

# Cloud Security



CS 1660: Introduction to  
Computer Systems Security

# Outline

- Cloud model and threats
- Cloud storage integrity
- Provable data possession
- Information leakage from storage deduplication and synchronization
- Computing on encrypted data

# "The Cloud"

- Provides various features to outsource various components of applications
- Speeds up development, reduces management cost, synchronization across many devices ...
- In modern times, many different types of services, depending on what you want to outsource



# "Types" of services

- User-facing applications
  - Gmail, Dropbox, OneDrive, ...
- Developer APIs
  - Google Cloud Platform, Microsoft Azure, Amazon Web services, ...
  - Various types of services, depending on what developer needs
  - Eg. Block storage, databases, whole VMs, cloud functions, ...

# Featured products

## Google Cloud products

[Overview](#)[Featured products](#)[AI and Machine Learning](#)[API Management](#)[Compute](#)[Containers](#)[Data Analytics](#)[Databases](#)[Developer Tools](#)[Financial Services](#)[Healthcare and Life Sciences](#)[Hybrid and Multicloud](#)[Internet of Things \(IoT\)](#)[Management Tools](#)

### Compute Engine

Virtual machines running in Google's data center.

### Cloud Storage

Object storage that's secure, durable, and scalable.

### Cloud SDK

Command-line tools and libraries for Google Cloud.

### Cloud SQL

Relational database services for MySQL, PostgreSQL, and SQL Server.

### Google Kubernetes Engine

Managed environment for running containerized apps.

### BigQuery

Data warehouse for business agility and insights.

### Cloud CDN

Content delivery network for delivering web and video.

### Dataflow

Streaming analytics for stream and batch processing.

### Operations

Monitoring, logging, and application performance suite.

### Cloud Run

Fully managed environment for running containerized apps.

### Anthos

Platform for modernizing existing apps and building new

### Cloud Functions

Event-driven compute platform for cloud services and apps.

# Cloud Models

- Outsourced storage
  - User file storage: Dropbox, Google Drive, OneDrive, iCloud, ...
  - Developer APIs: Amazon S3, Google Cloud Storage, ...
- Outsourced machines (VMs)
  - Amazon EC2, Google Compute Engine, Microsoft Azure
- Outsourced services/databases
  - Firebase, MongoDB, Cloud functions, ...
- Outsourced applications
  - E.g., Gmail, Docs, Workday, ...

# Cloud Threats

- Attacker outside the provider's system
- Inept or malicious provider
  - Lost data
  - Corrupt data
  - Stolen data
- Malicious cloud customer ("tenant")
  - Information leaks across virtual machines
  - Information leaks from cloud storage
  - Information leaks from cloud applications



...we will have **no liability to you for any ... ALTERATION, OR THE DELETION, DESTRUCTION, DAMAGE, LOSS OR FAILURE...** of any of your content or other data...  
**Amazon Web Services customer agreement** <https://aws.amazon.com/agreement/>

# Threats from Other Cloud Tenants

- Other cloud customers ("tenants") may attempt...
  - Data theft
  - Data tampering
- Vulnerabilities in sharing of hardware, software, and network resources among clients
  - Eg. Memory-based channel attacks





# **How is data secured in the cloud?**

# Cloud security fundamentals

- Encryption-at-rest: data is encrypted when it is stored on disk
- Encryption-in-transit: data is encrypted when it is moving between two points
- Encryption-in-use: data is encrypted *while it's being processed*

Definitions by Google, but ideas are common to all providers:

<https://cloud.google.com/compute/confidential-vm/docs/about-cvm>

# Cloud security fundamentals

- Encryption-at-rest: data is encrypted when it is stored on disk  
=> Most cloud providers do this
- Encryption-in-transit: data is encrypted when it is moving between two points  
=> Most cloud providers do this
- Encryption-in-use: data is encrypted *while it's being processed*  
=> *Requires trusted execution environment*  
*(harder, specialized applications only)*

Definitions by Google, but ideas are common to all providers:

<https://cloud.google.com/compute/confidential-vm/docs/about-cvm>

# Example: cloud storage

```
from google.cloud import storage

def write_read(bucket_name, blob_name):
    """Write and read a blob from GCS using file-like IO"""

    storage_client = storage.Client()
    bucket = storage_client.bucket(bucket_name)
    blob = bucket.blob(blob_name)

    with blob.open("w") as f:
        f.write("Hello world")

    with blob.open("r") as f:
        print(f.read())
```

# Points of encryption

Encryption at rest: Uses some form of file and/or encryption (whole disk, database object, both, ...)

Who holds the keys?

# Encryption at Rest: Who holds the keys?

Can be configurable by customer:

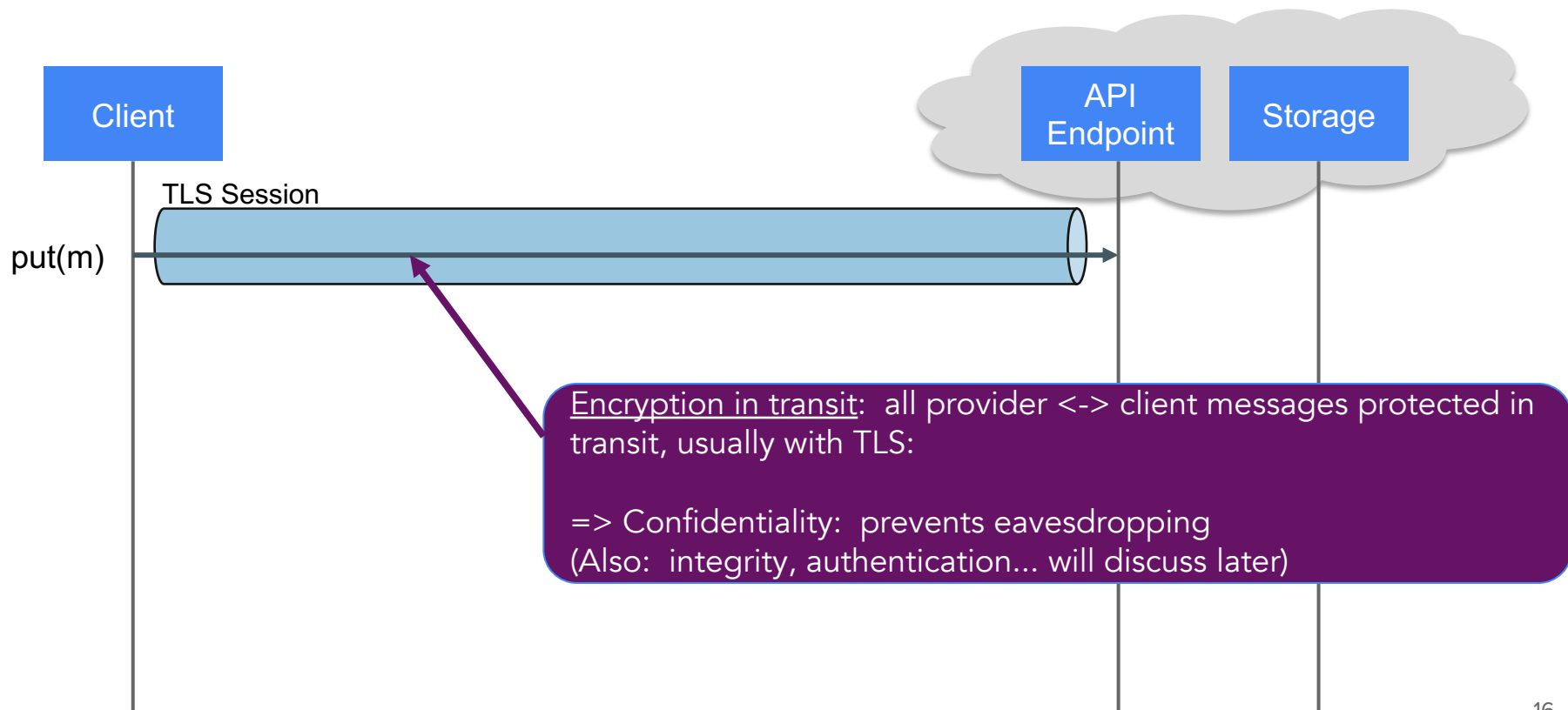
- Default method: key controlled by provider, encryption is transparent to user
- Customer-managed keys: provider has key generation service, customer decides which objects are encrypted with which keys
- Client-side keys: client application generates the keys, encrypts data before sending to provider  
=> End to end encryption

# Encryption in transit

Data encrypted when moving between points

- Client->Provider: TLS
- Between provider services, datacenters

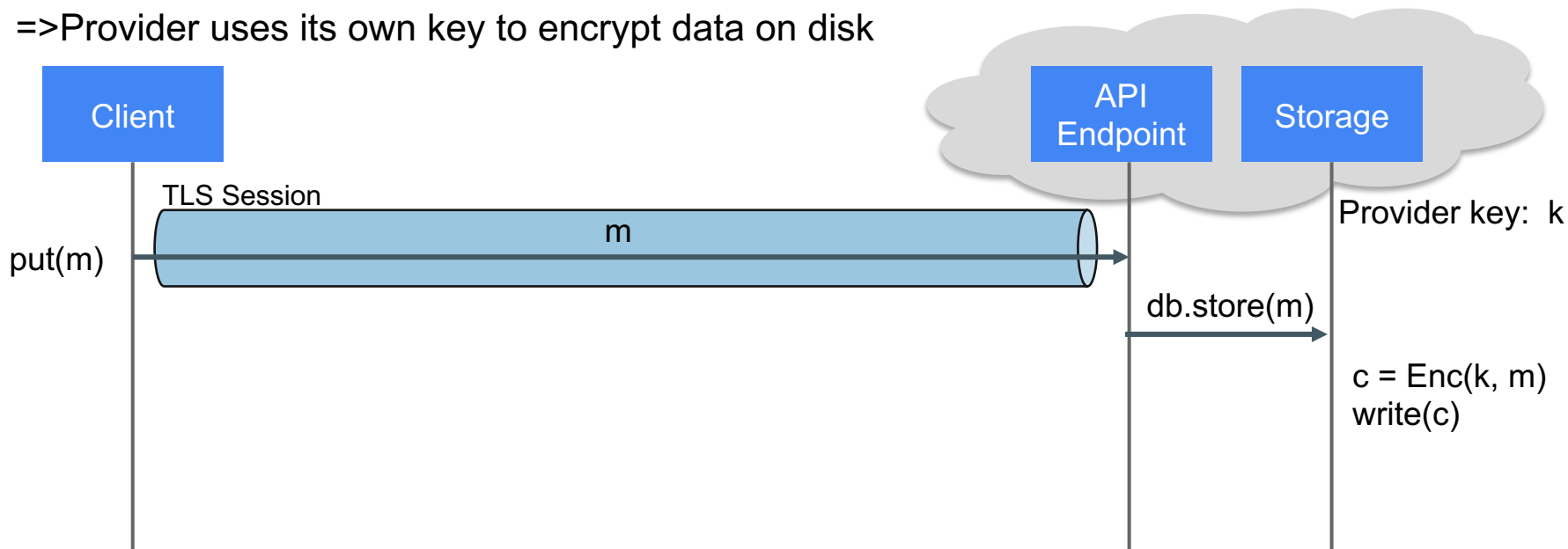
# Example: Default method





# Example: Default method

=> Provider uses its own key to encrypt data on disk



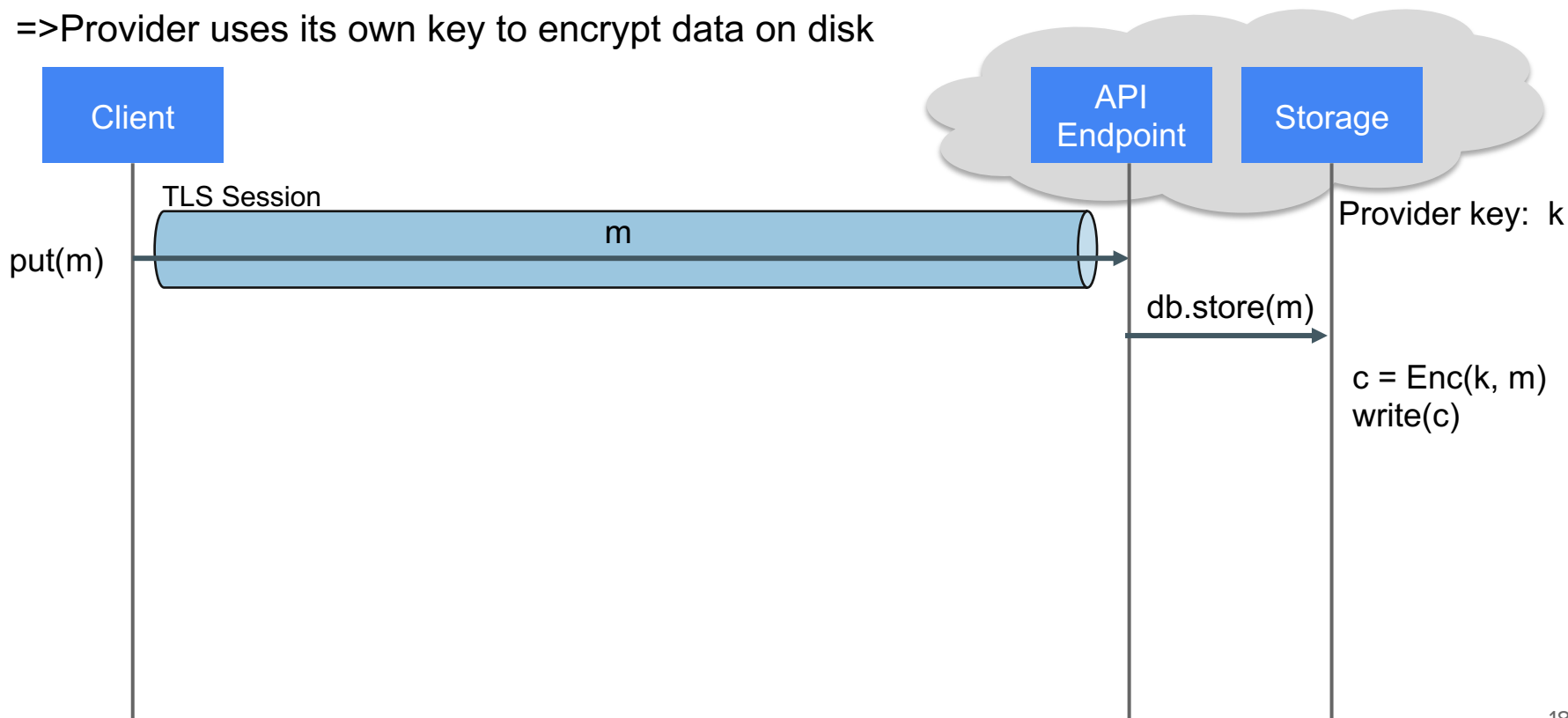
Problems?

=> Message is decrypted at some point while provider is storing it  
(might be very small, but nonzero)

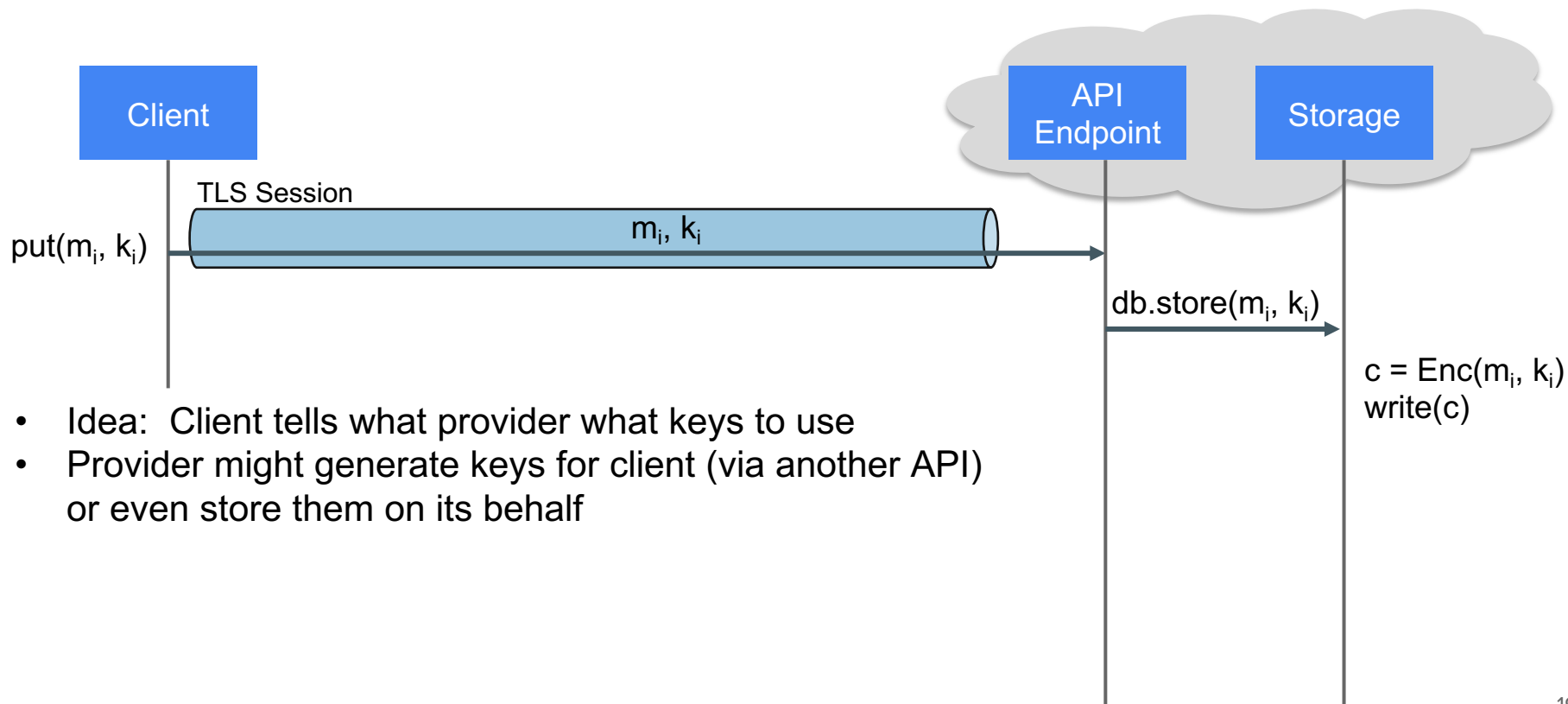
=> Provider controls where key is used, how many systems/customers it's used on...

# Example: Default method

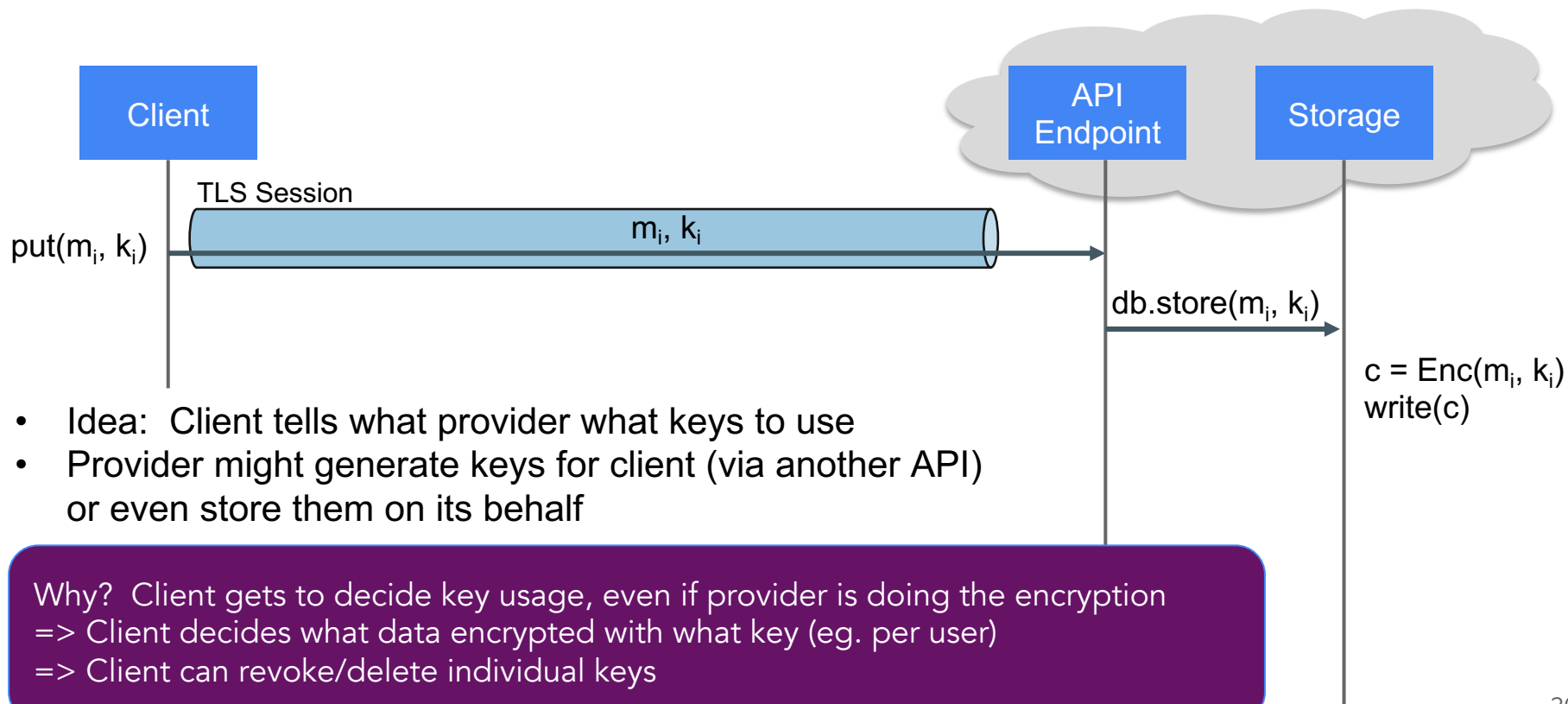
=> Provider uses its own key to encrypt data on disk



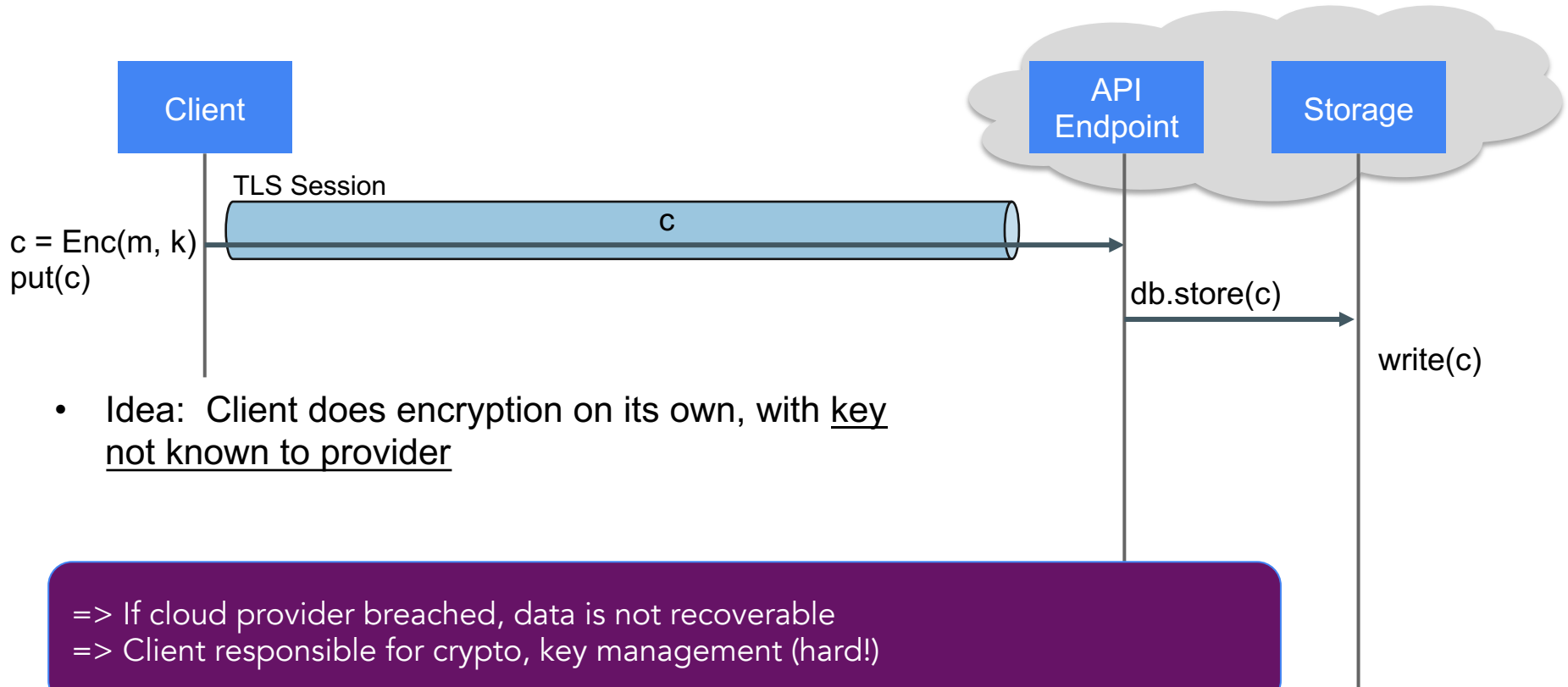
## Example: Customer-managed keys (one way)



## Example: Customer-managed keys (one way)



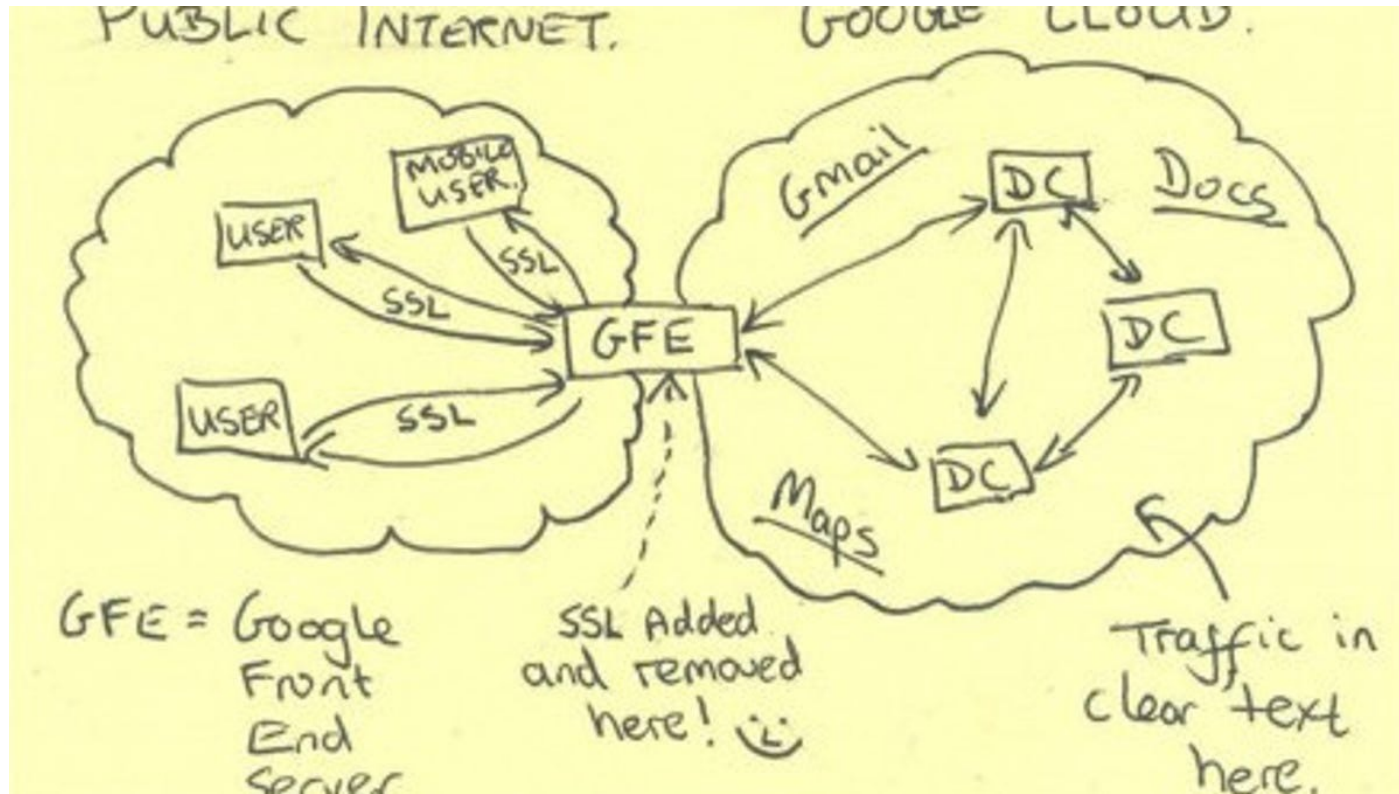
# Example: End to end encryption (client-side keys)



# End-to-end encryption

- Even if cloud provider is breached, data is not recoverable
- Client is more complex
  - Needs to manage keys
  - Cryptographic operations must happen client side
  - => You'll do this in Dropbox! (we give you the crypto library, though!)

End-to-end encryption not as common in cloud systems, but getting there  
=> In general, cloud providers are so critical there's incentive to provide (some) security features for customers



when using Cloud Storage. Below is a summary of the encryption options available to you:

- *Server-side encryption*: encryption that occurs after Cloud Storage receives your data, but before the data is written to disk and stored.
  - *Customer-managed encryption keys*: You can create and manage your encryption keys through [Cloud Key Management Service](#). Customer-managed encryption keys can be stored as software keys, in an [HSM cluster](#), or [externally](#).
  - *Customer-supplied encryption keys*: You can create and manage your own encryption keys. These keys act as an additional encryption layer on top of the standard Cloud Storage encryption.
- *Client-side encryption*: encryption that occurs before data is sent to Cloud Storage. Such data arrives at Cloud Storage already encrypted but also undergoes server-side encryption.



**Warning:** If you use customer-supplied encryption keys or client-side encryption, you must securely manage your keys and ensure that they are not lost. If you lose your keys, you are no longer able to read your data, and you continue to be charged for storage of your objects until you delete them.



# Advanced Data Protection for iCloud

Starting with iOS 16.2, iPadOS 16.2 and macOS 13.1, you can choose to enable Advanced Data Protection to protect the vast majority of your iCloud data, even in the case of a data breach in the cloud.

With Advanced Data Protection, the number of data categories that use end-to-end encryption rises to 23 and includes your iCloud Backup, Photos, Notes, and more. The table below lists the additional data categories that are protected by end-to-end encryption when you enable Advanced Data Protection.

## Data categories and encryption

The table below provides more detail on how iCloud protects your data when using standard data protection or Advanced Data Protection.

Data category	Standard data protection		Advanced Data Protection	
	Encryption	Key storage	Encryption	Key storage
iCloud Mail (1)	In transit & on server	Apple	In transit & on server	Apple
Contacts (2)	In transit & on server	Apple	In transit & on server	Apple



work in progress

[News](#)

[Work Culture](#)

[Made in Dropbox](#) ▾

[Collections](#) ▾

[Tech blog](#) ↗



**Dropbox**

+



**boxcryptor**

# Boxcryptor assets, bring end-to-end encryption to business users

By Dropbox Team

Published on November 29, 2022

# Cloud Storage Integrity

# Cloud Storage Integrity

- Alice outsources her files to Bob (cloud storage provider)
- How can Alice check whether a file downloaded from Bob has not been corrupted?

# Did the Cloud Corrupt my Files?

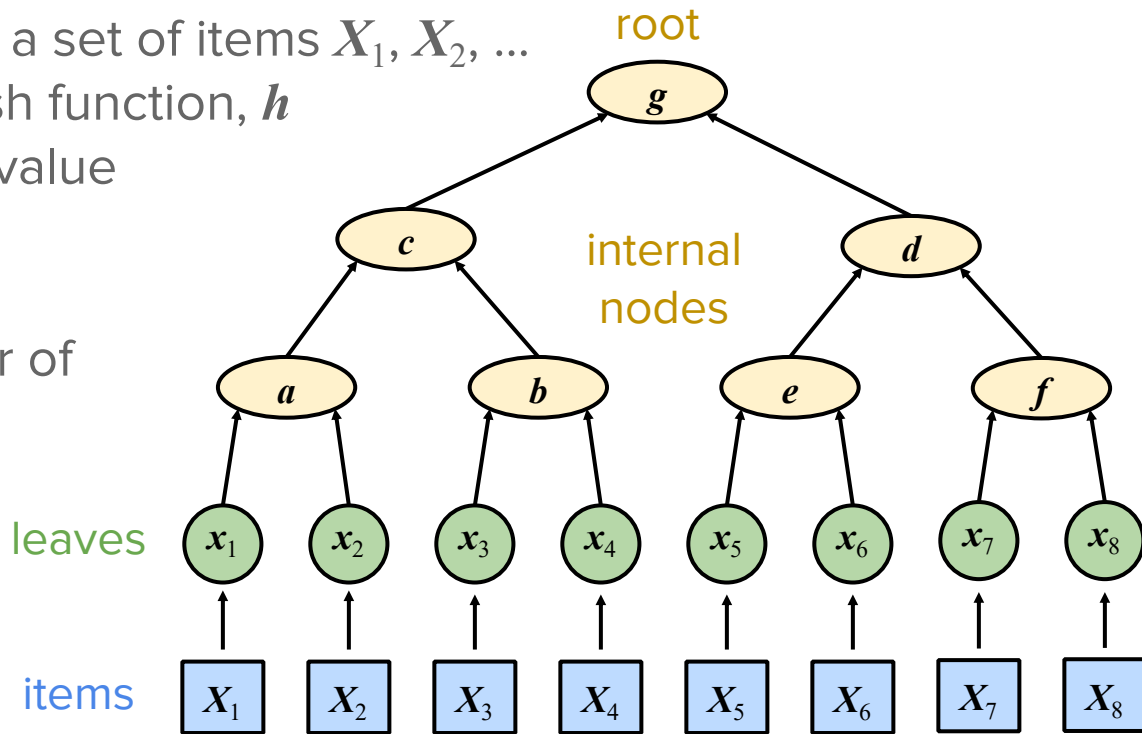
- Alice outsources her files to Bob (cloud storage provider)
- How can Alice check whether a file subsequently downloaded from Bob has not been corrupted?
- Basic solution
  - Alice computes and keeps cryptographic hashes of her files
  - Upon download of a file from Bob, Alice checks the hash of the downloaded file against the stored hash
- Alice detects any change in the file with overwhelming probability

# More Efficient Integrity Verification




- Storing  $n$  hashes is more efficient than storing  $n$  files for Alice
- However, the asymptotic space requirement for Alice is still  $O(n)$
- Improved solution
  - Using a cryptographic hash function, Alice builds a **Merkle tree** over her files, where leaves store hashes of files and internal nodes store hierarchically computed hash values
  - Alice keeps the **root hash** of the Merkle tree (and discards the rest of the tree)
  - The asymptotic space requirement for Alice is now only  $O(1)$

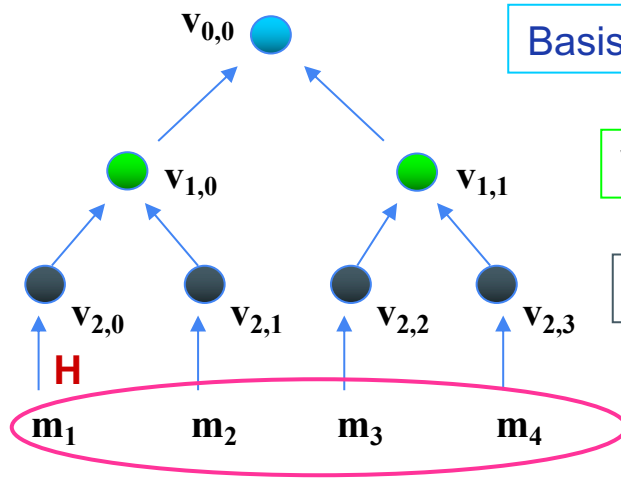
# What is a Merkle Tree

- Binary tree built on top of a set of items  $X_1, X_2, \dots$  using a cryptographic hash function,  $h$
- Each node stores a hash value
- Leaf: hash of item
  - $x_i = h(X_i)$
- Internal node: hash of pair of values at children
  - $a = h(x_1 x_2)$
  - $b = h(x_3 x_4)$
  - $c = h(a b)$
  - ...



# Hash Tree (Merkle): building

-  Basis: authenticated tree root
-  Authentication structure
-  Data hash



$$\text{Basis} = V_{0,0} = H [ (V_{1,0}) \parallel (V_{1,1}) ]$$

$$V_{1,1} = H [ (V_{2,2}) \parallel (V_{2,3}) ]$$

$$V_{2,2} = H(m_3)$$

$$V_{2,3} = H(m_4)$$

data must be ordered

H is a Hash function

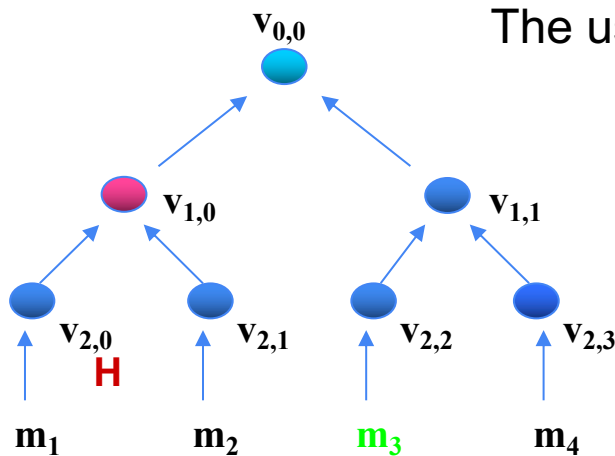


# Hash Tree (Merkle): test

A user would verify data authenticity of  $m_3$

Authenticated answer is made by:  $m_3$ ,  $V_{2,3}$ ,  $V_{1,0}$

And from the Basis signed by a CA.



The user can verify if  $m_3$  is authentic:

$$V_{2,2} = H(m_3)$$

$$V_{1,1} = H(V_{2,2} || V_{2,3})$$

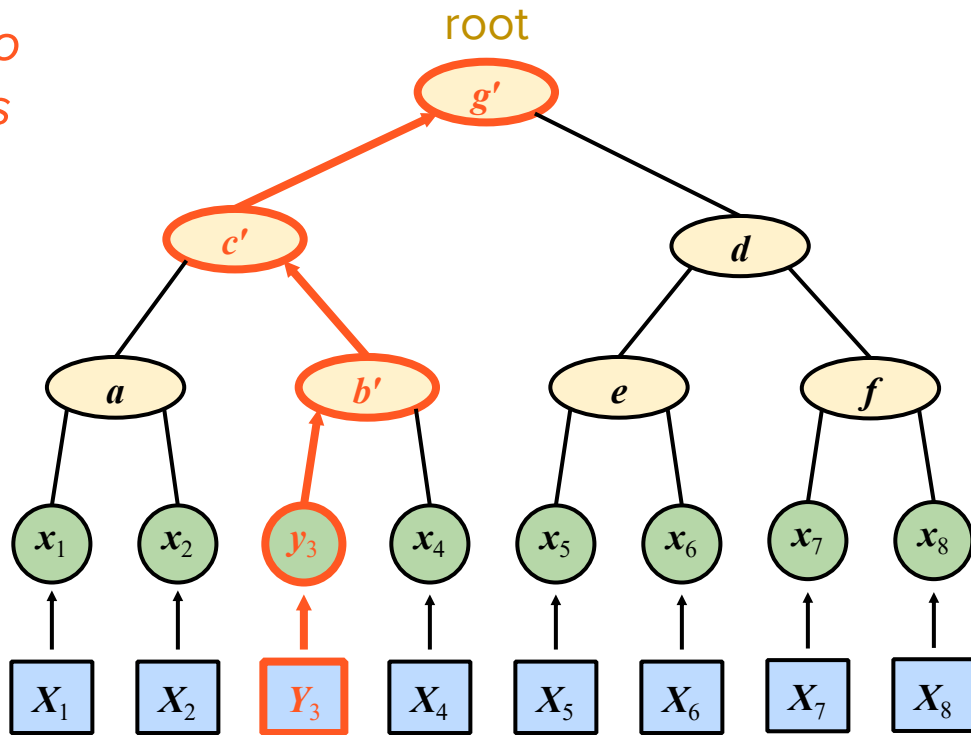
$$V_{0,0} = H(V_{1,0} || V_{1,1})$$

If Basis ==  $V_{0,0}$  then  $m_3$  is authentic

# Integrity Property of a Merkle Tree

*Given a Merkle tree, it is unfeasible to modify any nonempty subset of items without modifying also the root hash*

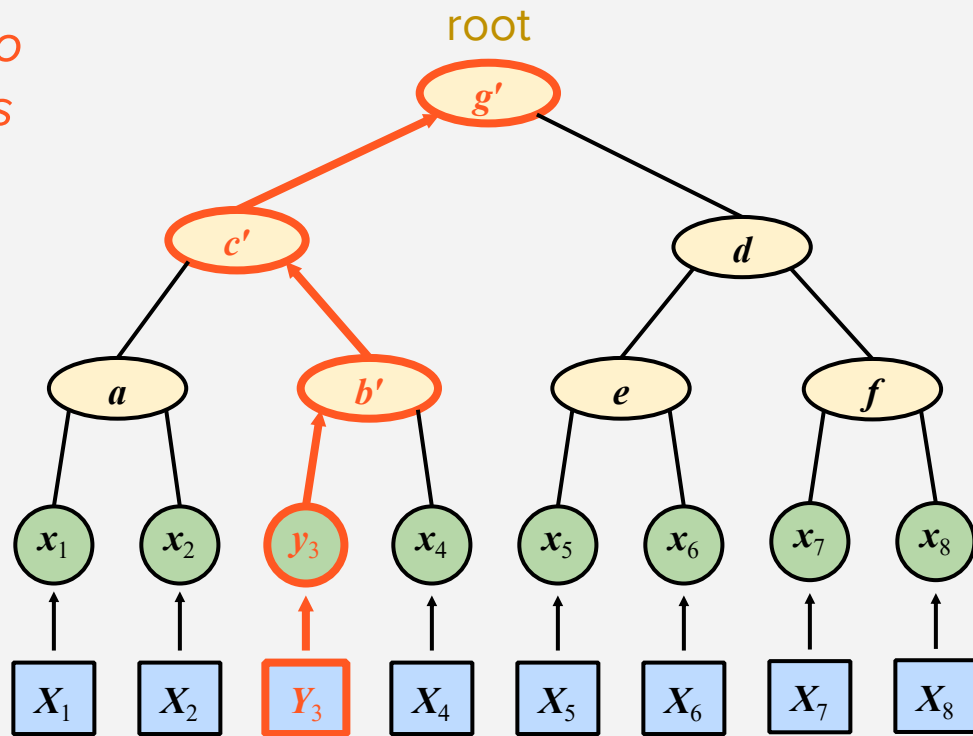
- Why?



# Integrity Property of a Merkle Tree

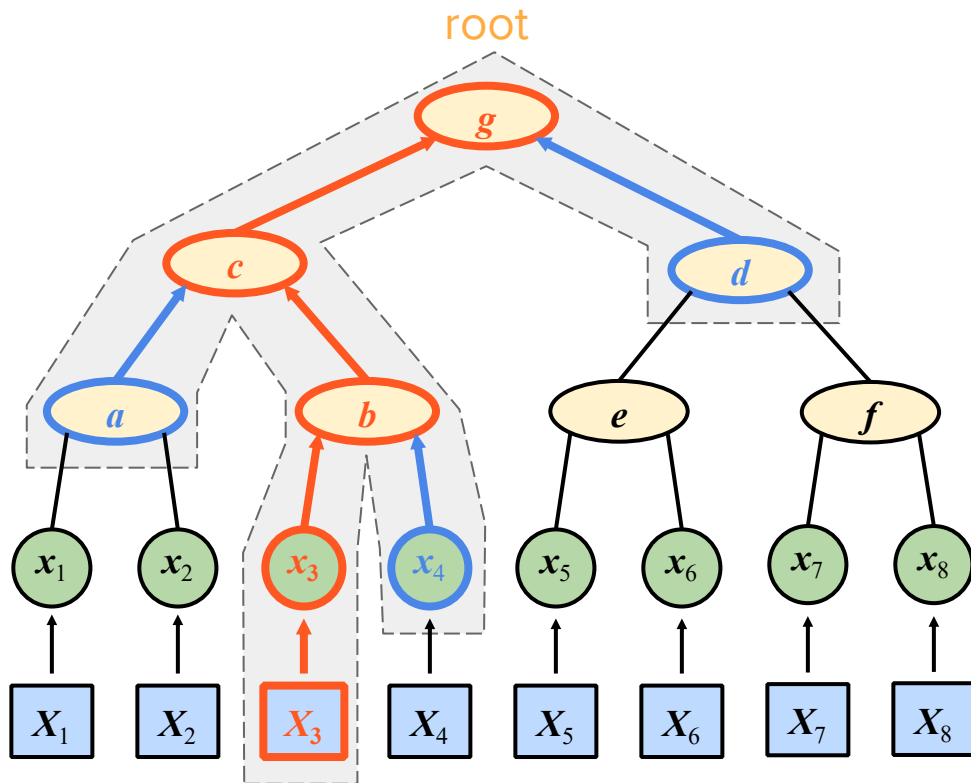
*Given a Merkle tree, it is unfeasible to modify any nonempty subset of items without modifying also the root hash*

- Follows from collision resistance of the hash function
- Suppose we modify an item, say  $X_3$ , into  $Y_3$
- The nodes from the leaf to the root change value, else we found a collision of the hash function somewhere along this path
- The argument generalizes to modifying multiple items



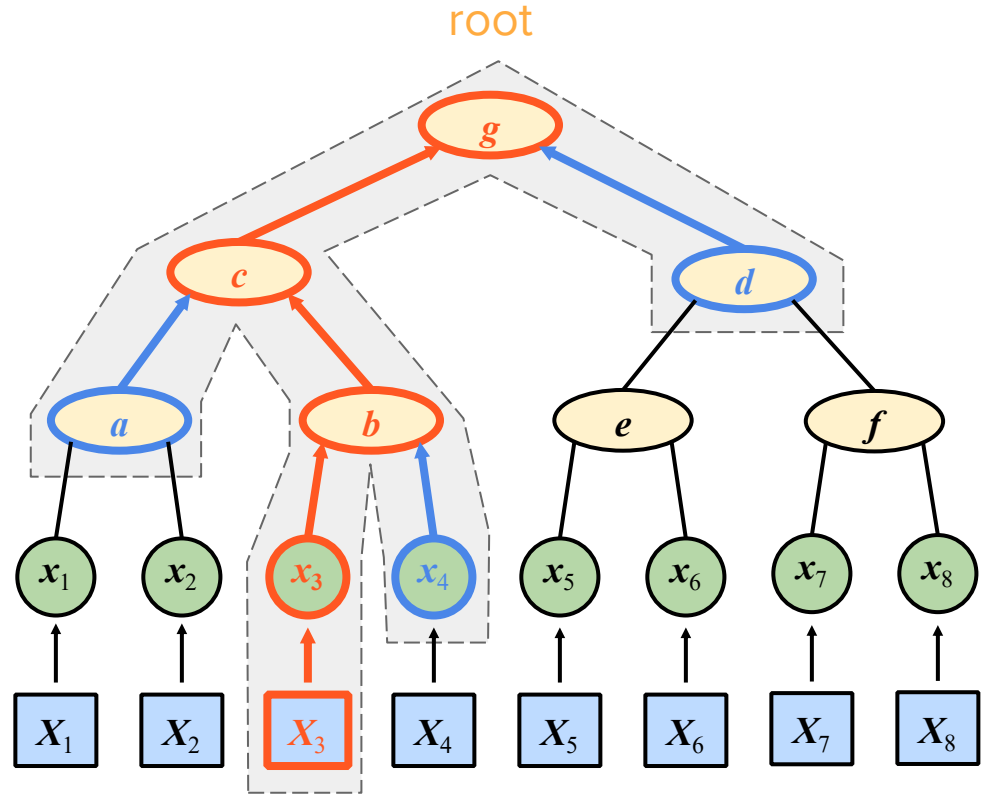
# Proof of an Item in a Merkle Tree

- A Merkle tree provides a proof that an item is in the set:
  - sequence of hash values and L/R (left/right) indicators
- To build the proof for an item:
  - Start at the leaf and go up to the root
  - At each node, pick hash value and side of sibling node
- Example: proof for  $X_3$ 
  - $(x_4, R), (a, L), (d, R)$



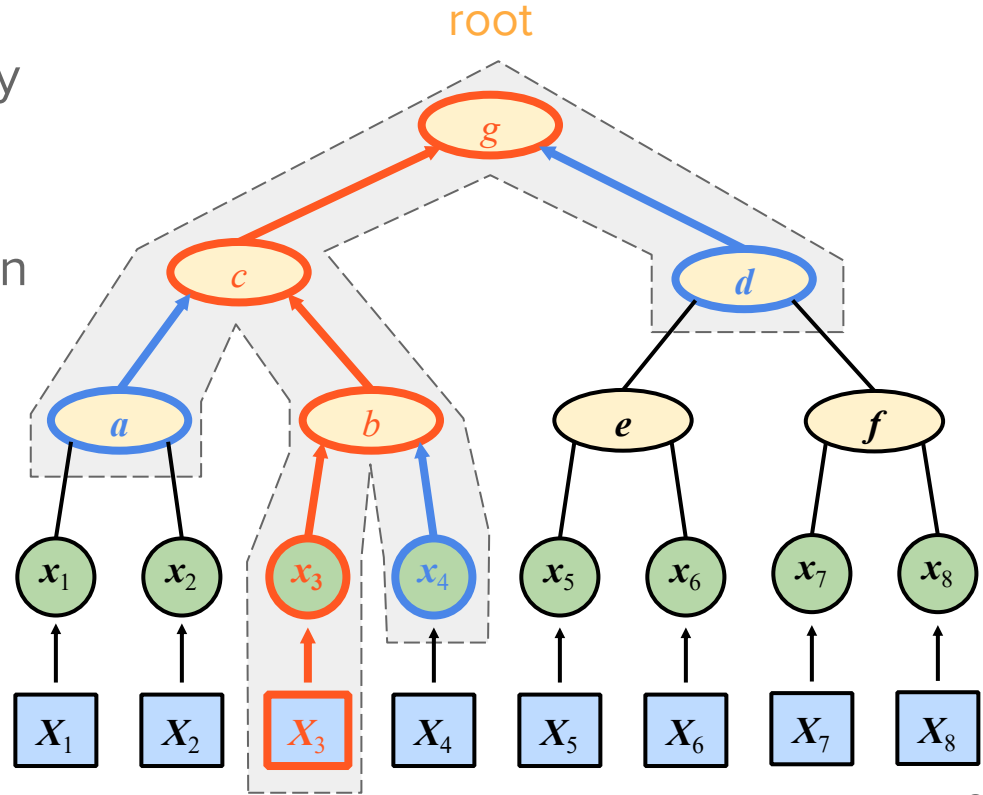
# Proof of an Item in a Merkle Tree (cont.)

- Proof verification:
  - Compare root hash with hash derived from item and proof
- Proof for  $X_3$ :
  - $(x_4, R), (a, L), (d, R)$
- Verification:
  - $g = h(h(a, h(x_3, x_4)), d)$
- The integrity property ensures one cannot forge proofs
- Proofs have size proportional to the logarithm of the number of items



# Proof of an Item in a Merkle Tree (cont.)

- The proof of an item is essentially a chain of hashes
- L/R indicators denote order of hashing at each node of the chain
- Size of proof (number of values) is height of tree, i.e., logarithm in base two of number of items
- Examples:
  - 8 items, proof size 3
  - 1,024 items, proof size 10
  - 1 M items, proof size 20
  - 1 B items, proof size 30



# Who is Merkle?

**Ralph C. Merkle**

A pioneer of modern cryptography

<http://www.merkle.com/>



Source: <http://www.merkle.com/>

# Selected Related Work

- Authenticated Data Structures

- Two party ADS model where the client maintains a proof of validity for the data [Goodrich Tamassia 03]
- Authenticated skip list embedded in a relational table [Di Battista Palazzi 07]

- Storage check

- Efficient integrity checking of untrusted network storage [Heitzmann, Palazzi, Papamanthou, Tamassia 08]

- Set operations

- Query Racing: Fast Completeness Certification of Query Results [Palazzi, Pizzonia, Pucacco 10]



A real Hack: Solarwinds

**How to manage an attack?**

# Today's discussion

1. What is the Solarwinds hack?
2. Why did it happen?
3. What should we do?

# SolarWinds Hack

- **What happened?**
  - Texas-based IT management company
  - Supply chain attack
    - malware inserted into update of Orion system (network monitoring software)
  - 100+ companies impacted (Microsoft included) + US government agencies
- **Response**
  - FireEye, a cybersecurity company impacted, discovered the hack
  - SolarWinds issued a security advisory
  - FBI Investigation to find the actors – Russian hacker group?
- **Direct cause**
  - Bad security practices
    - “solarwinds123” used as a password for secure server (security researcher already warned SolarWinds of this!)

# Solarwinds blamed intern for weak password — experts have doubts

Jeff Elder

Feb 26, 2021 | 9:45 PM ET



**SolarWinds CEO Sudhakar Ramakrishna testifies Tuesday on Capitol Hill.** Photo by Demetrius

Freeman-Pool/Getty Images

- **SolarWinds told Congress that using the password 'solarwinds123' was an intern's mistake.**

# Supply chain issues

- Centralization of software has created points of vulnerability that dramatically reduce hacker effort
  - SolarWinds allowed hackers to access 18,000 systems
- Vulnerabilities in one company's product have cascading effects beyond their immediate customers
  - CISA: 30% of SolarWinds victims did not use SolarWinds
- Example: 2017 NotPetya attack
  - Malware deployed by a malicious automatic update in MeDoc, Ukrainian tax preparation software
  - Caused \$10 billion damage
  - Damaged pharmaceutical production, global shipping, hospital systems

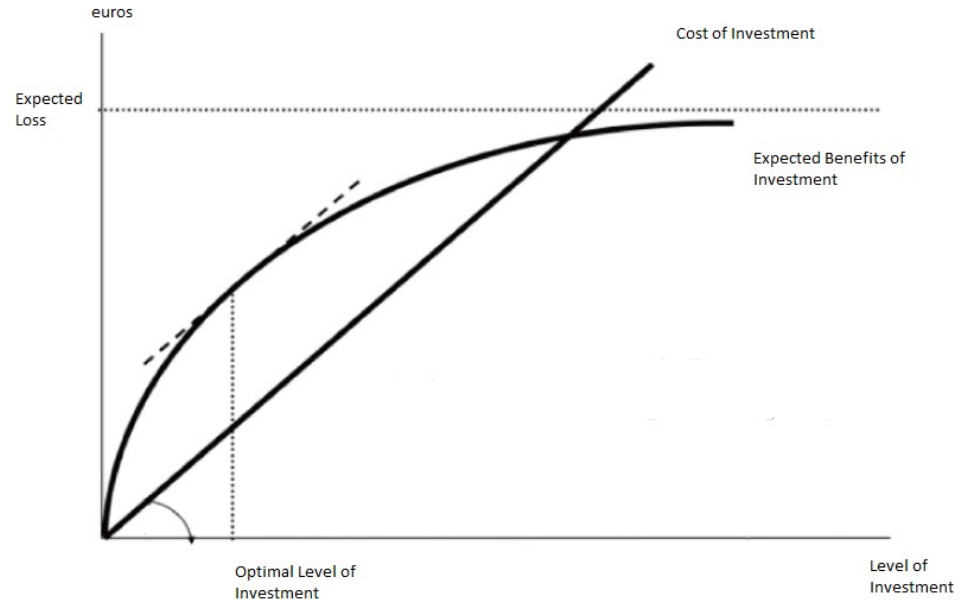
# Underinvestment in Cybersecurity

- “Employees say that under [CEO] Mr. Thompson ... every part of the business was examined for cost savings and **common security practices were eschewed because of their expense**. His approach helped almost triple SolarWinds’ annual profit margins to more than \$453 million in 2019 from \$152 million in 2010.”
- Bruce Schneier: “The market does not reward security, safety or transparency. It doesn't reward reliability past a bare minimum, and it doesn't reward resilience at all.”
- Core problem: limited economic incentives to invest in cybersecurity
  - Expense with diminishing returns
  - Limited legal liability
  - Small factor in customers’ decisions => small effect on share price

# Gordon-Loeb model

- Even with optimal incentives, firms will never invest more than 37% of expected damage from security breaches in cybersecurity
- Cybersecurity not generating profit and having diminishing returns on investment

Result: total damage caused by cybersecurity will always significantly exceed investment in cybersecurity



# Legal liability

- Having poor cybersecurity is legal
- Limited laws regulating cybersecurity standards
- Federal Trade Commission relies on “unfair or deceptive acts” to press charges
- Customers and shareholders need extreme cases of negligence or false statements
  - Class-action lawsuit against SolarWinds by shareholders, but only because they allege false and misleading statements
- As long as an honest effort is made, very little legal risk in having bad cybersecurity



# Lack of business consequences

- Over time, customers tend to forgive and forget data breaches
- Equifax, eBay, Adobe, and Marriott all recovered from their breaches
- In corporate context, incentives in procurement favour functionality and cost over possible cybersecurity risks
- Difficult to evaluate cybersecurity between companies
- Share prices usually drop heavily after a data breach, but studies show a negligible long-term effect
- More recent data breaches have had smaller share price drops due to “breach fatigue”

# Problem Recap

- Lack of economic incentive and legal liability
- Global cost of cybercrime \$6 trillion
  - Cybersecurity *spending* is about 1% of this globally
  - Theoretical limit is 37%
- Potential Solutions?

# Responsible Disclosure

- What happens if someone discovers a vulnerability in software?
  - 2008: MBTA sued three MIT students to prevent them from giving a talk about vulnerabilities in the subway fare system
  - 2019: researcher Jonathan Leitschuh discovered a vulnerability in Zoom, which they did not fix until he publicly disclosed it
- Today, many companies have bug bounty programs in place to encourage responsible disclosure of vulnerabilities
- Disclosure deadlines: amount of time researchers give companies to patch vulnerabilities before disclosure
  - Often varies by company and by how critical the vulnerability is

# Breach Notification Laws

- Data breach vs. security breach
  - Data breaches (involving PII) are well-regulated by EU and some U.S. states
  - Lack of PII means 100+ companies affected by SolarWinds are not acknowledging breaches publicly
  - Microsoft 3rd party vendor used to target CrowdStrike not publicly disclosed
- Time frame
  - 48-72 hours standard: is this too rushed?
  - Similar challenges to vulnerability disclosure
  - May be insufficient time to determine who is affected (breach fatigue)
- Who to report to
  - Customers deserve to know about security breaches
  - Can customers effectively evaluate cybersecurity risk?
  - May be preferable to coordinate first
- Legislation expected to be introduced
  - Rep. Bennie Thompson during hearings: “growing interest in a cybersecurity reporting law”, “we can enact incident notification legislation in the short order”
- Good first step, but problems remain

# Greater Liability

- How much responsibility should a firm have?
  - Company pays anyone impacted by breach
    - Helps to shift burden from customers to the firm
  - Company pays the government when there is a breach
  - What should the standard be?
  - Punitive damages?
- Problem: some mistakes are inevitable
  - Cybersecurity is offense-dominant
  - Was a company at fault or just unlucky?

# Cybersecurity Standards

- Create minimum standards and best practices, usually via NIST
- Proposed regulation
  - **Cybersecurity Act of 2012 (failed)**
    - Would have encouraged businesses to follow cybersecurity best practices by providing liability protection
  - **IOT Cybersecurity Improvement Act 2020 (succeeded)**
    - Directs NIST to publish standards that must be followed for Internet of Things devices purchased with government money
- Question: What about just disincentivizing cyber-crime + deterring state actors?
  - Challenges that come with this
  - Will be discussed in future lectures