Dependent Coeffects for Local Sensitivity Analysis

Victor Sannier Patrick Baillot

CRIStAL Laboratory, University of Lille, France

POPL 2026 @ Rennes, France

Outline

Goal

Design a type system to prove upper bounds on the local sensitivity of functional programs, with applications to differential privacy.

We will present the following points:

- (global) sensitivity and privacy preservation,
- linear logic and global sensitivity analysis,
- dependent coeffects and local sensitivity analysis (contribution),
- two case studies.

1. Introduction

From Differential Privacy to Linear Logic

Queries to Sensitive Data

A data analyst sends queries to a database containing private data.

Name	**Age**	**Condition**
Jane Doe	30	COVID-19
Richard Smith	18	smallpox
...	...	...

We want to provide an answer for statistical queries (such as the mean of a given column), but not to targeted queries.

Remark

Simply removing identifiers from the dataset is not enough. We need a formal definition of privacy preservation.

Differential Privacy[1]

Definition

A probabilistic query $q\colon \text{Data} \to X$ preserves *ϵ-differential privacy* whenever for all adjacent datasets D and D' (meaning $\operatorname{card}(D \triangle D') = 1$), we have

$$(\forall S \subseteq \operatorname{Range} q)\Big[\Pr(q(D) \in S) \leq \underbrace{\mathrm{e}^{\epsilon}}_{\text{close to 1}} \cdot \Pr(q(D') \in S)\Big]$$

Theorem

Differential privacy is a composable property.

[1] Cynthia Dwork et al. "Calibrating Noise to Sensitivity in Private Data Analysis". In: *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006*. Vol. 3876. Leibniz International Proceedings in Informatics (LIPIcs). 2006, pp. 265–284. DOI: 10.1007/11681878_14.

Differential Privacy[1]

Definition

A probabilistic query $q\colon \text{Data} \to X$ preserves ϵ-*differential privacy* whenever for all adjacent datasets D and D' (meaning $\text{card}(D \mathbin{\triangle} D') = 1$), we have

$$(\forall S \subseteq \text{Range}\, q)\Big[\Pr(q(D) \in S) \leq \underbrace{\mathrm{e}^{\epsilon}}_{\text{close to 1}} \cdot \Pr(q(D') \in S)\Big]$$

Theorem

Differential privacy is a composable property.

[1] Cynthia Dwork et al. "Calibrating Noise to Sensitivity in Private Data Analysis". In: *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006*. Vol. 3876. Leibniz International Proceedings in Informatics (LIPIcs). 2006, pp. 265–284. DOI: 10.1007/11681878_14.

Function (Global) Sensitivity

Definition

For all $s \in [0, +\infty]$, a function $f\colon (X, d_X) \to (Y, d_Y)$ is s-sensitive whenever the following holds:

$$(\forall x, x' \in X)\Big[d_Y\Big(f(x), f(x')\Big) \leq s \cdot d_X(x, x')\Big]$$

Example

For differentiable functions, the sensitivity of f is $\sup_{x \in X} |\mathrm{d}f/\mathrm{d}x\,(x)|$.

- $x \mapsto 2x + 3$ is 2-sensitive,
- $(x, y) \mapsto x \times y$ is ∞-sensitive.

The Laplace Mechanism[2]

If you know how sensitive a query is, then you know how to make it preserve differential privacy.

Theorem

If a query $q\colon \mathrm{Data} \to \mathbb{R}^n$ *is* s*-sensitive for* $s < +\infty$*, then* $D \mapsto f(D) + (Y_1, \ldots, Y_n)$*, preserves* ϵ*-differential privacy whenever the* Y_i *are i.i.d. random variables drawn from* $\mathrm{Lap}(s/\epsilon)$.

[2] Cynthia Dwork et al. "Calibrating Noise to Sensitivity in Private Data Analysis". In: *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006*. Vol. 3876. Leibniz International Proceedings in Informatics (LIPIcs). 2006, pp. 265–284. DOI: 10.1007/11681878_14.

The Fuzz Type System[3]

Reed and Pierce have introduced a linear type system for the λ-calculus to estimate the sensitivity of functional programs.

Fuzz′ judgements have the following form, where $s_1, \ldots, s_n$ are nonnegative real numbers: $x_1 :_{s_1} \sigma_1, \ldots, x_n :_{s_n} \sigma_n \vdash e : \tau$, and are interpreted as follows:

$$(\forall v_1, \ldots, v_n, v'_1, \ldots, v'_n)\left[d_\tau\left(\llbracket e \rrbracket(v_1, \ldots, v_n), \llbracket e \rrbracket(v'_1, \ldots, v'_n)\right) \leq \sum_{i=1}^{n} d_{\sigma_i}(v_i, v'_i)\right]$$

Example

$x :_2 \text{Int}, y :_1 \text{Int} \vdash 2x + y$ $\quad$ $x :_\infty \text{Int} \vdash x \times x : \text{Int}$ $\quad$ $l :_1 \text{List(Int)} \vdash \mathit{sort}\ l : \text{List(Int)}$

[3] Jason Reed and Benjamin C. Pierce. “Distance makes the types grow stronger: A calculus for differential privacy”. In: *ICFP’10: Proceedings of the 15th ACM SIGPLAN international conference on Functional programming*. 2010, pp. 157–168. DOI: 10.1145/1863543.1863568.

The Fuzz Type System[3]

Reed and Pierce have introduced a linear type system for the λ-calculus to estimate the sensitivity of functional programs.

Fuzz′ judgements have the following form, where $s_1, \ldots, s_n$ are nonnegative real numbers: $x_1 :_{s_1} \sigma_1, \ldots, x_n :_{s_n} \sigma_n \vdash e : \tau$, and are interpreted as follows:

$$(\forall v_1, \ldots, v_n, v_1', \ldots, v_n')\left[d_\tau\Big(\llbracket e \rrbracket(v_1, \ldots, v_n), \llbracket e \rrbracket(v_1', \ldots, v_n')\Big) \leq \sum_{i=1}^{n} d_{\sigma_i}(v_i, v_i')\right]$$

Example

$x :_2 \mathrm{Int}, y :_1 \mathrm{Int} \vdash 2x + y$ $\quad$ $x :_\infty \mathrm{Int} \vdash x \times x : \mathrm{Int}$ $\quad$ $l :_1 \mathrm{List}(\mathrm{Int}) \vdash \mathit{sort}\ l : \mathrm{List}(\mathrm{Int})$

---

[3] Jason Reed and Benjamin C. Pierce. "Distance makes the types grow stronger: A calculus for differential privacy". In: *ICFP'10: Proceedings of the 15th ACM SIGPLAN international conference on Functional programming*. 2010, pp. 157–168. DOI: 10.1145/1863543.1863568.

Fuzz's Typing Rules[4] (Subset)

$$\frac{s \geq 1}{\Gamma, x :_s \tau \vdash x : \tau}\ \text{var} \qquad \frac{\Gamma, x :_s \sigma \vdash e : \tau}{\Gamma \vdash \lambda x.e : !_s\sigma \multimap \tau} \to I$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \qquad \Delta \vdash e_2 : \tau_2}{\Gamma + \Delta \vdash (e_1, e_2) : \tau_1 \otimes \tau_2} \otimes I \qquad \frac{\Gamma \vdash e : \tau_1 \otimes \tau_2 \qquad \Delta, x :_r \tau_1, y :_r \tau_2 \vdash e' : \tau'}{\Delta + r\Gamma \vdash \mathbf{let}\ (x,y) = e\ \mathbf{in}\ e' : \tau'} \otimes E$$

where

- $\Gamma + \Delta$ is obtained by adding the sensitivity of the contexts variable-wise, and
- $s\Gamma$ is obtained by multiplying all sensitivities in Γ by s.

[4] Jason Reed and Benjamin C. Pierce. "Distance makes the types grow stronger: A calculus for differential privacy". In: *ICFP'10: Proceedings of the 15th ACM SIGPLAN international conference on Functional programming*. 2010, pp. 157–168. DOI: 10.1145/1863543.1863568.

Fuzz's Denotational Semantics[5]

Theorem

The category **Met** *of metric spaces and* 1*-sensitive maps is a monoidal closed category with weakening, finite (co)products, and a (compatible) graded comonad.*

Corollary

Met is a model of intuitionistic graded multiplicative-additive affine logic (i.e., a model of Fuzz).

This structure can be lifted to the category **MetCPO** of metric complete partial orders to interpret recursive types.

[5] Arthur Azevedo de Amorim et al. "A semantic account of metric preservation". In: *ACM SIGPLAN Notices*. Vol. 52. 1. 2017, pp. 545–556. DOI: 10.1145/3093333.3009890.

Fuzz's Denotational Semantics[5]

Theorem

The category **Met** *of metric spaces and* 1*-sensitive maps is a monoidal closed category with weakening, finite (co)products, and a (compatible) graded comonad.*

Corollary

Met is a model of intuitionistic graded multiplicative-additive affine logic (i.e., a model of Fuzz).

This structure can be lifted to the category **MetCPO** of metric complete partial orders to interpret recursive types.

[5] Arthur Azevedo de Amorim et al. "A semantic account of metric preservation". In: *ACM SIGPLAN Notices*. Vol. 52. 1. 2017, pp. 545–556. DOI: 10.1145/3093333.3009890.

2. Local Fuzz

From Global Sensitivity to Local Sensitivity

Beyond Global Sensitivity

- **the global sensitivity of many useful queries is infinite,** as asking for a uniform upper bound on $d_Y(f(x), f(x'))/d_X(x, x')$ is too strong,
- we are interested in the sensitivity of the query at only one specific point (the dataset), i.e., in the *local sensitivity*[6] of the query,
- algorithms such as the Propose-Test-Release (PTR) framework use the local sensitivity to answer to queries while preserving differential privacy.

[6] Kobbi Nassim, Sofya Raskhodnikova, and Adam Smith. "Smooth sensitivity and sampling in private data analysis". In: *STOC'07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. 2007, pp. 75–84. DOI: 10.1145/1250790.1250803.

Beyond Global Sensitivity

- **the global sensitivity of many useful queries is infinite,** as asking for a uniform upper bound on $d_Y(f(x), f(x'))/d_X(x, x')$ is too strong,
- we are interested in the sensitivity of the query at only one specific point (the dataset), i.e., in the *local sensitivity*[6] of the query,
- algorithms such as the Propose-Test-Release (PTR) framework use the local sensitivity to answer to queries while preserving differential privacy.

[6] Kobbi Nassim, Sofya Raskhodnikova, and Adam Smith. "Smooth sensitivity and sampling in private data analysis". In: *STOC'07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. 2007, pp. 75–84. DOI: 10.1145/1250790.1250803.

Beyond Global Sensitivity

- **the global sensitivity of many useful queries is infinite,** as asking for a uniform upper bound on $d_Y(f(x), f(x'))/d_X(x, x')$ is too strong,
- we are interested in the sensitivity of the query at only one specific point (the dataset), i.e., in the *local sensitivity*[6] of the query,
- algorithms such as the Propose-Test-Release (PTR) framework use the local sensitivity to answer to queries while preserving differential privacy.

[6] Kobbi Nassim, Sofya Raskhodnikova, and Adam Smith. "Smooth sensitivity and sampling in private data analysis". In: *STOC'07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. 2007, pp. 75–84. DOI: 10.1145/1250790.1250803.

Local Sensitivity at Radius k

Definition

A function $f\colon (X, d_X) \to (Y, d_Y)$ is s-sensitive **at point** $x \in X$ **and radius** $k \geq 0$ whenever the following holds:

$$(\forall x' \in X)\Big[d_X(x,x') \leq k \implies d_Y\Big(f(x), f(x')\Big) \leq s \cdot d_X(x,x')\Big]$$

This quantity so that it is both reasonably low and composable.

Example

For all $x \in \mathbb{R}$ and $k \geq 0$, we have $\mathrm{LS}_k(x \mapsto x^2, x) = 2|x|k$.

Local Fuzz's Types and Typing Judgements

Local Fuzz has the following types:

$$\sigma, \tau, \cdots ::= \mathsf{Unit} \mid \mathsf{Int} \mid \sigma \xrightarrow{s} \tau \mid \tau \otimes \tau \mid \tau \,\&\, \tau \mid \tau \oplus \tau \mid \mathsf{List}(\tau) \mid \mathsf{Set}(\tau)$$

where s ranges over $\mathcal{C}(\sigma)$, and

$$\mathcal{C}(\tau) := \{ s : \tau \to ([0, +\infty] \to [0, +\infty]) \mid s \text{ is a sensitivity modulus} \}.$$

and judgements have the following form:

$$x_1 : \sigma_1, \ldots, x_n : \sigma_n \,@\, s \vdash e : \tau$$

where the coeffect s is a sensitivity modulus, that is a term of the type $\Gamma \to ([0, +\infty]^n \to [0, +\infty])$.

Local Fuzz's Types and Typing Judgements

Local Fuzz has the following types:

$$\sigma, \tau, \cdots ::= \mathsf{Unit} \mid \mathsf{Int} \mid \sigma \xrightarrow{s} \tau \mid \tau \otimes \tau \mid \tau \mathbin{\&} \tau \mid \tau \oplus \tau \mid \mathsf{List}(\tau) \mid \mathsf{Set}(\tau)$$

where s ranges over $\mathcal{C}(\sigma)$, and

$$\mathcal{C}(\tau) := \{\, s : \tau \to ([0,+\infty] \to [0,+\infty]) \mid s \text{ is a sensitivity modulus} \,\}.$$

and judgements have the following form:

$$x_1 : \sigma_1, \ldots, x_n : \sigma_n \mathbin{@} s \vdash e : \tau$$

where the coeffect s is a sensitivity modulus, that is a term of the type $\Gamma \to ([0,+\infty]^n \to [0,+\infty])$.

Local Fuzz's Typing Rules (Subset)

$$\frac{}{x : \tau \,@\, \lambda x\,.\,\lambda k\,.\,k \vdash x : \tau}\ \text{var} \qquad \frac{\Gamma \,@\, s \vdash e_1 : \tau_1 \qquad \Delta \,@\, r \vdash e_2 : \tau_2}{\Gamma \cup \Delta \,@\, s + r \vdash (e_1, e_2) : \tau_1 \otimes \tau_2}\ \otimes I$$

$$\frac{\Gamma, x : \sigma \,@\, s + r \vdash e : \tau}{\Gamma \,@\, s \vdash \lambda x\,.\,e : \sigma \xrightarrow{r} \tau}\ \to I \qquad \frac{\Delta \,@\, r \vdash f : \sigma \xrightarrow{t} \tau \qquad \Gamma \,@\, s \vdash e : \sigma}{\Gamma \cup \Delta \,@\, t \times_e s + r \vdash f\,e : \tau}\ \to E$$

Theorem (Local Fuzz subsumes Fuzz)

When $s = \lambda x\,.\,\lambda k\,.\,s_1 k_1 + \cdots + s_n k_n$, where $s_1, \ldots, s_n$ are constants, then $x_1 : \sigma_1, \ldots, x_n : \sigma_n \,@\, s \vdash e : \tau$ behaves as the following Fuzz judgement:

$$x_1 :_{s_1} \sigma_1, \ldots, x_n :_{s_n} \sigma_n \vdash e : \tau.$$

Local Fuzz's Typing Rules (Subset)

$$\frac{}{x : \tau \mathbin{@} \lambda x\,.\,\lambda k\,.\,k \vdash x : \tau}\ \text{var} \qquad \frac{\Gamma \mathbin{@} s \vdash e_1 : \tau_1 \qquad \Delta \mathbin{@} r \vdash e_2 : \tau_2}{\Gamma \cup \Delta \mathbin{@} s + r \vdash (e_1, e_2) : \tau_1 \otimes \tau_2}\ \otimes I$$

$$\frac{\Gamma, x : \sigma \mathbin{@} s + r \vdash e : \tau}{\Gamma \mathbin{@} s \vdash \lambda x\,.\,e : \sigma \xrightarrow{r} \tau}\ \rightarrow I \qquad \frac{\Delta \mathbin{@} r \vdash f : \sigma \xrightarrow{t} \tau \qquad \Gamma \mathbin{@} s \vdash e : \sigma}{\Gamma \cup \Delta \mathbin{@} t \times_e s + r \vdash f\, e : \tau}\ \rightarrow E$$

Theorem (Local Fuzz subsumes Fuzz)

When $s = \lambda x\,.\,\lambda k\,.\,s_1 k_1 + \cdots + s_n k_n$, where $s_1, \ldots, s_n$ are constants, then $x_1 : \sigma_1, \ldots, x_n : \sigma_n \mathbin{@} s \vdash e : \tau$ behaves as the following Fuzz judgement:

$$x_1 :_{s_1} \sigma_1, \ldots, x_n :_{s_n} \sigma_n \vdash e : \tau.$$

Typing Rules for Primitives

For all primitives we want to add to our language ((×), *map*, *filter*, etc.), we write a corresponding typing rule.

$$\frac{\Gamma \mathbin{@} s \vdash a : \mathrm{Int} \qquad \Gamma \mathbin{@} r \vdash b : \mathrm{Int}}{\Gamma \mathbin{@} \lambda x \,.\, \lambda k \,.\, |a(x)| r(x,k) + |b(x)| s(x,k) + r(x,k) s(x,k) \vdash a \times b : \mathrm{Int}} \times$$

Example

We can derive $x : \mathrm{Int} \mathbin{@} \lambda x \,.\, \lambda k \,.\, 2|x|k + k^2 \vdash x^2 : \mathrm{Int}$.

Theorem

The category **PreMet** *of premetric spaces and* 1*-sensitive maps is a monoidal closed category with weakening, finite (co)products, and a (compatible) dependently graded comonad*[7].

Corollary

If $\Gamma \; @ \; s \vdash e : \tau$, then for all inputs x, we have $\mathrm{LS}_1(\llbracket e \rrbracket, x) \leq \llbracket s \rrbracket(x, 1)$.

If we can type a query, then we know how to make it preserve differential privacy.

[7] Matteo Capucci and David Jaz Myers. *Contextads as Wreaths; Kleisli, Para, and Span Constructions as Wreath Products.* 2024. arXiv: 2410.21889 [math.CT].

3. Case Studies

Global Sensitivity Analysis

In judgements where the coeffects does not depend on the variables, such as

$$x_1 : \tau_1, \ldots, x_n : \tau_n \mathbin{@} \lambda x \,.\, \lambda k \,.\, \sum_{i=1}^{n} s_i k_i \vdash e : \tau,$$

the coeffect is a bound on the *global sensitivity* of $[\![e]\!]$.

Theorem

There are terms for which we can derive better global sensitivity bounds in Local Fuzz than in Fuzz.

In these cases, we can apply the Laplace mechanism in order to build a privacy-preserving query.

Global Sensitivity Analysis

In judgements where the coeffects does not depend on the variables, such as

$$x_1 : \tau_1, \ldots, x_n : \tau_n \,@\, \lambda x \,.\, \lambda k \,.\, \sum_{i=1}^{n} s_i k_i \vdash e : \tau,$$

the coeffect is a bound on the *global sensitivity* of $[\![e]\!]$.

Theorem

There are terms for which we can derive better global sensitivity bounds in Local Fuzz than in Fuzz.

In these cases, we can apply the Laplace mechanism in order to build a privacy-preserving query.

Typical range of normally distributed values

Let X be a set of normally distributed values (real numbers). We would like to find an interval I containing 95% of these, say $I = [\mu - 2\sigma, \mu + 2\sigma]$, while preserving differential privacy.

Algorithm

```
let typical_range = fun x ->
  let mu = quartile 2 x in
  let iqr = quartile 3 x - quartile 1 x in
  let sigma = iqr / (2 * sqrt 2 * erfinv (0.5)) in
  (mu - 2*sigma, mu + 2*sigma)
```

This algorithm uses only one primitive: *quartile*.

Local Sensitivity of the Quartile Function

We can show that

$$\mathrm{LS}_k\Big(\mathit{quartile}_i, (x_1, \ldots, x_n)\Big) = \max\Big(\Big|x_{i\cdot(n+1)/4} - x_{i\cdot(n+1)/4+k}\Big|, \Big|x_{i\cdot(n+1)/4} - x_{i\cdot(n+1)/4-k}\Big|\Big)$$

so we can soundly introduce the following typing rule

$$\frac{i \in \{1,2,3\} \qquad \Gamma \mathbin{@} s \vdash l : \mathrm{Set}(\mathrm{Int})}{\Gamma \mathbin{@} q_i \vdash \mathit{quartile}_i(l) : \mathrm{Int}}\ \mathit{quartile}$$

where $q_i := \Big(\lambda x \,.\, \lambda k \,.\, \mathrm{LS}_k(\mathit{quartile}_i, x)\Big) \times_l s$

DP-Preserving Version of the Algorithm

In Local Fuzz, we can derive $\vdash$ *typical_range* : $\mathrm{Set}(\mathrm{Int}) \xrightarrow{s} \mathrm{Int} \otimes \mathrm{Int}$, where

$$s := \lambda x\,.\,\lambda k\,.\,2\mathrm{LS}_k(quartile_2, x) + (\sqrt{2}/\,erfinv(1/2)) \cdot (\mathrm{LS}_k(quartile_3, x) + \mathrm{LS}_k(quartile_1, x)).$$

We can apply the following theorem:

Theorem

If we know (an upper bound on) the local sensitivity of a query, we can use the Propose-Test-Release[8] framework to make it differentially private.

[8] Cynthia Dwork and Jing Lei. "Differential privacy and robust statistics". In: *STOC'09: Proceedings of the forty-first annual ACM symposium on Theory of computing*. 2009, pp. 371–380. DOI: 10.1145/1536414.1536466.

4. Conclusion

Summary and Future Work

Summary. In this work, we have designed a type system for deriving upper bounds on the local sensitivity of functional programs, with applications to differential privacy.

Future Work. We expect that any monoidal closed category endowed with a compatible dependently graded comonad will provide a model for a calculus much similar to Local Fuzz. We would also like to determine whether type checking is decidable.

Summary and Future Work

Summary. In this work, we have designed a type system for deriving upper bounds on the local sensitivity of functional programs, with applications to differential privacy.

Future Work. We expect that any monoidal closed category endowed with a compatible dependently graded comonad will provide a model for a calculus much similar to Local Fuzz. We would also like to determine whether type checking is decidable.

Summary and Future Work

Summary. In this work, we have designed a type system for deriving upper bounds on the local sensitivity of functional programs, with applications to differential privacy.

Future Work. We expect that any monoidal closed category endowed with a compatible dependently graded comonad will provide a model for a calculus much similar to Local Fuzz. We would also like to determine whether type checking is decidable.

Acknowledgements

This work was supported by the French National Agency for Research as part of the HOPR project (ANR-24-CE48-5521-01) and by the Hauts-de-France region.

Dependently Graded Comonads[9]

Definition

A *dependently graded comonad* on a category $\mathcal{C}$ consists of the following data, subject to a number of naturality and associativity conditions:

1. For each object $C \in \mathcal{C}$, a category $\mathcal{M}_C$ of *grades*, assigned contravariantly to C.
2. For each $C \in \mathcal{C}$ and $M \in \mathcal{M}_C$, an object $C \odot M \in \mathcal{C}$, representing the action of the grade on the object.
3. For each $C \in \mathcal{C}$, a unit grade $I_C \in \mathcal{M}_C$.
4. For any grades $M \in \mathcal{M}_C$ and $N \in \mathcal{M}_{C \odot M}$, a combined grade $M \otimes N \in \mathcal{M}_C$.
5. For each object C, a colax unitor $\varepsilon_C : C \odot I_C \to C$.
6. For each object C and grades M and N, a colax associator $\delta_{C,M,N} : C \odot (M \otimes N) \to (C \odot M) \odot N$.

[9] Matteo Capucci and David Jaz Myers. *Contextads as Wreaths; Kleisli, Para, and Span Constructions as Wreath Products*. 2024. arXiv: 2410.21889 [math.CT].

- Azevedo de Amorim, Arthur et al. "A semantic account of metric preservation". In: *ACM SIGPLAN Notices*. Vol. 52. 1. 2017, pp. 545–556. DOI: 10.1145/3093333.3009890.
- Capucci, Matteo and David Jaz Myers. *Contextads as Wreaths; Kleisli, Para, and Span Constructions as Wreath Products*. 2024. arXiv: 2410.21889 [math.CT].
- Dwork, Cynthia and Jing Lei. "Differential privacy and robust statistics". In: *STOC'09: Proceedings of the forty-first annual ACM symposium on Theory of computing*. 2009, pp. 371–380. DOI: 10.1145/1536414.1536466.
- Dwork, Cynthia et al. "Calibrating Noise to Sensitivity in Private Data Analysis". In: *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006*. Vol. 3876. Leibniz International Proceedings in Informatics (LIPIcs). 2006, pp. 265–284. DOI: 10.1007/11681878_14.

References II

- Nassim, Kobbi, Sofya Raskhodnikova, and Adam Smith. "Smooth sensitivity and sampling in private data analysis". In: *STOC'07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. 2007, pp. 75–84. DOI: 10.1145/1250790.1250803.
- Reed, Jason and Benjamin C. Pierce. "Distance makes the types grow stronger: A calculus for differential privacy". In: *ICFP'10: Proceedings of the 15th ACM SIGPLAN international conference on Functional programming*. 2010, pp. 157–168. DOI: 10.1145/1863543.1863568.