



Deploying Cryptography in Practice

Karn Seth <karn@google.com>

Overview

Private Computing Team/ Privacy Research

- Focused on advanced/ novel cryptography and applications
 - Secure Multiparty Computation
 - Fully Homomorphic Encryption
 - Etc.

Other Cryptography teams

- Information Security Engineering (ISE)
 - Security/ Design reviews for products
 - Common cryptographic libraries such as Tink
- BoringSSL
 - Make OpenSSL safe and “boring”
- Key storage and management systems
 - Manage key storage, rotation, access control
- Many product teams

In this talk

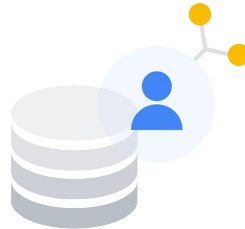
- Examples of deployments of MPC in practice
 - Secure Aggregation for Federated Learning
 - (if time) Exposure Notifications - Private Analytics
- Insights from ISE
 - Making Crypto easier to use correctly
 - Key management and rotation
- Overall:
 - Challenges, insights and lessons learned from deploying cryptography in practice

MPC Overview

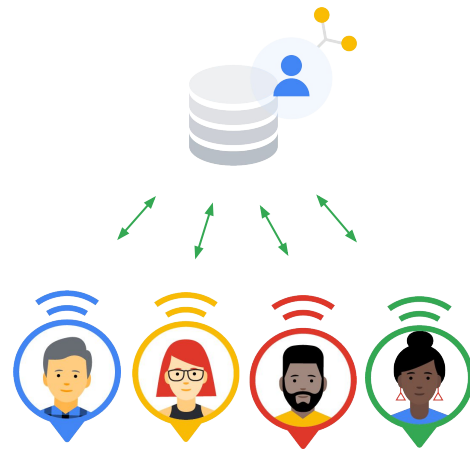
Different settings for MPC



Businesses/
large parties



User to Server

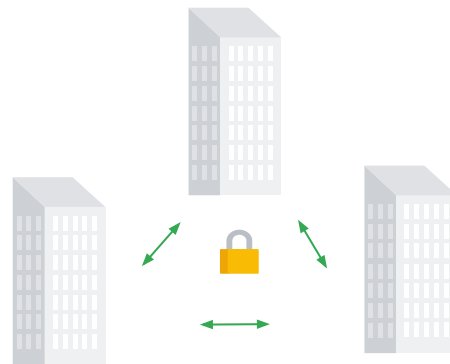


Multiple users to
Server

Examples

Business-to-Business

- Private Set Intersection
- Collaborative Private Statistics
- Private auctions
- ...

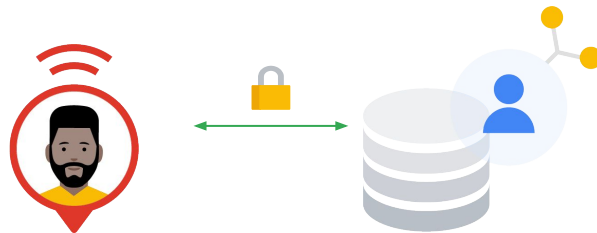


Businesses/
large parties

Examples

User to Server:

- Private Contact Discovery
- Private Information Retrieval
- ...

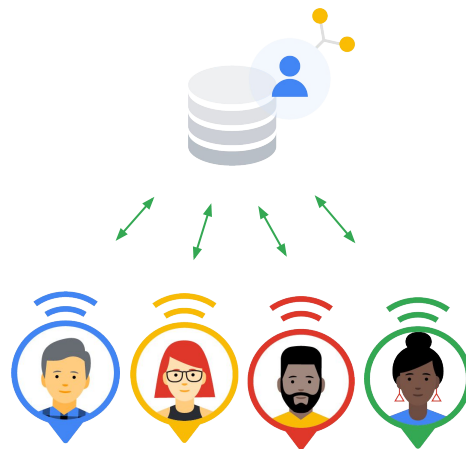


User to Server

Examples

Multiple users to Server(s):

- Collaborative ML
- Private metric collection
- ...



Multiple users to
Server

Examples

Multiple users to Server(s): ★

- Collaborative ML
- Private metric collection
- ...



Multiple users to
Server

Challenges in deploying MPC

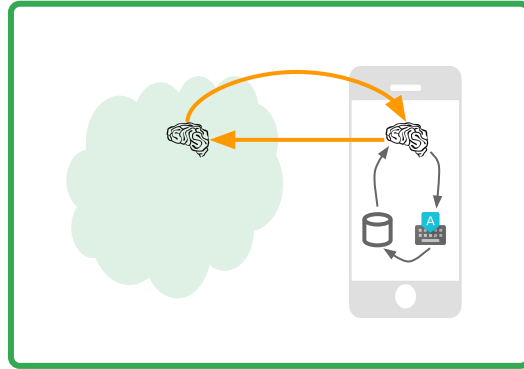
- **Costs:**
 - Communication, Computation
 - Generic protocols are common in the literature, but very expensive
 - “Tailored” protocols can be very helpful
 - Engineering Costs
 - Scalability, complex system architecture
 - Coordination across organization boundaries
 - Specialized Knowledge
- **But, strong privacy and security guarantees**
 - Relying only on cryptography, not on the security of any particular trusted environment

Secure Aggregation for Federated Learning

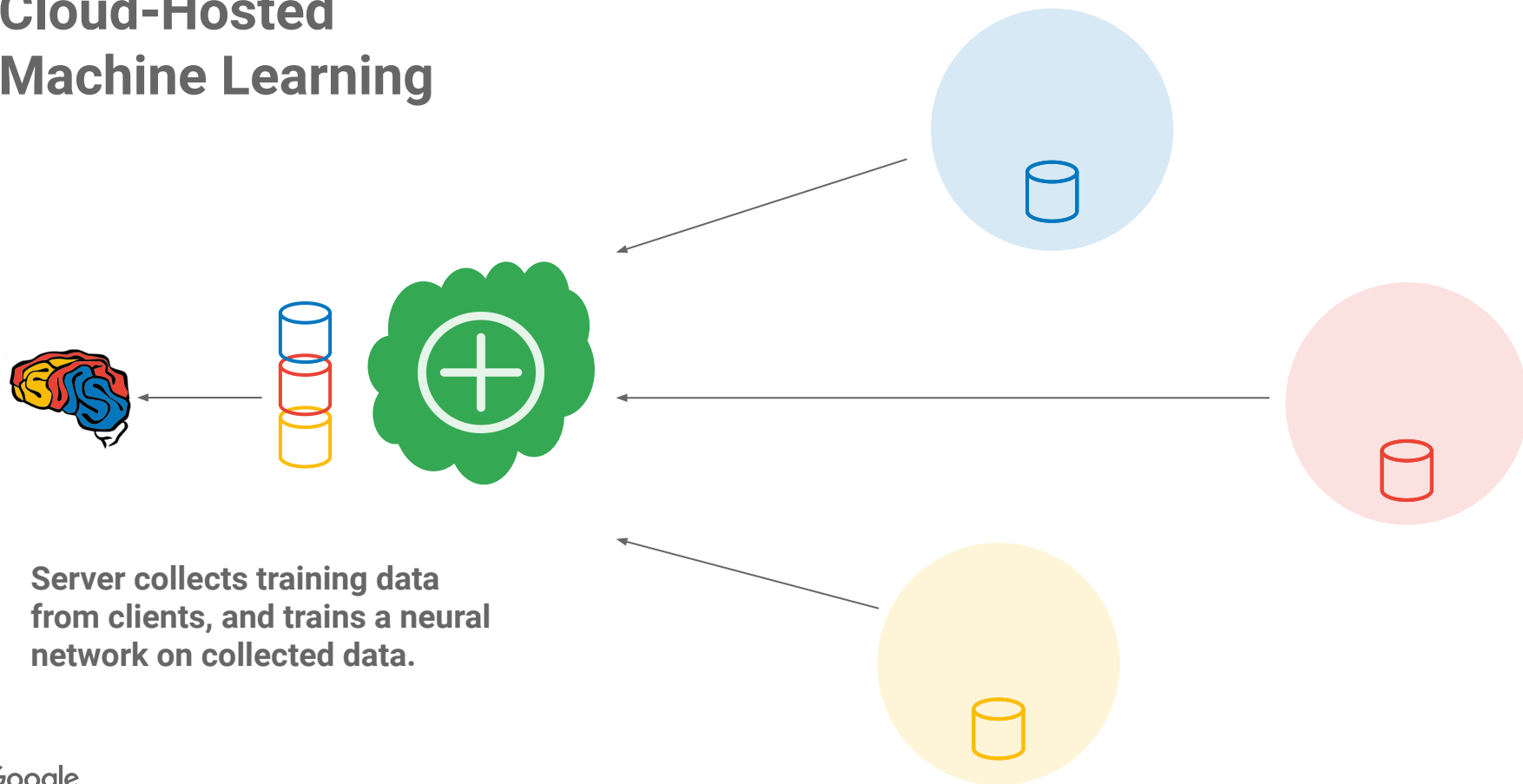
“Practical Secure Aggregation for Privacy-Preserving Machine Learning”
K. A. Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan
McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, Karn Seth.
CCS 2017

Federated Learning

Federated learning is the problem of training a shared global model under the coordination of a central server, from a federation of participating devices which store their training data locally.

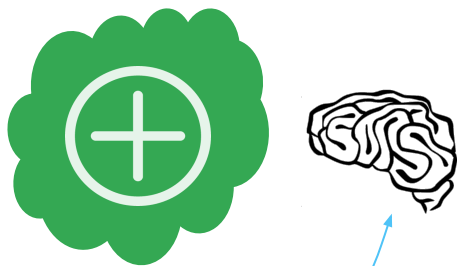


Cloud-Hosted Machine Learning



Federated Learning

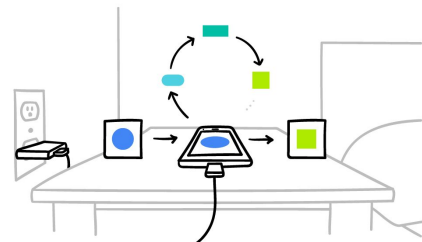
1. Server selects a sample of
e.g. 1000 online devices.



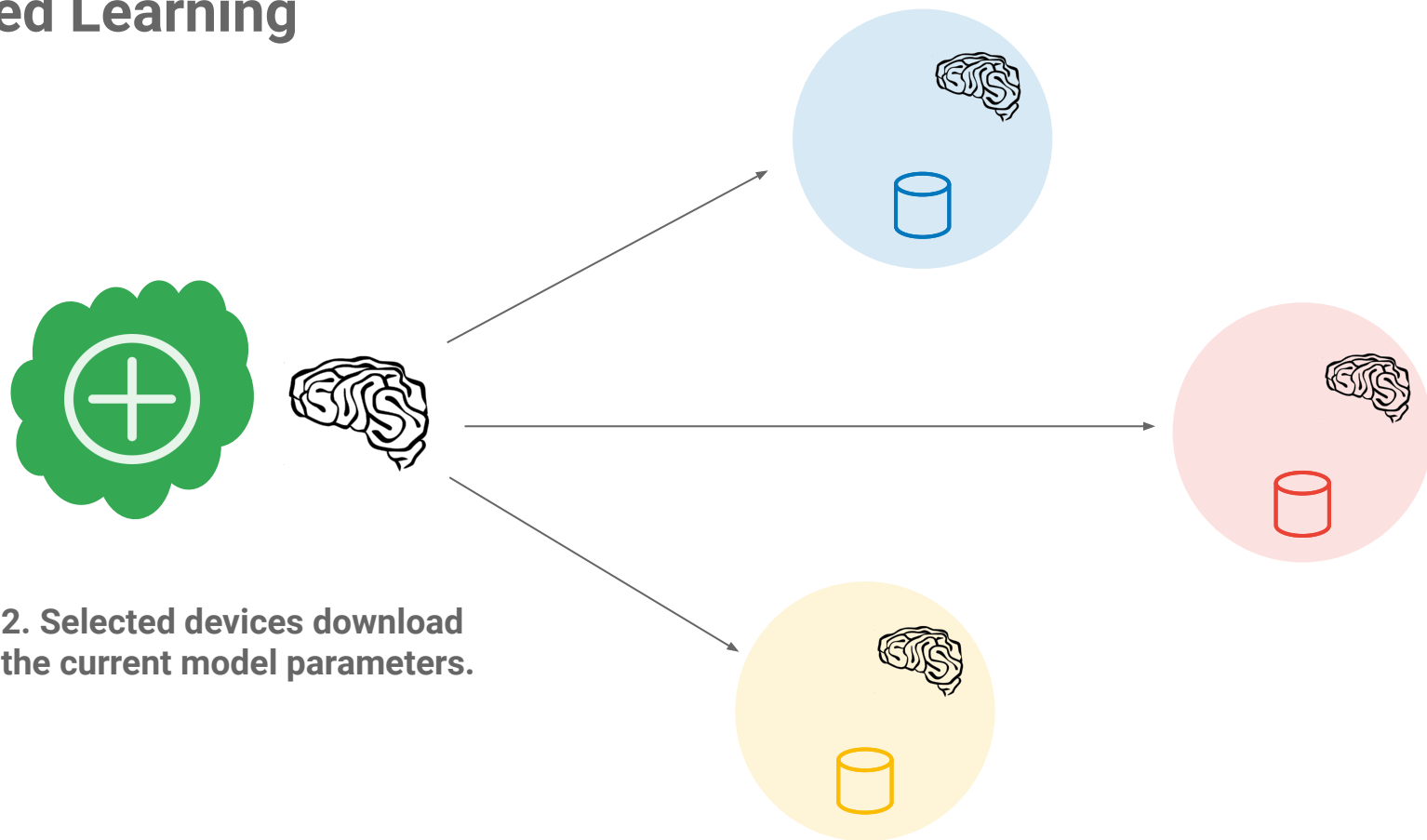
Current Model
Parameters

Mobile
Device

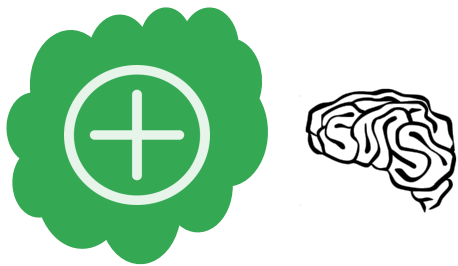
Local
Training
Data



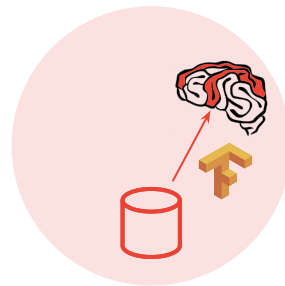
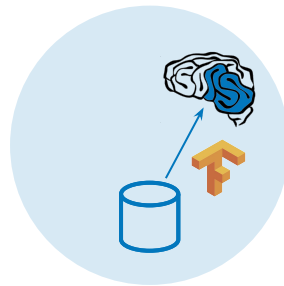
Federated Learning



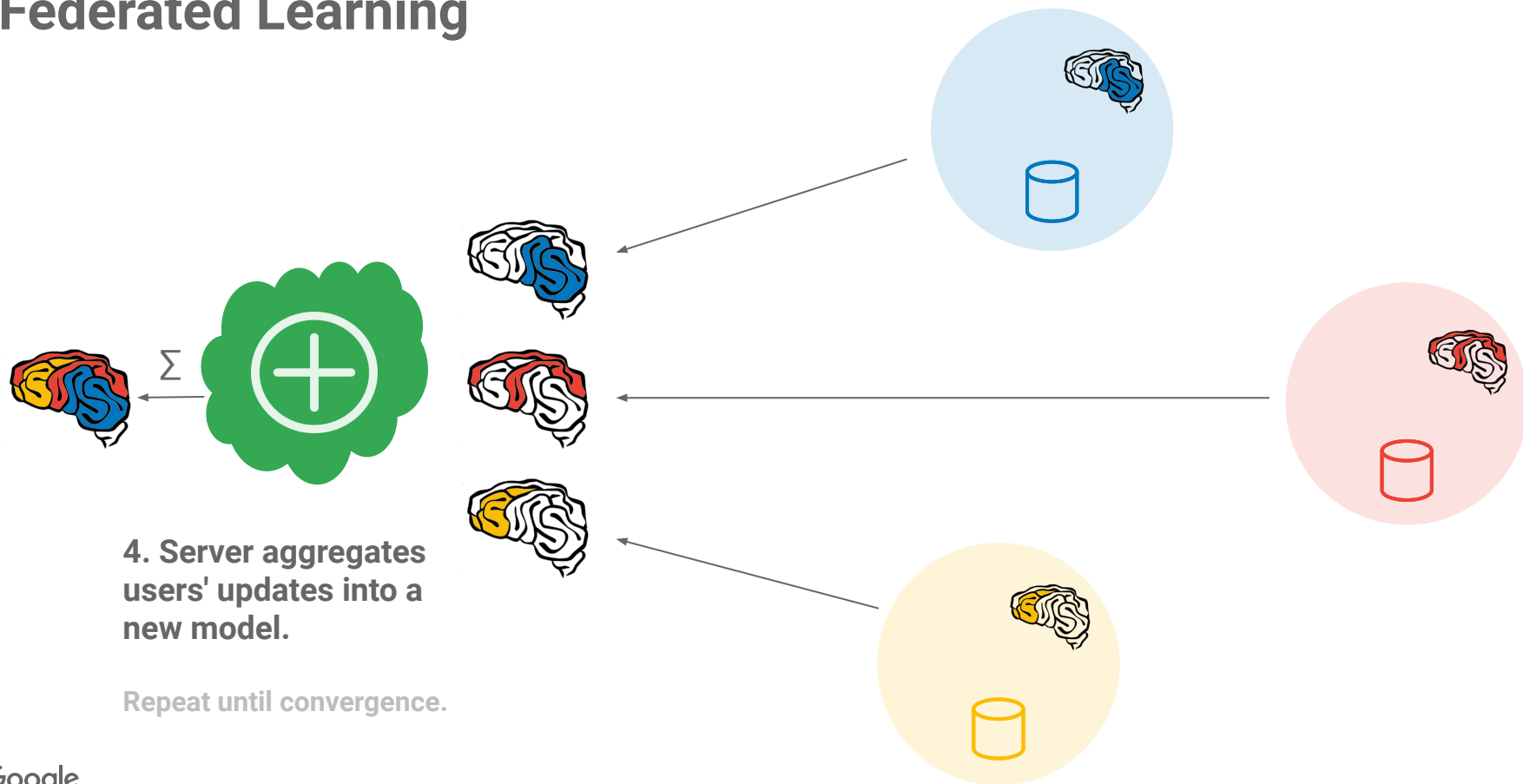
Federated Learning



3. Users run stochastic gradient descent on local training data



Federated Learning

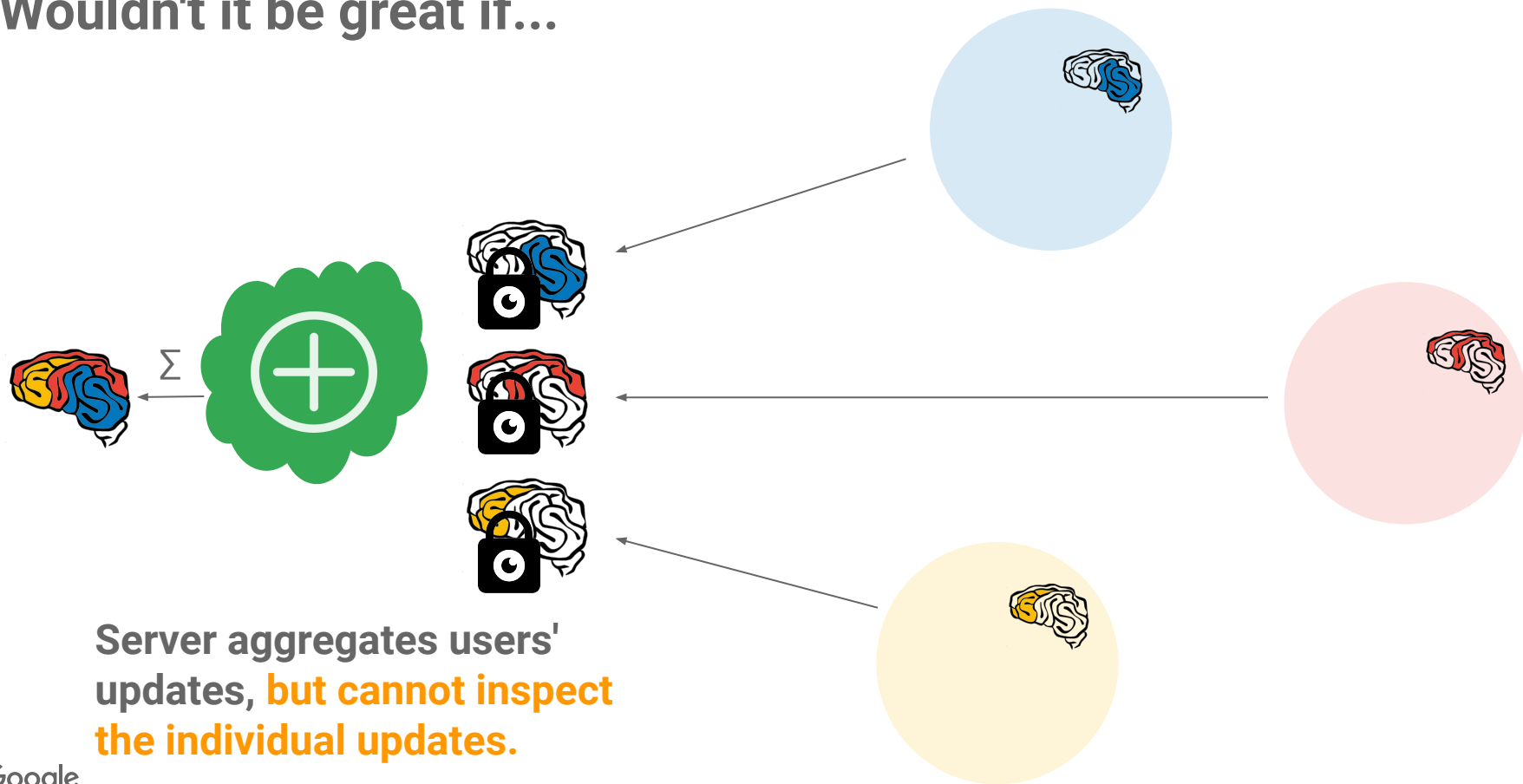


Federated Learning



**Might these updates
contain privacy-sensitive
data?**

Wouldn't it be great if...



Server aggregates users' updates, **but cannot inspect the individual updates.**

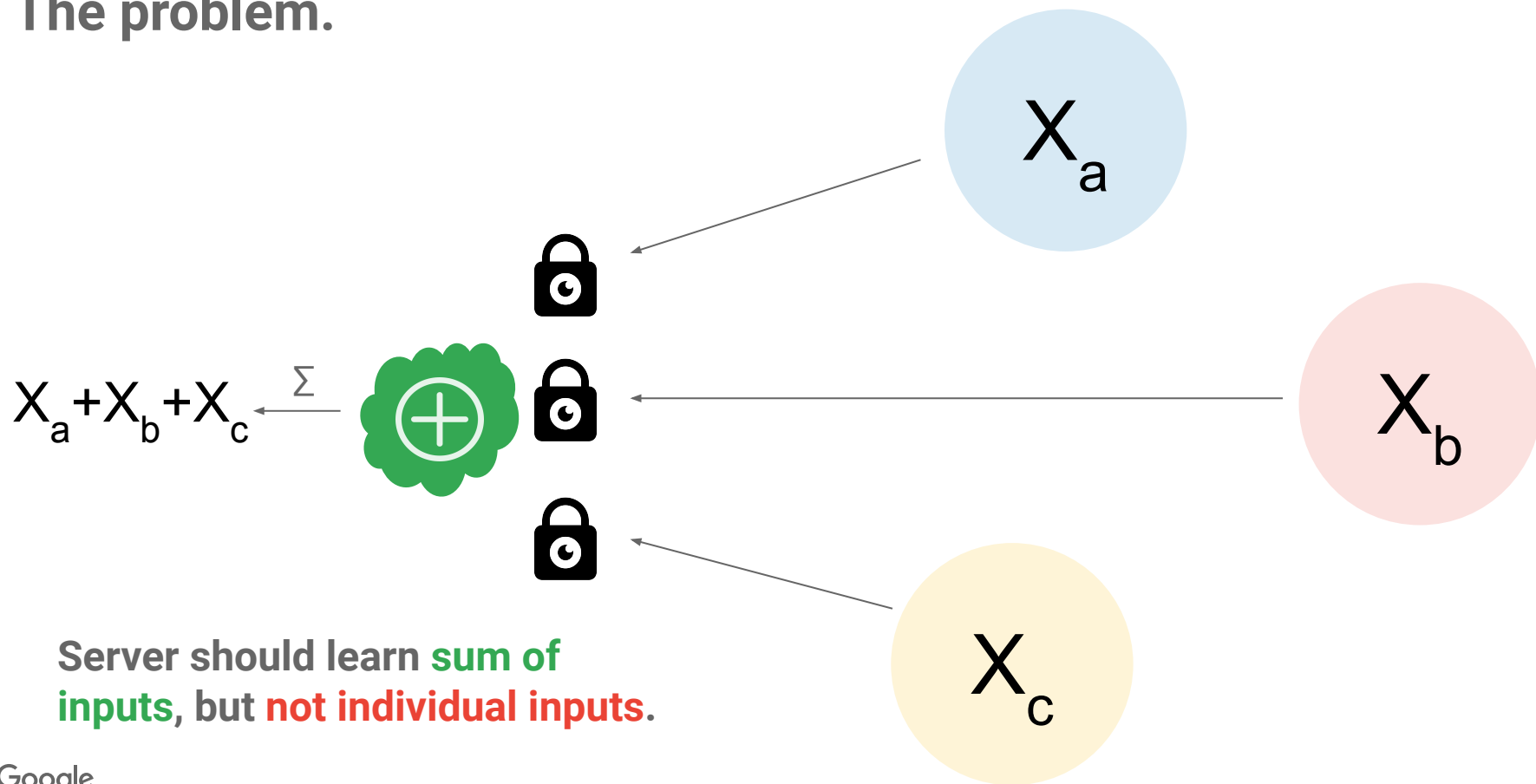
Wouldn't it be also great if...

- We could do this without too much additional cost?
 - At most 2x communication increase for clients
- We could do this scalably?
 - Handle 1000s of clients submitting values?
- We could do this reliably?
 - Handle dropouts, errors

Secure Aggregation.

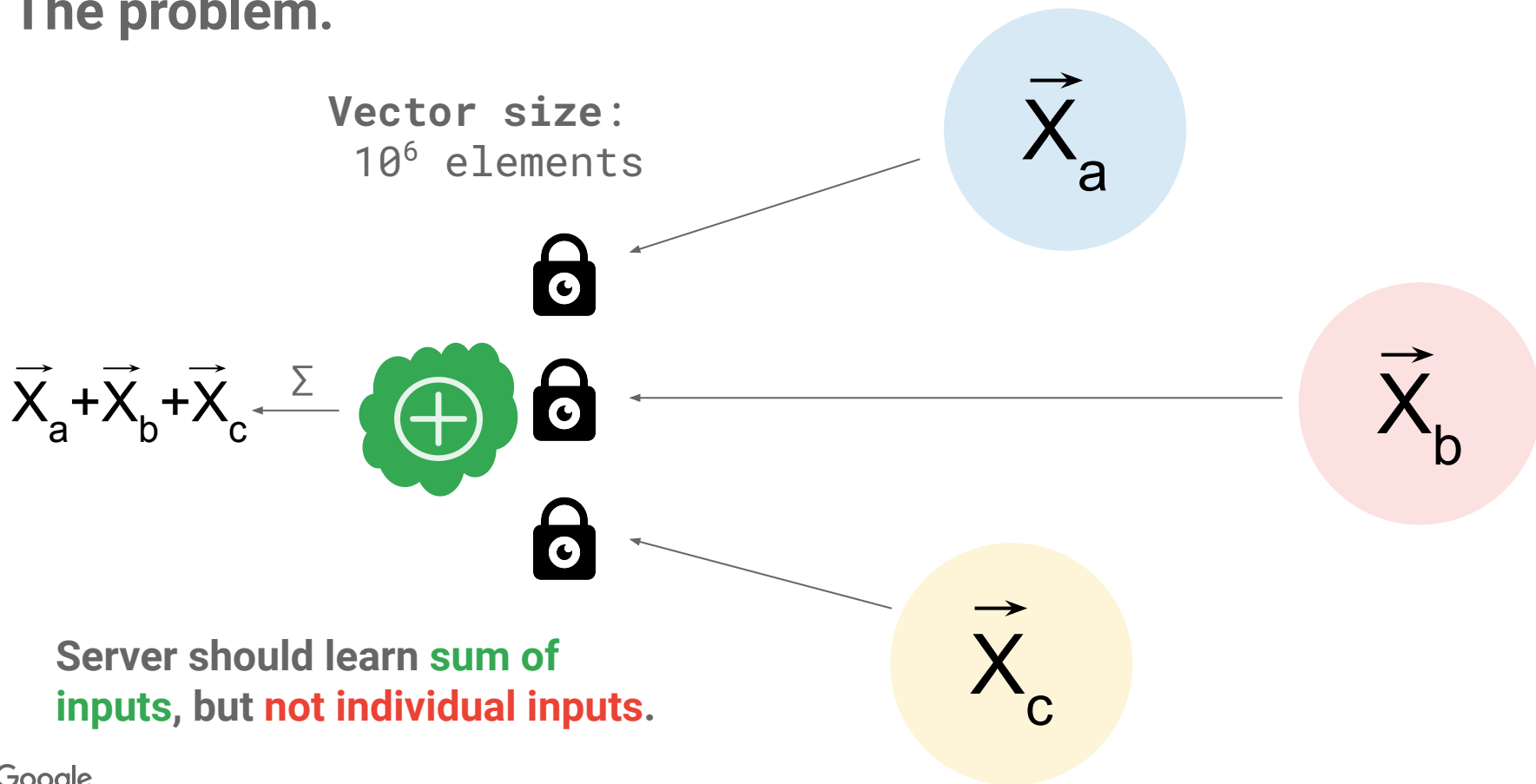
Secure Aggregation

The problem.



The problem.

Vector size:
 10^6 elements



A novel protocol for Secure Aggregation.

Existing protocols either:



Transmit
a *lot* of data

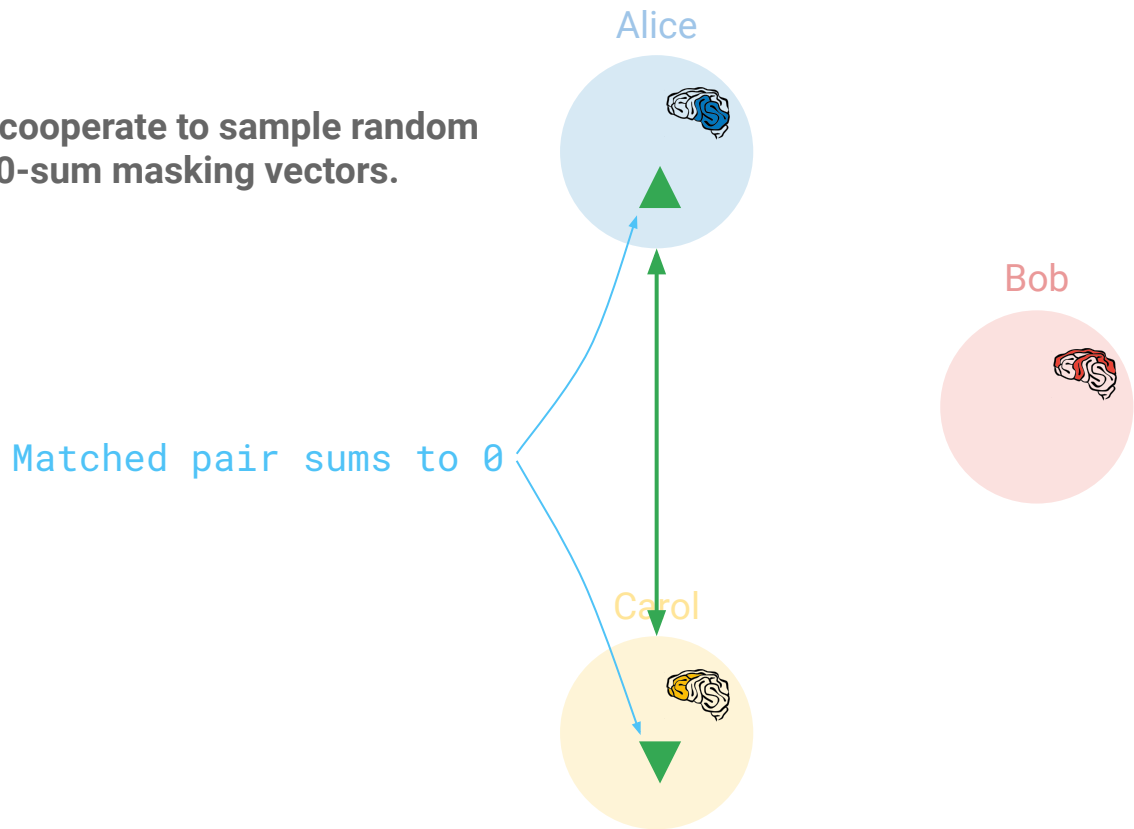


Fail when
users drop out

(or both)

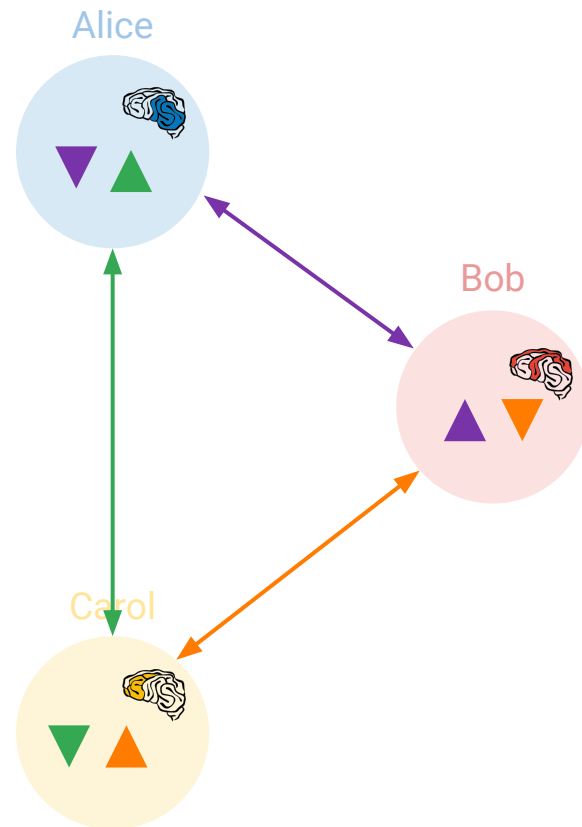
Random positive/negative pairs, *aka* antiparticles

Devices cooperate to sample random pairs of 0-sum masking vectors.

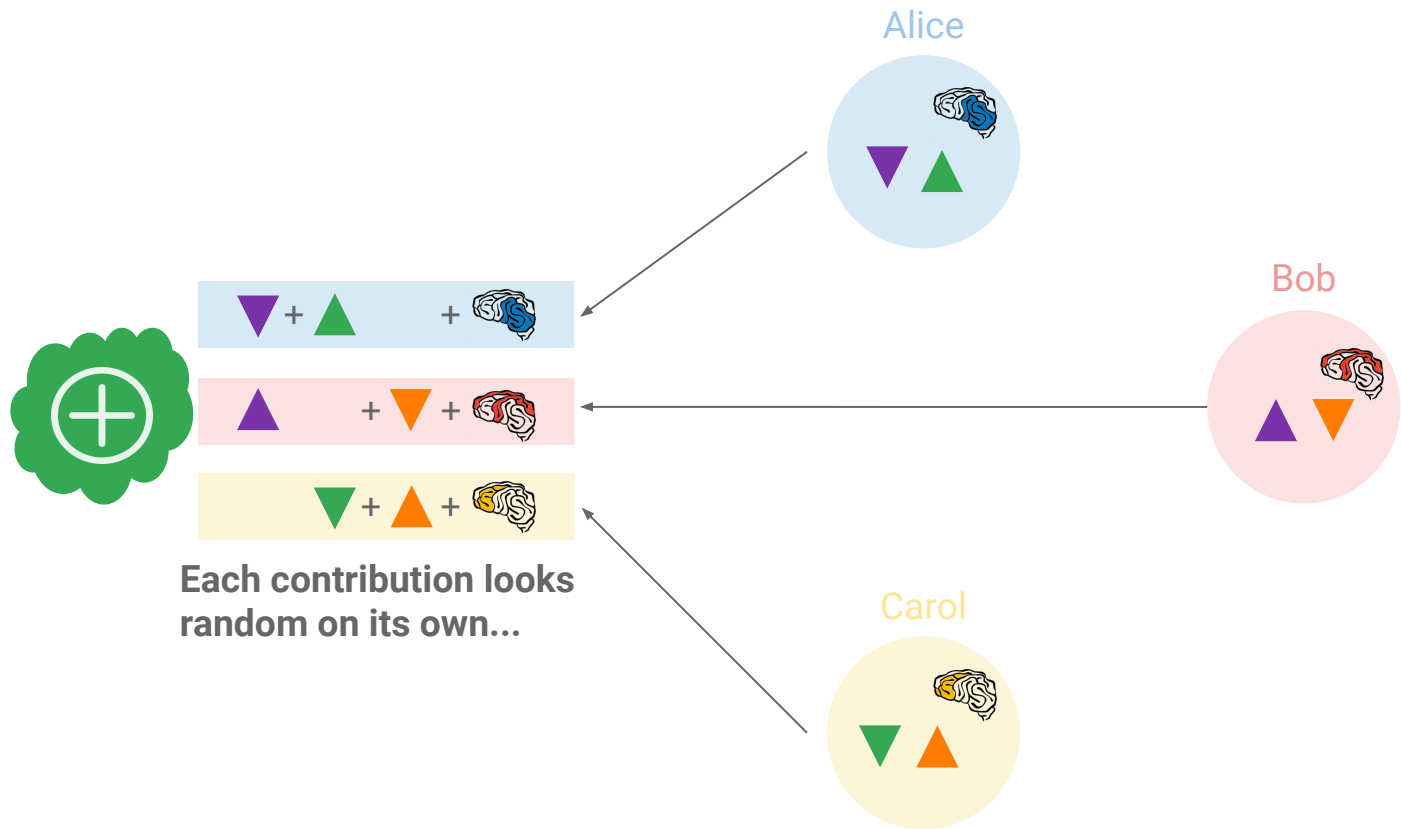


Random positive/negative pairs, *aka* antiparticles

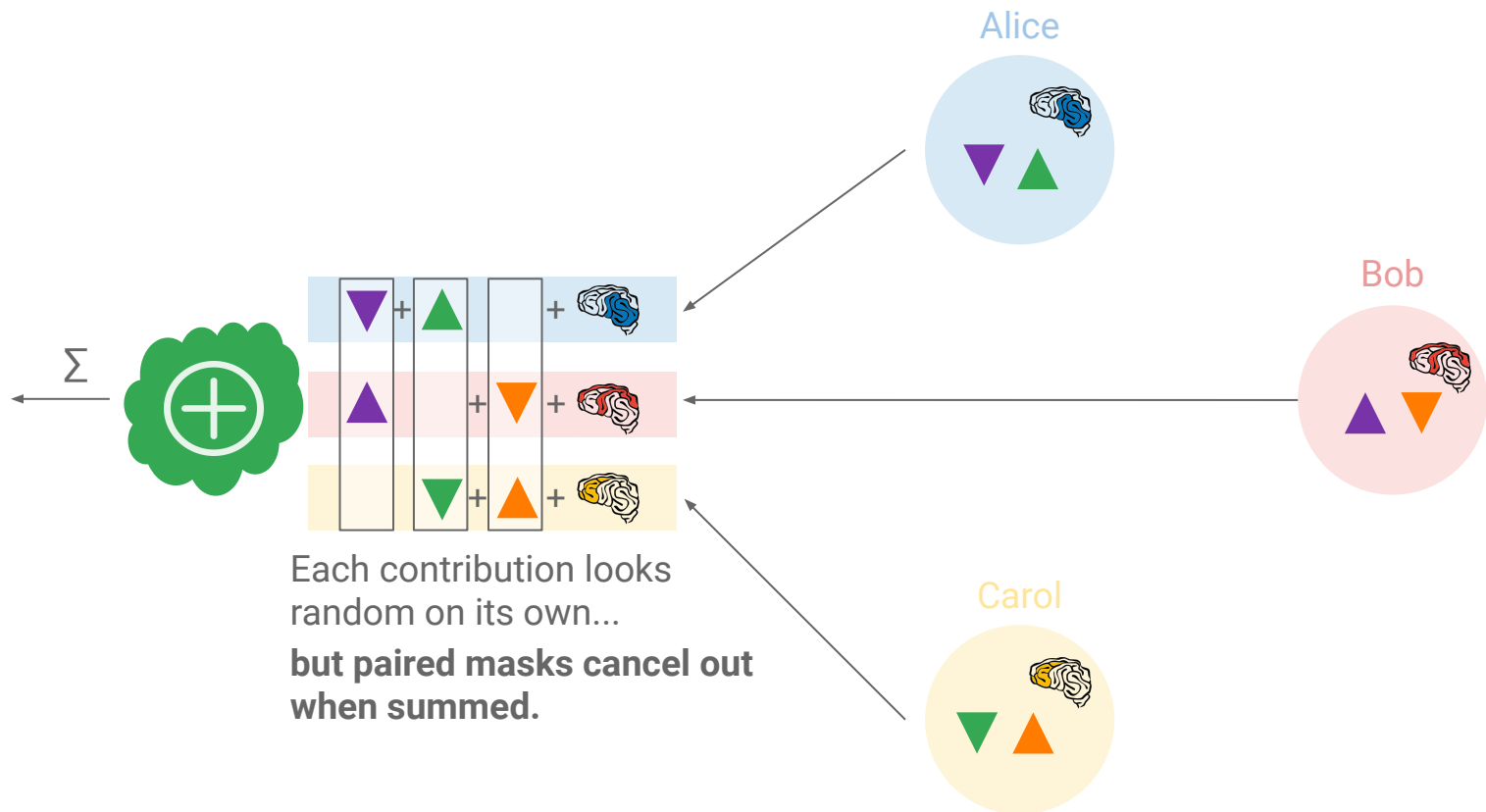
Devices cooperate to sample random pairs of 0-sum masking vectors.



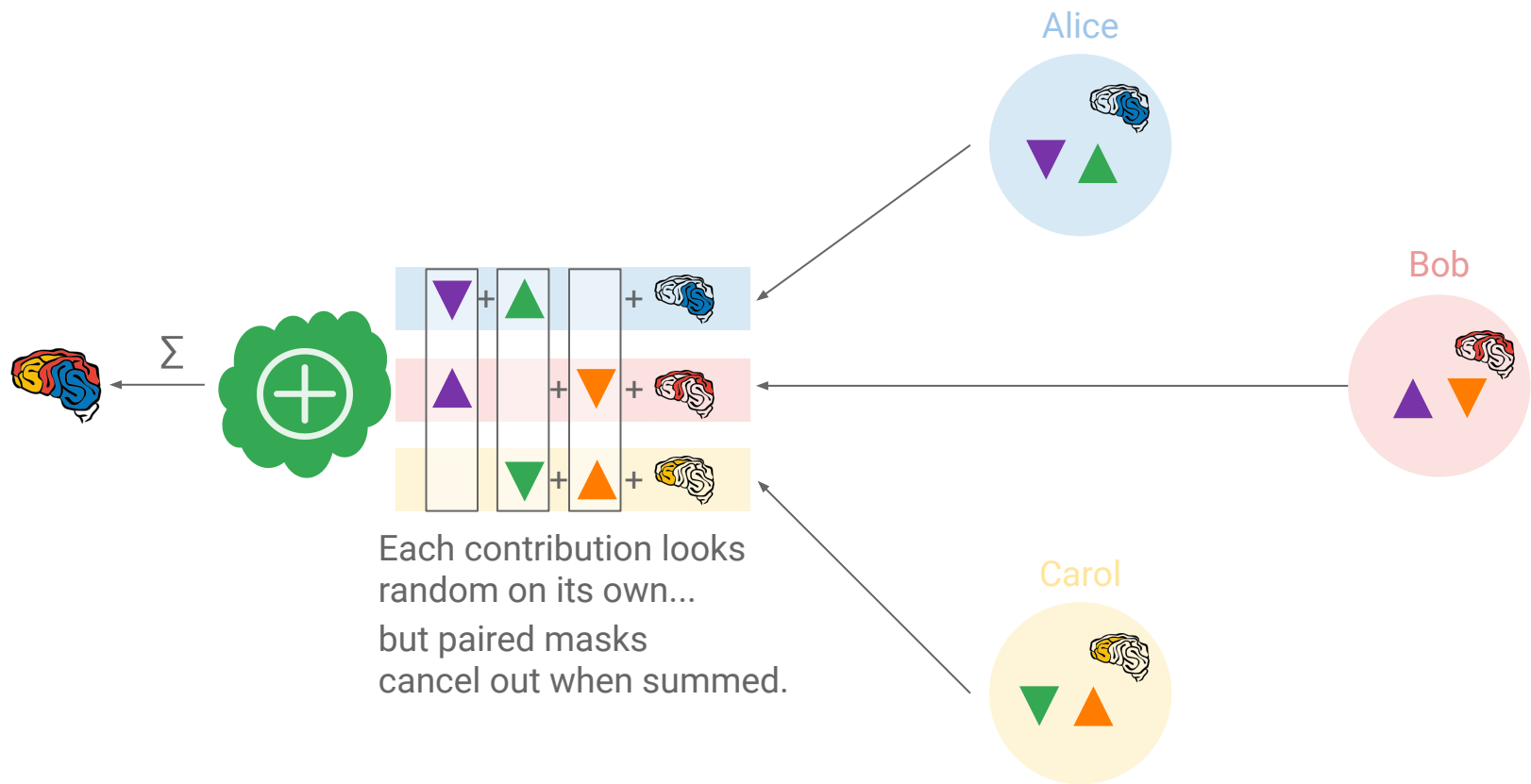
Add antiparticles before sending to the server



The antiparticles cancel when summing contributions

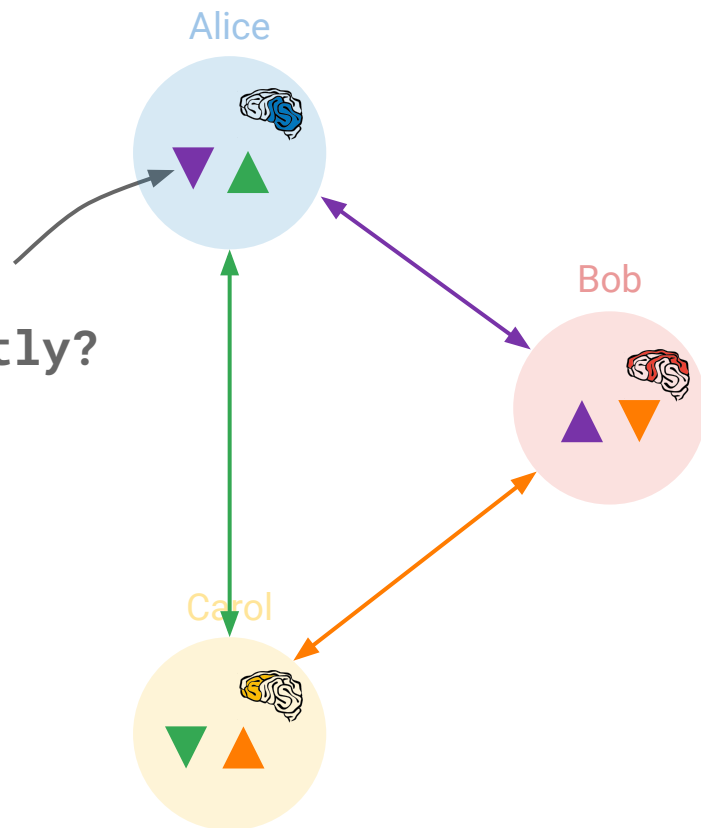


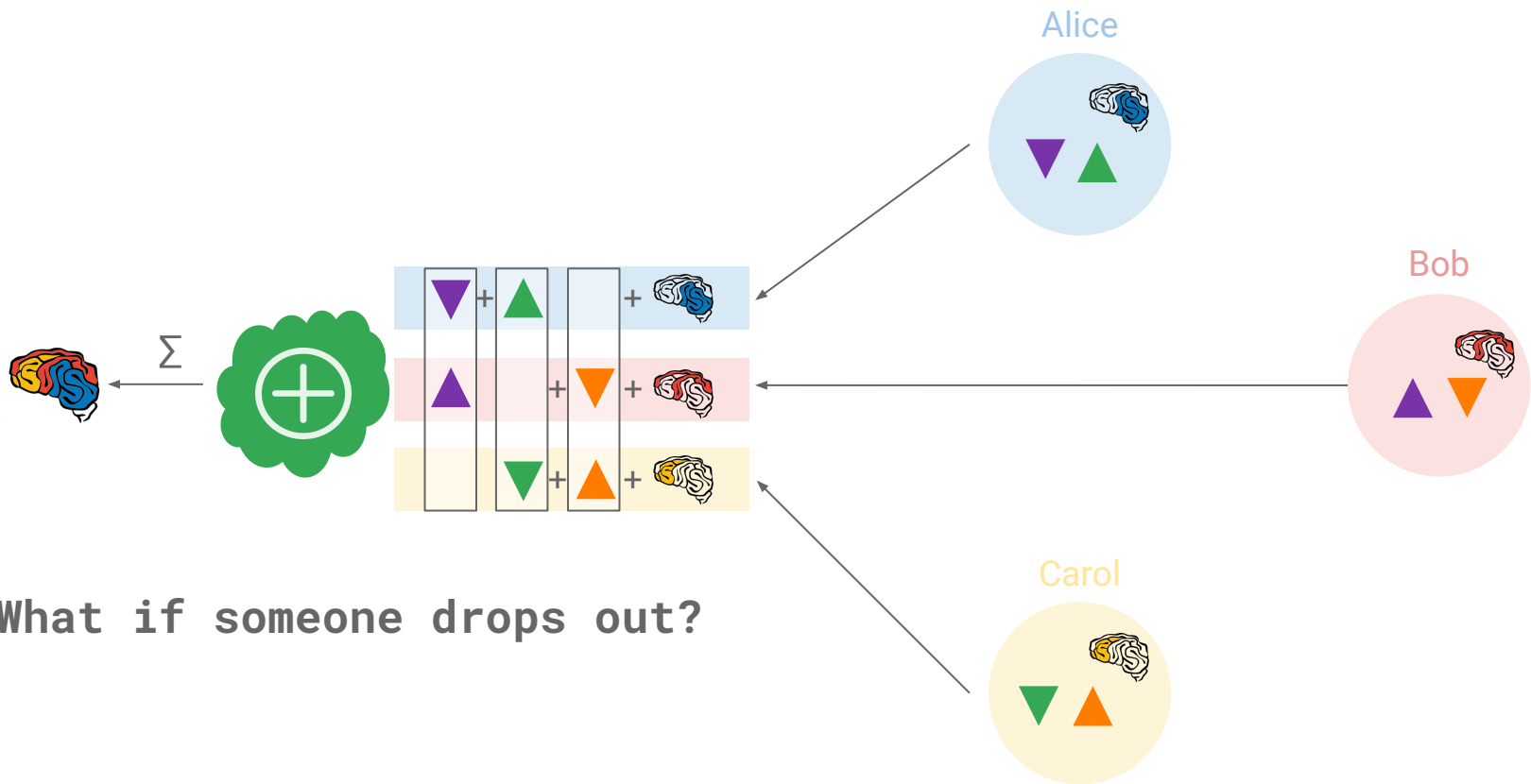
Revealing the sum.

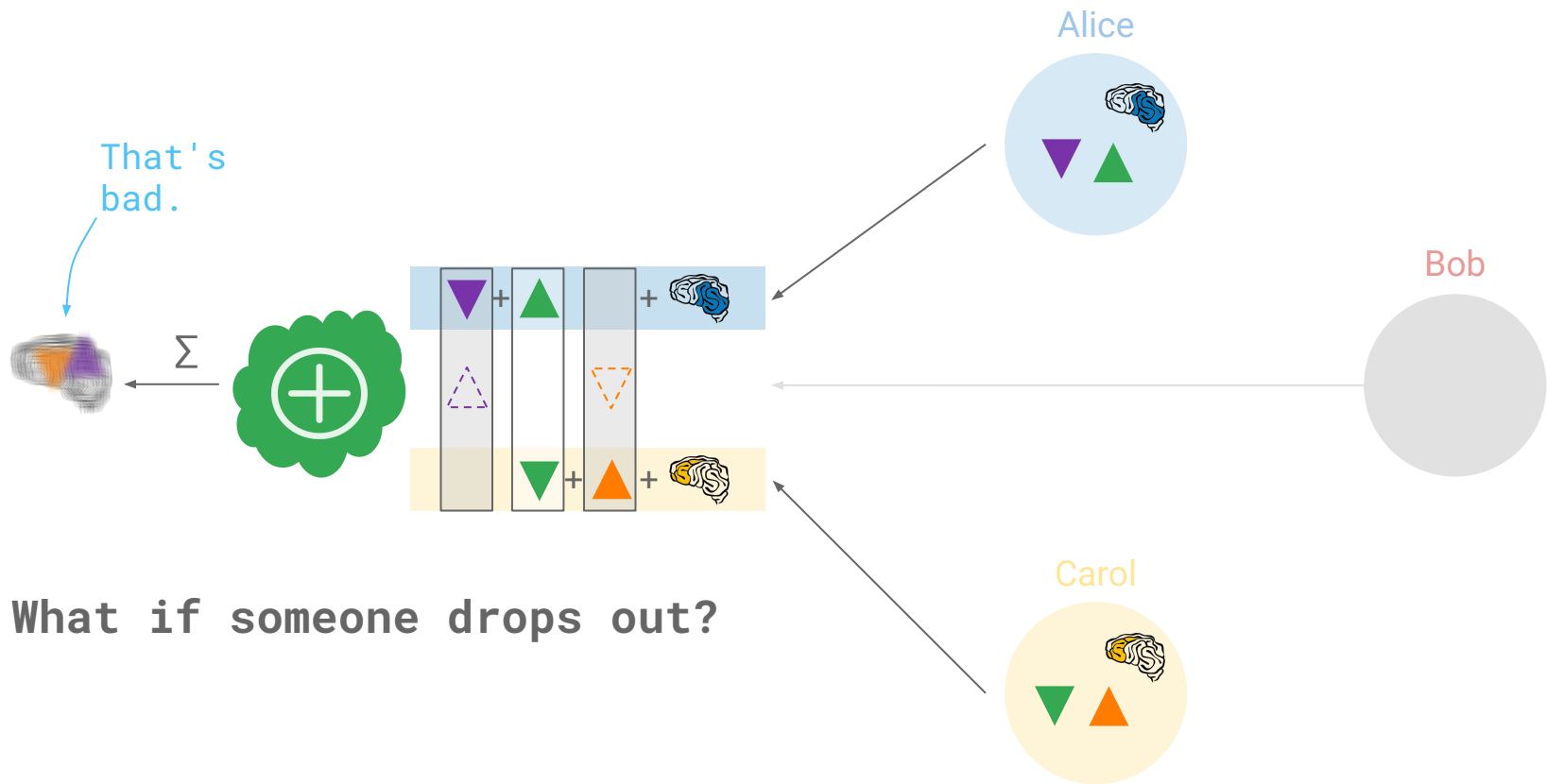


But there are two problems...

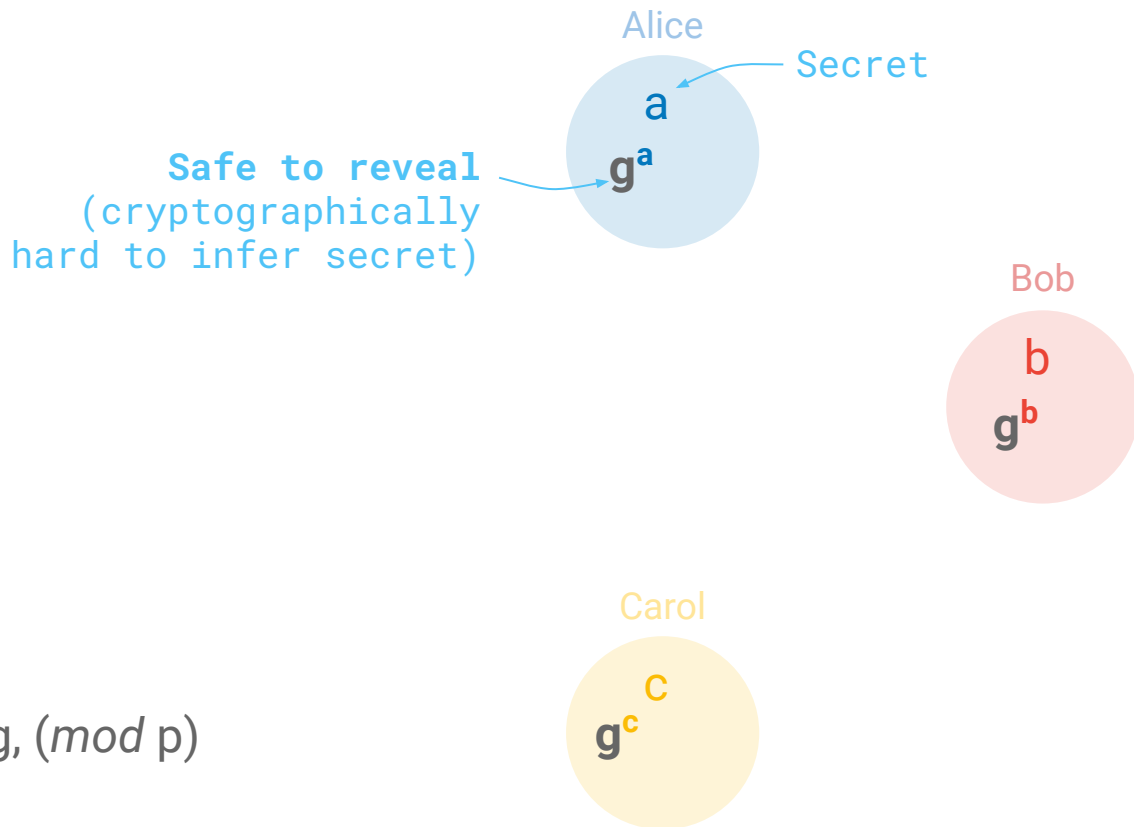
1. These vectors are big!
How do users agree efficiently?



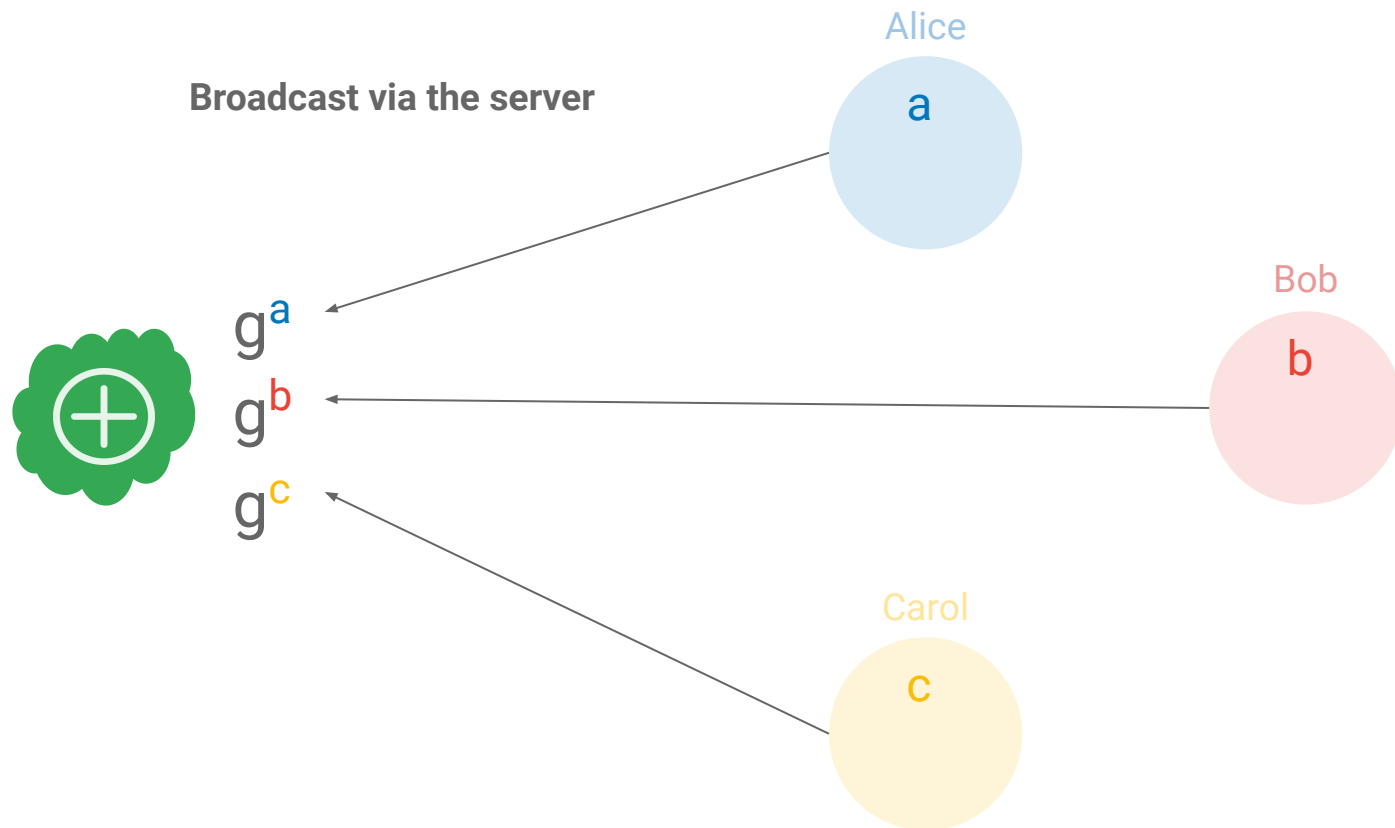




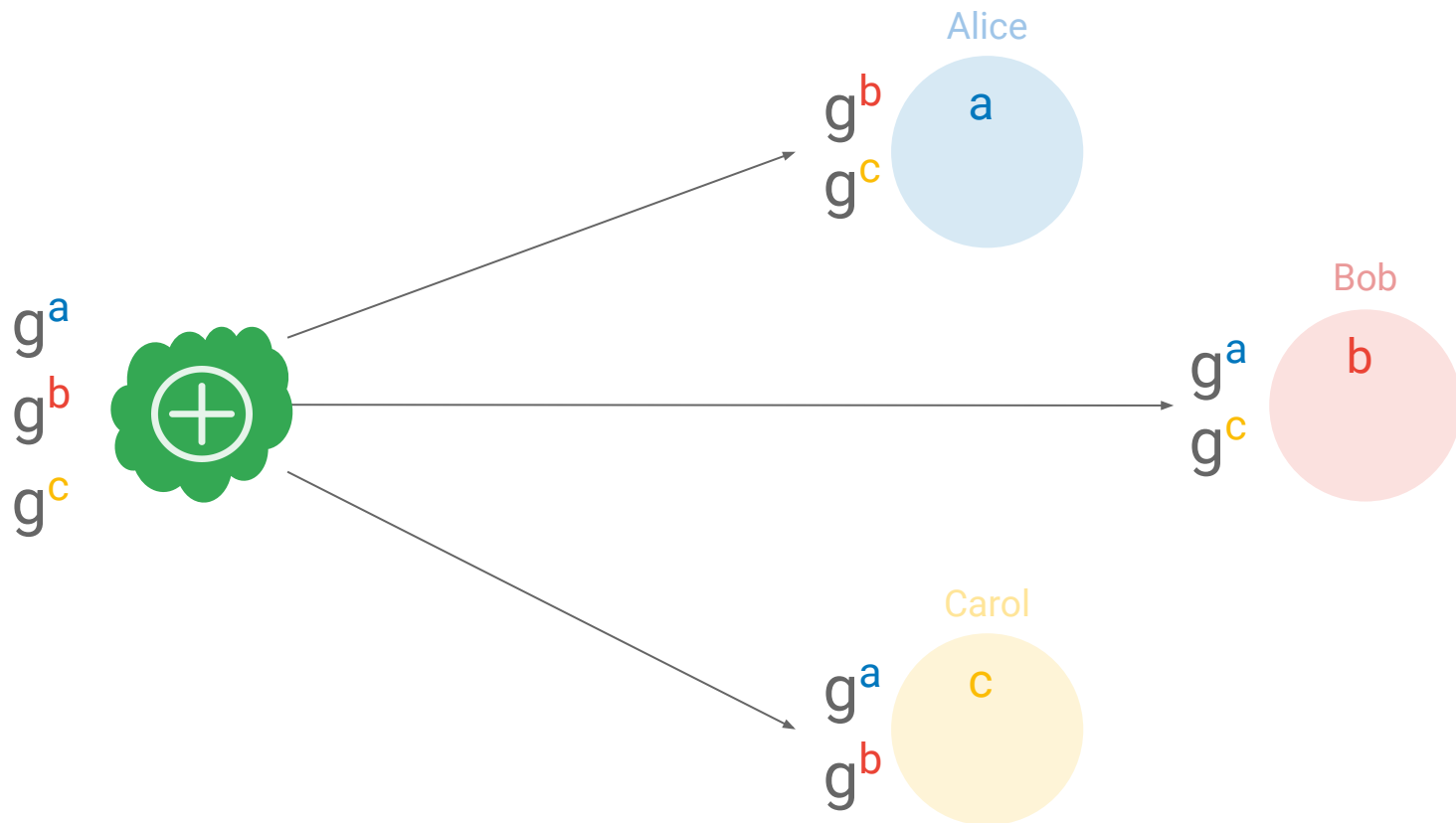
Pairwise Diffie-Hellman Key Agreement



Pairwise Diffie-Hellman Key Agreement

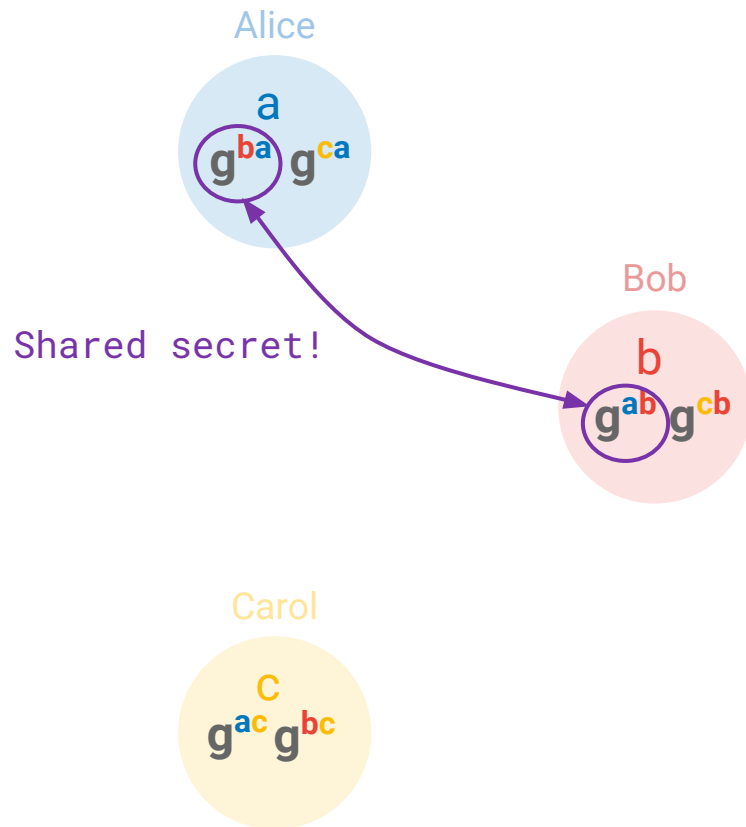


Pairwise Diffie-Hellman Key Agreement



Pairwise Diffie-Hellman Key Agreement

Secrets are scalars, but....

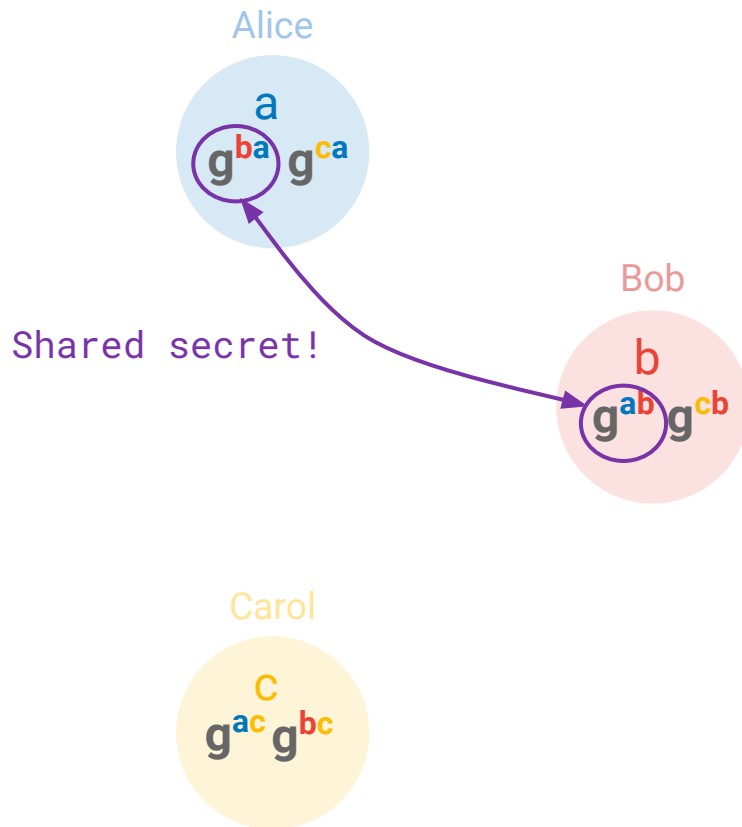


Pairwise Diffie-Hellman Key Agreement + PRNG Expansion

Secrets are scalars, but....

Use each secret to seed a
pseudorandom number generator,
generate paired antiparticle vectors.

$$\text{PRNG}(g^{ba}) \rightarrow \overrightarrow{\nabla} = -\overrightarrow{\blacktriangle}$$



Pairwise Diffie-Hellman Key Agreement + PRNG Expansion

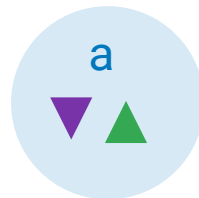
Secrets are scalars, but....

Use each secret to seed a pseudorandom number generator, generate paired antiparticle vectors.

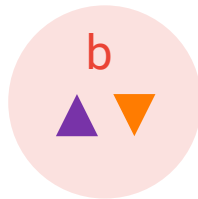
$$\text{PRNG}(g^{ba}) \rightarrow \overrightarrow{\nabla} = -\overrightarrow{\blacktriangle}$$

1. Efficiency via pseudorandom generator
2. Mobile phones typically don't support peer-to-peer communication anyhow.
3. Fewer secrets = easier recovery.

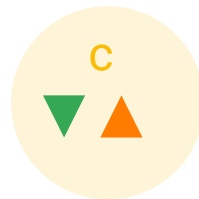
Alice

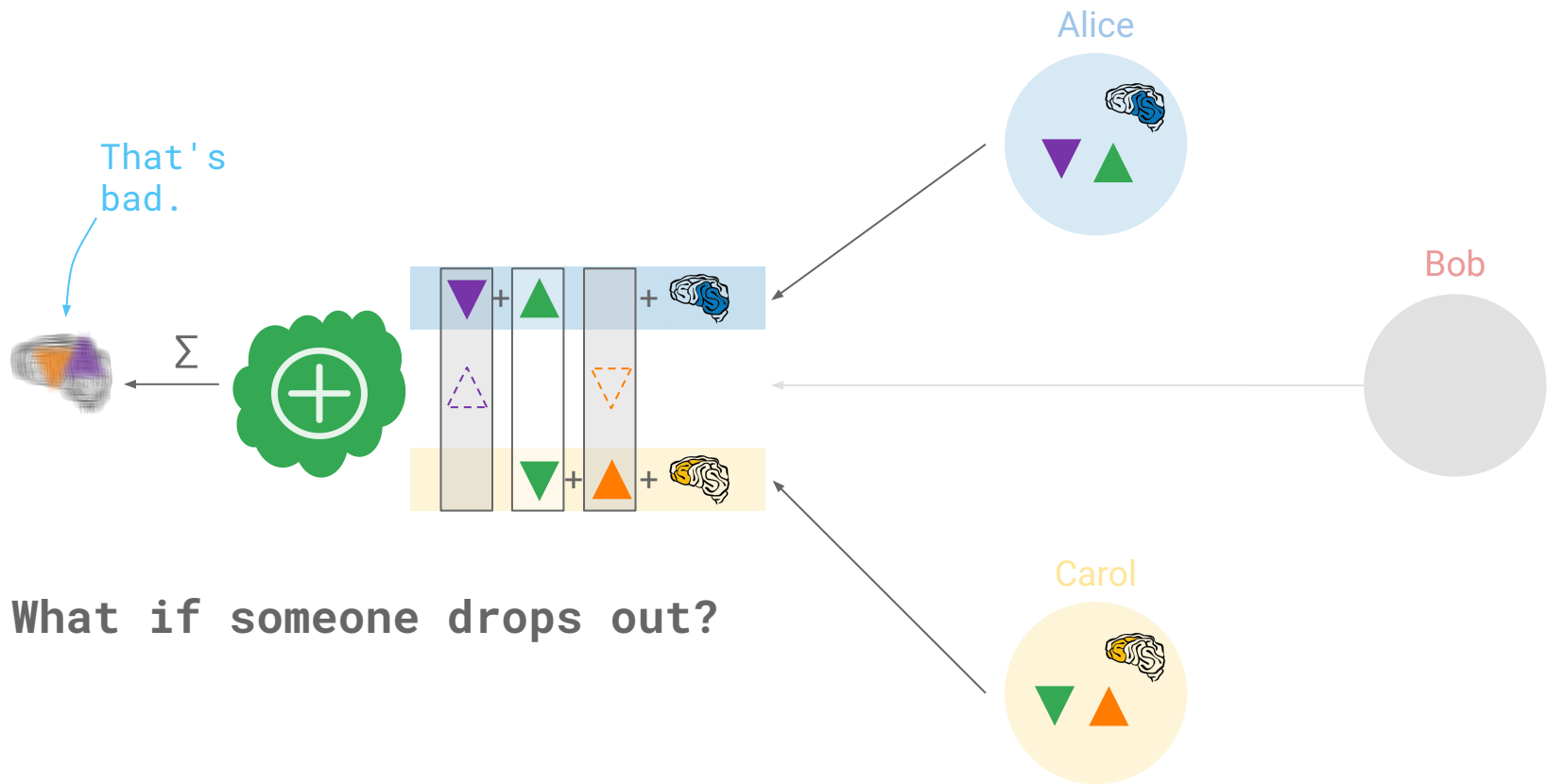


Bob



Carol



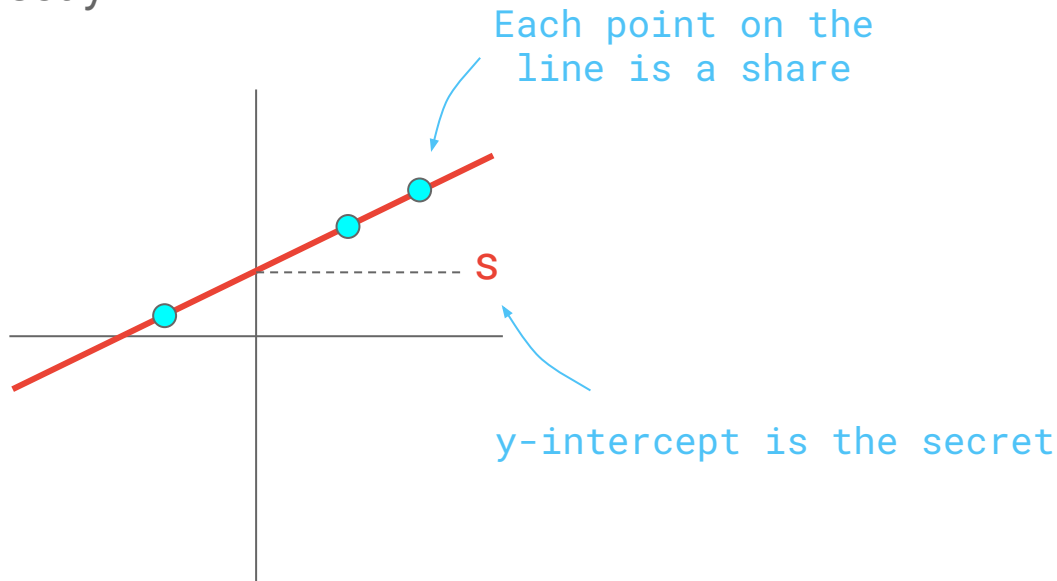


k -out-of- n Threshold Secret Sharing

Goal: Break a secret into n pieces, called shares.

- $< k$ shares: learn nothing
- $\geq k$ shares: recover s perfectly.

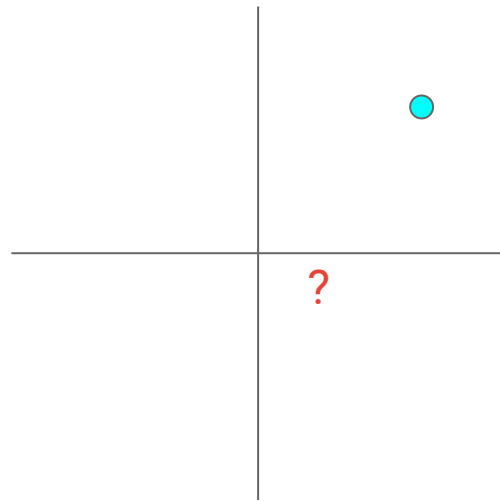
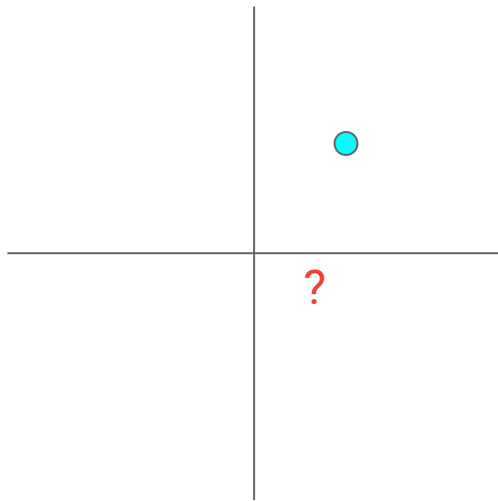
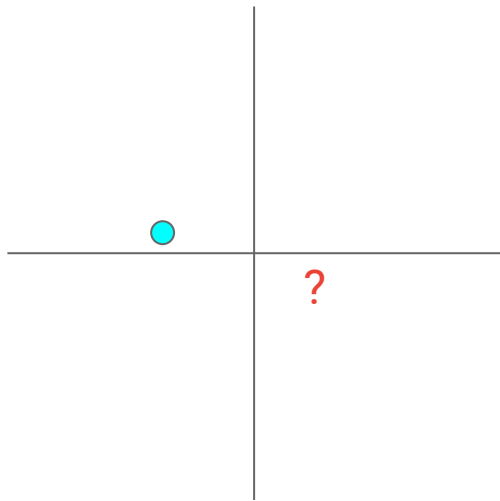
2-out-of-3 secret sharing:



k -out-of- n Threshold Secret Sharing

Goal: Break a secret into n pieces, called shares.

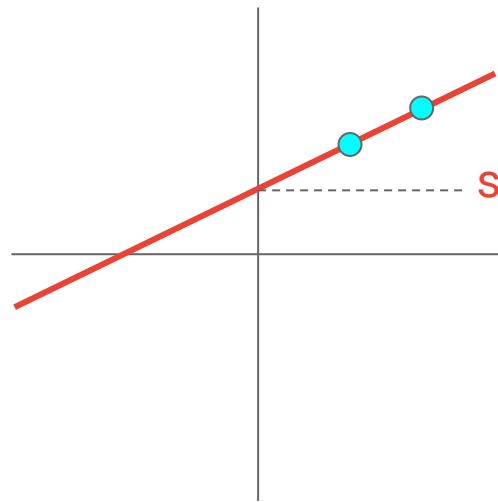
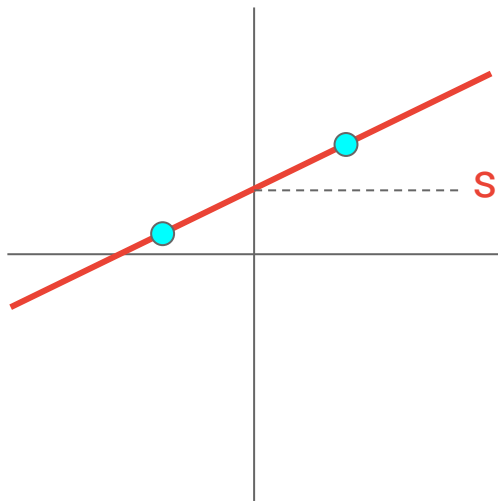
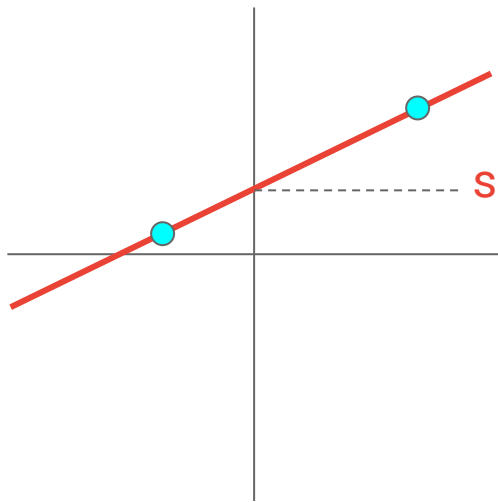
- **< k shares:** learn nothing
- **$\geq k$ shares:** recover s perfectly



k -out-of- n Threshold Secret Sharing

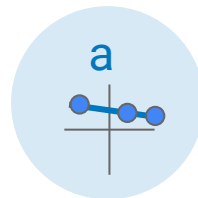
Goal: Break a secret into n pieces, called shares.

- **$< k$ shares:** learn nothing
- **$\geq k$ shares:** recover s perfectly

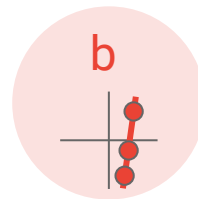


Users make shares of their secrets

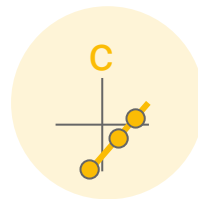
Alice



Bob

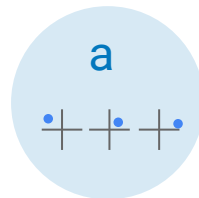


Carol

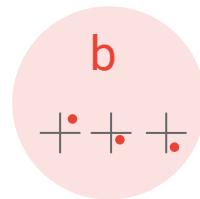


And exchange with their peers

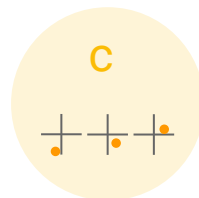
Alice



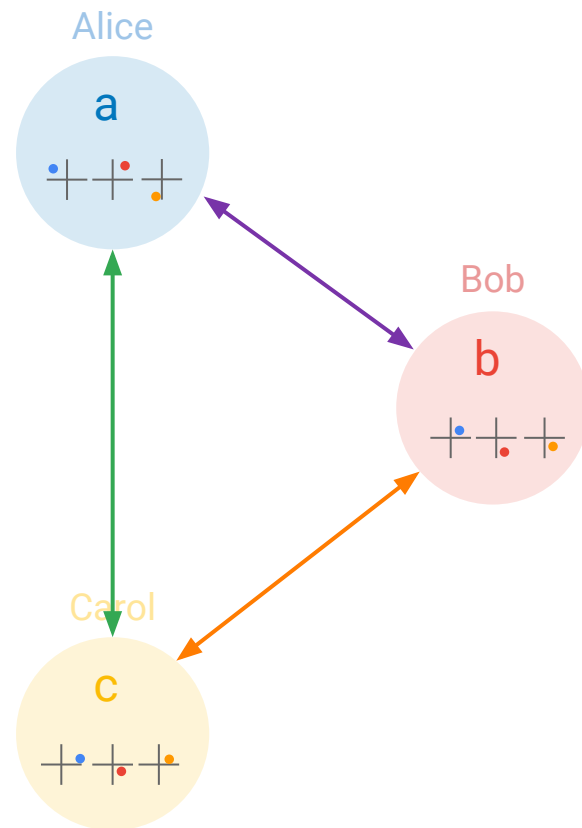
Bob



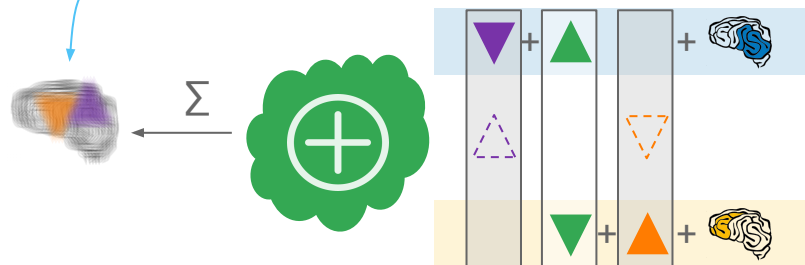
Carol



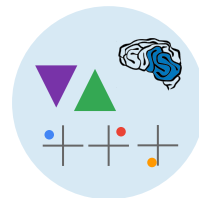
And exchange with their peers



That's bad.



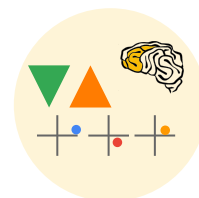
Alice

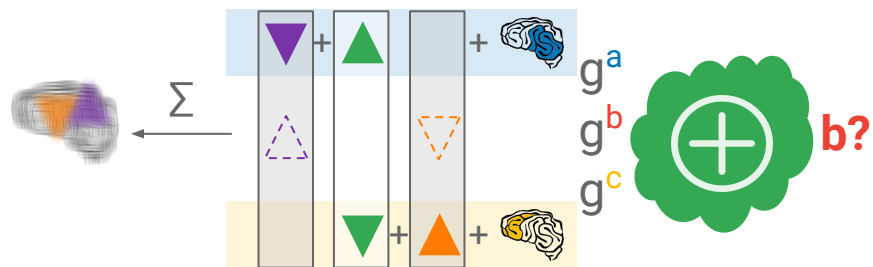


Bob

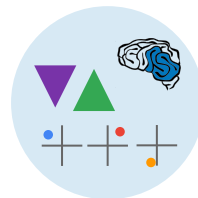


Carol

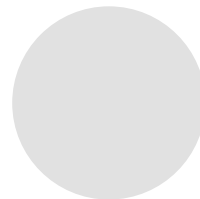




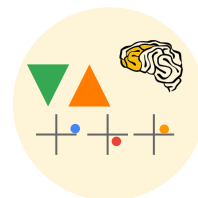
Alice

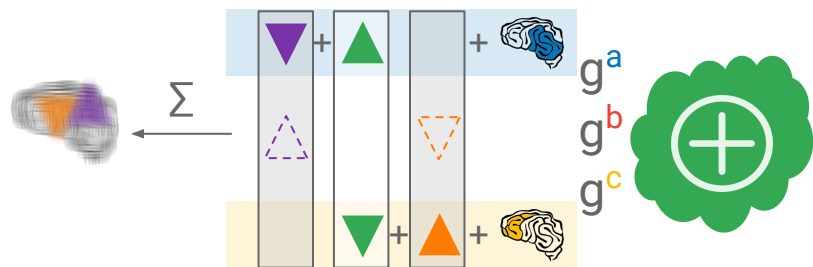


Bob



Carol

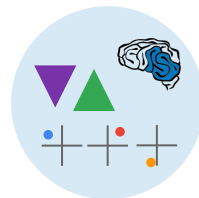




+

+

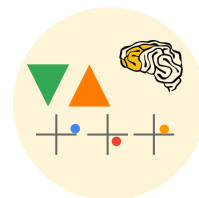
Alice

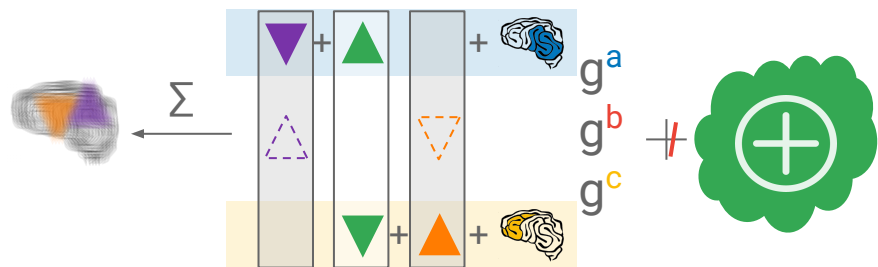


Bob

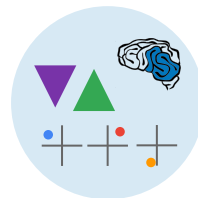


Carol





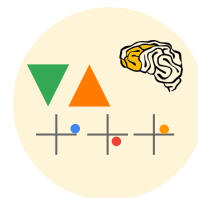
Alice

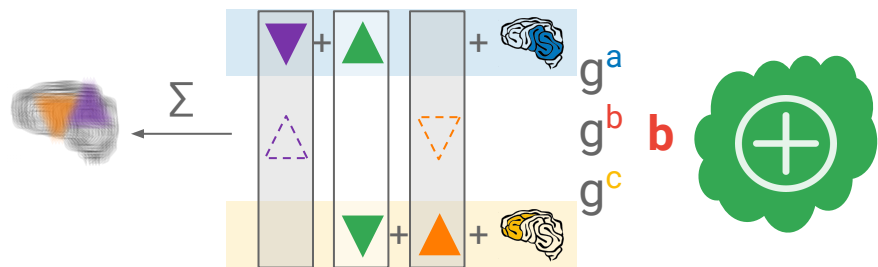


Bob

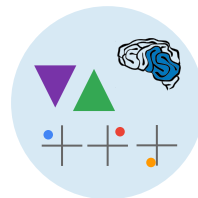


Carol





Alice

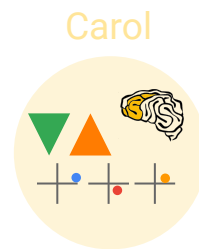
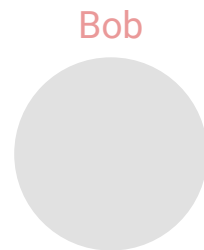
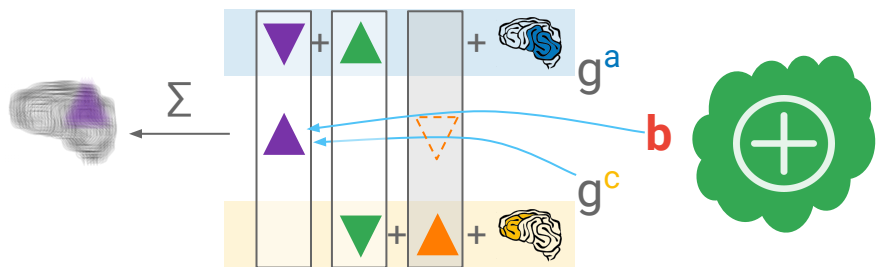


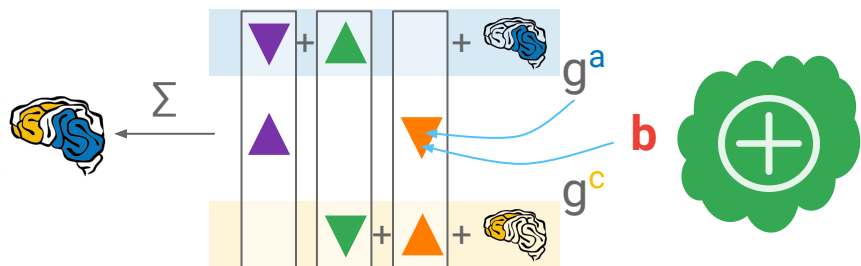
Bob



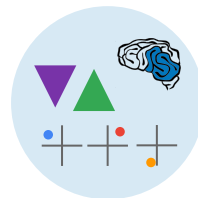
Carol



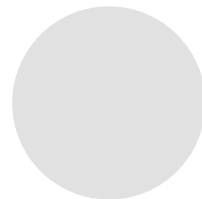




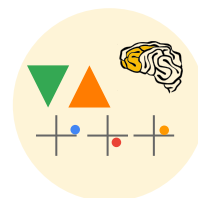
Alice

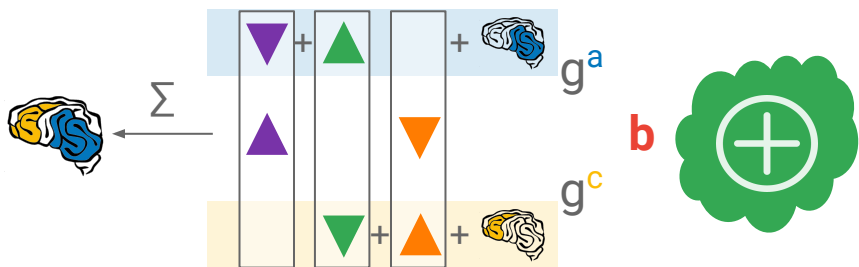


Bob



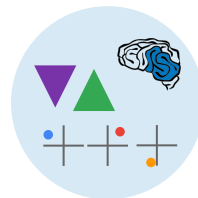
Carol





Enough honest users + a high enough threshold
 \Rightarrow server + dishonest users cannot reconstruct the secret.

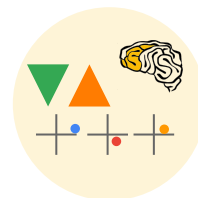
Alice

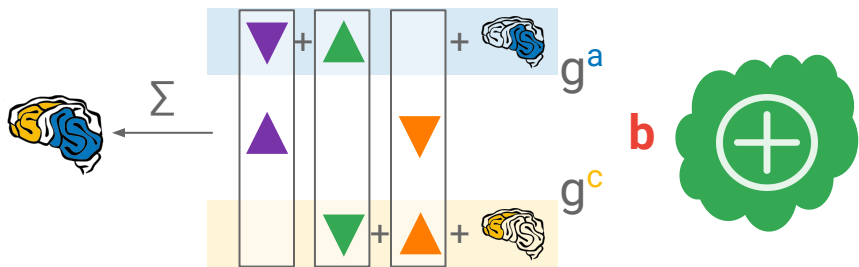


Bob



Carol



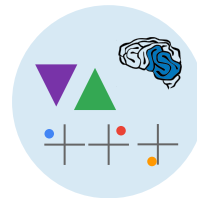


Enough honest users + a high enough threshold
 \Rightarrow server + dishonest users cannot reconstruct the secret.

However....

Google

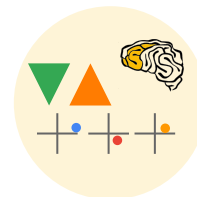
Alice

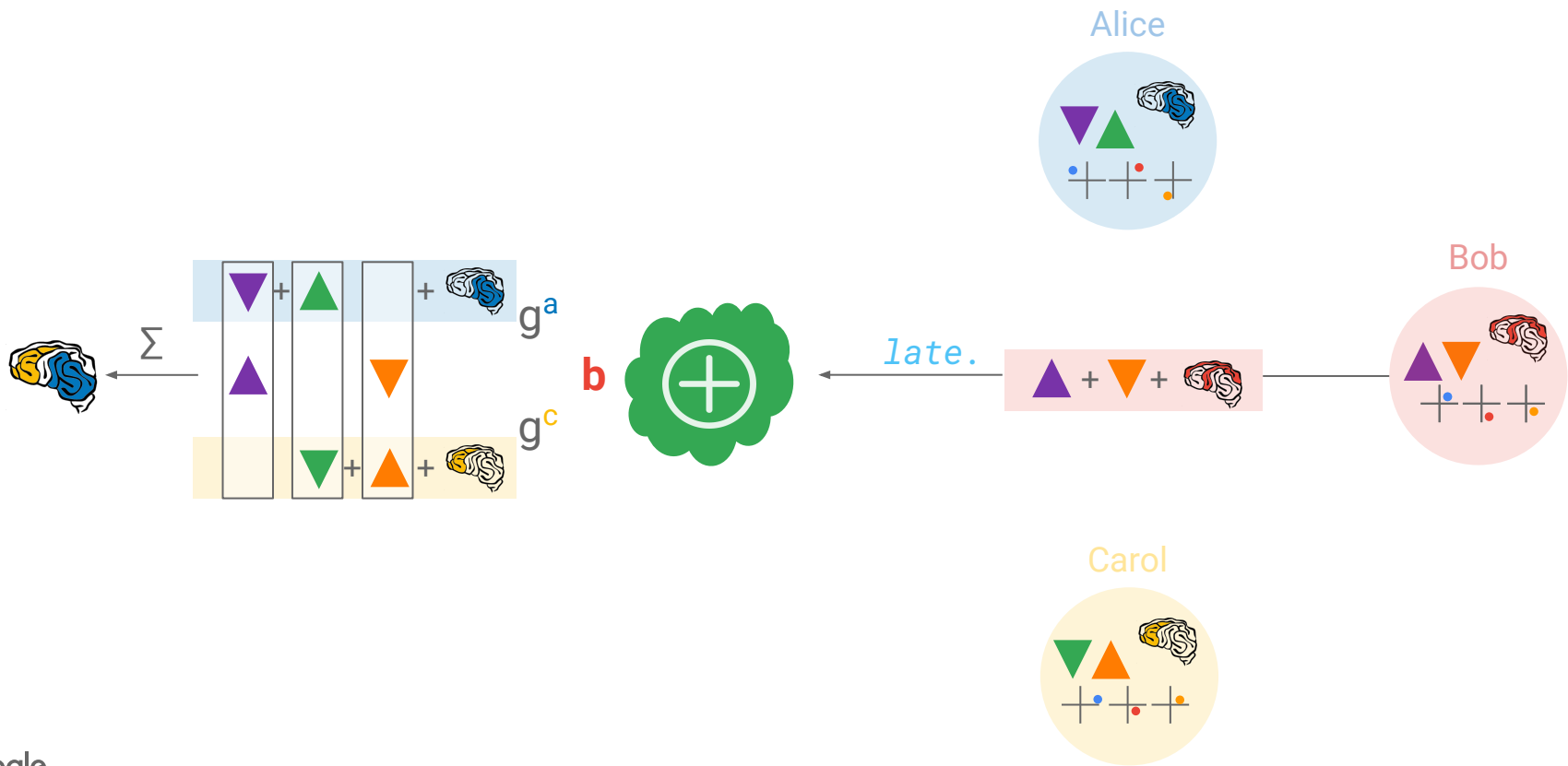


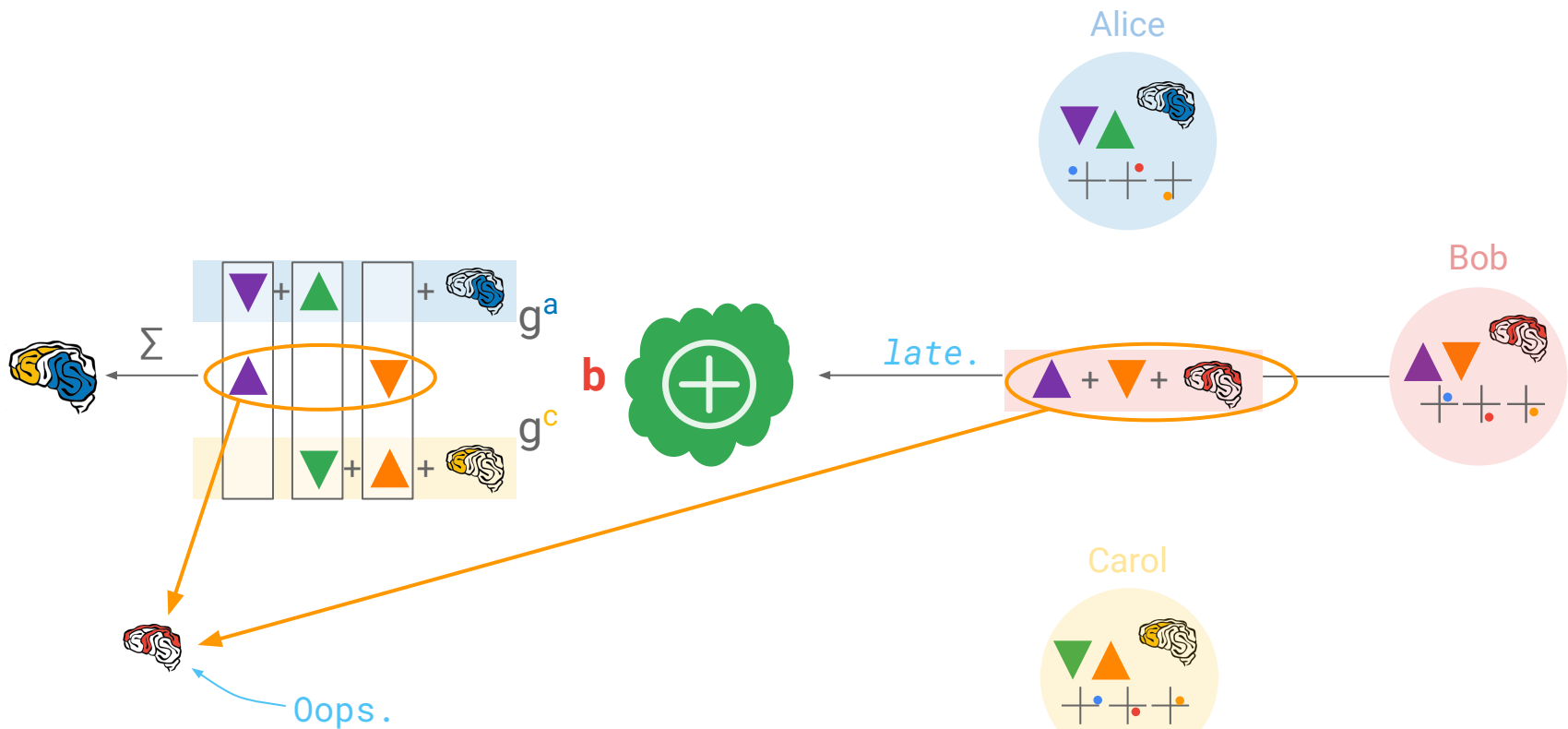
Bob

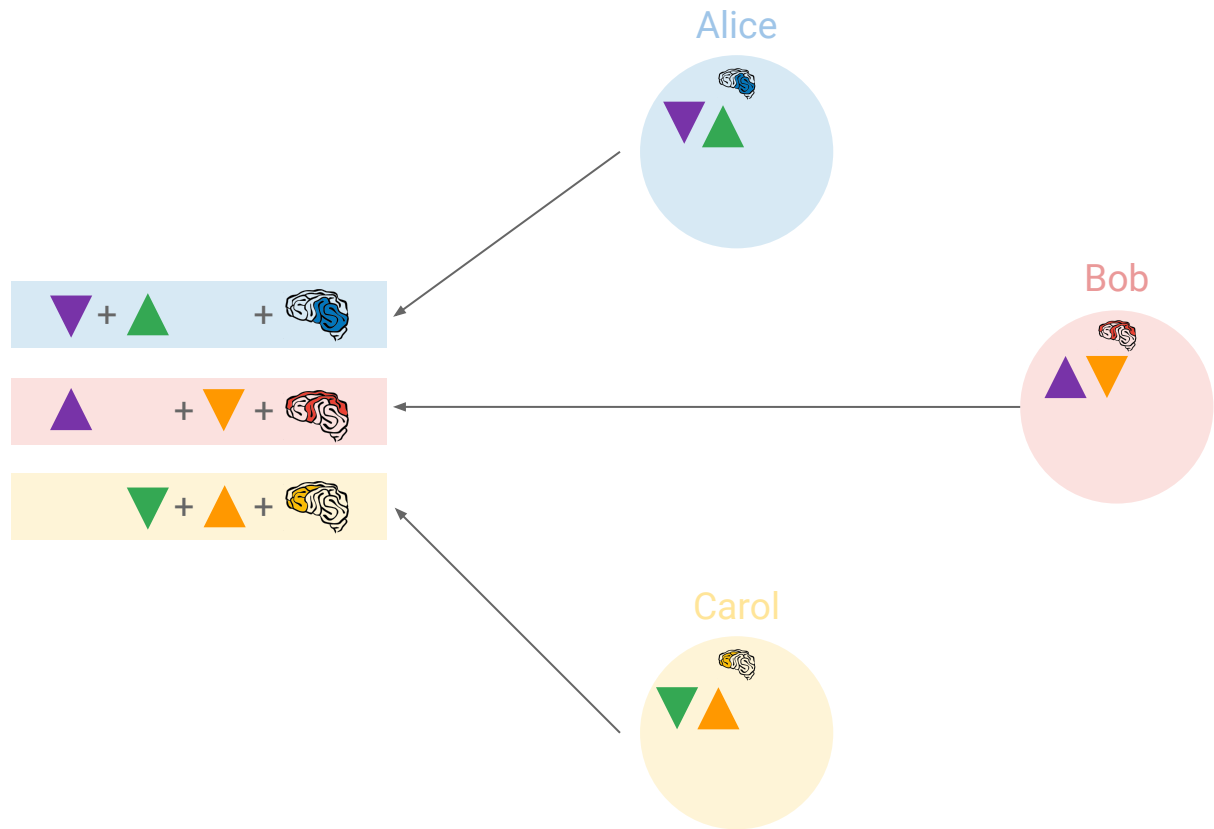


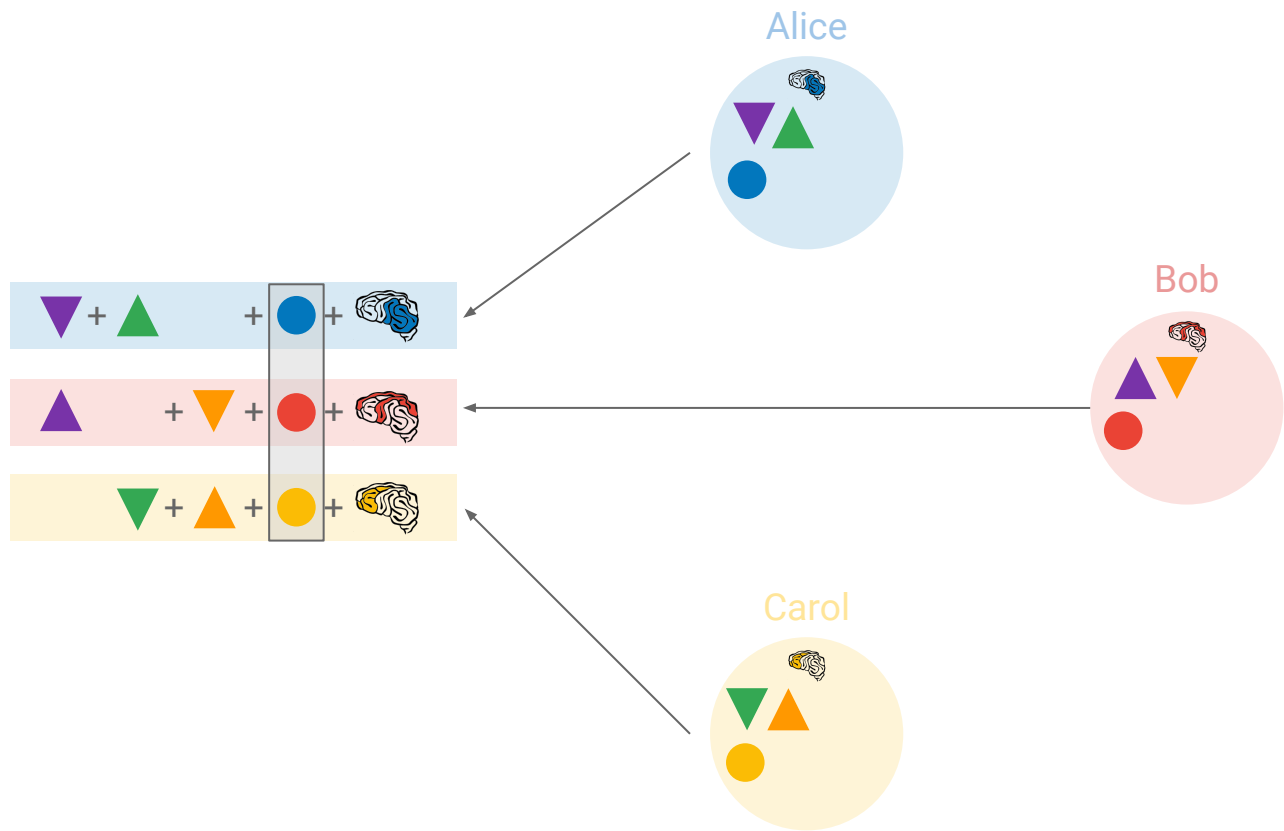
Carol



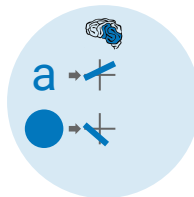




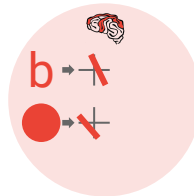




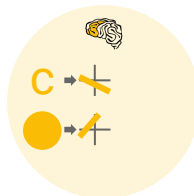
Alice

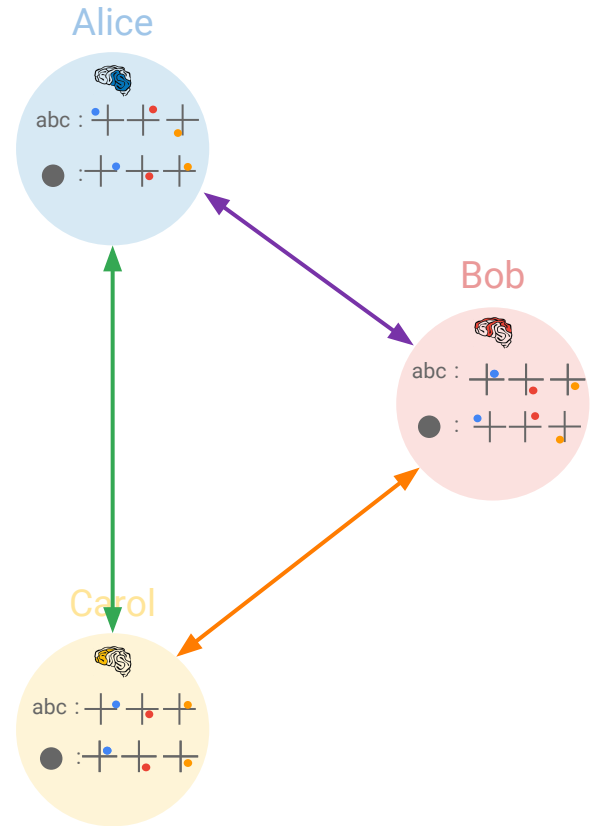


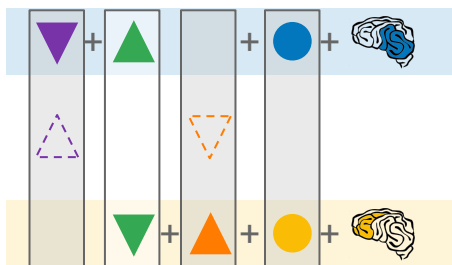
Bob



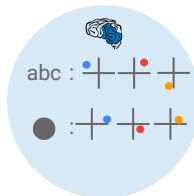
Carol







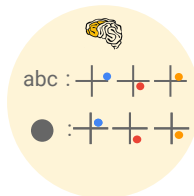
Alice

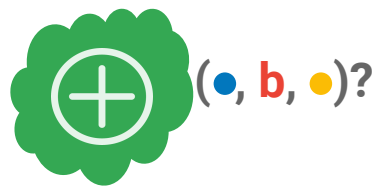
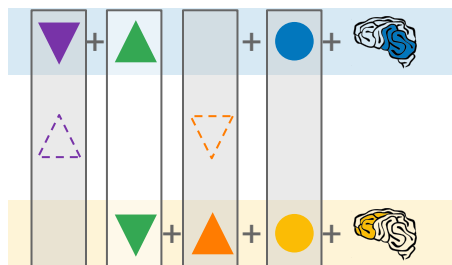


Bob

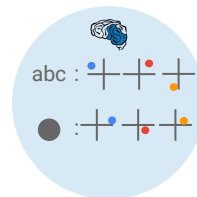


Carol





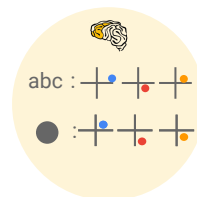
Alice

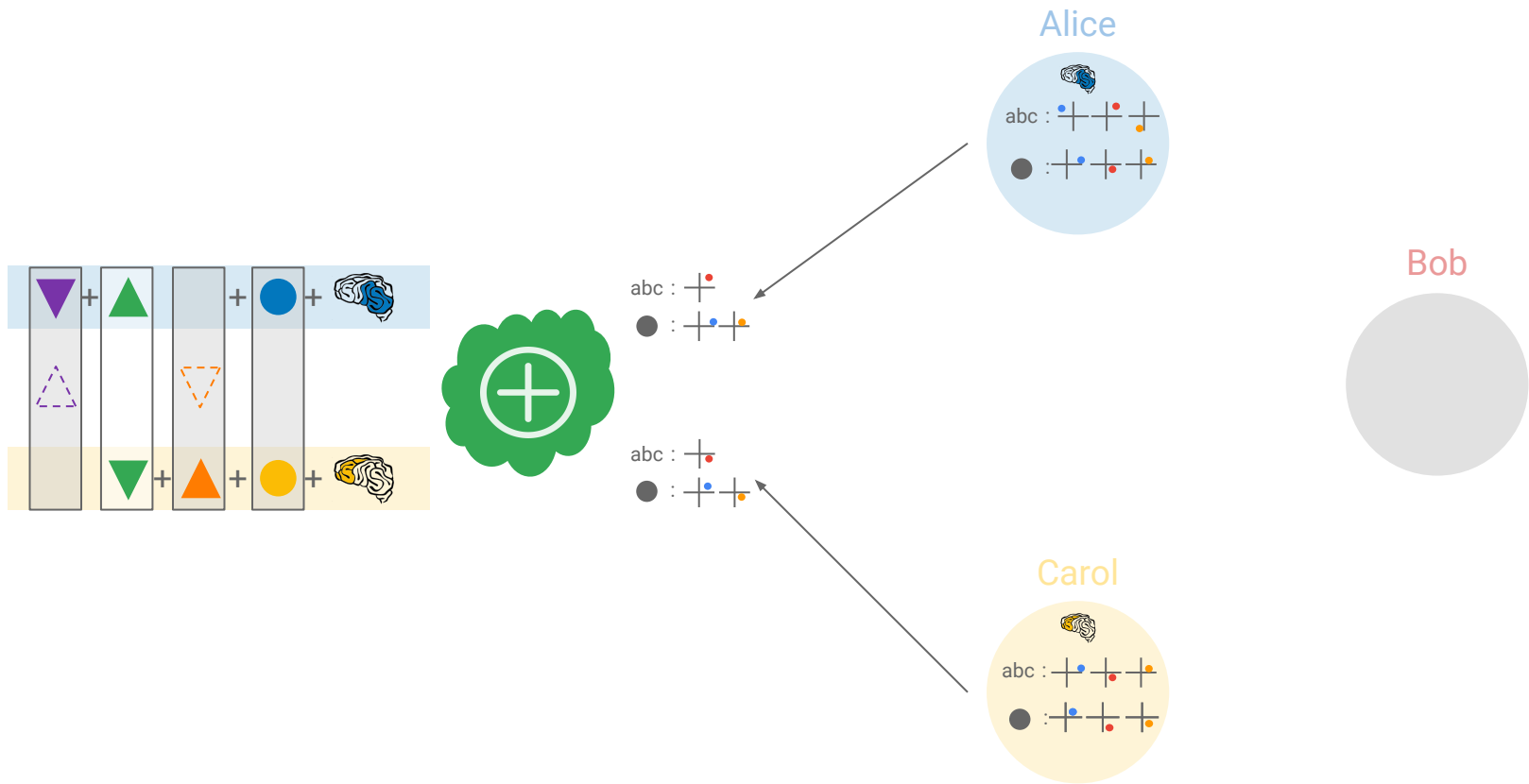


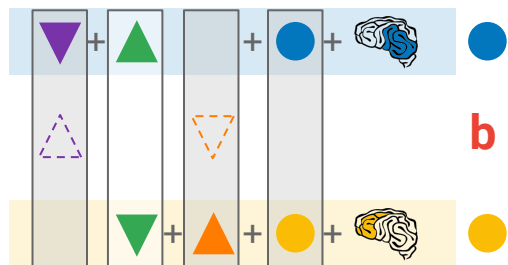
Bob



Carol



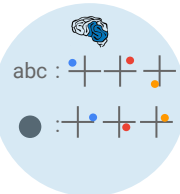




abc : +
 ● : +

abc : +
 ● : +

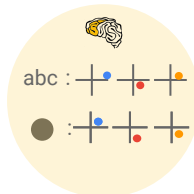
Alice

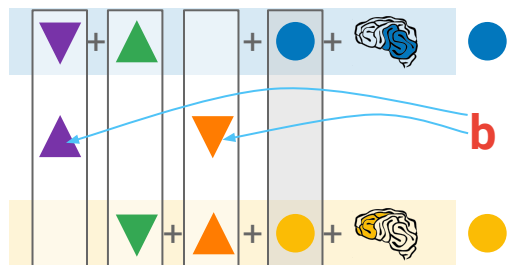


Bob



Carol

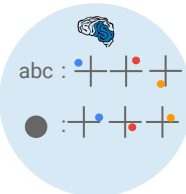




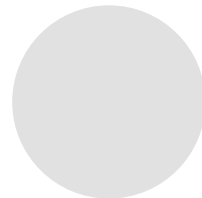
abc : +
● : +

abc : +
● : +

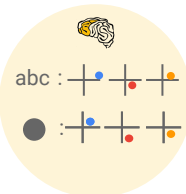
Alice

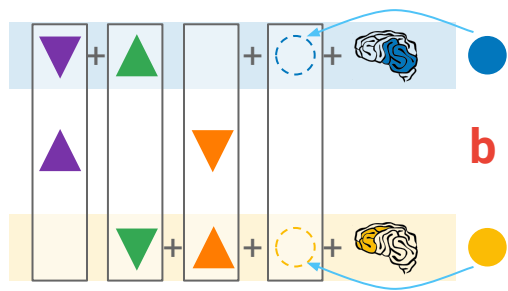


Bob



Carol

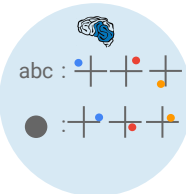




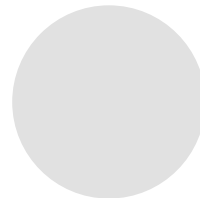
abc : +
● : +

abc : +
● : +

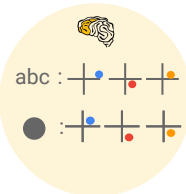
Alice

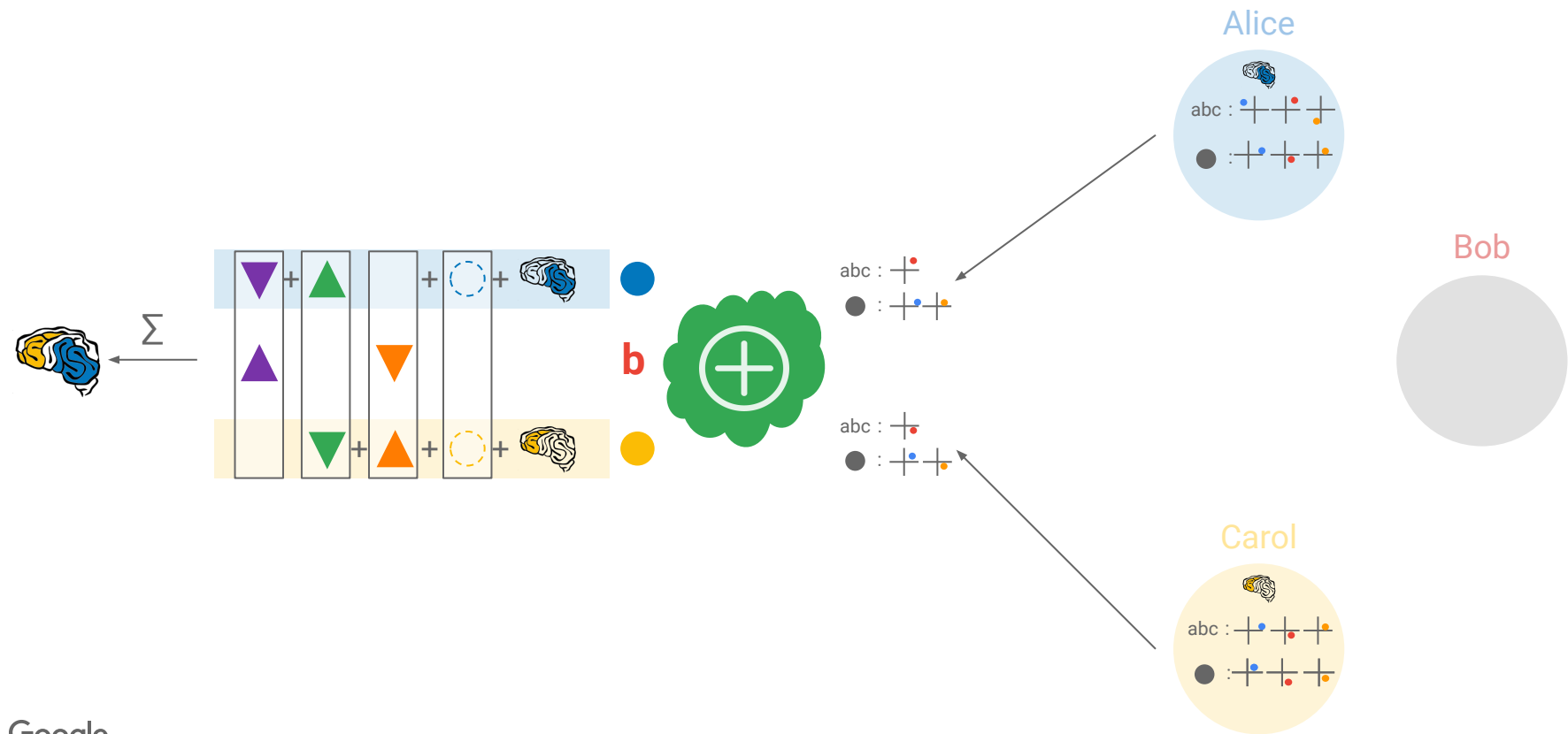


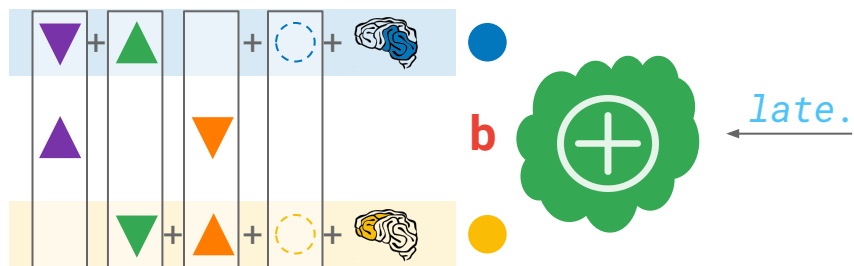
Bob



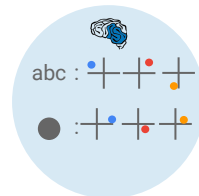
Carol



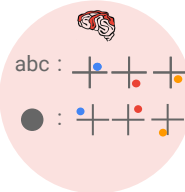




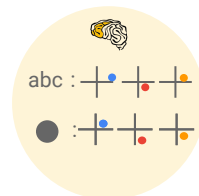
Alice

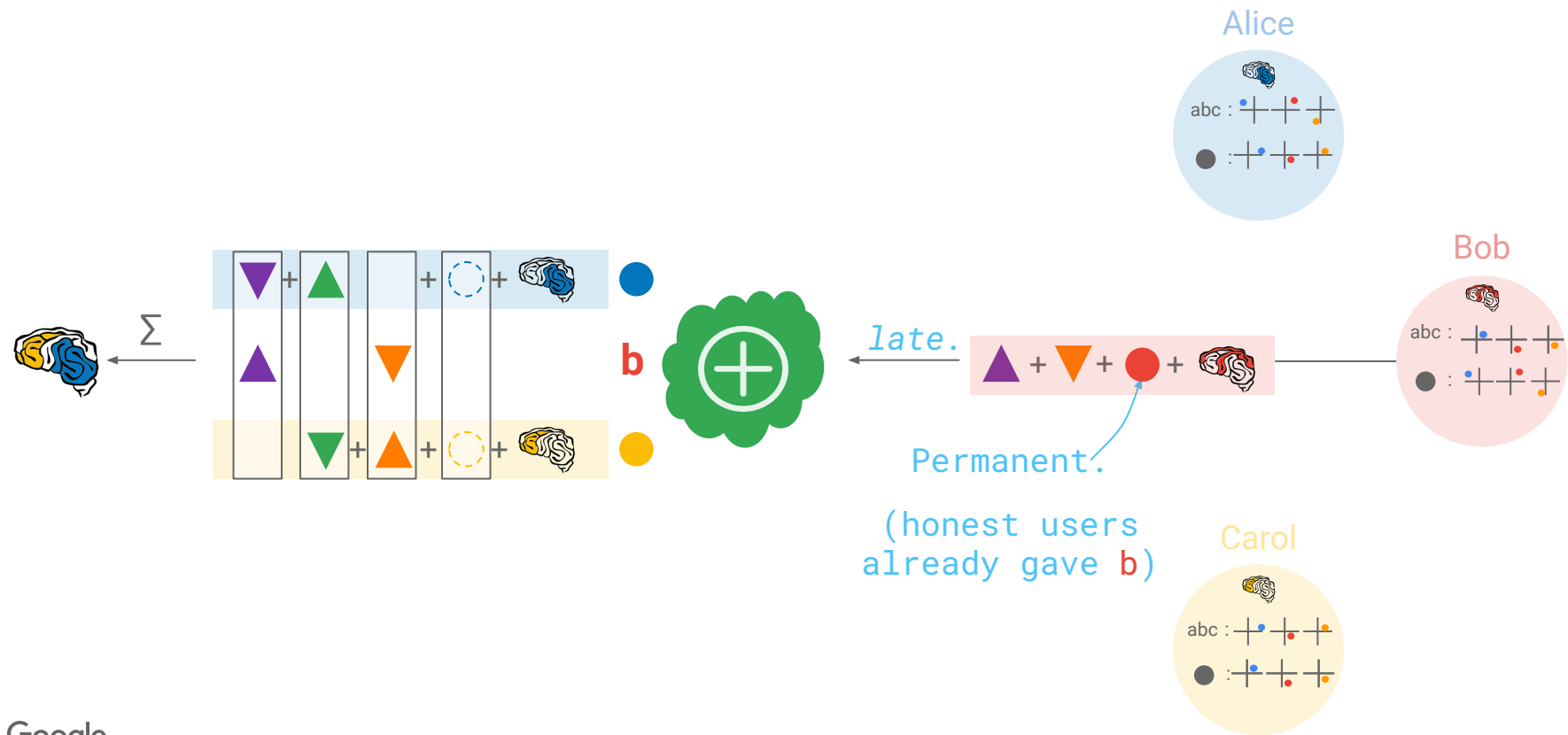


Bob



Carol





Security and Robustness

Implementer's Choice

Threat Model	Minimum Threshold	Minimum Inputs In Sum	Maximum Dropouts
Client-only Adversary	2	<i>threshold</i>	$n - \text{threshold}$
Server-only Adversary	$\lfloor n/2 \rfloor + 1$	<i>threshold</i>	$n - \text{threshold}$
Server-Clients Collusion ($\# \text{corrupt} < \lceil n/3 \rceil$)	$\lfloor 2n/3 \rfloor + 1$	$\text{threshold} - \# \text{corrupt}$	$n - \text{threshold}$

Security is against Honest-But-Curious or against Active adversaries

Protection against malicious server
requires Public Key Infrastructure
or honest Diffie-Hellman exchange

Secure: Provably secure against a wide range of honest-but-curious and active adversaries.

e.g.: secure for an adversary fully observing the server and having complete control of up to $\frac{1}{3}$ of the clients

Robust: Completes even if $\frac{1}{3}$ of clients drop out during protocol; secure no matter how many drop out.

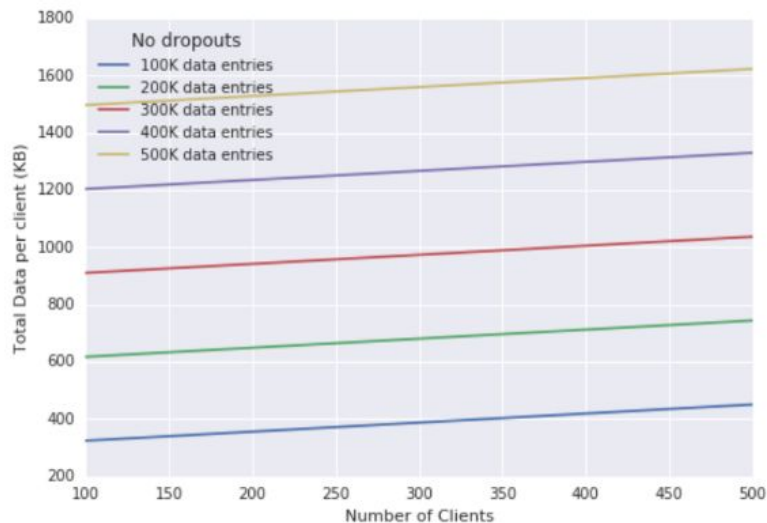
Communication Efficient:

# Parameters	Bits/Parameter	# Users	Bandwidth Expansion
$2^{20} = 1 \text{ m}$	16	$2^{10} = 1 \text{ k}$	1.73x
$2^{24} = 16 \text{ m}$	16	$2^{14} = 16 \text{ k}$	1.98x

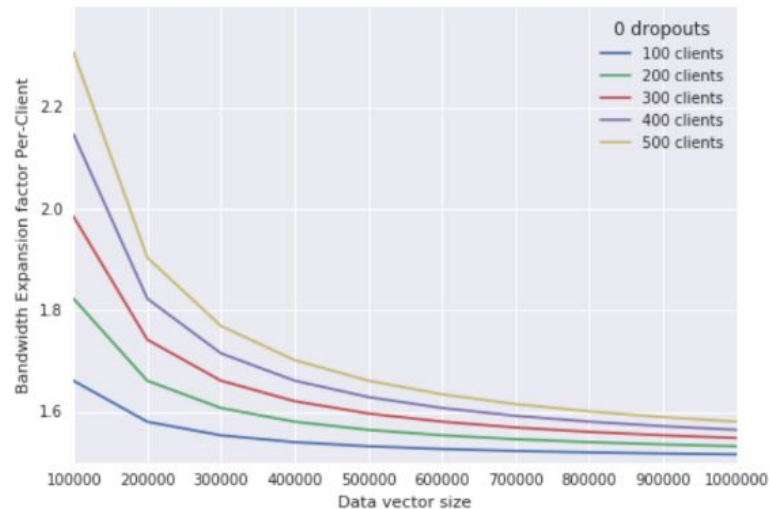
Performance

Communication per client:

Less than 2x expansion over sending in the clear



(c) Total data transfer per client, as the number of clients increases. Different lines show different data vector sizes. Assumes no dropouts.



(d) Total data expansion factor per client, as compared to sending the raw data vector to the server. Different lines represent different values of n . Assumes no dropouts.

A novel protocol for Secure Aggregation.

Our protocol *does not*:



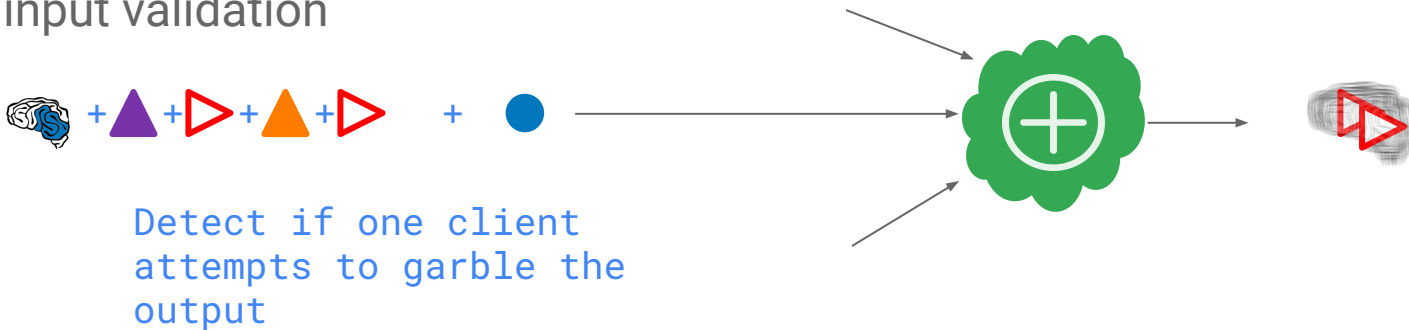
Transmit
a *lot* of data



Fail when
users drop out

Future Work

- Client input validation

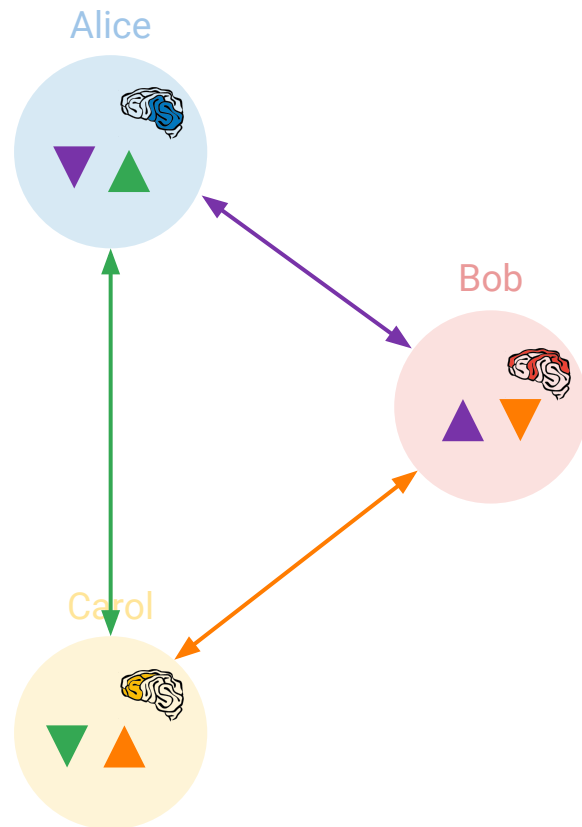


- Weakening “PKI” assumptions for malicious servers

Scaling exchange

As the number of clients becomes large, exchanging masks between all pairs becomes expensive:

- High communication
- High computation (PRNG expansions)



Scaling exchange

As the number of clients becomes large, exchanging masks between all pairs becomes expensive:

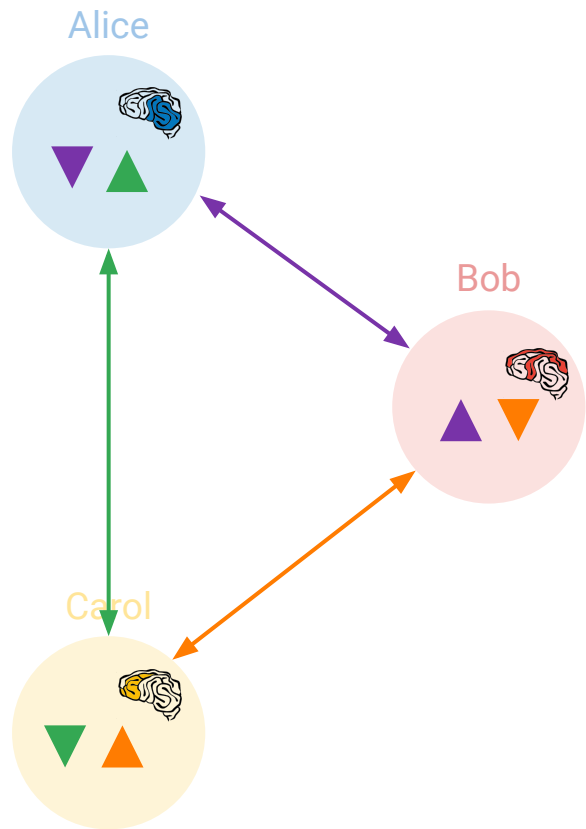
- High communication
- High computation (PRNG expansions)

Follow-up work* shows how to allow clients to communicate with a *logarithmic-sized subset* of other clients, while preserving

- Robustness to dropouts
- Security against malicious subsets of clients

*“Secure Single-Server Aggregation with (Poly)Logarithmic Overhead”

James Bell, K. A. Bonawitz, Adrià Gascón, Tancrede Lepoint, and Mariana Raykova



Further reducing computational complexity

- Clients need to perform many PRG expansions
- One idea: use a key-homomorphic PRG
- Allows adding together many keys, and doing a single PRG expansion
- Can greatly reduce the computational work per client, especially as the number of pairs grows.

Scaling the system: future questions

- The more clients, the longer the protocol may take
 - Higher variance in client response time
- Can we allow clients to come back later?
- Can we create a streaming cohort?
 - Continuous stream of users who join and leave the aggregation?

“Standard” cryptography and Key Management

Slides courtesy of Sophie Schmieg and Stefan Kölbl,
Presented at Real World Crypto 2023

Tink: What is it?

- Open-source cryptographic library
- Designed to be hard-to-misuse
- Born out of many, many product reviews
- Seeing common mistakes, errors in crypto use, ...



Hard-to-use interfaces (E.g. OpenSSL)

```
int EVP_EncryptInit_ex(  
    EVP_CIPHER_CTX *ctx, const EVP_CIPHER *type,  
    ENGINE *impl, unsigned char *key, unsigned char *iv);
```

```
int EVP_EncryptUpdate(  
    EVP_CIPHER_CTX *ctx, unsigned char *out,  
    int *outl, const unsigned char *in, int inl);
```

```
int EVP_EncryptFinal_ex(  
    EVP_CIPHER_CTX *ctx, unsigned char *out, int *outl);
```

But also, how to manage keys?

- How and where to store keys?
- How to make them available to services reliably?
- How to rotate them?
- How to enable changing schemes (e.g. to post-quantum-secure)?

“Fancy Crypto turns every problem into a Key Management problem”

- Sophie Schmieg, ISE Crypto Lead

Crypto Agility at Google

Enforce best practices



Don't burden the user



Reliability



Best practice – Key rotation

Key rotation should happen **automatically**

- Forward secrecy.
- Enables speedy recovery from compromise at low operational risk.
- Simplifies switching keys \Rightarrow Transition to post-quantum crypto

In practice **hard** to enforce:

- Reliability risks

User Perspective – Setup

Customer responsibility:

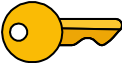


- Key type (e.g. AEAD), Key format (e.g. AES-GCM)
- TTL of ciphertext (e.g. 90 days, 1 year, ...)
- Cache time for a key (e.g. 1 day)

KMS takes care of:

- Generating, distributing and rotating key material.

User Perspective – Using keys

Two simple steps:

- 1) Retrieve key from KMS: `key={  ,  ,  }`
 - Key usage authenticated and authorized based on service identities.
 - Binary authorization ([BAB](#)).
- 2) Use key in cryptographic library:
 - `key.encrypt("message")`

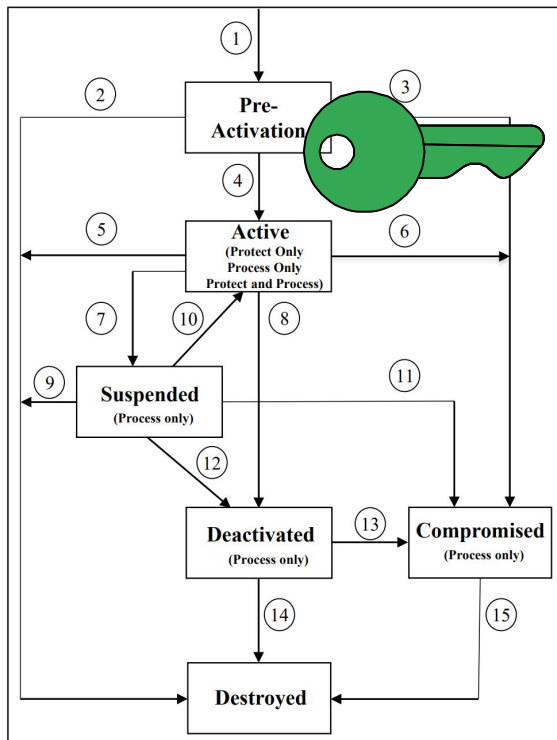
User Perspective - Using keys

Abstraction in cryptographic library (Tink) handles:

- Which key inside the keyset to use.
- Refreshing key material on a regular schedule.
- Keyset can have keys of different format.



Lifetime of a key



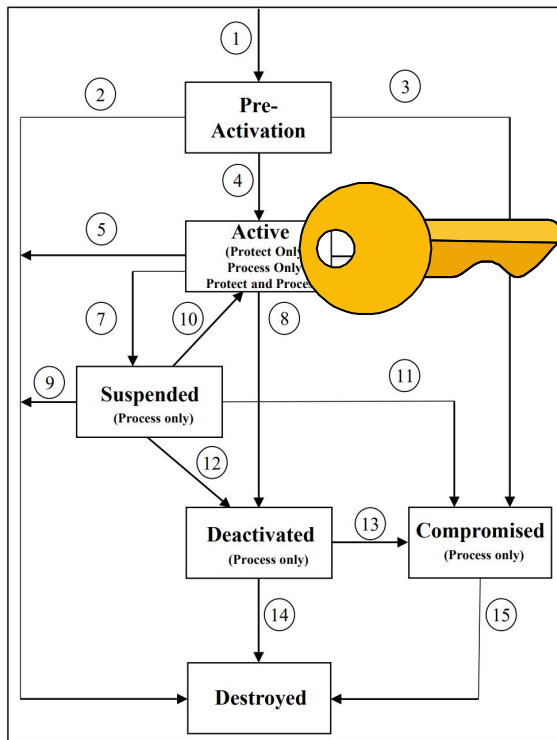
Pre-activation:

- Key material has been generated.
- Not actively being used.

Problem:

When is it safe to make the key active?

Lifetime of a key



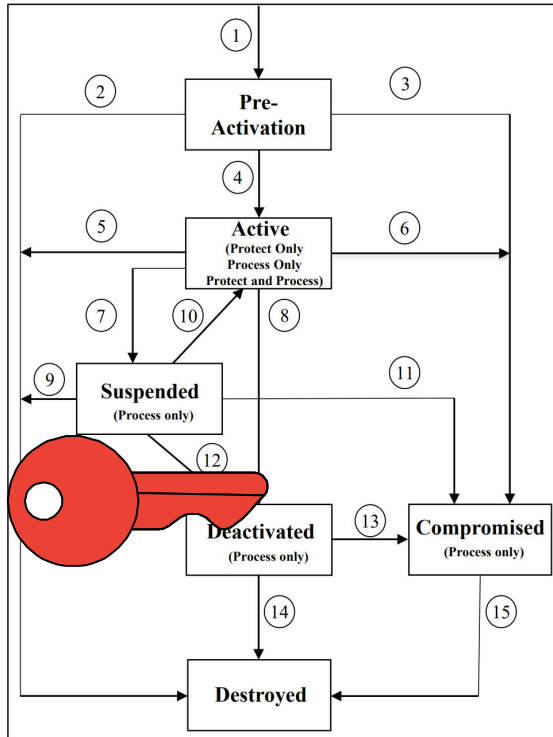
Active

- Key used for encryption or signing.

Problem:

Need to ensure that all other users have the key.

Lifetime of a key

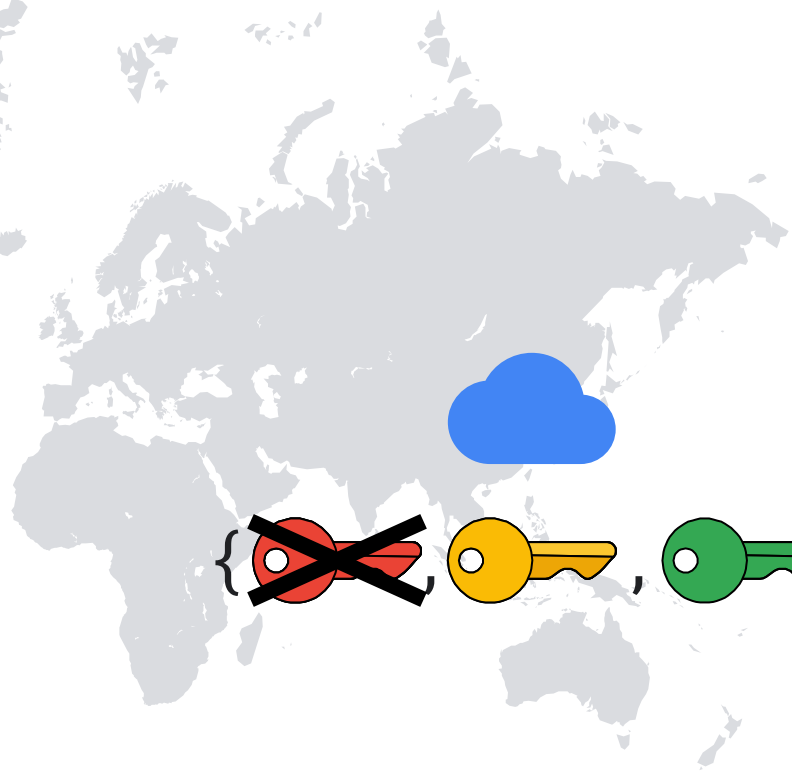
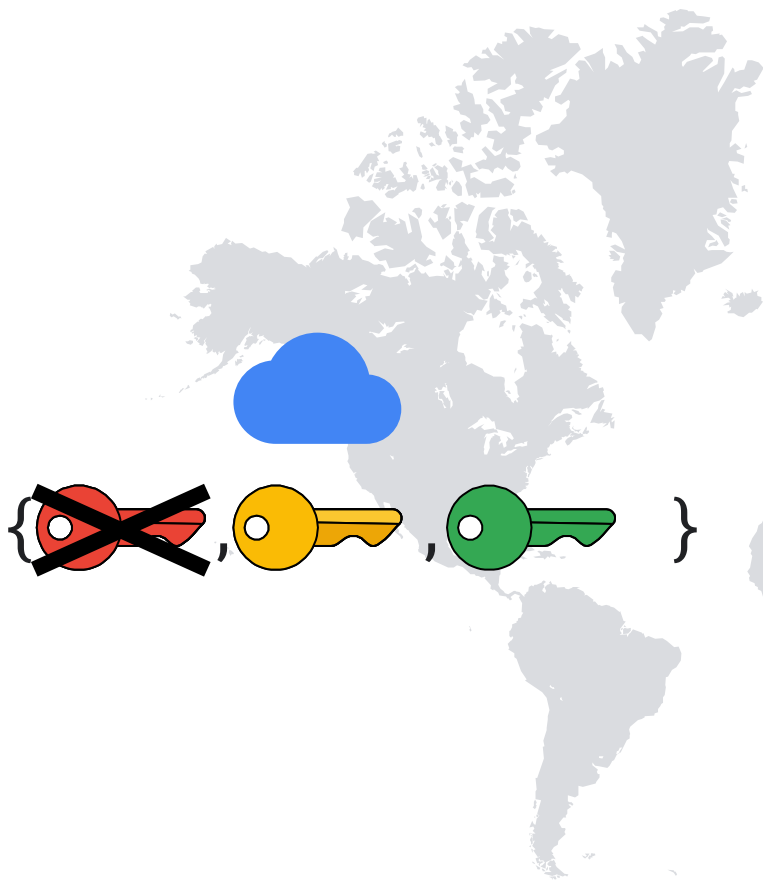


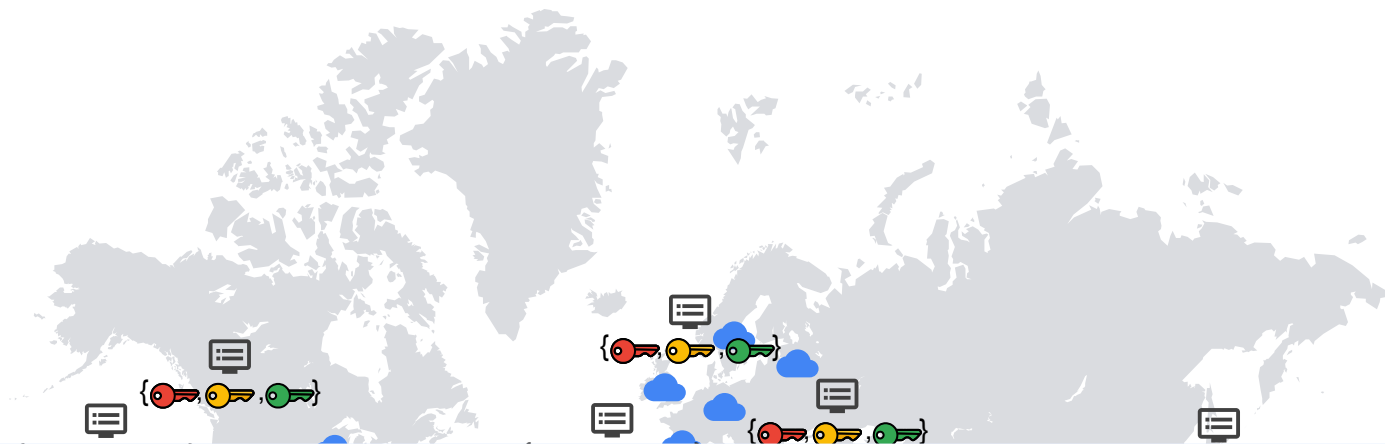
Suspended / Deactivated

- Key only used for decryption / verification.

Problem:

When can we delete that key?





Key management \Leftrightarrow a large scale distributed system problem.



What can go wrong in practice?

Service A

1) Fetch key = {🔑, 🔑, }

2) `signed_url = key.sign("url")`

User

1) Accesses `signed_url`

Service B

1) Fetch key = {🔑, , }

2) Receives `signed_url`

3) `key.verify(sig, "url")`

Verification fails

Scale of the problem

Google has thousands of teams with:

- Different release schedules.
- Different use cases and needs.
- Usage from highly infrequent to billion queries / day.

Need solution which can handle all these different factors.

Monitoring in the cryptographic library

Goals

- Do not burden users to keep track of their key usage.
- Add minimal overhead to cryptographic operations.

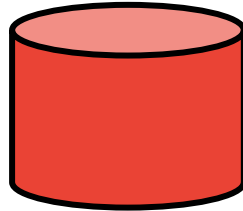
⇒ Horizontally monitor usage of cryptographic key material within our libraries.

Monitoring in the cryptographic library

Latency



Large data
volume



Integration in
crypto
libraries



Monitoring in the cryptographic library

Service A

```
...  
std::unique_ptr<Key> key = kms.FetchKey("urlsigner");  
std::string sig;  
sig = key->Sign("msg");  
...
```

Service B

```
...  
key = kms.fetchkey('urlsigner')  
sig = key.verify('msg', sig)  
...
```



KeyName	Service Name	API	Key Version	...
urlsigner	A	sign	3	...
urlsigner	B	verify	3	...

Monitoring in the cryptographic library

Allows to have reliable key rotation by the KMS

- Monitoring provides data points to inform key rotation:
 - When is it safe to make key active?
 - When can we delete a key?
- Notify teams if they violate their contract.
- Can block key deletion if it is still actively being used.

Monitoring in the cryptographic library

Allows us to:

- Enforce best practices across our services.
- Find keys which are used beyond their security bounds (e.g. GCM keys encrypting more than 2^{32} messages).
- Find usage of legacy keys and guide deprecation.
- Get usage data on migration from one algorithm to another...

Takeaways

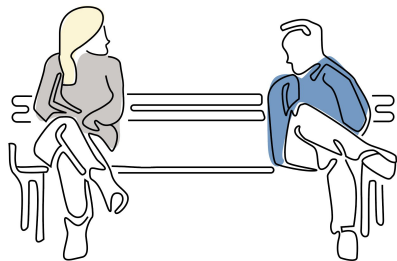
- Reliably managing keys at scale is challenging
- Stakes are high: no key, no services
- Not thinking ahead accrues heavy technical debt in the future
 - If you can't rotate keys, hard to do it when you have to

Thank you!

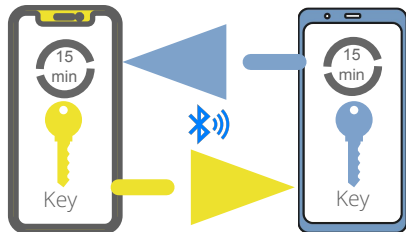
Questions?

ENPA

Two strangers, Alice and Bob, having a long conversation.



Their phones exchange non-identifiable Bluetooth beacons, which change frequently.

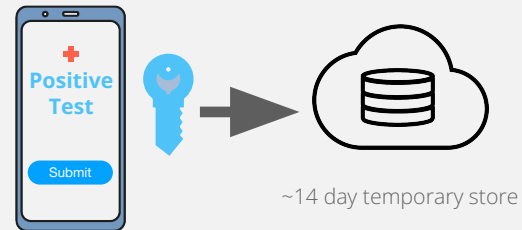


A few days later...

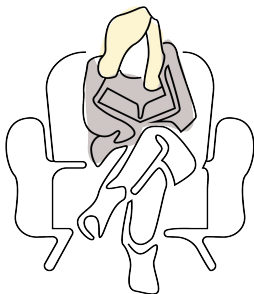
Bob is positively diagnosed for COVID-19.



With Bob's consent, his phone uploads the last 14 days of his Bluetooth exposure keys to the server.



Alice continues her day unaware she had been near a potentially contagious person.



Alice's phone periodically match against non-identifiable downloaded beacons of COVID-19 positive persons in her region.



A match is found

Google



Sometime later...

Alice receives a notification on her phone.



ALERT: You have recently come in contact with someone who has tested positive for Covid-19

Tap for more information -->



The notification includes information about what to do next.



Additional information and guidance is provided by the public health authority

Exposure Notifications: Privacy Principles

Opt-In

Required for enabling Exposure Notification

Required prior to uploading exposure keys to the Health Authority Key Server

No Linkable or Persistent Identifiers

Local observers should not be able to track users who never report positive

No Centralized Social Graph

Reported keys are downloaded and all matching for scanned Bluetooth beacons for any exposure events, are done on device

Problem:

How to get feedback to tune epidemiological parameters and evaluate effectiveness while upholding the privacy principles?

How many notifications are displayed per reported diagnosis?

How many people who received a notification end up reporting a positive diagnosis?

If we raised the risk threshold, how many more people would receive notifications?

If we lowered the risk threshold, how many people would be infected but not receive a notification?

Exposure Notifications Private Analytics Goals

Upholding Exposure Notification Privacy Principles

Aggregated Metrics

Individual contributions are not available and cannot be inferred

For Health Authorities

Only the Health Authorities have access to the aggregated metrics

Apple and Google servers don't learn individual contribution or aggregated metric

Robustness

No single user should be able to skew the result

Easy to Use and Understand

Provide an easy to use web portal to help health authorities make informed decisions about Exposure Notifications

Common Solution across Google and Apple

Agenda

Exposure Notifications (EN)

Exposure Notifications Private Analytics (ENPA)

Instantiation and Deployment of ENPA

Challenges and Recommendations

Prio

Prio: Private, Robust, and Scalable Computation of Aggregate Statistics

Henry Corrigan-Gibbs, Dan Boneh, NSDI 2017

Properties

- Multiparty aggregation system
- Protect individual user contributions
- Can be combined with differential privacy
- A malicious user can at most lie about their input

Privacy-preserving Firefox telemetry with Prio

Henry Corrigan-Gibbs
(EPFL → MIT CSAIL)

In collaboration with: Dan Boneh (Stanford),
Gary Chen, Steven Englehardt, Robert Helmer, Chris Hutten-Czapski,
Anthony Miyaguchi, Eric Rescorla, and Peter Saint-Andre (Mozilla)

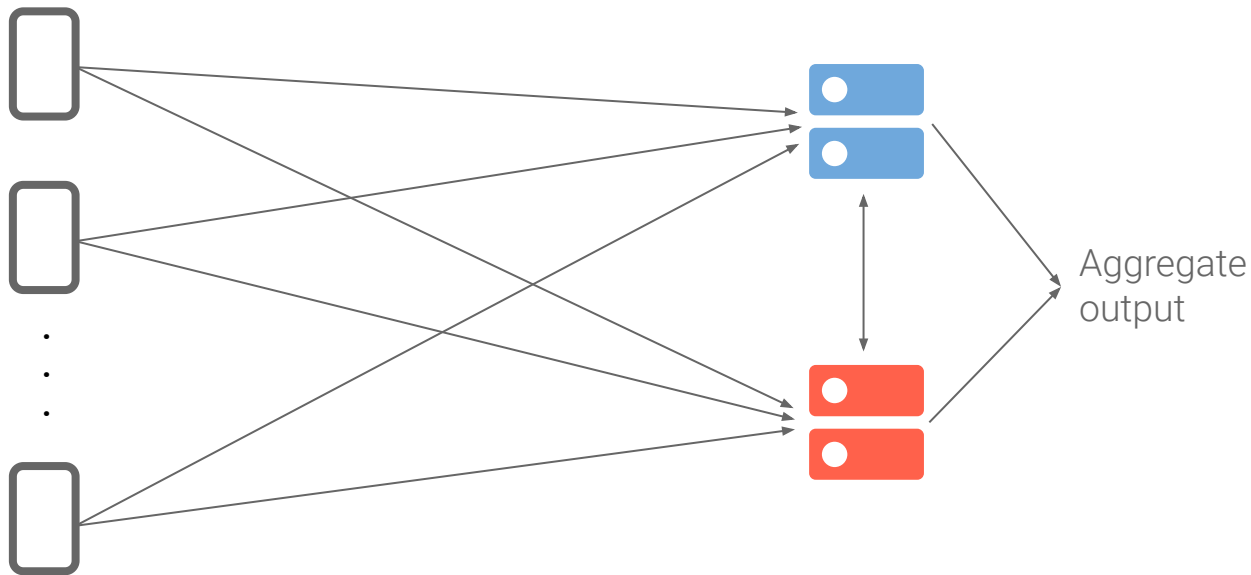
Real World Crypto 2020

<https://rwc.iacr.org/2020/slides/Gibbs.pdf>

Prio System Architecture

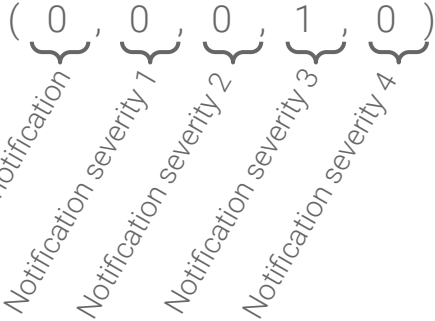
User Devices

Aggregation Servers



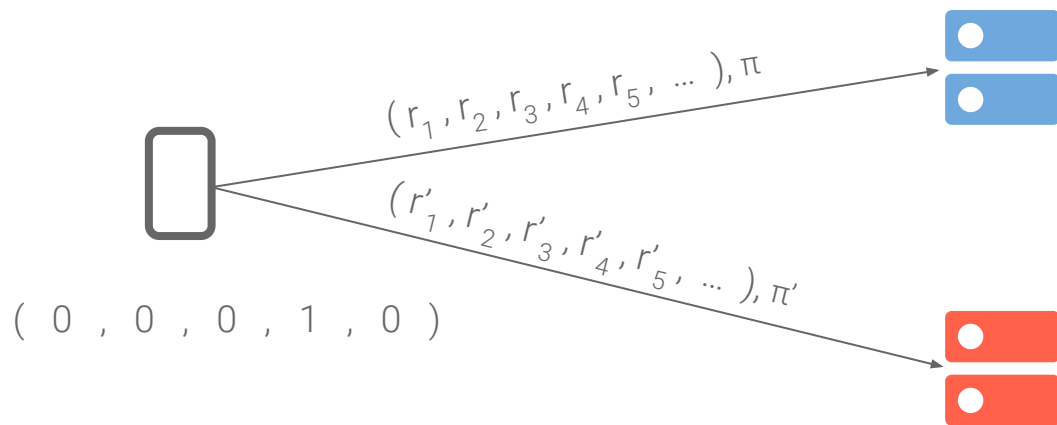
Non-collusion
assumption
among servers

Prio User Input



Fixed-size
bit-vector
representing the
metric of interest

Prio Secret Sharing



Secret share the
input vector and
prove that it is a
bitvector

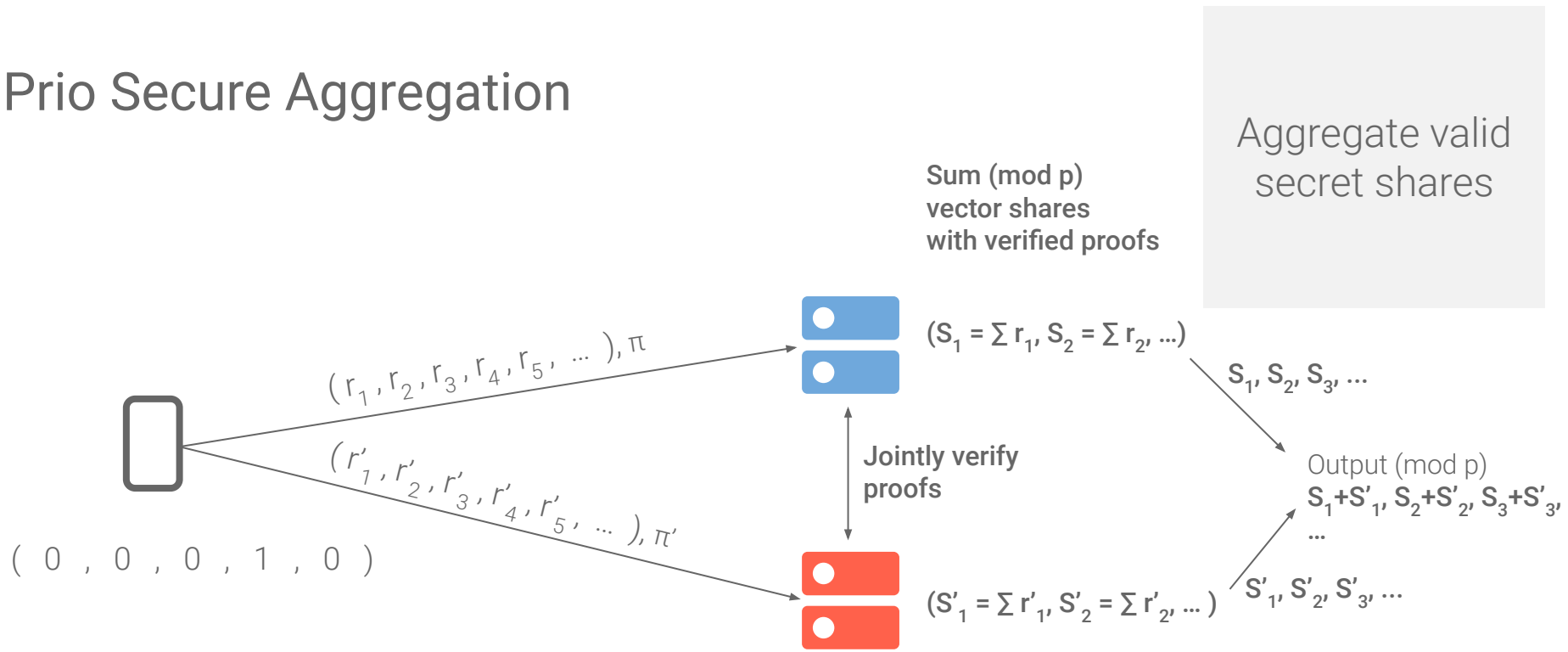
Compute secret shares

r_i random in \mathbb{Z}_p , $r'_i = b_i - r_i \pmod{p}$

Compute validity proofs: π and π'

Show the input vector is binary: $b_i(1-b_i) \equiv 0$

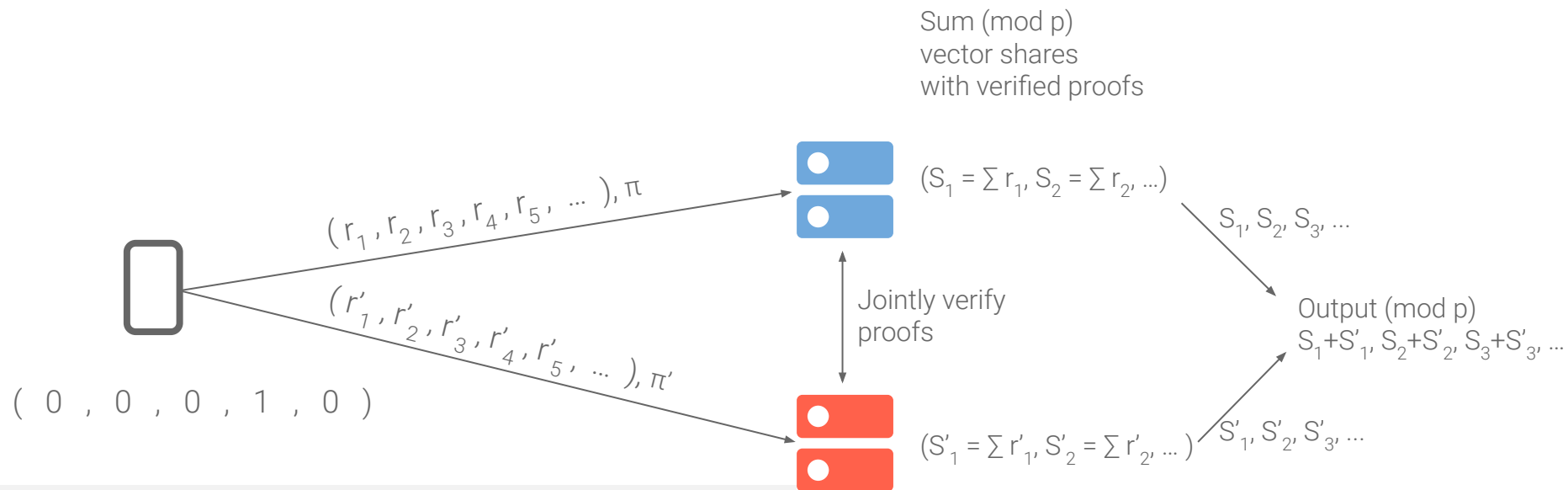
Prio Secure Aggregation



Compute secret shares
 r_i random in Z_p , $r'_i = b_i - r_i \pmod{p}$

Compute validity proofs: π and π'
 Show the input vector is binary: $b_i(1-b_i) \equiv 0$

Differential Privacy in Prio



Flip each input bit with small probability α
apply amplification by shuffling to determine DP guarantees

Compute secret shares

Compute validity proofs

Private Analytics: Meeting Design Goals

Data Privacy, Data Minimization, Data Integrity

Prio with non-colluding servers: [aggregated and differentially private data](#)

Well-defined metrics to [answer health authorities questions](#)

Validity proofs, device attestation

Efficiency

One-shot protocol for devices, one round of communications between aggregation servers

Common Solution across Google and Apple

Prio (published at NSDI 2017; presented at RWC 2020)

Agenda

Exposure Notifications (EN)

Exposure Notifications Private Analytics (ENPA)

Instantiation and Deployment of ENPA

Challenges and Recommendations

Instantiation of Prio for ENPA: a lot of moving parts

Parties Running Non-colluding Servers

Establishing Trust

Multi-Cloud Solution

Ingestion Servers

Aggregating over Time Windows

ENPA Health Authority Consent

ENPA User Consent

Device Attestation

Communication via Storage Buckets

Kubernetes Clusters

Cloud Identity Federation

Code Development

Serialization Format

Data Alignment

Parameter Choices

Differential Privacy Trade-offs

US deployment

Non-colluding Aggregation Servers

Servers built on Kubernetes and running on two distinct cloud providers



Internet Security Research Group

California public benefit corporation for digital infrastructure projects, operators of Let's Encrypt



Google



National Cancer Institute (NCI) at the National Institutes of Health (NIH)

Agency of the U.S. Department of Health and Human Services

Web Portal

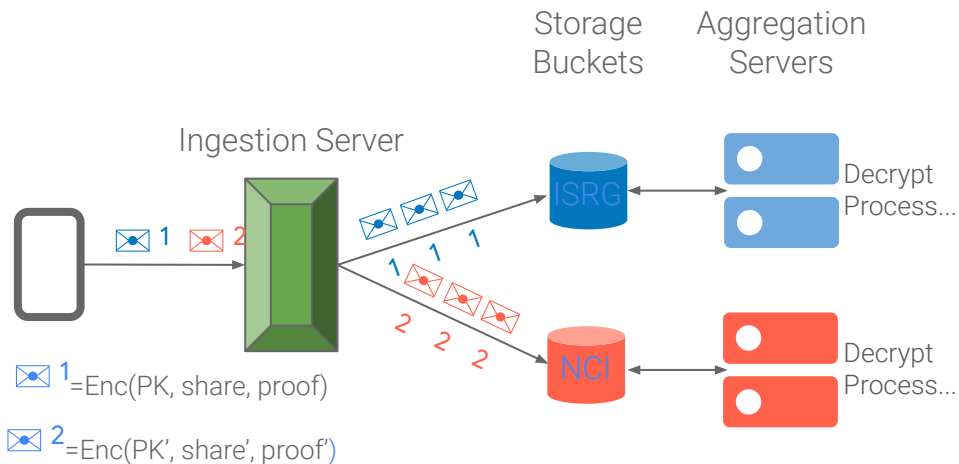
Data access and graphical representations of the differentially-private aggregates for all state health authorities



The MITRE Corporation

Not-for-profit organization

Ingestion Servers



Role of the ingestion servers run by Apple and Google

Always-online

Drop metadata (IP address, time)

Check device attestation (OS-specific)

Align the data (i.e., contribution forwarded to both aggregation servers)

Batch and reorder multiple contributions

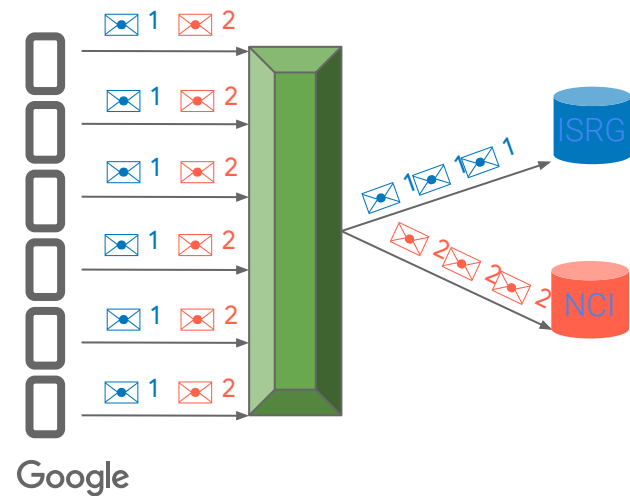
Authenticate batches

Generate random point for polynomial identity test

Asynchronous System

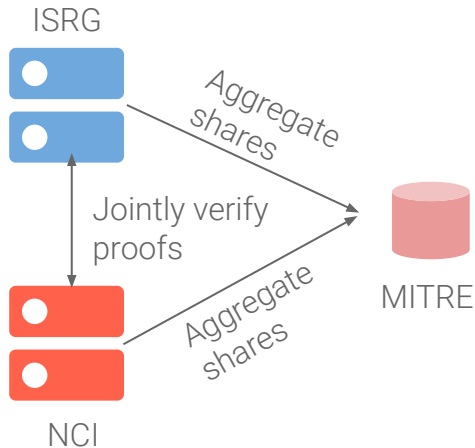
Data Collection

Receive continuous data from clients
Drop metadata
Periodically write shuffled batches of valid contributions to shared storage buckets



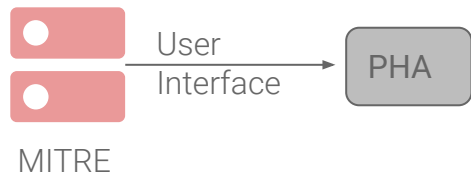
Data Aggregation

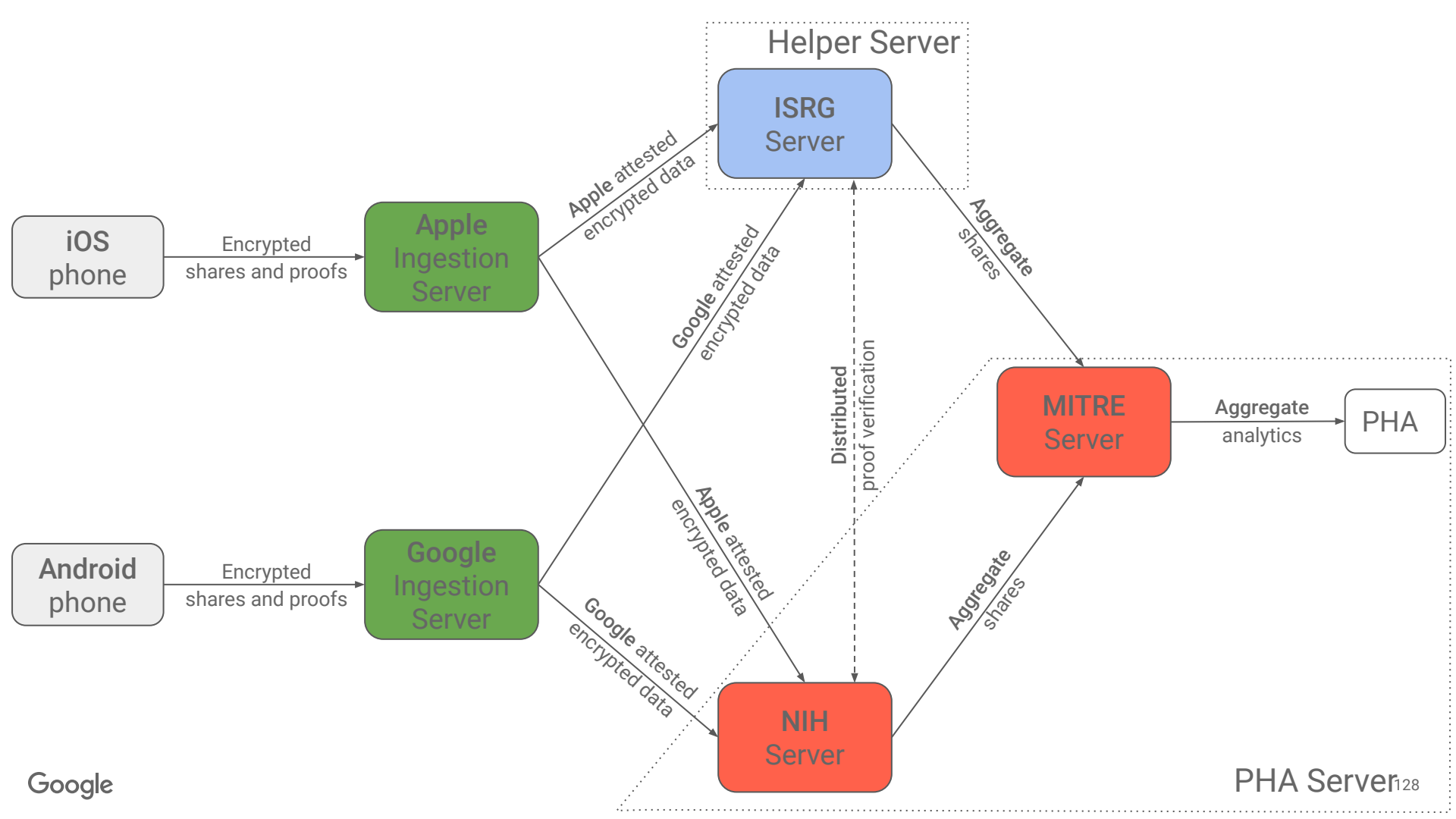
Periodically read data from shared storage
Decrypt, verify proofs, aggregate shares
Send aggregate parts to MITRE every 8 hours



Data Consumption

Sum aggregate parts over 24h
Provide UI for PHAs to access the aggregate data





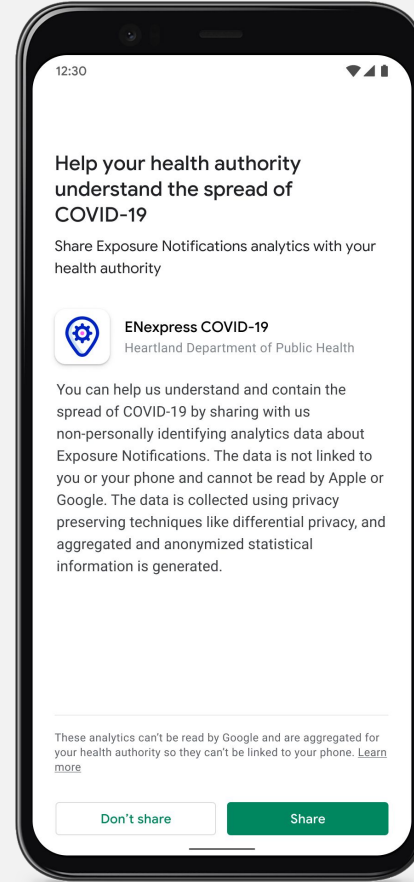
ENPA Consent

PHA consent

Each state public health authority (PHA) needs to elect to activate ENPA, by signing a contract with MITRE

User Consent

Each user consents to contribute their data to ENPA



Current Deployment

First deployment: December 2020

ENPA is used in 15 US states and 4 Mexican states. 9 metrics are collected and the aggregation servers process millions of contributions daily.

News Release



For immediate release: June 9, 2021 (21-149)

[Spanish](#)

Media contacts: [Teresa McCallion](#), DOH Communications, 360-701-7991

[Jake Ellison](#), UW Communications, 206-713-6420

Researchers from UW and DOH find WA Notify exposure notification tool is saving lives

OLYMPIA – Washington state's COVID-19 exposure notification tool, WA Notify, saved an estimated 30 to 120 lives and likely prevented about 6,000 COVID-19 cases during the first four months that it was in use.

Researchers at the University of Washington School of Public Health and the Washington State Department of Health (DOH) conducted a modeling study of the free tool and released their results in a [preprint version](#) on June 7, 2021. Their analysis offers the first estimate of the public health value of Bluetooth exposure notification systems in the U.S.



<https://www.doh.wa.gov/Newsroom/Articles/ID/2827/Researchers-from-UW-and-DOH-find-WA-Notify-exposure-notification-tool-is-saving-lives>

Appendix

References

- Papers:

- Secure Aggregation '17: <https://research.google/pubs/pub45808/>
- Secure Aggregation '20: <https://eprint.iacr.org/2020/704>
- [ENPA white paper](#)

- Presentations:

- [CCS 2017: Secure Aggregation](#)
- [RWC 2022: ENPA](#)
- [RWC 2023 : Crypto Agility and PQC](#)