

**And now  
for something  
completely different...**

**...enabling technologies**





# ***Who Authorises the Authorisers?***

 Advances in data agency and ownership" 



***Cryptography*** is a tool for turning  
lots of different problems into  
***key management problems***

Dr. Lea Kissner, Global Lead of Privacy Technologies at Google



Who Authorises the Authorisers?

***Brooklyn Zelenka @expede***

Who Authorises the Authorisers?

**Brooklyn Zelenka @expede**



[github.com/expede](https://github.com/expede)  
Vancouver 🇨🇦

# Who Authorises the Authorisers?

**Brooklyn Zelenka @expede**

- ♦ Researcher-in-Residence @ Ink & Switch 🐝
  - ♦ Lead the Keyhive (authz) project
- ♦ Spec editor at UCAN Working Group
- ♦ Prev. Ethereum core dev, mostly EVM but also access control
- ♦ PLs and DSys are my jam 🙌
- ♦ Many thanks to the organisers for making remote work 🙏🙏🙏



[github.com/expede](https://github.com/expede)  
Vancouver 🇨🇦



# Who Authorises the Authorisers

## *Topics*



# Who Authorises the Authorisers

## *Topics*

- ♦ *"How do we maintain **privacy** and **security** without handing over all of our trust to a single org?"*
- ♦ Core definitions / concepts
- ♦ Delegated auth
- ♦ Encryption
- ♦ Defence in depth
- ♦ Keeping the technical depth to high level diagrams
  - ♦ (...but we can go as deep as y'all want 🤪)





Who Authorises the Authorisers

***"Comparable Experience"***

If we do our job "perfectly", you don't **have to** know something is different



# Who Authorises the Authorisers

## "Comparable Experience"

Manage access

Create teamAdd peopleAdd teams

Direct accessOrganization access

Select all

role:admin

TypeRole

alexjg

Role: admin

beehive

@inkandswitch/beehive • 3 members

Role: admin

Brooklyn Zelenka

expede

Role: admin

web

@inkandswitch/web • 4 members

Role: admin

Add people, groups, and calendar events

People with access

hello@katiwilde.com

hello@katiwilde.com

Owner

Brooklyn Zelenka (you)

hello@brooklynzelenka.com

Editor

General access

Restricted

Only people with access can open with the link

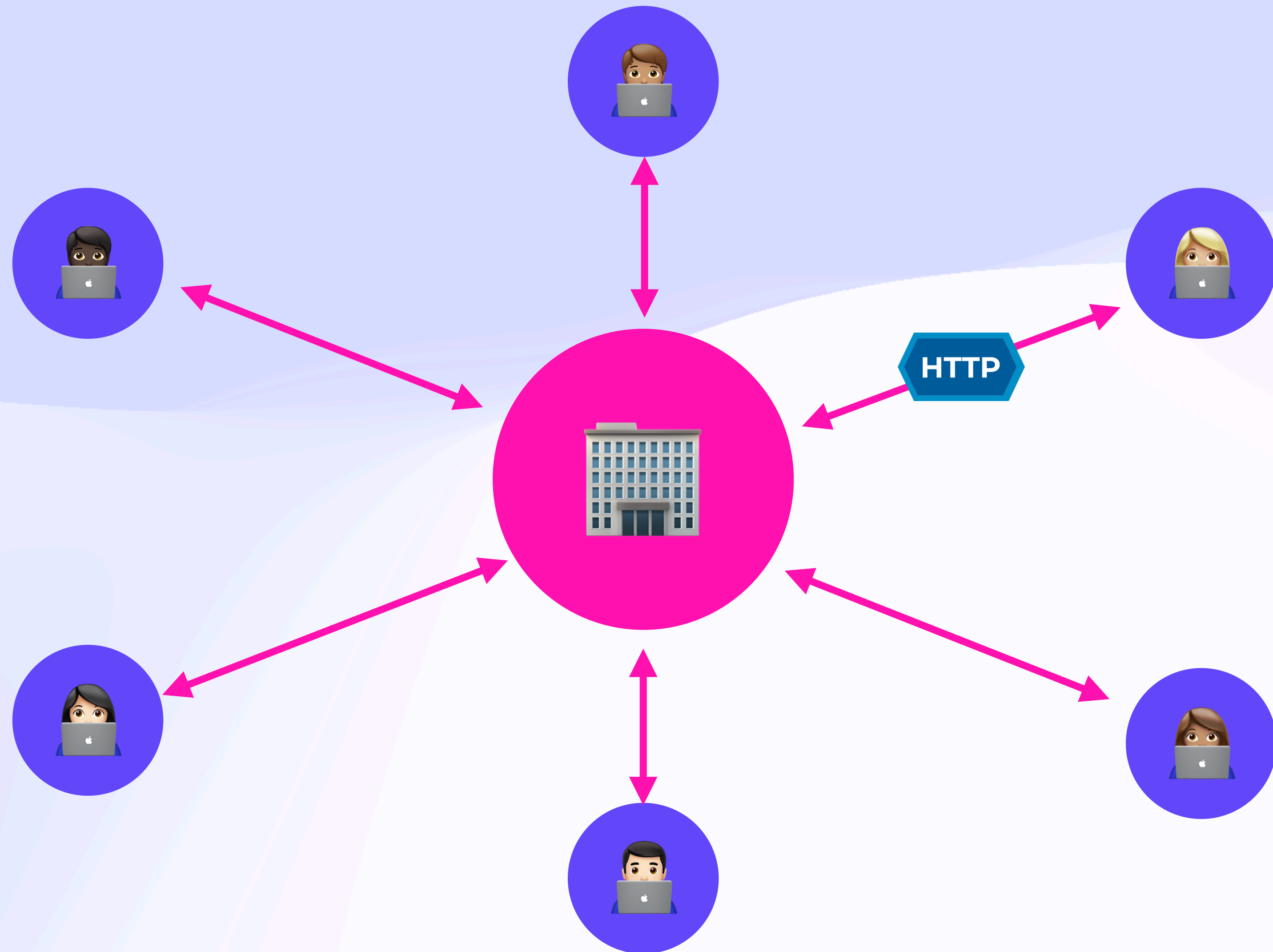
Copy link

Done

If we do our job "perfectly", you don't **have to** know something is different

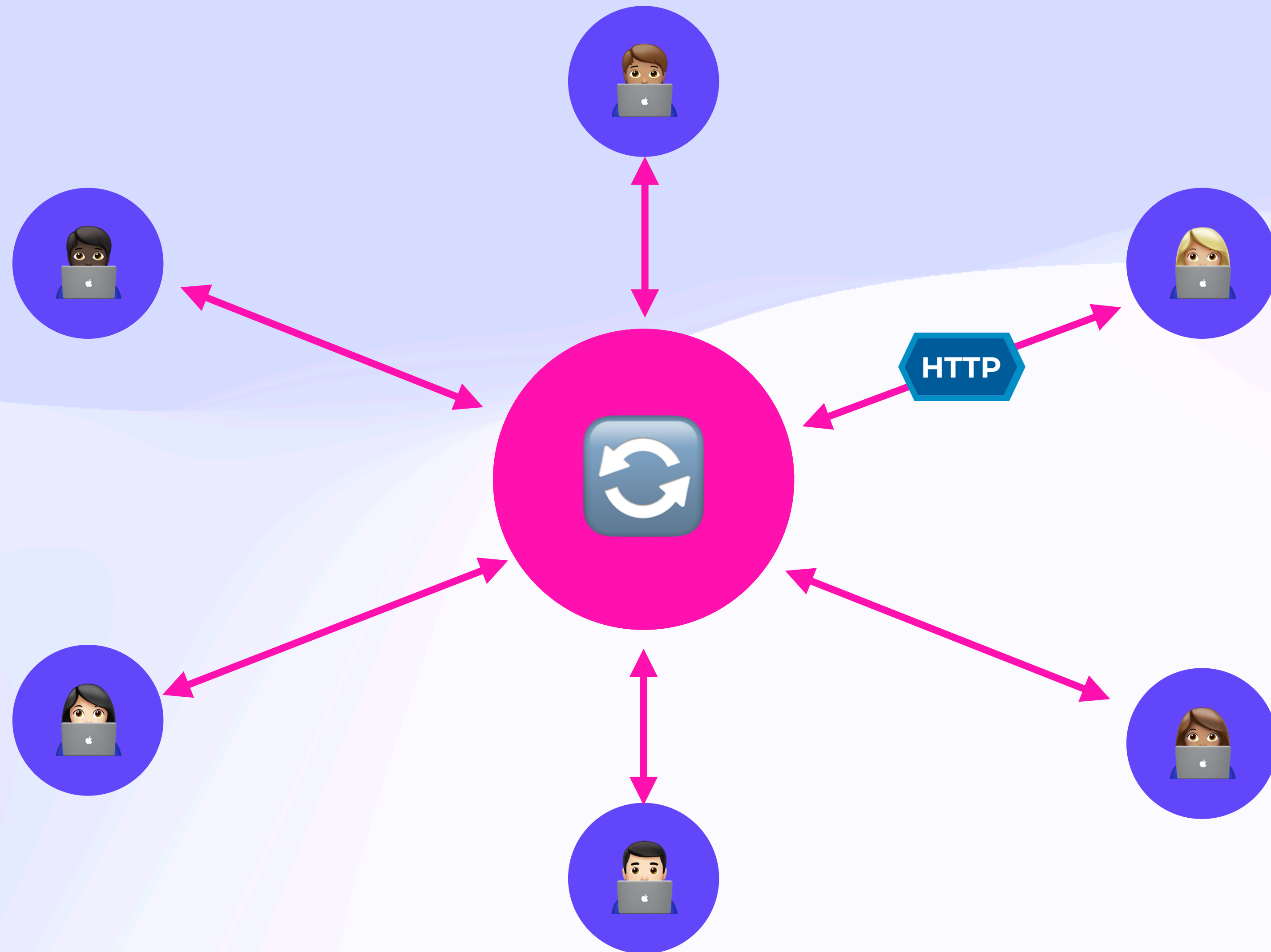
Who Authorises the Authorisers

# ***LoFi & Distributed Web Dynamics***



Who Authorises the Authorisers

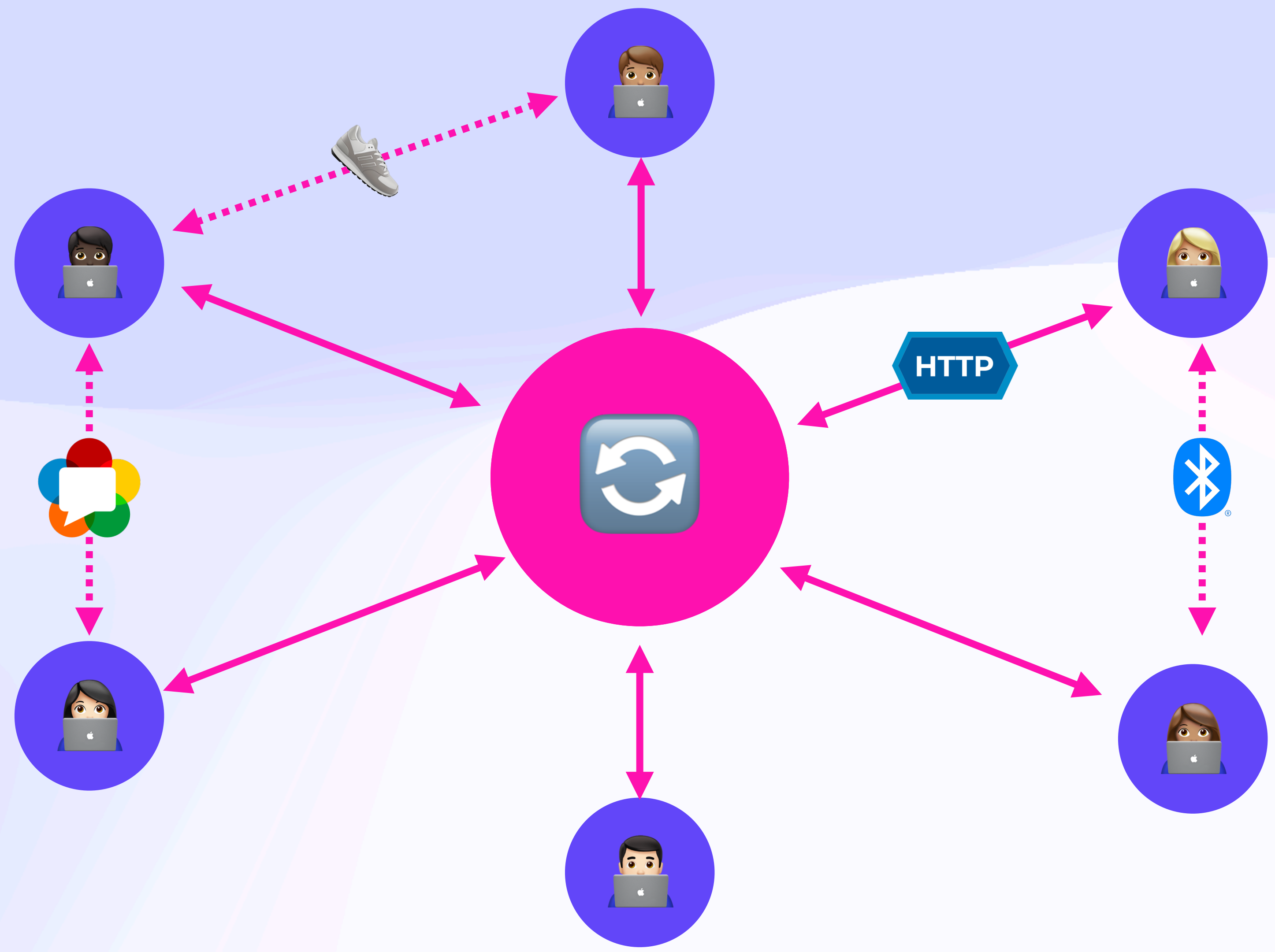
# ***LoFi & Distributed Web Dynamics***





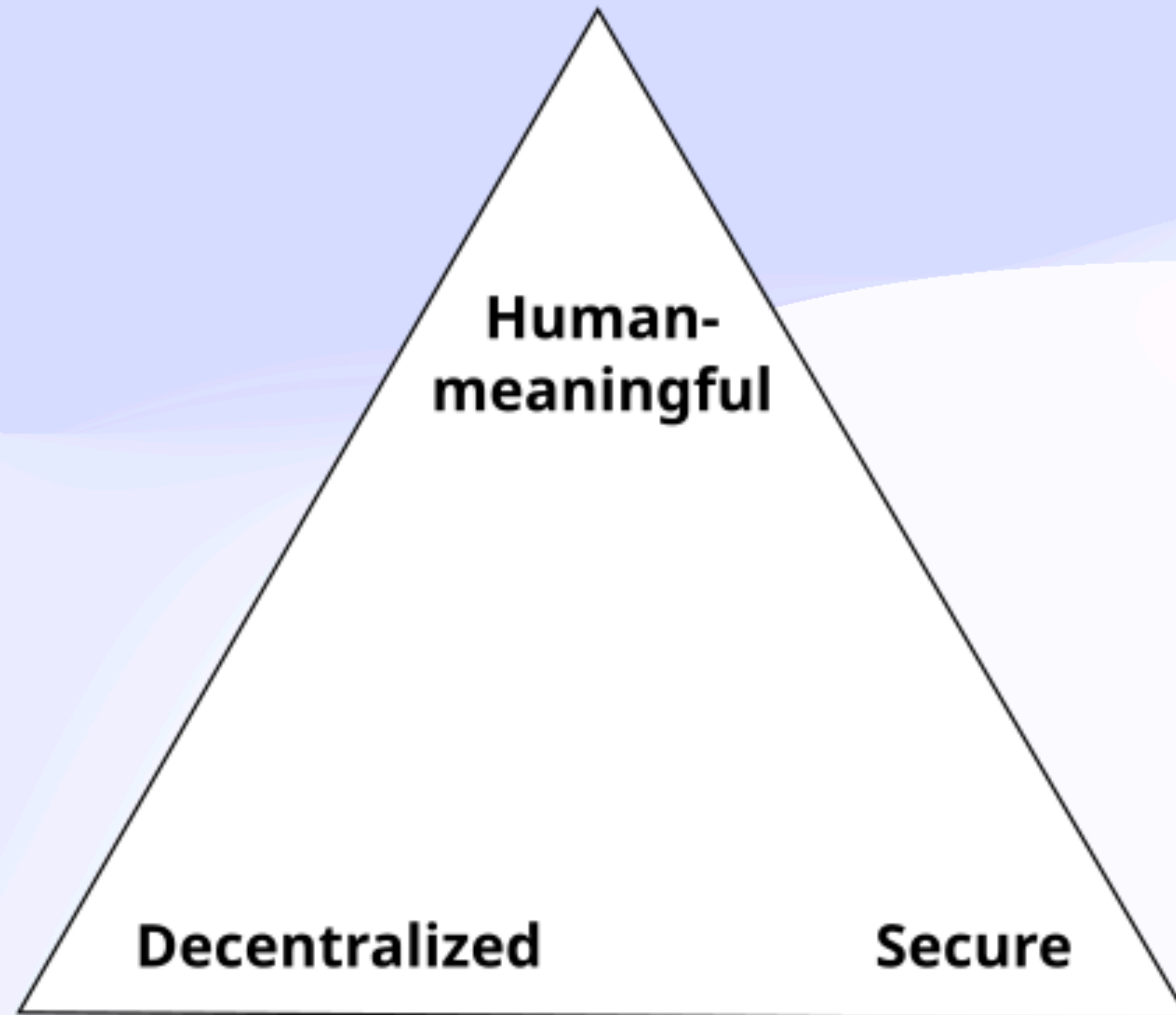
# Who Authorises the Authorisers

## *LoFi & Distributed Web Dynamics*



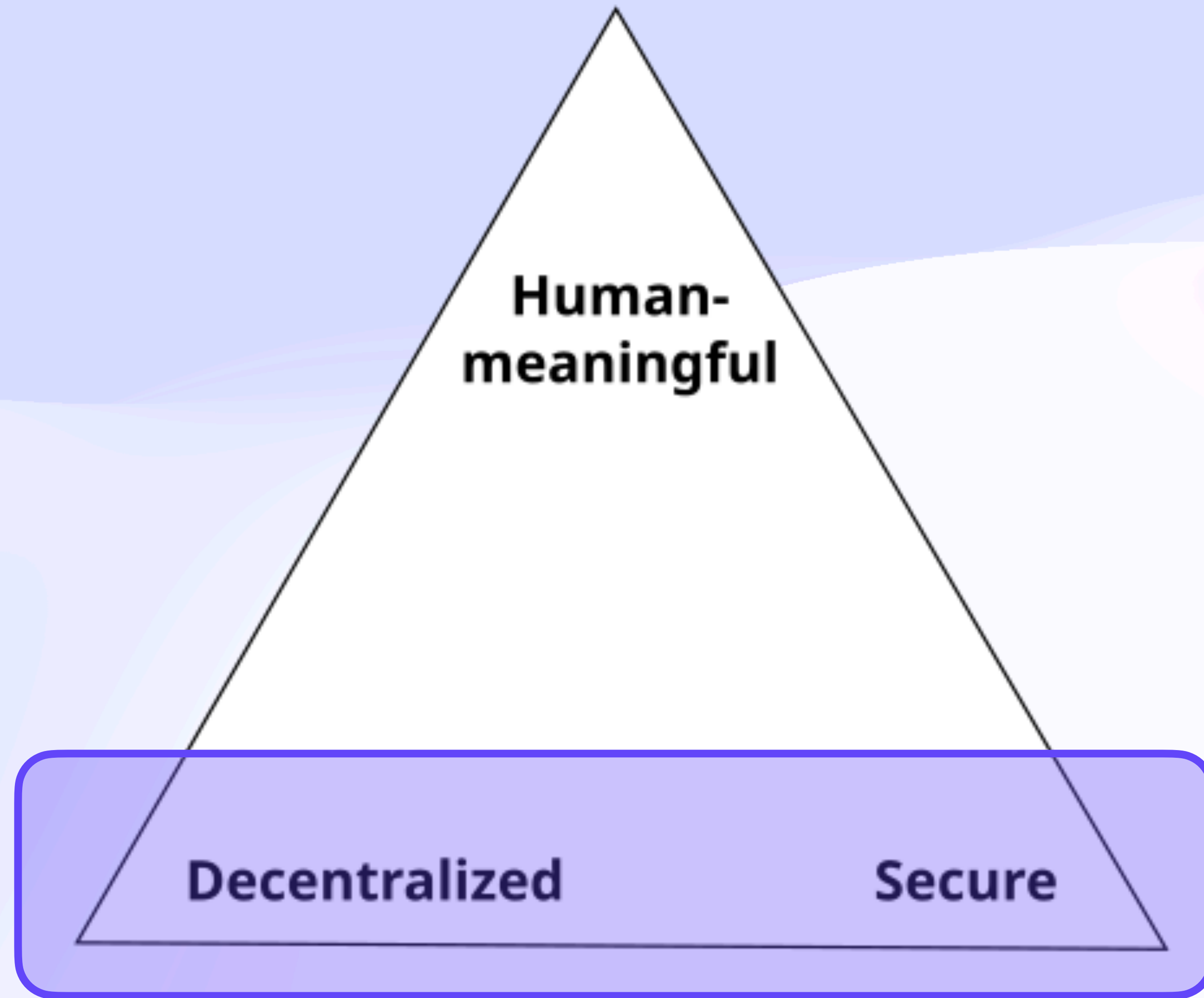
Who Authorises the Authorisers

# ***Zooko's Triangle (User Identifiers)***



Who Authorises the Authorisers

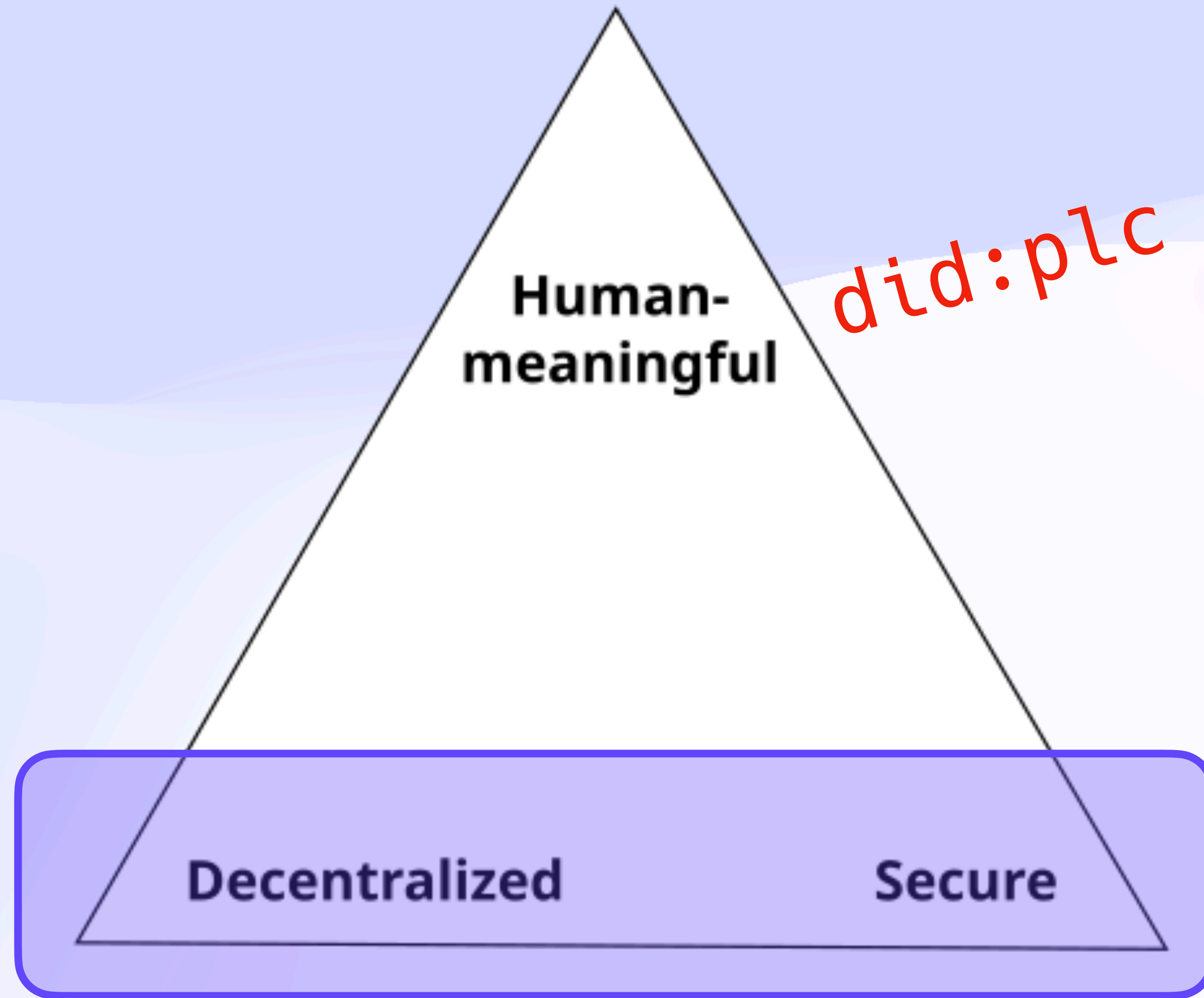
# ***Zooko's Triangle (User Identifiers)***





Who Authorises the Authorisers

# ***Zooko's Triangle (User Identifiers)***



Who Authorises the Authorisers

***AuthN vs AuthZ vs Identity***

Who Authorises the Authorisers

# ***AuthN vs AuthZ vs Identity***

## **AutheNtication**

"Which source did something come from?"



# Who Authorises the Authorisers

## ***AuthN vs AuthZ vs Identity***

### **AuthoriZation**

"Who's allowed to do this?"



### **AuthNtication**

"Which source did something come from?"





# Who Authorises the Authorisers

## ***AuthN vs AuthZ vs Identity***

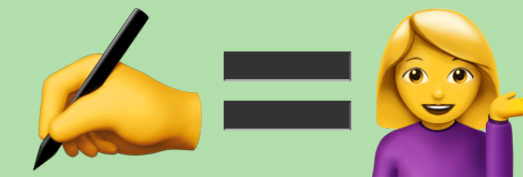
### **AuthoriZation**

"Who's allowed to do this?"



### **Identity (Binding)**

"Which human/org are you?"



### **AuthNtication**

"Which source did something come from?"



# Who Authorises the Authorisers

## ***AuthN vs AuthZ vs Identity***

### **AuthoriZation**

"Who's allowed to do this?"



### **AuthenTication**

"Which source did something come from?"



Who Authorises the Authorisers



***Keyhive at the Highest of Levels***



# Who Authorises the Authorisers

## ***Keyhive at the Highest of Levels***

- Comparable **experience** to cloud auth, minus the servers
- **Control** who can read and write to documents
- Compatible with swappable/**interoperable** sync servers
  - Not store data in plaintext on those servers (encryption at rest)
  - ...but still be efficient: support Automerge compaction
- **"Wikipedia-scale"** (100k+ docs, 10k+ E2EE readers, 1k+ writers, 100s admins)
- Secure "at least at the level of the FBI, but not necessarily NSA, MSS, or Mossad"

 Single Source of Truth 

***Cloud Auth***

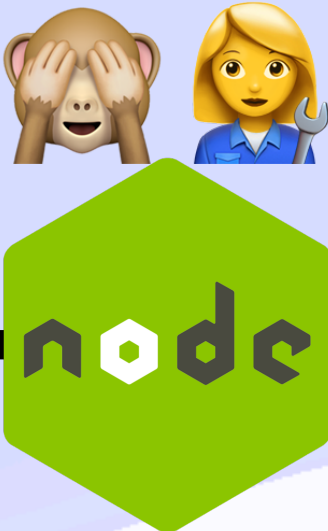
Cloud Auth

***Cloud Auth Flow*** ☁️



Cloud Auth

# Cloud Auth Flow



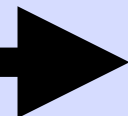
Cloud Auth

***Cloud Auth Flow*** ☁️



Cloud Auth

*Cloud Auth Flow* ☁️





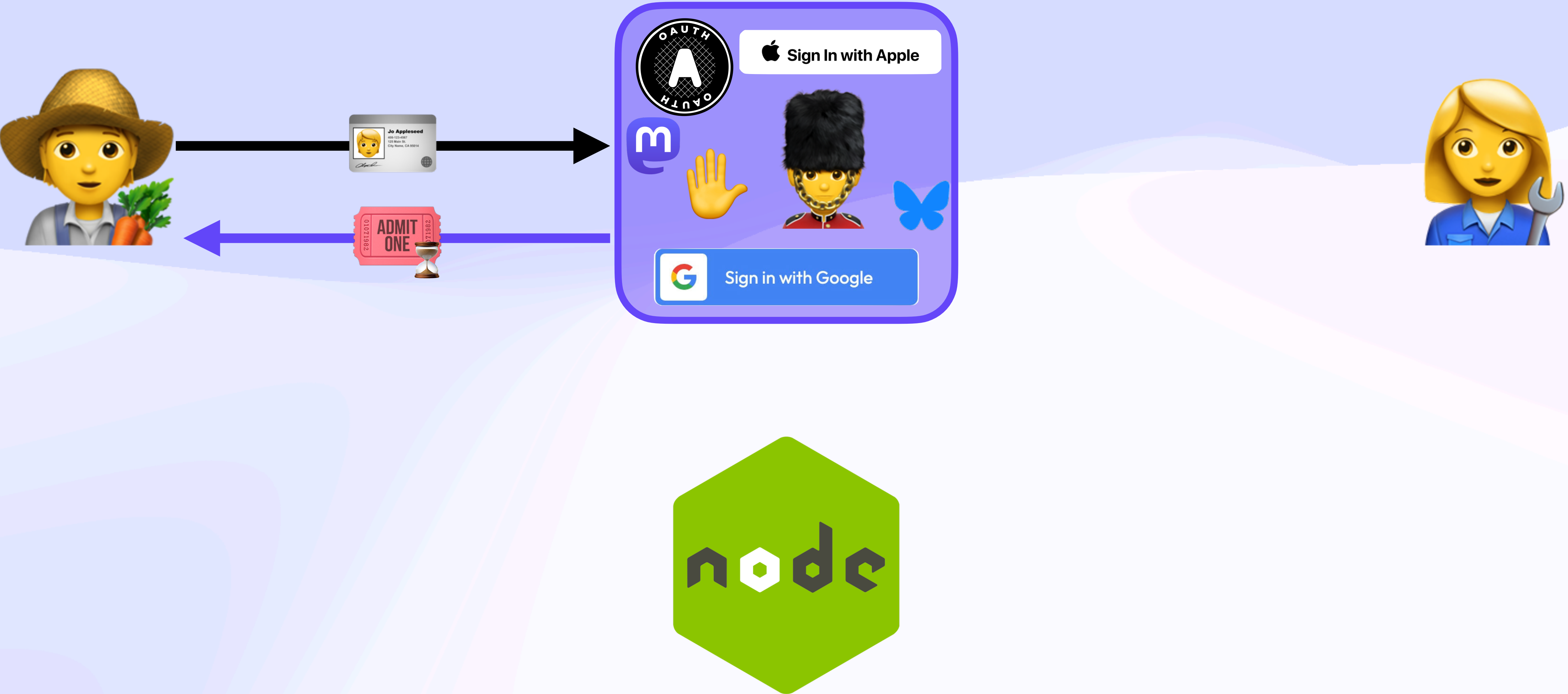
Cloud Auth

*Cloud Auth Flow* ☁️



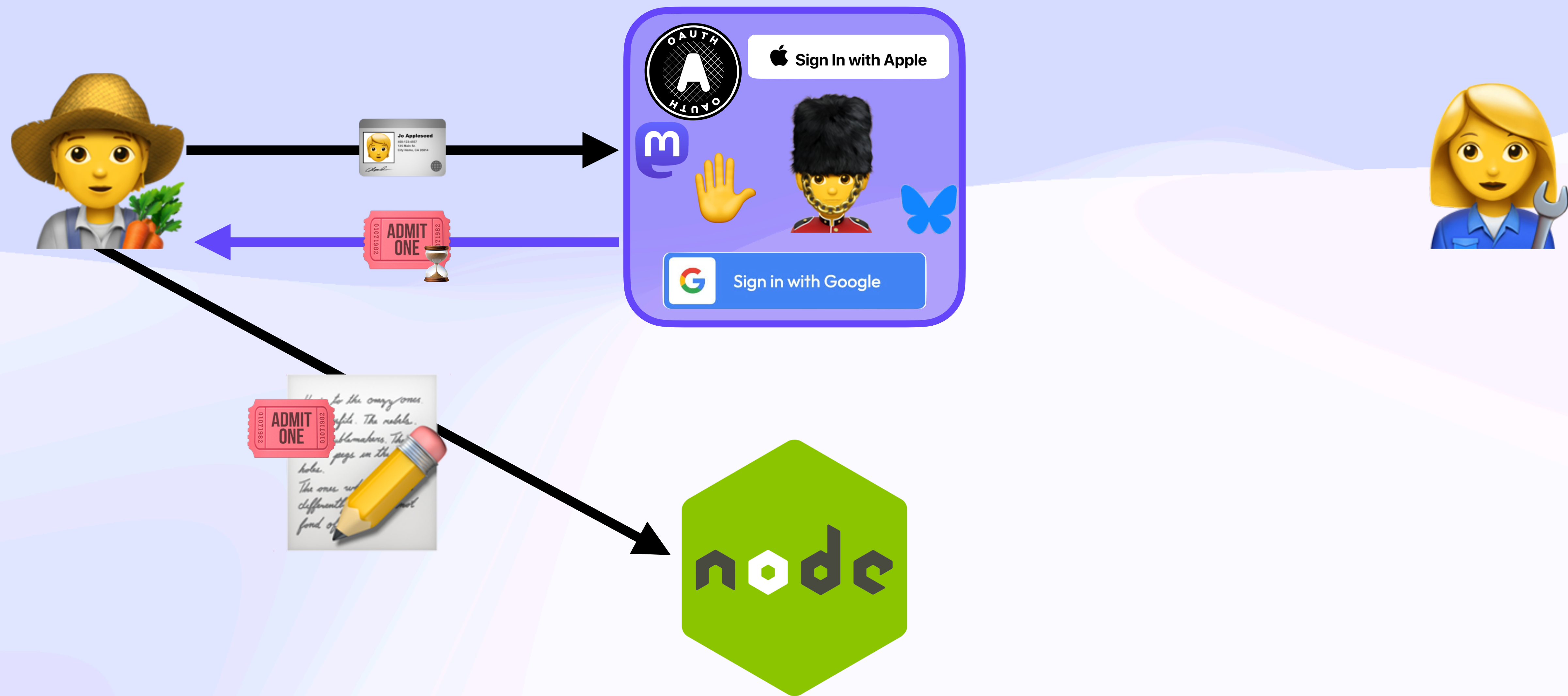
Cloud Auth

# Cloud Auth Flow



# Cloud Auth

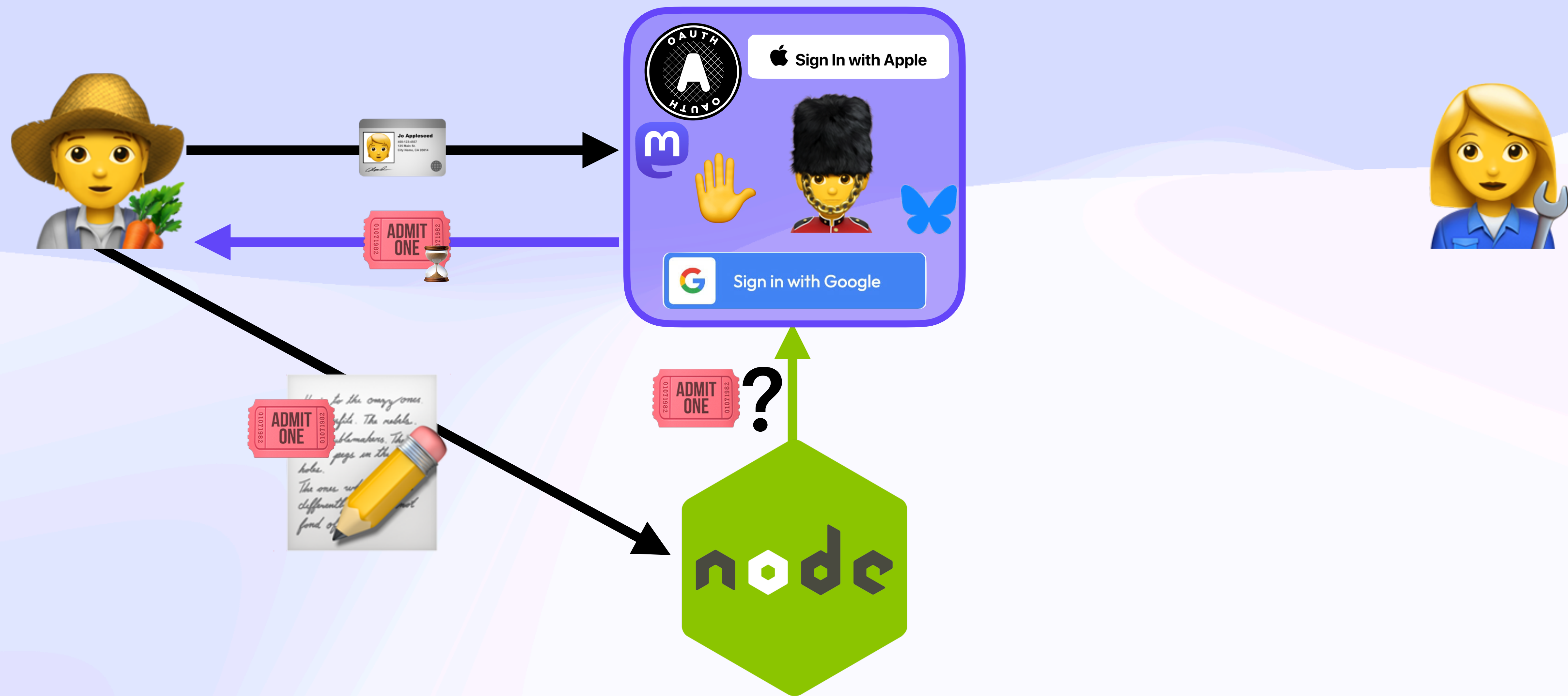
# Cloud Auth Flow





# Cloud Auth

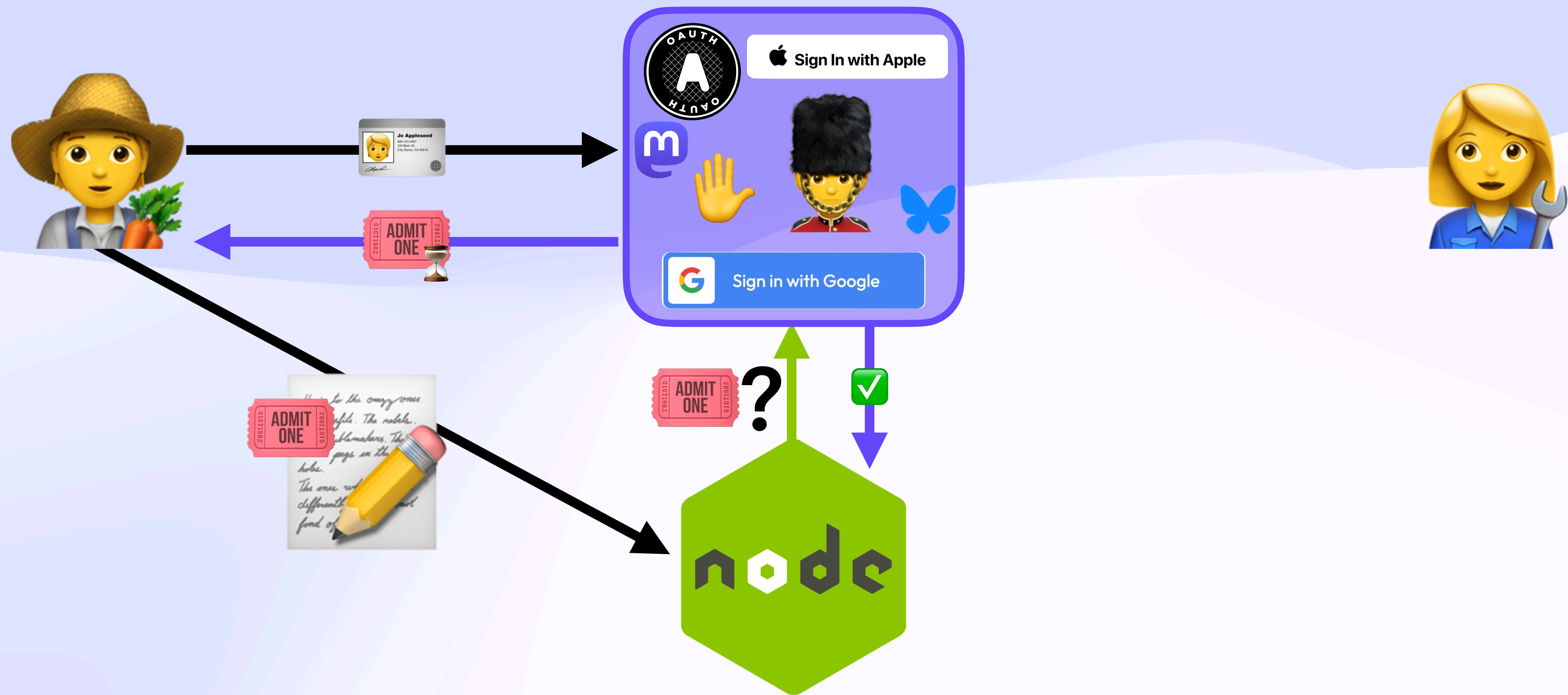
# Cloud Auth Flow





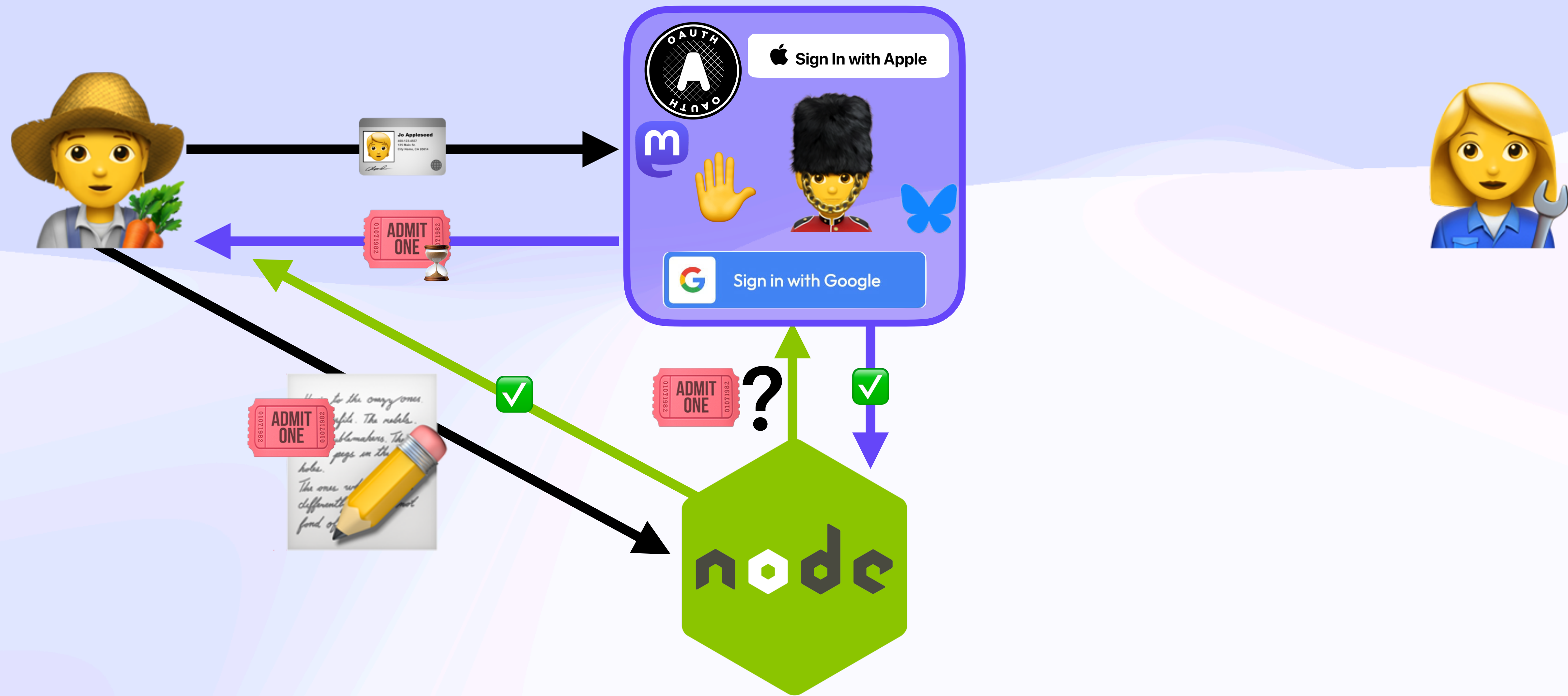
# Cloud Auth

# Cloud Auth Flow



# Cloud Auth

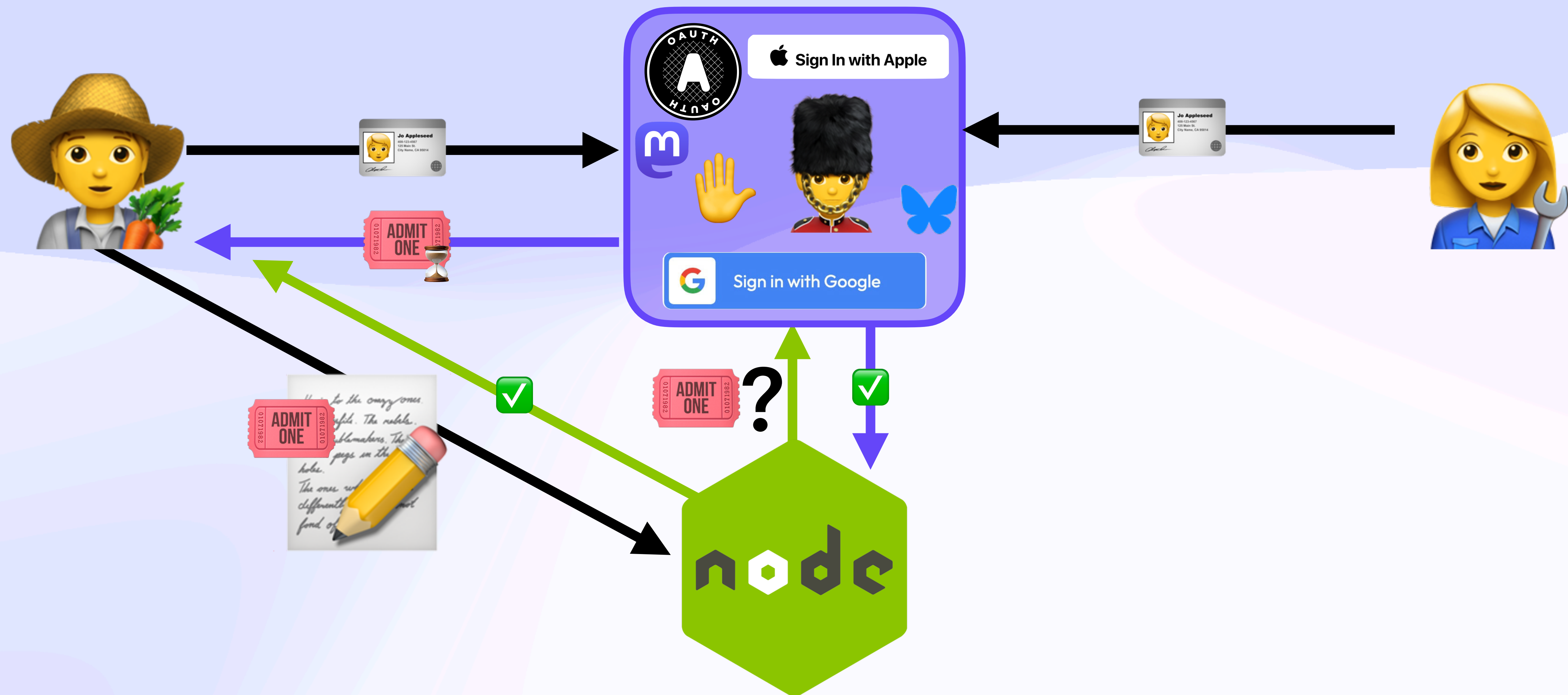
# Cloud Auth Flow ☁️





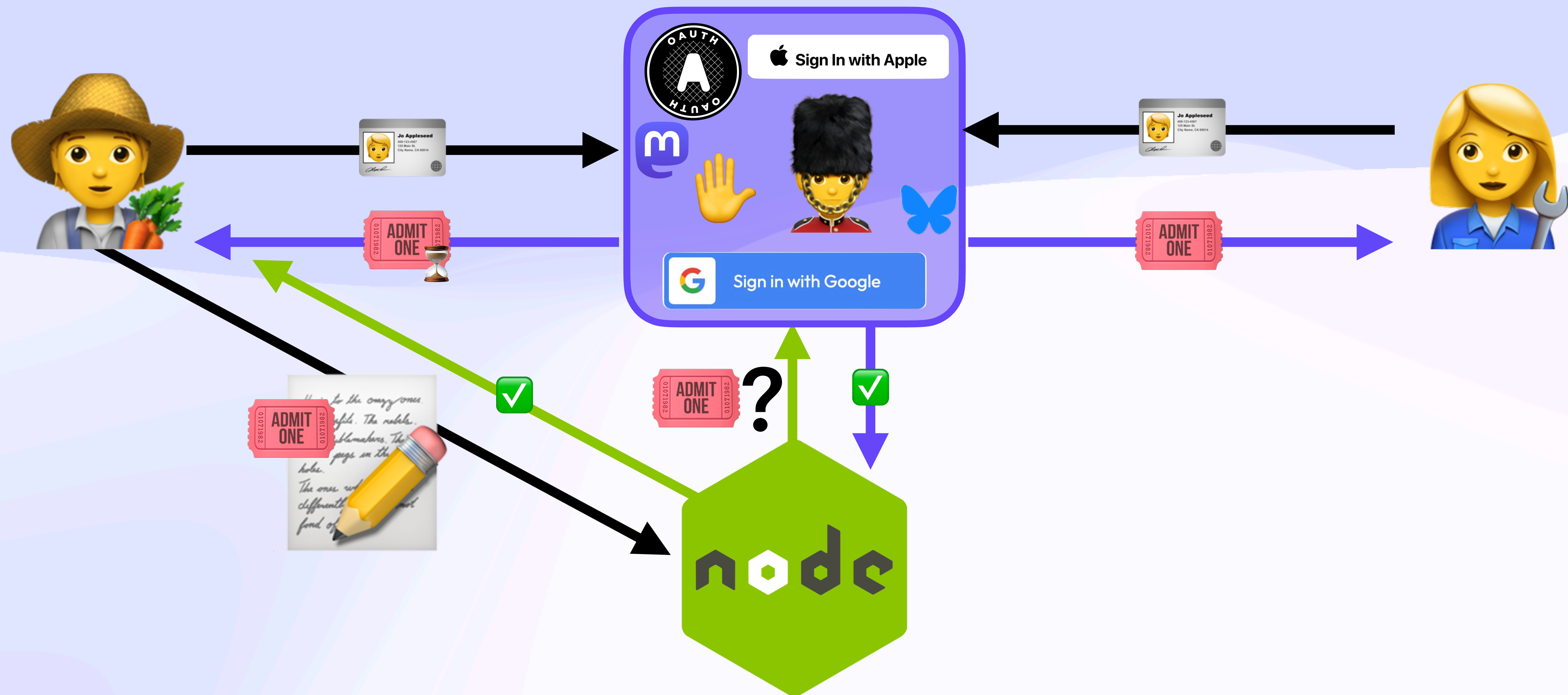
# Cloud Auth

## *Cloud Auth Flow* ☁️



# Cloud Auth

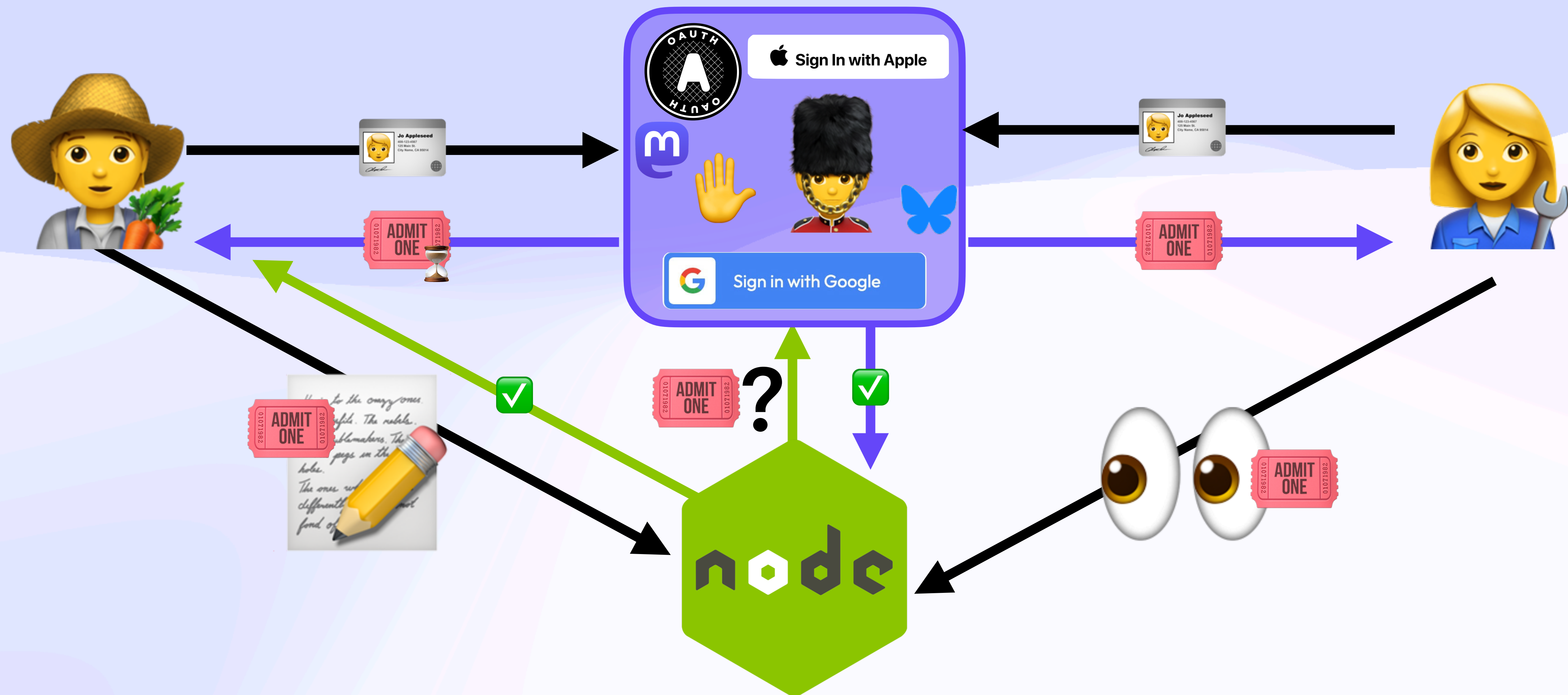
## Cloud Auth Flow ☁️





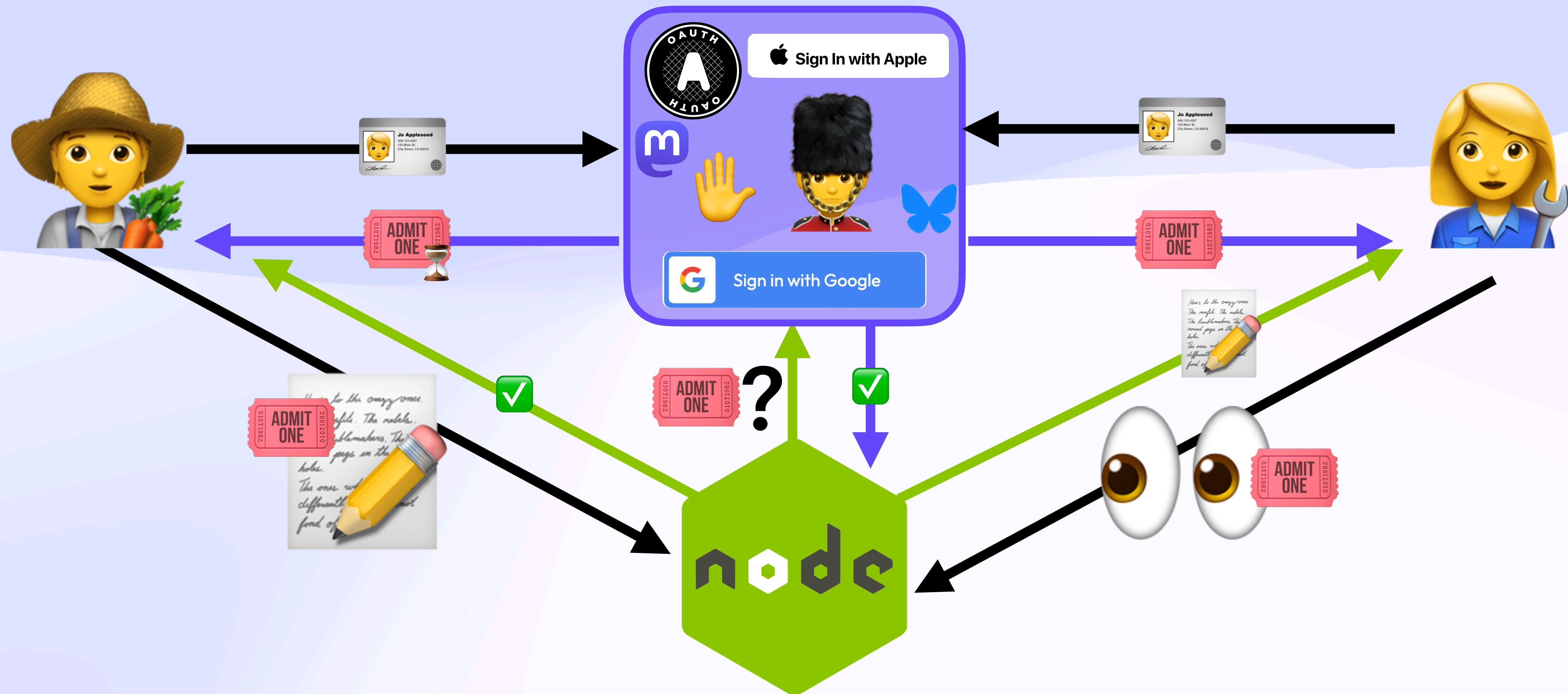
# Cloud Auth

## Cloud Auth Flow ☁️



# Cloud Auth

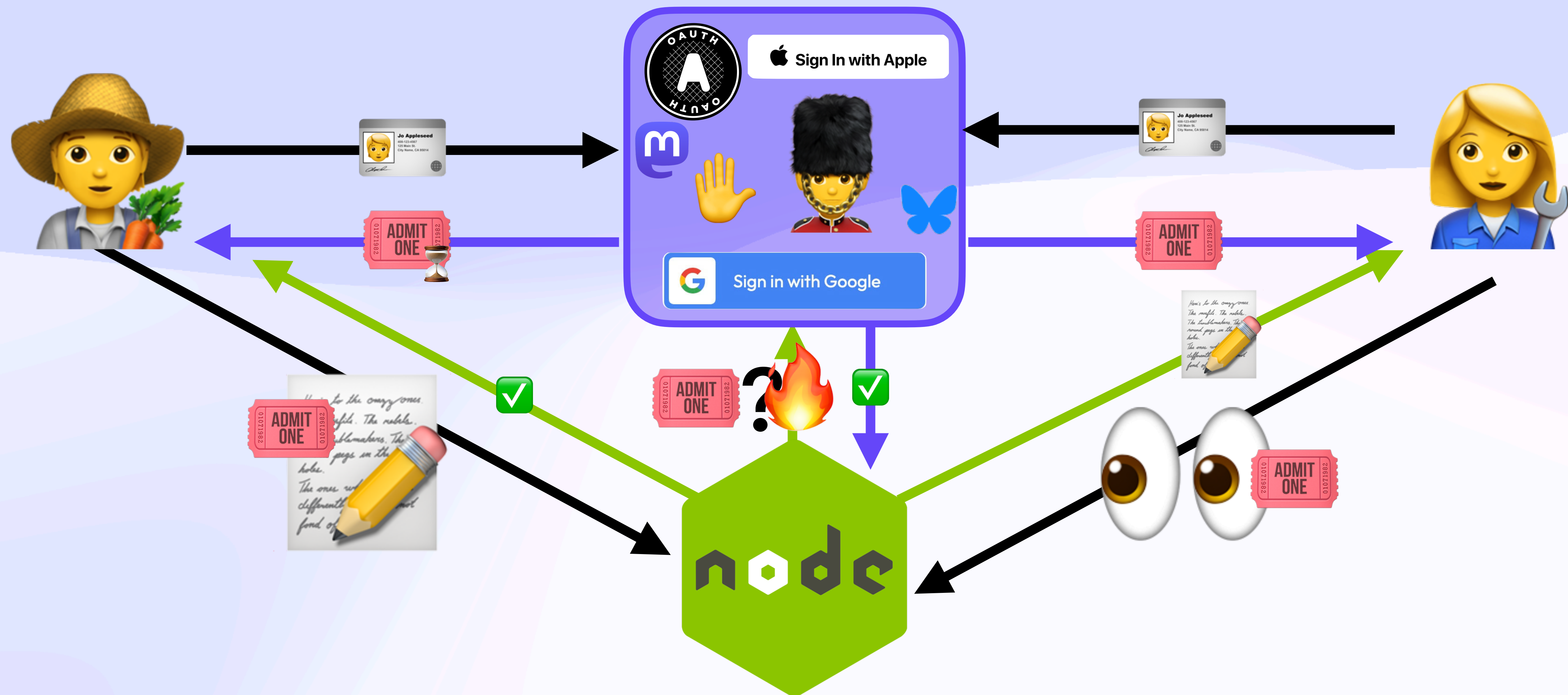
## Cloud Auth Flow





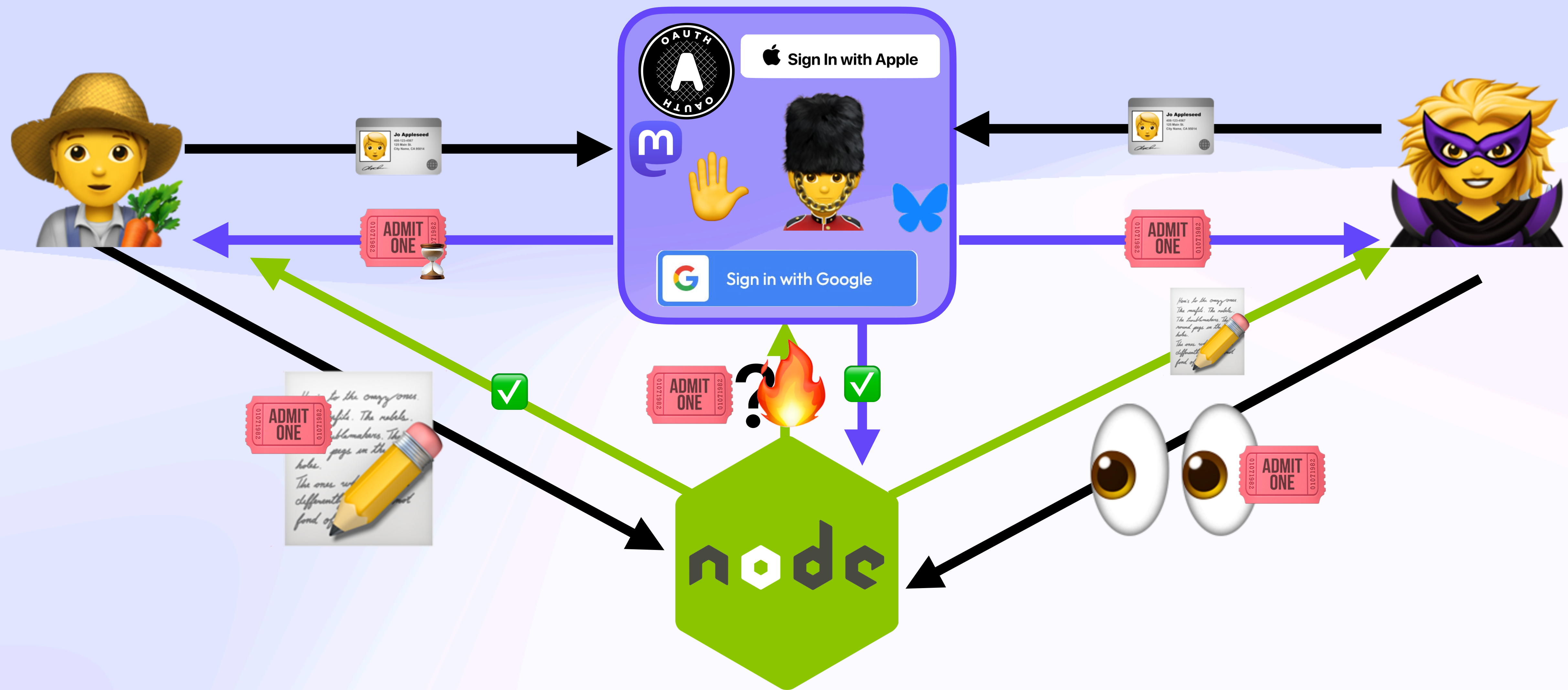
# Cloud Auth

## Cloud Auth Flow ☁️



# Cloud Auth

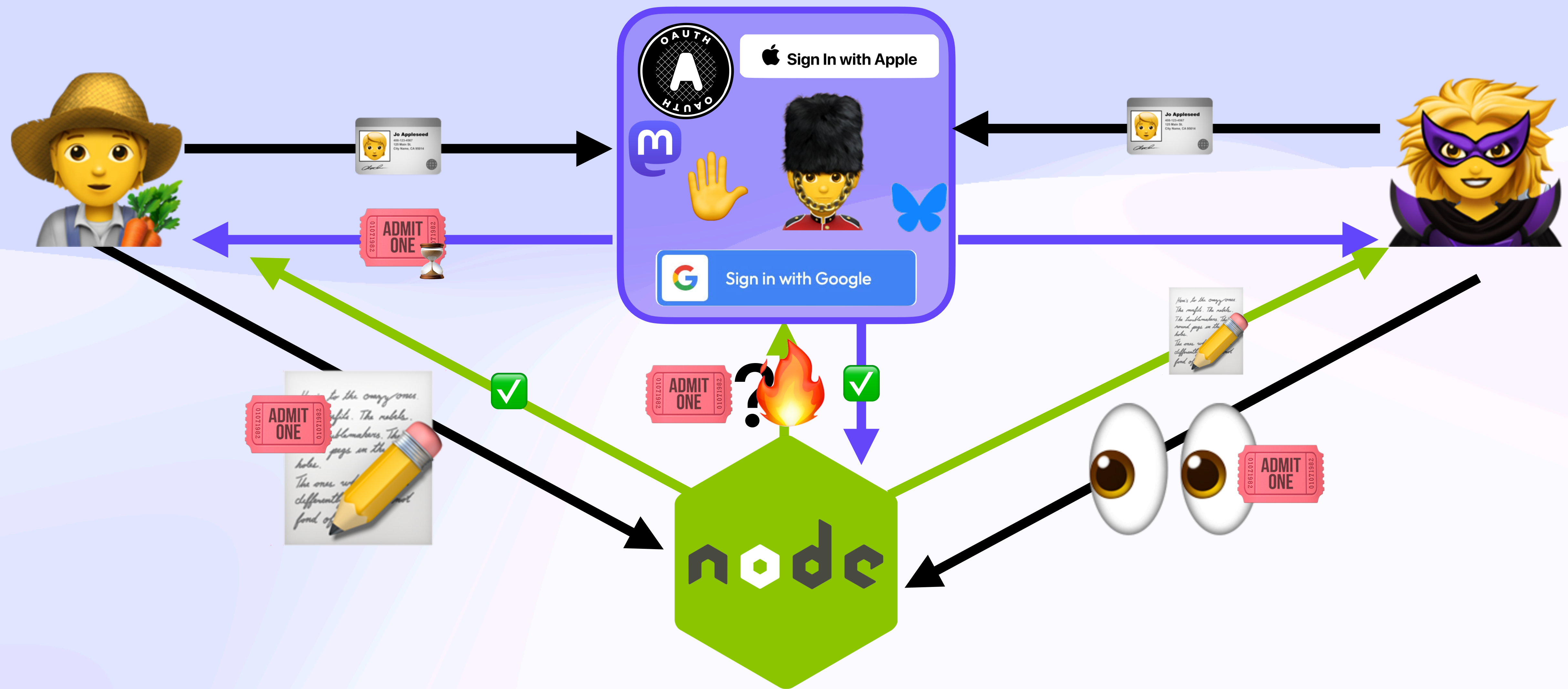
## Cloud Auth Flow ☁️





# Cloud Auth

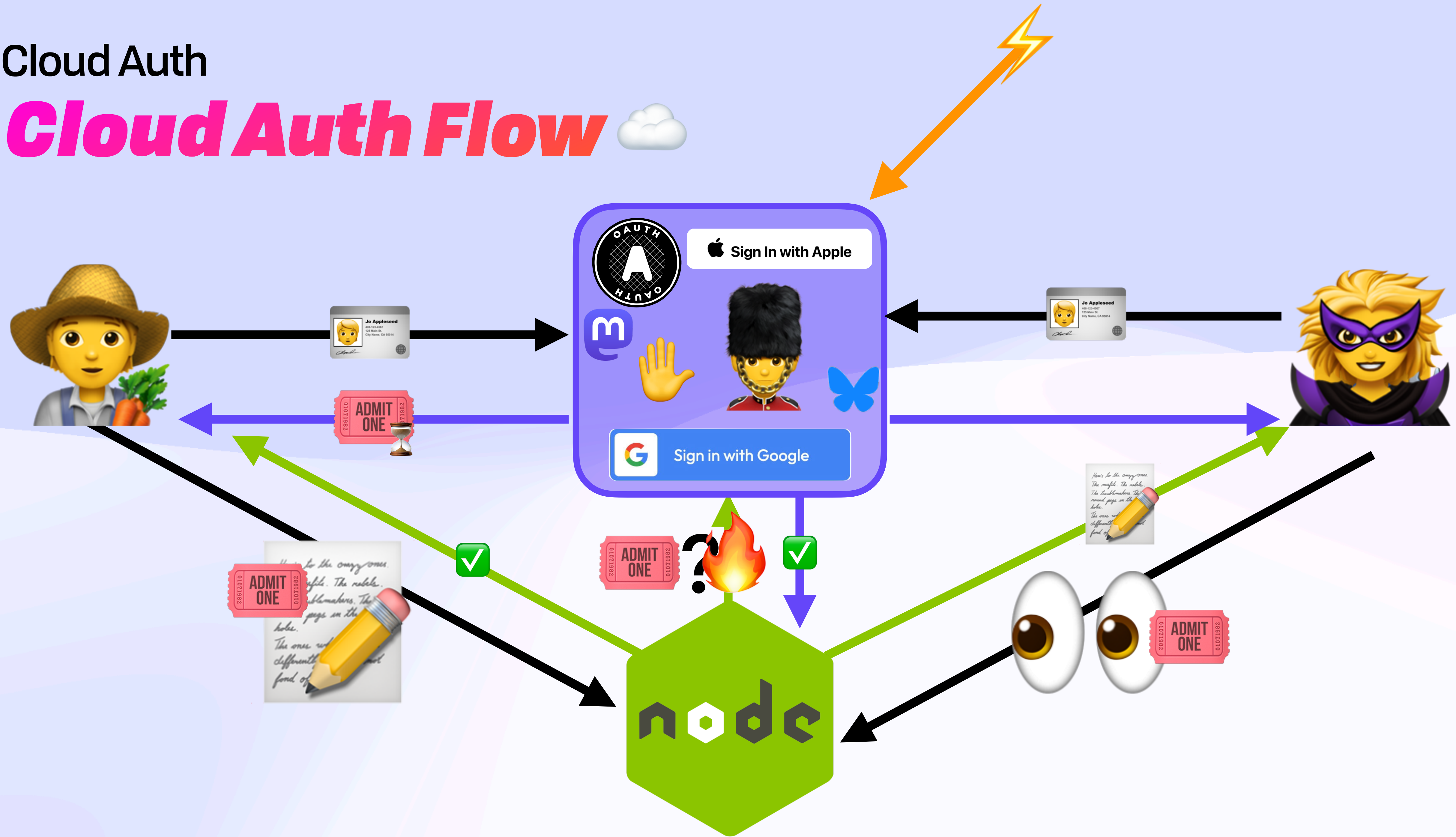
## Cloud Auth Flow ☁️



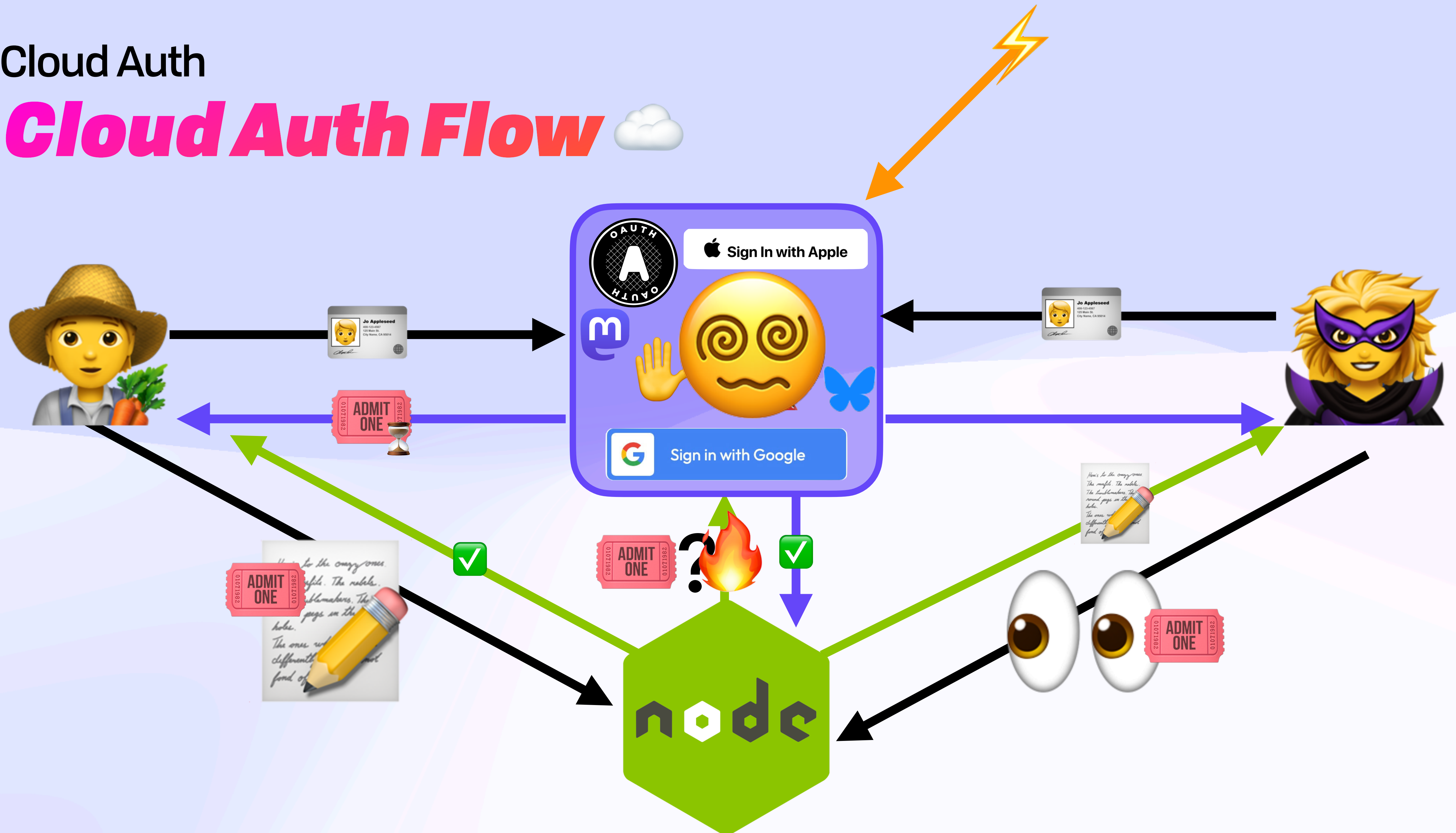


# Cloud Auth

## Cloud Auth Flow



# Cloud Auth Flow ☁



Cloud Auth

***Cloud: Auth-as-Place*** ☁️



# Cloud Auth

# *Cloud: Auth-as-Place* ☁️



# Cloud Auth

# *Cloud: Auth-as-Place* ☁️

"Over Here"



"Over There"





# Cloud Auth

## *Cloud: Auth-as-Place* ☁️

"Over Here"



"Over There"







**SERVERS? WHERE WE'RE GOING, WE  
DON'T NEED SERVERS**





**SERVERS? WHERE WE'RE GOING, WE  
DON'T NEED SERVERS**



Playing By New Rules

***A Different Context***



A Different Context

# ***Local-First in Pictures***

A Different Context

# ***Local-First in Pictures***





A Different Context

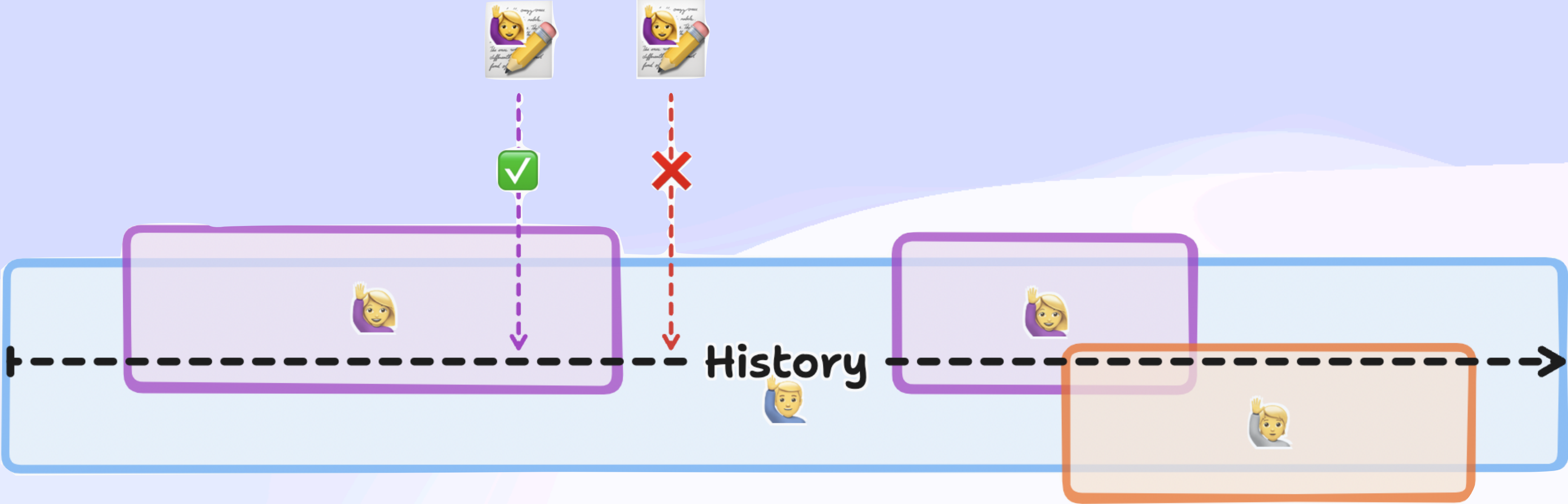
# *Local-First in Pictures*





A Different Context

# Adds & Removals Over Time





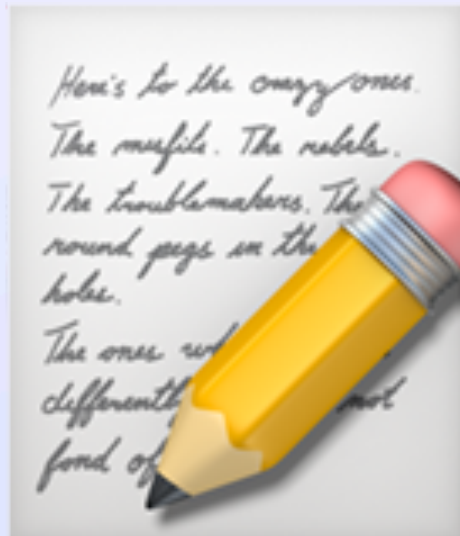
A Different Context

***Auth as Data: "Auth Must Travel with Data"***



A Different Context

# ***Auth as Data: "Auth Must Travel with Data"***



A Different Context

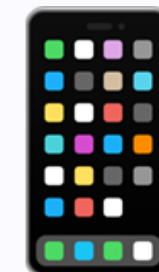
# ***Auth as Data: "Auth Must Travel with Data"***





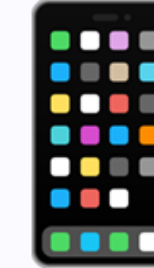
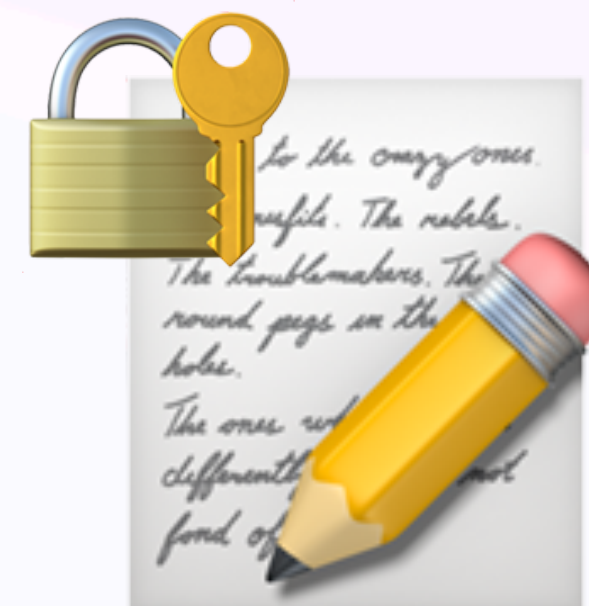
A Different Context

# ***Auth as Data: "Auth Must Travel with Data"***



A Different Context

# ***Auth as Data: "Auth Must Travel with Data"***



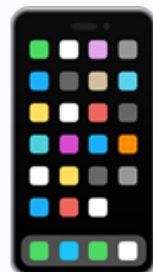
A Different Context

***Auth as Data: "Auth Must Travel with Data"***

"Auth Here"  
"Data Here"



"Auth There"  
"Data There"





# A Different Context

## ***Auth as Data: "Auth Must Travel with Data"***

"Auth Here"  
"Data Here"



"Auth There"  
"Data There"



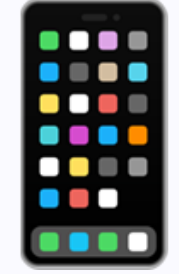
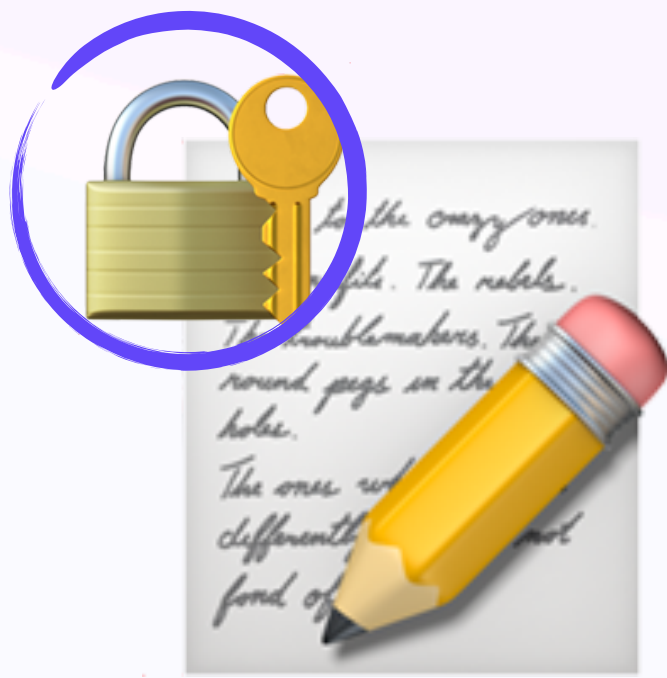
A Different Context

***Auth as Data: "Auth Must Travel with Data"***

"Auth Here"  
"Data Here"



"Auth There"  
"Data There"



# A Different Context



A Different Context

***Cloud***

***Local-First***



A Different Context


*Cloud*

Auth 



Compute 



Data 




*Local-First*

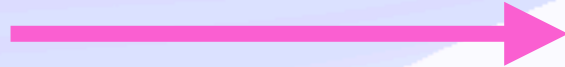
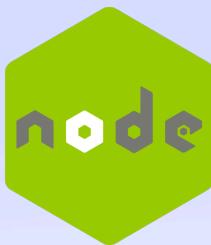
A Different Context

*Cloud*

Auth 


Compute 

Data 



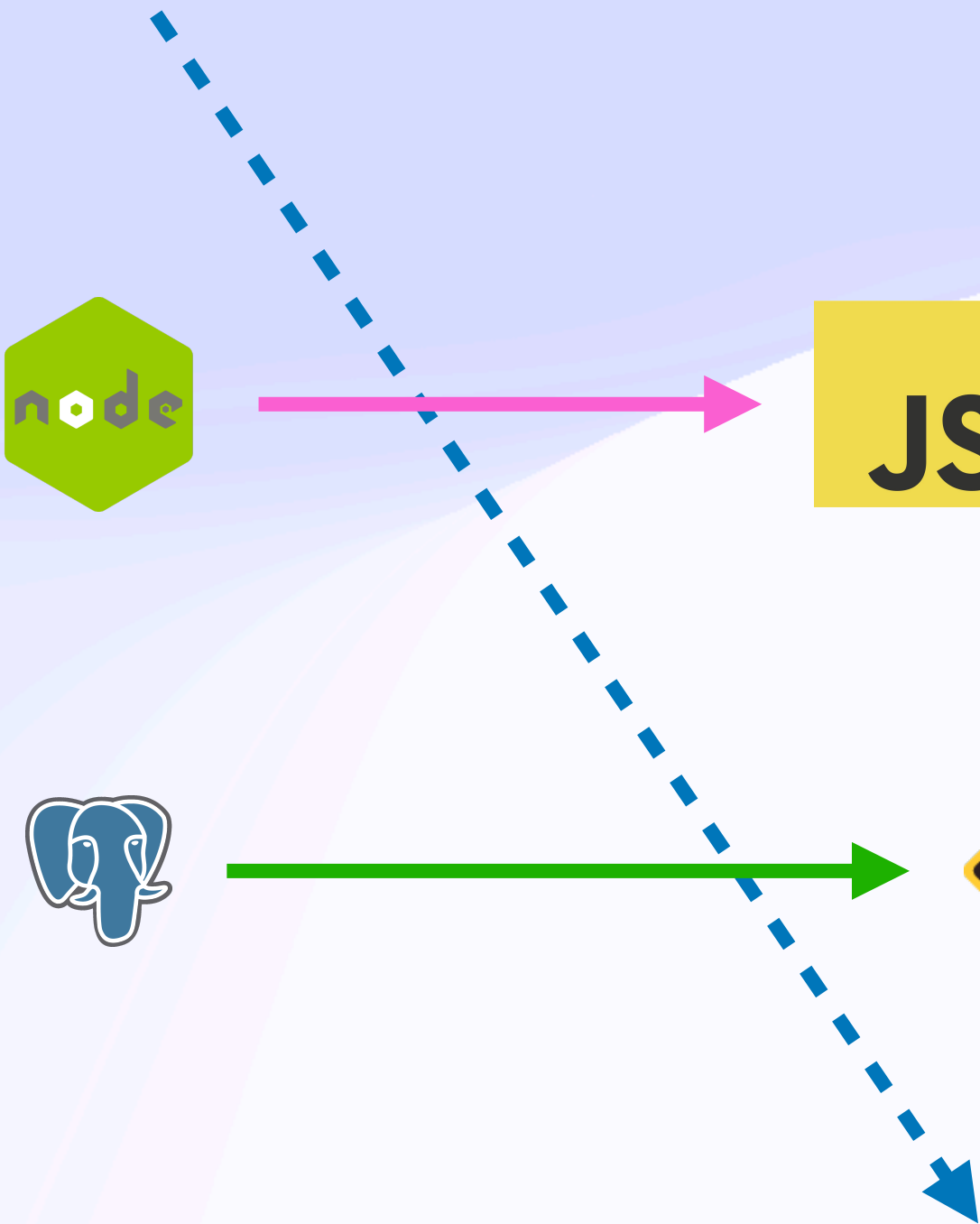
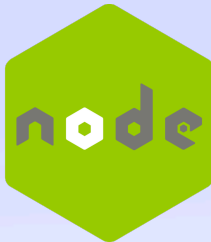
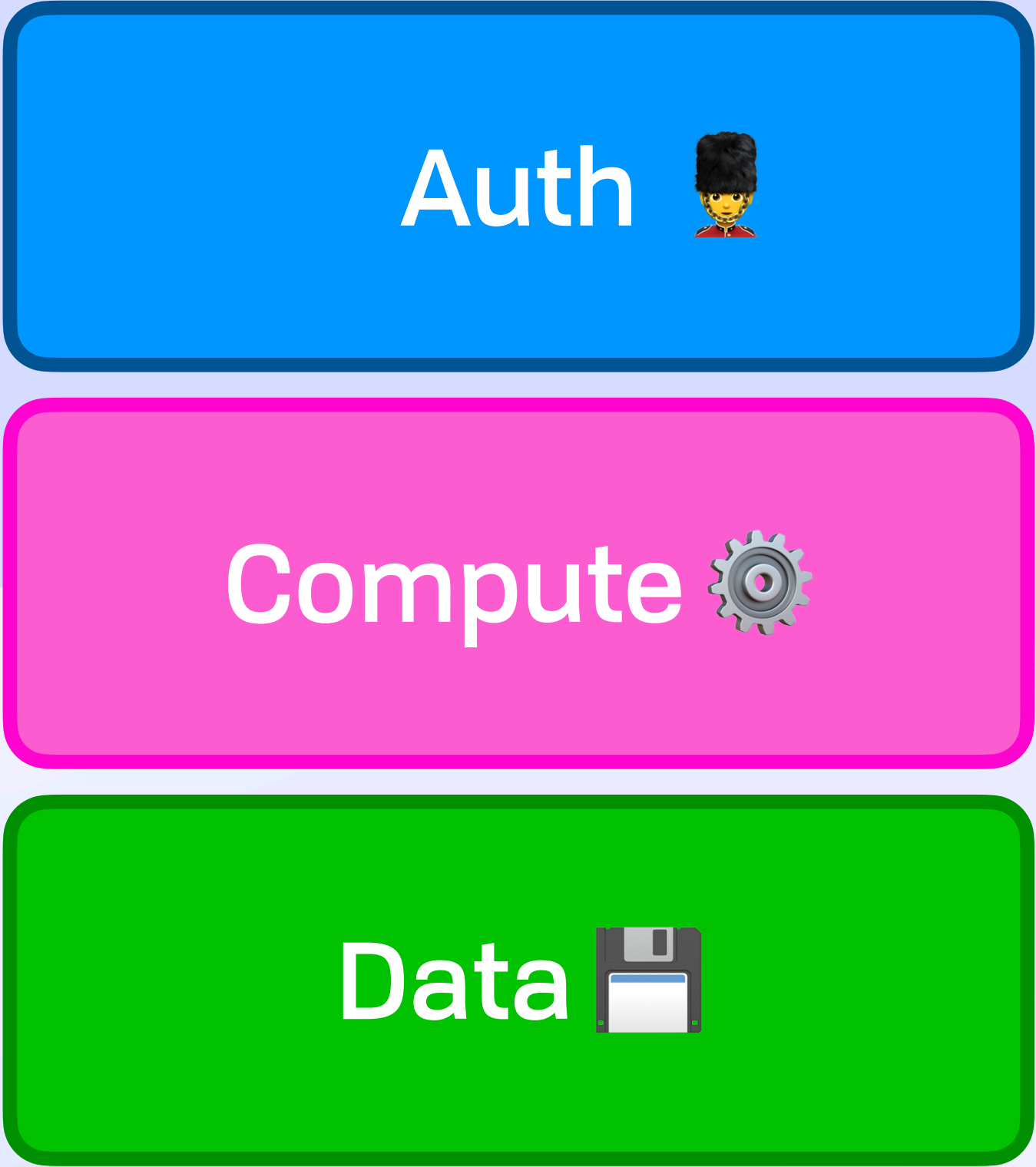
*Local-First*

Compute 

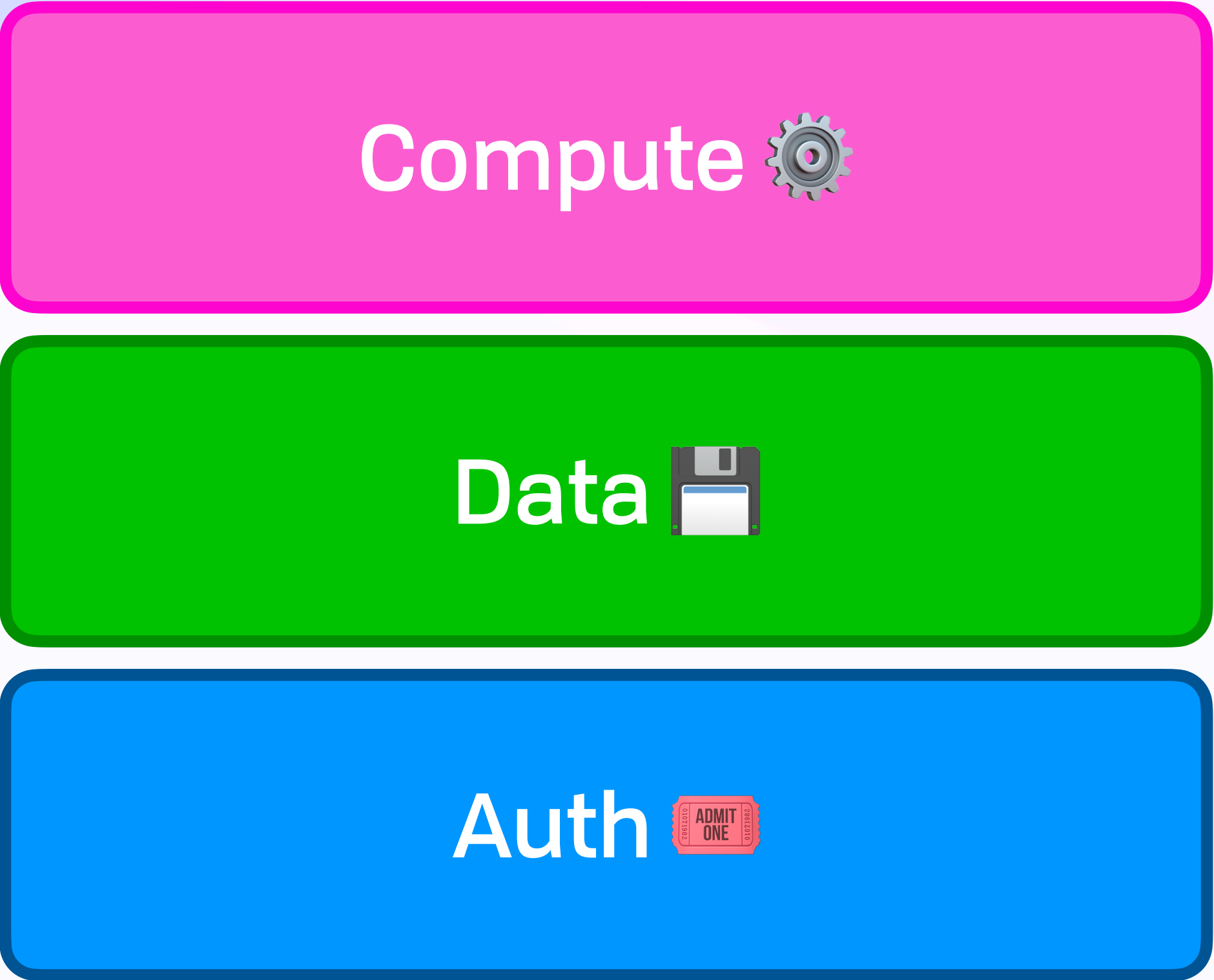
Data 

A Different Context

Cloud



Local-First





Expanding the Beehive, and UCAN Too

# *Convergent Capabilities*



# Convergent Capabilities



# Convergent Capabilities

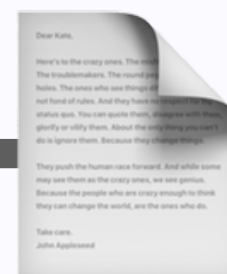
# Copy (Request)





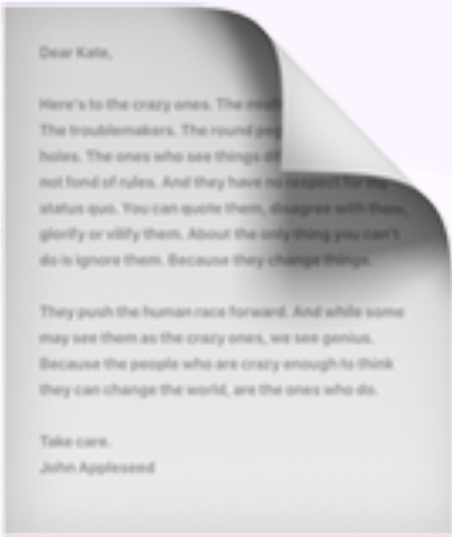
# Convergent Capabilities

# Copy (Request)



# Convergent Capabilities

Read  
(Decrypt)

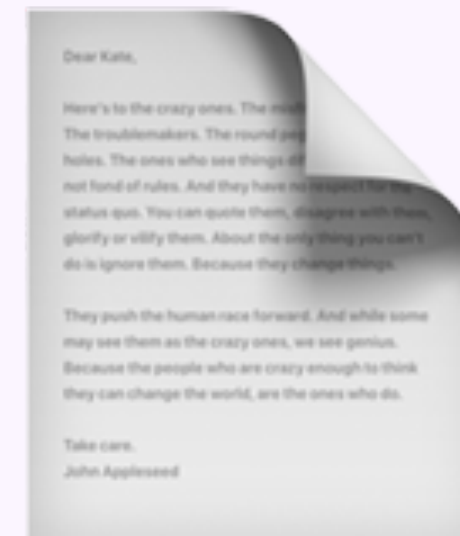
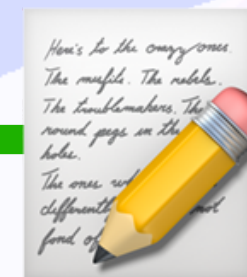


Copy  
(Request)

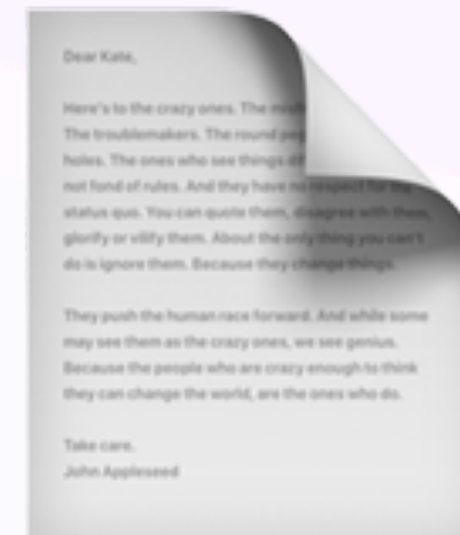
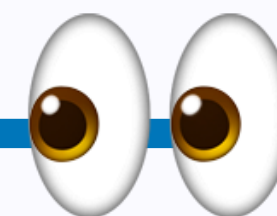


# Convergent Capabilities

Write  
(Update)



Read  
(Decrypt)



Copy  
(Request)



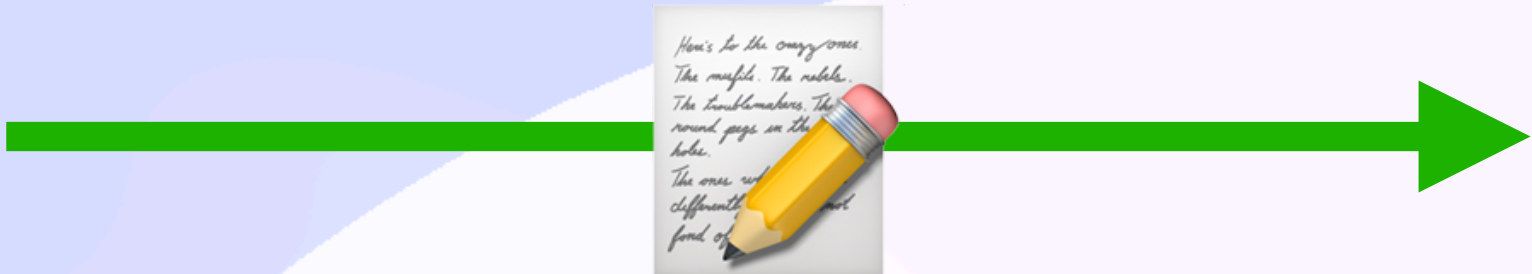


# Convergent Capabilities

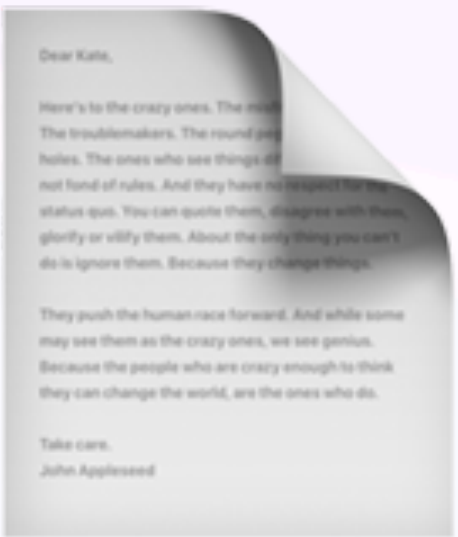
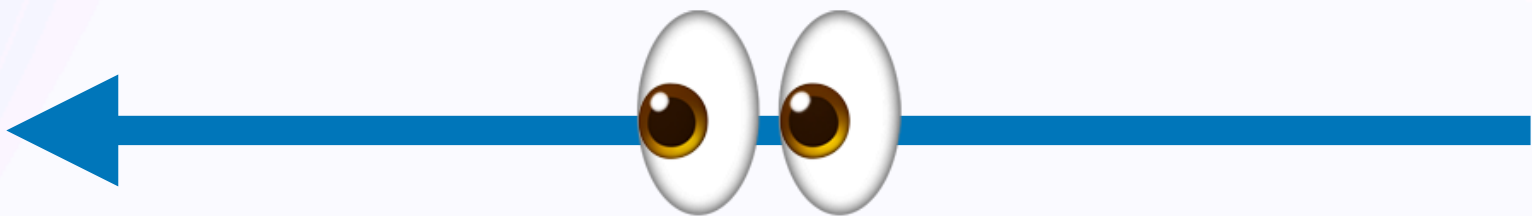
Admin  
(Revoke)



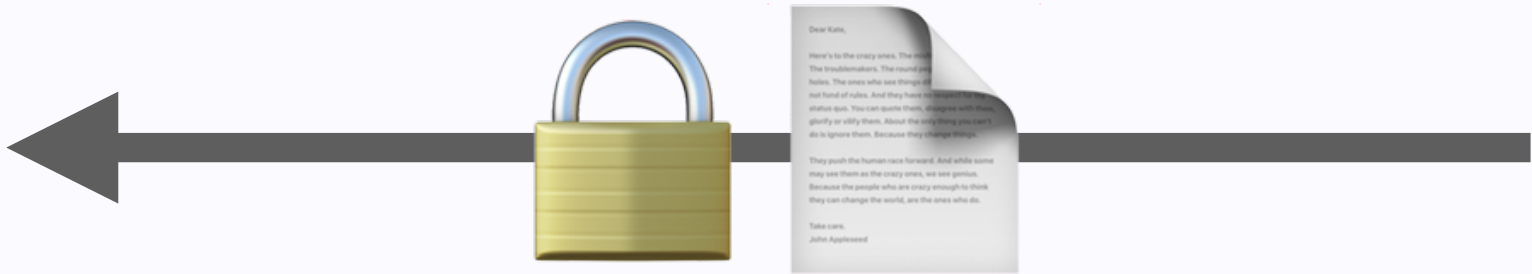
Write  
(Update)



Read  
(Decrypt)



Copy  
(Request)

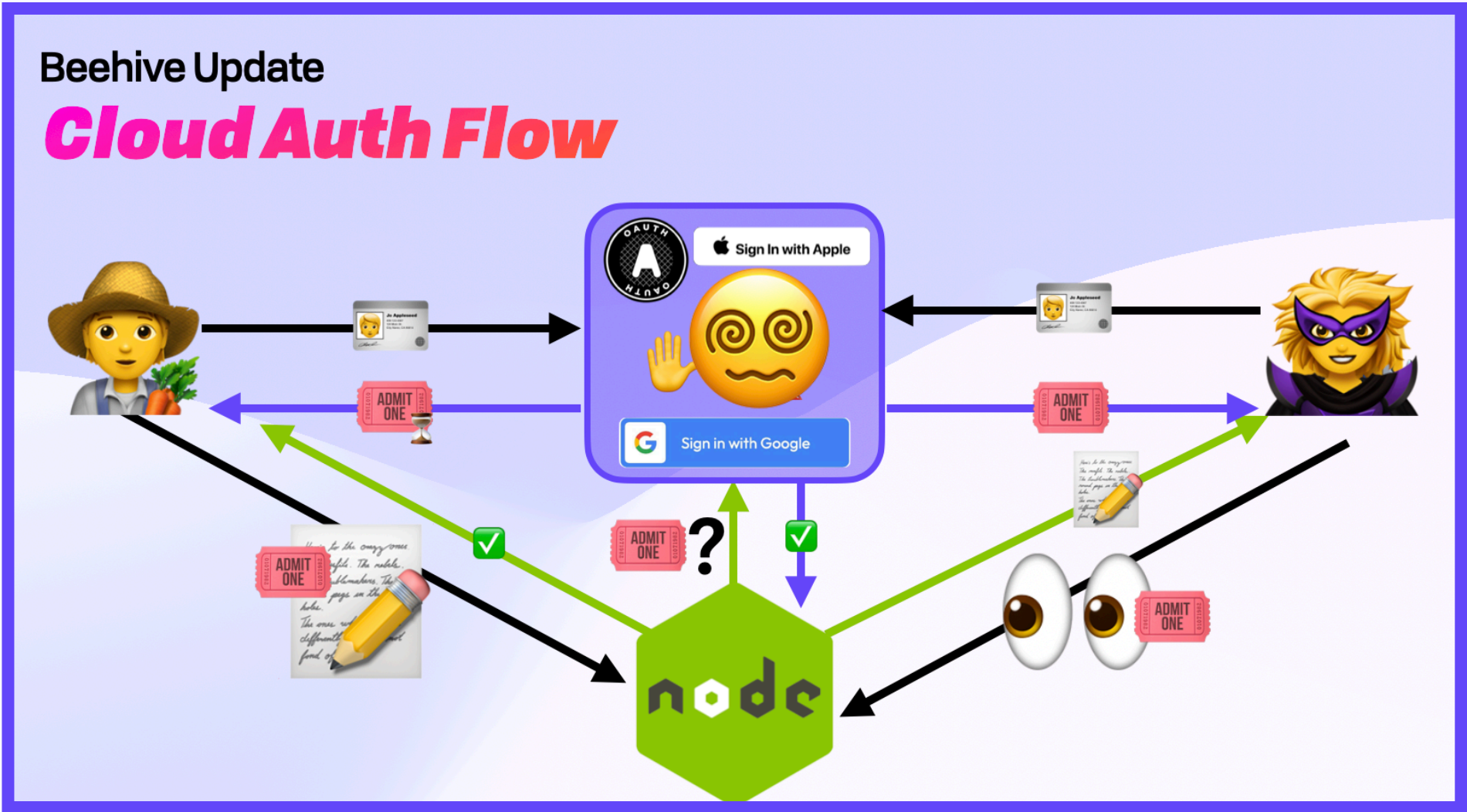


Convergent Capabilities

***Self-Authenticating Changes***

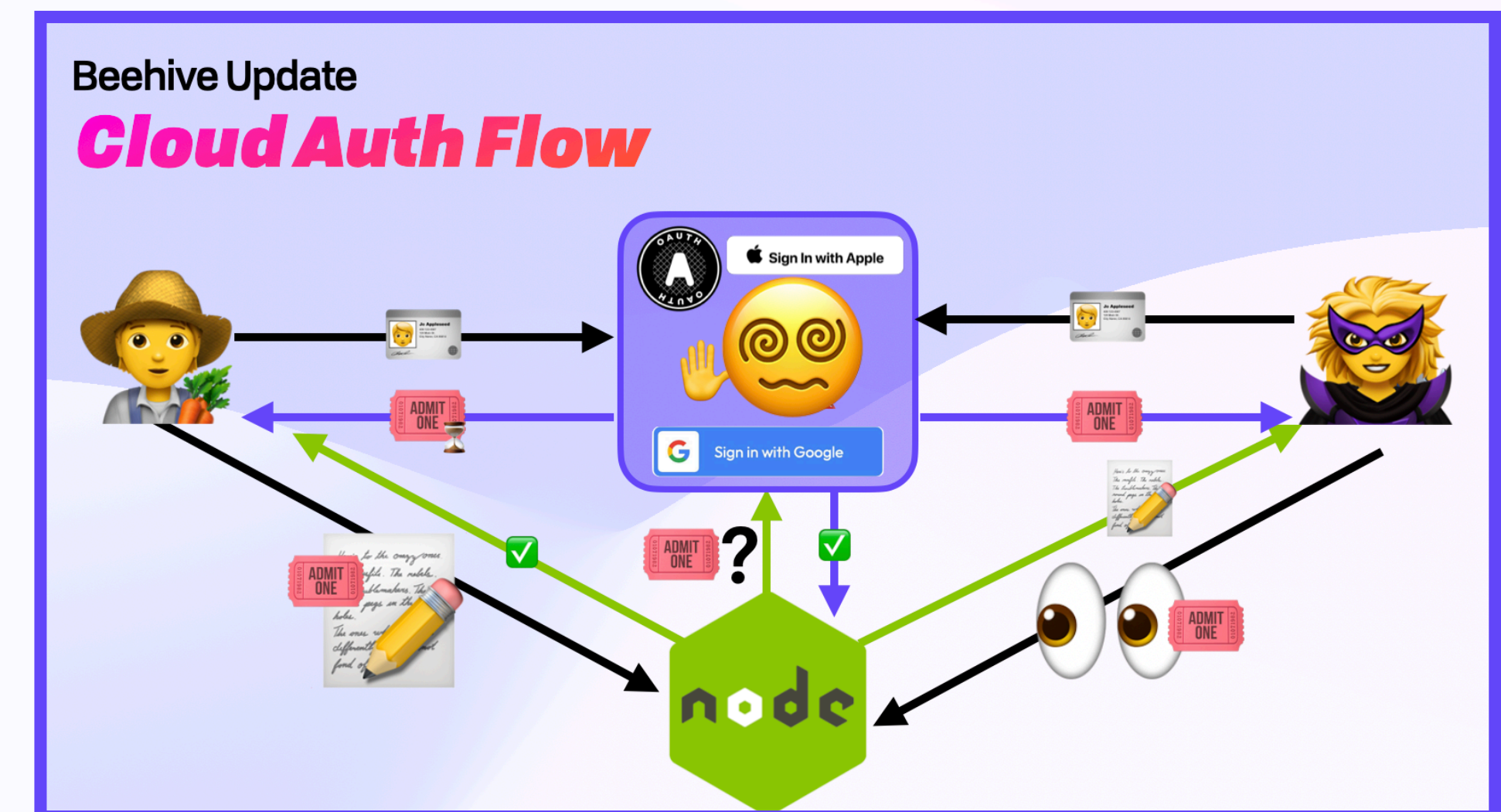
Convergent Capabilities

# Self-Authenticating Changes



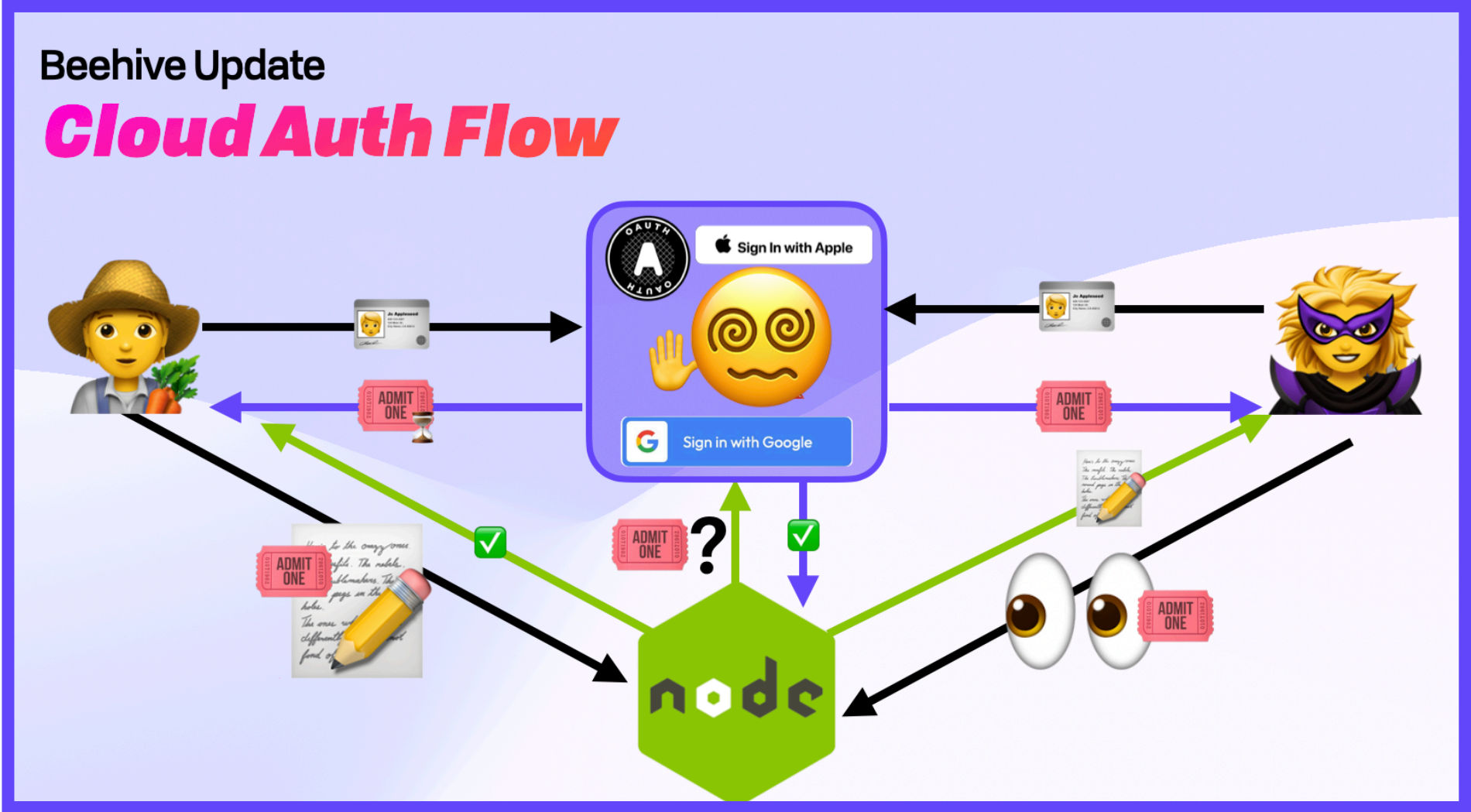


# Self-Authenticating Changes



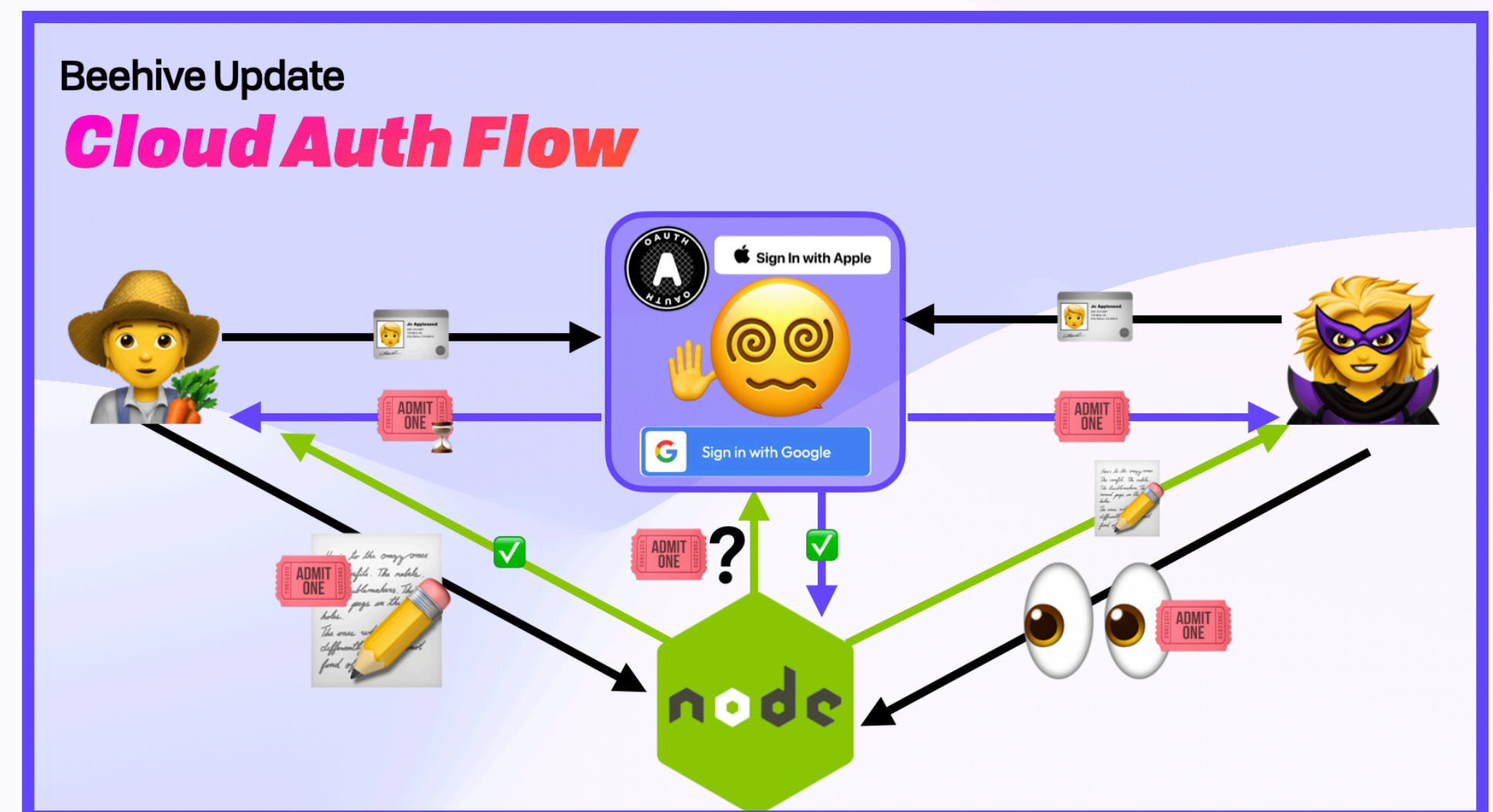
# Convergent Capabilities

# Self-Authenticating Changes





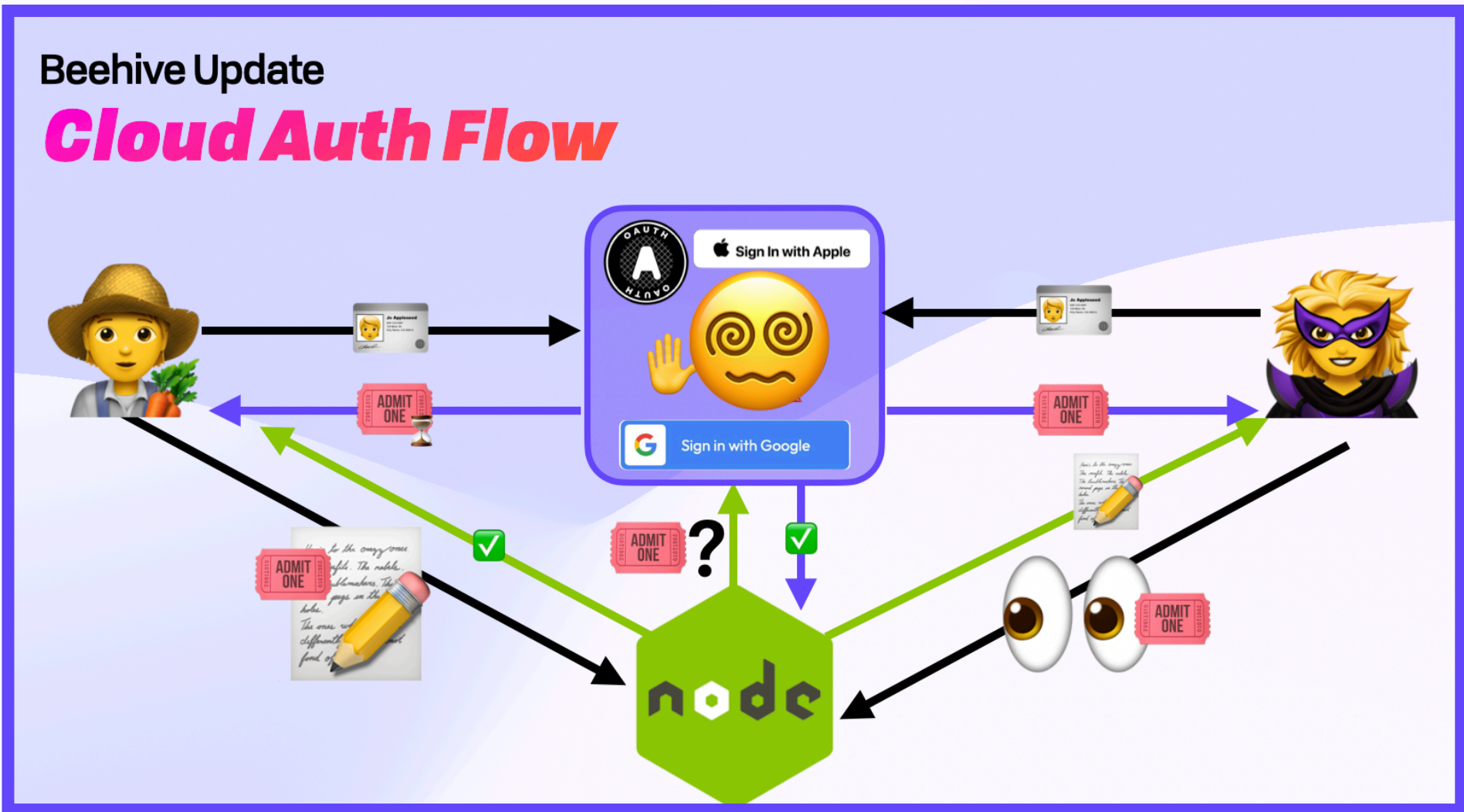
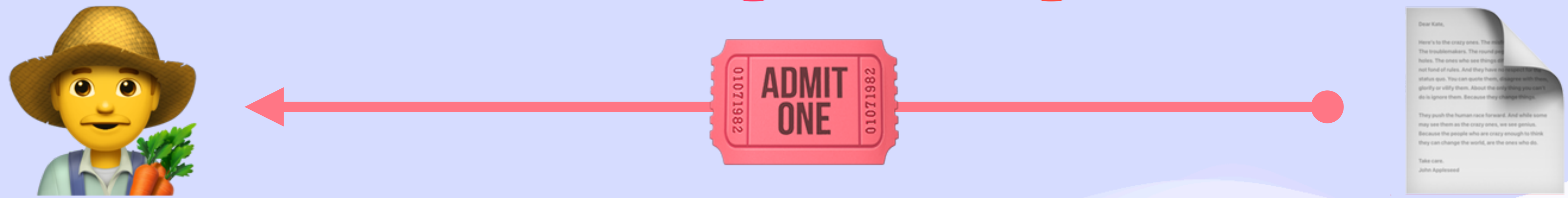
# Self-Authenticating Changes





# Convergent Capabilities

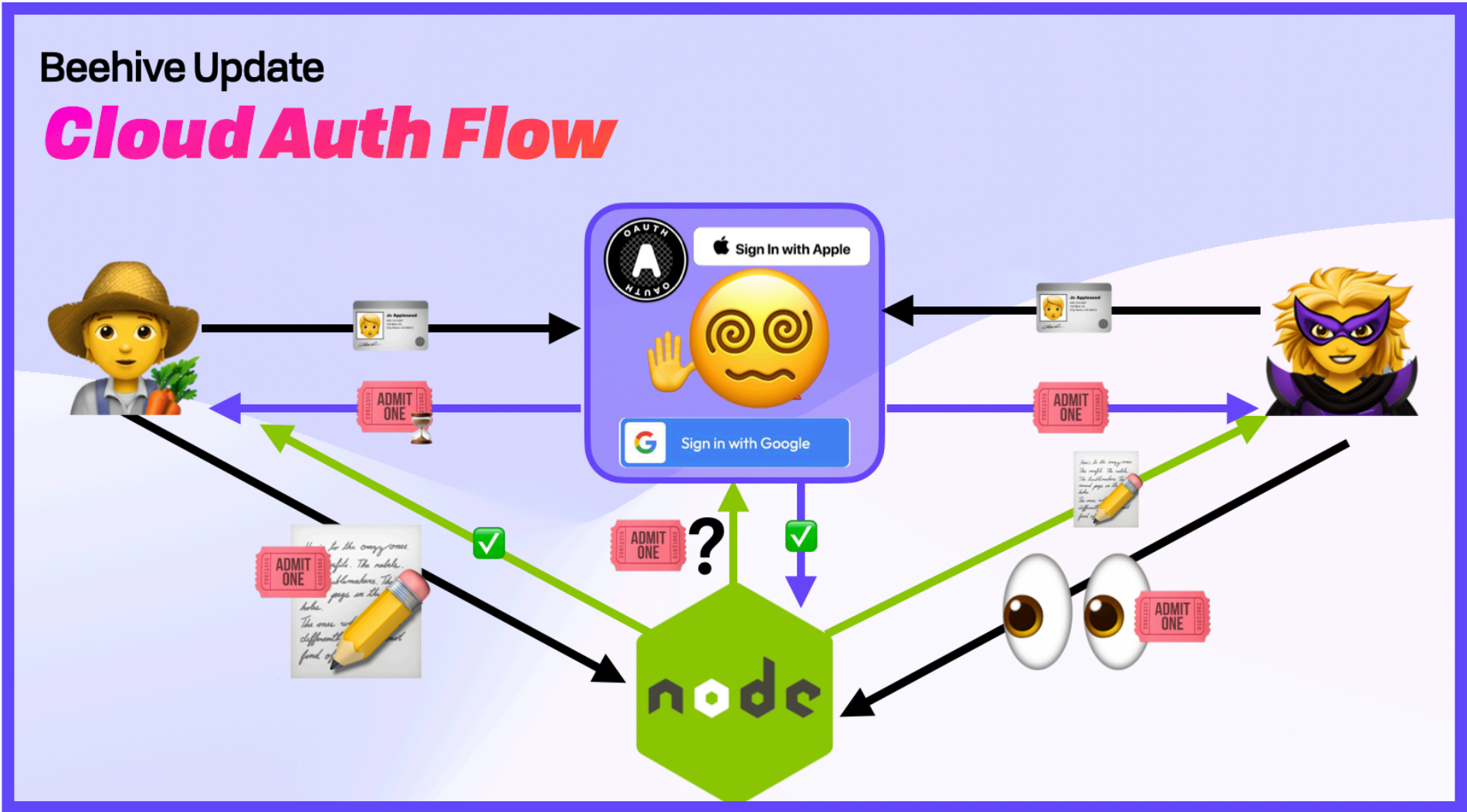
# Self-Authenticating Changes





# Convergent Capabilities

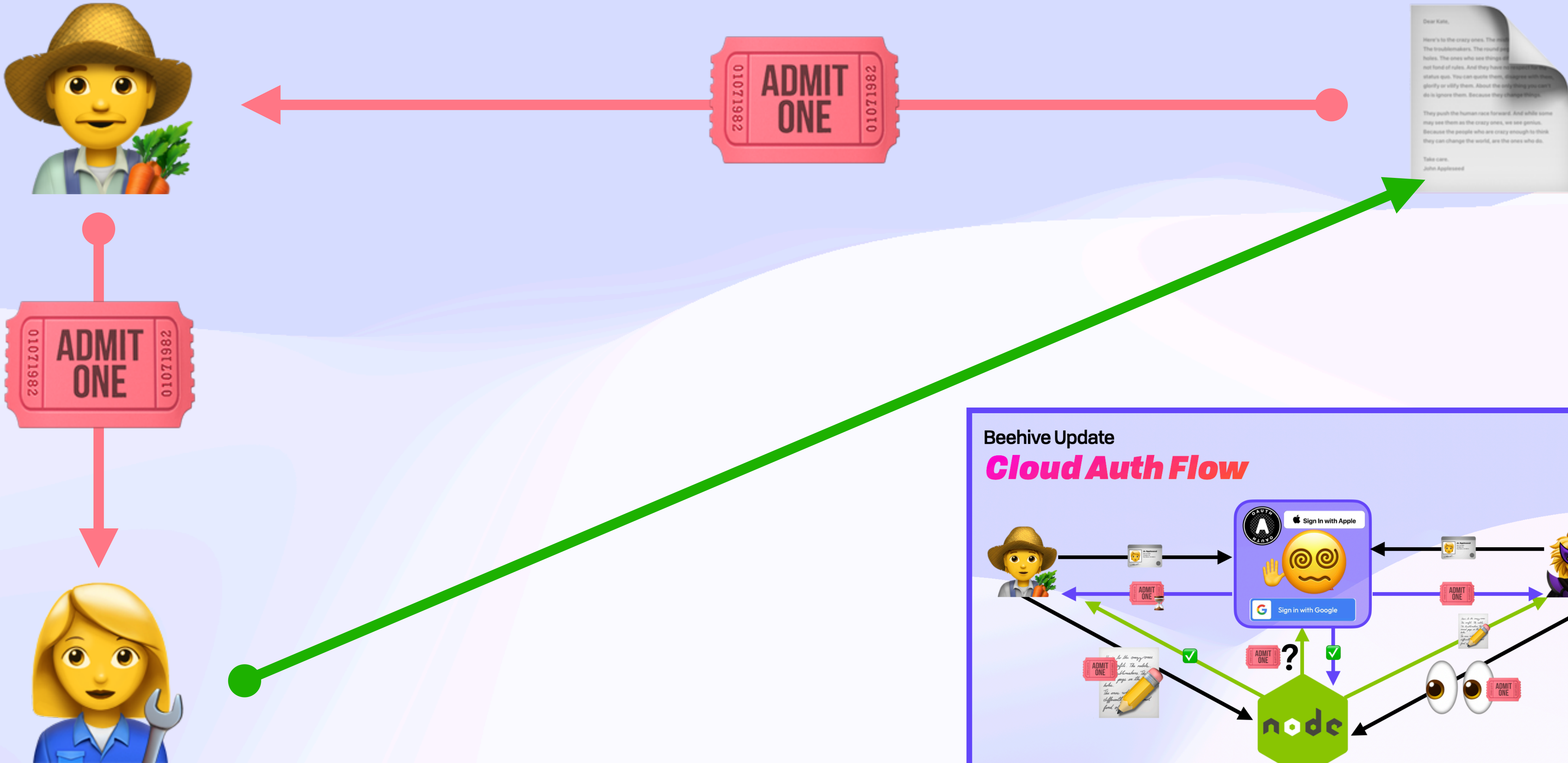
# Self-Authenticating Changes





# Convergent Capabilities

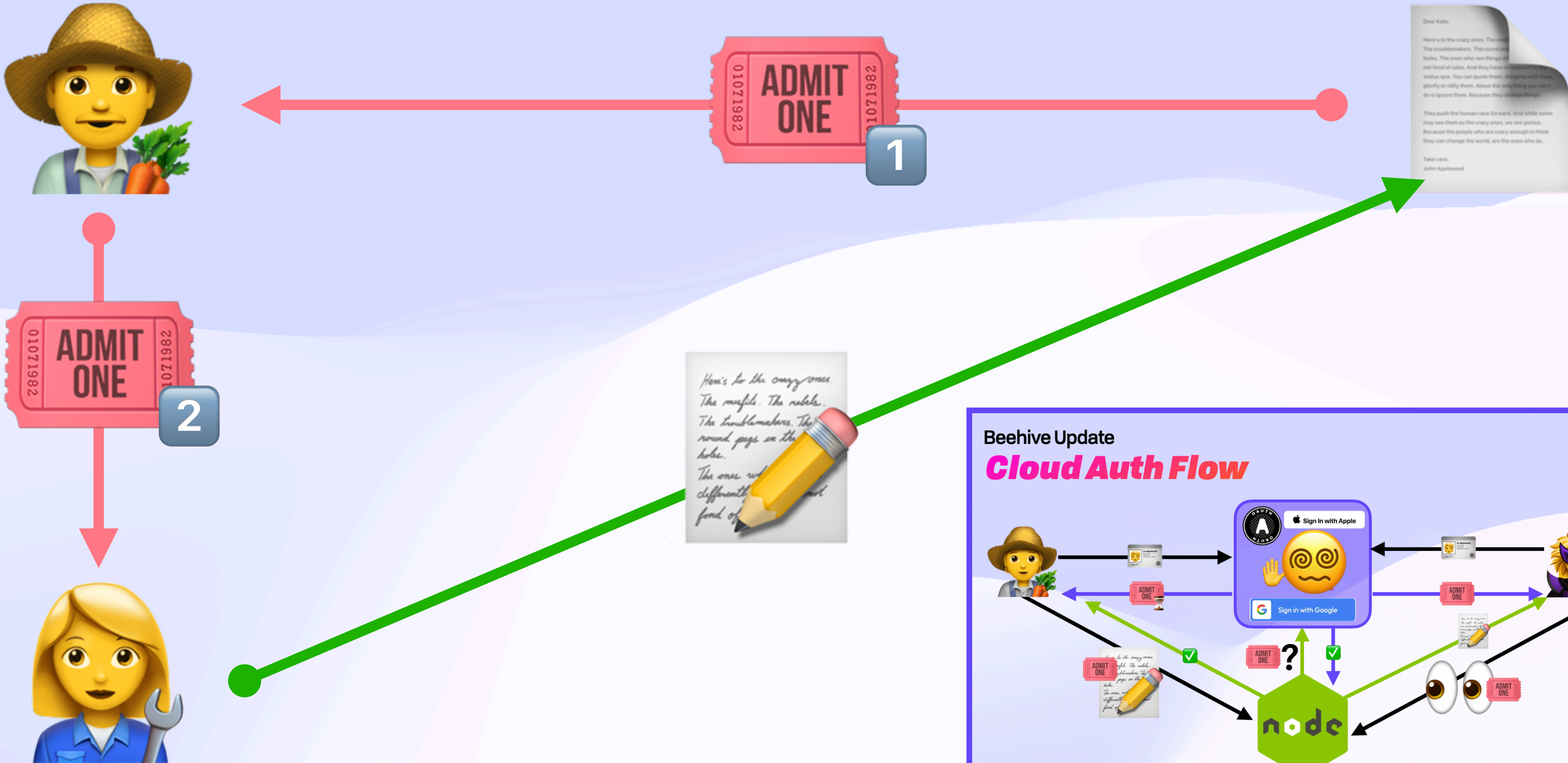
# Self-Authenticating Changes





# Convergent Capabilities

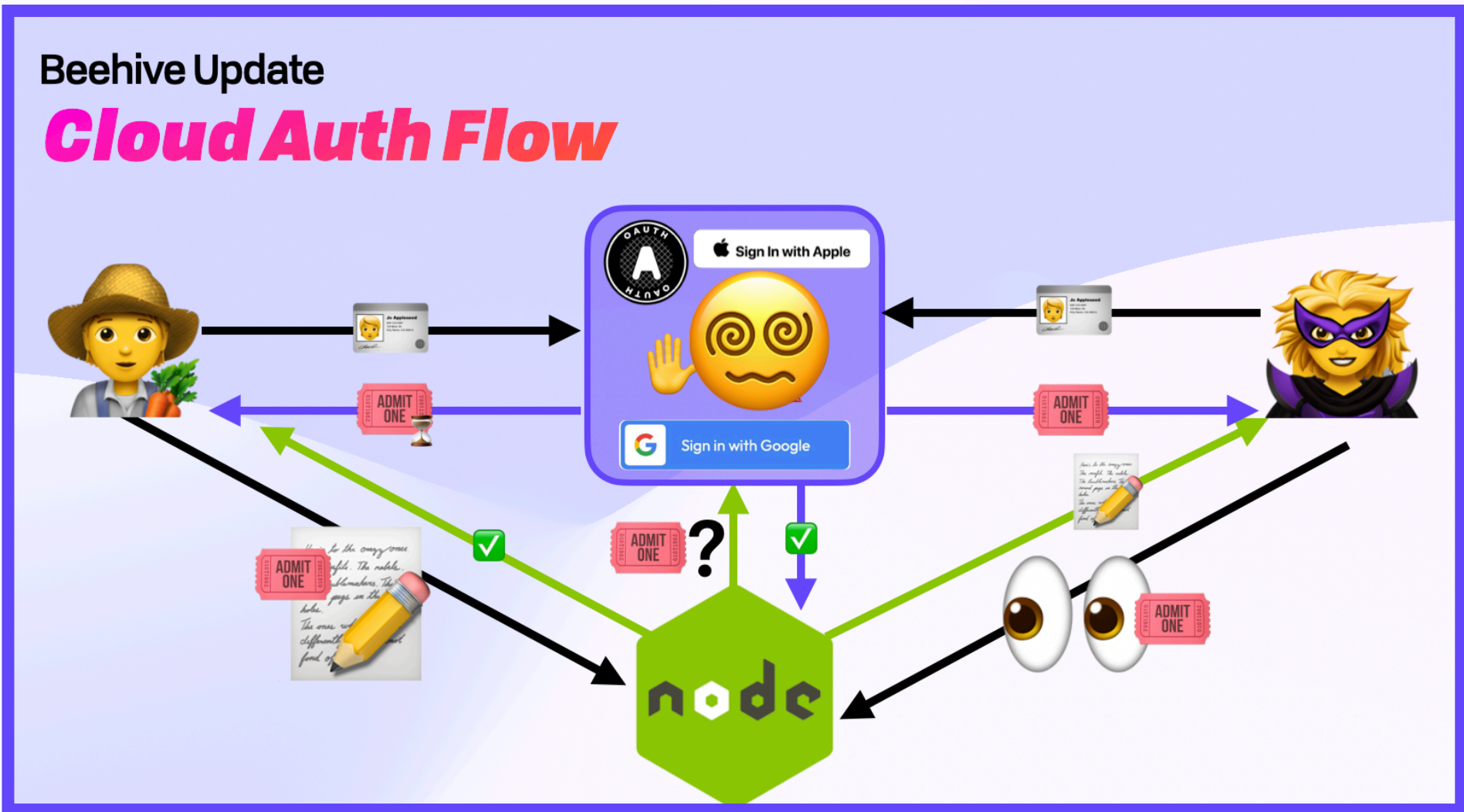
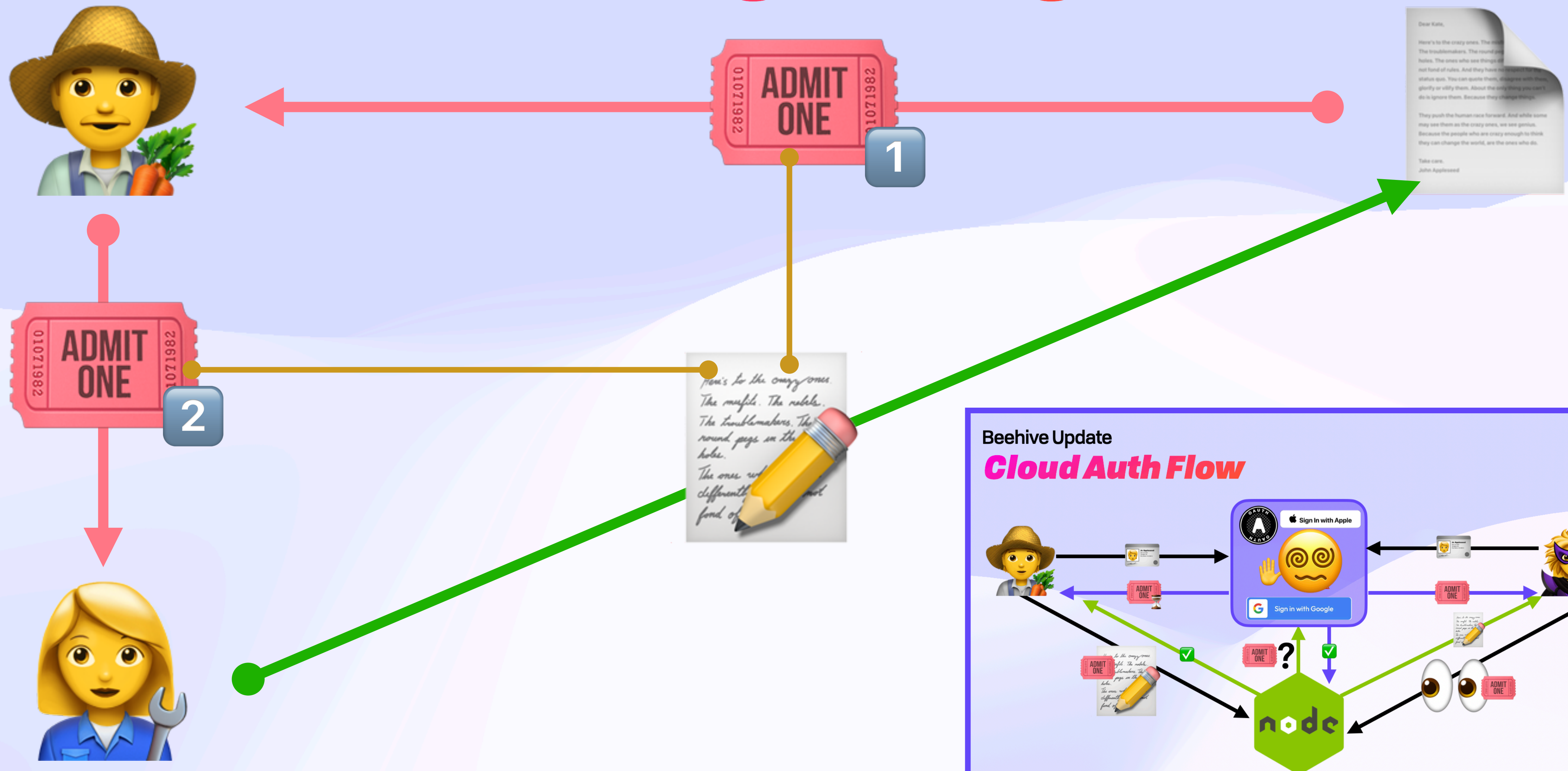
# Self-Authenticating Changes





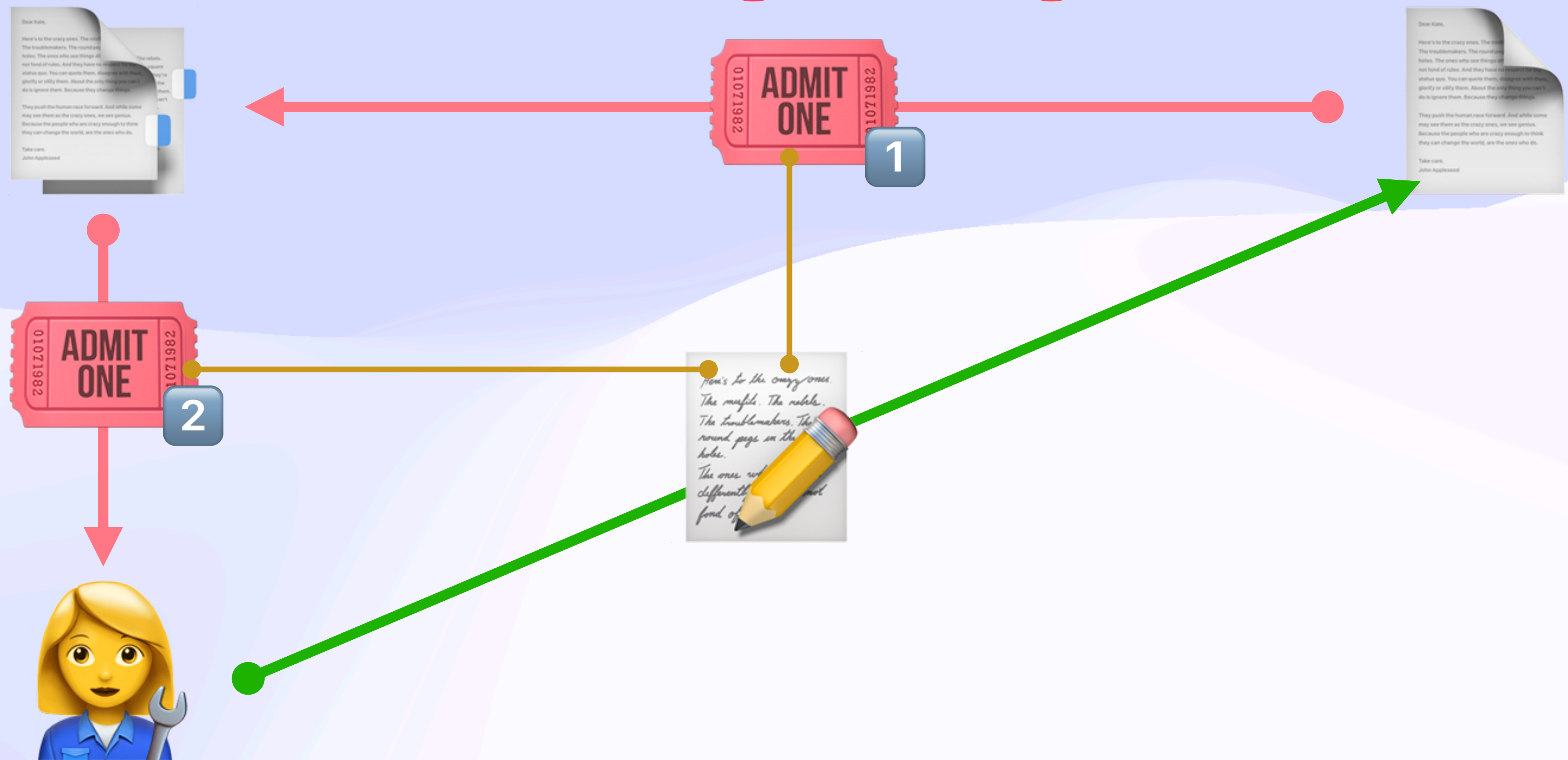
# Convergent Capabilities

# Self-Authenticating Changes



# Convergent Capabilities

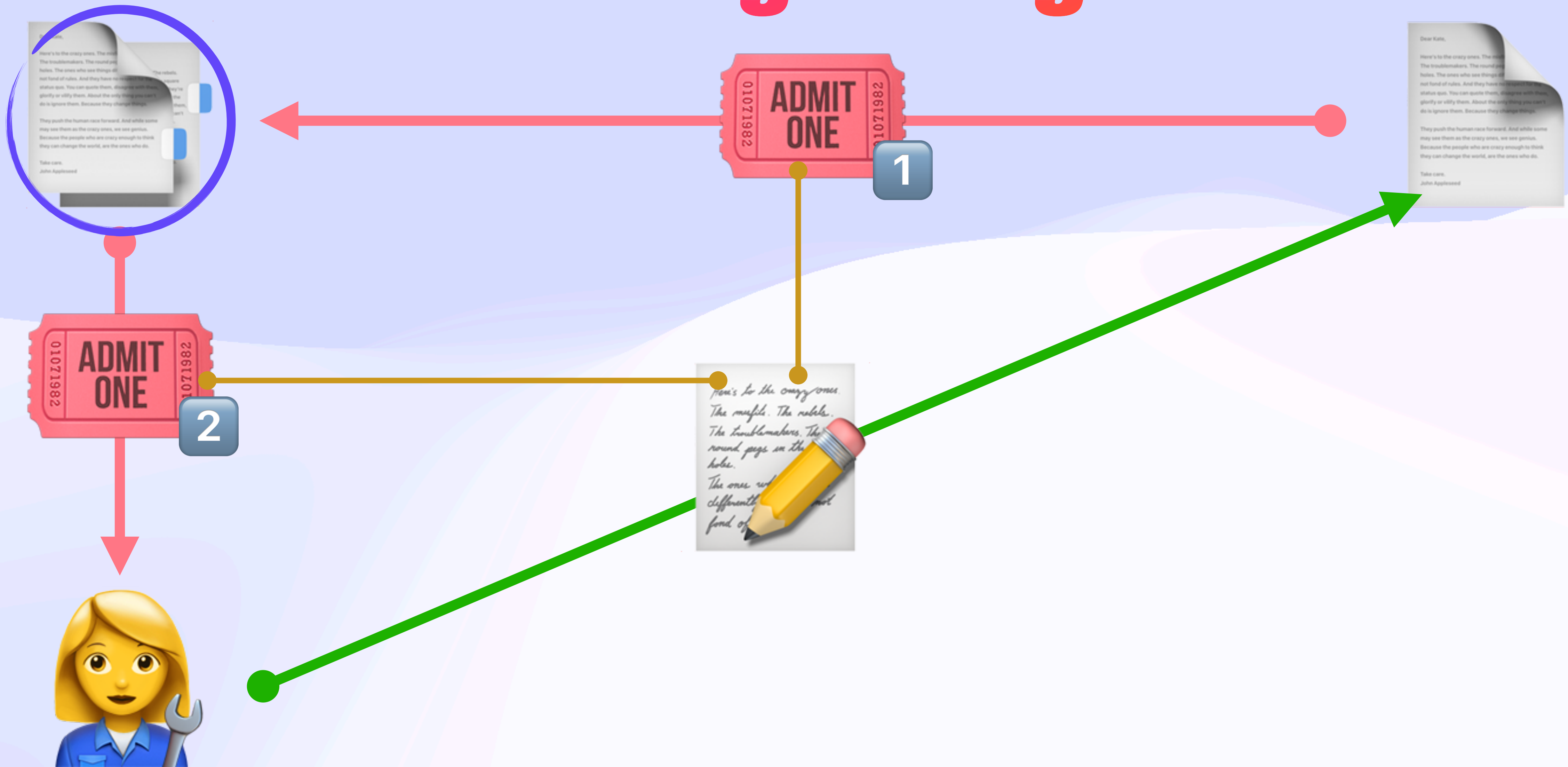
# Self-Authenticating Changes





# Convergent Capabilities

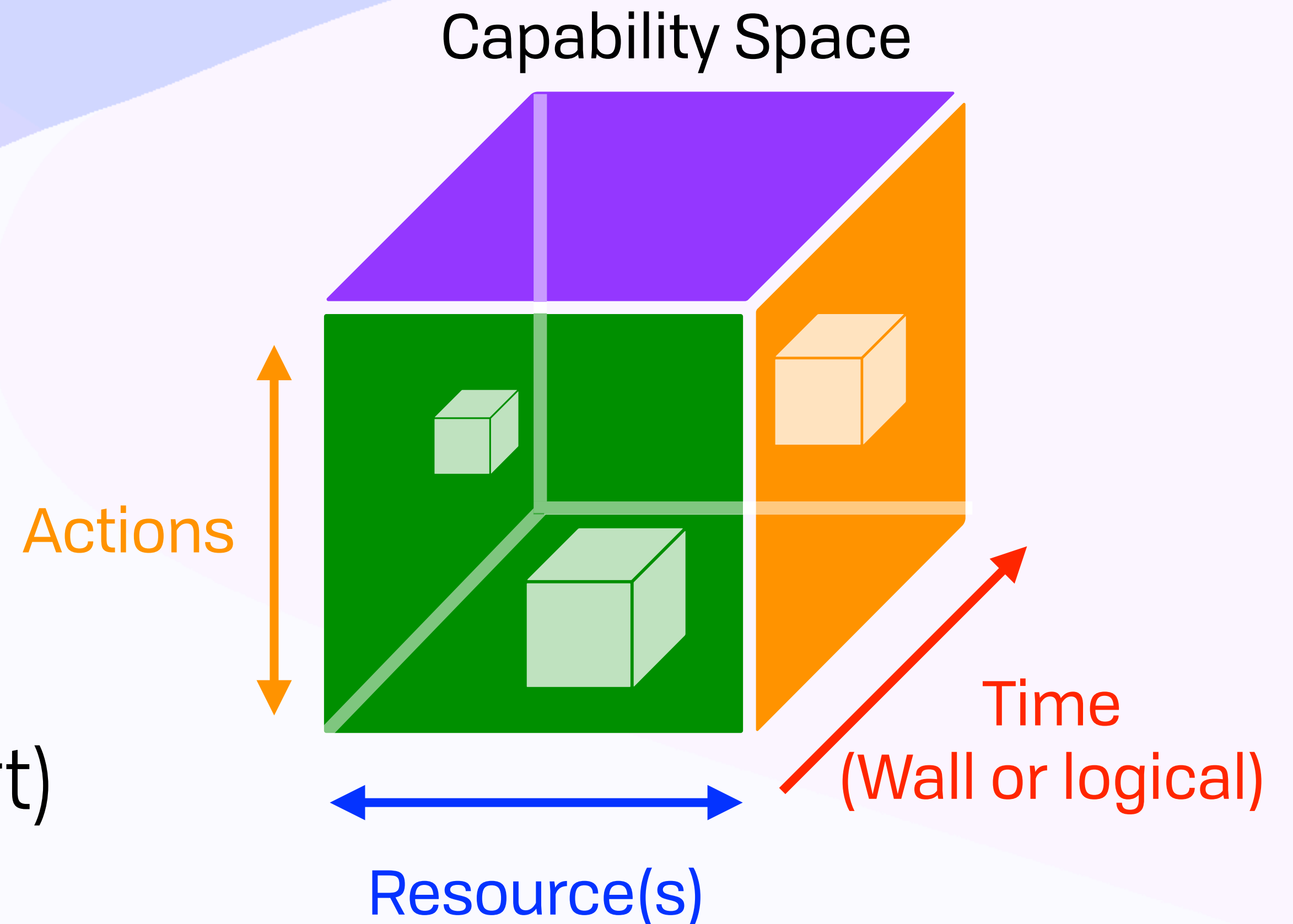
# Self-Authenticating Changes



# Convergent Capabilities

## *Capability Space* 🪐

- Make the box as small as possible!
- Fewest resources
- Fewest actions
- Least amount of time
- (Revocation considered a last resort)



Convergent Capabilities

# ***Honest Model, Simple to Use***

- "If you have it, you can use it & share it"
- Closely models auth **ground truth**
- Natural **programming** model "like passing handles"



Convergent Capabilities

***Example API***

# Convergent Capabilities

## ***Example API***

- `myDocument.addMember(alice, role)`

# Convergent Capabilities

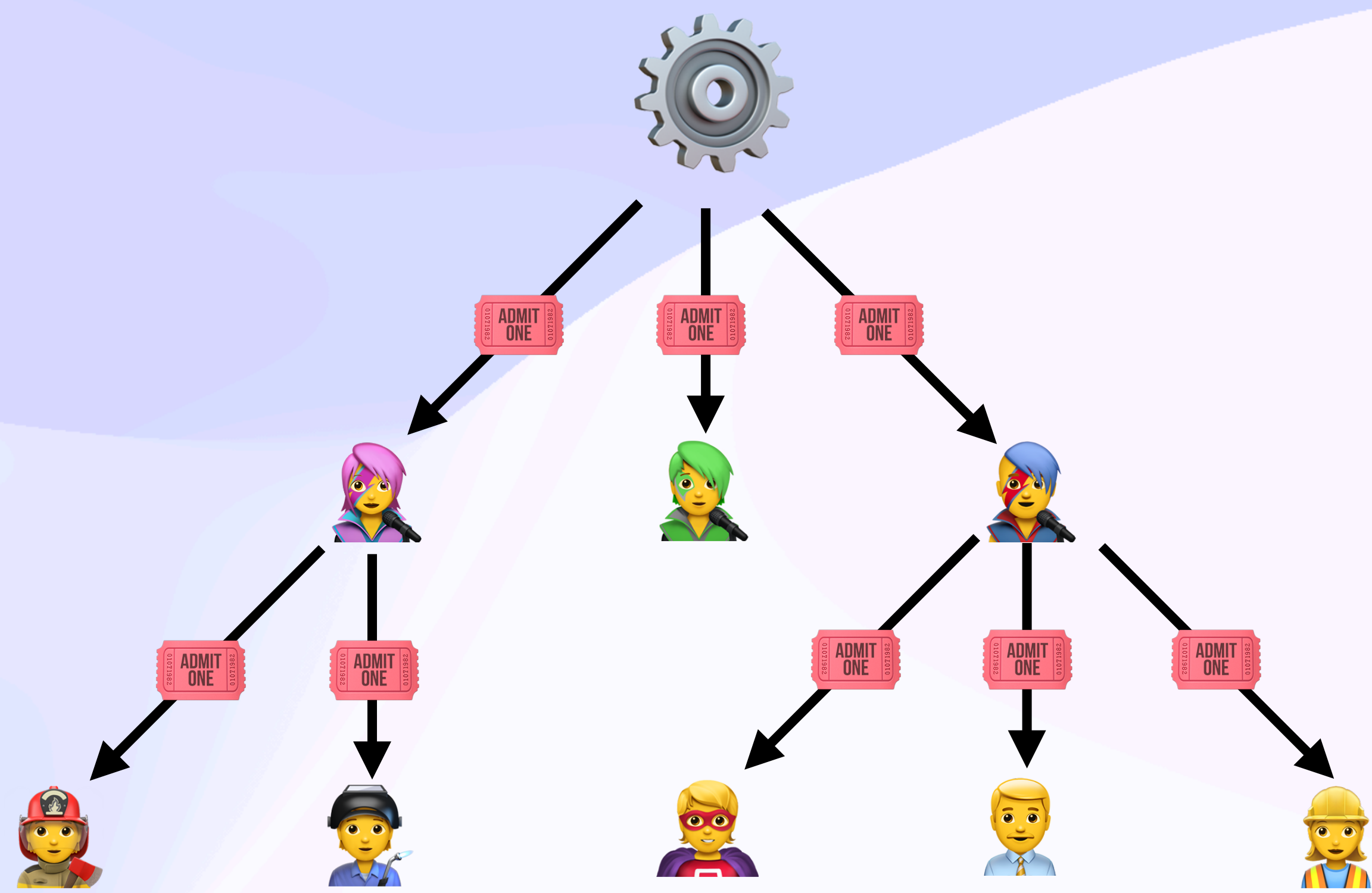
## ***Example API***

- `myDocument.addMember(alice, role)`
- `myDocument.addText("hello world")`



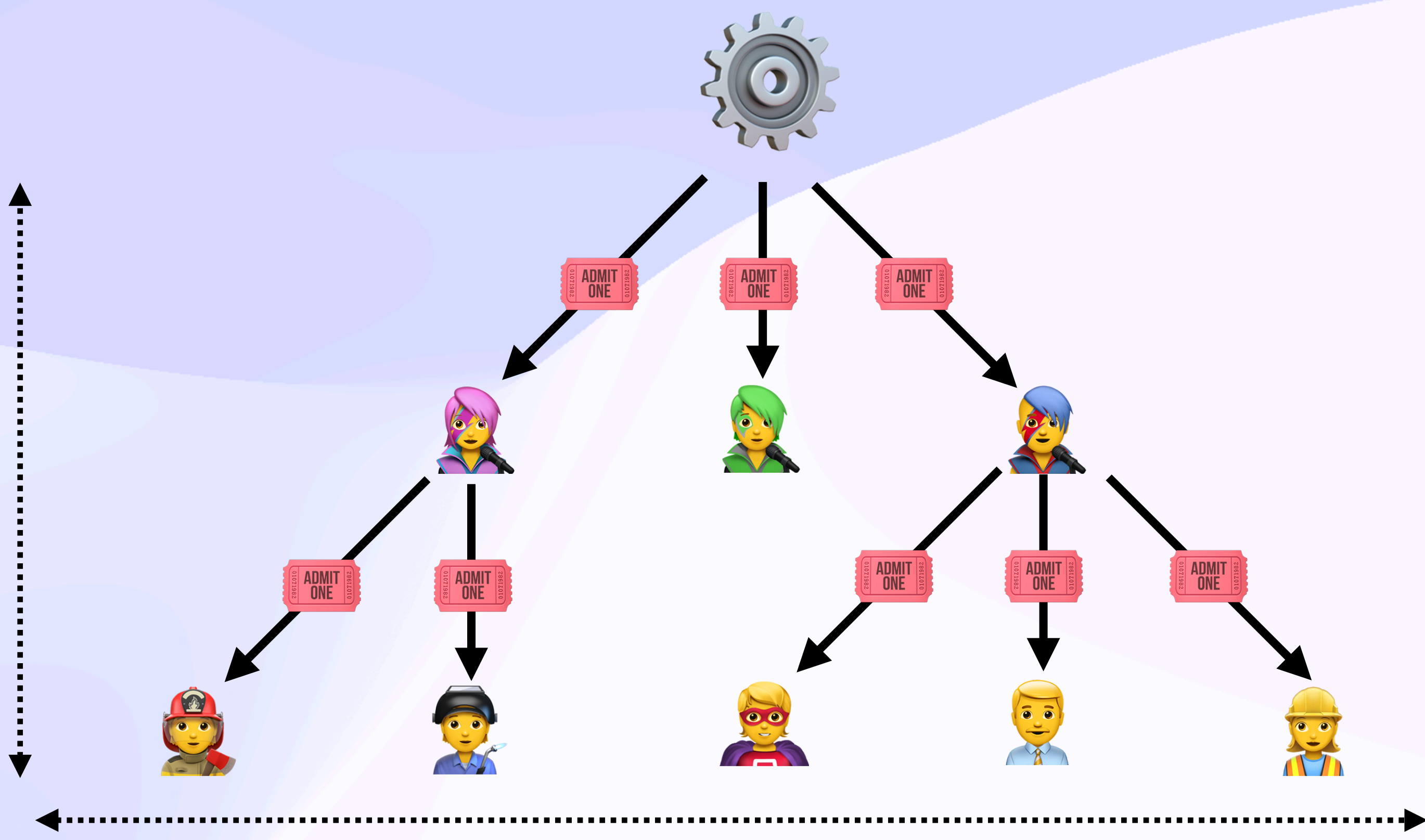
# Convergent Capabilities

# *On a Need to Know Basis*



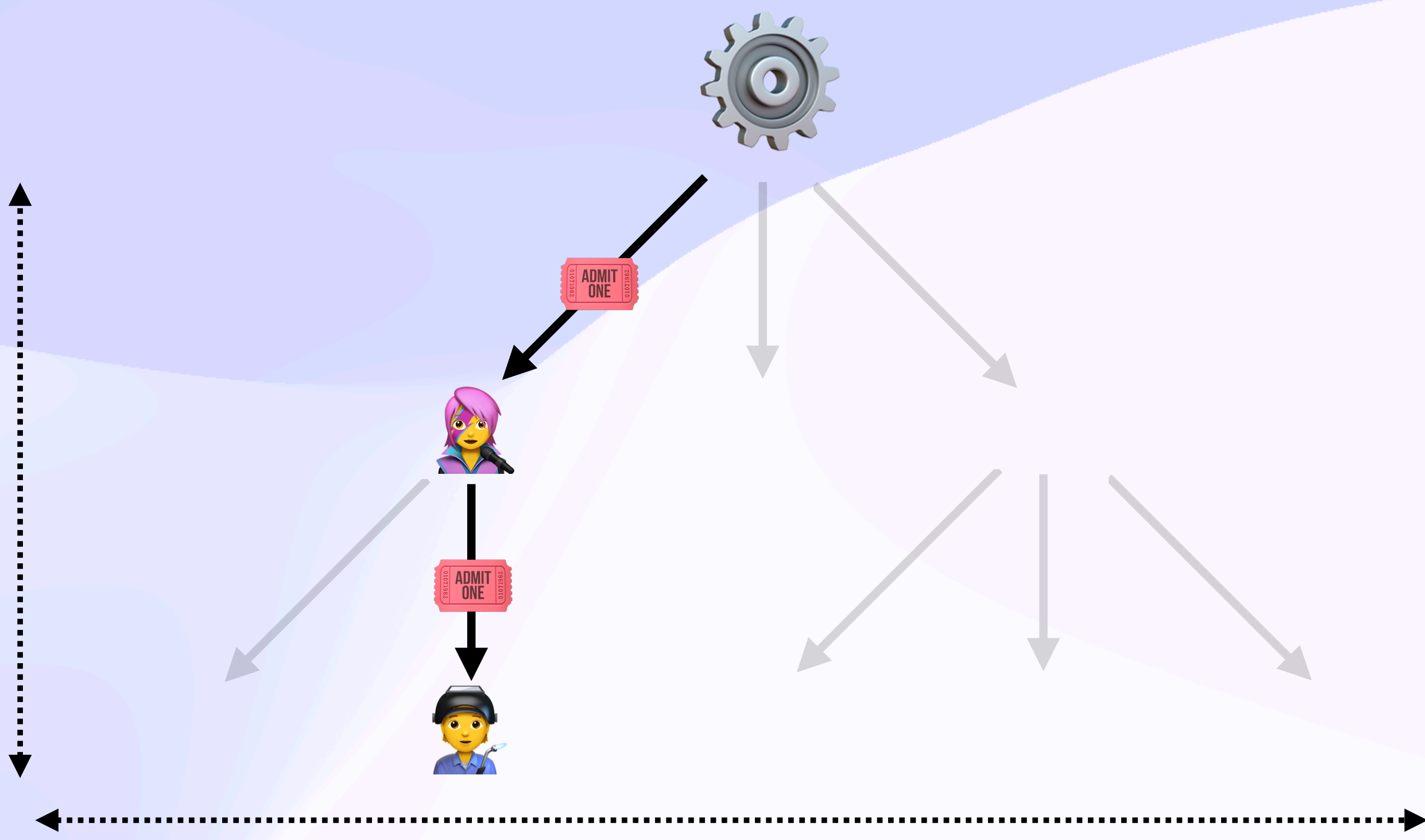
# Convergent Capabilities

# *On a Need to Know Basis*



# Convergent Capabilities

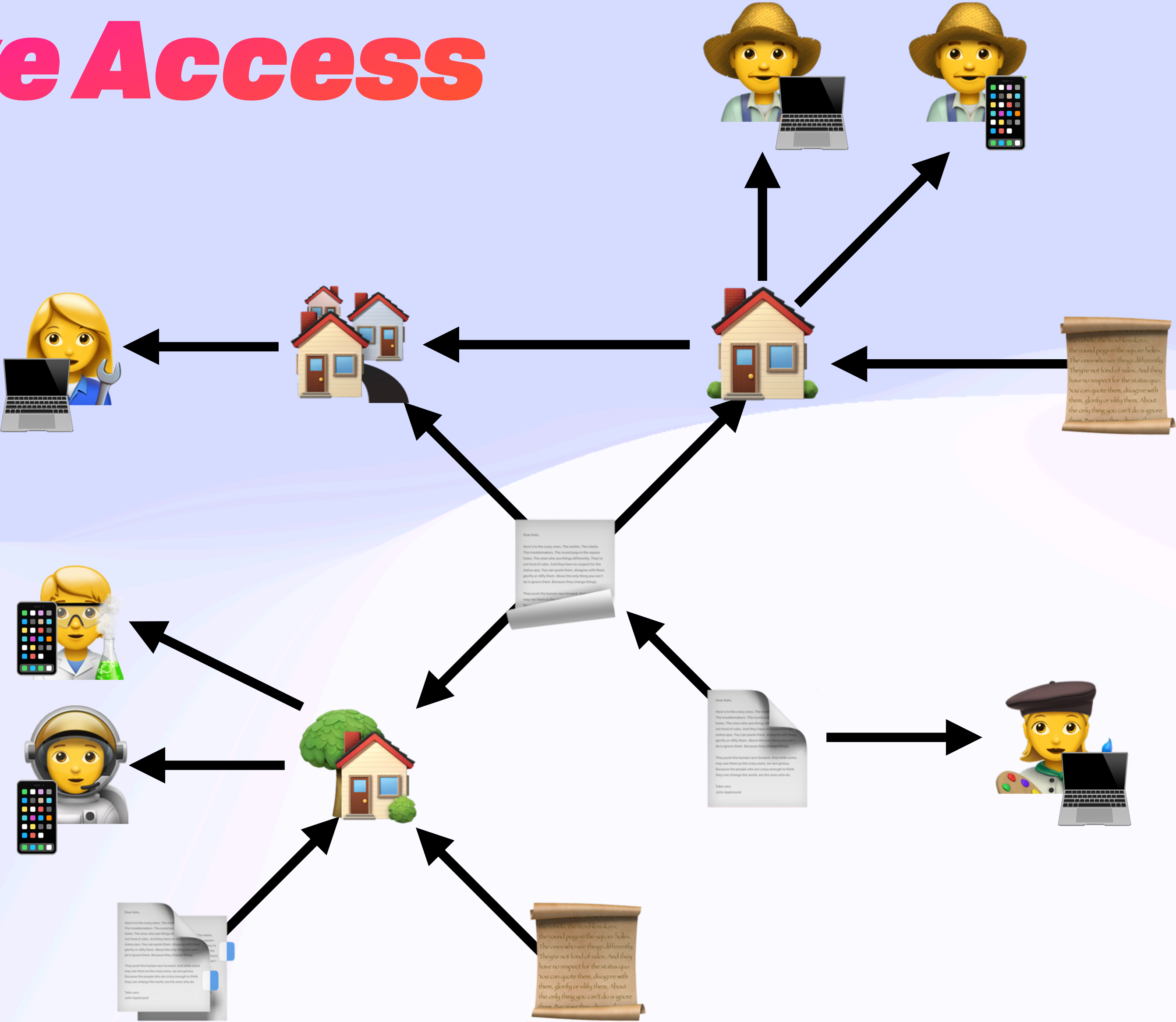
# *On a Need to Know Basis*





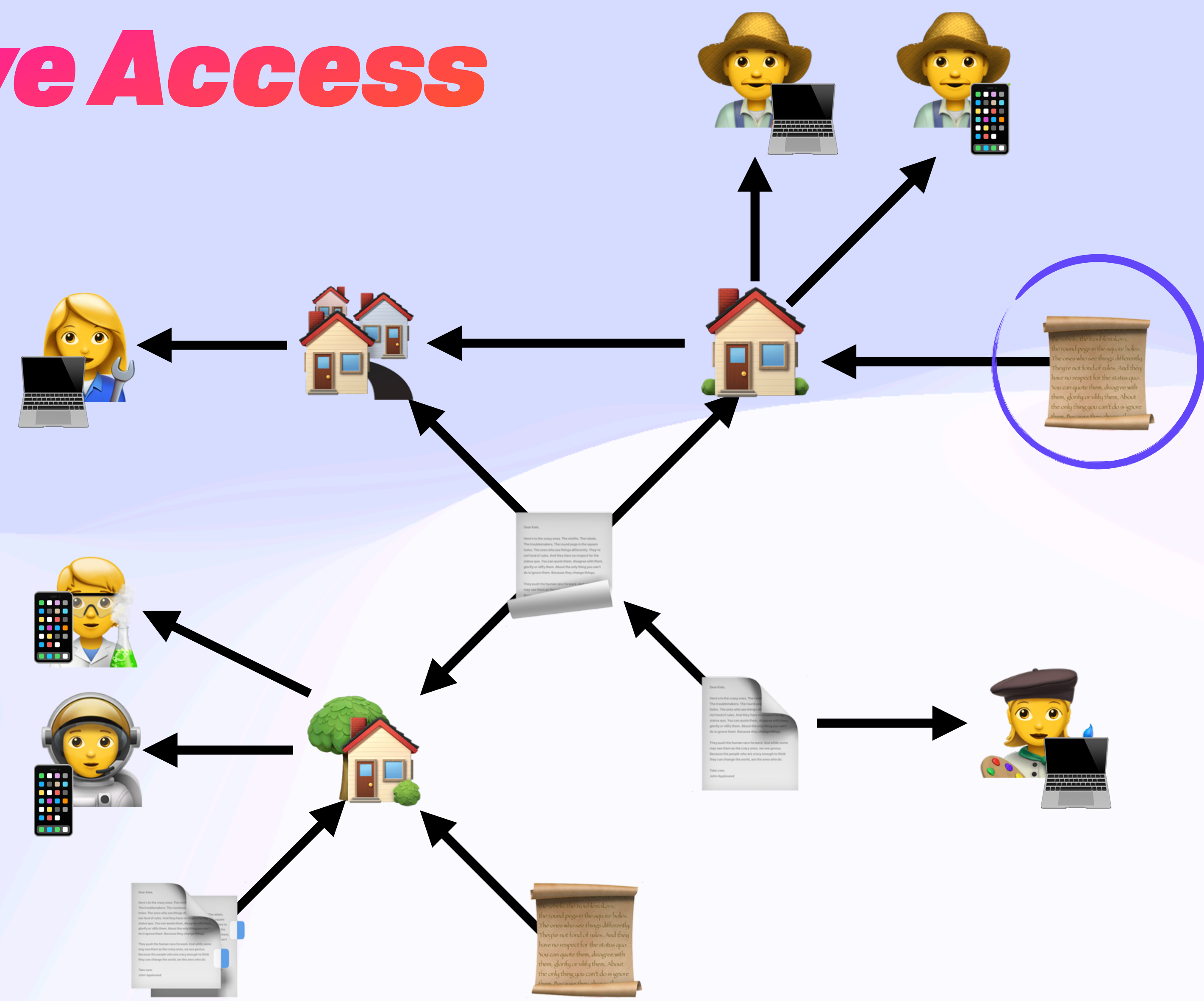
# Convergent Capabilities

# *Transitive Access*



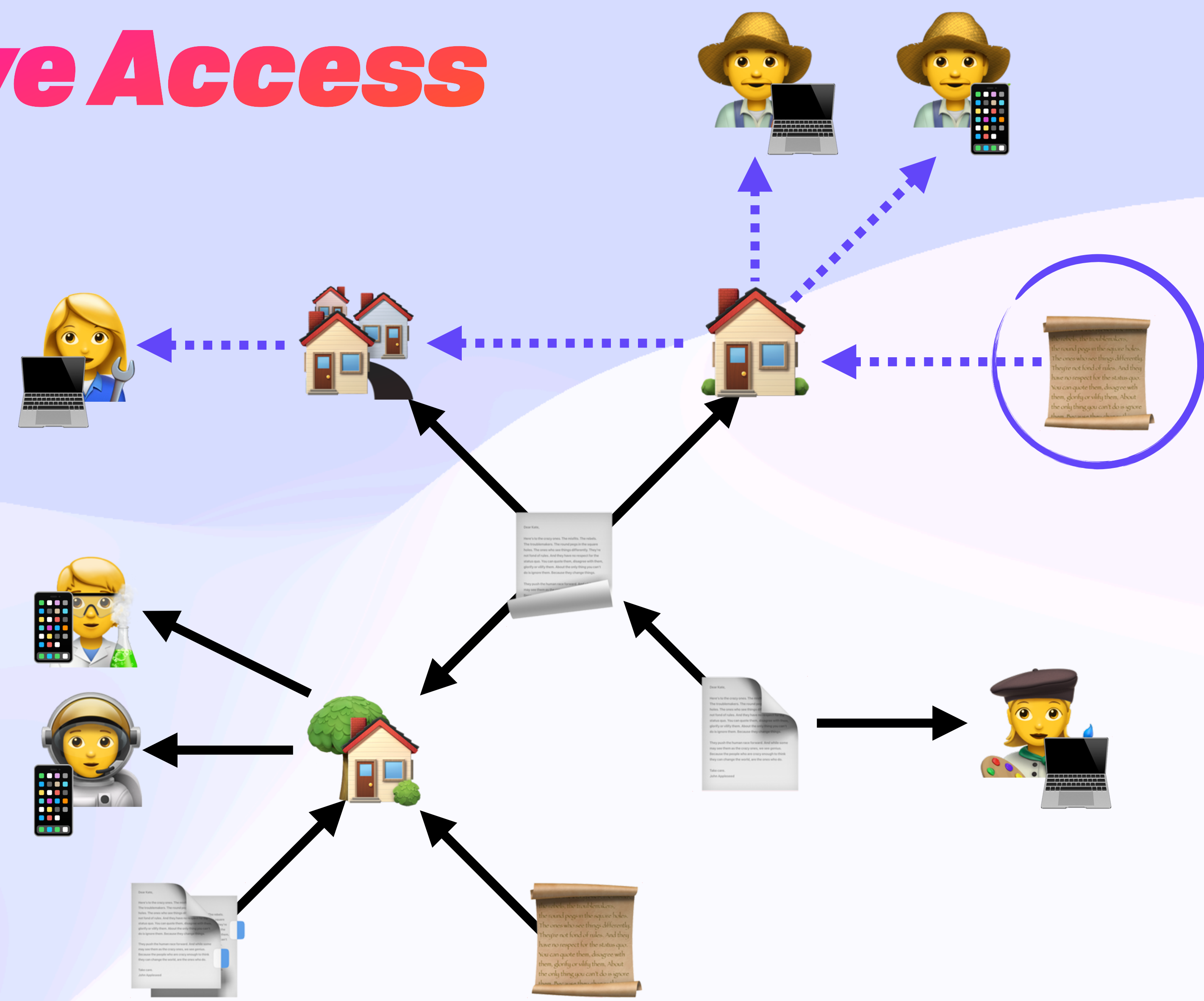
# Convergent Capabilities

# Transitive Access



# Convergent Capabilities

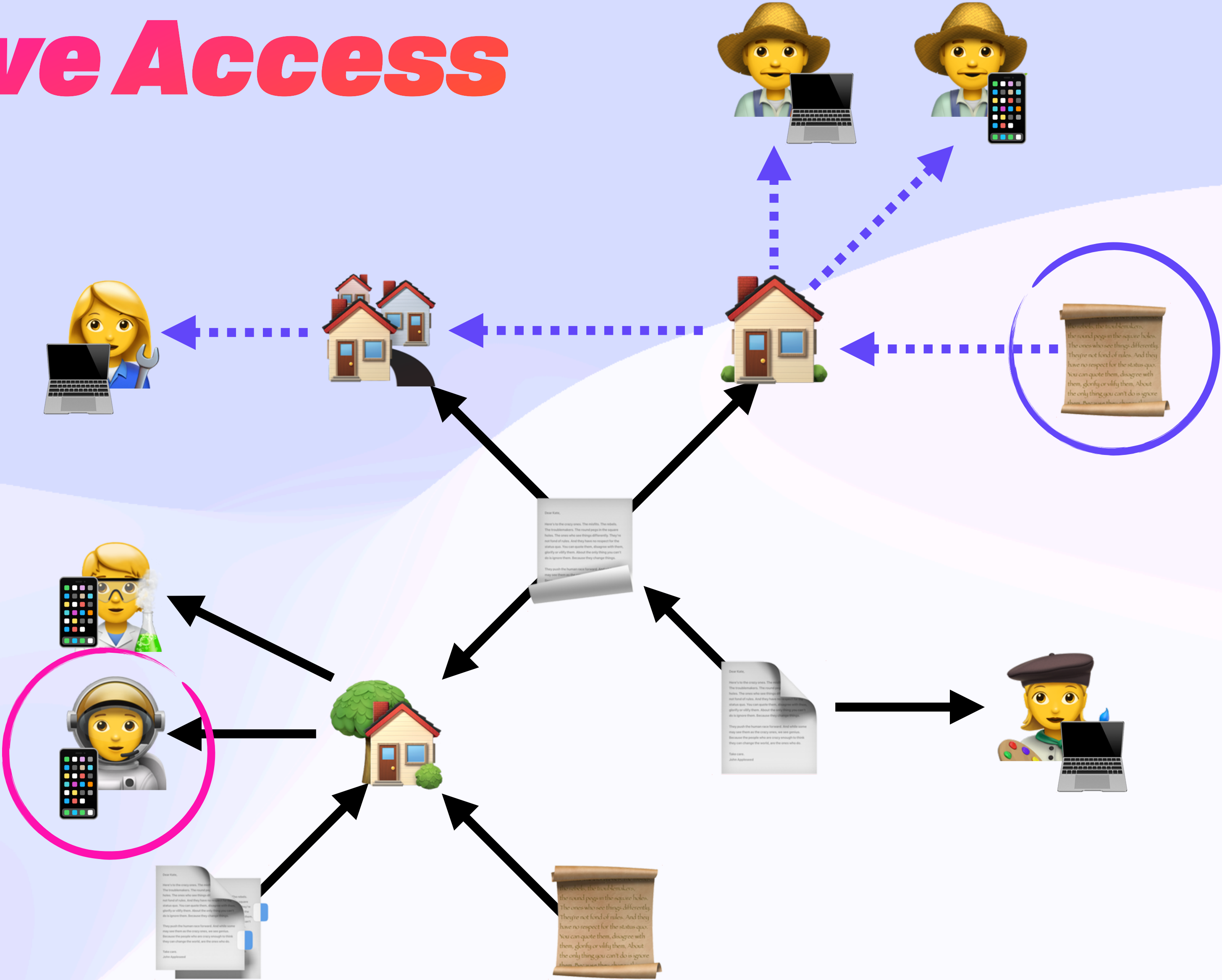
# Transitive Access





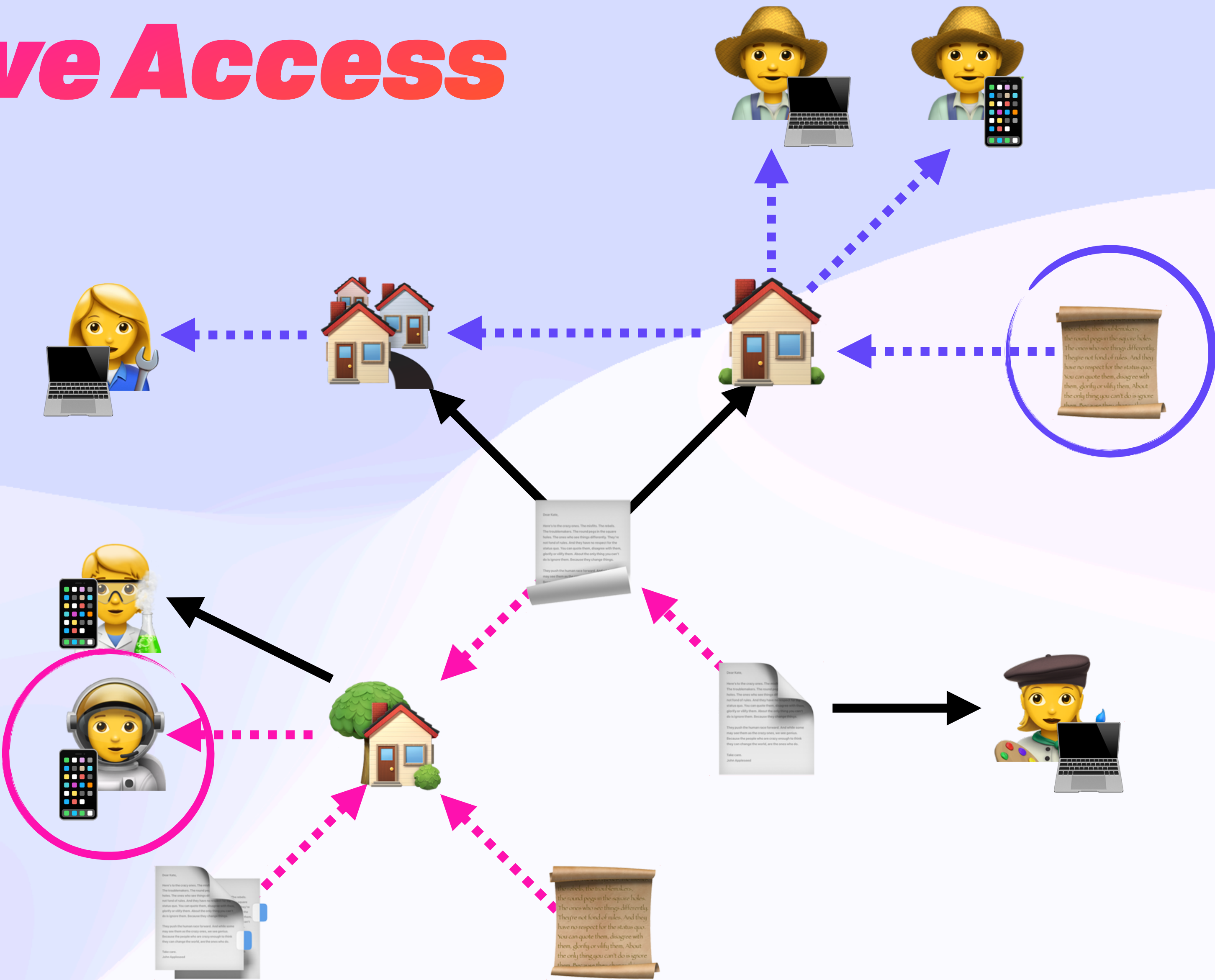
# Convergent Capabilities

# Transitive Access



# Convergent Capabilities

# *Transitive Access*



Convergent Capabilities

***Auth Accidents Will Happen***



Convergent Capabilities

***Auth Accidents Will Happen***

Bad Actor (Byzantine)





Convergent Capabilities

# ***Auth Accidents Will Happen***

Bad Actor (Byzantine)



Revocation (non-monotone)





# Convergent Capabilities

## ***Revocation***



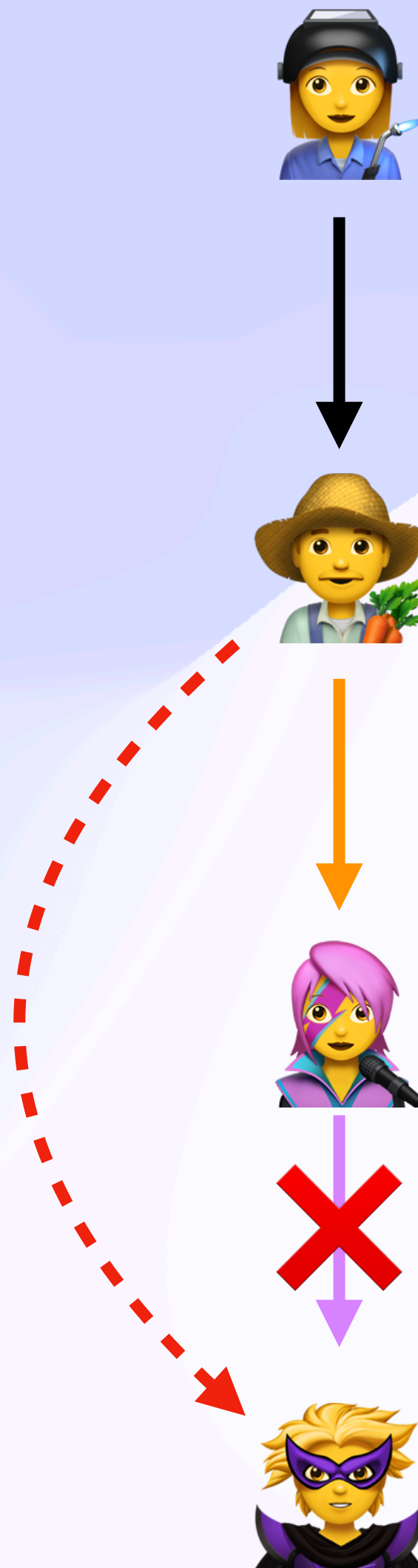
# Convergent Capabilities

## *Revocation*



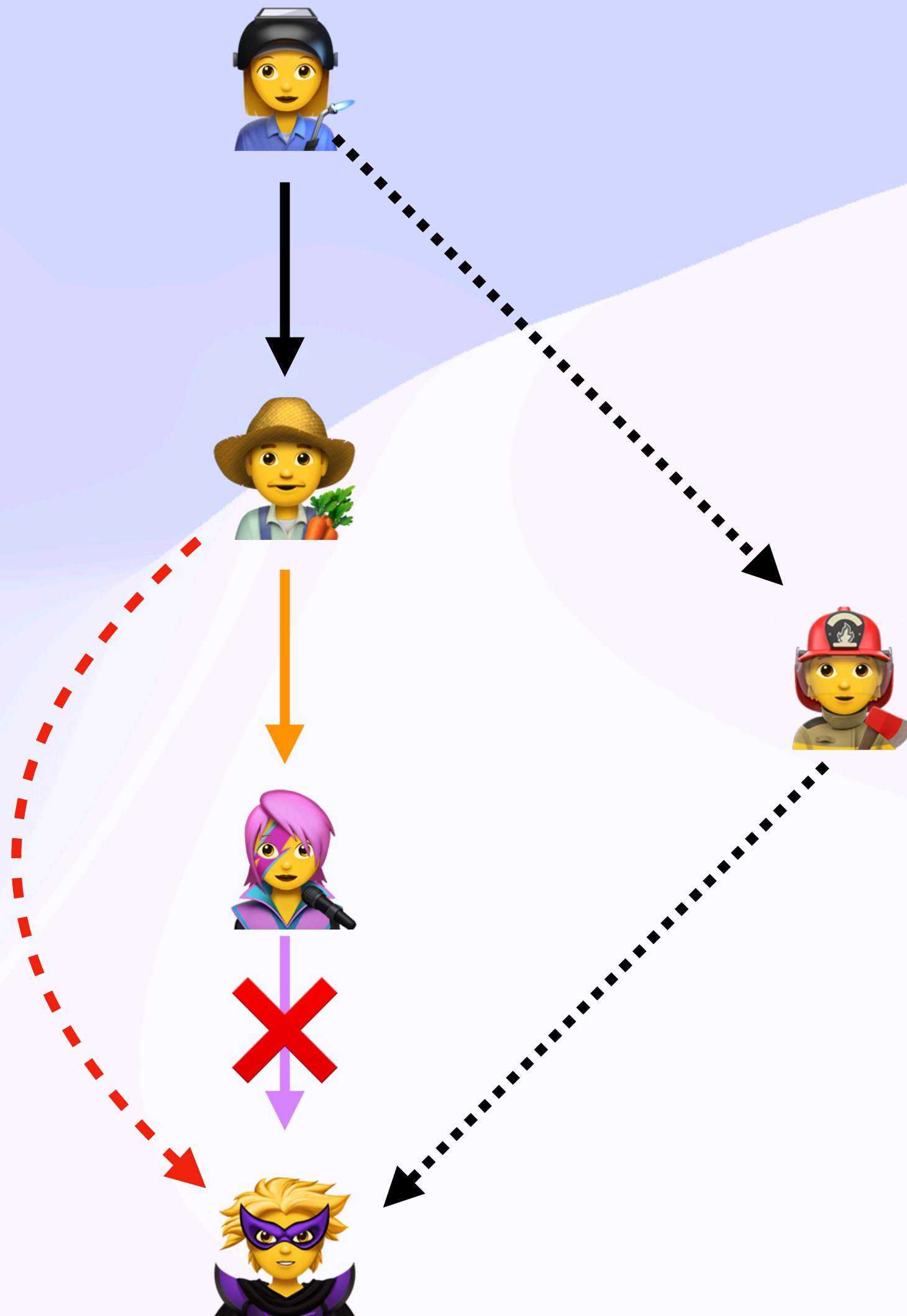
# Convergent Capabilities

# *Revocation*



# Convergent Capabilities

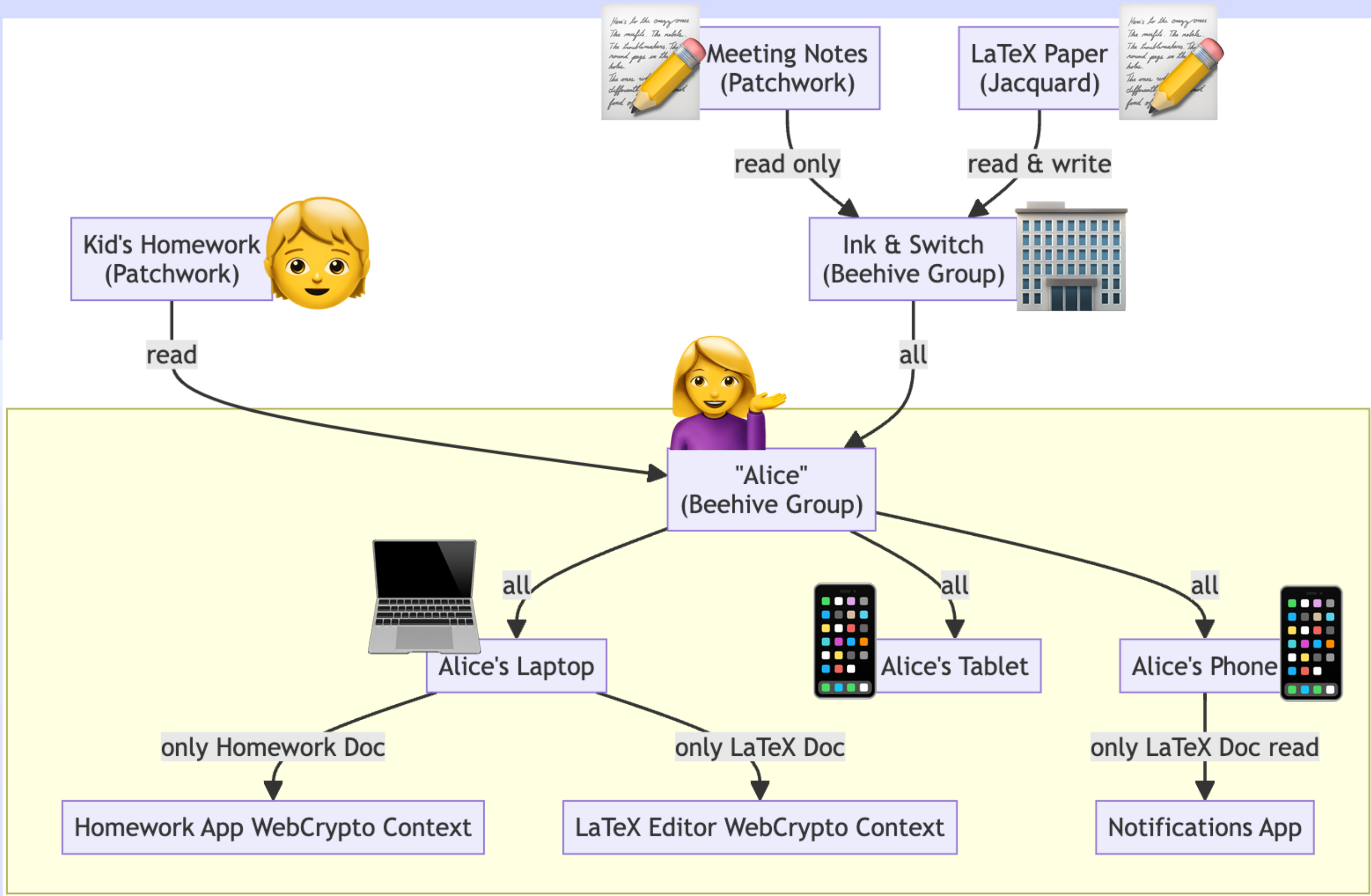
# *Revocation*





# Convergent Capabilities

# User Key Management, Rotation, Etc



Keeping Bytes Off the Wire

***Defence in Depth***

Keeping Bytes Off the Wire

# *Defence in Depth*





Keeping Bytes Off the Wire

# *Defence in Depth*



Keeping Bytes Off the Wire

# *Defence in Depth*



A Different Context

# ***Trust-Minimised Server Infrastructure***



## A Different Context

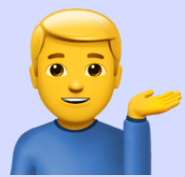
# ***Trust-Minimised Server Infrastructure***

- Servers are still *very convenient* in 2025
- What do we *trust* servers to do?
  - Hold our bytes and not delete them
  - Only send those bytes to someone with the right permissions
  - Trust them with the knowledge of our auth graph
    - Server knows who (IP address & public key) requests which doc IDs
- Defence-in-depth strategy against buggy or poorly run sync servers (or break-ins)

## A Different Context

# ***Trust-Minimised Server Infrastructure***

- Servers are still *very convenient* in 2025
- What do we *trust* servers to do?
  - Hold our bytes and not delete them
  - Only send those bytes to someone with the right permissions
  - Trust them with the knowledge of our auth graph
    - Server knows who (IP address & public key) requests which doc IDs
- Defence-in-depth strategy against buggy or poorly run sync servers (or break-ins)



## A Different Context

# ***Trust-Minimised Server Infrastructure***

- Servers are still *very convenient* in 2025
- What do we *trust* servers to do?
  - Hold our bytes and not delete them
  - Only send those bytes to someone with the right permissions
  - Trust them with the knowledge of our auth graph
    - Server knows who (IP address & public key) requests which doc IDs
- Defence-in-depth strategy against buggy or poorly run sync servers (or break-ins)





## A Different Context

# *Trust-Minimised Server Infrastructure*

- Servers are still *very convenient* in 2025
- What do we *trust* servers to do?
  - Hold our bytes and not delete them
  - Only send those bytes to someone with the right permissions
  - Trust them with the knowledge of our auth graph
    - Server knows who (IP address & public key) requests which doc IDs
- Defence-in-depth strategy against buggy or poorly run sync servers (or break-ins)



Securing Data Wherever It Happens To Be

# *End-to-End Encryption*



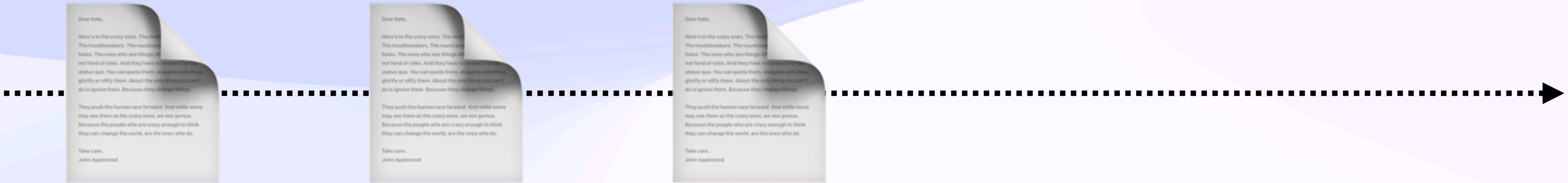
# Self-Healing Concurrent Group Encryption

## ***Security Over Time***



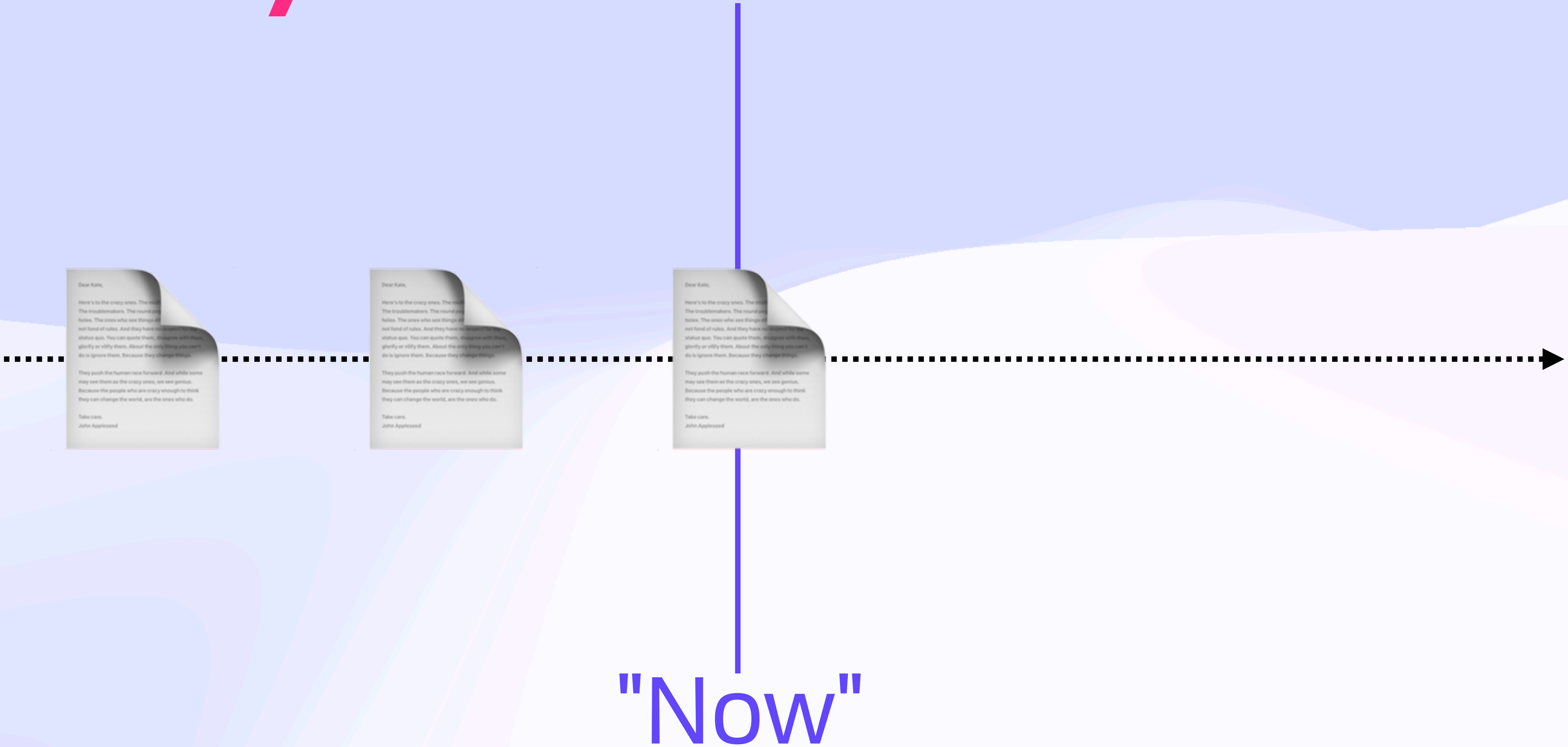
# Self-Healing Concurrent Group Encryption

## *Security Over Time*



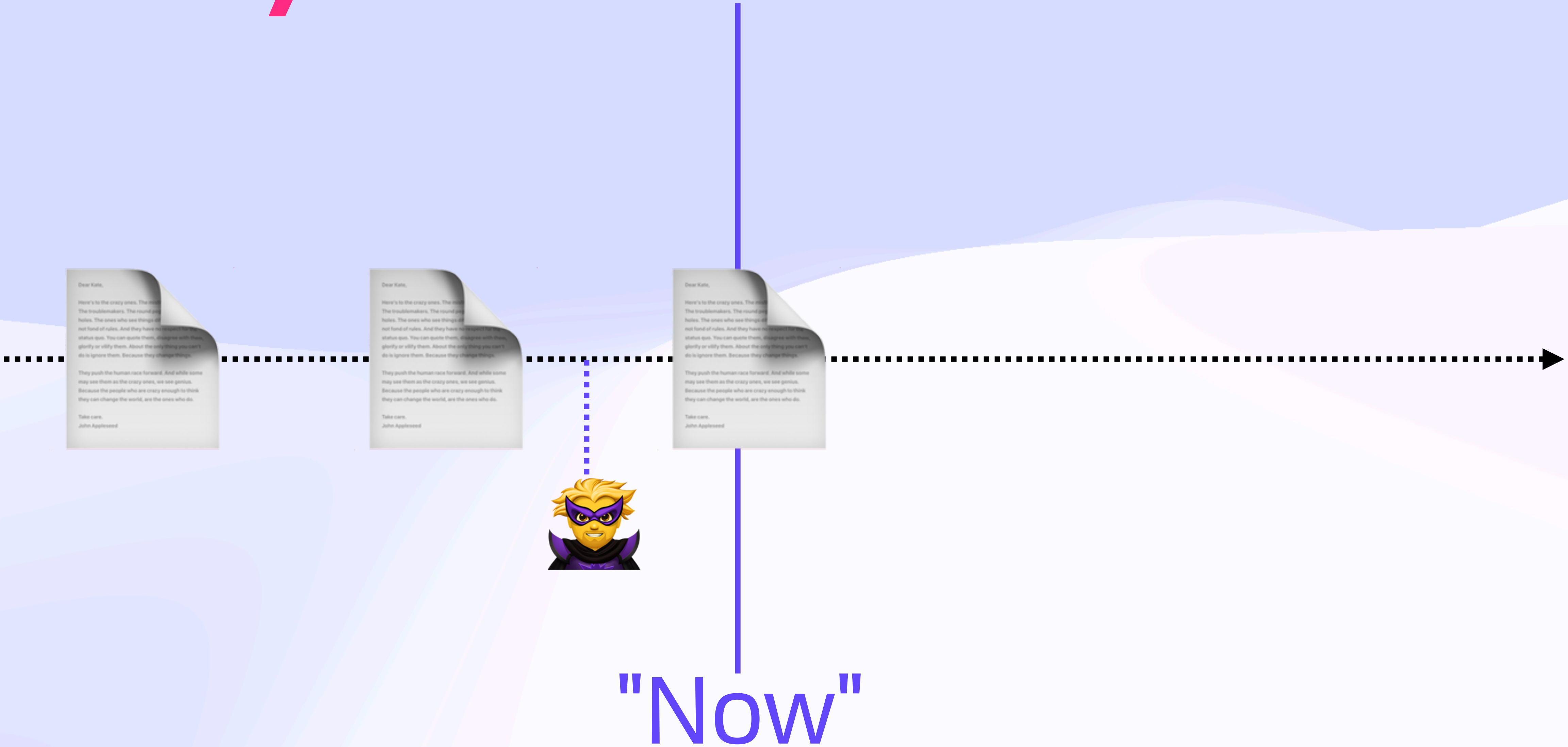
# Self-Healing Concurrent Group Encryption

## *Security Over Time*



# Self-Healing Concurrent Group Encryption

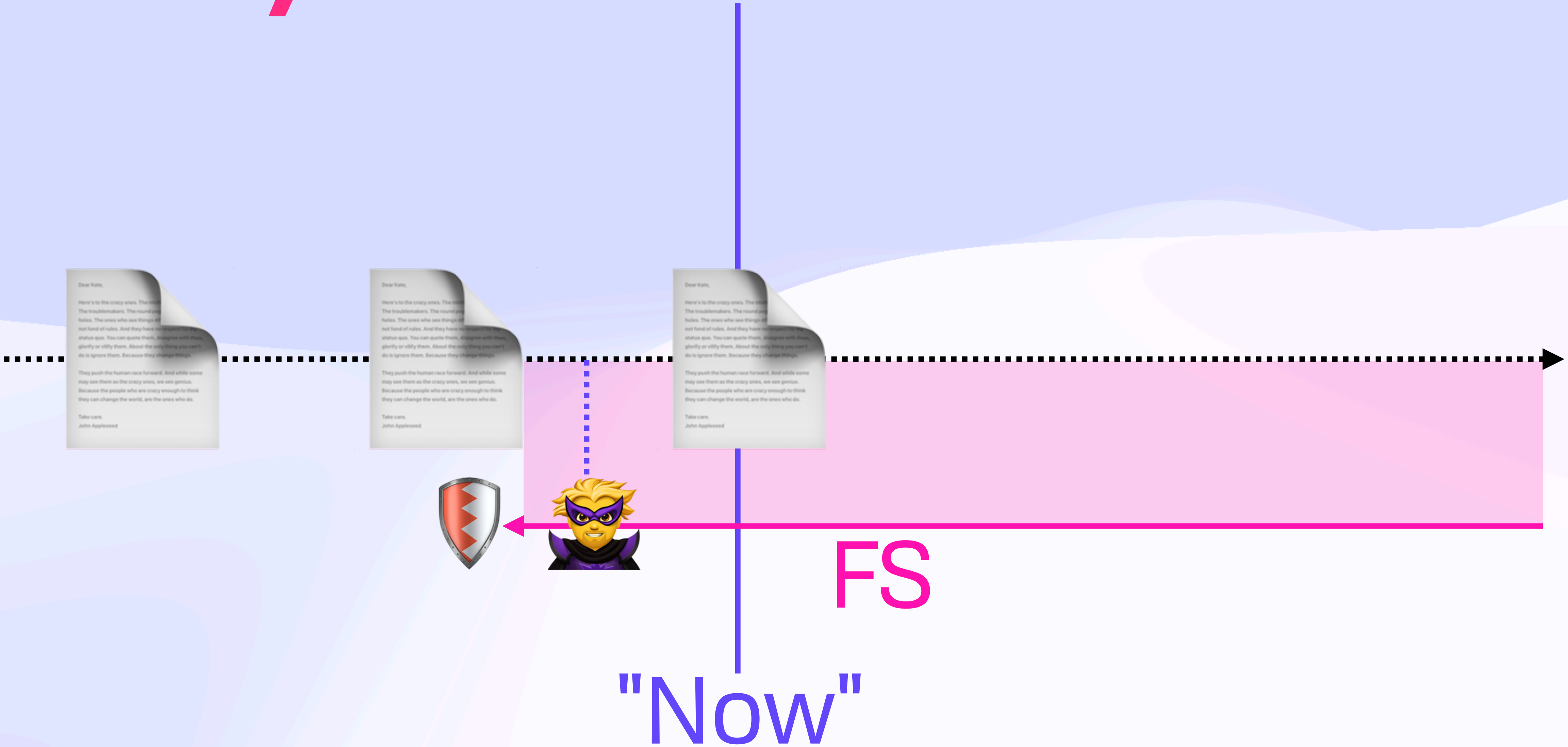
## Security Over Time





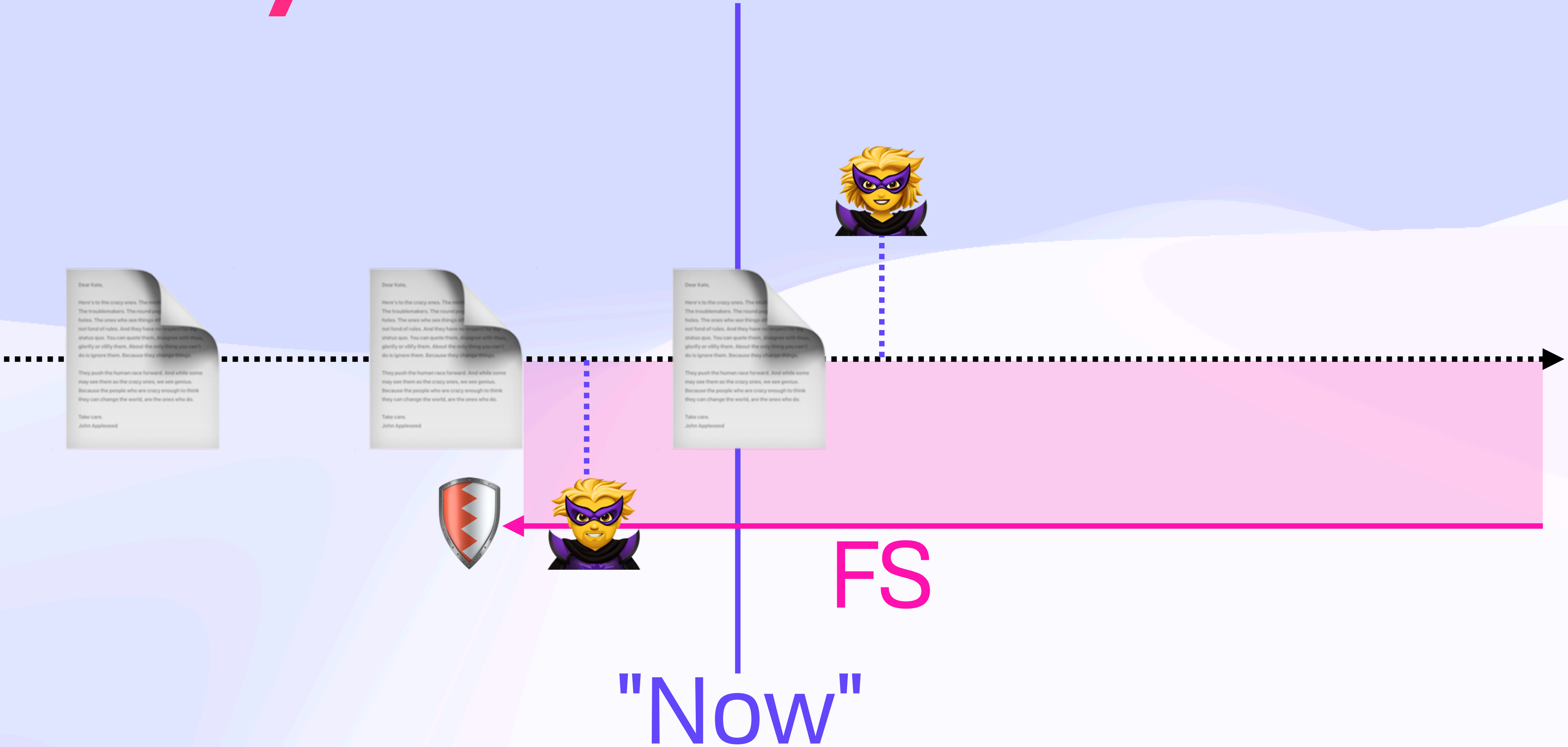
# Self-Healing Concurrent Group Encryption

## Security Over Time



# Self-Healing Concurrent Group Encryption

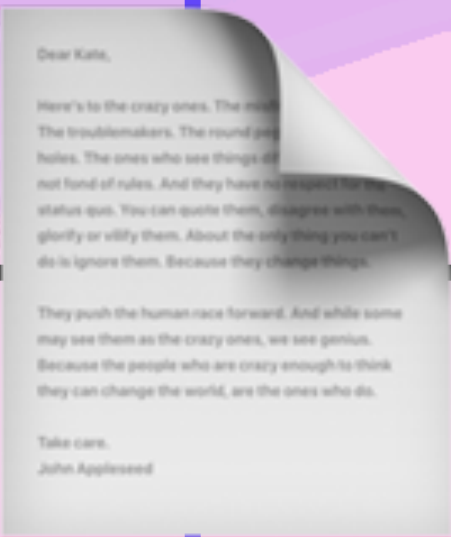
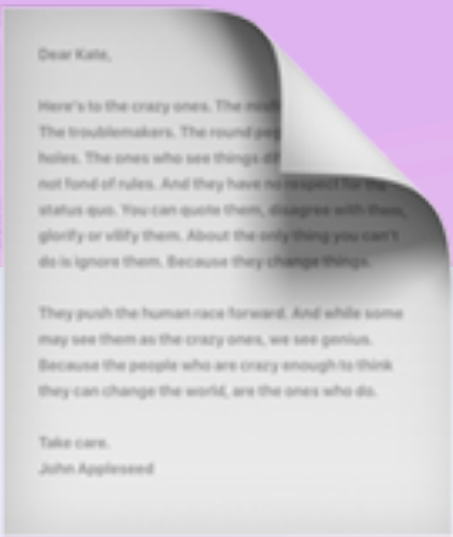
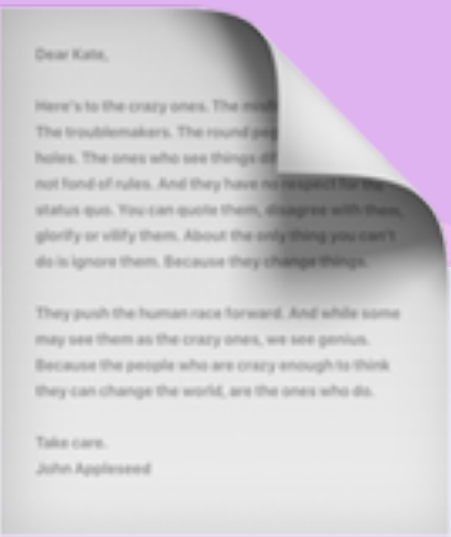
## Security Over Time



# Self-Healing Concurrent Group Encryption

## Security Over Time

PCS



FS

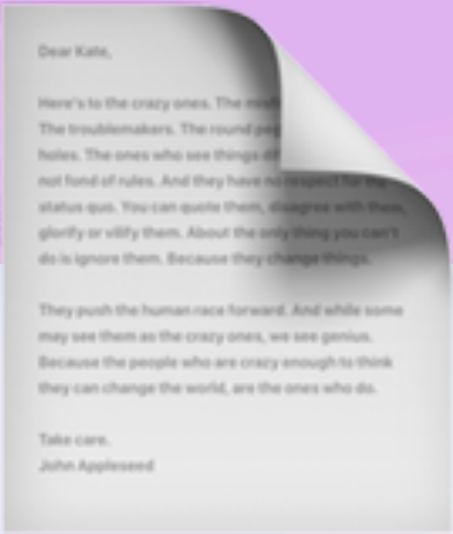
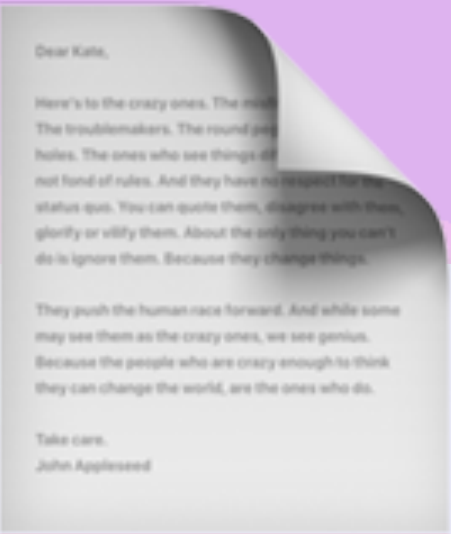
"Now"



# Self-Healing Concurrent Group Encryption

## Security Over Time

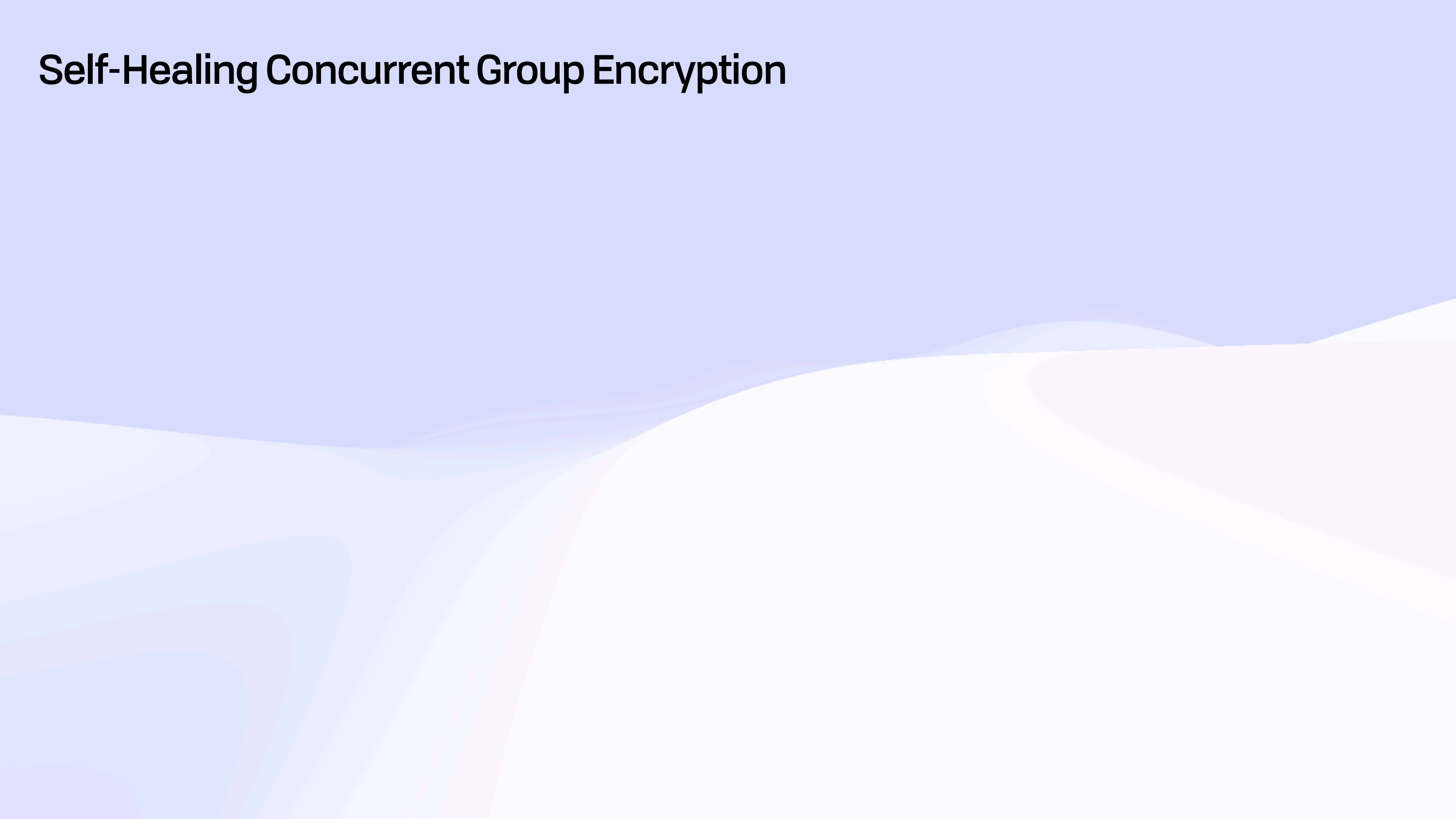
PCS



FS

"Now"

# Self-Healing Concurrent Group Encryption

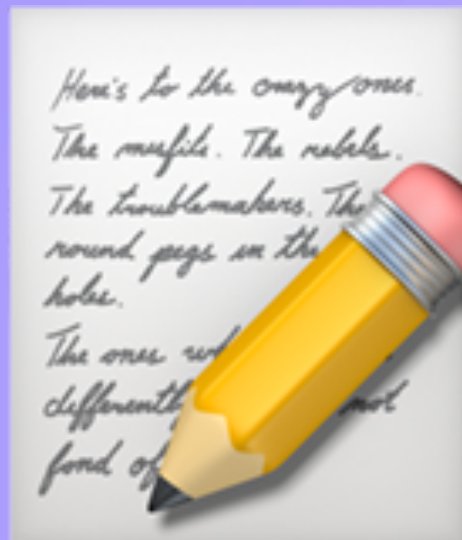
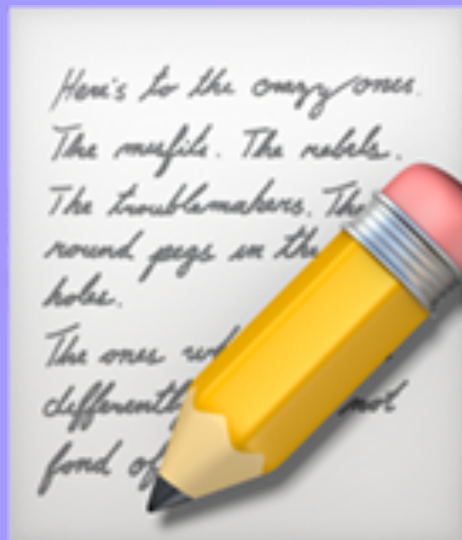
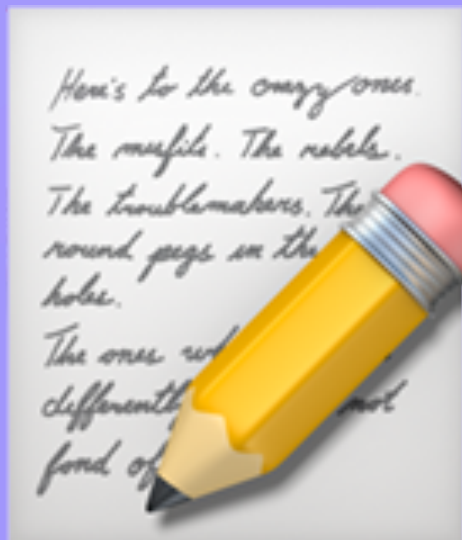


# Self-Healing Concurrent Group Encryption

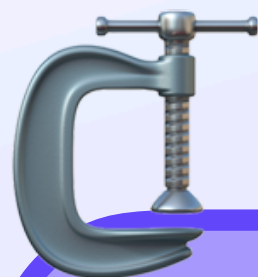




# Self-Healing Concurrent Group Encryption



# Self-Healing Concurrent Group Encryption

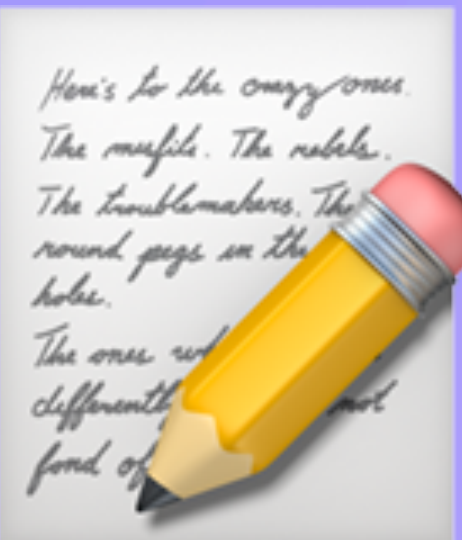
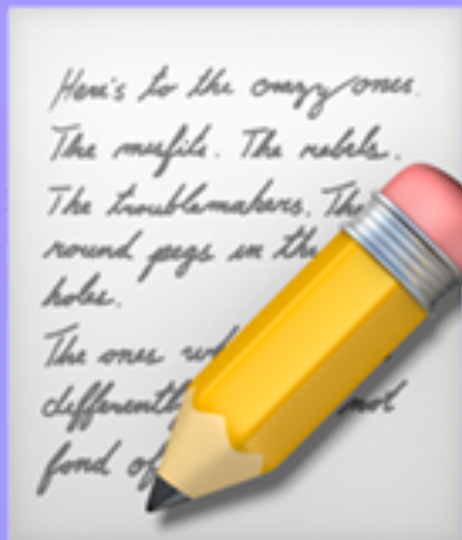
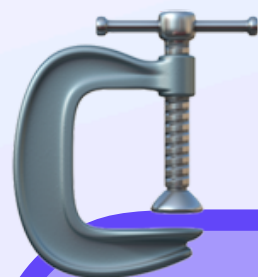


Here's to the crazy ones.  
The mavericks. The rebels.  
The troublemakers. The  
round pegs in the  
holes.  
The ones who  
different. They're not  
fond of

Here's to the crazy ones.  
The mavericks. The rebels.  
The troublemakers. The  
round pegs in the  
holes.  
The ones who  
different. They're not  
fond of

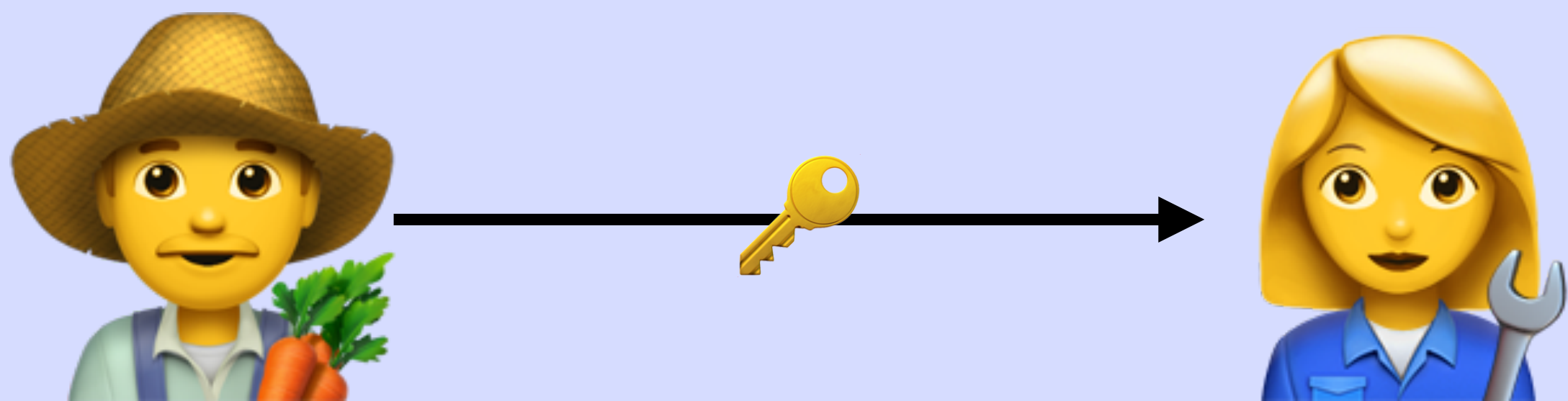
Here's to the crazy ones.  
The mavericks. The rebels.  
The troublemakers. The  
round pegs in the  
holes.  
The ones who  
different. They're not  
fond of

# Self-Healing Concurrent Group Encryption

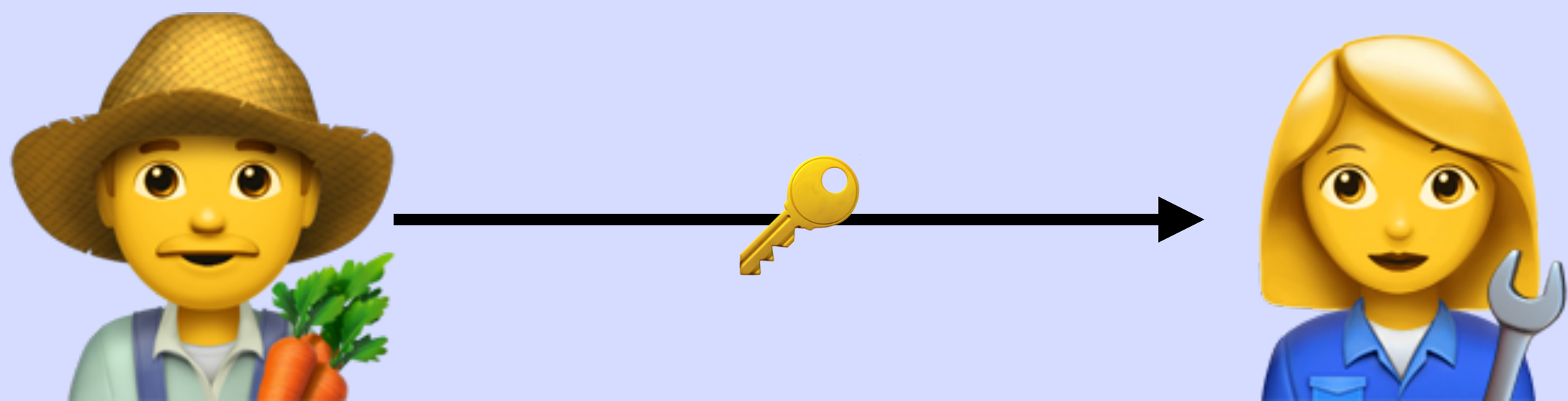




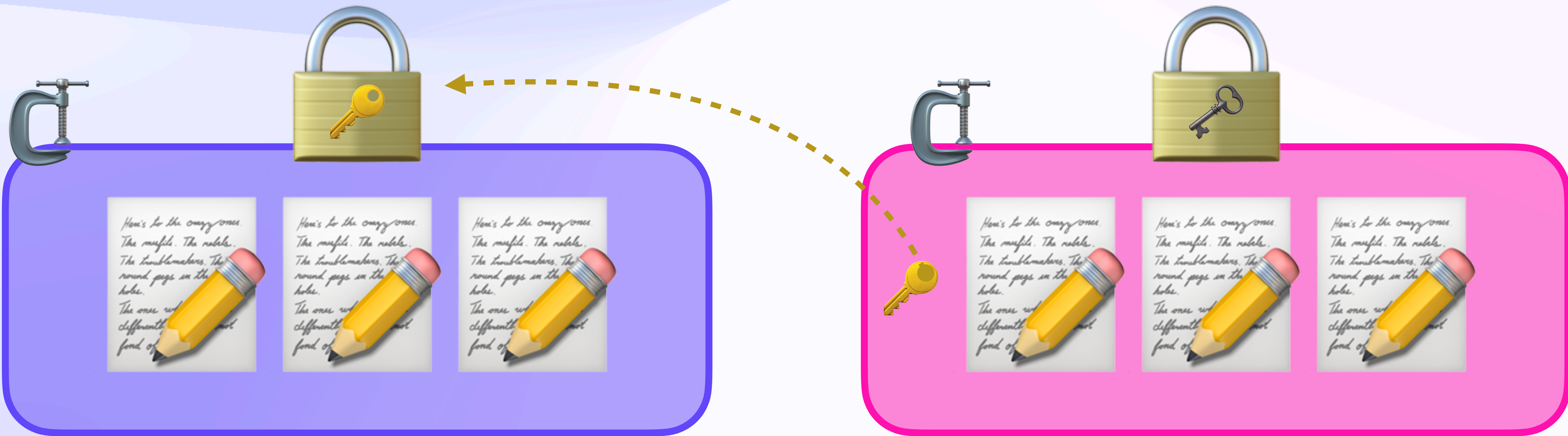
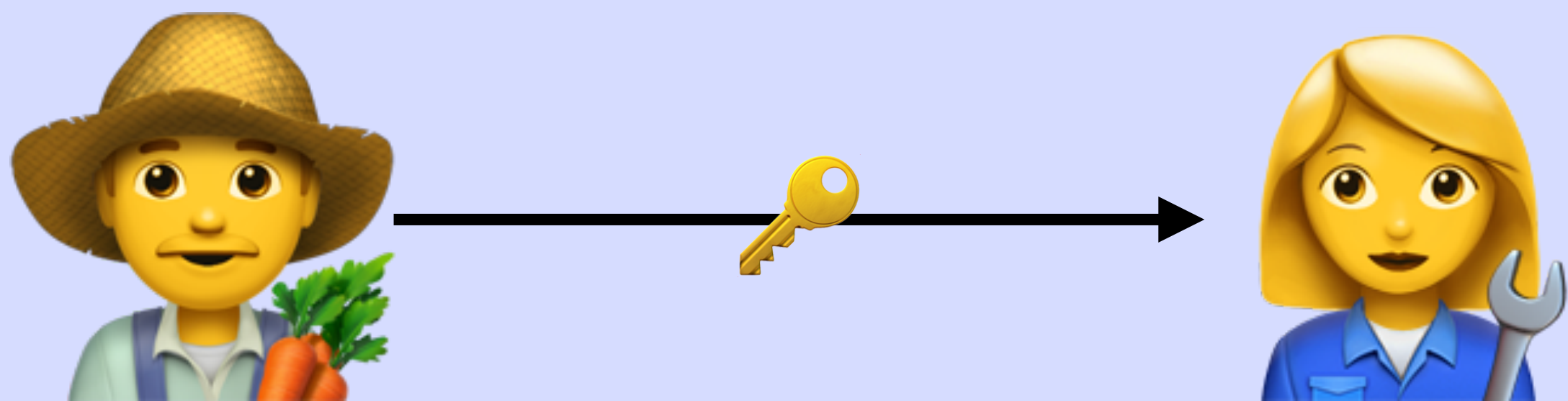
# Self-Healing Concurrent Group Encryption



# Self-Healing Concurrent Group Encryption

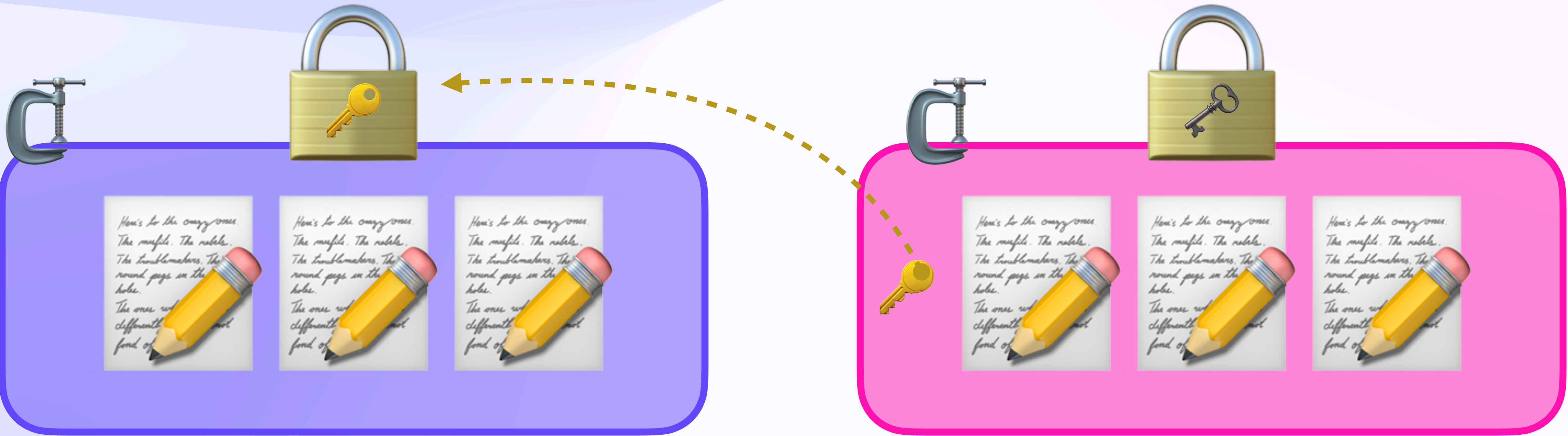
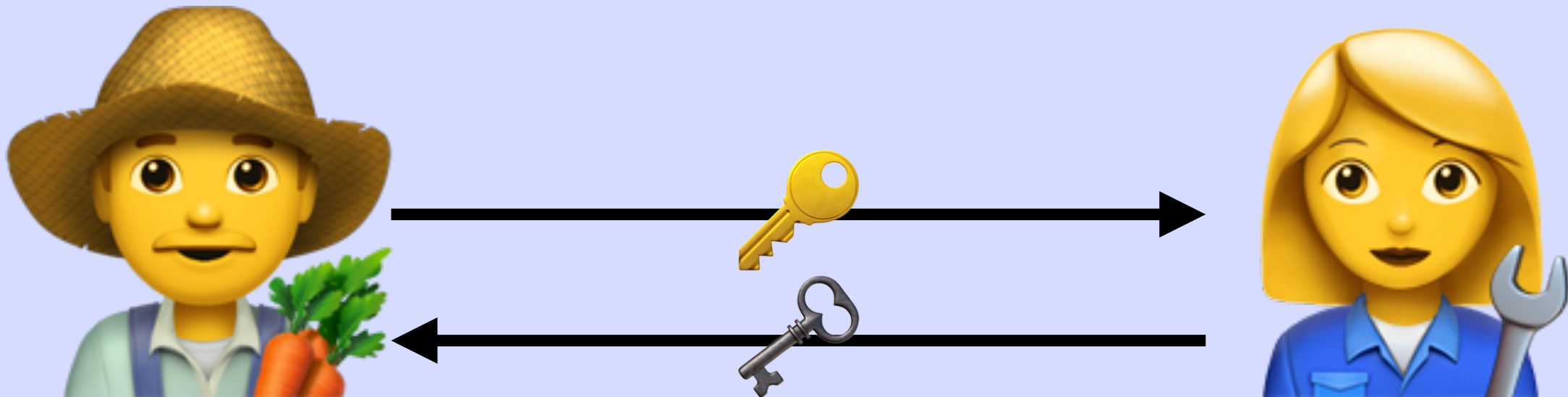


# Self-Healing Concurrent Group Encryption

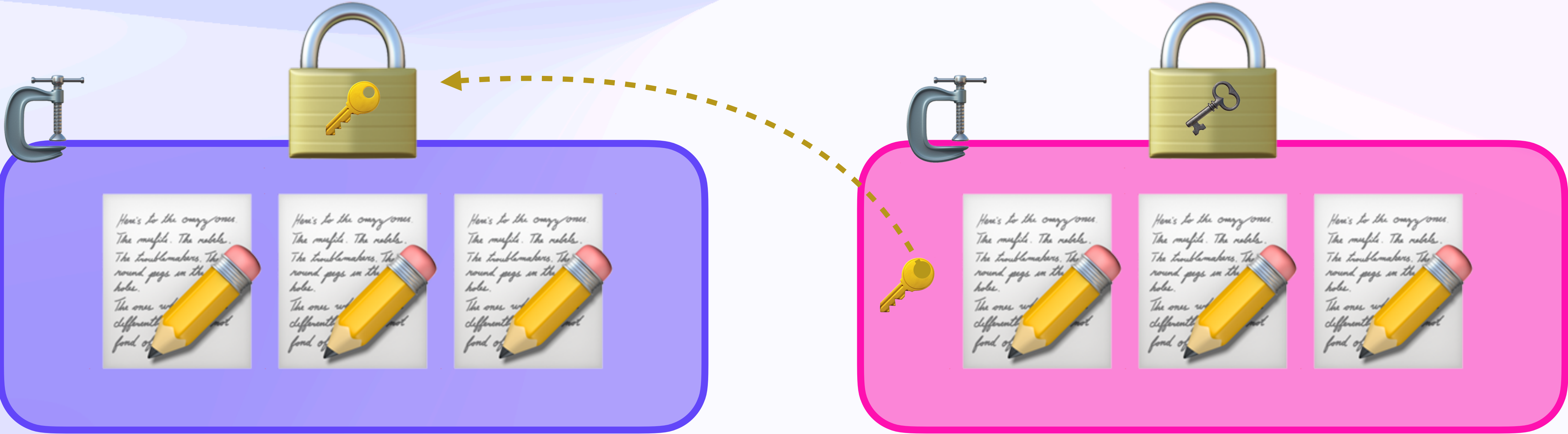
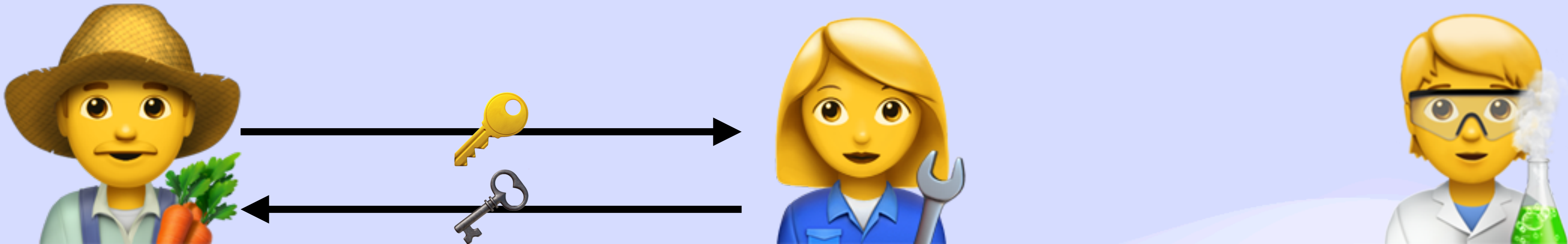




# Self-Healing Concurrent Group Encryption

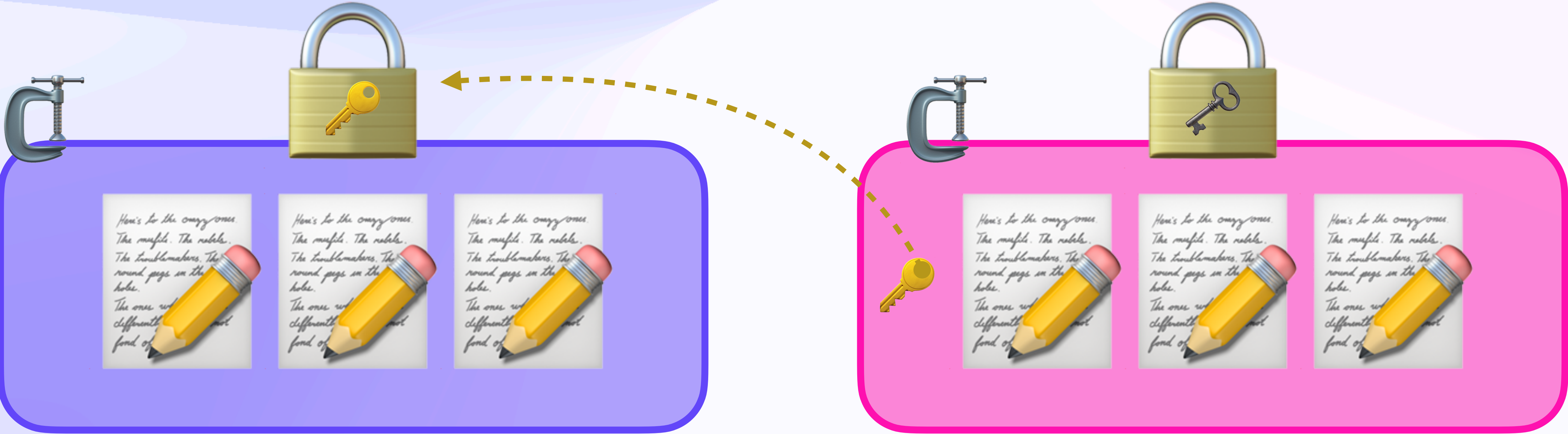
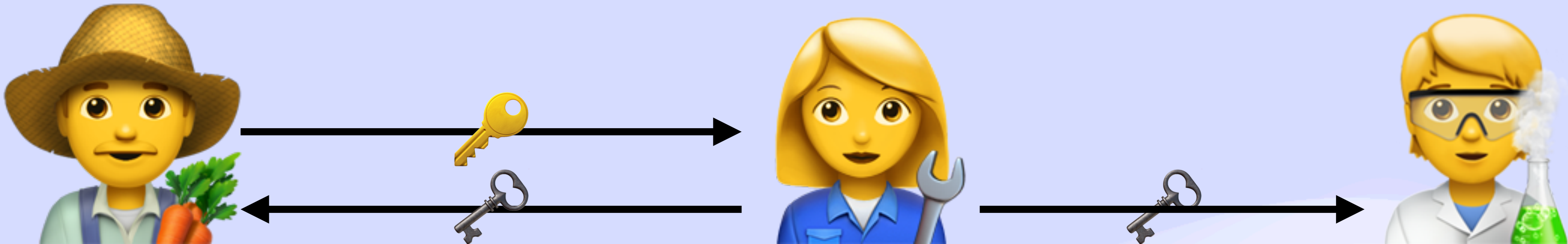


# Self-Healing Concurrent Group Encryption





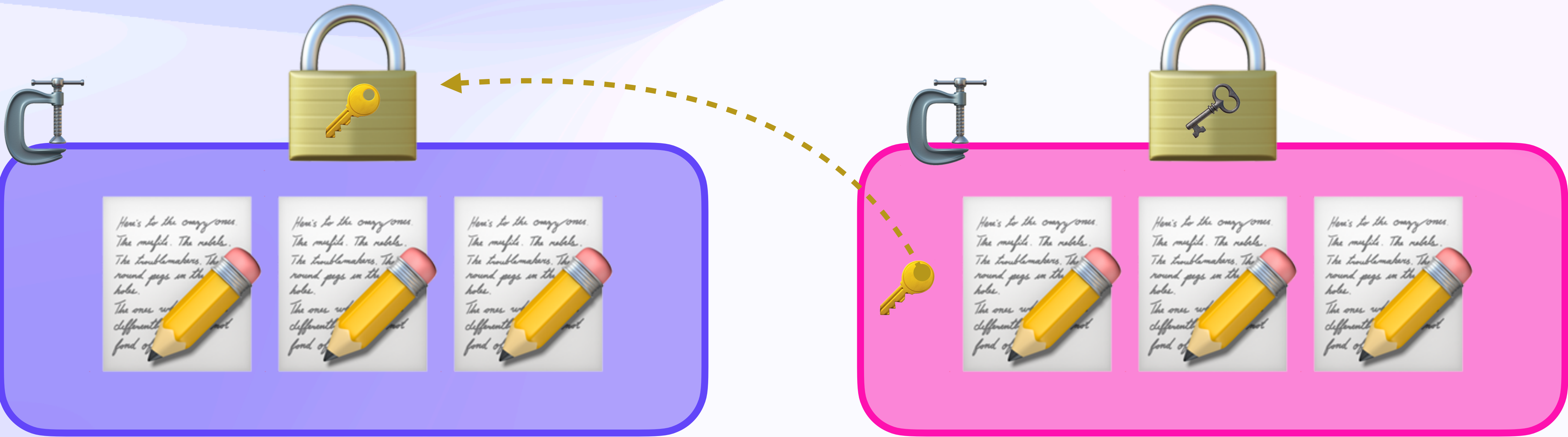
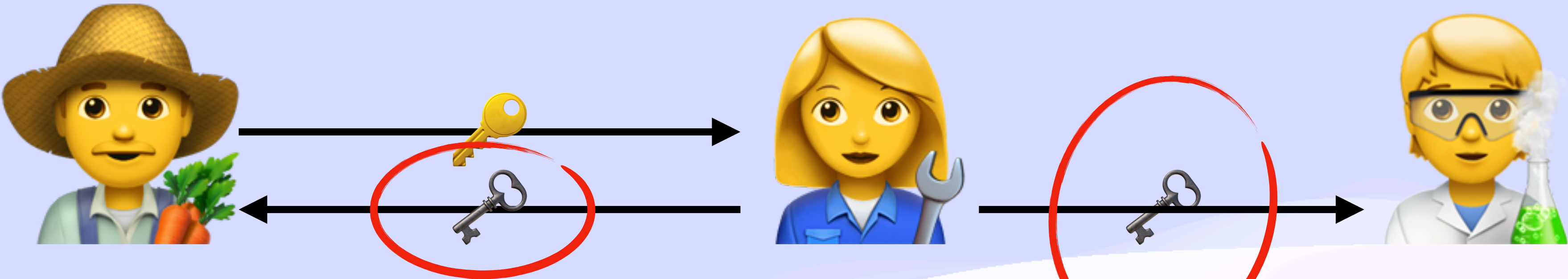
# Self-Healing Concurrent Group Encryption





# Self-Healing Concurrent Group Encryption

Well that's not going to scale to Wikipedia size



Self-Healing Concurrent Group Encryption

***Sooo Many Keys***

# Self-Healing Concurrent Group Encryption

## ***Sooo Many Keys***





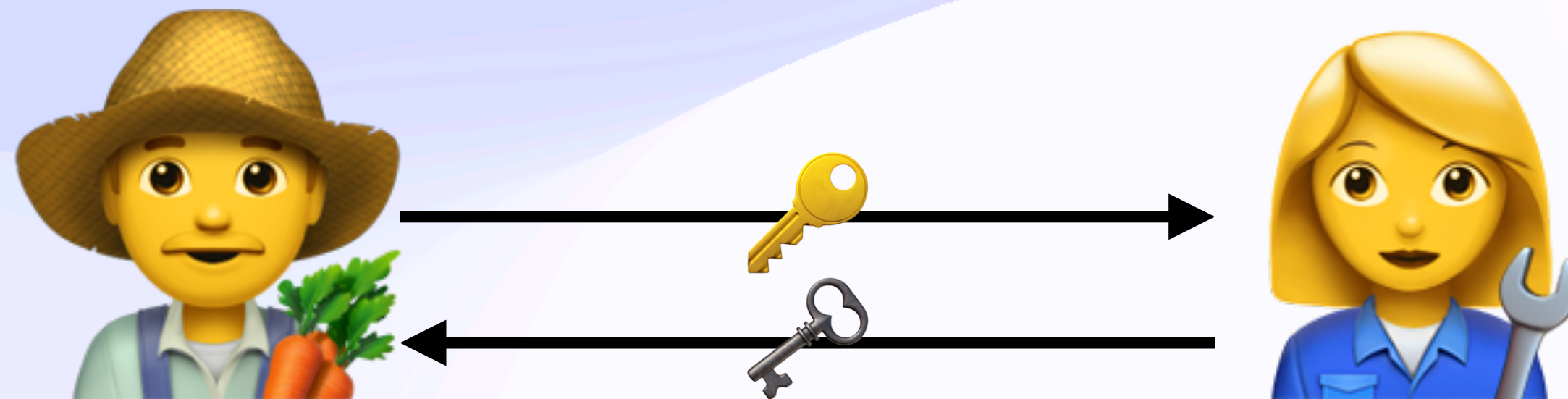
# Self-Healing Concurrent Group Encryption

## ***Sooo Many Keys***



# Self-Healing Concurrent Group Encryption

## ***Sooo Many Keys***



# Self-Healing Concurrent Group Encryption

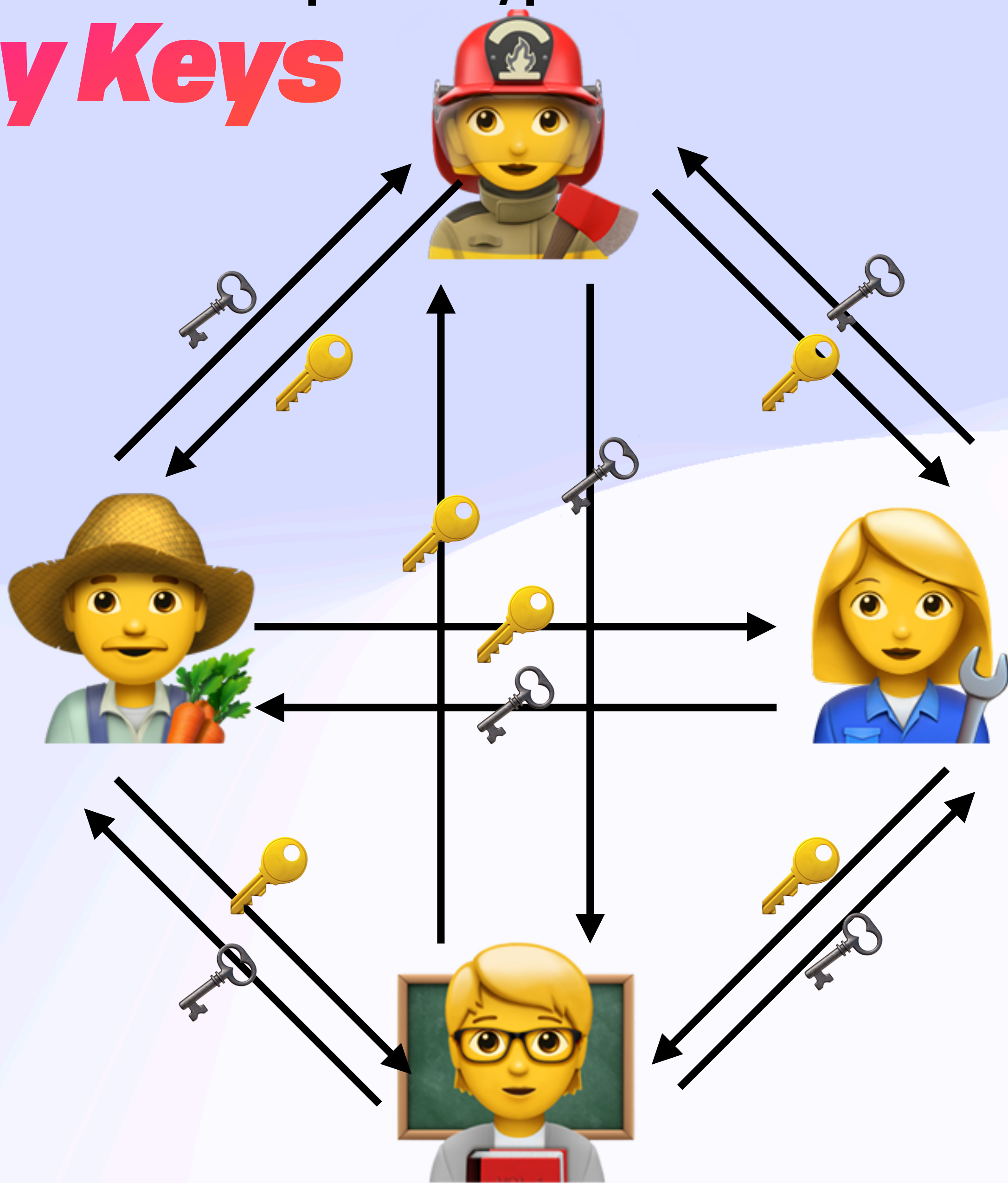
*Sooo Many Keys*





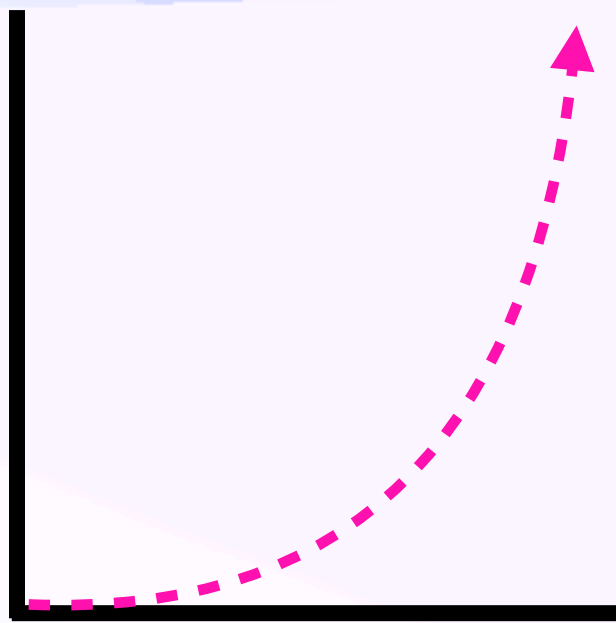
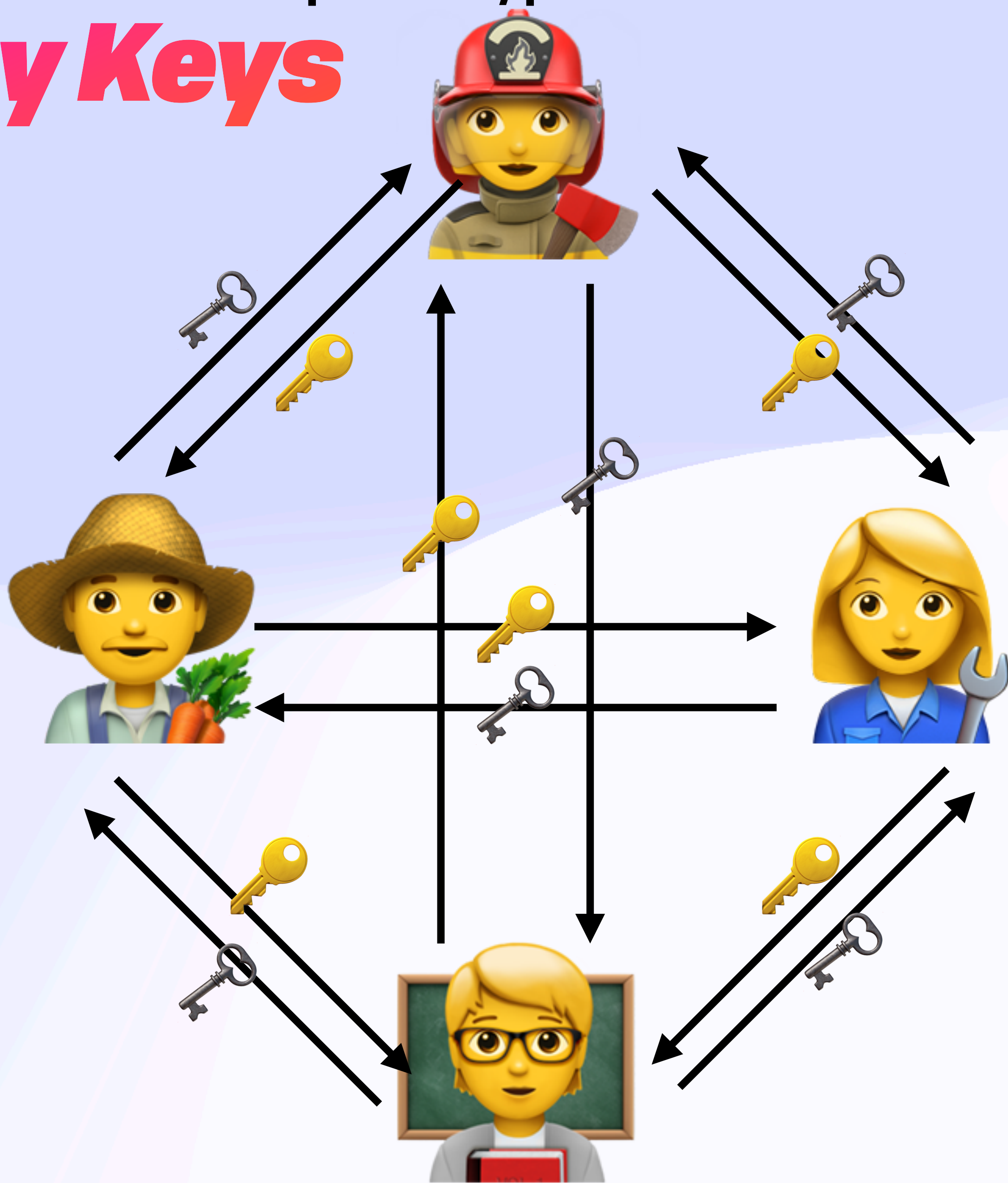
# Self-Healing Concurrent Group Encryption

## *Sooo Many Keys*



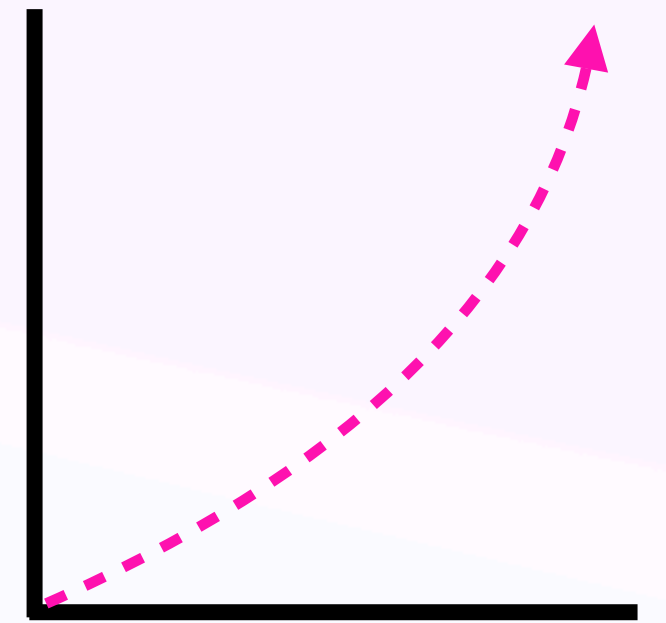
# Self-Healing Concurrent Group Encryption

## *Sooo Many Keys*



Self-Healing Concurrent Group Encryption

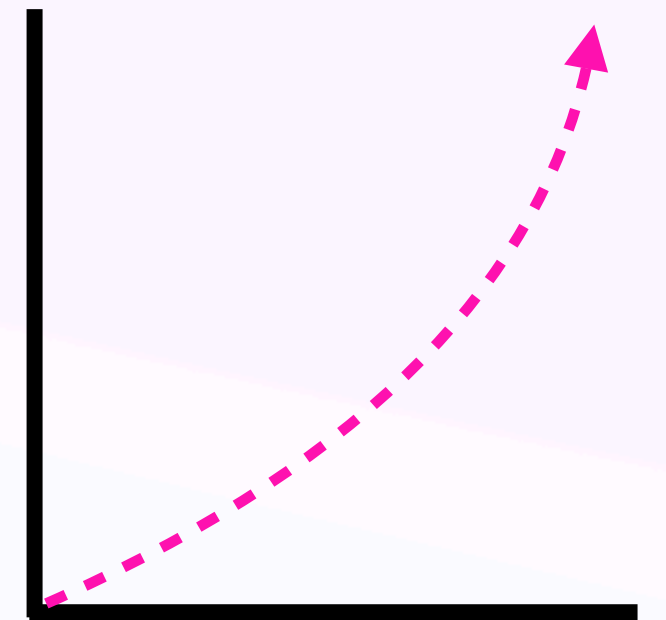
# ***Continuous Group Key Agreement***





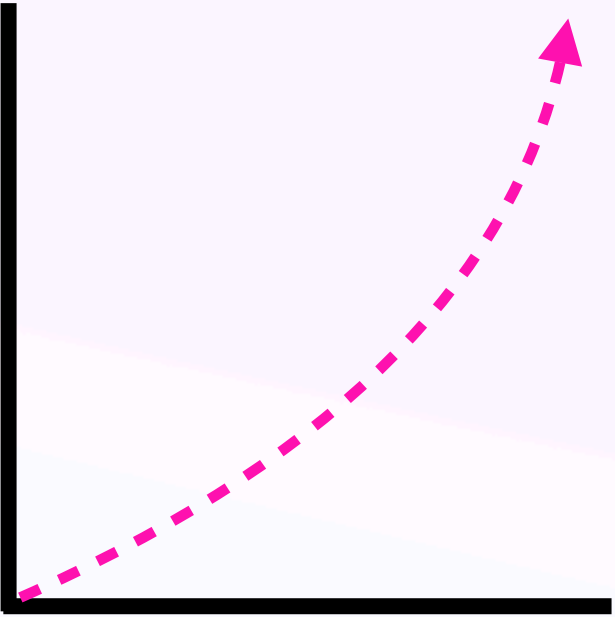
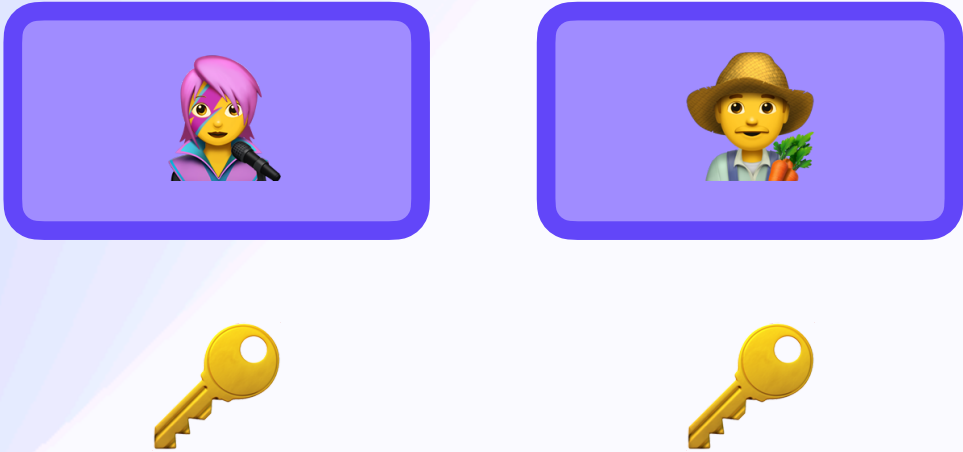
Self-Healing Concurrent Group Encryption

# ***Continuous Group Key Agreement***



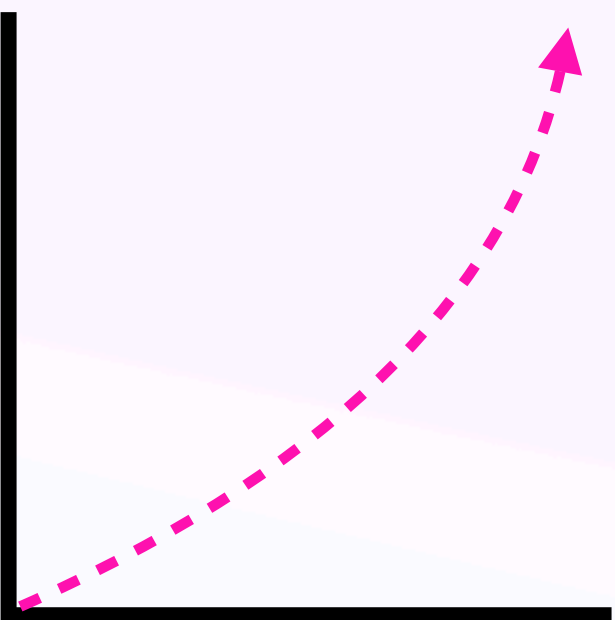
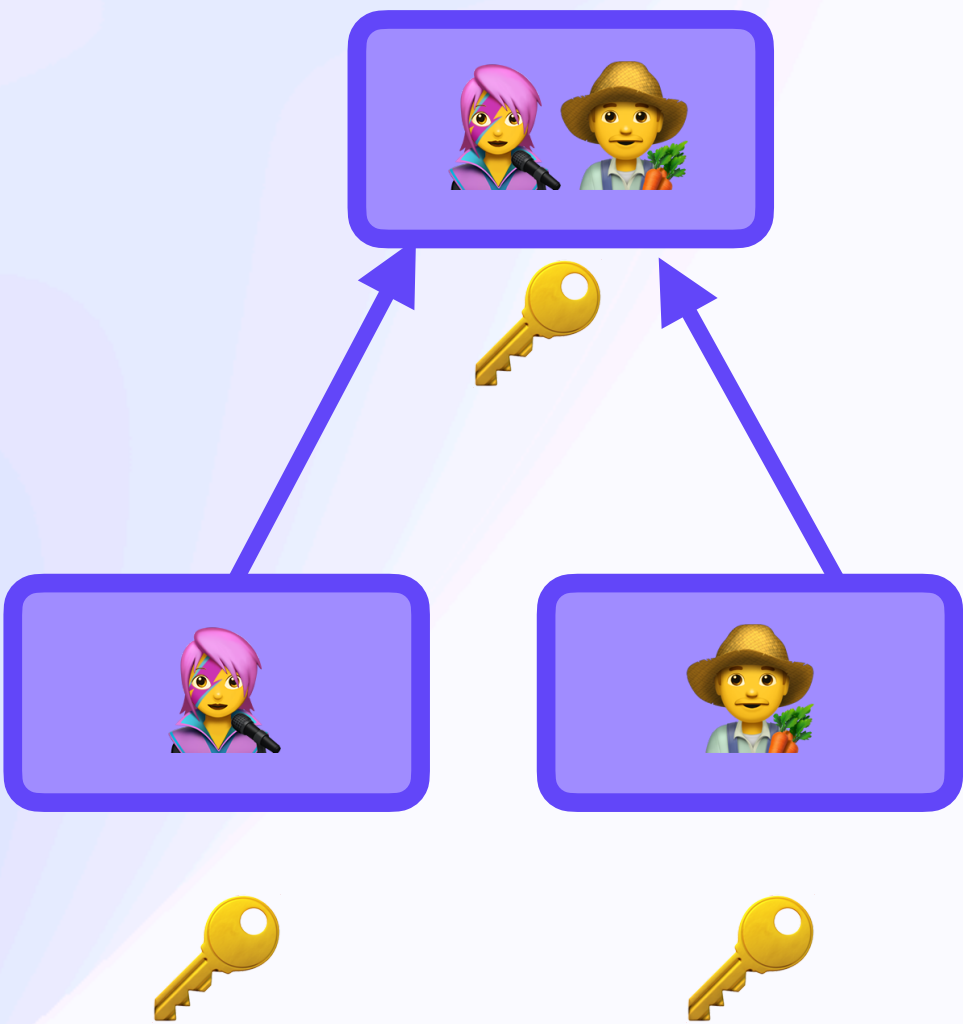
# Self-Healing Concurrent Group Encryption

# *Continuous Group Key Agreement*



# Self-Healing Concurrent Group Encryption

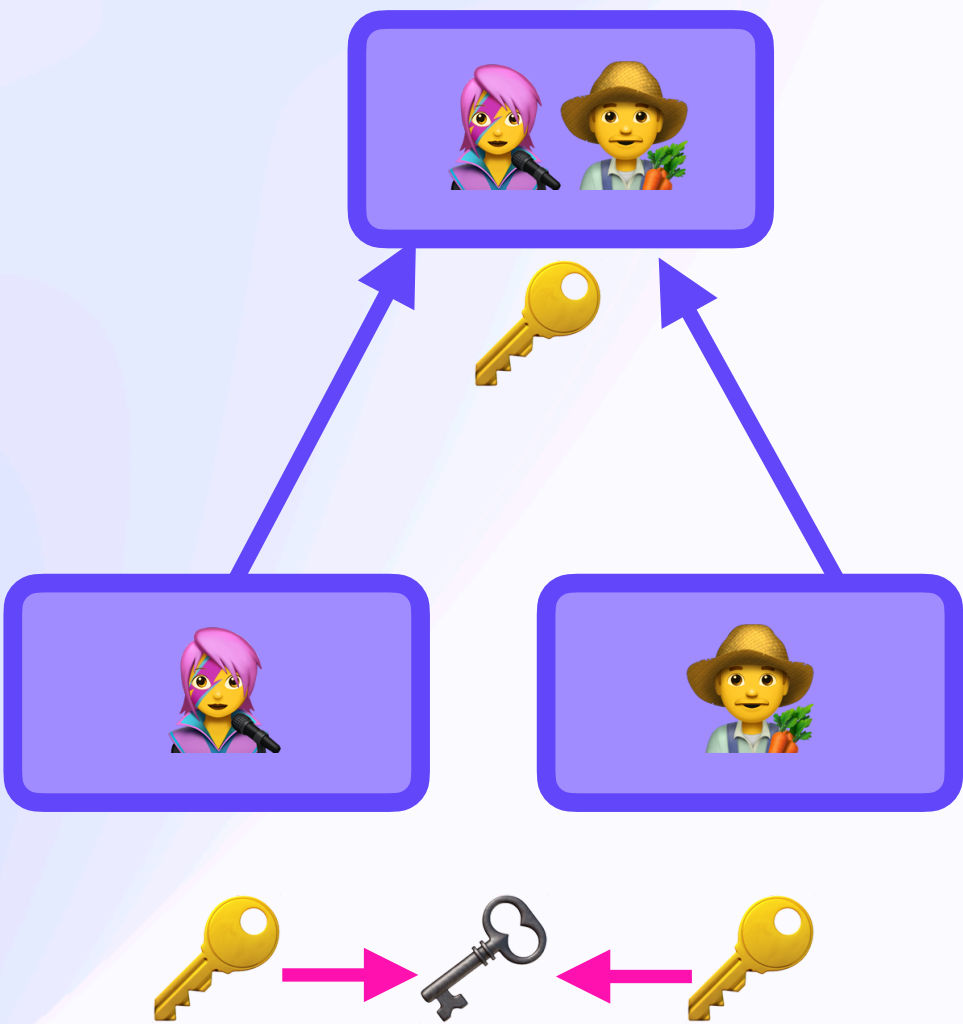
# *Continuous Group Key Agreement*



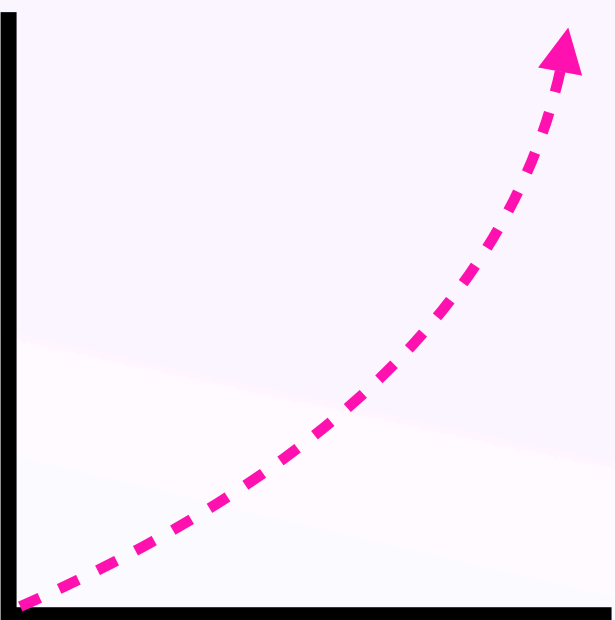


# Self-Healing Concurrent Group Encryption

## *Continuous Group Key Agreement*

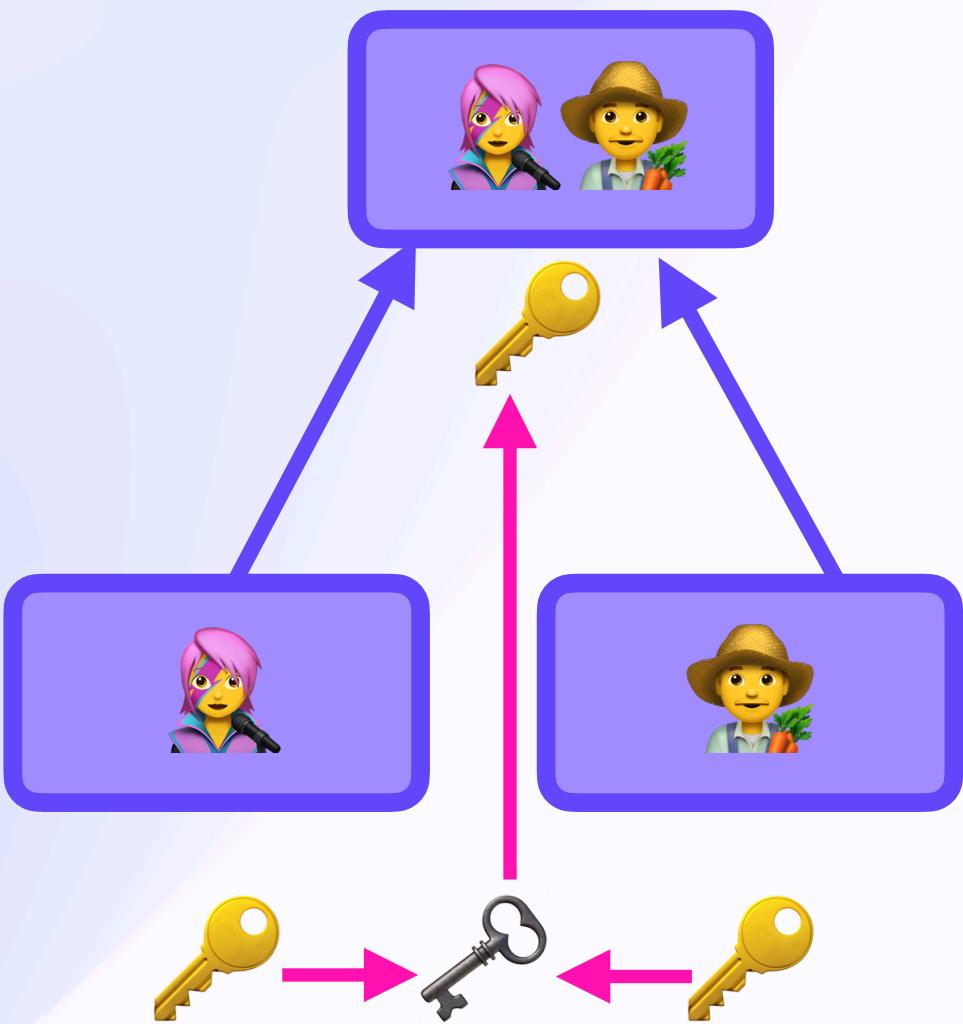


Diffie Hellman

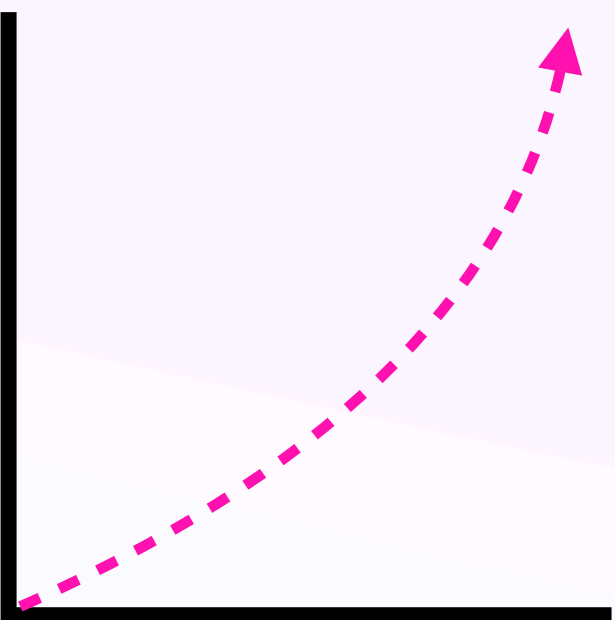


# Self-Healing Concurrent Group Encryption

## *Continuous Group Key Agreement*

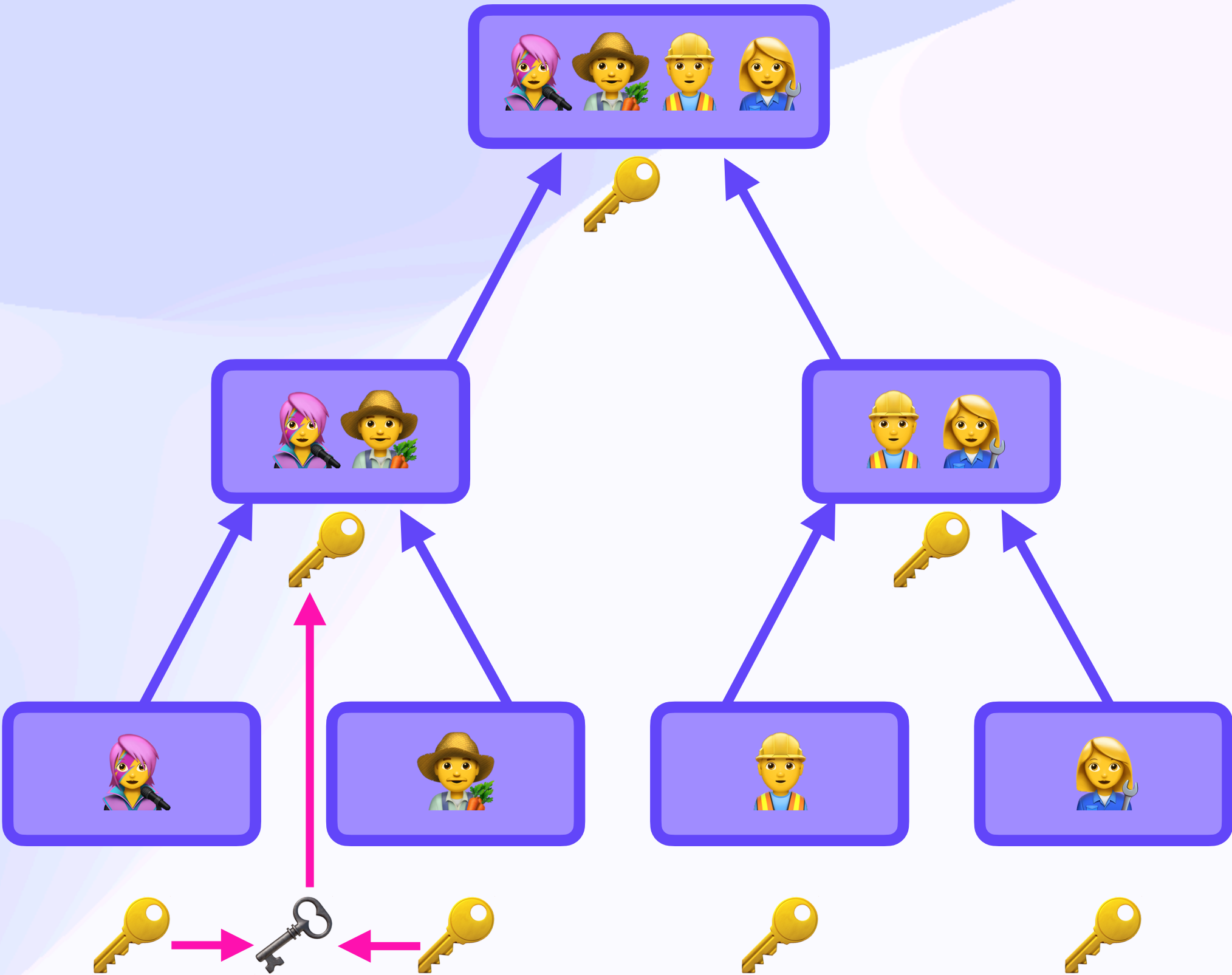


Diffie Hellman

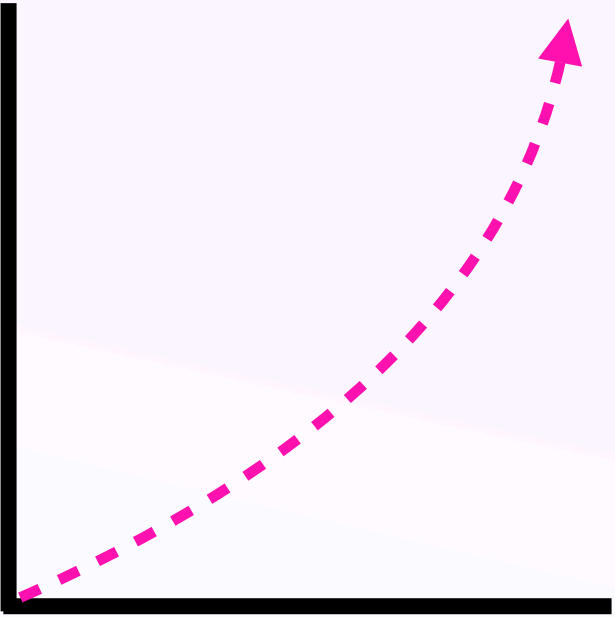


# Self-Healing Concurrent Group Encryption

## *Continuous Group Key Agreement*



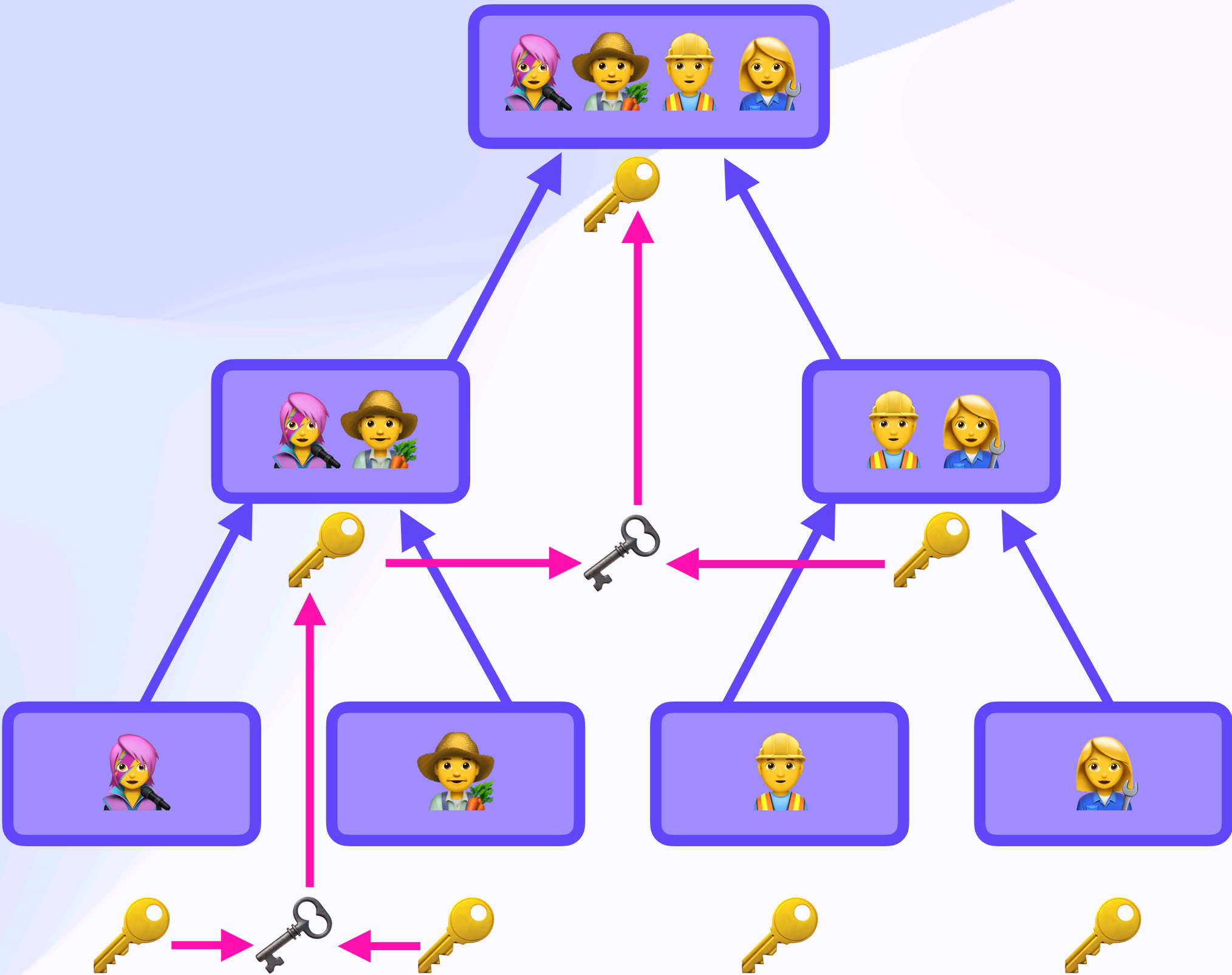
Diffie Hellman



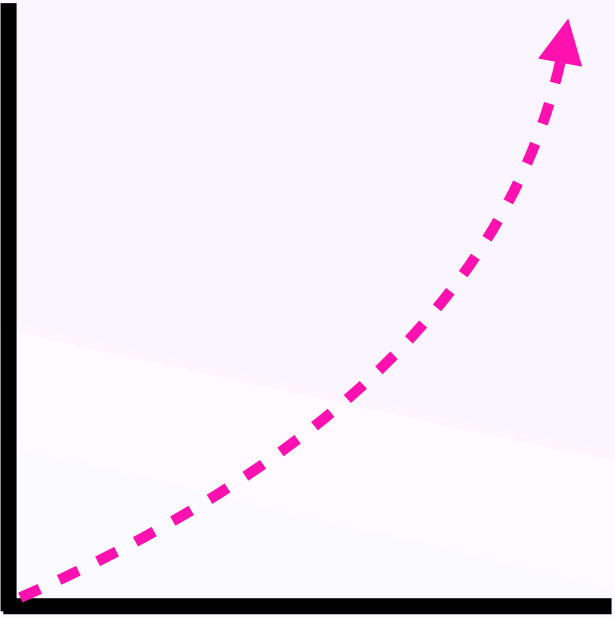


# Self-Healing Concurrent Group Encryption

## *Continuous Group Key Agreement*

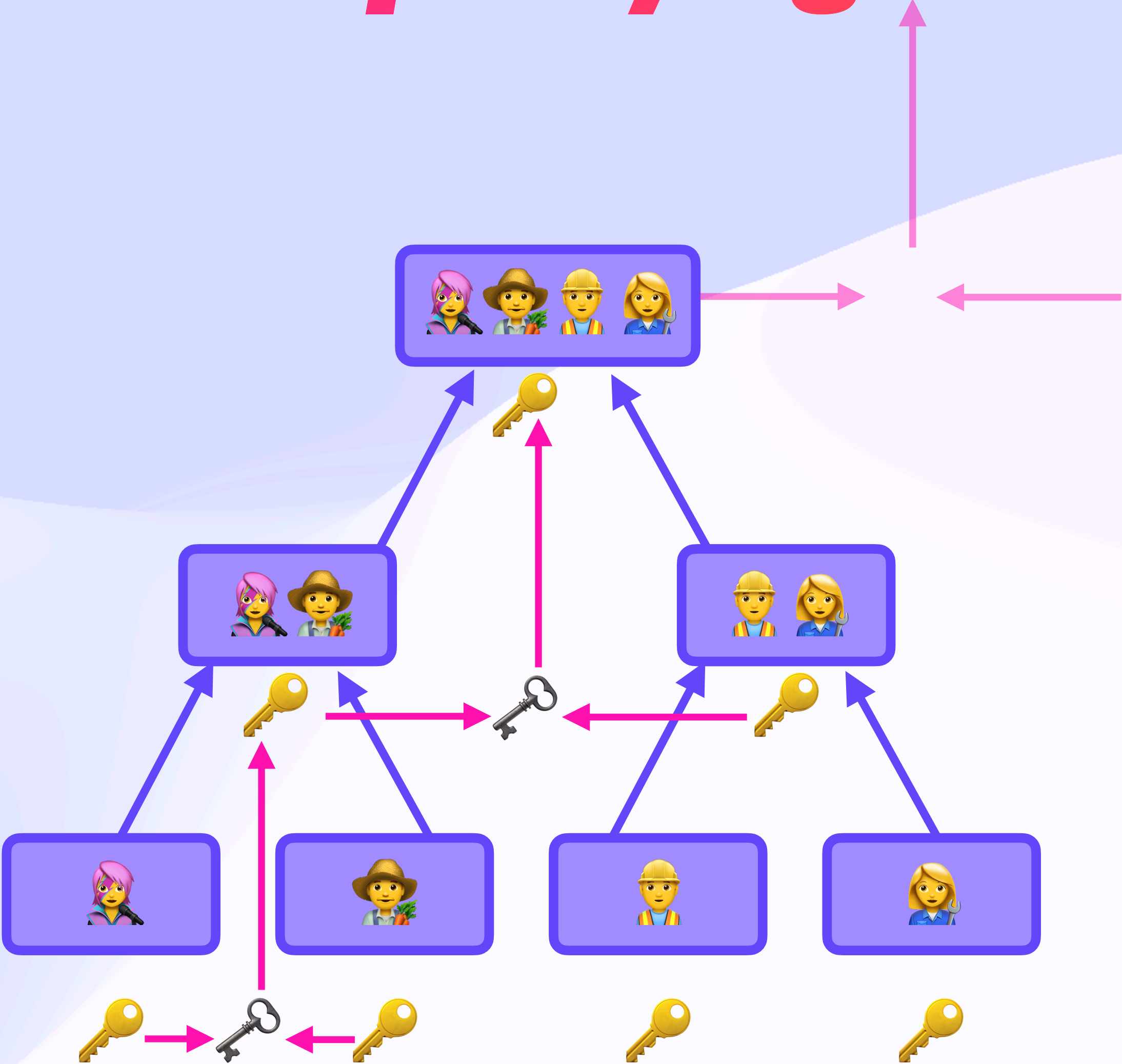


Diffie Hellman



# Self-Healing Concurrent Group Encryption

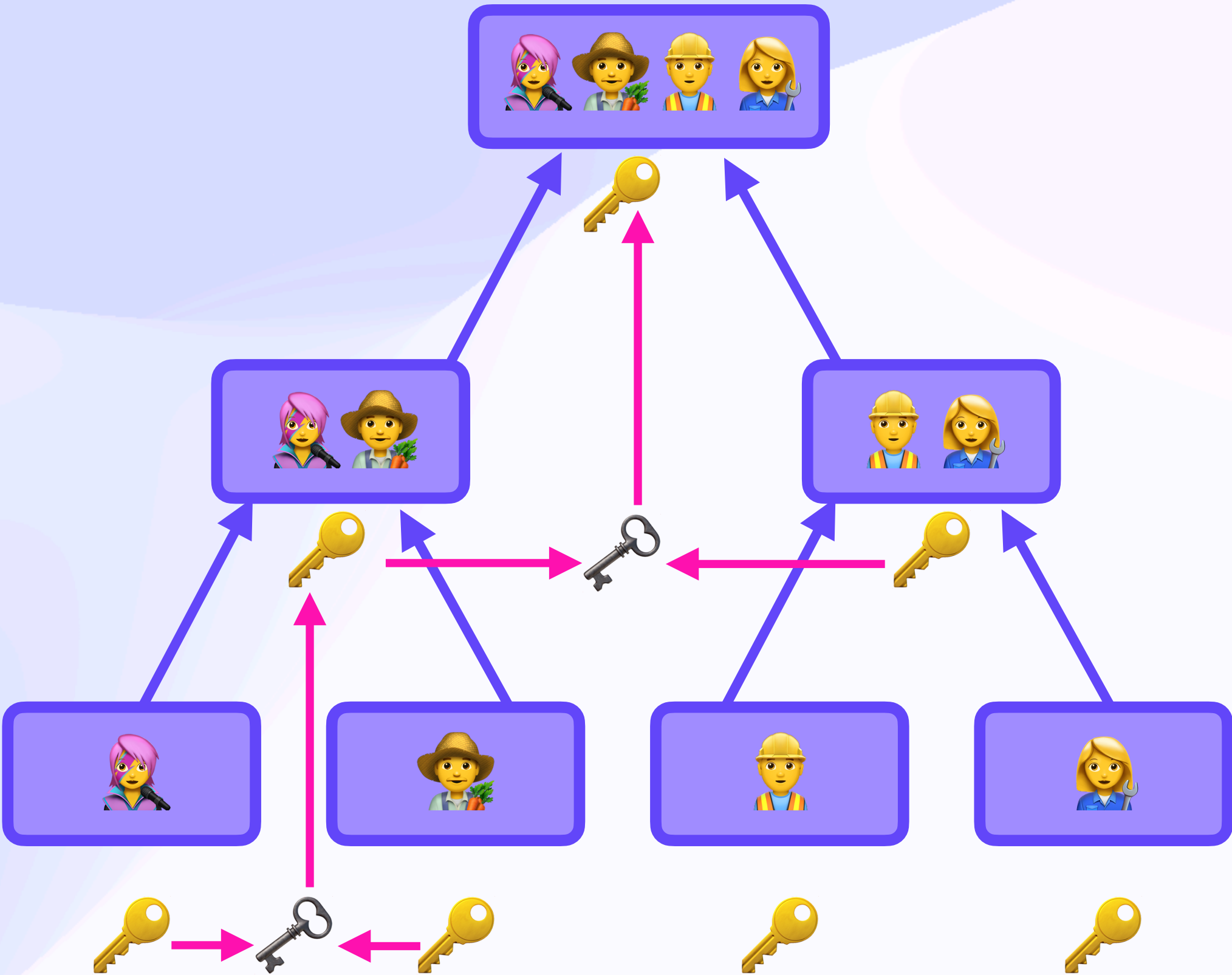
## *Continuous Group Key Agreement*



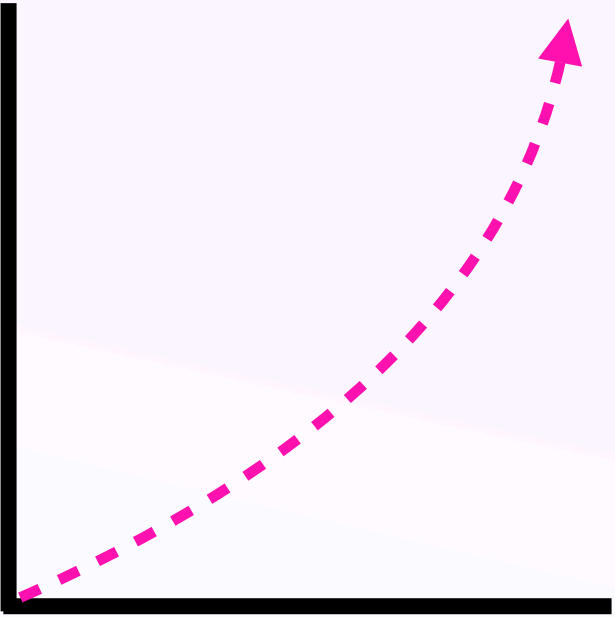
Diffie Hellman

# Self-Healing Concurrent Group Encryption

## *Continuous Group Key Agreement*



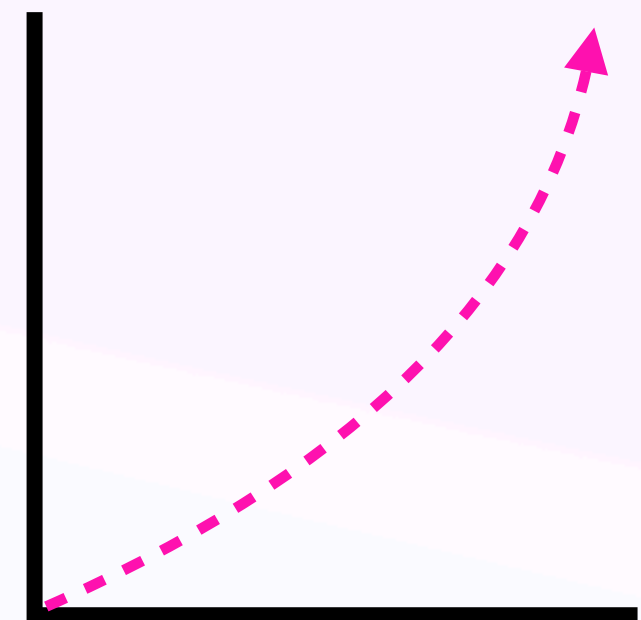
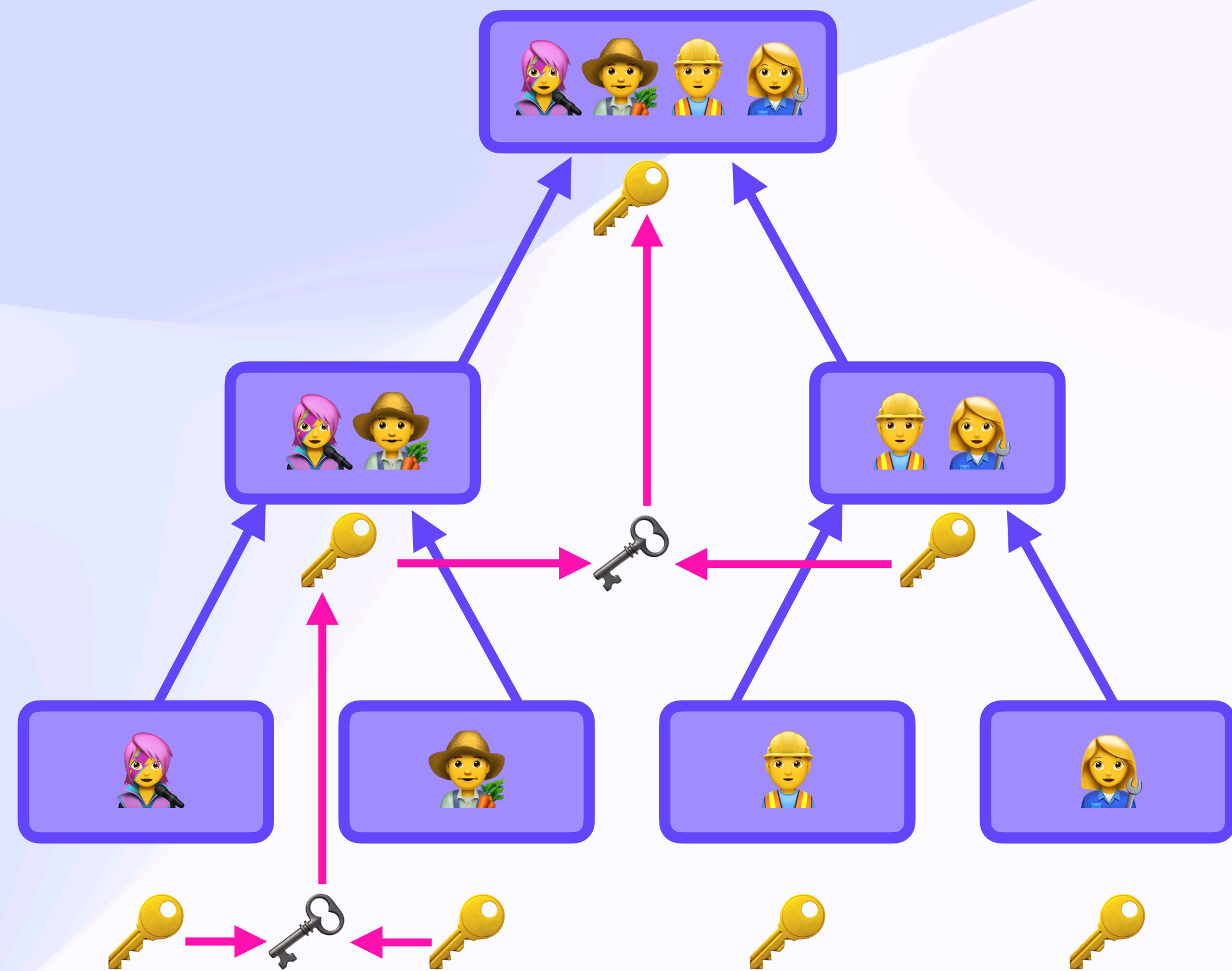
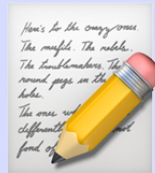
Diffie Hellman





# Self-Healing Concurrent Group Encryption

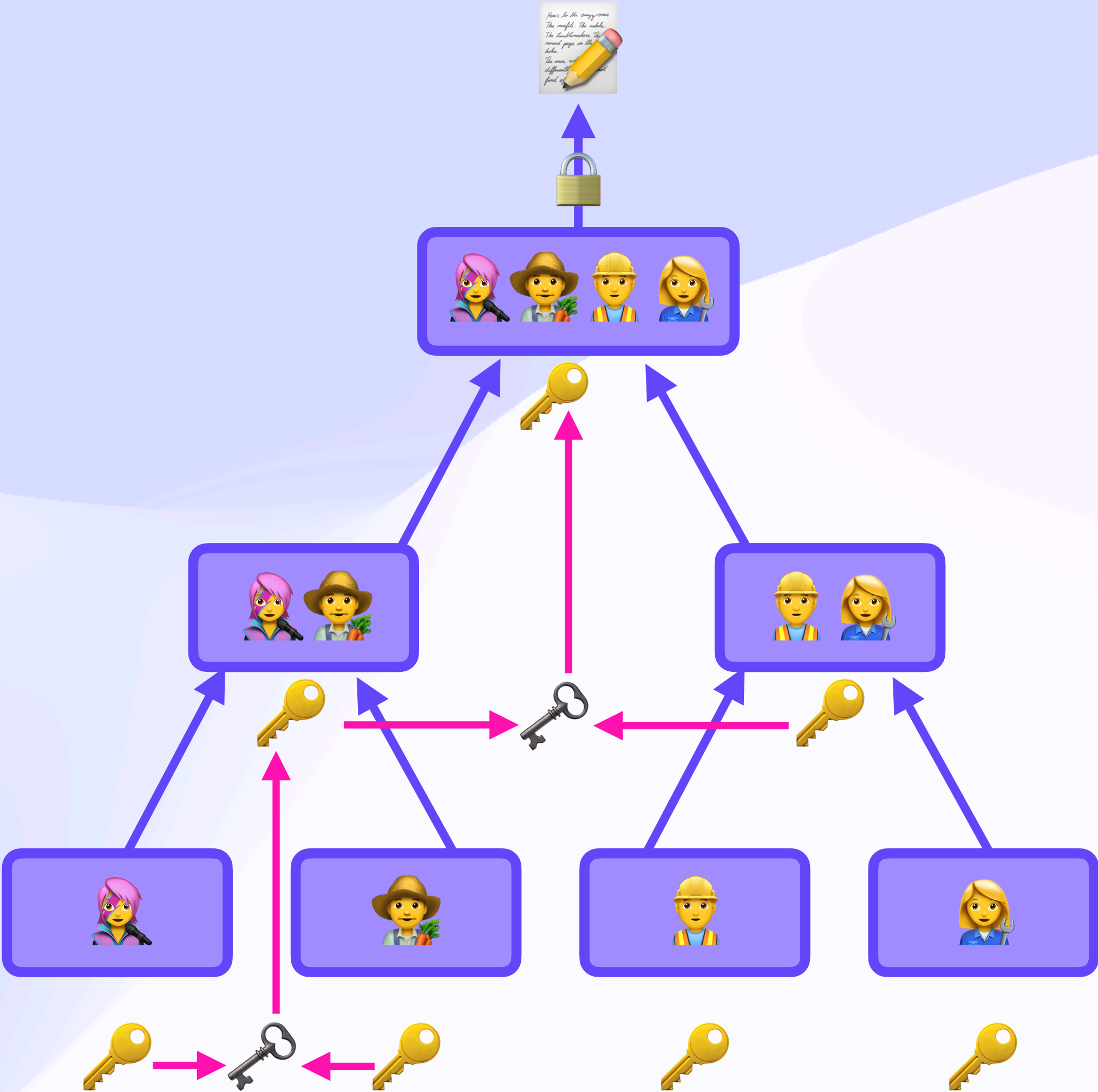
## *Continuous Group Key Agreement*



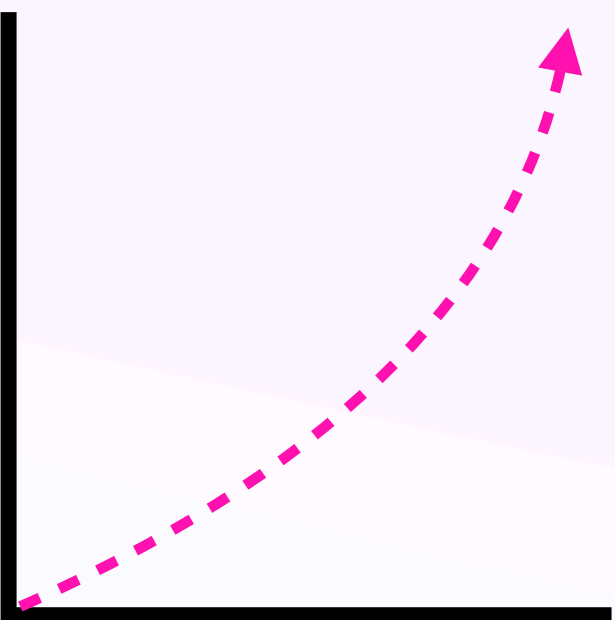
Diffie Hellman

# Self-Healing Concurrent Group Encryption

## *Continuous Group Key Agreement*

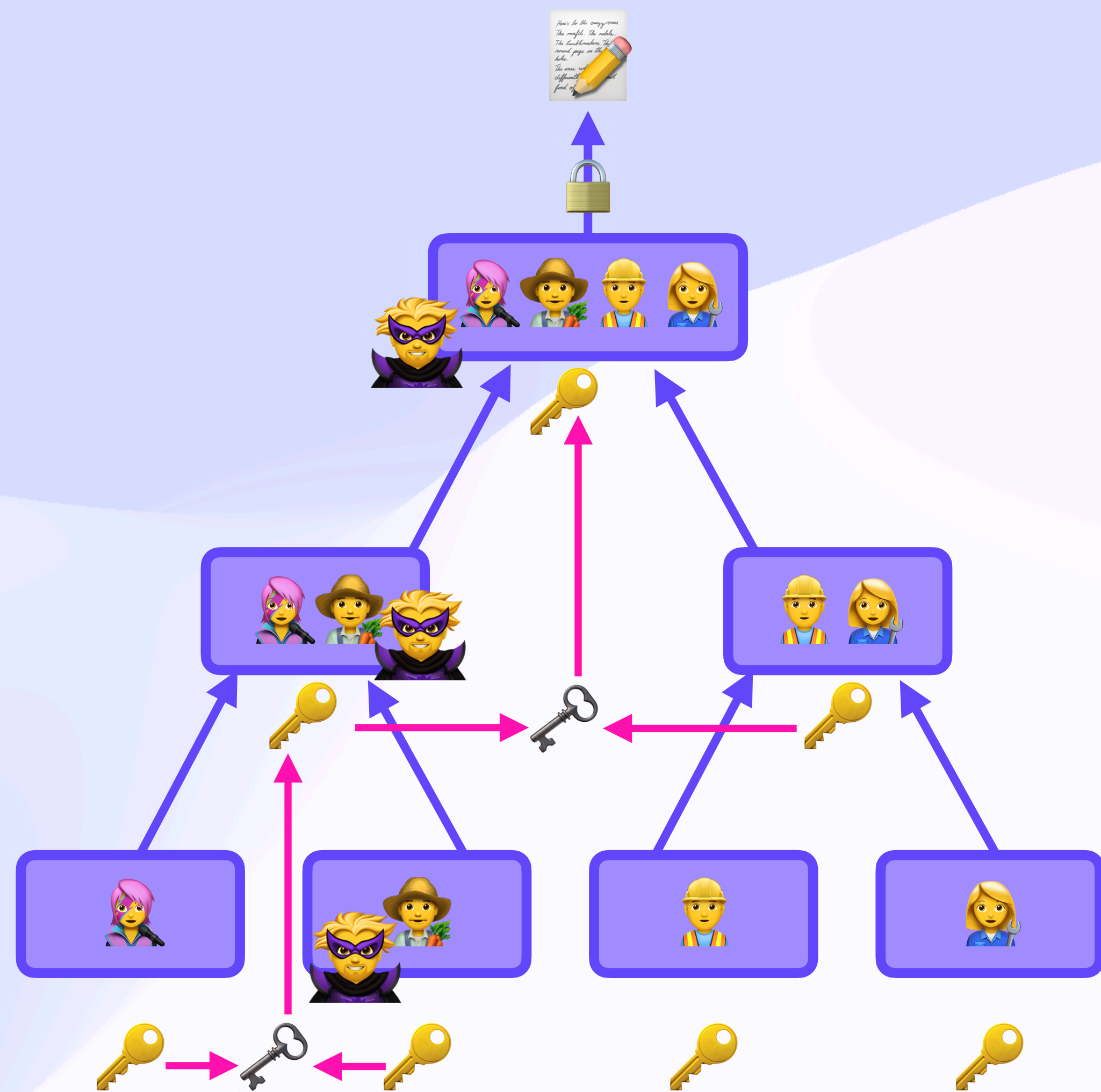


Diffie Hellman

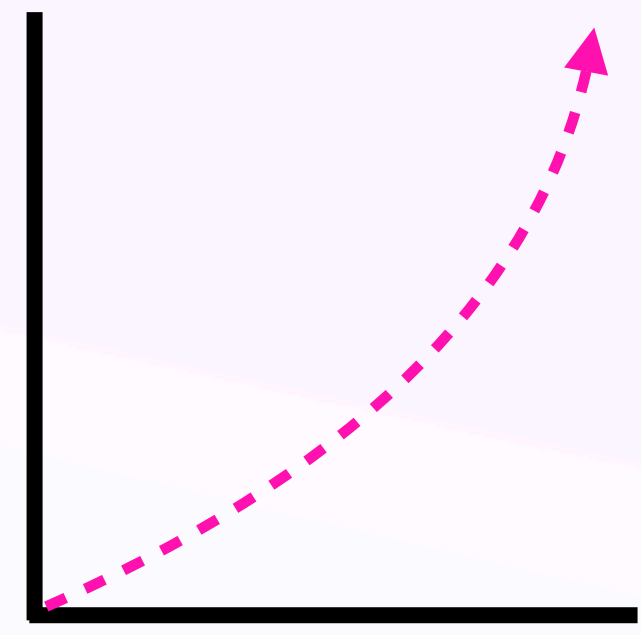


# Self-Healing Concurrent Group Encryption

## *Continuous Group Key Agreement*



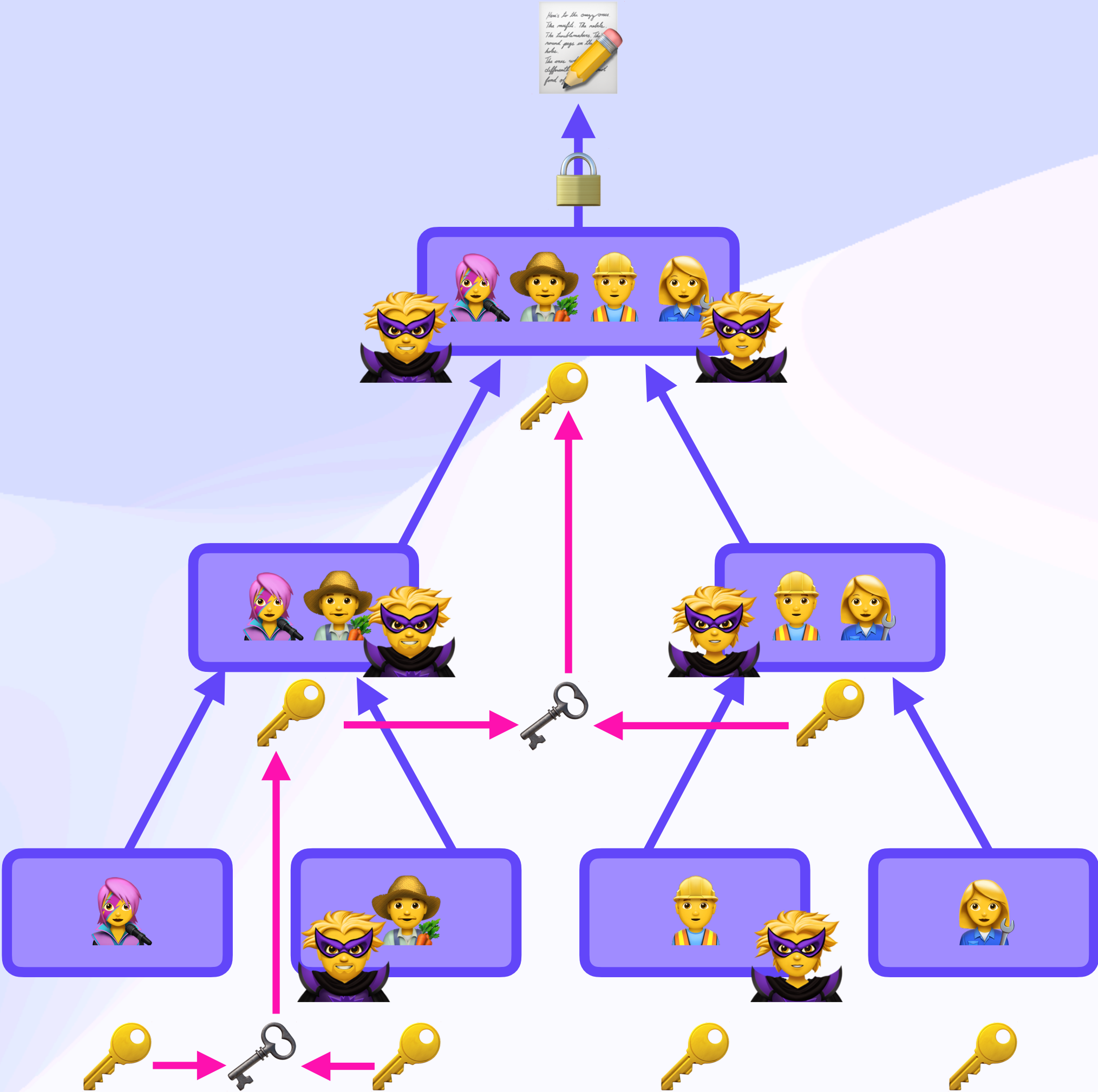
Diffie Hellman



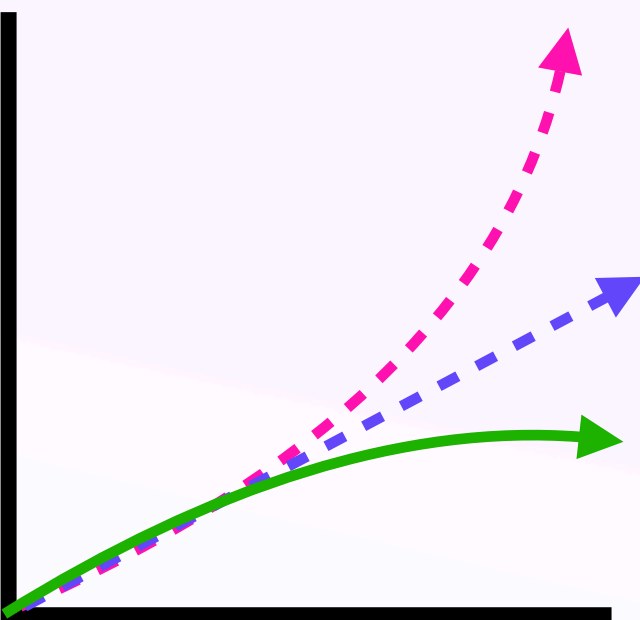


# Self-Healing Concurrent Group Encryption

## *Continuous Group Key Agreement*

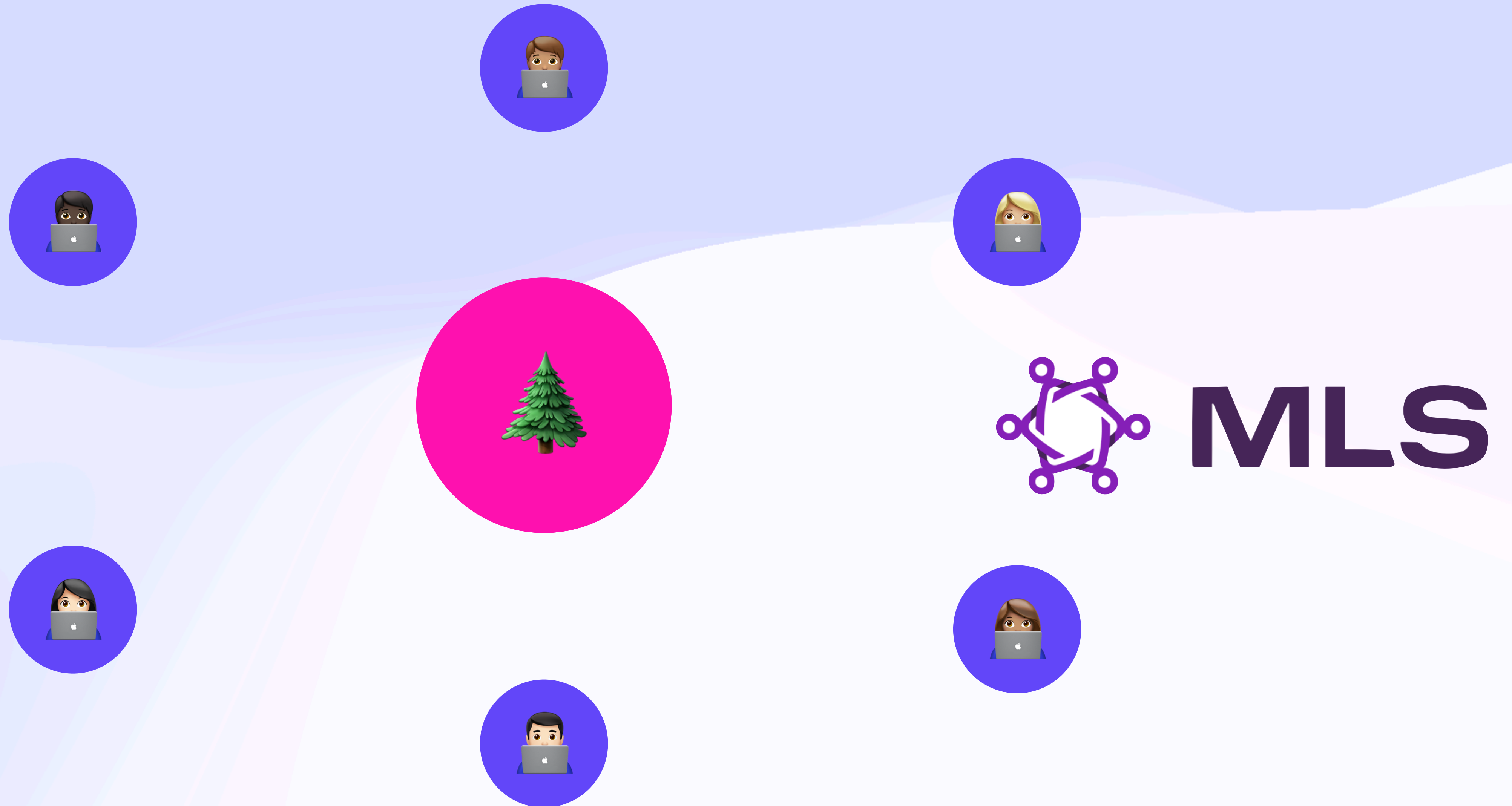


Diffie Hellman



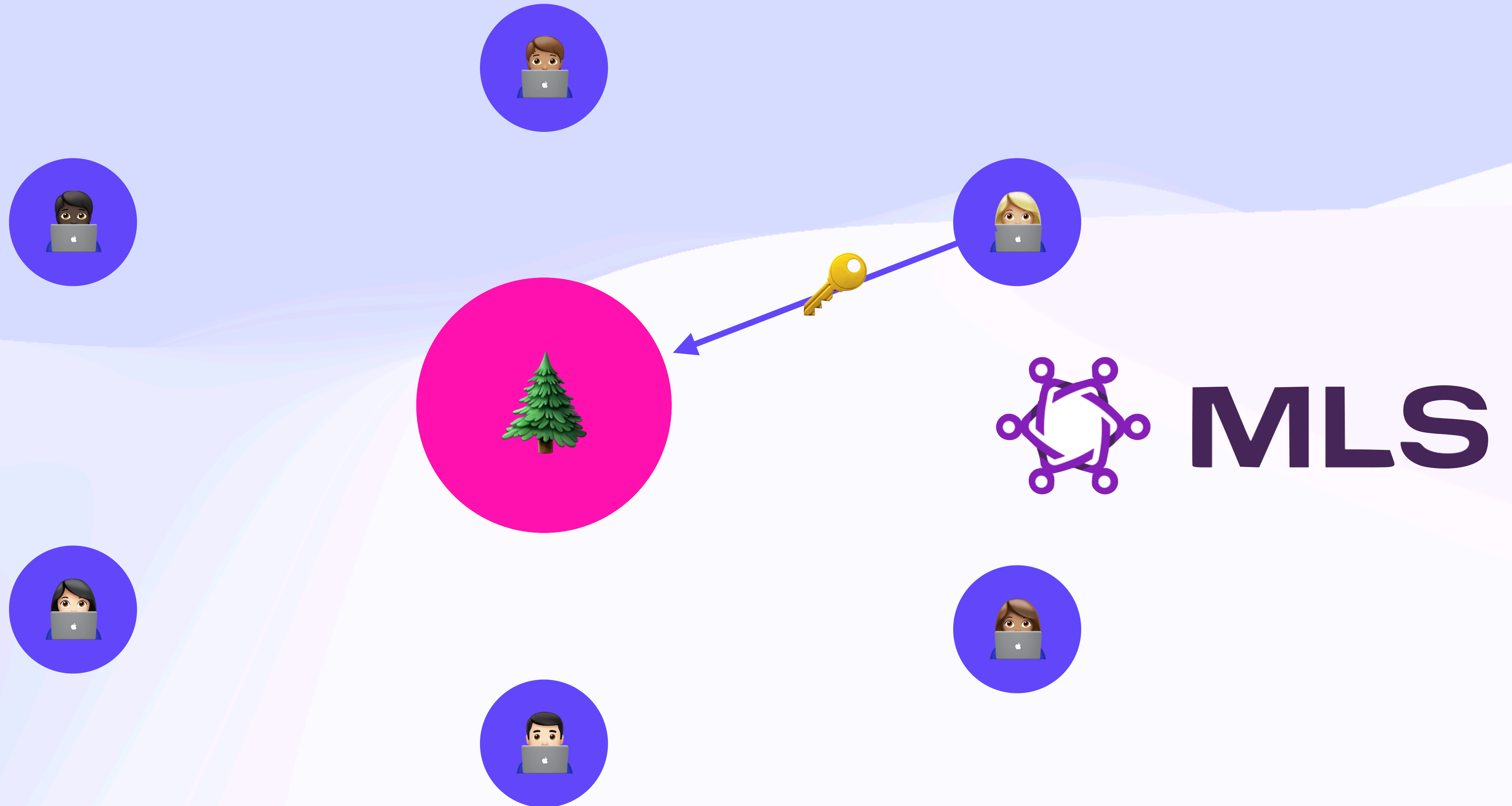
# Self-Healing Concurrent Group Encryption

***MLS: Forced To Use A Centralised Server***



# Self-Healing Concurrent Group Encryption

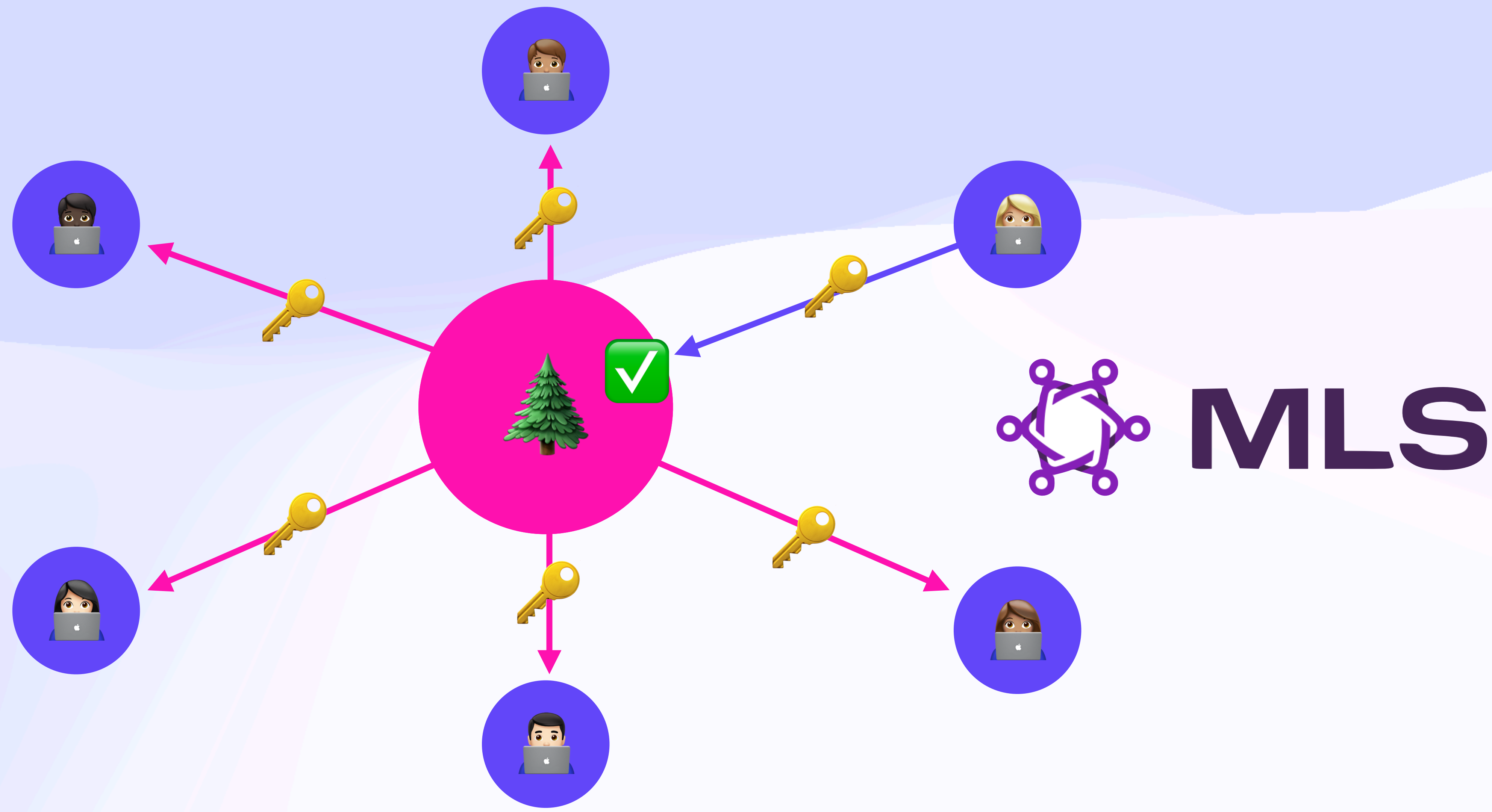
***MLS: Forced To Use A Centralised Server***





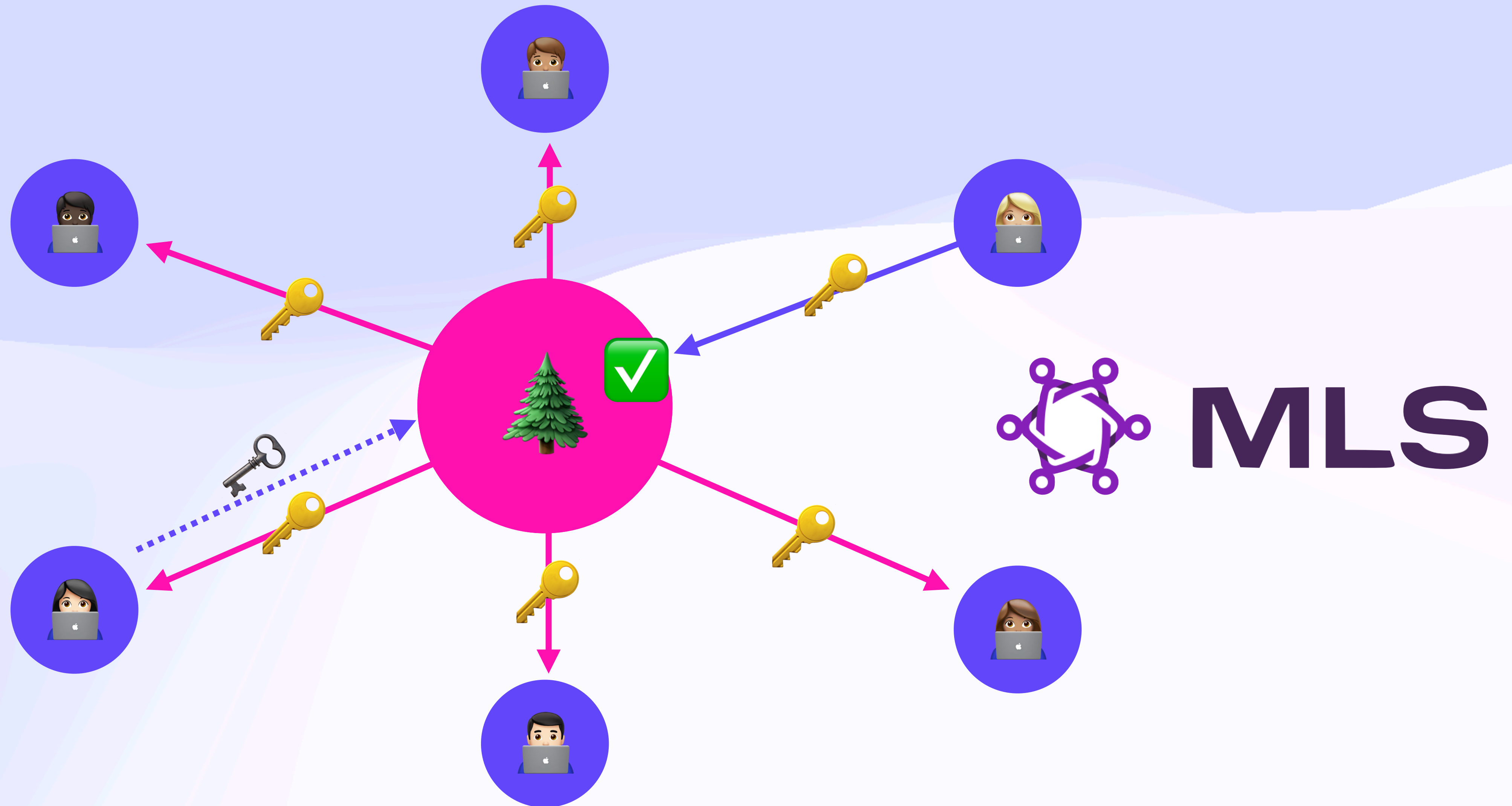
# Self-Healing Concurrent Group Encryption

***MLS: Forced To Use A Centralised Server***



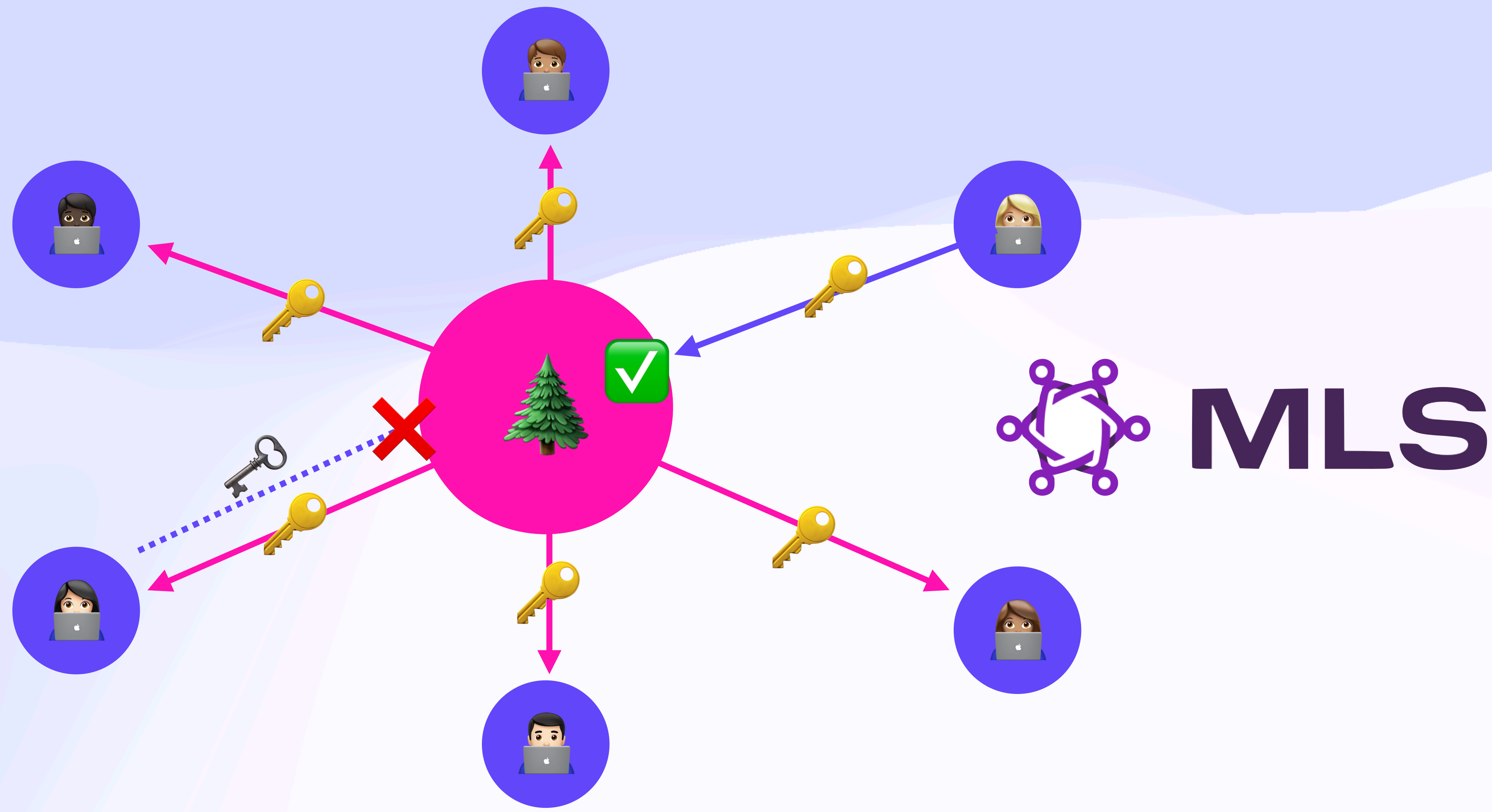
# Self-Healing Concurrent Group Encryption

***MLS: Forced To Use A Centralised Server***



# Self-Healing Concurrent Group Encryption

***MLS: Forced To Use A Centralised Server***

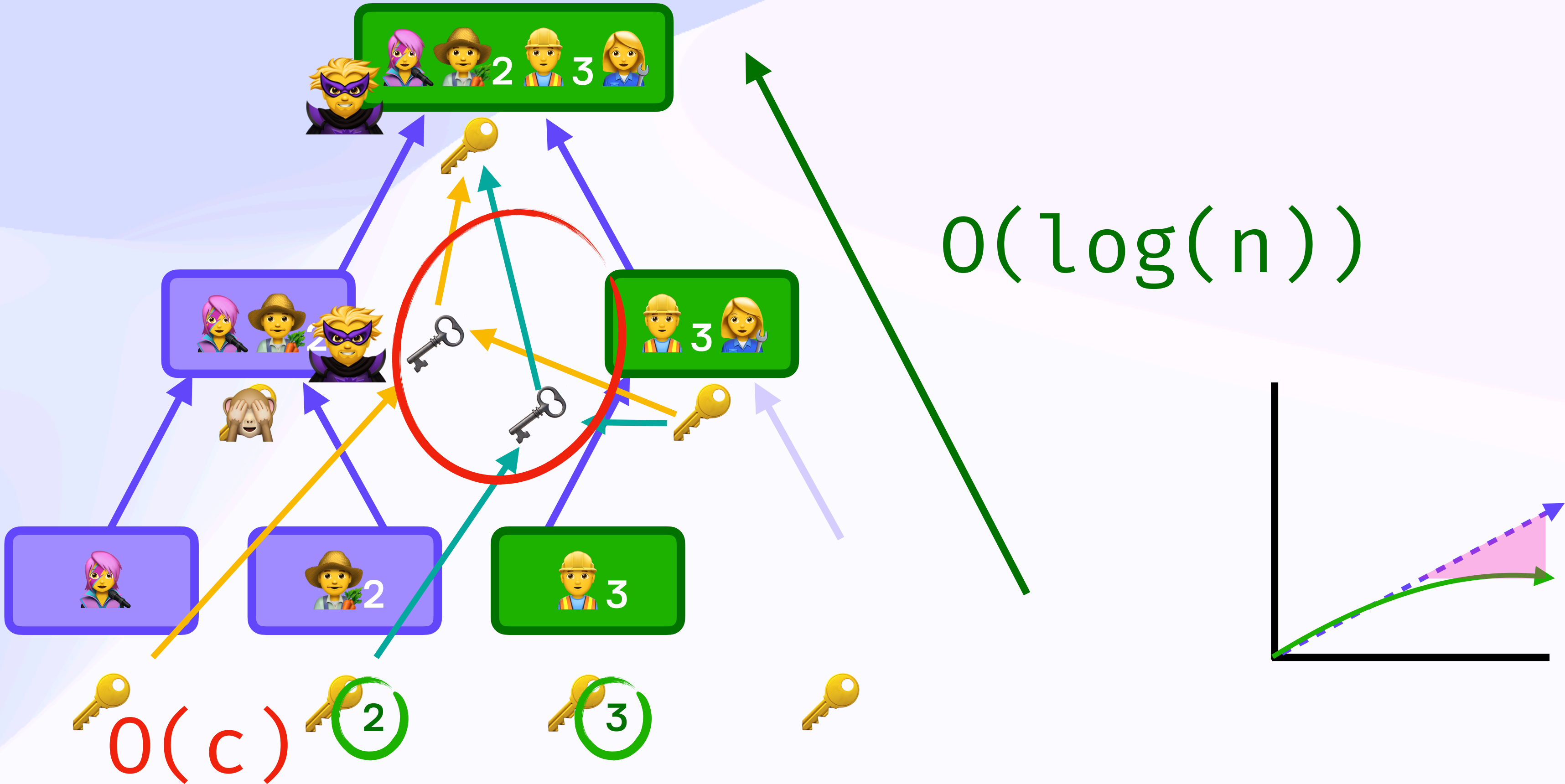




# Self-Healing Concurrent Group Encryption



Scales in approximately  
 $O(c + \log(n))$



# ***Wrap Up***



Wrap Up

***Wrap Up***




# Wrap Up


## *Wrap Up*

- ♦ Mutation Control
  - ♦ Convergent capabilities
  - ♦ Concurrent revocation
- ♦ Read Control
  - ♦ Causal encryption
  - ♦ Continuous group key agreement (CGKA)

People with access


 hello@katiewilde.com  
hello@katiewilde.com

Owner


 Brooklyn Zelenka (you)  
hello@brooklynzelenka.com

Editor ▼

General access

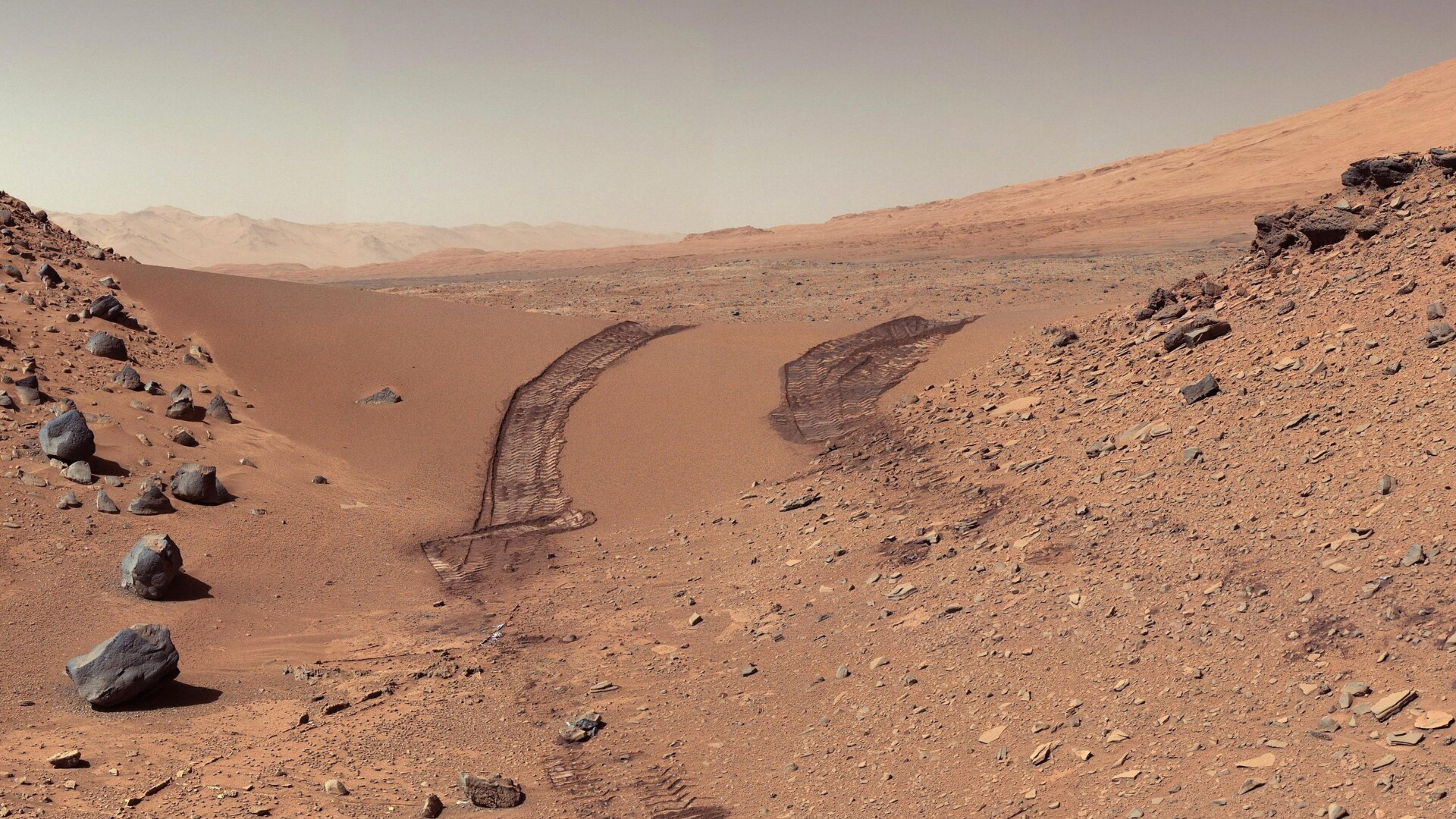
 Restricted ▼

Only people with access can open with the link

 Copy link

Done







 @expede.wtf

 @expede@types.pl

 notes.brooklynzelenka.com



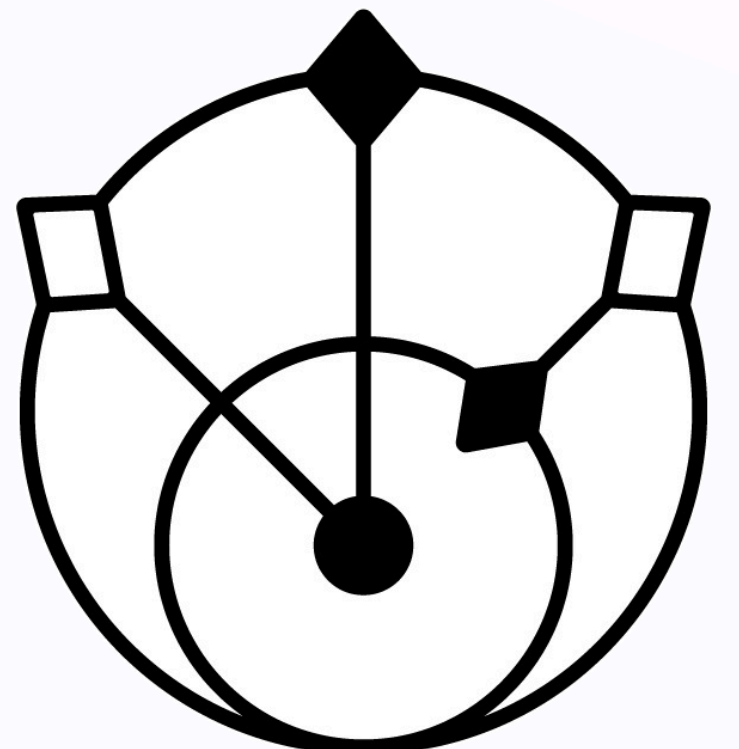
***Thank You, DWWeb Seattle***



github.com/ucan-wg



inkandswitch.com/keyhive





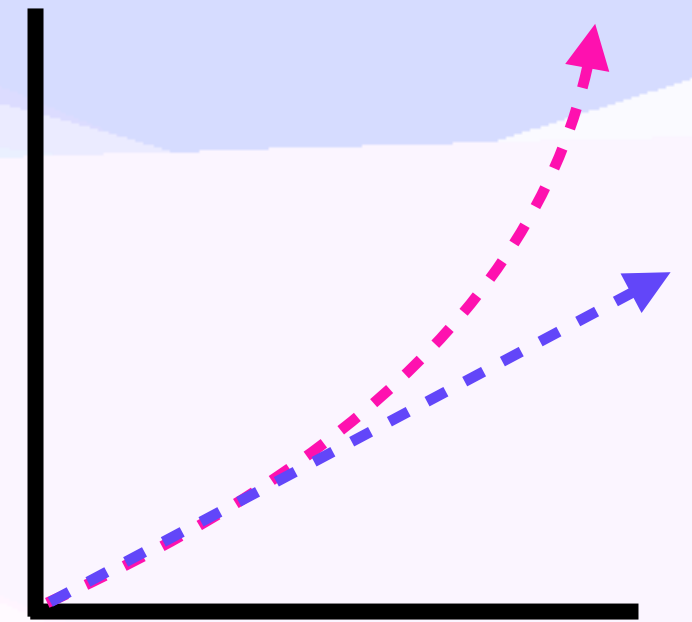
# Self-Healing Concurrent Encryption for Groups

## *BeeKEM*



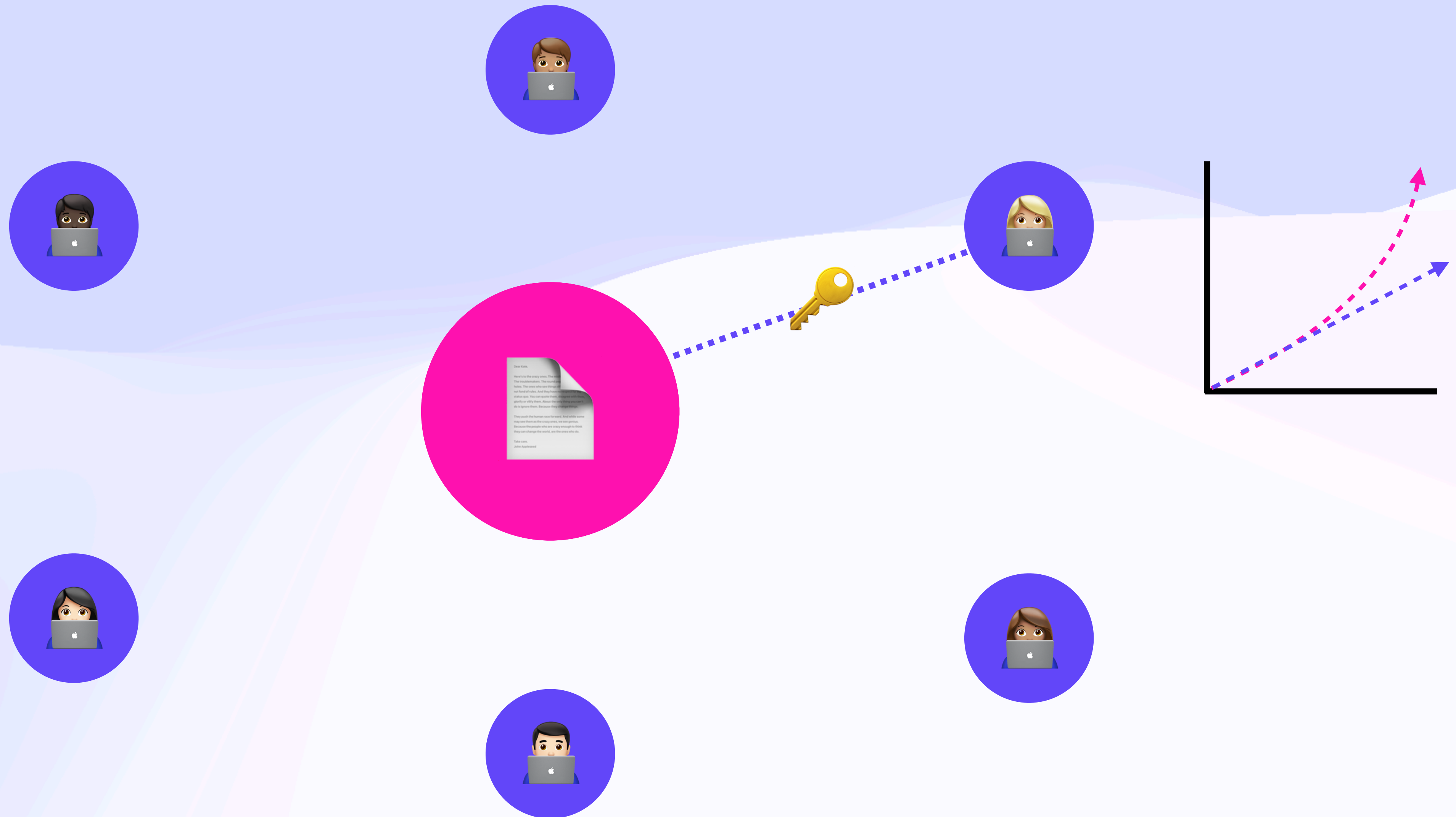
# Self-Healing Concurrent Group Encryption

# Thinking In Groups Already Better



# Self-Healing Concurrent Group Encryption

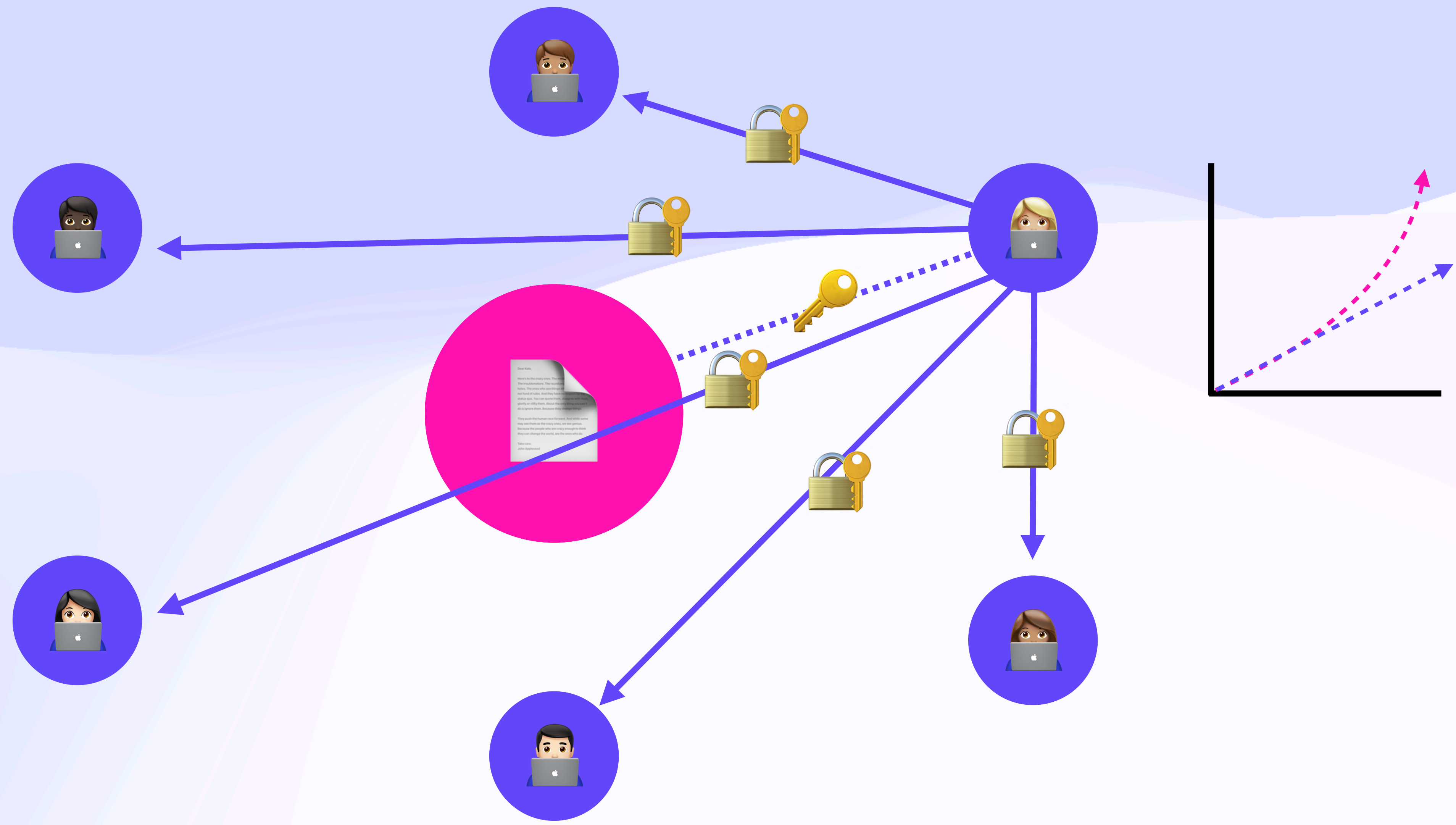
# Thinking In Groups Already Better





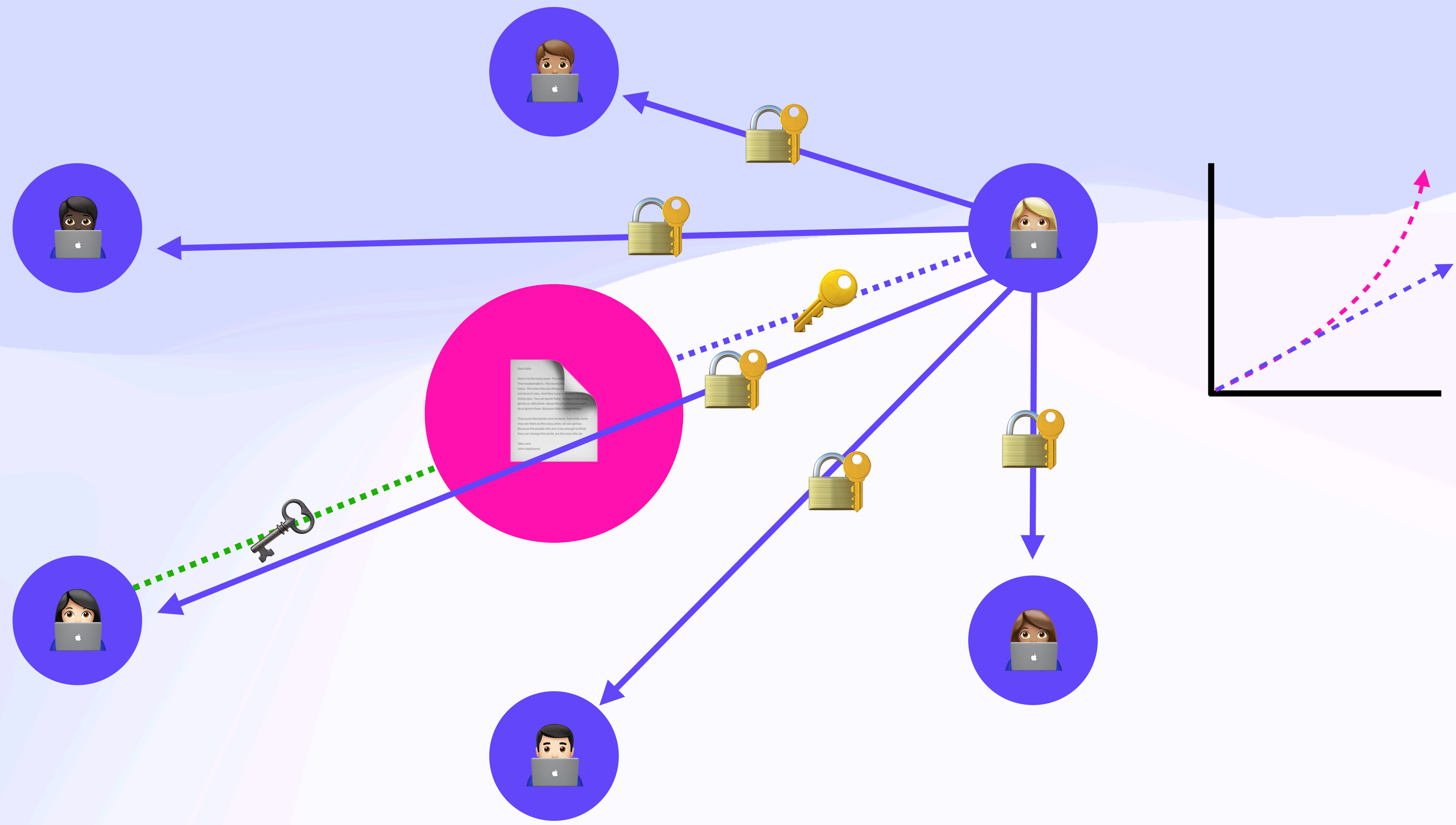
# Self-Healing Concurrent Group Encryption

## *Thinking In Groups Already Better*



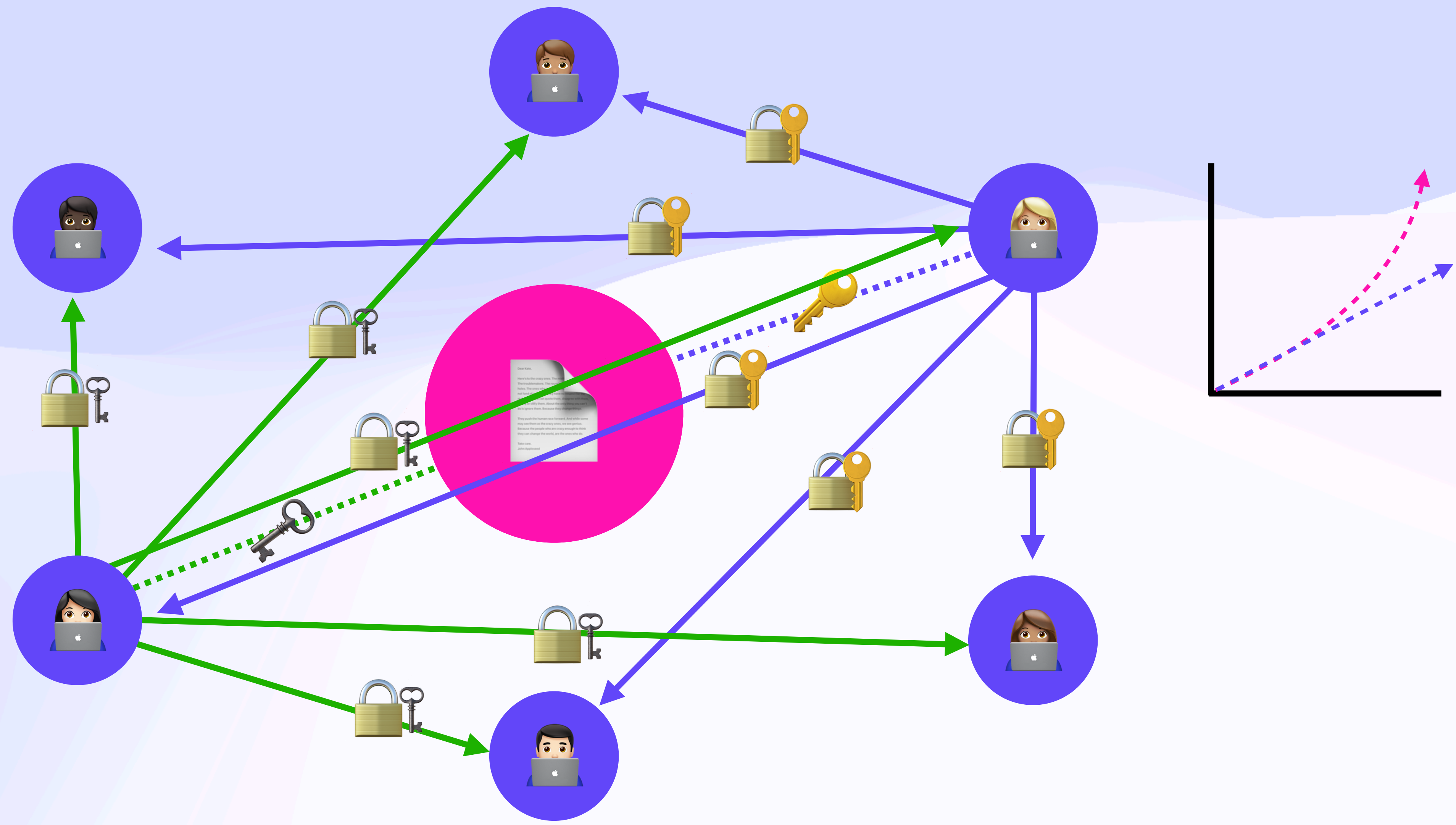
# Self-Healing Concurrent Group Encryption

## *Thinking In Groups Already Better*



# Self-Healing Concurrent Group Encryption

## *Thinking In Groups Already Better*



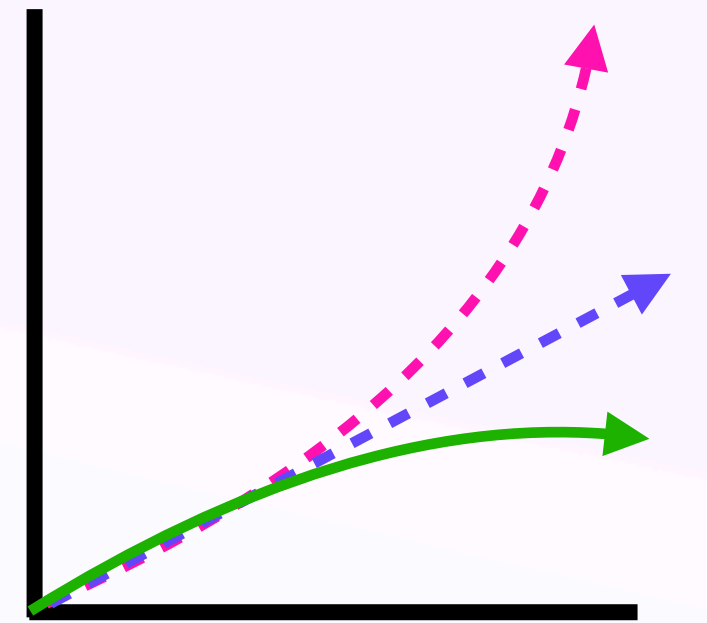


# Self-Healing Concurrent Group Encryption

## *TreeKEM*



**MLS**

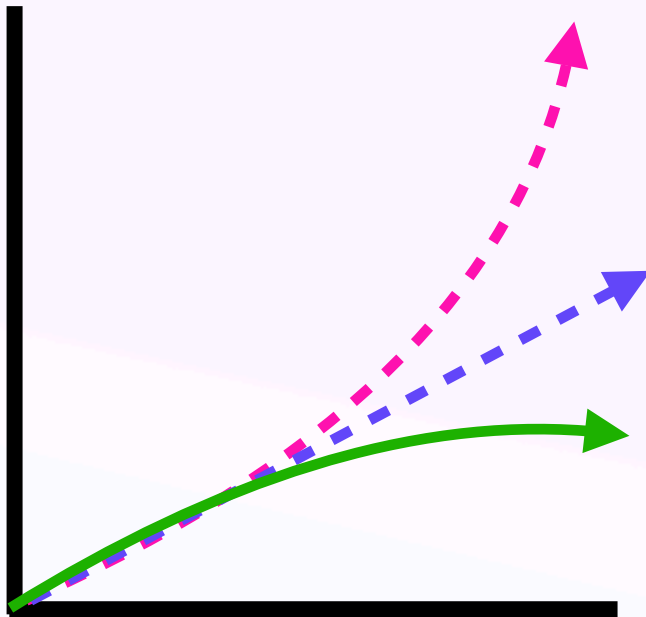
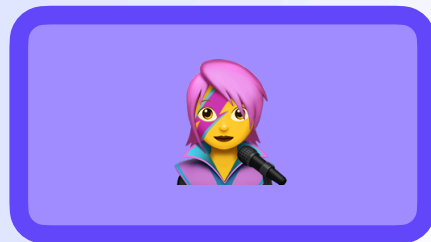


# Self-Healing Concurrent Group Encryption

*TreeKEM*



MLS

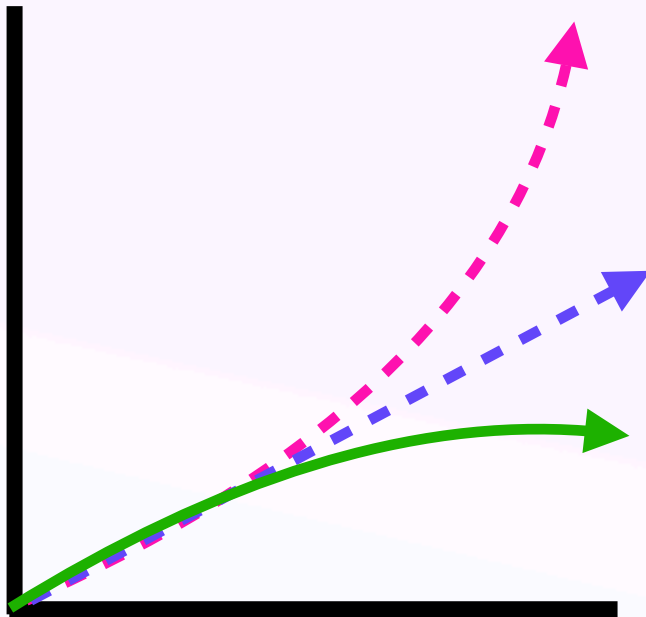
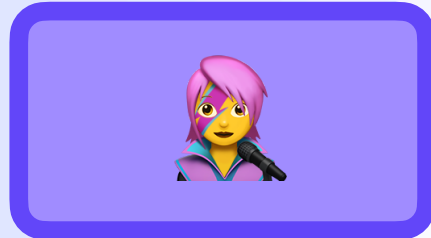


# Self-Healing Concurrent Group Encryption

*TreeKEM*



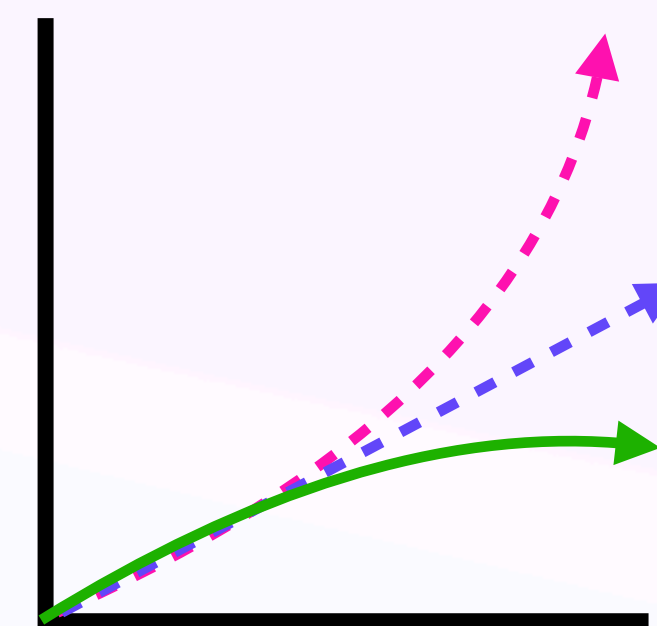
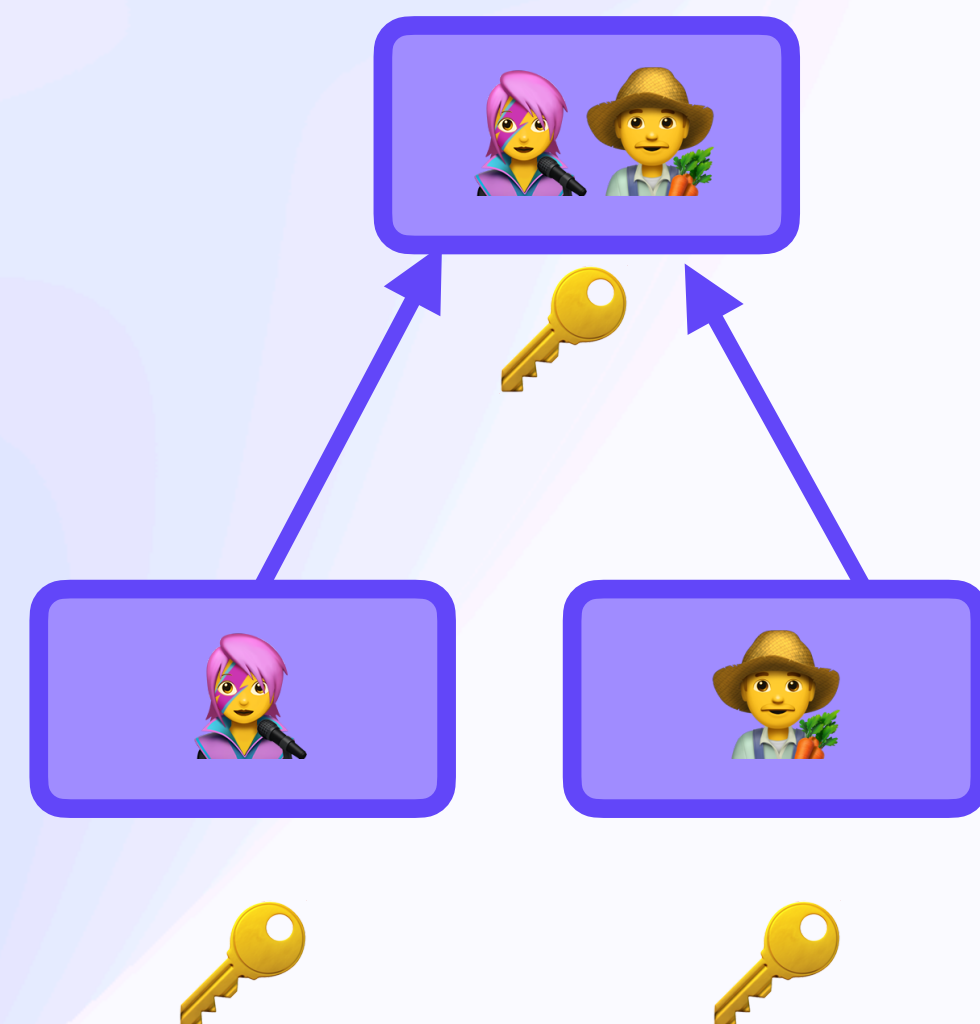
MLS





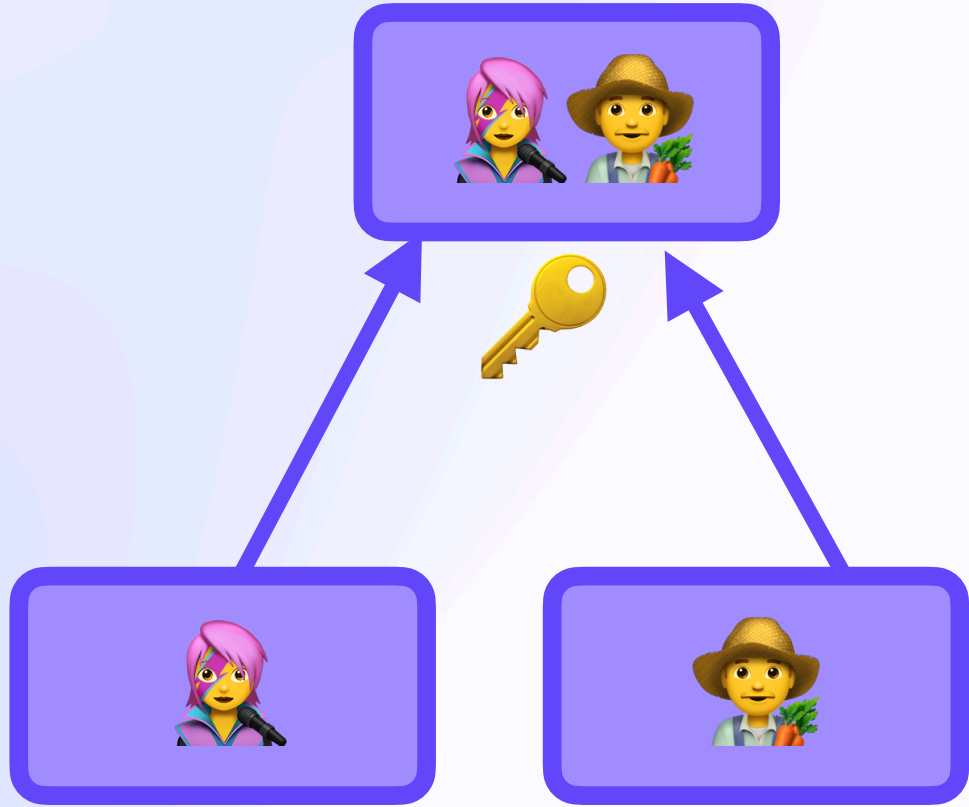
# Self-Healing Concurrent Group Encryption

## *TreeKEM*

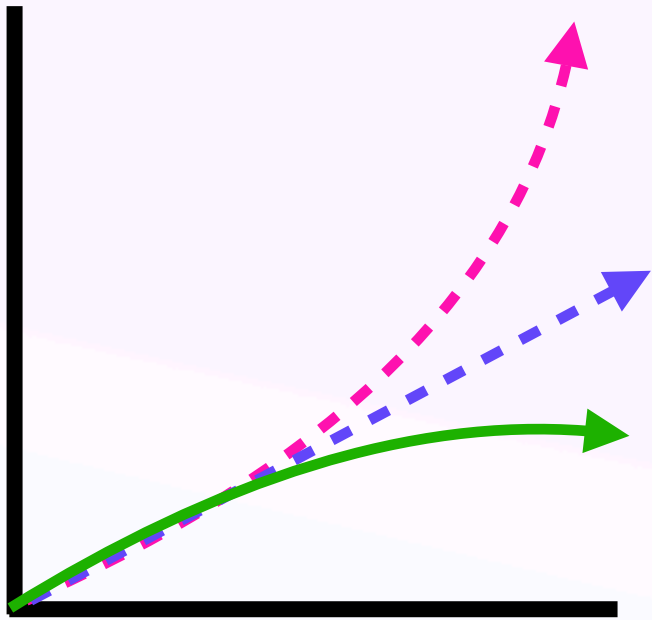


# Self-Healing Concurrent Group Encryption

# TreeKEM

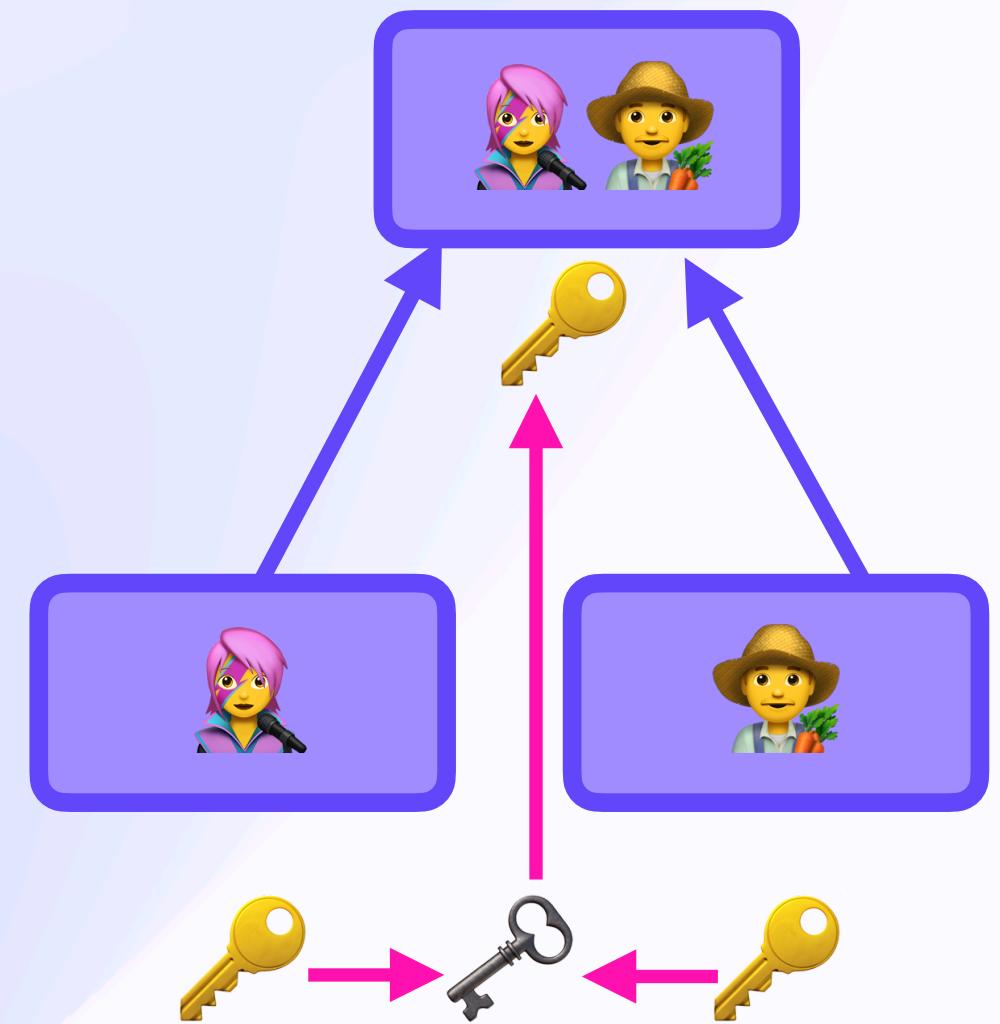


Diffie Hellman

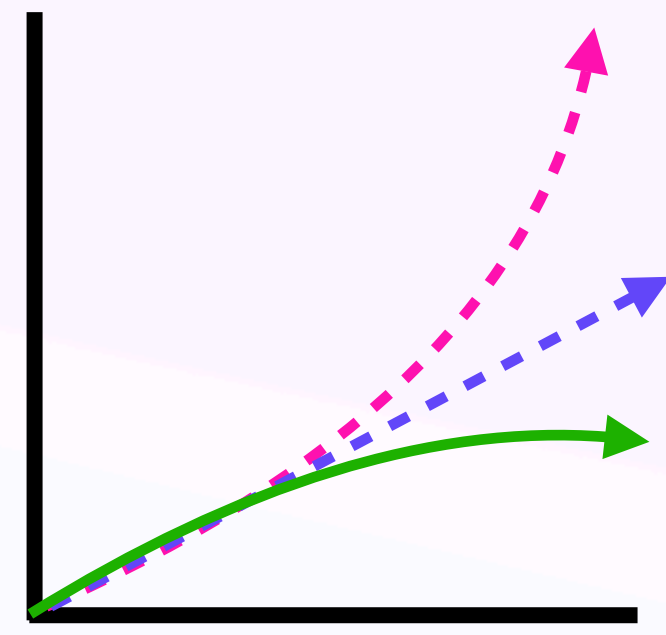


# Self-Healing Concurrent Group Encryption

## TreeKEM



Diffie Hellman

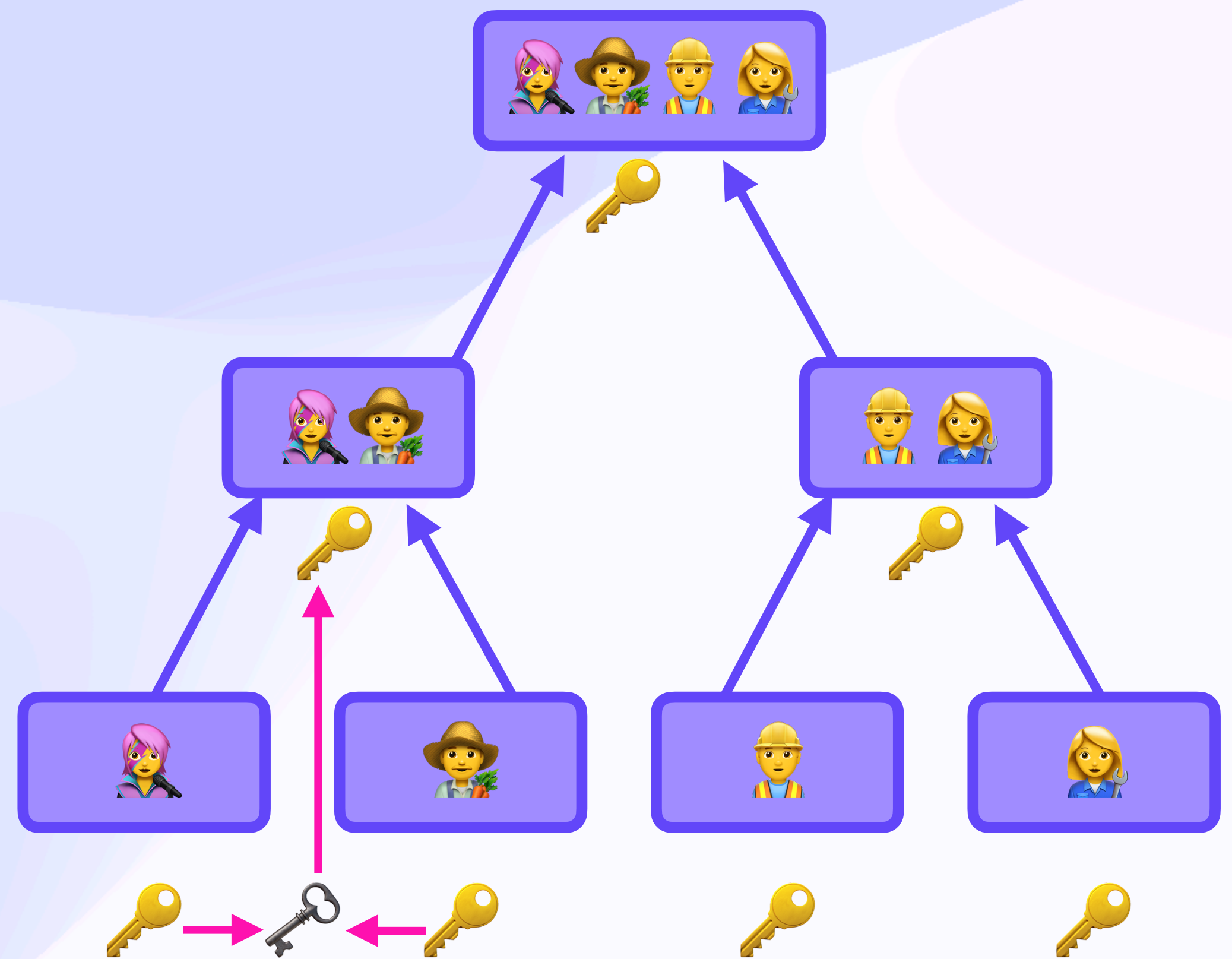




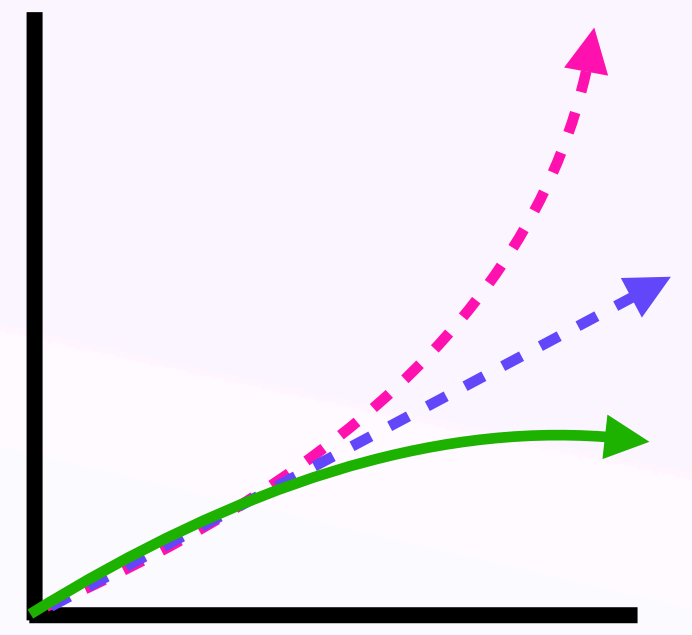
# Self-Healing Concurrent Group Encryption

## TreeKEM

 **MLS**



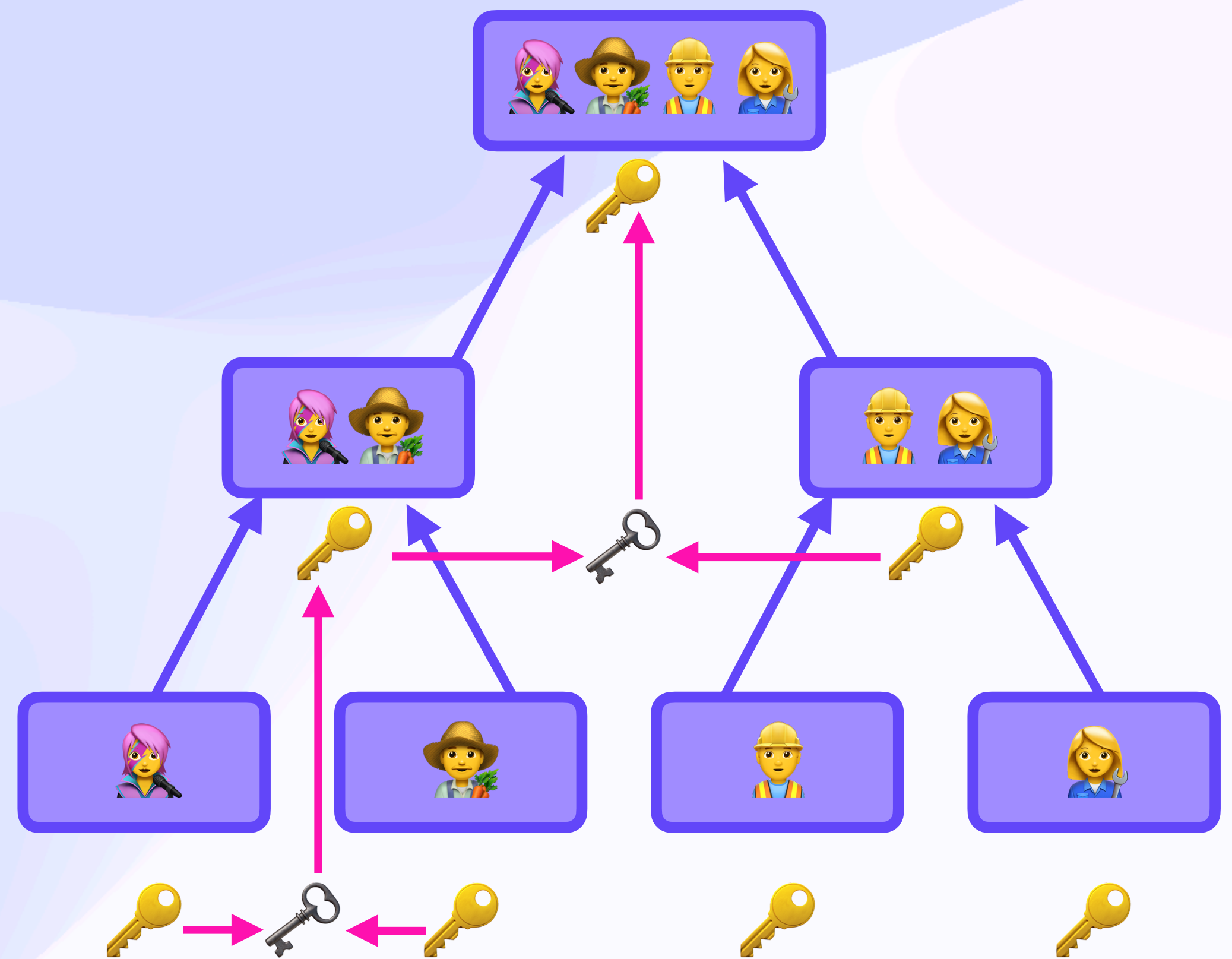
Diffie Hellman



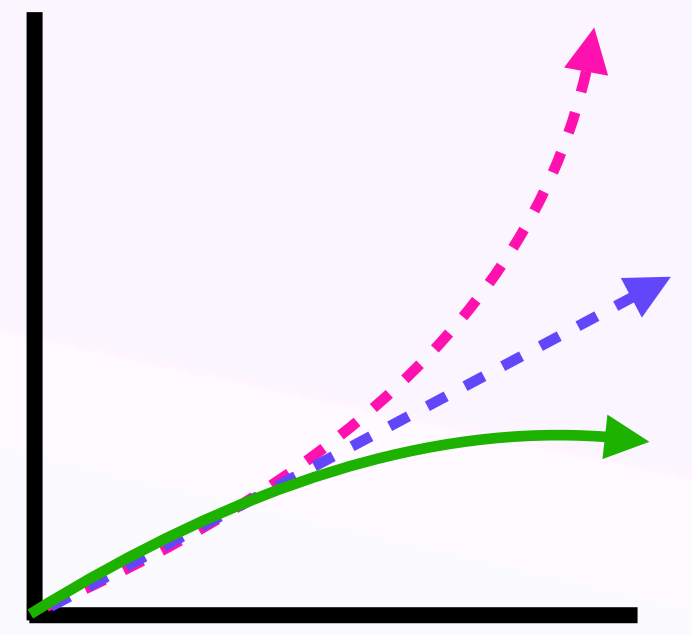
# Self-Healing Concurrent Group Encryption

## TreeKEM

 **MLS**

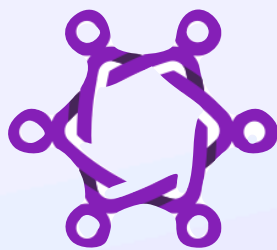


Diffie Hellman

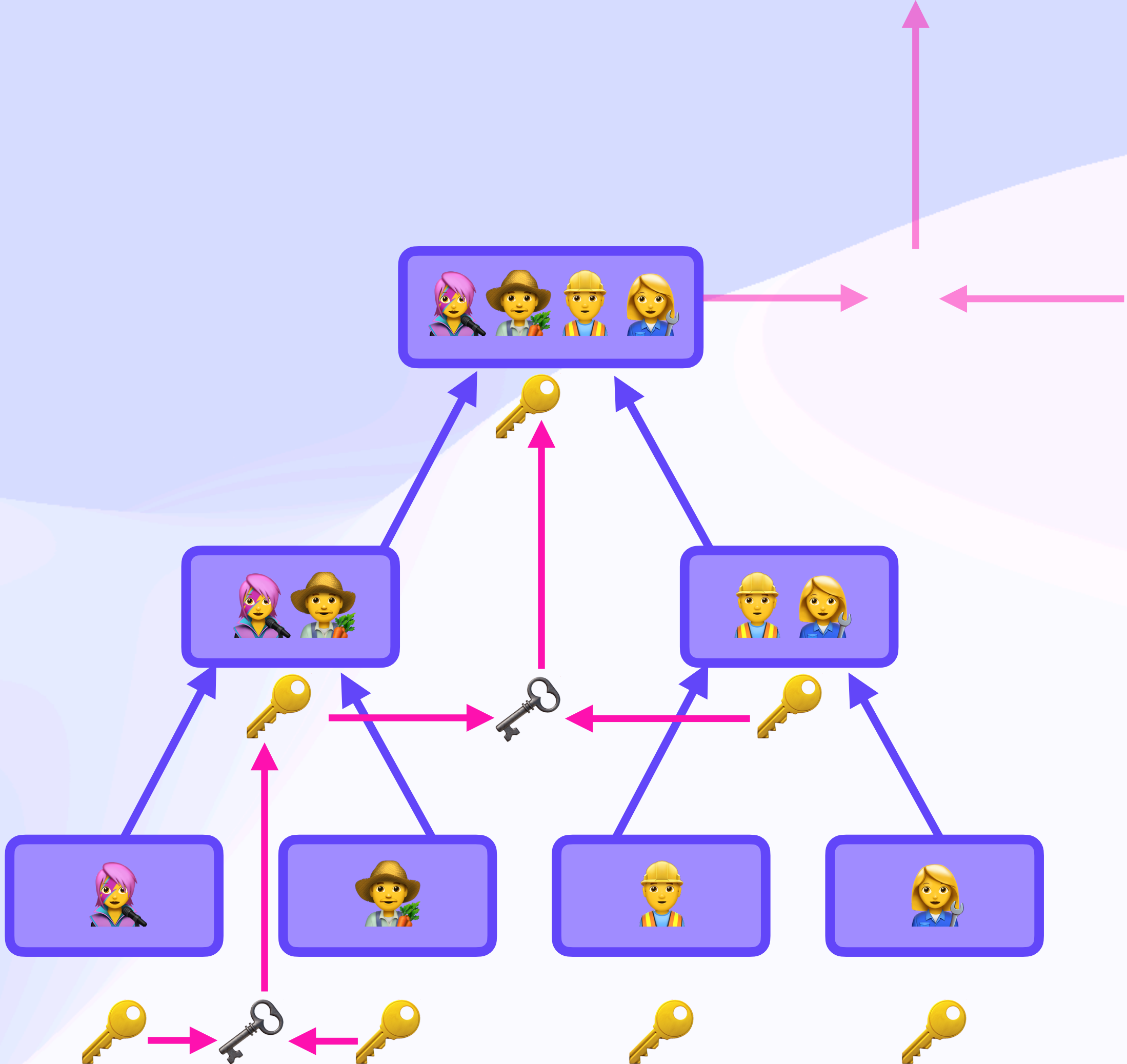


# Self-Healing Concurrent Group Encryption

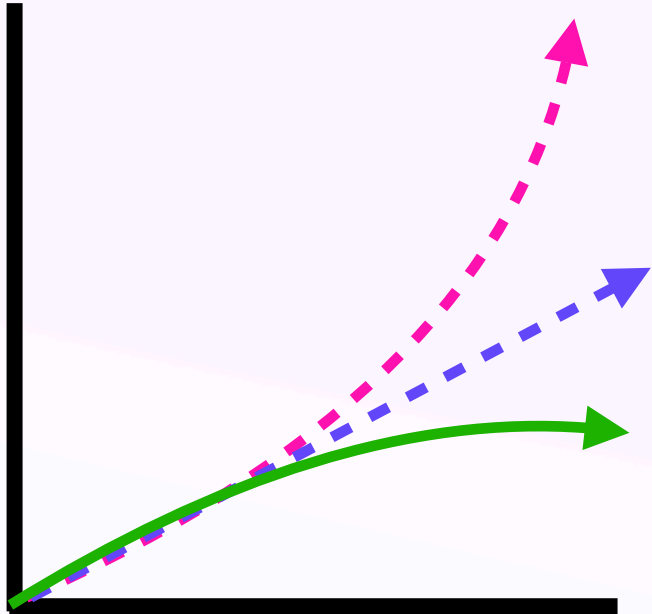
# TreeKEM



MLS



Diffie Hellman

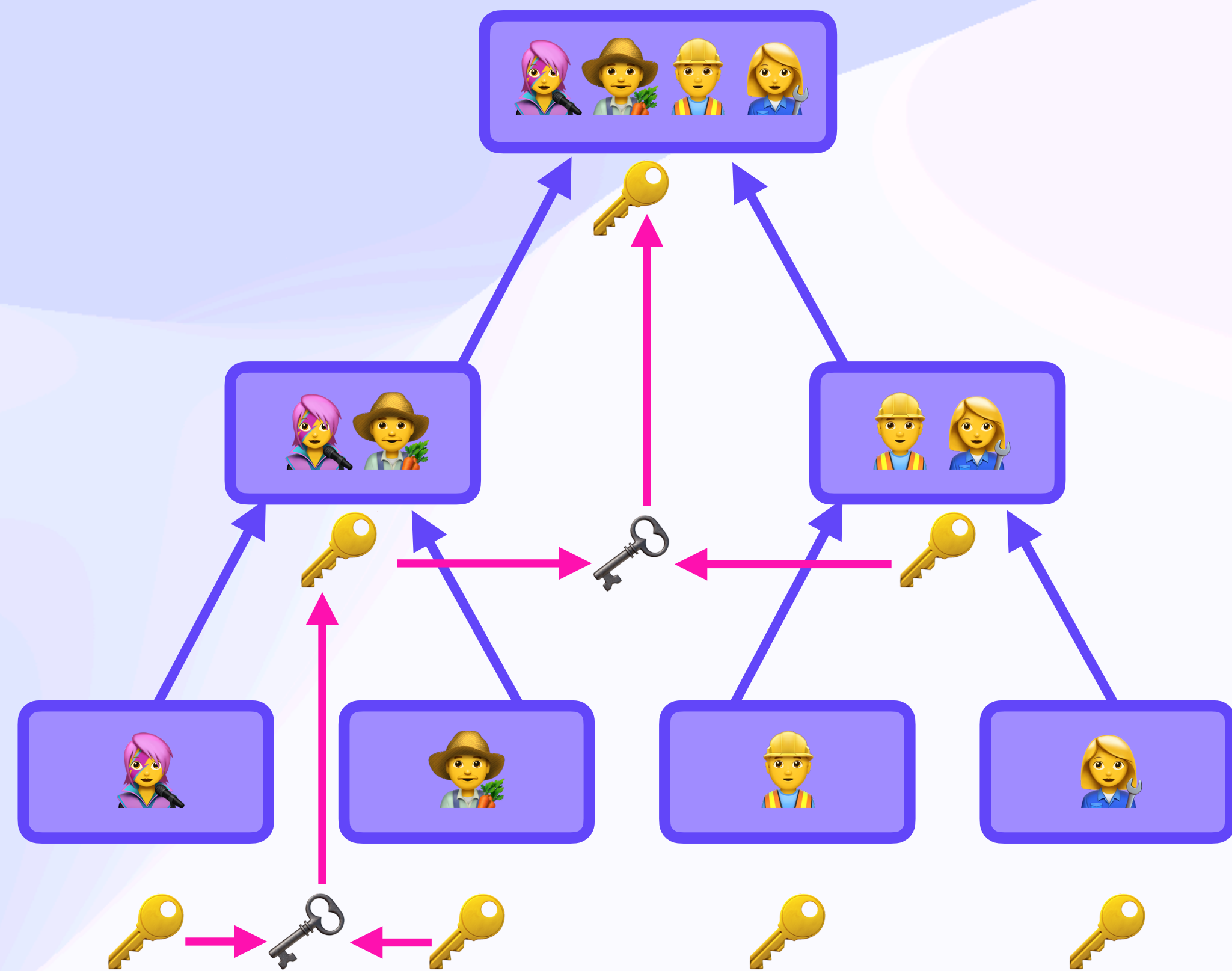




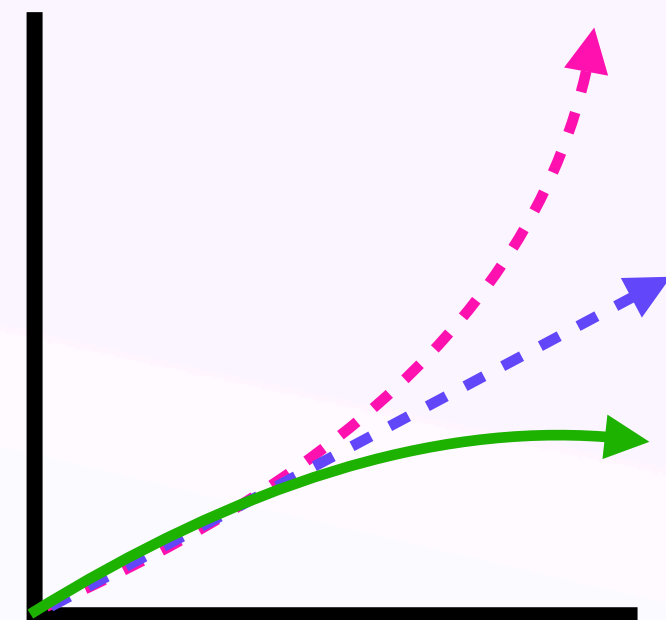
# Self-Healing Concurrent Group Encryption

## TreeKEM

 **MLS**

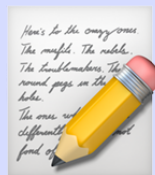


Diffie Hellman

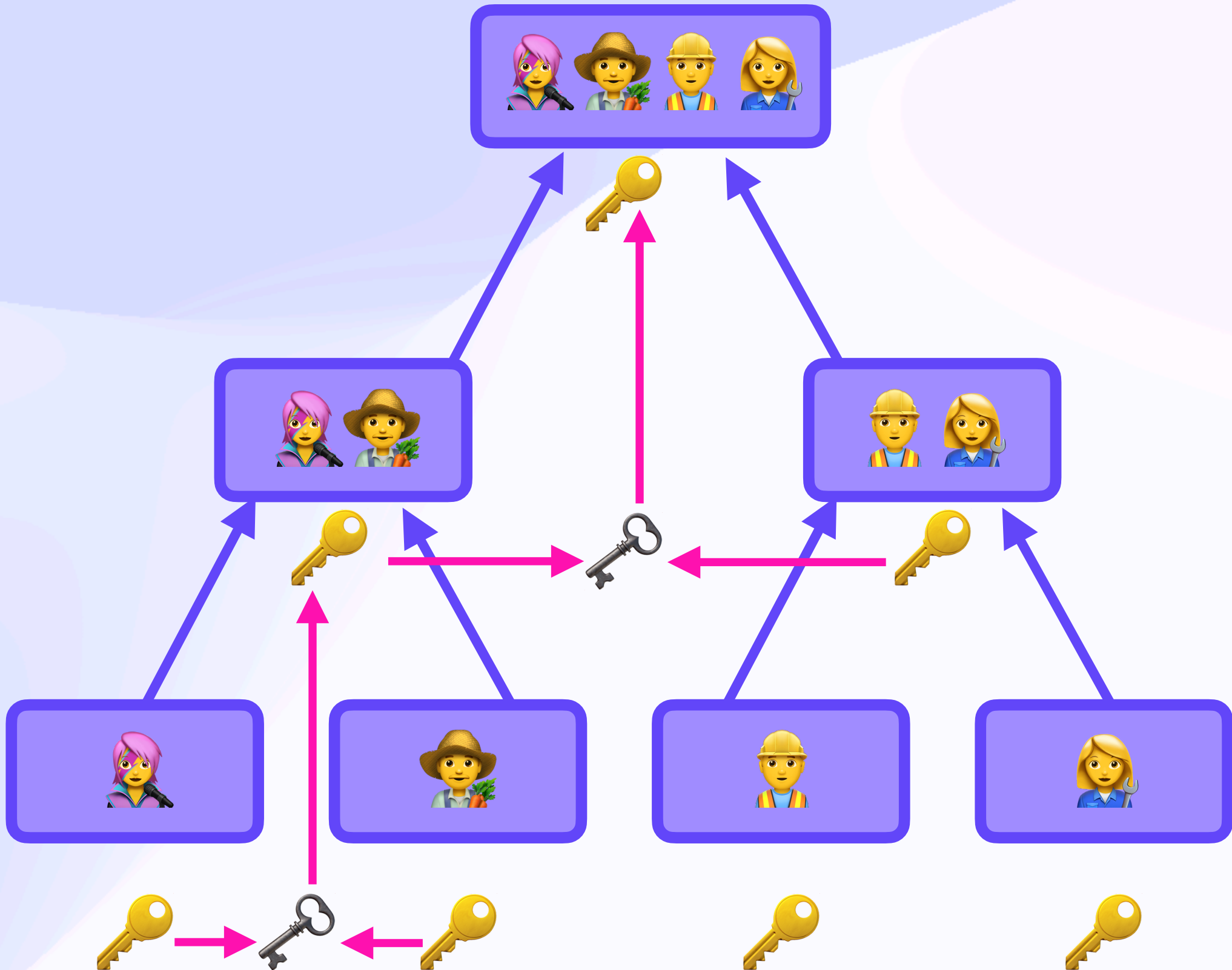


# Self-Healing Concurrent Group Encryption

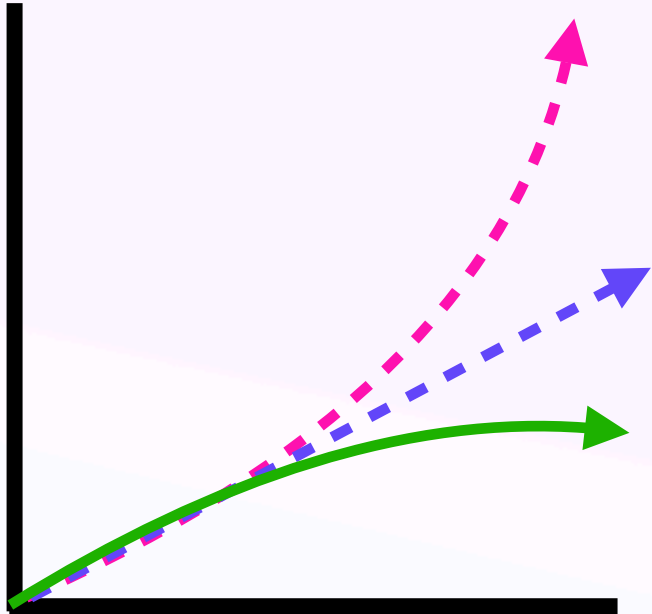
# TreeKEM



 **MLS**



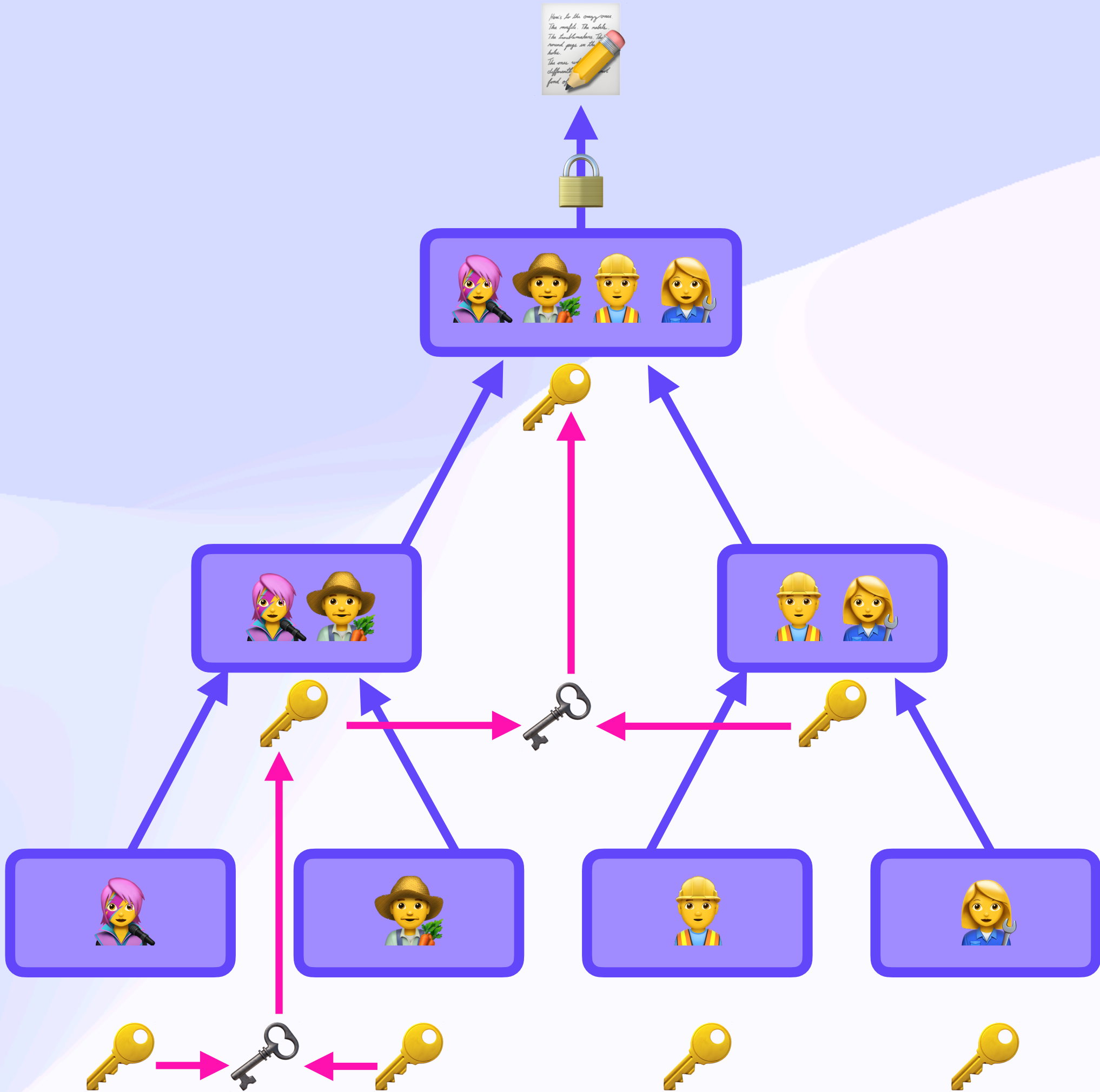
Diffie Hellman



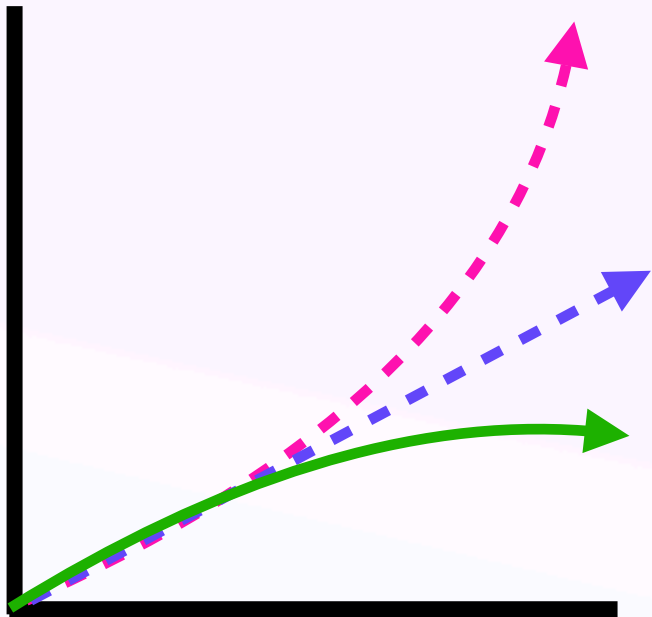
# Self-Healing Concurrent Group Encryption

# TreeKEM

 **MLS**



Diffie Hellman

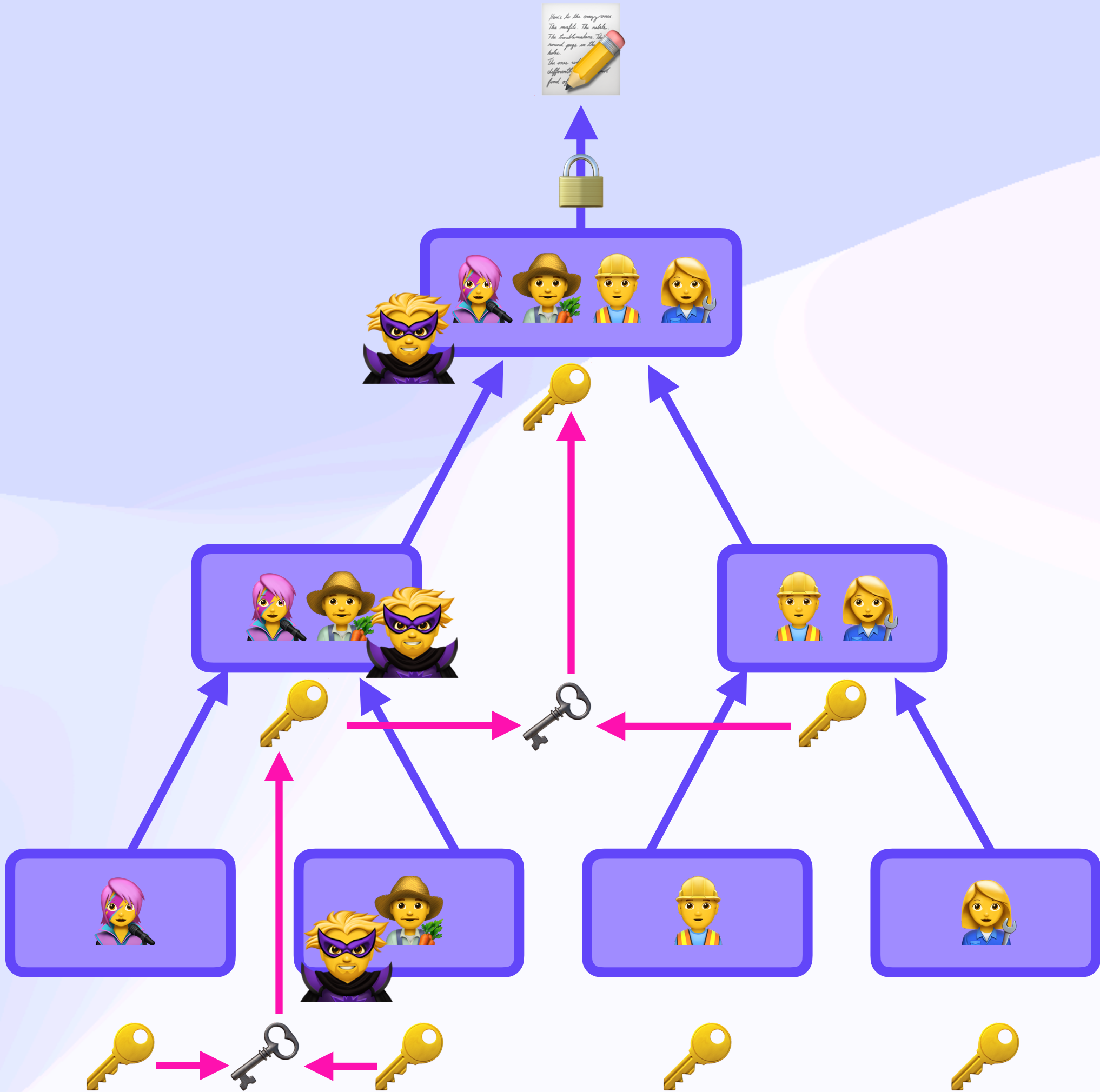




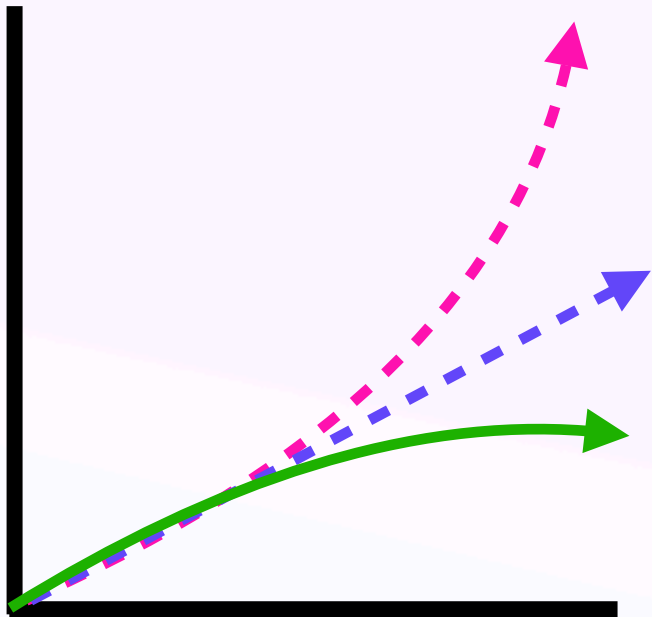
# Self-Healing Concurrent Group Encryption

# TreeKEM

 **MLS**



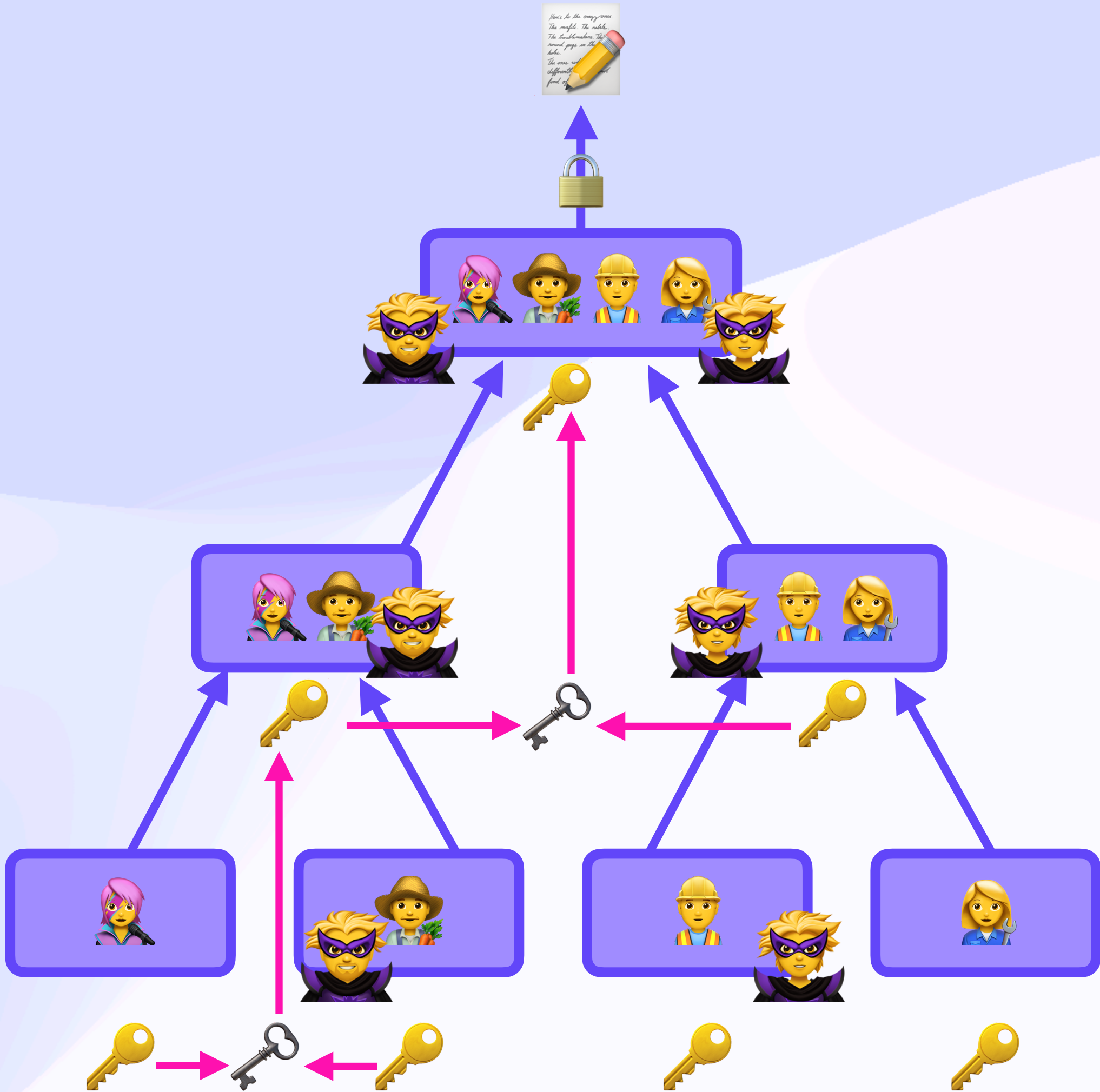
Diffie Hellman



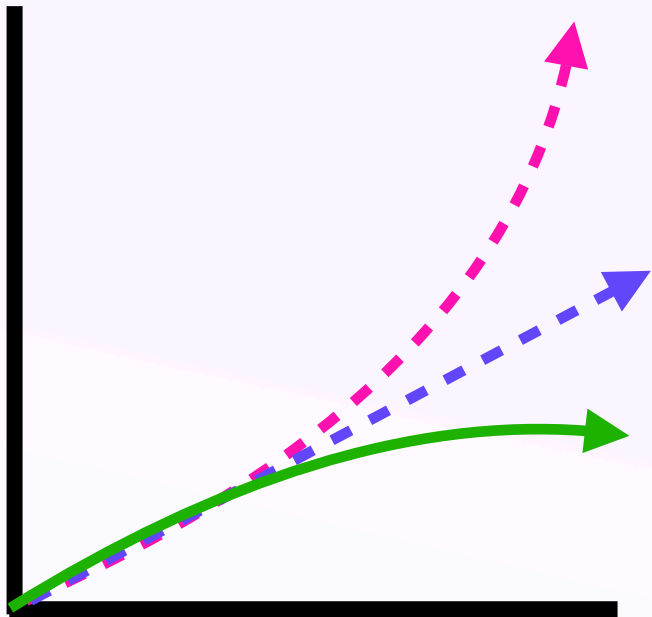
# Self-Healing Concurrent Group Encryption

# TreeKEM

 **MLS**



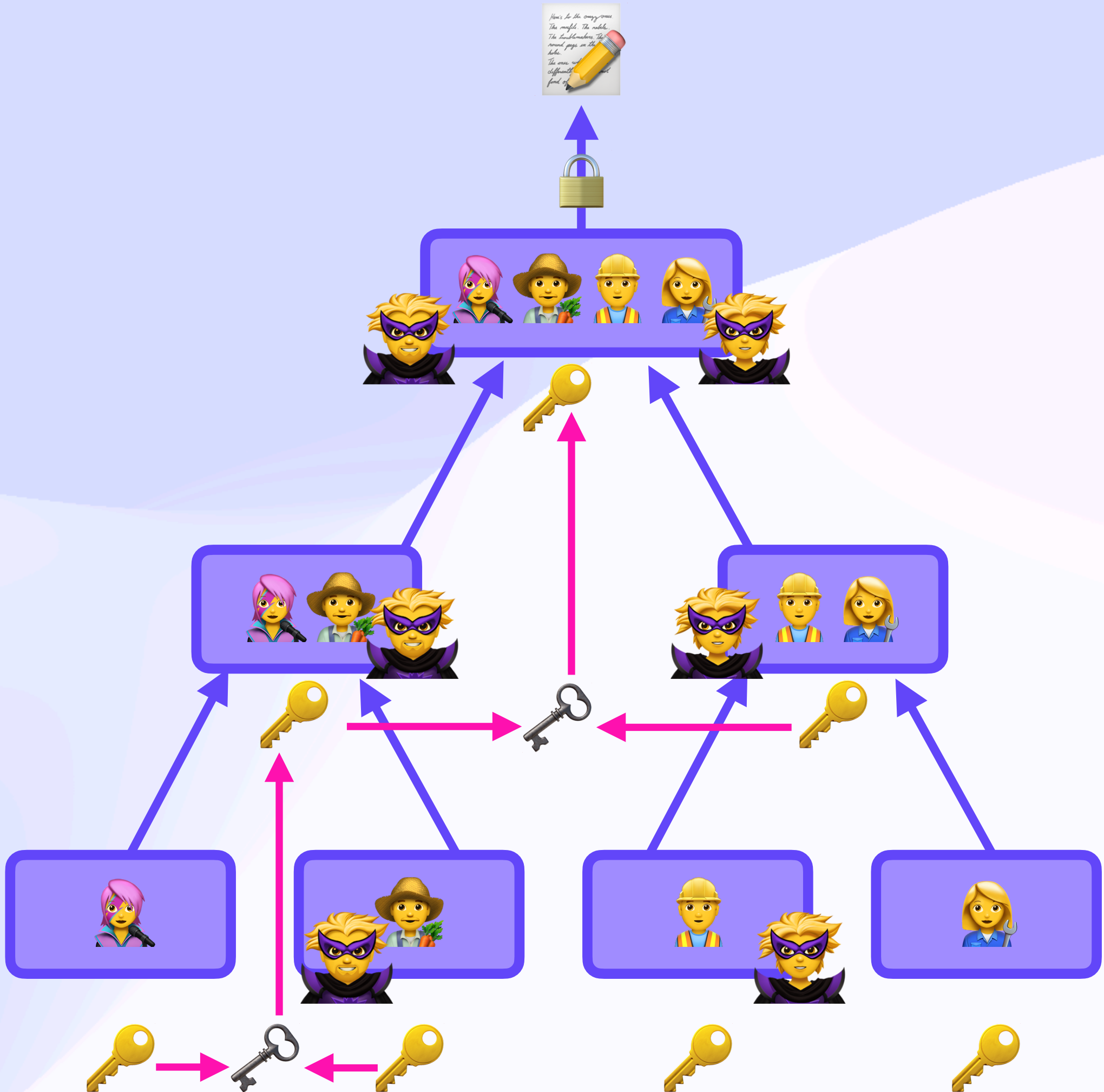
Diffie Hellman



# Self-Healing Concurrent Group Encryption

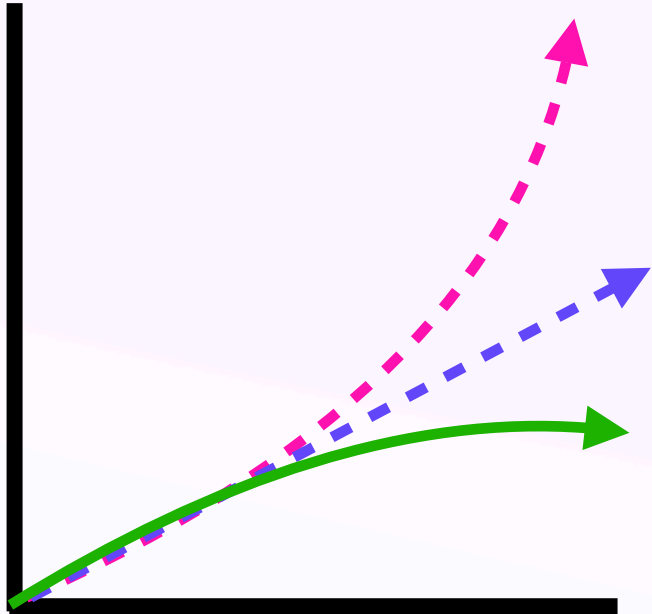
# TreeKEM

 **MLS**



Diffie Hellman

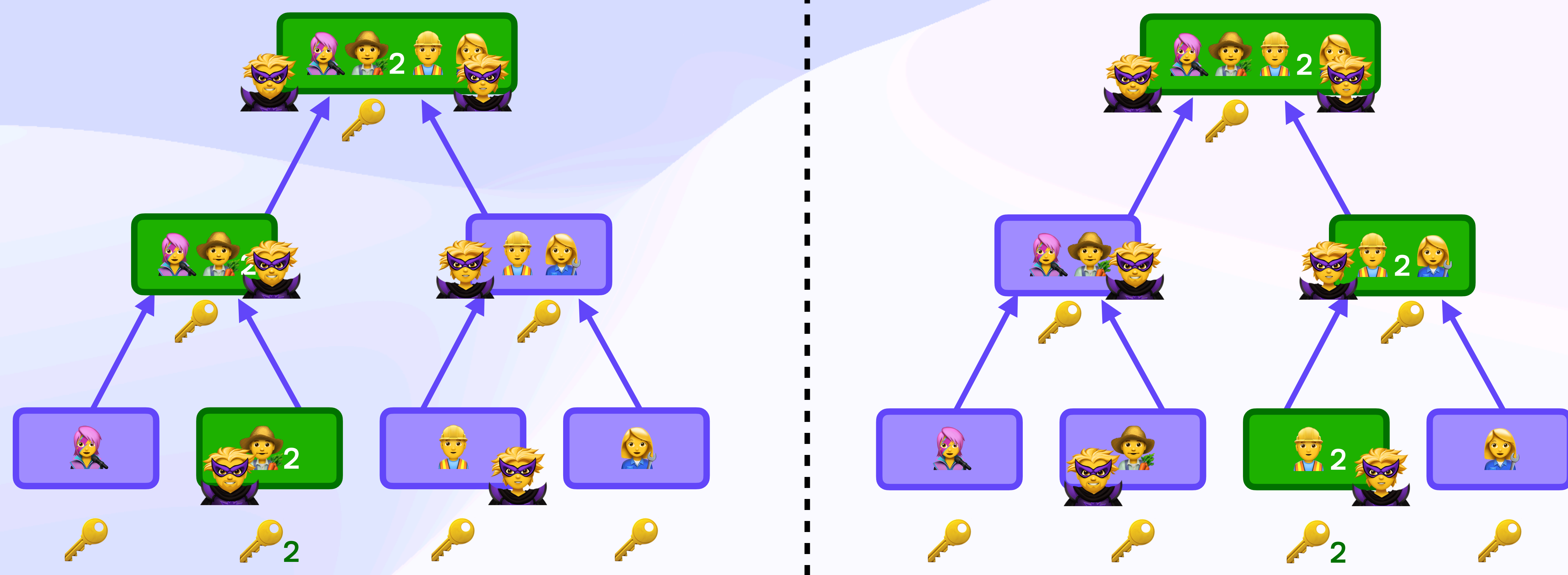
...but...





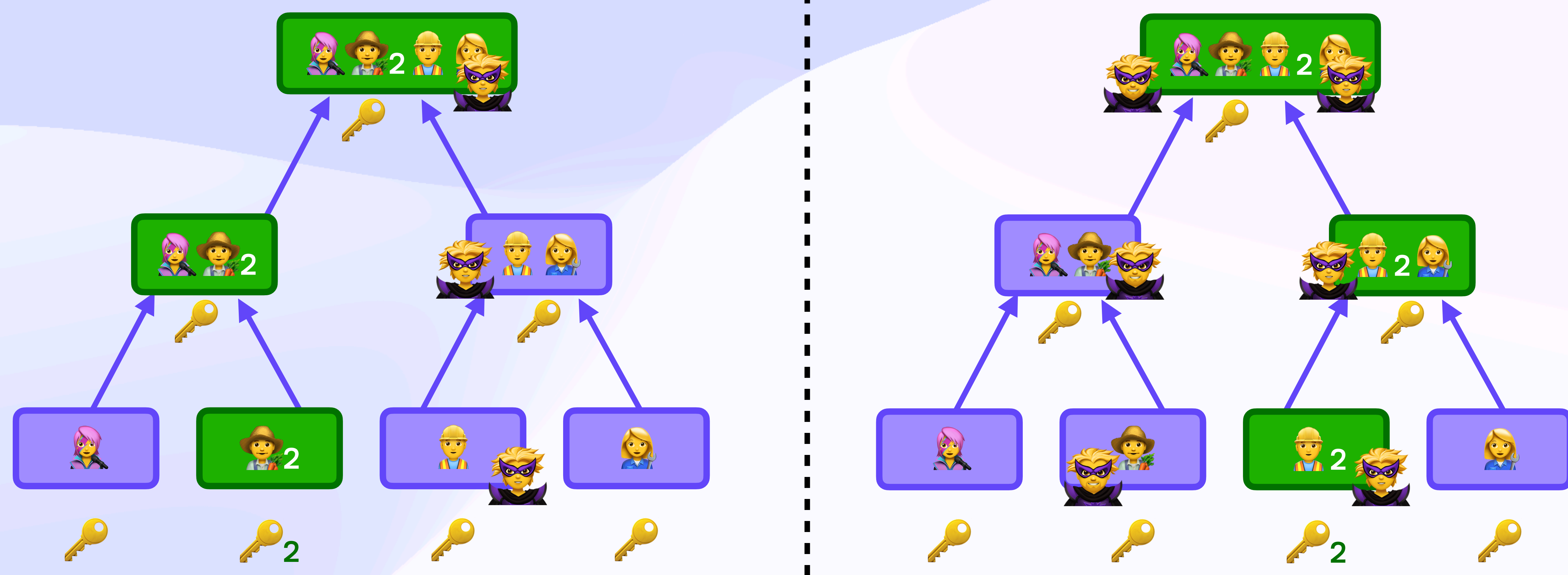
# Self-Healing Concurrent Group Encryption

## *Concurrency Problem!*



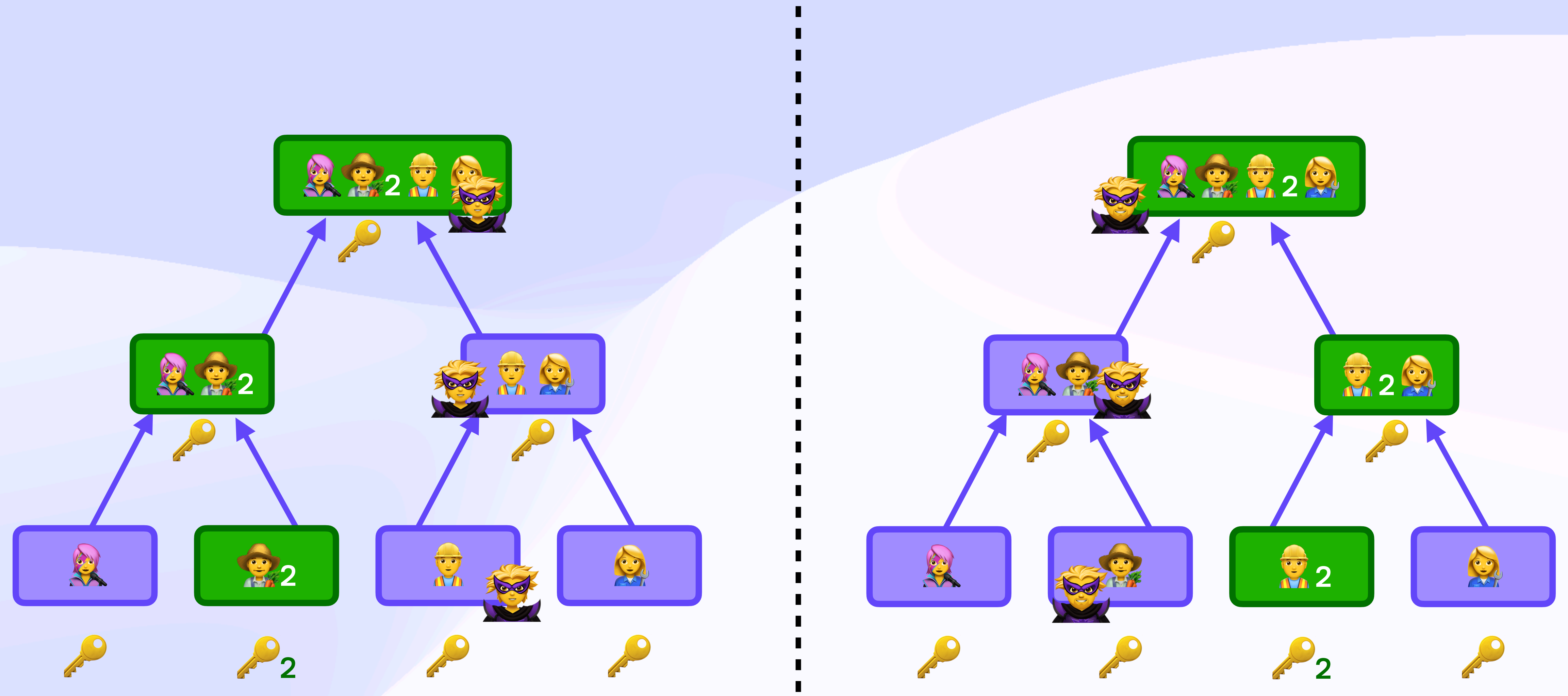
# Self-Healing Concurrent Group Encryption

## *Concurrency Problem!*



# Self-Healing Concurrent Group Encryption

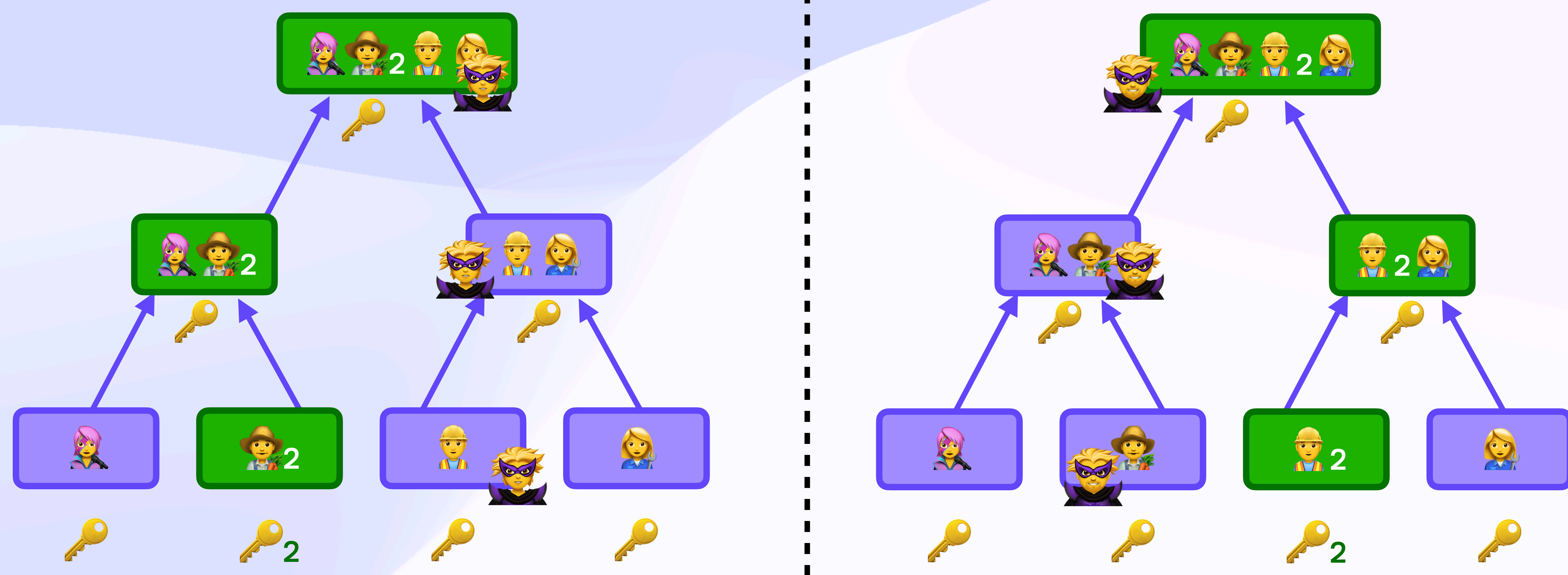
## *Concurrency Problem!*





# Self-Healing Concurrent Group Encryption

## Concurrency Problem!



# Self-Healing Concurrent Group Encryption

## *Middle Ground*





# Self-Healing Concurrent Group Encryption

## *Middle Ground*





# Self-Healing Concurrent Group Encryption

## *Middle Ground*





# Self-Healing Concurrent Group Encryption

## *Middle Ground*



Revocation

# ***What To Do With Revocation Cycles?***



Revocation

# ***What To Do With Revocation Cycles?***



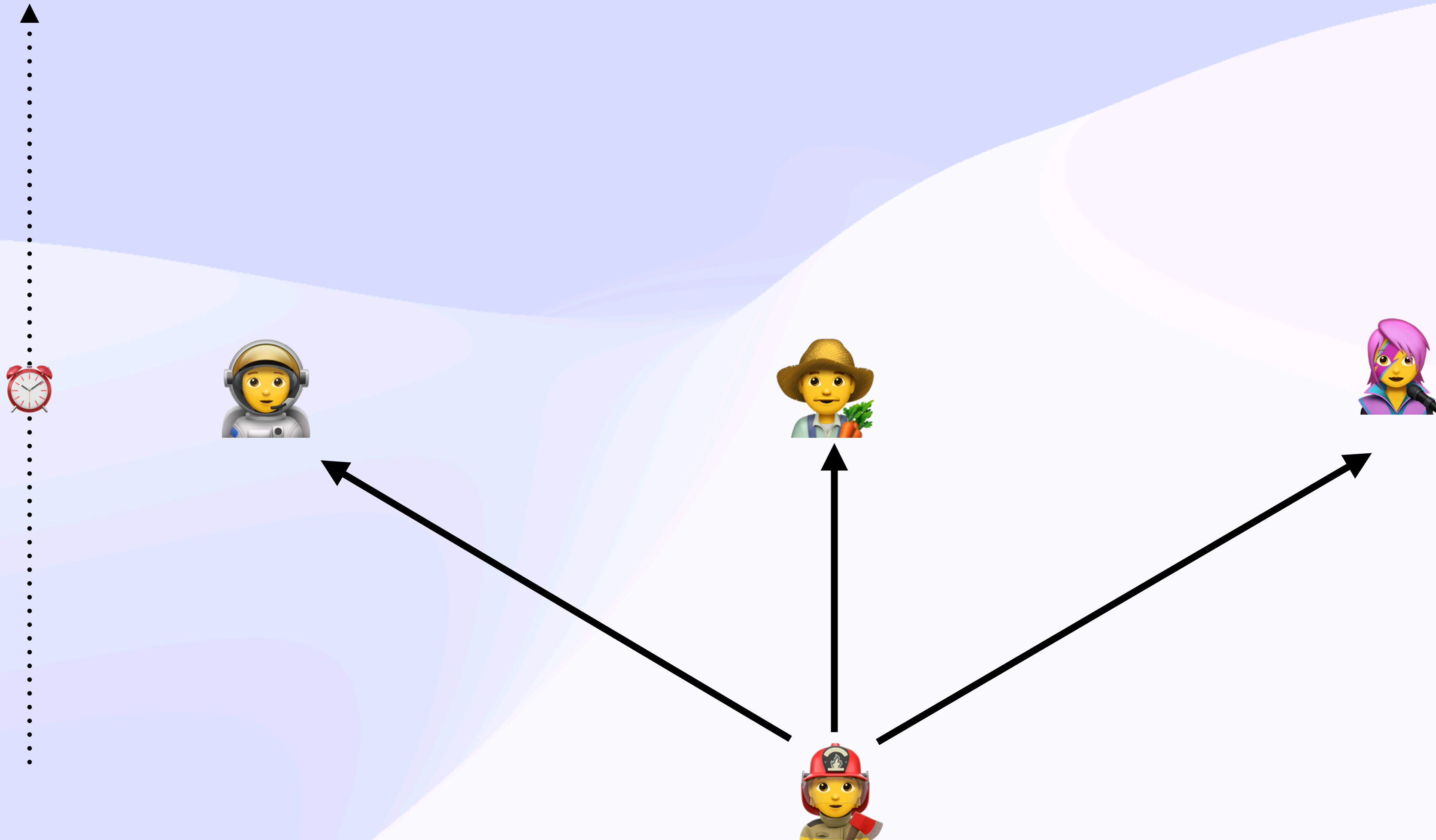
Revocation

# ***What To Do With Revocation Cycles?***



Revocation

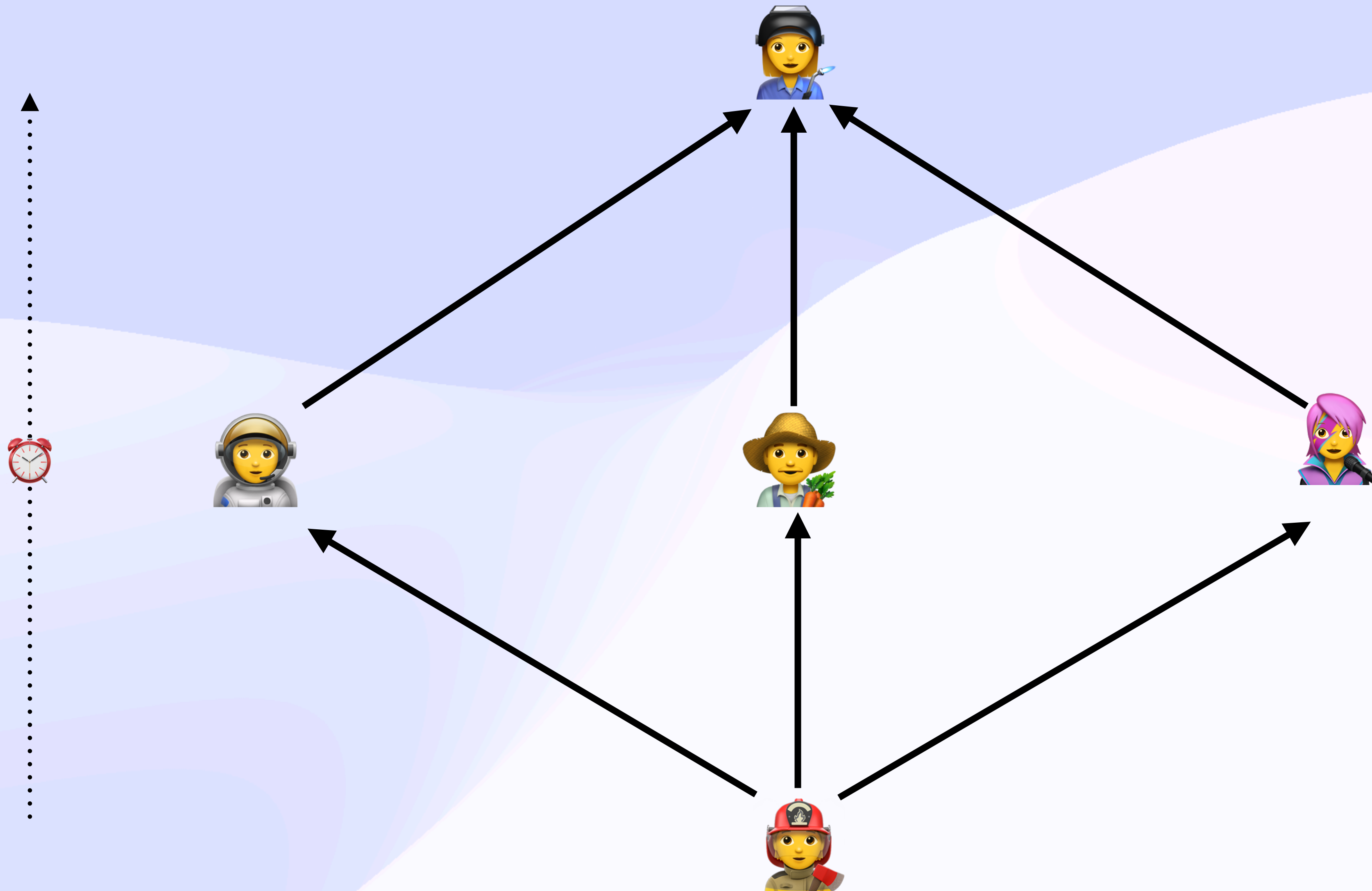
# ***What To Do With Revocation Cycles?***





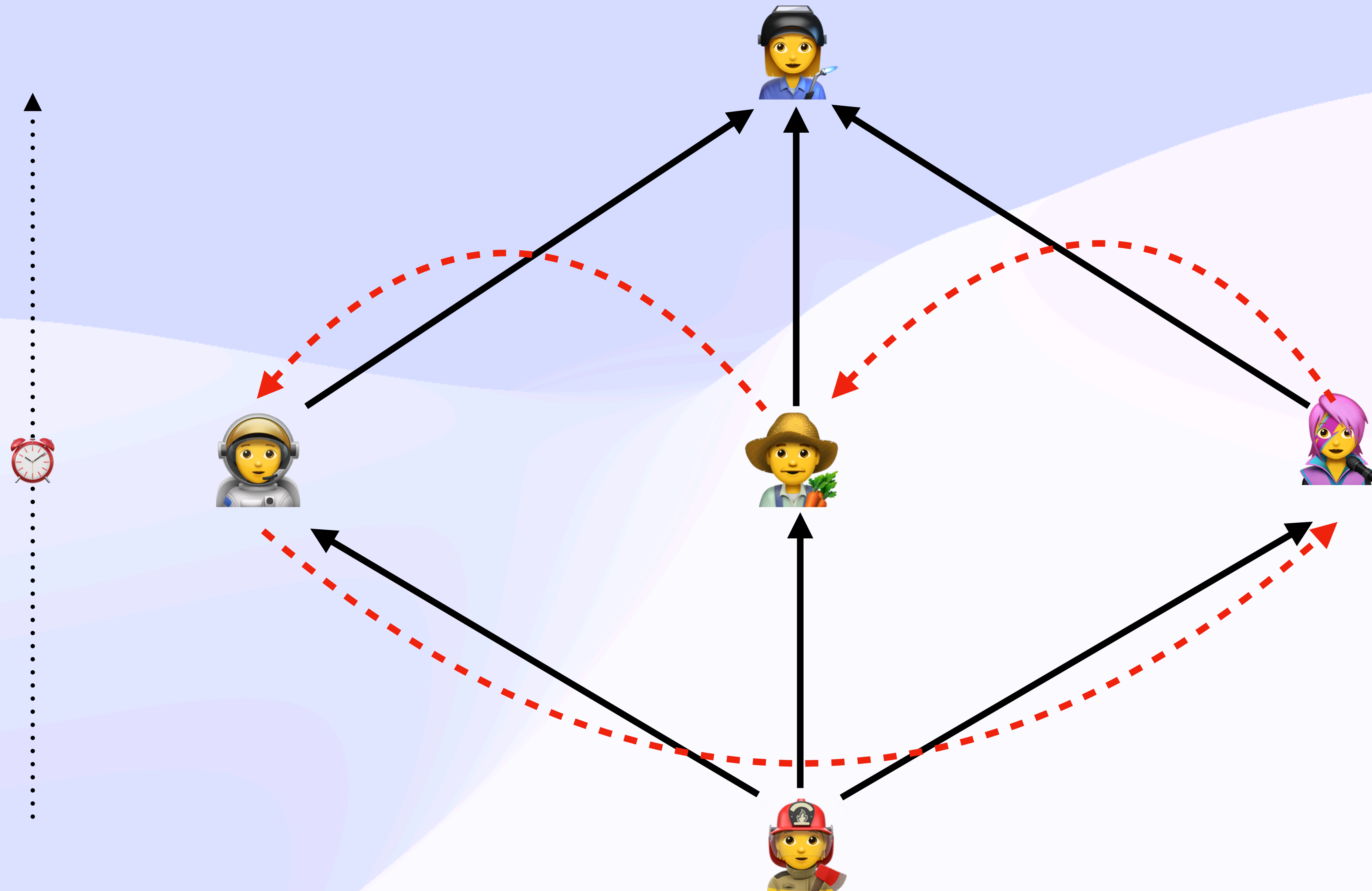
Revocation

# ***What To Do With Revocation Cycles?***



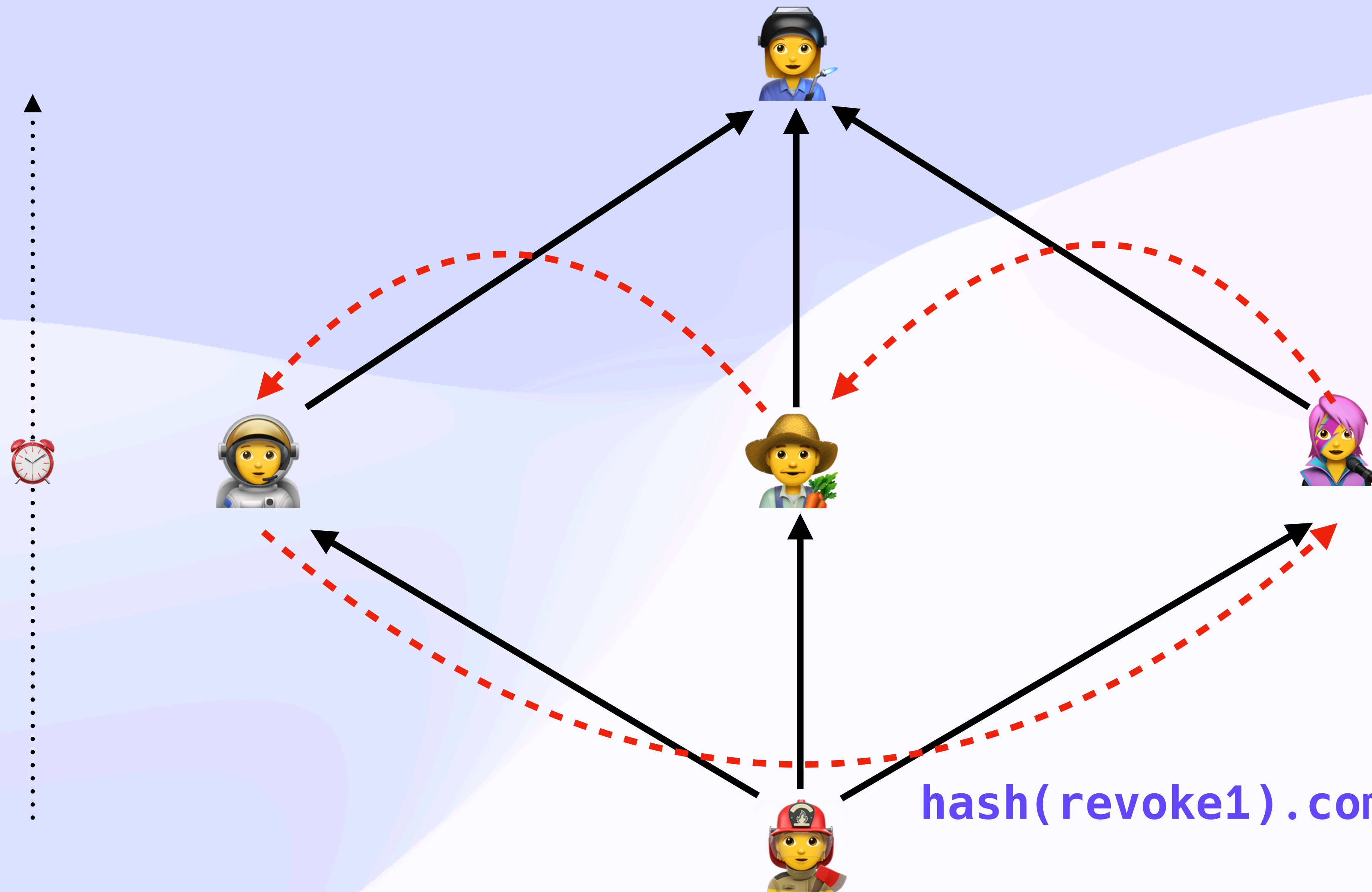
Revocation

# ***What To Do With Revocation Cycles?***



## Revocation

# *What To Do With Revocation Cycles?*

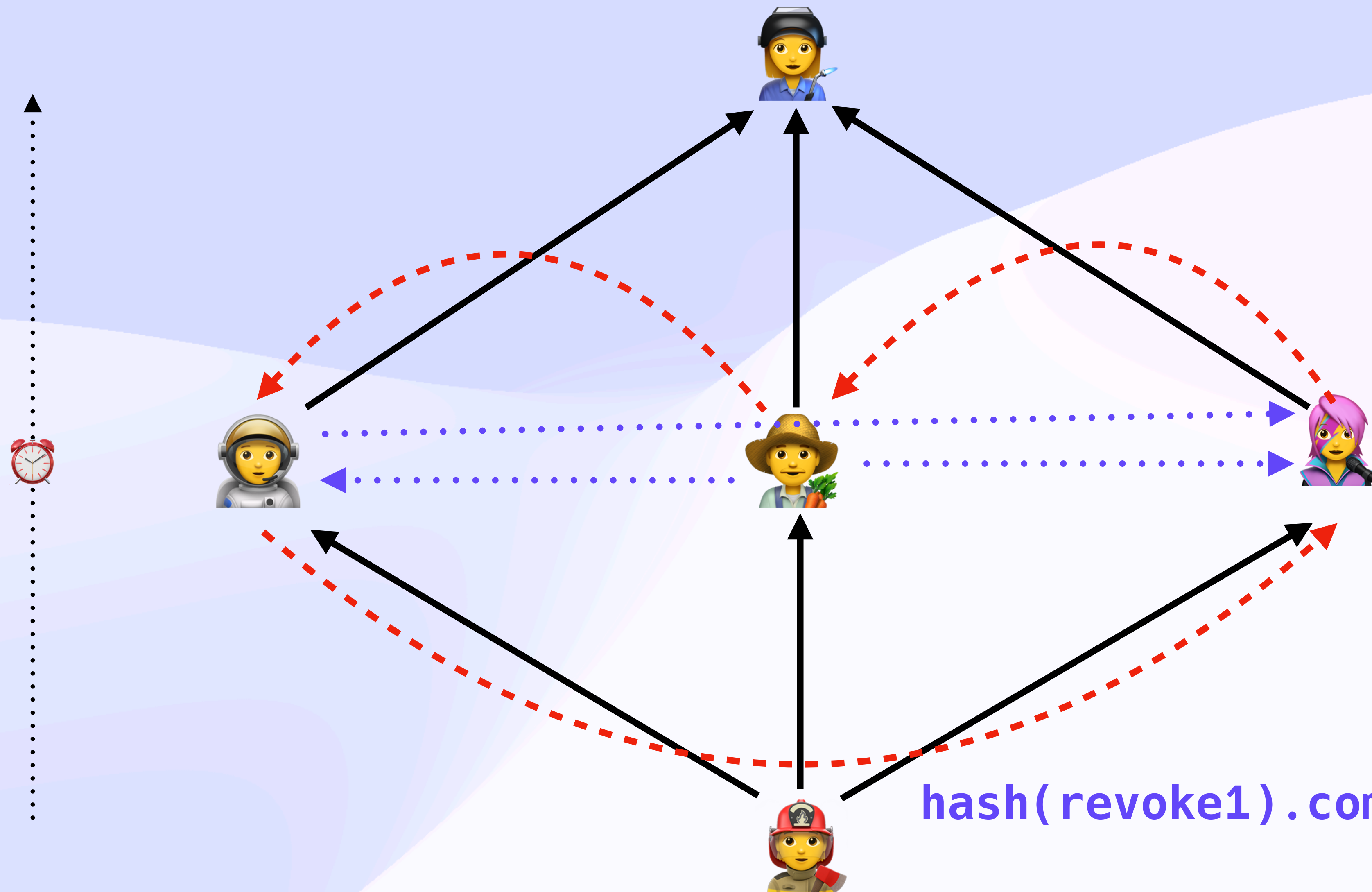


`hash(revoke1).compare(hash(revoke2))`



## Revocation

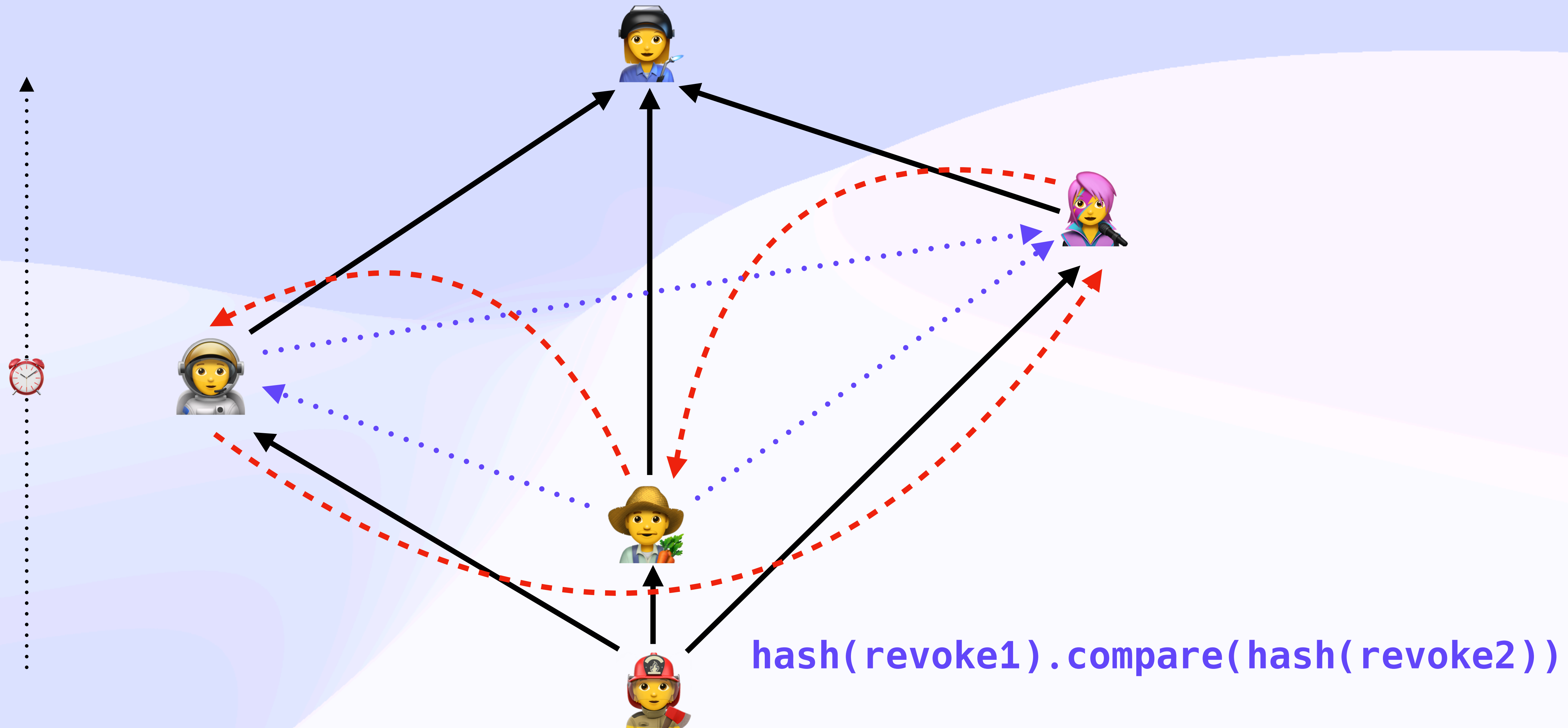
# *What To Do With Revocation Cycles?*



`hash(revoke1).compare(hash(revoke2))`

# Revocation

## *What To Do With Revocation Cycles?*



## Revocation

# *What To Do With Revocation Cycles?*

