Shortstack:
Distributed, Fault-tolerant, Oblivious Data Access



Midhul **Kushal** **Anurag** **Rachit**

Cornell University

Yale University

(First two authors contributed equally)

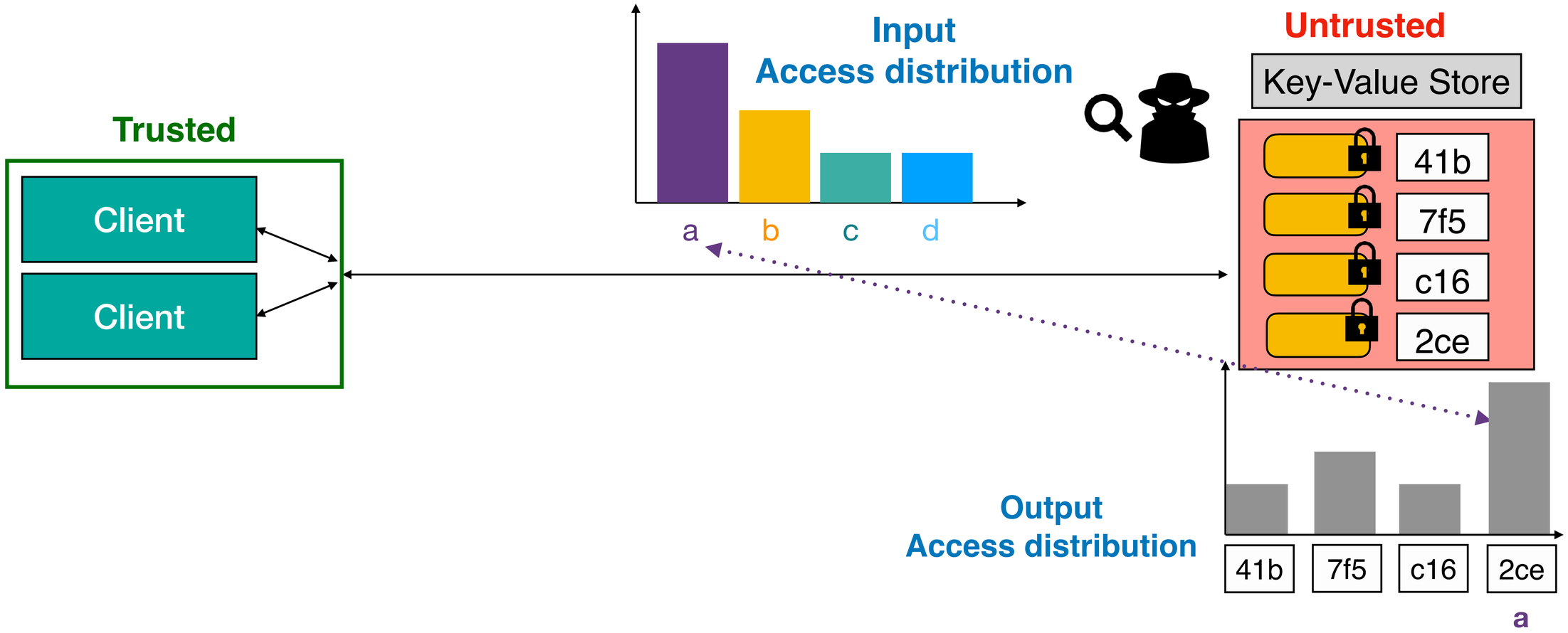
Oblivious Data Access

Oblivious Data Access

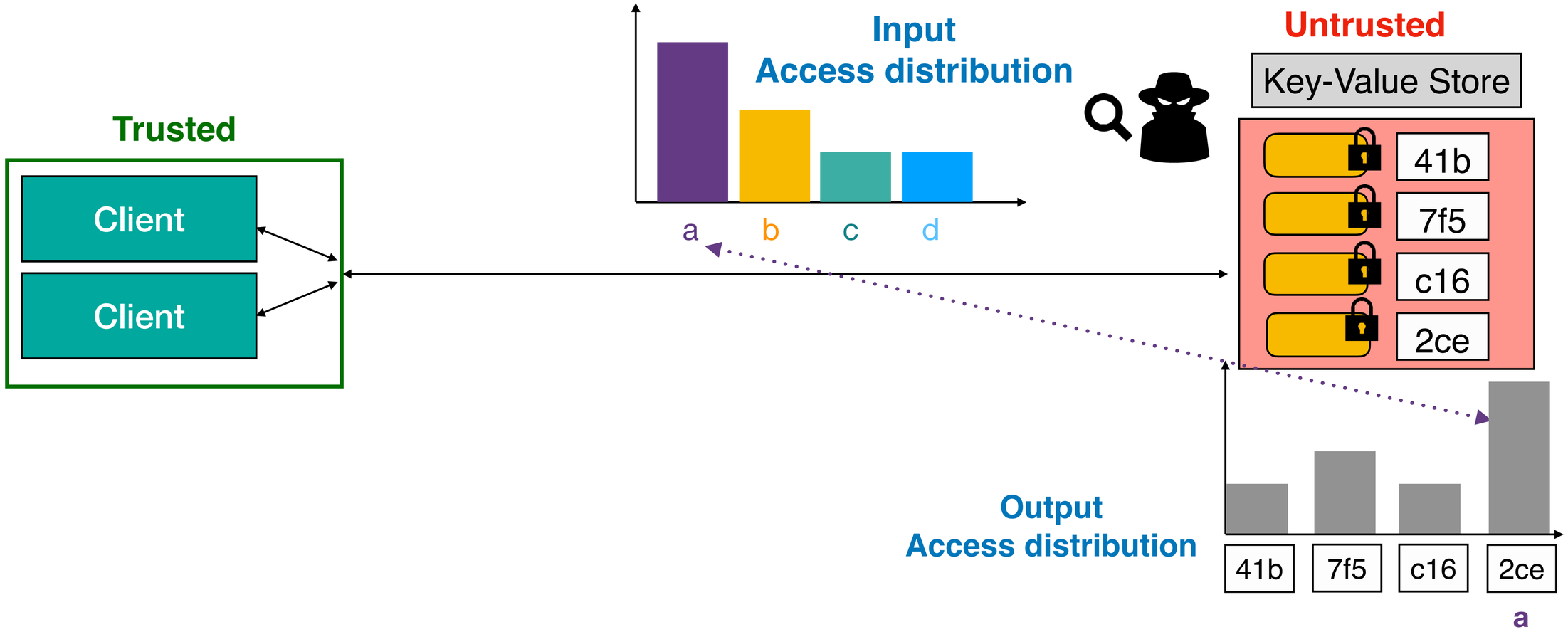
Oblivious Data Access

Oblivious Data Access

Oblivious Data Access

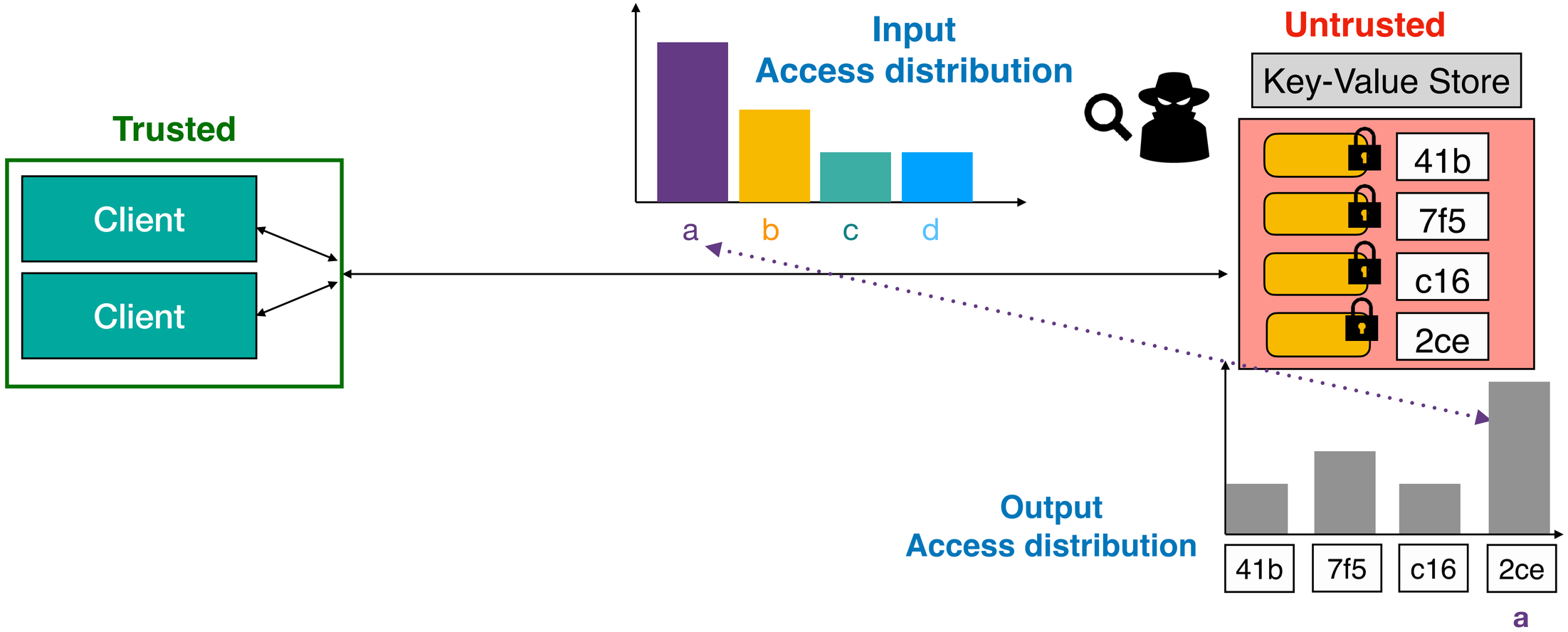


Oblivious Data Access



Encryption is not enough. Need to hide access patterns

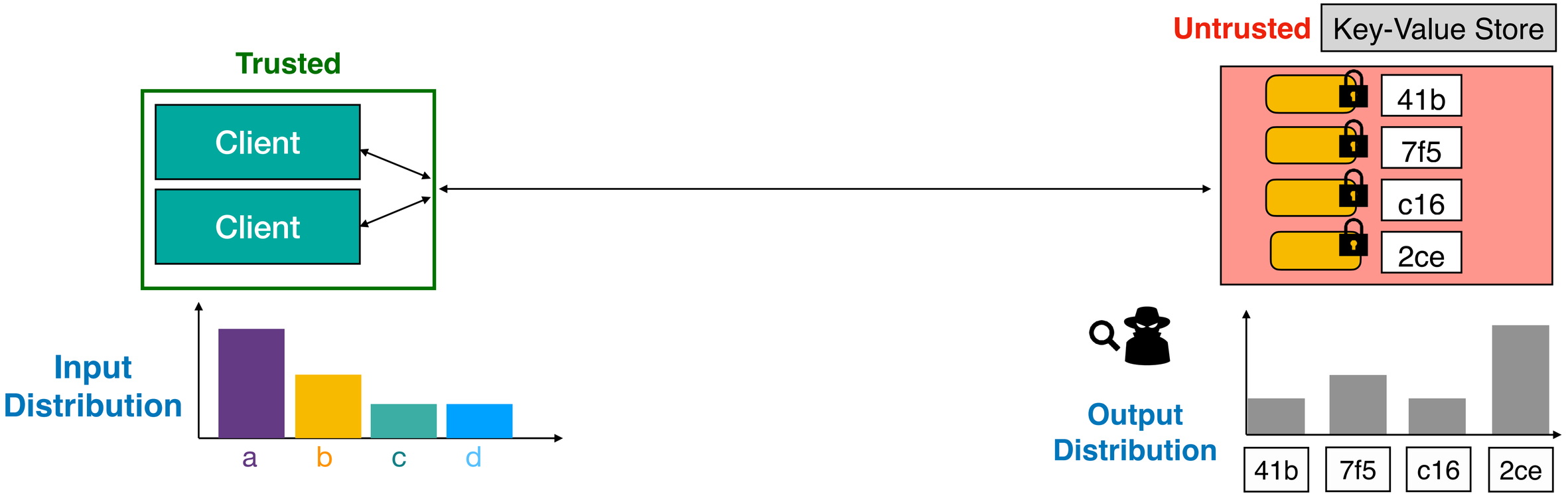
Oblivious Data Access



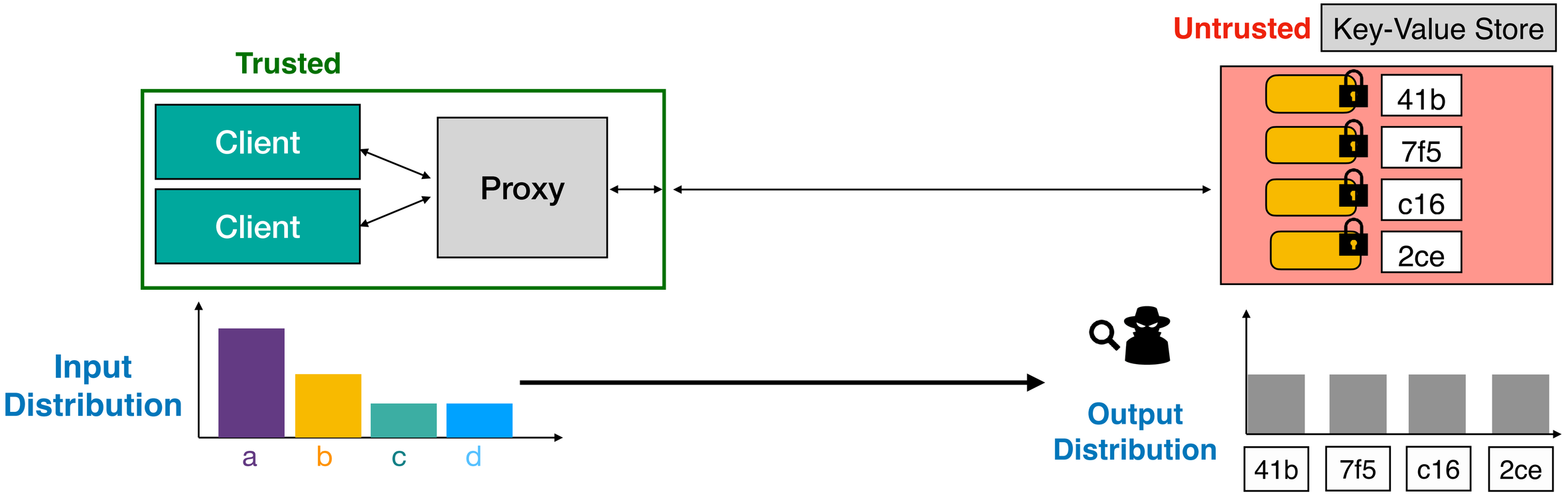
Encryption is not enough. Need to hide access patterns

Oblivious Data Access: Output distribution independent of input distribution

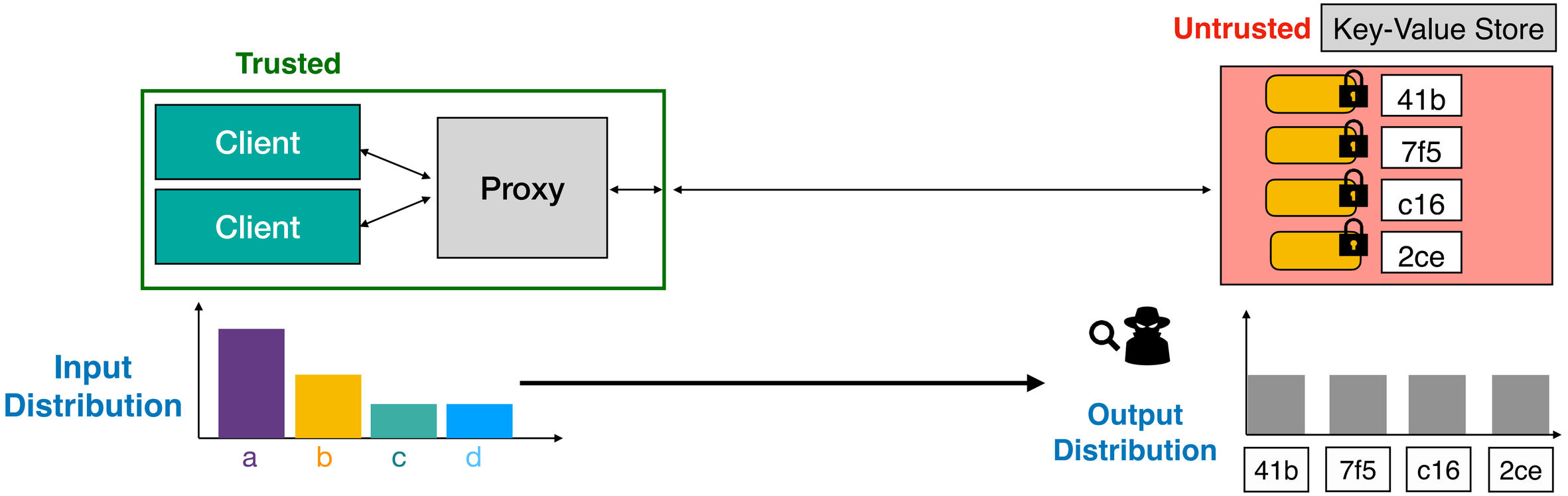
Existing Oblivious Data Access Techniques



Existing Oblivious Data Access Techniques

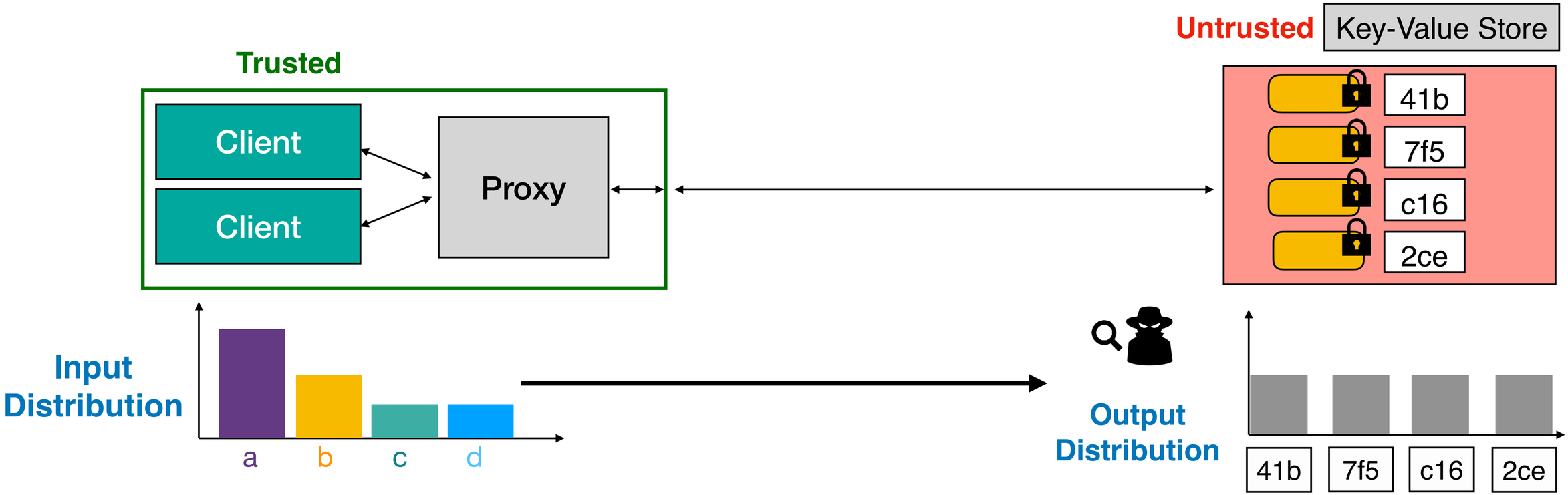


Existing Oblivious Data Access Techniques



Existing techniques “flatten” input access distribution into **uniform** output distribution

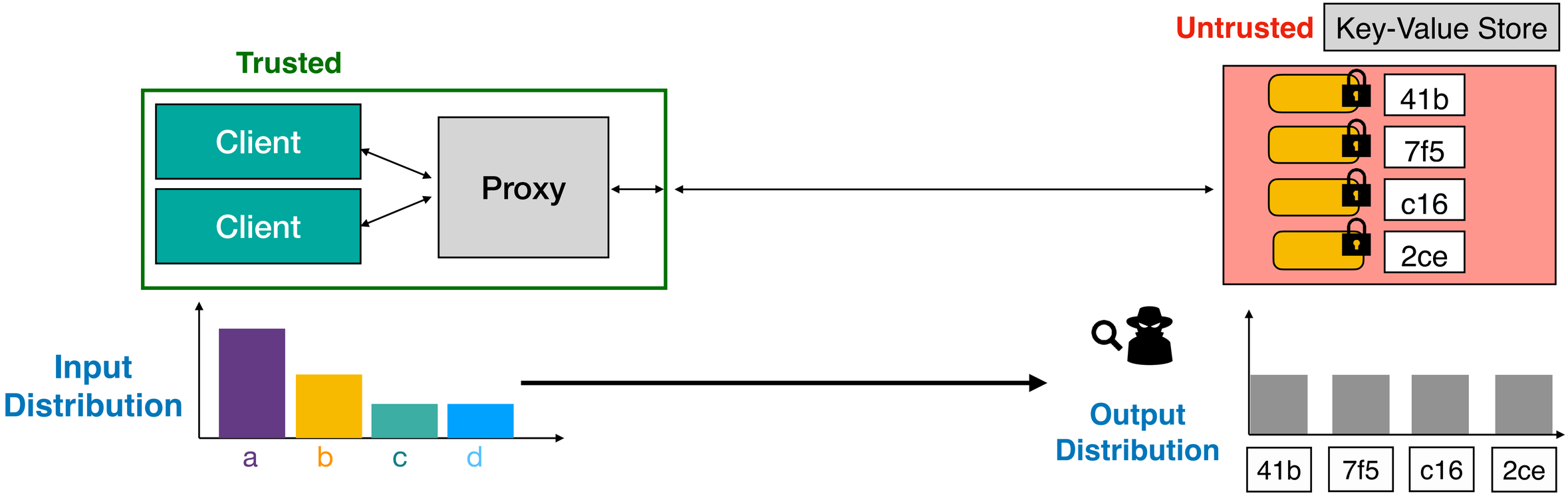
Existing Oblivious Data Access Techniques



Existing techniques “flatten” input access distribution into **uniform** output distribution

	ORAM
Bandwidth Overhead	Large O(log n)
Adversarial Model	Active (Can inject queries)

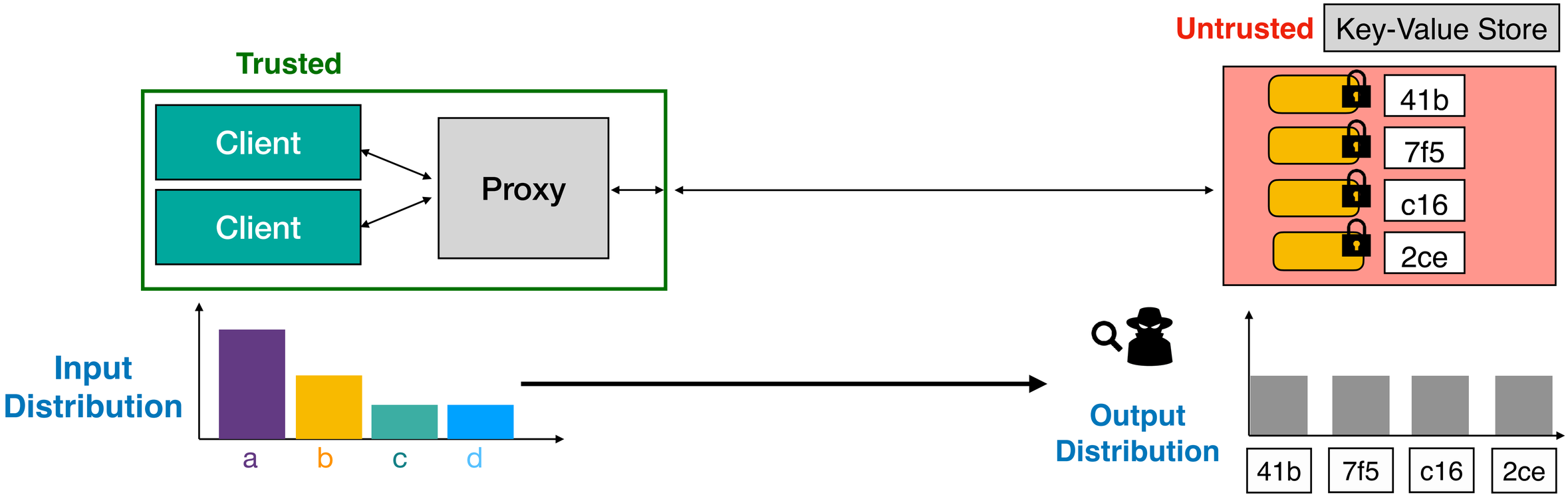
Existing Oblivious Data Access Techniques



Existing techniques “flatten” input access distribution into **uniform** output distribution

	ORAM	**Pancake**
Bandwidth Overhead	Large O(log n)	Constant (3x)
Adversarial Model	Active (Can inject queries)	Honest-but-curious (Does not inject queries)

Existing Oblivious Data Access Techniques



Existing techniques “flatten” input access distribution into **uniform** output distribution

	ORAM	Pancake
Bandwidth Overhead	Large O(log n)	Constant (3x)
Adversarial Model	Active (Can inject queries)	Honest-but-curious (Does not inject queries)

All existing oblivious data access techniques use centralized (stateful) proxy

This work

This work

Challenges with Centralized Oblivious Data Access Systems

1. Insecure or unavailable during failures
2. Scalability bottleneck

This work

Challenges with Centralized Oblivious Data Access Systems
1. Insecure or unavailable during failures
2. Scalability bottleneck

Design of Shortstack:
Distributed, Fault-tolerant, Oblivious Data Access System

This work

Challenges with Centralized Oblivious Data Access Systems
1. Insecure or unavailable during failures
2. Scalability bottleneck

Design of Shortstack:
Distributed, Fault-tolerant, Oblivious Data Access System

Security model:
Enables formal study of Distributed, Fault-tolerant, Oblivious Data Access systems

This work

Challenges with Centralized Oblivious Data Access Systems
1. Insecure or unavailable during failures
2. Scalability bottleneck

Design of Shortstack:
Distributed, Fault-tolerant, Oblivious Data Access System

Security model:
Enables formal study of Distributed, Fault-tolerant, Oblivious Data Access systems

Pancake overview

Pancake overview

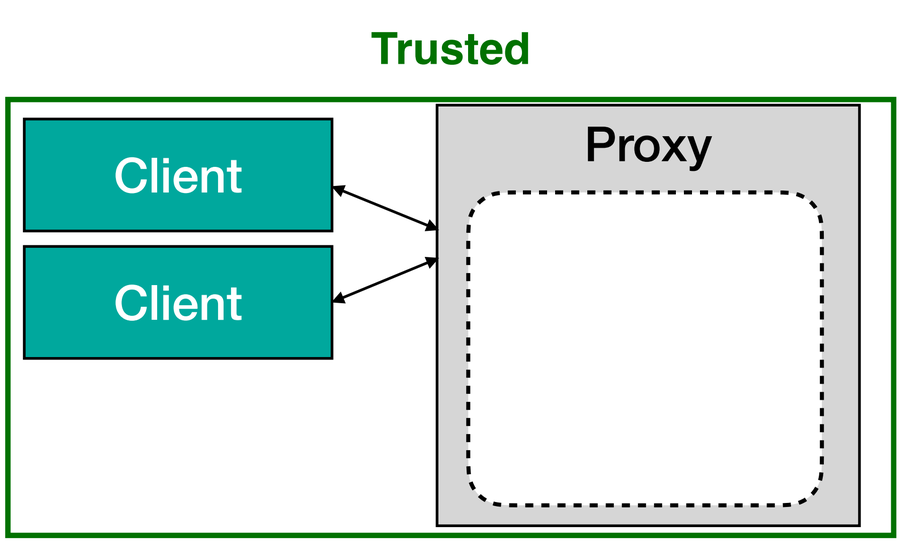
Three important functionalities:

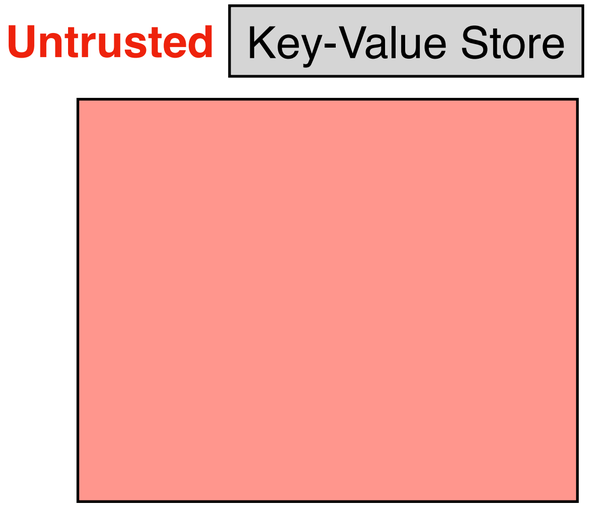
- **Replica Creation**
- **Query Generation**
- **Query Execution (+temporarily buffer writes to replicas)**

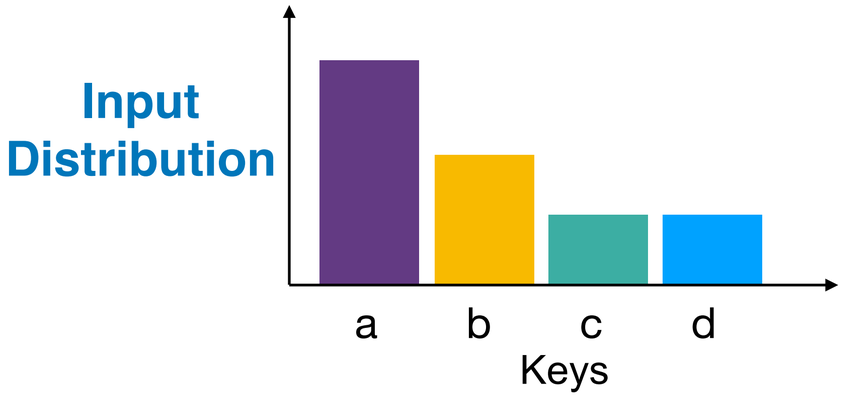
Pancake overview

Three important functionalities:

- **Replica Creation**
- **Query Generation**
- **Query Execution (+temporarily buffer writes to replicas)**



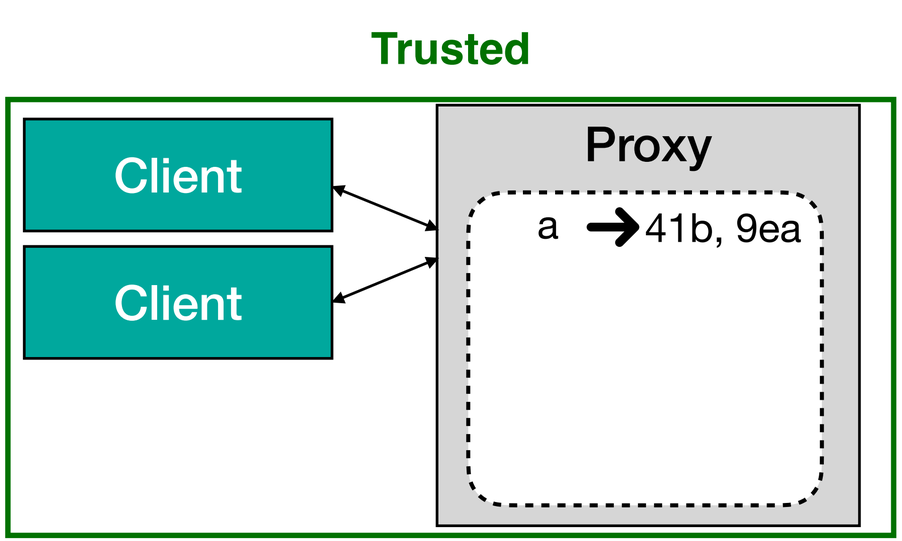


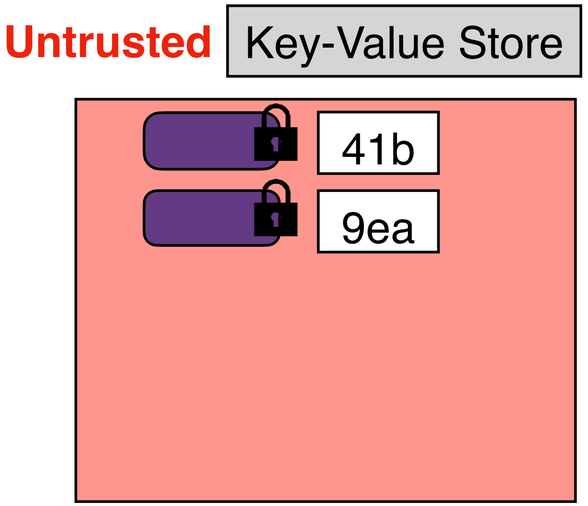


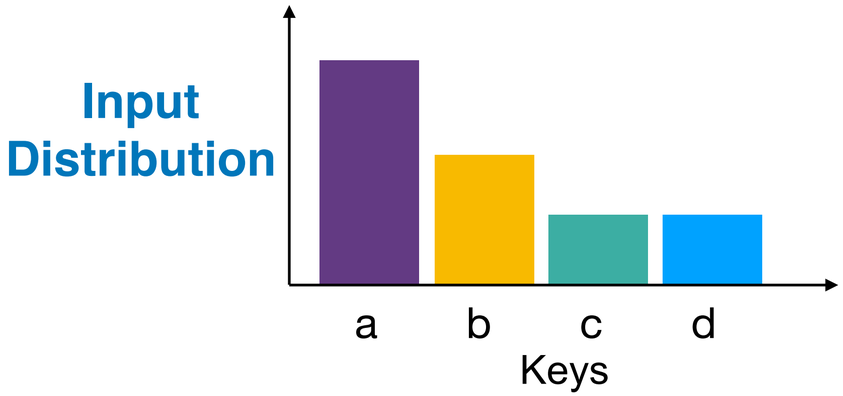
Pancake overview

Three important functionalities:

- **Replica Creation**
- **Query Generation**
- **Query Execution (+temporarily buffer writes to replicas)**



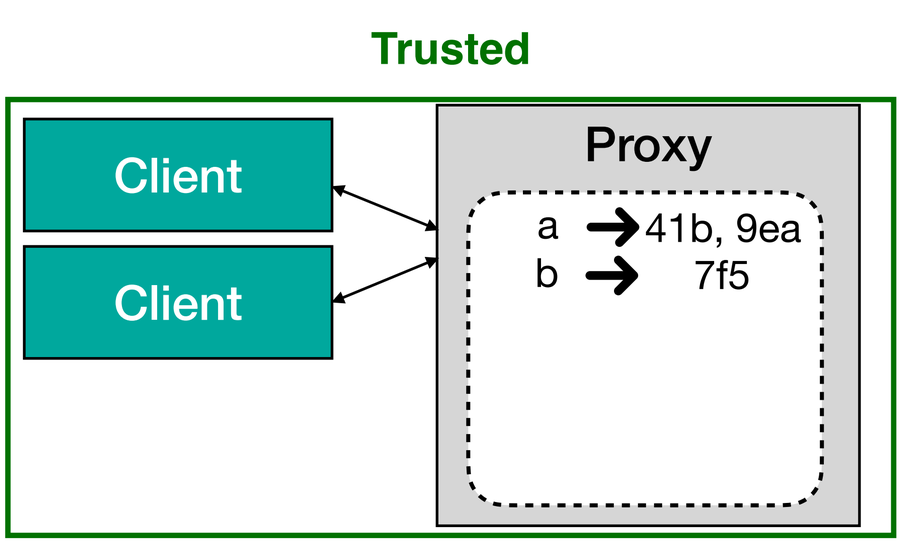


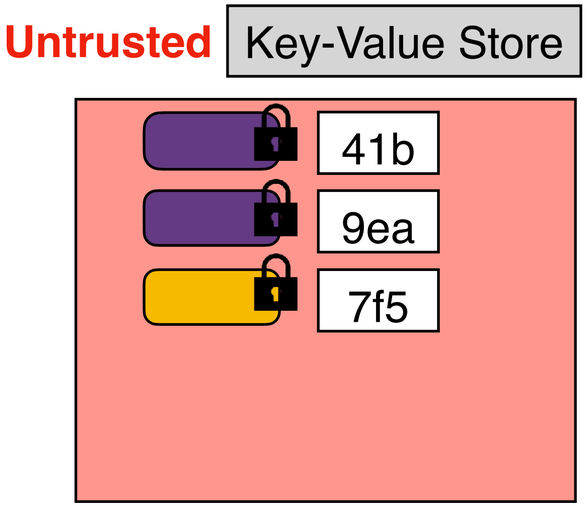


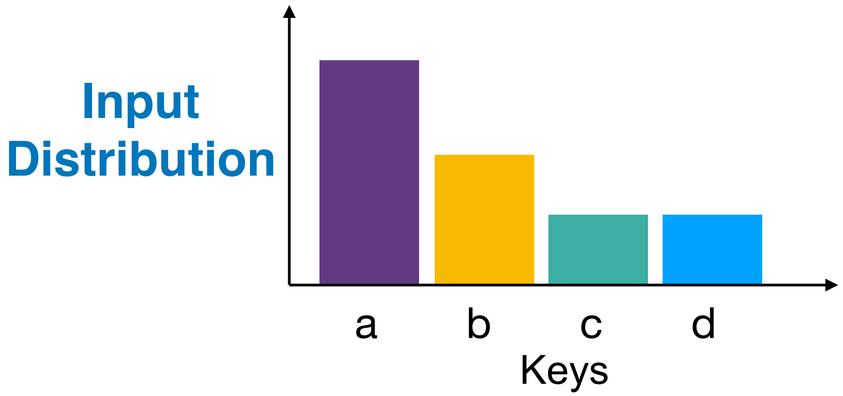
Pancake overview

Three important functionalities:

- **Replica Creation**
- **Query Generation**
- **Query Execution (+temporarily buffer writes to replicas)**



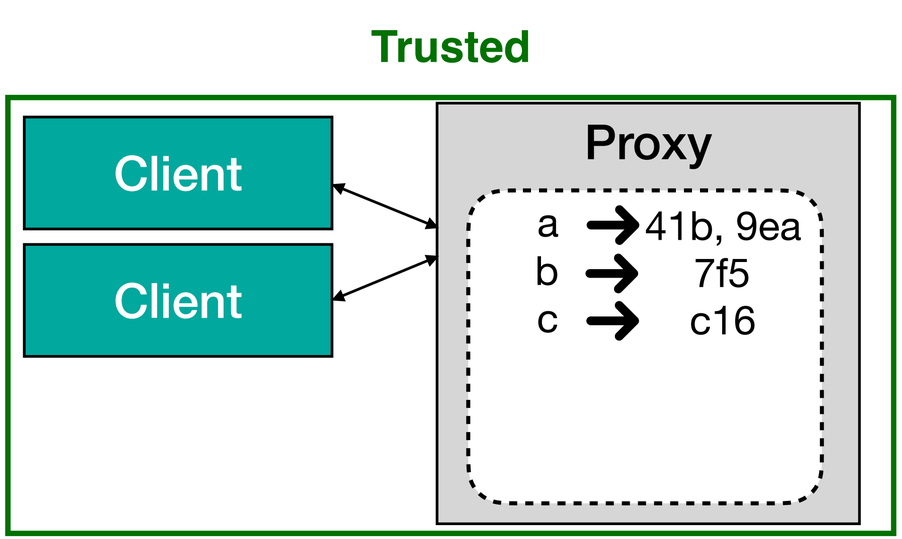


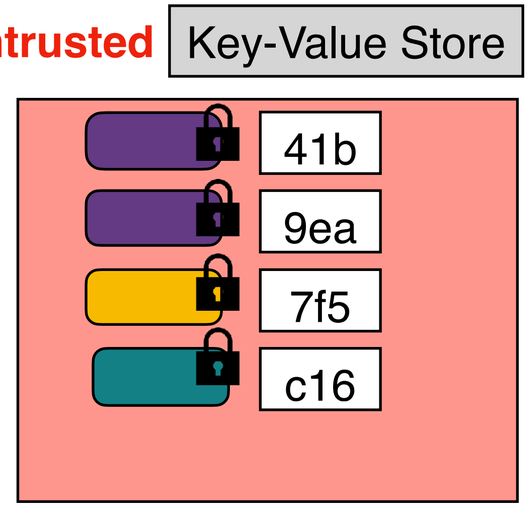


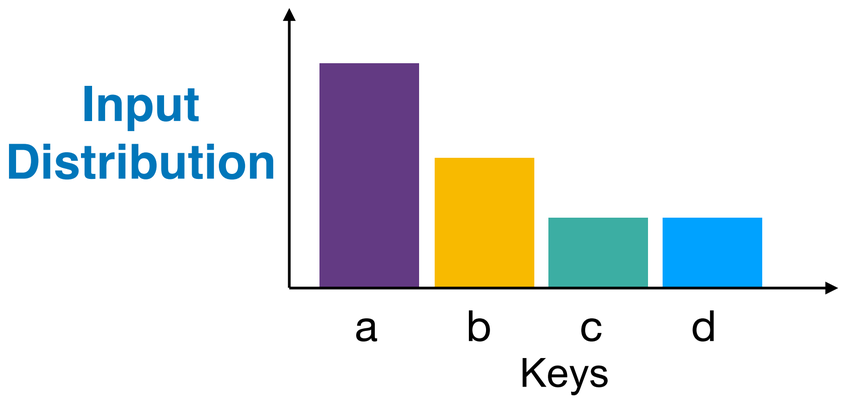
Pancake overview

Three important functionalities:

- **Replica Creation**
- **Query Generation**
- **Query Execution (+temporarily buffer writes to replicas)**



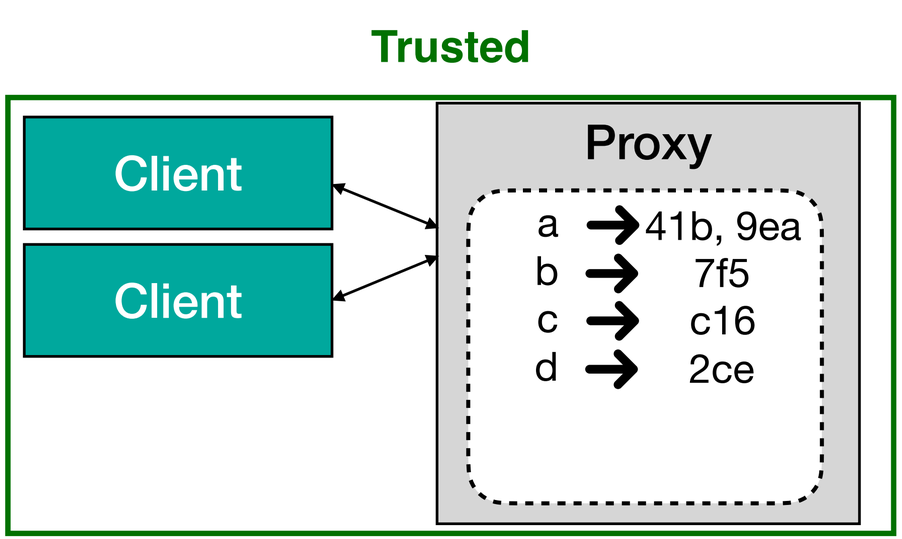


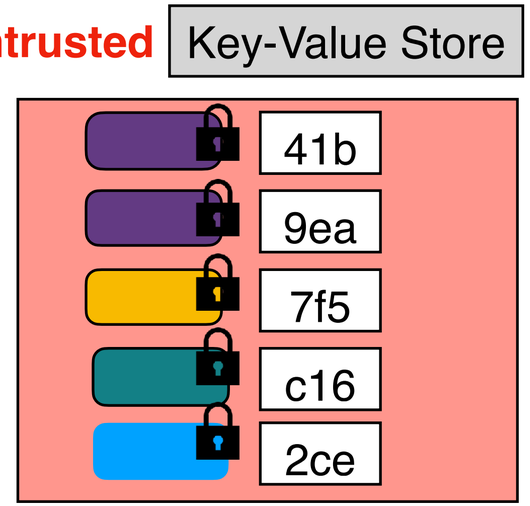


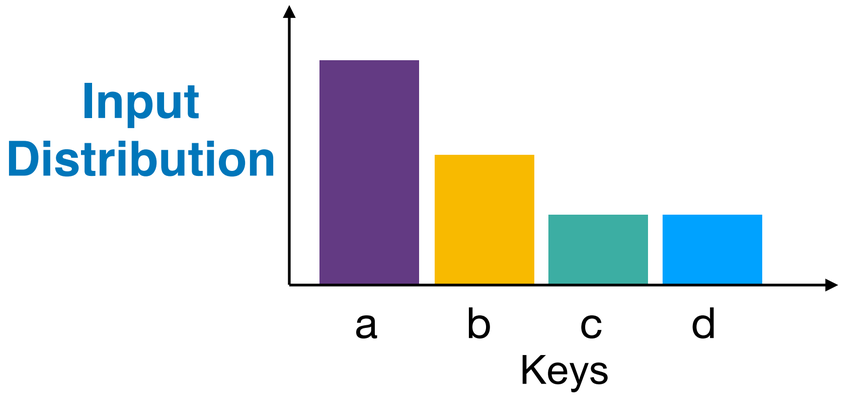
Pancake overview

Three important functionalities:

- **Replica Creation**
- Query Generation
- Query Execution (+temporarily buffer writes to replicas)



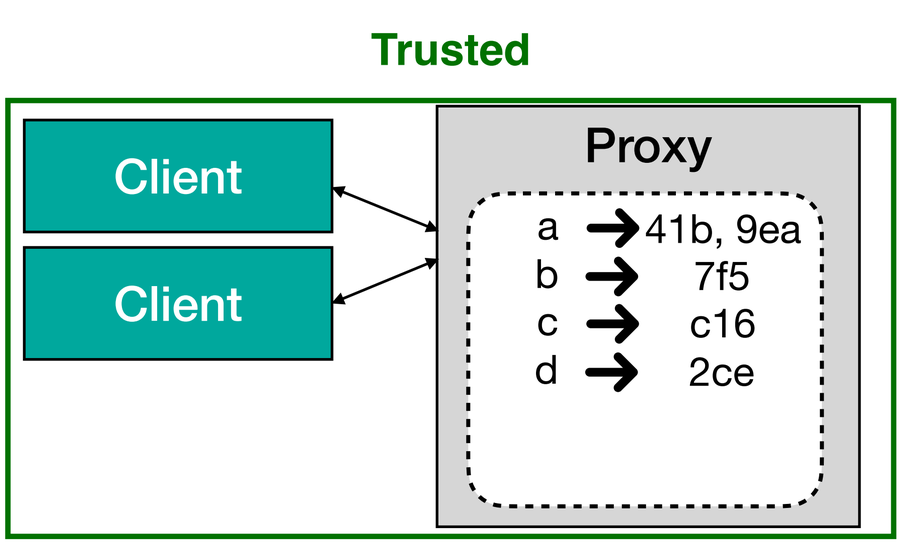


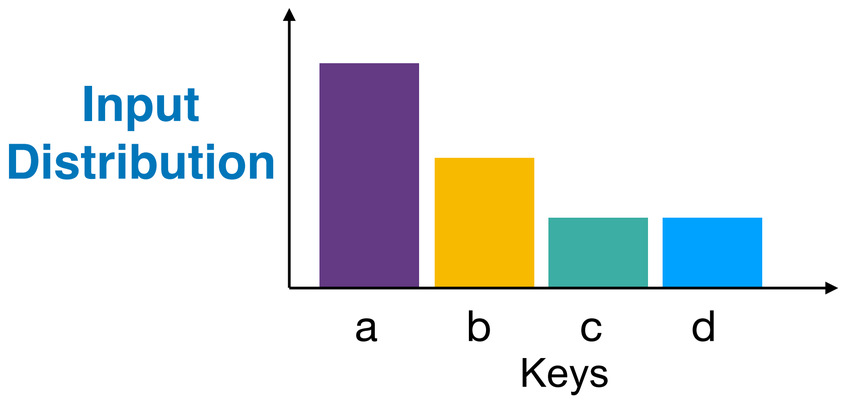


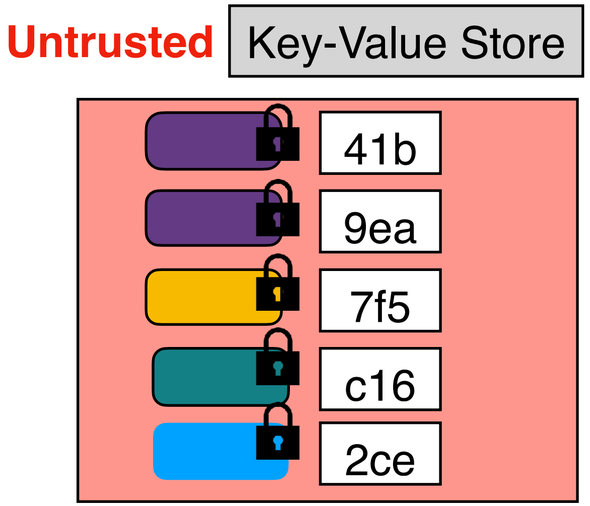
Pancake overview

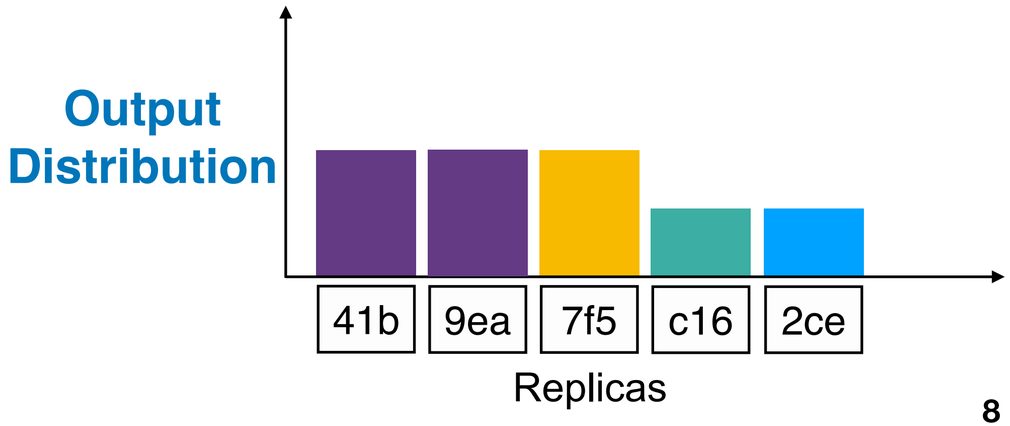
Three important functionalities:

- **Replica Creation**
- **Query Generation**
- **Query Execution (+temporarily buffer writes to replicas)**





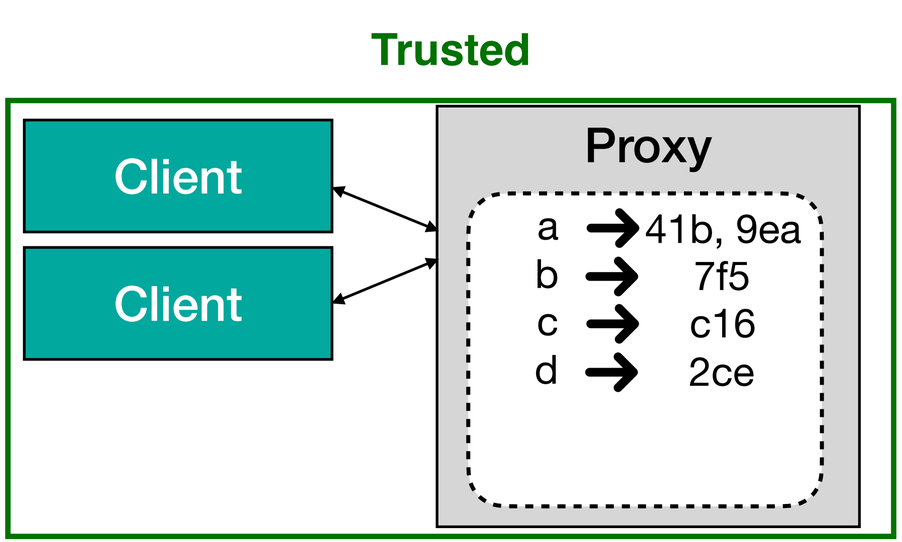


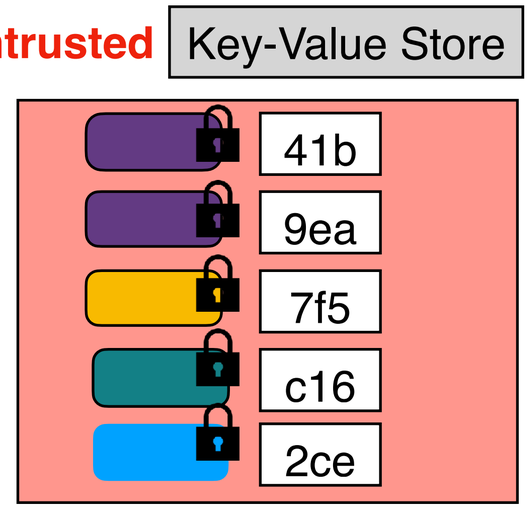


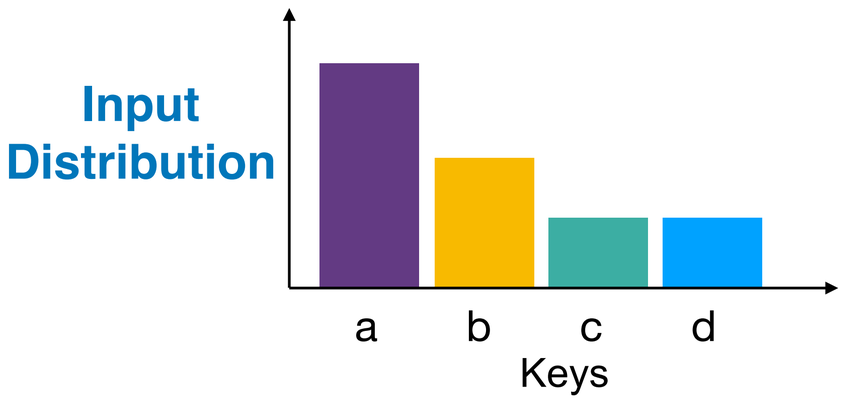
Pancake overview

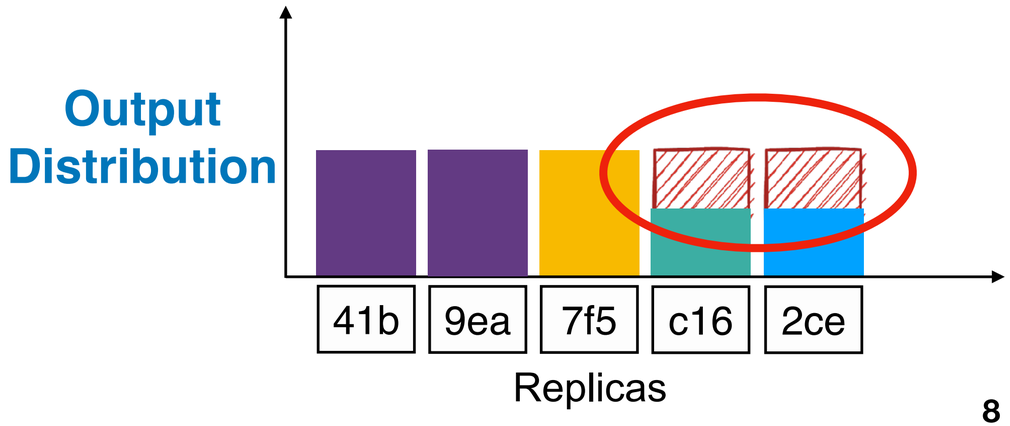
Three important functionalities:

- Replica Creation
- **Query Generation**
- Query Execution (+temporarily buffer writes to replicas)





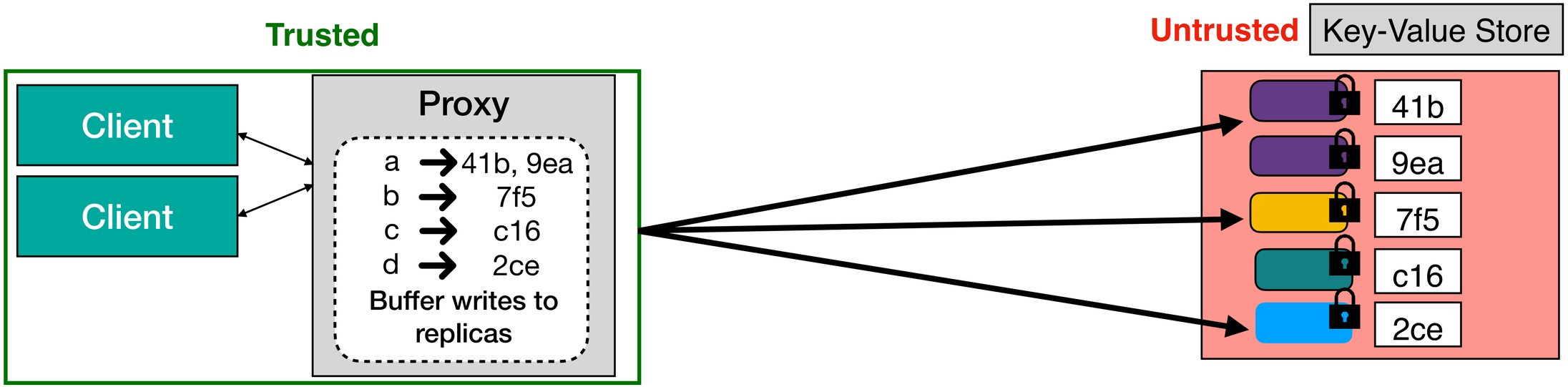


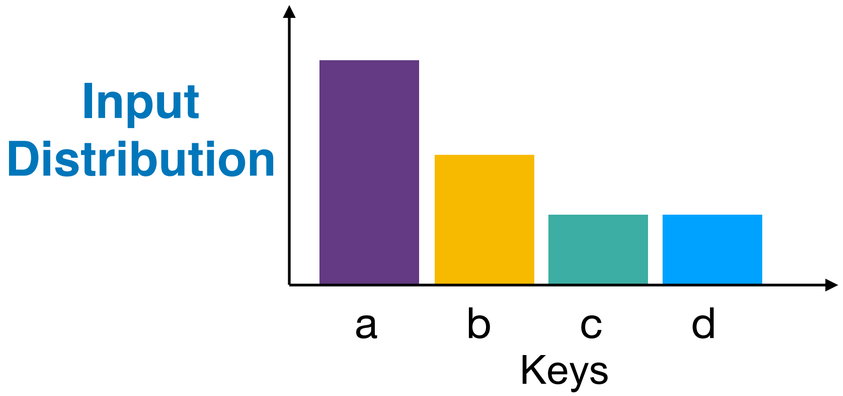


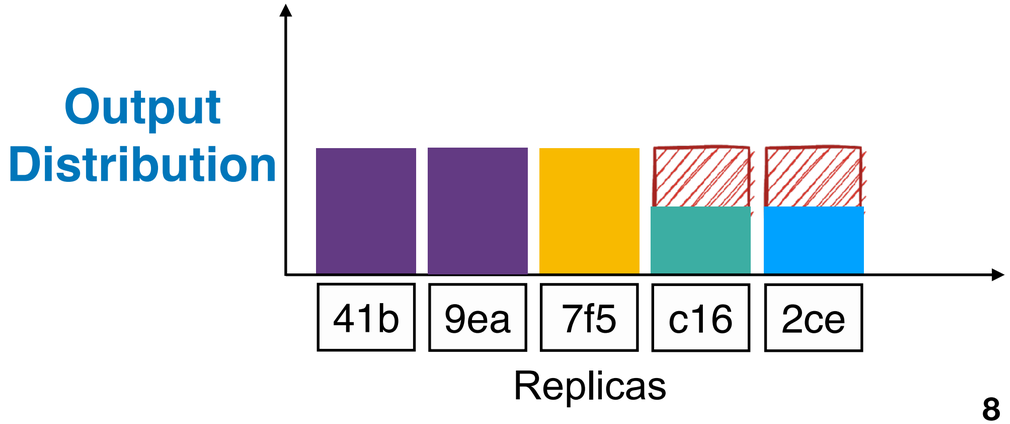
Pancake overview

Three important functionalities:

- Replica Creation
- Query Generation
- **Query Execution (+temporarily buffer writes to replicas)**



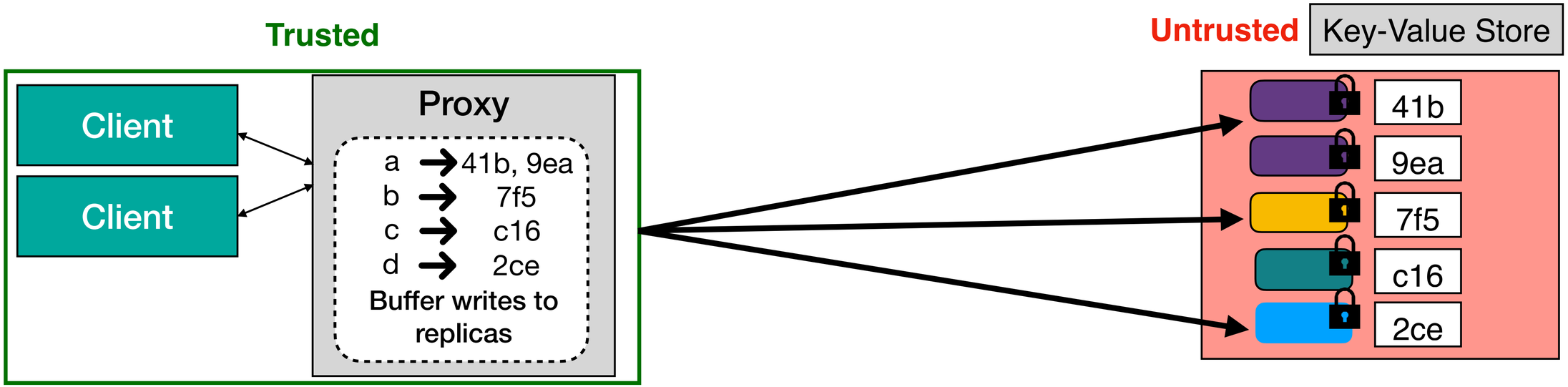


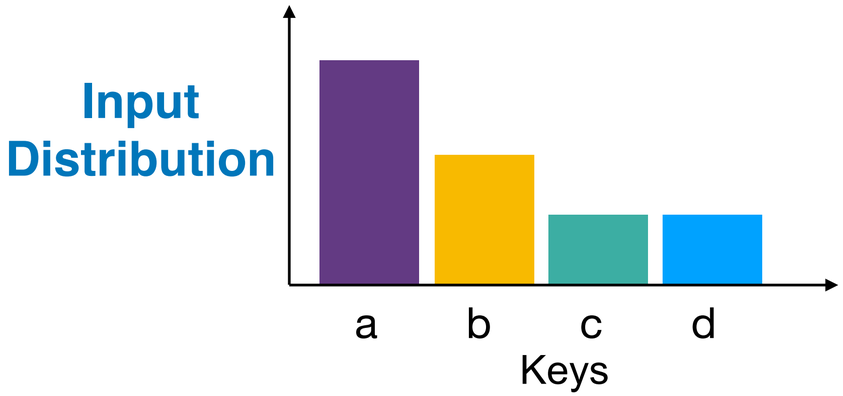


Pancake overview

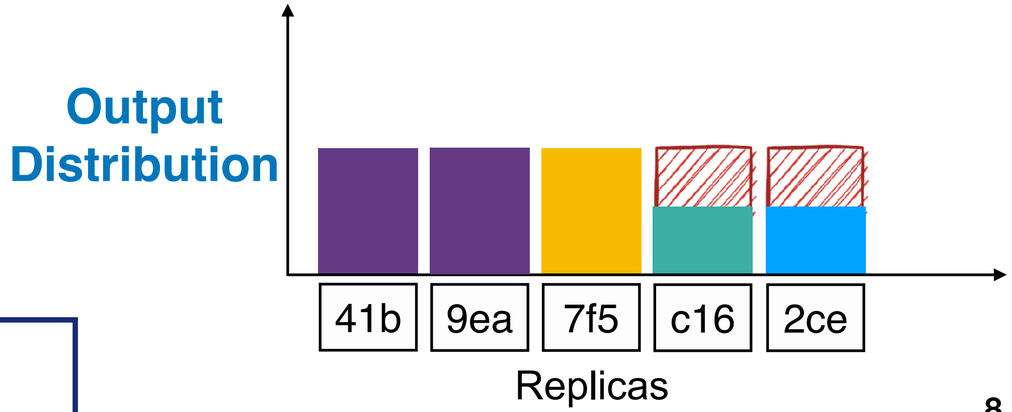
Three important functionalities:

- Replica Creation
- Query Generation
- **Query Execution (+temporarily buffer writes to replicas)**



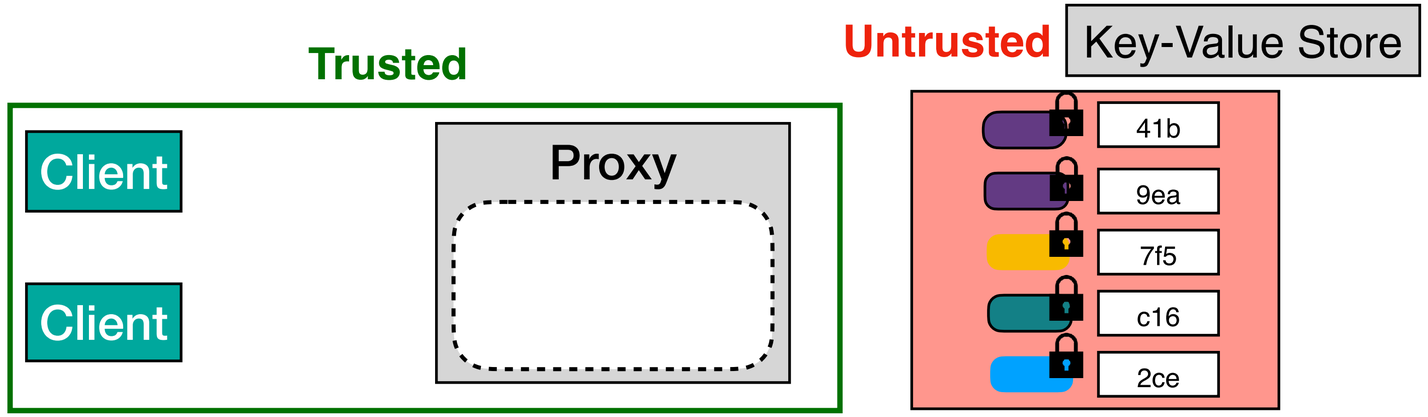


Proxy is stateful

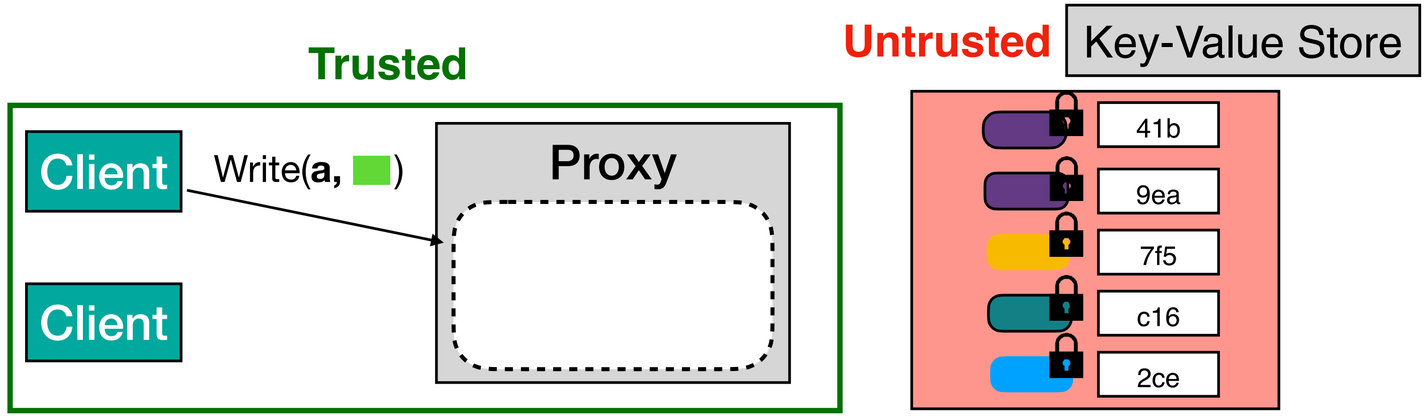


Challenges with Centralized and Stateful Proxy

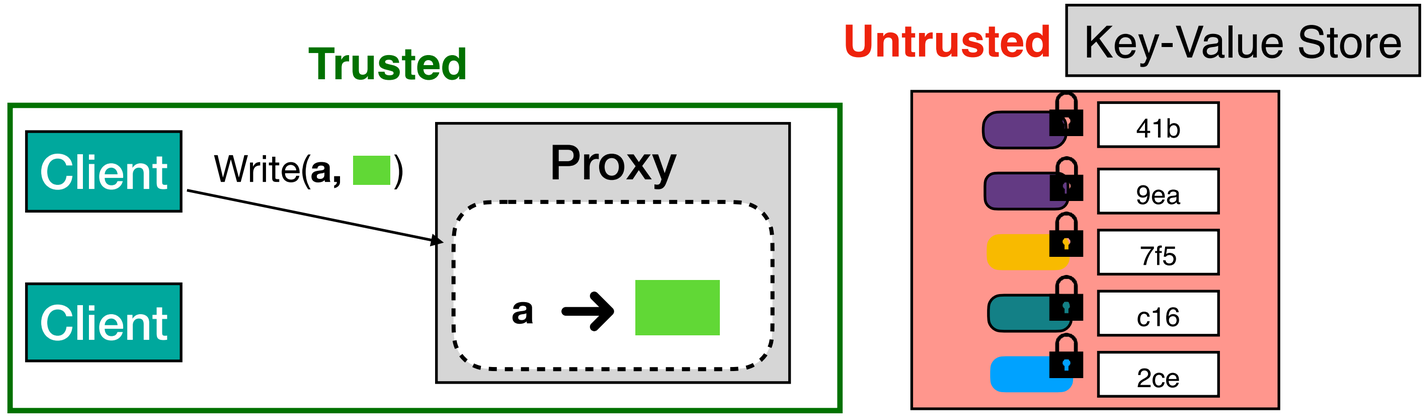
Challenges with Centralized and Stateful Proxy



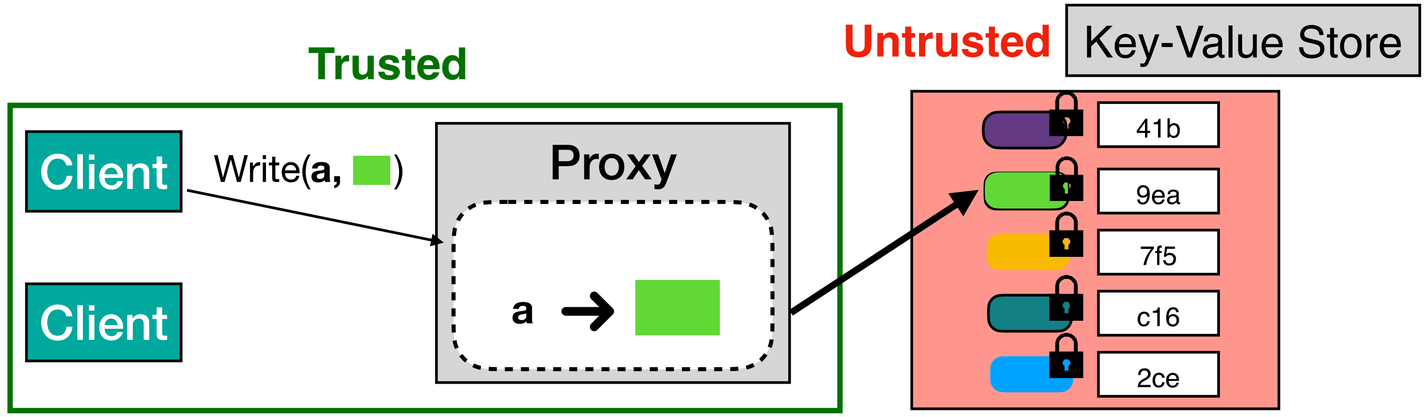
Challenges with Centralized and Stateful Proxy



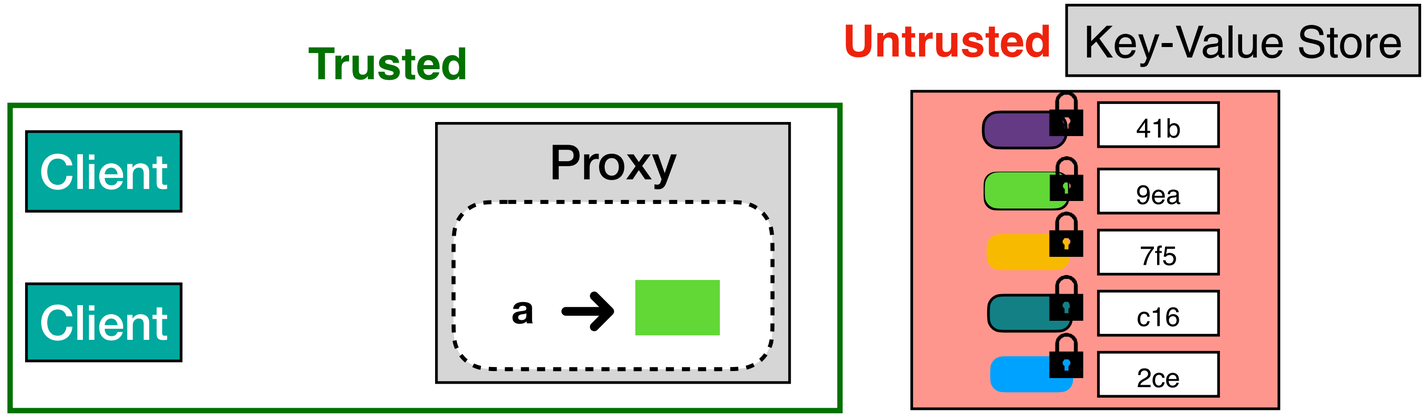
Challenges with Centralized and Stateful Proxy



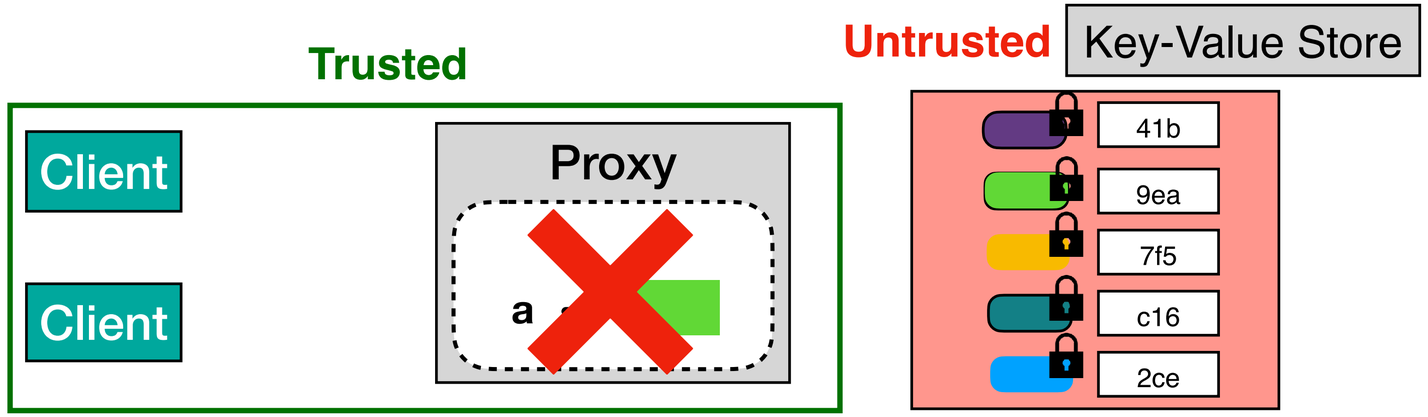
Challenges with Centralized and Stateful Proxy



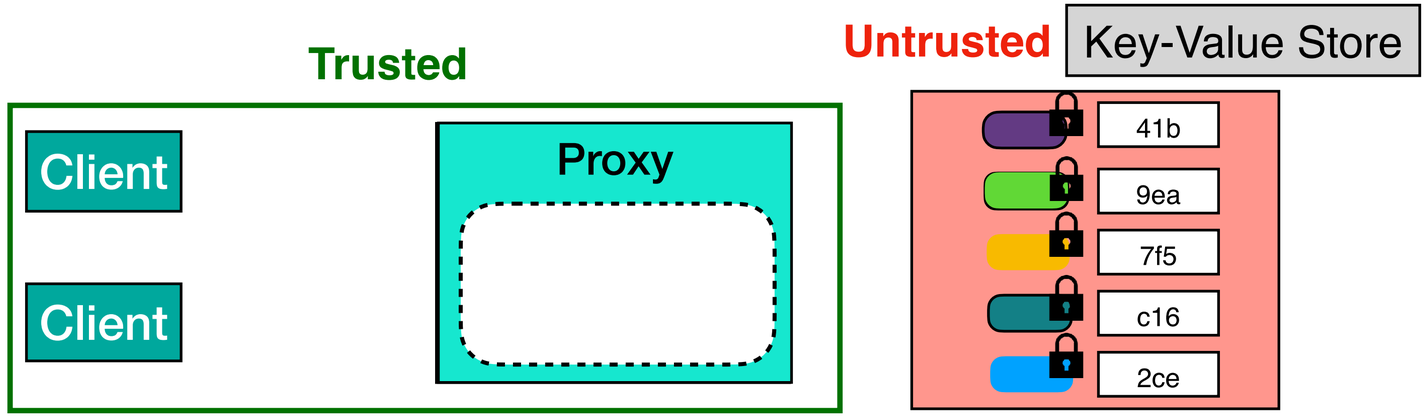
Challenges with Centralized and Stateful Proxy



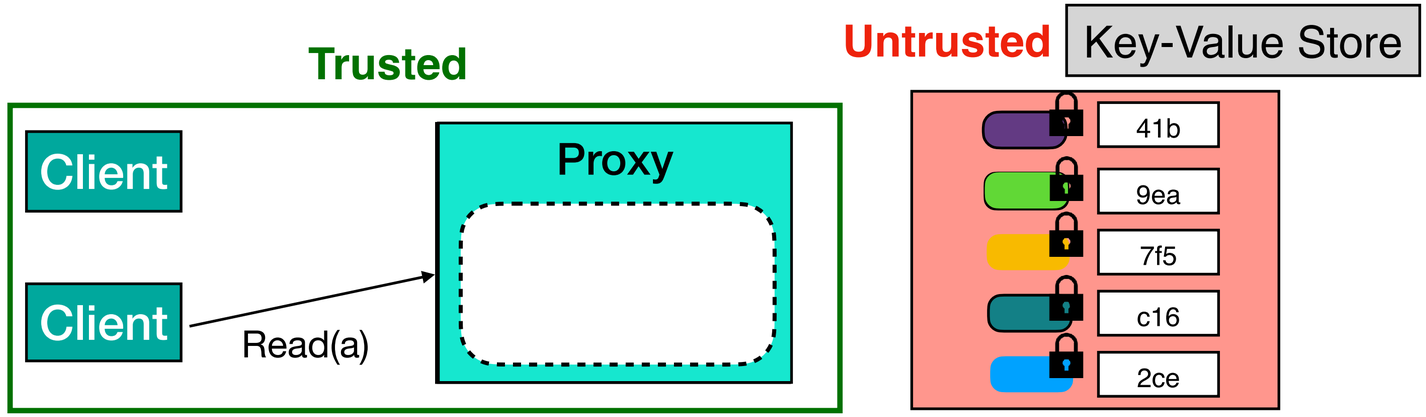
Challenges with Centralized and Stateful Proxy



Challenges with Centralized and Stateful Proxy

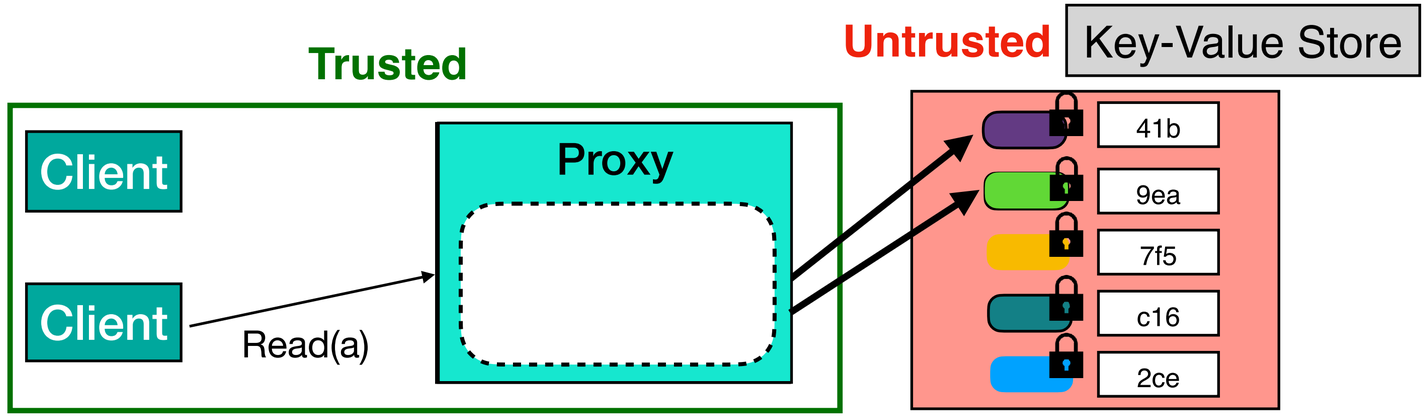


Challenges with Centralized and Stateful Proxy



Inconsistency across replicas

Challenges with Centralized and Stateful Proxy



Inconsistency across replicas

- Fetch all the replicas of **a** to determine the latest value

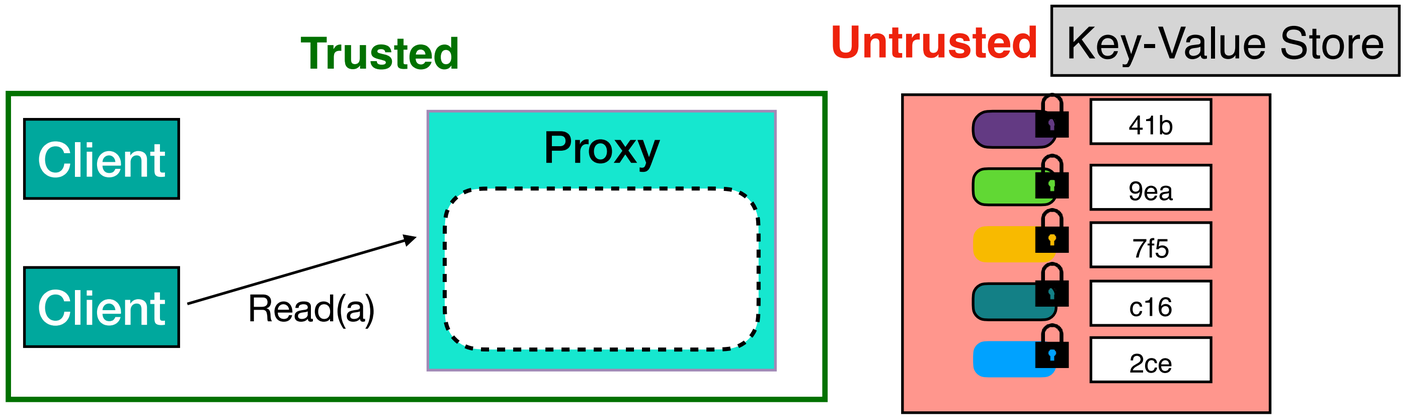
Violates Security
(correlated accesses)

Challenges with Centralized and Stateful Proxy

Inconsistency across replicas

- Fetch all the replicas of **a** to determine the latest value

Violates Security
(correlated accesses)

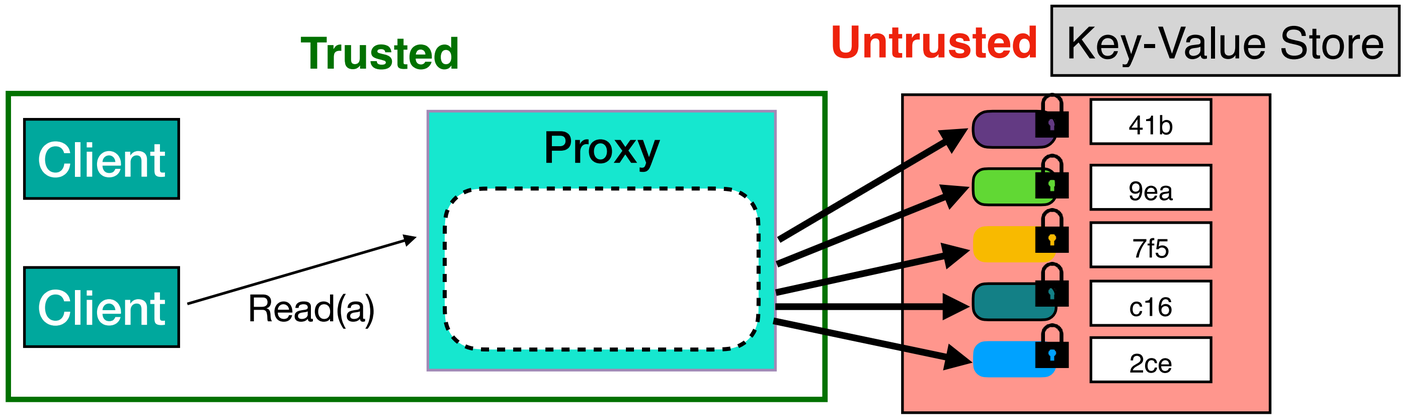


Challenges with Centralized and Stateful Proxy

Inconsistency across replicas

- Fetch all the replicas of **a** to determine the latest value

Violates Security
(correlated accesses)



- Fetch the entire database to avoid security violations

Huge bandwidth overhead
(large periods of unavailability)

Challenges with Centralized and Stateful Proxy

Inconsistency across replicas

- Fetch all the replicas of **a** to determine the latest value

Violates Security
(correlated accesses)

- Fetch the entire database to avoid security violations

Huge bandwidth overhead
(large periods of unavailability)

Centralized, Stateful, Proxy:
Security violation or long periods of unavailability

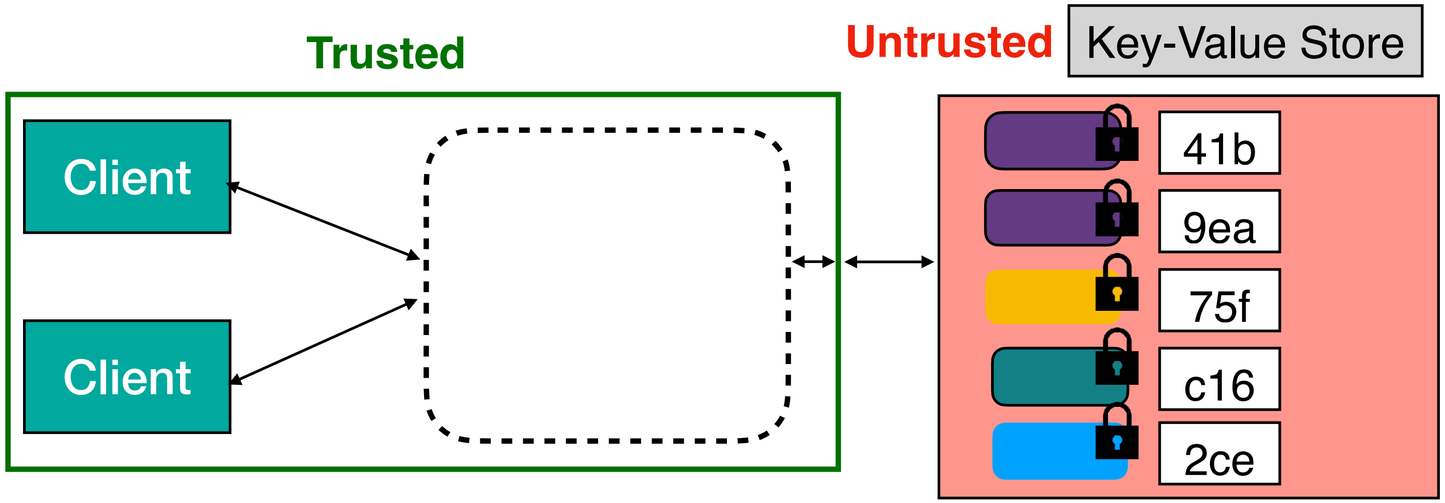
This work

Challenges with Centralized Oblivious Data Access Systems
1. Insecure or unavailable during failures
2. Scalability bottleneck

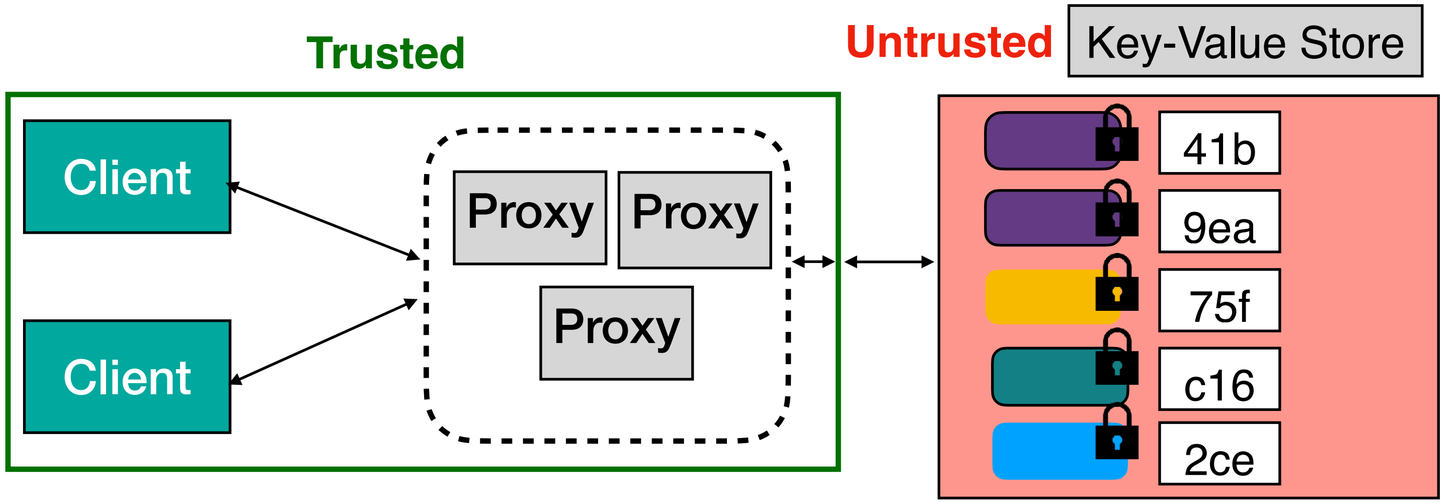
Design of Shortstack:
Distributed, Fault-tolerant, Oblivious Data Access System

Security model:
Enables formal study of Distributed, Fault-tolerant, Oblivious Data Access systems

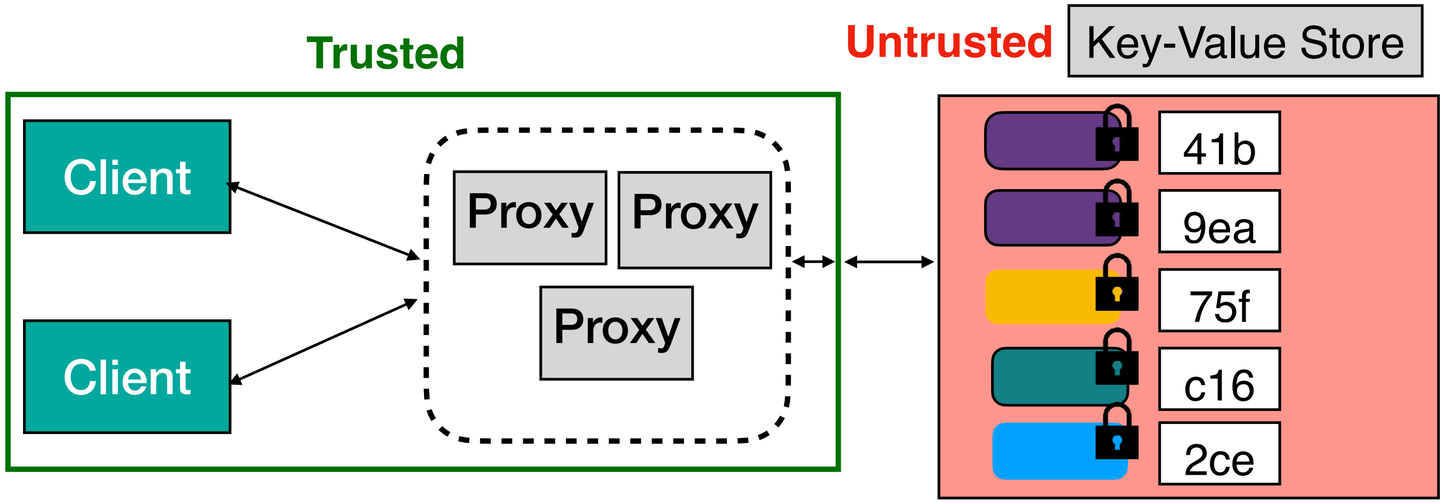
Shortstack Summary



Shortstack Summary

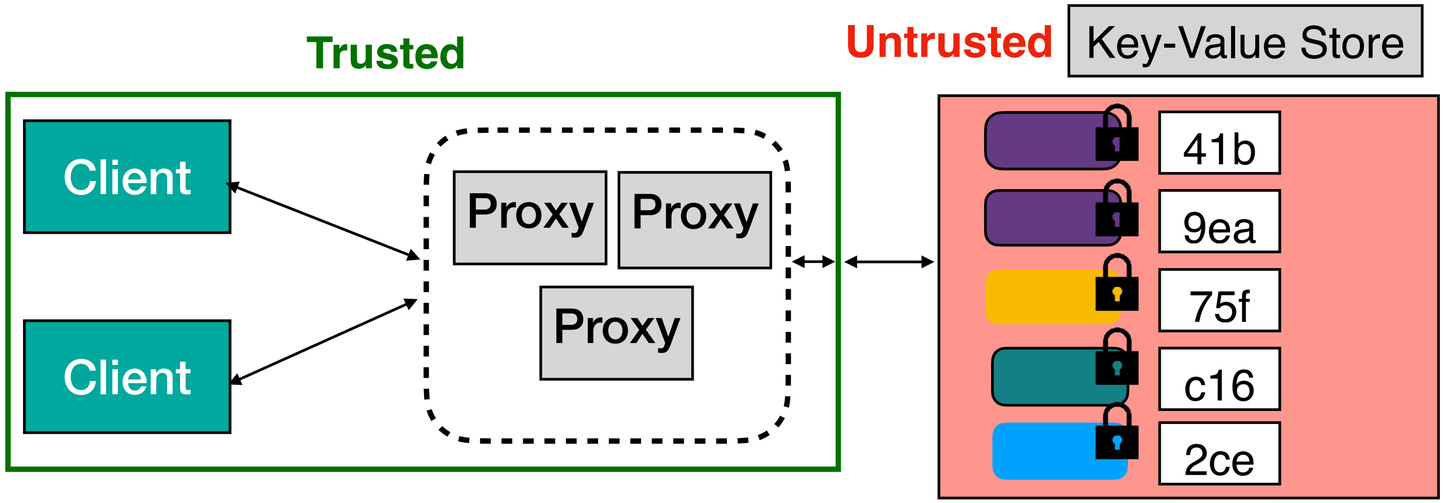


Shortstack Summary



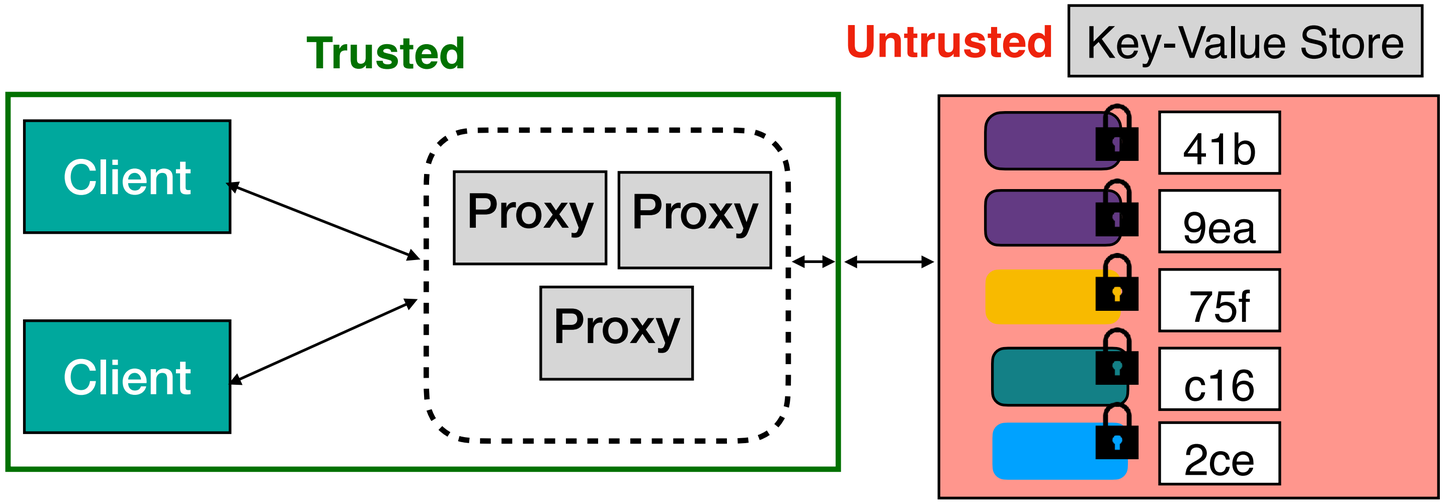
1. **Oblivious data access guarantees, even under failures**
   - Fail-stop failure model
   - Worst-case scenario : Arbitrary (bounded number of) failures at arbitrary times

Shortstack Summary



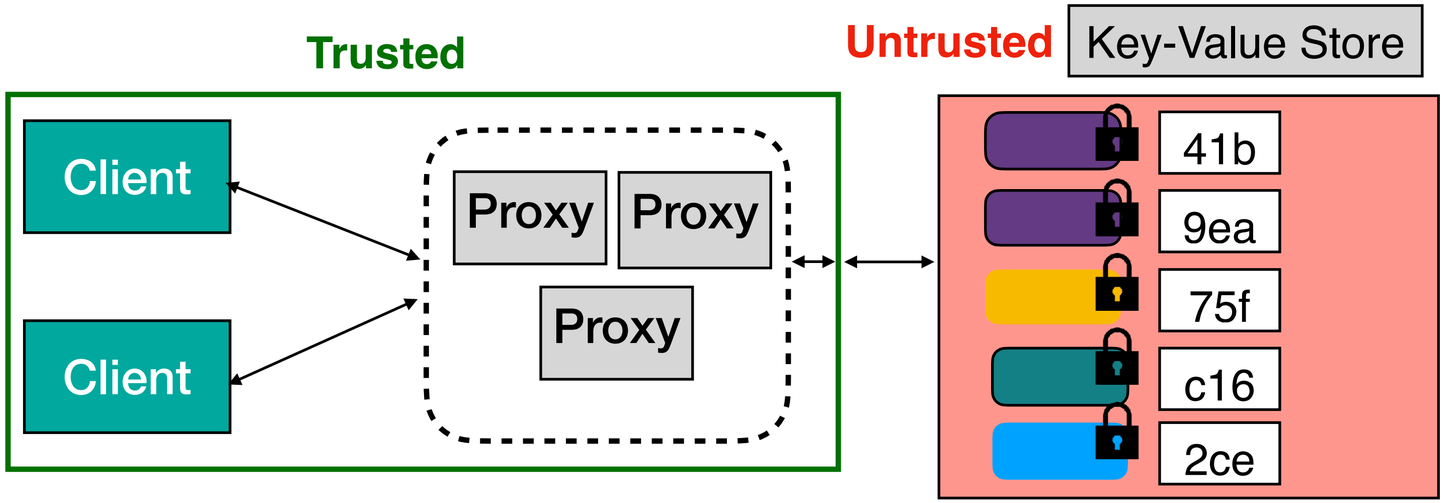
1. **Oblivious data access guarantees, even under failures**
   - Fail-stop failure model
   - Worst-case scenario : Arbitrary (bounded number of) failures at arbitrary times
2. **System Availability**

Shortstack Summary



1. **Oblivious data access guarantees, even under failures**
   - Fail-stop failure model
   - Worst-case scenario : Arbitrary (bounded number of) failures at arbitrary times
2. **System Availability**
3. **Scalability:** Alleviate bandwidth & compute bottlenecks, throughput linear in #physical-servers

Shortstack Summary



1. **Oblivious data access guarantees, even under failures**
   - Fail-stop failure model
   - Worst-case scenario : Arbitrary (bounded number of) failures at arbitrary times
2. **System Availability**
3. **Scalability:** Alleviate bandwidth & compute bottlenecks, throughput linear in #physical-servers

Threat Model: Honest-but-curious Adversary (Or, Passive persistent adversary—The Pancake model)

Shortstack Key Insight

Shortstack Key Insight

Obliviousness requires output distribution to be independent of input distribution

Shortstack Key Insight

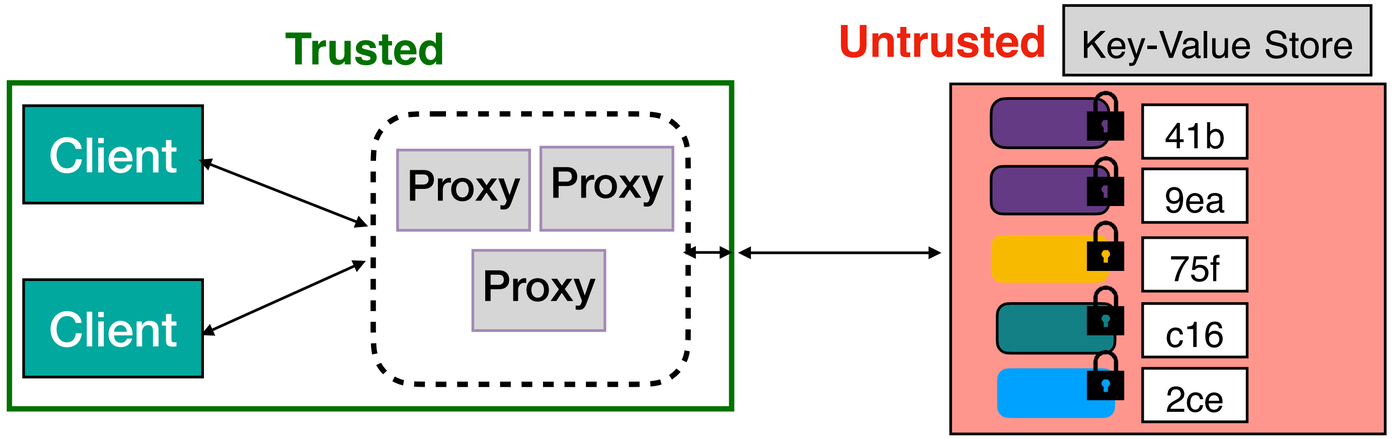
Obliviousness requires output distribution to be independent of input distribution

Uniform distribution is one but not the only way to achieve independence

Shortstack Key Insight

Obliviousness requires output distribution to be independent of input distribution

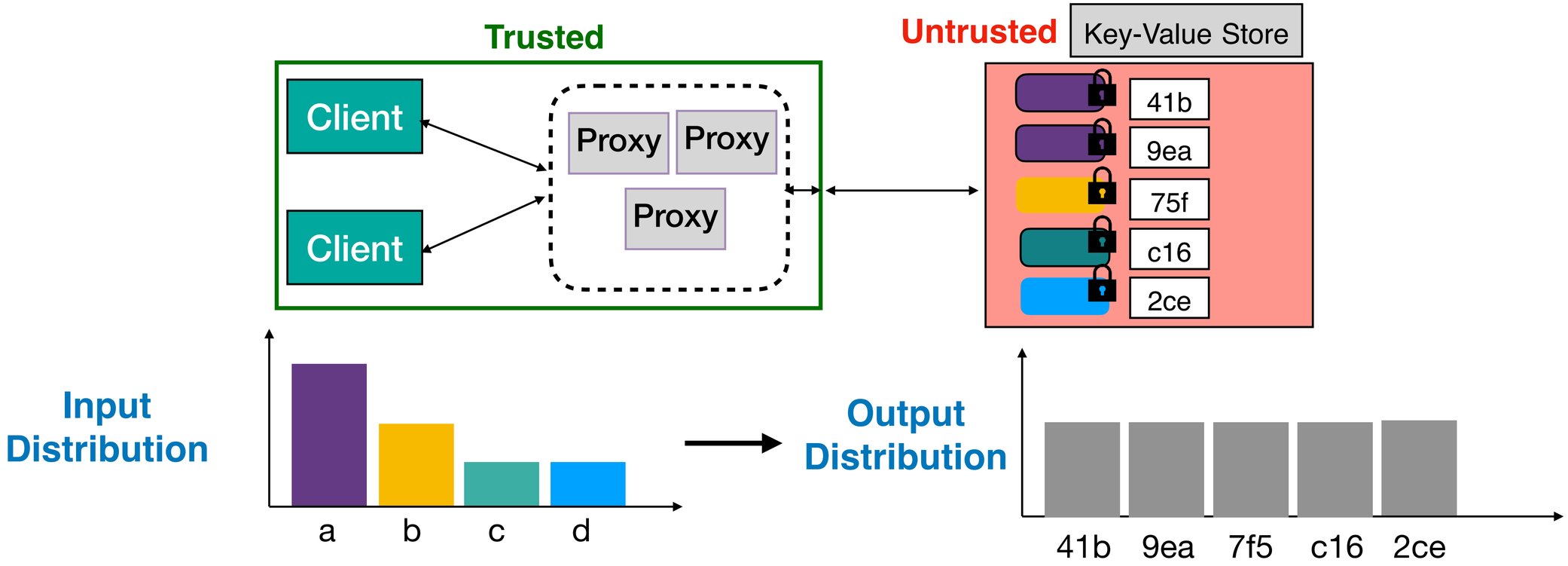
Uniform distribution is one but not the only way to achieve independence



Shortstack Key Insight

Obliviousness requires output distribution to be independent of input distribution

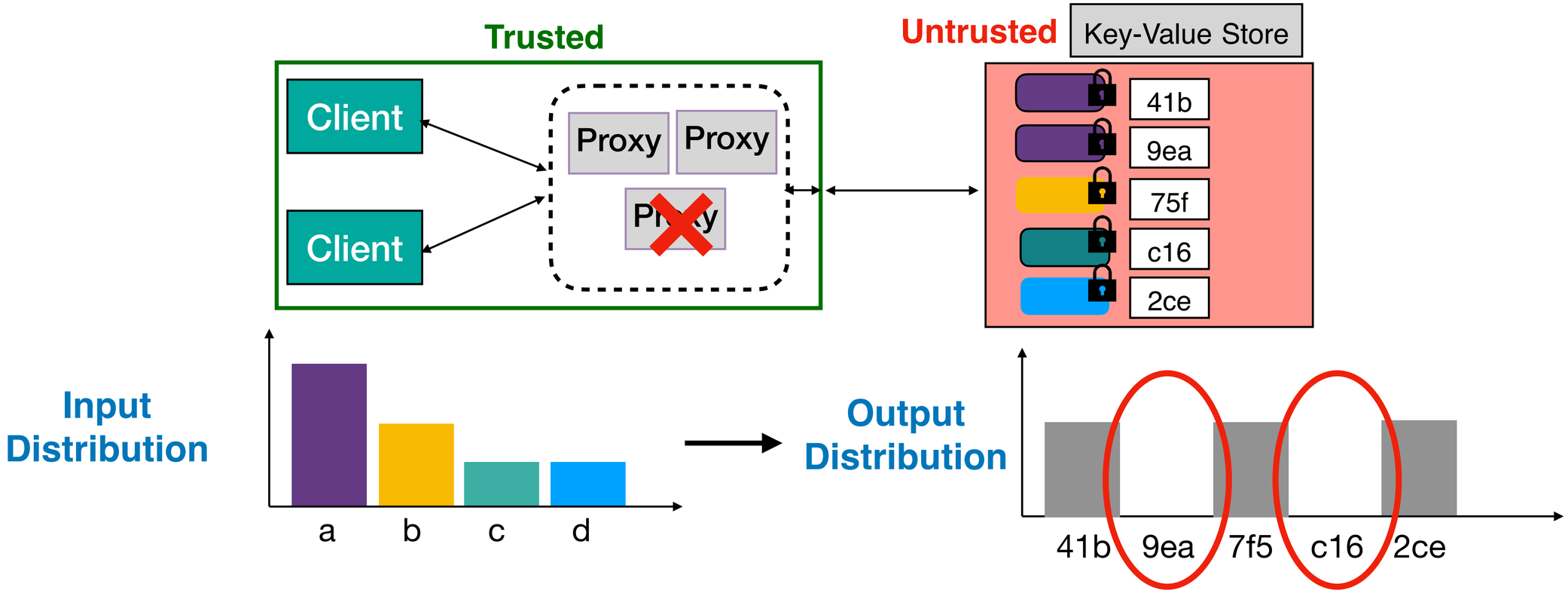
Uniform distribution is one but not the only way to achieve independence



Shortstack Key Insight

Obliviousness requires output distribution to be independent of input distribution

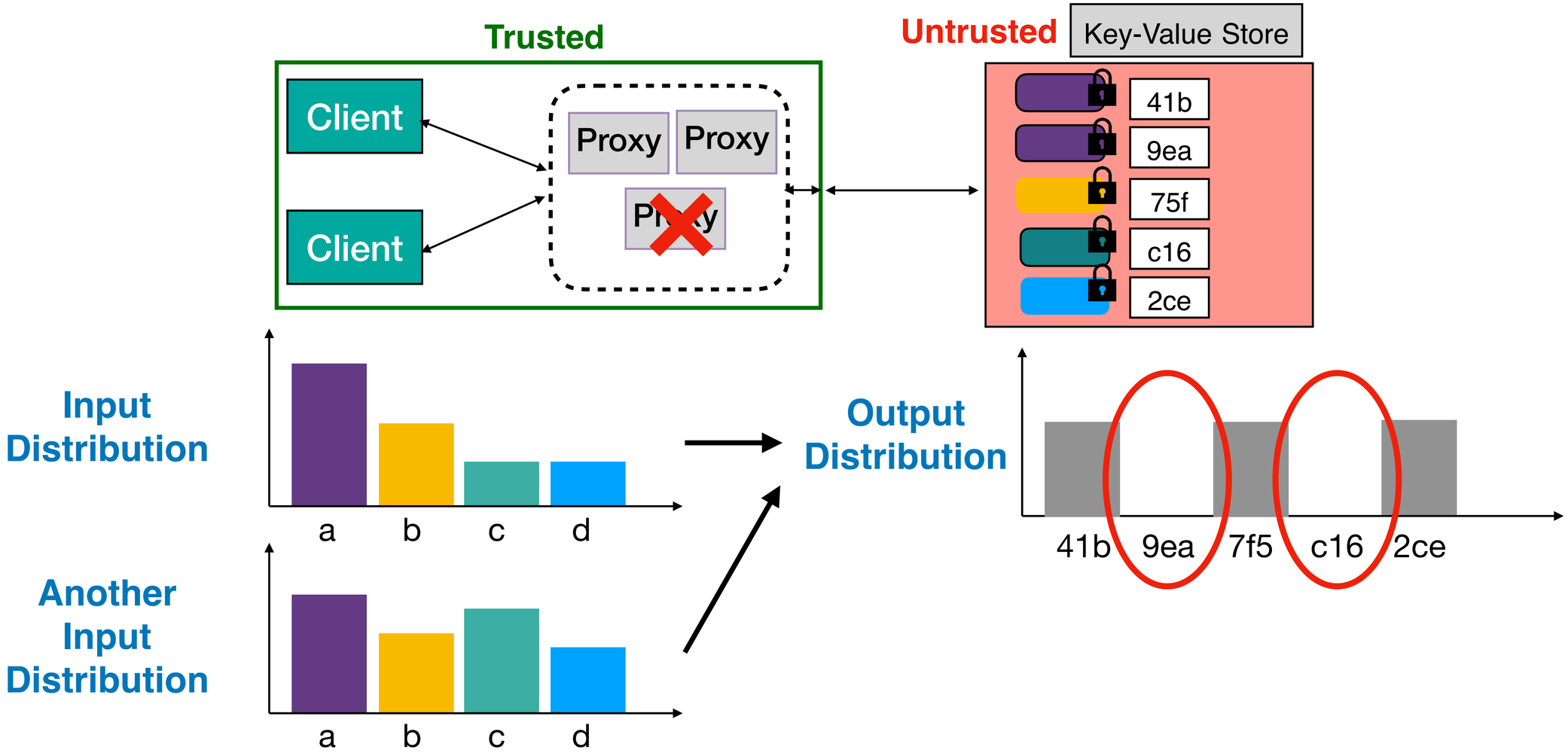
Uniform distribution is one but not the only way to achieve independence



Shortstack Key Insight

Obliviousness requires output distribution to be independent of input distribution

Uniform distribution is one but not the only way to achieve independence



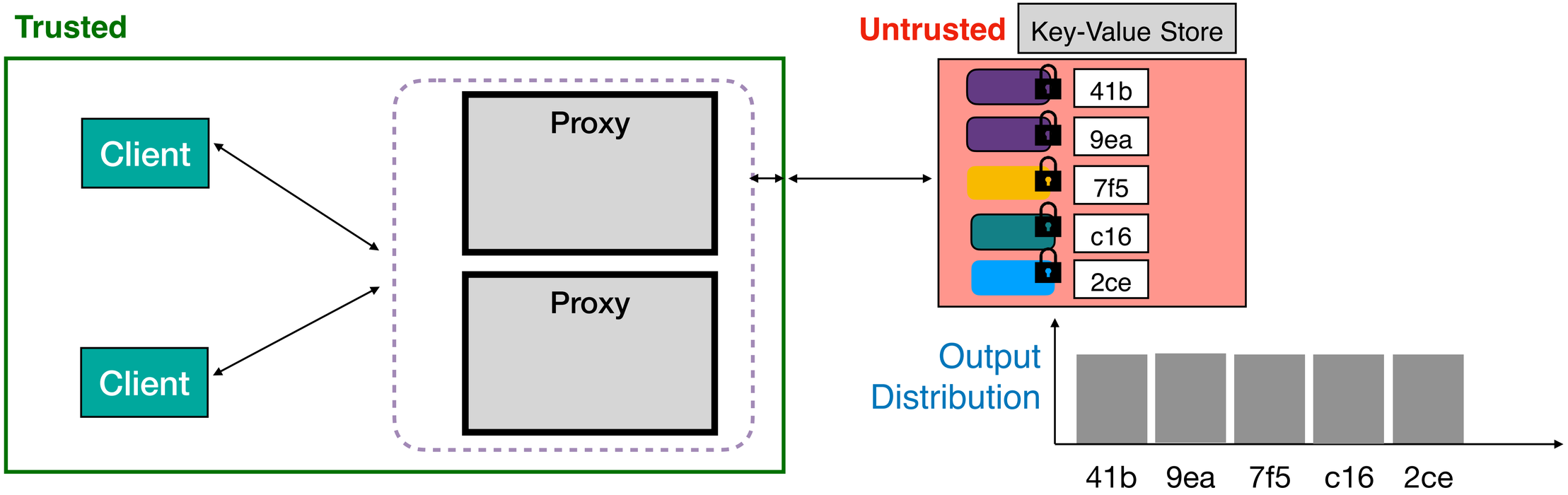
Shortstack Design Principle #1

Shortstack Design Principle #1

Partition query execution by Replicas

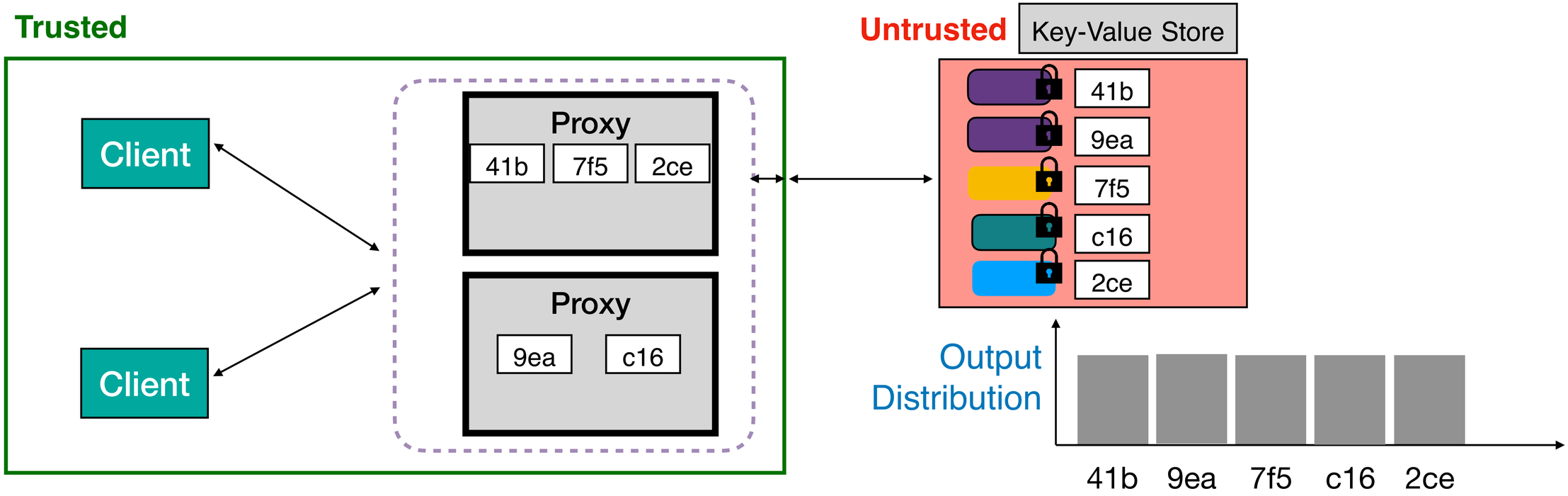
Shortstack Design Principle #1

Partition query execution by Replicas



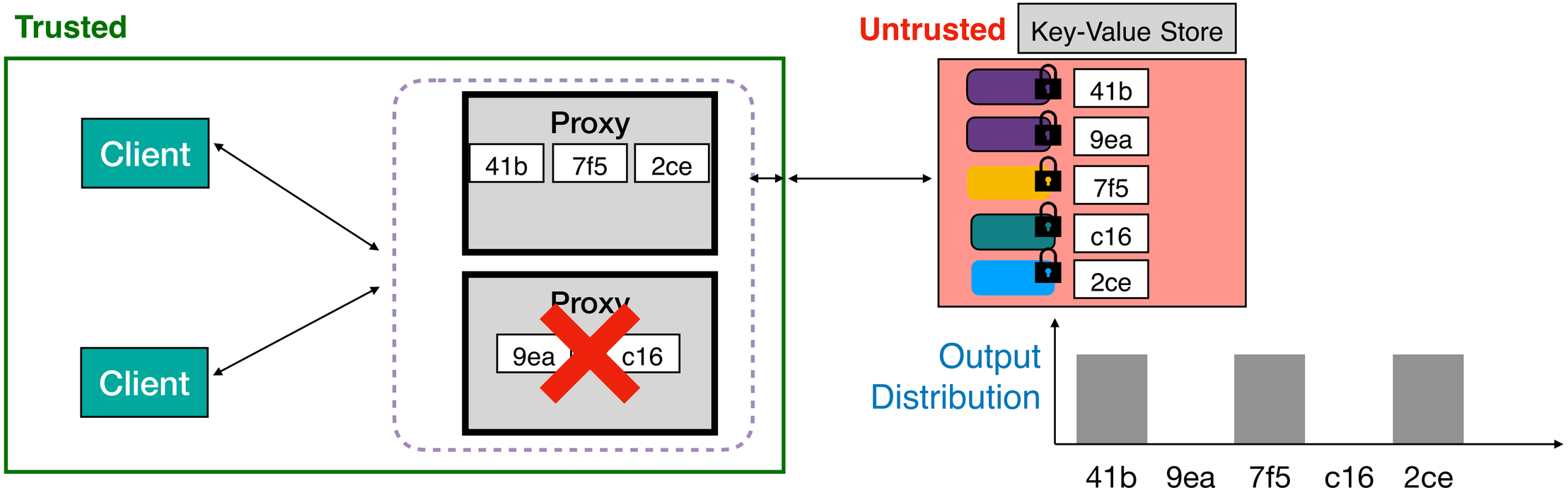
Shortstack Design Principle #1

Partition query execution by Replicas



Shortstack Design Principle #1

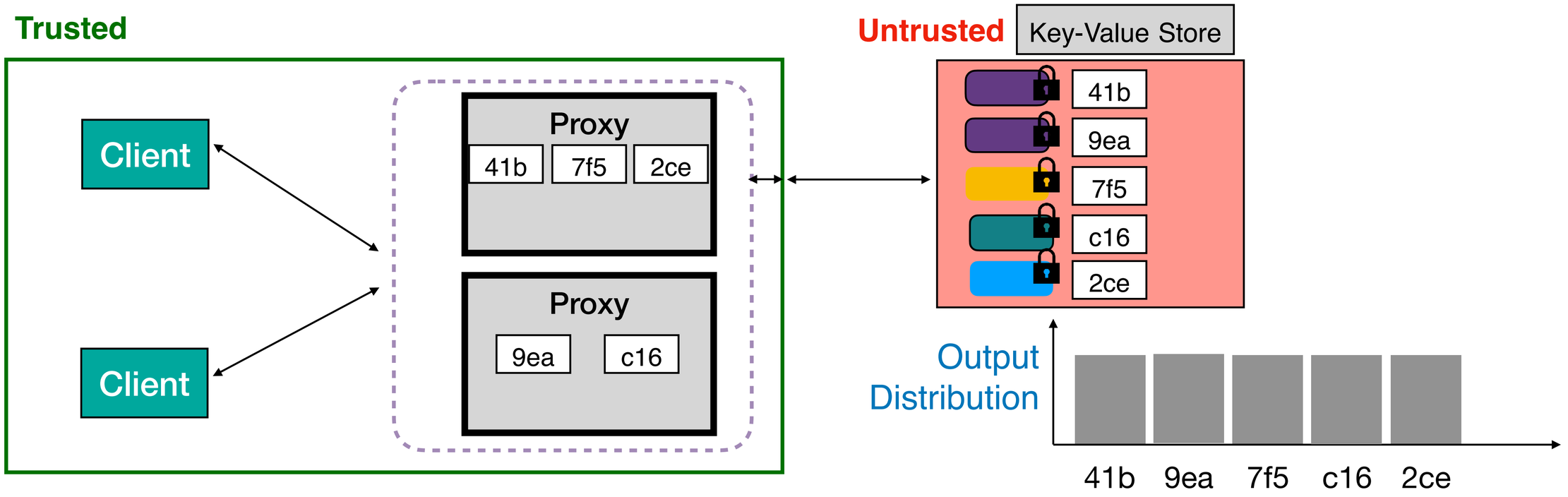
Partition query execution by Replicas



Upon failure, output distribution is uniform over a random subset of replicas

Output distribution is independent of input distribution (realizing our key insight)

Shortstack Design Principle #2

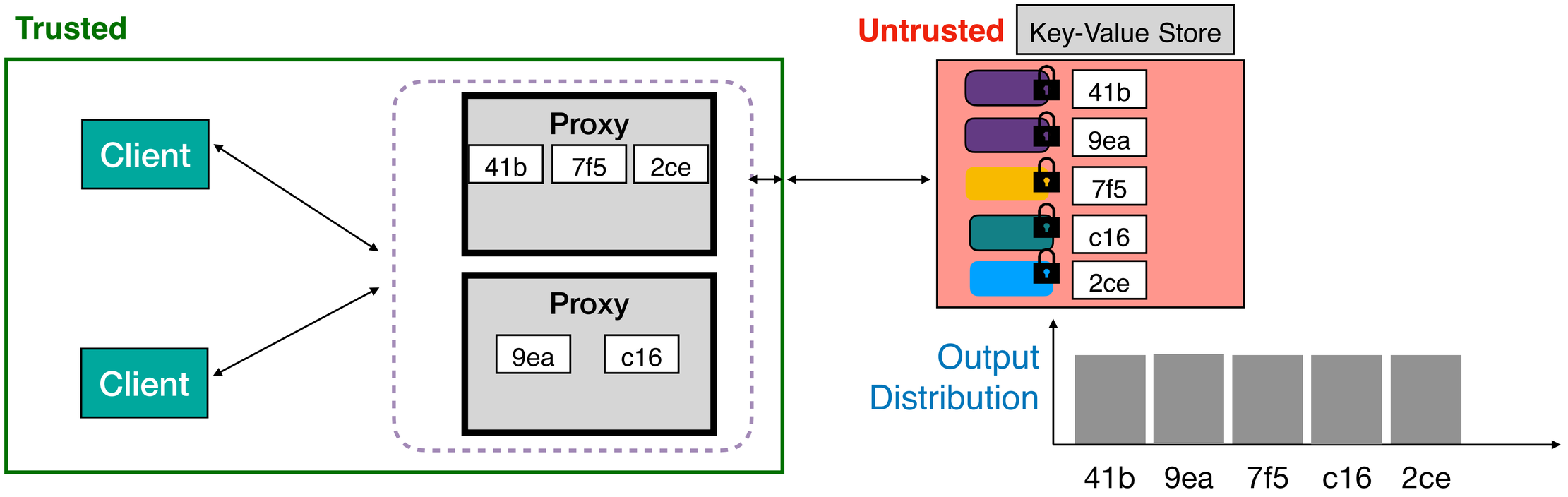


Shortstack Design Principle #2

Proxy State

a	41b, 9ea
b	7f5
c	c16
d	2ce

Partition Proxy state by Keys



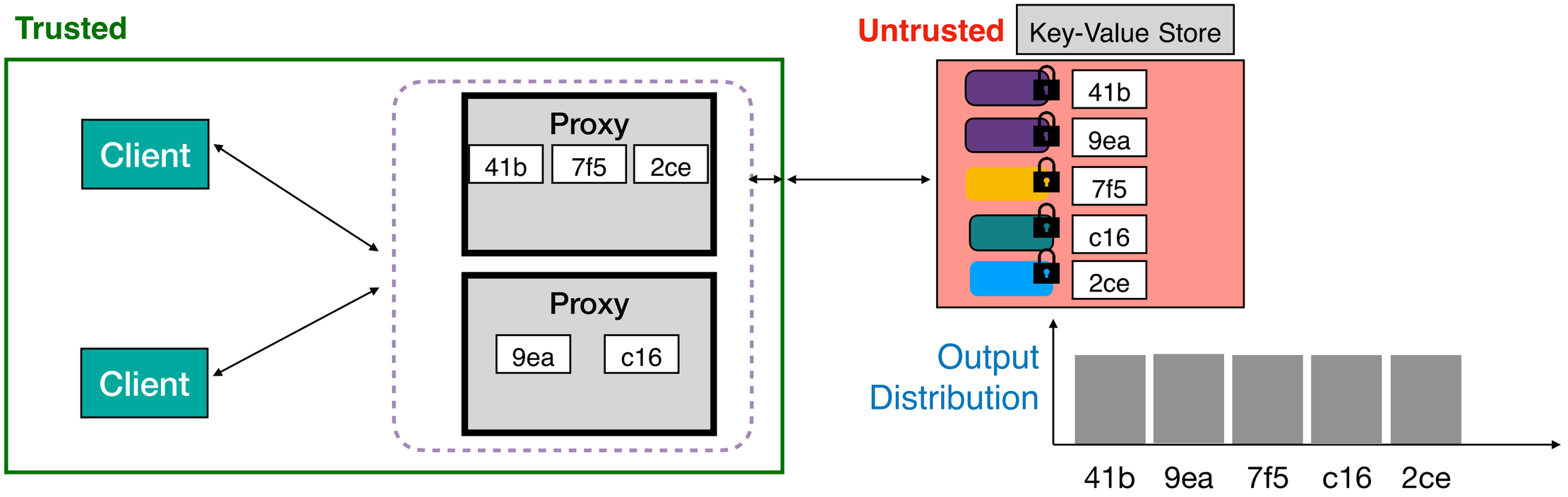
Shortstack Design Principle #2

Proxy State

a → 41b, 9ea
b → 7f5
c → c16
d → 2ce

Partition Proxy state by Keys

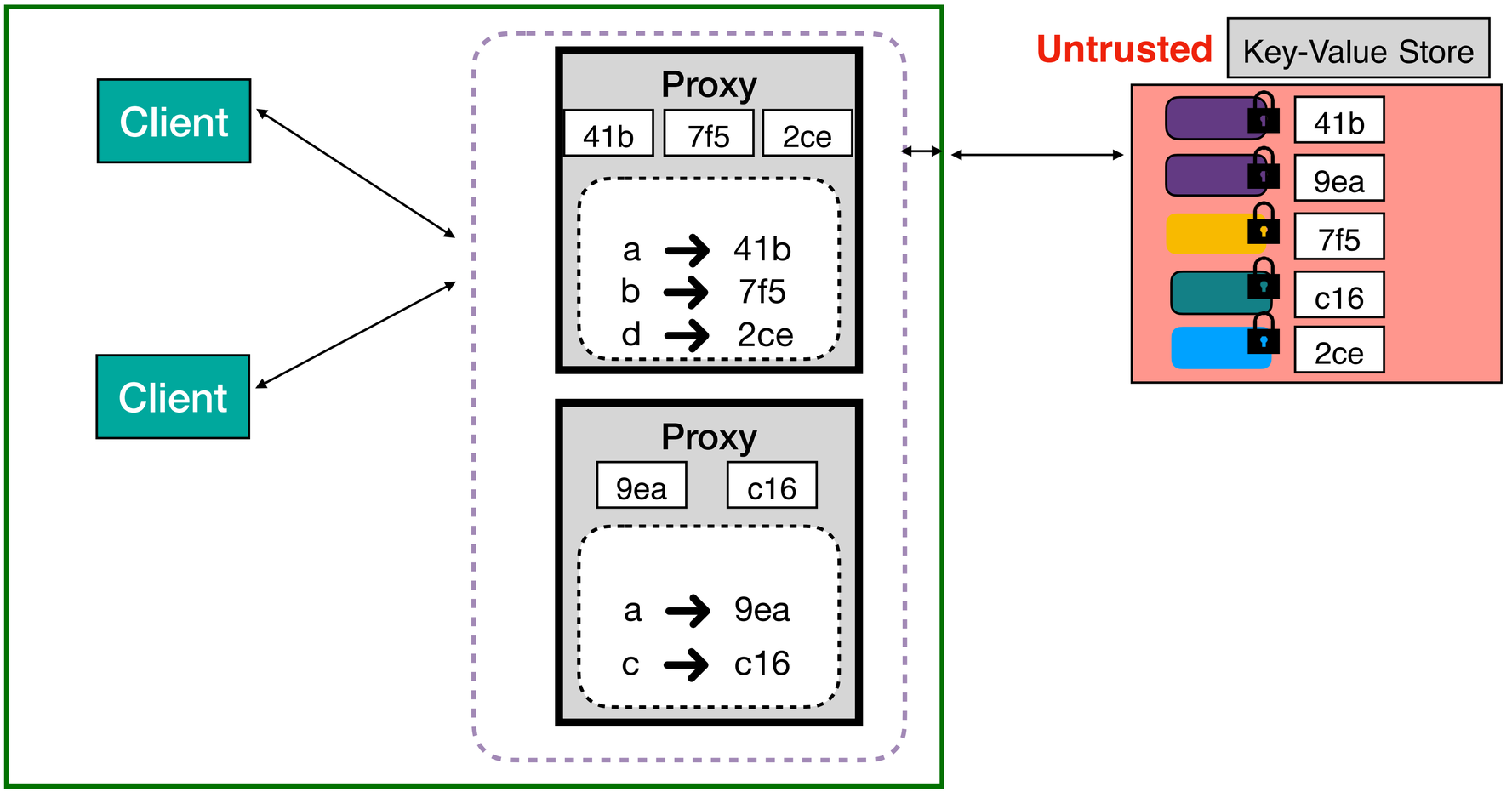
Example to provide intuition



Shortstack Design Principle #2

Partition Proxy state by Keys

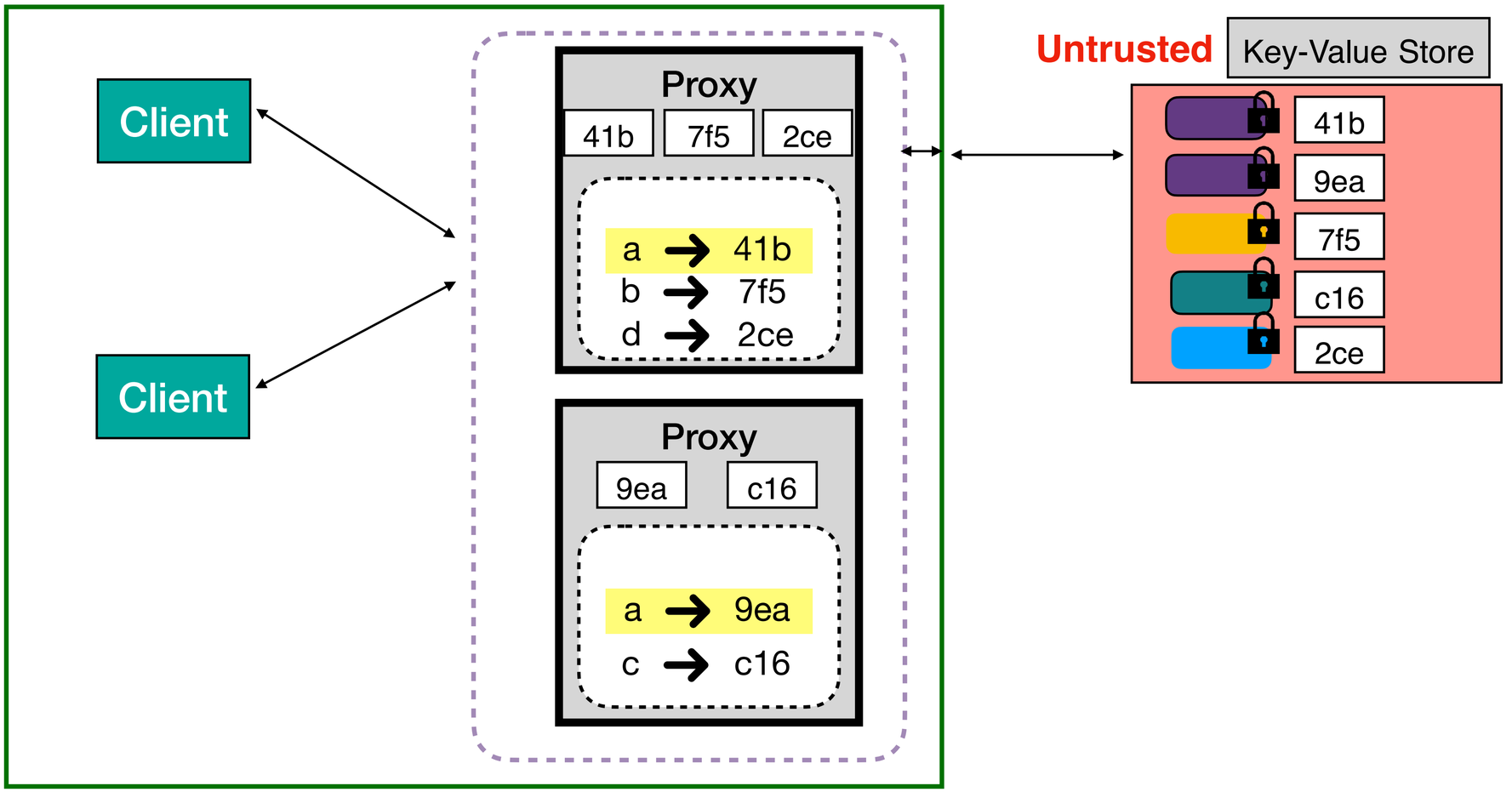
Example to provide intuition



Shortstack Design Principle #2

Partition Proxy state by Keys

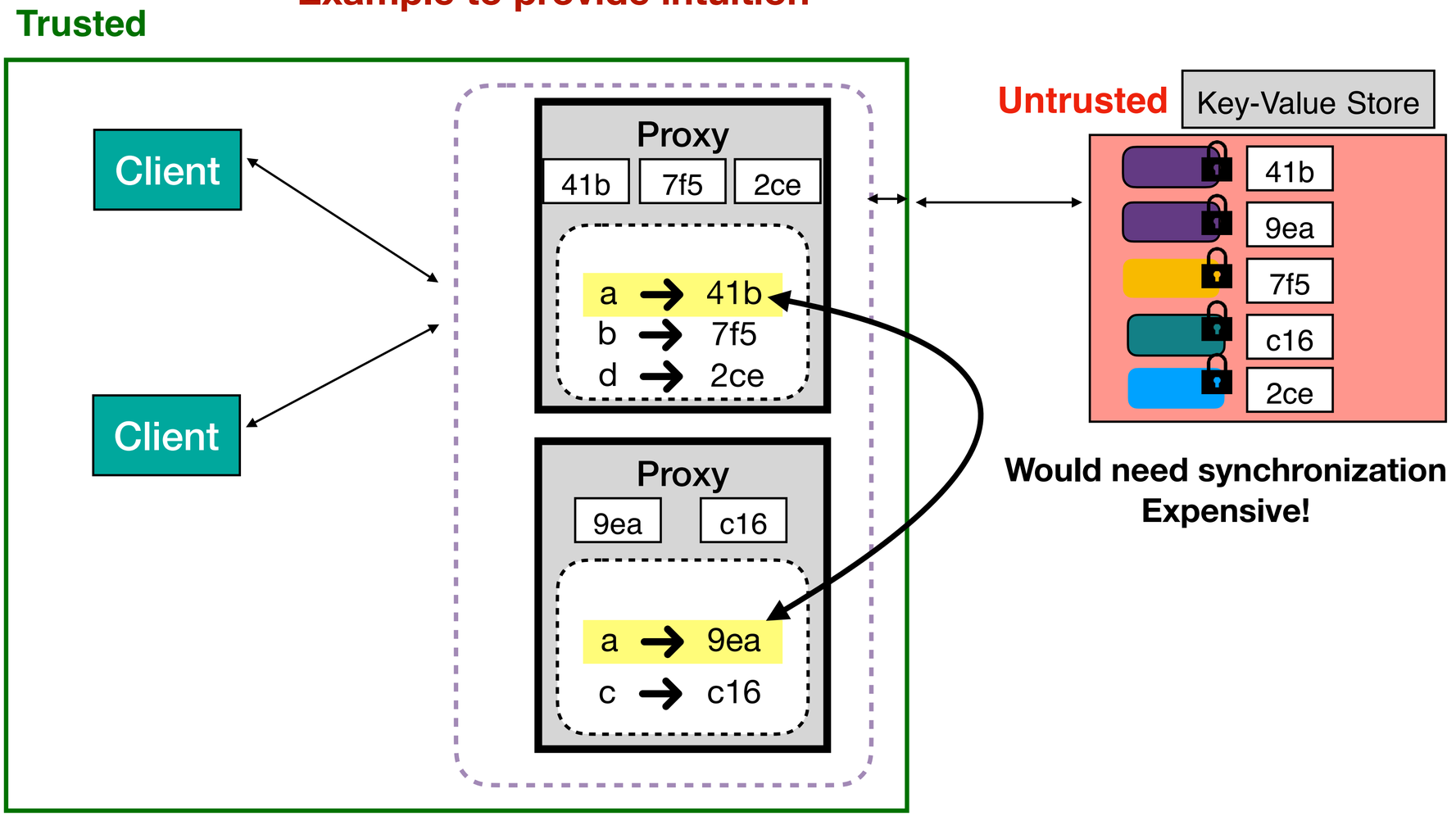
Example to provide intuition



Shortstack Design Principle #2

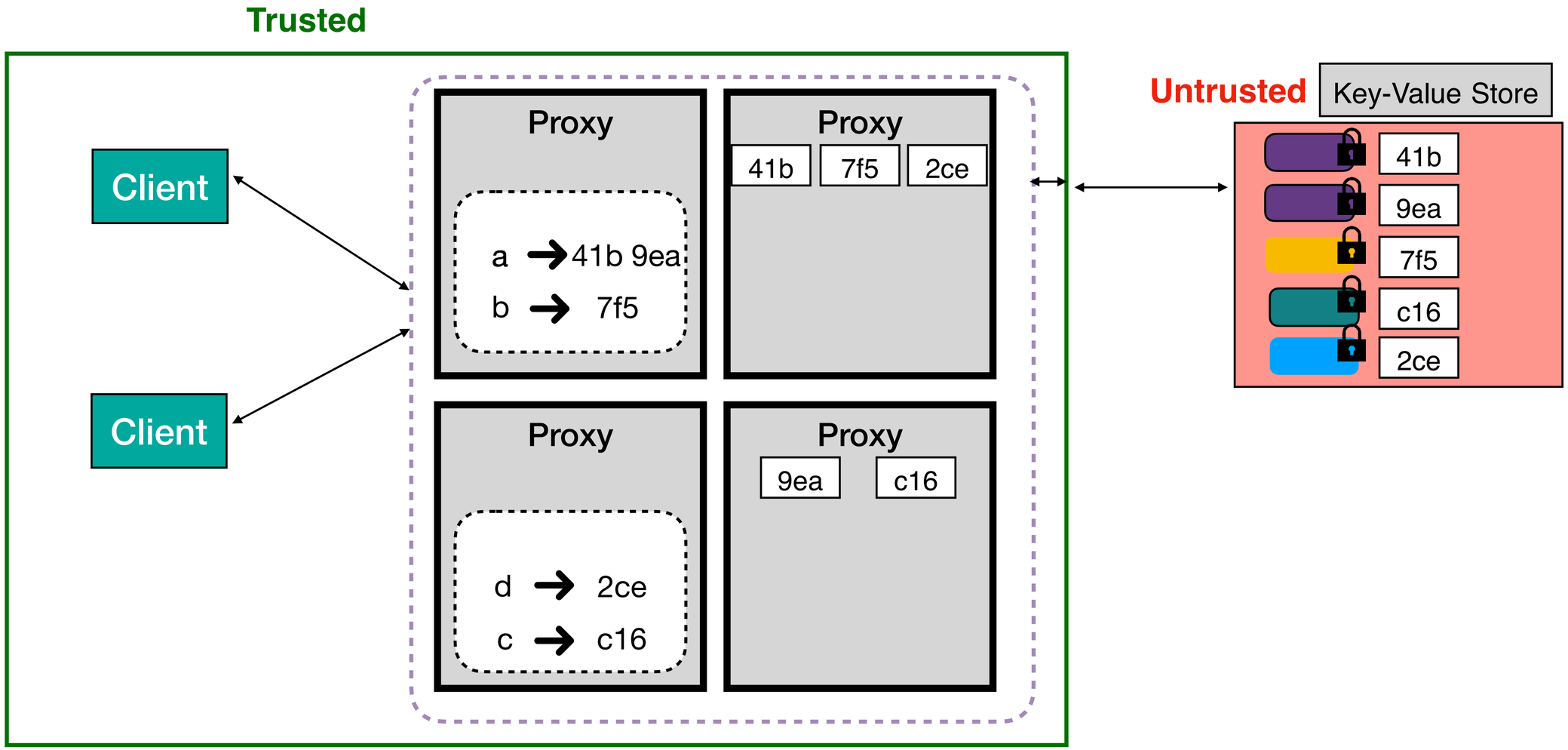
Partition Proxy state by Keys

Example to provide intuition



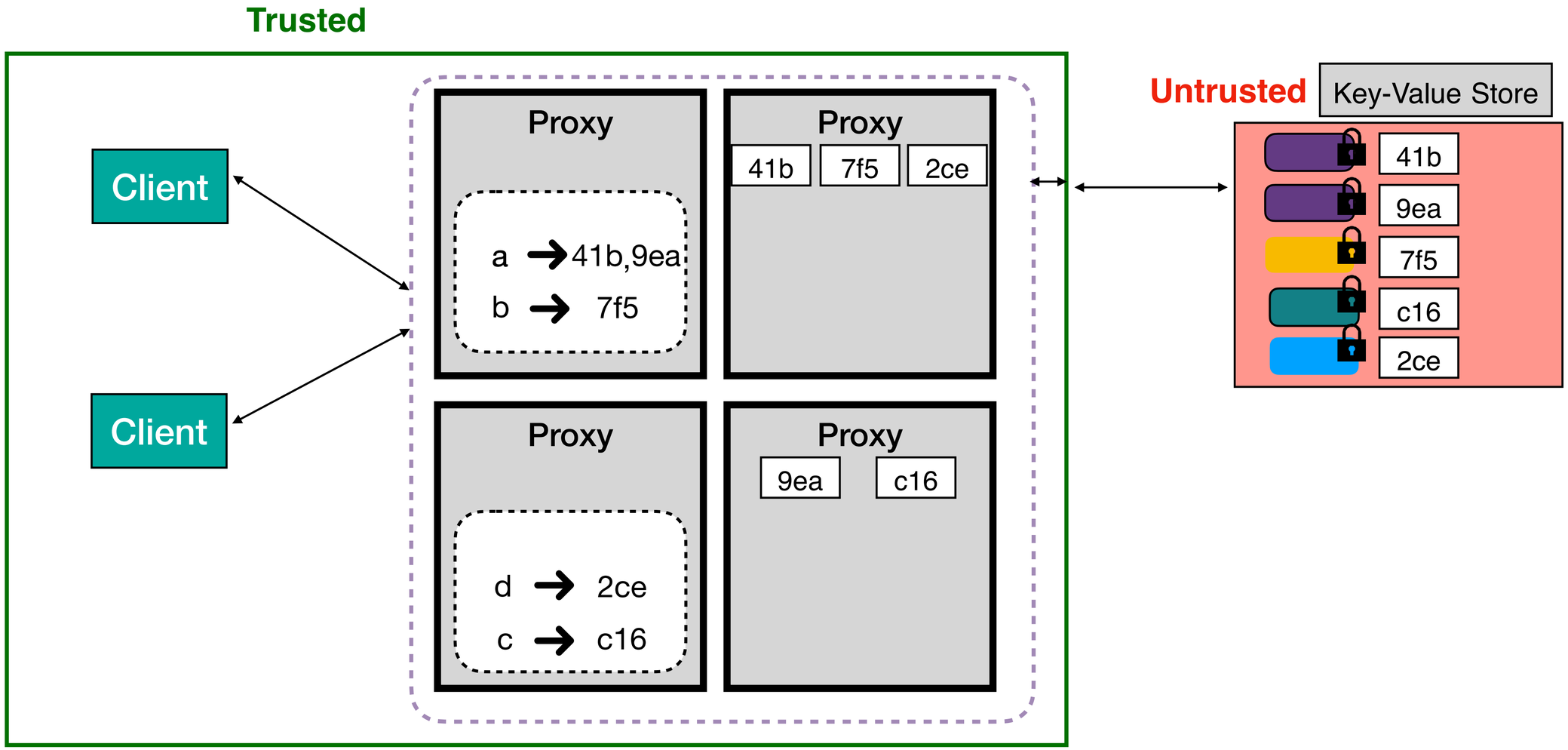
Shortstack Design Principle #2

Partition Proxy state by Keys

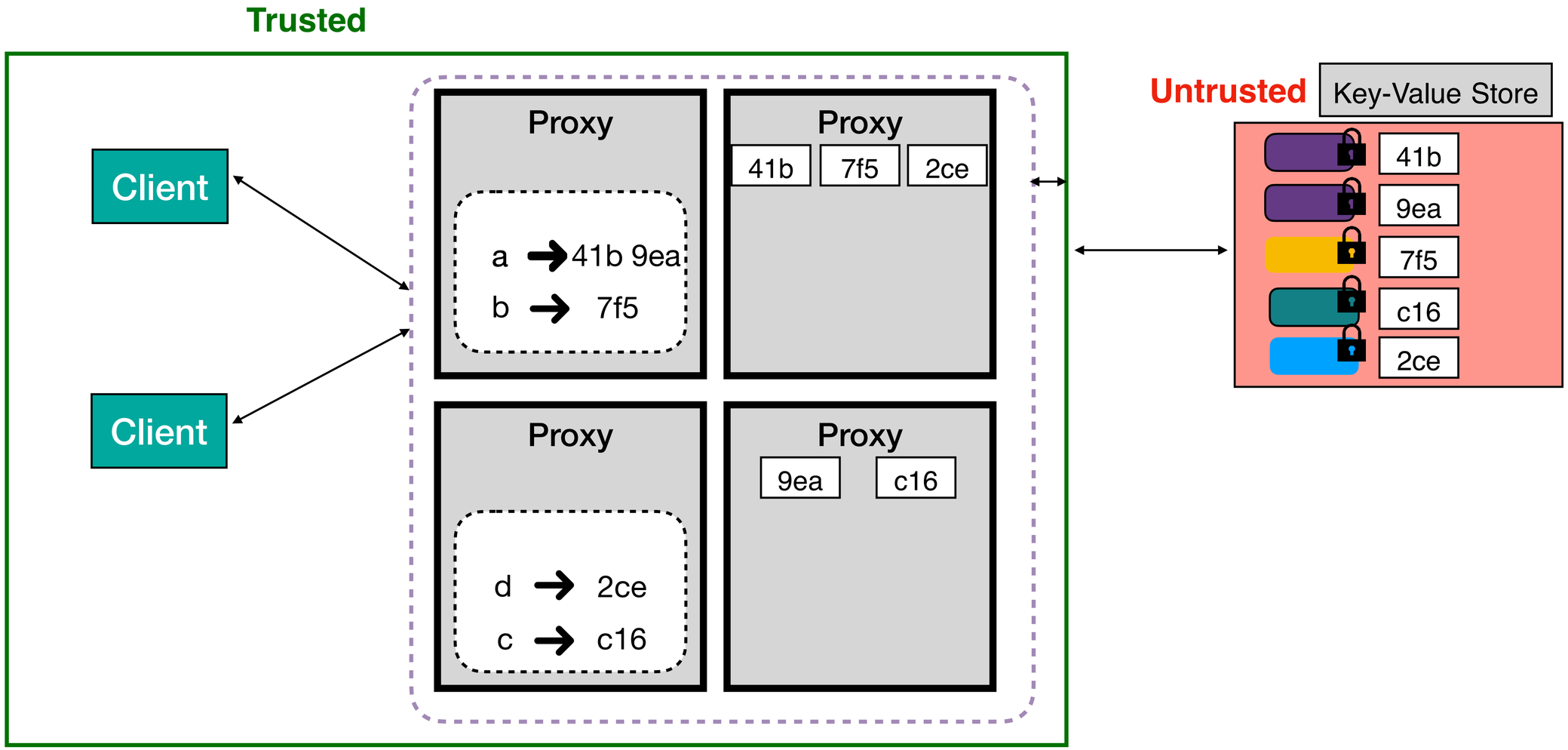


Shortstack Design Principle #2

Partition Proxy state by Keys

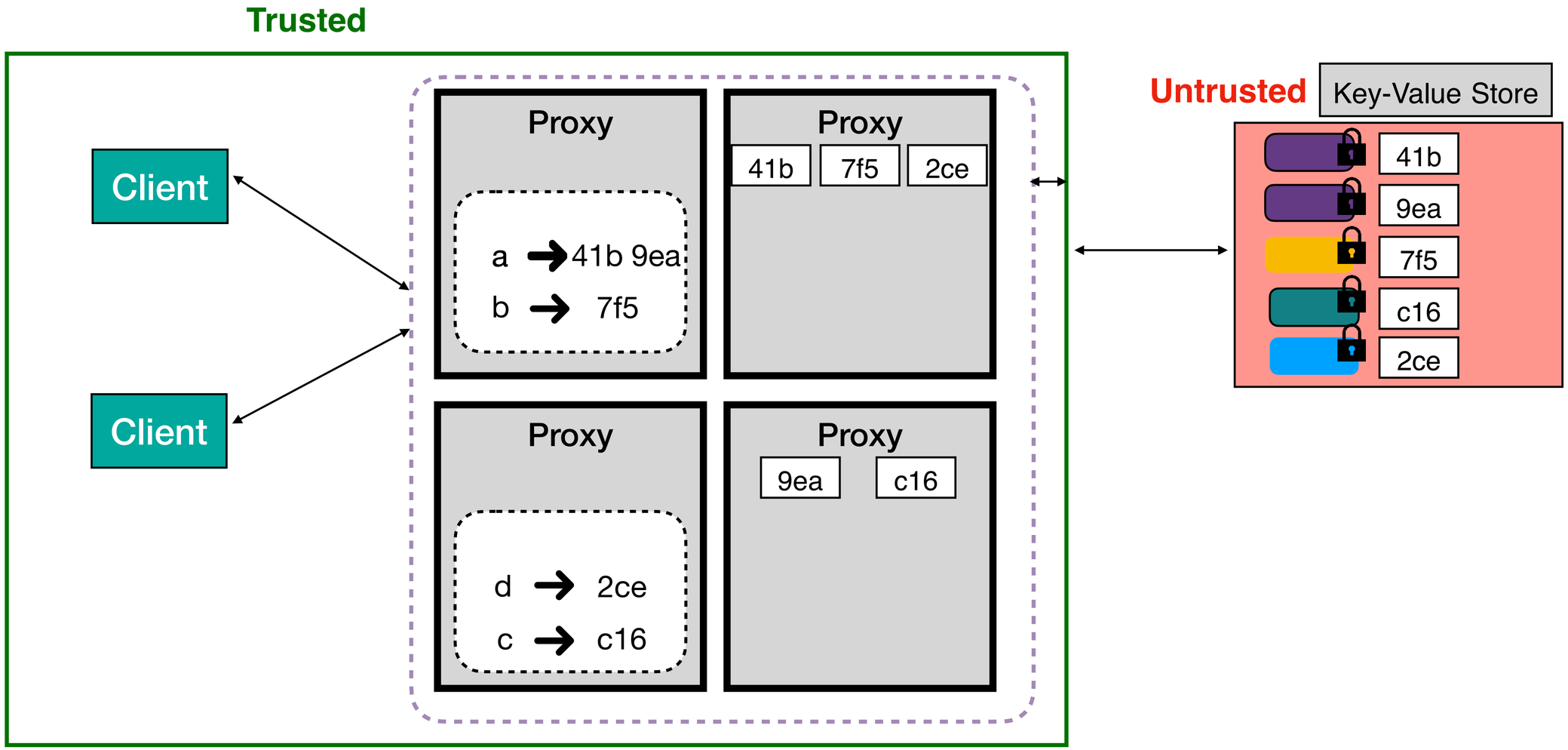


Shortstack Design Principle #3



Shortstack Design Principle #3

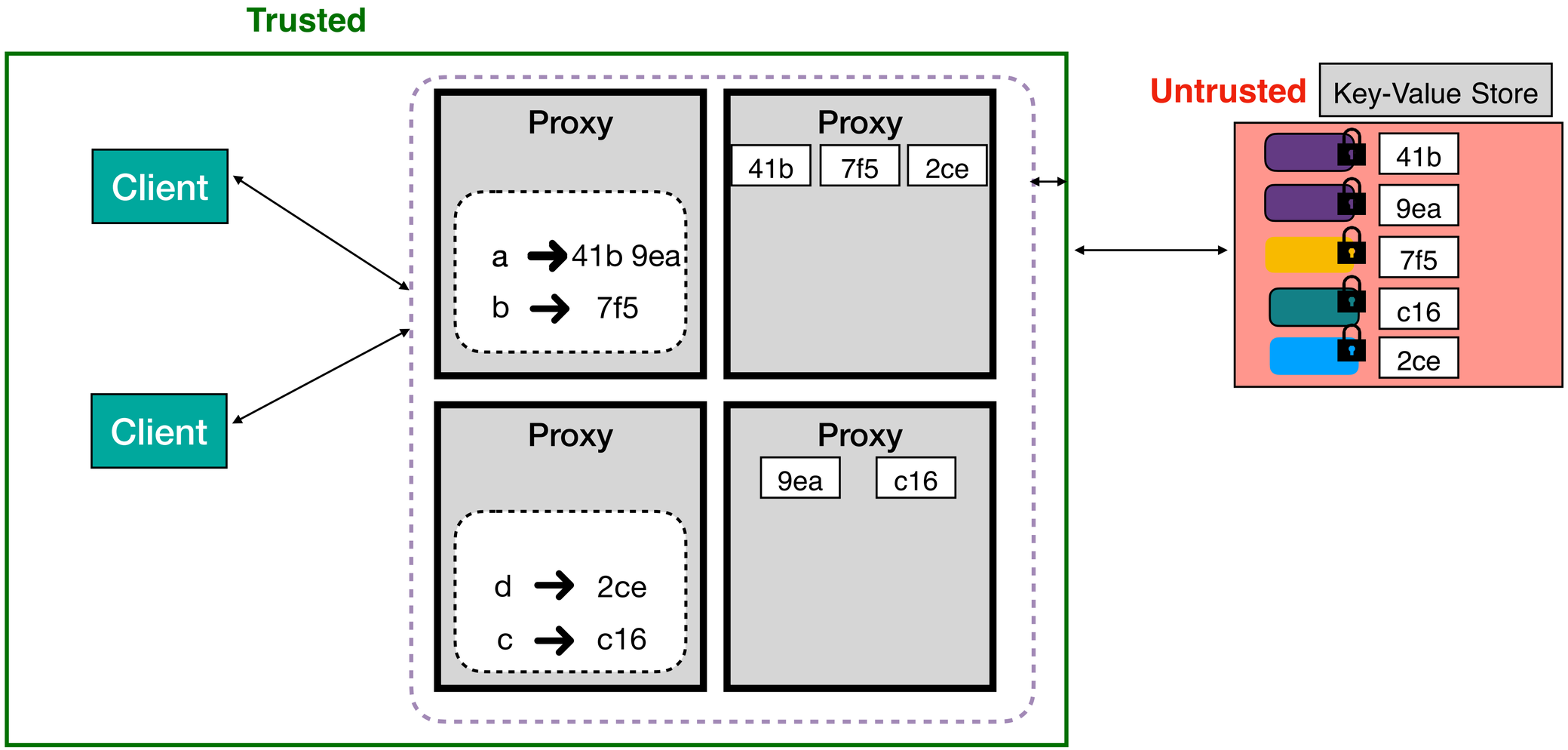
Query Generation over all Keys



Shortstack Design Principle #3

Query Generation over all Keys

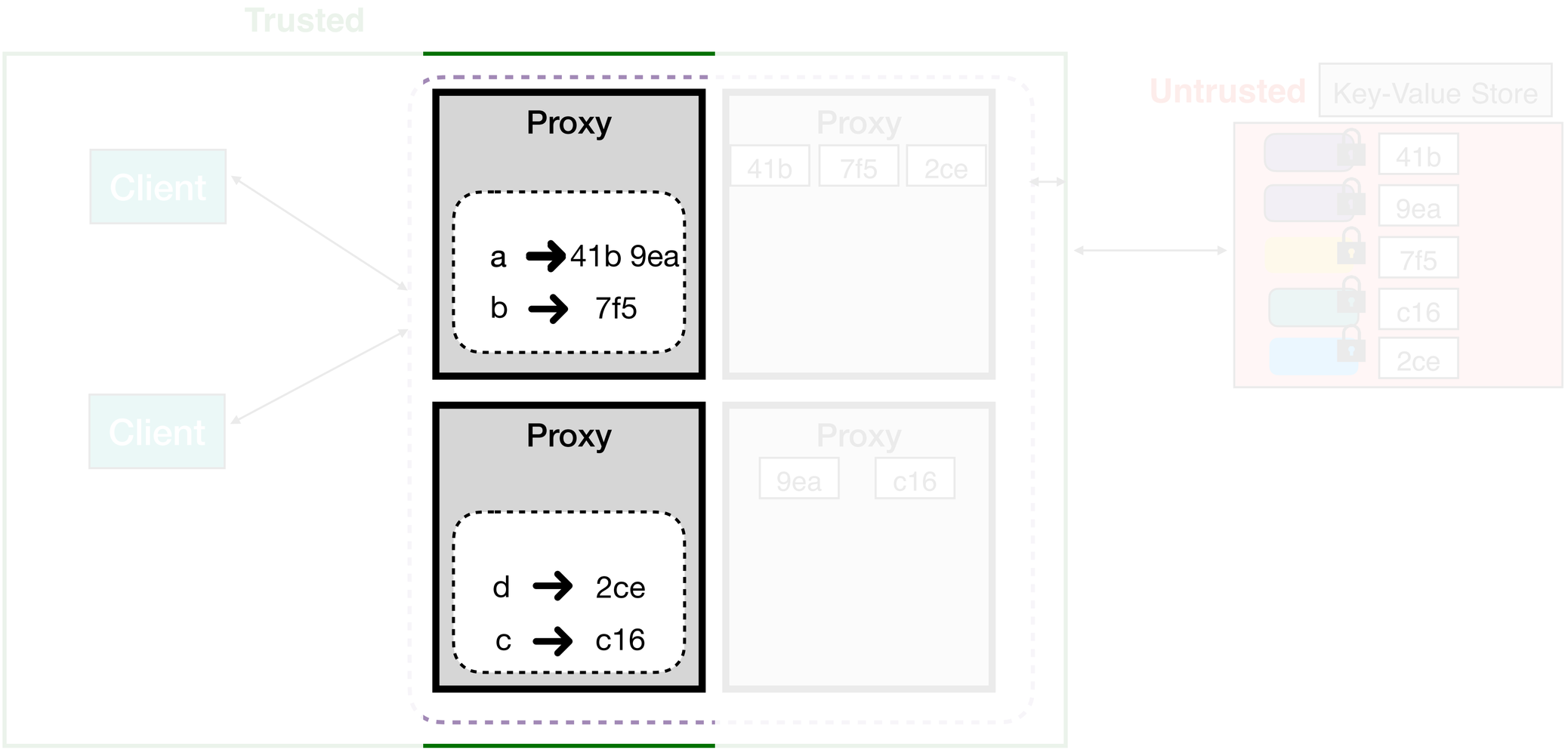
Example to provide intuition



Shortstack Design Principle #3

Query Generation over all Keys

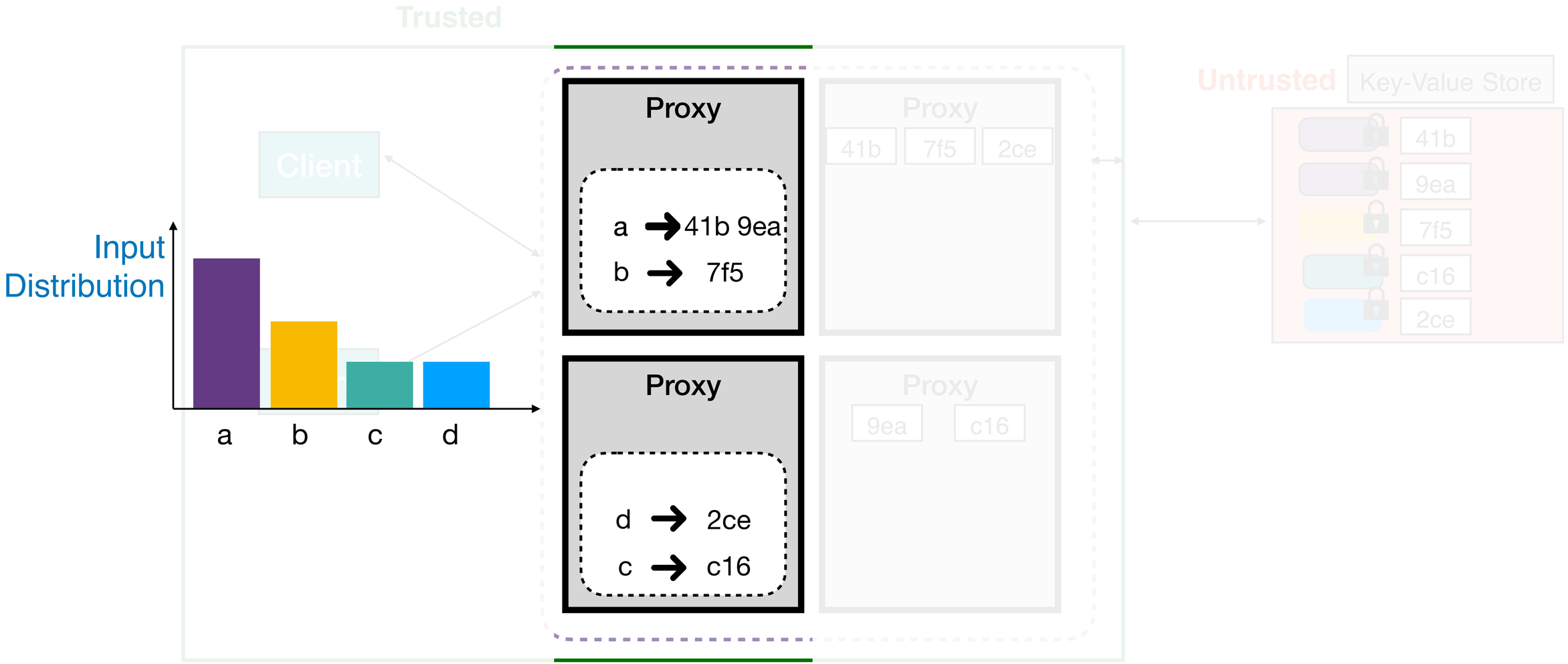
Example to provide intuition



Shortstack Design Principle #3

Query Generation over all Keys

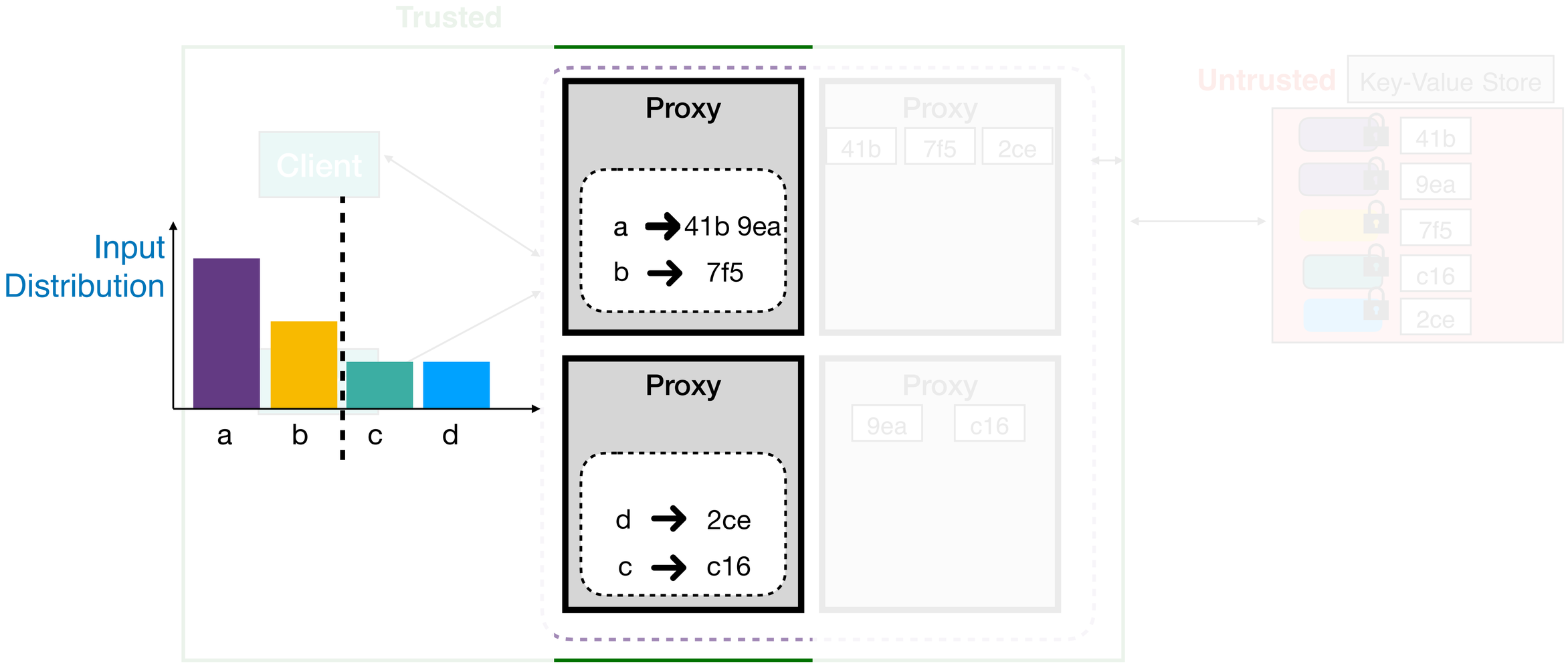
Example to provide intuition



Shortstack Design Principle #3

Query Generation over all Keys

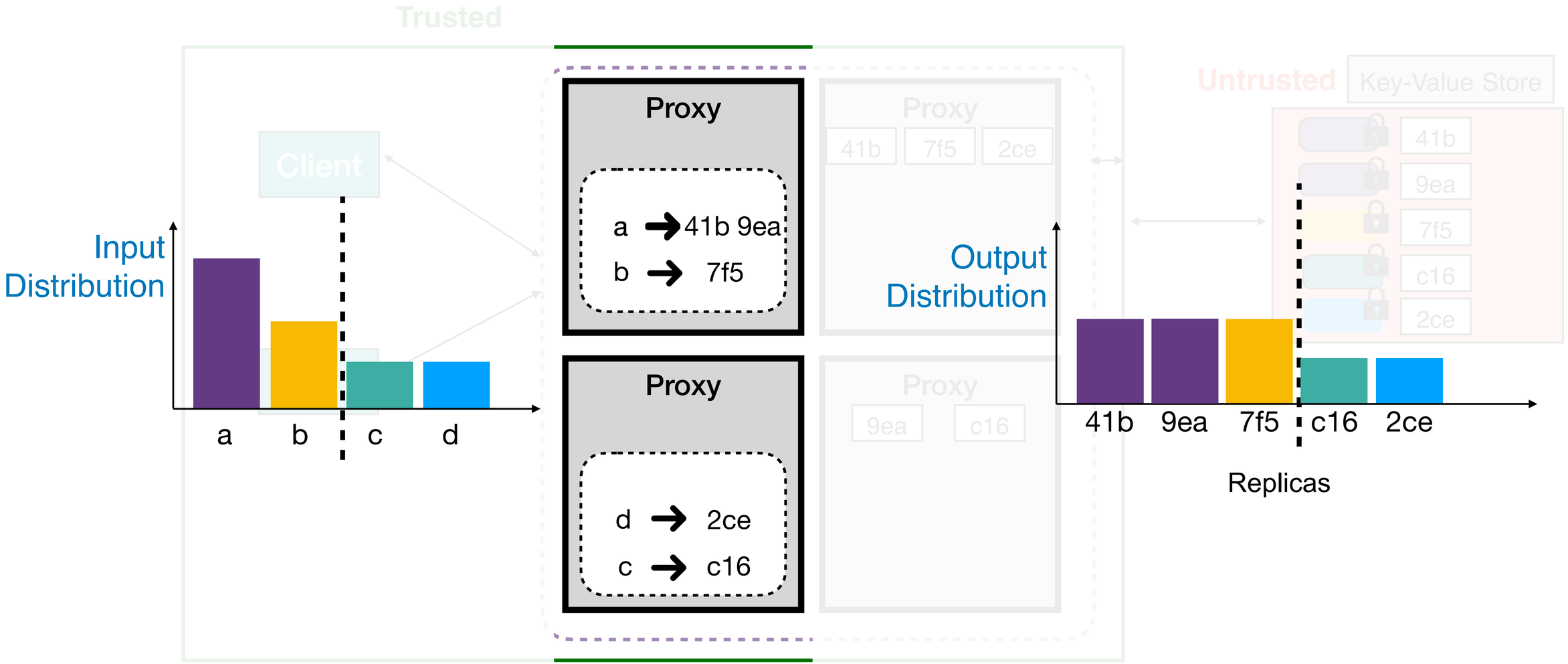
Example to provide intuition



Shortstack Design Principle #3

Query Generation over all Keys

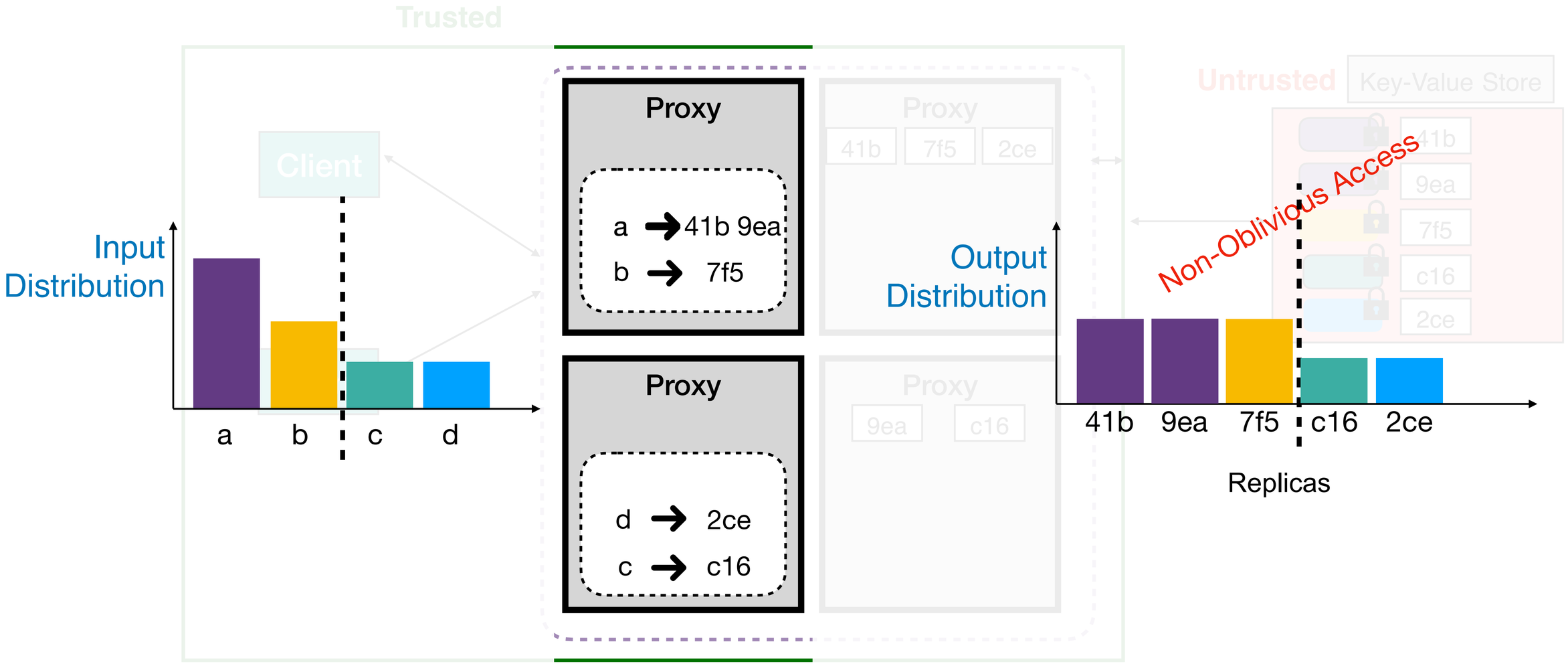
Example to provide intuition



Shortstack Design Principle #3

Query Generation over all Keys

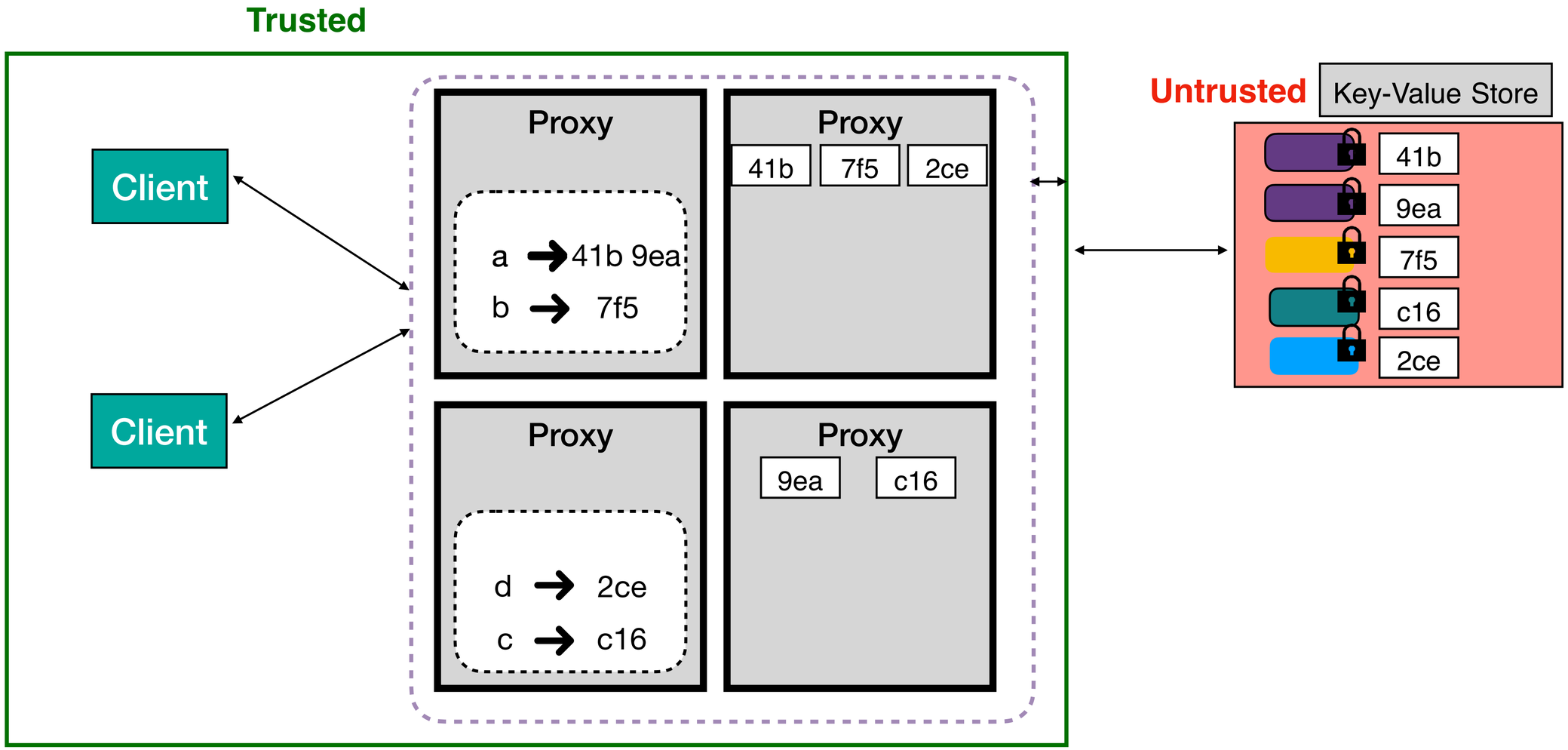
Example to provide intuition



Shortstack Design Principle #3

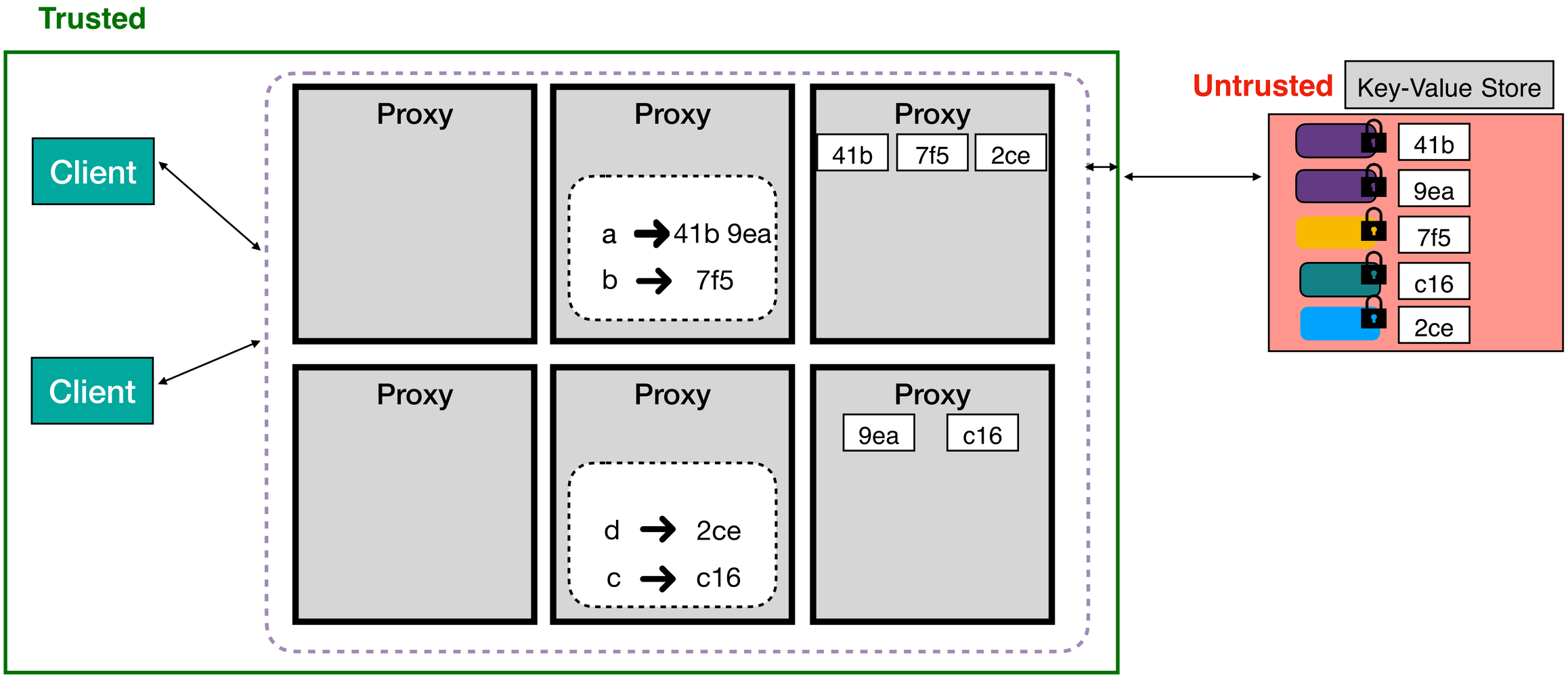
Query Generation over all Keys

Example to provide intuition



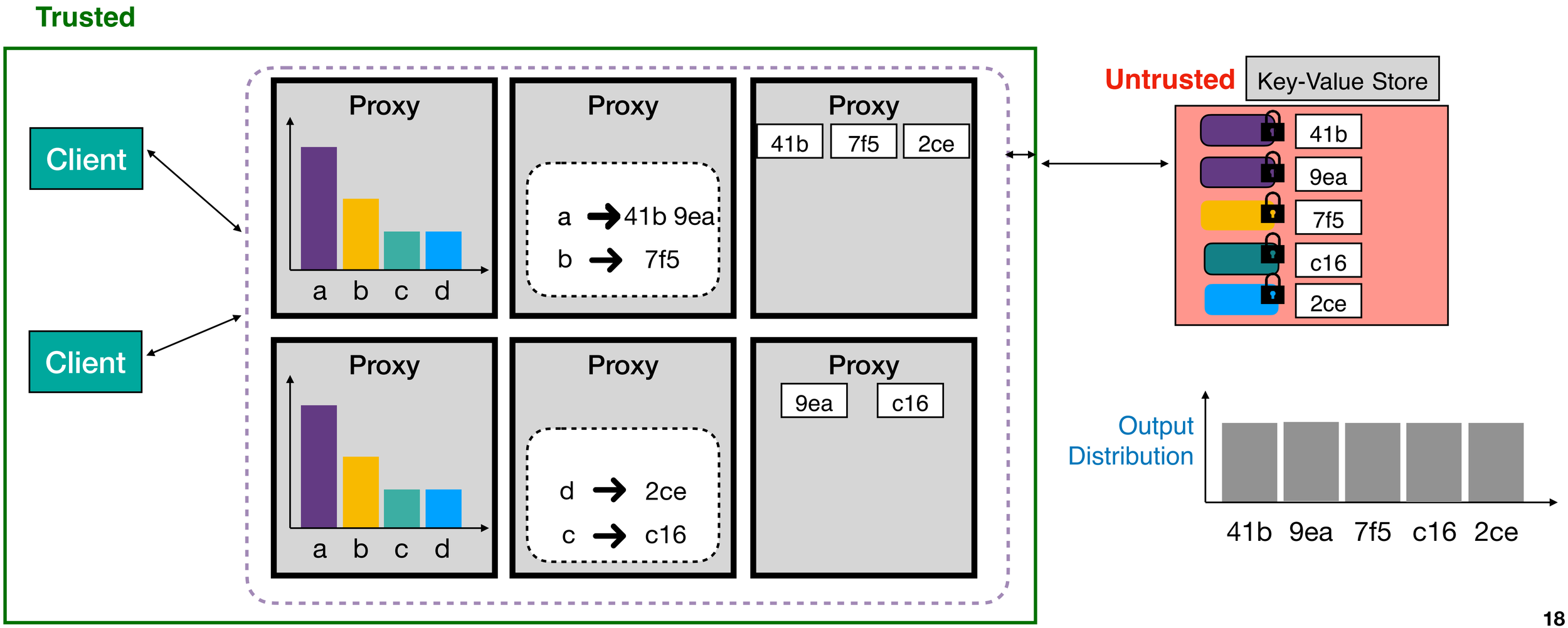
Shortstack Design Principle #3

Query Generation over all Keys

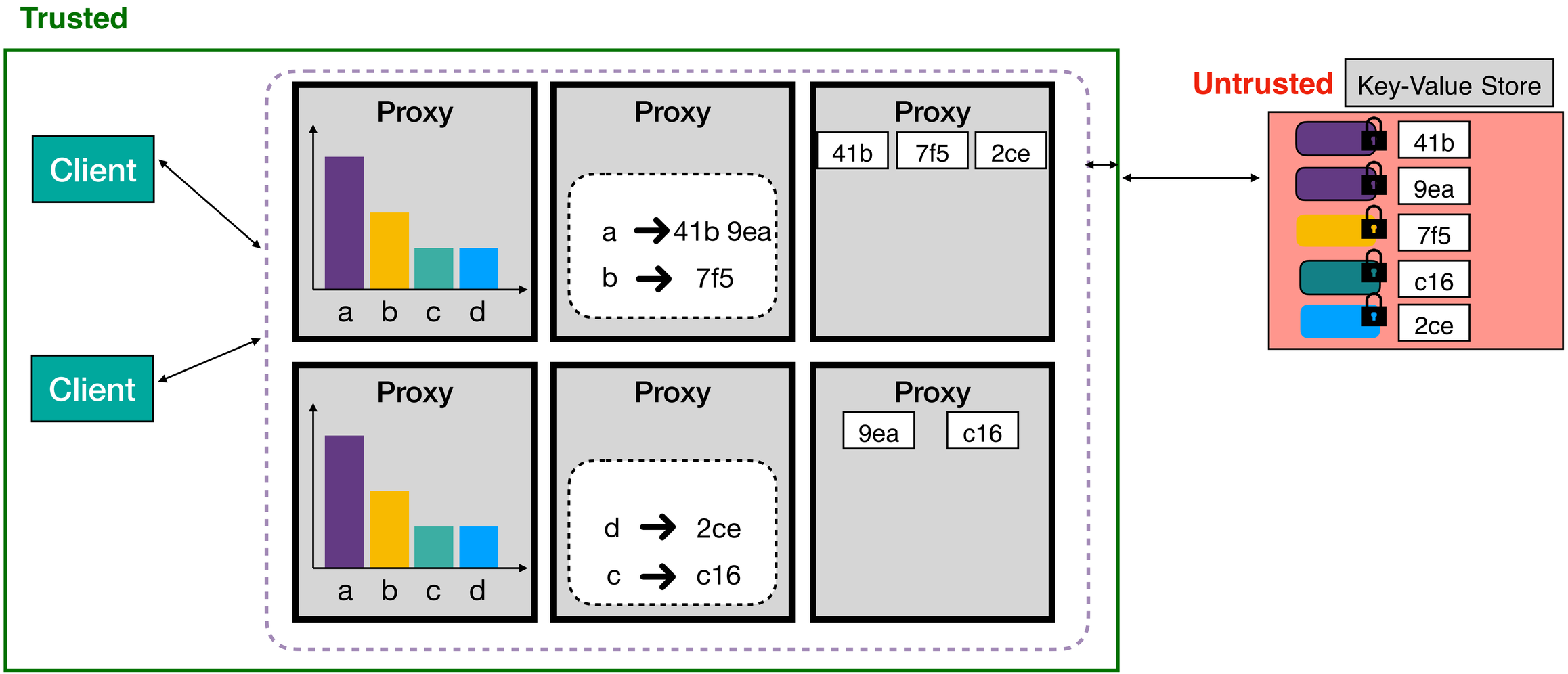


Shortstack Design Principle #3

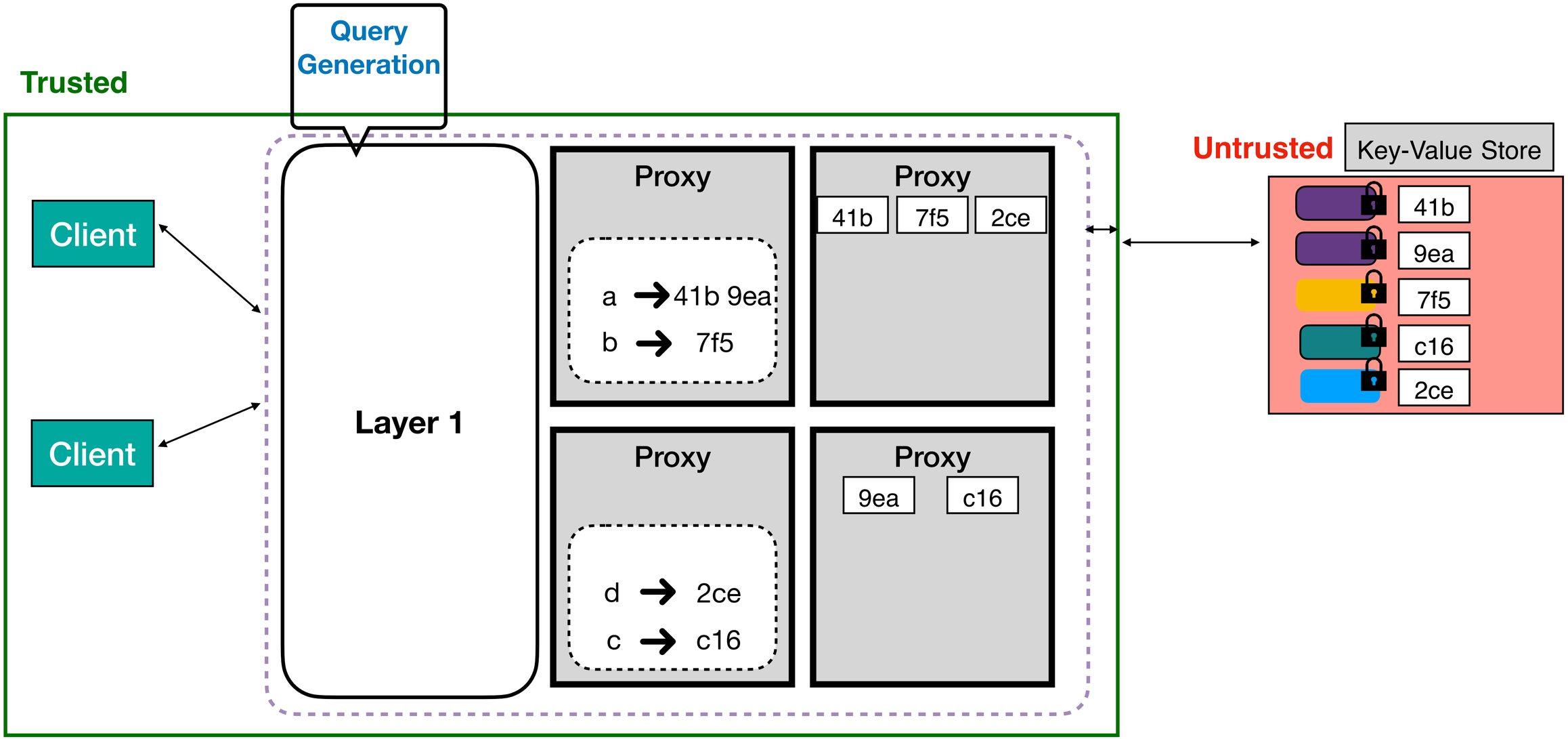
Query Generation over all Keys



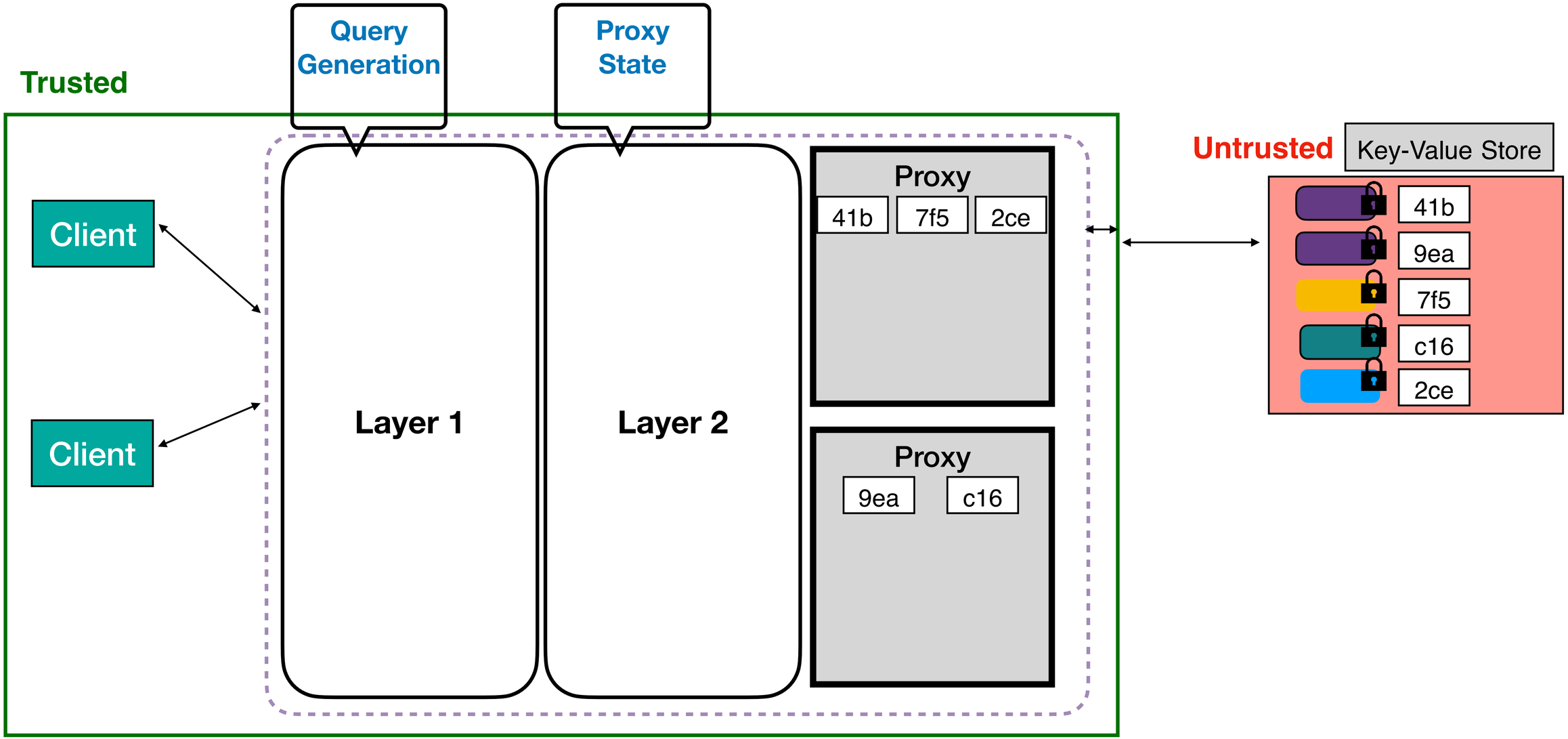
Shortstack Design Summary: Logical



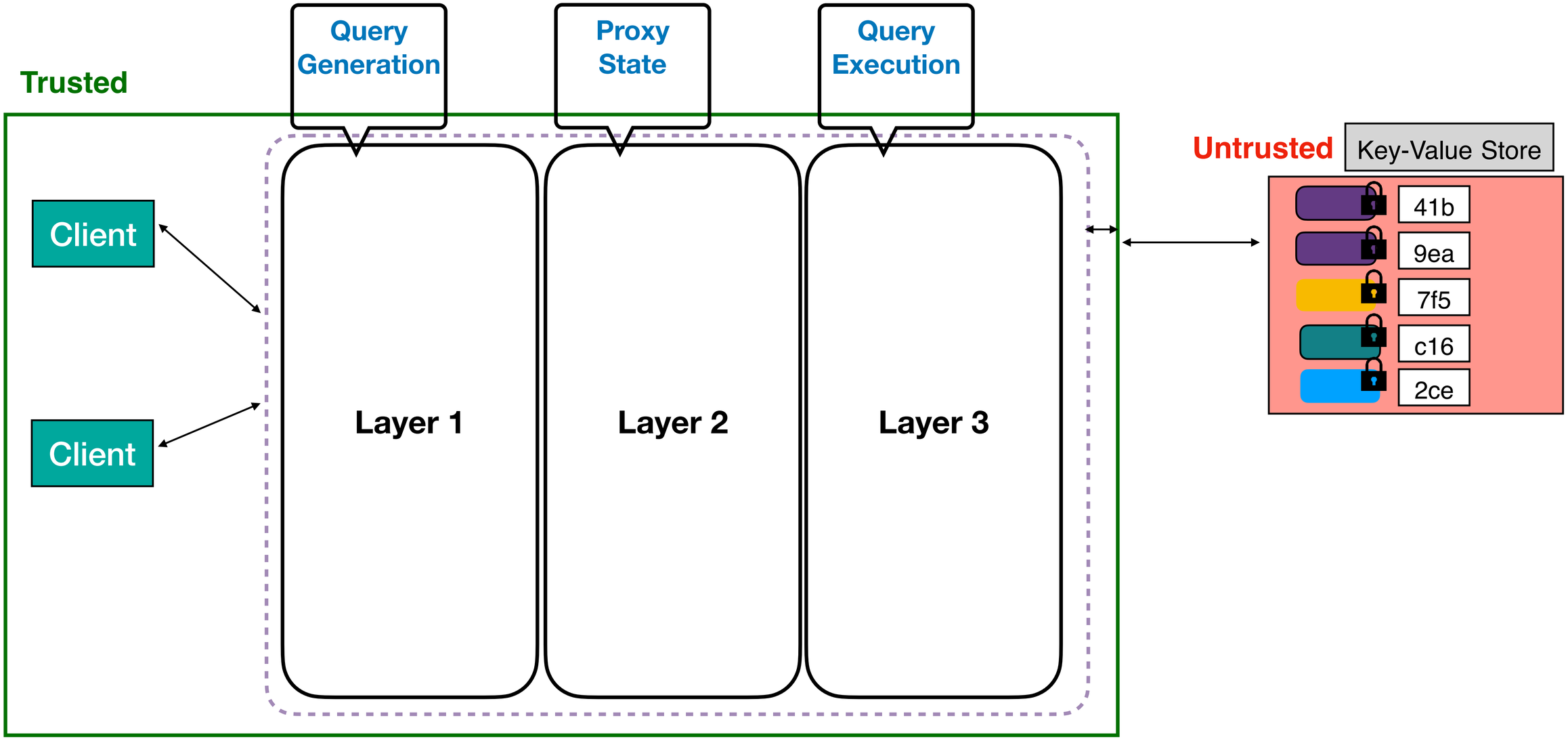
Shortstack Design Summary: Logical



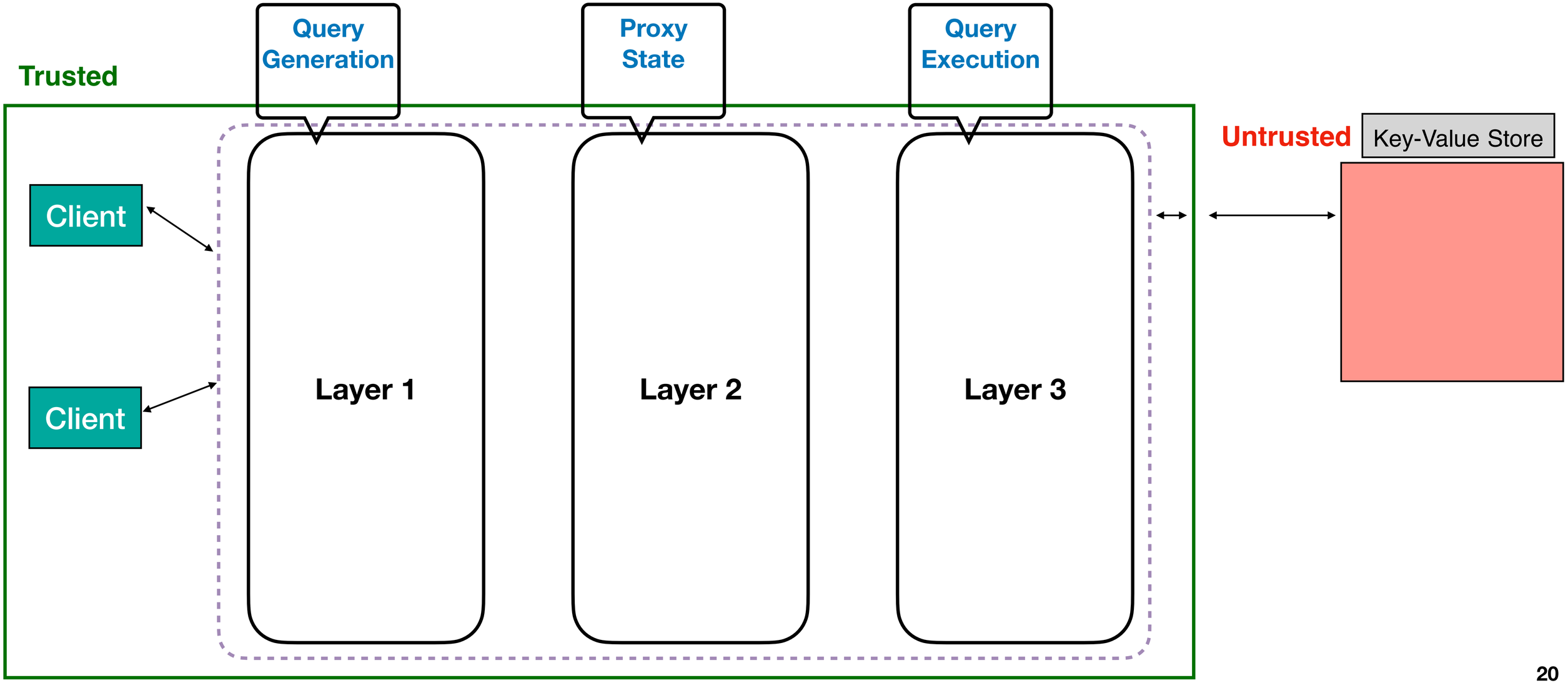
Shortstack Design Summary: Logical



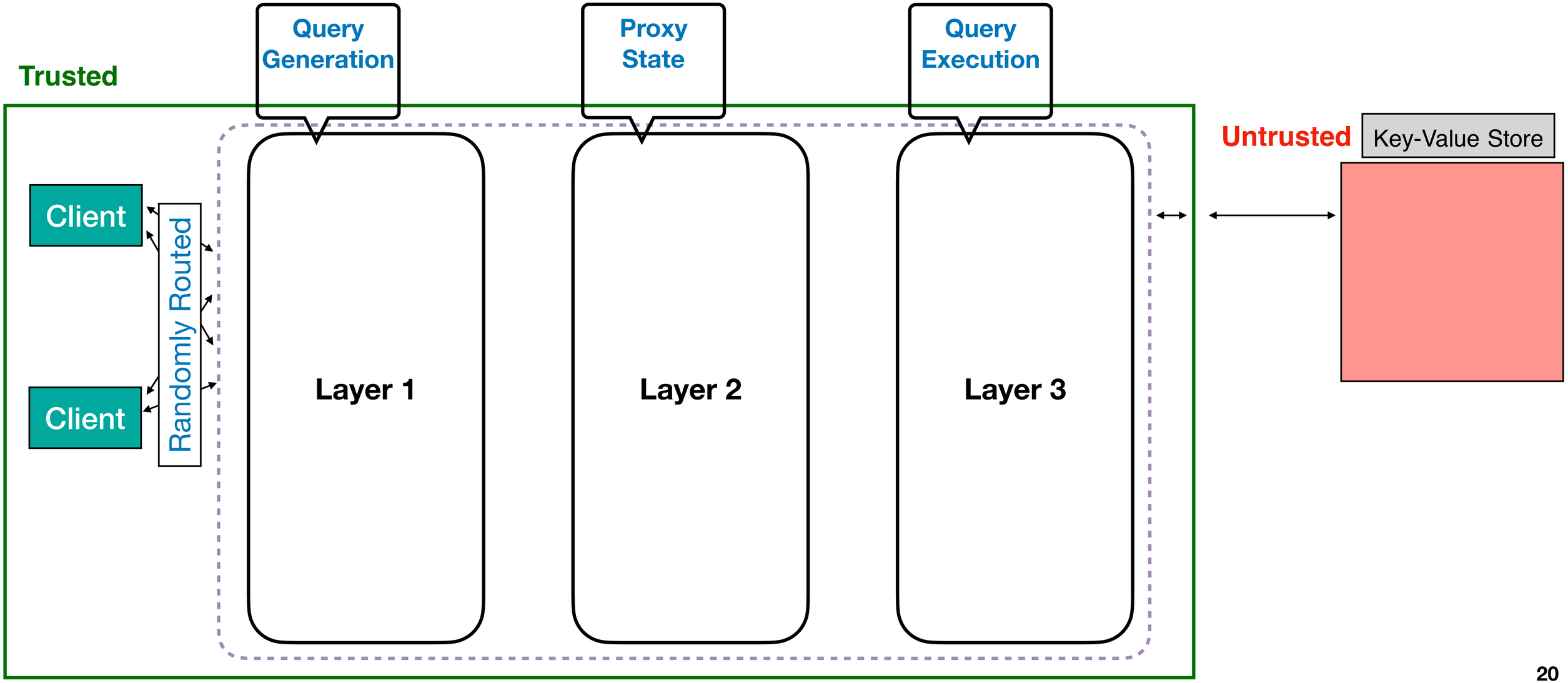
Shortstack Design Summary: Logical



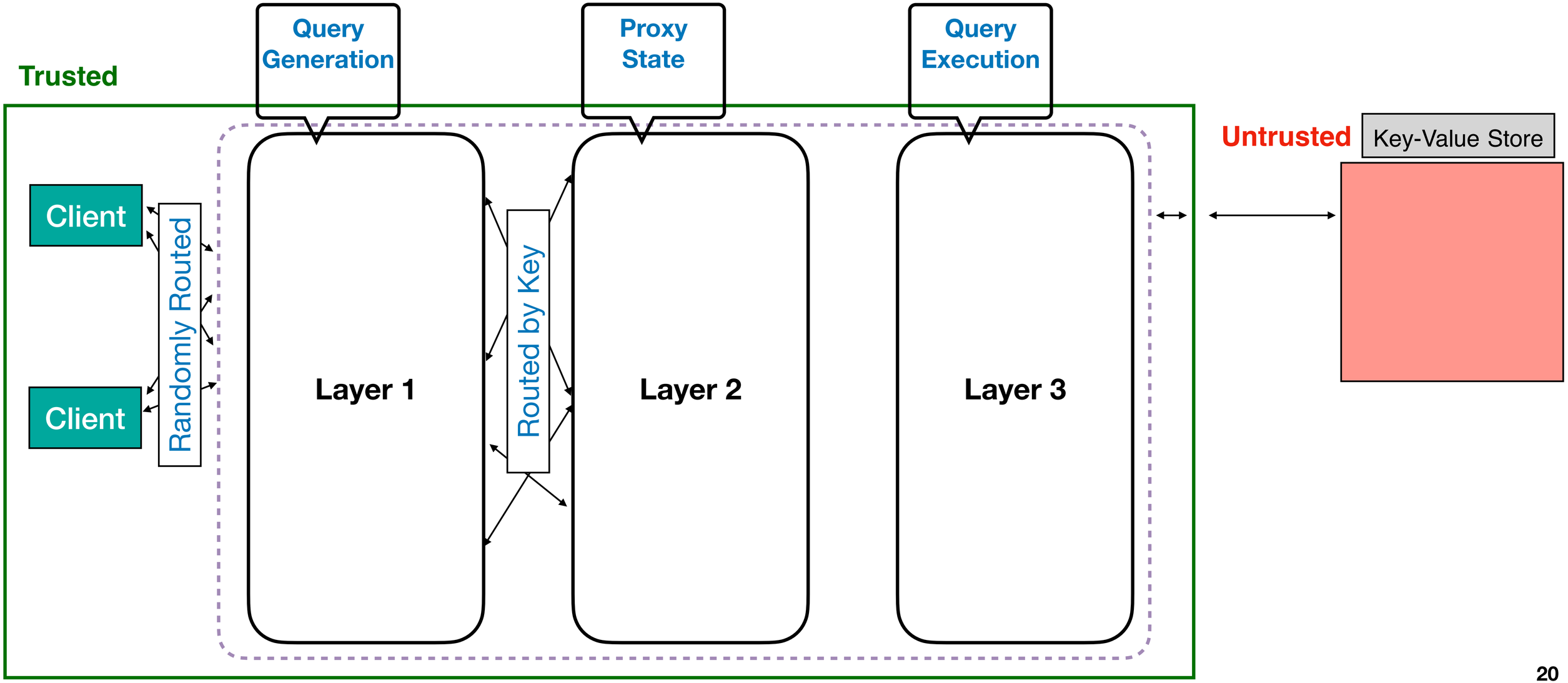
Shortstack Design Summary: Logical



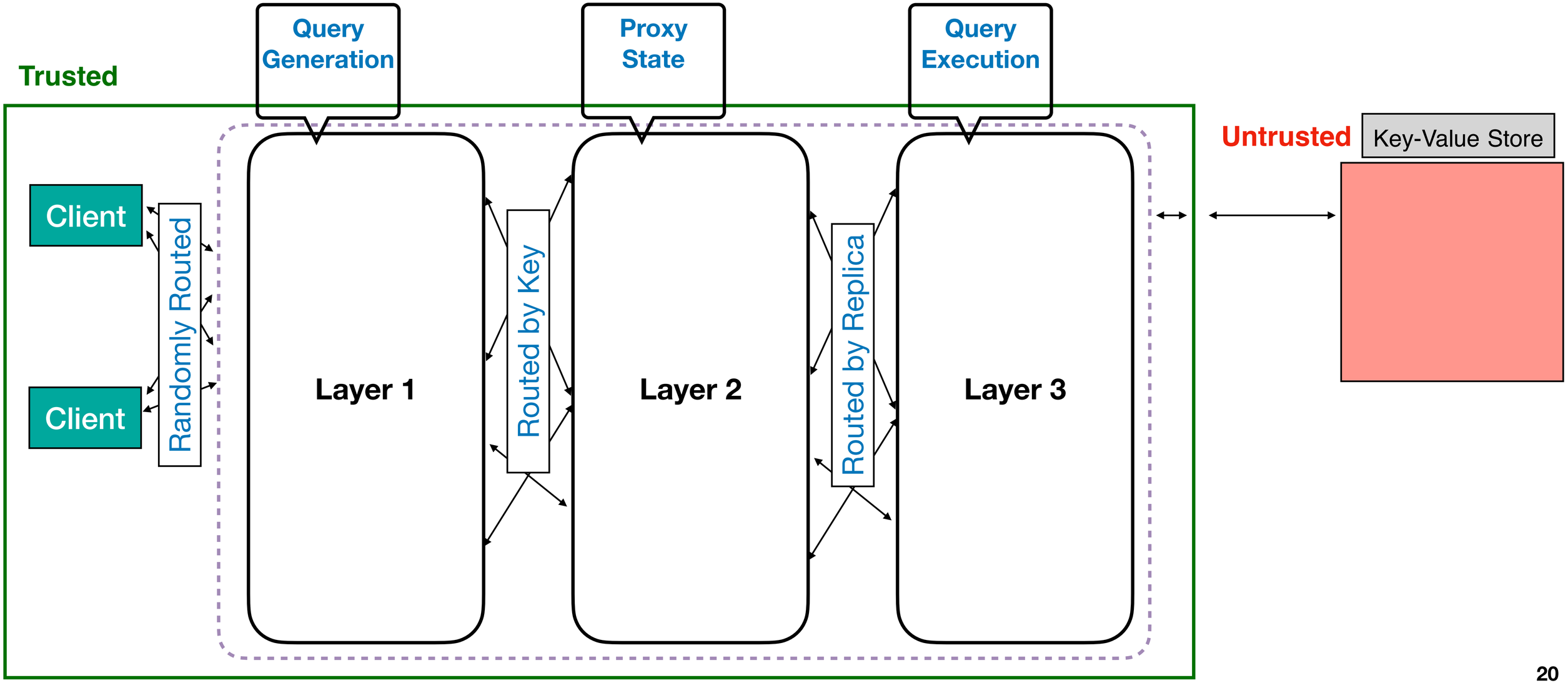
Shortstack Design Summary: Logical



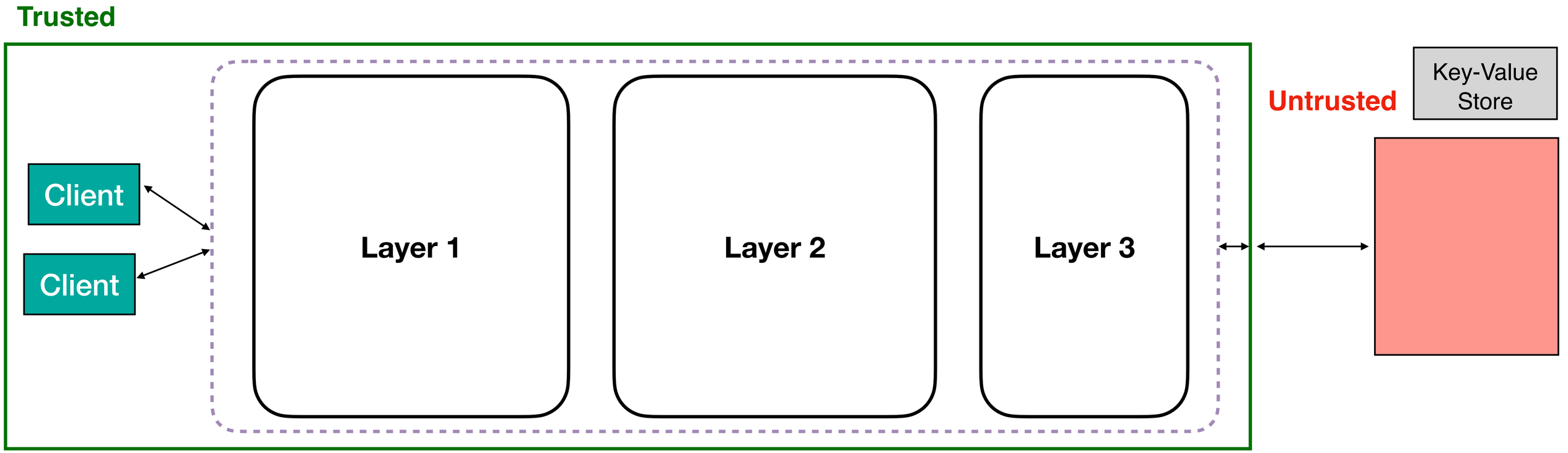
Shortstack Design Summary: Logical



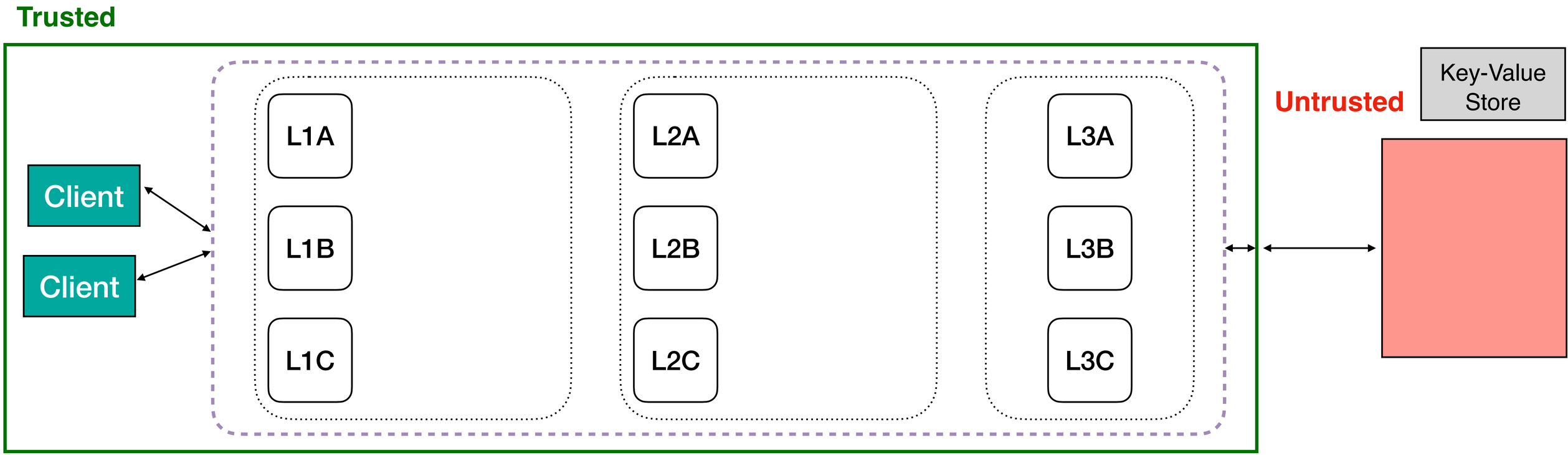
Shortstack Design Summary: Logical



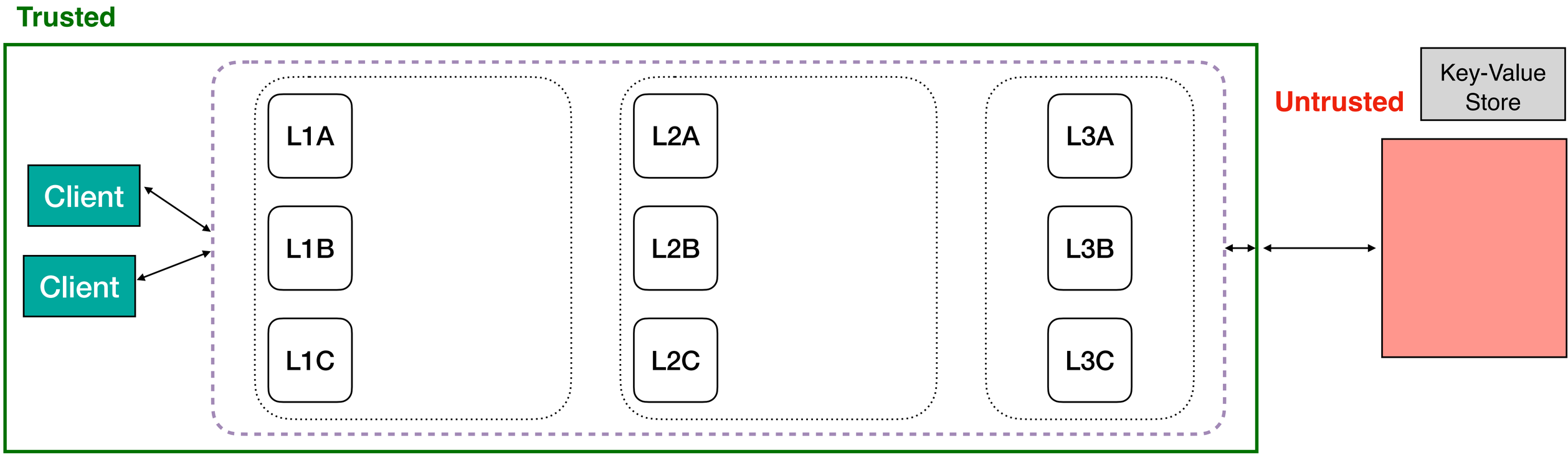
Shortstack Fault Tolerance



Shortstack Fault Tolerance

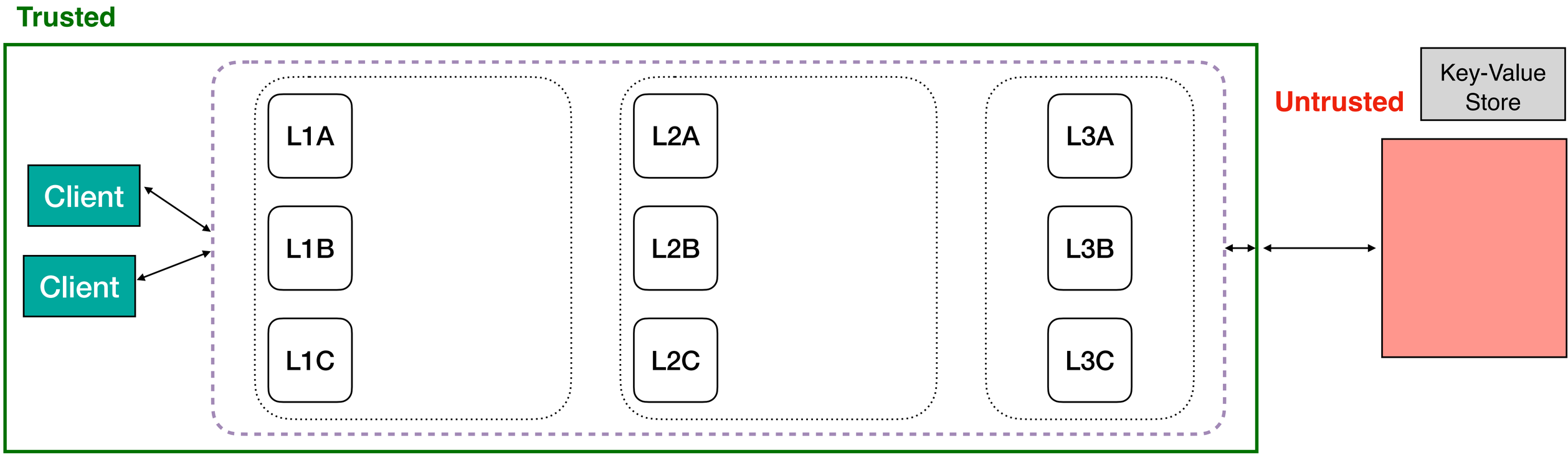


Shortstack Fault Tolerance



L3 failures: Redistribute requests across remaining servers

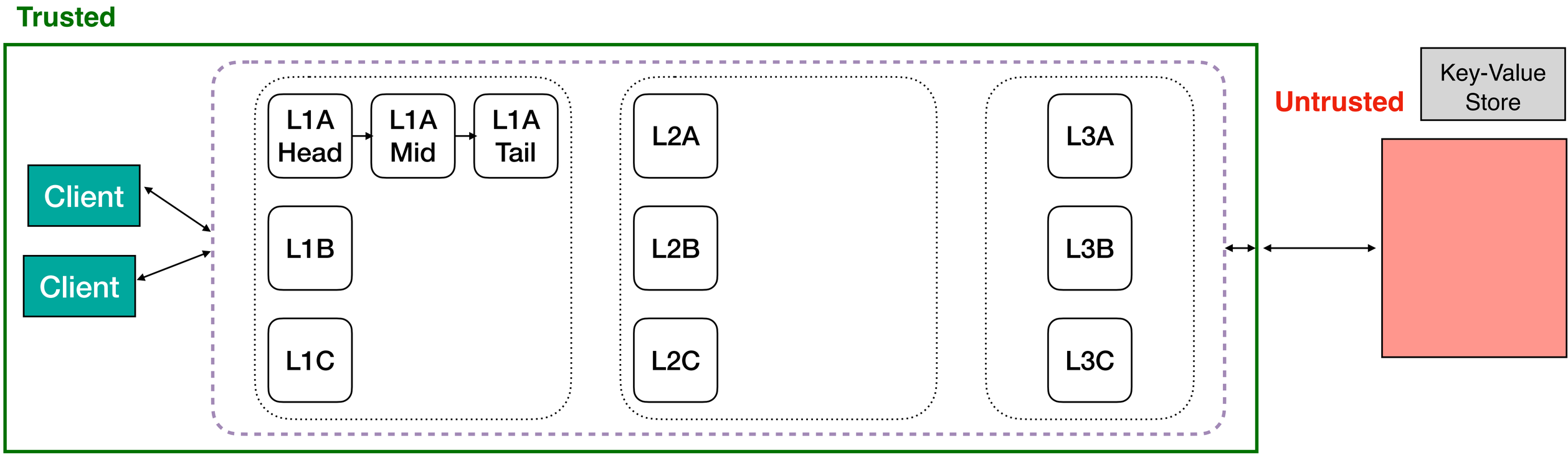
Shortstack Fault Tolerance



L3 failures: Redistribute requests across remaining servers

L1 and L2 failures handled through chain replication

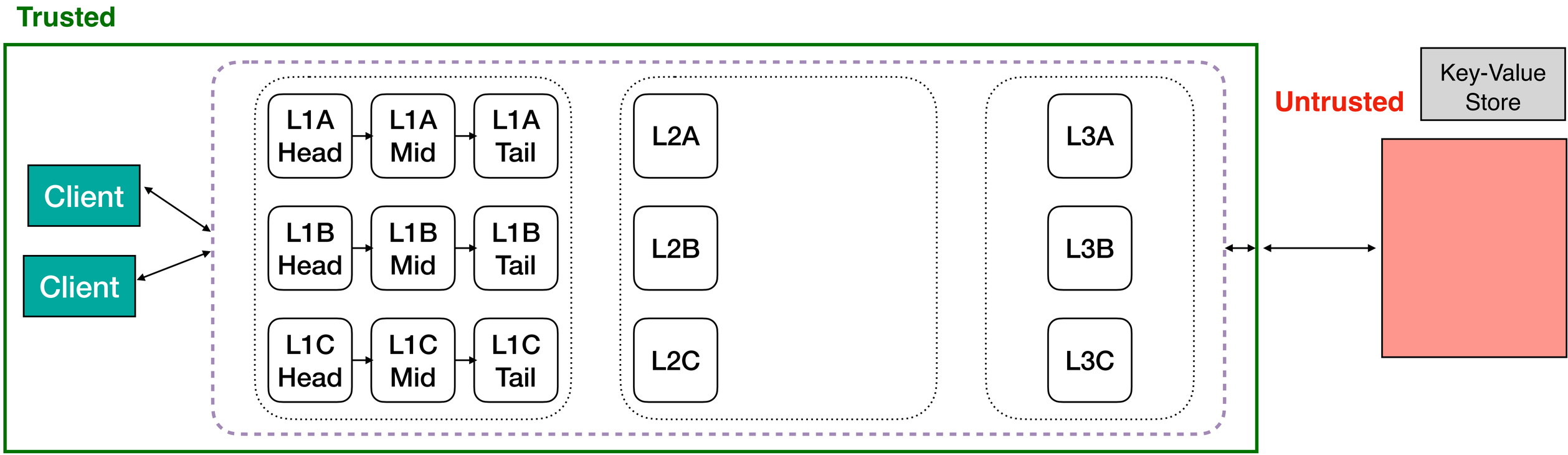
Shortstack Fault Tolerance



L3 failures: Redistribute requests across remaining servers

L1 and L2 failures handled through chain replication

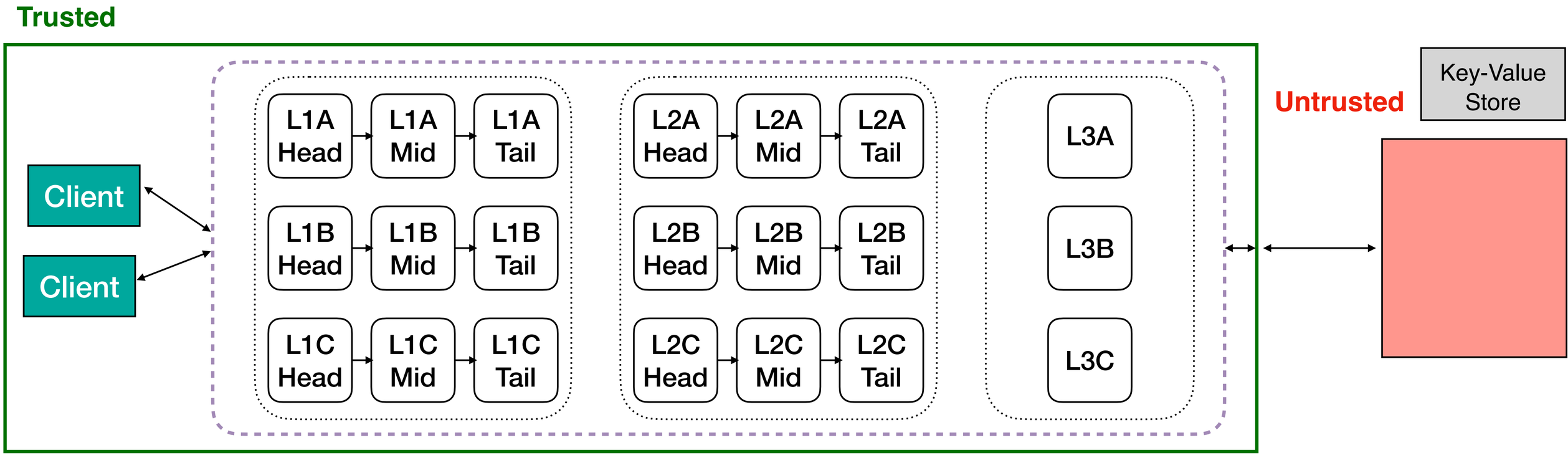
Shortstack Fault Tolerance



L3 failures: Redistribute requests across remaining servers

L1 and L2 failures handled through chain replication

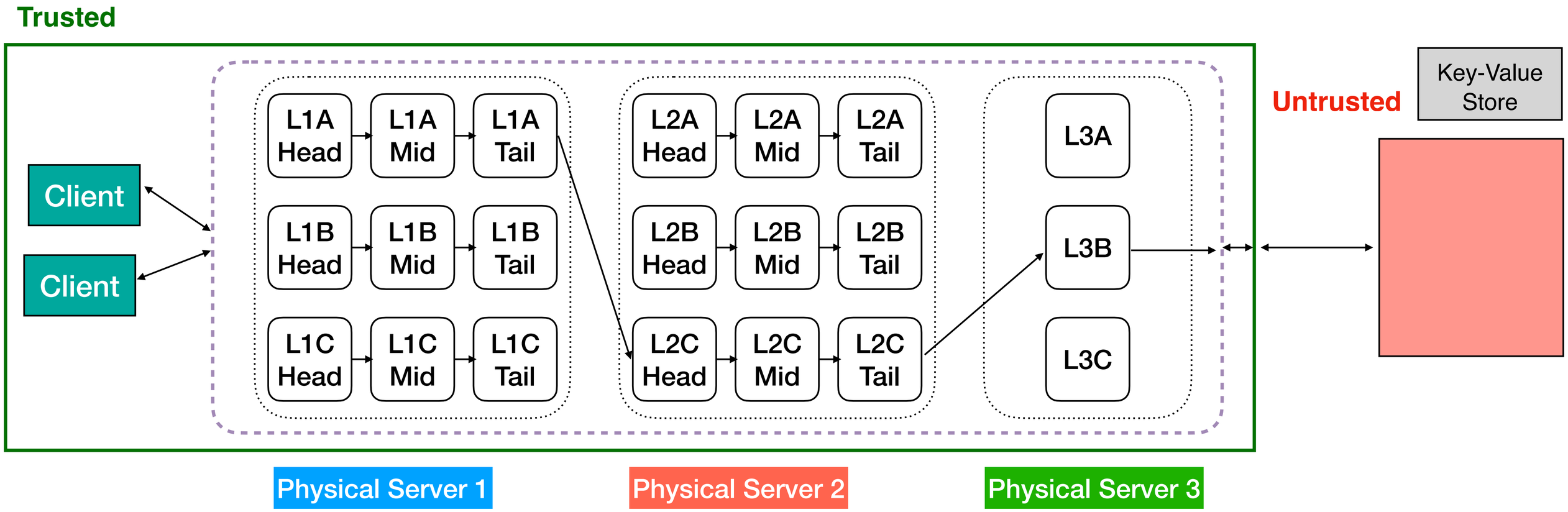
Shortstack Fault Tolerance



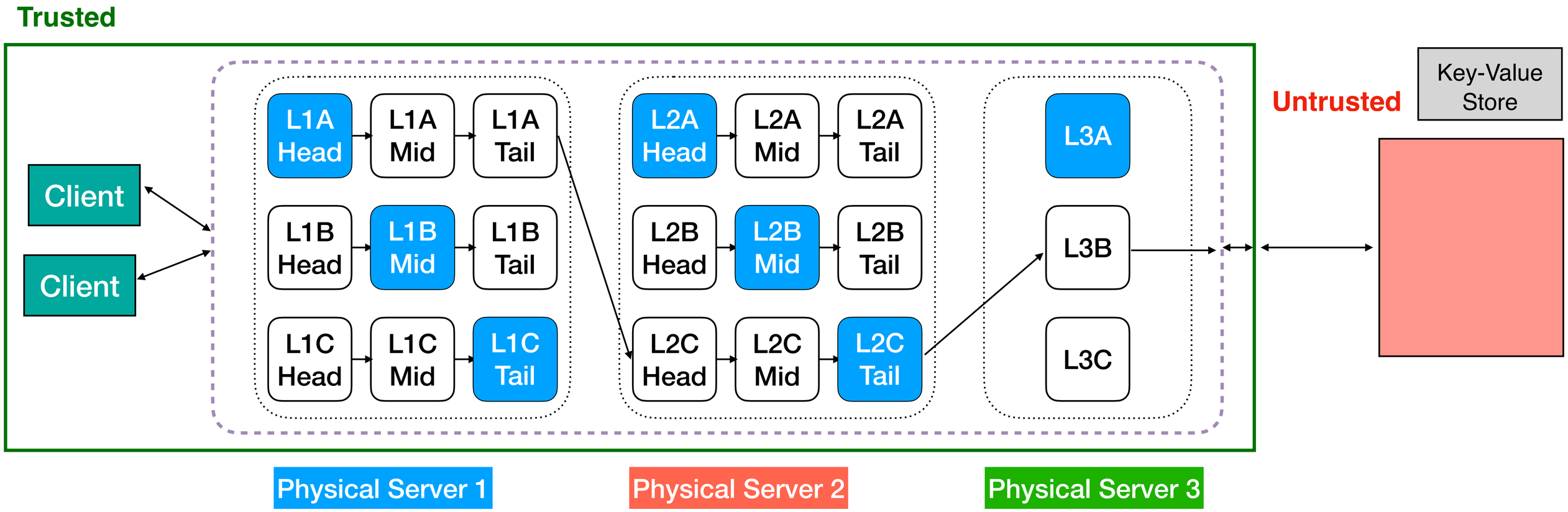
L3 failures: Redistribute requests across remaining servers

L1 and L2 failures handled through chain replication

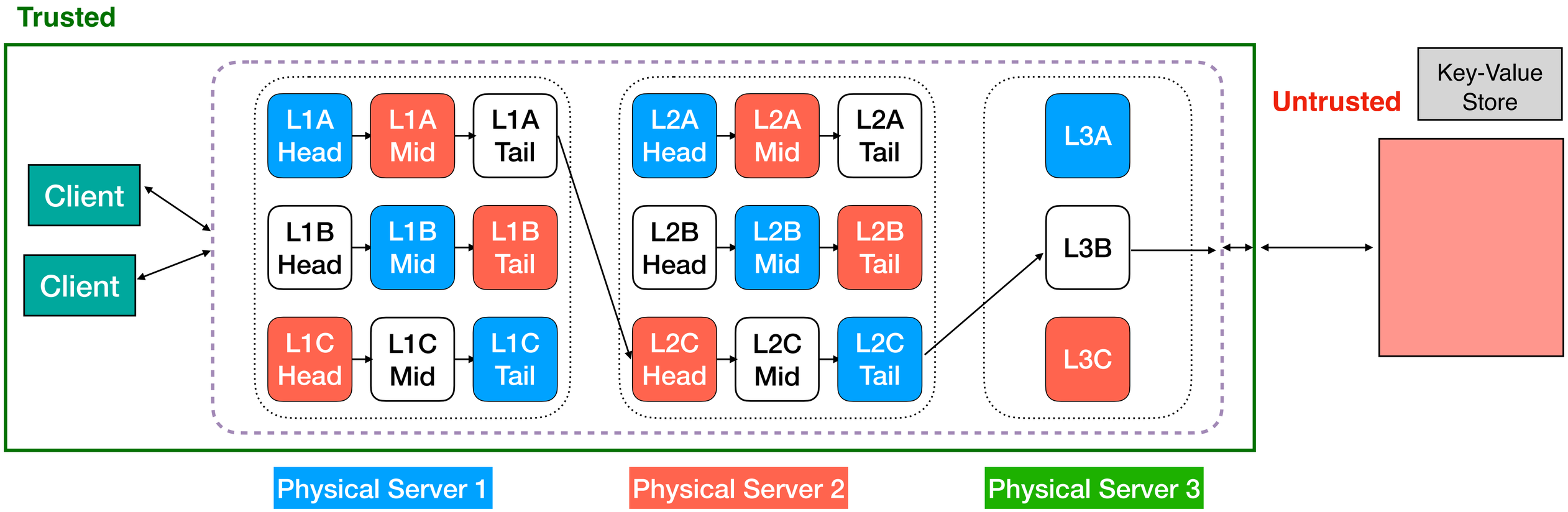
Logical→Physical Design



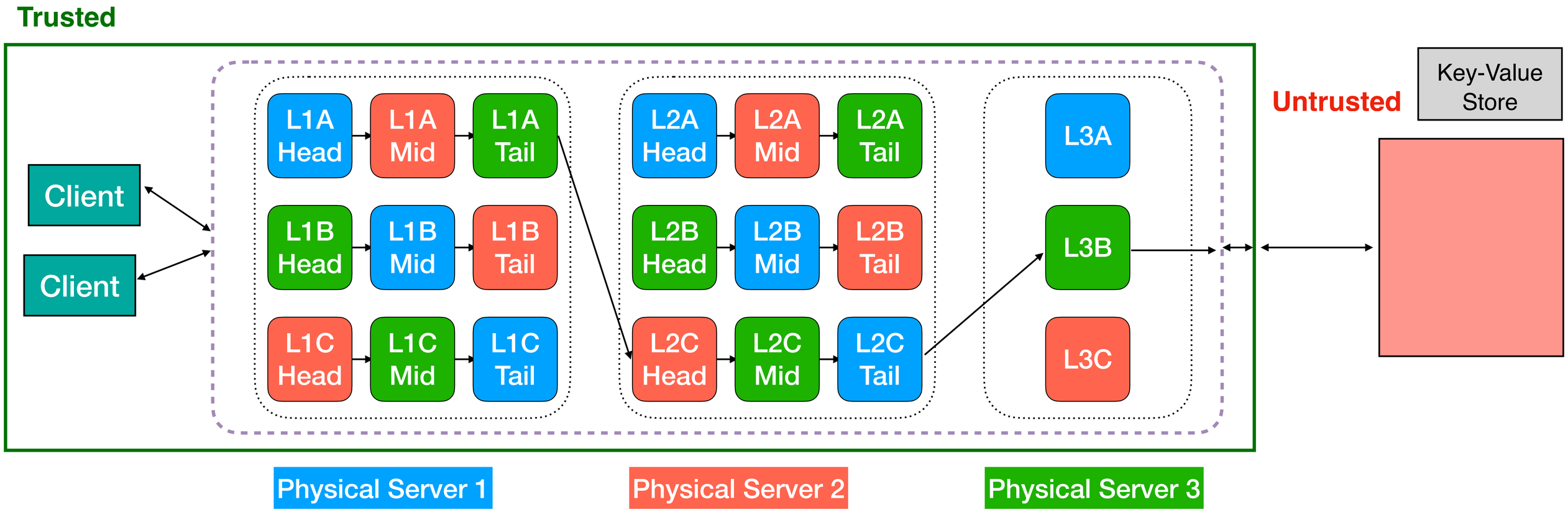
Logical→Physical Design



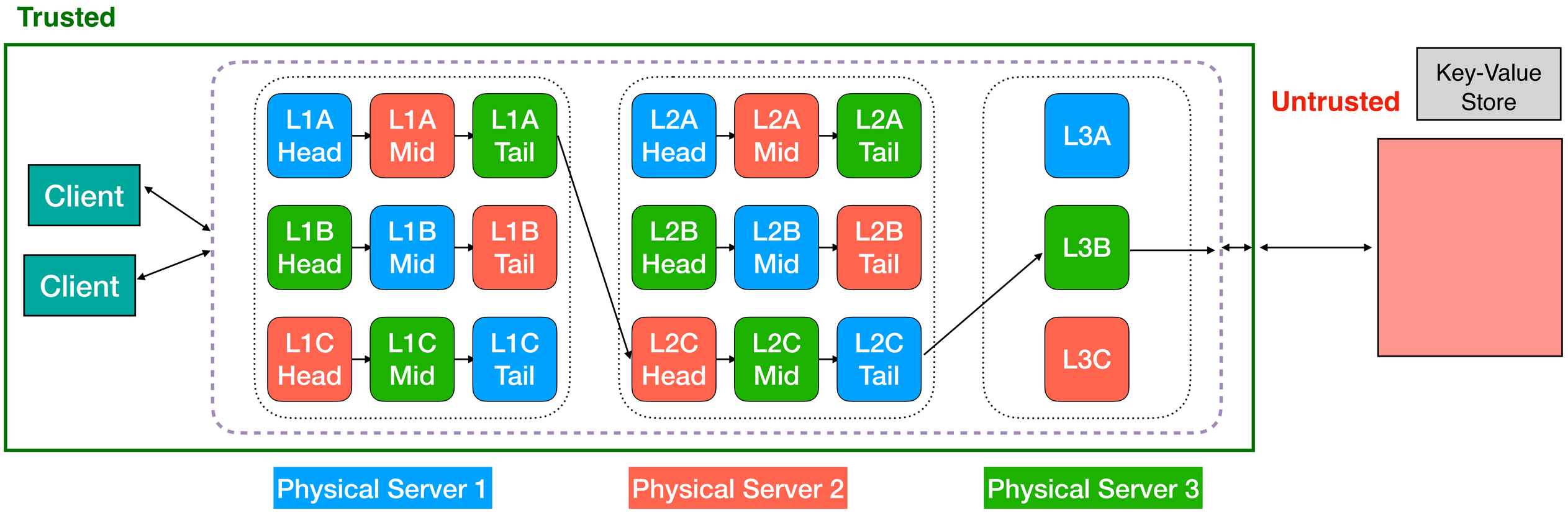
Logical→Physical Design



Logical→Physical Design

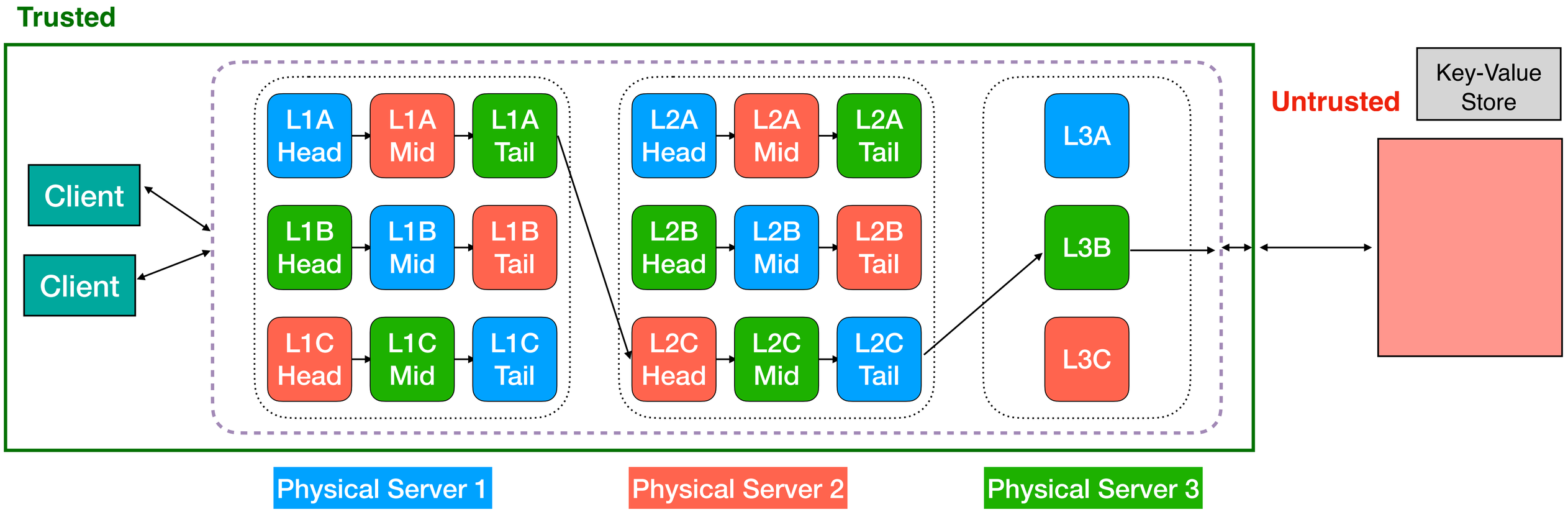


Logical→Physical Design



Shortstack needs 3 physical servers for handling 2 failures

Logical→Physical Design



Shortstack needs 3 physical servers for handling 2 failures

Shortstack needs *f+1* physical servers for handling *f* failures

This work

Challenges with Centralized Oblivious Data Access Systems
1. Insecure or unavailable during failures
2. Scalability bottleneck

Design of Shortstack:
Distributed, Fault-tolerant, Oblivious Data Access System

Security model:
Enables formal study of Distributed, Fault-tolerant, Oblivious Data Access systems

Security Model

Security Model

Existing security models do not capture failures

Security Model

Existing security models do not capture failures

New, General, Security Model

Security Model

Existing security models do not capture failures

New, General, Security Model

Powerful adversary that cause
arbitrary failures (bounded number)
at **arbitrary times**

Security Model

Existing security models do not capture failures

New, General, Security Model

Powerful adversary that cause **arbitrary failures** (bounded number) at **arbitrary times**

+

Output distribution independent of input distribution (Oblivious data access guarantee)

Security Model

Existing security models do not capture failures

New, General, Security Model

Powerful adversary that cause **arbitrary failures** (bounded number) at **arbitrary times**

+

Output distribution independent of input distribution (Oblivious data access guarantee)

(Formal definitions and proof of Shortstack security in paper)

Evaluation

Evaluation

- **ShortStack end-to-end implementation open-sourced**
    - Requires no modifications to the server side

Evaluation

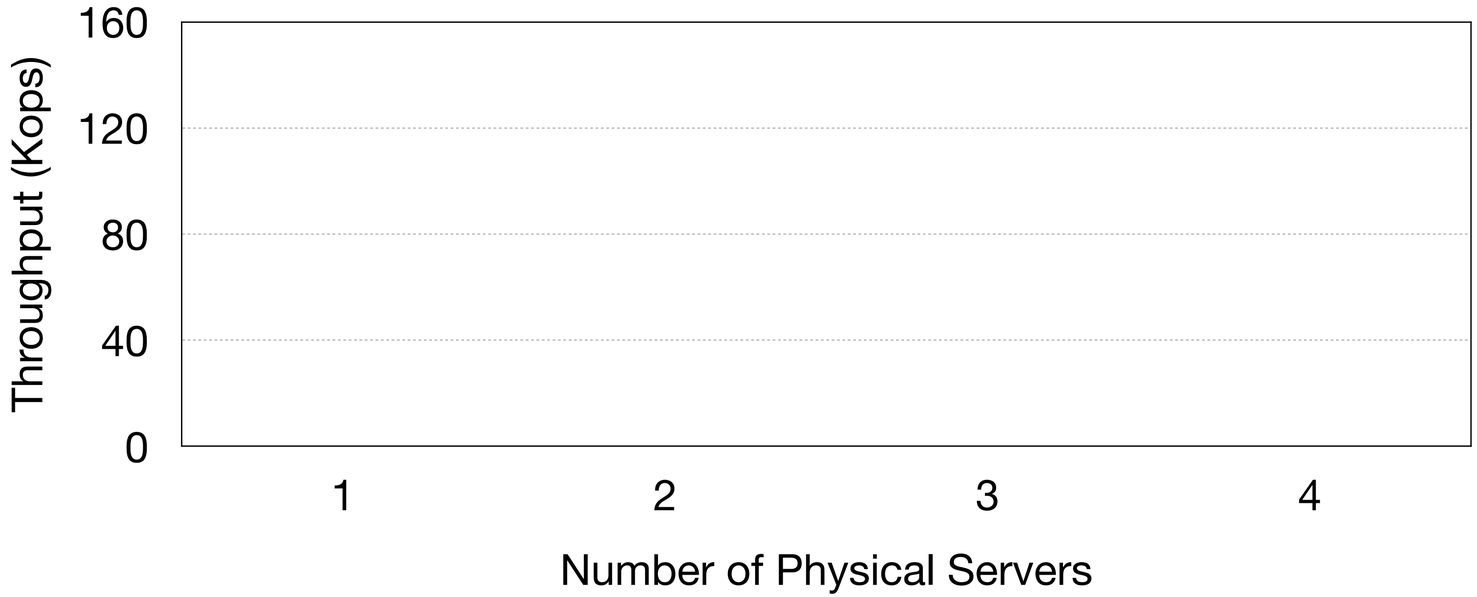
- **ShortStack end-to-end implementation open-sourced**
    - Requires no modifications to the server side
- **Evaluation on EC2 with Redis as the key-value store**
    - YCSB-A and YCSB-C workloads (more details in the paper)

Evaluation

- **ShortStack end-to-end implementation open-sourced**
    - Requires no modifications to the server side
- **Evaluation on EC2 with Redis as the key-value store**
    - YCSB-A and YCSB-C workloads (more details in the paper)
- **Goal: Demonstrating <u>throughput scalability with number of physical servers</u>**

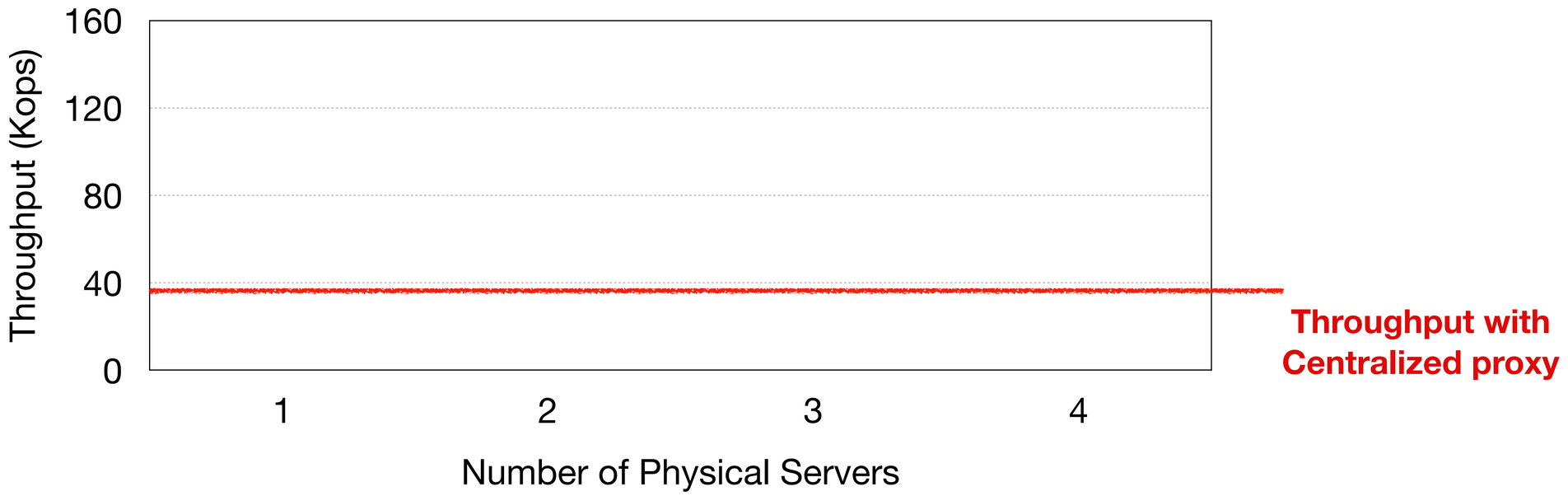
Evaluation

- **ShortStack end-to-end implementation open-sourced**
  - Requires no modifications to the server side
- **Evaluation on EC2 with Redis as the key-value store**
  - YCSB-A and YCSB-C workloads (more details in the paper)
- **Goal: Demonstrating <u>throughput scalability with number of physical servers</u>**



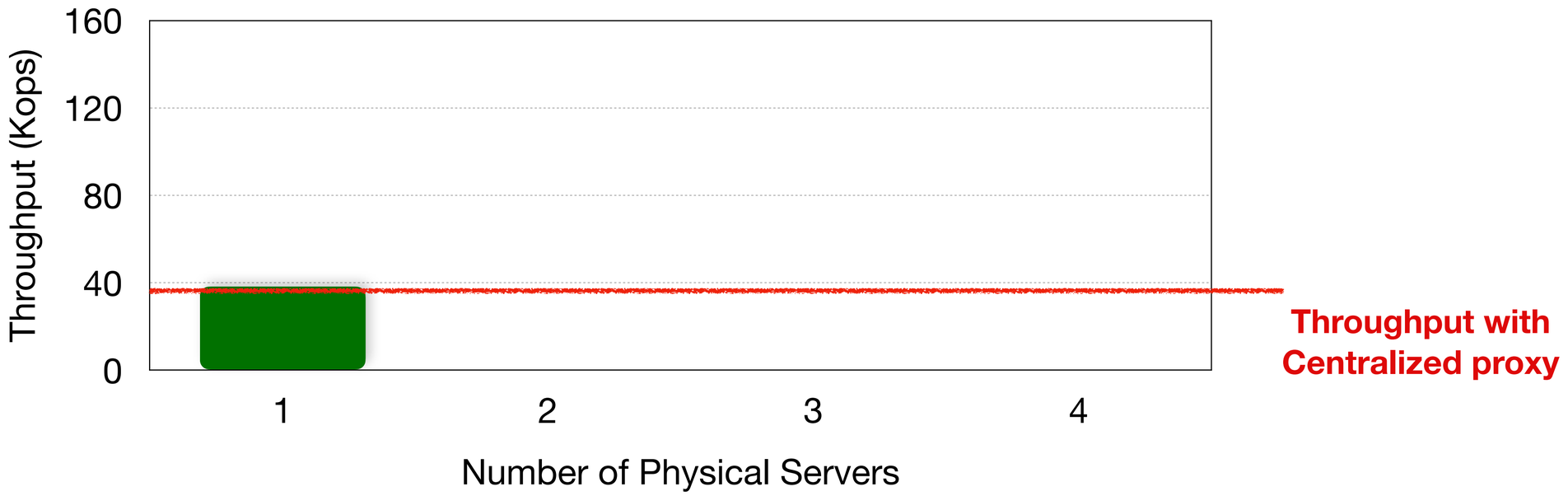
Evaluation

- **ShortStack end-to-end implementation open-sourced**
  - Requires no modifications to the server side
- **Evaluation on EC2 with Redis as the key-value store**
  - YCSB-A and YCSB-C workloads (more details in the paper)
- **Goal: Demonstrating <u>throughput scalability with number of physical servers</u>**



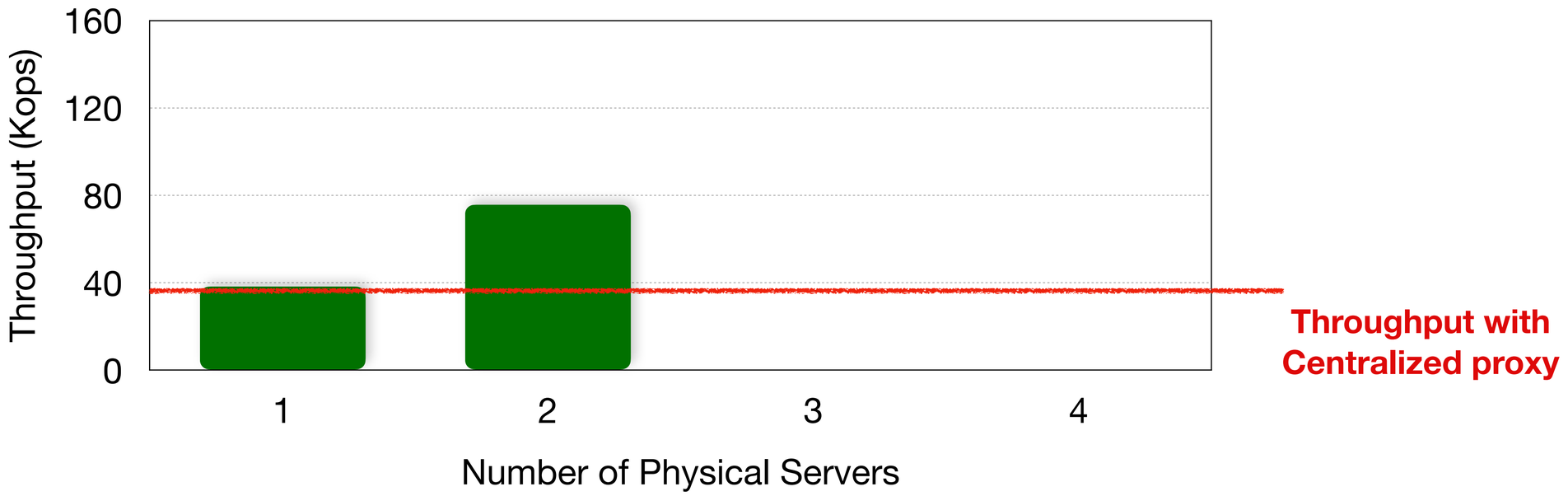
Evaluation

- **ShortStack end-to-end implementation open-sourced**
  - Requires no modifications to the server side
- **Evaluation on EC2 with Redis as the key-value store**
  - YCSB-A and YCSB-C workloads (more details in the paper)
- **Goal: Demonstrating <u>throughput scalability with number of physical servers</u>**



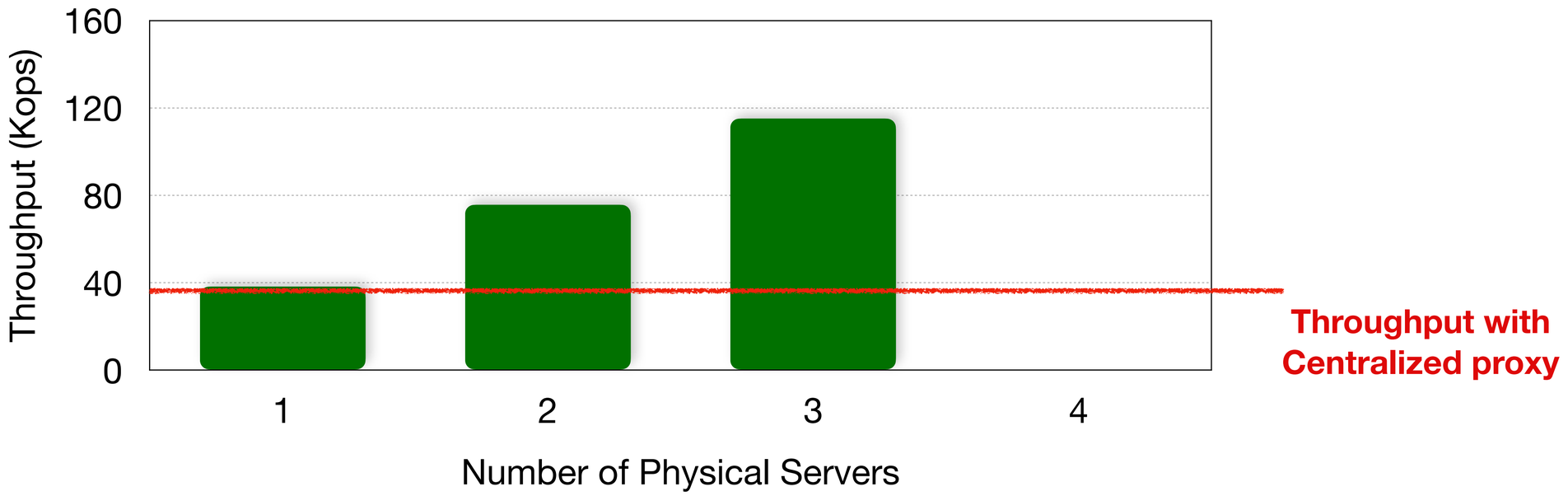
Evaluation

- **ShortStack end-to-end implementation open-sourced**
  - Requires no modifications to the server side
- **Evaluation on EC2 with Redis as the key-value store**
  - YCSB-A and YCSB-C workloads (more details in the paper)
- **Goal: Demonstrating <u>throughput scalability with number of physical servers</u>**



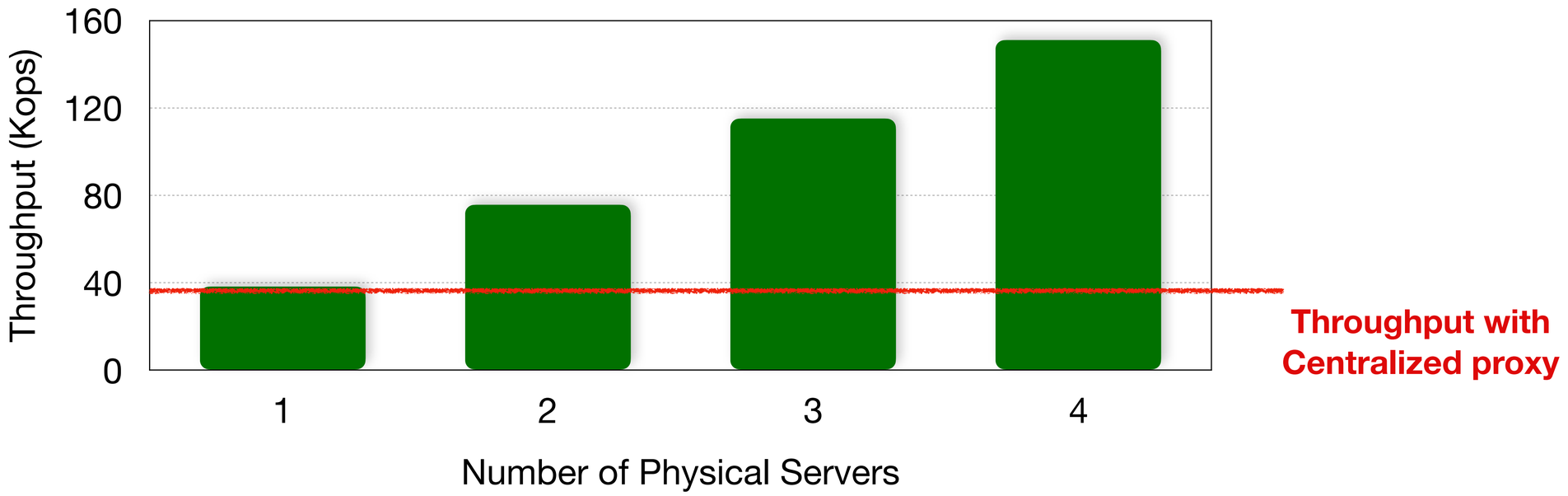
Evaluation

- **ShortStack end-to-end implementation open-sourced**
  - Requires no modifications to the server side
- **Evaluation on EC2 with Redis as the key-value store**
  - YCSB-A and YCSB-C workloads (more details in the paper)
- **Goal: Demonstrating <u>throughput scalability with number of physical servers</u>**



Evaluation

- **ShortStack end-to-end implementation open-sourced**
  - Requires no modifications to the server side
- **Evaluation on EC2 with Redis as the key-value store**
  - YCSB-A and YCSB-C workloads (more details in the paper)
- **Goal: Demonstrating <u>throughput scalability with number of physical servers</u>**



Evaluation

- **ShortStack end-to-end implementation open-sourced**
    - Requires no modifications to the server side
- **Evaluation on EC2 with Redis as the key-value store**
    - YCSB-A and YCSB-C workloads (more details in the paper)
- **Goal: Demonstrating throughput scalability with number of physical servers**
- Many additional results in the Paper…
    1. System Scalability
    2. Fault Tolerance
    3. Latency
    4. Bottlenecks in each layer
    5. Skewed workloads

Conclusion

Challenges with Centralized Oblivious Data Access Systems
1. Insecure or unavailable during failures
2. Scalability bottleneck

Design of Shortstack:
Distributed, Fault-tolerant, Oblivious Data Access System

Security model:
Enables formal study of Distributed, Fault-tolerant, Oblivious Data Access systems

https://github.com/pancake-security/shortstack