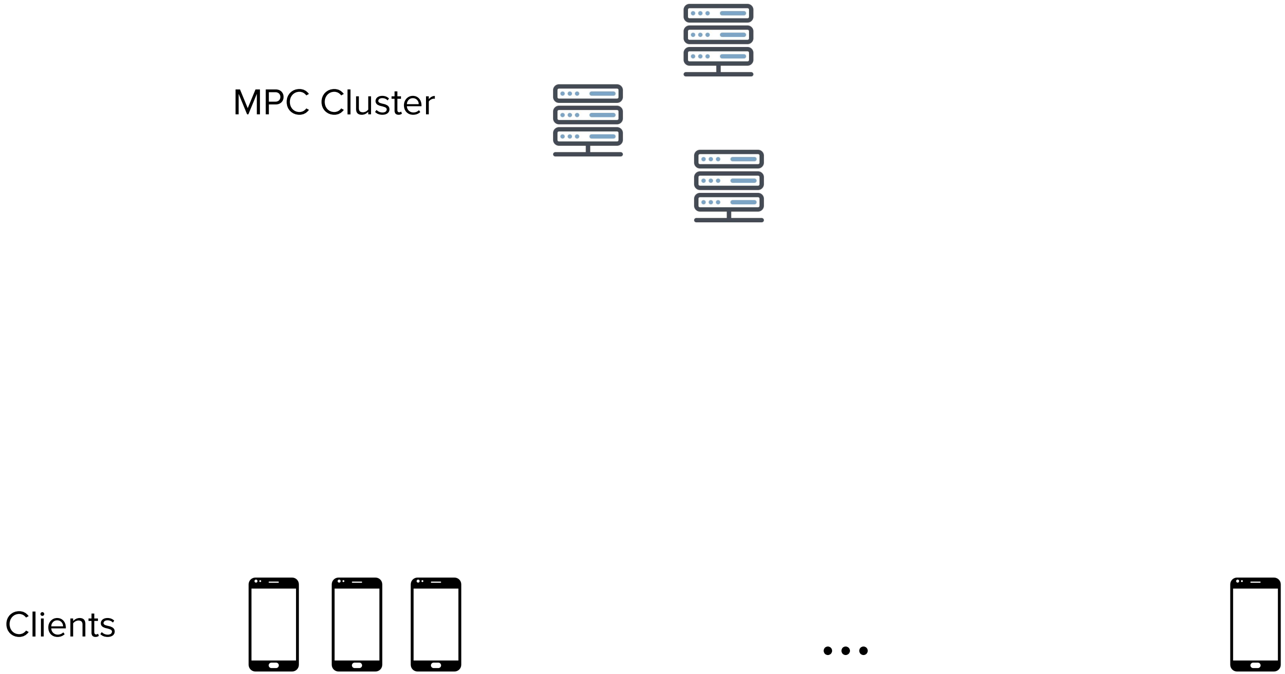
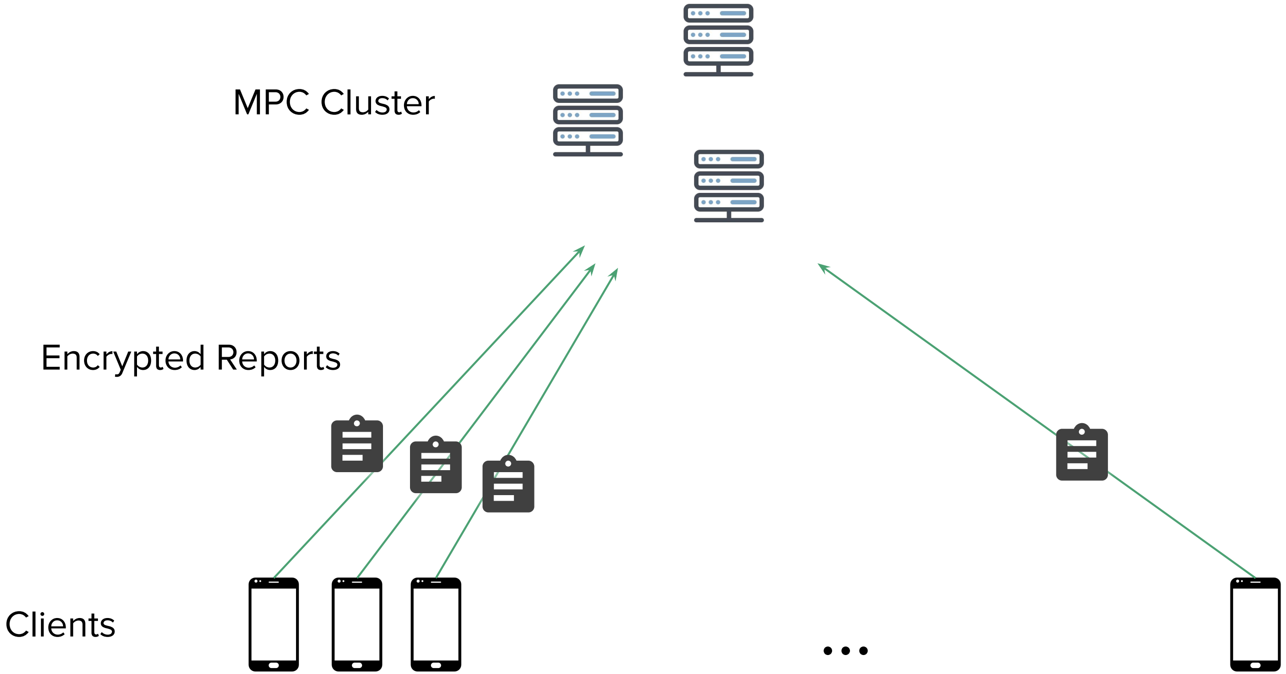
Secure Partitioning Protocols

Phillipp Schoppmann – IETF 116

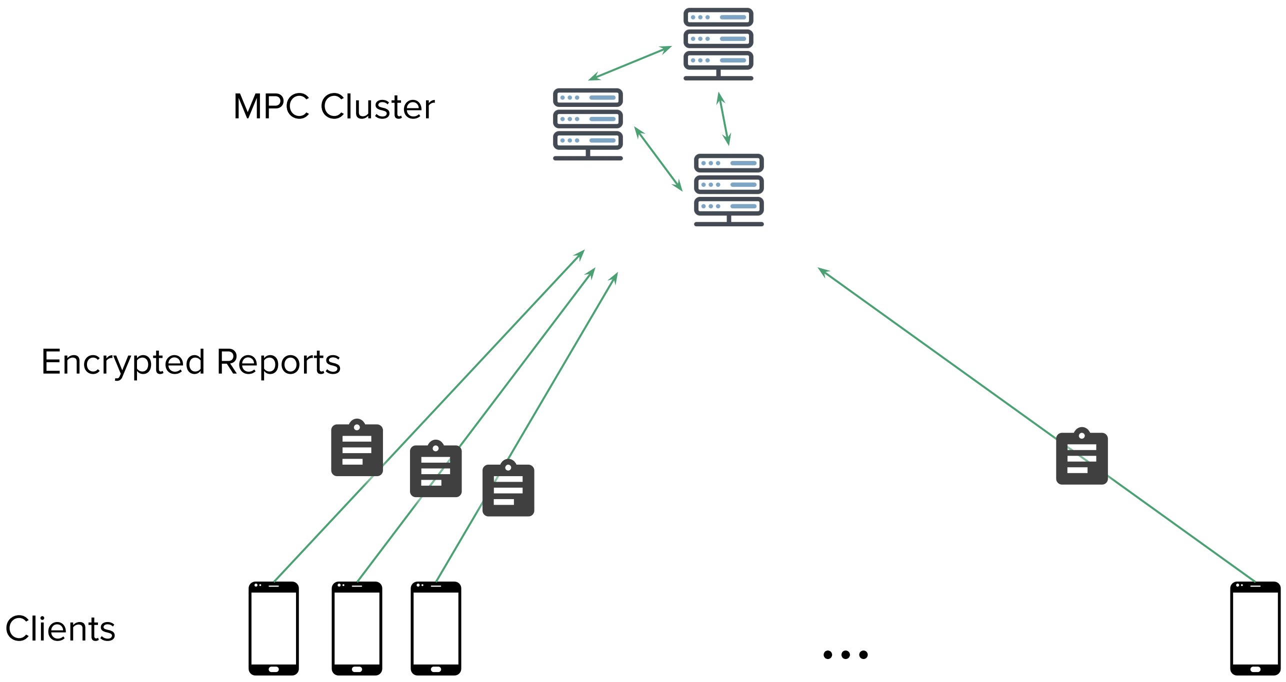
Background: Aggregate Statistics Measurements



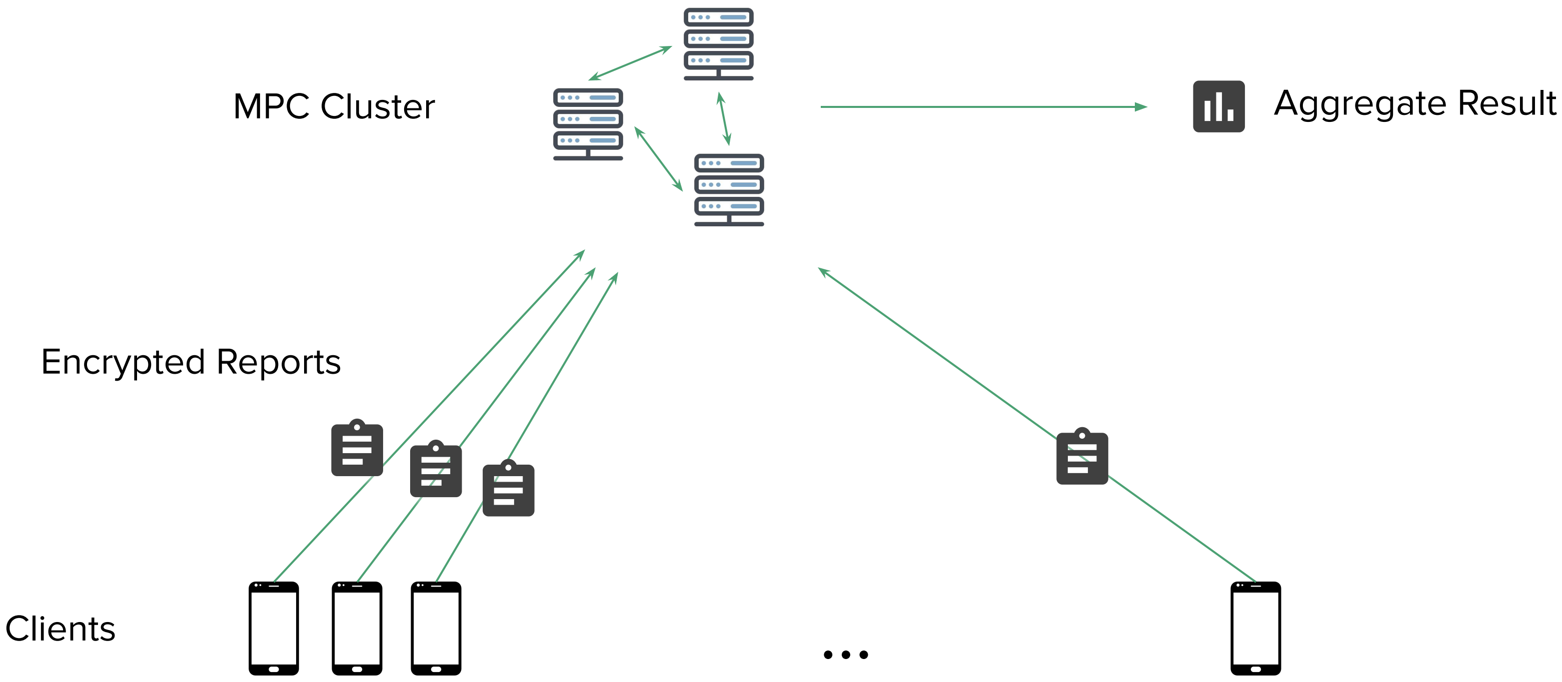
Background: Aggregate Statistics Measurements



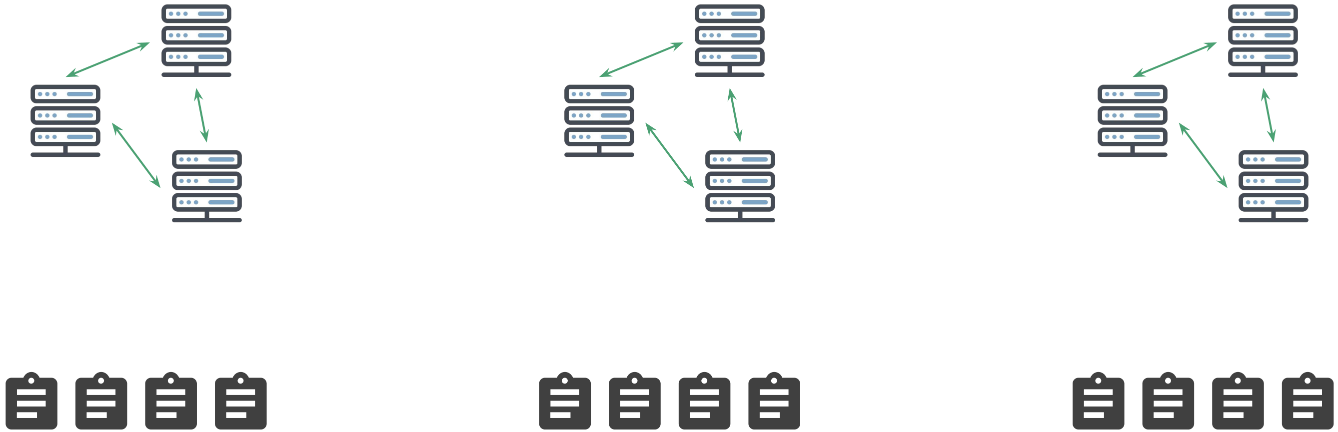
Background: Aggregate Statistics Measurements



Background: Aggregate Statistics Measurements



Sharding MPC Clusters



Challenge: How to partition reports across shards, s.t. all reports of the same client end up in the same shard?

Goals

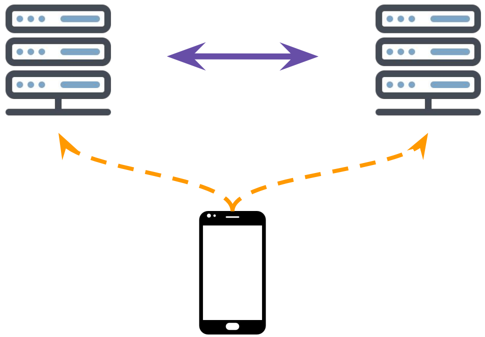
- Low overhead: Blow up communication per client by a *small* factor
- Low round complexity
- Partitioning must not affect correctness / utility of downstream computation

Assumptions

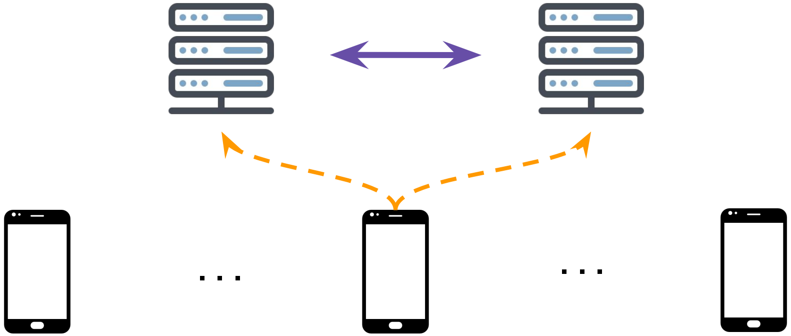
- Bound M on the number of contributions per client
- Lots of clients (billions), few shards (thousands)

Threat Model

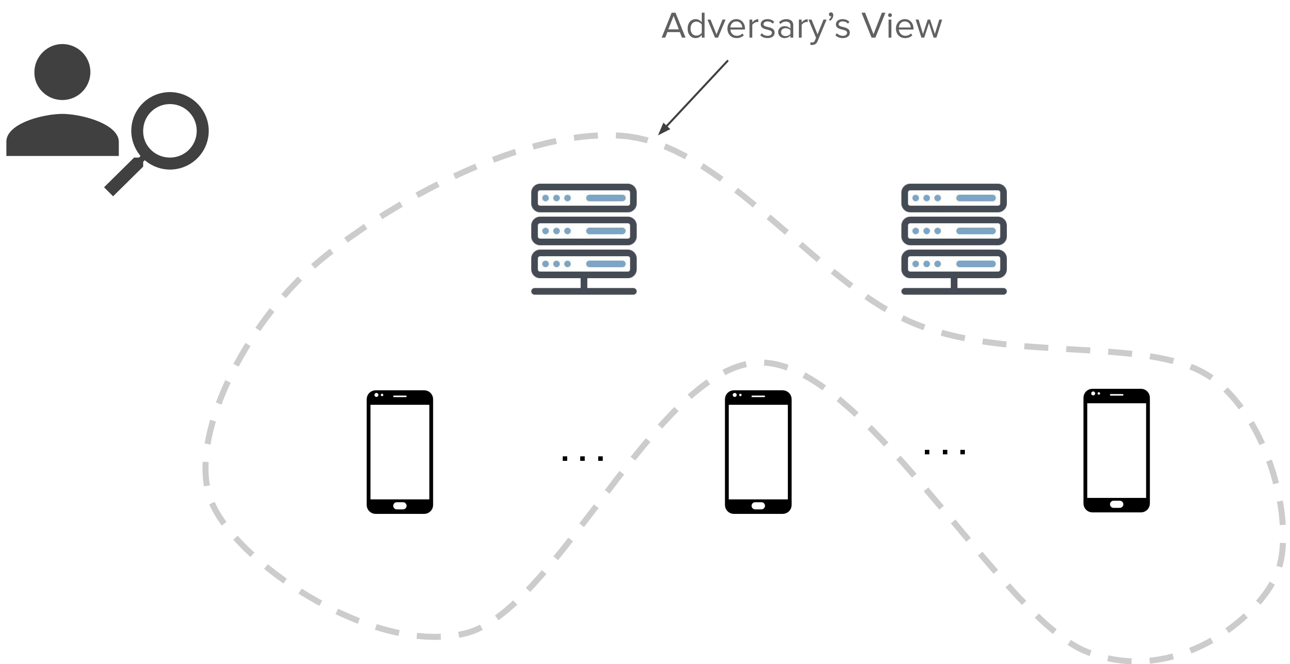
- Two (or more) non-colluding servers
- All parties are assumed to misbehave (as long as one server remains honest)
- Output of partitioning protocol must be *differentially private*



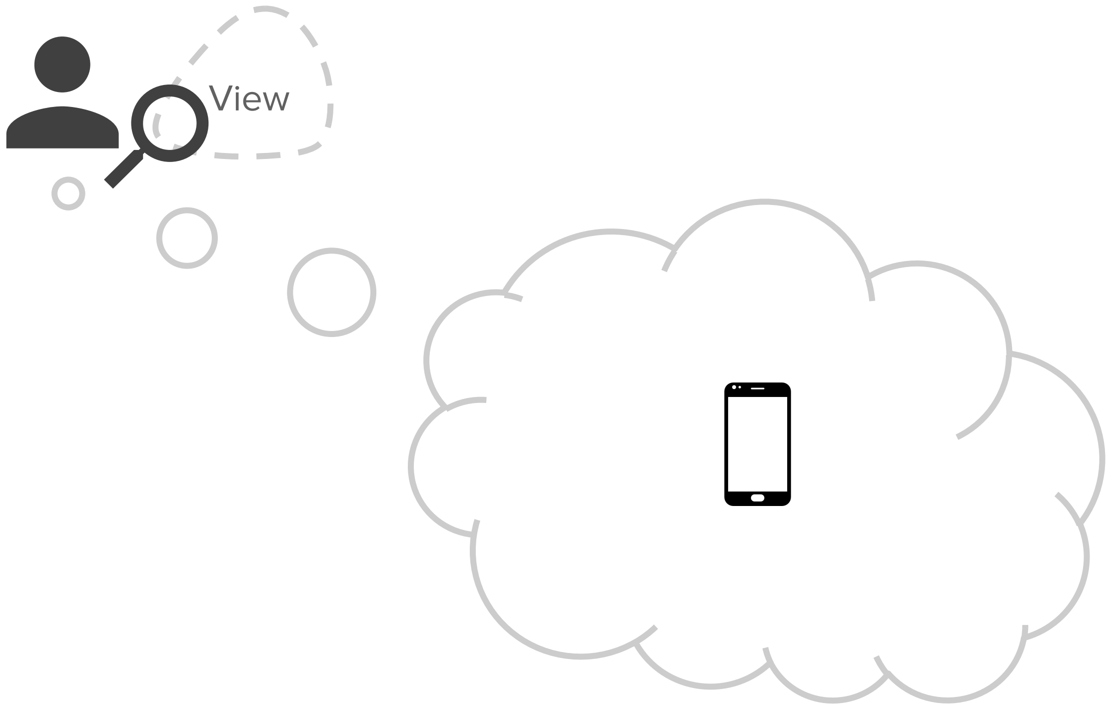
Threat Model



Threat Model



Threat Model

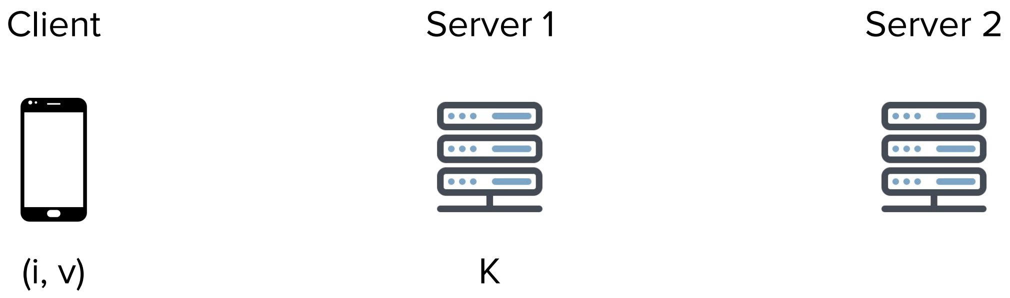


Differentially Private Views

$$\Pr(\text{View}) \leq \exp(\varepsilon) \cdot \Pr(\text{View}') + \delta$$

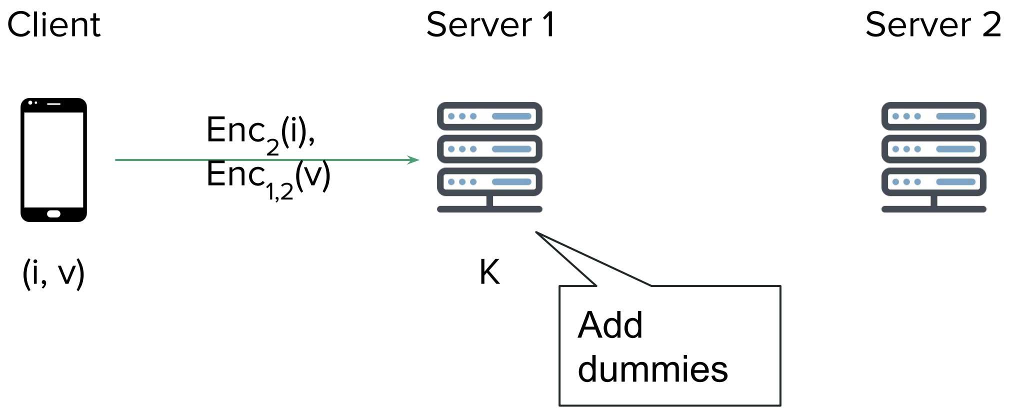
Client i's data changed

Blueprint: Partitioning from Distributed OPRFs



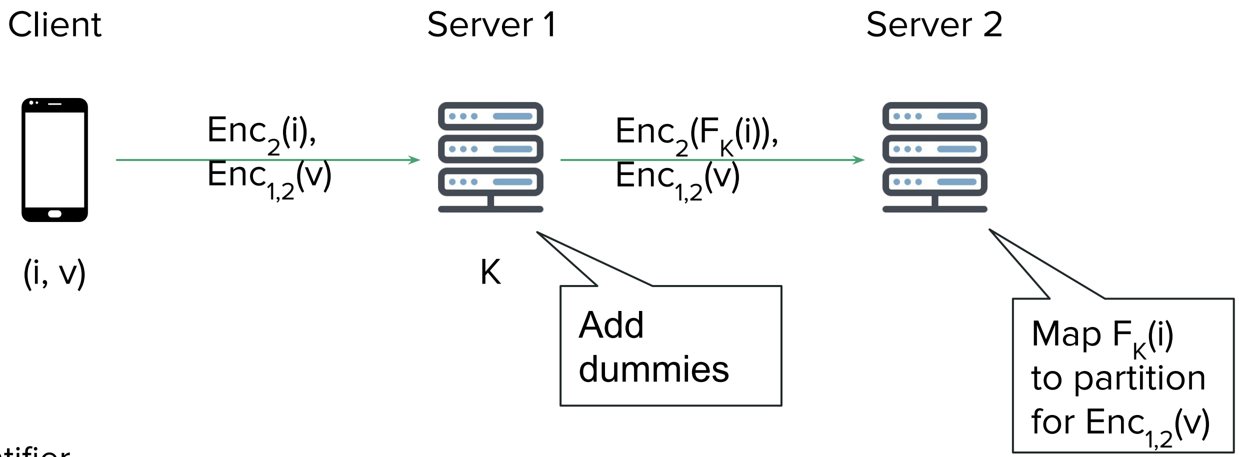
i: index / client identifier
v: value / payload

Blueprint: Partitioning from Distributed OPRFs



i: index / client identifier
v: value / payload
Enc: Encryption scheme that allows homomorphic evaluation of PRF, e.g. ElGamal or Dodis-Yampolski

Blueprint: Partitioning from Distributed OPRFs



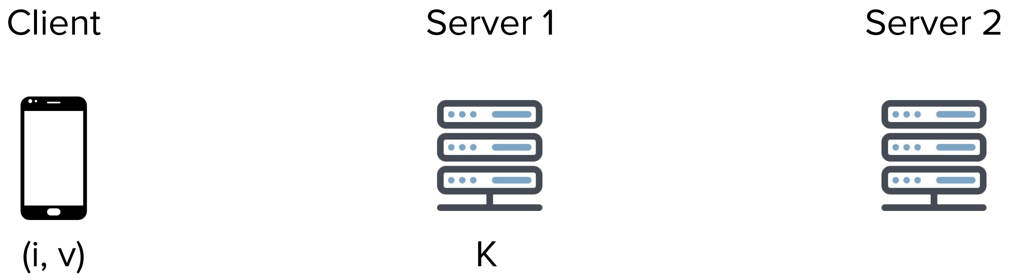
i: index / client identifier

v: value / payload

Enc: Encryption scheme that allows homomorphic evaluation of PRF, e.g. ElGamal or Dodis-Yampolski

Dense Partitioning: OPRF Output = Shard ID

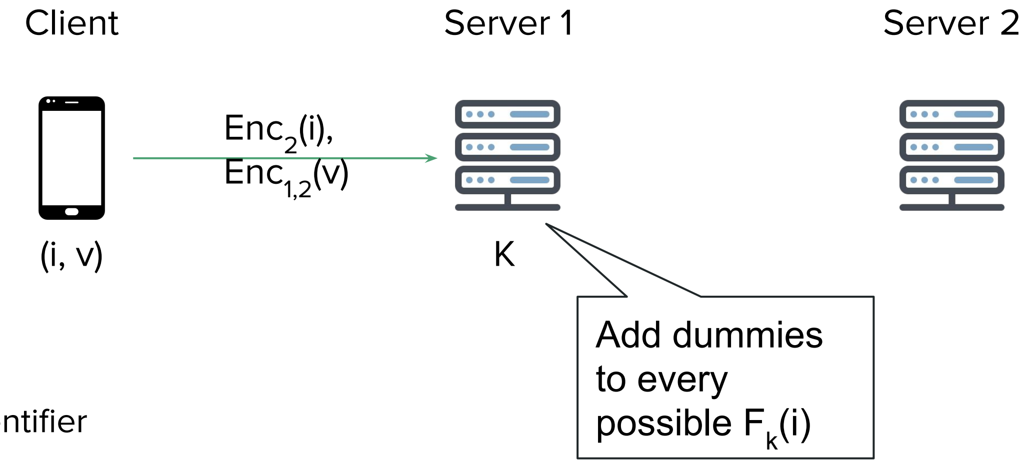
Assume there are exactly S shards, and let [S] be the range of F_K.



i: index / client identifier
v: value / payload
F_K: ID -> [S]

Dense Partitioning: OPRF Output = Shard ID

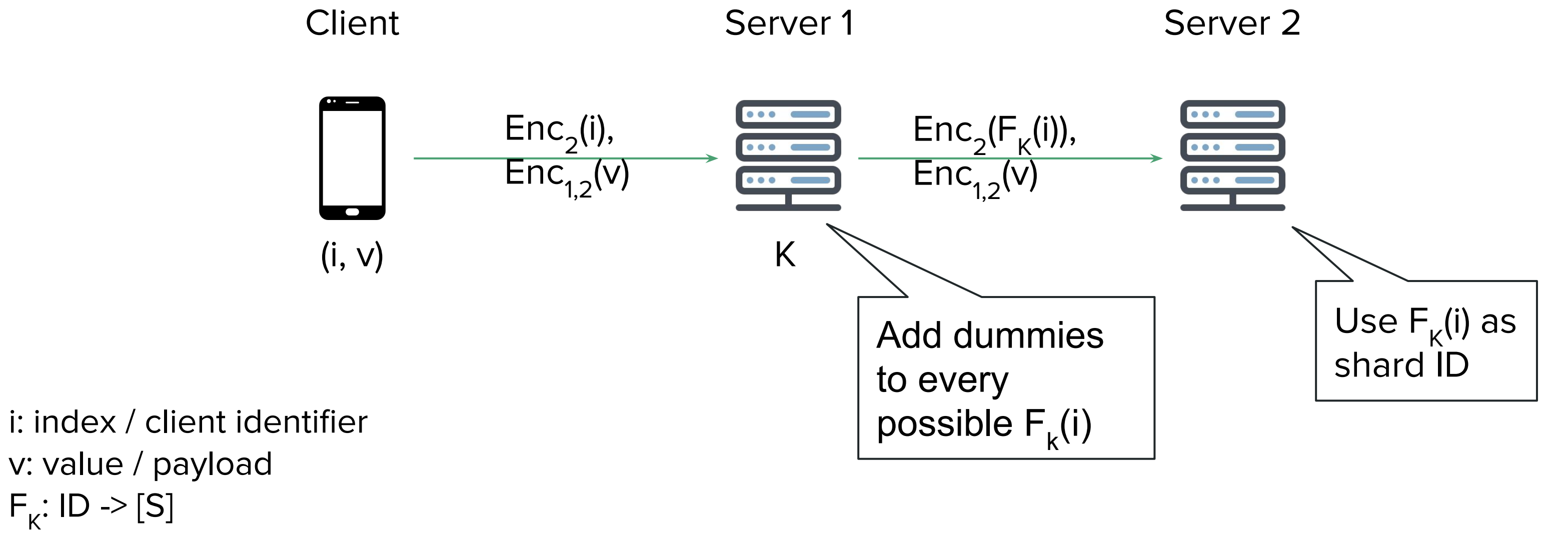
Assume there are exactly S shards, and let [S] be the range of F_K.



i: index / client identifier
v: value / payload
F_K: ID -> [S]

Dense Partitioning: OPRF Output = Shard ID

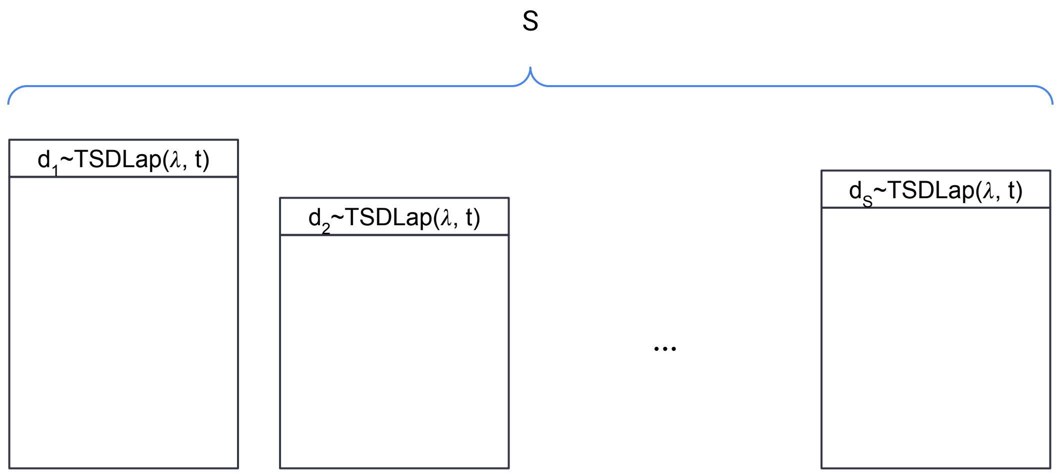
Assume there are exactly S shards, and let [S] be the range of F_K.



Dense Partitioning: Adding Dummies

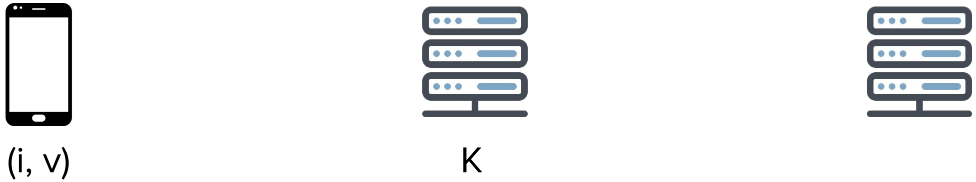
M: Upper bound on the number of ciphertexts with the same index / from the same client
S: Number of shards
TSDLap(λ, t): Truncated, shifted, discrete Laplace distribution with mean t and scale λ

Expected #dummies per bucket for $\varepsilon = 0.5$ and $\delta = 10^{-11}$: 49M per server



Sparse Partitioning: OPRF Output = Random Client ID

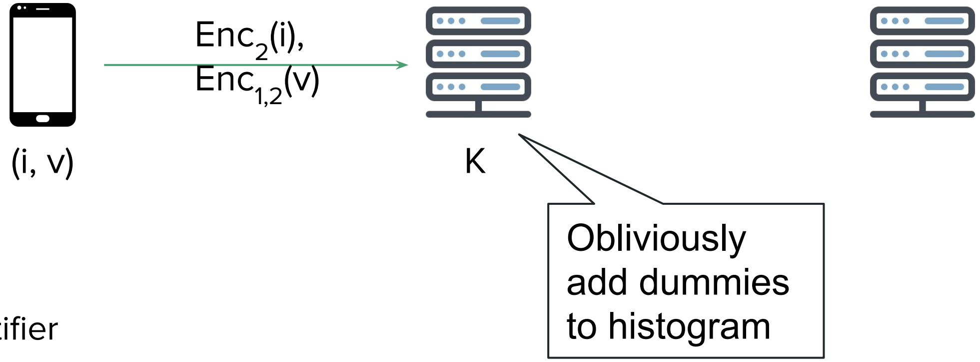
- If the OPRF codomain is large enough to make collisions unlikely, we can use the OPRF outputs as a pseudorandom client identifier.
- Allows *local* per-client aggregation (e.g., using Homomorphic Encryption)



i: index / client identifier
v: value / payload
F_K: ID -> $\{0,1\}^\sigma$

Sparse Partitioning: OPRF Output = Random Client ID

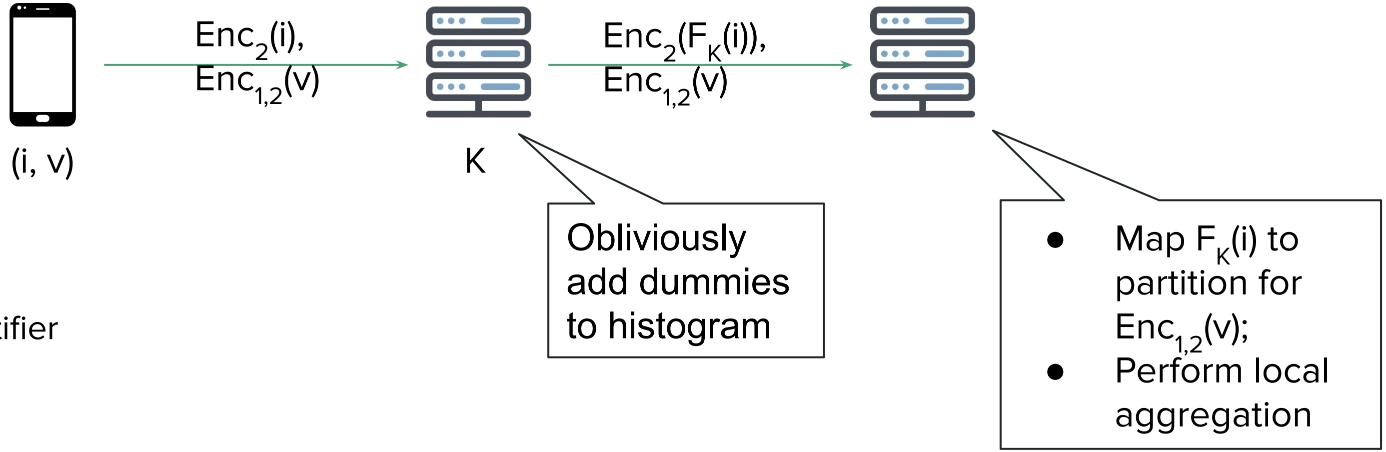
- If the OPRF codomain is large enough to make collisions unlikely, we can use the OPRF outputs as a pseudorandom client identifier.
- Allows *local* per-client aggregation (e.g., using Homomorphic Encryption)



i: index / client identifier
v: value / payload
F_K: ID -> $\{0,1\}^\sigma$

Sparse Partitioning: OPRF Output = Random Client ID

- If the OPRF codomain is large enough to make collisions unlikely, we can use the OPRF outputs as a pseudorandom client identifier.
- Allows *local* per-client aggregation (e.g., using Homomorphic Encryption)

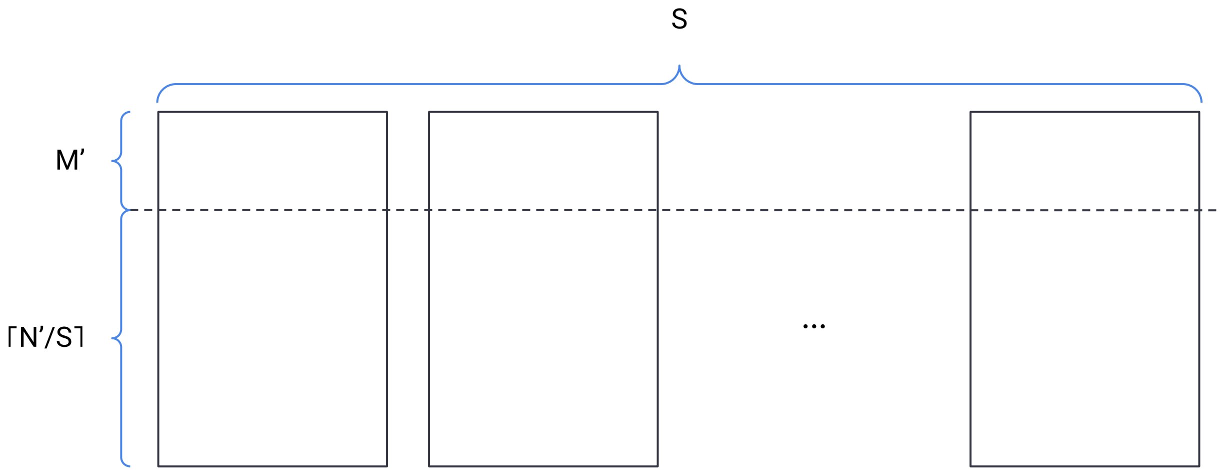


i: index / client identifier
v: value / payload
F_K: ID -> $\{0,1\}^\sigma$

Assigning Ciphertexts to Shards

N': Number of ciphertexts after adding dummies
M': Upper bound on the number of ciphertexts with the same index
S: Number of shards

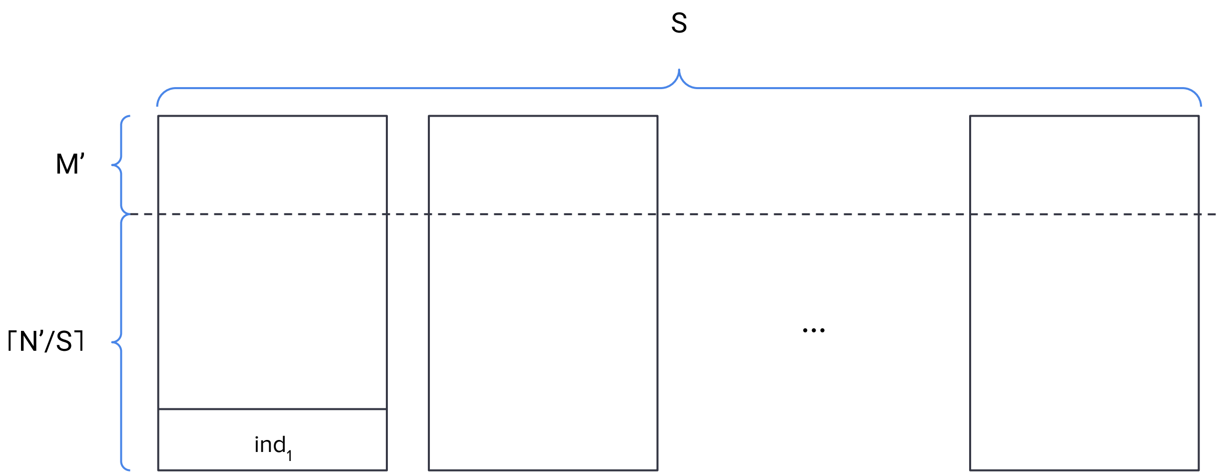
Observation: As long as $M' \ll \lceil N'/S \rceil$, the overhead will be small in practice.
But: N' might still be significantly larger than N. For $\varepsilon = 0.5$ and $\delta = 10^{-11}$, $N'/N = 1.1$



Assigning Ciphertexts to Shards

N’: Number of ciphertexts after adding dummies
M’: Upper bound on the number of ciphertexts with the same index
S: Number of shards

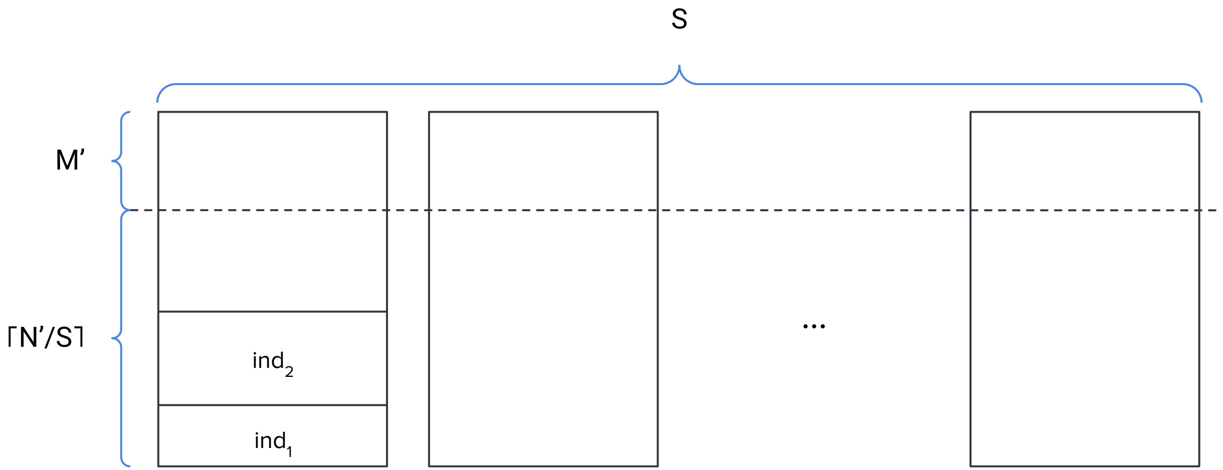
Observation: As long as M’ << ⌈N’/S⌉, the overhead will be small in practice.
But: N’ might still be significantly larger than N. For $\varepsilon = 0.5$ and $\delta = 10^{-11}$, N’/N = 1.1



Assigning Ciphertexts to Shards

N': Number of ciphertexts after adding dummies
M': Upper bound on the number of ciphertexts with the same index
S: Number of shards

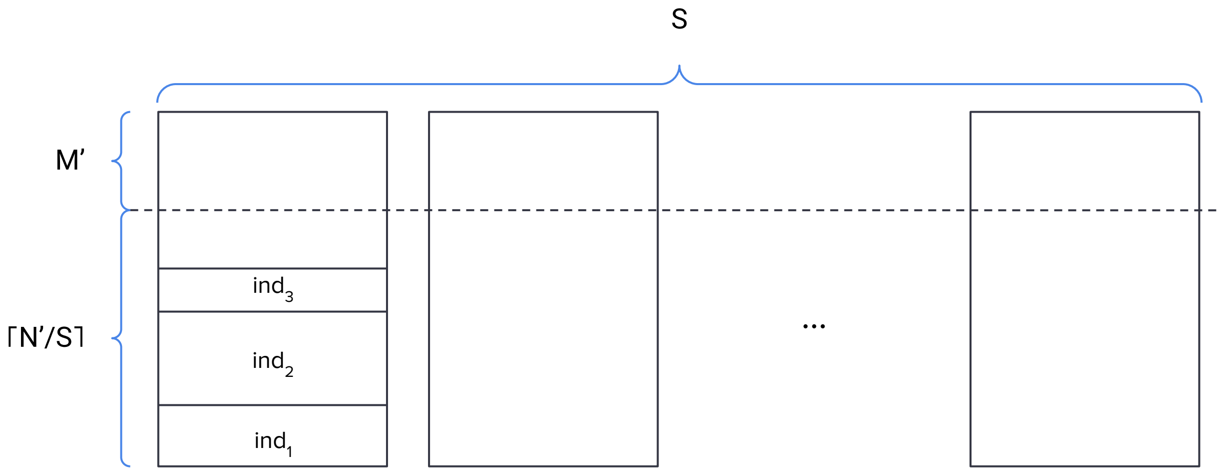
Observation: As long as $M' \ll \lceil N'/S \rceil$, the overhead will be small in practice.
But: N' might still be significantly larger than N. For $\varepsilon = 0.5$ and $\delta = 10^{-11}$, $N'/N = 1.1$



Assigning Ciphertexts to Shards

N’: Number of ciphertexts after adding dummies
M’: Upper bound on the number of ciphertexts with the same index
S: Number of shards

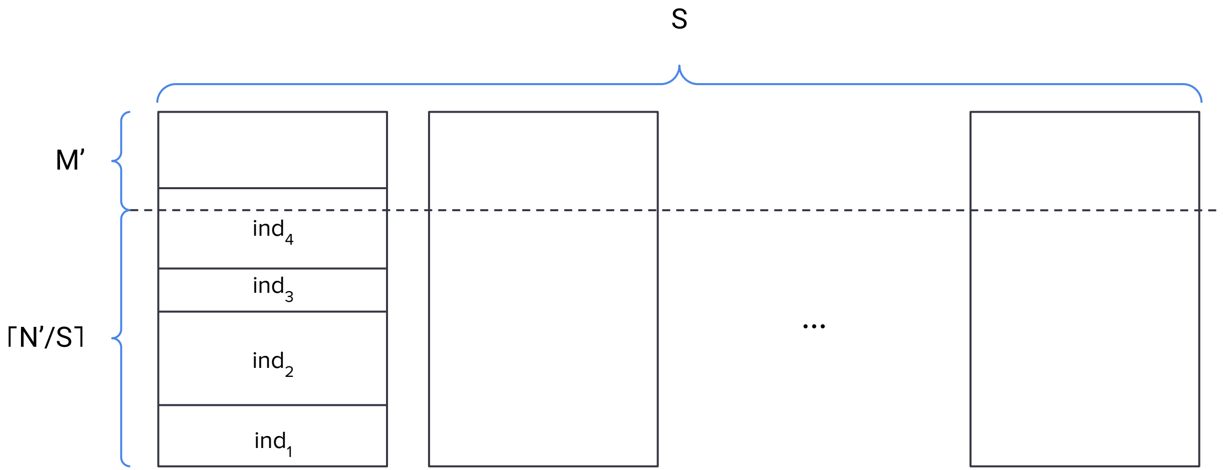
Observation: As long as $M' \ll \lceil N'/S \rceil$, the overhead will be small in practice.
But: N’ might still be significantly larger than N. For $\varepsilon = 0.5$ and $\delta = 10^{-11}$, $N'/N = 1.1$



Assigning Ciphertexts to Shards

N’: Number of ciphertexts after adding dummies
M’: Upper bound on the number of ciphertexts with the same index
S: Number of shards

Observation: As long as M’ << ⌈N’/S⌉, the overhead will be small in practice.
But: N’ might still be significantly larger than N. For $\varepsilon = 0.5$ and $\delta = 10^{-11}$, $N'/N = 1.1$



Conclusion

- Distributed OPRFs allow for efficient sharding protocols.
- When the number of shards is much smaller than the number of clients, the overhead is negligible.
- For a slightly larger (10%) overhead, we can enable local aggregation at one of the servers. Example application: Sparse histogram computation [1].

[1] Bell, James, Adrià Gascón, Badih Ghazi, Ravi Kumar, Pasin Manurangsi, Mariana Raykova, and Phillipp Schoppmann. "Distributed, Private, Sparse Histograms in the Two-Server Model." In Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, pp. 307-321. 2022.

Next Steps

- General interest from the working group in secure partitioning?
- Other protocols or settings where this might be useful?
- Do we need additional properties (e.g., keep the order of inputs)?