International Conference on Advanced Data Mining and Applications

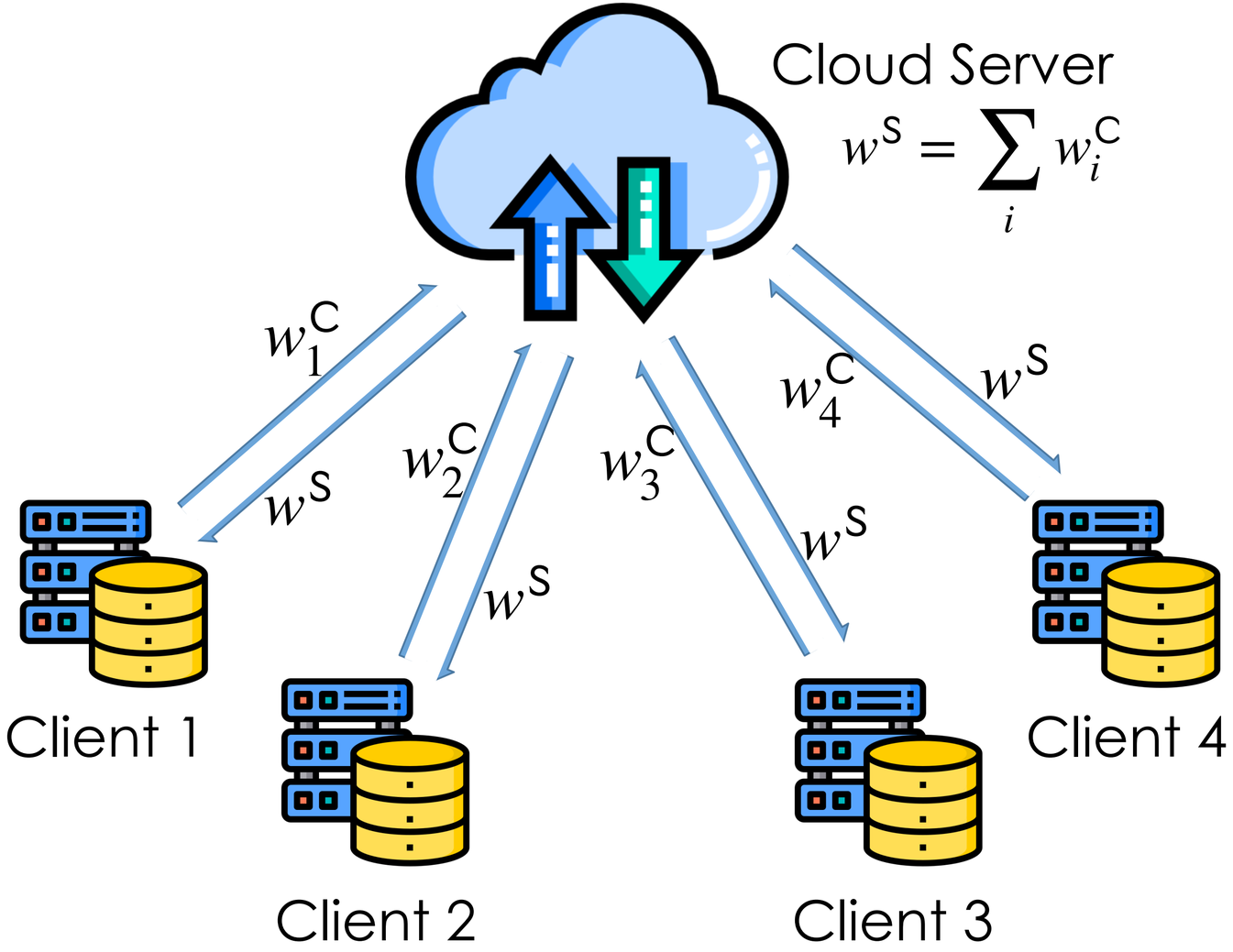
#59 Cryptography-Inspired Federated Learning for Generative Adversarial Networks and Meta Learning

Yu Zheng, Wei Song, Minxin Du, Sherman S.M. Chow, Qian Lou, Yongjun Zhao, Xiuhua Wang

香港中文大學
The Chinese University of Hong Kong

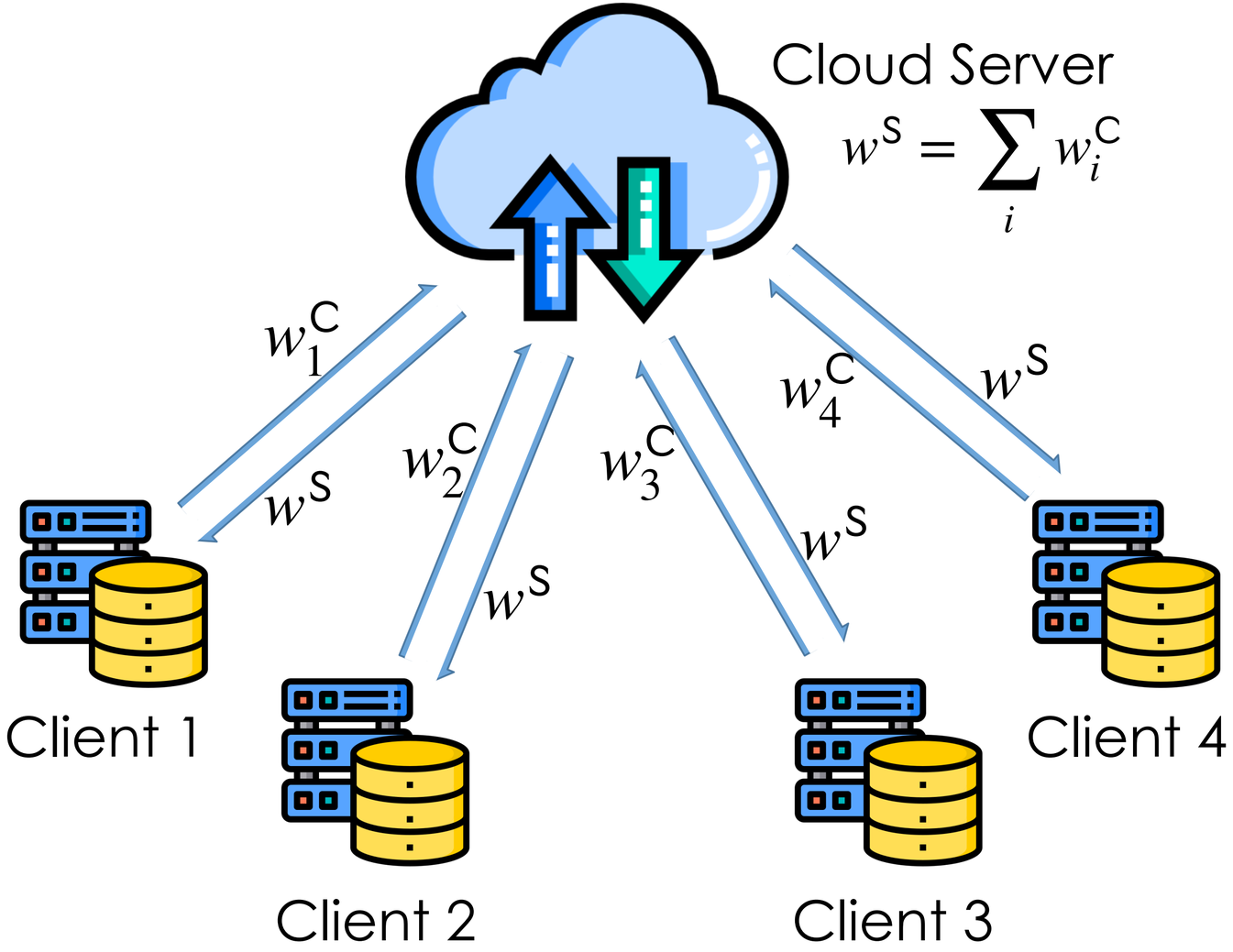
信息工程學系
Department of Information Engineering

Conventional Federated Learning (FL)



- Client $i \in \{1,2,3,4\}$ owns
  - local model w_i^C
  - local private dataset d_i
- Cloud server S owns
  - aggregated model w^S
  - no/public data

Conventional Federated Learning (FL)



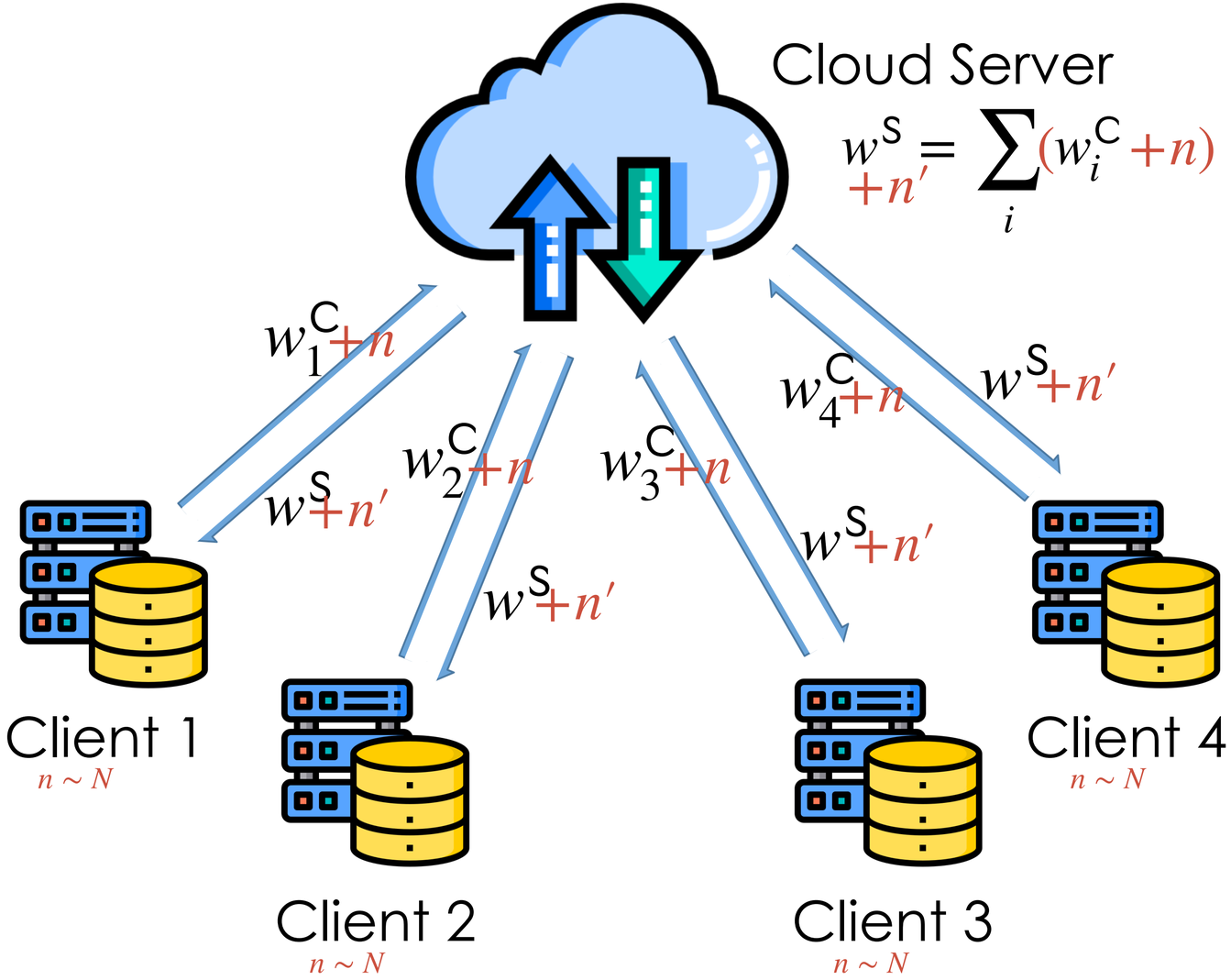
- Client $i \in \{1,2,3,4\}$ owns
  - local model w_i^C
  - local private dataset d_i
- Cloud server S owns
  - aggregated model w^S
  - no/public data

Guarantee: no access to d_i

Issue: expose w_i^C at each update

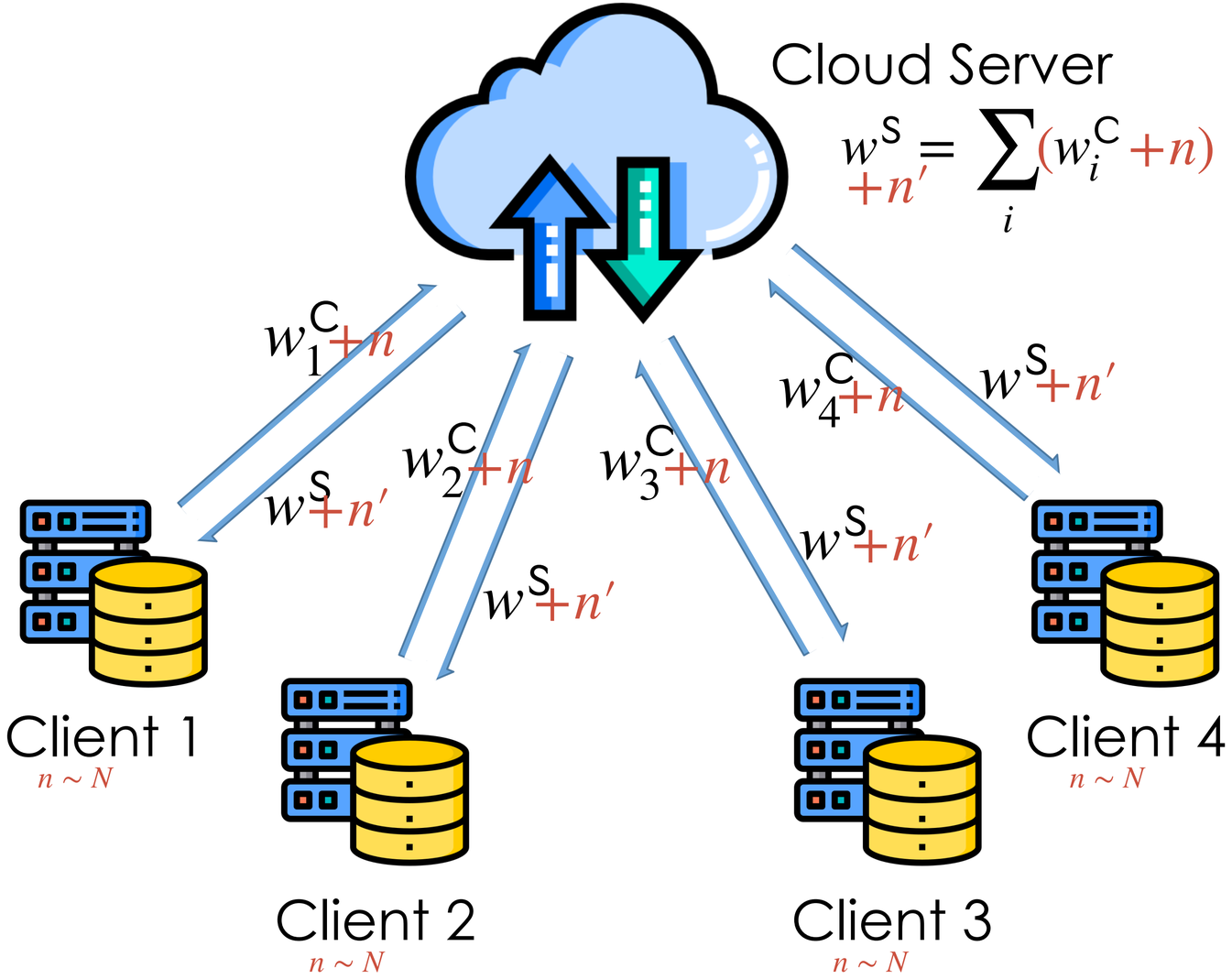
Imply: private data

Federated Learning with Privacy



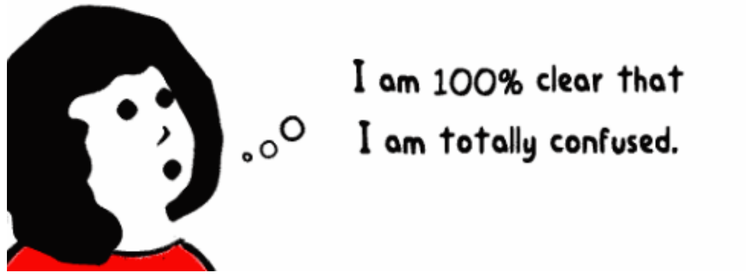
- Client i protects her data
  - Keep d_i locally
  - add i.i.d. noise n to w_i^C
- Server S observes
  - no access to d_i
  - Noisy $w_i^C + n$
  - aggregated model $w^S + n'$

Federated Learning with Privacy



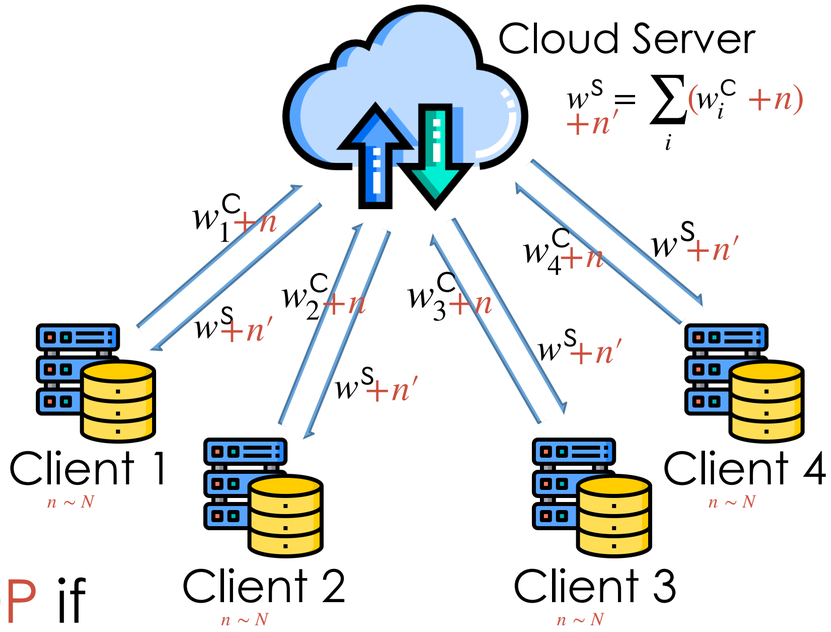
- Client i protects her data
  - Keep d_i locally
  - add i.i.d. noise n to w_i^C
- Server S observes
  - no access to d_i
  - Noisy $w_i^C + n$
  - aggregated model $w^S + n'$

➡ What is so called "noise"?



Differential Privacy (DP)

- "Noisy" protection — add DP noise
  - n sampled from Gaussian distribution $N(0,\sigma^2 \cdot S_2^2(f))$
  - Mean 0 and standard deviation $\sigma \cdot s_2(f)$

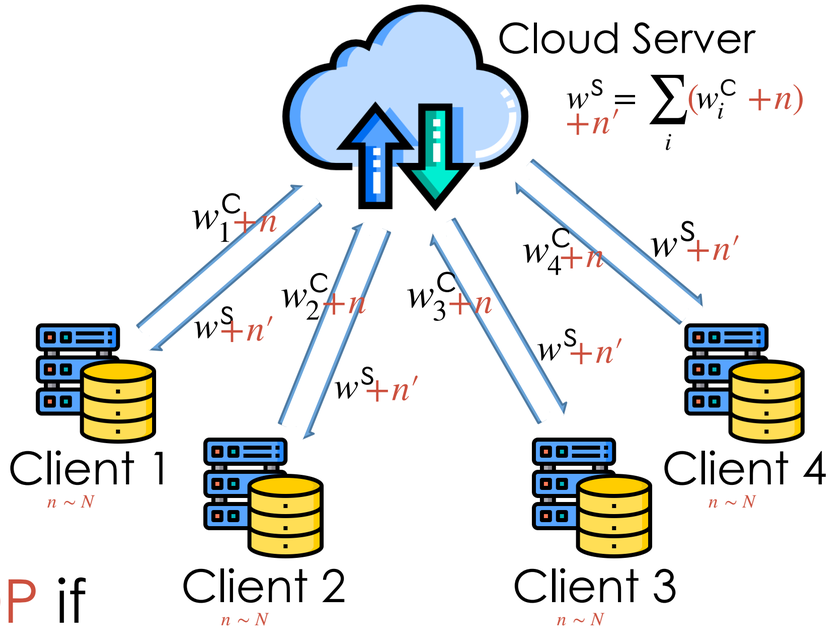


Formal Definition

- A randomized mechanism $M : \mathbb{D} \rightarrow \mathbb{R}$ ensures (ϵ, δ)-DP if
  - for any two neighboring datasets D, D' differing in a single example
  - the output space of M satisfying $\Pr(M(D) \in \mathbb{R}) \leq e^{\epsilon} \cdot \Pr(M(D') \in \mathbb{R}) + \delta$
  - where $\epsilon \geq 0$ and $0 \leq \delta \leq 1$

Differential Privacy (DP)

- "Noisy" protection — add DP noise
  - n sampled from Gaussian distribution $N(0,\sigma^2 \cdot S_2^2(f))$
  - Mean 0 and standard deviation $\sigma \cdot s_2(f)$



Formal Definition

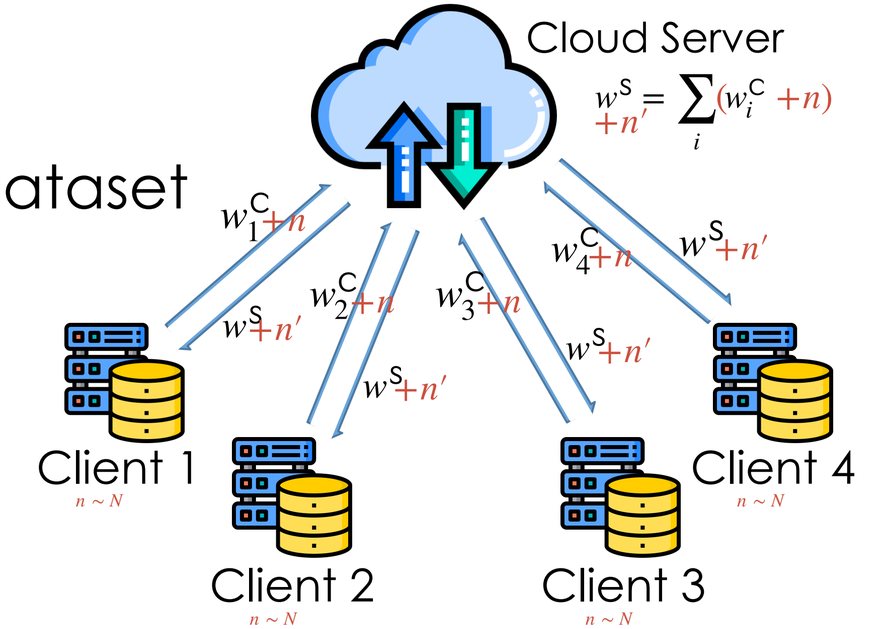
- A randomized mechanism $M : \mathbb{D} \rightarrow \mathbb{R}$ ensures (ϵ, δ)-DP if
  - for any two neighboring datasets D, D' differing in a single example
  - the output space of M satisfying $\Pr(M(D) \in \mathbb{R}) \leq e^{\epsilon} \cdot \Pr(M(D') \in \mathbb{R}) + \delta$
  - where $\epsilon \geq 0$ and $0 \leq \delta \leq 1$

- DP guarantee: de facto standard to protect individual privacy
  - a single data example incurs a very-limited impact on statistical analytics performed on the entire dataset of all individuals

Differential Privacy in Federated Learning

- What can DP do in FL?
  - protect individual data example in local private dataset
  - ensure limited variation on statistics of local data

➡ What if we wanna protect statistics? (**Privacy**)



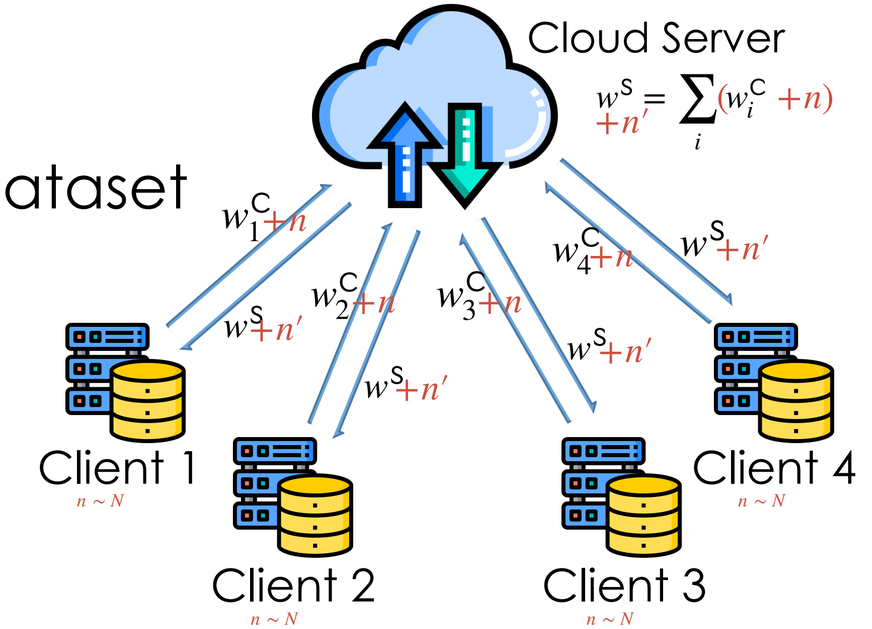
- Does DP protection come at a price?
  - utility-privacy trade-off
    - intuition: more noise <=> lower accuracy; less noise <=> higher accuracy
  - worse: accumulated effect of noise on the server

➡ Is there any protection with lossless accuracy? (**Functionality**)

Differential Privacy in Federated Learning

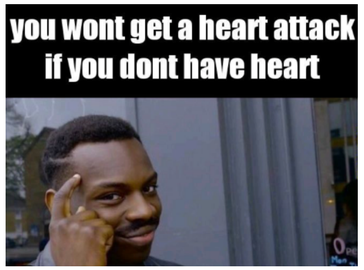
- What can DP do in FL?
  - protect individual data example in local private dataset
  - ensure limited variation on statistics of local data

➡ What if we wanna protect statistics? (**Privacy**)



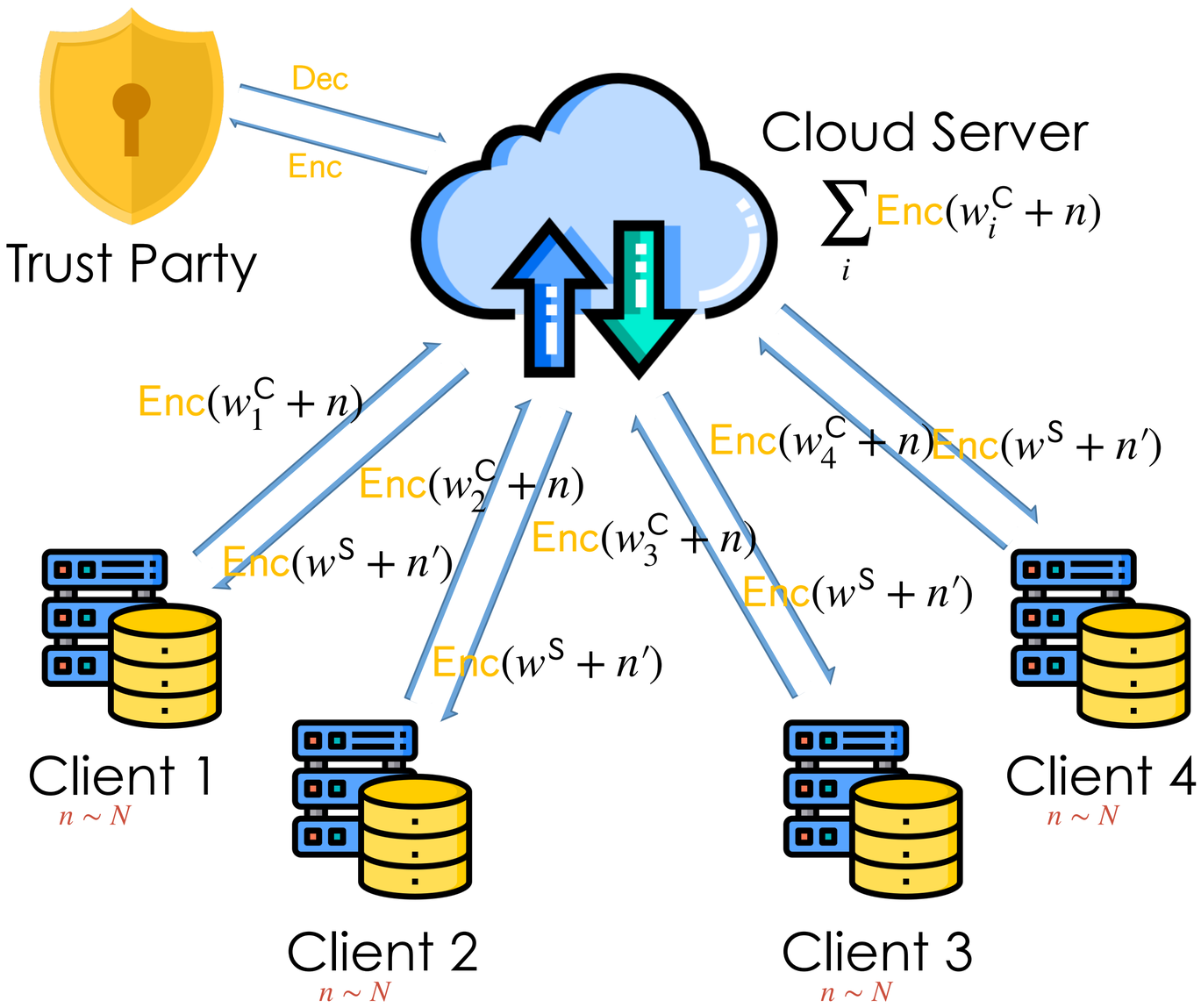
- Does DP protection come at a price?
  - utility-privacy trade-off
    - intuition: more noise <=> lower accuracy; less noise <=> higher accuracy
  - worse: accumulated effect of noise on the server

➡ Is there any protection with lossless accuracy? (**Functionality**)

➡ 

So easy! Cryptography helps! Let's encrypt w_i^C.

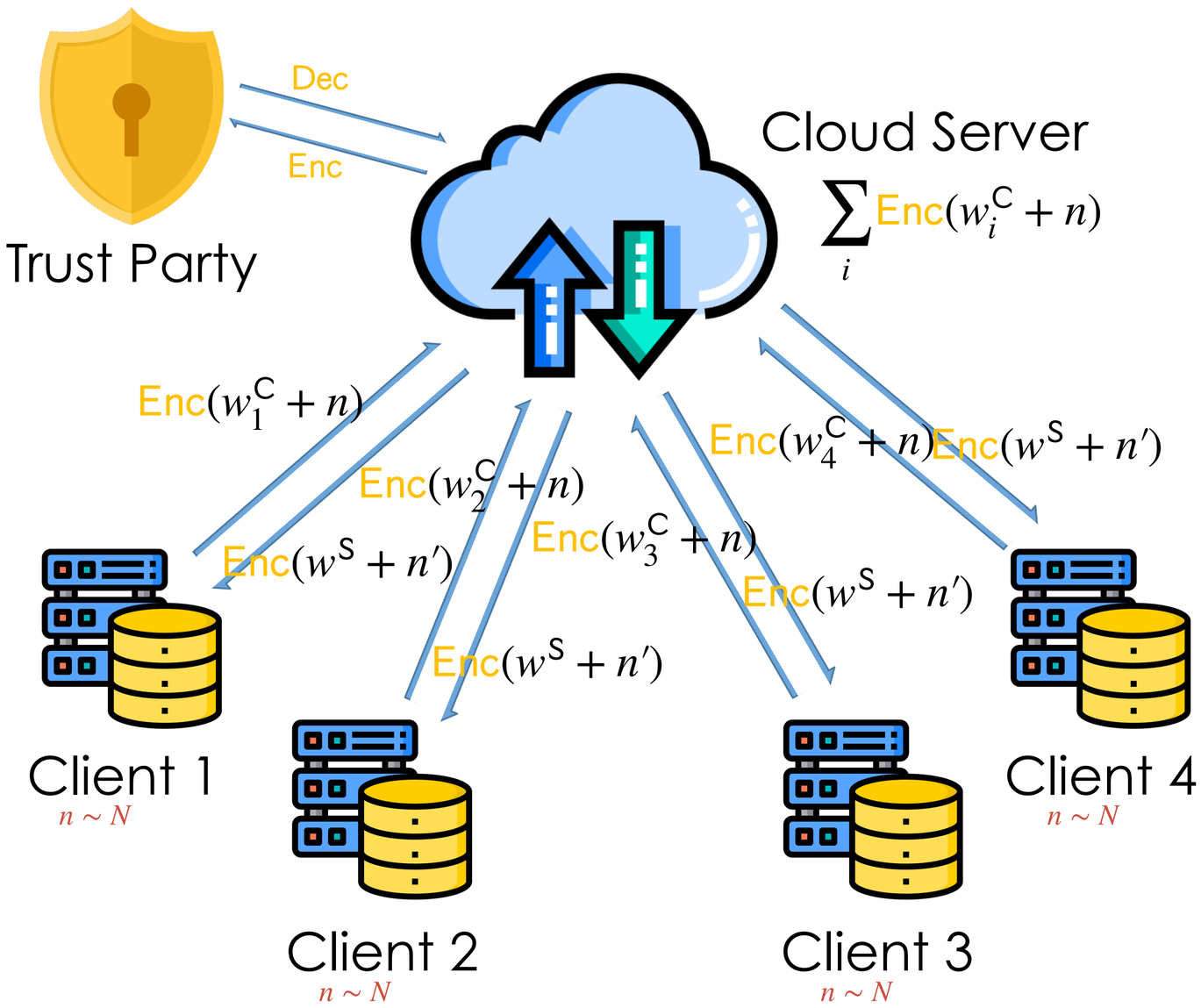
Let's encrypt "everything" transmitted …



- Client i protects her data
  - individual privacy - DP
  - local statistics - Crypto
- Server **S** operates
  - aggregation over **Enc**($\cdot$)
  - no access to $w_i^{\text{C}}, w_i^{\text{C}} + n$

➡ Obviously, communication/computation **efficiency** would be very low…

Let's encrypt "everything" transmitted …

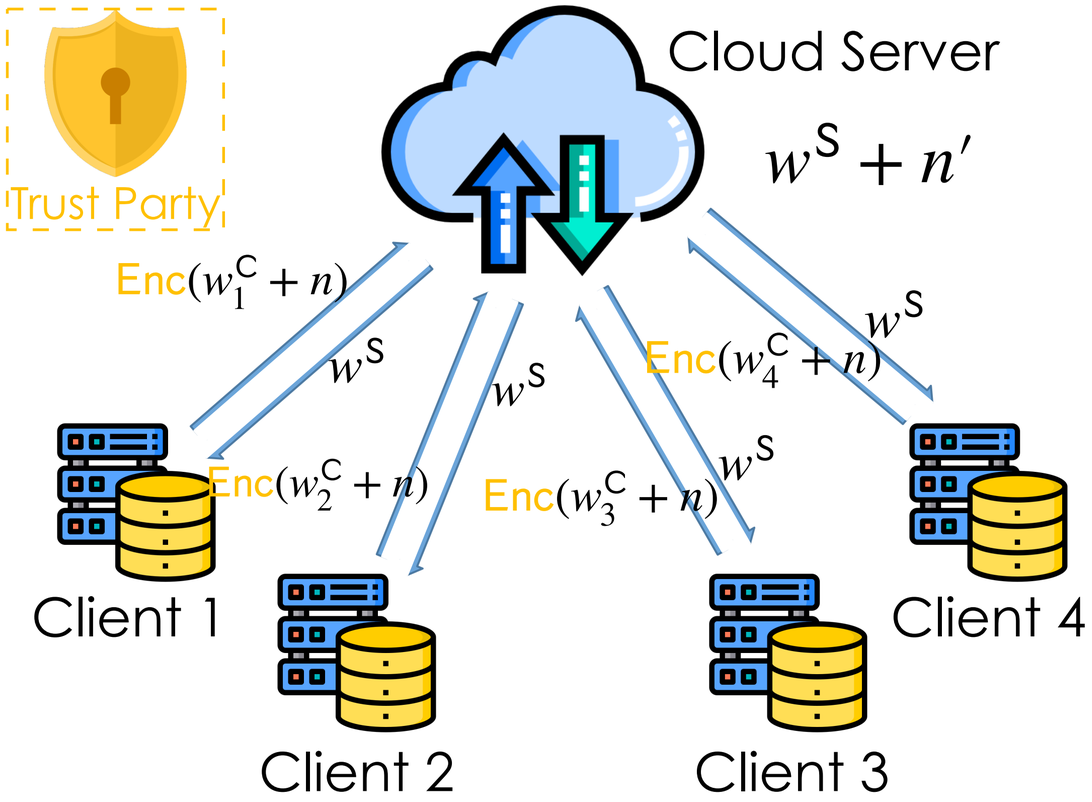


- Client i protects her data
  - individual privacy - DP
  - local statistics - Crypto
- Server S operates
  - aggregated over $\text{Enc}(\cdot)$
  - no access to $w_i^{\text{C}}, w_i^{\text{C}} + n$

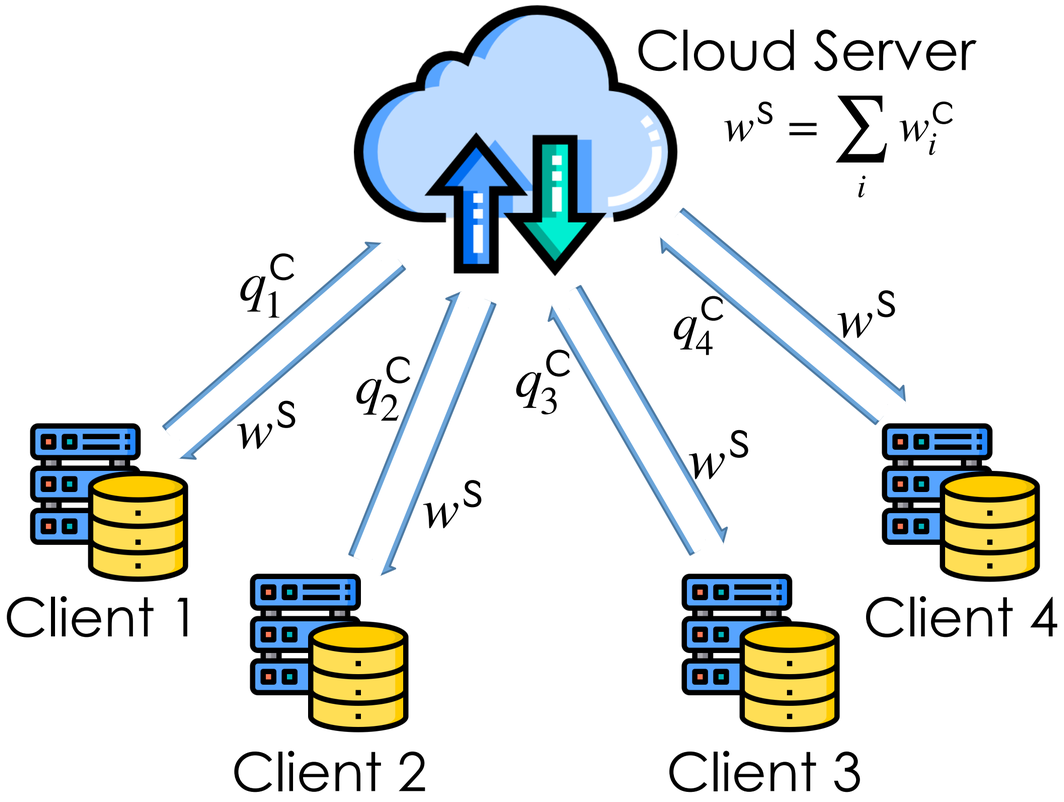
➡ Obviously, communication/computation **efficiency** would be very low…

Can we do better regarding **privacy**, **functionality**, and **efficiency**?

Our Hierarchical Protection in General

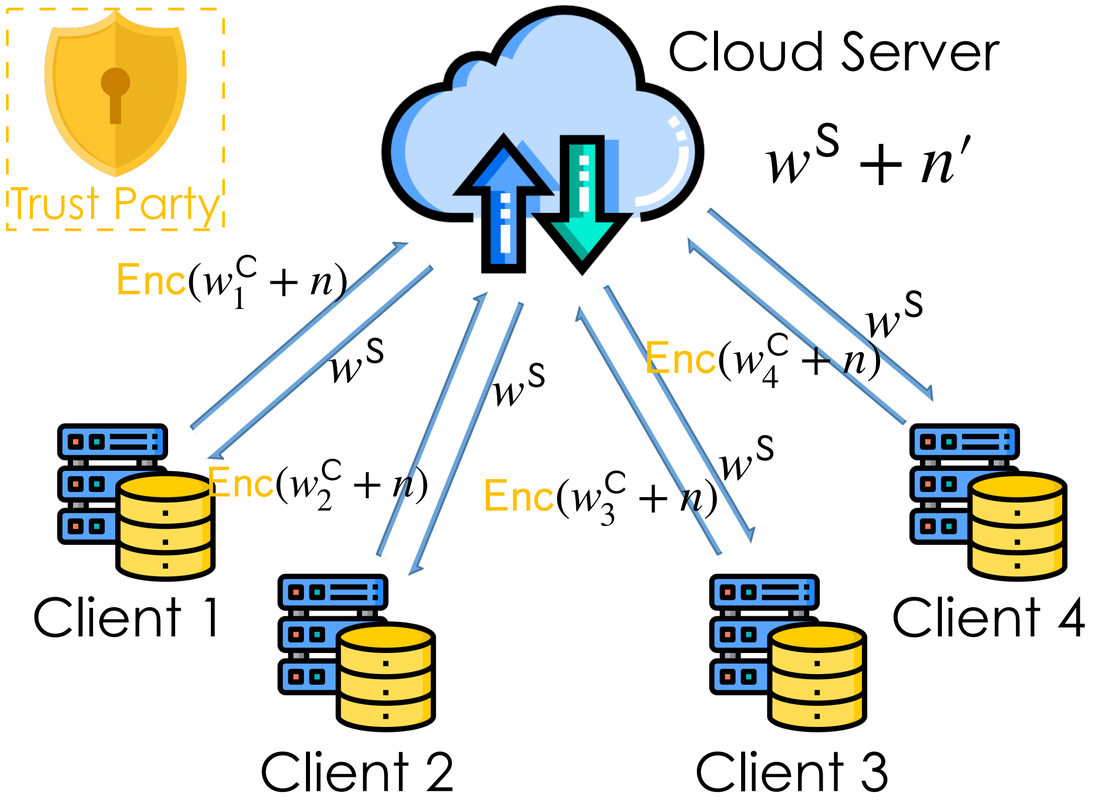


Phase 1: secure aggregation



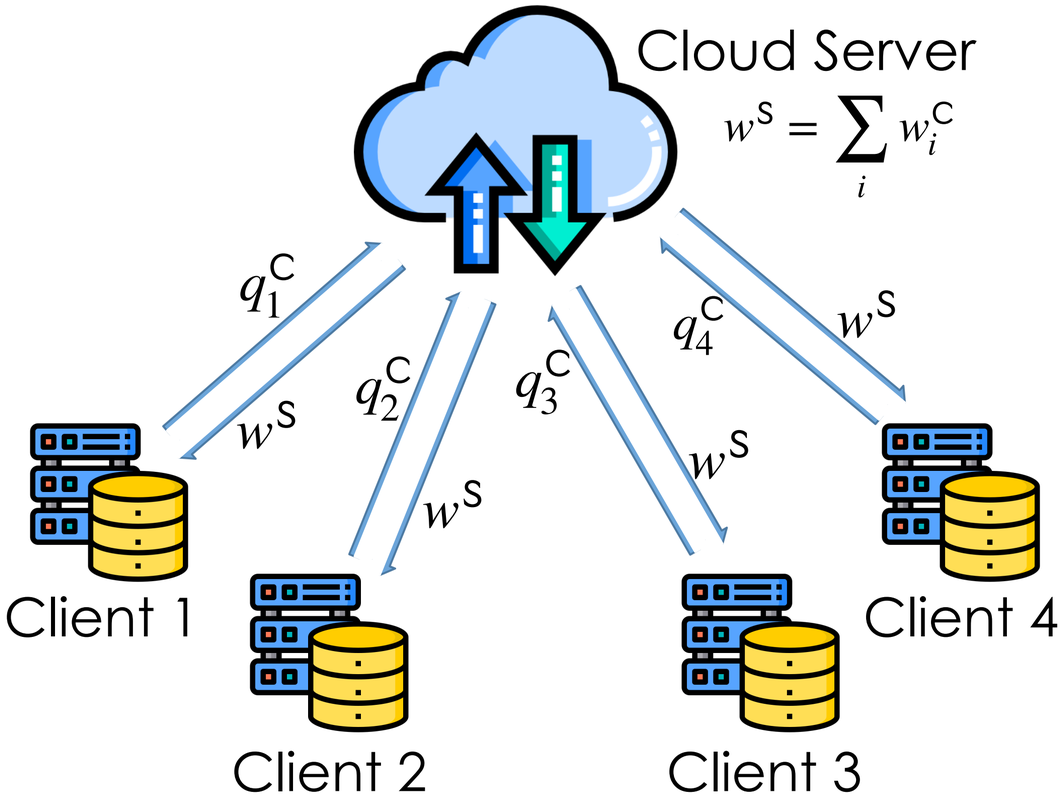
Phase 2: global tuning with DP

Our Hierarchical Protection in General



Phase 1: secure aggregation

- Client i sends $\mathrm{Enc}(W_i^C + n)$
- Crypto: guarantees only $W^S + n'$ revealed to the server
  - only once



Phase 2: global tuning with DP

- Client i slightly tunes the global model
- DP: guarantees individual privacy
  - Limited number of data
  - can be repeated

Concrete Instantiation and Outline

- Federated learning
  - server: train a global model over "indirect-use" data (i.e., global data)
  - client: train a local model over local private data
  - intuition: clients train over data for "knowledge", server trains over "knowledge"
- Cryptography: heavy communication/computation, thus only once
  - CKKS (**KeyGen**, **Enc**, **Dec**, **Add**): protect local statistics (phase 1)
  - remove trusted party: additive secret sharing
- Differential privacy
  - DP-Adam: converges better & example-level privacy (phase 2)
  - Following the style of DP-SGD
- Learning scenarios
  - Generative neural networks (GAN)
    - **S**: train a generative model for generating representative samples
  - Meta learning
    - **S**: train a new task by learning knowledge from related tasks

How do we construct Secure Federated GAN? (3 Steps)

- **Step 1.** Non-federated/centralized DP-GAN
  - devise DP-Adam as a plug-in module
  - GAN training
    - contain a generator θ^G and a discriminator θ^D
    - solve the min-max optimization
  - replace DP-Adam with Adam in GAN training
    - essentially add the DP noise to discriminator (input private data)
    - post-processing technique for DP guarantee

How do we construct Secure Federated GAN? (3 Steps)

- **Step 1.** Non-federated/centralized DP-GAN
  - devise DP-Adam as a plug-in module
  - GAN training
    - contain a generator θ^{G} and a discriminator θ^{D}
    - solve the min-max optimization
  - replace DP-Adam with Adam in GAN training
    - essentially add the DP noise to discriminator (input private data)
    - post-processing technique for DP guarantee
- **Step 2.** Construct Phase 1
  - each client holds a DP-GAN
  - client i encrypts local $\mathbf{G}_i$ and uploads it to the server
    - secure aggregation with CKKS
  - server obtains initial global model

How do we construct Secure Federated GAN? (3 Steps)

- **Step 1.** Non-federated/centralized DP-GAN
  - devise DP-Adam as a plug-in module
  - GAN training
    - contain a generator θ^{G} and a discriminator θ^{D}
    - solve the min-max optimization
  - replace DP-Adam with Adam in GAN training
    - essentially add the DP noise to discriminator (input private data)
    - post-processing technique for DP guarantee
- **Step 2.** Construct Phase 1
  - each client holds a DP-GAN
  - client i encrypts local $\mathbf{G}_i$ and uploads it to the server
    - secure aggregation with CKKS
  - server obtains initial global model
- **Step 3.** Construct Phase 2
  - client i tunes global model using its θ_i^{D}

DP-Adam

- Input: examples $\{x_1, \ldots, x_N\}$, hyper-parameters $\alpha, \beta_1, \beta_2, \gamma, L, C, \sigma$
- Output: model parameters θ_T

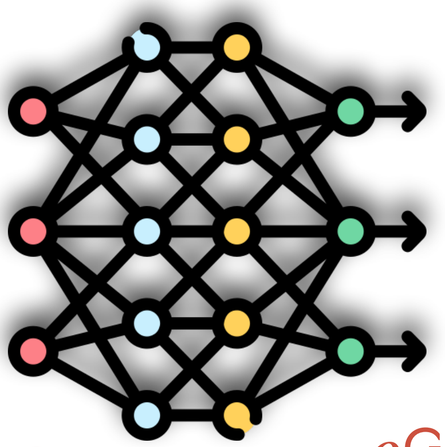
1 Initialize Θ_0 randomly;
2 **for** $t \in [T]$ **do**
3 Randomly sample L examples as a batch B_t;
4 Compute gradient $\mathbf{g}_t(x_i)$ for $\forall x_i \in B_t$;
5 Clip gradient $\bar{\mathbf{g}}_t(x_i) = \mathbf{g}_t(x_i)/\max(1, \frac{\|\mathbf{g}_t(x_i)\|_2}{C})$;
6 Add noise $\tilde{\mathbf{g}}_t = \frac{1}{L}(\sum_i \bar{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I}))$;
7 $\beta_1' = 1 - \beta_1, \beta_2' = 1 - \beta_2, u_t = \beta_1 \cdot u_{t-1} + \beta_1' \cdot \tilde{\mathbf{g}}_t, v_t = \beta_2 \cdot v_{t-1} + \beta_2' \cdot \tilde{\mathbf{g}}_t^2$;
8 $\bar{u}_t = u_t/\beta_1', \bar{v}_t = v_t/\beta_2'$;
9 $\Theta_{t+1} = \Theta_t - \alpha \cdot \bar{u}_t/(\sqrt{\bar{v}_t} + \gamma)$
10 **end**
11 Return Θ_T

DP-Adam

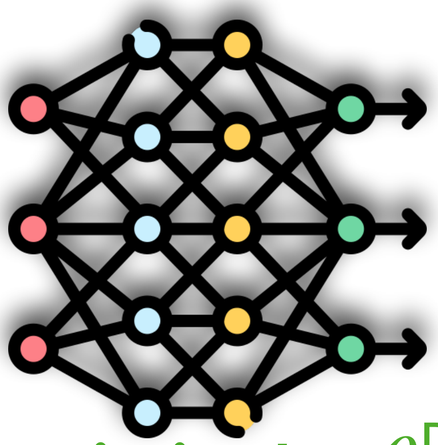
- Input: examples $\{x_1, \ldots, x_N\}$, hyper-parameters $\alpha, \beta_1, \beta_2, \gamma, L, C, \sigma$
- Output: model parameters θ_T

1 Initialize Θ_0 randomly;
2 **for** $t \in [T]$ **do**
3 Randomly sample L examples as a batch B_t;
4 Compute gradient $\mathbf{g}_t(x_i)$ for $\forall x_i \in B_t$;
5 Clip gradient $\bar{\mathbf{g}}_t(x_i) = \mathbf{g}_t(x_i)/\max(1, \frac{\|\mathbf{g}_t(x_i)\|_2}{C})$;
6 Add noise $\tilde{\mathbf{g}}_t = \frac{1}{L}(\sum_i \bar{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I}))$;
7 $\beta_1' = 1 - \beta_1, \beta_2' = 1 - \beta_2, u_t = \beta_1 \cdot u_{t-1} + \beta_1' \cdot \tilde{\mathbf{g}}_t, v_t = \beta_2 \cdot v_{t-1} + \beta_2' \cdot \tilde{\mathbf{g}}_t^2$;
8 $\bar{u}_t = u_t/\beta_1', \bar{v}_t = v_t/\beta_2'$;
9 $\Theta_{t+1} = \Theta_t - \alpha \cdot \bar{u}_t/(\sqrt{\bar{v}_t} + \gamma)$
10 **end**
11 Return Θ_T

DP-GAN



Generator θ_i^G



Discriminator θ_i^D

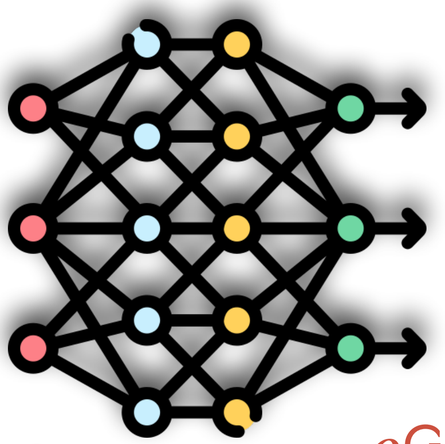
- θ^{G_i} — Adam
- θ^{D_i} — DP-Adam

- Each client holds a pair of $(\theta^{G_i}, \theta^{D_i})$
- Training over private data locally

- Produce initial θ^G
- θ^G's DP guarantee
  - post-processing technique

DP-GAN

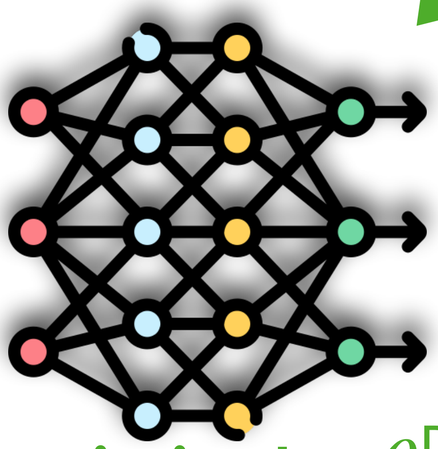
private real data



Generator θ_i^G

- θ^{G_i} — Adam
- θ^{D_i} — DP-Adam

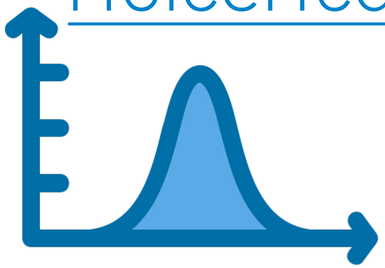
- Each client holds a pair of $(\theta^{G_i}, \theta^{D_i})$
- Training over private data locally



Discriminator θ_i^D

Predict	Target
0.1	0
~	or
0.9	1

Add DP noise sampled from

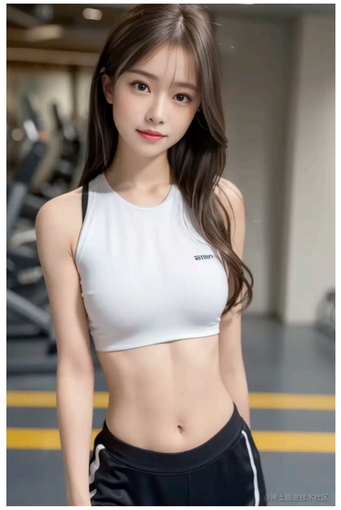


Protect real data only

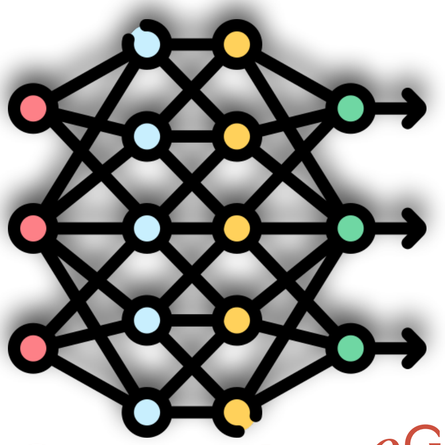
- Produce initial θ^G
- θ^G's DP guarantee
  - post-processing technique

DP-GAN

private real data



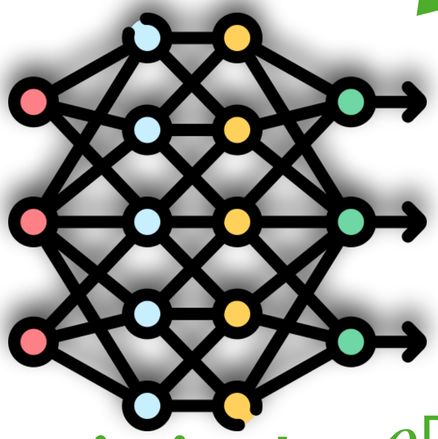
generated sample



Generator θ_i^G

feedback

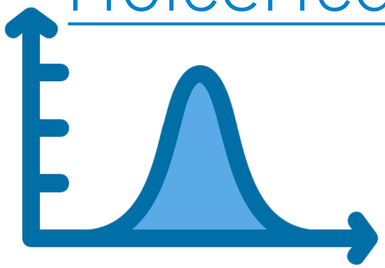
disturbance vector



Discriminator θ_i^D

Predict	Target
0.1	0
~	or
0.9	1

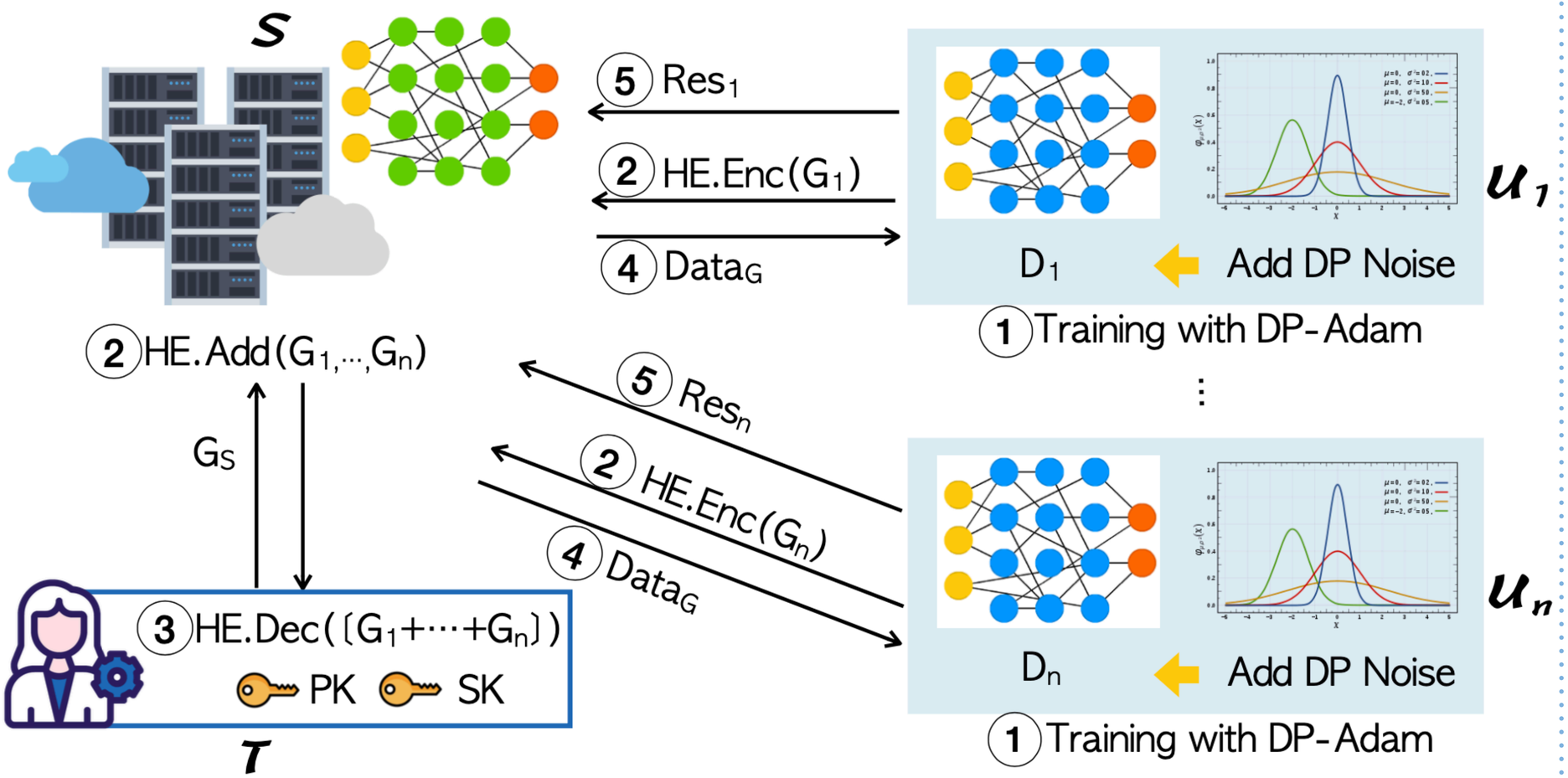
Add DP noise sampled from



Protect real data only

- θ^{G_i} — Adam
- θ^{D_i} — DP-Adam
- Each client holds a pair of $(\theta^{G_i}, \theta^{D_i})$
- Training over private data locally
- Produce initial θ^G
- θ^G's DP guarantee
  - post-processing technique

Secure Federated GAN — Our Framework

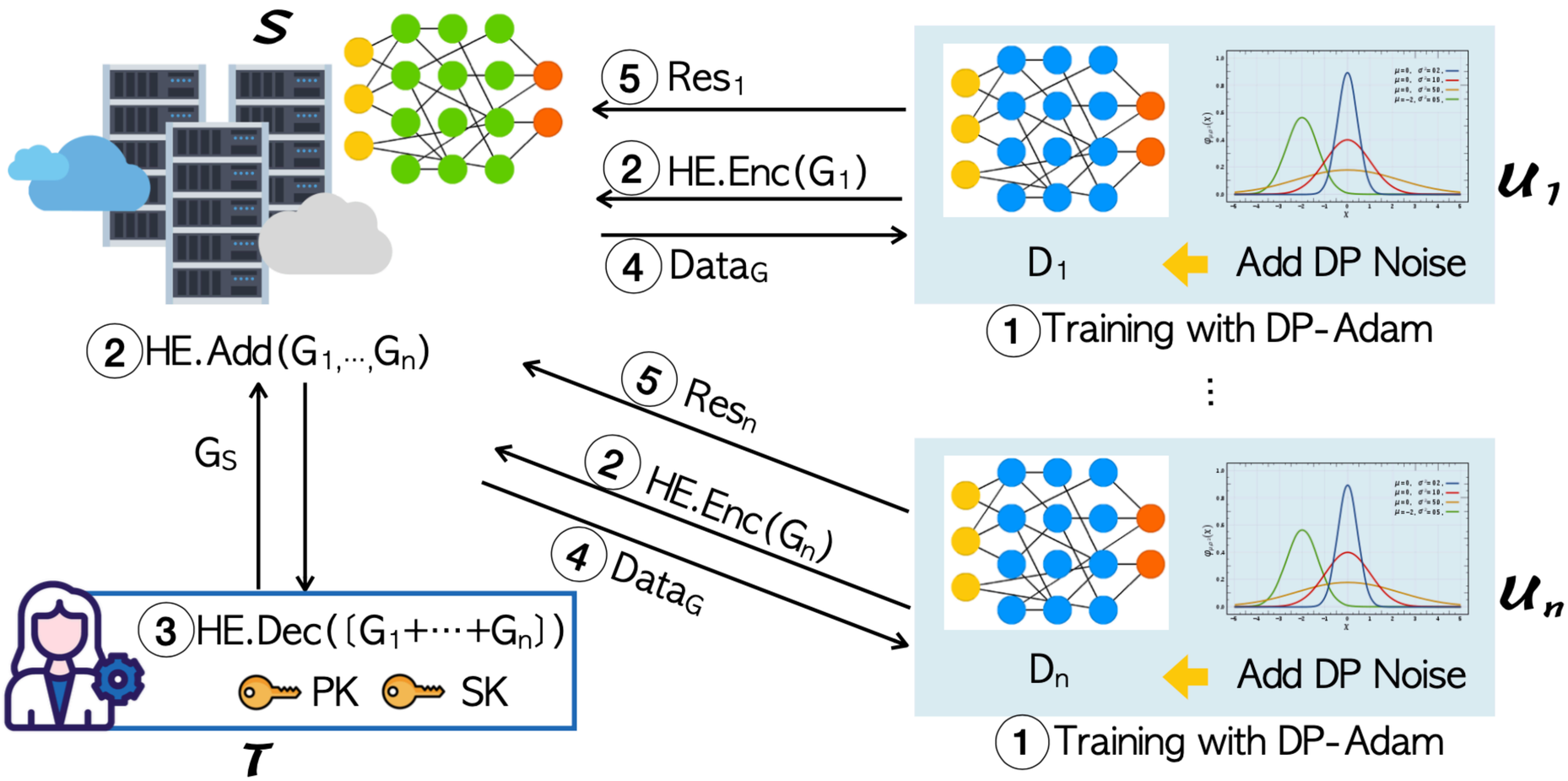


- User, Server,
- (Trust party)

- Integrate CKKS + DP+ federated learning

- 1: pretraining
- 2,3: phase 1
- 4,5: phase 2

Secure Federated GAN — Our Framework



- User, Server,
- (Trust party)

- Integrate CKKS + DP+ federated learning

- 1: pretraining
- 2,3: phase 1
- 4,5: phase 2

- Generalization with meta-learning
  - S holds a meta-leaner capturing short-term knowledge within a task and long-term knowledge among tasks

Experiments

- Commodity server: CentOS, 24-core CPU, 62.5GiB RAM, GeForce RTX 2080Ti GPU
- Datasets: MNIST, Mini-ImageNet
- Configuration: $\delta = 10^{-5}, \sigma = \{0.005, 0.5, 5, 8\}$
- Findings for DP training
  - Generally, DP-Adam converges more stable than DP-SGD.
  - Adding noise with lager σ increases the training instability, making the convergence slower
  - The noise "tolerance" depends on the parameter configuration.
- Communication costs compared with BatchCrypt
  - 73 vs 211GB (250 epochs)
  - 90 vs 1090 GB (100 epochs)

Conclusion

- Jump out of "folklore FL approach" — **Motivate further research**
  - **new thinking**
    - different types of privacy required in the FL process
  - **new route**
    - non-trivial combination of protective techniques
    - for a stronger protection

- Practical Hierarchical protection
  - (DP and crypto) in privacy-efficiency design space

- Potentiality of privacy-preserving application tailored with various learning **applications**