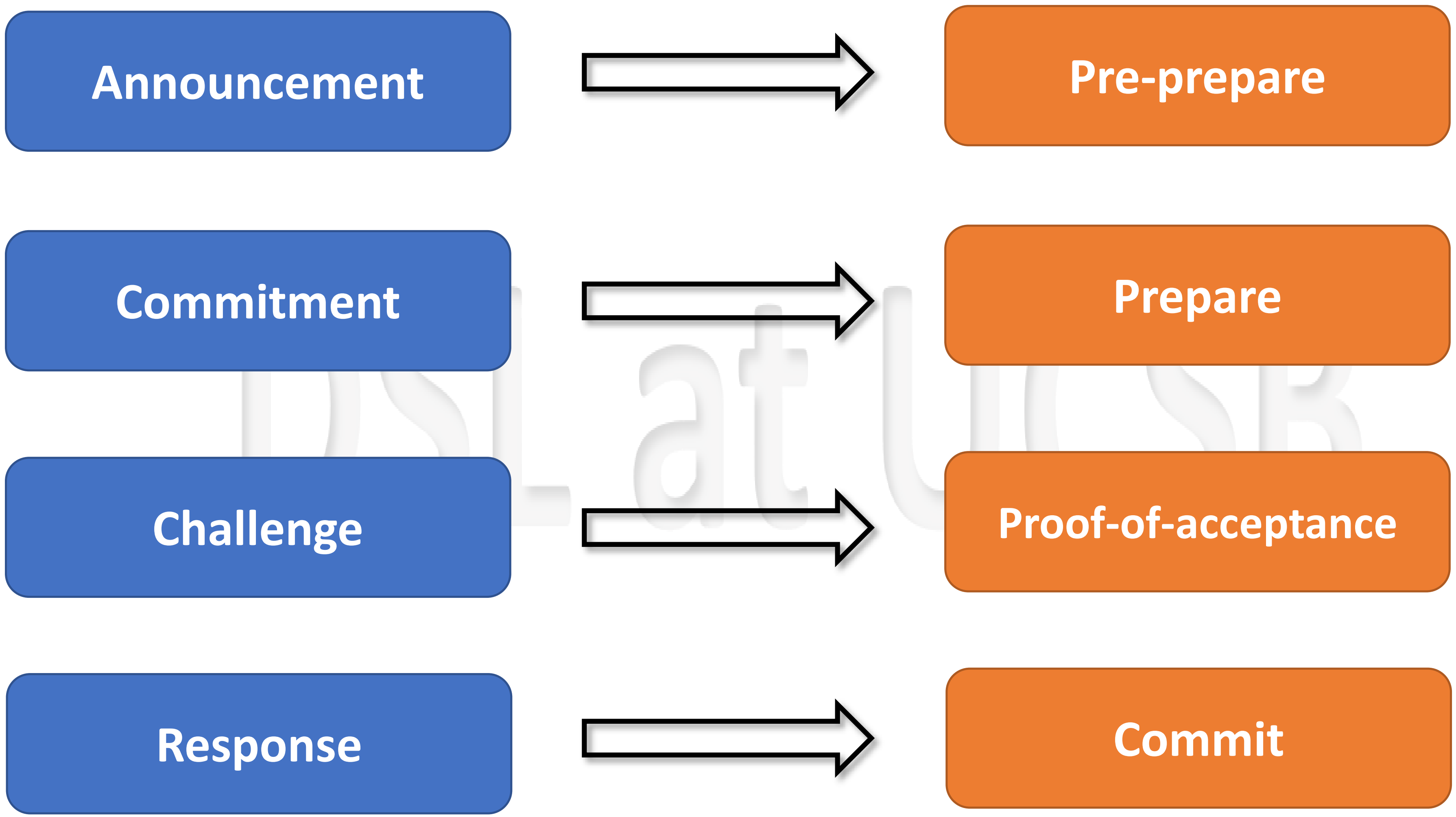
DSL at UCSB

Step 4: Using CoSi to achieve PBFT



DSL at UCSB

Step 4: Using CoSi to achieve PBFT



DSL at UCSB

Step 4: Using CoSi to achieve PBFT

DSL at UCSB

Step 4: Using CoSi to achieve PBFT

- **PBFT** is made **scalable** to thousands of nodes by clubbing with **CoSi**

DSL at UCSB

Step 4: Using CoSi to achieve PBFT

- **PBFT** is made **scalable** to thousands of nodes by clubbing with **CoSi**
- Need **two-third** super majority signatures in each phase

DSL at UCSB

Step 4: Using CoSi to achieve PBFT

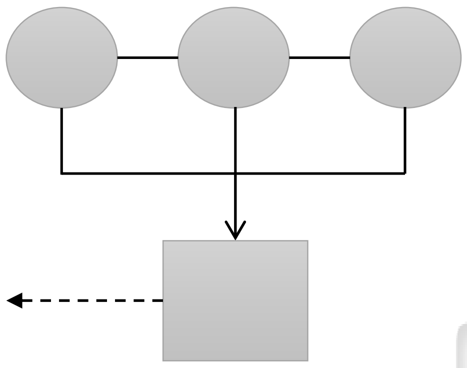
- **PBFT** is made **scalable** to thousands of nodes by clubbing with **CoSi**
- Need **two-third** super majority signatures in each phase
- **Double spending** by malicious leader circumvented due to **overlap** in the two phases on CoSi

DSL at UCSB

ByzCoin design

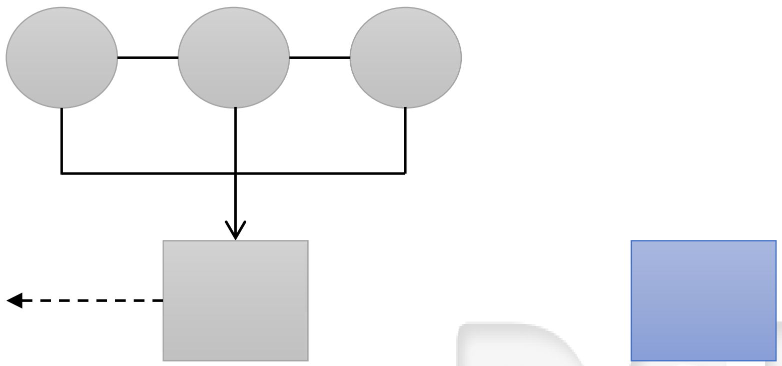
DSL at UCSB

ByzCoin design



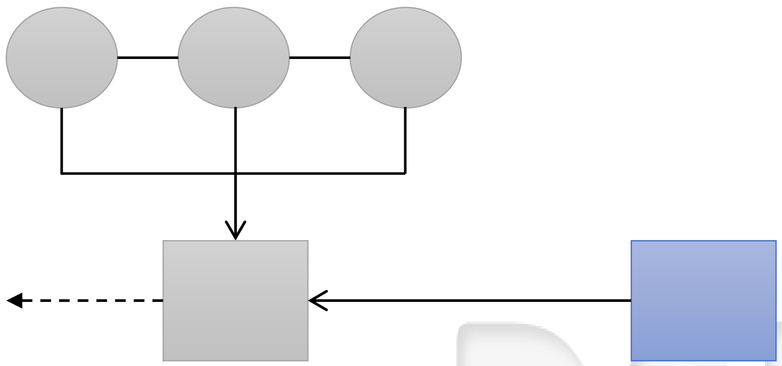
DSL at UCSB

ByzCoin design



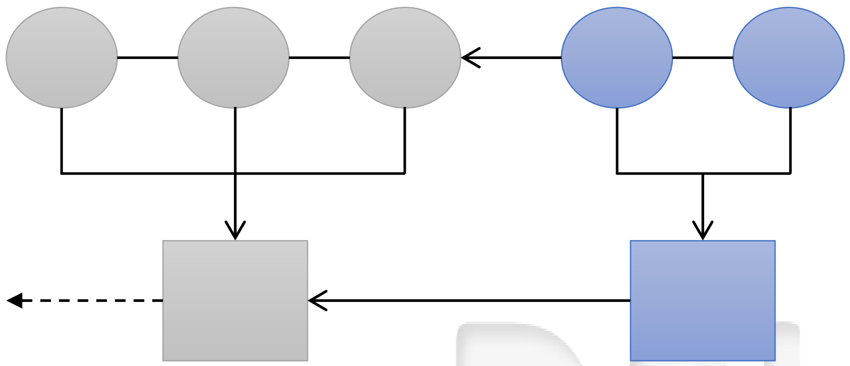
DSL at UCSB

ByzCoin design



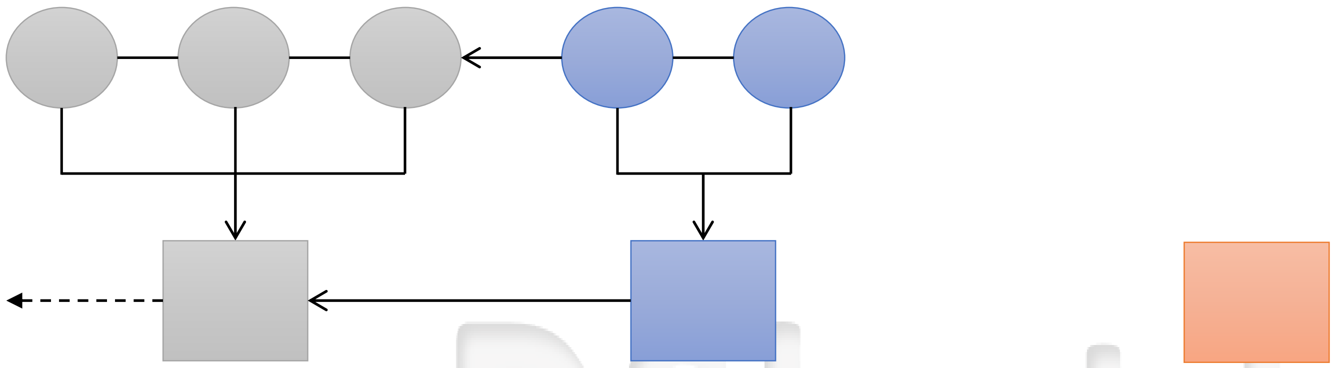
DSL at UCSB

ByzCoin design



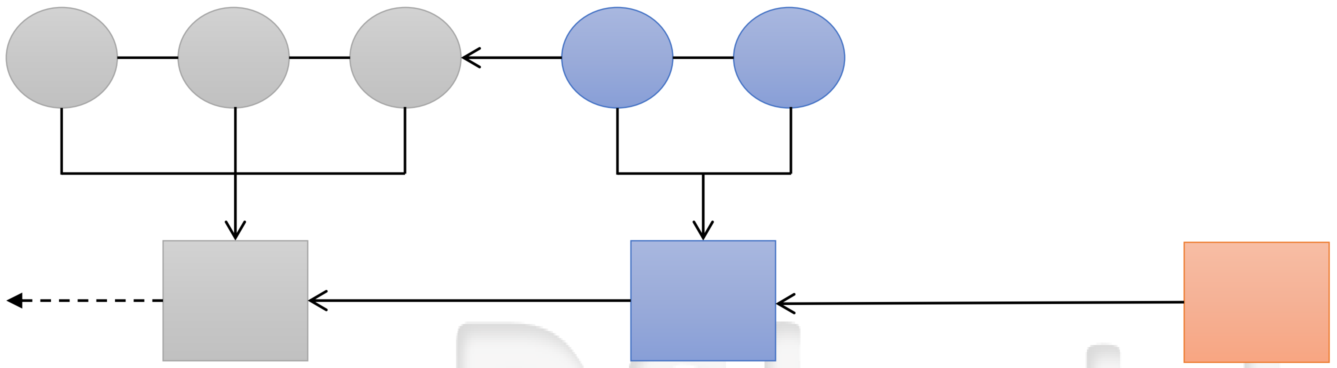
DSL at UCSB

ByzCoin design



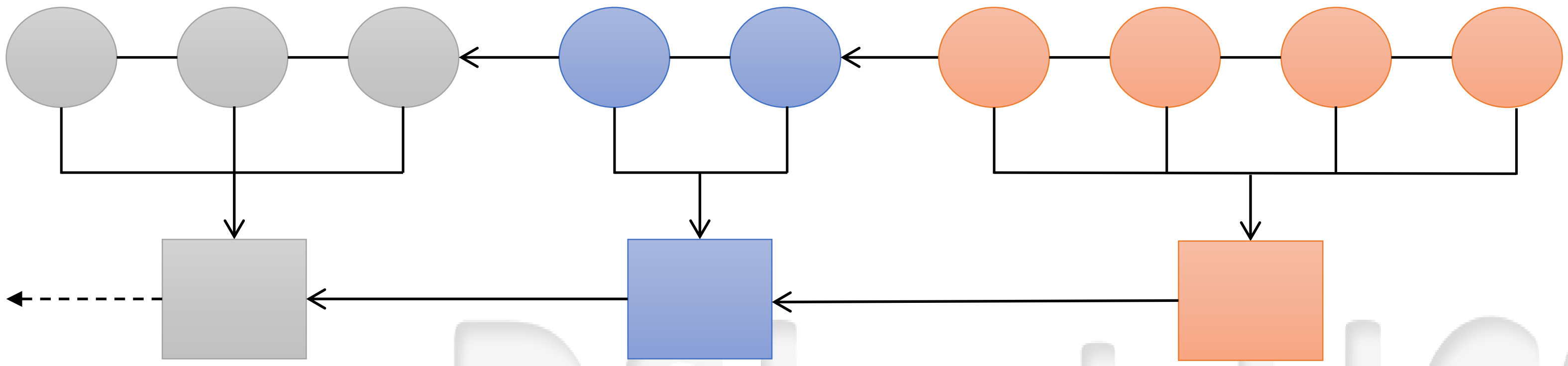
DSL at UCSB

ByzCoin design



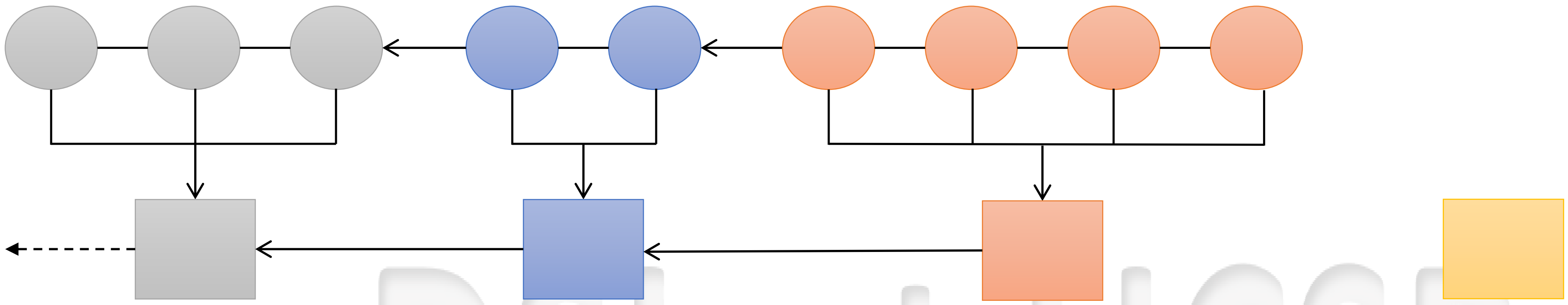
DSL at UCSB

ByzCoin design



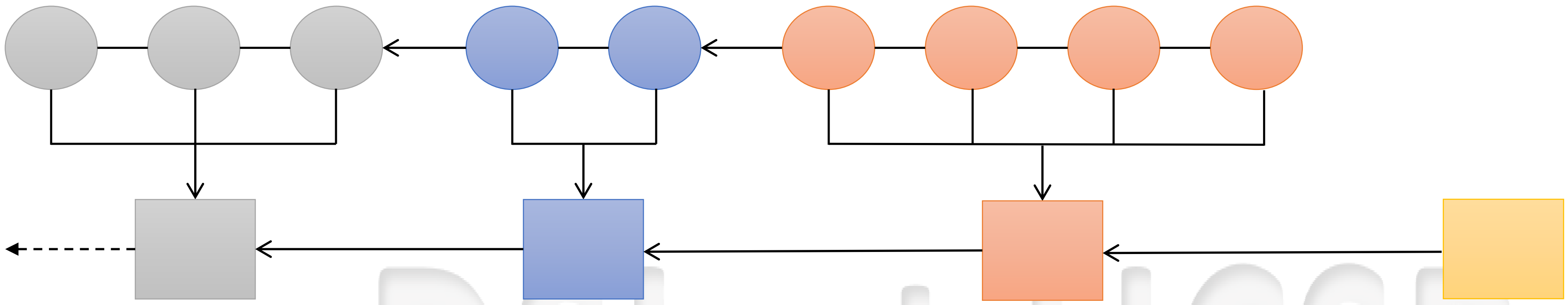
DSL at UCSB

ByzCoin design



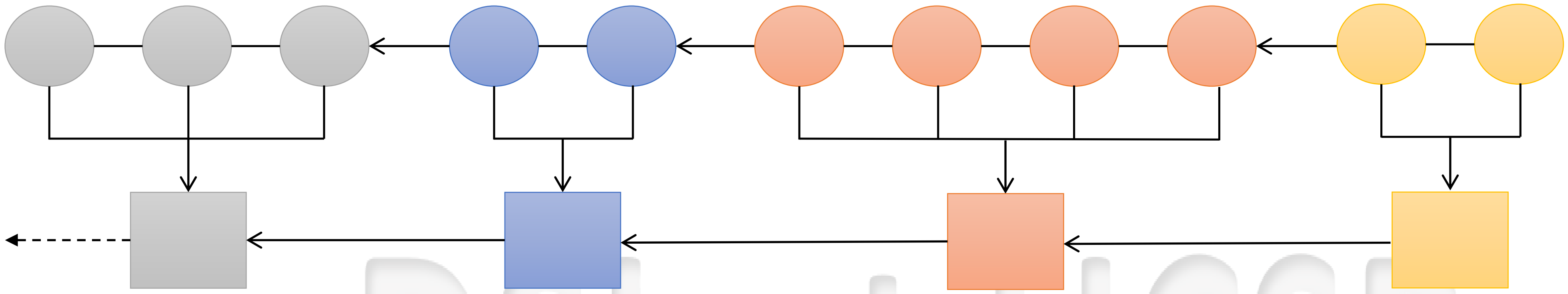
DSL at UCSB

ByzCoin design



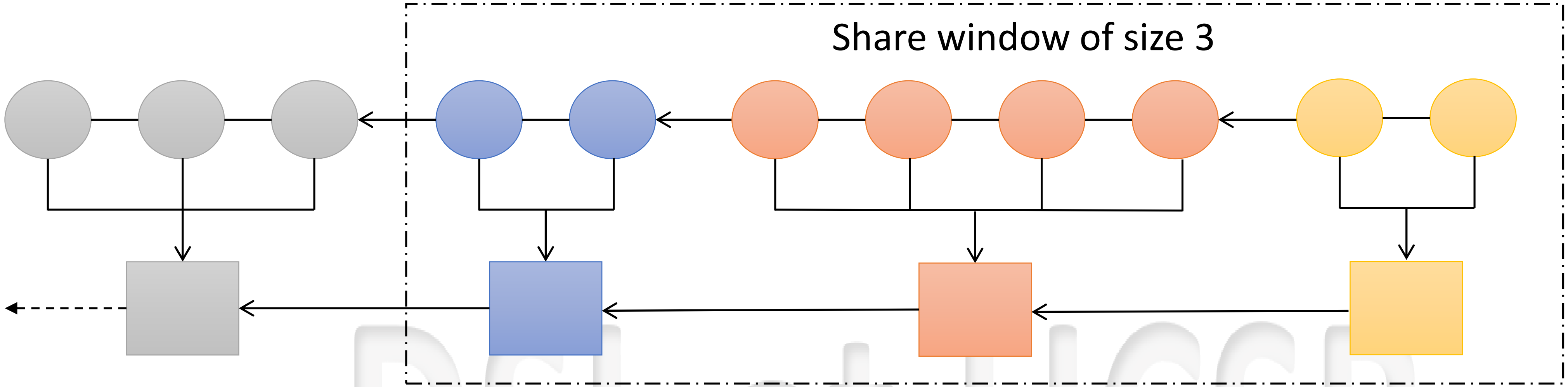
DSL at UCSB

ByzCoin design



DSL at UCSB

ByzCoin design

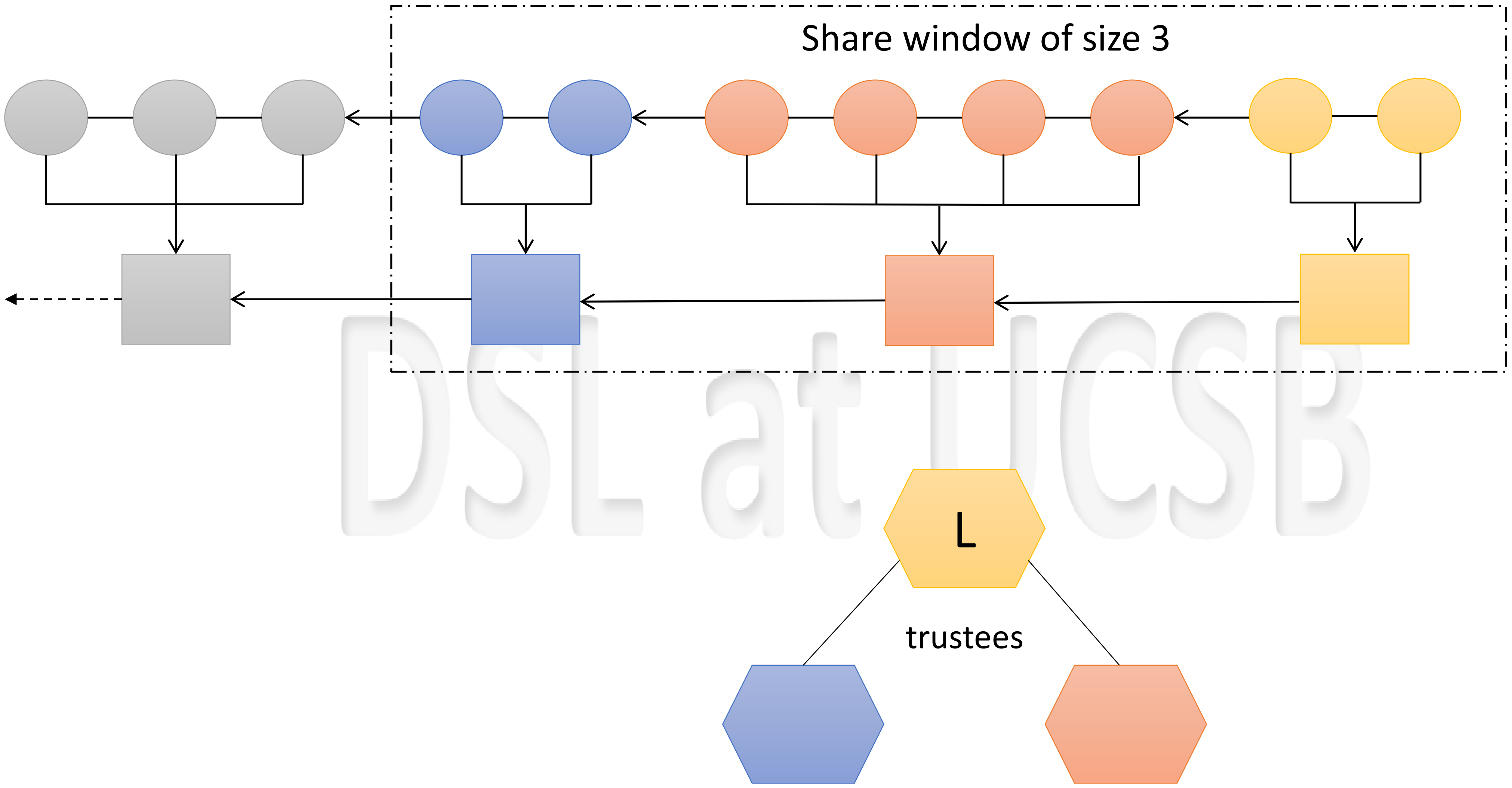


DSL at UCSB

DSL

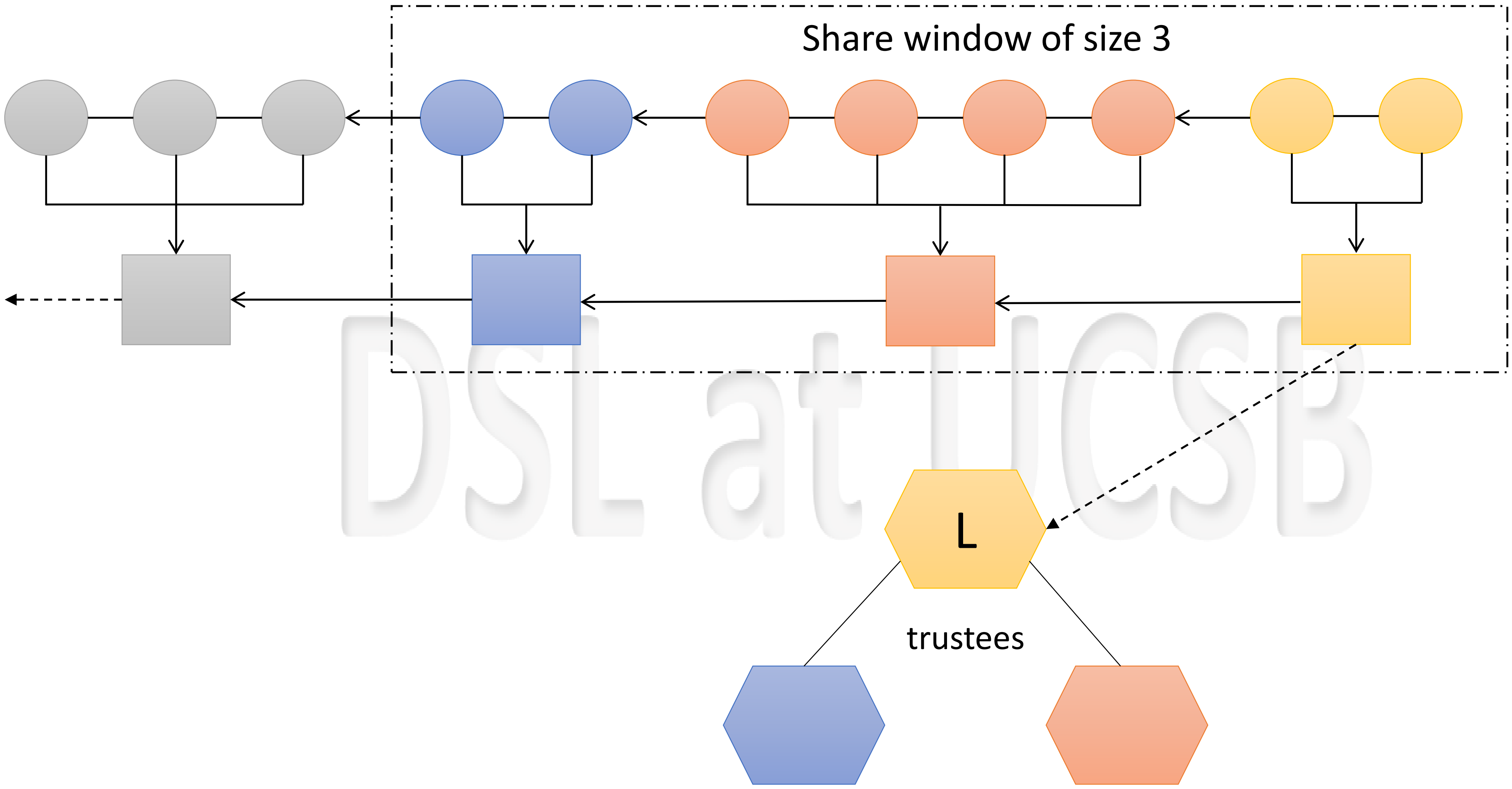
UCSB

ByzCoin design



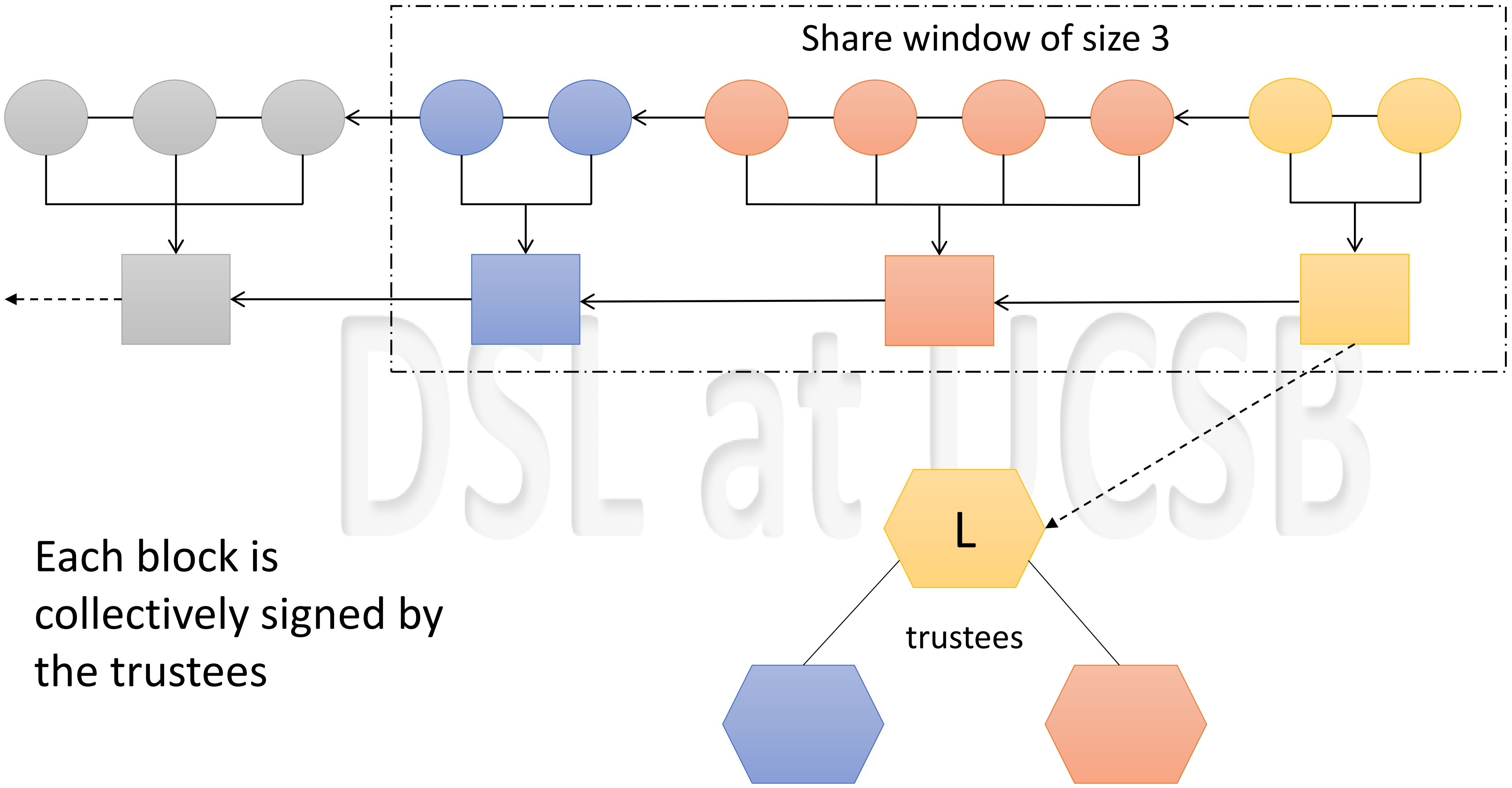
DSL at UCSB

ByzCoin design



DSL at UCSB

ByzCoin design



Each block is collectively signed by the trustees

DSL at UCSB

DSL

UCSB

Dealing with Keyblock conflicts and Selfish Mining

DSL at UCSB

Dealing with Keyblock conflicts and Selfish Mining

- Forks in microblock chain not possible due to **PBFT**

DSL at UCSB

Dealing with Keyblock conflicts and Selfish Mining

- Forks in microblock chain not possible due to **PBFT**
- But **forks** possible in **keyblock** chain

DSL at UCSB

Dealing with Keyblock conflicts and Selfish Mining

- Forks in microblock chain not possible due to **PBFT**
- But **forks** possible in **keyblock** chain

How to resolve keyblock conflicts?

DSL at UCSB

Dealing with Keyblock conflicts and Selfish Mining

- Forks in microblock chain not possible due to **PBFT**
- But **forks** possible in **keyblock** chain

How to resolve keyblock conflicts?

- **Deterministic** function to decide on one of the contending forks

DSL at UCSB

DSL

UCSB

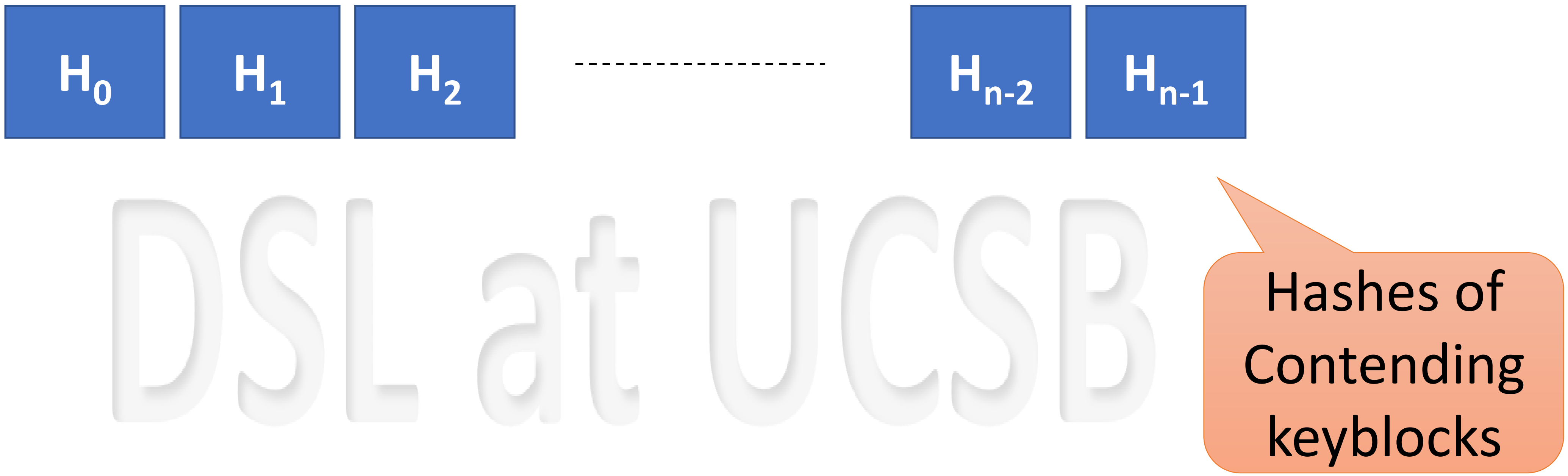
Dealing with Keyblock conflicts and Selfish Mining

DSL at UCSB

DSL

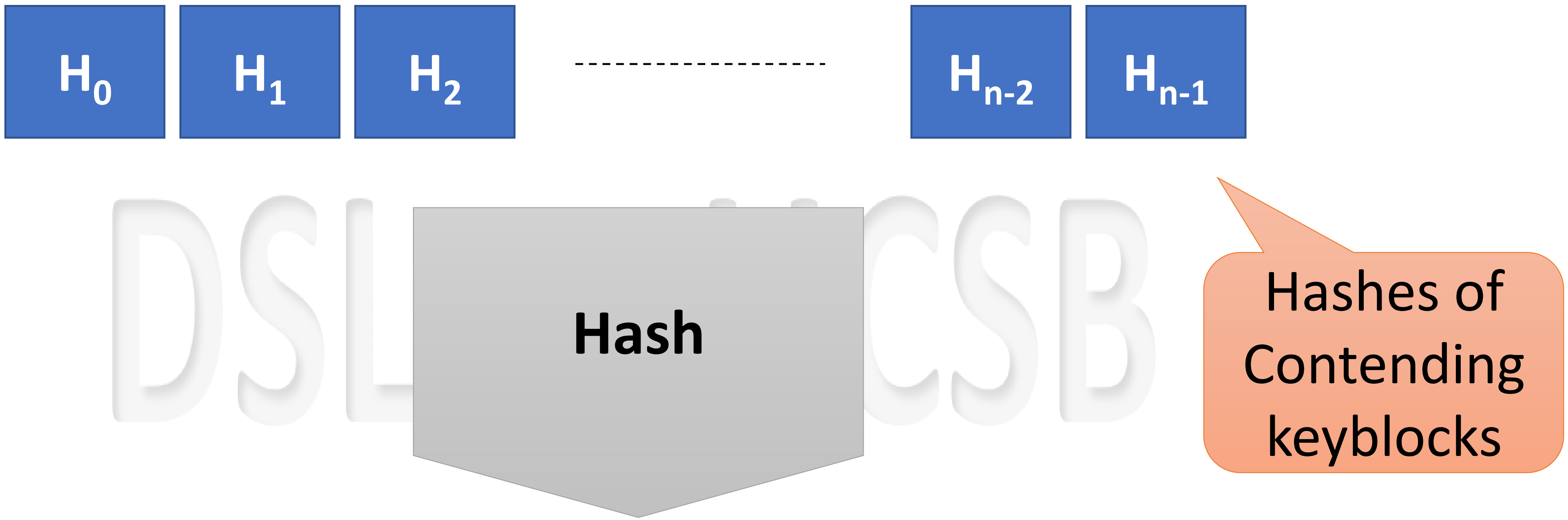
UCSB

Dealing with Keyblock conflicts and Selfish Mining

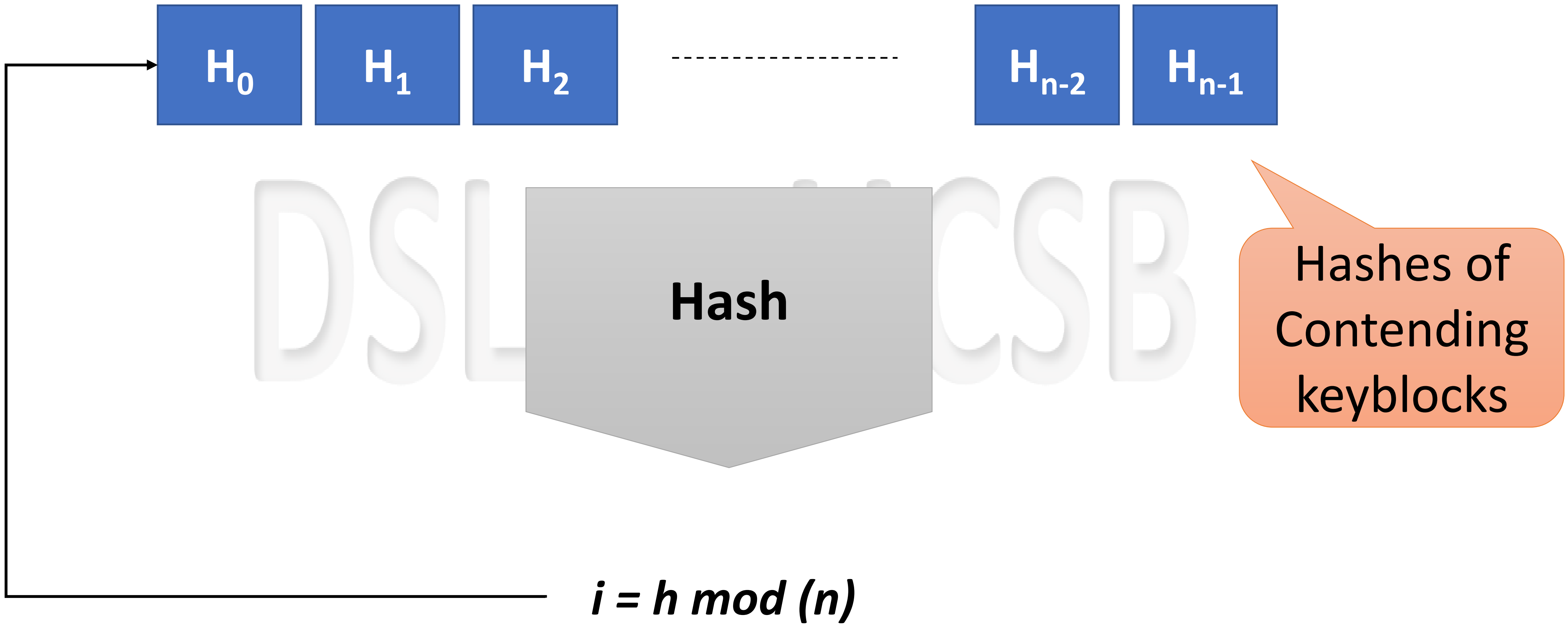


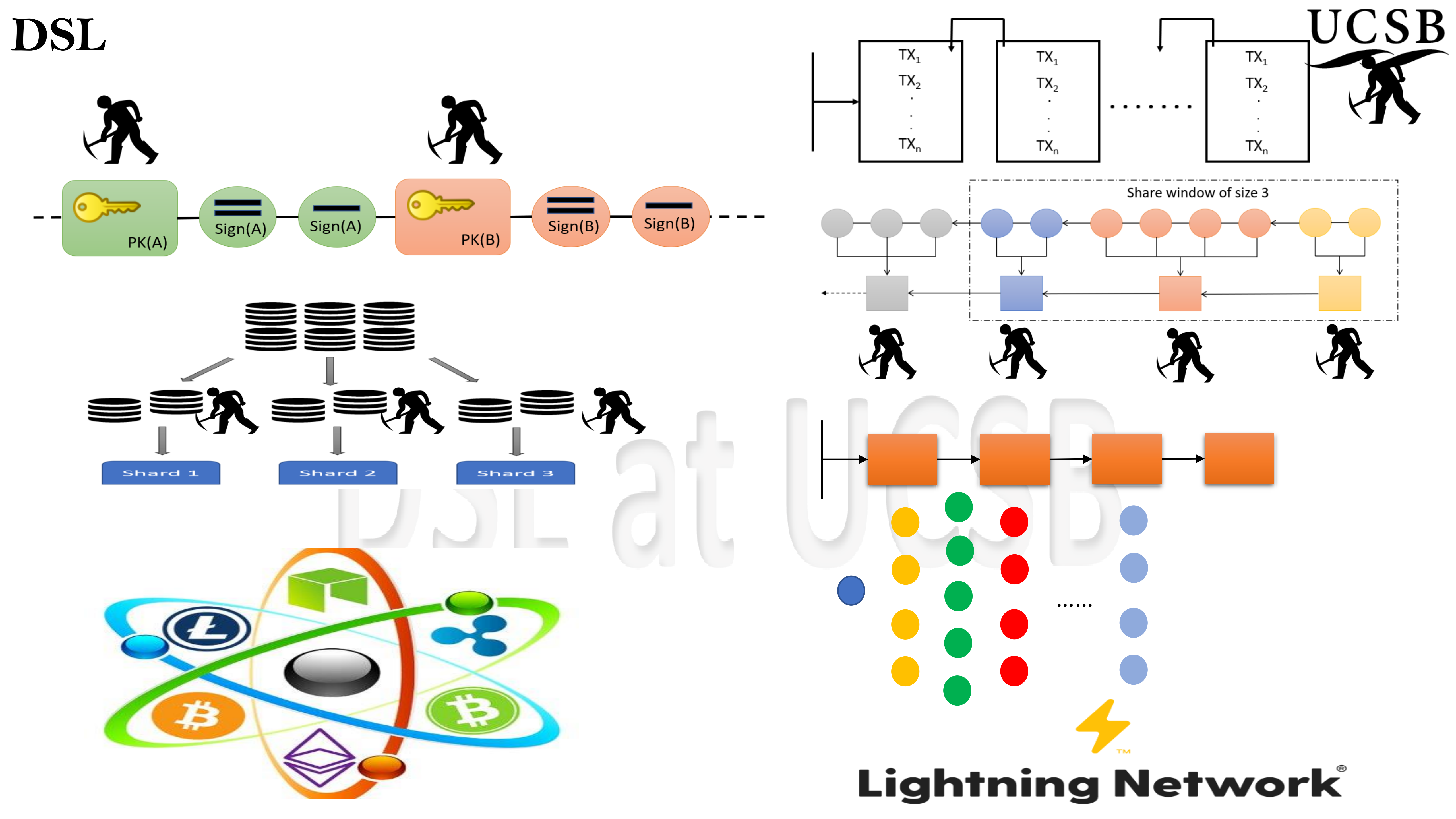
DSL at UCSB

Dealing with Keyblock conflicts and Selfish Mining



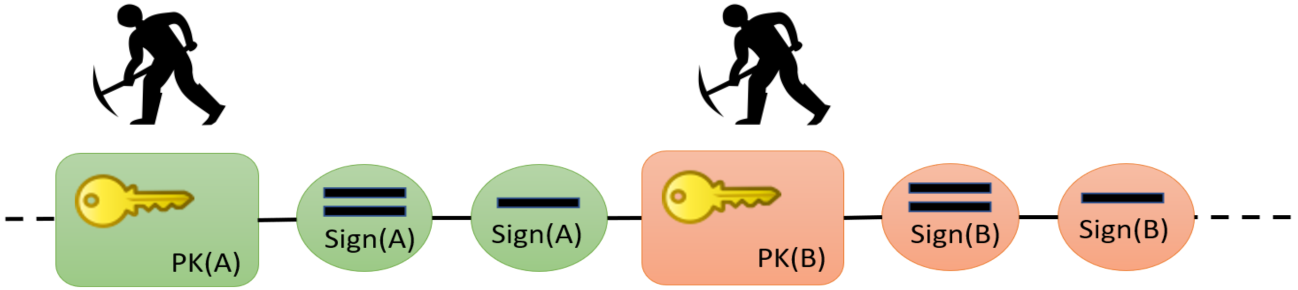
Dealing with Keyblock conflicts and Selfish Mining

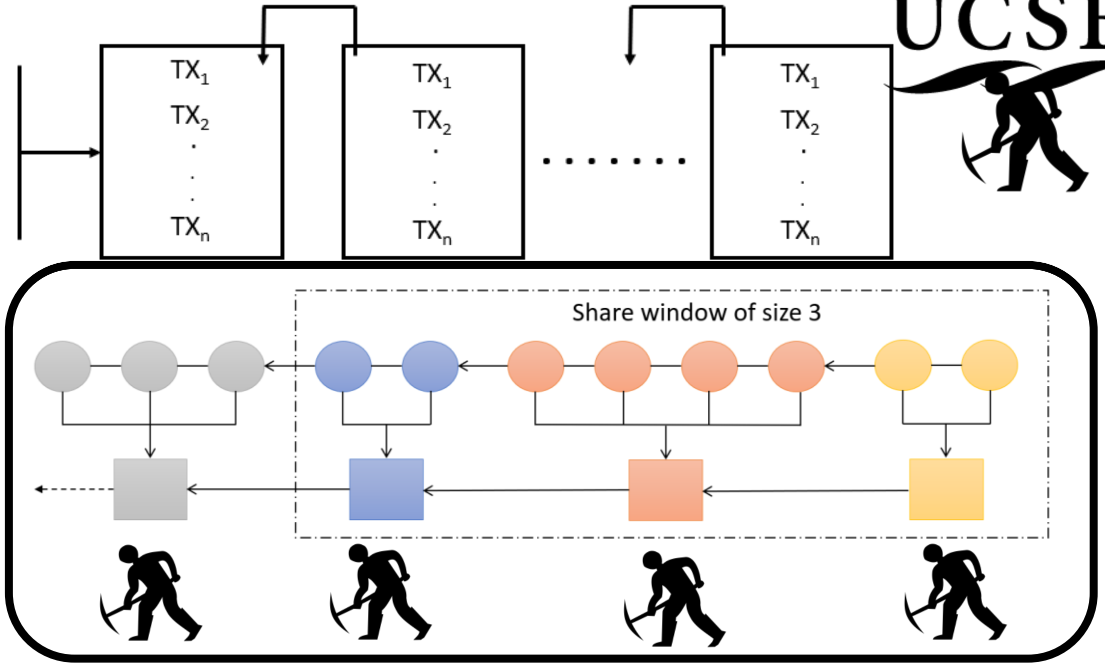


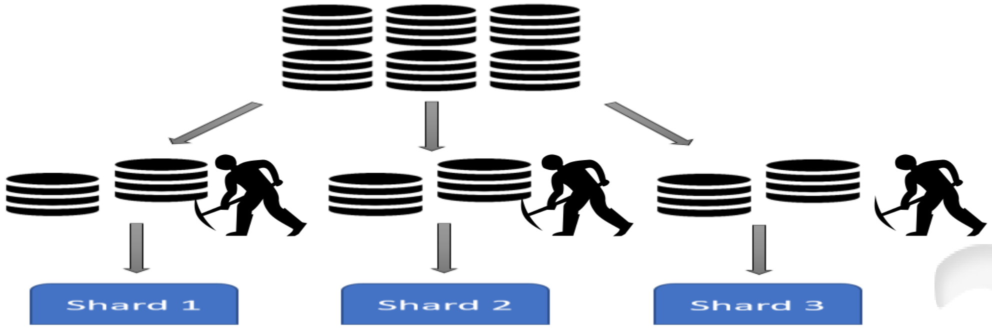

DSL
UCSB
TX1
TX2
TXn
Share window of size 3
PK(A)
Sign(A)
Sign(A)
PK(B)
Sign(B)
Sign(B)
Shard 1
Shard 2
Shard 3
DSL at UCSB
Lightning Network

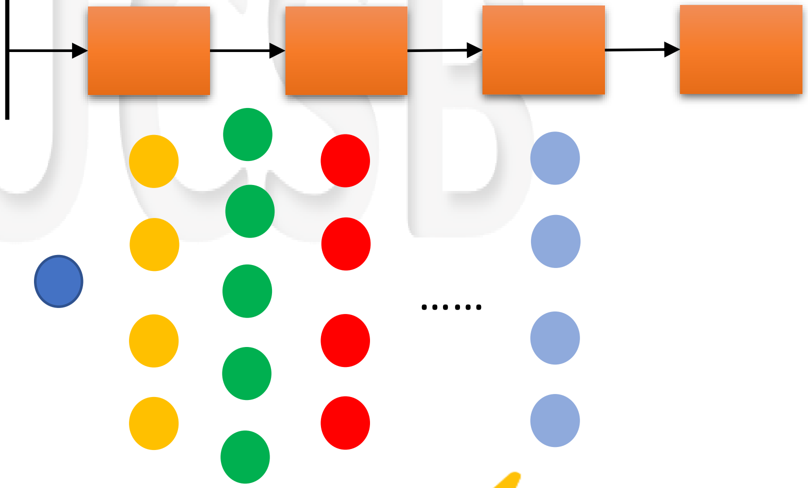
DSL

UCSB









DSL at UCSB

Lightning Network®

SOLUTION 3

Mine once, publish txns many times	BitcoinNG
Form a committee to vouch for new block	ByzCoin
Shard txns across different committees	**Elastico**
Using committees with Proof-of-stake	Algorand

DSL at UCSB

DSL

UCSB

Elastico

A Secure Sharding Protocol For Open Blockchains

DSL at UCSB

Elastico

A Secure Sharding Protocol For Open Blockchains

Scale Bitcoin-like cryptocurrency by adapting ‘shards’

DSL at UCSB

Elastico

A Secure Sharding Protocol For Open Blockchains

Scale Bitcoin-like cryptocurrency by adapting ‘shards’

Uniformly partitions the mining network into smaller committees, each of which processes a disjoint set of txns (or ‘shards’)

DSL

UCSB

Elastico

A Secure Sharding Protocol For Open Blockchains

Scale Bitcoin-like cryptocurrency by adapting 'shards'

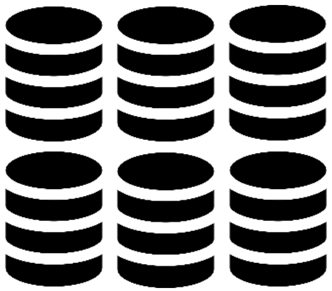
Uniformly partitions the mining network into smaller committees, each of which processes a disjoint set of txns (or 'shards')

Luu, Loi, et al. "A secure sharding protocol for open blockchains." *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016.

Sharding in Elastico

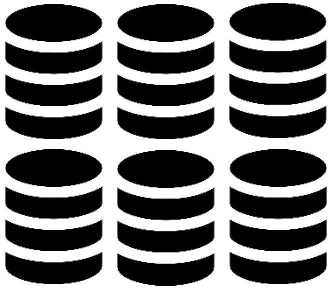
DSL at UCSB

Sharding in Elastico



DSL at UCSB

Sharding in Elastico



- - - - - - - - - - - - ->

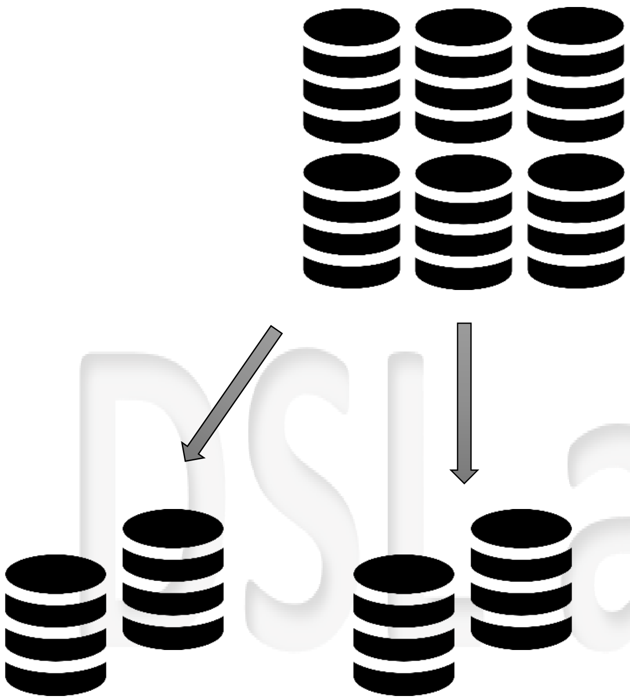
Network of nodes

DSL at UCSB

Sharding in Elastico

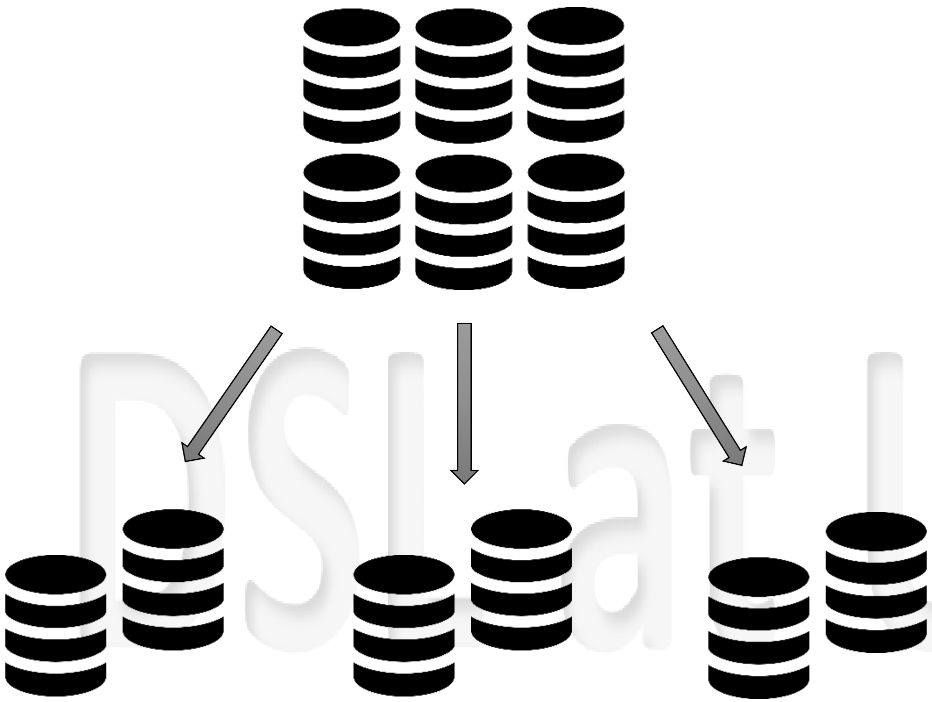
DSL at UCSB

Sharding in Elastico



DSL at UCSB

Sharding in Elastico

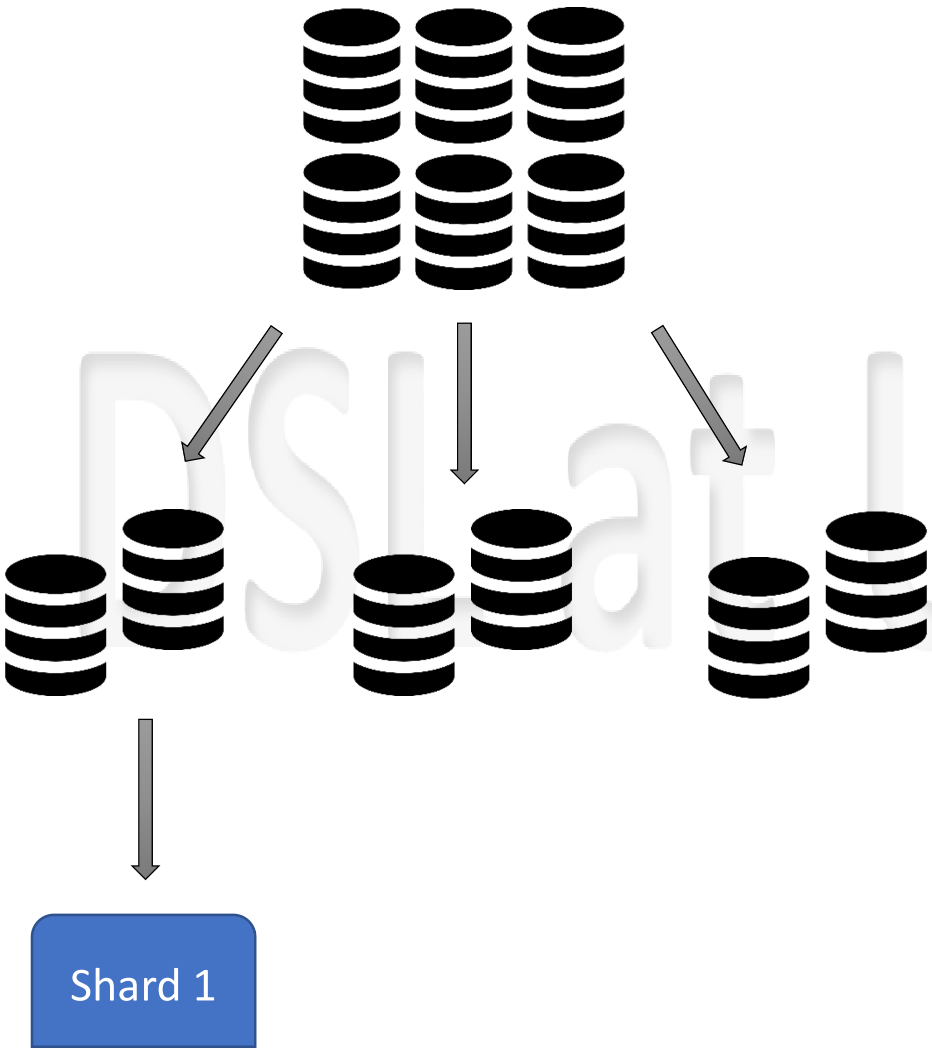


DSL at UCSB

Sharding in Elastico

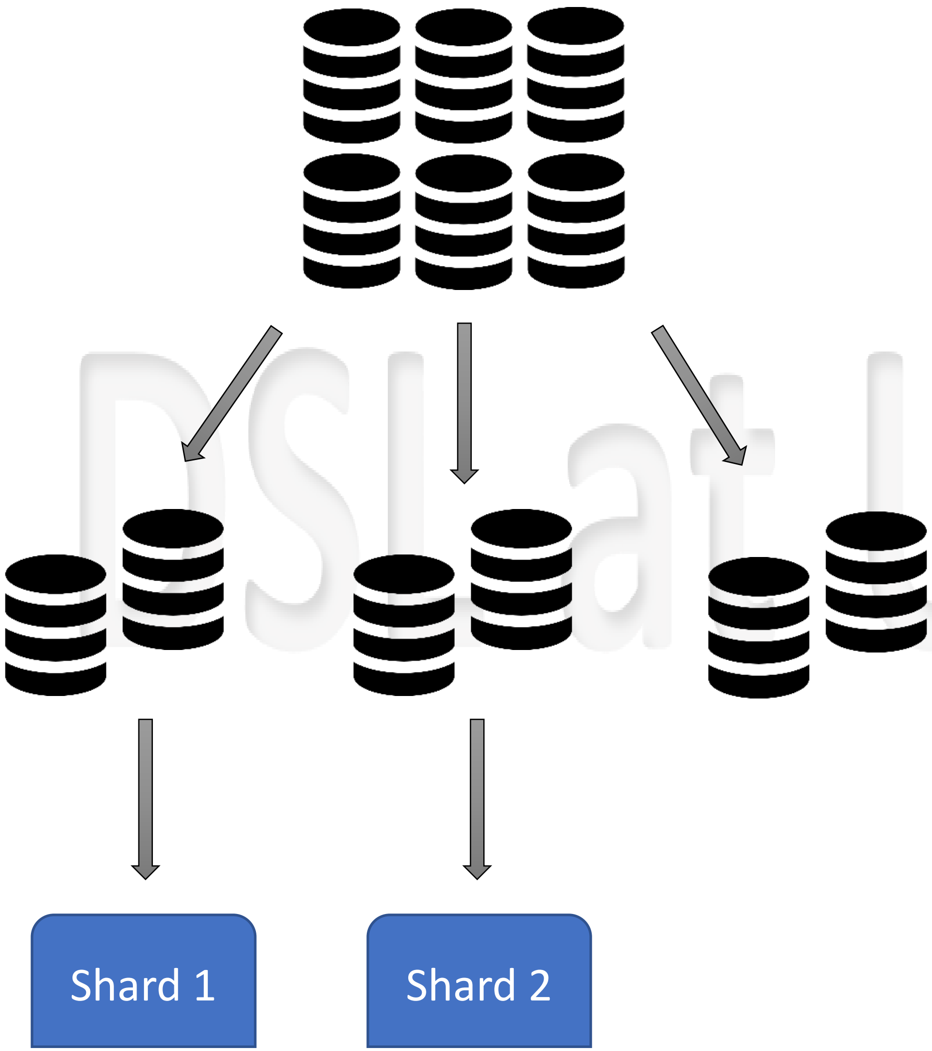
DSL at UCSB

Sharding in Elastico



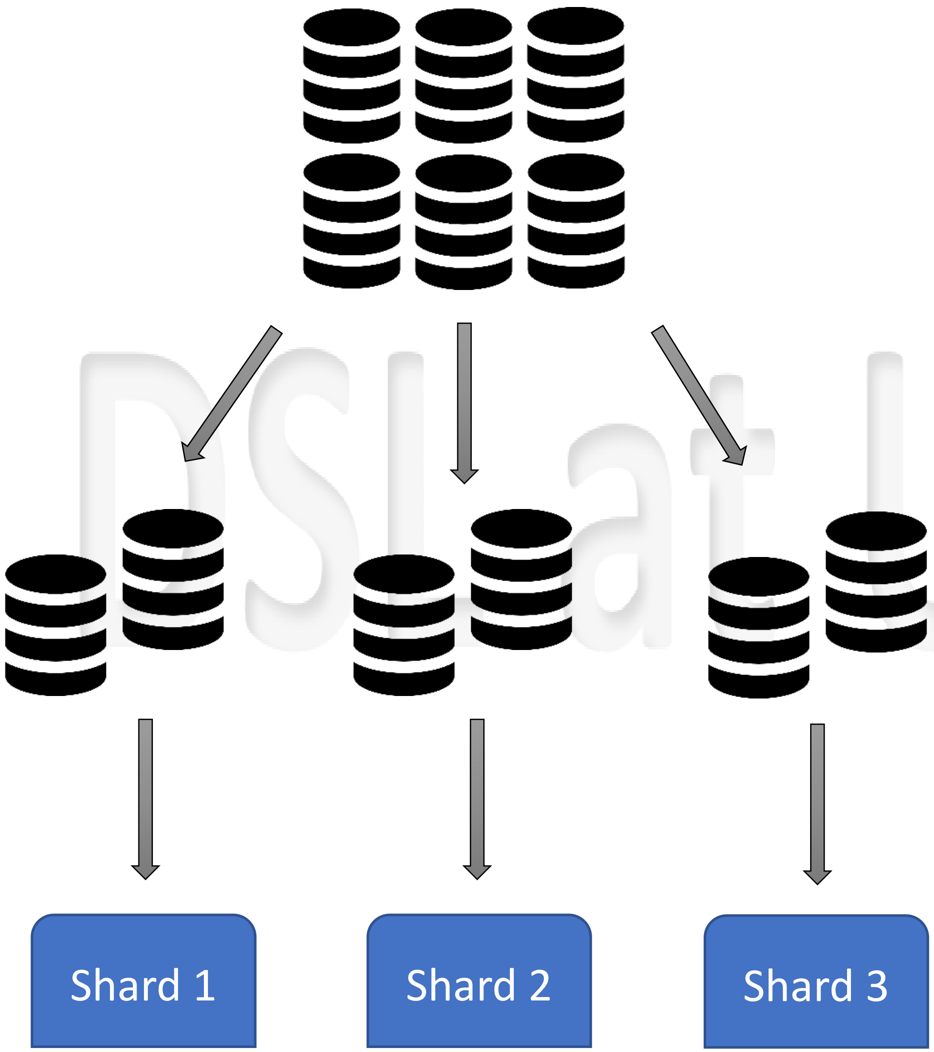
DSL at UCSB

Sharding in Elastico



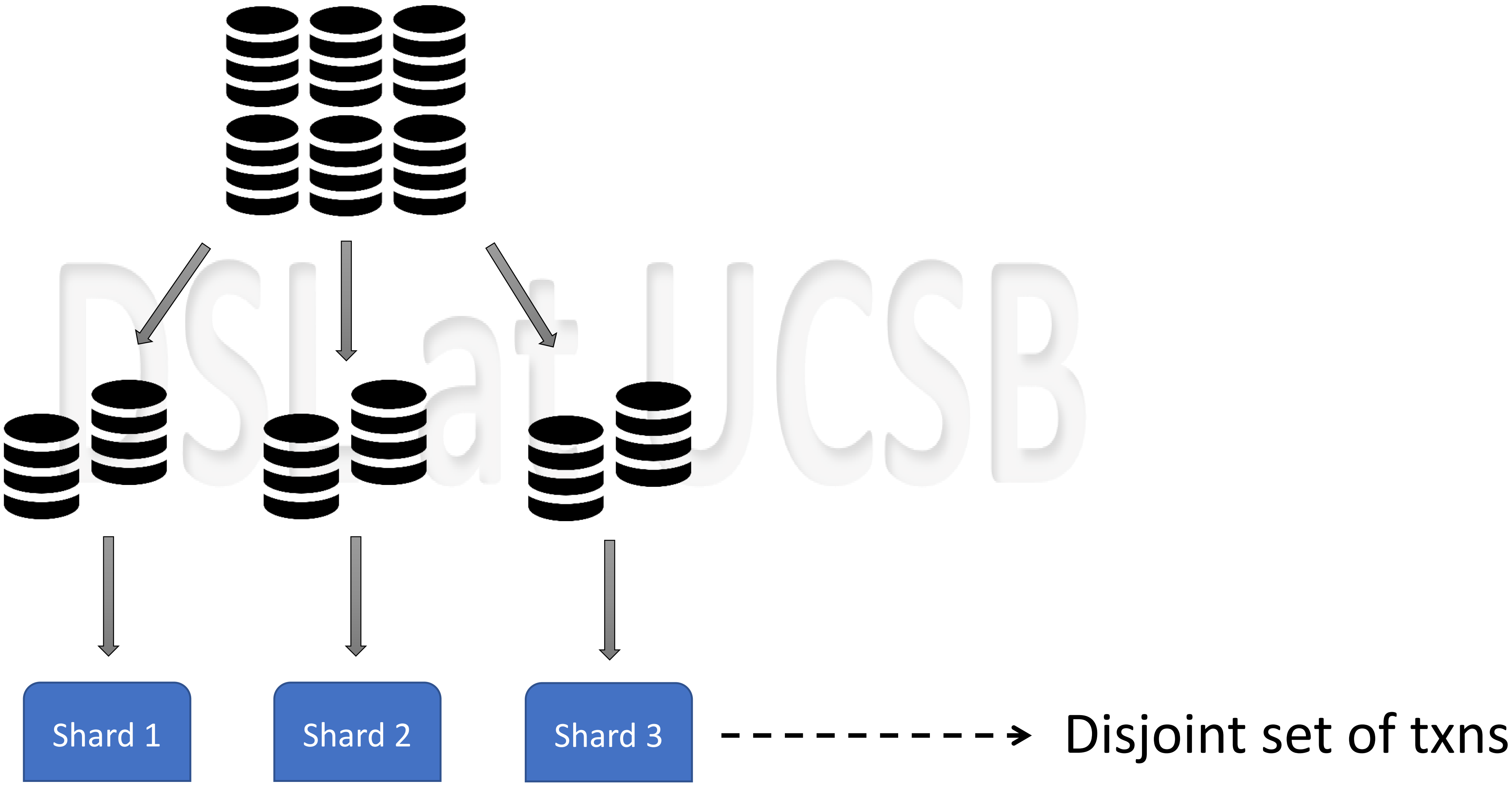
DSL at UCSB

Sharding in Elastico



DSL at UCSB

Sharding in Elastico

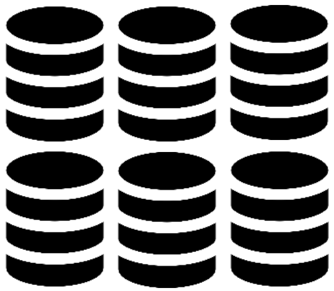


DSL at UCSB

Naïve Strawman Solution

DSL at UCSB

Naïve Strawman Solution



Assumptions:

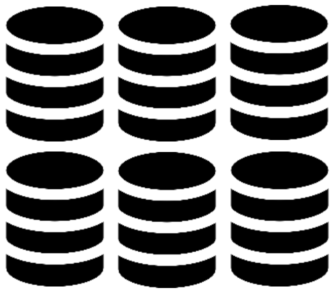
- The list of nodes is known for each epoch
- Common random coin

DSL at UCSB

DSL

UCSB

Naïve Strawman Solution



Assumptions:

- The list of nodes is known for each epoch
- Common random coin

H(coin || PK)

DSL at UCSB

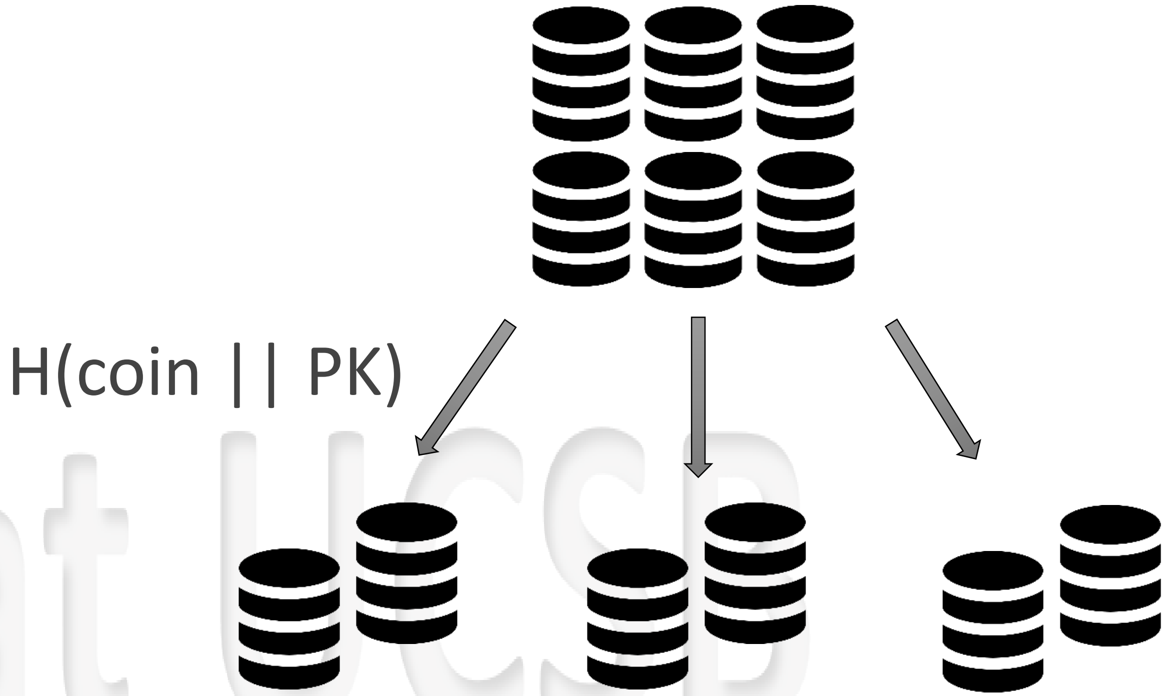
DSL

UCSB

Naïve Strawman Solution

Assumptions:

- The list of nodes is known for each epoch
- Common random coin

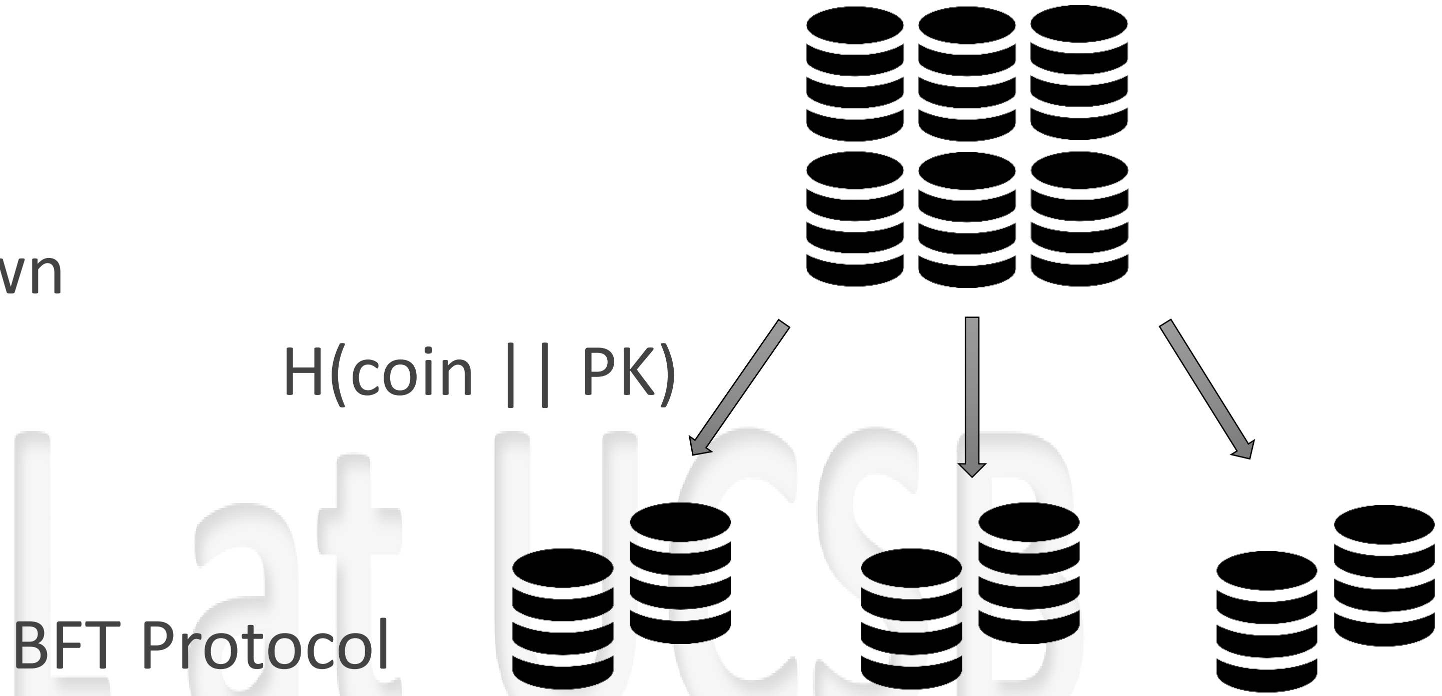


DSL at UCSB

Naïve Strawman Solution

Assumptions:

- The list of nodes is known for each epoch
- Common random coin



DSL at UCSB

Naïve Strawman Solution

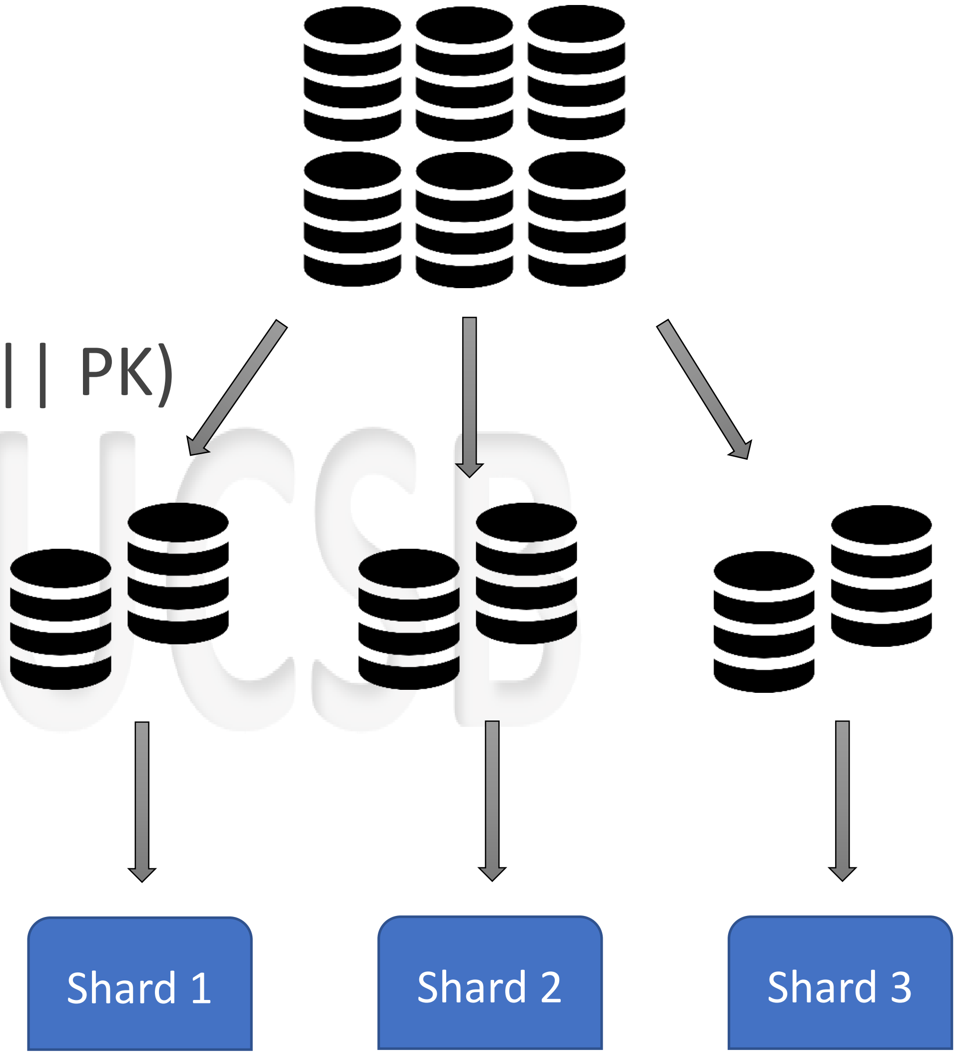
Assumptions:

- The list of nodes is known for each epoch
- Common random coin

H(coin || PK)

BFT Protocol

Broadcast all shards



DSL at UCSB

Step 1: Identity establishment

DSL at UCSB

Step 1: Identity establishment

ID = H(epochRandomness || IP || PK || nonce) < D

DSL at UCSB

Step 1: Identity establishment

$$ID = H(epochRandomness \,||\, IP \,||\, PK \,||\, nonce) < D$$

Random seed for PoW

DSL at UCSB

Step 1: Identity establishment

$$ID = H(epochRandomness \,||\, IP \,||\, PK \,||\, nonce) < D$$

Random seed for PoW

IP and Public Key

DSL at UCSB

Step 1: Identity establishment

$$ID = H(epochRandomness \,||\, IP \,||\, PK \,||\, nonce) < D$$

Random seed for PoW

IP and Public Key

Difficulty

DSL at UCSB

Step 1: Identity establishment

ID = H(epochRandomness || IP || PK || nonce) < D

Random seed for PoW

IP and Public Key

Difficulty

DSL at UCSB

The last ***s*** bits of ID specifies which (***s***-bit) committee id the node belongs to

Step 1: Committee assignment based on ID

DSL at UCSB

Step 1: Committee assignment based on ID

Node	ID
1	000001.....101
2	000001......110
3	000000......010
4	000001......001

UCSB

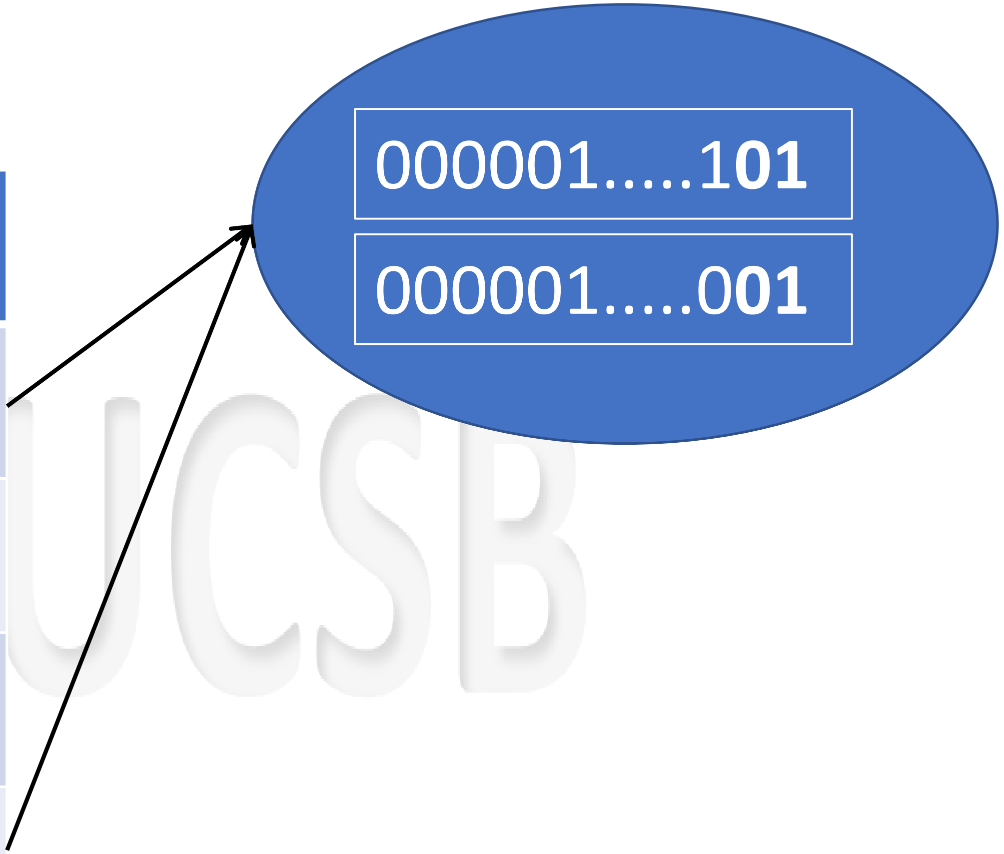
Step 1: Committee assignment based on ID

Node	ID
1	000001.....101
2	000001......110
3	000000......010
4	000001......001

UCSB

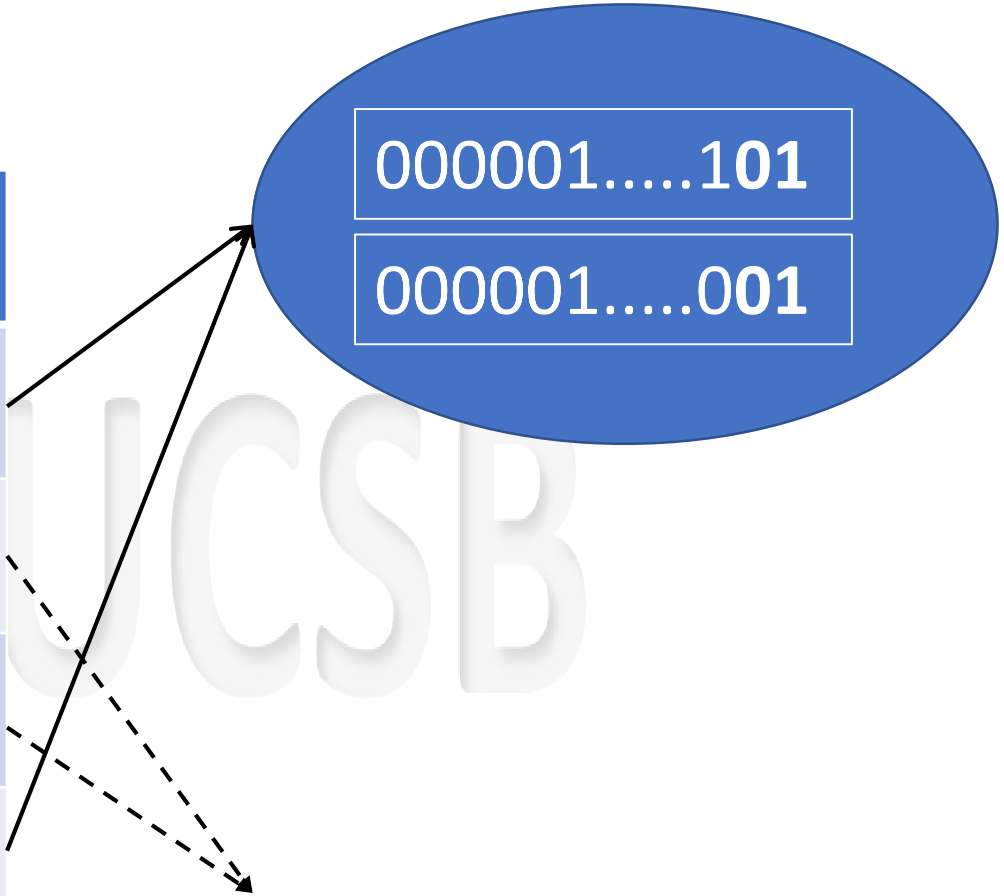
Step 1: Committee assignment based on ID

Node	ID
1	000001.....101
2	000001......110
3	000000......010
4	000001......001



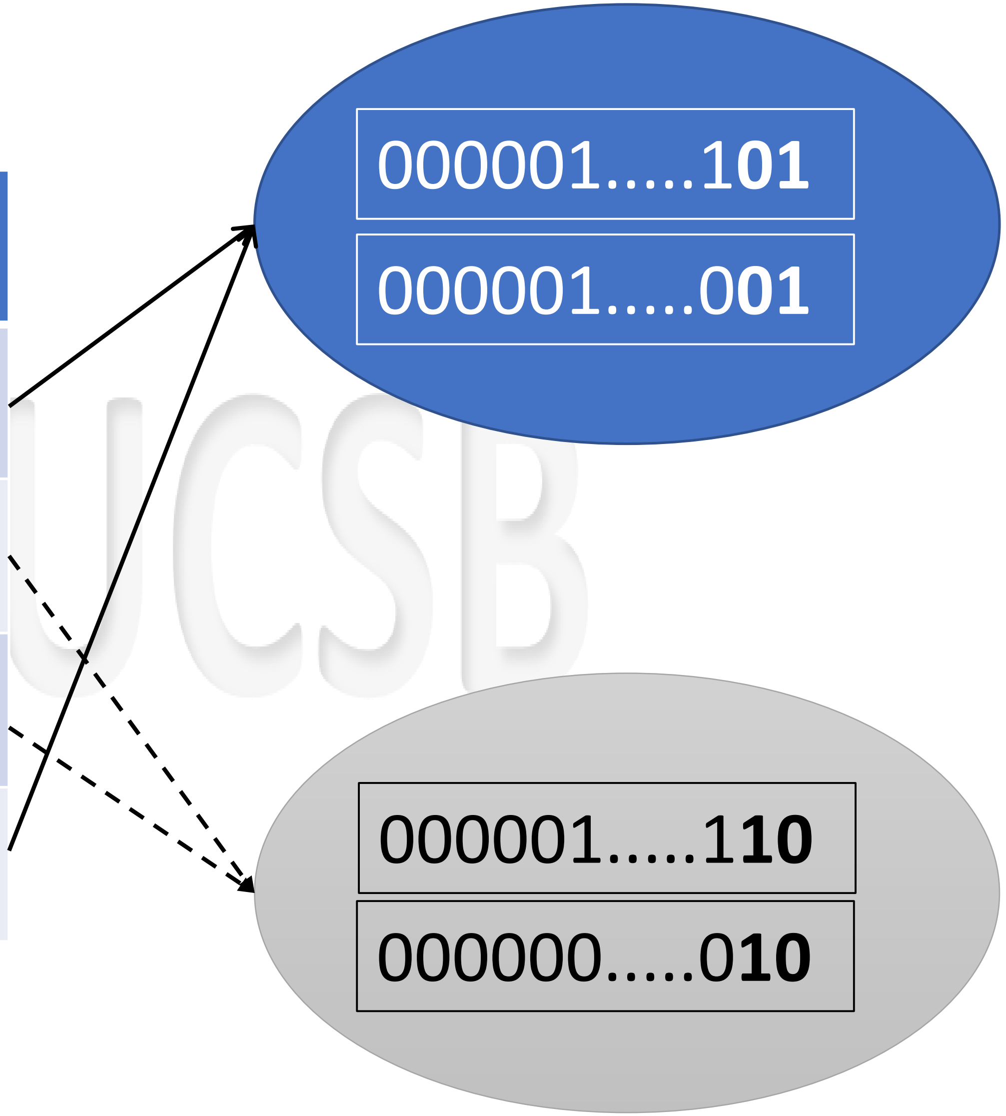
Step 1: Committee assignment based on ID

Node	ID
1	000001.....101
2	000001......110
3	000000......010
4	000001......001



Step 1: Committee assignment based on ID

Node	ID
1	000001.....101
2	000001......110
3	000000......010
4	000001......001



Identify committee members

DSL at UCSB

Identify committee members

How to **identify** other committee members?

DSL at UCSB

Identify committee members

How to **identify** other committee members?

- Naïve solution: **Broadcast** to all

DSL at UCSB

Identify committee members

How to **identify** other committee members?

- Naïve solution: **Broadcast** to all

  Complexity **$O(n^2)$**

DSL at UCSB

Identify committee members

How to **identify** other committee members?

- Naïve solution: **Broadcast** to all

  Complexity $O(n^2)$

- A special committee: **Directories** of size c

DSL at UCSB

Identify committee members

How to **identify** other committee members?

- Naïve solution: **Broadcast** to all

  Complexity **$O(n^2)$**

- A special committee: **Directories** of size ***c***

  Complexity **O(nc)**

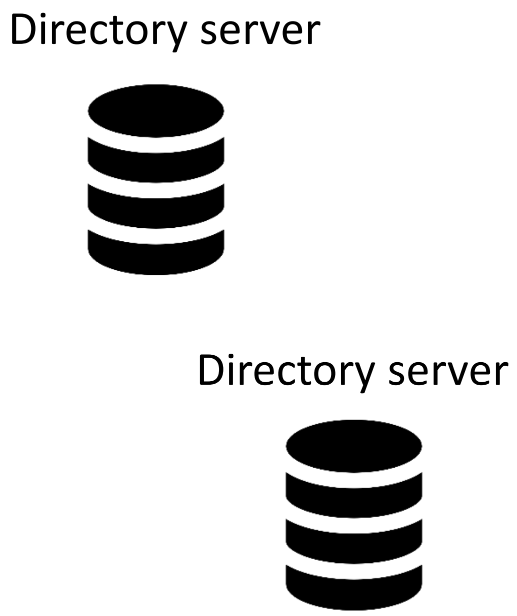
DSL at UCSB

Step 2: Directory committees

DSL at UCSB

Step 2: Directory committees

First ***c*** identities become directory servers

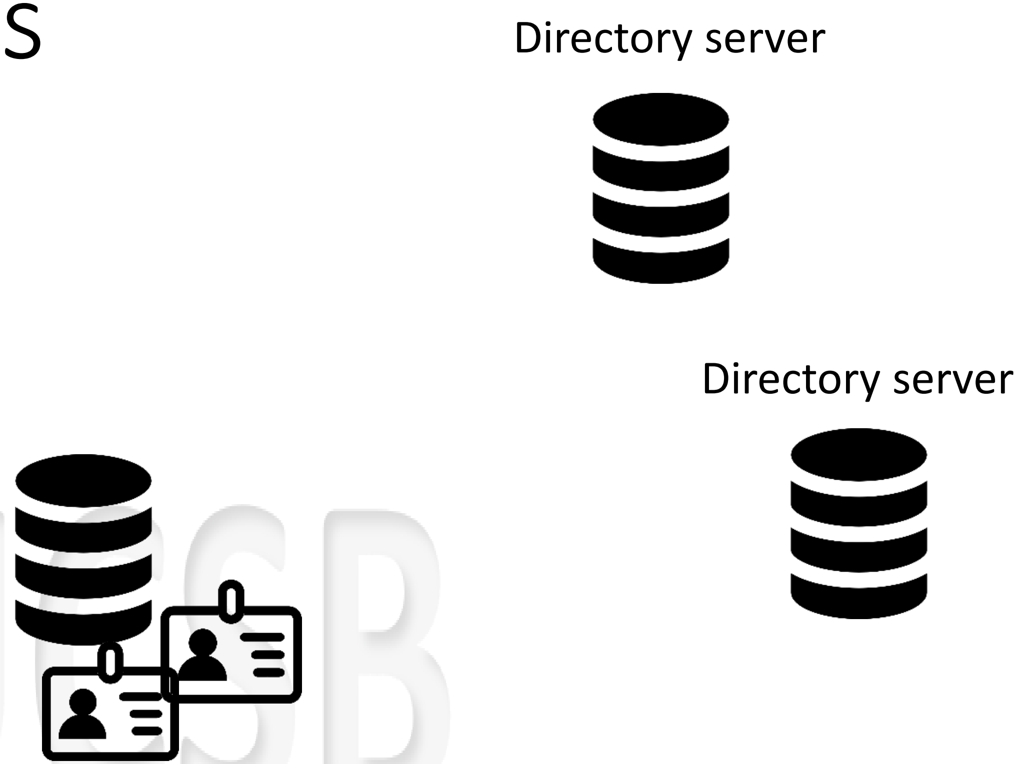


DSL at UCSB

Step 2: Directory committees

First ***c*** identities become directory servers

Latter nodes send IDs to directories

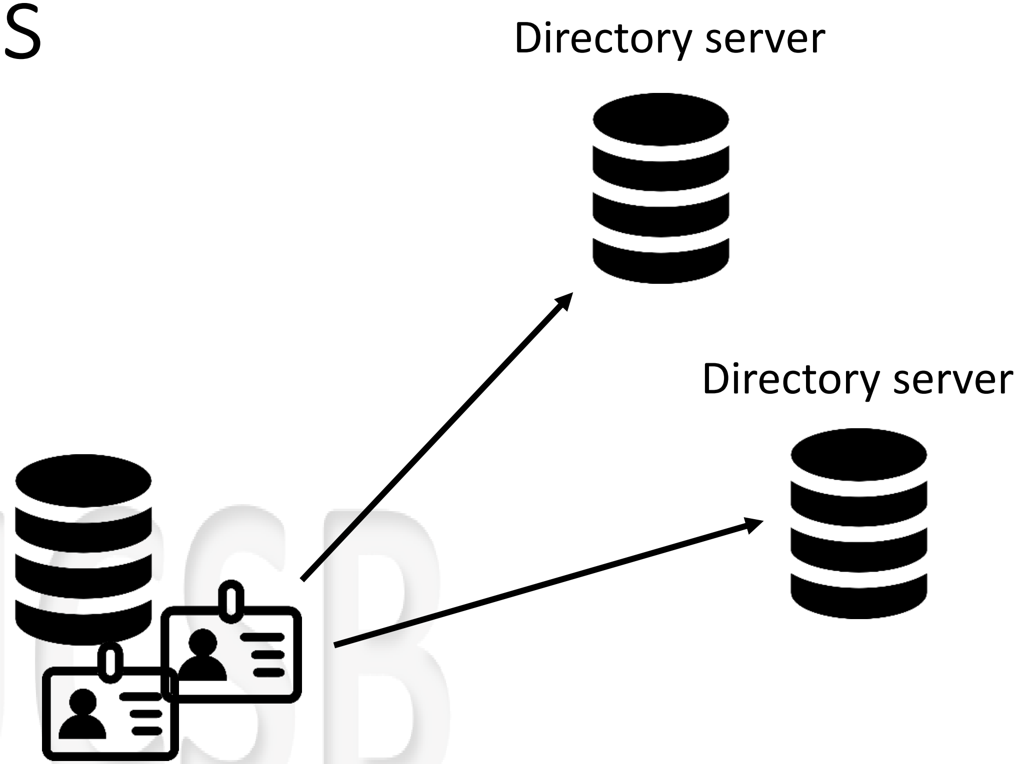


DSL at UCSB

Step 2: Directory committees

First ***c*** identities become directory servers

Latter nodes send IDs to directories

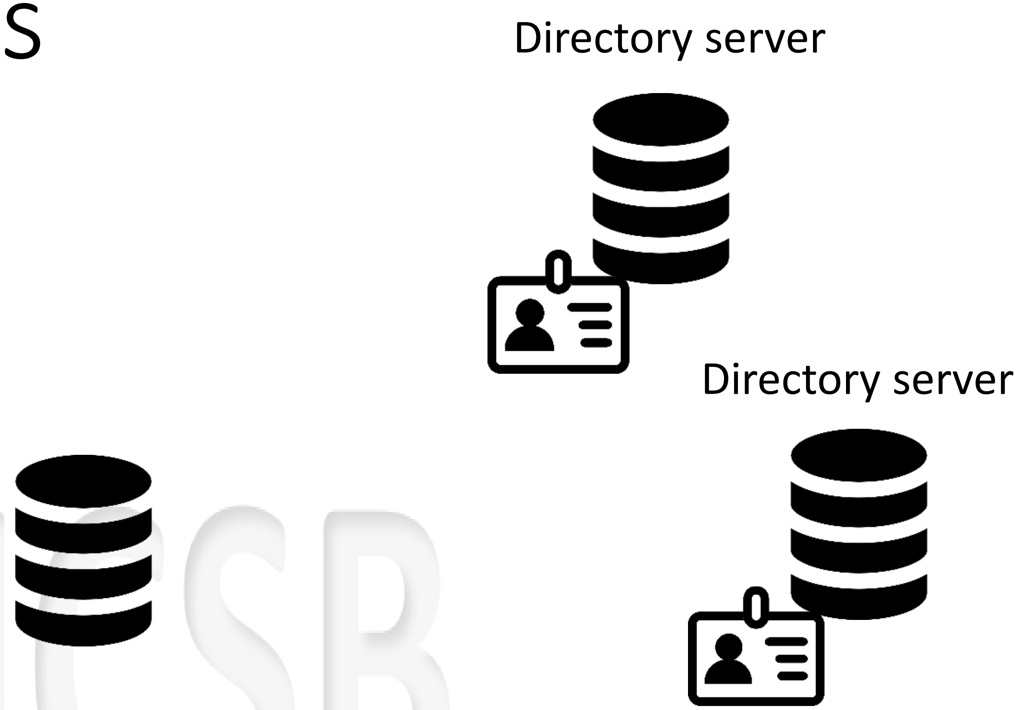


DSL at UCSB

Step 2: Directory committees

First *c* identities become directory servers

Latter nodes send IDs to directories

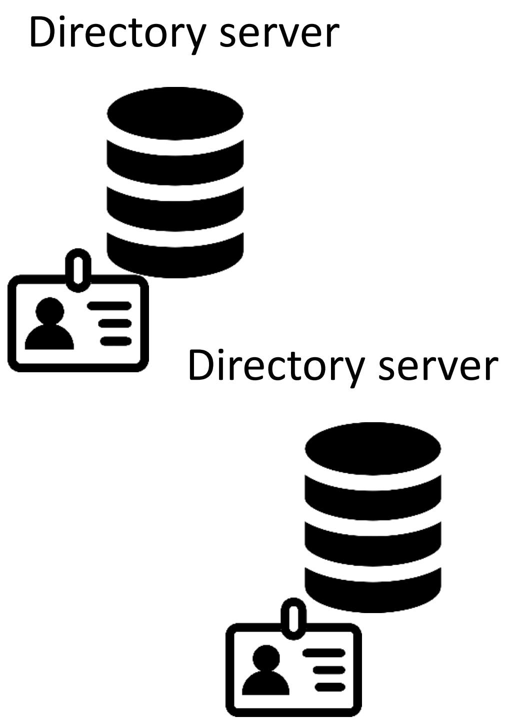


DSL at UCSB

Step 2: Directory committees

First ***c*** identities become directory servers

Latter nodes send IDs to directories



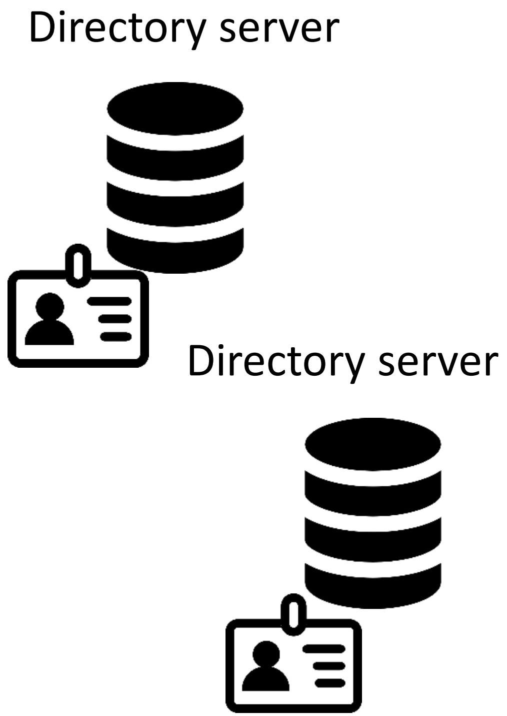
DSL at UCSB

Step 2: Directory committees

First *c* identities become directory servers

Latter nodes send IDs to directories

Directories send committee list to nodes



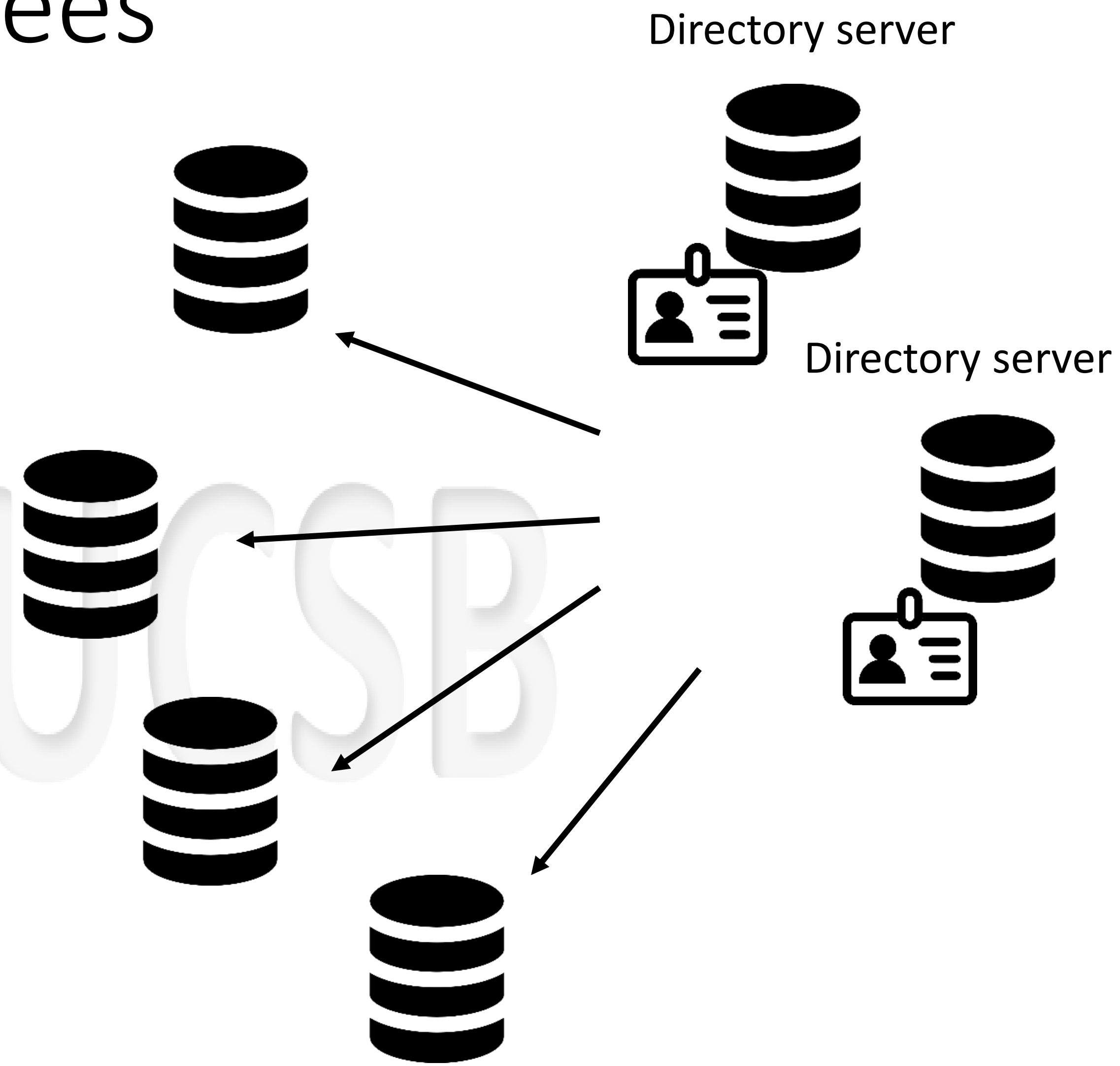
DSL at UCSB

Step 2: Directory committees

First *c* identities become directory servers

Latter nodes send IDs to directories

Directories send committee list to nodes



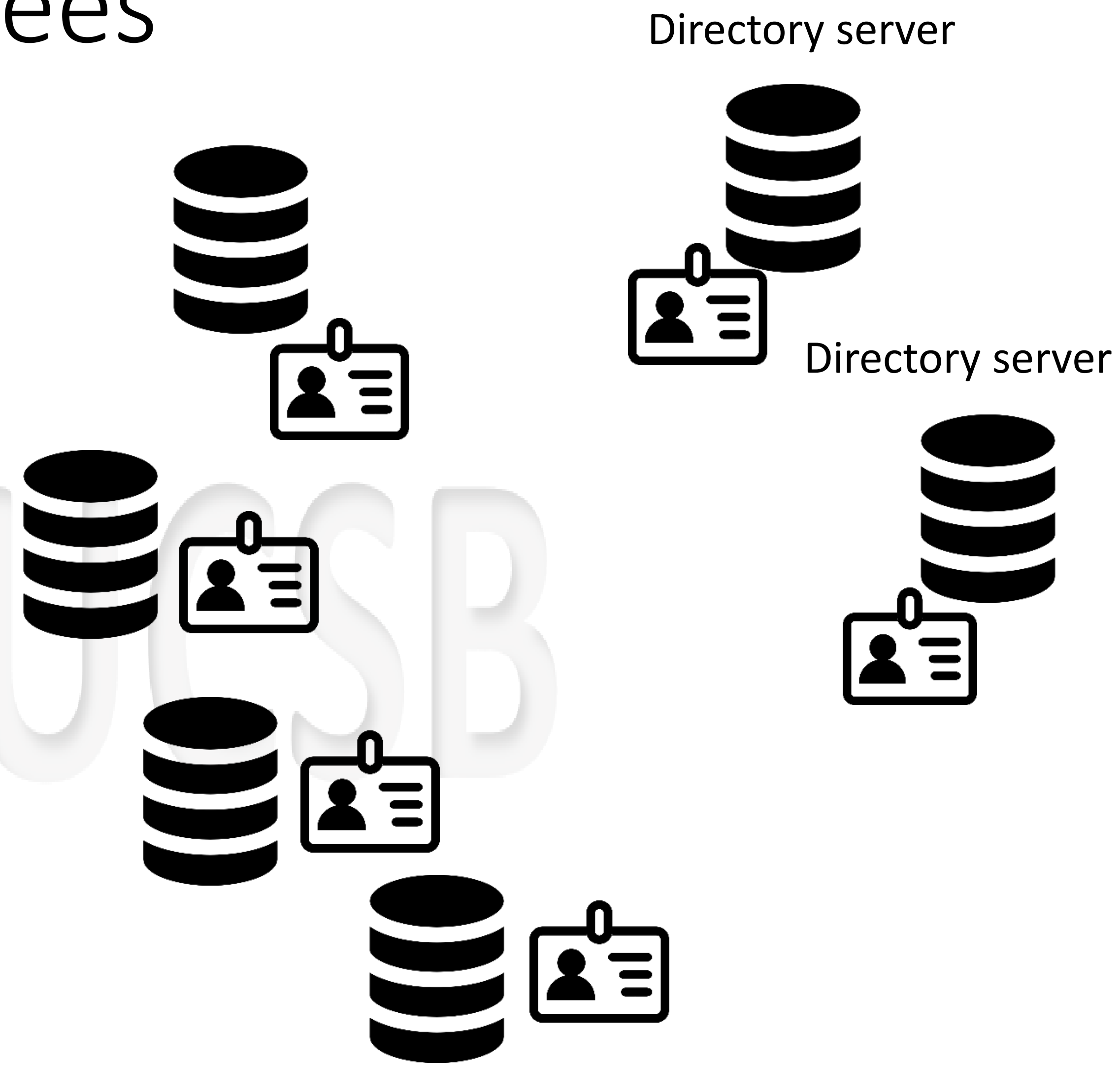
DSL at UCSB

Step 2: Directory committees

First *c* identities become directory servers

Latter nodes send IDs to directories

Directories send committee list to nodes



DSL at UCSB

Step 3: Block Proposals Within Committees

DSL at UCSB

Step 3: Block Proposals Within Committees

DSL at UCSB

Transactions in committee 1

Txns
Tx-11
Tx-21 …
Tx-n1

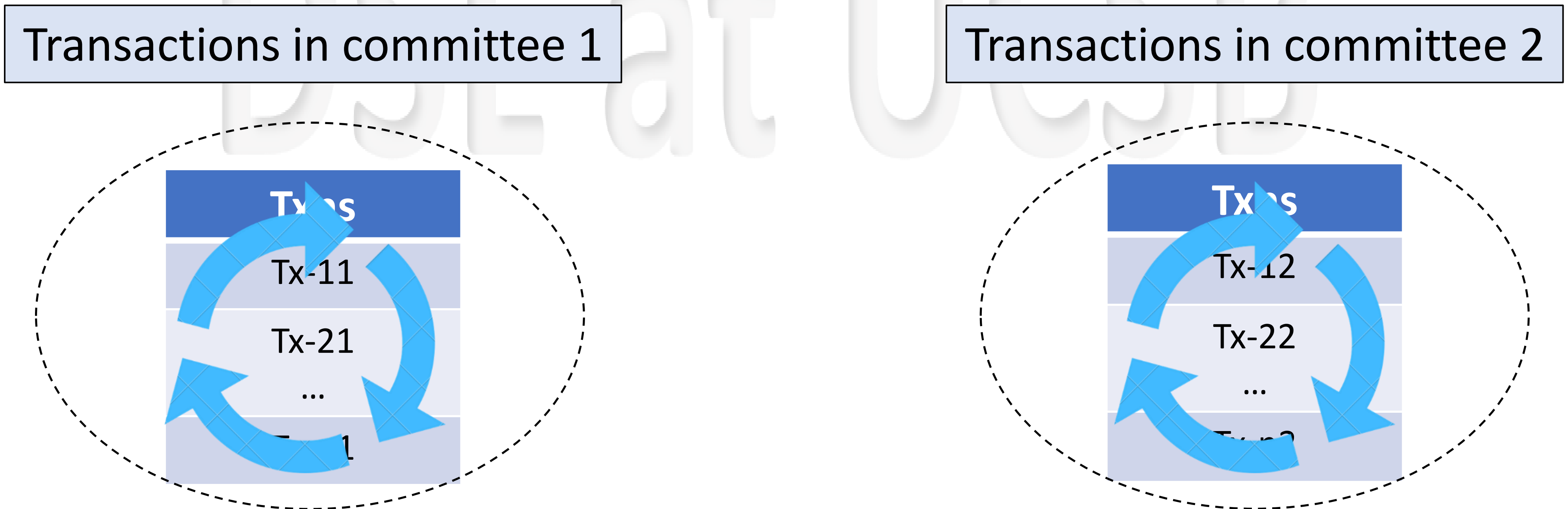
Transactions in committee 2

Txns
Tx-12
Tx-22 …
Tx-n2

Step 3: Block Proposals Within Committees

- Run classical **Byzantine agreement** protocol

DSL at UCSB

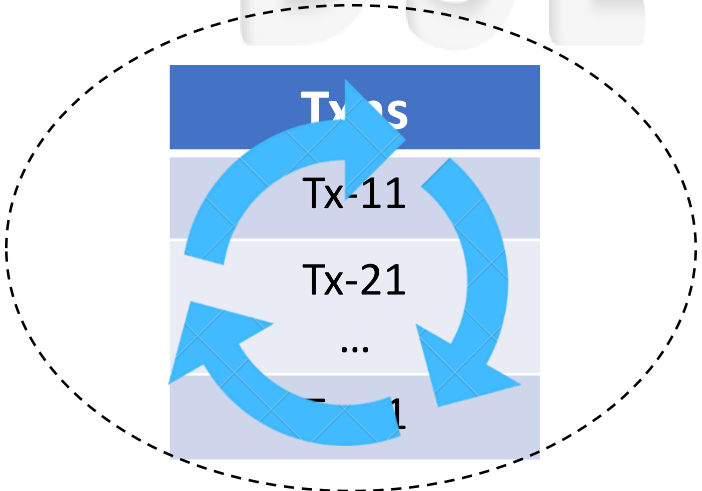


Step 3: Block Proposals Within Committees

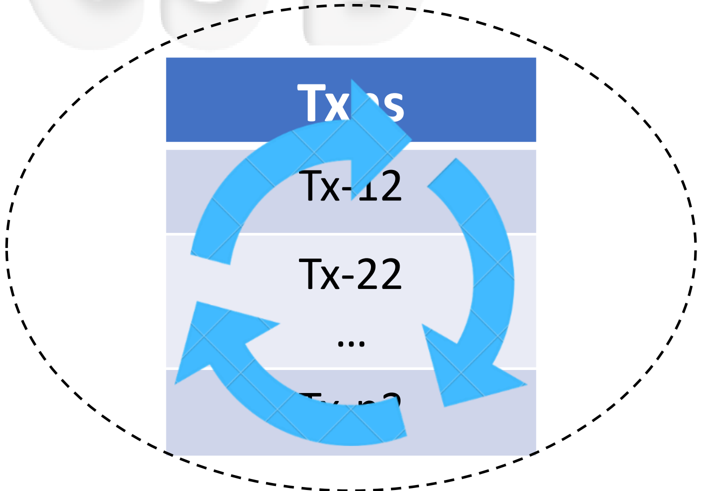
- Run classical **Byzantine agreement** protocol
- Members agree and sign on **one** set of txns

DSL at UCSB

Transactions in committee 1

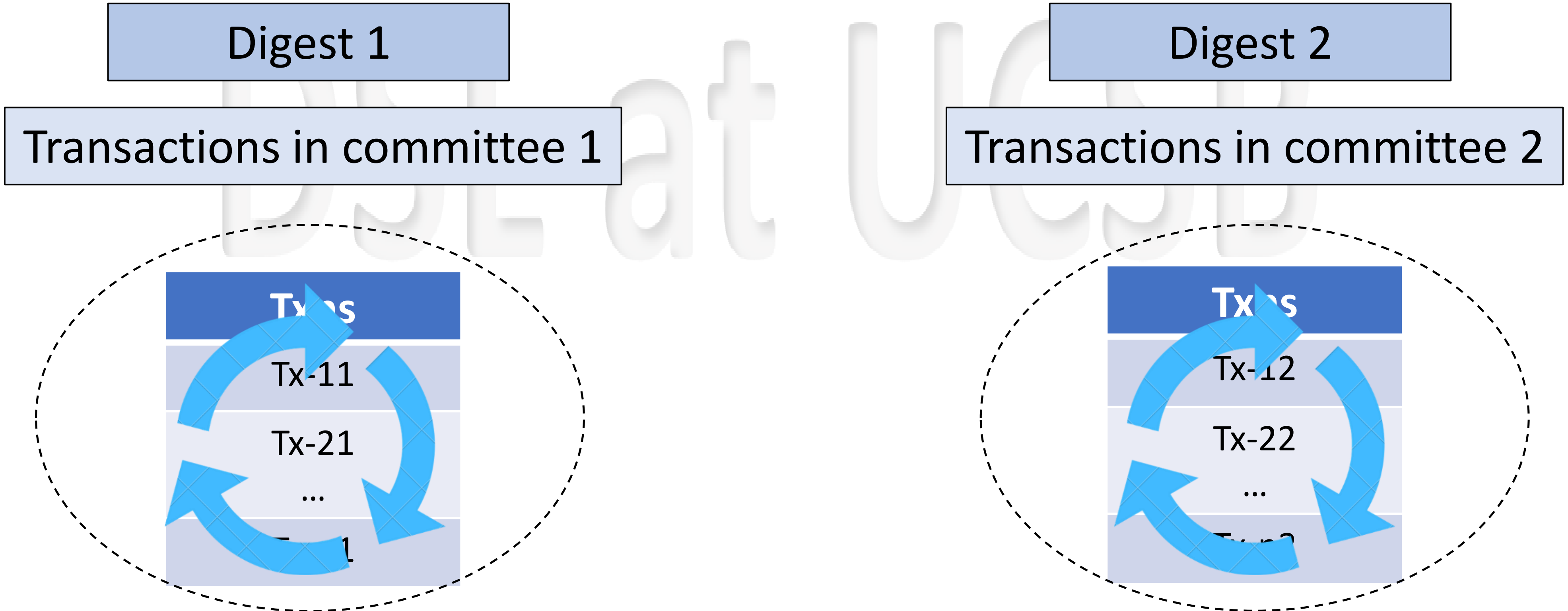


Transactions in committee 2



Step 3: Block Proposals Within Committees

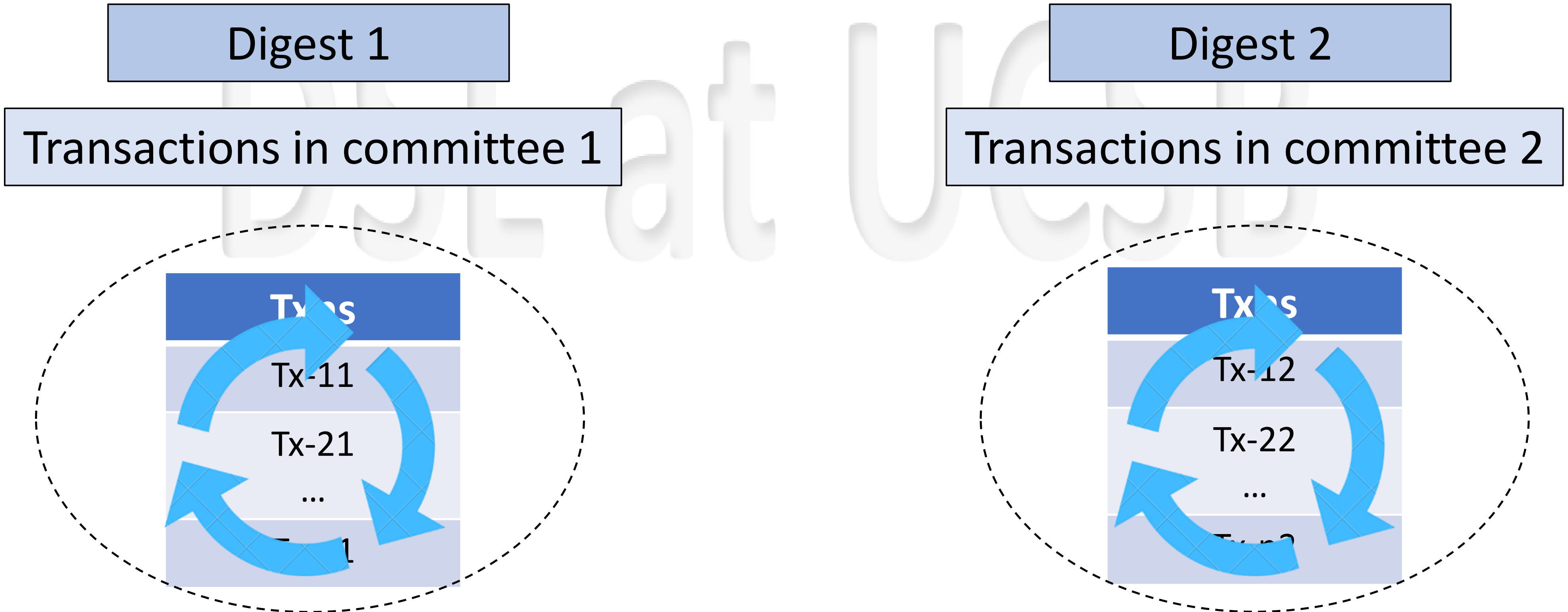
- Run classical **Byzantine agreement** protocol
- Members agree and sign on **one** set of txns



DSL at UCSB

Step 3: Block Proposals Within Committees

- Run classical **Byzantine agreement** protocol
- Members agree and sign on **one** set of txns
- # of messages **O(c^2)**



DSL at UCSB

Step 4: Final Committee

DSL at UCSB

Step 4: Final Committee

- A **special** committee to **finalize** on the next block

DSL at UCSB

Step 4: Final Committee

- A **special** committee to **finalize** on the next block
- Why??

DSL at UCSB

Step 4: Final Committee

- A **special** committee to **finalize** on the next block

- Why??
- To avoid **forks**

DSL at UCSB

Step 4: Final Committee

- A **special** committee to **finalize** on the next block

- Why??
- To avoid **forks**

- To **verify** if each committee block is **signed** by enough committee members

DSL at UCSB

Step 4: Final Committee

- A **special** committee to **finalize** on the next block

- Why??
- To avoid **forks**

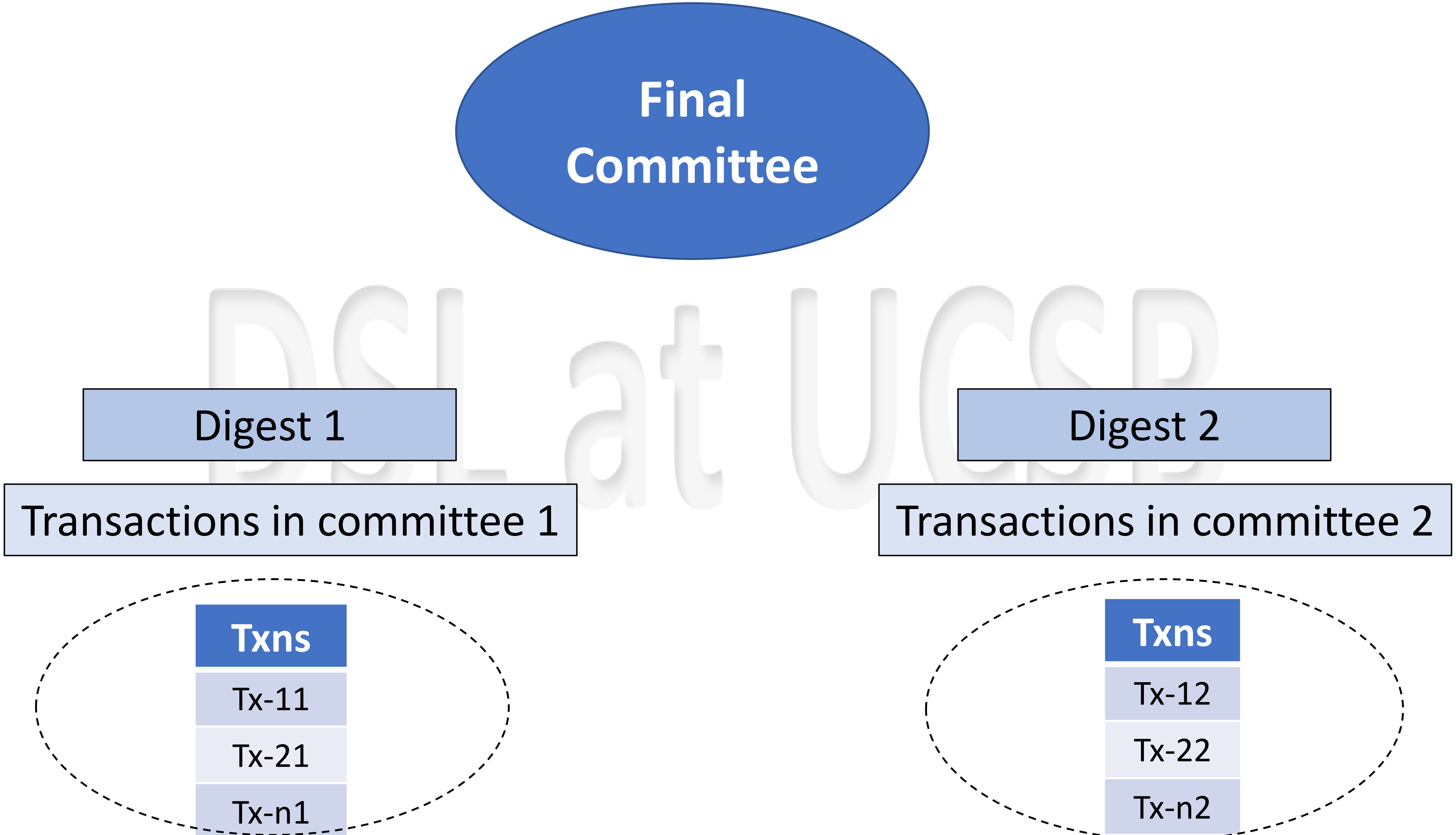
- To **verify** if each committee block is **signed** by enough committee members

- To **generate random** values for next epoch

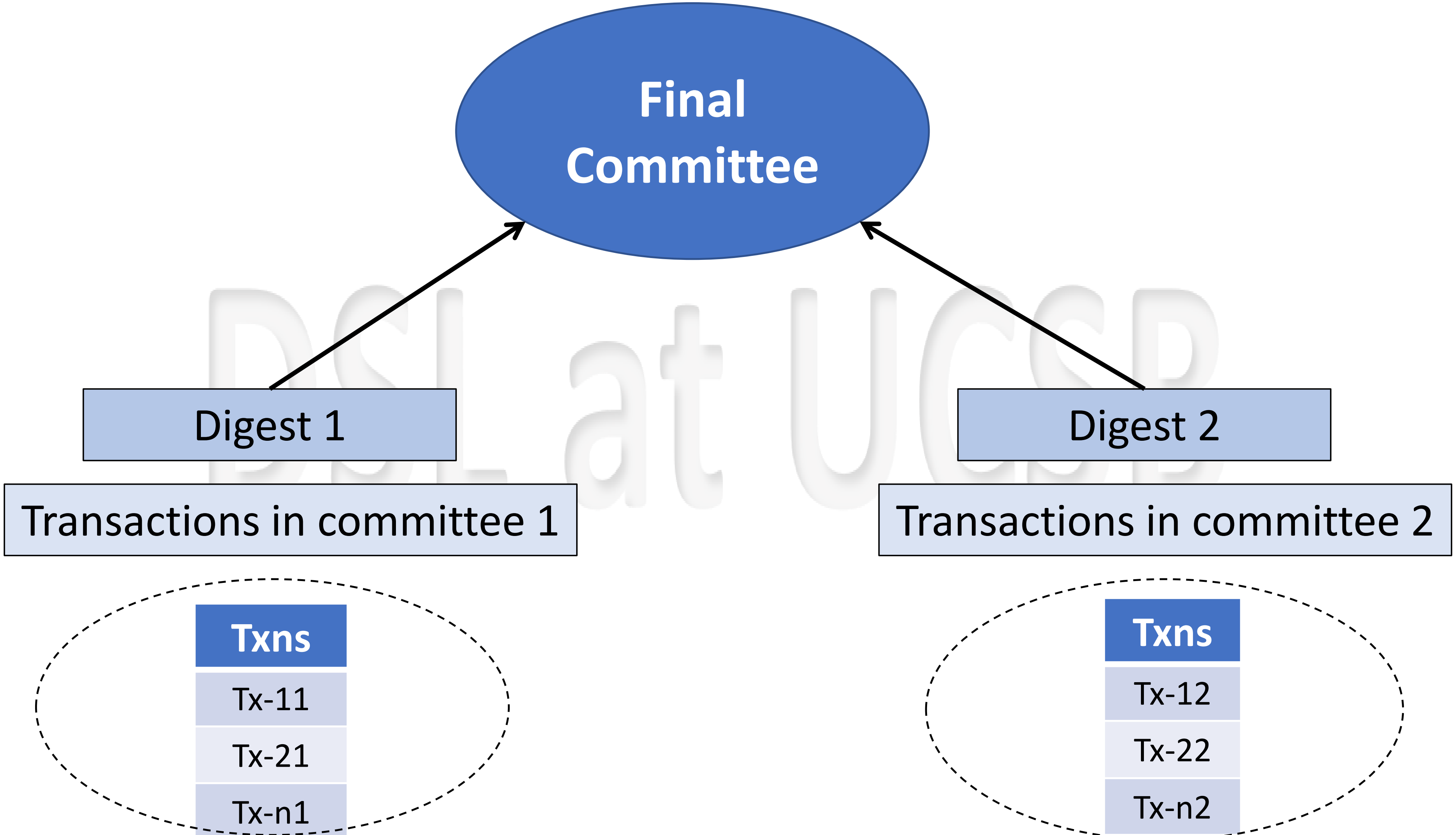
DSL at UCSB

Step 4: Final Committee Union of Blocks

DSL at UCSB

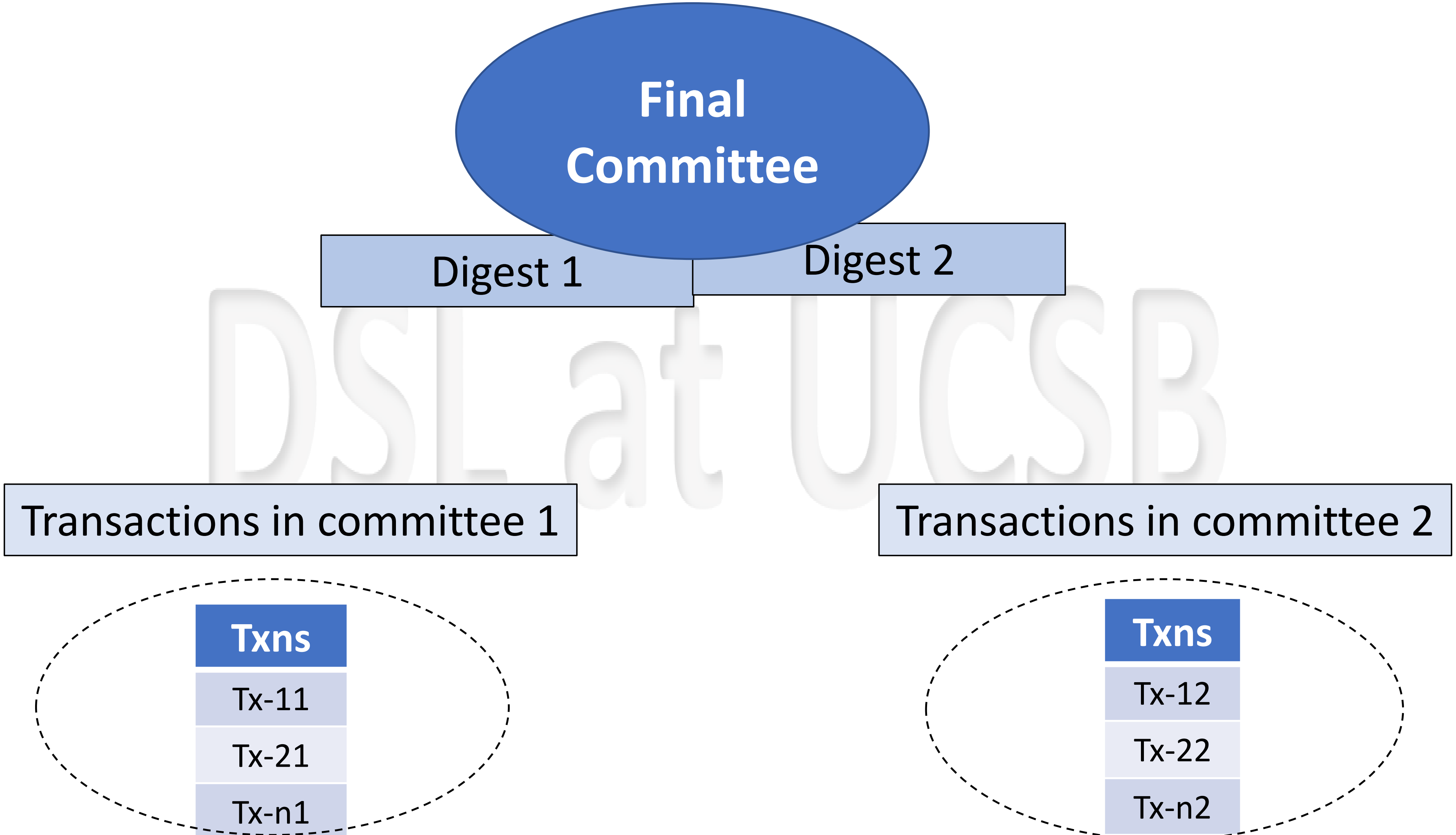


Step 4: Final Committee Union of Blocks



DSL at UCSB

Step 4: Final Committee Union of Blocks

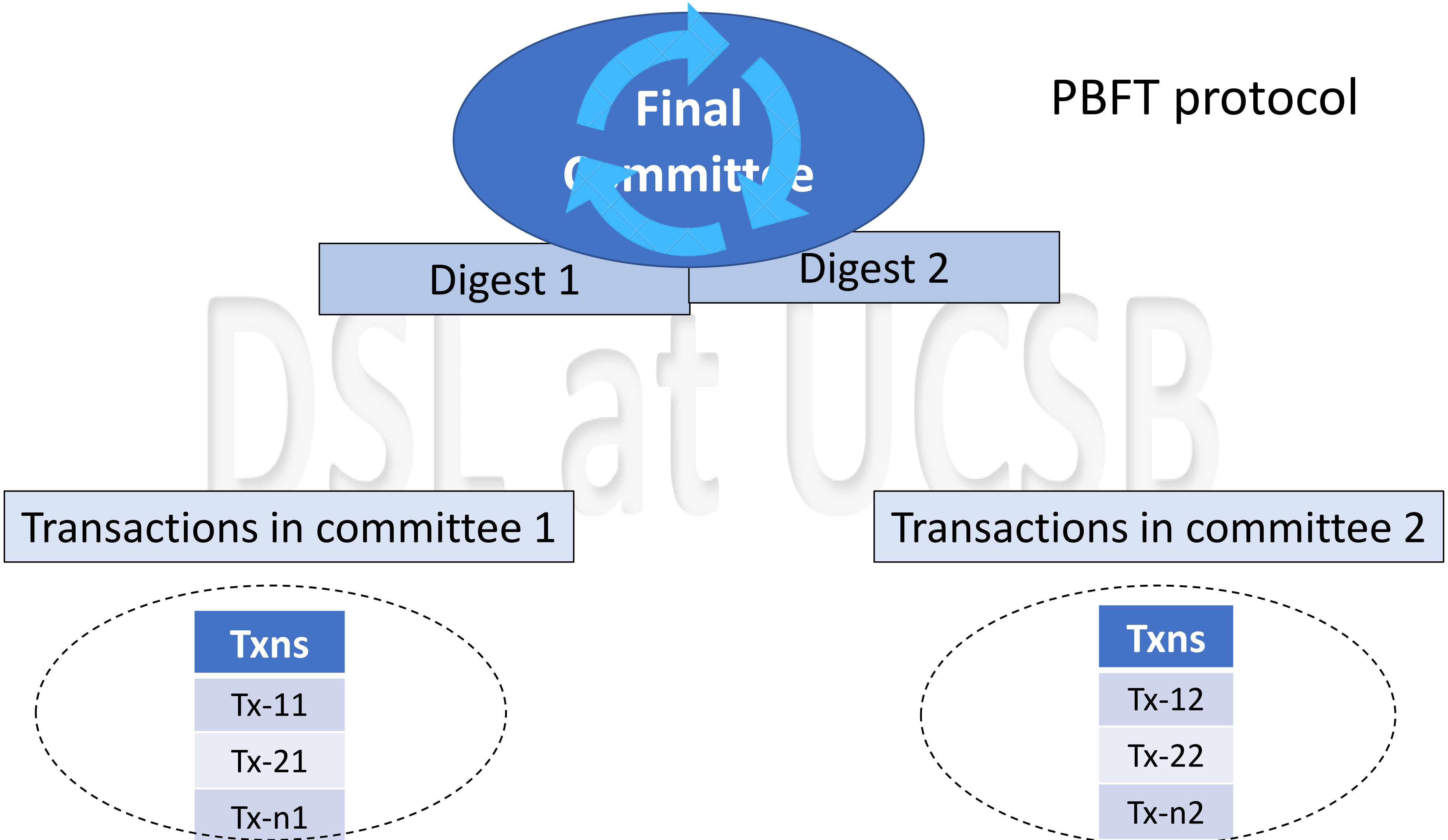


DSL at UCSB

DSL

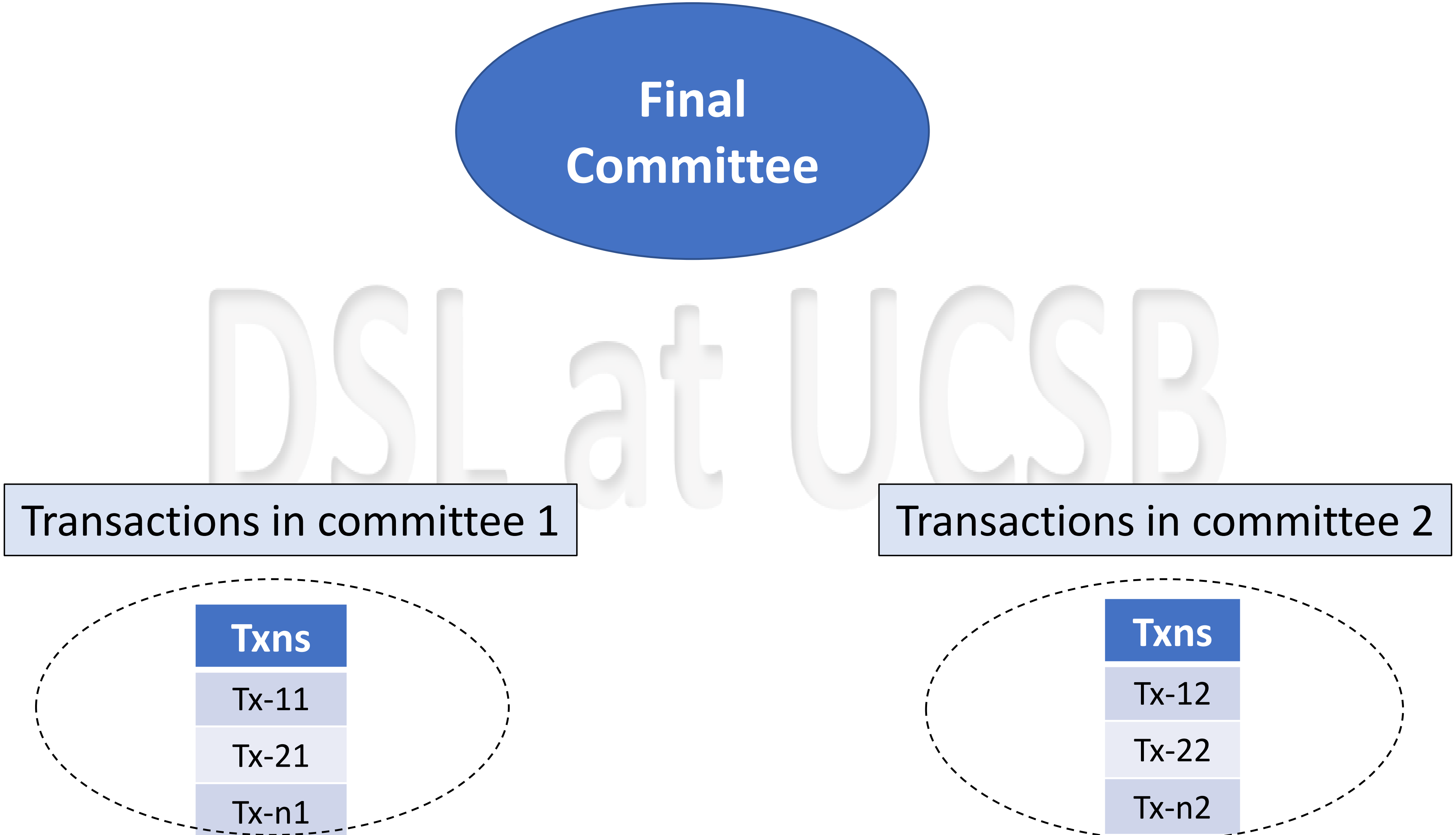
UCSB

Step 4: Final Committee Union of Blocks



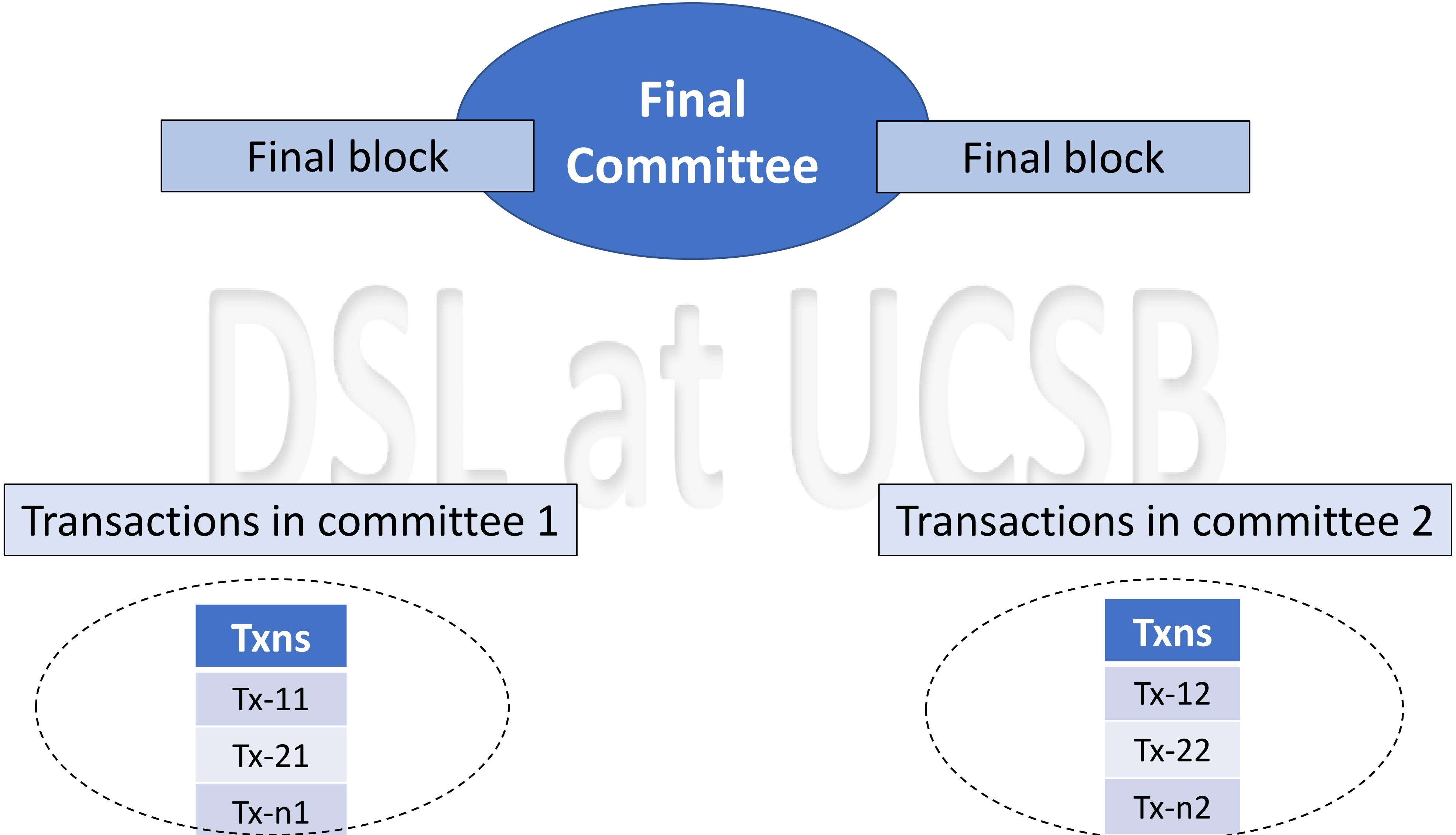
DSL at UCSB

Step 4: Final Committee Union of Blocks



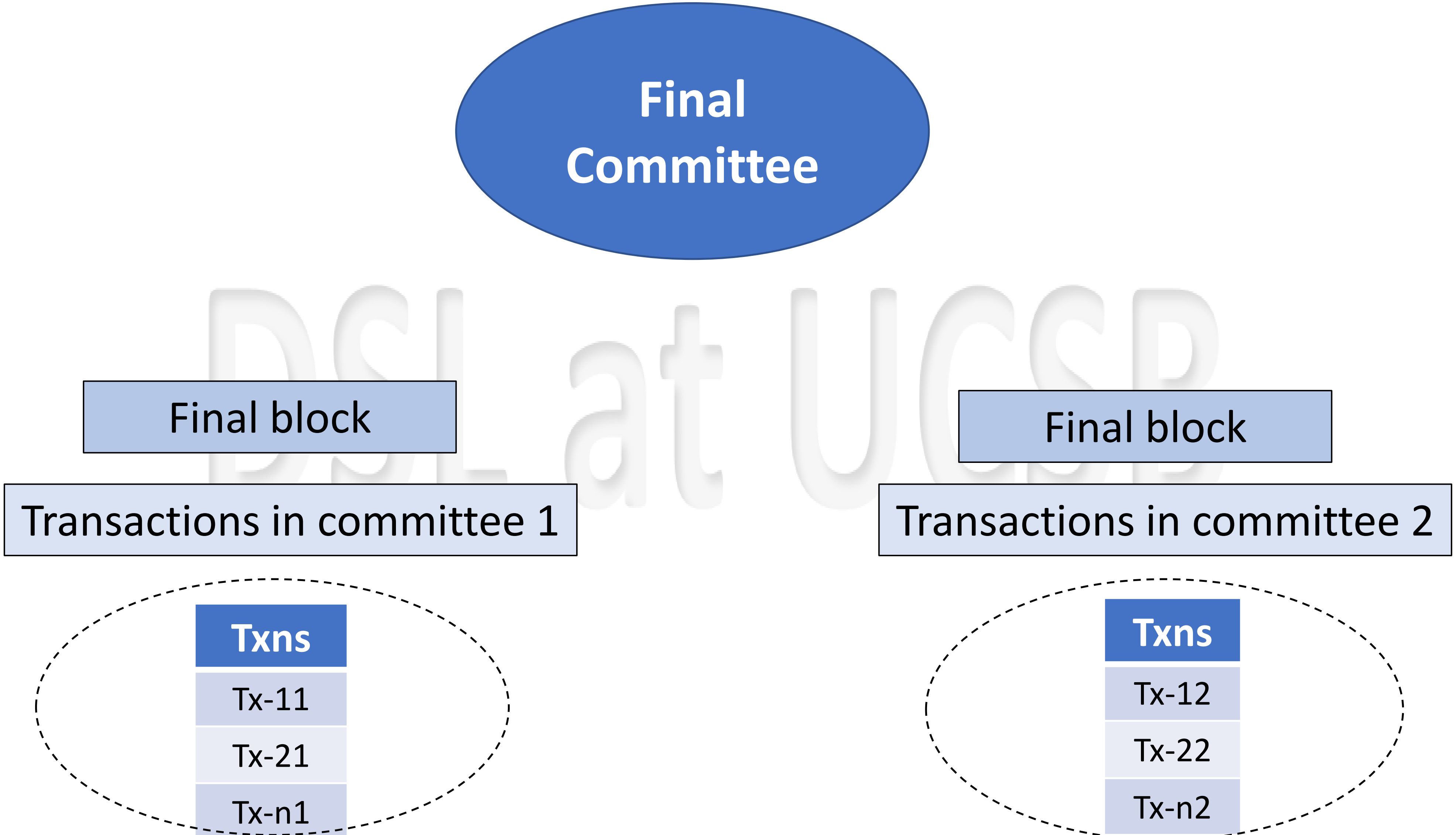
DSL at UCSB

Step 4: Final Committee Union of Blocks

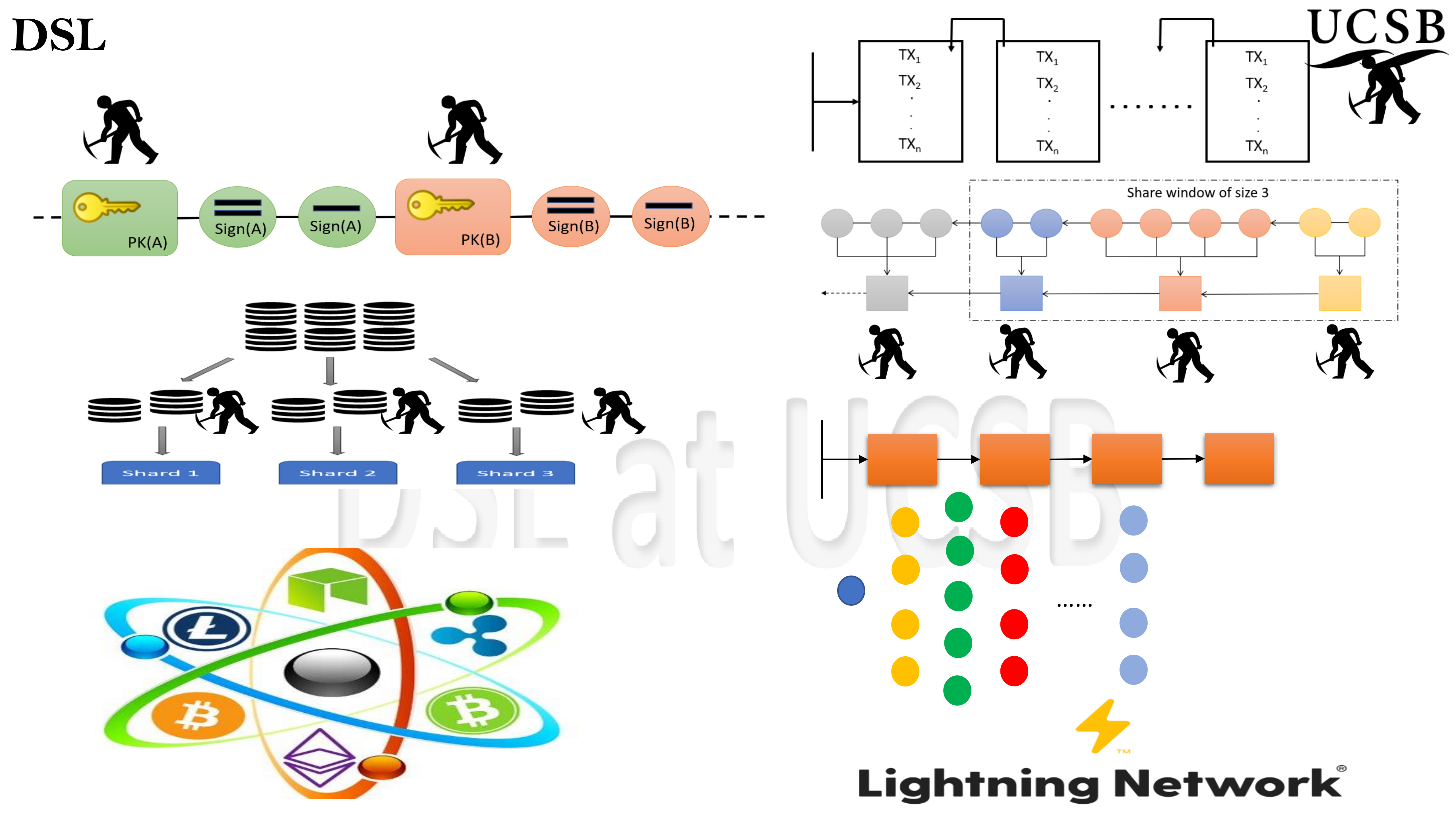


DSL at UCSB

Step 4: Final Committee Union of Blocks

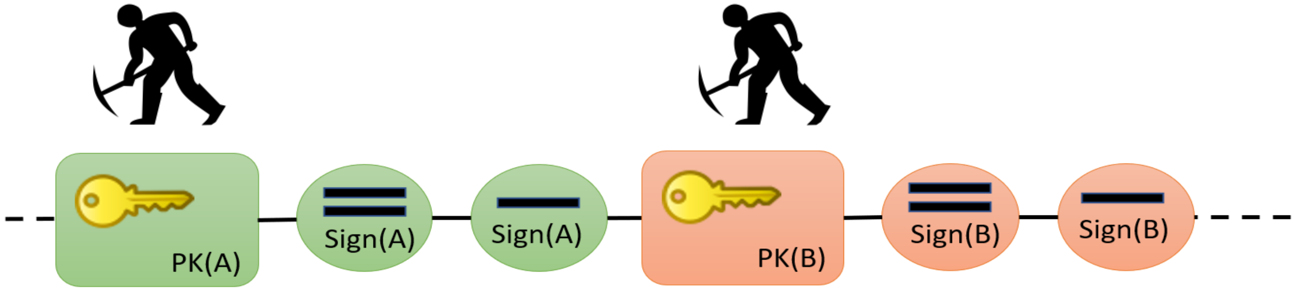


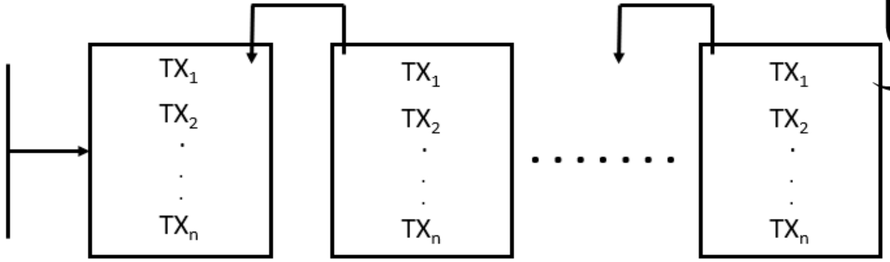
DSL at UCSB

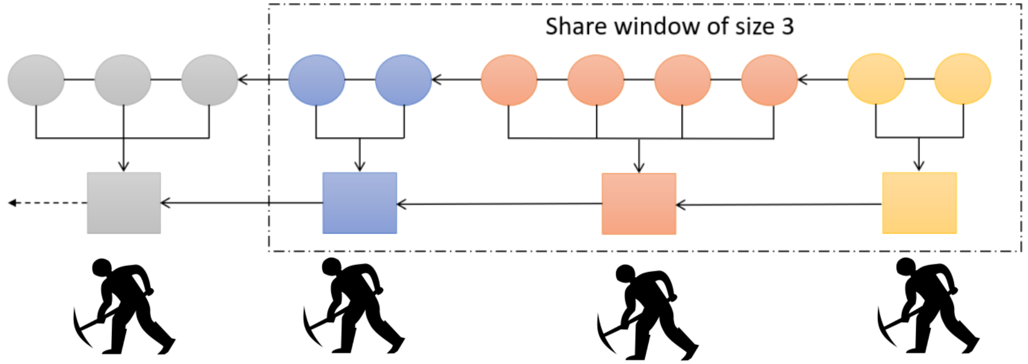

DSL
UCSB
TX1
TX2
TXn
PK(A)
Sign(A)
Sign(A)
PK(B)
Sign(B)
Sign(B)
Share window of size 3
Shard 1
Shard 2
Shard 3
DSL at UCSB
Lightning Network

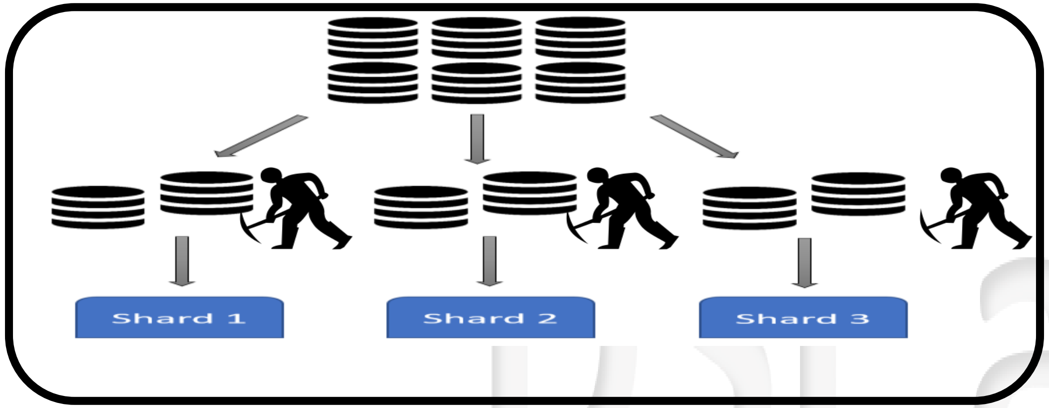
DSL

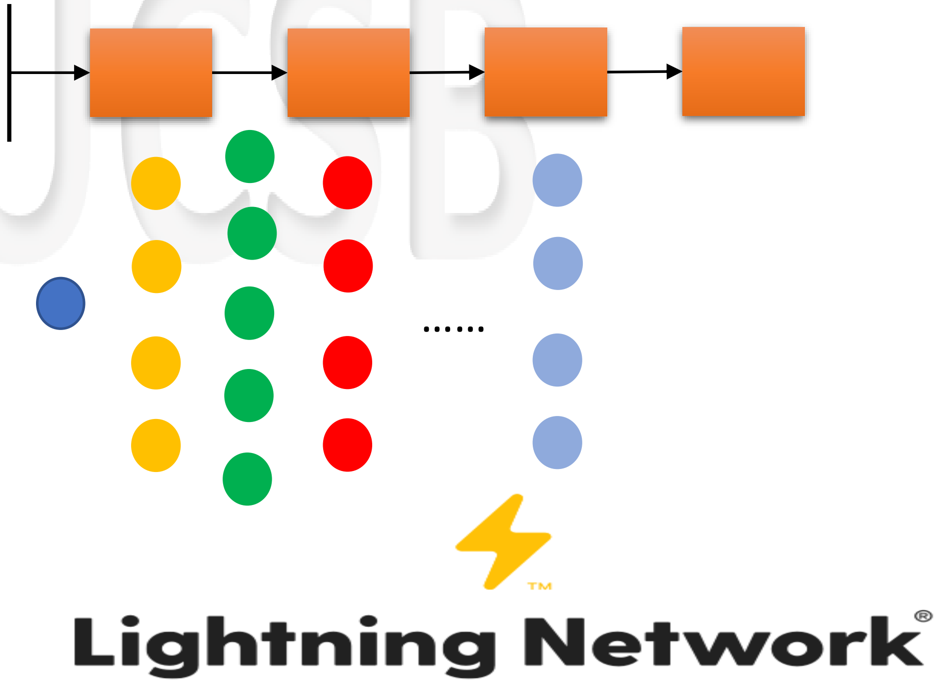
UCSB











DSL at UCSB

DSL

UCSB

SOLUTION 4

Mine once, publish txns many times	BitcoinNG
Form a committee to vouch for new block	ByzCoin
Shard txns across different committees	Elastico
Using committees with Proof-of-stake	**Algorand**

DSL at UCSB

Algorand

Scaling Byzantine Agreements for Cryptocurrencies

DSL at UCSB

Algorand

Scaling Byzantine Agreements for Cryptocurrencies

To commit txns with low latency and scale to many users by avoiding forks

DSL at UCSB

Algorand

Scaling Byzantine Agreements for Cryptocurrencies

To commit txns with low latency and scale to many users by avoiding forks

A new Byzantine Agreement protocol (**BA***) to reach consensus on the next set of txns

DSL at UCSB

Algorand

Scaling Byzantine Agreements for Cryptocurrencies

To commit txns with low latency and scale to many users by avoiding forks

DSL at UCSB

A new Byzantine Agreement protocol (**BA***) to reach consensus on the next set of txns

Gilad, Yossi, et al. "Algorand: Scaling byzantine agreements for cryptocurrencies." *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 2017.

Algorand: Goals

DSL at UCSB

Algorand: Goals

- Prevents Sybil attacks

DSL at UCSB

Algorand: Goals

- Prevents Sybil attacks
  → By using **Weighted users** proportional to money in their account

DSL at UCSB

Algorand: Goals

- Prevents Sybil attacks
  → By using **Weighted users** proportional to money in their account
- Scalability

DSL at UCSB

Algorand: Goals

- Prevents Sybil attacks
  → By using **Weighted users** proportional to money in their account

- Scalability
  → Use of **BA***: Runs consensus on a small set of nodes

DSL at UCSB

Algorand: Goals

- Prevents Sybil attacks
  → By using **Weighted users** proportional to money in their account

- Scalability
  → Use of **BA***: Runs consensus on a small set of nodes

- Resilient to denial of service

DSL at UCSB

Algorand: Goals

- Prevents Sybil attacks
  → By using **Weighted users** proportional to money in their account
- Scalability
  → Use of **BA***: Runs consensus on a small set of nodes
- Resilient to denial of service
  → Randomly choose committee using **Cryptographic Sortition** based on weight

DSL at UCSB

Algorand: Goals

- Prevents Sybil attacks
  - → By using **Weighted users** proportional to money in their account
- Scalability
  - → Use of **BA***: Runs consensus on a small set of nodes
- Resilient to denial of service
  - → Randomly choose committee using **Cryptographic Sortition** based on weight
  - → **Replace** participants after each round

DSL at UCSB

Algorand: Assumptions

DSL at UCSB

Algorand: Assumptions

- Honest majority of **money $$**

DSL at UCSB

DSL

UCSB

Algorand: Assumptions

- Honest majority of **money $$**

- An adversary cannot manipulate the network at large scale

DSL at UCSB

Algorand: Assumptions

- Honest majority of **money $$**
- An adversary cannot manipulate the network at large scale
- **Strong synchrony**
  Tolerates temporary asynchronous network but must be followed by a longer synchronous network

DSL at UCSB

Algorand: Overview

DSL at UCSB

Algorand: Overview

- Gossip protocol

DSL at UCSB

Algorand: Overview

- Gossip protocol

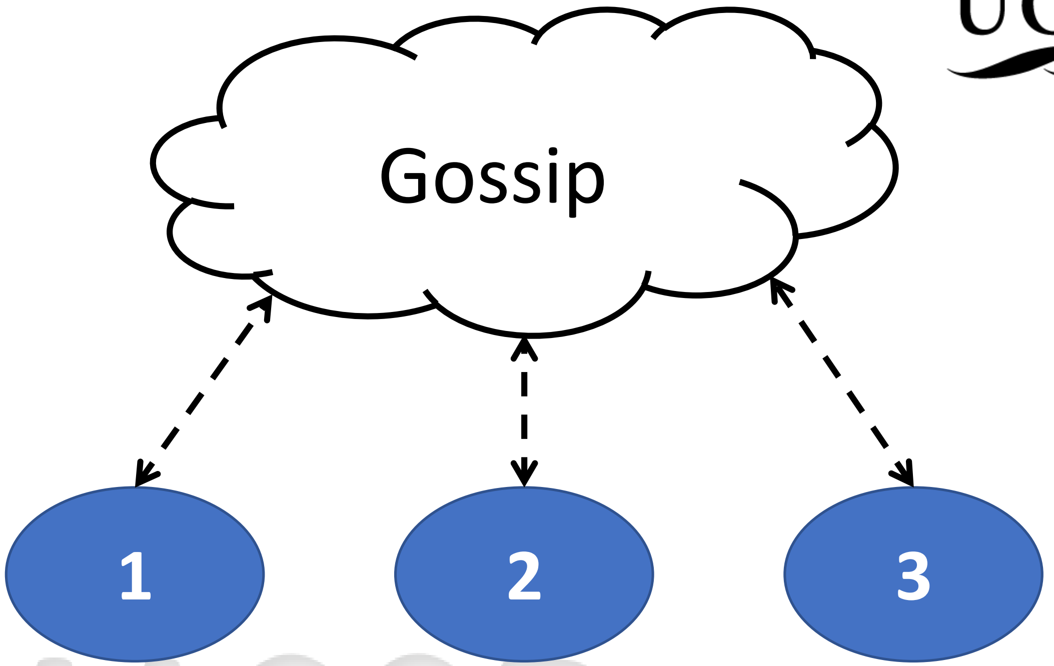
DSL at UCSB

DSL

UCSB

Algorand: Overview

- Gossip protocol



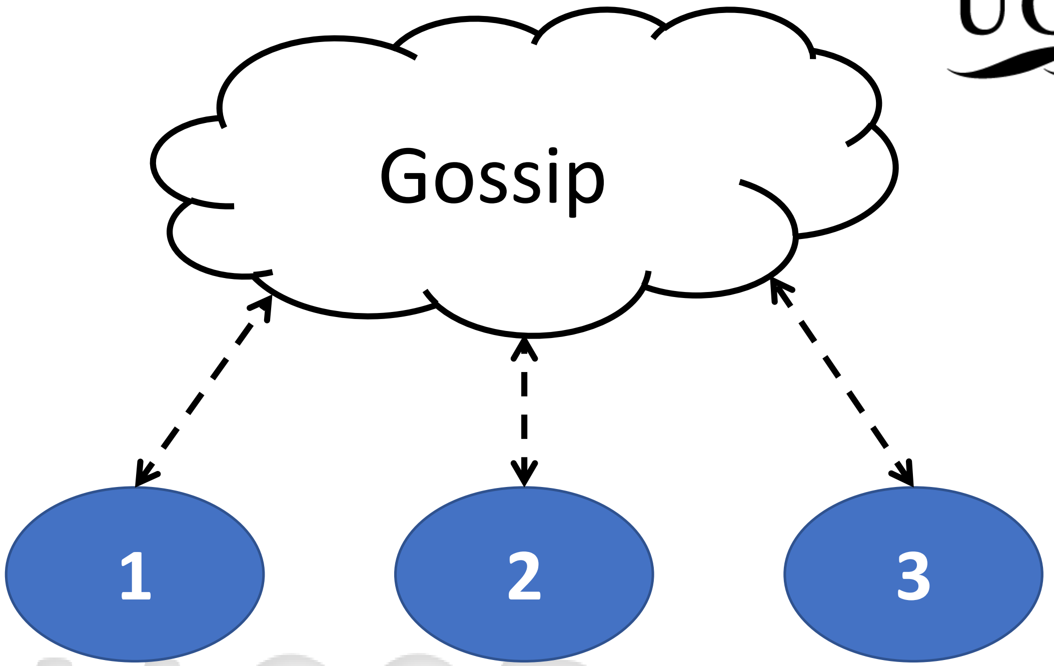
DSL at UCSB

DSL

UCSB

Algorand: Overview

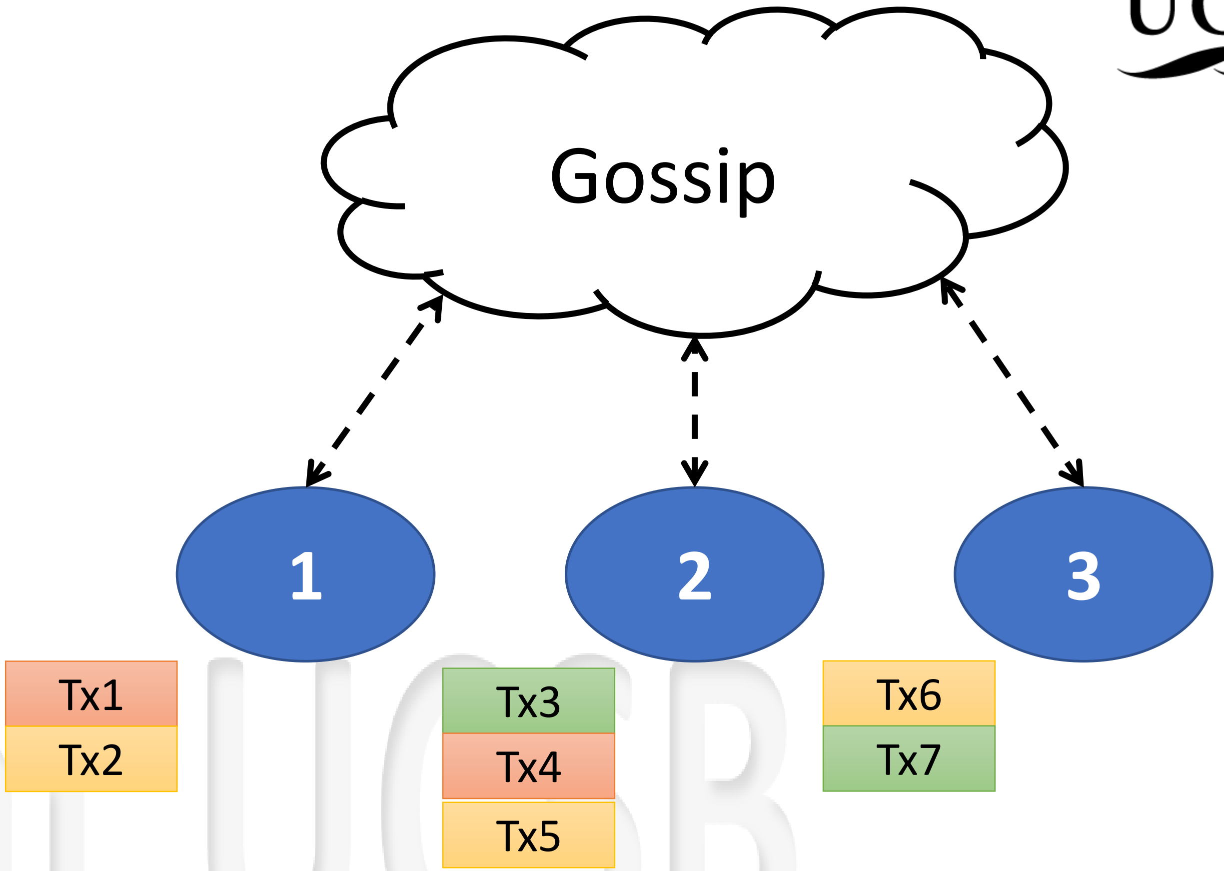
- Gossip protocol
  - Each node collects pending txns



DSL at UCSB

Algorand: Overview

- Gossip protocol
  - Each node collects pending txns



DSL at UCSB

Algorand: Overview

- Gossip protocol
- Block proposal

DSL at UCSB

Algorand: Overview

- Gossip protocol
- Block proposal

DSL at UCSB

Algorand: Overview

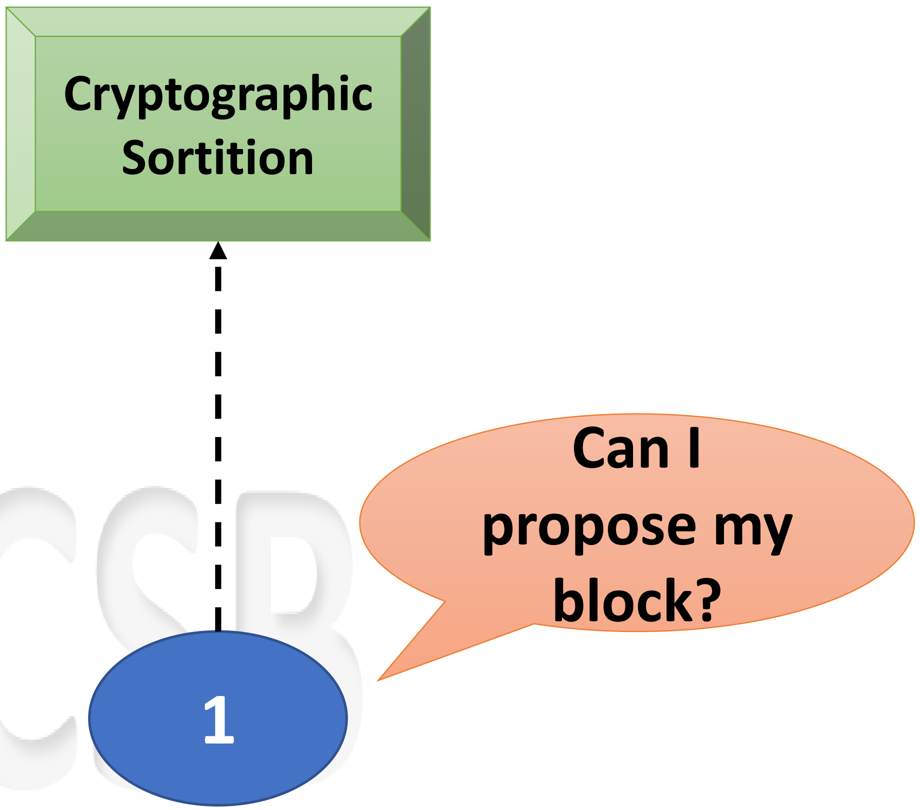
- Gossip protocol

- Block proposal
  - Sortition ensures **small fraction** of users selected based on their **weights**

DSL at UCSB

Algorand: Overview

- Gossip protocol
- Block proposal
  - Sortition ensures **small fraction** of users selected based on their **weights**



DSL at UCSB

Algorand: Overview

- Gossip protocol
- Block proposal
  - Sortition ensures **small fraction** of users selected based on their **weights**

DSL at UCSB

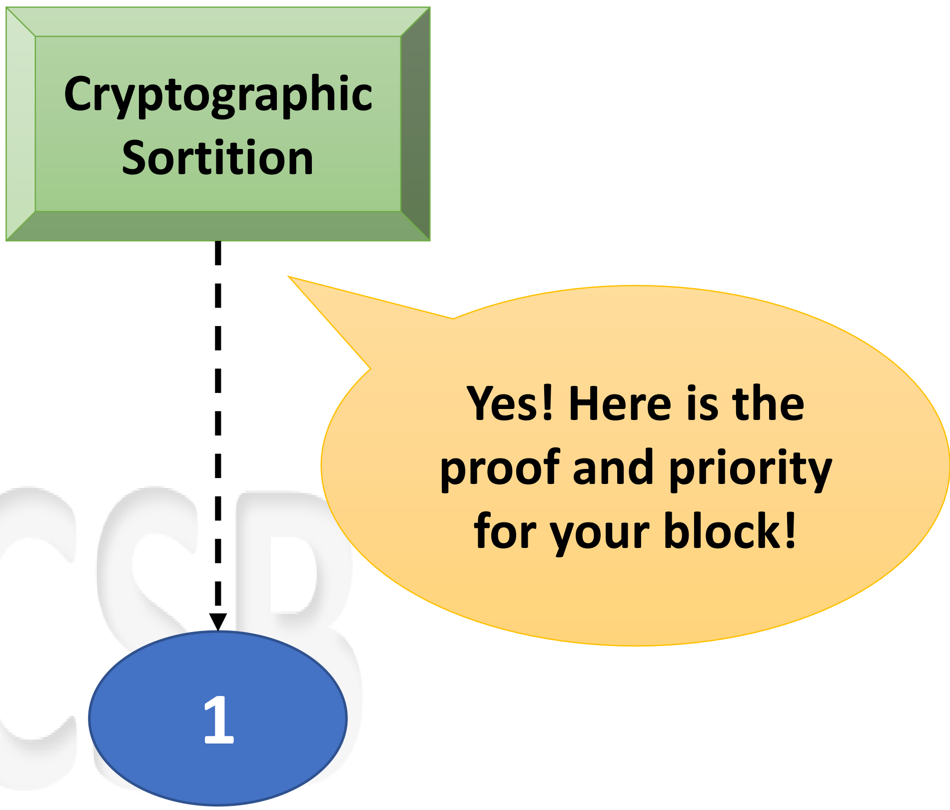
Algorand: Overview

- Gossip protocol
- Block proposal
  - Sortition ensures **small fraction** of users selected based on their **weights**
  - Provides ***proof*** of selection and ***priority*** for each block

DSL at UCSB

Algorand: Overview

- Gossip protocol
- Block proposal
  - Sortition ensures **small fraction** of users selected based on their **weights**
  - Provides ***proof*** of selection and ***priority*** for each block



DSL at UCSB

Algorand: Overview

- Gossip protocol
- Block proposal
  - Sortition ensures **small fraction** of users selected based on their **weights**
  - Provides ***proof*** of selection and ***priority*** for each block

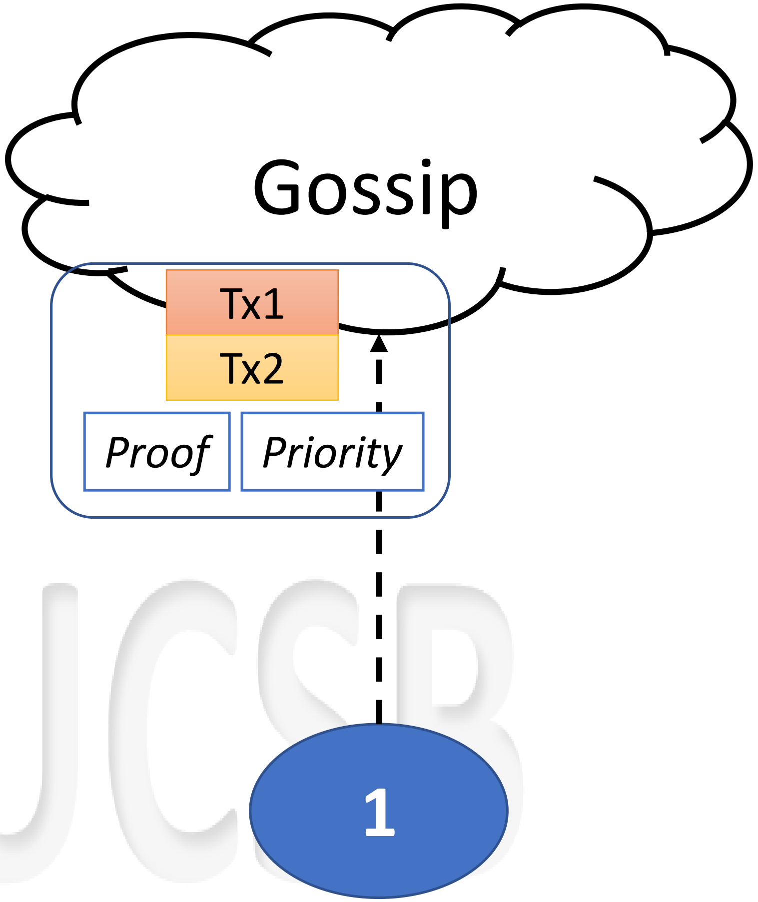
DSL at UCSB

DSL

UCSB

Algorand: Overview

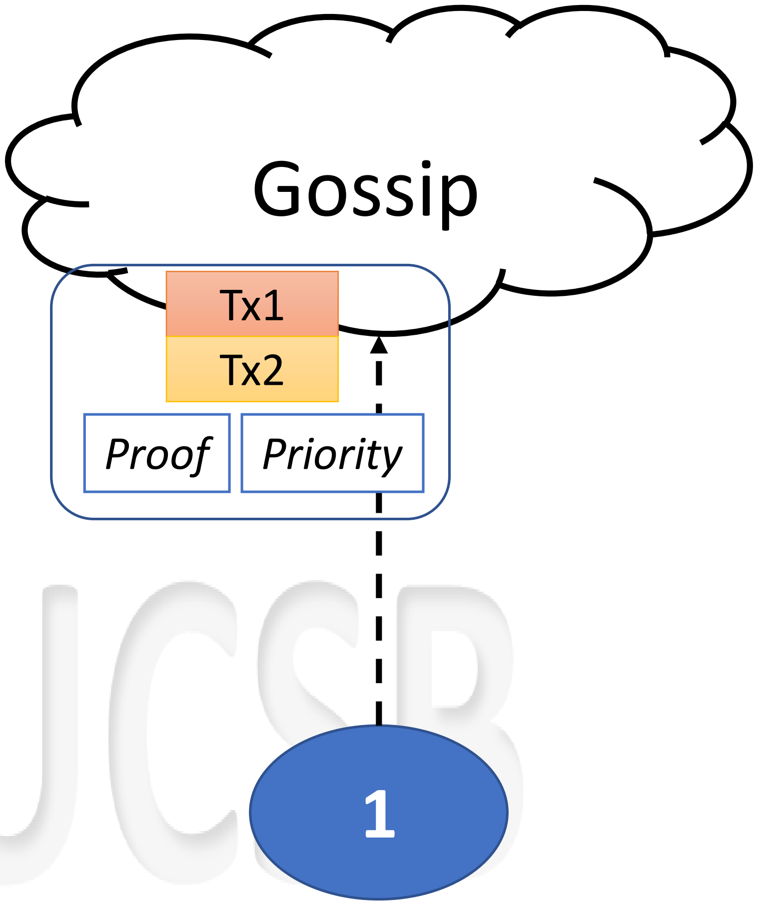
- Gossip protocol
- Block proposal
  - Sortition ensures **small fraction** of users selected based on their **weights**
  - Provides ***proof*** of selection and ***priority*** for each block



DSL at UCSB

Algorand: Overview

- Gossip protocol
- Block proposal
  - Sortition ensures **small fraction** of users selected based on their **weights**
  - Provides ***proof*** of selection and ***priority*** for each block
- Agreement using BA*



DSL at UCSB

BA* Overview

DSL at UCSB

BA* Overview

- Two phase protocol

DSL at UCSB

BA* Overview

- Two phase protocol
  - **Phase 1:** 2 steps

DSL at UCSB

BA* Overview

- Two phase protocol
  - **Phase 1:** 2 steps
  - **Phase 2:** 2 – 11 steps

DSL at UCSB

BA* Overview

- Two phase protocol
  - **Phase 1:** 2 steps
  - **Phase 2:** 2 – 11 steps

- Each step calls **Sortition** to create a committee

DSL at UCSB

BA* Overview

- Two phase protocol
  - **Phase 1:** 2 steps
  - **Phase 2:** 2 – 11 steps
- Each step calls **Sortition** to create a committee
- Each committee member will **broadcast** their **vote** for their block

DSL at UCSB

BA* Overview

- Two phase protocol
  - **Phase 1:** 2 steps
  - **Phase 2:** 2 – 11 steps

- Each step calls **Sortition** to create a committee

- Each committee member will **broadcast** their **vote** for their block
  - Vote for **highest priority** block

DSL at UCSB

BA* Overview

- Two phase protocol
  - **Phase 1:** 2 steps
  - **Phase 2:** 2 – 11 steps

- Each step calls **Sortition** to create a committee

- Each committee member will **broadcast** their **vote** for their block
  - Vote for **highest priority** block
  - All users can see this message

DSL at UCSB

DSL

UCSB

BA* Overview

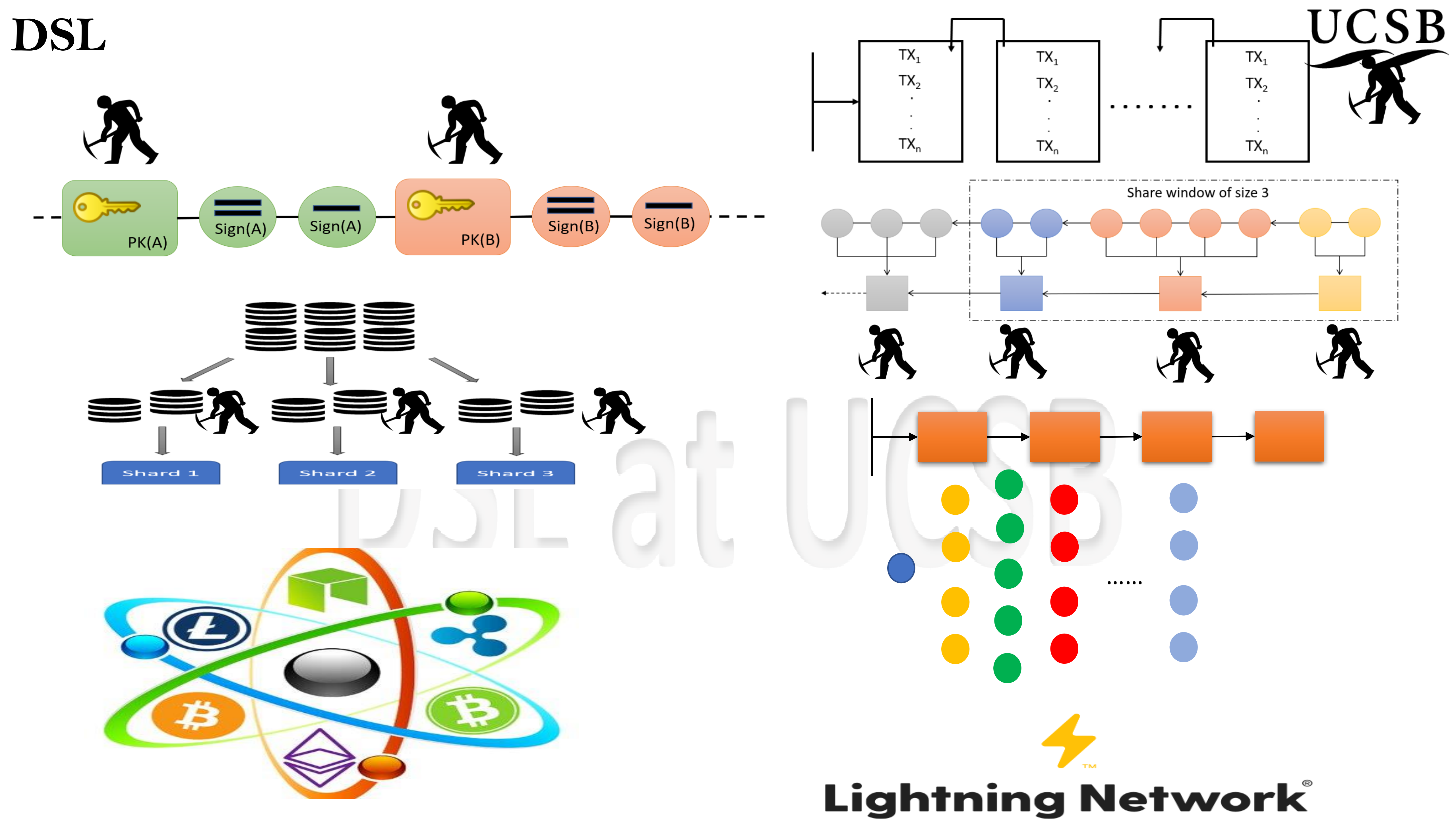
- Two phase protocol
  - **Phase 1:** 2 steps
  - **Phase 2:** 2 – 11 steps

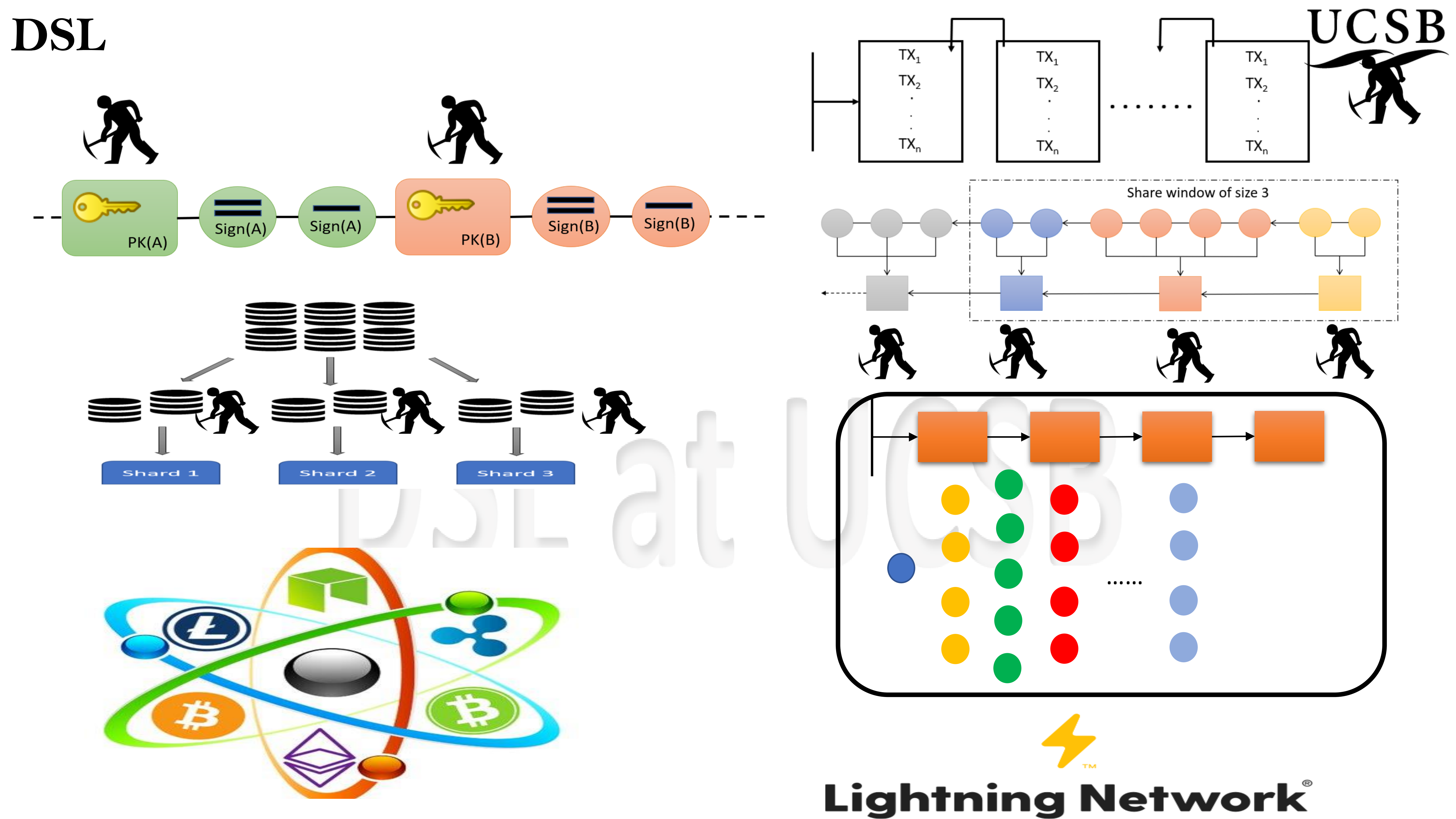
- Each step calls **Sortition** to create a committee

- Each committee member will **broadcast** their **vote** for their block
  - Vote for **highest priority** block
  - All users can see this message

- Users that receive more than a **threshold** of votes for a block will **hold onto** that block

DSL at UCSB


DSL
UCSB
TX_1
TX_2
TX_n
PK(A)
Sign(A)
Sign(A)
PK(B)
Sign(B)
Sign(B)
Share window of size 3
Shard 1
Shard 2
Shard 3
DSL at UCSB
Lightning Network

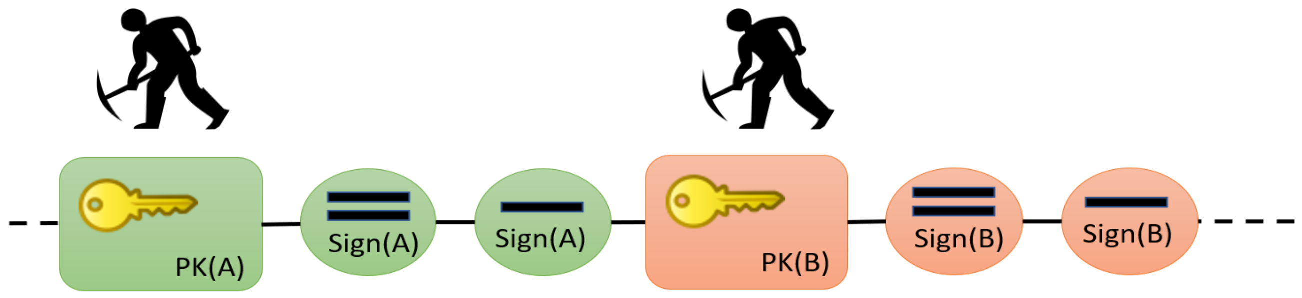

DSL
UCSB
PK(A)
Sign(A)
Sign(A)
PK(B)
Sign(B)
Sign(B)
Shard 1
Shard 2
Shard 3
DSL at UCSB
TX1
TX2
TXn
Share window of size 3
Lightning Network

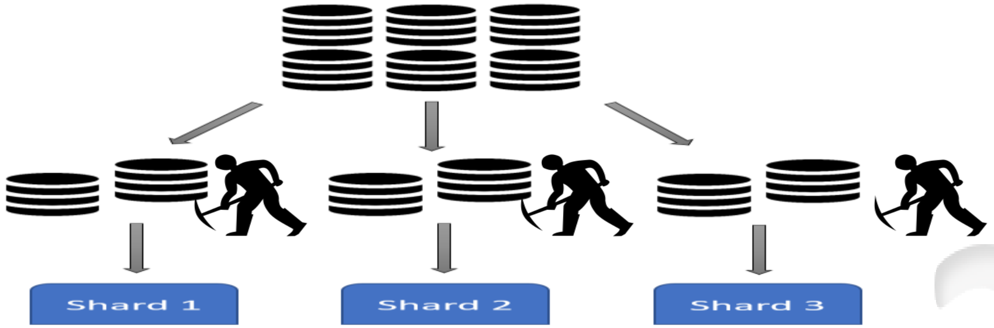
ATOMIC SWAPS

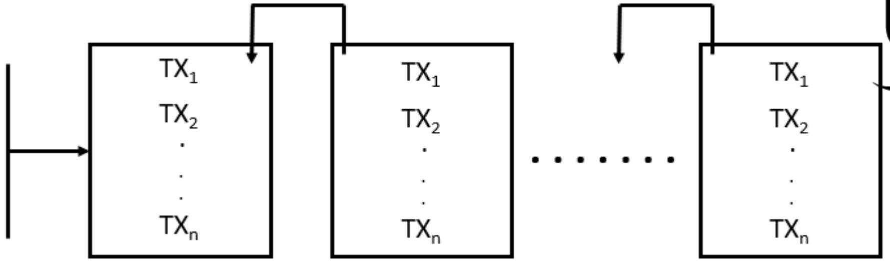
DSL at UCSB

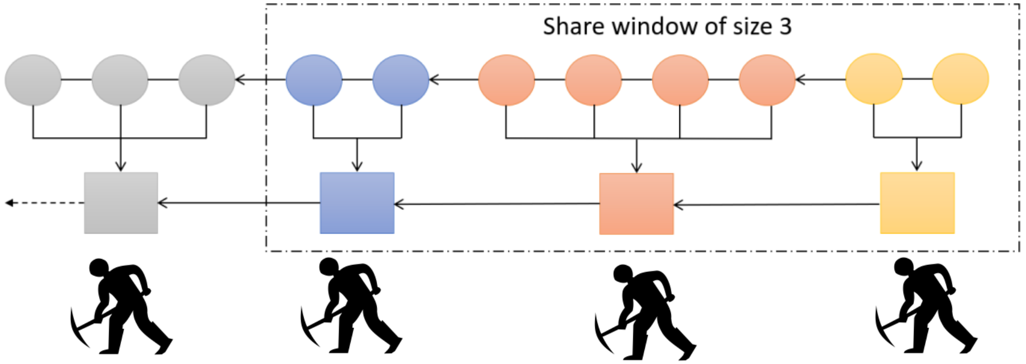
DSL

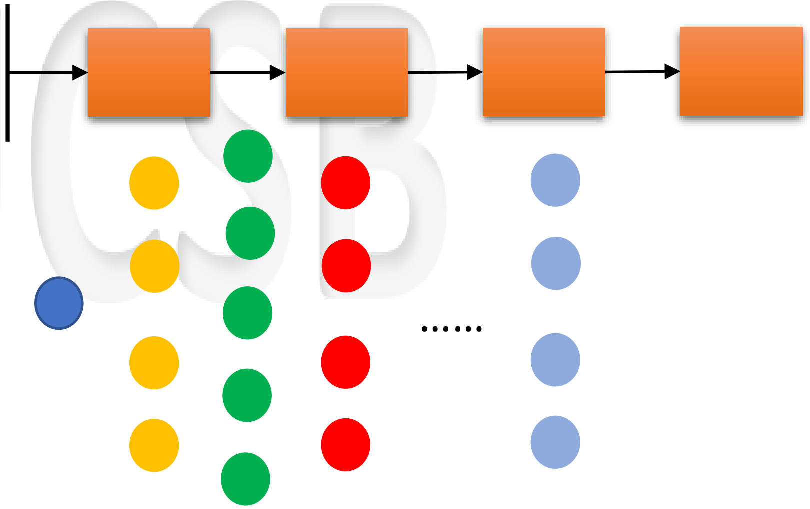
UCSB



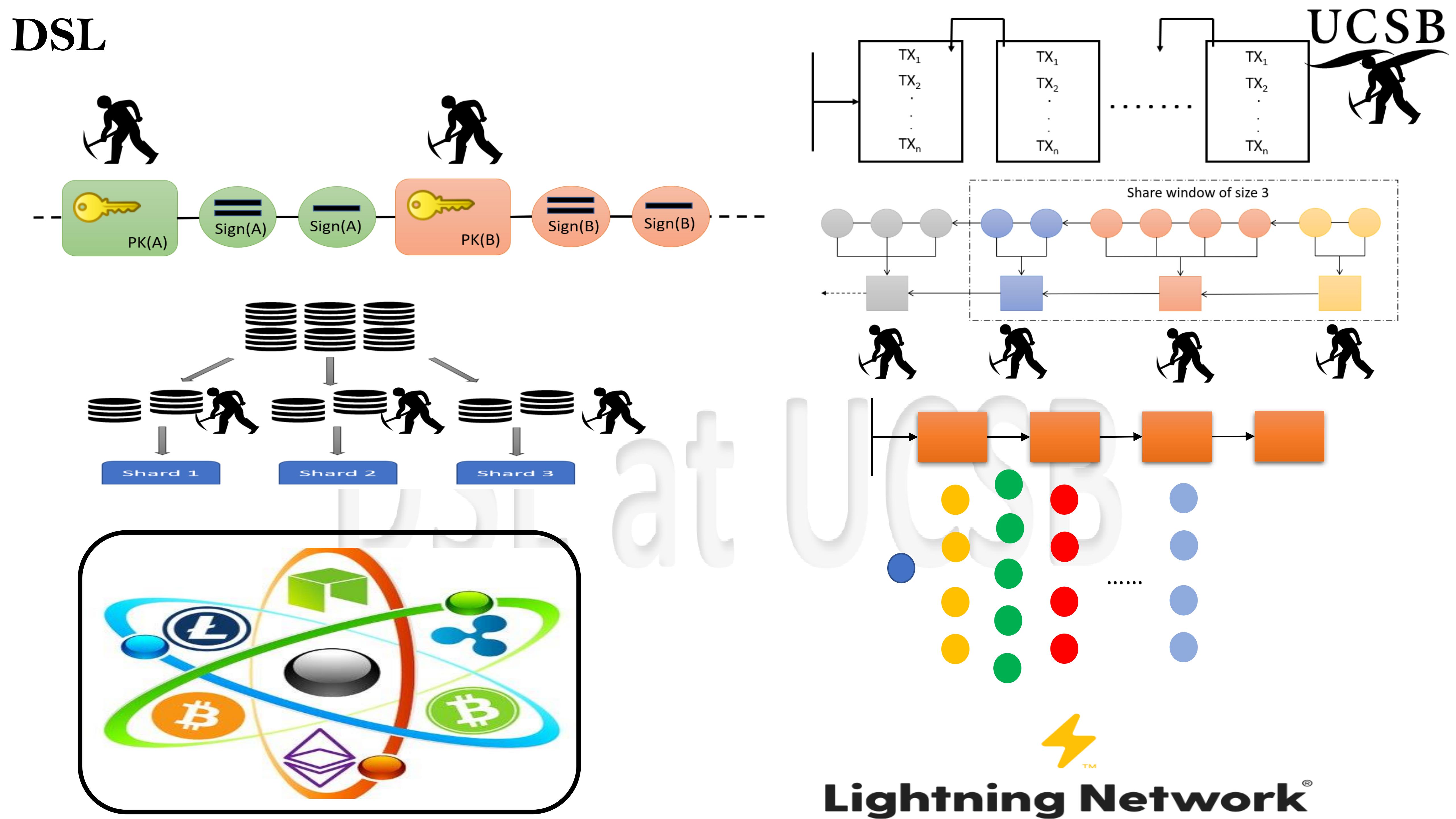








Lightning Network®


DSL
UCSB
TX_1
TX_2
TX_n
Share window of size 3
PK(A)
Sign(A)
Sign(A)
PK(B)
Sign(B)
Sign(B)
Shard 1
Shard 2
Shard 3
DSL at UCSB
Lightning Network

Atomic Swaps

- Allow transactions to span multiple blockchains
  - E.g., swap Bitcoin with Ethereum
- The goal:
  - Swap assets across multiple blockchains
- If all parties conform to the protocol:
  - All swaps take place
  - If some coalition deviates from the protocol, then no conforming party ends up worse off
  - No coalition has an incentive to deviate from the protocol

DSL at UCSB

[TierNolan](https://bitcointalk.org/index.php?action=profile;u=105224), **Atomic swap using cut and choose,** https://bitcointalk.org/index.php?topic=193281.msg2224949#msg2224949 **(2013)**
Herlihy, Maurice. "Atomic cross-chain swaps." *PODC 2018*

Atomic Swaps

- Exchanges enable trading among different cryptocurrenices
  - Usually happens through USD ($)
- Exchanges make the system centralised
- Atomic swaps allow trading different assets without an arbiter
- Atomic swaps use:
  - Smart Contracts
  - Hashlocks
  - Timelocks

DSL at UCSB

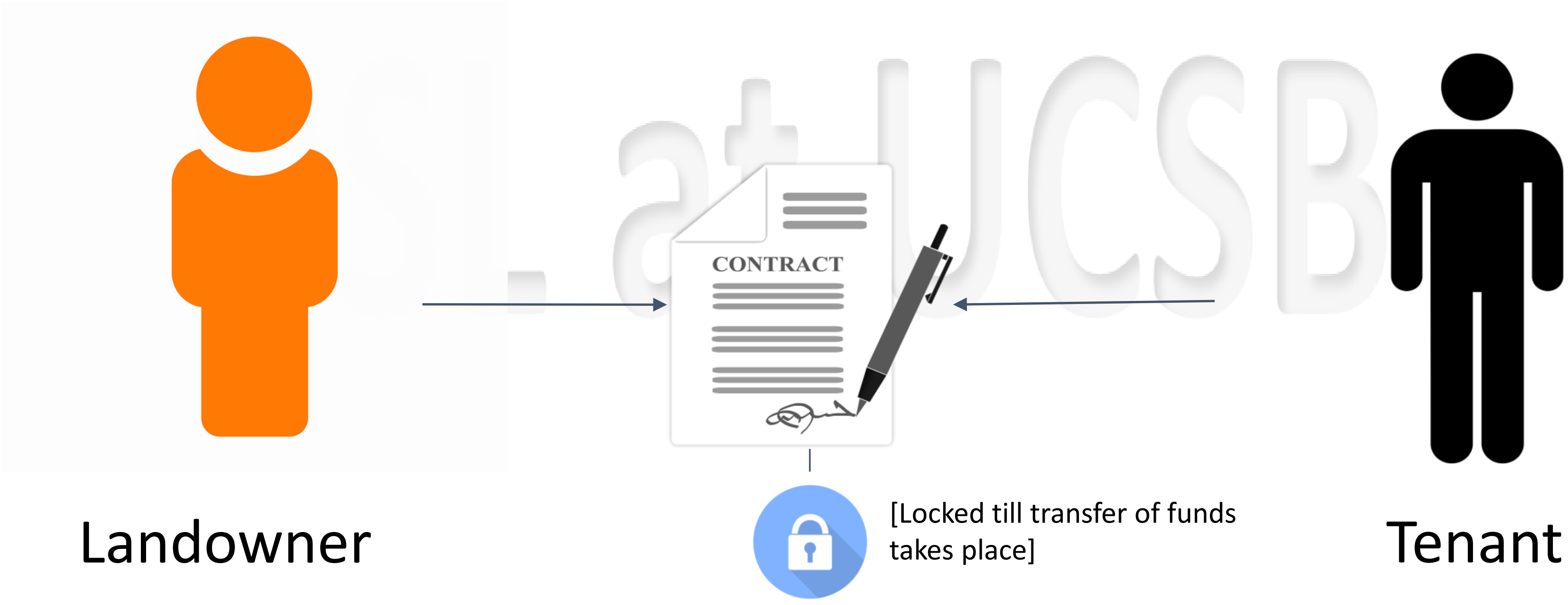
Smart Contracts

- Digital self-executing contract
- Stores rules for negotiating the terms of an agreement
- Automatically verifies fulfillment, and then executes the agreed terms
- E.g., move 10 Bitcoins from Alice to Bob if Bob provides a secret (s)
- Contracts are published in the blockchain
- Contracts are executed if its conditions are met
  - Bob provides secret (s) to the contract

DSL at UCSB

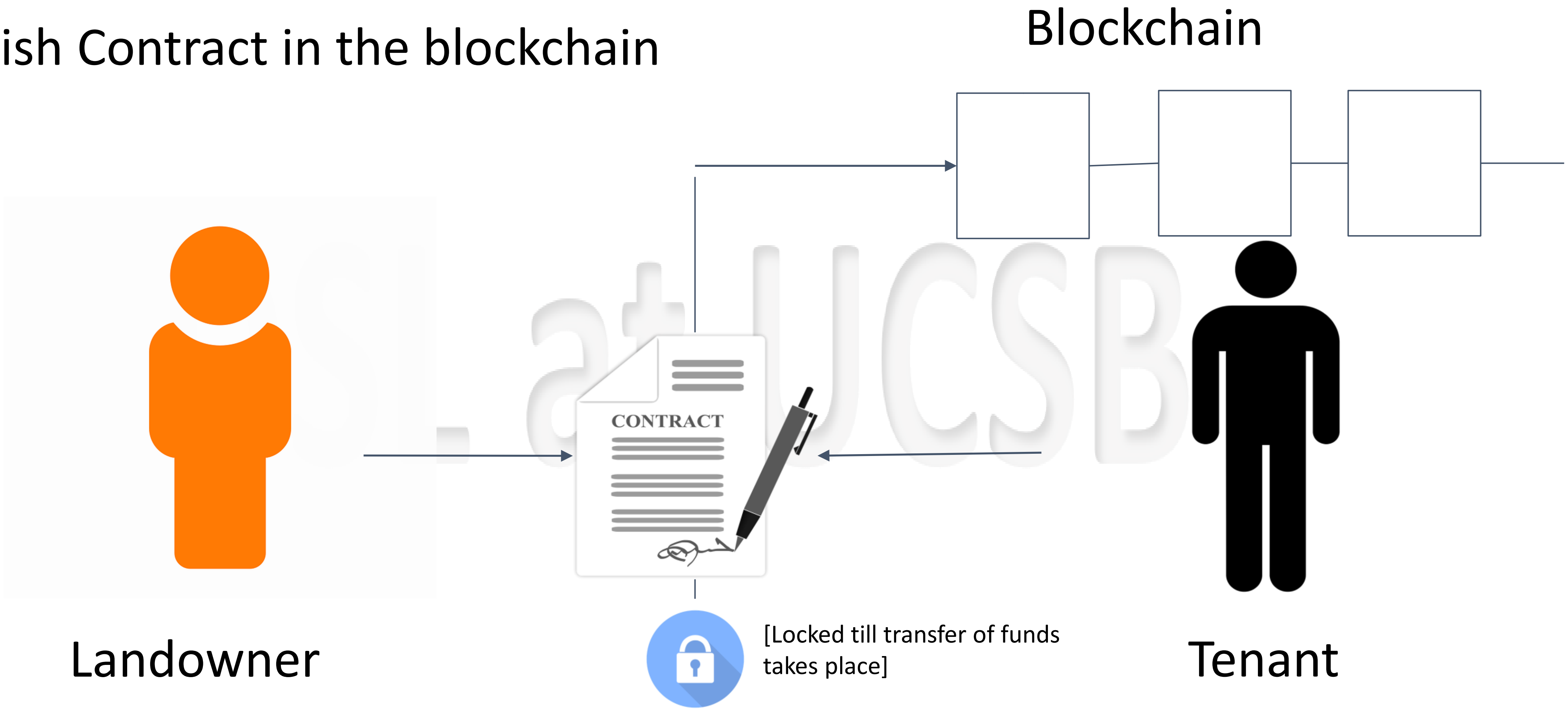
Example

- Landowner wants to rent out her place to a tenant
  - Send house unlock code to Tenant if they transfer funds to landowner



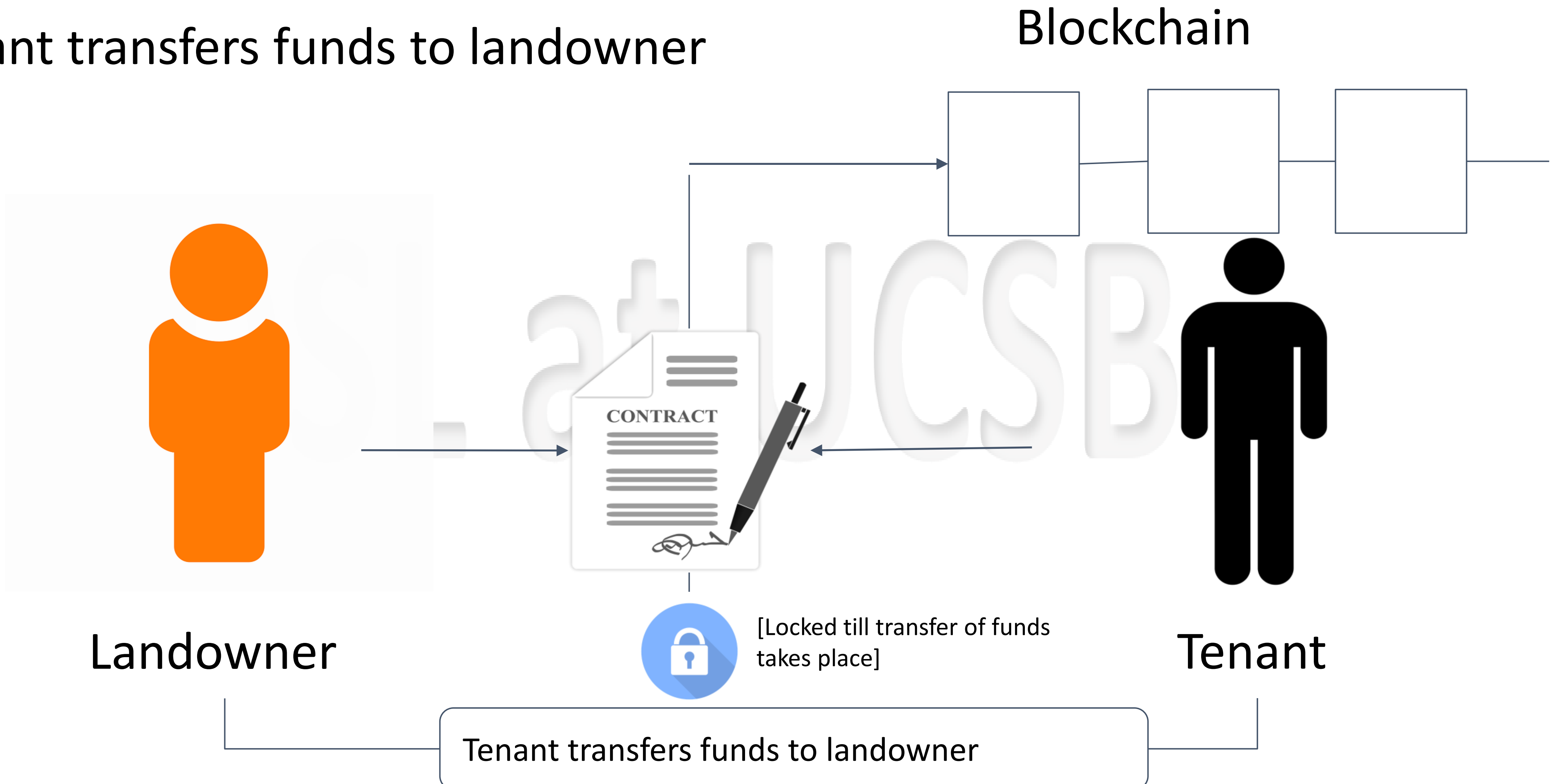
Example

- Publish Contract in the blockchain



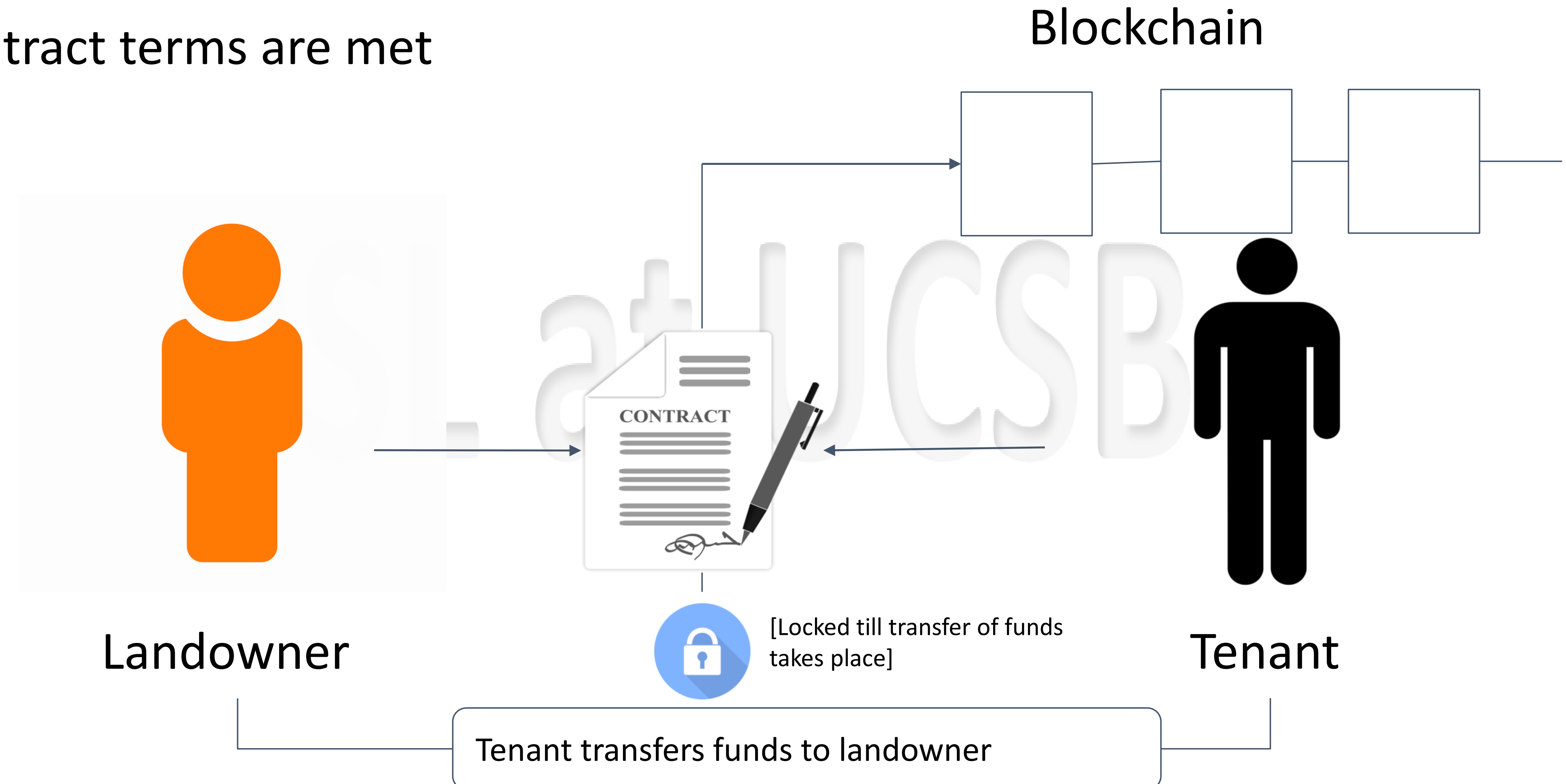
Example

- Tenant transfers funds to landowner



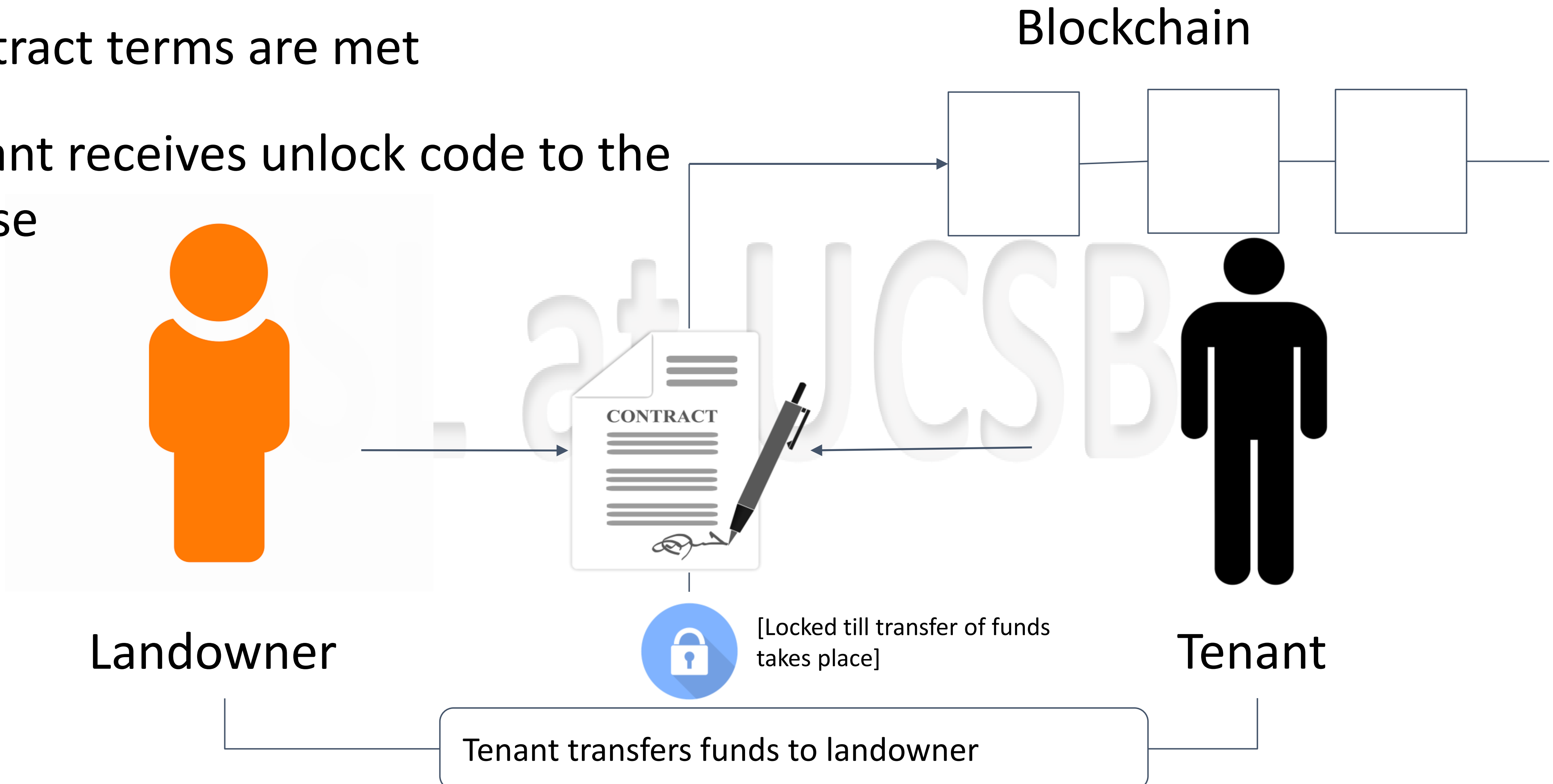
Example

- Contract terms are met



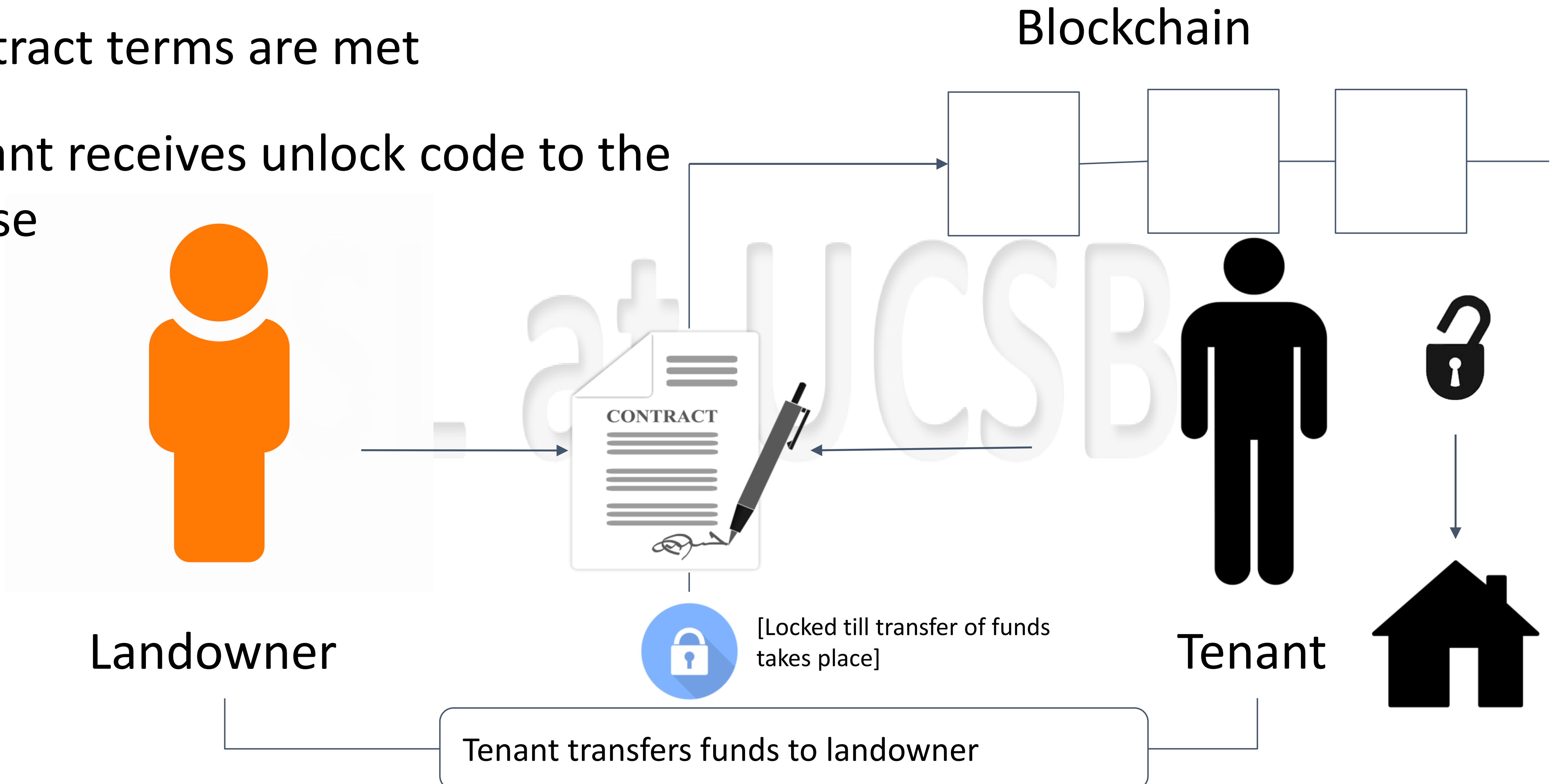
Example

- Contract terms are met
- Tenant receives unlock code to the house



Example

- Contract terms are met
- Tenant receives unlock code to the house



Hashlocks and Timelocks

- Hashlock h

DSL at UCSB

Hashlocks and Timelocks

- Hashlock h
  - Transfer X Bitcoins from Alice to Bob if Bob provides a secret **s** such that **h = H(s)**
  - H is a cryptographic one-way hash function
  - The contract irrevocably transfers ownership of X Bitcoins from Alice to Bob

DSL at UCSB

Hashlocks and Timelocks

- Hashlock h
  - Transfer X Bitcoins from Alice to Bob if Bob provides a secret **s** such that **h = H(s)**
  - H is a cryptographic one-way hash function
  - The contract irrevocably transfers ownership of X Bitcoins from Alice to Bob
- Timelock t

DSL at UCSB

Hashlocks and Timelocks

- Hashlock h
  - Transfer X Bitcoins from Alice to Bob if Bob provides a secret **s** such that **h = H(s)**
  - H is a cryptographic one-way hash function
  - The contract irrevocably transfers ownership of X Bitcoins from Alice to Bob
- Timelock t
  - If Bob fails to produce that **s** before time **t** elapses, then X Bitcoins are refunded to Alice

DSL at UCSB

Atomic Swap Example

- Alice wants to trade Bitcoin for Ethereum with Bob

DSL at UCSB

Atomic Swap Example

- Alice wants to trade Bitcoin for Ethereum with Bob

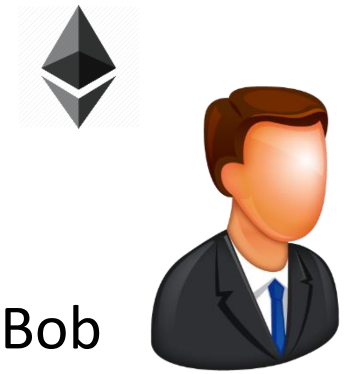
DSL at UCSB

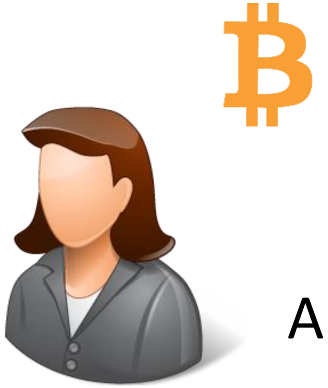
Atomic Swap Example

- Alice wants to trade Bitcoin for Ethereum with Bob

DSL at UCSB

- Create a secret s
- Calculate its hash h = H(s)





Atomic Swap Example

- Alice wants to trade Bitcoin for Ethereum with Bob

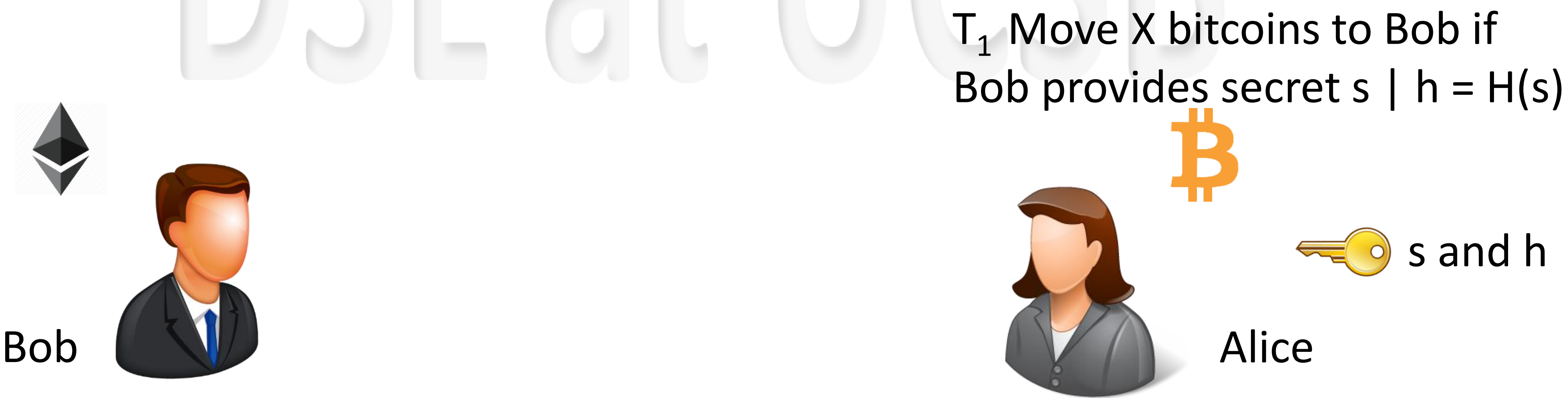
DSL at UCSB

- Create a secret s
- Calculate its hash h = H(s)

Atomic Swap Example

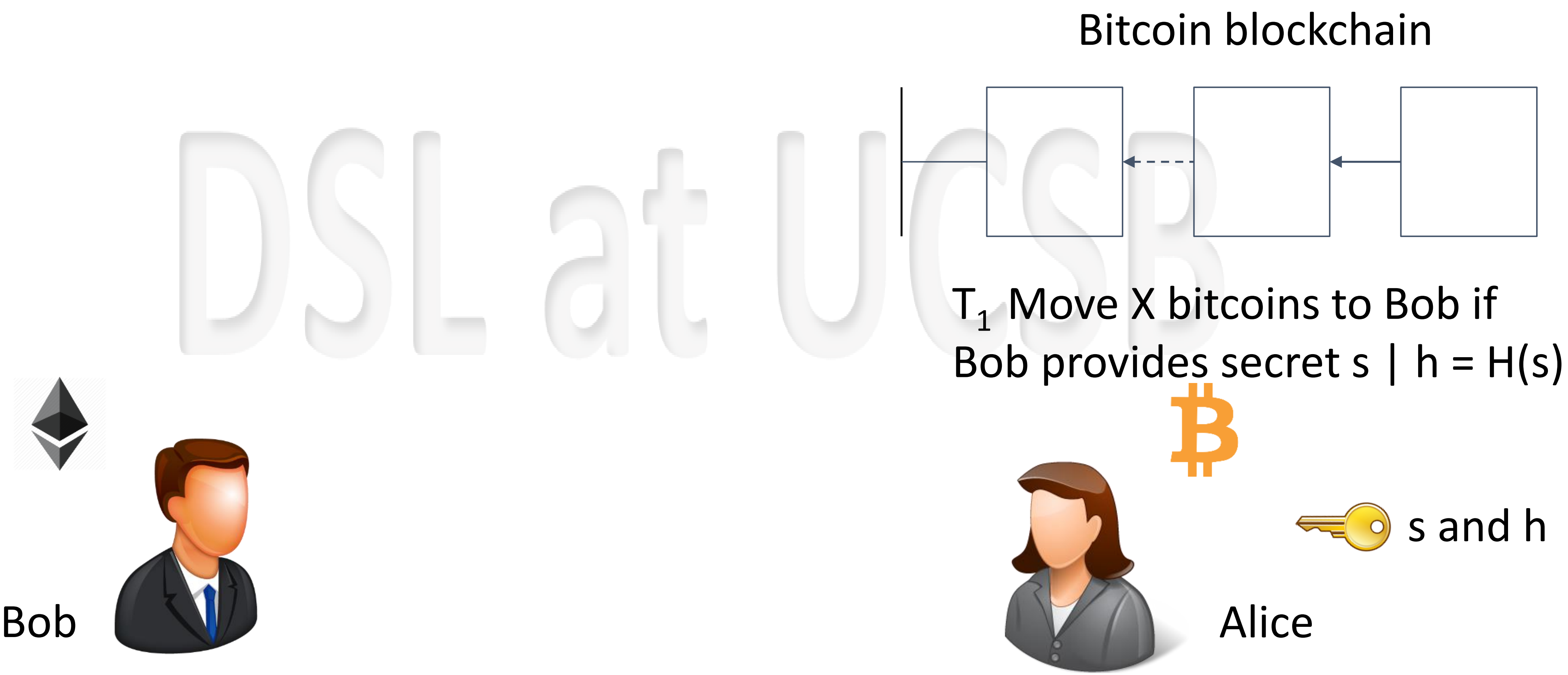
- Alice wants to trade X Bitcoin for Y Ethereum with Bob

DSL at UCSB



Atomic Swap Example

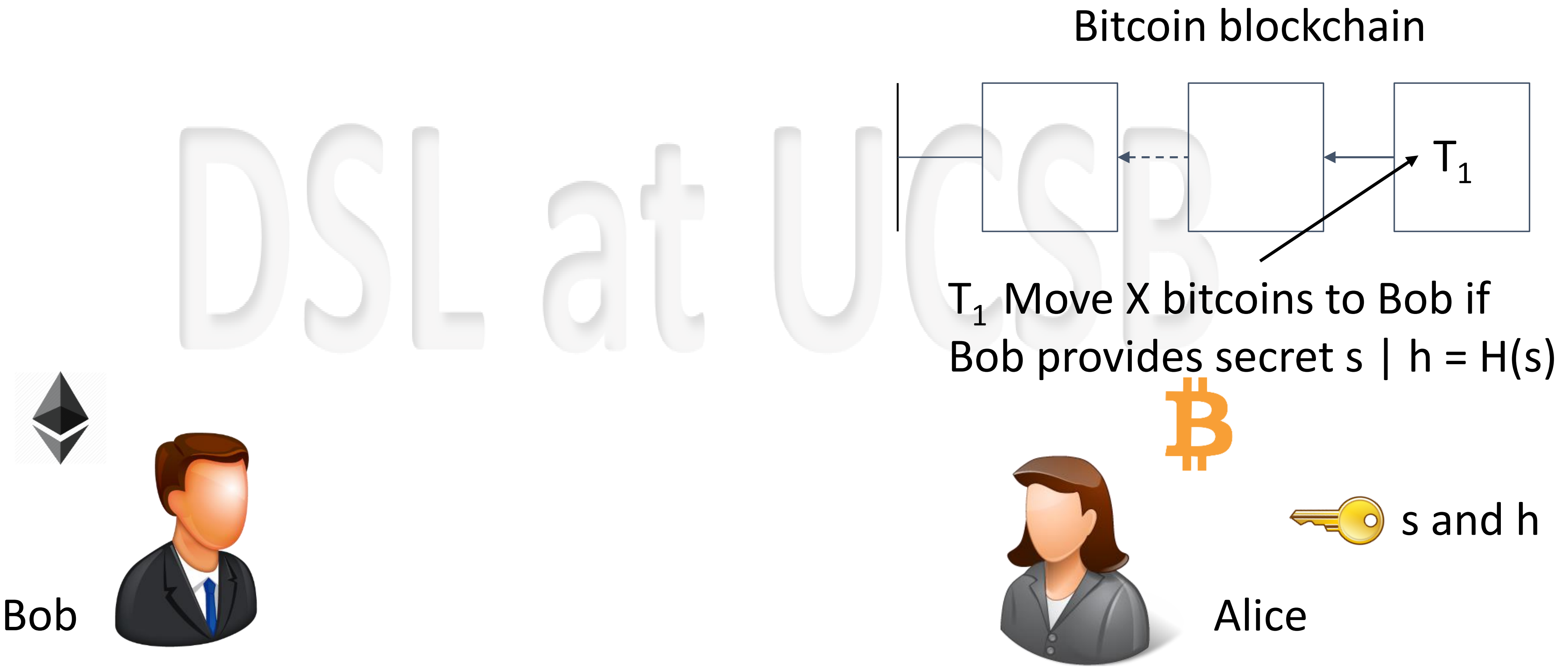
- Alice wants to trade X Bitcoin for Y Ethereum with Bob



DSL at UCSB

Atomic Swap Example

- Alice wants to trade X Bitcoin for Y Ethereum with Bob

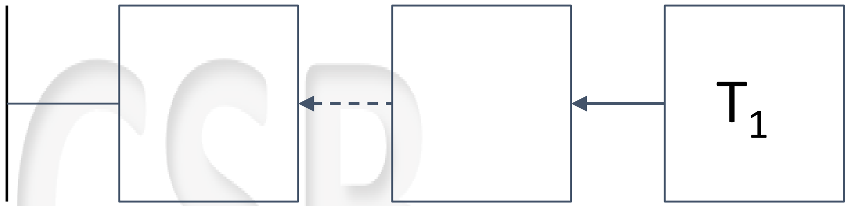


DSL at UCSB

Atomic Swap Example

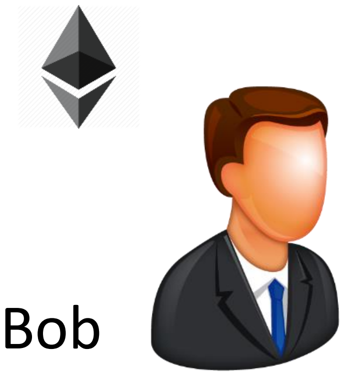
- Now, h is announced in Bitcoin blockchain and made public

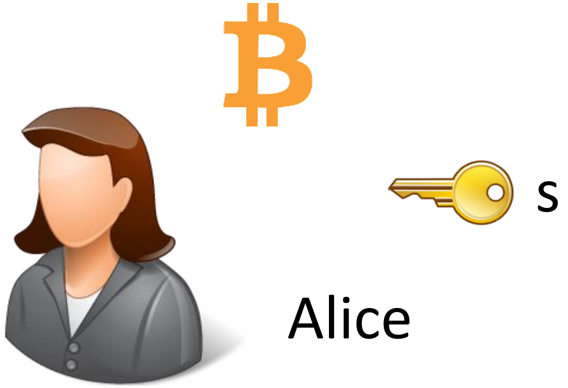
Bitcoin blockchain



DSL at UCSB

Alice's X bitcoins are locked in T_1's smart contract

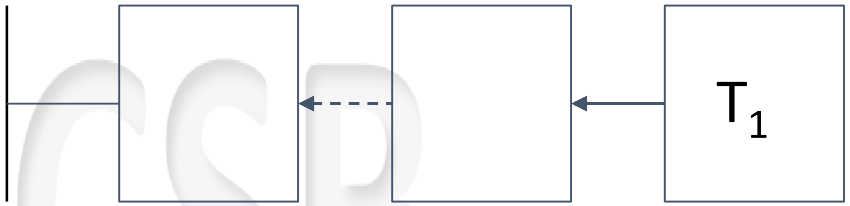




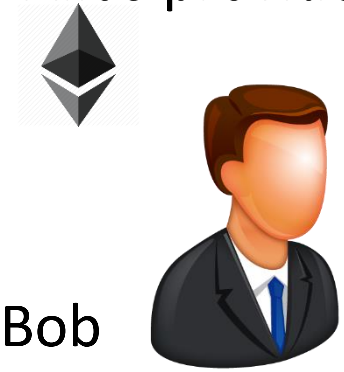
Atomic Swap Example

- Now, h is announced in Bitcoin blockchain and made public

Bitcoin blockchain

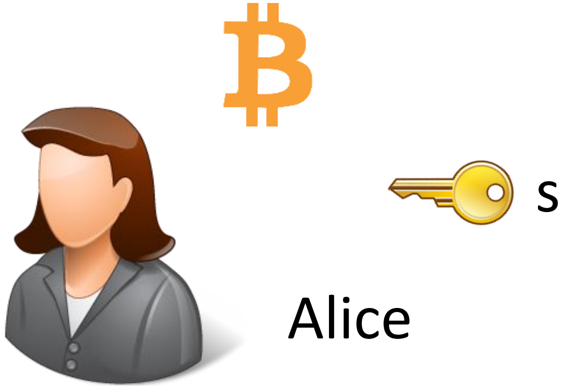


T_2 Move Y Ethereum to Alice if Alice provides secret s | h = H(s)



Bob

Alice's X bitcoins are locked in T_1's smart contract

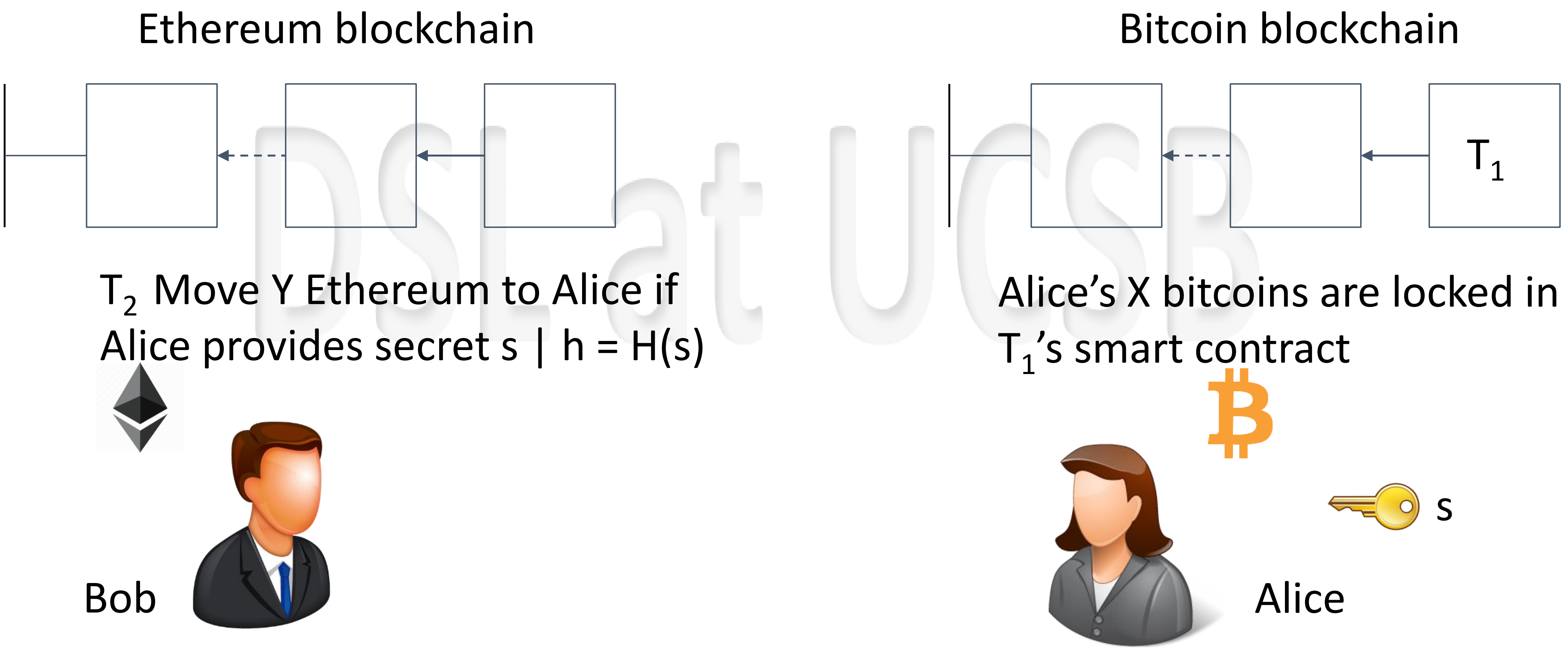


Alice

DSL at UCSB

Atomic Swap Example

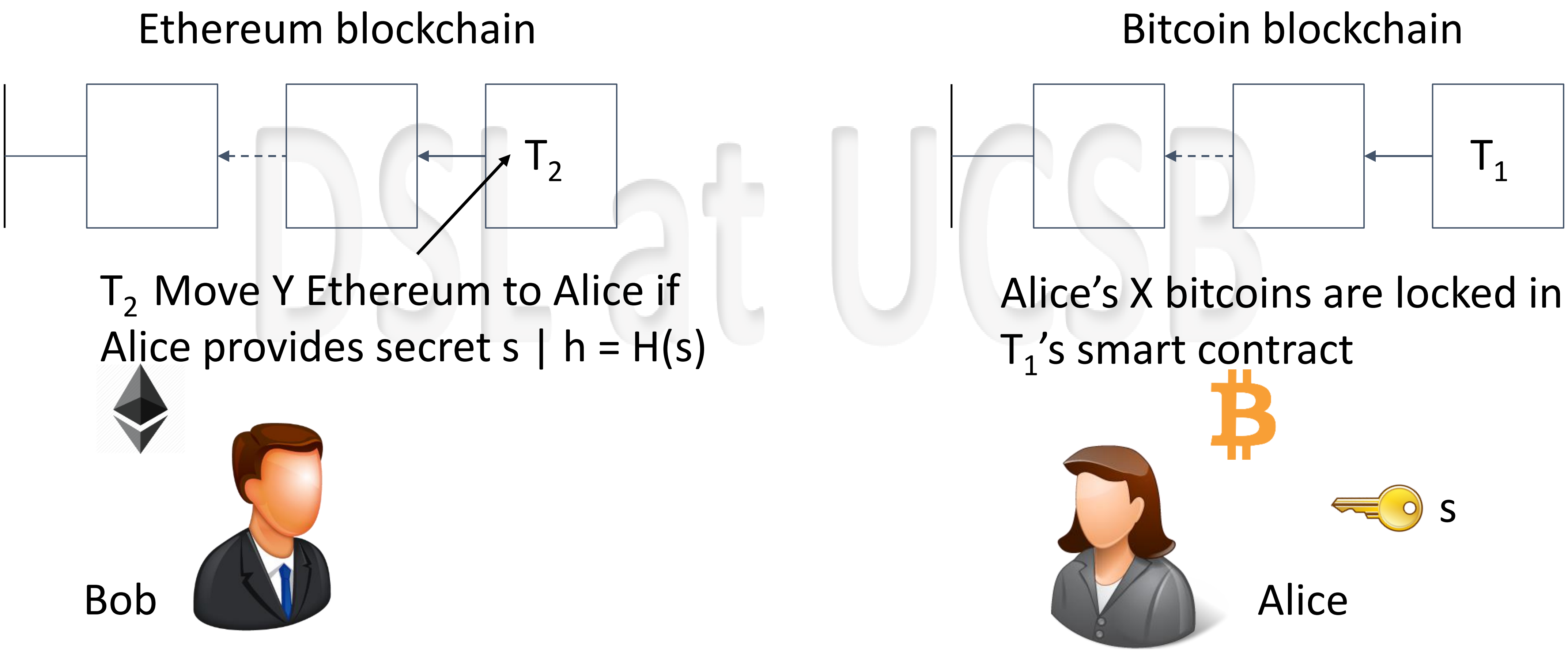
- Now, h is announced in Bitcoin blockchain and made public



DSL at UCSB

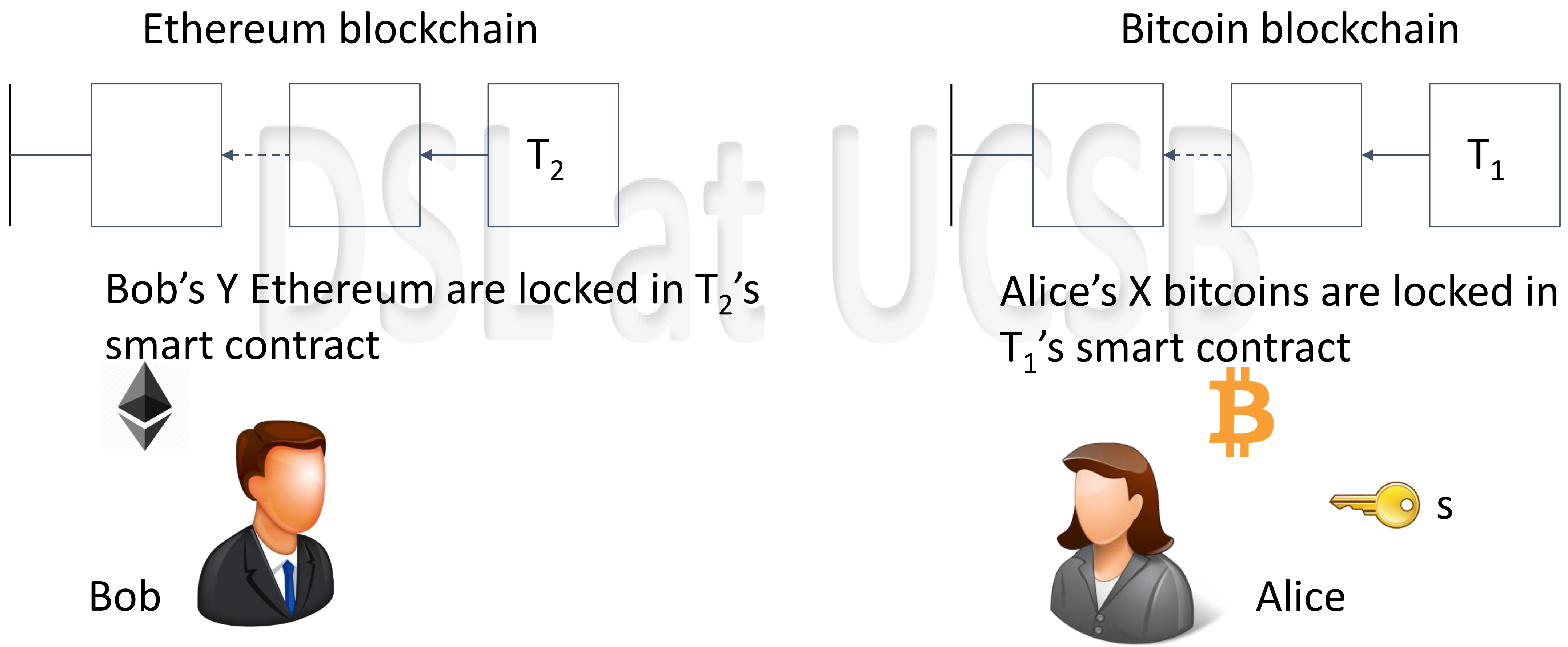
Atomic Swap Example

- Now, h is announced in Bitcoin blockchain and made public



Atomic Swap Example

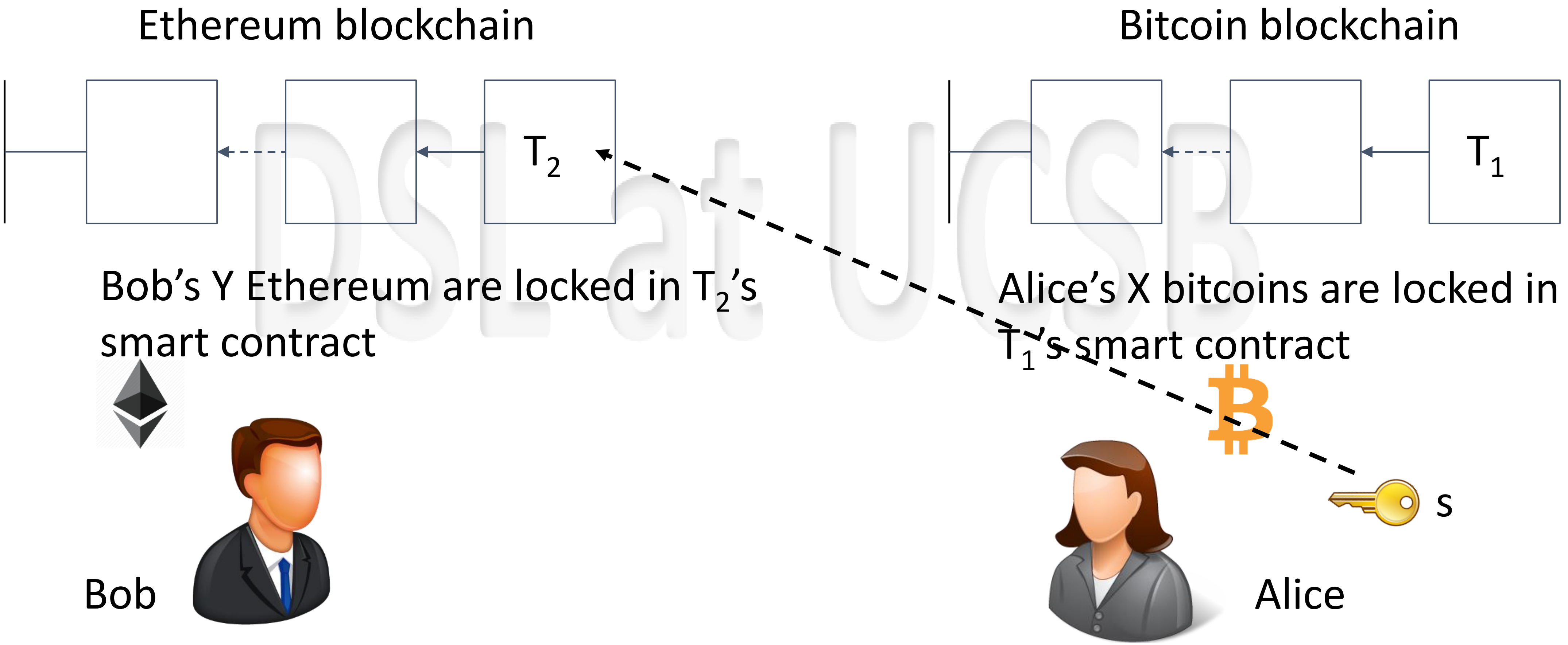
- Now, for Alice to execute T_2 and redeem Y Ethereum, she reveals s



DSL at UCSB

Atomic Swap Example

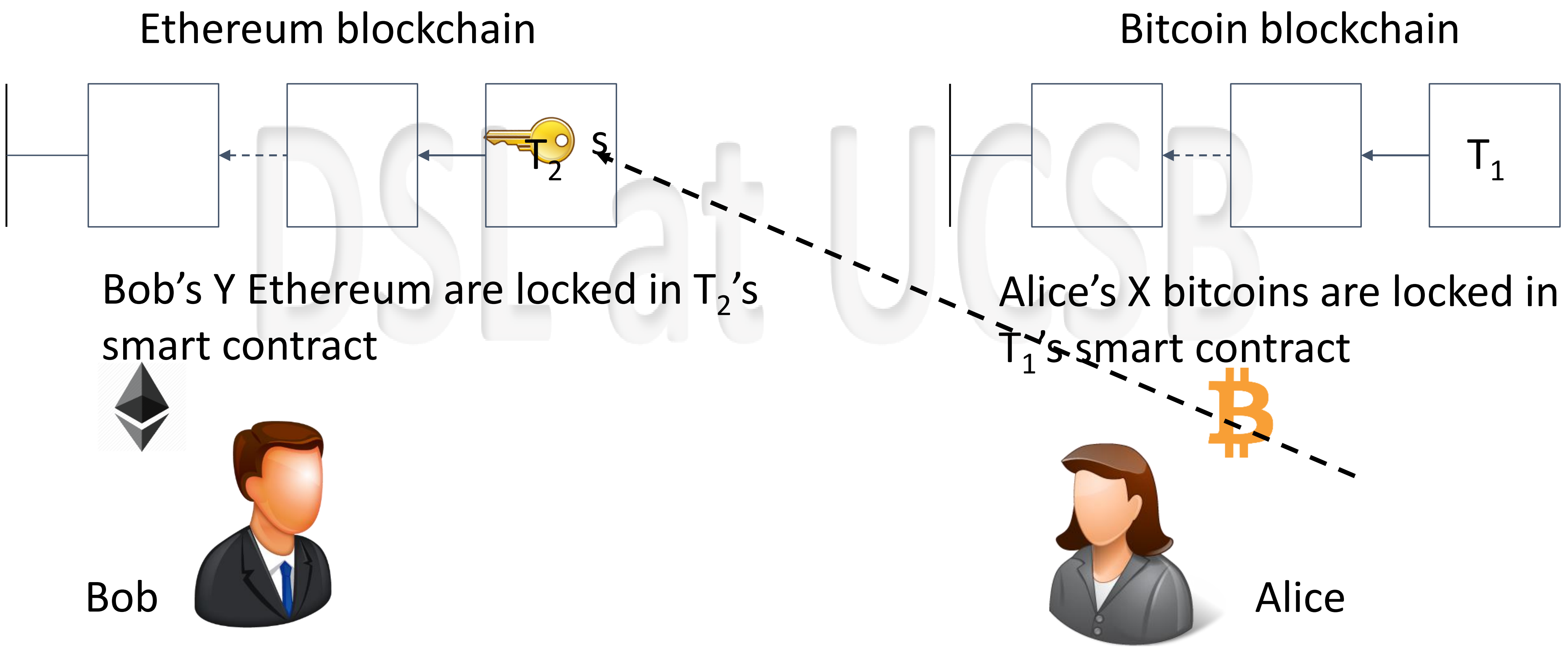
- Now, for Alice to execute T_2 and redeem Y Ethereum, she reveals s



DSL at UCSB

Atomic Swap Example

- Now, for Alice to execute T_2 and redeem Y Ethereum, she reveals s



DSL at UCSB

Atomic Swap Example

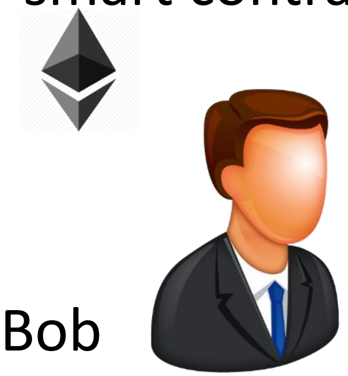
- Revealing s, executes T_2. Now s is public in Ethereum's blockchain

Ethereum blockchain

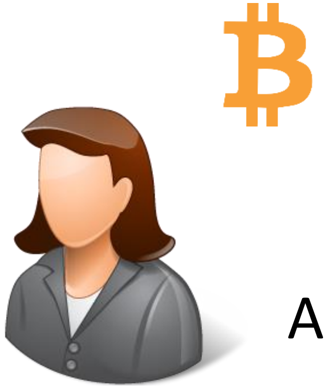
Bitcoin blockchain

Bob's Y Ethereum are locked in T_2's smart contract

Alice's X bitcoins are locked in T_1's smart contract



Bob



Alice

DSL at UCSB

Atomic Swap Example

- Now, Bob uses s to execute T_1 and redeem his Bitcoins

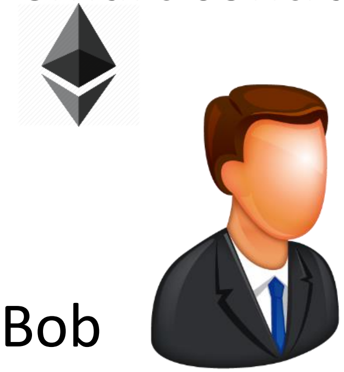
Ethereum blockchain

Bitcoin blockchain

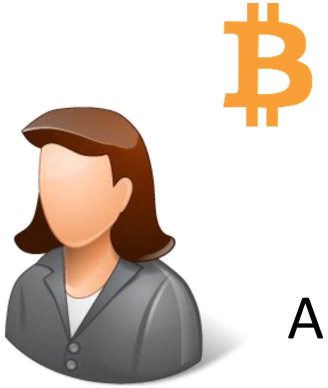
DSL at UCSB

Bob's Y Ethereum are locked in T_2's smart contract

Alice's X bitcoins are locked in T_1's smart contract



Bob

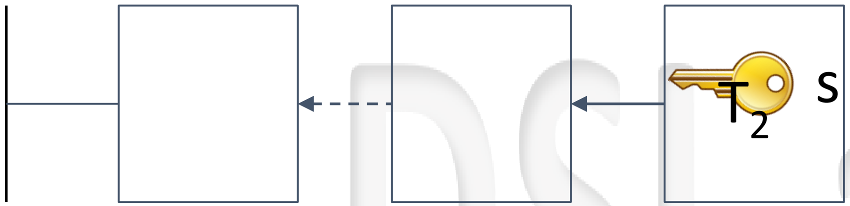


Alice

Atomic Swap Example

- Now, Bob uses s to execute T_1 and redeem his Bitcoins

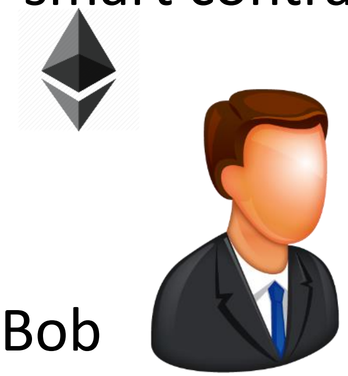
Ethereum blockchain



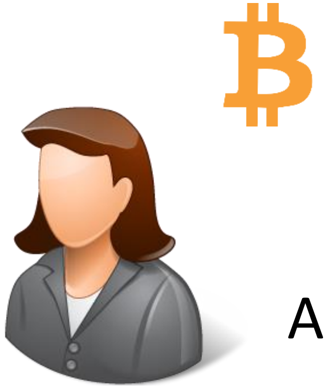
Bitcoin blockchain

Bob's Y Ethereum are locked in T_2's smart contract

Alice's X bitcoins are locked in T_1's smart contract



Bob



Alice

DSL at UCSB

Atomic Swap Example: What can go wrong?

- Alice locks her X Bitcoins in Bitcoin's blockchain through T_1

DSL at UCSB

Atomic Swap Example: What can go wrong?

- Alice locks her X Bitcoins in Bitcoin's blockchain through T_1
- Bob sees T_1 but refuses to insert T_2

DSL at UCSB

Atomic Swap Example: What can go wrong?

- Alice locks her X Bitcoins in Bitcoin's blockchain through T_1
- Bob sees T_1 but refuses to insert T_2
- Now, Alice's Bitcoins are locked for good
  - A conforming party (Alice) ends up worse off because Bob doesn't follow the protocol

DSL at UCSB

Atomic Swap Example: What can go wrong?

- Alice locks her X Bitcoins in Bitcoin's blockchain through T_1
- Bob sees T_1 but refuses to insert T_2
- Now, Alice's Bitcoins are locked for good
  - A conforming party (Alice) ends up worse off because Bob doesn't follow the protocol
- Prevention
  - Use timelocks to expire a contract
  - Specify that an expired contract is refunded to the creator of this contract

DSL at UCSB

Atomic Swap Example: Timelocks

DSL at UCSB

Atomic Swap Example: Timelocks

DSL at UCSB

T_3: Refund T_1 to Alice if Bob does not execute T_1 before **48** hours

T_1: Move X bitcoins to Bob if Bob provides secret s | h = H(s)

Bob

Alice

Atomic Swap Example: Timelocks

DSL at UCSB

T_4: Refund T_2 to Bob if Alice does not execute T_2 before **24** hours

T_2: Move Y Ethereum to Alice if Alice provides secret s | h = H(s)

Bob

T_3: Refund T_1 to Alice if Bob does not execute T_1 before **48** hours

T_1: Move X bitcoins to Bob if Bob provides secret s | h = H(s)

Alice

Atomic Swap Example: Timelocks

DSL at UCSB

T_4: Refund T_2 to Bob if Alice does not execute T_2 before **24** hours

T_2: Move Y Ethereum to Alice if Alice provides secret s | h = H(s)

Bob

T_3: Refund T_1 to Alice if Bob does not execute T_1 before **48** hours

T_1: Move X bitcoins to Bob if Bob provides secret s | h = H(s)

Alice

Atomic Swap Example: Timelocks

How to determine the time period of a timelock?

T_4: Refund T_2 to Bob if Alice does not execute T_2 before **24** hours

T_2: Move Y Ethereum to Alice if Alice provides secret s | h = H(s)

T_3: Refund T_1 to Alice if Bob does not execute T_1 before **48** hours

T_1: Move X bitcoins to Bob if Bob provides secret s | h = H(s)

Bob

Alice

DSL at UCSB

Timelocks

- Timelocks are set to prevent any conforming party to end up worse off
- If Alice sets her timelock to 12 hours and Bob to 24 hours
  - Alice can wait until her contract expires (gets a refund)
  - Then, Alice executes T_2 claiming T_2's Ethereum coins

T_4: Refund T_2 to Bob if Alice does not execute T_2 before **24** hours

T_2: Move Y Ethereum to Alice if Alice provides secret s | h = H(s)

Bob

T_3: Refund T_1 to Alice if Bob does not execute T_1 before **12** hours

T_1: Move X bitcoins to Bob if Bob provides secret s | h = H(s)

Alice

DSL at UCSB

Timelocks

- Bob's timelock should be set to achieve the following:
  - Forces Alice to reveal **s** before Alice's contract expires
  - Allows enough time for Bob to execute T_1 after Alice executes T_2
  - If Alice does not reveal **s**, both contracts should expire and be refunded

DSL at UCSB

T_4: Refund T_2 to Bob if Alice does not execute T_2 before **24** hours

T_2: Move Y Ethereum to Alice if Alice provides secret s | h = H(s)

Bob

T_3: Refund T_1 to Alice if Bob does not execute T_1 before **12** hours

T_1: Move X bitcoins to Bob if Bob provides secret s | h = H(s)

Alice

Atomic Swap Modeling

- A cross-chain swap is modeled as a directed graph D = (V,A)

DSL at UCSB

Atomic Swap Modeling

- A cross-chain swap is modeled as a directed graph D = (V,A)
- Vertices V are parties and arcs A are proposed asset transfers

DSL at UCSB

Atomic Swap Modeling

- A cross-chain swap is modeled as a directed graph D = (V,A)
- Vertices V are parties and arcs A are proposed asset transfers
- Assumptions:
  - Every party is rational
    - E.g., Bob sets his timelock to 6 hours instead of 24 hours

DSL at UCSB

Atomic Swap Modeling

- A cross-chain swap is modeled as a directed graph D = (V,A)
- Vertices V are parties and arcs A are proposed asset transfers
- Assumptions:
  - Every party is rational
    - E.g., Bob sets his timelock to 6 hours instead of 24 hours
  - The directed graph must be strongly connected
    - There is a path between any two pairs of nodes

DSL at UCSB

DSL

UCSB

Atomic Swap Modeling

- A cross-chain swap is modeled as a directed graph D = (V,A)
- Vertices V are parties and arcs A are proposed asset transfers
- Assumptions:
  - Every party is rational
    - E.g., Bob sets his timelock to 6 hours instead of 24 hours
  - The directed graph must be strongly connected
    - There is a path between any two pairs of nodes
  - There is known time bound **Δ**
    - **Δ** should be enough for one party to publish a contract to a blockchain and for a second party to confirm that the contract has been published

DSL at UCSB

Multi-party Atomic Swap Example

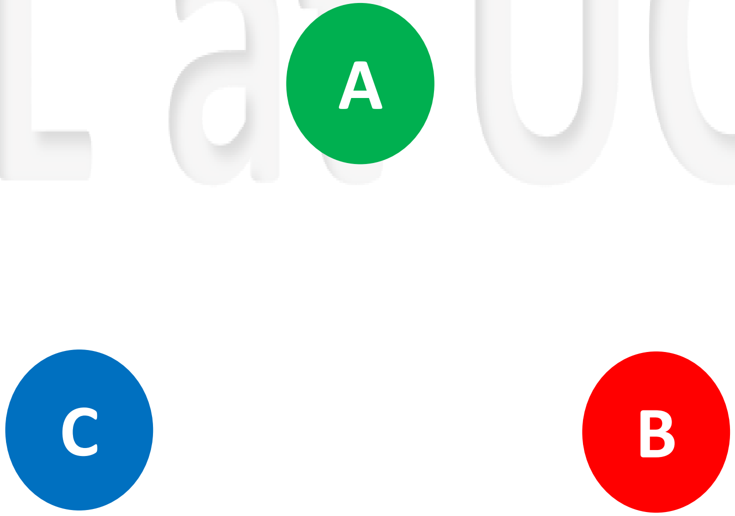
- Alice wants to buy Carol's car with Bitcoins
- Carol wants to sell her car for Ethereum
- Luckily, Bob wants to exchange Ethereum for Bitcoin

DSL at UCSB

Multi-party Atomic Swap Example

- Alice wants to buy Carol's car with Bitcoins
- Carol wants to sell her car for Ethereum
- Luckily, Bob wants to exchange Ethereum for Bitcoin

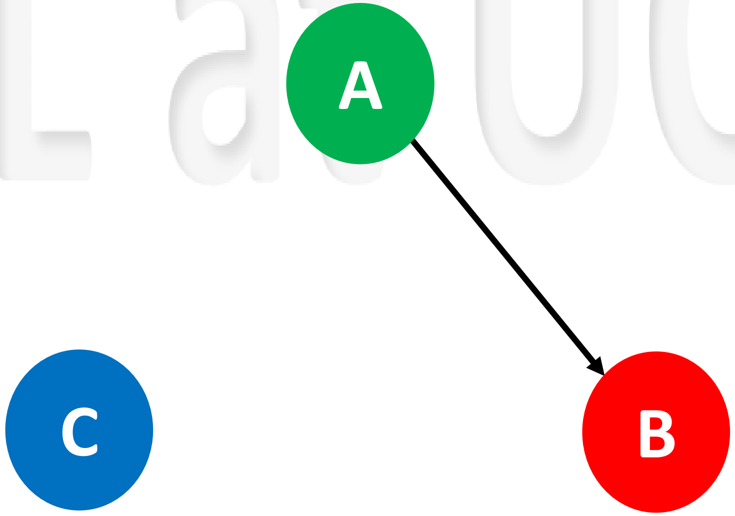
DSL at UCSB



Multi-party Atomic Swap Example

- Alice wants to buy Carol’s car with Bitcoins
- Carol wants to sell her car for Ethereum
- Luckily, Bob wants to exchange Ethereum for Bitcoin

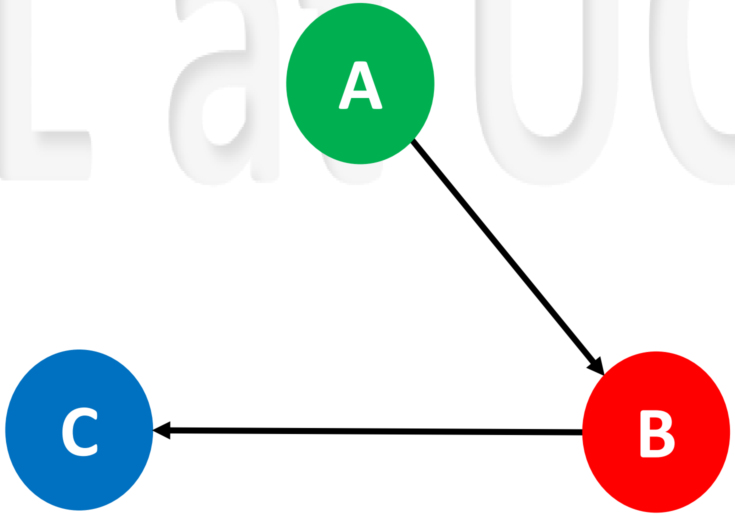
DSL at UCSB



Multi-party Atomic Swap Example

- Alice wants to buy Carol's car with Bitcoins
- Carol wants to sell her car for Ethereum
- Luckily, Bob wants to exchange Ethereum for Bitcoin

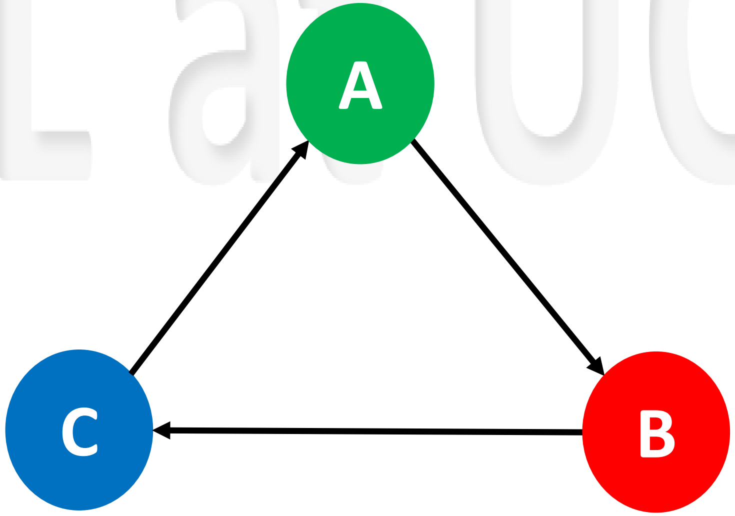
DSL at UCSB



Multi-party Atomic Swap Example

- Alice wants to buy Carol's car with Bitcoins
- Carol wants to sell her car for Ethereum
- Luckily, Bob wants to exchange Ethereum for Bitcoin

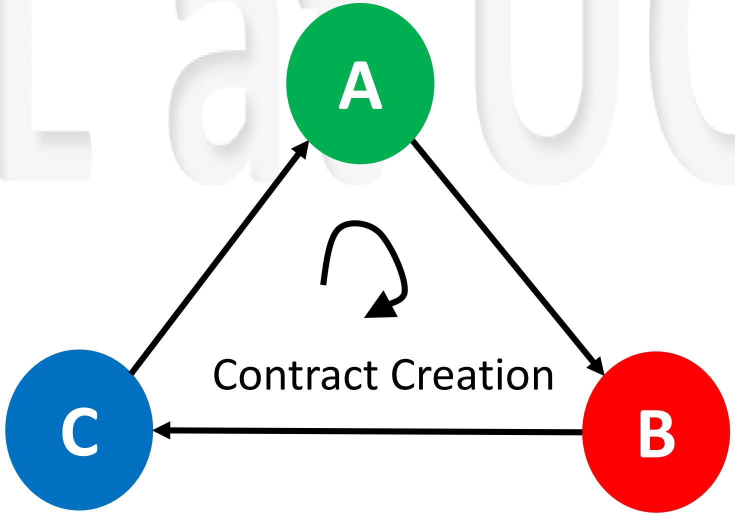
DSL at UCSB



Multi-party Atomic Swap Example

- Alice wants to buy Carol's car with Bitcoins
- Carol wants to sell her car for Ethereum
- Luckily, Bob wants to exchange Ethereum for Bitcoin

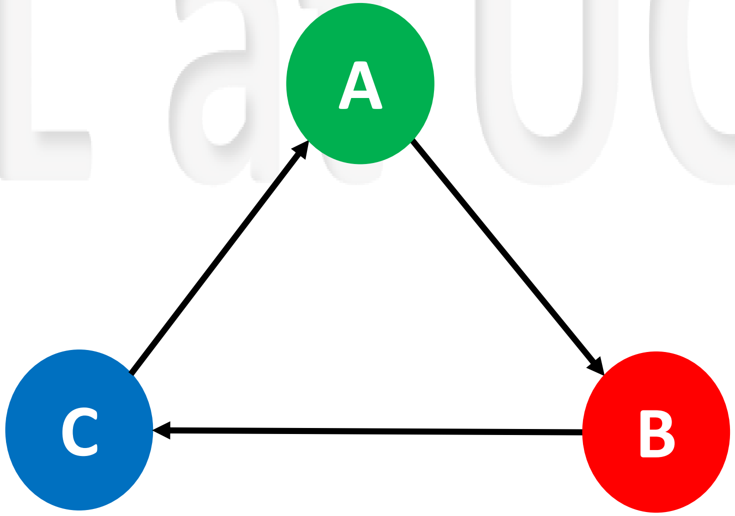
DSL at UCSB



Multi-party Atomic Swap Example

- Alice wants to buy Carol's car with Bitcoins
- Carol wants to sell her car for Ethereum
- Luckily, Bob wants to exchange Ethereum for Bitcoin

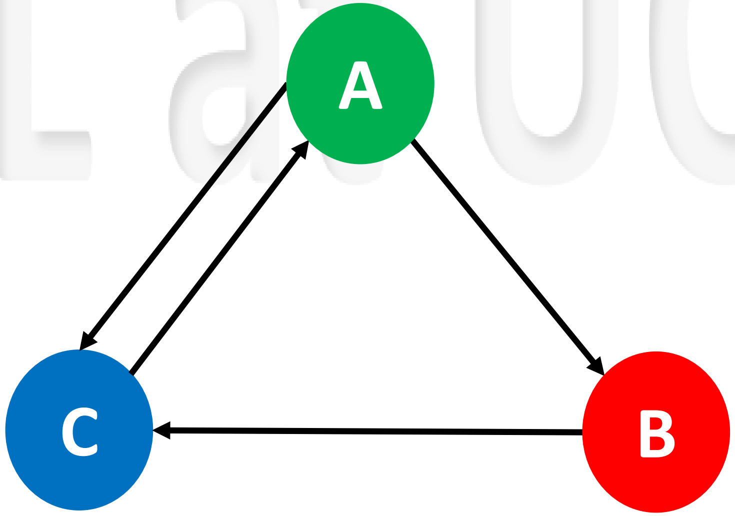
DSL at UCSB



Multi-party Atomic Swap Example

- Alice wants to buy Carol's car with Bitcoins
- Carol wants to sell her car for Ethereum
- Luckily, Bob wants to exchange Ethereum for Bitcoin

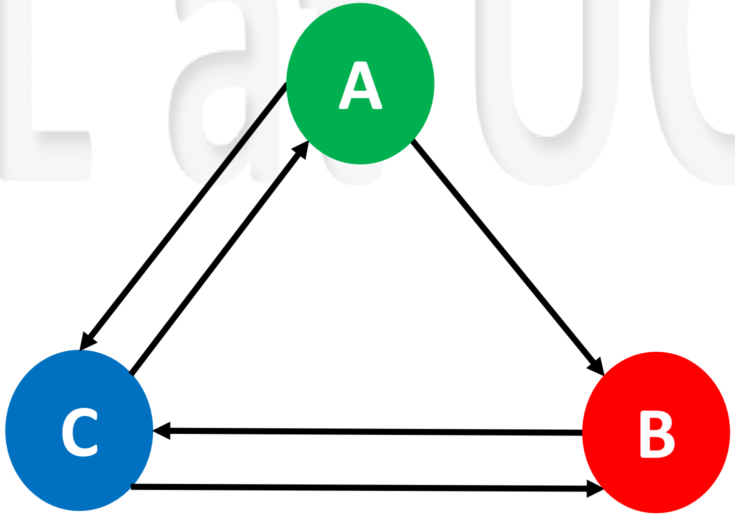
DSL at UCSB



Multi-party Atomic Swap Example

- Alice wants to buy Carol's car with Bitcoins
- Carol wants to sell her car for Ethereum
- Luckily, Bob wants to exchange Ethereum for Bitcoin

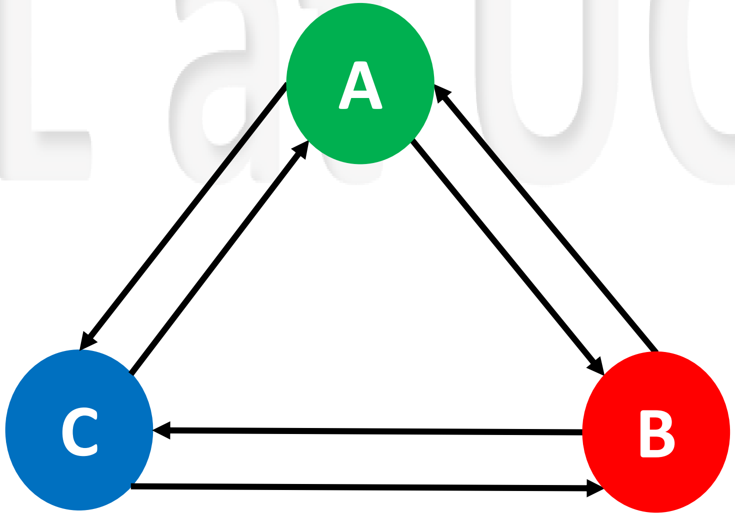
DSL at UCSB



Multi-party Atomic Swap Example

- Alice wants to buy Carol's car with Bitcoins
- Carol wants to sell her car for Ethereum
- Luckily, Bob wants to exchange Ethereum for Bitcoin

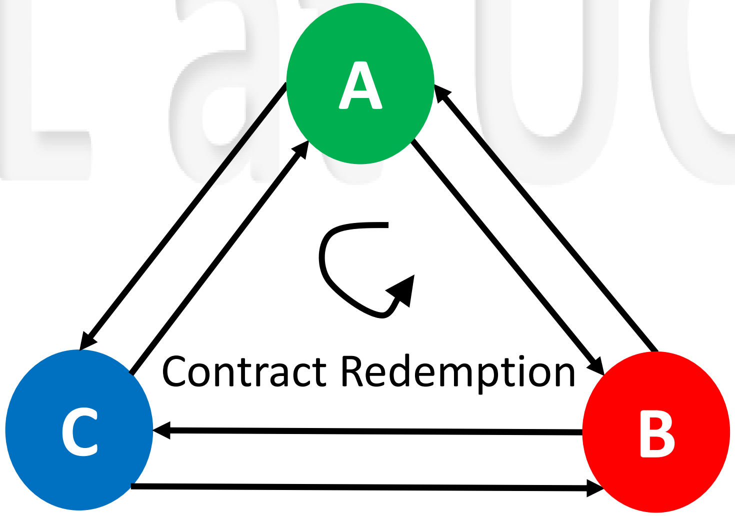
DSL at UCSB



Multi-party Atomic Swap Example

- Alice wants to buy Carol's car with Bitcoins
- Carol wants to sell her car for Ethereum
- Luckily, Bob wants to exchange Ethereum for Bitcoin

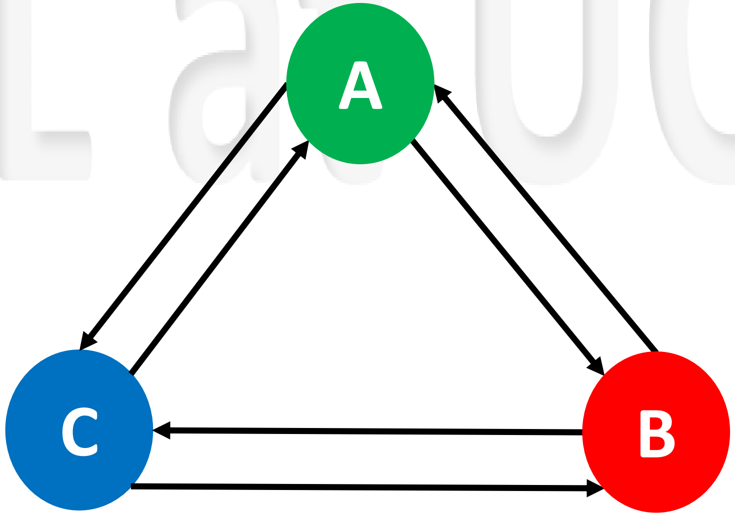
DSL at UCSB



Multi-party Atomic Swap Example

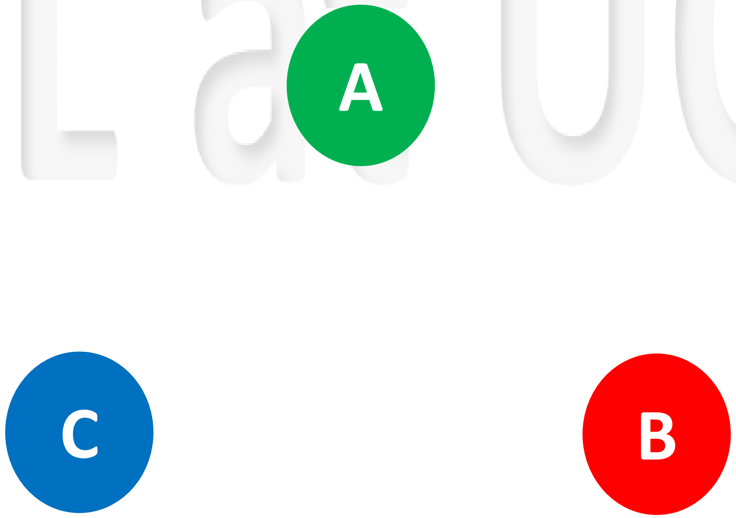
- Alice wants to buy Carol's car with Bitcoins
- Carol wants to sell her car for Ethereum
- Luckily, Bob wants to exchange Ethereum for Bitcoin

DSL at UCSB

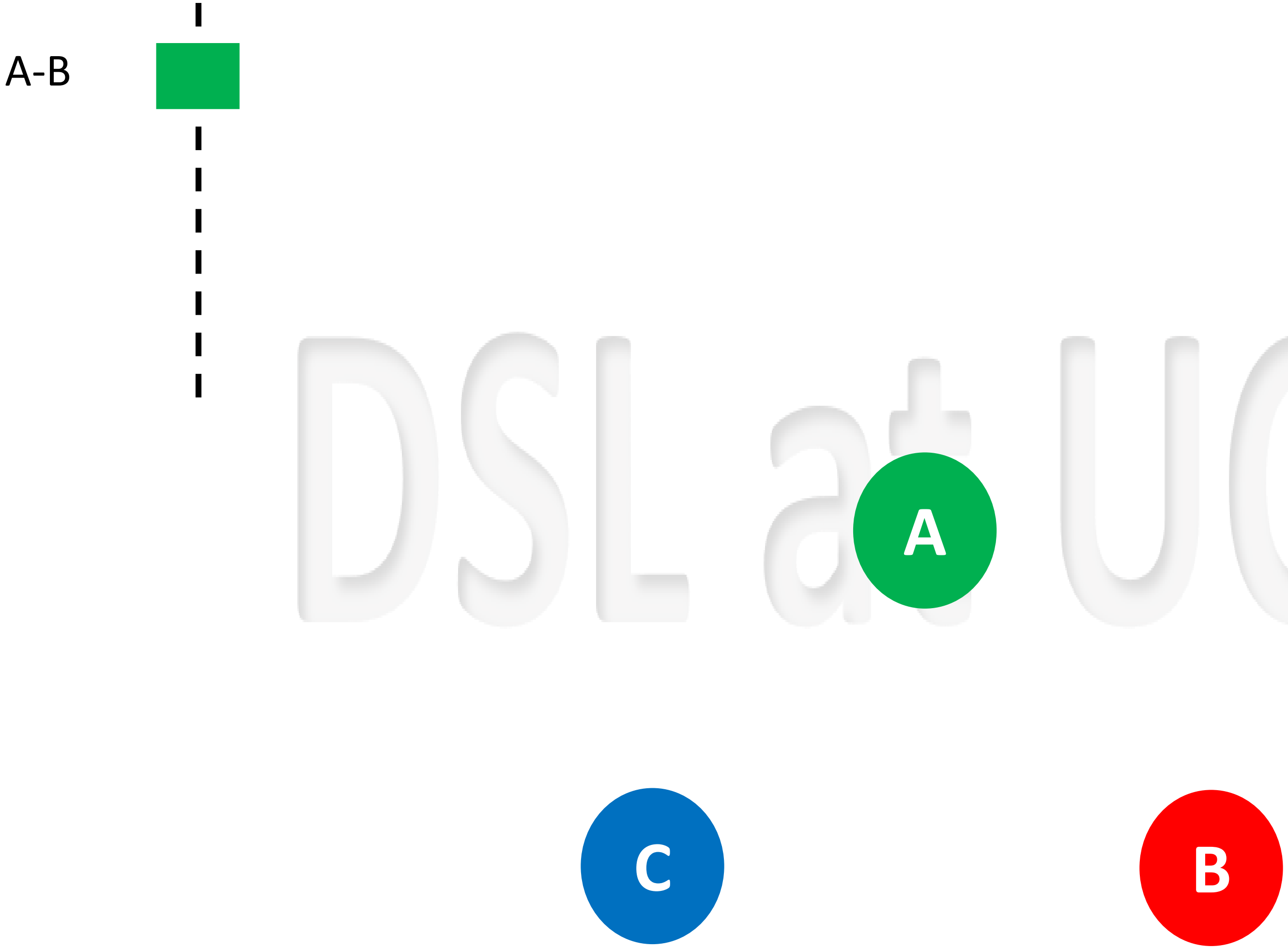


Multi-party Atomic Swap Example

DSL at UCSB

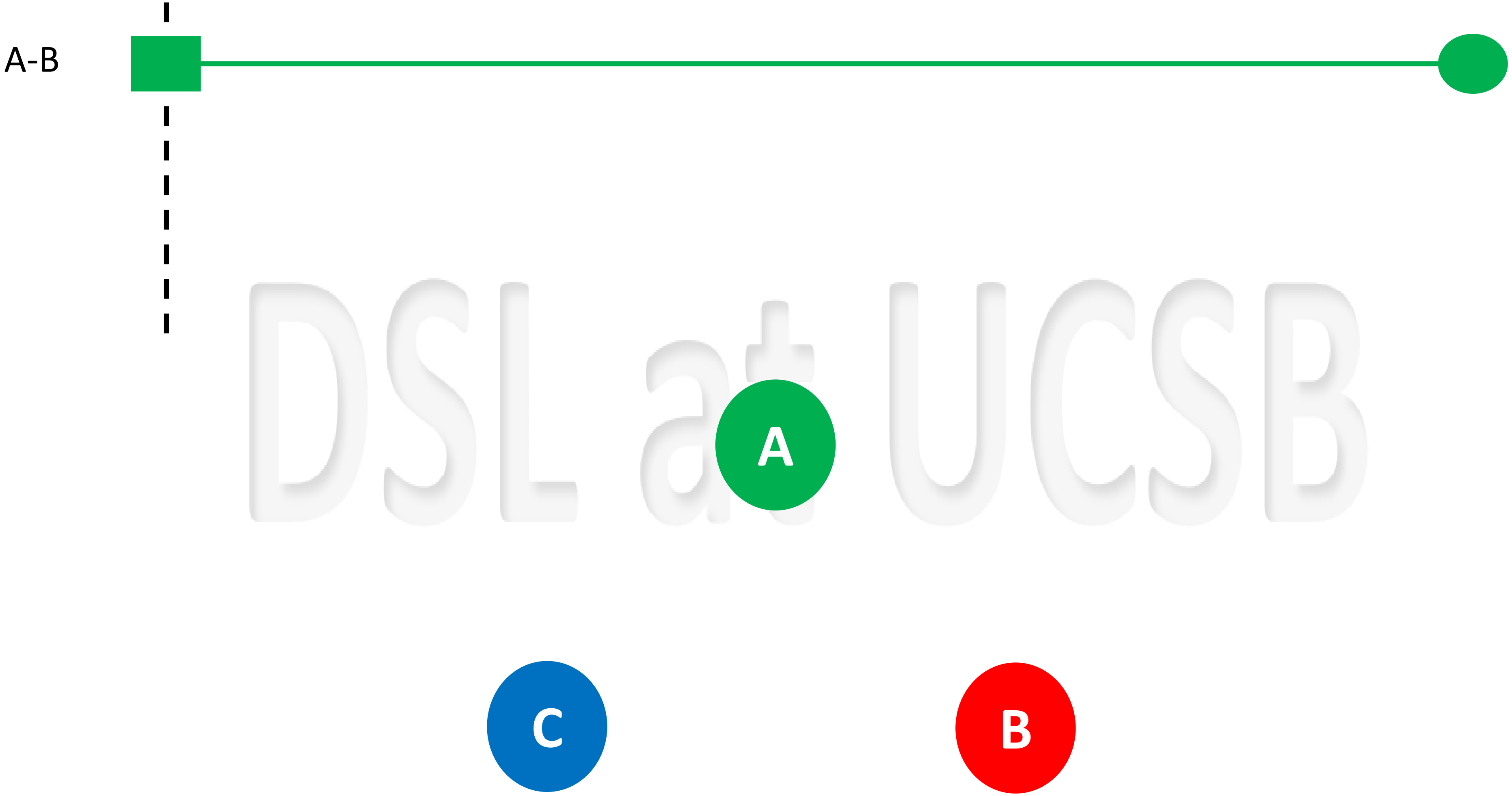


Multi-party Atomic Swap Example



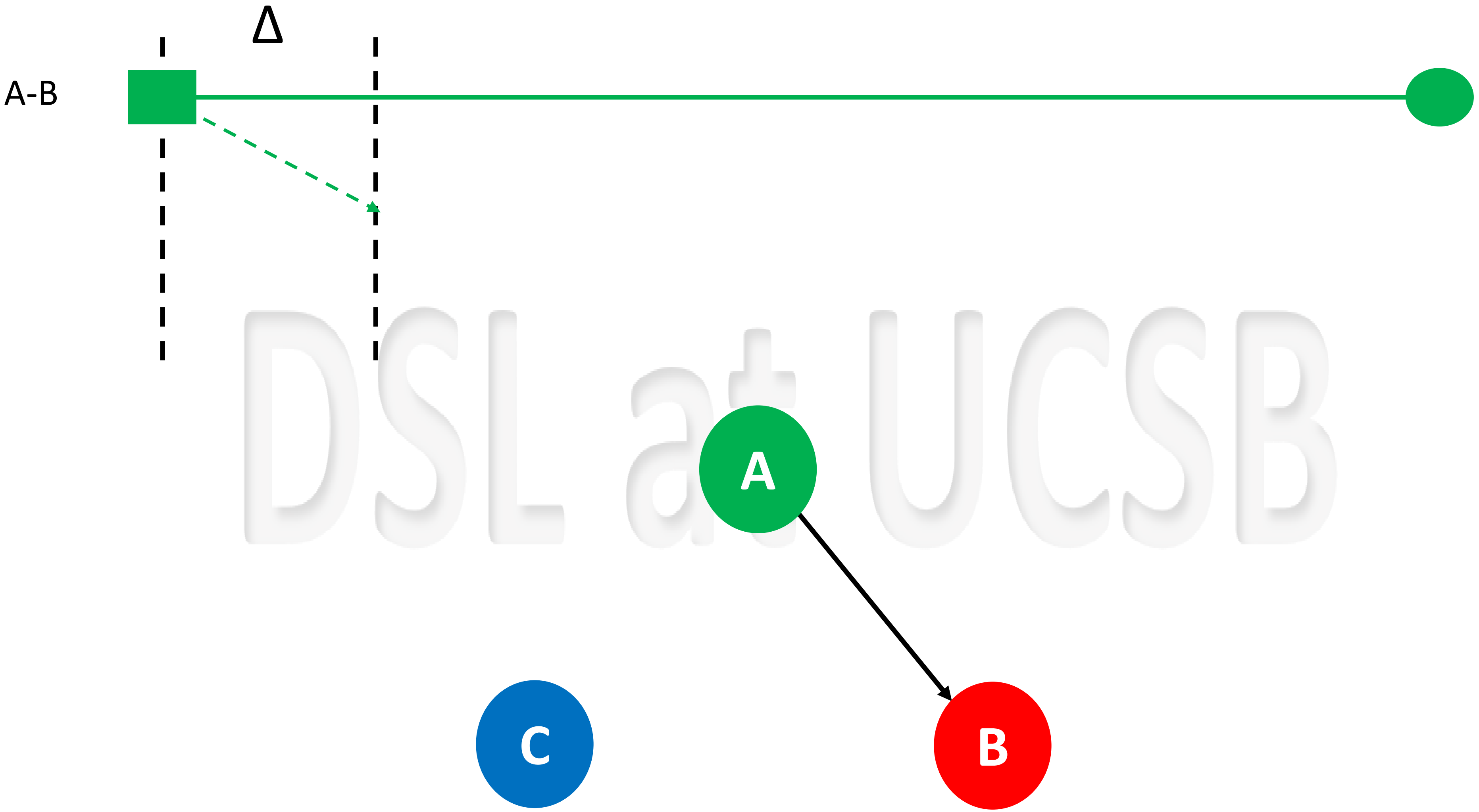
DSL at UCSB

Multi-party Atomic Swap Example



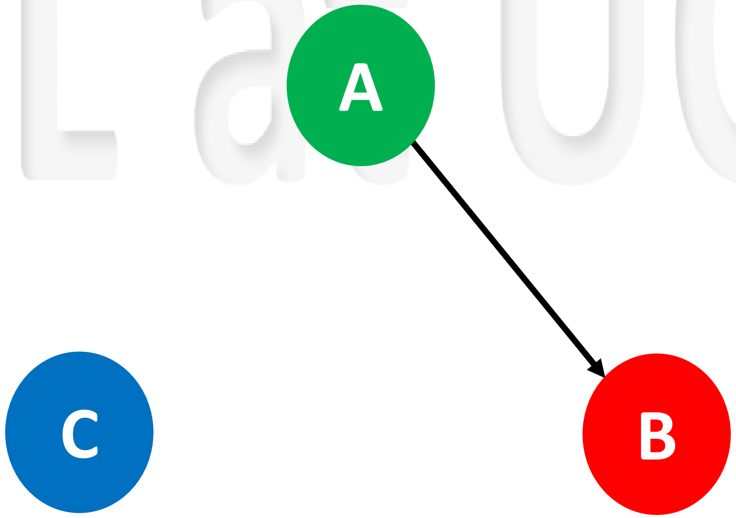
DSL at UCSB

Multi-party Atomic Swap Example



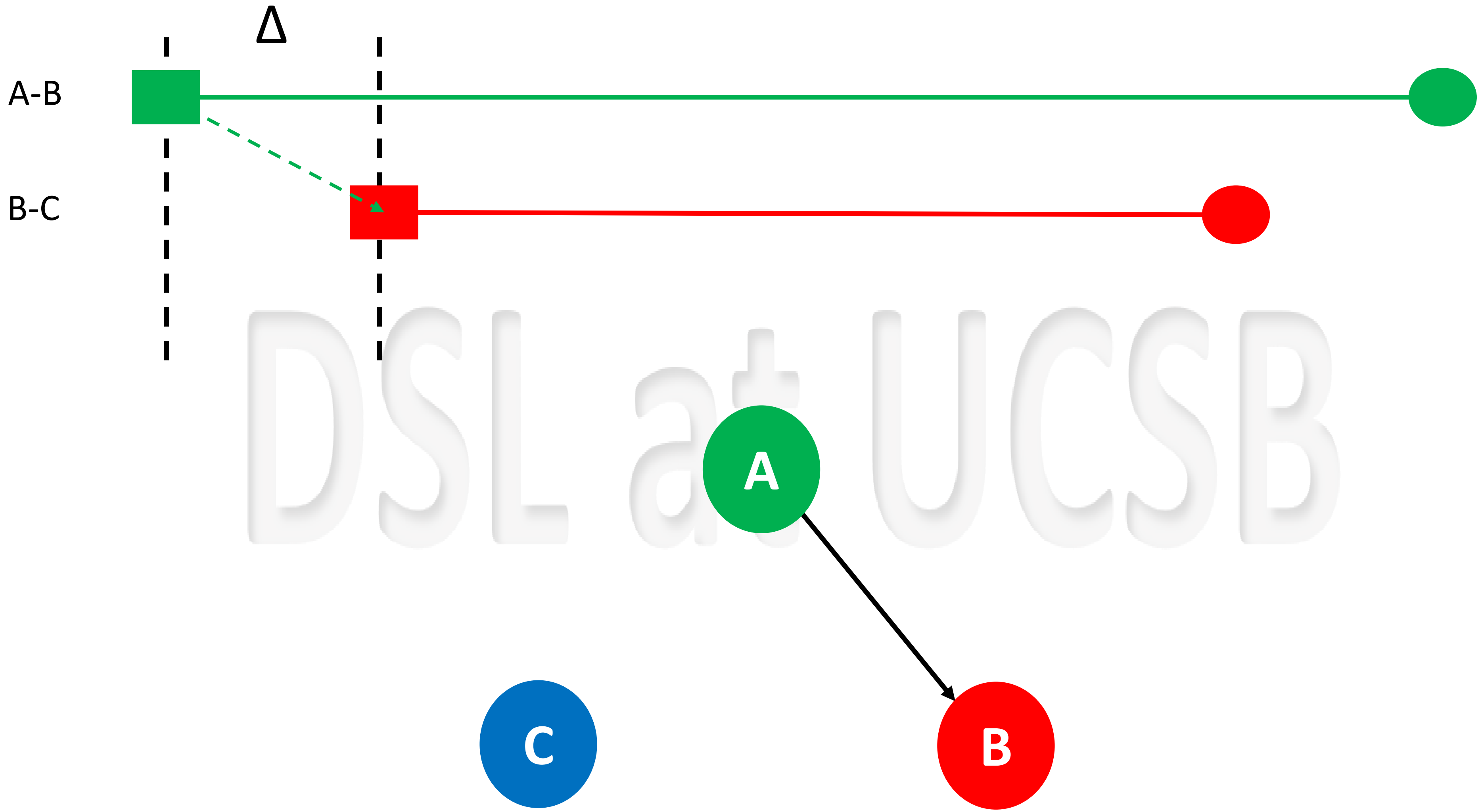
DSL at UCSB

Multi-party Atomic Swap Example



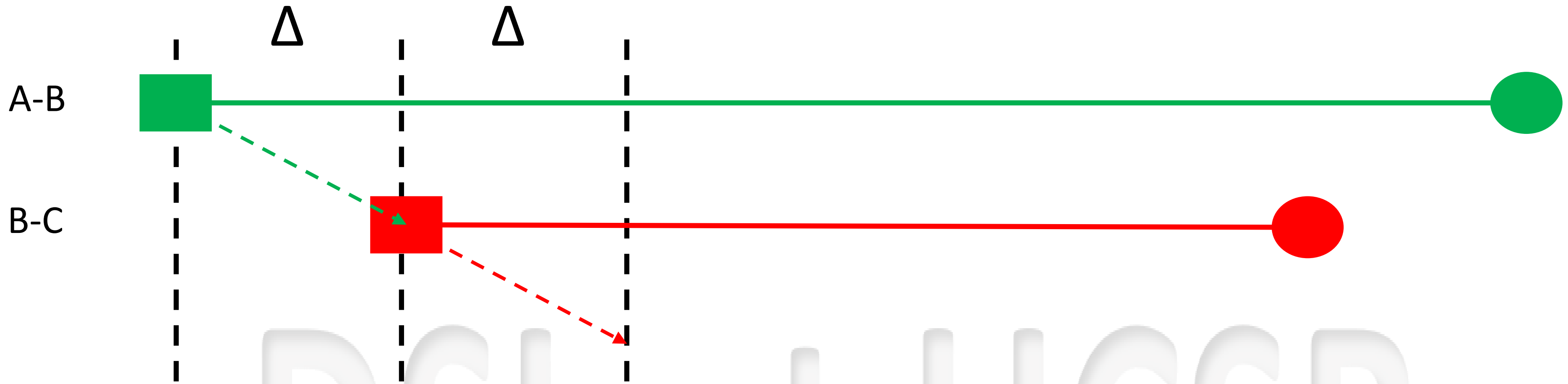
DSL at UCSB

Multi-party Atomic Swap Example

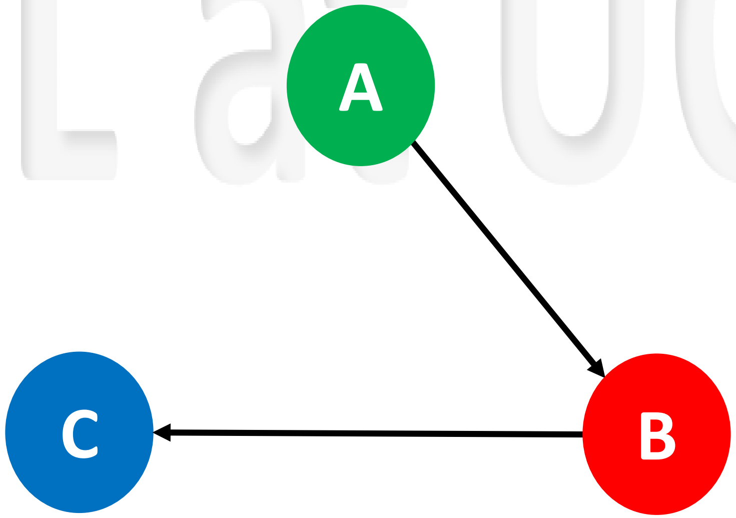


DSL at UCSB

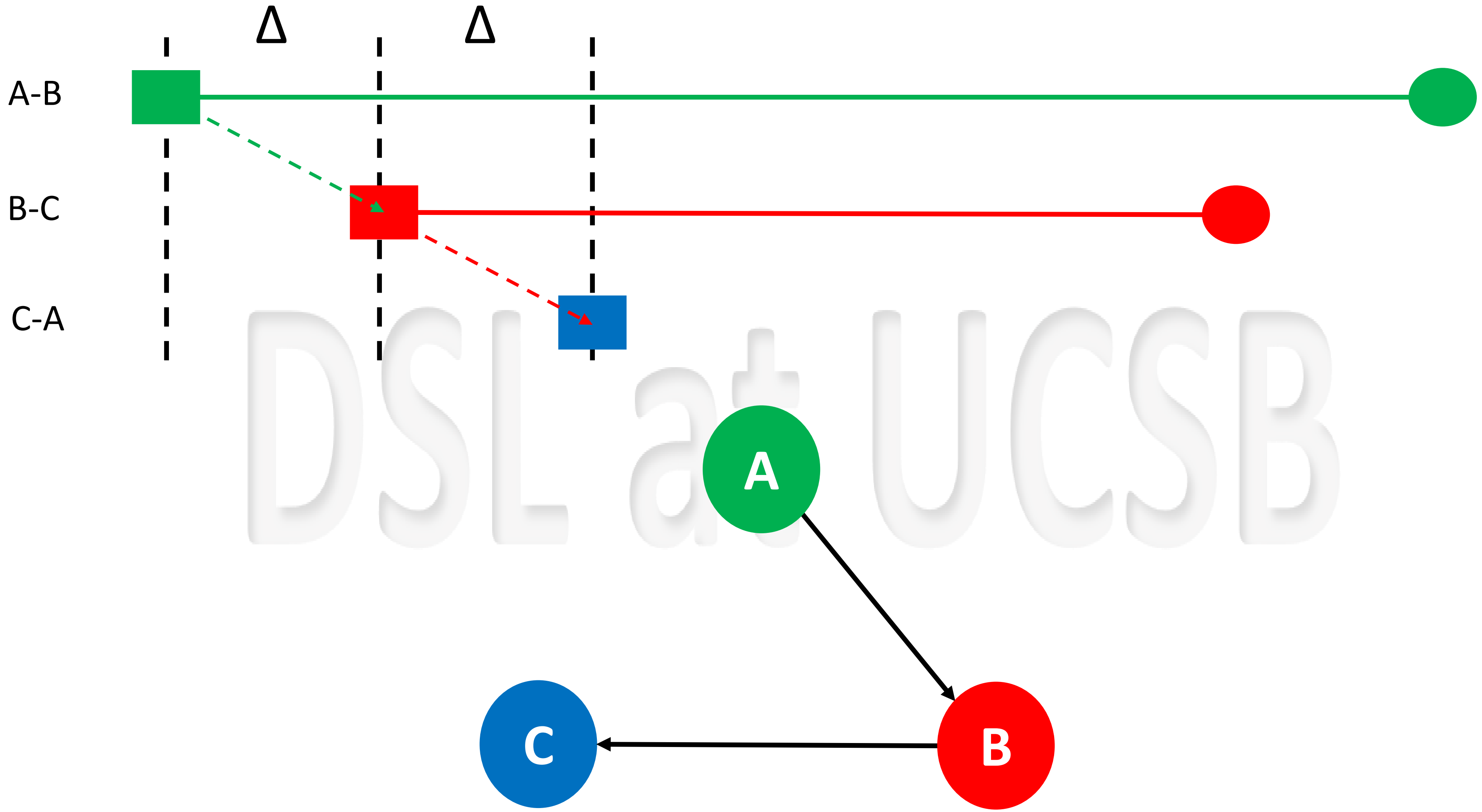
Multi-party Atomic Swap Example



DSL at UCSB

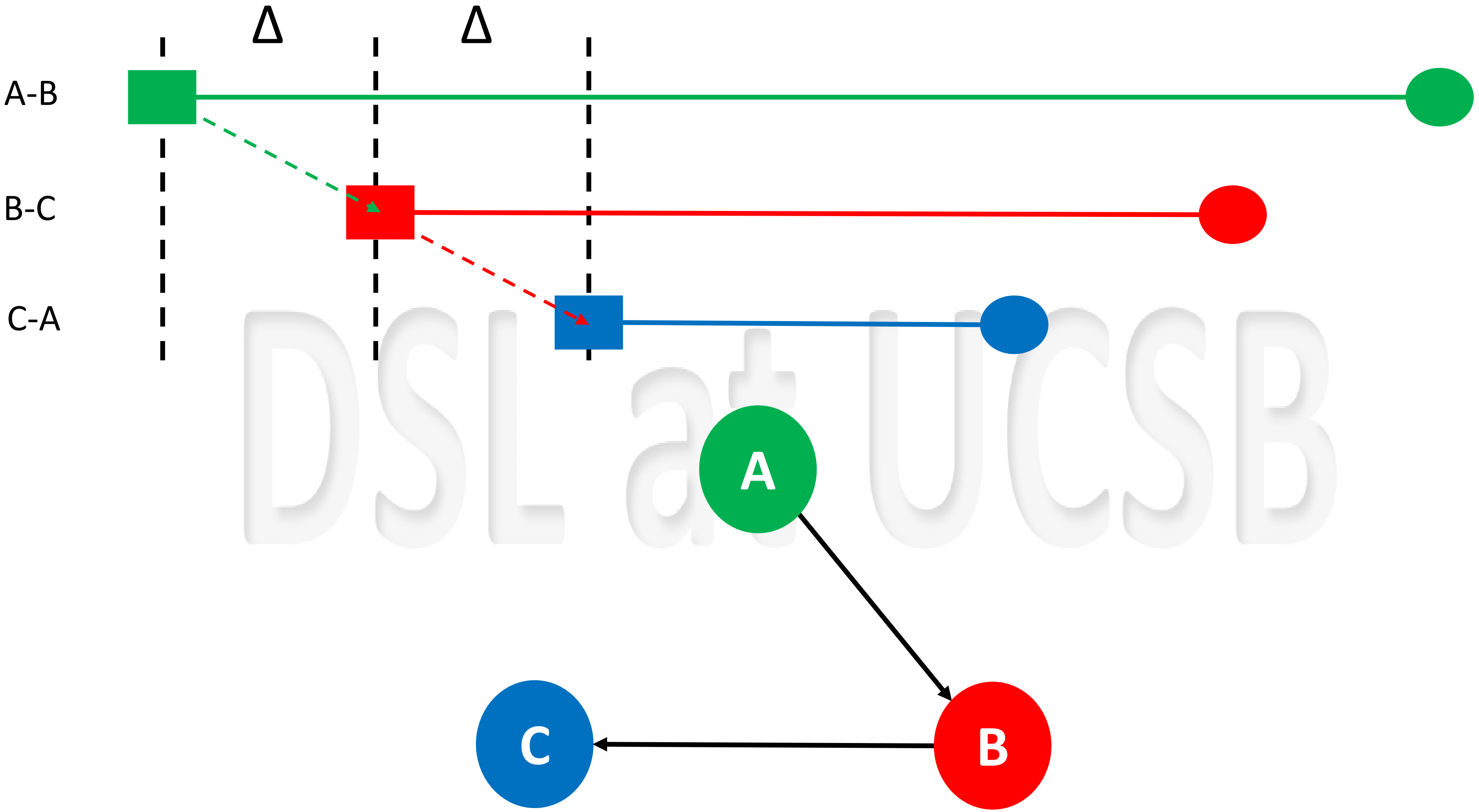


Multi-party Atomic Swap Example



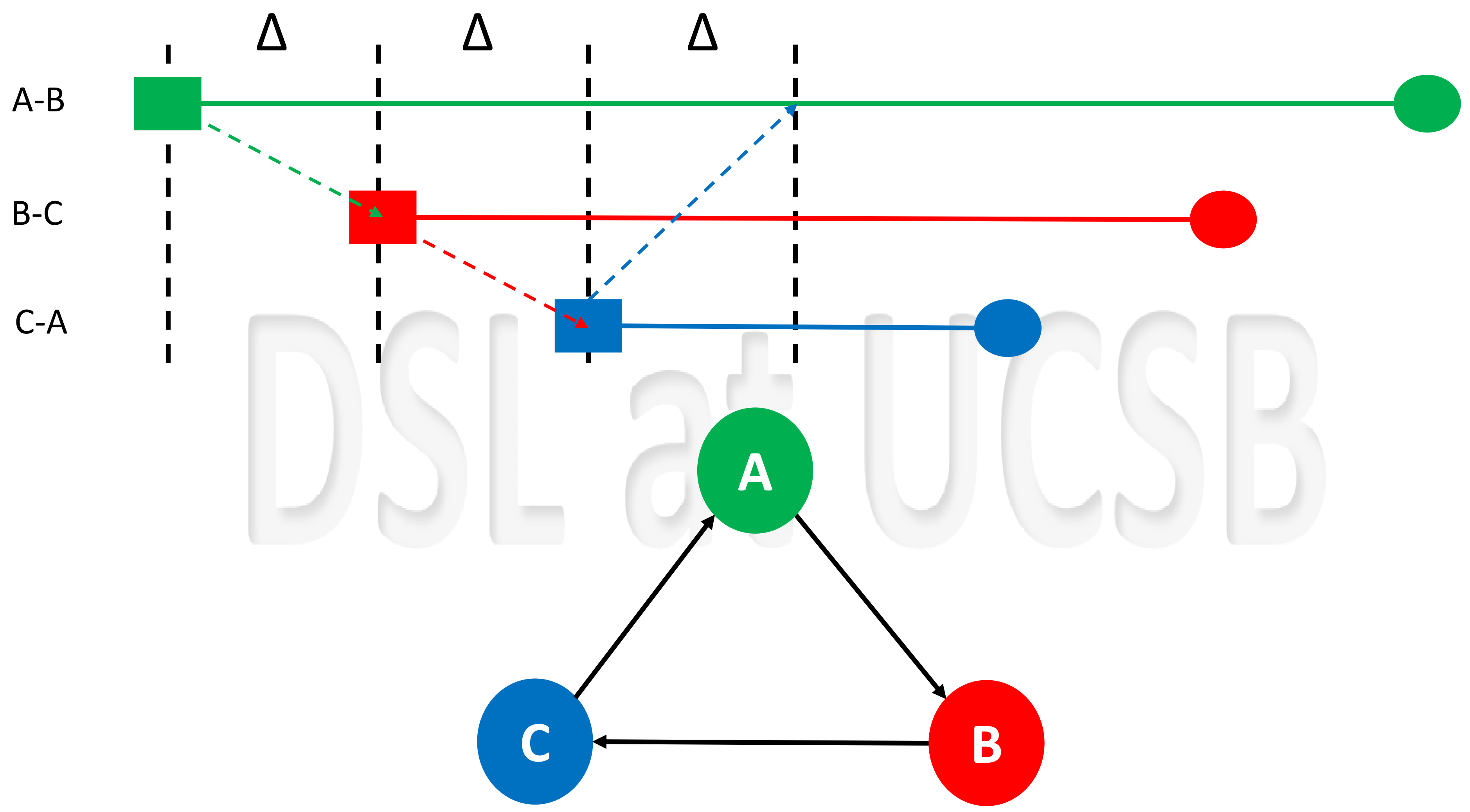
DSL at UCSB

Multi-party Atomic Swap Example



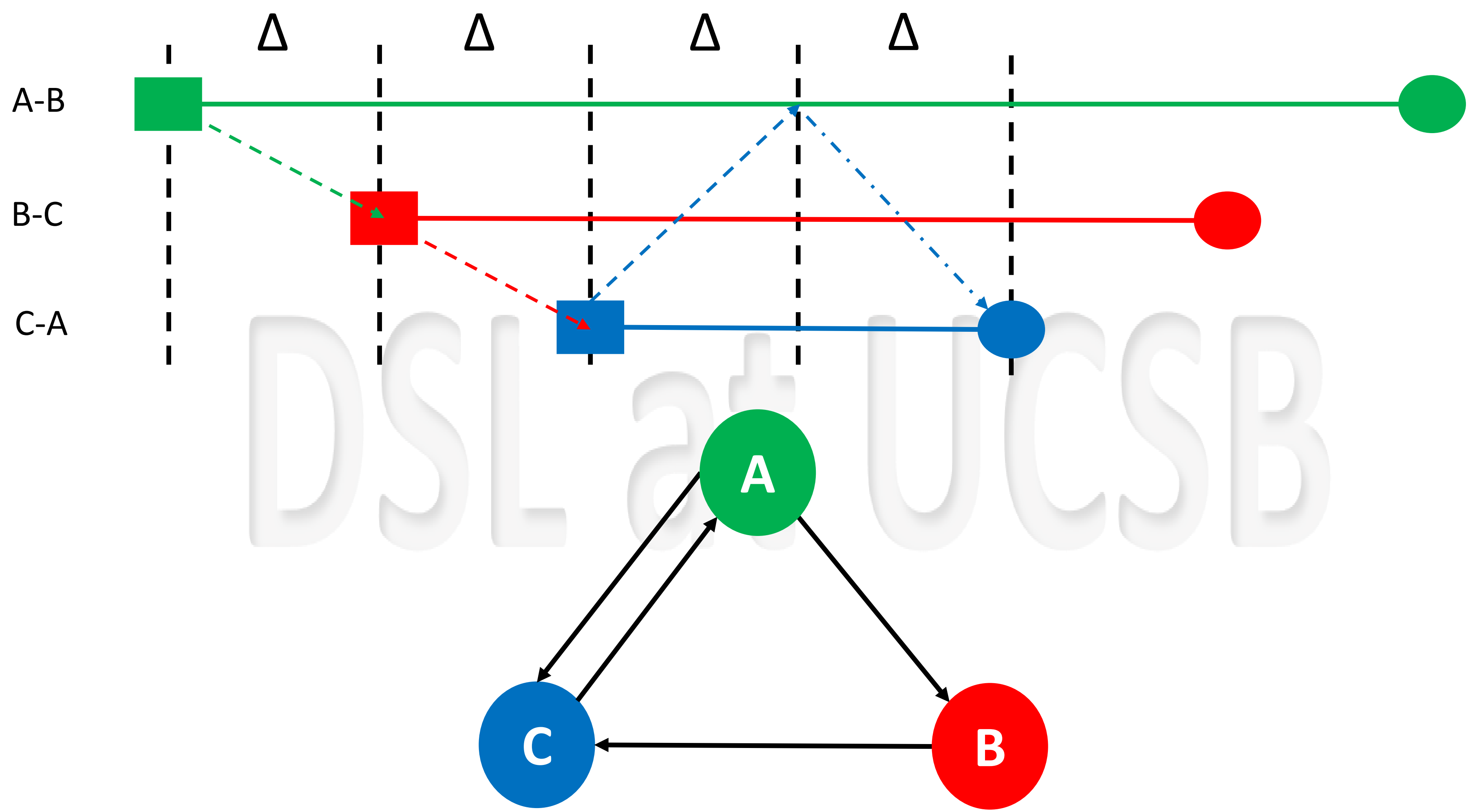
DSL at UCSB

Multi-party Atomic Swap Example



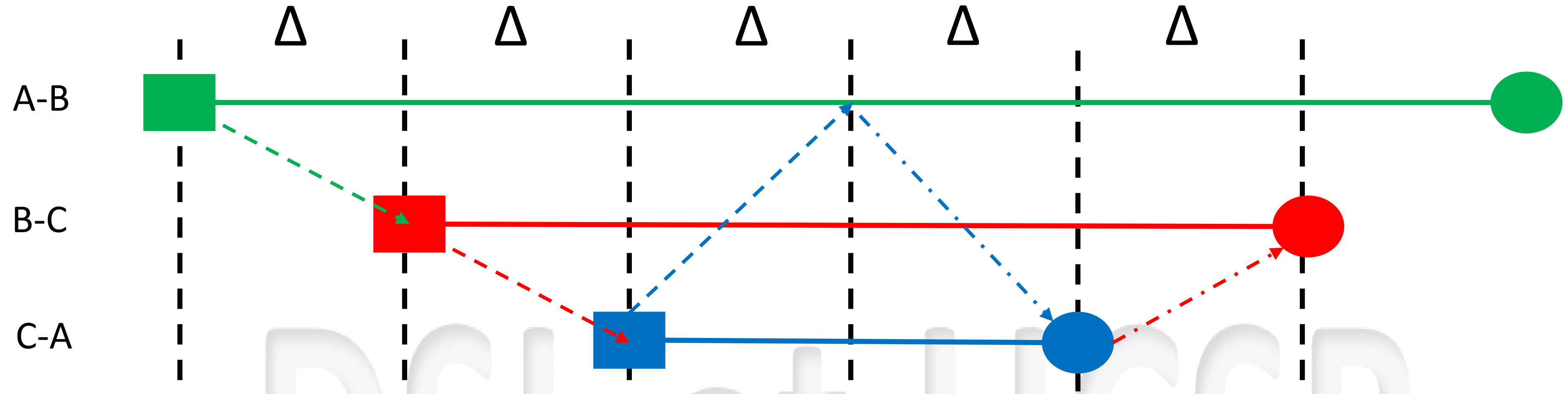
DSL at UCSB

Multi-party Atomic Swap Example

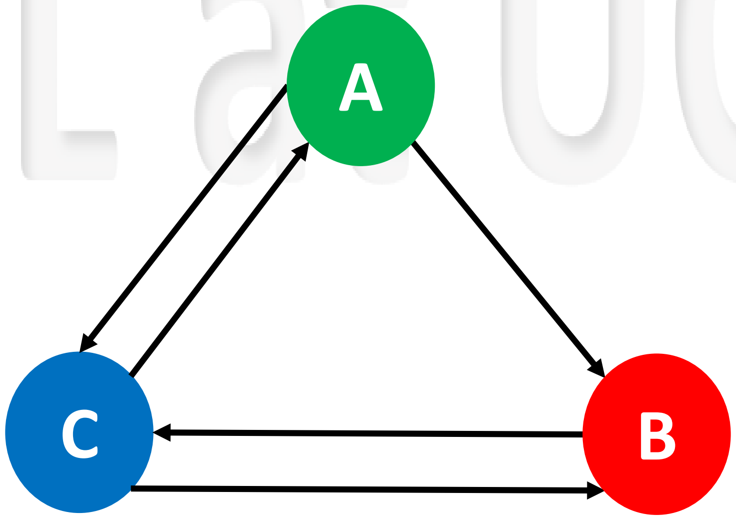


DSL at UCSB

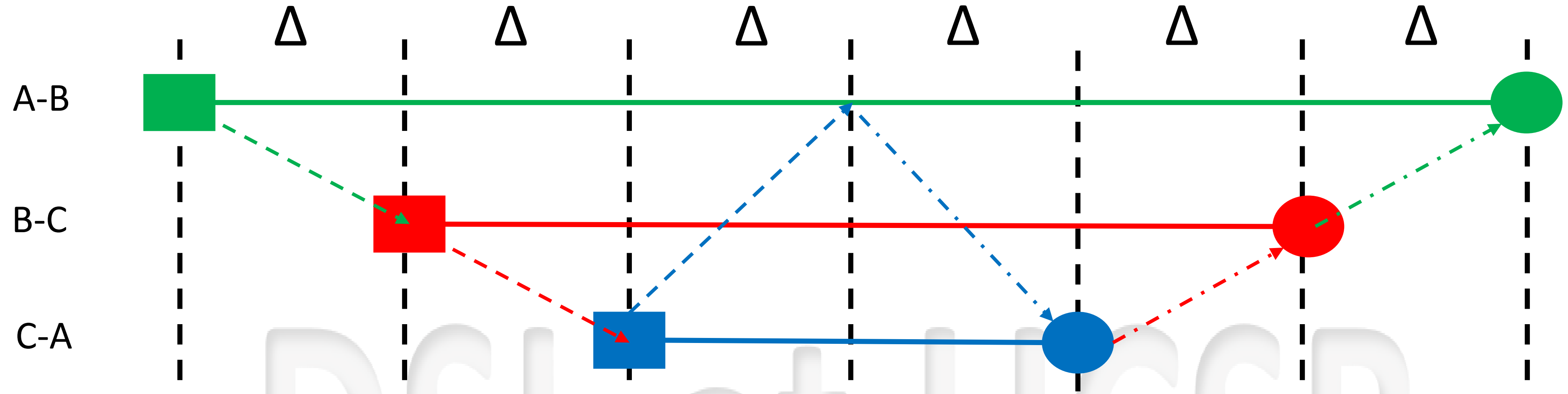
Multi-party Atomic Swap Example

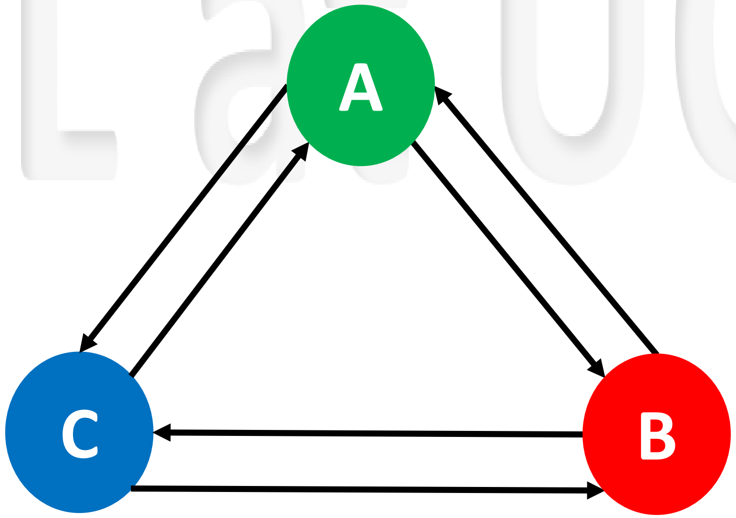


DSL at UCSB



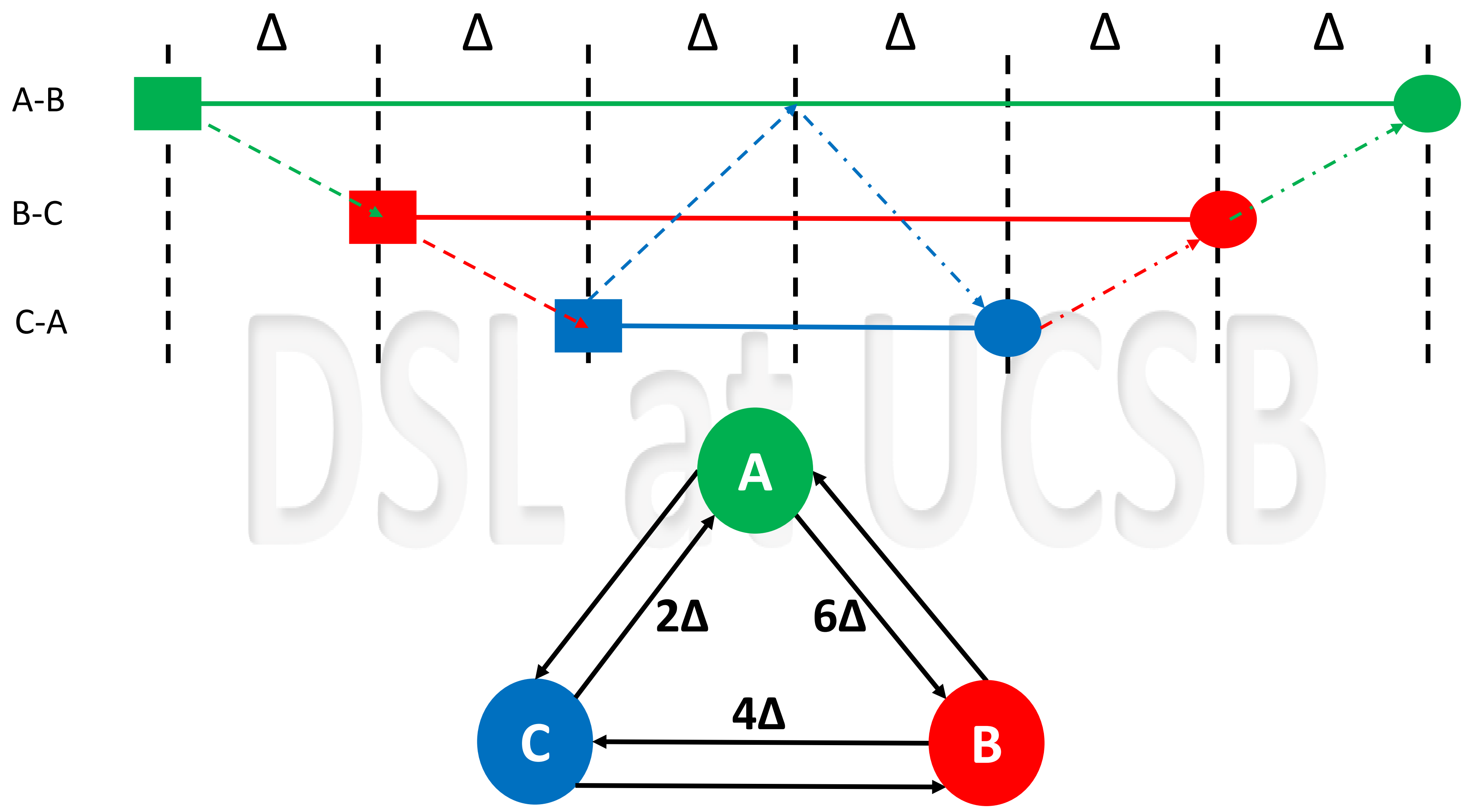
Multi-party Atomic Swap Example





DSL at UCSB

Multi-party Atomic Swap Example

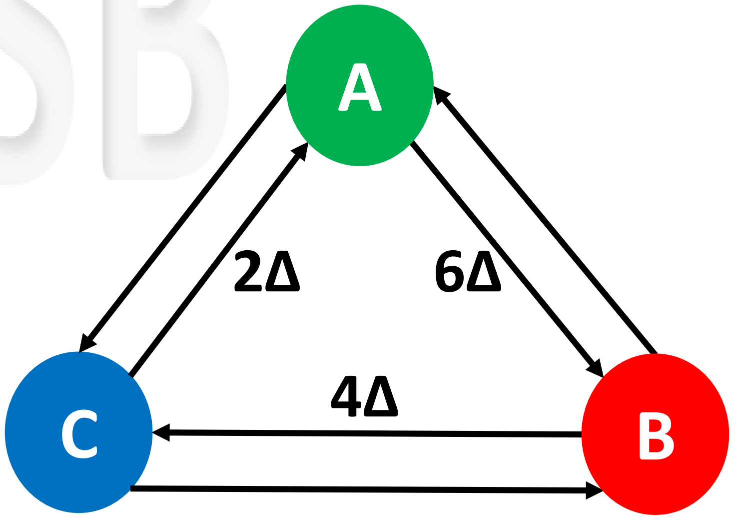


DSL at UCSB

Multi-party Atomic Swap Example

- v’ is the leader (A in this case)

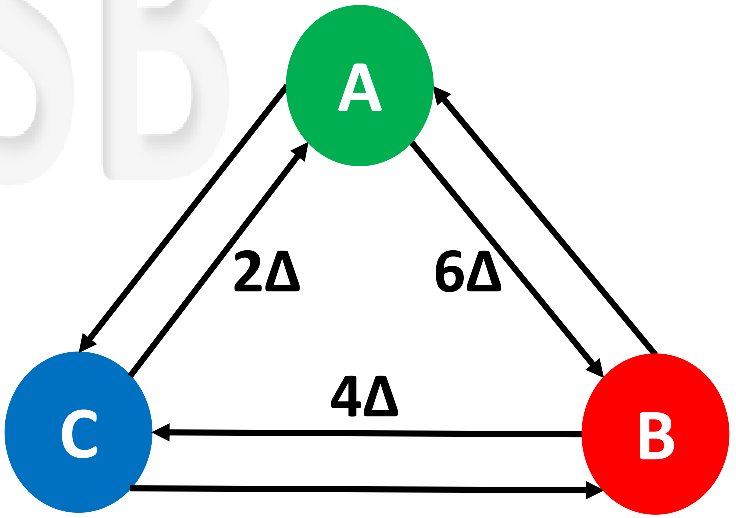
DSL at UCSB



Multi-party Atomic Swap Example

- v’ is the leader (A in this case)
- D(v, v’) the length of the longest path from node v to v’

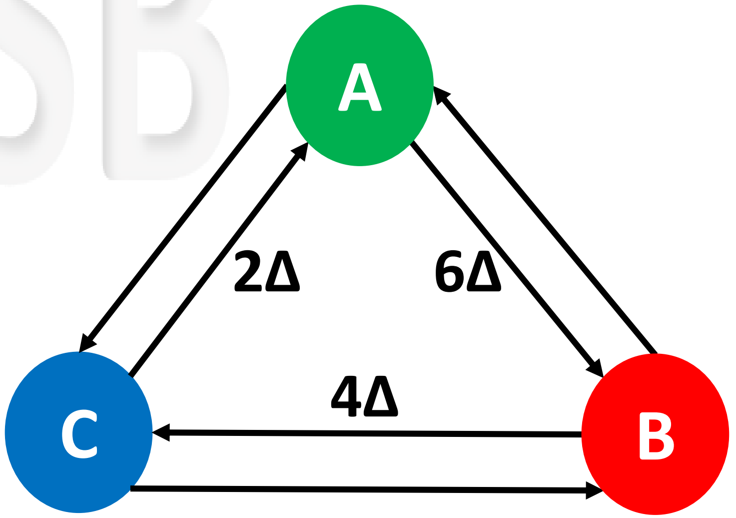
DSL at UCSB



Multi-party Atomic Swap Example

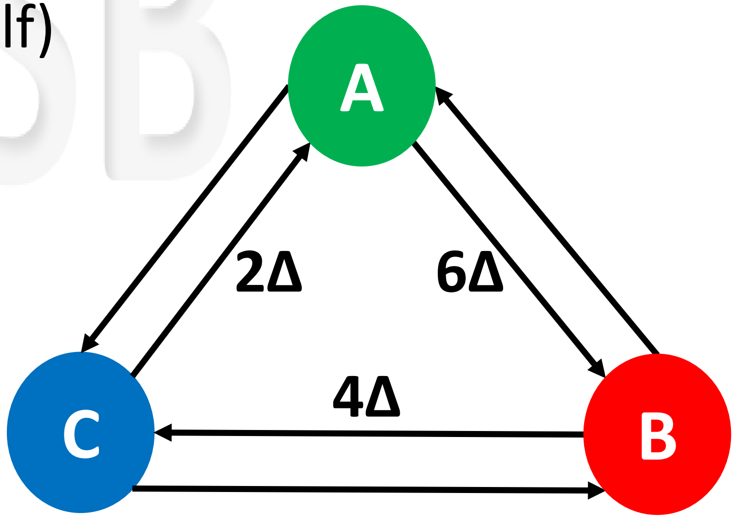
- v’ is the leader (A in this case)
- D(v, v’) the length of the longest path from node v to v’
- D(A, A) = 0, D(B, A) = 2, D(C, A) = 1

DSL at UCSB



Multi-party Atomic Swap Example

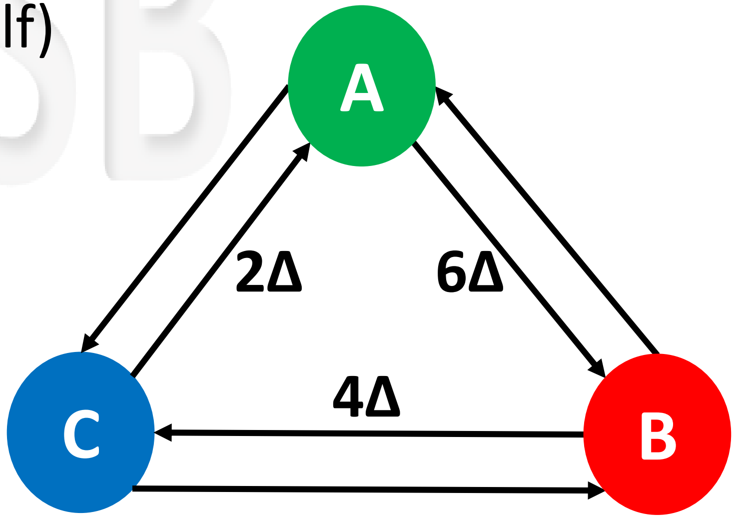
- v' is the leader (A in this case)
- D(v, v') the length of the longest path from node v to v'
- D(A, A) = 0, D(B, A) = 2, D(C, A) = 1
- Diam(D) is the diameter of Graph D
  - Longest path from one node to another (including itself)
  - Diam(D) = 3



DSL at UCSB

Multi-party Atomic Swap Example

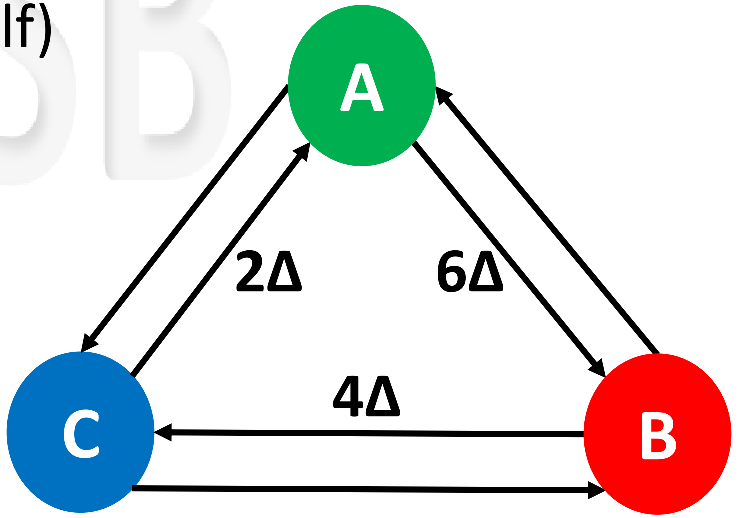
- v’ is the leader (A in this case)
- D(v, v’) the length of the longest path from node v to v’
- D(A, A) = 0, D(B, A) = 2, D(C, A) = 1
- Diam(D) is the diameter of Graph D
  - Longest path from one node to another (including itself)
  - Diam(D) = 3
- Hashlock on (u,v) = 2 . (D(v, v’) + 1) . Δ



DSL at UCSB

Multi-party Atomic Swap Example

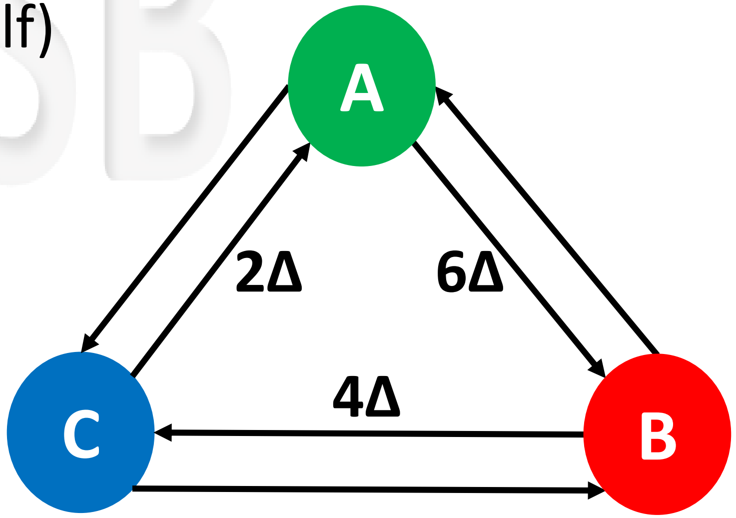
- v' is the leader (A in this case)
- D(v, v') the length of the longest path from node v to v'
- D(A, A) = 0, D(B, A) = 2, D(C, A) = 1
- Diam(D) is the diameter of Graph D
  - Longest path from one node to another (including itself)
  - Diam(D) = 3
- Hashlock on (u,v) = 2 . (D(v, v') + 1) . Δ
- D(v, v') + 1 [path from u to v'] creation path



DSL at UCSB

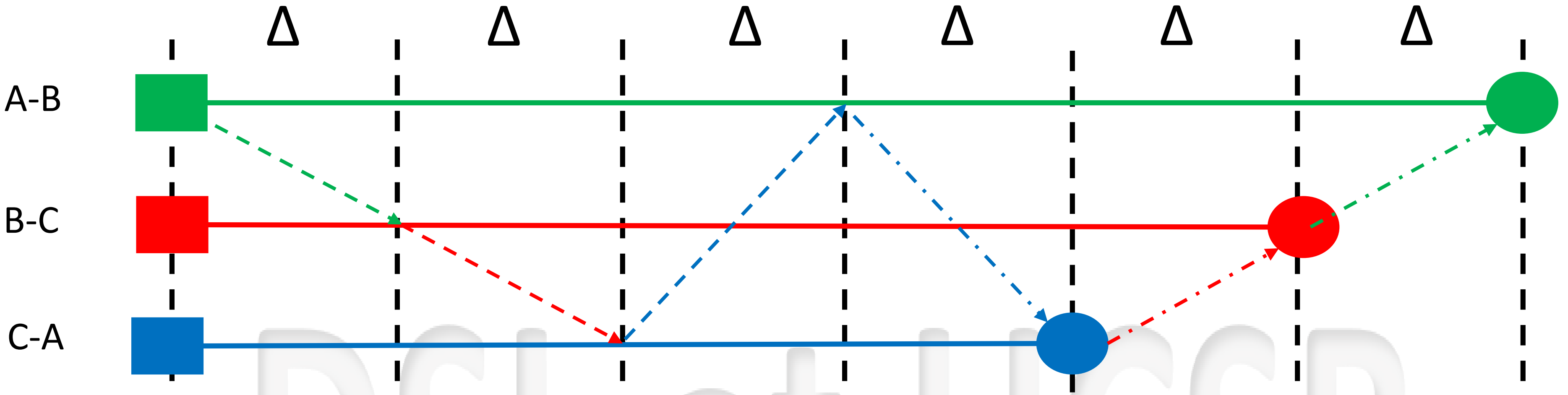
Multi-party Atomic Swap Example

- v' is the leader (A in this case)
- D(v, v') the length of the longest path from node v to v'
- D(A, A) = 0, D(B, A) = 2, D(C, A) = 1
- Diam(D) is the diameter of Graph D
  - Longest path from one node to another (including itself)
  - Diam(D) = 3
- Hashlock on (u,v) = 2 . (D(v, v') + 1) . Δ
- D(v, v') + 1 [path from u to v'] creation path
- D(v, v') + 1 [path from v' to u] redemption path

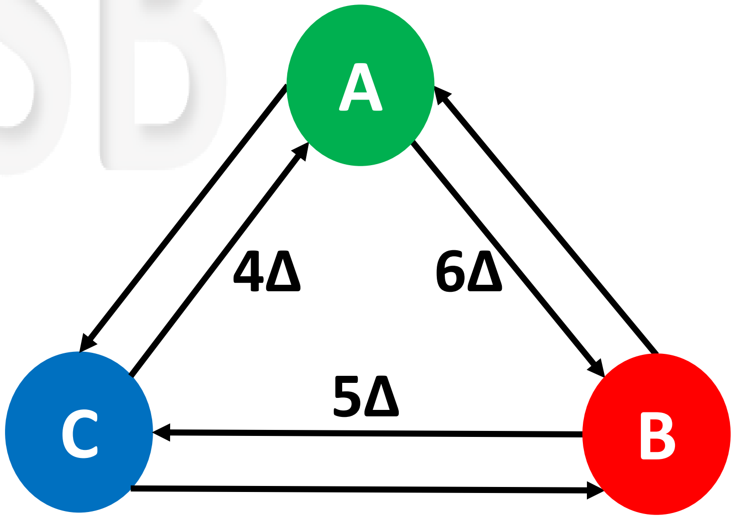


DSL at UCSB

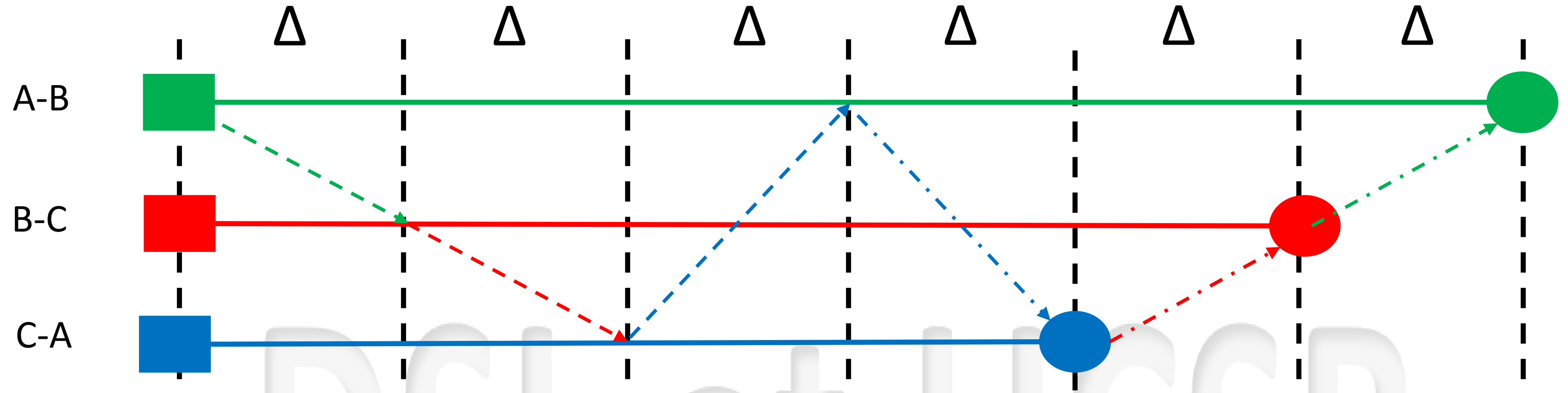
Multi-party Atomic Swap Example



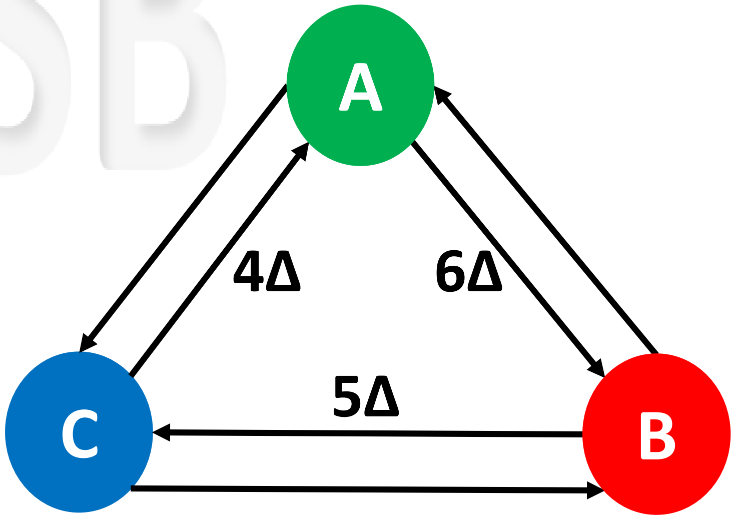
DSL at UCSB



Multi-party Atomic Swap Example

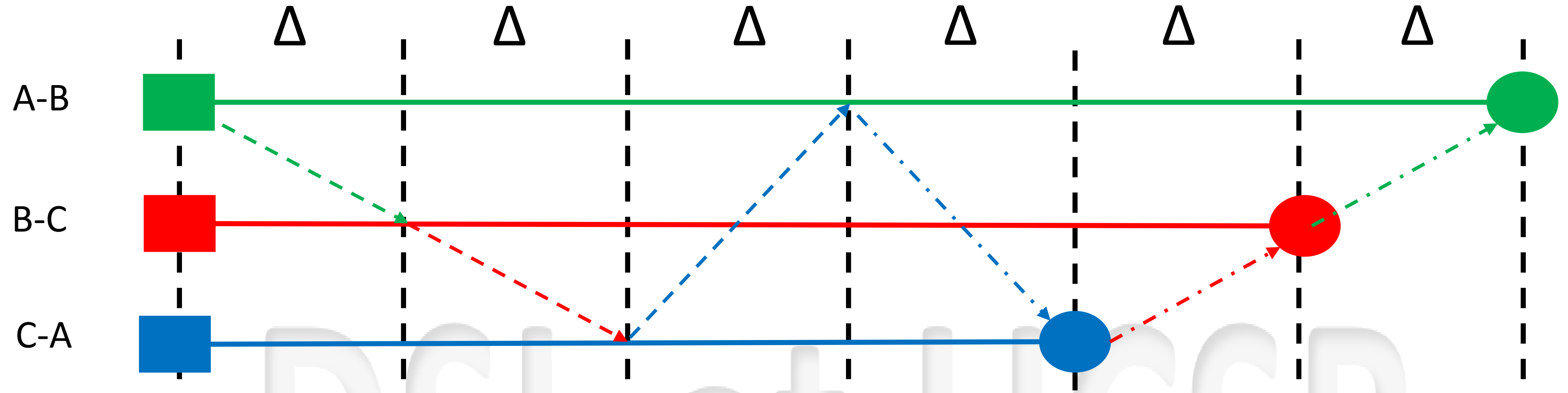


- Hashlock on (u,v) = (Diam(D) + D(v, v') + 1) . Δ



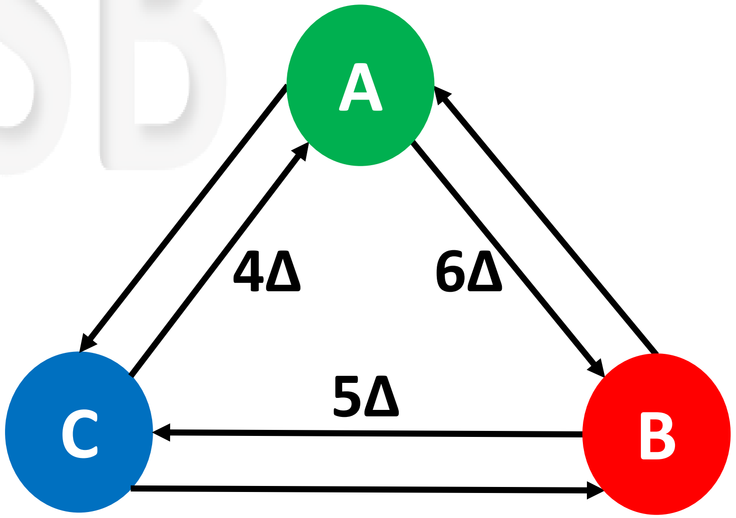
DSL at UCSB

Multi-party Atomic Swap Example

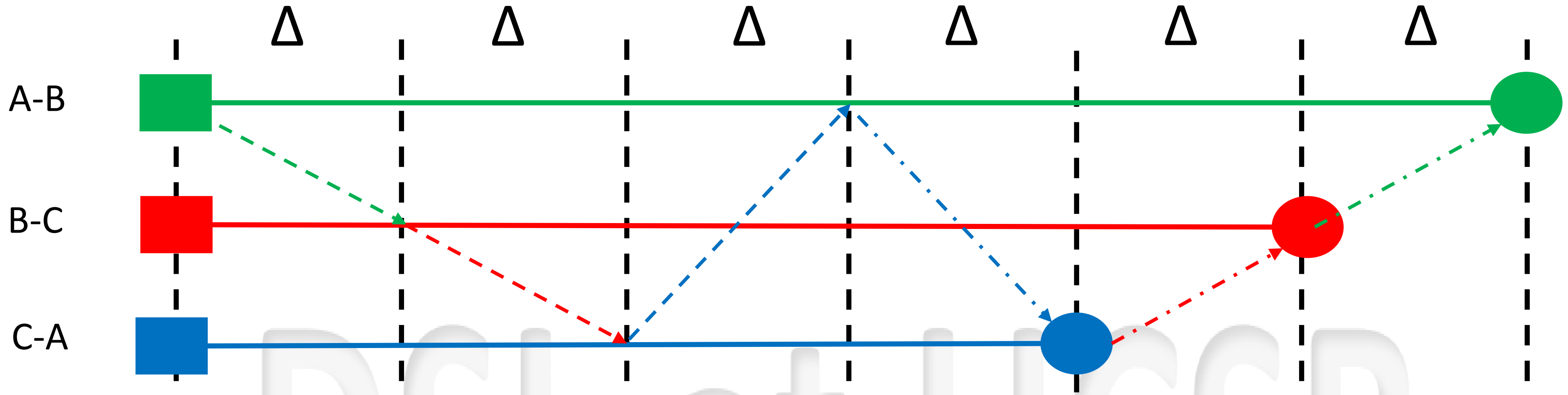


DSL at UCSB

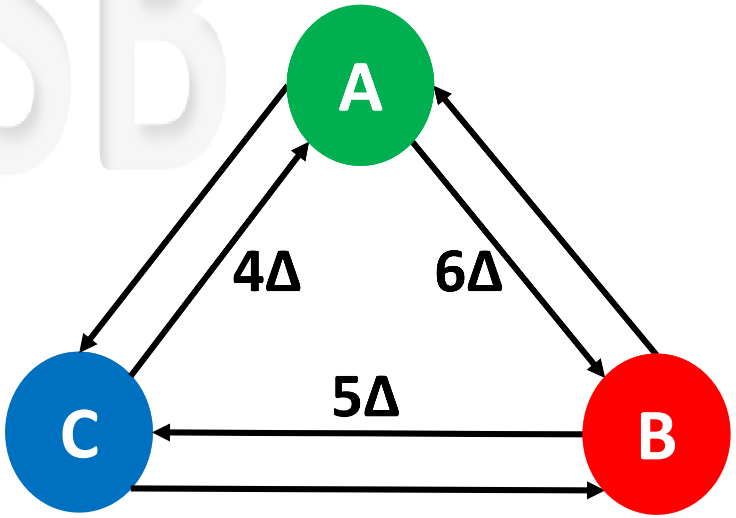
- Hashlock on (u,v) = (Diam(D) + D(v, v') + 1) . Δ
- Hashlock on A-B = (3 + 2 + 1) . Δ = 6Δ



Multi-party Atomic Swap Example

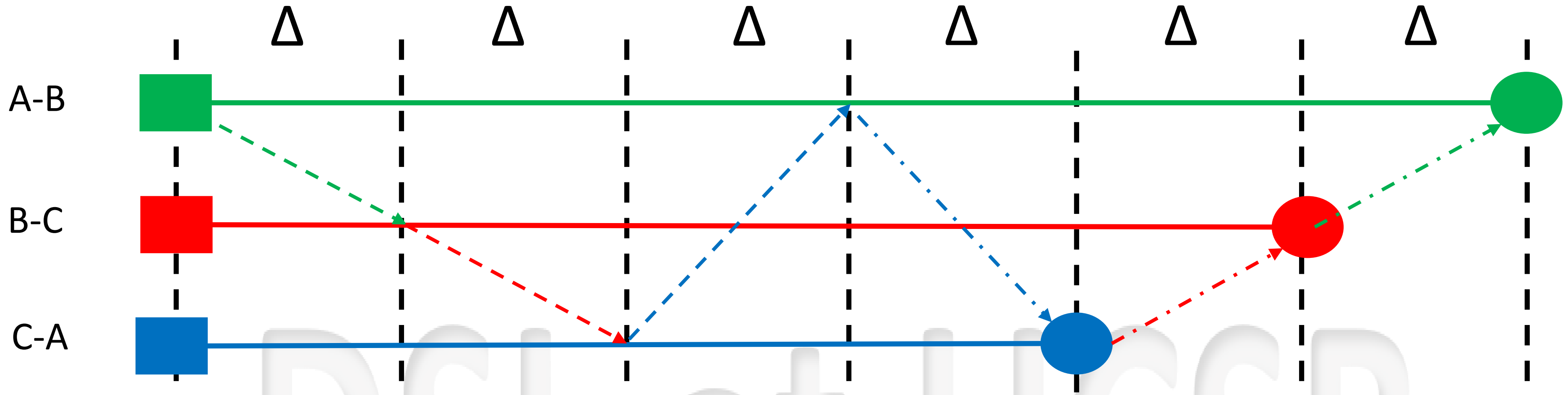


- Hashlock on (u,v) = (Diam(D) + D(v, v’) + 1) . Δ
- Hashlock on A-B = (3 + 2 + 1) . Δ = 6Δ
- Hashlock on B-C = (3 + 1 + 1) . Δ = 5Δ

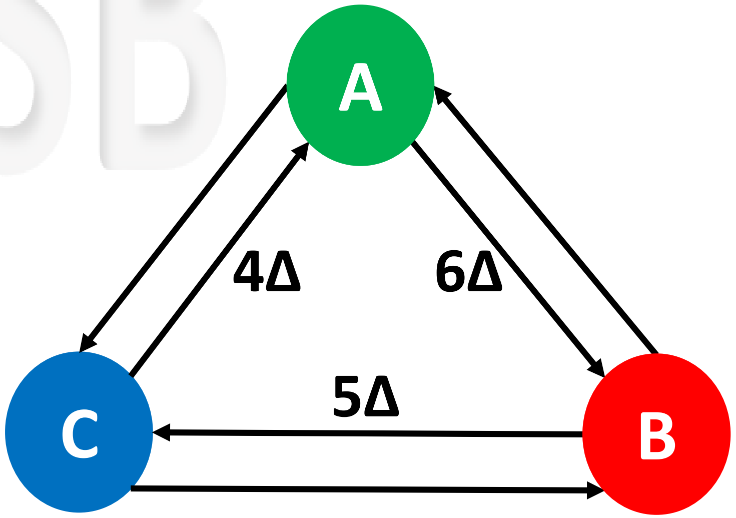


DSL at UCSB

Multi-party Atomic Swap Example



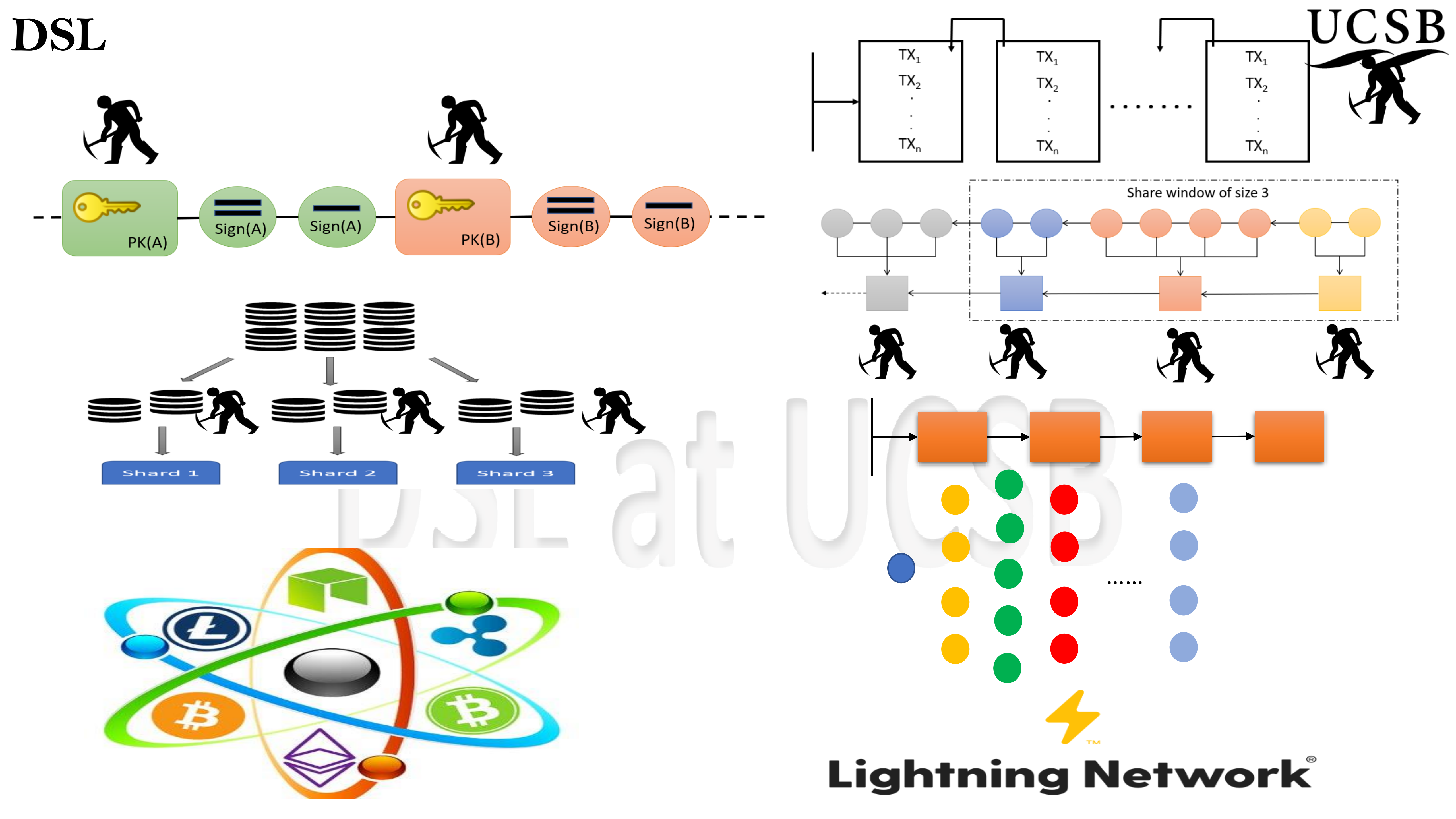
- Hashlock on (u,v) = (Diam(D) + D(v, v') + 1) . Δ
- Hashlock on A-B = (3 + 2 + 1) . Δ = 6Δ
- Hashlock on B-C = (3 + 1 + 1) . Δ = 5Δ
- Hashlock on C-A = (3 + 0 + 1) . Δ = 4Δ



DSL at UCSB

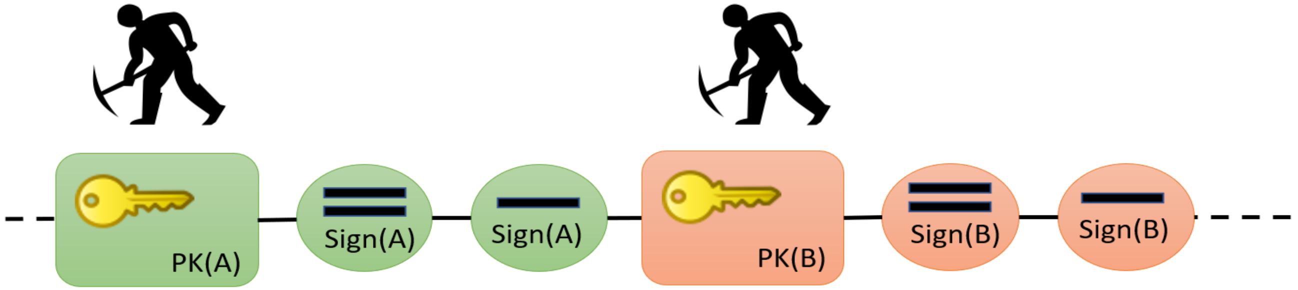
Lightning Network

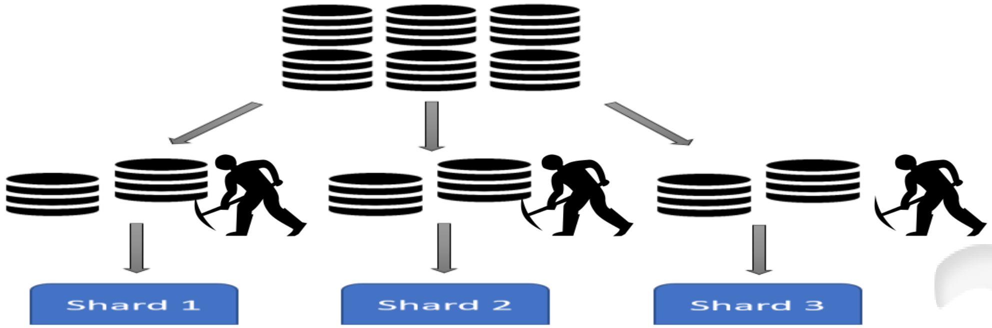
DSL at UCSB

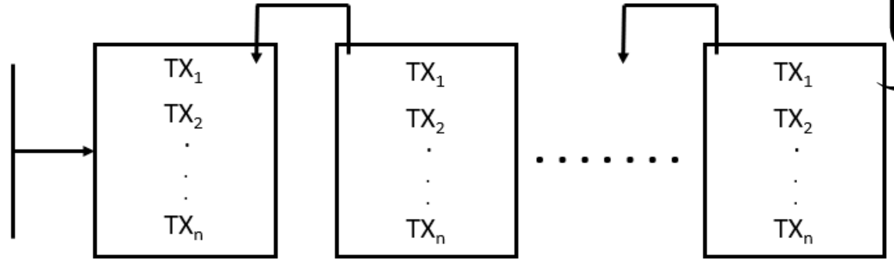

DSL
UCSB
TX1
TX2
TXn
Share window of size 3
PK(A)
Sign(A)
Sign(A)
PK(B)
Sign(B)
Sign(B)
Shard 1
Shard 2
Shard 3
DSL at UCSB
Lightning Network

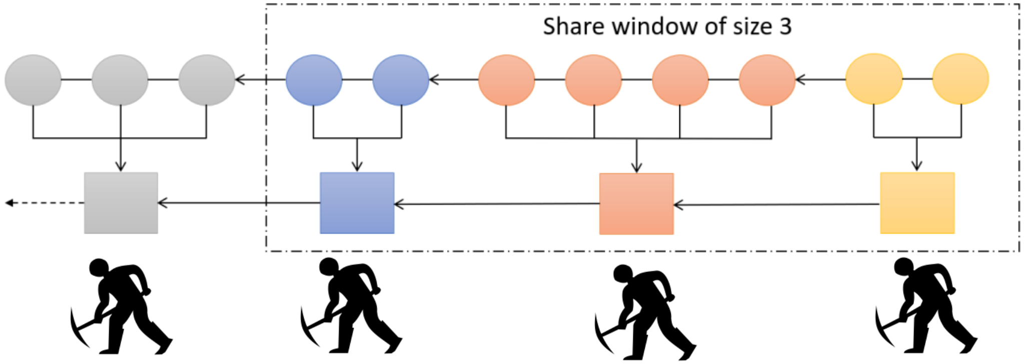
DSL

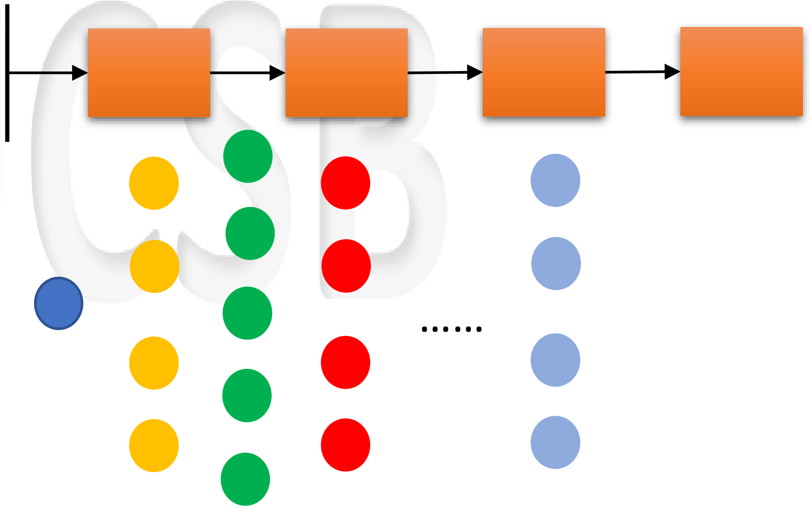
UCSB









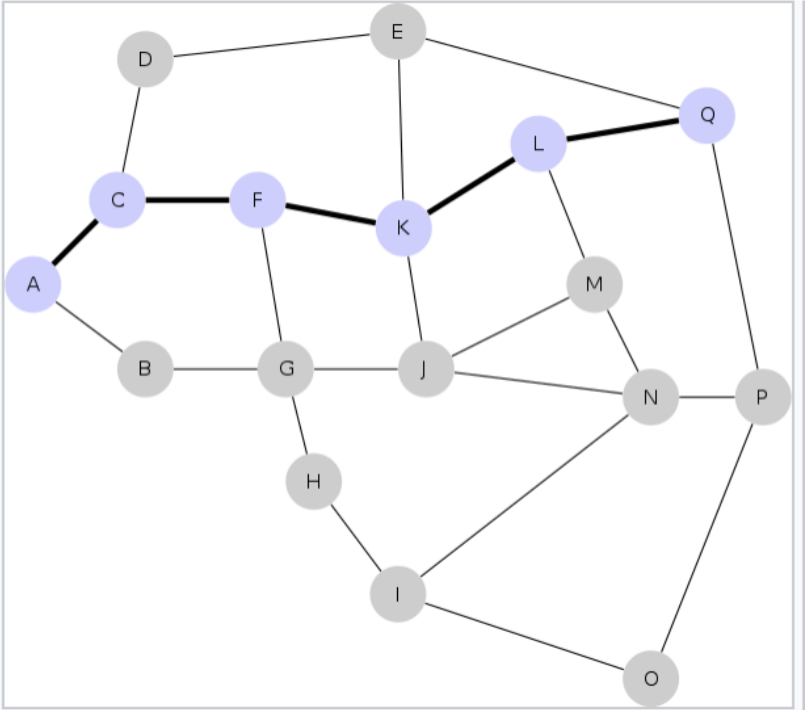


Lightning Network®

DSL at UCSB

What is Lightning Network?

“Lightning is a decentralized network using smart contract functionality in the blockchain to enable instant payments across a network of participants.”



https://lightning.network/

DSL at

The Setting: Two-party transactions

- Alice and Bob frequently need to transact with each other:
  - Alice → Bob: $x
  - …
  - Bob → Alice: $y
  - ….
- Each of the above transaction can be put on-chain.
- Is there an alternative?

DSL at UCSB

The Idea of Lightening

- Frequent two-party interactions can be modeled as off-chain transactions.
- On-chain interaction only to establish payment channels between Alice and Bob.
- The key challenge:
  - Off-chain interactions must remain honest, i.e., prevent Alice or Bob trying to cheat each other.

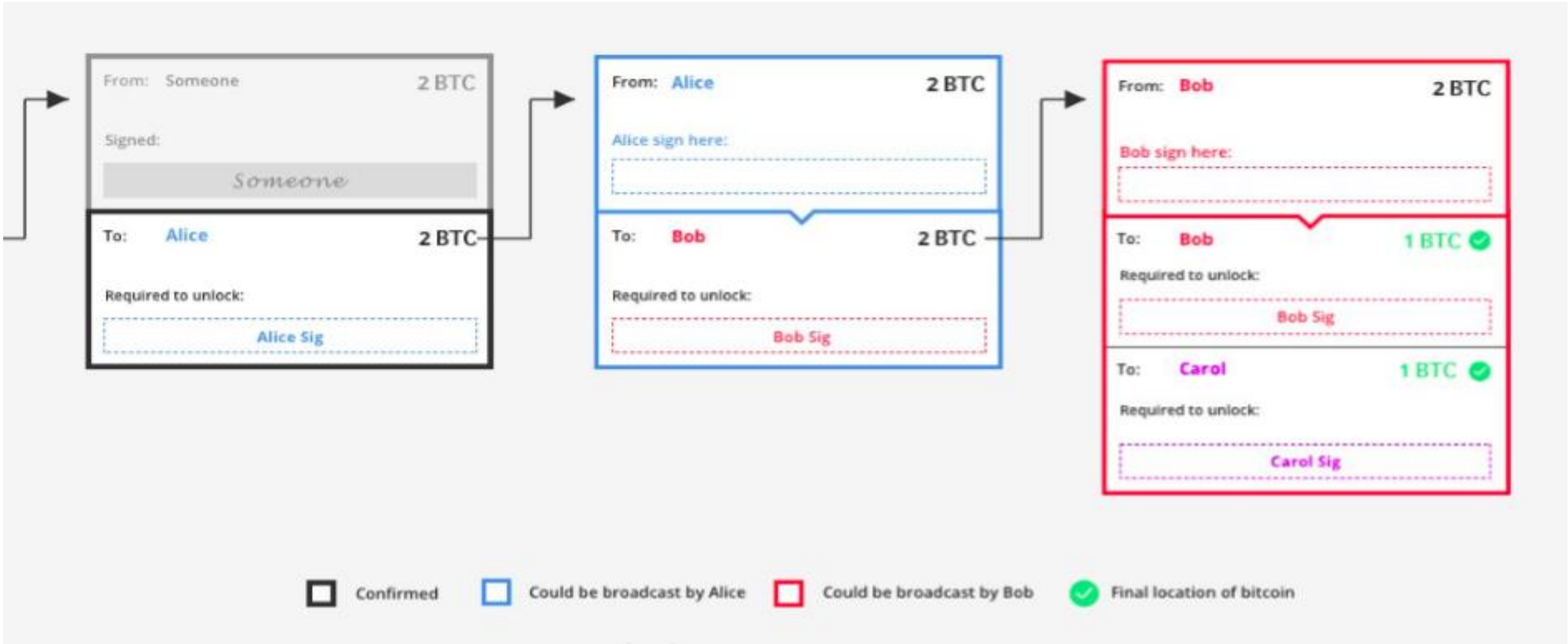
DSL at UCSB

Outline of the protocol

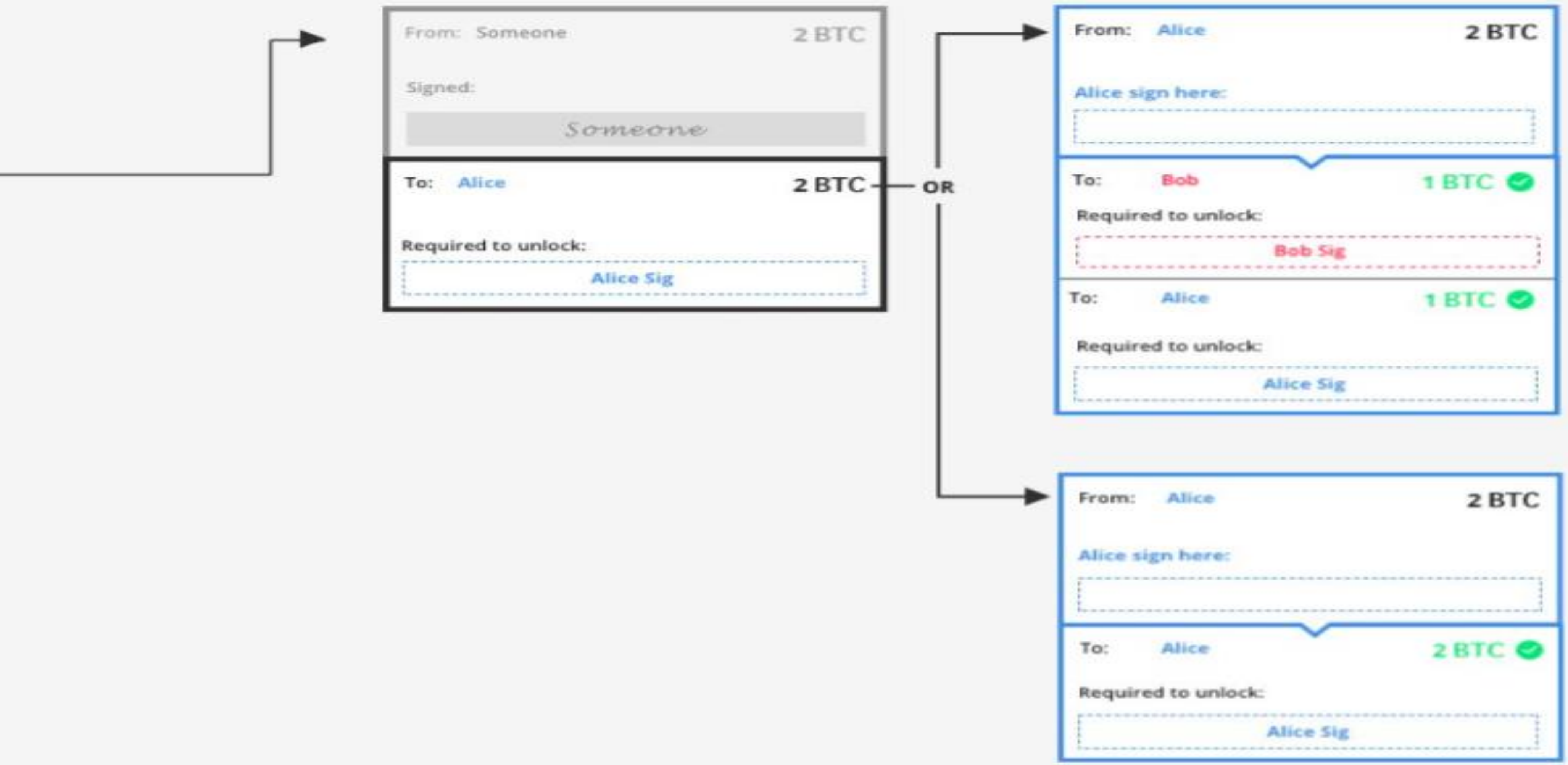
1. Open a bidirectional channel
   a. Both parties make deposits to a shard on-chain wallet
2. Initiate a transaction by making a contract
   a. Signed by both parties
3. Update the contract when making more transactions
   a. Keep exchanging the updated contract off-chain
4. Push the most updated contract to the blockchain to withdraw
   a. Thus the bidirectional channel is closed

DSL at UCSB

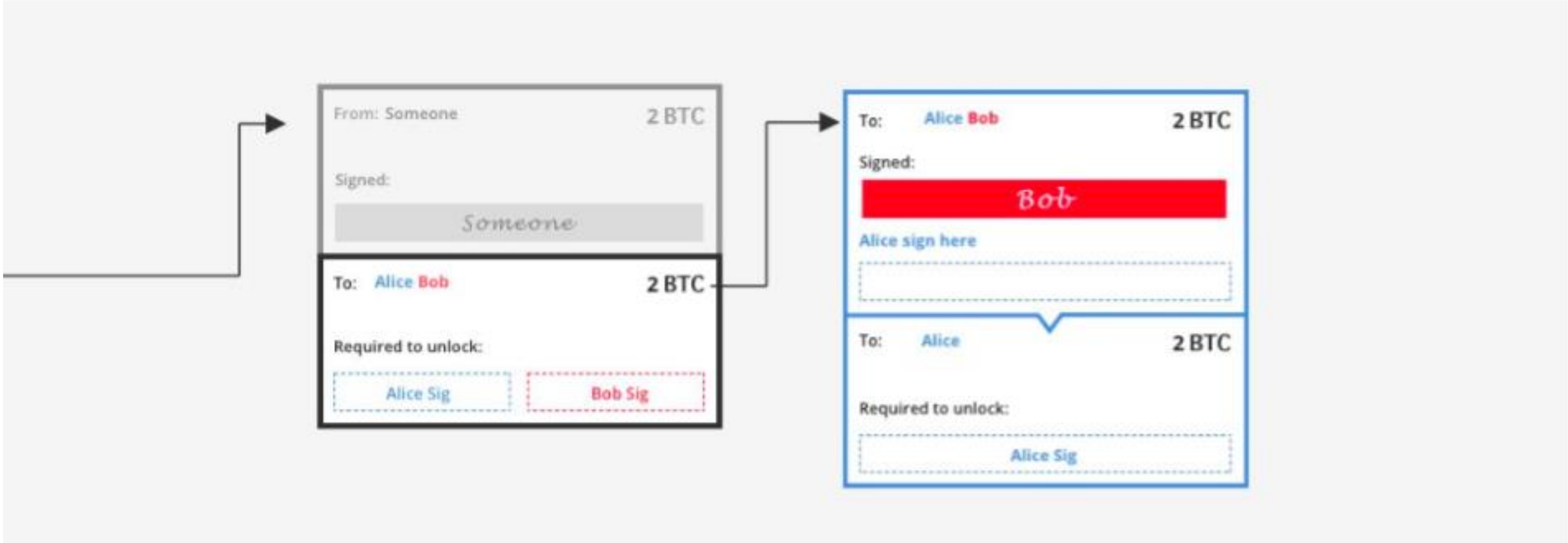
Building Block #1: Transactions



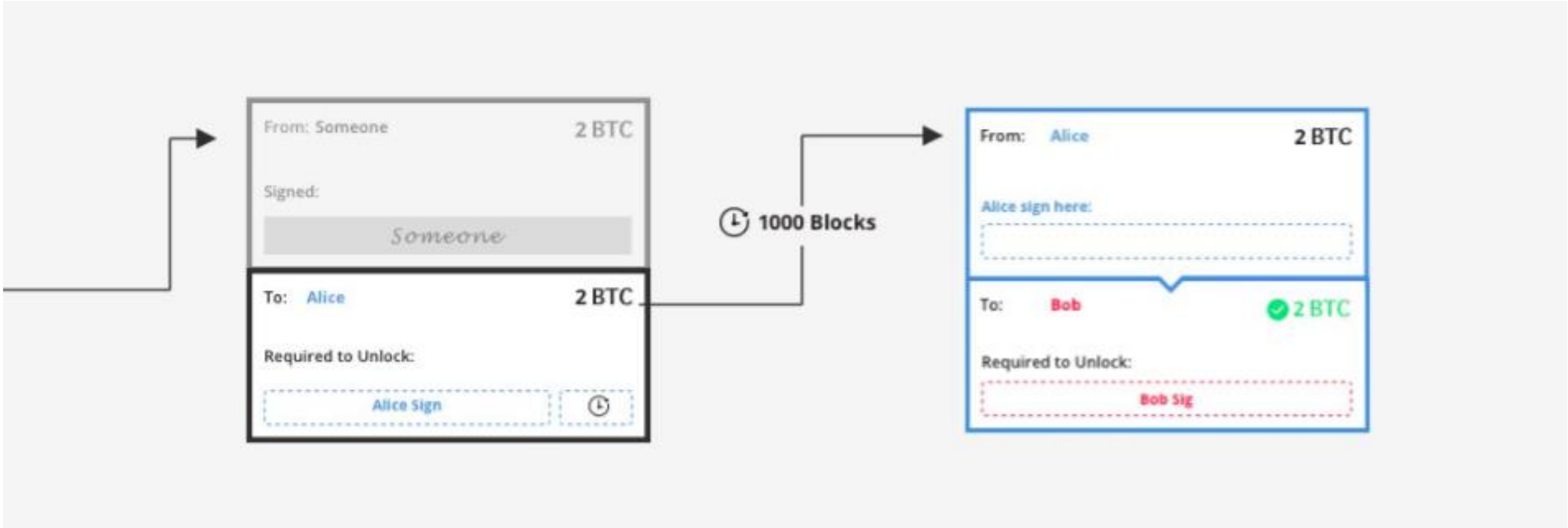
Building Block #2: Double Spend



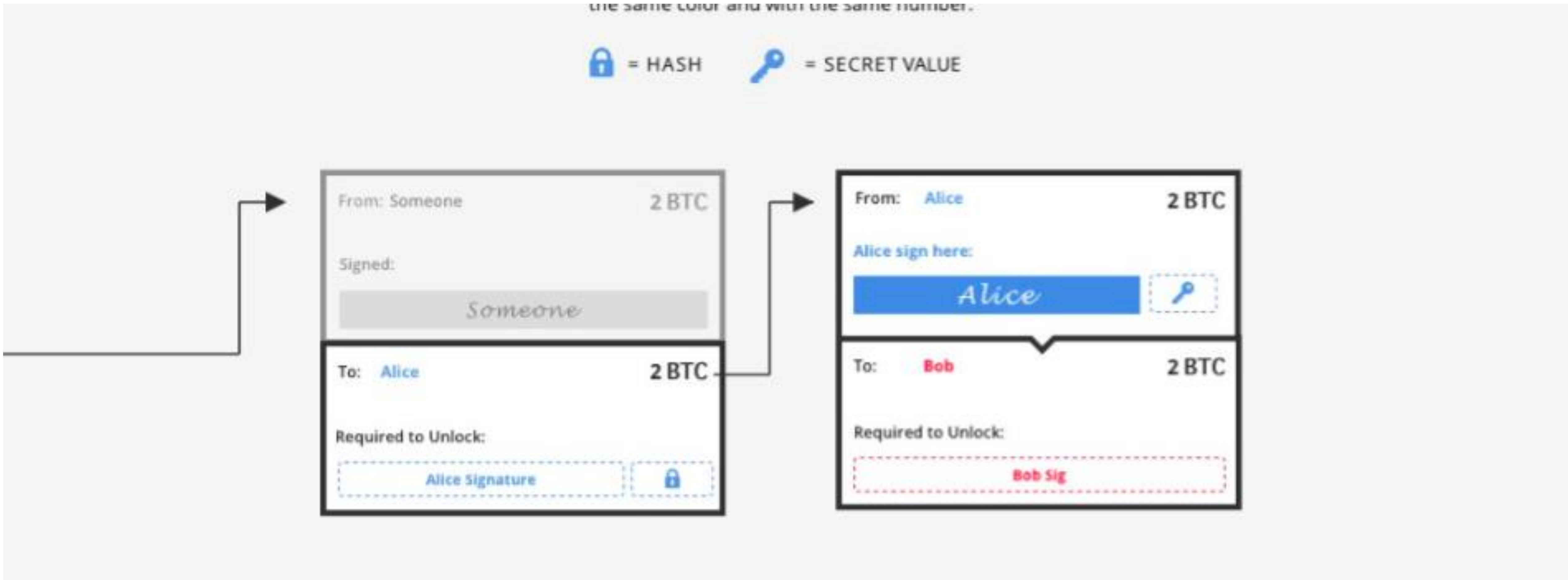
Building Block #3: Multi-signature (2-of-2)



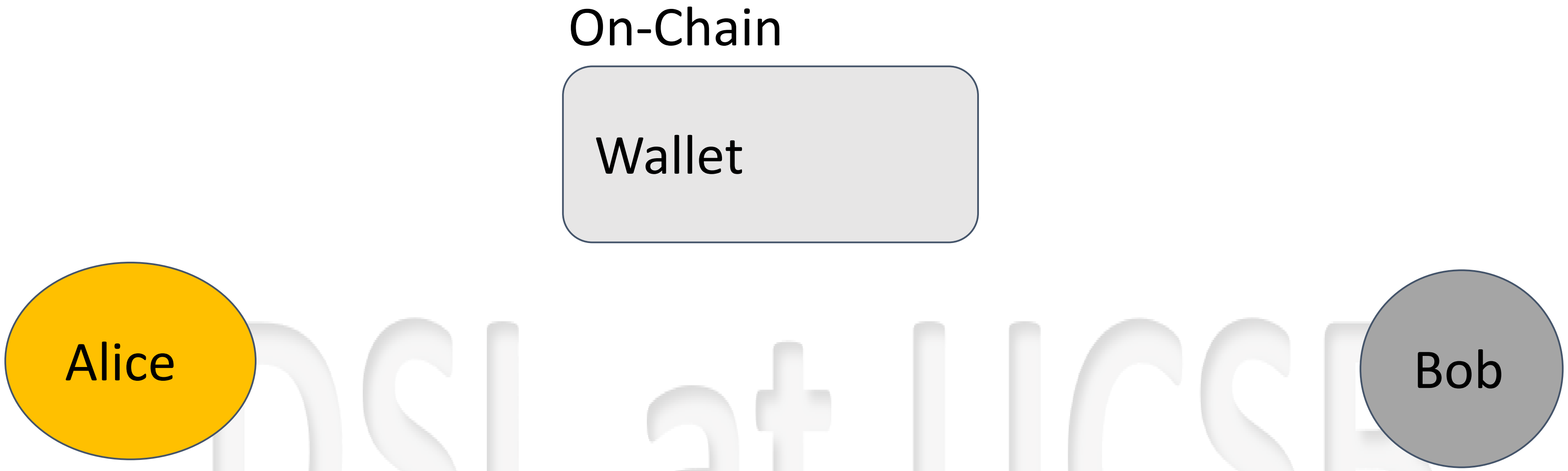
Building Block #4: TimeLock



Building Block #5: Hash Values & Secrets



Lightning Network <high level protocol>

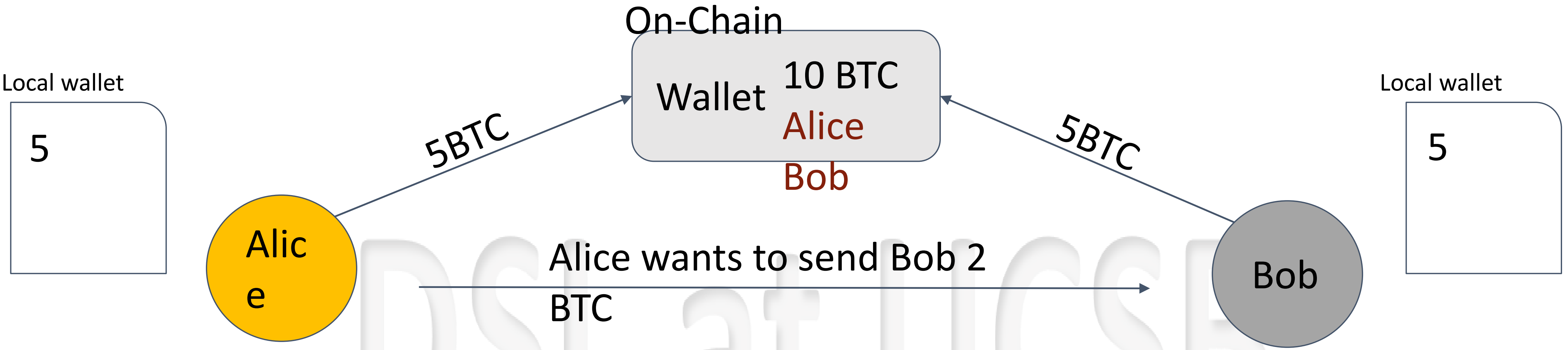


DSL at UCSB

Lightning Network <high level protocol>

DSL at UCSB

Lightning Network <high level protocol>



DSL at UCSB

Alice ➔ Bob: $2.0

- Alice sends $3.0 to herself,
- $7.0 to a multisig address:
    - can be unlocked by Bob on his own, but after 1000 blocks have been mined
    - Or, it can be opened by Alice on her own, but only if she includes the **S** of **H(S)** from Bob.
- Alice signs her end of this commitment transaction, and gives it to Bob.
- Bob does the same: $7.0 to himself; and $3.0 to multisig address with TimeLock & HashLock.

DSL at UCSB

Alice ➔ Bob: $2.0 (contd.)

- Both Alice and Bob could sign and broadcast the half-valid transaction.
- If Alice does:
  - Bob gets $7.0 immediately but Alice must wait for 1000 blocks
- If Bob does:
  - Alice get $3.0 immediately but Bob must wait

➔ **Therefore, neither sign and broadcast their half of the transaction.**

DSL at UCSB

Updating the Payment Channel: Bob ➔ Alice: $1.0

- Bob:
  - $4.0 to multisig address (with TimeLock+HashLock)
  - $6.0 to himself
- Alice:
  - $ 4.0 to herself
  - $6.0 to multisig address (with TimeLock+HashLock)
- Alice & Bob hand each other their ***first secrets***

DSL at UCSB

Can Bob be dishonest?

- What is stopping Bob from broadcasting the first transaction and benefiting with $7.0 instead of $6.0?
- Bob is prevented from this because he has revealed the ***first*** secret to Alice:
  - Broadcasting will require him to wait 1000 blocks
  - Alice will have enough time to beat Bob and claim $7.0 for herself.

DSL at UCSB

Lightening Networks

- Closure of payment channel in Lightning Networks
- Extending the lightning networks from two-parties to multiple-parties:
  - Option 1:
    - N parties ➔ N^2 payment channels
  - Option 2:
    - Transitivity of Transactions via intermediaries
    - Alice ➔ Carol: (i) Alice ➔ Bob && (ii) Bob ➔ Carol

DSL at UCSB

Open Problems and Criticism

DSL at UCSB

Open Problems and Criticism

DSL at UCSB

Open Problems and Criticism

Bitcoin mining consumes more electricity a year than Ireland

The Guardian International edition

Network's estimated power use also exceeds that of 19 other European countries, consuming more than five times output of continent's largest windfarm

UCSB

DSL

UCSB

Open Problems and Criticism

Bitcoin mining consumes more electricity a year than Ireland

The Guardian International edition

Network’s estimated power use also exceeds that of 19 other Europe
contine

New study quantifies bitcoin’s ludicrous energy consumption

Bitcoin could consume 7.7 gigawatts by the end of 2018.

TIMOTHY B. LEE - 5/17/2018, 10:23 AM

ars TECHNICA

Open Problems and Criticism

Bitcoi
electri

Network's e
Europe
contine

Bitcoin Mining Now Consuming More Electricity Than 159 Countries Including Ireland & Most Countries In Africa

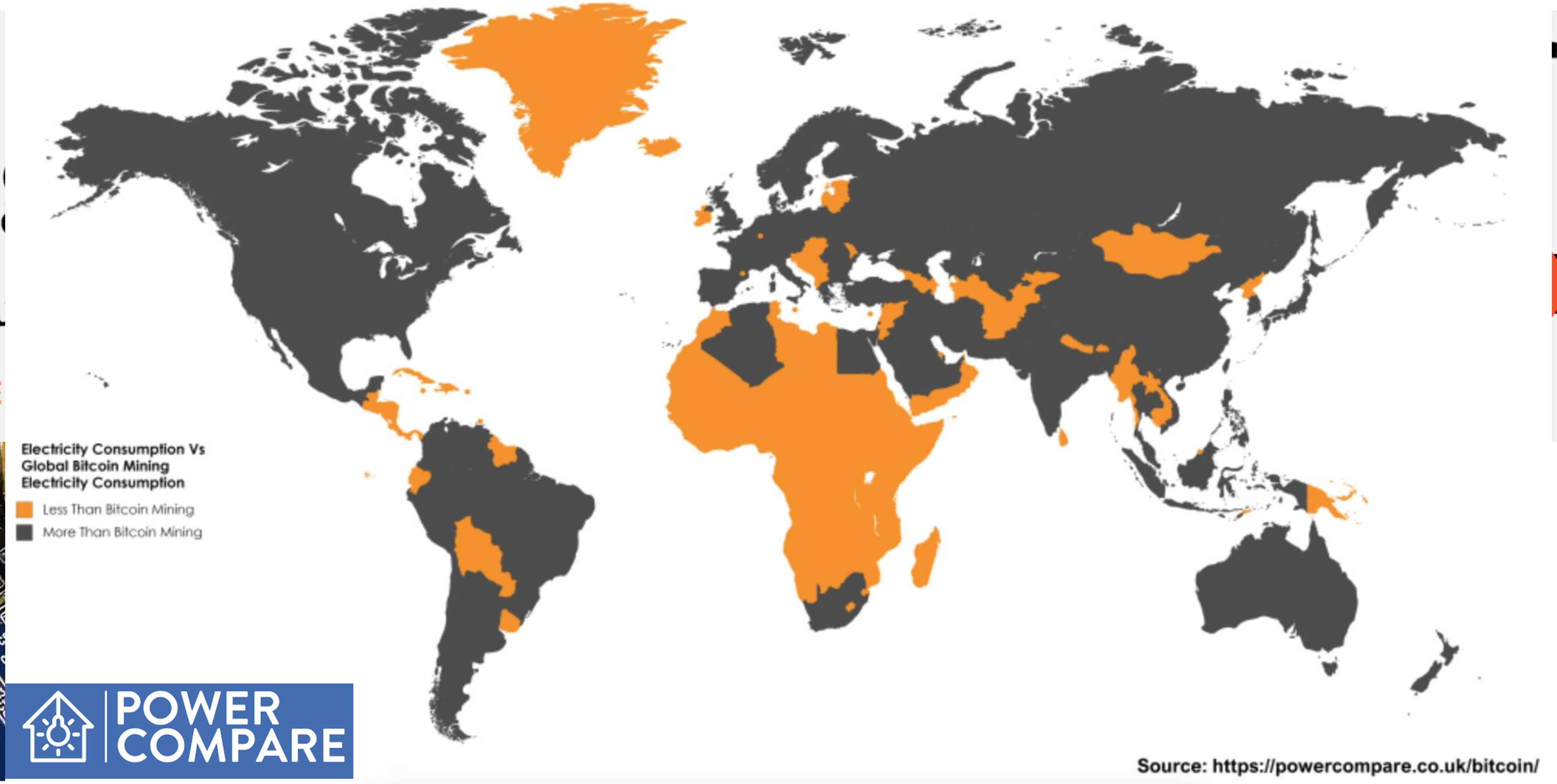
New
ener

Bitcoin cou

TIMOTHY B. LEE

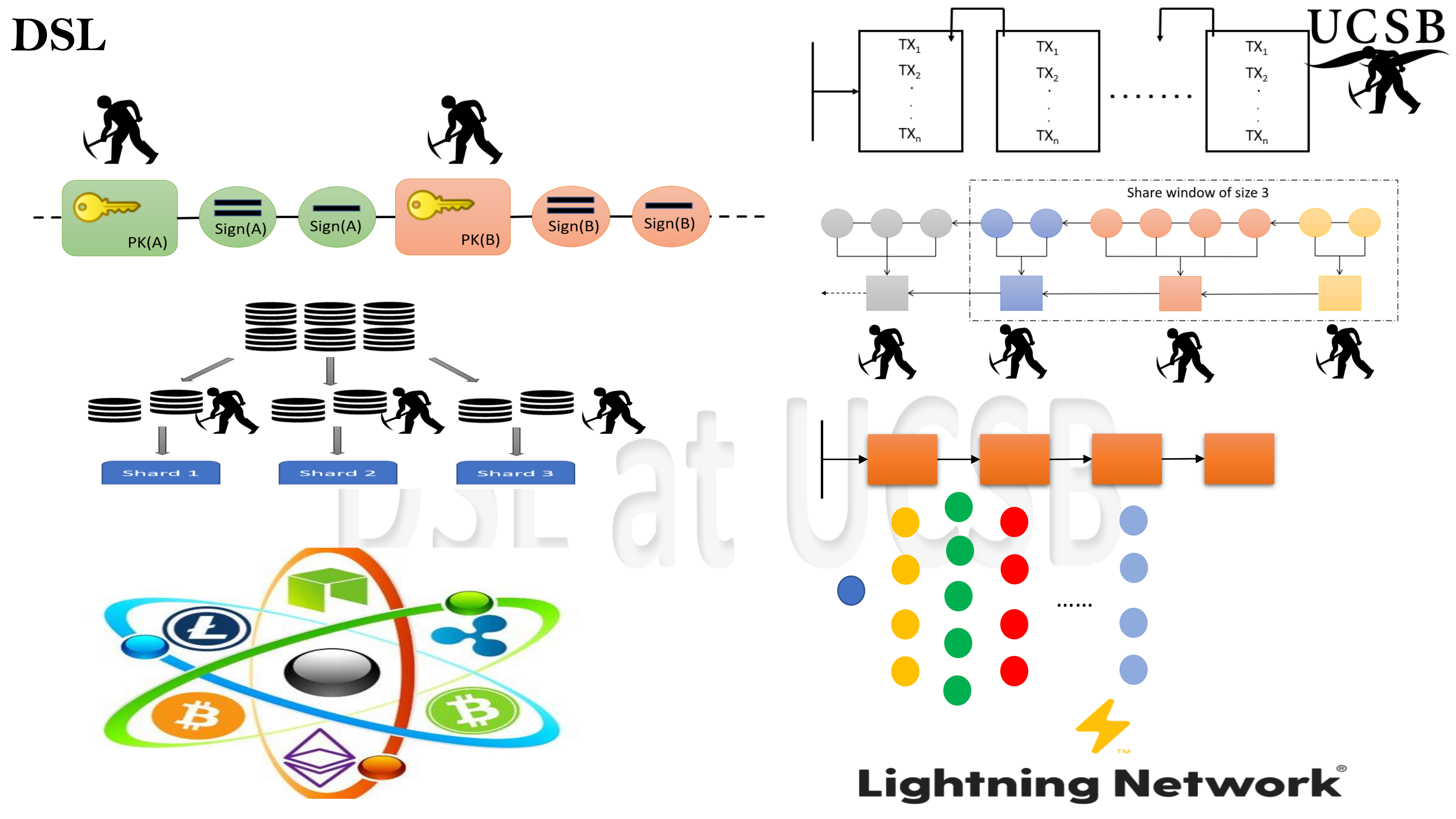
ous

TECHNICA



Questions and Open Discussion

DSL at UCSB


DSL
UCSB
TX1
TX2
TXn
Share window of size 3
PK(A)
Sign(A)
Sign(A)
PK(B)
Sign(B)
Sign(B)
Shard 1
Shard 2
Shard 3
DSL at UCSB
......
Lightning Network

Blockchain: Panacea for all our data problems?

- Resource cost:
  - Proof-of-work consumes resources at the planetary scale
- Mythical notion of democratization:
  - Handful of miners control the progress of Bitcoin blockchain
- False notion of security:
  - An Individual vulnerable to the security of his/her key
- Extreme distribution:
  - is it really worth it?
- Extreme redundancy:
  - is it really necessary?
- Social consequences:
  - Are we comfortable if this technology is used for dark causes?

DSL at UCSB

Contact Us

- Sujaya Maiyaa: sujaya_maiyya@ucsb.edu
- Victor Zakhary: victorzakhary@ucsb.edu
- Divyakant Agrawal: divyagrawal@ucsb.edu
- Amr El Abbadi: elabbadi@ucsb.edu

DSL at UCSB