PrivacyGo: Privacy-Preserving Ad Measurement with Multidimensional Intersection

Jian Du, Haohao Qian, Shikun Zhang, Wen-jie Lu, Donghang Lu,
Yongchuan Niu, Bo Jiang, Yongjun Zhao, and Qiang Yan
TikTok Inc.

ABSTRACT

In digital advertising, accurate measurement is essential for optimizing ad performance, requiring collaboration between advertisers and publishers to compute aggregate statistics—such as total conversions—while preserving user privacy. Traditional secure two-party computation methods allow joint computation on single-identifier data without revealing raw inputs, but they fall short when multidimensional matching is needed and leak the intersection size, exposing sensitive information to privacy attacks.

This paper tackles the challenging and practical problem of multi-identifier private user profile matching for privacy-preserving ad measurement, a cornerstone of modern advertising analytics. We introduce a comprehensive cryptographic framework leveraging reversed Oblivious Pseudorandom Functions (OPRF) and novel blind key rotation techniques to support secure matching across multiple identifiers. Our design prevents cross-identifier linkages and includes a differentially private mechanism to obfuscate intersection sizes, mitigating risks such as membership inference attacks.

We present a concrete construction of our protocol that achieves both strong privacy guarantees and high efficiency. It scales to large datasets, offering a practical and scalable solution for privacy-centric applications like secure ad conversion tracking. By combining rigorous cryptographic principles with differential privacy, our work addresses a critical need in the advertising industry, setting a new standard for privacy-preserving ad measurement frameworks.

1 INTRODUCTION

Private Ad in Practice: Digital advertising relies on cross-site and cross-app user tracking to link ad impressions stored by ad providers with user conversions recorded by advertisers, using third-party cookies or mobile device identifiers. This connection provides insights into ad performance and audience reach, enabling measurement of key metrics such as click-through rates, conversions, and return on ad spend (ROAS). By analyzing user behavior across platforms, marketers can further refine campaign strategies.

However, growing privacy concerns over tracking consumer behavior across ad providers and advertisers—driven by regulations such as GDPR and CCPA, along with industry changes such as third-party cookie deprecation and mobile identifier restrictions—are transforming the digital advertising landscape. In response, privacy-preserving technologies are becoming essential to balance effective ad measurement with robust data protection.

Ideally, if a unique identifier is used consistently across both the ad provider and the advertiser sides, cryptographic techniques that enable privacy-preserving set intersection can compute shared statistics without revealing sensitive data from either party, effectively preventing cross-platform tracking. However, in practice, matching user identities across platforms is complex because users often have different identifiers, such as varying registration emails or phone numbers. To improve accuracy, multiple identifiers must be aligned based on specific rules. Techniques like the waterfall matching approach (also known as advanced matching [1]), widely used in the advertising industry, prioritize identifier hierarchy to reduce false positives and enhance reliability. This method systematically resolves identity discrepancies by sequentially matching identifiers from the most reliable to the least reliable, ensuring more precise attribution. The underlying mathematics extends beyond the simple set intersection and is known as *multidimensional intersection*, where multiple attributes are matched simultaneously to achieve context-aware identity resolution with higher accuracy.

1.1 Problem Characteristics and Requirements

In Figure 1, we provide a formal description of the functionality that we aim to compute privately. This figure illustrates the process by which one party, P_B, learns the aggregated sum c according to the logic of the waterfall matching[1] for a multidimensional intersection. At first glance, the target problem illustrated in Figure 1 may appear similar to the private-match-for-compute protocols such as [2–7]. However, we highlight two key differences that distinguish our target problem from theirs:

1) Multidimensional Intersection. For the case of a single ID, two databases are joined using exact matching on the common identifier, which is handled by most existing private set intersection protocols. However, when it comes to multiple IDs, it often results in many-to-many connections. For instance, some records in the databases may be matched by one identifier, e.g., email address, while other records are matched by a different identifier, e.g., phone number. The waterfall matching converts the many-to-many connections to one-to-one connections in a *hierarchically iterative way*. Specifically, the waterfall matching approach ranks identifiers in a predefined order. A record is considered matched if one of its identifiers matches. To avoid many-to-many connections, the matched rows are removed before the matching procedure advances to the next identifier. Waterfall matching for multidimensional intersections is effective because:

(1) Each record is matched only once, preventing many-to-many connections.
(2) Higher-priority identifiers take precedence, reducing false positives and enhancing matching reliability.

2) Hiding Cross-ID Leakages. Leveraging multiple identifiers improves match rates, which is critical for the advertising business. However, a simple approach that runs a single-ID matching protocol

*Corresponding author: Jian Du. Email: jian.du@tiktok.com.

[1] Indeed, the waterfall matching can work for more than two identifiers. To simplify the presentation, we only consider two identifiers here.

Functionality $\mathcal{F}_{wfm}$

On receiving identifiers $\mathsf{ID}_A^1, \mathsf{ID}_A^2$ from P_A, $\mathsf{ID}_B^1, \mathsf{ID}_B^2$ and payloads T from P_B, this functionality do:

(1) Match on the 1st identifier

$$J_A^1 = \{i : \mathsf{ID}_A^1[i] \in \mathsf{ID}_B^1\}, \quad J_B^1 = \{j : \mathsf{ID}_B^1[j] \in \mathsf{ID}_A^1\}$$

(2) Unmatched records on the 2nd identifier

$$\mathsf{rID}_A^2 = \{\mathsf{ID}_A^2[i]\}_{i \notin J_A^1}, \quad \mathsf{rID}_B^2 = \{\mathsf{ID}_B^2[j]\}_{j \notin J_B^1}$$

(3) Match on the 2nd identifier

$$J_A^2 = \{i : \mathsf{ID}_A^2[i] \in \mathsf{rID}_B^2\} \setminus J_A^1$$
$$J_B^2 = \{j : \mathsf{ID}_B^2[j] \in \mathsf{rID}_A^2\} \setminus J_B^2$$

(4) Give P_A the sizes $s_1 = |J_A^1|$ and $s_2 = |J_B^2|$.

(5) Give P_B the sum $c = \sum_{i \in J_B^1 \cup J_B^2} T[i]$ and s_1, s_2.

Figure 1: The waterfall matching and sum functionality. Note that the matches on the 2nd ID column can also be given as $\{i : \mathsf{rID}_A^2[i] \in \mathsf{rID}_B^2\}$ and $\{j : \mathsf{rID}_B^2[j] \in \mathsf{rID}_A^2\}$, which needs an extra mapping to a global index in the payload column.

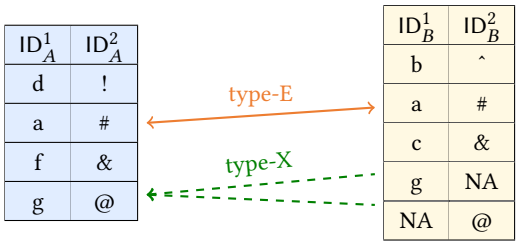


Figure 2: Example of the type-E cross-ID leakage and the type-X cross-ID leakage. 'NA' indicates a missing value.

separately for each identifier can unintentionally reveal additional information.

Figure 1 defines two types of cross-ID leakages: **type-E** and **type-X** (see Figure 2 for an example). Type-E leakages occur when parties learn that a record is matched by multiple IDs, while type-X leakages occur when one party (say, P_A) learns that a record from P_B matches multiple records on P_A's side. For example, in Figure 2, the fourth row of the left table matches both the fourth and fifth rows of the right table. Type-X leakages are common in ad applications, where users may have multiple accounts on a publisher's platform registered with different identifiers, such as phone numbers and email addresses.

Both type-E and type-X leakages increase the risk of membership leakage, as demonstrated by membership inference attacks described in the literature [8, 9]. From a business perspective, these leakages allow either party to track user behaviour on the other side. For instance, the ad provider could learn if a user has converted on the advertiser's side, which could be particularly sensitive for industries like healthcare and finance. Addressing these leakages is crucial to ensure privacy while enabling effective matching.

2 TECHNICAL OVERVIEW

2.1 Distributed Evaluation of Reversed and Blindly Updatable Pseudorandom Function

We first recap the recipe for building a private set intersection protocol. Suppose P_A inputs set X and P_B inputs set Y. They invoke an Oblivious Pseudorandom Function (OPRF) protocol, which gives the PRF evaluations $\{F_k(x) \mid x \in X\}$ to P_A and gives the key k to P_B. This directly allows obtaining the intersection $X \cap Y$ by having P_B send $\{F_k(y) \mid y \in Y\}$ to P_A. Since the interest is in evaluating a function f over the intersection, rather than the intersection itself, it is natural to perform the OPRF in a reversed manner. For example, P_B obtains $\{F_k(x) \mid x \in X\}$, the PRF evaluations on P_A's input. By performing the reversed OPRF twice and exchanging the PRFs, P_A and P_B can obtain the set cardinality $f(X \cap Y) = |X \cap Y|$ without revealing the intersection. This *reversed OPRF* implicitly demands a *distributed OPRF* evaluation, i.e., the corresponding PRF key is secretly shared between the two parties.

We demonstrate that the reversed OPRF already enables efficient row deletion required by the waterfall matching logic. Specifically, P_A can locally remove matched rows from $F_k(y) \mid y \in Y$ before exchanging PRF evaluations with P_B. Since the key remains unknown to both parties, P_B is oblivious to which rows have been deleted. This private row deletion inherently mitigates type-E leakages.

To address type-X leakages, we introduce a novel mechanism for blind PRF key updates. Given an evaluation $F_k(x)$, P_B interacts with P_A to obtain $F_{k'}(x)$—an evaluation under a new shared key—without revealing the linkage between $F_k(x)$ and x. Importantly, this ensures that even if P_A knows the input x, it remains unaware of the key transition, thus preserving input privacy across key updates. Our method represents a significant advancement in leakage resilience by seamlessly integrating key blinding into private matching protocols.

Finally, we formalize the desired OPRF functionality in Figure 3, defining it as a *Distributed, Blindly Updatable, and Reversed Pseudorandom Function (durPRF)*. This new $\mathcal{F}_{durPRF}$ construction enables a privacy-preserving multidimensional intersection for implementing a waterfall matching protocol with robust leakage resistance. Our blind key update mechanism addresses type-X leakages by fully decoupling key transitions from input knowledge. This ensures that even if an input is partially known to P_A, the linkage between evaluations under different keys remains hidden, preserving privacy across updates. Additionally, the reversed OPRF design allows P_A to efficiently delete matched rows from $F_k(y) \mid y \in Y$ without exposing deletion patterns to P_B, inherently mitigating type-E leakages. By integrating these capabilities, our protocol delivers privacy guarantees against adversaries seeking to exploit linkage or key-correlated information, providing a comprehensive defense against both type-E and type-X leakages.

2.2 Enhancing Privacy via Differential Privacy

Recent research highlights that size-revealing (single-ID) private set intersection (PSI) protocols are susceptible to membership inference attacks, posing significant privacy risks in advertising contexts [8, 9]. In these attacks, an ad publisher can infer which users on its platform viewed an ad but did not complete a purchase

Functionality $\mathcal{F}_{durPRF}$

There are two parties P_A and P_B.

Initialization: Upon receiving $\textbf{Init}.(\mathsf{sid}, k_A)$ from P_A, and $\textbf{Init}.(\mathsf{sid}, k_B)$ from P_B, initialize an empty table T_{sid}, and record $(\mathsf{sid}, k_A \oplus k_B)$. Ignore all subsequent init queries for the session ID sid.

Evaluation: On $\textbf{Send}.(\mathsf{sid}, x_1, \cdots, x_n)$ from P_A (sender), and $\textbf{Recv}.(\mathsf{sid}, n)$ from P_B (receiver) do:

- Set $T_{\mathsf{sid}}[x_i] \xleftarrow{\$} \{0,1\}^\lambda$ if any empty entry.
- Give $T_{\mathsf{sid}}[x_1], \cdots, T_{\mathsf{sid}}[x_n]$ to P_B.

The roles can be flipped, *i.e.*, P_B acts as the sender and P_A acts as the receiver.

Update: Upon receiving $\textbf{Update}.(\mathsf{sid}, \mathsf{sid}', z_1, \cdots, z_m)$ from P_B, and $\textbf{Update}.(\mathsf{sid}, \mathsf{sid}', m)$ from P_A do:

- Set $z'_j \xleftarrow{\$} \{0,1\}^\lambda$ for $j = 1, 2, \cdots, m$.
- For each z_j, find an entry x_j in T_{sid} such that $T_{\mathsf{sid}}[x_j] = z_j$. If not any then set $x_j = \perp$.
- If $x_j \neq \perp$ and the entry $T_{\mathsf{sid}'}[x_j]$ is not empty, then overwrite $z'_j \leftarrow T_{\mathsf{sid}'}[x_j]$. Otherwise if $x_j \neq \perp$ and the entry $T_{\mathsf{sid}'}[x_j]$ is empty, then set the entry $T_{\mathsf{sid}'}[x_j] \leftarrow z'_j$.
- Finally return $z'_1, \cdots, z'_m$ to P_B.

The roles can be flipped too, *i.e.*, P_B acts as the assistant and P_A acts as the updater.

Figure 3: The functionality of *durPRF*.

with the advertiser. This compromises sensitive business data, as the advertiser may not wish to disclose granular user conversion behavior. Such leakages not only undermine the advertiser's competitive edge but also raise potential compliance concerns under privacy regulations such as the GDPR, which mandates strict limits on personal data processing and requires minimizing data exposure between parties. Addressing these vulnerabilities is critical to ensuring privacy-preserving advertising practices that align with regulatory obligations and protect proprietary business insights.

Multidimensional intersections are more vulnerable to membership inference attacks. Particularly, we conduct the membership inference attacks [8, 9] against the size-revealing and multi-ID protocol [10]. The results show that the success rates doubled (*e.g.*, from 2.5% to 6.6%) when the same attacks were carried out over a dataset with 4 IDs. See Figure 7(a) in §6.

To mitigate intersection size leakages, we adopt a strategy based on differentially private (DP) mechanisms. Specifically, we introduce a novel *dual-sided DP* mechanism that enables both parties to independently sample and inject dummy identifiers into their datasets, ID_A and ID_B, creating two augmented sets, $\widetilde{\mathsf{ID}}_A$ and $\widetilde{\mathsf{ID}}_B$, respectively.

Input privacy is preserved by the private waterfall matching protocol, while the intersection size privacy is guaranteed by ensuring that the distribution of $|\widetilde{\mathsf{ID}}_A \cap \widetilde{\mathsf{ID}}_B|$ is differentially private with respect to the true size $|\mathsf{ID}_A \cap \mathsf{ID}_B|$. This mechanism obfuscates the intersection size, making it indistinguishable within a carefully controlled privacy budget.

The number of dummy identifiers added depends on the size of the original data set and the desired privacy parameters. By leveraging our *durPRF* construction, which enables independent processing of dimensions, we achieve a tighter DP analysis. This design allows us to apply parallel composition across multiple dimensions, significantly reducing the exponential overhead of dummy identifiers required under a sequential composition approach. Consequently, our method scales more efficiently while maintaining rigorous privacy guarantees, offering a substantial improvement in the practicality of privacy-preserving multi-dimensional set operations.

On DP to Multi-Dimension PSI. A naive approach of applying DP to PSI protocols dimension-by-dimension may appear feasible at first glance. However, while DP mechanisms can theoretically mitigate privacy risks posed by type-E and type-X leakages, this approach incurs exponentially increasing overheads. This stems from the fact that the intersection size can vary by up to the full dataset size, as demonstrated by the scenario where all records of one party could potentially match different identifiers under type-E/X leakage conditions. In contrast, our waterfall matching protocol **eliminates** these leakage vectors entirely, enabling a significantly reduced overhead when integrated with DP mechanisms.

On the Input Validations. The intersection sizes are effectively randomized by our *dual-sided DP* mechanism, provided both parties correctly sample and add dummies to their input sets. However, an adversary could bypass this protection and reveal the exact intersection size by simply skipping the sampling step. To prevent this "simple attack," we employ secure hardware, such as Intel SGX, for code attestation. Each party uses its own secure hardware to verify that the (encrypted) messages they send are generated by the intended code, including the correct dummy sampling. Since the dummy set is common information, it can be hardcoded within the attested code. Notably, confidentiality of the hardware is unnecessary in this setting, as it is managed and executed by the same party.

2.3 Our Contributions

Our contributions are summarized as follows:

- **A general framework for private waterfall matching without cross-ID leakage.**
  We propose a novel framework to achieve the waterfall matching functionality while eliminating cross-ID leakages. Central to our approach is a functionality we define as the *Distributed, Updatable, and Reversed Pseudorandom Function* ($\mathcal{F}_{durPRF}$), described in Figure 3. We present two constructions for $\mathcal{F}_{durPRF}$ using hashed Diffie-Hellman (DH) and Yao's Garbled Circuit [11].
  Preventing cross-ID leakage is critical because it allows the use of lighter, more efficient differential privacy (DP) mechanisms to enhance privacy. Applying DP to protocols with cross-ID leakage would result in significantly higher overhead, as privacy loss grows exponentially with the number of IDs. Our solution effectively decouples privacy loss from cross-ID matches, leading to more scalable and practical implementations.
- **A tightly bounded DP mechanism for protecting intersection sizes.** We design a differentially private mechanism where both parties independently sample and add dummy

identifiers to their input sets. This protects the intersection size with minimal overhead.
We present two analyses to reduce dummy size, improving efficiency. The first uses $\mathcal{F}_{durPRF}$ to decompose cross-ID leakage, allowing parallel composition with linear dummy growth relative to the number of identifiers, compared to the exponential growth in sequential composition. The second uses convolution techniques from [12] to minimize dummies across multiple waterfall matching executions, such as aggregating conversion data from two ad campaigns.

- **Two variants and an optimized implementation.** We extend our protocol to support both P_A and P_B providing payloads for aggregation and enable general functions beyond summation by sharing matched homomorphic encryption (HE) ciphertexts for secret sharing.
We implement the protocol using DH-based *durPRF* and a lattice-based HE scheme, with HE optimizations that handle two million records across three ID columns in about 2 minutes (8 threads) over a 100Mbps connection. Our DP-enhanced protocol achieves practical performance, and we plan to release our implementation publicly.

2.4 Related Works

In Private Set Intersection (PSI) two parties, P_A with a set X and P_B with a set Y , securely compute the intersection $X \cap Y$, without leaking the information of the items that are not in the intersection has been studied extensively in a long sequence of works [2, 4, 5, 5, 6, 10, 13–21].

In many settings, the goal is to compute some function f over the intersection set, i.e., $f(X \cap Y)$, rather than knowing the intersection itself [2, 22–24]. PSI-cardinality is one example of such an the two parties are limited to learning only the cardinality (or size) of the intersection [23, 24]. The private intersection-sum functionality introduced by Ion *et al.* [2] is another example where one of the input sets has integer values associated with the elements in the set and the two parties aggregate of the integer values associated with the intersection set. The circuit-based PSI protocols such as [5, 6, 25] enable the computation of arbitrary symmetric functions securely over the intersection. To the best of our knowledge, most of the existing PSI protocols are designed for the single identifier.

Private-ID [10] is the only publicly available solution for multi-ID matching that comes with membership leakage. More precisely, the participants in [10] can know the intersection size of any subset of IDs, leading to an exponential leakage of membership information.

Kacsmar *et al.* [26] employ DP mechanisms for the cardinality function $f(X \cap Y) = |X \cap Y|$ in a client-server scenario in where only the client can know the result. Consequently, only central DP is necessary for their setting. However, in our context, where each party is inquisitive about membership and there is a potential for membership attacks, the application of central DP is not suitable. Instead, a distributed DP approach is required.

Oblivious PRF. In the literature, the oblivious evaluation of a PRF is commonly defined as a two-party protocol where P_A provides a key k and the other party P_B provides an input x. At the end of the protocol execution, P_B obtains $F_k(x)$. There are many constructions for oblivious PRF protocols, such as [27–29]. Moreover, we can modify an OPRF protocol to a key-distributed and output-reversed counterpart, i.e., the key is distributed between the two parties and P_A obtains the evaluation, using homomorphic encryption as described by [2]. The double-hash approach from [23] is a reversed OPRF but does not provide key rotation capability.

Many OPRF protocols based on the Dodis-Yampolskiy (DY) PRF can support key rotation (i.e., updating the PRF key). However, it seems to require nontrivial efforts to support blind key rotation when the DY-PRF key is distributed between two parties. We refer to the survey by Casacuberta *et al.* [30] for more details on recent oblivious PRF constructions.

Circuit-based Solutions. One can also leverage the technique called circuit-based PSI (CPSI), such as [5, 6, 25], which can eliminate the leakage of the intersection sizes. However, the modification of the current single-ID CPSI protocols to the multi-ID setting involves non-trivial work. A CPSI protocol is basically an asymmetric set membership testing. That is it allows P_B to query whether its ID match the ones own by P_A. At the end of the protocol execution, P_B obtains a secretly shared testing bits $b_i = 1$ if $\mathsf{id}^1_{B,i} \in \mathsf{ID}^1_A$, or $b_i = 0$ otherwise. One of the difficulties of adopting the CPSI to the multi-ID setting is to obliviously delete records that already matched from the previous IDs. That is because the position of each ID is computed using a hash function. For instance, the position of the identifier "a@123" is computed via $i = \mathsf{Hash}(\text{"a@123"})$, and then it is assigned to the i-th position in the query vector. It renders an inconsistent order of result bits when performing the queries on a record of different IDs. Communication extensive cryptographic protocols (e.g., oblivious permutation) are needed to align the order of the of testing bits (and the payloads).

In the ad applications, the intersection size can be significantly smaller than the input sets, *i.e.*, $|X \cap Y| \ll \min(|X|, |Y|)$. For instance, a typical ratio is less than 1%. In other words, to evaluate a downstream function over the intersection using fully private techniques might introduce a large overhead on the downstream computation.

3 PRELIMINARIES

3.1 Definitions

Definition 1. *(Pseudorandom Functions [31]) Let* $F : \{0,1\}^* \times \{0,1\}^* \mapsto \{0,1\}^*$ *be an efficient, length-preserving, keyed function. We say* F *is a pseudorandom function if for all probabilistic polynomial-time distinguishers* $\mathcal{D}$*, there exists a negligible function negl such that:* $\Pr[\mathcal{D}^{F_k(\cdot)}(1^n) = 1] - \Pr[\mathcal{D}^{f_n(\cdot)}(1^n) = 1] \leq \mathsf{negl}(n)$*, where* $k \xleftarrow{\$} \{0,1\}^n$ *is chosen uniformly at random and* f_n *is chosen uniformly at random from the set of functions mapping n-bit strings to n-bit strings.*

Honest-but-Curious Privacy. We recap the privacy definition from [32, 33]. Let $F : \{0,1\}^* \times \{0,1\}^* \mapsto \{0,1\}^*$ be a deterministic functionality where $F_0(x_0, x_1)$ (resp. $F_1(x_0, x_1)$) denotes the 1st element (resp. the 2nd) of $F(x_0, x_1)$, and let Π be a two-party protocol for computing F. The view of P_A (resp. P_B) during an execution of Π on (x_0, x_1) is denoted $\mathsf{View}^{\Pi}_A(x_0, x_1)$ (resp. $\mathsf{View}^{\Pi}_B(x_0, x_1)$).

Definition 2. *(Honest-but-Curious Privacy) For a function* F*, we say that* Π *privately computes* F *if there exist probabilistic polynomial*

time algorithms, denoted Sim_0 *and* Sim_1*, such that*

$$\{\mathsf{Sim}_0(x_0, F_0(x_0, x_1))\}_{x_0,x_1} \overset{c}{\equiv} \{\mathsf{View}_A^{\Pi}(x_0, x_1)\}_{x_0,x_1}$$
$$\{\mathsf{Sim}_1(x_1, F_1(x_0, x_1))\}_{x_0,x_1} \overset{c}{\equiv} \{\mathsf{View}_B^{\Pi}(x_0, x_1)\}_{x_0,x_1},$$

where $\overset{c}{\equiv}$ *denotes computational indistinguishability. This definition states that the views of the parties can be properly constructed by a polynomial time algorithm given the party's input and output solely. Also, the parties here are semi-honest and the view is therefore exactly according to the protocol specification.*

Differential Privacy. We recall a definition of approximate differential privacy for interactive two-party protocols, following [34]. Let Σ be a finite alphabet and for strings $x, y \in \Sigma^n$, let $|x - y|_H$ denote the Hamming distance between x and y.

Definition 3. *((ϵ, δ)-Differential Privacy) A mechanism $\mathcal{M}$ on Σ^n is a family of probability distributions $\{\mu_x : x \in \Sigma^n\}$ on $\mathcal{R}$. The mechanism is (ϵ, δ)-differentially private if for every x and x' such that $|x - x'|_H = 1$ and every measurable subset $S \subset \mathcal{R}$ we have*

$$\mu_x(S) \leq \exp(\epsilon)\mu_{x'}(S) + \delta.$$

The definition of differential privacy naturally extends to interactive protocols, by requiring that the views of all parties be differentially private in respect to other parties' inputs. More specifically, let $\mathsf{View}_{\Pi}^{A}(x, y)$ be the joint probability distribution over x, the transcript of the protocol Π, private randomness of P_A, where the probability space is private randomness of both parties. For each x, $\mathsf{View}_{\Pi}^{A}(x, y)$ is a mechanism over the y's. Let $\mathsf{View}_{\Pi}^{B}(x, y)$ be similarly defined view of P_B whose input is y.

Definition 4. *(Differential privacy for two-party protocols) We say that a protocol Π enables (ϵ, δ)-differential privacy if the mechanism $\mathsf{View}_{\Pi}^{A}(x, y)$ is (ϵ, δ)-differentially private for all values of x, and same holds for $\mathsf{View}_{\Pi}^{B}(x, y)$ for all values y.*

3.2 Security Model

We chose to target the security against honest-but-curious, *i.e.* semi-honest adversaries, due to the strong efficiency and monetary requirements of ad conversion applications. The semi-honest model provides strong privacy protections against data breaches on either side since semi-honest protocols leak nothing beyond the prescribed protocol output.

We protect the intersection size by having each party independently sample and add dummy identifiers from a common set of dummies. Even under the semi-honest setting, all parties should follow the protocol descriptions, e.g., sampling dummy identifiers from the proper set. However, there is a prisoner's dilemma where an adversary can learn the exact intersection size if he/she simply skips the sampling step, and the honest party cannot detect this behavior. To guarantee that each party correctly performs the DP mechanism, we rely on secure hardware such as Intel SGX to provide code attestation. In brief, the outputs generated by the hardware are indeed from the code that it attested to. Note that we **do not** assume confidentiality for the secure hardware since, in our case, a party uses his/her own hardware to convince the other party that he/she has executed a public piece of code properly. This practice of using secure hardware to achieve code attestation is also employed by other works such as [35].

3.3 Additively Homomorphic Encryption (AHE)

An AHE scheme $\mathcal{HE}$ = (Gen, Enc, Dec, Sum, Refresh) consists of 5 algorithms. Gen outputs a public-private key pair (pk, sk), and specifies a message space $\mathcal{M}$. Given the public key pk and a plaintext message $m \in \mathcal{M}$, one can compute a ciphertext $\mathcal{HE}.\mathsf{Enc}_{\mathsf{pk}}(m)$, an encryption of m under pk. Given the secret key sk and a ciphertext $C = \mathsf{Enc}(m)$, one can run $\mathsf{Dec}(C)$ to recover the plaintext m.

Given the public key pk and a set of ciphertexts $\{C_i\}$ encrypting messages $\{m_i\}$, one can homomorphically compute a ciphertext encrypting the sum of the underlying messages, which we denote for ease of exposition as: $\mathsf{Enc}(\sum_i m_i) = \mathsf{Sum}(\{C_i\}_i)$. We will also use the property that one can randomize ciphertexts using a randomized procedure denoted as Refresh. The two distributions $\mathsf{Refresh}(\mathsf{Sum}(\{\mathsf{Enc}(m_i)\}_i))$ and $\mathsf{Refresh}(\mathsf{Enc}(\sum_i m_i))$ are statistically close, even against an adversary that holds the decryption key sk.

We can instantiate the AHE scheme by the Brakerski/Fan-Vercauteren scheme (BFV) scheme [36, 37] or the Paillier scheme [38]. Particularly, the current work [2] used the Paillier scheme while we present some optimizations for the BFV scheme.

4 PROPOSED PROTOCOL FOR WATERFALL MATCHING

4.1 Private Waterfall Matching with Sum Protocol from $\mathcal{F}_{durPRF}$

The concrete construction of our private waterfall matching protocol is given in Figure 4. Our protocol operates in five stages.

(1) In the first stage, P_B sends homomorphic encryption of P_B's payloads to P_A along with his public key.
(2) The two parties first send the "random tags" of their IDs using multiple *durPRF*. For instance, on the b-th ID column, P_B sends $\mathcal{F}_{durPRF}.\mathbf{Send}(\mathsf{sid}_b, \mathsf{ID}_B^b)$ and P_A receives $\mathsf{Tag}_B^b \leftarrow \mathcal{F}_{durPRF}.\mathbf{Recv}(\mathsf{sid}_b)$. That is P_A obtains the PRF evaluations on P_B's b-th ID column under a shared key k_b. On the other hand, P_B receives the PRF evaluations on P_A's IDs under the corresponding key, denoted as Tag_A^b.
(3) By exchanging the tags on the 1st ID column, P_A and P_B can obtain the intersection in a reversed way. For instance, after receiving $\widetilde{\mathsf{Tag}}_A^1$ in a shuffled order from P_B, P_A knows the matching positions $J_B^1 = \{i : \mathsf{Tag}_B^1[i] \in \widetilde{\mathsf{Tag}}_A^1\}$ on the 1st ID column, but P_A does not learn the exact matched row index in P_A's dataset due to the shuffling. Similarly P_B knows $J_A^1 = \{i : \mathsf{Tag}_A^1[i] \in \widetilde{\mathsf{Tag}}_B^1\}$.
(4) Moreover, to avoid the type-E cross-ID leakage, P_A and P_B can first remove the matched rows from the received tags of the 2nd ID column. Particularly, P_A prepares $\mathsf{rTag}_B^2 = \{\mathsf{Tag}_B^2[i] = F_{k_2}(\mathsf{ID}^B[i])\}_{i \notin J_B^1}$ and P_B prepares $\mathsf{rTag}_A^2 = \{\mathsf{Tag}_A^2[i] = F_{k_2}(\mathsf{ID}_A^2[i])\}_{i \notin J_A^1}$. Then they perform the matching on the remaining tags. For instance, on receiving $\widetilde{\mathsf{rTag}}_A^2$ from P_B, P_A can know the matching positions due to the 2nd ID column: $J_B^2 = \{\sigma(i) : \mathsf{rTag}_B^2[i] \in \widetilde{\mathsf{rTag}}_A^2\}$ where σ is an index mapping from $|\mathsf{rTag}_A^2| \mapsto |\mathsf{ID}_A^1|$ that maps the matching positions in rTag_A^2 to their global positions in

Private Waterfall Matching and Sum Protocol Π_{wfm}

	P_A		P_B
1 :	P_A : identifiers $\mathsf{ID}_A^1, \mathsf{ID}_A^2$		P_B : identifiers $\mathsf{ID}_B^1, \mathsf{ID}_B^2$, and payloads T
	..... Stage-1: Setups		
2 :	$k_A^1, k_A^2, k_A^3 \xleftarrow{\$} \mathbb{Z}_q$		$k_B^1, k_B^2, k_B^3 \xleftarrow{\$} \mathbb{Z}_q$
3 :	$\mathcal{F}_{durPRF}.\mathbf{init}(\mathsf{sid}_b, k_A^b)$ for $b = 1,2,3$		$\mathcal{F}_{durPRF}.\mathbf{init}(\mathsf{sid}_b, k_B^b)$ for $b = 1,2,3$
4 :			$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathcal{HE}.\mathsf{Gen}(1^\lambda)$
5 :		$\xleftarrow{\mathsf{pk}, C}$	$C = \{\mathcal{HE}.\mathsf{Enc}_{\mathsf{pk}}(T[j])\}_{j=1}^{\lvert T\rvert}$
	..... Stage-2: PRF on all identifiers		
6 :	$\mathcal{F}_{durPRF}.\mathbf{Send}(\mathsf{sid}_b, \mathsf{ID}_A^b)$ for $b = 1,2$		$\mathsf{Tag}_A^b \leftarrow \mathcal{F}_{durPRF}.\mathbf{Recv}(\mathsf{sid}_b)$ for $b = 1,2$
7 :	$\mathsf{Tag}_B^b \leftarrow \mathcal{F}_{durPRF}.\mathbf{Recv}(\mathsf{sid}_b)$ for $b = 1,2$		$\mathcal{F}_{durPRF}.\mathbf{Send}(\mathsf{sid}_b, \mathsf{ID}_B^b)$ for $b = 1,2$
	..... Stage-3: Matching on the first identifier		
8 :		$\xleftarrow{\widetilde{\mathsf{Tag}}_A^1 \text{ in shuffled order}}$	
9 :		$\xrightarrow{\widetilde{\mathsf{Tag}}_B^1 \text{ in shuffled order}}$	
10 :	Matches $J_B^1 \leftarrow \{j : \mathsf{Tag}_B^1[j] \in \widetilde{\mathsf{Tag}}_A^1\}$		$J_A^1 \leftarrow \{j : \mathsf{Tag}_A^1[j] \in \widetilde{\mathsf{Tag}}_B^1\}$
11 :	Mismatches $D_B^1 \leftarrow \{1, 2, \cdots, \lvert\mathsf{Tag}_B^1\rvert\} \setminus J_B^1$		$D_A^1 \leftarrow \{1, 2, \cdots, \lvert\mathsf{Tag}_A^1\rvert\} \setminus J_A^1$
12 :	Book-keep: $\sigma_B^1[j] \leftarrow D_B^1[j]$ for $1 \le j \le \lvert D_B^1\rvert$		
	..... Stage-4: Key-update then matching on the second identifier		
13 :	Keep the unmatched records $\mathsf{rTag}_B^2 \leftarrow \{\mathsf{Tag}_B^2[i]\}_{i \in D_B^1}$		$\mathsf{rTag}_A^2 \leftarrow \{\mathsf{Tag}_A^2[i]\}_{i \in D_A^1}$
14 :	$\mathsf{rTag}_B^2 \leftarrow \mathcal{F}_{durPRF}.\mathbf{Update}(\mathsf{sid}_2, \mathsf{sid}_3, \mathsf{rTag}_B^2)$		$\mathcal{F}_{durPRF}.\mathbf{Update}(\mathsf{sid}_2, \mathsf{sid}_3)$
15 :	$\mathcal{F}_{durPRF}.\mathbf{Update}(\mathsf{sid}_2, \mathsf{sid}_3)$		$\mathsf{rTag}_A^2 \leftarrow \mathcal{F}_{durPRF}.\mathbf{Update}(\mathsf{sid}_2, \mathsf{sid}_3, \mathsf{rTag}_A^2)$
16 :		$\xleftarrow{\widetilde{\mathsf{rTag}}_A^2}$	
17 :	$J_B^2 \leftarrow \{\sigma_B^1[j] : \mathsf{rTag}_B^2[j] \in \widetilde{\mathsf{rTag}}_A^2\}$		
	.....Final stage: aggregation.....		
18 :	Sum $\mathsf{ct} \leftarrow \mathcal{HE}.\mathsf{Sum}(\{\widetilde{C}[j]\}_{j \in J_B^1 \cup J_B^2})$	$\xrightarrow{\mathsf{ct}' \leftarrow \mathcal{HE}.\mathsf{Refresh}(\mathsf{ct})}$	Output $\mathcal{HE}.\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}')$.

Figure 4: Our private waterfall matching protocol under $\mathcal{F}_{durPRF}$-hybrid model. $\mathcal{HE}$ is an AHE scheme.

ID_A^1. **The type-E cross-ID leakage is avoided from the sense that $J_B^1 \cap J_B^2 = \emptyset$.**

However, it might introduce the type-X cross-ID leakage. For instance, when $\exists i \in J_B^1 \Rightarrow \mathsf{Tag}_B^2[i] \in \widetilde{\mathsf{rTag}}_A^2$, it reveals to P_A that a record i of P_B is matched by two records of P_A, one due to the 1st ID, and the other is due to the 2nd ID. To mitigate the type-X cross-ID leakage, we further leverage the blind-update command as follows

P_B : $\mathsf{rTag}_A^2 \leftarrow \mathcal{F}_{durPRF}.\mathbf{Update}(\mathsf{sid}_2, \mathsf{sid}_2', \mathsf{rTag}_A^2)$

P_A : $\mathsf{rTag}_B^2 \leftarrow \mathcal{F}_{durPRF}.\mathbf{Update}(\mathsf{sid}_2, \mathsf{sid}_2', \mathsf{rTag}_B^2)$

so that $\mathsf{Tag}_B^2 \cap \mathsf{rTag}_A^2 = \emptyset$ and $\mathsf{Tag}_A^2 \cap \mathsf{rTag}_B^2 = \emptyset$ due to the distinct PRF keys. Thus the type-X leakage is also avoided.

(5) P_A can then aggregate the HE ciphertexts using the matching lists J_B^1 and J_B^2.

Theorem 1. *If the AHE scheme $\mathcal{HE}$ is semantic secure then the protocol Π_{wmf} in Figure 4 securely realizes the $\mathcal{F}_{wmf}$ functionality of Figure 1 against semi-honest adversaries under $\mathcal{F}_{durPRF}$-hybrid.*

Proof. Let $\mathcal{A}$ be a polynomial-time adversary that may corrupt P_A and P_B. We construct a simulator $\mathtt{Sim}$ with access to functionality $\mathcal{F}_{durPRF}$ that runs $\mathcal{A}$ as a subroutine. Moreover the $\mathtt{Sim}$ is also given the intersection sizes s_1 and s_2. We write $\mathcal{O}$ to denote the

domain of the PRF, and write n_A and n_B to denote the database sizes of P_A and P_B, respectively.

Corrupted P_A with a honest P_B.

(1) Let (k_A^1, k_A^2, k_A^3) be the keys $\mathcal{A}$ sends to $\mathcal{F}_{durPRF}$.**Init**. Sim sends an AHE public key, and n_B AHE ciphertexts of zero to $\mathcal{A}$ on behalf of P_B.
(2) Let sid_b $(b = 1, 2, 3)$ be the session id $\mathcal{A}$ sends to $\mathcal{F}_{durPRF}$.**Recv**. Sim sends n_B random elements of O to $\mathcal{A}$ on behalf of $\mathcal{F}_{durPRF}$.
(3) Sim sends n_A random elements from O to $\mathcal{A}$ on behalf of P_B.
(4) Let $\mathsf{sid}_2, \mathsf{sid}_3$ be the session ids $\mathcal{A}$ sends to $\mathcal{F}_{durPRF}$.**Update**. Sim samples $n_B - s_1$ random elements (designated by R) from O to $\mathcal{A}$ on behalf of $\mathcal{F}_{durPRF}$.
(5) Sim samples $n_A - s_1$ random elements of O and randomly replaces s_2 of them by a subset of R. Then Sim sends them to $\mathcal{A}$ on behalf of P_B.

It is not hard to see that the simulation is computationally close to an execution of Π_{wfm} in the $\mathcal{F}_{durPRF}$-hybrid world. In particular, the simulation relies on the fact that messages sent by P_B are either AHE ciphertexts or PRF outputs, and the matching positions (e.g. J_B^1, J_B^2) are randomly shuffled.

The simulation for P_B's view is similar except that we additionally simulate an AHE of aggregation result in Step 18 where the privacy is guaranteed by the $\mathcal{HE}$.Refresh function. □

4.1.1 AHE Instantiation from (R)LWE. Learning-With-Errors (LWE)-based encryption schemes are usually quite computationally efficient compared to schemes like Paillier and DGK, since they do not involve expensive modular-exponentiation operations to encrypt and decrypt. However, the batch of LWE ciphertexts of the payloads can be large in volume. To further reduce the communication overhead, we can first transfer a batch of N (*e.g.*, $N = 4096$) payloads inside a Ring LWE ciphertext instead of a batch of LWE ciphertexts. This reduces the communication by a factor of $O(N)$. Then, the ciphertext receiver (*e.g.*, P_A in our case) can unpack the encrypted payloads from the received Ring LWE ciphertext, *i.e.*, one LWE ciphertext for each payload value, by performing some inexpensive local operations. The aggregation is then performed over the LWE ciphertexts.

We explicitly export the Ring LWE as an encapsulation layer of the LWE ciphertexts to be more efficient and to offer more flexibility. Particularly, for the efficiency, the sum step over LWE ciphertexts is about 2× faster than the corresponding homomorphic additions over the Ring LWE ciphertexts. For the flexibility, when comes to the secret sharing variant described in the following section, P_A can return a "packed" version of the matched LWE ciphertexts $\{C[j]\}_{j \in J_B^1 \cup J_B^2}$ to P_B using the technique from [39].

4.2 Variants

We considered variants of the waterfall matching protocol that modify the functionality and offer important flexibility.

4.2.1 Multiple Payloads. Our protocol generalizes easily to cases where each row in P_B's dataset has multiple types of payloads (*e.g.* amount spent, and label of conversion), and parties wish to learn the sum of each type of payload. This is handled by simply encrypting each column separately using the AHE, and creating one sum for each type of payload.

4.2.2 Secret Sharing the Matched Payloads. One variant is to let P_A reshare the matched HE payloads to P_B in an additively secret shared form, rather than computing the aggregated sum. This variant allows more complicated functions beyond the aggregated sum to be evaluated on the intersection set. To achieve this, P_A now sends $\mathcal{HE}.\mathsf{Refresh}\left(\mathcal{HE}.\mathsf{Enc}_{\mathsf{pk}}(C[j] + r_j)\right)$ a batch of AHE ciphertexts to P_B in Step 17. for $j \in J_B^1 \cup J_B^2$. The masking values $\{r_j\}$ are sampled uniformly from the message space of $\mathcal{HE}$. Then, the matched payloads $\{T[j]\}$ are now additively shared between the two parties, with P_B holding $T[j] + r_j$ and P_A holding $-r_j$, while P_B is unaware of the matching positions $j \in J_B^1 \cup J_B^2$.

4.2.3 Payloads from Both Parties. Some ad conversion computations involve payloads from both the publisher and the advertiser. Our protocol also generalizes easily to support payloads from both sides. Particularly, we make the following modifications over Figure 4 to support payloads from both sides:

(1) P_A now generates the AHE key-pair and sends the public key to P_B in Step 4.
(2) P_A also encrypts its payloads using its public key and attaches the AHE ciphertexts in Step 4.
(3) P_A now needs to send the tags rTag_B^2 to P_B in Step 15 in a shuffled order. This enables P_B to obtain the matching positions on the second identifier J_A^2. By this point, P_B is already able to pick up P_A's (encrypted) payloads in the intersection, *i.e.*, $J_A^1 \cup J_A^2$.

4.3 Concrete Constructions of $\mathcal{F}_{durPRF}$

We present two constructions of $\mathcal{F}_{durPRF}$ using Hashed Diffie-Hellman (HashDH) and Yao's Garbled circuit [11], respectively. Particularly, we leverage a functionality $\mathcal{F}_{GC}$ which takes inputs x and y from a circuit generator and a circuit evaluator, respectively, and returns the evaluation $BC(x, y)$ to the circuit evaluator for a Boolean circuit $BC : \{0,1\}^* \times \{0,1\}^* \mapsto \{0,1\}^*$ that is agreed upon by both the generator and the evaluator.

Specifically, in the HashDH-based construction, we leverage a PRF function defined as $F_k(x) = \mathsf{H}(x)^k$ where H is a function that maps bit strings to elements of $\mathbb{G}$.

Theorem 2. *If H is modeled as a random oracle, and DDH assumption holds for the group $\mathbb{G}$, the protocol $\Pi_{durPRF}^{\mathsf{DDH}}$ in Figure 5 securely realizes the functionality $\mathcal{F}_{durPRF}$ with the PRF defined as $F_k(x) = \mathsf{H}(x)^k$.*

We give a sketch proof here, and defer the formalized proof to Appendix.

Proof. (Sketch.) The proof follows the arguments in [40, 41]. w.l.o.g, we assume P_A acts as the Sender with input X and P_B acts as the Receiver who will invoke the **Update** command.

Correctness. For the evaluation part, P_B obtains $a_i = \mathsf{H}(x_i)_A^k$ which is then lifted to $a_i^{k_B}$ and forms $\mathsf{H}(x_i)^{k_A k_B \bmod q}$. For the update part,

$$\left((\mathsf{H}(x_j)^k)^{k'_B/k_B}\right)^{k'_A/k_A} = (\mathsf{H}(x_j)^{k'_B k_A})^{k'_A/k_A} = \mathsf{H}(x_j)^{k'_B k'_A},$$

Protocol $\Pi^{\mathsf{DDH}}_{durPRF}$ from HashDH

Initialization:
P_A samples $k_A, k'_A \xleftarrow{\$} \mathbb{Z}_q$. P_B samples $k_B, k'_B \xleftarrow{\$} \mathbb{Z}_q$.
Evaluation: [40] P_A inputs $X = \{x_i\}_{i=1}^m$.

(1) P_A: $a_i \leftarrow \mathsf{H}(x_i)^{k_A}$ for $x_i \in X$.
(2) $\mathsf{P}_A \rightarrow \mathsf{P}_B : \{a_i \in \mathbb{G}\}_i$.
(3) P_B outputs $Z = \{a_i^{k_B}\}$.

Update: [41] P_B selects a subset $Z' \subset Z$.

(4) $\mathsf{P}_B : b_j \in \mathbb{G} \leftarrow z_j^{k'_B/k_B}$ for $z_j \in Z'$.
(5) $\mathsf{P}_A \leftarrow \mathsf{P}_B : \{b_j\}_j$ in a shuffled order.
(6) $\mathsf{P}_A : b'_j = b_j^{k'_A/k_A}$.
(7) $\mathsf{P}_A \rightarrow \mathsf{P}_B : \widetilde{Z} = \{b'_j\}_j$.
(8) P_B outputs $\widetilde{Z}$ by undoing the shuffle in Step 2.

Figure 5: The *durPRF* protocol from HashDH. Both parties agree upon a group $\mathbb{G}$ of a prime order q that the DH assumption holds, and a hash function $\mathsf{H} : \{0,1\}^* \rightarrow \mathbb{G}$ that maps bit strings to elements of $\mathbb{G}$. The corresponding PRF is defined as $F_k(x) = \mathsf{H}(x)^k$.

which is the PRF evaluation $F_{k'}(x_j)$ under the new key $k' = k'_A \cdot k'_B$.
Sender's (P_A) Privacy. For the evaluation part, we claim that the views of the receiver (*i.e.*, P_B) - *i.e.*, $a_i = \mathsf{H}(x_i)^{k_A}$ for $i = 1, \cdots, m$, where H is modeled as a random oracle - is indistinguishable from $r_1, \cdots, r_m$ with $r_i \xleftarrow{\$} \mathbb{G}$. For the update part, P_B knows $\{b_j\}$ but P_B can not distinguish $\{(b_j)^{k'_A/k_A}\}_j$ from random elements from $\mathbb{G}$ under the DDH assumption and random oracle model.
Receiver's (P_B) Privacy. For the evaluation part, P_B's privacy is achieved directly from the CDH assumption. That is, P_A who knows a_i can not derive P_B's secret k_B after seeing $a_i^{k_B}$. For the update part, the views of the sender (P_A), *i.e.*, $(\mathsf{H}(x_j)^{k_A})^{k'_B}$, is indistinguishable from random group elements from $\mathbb{G}$. □

For the GC-based construction in Figure 6, we use the standard AES encryption $\mathsf{AES}_k(\mathsf{H}(x))$ as a PRF function, where H maps the input to an AES block. The circuit generator/evaluator paradigm of GC naturally allows an oblivious evaluation of the PRF in a reversed manner by having the PRF receiver act as the circuit evaluator. For the update part, we **do not** perform the key-switching from $\mathsf{AES}_k(x)$ to $\mathsf{AES}_{k'}(x)$, which might require a circuit twice the size of the evaluation part. Instead, we define the PRF after the update by double AES encryption $F_{k'}(x) = \mathsf{AES}_{k'}(\mathsf{AES}_k(\mathsf{H}'(x)))$, which also follows our interface by treating the inner evaluation $\mathsf{AES}_k(\mathsf{H}'(x))$ as a hash from $\{0,1\}^* \mapsto \{0,1\}^\lambda$, given that the inner AES encryption acts as a pseudorandom permutation.

Theorem 3. *If H' is modeled as a random oracle, the protocol $\Pi^{\mathsf{GC}}_{durPRF}$ in Figure 6 securely realizes the functionality $\mathcal{F}_{durPRF}$ under the $\mathcal{F}_{GC}$ hybrid. The PRF is defined as $F_k(x) = \mathsf{AES}_k(\mathsf{H}'(x))$.*

The correctness and security simply reduce to the correctness and security of $\mathcal{F}_{GC}$.

4.3.1 Complexities. Let $n = |X|$, $n' = |Z|$ and C be the size of the AES encryption circuit. In total $\Pi^{\mathsf{DDH}}_{durPRF}$ requires computing $2(n+n')$ exponentiations. $\Pi^{\mathsf{DDH}}_{durPRF}$ exchanges $O(n(|\mathbb{G}|+1))$ bits for the PRF evaluation, and $O(n'(|\mathbb{G}|+1))$ bits for the update step. On the other hand, $\Pi^{\mathsf{GC}}_{durPRF}$ exchanges $O(2nC\lambda + \lambda^2)$ bits for the PRF evaluation, and $O(n'C\lambda + \lambda^2)$ bits for the update step.

Protocol $\Pi^{\mathsf{GC}}_{durPRF}$ from $\mathcal{F}_{GC}$

Initialization:
P_A samples $k_A, k'_A \xleftarrow{\$} \{0,1\}^\lambda$. P_B samples $k_B, k'_B \xleftarrow{\$} \{0,1\}^\lambda$. Both parties aggree on a hash function $\mathsf{H}' : \{0,1\}^* \rightarrow \{0,1\}^\lambda$.
Evaluation: P_A inputs $X = \{x_i\}_{i=1}^m$.

(1) P_A: $a_i \leftarrow \mathsf{H}'(x_i)$ for $x_i \in X$.
(2) P_A and P_B invokes $\mathcal{F}_{GC}$ to evaluate the standard AES encryptions $\mathsf{AES}_k(\cdot)$. Particularly, P_A is the circuit generator with inputs $(k_A, \{a_i\})$, and P_B is the circuit evaluator with inputs k_B. After the GC execution, P_B obtains $Z = \{\mathsf{AES}_k(a_i)\}_{i=1}^m$ where the key is defined as $k = k_A \oplus k_B$.

Update: P_B inputs $Z' \subset Z$.
P_A and P_B invokes $\mathcal{F}_{GC}$ to evaluate the standard AES encryption. Particularly, P_A is the circuit generator with inputs k'_A, and P_B is the circuit evaluator with inputs (k'_B, Z'). After the GC execution, P_B obtains $\{\mathsf{AES}_{k'}(z_j)\}_{z_j \in Z'}$ with the key $k' = k'_A \oplus k'_B$.

Figure 6: The *durPRF* protocol from Garbled Circuit (GC). The corresponding PRF is defined as $F_k(x) = \mathsf{AES}_k(\mathsf{H}(x))$ where $\mathsf{H} : \{0,1\}^* \rightarrow \{0,1\}^\lambda$ that maps bit strings to an AES block.

5 DP MECHANISM FOR WATERFALL MATCHING

5.1 DP Mechanism Design

We first consider a mechanism to protect the intersection-size for the single identifier setting, and then we extend it to the waterfall matching setting. Randomness is introduced by padding dummy entries to both input sets A and B to create $\widetilde{A}$ and $\widetilde{B}$, respectively. This is done by randomly selecting and adding τ dummy entries from a common set D with 2τ rows, which is disjoint from both A and B. The specific steps are outlined in Algorithm 1.

The following Theorem 4 provides a simple way to enhance the privacy of any private matching protocol that leaks the intersection size. For a given target privacy profile (ϵ, δ), one can calculate the parameter τ from (1), which defines the size of the dummy set D.

Theorem 4. *Let Π be a semi-honest secure two-party protocol that takes private input sets A and B from P_A and P_B, respectively, and computes the intersection size, i.e. $|A \cap B|$. Also, let D be a common set of dummies of 2τ entries such that $D \cap A = \emptyset$ and $D \cap B = \emptyset$. Then the composited protocol $\Pi'(A, B) := \Pi(\mathcal{M}^\tau_{\mathsf{one}}(A, D), \mathcal{M}^\tau_{\mathsf{one}}(B, D))$ is a semi-honest secure two-party protocol offering (ϵ, δ)-differential privacy, where DP profile is given by*

$$\delta(\epsilon) = \frac{1}{2^\tau \mathsf{C}_\tau} \left(1 + \sum_{z=\left\lceil \frac{\tau \cdot e^{\epsilon/2}-1}{e^{\epsilon/2}+1} \right\rceil}^{\tau-1} ({}^\tau\mathsf{C}_z)^2 - e^\epsilon \cdot ({}^\tau\mathsf{C}_{z+1})^2 \right). \quad (1)$$

To prove Theorem 4, we utilize the following. We write $\mathcal{M}(A, B)$ to denote a variable of the intersection size between two (single-ID)

Algorithm 1: $\mathcal{M}^{\tau}_{\mathsf{one}}$ Mechanism for single ID

Input : Set S of n entries. Set D of $\tau' > \tau$ entries such that $D \cap S = \emptyset$.
Output: A new set $\tilde{S}$ of $\tilde{n} = n + \tau$ entries.

1: $D' \subset D$ by randomly selecting a subset of τ entries.
2: Output $\tilde{S} = S + D'$ by appending D' to S.

sets. Note that, up to this point, $\mathcal{M}(A, B)$ is independent of the specific private matching protocol.

Lemma 5. *The size of the intersection between two padded sets in Algorithm 1 can be represented as*

$$\mathcal{M}(A, B) = |\mathcal{M}^{\tau}_{\mathsf{one}}(A, D) \cap \mathcal{M}^{\tau}_{\mathsf{one}}(B, D)| = |A \cap B| + z, \quad (2)$$

where z with axisymmetrially Probability Mass Function (PMF):

$$\Pr(z = z') = \begin{cases} \frac{({}^{\tau}\mathrm{C}_{z'})^2}{{}^{2\tau}\mathrm{C}_{\tau}} & 0 < z' \leq \tau, \\ 0 & \textit{others.} \end{cases} \quad (3)$$

where ${}^{n}\mathrm{C}_{r}$ is the combinations out of a group of n.

In cases where we concentrate on one side, say A, the inclusion of B is omitted from $\mathcal{M}(A, B)$ for simplicity, *i.e.*, only $\mathcal{M}(A)$.

Proof. (Theorem 4) We introduce the privacy loss random variable [42] for a tight DP analysis.

Definition 5. *The privacy loss random variable of $\mathcal{M}(A)$ over $\mathcal{M}(A')$ for any pair of neighboring inputs $A \sim A'$ is defined as follows:*
(i) If both $\Pr(\mathcal{M}(A) = o) \neq 0$ and $\Pr(\mathcal{M}(A') = o) \neq 0$, then

$$\boldsymbol{\gamma}_{AA'} = \ln\left(\frac{\Pr(\mathcal{M}(A) = o)}{\Pr(\mathcal{M}(A') = o)}\right). \quad (4)$$

(ii) If $\Pr(\mathcal{M}(A') = o) = 0$ and $\Pr(\mathcal{M}(A') = o) \neq 0$, then $\boldsymbol{\gamma}_{AA'} = \infty$.

The quantity $\boldsymbol{\gamma}_{AA'}$ is the logarithmic ratio between the probability of observing outcome o on input A, B compared to input A', B, which is referred to as the privacy loss variable. From Definition 5, it is apparent that the privacy analysis involves both $\boldsymbol{\gamma}_{AA'}$ and $\boldsymbol{\gamma}_{A'A}$ by considering the order of A and A'. According to Definition 3, $\mathcal{M}$ assures tightly (ϵ, δ)-DP with δ computed as follows: $\delta(\epsilon) = \max(\delta_{AA'}(\epsilon), \delta_{A'A}(\epsilon))$ where

$$\delta_{AA'}(\epsilon) = \sum_{o \in O} \max\left(\Pr(\mathcal{M}(A) = o) - e^{\epsilon} \Pr(\mathcal{M}(A') = o), 0\right), \quad (5)$$

$$\delta_{A'A}(\epsilon) = \sum_{o \in O} \max\left(\Pr(\mathcal{M}(A') = o) - e^{\epsilon} \Pr(\mathcal{M}(A) = o), 0\right). \quad (6)$$

To streamline the analysis, we demonstrate the property that when employing a DP mechanism with axisymmetrially distributed additive noise such as that proposed in Algorithm 1, the order of neighboring data sets can be disregarded in the privacy analysis.

Property 1. *If and only if the PMF of $\mathcal{M}(A)$ is axisymmetrically distributed, as exemplified by $\Pr(z)$ in (3), the following relationship holds for any non-negative ϵ: $\delta_{A'A}(\epsilon) = \delta_{AA'}(\epsilon)$.*

Algorithm 2: $\mathcal{M}_{\mathtt{mult}}$ Mechanism for multiple IDs

Input : Sets $\{S_\ell\}_{\ell=1}^{m}$, $\{D_\ell\}_{\ell=1}^{m}$, and $\{\hat{D}_\ell\}_{\ell=1}^{m}$. S_ℓ is a set of n entries. $D_\ell, \hat{D}_\ell$ are sets of 2τ and $2\tau m$ entries, respectively.
Output: New sets $\{\tilde{S}_\ell\}_{\ell=1}^{m}$ of $n + m\tau$ entries each.

1: **for** $1 \leq k \leq m$ **do**
2: Initialize $\tilde{S}_k = S_k$.
3: **for** $1 \leq \ell < k$ **do**
4: Update the set $\tilde{S}_k \leftarrow \mathcal{M}^{\tau}_{\mathsf{one}}(\tilde{S}_k, \hat{D}_\ell)$
5: **end for**
6: Update the set $\tilde{S}_k \leftarrow \mathcal{M}^{\tau}_{\mathsf{one}}(\tilde{S}_k, D_k)$
7: **for** $k < \ell \leq m$ **do**
8: Update the set $\tilde{S}_k \leftarrow \mathcal{M}^{\tau}_{\mathsf{one}}(\tilde{S}_k, \hat{D}_\ell)$
9: **end for**
10: **end for**

Property 1 simplifies the DP analysis by allowing the privacy accountant with either (5) or (6). Therefore, simplify $\boldsymbol{\gamma}_{AA'}$ as $\boldsymbol{\gamma}$. Substituting (3) into (4), we have

$$\Pr_{\boldsymbol{\gamma}}(\boldsymbol{\gamma} = \ln({}^{\tau}\mathrm{C}_z / {}^{\tau}\mathrm{C}_{z+1})) = \frac{({}^{\tau}\mathrm{C}_z)^2}{{}^{2\tau}\mathrm{C}_{\tau}}, \quad \forall z \in [0, \ldots, \tau]. \quad (7)$$

Meanwhile, (5) can be reformulated as

$$\begin{aligned} \delta(\epsilon) &\geq \mathbb{E}_{\boldsymbol{\gamma}}[\max\{0, 1 - \exp(\epsilon - \gamma)\}] \\ &= \delta(+\infty) + \int_{\epsilon}^{\infty} (1 - \exp(\epsilon - \gamma)) \Pr_{\boldsymbol{\gamma}}(\gamma) d\gamma. \end{aligned} \quad (8)$$

In this context, $\delta(+\infty)$ addresses failures resulting from the support discrepancy between $\mathcal{M}(A)$ and $\mathcal{M}(A')$, where a single occurrence leads to infinite leakage, *i.e.*,

$$\delta(+\infty) = \Pr(\mathcal{M}(A) = |A \cap B|). \quad (9)$$

By further substituting (9) into the second term in the right-hand-side of (8), we acquire the intersection size's DP profile using Algorithm 1, as outlined in Theorem 4 below. □

5.2 DP Mechanism for the Multi-ID Matching

DP protection in multi-ID scenarios can be accomplished by applying the single-ID DP mechanism to each ID column. It is well-known that the composition of DP mechanisms compromises the privacy of DP protection. Suppose $\mathcal{M}^{\tau}_{\mathsf{one}}$ (*i.e.*, Algorithm 1) achieves $(\epsilon_\ell, \delta_\ell)$-DP for its intersection size for the ℓ-th ID column. The basic composition reveals that DP protection, considering the observation of each intersection size of each ID column from the two parties, is (ϵ, δ)-DP, where $\epsilon = \sum_{\ell=1}^{m} \epsilon_\ell$ and $\delta = \sum_{\ell=1}^{m} \delta_\ell$. Consequently, the protective capacity of DP diminishes linearly with an increase in the number of IDs.

The linear rate of diminishing DP is attributed to the fact that different IDs matched for each column could correspond to the same record. If there is a way to ensure that the cross-ID dummies cover disjoint subsets of the data records, the sequential composition is no longer a tight DP analysis. Instead, the combined privacy loss is the maximum of ϵ_ℓ for all $\ell \in [1, m]$ from the parallel composition theorem. To ensure that cross-ID dummies cover disjoint subsets of the dummy records, we propose Algorithm 2 to construct dummy variables. The sufficient condition for parallel composition is that

the dummies padded for different ID columns come from disjoint spaces and cannot match.

Property 2. *(DP profile for Waterfall Matching.) In Algorithm 2, if the mechanism $\mathcal{M}^{\tau}_{\mathsf{one}}$ provides achieving (ϵ, δ)-DP for each ID column, then the $\mathcal{M}_{\mathtt{mult}}$ mechanism also achieves (ϵ, δ)-DP for the waterfall matching.*

5.3 DP-enhanced Waterfall Matching Protocol

The DP-enhanced waterfall matching $\Pi_{dp\text{-}wmf}$ operates as follows.

(1) P_A and P_B compute the dummies size τ according to (1) under the DP profile (ϵ, δ).
(2) P_A and P_B agree on dummy sets $\mathsf{Dmy} = (D_1, \hat{D}_1, D_2, \hat{D}_2)$. Particularly, $|D_1| = |D_2| = 2\tau$ and $|\hat{D}_1| = |\hat{D}_2| = 4\tau$, and there are disjoint to each other, *i.e.*, $D_1 \cap D_2 = \emptyset$, $\hat{D}_1 \cap \hat{D}_2 = \emptyset$, $D_1 \cap \hat{D}_2 = \emptyset$, and $\hat{D}_1 \cap D_2 = \emptyset$.
(3) P_A samples $\{\widetilde{\mathsf{ID}}^b_A\}_b \leftarrow \mathcal{M}_{\mathtt{mult}}(\{\mathsf{ID}^b_A\}_b, \{D_b\}_b, \{\hat{D}_b\}_b)$ according to Algorithm 2.
(4) P_B samples $\{\widetilde{\mathsf{ID}}^b_B\}_b \leftarrow \mathcal{M}_{\mathtt{mult}}(\{\mathsf{ID}^b_B\}_b, \{D_b\}_b, \{\hat{D}_b\}_b)$ according to Algorithm 2.
(5) P_B appends $\widetilde{T} \leftarrow T \| 0$ with zeros to $|\widetilde{T}| = |\widetilde{\mathsf{ID}}^1_B|$.
(6) P_A and P_B jointly run Π_{wmf} on the modified inputs, *i.e.*, $(\widetilde{\mathsf{ID}}^1_A, \widetilde{\mathsf{ID}}^2_A)$ and $(\widetilde{\mathsf{ID}}^1_B, \widetilde{\mathsf{ID}}^2_B, \widetilde{T})$.

This DP-enhanced protocol $\Pi_{dp\text{-}wmf}$ is clearly a secure two-party protocol as long as the base protocol Π_{wmf} is secure. Note that P_A and P_B's views in $\Pi_{dp\text{-}wmf}$ differ from their views in the base protocol Π_{wmf} only with respect to the intersection size. Thus, the $\Pi_{dp\text{-}wmf}$ protocol ensures (ϵ, δ)-differential privacy (Definition 4) for the intersection sizes by a similar composition argument in Theorem 4.

Discussions. P_A and P_B can generate a common set of randomness by sharing a random seed. Moreover, to make sure the disjoint requirements (*i.e.*, $D \cap A = \emptyset$ and $D \cap B = \emptyset$), they can leverage the specific type of the identifier. For instance, suppose the target identifier is the "phone numbers", then P_A and P_B can generate the dummies from the alphabet set, or they can generate the dummies with longer digits than a valid phone number.

5.4 A Tighter Privacy Profile for Multiple Executions of Waterfall Matching

There are real needs to perform matching over the same ID sets multiple times. For instance:

- The ad publisher and the advertiser may perform the matching twice to aggregate ad conversion data across two ad campaigns launched in close proximity.
- The advertiser may want to change the payload and perform one more matching after having seen the aggregated value from the first matching.

From the privacy perspective, the notion of combining multiple DP intersection results while preserving the overall privacy guarantee is referred to as the sequential DP composition problem. One widely adopted analysis is the sequential composition, revealing a linear increasing rate of ϵ. Given the inversely proportional relationship between the dummy size τ and ϵ, it is natural to explore a tight DP composition that decreases τ the number of dummies

Algorithm 3: Find the minimal τ for multiple executions.

Input : Total DP budget (ϵ, δ); Total numbers of executions: k;
Output : The number of dummies added for DP: τ.

procedure Estimate-$\delta(\tau, k, \epsilon, n_x)$
  FFT window: $W = (4k - 2)\ln\tau$.
  FFT resolution: $\Delta_x = 2W/n_x$.
  Initialize PLD vector as an all zeros vector $P_\Gamma \in \mathbb{R}^{n_x}$.
  for $o \in [0, \ldots, \tau - 1]$: **do**
    Update: $P_\Gamma\left[\lceil (W + 2\ln{}^{\tau}\mathsf{C}_o - 2\ln{}^{\tau}\mathsf{C}_{o+1})/\Delta_x \rceil\right] = ({}^{\tau}\mathsf{C}_o)^2/{}^{2\tau}\mathsf{C}_\tau$
  end for
  Compute the convolutions via FFT: $b = D\mathcal{F}^{-1}\left(\mathcal{F}(D \cdot P_\Gamma)^k\right)$, where $D = \begin{bmatrix} 0 & I_{n_x/2} \\ I_{n_x/2} & 0 \end{bmatrix}$.
  Compute the starting point: $\gamma_\epsilon = \left\lceil \frac{\epsilon + W}{\Delta_x} \right\rceil$.
  Return $\delta' = 1 - \left(1 - \frac{1}{2\tau \mathsf{C}_\tau}\right)^k + \sum_{\gamma=\gamma_\epsilon}^{n_x} (1 - e^{\epsilon + W - \gamma\Delta_x}) b[\gamma]$.
end procedure

Set $\tau_{\mathrm{lo}} = 0$
Increase $\tau_{\mathrm{hi}} = 0, 1, \cdots$, until $\delta \le$ Estimate-$\delta(\tau_{\mathrm{hi}}, k, \epsilon, n_x)$.
while $\tau_{\mathrm{lo}} < \tau_{\mathrm{hi}}$ **do**
  Set $\tau' = \lceil (\tau_{\mathrm{lo}} + \tau_{\mathrm{hi}})/2 \rceil$.
  $\delta' \leftarrow$ Estimate-$\delta(\tau', m, \epsilon, W, n_x)$.
  Set $\tau_{\mathrm{lo}} \leftarrow \tau'$ if $\delta' \ge \delta$. Otherwise set $\tau_{\mathrm{hi}} \leftarrow \tau'$ if $\delta' < \delta$.
end while
Output τ_{hi}.

under the same privacy budget ϵ. We adapt the composition results shown in [12] to our multiple execution cases. Given $\Pr_{\boldsymbol{\gamma}}$ in (7), after k times of execution of our waterfall matching protocol, the composition is tightly (ϵ, δ)-DP for $\delta^m(\epsilon)$ given by

$$\delta^k(\epsilon) = 1 - (1 - \delta(+\infty))^k + \int_{\epsilon}^{\infty} \left(1 - \mathrm{e}^{\epsilon - \gamma}\right) \left(\Pr_{\boldsymbol{\gamma}} *^k \Pr_{\boldsymbol{\gamma}}\right)(\gamma) \mathrm{d}\gamma, \tag{10}$$

where $\delta^k(\epsilon)$ denotes the post-composition privacy profile, and $*^k$ denotes the k times convolution. According to (9), it is straightforward that,

$$\delta^k(\epsilon) = 1 - \left(1 - \frac{1}{2\tau \mathsf{C}_\tau}\right)^k + \sum_{\gamma: \epsilon < \gamma < \infty} (1 - e^{\epsilon - \gamma}) \Pr_{\boldsymbol{\gamma}}(\gamma)^{\circledast k}, \tag{11}$$

where $(\cdot)^{\circledast k}$ denotes the k-fold convolution. Note that directly solving $\Pr_{\boldsymbol{\gamma}}(\gamma)^{\circledast k}$ exhibits exponentially computation complexity w.r.t. k. Following similar steps as presented in [12], we present the two-party DP composition accounting algorithm based on FFT in Algorithm 3, subsequently, we outline steps to numerically derive τ that guarantees (ϵ, δ)-DP after k-fold composition in Algorithm 3.

FFT-based composition for privacy loss involves representing the distribution as a sparse vector, marking specific locations indicating data leakage and their probabilities. This vector is transformed into the frequency domain using FFT, followed by multiplication operations to handle convolution. Matrix expressions simplify these complex operations into linear transformations and power computations. The processed sequence is then reverted to the time domain using the inverse FFT transformation.

FFT significantly improves the computation cost from $O(n^2)$ to $O(n \log(n))$, where n denotes the length of $\Pr_{\boldsymbol{\gamma}}(\gamma)^{\circledast k}$. However, FFT requires the input sequence to be periodic. Further, the input

Table 1: The minimum dummies size τ is a function $\tau = \tau(n, m, \epsilon, \delta, k)$ of n the input size n, m the number of IDs, (ϵ, δ) the DP profile, and k the maximum sequential matching. The below are fixed by $m = 3$.

$k=1, \epsilon=1, \delta=1/(10n)$					$k=6, \epsilon=1, \delta=1/(10n)$				
n	10^4	10^5	10^6	10^7	n	10^4	10^5	10^6	10^7
τ	114	141	170	201	τ	285	353	425	503
$k=1, \epsilon=2, \delta=1/(10n)$					$k=6, \epsilon=2, \delta=1/(10n)$				
n	10^4	10^5	10^6	10^7	n	10^4	10^5	10^6	10^7
τ	77	96	116	137	τ	191	239	289	340

sequence should be cast to a periodical window with size, say W, no less than the length of FFT result to avoid information loss due to spectrum overlap. To this end, we proved the following property with details in Appendix, and it is used to set the parameter W required in our waterfall matching protocol.

Property 3. *(Minimal FFT window) The minimal window size for FFT to guarantee no spectrum overlap is* $W \geq (4K - 2) \ln \tau$.

Further, casting continuously-valued leakages to discrete vector locations incurs information loss. The discretization is handled by the resolution factor n_x that represents the total number of entries on the input sequence vector. Intuitively, larger n_x leads to high input resolution, and thereby reduces information loss. However, large n_x inevitably increases the computation complexity. By examining line 6 of the Estimate-δ procedure in Algorithm 3, this line projects the continuously valued leakage onto the closest discretized location to its right. In essence, this means that the leakage is consistently amplified or increased, thereby ensuring that the calculated δ' is always greater than or equal to the true δ that accurately reflects the exact leakage.

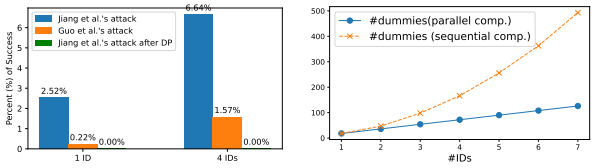
In summary, for a given target (ϵ, δ), one can calculate the parameter τ using Algorithm 3.

6 EXPERIMENTS

Testbed. All computational cost measurements for our protocols are in terms of total wall-clock runtime for both parties, running on cloud instances with an Intel(R) Xeon(R) Platinum 8336C CPU (2.30GHz) and 64GB of RAM. Communication cost includes the inbound and out-bound traffice of the two parties. We consider three network conditions, including LAN (10Gbps, 0.2ms ping), MAN (1Gbps, 2ms ping), and WAN (100Mbps, 20ms ping).

Synthetic Data. We consider balanced databases $|X| = |Y| = n$ for $10^3 \leq n \leq 10^7$ with the number of IDs $m = 2, 3$. The total intersection size is set as $|X \cap Y| = 0.02n$. The payloads are at most 32 bits long.

Implementations of *durPRF*. For the DDH-based construction, we use OpenSSL's implementation "prime256v1", a NIST elliptic curve with 256-bit group elements, as the group $\mathbb{G}$. For the random oracle, we use SHA-256 and apply the "try-and-increment" method to map bit-string to group elements [43]. We mention a common trick for the DDH-based construction. For the update command, we can truncate the updated group element (*i.e.*, b'_j in Step 4 in Figure 5) to save some communication. For instance, we only send



(a) Revealing the intersection sizes in the multi-ID setting are more susceptible to membership inference attacks.

(b) The parallel composition introduces less dummies than the sequential composition under the profile ($\epsilon = 1.0, \delta = 10^{-6}$).

Figure 7: (a) The robustness of the DP-enhanced protocol against the two membership inference attacks. (b) The parallel composition introduces less dummies.

the least significant γ bits of b'_j. This is suffice for the private matching function given that the the least significant bits of a random element of $\mathbb{G}$ are indistinguishable from a uniform bit-string [44]. Particularly, we set $\gamma = 96$. For the GC-based construction, we use the EMP-toolkit [45] and set the AES block size $\lambda = 128$. For the random oracle, we use the BLAKE2b hash function to produce 128-bit digests. For the AHE, we use the SEAL library [46] for the BFV scheme with the AVX512 acceleration [47]. Specifically, we use BFV with an 118-bit ciphertext modulus and a 64-bit plaintext modulus, with 8192 coefficients per ciphertext. Each ciphertext is 236 KB[2] and can hold up to 8192 payloads of 64-bit each.

6.1 Enhancing Privacy via DP

Table 1 shows the corresponding dummy sizes under different input settings according to Algorithm 3. Briefly, the dummy size is significantly smaller than the input size by virtue of the parallel decomposition and FFT technique applied. In Figure 7(a), we demonstrate the robustness of the DP-enhanced protocol against two membership inference attacks. Specifically, it illustrates that the membership leakage after $k = 20$ executions of a size-revealing private matching protocol can be up to 6% using recent attacks [8, 48]. However, after applying our DP protection for the intersection size, we do not observe any successful inferences using these attacks. Figure 7(b) depicts the effectiveness of the parallel decomposition for introducing less number of dummies. This also translates to the FFT-based composition technique used for multiple executions of the waterfall matching.

6.2 Micro-benchmarks

Table 2 depicts the micro-benchmarks for the two constructions of *durPRF*. Specifically, the DDH-based construction is more computation intensive but has significantly less communication overhead. On the other hand, the GC-based construction introduces a significantly larger communication overhead.

A Note on PSI Based on Symmetric Primitives It is often perceived that the DDH-based approach may be less efficient than conventional symmetric-primitive-based PSI protocols. However, existing experimental results challenge this misconception. For instance, Peter et al. reported a circuit-based PSI implementation

[2]We can leverage the symmetric version of BFV to reduce the size to 118KB.

Table 2: Benchmarks of *durPRF*. Single threaded. The time was measured under the LAN setting.

	Send Command			
	DDH		GC	
Input n	Time	Commu.	Time	Comm.
10^3	0.19s	32.23KB	0.52s	209.74MB
10^4	1.88s	322.26KB	5.21s	2.05GB
10^5	18.56s	3.15MB	51.23s	20.46GB
10^6	185.61s	31.52MB	512.49s	204.56GB

	Update Command			
	DDH		GC	
Input n	Time	Commu.	Time	Comm.
10^3	0.16s	43.95KB	0.52s	209.74MB
10^4	1.62s	439.45KB	5.21s	2.05GB
10^5	16.48s	4.29MB	51.23s	20.46GB
10^6	163.06s	41.92MB	512.49s	204.56GB

† 1GB = $2^1$0 MB = 2^{20} KB = 2^{30} bytes

Table 3: Benchmarks the proposed waterfall matching protocol using $\Pi^{\mathsf{DDH}}_{\boldsymbol{durPRF}}$. 8 threads were used for each party.

	Matching for Sum (#IDs $m = 2$)			
n	Commu.	LAN	MAN	WAN
10^4	2.91MB	1.89s	2.03s	3.51s
10^5	26.61MB	6.93s	7.22s	10.54s
10^6	263.1MB	68.01s	69.86s	88.18s
10^7	2631MB	697s	719s	911s
	Matching for Share (#IDs $m = 3$)			
n	Comm.	LAN	MAN	WAN
10^4	4.14MB	1.42s	1.48s	3.58s
10^5	41.36MB	11.29s	11.41s	15.44s
10^6	410.64MB	106.68s	109.71s	139.35s
10^7	4102MB	1095s	1121s	1430s
	Matching for Share (#IDs $m = 2$)			
n	Comm.	LAN	MAN	WAN
10^4	3.16MB	0.95s	1.18s	2.63s
10^5	26.85MB	6.93s	7.73s	10.98s
10^6	237.19MB	72.46s	74.17s	94.09s
10^7	2637MB	732s	742s	940s
	Matching for Share (#IDs $m = 3$)			
n	Comm.	LAN	MAN	WAN
10^4	4.64MB	2.36s	2.16s	3.76s
10^5	41.59MB	11.99s	12.34s	16.69s
10^6	411.36MB	110.02s	114.28s	145.79s
10^7	4109MB	1125s	1153s	1456s

(single identifier) requiring 8 CPU cores and 103 seconds (277 MB memory) for a dataset of $n = 10^6$ records [49, Table 4]. This demonstrates 2× higher CPU workloads and 3.7× greater communication overhead compared to our DDH-based method.

6.3 Benchmark the Private Waterfall Matching

Table 3 shows the performance of our waterfall matching protocol under different network conditions. From the results, we see that the Matching-for-Sum and the Matching-for-Share variant do not change much. That is because the *durPRF* part (*i.e.*, DDH-based *durPRF*) dominates the computation and communication. The AHE part only takes a very small portion of the whole protocol for two reasons. First, the BFV is very efficient for encrypting a long vector, *i.e.*, million records per second. Second, the intersection size is significantly smaller than the input size, leading a relatively smaller workload for the AHE, *e.g.*, addition and decryption. Overall, our waterfall matching protocol is practical to handle large-scale input sets.

6.4 Comparison with Existing Solutions

Figure 8 shows a comparison with two existing approaches. To the best of our knowledge, none of the existing approaches fully realize waterfall matching without cross-ID leakages. This is because most of them are designed for the single ID setting.

6.4.1 Private-Join-then-Sum (PJS) [2]. Ion *et al.* present a matching-for-sum protocol using hashed DDH and the Paillier AHE scheme. Their approach works for a single ID and reveals the exact intersection size. We re-ran their implementation[3] in our environment. Due to the superiority of the BFV scheme over the Paillier scheme, our approach is about 20× faster, with approximately one-tenth the communication overhead compared to their approach.

6.4.2 PS^{3}I [17], a matching-for-share protocol with payloads from both sides. We also perform the benchmark under the single-ID setting with two payload columns, and we compare the performance using their implementation[4]. In general, our protocol is about 2 order of magnitude faster than PS^{3}I with 85% less communication. This improvement is a result of multiple factors. PS^{3}I [17] utilizes the Paillier scheme which is less efficient than the BFV scheme we used. Also, PS^{3}I [17] transfer all encrypted payloads while our approach only transfer the encrypted payload within the intersection.

CONCLUSIONS

In this paper, we propose a practical and privacy-preserving solution to the multidimensional matching problem. By integrating a novel two-party distributed differential privacy mechanism with a specially tailored matching protocol, our approach effectively mitigates membership inference leakage. Our design satisfies the practical requirements for handling multiple identifiers while minimizing the overhead of additional dummy elements needed for privacy protection. This combination of robust privacy guarantees and computational efficiency represents a significant advancement in privacy-preserving data matching, offering both theoretical rigour and practical applicability for real-world deployment.

REFERENCES

[1] "Advanced Matching," https://developers.facebook.com/docs/meta-pixel/advanced/advanced-matching, 2025.

[3] https://github.com/google/private-join-and-compute
[4] https://github.com/facebookresearch/Private-ID

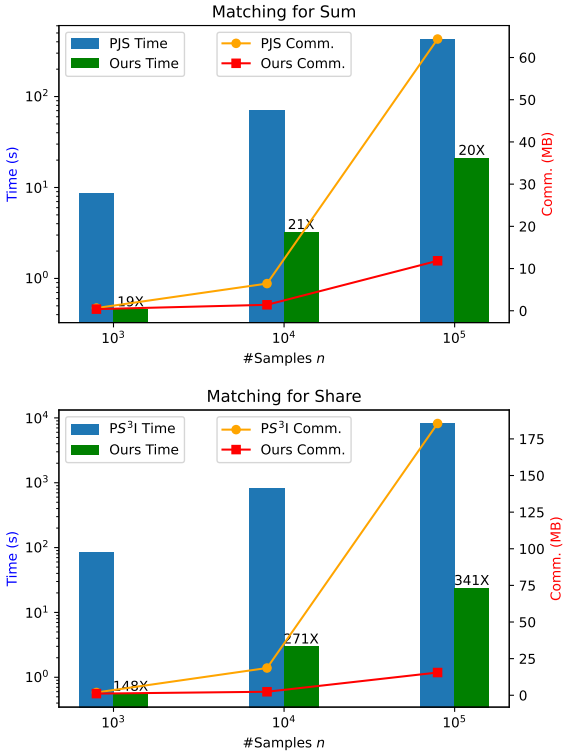


Figure 8: Comparison with the PJS [2] protocol and the PS^{3}I [17] protocol. For our DDH-based protocol, we configure the DP profile with the parameters $(k = 6, m = 3, \epsilon = 2, \delta = 1/(10n))$. The experiments were conducted under the LAN network setting, utilizing one single thread per party.

[2] M. Ion, B. Kreuter, A. E. Nergiz, S. Patel, S. Saxena, K. Seth, M. Raykova, D. Shanahan, and M. Yung, "On deploying secure computing: Private intersection-sum-with-cardinality," in *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2020, pp. 370–389.

[3] P. Buddhavarapu, A. Knox, P. Mohassel, S. Sengupta, E. Taubeneck, and V. Vlaskin, "Private matching for compute." *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 599, 2020.

[4] M. Ion, B. Kreuter, A. E. Nergiz, S. Patel, S. Saxena, K. Seth, M. Raykova, D. Shanahan, and M. Yung, "On deploying secure computing: Private intersection-sum-with-cardinality," in *IEEE European Symposium on Security and Privacy, EuroS&P*, 2020. [Online]. Available: https://doi.org/10.1109/EuroSP48549.2020.00031

[5] B. Pinkas, T. Schneider, O. Tkachenko, and A. Yanai, "Efficient circuit-based PSI with linear communication," in *Advances in Cryptology - EUROCRYPT 2019*, Y. Ishai and V. Rijmen, Eds., 2019.

[6] N. Chandran, D. Gupta, and A. Shah, "Circuit-psi with linear complexity via relaxed batch OPPRF," *Proc. Priv. Enhancing Technol.*, vol. 2022, no. 1, pp. 353–372, 2022. [Online]. Available: https://doi.org/10.2478/popets-2022-0018

[7] G. Garimella, P. Mohassel, M. Rosulek, S. Sadeghian, and J. Singh, "Private set operations from oblivious switching." in *Public Key Cryptography (2)*, 2021, pp. 591–617.

[8] X. Guo, Y. Han, Z. Liu, D. Wang, Y. Jia, and J. Li, "Birds of a feather flock together: How set bias helps to deanonymize you via revealed intersection sizes," in *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Aug. 2022. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/guo

[9] B. Jiang, J. Du, and Q. Yan, "Anonpsi: An anonymity assessment framework for psi," in *The Network and Distributed System Security Symposium (NDSS) 2024*, Boston, MA, Feb. 2024.

[10] P. Buddhavarapu, B. M. Case, L. Gore, A. Knox, P. Mohassel, S. Sengupta, E. Taubeneck, and M. Xue, "Multi-key private matching for compute," *Cryptology ePrint Archive*, 2021.

[11] A. C.-C. Yao, "How to generate and exchange secrets," in *27th annual symposium on foundations of computer science (Sfcs 1986)*. IEEE, 1986, pp. 162–167.

[12] A. Koskela, J. Jälkö, L. Prediger, and A. Honkela, "Tight approximate differential privacy for discrete-valued mechanisms using fft," *arXiv preprint arXiv:2006.07134*, 2020.

[13] H. Chen, K. Laine, and P. Rindal, "Fast private set intersection from homomorphic encryption," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1243–1255.

[14] B. Pinkas, T. Schneider, and M. Zohner, "Faster private set intersection based on OT extension," in *23rd USENIX Security Symposium (USENIX Security 14)*, 2014, pp. 797–812.

[15] V. Kolesnikov, R. Kumaresan, M. Rosulek, and N. Trieu, "Efficient batched oblivious prf with applications to private set intersection," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 818–829.

[16] M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *International conference on the theory and applications of cryptographic techniques*. Springer, 2004, pp. 1–19.

[17] P. Buddhavarapu, A. Knox, P. Mohassel, S. Sengupta, E. Taubeneck, and V. Vlaskin, "Private matching for compute," *Cryptology ePrint Archive*, 2020.

[18] F. Karakoç and A. Küpçü, "Linear complexity private set intersection for secure two-party protocols," in *International Conference on Cryptology and Network Security*. Springer, 2020, pp. 409–429.

[19] P. Branco, N. Döttling, and S. Pu, "Multiparty cardinality testing for threshold private intersection," in *PKC 2021*, ser. Lecture Notes in Computer Science, J. A. Garay, Ed., vol. 12711. Springer, 2021, pp. 32–60. [Online]. Available: https://doi.org/10.1007/978-3-030-75248-4_2

[20] S. Ghosh and M. Simkin, "The communication complexity of threshold private set intersection," in *CRYPTO 2019*, ser. Lecture Notes in Computer Science, vol. 11693. Springer, 2019, pp. 3–29. [Online]. Available: https://doi.org/10.1007/978-3-030-26951-7_1

[21] E. Zhang, J. Chang, and Y. Li, "Efficient threshold private set intersection," *IEEE Access*, vol. 9, pp. 6560–6570, 2021. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.3048743

[22] M. Ciampi and C. Orlandi, "Combining Private Set-Intersection with Secure Two-Party Computation," in *SCN*, 2018, pp. 464–482.

[23] E. D. Cristofaro, P. Gasti, and G. Tsudik, "Fast and Private Computation of Cardinality of Set Intersection and Union," in *CANS*, 2012, pp. 218–231.

[24] S. K. Debnath and R. Dutta, "Secure and Efficient Private Set Intersection Cardinality Using Bloom Filter," in *ISC*, 2015, pp. 209–226.

[25] Y. Huang, D. Evans, and J. Katz, "Private Set Intersection: Are Garbled Circuits Better than Custom Protocols?" in *NDSS*, 2012.

[26] B. Kacsmar, B. Khurram, N. Lukas, A. Norton, M. Shafieinejad, Z. Shang, Y. Baseri, M. Sepehri, S. Oya, and F. Kerschbaum, "Differentially private two-party set operations," in *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2020, pp. 390–404.

[27] M. Naor and O. Reingold, "Number-theoretic Constructions of Efficient Pseudo-random Functions," in *FOCS*, 1997, pp. 458–467.

[28] Y. Dodis and A. Yampolskiy, "A verifiable random function with short proofs and keys," in *PKC*, 2005, pp. 416–431.

[29] V. Kolesnikov, R. Kumaresan, M. Rosulek, and N. Trieu, "Efficient Batched Oblivious PRF with Applications to Private Set Intersection," in *CCS*, 2016, pp. 818–829.

[30] S. Casacuberta, J. Hesse, and A. Lehmann, "Sok: Oblivious pseudorandom functions," in *EuroS&P*, 2022, pp. 625–646.

[31] J. Katz and Y. Lindell, *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014.

[32] O. Goldreich, *The Foundations of Cryptography - Volume 2: Basic Applications*. Cambridge University Press, 2004. [Online]. Available: http://www.wisdom.weizmann.ac.il/%7Eoded/foc-vol2.html

[33] R. Canetti, "Security and Composition of Multiparty Cryptographic Protocols," *J. Cryptol.*, vol. 13, no. 1, pp. 143–202, 2000.

[34] A. McGregor, I. Mironov, T. Pitassi, O. Reingold, K. Talwar, and S. Vadhan, "The limits of two-party differential privacy," in *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. IEEE, 2010, pp. 81–90.

[35] N. Kumar, M. Rathee, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, "Cryptflow: Secure tensorflow inference," in *Symposium on Security and Privacy*, 2020, pp. 336–353.

[36] Z. Brakerski, "Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP," in *CRYPTO*, 2012, pp. 868–886.

[37] J. Fan and F. Vercauteren, "Somewhat Practical Fully Homomorphic Encryption," *IACR Cryptol. ePrint Arch.*, 2012.

[38] P. Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes," in *EUROCRYPT*, 1999, pp. 223–238.

[39] H. Chen, W. Dai, M. Kim, and Y. Song, "Efficient homomorphic conversion between (ring) LWE ciphertexts," in *ACNS*, 2021, pp. 460–479.

[40] S. Jarecki and X. Liu, "Fast secure computation of set intersection," in *SCN*, 2010, pp. 418–435.

[41] A. Everspaugh, R. Chatterjee, S. Scott, A. Juels, and T. Ristenpart, "The pythia PRF service," in *USENIX Security*, 2015, pp. 547–562.
[42] C. Dwork and G. N. Rothblum, "Concentrated differential privacy," *arXiv preprint arXiv:1603.01887*, 2016.
[43] D. Boneh and M. K. Franklin, "Identity-Based Encryption from the Weil Pairing," in *CRYPTO*, 2001, pp. 213–229.
[44] C. Chevalier, P. Fouque, D. Pointcheval, and S. Zimmer, "Optimal Randomness Extraction from a Diffie-Hellman Element," in *EUROCRYPT*, 2009, pp. 572–589.
[45] X. Wang, A. J. Malozemoff, and J. Katz, "EMP-toolkit: Efficient MultiParty computation toolkit," https://github.com/emp-toolkit, 2016.
[46] "Microsoft SEAL (release 4.1)," https://github.com/Microsoft/SEAL, Jan. 2023, microsoft Research, Redmond, WA.
[47] F. Boemer, S. Kim, G. Seifu, F. D. de Souza, V. Gopal *et al.*, "Intel HEXL (release 1.2)," https://github.com/intel/hexl, 2021.
[48] X. Jiang, X. Zhou, and J. Grossklags, "Comprehensive analysis of privacy leakage in vertical federated learning during prediction." *Proc. Priv. Enhancing Technol.*, vol. 2022, no. 2, pp. 263–281, 2022.
[49] P. Rindal and P. Schoppmann, "Vole-psi: Fast oprf and circuit-psi from vector-ole," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2021, pp. 901–930.

A APPENDIX

A.1 Proofs

A.1.1 Proof of Lemma 5.

Proof. We denote D'_A and D'_B the sampled set in Algorithm 1 by P_A and P_B, respectively. According to Algorithm 1, it is established that both D'_A and D'_B are independent of both A and B. Consequently, it becomes evident that $|\widetilde{A} \cap \widetilde{B}| = |A \cap B| + z$ where $z = |D'_A \cap D'_B|$. To calculate $\Pr(z = z')$, we need to determine the number of outcomes where the intersection size of $D'_A \cap D'_B$ is z and then divide by the total number of possible outcomes. There are ${}^{2\tau}\mathrm{C}_\tau$ possible outcomes for both D'_A and D'_B since there are random subsets of a dummy set with 2τ elements. Therefore, the total number of outcomes is ${}^{2\tau}\mathrm{C}_\tau \cdot {}^{2\tau}\mathrm{C}_\tau$.

The number of outcomes for D'_B alone is ${}^{2\tau}\mathrm{C}_\tau$. For any $|D'_A \cap D'_B| = z$, it implies that there are z elements in D'_B belonging to D'_A, with ${}^{\tau}\mathrm{C}_z$ possible outcomes. Simultaneously, the remaining $\tau - z$ elements in D'_B must be randomly selected from $D \setminus D'_A$, offering ${}^{\tau}\mathrm{C}_{\tau-z}$ possibilities. Consequently, we can compute the PMF of $z = |D'_A \cap D'_B|$ as follows:

$$\Pr(z = z') = \frac{{}^{2\tau}\mathrm{C}_\tau \cdot {}^{\tau}\mathrm{C}_{z'} \cdot {}^{\tau}\mathrm{C}_{\tau-z'}}{{}^{2\tau}\mathrm{C}_\tau \cdot {}^{2\tau}\mathrm{C}_\tau} = \frac{({}^{\tau}\mathrm{C}_{z'})^2}{{}^{2\tau}\mathrm{C}_\tau}.$$

□

A.1.2 Proof of Property 1.

Proof. Denote $\{0, \ldots, \tau\}$ as the support of z, the support of $\mathcal{M}(A)$ becomes: $\{|A \cap B|, ..., |A \cap B| + \tau\}$. Given the asymmetrically distributed property, we have $\Pr(z = \tau - o) = \Pr(z = o)$, $\forall o \; in [0, 1, \ldots, \tau]$. Given the sensitivity of the operation of $|A \cap B|$ is 1, The worst-case neighboring dataset $\mathcal{M}(A')$ has the support of $\{|A \cap B|+1, \ldots, |A \cap B|+\tau+1\}$, and the mapping relationships of the two neighboring datasets become $\Pr(\mathcal{M}(A) = o) = \Pr(\mathcal{M}(A') = o + 1)$. Define O as the support of the output of the mechanism, O^+ as the subset of O corresponds to the criterion:

$$O^+ = \{o : \Pr(\mathcal{M}(A) = o) - e^{\varepsilon} \Pr(\mathcal{M}(A') = o) \geq 0\}.$$

Simulator Sim_A

Evaluation: P_A does not receive message in this stage, and thus Sim_A does nothing.
Update:

- Sim_A samples $\tilde{b}_j \leftarrow \mathbb{G}$ for $j = 1, 2, \cdots, n'$.
- $\mathsf{P}_A \leftarrow \mathsf{Sim}_A : \{\tilde{b}_j\}_{j=1}^{n'}$ as P_A's received message in Step (5) in $\Pi^{\mathsf{DH}}_{durPRF}$.

Figure 9: The simulator for P_A in the $\Pi^{\mathsf{DH}}_{durPRF}$ protocol .

Then, according to the definition of DP, the failure probability δ can be expressed as:

$$\begin{aligned}\delta &= \sum_{o \in O} \max\left(\Pr(\mathcal{M}(A) = o) - e^{\varepsilon} \Pr(\mathcal{M}(A') = o), 0\right) \\ &= \sum_{o \in O^+} \left(\Pr(\mathcal{M}(A) = o) - e^{\varepsilon} \Pr(\mathcal{M}(A') = o\right) \\ &= \sum_{o \in O^+} \left(1 - e^{\varepsilon} \frac{\Pr(\mathcal{M}(A') = o)}{\Pr(\mathcal{M}(A) = o)}\right) \Pr(\mathcal{M}(A) = o),\end{aligned}$$

This shows how (8) is derived. For the first term in (8)

$$\begin{aligned}\delta_{AA'}(\infty) &= \sum_{\substack{\{o | \Pr(\mathcal{M}(A)=o)>0, \\ \Pr(\mathcal{M}(A')=o)=0\}}} \Pr(\mathcal{M}(A) = o) \\ &= \sum_{o=|A\cap B|}^{|A\cap B|} \Pr(\mathcal{M}(A) = o) \\ &= \Pr(\mathcal{M}(A) = |A \cap B|)\end{aligned} \tag{12}$$

For the other direction,

$$\begin{aligned}\delta_{A'A}(\infty) &= \sum_{\substack{\{o | \Pr(\mathcal{M}(A')=o)>0, \\ \Pr(\mathcal{M}(A)=o)=0\}}} \Pr(\mathcal{M}(A') = o) \\ &= \sum_{o=|A\cap B|+\tau}^{|A\cap B|+\tau} \Pr(\mathcal{M}(A) = o) \\ &= \Pr(\mathcal{M}(A) = |A \cap B| + \tau) \\ &= \Pr(\mathcal{M}(A) = |A \cap B|)\end{aligned} \tag{13}$$

which means $\delta_{AA'}(\infty) = \delta_{A'A}(\infty)$.

For the second part of (8), first, let $\boldsymbol{\gamma}_{AA'}(o)$ and $\boldsymbol{\gamma}_{A'A}(o)$ be:

$$\begin{aligned}\boldsymbol{\gamma}_{AA'}(o) &= \ln\left(\frac{\Pr(\mathcal{M}(A) = o)}{\Pr(\mathcal{M}(A') = o)}\right), \\ \boldsymbol{\gamma}_{A'A}(o) &= \ln\left(\frac{\Pr(\mathcal{M}(A') = o)}{\Pr(\mathcal{M}(A) = o)}\right)\end{aligned}$$

Then, $\boldsymbol{\gamma}_{AA'}(o) = -\boldsymbol{\gamma}_{A'A}(o)$, and by the symmetric property, $\boldsymbol{\gamma}_{AA'}(o) = \boldsymbol{\gamma}_{A'A}(\tau + 1 - o)$. Then, the second part $\delta(\boldsymbol{\gamma} > \epsilon)$ for each term can be expressed as:

$$\delta(\boldsymbol{\gamma}_{AA'} > \epsilon) = \sum_{o=1}^{\boldsymbol{\gamma}^{-1}_{AA'}(\epsilon)} (1 - e^{\epsilon - \boldsymbol{\gamma}_{AA'}(o)}) \Pr(\mathcal{M}(A) = o)$$

Simulator Sim_B

Evaluation:

- Sim_B samples $\tilde{a}_i \leftarrow \mathbb{G}$ for $i = 1, 2, \cdots, n$.
- $\mathsf{Sim}_B \rightarrow \mathsf{P}_B : \{\tilde{a}_i\}_{i=1}^{n}$ as P_B's received messages in Step (2).

Update:

- $\mathsf{Sim}_B \leftarrow \mathsf{P}_B : \{b_j \in \mathbb{G}\}_{j=1}^{n'}$ where b_js are chosen by P_B.
- $\mathsf{Sim}_B \rightarrow \mathsf{P}_B : \{b_j^r\}_{j=1}^{n'}$ using random $r \leftarrow \mathbb{Z}_q$ as P_B's received messages in Step (7).

Figure 10: The simulator for P_B in the $\Pi^{\mathsf{DH}}_{durPRF}$ protocol.

and

$$\begin{aligned}
&\delta(\boldsymbol{\gamma}_{A'A} > \epsilon) \\
&= \sum_{o=\boldsymbol{\gamma}^{-1}_{A'A}(\epsilon)}^{\tau} (1 - e^{\epsilon - \boldsymbol{\gamma}_{A'A}(o)}) \Pr(\mathcal{M}(A') = o) \\
&= \sum_{o=1}^{\boldsymbol{\gamma}^{-1}_{AA'}(\epsilon)} (1 - e^{\epsilon - \boldsymbol{\gamma}_{AA'}(o)}) \Pr(\mathcal{M}(A) = o).
\end{aligned}$$

Therefore, $\delta(\boldsymbol{\gamma}_{AA'} > \epsilon) = \delta(\boldsymbol{\gamma}_{A'A} > \epsilon)$. □

A.1.3 Proof of Property 3.

Proof. W is lower bounded by the maximum leakage after k-fold convolution. From (7), the maximum leakage of the mechanism can be expressed as

$$2 \ln \left(\max_{z \in [0, \tau-1]} \frac{z+1}{\tau - z} \right) = 2 \ln(\tau)$$

The length of the PLD is:

$$2 \ln \left(\max_{z \in [0, \tau-1]} \frac{z+1}{\tau - z} \right) - 2 \ln \left(\min_{z \in [0, \tau-1]} \frac{z+1}{\tau - z} \right) = 4 \ln \tau$$

Then, the maximum leakage after k-fold composition becomes:

$$2 \ln(\tau) + 4(k-1) \ln(\tau) = (4k - 2) \ln(\tau)$$

This completes the proof. □

A.1.4 Security Proof for $\Pi^{\mathsf{DDH}}_{durPRF}$. This section contains the full version of the security proof of $\Pi^{\mathsf{DDH}}_{durPRF}$.

Let $\mathsf{View}^{\Pi^{\mathsf{DDH}}_{durPRF}}_{\sigma,\lambda}(\{x_i\}_{i=1}^{n}, \emptyset)$ be a random variable representing the view of P_σ ($\sigma = A, B$) in real protocol execution, where the random variable ranges over the internal randomness of all parties, and the randomness in the setup phase (including that of the Random Oracle). The the view of a party consists of its internal state (including its input and randomness) and all messages this party received from the other party. The messages sent by this party do not need to be part of the view because they can be determined using the other elements of its view.

$$\begin{aligned}
\mathsf{View}^{\Pi^{\mathsf{DH}}_{durPRF}}_{A,\lambda}(\{x_i\}_{i=1}^{n}) &= (\emptyset, \{\mathsf{H}(x_j)^{k'_B}\}_{j \in Z}) \\
\mathsf{View}^{\Pi^{\mathsf{DH}}_{durPRF}}_{B,\lambda}(\{x_i\}_{i=1}^{n}) &= (\{\mathsf{H}(x_i)^{k_A}\}_{i=1}^{n}, \{\mathsf{H}(x_j)^{k'_A}\}_{j \in Z})
\end{aligned}$$

We now show that P_A's view can be simulated given only P_A's input but not P_B's input. The simulator Sim_A is given in Figure 9. We argue that $\{\mathsf{H}(x_j)^{k'_B}\}_{j \in Z}$ is indistinguishable from random group elements in $\mathbb{G}$ assuming the hardness of DDH under the random oracle model. Specifically, the existence of an efficient distinguisher $\mathcal{D}$ that outputs 0 when presented with $\tilde{b}_1, \cdots, \tilde{b}_{n'}$ and outputs 1 when it observes $\{\mathsf{H}(x_j)^{k'_B}\}_{j \in Z}$ allows us to construct a simulator Sim that violates the DDH assumption, as follows.

Upon receiving a DDH challenge (g, g^x, g^y, g^z), Sim does the following:

- Selects random values $d_1, \cdots, d_{n'-2}$ from $\mathbb{Z}_q$.
- Answers queries for H as follows: $\mathsf{H}(x_{j_1}) = g, \mathsf{H}(x_{j_2}) = g^x$, and $\mathsf{H}(x_{j_k}) = g^{d_{k-2}}$ for $k > 2$ for consistency.
- Sends $\{g^y, g^z, (g^y)d_1, \cdots, (g^y)^{d_{n'-2}}\}$ to $\mathcal{D}$.

If (g, g^x, g^y, g^z) is a Diffie-Hellman tuple, i.e., $z = xy$, then

$$\begin{aligned}
&\{g^y, g^z, (g^y)^{d_1}, \cdots, (g^y)^{d_{n'-2}}\} \\
=&\{g^y, (g^x)^y, (g^{d_1})^y, \cdots, (g^{d_{n'-2}})^y\},
\end{aligned}$$

which is distributed like $\{\mathsf{H}(x_{j_1})^{k'_B}, \cdots, \mathsf{H}(x_{j_{n'}})^{k'_B}\}$. Thus, $\mathcal{D}$ must return 1. On the other hand, when $z \neq xy$, the term $g^z \neq \mathsf{H}(x_{j_2})^y$. As a result, Sim can use $\mathcal{D}$'s output to respond to the DDH challenge correctly.

The simulator Sim_B is given in Figure 10. For the evaluation part, the values in $\{\mathsf{H}(x_i)^{k_A}\}$ are indistinguishable from random group elements under the DDH assumption. Similarly, for the update part, the values in $\{\mathsf{H}(x_j)^{k'_A}\}$ are also indistinguishable from random group elements in $\mathbb{G}$.

B FULL PROTOCOL WITH MORE IDENTIFIERS

Figure 12 depicts the full version of our waterfall matching protocol that handles $m \geq 2$ identifiers.

Complexities. Let the databases sizes being $n_A = |\mathsf{ID}^b_A|$ and $n_B = |\mathsf{ID}^b_B|$, and $|J^b|$ being the intersection size on the b-th identifier. When instantiating $\mathcal{F}_{durPRF}$ as the DDH-based construction, the computation of Figure 12 includes $O(2m(n_A + n_A) - 2\sum_{b=1}^{m-1} |J^b|)$ group operations, and $O(n_B + \sum_{b=1}^{m} |J^b|)$ AHE operations. The communication includes $O(m(n_A + n_B) + n_B - \sum_{b=1}^{m-1} |J^b|)$ group elements, and $O(n_B)$ AHE ciphertexts.

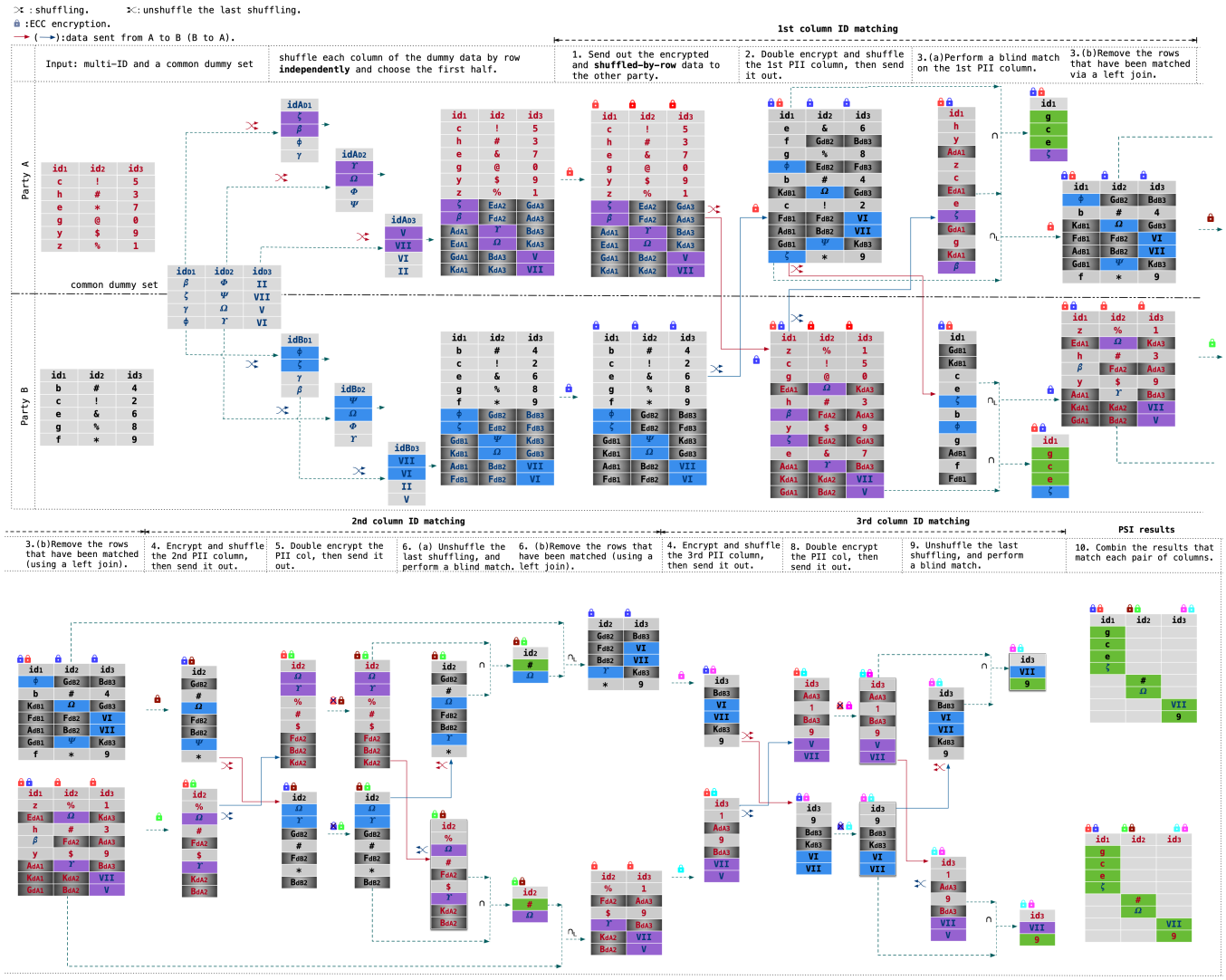


Figure 11: An illustration of the DDH-based Waterfall Matching.

Private Waterfall Matching and Sum Protocol Π_{wfm}

	P_A		P_B
1 :	P_A : identifiers $\mathsf{ID}_A^1, \mathsf{ID}_A^2, \cdots, \mathsf{ID}_A^m$		P_B : identifiers $\mathsf{ID}_B^1, \mathsf{ID}_B^2, \mathsf{ID}_B^m$, and payloads T
	Stage-1: Setups		
2 :	$k_A^1, k_A^b, \hat{k}_A^b \xleftarrow{\$} \mathbb{Z}_q$ for $b = 2, 3, \cdots, m$		$k_B^1, k_B^b, \hat{k}_B^b \xleftarrow{\$} \mathbb{Z}_q$ for $b = 2, 3, \cdots, m$
3 :	$\mathcal{F}_{durPRF}.\mathbf{init}(\mathsf{sid}_b, k_A^b)$ for $b = 1, \cdots, m$		$\mathcal{F}_{durPRF}.\mathbf{init}(\mathsf{sid}_b, k_B^b)$ for $b = 1, \cdots, m$
4 :	$\mathcal{F}_{durPRF}.\mathbf{init}(\hat{\mathsf{sid}}_b, \hat{k}_A^b)$ for $b = 2, \cdots, m$		$\mathcal{F}_{durPRF}.\mathbf{init}(\hat{\mathsf{sid}}_b, \hat{k}_B^b)$ for $b = 2, \cdots, m$
5 :			$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathcal{HE}.\mathsf{Gen}(1^\lambda)$
6 :		$\xleftarrow{\mathsf{pk}, C}$	$C = \{\mathcal{HE}.\mathsf{Enc}_{\mathsf{pk}}(T[j])\}_{j=1}^{\lvert T\rvert}$
	Stage-2: PRF on all identifiers		
7 :	$\mathcal{F}_{durPRF}.\mathbf{Send}(\mathsf{sid}_b, \mathsf{ID}_A^b)$ for $b = 1, \cdots, m$		$\mathsf{Tag}_A^b \leftarrow \mathcal{F}_{durPRF}.\mathbf{Recv}(\mathsf{sid}_b)$ for $b = 1, \cdots, m$
8 :	$\mathsf{Tag}_B^b \leftarrow \mathcal{F}_{durPRF}.\mathbf{Recv}(\mathsf{sid}_b)$ for $b = 1, \cdots, m$		$\mathcal{F}_{durPRF}.\mathbf{Send}(\mathsf{sid}_b, \mathsf{ID}_B^b)$ for $b = 1, \cdots, m$
	Stage-3: Matching on the first identifier		
9 :		$\xleftarrow{\widetilde{\mathsf{Tag}}_A^1 \text{ in shuffled order}}$	
10 :		$\xrightarrow{\widetilde{\mathsf{Tag}}_B^1 \text{ in shuffled order}}$	
11 :	Matches: $J_B^1 \leftarrow \{j : \mathsf{Tag}_B^1[j] \in \widetilde{\mathsf{Tag}}_A^1\}$		$J_A^1 \leftarrow \{j : \mathsf{Tag}_A^1[j] \in \widetilde{\mathsf{Tag}}_B^1\}$
12 :	Mismatches: $D_B^1 \leftarrow \{1, 2, \cdots, \lvert\mathsf{Tag}_B^1\rvert\} \setminus J_B^1$		$D_A^1 \leftarrow \{1, 2, \cdots, \lvert\mathsf{Tag}_A^1\rvert\} \setminus J_A^1$
13 :	Book-keep: $\sigma_B^1[j] \leftarrow D_B^1[j]$ for $1 \le j \le \lvert D_B^1\rvert$		$\sigma_A^1[j] \leftarrow D_A^1[j]$ for $1 \le j \le \lvert D_A^1\rvert$
	Stage-4: Key-update then matching on the b-th identifier ($b = 2, \cdots, m$)		
14 :	Keep the unmatched records $\mathsf{rTag}_B^b \leftarrow \{\mathsf{Tag}_B^b[i]\}_{i \in D_B^{b-1}}$		$\mathsf{rTag}_A^b \leftarrow \{\mathsf{Tag}_A^b[i]\}_{i \in D_A^{b-1}}$
15 :	(Skip this if $b = m$) $\mathsf{rTag}_B^b \leftarrow \mathcal{F}_{durPRF}.\mathbf{Update}(\mathsf{sid}_b, \hat{\mathsf{sid}}_b, \mathsf{rTag}_B^b)$		(Skip this if $b = m$) $\mathcal{F}_{durPRF}.\mathbf{Assit}(\mathsf{sid}_b, \hat{\mathsf{sid}}_b)$
16 :	$\mathcal{F}_{durPRF}.\mathbf{Assit}(\mathsf{sid}_b, \hat{\mathsf{sid}}_b)$		$\mathsf{rTag}_A^b \leftarrow \mathcal{F}_{durPRF}.\mathbf{Update}(\mathsf{sid}_b, \hat{\mathsf{sid}}_b, \mathsf{rTag}_A^b)$
17 :		$\xleftarrow{\widetilde{\mathsf{rTag}}_A^b \text{ in shuffled order}}$	
18 :	(Skip this line if $b = m$)	$\xrightarrow{\widetilde{\mathsf{rTag}}_B^b \text{ in shuffled order}}$	
19 :	Matches: $J_B^b \leftarrow \{\sigma_B^{b-1}[j] : \mathsf{rTag}_B^b[j] \in \widetilde{\mathsf{rTag}}_A^b\}$		$J_A^b \leftarrow \{\sigma_A^{b-1}[j] : \mathsf{rTag}_A^b[j] \in \widetilde{\mathsf{rTag}}_B^b\}$
20 :	Mismatches: $D_B^b \leftarrow D_B^{b-1} \setminus J_B^b$		$D_A^b \leftarrow D_A^{b-1} \setminus J_A^b$
21 :	Book-keep: $\sigma_B^b[j] \leftarrow D_B^b[j]$ for $1 \le j \le \lvert D_B^b\rvert$		$\sigma_A^b[j] \leftarrow D_A^b[j]$ for $1 \le j \le \lvert D_A^b\rvert$
	Final stage: aggregation		
22 :	Sum ct $\leftarrow \mathcal{HE}.\mathsf{Sum}(\{\widetilde{C}[j]\}_{j \in J_B^1 \cup \cdots \cup J_B^b})$	$\xrightarrow{\mathsf{ct}' \leftarrow \mathcal{HE}.\mathsf{Refresh}(\mathsf{ct})}$	Output $\mathcal{HE}.\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}')$.

Figure 12: (Full version) Our private waterfall matching protocol under $\mathcal{F}_{durPRF}$-hybrid model. $\mathcal{HE}$ is an AHE scheme.