Session Types for the Concurrent Composition of Interactive Differential Privacy

Victor Sannier[1] Patrick Baillot[1] Marco Gaboardi[2]

[1]CRIStAL Laboratory, Lille, France
[2]Boston University, MA, USA

2024 Days of the SCALP Working Group

1. Differential Privacy

Differentially Private Mechanisms

A private algorithm is one that does not allow anyone to deduce the presence or absence of any specific individual within the database based on its output.

Differentially Private Mechanisms

A private algorithm is one that does not allow anyone to deduce the presence or absence of any specific individual within the database based on its output.

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:

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

Differentially Private Mechanisms

A private algorithm is one that does not allow anyone to deduce the presence or absence of any specific individual within the database based on its output.

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:

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

Differentially Private Mechanisms

A private algorithm is one that does not allow anyone to deduce the presence or absence of any specific individual within the database based on its output.

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:

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

The parameters $(\epsilon, \delta) = (0, 1)$ provide no differential privacy guarantee.

Differentially Private Mechanisms

A private algorithm is one that does not allow anyone to deduce the presence or absence of any specific individual within the database based on its output.

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:

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

The parameters $(\epsilon, \delta) = (0, 1)$ provide no differential privacy guarantee.

A common strategy to ensure differential privacy is to add random noise to the output of an algorithm.

Composition of Differential Privacy

Differential privacy is a compositional property.

Theorem (McSherry 2009, Theorem 3)

Let M_i be mechanisms, each providing $(\epsilon_i, 0)$-differential privacy. The sequence of the M_i provides $(\sum_i \epsilon_i, 0)$-differential privacy.

Many other results can be found in the literature for $\delta \neq 0$, such as the advanced composition theorem.

Composition of Differential Privacy

Differential privacy is a compositional property.

Theorem (McSherry 2009, Theorem 3)

Let M_i be mechanisms, each providing $(\epsilon_i, 0)$-differential privacy. The sequence of the M_i provides $(\sum_i \epsilon_i, 0)$-differential privacy.

Many other results can be found in the literature for $\delta \neq 0$, such as the advanced composition theorem.

As a result, one can ensure that an algorithm is differentially private by individually verifying its components.

The Fuzz Calculus

Reed and Pierce have introduced a calculus inspired by linear logic Girard (1987) to track the sensitivity and, consequently, the privacy of programs written in a functional programming language.

The Fuzz Calculus

Reed and Pierce have introduced a calculus inspired by linear logic Girard (1987) to track the sensitivity and, consequently, the privacy of programs written in a functional programming language.

Theorem (Reed and Pierce 2010, Corollary 4.3)

Closed programs e such that $\vdash e : A \multimap \bigcirc_\epsilon B$ *are an* ϵ*-differentially private function from* A *to* B*.*

(In the equation above, $\bigcirc_\epsilon$ is a probability monad equipped with an appropriate distance.)

Composition Theorems as Typing Rules

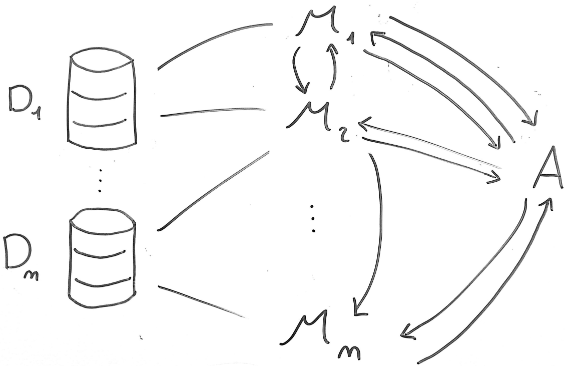
The previous composition theorem can be obtained in Fuzz through the application of typing rules for sensitivity to the probability monad (Reed and Pierce 2010, Section 5).

For example, below is the rule for introducing the tensor product:

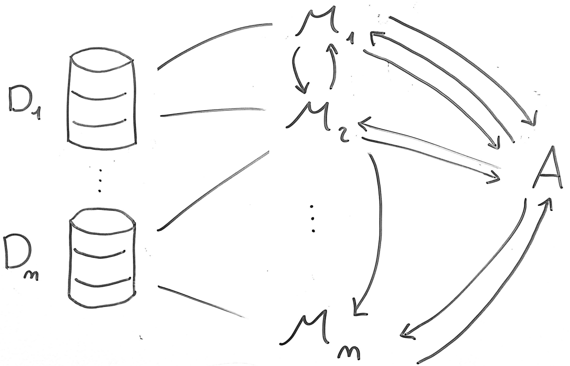
$$\frac{[x : A]_{s_1} \vdash b : B \qquad [x : A]_{s_2} \vdash c : C}{[x : A]_{s_1+s_2} \vdash (b, c) : B \otimes C} \; [\text{T-}\otimes\text{I}]$$

2. Interactive Differential Privacy

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



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?

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* $\mathrm{View}(A \parallel M)$ 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',

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* $\mathrm{View}(A \parallel M)$ 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,

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 $(M_0, \ldots, M_1)$ *are each* (ϵ, δ)*-differentially private, then their concurrent composition* $\mathrm{ConComp}(M_0, \ldots, M_1)$ *is* $(k\epsilon, \frac{e^{k\epsilon-1}}{e^{\epsilon-1}}\delta)$*-DP.*

Concurrent Composition of Interactive Differential Privacy

Theorem (Vadhan and Wang 2021, Theorem 1.8)

If interactive mechanisms $(M_0, \ldots, M_1)$ *are each* (ϵ, δ)*-differentially private, then their concurrent composition* $\mathrm{ConComp}(M_0, \ldots, M_1)$ *is* $(k\epsilon, \frac{e^{k\epsilon}-1}{e^{\epsilon}-1}\delta)$*-DP.*

In the same way as for (centralised) differential privacy, we can prove that an interactive process is differentially private by proving that each of its components is.

Theorem (Vadhan and Wang 2021, Theorem 1.8)

If interactive mechanisms $(M_0, \ldots, M_1)$ are each (ϵ, δ)-differentially private, then their concurrent composition $\mathrm{ConComp}(M_0, \ldots, M_1)$ *is* $\left(k\epsilon, \frac{e^{k\epsilon-1}}{e^{\epsilon-1}}\delta\right)$*-DP.*

In the same way as for (centralised) differential privacy, we can prove that an interactive process is differentially private by proving that each of its components is.

centralised DP : Fuzz :: interactive DP : ?

3. Process Calculi and Session Types

The π-calculus

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

The π-calculus

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

The π-calculus was introduced by Milner, Parrow, and Walker (1992)

- 0 is the null process,

The π-calculus

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

The π-calculus was introduced by Milner, Parrow, and Walker (1992)

- 0 is the null process,
- **if** e **then** P **else** Q behaves as P if e evaluates to $\top$, and as Q otherwise,

The π-calculus

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

The π-calculus was introduced by Milner, Parrow, and Walker (1992)

- 0 is the null process,
- **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 π-calculus

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

The π-calculus was introduced by Milner, Parrow, and Walker (1992)

- 0 is the null process,
- **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,
- ...

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

The π-calculus was introduced by Milner, Parrow, and Walker (1992)

- 0 is the null process,
- **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,
- $\ldots$

Example

$(k![1] \,.\, k?[x] \,.\, \ldots) \parallel (k?[x] \,.\, k![x + x] \,.\, \ldots)$

Session Types

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

Session Types

- 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

- 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 $\Gamma \vdash P \triangleright \Delta$.

Session Types

- 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 $\Gamma \vdash P \triangleright \Delta$.

Example

The type $(\alpha, \bar{\alpha})$, where $\alpha = ?\mathsf{Int} \,.\, !\mathsf{Bool} \,.\, \mathsf{end}$, can be given to the session between a process that sends a number and a process that determines if the number is even.

Session Types

- 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 $\Gamma \vdash P \triangleright \Delta$.

Example

The type $(\alpha, \bar{\alpha})$, where $\alpha = ?\mathsf{Int}\,.\,!\mathsf{Bool}\,.\,\mathsf{end}$, can be given to the session between a process that sends a number and a process that determines if the number is even.

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:

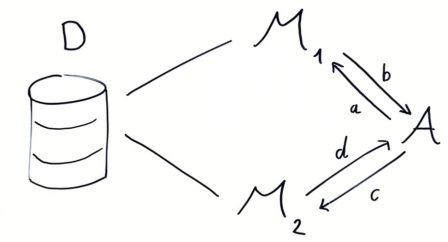
- $\mathbf{Lap}_b(x) \cdot P$ to sample a random number from the (discrete) Laplace distribution with parameter b and continue according to P,

New Constructs for the π-calculus

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

- $\mathbf{Lap}_b(x) \cdot P$ 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 substitute for recursive processes).

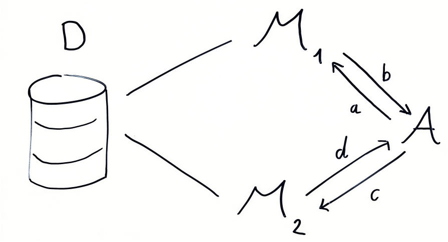
Example of a Concurrent Composition



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

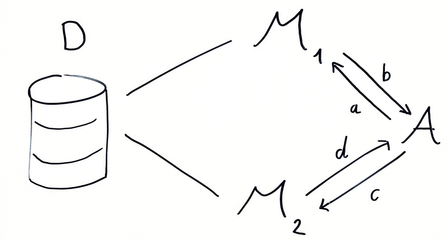


Let M_1 and M_2 be two differentially private mechanisms.

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

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

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 $\alpha = ?(db \multimap \mathsf{Num})\,.\,!\mathsf{Num}\,.\,\mathsf{end}$,

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

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

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

Examples of Typing Rules

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

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

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

Examples of Reduction Rules

We provide an operational semantics using a fully probabilistic labelled transition system (FPLTS) with binary trees as labels.

Examples of Reduction Rules

We provide an operational semantics using a fully probabilistic labelled transition system (FPLTS) with binary trees as labels.

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

Examples of Reduction Rules

We provide an operational semantics using a fully probabilistic labelled transition system (FPLTS) with binary trees as labels.

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

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

Examples of Reduction Rules

We provide an operational semantics using a fully probabilistic labelled transition system (FPLTS) with binary trees as labels.

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

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

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

The last reduction rule is the only non-deterministic one that does not simply transfer the probability from the hypothesis to the conclusion.

Differential Privacy as Approximate Trace Equivalence

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 $\Pr[\mathrm{Trace}(P) = t_i] = p_i$.

Differential Privacy as Approximate Trace Equivalence

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

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: $\mathrm{View}(A \parallel M) = \mathrm{Left}\big(\mathrm{Trace}(A \parallel M)\big)$.

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

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: $\mathrm{View}(A \parallel M) = \mathrm{Left}\big(\mathrm{Trace}(A \parallel M)\big)$.

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

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: $\mathrm{View}(A \parallel M) = \mathrm{Left}\big(\mathrm{Trace}(A \parallel M)\big)$.

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

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: $\mathrm{View}(A \parallel M) = \mathrm{Left}\big(\mathrm{Trace}(A \parallel M)\big)$.

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,

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

Soundness

Lemma

The typing rule [T-Conc] is sound.

We can adapt the proof given by Vadhan and Wang 2021 as the view of a process in our language behaves in the same manner as the view of a party.

Soundness

Lemma

The typing rule [T-Conc] is sound.

We can adapt the proof given by Vadhan and Wang 2021 as the view of a process in our language behaves in the same manner as the view of a party.

Theorem

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 sessions that possesses good metatheoretical properties,

Summary

In this work, we have:

- introduced a process calculus similar to the π-calculus with sessions that possesses good metatheoretical properties,
- syntactically defined interactive differential privacy,

Summary

In this work, we have:

- introduced a process calculus similar to the π-calculus with sessions that possesses good metatheoretical properties,
- syntactically defined interactive differential privacy,
- 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 sessions that possesses good metatheoretical properties,
- syntactically defined interactive differential privacy,
- defined typing rules for tracking interactive differential privacy, and
- provided examples, notably from Lyu (2022), demonstrating how private programs can be implemented within our calculus.

Future Work

Future work may include:

- studying local differential privacy using our process calculus,

Future Work

Future work may include:

- studying local differential privacy using our process calculus,
- exploring alternative methods for handling replication or random number generation,

Future Work

Future work may include:

- studying local differential privacy using our process calculus,
- exploring alternative methods for handling replication or random number generation,
- defining 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.

McSherry, Frank D. (2009). “Privacy Integrated Queries”. In: *SIGMOD’09: Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pp. 19–30. DOI: 10.1145/1559845.1559850.

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 $d_Y(f(x), f(x')) \leq s \cdot d_X(x, x')$.

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 $d_Y(f(x), f(x')) \leq s \cdot d_X(x, x')$.

Types are interpreted as metric spaces:

- $[\![A \otimes B]\!] = [\![A]\!] \times [\![B]\!]$, and $[\![A \mathbin{\&} B]\!] = [\![A]\!] \sqcup [\![B]\!]$,
- $[\![!_s A]\!] = (\pi_1([\![A]\!]), s \cdot \pi_2([\![A]\!]))$,
- $[\![\bigcirc_\epsilon A]\!] = (\mathrm{Dist}(A), d_\epsilon)$
- 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 $d_Y(f(x), f(x')) \leq s \cdot d_X(x, x')$.

Types are interpreted as metric spaces:

- $[\![A \otimes B]\!] = [\![A]\!] \times [\![B]\!]$, and $[\![A \mathbin{\&} B]\!] = [\![A]\!] \sqcup [\![B]\!]$,
- $[\![!_s A]\!] = (\pi_1([\![A]\!]), s \cdot \pi_2([\![A]\!]))$,
- $[\![\bigcirc_\epsilon A]\!] = (\mathrm{Dist}(A), d_\epsilon)$
- 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 $(x_1, \ldots, x_n) \mapsto [\![b]\!](x_1, \ldots, b_n)$ 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.