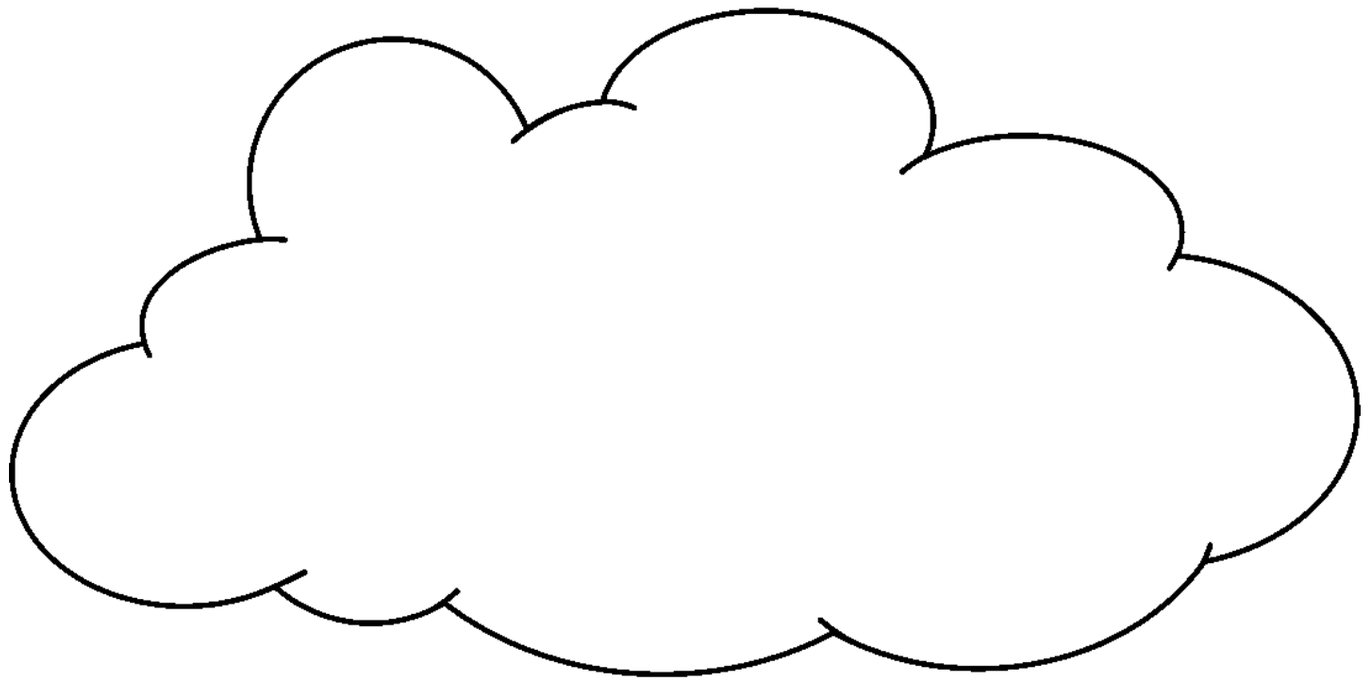
MiniCrypt: Reconciling Compression and Encryption in Big Data Stores

Wenting Zheng, Frank Li,
Raluca Ada Popa, Ion Stoica, Rachit Agarwal

A lot of sensitive data is collected

Sensitive data



client

cloud provider

A lot of sensitive data is collected

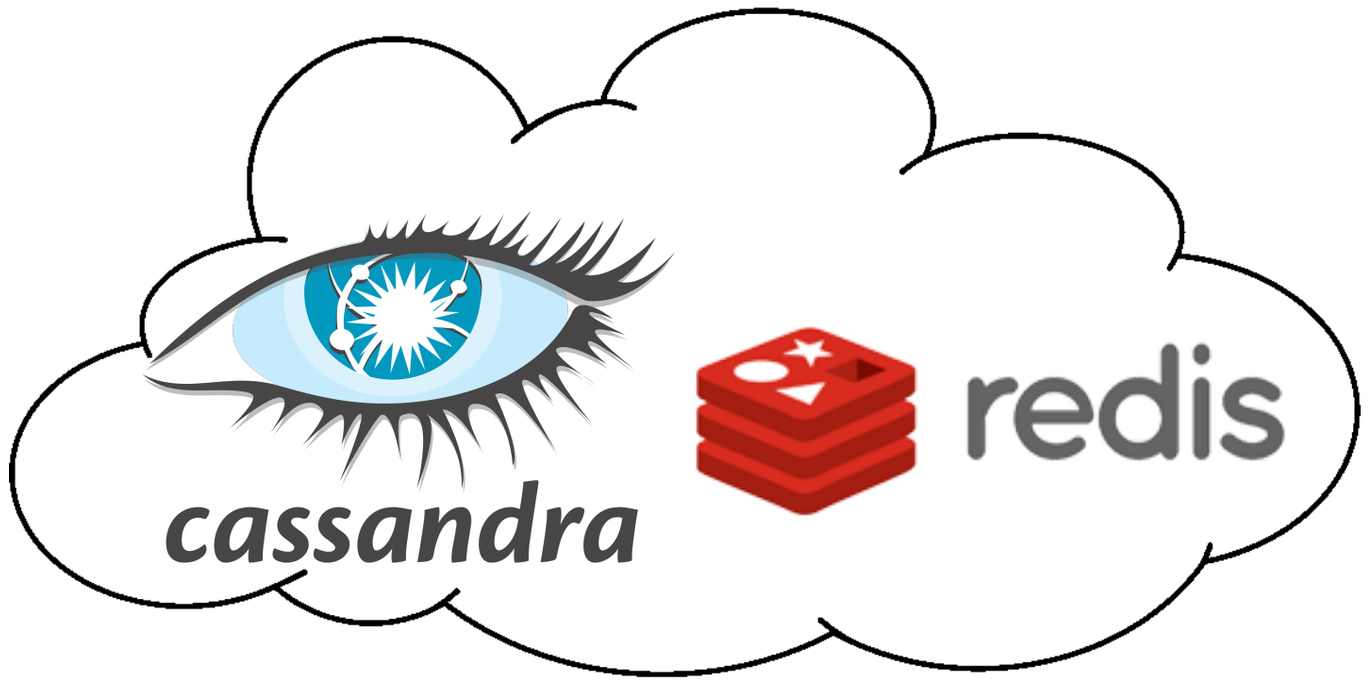
Sensitive data

client

cloud provider

A lot of sensitive data is collected

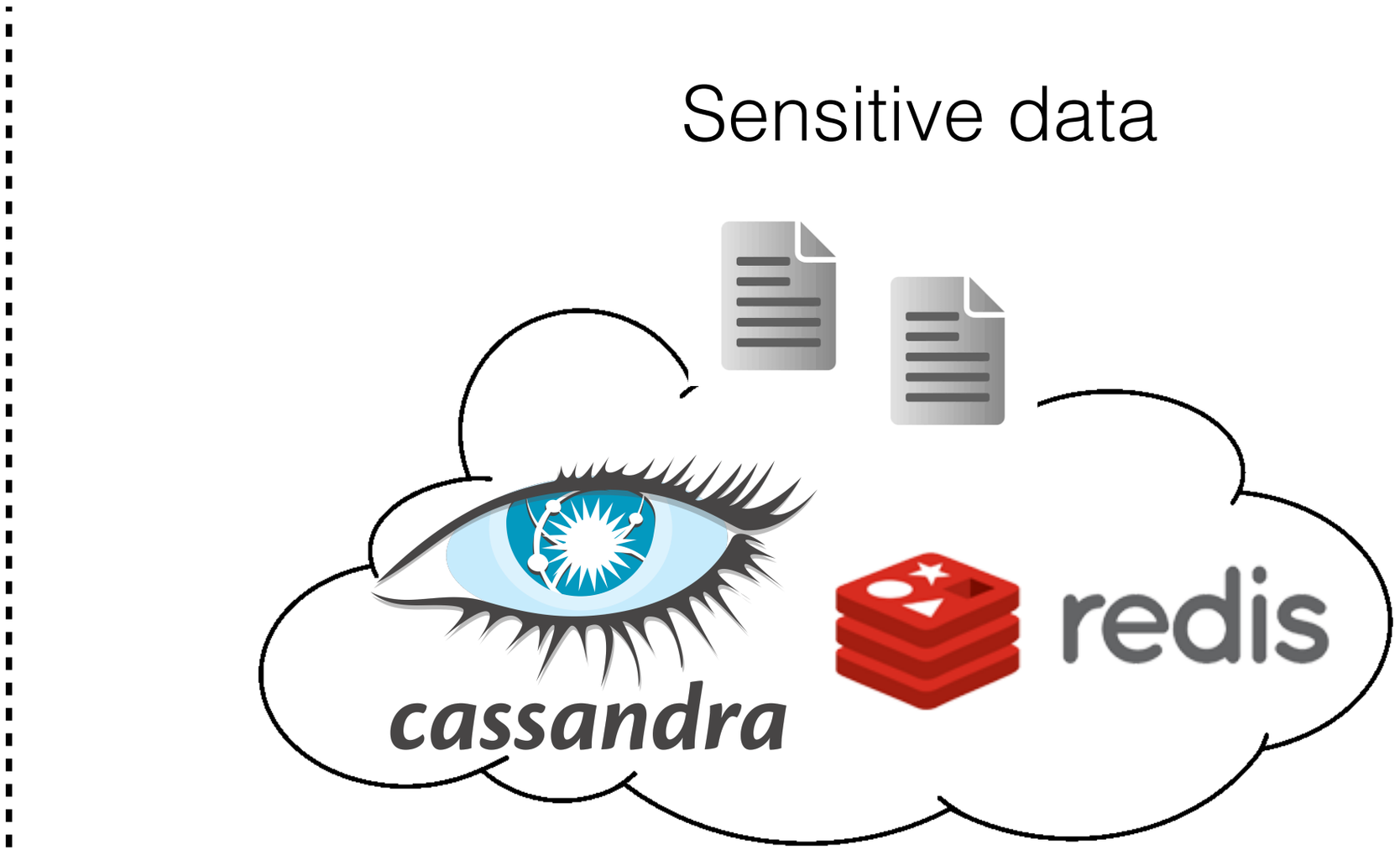
Sensitive data



client

cloud provider

A lot of sensitive data is collected



client

cloud provider

Existing systems

Existing systems

Systems	Compression	Encryption

Existing systems

Systems	Compression	Encryption
Cassandra [LM10], MongoDB, Succinct [AKS15]		

Existing systems

Systems	Compression	Encryption
Cassandra [LM10], MongoDB, Succinct [AKS15]		

Existing systems

Systems	Compression	Encryption
Cassandra [LM10], MongoDB, Succinct [AKS15]	✓	✗
CryptDB [PRZB11], Plutus [KRSWF03], Sirius [GSMB03], etc.		

Existing systems

Systems	Compression	Encryption
Cassandra [LM10], MongoDB, Succinct [AKS15]	✓	✗
CryptDB [PRZB11], Plutus [KRSWF03], Sirius [GSMB03], etc.	✗	✓

How to combine compression and encryption?

MiniCrypt

MiniCrypt

Provides *both compression and encryption* for key-value stores

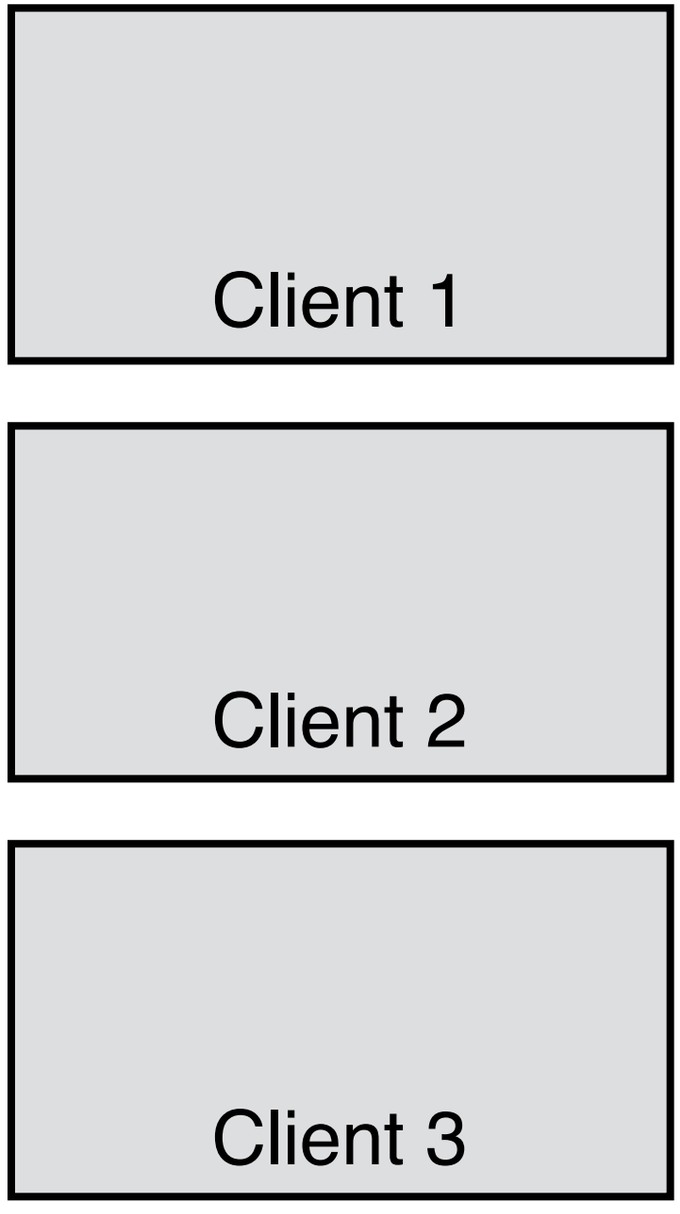
MiniCrypt

Provides *both* *compression* *and* *encryption* for key-value stores

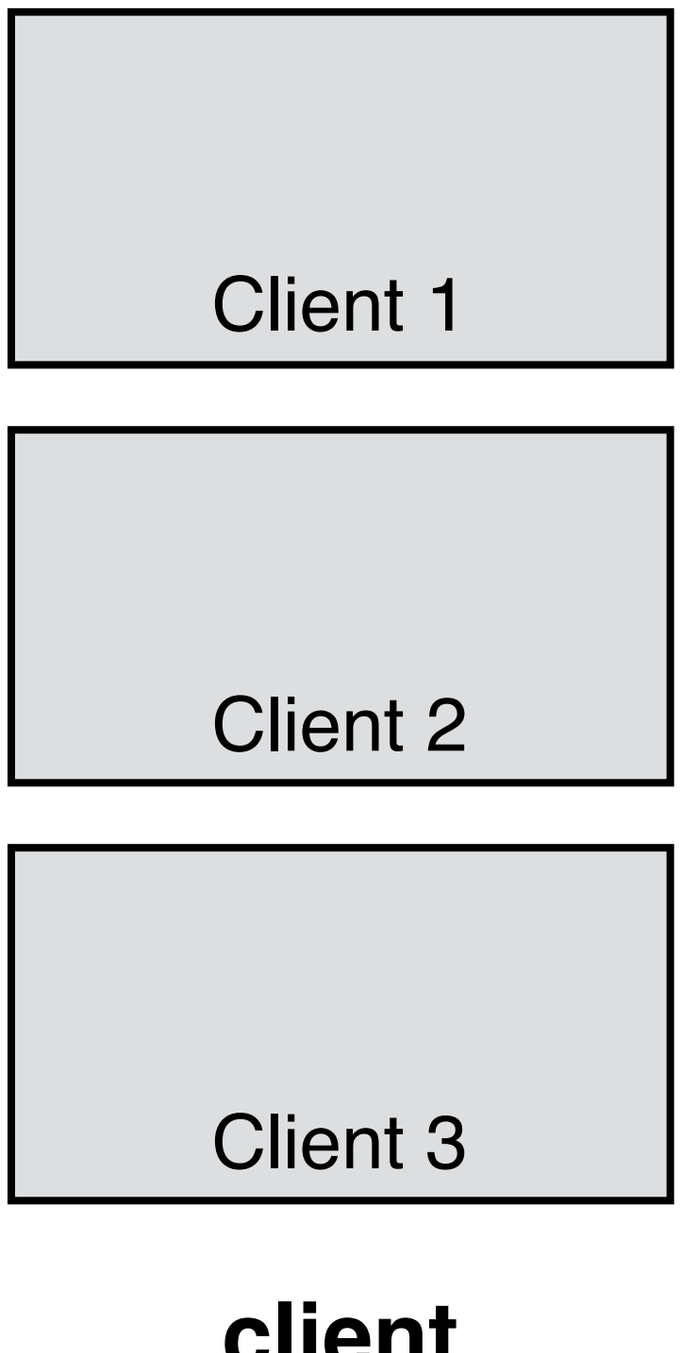
Works on top of *existing* key-value stores without modification

System setup

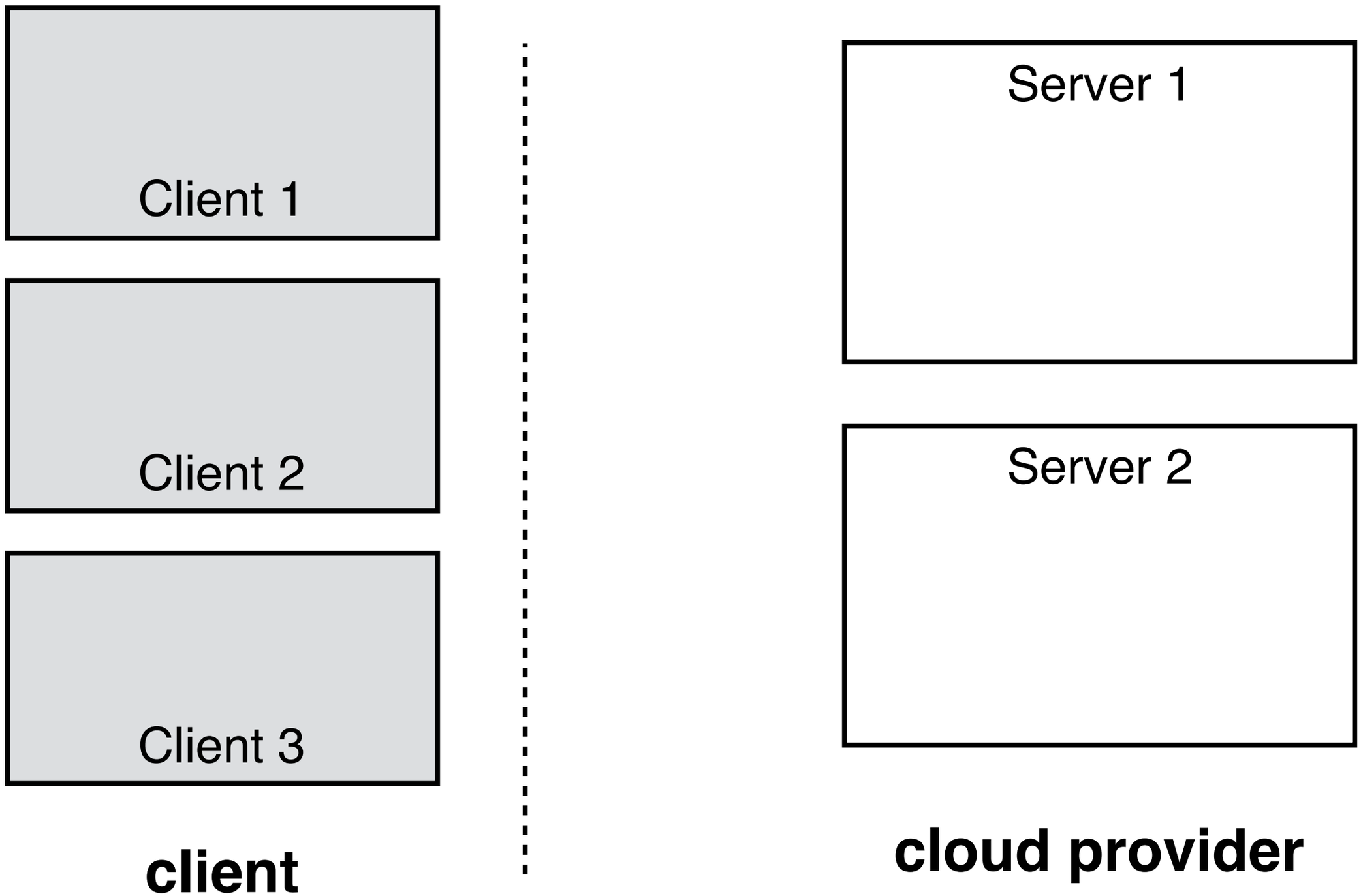
System setup



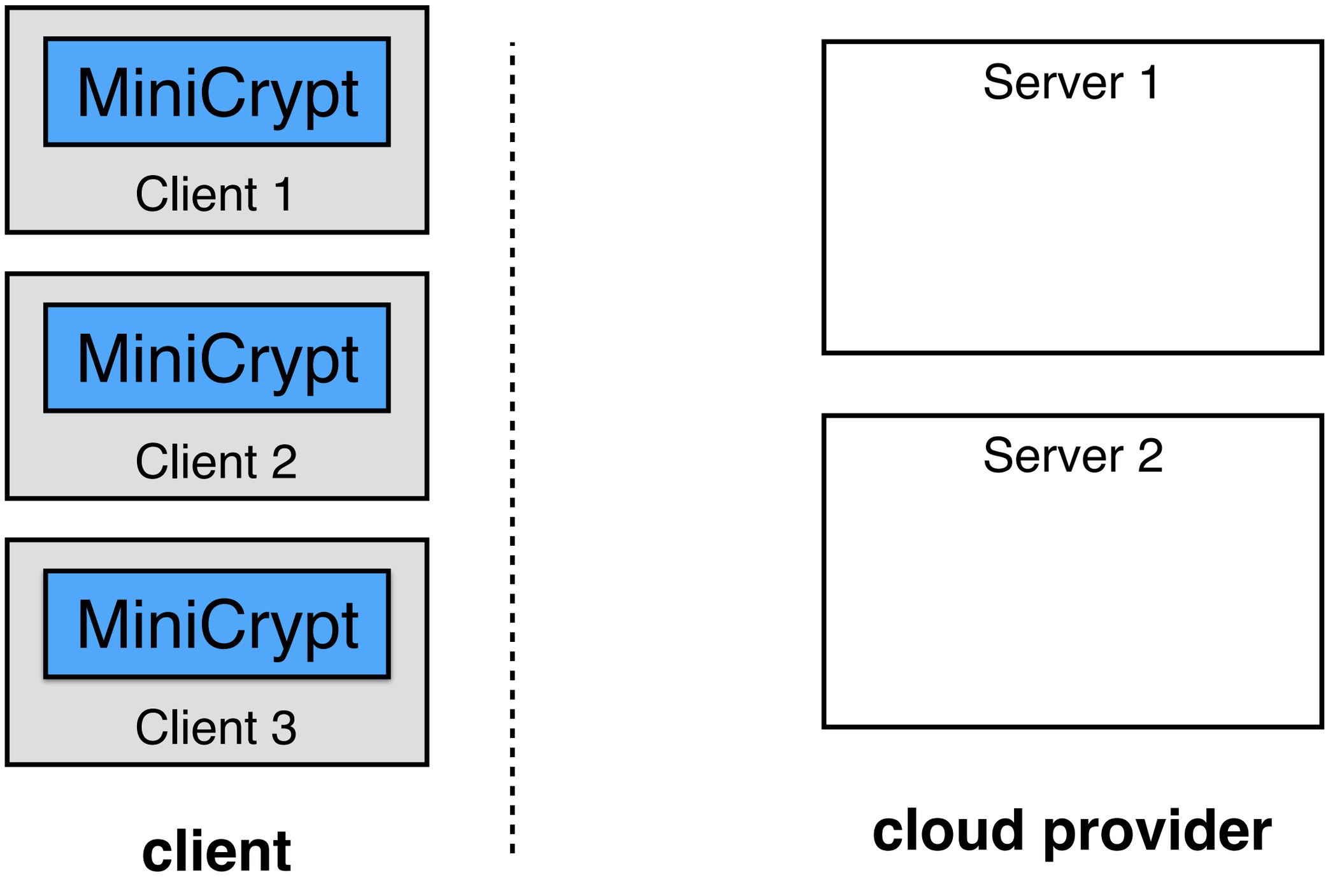
System setup



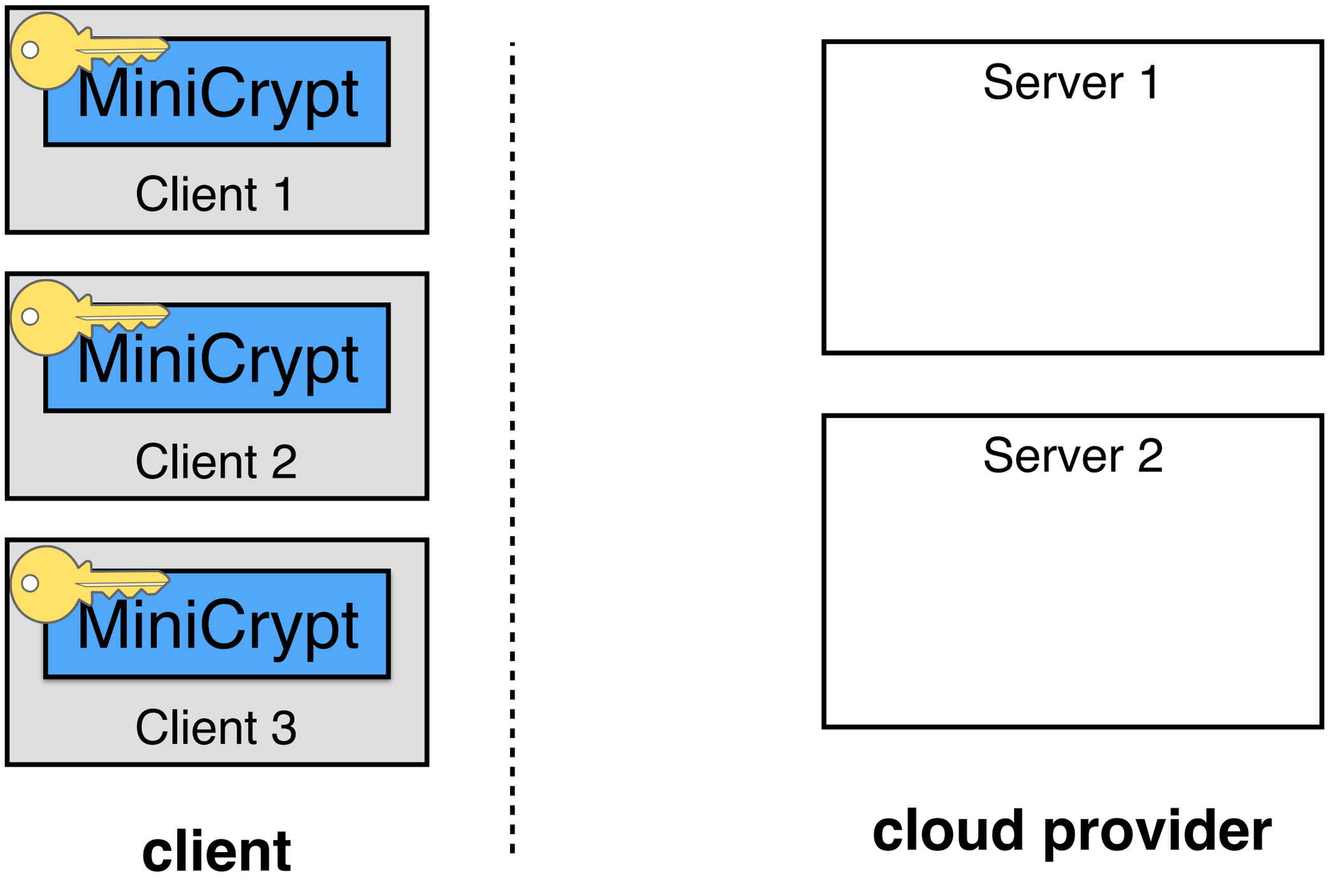
System setup



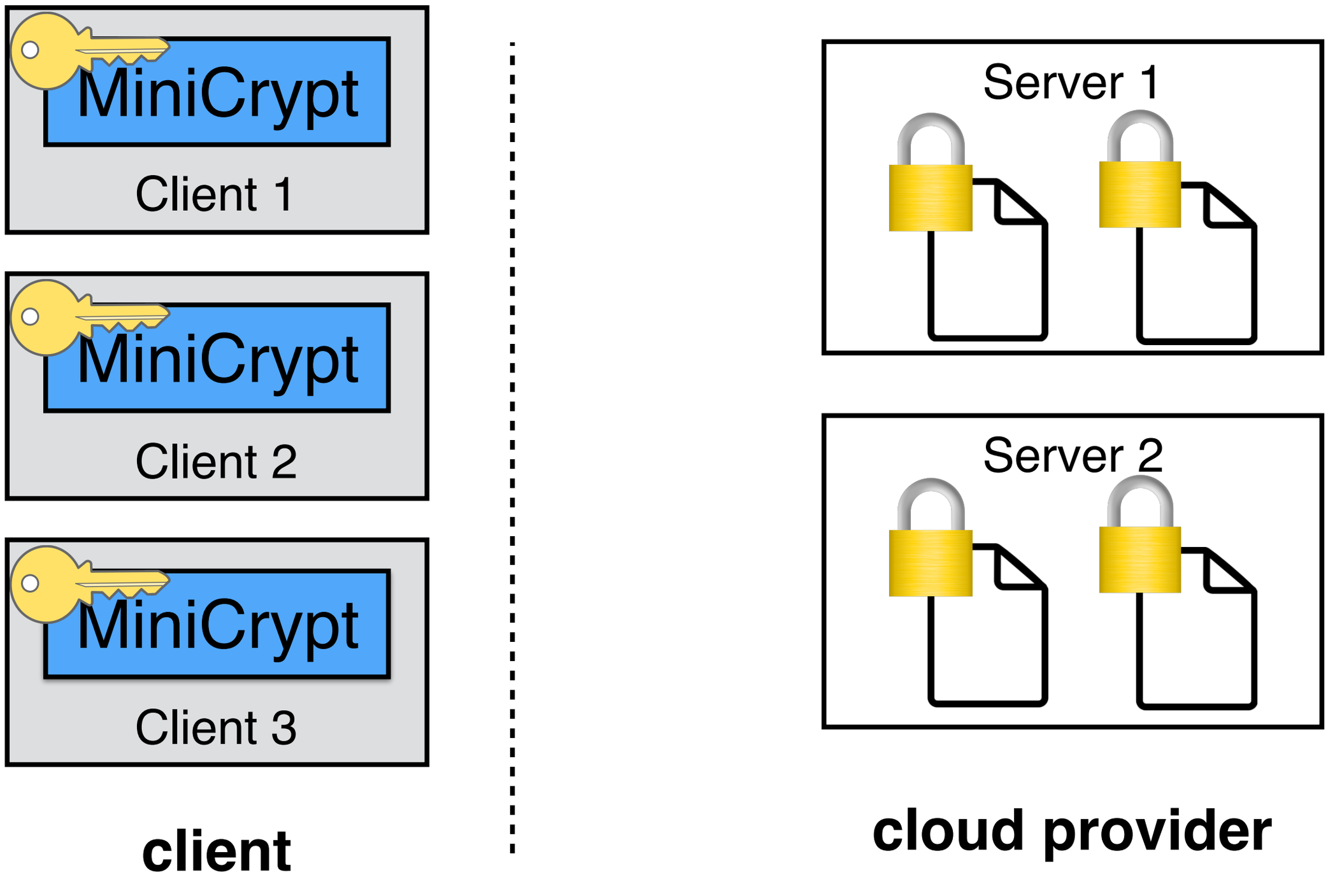
System setup



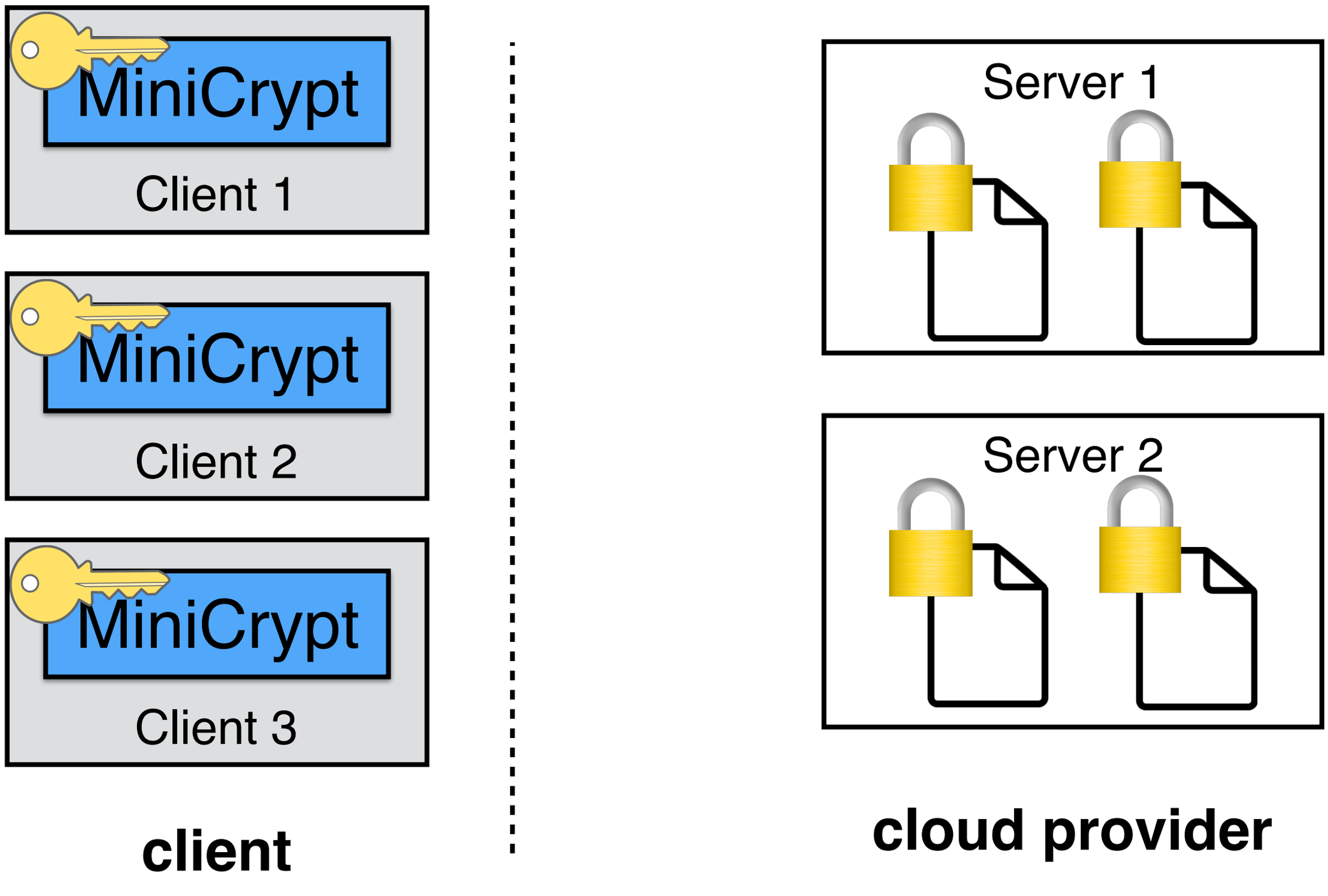
System setup



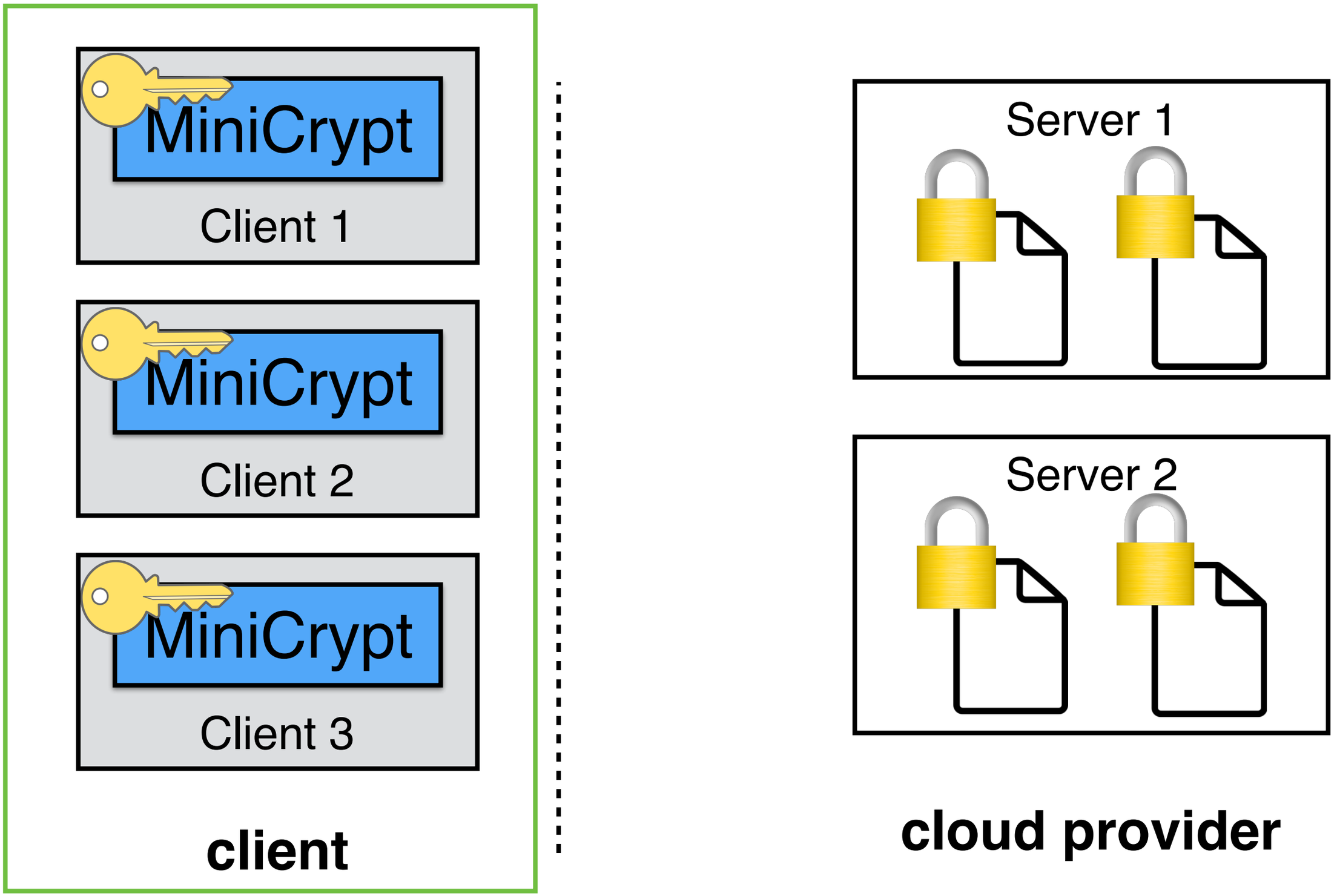
System setup



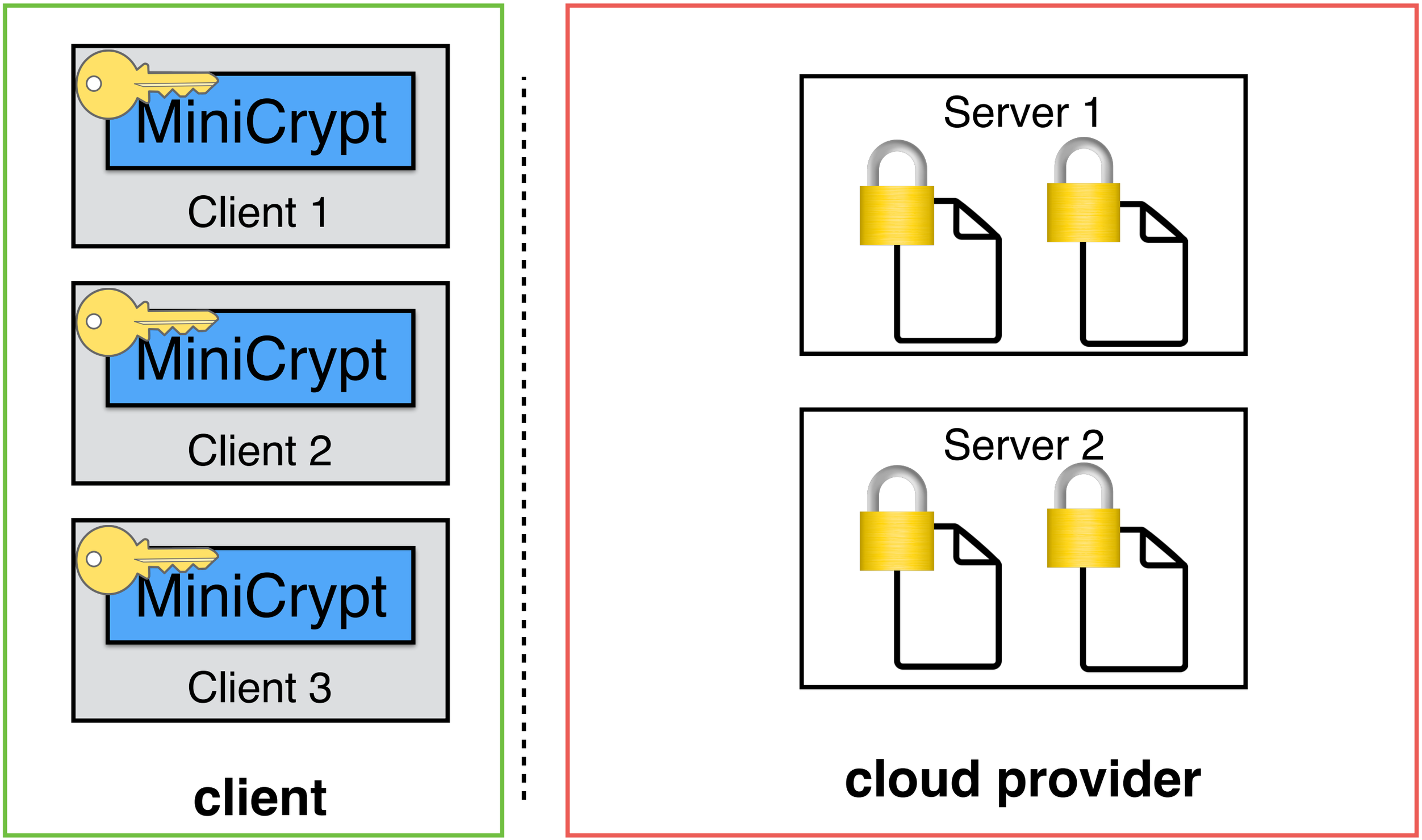
Threat model



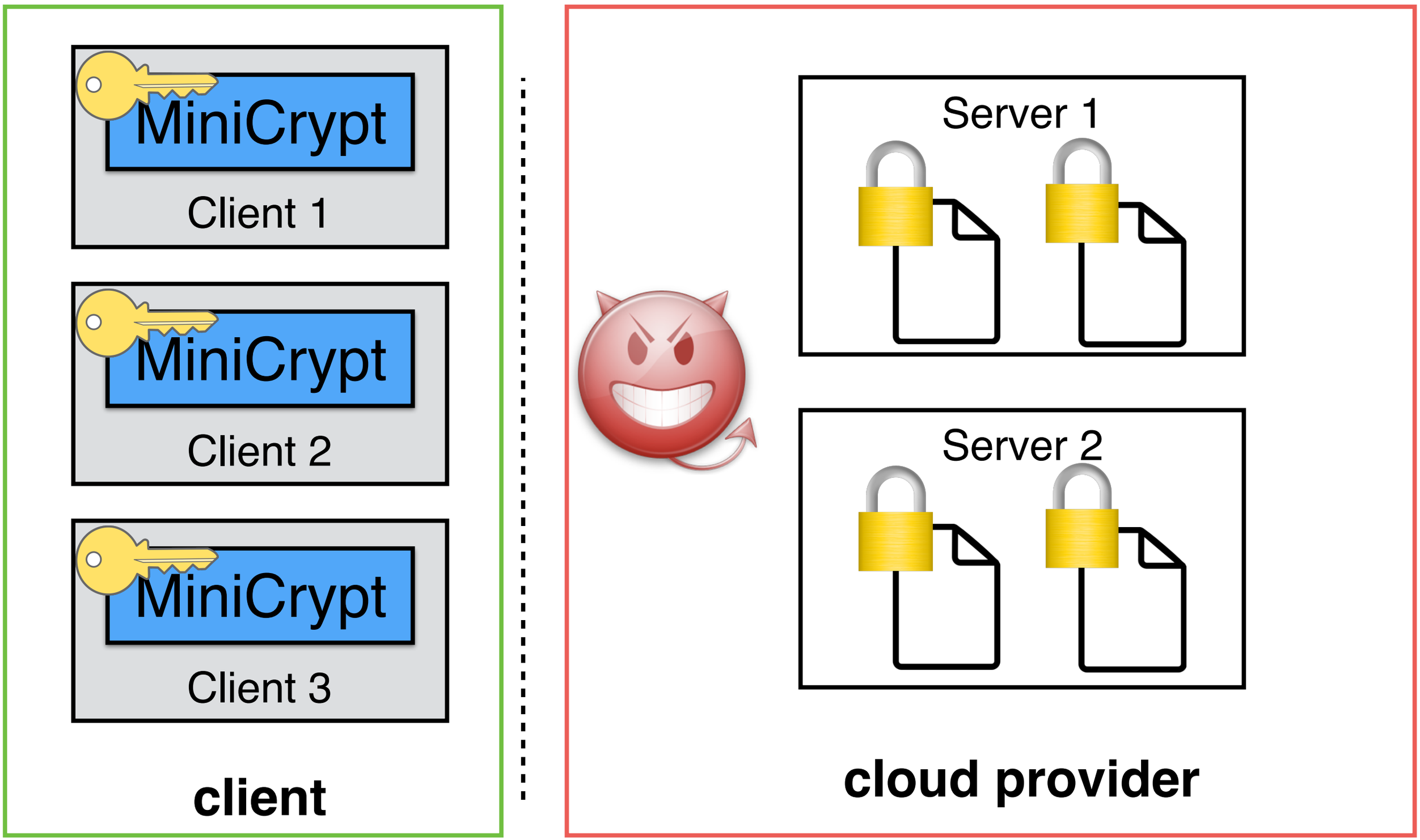
Threat model



Threat model



Threat model



Encryption before compression?

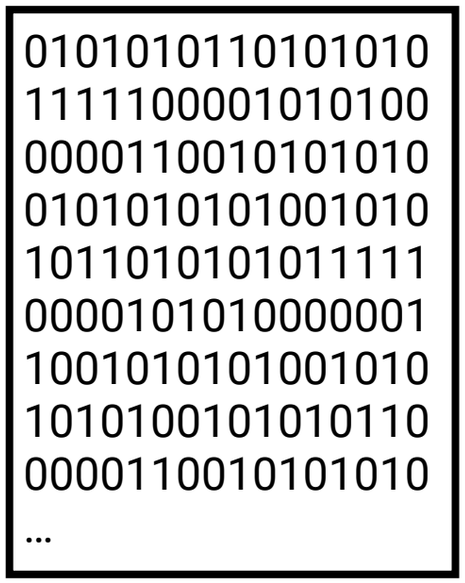
Encryption before compression?

Ticking away the moments that make up a dull day
Fritter and waste the hours in an off-hand way
Kicking around on a piece of ground in your home town
…

Encryption before compression?

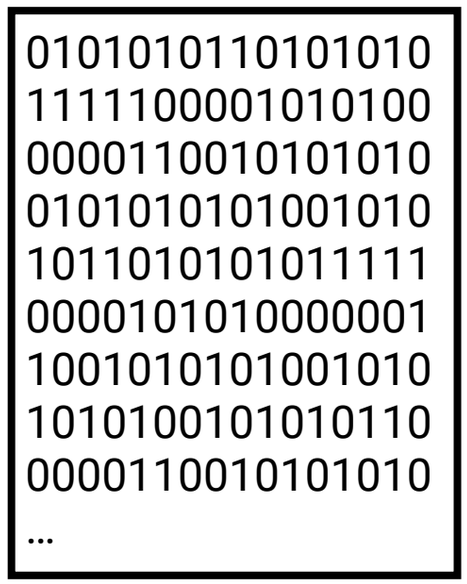
Encryption before compression?

Encryption before compression?



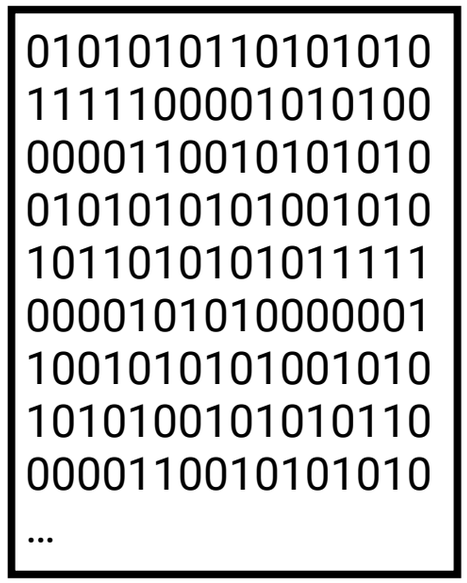
Encryption before compression?

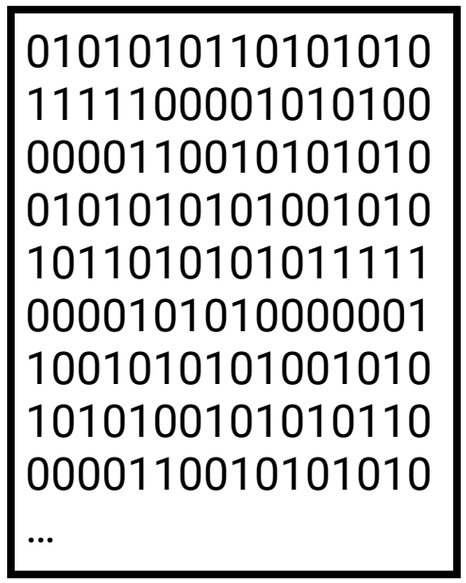
Ticking away the
moments that make
up a dull day
Fritter and waste the
hours in an off-hand
way
Kicking around on a
piece of ground in
your home town
…



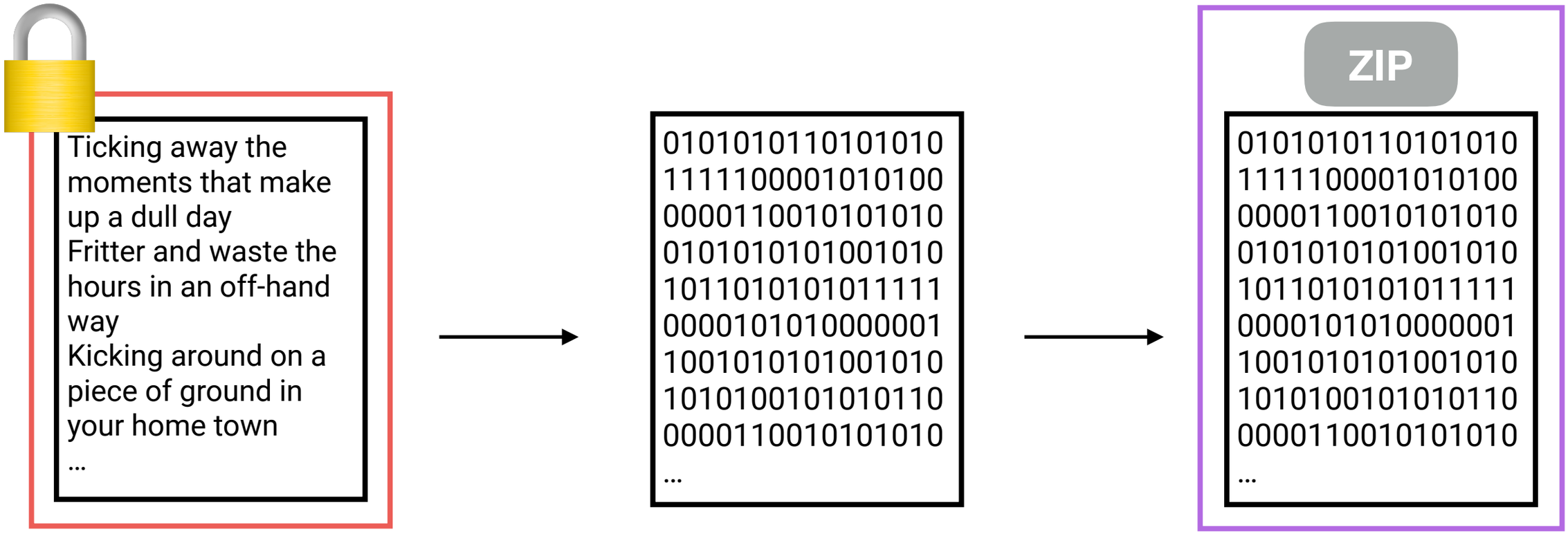
0101010110101010
1111100001010100
0000110010101010
0101010101001010
1011010101011111
0000101010000001
1001010101001010
1010100101010110
0000110010101010
…

Encryption before compression?

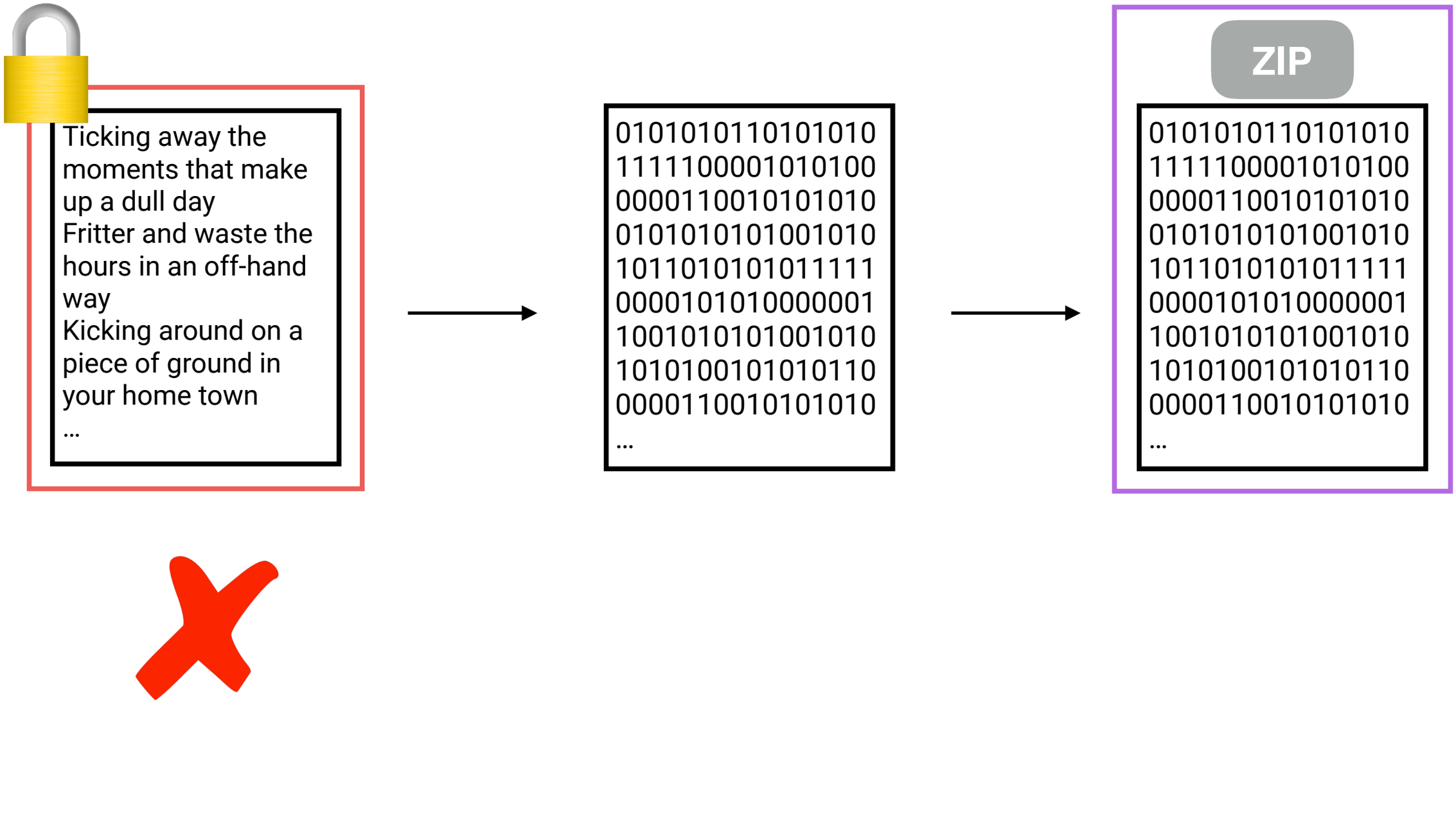




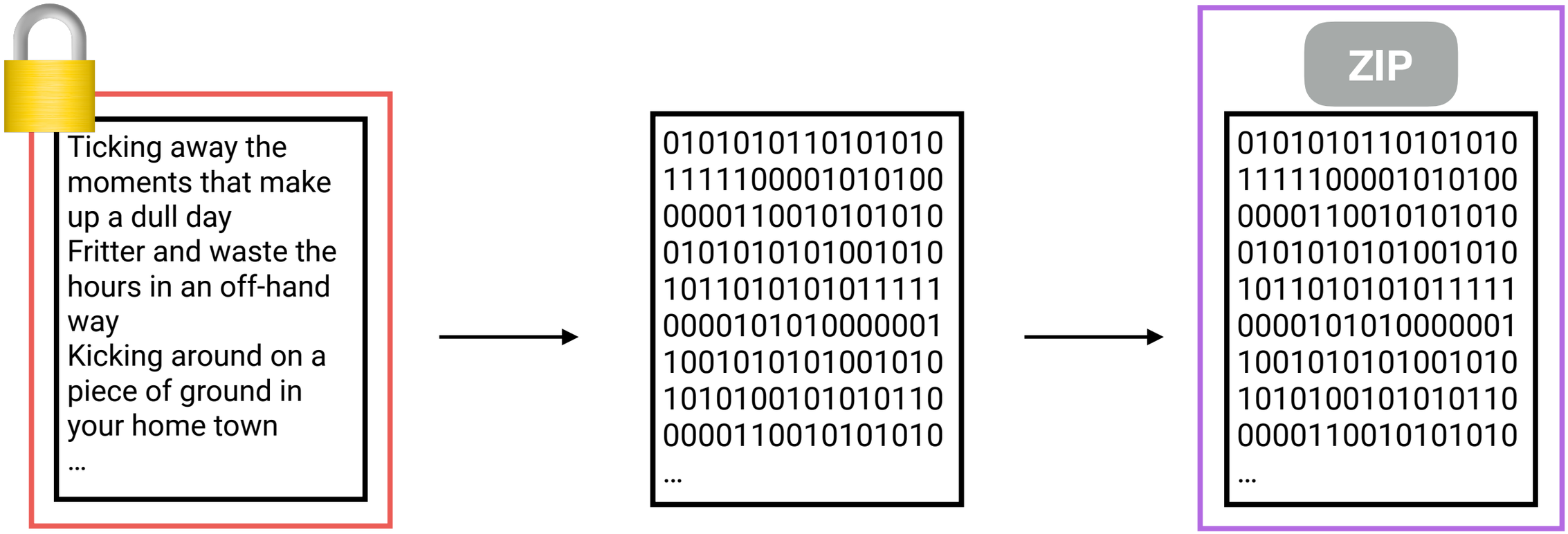
Encryption before compression?



Encryption before compression?



Encryption before compression?



Pseudorandom data is not compressible

Compression before encryption?

Compression before encryption?

Ticking away the moments that make up a dull day
Fritter and waste the hours in an off-hand way
Kicking around on a piece of ground in your home town
…

Compression before encryption?

ZIP

Ticking away the moments that make up a dull day
Fritter and waste the hours in an off-hand way
Kicking around on a piece of ground in your home town
…

Compression before encryption?

ZIP

Ticking away the moments that make up a dull day
Fritter and waste the hours in an off-hand way
Kicking around on a piece of ground in your home town
…

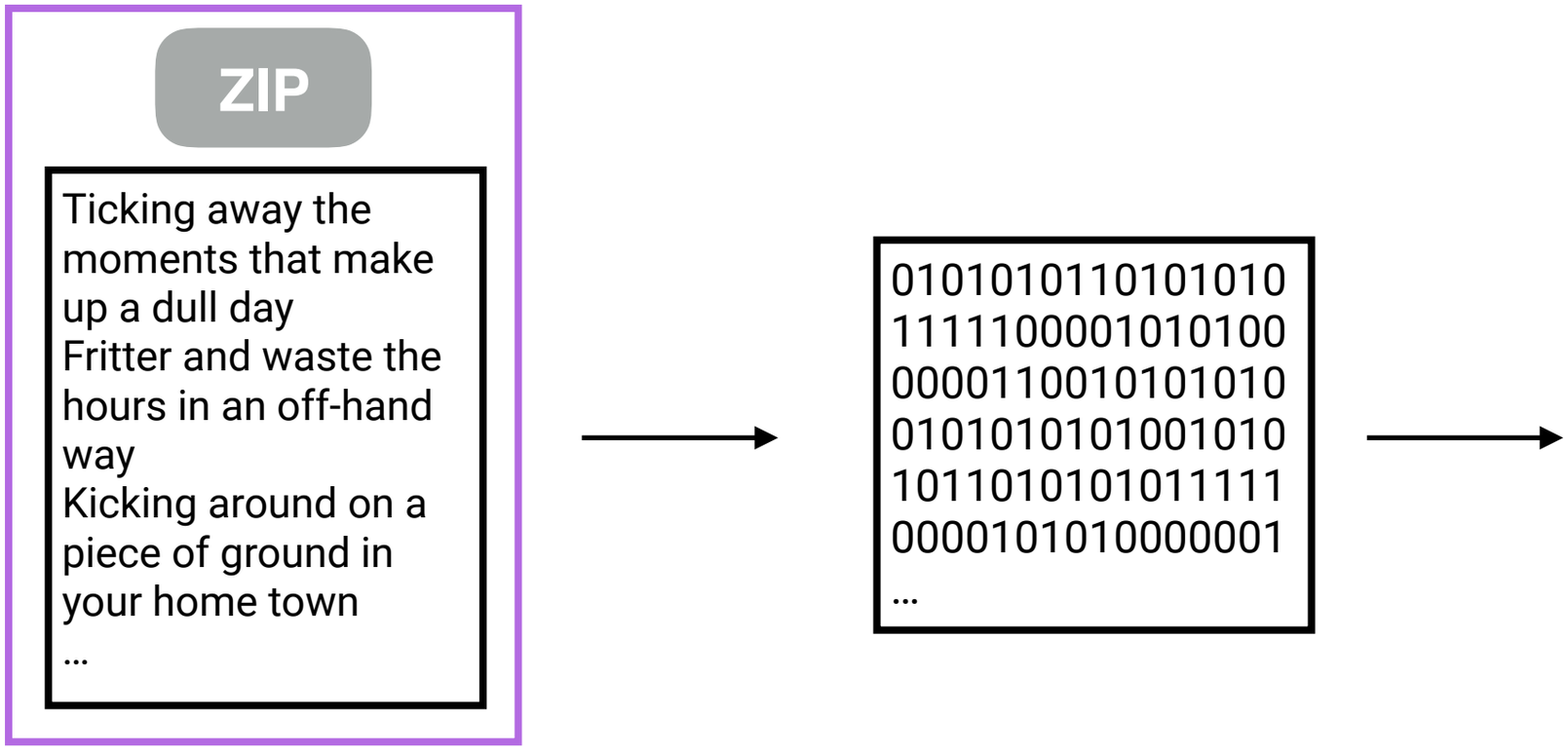
Compression before encryption?

ZIP

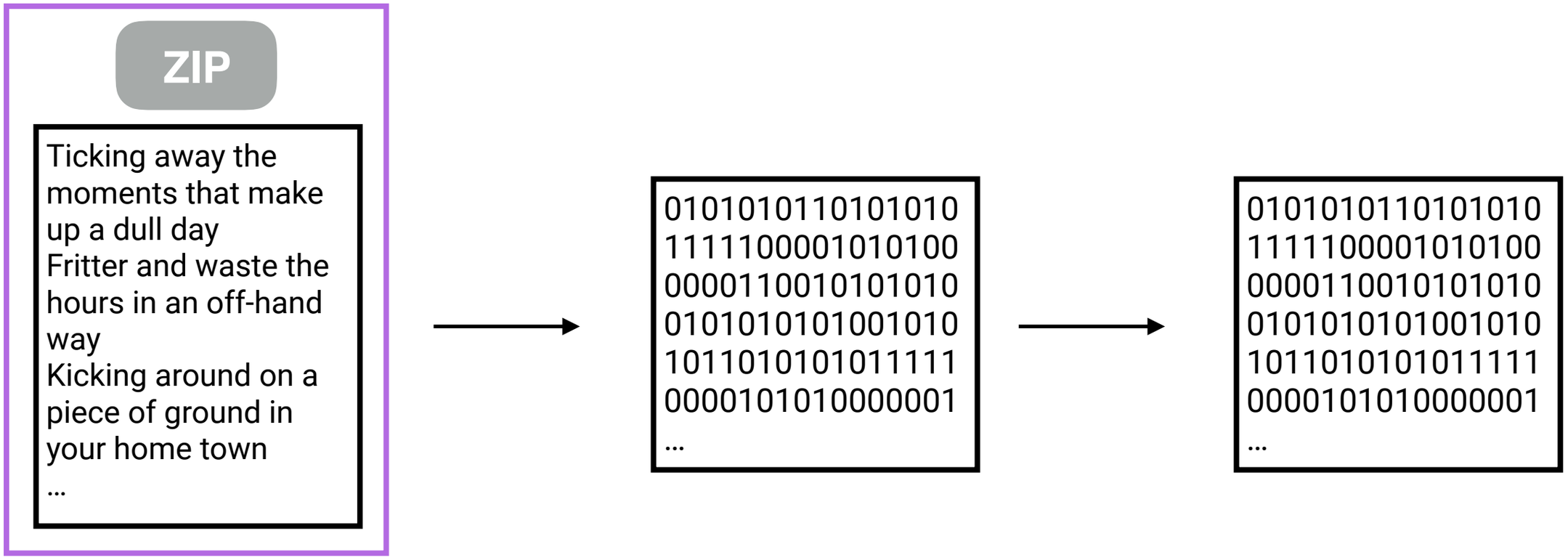
Ticking away the moments that make up a dull day
Fritter and waste the hours in an off-hand way
Kicking around on a piece of ground in your home town
…

0101010110101010
1111100001010100
0000110010101010
0101010101001010
1011010101011111
0000101010000001
…

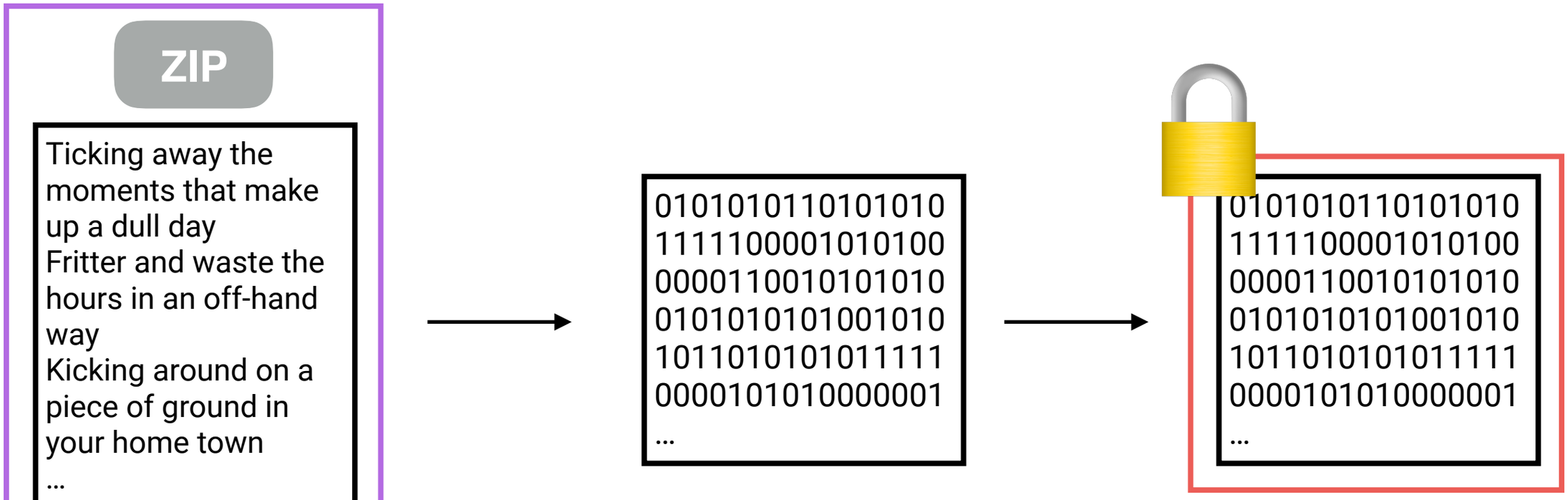
Compression before encryption?



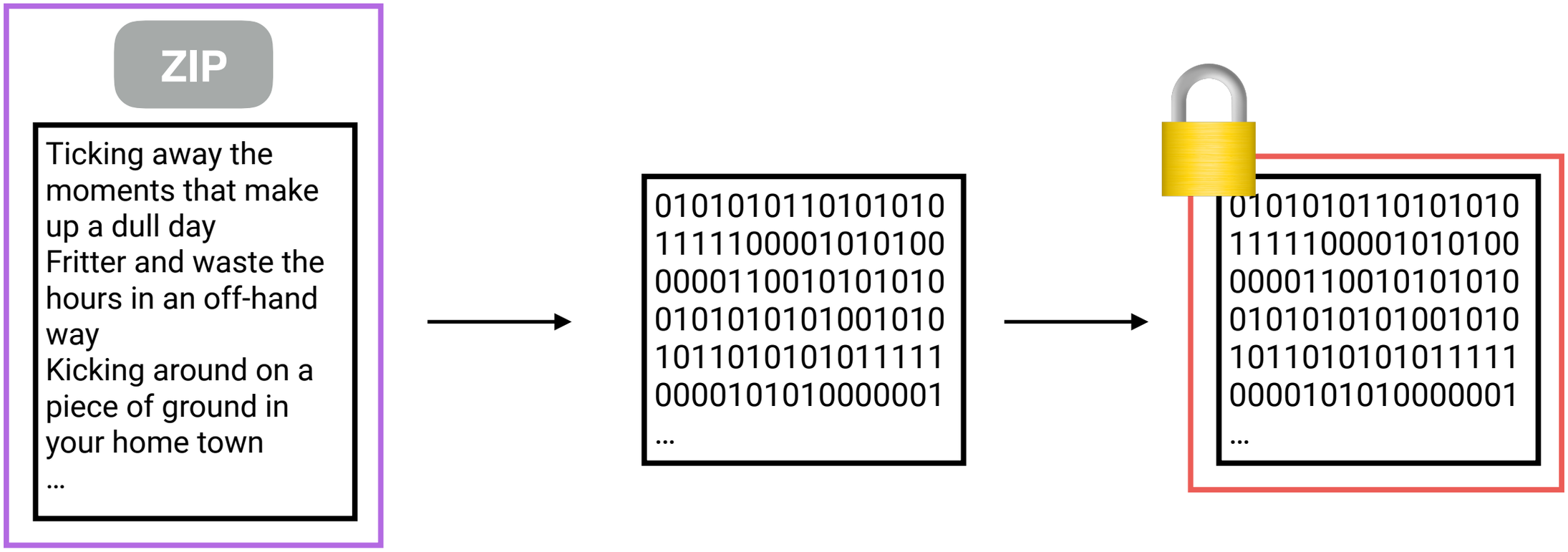
Compression before encryption?



Compression before encryption?



Compression before encryption?

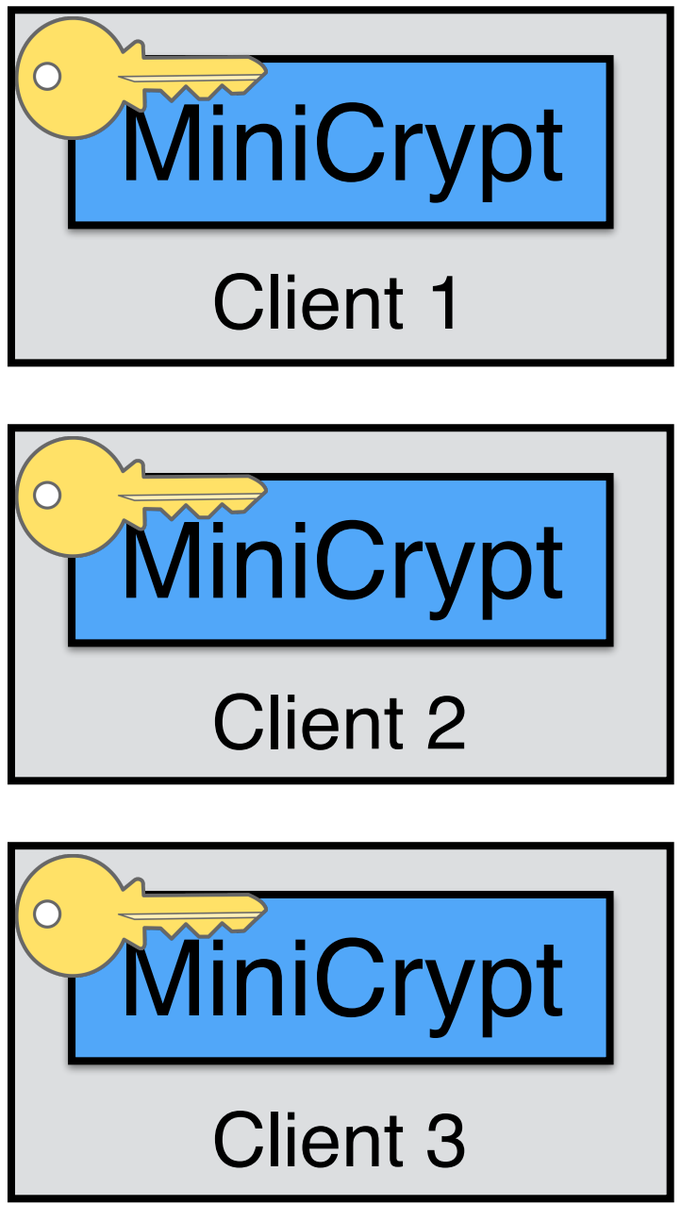


Compression and encryption

Strawman #1:
compress & encrypt entire database

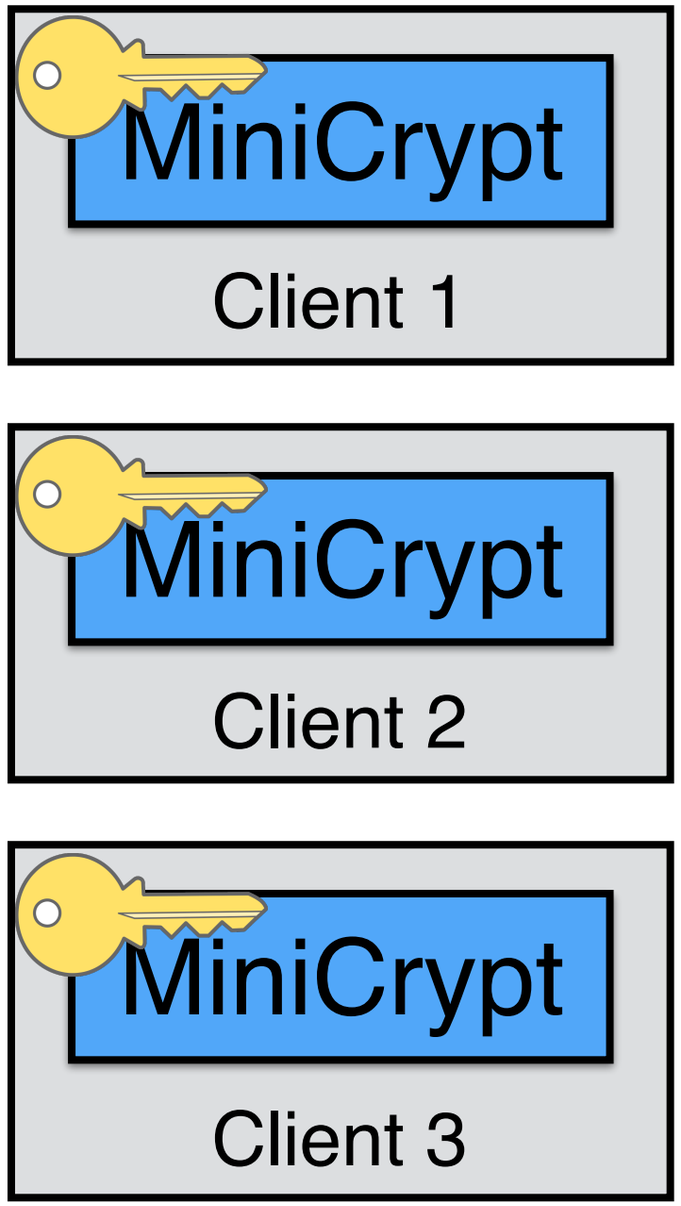
Strawman #1:
compress & encrypt entire database

Strawman #1:
compress & encrypt entire database



12809	Amanda D. Edwards	40	Diabetes
29489	Robert R. McGowan	56	Diabetes
13744	Kimberly R. Seay	51	Cancer
18740	Dennis G. Bates	32	Diabetes
98329	Ronald S. Ogden	53	Cancer
32591	Donna R. Bridges	26	Diabetes

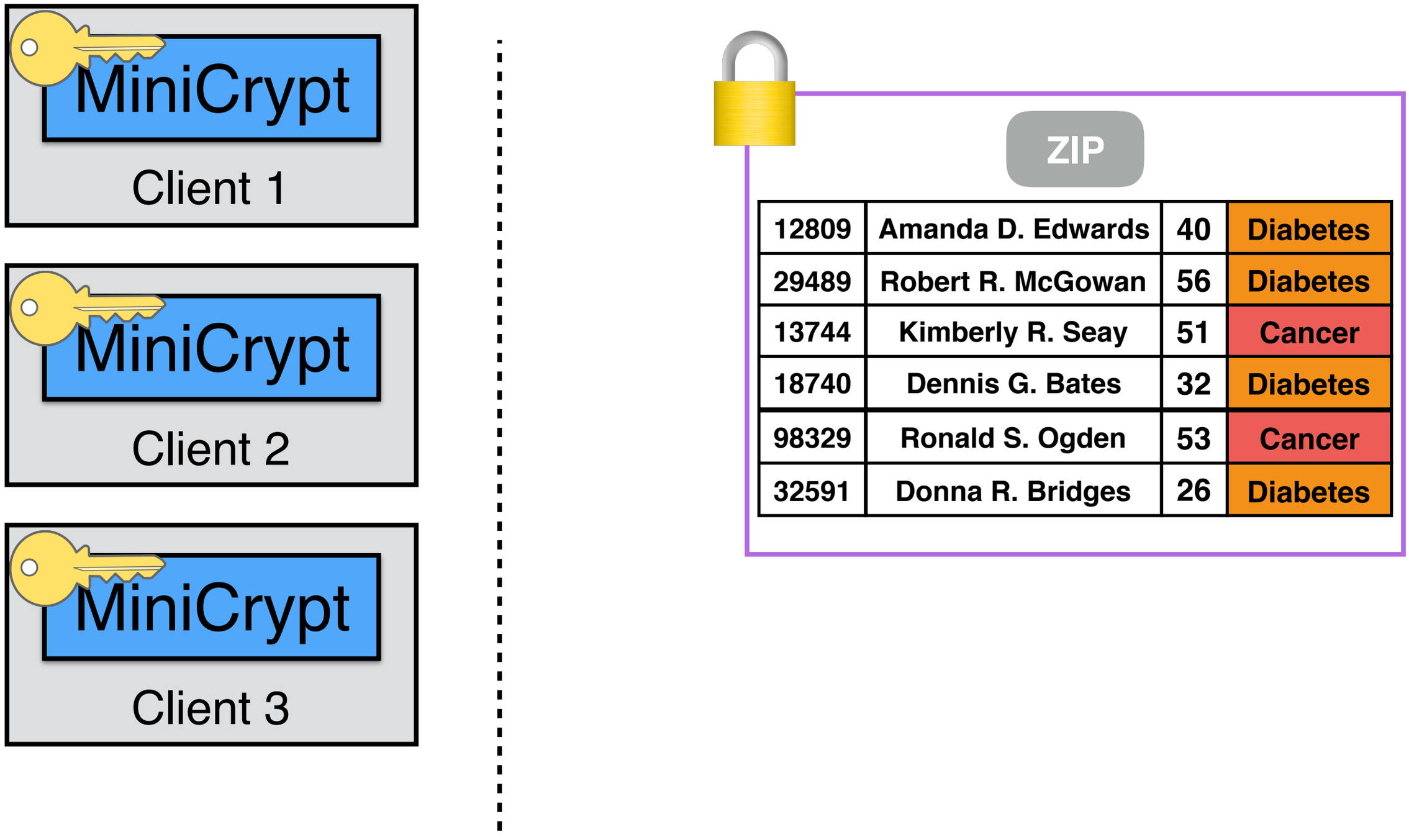
Strawman #1:
compress & encrypt entire database



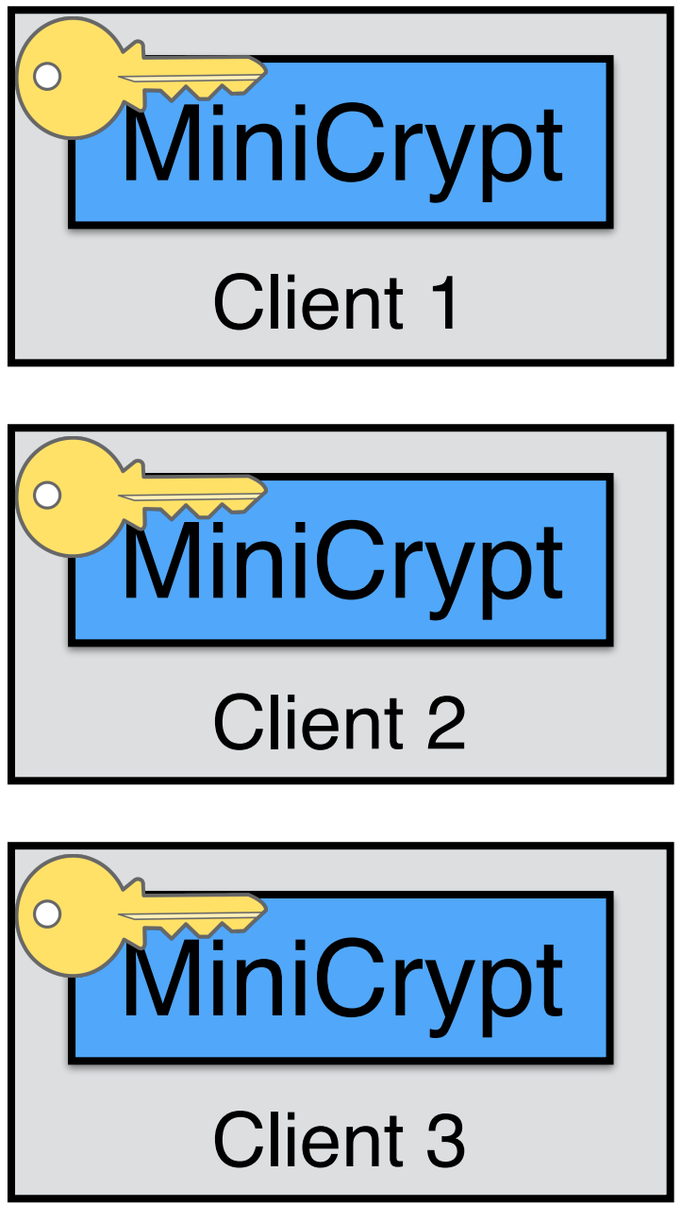
ZIP

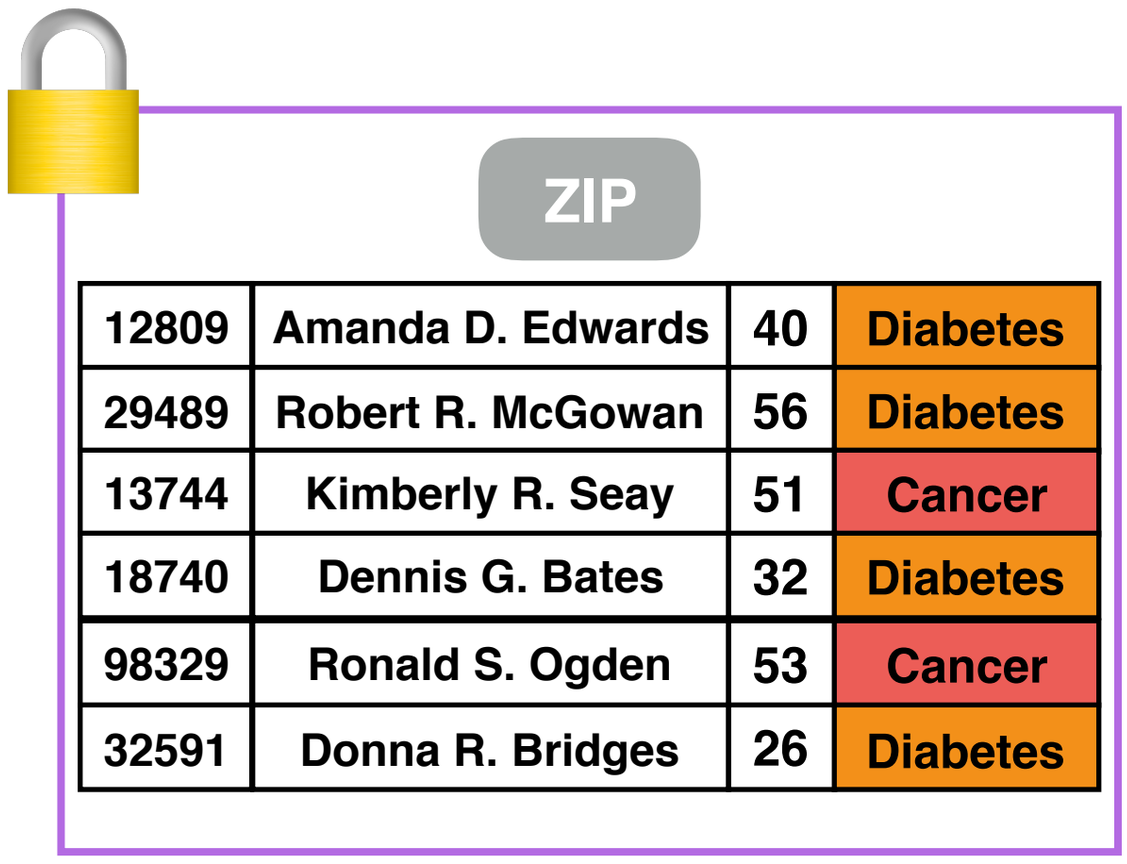
12809	Amanda D. Edwards	40	Diabetes
29489	Robert R. McGowan	56	Diabetes
13744	Kimberly R. Seay	51	Cancer
18740	Dennis G. Bates	32	Diabetes
98329	Ronald S. Ogden	53	Cancer
32591	Donna R. Bridges	26	Diabetes

Strawman #1:
compress & encrypt entire database



Strawman #1: compress & encrypt entire database

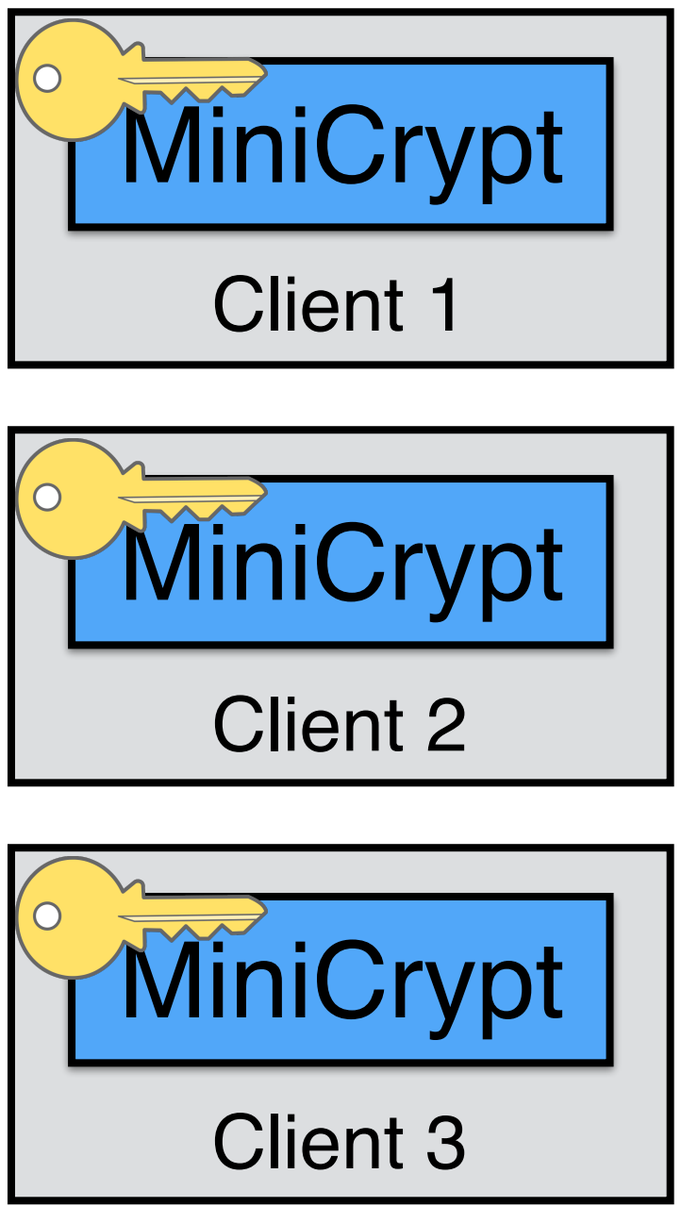




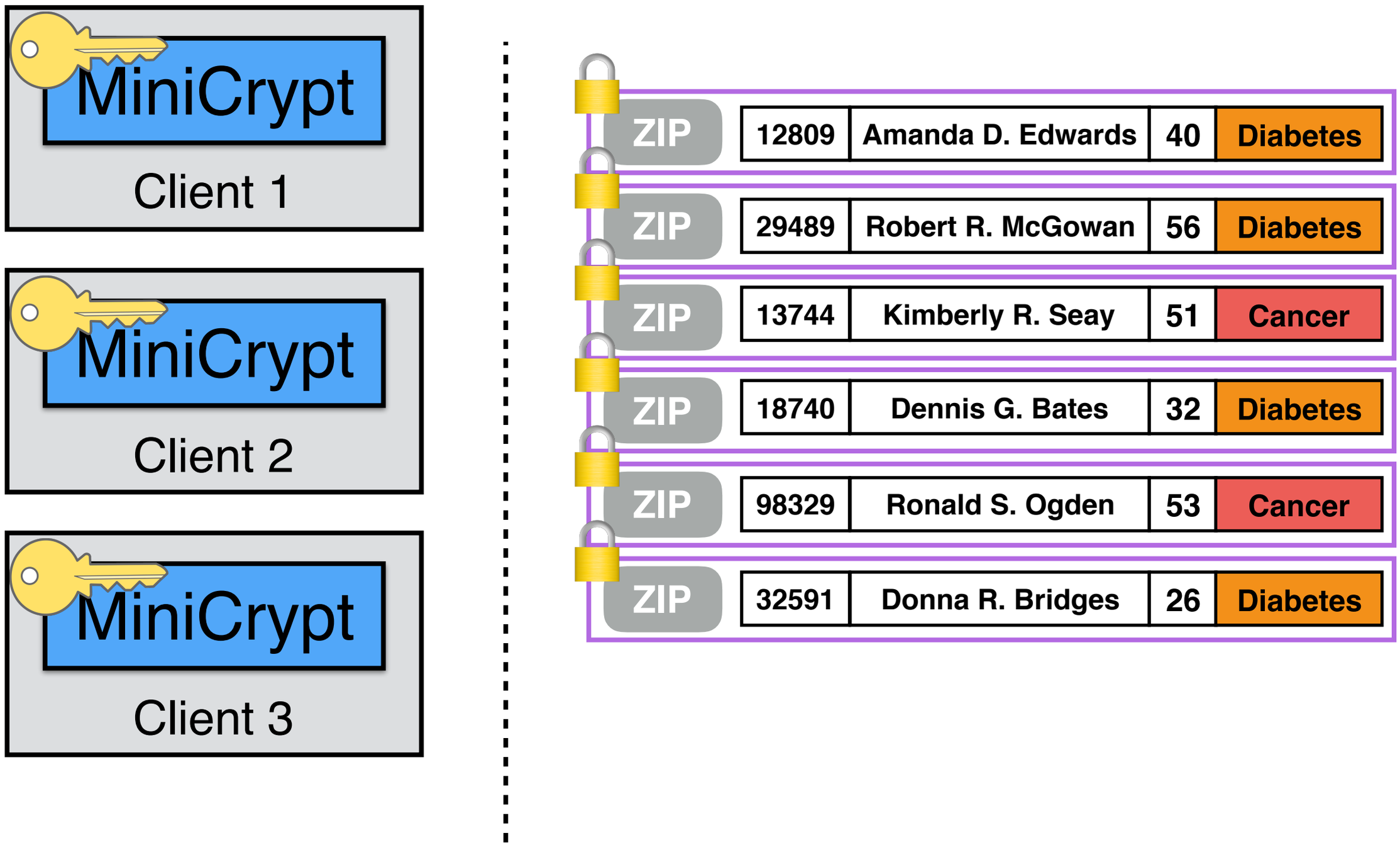
Problem: high network bandwidth

Strawman #2:
compress & encrypt single row

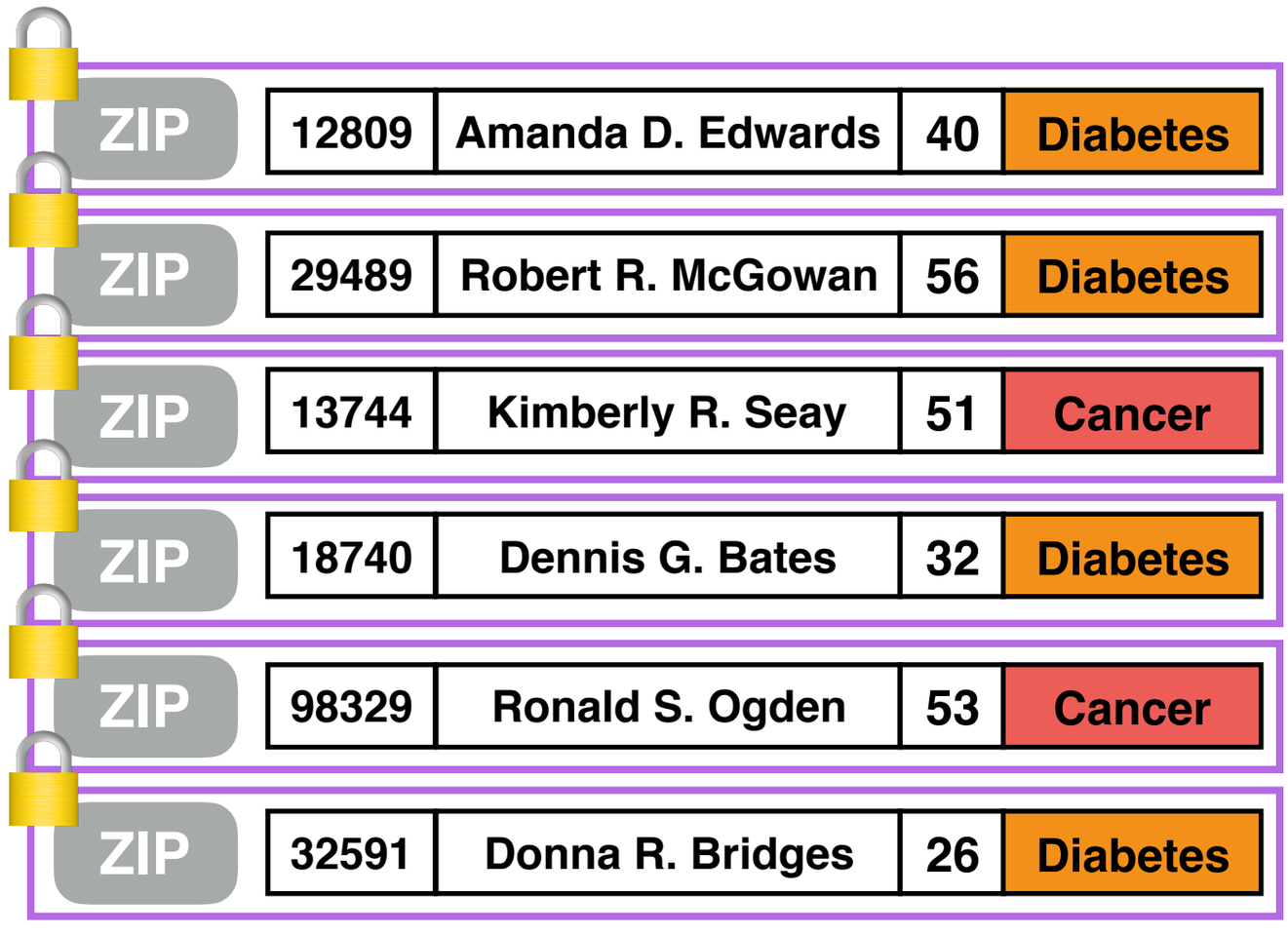
Strawman #2:
compress & encrypt single row



Strawman #2: compress & encrypt single row



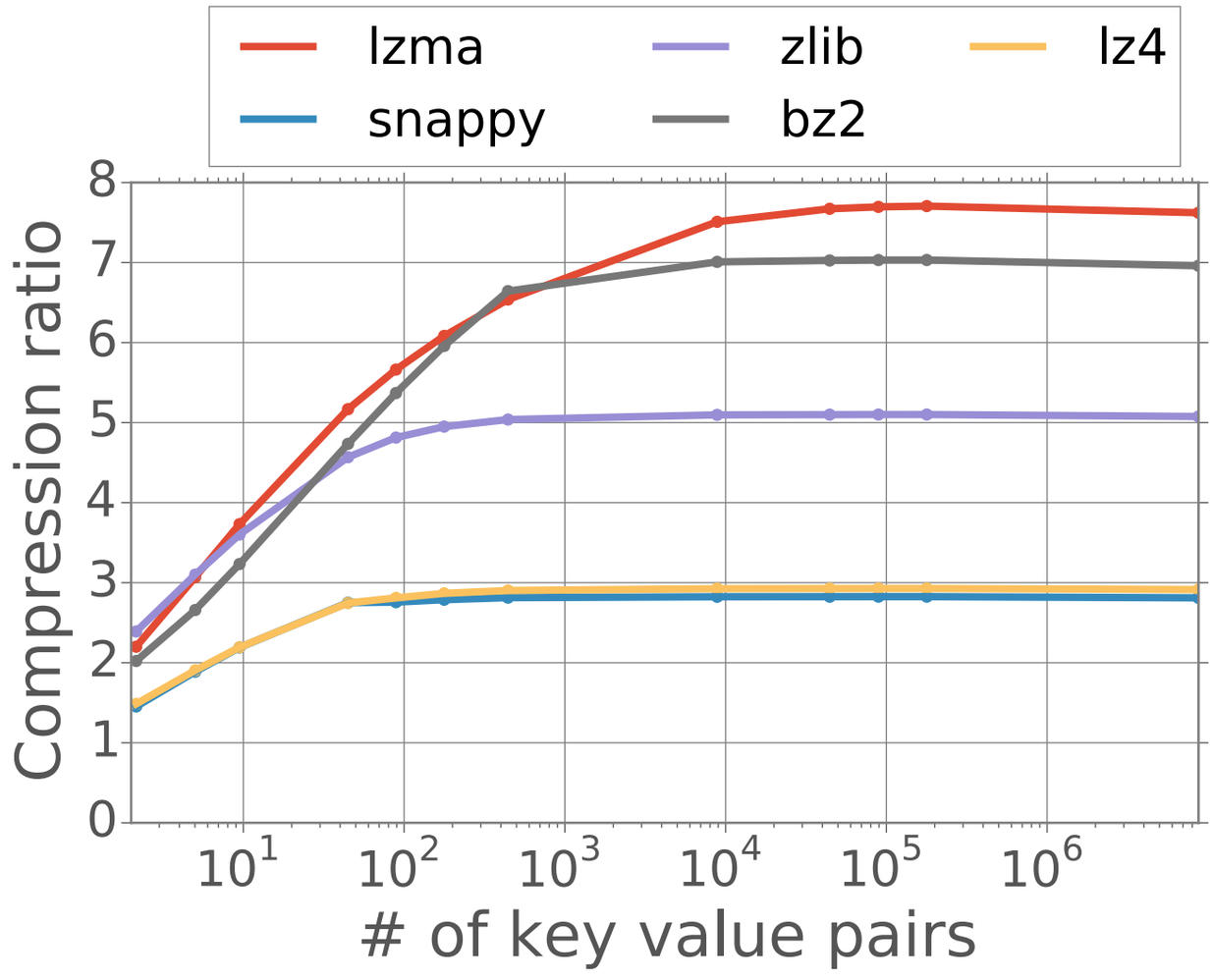
Strawman #2: compress & encrypt single row



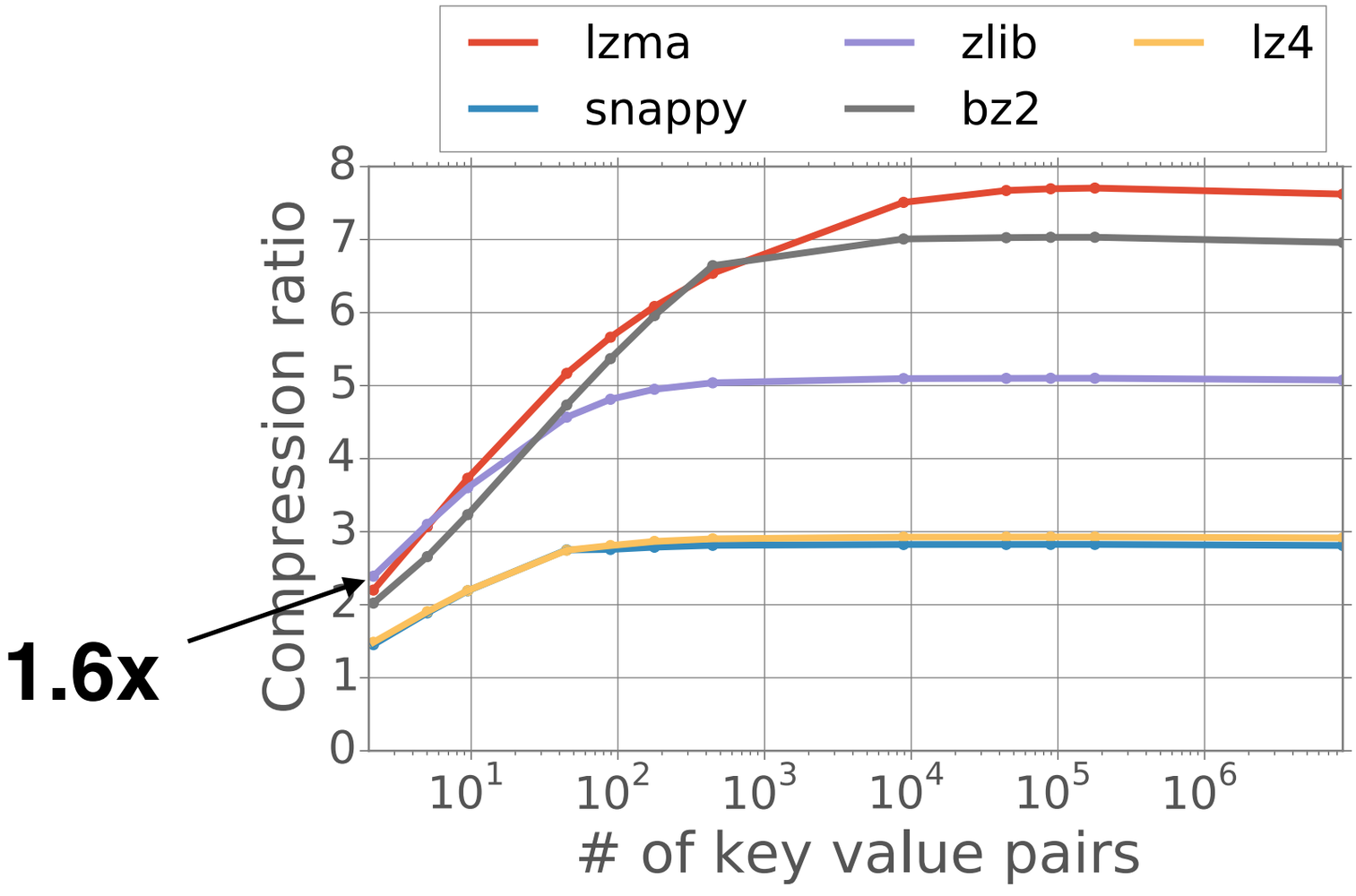
Problem: small compression ratio

Observation

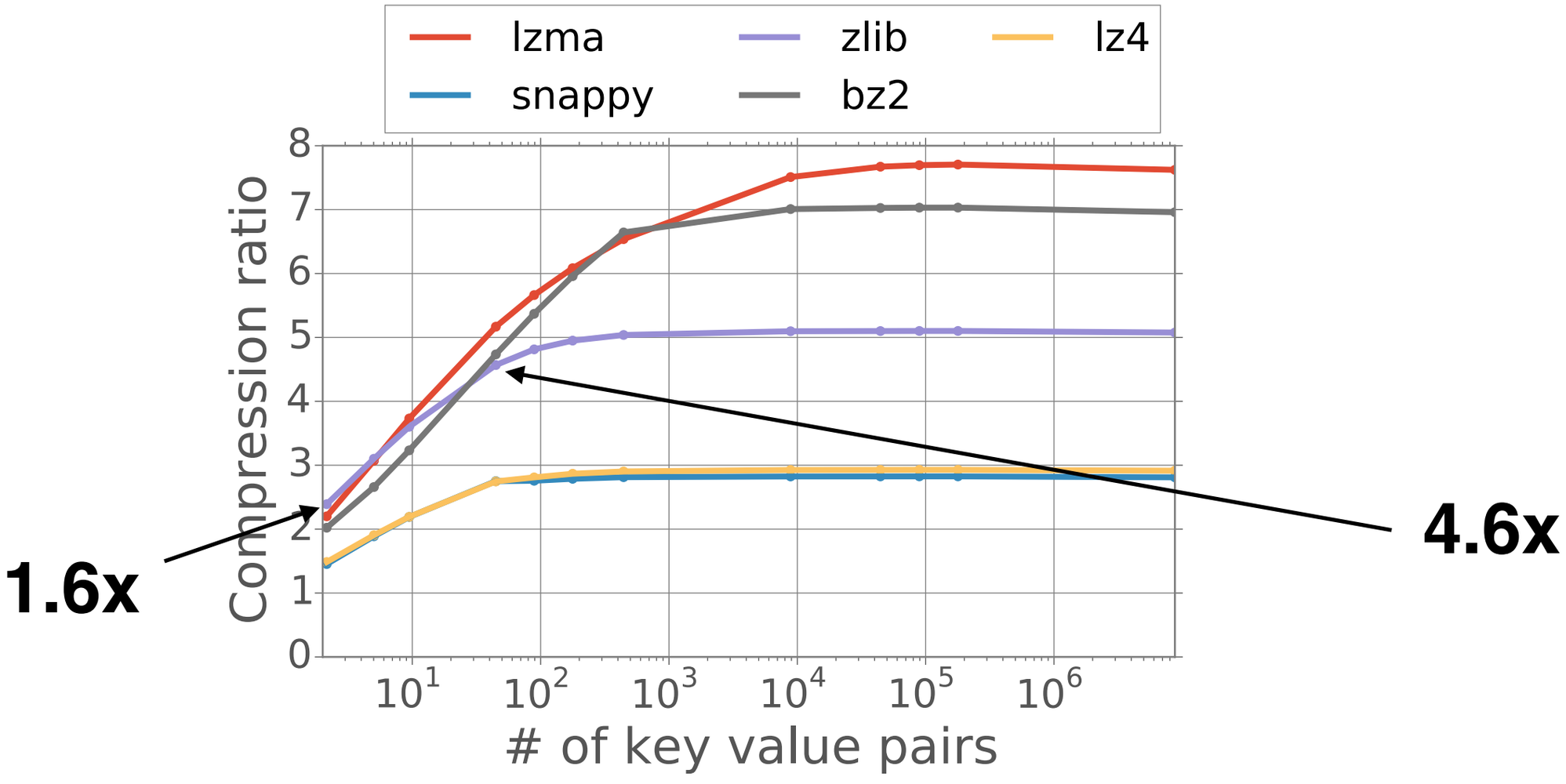
Observation



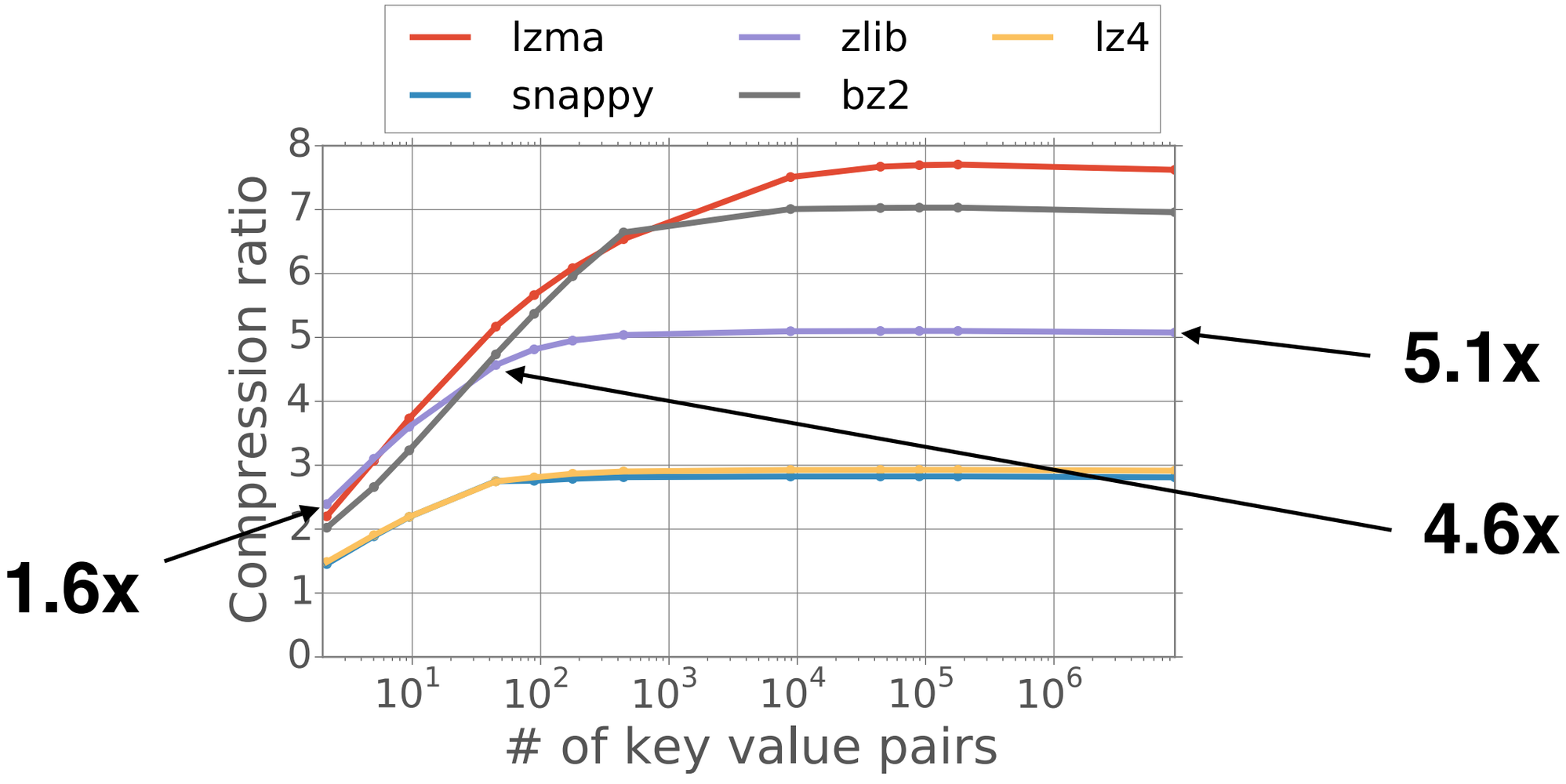
Observation



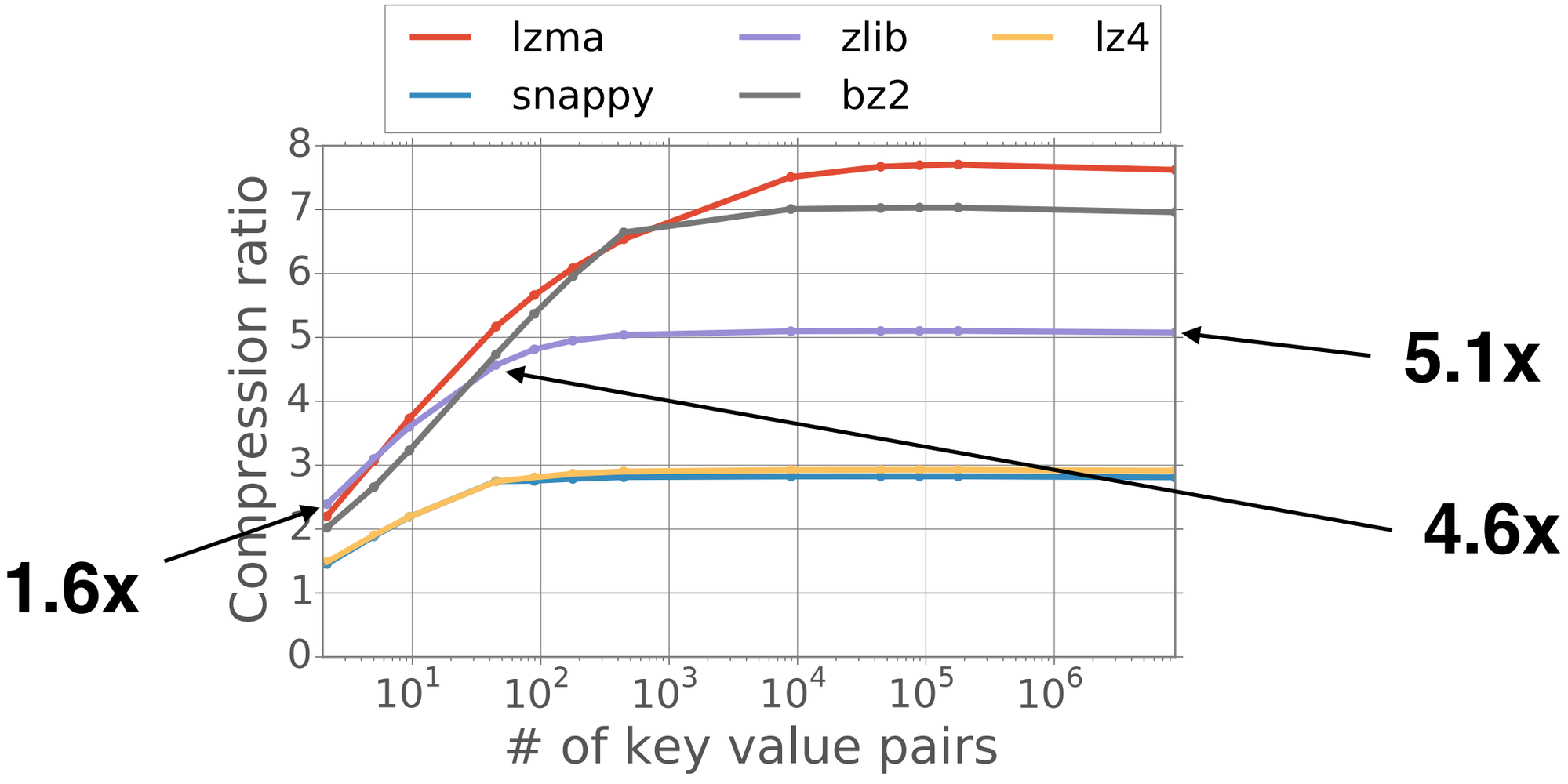
Observation



Observation



Observation



Small # of key-value pairs gives high compression

Approach

Approach

1. Combine small number of key-value pairs into packs

Approach

1. Combine small number of key-value pairs into packs

2. Compress and encrypt each pack

Challenges

Challenges

1. Server cannot decrypt packs

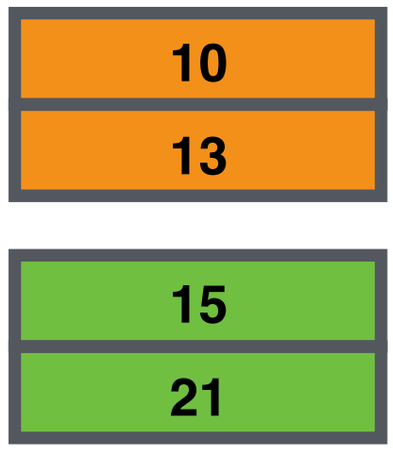
Challenges

1. Server cannot decrypt packs

2. Multi-key transactions are expensive and not available for some key-value stores

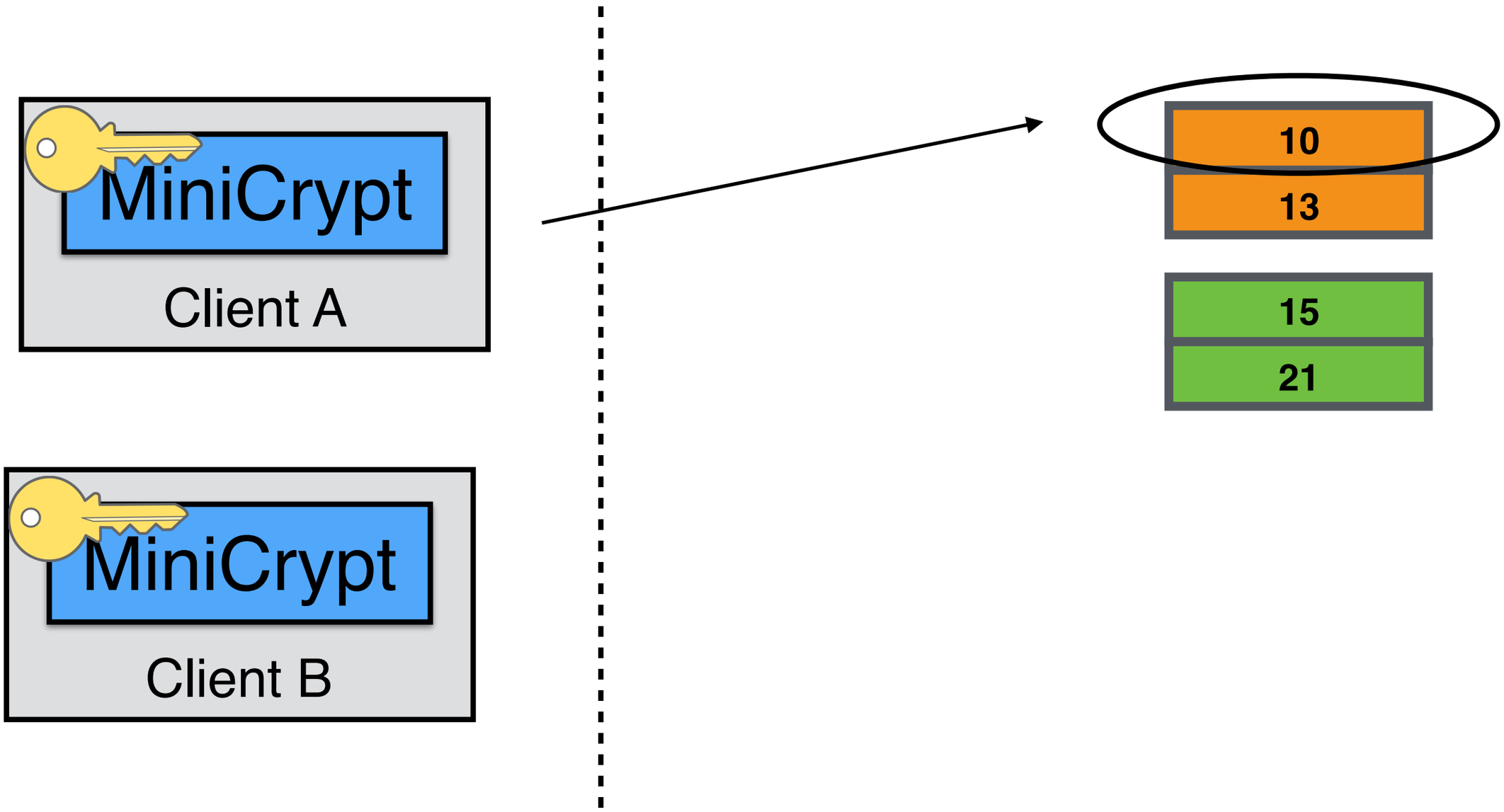
Problem 1

Updates to different keys within a pack can overwrite each other



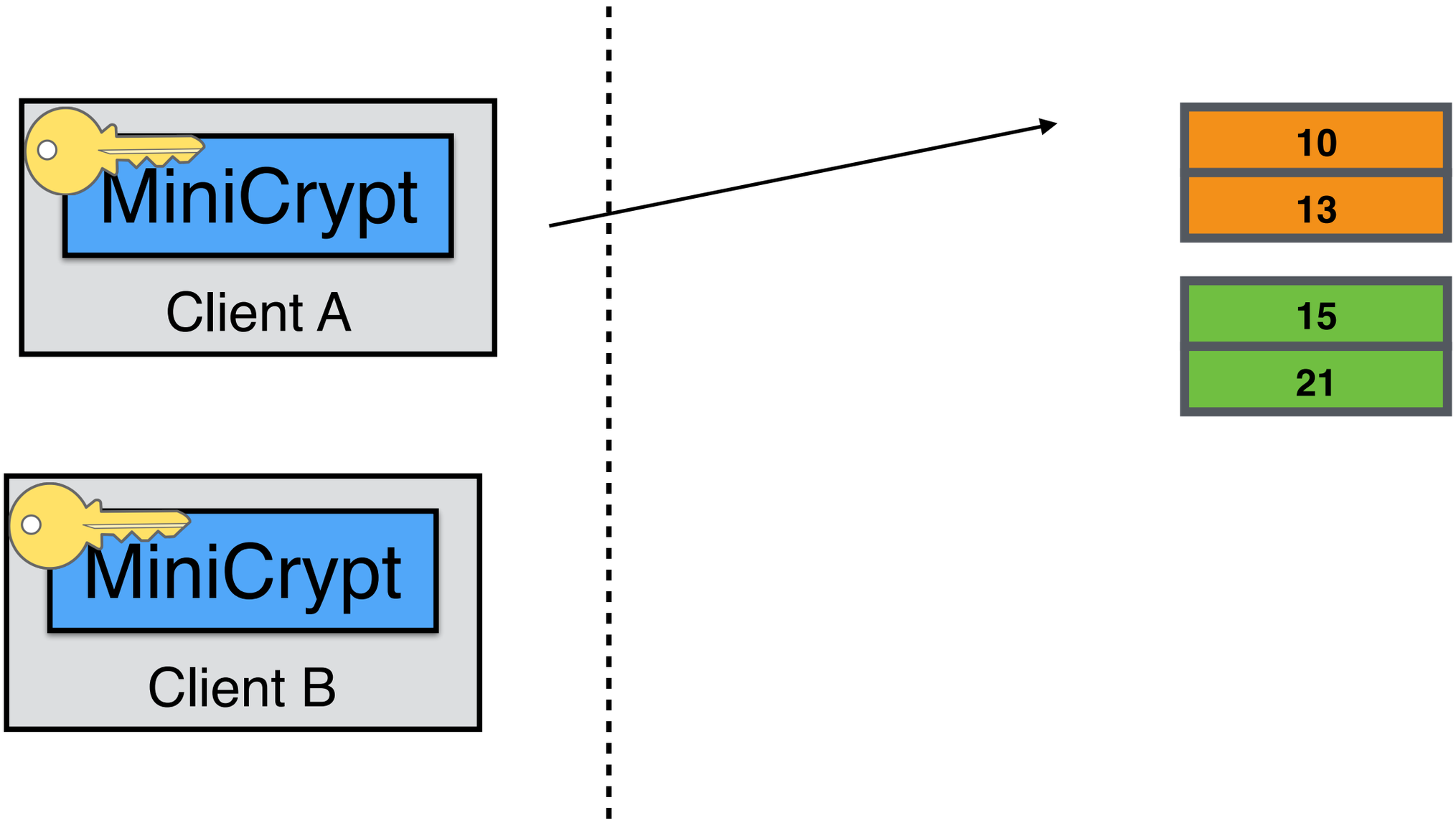
Problem 1

Updates to different keys within a pack can overwrite each other



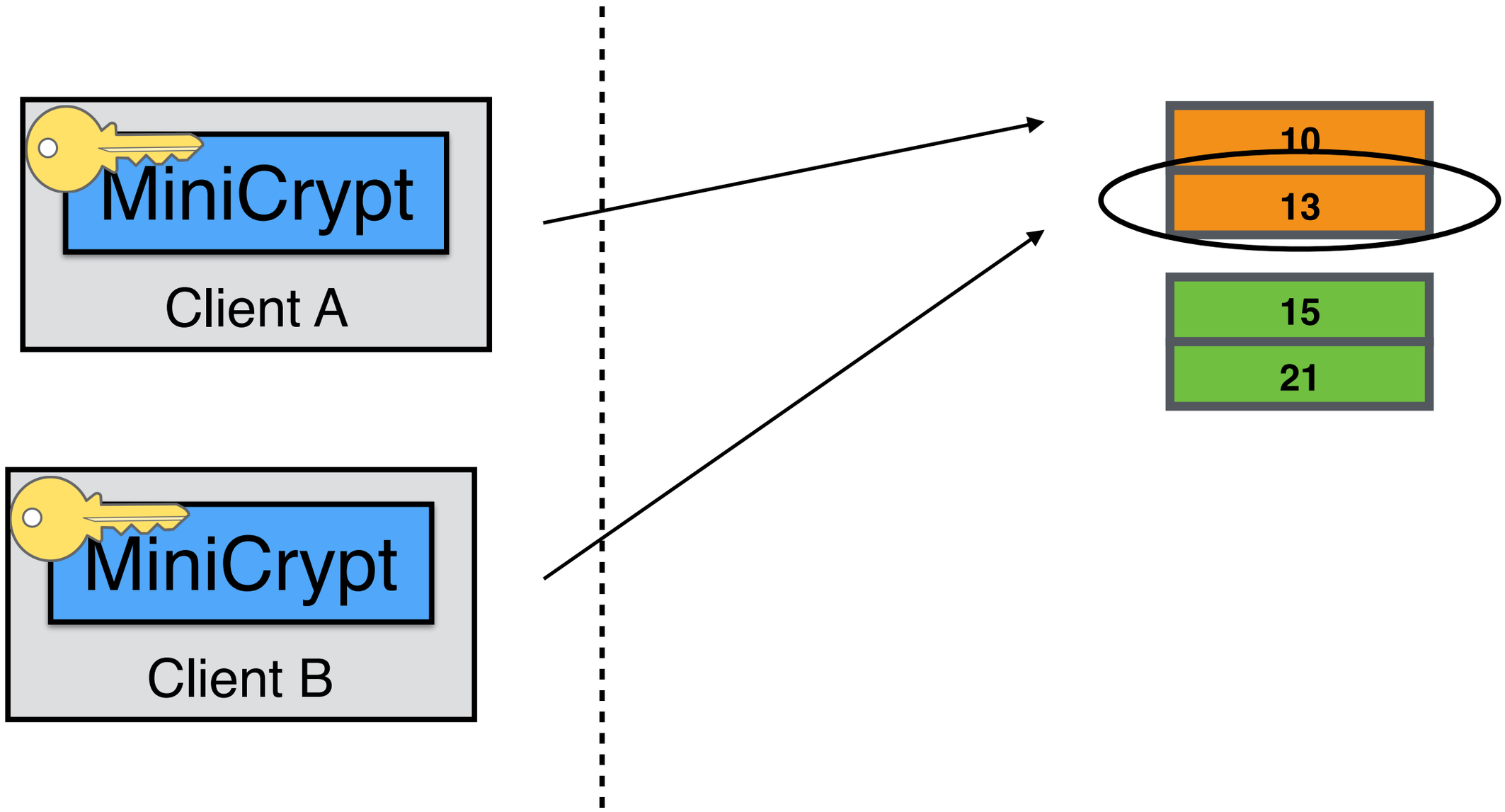
Problem 1

Updates to different keys within a pack can overwrite each other



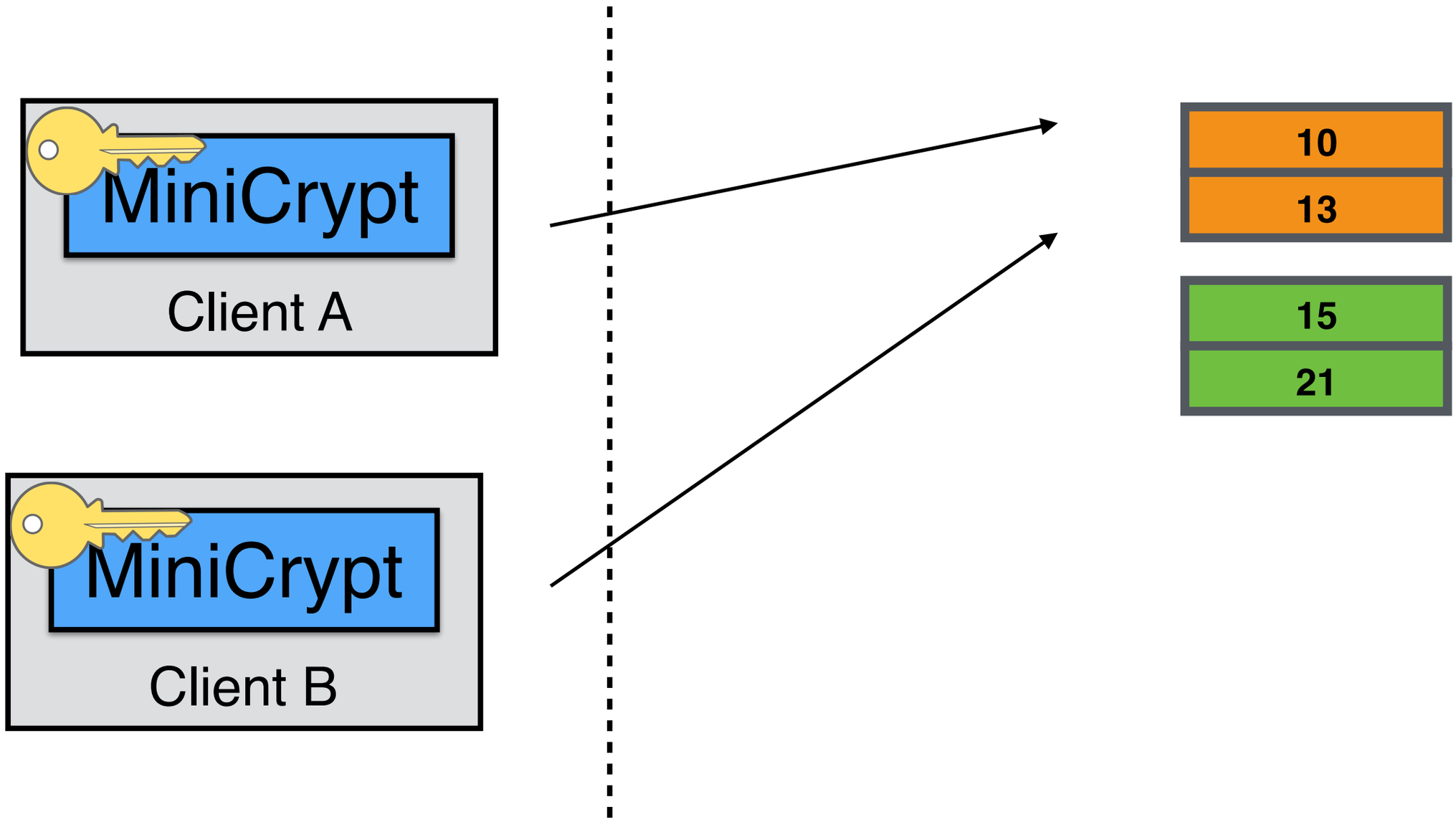
Problem 1

Updates to different keys within a pack can overwrite each other



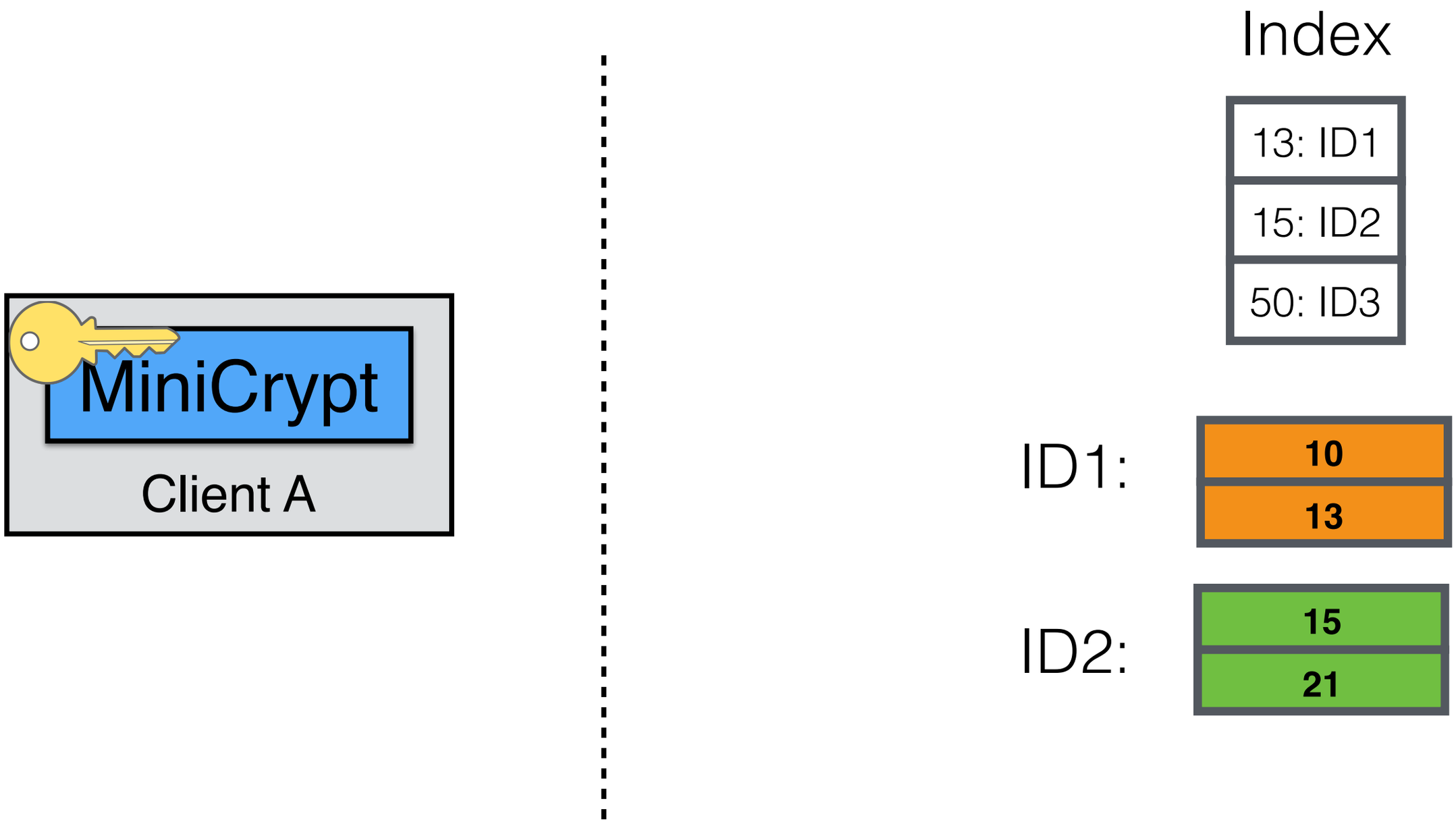
Problem 1

Updates to different keys within a pack can overwrite each other



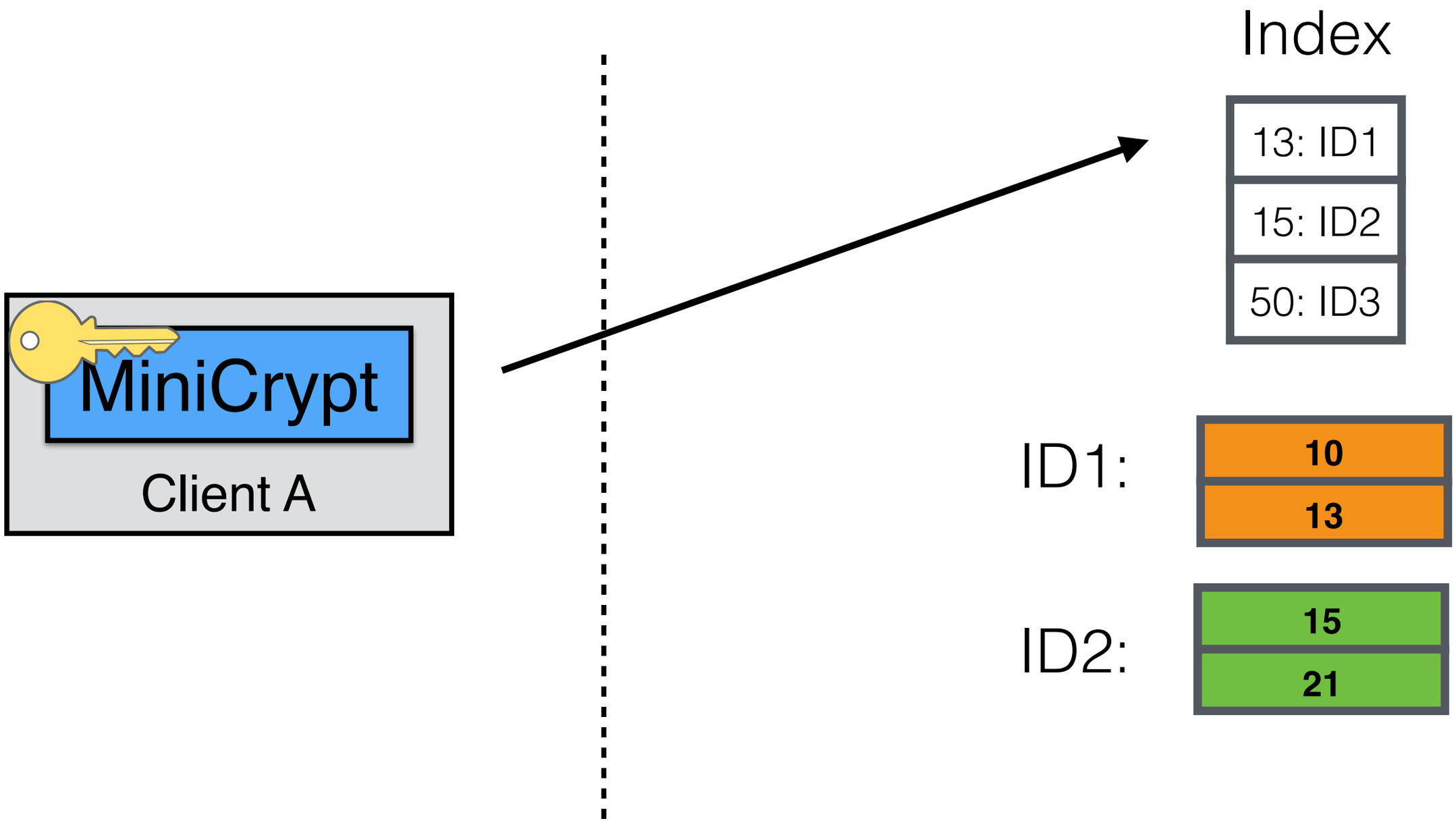
Problem 2

Additional index is difficult to keep consistent



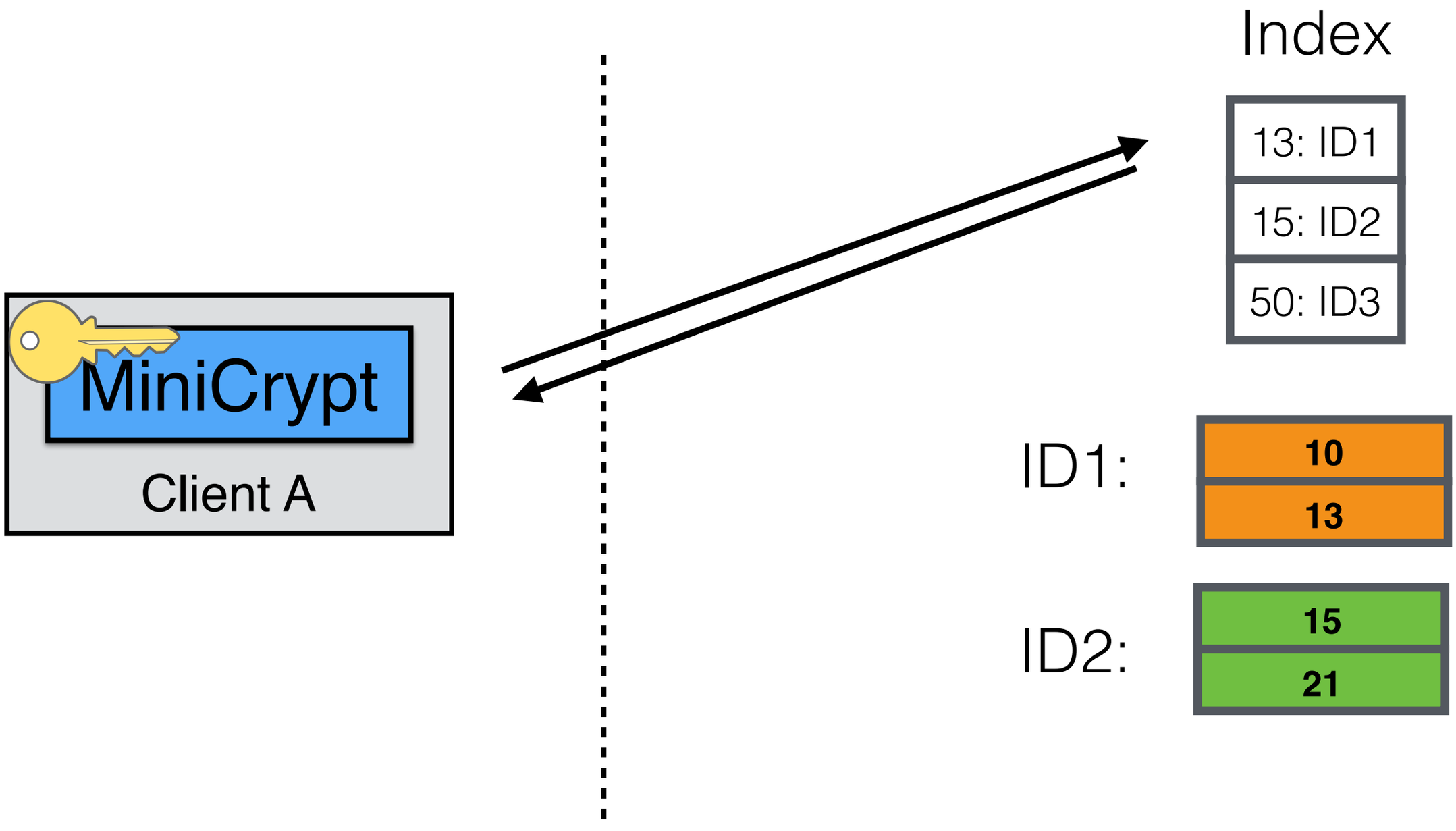
Problem 2

Additional index is difficult to keep consistent



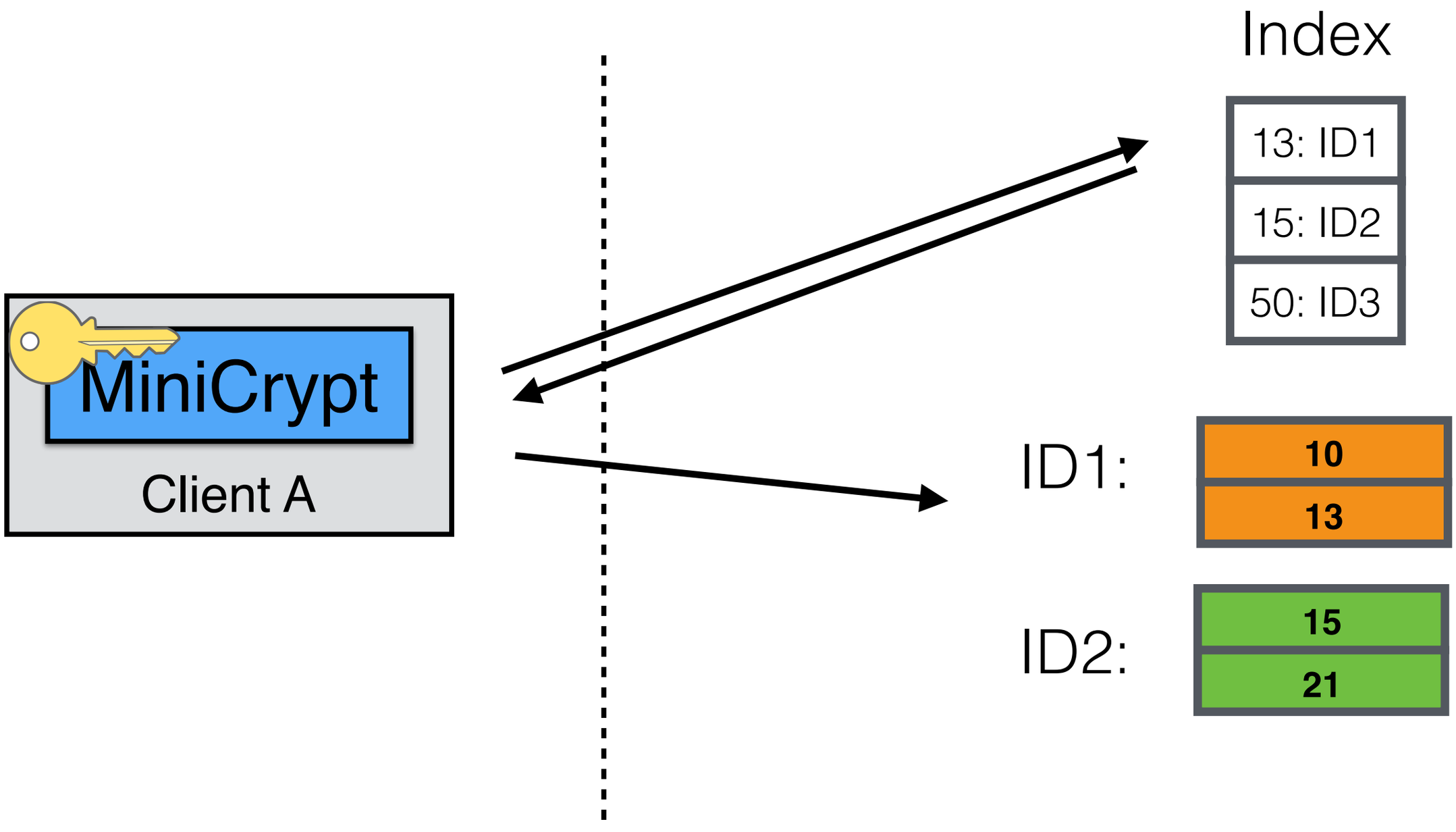
Problem 2

Additional index is difficult to keep consistent



Problem 2

Additional index is difficult to keep consistent



MiniCrypt's techniques

- lookup algorithm
- update algorithm
- splitting algorithm
- epoch-based appends
- merge algorithm

MiniCrypt's techniques

- lookup algorithm
- update algorithm
- splitting algorithm
- epoch-based appends
- merge algorithm

using only

MiniCrypt's techniques

- lookup algorithm
- update algorithm
- splitting algorithm
- epoch-based appends
- merge algorithm

using only

- single-key atomic compare-and-swap

MiniCrypt's techniques

- lookup algorithm
- update algorithm
- splitting algorithm
- epoch-based appends
- merge algorithm

using only

- single-key atomic compare-and-swap
- sorted index on the primary key

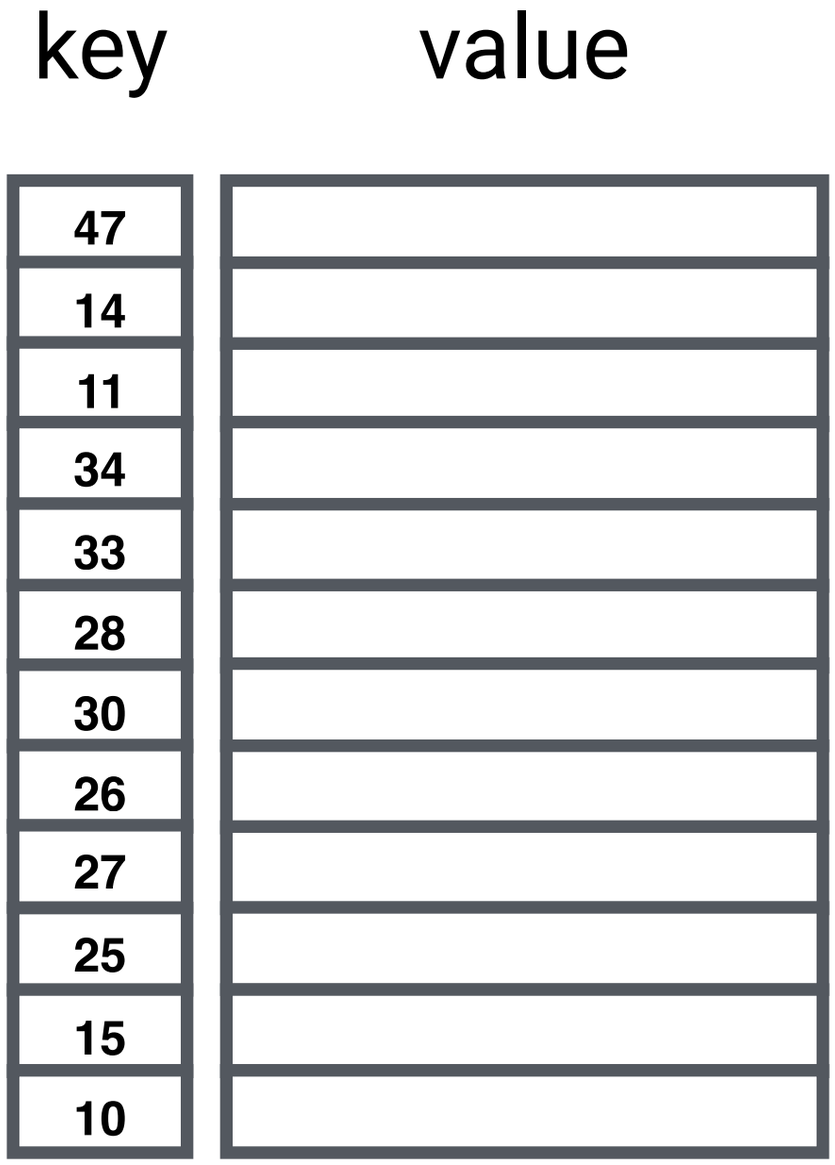
API

- **get(key)**
- **put(key, value)**
- **delete(key)**
- **range(low_key, high_key)**

API

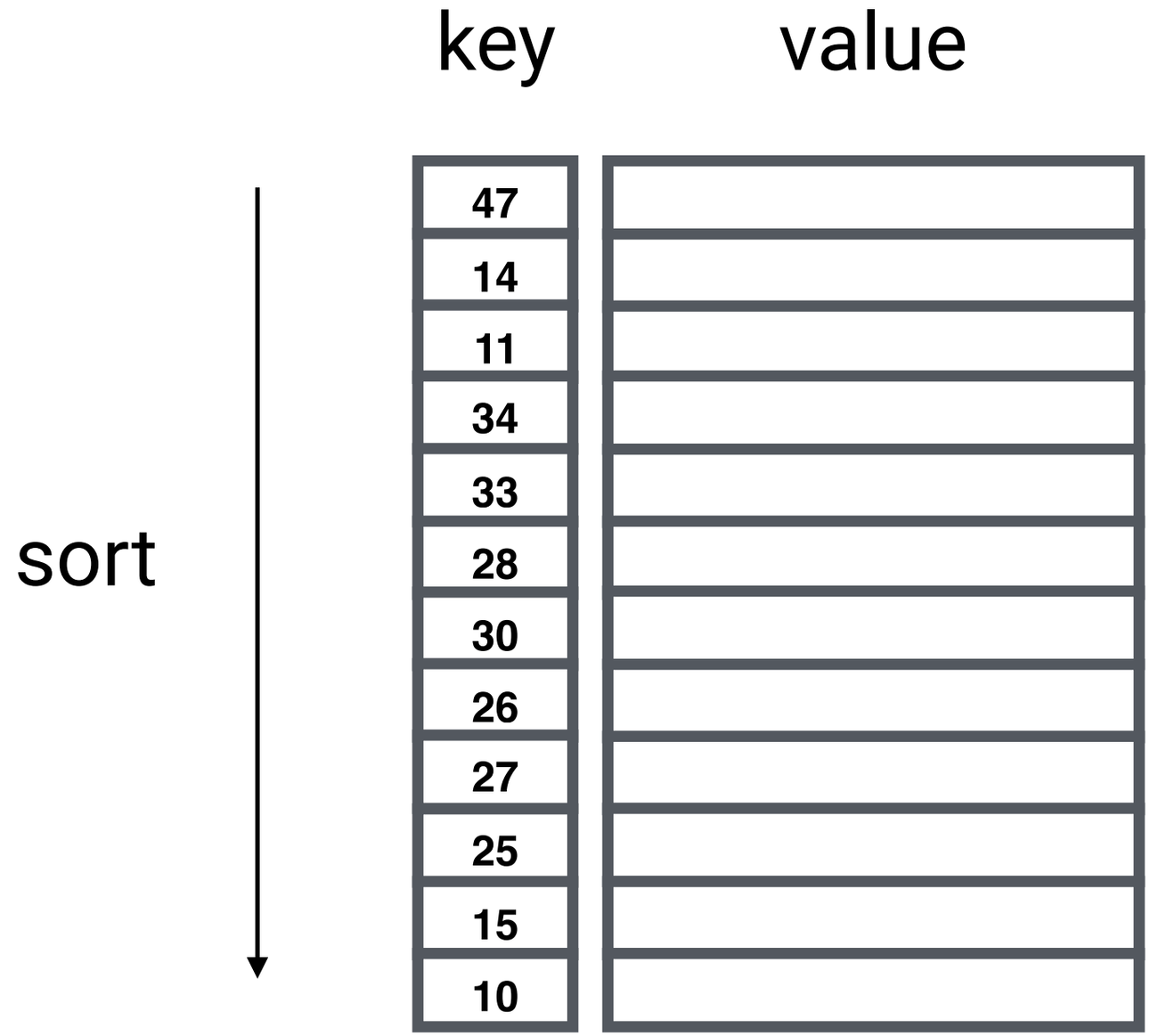
- **get(key)**
- **put(key, value)**
- **delete(key)**
- **range(low_key, high_key)**

Packing



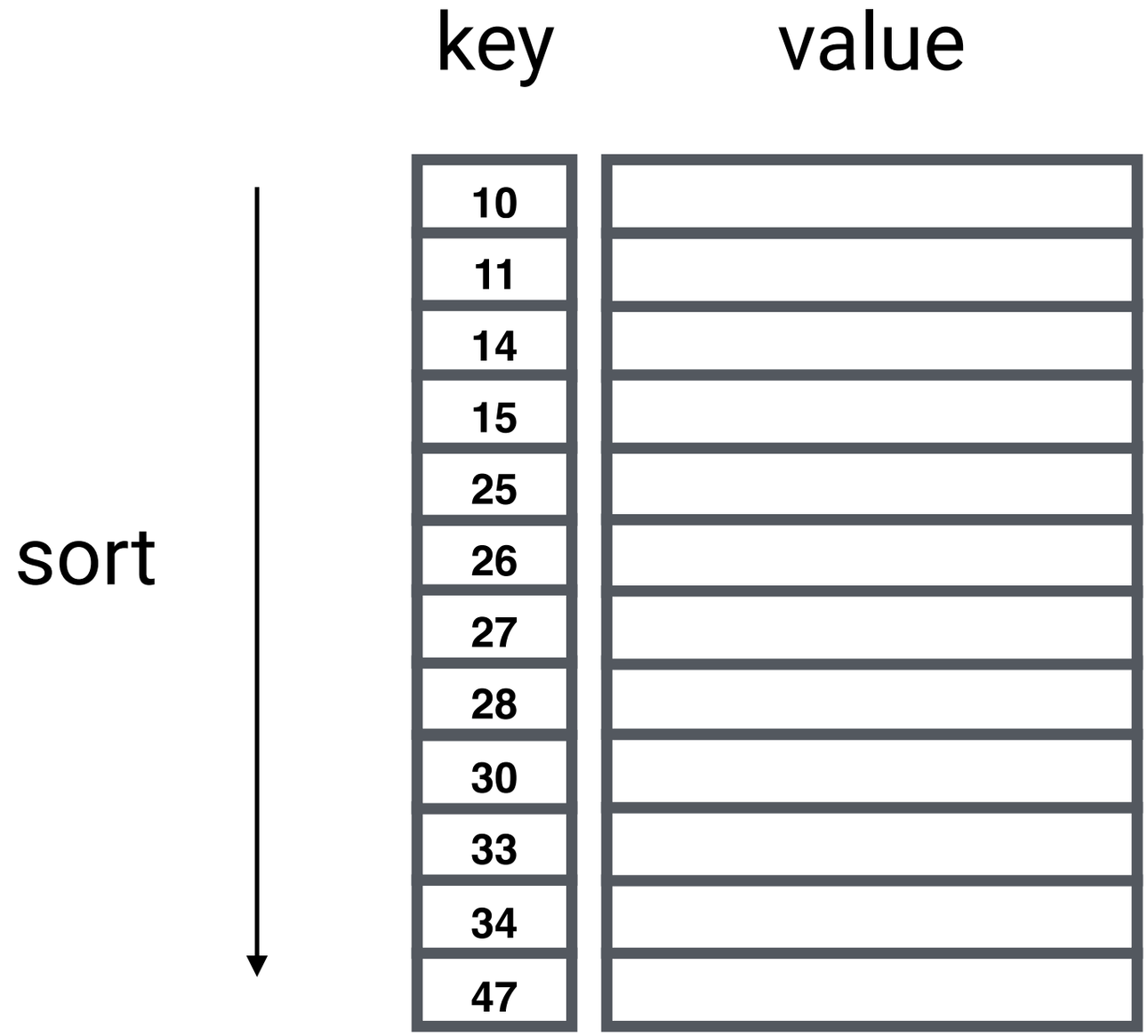
Sort by primary key

Packing



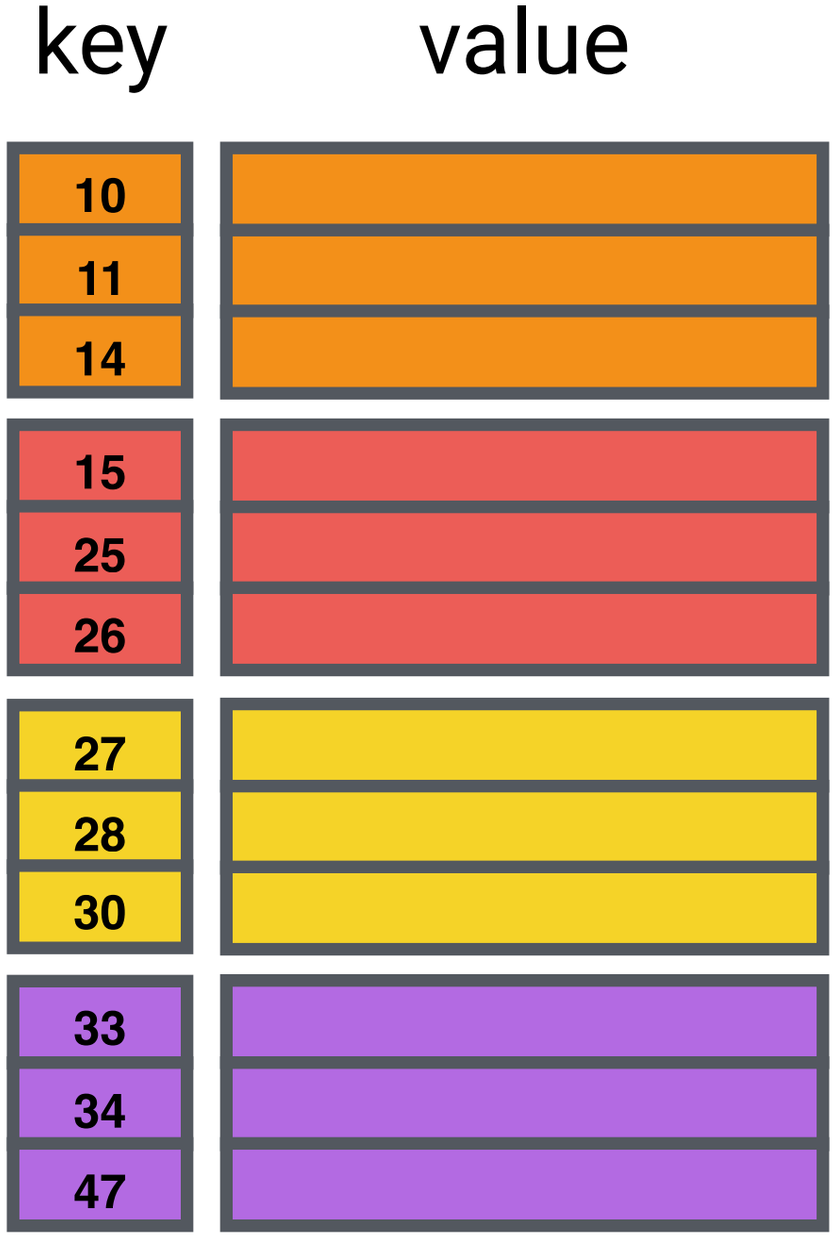
Sort by primary key

Packing



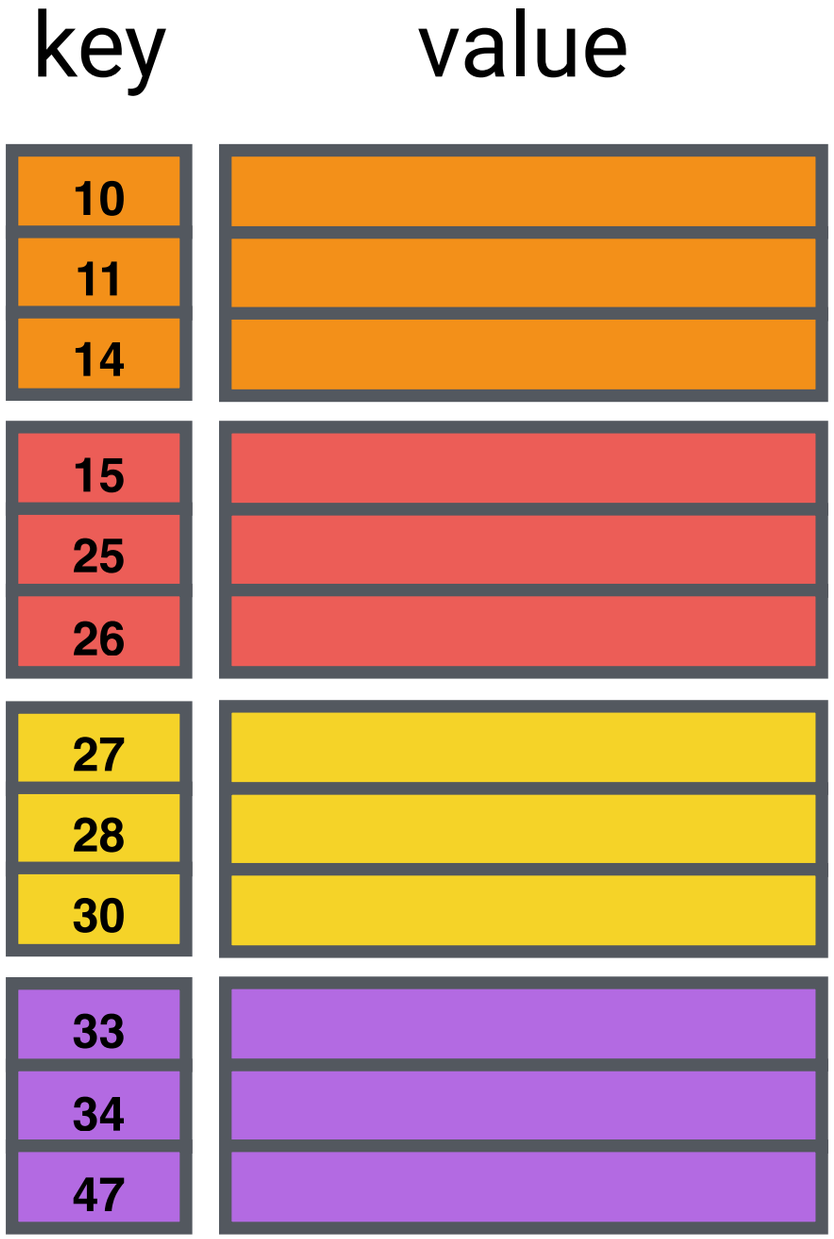
Sort by primary key

Packing

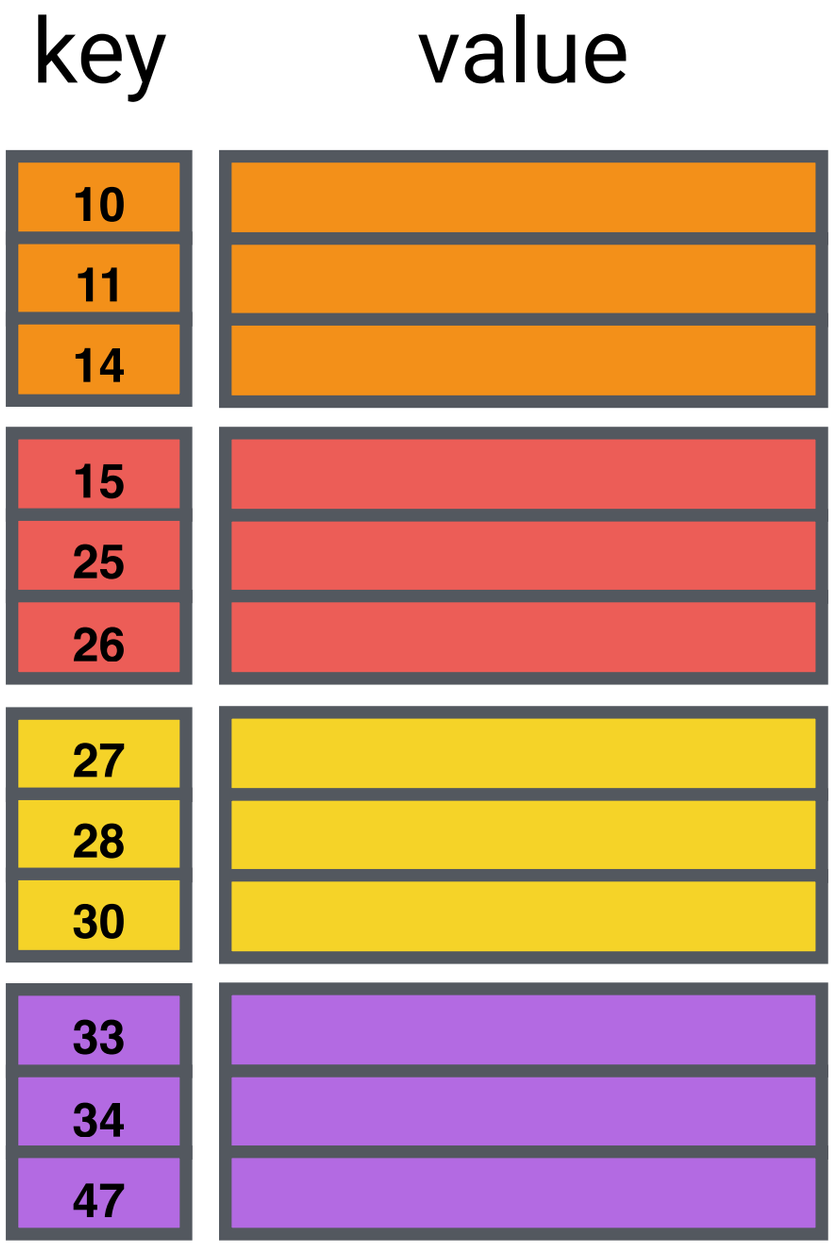


Partition into packs

Packing

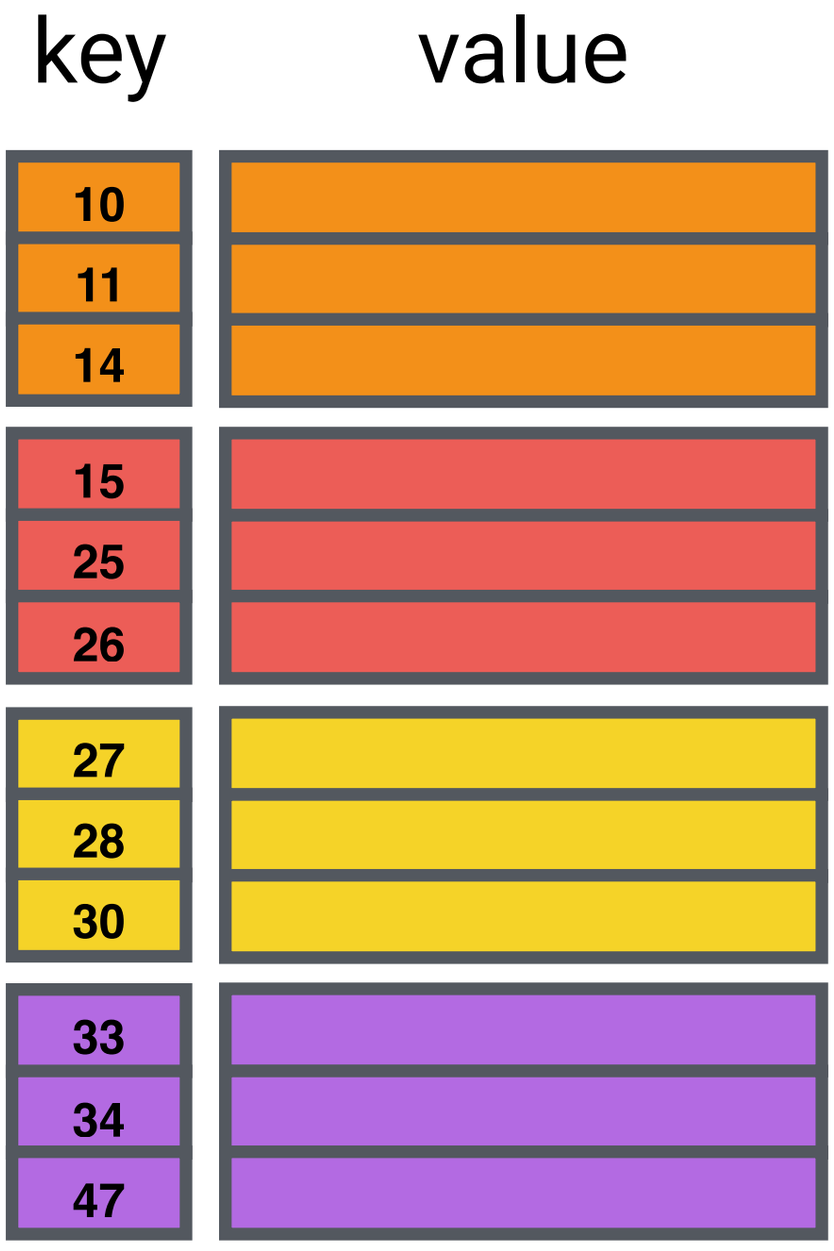


Packing



New key =
smallest key
in the pack

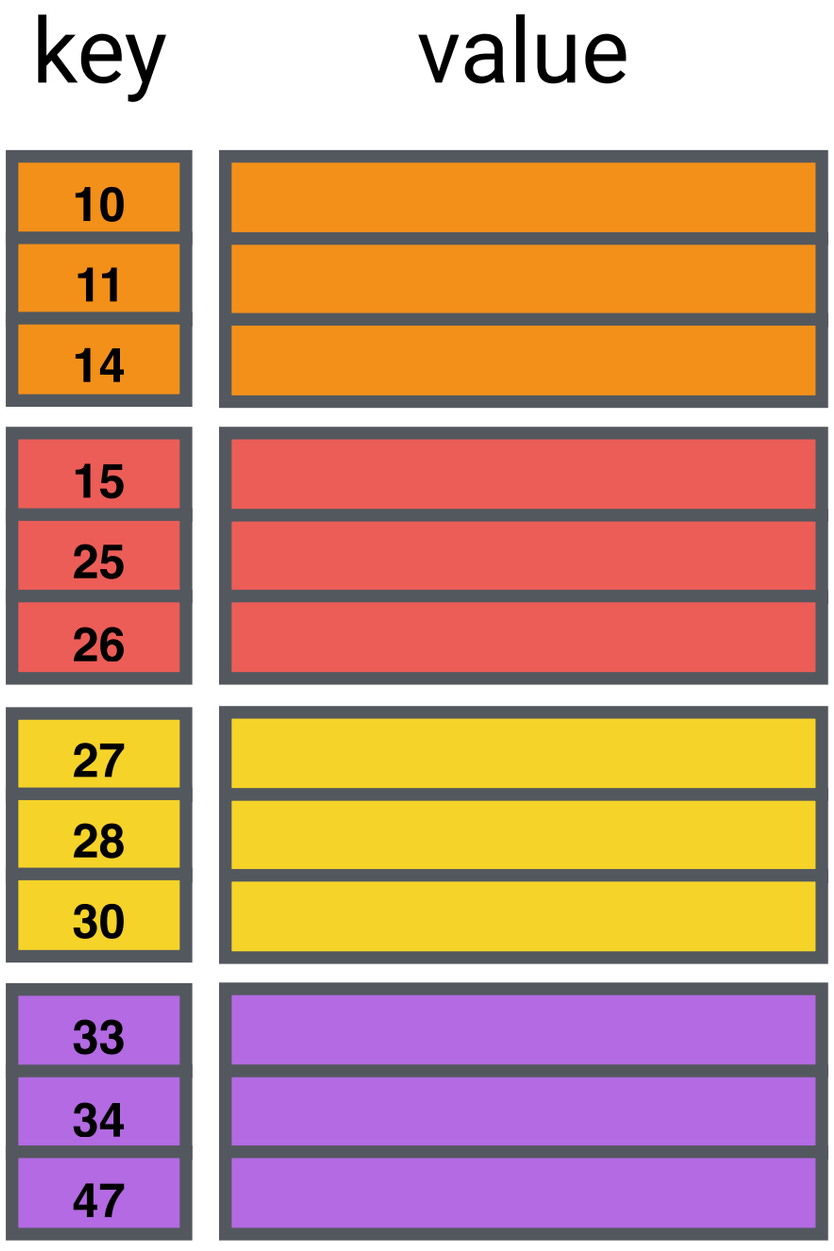
Packing



New key = smallest key in the pack

New value = a pack of key-value pairs

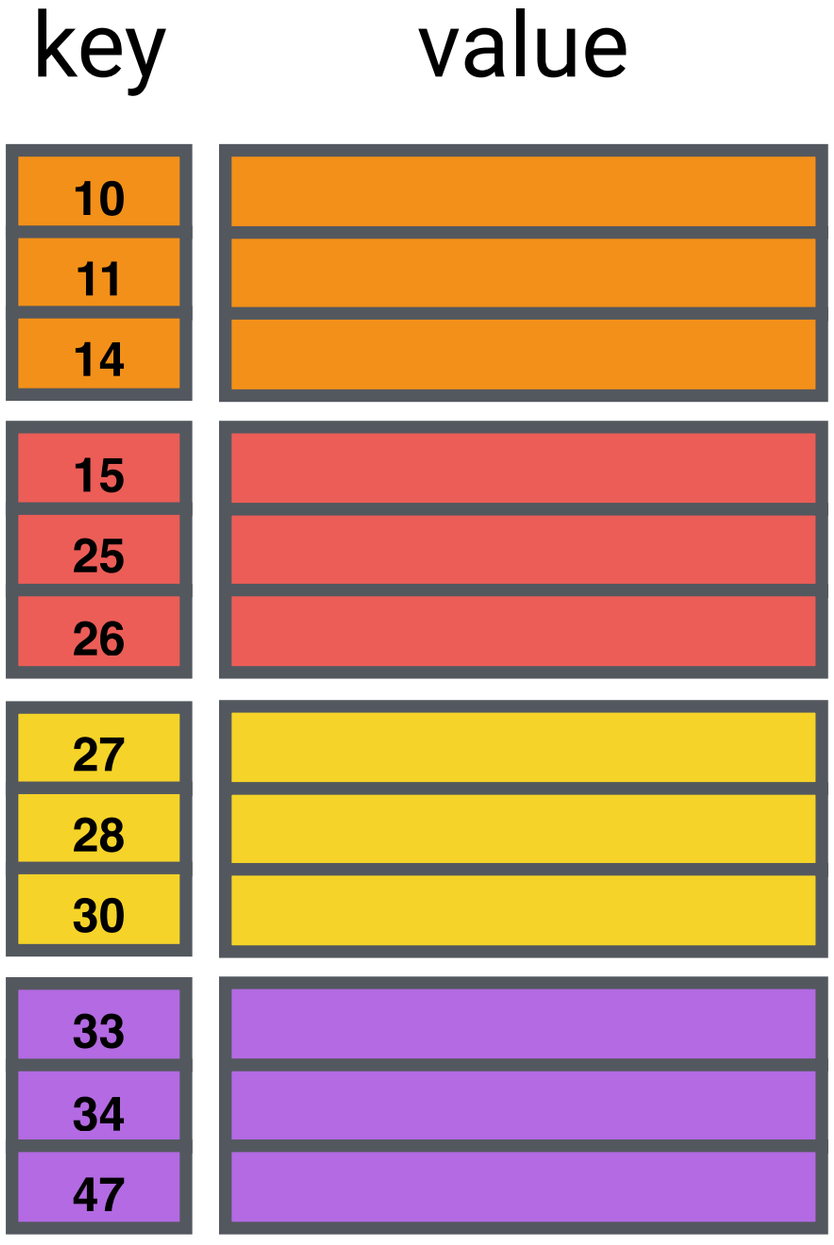
Packing



New key =
smallest key
in the pack

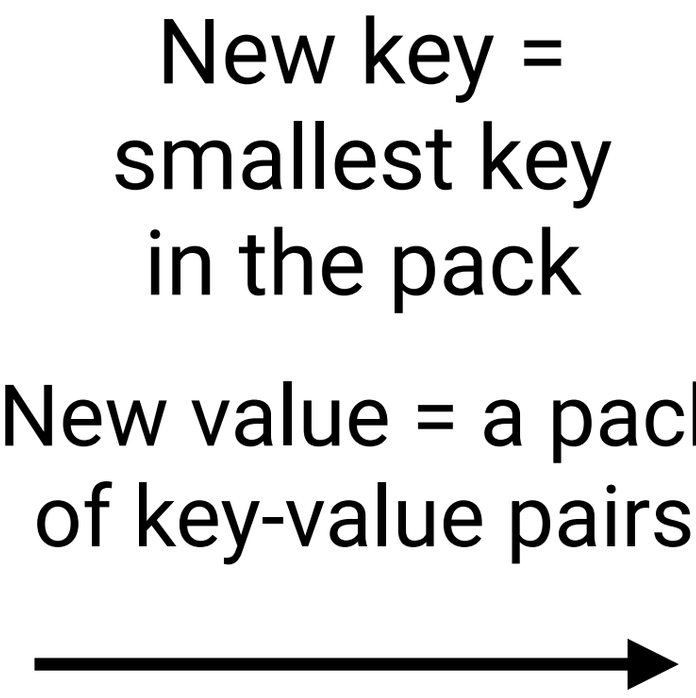
New value = a pack
of key-value pairs

Packing



New key = smallest key in the pack

New value = a pack of key-value pairs

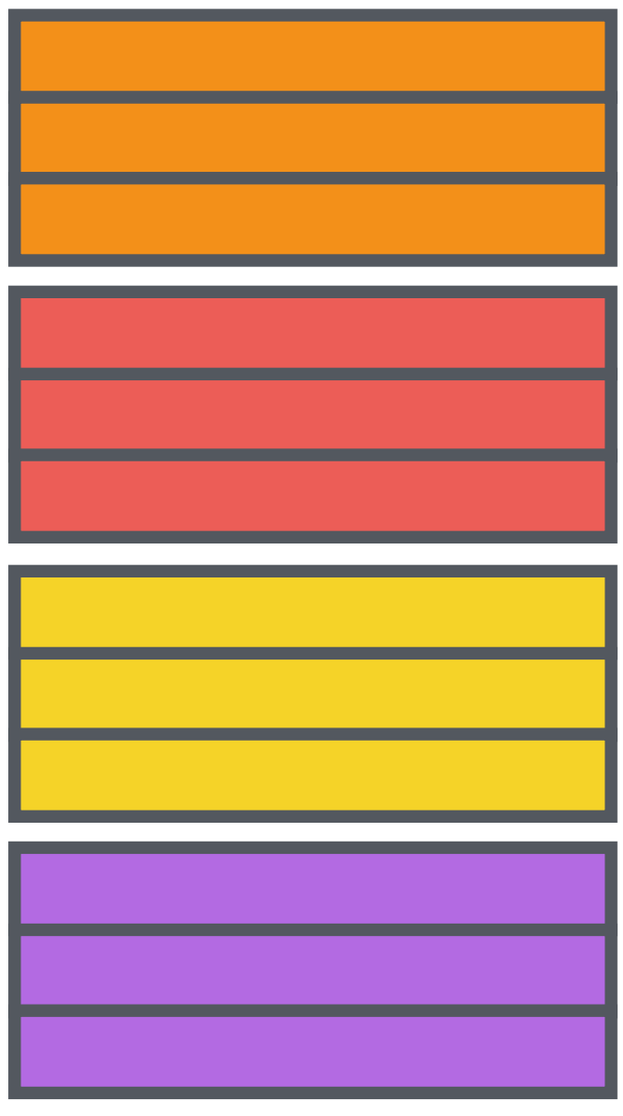


10:

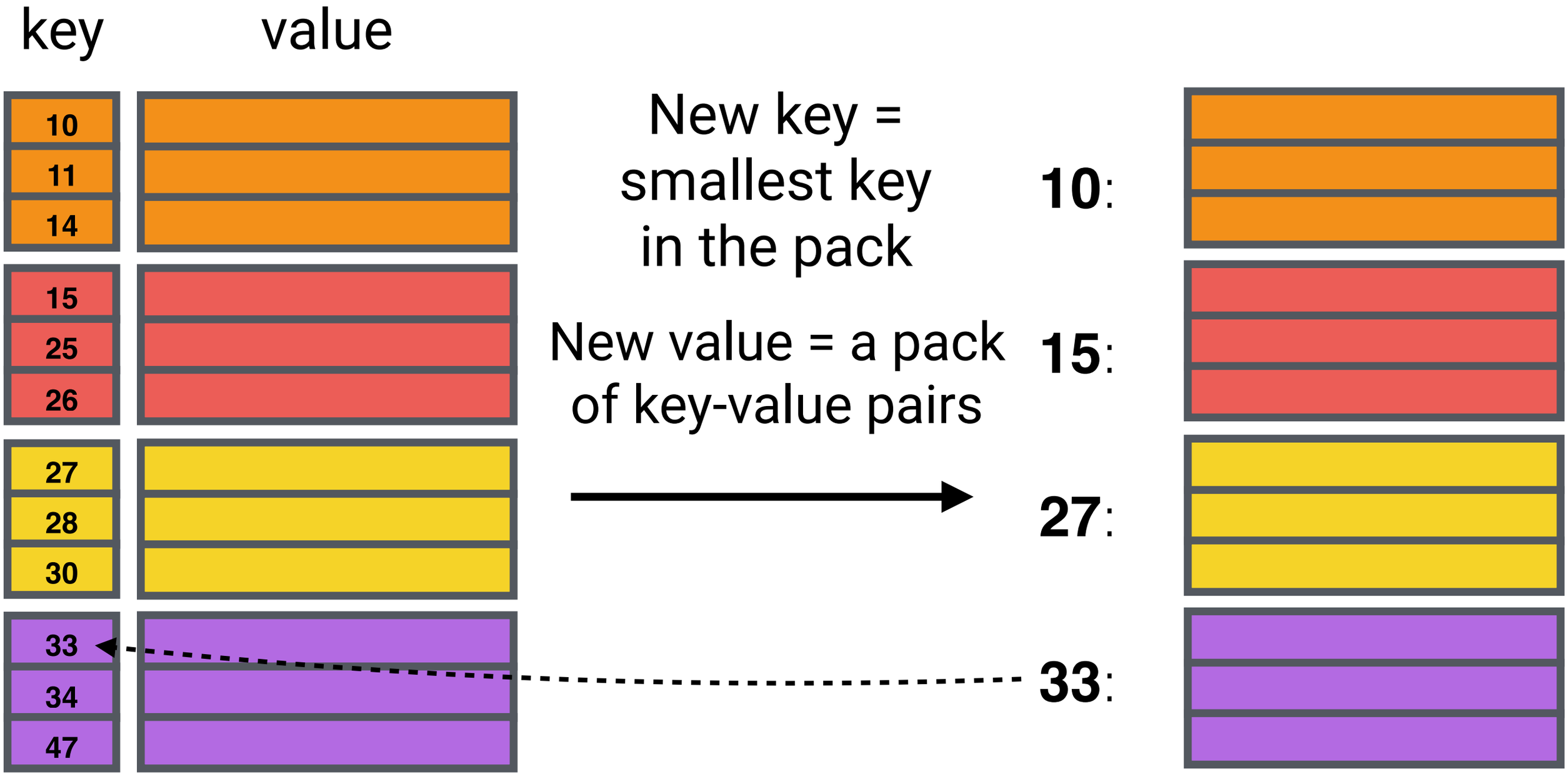
15:

27:

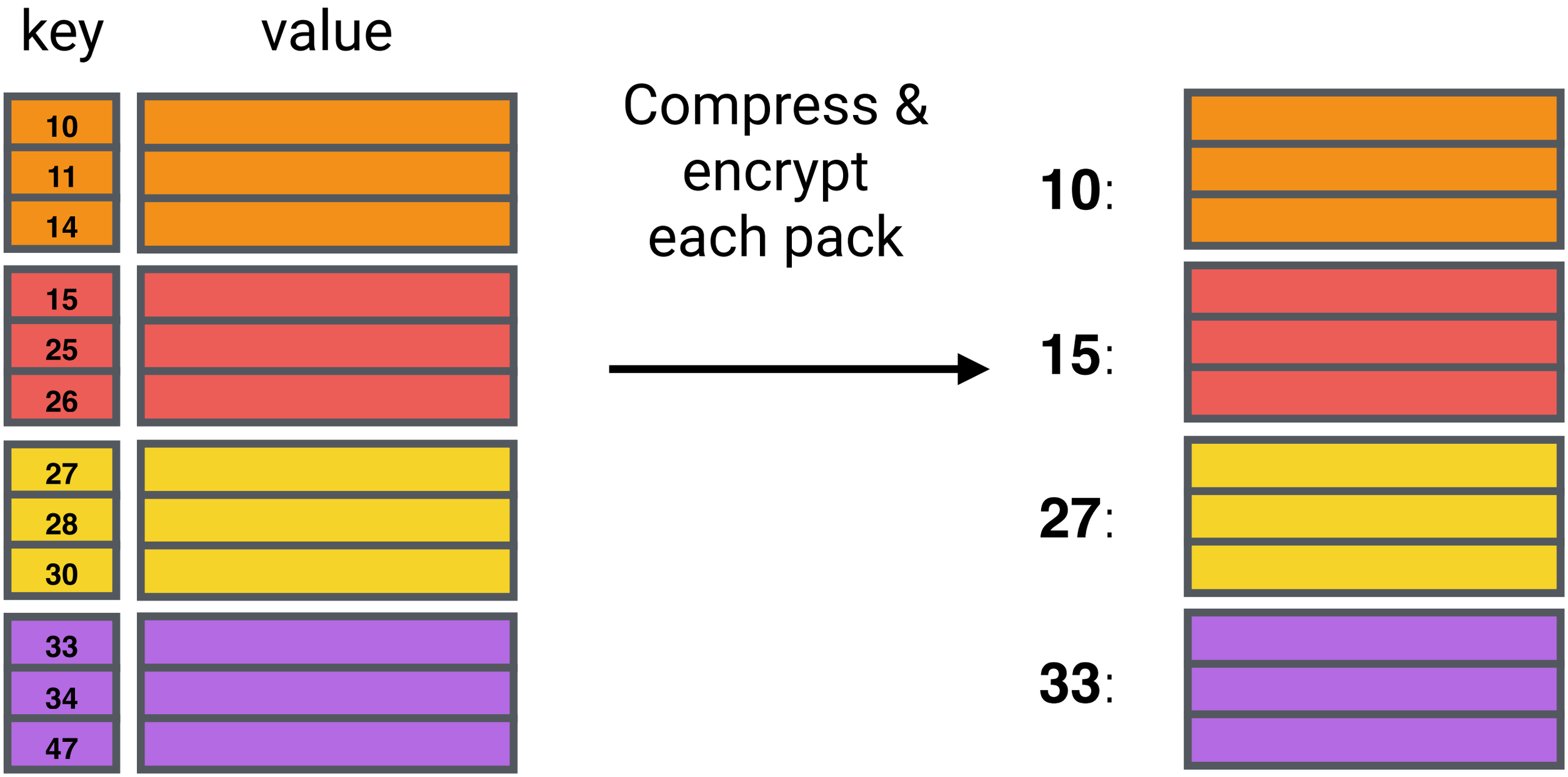
33:



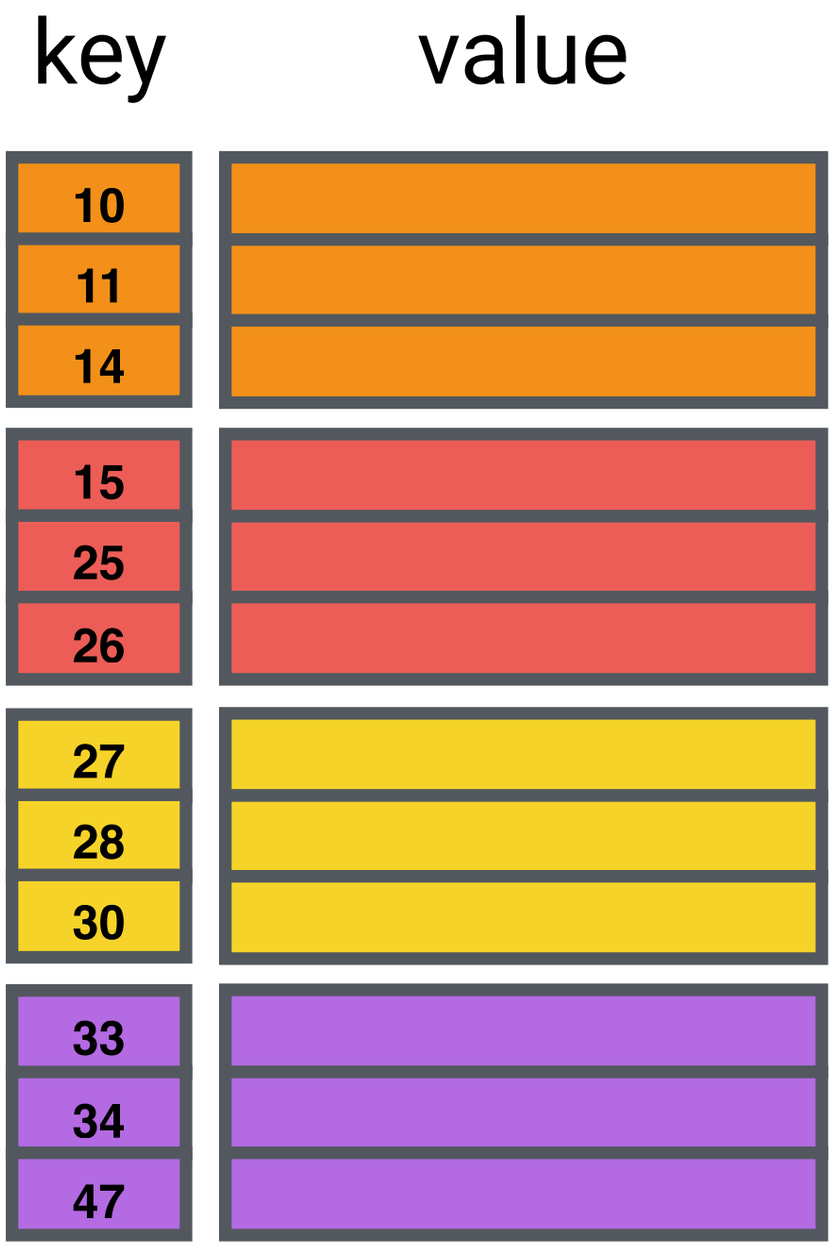
Packing


key
value
10
11
14
15
25
26
27
28
30
33
34
47
New key = smallest key in the pack
New value = a pack of key-value pairs
10:
15:
27:
33:

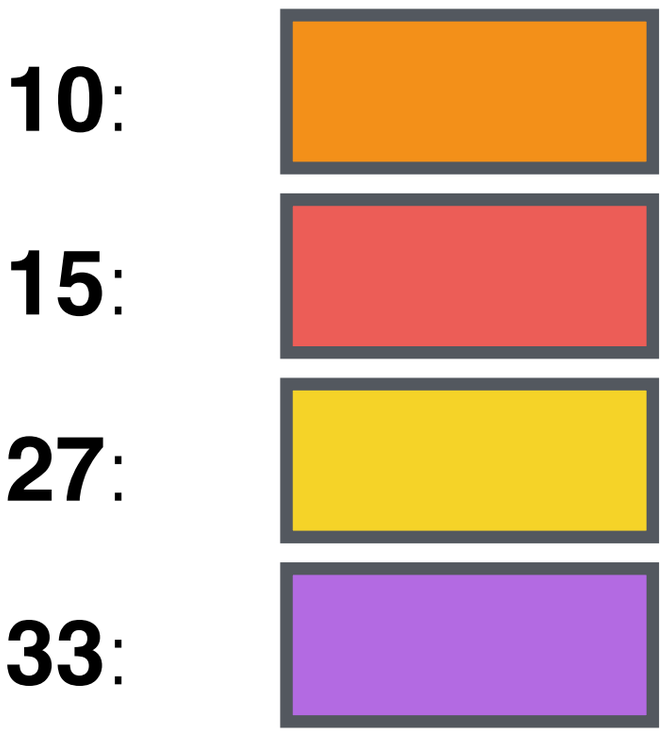
Packing


key
value
10
11
14
15
25
26
27
28
30
33
34
47
Compress &
encrypt
each pack
10:
15:
27:
33:

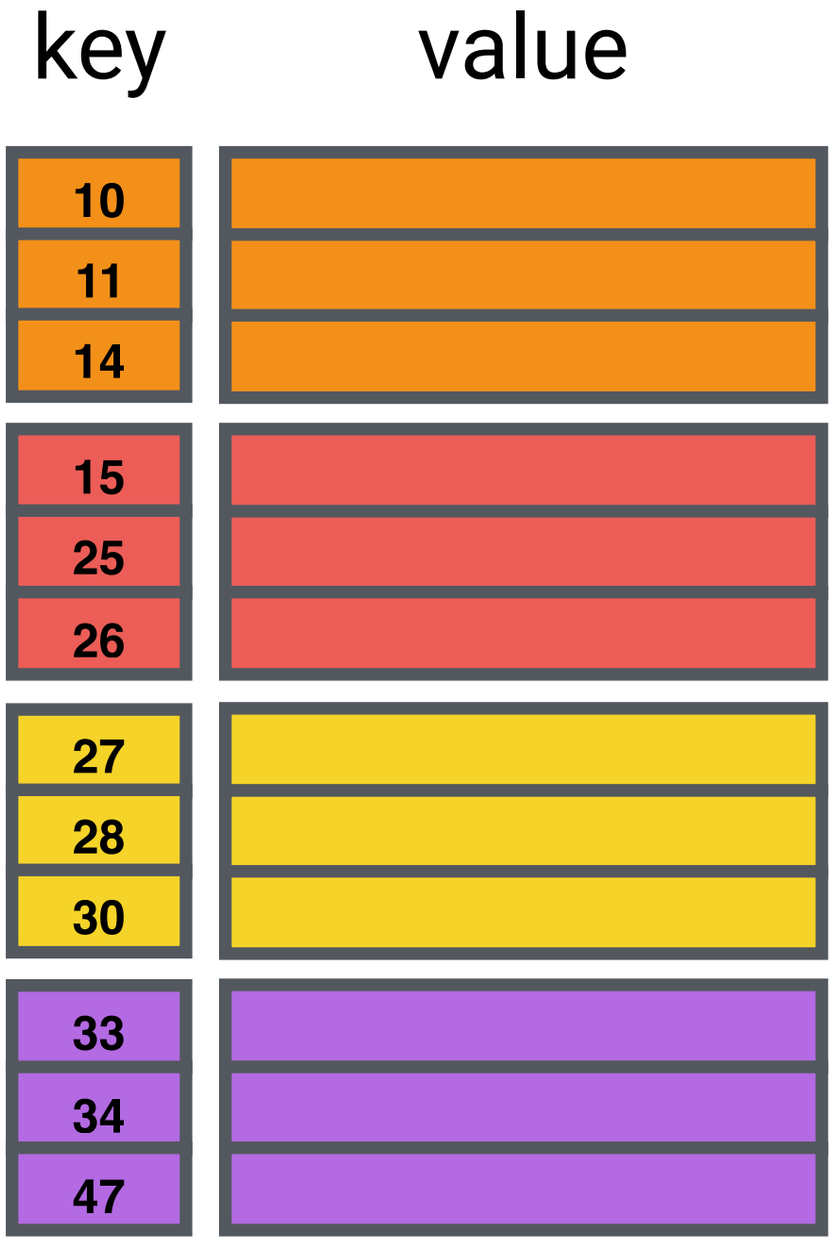
Packing



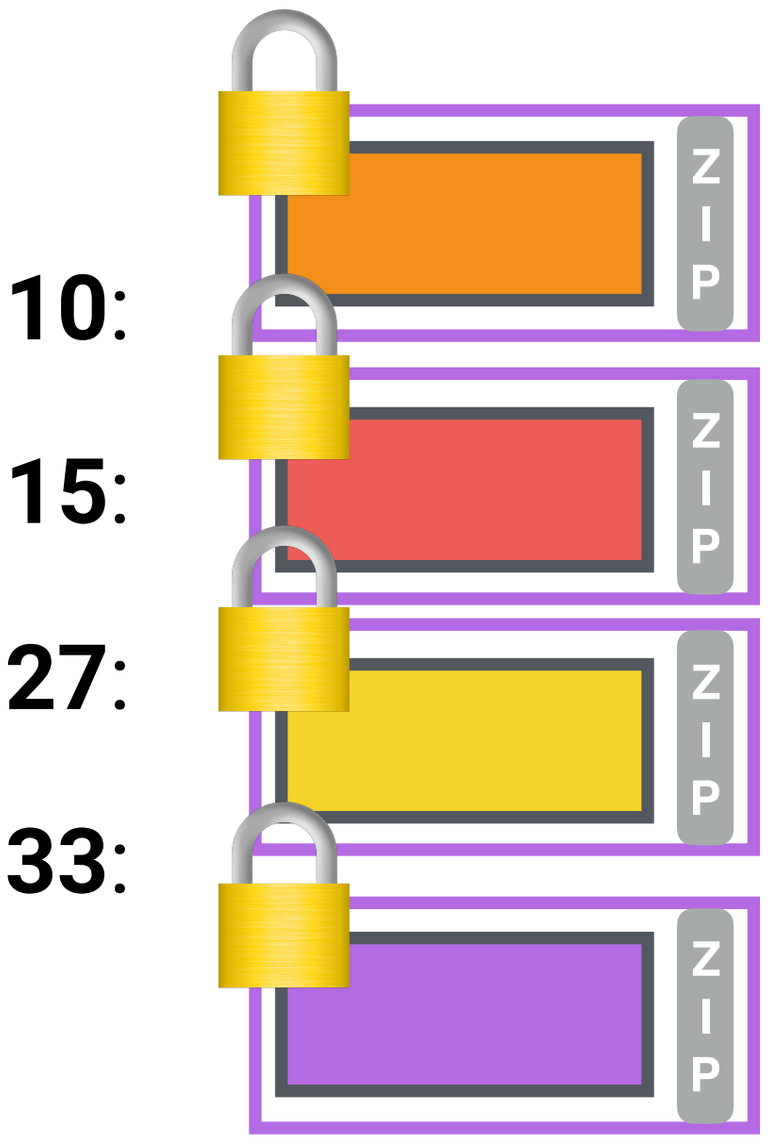
Compress & encrypt each pack



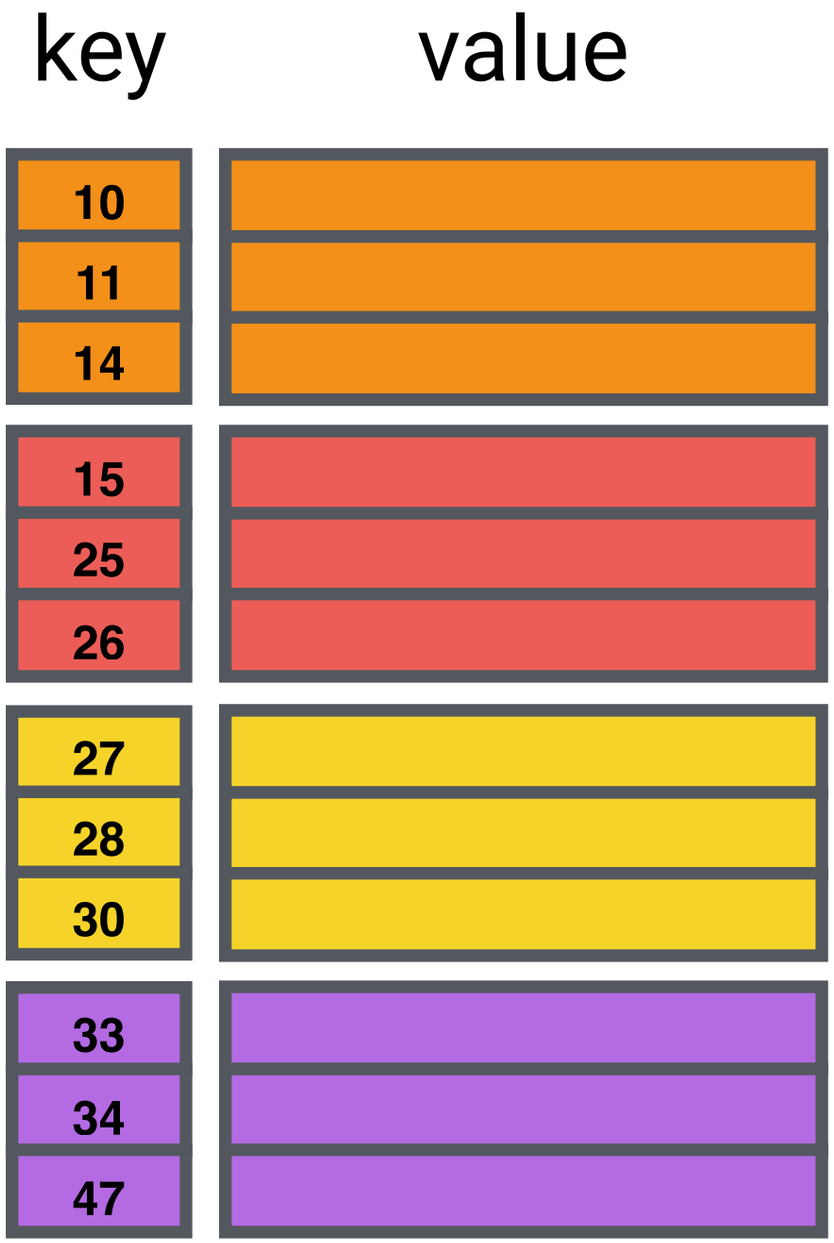
Packing



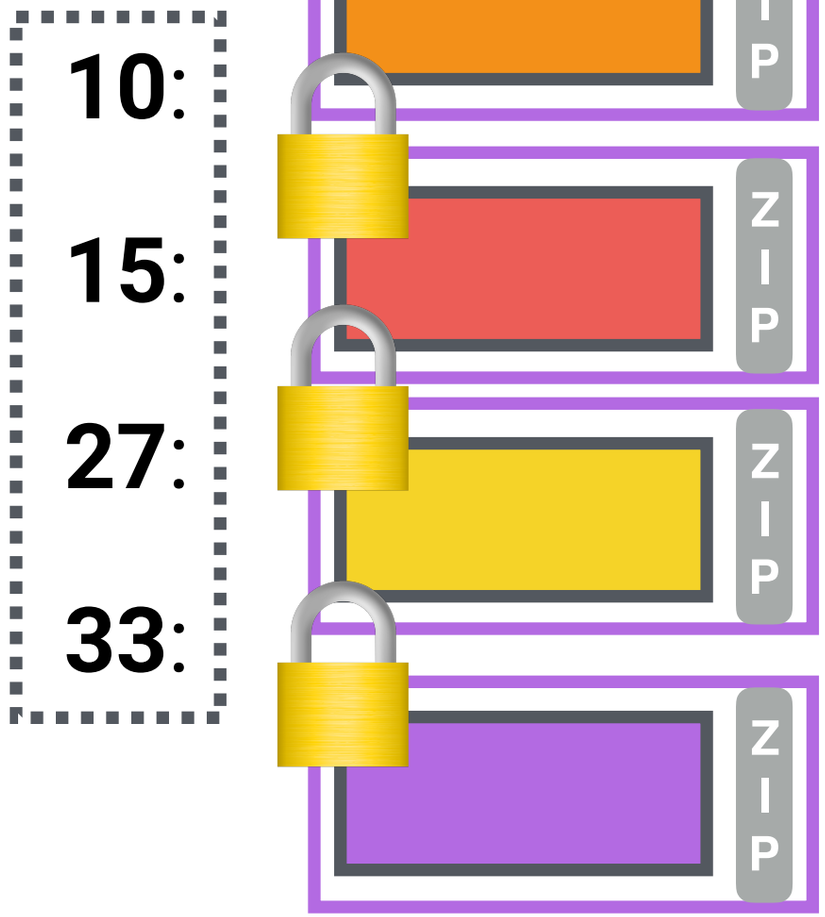
Compress & encrypt each pack



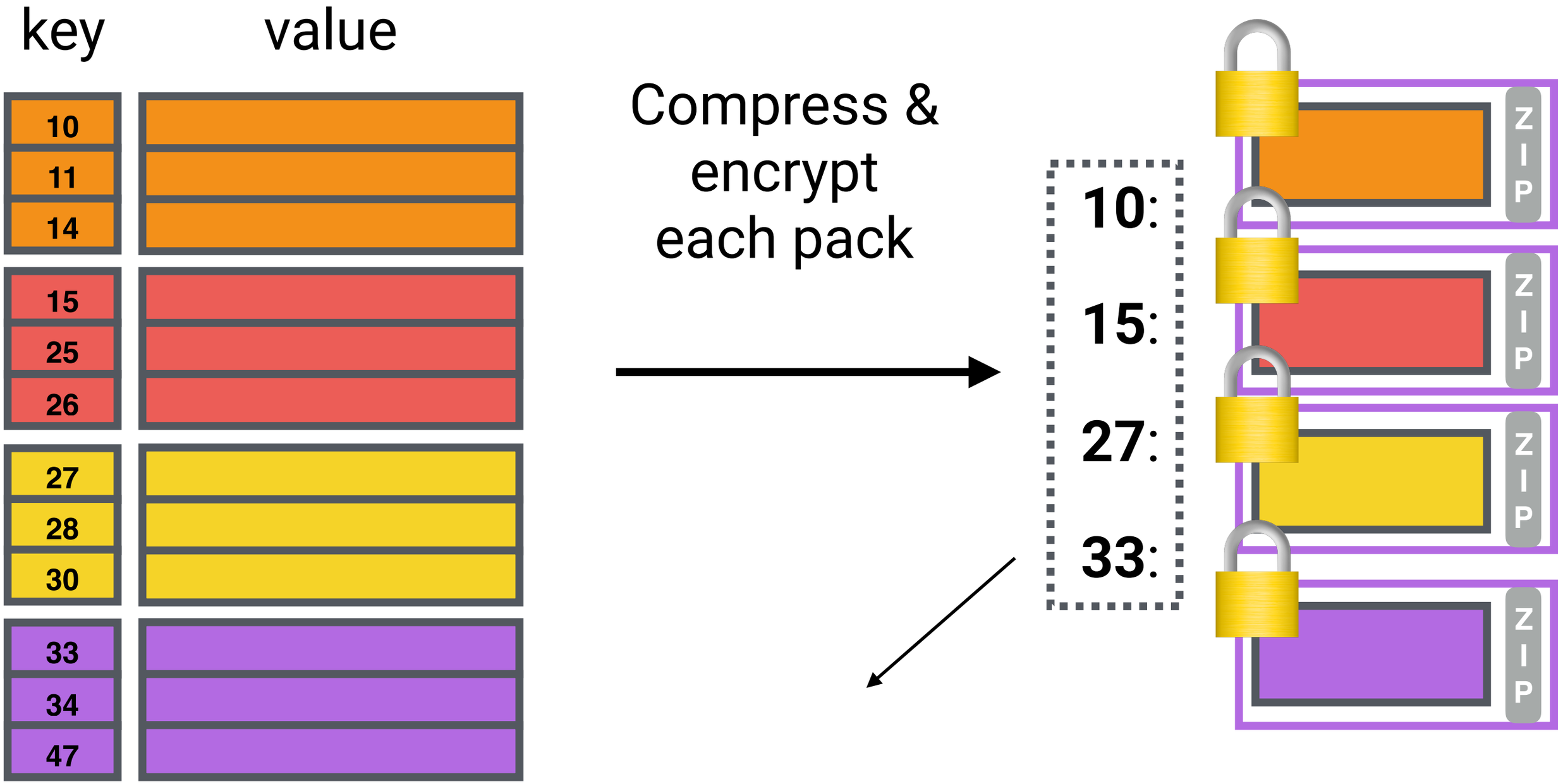
Packing



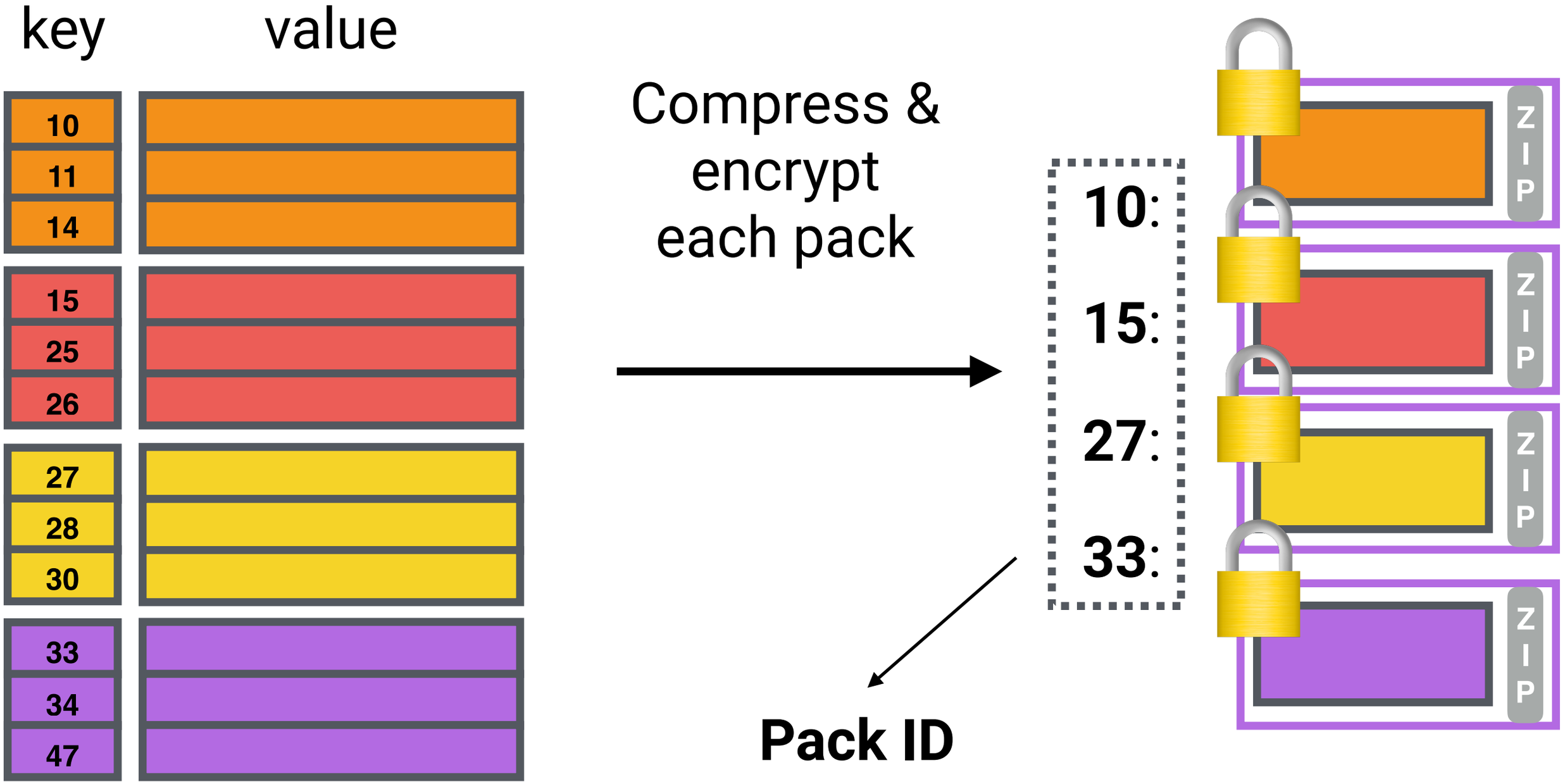
Compress & encrypt each pack



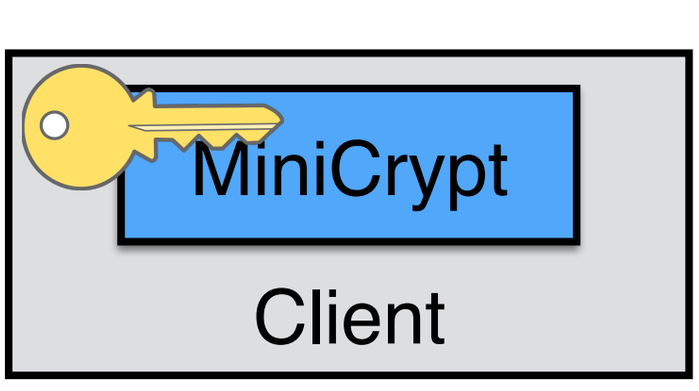
Packing

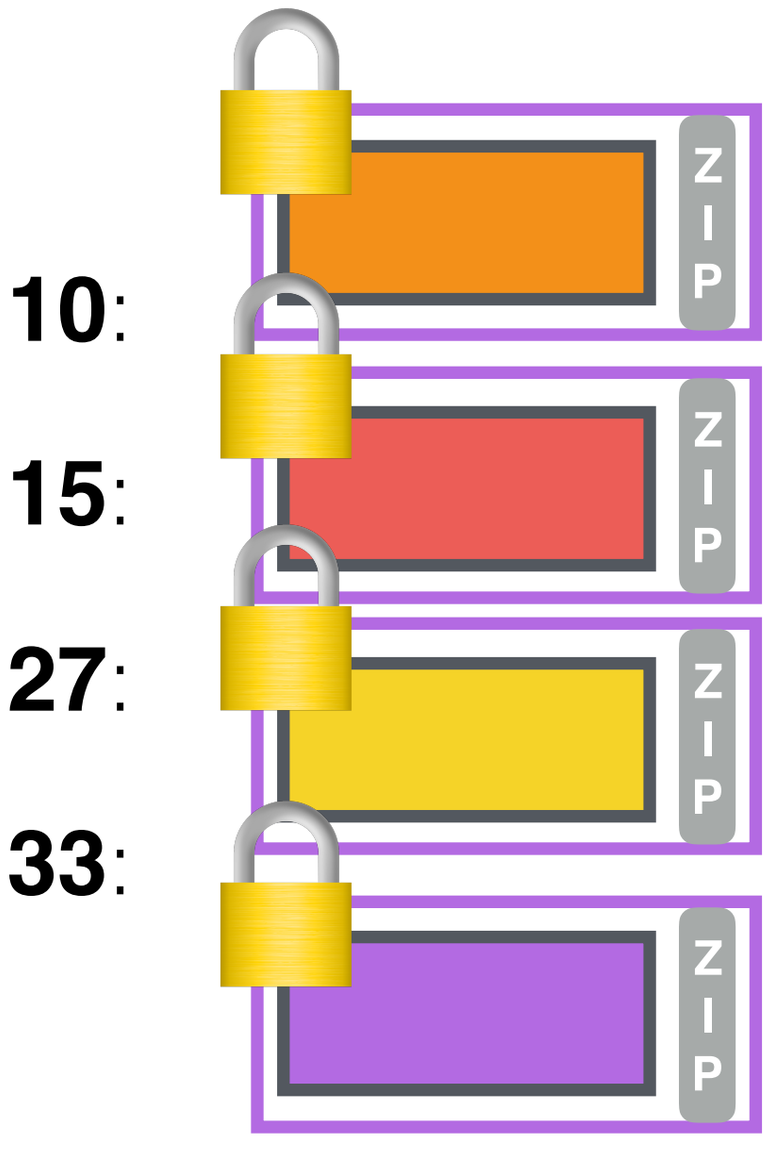

key
value
10
11
14
15
25
26
27
28
30
33
34
47
Compress &
encrypt
each pack
10:
15:
27:
33:
ZIP
ZIP
ZIP
ZIP

Packing


key
value
10
11
14
15
25
26
27
28
30
33
34
47
Compress &
encrypt
each pack
10:
15:
27:
33:
ZIP
ZIP
ZIP
ZIP
Pack ID

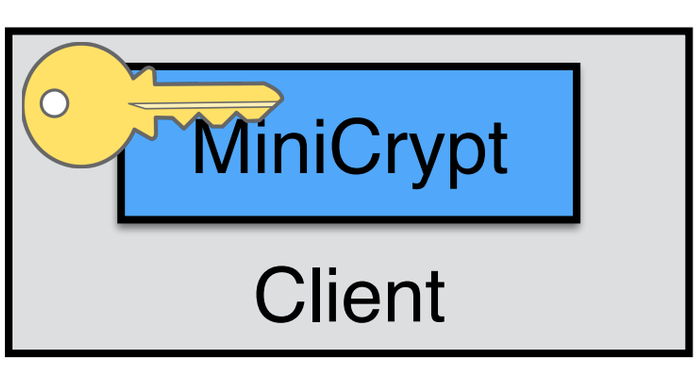
Single-key get



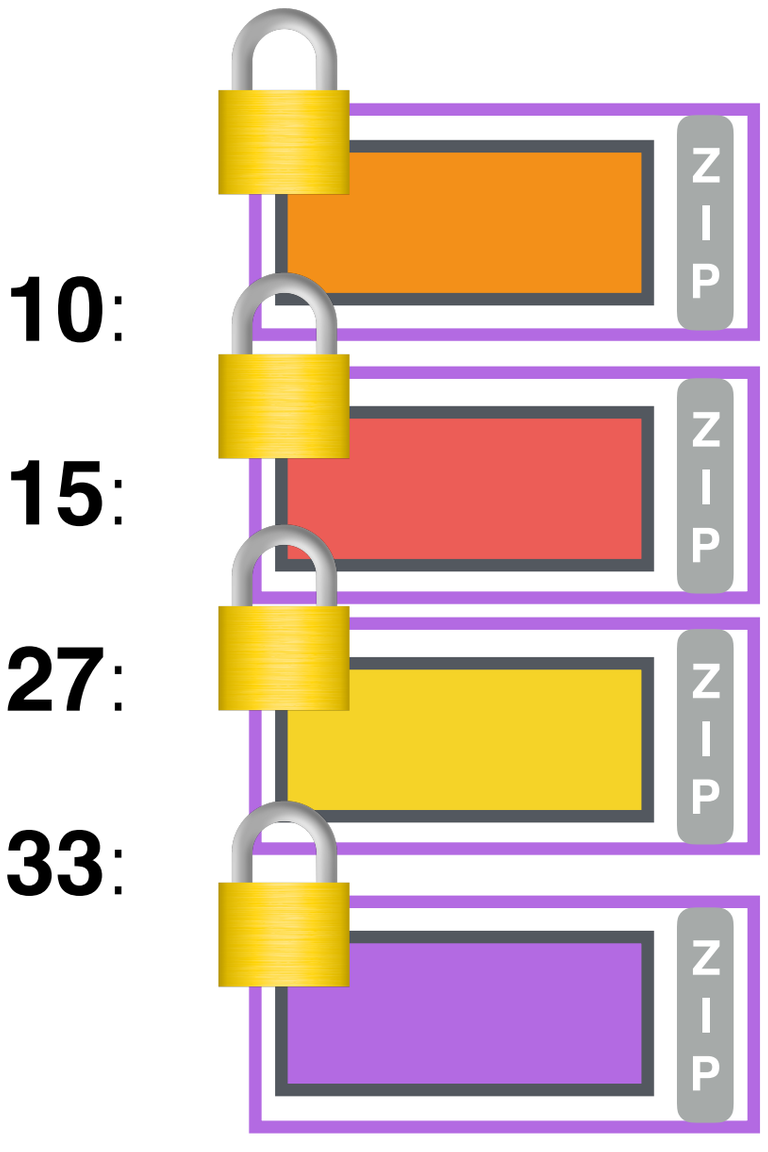


get(*key*): select pack with **highest pack ID** from all pack ID's that are at most *key*

Single-key get

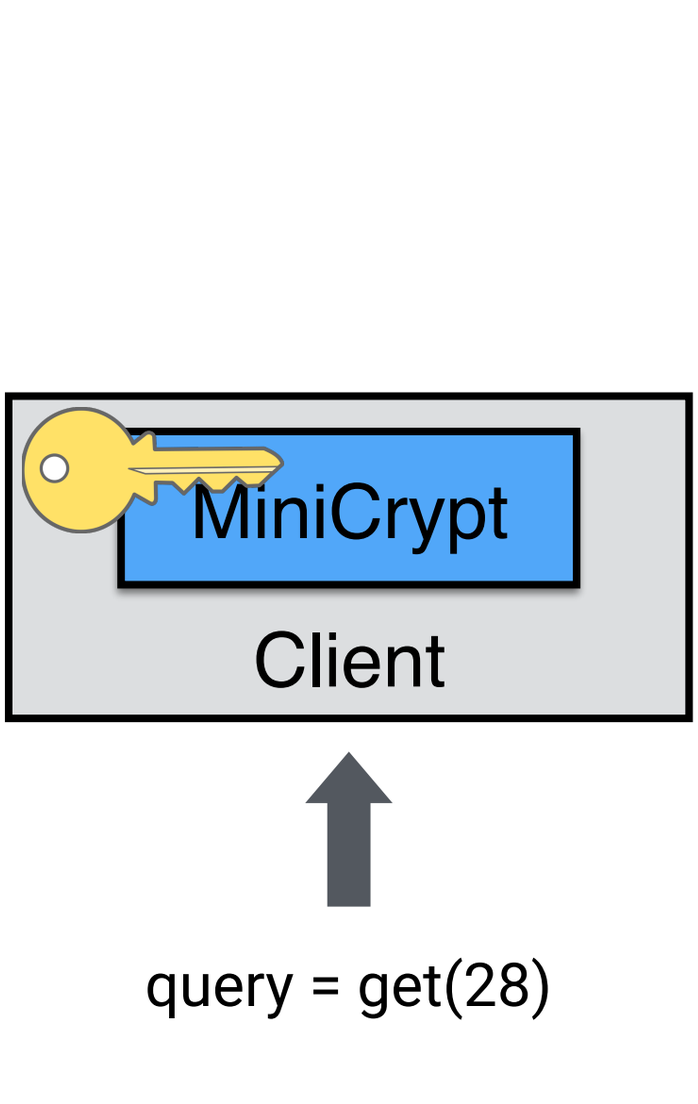


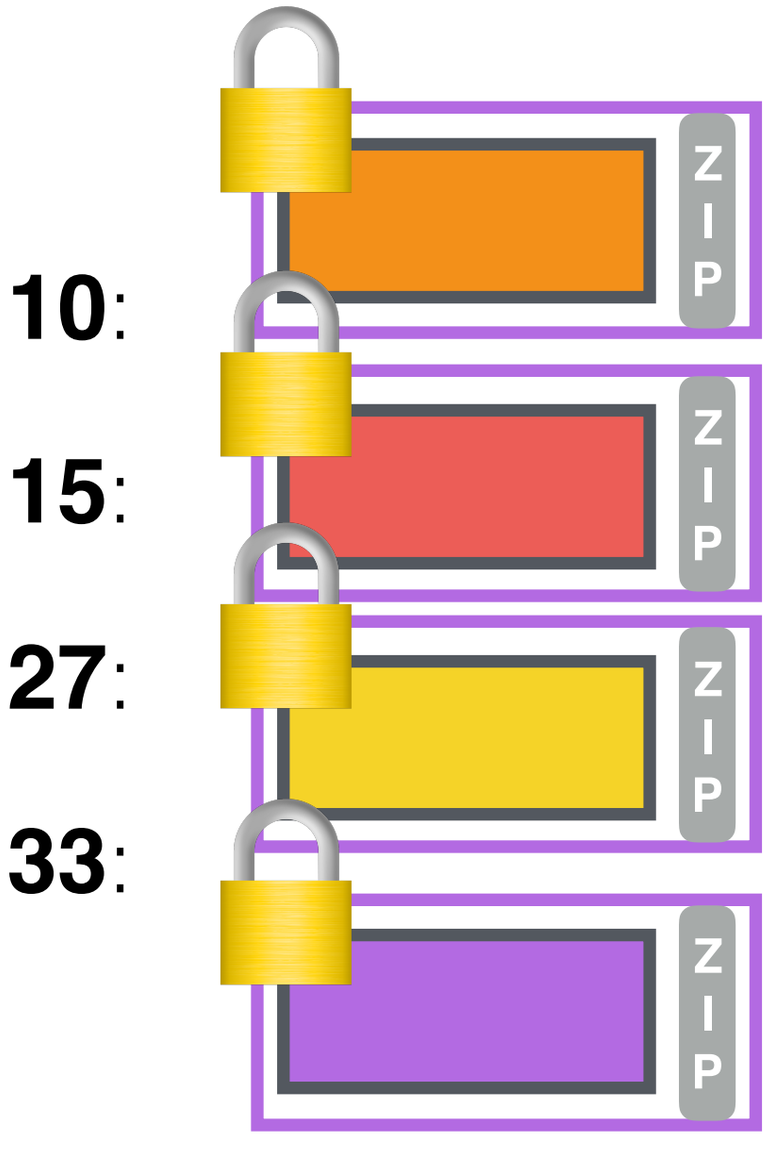
query = get(28)



get(*key*): select pack with **highest pack ID** from all pack ID's that are at most *key*

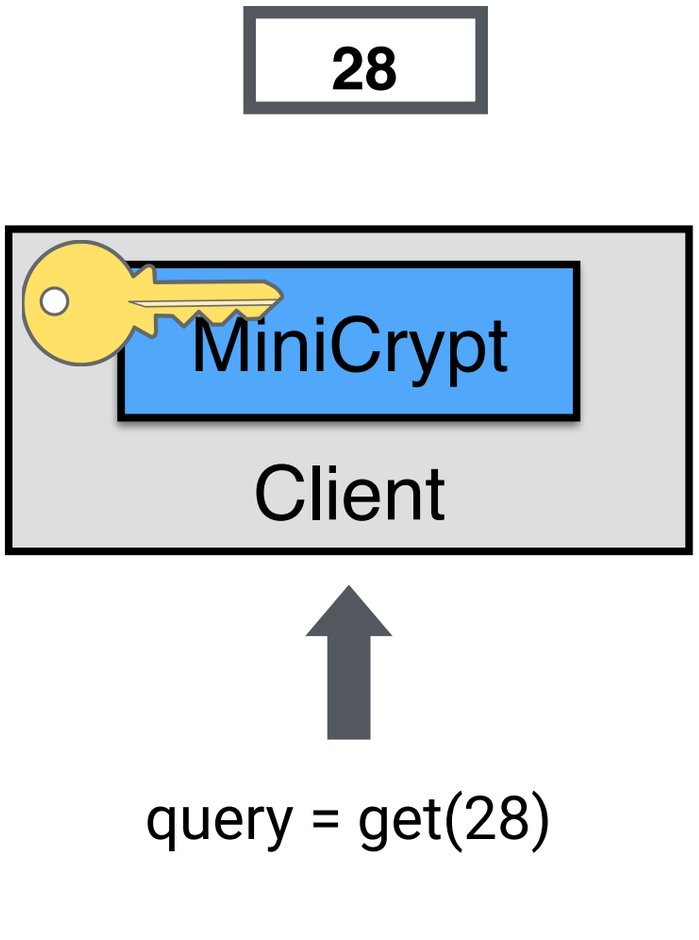
Single-key get

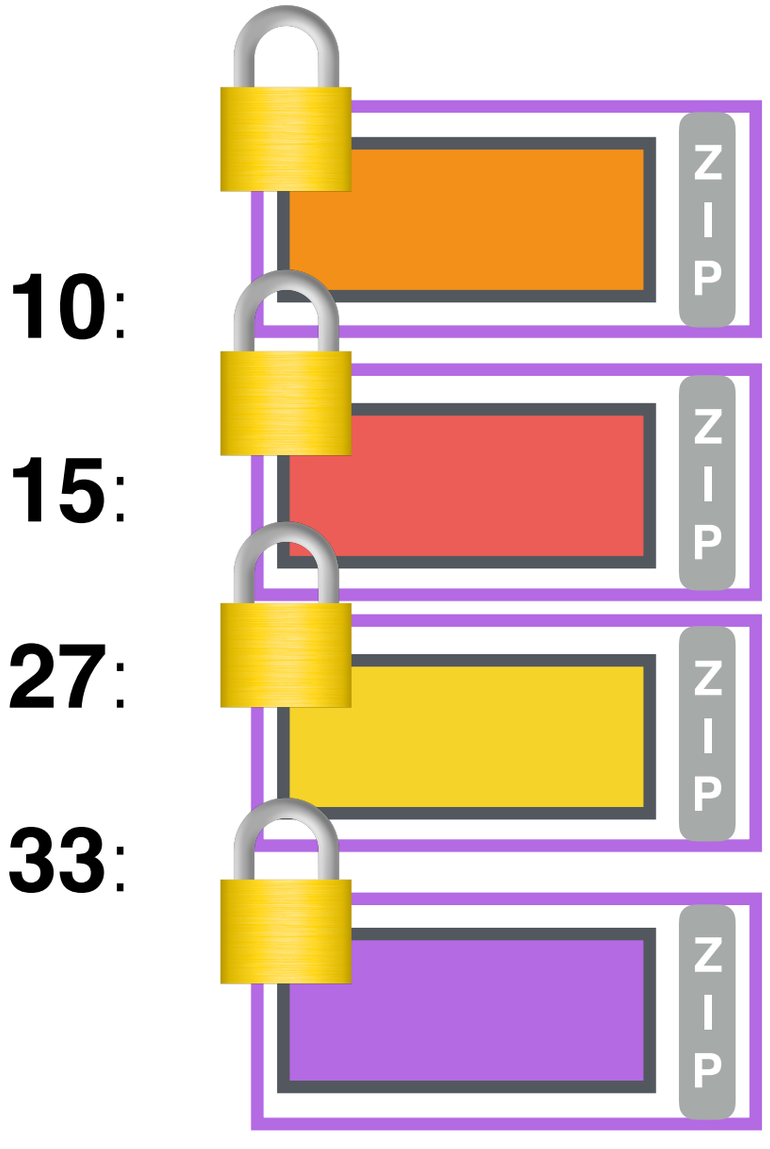




get(*key*): select pack with **highest pack ID** from all pack ID's that are at most *key*

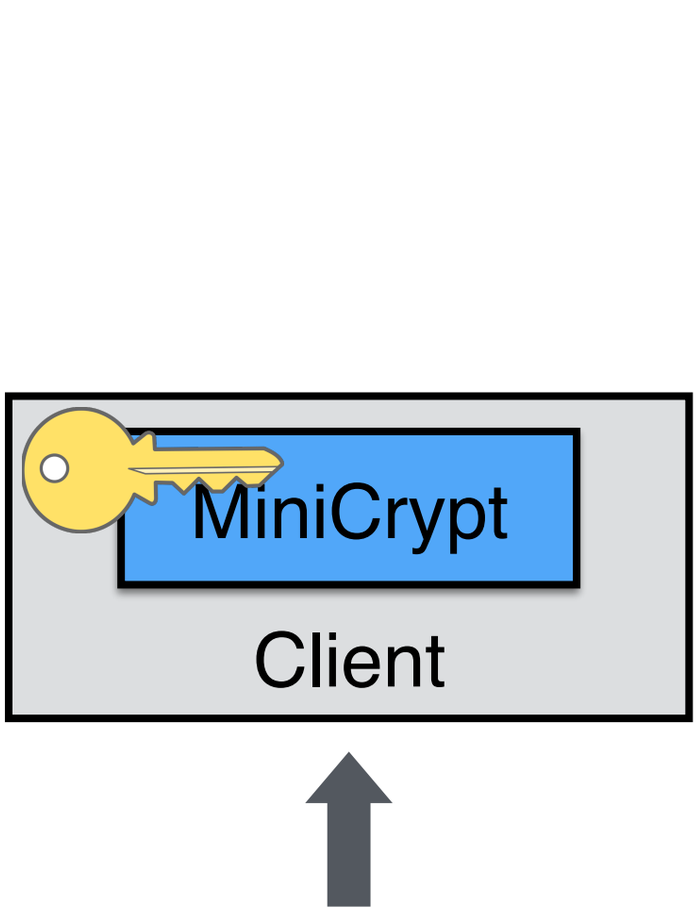
Single-key get

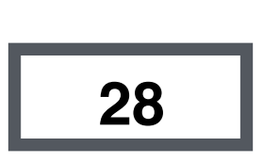


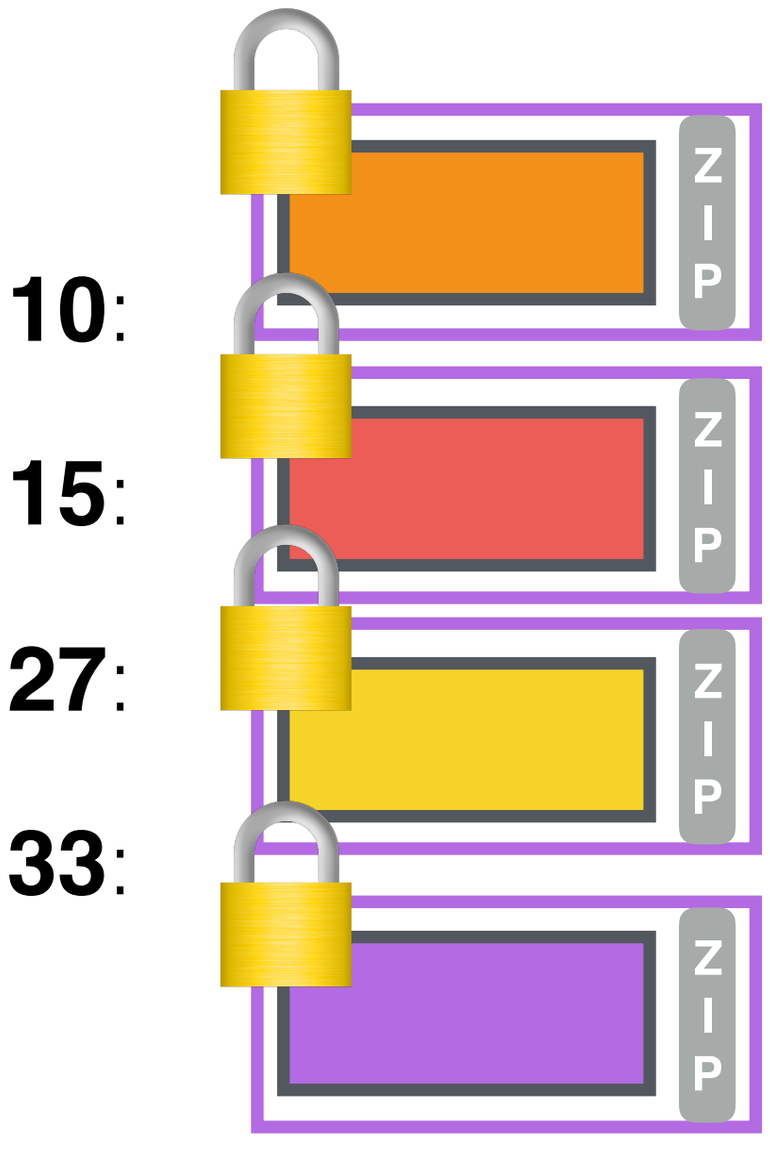


get(*key*): select pack with **highest pack ID** from all pack ID's that are at most *key*

Single-key get

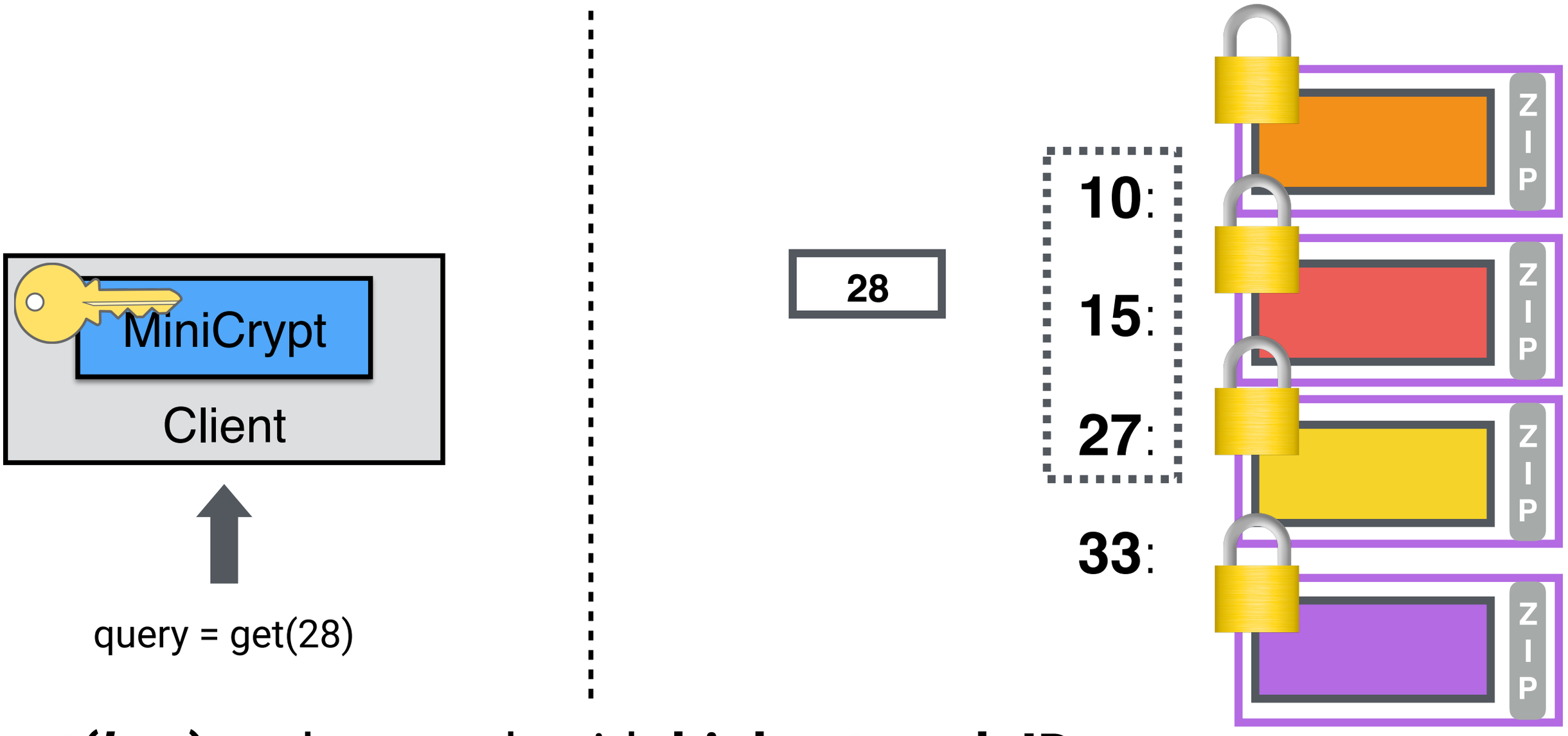






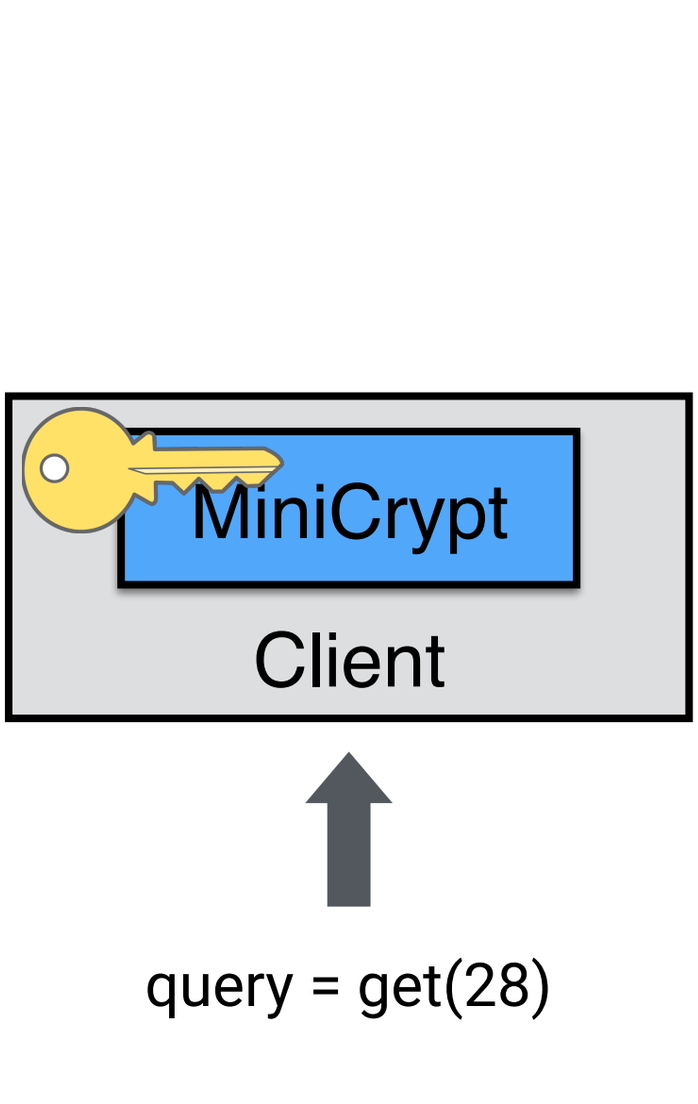
get(*key*): select pack with **highest pack ID** from all pack ID's that are at most *key*

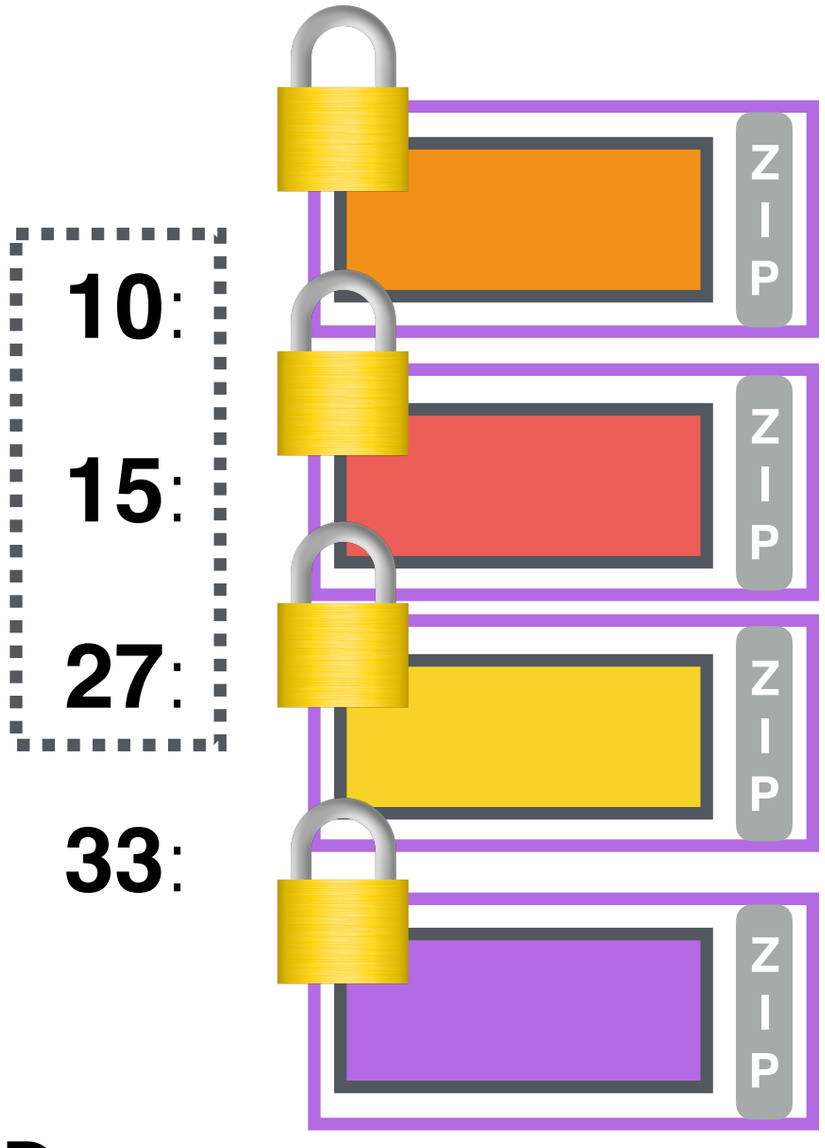
Single-key get



get(*key*): select pack with **highest pack ID** from all pack ID's that are at most *key*

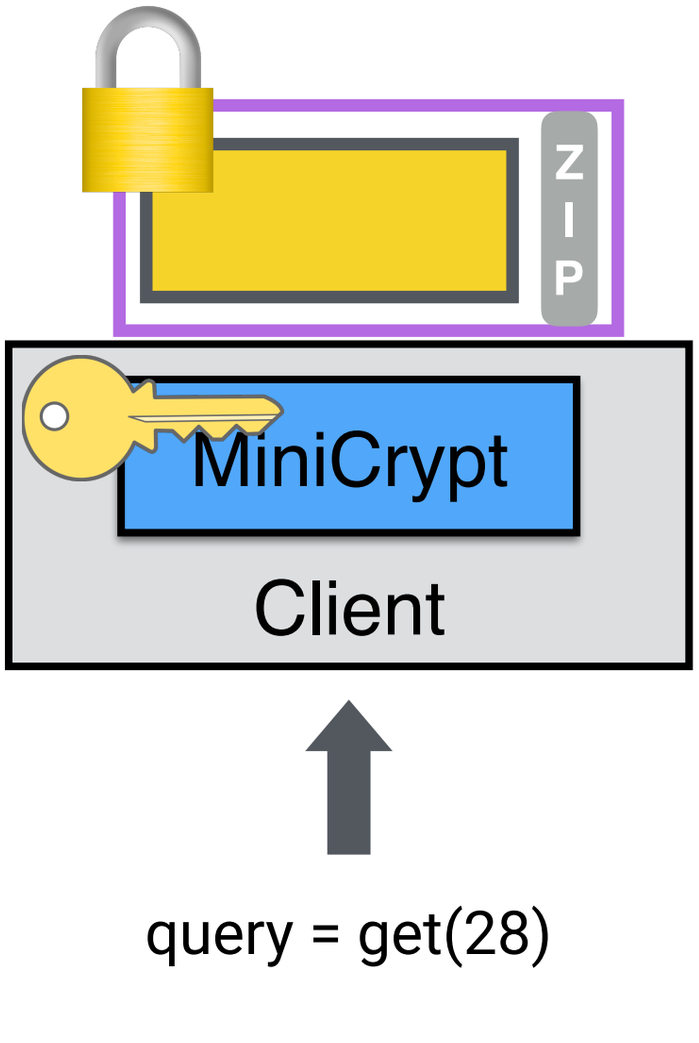
Single-key get

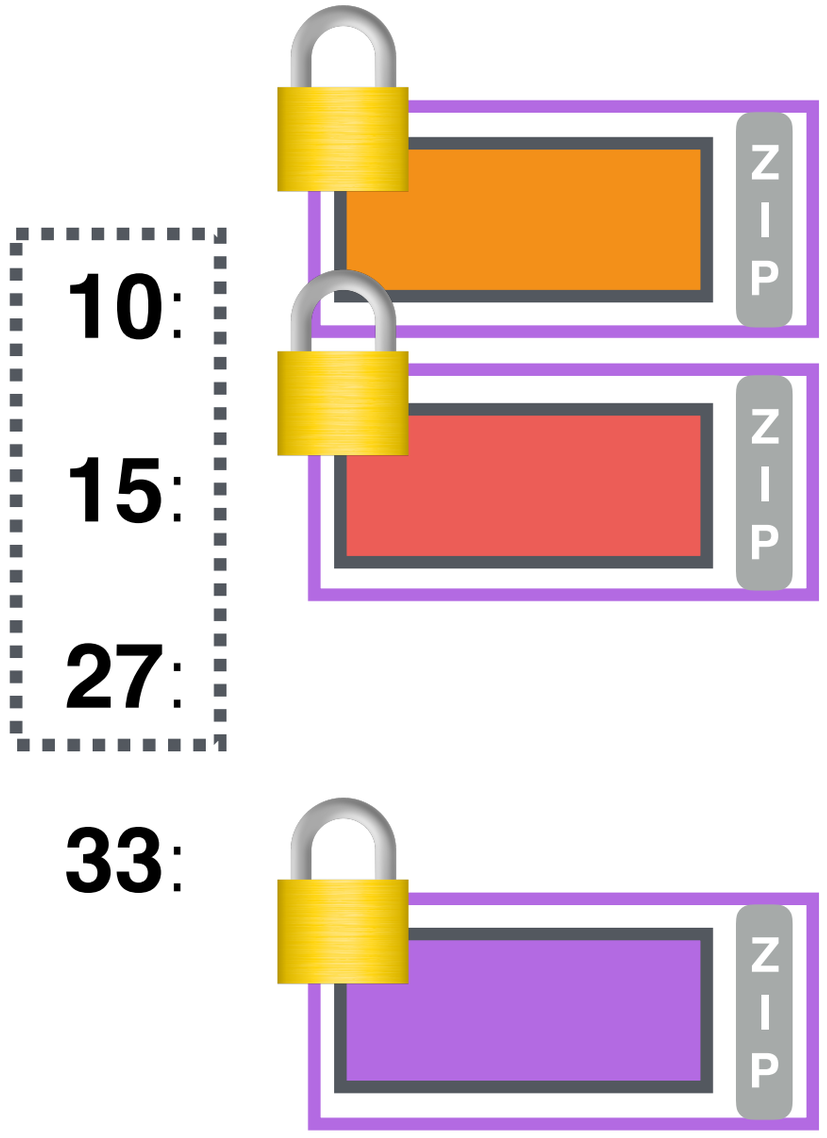




get(*key*): select pack with **highest pack ID** from all pack ID's that are at most *key*

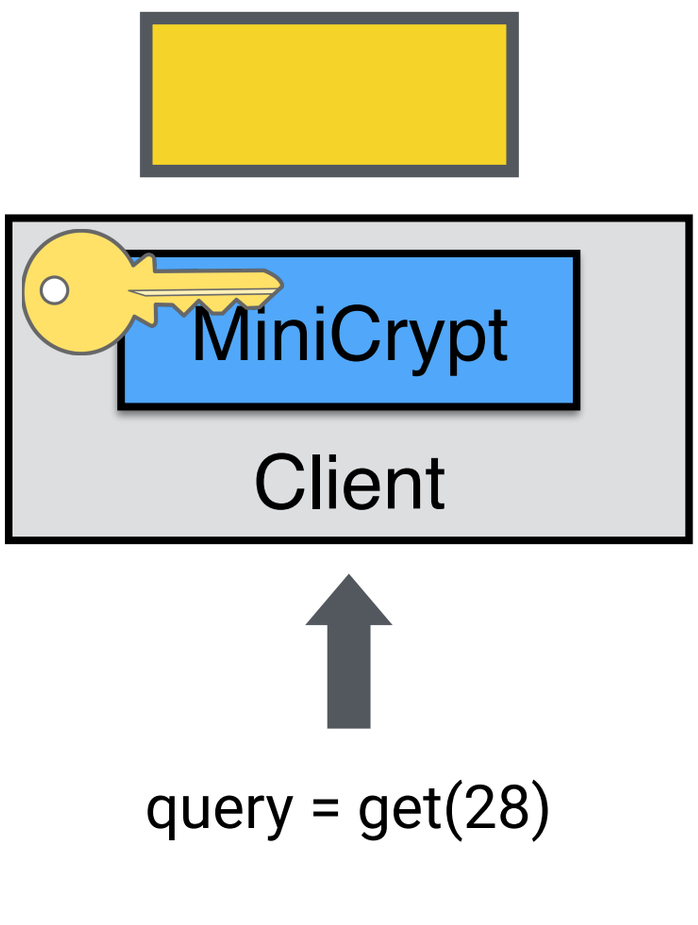
Single-key get

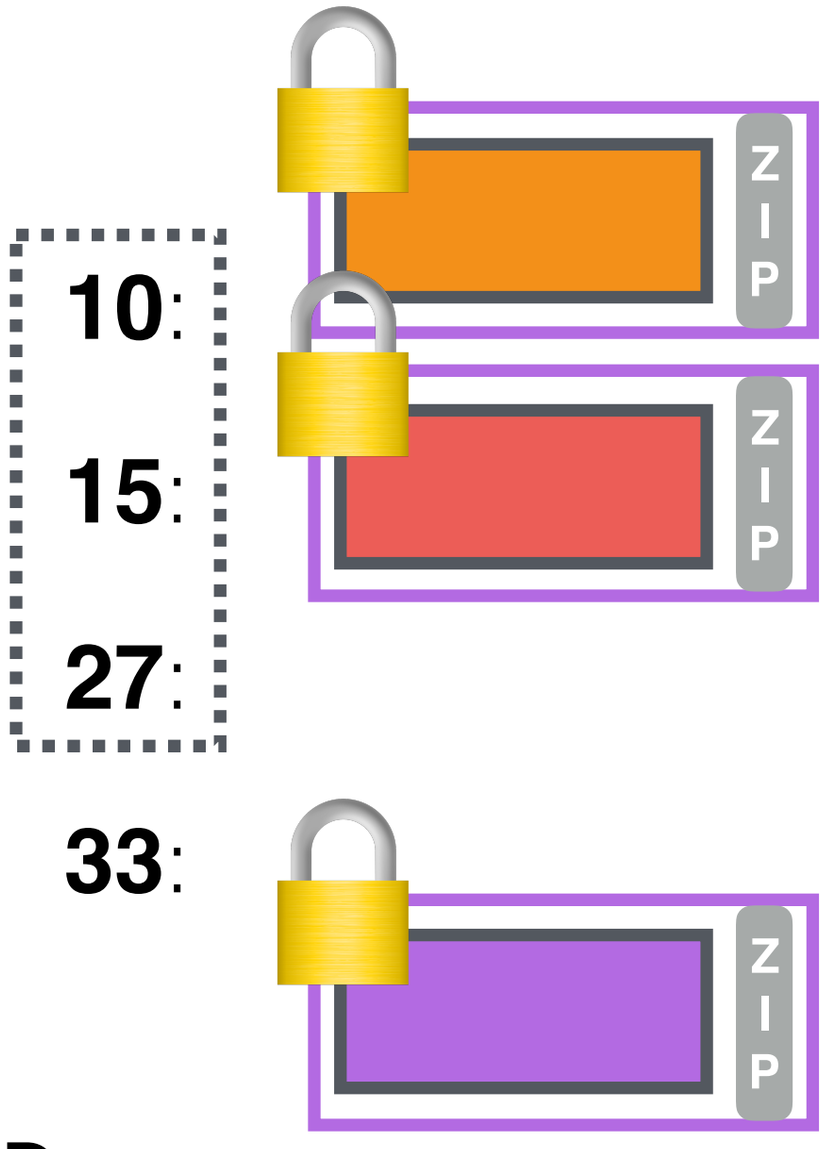




get(*key*): select pack with **highest pack ID** from all pack ID's that are at most *key*

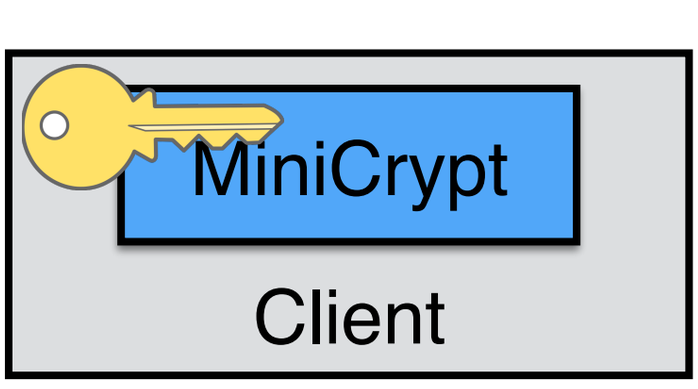
Single-key get





get(*key*): select pack with **highest pack ID** from all pack ID's that are at most *key*

Single-key put



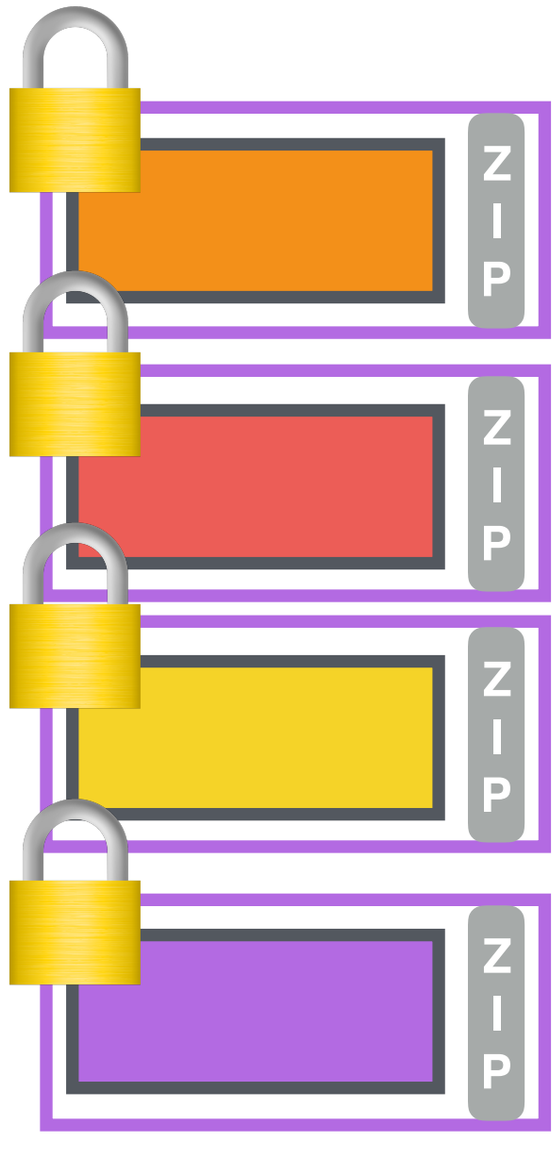
put(*key, value*):
find pack using get()

10:

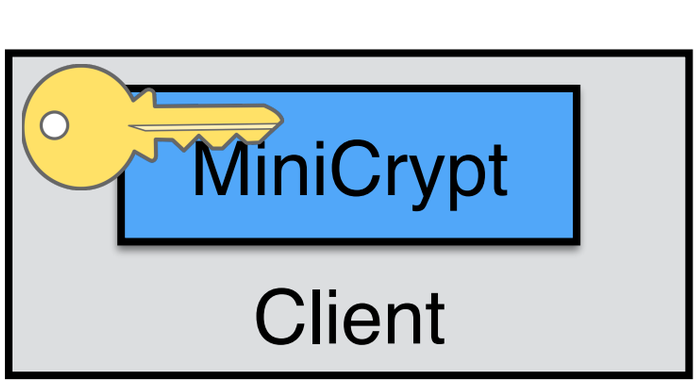
15:

27:

33:

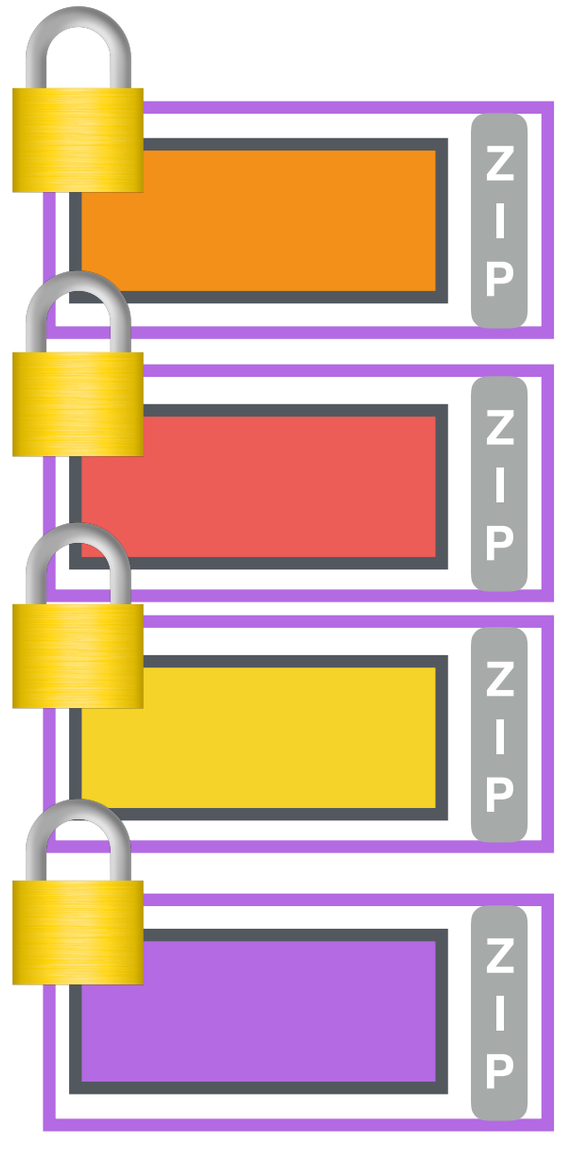


Single-key put

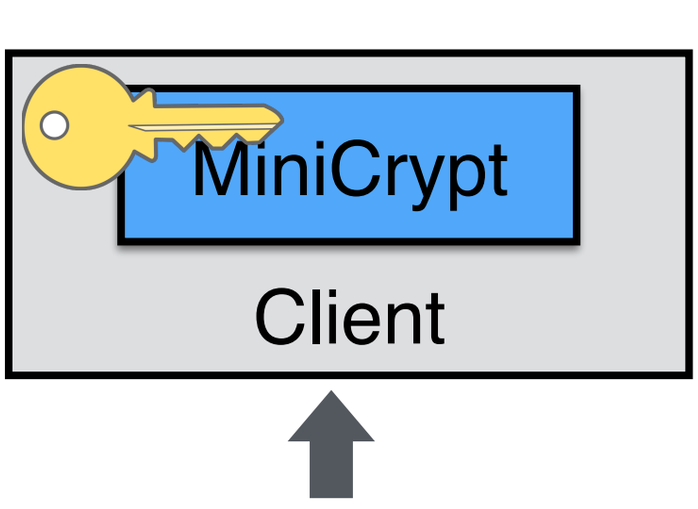


query = put(28, 5)

put(*key, value*):
find pack using get()

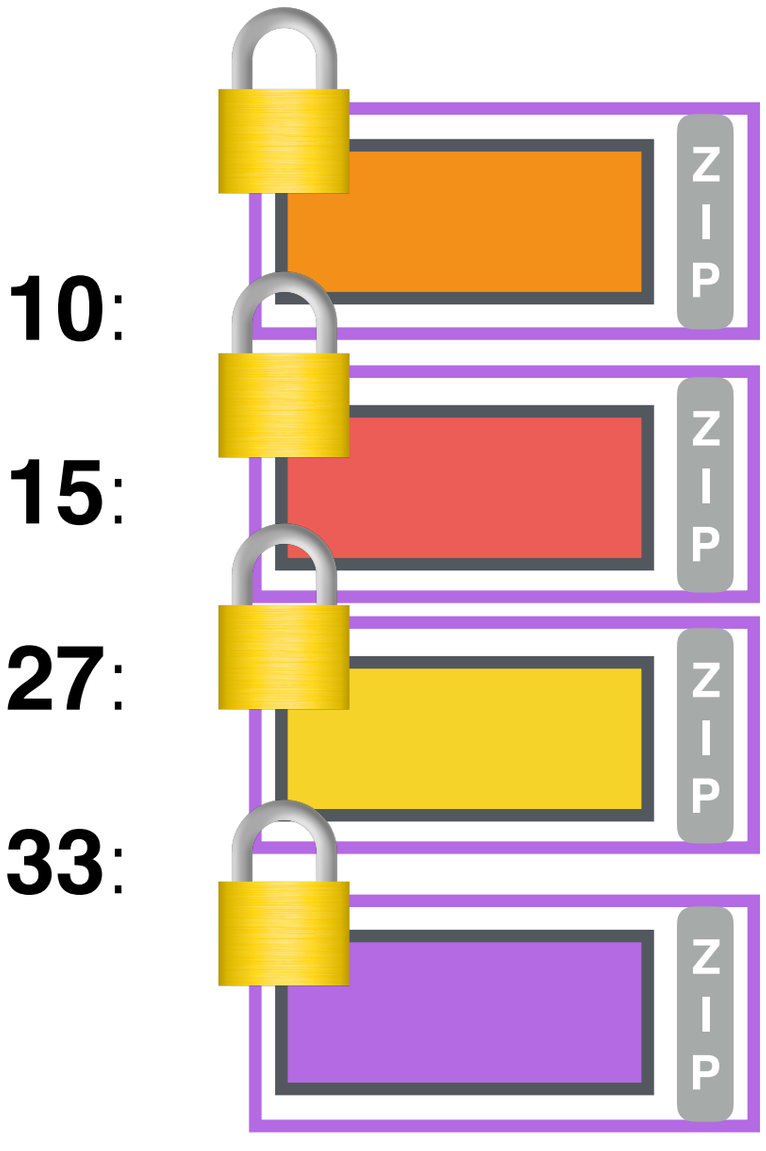


Single-key put

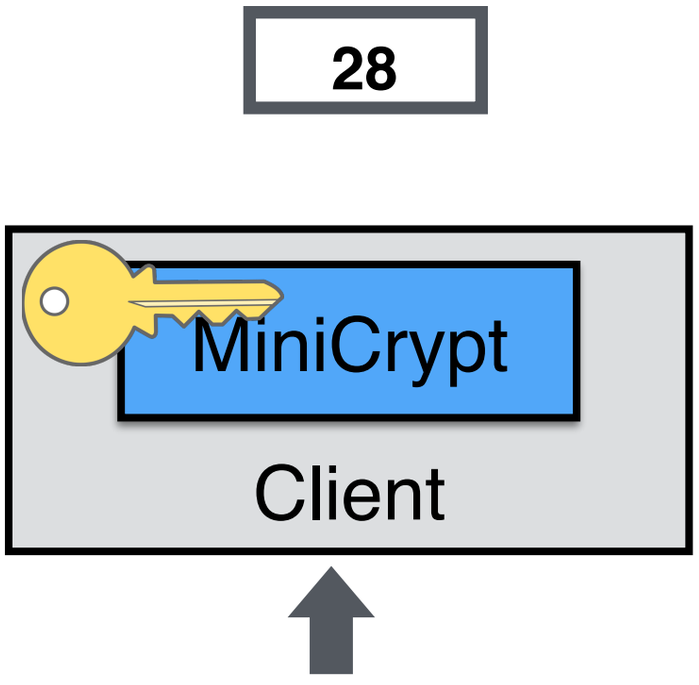


query = put(28, 5)

put(*key, value*):
find pack using get()

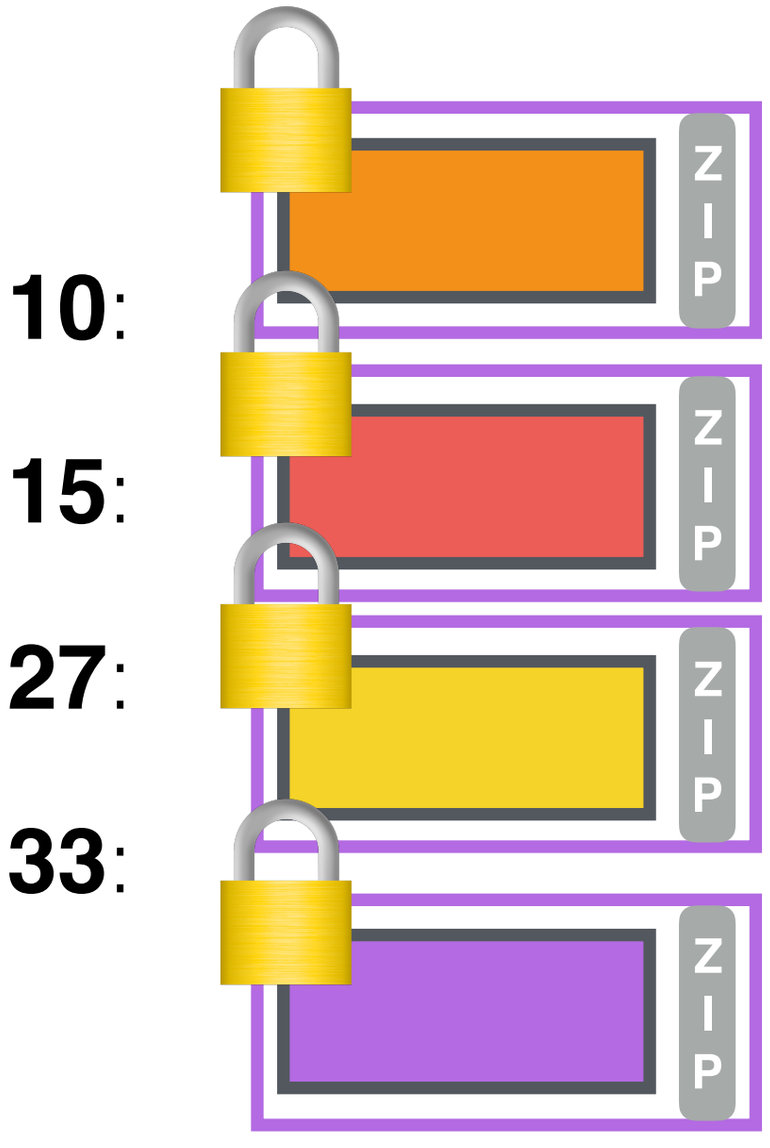


Single-key put

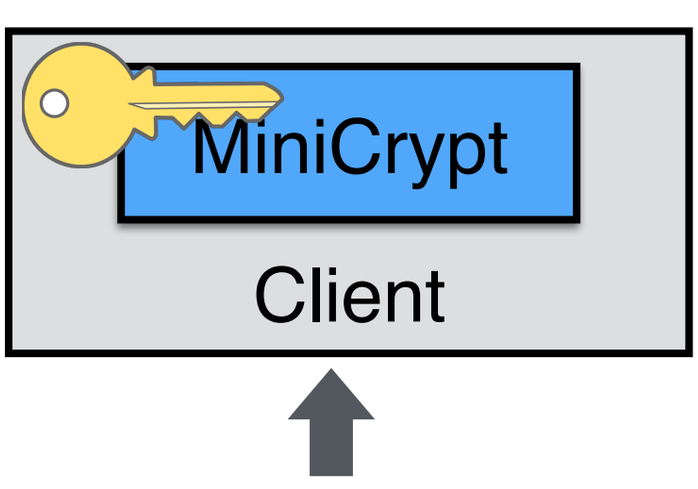


query = put(28, 5)

put(*key, value*):
find pack using get()



Single-key put



query = put(28, 5)

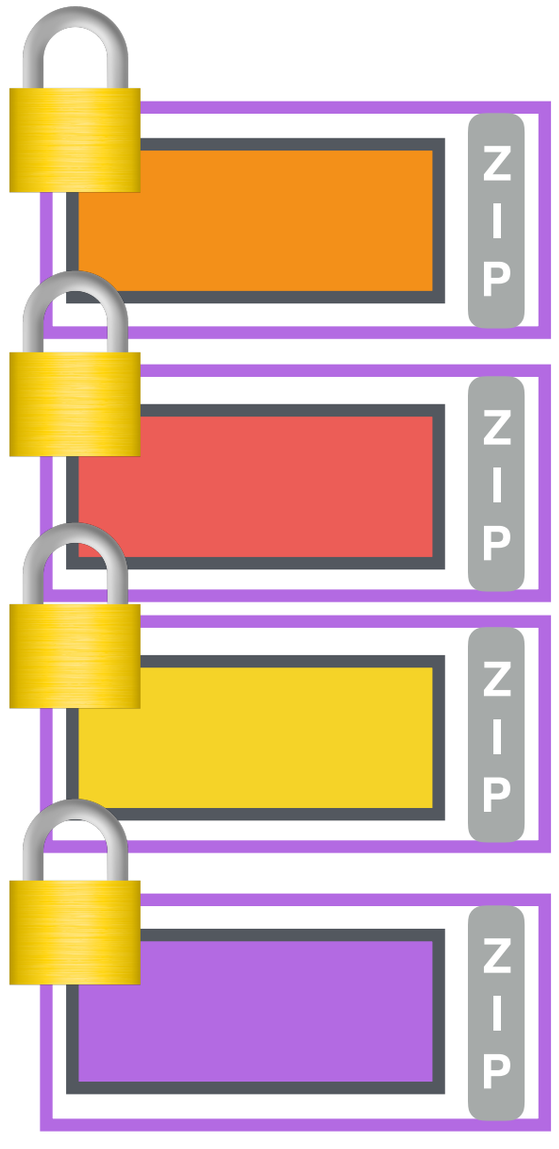
put(*key, value*):
find pack using get()

10:

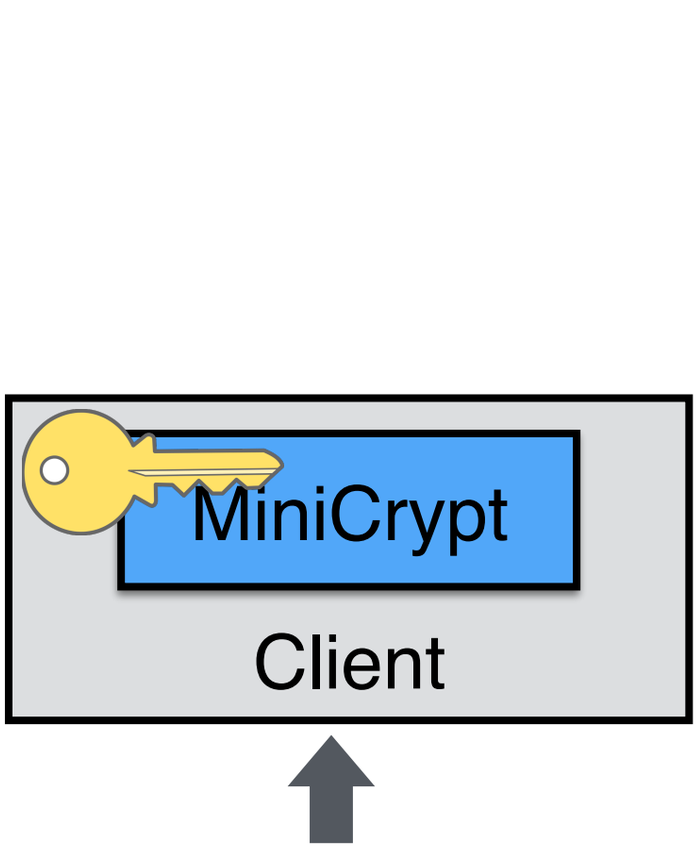
15:

27:

33:

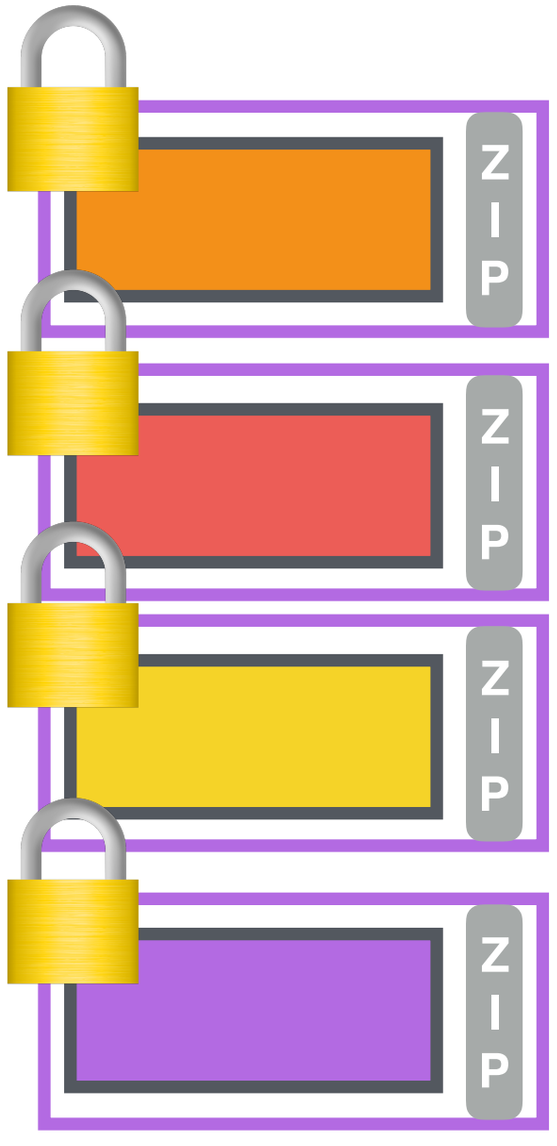


Single-key put

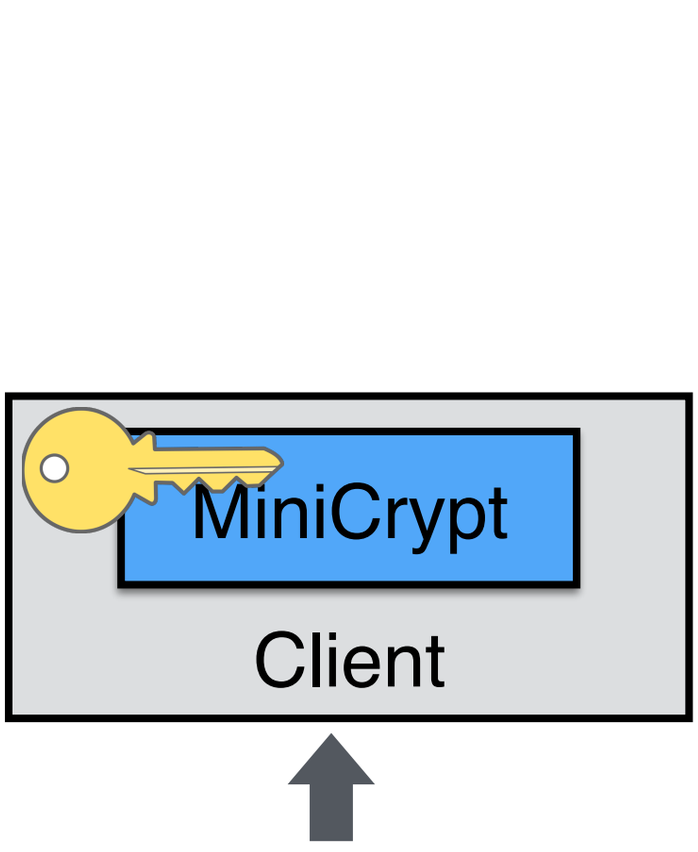


query = put(28, 5)

put(*key, value*):
find pack using get()

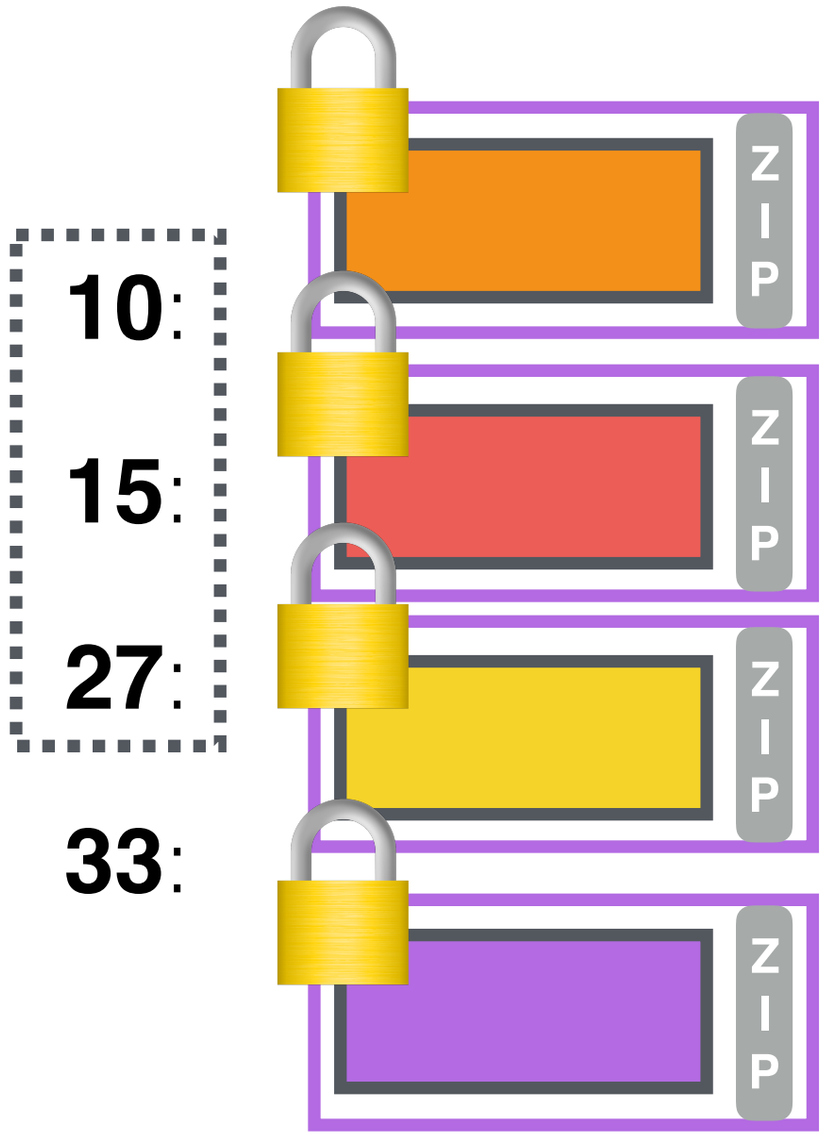


Single-key put

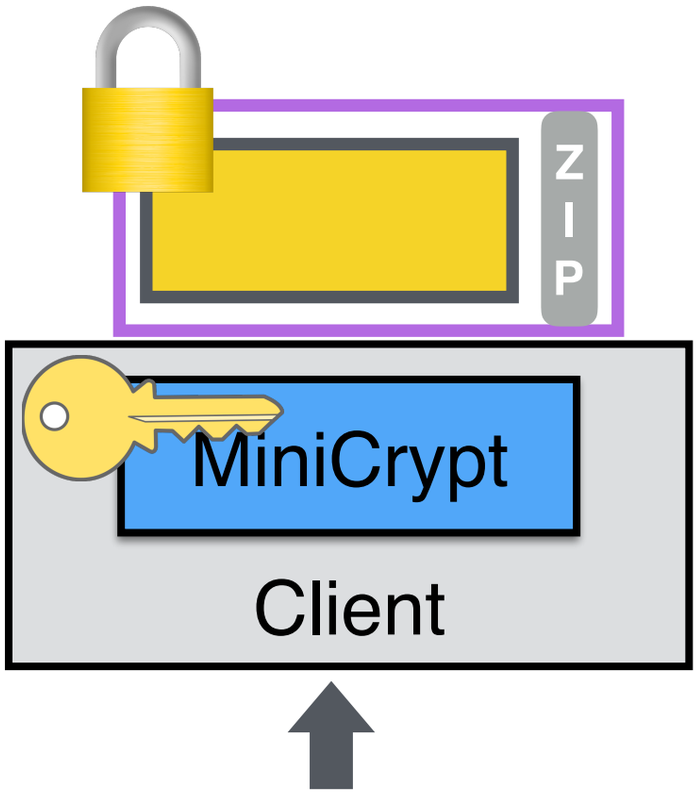


query = put(28, 5)

put(*key, value*):
find pack using get()

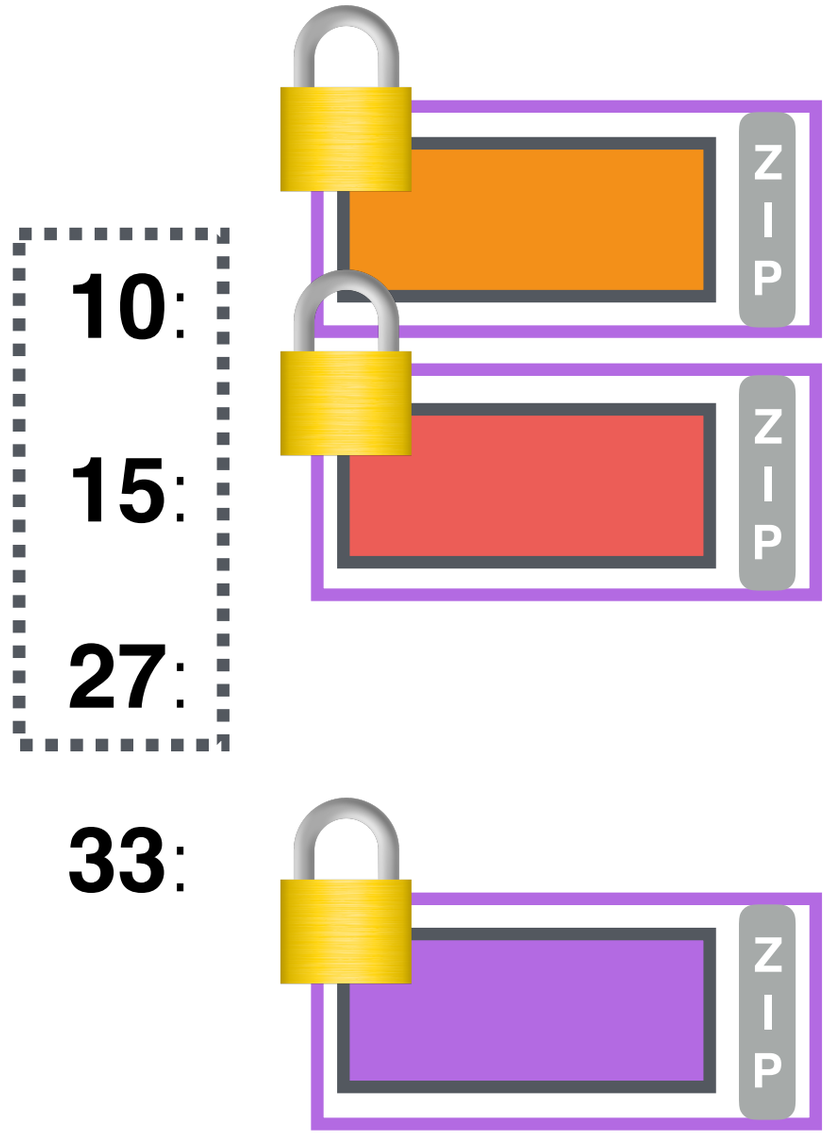


Single-key put

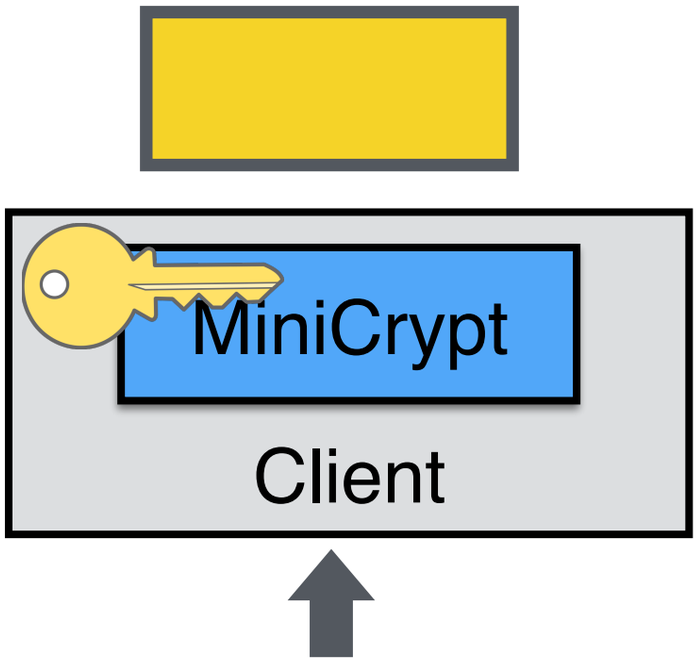


query = put(28, 5)

put(*key, value*):
find pack using get()

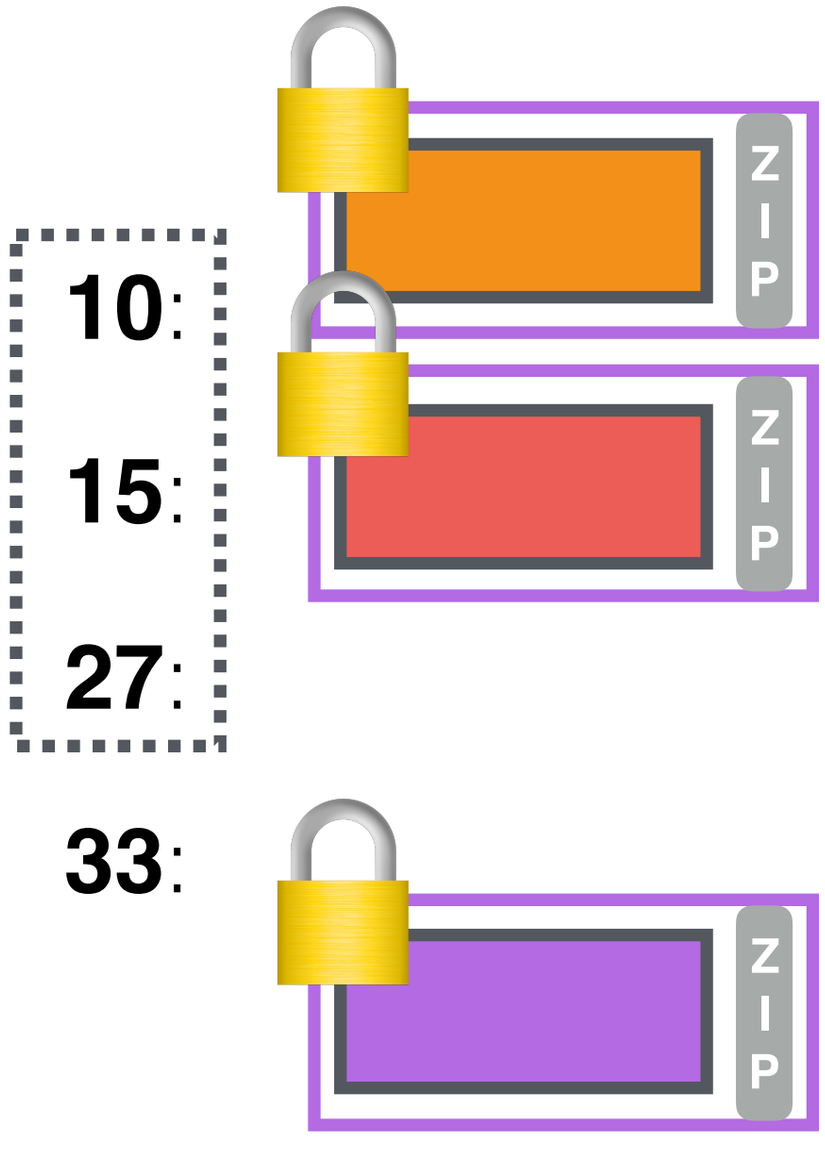


Single-key put

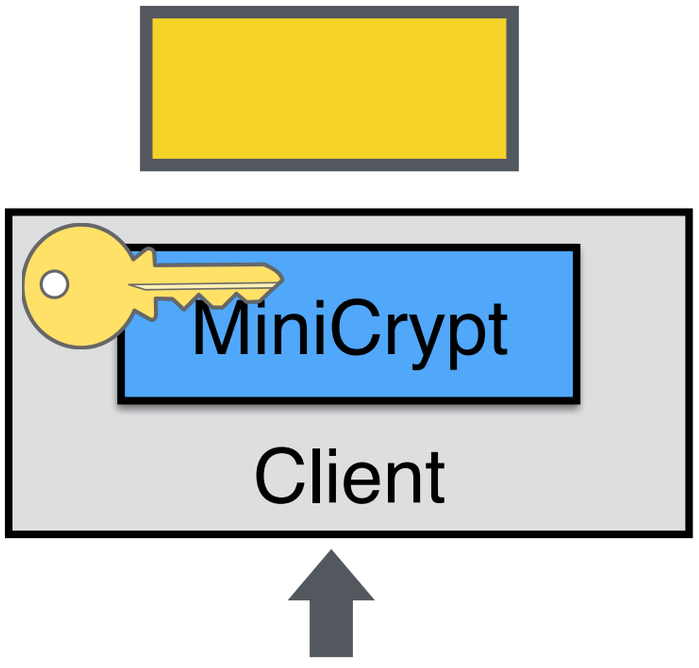


query = put(28, 5)

put(*key, value*):
find pack using get()

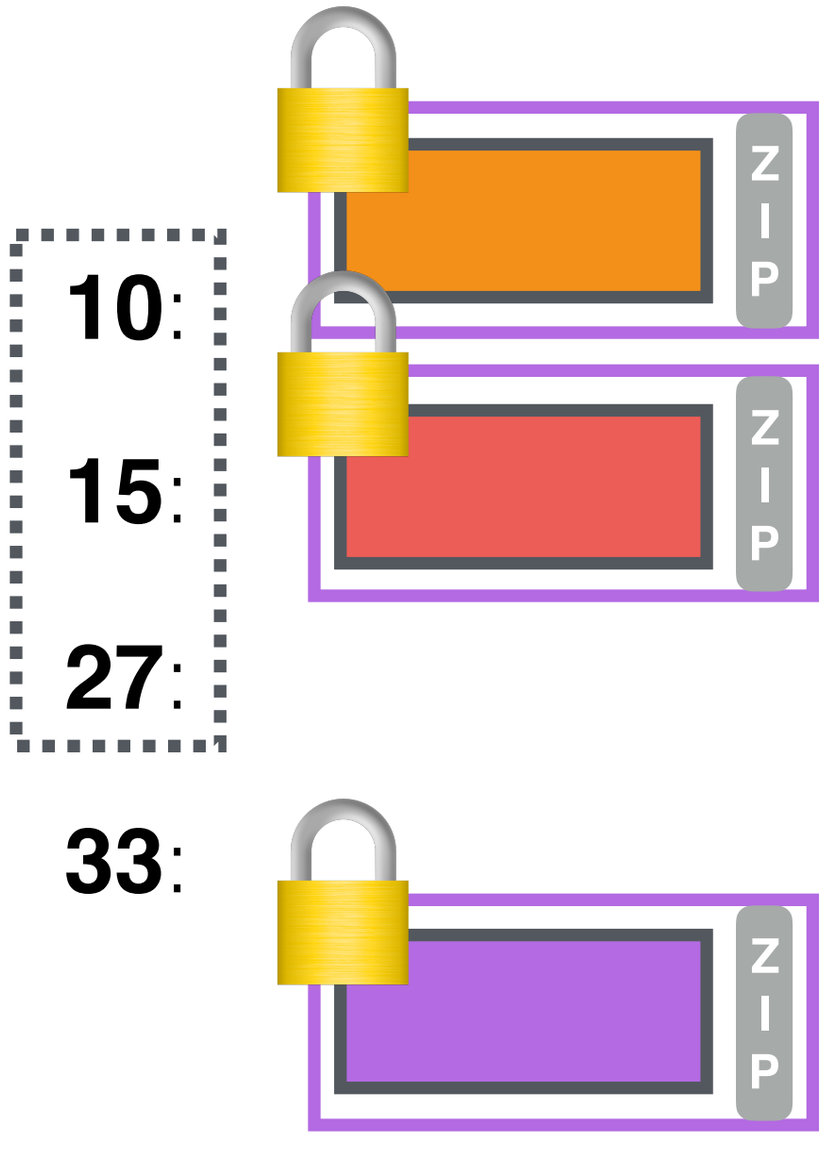


Single-key put

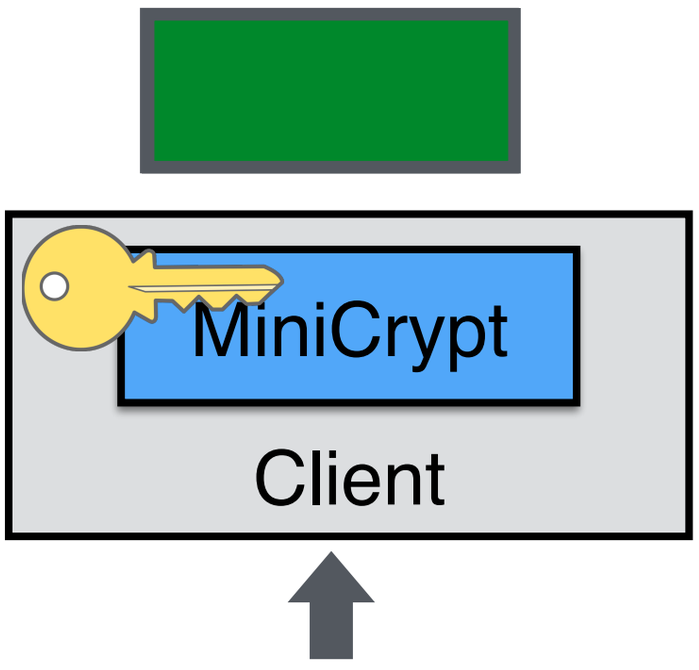


query = put(28, 5)

put(*key, value*):
find pack using get()
modify pack

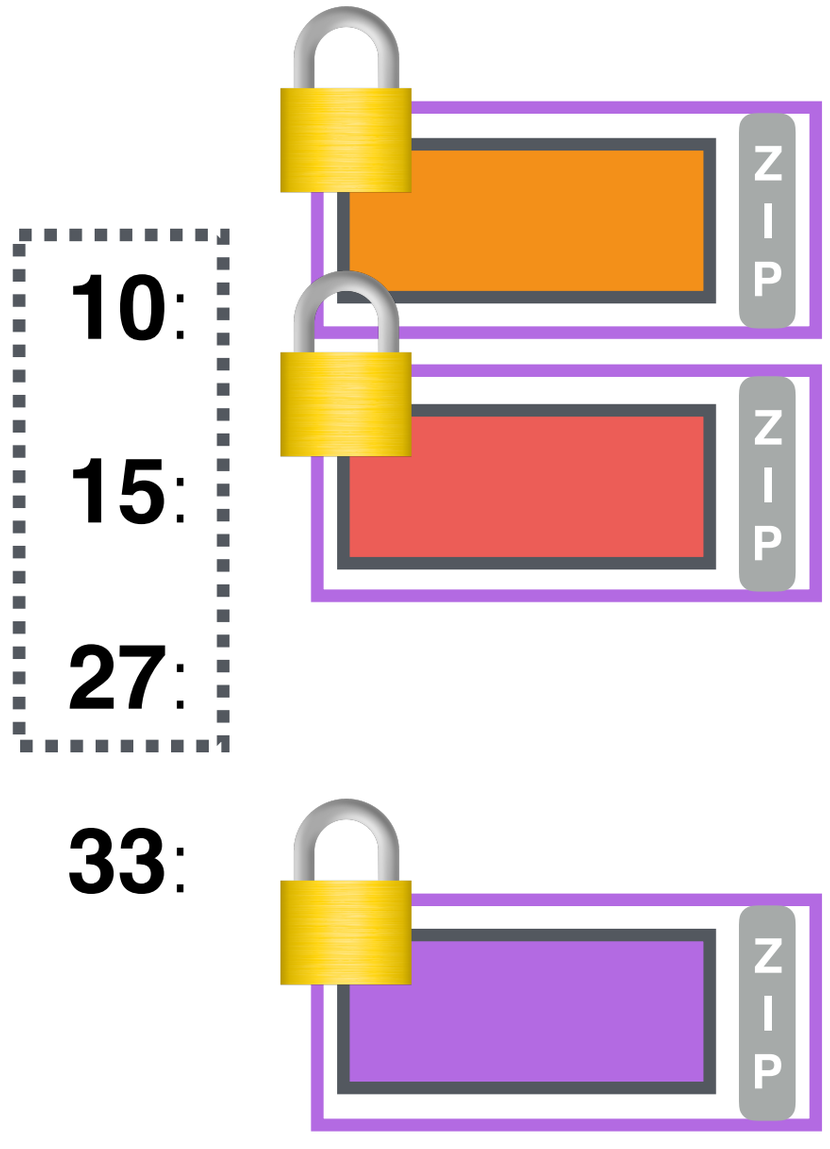


Single-key put

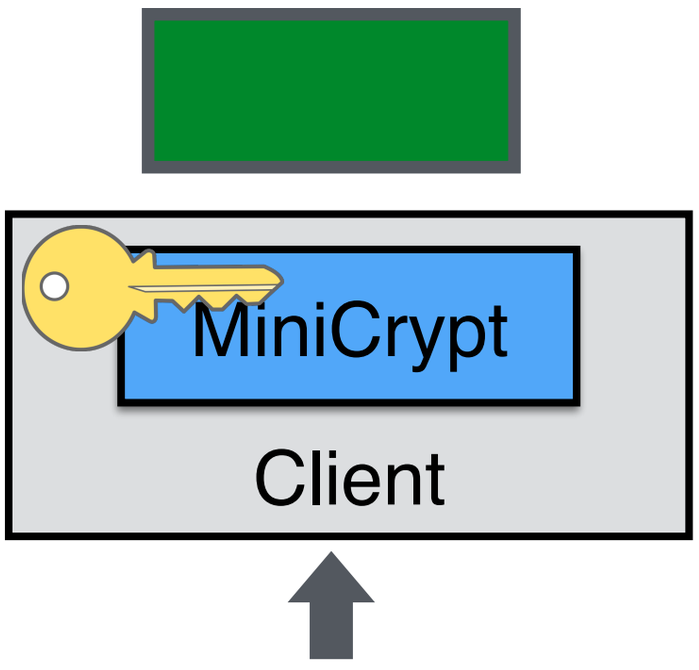


query = put(28, 5)

put(*key, value*):
find pack using get()
modify pack

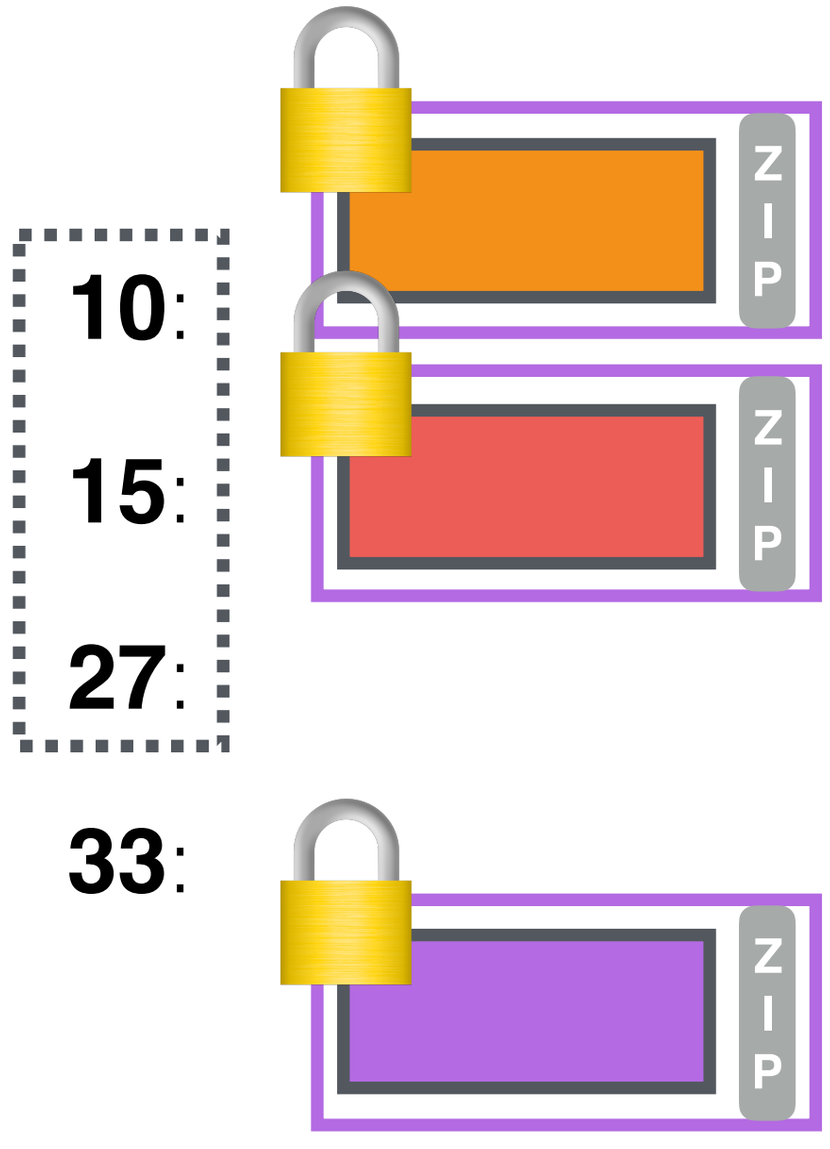


Single-key put

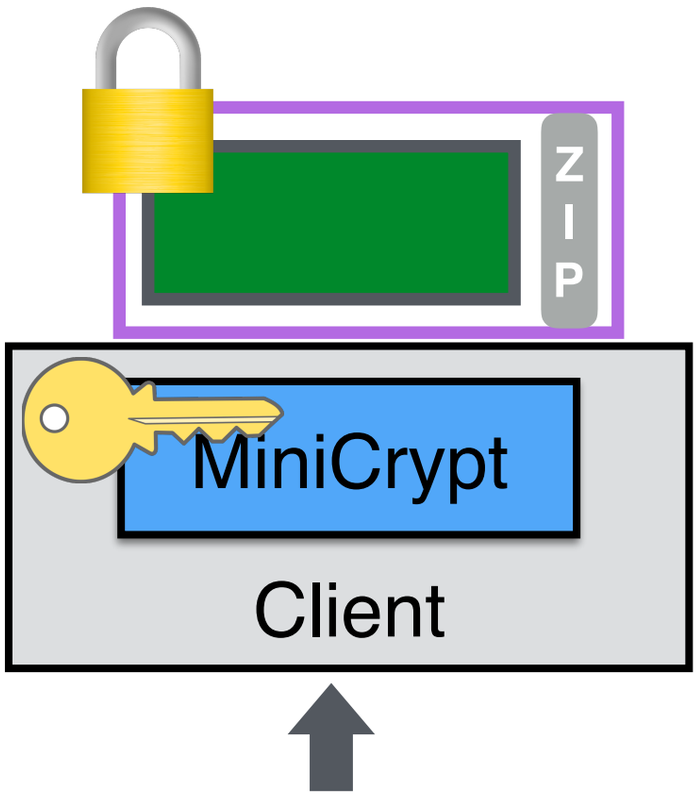


query = put(28, 5)

put(*key, value*):
find pack using get()
modify pack

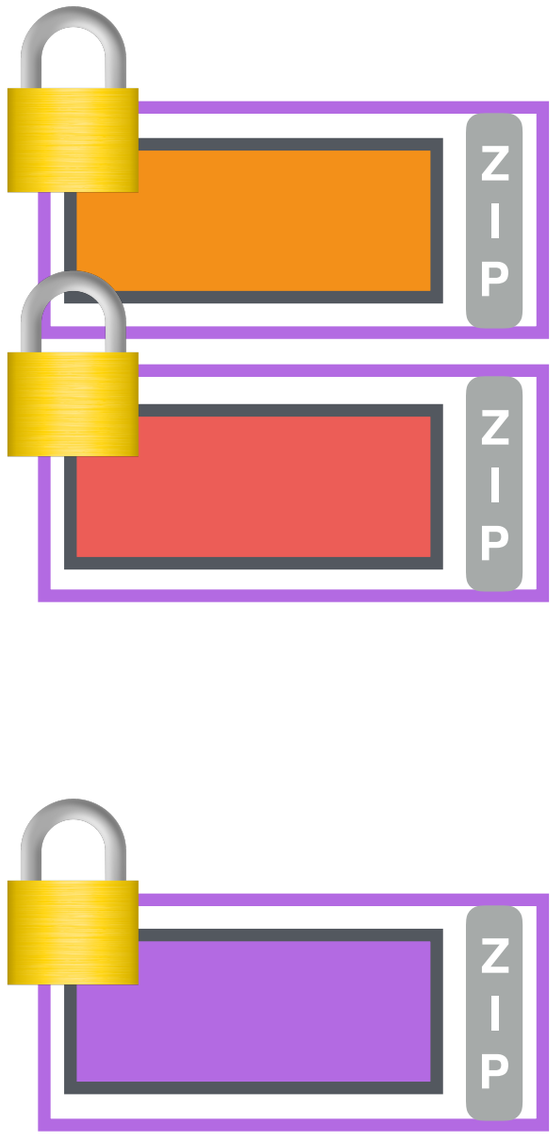


Single-key put

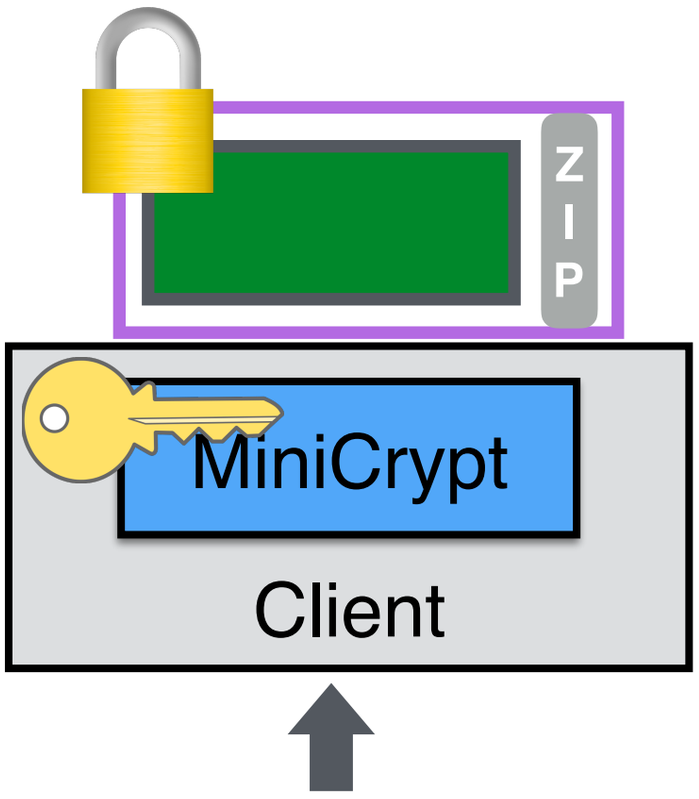


query = put(28, 5)

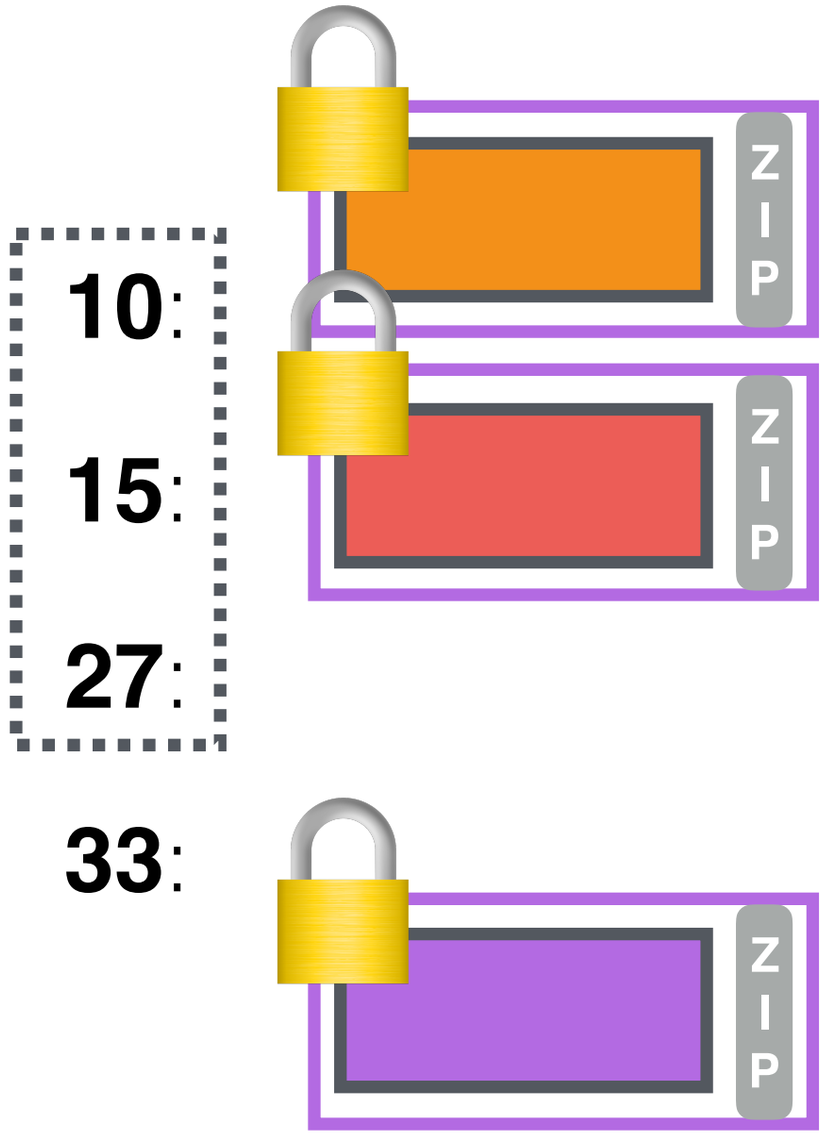
put(*key, value*):
find pack using get()
modify pack



Single-key put

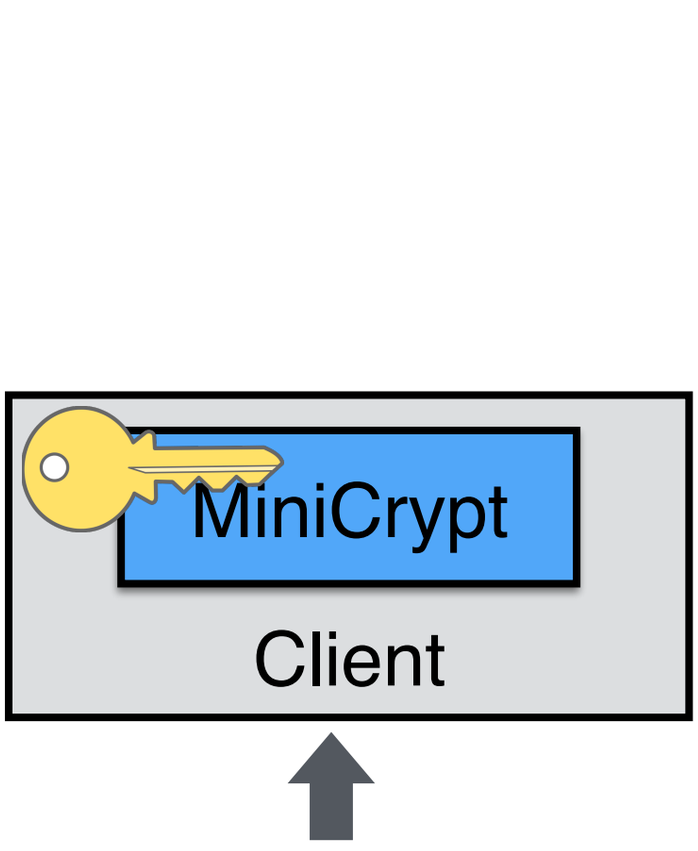


query = put(28, 5)

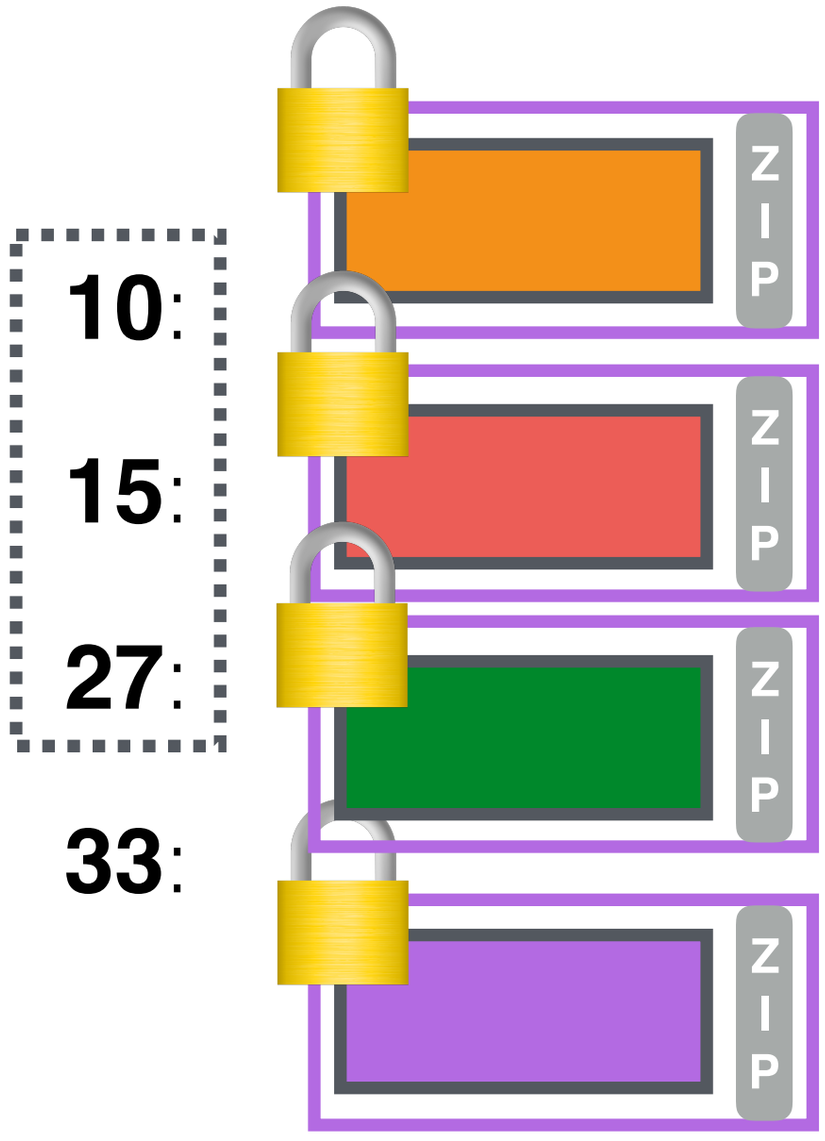


put(*key, value*):
find pack using get()
modify pack
write back using compare-and-swap

Single-key put



query = put(28, 5)



put(*key, value*):
find pack using get()
modify pack
write back using compare-and-swap

Problem: writes are expensive

Problem: writes are expensive

- Solution: append-only mode

Problem: writes are expensive

- Solution: append-only mode
  - Inserted keys are roughly monotonically increasing

Problem: writes are expensive

- Solution: append-only mode
  - Inserted keys are roughly monotonically increasing
  - Key-value pairs can be modified within some bounded time

Problem: writes are expensive

- Solution: append-only mode
  - Inserted keys are roughly monotonically increasing
  - Key-value pairs can be modified within some bounded time
  - Applications: streaming data, time-series data

Approach

Approach

- Server keeps track of *epoch*

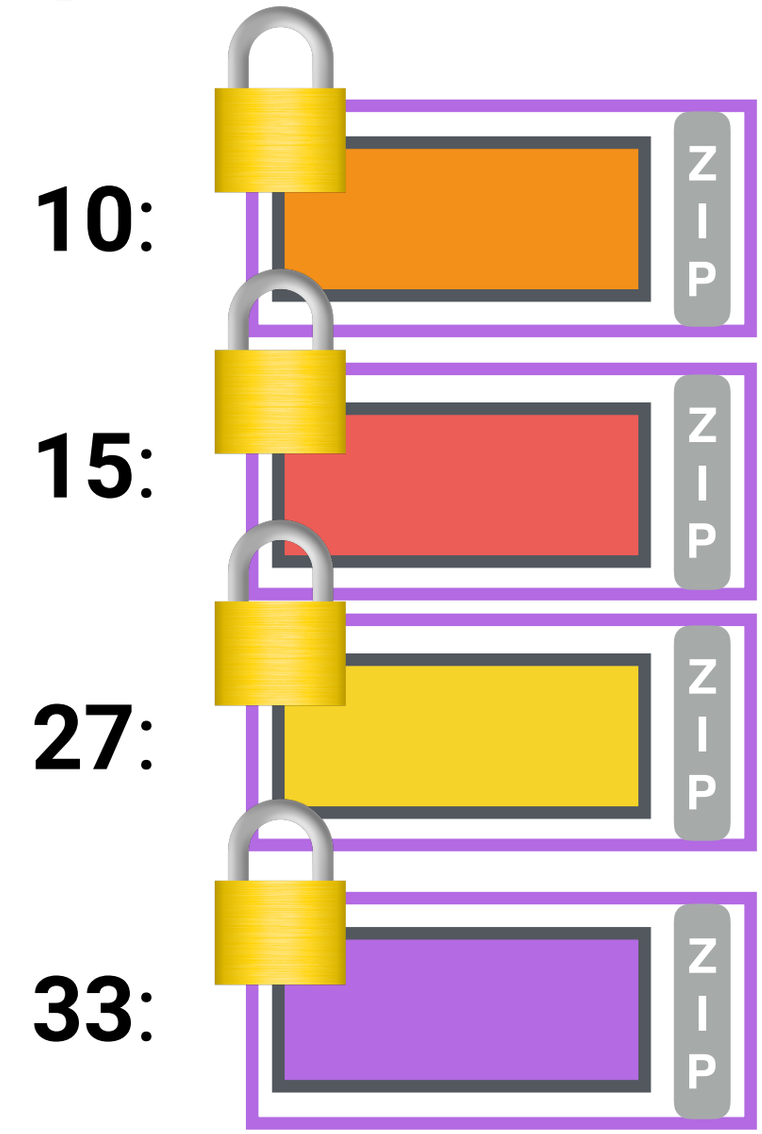
Approach

- Server keeps track of *epoch*
- Each insert assigned an epoch
  - Each new record inserted normally

Approach

- Server keeps track of *epoch*
- Each insert assigned an epoch
  - Each new record inserted normally
- Merge epochs into packs in the background
  - merge deterministically

Append mode



Append mode

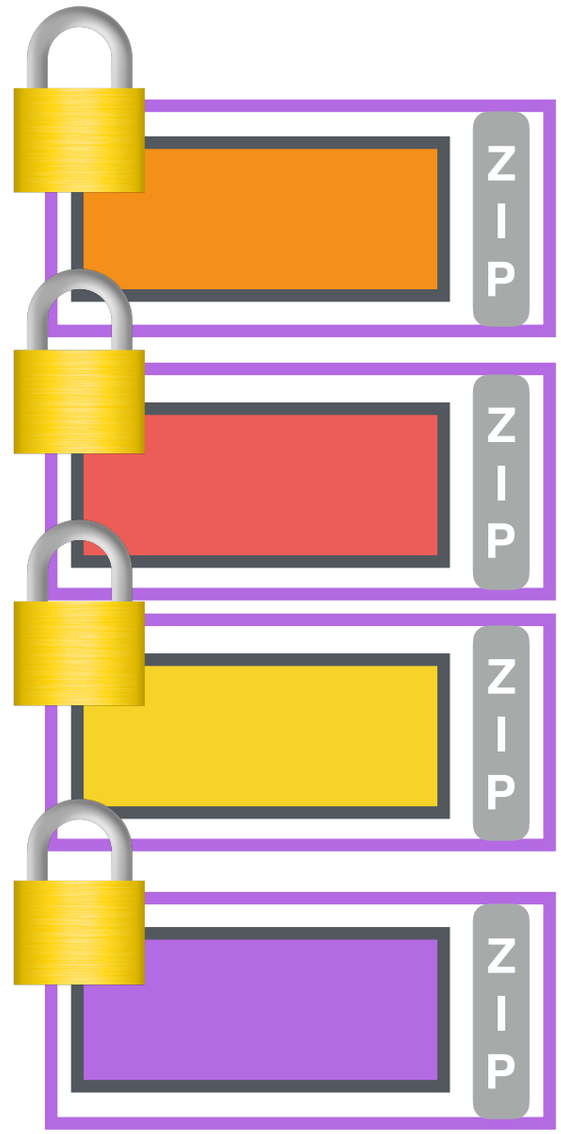
Current epoch: 4

10:

15:

27:

33:



Append mode

Current epoch: 4

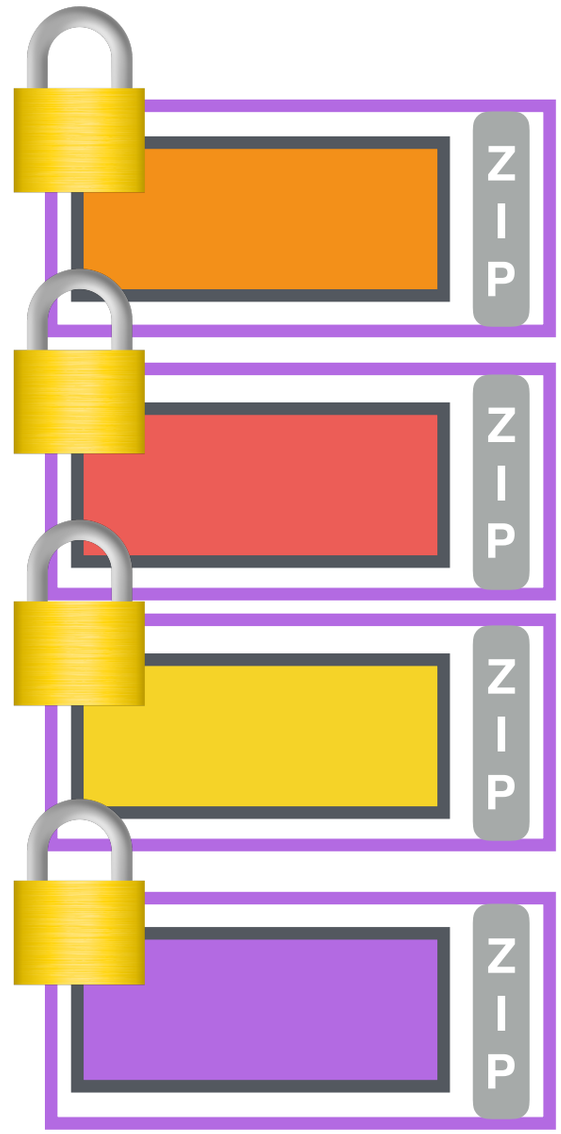
time

10:

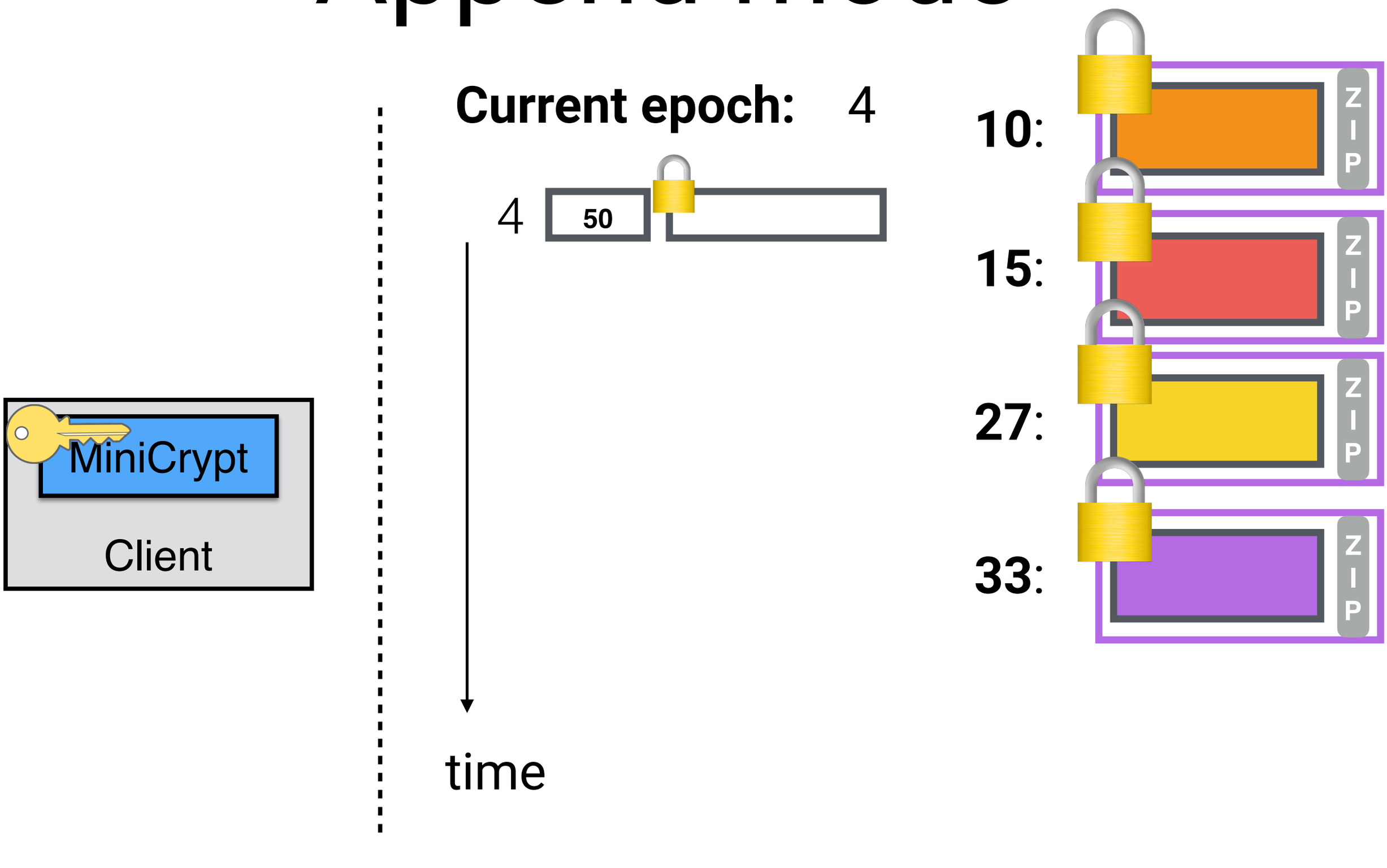
15:

27:

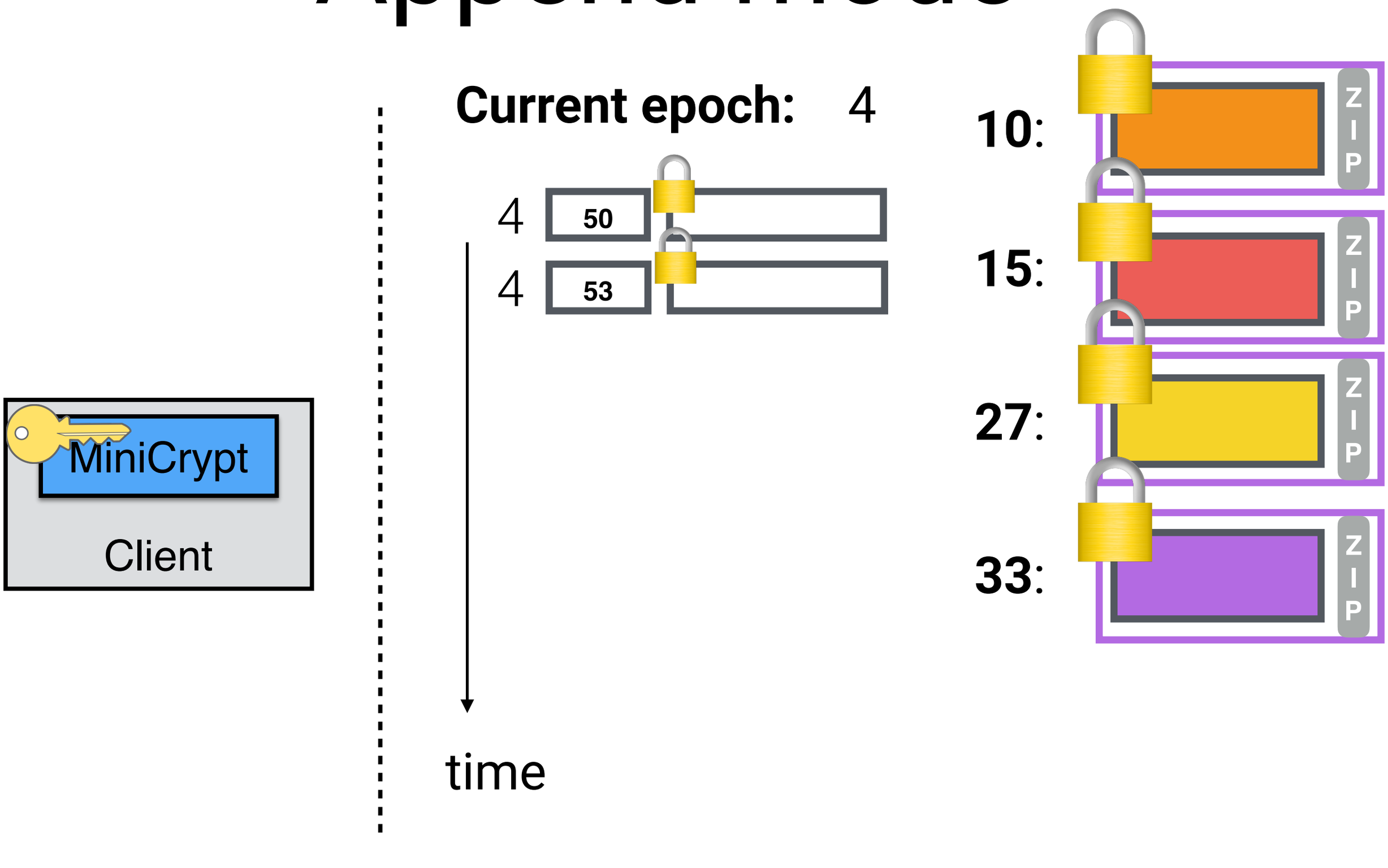
33:



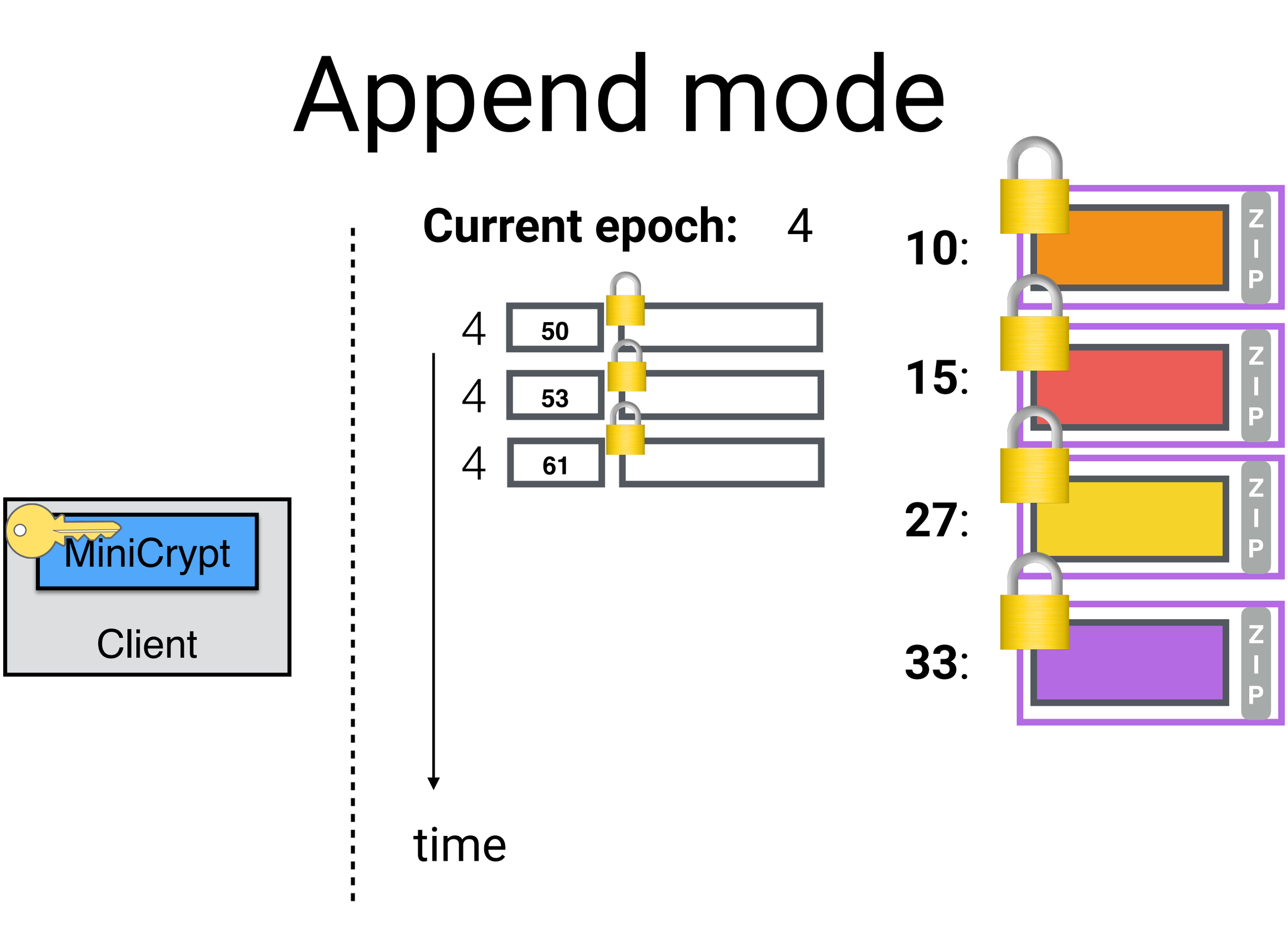
Append mode


Current epoch: 4
4
50
MiniCrypt
Client
time
10:
15:
27:
33:
ZIP
ZIP
ZIP
ZIP

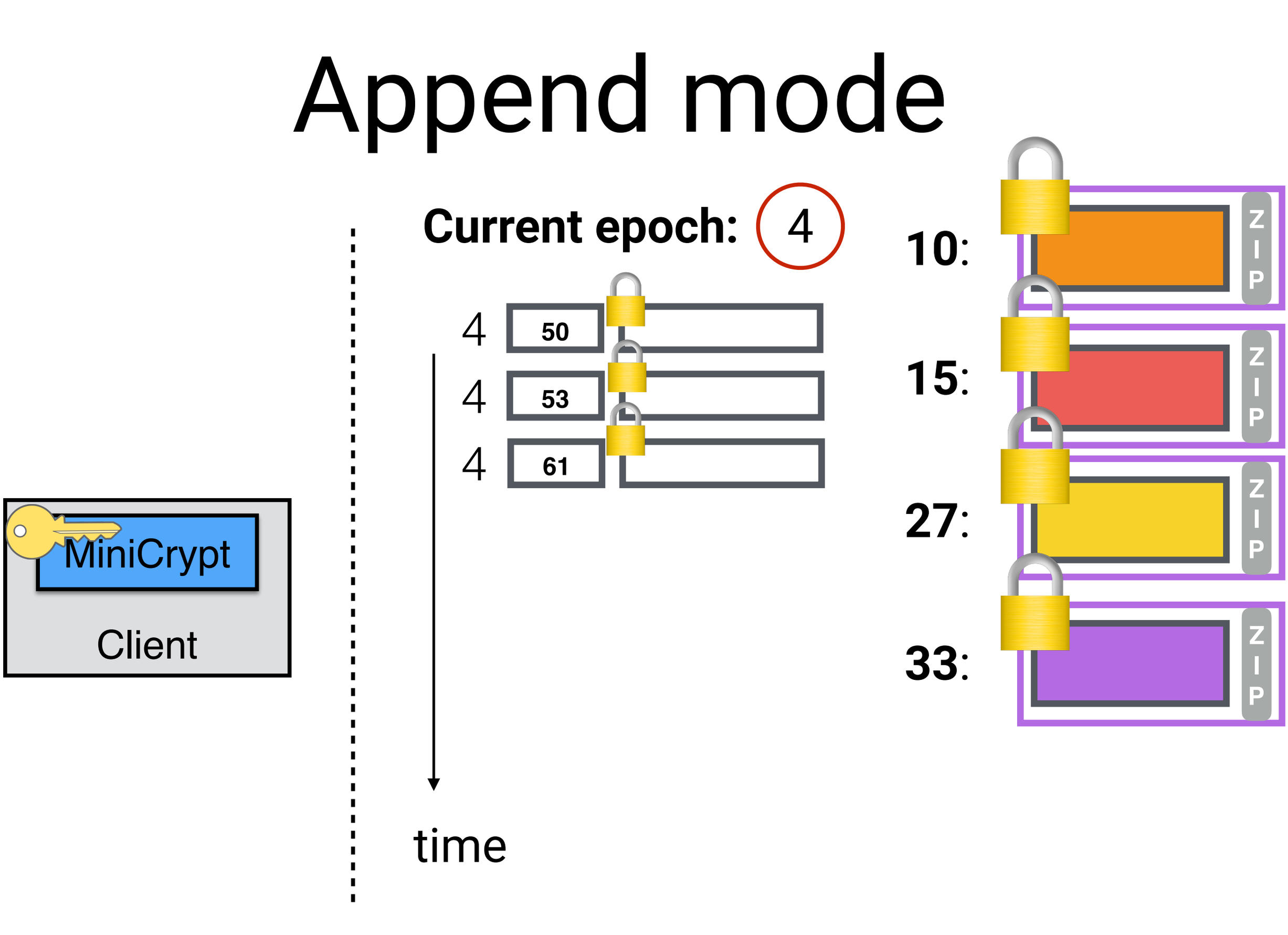
Append mode


Current epoch: 4
4
50
4
53
10:
15:
27:
33:
ZIP
ZIP
ZIP
ZIP
MiniCrypt
Client
time

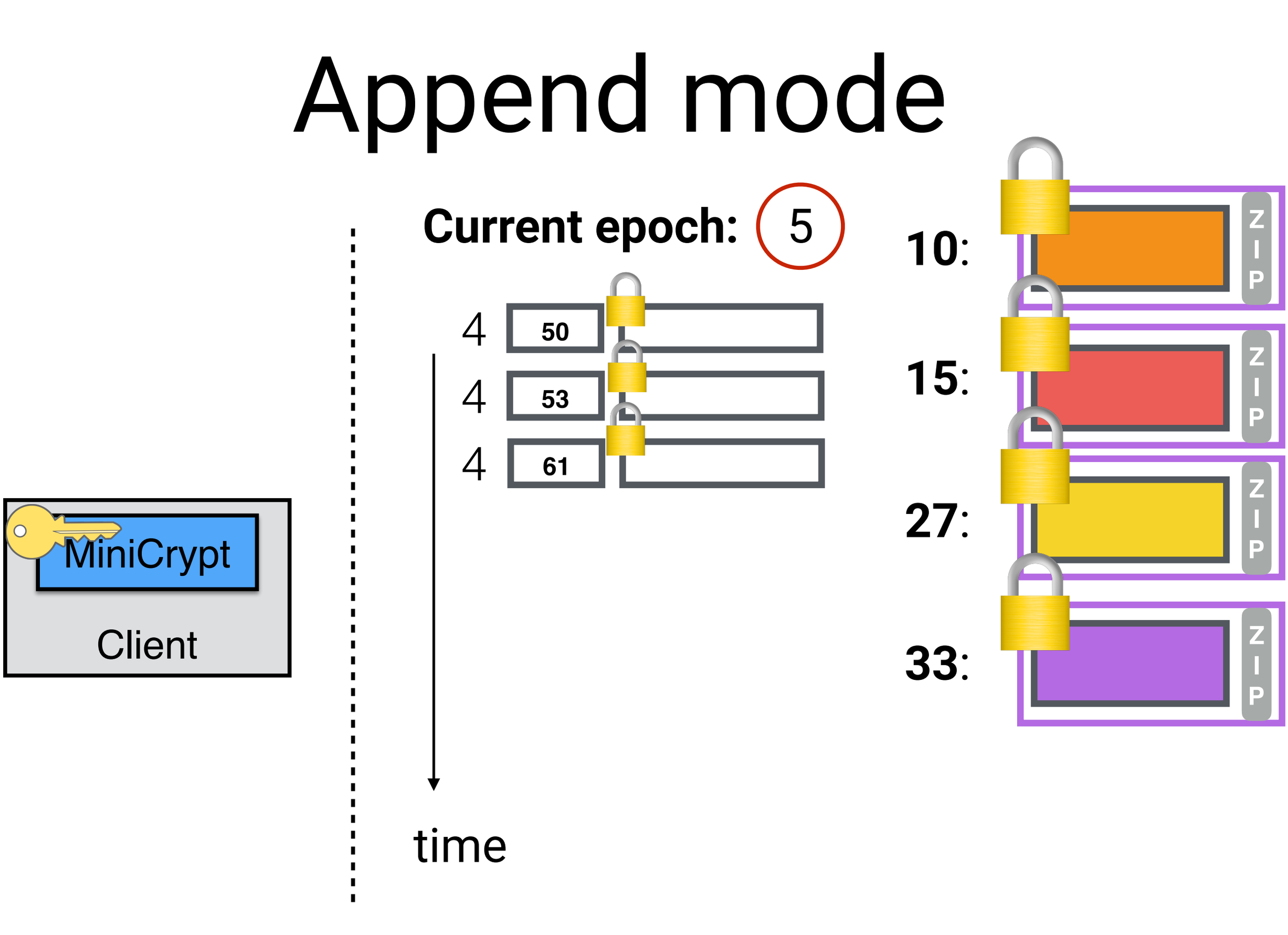
Append mode


Current epoch: 4
4
50
4
53
4
61
time
MiniCrypt
Client
10:
15:
27:
33:
ZIP
ZIP
ZIP
ZIP

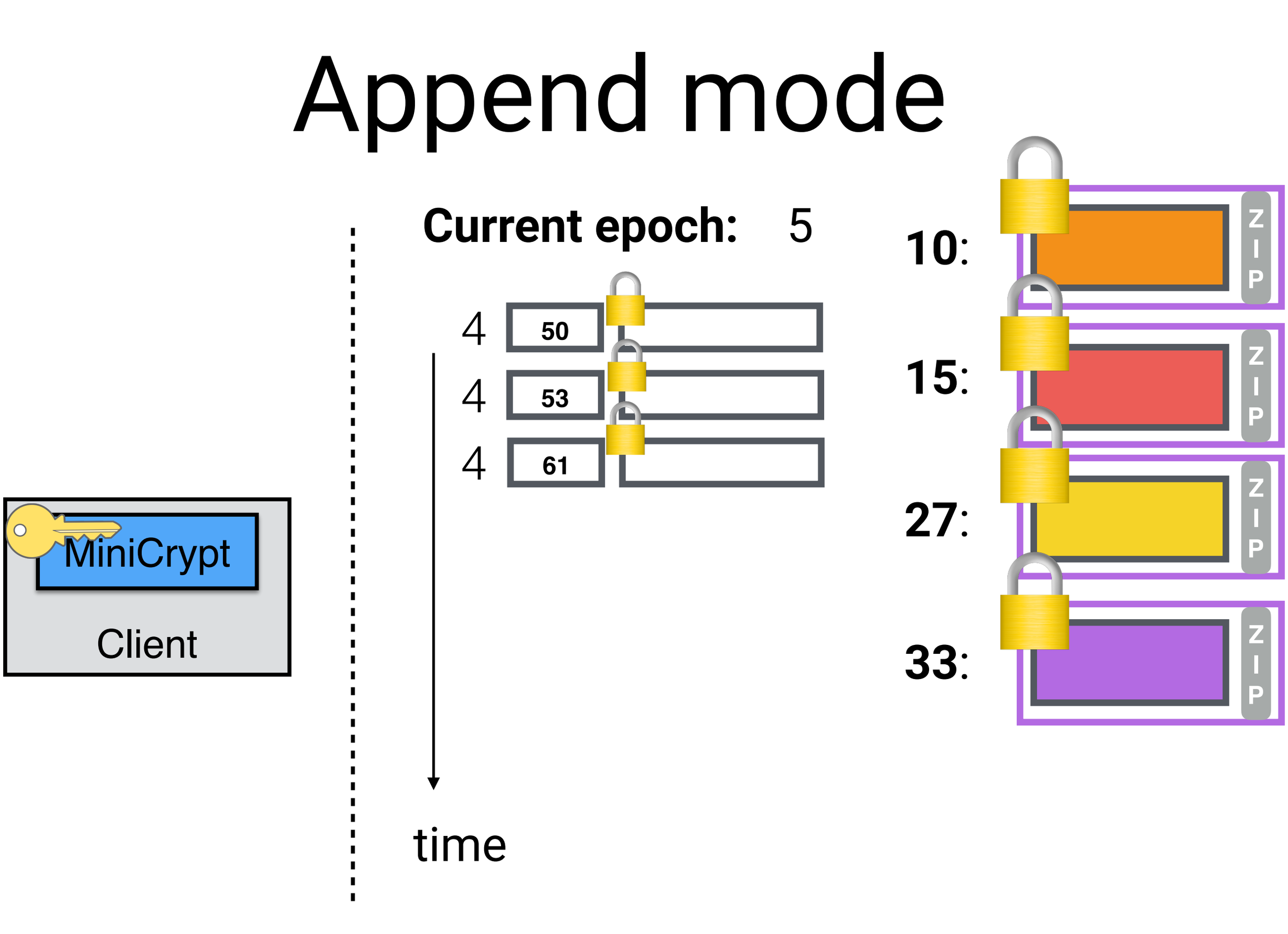
Append mode


Current epoch: 4
4 50
4 53
4 61
time
MiniCrypt
Client
10:
15:
27:
33:
ZIP
ZIP
ZIP
ZIP

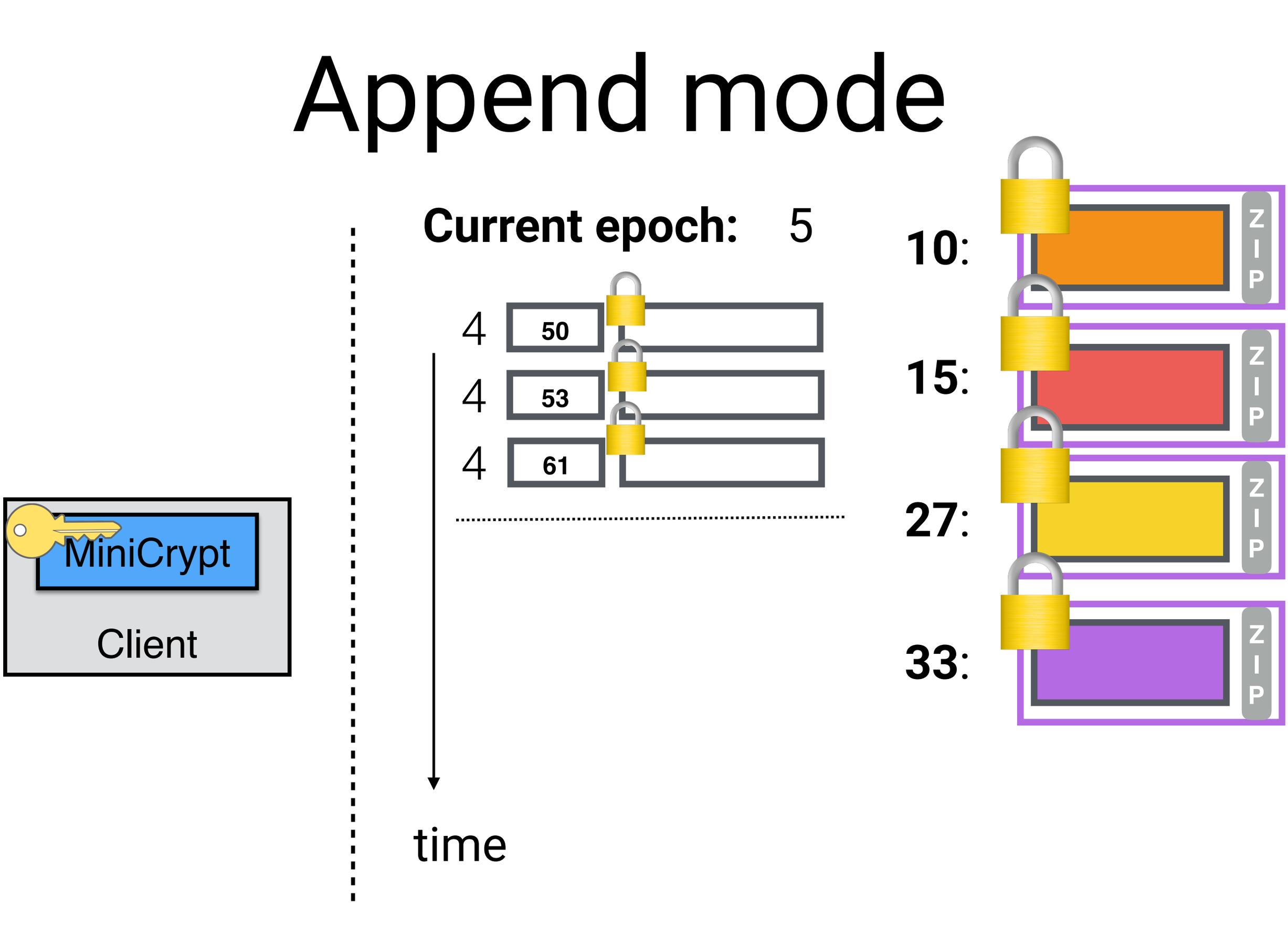
Append mode


Current epoch: 5
4
50
4
53
4
61
time
MiniCrypt
Client
10:
15:
27:
33:
ZIP
ZIP
ZIP
ZIP

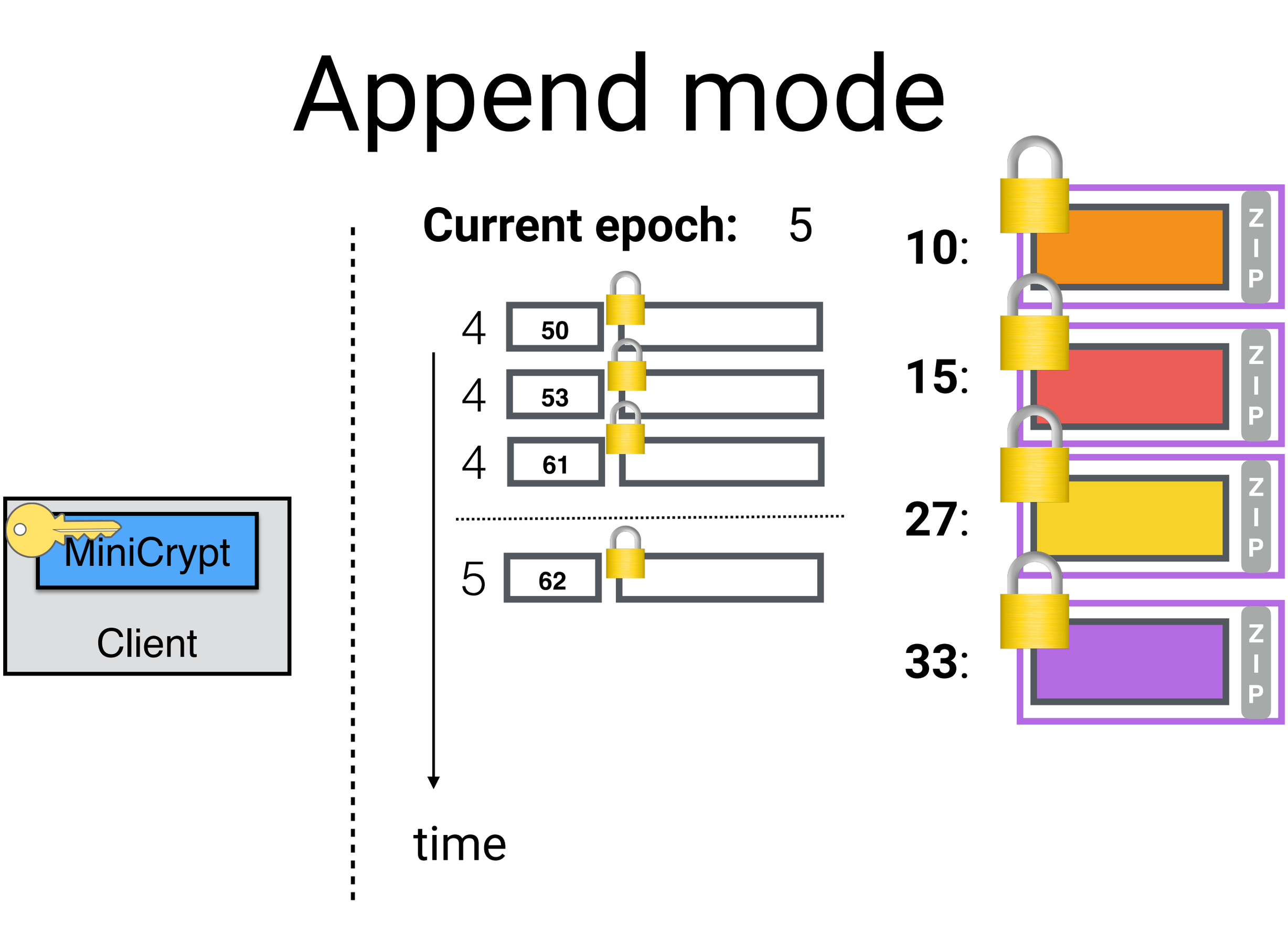
Append mode


Current epoch: 5
4
50
4
53
4
61
time
MiniCrypt
Client
10:
15:
27:
33:
ZIP
ZIP
ZIP
ZIP

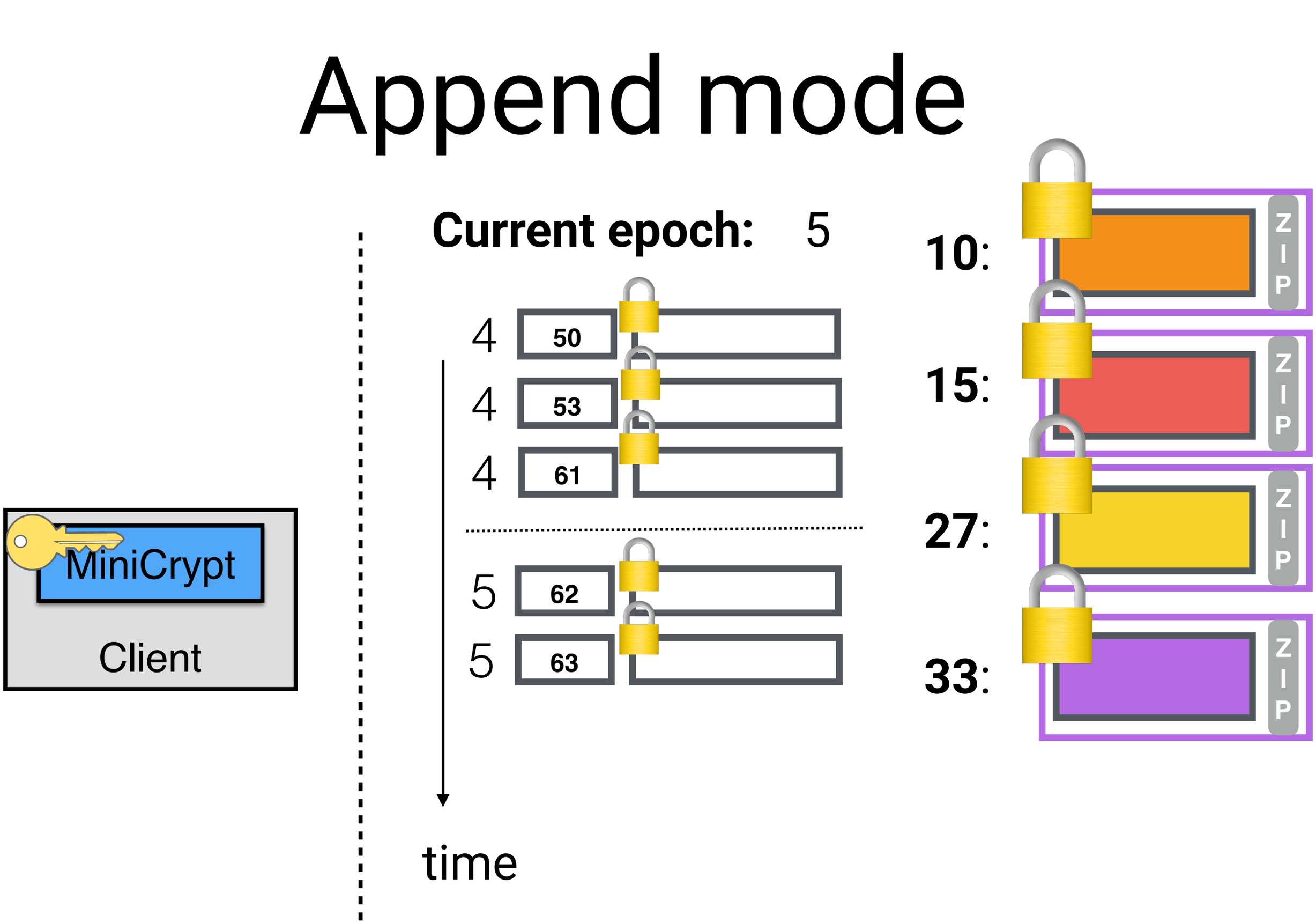
Append mode


Current epoch: 5
4
50
4
53
4
61
time
MiniCrypt
Client
10:
15:
27:
33:
ZIP
ZIP
ZIP
ZIP

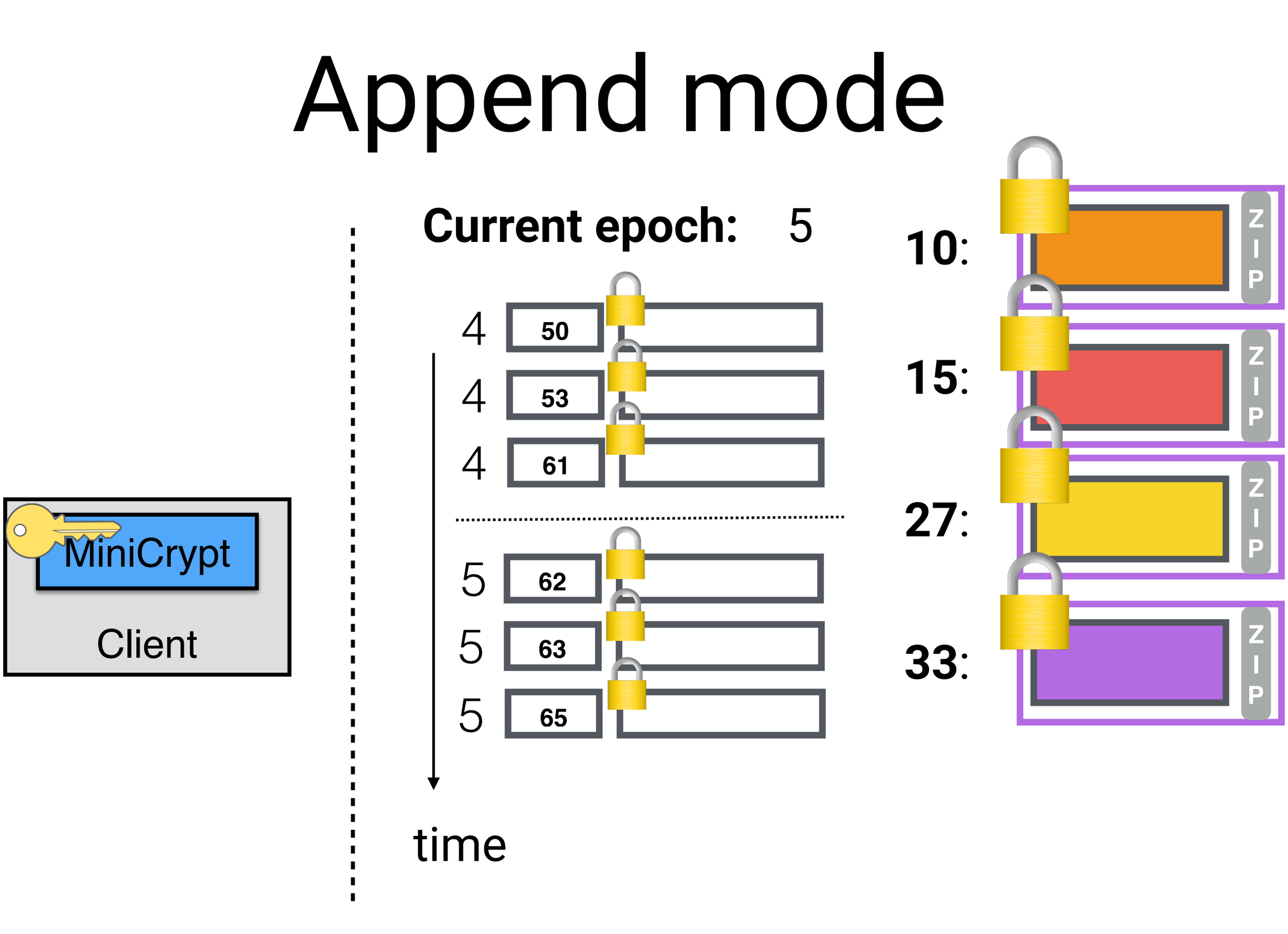
Append mode


Current epoch: 5
4
50
4
53
4
61
5
62
time
MiniCrypt
Client
10:
15:
27:
33:
ZIP
ZIP
ZIP
ZIP

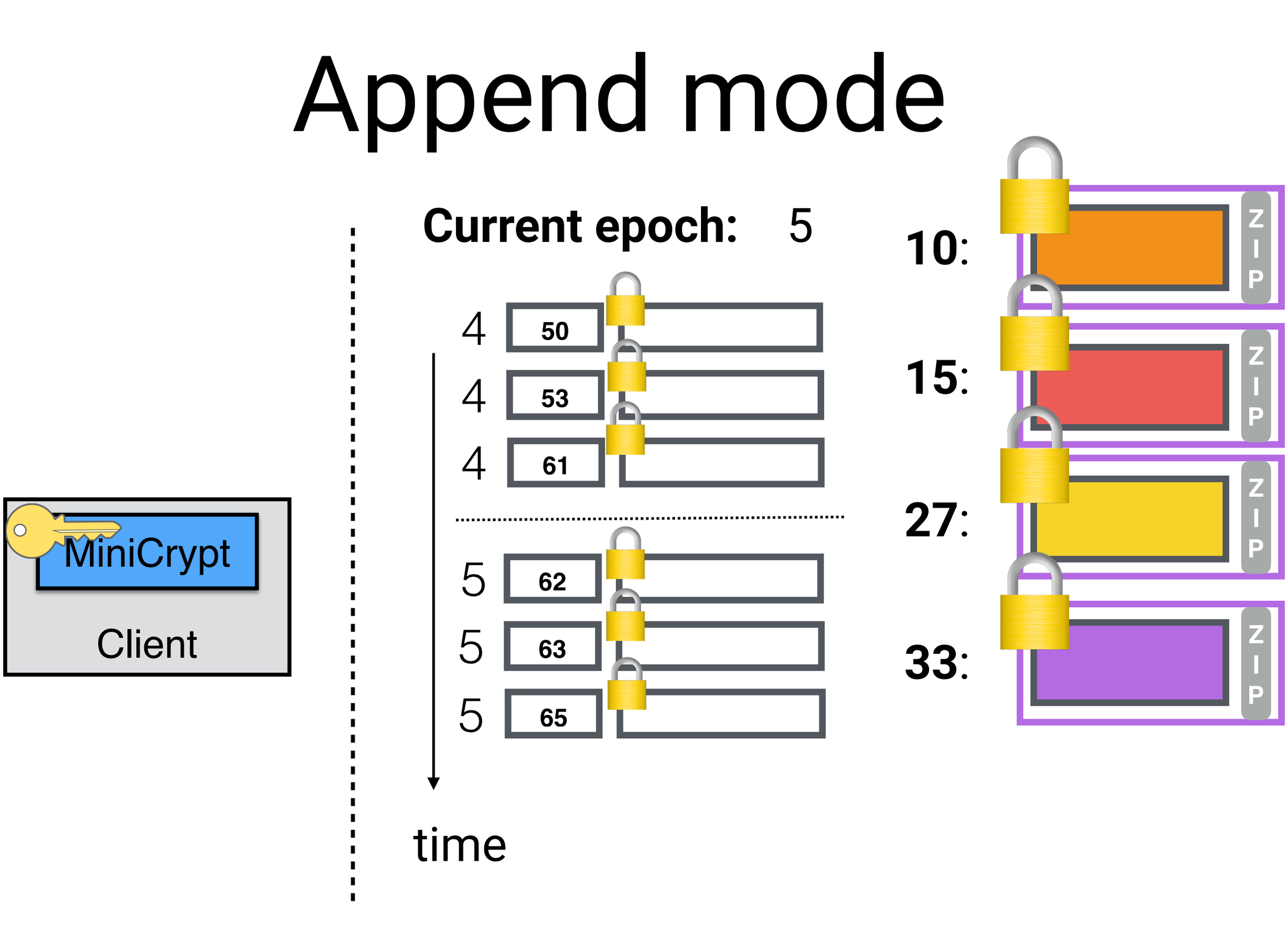
Append mode


MiniCrypt
Client
Current epoch: 5
4
50
4
53
4
61
5
62
5
63
time
10:
15:
27:
33:
ZIP
ZIP
ZIP
ZIP

Append mode

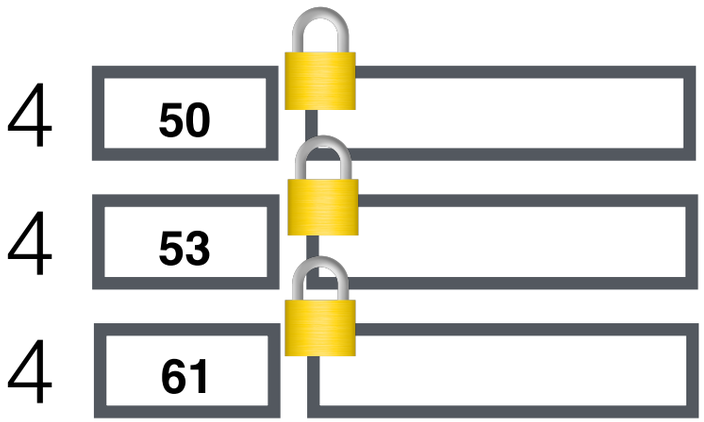

Current epoch: 5
4
50
4
53
4
61
5
62
5
63
5
65
time
MiniCrypt
Client
10:
15:
27:
33:
ZIP
ZIP
ZIP
ZIP

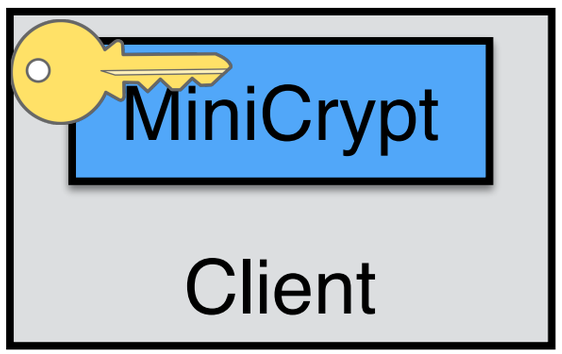
Append mode

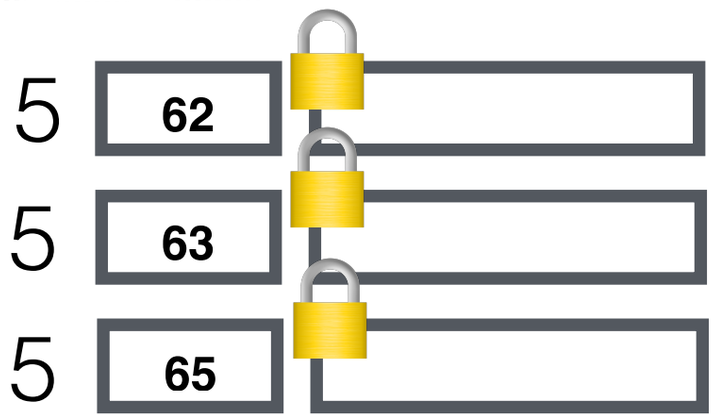

Current epoch: 5
4
50
4
53
4
61
5
62
5
63
5
65
time
MiniCrypt
Client
10:
15:
27:
33:
ZIP
ZIP
ZIP
ZIP

Append mode

Current epoch: 5







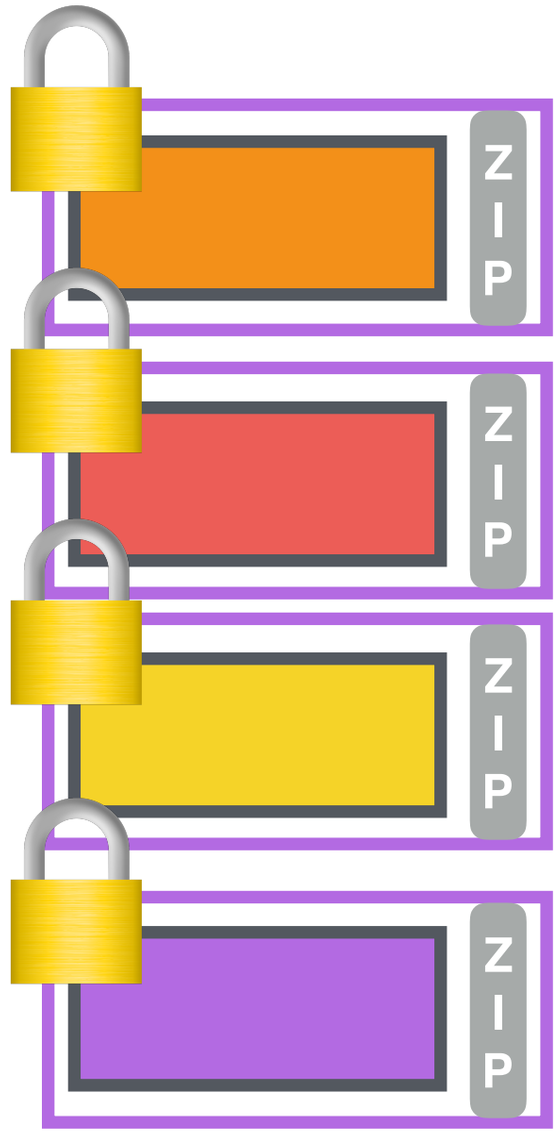
time

10:

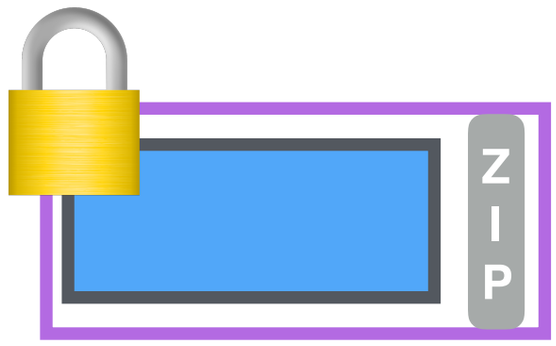
15:

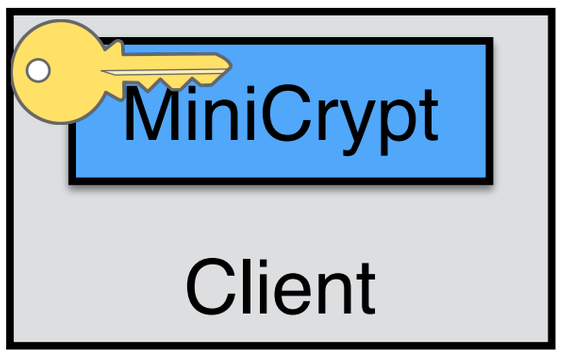
27:

33:

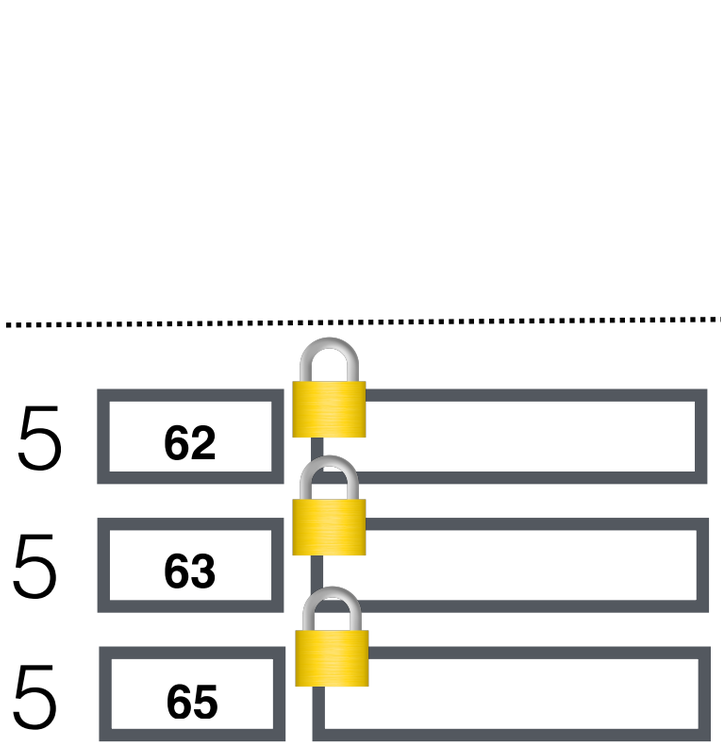


Append mode





Current epoch: 5



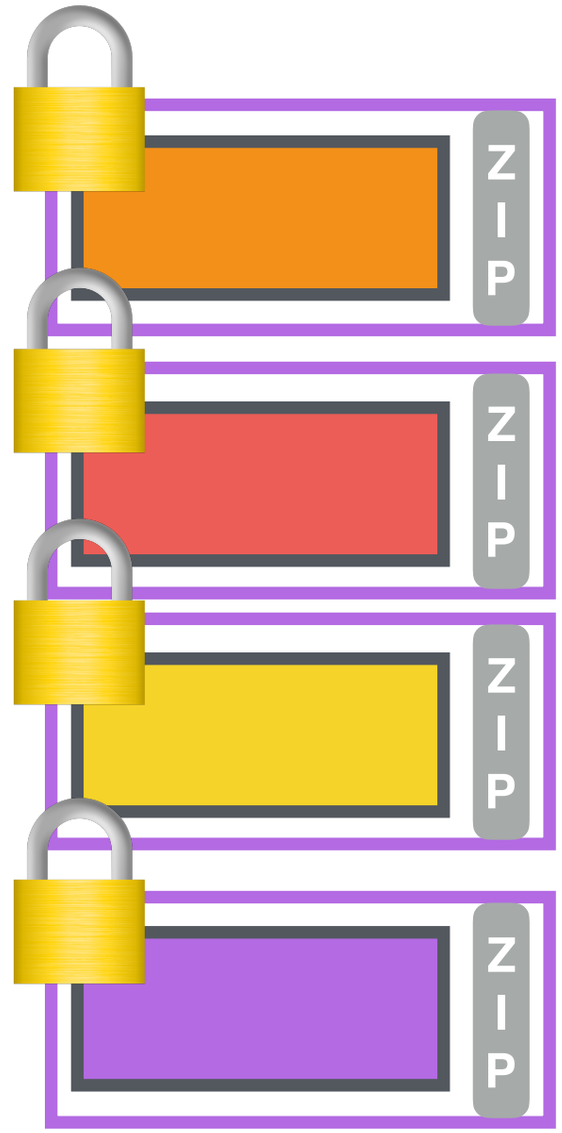
time

10:

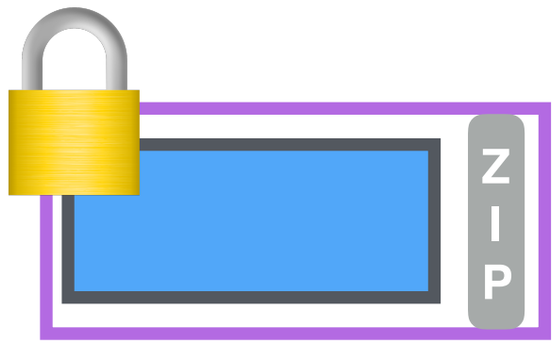
15:

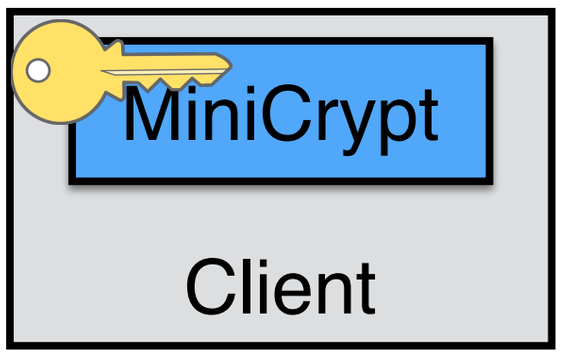
27:

33:

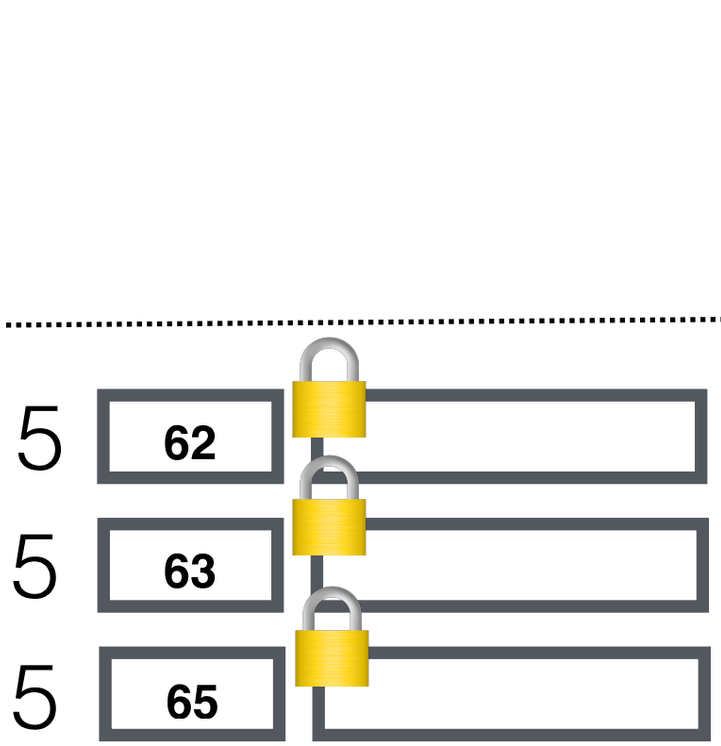


Append mode





Current epoch: 5



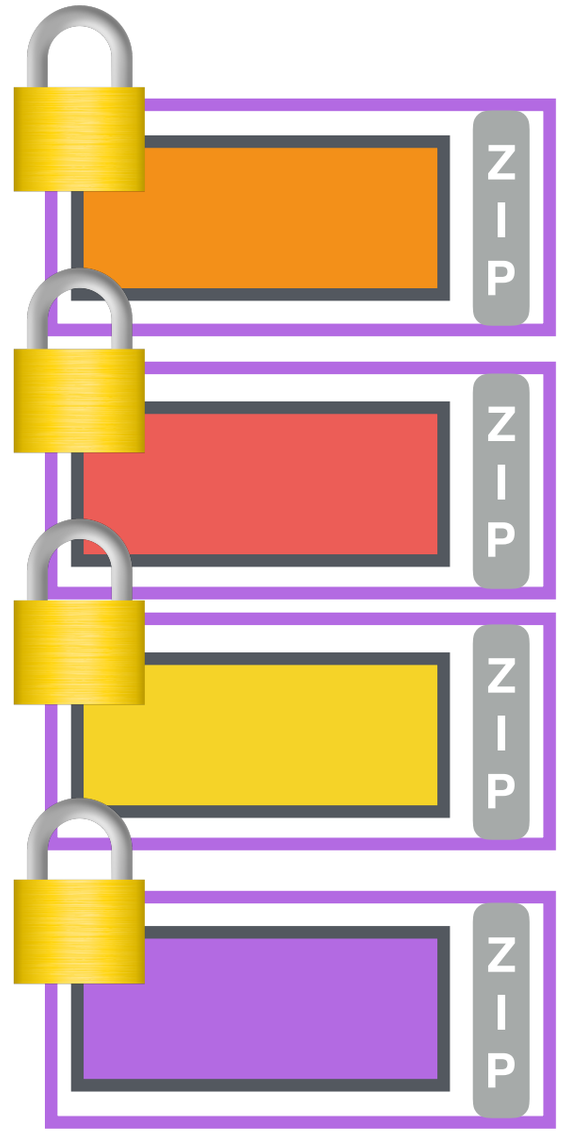
time

10:

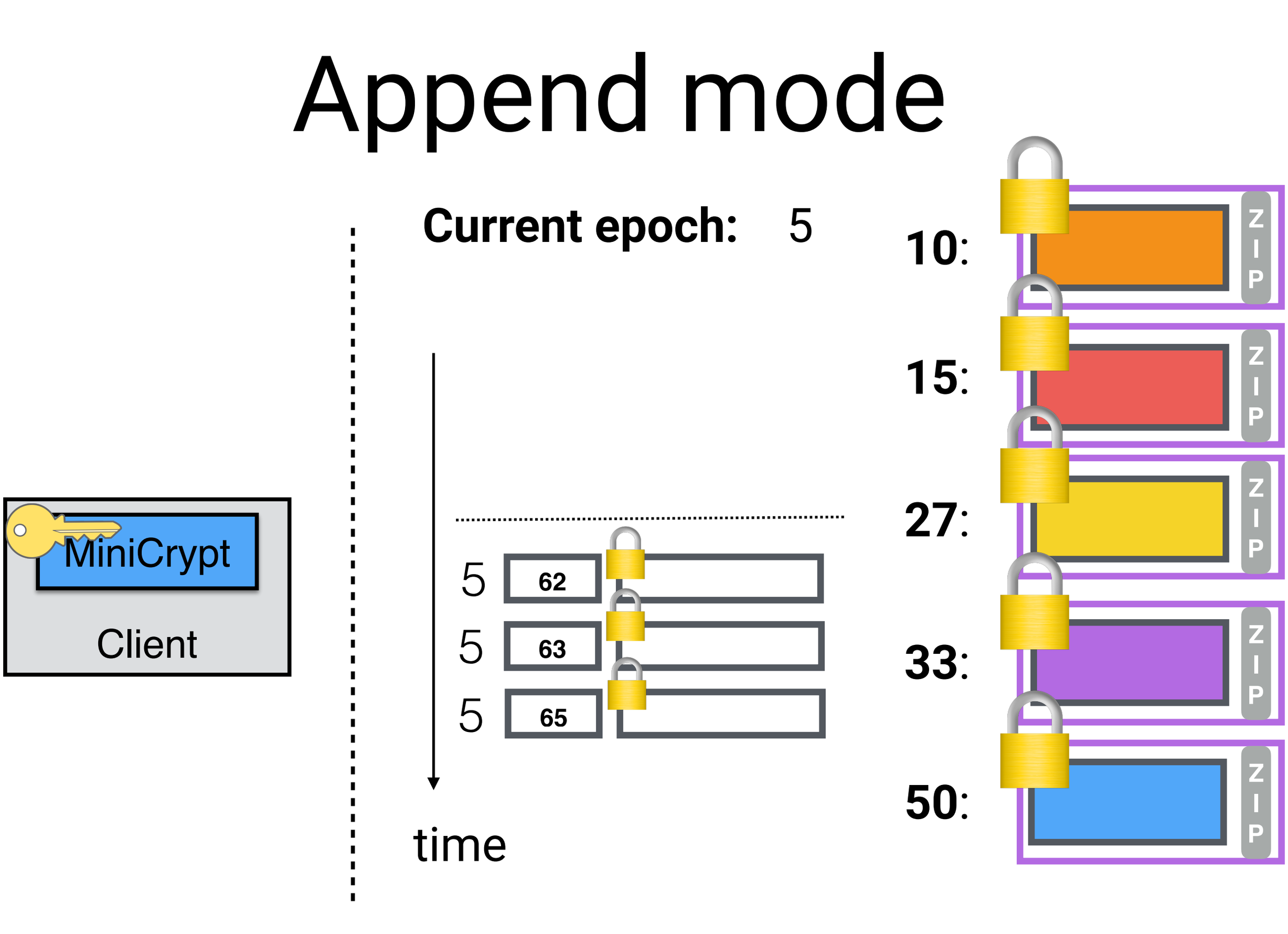
15:

27:

33:



Append mode


Current epoch: 5
MiniCrypt
Client
5
62
5
63
5
65
time
10:
15:
27:
33:
50:
ZIP
ZIP
ZIP
ZIP
ZIP

Evaluation

Evaluation

- MiniCrypt built on ***unmodified*** Cassandra

Evaluation

- MiniCrypt built on ***unmodified*** Cassandra
- Comparison with

Evaluation

- MiniCrypt built on ***unmodified*** Cassandra
- Comparison with
  - Baseline: each row compressed and encrypted

Evaluation

- MiniCrypt built on ***unmodified*** Cassandra
- Comparison with
  - Baseline: each row compressed and encrypted
  - Vanilla: no encryption

Evaluation

- MiniCrypt built on ***unmodified*** Cassandra
- Comparison with
  - Baseline: each row compressed and encrypted
  - Vanilla: no encryption
- YCSB benchmark

Evaluation

- MiniCrypt built on ***unmodified*** Cassandra
- Comparison with
  - Baseline: each row compressed and encrypted
  - Vanilla: no encryption
- YCSB benchmark
  - Anonymized Conviva data (video analytics)

Evaluation setup

Evaluation setup

- Amazon EC2 cluster of 3 instances

Evaluation setup

- Amazon EC2 cluster of 3 instances
  - 15 GB of memory

Evaluation setup

- Amazon EC2 cluster of 3 instances
  - 15 GB of memory
  - 8 cores

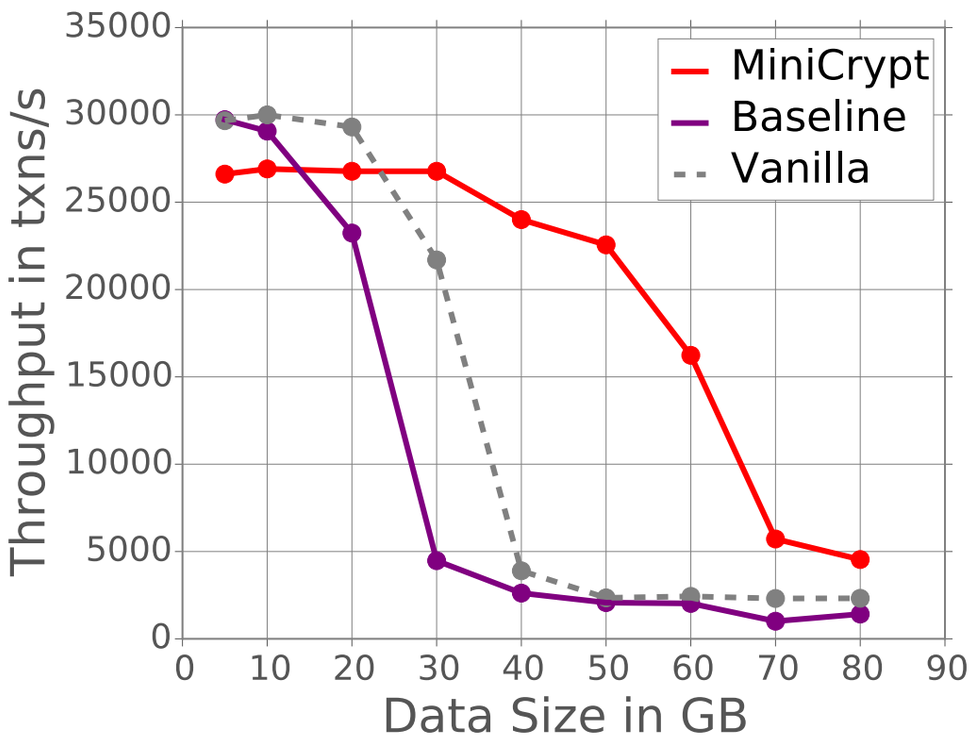
Evaluation setup

- Amazon EC2 cluster of 3 instances
  - 15 GB of memory
  - 8 cores
- Zlib compression

Evaluation setup

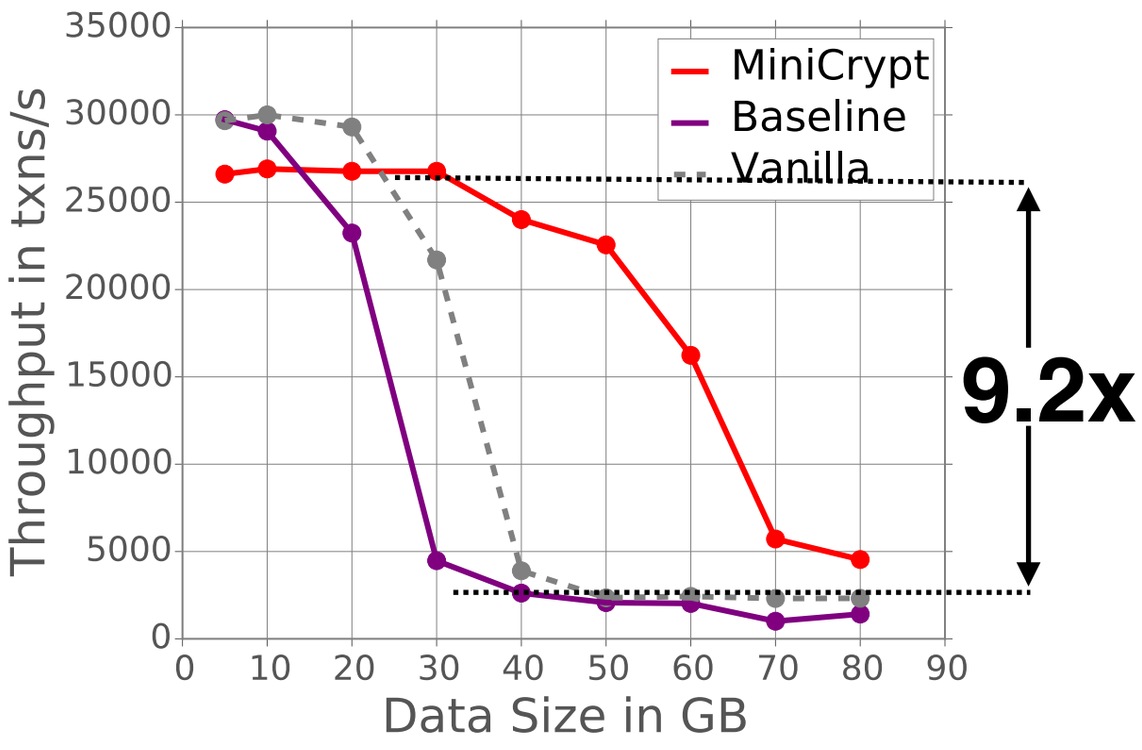
- Amazon EC2 cluster of 3 instances
  - 15 GB of memory
  - 8 cores
- Zlib compression
- AES-256 CBC for encryption

YCSB
100% single-record read



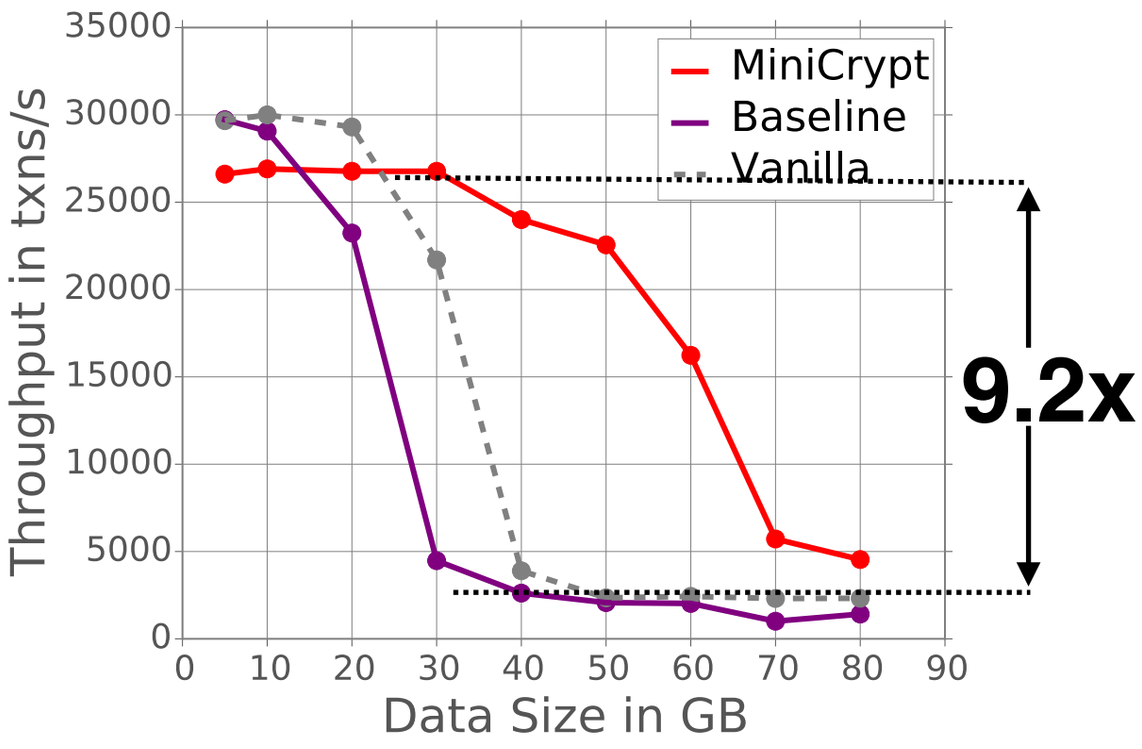
SSD

YCSB
100% single-record read

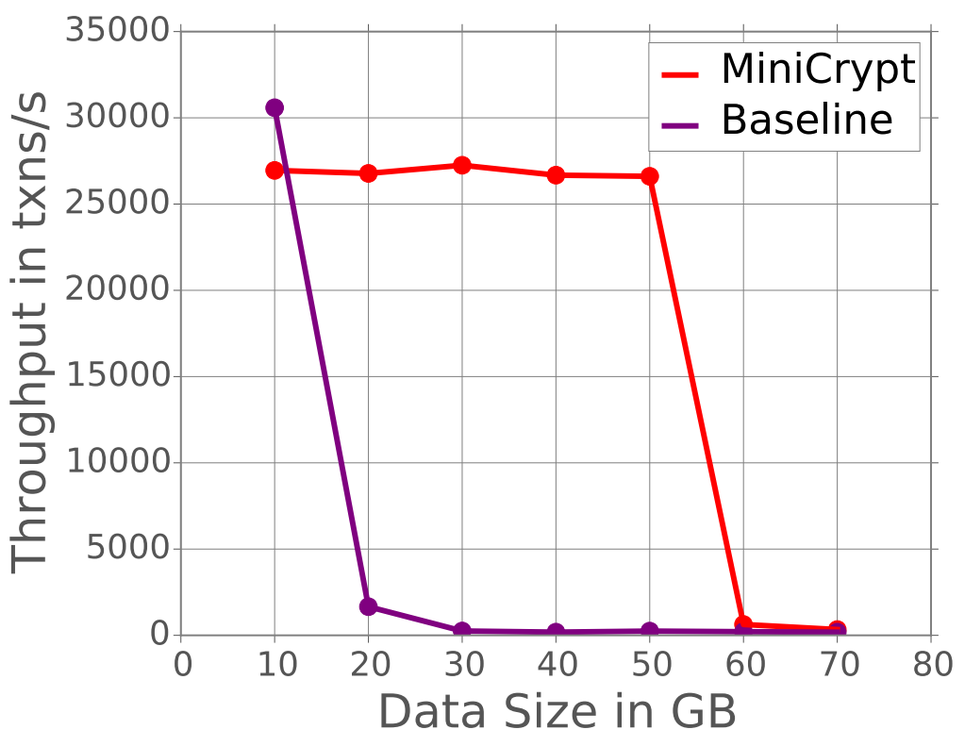


SSD

YCSB
100% single-record read

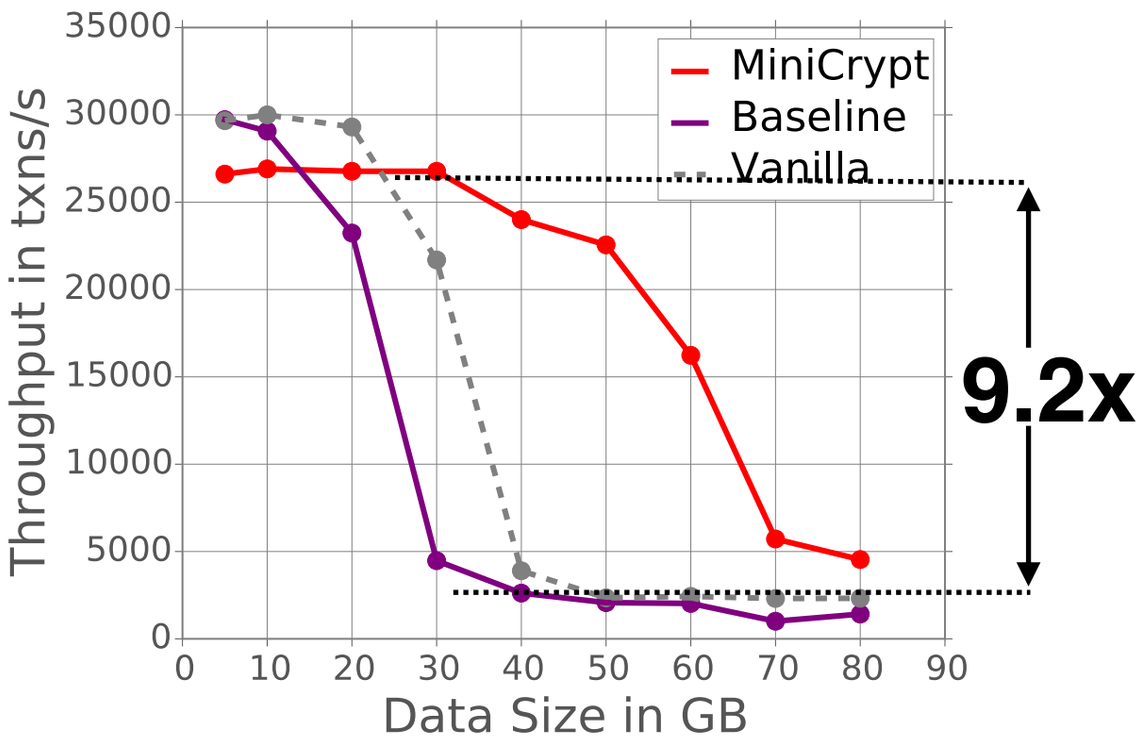


SSD

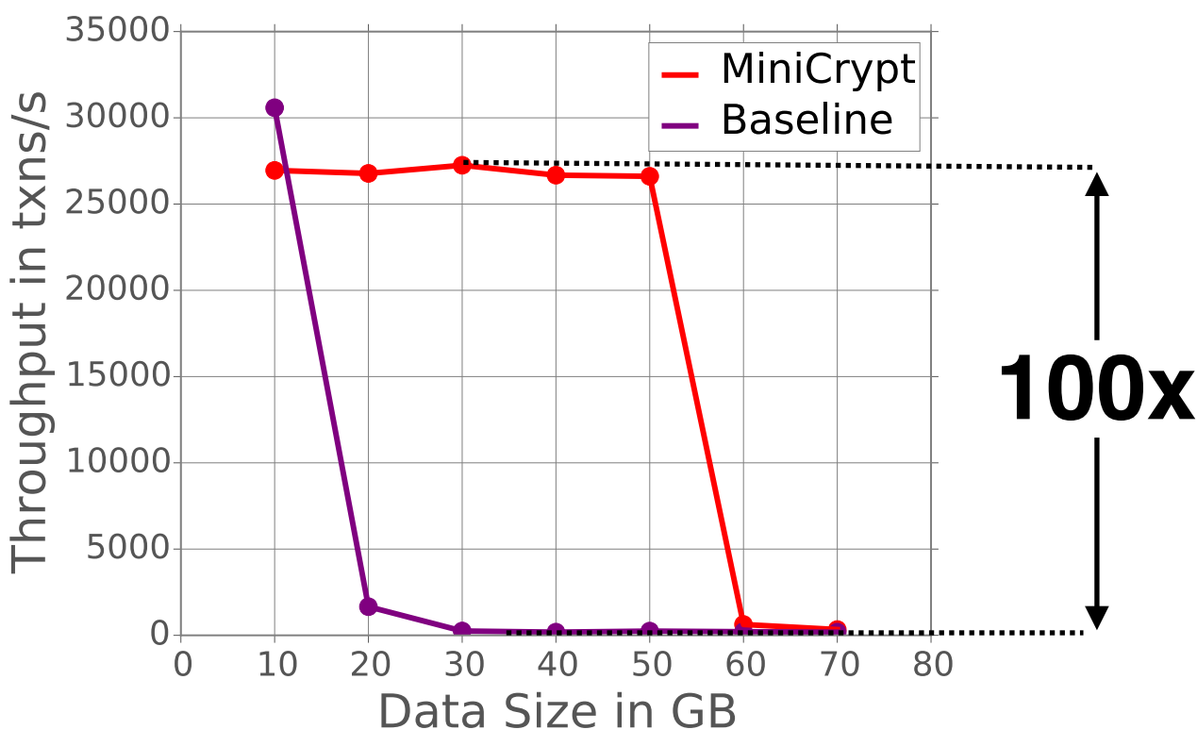


Disk

YCSB
100% single-record read

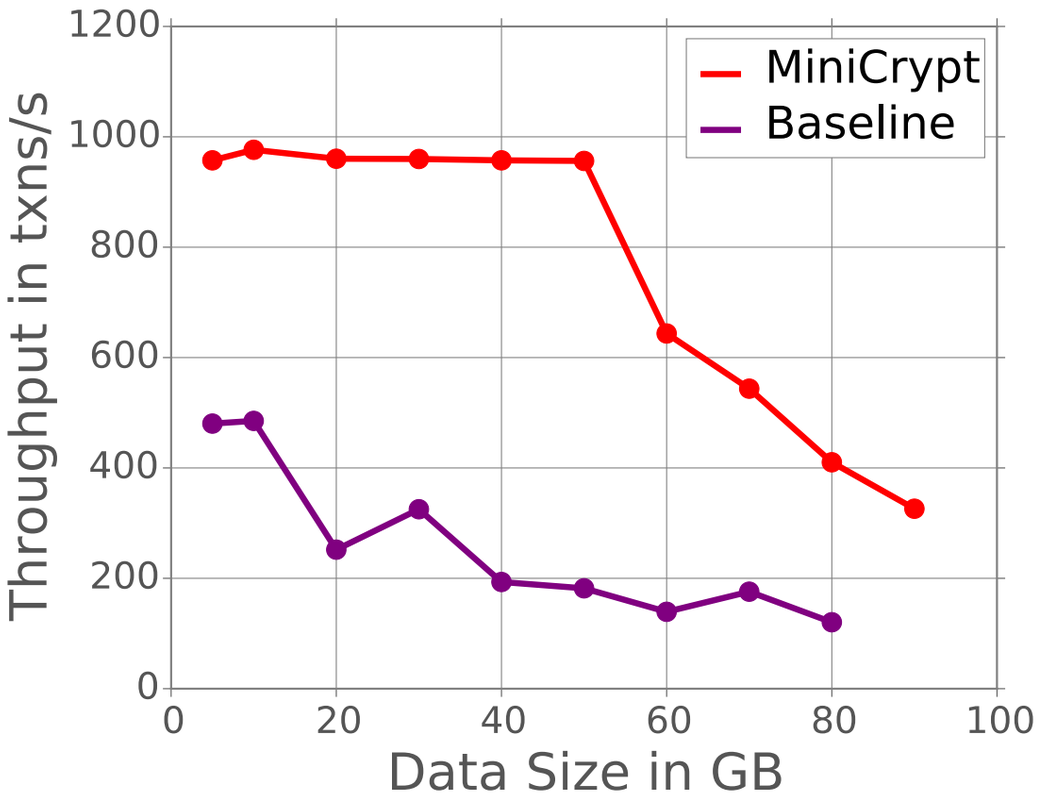


SSD



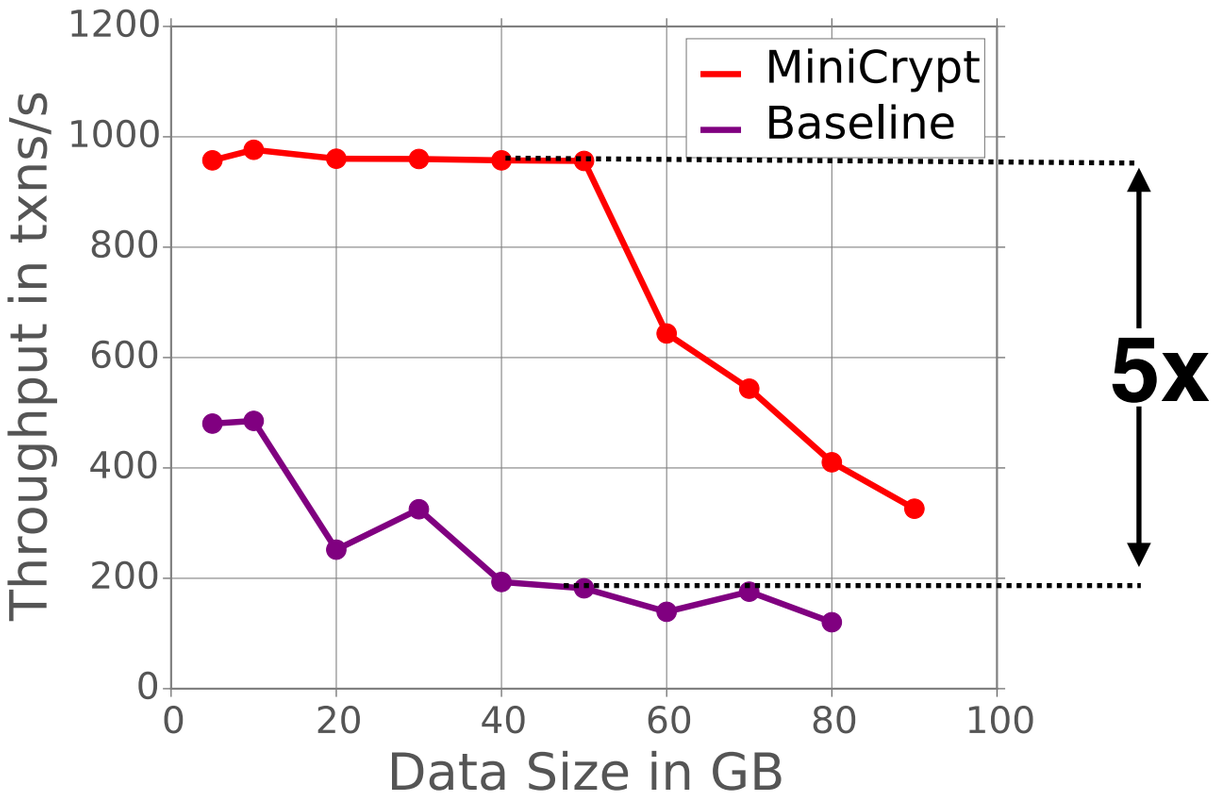
Disk

YCSB
100% range query



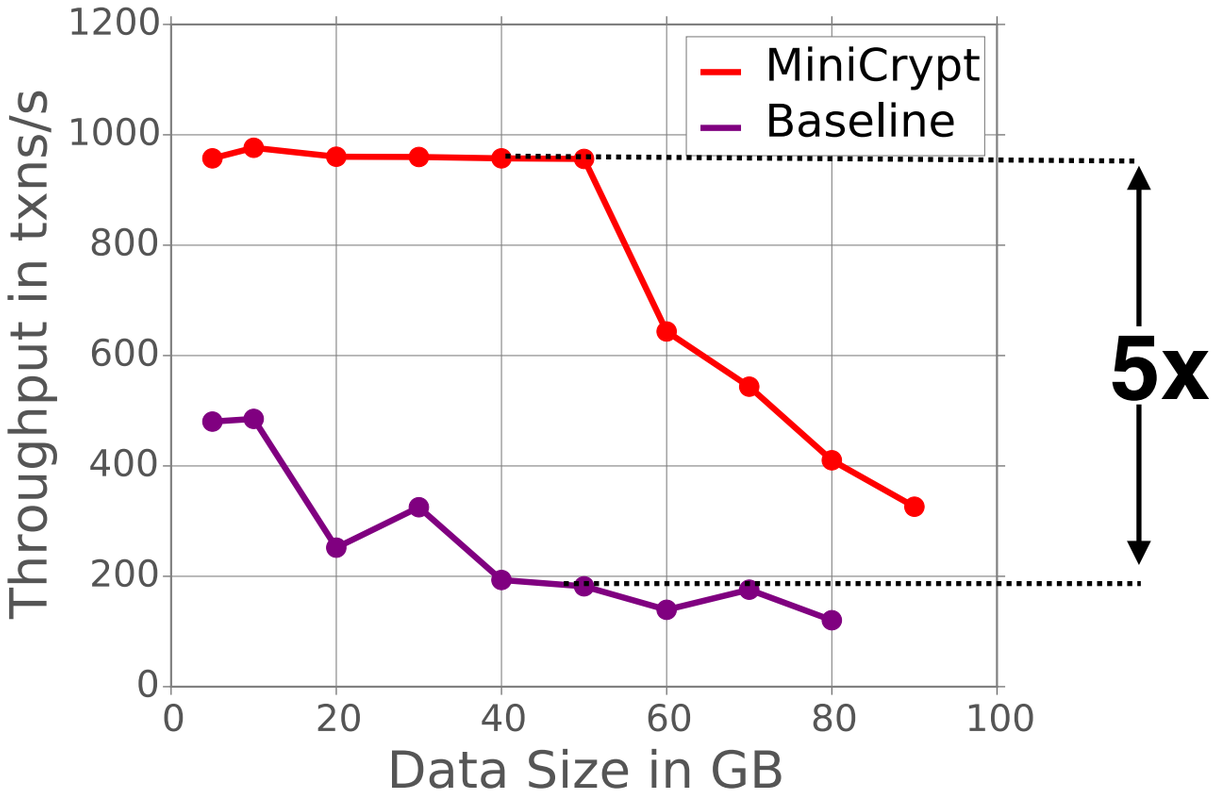
SSD

YCSB
100% range query

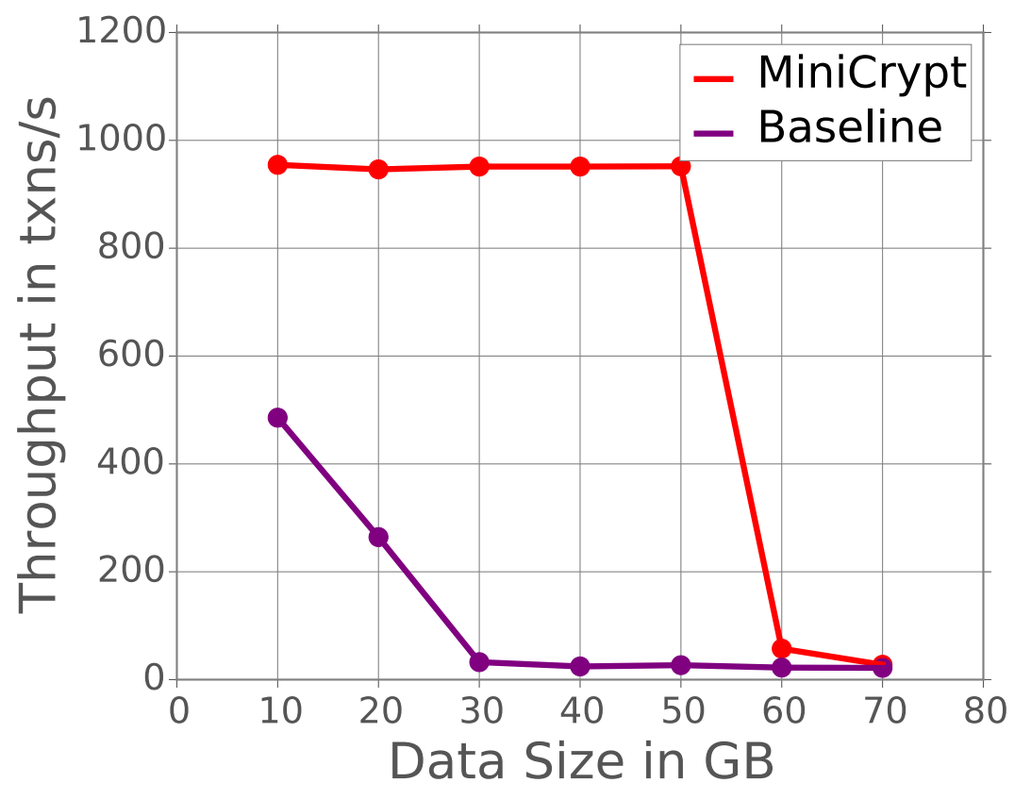


SSD

YCSB
100% range query

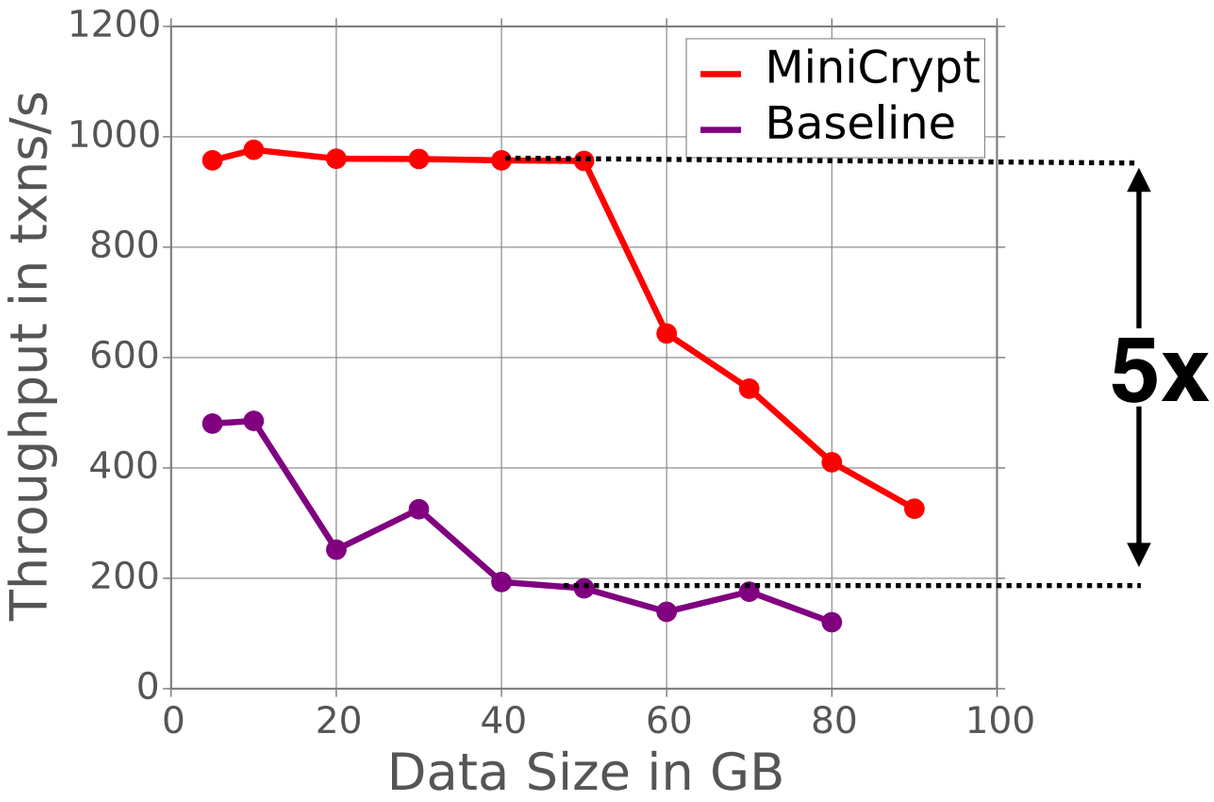


SSD

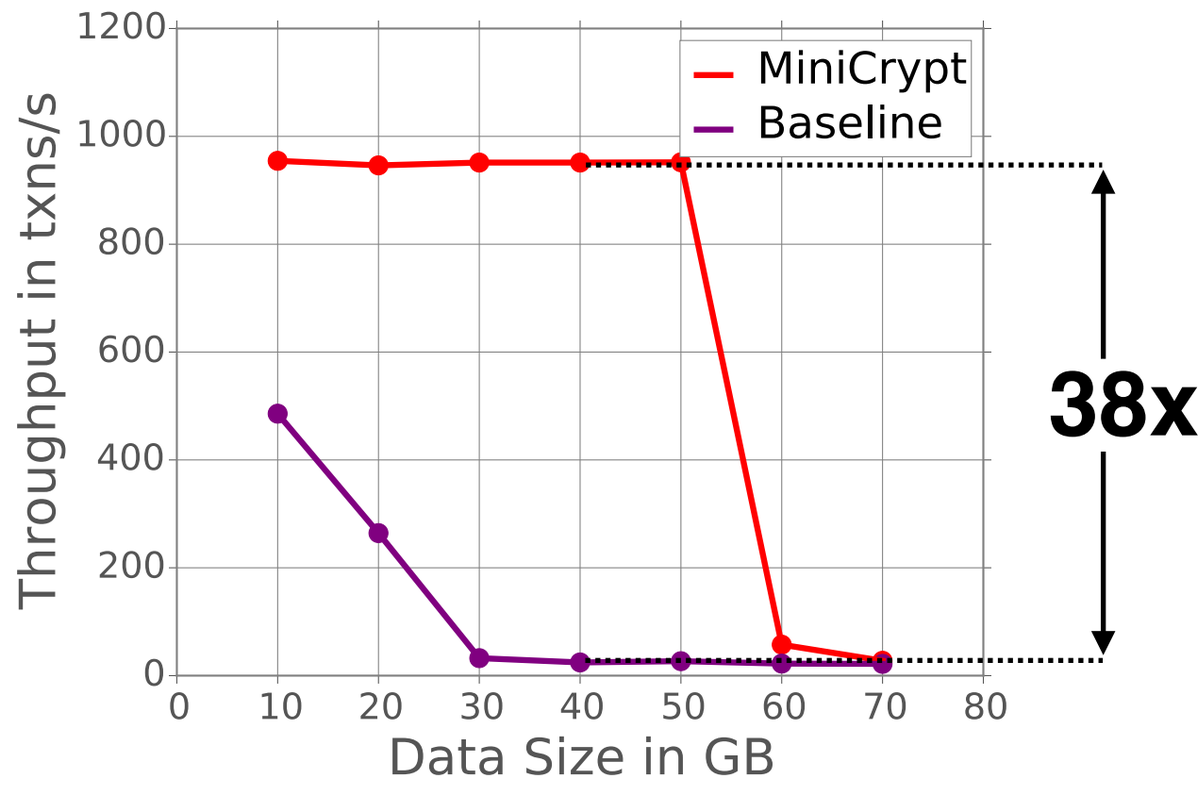


Disk

YCSB
100% range query



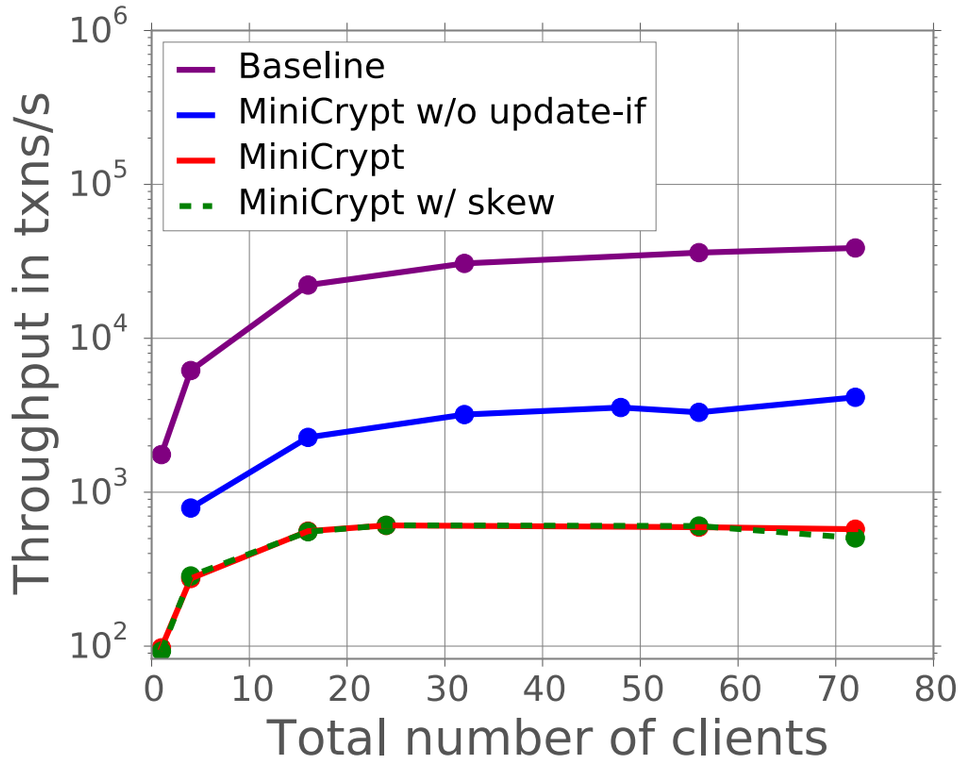
SSD



Disk

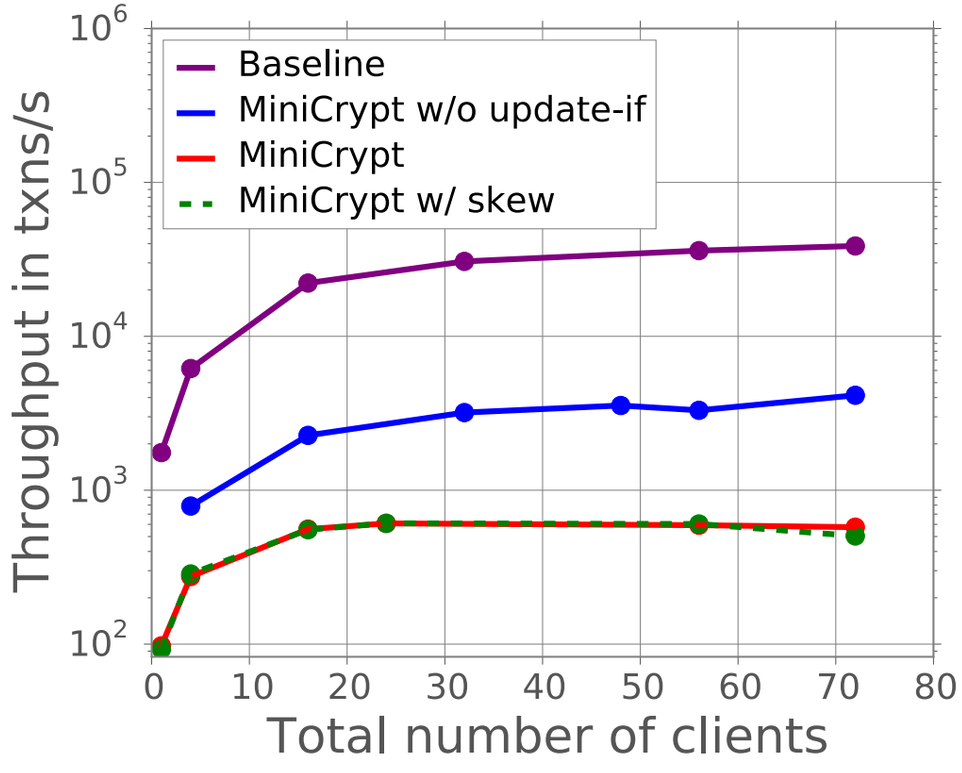
YCSB
100% single-record write

Default

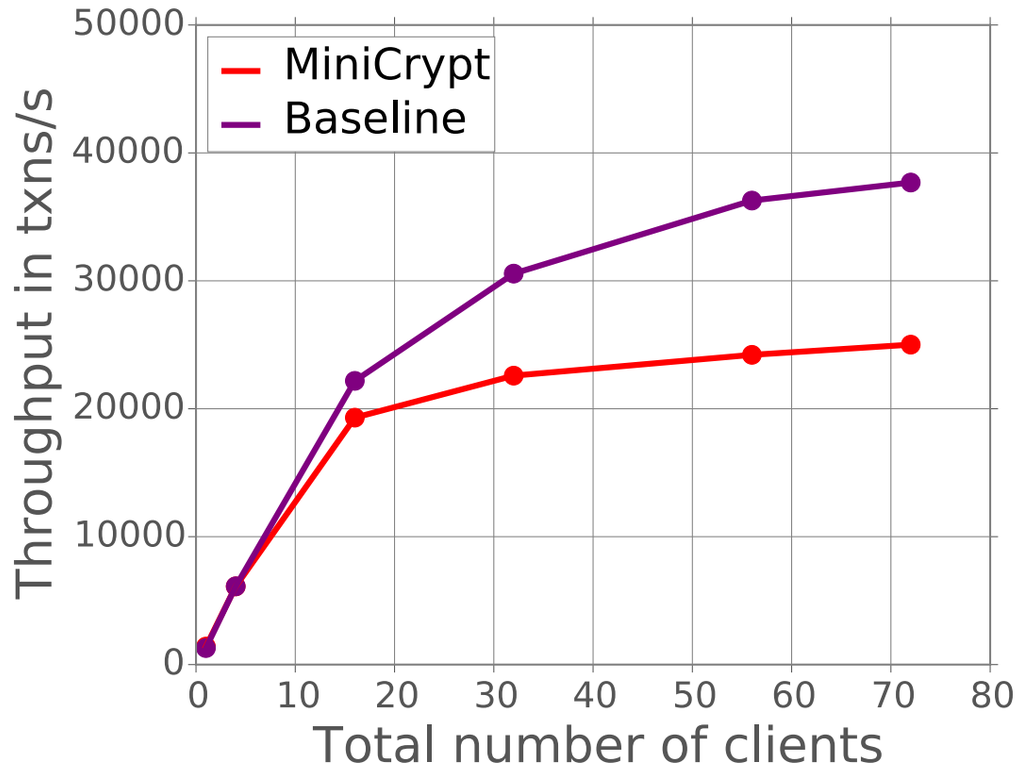


YCSB
100% single-record write

Default



Append-only



Conclusion

MiniCrypt shows that we can achieve compression and encryption using systems techniques

Conclusion

MiniCrypt shows that we can achieve compression and encryption using systems techniques

Thank you!