Akeso: Bringing Post-Compromise Security to Cloud Storage

Lily Gloudemans Pankaj Niroula Aashutosh Poudel

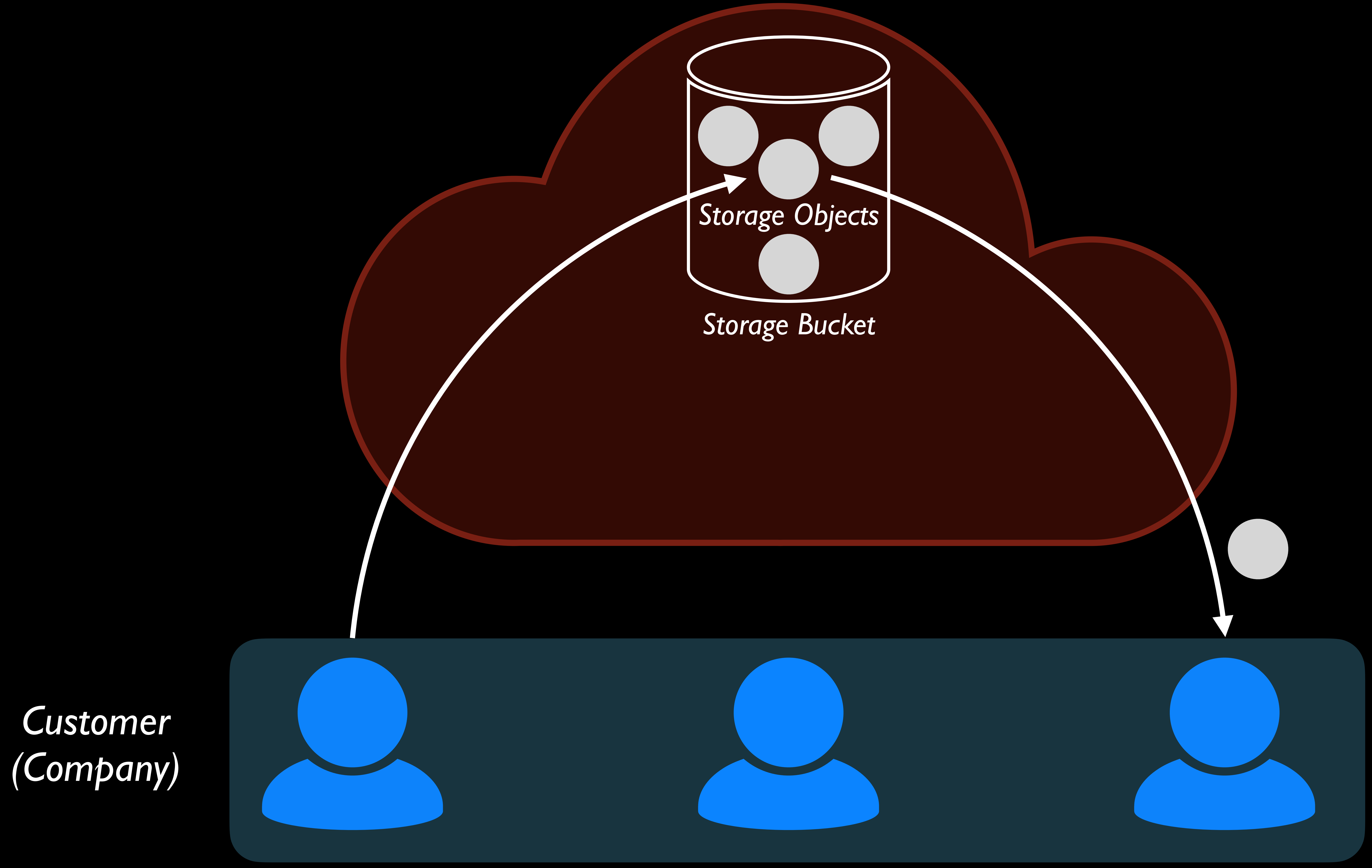
Collin MacDonald Stephen Herwig

WILLIAM & MARY

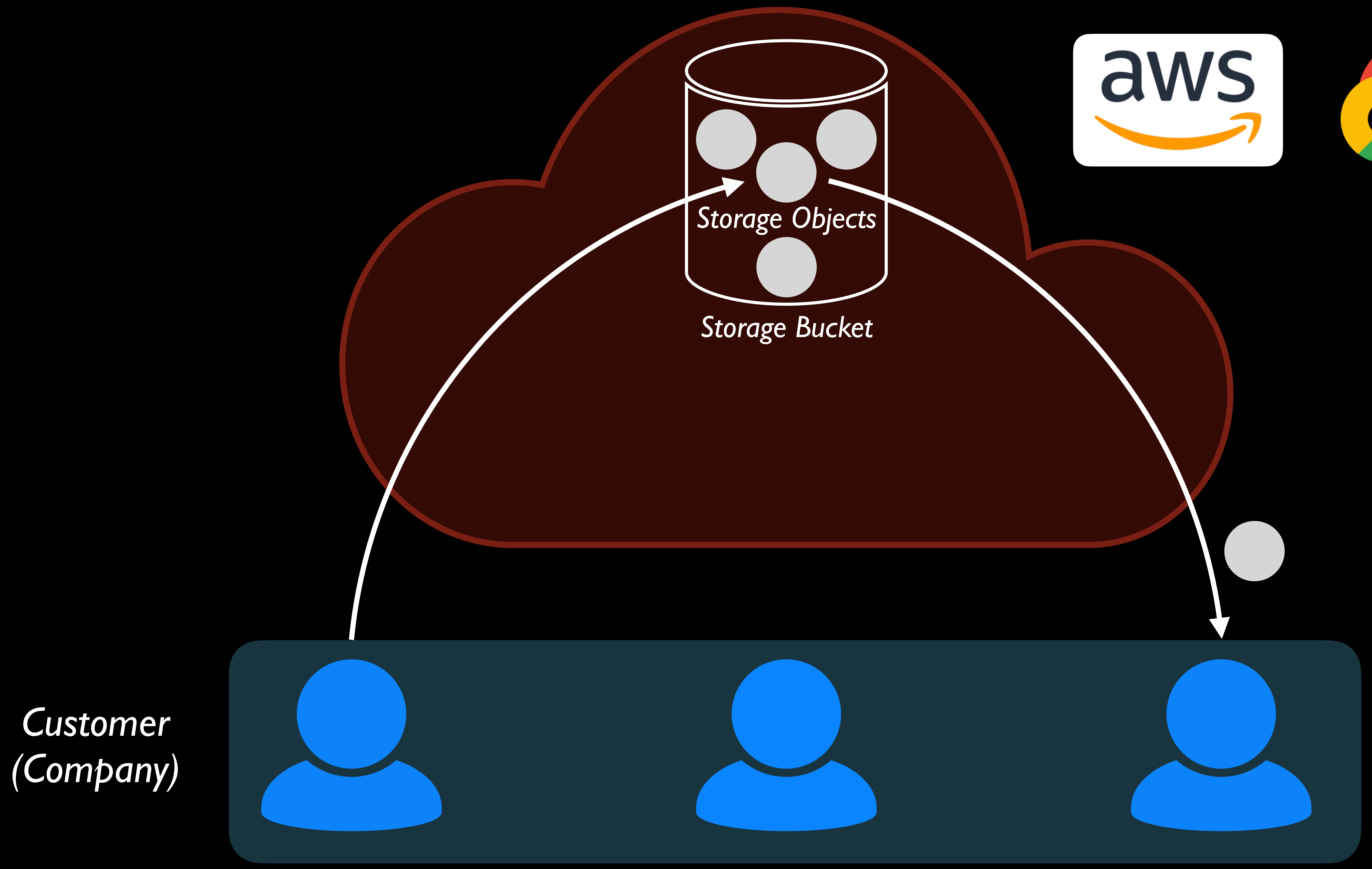
CHARTERED 1693

/etc/lab
Extending Trust in Computing Lab

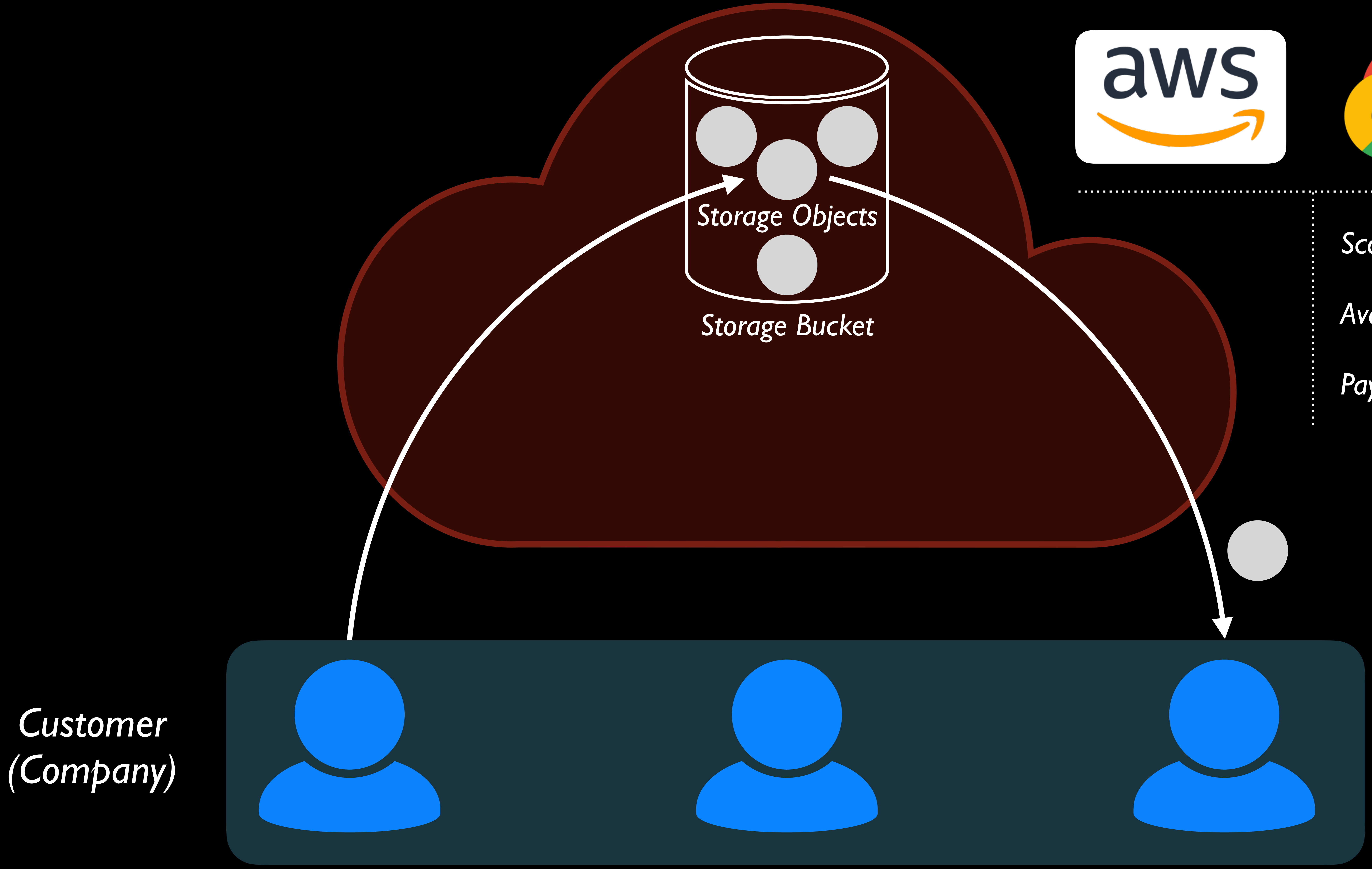
Cloud Object Storage



Cloud Object Storage



Cloud Object Storage

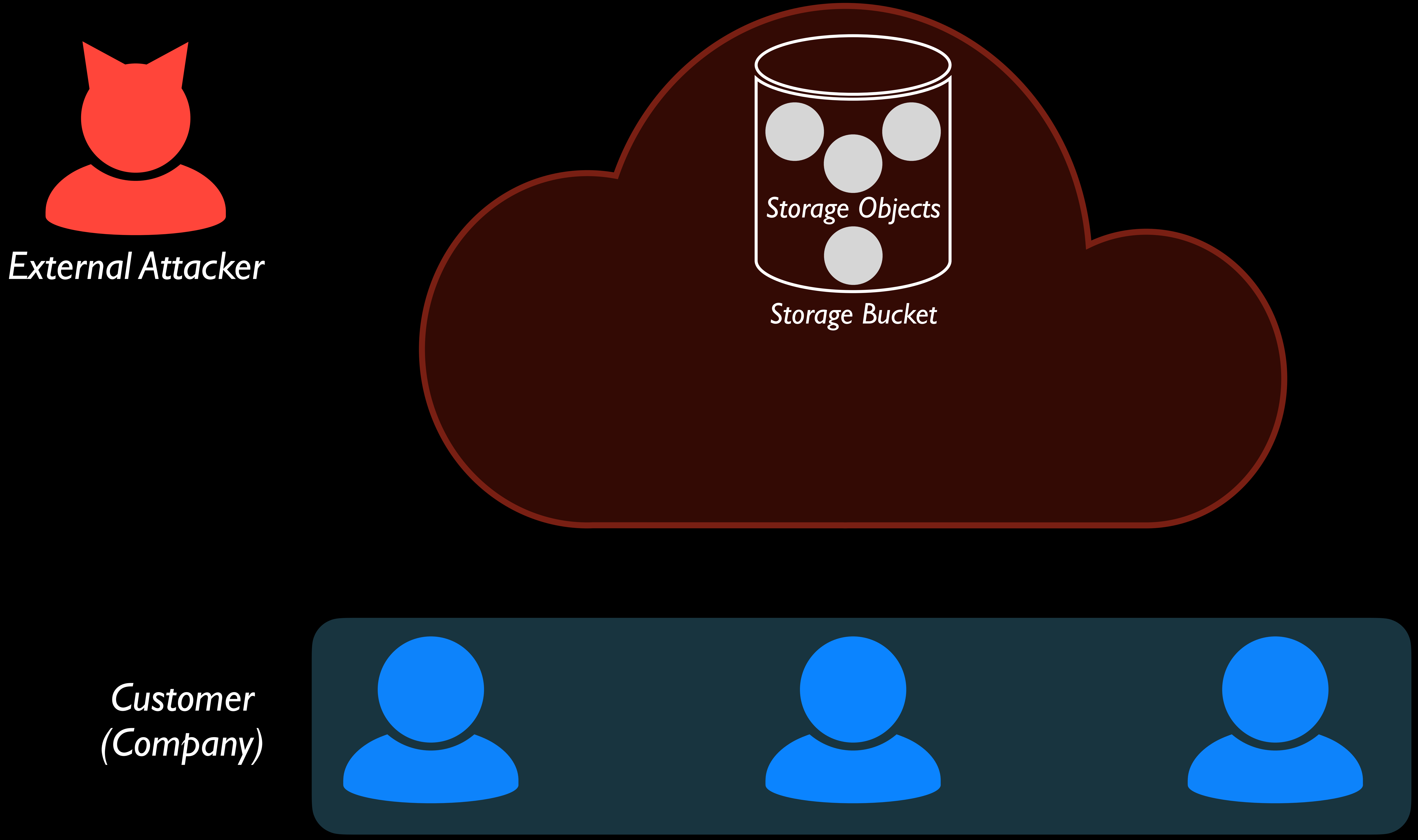


Scalability

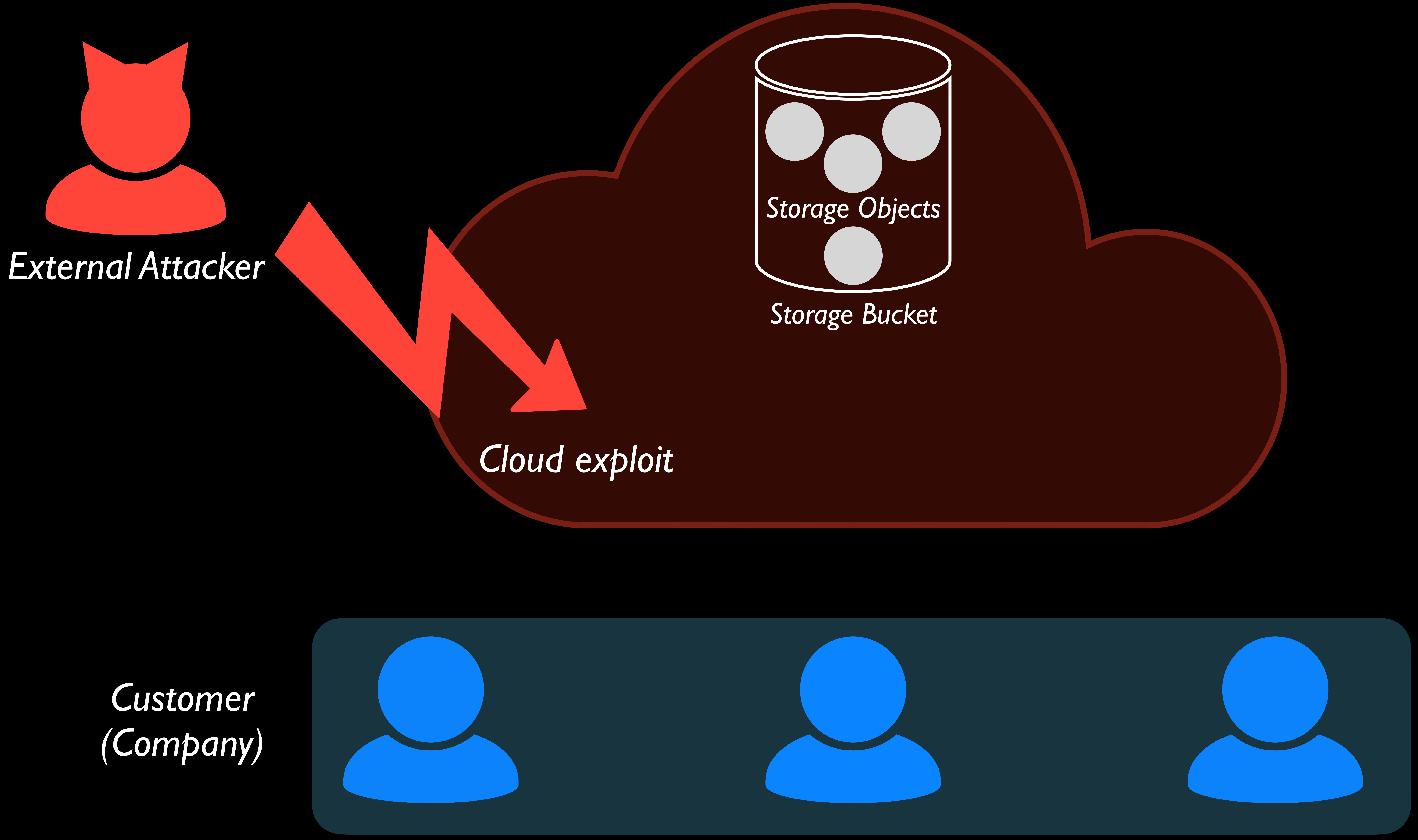
Availability

Pay-as-you-go

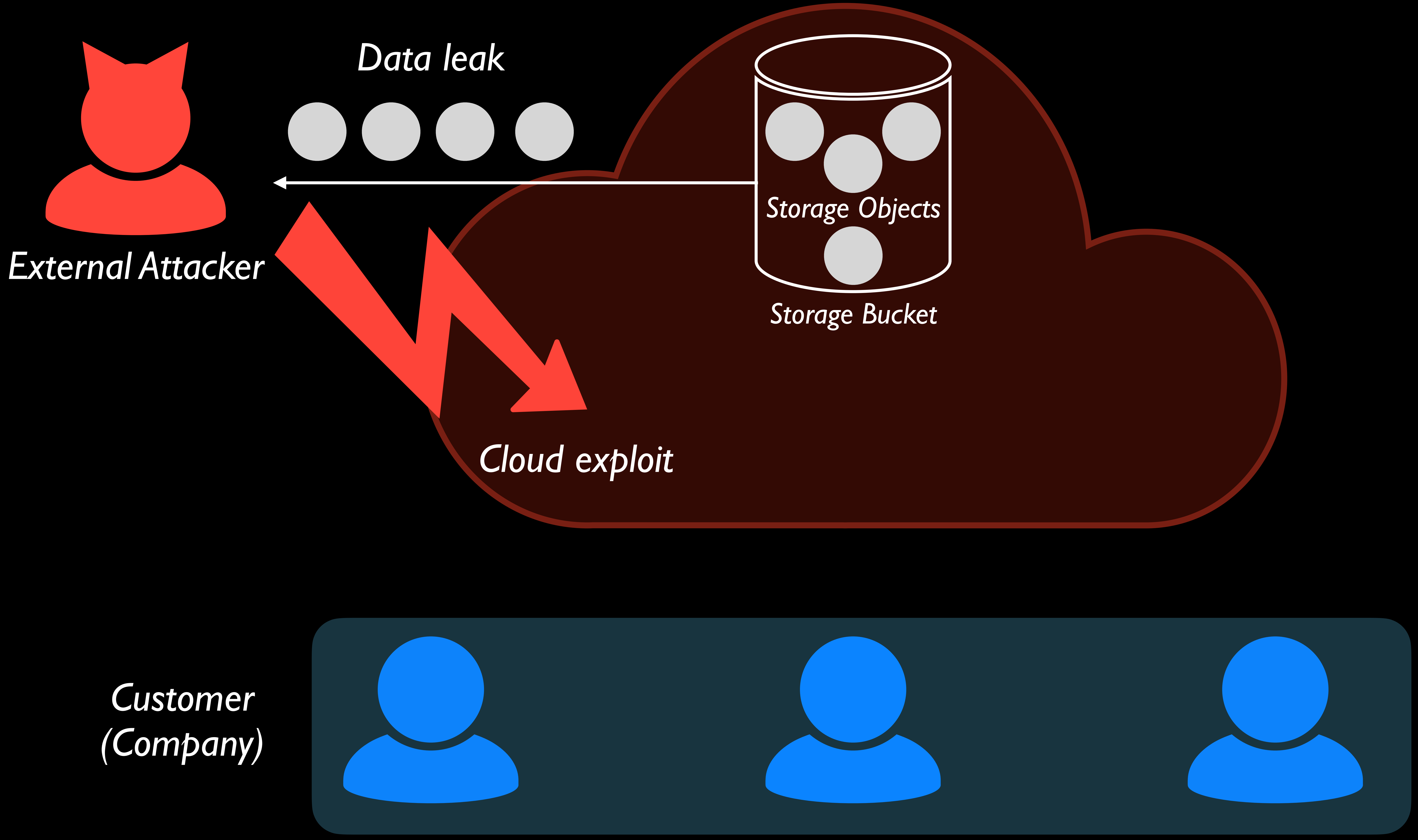
External Adversary



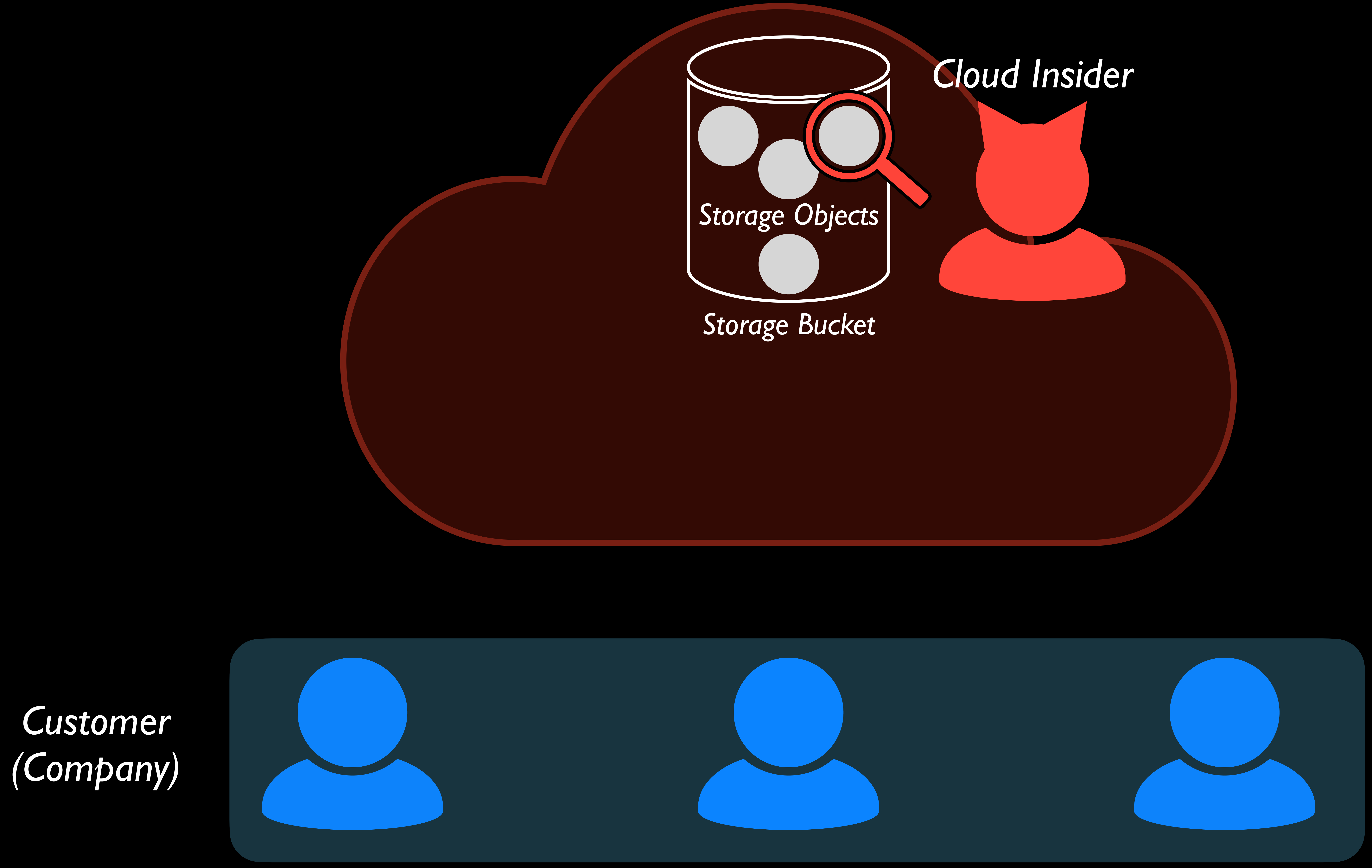
External Adversary



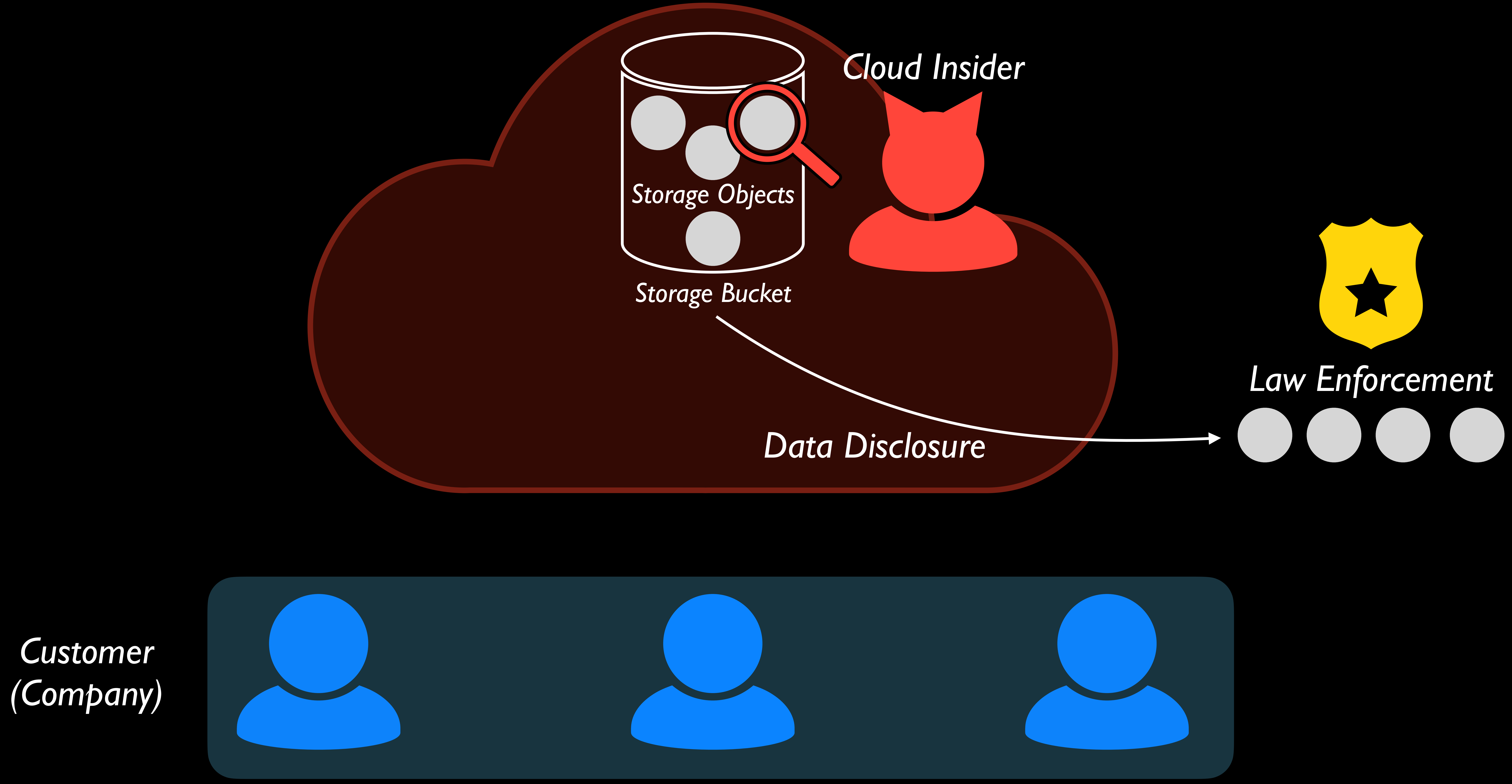
External Adversary



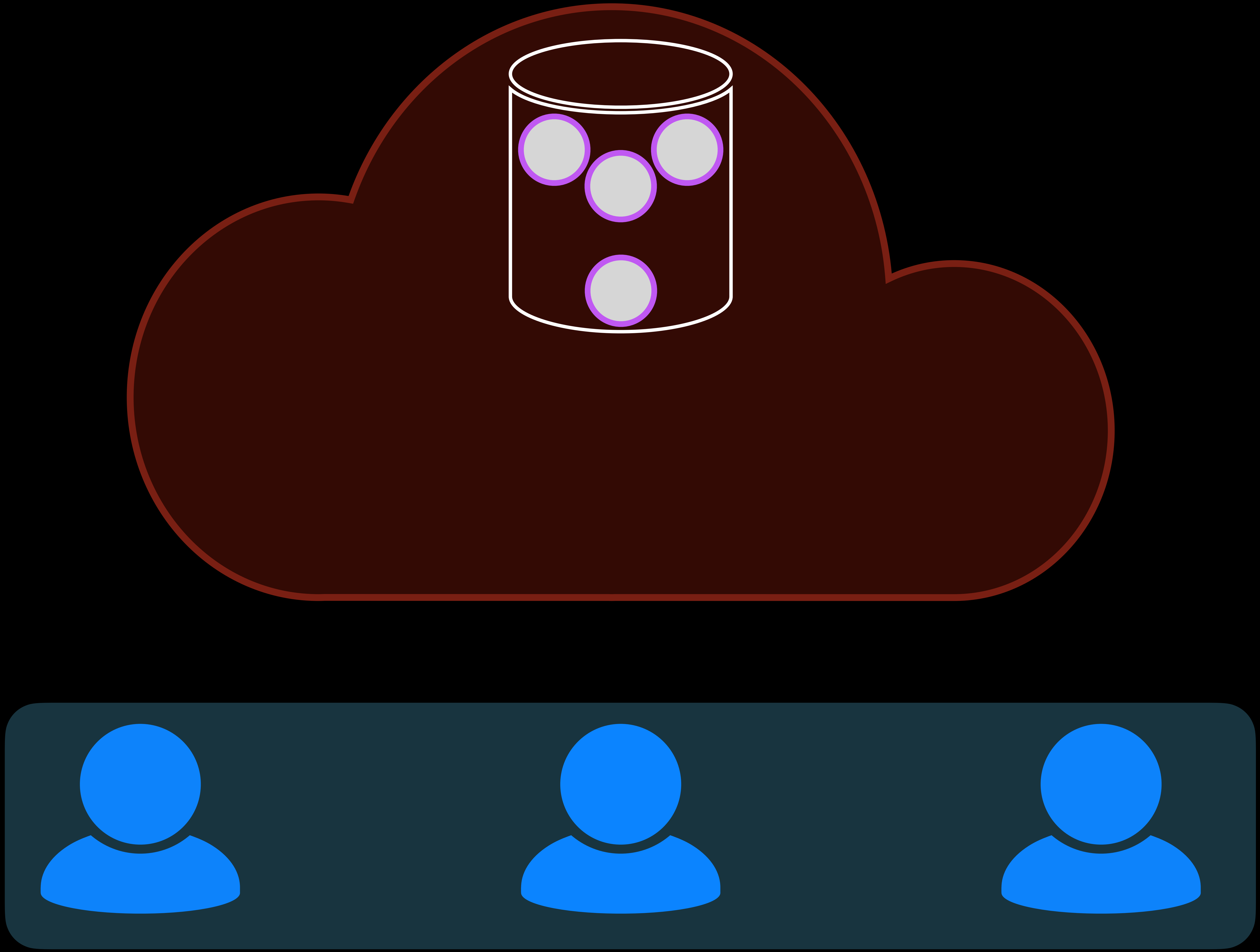
Cloud Adversary



Cloud Adversary

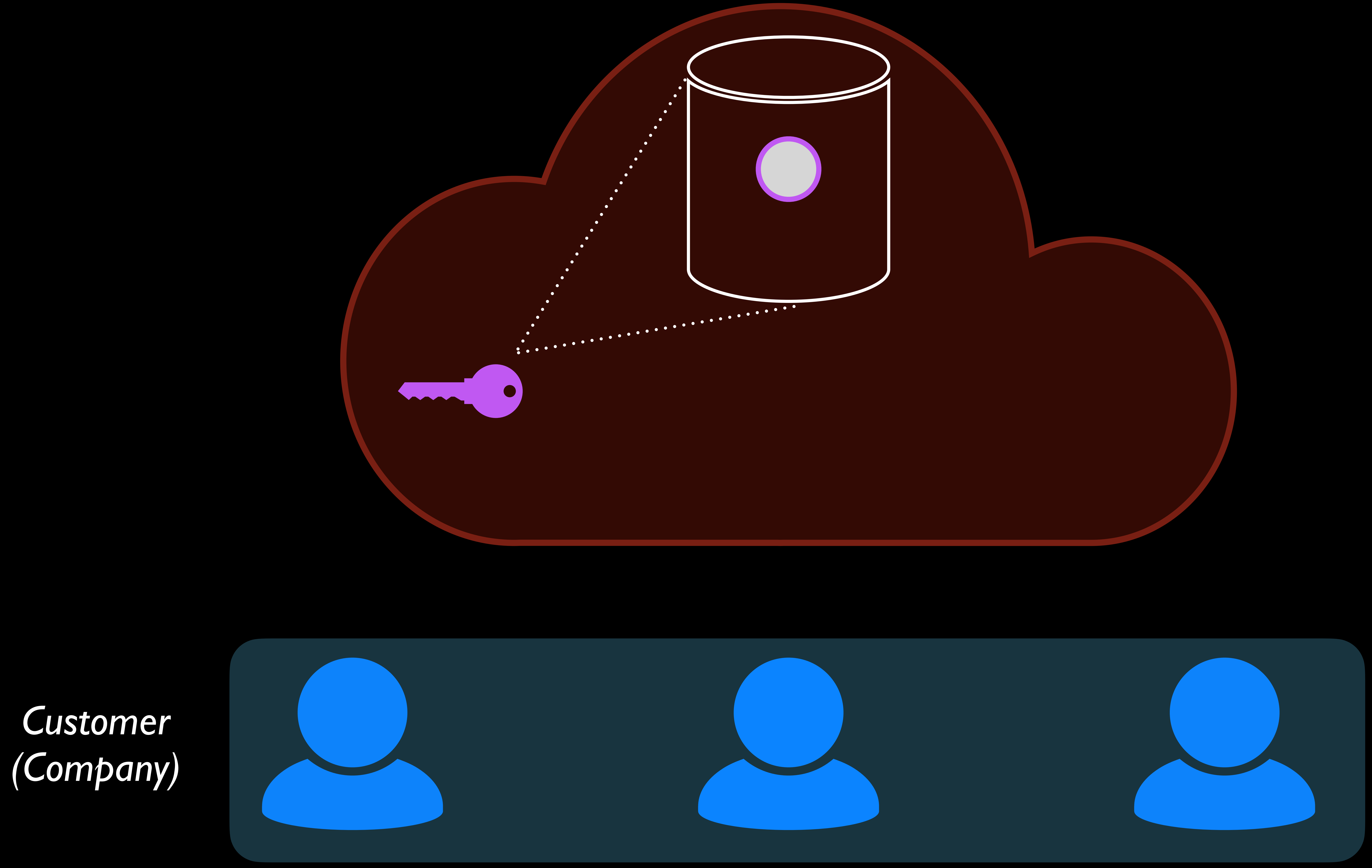


Cloud-Side Encryption

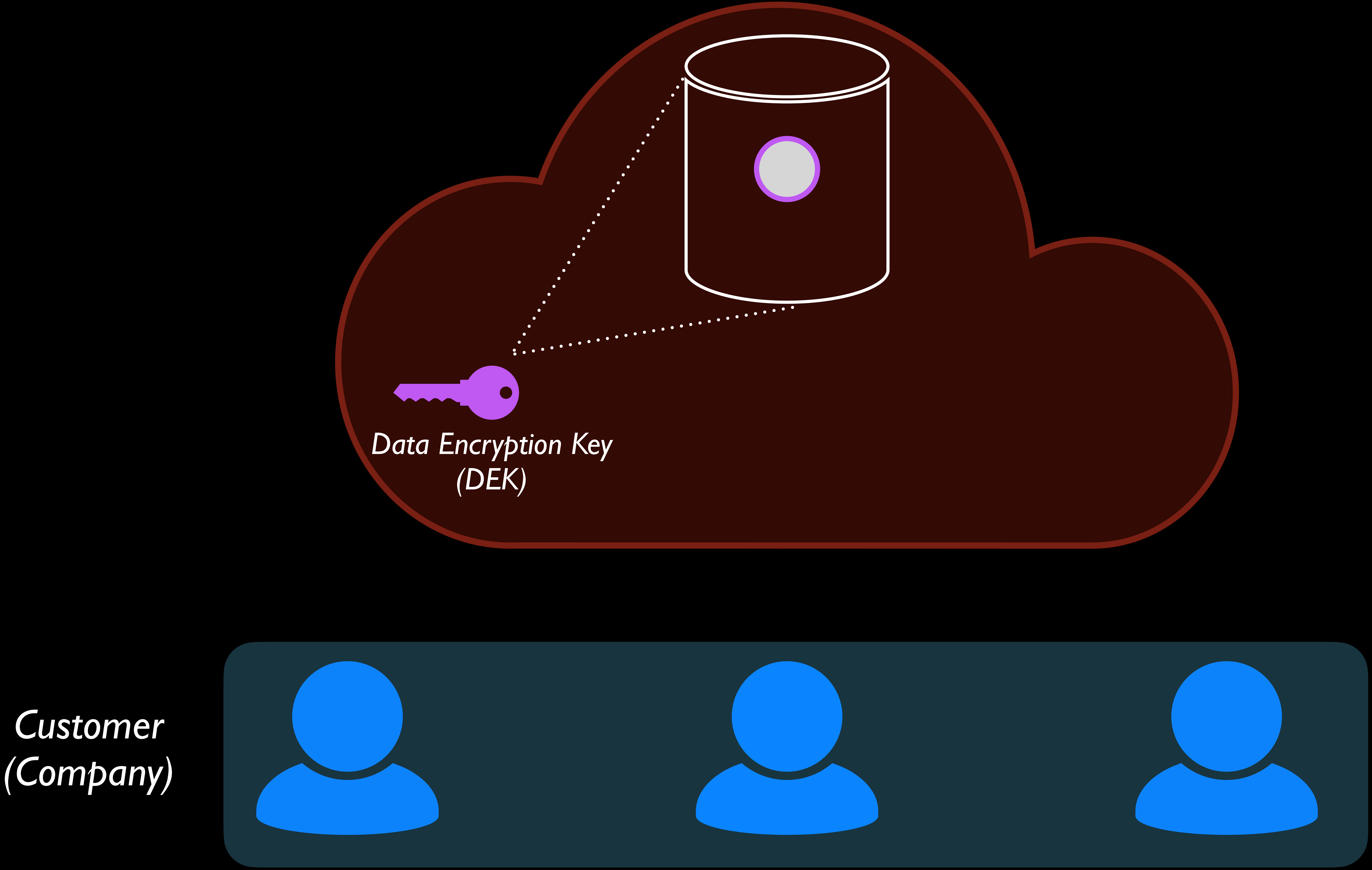


Customer (Company)

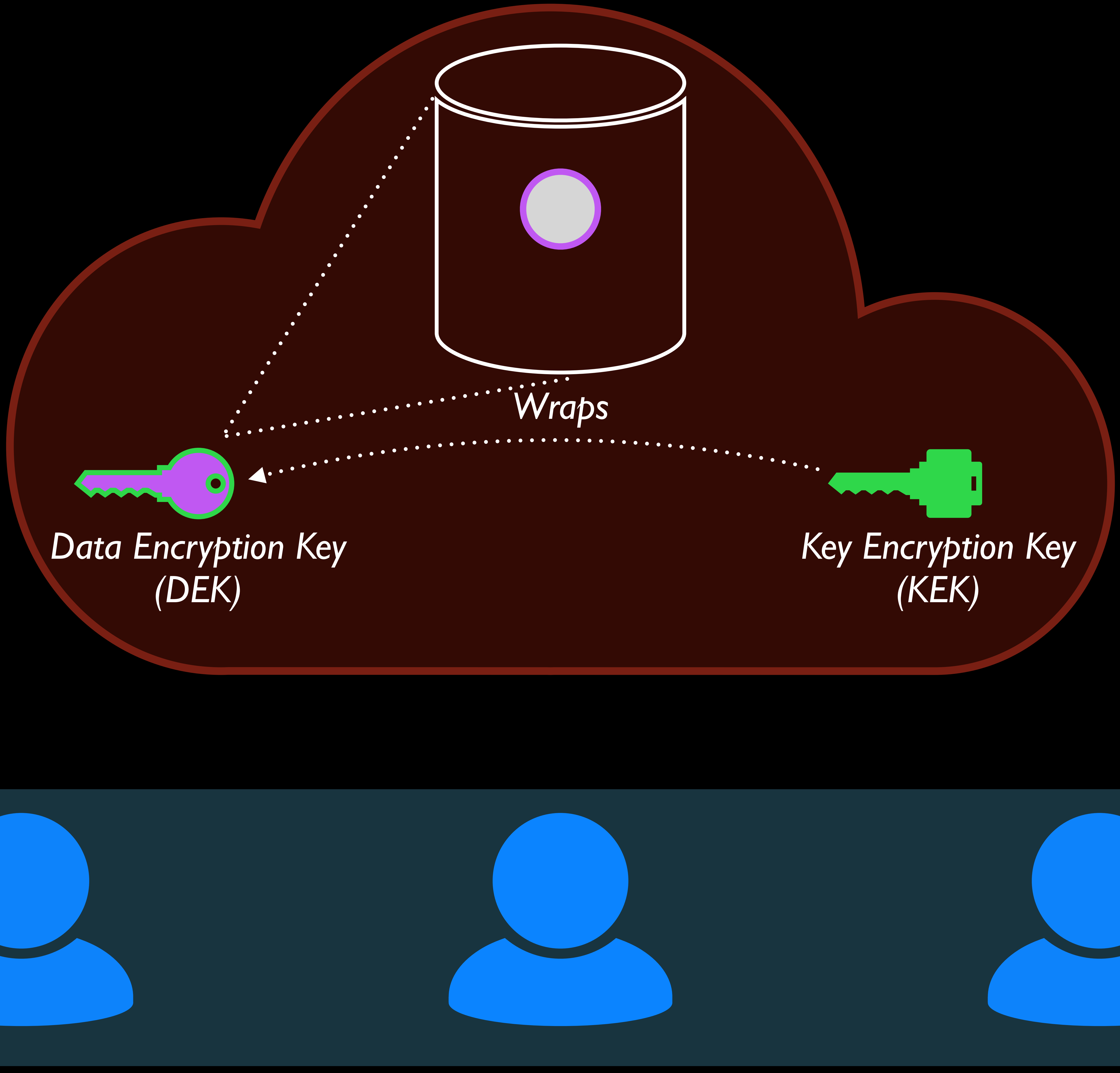
Cloud-Side Encryption



Cloud-Side Encryption

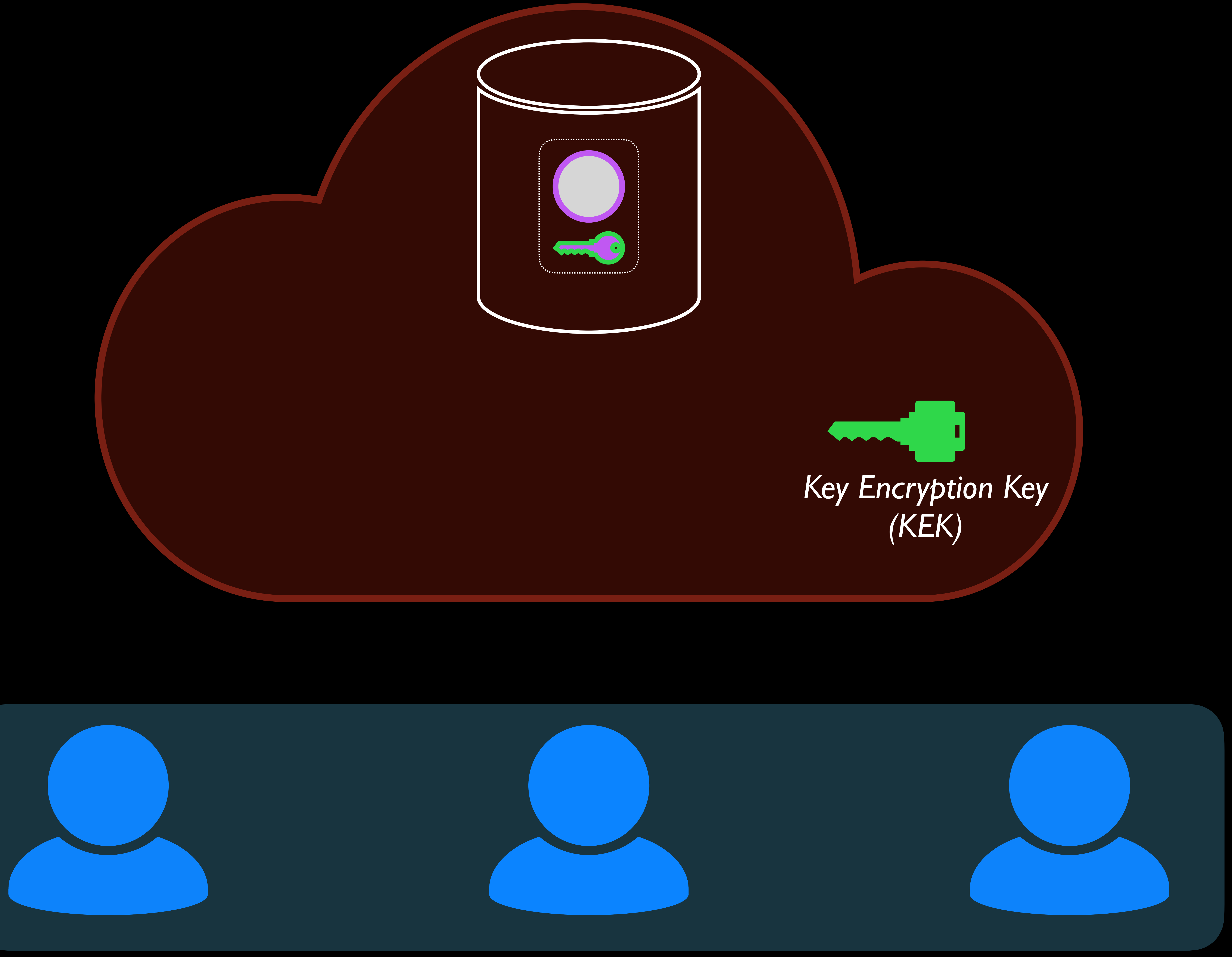


Cloud-Side Encryption

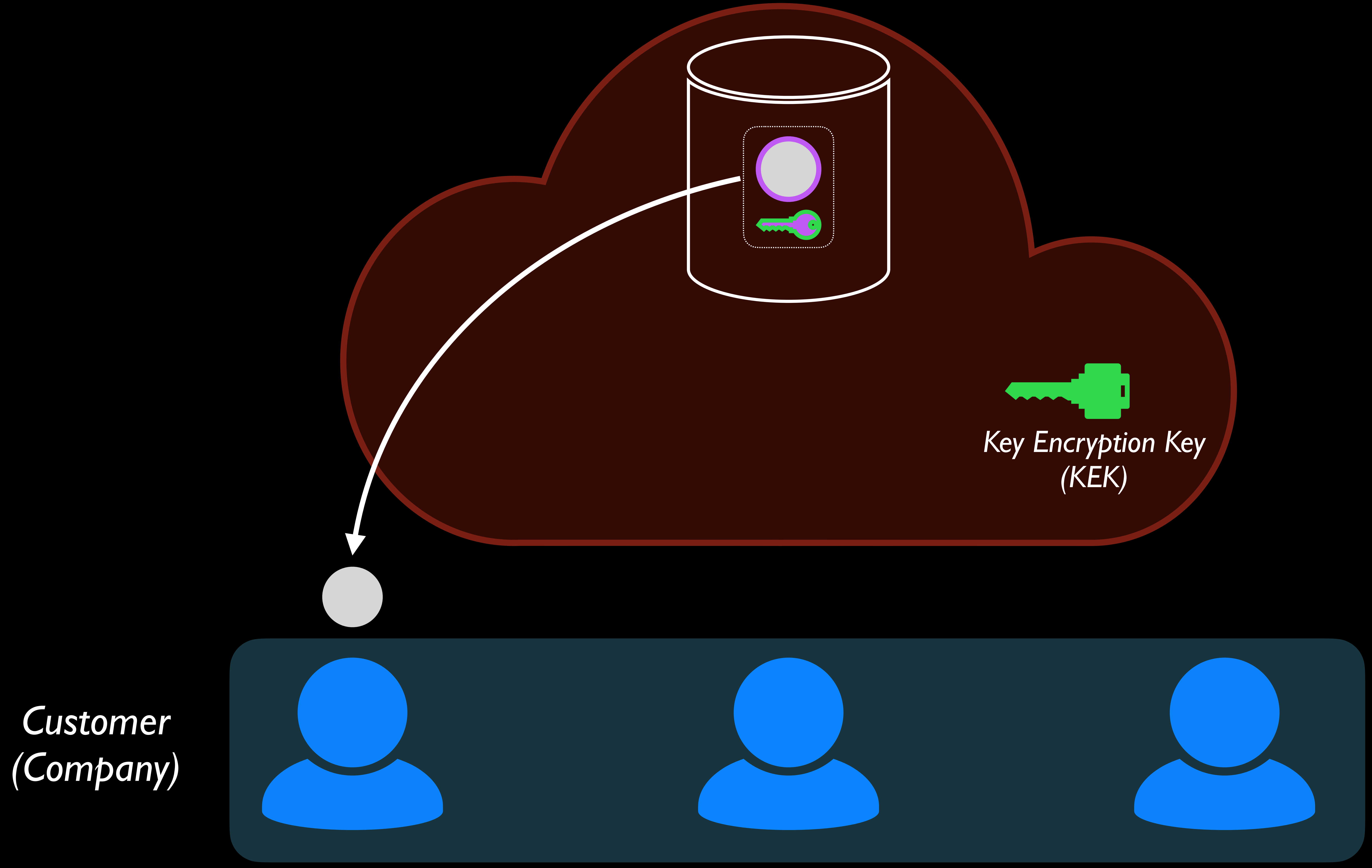


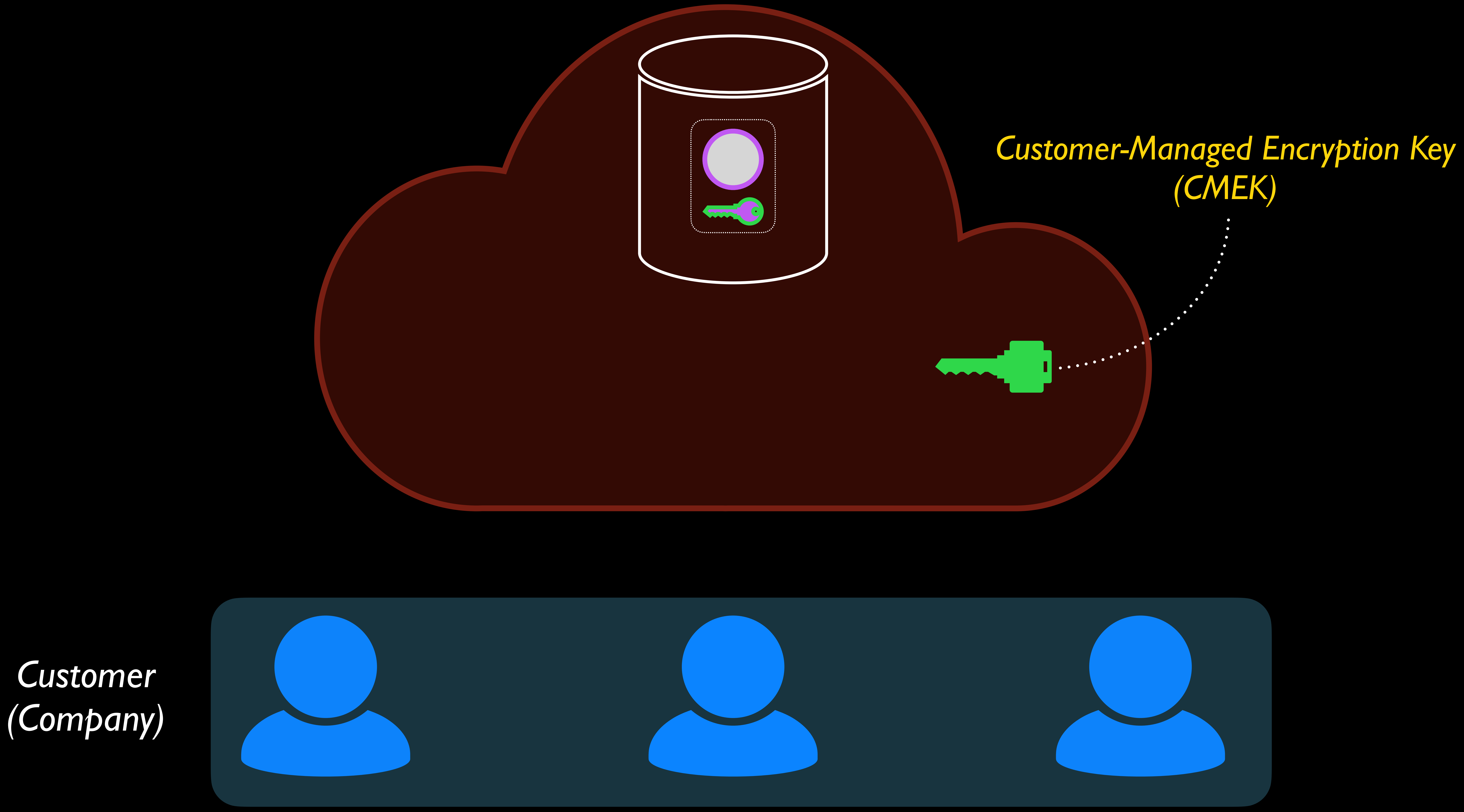
Customer
(Company)

Cloud-Side Encryption



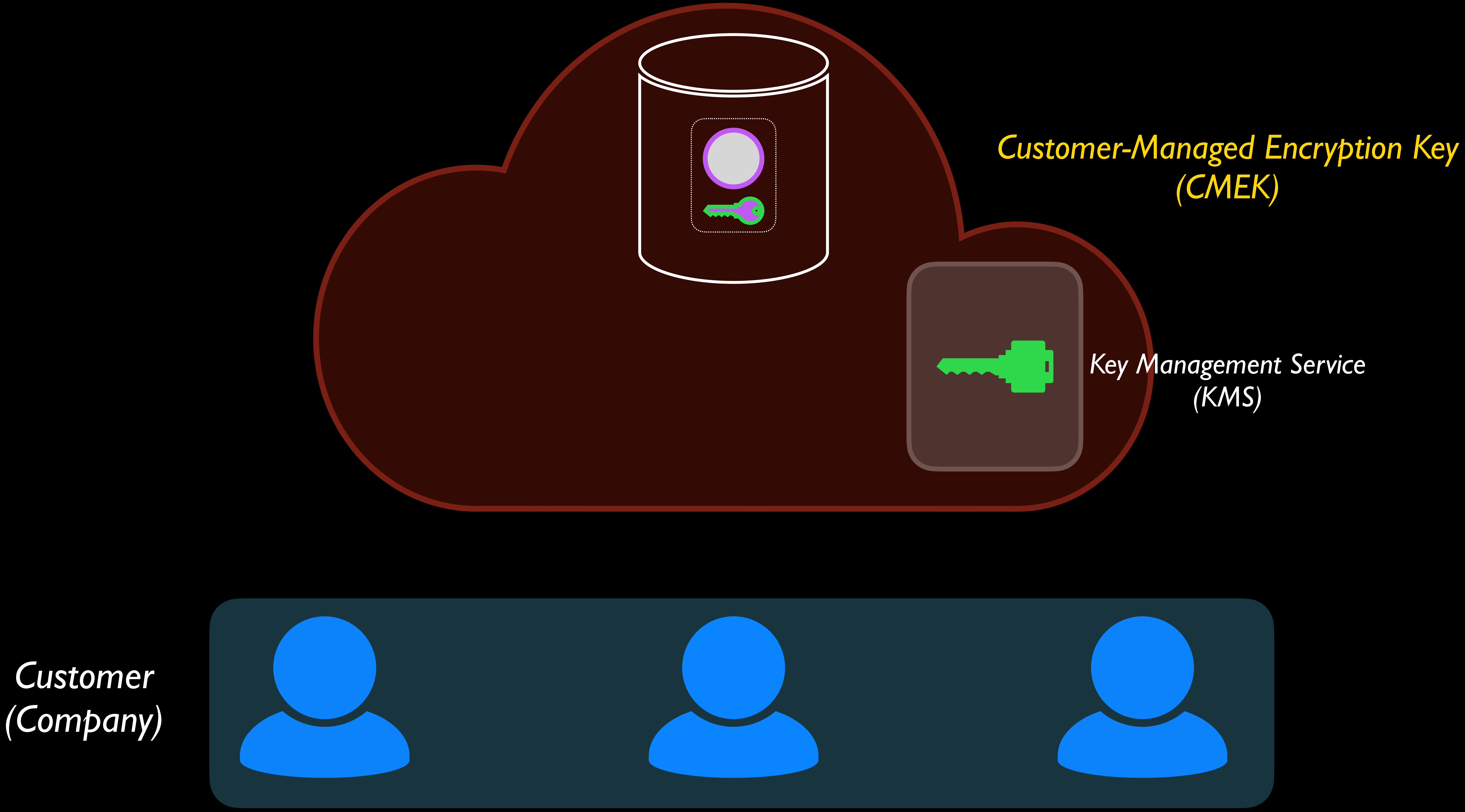
Cloud-Side Encryption

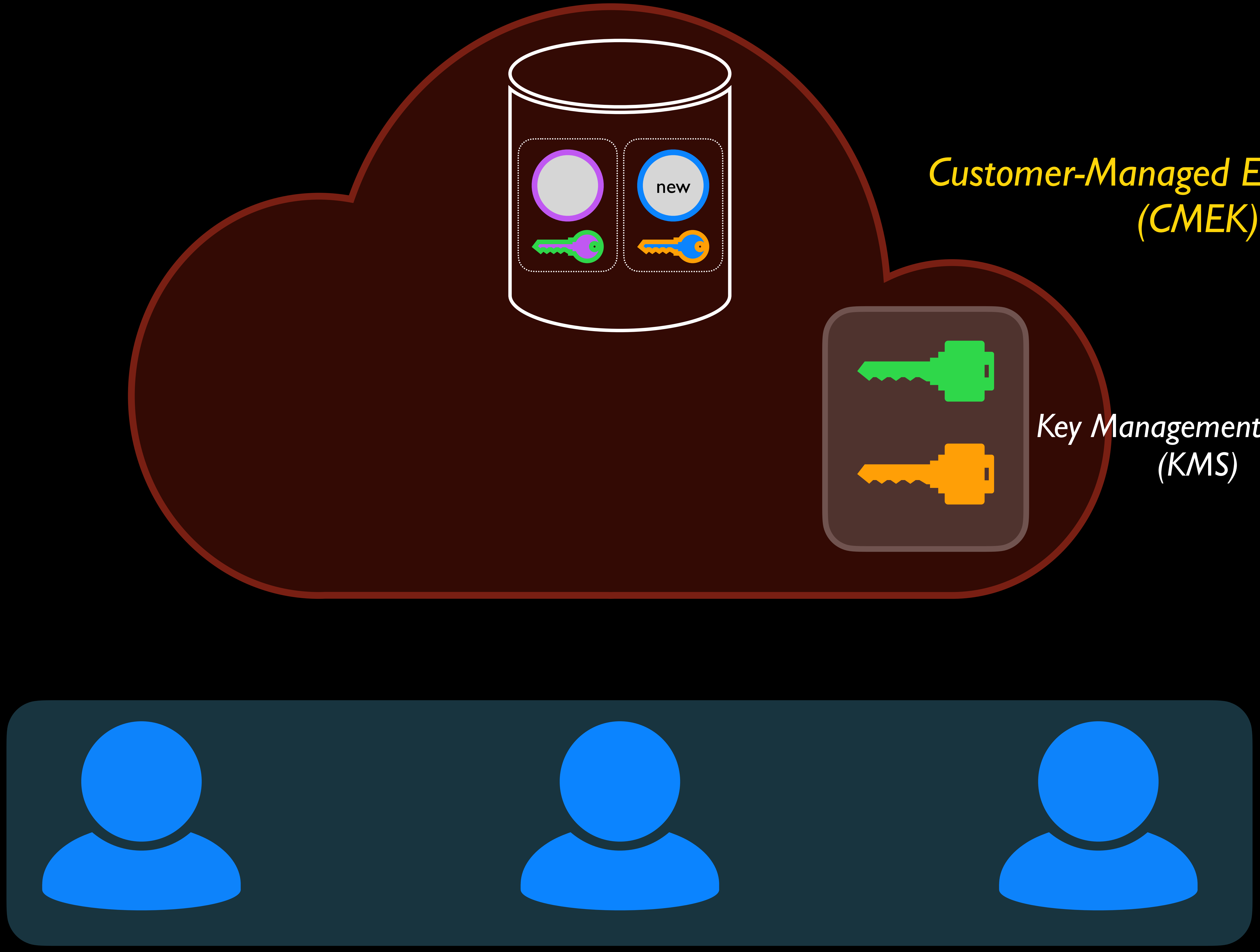




Customer-Managed Encryption Key (CMEK)

Customer (Company)

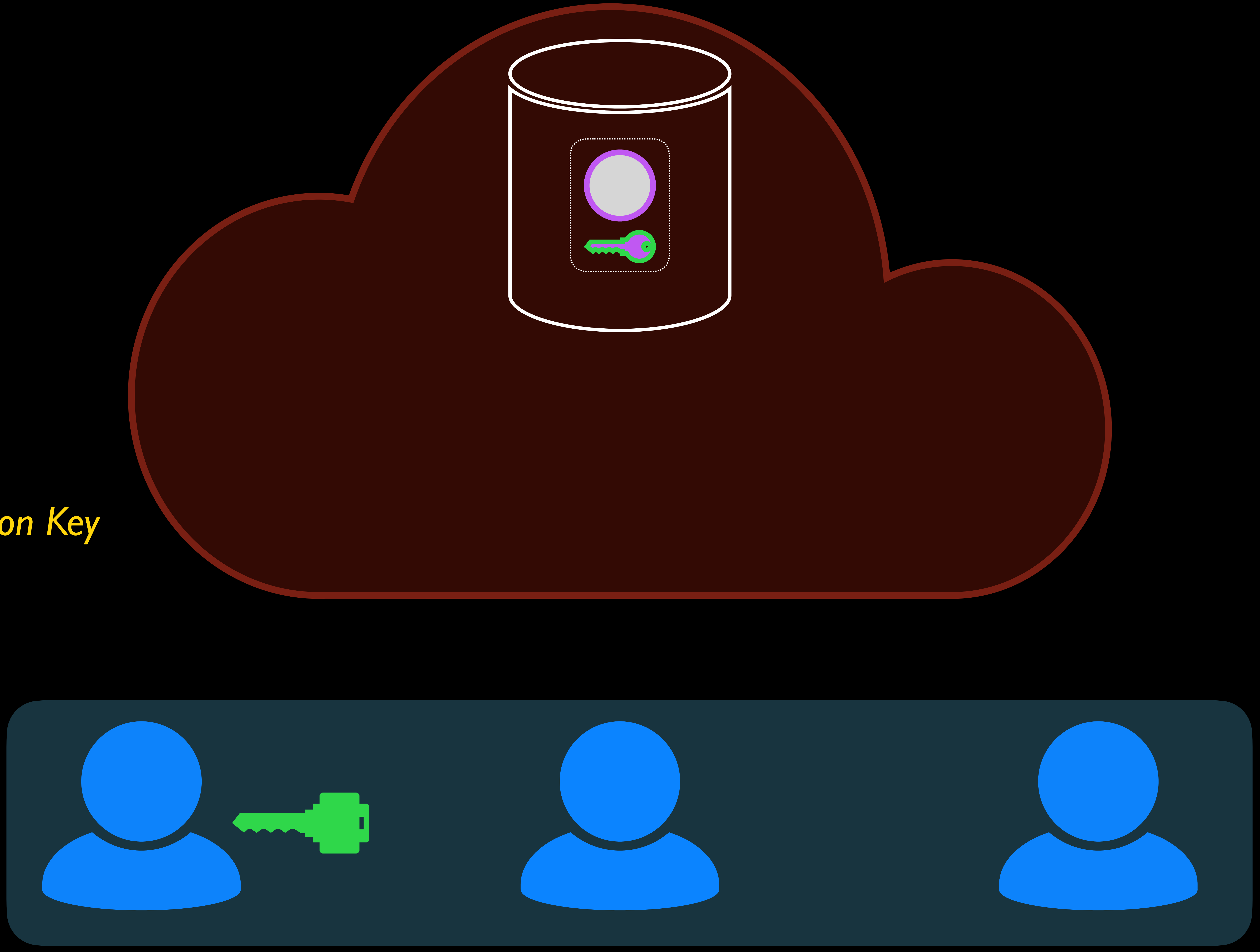

Customer-Managed Encryption Key
(CMEK)
Key Management Service
(KMS)
Customer
(Company)



Customer-Managed Encryption Key
(CMEK)

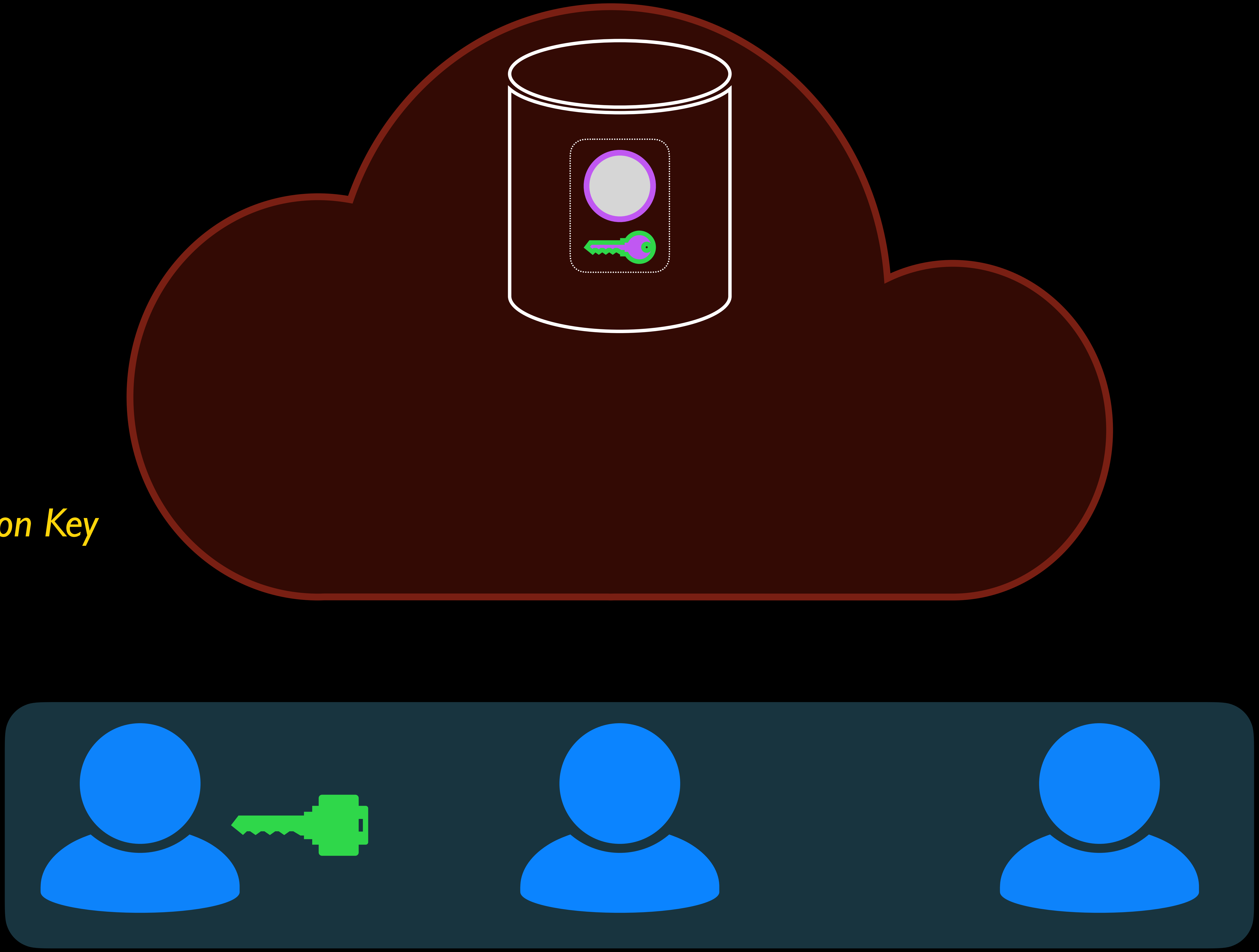
Key Management Service
(KMS)

Customer
(Company)



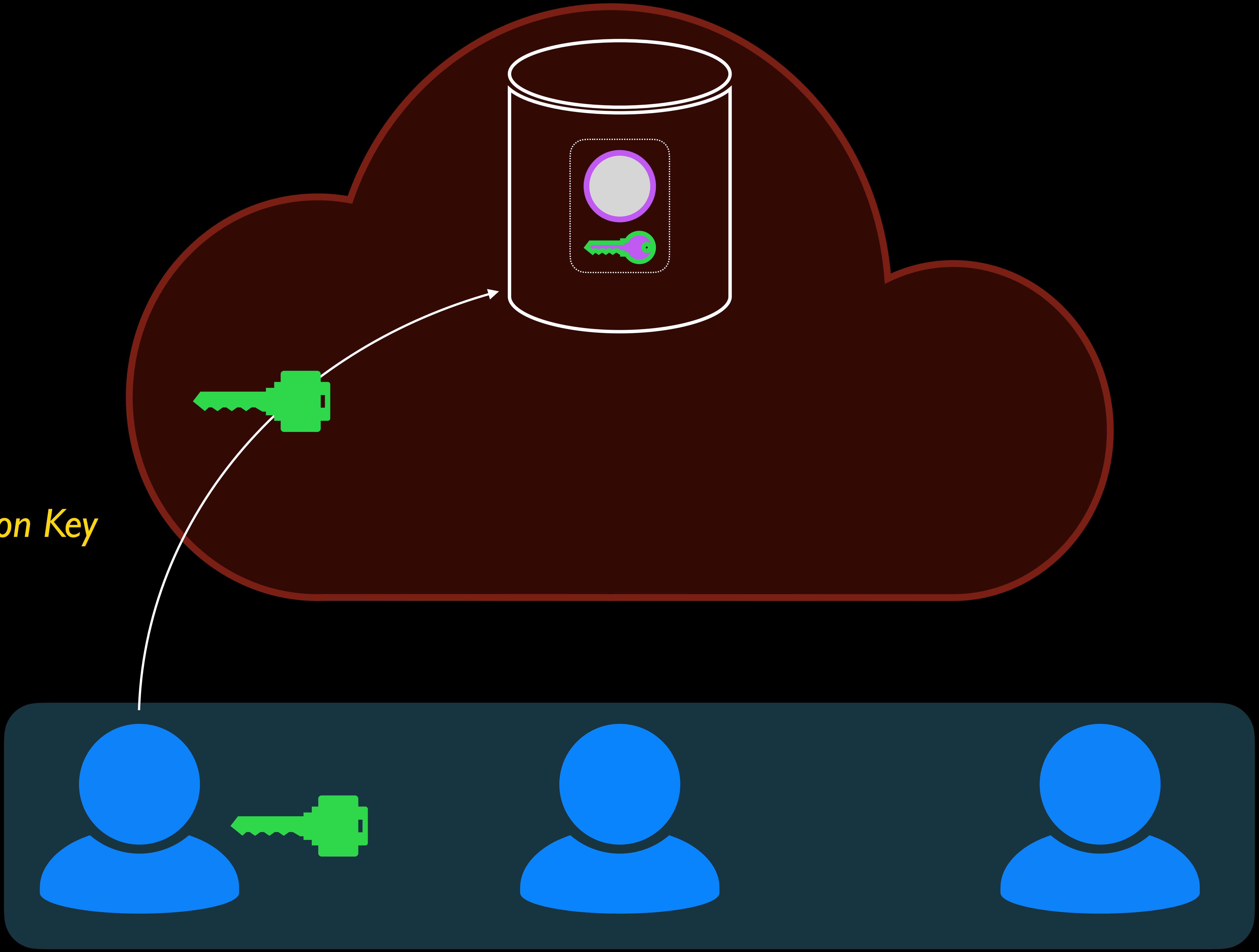
Customer-Supplied Encryption Key
(CSEK)

Customer
(Company)



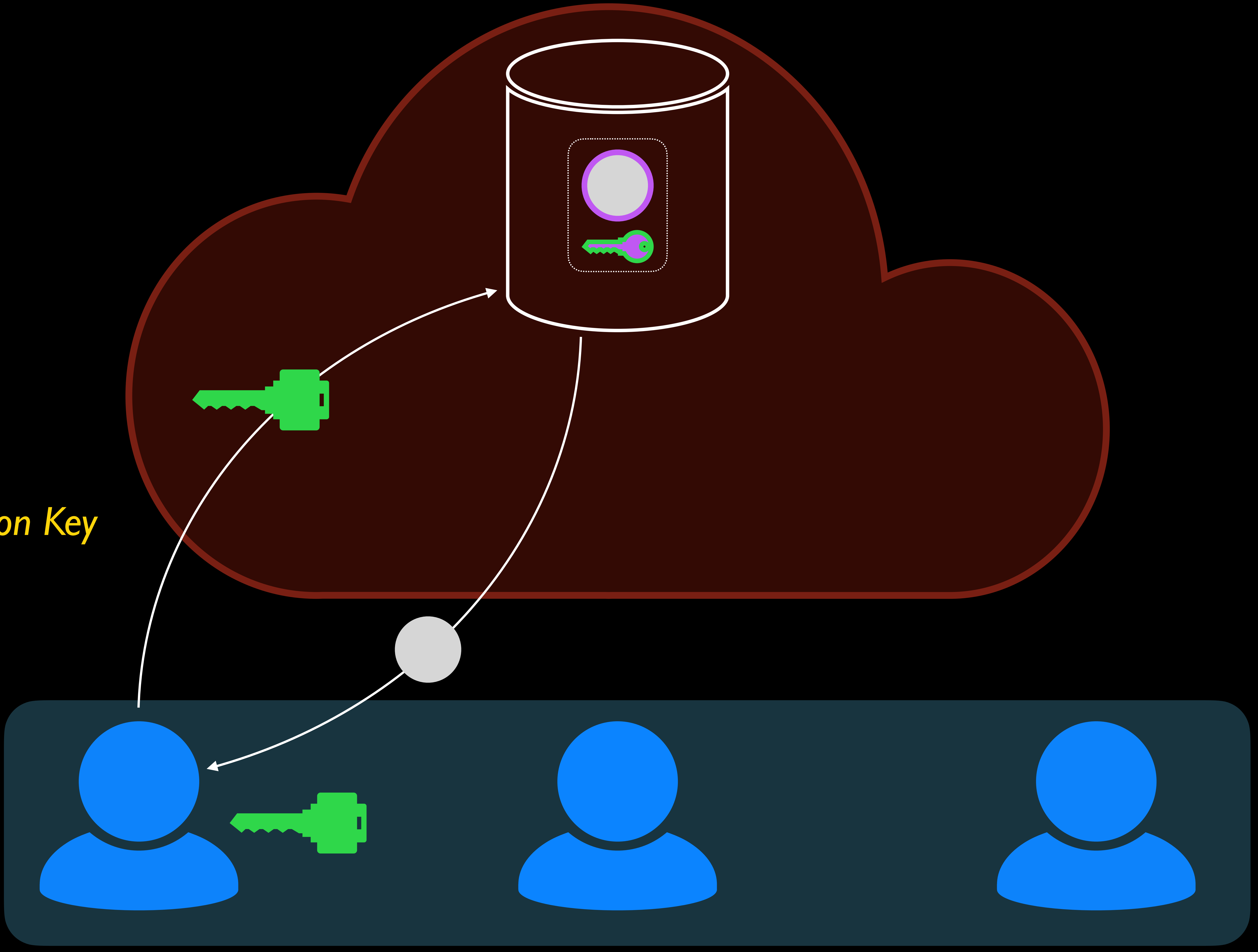
Customer-Supplied Encryption Key
(CSEK)

Customer
(Company)



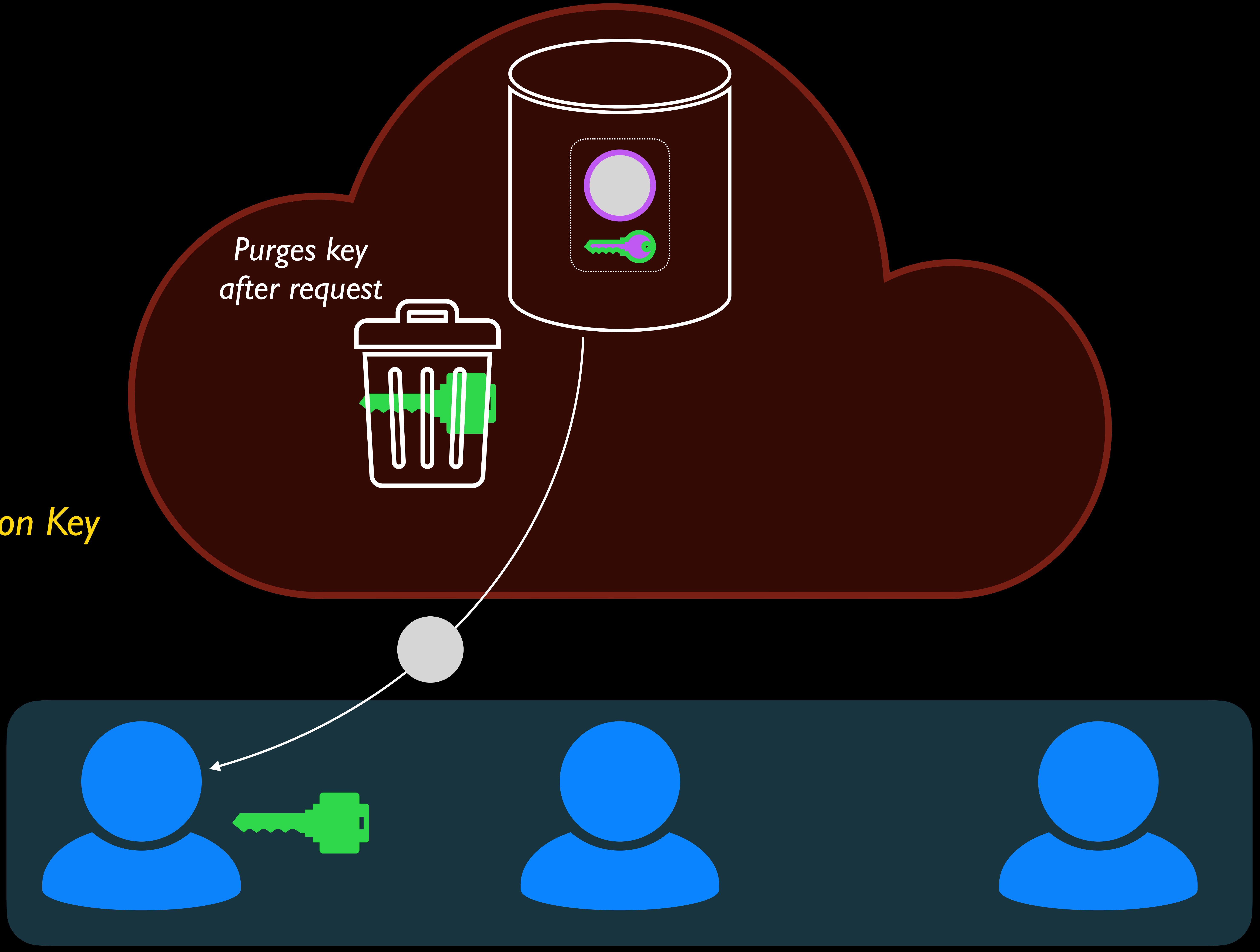
Customer-Supplied Encryption Key (CSEK)

Customer (Company)



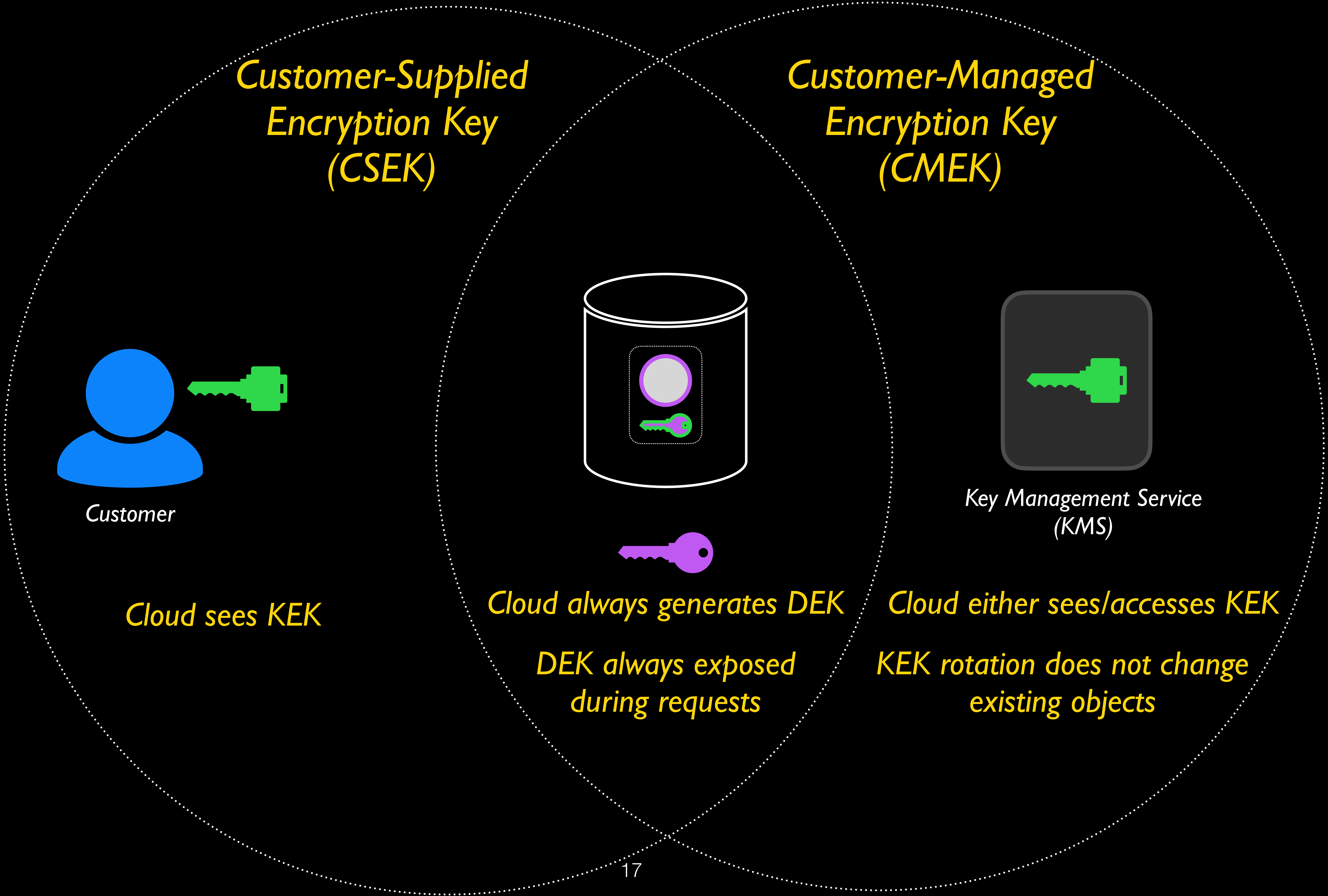
Customer-Supplied Encryption Key
(CSEK)

Customer
(Company)



Customer-Supplied Encryption Key
(CSEK)

Customer
(Company)



Customer-Supplied Encryption Key (CSEK)

Customer-Managed Encryption Key (CMEK)

Cloud sees KEK

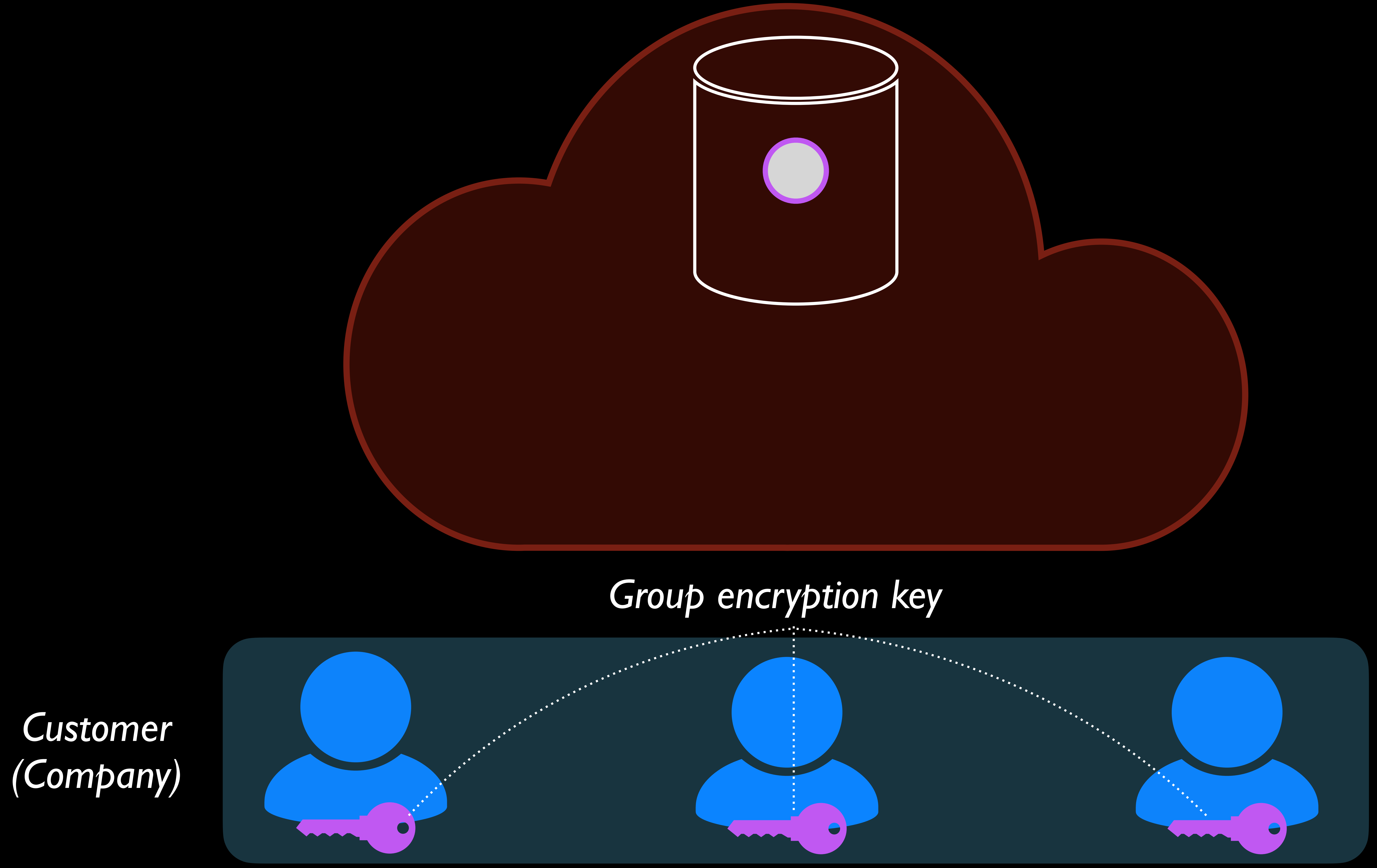
Cloud always generates DEK

DEK always exposed during requests

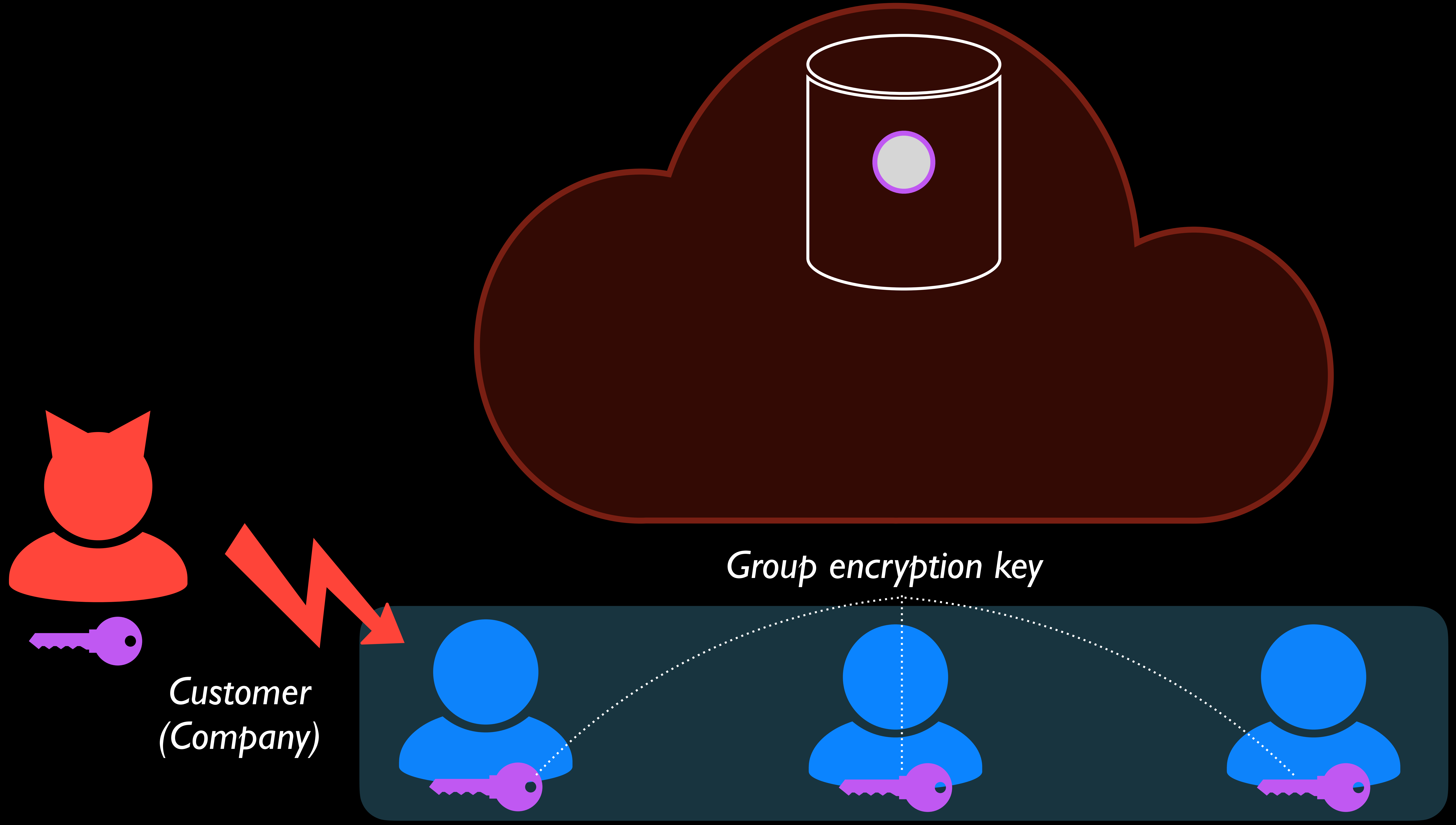
Cloud either sees/accesses KEK

KEK rotation does not change existing objects

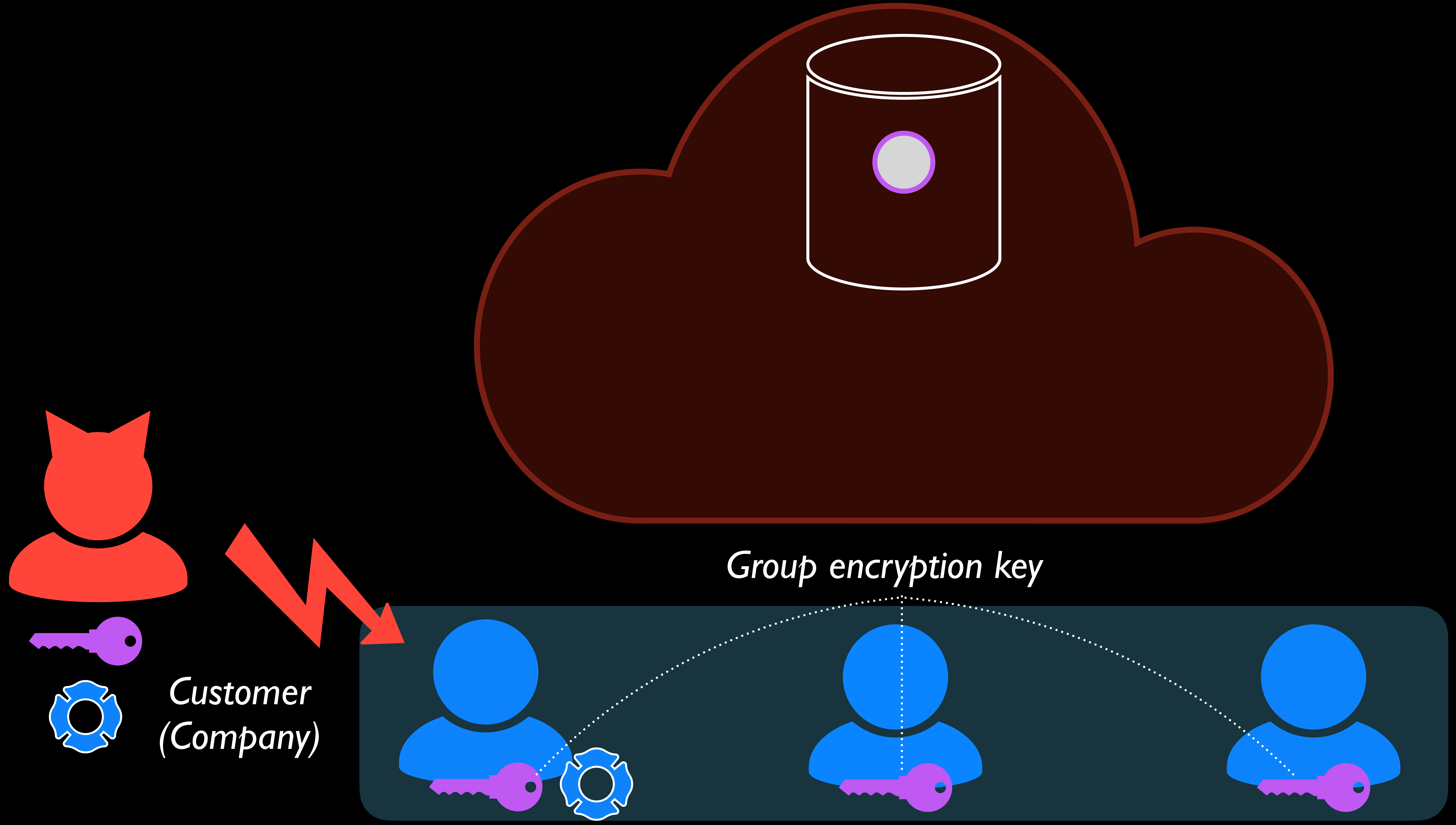
Client-Side Encryption



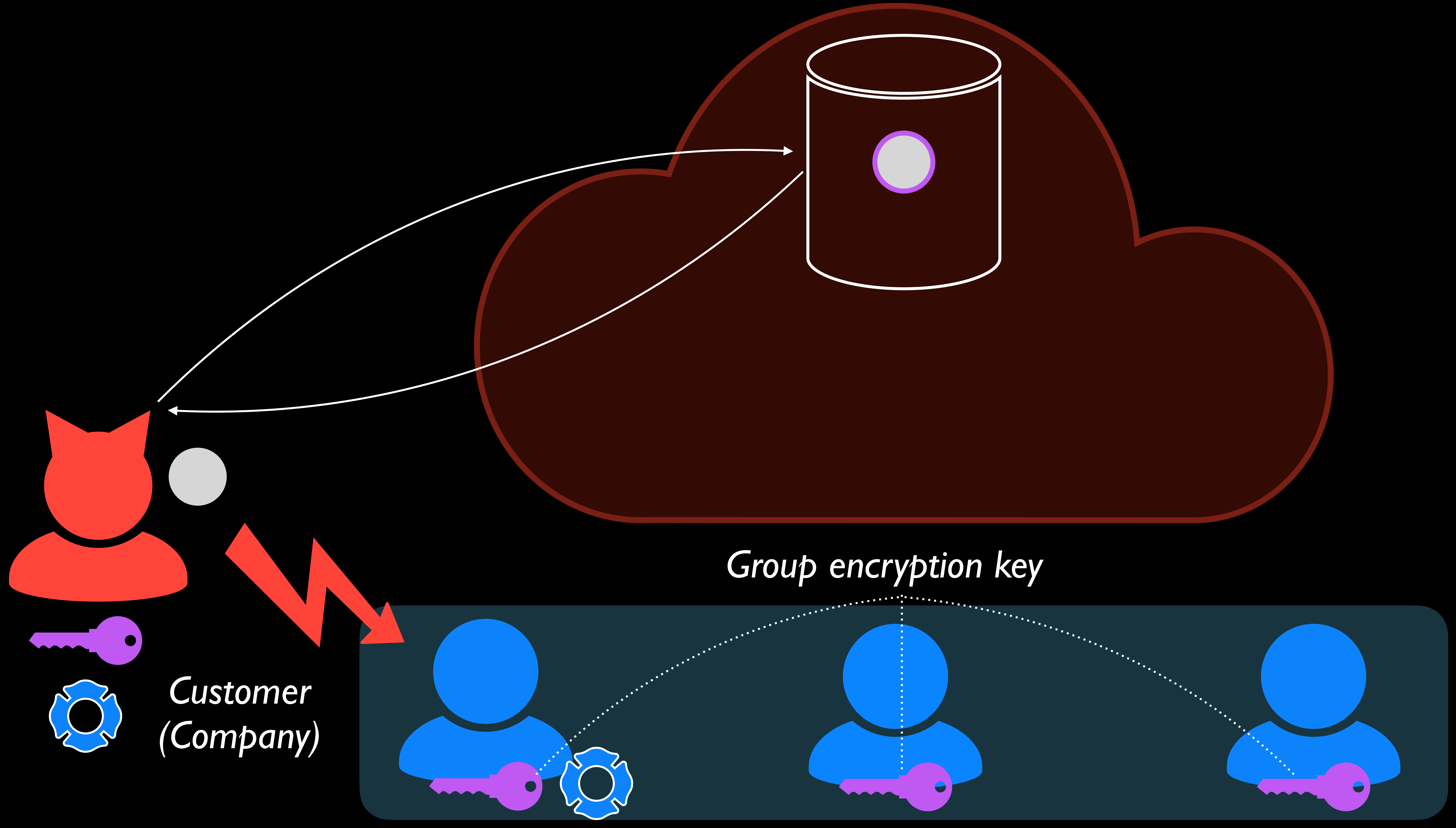
Client-Side Encryption



Client-Side Encryption

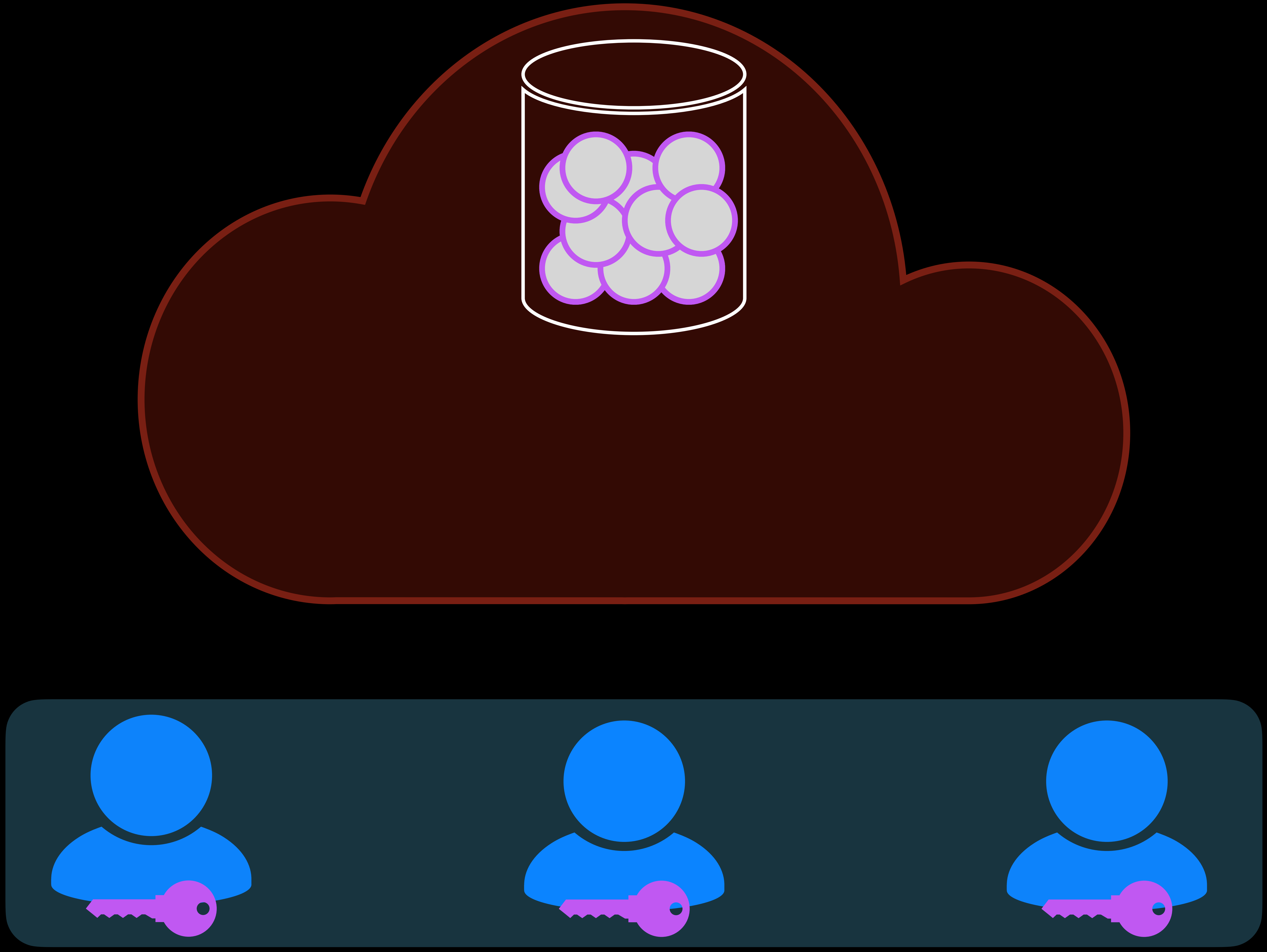


Client-Side Encryption



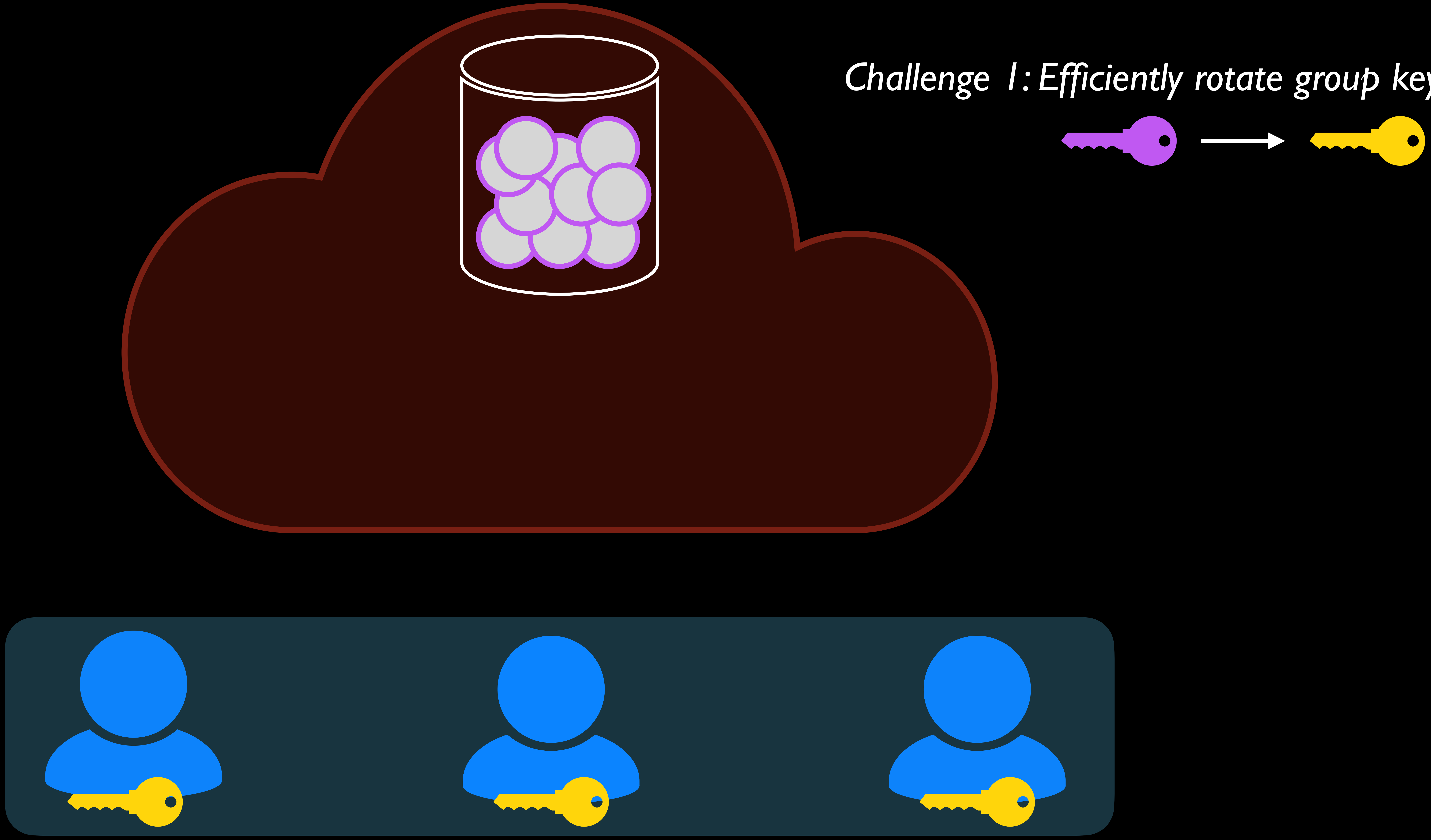
Cloud credential

Client-Side Encryption



Customer (Company)

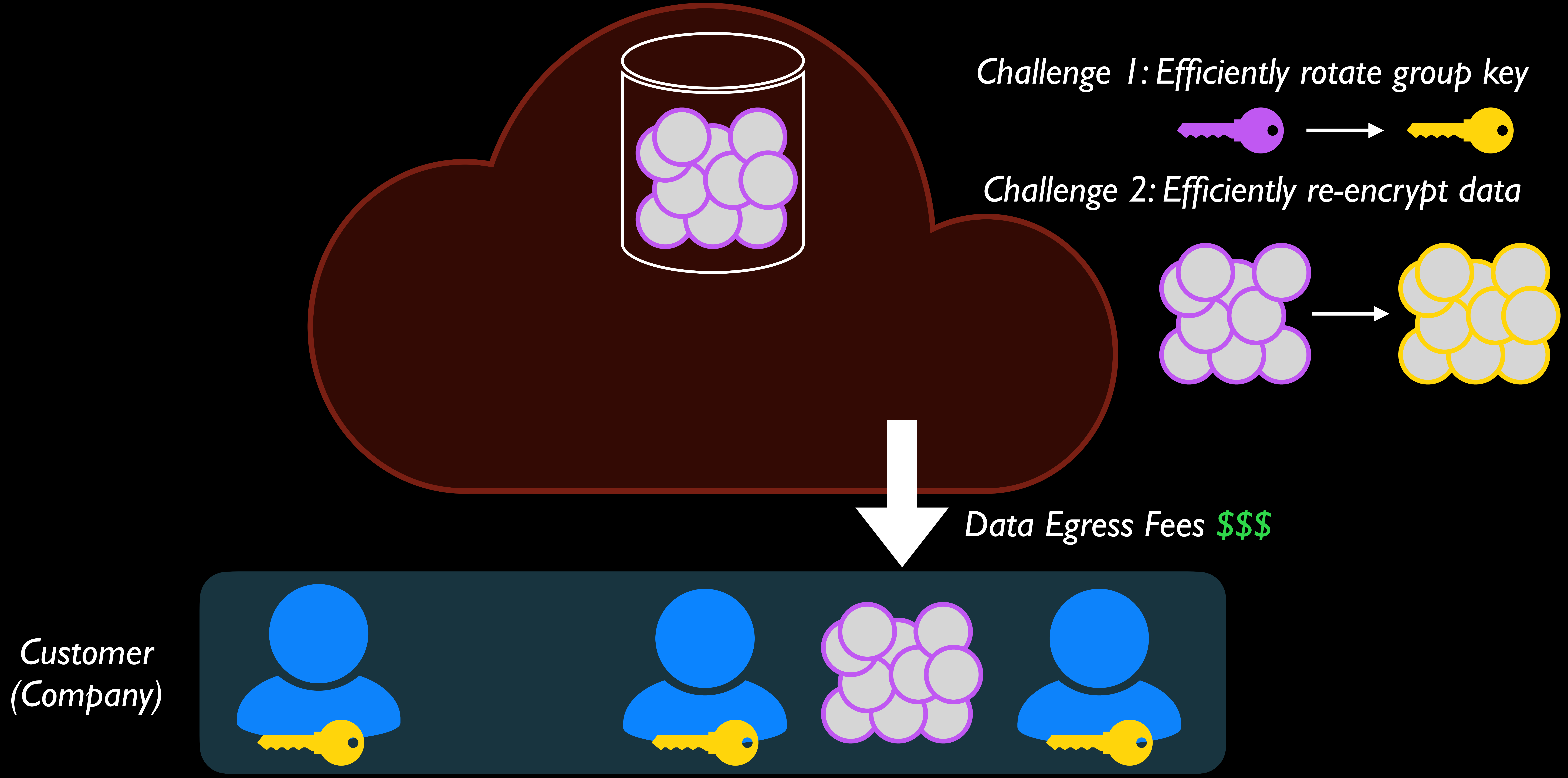
Client-Side Encryption



Challenge 1: Efficiently rotate group key

Customer (Company)

Client-Side Encryption



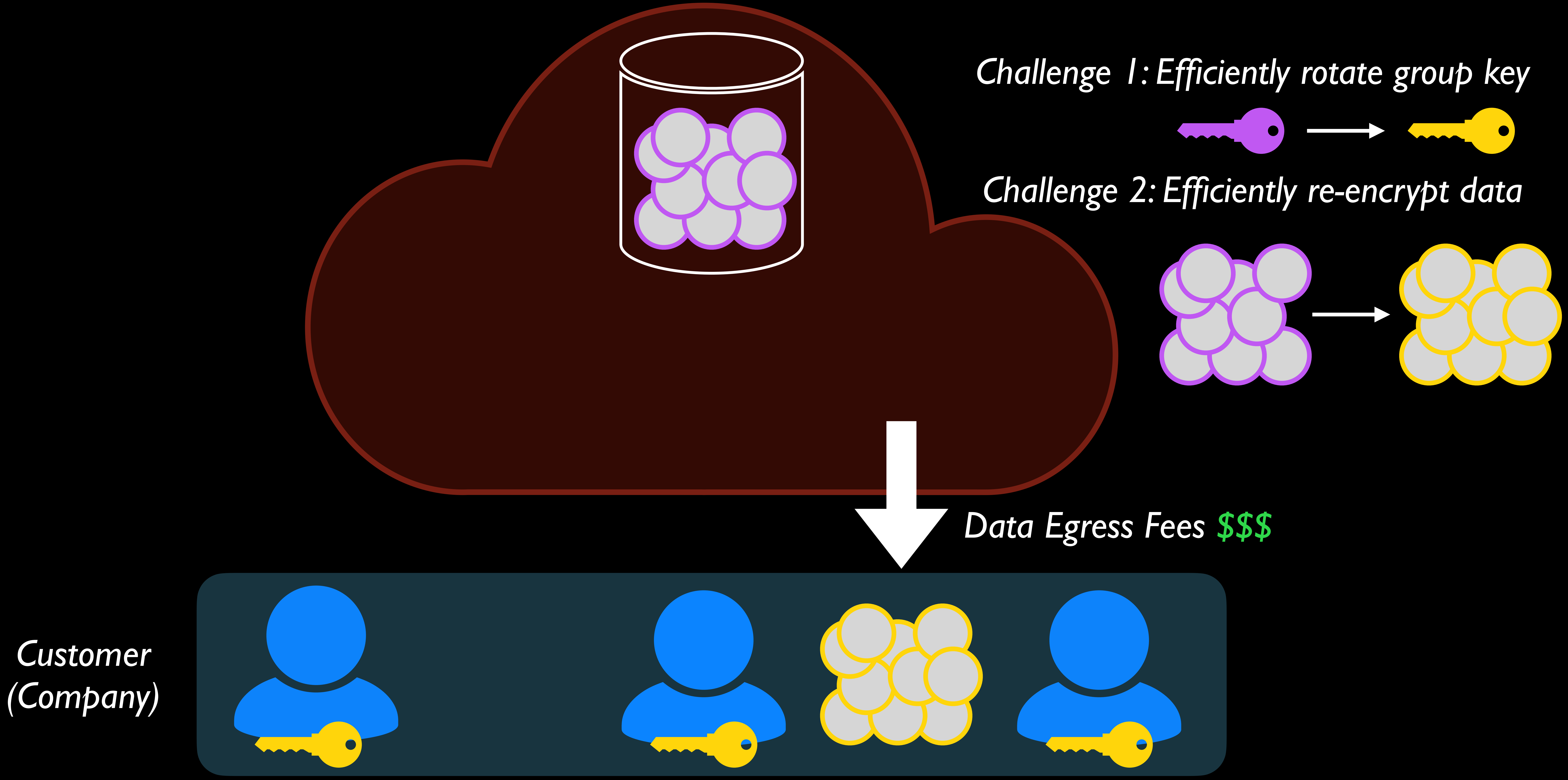
Challenge 1: Efficiently rotate group key

Challenge 2: Efficiently re-encrypt data

Data Egress Fees $$$

Customer (Company)

Client-Side Encryption



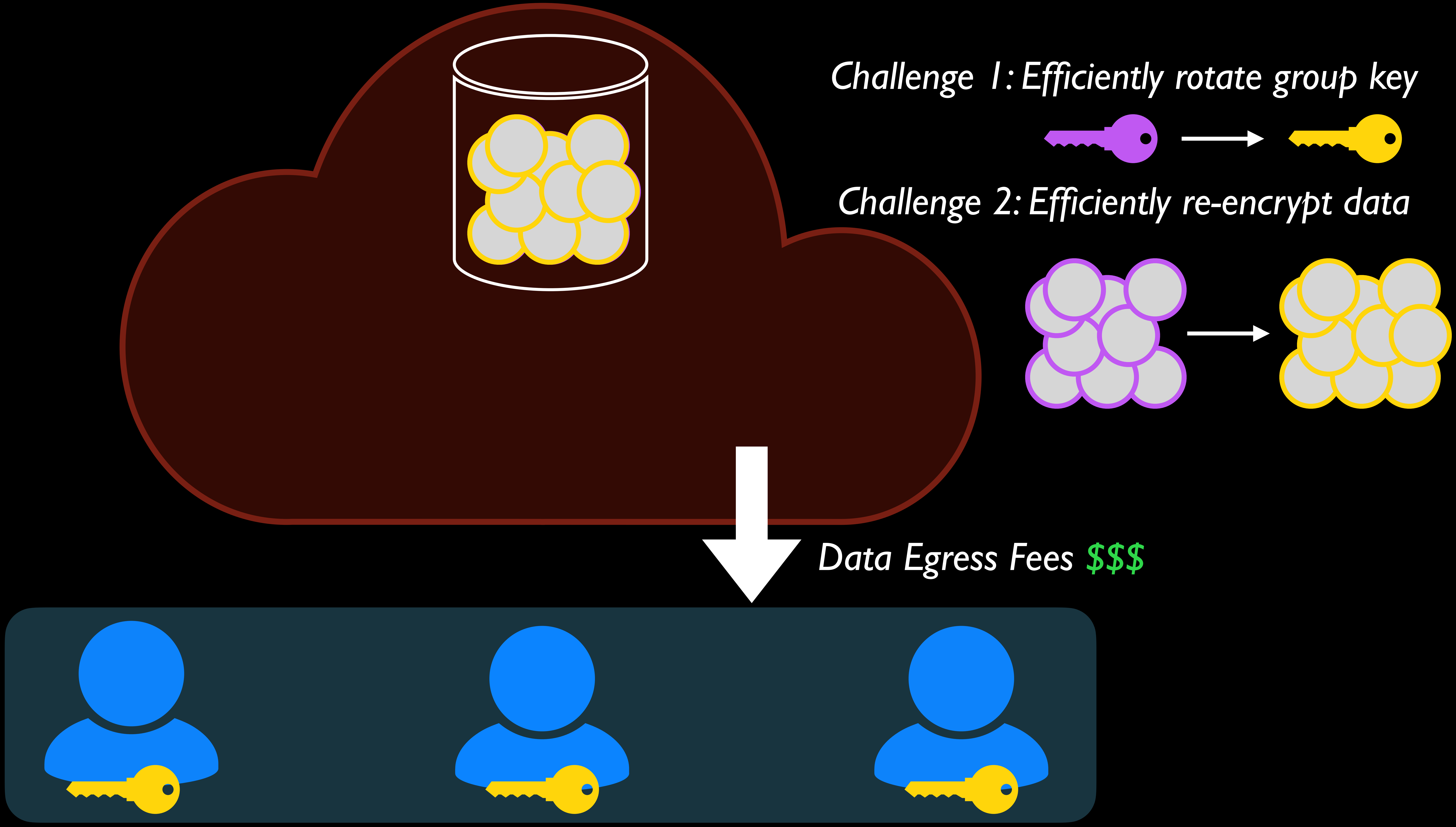
Challenge 1: Efficiently rotate group key

Challenge 2: Efficiently re-encrypt data

Data Egress Fees $$$

Customer (Company)

Client-Side Encryption



Challenge 1: Efficiently rotate group key

Challenge 2: Efficiently re-encrypt data

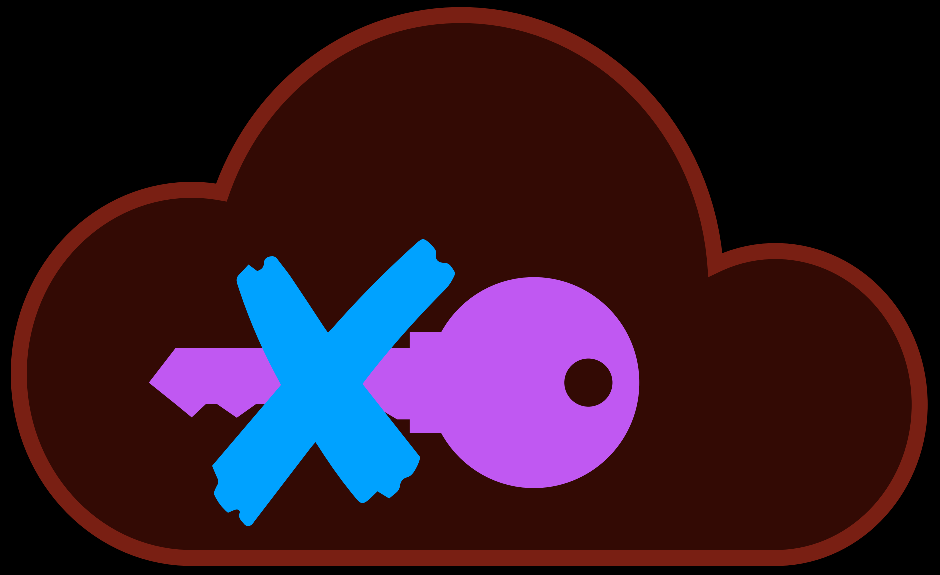
Data Egress Fees $$$

Customer (Company)

Requirements

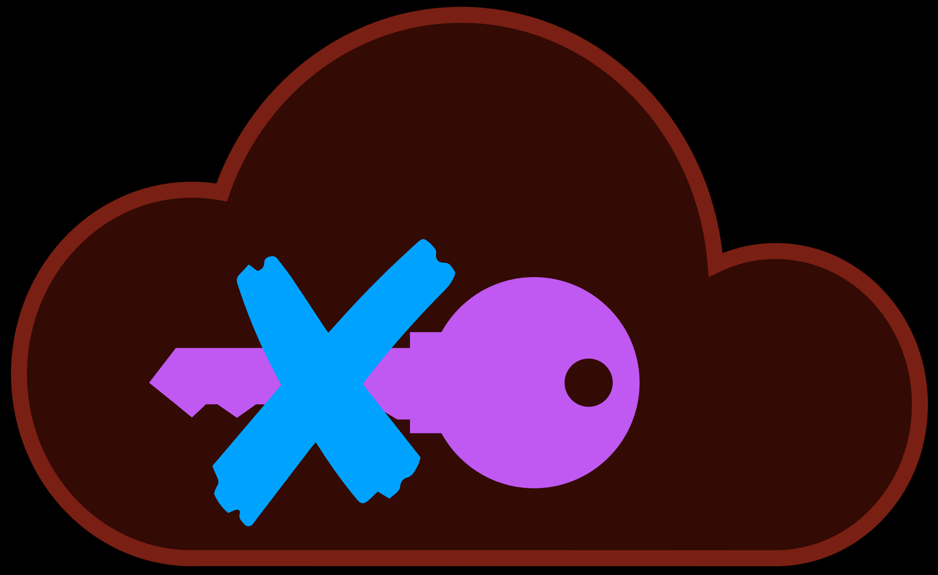
Requirements

1 The cloud shouldn’t have access to keys

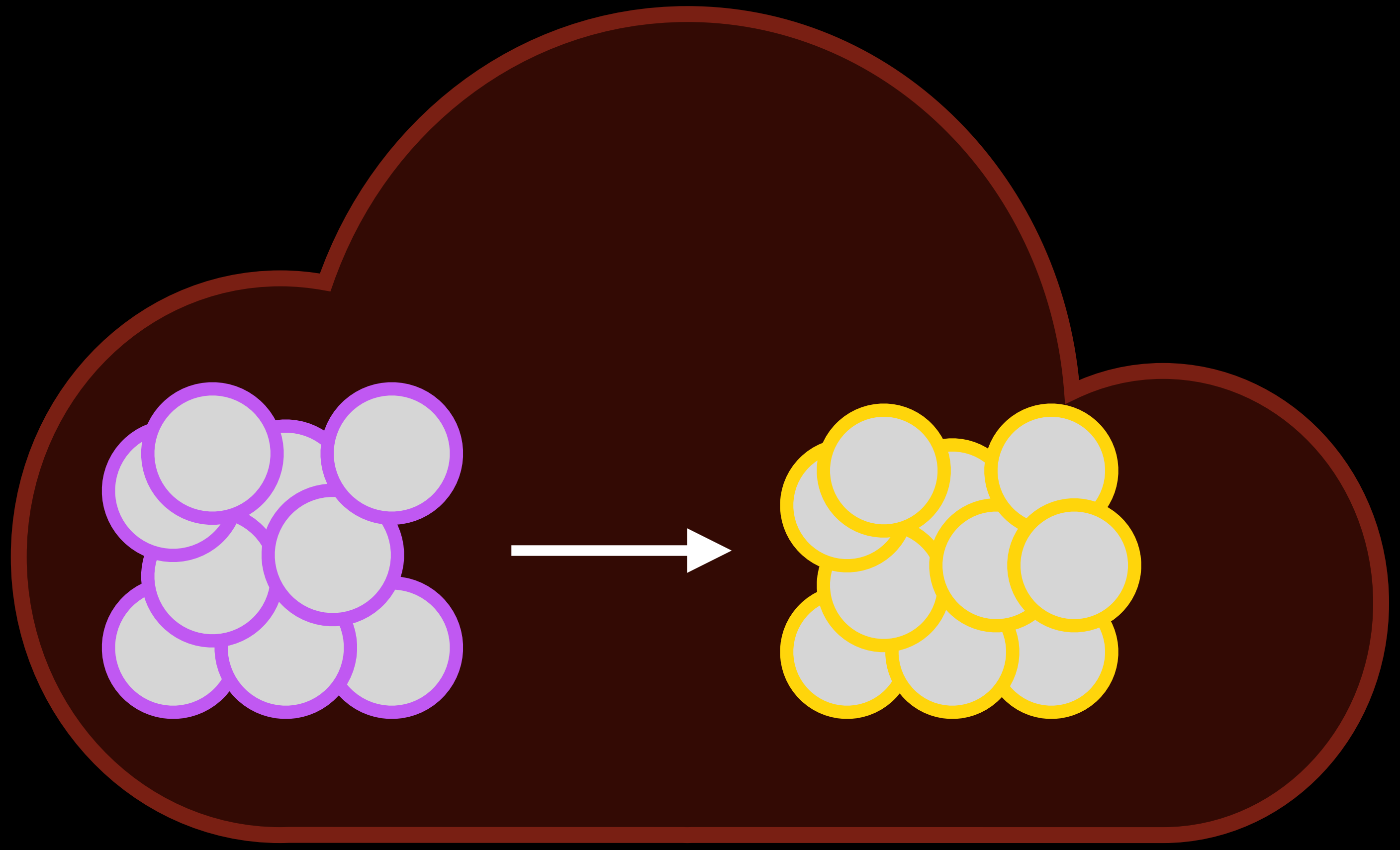


Requirements

1 The cloud shouldn't have access to keys

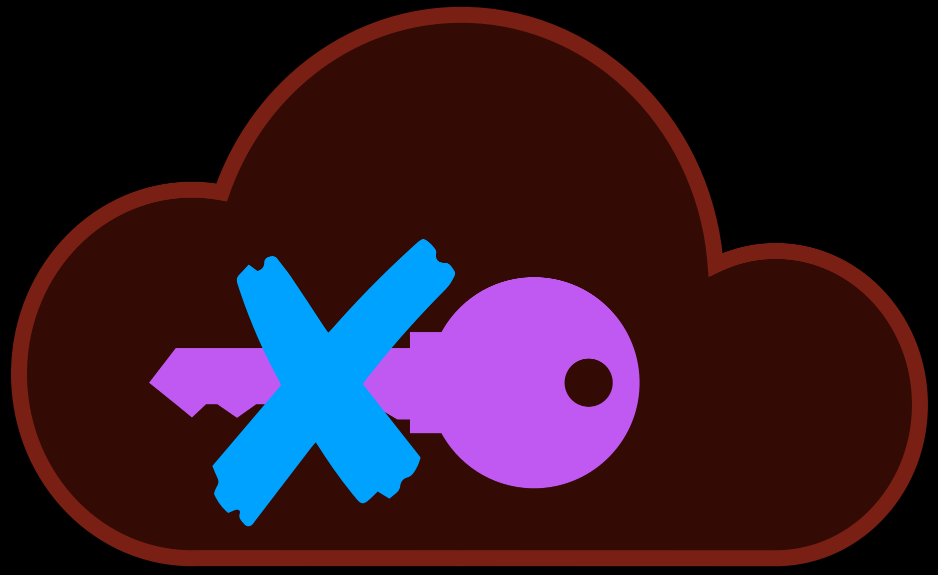


2 Data must be re-encrypted in the cloud

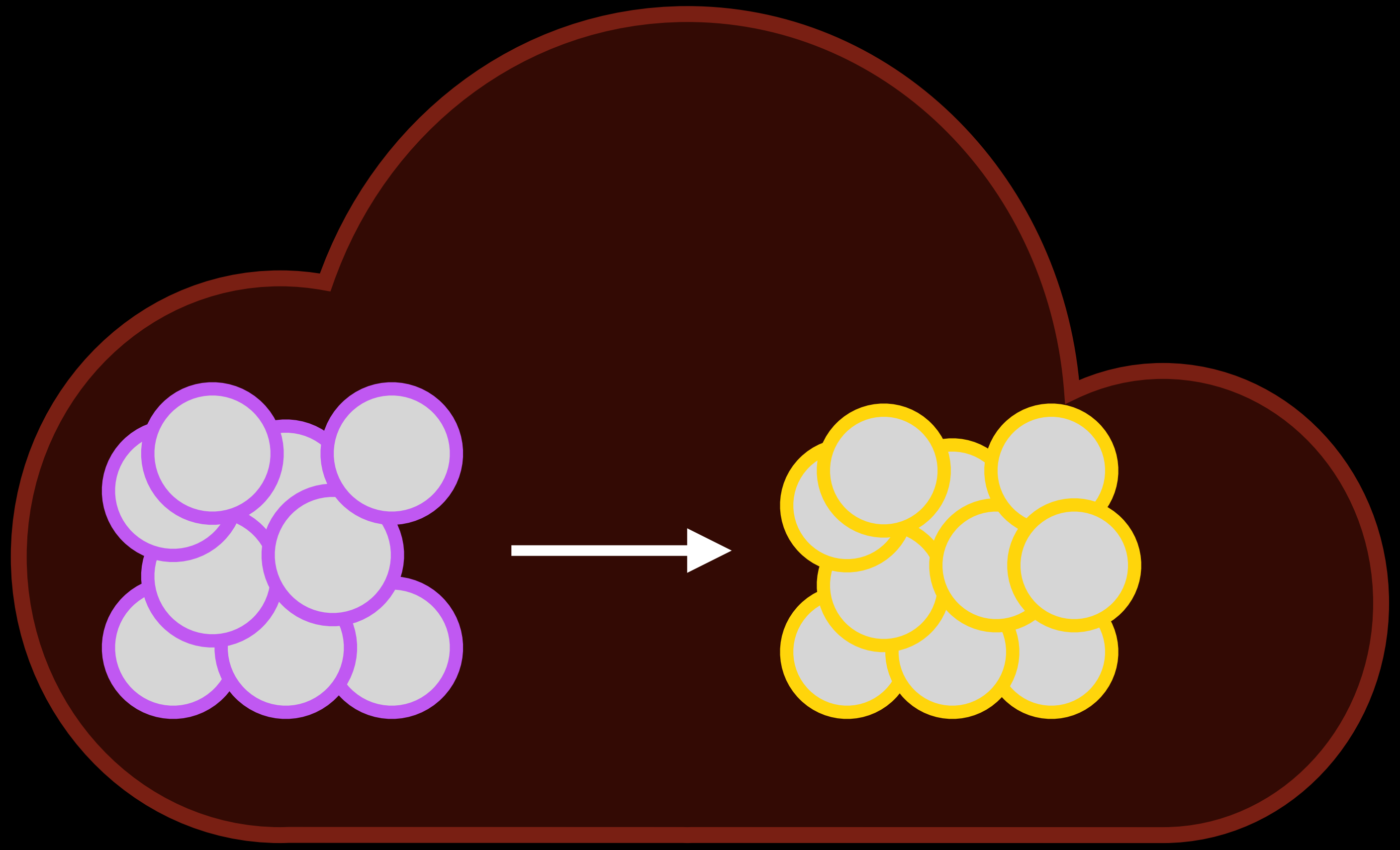


Requirements

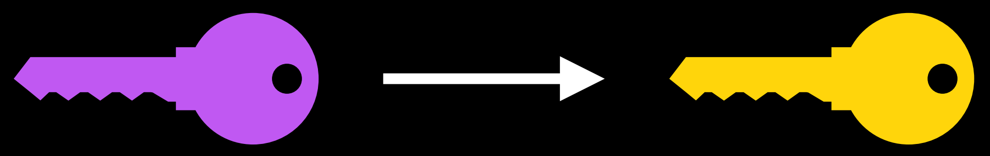
1 The cloud shouldn’t have access to keys



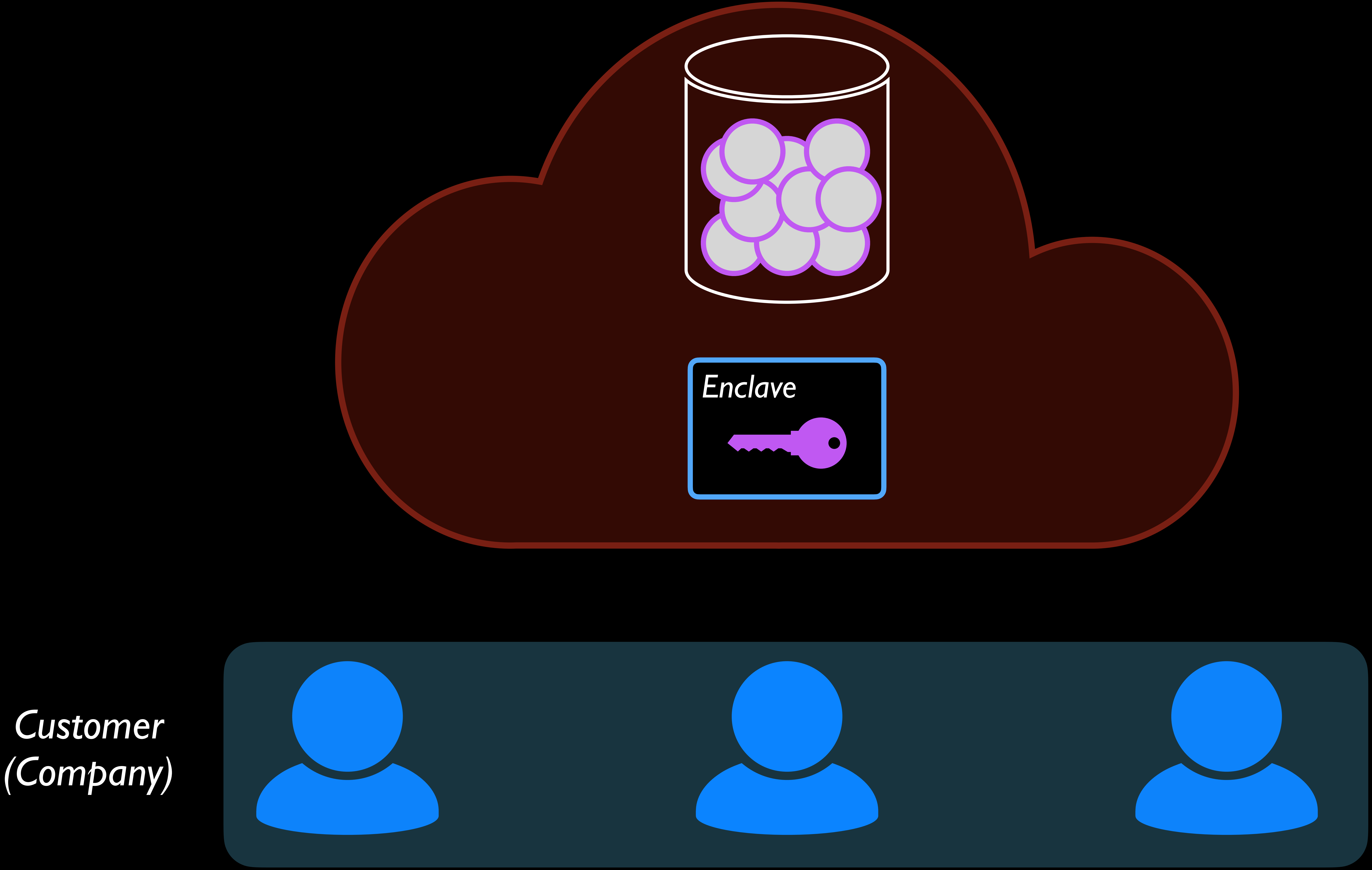
2 Data must be re-encrypted in the cloud



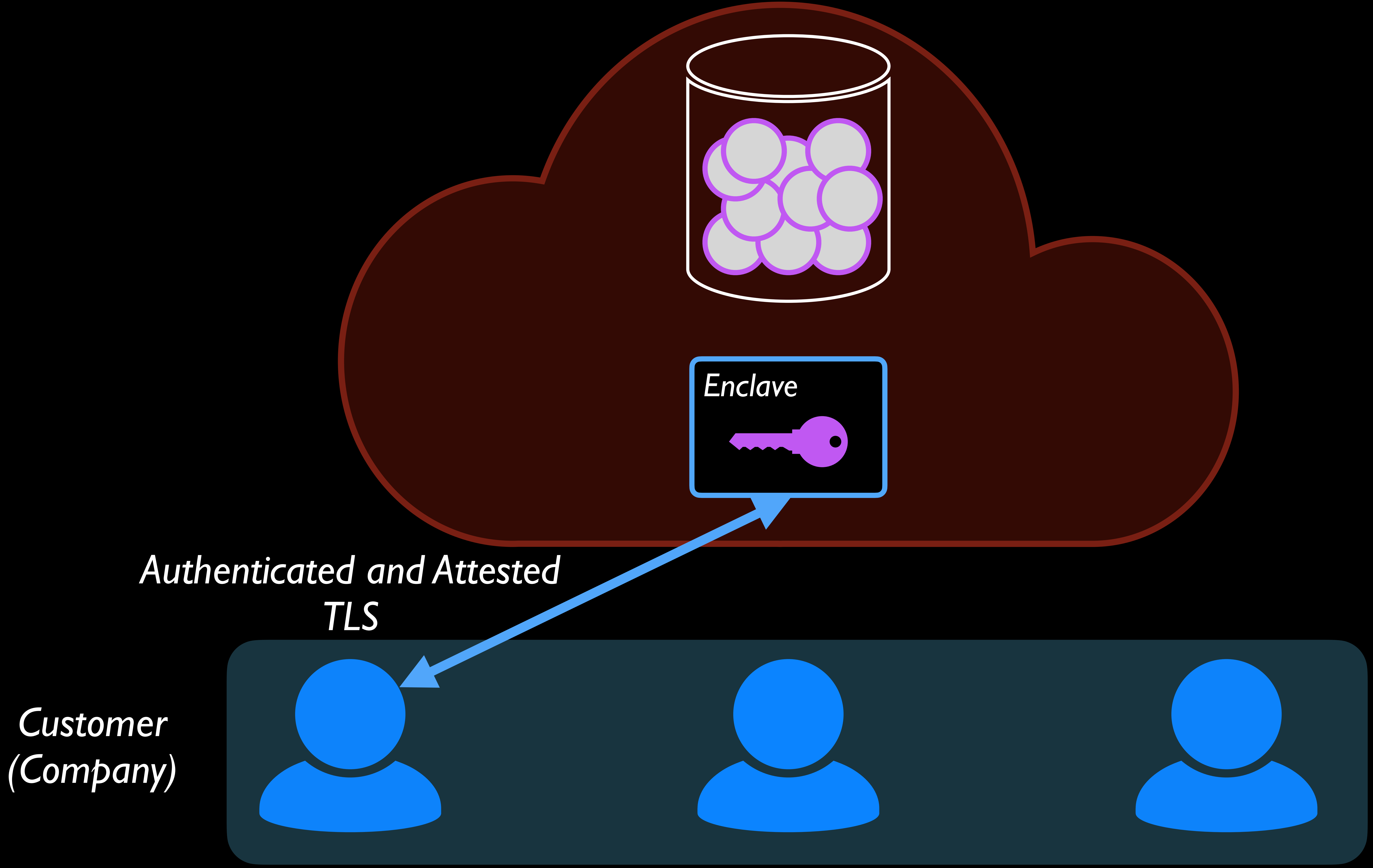
3 Rotating a key must be cheap



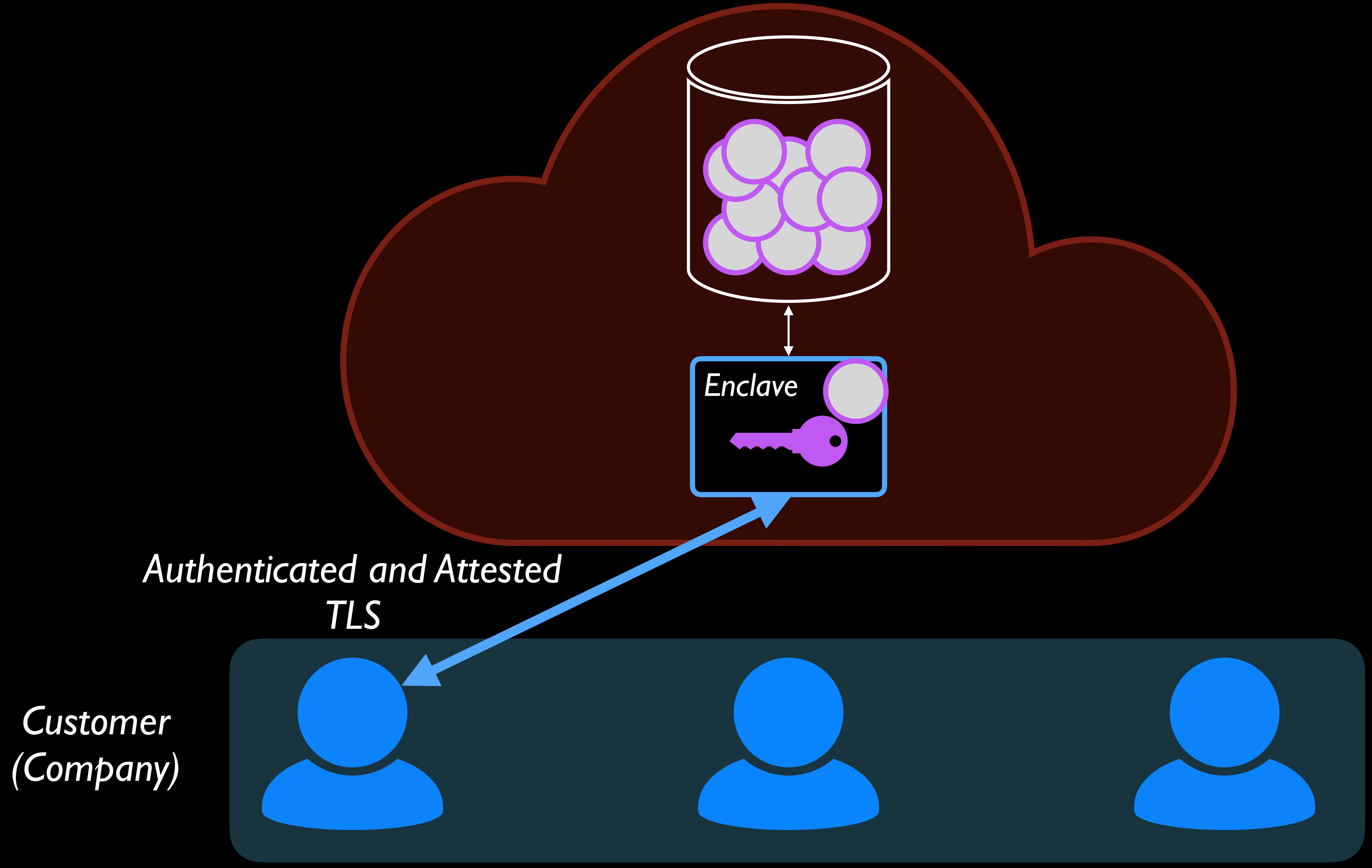
Enclave Strawman



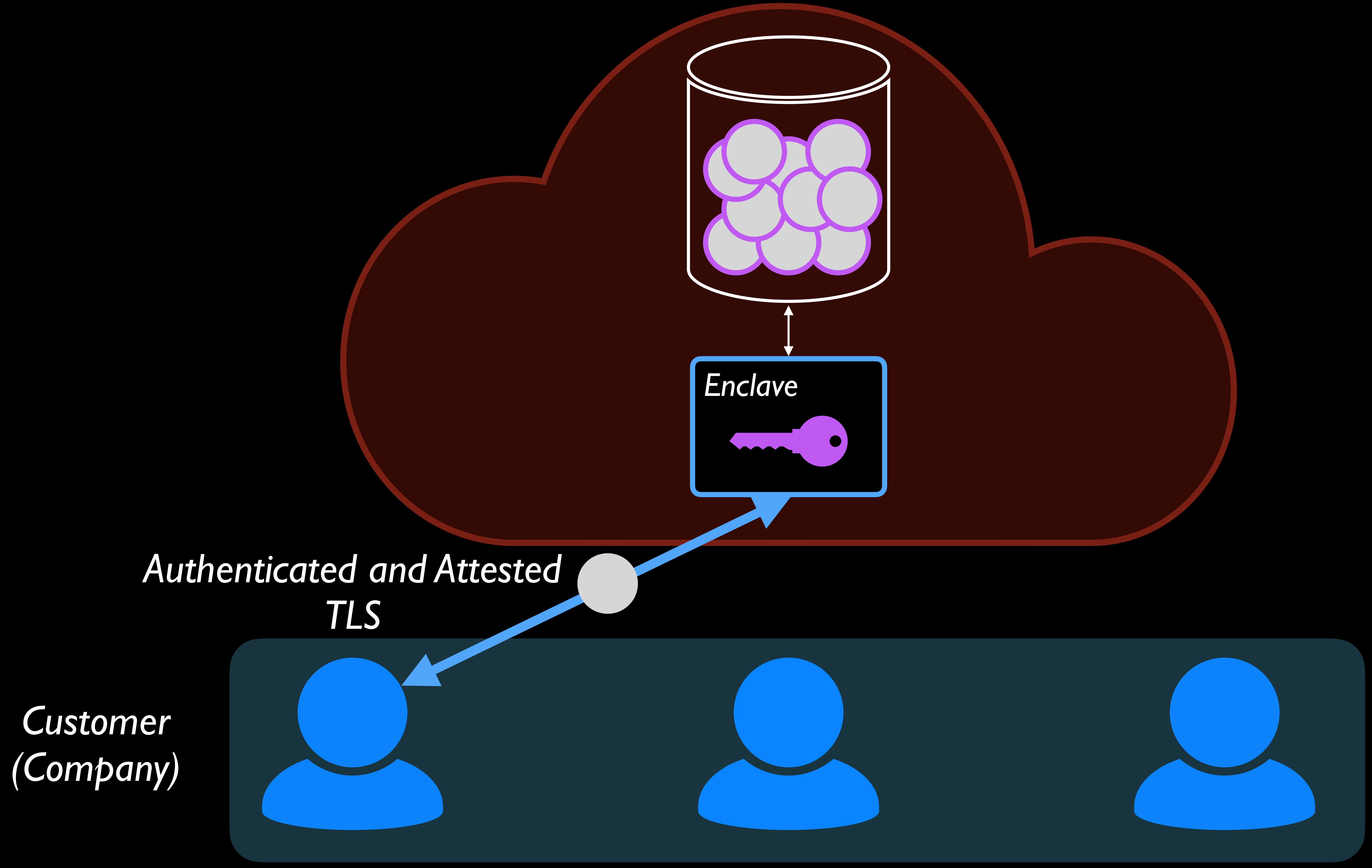
Enclave Strawman



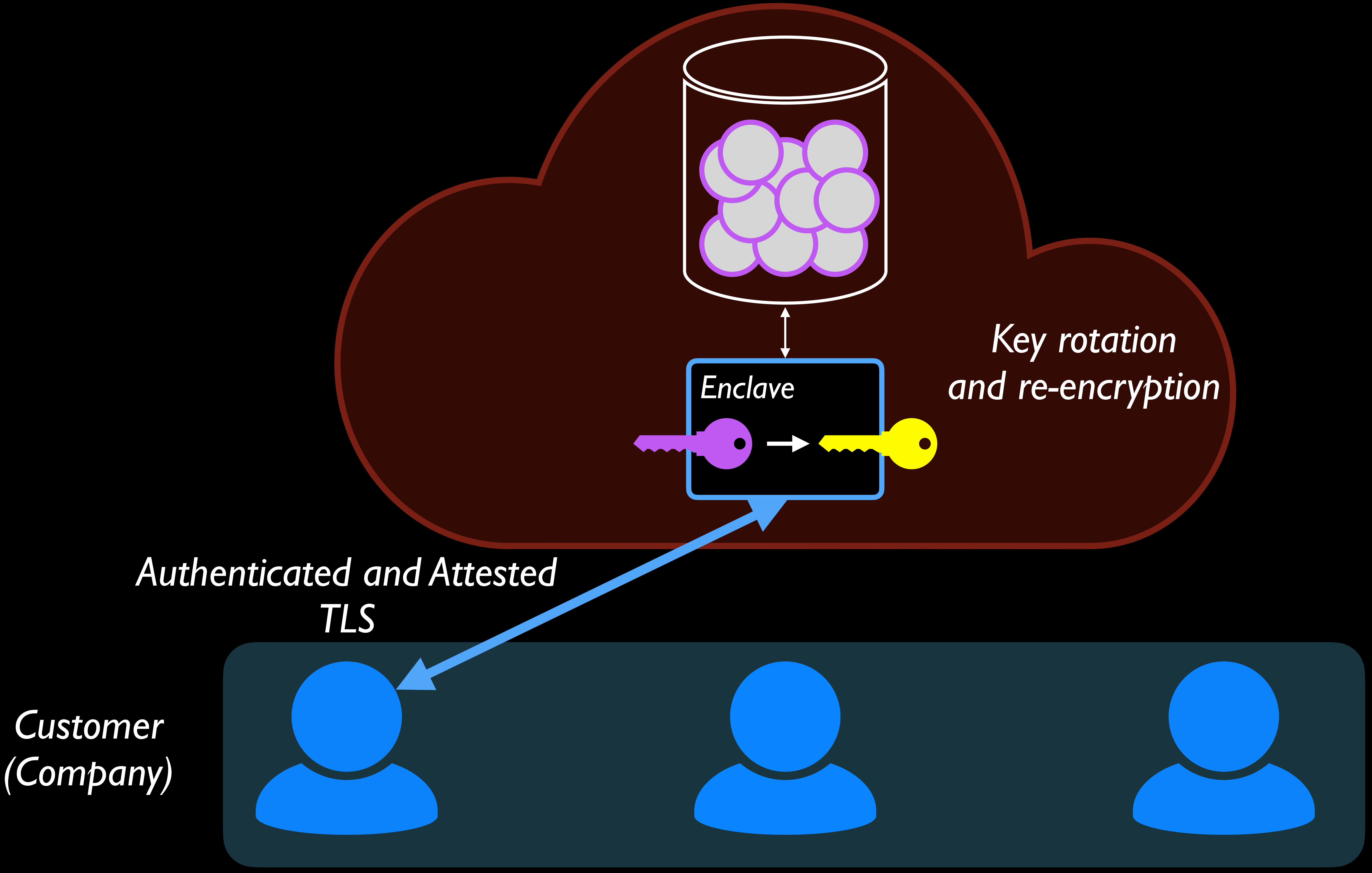
Enclave Strawman



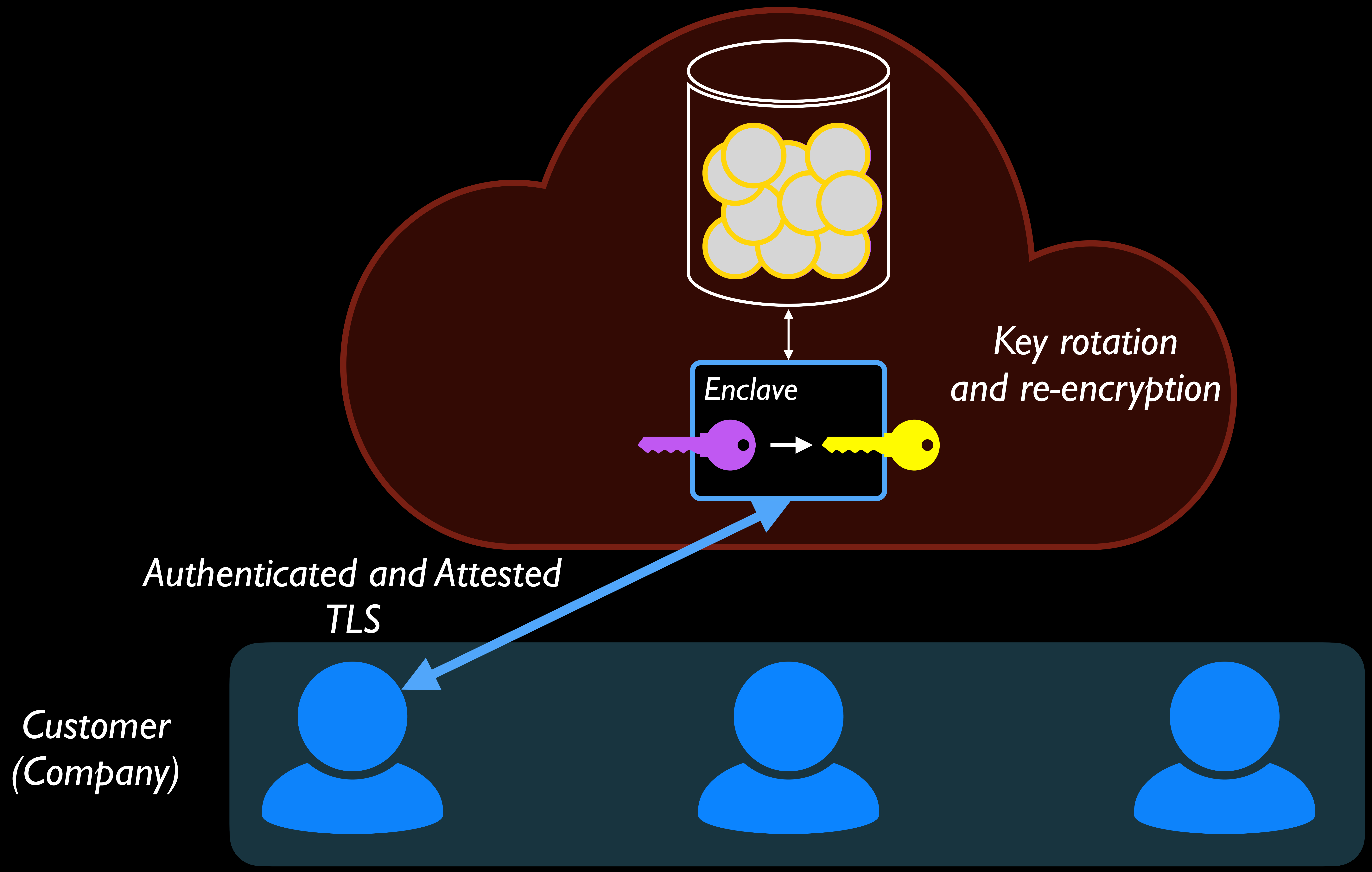
Enclave Strawman



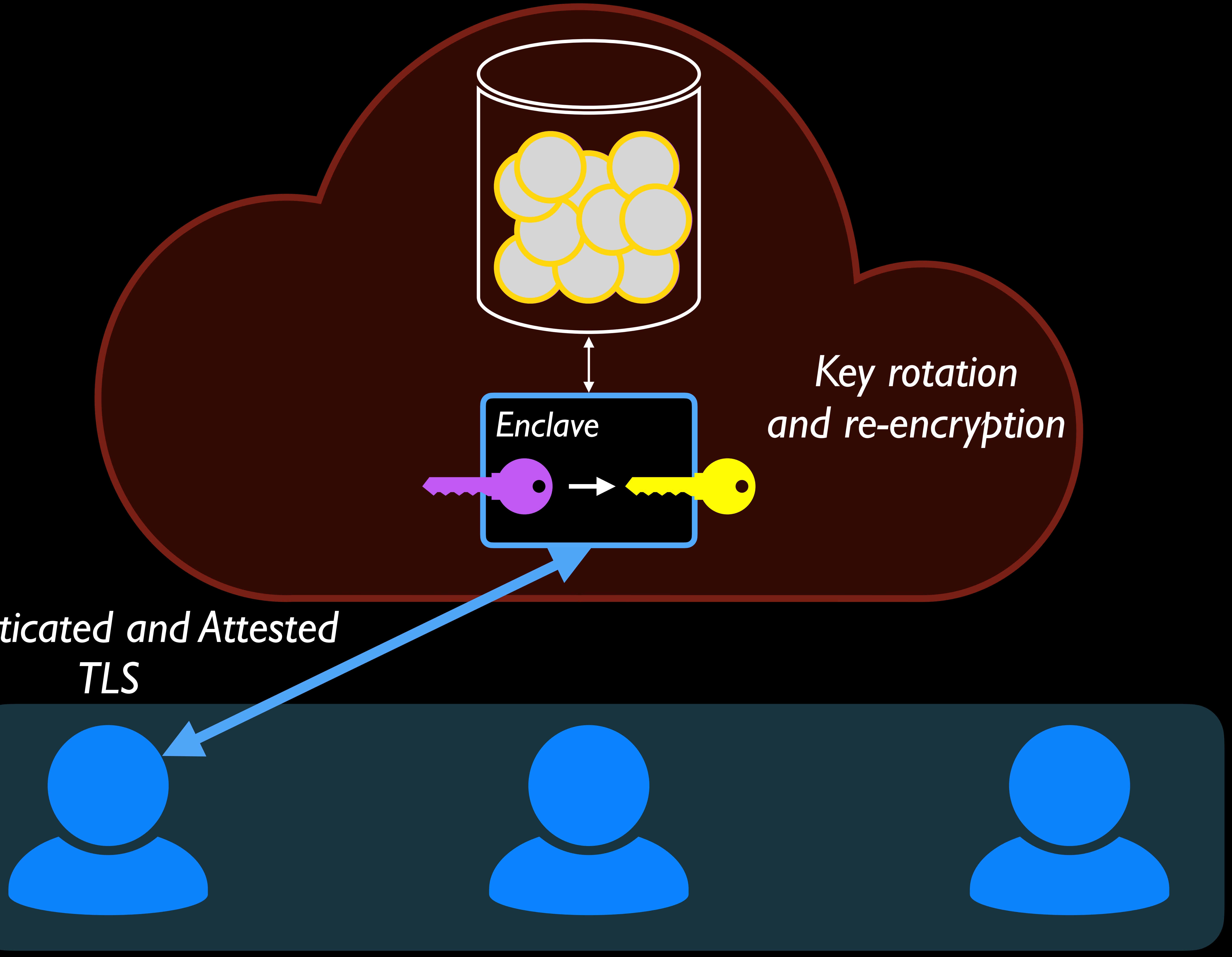
Enclave Strawman



Enclave Strawman

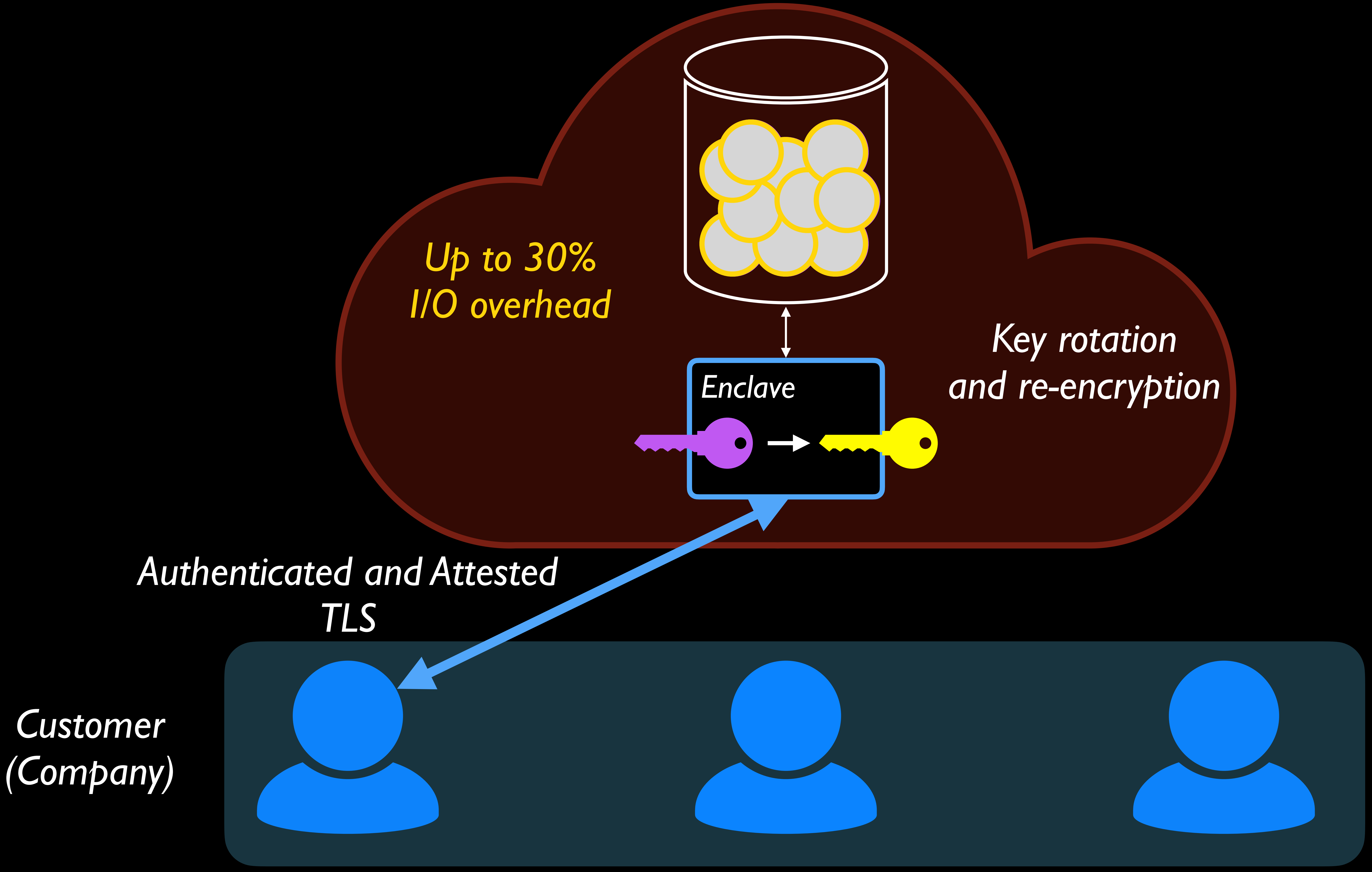


Enclave Strawman



Enclave
Key rotation and re-encryption
Authenticated and Attested TLS
Customer (Company)

Enclave Strawman



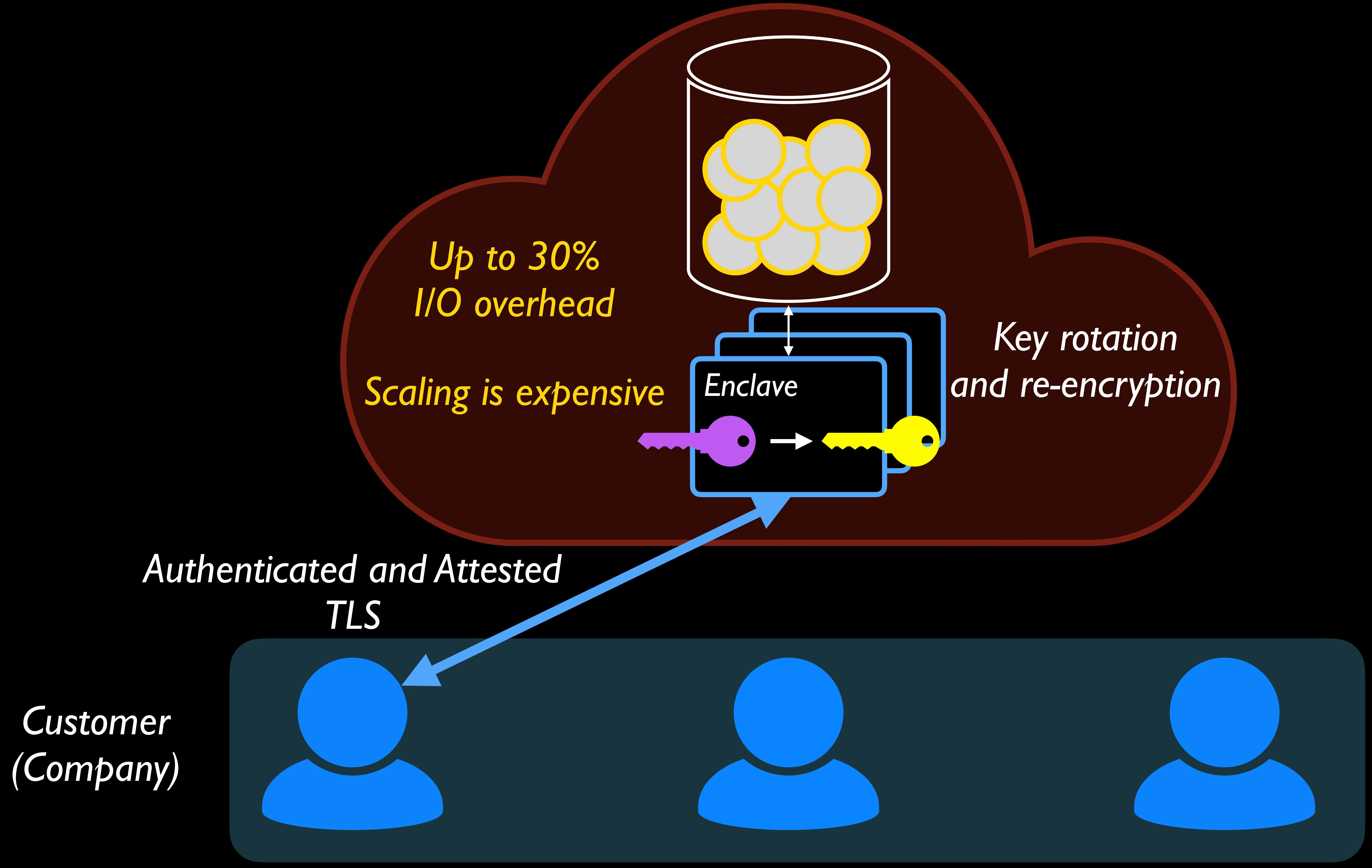
Up to 30%
I/O overhead

Key rotation
and re-encryption

Authenticated and Attested
TLS

Customer
(Company)

Enclave Strawman



Up to 30% I/O overhead

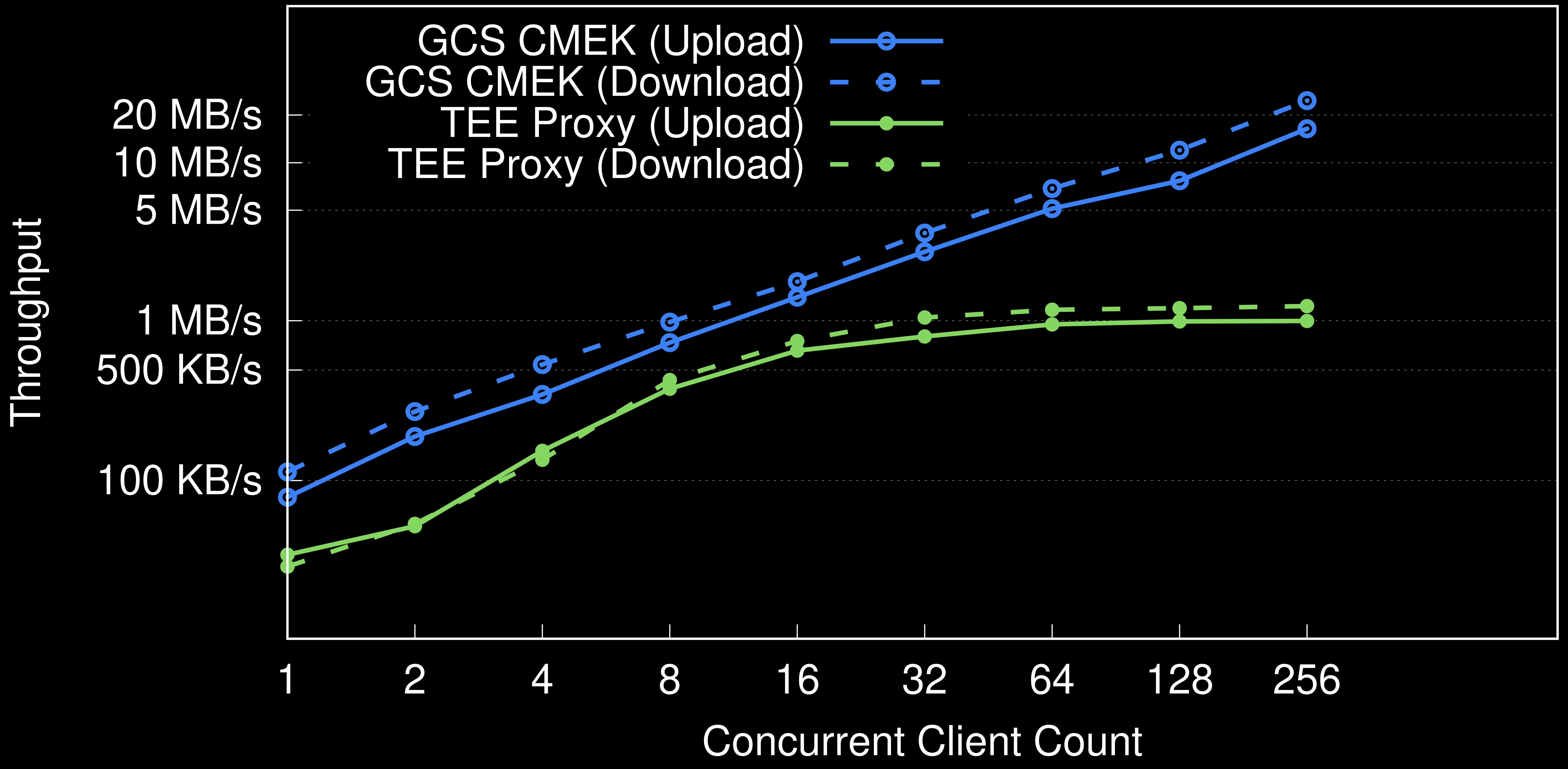
Scaling is expensive

Key rotation and re-encryption

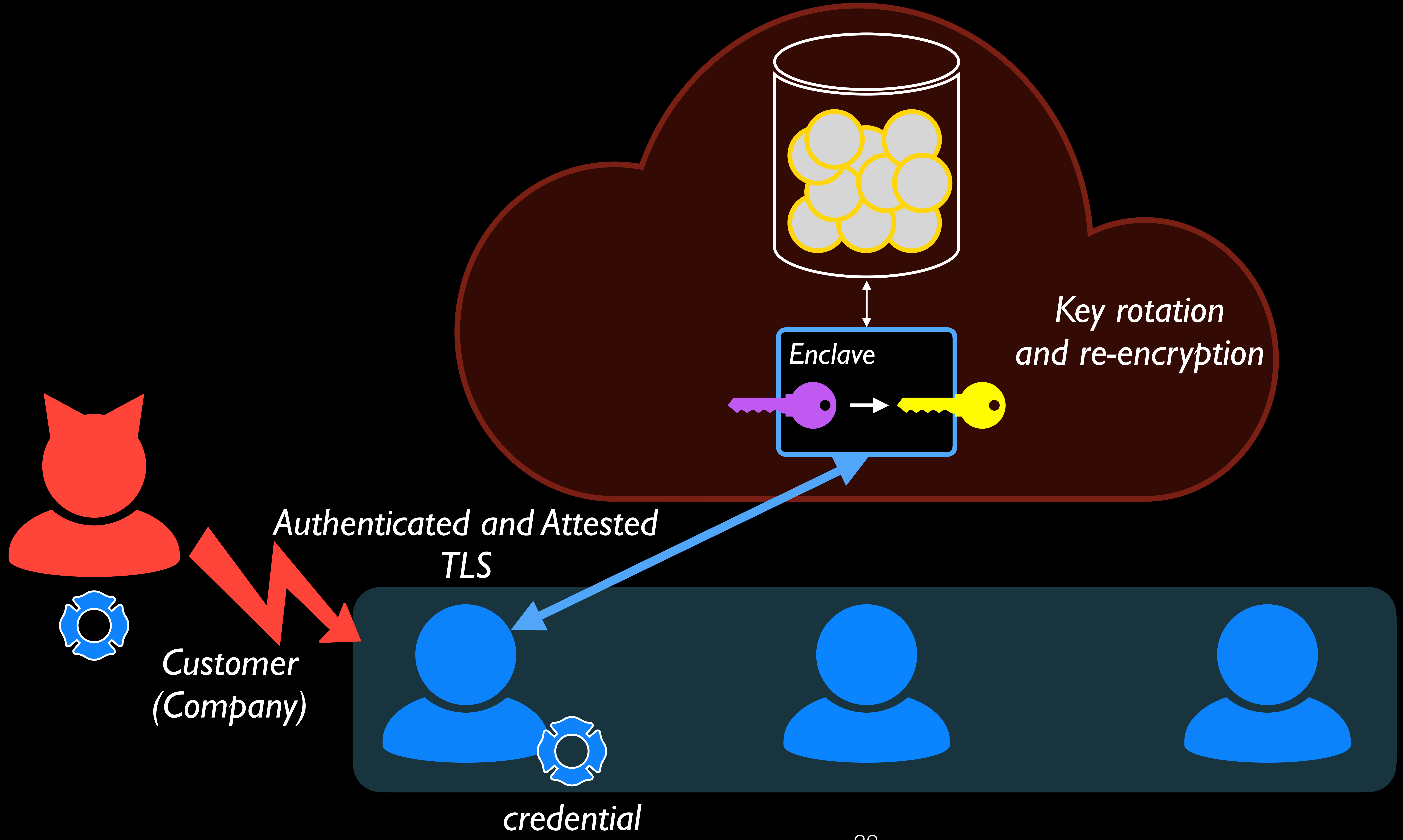
Authenticated and Attested TLS

Customer (Company)

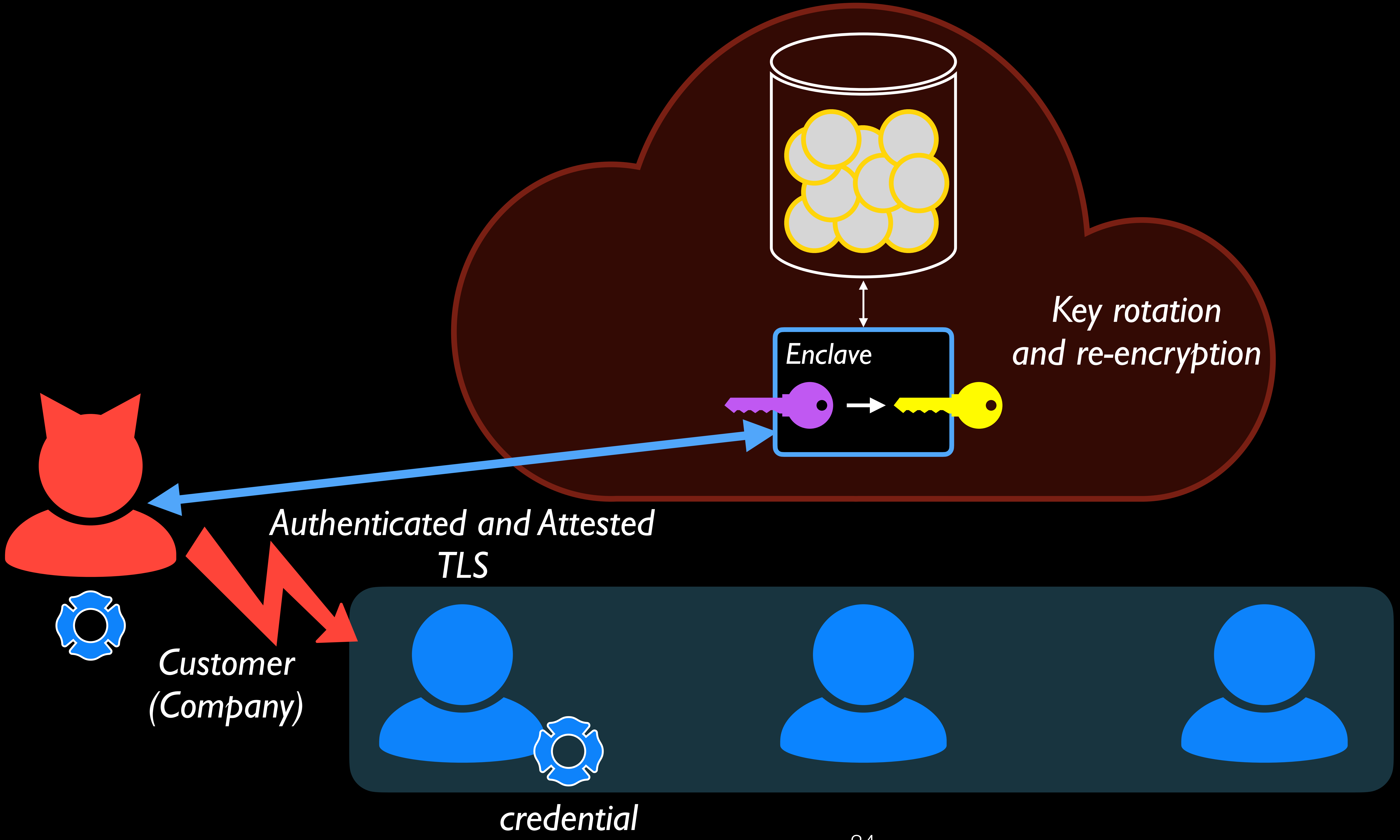
Enclave Strawman Microbenchmark



Enclave Strawman

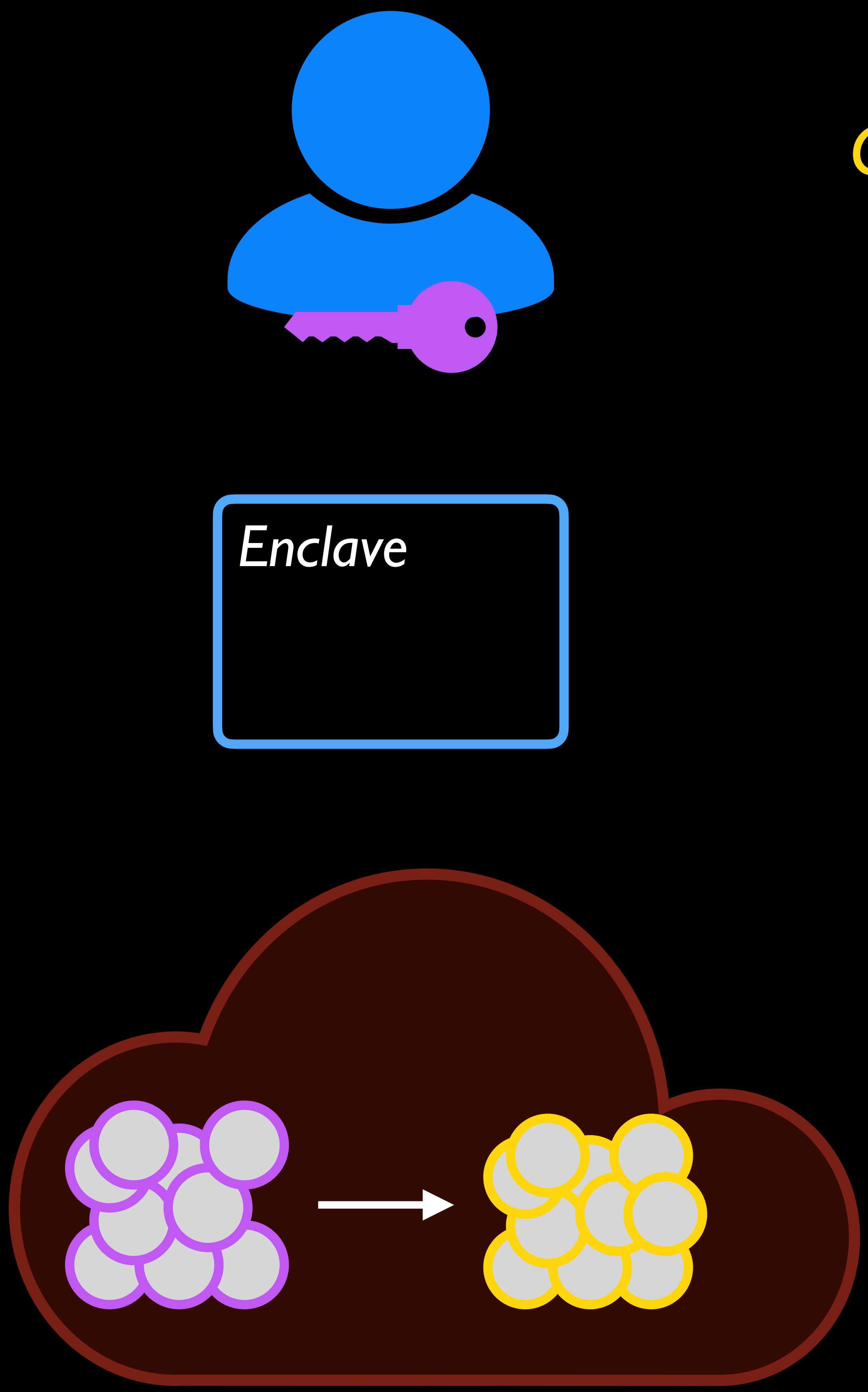


Enclave Strawman



Akeso

Akeso - Greek goddess of well-being and healing

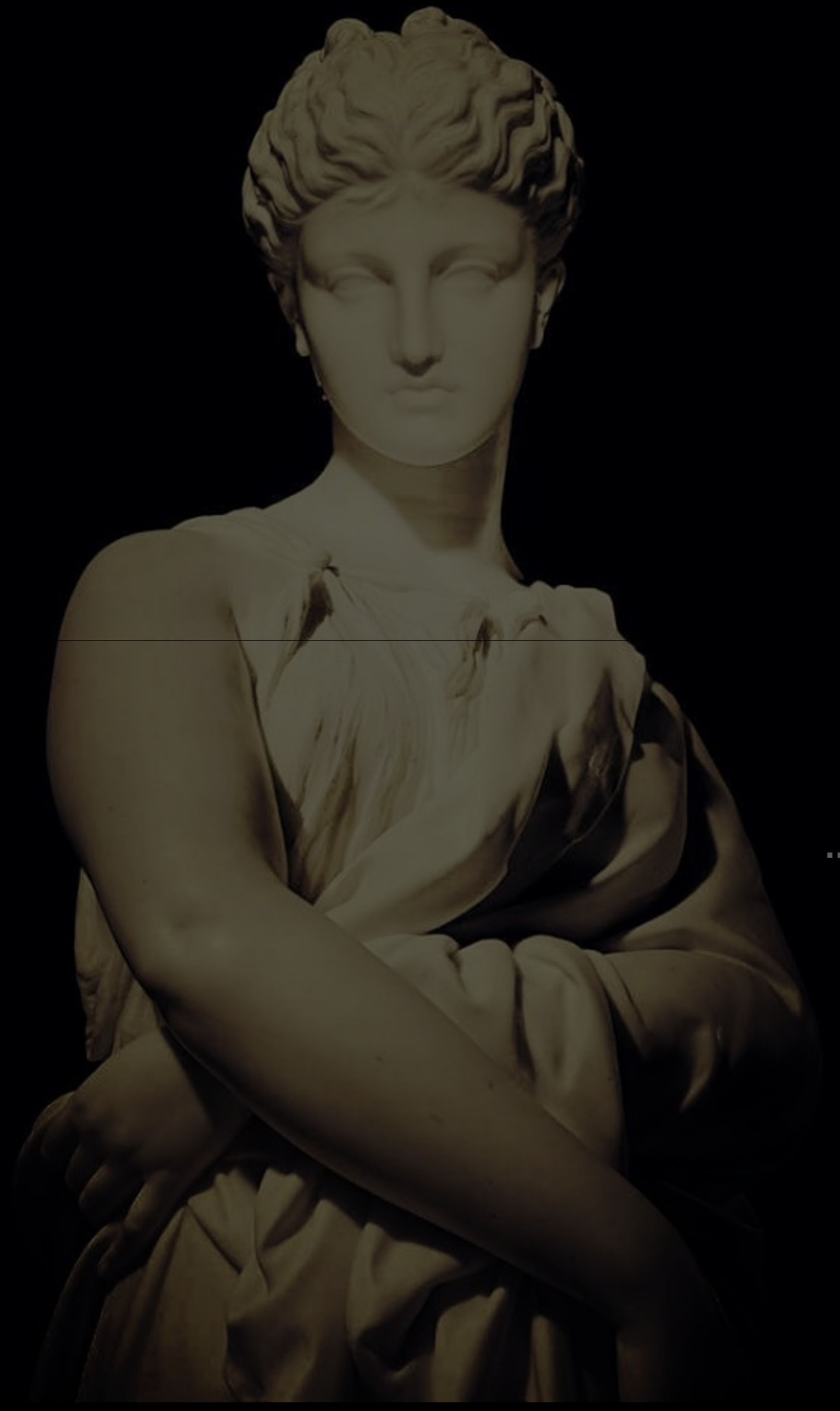


Client-side encryption and efficient key rotation

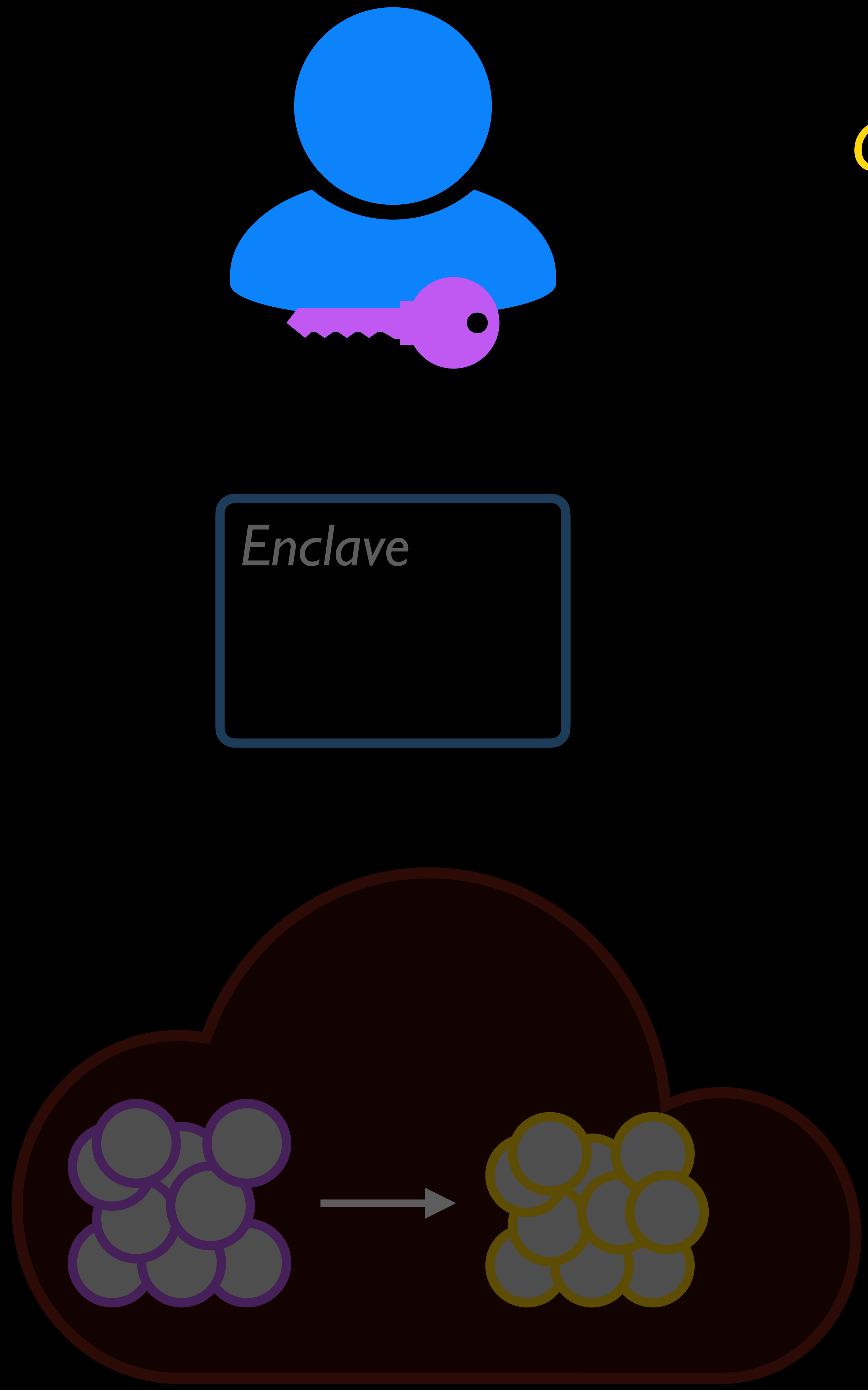
Minimal use of TEEs

Updatable Encryption

Akeso



Akeso - Greek goddess of well-being and healing

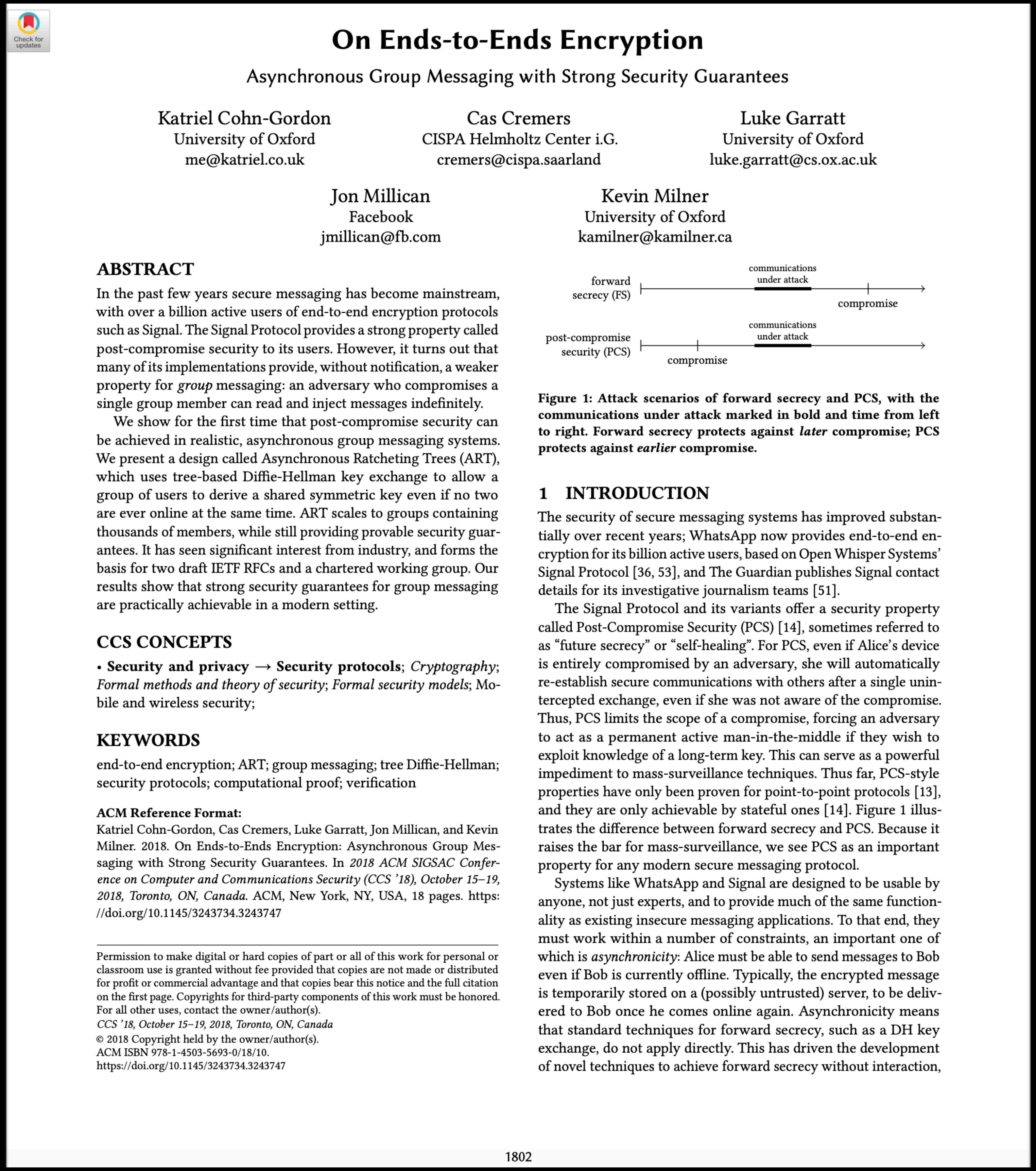


Client-side encryption and efficient key rotation

Minimal use of TEEs

Updatable Encryption

Continuous Group Key Agreement



On Ends-to-Ends Encryption

Asynchronous Group Messaging with Strong Security Guarantees

Katriel Cohn-Gordon
University of Oxford
me@katriel.co.uk

Cas Cremers
CISPA Helmholtz Center i.G.
cremers@cispa.saarland

Luke Garratt
University of Oxford
luke.garratt@cs.ox.ac.uk

Jon Millican
Facebook
jmillican@fb.com

Kevin Milner
University of Oxford
kamilner@kamilner.ca

ABSTRACT

In the past few years secure messaging has become mainstream, with over a billion active users of end-to-end encryption protocols such as Signal. The Signal Protocol provides a strong property called post-compromise security to its users. However, it turns out that many of its implementations provide, without notification, a weaker property for *group* messaging: an adversary who compromises a single group member can read and inject messages indefinitely.

We show for the first time that post-compromise security can be achieved in realistic, asynchronous group messaging systems. We present a design called Asynchronous Ratcheting Trees (ART), which uses tree-based Diffie-Hellman key exchange to allow a group of users to derive a shared symmetric key even if no two are ever online at the same time. ART scales to groups containing thousands of members, while still providing provable security guarantees. It has seen significant interest from industry, and forms the basis for two draft IETF RFCs and a chartered working group. Our results show that strong security guarantees for group messaging are practically achievable in a modern setting.

CCS CONCEPTS

• **Security and privacy** → **Security protocols**; *Cryptography*; *Formal methods and theory of security*; *Formal security models*; Mobile and wireless security;

KEYWORDS

end-to-end encryption; ART; group messaging; tree Diffie-Hellman; security protocols; computational proof; verification

ACM Reference Format:
Katriel Cohn-Gordon, Cas Cremers, Luke Garratt, Jon Millican, and Kevin Milner. 2018. On Ends-to-Ends Encryption: Asynchronous Group Messaging with Strong Security Guarantees. In *2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18), October 15–19, 2018, Toronto, ON, Canada.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3243734.3243747

CCS '18, October 15–19, 2018, Toronto, ON, Canada
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5693-0/18/10.
https://doi.org/10.1145/3243734.3243747

Figure 1: Attack scenarios of forward secrecy and PCS, with the communications under attack marked in bold and time from left to right. Forward secrecy protects against *later* compromise; PCS protects against *earlier* compromise.

1 INTRODUCTION

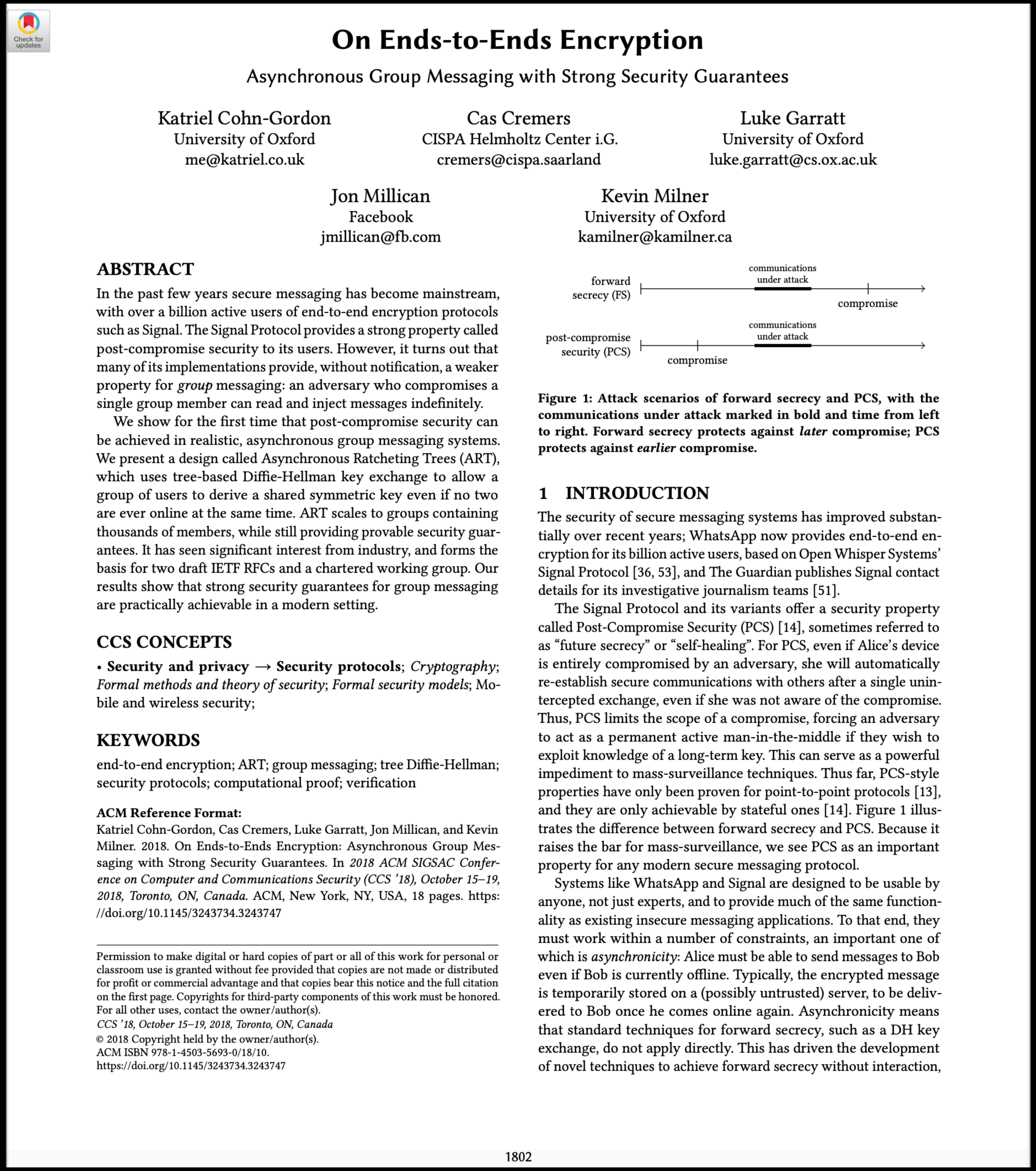
The security of secure messaging systems has improved substantially over recent years; WhatsApp now provides end-to-end encryption for its billion active users, based on Open Whisper Systems' Signal Protocol [36, 53], and The Guardian publishes Signal contact details for its investigative journalism teams [51].

The Signal Protocol and its variants offer a security property called Post-Compromise Security (PCS) [14], sometimes referred to as "future secrecy" or "self-healing". For PCS, even if Alice's device is entirely compromised by an adversary, she will automatically re-establish secure communications with others after a single unintercepted exchange, even if she was not aware of the compromise. Thus, PCS limits the scope of a compromise, forcing an adversary to act as a permanent active man-in-the-middle if they wish to exploit knowledge of a long-term key. This can serve as a powerful impediment to mass-surveillance techniques. Thus far, PCS-style properties have only been proven for point-to-point protocols [13], and they are only achievable by stateful ones [14]. Figure 1 illustrates the difference between forward secrecy and PCS. Because it raises the bar for mass-surveillance, we see PCS as an important property for any modern secure messaging protocol.

Systems like WhatsApp and Signal are designed to be usable by anyone, not just experts, and to provide much of the same functionality as existing insecure messaging applications. To that end, they must work within a number of constraints, an important one of which is *asynchronicity*: Alice must be able to send messages to Bob even if Bob is currently offline. Typically, the encrypted message is temporarily stored on a (possibly untrusted) server, to be delivered to Bob once he comes online again. Asynchronicity means that standard techniques for forward secrecy, such as a DH key exchange, do not apply directly. This has driven the development of novel techniques to achieve forward secrecy without interaction,

Cohn-Gordon et al., CCS 2018

Continuous Group Key Agreement



On Ends-to-Ends Encryption

Asynchronous Group Messaging with Strong Security Guarantees

Katriel Cohn-Gordon
University of Oxford
me@katriel.co.uk

Cas Cremers
CISPA Helmholtz Center i.G.
cremers@cispa.saarland

Luke Garratt
University of Oxford
luke.garratt@cs.ox.ac.uk

Jon Millican
Facebook
jmillican@fb.com

Kevin Milner
University of Oxford
kamilner@kamilner.ca

ABSTRACT

In the past few years secure messaging has become mainstream, with over a billion active users of end-to-end encryption protocols such as Signal. The Signal Protocol provides a strong property called post-compromise security to its users. However, it turns out that many of its implementations provide, without notification, a weaker property for *group* messaging: an adversary who compromises a single group member can read and inject messages indefinitely.

We show for the first time that post-compromise security can be achieved in realistic, asynchronous group messaging systems. We present a design called Asynchronous Ratcheting Trees (ART), which uses tree-based Diffie-Hellman key exchange to allow a group of users to derive a shared symmetric key even if no two are ever online at the same time. ART scales to groups containing thousands of members, while still providing provable security guarantees. It has seen significant interest from industry, and forms the basis for two draft IETF RFCs and a chartered working group. Our results show that strong security guarantees for group messaging are practically achievable in a modern setting.

CCS CONCEPTS

• **Security and privacy** → **Security protocols**; *Cryptography*; *Formal methods and theory of security*; *Formal security models*; Mobile and wireless security;

KEYWORDS

end-to-end encryption; ART; group messaging; tree Diffie-Hellman; security protocols; computational proof; verification

ACM Reference Format:
Katriel Cohn-Gordon, Cas Cremers, Luke Garratt, Jon Millican, and Kevin Milner. 2018. On Ends-to-Ends Encryption: Asynchronous Group Messaging with Strong Security Guarantees. In *2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18), October 15–19, 2018, Toronto, ON, Canada.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3243734.3243747

CCS '18, October 15–19, 2018, Toronto, ON, Canada
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5693-0/18/10.
https://doi.org/10.1145/3243734.3243747

Figure 1: Attack scenarios of forward secrecy and PCS, with the communications under attack marked in bold and time from left to right. Forward secrecy protects against *later* compromise; PCS protects against *earlier* compromise.

1 INTRODUCTION

The security of secure messaging systems has improved substantially over recent years; WhatsApp now provides end-to-end encryption for its billion active users, based on Open Whisper Systems' Signal Protocol [36, 53], and The Guardian publishes Signal contact details for its investigative journalism teams [51].

The Signal Protocol and its variants offer a security property called Post-Compromise Security (PCS) [14], sometimes referred to as "future secrecy" or "self-healing". For PCS, even if Alice's device is entirely compromised by an adversary, she will automatically re-establish secure communications with others after a single unintercepted exchange, even if she was not aware of the compromise. Thus, PCS limits the scope of a compromise, forcing an adversary to act as a permanent active man-in-the-middle if they wish to exploit knowledge of a long-term key. This can serve as a powerful impediment to mass-surveillance techniques. Thus far, PCS-style properties have only been proven for point-to-point protocols [13], and they are only achievable by stateful ones [14]. Figure 1 illustrates the difference between forward secrecy and PCS. Because it raises the bar for mass-surveillance, we see PCS as an important property for any modern secure messaging protocol.

Systems like WhatsApp and Signal are designed to be usable by anyone, not just experts, and to provide much of the same functionality as existing insecure messaging applications. To that end, they must work within a number of constraints, an important one of which is *asynchronicity*: Alice must be able to send messages to Bob even if Bob is currently offline. Typically, the encrypted message is temporarily stored on a (possibly untrusted) server, to be delivered to Bob once he comes online again. Asynchronicity means that standard techniques for forward secrecy, such as a DH key exchange, do not apply directly. This has driven the development of novel techniques to achieve forward secrecy without interaction,

1802

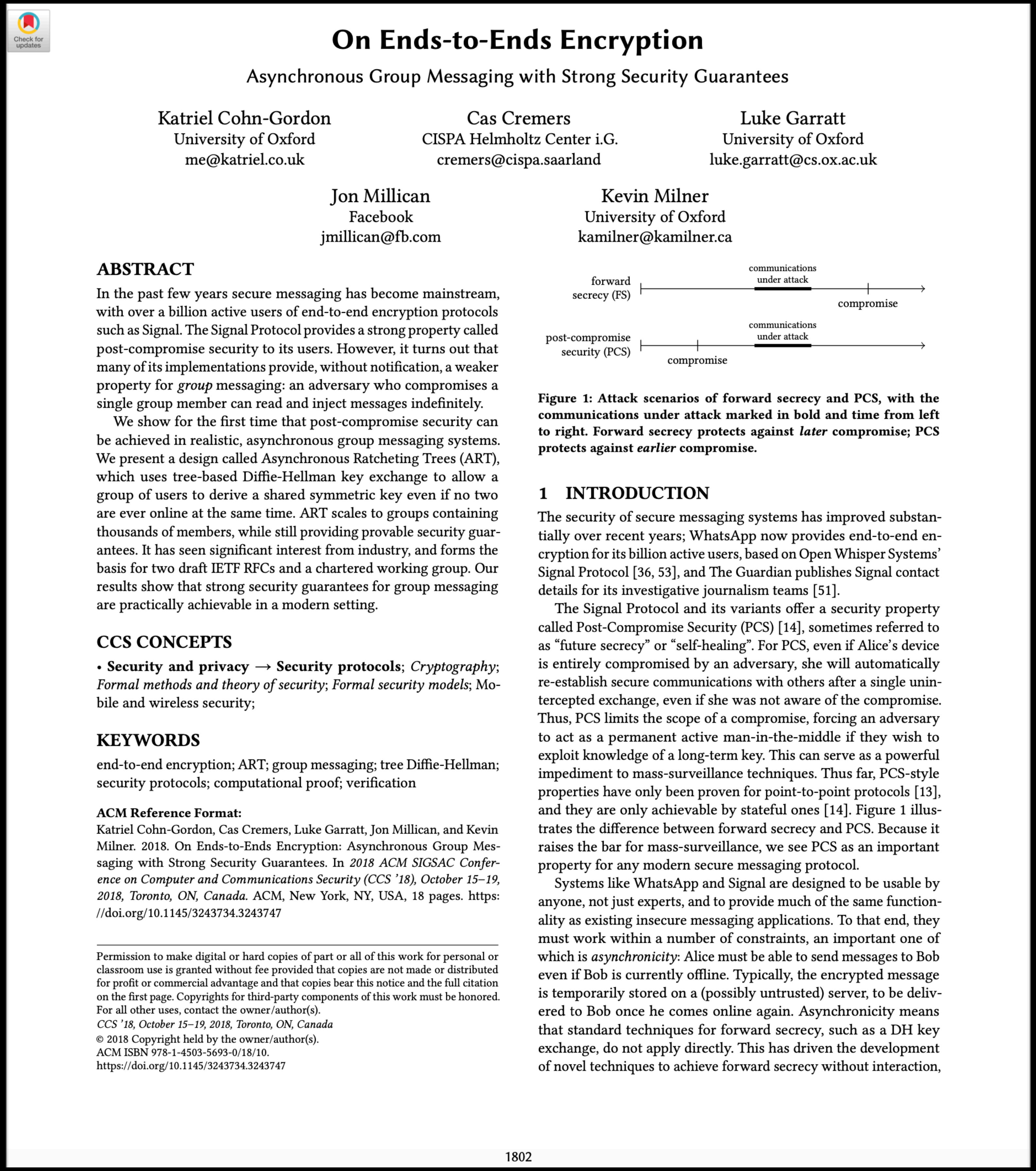
Cohn-Gordon et al., CCS 2018

1

Efficient

Key updates are O(log N), where N is the group size

Continuous Group Key Agreement



On Ends-to-Ends Encryption

Asynchronous Group Messaging with Strong Security Guarantees

Katriel Cohn-Gordon
University of Oxford
me@katriel.co.uk

Cas Cremers
CISPA Helmholtz Center i.G.
cremers@cispa.saarland

Luke Garratt
University of Oxford
luke.garratt@cs.ox.ac.uk

Jon Millican
Facebook
jmillican@fb.com

Kevin Milner
University of Oxford
kamilner@kamilner.ca

ABSTRACT

In the past few years secure messaging has become mainstream, with over a billion active users of end-to-end encryption protocols such as Signal. The Signal Protocol provides a strong property called post-compromise security to its users. However, it turns out that many of its implementations provide, without notification, a weaker property for *group* messaging: an adversary who compromises a single group member can read and inject messages indefinitely.

We show for the first time that post-compromise security can be achieved in realistic, asynchronous group messaging systems. We present a design called Asynchronous Ratcheting Trees (ART), which uses tree-based Diffie-Hellman key exchange to allow a group of users to derive a shared symmetric key even if no two are ever online at the same time. ART scales to groups containing thousands of members, while still providing provable security guarantees. It has seen significant interest from industry, and forms the basis for two draft IETF RFCs and a chartered working group. Our results show that strong security guarantees for group messaging are practically achievable in a modern setting.

CCS CONCEPTS

• **Security and privacy** → **Security protocols**; *Cryptography*; *Formal methods and theory of security*; *Formal security models*; Mobile and wireless security;

KEYWORDS

end-to-end encryption; ART; group messaging; tree Diffie-Hellman; security protocols; computational proof; verification

ACM Reference Format:
Katriel Cohn-Gordon, Cas Cremers, Luke Garratt, Jon Millican, and Kevin Milner. 2018. On Ends-to-Ends Encryption: Asynchronous Group Messaging with Strong Security Guarantees. In *2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18), October 15–19, 2018, Toronto, ON, Canada.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3243734.3243747

CCS '18, October 15–19, 2018, Toronto, ON, Canada
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5693-0/18/10.
https://doi.org/10.1145/3243734.3243747

forward secrecy (FS)
communications under attack
compromise
post-compromise security (PCS)
communications under attack
compromise

Figure 1: Attack scenarios of forward secrecy and PCS, with the communications under attack marked in bold and time from left to right. Forward secrecy protects against *later* compromise; PCS protects against *earlier* compromise.

1 INTRODUCTION

The security of secure messaging systems has improved substantially over recent years; WhatsApp now provides end-to-end encryption for its billion active users, based on Open Whisper Systems' Signal Protocol [36, 53], and The Guardian publishes Signal contact details for its investigative journalism teams [51].

The Signal Protocol and its variants offer a security property called Post-Compromise Security (PCS) [14], sometimes referred to as "future secrecy" or "self-healing". For PCS, even if Alice's device is entirely compromised by an adversary, she will automatically re-establish secure communications with others after a single unintercepted exchange, even if she was not aware of the compromise. Thus, PCS limits the scope of a compromise, forcing an adversary to act as a permanent active man-in-the-middle if they wish to exploit knowledge of a long-term key. This can serve as a powerful impediment to mass-surveillance techniques. Thus far, PCS-style properties have only been proven for point-to-point protocols [13], and they are only achievable by stateful ones [14]. Figure 1 illustrates the difference between forward secrecy and PCS. Because it raises the bar for mass-surveillance, we see PCS as an important property for any modern secure messaging protocol.

Systems like WhatsApp and Signal are designed to be usable by anyone, not just experts, and to provide much of the same functionality as existing insecure messaging applications. To that end, they must work within a number of constraints, an important one of which is *asynchronicity*: Alice must be able to send messages to Bob even if Bob is currently offline. Typically, the encrypted message is temporarily stored on a (possibly untrusted) server, to be delivered to Bob once he comes online again. Asynchronicity means that standard techniques for forward secrecy, such as a DH key exchange, do not apply directly. This has driven the development of novel techniques to achieve forward secrecy without interaction,

Cohn-Gordon et al., CCS 2018

1

Efficient

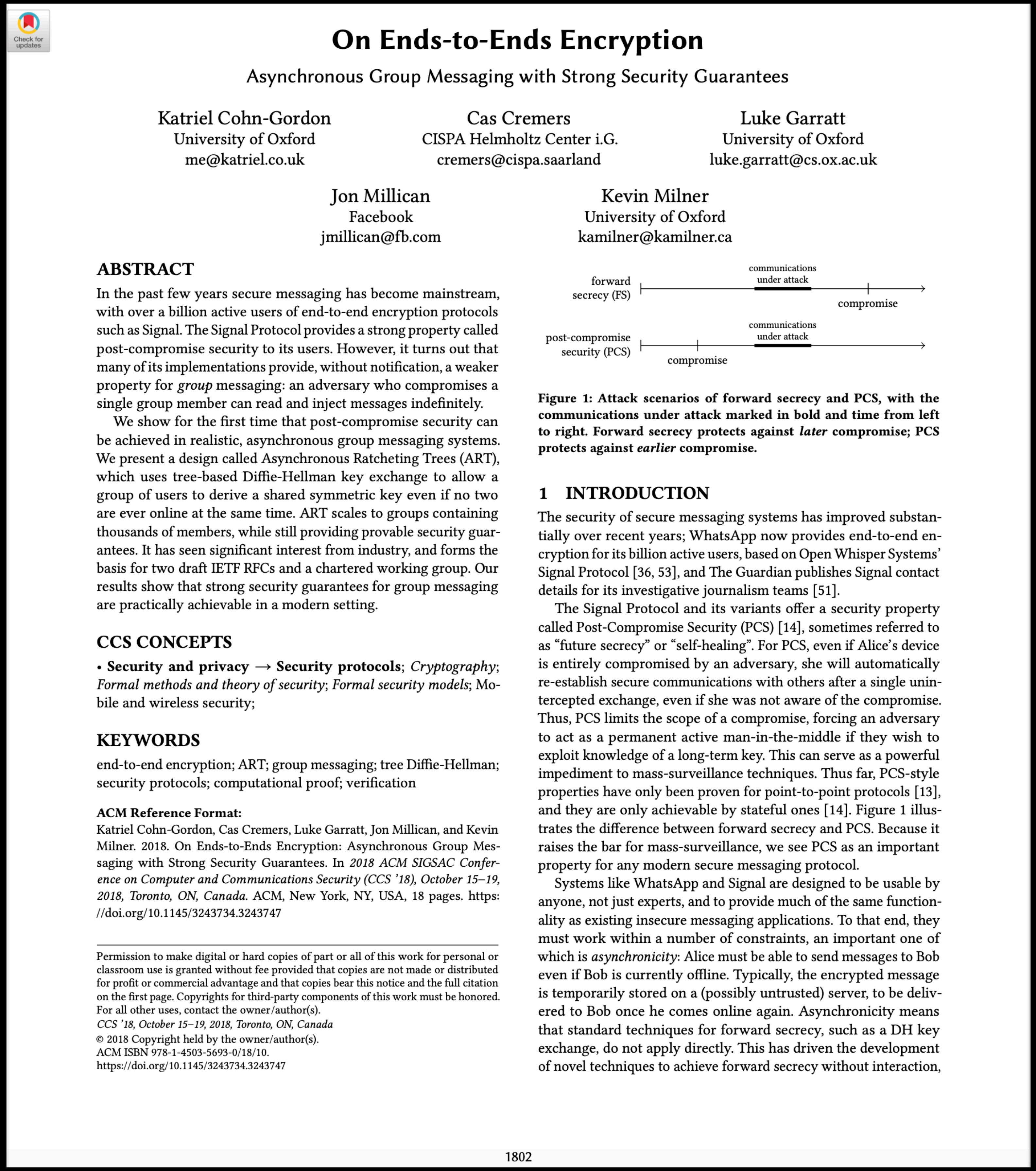
Key updates are O(log N), where N is the group size

2

Asynchronous

Achieves key agreement even if some members are offline

Continuous Group Key Agreement



On Ends-to-Ends Encryption

Asynchronous Group Messaging with Strong Security Guarantees

Katriel Cohn-Gordon
University of Oxford
me@katriel.co.uk

Cas Cremers
CISPA Helmholtz Center i.G.
cremers@cispa.saarland

Luke Garratt
University of Oxford
luke.garratt@cs.ox.ac.uk

Jon Millican
Facebook
jmillican@fb.com

Kevin Milner
University of Oxford
kamilner@kamilner.ca

ABSTRACT

In the past few years secure messaging has become mainstream, with over a billion active users of end-to-end encryption protocols such as Signal. The Signal Protocol provides a strong property called post-compromise security to its users. However, it turns out that many of its implementations provide, without notification, a weaker property for *group* messaging: an adversary who compromises a single group member can read and inject messages indefinitely.

We show for the first time that post-compromise security can be achieved in realistic, asynchronous group messaging systems. We present a design called Asynchronous Ratcheting Trees (ART), which uses tree-based Diffie-Hellman key exchange to allow a group of users to derive a shared symmetric key even if no two are ever online at the same time. ART scales to groups containing thousands of members, while still providing provable security guarantees. It has seen significant interest from industry, and forms the basis for two draft IETF RFCs and a chartered working group. Our results show that strong security guarantees for group messaging are practically achievable in a modern setting.

CCS CONCEPTS

• **Security and privacy** → **Security protocols**; *Cryptography*; *Formal methods and theory of security*; *Formal security models*; Mobile and wireless security;

KEYWORDS

end-to-end encryption; ART; group messaging; tree Diffie-Hellman; security protocols; computational proof; verification

ACM Reference Format:
Katriel Cohn-Gordon, Cas Cremers, Luke Garratt, Jon Millican, and Kevin Milner. 2018. On Ends-to-Ends Encryption: Asynchronous Group Messaging with Strong Security Guarantees. In *2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18), October 15–19, 2018, Toronto, ON, Canada.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3243734.3243747

forward secrecy (FS)
communications under attack
compromise
post-compromise security (PCS)
communications under attack
compromise

Figure 1: Attack scenarios of forward secrecy and PCS, with the communications under attack marked in bold and time from left to right. Forward secrecy protects against *later* compromise; PCS protects against *earlier* compromise.

1 INTRODUCTION

The security of secure messaging systems has improved substantially over recent years; WhatsApp now provides end-to-end encryption for its billion active users, based on Open Whisper Systems' Signal Protocol [36, 53], and The Guardian publishes Signal contact details for its investigative journalism teams [51].

The Signal Protocol and its variants offer a security property called Post-Compromise Security (PCS) [14], sometimes referred to as "future secrecy" or "self-healing". For PCS, even if Alice's device is entirely compromised by an adversary, she will automatically re-establish secure communications with others after a single unintercepted exchange, even if she was not aware of the compromise. Thus, PCS limits the scope of a compromise, forcing an adversary to act as a permanent active man-in-the-middle if they wish to exploit knowledge of a long-term key. This can serve as a powerful impediment to mass-surveillance techniques. Thus far, PCS-style properties have only been proven for point-to-point protocols [13], and they are only achievable by stateful ones [14]. Figure 1 illustrates the difference between forward secrecy and PCS. Because it raises the bar for mass-surveillance, we see PCS as an important property for any modern secure messaging protocol.

Systems like WhatsApp and Signal are designed to be usable by anyone, not just experts, and to provide much of the same functionality as existing insecure messaging applications. To that end, they must work within a number of constraints, an important one of which is *asynchronicity*: Alice must be able to send messages to Bob even if Bob is currently offline. Typically, the encrypted message is temporarily stored on a (possibly untrusted) server, to be delivered to Bob once he comes online again. Asynchronicity means that standard techniques for forward secrecy, such as a DH key exchange, do not apply directly. This has driven the development of novel techniques to achieve forward secrecy without interaction,

CCS '18, October 15–19, 2018, Toronto, ON, Canada
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5693-0/18/10.
https://doi.org/10.1145/3243734.3243747

Cohn-Gordon et al., CCS 2018

1

Efficient

Key updates are O(log N), where N is the group size

2

Asynchronous

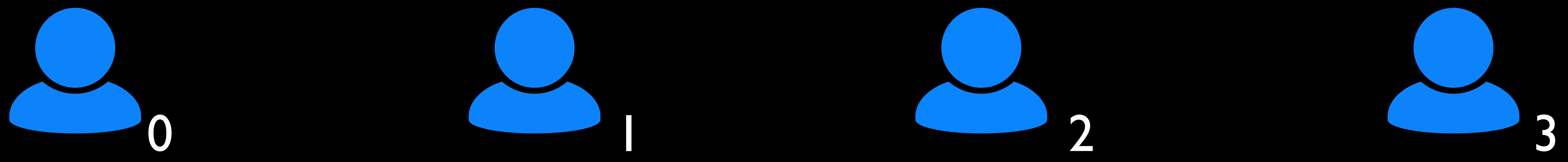
Achieves key agreement even if some members are offline

3

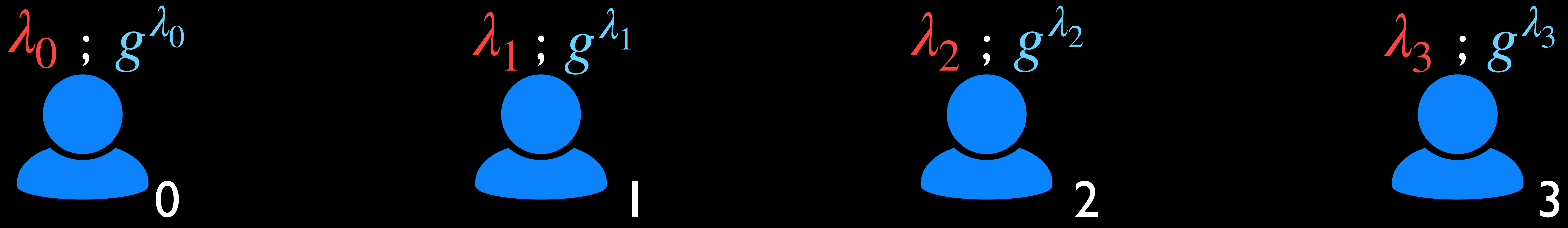
Post-Compromise Secure

Regains security of group key upon one key rotation without adversarial interference

Asynchronous Ratcheting Tree (ART)



Asynchronous Ratcheting Tree (ART)



λ_0 ; g^{λ_0}

0

λ_1 ; g^{λ_1}

1

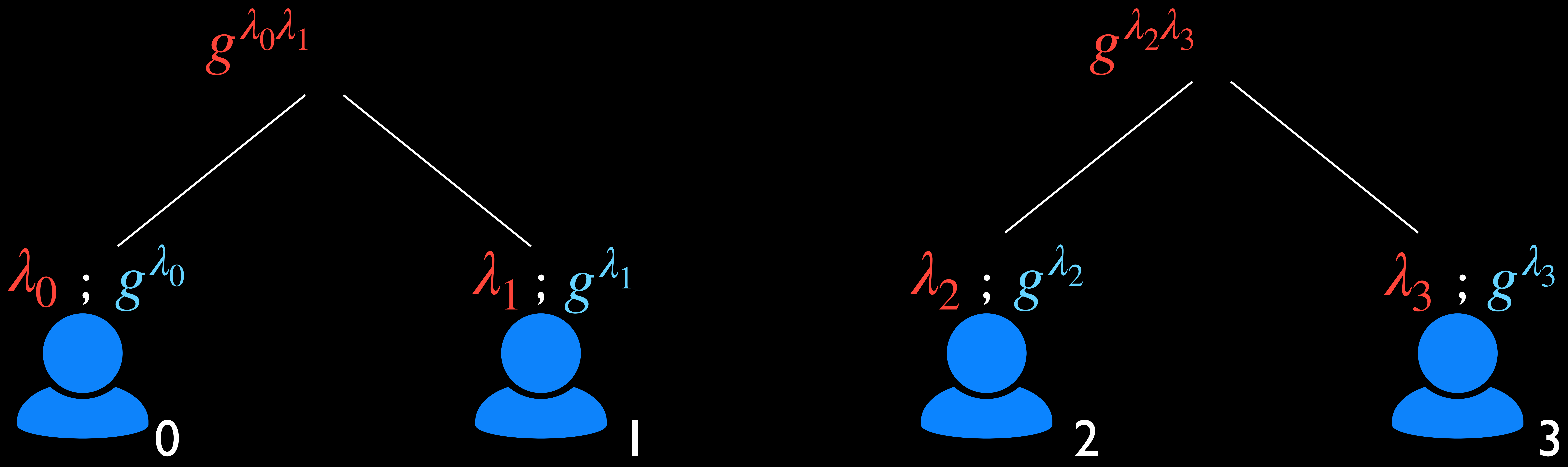
λ_2 ; g^{λ_2}

2

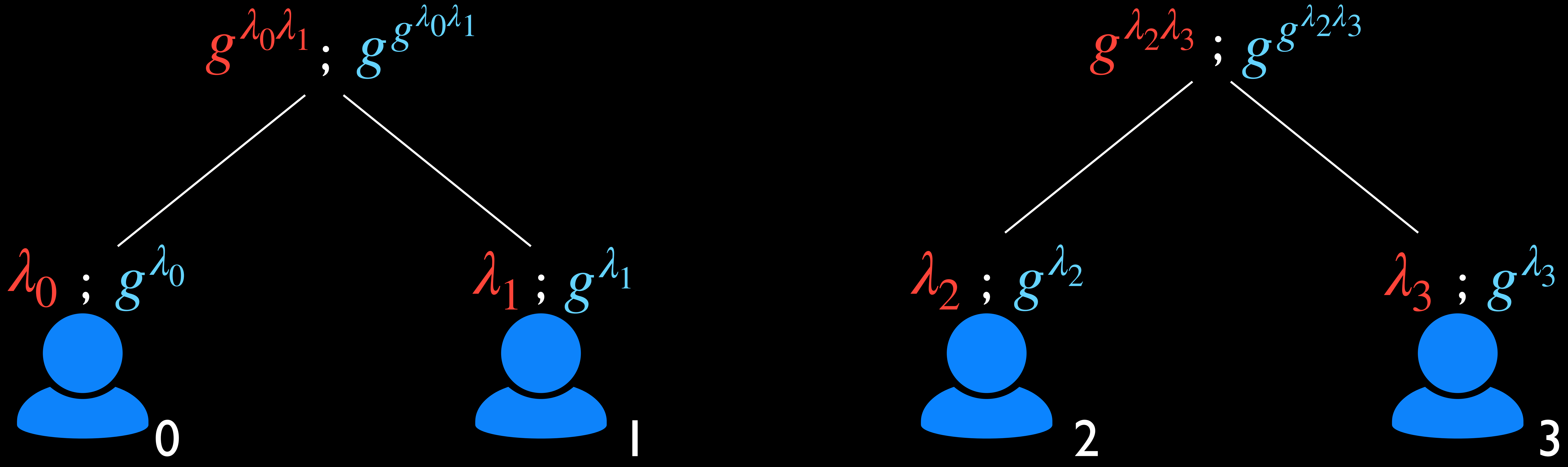
λ_3 ; g^{λ_3}

3

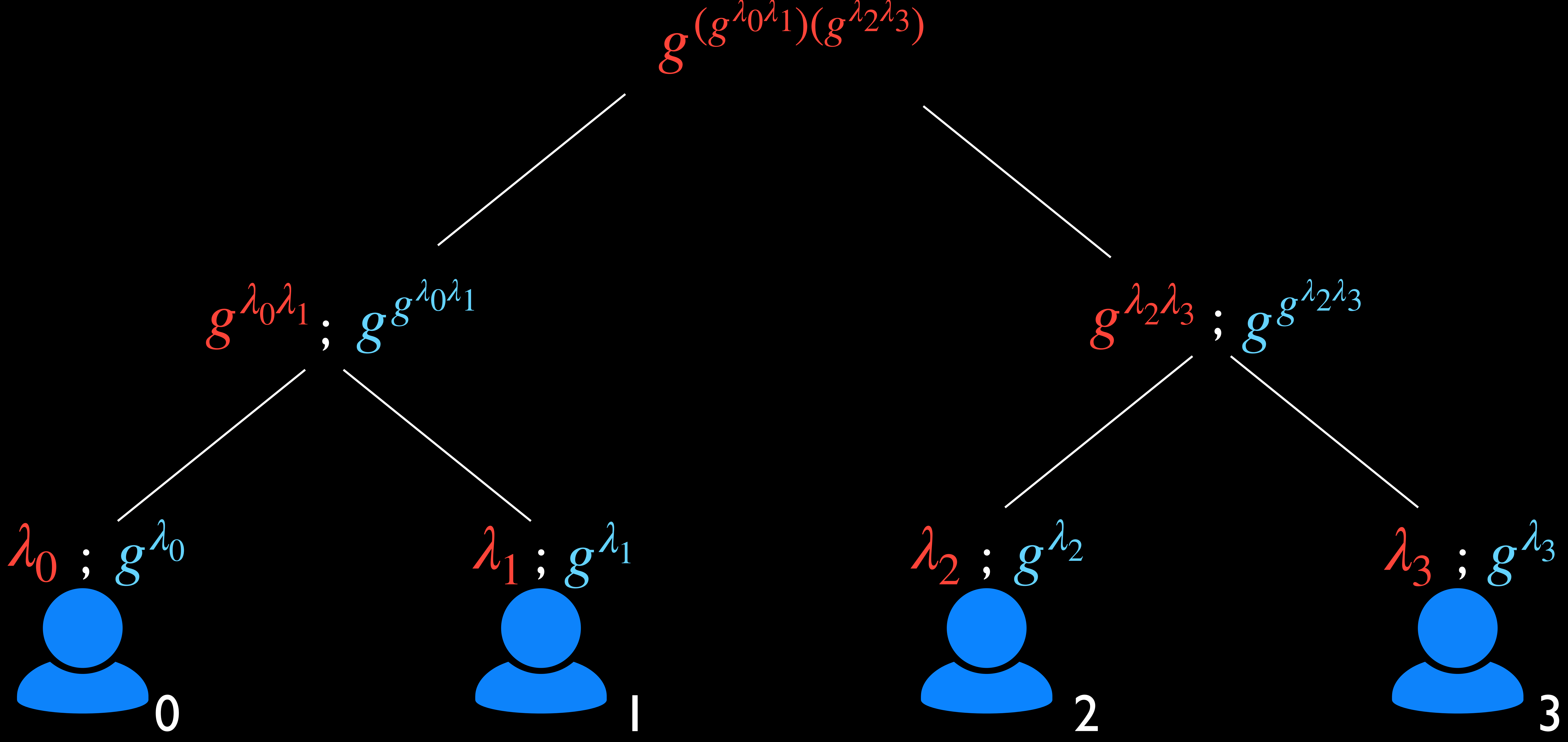
Asynchronous Ratcheting Tree (ART)


$g^{\lambda_0\lambda_1}$
$g^{\lambda_2\lambda_3}$
λ_0 ; g^{λ_0}
λ_1 ; g^{λ_1}
λ_2 ; g^{λ_2}
λ_3 ; g^{λ_3}
0
1
2
3

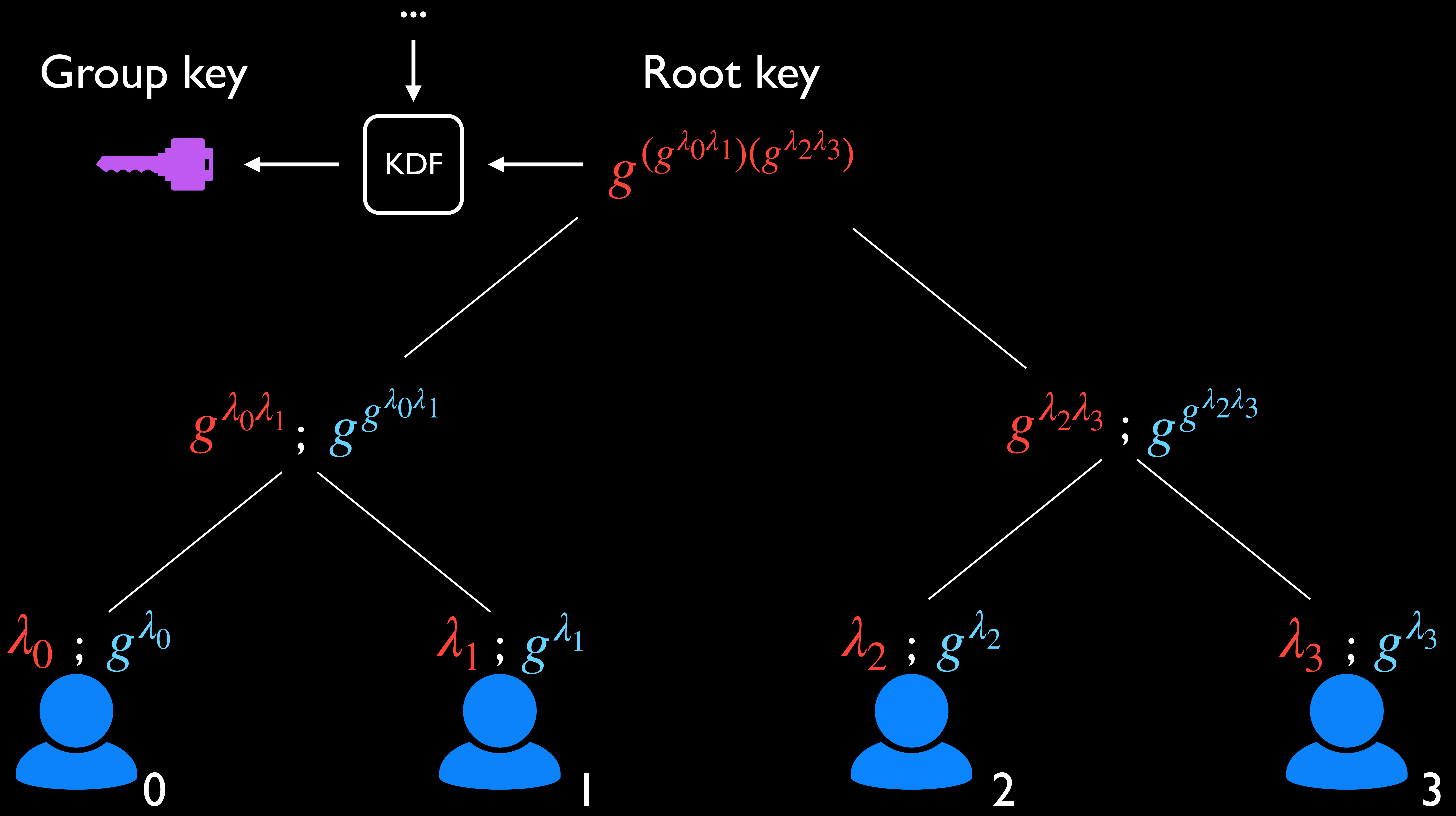
Asynchronous Ratcheting Tree (ART)


$g^{\lambda_0\lambda_1}$; $g^{g^{\lambda_0\lambda_1}}$
$g^{\lambda_2\lambda_3}$; $g^{g^{\lambda_2\lambda_3}}$
λ_0 ; g^{λ_0}
λ_1 ; g^{λ_1}
λ_2 ; g^{λ_2}
λ_3 ; g^{λ_3}
0
1
2
3

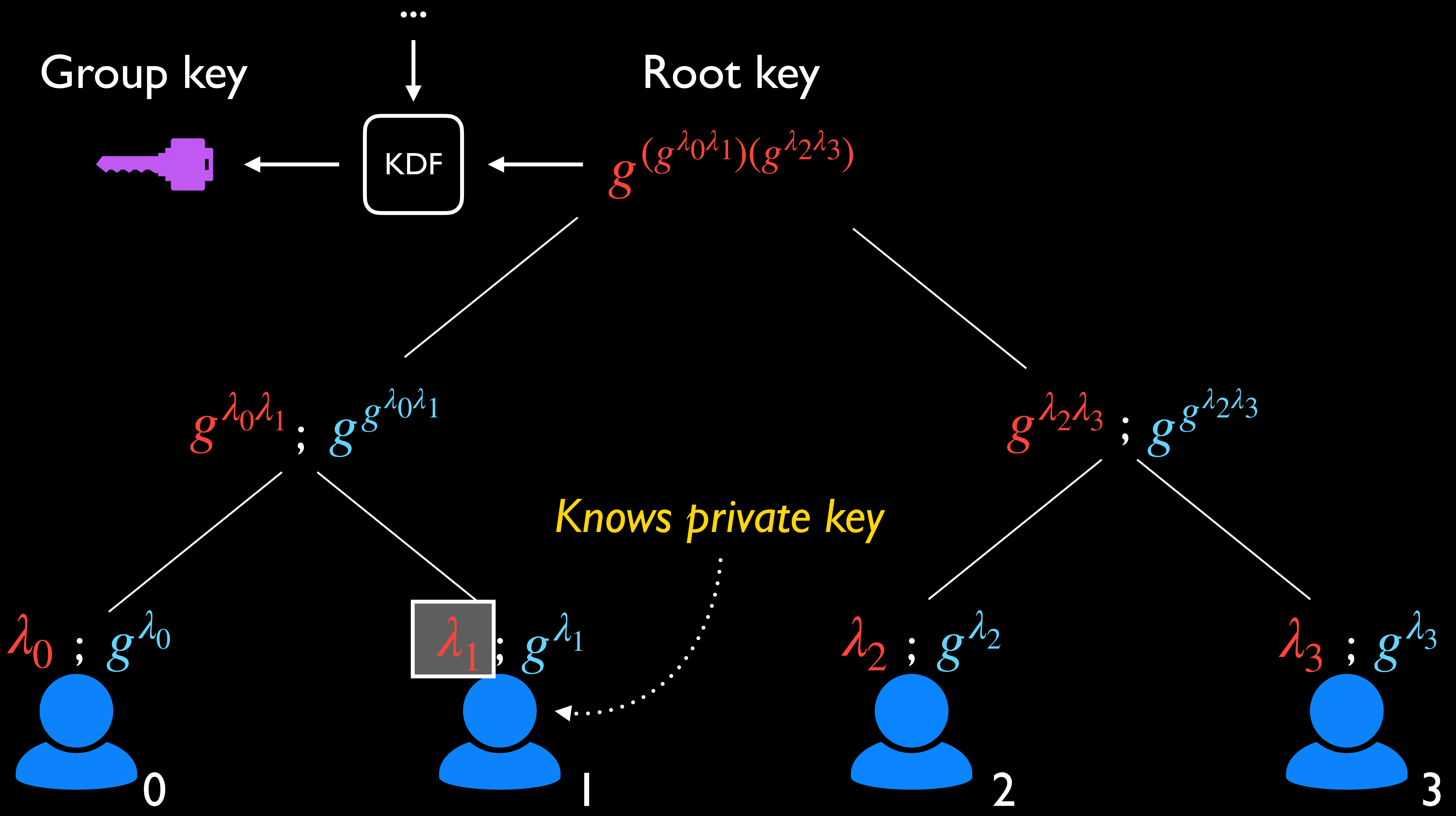
Asynchronous Ratcheting Tree (ART)


$g^{(g^{\lambda_0\lambda_1})(g^{\lambda_2\lambda_3})}$
$g^{\lambda_0\lambda_1}$; $g^{g^{\lambda_0\lambda_1}}$
$g^{\lambda_2\lambda_3}$; $g^{g^{\lambda_2\lambda_3}}$
λ_0 ; g^{λ_0}
λ_1 ; g^{λ_1}
λ_2 ; g^{λ_2}
λ_3 ; g^{λ_3}
0
1
2
3

Asynchronous Ratcheting Tree (ART)

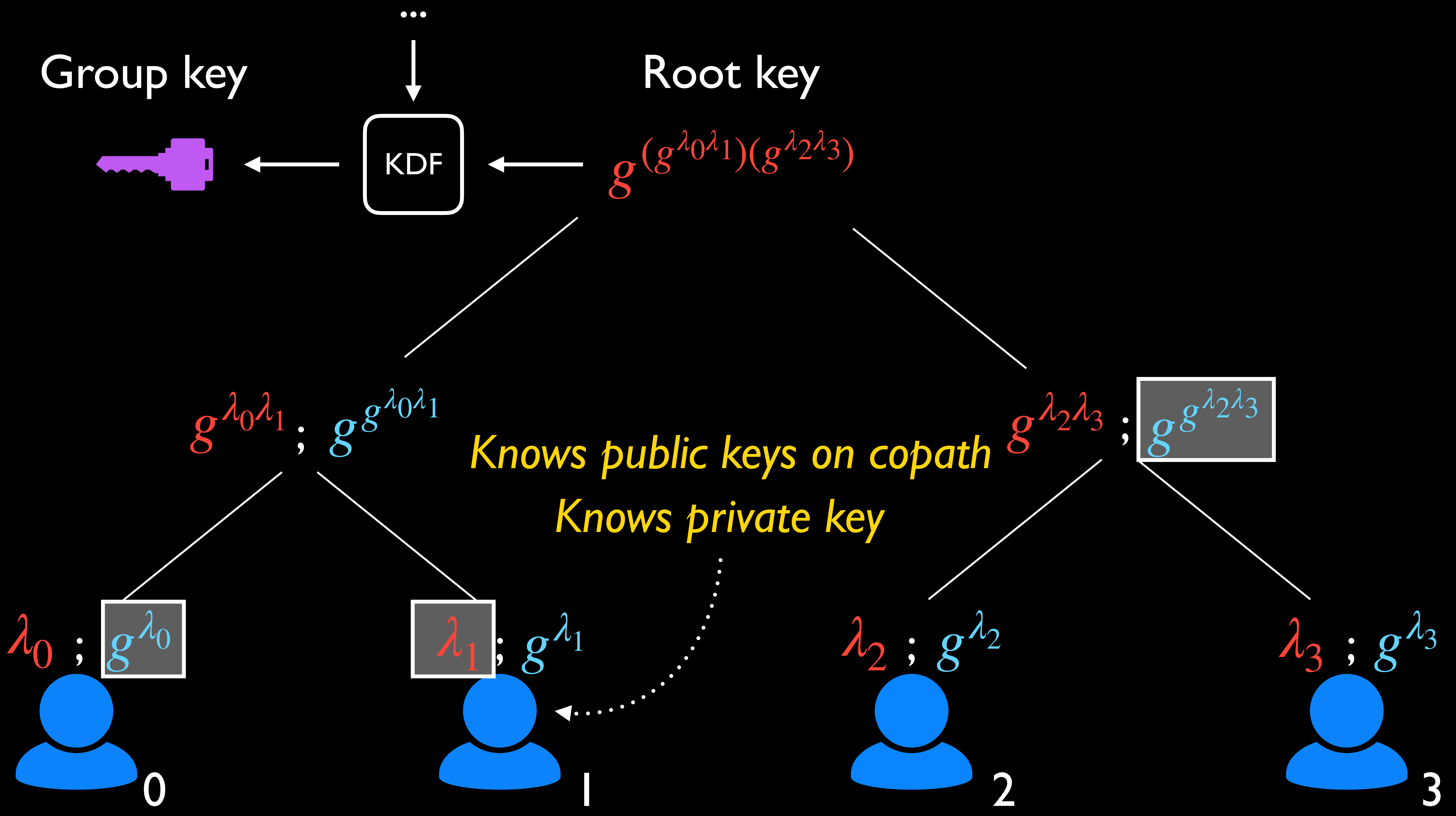

...
Group key
KDF
Root key
$g^{(g^{\lambda_0\lambda_1})(g^{\lambda_2\lambda_3})}$
$g^{\lambda_0\lambda_1}$; $g^{g^{\lambda_0\lambda_1}}$
$g^{\lambda_2\lambda_3}$; $g^{g^{\lambda_2\lambda_3}}$
λ_0 ; g^{λ_0}
λ_1 ; g^{λ_1}
λ_2 ; g^{λ_2}
λ_3 ; g^{λ_3}
0
1
2
3

Asynchronous Ratcheting Tree (ART)

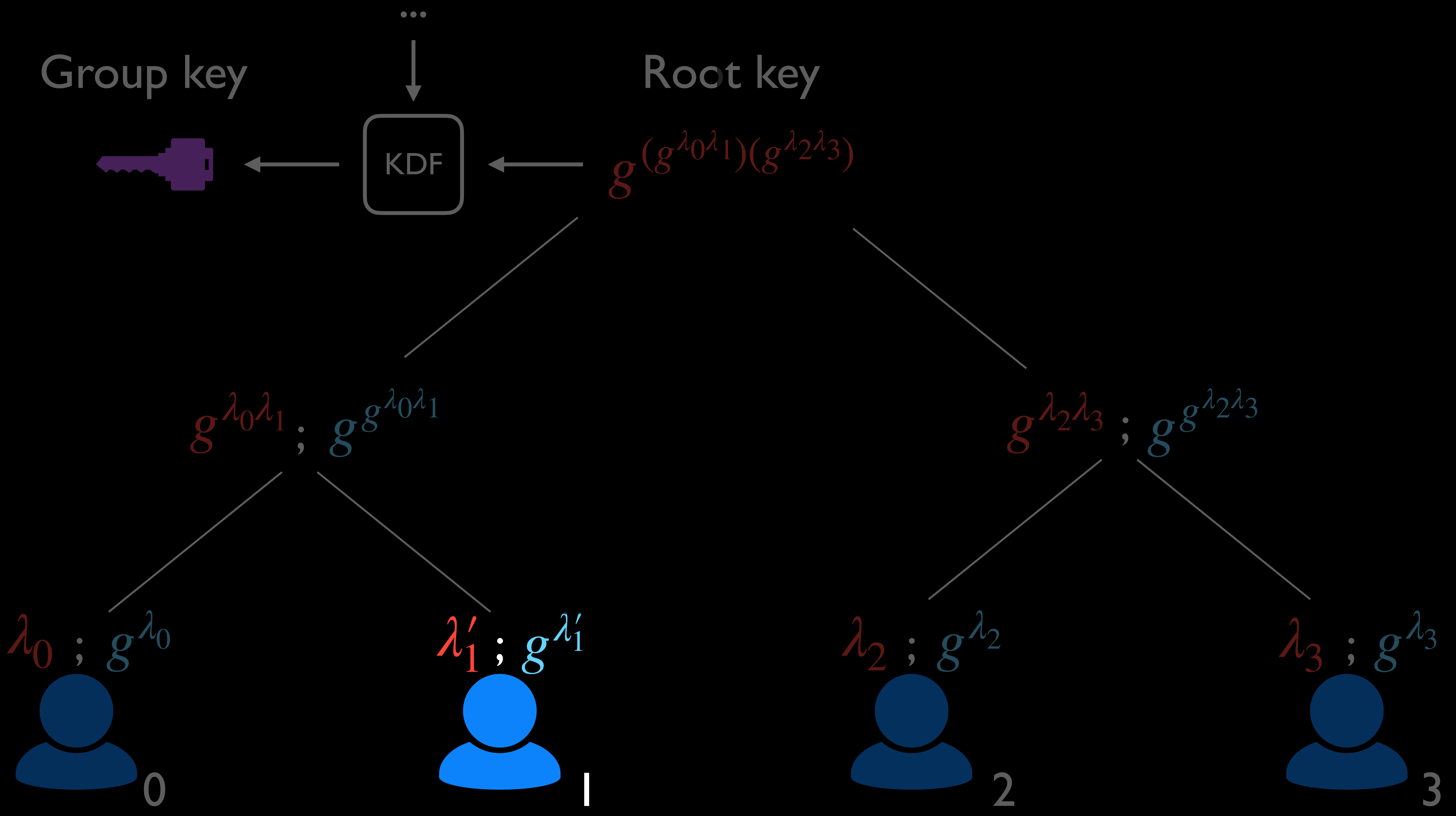


...
Group key
KDF
Root key
$g^{(g^{\lambda_0\lambda_1})(g^{\lambda_2\lambda_3})}$
$g^{\lambda_0\lambda_1}$; $g^{g^{\lambda_0\lambda_1}}$
$g^{\lambda_2\lambda_3}$; $g^{g^{\lambda_2\lambda_3}}$
Knows private key
λ_0 ; g^{λ_0}
λ_1 ; g^{λ_1}
λ_2 ; g^{λ_2}
λ_3 ; g^{λ_3}
0
1
2
3

Asynchronous Ratcheting Tree (ART)

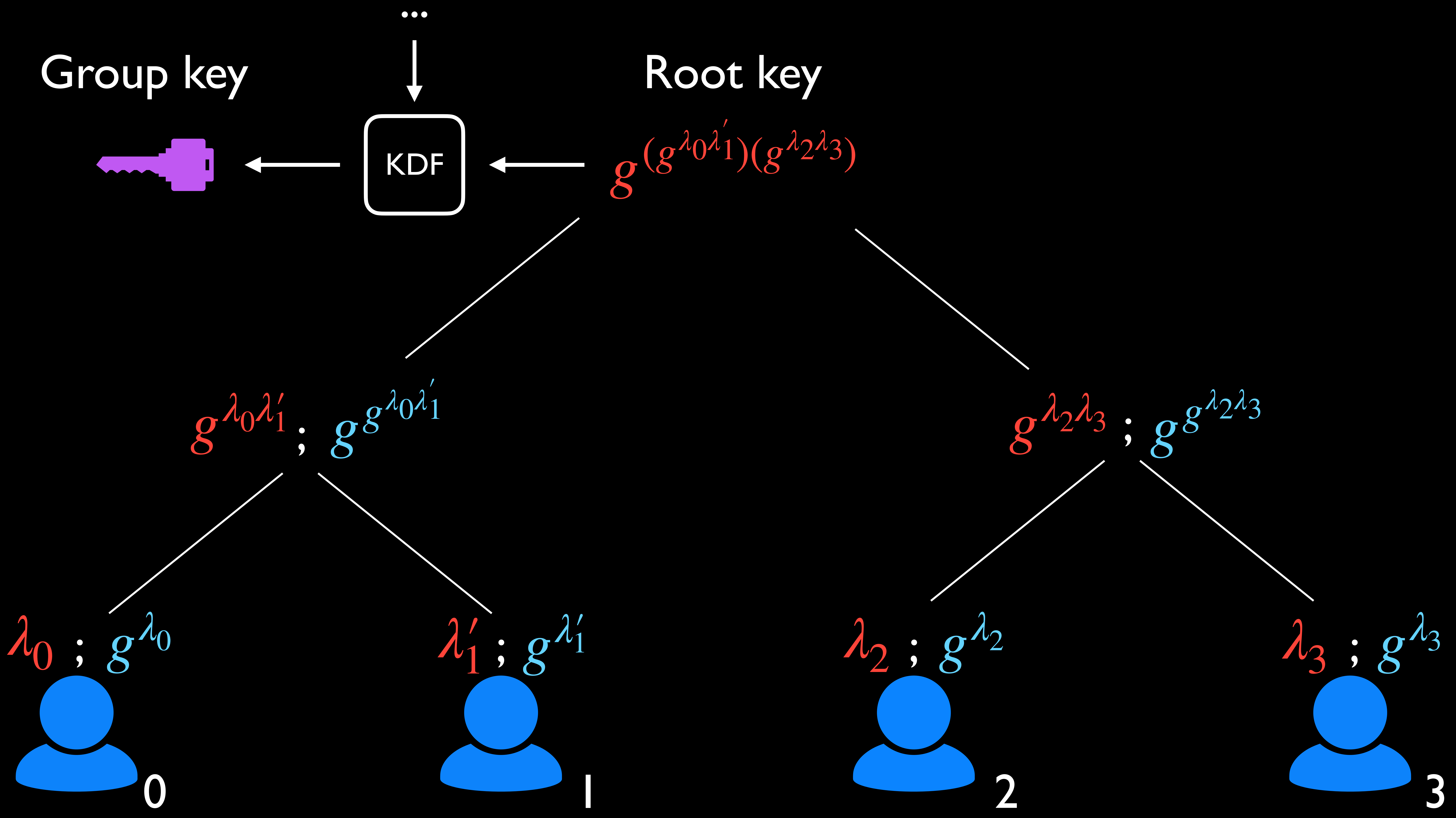

...
Group key
KDF
Root key
$g^{(g^{\lambda_0\lambda_1})(g^{\lambda_2\lambda_3})}$
$g^{\lambda_0\lambda_1}$; $g^{g^{\lambda_0\lambda_1}}$
$g^{\lambda_2\lambda_3}$; $g^{g^{\lambda_2\lambda_3}}$
Knows public keys on copath
Knows private key
λ_0 ; g^{λ_0}
λ_1 ; g^{λ_1}
λ_2 ; g^{λ_2}
λ_3 ; g^{λ_3}
0
1
2
3

Key Update



...
Group key
KDF
Root key
$g^{(g^{\lambda_0\lambda_1})(g^{\lambda_2\lambda_3})}$
$g^{\lambda_0\lambda_1}$; $g^{g^{\lambda_0\lambda_1}}$
$g^{\lambda_2\lambda_3}$; $g^{g^{\lambda_2\lambda_3}}$
λ_0 ; g^{λ_0}
λ_1' ; $g^{\lambda_1'}$
λ_2 ; g^{λ_2}
λ_3 ; g^{λ_3}
0
1
2
3

Key Update



...

Group key

KDF

Root key

$g^{(g^{\lambda_0\lambda_1'})(g^{\lambda_2\lambda_3})}$

$g^{\lambda_0\lambda_1'}$; $g^{g^{\lambda_0\lambda_1'}}$

$g^{\lambda_2\lambda_3}$; $g^{g^{\lambda_2\lambda_3}}$

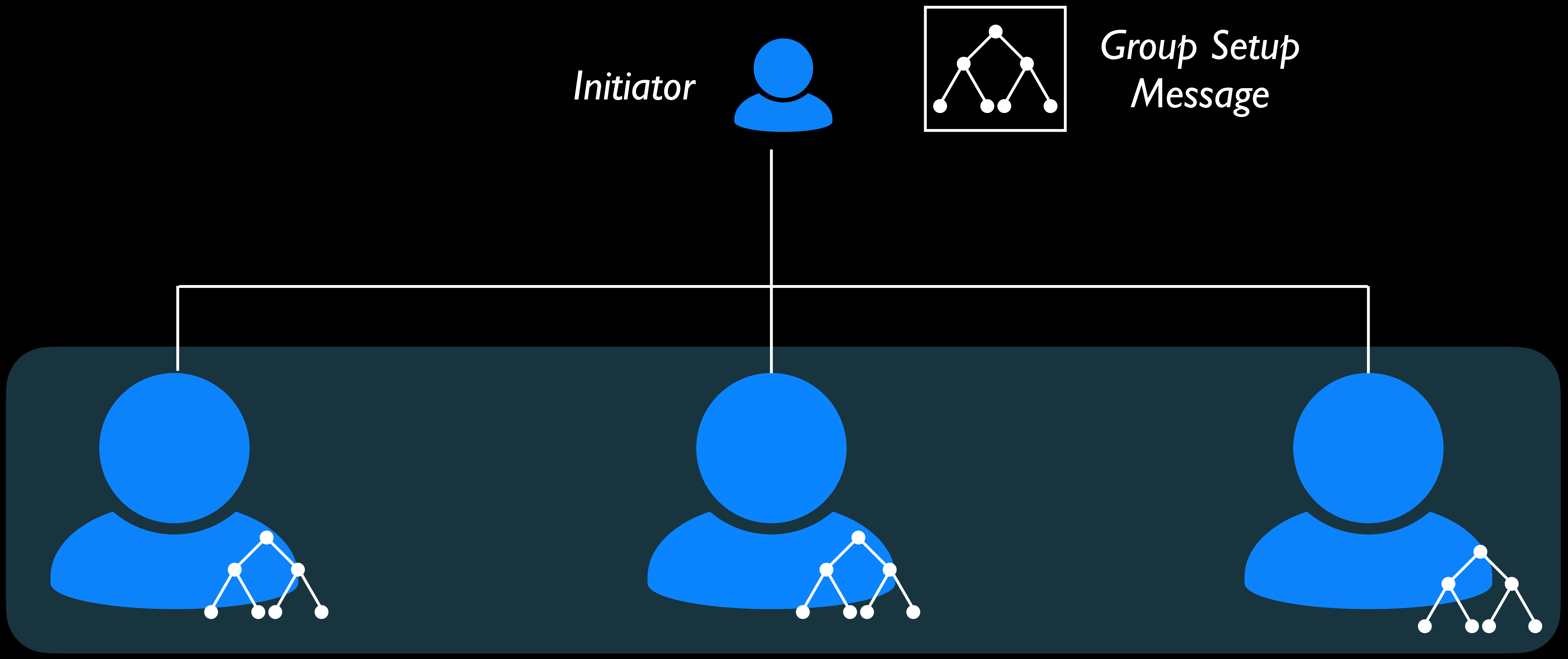
λ_0 ; g^{λ_0}

λ_1' ; $g^{\lambda_1'}$

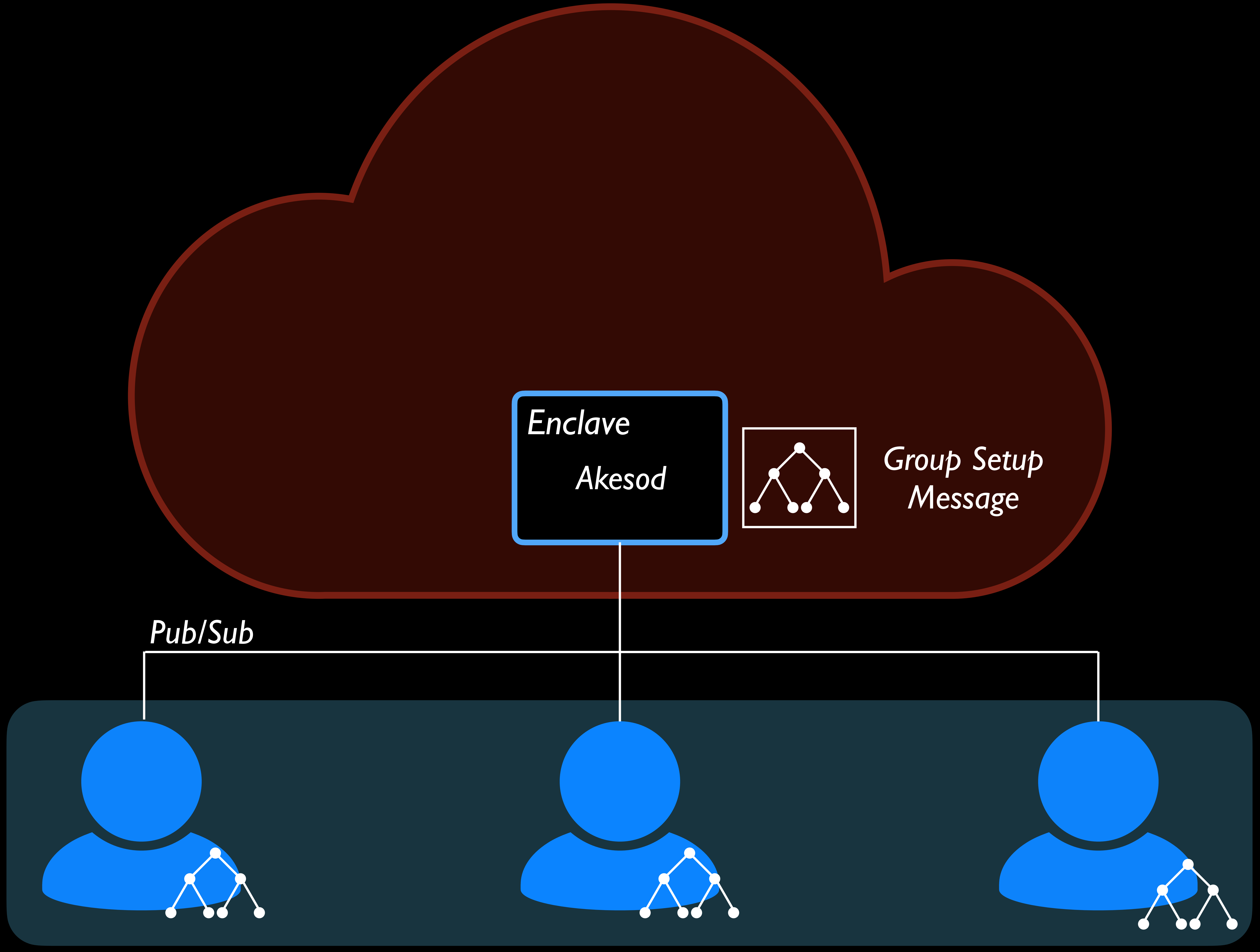
λ_2 ; g^{λ_2}

λ_3 ; g^{λ_3}

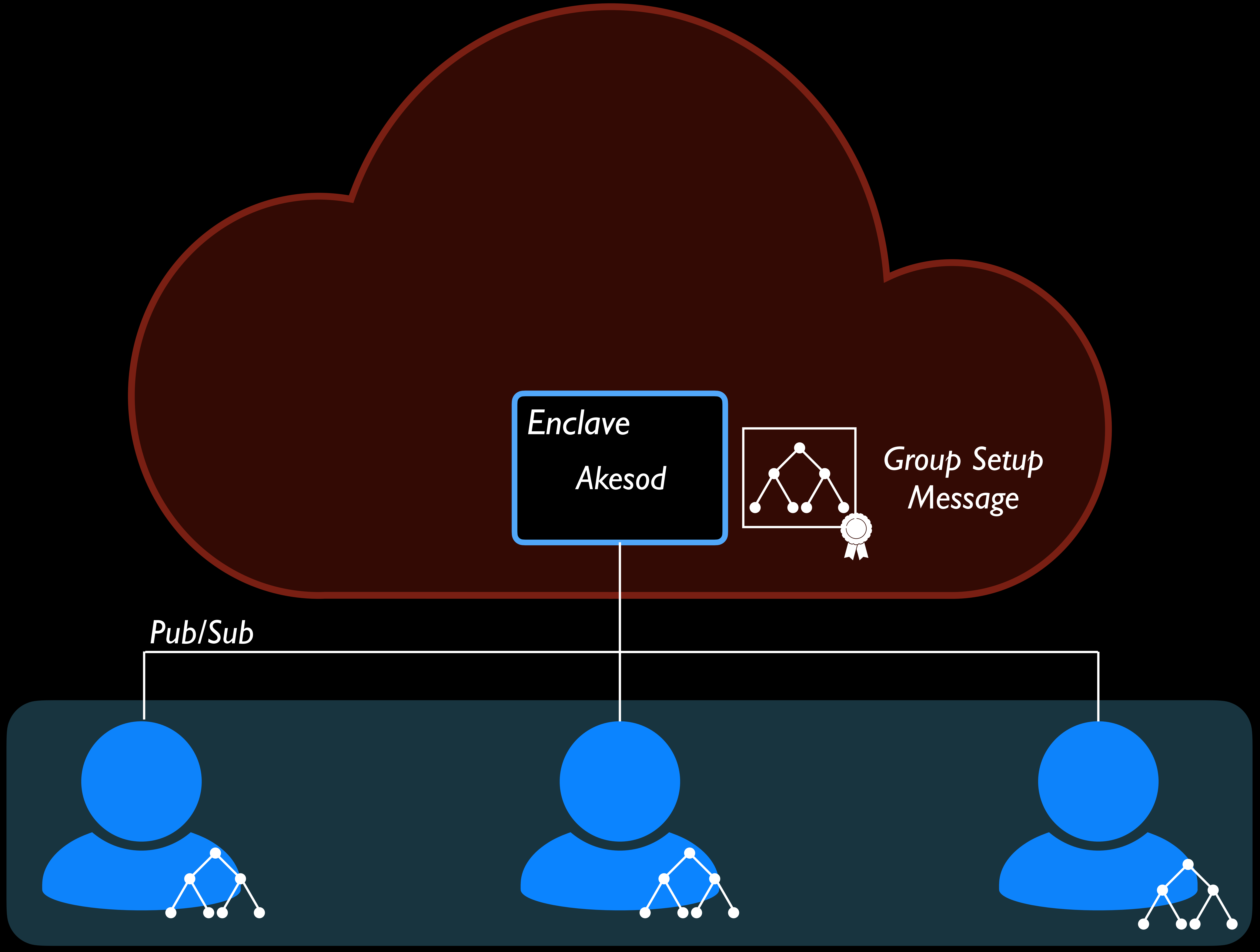
Tree Setup



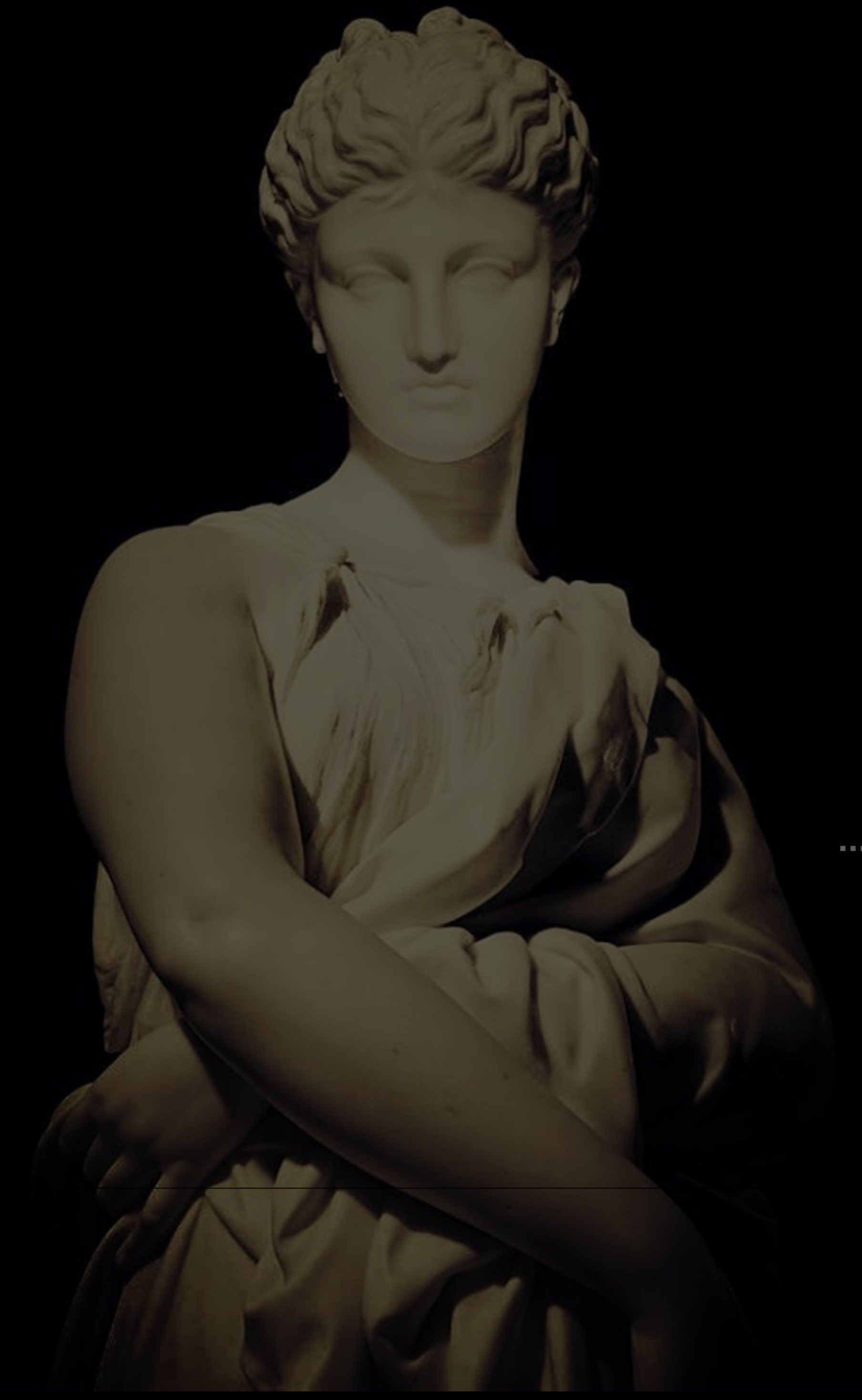
Tree Setup



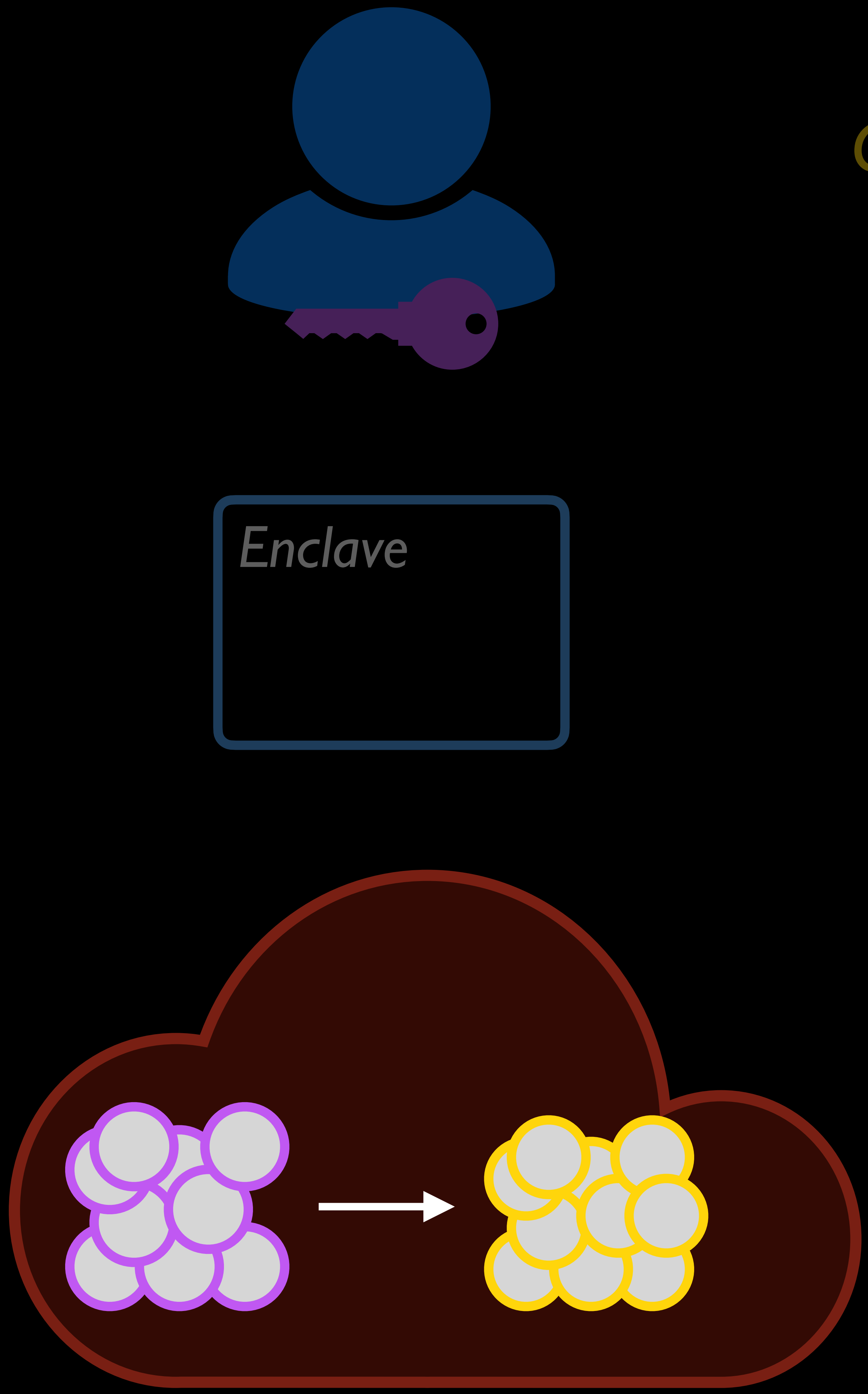
Tree Setup



Akeso



Akeso - Greek goddess of well-being and healing

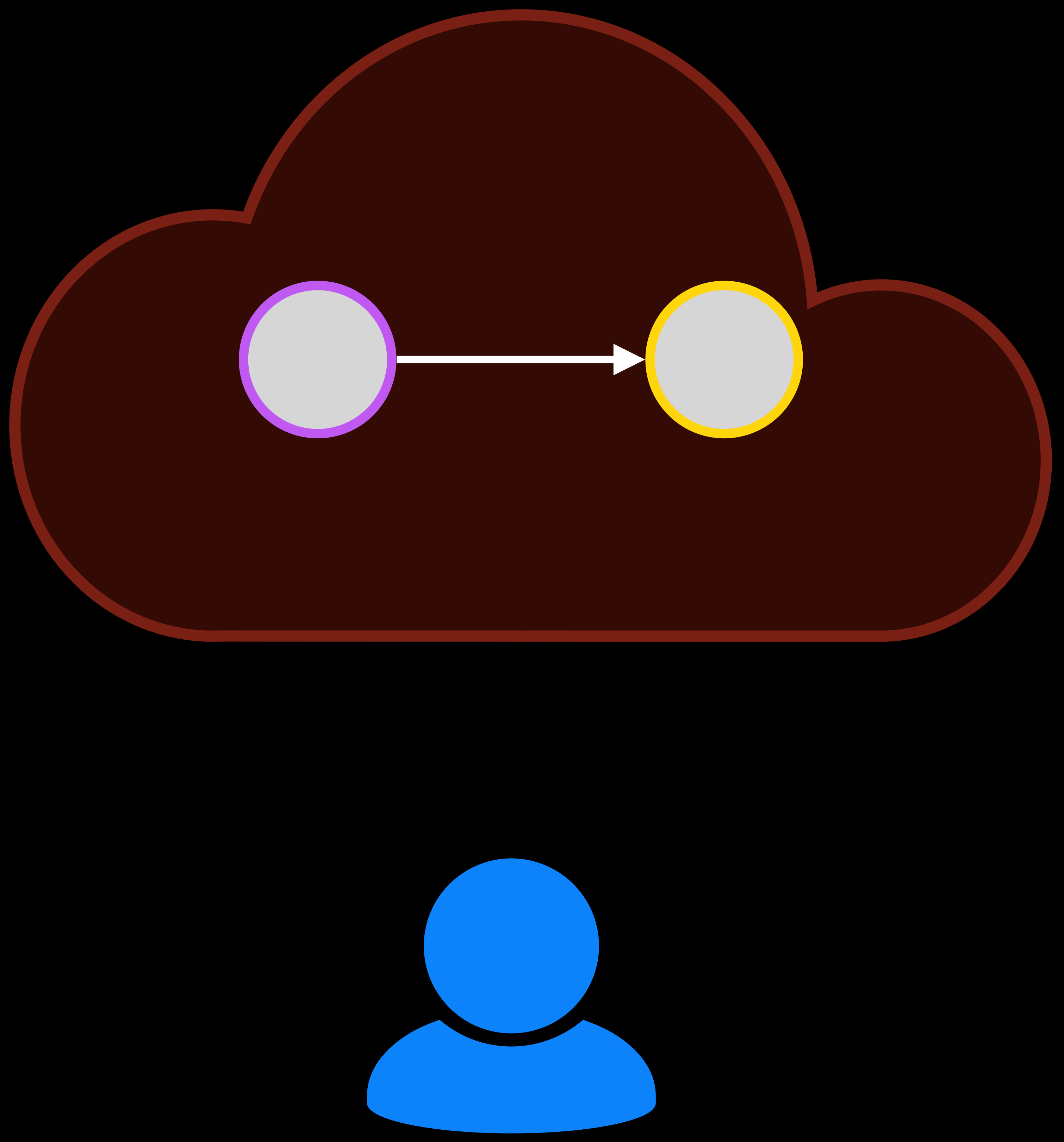


Client-side encryption and efficient key rotation

Minimal use of TEEs

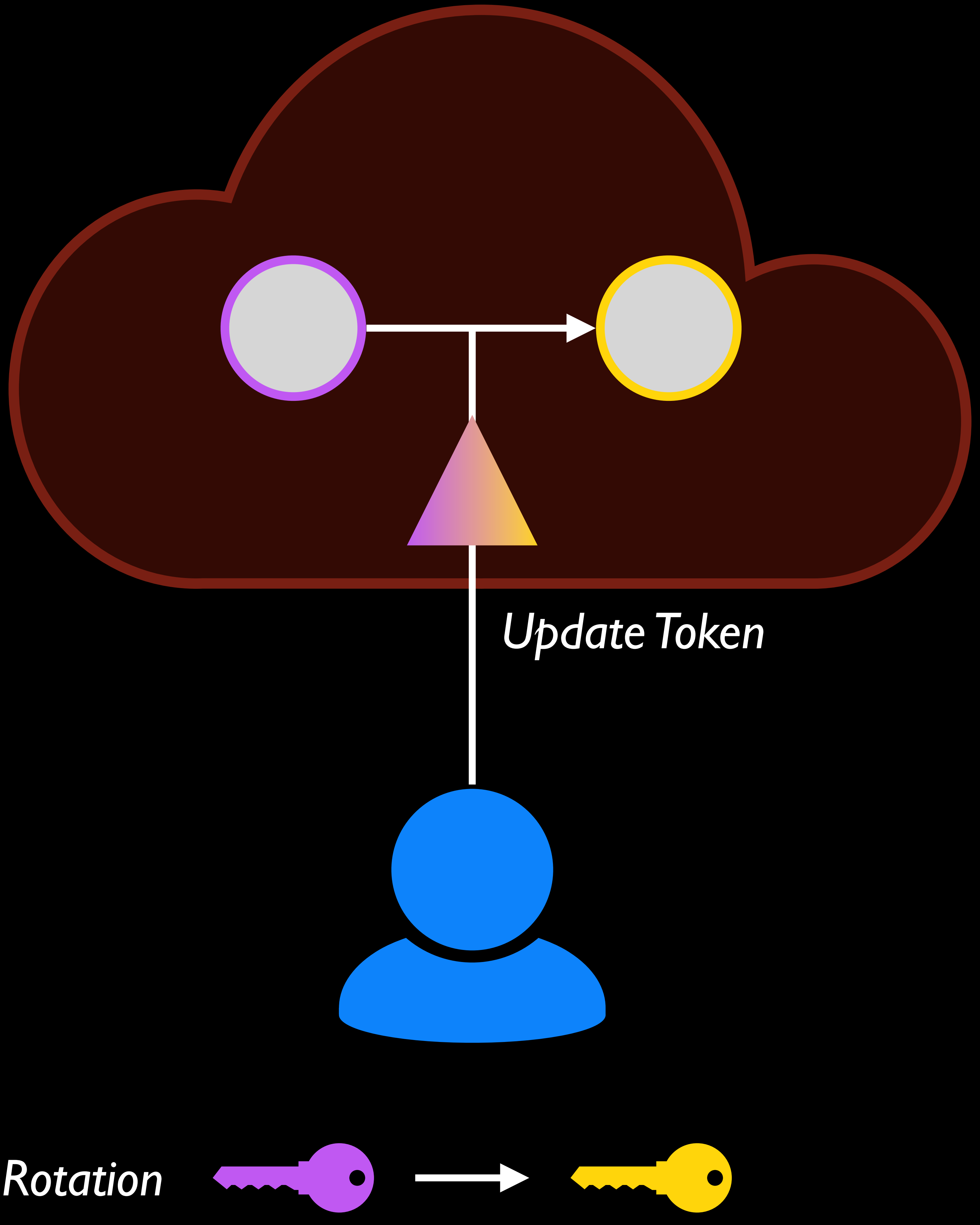
Updatable Encryption

Updatable Encryption

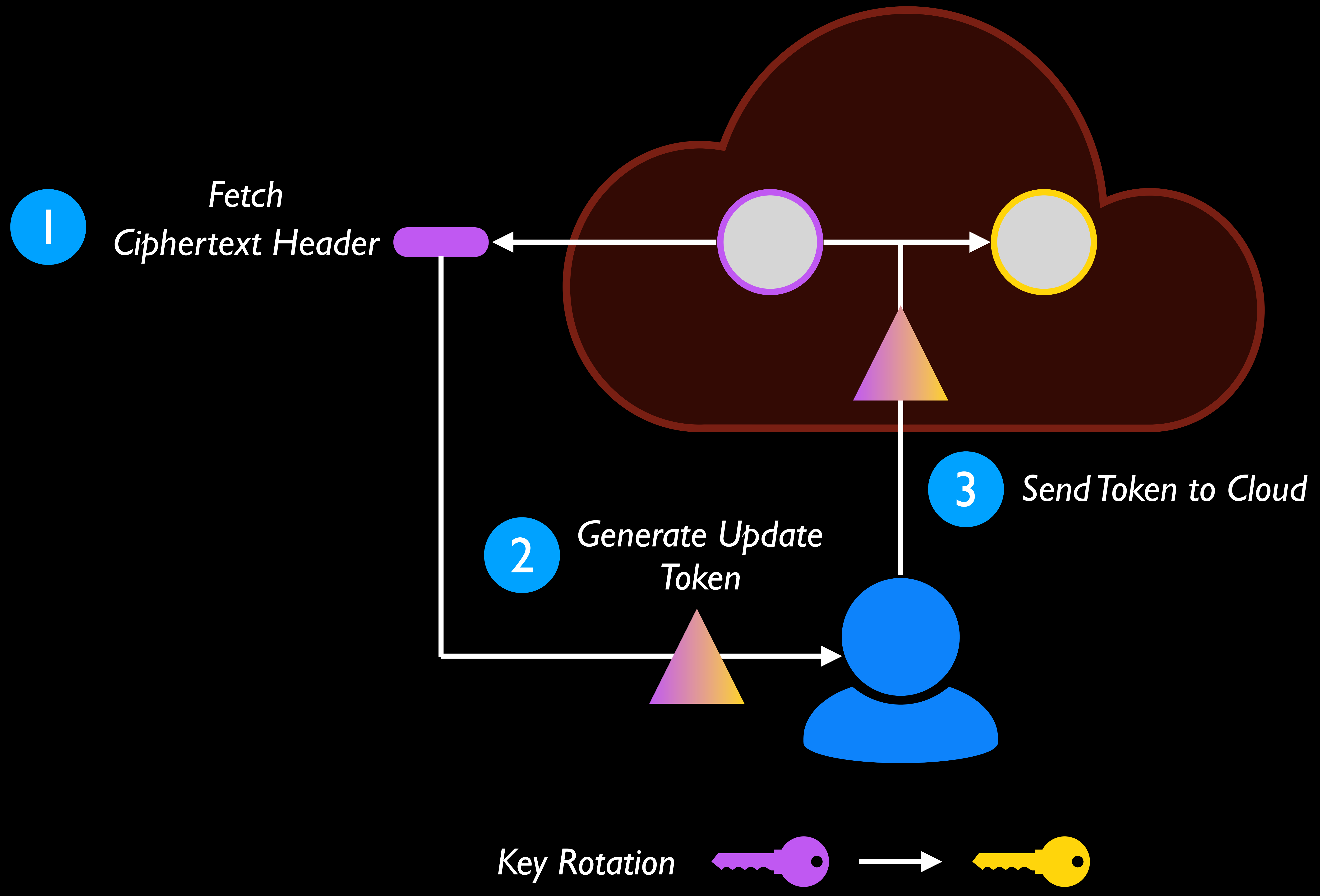


Key Rotation 🔑 → 🔑

Updatable Encryption



Ciphertext-Dependent Updatable Encryption



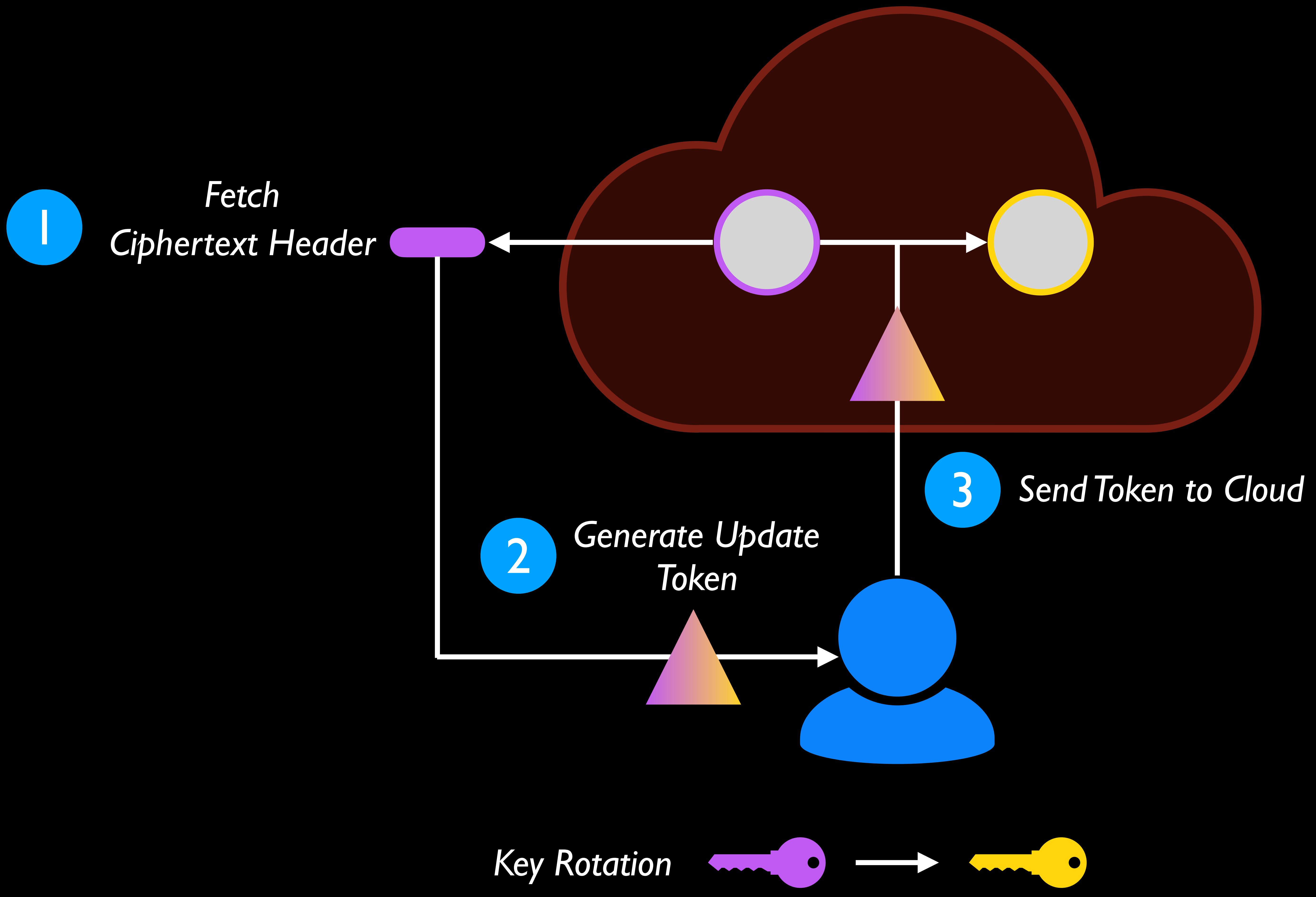
1 *Fetch Ciphertext Header*

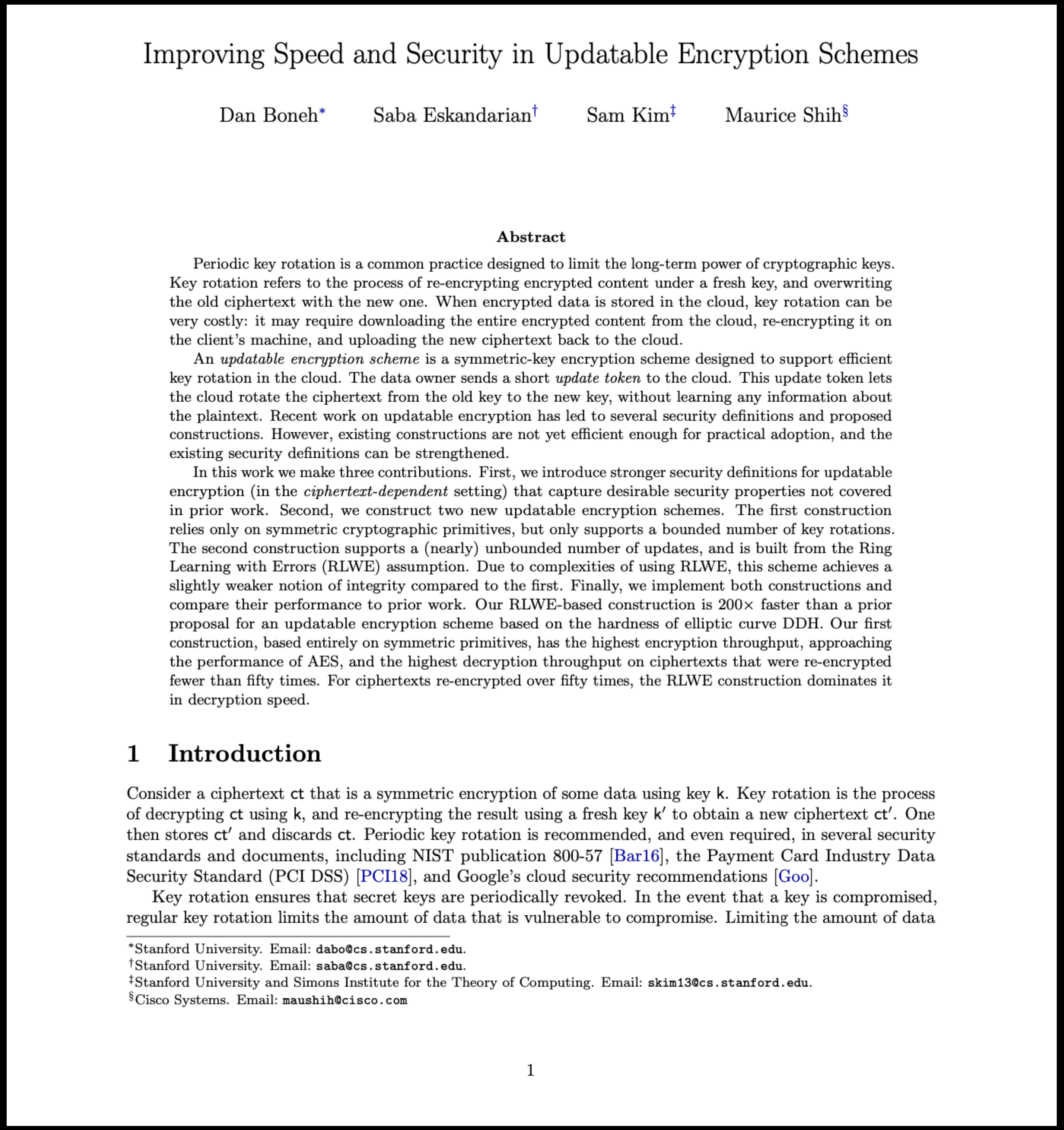
2 *Generate Update Token*

3 *Send Token to Cloud*

Key Rotation

Ciphertext-Dependent Updatable Encryption





Improving Speed and Security in Updatable Encryption Schemes

Dan Boneh[*] Saba Eskandarian[†] Sam Kim[‡] Maurice Shih[§]

Abstract

Periodic key rotation is a common practice designed to limit the long-term power of cryptographic keys. Key rotation refers to the process of re-encrypting encrypted content under a fresh key, and overwriting the old ciphertext with the new one. When encrypted data is stored in the cloud, key rotation can be very costly: it may require downloading the entire encrypted content from the cloud, re-encrypting it on the client's machine, and uploading the new ciphertext back to the cloud.

An *updatable encryption scheme* is a symmetric-key encryption scheme designed to support efficient key rotation in the cloud. The data owner sends a short *update token* to the cloud. This update token lets the cloud rotate the ciphertext from the old key to the new key, without learning any information about the plaintext. Recent work on updatable encryption has led to several security definitions and proposed constructions. However, existing constructions are not yet efficient enough for practical adoption, and the existing security definitions can be strengthened.

In this work we make three contributions. First, we introduce stronger security definitions for updatable encryption (in the *ciphertext-dependent* setting) that capture desirable security properties not covered in prior work. Second, we construct two new updatable encryption schemes. The first construction relies only on symmetric cryptographic primitives, but only supports a bounded number of key rotations. The second construction supports a (nearly) unbounded number of updates, and is built from the Ring Learning with Errors (RLWE) assumption. Due to complexities of using RLWE, this scheme achieves a slightly weaker notion of integrity compared to the first. Finally, we implement both constructions and compare their performance to prior work. Our RLWE-based construction is 200× faster than a prior proposal for an updatable encryption scheme based on the hardness of elliptic curve DDH. Our first construction, based entirely on symmetric primitives, has the highest encryption throughput, approaching the performance of AES, and the highest decryption throughput on ciphertexts that were re-encrypted fewer than fifty times. For ciphertexts re-encrypted over fifty times, the RLWE construction dominates it in decryption speed.

1 Introduction

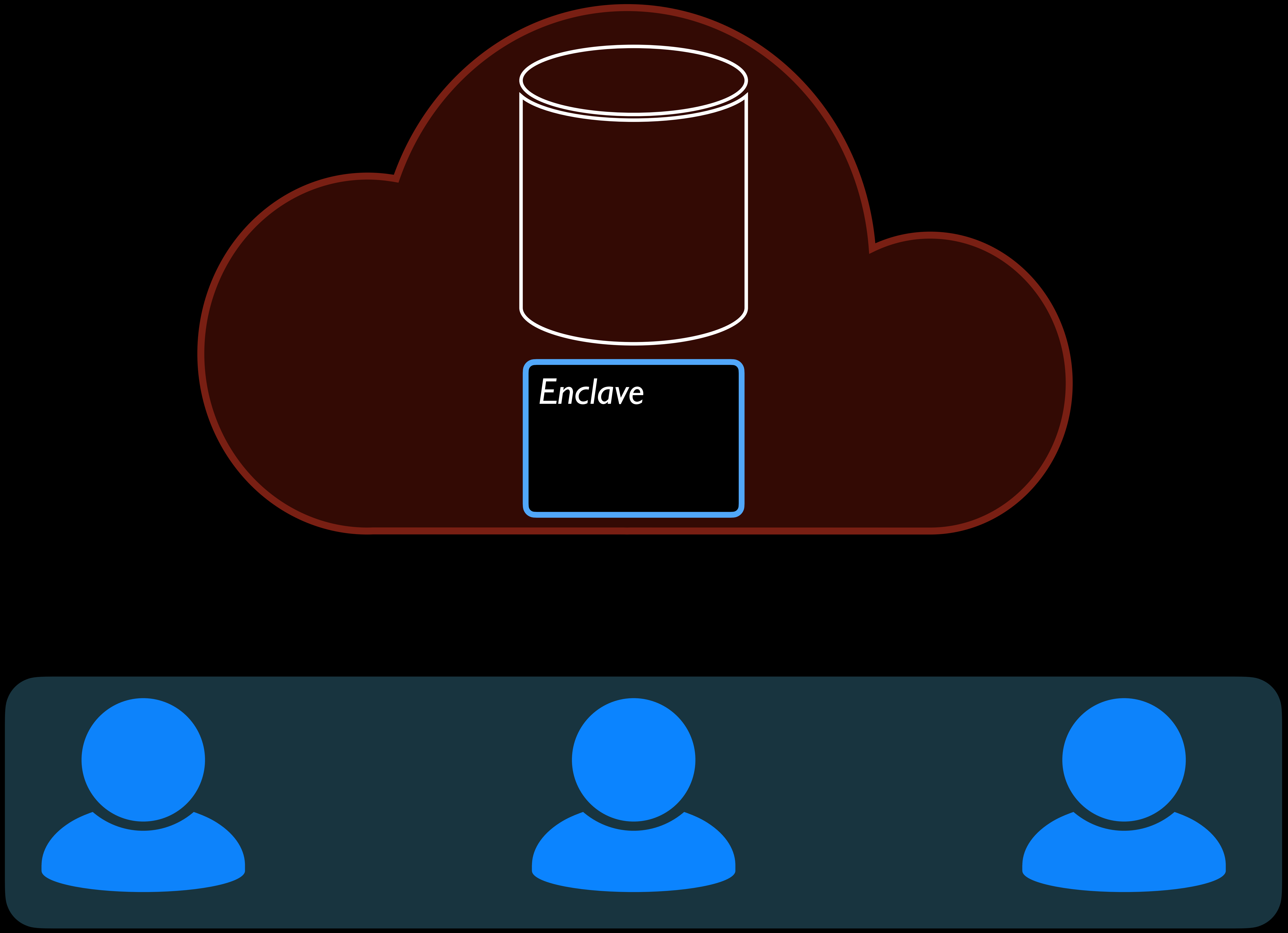
Consider a ciphertext ct that is a symmetric encryption of some data using key k. Key rotation is the process of decrypting ct using k, and re-encrypting the result using a fresh key k' to obtain a new ciphertext ct'. One then stores ct' and discards ct. Periodic key rotation is recommended, and even required, in several security standards and documents, including NIST publication 800-57 [Bar16], the Payment Card Industry Data Security Standard (PCI DSS) [PCI18], and Google's cloud security recommendations [Goo].

Key rotation ensures that secret keys are periodically revoked. In the event that a key is compromised, regular key rotation limits the amount of data that is vulnerable to compromise. Limiting the amount of data

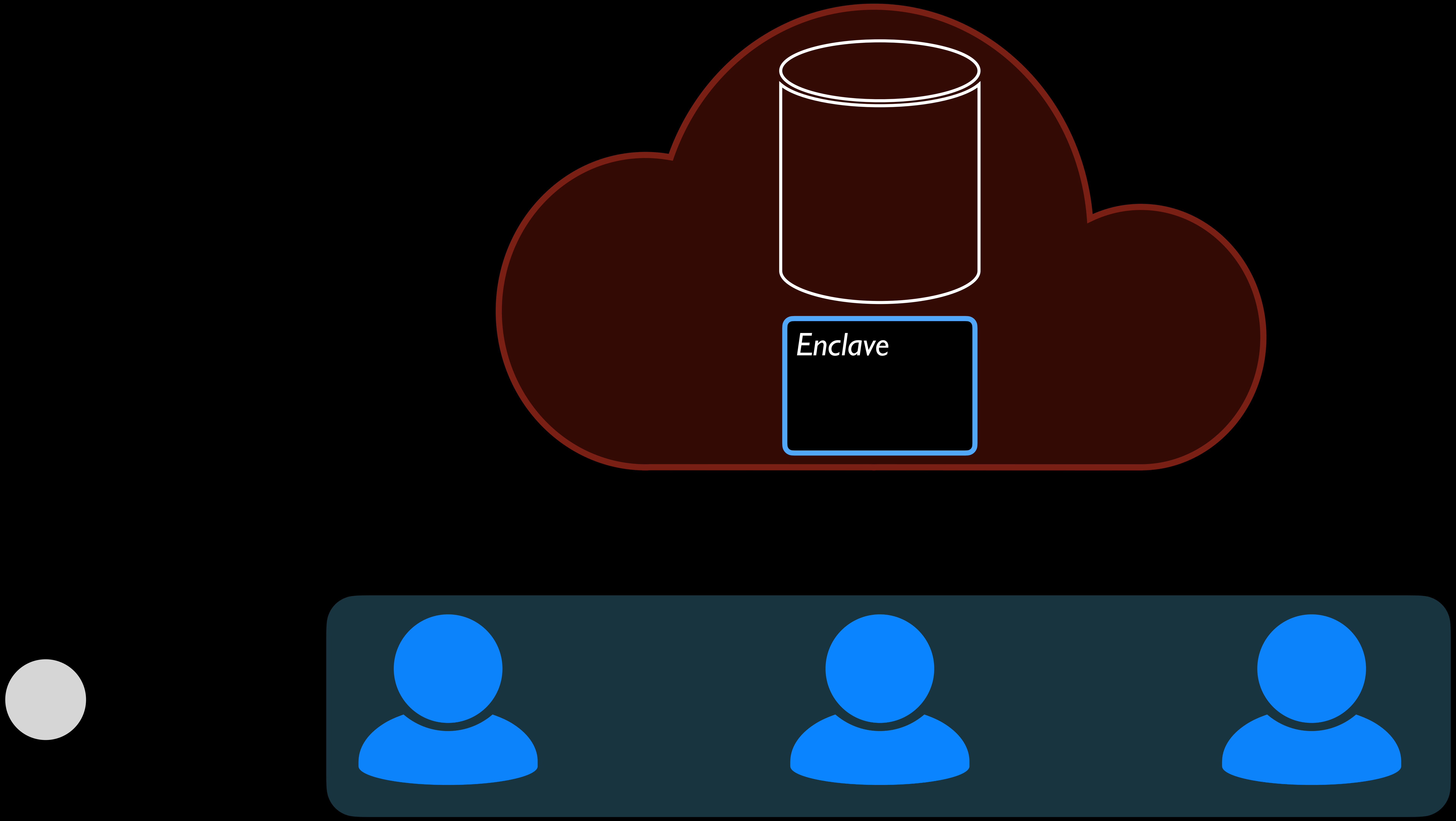
[*] Stanford University. Email: dabo@cs.stanford.edu.
[†] Stanford University. Email: saba@cs.stanford.edu.
[‡] Stanford University and Simons Institute for the Theory of Computing. Email: skim13@cs.stanford.edu.
[§] Cisco Systems. Email: maushih@cisco.com

Boneh et al., ASIACRYPT 2020

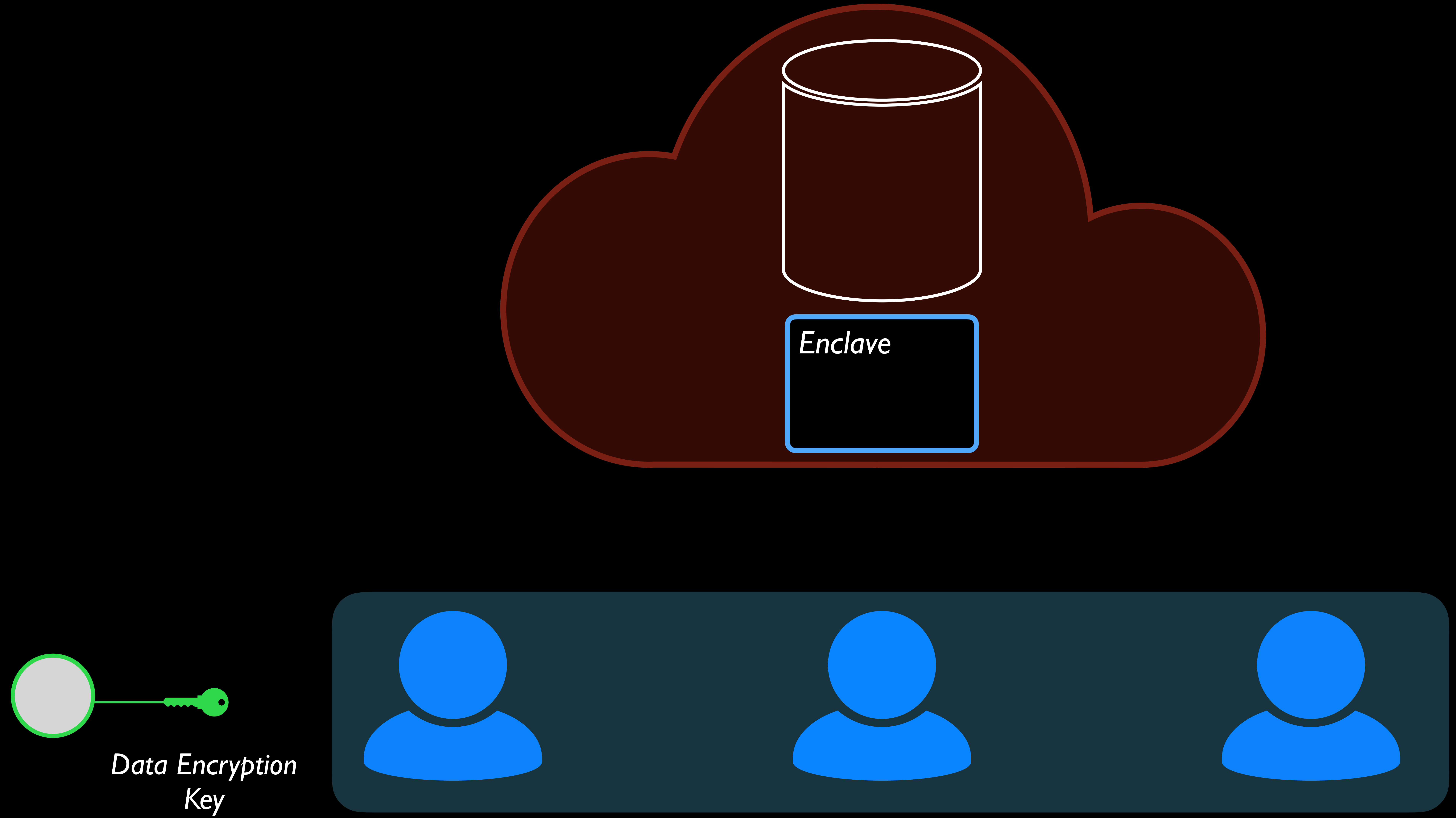
Nested AES Updatable Encryption



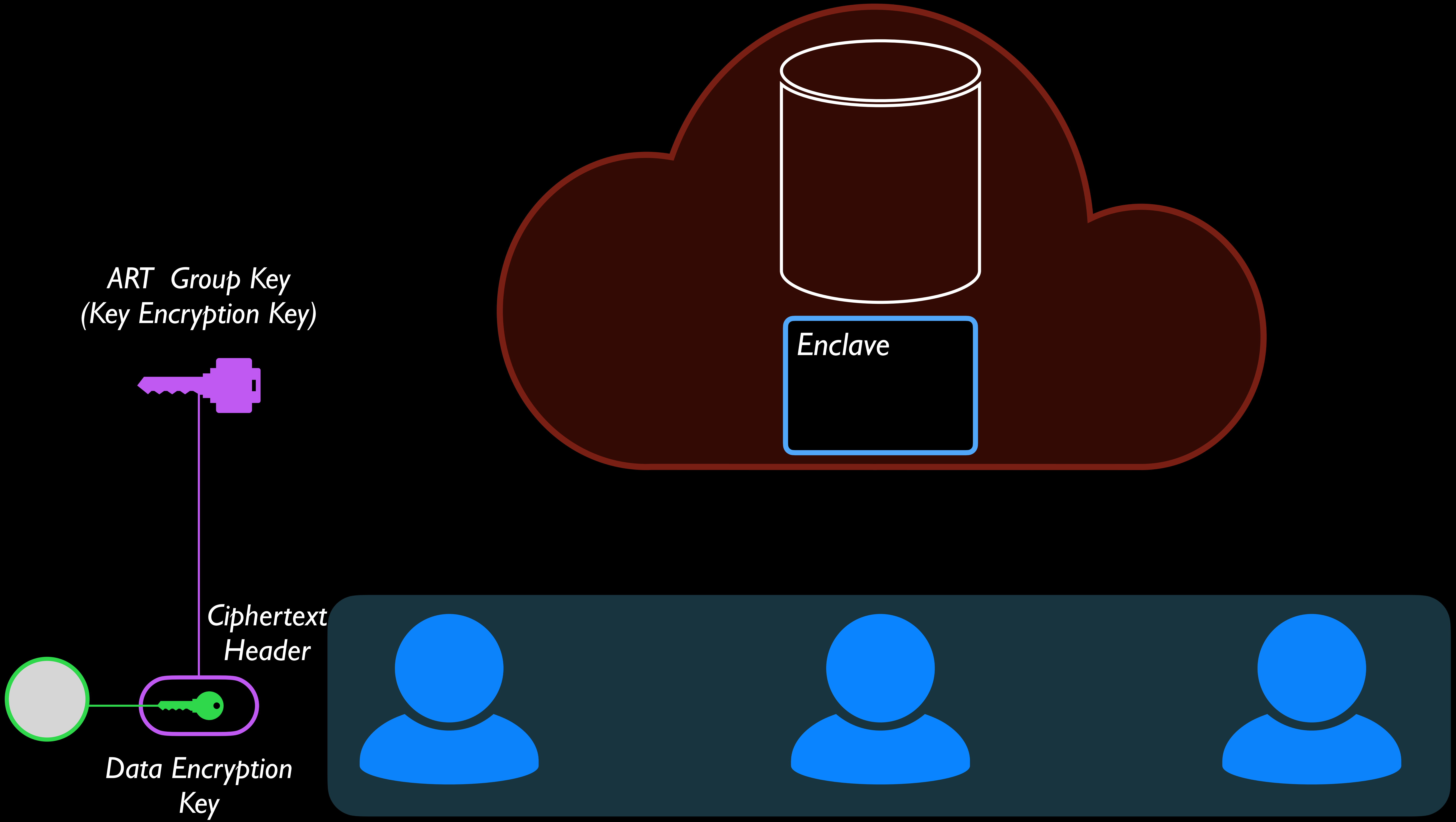
Nested AES Updatable Encryption



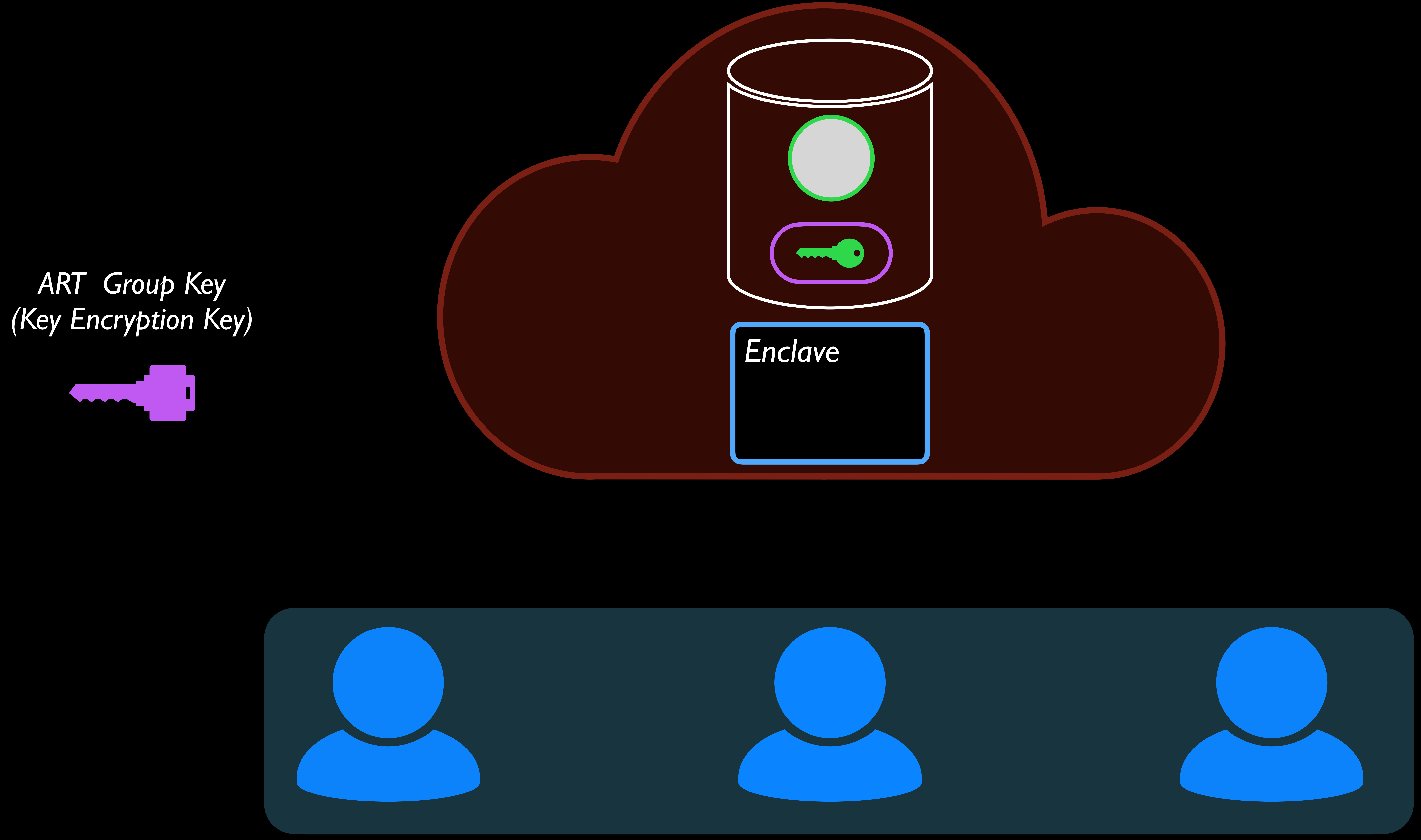
Nested AES Updatable Encryption



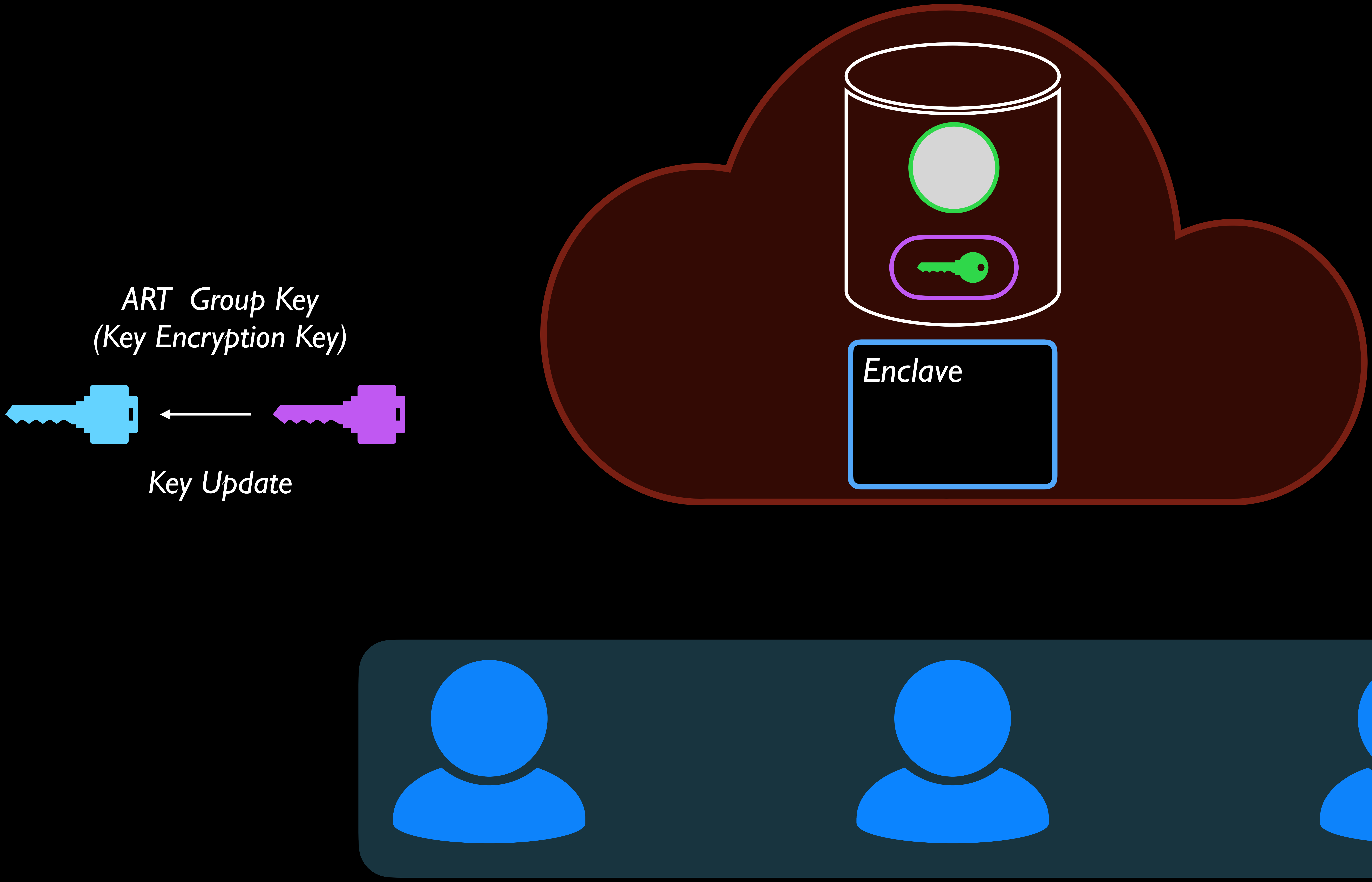
Nested AES Updatable Encryption



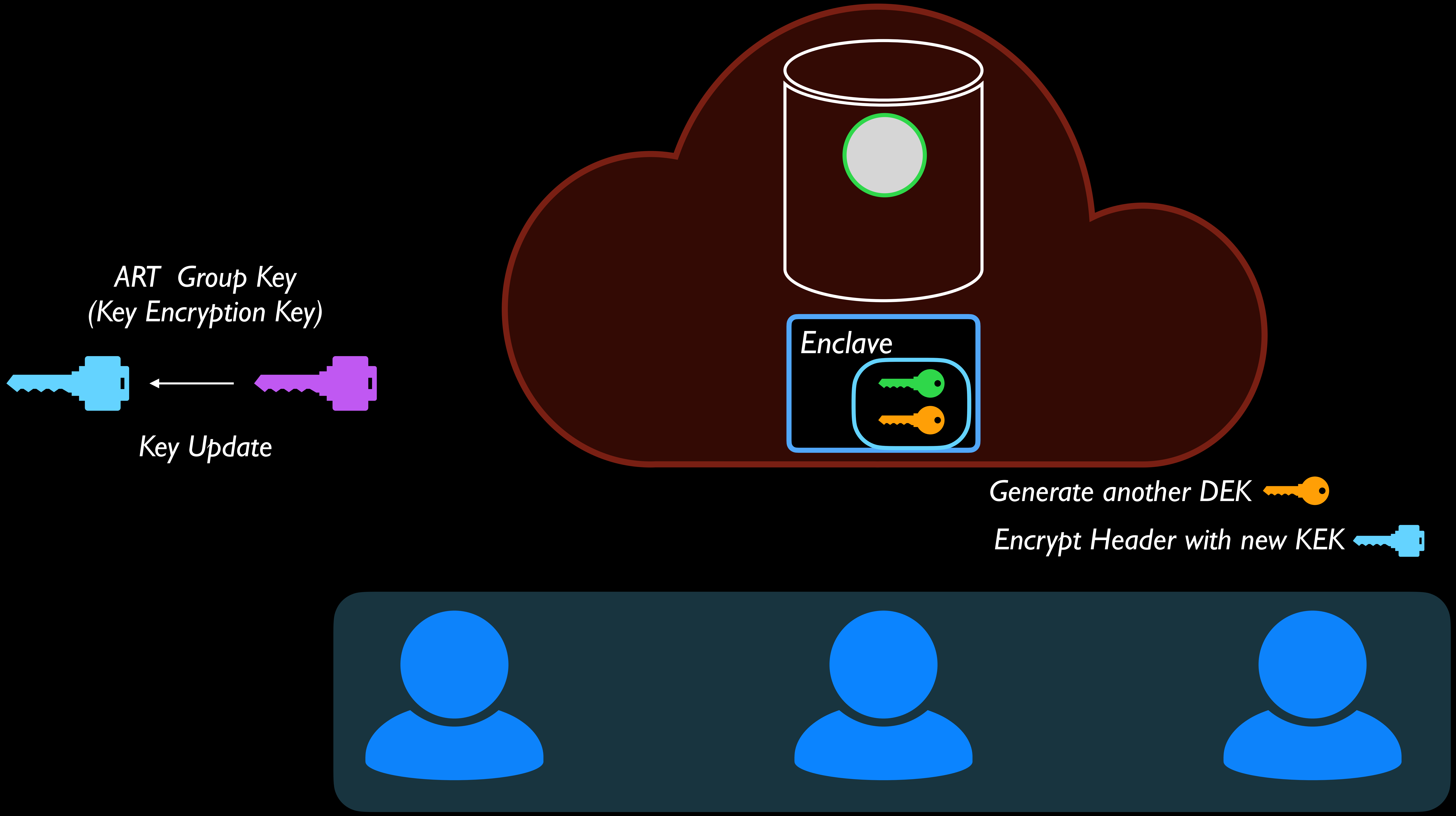
Nested AES Updatable Encryption



Nested AES Updatable Encryption



Nested AES Updatable Encryption



ART Group Key
(Key Encryption Key)

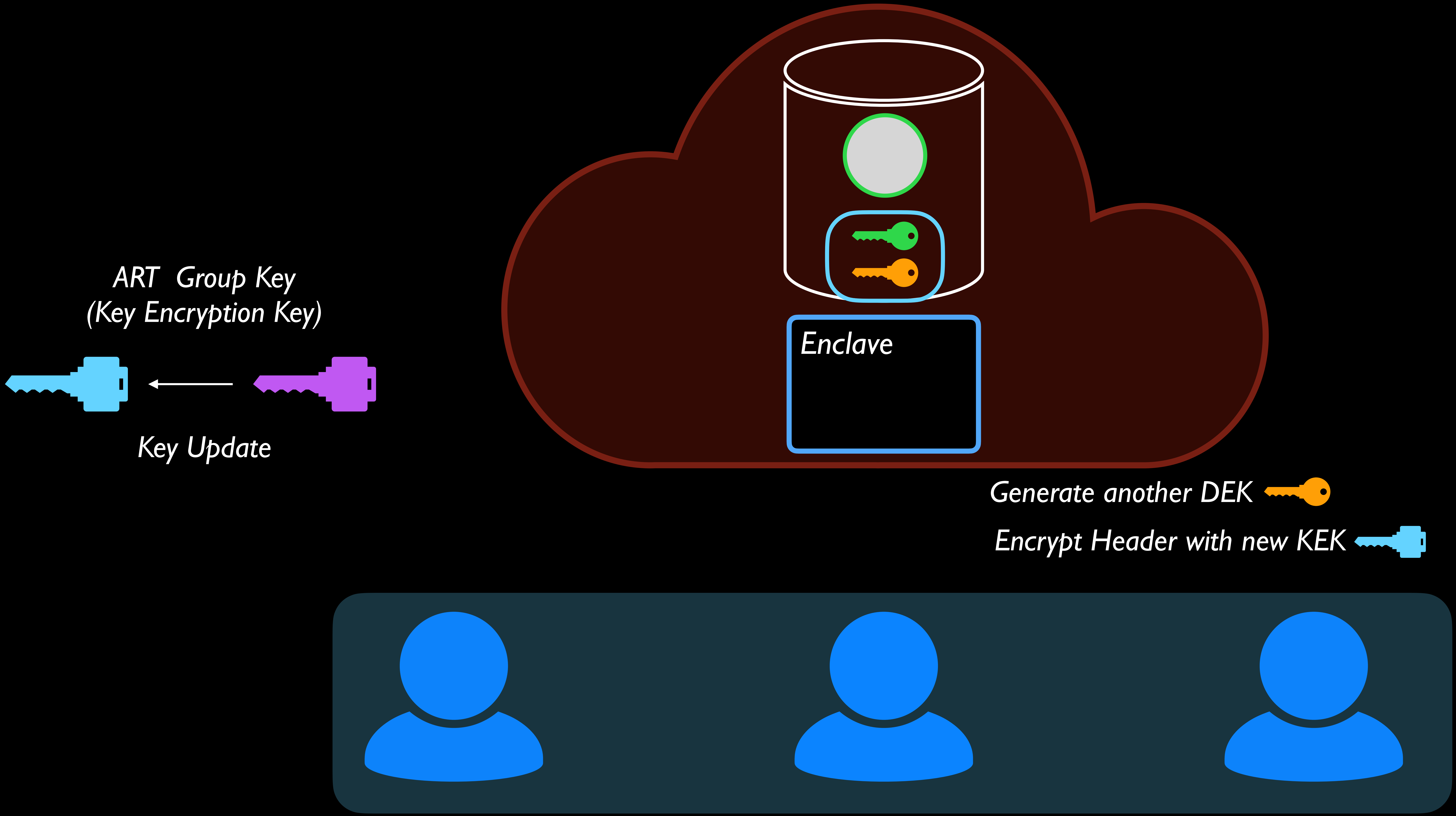
Key Update

Enclave

Generate another DEK

Encrypt Header with new KEK

Nested AES Updatable Encryption



ART Group Key
(Key Encryption Key)

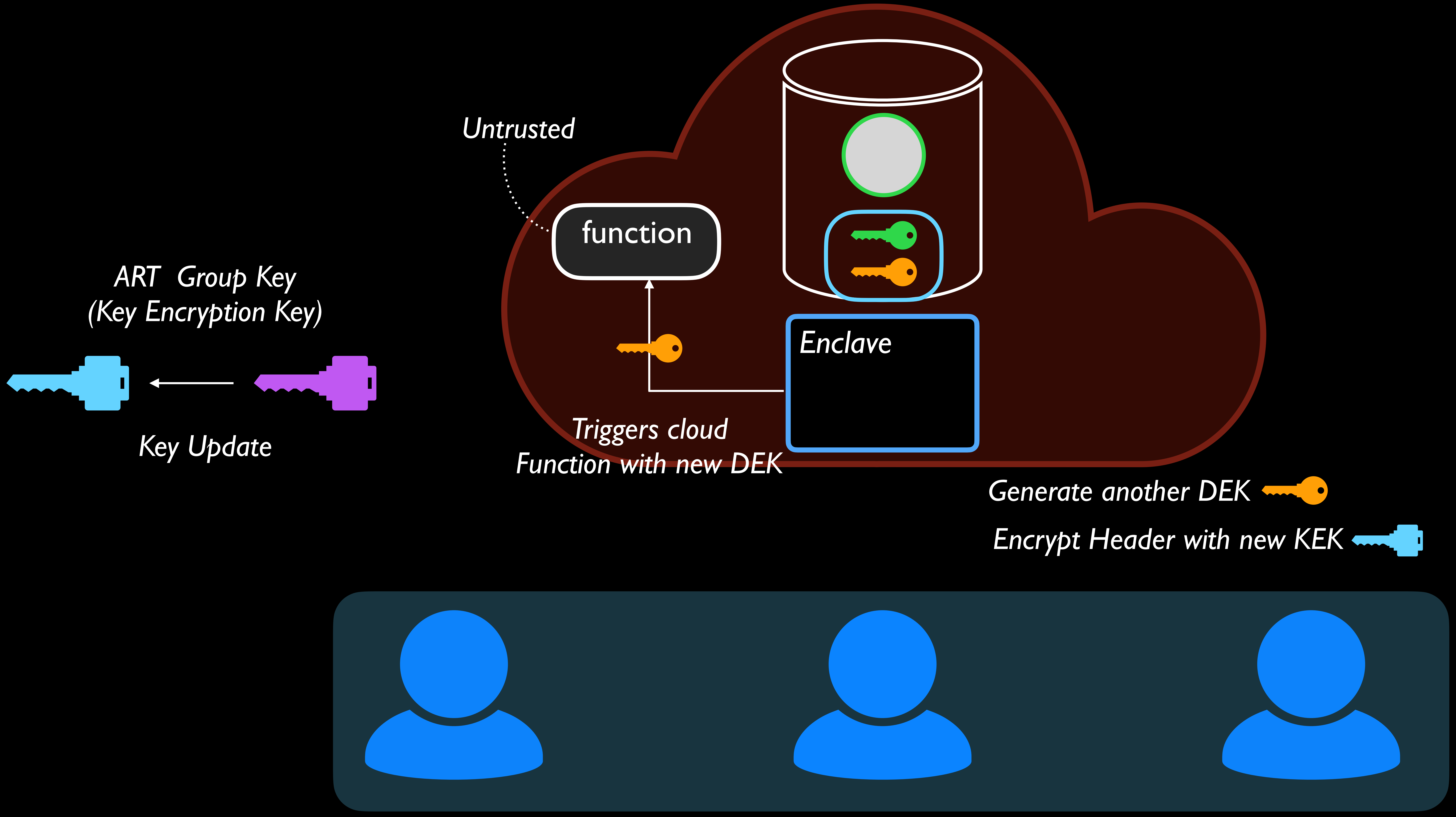
Key Update

Enclave

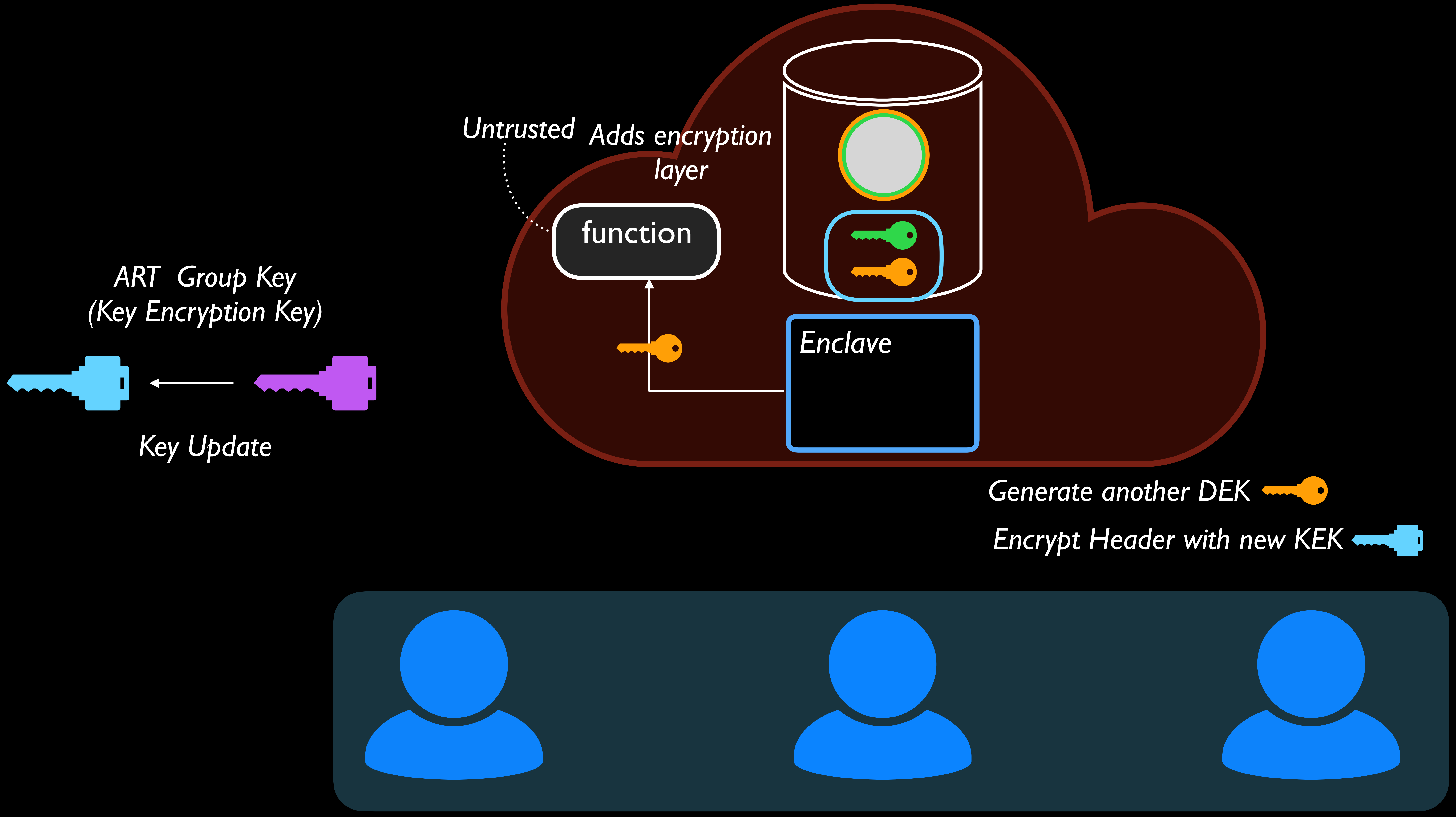
Generate another DEK

Encrypt Header with new KEK

Nested AES Updatable Encryption



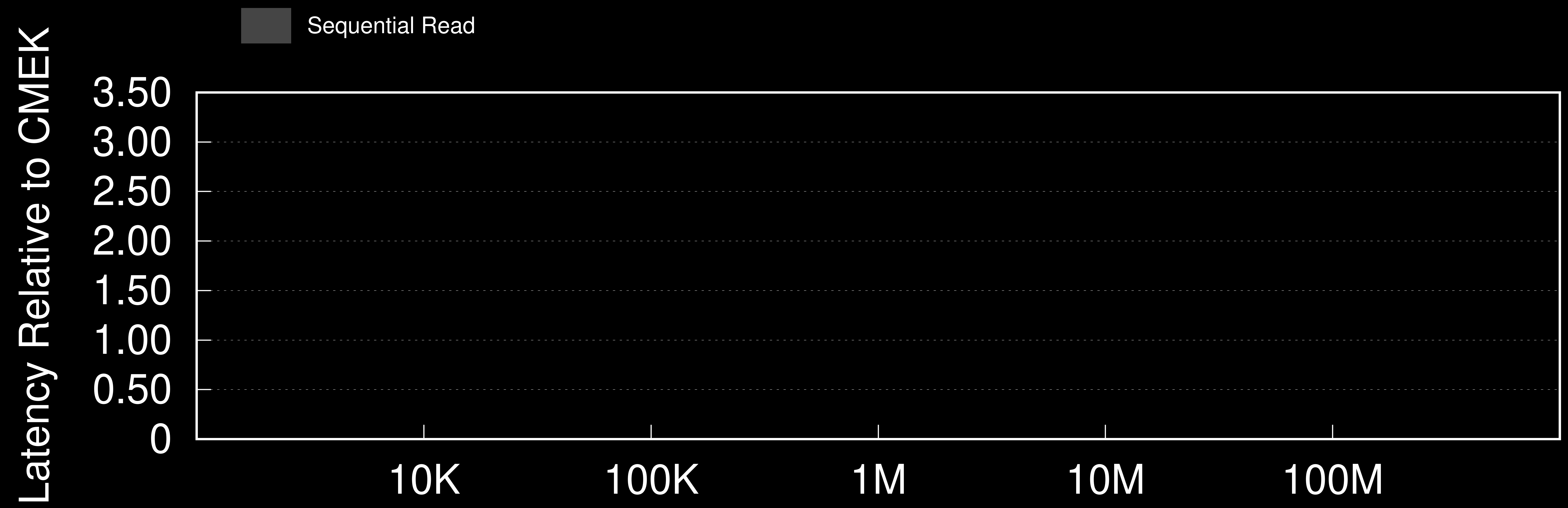
Nested AES Updatable Encryption



Latency to Read/Write an Object

Integrated Akeso into GCSFuse

(Benchmark uses a single layer of encryption)



Object Size

Real-World Object Size Percentiles

50th

90th

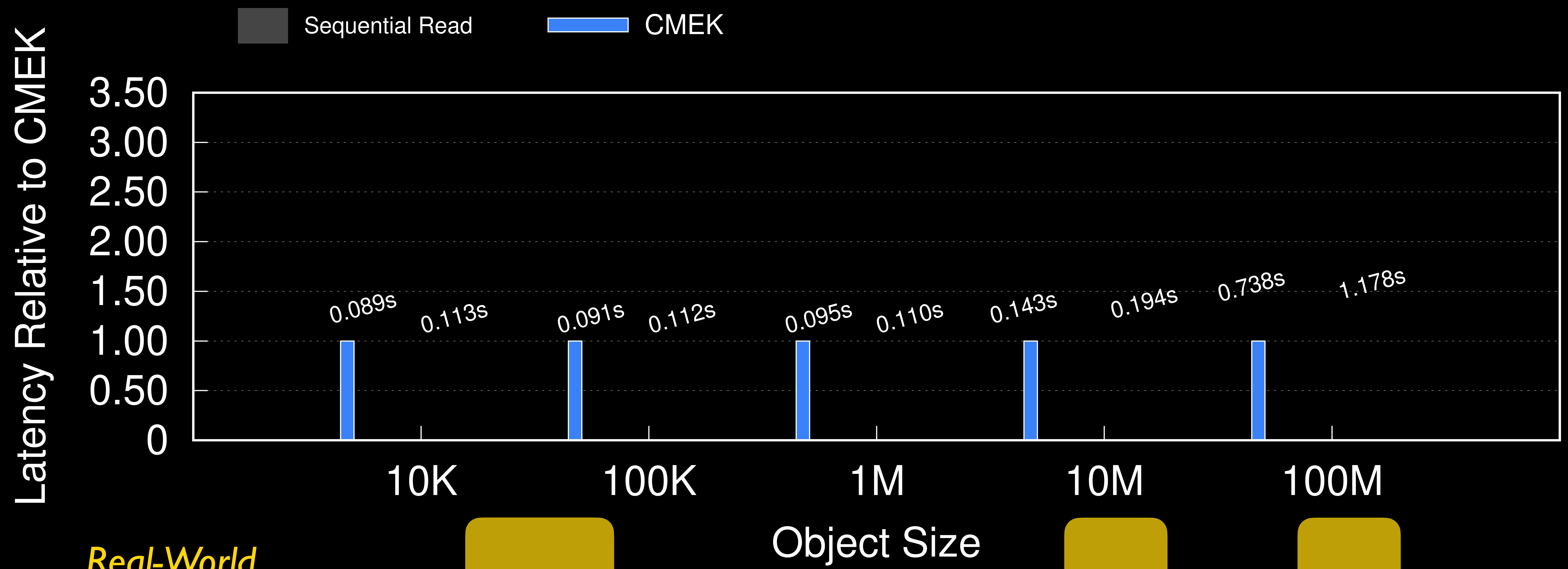
95th

99th

Latency to Read/Write an Object

Integrated Akeso into GCSFuse

(Benchmark uses a single layer of encryption)



Real-World Object Size Percentiles

50th

90th

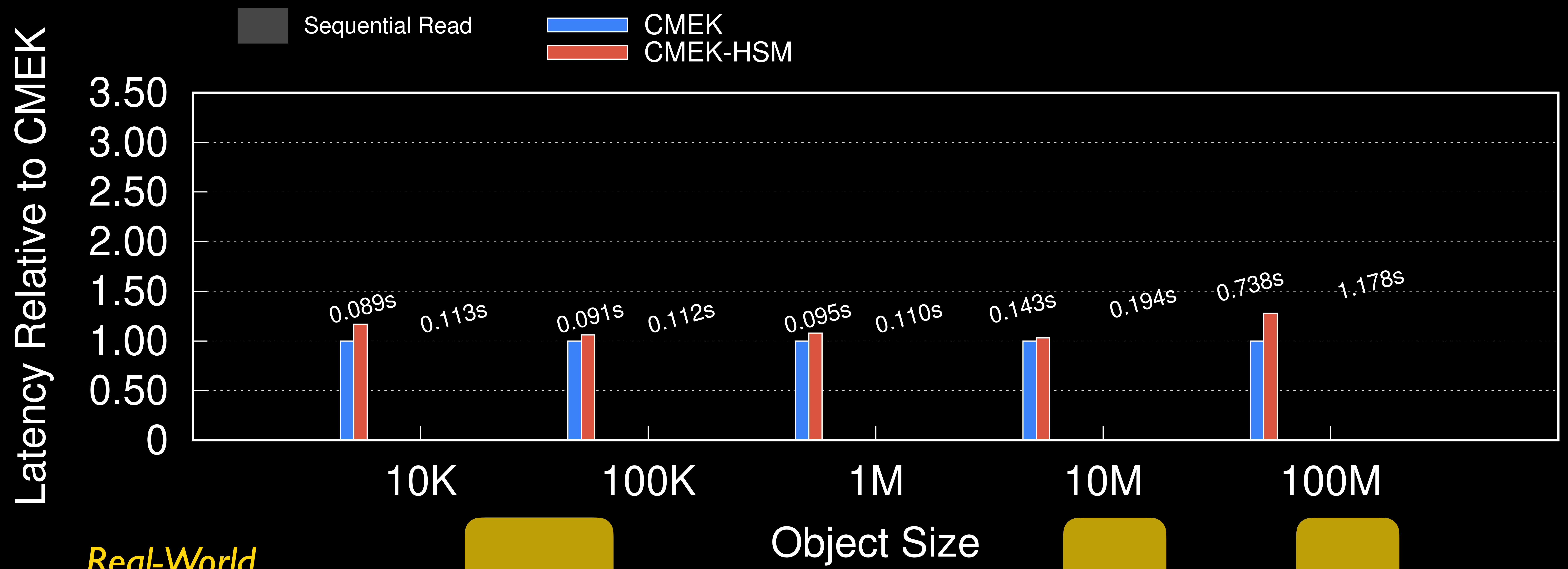
95th

99th

Latency to Read/Write an Object

Integrated Akeso into GCSFuse

(Benchmark uses a single layer of encryption)



Real-World Object Size Percentiles

50th

90th

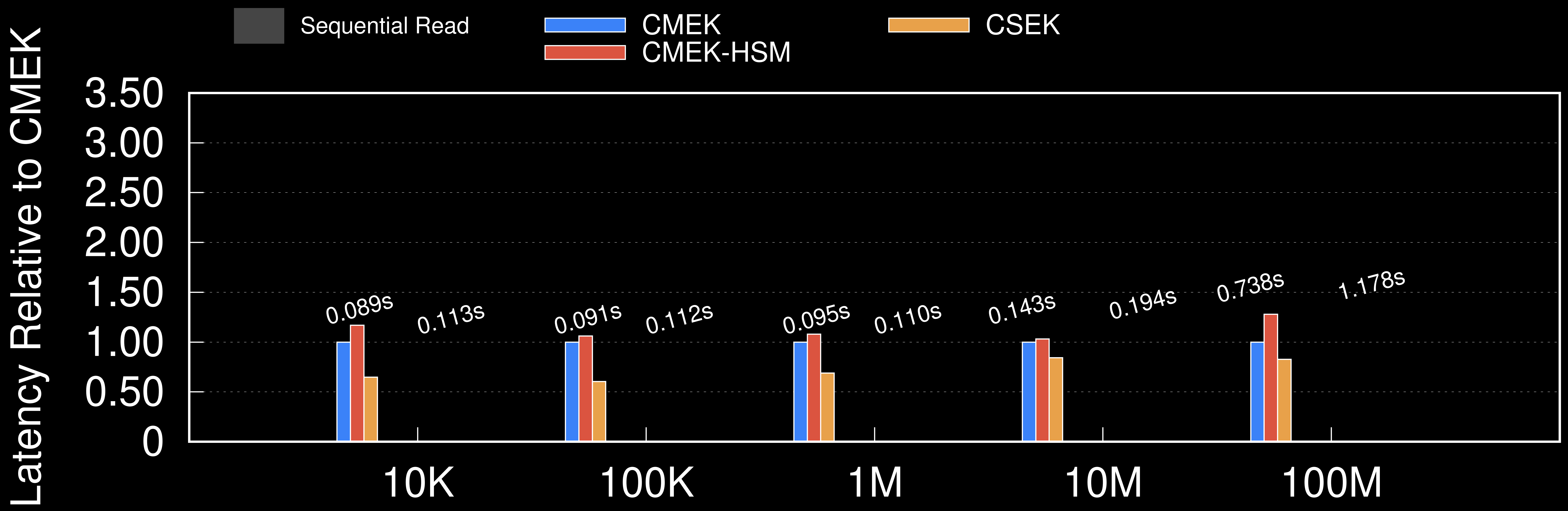
95th

99th

Latency to Read/Write an Object

Integrated Akeso into GCSFuse

(Benchmark uses a single layer of encryption)



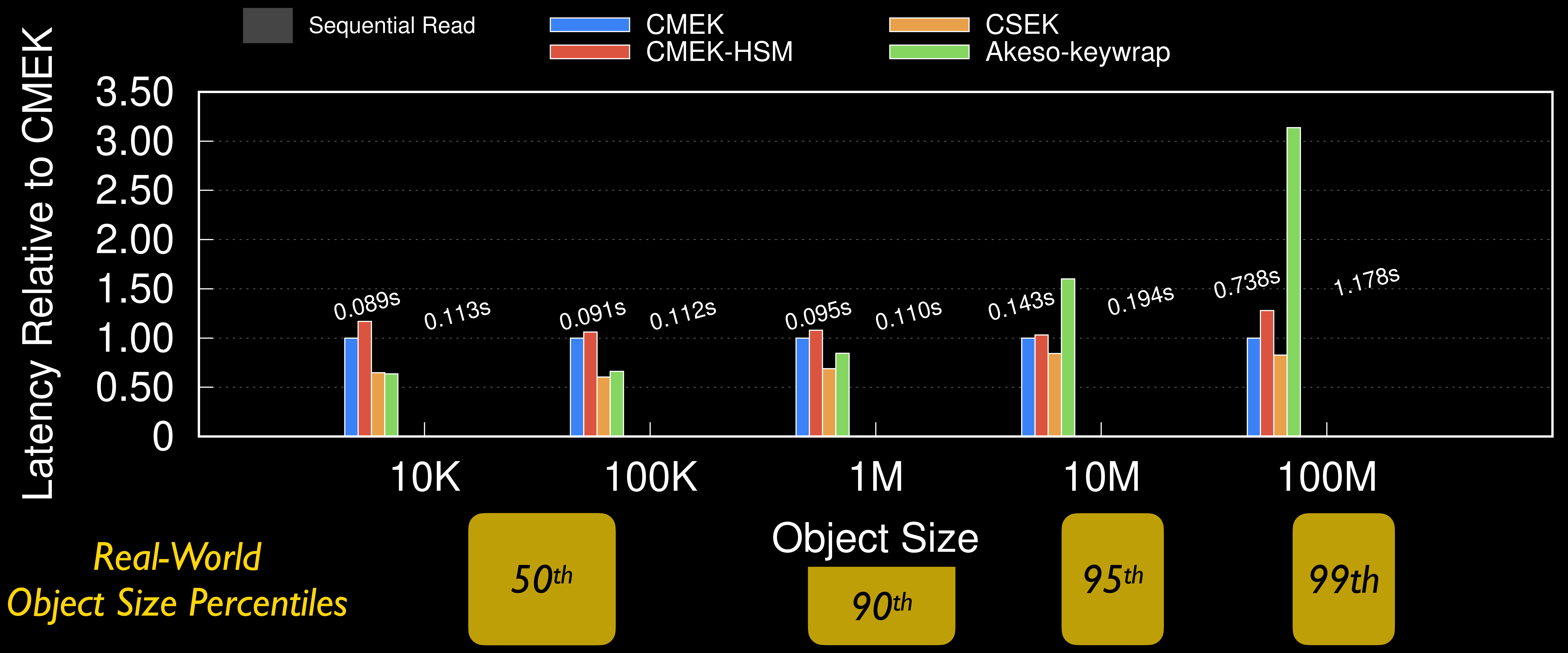
Real-World Object Size Percentiles

50th *90th* *95th* *99th*

Latency to Read/Write an Object

Integrated Akeso into GCSFuse

(Benchmark uses a single layer of encryption)



Real-World Object Size Percentiles

50th

90th

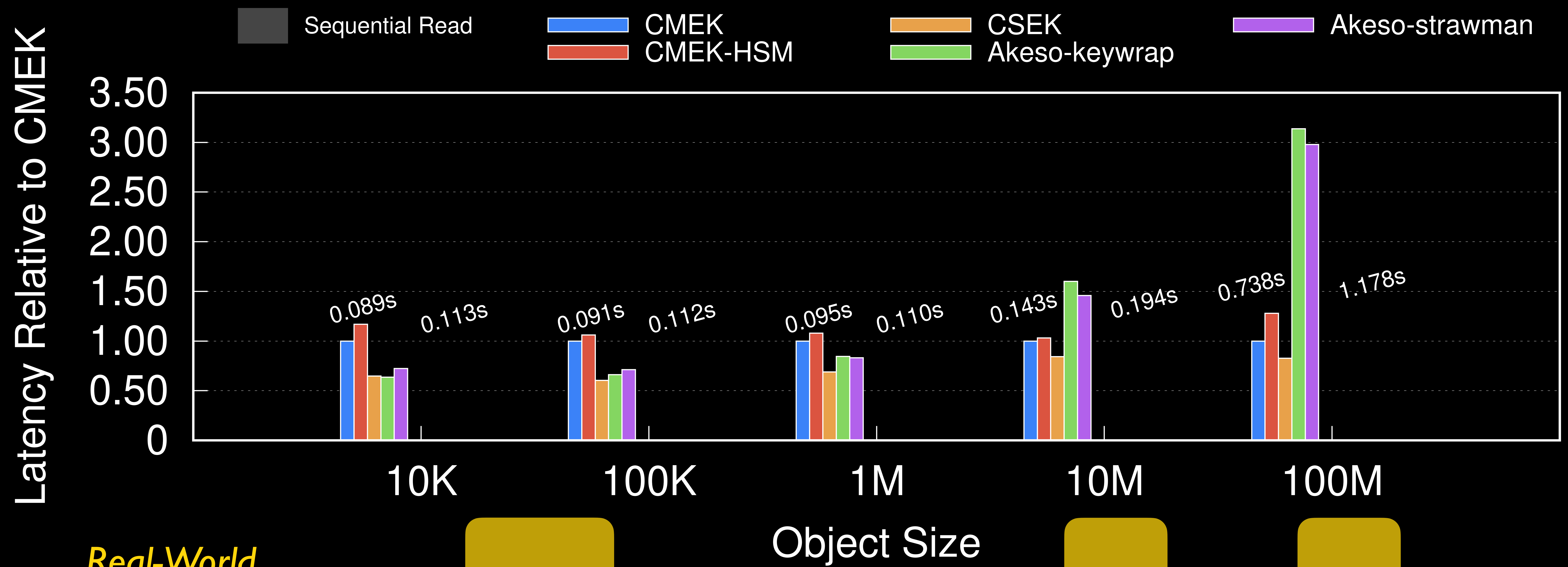
95th

99th

Latency to Read/Write an Object

Integrated Akeso into GCSFuse

(Benchmark uses a single layer of encryption)



Real-World Object Size Percentiles

50th

90th

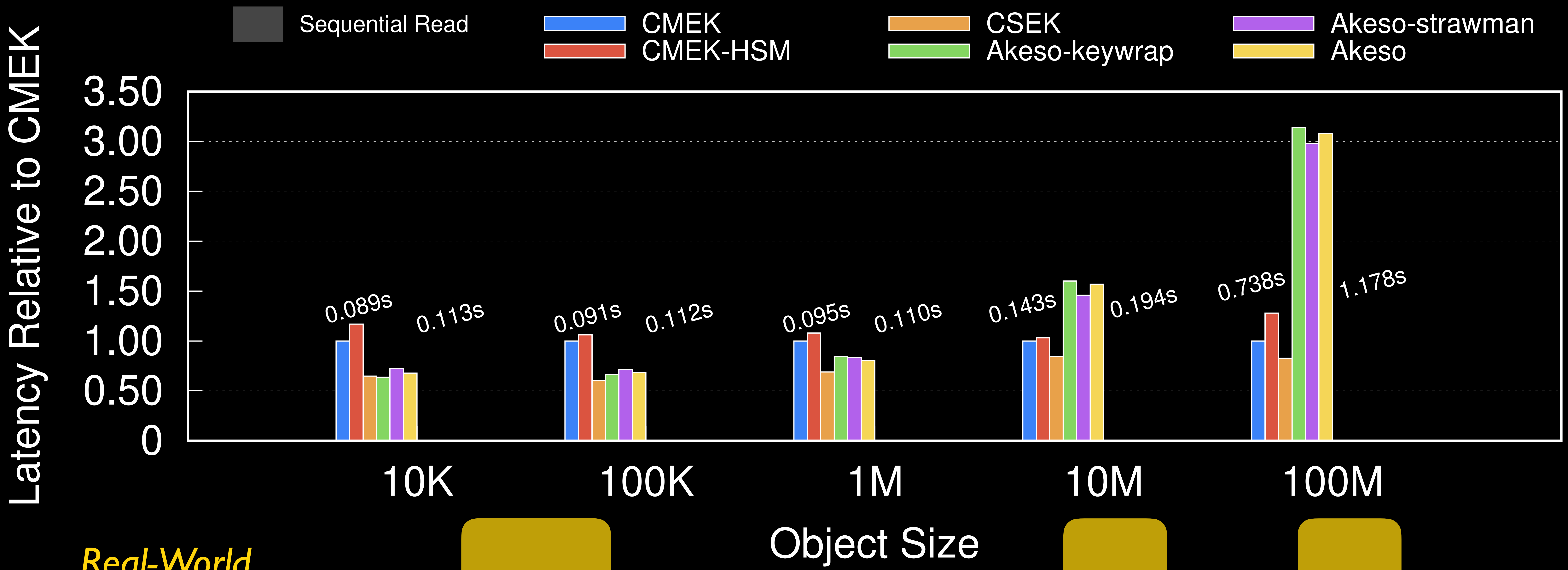
95th

99th

Latency to Read/Write an Object

Integrated Akeso into GCSFuse

(Benchmark uses a single layer of encryption)



Real-World Object Size Percentiles

50th

90th

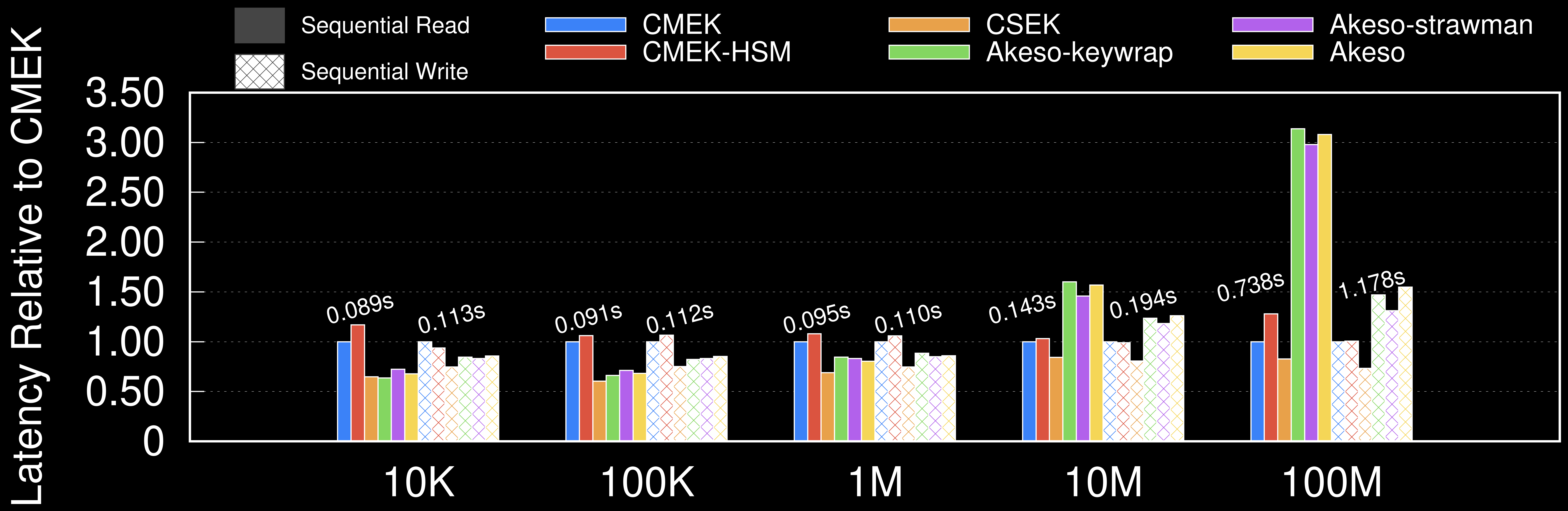
95th

99th

Latency to Read/Write an Object

Integrated Akeso into GCSFuse

(Benchmark uses a single layer of encryption)



Object Size

Real-World
Object Size Percentiles

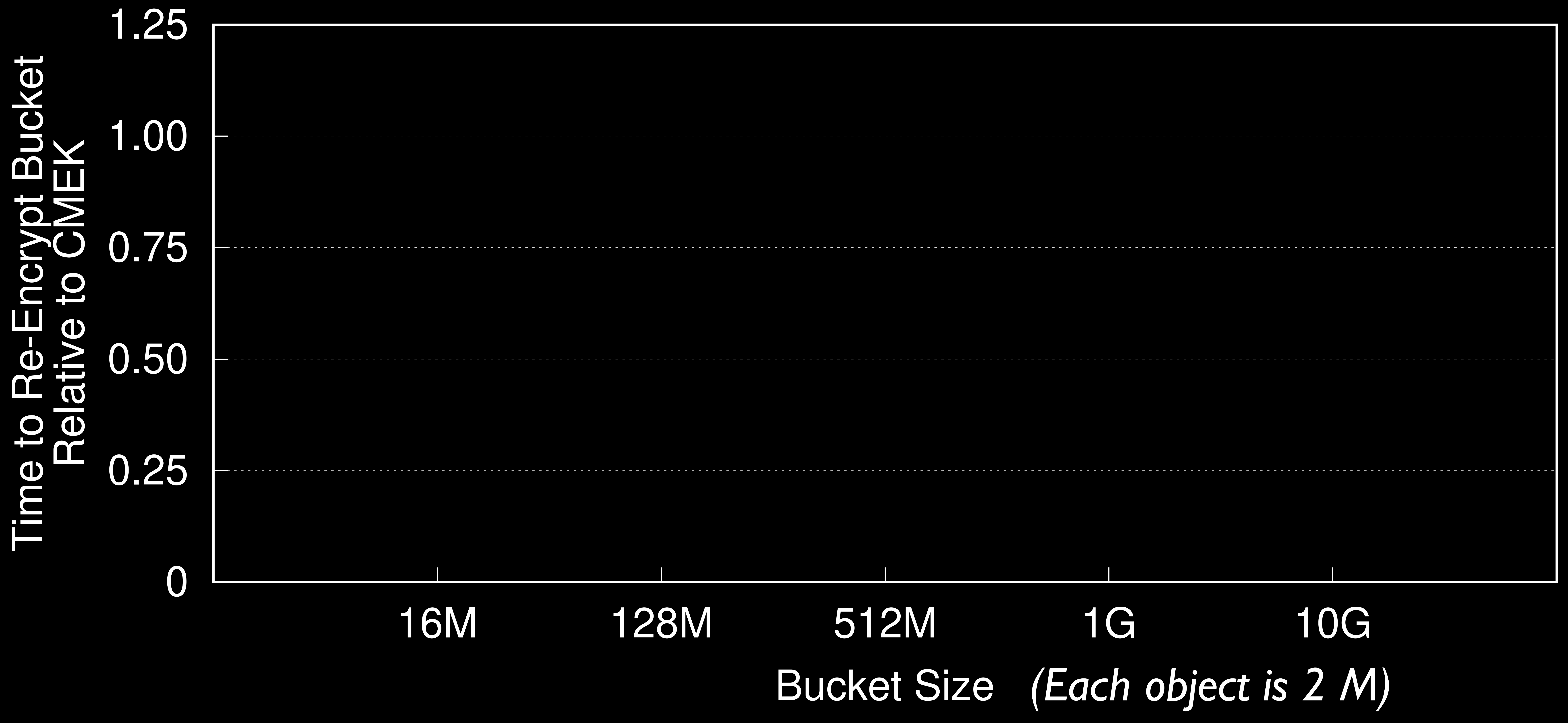
50th

90th

95th

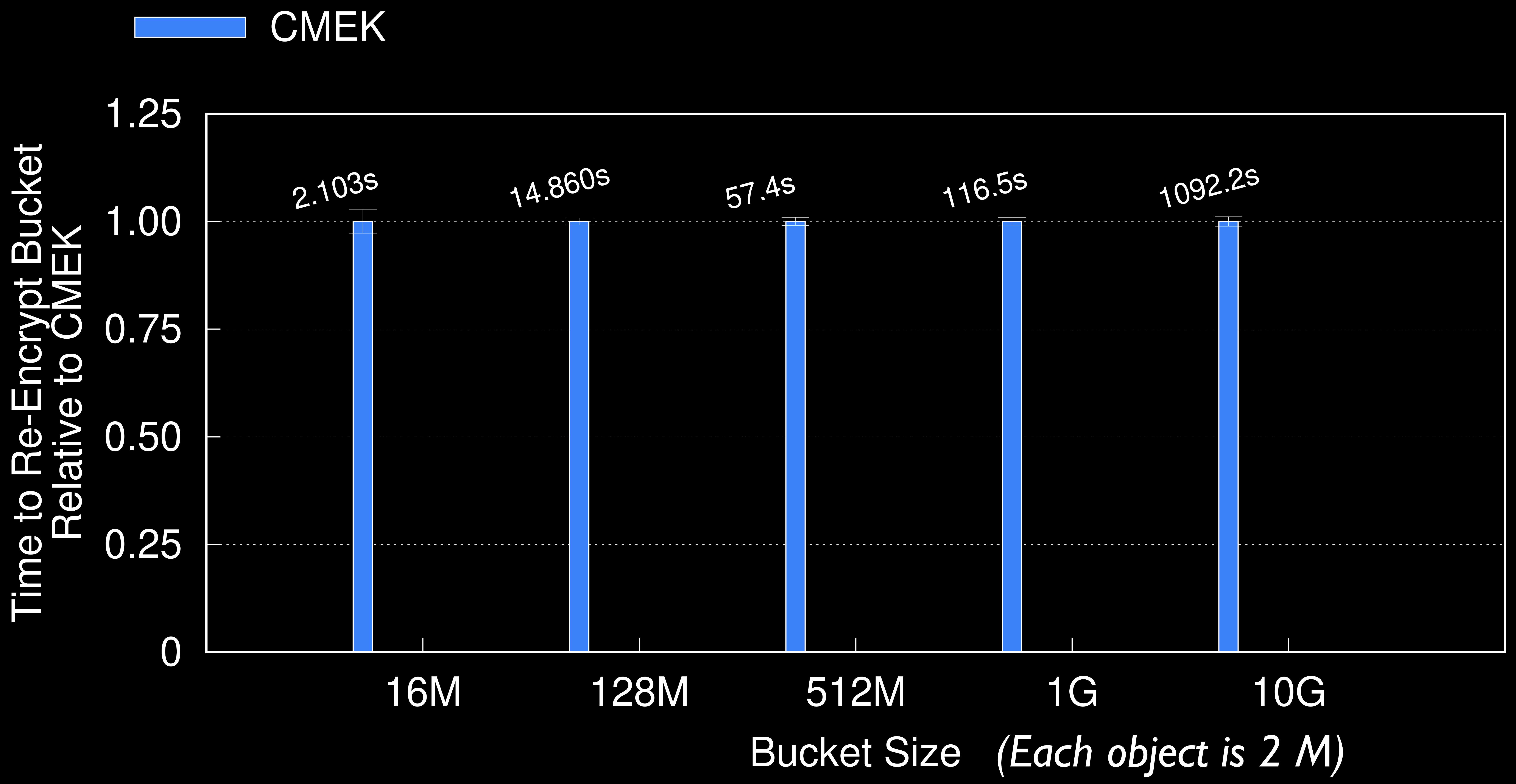
99th

Time to Re-Encrypt Bucket



(Each object is 2 M)

Time to Re-Encrypt Bucket

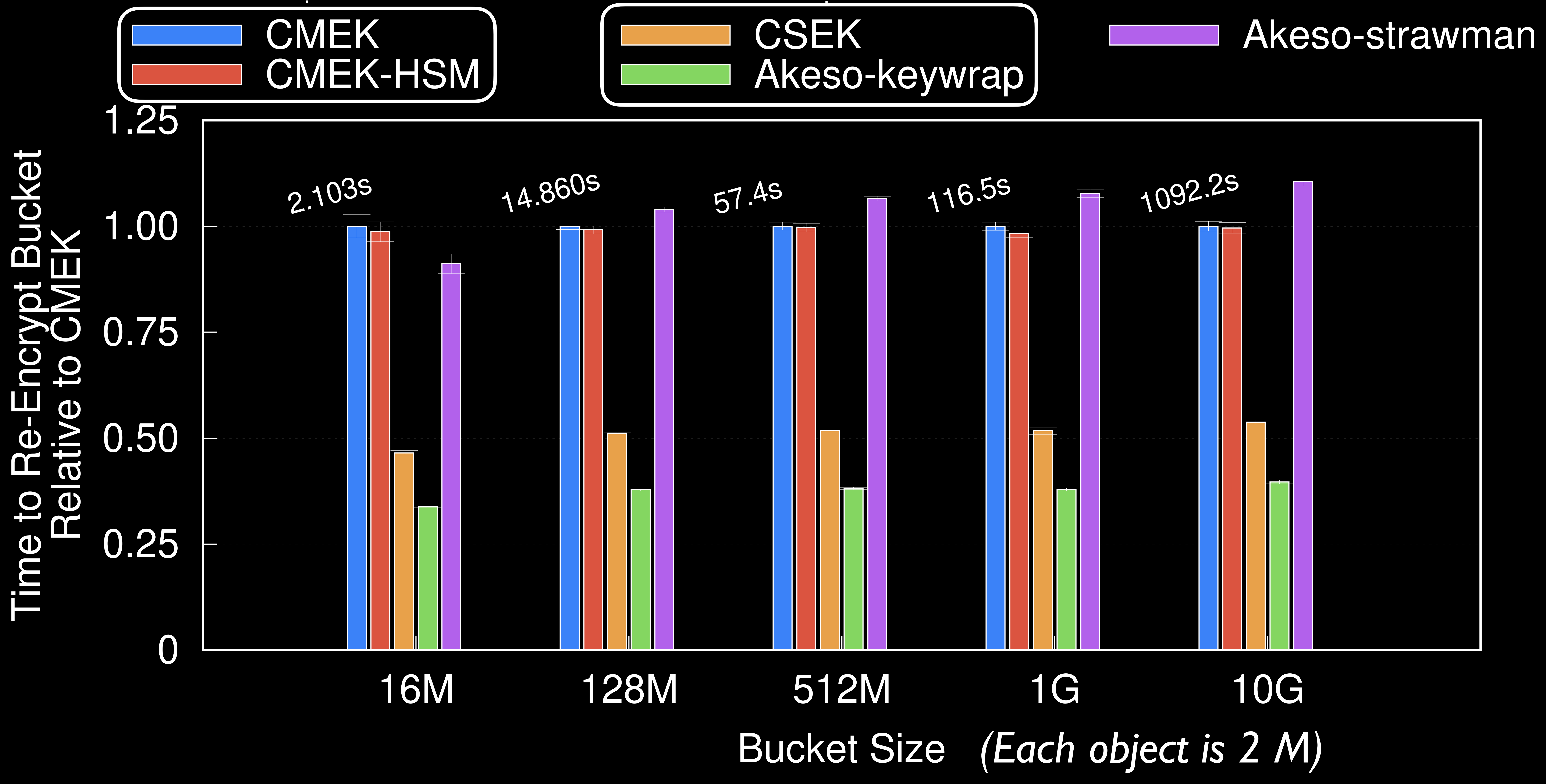


(Each object is 2 M)

Time to Re-Encrypt Bucket

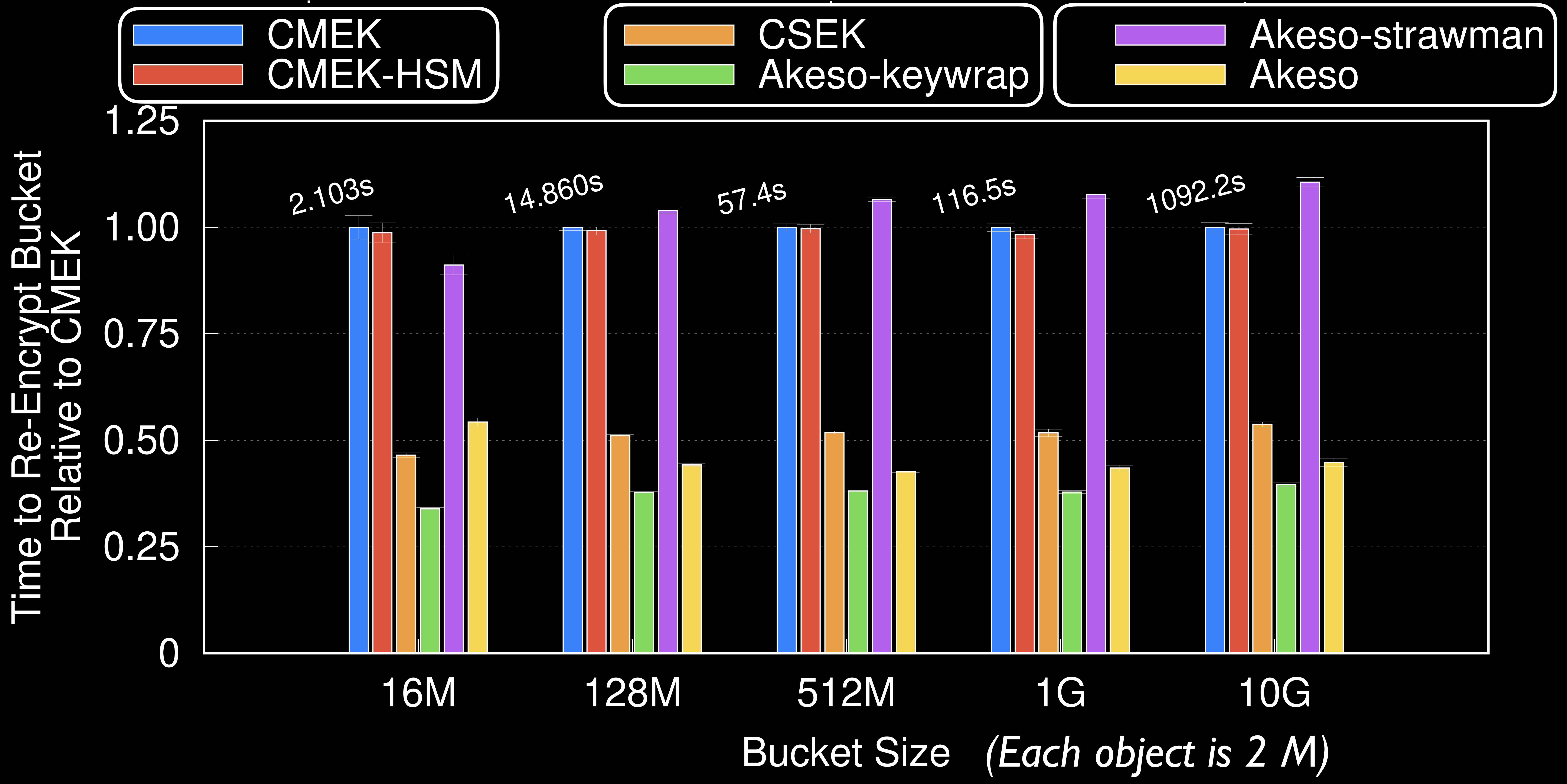
Changes DEK & KEK

Changes only KEK



(Each object is 2 M)

Time to Re-Encrypt Bucket



(Each object is 2 M)

Changes DEK & KEK

Changes only KEK

Akeso outperforms approaches that have the same or even weaker security

Monthly Costs (USD)

One key rotation per month

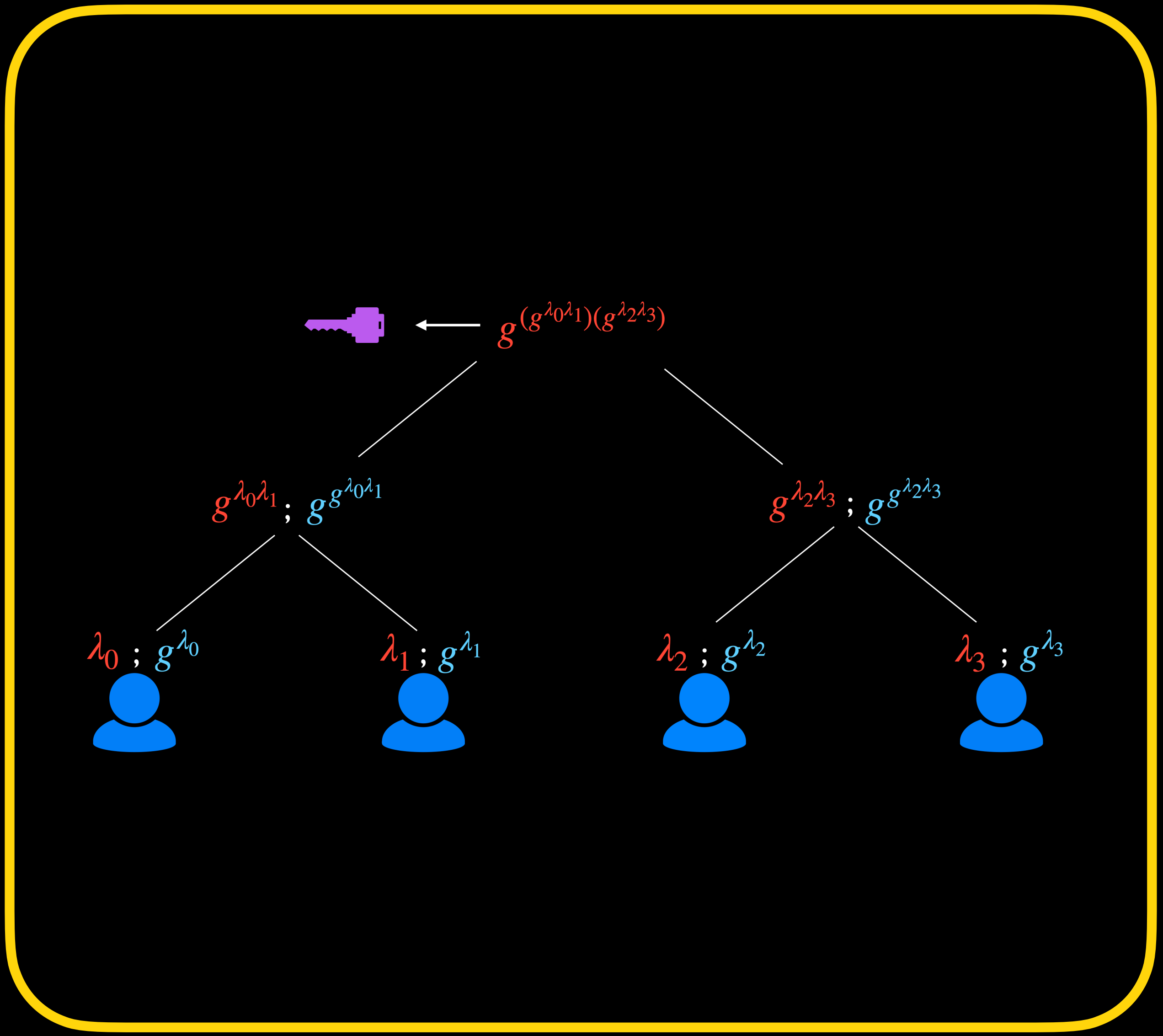
Bucket Size	CMEK	CMEK-HSM	CSEK	Akeso	Premium
10 GB	63.96	65.84	63.78	76.12	15.6% – 19.3%

Monthly Costs (USD)

One key rotation per month

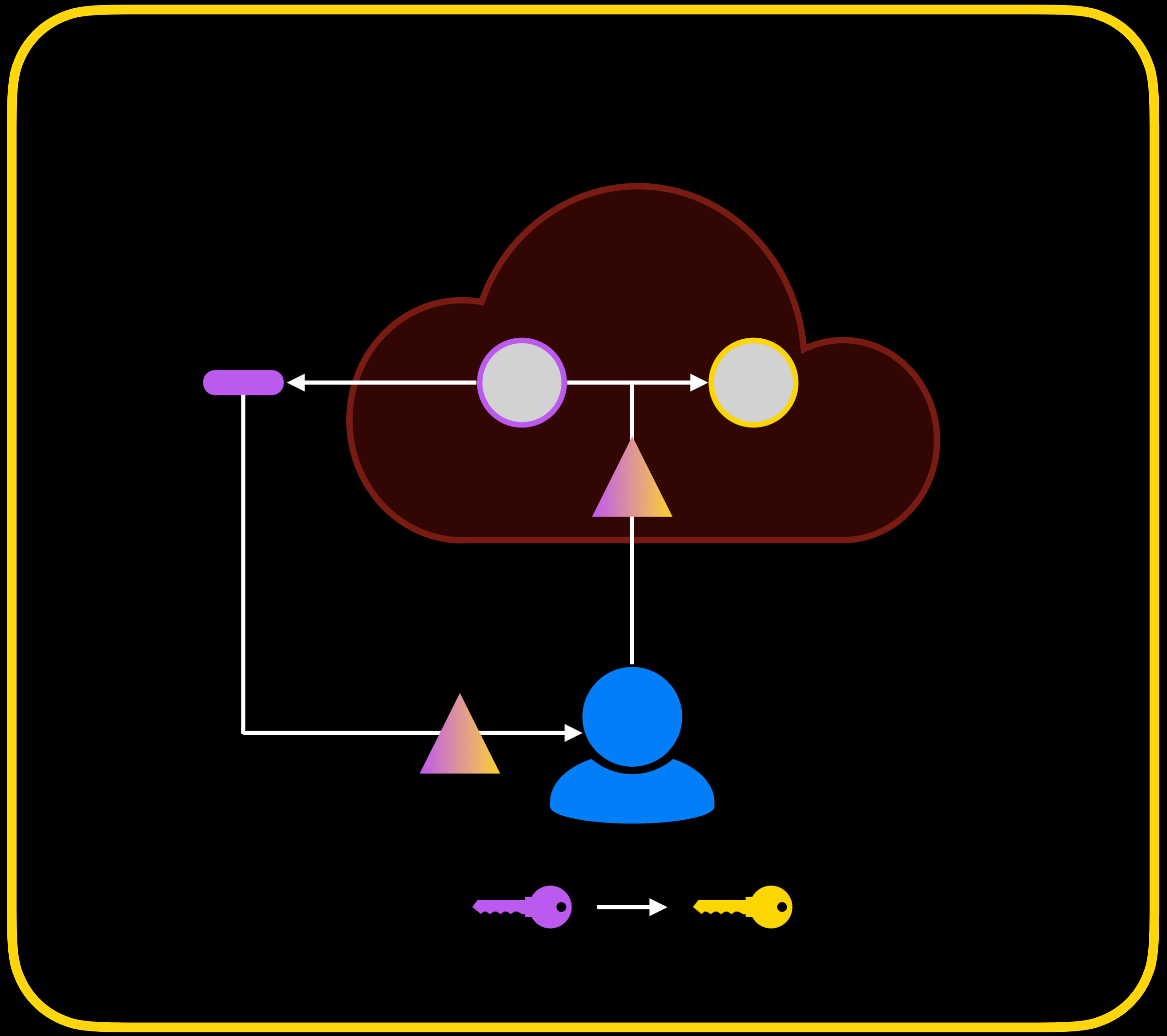
Bucket Size	CMEK	CMEK-HSM	CSEK	Akeso	Premium
10 GB	63.96	65.84	63.78	76.12	15.6% – 19.3%
100 GB	68.31	70.19	67.58	80.38	14.5% – 18.9%
1 TB	87.33	89.21	84.06	98.67	10.5% – 17.3%
10 TB	299.96	301.84	268.38	307.04	1.7% – 14.4%

https://github.com/etclab/akeso-artifact

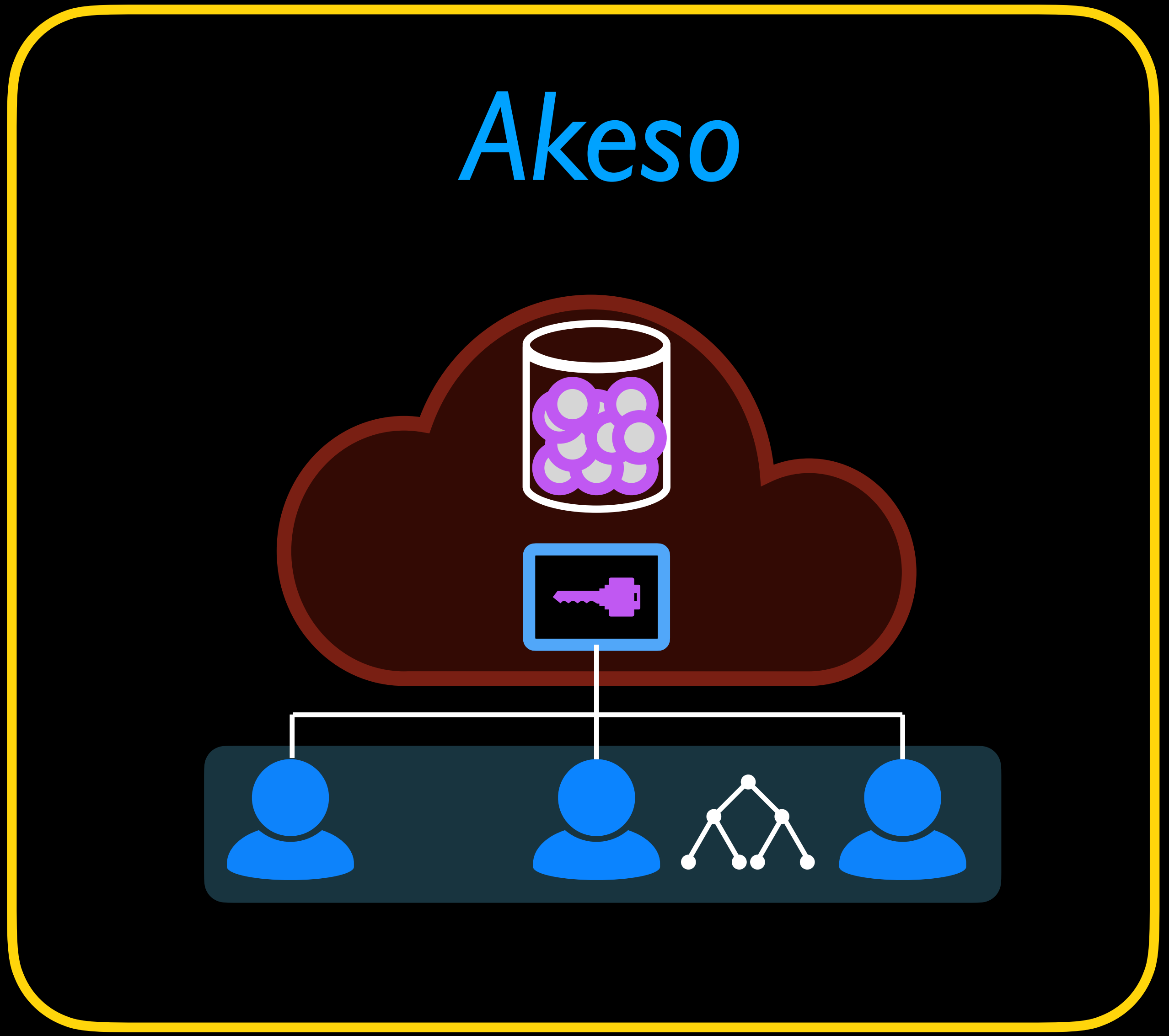


$g^{(g^{\lambda_0\lambda_1})(g^{\lambda_2\lambda_3})}$
$g^{\lambda_0\lambda_1}$; $g^{g^{\lambda_0\lambda_1}}$
$g^{\lambda_2\lambda_3}$; $g^{g^{\lambda_2\lambda_3}}$
λ_0 ; g^{λ_0}
λ_1 ; g^{λ_1}
λ_2 ; g^{λ_2}
λ_3 ; g^{λ_3}

Continuous Group Key Agreement



Updatable Encryption



PCS Cloud Storage

WILLIAM & MARY

CHARTERED 1693

/etc/lab

Extending Trust in Computing Lab