# Private Data Aggregation on a Budget

**Morten Dahl**
Snips

**Valerio Pastro**
Yale University

**Mathieu Poumeyrol**
Snips

## Abstract

We propose a practical solution to performing simple cross-user machine learning on a sensitive dataset distributed among a set of users with privacy concerns.

We focus on a scenario in which a single company wishes to obtain the distribution of aggregate features, while ensuring a high level of privacy for the users. We are interested in the case where users own devices that are not necessarily powerful or online at all times, like smartphones or web browsers. This premise makes general solutions, such as general multiparty computation (MPC), less applicable.

We design an efficient special-purpose MPC protocol that outputs aggregate features to the company, while keeping online presence and computational complexity on the users' side at a minimum. This basic protocol is secure against a majority of corrupt users, as long as they do not collude with the company. If they do, we still guarantee security, as long as the fraction of corrupt users is lower than a certain, tweakable, parameter. We propose different enhancements of this solution: one guaranteeing some degree of active security, and one that additionally ensures differential privacy (DP). Finally, we report on the performance of our implementation of the above solutions.

## 1 Introduction

Many people make decisions based on recommendations: from trivial tasks like choosing restaurants, to more important ones such as choosing the right school or doctor. With the advent of the Internet, these recommendations are shifting from being a word of mouth to being delivered to users through their computers or phones, via data aggregation services that compile several users' experiences into easy-to-understand recommendations.

This change yields immediate and more accurate answers to the users, and is a profitable opportunity for the service-providing companies. Moreover, the techniques used to solve this type of problems are also extendable to many other settings such as surveys, analytics, and probabilistic models; in general, these applications fall in the area of machine learning and (automated) statistics.

However, this shift introduces new challenges that were not apparent before. By large, data aggregation is performed on the users' data after it is stored in the clear on the company's servers. This can be a risk for both the users and the company: the users may be concerned about how their data is used by the company beyond the intended service, and hence they may be reluctant to share relevant but sensitive data; at the same time, the company may worry about possible data breaches, which can compromise the users' trust in the company and damage its reputation and business, or even worse, have legal consequences.

Data aggregation, providing immense potential but at the same time posing great privacy issues, hence looks like a double-edged sword – this is not the case though, and while there exist general techniques to mitigate the downsides, we are here interested in a more specific practical question:

*Can we develop a real-world system that allows statistical computation over the users' private inputs, without revealing them?*

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

In this paper we answer this question by proposing an efficient technical solution for performing data aggregation in a way that minimises privacy-related issues for both the service-providing company and the users. We can think of a set of $N$ users, where each user $P_i$ has a private input $x_i$, and the company has a server $S$ and wants to obtain the (anonymized) distribution $D$ of the $x_i$ (so that $S$ can compute statistical functions or run some machine learning on it), and nothing more.

**General Setting** We assume that the set of users is dynamic (as in: changing over time) and not necessarily large, and that user devices have limited computational power and are sporadically available. As for the company, we may imagine a small company or a start-up launching a new digital assistant on smartphones, or any other app that offers data aggregation services.

This setting models many real-world scenarios, such as website users who only make a few visits, mobile users who download an app and uninstall it after a while, and sensors with limited lifespan. While this is our primary focus, our solution will work well even for less challenging scenarios, such as bigger companies that have the resources to partner with external parties[1] that are somehow trusted by the users – this makes matters easier because it allows us to reliably run a secure multi-party computation (MPC) protocol between the company and its partners to deliver (only) the result to the company; however, we seek *not* to rely on such partners, since a small company or a start-up developing a new service might not have the means to establish these partnerships. Specifically, we are in a scenario where we would like to use MPC, but there is only a single powerful party, the company, who is reliably online, and the rest are computationally weak and seldom online.

**Corruptions** Our aim is to protect against possible hackers who retrieve the company's data after the computation, and against company employees who have access to all the data stored at the company. However, we assume that the code run by honest players follows the protocol, even if this code is provided by the company. One justification for this assumption might be that the company judges it irrational to modify the source code, perhaps because the risk of honest employees or reverse engineers raising a flag would be too high. We formalise this by considering active security. Also, we want both the honest users' inputs and the output to be kept private from any set of corrupt parties (not colluding with the company).

These are the vital privacy and security guarantees that we require; jumping ahead, our protocol satisfies even stronger ones: namely, we maintain security also if the server gets corrupted together with any set of other players, as long as among these players there is only a small number of special users, called *clerks*, that in some sense simulate a partner (as described above) for the company.

### 1.1 Related work

**Multi-party computation (MPC)** General MPC solutions [Yao82, GMW87, BGW88, CCD88], even the most recent and practical ones [DPSZ12, DKL$^+$13, ZRE15, KOS16], are not particularly well-suited for our setting, because they usually require several players to have a (powerful) computer whose public identity is known by everyone and which is online essentially at all times during the computation. These systems, however, may be used to compute arbitrary functions and in some cases offer stronger security guarantees with respect to corruption.

**Server-Aided MPC** Our setting carries similarities to server-aided MPC [FKN94, DI05, DIK$^+$08, BCD$^+$09, KMR11], where the players are split into input providers and a few servers who receive the inputs and carry out the computation between them. In our setting, though, there is only one server, and the input providers are weak and seldom online. We argue that this scenario is closer to the real world for many companies, where users can join and leave the computation at any point, and are only willing to spend a small amount of resources, while the company, with substantial benefits from offering the service, has a strong incentive to invest in computational power and online presence.

**(Somewhat) Homomorphic Encryption** Homomorphic encryption schemes [BV11, BGV12, FV12] are good candidates for our task: users encrypt their inputs under the same public encryption key, send these encryptions to the company who, not knowing the decryption key, homomorphically

[1] For example, governments or independent institutions such as the Electronic Frontier Foundation.

aggregates all values into a single encryption.[2] This is then sent to a *decryptor* that has the private decryption key and can deliver the output. This puts large requirements on the decryptor, both in protecting the key and ensuring availability. Splitting the decryption key [DJ01, RN10] between a group of decryptors is one remedy, but can have the downside of a complex key generation protocol, with expensive interaction or orchestration between the decryptors [HMRT12, LTV12, HLP11]. Our protocol follows a similar approach, but requires almost no interaction and no full-time online presence.

**Differential Privacy** An alternative to the cryptographic solutions is for the users to add a small amount of "noise" to their inputs before sending them unencrypted to the server [EPK14]. This approach guarantees privacy yet requires no third party to hold the decryption key. Moreover, it is extremely efficient both computationally and in terms of data transmission, and allows the company to mix and re-use the noisy inputs for different computations without having to re-run any input protocols. However, because the noise accumulates in the output, large amounts of data is needed before the signal overpowers the noise, making this more appropriate for "Internet scale" data sets and less attractive for e.g. smaller companies and start-ups.

## 2 Secure aggregation

Our protocol for aggregating the input vectors into a sum is divided into two phases: in the *input phase*, each user processes his input and sends an encryption of this to the server, as well as shares the decryption key among the clerks; in the *computation phase*, the server combines the encrypted inputs to form an encryption (via a joint key) of the aggregation, and the clerks locally combine their key shares and send them to the server, so that the latter can retrieve the joint key that decrypts (only) the aggregation. Notice that the users are only required to participate in one round (the input phase). This makes it easy to achieve active security, if we assume an honest bulletin board or point-to-point channels. Moreover, by using standard public key techniques we may accumulate the decryption keys on a public server, thereby eliminating the need online presence for the clerks in the input phase.

The encryption scheme for our concrete applications is a simple one-time pad, which satisfies the properties needed and is very lightweight. Moreover, it ensures that the clerks can combine the decryption keys in a short session and without interacting with each other. To gain flexibility in the choice of clerks, we use a secret sharing scheme [Sha79] to split each decryption key into shares sent to the clerks. This not only allows a fraction of the clerks to be offline at decryption time, but also raises the number of clerks that must collaborate with the server to collectively break privacy.

More specifically, our basic aggregation protocol operates as follows: to share input vector $x_i$, user $P_i$ samples a uniform one-time pad $p_i$, sends $x_i + p_i$ to the server, and uses a $(n, t)$-linear secret sharing [CDM00] to split $p_i$ into $n$ shares that are finally distributed to the $n$ clerks. Here, $t$, the privacy threshold, is a public parameter chosen together with $n$, the number of clerks. In the computation phase, the clerks simply sum their individual shares and send this to the server, allowing it to recover $p = \sum p_i$. Subtracting $p$ from $\sum(x_i + p_i)$ yields $x = \sum x_i$ as desired, and nothing else.

**Theorem 2.1** (informal)**.** *The above protocol is secure against an adversary that corrupts any set of participants, as long as* either*: 1) the server is* not *corrupt; or 2) the server is corrupt together with* at most $t$ *clerks. Security, here, is in the universally composable model [Can01].*

Notice that, as previously mentioned, not all clerks are required to be online for the computation phase: specifically, only $r$ of them are required to be online, where $r$ is the reconstruction threshold of the secret sharing scheme used (for plain Shamir secret sharing, $r = t + 1$).

By replacing the secret sharing with a packed/ramp one, we may better distribute the load across clerks. This can be achieved by using a variant of Shamir secret sharing: instead of sampling a polynomial of degree $t$ per input, we can sample a polynomial of degree $t + k - 1$ that packs $k$ values together, without increasing the number or size of the shares. This yields a reconstruction threshold of $t + k$, and forces $n \geq t + k$. While to achieve the same privacy the number of clerks increases *additively* by $k$, the number of shares sent to each clerk is decreased *multiplicatively* by $k$.

[2] Notice that the degree of homomorphicity can be lowered to additive, such as Paillier [Pai99] or LWE-based [Reg05, LP11], if the aggregation is a linear function.

## 3 Differentially private aggregation

While the secure aggregation protocol ensures privacy in a cryptographical sense, it doesn't account for information leakage from the final aggregation. Following by-now standard practice, we may employ (global) *differential privacy* [DR14] as a way of quantifying and mitigating leakage. In this framework, uncertainty is used to reason about the privacy loss endured by any single individual, with our specific approach being to add randomness drawn according to certain distributions to the output. However, for privacy it is crucial that this randomness, or *noise*, remains unknown.

It is well-known that adding noise drawn according to a Gaussian distribution can be used to ensure differential privacy, and the divisibility of this distribution makes it particular well-suited for our distributed setting: each clerk simply draws a sample, secret shares this with the other clerks, and append these noise shares to the existing set of input shares. This exchange requires a synchronisation between the clerks, but may be run in parallel with the input providers delivering their inputs.

**Theorem 3.1** (informal)**.** *The above protocol for data aggregation ensures differential privacy additionally to security against an adversary like in Theorem 2.1.*

Adding noise from a Laplace distribution in a distributed setting is less straight-forward. However, since it may be characterised by the squared sum of four Gaussian distributions, we may use a multiplicative property of the secret sharing scheme to achieve this for a lower privacy threshold.

## 4 Applications and experiments

The ability to sum vectors has several applications, and besides the cases of analytics and surveys, we may also generate probability distributions for Bayesian graphical models. In both cases below, we assume a minimum privacy threshold of $\frac{n}{5}$ and a maximum reconstruction threshold of $\frac{4n}{5}$.

### 4.1 Drug-use survey

To test the applicability of our protocol we consider a real-world study [BF15] on the drug use among different age groups; quoting, the aim of the study is *"to see what drugs baby boomers are taking now, whether their patterns of use different from other age groups, and how similar people within the baby-boomer cohort are when it comes to drug use"*. Given the sensitive nature of these questions, surveys for building such data sets seem likely to benefit from strong privacy guarantees.

We ask under which parameter settings our protocol would have been efficient enough to carry out the data collection process for the more than $55,000$ users. We focus on clerks running on mobile phones or in web browsers and estimate the amount of data each of them must download.

To match the study we consider 17 age groups and 13 drugs that users may or may not have used. To manage the dimensionality, we essentially run 13 aggregations in parallel on input vectors of dimension $17 \cdot 2$. For a setting with 27 clerks, the estimated download size of each is less than $15\text{MB}$, and for a setting with 81 clerks this drops to less than $5\text{MB}$.

### 4.2 Next-place recommendation

Using their built-in sensors, mobile phones may autonomously compile local data sets allowing them to make predictions regarding their user. For example, a history of the user's previous whereabouts may be used to recommend relavant places of interest in the current context. Concretely, by maintaining the corresponding frequency table we can use likelihood function

$$\Pr\left[\mathsf{Category}, \mathsf{Popularity}, \mathsf{Home}, \mathsf{Work} \mid \mathsf{Weather}, \mathsf{Time}, \mathsf{Day}\right]$$

to suggest the most likely next place(s) given a set of nearby places (as e.g. provided through a third-party service such as Foursquare).

However, to share patterns discovered in the behaviour of several users, and to provide new users with a good model straight away, we must combine such frequency tables with sensitive data from several users. Using a setting with 728 clerks, we can combine tables of cardinality $20,160$ from $10,000$ users by asking each clerk to download less than $3\text{MB}$.

## References


[BCD+09] Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas P. Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael I. Schwartzbach, and Tomas Toft. Secure multiparty computation goes live. In Roger Dingledine and Philippe Golle, editors, *Financial Cryptography*, volume 5628 of *Lecture Notes in Computer Science*, pages 325–343. Springer, 2009.

[BF15] Anna Maria Barry-Jester and Andrew Flowers. How baby boomers get high, 2015. `http://fivethirtyeight.com/datalab/how-baby-boomers-get-high/`.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 309–325, New York, NY, USA, 2012. ACM.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 1–10, 1988.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. In *Proceedings of the 2011 IEEE 52Nd Annual Symposium on Foundations of Computer Science*, FOCS '11, pages 97–106, Washington, DC, USA, 2011. IEEE Computer Society.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 136–145. IEEE Computer Society, 2001.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 11–19, 1988.

[CDM00] Ronald Cramer, Ivan Damgård, and Ueli M. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In Bart Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, volume 1807 of *Lecture Notes in Computer Science*, pages 316–334. Springer, 2000.

[DI05] Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 378–394. Springer, 2005.

[DIK+08] Ivan Damgård, Yuval Ishai, Mikkel Krøigaard, Jesper Buus Nielsen, and Adam D. Smith. Scalable multiparty computation with nearly optimal work and resilience. In David Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 241–261. Springer, 2008.

[DJ01] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography: Public Key Cryptography*, PKC '01, pages 119–136, London, UK, UK, 2001. Springer-Verlag.

[DKL+13] Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure mpc for dishonest majority - or: Breaking the spdz limits. In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *ESORICS*, volume 8134 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2013.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662. Springer, 2012.

[DR14] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[EPK14] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 1054–1067, New York, NY, USA, 2014. ACM.

[FKN94] Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation (extended abstract). In Frank Thomson Leighton and Michael T. Goodrich, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 554–563. ACM, 1994.

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. `http://eprint.iacr.org/2012/144`.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In Alfred V. Aho, editor, *STOC*, pages 218–229. ACM, 1987.

[HLP11] Shai Halevi, Yehuda Lindell, and Benny Pinkas. Secure computation on the web: Computing without simultaneous interaction. In *Proceedings of the 31st Annual Conference on Advances in Cryptology*, CRYPTO'11, pages 132–150, Berlin, Heidelberg, 2011. Springer-Verlag.

[HMRT12] Carmit Hazay, Gert Læssøe Mikkelsen, Tal Rabin, and Tomas Toft. Efficient rsa key generation and threshold paillier in the two-party setting. In *Proceedings of the 12th Conference on Topics in Cryptology*, CT-RSA'12, pages 313–331, Berlin, Heidelberg, 2012. Springer-Verlag.

[KMR11] Seny Kamara, Payman Mohassel, and Mariana Raykova. Outsourcing multi-party computation. *IACR Cryptology ePrint Archive*, 2011:272, 2011.

[KOS16] Marcel Keller, Emmanuela Orsini, and Peter Scholl. MASCOT: faster malicious arithmetic secure computation with oblivious transfer. *IACR Cryptology ePrint Archive*, 2016:505, 2016.

[LP11] Richard Lindner and Chris Peikert. *Better Key Sizes (and Attacks) for LWE-Based Encryption*, pages 319–339. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[LTV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 1219–1234, New York, NY, USA, 2012. ACM.

[Pai99] Pascal Paillier. *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*, pages 223–238. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93. ACM, 2005.

[RN10] Vibhor Rastogi and Suman Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 735–746, New York, NY, USA, 2010. ACM.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *FOCS*, pages 160–164. IEEE Computer Society, 1982.

[ZRE15] Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 220–250. Springer, 2015.