S

Secure Computation of Differentially Private Mechanisms

Jonas Böhler
SAP Security Research, SAP SE, Karlsruhe, Germany

Synonyms

Distributed differential privacy; Secure multi–party computation of differentially private mechanisms

Definition

Secure multiparty computation and differential privacy compose nicely and provide orthogonal protections. The former cryptographically protects distributed *inputs* of a joint computation, only revealing the output, while the latter bounds what a randomized *output* reveals about any input.

Background

Secure multi-party computation (MPC) allows a set of parties $\mathscr{P} = \{P_1, \ldots, P_n\}$, where P_i holds private input d_i, to jointly compute a function f over their inputs while learning only the output $y = f(d_1, \ldots, d_n)$. Typically, MPC distinguishes *semi-honest* (passive) adversaries, which correctly compute the function but try to infer additional information from all received messages, and *malicious* (active) adversaries, which can deviate from the computation (e.g., alter messages). MPC protects the inputs during the computation of f but does not restrict what y reveals about the inputs.

In contrast, differential privacy (DP) restricts what the output y of a (randomized) mechanism $\mathscr{M}$ evaluated on a database $D = \{d_1, \ldots, d_n\}$ reveals about the underlying data. DP ensures that when D changes in a single element, creating a neighbor D', the effect on the output is bounded as $\Pr[\mathscr{M}(D) \in S] \leq \exp(\epsilon) \cdot \Pr\left[\mathscr{M}(D') \in S\right] + \delta$ where S is the set of all possible outputs of $\mathscr{M}$, parameter ϵ is a small constant, and $\delta \geq 0$ is typically negligible in the datasize (say $1/n^2$). Early DP literature assumed a trusted third party has centralized access to the raw data. Later work expanded it to a distributed setting (Kasiviswanathan et al. 2011) and considered computationally bounded parties (Mironov et al. 2009).

Theory

In the following MPC-based definition of DP, $\mathsf{View}^p_{\Pi}(D)$ denotes the view of party p during the execution of protocol Π on input D, including all exchanged messages and internal state. We write $\mathrm{negl}(x)$ to indicate that a value is negligible in

S. Jajodia et al. (eds.), *Encyclopedia of Cryptography, Security and Privacy*,
https://doi.org/10.1007/978-3-642-27739-9_1714-1

x. MPC security parameter κ controls the computational hardness for an adversary (e.g., length of an encryption key), and DP privacy parameter ϵ bounds the privacy loss, i.e., an adversary's maximum information gain after learning the output.

Definition 1 (Distributed Differential Privacy) A probabilistic protocol Π executed by n parties $\mathscr{P} = \{P_1, \ldots, P_n\}$ satisfies *Distributed Differential Privacy* w.r.t. a coalition $\mathscr{C} \subset \mathscr{P}$ if for any databases D, D' differing in a single record and any possible set S of views for protocol Π

$$\Pr\left[\mathsf{View}_{\Pi}^{\mathscr{C}}(D) \in S\right] \leq e^{\epsilon} \cdot \Pr\left[\mathsf{View}_{\Pi}^{\mathscr{C}}(D') \in S\right] + \delta(\kappa),$$

where $\delta(\kappa) = \delta + \mathrm{negl}(\kappa)$.

The definition considers adversaries which are computationally bounded in κ, i.e., Π provides computational security. For information-theoretic security, i.e., computationally unbounded adversaries, $\mathrm{negl}(\kappa)$ can be omitted (as it models, e.g., guessing an encryption key with negligible probability) (Mironov et al. 2009). The definition covers the entire view for all possible neighbors. Relaxations of the definition are possible, e.g., consider private record linkage where exact matches are released but the count of nonmatching records is differentially private (He et al. 2017).

Models

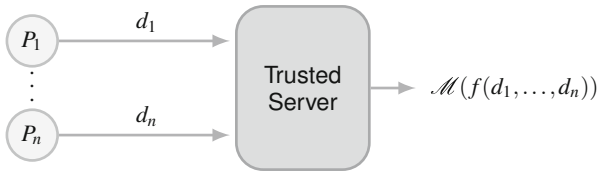
Implementation models for DP mechanisms are visualized in Fig. 1 (Böhler and Kerschbaum 2020). Secure computation of DP mechanisms provides an alternative to implementation models based on different trust assumptions.

In the *central model* (Dwork 2006; Dwork et al. 2006) every party sends their unprotected data to a trusted, central server which runs the mechanism $\mathscr{M}$ on the raw data values. The central model provides the highest accuracy as the randomization is only applied once. In the *local model* (Kasiviswanathan et al. 2011) parties locally apply the randomization themselves and send randomized values to an untrusted server for aggregation. The accuracy is limited as the randomization is applied multiple times; however, no trusted party is required. The intermediate *shuffle model* (Bittau et al. 2017; Cheu et al. 2019) places a trusted party, the shuffler, between the parties and server in the local model. The shuffler permutes and forwards the randomized values, ensuring anonymous communication, which reduces randomization requirements. The shuffle model comes in different "flavors" with different accuracy guarantees: a standard version, allowing either a single or multiple messages per party, and a crypto-augmented version using MPC primitives (e.g., secret sharing). To illustrate the separation between the models, consider counting queries with n parties: The central model achieves accuracy $O(1)$, the standard shuffle model $O(\log n)$, and the local model $O\left(\sqrt{n}\right)$ (Cheu et al. 2019).

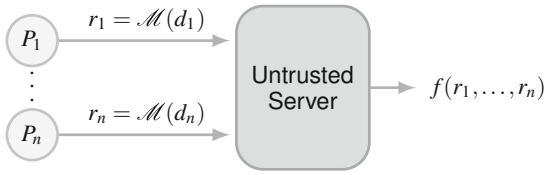
The *MPC model* combines the advantages of the respective models, i.e., high accuracy (as in the central model) without a trusted third party (as in the local model), closing the gap occupied by the shuffle model. The disadvantage is that general-purpose MPC incurs additional computation and communication overhead compared to the other models, which can be prohibitive for large data sets. However, MPC can be outsourced as shown in Fig. 1d: the n parties send their encrypted inputs, e.g., via secret sharing or homomorphic encryption, to m computation parties, who perform the computation on their behalf. In outsourced MPC trust is distributed among m non-colluding parties and not concentrated in a single point of failure as in the central model.

Distributed Noise Generation

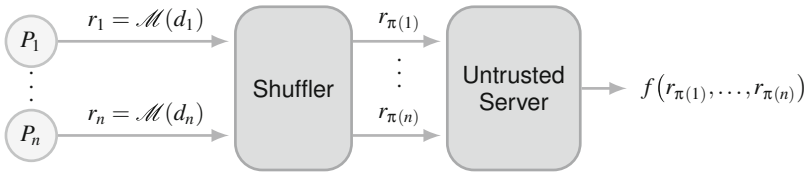
One standard way to satisfy DP for a function f is the Laplace mechanism where noise sampled from the Laplace distribution, denoted $\eta \sim \mathrm{Lap}(b)$, is added to a function evaluation, i.e., $f(D) + \eta$. Noise sampling via MPC is inefficient for semi-honest parties, as they have to securely evaluate complex functions (e.g., sampling $\mathrm{Lap}(b)$ is equivalent to computing $\mathbf{log}(r)$ with



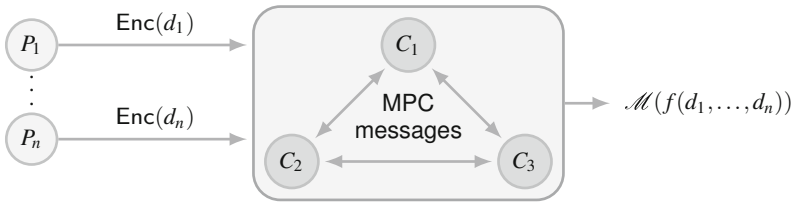
(a) Central Model



(b) Local Model



(c) Shuffle Model with Permutation π



(d) MPC Model with 3 Computation Parties C_i

Secure Computation of Differentially Private Mechanisms, Fig. 1 Implementation models for DP mechanism $\mathscr{M}$. Party P_i sends a message – raw d_i or randomized data r_i– to a server, who combines all messages with function f. For MPC, party P_i sends encrypted Enc(d_i) to m computation parties. (**a**) Central model. (**b**) Local model. (**c**) Shuffle model with permutation π. (**d**) MPC model with 3 computation parties C_i

S

uniformly random $r \in (0, 1]$). Distributed noise generation, where each party locally computes partial noises which are securely combined, is a more efficient and often found alternative (Goryczka and Xiong 2017). For example, Laplace noise with scale b is equivalent to the sum of gamma distributed random variables, $\sum_{j=1}^{n}(Y_j^1 - Y_j^2)$ for $Y_j^1, Y_j^2 \sim \text{Gamma}(\frac{1}{n}, b)$, where gamma distribution $\text{Gamma}(k, b)$ with shape k and scale b has density $\text{P}(x; k, b) = \frac{x^{k-1}}{\Gamma(k)b^k}\exp(-x/b)$. In general, samples from distributions that are infinitely divisible (e.g., Gauss, Laplace) can be expressed as a sum of independent and identically distributed random variables.

Applications

Google introduced the shuffle model to improve the accuracy of their DP telemetry data collection, as even billions of daily reports in the local model are insufficient for statistical values that are not heavy hitters (Bittau et al. 2017). A trusted

execution environment on Intel CPUs (SGX) performs the shuffling which can be replaced by MixNets, i.e., secure shufflers based on MPC.

Electricity suppliers deploy smart metering systems to learn aggregated statistics of a power grid with high accuracy, allowing user profiling (e.g., when users are at home, watch TV, turn on appliances). MPC with distributed noise generation protects the users and prevents such profiling (Goryczka and Xiong 2017).

MPC can hinder data poisoning attacks in the local model, where a set of parties provide faulty values (hidden by the noise) to skew a statistic. As a first step of a private statistical evaluation, MPC can ensure that values are plausible (e.g., in a fixed range) and detect such attacks (Cheu et al. 2021).

Computational efficiency of MPC can be improved by allowing controlled information leakage, e.g., revealing (partial) search patterns or record matches over private data. The information gain from such leakage can be bounded by differential privacy, e.g., by randomly adding dummy elements to obscure the true count of nonmatching records (He et al. 2017).

Open Problems and Future Directions

Further improvements of the efficiency of general-purpose MPC as well as designing specialized protocols, which only solve specific problems but more efficiently, are active research areas.

Cross-References

- Differential Privacy
- Homomorphic Encryption
- Multiparty Computation
- Secret Sharing Schemes

References

Bittau A, Erlingsson Ú, Maniatis P, Mironov I, Raghunathan A, Lie D, Rudominer M, Kode U, Tinnes J, Seefeld B (2017) Prochlo: strong privacy for analytics in the crowd. In: Proceedings of the Symposium on Operating Systems Principles, SOSP

Böhler J, Kerschbaum F (2020) Secure sublinear time differentially private median computation. In: Network and distributed systems security symposium (NDSS)

Cheu A, Smith A, Ullman J, Zeber D, Zhilyaev M (2019) Distributed differential privacy via shuffling. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT

Cheu A, Smith A, Ullman J (2021) Manipulation attacks in local differential privacy. In: IEEE Symposium on Security and Privacy, SP

Dwork C (2006) Differential privacy. In: International Colloquium on Automata, Languages, and Programming, ICALP

Dwork C, Kenthapadi K, McSherry F, Mironov I, Naor M (2006) Our data, ourselves: privacy via distributed noise generation. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT

Goryczka S, Xiong L (2017) A comprehensive comparison of multiparty secure additions with differential privacy. In: IEEE Transactions on Dependable and Secure Computing, TDSC

He X, Machanavajjhala A, Flynn C, Srivastava D (2017) Composing differential privacy and secure computation: a case study on scaling private record linkage. In: Proceedings of the Annual ACM Conference on Computer and Communications Security, CCS

Kasiviswanathan SP, Lee HK, Nissim K, Raskhodnikova S, Smith A (2011) What can we learn privately? SIAM J Comput 40:793–826

Mironov I, Pandey O, Reingold O, Vadhan S (2009) Computational differential privacy. In: Annual International Cryptology Conference, CRYPTO