差分プライバシーを対象とした検証ツールの評価と比較

山本 充子[1,a)]　中林 美郷[1,b)]

概要：差分プライバシーはデータに含まれる個人のプライバシーを保護する安全性指標として注目されている。しかし、差分プライバシーを実現するためのアルゴリズムの設計や実装は複雑であり、誤りが入りやすい。そこで、近年では差分プライバシーを対象とした自動・半自動の検証ツールの研究がさかんに行われている。本研究では、差分プライバシーを対象とした既存の検証ツールを調査し、各ツールが検証可能な要件、対象とするプログラムやアルゴリズムの範囲、検証アプローチ、実用性に焦点を当てて評価と比較を行う。また、検証ツールを差分プライバシーに適用する際の課題や、各ツールが有する制限を明らかにし、それらを体系的にまとめる。

Evaluation and Comparison of Verification Tools for Differential Privacy

Juko Yamamoto[1,a)]　Misato Nakabayashi[1,b)]

Abstract: Differential privacy has attracted significant attention as a security metric for protecting the privacy of individuals contained in datasets. However, the design and implementation of algorithms that achieve differential privacy are complex and prone to errors. In recent years, research on automated and semi-automated verification tools for differential privacy has been actively conducted. In this study, we survey existing verification tools for differential privacy and evaluate and compare them with a focus on the verifiable requirements, the scope of target programs and algorithms, verification approaches, and practicality. Furthermore, we identify the challenges in applying verification tools to differential privacy, clarify the limitations of each tool, and systematically summarize these findings.

1. はじめに

近年、IT 技術の発展に伴いデータ利活用の需要が高まっている。その一方で、データ解析を行った結果からデータに含まれる個人の情報が漏洩する危険性があることから、プライバシーを保護するための技術の研究が進んでいる。こうした背景から 2006 年に差分プライバシー [13, 15] が提案され、研究者の中ではプライバシー保護の指標におけるデファクトスタンダードとして考えられている。差分プライバシーは、プライバシーを保護するためのアルゴリズムがデータセットに含まれる個人をどの程度守れているかを評価する安全性指標である。データ解析アルゴリズムが差分プライバシーを満たしていることは数学的な証明に基づいて保証される。

差分プライバシーは合成則 [19] や事後処理則 [15] という性質が成り立ち、プライバシーを保護するアルゴリズムを組み合わせた場合のプライバシー保護性も評価することができる。しかし、それに伴い複雑にアルゴリズムが組み合わさることでプライバシーが保護されていることの判定がより難しくなる。また、差分プライバシーの概念自体への理解が難しいことから間違った設計や実装を招きやすく、実際に誤りも指摘されている [11, 12, 14, 22, 30, 40]。一見類似したアルゴリズムでも、保護すべき処理に対して、最終結果にノイズを加える、処理の過程にノイズを加える、もしくは入力データ自体にノイズを加えるなどと、同じ安全性を保証するノイズの加え方は複数ある。それらの方法は多岐にわたり、アルゴリズム全体の流れを把握していないとどこにノイズを入れるべきか設計者・実装者本人ですらわからなくなってしまう。また、Near らによるレポー

[1] NTT 社会情報研究所
NTT Social Informatics Laboratories
[a)] juko.yamamoto@ntt.com
[b)] misato.nakabayashi@ntt.com

ト [25] では、出力から差分プライバシーが保証されているかどうかを判断しづらいことなどから実装の誤りが見つかりづらいことも指摘されている。

そこで、アルゴリズムの仕様および実装が差分プライバシーを満たしていることを検証ツールを用いて保証するという取り組みが行われている。検証ツールを用いることでアルゴリズムの仕様および実装が差分プライバシを満たしているかどうかを機械的に確認することができるため、複雑なアルゴリズムや見落とされがちなミスであっても検証することができる。しかし、現状差分プライバシーを対象とする検証ツールは多数公開されているものの、実アルゴリズムへの応用は多くない。

そこで本研究では、差分プライバシーに対する検証ツールの活用を促進し、より強固なプライバシ保護と安全なデータ利活用が両立する社会の実現に資することを目的とし、差分プライバシーを対象とした検証ツールの評価と比較を行う。特に、各ツールが検証可能な要件、検証アプローチ、実用性に焦点を当てて評価と比較を行う。さらに、形式検証ツールを差分プライバシーに適用する際の課題や各ツールが有する制限を明らかにし、それらを体系的にまとめることで今後の研究や利用を促進することを目指す。

2. 準備

この章では差分プライバシーと検証ツールの一つのアプローチである形式検証技術について紹介する。

2.1 差分プライバシー

差分プライバシーとは、プライバシーを保護するためのアルゴリズムがデータセットに含まれる個人をどの程度守れているかを評価する安全性指標である [13, 15]。厳密には、下記のように定義される。

定義 2.1 (隣接データセット). $\mathcal{D}$ をデータセットの候補全体の集合とする。データセット $D \in \mathcal{D}$ に対して、D のレコードを 1 人分だけ変えたデータセット D' を D の**隣接データセット**と呼ぶ。データセット $D \in \mathcal{D}$ に対して、D の隣接データセット全体の集合を $N(D)$ とおく。

定義 2.2 ((ϵ, δ)-差分プライバシー [15]). $\mathcal{M} : \mathcal{D} \to \mathcal{Y}$ をランダム化アルゴリズムとする。任意の隣接データセット $D, D' \in \mathcal{D}$ と任意の出力レンジ $O \subset \mathcal{Y}$ に対して、次が成り立つとき、アルゴリズム $\mathcal{M}$ は (ϵ, δ)-**差分プライバシー**（(ϵ, δ)-**DP**）を満たすという。

$$Pr[\mathcal{M}(D) \in O] \leq e^{\epsilon} \cdot Pr[\mathcal{M}(D') \in O] + \delta.$$

ここで、ϵ はプライバシー損失予算であり、各データセットを入力した際の出力の分布の近さを表す。

直感的には、差分プライバシーは個々のレコードがデータセット内にあるかどうかに関係なく、$\mathcal{M}$ の公開された出力分布がほぼ同じであることを保証する。差分プライバシーの定義は、隣接データセットに対して出力分布がどの程度変化し得るかを制限するものである。具体的な $\mathcal{M}$ を設計する際には、関数の出力が入力データの変化によって最大でどれほど変わるかを評価する必要がある。この最大変化量を定量化する指標が敏感度である。クエリに対して敏感度が求まると、その値と採用するメカニズムに応じて必要なノイズの大きさを決定できる。

定義 2.3 (敏感度). 関数 $q : \mathcal{D} \to \mathbb{R}^m$ に対して、次の値をクエリの**敏感度**と呼ぶ。

$$\Delta := \sup_{D \in \mathcal{D}, D' \in N(D)} ||q(D) - q(D')||_1$$

ここで $||\cdot||_1$ は L_1 ノルムを表す。

2.2 形式検証

形式検証とは、対象となるシステムと要件を形式化し、システムがその要件を満たすことを数学的に証明する手法である。シミュレーションによる検証では限られた入力やケースのみに対して検証結果が得られるのに対し、形式検証では与えられたモデルが取りうる全ての実行を検証できるため、システムがあらゆる状況下で要件を満たすことを検証することができる。

近年では差分プライバシーの検証に形式検証を応用する研究も多く行われており、大きく次の二つに分類できる。一つ目は型システムに基づく手法であり、アルゴリズムの各変数や関数の型に性質を付与し、型推論や型検査を用いてコンパイル時に安全性を保証する。差分プライバシーの検証では敏感度やプライバシー予算を型に埋め込み、演算によるそれらの消費を計算する。二つ目はプログラムロジックに基づく手法である。この手法では、差分プライバシーを論理体系の中で形式化し、モデル検査や定理証明系の技法を用いてアルゴリズムが差分プライバシーを満たすことを自動または対話的に証明する。

3. 差分プライバシーの自動検証ツール

まず、動的に解析するツールとして、PINQ [23] がある。これは C#に基づく LINQ 風のクエリ言語を拡張し、非専門家でも SQL ライクに差分プライベートな処理を記述できるようにした最初期のシステムである。その拡張版である wPINQ [27] は、データに重み付けを導入することでクエリ実行時の敏感度を一定に保ち、より一般的な結合処理を可能にしている。さらに、分散処理基盤に差分プライバシーを組み込んだ Airavat [29]、専用 DBMS と統合されたアーキテクチャを持つ Chorus [18]、Python 環境で実行可能な DDuo [3] なども提案されている.

これに対して、型システムを拡張して静的に検証を行うアプローチも広く研究されてきた。代表的な例は Fuzz [28] であり、線形型を利用してプログラムの敏感度を追跡し、差分プライバシーを保証する先駆的な言語である。その後、依存型を導入してより柔軟な敏感度解析を可能にした DFuzz [17]、プライバシーフィルタを導入して実行過程

に応じた動的制御を可能にした Adaptive Fuzz [37]、三層（値・分布・関係）のプログラム論理に基づく Fuzzi [39]、敏感度とプライバシーコストを分離した二重型システムを採用する Duet [26] などへと発展している。また、andomness alignment に基づく relation 型を用いる LightDP [38] や、shadow execution によって複雑なアルゴリズムにも対応する ShadowDP [35] もこの系列に位置づけられる。

プログラムロジックを基盤にした手法では、例えば CertiPriv [7] は Rocq [1] 上で差分プライバシーを証明可能なフレームワークであり、ラプラスメカニズム自体の正当性を証明できるほど高い表現力を備えている。HOARE^2 [6] は Refinement 型を用いて確率的性質を記述可能にし、さらにその拡張である PrivInfer [5] はベイジアン推論アルゴリズムの検証にも対応する。こうしたプログラムロジック系のツールは一般に表現力が高い。型システムに基づく検証は自動化に強みを持つ一方で、(ϵ, δ)-DP などの高度なプライバシーの概念の扱いには制約がある。それに対しプログラムロジック系のツールでは差分プライバシーの性質を関係論理等を用いて直接的に表現するため、それらの性質も扱うことができる。複雑な確率的アルゴリズムに対して厳密な証明を可能にする一方で、専門的な知識や多くのアノテーション作業を必要とする傾向がある。

加えて、近年は他のアプローチも現れている。Haskell の型システムを利用して敏感度解析を行い、Haskell ライブラリとして提供可能な DPella [20]、統計的検定や入力生成に基づいて反例を探索する StatDP [14]、randomness alignment を利用して証明と反例検出を統合する CheckDP [34]、識別モデルに基づきプライバシー違反を検出する DP-Sniper [9]、さらにプログラム合成技術を取り入れて新たに差分プライベートな機構を自動生成する DPGen [36] などが提案され、研究の幅は大きく広がっている。

4. 計算能力の評価

差分プライバシーに対する検証手法は多様に提案されているが、同じ検証という枠組みの中でも保証する性質・採用する手法などが大きく異なる。そこで、前章で挙げた各ツールについて、その特徴を体系的に整理するために次の 4 項目を評価基準に設定し、比較を行った。

検証要件　差分プライバシーには複数の形式的定義が存在する。最も基本的な定義が ϵ-DP であり、それを緩和し確率 δ 以下でこの制約が破られる可能性を許容したのが (ϵ, δ)-DP である。さらに、プライバシー損失の合成解析を容易にし、よりタイトな評価を可能にする定義として zCDP (Zero-Concentrated Differential Privacy) [10] や RDP (Rényi Differential Privacy) [24] が提案されている。既存の検証ツールがこれら各種定義のいずれを検証可能かを評価した。また、2 章で見た通り、差分プライバシーを満たすためには、入力データベース間の差異が関数の出力にどの程度影響するかを測る敏感度の値が重要である。この敏感度をシステム内で形式的に推論可能かを評価した。

検証レベル　既存の検証手法には、内部をブラックボックス化してより自動的に結果を得られるように設計されたものと、厳密な保証のために詳細な証明の記述を利用者に求めるものがある。本研究では前者の手法をブラックボックス、後者をホワイトボックスとしてこの点を評価の基準とした。

アプローチの種類　実行前にコードや設計を解析する静的検証と、実際に動作させて確認する動的検証がある。静的検証は早期発見・低コスト修正に有効で、動的検証は実行環境での挙動確認に有効である。また、目的・粒度に応じて多様な検証アプローチが存在する。ここでは検証アプローチとして保証のタイミングと、静的検証に関しては検証手法についても整理した。

機能　既存の検証手法は、与えられたメカニズムの正しさを証明する手法と、誤りを示す反例を探索する手法が存在する。前者はメカニズムが正しい場合にはその正当性を強力に保証できる一方で、誤りがある場合には単に証明が成立しないだけであり、どのように誤っているのかを具体的に示すことはできない場合がある。後者は違反を効果的に発見できるが、正しい場合には保証を与えないため不十分である。さらに、両者を統合し、正しい場合には証明を、誤っている場合には反例を提示することで双方向から検証を行う枠組みも提案されてきた。そこで、各手法を検証特化型・反例検出型・ハイブリッド型の 3 つに分類した。

4.1 各ツールの計算能力の評価

評価項目に基づいた結果を表 1 にまとめる。

4.1.1 検証要件

表から分かる通り (ϵ, δ)-DP や RDP や zCDP のような新しい定義に対応している手法は限られている。敏感度解析は型システムを利用する手法ではほとんどの場合で可能である。しかし、線形型を用いる Fuzz 系では、$(\epsilon, 0)$-DP に制限されている。これは、$(\epsilon, 0)$-DP では敏感度が 2 倍になれば ϵ も 2 倍になるが、(ϵ, δ)-DP では δ が同様にスケーリングする保証はないため、(ϵ, δ)-DP への拡張は困難であることによる。この問題に対し、Duet は敏感度解析とプライバシー解析の型システムを分離することで (ϵ, δ)-DP の証明を可能とした。また、CertiPriv のようなプログラム論理を用いる方法では、apRHL（approximate probabilistic Relational Hoare Logic）という論理体系を導入することで、(ϵ, δ)-DP を形式的に証明可能にした。具体的には、出力分布間の差を測る尺度として α-distance を導入し、状態間の関係を確率分布に拡張して比較する (α, δ)-lifting を定

		検証要件		検証レベル	アプローチの種類		機能
		プライバシー	感度解析		タイミング	検証手法	
PINQ	[23]	○	○	●	○	-	○
↳ wPINQ	[27]	○	○	●	○	-	○
Airavat	[29]	◑	○	●	○	-	○
Chorus	[18]	●	●	●	◑	抽象解釈	○
DDuo	[3]	●	○	●	○	-	○
Fuzz	[28]	○	●	●	●	線形型システム	○
↳ DFuzz	[17]	○	●	●	●	線形依存型システム	○
↳ AdaptiveFuzz	[37]	◑	●	●	◑	線形型システム + プライバシーフィルタ	○
↳ Fuzzi	[39]	●†	●	○	●	線形型システム	○
Duet	[26]	●	●	●	●	線形依存型システム	○
Solo	[2]	●	●	●	●	Type-and-Effect System	○
Spar	[21]	●†	●	●	●	Parametric Polymorphism	○
GSoul	[4]	○	●	●	◑	Gradual Type System	○
CertiPriv	[7]	◑	●	○	●	apRHL	○
HOARe^2	[6]	●†	●	○	●	Relational Refinement Type System	○
Privinfer	[5]	◑	●	○	●	Relational Refinement Type System	○
LightDP	[38]	●†	○	○	●	Relational Type System	○
ShadowDP	[35]	○	●	○	●	Flow-Sensitive Type System	○
DPella	[20]	◑	●	●	●	Symbolic Execution	○
DP-finder	[8]	○	○	●	○	-	●
StatDP	[14]	○	○	●	○	-	●
CheckDP	[34]	○	○	●	●	Flow-Sensitive Type System	◑
CRSE	[16]	◑	●	○	●	確率的カップリング	◑
DP-Sniper	[9]	○	○	●	○	-	●
DPGen	[36]	○	○	●	◑	Flow-Sensitive Type System	プログラム合成

プライバシー	感度解析	検証レベル	タイミング	機能
●：ϵ-DP/(ϵ,δ)-DP/zCDP/tCDP/RDP	●：感度解析可能	●：ブラックボックス	●：実行前検証	●：反例検出
◑：ϵ-DP/(ϵ,δ)-DP	○：サポートなし	○：ホワイトボックス	◑：ハイブリッド	◑：検証 + 反例検出
○：ϵ-DP			○：実行時検証	○：検証特化

†･･･ 論文では明記されていないが、拡張によって可能

表 1 各検証ツールの計算能力

義することで、(ϵ,δ)-DP の検証を可能にしている。

4.1.2 検証レベル

型システムを利用するアプローチでは、異なるメカニズムを安全に組み合わせることに焦点が置かれており、形式的に誤りを防ぐ枠組みを提供している。これに対して、定理証明系を用いる方法はメカニズムをブラックボックス化せずに厳密な証明を可能にする。代表的な例として CertiPriv が挙げられ、これは高い表現力を有し、(ϵ,δ)-DP に対応できる。

4.1.3 アプローチの種類

表から型システムやプログラム論理を用いる手法のほとんどが実行前に解析を行っていることが分かる。これらの手法は理論的な厳密性を確保できる一方で、表現力や利用のしやすさに制約を伴いやすい。PINQ や DDuo のように実行時に解析を行う手法は、プログラムの実行に応じて敏感度やプライバシーバジェットを計算できるため柔軟性が高く、予算を効率的に活用できる場合があるが、実行を伴うため形式的な保証を与えるのは難しい。Adaptive Fuzz や GSoul のように両者を組み合わせたハイブリッド型の手法も提案されており、これらは用途によって柔軟性と保証性のバランスを取る選択肢となり得る。

4.1.4 機能

表からは、多くのツールが検証特化型として設計されていることが分かる。Fuzz 系や CertiPriv、HOARe^2 などはその代表例であり、形式的な証明を通じて正当性の保証を与える方向性が主流である。他方で、DP-finder や StatDP、DP-Sniper のように反例検出に特化したツールは限られているものの、既存アルゴリズムの不備を直接的に明らかにする実用的な役割を担っている。また、CheckDP や CRSE のように両機能を備えたツールも存在し、誤りの発見と正しさの保証を両立しようとする試みが見られる。さらに、DPGen のようにプログラム合成までを視野に入れる研究も一部にあり、差分プライバシーの検証にとどまらず設計支援へと展開する方向性を示している。

4.2 既存ツールの計算能力に対する課題

差分プライバシーの検証では様々なアプローチが存在し、それぞれに特有の限界がある。まず、(ϵ,δ)-DP の厳密な証明を与えることができるツールは未だ多くない。また、表に整理しきれなかった観点として検証可能な対象の範囲がある。例えば型システムに基づくアプローチの中でも行列演算や勾配計算といった機械学習に不可欠な操作を直接サ

ポートできるのは Duet や Fuzzi などのごく少数に限られる。一方で、Sparse Vector Technique（SVT）の実装を形式的に検証できるかどうかも重要な相違点であり、これを扱えるのは LightDP、shadowDP、CheckDP、DPGen など一部のツールに留まっている。[41] では、固有値を求める関数や Bingham 分布を用いた指数メカニズムの型規則を追加して Duet を拡張しており、対象の範囲の拡張を行うこともできる。

検証手法の自動化と多様な種類のアルゴリズムを扱えるという柔軟性の両立は難しい課題である。型システムを利用した方法はある程度の自動化を実現しているが、その一方で柔軟性には制約があり、例えば SVT のようなアルゴリズムを正しく扱うことは難しい。これに対して、確率的プログラムやモナドの知識を前提とする手法も存在し、これらはより厳密かつ柔軟な証明を可能にするが、利用の難しさが障壁となっている。このような両者の間を埋める試みとして、Fuzzi のように型システム的な自動化とプログラム論理に基づく手動証明を組み合わせることで、中間的な立場から検証を支援しようとする試みも提案されている。したがって、いずれの手法も一長一短であり、適用対象や利用者の前提知識に応じて使い分ける必要がある。

5. 実用性と運用性の評価

本研究では、既存の差分プライバシーの検証ツールを比較・整理するために、入力言語・自動化レベル・オープンソース性・想定ユーザ層の 4 つを評価軸として設定した。まず「入力言語」は、ツールの利用対象や想定ユーザ層を大きく規定する要素であり、実務システムとの統合可能性や研究用途における形式的検証の容易さに直結するため重要である。「自動化レベル」は、ユーザがどの程度の知識や介入なしに差分プライバシーを確保できるかを示す指標であり、実運用への導入可能性を測る観点となる。「オープンソース性」は、他者による再現性の検証や、発展的な応用を可能にするかどうかを判断する観点から、実装の公開有無を評価対象とした。「想定ユーザ層」は、ツールの導入・活用におけるハードルがユーザの専門性に大きく依存することから、技術的背景を持たない実務者でも利用可能であるかを判断するための指標として設けた。これは、論文内で非専門家向けを想定したツールという記述の有無で評価した。これら 4 つの評価軸を組み合わせることで、各ツールの設計思想や実用上の特性を多角的かつバランスよく把握できると考える。また、ツールの典型的な利用フローを示すために SQL ベースのツールである Chorus と Python ベースのツールである DDuo、型システムベースのツールである Duet の代表的なシナリオに基づく入出力例を示す。

5.1 各ツールの実用性と運用性の評価

表 2 は、前章の 4 つの観点で横断評価した結果である。

		入力言語	自動化レベル	オープンソース性	想定ユーザ層
PINQ	[23]	C#	●	●	●
↳ wPINQ	[27]	C#	●	○	●
Airavat	[29]	Java	●	●	●
Chorus	[18]	SQL	●	●	●
DDuo	[3]	Python	●	●	●
Fuzz	[28]	独自言語	◑	●	○
↳ DFuzz	[17]	独自言語	◑	●	○
↳ AdaptiveFuzz	[37]	独自言語	◑	●	○
↳ Fuzzi	[39]	独自言語	◑	●	●
Duet	[26]	独自言語	●	●	○
Solo	[2]	Haskell	●	●	○
Spar	[21]	Haskell	●	●	●
GSoul	[4]	独自言語	◑	●	○
CertiPriv	[7]	Rocq	○	●	○
HOARe^2	[6]	独自言語	◑	●	○
Privinfer	[5]	独自言語	◑	○	○
LightDP	[38]	Python	◑	●	●
ShadowDP	[35]	C	◑	●	●
DPella	[20]	Haskell	●	○	●
DP-finder	[8]	Python	●	●	●
StatDP	[14]	Python	●	●	●
CheckDP	[34]	C	●	●	●
CRSE	[16]	独自言語	◑	○	○
DP-Sniper	[9]	Python	◑	●	●
DPGen	[36]	Python	●	●	●

自動化レベル：●：仕様入力のみ　◑：仕様＋追加情報　○：証明を手動記述
オープンソース性：●：公開　○：非公開 or リンク切れ
想定ユーザ層：●：非専門家　○：専門家

表 2　各検証ツールの実用性評価

まず、Fuzz や DFuzz、Duet のような型システムに基づく手法の多くは独自言語を採用している。これは、差分プライバシーの検証に必要とされる線形型や依存型といった特殊な型システムが主流の汎用言語には備わっていないためである。この設計は普及の妨げとなる側面もある。汎用言語での検証は、関数感度の注釈や再帰・分岐の扱いが難しい点が課題とされてきたが、Solo は敏感度をデータソース単位で管理し、Haskell の型レベル多相性や型推論を活用することで、専用言語を用いずとも差分プライバシーの検証を可能にする新しい方向性を示している。

一方で、動的解析を基盤とするツールの多くは Python などの汎用言語を入力として受け付けるが、特定の構文しか利用できない場合もある。加えて、これらの手法は仕様を記述するだけで検証が可能なものが多いのに対し、型システムを利用するアプローチでは追加情報の記述が求められる場合が多い。より厳密な証明を目指すほど、ユーザに要求される注釈や補助情報が増える傾向がある。

オープンソース性では、多くのツールは GitHub 上にオープンソースとして公開されているが、その多くは論文公開時点でのリリースにとどまり、継続的に更新されていないものが大半である。中にはリンクが失われて利用できないものや、CertiPriv のように公式には公開されず、第三

者が過去のコードを保管しているに過ぎないものもあった。

想定されるユーザ層については、動的解析を基盤とするツールは非専門家でも利用できるよう設計されている場合が多い。型システムを利用するツールでも近年は非専門家を意識した設計が見られるが、実際には形式検証に特有の知識を必要とする追加情報の入力が求められる場合も少なくない。また、Duet や Solo のように明示的に非専門家向けとされてはいないものの、比較的利用者の負担が少なく、非専門家にも開かれた設計となっている例もある。

5.2 ツールの入出力例

ツールの典型的な利用フローを示すため、代表的なシナリオに基づく入出力例を以下に示す。ここでは、差分プライバシー付きクエリ実行ツールを想定し、SQL ベースのツールと Python ベースのツールそれぞれの例を示す。

5.2.1 SQL ベースのツール（例: Chorus [31]）

Chorus では、通常の SQL クエリを入力として与え、その実行結果に対して差分プライバシーメカニズムを適用できる。以下の例では、`orders` テーブルから特定の製品（`product_id = 1`）に対する `order_cost` の合計を計算し、ラプラスメカニズムを用いてノイズを付加している。

```
val database = Schema.getDatabase("test")
val config = new RewriterConfig(database)

val query1 = "SELECT SUM(order_cost) FROM orders
      WHERE product_id = 1"
val root1 = QueryParser.parseToRelTree(query1, database)

val accountant = new EpsilonCompositionAccountant()
val r1 = new LaplaceMechClipping(1.0, 0, 10, root1, config).
      execute(accountant)

println("Sum query result: " + r1)
println("Total privacy cost: " + accountant.getTotalCost())
```

実行結果は次のようになる。合計値にはノイズが加えられており、さらに `accountant` によって総プライバシーコスト（この場合は $\epsilon = 1.0$）が明示的に記録されている。

```
Sum query result: List(Row(List(1.0272609898682492)))
Total privacy cost: EpsilonDPCost(1.0)
```

上記の例は Scala による実装だが、実際に利用者が入力するのは SQL のみであり、Chorus は内部でクエリの書き換え・解析・事後処理を行い、既存の DBMS と協調して差分プライバシーをスケーラブルに実現する特徴を持つ。

5.2.2 Python ベースのツール（例: DDuo [33]）

Python ベースの DDuo では、ユーザは通常のデータ解析コードに近い形で処理を記述できる。以下の例は、`adult.csv` の年齢列を読み込み、値をクリップした上で合計を求め、ラプラスメカニズムによってノイズを付加している。

```
from duet import pandas as pd

df = pd.read_csv("adult.csv")
clipped_df = df['Age'].clip(0, 100)
noisy_sum = duet.laplace(clipped_df.sum(), epsilon=1.0)

print(noisy_sum)
print(clipped_df)
```

実行結果の一例を下に示す。ここで出力される `noisy_sum`（1 行目）はノイズ付きの合計値であり、`clipped_df`（2 行目）には各個人の年齢が 0 から 100 の範囲にクリップされた値が格納されている。

```
1256242.340188153
DuetWrapper(<class 'pandas.core.series.Series'>, [DataSource(
      adult.csv): 100], <duet.duet_vals.L1 object at 0
      x7f2966389e20>)
```

DDuo では、既存の pandas API と同様の記法でプライバシー保護処理を組み込める点が特徴である。また、出力に含まれる `DuetWrapper` オブジェクトは、内部的にどのデータソースが利用され、どのような敏感度制御（ここでは L_1）が適用されたかを示しており、実行ログとしてプライバシー保証の根拠を確認できる。

5.2.3 型システムベースのツール（例: Duet [32]）

このコードでは、まず入力データ `X` に対して L_1 ノルムで寄与を 1 にクリップし（`mclip[L1]`）、その上で畳み込み（`mfold`）によって合計を計算している。こうして得られる関数は敏感度 1 であるため、ラプラスメカニズム `laplace[1.0]` を適用することで ϵ-DP を満たすノイズ付き合計値が得られる。

```
let sum = pλ d : ℕ, n : ℕ, ϵ : ℝ+.
            ϵ : ℝ+[ϵ], X : 𝕄 [L1, U|d, n · 𝔻] ⇒
      noisysum ← laplace [ℝ+[1.0], ϵ] ⟨X⟩ {
        mfold 0.0, (mclip[L1] X) {a, x ⇒ a + x}};
      return noisysum
in sum
```

ツールは字句構文解析・型検査の各ステップを経て、最終的に以下の型を導出する：

```
READING
(4.43e−3s)
TOKENIZING
(0.61181s)
PARSING
(5.3395e−2s)
TYPE CHECKING
(7.36e−4s)
DONE
Inr ⟨{},∀ d:ℕ , n:ℕ , ϵ:ℝ+ . ℝ+[ϵ]@⊤ ,
```

```
11 |  M [L1,U|d,(n · D )]@ϵ ⇒ R⟩
```

以上から任意のデータの行数 d、列数 n、プライバシー予算 ϵ に対して、入力が L_1 メトリックのデータであれば、出力された実数は ϵ-DP で保護されている、と形式的に確認できる。

5.3 既存ツールの実用性と運用性に対する課題

差分プライバシー検証ツールを実際に導入・使用する観点からは、利用環境や前提知識に関して一定の検討を要する場面が見られた。動的解析に基づくツールの多くは、Python などの汎用プログラミング言語を入力として利用可能であり、実行も比較的容易である。一方で、任意のプログラムをそのまま入力できるわけではなく、典型的には提供されたサンプルプログラムに準じた記述が求められる。また、エラーメッセージの内容が分かりにくい場合もあり、非専門家にとっては障壁となることが多い。

型システムやプログラム論理に基づく手法では、導入や運用に高い専門性が要求される傾向にある。例えば、CertiPriv は強力な証明能力を有しており、Rocq の定理証明環境と連携することで柔軟な証明が可能であるが、その使用には Hoare 論理や型理論、さらには差分プライバシーに関する深い理解が必要とされる。加えて、公式なリリースが存在せず、最新の Rocq バージョンに対応させるためには多くの補題を修正する必要があるといった実装上の制約も確認された。同様に、他のツールでも古い実行環境での動作を前提としているものが多く、最新のシステムとの互換性に課題を抱えていた。さらに、Duet や Fuzzi などの自動化に強みがあるツールでは、言語を拡張しようとすると差分プライバシーと型システムの両面に跨る高度な専門知識が要求される．加えて、検証ツール全体に共通して言えることとして、多くのレポジトリが長期間更新されておらず、環境構築や実行において慎重な作業が必要であった。

これらの点から、現時点では非専門家にも広く推奨できるような標準的かつ代表的な検証ツールは存在せず、今後そのような手法が確立されることが求められる。

6. まとめ

本研究を通じて、差分プライバシーの検証ツールにはそれぞれ固有の長所と短所があることが改めて確認された。厳密性を必ずしも求めない場面では、動的解析に基づくツールを用いてプライバシーバジェットの超過を簡易的に確認するだけでも一定の効果があると考えられる。一方で、厳密性を重視する場合には、Rocq 環境と連携した CertiPriv が強力な選択肢となるが、その利用には高度な専門知識や環境整備の負担が伴うことも事実である。その中間的な立場として、型システムやプログラム論理を組み合わせた検証手法が存在し、Fuzzi のように自動化と柔軟性の両面をある程度両立させようとする試みも報告されている。これらは完全な形式的保証には至らないものの、幅広いプログラムを柔軟に扱える点で意義があると考えられる。

以上の検討を踏まえると、今後の検証ツールには、実務で広く用いられている汎用言語の上で動作し、ある程度の厳密性を担保しながらも利用者に過度の専門性を要求しない設計が望ましいと考えられる。また、実行後に結果を確認するのではなく、可能であれば実行前の段階で検証を行える仕組みが導入されると、実務上の利用価値は一層高まると考えられる。さらに、多くの既存ツールが長期間更新されていない現状を踏まえると、継続的にメンテナンスされ、最新の開発環境に対応し続ける運用方法も重要である。形式的な保証を備えつつ、非専門家でも扱いやすい検証ツールの在り方を考えていくことで、差分プライバシー技術のより広範な普及と定着に貢献できると考えている。

参考文献

[1] Rocq. `https://rocq-lang.org/`. Accessed: 2025-08-22.

[2] Chiké Abuah, David Darais, and Joseph P Near. Solo: A lightweight static analysis for differential privacy. *Proceedings of the ACM on Programming Languages*, 6(OOPSLA2):699–728, 2022.

[3] Chike Abuah, Alex Silence, David Darais, and Joseph P Near. Dduo: General-purpose dynamic analysis for differential privacy. In *2021 IEEE 34th Computer Security Foundations Symposium (CSF)*, pages 1–15. IEEE, 2021.

[4] Damian Arquez, Matías Toro, and Éric Tanter. Gradual sensitivity typing. *arXiv preprint arXiv:2308.02018*, 2023.

[5] Gilles Barthe, Gian Pietro Farina, Marco Gaboardi, Emilio Jesús Gallego Arias, Andy Gordon, Justin Hsu, and Pierre-Yves Strub. Differentially private bayesian programming. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 68–79, 2016.

[6] Gilles Barthe, Marco Gaboardi, Emilio Jesús Gallego Arias, Justin Hsu, Aaron Roth, and Pierre-Yves Strub. Higher-order approximate relational refinement types for mechanism design and differential privacy. *ACM SIGPLAN Notices*, 50(1):55–68, 2015.

[7] Gilles Barthe, Boris Köpf, Federico Olmedo, and Santiago Zanella Beguelin. Probabilistic relational reasoning for differential privacy. In *Proceedings of the 39th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 97–110, 2012.

[8] Benjamin Bichsel, Timon Gehr, Dana Drachsler-Cohen, Petar Tsankov, and Martin Vechev. Dp-finder: Finding differential privacy violations by sampling and optimization. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 508–524, 2018.

[9] Benjamin Bichsel, Samuel Steffen, Ilija Bogunovic, and Martin Vechev. Dp-sniper: Black-box discovery of differential privacy violations using classifiers. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 391–409. IEEE, 2021.

[10] Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of cryptography conference*, pages 635–658. Springer, 2016.

[11] Sílvia Casacuberta, Michael Shoemate, Salil Vadhan, and Connor Wagaman. Widespread underestimation of sensitivity in differentially private libraries and how to fix it. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 471–484, 2022.

[12] Yan Chen and Ashwin Machanavajjhala. On the privacy properties of variants on the sparse vector technique. *arXiv preprint arXiv:1508.07306*, 2015.

[13] Dwork Cynthia. Differential privacy. *Automata, languages and programming*, pages 1–12, 2006.

[14] Zeyu Ding, Yuxin Wang, Guanhong Wang, Danfeng Zhang, and Daniel Kifer. Detecting violations of differential privacy. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 475–489, 2018.

[15] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[16] Gian Pietro Farina, Stephen Chong, and Marco Gaboardi. Coupled relational symbolic execution for differential privacy. In *European Symposium on Programming*, pages 207–233. Springer International Publishing Cham, 2021.

[17] Marco Gaboardi, Andreas Haeberlen, Justin Hsu, Arjun Narayan, and Benjamin C Pierce. Linear dependent types for differential privacy. In *Proceedings of the 40th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 357–370, 2013.

[18] Noah Johnson, Joseph P Near, Joseph M Hellerstein, and Dawn Song. Chorus: a programming framework for building scalable differential privacy mechanisms. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 535–551. IEEE, 2020.

[19] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The composition theorem for differential privacy. In *International conference on machine learning*, pages 1376–1385. PMLR, 2015.

[20] Elisabet Lobo-Vesga, Alejandro Russo, and Marco Gaboardi. A programming framework for differential privacy with accuracy concentration bounds. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 411–428. IEEE, 2020.

[21] Elisabet Lobo-Vesga, Alejandro Russo, Marco Gaboardi, and Carlos Tomé Cortiñas. Sensitivity by parametricity. *Proceedings of the ACM on Programming Languages*, 8(OOPSLA2):415–441, 2024.

[22] Min Lyu, Dong Su, and Ninghui Li. Understanding the sparse vector technique for differential privacy. *arXiv preprint arXiv:1603.01699*, 2016.

[23] Frank D McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 19–30, 2009.

[24] Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th computer security foundations symposium (CSF)*, pages 263–275. IEEE, 2017.

[25] Joseph Near and Darais David. Differential privacy bugs and why they're hard to find, 2021. `https://www.nist.gov/blogs/cybersecurity-insights/differential-privacy-bugs-and-why-theyre-hard -find`. Accessed: 2024-8-23.

[26] Joseph P Near, David Darais, Chike Abuah, Tim Stevens, Pranav Gaddamadugu, Lun Wang, Neel Somani, Mu Zhang, Nikhil Sharma, Alex Shan, et al. Duet: an expressive higher-order language and linear type system for statically enforcing differential privacy. *Proceedings of the ACM on Programming Languages*, 3(OOPSLA):1–30, 2019.

[27] Davide Proserpio, Sharon Goldberg, and Frank McSherry. Calibrating data to sensitivity in private data analysis. *arXiv preprint arXiv:1203.3453*, 2012.

[28] Jason Reed and Benjamin C Pierce. Distance makes the types grow stronger: a calculus for differential privacy. In *Proceedings of the 15th ACM SIGPLAN international conference on Functional programming*, pages 157–168, 2010.

[29] Indrajit Roy, Srinath T. V. Setty, Ann Kilzer, Vitaly Shmatikov, and Emmett Witchel. Airavat: Security and privacy for mapreduce. In *Usenix Org*, pages 297–312, 2011.

[30] Jayshree Sarathy, Sophia Song, Audrey Haque, Tania Schlatter, and Salil Vadhan. Don't look at the data! how differential privacy reconfigures the practices of data science. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–19, 2023.

[31] uvm-plaid. chorus. Updated implementation of CHORUS system for scalable differential privacy.

[32] uvm-plaid. Duet: A language for differential privacy (github repository). `https://github.com/uvm-plaid/duet`, 2019. commit master branch (latest as of 2025-08-23), accessed: 2025-08-23.

[33] uvm-plaid. dduo-python: Differentially private data utilities (github repository). `https://github.com/uvm-plaid/dduo-python/tree/main`, 2025. commit main branch, accessed: 2025-08-23.

[34] Yuxin Wang, Zeyu Ding, Daniel Kifer, and Danfeng Zhang. Checkdp: An automated and integrated approach for proving differential privacy or finding precise counterexamples. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 919–938, 2020.

[35] Yuxin Wang, Zeyu Ding, Guanhong Wang, Daniel Kifer, and Danfeng Zhang. Proving differential privacy with shadow execution. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 655–669, 2019.

[36] Yuxin Wang, Zeyu Ding, Yingtai Xiao, Daniel Kifer, and Danfeng Zhang. Dpgen: Automated program synthesis for differential privacy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 393–411, 2021.

[37] Daniel Winograd-Cort, Andreas Haeberlen, Aaron Roth, and Benjamin C Pierce. A framework for adaptive differential privacy. *Proceedings of the ACM on Programming Languages*, 1(ICFP):1–29, 2017.

[38] Danfeng Zhang and Daniel Kifer. Lightdp: Towards automating differential privacy proofs. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*, pages 888–901, 2017.

[39] Hengchu Zhang, Edo Roth, Andreas Haeberlen, Benjamin C Pierce, and Aaron Roth. Fuzzi: A three-level logic for differential privacy. *Proceedings of the ACM on Programming Languages*, 3(ICFP):1–28, 2019.

[40] 山本充子, 三浦尭之, and 中林美郷. プログラム実装時に生じる差分プライバシーの誤りの調査と改善策の提案. **コンピュータセキュリティシンポジウム** *2024* **論文集**, 2024.

[41] 山本充子, 中林美郷, and 三浦尭之. 差分プライベートな行列計算の型システムによる安全性検証. **暗号と情報セキュリティシンポジウム** *2023* **論文集**, 2023.