Private Sampling with Malicious Samplers

César Sabater

INRIA - Lille

October 27, 2021

Work supervised by Jan Ramon and improved by discussions with Andreas Peter (Univ. of Twente, Netherlands).

Outline

Introduction and Problem

Existent tools

Our solutions

Outline

Introduction and Problem

Existent tools

Our solutions

Context

- Privacy Preserving Machine Learning
- Many parties with sensitive data
- No trusted party to share this data

Context

- Privacy Preserving Machine Learning
- Many parties with sensitive data
- No trusted party to share this data

Current Solutions

- Mostly imply decentralized computations (e.g. PP Federated Learning [Kairouz et al., 2019])
- Data storage and computations are locally private (e.g. held in devices)

Context

- Privacy Preserving Machine Learning
- Many parties with sensitive data
- No trusted party to share this data

Current Solutions

- Mostly imply decentralized computations (e.g. PP Federated Learning [Kairouz et al., 2019])
- Data storage and computations are locally private (e.g. held in devices)

No control over the correctness of computations

An Example

- Set of $P_1, \ldots, P_s$ of store owners
- Each P_i has a private dataset D_i

An Example

- Set of $P_1, \ldots, P_s$ of store owners
- Each P_i has a private dataset D_i

Run a PP decentralized $\mathcal{A}(D_1, \ldots, D_s) \rightarrow \mathcal{M}$ to learn customer preferences

An Example

- Set of $P_1, \ldots, P_s$ of store owners
- Each P_i has a private dataset D_i

Run a PP decentralized $\mathcal{A}(D_1, \ldots, D_s) \rightarrow \mathcal{M}$ to learn customer preferences

A malicious P_j can poison $\mathcal{M}$ to

- decrease customers of other stores
- increase its own profit

Similar settings

Decentralized systems with untrusted participants

- financial systems [Ben Sasson et al., 2014]
- digital contracts contracts [Bünz et al., 2020]
- Provide privacy while proving consistency of payments

Similar settings

Decentralized systems with untrusted participants

- financial systems [Ben Sasson et al., 2014]
- digital contracts contracts [Bünz et al., 2020]
- Provide privacy while proving consistency of payments

Commit and Prove

- Publish encrypted inputs (e.g. in a blockchain)
- Prove correctness over computations

Similar settings

Decentralized systems with untrusted participants

- financial systems [Ben Sasson et al., 2014]
- digital contracts contracts [Bünz et al., 2020]
- Provide privacy while proving consistency of payments

Commit and Prove

- Publish encrypted inputs (e.g. in a blockchain)
- Prove correctness over computations

In our ML setting

- Input remains private, but consistent
- If a party lies, it has to lie repeatedly
- This also holds in non-private ML: not possible to ensure truth on the input

Challenges

ML Computations

- Their domain is $\mathbb{R}$
- Involve transcendental functions (e.g. e^x, $\ln(x)$, ... for activation filters)
- Sample numbers from Gaussian, Laplacian distributions (e.g. for Differential Privacy [Dwork, 2006])

Challenges

ML Computations

- Their domain is $\mathbb{R}$
- Involve transcendental functions (e.g. e^x, $\ln(x)$, ... for activation filters)
- Sample numbers from Gaussian, Laplacian distributions (e.g. for Differential Privacy [Dwork, 2006])

The domain of proving frameworks is $\mathbb{Z}$ (or $\mathbb{Z}_p$, for some big prime p)

Challenges

ML Computations

- Their domain is $\mathbb{R}$
- Involve transcendental functions (e.g. e^x, $\ln(x)$, ... for activation filters)
- Sample numbers from Gaussian, Laplacian distributions (e.g. for Differential Privacy [Dwork, 2006])

The domain of proving frameworks is $\mathbb{Z}$ (or $\mathbb{Z}_p$, for some big prime p)

Challenges

ML Computations

- Their domain is $\mathbb{R}$
- Involve transcendental functions (e.g. e^x, $\ln(x)$, ... for activation filters)
- Sample numbers from Gaussian, Laplacian distributions (e.g. for Differential Privacy [Dwork, 2006])

The domain of proving frameworks is $\mathbb{Z}$ (or $\mathbb{Z}_p$, for some big prime p)

Our contribution

We focus on sampling: prove that a private value x is sampled from a distribution $\mathcal{D}$.
- But we also contribute in transcendental computations.

Problem Statement

Let

- s malicious parties $P_1, \ldots, P_s$ that can tamper with the protocol.
- a well known distribution $\mathcal{D}$.

For some $i \in \{1, \ldots, s\}$, sample $x \in \mathbb{R}$ such that

1. $x \sim \mathcal{D}$
2. x is private to P_i

if at least one party is honest.

Problem Statement

Let

- s malicious parties $P_1, \ldots, P_s$ that can tamper with the protocol.
- a well known distribution $\mathcal{D}$.

For some $i \in \{1, \ldots, s\}$, sample $x \in \mathbb{R}$ such that

1. $x \sim \mathcal{D}$
2. x is private to P_i

if at least one party is honest.

Example 2 Differentially Private Federated Learning

for t = 1 **to** T **do**
  At each party P_i**:** sample $\eta \sim \mathcal{D}$, compute $\Theta_u^t \leftarrow \textsc{LocalUpdate}(\Theta^{t-1}, \Theta_u^{t-1}) + \eta$
  Compute $\Theta^t \leftarrow \frac{1}{n} \sum_u \hat{\Theta}_u^t$
end for

Outline

Private Proofs

Commitments [Blum, 1983]

Commit to a value in $\mathbb{Z}_p$ while keeping it hidden

- Binding: the value cannot be changed once committed
- Similar to an encrypted value, but not neccesarily decryptable

Private Proofs

Commitments [Blum, 1983]

Commit to a value in $\mathbb{Z}_p$ while keeping it hidden

- Binding: the value cannot be changed once committed
- Similar to an encrypted value, but not neccesarily decryptable

Zero Knowledge Proofs [Cramer, 1997, Attema and Cramer, 2020]

- $x_1, \ldots, x_n$ committed values and $C : \mathbb{Z}_p^m \to \mathbb{Z}_p^k$ circuit (only modular $+$ and $\times$)
  $\implies$ can prove $C(x_1, \ldots, x_n) = \bar{0}$

Private Proofs

Commitments [Blum, 1983]

Commit to a value in $\mathbb{Z}_p$ while keeping it hidden

- Binding: the value cannot be changed once committed
- Similar to an encrypted value, but not neccesarily decryptable

Zero Knowledge Proofs [Cramer, 1997, Attema and Cramer, 2020]

- $x_1, \ldots, x_n$ committed values and $C : \mathbb{Z}_p^m \to \mathbb{Z}_p^k$ circuit (only modular $+$ and $\times$)
  $\implies$ can prove $C(x_1, \ldots, x_n) = \bar{0}$
- can prove S_1 and $S_2 \implies$ can prove $S_1 \vee S_2$

Private Proofs

Commitments [Blum, 1983]

Commit to a value in $\mathbb{Z}_p$ while keeping it hidden

- Binding: the value cannot be changed once committed
- Similar to an encrypted value, but not neccesarily decryptable

Zero Knowledge Proofs [Cramer, 1997, Attema and Cramer, 2020]

- $x_1, \ldots, x_n$ committed values and $C : \mathbb{Z}_p^m \to \mathbb{Z}_p^k$ circuit (only modular $+$ and $\times$)
  $\implies$ can prove $C(x_1, \ldots, x_n) = \bar{0}$
- can prove S_1 and S_2 $\implies$ can prove $S_1 \vee S_2$
- Cost: $O(C)$ and $O(cost(S_1) + cost(S_2))$ for proving/verifying

Private Proofs

Commitments [Blum, 1983]

Commit to a value in $\mathbb{Z}_p$ while keeping it hidden

- Binding: the value cannot be changed once committed
- Similar to an encrypted value, but not neccesarily decryptable

Zero Knowledge Proofs [Cramer, 1997, Attema and Cramer, 2020]

- $x_1, \ldots, x_n$ committed values and $C : \mathbb{Z}_p^m \to \mathbb{Z}_p^k$ circuit (only modular $+$ and $\times$)
  $\implies$ can prove $C(x_1, \ldots, x_n) = \bar{0}$
- can prove S_1 and $S_2 \implies$ can prove $S_1 \vee S_2$
- Cost: $O(C)$ and $O(cost(S_1) + cost(S_2))$ for proving/verifying

This is not FHE: We are just proving relations, not computing over encryptions

Private Proofs: Examples

- $x = y$: $x - y = 0$

Private Proofs: Examples

- $x = y$: $x - y = 0$
- b is a bit: $b(1 - b) = 0$

Private Proofs: Examples

- $x = y$: $x - y = 0$
- b is a bit: $b(1 - b) = 0$
- $x \in [0, 2^n - 1]$: $x - \sum_{i=1}^{n} 2^{i-1} b_i = 0$ for $b_1, \ldots, b_n$ bits

Private Proofs: Examples

- $x = y$: $x - y = 0$
- b is a bit: $b(1 - b) = 0$
- $x \in [0, 2^n - 1]$: $x - \sum_{i=1}^{n} 2^{i-1} b_i = 0$ for $b_1, \ldots, b_n$ bits
- Any polynomial relation in $\mathbb{Z}_p$

Private Proofs: Examples

- $x = y$: $x - y = 0$
- b is a bit: $b(1 - b) = 0$
- $x \in [0, 2^n - 1]$: $x - \sum_{i=1}^{n} 2^{i-1} b_i = 0$ for $b_1, \ldots, b_n$ bits
- Any polynomial relation in $\mathbb{Z}_p$
- If $x = A$ then S_1 else if $x = B$ then S_2:
  $(x - A = 0 \wedge S_1) \vee (x - B = 0 \wedge S_2)$
- ...

Sampling in the clear

Generate $u_1, u_2, \ldots$ uniform seeds in $(0, 1)$

Sampling in the clear

Generate $u_1, u_2, \ldots$ uniform seeds in $(0, 1)$

Arbitrary Distribution $\mathcal{D}$

Inversion method: $x \leftarrow CDF_{\mathcal{D}}^{-1}(u_1)$, if $CDF_{\mathcal{D}}^{-1}(x)$ is efficient

Sampling in the clear

Generate $u_1, u_2, \ldots$ uniform seeds in $(0, 1)$

Arbitrary Distribution $\mathcal{D}$

Inversion method: $x \leftarrow CDF_{\mathcal{D}}^{-1}(u_1)$, if $CDF_{\mathcal{D}}^{-1}(x)$ is efficient

Gaussian Distribution

- Central Limit Theorem Approach: $x \leftarrow \frac{1}{k} \sum_{i=1}^{k} u_i$

Sampling in the clear

Generate $u_1, u_2, \ldots$ uniform seeds in $(0,1)$

Arbitrary Distribution $\mathcal{D}$

Inversion method: $x \leftarrow CDF_{\mathcal{D}}^{-1}(u_1)$, if $CDF_{\mathcal{D}}^{-1}(x)$ is efficient

Gaussian Distribution

- Central Limit Theorem Approach: $x \leftarrow \frac{1}{k}\sum_{i=1}^{k} u_i$
- Box Müller:

$$\begin{cases} x_1 & \leftarrow \sqrt{-2\ln(u_1)}\sin(2\pi u_2) \\ x_2 & \leftarrow \sqrt{-2\ln(u_1)}\cos(2\pi u_2) \end{cases}$$

Sampling in the clear

Generate $u_1, u_2, \ldots$ uniform seeds in $(0, 1)$

Arbitrary Distribution $\mathcal{D}$

Inversion method: $x \leftarrow CDF_{\mathcal{D}}^{-1}(u_1)$, if $CDF_{\mathcal{D}}^{-1}(x)$ is efficient

Gaussian Distribution

- Central Limit Theorem Approach: $x \leftarrow \frac{1}{k}\sum_{i=1}^{k} u_i$
- Box Müller:

$$\begin{cases} x_1 & \leftarrow \sqrt{-2\ln(u_1)}\sin(2\pi u_2) \\ x_2 & \leftarrow \sqrt{-2\ln(u_1)}\cos(2\pi u_2) \end{cases}$$

- Polar Method: $u_1, u_2 \in (-1, 1)$

$$\begin{cases} \rho & = u_1^2 + u_2^2 \qquad \text{(if } \rho \geq 1 \text{ or } \rho = 0\text{, re-sample } u_1, u_2) \\ x_1 & \leftarrow u_1\sqrt{-2\ln(\rho)/\rho} \\ x_2 & \leftarrow u_2\sqrt{-2\ln(\rho)/\rho} \end{cases}$$

Outline

Introduction and Problem

Existent tools

Our solutions

Uniform Seeds

For $m \in \mathbb{N}$,

Uniform Seeds

For $m \in \mathbb{N}$,

1. P_i commits to a random $r \in [0, m)$

Uniform Seeds

For $m \in \mathbb{N}$,

1. P_i commits to a random $r \in [0, m)$
2. each party P_j: commit to a random $r_j \in [0, m)$
3. each party P_j: reveal r_j

Uniform Seeds

For $m \in \mathbb{N}$,

1. P_i commits to a random $r \in [0, m)$
2. each party P_j: commit to a random $r_j \in [0, m)$
3. each party P_j: reveal r_j
4. P_i commit to u and prove that $u = \left(r + \sum_{i=1}^{s} r_i\right) \bmod m$

Uniform Seeds

For $m \in \mathbb{N}$,

1. P_i commits to a random $r \in [0, m)$
2. each party P_j: commit to a random $r_j \in [0, m)$
3. each party P_j: reveal r_j
4. P_i commit to u and prove that $u = \left(r + \sum_{i=1}^{s} r_i\right) \bmod m$

We know that $u \sim \mathcal{U}\{0, \ldots, m-1\}$, but we don't know u.

Uniform Seeds

For $m \in \mathbb{N}$,

1. P_i commits to a random $r \in [0, m)$
2. each party P_j: commit to a random $r_j \in [0, m)$
3. each party P_j: reveal r_j
4. P_i commit to u and prove that $u = \left(r + \sum_{i=1}^{s} r_i\right) \bmod m$

We know that $u \sim \mathcal{U}\{0, \ldots, m-1\}$, but we don't know u.

Can amortize the generation of s uniforms with cost O(1) per party.

Proving Transcendental computations

Cryptographic Primitives for $\mathbb{R}$ (fixed-precision)

- Encode reals in $\mathbb{Z}$ (up to a certain fixed precision)
- Use integer proofs to implement computer operations: $+, \times$, bit-shift $(>>)$, $\div$
- requires dealing with rounding issues

Proving Transcendental computations

Cryptographic Primitives for $\mathbb{R}$ (fixed-precision)

- Encode reals in $\mathbb{Z}$ (up to a certain fixed precision)
- Use integer proofs to implement computer operations: $+, \times$, bit-shift $(>>)$, $\div$
- requires dealing with rounding issues

Use numerical approximations

From computer operations can compute

- $\sin, \cos, \log, e^x, \sqrt{x}$ with CORDIC algorithm [Walther, 1971] (mostly requires $+$ and $>>$)
- Gaussian $CDF^{-1}(x)$ with rational functions and Taylor polynomials

Proving Transcendental computations

Cryptographic Primitives for $\mathbb{R}$ (fixed-precision)

- Encode reals in $\mathbb{Z}$ (up to a certain fixed precision)
- Use integer proofs to implement computer operations: $+, \times$, bit-shift $(>>)$, $\div$
- requires dealing with rounding issues

Use numerical approximations

From computer operations can compute

- $\sin, \cos, \log, e^x, \sqrt{x}$ with CORDIC algorithm [Walther, 1971] (mostly requires $+$ and $>>$)
- Gaussian $CDF^{-1}(x)$ with rational functions and Taylor polynomials

We prove their correct execution

Preliminar results

Group Exponentiations (GExp) are the dominant computations

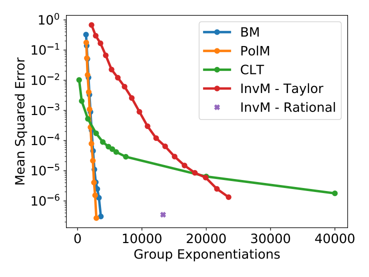
- Prove $\sin$, $\cos$, $\log$, e^x, $\sqrt{x}$ with n bits of precision with $O(n^2)$ GExp
  (Of independent interest in ML)

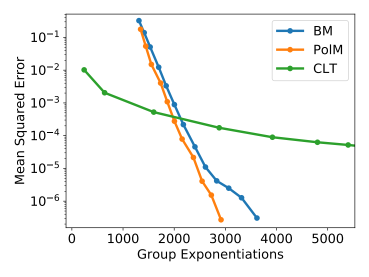
Simulated Gaussian sampling proofs

- Central Limit Theorem Approach (CLT)
- Box Muller (BM) and Polar Method (PolM)
- Inversion Method (InvM) with Taylor and rational approximations

Experiments

Measured MSE wrt to a quality Gaussian [1] over 10^7 samples per method



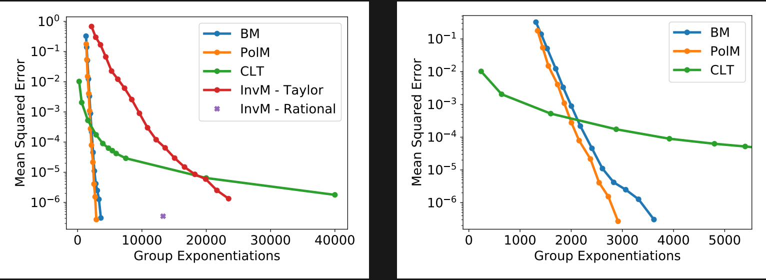


[1] Implemented with C++ boost library

2

Experiments

Measured MSE wrt to a quality Gaussian [1] over 10^7 samples per method



A quality sample requires < 3000 GExp

- ~ 0.17 seconds in an Intel Core i7 [2] (but largely optimizable)

[1] Implemented with C++ boost library

[2] With the implementation by [Franck and Großschädl, 2017]

Arbitrary Distributions: a sketch

Set Membership: Can prove $x \in S$ for private x and public $S \subset \mathbb{Z}_p$ [Camenisch et al., 2008]

Arbitrary Distributions: a sketch

Set Membership: Can prove $x \in S$ for private x and public $S \subset \mathbb{Z}_p$ [Camenisch et al., 2008]

Inversion Method with *table lookups* (in the clear)

- Sample from $\mathcal{D}$ distribution from 2^n equiprobable bins
- Precompute 2^n points $t_1, \ldots, t_{2^n}$ of CDF^{-1} in $(0, 1)$
- Sample uniformly $u \in \{1, \ldots, 2^n\}$ and return the u-th point

Arbitrary Distributions: a sketch

Set Membership: Can prove $x \in S$ for private x and public $S \subset \mathbb{Z}_p$ [Camenisch et al., 2008]

Inversion Method with *table lookups* (in the clear)

- Sample from $\mathcal{D}$ distribution from 2^n equiprobable bins
- Precompute 2^n points $t_1, \ldots, t_{2^n}$ of *CDF*$^{-1}$ in $(0, 1)$
- Sample uniformly $u \in \{1, \ldots, 2^n\}$ and return the u-th point

Private table lookups

- Let $D = \{\mathrm{enc}(1, t_1), \ldots, \mathrm{enc}(2^n, t_{2^n})\}$ for some integer encoding enc
- $u \sim \mathcal{U}\{1, \ldots, 2^n\} \wedge \mathrm{enc}(u, x) \in D \implies x \sim \mathcal{D}$

Arbitrary Distributions: a sketch

Set Membership: Can prove $x \in S$ for private x and public $S \subset \mathbb{Z}_p$ [Camenisch et al., 2008]

Inversion Method with *table lookups* (in the clear)

- Sample from $\mathcal{D}$ distribution from 2^n equiprobable bins
- Precompute 2^n points $t_1, \ldots, t_{2^n}$ of CDF^{-1} in $(0, 1)$
- Sample uniformly $u \in \{1, \ldots, 2^n\}$ and return the u-th point

Private table lookups

- Let $D = \{\mathrm{enc}(1, t_1), \ldots, \mathrm{enc}(2^n, t_{2^n})\}$ for some integer encoding enc
- $u \sim \mathcal{U}\{1, \ldots, 2^n\} \wedge \mathrm{enc}(u, x) \in D \implies x \sim \mathcal{D}$

$O(1)$ GExp per sample but $O(2^n)$ GExp of preprocessing (describing D)

Conclusion

- new approach to prove consiscency in Machine Learning
- computationally tractable proofs of transcendental relations and statisticals distributions

Conclusion

- new approach to prove consiscency in Machine Learning
- computationally tractable proofs of transcendental relations and statisticals distributions

Future Work

- Optimize numerical algorithms for cryptographic primitives
- Try other ZKP frameworks: compare prover work - verifier work - communication trade offs
- Plug our methods to Multiparty Computation frameworks (e.g ABY3 [Mohassel and Rindal, 2018])

Thank you!

References I

Attema, T. and Cramer, R. (2020).
Compressed $$\varSigma $$-Protocol Theory and Practical Application to Plug & Play Secure Algorithmics.
In Micciancio, D. and Ristenpart, T., editors, *Advances in Cryptology – CRYPTO 2020*, Lecture Notes in Computer Science, pages 513–543, Cham. Springer International Publishing.

Ben Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., and Virza, M. (2014).
Zerocash: Decentralized Anonymous Payments from Bitcoin.
In *2014 IEEE Symposium on Security and Privacy*, pages 459–474.
ISSN: 2375-1207.

References II

Blum, M. (1983).
Coin flipping by telephone a protocol for solving impossible problems.
ACM SIGACT News, 15(1):23–27.

Bünz, B., Agrawal, S., Zamani, M., and Boneh, D. (2020).
Zether: Towards Privacy in a Smart Contract World.
In Bonneau, J. and Heninger, N., editors, *Financial Cryptography and Data Security*, Lecture Notes in Computer Science, pages 423–443, Cham. Springer International Publishing.

Camenisch, J., Chaabouni, R., and Shelat, A. (2008).
Efficient Protocols for Set Membership and Range Proofs.
In *ASIACRYPT*.

References III

Cramer, R. (1997).
Modular Design of Secure yet Practical Cryptographic Protocols.
PhD thesis, University of Amsterdam.

Dwork, C. (2006).
Differential Privacy.
In *ICALP*.

Franck, C. and Großschädl, J. (2017).
Efficient Implementation of Pedersen Commitments Using Twisted Edwards Curves.
In Bouzefrane, S., Banerjee, S., Sailhan, F., Boumerdassi, S., and Renault, E., editors, *Mobile, Secure, and Programmable Networking*, volume 10566, pages 1–17. Springer International Publishing, Cham.
Series Title: Lecture Notes in Computer Science.

References IV

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Rouayheb, S. E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konečný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Raykova, M., Qi, H., Ramage, D., Raskar, R., Song, D., Song, W., Stich, S. U., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F. X., Yu, H., and Zhao, S. (2019). Advances and Open Problems in Federated Learning. Technical report, arXiv:1912.04977.

References V

Mohassel, P. and Rindal, P. (2018).
ABY3: A Mixed Protocol Framework for Machine Learning.
Technical Report 403.

Walther, J. S. (1971).
A unified algorithm for elementary functions.
In *Proceedings of the May 18-20, 1971, spring joint computer conference*, AFIPS '71 (Spring), pages 379–385, New York, NY, USA. Association for Computing Machinery.