- Presented by :
- -Arnaud Barral
- -Guillaume Weghsteen
- -Mahdi Kallel
- -Natan Talon
- -Martin Dubois

PRIVACY PRESERVING AGGREGATION OF TIME-SERIES DATA :

WHAT IS THE PROBLEM?

Compute a statistic (sum) of a certain quantity provided by several users over different periods. with two constraints:

1. We do not trust the aggregator
2. The result should not differ too much because of a specific individual

- Applications :

Smart Metering

Cloud computations

Public health

Population sensing

- We present an **AGGREGATOR OBLIVIOUS** construction allowing the computation sum of some users data all while conserving **DISTRIBUTED DIFFERENTIAL PRIVACY**.

THE SOLUTION :

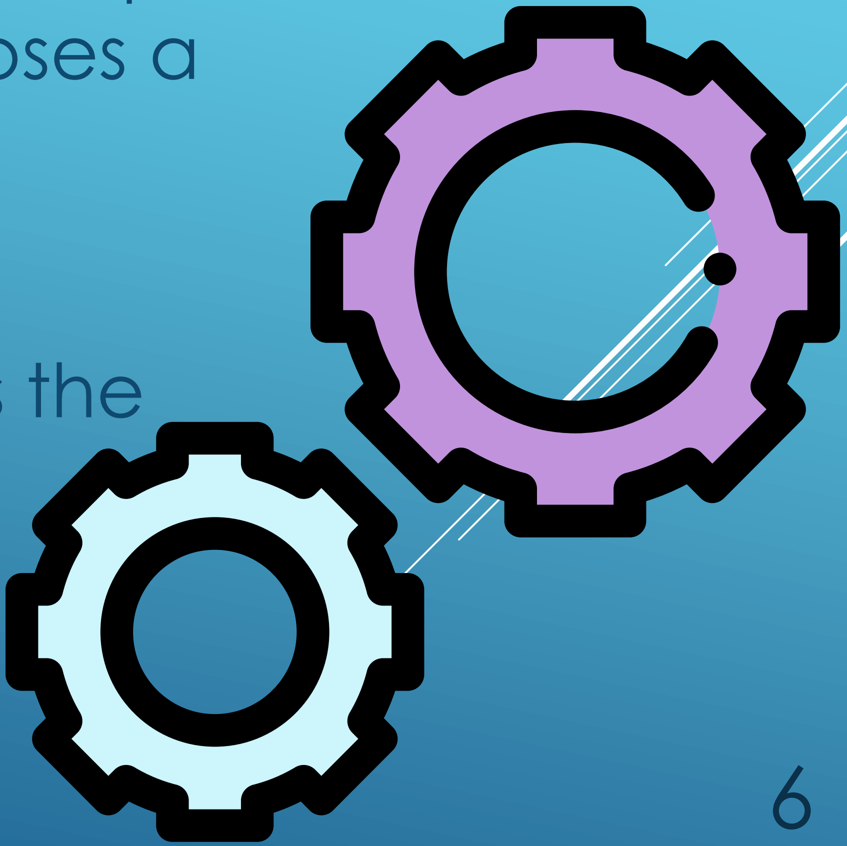
- The aggregator can only learn a noisy sum of each time period.
- Subsets of malicious users cannot learn anything without knowing the aggregator capability.
- In case of a collusion between a group of malicious users and the aggregator. The aggregator can only learn about remaining users sum and nothing more.

AGGREGATOR OBLIVIOUS :

HOW DOES THIS WORK?

Step 1 : Setup

- A trusted dealer (TTP) fixes a cyclic group of prime order P we note it **G** and chooses a generator **g** ∈ **G** .
- TTP chooses S0,S1…,Sn such that:
- S0 + S1 + S2 + … + Sn = 0 mod P (S0 is the capability of the aggregator)



HOW DOES THIS WORK?

Step 2 : Noisy Encryption

- For each user « i » :
  - Add noise to data :

$$\widehat{x_i} \leftarrow x_i + r$$

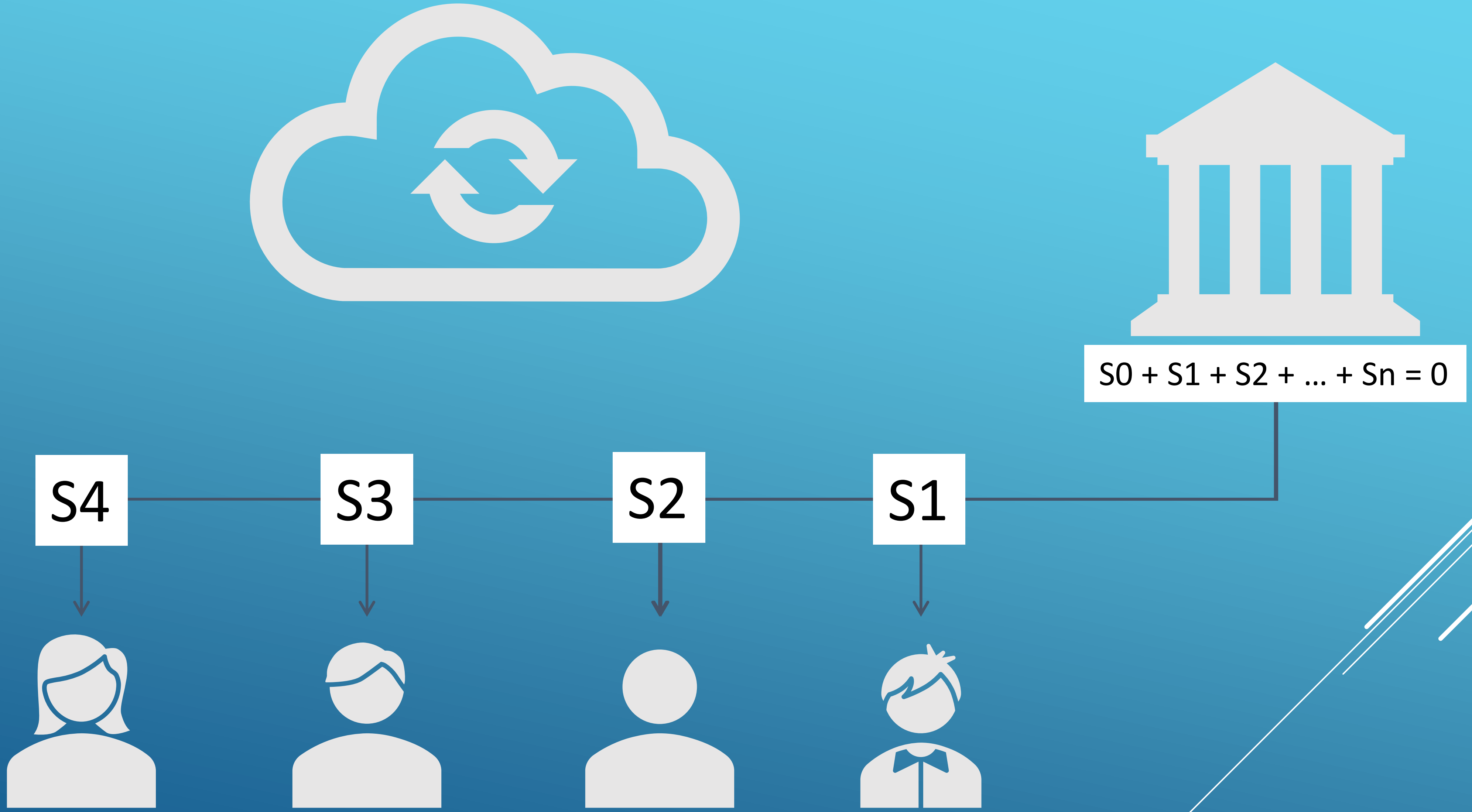
  - For time step t compute encryption :

$$c_i \leftarrow g^{\widehat{x_i}} H(t)^{sk_i}$$

HOW DOES THIS WORK?

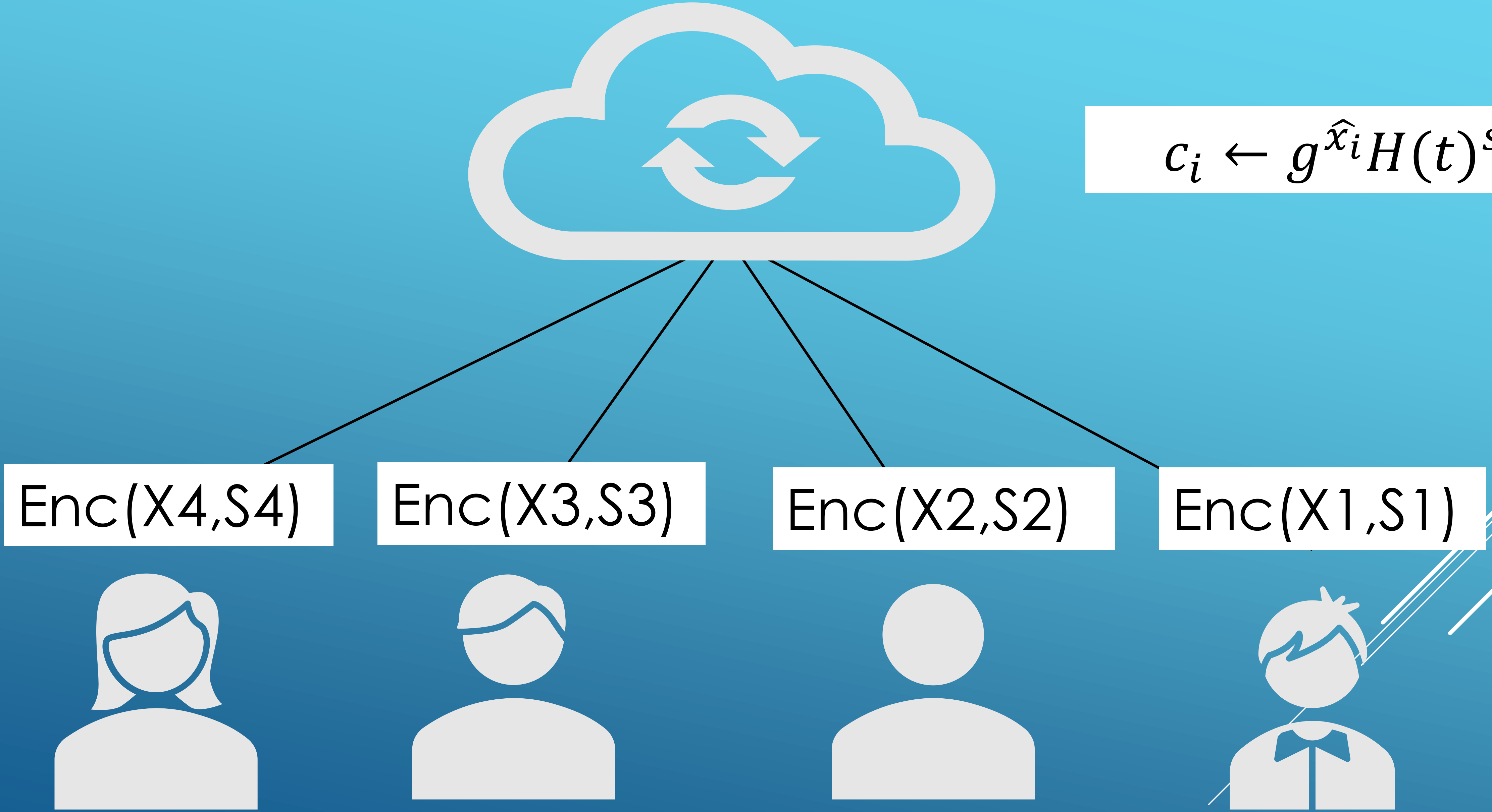
Step 3 : Decryption

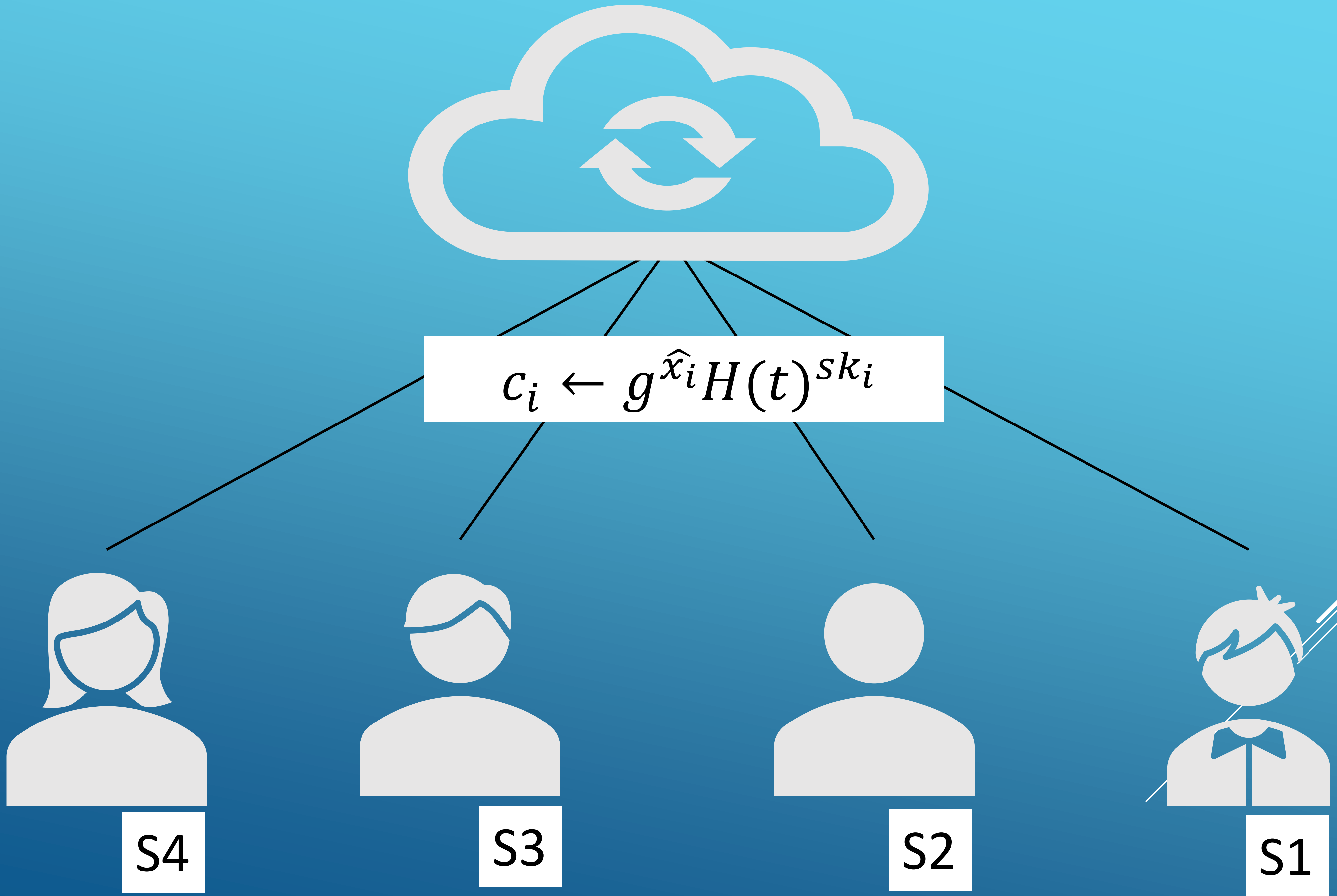
- At the aggregator:
  - Aggregate :
    - $V \leftarrow H(t)^{sk_0} \prod_{i=1}^{n} c_i$
  - Decrypt :
    - $V = g^{\sum_{i=1}^{n} \hat{x}_i}$
    - Compute discrete log of **V** base **g.**



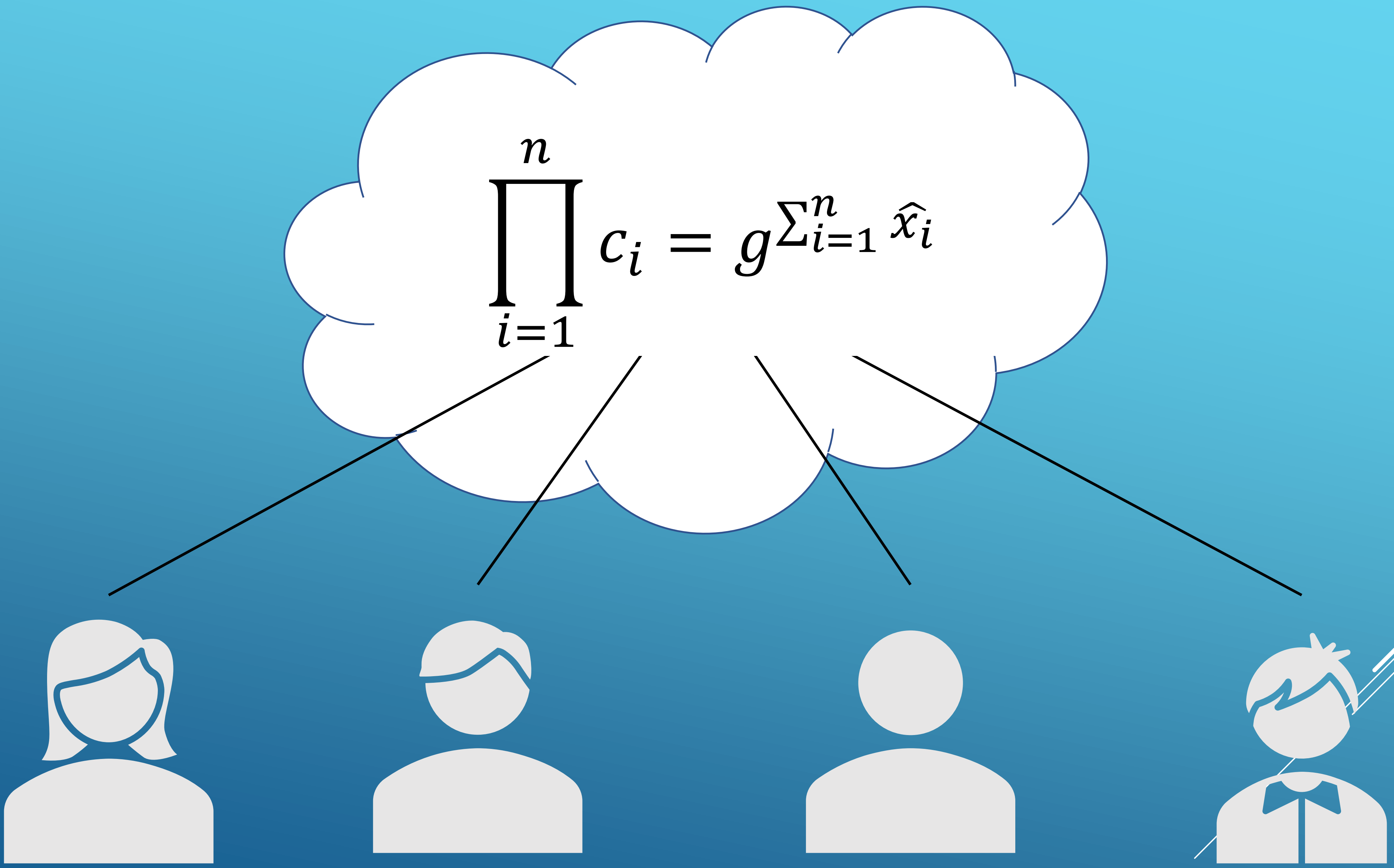
S0 + S1 + S2 + … + Sn = 0
S4
S3
S2
S1

$$c_i \leftarrow g^{\hat{x}_i} H(t)^{sk_i}$$

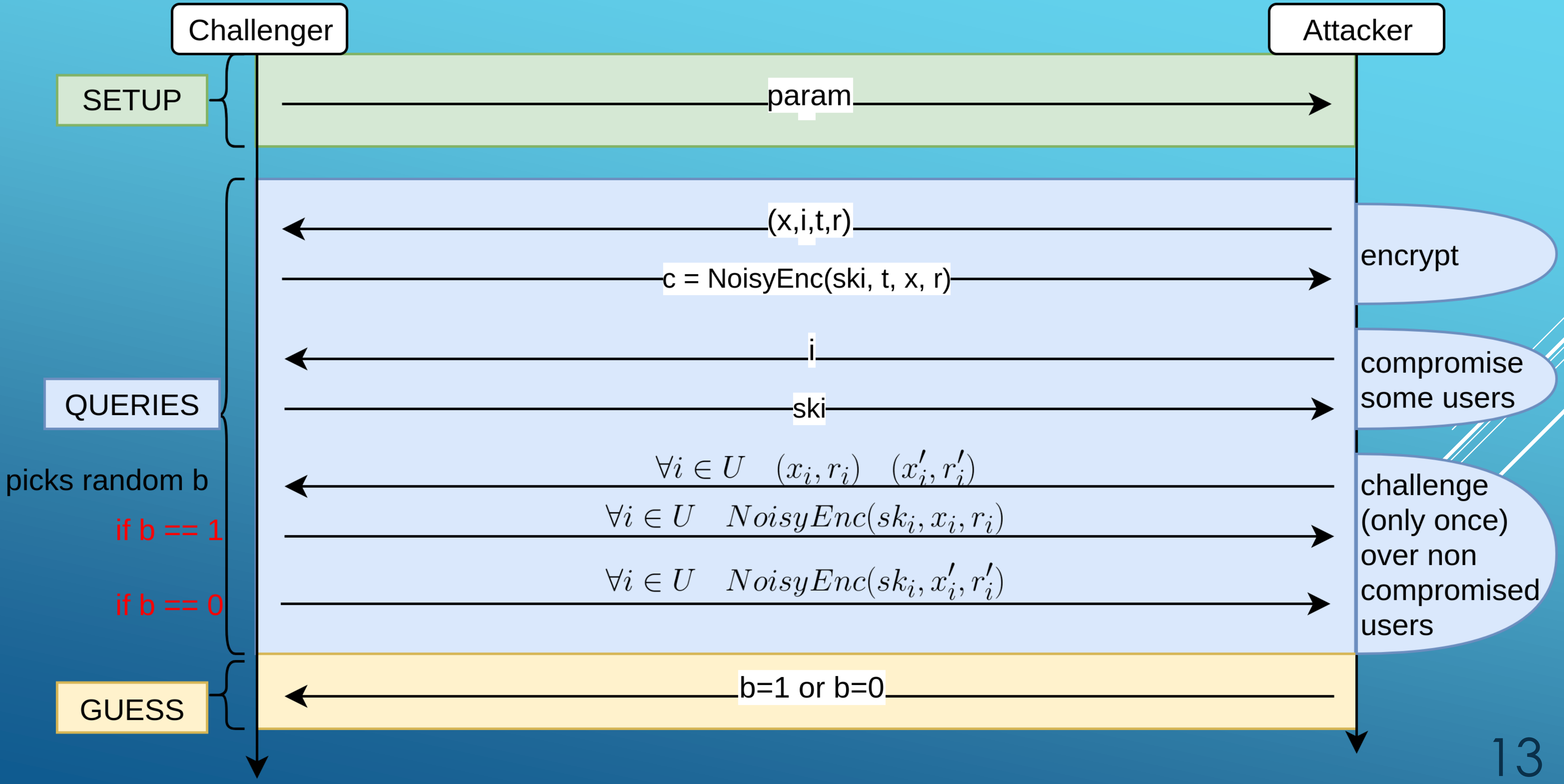




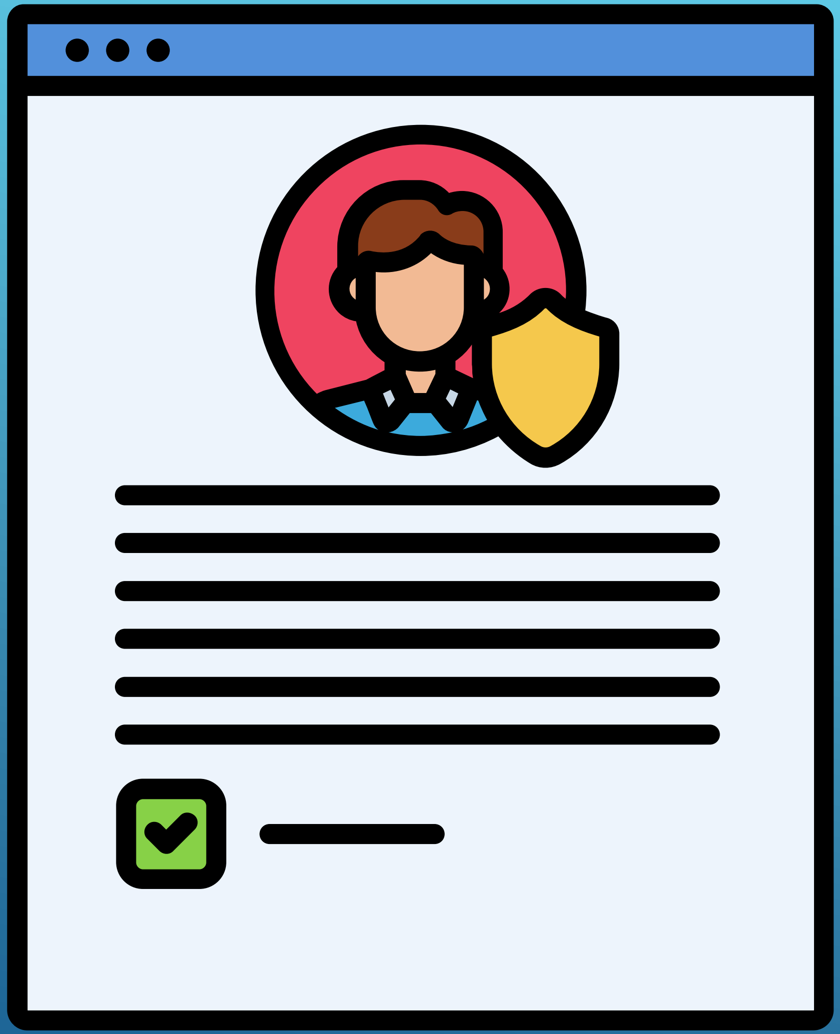
$c_i \leftarrow g^{\widehat{x}_i} H(t)^{sk_i}$
S4
S3
S2
S1



$$\prod_{i=1}^{n} c_i = g^{\sum_{i=1}^{n} \hat{x}_i}$$



Challenger
Attacker
SETUP
param
QUERIES
(x,i,t,r)
c = NoisyEnc(ski, t, x, r)
encrypt
i
ski
compromise some users
picks random b
$\forall i \in U \quad (x_i, r_i) \quad (x'_i, r'_i)$
if b == 1
$\forall i \in U \quad NoisyEnc(sk_i, x_i, r_i)$
if b == 0
$\forall i \in U \quad NoisyEnc(sk_i, x'_i, r'_i)$
challenge (only once) over non compromised users
GUESS
b=1 or b=0



DISTRIBUTED DIFFERENTIAL PRIVACY:

- Let A be a randomized algorithm. For any two neighbouring data bases $D1, D2$ any subset $S \subseteq range(A)$.
- A is ε-differentially private iff:
  - $\Pr[A(D1) \in S] \leq e^{\varepsilon} * \Pr[A(D2) \in S] + \delta$

WHY DO WE NEED DDP?

- To ensure privacy for all participants we must add some random noise to their data.
- One naive solution is to let each participant decides the magnitude of noise to his data
- This solution is not good. Indeed if each participant thinks the data of others are compromised, they will add too much noise and the final statistic will have a great error within it.

A NAIVE APPROACH

- DD-Private Data Randomization procedure :

Algorithm 1: DD-Private Data Randomization Procedure.

Let $\alpha := \exp(\frac{\epsilon}{\Delta})$ and $\beta := \frac{1}{\gamma n} \log \frac{1}{\delta}$.

Let $\mathbf{x} = (x_1, \ldots x_n)$ denote all participants' data in a certain time period.

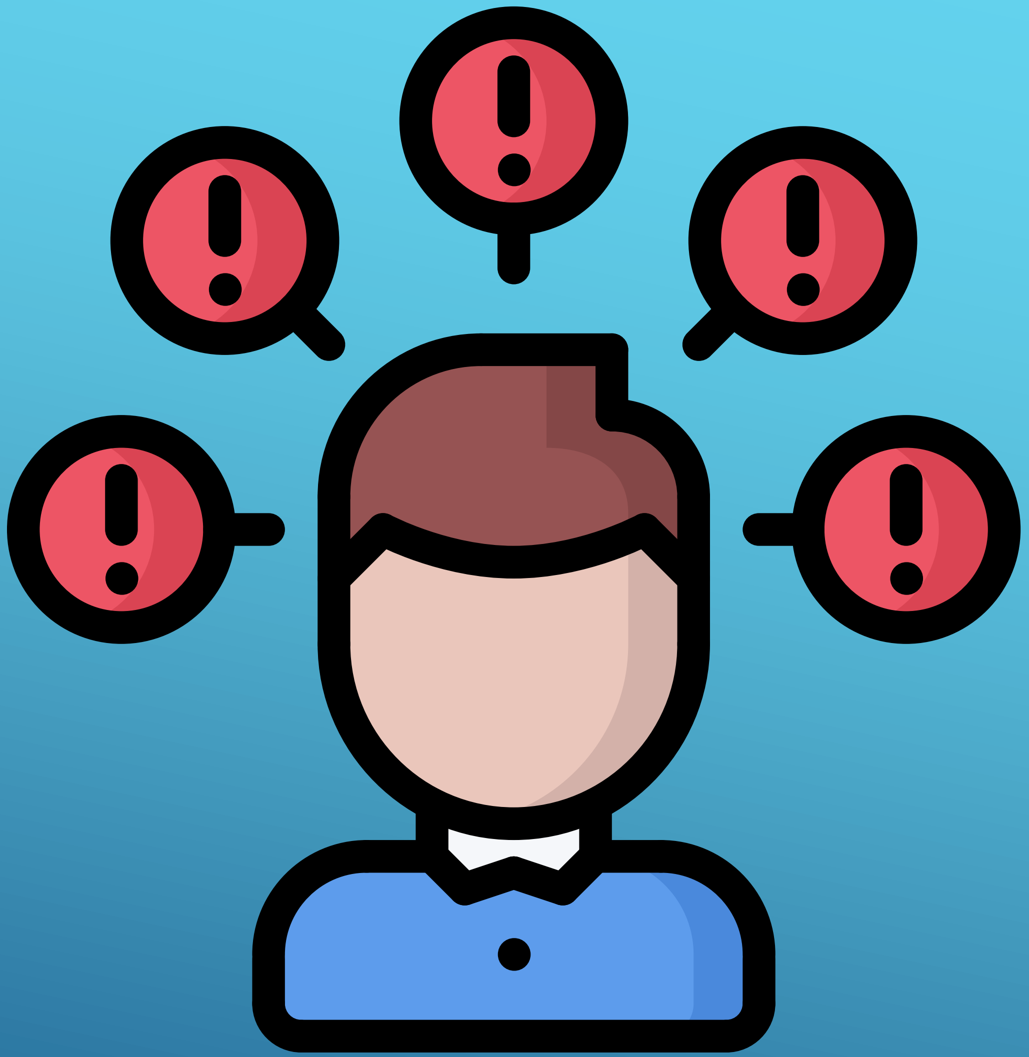
foreach participant $i \in [n]$ **do**

  Sample noise r_i according to the following distribution.

$$r_i \leftarrow \begin{cases} \mathsf{Geom}(\alpha) & \text{with probability } \beta \\ 0 & \text{with probability } 1 - \beta \end{cases}$$

  Randomize data by computing $\widehat{x}_i \leftarrow x_i + r_i \mod p$.

A BETTER APPROACH



QUESTIONS?