Differentially Private Aggregation of Distributed Time-series with Transformation and Encryption

Part 3 — Differentially Private Aggregation

presenter: Le Chen

Nanyang Technological University

lechen0213@gmail.com

October 20, 2013

Outline

Reference

Cynthia Dwork.
Differential Privacy.
Invited talk at ICALP, Venice, Italy, July 10-14, 2006.
Automata, Languages and Programming, Lecture Notes in Computer Science Volume 4052, 2006, pp 1-12.

Wikipedia.
<http://en.wikipedia.org/wiki/Differential_privacy>

Elaine Shi, T-H. Hubert Chan, Eleanor Rieffel, Richard Chow and Dawn Song.
Privacy-Preserving Aggregation of Time-Series Data.
In Network and Distributed System Security Symposium (NDSS), 2011.

Reference

T-H. Hubert Chan, Elaine Shi, and Dawn Song
Privacy-Preserving Stream Aggregation with Fault Tolerance.
16th International Conference, FC 2012, Kralendijk, Bonaire, Februray 27-March 2, 2012. Financial Cryptography and Data Security, Lecture Notes in Computer Science Volume 7397, 2012, pp 200-214.

Vibhor Rastogi, Suman Nath
Differentially Private Aggregation of Distributed Time-series with Transformation and Encryption.
ACM SIGMOD 2010, New York, NY. Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, pages 735-746.

Motivation of Aggregation

model.jpg

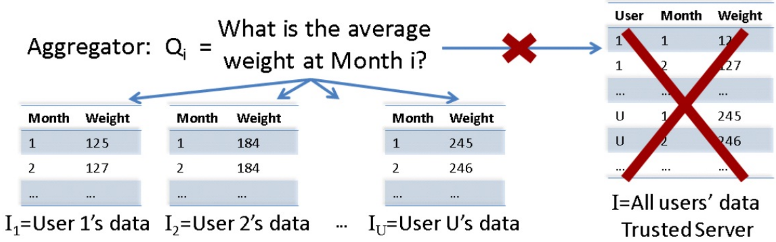

Figure 1: System Model (Users with data $I_1, \ldots, I_U$ Aggregator issues recurring query $\mathbf{Q} = \mathbf{Q}_1, \ldots, \mathbf{Q}_n$ No trusted server has $I = I_1 \cup I_2 \ldots \cup I_U$ to evaluate $\mathbf{Q}(I)$)

System Model

- $I = I_1 \cup I_2 \cdots \cup I_U$
- $nbrs(I)$: the data obtained from adding/removing one user's data from I.
- $Q = \{Q_1, Q_2, \cdots Q_n\}$
- $Q(I) = \{Q_1(I), Q_2(I), \cdots Q_n(I)\}$

Differential Privacy

- For all I, and $I' \in nbrs(I)$

$$Pr[A(I)] = x \leq e^{\epsilon} Pr[A(I') = x]$$

- Sensitivity: $\Delta(Q) = \max |Q(I) - Q(I')|$
- Laplace noise: $LAP(\lambda)$

Laplace Perturbation Algorithm

- Laplace Perturbation Algorithm (LPA):
  $LPA(Q, \lambda)$ is ϵ-differentially private for $\lambda = \Delta(Q)/\epsilon$
- Error: $error(LPA) = \Delta(Q)/\epsilon$

Distributed LPA

- Let x_u be the value of user u, the aggregate-sum query $Q(I) = \sum_{u=1}^{U} x_u$.
- Perturb: each user u adds a share of noise, n_u, to his private value x_u.
- To keep the estimation error small, the noise shares are chosen such that $\sum_{u=1}^{U} n_u$ is sufficient for differential privacy, but n_u alone is not sufficient: thus the value $x_u + n_u$ can not directly be sent to the aggregator.

Basic Distributed Protocol

protocol.jpg

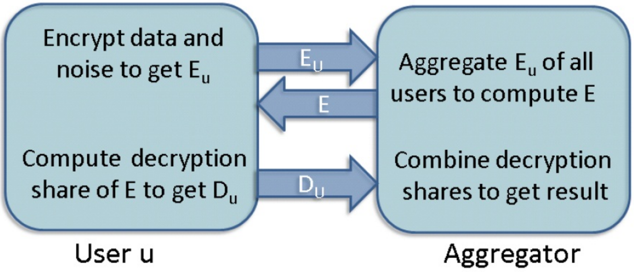

Figure 2: Basic Distributed Protocol (homomorphic property exploited to aggregate users' encryption & threshold property to combine users' decryption shares)

Challenges

- The noise shares have to be generated in a way so that their sum is sufficient for differential privacy.
- The aggregator can be malicious: the aggregator can cheat and request the decryption of wrong values, for instance, the encrypted private value of a single user, in which case the users will be inadvertently decrypting the private value of that user.

Basics: Encryption Scheme

Paillier Encryption

- Parameters: private key λ, public key N, g, g^{λ}.
- Encryption: $c = g^t r^N$
- Decryption: let $L(u) = (u-1)/N$, $Dec(c) = \frac{L(c^{\lambda} \mod N^2)}{L(g^{\lambda} \mod N^2)}$.

Basics: Encryption Scheme

- Distributed decryption: Suppose the private key λ is shared by U users as $\lambda = \sum_u \lambda_u$ where λ_u is the private key for user u.
- Each user u computes his decryption share $c_u = c^{\lambda_u}$.
- The decryption shares are combined as $c' = \prod_{u=1}^{U} c_u$.
- Finally the decryption $t = \frac{L(c' \mod N^2)}{L(g^\lambda \mod N^2)}$ is computed.

Protocol for Computing Exact Sum

- Encrypt-Sum(x_u, r_u): each user u encrypts his private value, x_u, added to a randomly generated r_u. Note that r_u is known only to user u.
- The aggregator obtains all the encryptions and multiples them to compute c. Due to the homomorphic properties of the encryption, the obtained c is an encryption of $\sum_{u=1}^{U}(x_u + r_u) = Q + \sum_{u=1}^{U} r_u$.
- Modified distributed decryption: Decrypt-Sum(c, r_u).

Protocol for Computing Exact Sum

Decrypt-Sum(c, r_u)

- The aggregator sends c to each user u for decryption.
- User u computes decryption share $c'_u = c^{\lambda_u} g^{-r_u \lambda}$.
- The aggregator collects c'_u from each user, combines them to get $c' = \prod_{u=1}^{U} c'_u$, and computes the final decryption $Q = \frac{L(c' \mod N^2)}{L(g^{\lambda} \mod N^2)}$.
- Except for $\sum_{u=1}^{U} x_u$, no other linear combinations can be computed.

Protocol for Computing Noisy Sum

- Remember that LPA requires us to compute $\tilde{Q} = Q + Lap(\lambda)$
- Let $Y_i \sim N(0, \lambda)$ for $i \in \{1, 2, 3, 4\}$ be four Gaussian random variables. Then $Z = Y_1^2 + Y_2^2 - Y_3^2 - Y_4^2$ is a $Lap(2\lambda^2)$ random variable.

Encrypt Noisy Sum

noisy sum.jpg

Algorithm 5.4 Encrypt-Noisy-Sum(x_u, r_u)

1: User u chooses five random numbers $r_u^1, r_u^2, \ldots, r_u^5$ from $\mathbb{Z}_m$ and computes $r_u = r_u^1 + r_u^2 - r_u^3 - r_u^4 + r_u^5$.
2: User u generates four $N(0, \sqrt{2\lambda}/U)$ random variables $y_u^1, \ldots, y_u^4$.
3: Let c^j =Encrypt-Sum-Squared(y_u^j, r_u^j) for $j \in \{1, 2, 3, 4\}$.
4: Let c^5 =Encrypt-Sum(x_u, r_u^5)
5: Aggregator computes $c = \frac{c^1 c^2 c^5}{c^3 c^4}$.

Encrypt Sum Squared

sum squared.jpg

Algorithm 5.3 Encrypt-Sum-Squared(y_u, r_u) Protocol

1: User u computes $c_u = Enc(y_u + a_u + b_u)$ and sends it to the aggregator.
2: The aggregator computes $c = \prod_{u=1}^{U} c_u$ and sends it to each user u.
3: Each user u generates a random $r_u \in \mathbb{Z}_m$, computes $c_u = c^{y_u - a_u + b_u} Enc(r_u)$.
4: The aggregator collects c_u from each user and computes $c' = (\prod_{u=1}^{U} c_u) Enc(a^2)$

where $a = \sum_u a_u$, $Enc(a^2)$ is computed and made public(How?), and $\sum_u b_u = 0$.

Theorem

THEOREM 5.2 (PRIVACY). *Let c =Encrypt-Noisy-Sum(x_u,r_u) and $\tilde{Q}$ =decrypt-sum(c, r_u). If there are at least $U/2$ honest users, then $\tilde{Q} = Q + Lap(\lambda)$+Extra-Noise, where $Lap(\lambda)$ is the noise generated by honest users and the Extra-Noise is that generated by malicious users. Thus for $\lambda = \Delta(Q)/\epsilon$, ϵ-differential privacy is guaranteed independent of what the malicious users and aggregator choose to do.*

THEOREM 5.3 (UTILITY). *Let c =Encrypt-Noisy-Sum(x_u,r_u) and $\tilde{Q}$ =decrypt-sum(c, r_u). If there are no malicious users, then $\tilde{Q} = Q + Lap(2\lambda)$. Finally, in presence of l malicious users that are all liars and no breakers, $\tilde{Q}$ can deviate from $Q + Lap(2\lambda)$ by at most $l \times \Delta(Q)$.*

Conclusion & Discussion

- We introduced an aggregation protocol supports distributed differential privacy and distributed decryption.
- The distributed algorithms need interaction.
- In the decryption of exact sum Decrypt-Sum, can we change r_u to $r_u - n_u$ such that a noise is left in the sum?
- In the extension part, the paper indicates that it can support 'fault tolerant' with threshold decryption. However, will it cause privacy problems?