Doppio: Communication-Efficient and Secure Multi-Party Shuffle Differential Privacy

Wentao Dong
City University of Hong Kong
wentdong-c@my.cityu.edu.hk

Yang Cao
Institute of Science Tokyo
cao@c.titech.ac.jp

Cong Wang
City University of Hong Kong
congwang@cityu.edu.hk

Wei-Bin Lee
Hon Hai Research Institute; Feng Chia University
wei-bin.lee@foxconn.com

ABSTRACT

Modern database ecosystems increasingly process large-scale distributed user data, heightening the intrinsic tension between analytical utility and individual privacy. Shuffle differential privacy (*shuffle DP*) has recently emerged as a promising paradigm between the local and central models, offering favorable privacy-utility trade-offs by introducing a centralized, trusted shuffler. However, this architectural shift also poses new challenges in trust assumptions, system overhead, security risks, and workload limitations. To address them, we propose the *augmented multi-party shuffle DP* (AMP-SDP) model, which re-architects the data pipeline with a lightweight, versatile secret-shared intermediary layer. AMP-SDP (1) decentralizes trust while minimizing online communication costs; (2) provides structural security hardening against both shuffler compromise and user-side poisoning risks; and (3) augments shuffle DP for broader, more flexible workloads. Atop this model, we instantiate Doppio, a privacy-preserving crowdsourcing and data analytics framework. Our results show Doppio outperforms the state-of-the-art decentralized shuffle DP mechanism (Network Shuffling, SIGMOD'22) across many key metrics, affirming its effectiveness and efficiency in modern privacy-aware data management.

PVLDB Reference Format:
Wentao Dong, Yang Cao, Cong Wang, and Wei-Bin Lee. Doppio: Communication-Efficient and Secure Multi-Party Shuffle Differential Privacy. PVLDB, 19(2): 113 - 126, 2025.

doi:10.14778/3773749.3773752

PVLDB Artifact Availability:

The source code, data, and/or other artifacts have been made available at https://github.com/dongdongdoge/mpsdp.

1 INTRODUCTION

The proliferation of modern database ecosystems makes individual privacy a critical imperative, especially when collecting and analyzing vast amounts of distributed user data. Differential privacy (DP) [29] has emerged as a de facto standard to address this, providing rigorous, mathematically proven guarantees against privacy leaks. However, its two dominant paradigms—*central DP* and *local DP*—present a stark trade-off: the central model yields high utility but mandates user trust in a centralized curator, while the local model eliminates this trust yet incurs substantial utility degradation due to excessive overall noise.

Vanilla shuffle DP and its limitations. Recent *shuffle DP* model has emerged as a compelling alternative, offering favorable privacy-utility trade-offs [19, 34]. Its key idea, known as *privacy amplification by shuffling*, is to let a trusted shuffler anonymize user reports so that each user can apply weaker ε_0-local noise while the system collectively achieves a much stronger global ε'-DP guarantee, where $\varepsilon' \ll \varepsilon_0$. However, despite the theoretical appeal, its practical adoption still faces several unique challenges.

- *Trust-communication dilemma:* Vanilla shuffle DP relies on a centralized shuffler [10, 34], which reintroduces the very centralized trust point that local models seek to avoid. Existing distributed shuffle alternatives [58, 75], however, imply substantial communication costs, potentially negating shuffle model's key benefits and limiting its practicality at scale.
- *Unique security landscape:* As in local DP, the shuffle model remains vulnerable to data *poisoning attacks* [14, 18, 78], where adversarial users misreport to manipulate results. Critically, the very anonymity that amplifies honest users' privacy may also paradoxically amplify the impact of targeted, stealthy poisoning—a phenomenon we term *poisoning amplification*—by shielding malicious users. Furthermore, like the central DP curator, the shuffler itself is a single point of trust and potential failure [58]; its compromise could threaten privacy system-wide.
- *Structural model limitations:* By design, shuffle DP mandates the use of indistinguishable local randomizers [34], since any heterogeneity could be exploited to de-anonymize users and degrade privacy. In addition, the vanilla one-way shuffle DP architecture lacks native support for workloads involving interactive, adaptive queries [54, 70, 71]. To secure key privacy amplification in such two-way settings, user anonymity must persist throughout all interaction rounds, thus necessitating an extra mechanism to anonymously route curator responses backward.

These architectural and functional limitations prevent the vanilla shuffle model from serving as a drop-in replacement for local DP in real-world data systems, highlighting the need for a more efficient, resilient, and adaptable shuffle DP architecture.

Our approach: *augmented multi-party shuffle DP* (AMP-SDP). To bridge this gap, we introduce the AMP-SDP model. As depicted

*Cong Wang is the corresponding author.

Proceedings of the VLDB Endowment, Vol. 19, No. 2 ISSN 2150-8097.
doi:10.14778/3773749.3773752

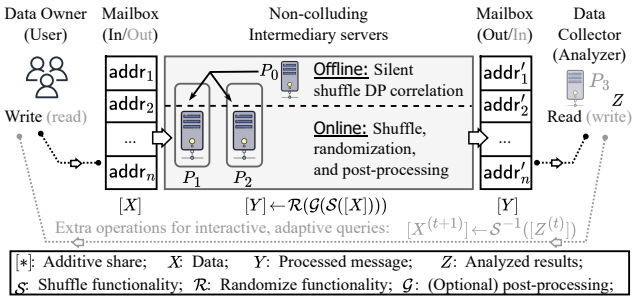


Figure 1: The architectural overview of AMP-SDP model.

in Figure 1, the model re-architects the vanilla shuffle DP pipeline with a lightweight, versatile *secret-shared intermediary* layer jointly maintained by a set of non-colluding servers.[1] This co-design of multi-party computation (MPC) and distributed shuffle DP protocols optimizes the trust-communication trade-off, enhances system security, and improves model adaptability. AMP-SDP model orchestrates a clean, modular workflow spanning input submission, secure processing, and result retrieval (and beyond). Cast in modern terms by Balle *et al.* [6], these intermediary servers jointly spread a *distributed privacy blanket*, amplifying each individual's privacy in a decentralized and cryptographically secure manner.

To showcase our techniques, we introduce and evaluate Doppio, an AMP-SDP-based privacy-preserving crowdsourcing and data analytics framework. Following similar designs [1, 22, 28], Doppio is envisioned to be operated by mutually distrusting yet reputable entities (e.g., big techs, research institutes, nonprofits) with both incentives and capabilities to securely collect and analyze distributed user data. Its potential use cases span learning emerging trends [10, 35], improving advertising strategies [2, 81], and advancing societal insights [2, 32]. We benchmarked Doppio over synthetic and realistic datasets [24, 65], showing it outperforms the state-of-the-art decentralized shuffle DP mechanism [58] across many key dimensions.

1.1 Technique Overview

A naïve starting point is to replace the centralized, trusted shuffler entity with a multi-server shuffle protocol. Yet this alone falls short in addressing the broader system-level challenges discussed above. We next outline our key technical contributions and rationales by an order of improving upon the above naïve approach.

Security-hardened shuffle DP architecture. Multi-server designs protect against single shuffler compromise concerns. However, akin to local DP [14], the vanilla shuffle DP remains structurally vulnerable to malicious user poisoning—with even greater risks due to anonymity-induced side effects. To counter this, we re-architect the conventional "local randomize-then-shuffle" pipeline, delegating the DP randomization to intermediary servers (detailed in §5). This architectural shift is guided by a key insight [18, 56]: directly faking the already-randomized message can be far more damaging than poisoning raw data alone. By eliminating direct user control over the randomization, our model confines adversarial influence to the less damaging level, where server-side DP mechanisms may further neutralize manipulations.

Low-communication secret-shared shuffle DP protocols. Securely performing shuffling and DP computations in a multi-party setting often incurs significant communication overhead. Existing MPC-based protocols mandate high inter-server communication that scales with the number of users n, data length ℓ, circuit depth h, and protocol repetition times B [16, 39]. To overcome this trade-off, we introduce a new cryptographic primitive termed *silent shuffle DP correlation* (see §4.1 and §4.3). This correlated randomness can be precomputed offline by the assisted server and enable a lightweight, non-interactive online protocol, where two non-colluding computing servers execute secure shuffling and randomization over secret-shared data using local computation only.

Augmented multi-party shuffle DP variants. Having a secure and communication-efficient foundation in place, we further augment our model with several useful extensions:

- *Privacy-boosting shuffle:* AMP-SDP integrates *sampleable* and *dummy-adding* shuffle variants, offering stronger privacy amplification potential with minimal overhead (see §4.2).
- *Reversible shuffle:* AMP-SDP provides native support for efficient, anonymous backward delivery of user-specific intermediate results—a vital capability for multi-round, adaptive query workload that vanilla shuffle DP has largely overlooked (see §4.2).
- *Fine-grained DP randomization:* AMP-SDP enables the application of non-uniform DP noise levels across records—akin to personalized DP [17], relaxing default indistinguishable randomizer constraints without new, alternative assumptions (see §4.4).

To streamline AMP-SDP's protocol orchestration, we introduce a *distributed virtual mailbox* abstraction (elaborated on in §3.2) that provides a stateful, secret-shared read/write interface, decoupling data transitions from the core computation logic to facilitate diverse database query workloads.

1.2 Comparison with Relevant DP Models

This work synthesizes concepts from two active research threads in the DP literature—shuffle DP and multi-party/distributed DP—to establish our unique design point. We next review these foundational areas to articulate the necessity of our new paradigm.

Shuffle DP. The idea of shuffle DP was initiated by Bittau *et al.* [10], and later formalized by Cheu *et al.* [19] and Erlingsson *et al.* [34]. They showed that secure shuffling can amplify privacy, reducing the real privacy budget from original ε_0 to $\varepsilon' = O(\frac{e^{\varepsilon_0}}{\sqrt{n}})$ for n users. Subsequent research has rapidly expanded the shuffle model along several dimensions. Tighter privacy amplification bounds were derived in [6, 37, 74]; new shuffle DP protocols tailored to specific analytical tasks were proposed [20, 40, 41, 73, 75]; and generalizations to Rényi shuffle DP [42] and more powerful multi-message models [7, 61] were developed. More recently, a few augmented shuffle DP techniques have been explored to strengthen robustness against inference or poisoning attacks [62, 64].

Complementing these theoretical advances, various system-level instantiations of the core shuffler have been proposed. Bittau *et al.* [10] leverage trusted hardware (e.g., Intel SGX), while Cheu *et al.* [19], Wang *et al.* [75], and Liew *et al.* [58] pursue decentralized

[1] AMP-SDP adopts a "2+1-party" (dealer) model, comprising two computing servers and an assisting server—a configuration widely employed in MPC [36, 44, 48, 50].

Table 1: Comparison of core shuffle protocols in Doppio and prior shuffle DP systems.

Ref.	Bittau *et al.* [10]	Balle *et al.* [5]	Cheu *et al.* [19]	Wang *et al.* [75]	Liew *et al.* [58]	Ours
Shuffle tech.	oblivious sorting	alternating shuffle	mix-nets	encrypted shuffle	network shuffling	silent shuffle
Shuffle type[†]	uniform	approximate	uniform	uniform	approximate	uniform (generalized)
Trust model	centralized	centralized	distributed	distributed	distributed	distributed
Communication[‡]	N/A	N/A	$O(Ln\ell)$	$O(n\ell + n\lambda)$	$O(cn \log n\ell)$	0

[†] Uniform shuffle protocols enable known optimal privacy amplification capabilities [6, 37, 74], whereas approximate shuffle schemes offer weaker guarantees.
[‡] Refers to the inter-server/inter-user communication during secure shuffling (excluding shuffler-curator communication). N/A denotes no such cost for the centralized shuffler systems. Variables L, ℓ, λ, c denote mix-net chain length, per-message size, security parameter, and network topology parameter, respectively.

Table 2: Comparison of DP sampling and randomization strategies in Doppio and prior multi-party/distributed DP systems.

Ref.	Keller *et al.* [51]	Roy *et al.* [67]	Fu *et al.* [39]	Heikkila *et al.* [45]	Liew *et al.* [58]	Ours
Sampling tech.[†]	2-party DP	2-party DP	2-party/distributed DP	distributed DP	distributed DP	2+1-party DP
Mechanism[‡]	additive	additive	additive	additive (infinitely divisible)	generic	generic
Cost model[♮]	$O(h)+O(1)$	$O(h)+O(1)$	$O(h)+O(1)$	$O(1)+O(1)$	$O(1)+O(1)$	$O(1)+O(1)$

[†] Specifies how DP noise is generated (sampling). In our design, the noise is modeled as a form of *correlated randomness*, which can be precomputed offline by the trusted dealer.
[‡] Specifies how DP noise is applied (randomization). To enable a general, mechanism-agnostic silent DP execution flow, we intentionally classify DP mechanisms based on their underlying arithmetic strategy: additive (e.g., Laplace) versus non-additive (e.g., k-RR).
[♮] We slightly abuse the asymptotic notation to represent the per-instance multi-party DP cost as $a + b$, where a captures the offline correlated DP randomness generation cost (e.g., $O(h)$ denotes the noise sampling circuit depth), and b captures the multi-party DP application cost.

alternatives using mix-nets or distributed shuffle protocols. To further improve performance, several works investigate reducing the uniform, oblivious shuffle to approximate variants [5, 43, 58, 77, 80], albeit at the cost of weaker privacy amplification guarantees.

Multi-party DP and distributed DP. In parallel, a large body of literature explores the integration of DP with distributed protocols. Many systems emulate the central model by executing both analytics and DP pipeline in MPC frameworks, enabling end-to-end secure and differentially private data analysis [11, 15, 21, 28, 31, 33, 67, 76]. While offering favorable trust boundaries, these approaches often suffer from high overhead, especially for complex workloads or when run with small privacy budgets and large user scales.

Distributed DP techniques, in contrast, typically adhere to the local DP flow [45, 47, 66, 68, 69, 79]. In this setup, each user independently masks their input (not necessarily satisfying ε_0-DP for each individual submission), and an untrusted aggregator collects the reports to compute an approximate ε_0-DP aggregate. These designs, however, are vulnerable to data poisoning from malicious users [14, 18, 72, 78] and exhibit significant utility loss.

Beyond these decentralized approaches, another natural strategy is to substitute the trusted shuffler with distributed shuffle protocols [19, 58, 75, 77]. However, as summarized in Table 1 and Table 2, all these approaches often entail high communication overhead and fail to holistically address the unique security and functionality challenges of the shuffle DP paradigm. These critical gaps jointly motivate the design of our AMP-SDP model.

2 PRELIMINARIES

Differential privacy. For clarity, we use LDP, CDP (or DP), and SDP to refer to local, central, and shuffle DP, respectively. [2]

Definition 2.1 ((ε, δ)-DP [30]). *Given non-negative parameters ε and δ, a randomized mechanism $\mathcal{R} : \mathcal{X} \to \mathcal{Y}$ satisfies (ε, δ)-DP if, for any neighboring datasets $X, \bar{X} \in \mathcal{X}^n$ differing in one element and for any $Y \in \mathcal{Y}^n$, the following holds:*

$$\Pr[\mathcal{R}(X) \in Y] \leq e^{\varepsilon} \cdot \Pr[\mathcal{R}(\bar{X}) \in Y] + \delta.$$

Conventionally, smaller ε indicate stronger privacy guarantees, and $\delta \geq 0$ denotes the probability that privacy is violated. If X contains only one element, we call $\mathcal{R}$ as a local randomizer:

Definition 2.2 (Local randomizer [30]). *A mechanism $\mathcal{R} : \mathcal{X} \to \mathcal{Y}$ is (ε, δ)-LDP if, for all $x_i, x_j \in \mathcal{X}$, the outputs $\mathcal{R}(x_i)$ and $\mathcal{R}(x_j)$ are (ε, δ)-indistinguishable.*

Randomization mechanisms. As Table 2 shows, we classify differentially private mechanisms into two types—additive and non-additive—to guide our silent multi-party DP protocol designs. This work considers the Laplace mechanism and k-ary randomized response (k-RR) as representatives. Other more complex mechanisms, such as local hashing [8], can be derived similarly.

In the additive Laplace mechanism, the output y is the sum of the raw input x_i and independent noise r_i:

$$y = \mathcal{R}(x_i) = x_i + r_i, \quad \text{where } r_i \sim \mathsf{Lap}^d_{\varepsilon_0}(\lambda = \Delta_1(f)/\varepsilon_0).$$

In the non-additive k-RR, the output y is either the raw input x_i or replaced with a uniformly sampled dummy value:

$$y = \mathcal{R}(x_i) = \begin{cases} x_i & \text{w/ probability } p = \frac{e^{\varepsilon_0}}{e^{\varepsilon_0}+k-1}, \\ r_i \sim \mathsf{Uni}(k) & \text{w/ probability } p = \frac{1}{e^{\varepsilon_0}+k-1}. \end{cases}$$

Privacy amplification via shuffling. Due to DP's post-processing property, n independent ε-LDP messages remain ε-DP under central aggregation. However, shuffling further enhances privacy.

Definition 2.3 (Shuffle DP [6, 19, 34, 37]). *Given a local randomizer $\mathcal{R} : \mathcal{X} \to \mathcal{Y}$, the message set $Y \in \mathcal{Y}^n$, a random permutation $\mathcal{S} : \mathcal{Y}^n \to \mathcal{Y}^n$, and an analytics function $\mathcal{A} : \mathcal{Y}^n \to \mathcal{Z}$, the shuffle DP functionality is defined as:*

$$\mathcal{F}_{\mathsf{SDP}} := \mathcal{A}(\mathcal{S}(\mathcal{R}(x_1), \ldots, \mathcal{R}(x_n))).$$

[2] For simplicity, we use set notation for inputs in SDP, though the elements are ordered.

Table 3: Error bounds under poisoning attacks in ε_0-LDP [18].

Ref.†	frequency	histogram	degree-d Stat.
No poisoning	$\Theta\left(\sqrt{\frac{d^2}{\varepsilon_0^2 n}}\right)$	$\Theta\left(\sqrt{\frac{\log d}{\varepsilon_0^2 n}}\right)$	$\Theta\left(\sqrt{\frac{d \log d}{\varepsilon_0^2 n}}\right)$
IPA	$\Omega\left(\frac{n_c}{n} \cdot \frac{\sqrt{d}}{\varepsilon_0\sqrt{\log n}}\right)$	$\Omega\left(\frac{n_c}{n} \cdot \frac{1}{\varepsilon_0}\right)$	$\Omega\left(\frac{n_c}{n} \cdot \frac{1}{\varepsilon_0}\right)$
OPA	$O\left(\frac{n_c}{n} \cdot \frac{\sqrt{d}}{\varepsilon_0}\right)$	$O\left(\frac{n_c}{n} \cdot \frac{\log d}{\varepsilon_0}\right)$	$O\left(\frac{n_c}{n} \cdot \frac{1}{\varepsilon_0}\right)$

† n: number of users; n_c: number of malicious users; d: domain size.

Here, $\mathcal{S}$ rearranges the multi-set of randomized user messages, anonymizing the origin of each. This model achieves privacy amplification over LDP, with an approximate bound below [74]:

$$\varepsilon' = O\left(\sqrt{\frac{\beta(e^{\varepsilon_0}-1)\log(1/\delta)}{n}}\right),$$

where ε_0 is $\mathcal{R}$'s privacy parameter, and $\beta \in \left[0, \frac{p-1}{p+1}\right]$ with $p = e^{\varepsilon_0}$.

Malicious user poisoning attacks. Local DP mechanisms are vulnerable to adversarial manipulation.

Definition 2.4 (Poisoning attacks [14, 18]). *Given a raw data x and randomized message $y = \mathcal{R}(x)$, an adversarial user may launch:*

- *Input poisoning attack (IPA): misreport $\mathcal{R}(x_c)$ where $x_c \neq x$.*
- *Output poisoning attack (OPA): bypass $\mathcal{R}$ and directly fake arbitrary message $y_c \neq \mathcal{R}(x)$.*

Table 3 shows that OPA typically results in more severe accuracy degradation than the trivial IPA.

Multi-party computation and shuffle. An MPC protocol allows m non-colluding servers to jointly compute a function f over encrypted user data without learning input privacy. For the common 2-party setting, we define a value x is $[x]$-shared if $x = [x]_1 + [x]_2$. Modern MPC frameworks usually adhere to an offline-online model, where servers precompute some forms of input-independent *correlated randomness* [46] for better online efficiency and simplicity.

Chase *et al.* [16] present a 2-party shuffle paradigm to permute a dataset X. The protocol relies on offline-generated *shuffle correlation* that embeds a private permutation π, followed by an interactive online phase between P_1 and P_2 that securely executes:

$$X' = \pi(X) = \sum [\pi(X)] = \sum [X'].$$

Subsequent works extend this to 2+1-party settings for improved efficiency [3, 36]. Another line of work models shuffling as matrix-vector multiplication [25, 60], exploring different trade-offs.

Multi-party DP noise sampling and randomization. DP techniques were originally built over continuous domains, while MPC often works in finite fields/rings. To reconcile this gap, some research introduced the notion of distributed computational DP:

Definition 2.5 (Distributed computational DP [9]). *Given security parameter κ, polynomial-time distinguisher D, a protocol Π satisfies $(\varepsilon, \delta + \mathsf{negl}(\kappa))$-distributed computational DP if*

$$\Pr[D(\mathsf{View}_\Pi^{\mathsf{Adv}}(X)) = 1] \leq e^\varepsilon \cdot \Pr[D(\mathsf{View}_\Pi^{\mathsf{Adv}}(\bar{X})) = 1] + \delta + \mathsf{negl}(\kappa).$$

The extra $\mathsf{negl}(\kappa)$ term accounts for cryptographic failure probability. A 2-party noise sampling protocol outputs $[r]_i$ such that $r = [r]_1 + [r]_2$ and r satisfies $(\varepsilon, \delta + \mathsf{negl}(\kappa))$-DP. With secret-shared user input $[x]$, randomization proceeds as:

$$y = \mathcal{R}(x) = \begin{cases} x + r = \sum([x] + [r]), & \text{Additive (Laplace)} \\ \mathsf{Rand}(x, r) = \sum \mathsf{Rand}([x], [r]), & \text{Non-additive } (k\text{-RR}) \end{cases}$$

Interactive and adaptive analytics. Modern database systems increasingly feature adaptive, multi-round queries, such as personalized query refinement, iterative training, and exploratory OLAP workloads [23, 54, 70, 71].

In each round t, the curator provides individualized feedback $Z^{(t)} = \mathcal{A}^{(t)}(X^{(t)})$. Each user then refines their next-round input by evaluating $x_i^{(t+1)} = \mathsf{Update}_i(x_i^{(t)}, z_i^{(t)})$. Such queries pose structural challenges for existing one-way shuffle DP pipeline.

3 AMP-SDP MODEL

Goals and non-goals. AMP-SDP re-architects and augments the vanilla shuffle model with three overarching objectives: (1) decentralize trust while minimizing communication costs; (2) enhance system robustness against adversaries; and (3) address structural model limitations for broader use cases. Meanwhile, AMP-SDP does not seek to derive tighter amplification bounds, and instead adheres to established privacy guarantees [6, 37, 74]. Nor is AMP-SDP tied to any specific mechanism or workload. Rather, it is designed in a mechanism- and application-agnostic, extensible manner for better integrating shuffle DP into diverse database applications.

3.1 Threat Model

AMP-SDP consists of four main logical roles:

- *Users (C)* may include malicious clients capable of misreporting their raw inputs (e.g., poisoning attacks).
- *Computing servers (P_1 and P_2)* are *honest-but-curious*: they follow the protocol specification but may attempt to infer sensitive information from users from local views.
- *Assisting server (P_0)* functions as a semi-trusted offline dealer only, responsible for generating and distributing correlation.
- *Curator server (P_3)* retrieves and aggregates the processed outputs, having no access to raw inputs or intermediate states.

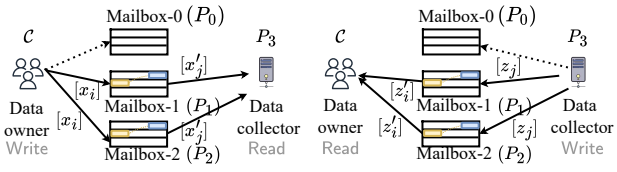
Consistent with prior MPC systems [1, 22, 28, 36, 75], we assume that all intermediary servers (P_0, P_1, P_2) are mutually *non-colluding*, and none colludes with the curator P_3. These reflect a common federated deployment scenario across independent, reputably-separated organizations (e.g., big techs, research institutes, nonprofits, and regulators). In §4.2, we will discuss how to relax this.

3.2 Communication Model

Figure 2 illustrates the message flow among the main entities.

Distributed virtual mailbox. To streamline multi-party protocol designs, AMP-SDP harnesses a *distributed virtual mailbox* T across intermediary servers, with each P_j maintaining its local holdings T^j. Upon registration, each C_i is assigned a unique index i, which maps to dedicated slots (T_i^0, T_i^1, T_i^2) residing at P_0, P_1, and P_2, respectively. This abstraction provides a stateful read/write interface, decoupling communication flow from the core computation logic.

Message forward and (optional) response backward. For data submission, user C_i simply writes its secret-shared input into slots



(a) Client data forward **(b) Curator response backward**

Figure 2: Overview of AMP-SDP's distributed virtual mailbox abstraction and its communication flow.

T_i^1 and T_i^2. After secure processing, the curator P_3 retrieves and reconstructs the final outputs from T_*^1 and T_*^2, thereby completing the message forward pipeline.

Mirroring this path, AMP-SDP enables anonymous response delivery through a customized *reversible shuffle* mechanism, where P_3 computes and writes user-specific responses back to the same physical slots read during the forward stage. Then after reversible shuffling, user C_i can retrieve its unique, private response shares by reading from the exact slots T_i^1 and T_i^2 it originally wrote to—achieving $O(1)$ lookup complexity. This design efficiently provides the persistent two-way anonymity required for multi-round, adaptive data analytics within the shuffle DP model.

3.3 Ideal Functionality and Design Choices

Definition 3.1 (Augmented multi-party shuffle DP). *Let $\mathcal{C} = \{C_1, \ldots, C_n\}$ be the user set; $\mathcal{P} = \{P_1, P_2\}$ the two non-colluding computing servers; P_0 the assisting server; and P_3 the curator. AMP-SDP functionality proceeds as follows:*

- *Offline: All entities P_0, P_1, P_2, P_3, and $\mathcal{C}$ together initialize the system; $\mathcal{C}$ register; P_0 generates and distributes the data-independent shuffle DP correlation* CR.
- *Online (forward): $\mathcal{C}$ submit input shares $[X] = \{[x_1], \ldots, [x_n]\}$ to $\mathcal{P}$, which securely perform: (1) uniform shuffle $\mathcal{S}$; (optional) post-processing $\mathcal{G}$; and (3) (relaxed) randomization $\mathcal{R}_+$. P_3 then retrieves output shares $[Y] = \{\mathcal{R}_1(\mathcal{G}([x_{\pi(1)}])), \ldots, \mathcal{R}_n(\mathcal{G}([x_{\pi(n)}]))\}$, reconstructs Y, and applies analytics algorithm $\mathcal{A}$, denoted as:*

$$\mathcal{F}_{\text{AMP-SDP}} := \mathcal{A}\big(\mathcal{R}_+\big(\mathcal{G}\big(\mathcal{S}([X])\big)\big)\big).$$

- *Online (backward): For interactive, adaptive queries, $\mathcal{P}$ anonymously route responses $Z = [z_1, \ldots, z_n]$ back to users via the inverse shuffle $\mathcal{S}^{-1}([Z])$ to facilitate the next computation round.*

This functionality captures the architectural blueprint of AMP-SDP. We next examine the rationales behind our design choices.

Why multi-party setup. AMP-SDP adopts a 2+1-party architecture to decentralize trust and eliminates single-shuffler failure concerns—also a key goal in recent shuffle DP research [58, 75].

Why generalized shuffle variants. By composing shuffle $\mathcal{S}$ with (optional) post-processing $\mathcal{G}$, AMP-SDP yields generalized multi-party shuffle variants, offering better privacy guarantees [55, 57, 64].

Why shuffle-then-randomize pipeline. Departing from the classical "local randomize-then-shuffle" paradigm [10, 37, 38, 74], AMP-SDP opts for the "server shuffle-then-randomize" approach to structurally enhance model privacy and security. This architectural shift also facilitates augmented shuffle DP variants.

Why relaxed randomization. Recent *personalized shuffle DP* work seeks better utility [17, 59] but typically adds side assumptions—e.g., negligible metadata leakage—to circumvent the indistinguishable-local-randomizer premise [19, 34]. AMP-SDP natively relaxes this constraint without introducing new assumptions.

Why reversible shuffle mechanism. The vanilla one-way shuffle DP pipeline targets single-round collection and lacks an efficient path for user-specific feedback. Yet, such anonymized, personalized feedback is essential for both interactive, adaptive workloads and preserving privacy amplification across rounds. AMP-SDP introduces a reversible shuffle—a slot-based backchannel that enables efficient two-way interaction under sustained anonymity.

4 DOPPIO FRAMEWORK

Building on the AMP-SDP model, we now present Doppio, a privacy-preserving crowdsourcing and analytics framework. As shown in Figure 3, this section focuses on its two core technical pillars—secure shuffle and DP computation—along with their augmented variants. Full protocols are detailed in Figure 8 and Figure 9.

4.1 Shuffle Correlation and Silent Shuffle

A typical oblivious shuffle applies a random permutation matrix $M \in \{0,1\}^{n \times n}$ to an input vector $X \in \mathbb{F}^n$, producing $X' = M \cdot X$. This amounts to evaluating n secure inner products in MPC, each over n pairs of elements. Typically, total communication costs scale with user number n, message size ℓ, and protocol running times [60, 63], leading to huge bandwidth cost in large-scale and long-term data ingestion services.

Pre-computability of random data mask shares and permutation shares. Our approach adapts the common 2-party additive sharing scheme ($x = [x]_1 + [x]_2$) in the preprocessing model. The first share, $[x]_1$, is treated as input-independent randomness and precomputed offline, while the complementary share, $[x]_2 = x - [x]_1$, is computed online once the user's true value x is available.

Concretely, upon user C_i's registration, the assisting server P_0 sets up a pseudorandom function PRF and negotiates a private pseudorandom seed s_i with C_i, after which they can sample the mask share $[x_i]_1$ by evaluating $\mathsf{PRF}(s_i)$ and initialize the dedicated distributed mailbox slots. Additionally, the random shuffle permutation matrix M, inherently being input-independent, can also be sampled offline by P_0, further reducing online overhead.

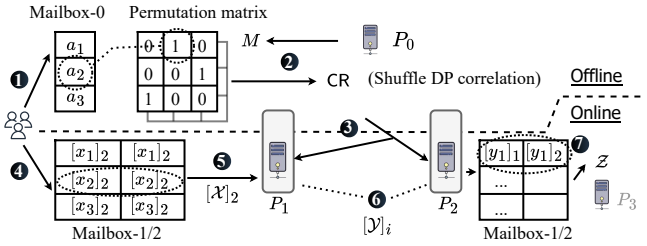


Figure 3: Overview of Doppio's workflow (*forward pipeline*), consisting of: ❶ user registration; ❷, ❸ shuffle DP correlation generation and distribution; ❹, ❺ user data submission; ❻ online silent processing; and ❼ processed result analytics.

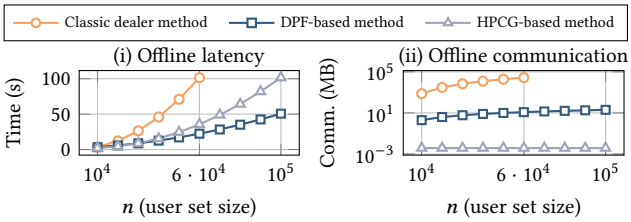


Figure 4: Comparison of offline costs for different shuffle correlation generation techniques.

Silent 2+1-party secure multiplication and shuffle. With pre-computed shares, we first illustrate how silent secure multiplication can be realized in the dealer model. Let $x = [x]_1 + [x]_2$ and $y = [y]_1 + [y]_2$ be two shared operands, their product expands as:

$$z = x \cdot y = \overbrace{[x]_1 \cdot y + c - c}^{\text{correlation pre-computed by } P_0} + [x]_2 \cdot [y]_1 + [x]_2 \cdot [y]_2$$
$$= \underbrace{[x]_2 \cdot [y]_1 + ([x]_1 \cdot y + c)}_{[z]_1 \text{ locally computed at } P_1} + \underbrace{[x]_2 \cdot [y]_2 - c}_{[z]_2 \text{ locally computed at } P_2},$$

where c is a pre-computed random mask term. The sum of result shares yields $x \cdot y$, requiring no other interaction among servers.

When extended to vectorized inputs, this yields a non-interactive 2+1-party shuffle protocol. Let $[X]_1 \in \mathbb{F}^n$ be the offline user mask shares, $[X]_2$ the online complement shares, and $M = [M]_1 + [M]_2$ the secret-shared random permutation matrix. The assisting server P_0 precomputes $\vec{\alpha} = M \cdot [X]_1$, shared as $[\vec{\alpha}]_1$ and $[\vec{\alpha}]_2$. To preserve shuffle privacy against P_0, computing servers can apply an extra random permutation π_{12} at runtime. Each P_i then computes:

$$[X']_i = \pi_{12}([M]_i \cdot [X]_2 + [\vec{\alpha}]_i), \quad i \in \{1, 2\}.$$

The sum of these outputs reconstructs the shuffled input vector X', while hiding the final permutation order from all parties. For simplicity, we omit π_{12} in subsequent protocol descriptions.

Offline shuffle correlation distribution. We formalize the above preprocessing output as the *silent shuffle correlation*:

Definition 4.1 (2-party silent shuffle correlation). *We let $\vec{a} \in \mathbb{F}^n$ be the offline masks and $M \in \{0,1\}^{n \times n}$ a random permutation matrix. Define $\vec{\alpha} = M \cdot \vec{a}$. The silent shuffle correlation is the pair $([M]_i, [\vec{\alpha}]_i)$, where $M = [M]_1 + [M]_2$, and $\vec{\alpha} = [\vec{\alpha}]_1 + [\vec{\alpha}]_2$.*

Distributing the above silent shuffle correlation under the naïve dealer model requires $O(n^2\ell)$ bits offline communication. To optimize this, Doppio explores two correlation compression techniques:

- *Distributed point functions (DPFs):* Each column of M (a one-hot vector) can be compactly encoded using a DPF key pair [13], reducing per-column communication to the $O(\log n)$ scale.
- *Hardware-assisted pseudorandom correlation generators (HPCGs):* With extra hardware trust, one can locally expand private, short seed into virtually unbounded shuffle correlations [26, 27], thereby eliminating the need to transmit full correlation shares.

Figure 4 compares the offline costs of three approaches (Classic dealer, DPF- and HPCG-based) under typical WAN settings. HPCG solution reports minimal communication costs, while DPF-based method provides a reasonable bandwidth-latency trade-off.

4.2 Generalized, Reversible, Scalable Shuffle

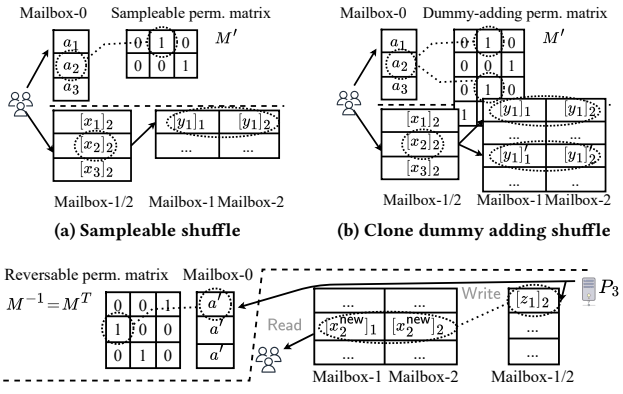
Sampleable and dummy-adding shuffle variants. Doppio extends the base shuffle $\mathcal{S}$ with a post-processing function $\mathcal{G}$, forming a composed transformation $\mathcal{G} \circ \mathcal{S}$. We consider two instantiations: *sampleable shuffle* and *dummy-adding shuffle*. The former applies random subsampling, providing amplification proportional to the sampling rate ϕ [4, 42, 49, 55]. The latter, aka *dummy blanket* [57], injects fake records to enlarge the anonymity set, yielding amplification that scales with $\sqrt{\phi n}$ [20, 64].

As depicted in Figure 5a and Figure 5b, optional post-processing $\mathcal{G}$ can be aptly implemented in Doppio through randomly duplicating or dropping rows in the shuffle correlation matrix M. For simplicity, we reuse ϕ to denote either the sampling rate ($\phi < 1$) or the dummy inflation factor ($\phi > 1$).

Reversible shuffle mechanism. As discussed, securing privacy amplification in adaptive, multi-round settings requires an anonymous route for curator responses—a mechanism vanilla shuffle DP lacks. A naïve method, re-shuffling the response vector Z, would double offline costs and impose an $O(n)$ lookup per user. Doppio instead employs a lightweight reversible shuffle: P_3 writes shared responses $[z_i]$ to the same slots that previously held $[y_i]$, after which $\mathcal{P}$ applies the inverse permutation to map responses back to each C_i's dedicated slots. This enables $O(1)$ anonymous retrieval per user, as summarized in Table 4. The key technical enabler for this pass is its offline efficiency. The permutation matrix M is orthogonal, allowing its inverse shares to be locally derived from its transpose rather than regenerated:

$$[M^{-1}]_1 + [M^{-1}]_2 = M^{-1} = M^T = [M^T]_1 + [M^T]_2.$$

Furthermore, unlike the forward pass, this backward pass involves only a single sender (i.e., the curator P_3) and thus requires



(a) Sampleable shuffle **(b) Clone dummy adding shuffle**

(c) Anonymous response backward routing via reversible shuffle

Figure 5: Generalized, reversible multi-party shuffle variants.

Table 4: Asymptotic communication complexity of AMP-SDP for the interactive, adaptive query workloads.

Ref.	C_i-$\mathcal{P}$	P_0-$\mathcal{P}$	$\mathcal{P}$=$\{P_1, P_2\}$	$\mathcal{P}$-P_3
Message forward	$O(1)$	$O(n \log n)$	0	$O(n)$
Response backward	$O(1)$	$O(n)$	0	$O(n)$

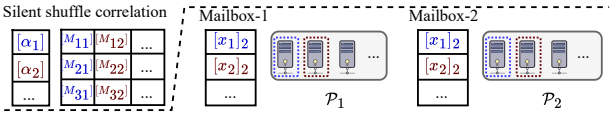
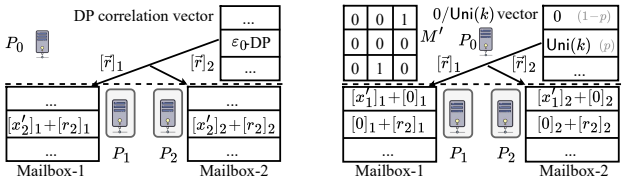


Figure 6: Sharded $2m+1$-party shuffle architecture. Each sub-server $P^i_* \in \mathcal{P}_*$ handles a disjoint workload shard.



(a) Shared Laplace mechanism **(b) Shared k-RR mechanism**

Figure 7: DP correlation is sampled offline (by P_0) and applied online (by P_1 and P_2). Non-additive mechanisms slightly differ from additive ones regarding the correlation form.

just one uniform mask a' for all n responses (see Figure 8, Step 4a). Also, the entire offline costs—both forward and backward—can then be amortized across all subsequent interaction rounds.[3]

Sharded ($2m+1$)-party shuffle architecture. Although 2+1-party setup is considered efficient and deployment-friendly [44, 50], its security relies on a rigid non-collusion assumption. To relax this and enhance scalability, Doppio explores a sharded $2m+1$-party architecture by expanding the original computing servers into logical computing groups, $\mathcal{P}_1$ and $\mathcal{P}_2$, each with m sub-servers (see Figure 6). Unlike conventional MPC-style scaling schemes that often increase the total share size per secret, our sharded scaling strategy partitions the workload itself, directly leveraging the computational parallelism and non-interactive shuffle techniques. This (1) further decentralizes trust by raising the collusion threshold from the original 2 to $m+1$; (2) reduces the per-server computational cost by splitting the workload and having each sub-server processes only a slice of the data (concrete performance gains depend on m); and (3) provides somewhat fault tolerance from redundancy, withstanding the failure of up to $m-1$ sub-servers per group.

The communication efficiency is largely preserved, with only additional overhead from intra-group aggregation for message reconstruction. This design enables shard-aware tuning of the system's balance between security, performance, and resilience.

4.3 DP Correlation and Silent Randomization

With the shuffled data held in additive shares, the following action is to securely and efficiently randomize them.

Correlated DP randomness. We decompose the whole DP process into (1) offline noise generation; and (2) online noise application. As Figure 7 shows, P_0 precomputes expensive yet input-independent DP noise offline, generating what we term *DP correlation*—a shared vector where each entry encodes a random DP noise sample. P_1 and P_2 then apply them online to perturb data. We define this below:

Definition 4.2 (2-party silent DP correlation). *We let $\vec{r} \in \mathbb{F}_q^n$ be a discrete DP noise vector, where each entry r_i is drawn from a computational ε_0, δ-DP distribution. Then, a silent DP correlation is defined as $[\vec{r}]_i$, where $\vec{r} = [\vec{r}]_1 + [\vec{r}]_2$.*

This design integrates naturally into the 2+1-party (dealer) setting. While the dealer can be substituted with alternatives such as distributed noise generation (DNG) [47] or fully MPC-based DP protocols [51], these dealer-less approaches generally suffer from low throughput (often below 10^3 samples/s) and restricted mechanism flexibility. They are therefore inefficient for typical shuffle DP systems, particularly when dealing with massive user populations, tight privacy budgets, or complex mechanisms.

[3]For uniform-response tasks (e.g., FedAvg), the reverse shuffle is unnecessary; the same shared output z_0 can simply be broadcast to all users.

Silent randomization for additive DP mechanisms. With DP correlation prepared offline, the online noise application for additive mechanisms (e.g., discrete Laplace) reduces to a simple, non-interactive secret-shared addition:

$$\mathcal{R}_{\mathsf{Lap}}(x'_i) = \underbrace{[x'_i]_1 + [r_i]_1}_{\text{locally computed at } P_1} + \underbrace{[x'_i]_2 + [r_i]_2}_{\text{locally computed at } P_2}.$$

Yet, summation inherently risks modular overflow and potentially incorrect results because shared inputs and noise are defined over the same field $\mathbb{F}_q$. For computational correctness, intermediary servers may further perform a lightweight arithmetic-to-arithmetic (A2A) sharing conversion [28] to lift the shares to a larger field $\mathbb{F}_{q'}$ before noise addition. This process incurs an extra cost of 2+1-party oblivious transfer (OT) per data item [52].

Silent randomization for non-additive DP mechanisms. Non-additive mechanisms like k-RR present a greater challenge, as they rely on a non-linear, probabilistic choice. To enable its silent execution, we first algebraically reformulate the mechanism as:

$$\mathcal{R}_{\mathsf{krr}}(x') = r + b \cdot (x' - r), \quad \text{where } r \sim \mathsf{Uni}(k),\ b \sim \mathsf{Bern}(p),$$

with $p = \frac{e^{\varepsilon_0}}{e^{\varepsilon_0}+k-1}$. Different from the additive mechanism, we encode the probabilistic choice into the precomputed correlation structures. Specifically, P_0 prepares a DP correlation vector $\vec{r}$ where entries are either a random category from $\mathsf{Uni}(k)$ or zero. To suppress a user's input x' (the case where $b=0$), we zeroize the corresponding entry in the shuffle correlation matrix M (see Figure 7b). Therefore, both cases reduce to a simple sum of two components, resulting in a unified, non-interactive execution:

$$\mathcal{R}_{\mathsf{krr}}(x') = \begin{cases} \underbrace{[x'_i]_1 + [0]_1}_{\text{locally computed at } P_1} + \underbrace{[x'_i]_2 + [0]_2}_{\text{locally computed at } P_2} = x'_i, & (\text{w/ } p), \\ \underbrace{[0]_1 + [r_i]_1}_{\text{locally computed at } P_1} + \underbrace{[0]_2 + [r_i]_2}_{\text{locally computed at } P_2} = r_i, & (\text{w/ } 1-p). \end{cases}$$

By jointly tuning shuffle DP correlations, our framework achieves a unified, non-interactive local-addition workflow capable of supporting both additive and non-additive DP mechanisms.

4.4 Relaxed Shuffle DP Randomization

The mandatory reliance on indistinguishable local randomizers [19, 34] structurally limits vanilla shuffle DP from optimizing utility via personalized noise strategies. Naïvely bypassing this restriction is insecure, as non-uniform randomizers may introduce leakage exploitable for de-anonymization attacks and shattering the key privacy amplification guarantees.

$\Pi_{\text{Doppio-off}}$: Offline Protocol

To initialize the entire system, generate and distribute data-independent correlations across assisting server P_0, user set C, and computing servers $\mathcal{P} = \{P_1, P_2\}$.

(1) **System initialization.**
   (a) The intermediary servers P_0, P_1, P_2 jointly instantiate the distributed virtual mailbox T, with each P_j initializes its portion of the slots T^j as empty.
   (b) For each user $C_i \in C$, P_0 assigns a unique identifier $i \in [n]$, defining the address of the user's static virtual mailbox addresses (T_i^0, T_i^1, T_i^2).

(2) **User registration and preparation.** For $C_i \in C$:
   (a) P_0 negotiates with C_i, generates a pseudorandom seed $s_i \leftarrow \{0,1\}^\kappa$, samples a random value $a_i \leftarrow \mathsf{PRF}(s_i)$ and stores it in T_i^0.
   (b) P_0 samples a unique permutation index $b_i \in [n]$ and generates the DPF key pair encoding permutation matrix M's non-zero entry in column i:
   $$(k_{i1}, k_{i2}) \leftarrow \mathsf{DPF.keyGen}(b_i).$$
   (c) P_0 samples DP noise r_i from an ε_i-DP distribution and encodes it into $([r_i]_1, [r_i]_2)$.

(3) **Correlation construction and distribution.** After processing all n users in C:
   (a) P_0 computes and constructs the permuted mask vector $[\vec{\alpha}]_j = [M \cdot \vec{a}]_j$; assembles the DPF key vector $\vec{k}_j$ and DP noise vector $[\vec{r}]_j$; and distributes these vector shares $[\vec{\alpha}]_j$, $\vec{k}_j$, and $[\vec{r}]_j$ to each $P_j \in \mathcal{P}$.
   (b) Each P_j locally expands $\vec{k}_j$ to derive matrix share $[M]_j$, constructing shuffle DP correlation as:
   $$\mathsf{CR}_j := \big([M]_j, [\vec{\alpha}]_j, [\vec{r}]_j\big).$$
   And then store CR_j in slots T^j.

(4) **(Optional) Reverse correlation construction and distribution.** If backward delivery is enabled:
   (a) P_0 negotiates with P_3, generates seed $s' \leftarrow \{0,1\}^\kappa$, samples $a' \leftarrow \mathsf{PRF}(s')$, and stores it to all T^0 slots;
   (b) P_0 constructs mask vector $\vec{\alpha}'$ and distributes its share $[\vec{\alpha}]'_j$ to $P_j \in \mathcal{P}$.
   (c) Each P_j locally computes $[M^{-1}]_j = [M^T]_j$, constructing the reversible correlation as:
   $$\mathsf{CR}' := ([M^{-1}], [\vec{\alpha}']_j).$$
   And then store CR_j in slots T^j as well.

Figure 8: Doppio's offline protocol.

Instead of the "*randomize-then-shuffle*" (RtS) paradigm, this work adopts a "*shuffle-then-randomize*" (StR) pipeline. While prior work proved that the shuffle-randomize order can be originally irrelevant to final privacy guarantees [34], we show this equivalence breaks down when DP mechanisms are non-uniform (or personalized).

Proposition 4.1 (Shuffle-randomize order (revisited)). *Let $X = \{x_1, \ldots, x_n\}$ be ordered inputs, and $\mathcal{S}$ be a uniform, random permutation. Suppose each x_i is randomized via a non-uniform mechanism $\mathcal{R}_i \in \mathcal{R}_+$, yielding a composition $\mathcal{R}_+(X) = \{\mathcal{R}_1(x_1), \ldots, \mathcal{R}_n(x_n)\}$. Let $\mathcal{L}(\mathcal{R}_+)$ denote leakage information about the family of randomizers, and let D be any PPT adversary with access to $\mathcal{L}(\mathcal{R}_+)$. Define η as the ratio between the effective local privacy parameter ε_+ (derived from ε_i) and the amplified privacy parameter ε'. Then we have:*

$$\eta_{\text{RtS}}^{D_{\mathcal{L}(\mathcal{R}_+)}} := \eta\big(\mathcal{S} \circ \mathcal{R}_+(X)\big) \le \eta\big(\mathcal{R}_+ \circ \mathcal{S}(X)\big) := \eta_{\text{StR}}^{D_{\mathcal{L}(\mathcal{R}_+)}}.$$

Proof sketch: This inequality holds because, in the RtS paradigm, the adversary D can exploit (public) knowledge of which non-uniform randomizer $\mathcal{R}_i$ is applied to each shuffled output y_j. This leakage enables D to partition the n-user anonymity set into smaller disjoint subsets $\{X_1, \ldots, X_t\}$, confining privacy amplification to the subgroup sizes $|X_j|$. As a result, the overall guarantee is comparable to that of weaker approximate shuffle DP schemes [5, 58, 80], where amplification is known to be less effective. In contrast, StR shuffles before randomization, breaking the link between each output y_j and

$\Pi_{\text{Doppio-on}}$: Online Protocol

To perform the AMP-SDP pipeline across the user set C, intermediary servers P_0, P_1, P_2, and curator server P_3

(1) **User submission.** Each user C_i computes its online share $[x_i]_2 = x_i - a_i$ and submits it to both computing servers. Each server $P_j \in \mathcal{P}$ then stores the received share in the user's designated mailbox slot T_i^j.

(2) **Silent processing.** Once all n shares are received, each computing server P_j executes locally:
   (a) Construct the shared user input vector $[X]_2$ by reading all shares from its mailbox T^j.
   (b) (Silent shuffle) Compute the shuffled data shares:
   $$[X']_j := [\vec{\alpha}]_j + [M]_j \cdot [X]_2.$$
   (c) (Silent randomization) Compute the randomized, shuffled data shares::
   $$[Y]_j := [X']_j + [\vec{r}]_j.$$
   Then write $[Y]_j$ to its local mailbox slots T^j.

(3) **Output reconstruction.** The curator P_3 retrieves $[Y]_j$ from T^j, reconstructs $Y = \sum_j [\mathcal{Y}]_j$, and performs the final analytical workload:
$$Z = f(Y).$$

(4) **(Optional) Anonymous response delivery.** If backward delivery is enabled:
   (a) (Curator as sender) P_3 constructs $[Z]_2$, with each $[z_i]_2 = z_i - a'$, writing back to slot T_i^1 and T_i^2.
   (b) (Silent reversible shuffle) Each P_j update T^j by locally computing the permuted response shares:
   $$[Z']_j := [\vec{\alpha}']_j + [M^T]_j \cdot [Z]_2.$$
   (c) (Users as receivers) Each $C_{i'}$ then retrieves $[z_{i'}]_j$ by reading its own slots $T_{i'}^j$ and reconstruct $z_{i'}$ to bootstrap the next interactive round.

Figure 9: Doppio's online protocol.

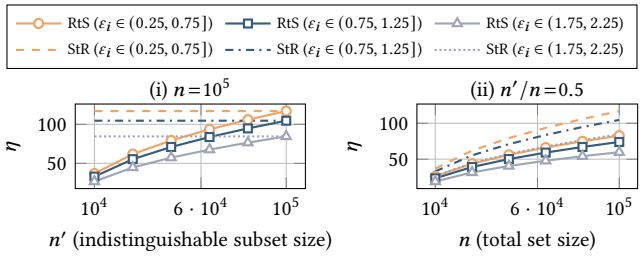


Figure 10: Numerical privacy amplification bounds in shuffle DP for different shuffle-randomize orders.

its origin C_i. Although the adversary still observes non-uniform outputs, the leakage $\mathcal{L}(R_+)$ remains uninformative, preserving privacy amplification over the entire n-user set. □

The numerical simulations in Figure 10 also confirm this analysis. Specifically, we sample different ε_i from certain ranges for different records, intentionally constructing statistically distinguishable output groups. The results, obtained by varying the indistinguishable subset size n' and the total user set size n, show the amplification rate generally increases with both of these factors.

5 SECURITY ANALYSIS

AMP-SDP architecturally addresses two critical security concerns of the vanilla shuffle model: (1) the centralized shuffler risk; and (2) the vulnerability to user-side poisoning attacks.

Shuffler-side trust and security. Compared with conventional designs having a centralized trusted shuffler, Doppio distributes trust among multiple non-colluding intermediary servers, ensuring that no single server can learn users' raw data. As shown in Figure 8 and Figure 9, the assisting server P_0 is active only during the offline, data-independent phase, while the computing servers P_1 and P_2 can only ever observe secret shares of user data and the corresponding cryptographic correlated randomness.

Proposition 5.1 (Semi-honest security). *Protocol* $\Pi_{\text{Doppio-on}}$ *is secure against semi-honest adversaries corrupting either* P_1 *or* P_2*, assuming correctly generated correlated randomness.*

Proof sketch: Assume, without loss of generality, P_1 is corrupted. A simulator Sim can generate a computationally indistinguishable view for P_1 using only its local input shares $[X]_2$ and offline correlation shares (CR_1). This reduction is possible because the online protocol is non-interactive; consequently, all online computations and message flows are entirely predetermined by the pre-sampled correlation and its local information. The case where P_2 is corrupted follows symmetrically. □

Collusion resilience and trust separation. Following MPC assumptions, all servers $\{P_0, P_1, P_2, P_3\}$ are mutually non-colluding, reflecting real-world deployments operated by independent stakeholders [1, 22, 28, 36, 44, 48]. To relax this and increase collusion threshold, Doppio extends a $2m+1$-party variant (see §4.2): even if the curator P_3 colludes with a subset of users or sub-servers, the joint permutation remains hidden and private.

User-side poisoning and security. Another critical threat is data poisoning. This attack, widely studied under local DP [14, 18, 56, 72, 78], also persists in the shuffle model, while presenting a unique security landscape defined by a fundamental dichotomy inherent to the model's structural anonymity:

- *Inherent robustness:* On one hand, anonymity amplifies privacy, allowing larger local parameters ε_0 under a given, expected global budget ε'. This lowers the true noise magnitude and suppresses the relative influence of individual malicious inputs—an effect also reflected in Table 3.
- *Poisoning amplification:* On the other hand, the same anonymity simultaneously nullifies accountability. Without attribution, malicious users can arbitrarily inject or coordinate biased submissions undetected, enabling stealthy, large-scale poisoning that abuses the very anonymity meant to protect honest users.

To analyze this, we first define the base corruption ratio as $\gamma = \frac{n_c}{n}$ and introduce an amplification factor μ. Instead of directly modeling a complex, incremental attack, we reduce this to an analogous, analytically simpler problem: an adversary manipulating n_c users submits μ malicious records per user. This simplification is made for analytical convenience, but it effectively models the core issue—the inflation of corrupted inputs within the total dataset—and yields an effective corruption ratio $\gamma' = \frac{\mu \cdot n_c}{n + (\mu - 1) \cdot n_c}$.

As demonstrated in Figure 11, we apply targeted maximal gain attacks (MGA) on two representative workloads with $n = 10^4$ users. For frequency estimation, we employ the k-RR mechanism ($k = 20$) over $\{1, \ldots, 20\}$, simulating a targeted attack where corrupted users inject a fixed item of 5; for variance estimation, we use the Laplace mechanism over $[0, 1]$, simulating an MGA that maximizes the total variance. Across both settings (also see Figures 12 and 13), shuffle DP exhibits higher robustness in the ideal case ($\mu = 1$) than local counterpart, but its advantage diminishes rapidly as μ increases.

Structural poisoning mitigation. AMP-SDP mitigates poisoning not by detecting malicious clients, but by structurally constraining what any user can manipulate.

- *OPA elimination.* By delegating randomization to intermediary servers, AMP-SDP structurally removes OPAs.
- *IPA mitigation.* Inspired by prior work, AMP-SDP further incorporates several complementary strategies to constrain IPAs:
  - *Domain enforcement [1, 28]:* encoding numerical inputs in a finite field $\mathbb{F}_q$ to natively bound its range, preventing adversaries from injecting excessively large secret-shared values to gain disproportionate influence.
  - *Contribution dilution [57]:* utilizing dummy-adding shuffle to create a group of well-formed, indistinguishable dummy records that dilute the influence of potential malicious inputs.

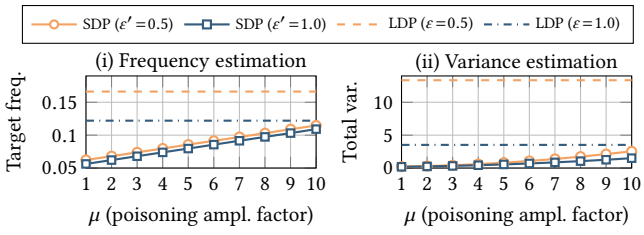


Figure 11: Poisoning amplification in shuffle DP over both frequency estimation and variance estimation tasks.

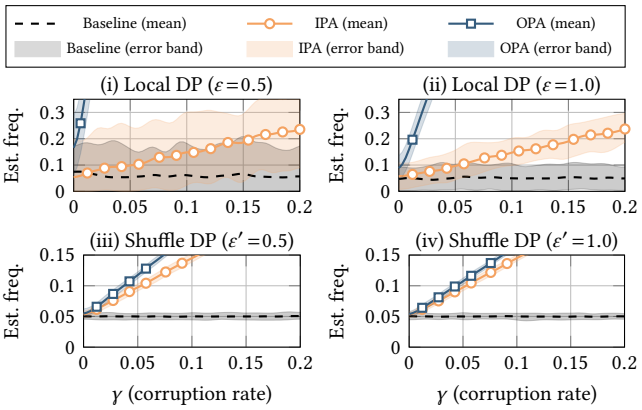


Figure 12: Comparison of poisoning robustness in local DP and shuffle DP for the *frequency estimation* task.

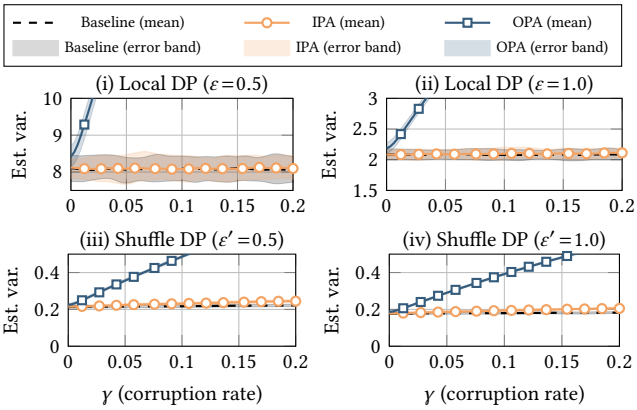


Figure 13: Comparison of poisoning robustness in local DP and shuffle DP for the *variance estimation* task.

These structural defenses are mostly mechanism-agnostic, applying to a broad class of query workloads in database systems.

Proposition 5.2 (Attack surface reduction). *The AMP-SDP model reduces the attack surface of malicious users compared to the vanilla shuffle DP model.*

Proof sketch: As discussed above, AMP-SDP rules out OPA threats and restricts IPAs to bounded domains. □

These defenses bound the worst-case deviation, yielding stronger robustness.[4] This structural advantage is corroborated by the empirical results in Figures 12 and 13. Across poisoning simulations with $n = 10^4$ users under different ε' and γ parameters, Doppio consistently confines poisoning effects to the IPA regime, exhibiting a clear robustness separation to the vanilla model.

6 PROTOTYPE AND EVALUATION

To further showcase our AMP-SDP model and Doppio framework, we built a prototype by adapting Microsoft's Precio system [28] for the online MPC backend and adopting Google's DPF library [12] for offline correlation generation. All computations are performed over the finite fields and optimized using AVX2 vectorization and multi-threading. The experiments were conducted on Ubuntu 20.04 servers (Intel i7-9700K CPU, 64GB RAM), simulating a typical WAN setting (1 Gbps bandwidth, 60–120 ms RTT).

Dataset and baseline. We utilized two classes of datasets to showcase our designs: (1) a synthetic dataset generated under a uniform distribution and used for microbenchmarks on performance, privacy, and robustness, featuring varying record counts $n \in [10^4, 10^5]$ and bit-widths $\ell \in \{64, 1024\}$; and (2) two real-world datasets used to measure practical utility and query performance, both derived from larger municipal data collections, including:

- *Fire [24]:* $n = 112{,}436$ records filtered by the call type. It contains 33 attributes, with each record capped at approximately 400 B.
- *Retirement [65]:* $n = 530{,}175$ records filtered by payments. It comprises 22 attributes, with each record capped at 250 B.

We primarily compare Doppio with (1) network shuffling [58]—the state-of-the-art decentralized shuffle DP scheme, and (2) vanilla shuffle DP paradigm with a trusted shuffler [74].

Microbenchmark: online performance. Doppio's core online protocol proceeds as a masked matrix-vector multiplication over the finite field for both its message forward $[Y]_j = [\vec{\alpha}]_j + [M]_j \cdot [X]_2 + [\vec{r}]_j$ and response backward passes $[Z']_j = [\vec{\alpha'}]_j + [M^T]_j \cdot [Z]_2$. As shown in Table 5, Doppio achieves zero online inter-server communication across all variants due to the silent online protocol designs. This starkly contrasts with network shuffling [58] that incurs substantial communication overhead, which is especially borne by the clients with more limited bandwidth and computational resources. For instance, at $n = 10^5$ and $\ell = 1024$, Doppio's online server communication is optimal 0, while network shuffling requires iterative peer-to-peer exchanges totaling gigabytes of data.

With communication costs eliminated, online latency is dominated by local computation. The entire secure shuffle DP pipeline completes in < 4 seconds for 10^4 users, showing high efficiency. However, when serving a larger user base ($n = 10^5$), the latency grows super-linearly due to memory constraints and quadratic computation complexity, highlighting the importance of our architectural extensions. By adopting the sharded scaling variant ($m = 2$), we can effectively halve the per-server load, and therefore improve the system throughput. Furthermore, the reversible shuffle mechanism efficiently extends the system to support interactive, adaptive analytics, adding only a symmetric computational cost per round and minimal communication per user.

Microbenchmark: privacy amplification capability. Figure 14 highlights the amplification advantage of Doppio over non-uniform network shuffling [58]. The observed privacy gain confirms the theoretical distinction between uniform, oblivious shuffle mechanisms [6, 34, 37] and their non-uniform or approximate counterparts [43, 53, 80]. Furthermore, AMP-SDP conditionally provides better privacy guarantees than pure shuffle DP [74]. These gains, determined by the factor ϕ, stem from two privacy-boosting extensions: sampleable shuffle and dummy-adding shuffle. These findings are also consistent with recent analyses [55, 57, 64] and demonstrate Doppio's architectural advantages in privacy guarantees.

[4] We acknowledge IPAs are a common, fundamental threat to any system accepting user input; a general-purpose IPA defense is outside the scope of this work.

Table 5: Comparison of online performance between Doppio and [58] under varying input numbers (n) and bit-widths (ℓ).

Ref.[†,‡,★,♮,♯]	$n = 10^4$, $\ell = 64$		$n = 10^4$, $\ell = 1024$		$n = 10^5$, $\ell = 64$		$n = 10^5$, $\ell = 1024$	
Shuffle Variant	**Time (s)**	**Comm (MB)**	**Time (s)**	**Comm (MB)**	**Time (s)**	**Comm (MB)**	**Time (s)**	**Comm (MB)**
Network shuffling [58]	≈120.55	≈10.15 / 0.08	≈130.34	≈162.50 / 1.28	≈159.36	≈126.60 / 0.80	≈282.20	≈2026.00 / 12.80
Doppio	0.42	0.00 / 0.16	3.68	0.00 / 2.56	13.17	0.00 / 1.60	168.24	0.00 / 25.60
Doppio (Sampleable)	0.38	0.00 / 0.14	3.31	0.00 / 2.40	11.88	0.00 / 1.44	151.42	0.00 / 23.04
Doppio (Dummy adding)	0.46	0.00 / 0.18	4.05	0.00 / 2.82	14.54	0.00 / 1.77	185.06	0.00 / 28.16
Doppio (Sharded scaling)	0.21	0.00 / 0.32	1.84	0.00 / 5.12	6.23	0.00 / 3.20	69.29	0.00 / 51.20
Doppio (Reversible)	+0.42	+0.00 / 1.60×10^{-5}	+3.68	+0.00 / 2.56×10^{-4}	+13.17	+0.00 / 1.60×10^{-5}	+168.24	+0.00 / 2.56×10^{-4}

† Network shuffling is estimated to perform $c^{-1}\log n$ rounds of ℓ-bit pairwise exchanges for approximately mixing all records [58]. As reported in the work, here we set $c = \frac{1}{100}$ and assume RTT = 90. Its shuffling-related communication occurs among users (instead of servers).
‡ We set sampleable shuffle uses sampling rate $\phi = 0.9$; dummy-adding shuffle uses inflation ratio $\phi = 1.1$, the total cost grows almost linearly.
★ Online communication is split into inter-server computation and reconstruction phases, denoted as 'a / b'; the former in Doppio is zero across all silent shuffle DP variants.
♮ The sharded scaling variant uses $m = 2$ sub-servers per computing group $\mathcal{P}_*$. This maintains zero online inter-server cost yet increases message reconstruction communication.
♯ The '+' prefix indicates the additional cost incurred during the anonymous response backward phase. Importantly, reconstruction cost is measured per user in this row.

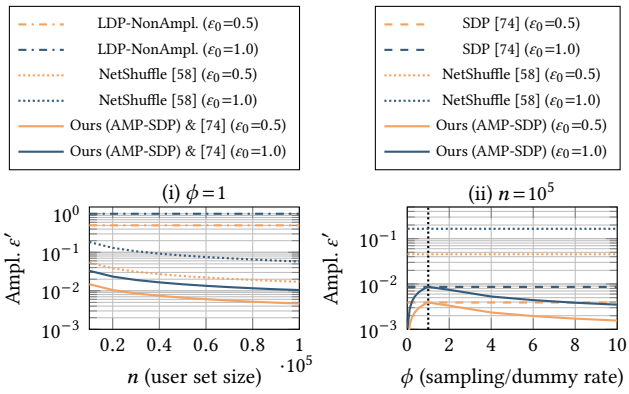
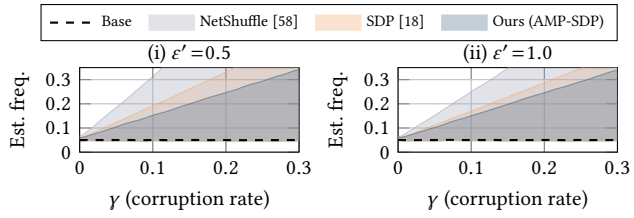
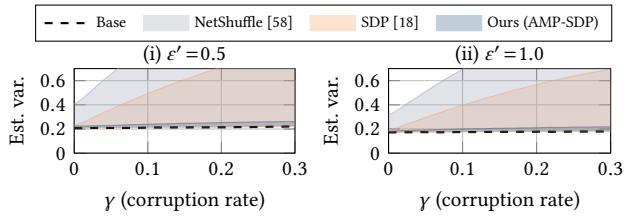


Figure 14: Comparison of privacy amplification capability among vanilla shuffle DP [74], network shuffling [58], and our AMP-SDP models (measured by privacy budget).



(a) Attack upper bounds for the frequency estimation (k-RR mechanism).



(b) Attack upper bounds for the variance estimation (Laplace mechanism).

Figure 15: Comparison of data poisoning robustness among vanilla shuffle DP [18], network shuffling [58], and our AMP-SDP models (measured by estimation error band).

Microbenchmark: security and robustness implication. We evaluate Doppio's system-level defense against malicious user poisoning—a critical threat in real-world data management. By delegating randomization to the server side, AMP-SDP eliminates the most damaging OPAs by construction. This contrasts with baselines such as pure shuffle DP [74] and network shuffling [58], which depend on user-side coordination and thereby expose a broader attack surface to adversarial manipulation.

To quantify this effect, we conduct a targeted MGA attack [14] on two representative workloads using uniformly sampled synthetic dataset: *frequency estimation* using the k-RR mechanism ($k = 20$) and *variance estimation* using the Laplace mechanism over $[0, 1]$. We fix the number of users at $n = 10^4$ and vary the actual corruption ratio γ from 0 to 0.3, representing increasing adversarial participation. Figure 15 reports the empirical error bands, confirming Doppio's structural robustness advantage.

Microbenchmark: utility on realistic datasets and database workloads. To complement the preceding synthetic-data evaluations, we further examine Doppio's practical utility on two real-world datasets—*Retirement* [65] and *Fire* [24]—and benchmark four representative types of SQL query workloads:

(1) *Scalar aggregation (average)*:
```
SELECT AVG(Retirement_amount) FROM Retirement;
```
(2) *Group-by aggregation (histogram)*:
```
SELECT Job_family, COUNT(*) FROM Retirement
GROUP BY Job_family;
```
(3) *Range query (proportion)*:
```
SELECT CAST(COUNT(*) AS REAL) /
(SELECT COUNT(*) FROM Fire) FROM Fire
WHERE Dispatch_DtTm - Received_DtTm < 2;
```
(4) *Multi-round data exploration (drill down)*:
(*Round 1*)
```
SELECT Dept, COUNT(*) AS dept_cnt
FROM Retirement GROUP BY Dept
ORDER BY dept_cnt DESC LIMIT 5;
```
(*Round 2*)
```
SELECT AVG(Salary) FROM Retirement
WHERE dept In Dept;
```

Our experimental evaluation on two real-world datasets demonstrates that Doppio consistently achieves higher data utility than the state-of-the-art distributed network shuffling mechanism [58],

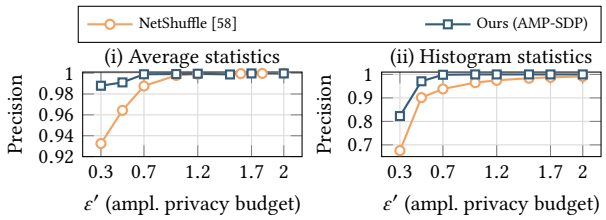


Figure 16: Comparison of estimation precision between network shuffling [58] and our AMP-SDP model for average and histogram statistics on the *Retirement* [65] dataset.

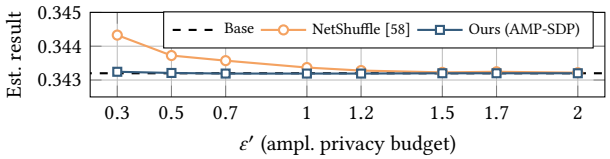


Figure 17: Comparison of estimation result between network shuffling [58] and our AMP-SDP model for range query on the Fire [24] dataset.

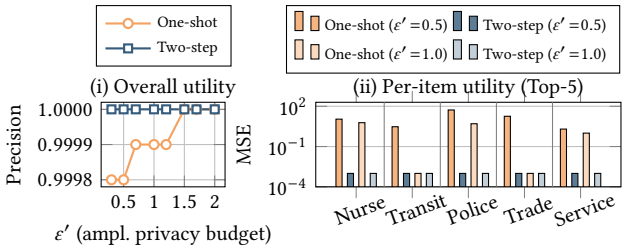


Figure 18: Comparison of analytical utility between one-shot estimation and two-step drill-down methods for our AMP-SDP model on the Retirement [65] dataset.

with the advantage widening as the analytical task becomes more complex. As shown in Figure 16, for the simple average estimation on the Retirement dataset, Doppio already attains a precision of 0.998 under a strict privacy budget of $\varepsilon' = 0.7$, surpassing network shuffling's 0.938. This gap persists for more demanding histogram-based analyses, where Doppio is capable of better preserving fine-grained statistical patterns. Similarly, for time-series range queries on the Fire dataset (as reflected in Figure 17), Doppio achieves substantially lower relative error—over 50% reduction at $\varepsilon' = 0.5$—in estimating valid record counts, demonstrating superior accuracy even under tight privacy constraints.

We further evaluate Doppio under an interactive, adaptive workload using a two-round "drill-down" pipeline on the *Retirement* [65] dataset, compared to a naïve one-shot execution. In Round 1, users report their `Job_family` via k-RR to identify the Top-5 categories; in Round 2, Doppio then estimates the average `Retirement` amount within these categories using the Laplace mechanism. To achieve better utility, all users participate—those in the Top-5 contribute perturbed true values, while others send perturbed dummy values (e.g., ≈0). In particular, to enforce this conditional participation, the curator P_3 informs the dealer P_0 of dummy-user indices so that P_0 can zeroize the corresponding columns in the shuffle matrix M, preventing unqualified users from injecting non-dummy data. As shown in Figure 18, this two-step pipeline can improve analytical accuracy and reduces MSE across all categories, demonstrating the potential of structured multi-round shuffle DP analytics.

7 DISCUSSION

We next discuss several deployment-oriented aspects of Doppio.

Chunked processing. Practical deployments must handle some high-dimensional user records with large bit-widths (e.g., $\ell \approx 1$ KB). To maintain scalability and avoid the inefficiency of encoding such records within a single, large finite field, Doppio partitions each user's ℓ-bit record into k fixed-size chunks, denoted as $x_i := x_{i,1} \| \ldots \| x_{i,k}$. The silent shuffle then generalizes to a matrix–matrix multiplication that reuses and amortizes a single shuffle correlation across all k chunks. For example, on the realistic Fire [24] dataset, each record can be split into 50×8 B or 4×128 B chunks, incurring modest and predictable overhead with a tunable trade-off.

On-the-fly preprocessing. Doppio features a lean online phase, with communication- and computation-intensive operations shifted to the offline. In secure data crowdsourcing, user submissions naturally arrive asynchronously, whereas shuffling requires a complete batch. Doppio leverages this inherent interval to pipeline on-the-fly preprocessing during data collection, effectively hiding the offline workload and avoiding idle resources.

Limitations. Doppio currently operates in a static mode with fixed batch size n and pre-registered users, assumes semi-honest servers, and—due to the nature of shuffle DP—supports only commutative statistics whose outputs are invariant to record order.

8 CONCLUSION

This work introduced the AMP-SDP model and the Doppio framework. By co-designing MPC and shuffle DP techniques, our model decentralizes trust, optimizes efficiency, enhances security, and expands the known technical scope, establishing Doppio as a new, promising paradigm for privacy-aware data management. Looking ahead, we aim to address the current model limitations and further generalize AMP-SDP across rich, diverse data-processing pipelines. We also plan to integrate Doppio with large-scale database infrastructures, enabling practical, end-to-end privacy-preserving data analytics in real-world deployments.

ACKNOWLEDGMENTS

The authors sincerely thank all reviewers for their insightful feedback. This research was supported in part by Hong Kong Research Grants Council (RGC) under Grants 11217620, 11218521, 11218322, 11219025, R6021-20F, R1012-21, RFS2122-1S04, C2004-21G, C1029-22G, C6015-23G, and N_CityU139/21; Hong Kong Innovation and Technology Commission (ITC) under Project MHP/135/23; Japan Society for the Promotion of Science (JSPS) under Grant KAKENHI JP23K24851; and Japan Science and Technology Agency (JST) under Grants PRESTO JPMJPR23P5, CREST JPMJCR21M2, and NEXUS JPMJNX25C4. This work was also substantially supported by the InnoHK initiative, the Government of the HKSAR, and the Laboratory for AI-Powered Financial Technologies (AIFT).

REFERENCES

[1] Surya Addanki, Kevin Garbe, Eli Jaffe, Rafail Ostrovsky, and Antigoni Polychroniadou. 2022. Prio+: Privacy preserving aggregate statistics via boolean shares. In *Proc. of SCN*. 516–539.

[2] Apple. 2020. Apple and Google partner on COVID-19 contact tracing technology. https://www.apple.com/hk/en/newsroom/2020/04/apple-and-google-partner-on-covid-19-contact-tracing-technology/ [Online; accessed: Apr./2025].

[3] Nuttapong Attrapadung, Goichiro Hanaoaka, Takahiro Matsuda, Hiraku Morita, Kazuma Ohara, Jacob CN Schuldt, Tadanori Teruya, and Kazunari Tozawa. 2021. Oblivious linear group actions and applications. In *Proc. of ACM CCS*. 630–650.

[4] Borja Balle, Gilles Barthe, and Marco Gaboardi. 2018. Privacy amplification by subsampling: Tight analyses via couplings and divergences. In *Proc. of NeurIPS*.

[5] Borja Balle, James Bell, and Adrià Gascón. 2023. Amplification by Shuffling without Shuffling. In *Proc. of ACM CCS*, Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda (Eds.). 2292–2305.

[6] Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. 2019. The privacy blanket of the shuffle model. In *Proc. of CRYPTO*. 638–667.

[7] Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. 2020. Private Summation in the Multi-Message Shuffle Model. In *Proc. of ACM CCS*. 657–676.

[8] Raef Bassily and Adam Smith. 2015. Local, private, efficient protocols for succinct histograms. In *Proc. of ACM STOC*. 127–135.

[9] Amos Beimel, Kobbi Nissim, and Eran Omri. 2008. Distributed private data analysis: Simultaneously solving how and what. In *Proc. of CRYPTO*. 451–468.

[10] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. 2017. Prochlo: Strong privacy for analytics in the crowd. In *Proc. of ACM SOSP*. 441–459.

[11] Jonas Böhler and Florian Kerschbaum. 2020. Secure multi-party computation of differentially private median. In *Proc. of USENIX Security*. 2147–2164.

[12] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. 2021. Lightweight techniques for private heavy hitters. In *Proc. of IEEE S&P*. 762–776.

[13] Elette Boyle, Niv Gilboa, and Yuval Ishai. 2016. Function secret sharing: Improvements and extensions. In *Proc. of ACM CCS*. 1292–1303.

[14] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. 2021. Data poisoning attacks to local differential privacy protocols. In *Proc. of USENIX Security*. 947–964.

[15] Jeffrey Champion, Abhi Shelat, and Jonathan Ullman. 2019. Securely sampling biased coins with applications to differential privacy. In *Proce. of ACM CCS*. 603–614.

[16] Melissa Chase, Esha Ghosh, and Oxana Poburinnaya. 2020. Secret-Shared Shuffle. In *Proc. of ASIACRYPT*, Shiho Moriai and Huaxiong Wang (Eds.). 342–372.

[17] E Chen, Yang Cao, and Yifei Ge. 2024. A Generalized Shuffle Framework for Privacy Amplification: Strengthening Privacy Guarantees and Enhancing Utility. In *Proc. of AAAI*. 11267–11275.

[18] Albert Cheu, Adam Smith, and Jonathan Ullman. 2021. Manipulation attacks in local differential privacy. In *Proc. of IEEE S&P*. 883–900.

[19] Albert Cheu, Adam Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. 2019. Distributed differential privacy via shuffling. In *Proc. of EUROCRYPT*. 375–403.

[20] Albert Cheu and Maxim Zhilyaev. 2022. Differentially private histograms in the shuffle model from fake users. In *Proc. of IEEE S&P*. 440–457.

[21] Christopher A. Choquette-Choo, Natalie Dullerud, Adam Dziedzic, Yunxiang Zhang, Somesh Jha, Nicolas Papernot, and Xiao Wang. 2021. CaPC Learning: Confidential and Private Collaborative Learning. In *Proc. of ICLR*.

[22] Henry Corrigan-Gibbs and Dan Boneh. 2017. Prio: Private, robust, and scalable computation of aggregate statistics. In *Proc. of USENIX NSDI*. 259–282.

[23] TuGraph DB. 2023. HTAP-TuGraph documentation. https://tugraph-db.readthedocs.io/en/v4.3.2/2.introduction/5.characteristics/3.htap.html [Online; accessed: Jul./2025].

[24] San Francisco City Fire Department. 2024. San Francisco Fire Department Public Dataset. https://www.kaggle.com/datasets/imankity/san-francisco-fire-department-public-dataset [Online; accessed: Jul./2025].

[25] Wentao Dong, Peipei Jiang, Huayi Duan, Cong Wang, Lingchen Zhao, and Qian Wang. 2025. Ring of Gyges: Accountable Anonymous Broadcast via Secret-Shared Shuffle. In *Proc. of NDSS*.

[26] Wentao Dong and Cong Wang. 2023. Poster: Towards Lightweight TEE-Assisted MPC. In *Proc. of ACM CCS*. 3609–3611.

[27] Wentao Dong, Lei Xu, Leqian Zheng, Huayi Duan, Cong Wang, and Qian Wang. 2025. Do Not Skip Over the Offline: Verifiable Silent Preprocessing From Small Security Hardware. *IEEE Trans. on Information Forensics and Security* 20 (2025), 4860–4873.

[28] F. Betül Durak, Chenkai Weng, Erik Anderson, Kim Laine, and Melissa Chase. 2024. Precio: Private Aggregate Measurement via Oblivious Shuffling. In *Proc. of ACM CCS*. 1819–1833.

[29] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our data, ourselves: Privacy via distributed noise generation. In *Proc. of EUROCRYPT*. 486–503.

[30] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

[31] Fabienne Eigner, Aniket Kate, Matteo Maffei, Francesca Pampaloni, and Ivan Pryvalov. 2014. Differentially private data aggregation with optimal utility. In *Proc. of ACSAC*. 316–325.

[32] Ahmed Roushdy Elkordy, Yahya H Ezzeldin, Shanshan Han, Shantanu Sharma, Chaoyang He, Sharad Mehrotra, Salman Avestimehr, et al. 2023. Federated analytics: A survey. *APSIPA Transactions on Signal and Information Processing* 12, 1 (2023).

[33] Reo Eriguchi, Atsunori Ichikawa, Noboru Kunihiro, and Koji Nuida. 2023. Efficient Noise Generation Protocols for Differentially Private Multiparty Computation. *IEEE Trans. on Dependable and Secure Computing* 20, 6 (2023), 4486–4501.

[34] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. 2019. Amplification by shuffling: From local to central differential privacy via anonymity. In *Proc. of ACM-SIAM SODA*. 2468–2479.

[35] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proc. of ACM CCS*. 1054–1067.

[36] Saba Eskandarian and Dan Boneh. 2022. Clarion: Anonymous communication from multiparty shuffling protocols. In *Proc. of NDSS*.

[37] Vitaly Feldman, Audra McMillan, and Kunal Talwar. 2022. Hiding among the clones: A simple and nearly optimal analysis of privacy amplification by shuffling. In *Proc. of IEEE FOCS*. 954–964.

[38] Vitaly Feldman, Ilya Mironov, Kunal Talwar, and Abhradeep Thakurta. 2018. Privacy amplification by iteration. In *Proc. of IEEE FOCS*. 521–532.

[39] Yucheng Fu and Tianhao Wang. 2024. Benchmarking Secure Sampling Protocols for Differential Privacy. In *Proc. of ACM CCS*. 318–332.

[40] Badih Ghazi, Ravi Kumar, Pasin Manurangsi, Rasmus Pagh, and Amer Sinha. 2021. Differentially private aggregation in the shuffle model: Almost central accuracy in almost a single message. In *Proc. of ICML*. 3692–3701.

[41] Antonious Girgis, Deepesh Data, Suhas Diggavi, Peter Kairouz, and Ananda Theertha Suresh. 2021. Shuffled model of differential privacy in federated learning. In *Proc. of AISTATS*. 2521–2529.

[42] Antonious Girgis, Deepesh Data, Suhas Diggavi, Ananda Theertha Suresh, and Peter Kairouz. 2021. On the renyi differential privacy of the shuffle model. In *Proc. of ACM CCS*. 2321–2341.

[43] Dov Gordon, Jonathan Katz, Mingyu Liang, and Jiayu Xu. 2022. Spreading the privacy blanket: Differentially oblivious shuffling for differential privacy. In *Proc. of ACNS*. 501–520.

[44] Matan Hamilis, Elette Boyle, Niv Gilboa, Yuval Ishai, and Ariel Nof. 2025. Preprocessing for Life: Dishonest-Majority MPC with a Trusted or Untrusted Dealer. In *Proc. of IEEE S&P*. 2433–2452.

[45] Mikko Heikkilä, Eemil Lagerspetz, Samuel Kaski, Kana Shimizu, Sasu Tarkoma, and Antti Honkela. 2017. Differentially private bayesian learning on distributed data. In *Proc. of NeurIPS*.

[46] Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, Claudio Orlandi, and Anat Paskin-Cherniavsky. 2013. On the power of correlated randomness in secure computation. In *Proc. of TCC*.

[47] Peter Kairouz, Ziyu Liu, and Thomas Steinke. 2021. The distributed discrete gaussian mechanism for federated learning with secure aggregation. In *Proc. of ICML*. 5201–5212.

[48] Banashri Karmakar, Nishat Koti, Arpita Patra, Sikhar Patranabis, Protik Paul, and Divya Ravi. 2024. Asterisk: Super-fast MPC with a Friend. In *Proc. of IEEE S&P*. 542–560.

[49] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2011. What can we learn privately? *SIAM J. Comput.* 40, 3 (2011), 793–826.

[50] Darya Kaviani and Raluca Ada Popa. 2023. MPC Deployments. https://mpc.cs.berkeley.edu/ [Online; accessed: Apr./2025].

[51] Hannah Keller, Helen Möllering, Thomas Schneider, Oleksandr Tkachenko, and Liang Zhao. 2024. Secure Noise Sampling for DP in MPC with Finite Precision. In *Proc. of ARES*. 1–12.

[52] Ryo Kikuchi, Dai Ikarashi, Takahiro Matsuda, Koki Hamada, and Koji Chida. 2018. Efficient bit-decomposition and modulus-conversion protocols with an honest majority. In *Proc. of ACISP*. 64–82.

[53] Antti Koskela, Mikko A. Heikkilä, and Antti Honkela. 2024. Numerical Accounting in the Shuffle Model of Differential Privacy. In *ICLR*.

[54] Avinash Kumar. 2022. *Towards interactive, adaptive and result-aware big data analytics*. Ph.D. Dissertation. University of California, Irvine.

[55] Xiaoyu Li, Yang Cao, and Masatoshi Yoshikawa. 2023. Locally Private Streaming Data Release with Shuffling and Subsampling. In *Proc. of IEEE ICDEW*. 125–131.

[56] Xiaoguang Li, Ninghui Li, Wenhai Sun, Neil Zhenqiang Gong, and Hui Li. 2023. Fine-grained poisoning attack to local differential privacy protocols for mean and variance estimation. In *Proc. of USENIX Security*. 1739–1756.

[57] Xiaochen Li, Weiran Liu, Hanwen Feng, Kunzhe Huang, Yuke Hu, Jinfei Liu, Kui Ren, and Zhan Qin. 2024. DUMP: Privacy Enhancement Via Dummy Points in the Shuffle Model. *IEEE Trans. on Dependable and Secure Computing* 21, 3 (2024),

1001–1016.

[58] Seng Pei Liew, Tsubasa Takahashi, Shun Takagi, Fumiyuki Kato, Yang Cao, and Masatoshi Yoshikawa. 2022. Network shuffling: Privacy amplification via random walks. In *Proc. of SIGMOD*. 773–787.

[59] Yixuan Liu, Suyun Zhao, Li Xiong, Yuhan Liu, and Hong Chen. 2023. Echo of neighbors: privacy amplification for personalized private federated learning with shuffle model. In *Proc. of AAAI*. 11865–11872.

[60] Donghang Lu and Aniket Kate. 2023. RPM: Robust anonymity at scale. In *Proc. of PETs*.

[61] Qiyao Luo, Yilei Wang, and Ke Yi. 2022. Frequency Estimation in the Shuffle Model with Almost a Single Message. In *Proc. of ACM CCS*. 2219–2232.

[62] Casey Meehan, Amrita Roy Chowdhury, Kamalika Chaudhuri, and Somesh Jha. 2022. Privacy implications of shuffling. In *Proc. of ICLR*.

[63] Payman Mohassel and Yupeng Zhang. 2017. SecureML: A system for scalable privacy-preserving machine learning. In *Proc. of IEEE S&P*. 19–38.

[64] Takao Murakami, Yuichi Sei, and Reo Eriguchi. 2025. Augmented Shuffle Protocols for Accurate and Robust Frequency Estimation under Differential Privacy. In *Proc. of IEEE S&P*. 3892–3911.

[65] San Francisco Controller's Office. 2024. San Francisco Employee Compensation Dataset. https://www.kaggle.com/datasets/san-francisco/sf-employee-compensation [Online; accessed: Jul./2025].

[66] Vibhor Rastogi and Suman Nath. 2010. Differentially private aggregation of distributed time-series with transformation and encryption. In *Proc. of ACM SIGMOD*. 735–746.

[67] Amrita Roy Chowdhury, Chenghong Wang, Xi He, Ashwin Machanavajjhala, and Somesh Jha. 2020. Crypte: Crypto-assisted differential privacy on untrusted servers. In *Proc. of ACM SIGMOD*. 603–619.

[68] Elaine Shi, HTH Chan, Eleanor Rieffel, Richard Chow, and Dawn Song. 2011. Privacy-preserving aggregation of time-series data. In *Proc. of NDSS*.

[69] Elaine Shi, T.-H. Hubert Chan, Eleanor Gilbert Rieffel, and Dawn Song. 2017. Distributed Private Data Analysis: Lower Bounds and Practical Constructions. *ACM Trans. on Algorithms* 13, 4 (2017), 50:1–50:38.

[70] Hyejin Shin, Sungwook Kim, Junbum Shin, and Xiaokui Xiao. 2018. Privacy enhanced matrix factorization for recommendation with local differential privacy. *IEEE Trans. Knowledge and Data Engine.* 30, 9 (2018), 1770–1782.

[71] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. 2022. Towards personalized federated learning. *IEEE Trans. on Neural Networks and Learning Systems* 34, 12 (2022), 9587–9603.

[72] Wei Tong, Haoyu Chen, Jiacheng Niu, and Sheng Zhong. 2024. Data Poisoning Attacks to Locally Differentially Private Frequent Itemset Mining Protocols. In *Proc. of ACM CCS*. 3555–3569.

[73] Shaowei Wang, Changyu Dong, Di Wang, and Xiangfu Song. 2025. Beyond Statistical Estimation: Differentially Private Individual Computation in the Shuffle Model. In *Proc. of USENIX Security*. 2789–2808.

[74] Shaowei Wang, Yun Peng, Jin Li, Zikai Wen, Zhipeng Li, Shiyu Yu, Di Wang, and Wei Yang. 2024. Privacy Amplification via Shuffling: Unified, Simplified, and Tightened. *Proc. of VLDB* 8 (2024), 1870–1883.

[75] Tianhao Wang, Bolin Ding, Min Xu, Zhicong Huang, Cheng Hong, Jingren Zhou, Ninghui Li, and Somesh Jha. 2020. Improving utility and security of the shuffler-based differential privacy. *Proc. of VLDB* 13 (2020), 3545–3558.

[76] Chengkun Wei, Ruijing Yu, Yuan Fan, Wenzhi Chen, and Tianhao Wang. 2023. Securely Sampling Discrete Gaussian Noise for Multi-Party Differential Privacy. In *Proc. of ACM CCS*. 2262–2276.

[77] Hao Wu, Olga Ohrimenko, and Anthony Wirth. 2022. Walking to Hide: Privacy Amplification via Random Message Exchanges in Network. In *ArXiv Preprint*.

[78] Yongji Wu, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. 2022. Poisoning Attacks to Local Differential Privacy Protocols for Key-Value Data. In *Proc. of USENIX Security*. 519–536.

[79] Lihua Yin, Jiyuan Feng, Hao Xun, Zhe Sun, and Xiaochun Cheng. 2021. A Privacy-Preserving Federated Learning for Multiparty Data Sharing in Social IoTs. *IEEE Trans. on Network Science and Engineering* 8, 3 (2021), 2706–2718.

[80] Mingxun Zhou and Elaine Shi. 2022. The power of the differentially oblivious shuffle in distributed privacy mechanisms. *Cryptology ePrint Archive* (2022).

[81] Wennan Zhu, Peter Kairouz, Brendan McMahan, Haicheng Sun, and Wei Li. 2020. Federated heavy hitters discovery with differential privacy. In *Proc. of AISTATS*. 3837–3847.