Session Types for the Concurrent Composition of Interactive Differential Privacy

Victor Sannier
Université de Lille, France

Patrick Baillot
CNRS, Lille, France

Marco Gaboardi
Boston University, MA

2025 IEEE 38th Computer Security Foundations Symposium (CSF)

Work supported by the French National Agency for Research (ANR-24-CE48-5521-01)
and by the Hauts-de-France region.

1. Differential Privacy

Motivation of Differential Privacy

A data analyst queries a database containing sensitive information.
We would like to define what is a *privacy-preserving query*.

Motivation of Differential Privacy

A data analyst queries a database containing sensitive information.
We would like to define what is a *privacy-preserving query*.

✓ What is the number of entries in the database?

Motivation of Differential Privacy

A data analyst queries a database containing sensitive information.
We would like to define what is a *privacy-preserving query*.

- ✓ What is the number of entries in the database?
- ✓ What is the average salary of employees at General Informatic?

Motivation of Differential Privacy

A data analyst queries a database containing sensitive information.
We would like to define what is a *privacy-preserving query*.

- ✓ What is the number of entries in the database?
- ✓ What is the average salary of employees at General Informatic?
- ✗ How much does Jane Doe earn?

Motivation of Differential Privacy

A data analyst queries a database containing sensitive information.
We would like to define what is a *privacy-preserving query*.

- ✓ What is the number of entries in the database?
- ✓ What is the average salary of employees at General Informatic?
- ✗ How much does Jane Doe earn?

The answer to a privacy-preserving query does not depend on any individual in particular.

Formal Definition of Differentially Privacy

Definition (Dwork et al. 2006, Definition 1)

A probabilistic algorithm M is (ϵ, δ)-*differentially private* if, for any pair of adjacent databases D and D', the following condition holds:

$$M(D) \underset{(\epsilon,\delta)}{\approx} M(D')$$

Formal Definition of Differentially Privacy

Definition (Dwork et al. 2006, Definition 1)

A probabilistic algorithm M is (ϵ, δ)-*differentially private* if, for any pair of adjacent databases D and D', the following condition holds:

$$\forall X \,.\, \Pr[M(D) \in X] \leq e^{\epsilon} \Pr[M(D') \in X] + \delta$$

- ϵ: privacy loss (the smaller the better),
- δ: probability that the differential privacy guarantee fails,
- $(\epsilon, \delta) = (0, 1)$ provides no differential privacy guarantee.

Formal Definition of Differentially Privacy

Definition (Dwork et al. 2006, Definition 1)

A probabilistic algorithm M is (ϵ, δ)-*differentially private* if, for any pair of adjacent databases D and D', the following condition holds:

$$\forall X \,.\, \Pr[M(D) \in X] \leq e^{\epsilon} \Pr[M(D') \in X] + \delta$$

- ϵ: privacy loss (the smaller the better),
- δ: probability that the differential privacy guarantee fails,
- $(\epsilon, \delta) = (0, 1)$ provides no differential privacy guarantee.

Theorem

Differential privacy is compositional.

Differential Privacy and Noise Addition

Theorem (Laplace mechanism)

If $q : \text{Data} \to \mathbb{R}$ *is a query such that for some* k

$$\forall D, D' \,.\, D \sim D' \Rightarrow |q(D) - q(D')| \leq k,$$

then the query

$$D \mapsto q(D) + noise(k)$$

preserves differential privacy, where the noise is drawn from a Laplace distribution.

Differential Privacy and Noise Addition

Theorem (Laplace mechanism)

If $q : \text{Data} \to \mathbb{R}$ *is a query such that for some* k

$$\forall D, D' \,.\, D \sim D' \Rightarrow |q(D) - q(D')| \leq k,$$

then the query

$$D \mapsto q(D) + noise(k)$$

preserves differential privacy, where the noise is drawn from a Laplace distribution.

If you know how sensitive a request is, you know how to make it privacy-preserving.

The Fuzz Calculus

Reed and Pierce have introduced Fuzz: a calculus inspired by (graded) linear logic Girard (1987).

The Fuzz Calculus

Reed and Pierce have introduced Fuzz: a calculus inspired by (graded) linear logic Girard (1987).

$$\frac{[x : \mathrm{Data}]_r \vdash f(x) : \mathbb{R} \qquad [x : \mathrm{Data}]_s \vdash g(x) : \mathbb{R}}{[x : \mathrm{Data}]_{r+s} \vdash f(x) + g(x) : \mathbb{R}}$$

The Fuzz Calculus

Reed and Pierce have introduced Fuzz: a calculus inspired by (graded) linear logic Girard (1987).

$$\frac{[x : \mathrm{Data}]_r \vdash f(x) : \mathbb{R} \qquad [x : \mathrm{Data}]_s \vdash g(x) : \mathbb{R}}{[x : \mathrm{Data}]_{r+s} \vdash f(x) + g(x) : \mathbb{R}}$$

Definition

A function f between two metric spaces (X, d_X) and (Y, d_Y) is s-sensitive if for all x and x' in X, we have $d_Y\big(f(x), f(x')\big) \leq s \cdot d_X(x, x')$.

The Fuzz Calculus

Reed and Pierce have introduced Fuzz: a calculus inspired by (graded) linear logic Girard (1987).

$$\frac{[x : \mathrm{Data}]_r \vdash f(x) : \mathbb{R} \qquad [x : \mathrm{Data}]_s \vdash g(x) : \mathbb{R}}{[x : \mathrm{Data}]_{r+s} \vdash f(x) + g(x) : \mathbb{R}}$$

Definition

A function f between two metric spaces (X, d_X) and (Y, d_Y) is s-sensitive if for all x and x' in X, we have $d_Y\big(f(x), f(x')\big) \leq s \cdot d_X(x, x')$.

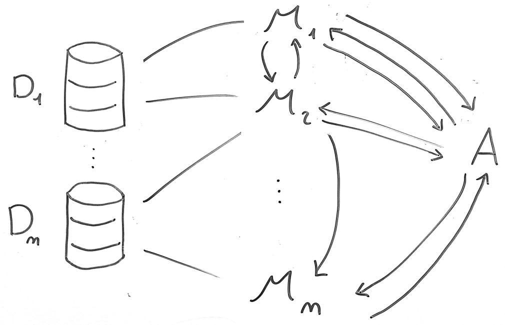
Theorem (Soundness)

If $[x : \sigma]_s \vdash f : \tau$*, then* $[\![f]\!]$ *is s-sensitive, and* $[\![f]\!] + \mathrm{noise}(s)$ *preserves DP.*

2. Interactive Differential Privacy

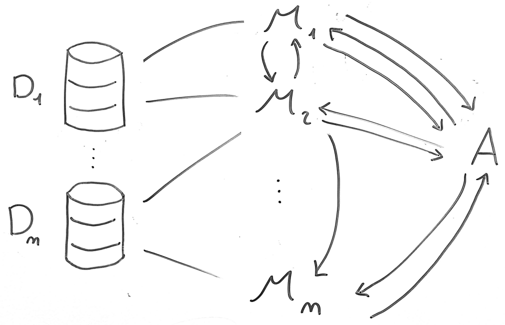
Interactive Mechanisms

What if we enable communication through multiple mechanisms, allowing, for example, the answer from one mechanism to be used to query another?



Interactive Mechanisms

What if we enable communication through multiple mechanisms, allowing, for example, the answer from one mechanism to be used to query another?



What would serve as the output of the mechanisms?

Interactive Differential Privacy

Definition (Vadhan and Wang 2021, Definition 1.6)

The *view* $\mathrm{View}(A \parallel M)$ of a party A interacting with M consists of

- all the messages it receives during the interaction,
- its private input, and
- the random numbers it generates.

Interactive Differential Privacy

Definition (Vadhan and Wang 2021, Definition 1.6)

The *view* $\text{View}(A \parallel M)$ of a party A interacting with M consists of

- all the messages it receives during the interaction,
- its private input, and
- the random numbers it generates.

Definition (Vadhan and Wang 2021, Definition 1.7)

M is an (ϵ, δ)-differentially private interactive mechanism if, for every pair of adjacent datasets D and D',

Interactive Differential Privacy

Definition (Vadhan and Wang 2021, Definition 1.6)

The *view* $\mathrm{View}(A \parallel M)$ of a party A interacting with M consists of

- all the messages it receives during the interaction,
- its private input, and
- the random numbers it generates.

Definition (Vadhan and Wang 2021, Definition 1.7)

M is an (ϵ, δ)-differentially private interactive mechanism if, for every pair of adjacent datasets D and D', every adversary A,

Interactive Differential Privacy

Definition (Vadhan and Wang 2021, Definition 1.6)

The *view* $\mathrm{View}(A \,\|\, M)$ of a party A interacting with M consists of

- all the messages it receives during the interaction,
- its private input, and
- the random numbers it generates.

Definition (Vadhan and Wang 2021, Definition 1.7)

M is an (ϵ, δ)-differentially private interactive mechanism if, for every pair of adjacent datasets D and D', every adversary A, and every set X,

Interactive Differential Privacy

Definition (Vadhan and Wang 2021, Definition 1.6)

The *view* $\mathrm{View}(A \parallel M)$ of a party A interacting with M consists of

- all the messages it receives during the interaction,
- its private input, and
- the random numbers it generates.

Definition (Vadhan and Wang 2021, Definition 1.7)

M is an (ϵ, δ)-differentially private interactive mechanism if, for every pair of adjacent datasets D and D', every adversary A, and every set X,

$$\Pr\left[\mathrm{View}(A \parallel M(D)) \in X\right] \leq e^{\epsilon} \Pr\left[\mathrm{View}(A \parallel M(D')) \in X\right] + \delta\,.$$

Concurrent Composition of Interactive Differential Privacy

Theorem (Vadhan and Wang 2021, Theorem 1.8)

If interactive mechanisms $(M_1, \ldots, M_k)$ *are each* (ϵ, δ)*-differentially private, then their concurrent composition* $\mathrm{ConComp}(M_1, \ldots, M_k)$ *is* $\left(k\epsilon, \frac{e^{k\epsilon-1}}{e^{\epsilon-1}}\delta\right)$*-DP.*

Concurrent Composition of Interactive Differential Privacy

Theorem (Vadhan and Wang 2021, Theorem 1.8)

If interactive mechanisms $(M_1, \ldots, M_k)$ *are each* (ϵ, δ)*-differentially private, then their concurrent composition* $\mathrm{ConComp}(M_1, \ldots, M_k)$ *is* $\left(k\epsilon, \frac{e^{k\epsilon-1}}{e^{\epsilon-1}}\delta\right)$*-DP.*

As with (centralised) differential privacy, an interactive process can be proven to preserve differential privacy by proving that each of its components does.

Concurrent Composition of Interactive Differential Privacy

Theorem (Vadhan and Wang 2021, Theorem 1.8)

If interactive mechanisms $(M_1, \ldots, M_k)$ *are each* (ϵ, δ)*-differentially private, then their concurrent composition* $\mathrm{ConComp}(M_1, \ldots, M_k)$ *is* $\left(k\epsilon, \frac{e^{k\epsilon-1}}{e^{\epsilon-1}}\delta\right)$*-DP.*

As with (centralised) differential privacy, an interactive process can be proven to preserve differential privacy by proving that each of its components does.

centralised DP : Fuzz :: interactive DP : ?

3. Process Calculi and Session Types

The (untyped) π-calculus

Just as λ-calculus serves as a model for non-interactive computation, we need a model for interactive computation.

The (untyped) π-calculus

Just as λ-calculus serves as a model for non-interactive computation, we need a model for interactive computation.

The π-calculus is a calculus of concurrent processes introduced by Milner, Parrow, and Walker (1992).

- **if** e **then** P **else** Q behaves as P if e evaluates to $\top$, and as Q otherwise,

The (untyped) π-calculus

Just as λ-calculus serves as a model for non-interactive computation, we need a model for interactive computation.

The π-calculus is a calculus of concurrent processes introduced by Milner, Parrow, and Walker (1992).

- **if** e **then** P **else** Q behaves as P if e evaluates to $\top$, and as Q otherwise,
- $k![e] \,.\, P$ and $k?(e) \,.\, P$ respectively sends and waits for an expression e over a channel k, and then continues as process P,

The (untyped) π-calculus

Just as λ-calculus serves as a model for non-interactive computation, we need a model for interactive computation.

The π-calculus is a calculus of concurrent processes introduced by Milner, Parrow, and Walker (1992).

- **if** e **then** P **else** Q behaves as P if e evaluates to $\top$, and as Q otherwise,
- $k![e] \,.\, P$ and $k?(e) \,.\, P$ respectively sends and waits for an expression e over a channel k, and then continues as process P,
- $P \parallel Q$ is the parallel composition of P and Q,
- ...

The (untyped) π-calculus

Just as λ-calculus serves as a model for non-interactive computation, we need a model for interactive computation.

The π-calculus is a calculus of concurrent processes introduced by Milner, Parrow, and Walker (1992).

- **if** e **then** P **else** Q behaves as P if e evaluates to $\top$, and as Q otherwise,
- $k![e] . P$ and $k?(e) . P$ respectively sends and waits for an expression e over a channel k, and then continues as process P,
- $P \parallel Q$ is the parallel composition of P and Q,
- ...

Example

$$\big(k![1] . k?[x] . \dots\big) \parallel \big(k?[x] . k![x + x] . \dots\big)$$

Session Types

To statically assert properties of processes, we type the interactions between them.

Session Types

To statically assert properties of processes, we type the interactions between them.

- Sessions types were originally introduced by Takeuchi, Honda, and Kubo (1994) and further developed by Honda, Vasconcelos, and Kubo (1998).

Session Types

To statically assert properties of processes, we type the interactions between them.

- Sessions types were originally introduced by Takeuchi, Honda, and Kubo (1994) and further developed by Honda, Vasconcelos, and Kubo (1998).
- A session is defined as a sequence of reciprocal interactions between two parties.

Session Types

To statically assert properties of processes, we type the interactions between them.

- Sessions types were originally introduced by Takeuchi, Honda, and Kubo (1994) and further developed by Honda, Vasconcelos, and Kubo (1998).
- A session is defined as a sequence of reciprocal interactions between two parties.
- The typing judgements have the form $\Gamma \vdash P \triangleright \Delta$ (Γ: types for variables and session names, Δ: types for session channels)

Session Types

To statically assert properties of processes, we type the interactions between them.

- Sessions types were originally introduced by Takeuchi, Honda, and Kubo (1994) and further developed by Honda, Vasconcelos, and Kubo (1998).
- A session is defined as a sequence of reciprocal interactions between two parties.
- The typing judgements have the form $\Gamma \vdash P \triangleright \Delta$ (Γ: types for variables and session names, Δ: types for session channels)

Theorem

- *Typing is preserved by reduction.*
- *A typable program never reduces into an error.*

4. Session Types for Interactive Differential Privacy

New Constructs for the π-calculus

We introduce two new constructs to the standard π-calculus:

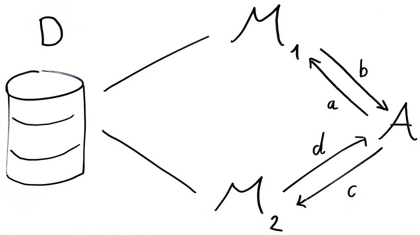
- $\mathbf{Lap}_b(x)\,.\,P$ to sample a random number from the (discrete) Laplace distribution with parameter b and continue according to P,

New Constructs for the π-calculus

We introduce two new constructs to the standard π-calculus:

- $\mathbf{Lap}_b(x)\,.\,P$ to sample a random number from the (discrete) Laplace distribution with parameter b and continue according to P,
- $*_n P$ for the replication of the process P n times (this serves as a as a partial replacement for recursive processes).

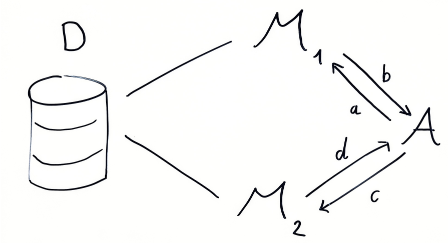
Example of a Concurrent Composition



Let M_1 and M_2 be two differentially private mechanisms.

$$M_i = k_i?(f) \,.\, \mathbf{Lap}_{1/\epsilon}?(r) \,.\, k_i![f(D) + r] \,.\, 0$$

Example of a Concurrent Composition

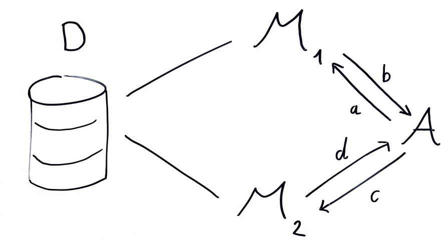


Let M_1 and M_2 be two differentially private mechanisms.

$M_i = k_i?(f)\,.\,\mathbf{Lap}_{1/\epsilon}?(r)\,.\,k_i![f(D)+r]\,.\,0$

The session between A and M_i will have the type $(\alpha, \bar{\alpha})$. where $\alpha = ?(\text{Data} \multimap \text{Num})\,.\,!\text{Num}\,.\,\text{end}$,

Example of a Concurrent Composition



Let M_1 and M_2 be two differentially private mechanisms.

$$M_i = k_i?(f) \,.\, \mathbf{Lap}_{1/\epsilon}?(r) \,.\, k_i![f(D)+r] \,.\, 0$$

The session between A and M_i will have the type $(\alpha, \bar{\alpha})$. where $\alpha = ?(\mathrm{Data} \multimap \mathrm{Num}) \,.\, !\mathrm{Num} \,.\, \mathrm{end}$,

$M_1 \parallel M_2$ is also differentially private, which means that it does not leak private information when interacting with *any* adversary.
For example, one possible adversary is

$$A = k_1![f] \,.\, k_1?(y_1) \,.\, k_2![g(y_1)] \,.\, k_2?(y_2) \,.\, \ldots \,.$$

Typing Judgements

We consider two forms of typing judgements:

- the first one applies to expressions from a standard functional language.

$$\Gamma \vdash e : A\,,$$

(Expressions are exchanged between processes through channels.)

Typing Judgements

We consider two forms of typing judgements:

- the first one applies to expressions from a standard functional language.

  $$\Gamma \vdash e : A\,,$$

  (Expressions are exchanged between processes through channels.)
- the second one concerns processes

  $$\Gamma \vdash P \triangleright \Delta; (\epsilon, \delta)\,.$$

  (Read "P is a well-typed (ϵ, δ)-differentially private process.")

Typing Judgements

We consider two forms of typing judgements:

- the first one applies to expressions from a standard functional language.

  $$\Gamma \vdash e : A\,,$$

  (Expressions are exchanged between processes through channels.)
- the second one concerns processes

  $$\Gamma \vdash P \triangleright \Delta; (\epsilon, \delta)\,.$$

  (Read "P is a well-typed (ϵ, δ)-differentially private process.")

In practice, we use Fuzz as our expression language to benefit from its capability for sensitivity analysis in our typing rules.

Examples of Typing Rules

$$\frac{\Gamma \vdash e : \mathsf{Bool} \qquad \Gamma \vdash P \triangleright \Delta; (\epsilon_P, \delta_P) \qquad \Gamma \vdash Q \triangleright \Delta; (\epsilon_Q, \delta_Q)}{\Gamma \vdash \textbf{if } e \textbf{ then } P \textbf{ else } Q \triangleright \Delta; (0, 1)} \text{ [T-If]}$$

Examples of Typing Rules

$$\frac{\Gamma \vdash e : \mathrm{Bool} \qquad \Gamma \vdash P \triangleright \Delta; (\epsilon_P, \delta_P) \qquad \Gamma \vdash Q \triangleright \Delta; (\epsilon_Q, \delta_Q)}{\Gamma \vdash \textbf{if } e \textbf{ then } P \textbf{ else } Q \triangleright \Delta; (0, 1)} \text{[T-If]}$$

$$\frac{\Gamma \vdash P_1 \triangleright \Delta_1; (\epsilon_1, \delta_1) \qquad \Gamma \vdash P_2 \triangleright \Delta_2; (\epsilon_2, \delta_2) \qquad \Delta_1 \asymp \Delta_2}{\Gamma \vdash P_1 \parallel P_2 \triangleright \Delta_1 \circ \Delta_2; (\epsilon_1, \delta_1) \star (\epsilon_2, \delta_2)} \text{[T-Conc]}$$

where $(\epsilon_1, \delta_1) \star (\epsilon_2, \delta_2) = \big(\epsilon_1 + \epsilon_2, e^{\epsilon_1 + \epsilon_2} \cdot (\delta_1 + \delta_2)\big)$.

Examples of Typing Rules

$$\frac{\Gamma \vdash e : \text{Bool} \qquad \Gamma \vdash P \triangleright \Delta; (\epsilon_P, \delta_P) \qquad \Gamma \vdash Q \triangleright \Delta; (\epsilon_Q, \delta_Q)}{\Gamma \vdash \textbf{if } e \textbf{ then } P \textbf{ else } Q \triangleright \Delta; (0, 1)} \text{ [T-If]}$$

$$\frac{\Gamma \vdash P_1 \triangleright \Delta_1; (\epsilon_1, \delta_1) \qquad \Gamma \vdash P_2 \triangleright \Delta_2; (\epsilon_2, \delta_2) \qquad \Delta_1 \asymp \Delta_2}{\Gamma \vdash P_1 \parallel P_2 \triangleright \Delta_1 \circ \Delta_2; (\epsilon_1, \delta_1) \star (\epsilon_2, \delta_2)} \text{ [T-Conc]}$$

where $(\epsilon_1, \delta_1) \star (\epsilon_2, \delta_2) = \left(\epsilon_1 + \epsilon_2, e^{\epsilon_1 + \epsilon_2} \cdot (\delta_1 + \delta_2)\right)$.

$$\frac{\Gamma_1 \vdash P_1 \triangleright \Delta_1; (\epsilon, 0) \qquad \Gamma_2 \vdash P_2 \triangleright \Delta_2; (\epsilon, 0) \qquad \Delta_1 \asymp \Delta_2}{\Gamma_1 \coprod \Gamma_2 \vdash P_1 \parallel P_2 \triangleright \Delta_1 \circ \Delta_2; (\epsilon, 0)} \text{ [T-Par]}$$

Examples of Reduction Rules

We provide an operational semantics in terms of probabilistic automata with binary trees as labels.

We provide an operational semantics in terms of probabilistic automata with binary trees as labels.

$$\frac{P\left\{\xrightarrow[p_i]{t_i} P_i\right\}_i}{P \parallel Q\left\{\xrightarrow[p_i]{(t_i,\varnothing)} P_i \parallel Q\right\}_i} \text{ [R-Conc]}$$

Examples of Reduction Rules

We provide an operational semantics in terms of probabilistic automata with binary trees as labels.

$$\frac{P\left\{\xrightarrow[p_i]{t_i} P_i\right\}_i}{P \parallel Q\left\{\xrightarrow[p_i]{(t_i,\varnothing)} P_i \parallel Q\right\}_i}\ \text{[R-Conc]}$$

$$\frac{e \downarrow v}{k![e]\,.\,P \parallel k?(x)\,.\,Q\left\{\xrightarrow[1]{(\alpha_v,\alpha_v)} P \parallel Q[v/x]\right\}}\ \text{[R-Val]}$$

Examples of Reduction Rules

We provide an operational semantics in terms of probabilistic automata with binary trees as labels.

$$\frac{P\left\{\xrightarrow[p_i]{t_i} P_i\right\}_i}{P \parallel Q\left\{\xrightarrow[p_i]{(t_i,\emptyset)} P_i \parallel Q\right\}_i} \text{[R-Conc]}$$

$$\frac{e \downarrow v}{k![e] \,.\, P \parallel k?(x) \,.\, Q\left\{\xrightarrow[1]{(\alpha_v,\alpha_v)} P \parallel Q[v/x]\right\}} \text{[R-Val]}$$

$$\frac{}{\mathbf{Lap}_b?(x) \,.\, P\left\{\xrightarrow[p_{n,b}]{\gamma_n} P[n/x]\right\}_{n\in\mathbf{Z}}} \text{[R-Lap]}$$

View of a process

The *trace* of the execution of P is the unique random variable $\mathrm{Trace}(P)$ such that if $P\left\{\xrightarrow[p_i]{t_i}{}^{*} P_i\right\}$, then $\Pr[\mathrm{Trace}(P) = t_i] = p_i$.

View of a process

The *trace* of the execution of P is the unique random variable $\mathrm{Trace}(P)$ such that if $P\left\{\xrightarrow[p_i]{t_i} {}^* P_i\right\}$, then $\Pr[\mathrm{Trace}(P) = t_i] = p_i$.

Definition

The *view* of a process A interacting with a process M, is the following random variable: $\mathrm{View}(A \parallel M) = \mathrm{Left}\big(\mathrm{Trace}(A \parallel M)\big)$.

Differential Privacy as Approximate Trace Equivalence

We can use the same definition of interactive differential privacy as Vadhan, where View is formally defined as above.

Definition (Vadhan and Wang 2021, Definition 1.7)

M is an (ϵ, δ)-differentially private interactive mechanism if, for every pair of adjacent datasets D and D',

Differential Privacy as Approximate Trace Equivalence

We can use the same definition of interactive differential privacy as Vadhan, where View is formally defined as above.

Definition (Vadhan and Wang 2021, Definition 1.7)

M is an (ϵ, δ)-differentially private interactive mechanism if, for every pair of adjacent datasets D and D', every adversary A,

Differential Privacy as Approximate Trace Equivalence

We can use the same definition of interactive differential privacy as Vadhan, where View is formally defined as above.

Definition (Vadhan and Wang 2021, Definition 1.7)

M is an (ϵ, δ)-differentially private interactive mechanism if, for every pair of adjacent datasets D and D', every adversary A, and every set X,

Differential Privacy as Approximate Trace Equivalence

We can use the same definition of interactive differential privacy as Vadhan, where View is formally defined as above.

Definition (Vadhan and Wang 2021, Definition 1.7)

M is an (ϵ, δ)-differentially private interactive mechanism if, for every pair of adjacent datasets D and D', every adversary A, and every set X,

$$\Pr\left[\mathrm{View}(A \parallel M(D)) \in X\right] \leq e^{\epsilon} \Pr\left[\mathrm{View}(A \parallel M(D')) \in X\right] + \delta\,.$$

Soundness

$$\frac{\Gamma \vdash P_1 \triangleright \Delta_1; (\epsilon_1, \delta_1) \quad \Gamma \vdash P_2 \triangleright \Delta_2; (\epsilon_2, \delta_2) \quad \Delta_1 \asymp \Delta_2}{\Gamma \vdash P_1 \parallel P_2 \triangleright \Delta_1 \circ \Delta_2; (\epsilon_1, \delta_1) \star (\epsilon_2, \delta_2)} \text{ [T-Conc]}$$

Lemma

The typing rule [T-Conc] is sound.

The view of a process in our language behaves in the same manner as the view of a party.

Soundness

$$\frac{\Gamma \vdash P_1 \triangleright \Delta_1; (\epsilon_1, \delta_1) \qquad \Gamma \vdash P_2 \triangleright \Delta_2; (\epsilon_2, \delta_2) \qquad \Delta_1 \asymp \Delta_2}{\Gamma \vdash P_1 \parallel P_2 \triangleright \Delta_1 \circ \Delta_2; (\epsilon_1, \delta_1) \star (\epsilon_2, \delta_2)} \text{ [T-Conc]}$$

Lemma

The typing rule [T-Conc] is sound.

The view of a process in our language behaves in the same manner as the view of a party.

Hence, as all other rules are sound, our main theorem follows.

Theorem (Soundess)

If $\Gamma \vdash M \triangleright \Delta; (\epsilon, \delta)$*, then M is an* (ϵ, δ)*-differentially private process.*

5. Conclusion

Summary

In this work, we have:

- introduced a process calculus similar to the π-calculus with session types that possesses good metatheoretical properties,

Summary

In this work, we have:

- introduced a process calculus similar to the π-calculus with session types that possesses good metatheoretical properties,
- reformulated interactive differential privacy in a formal operational way,

In this work, we have:

- introduced a process calculus similar to the π-calculus with session types that possesses good metatheoretical properties,
- reformulated interactive differential privacy in a formal operational way,
- defined typing rules for tracking interactive differential privacy, and

Summary

In this work, we have:

- introduced a process calculus similar to the π-calculus with session types that possesses good metatheoretical properties,
- reformulated interactive differential privacy in a formal operational way,
- defined typing rules for tracking interactive differential privacy, and
- provided examples, notably from Lyu (2022), demonstrating how privacy-preserving programs can be implemented in our calculus.

Future Work

- explore alternative methods for handling replication or random number generation,

Future Work

- explore alternative methods for handling replication or random number generation,
- define interactive differential privacy in terms of approximate bisimulation rather than approximate trace equivalence.

References I

- Dwork, Cynthia et al. (2006). "Calibrating Noise to Sensitivity in Private Data Analysis". In: *Theory of Cryptography*. Lecture Notes in Computer Science, pp. 265–284. DOI: 10.1007/11681878_14.
- Girard, Jean-Yves (1987). "Linear logic". In: *Theoretical Computer Science* 50.1, pp. 1–101. DOI: 10.1016/0304-3975(87)90045-4.
- Honda, Kohei, Vasco T. Vasconcelos, and Makoto Kubo (1998). "Language primitives and type discipline for structured communication-based programming". In: *Programming Languages and Systems*. Vol. 1381. Lecture Notes in Computer Science, pp. 122–138.
- Lyu, Xin (2022). "Composition theorems for interactive differential privacy". In: *NIPS'22: Proceedings of the 36th International Conference on Neural Information Processing Systems*, pp. 9700–8712. DOI: 10.5555/3600270.3600975.

References II

- Milner, Robin, Joachim Parrow, and David Walker (1992). "A calculus of mobile processes". In: *Information and Computation* 100.1, pp. 1–40. DOI: 10.1016/0890-5401(92)90008-4.
- Reed, Jason and Benjamin C. Pierce (2010). "Distance makes the types grow stronger". In: *ICFP'10: Proceedings of the 15th ACM SIGPLAN international conference on Functional programming*. DOI: 10.1145/1863543.1863568.
- Takeuchi, Kaku, Kohei Honda, and Makoto Kubo (1994). "An interaction-based language and its typing system". In: *PARLE'94 Parallel Architectures and Languages Europe*. Vol. 817. Lecture Notes in Computer Science, pp. 398–413. DOI: 10.1007/3-540-58184-7_118.

Vadhan, Salil and Tianhao Wang (2021). "Concurrent Composition of Differential Privacy". In: *Theory of Cryptography Conference*. Lecture Notes in Computer Science 13043, pp. 582–604. DOI: 10.1007/978-3-030-90453-1_20.

More Details on the Fuzz Language

Definition

A function f between two metric spaces (X, d_X) and (Y, d_Y) is s-sensitive if for all x and x' in X, we have $d_Y\big(f(x), f(x')\big) \leq s \cdot d_X(x, x')$.

More Details on the Fuzz Language

Definition

A function f between two metric spaces (X, d_X) and (Y, d_Y) is s-sensitive if for all x and x' in X, we have $d_Y\big(f(x), f(x')\big) \leq s \cdot d_X(x, x')$.

Types are interpreted as metric spaces:

- $[\![A \otimes B]\!] = [\![A]\!] \times [\![B]\!]$, and $[\![A \,\&\, B]\!] = [\![A]\!] \sqcup [\![B]\!]$,
- $[\![!_s A]\!] = \big(\pi_1([\![A]\!]), s \cdot \pi_2([\![A]\!])\big)$,
- $[\![\bigcirc_\epsilon A]\!] = (\mathrm{Dist}(A), d_\epsilon)$
- etc.

More Details on the Fuzz Language

Definition

A function f between two metric spaces (X, d_X) and (Y, d_Y) is s-sensitive if for all x and x' in X, we have $d_Y\big(f(x), f(x')\big) \leq s \cdot d_X(x, x')$.

Types are interpreted as metric spaces:

- $[\![A \otimes B]\!] = [\![A]\!] \times [\![B]\!]$, and $[\![A \,\&\, B]\!] = [\![A]\!] \sqcup [\![B]\!]$,
- $[\![!_s A]\!] = \big(\pi_1([\![A]\!]), s \cdot \pi_2([\![A]\!])\big)$,
- $[\![\bigcirc_\epsilon A]\!] = (\mathrm{Dist}(A), d_\epsilon)$
- etc.

Typing judgements have the form $[x_1 : A_1]_{s_1}, \ldots, [x_n : A_n]_{s_n} \vdash b : B$ and mean that $(x_1, \ldots, x_n) \mapsto [\![b]\!](x_1, \ldots, x_n)$ is a 1-sensitive function from $!_{s_1}[\![A_1]\!] \otimes \cdots \otimes !_{s_n}[\![A_n]\!]$ to $[\![B]\!]$.

Finite Replication and Recursive Processes

We permit finite process replication instead of recursive processes or arbitrary replication. This way, a process will never generate an infinite number of random numbers during its execution.

Finite Replication and Recursive Processes

We permit finite process replication instead of recursive processes or arbitrary replication. This way, a process will never generate an infinite number of random numbers during its execution.

Indeed, we aim to develop a formal framework for interactive differential privacy, rather than extending the existing notion.

- Vadhan and Wang (2021) generate binary strings before the interaction.
- Lyu (2022) explicitly bounds the number of interaction rounds.

$\rho \leftarrow \mathcal{L}(1/\epsilon)$;
for $i = 1, 2, \ldots, T$ **do**
  Receive the next query (f_i, τ_i);
  $\gamma_i \leftarrow \mathcal{L}(c/\epsilon)$;
  if $|f_i(X) - \tau_i| + \gamma_i \geq E + \rho$ **then**
    $v_i \leftarrow f(X) + \mathcal{L}(c/\epsilon)$;
    report (wrong, v_i);
    $t \leftarrow t + 1$;
    if $t = c$ **then halt**;
  else
    report pass
  end
end

Algorithm 1: Private Guess-and-Check Lyu 2022, Algorithm 1

```
SVT(c, E, N, D, k, a) =
  Lap(1/epsilon)?(rho);
  a.write 1;
  repeat N times
    k?(f, v);
    let t = a.read () in
    if t >= c then k![0] else
      Lap(c/epsilon)?(gamma);
      k![abs (f(D) - v) + gamma < E +
    end
    a.write (t + 1);
  end
```

Guess-and-Check (II)

Theorem

Given the environment $\Gamma = \{c : \mathrm{Nat}, E : \mathrm{Int}, N : \mathrm{Nat}, D : \mathrm{Data}, a : \mathrm{Cell}(\mathrm{Nat})\}$ *and the typing* $\Delta = \{k : *_N?\big((\mathrm{Data} \multimap \mathrm{Int}) \otimes \mathrm{Int}\big) . !\mathrm{Int} . \mathrm{end}\}$, *the following typing rule is sound:*

$$\frac{}{\Gamma \vdash \mathrm{SVT}(c, E, N, D, k, a) \triangleright \Delta; (3\epsilon, 0)} \text{ [T-SVT]} \tag{1}$$