Formal verification of differentially private computations

Gilles Barthe
IMDEA Software Institute, Madrid, Spain

August 11, 2017

Data analysis

For companies:

- Know users
- Provide better services
- Reduce fraud

For health organizations:

- Establish genetic correlations
- Monitor epidemia
- Decision making

For users: your data, but not *your* data

- Accurate computations
- Individual privacy

Trade-off

- Individual privacy
- Overall accuracy

NB: individual privacy should entail group privacy

Data anonymization

Name	Gender	Birth data	Zip code	Status
Ann	F	12/01/1957	28000	Y
Bob	M	25/06/1970	28006	X
Eve	F	13/05/1953	28001	X
Tom	M	06/11/1966	28001	Y

Data anonymization

Name	Gender	Birth data	Zip code	Status
bafkb	F	12/01/1957	28000	*Y*
kaaui	M	25/06/1970	28006	*X*
jmbag	F	13/05/1953	28001	*X*
cjhuf	M	06/11/1966	28001	*Y*

Can infer Tom's status from knowing

- Tom is in the database
- his gender, birth date, zip code

Data anonymization

Name	Gender	Birth data	Zip code	Status
bafkb	F	12/01/1957	28000	*Y*
kaaui	M	25/06/1970	28006	*X*
jmbag	F	13/05/1953	28001	*X*
cjhuf	M	06/11/1966	28001	*Y*

Can infer Tom's status from knowing

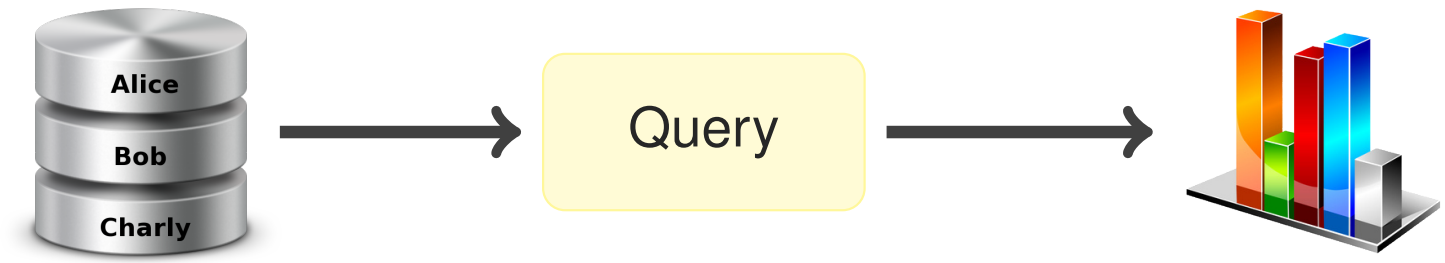
- Tom is in the database
- his gender, birth date, zip code

A real problem

- Uniquely identify $\geq$85% of individuals
- Adding noise does not help

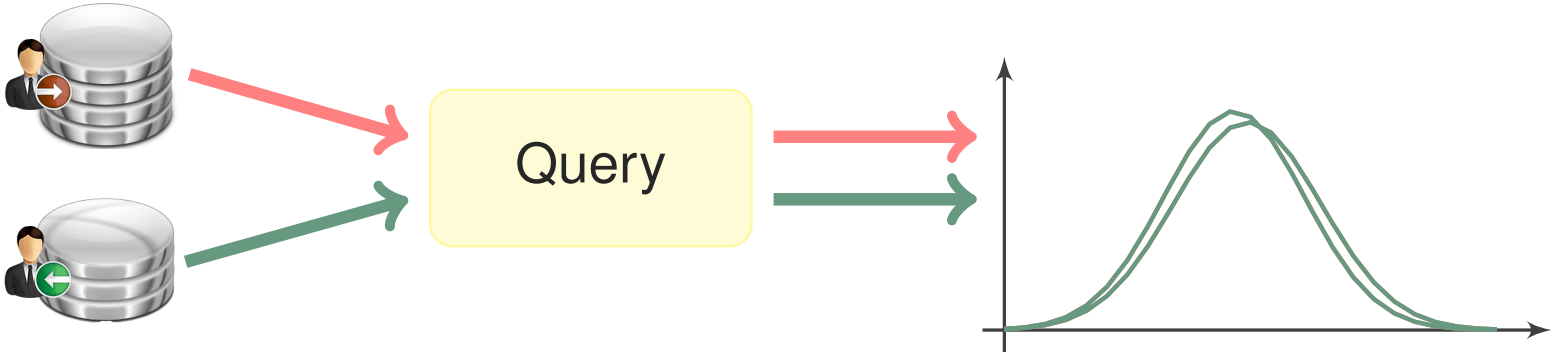
Differential privacy

(Dwork, McSherry, Nissim, and Smith, 2006)



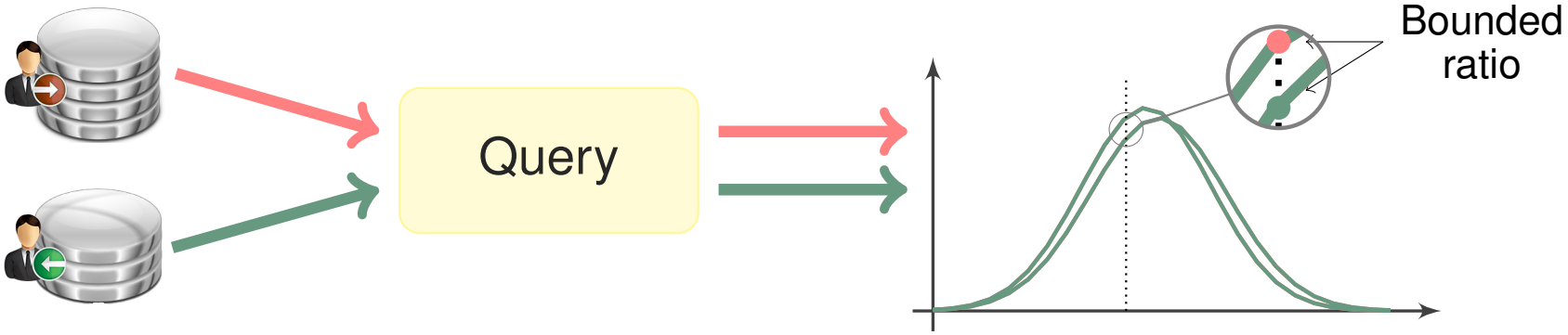
Differential privacy

(Dwork, McSherry, Nissim, and Smith, 2006)



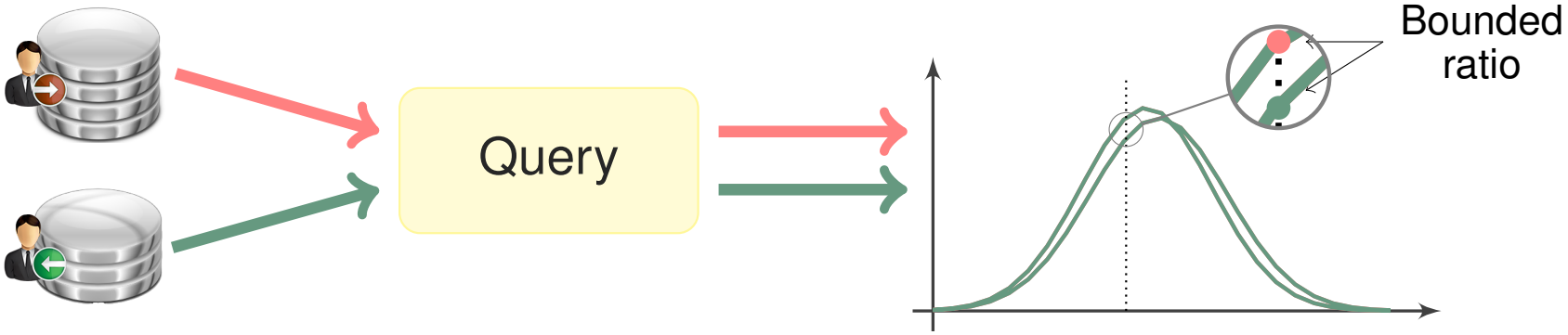
Differential privacy

(Dwork, McSherry, Nissim, and Smith, 2006)



Differential privacy

(Dwork, McSherry, Nissim, and Smith, 2006)

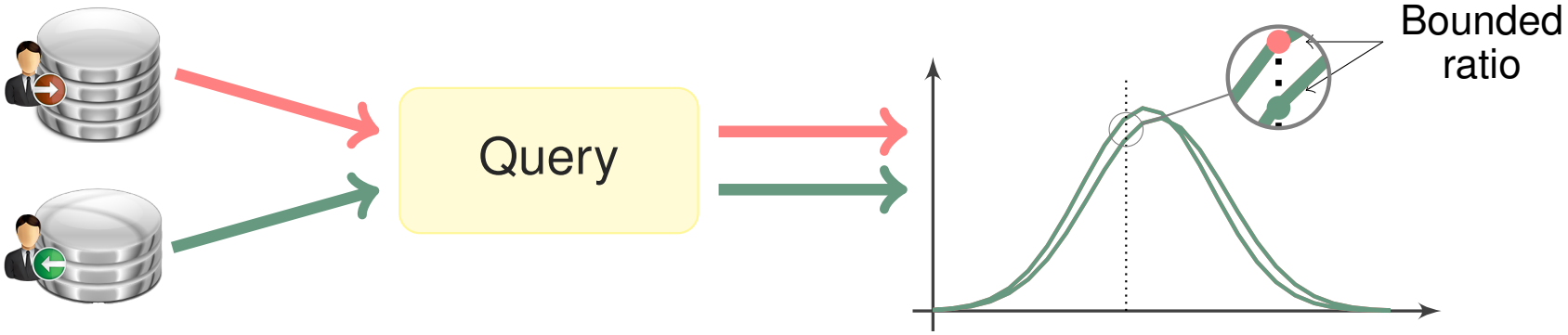


A randomized algorithm $\mathcal{K}$ is (ϵ, δ)-differentially private w.r.t. Φ iff for all databases D_1 and D_2 s.t. $\Phi(D_1, D_2)$

$$\forall S.\ \Pr[\mathcal{K}(D_1) \in S] \leq \exp(\epsilon) \cdot \Pr[\mathcal{K}(D_2) \in S] + \delta$$

Differential privacy

(Dwork, McSherry, Nissim, and Smith, 2006)



A randomized algorithm $\mathcal{K}$ is (ϵ, δ)-differentially private w.r.t. Φ iff for all databases D_1 and D_2 s.t. $\Phi(D_1, D_2)$

$$\forall S.\ \Pr[\mathcal{K}(D_1) \in S] \leq \exp(\epsilon) \cdot \Pr[\mathcal{K}(D_2) \in S] + \delta$$

Special case

If $\epsilon \approx 0$ and $\delta = 0$, then it suffices that for all databases D_1 and D_2 s.t. $\Phi(D_1, D_2)$

$$\forall x.\ \Pr[\mathcal{K}(D_1) = x] \approx (1 + \epsilon) \cdot \Pr[\mathcal{K}(D_2) = x]$$

Interpretation

- ϵ can be seen as an overall privacy budget for a given collection of data
  - each access to the database consumes part of the budget (composition theorem)
  - when the budget is consumed the data cannot be accessed anymore
- δ provides a methods to trade off accuracy and privacy: one can accept a risk of the private data to be revealed in exchange of a more accurate answer.

Note:

- Very small and very big values of ϵ can be problematic in practice.
- Very big values of δ ruin the privacy guarantee

Advantages of differential privacy

- Mathematically rigorous
- Many algorithms have a private and accurate realization
- Mechanisms for achieving privacy via output perturbation
- Composition theorems for private algorithms

Randomized response

Let q be a boolean-valued query and d be an input:

- flip a coin b;
- if $b = true$, then return $q(d)$;
- else return uniformly chosen answer

Assume $q(d)$ but not $q(d')$. Then

$$\Pr[\mathcal{K}(d) = true] = 3/4 \qquad \Pr[\mathcal{K}(d') = true] = 1/4$$

and

$$\Pr[\mathcal{K}(d) = false] = 1/4 \qquad \Pr[\mathcal{K}(d') = false] = 3/4$$

Randomized response is $(ln\, 3, 0)$-differentially private.

Output perturbation

- Population is modelled as lists of values in $[0, 1]$
- How to compute its mean privately?
- For list $(x_1, \ldots, x_n)$, return true mean with noise, i.e. $m + \mathrm{Lap}_\epsilon(0)$

Algorithm is $(\epsilon, 0)$-differentially private.
Assume $(x_1, \ldots, x_n)$ and $(x_1, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots, x_n)$ are two adjacent lists. We have:

$$\Pr[\mathcal{K}(x_1, \ldots, x_n) = k] = \exp\left(\epsilon \cdot \left|\sum_{1 \le i \le n} x_i - k\right|\right)$$

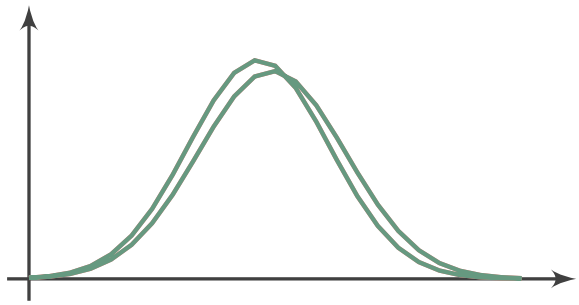
Therefore

$$\begin{aligned} \frac{\Pr[\mathcal{K}(x_1,\ldots,x_n)=k]}{\Pr[\mathcal{K}(x_1,\ldots,x_{i-1},x_i',x_{i+1},\ldots,x_n)=k]} &= \frac{\exp(\epsilon \cdot |\sum_{1 \le i \le n} x_i - k|)}{\exp(\epsilon \cdot |\sum_{1 \le i \le n} x_i' - k|)} \\ &\le \exp(\epsilon) \end{aligned}$$

Differential privacy via output perturbation

Let f be k-sensitive w.r.t. Φ:

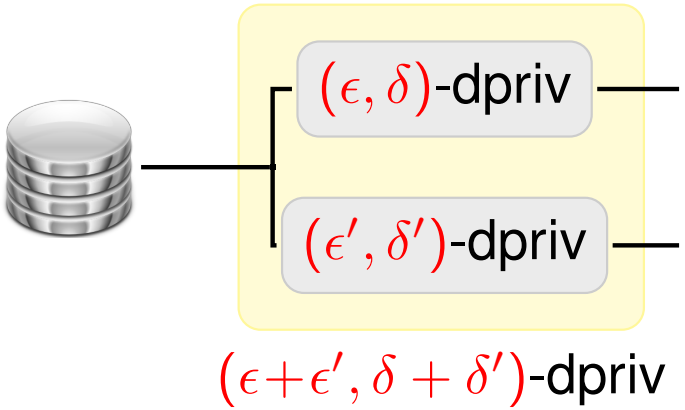
$$\Phi(a, a') \Longrightarrow |f\ a - f\ a'| \leq k$$



Then $a \mapsto \mathrm{Lap}_\epsilon(f(a))$ is $(k \cdot \epsilon, 0)$-differentially private w.r.t. Φ

Differential privacy by sequential composition

- If $\mathcal{K}$ is (ϵ, δ)-differentially private, and
- $\lambda a.\ \mathcal{K}'(a, b)$ is (ϵ', δ')-differentially private for every $b \in B$,
- then $\lambda a.\ \mathcal{K}'(a, \mathcal{K}(a))$ is $(\epsilon + \epsilon', \delta + \delta')$-differentially private



Differential privacy by parallel composition

- If $\mathcal{K}$ is (ϵ, δ)-differentially private wrt ϕ, and
- $\mathcal{K}'$ is (ϵ, δ)-differentially private with ψ,
- then $\lambda(a, b).\ (\mathcal{K}(a), \mathcal{K}(b))$ is (ϵ, δ)-differentially private wrt $\phi \uplus \psi$

Differential privacy by advanced composition

Trade-off ϵ for δ

- If $\mathcal{K}_i$ is $(\epsilon, 0)$-differentially private,
- then their sequential composition $\mathcal{K}$ is $(\sqrt{n} \cdot \epsilon, \delta)$-differentially private

Application: noisy sums

```
function NoisySum_1(a)
s = 0; i = 0;
while i < length(a) do
    s = s + a[i];
    i = i + 1;
end;
s = Lap_ε(s);
return s
```

NoisySum$_1$ is $(b \cdot \epsilon, 0)$-differentially private

Application: noisy sums

function NoisySum$_2(a)$
$s = 0; i = 0;$
while $i <$ length(a) do
  $\tilde{a} = \text{Lap}_\epsilon(a[i]);$
  $s = s + \tilde{a};$
  $i = i + 1;$
end;
return s

NoisySum$_2$ is $(b \cdot \epsilon, 0)$-differentially private

Iterative database construction

- Start with a uniform database
- At each iteration, pick a query privately (using Exponential Mechanism)
- Query is evaluated with noise in real database
- Apply deterministic update to synthetic database — takes as input (database, value, query) and return database
- After t steps release the synthetic database.

Big gain is that we do not need to limit the number of queries

Differential privacy beyond sequential composition

There is much more to differential privacy

- Exponential mechanism
- Optimal composition
- Adaptive adversaries
- Accuracy-dependent privacy
- Also, many variants of differential privacy

Issues

- Proofs are intricate and may be wrong
- Proofs, when correct, are messy
- Hard to predict when altering an algorithm breaks privacy

The Sparse Vector Technique

$\mathsf{SparseVector}_{bt}(a, b, M, N, d) :=$
$i = 0; l = [];$
$u \xleftarrow{\$} \mathsf{Lap}_\epsilon(0); A = a - u; B = b + u;$
while $i < N$ do
  $q = \mathcal{A}(l);$
  $S \xleftarrow{\$} \mathsf{Lap}_\epsilon(\mathsf{evalQ}(q, d));$
  if $(A \leq S \leq B \wedge |l| < M)$ then $l = i :: l;$
  $i = i + 1;$
return l

IF all queries are 1-sensitive,
THEN algorithm achieves $(\sqrt{M}\epsilon, \delta')$-differential privacy
EVEN IF $M \ll N$

Proof

3.1 Privacy Proof for Algorithm 1

We now prove the privacy of Algorithm 1. We break the proof down into two steps, to make the proof easier to understand, and, more importantly, to point out what confusions likely caused the different non-private variants of SVT to be proposed. In the first step, we analyze the situation where the output is $\perp^\ell$, a length-ℓ vector $\langle \perp, \cdots, \perp \rangle$, indicating that all ℓ queries are tested to be below the threshold.

LEMMA 1. *Let $\mathcal{A}$ be Algorithm 1. For any neighboring datasets D and D', and any integer ℓ, we have*

$$\mathsf{Pr}\Big[\mathcal{A}(D) = \perp^\ell\Big] \le e^{\frac{\epsilon}{2}} \mathsf{Pr}\Big[\mathcal{A}(D') = \perp^\ell\Big].$$

PROOF. We have

$$\mathsf{Pr}\Big[\mathcal{A}(D) = \perp^\ell\Big] = \int_{-\infty}^{\infty} f_\perp(D, z, L)\, dz,$$

$$\text{where } f_\perp(D, z, L) = \mathsf{Pr}[\rho = z] \prod_{i \in L} \mathsf{Pr}[q_i(D) + \nu_i < T_i + z], \quad (1)$$

$$\text{and } L = \{1, 2, \cdots, \ell\}.$$

The probability of outputting $\perp^\ell$ over D is the summation (or integral) of terms $f_\perp(D, z, L)$, each of which is the product of $\mathsf{Pr}[\rho = z]$, the probability that the threshold noise equals z, and $\prod_{i \in L} \mathsf{Pr}[q_i(D) + \nu_i < T_i + z]$, the conditional probability that $\perp^\ell$ is the output on D given that the threshold noise ρ is z. (Note that given D, $\mathbf{T}$, the queries, and ρ, whether one query results in $\perp$ or not depends completely on the noise ν_i and is independent from whether any other query results in $\perp$.) If we can prove

$$f_\perp(D, z, L) \le e^{\frac{\epsilon}{2}} f_\perp(D', z + \Delta, L), \quad (2)$$

then we have

$$\begin{aligned}
\mathsf{Pr}\Big[\mathcal{A}(D) = \perp^\ell\Big] &= \int_{-\infty}^{\infty} f_\perp(D, z, L)\, dz \\
&\le \int_{-\infty}^{\infty} e^{\frac{\epsilon}{2}} f_\perp(D', z + \Delta, L)\, dz \qquad \text{from (2)} \\
&= e^{\frac{\epsilon}{2}} \int_{-\infty}^{\infty} f_\perp(D', z', L)\, dz' \qquad \text{let } z' = z + \Delta \\
&= e^{\frac{\epsilon}{2}} \mathsf{Pr}\Big[\mathcal{A}(D') = \perp^\ell\Big].
\end{aligned}$$

This proves the lemma. It remains to prove Eq (2). For any query q_i, because $|q_i(D) - q_i(D')| \le \Delta$ and thus $-q_i(D) \le \Delta - q_i(D')$, we have

$$\begin{aligned}
\mathsf{Pr}[q_i(D) + \nu_i < T_i + z] &= \mathsf{Pr}[\nu_i < T_i - q_i(D) + z] \\
&\le \mathsf{Pr}\big[\nu_i < T_i + \Delta - q_i(D') + z\big] \\
&= \mathsf{Pr}\big[q_i(D') + \nu_i < T_i + (z + \Delta)\big] \quad (3)
\end{aligned}$$

With (3), we prove (2) as follows:

$$\begin{aligned}
f_\perp(D, z, L) &= \mathsf{Pr}[\rho = z] \prod_{i \in L} \mathsf{Pr}[q_i(D) + \nu_i < T_i + z] \\
&\le e^{\frac{\epsilon}{2}} \mathsf{Pr}[\rho = z + \Delta] \prod_{i \in L} \mathsf{Pr}\big[q_i(D') + \nu_i < T_i + (z + \Delta)\big] \\
&= e^{\frac{\epsilon}{2}} f_\perp(D', z + \Delta, L).
\end{aligned}$$

□

That is, by using a noisy threshold, we are able to bound the probability ratio for all the negative query answers (i.e., $\perp$'s) by $e^{\frac{\epsilon}{2}}$, no matter how many negative answers there are.

We can obtain a similar result for positive query answers in the same way.

$$\text{Let} \quad f_\top(D, z, L) = \mathsf{Pr}[\rho = z] \prod_{i \in L} \mathsf{Pr}[q_i(D) + \nu_i \ge T_i + z].$$

$$\text{We have } f_\top(D, z, L) \le e^{\frac{\epsilon}{2}} f_\top(D', z - \Delta, L), \text{ and thus}$$

$$\mathsf{Pr}\Big[\mathcal{A}(D) = \top^\ell\Big] \le e^{\frac{\epsilon}{2}} \mathsf{Pr}\Big[\mathcal{A}(D') = \top^\ell\Big].$$

This likely contributes to the misunderstandings behind Algorithms 5 and 6, which treat positive and negative answers exactly the same way. The problem is that while one is free to choose to bound positive or negative side, one cannot bound both.

We also observe that the proof of Lemma 1 will go through if no noise is added to the query answers, i.e., $\nu_i = 0$, because Eq (3) holds even when $\nu_i = 0$. It is likely because of this observation that Algorithm 5 adds no noise to query answers. However, when considering outcomes that include both positive answers ($\top$'s) and negative answers ($\perp$'s), one has to add noises to the query answers, as we show below.

THEOREM 2. *Algorithm 1 is ϵ-DP.*

PROOF. Consider any output vector $\boldsymbol{a} \in \{\perp, \top\}^\ell$. Let $\boldsymbol{a} = \langle a_1, \cdots, a_\ell \rangle$, $\mathbf{I}_\top^{\boldsymbol{a}} = \{i : a_i = \top\}$, and $\mathbf{I}_\perp^{\boldsymbol{a}} = \{i : a_i = \perp\}$. Clearly, $|\mathbf{I}_\top^{\boldsymbol{a}}| \le c$. We have

$$\mathsf{Pr}[\mathcal{A}(D) = \boldsymbol{a}] = \int_{-\infty}^{\infty} g(D, z)\, dz, \text{ where}$$

$$g(D, z) = \mathsf{Pr}[\rho = z] \prod_{i \in \mathbf{I}_\perp^{\boldsymbol{a}}} \mathsf{Pr}[q_i(D) + \nu_i < T_i + z] \prod_{i \in \mathbf{I}_\top^{\boldsymbol{a}}} \mathsf{Pr}[q_i(D) + \nu_i \ge T_i + z].$$

We want to show that $g(D, z) \le e^\epsilon g(D', z + \Delta)$. This suffices to prove that $\mathsf{Pr}[\mathcal{A}(D) = a] \le e^\epsilon \mathsf{Pr}[\mathcal{A}(D') = \boldsymbol{a}]$. Note that $g(D, z)$ can be written as:

$$g(D, z) = f_\perp(D, z, \mathbf{I}_\perp^{\boldsymbol{a}}) \prod_{i \in \mathbf{I}_\top^{\boldsymbol{a}}} \mathsf{Pr}[q_i(D) + \nu_i \ge T_i + z].$$

Following the proof of Lemma 1, we can show that $f_\perp(D, z, \mathbf{I}_\perp^{\boldsymbol{a}}) \le e^{\frac{\epsilon}{2}} f_\perp(D', z + \Delta, \mathbf{I}_\perp^{\boldsymbol{a}})$, and it remains to show

$$\prod_{i \in \mathbf{I}_\top^{\boldsymbol{a}}} \mathsf{Pr}[q_i(D) + \nu_i \ge T_i + z] \le e^{\frac{\epsilon}{2}} \prod_{i \in \mathbf{I}_\top^{\boldsymbol{a}}} \mathsf{Pr}\big[q_i(D') + \nu_i \ge T_i + z + \Delta\big]. \quad (4)$$

Because $\nu_i = \mathsf{Lap}\left(\frac{4c\Delta}{\epsilon}\right)$ and $|q_i(D) - q_i(D')| \le \Delta$, we have

$$\begin{aligned}
\mathsf{Pr}[q_i(D) + \nu_i \ge T_i + z] &= \mathsf{Pr}[\nu_i \ge T_i + z - q_i(D)] \\
&\le \mathsf{Pr}\big[\nu_i \ge T_i + z - \Delta - q_i(D')\big] \quad (5) \\
&\le e^{\frac{\epsilon}{2c}} \mathsf{Pr}\big[\nu_i \ge T_i + z - \Delta - q_i(D') + 2\Delta\big] \quad (6) \\
&= e^{\frac{\epsilon}{2c}} \mathsf{Pr}\big[q_i(D') + \nu_i \ge T_i + z + \Delta\big].
\end{aligned}$$

Eq (5) is because $-q_i(D) \ge -\Delta - q_i(D')$, and Eq (6) is from the Laplace distribution's property. This proves Eq (4). □

The basic idea of the proof is that when comparing $g(D, z)$ with $g(D', z + \Delta)$, we can bound the probability ratio for all outputs of $\perp$ to no more than $e^{\frac{\epsilon}{2}}$ by using a noisy threshold, no matter how many such outputs there are. To bound the ratio for the $\top$ outputs to no more than $e^{\frac{\epsilon}{2}}$, we need to add sufficient Laplacian noises, which should scale with c, the number of positive outputs.

Now we turn to Algorithms 3-6 to clarify what are wrong with their privacy proofs and to give their DP properties.

Proof

3.1 Privacy Proof for Algorithm 1

We now prove the privacy of Algorithm 1. We break the proof down into two steps, to make the proof easier to understand, and, more importantly, to point out what confusions likely caused the different non-private variants of SVT to be proposed. In the first step, we analyze the situation where the output is $\perp^\ell$, a length-ℓ vector $\langle \perp, \cdots, \perp \rangle$, indicating that all ℓ queries are tested to be below the threshold.

LEMMA 1. *Let $\mathcal{A}$ be Algorithm 1. For any neighboring datasets D and D', and any integer ℓ, we have*

$$\Pr\left[\mathcal{A}(D) = \perp^\ell\right] \le e^{\frac{\epsilon}{2}} \Pr\left[\mathcal{A}(D') = \perp^\ell\right].$$

PROOF. We have

$$\Pr\left[\mathcal{A}(D) = \perp^\ell\right] = \int_{-\infty}^{\infty} f_\perp(D, z, L)\, dz,$$

$$\text{where } f_\perp(D, z, L) = \Pr[\rho = z] \prod_{i \in L} \Pr[q_i(D) + \nu_i < T_i + z], \quad (1)$$

$$\text{and } L = \{1, 2, \cdots, \ell\}.$$

The probability of outputting $\perp^\ell$ over D is the summation (or integral) of terms $f_\perp(D, z, L)$, each of which is the product of $\Pr[\rho = z]$, the probability that the threshold noise equals z, and $\prod_{i \in L} \Pr[q_i(D) + \nu_i < T_i + z]$, the conditional probability that $\perp^\ell$ is the output on D given that the threshold noise ρ is z. (Note that given D, $\mathbf{T}$, the queries, and ρ, whether one query results in $\perp$ or not depends completely on the noise ν_i and is independent from whether any other query results in $\perp$.) If we can prove

$$f_\perp(D, z, L) \le e^{\frac{\epsilon}{2}} f_\perp(D', z + \Delta, L), \quad (2)$$

then we have

$$\begin{aligned}
\Pr\left[\mathcal{A}(D) = \perp^\ell\right] &= \int_{-\infty}^{\infty} f_\perp(D, z, L)\, dz \\
&\le \int_{-\infty}^{\infty} e^{\frac{\epsilon}{2}} f_\perp(D', z + \Delta, L)\, dz \qquad \text{from (2)} \\
&= e^{\frac{\epsilon}{2}} \int_{-\infty}^{\infty} f_\perp(D', z', L)\, dz' \qquad \text{let } z' = z + \Delta \\
&= e^{\frac{\epsilon}{2}} \Pr\left[\mathcal{A}(D') = \perp^\ell\right].
\end{aligned}$$

This proves the lemma. It remains to prove Eq (2). For any query q_i, because $|q_i(D) - q_i(D')| \le \Delta$ and thus $-q_i(D) \le \Delta - q_i(D')$, we have

$$\begin{aligned}
\Pr[q_i(D) + \nu_i < T_i + z] &= \Pr[\nu_i < T_i - q_i(D) + z] \\
&\le \Pr\left[\nu_i < T_i + \Delta - q_i(D') + z\right] \\
&= \Pr\left[q_i(D') + \nu_i < T_i + (z + \Delta)\right] \quad (3)
\end{aligned}$$

With (3), we prove (2) as follows:

$$\begin{aligned}
f_\perp(D, z, L) &= \Pr[\rho = z] \prod_{i \in L} \Pr[q_i(D) + \nu_i < T_i + z] \\
&\le e^{\frac{\epsilon}{2}} \Pr[\rho = z + \Delta] \prod_{i \in L} \Pr\left[q_i(D') + \nu_i < T_i + (z + \Delta)\right] \\
&= e^{\frac{\epsilon}{2}} f_\perp(D', z + \Delta, L).
\end{aligned}$$

□

That is, by using a noisy threshold, we are able to bound the probability ratio for all the negative query answers (i.e., $\perp$'s) by $e^{\frac{\epsilon}{2}}$, no matter how many negative answers there are.

We can obtain a similar result for positive query answers in the same way.

$$\text{Let} \quad f_\top(D, z, L) = \Pr[\rho = z] \prod_{i \in L} \Pr[q_i(D) + \nu_i \ge T_i + z].$$

$$\text{We have } f_\top(D, z, L) \le e^{\frac{\epsilon}{2}} f_\top(D', z - \Delta, L), \text{ and thus}$$

$$\Pr\left[\mathcal{A}(D) = \top^\ell\right] \le e^{\frac{\epsilon}{2}} \Pr\left[\mathcal{A}(D') = \top^\ell\right].$$

This likely contributes to the misunderstandings behind Algorithms 5 and 6, which treat positive and negative answers exactly the same way. The problem is that while one is free to choose to bound positive or negative side, one cannot bound both.

We also observe that the proof of Lemma 1 will go through if no noise is added to the query answers, i.e., $\nu_i = 0$, because Eq (3) holds even when $\nu_i = 0$. It is likely because of this observation that Algorithm 5 adds no noise to query answers. However, when considering outcomes that include both positive answers ($\top$'s) and negative answers ($\perp$'s), one has to add noises to the query answers, as we show below.

THEOREM 2. *Algorithm 1 is ϵ-DP.*

PROOF. Consider any output vector $\boldsymbol{a} \in \{\perp, \top\}^\ell$. Let $\boldsymbol{a} = \langle a_1, \cdots, a_\ell \rangle$, $\mathbf{I}_\top^{\boldsymbol{a}} = \{i : a_i = \top\}$, and $\mathbf{I}_\perp^{\boldsymbol{a}} = \{i : a_i = \perp\}$. Clearly, $|\mathbf{I}_\top^{\boldsymbol{a}}| \le c$. We have

$$\Pr[\mathcal{A}(D) = \boldsymbol{a}] = \int_{-\infty}^{\infty} g(D, z)\, dz, \text{ where}$$

$$g(D, z) = \Pr[\rho = z] \prod_{i \in \mathbf{I}_\top^{\boldsymbol{a}}} \Pr[q_i(D) + \nu_i < T_i + z] \prod_{i \in \mathbf{I}_\perp^{\boldsymbol{a}}} \Pr[q_i(D) + \nu_i \ge T_i + z].$$

We want to show that $g(D, z) \le e^\epsilon g(D', z + \Delta)$. This suffices to prove that $\Pr[\mathcal{A}(D) = a] \le e^\epsilon \Pr[\mathcal{A}(D') = \boldsymbol{a}]$. Note that $g(D, z)$ can be written as:

$$g(D, z) = f_\perp(D, z, \mathbf{I}_\perp^{\boldsymbol{a}}) \prod_{i \in \mathbf{I}_\top^{\boldsymbol{a}}} \Pr[q_i(D) + \nu_i \ge T_i + z].$$

Following the proof of Lemma 1, we can show that $f_\perp(D, z, \mathbf{I}_\perp^{\boldsymbol{a}}) \le e^{\frac{\epsilon}{2}} f_\perp(D', z + \Delta, \mathbf{I}_\perp^{\boldsymbol{a}})$, and it remains to show

$$\prod_{i \in \mathbf{I}_\top^{\boldsymbol{a}}} \Pr[q_i(D) + \nu_i \ge T_i + z] \le e^{\frac{\epsilon}{2}} \prod_{i \in \mathbf{I}_\top^{\boldsymbol{a}}} \Pr\left[q_i(D') + \nu_i \ge T_i + z + \Delta\right]. \quad (4)$$

Because $\nu_i = \mathsf{Lap}\left(\frac{4c\Delta}{\epsilon}\right)$ and $|q_i(D) - q_i(D')| \le \Delta$, we have

$$\begin{aligned}
\Pr[q_i(D) + \nu_i \ge T_i + z] &= \Pr[\nu_i \ge T_i + z - q_i(D)] \\
&\le \Pr\left[\nu_i \ge T_i + z - \Delta - q_i(D')\right] \quad (5) \\
&\le e^{\frac{\epsilon}{2c}} \Pr\left[\nu_i \ge T_i + z - \Delta - q_i(D') + 2\Delta\right] \quad (6) \\
&= e^{\frac{\epsilon}{2c}} \Pr\left[q_i(D') + \nu_i \ge T_i + z + \Delta\right].
\end{aligned}$$

Eq (5) is because $-q_i(D) \ge -\Delta - q_i(D')$, and Eq (6) is from the Laplace distribution's property. This proves Eq (4). □

The basic idea of the proof is that when comparing $g(D, z)$ with $g(D', z + \Delta)$, we can bound the probability ratio for all outputs of $\perp$ to no more than $e^{\frac{\epsilon}{2}}$ by using a noisy threshold, no matter how many such outputs there are. To bound the ratio for the $\top$ outputs to no more than $e^{\frac{\epsilon}{2}}$, we need to add sufficient Laplacian noises, which should scale with c, the number of positive outputs.

Now we turn to Algorithms 3-6 to clarify what are wrong with their privacy proofs and to give their DP properties.

Figure 1: A Selection of SVT Variants

Input/Output shared by all SVT Algorithms

Input: A private database D, a stream of queries $Q = q_1, q_2, \cdots$ each with sensitivity no more than Δ, either a sequence of thresholds $\mathbf{T} = T_1, T_2, \cdots$ or a single threshold T (see footnote *), and c, the maximum number of queries to be answered with $\top$.

Output: A stream of answers $a_1, a_2, \cdots$, where each $a_i \in \{\top, \perp\} \cup \mathbb{R}$ and $\mathbb{R}$ denotes the set of all real numbers.

Algorithm 1 An instantiation of the SVT proposed in this paper.

Input: $D, Q, \Delta, \mathbf{T} = T_1, T_2, \cdots, c$.
$\rho = \mathsf{Lap}\left(\frac{2\Delta}{\epsilon}\right)$, count = 0
for each query $q_i \in Q$ **do**
  $\nu_i = \mathsf{Lap}\left(\frac{4c\Delta}{\epsilon}\right)$
  if $q_i(D) + \nu_i \ge T_i + \rho$ **then**
    Output $a_i = \top$
    count = count + 1, **Abort** if count $\ge c$.
  else
    Output $a_i = \perp$
  end if
end for

Algorithm 2 SVT in Dwork and Roth 2014 [8].

Input: D, Q, Δ, T, c.
$\rho = \mathsf{Lap}\left(\frac{2c\Delta}{\epsilon}\right)$, count = 0
for each query $q_i \in Q$ **do**
  $\nu_i = \mathsf{Lap}\left(\frac{4c\Delta}{\epsilon}\right)$
  if $q_i(D) + \nu_i \ge T + \rho$ **then**
    Output $a_i = \top$, $\rho = \mathsf{Lap}\left(\frac{2c\Delta}{\epsilon}\right)$
    count = count + 1, **Abort** if count $\ge c$.
  else
    Output $a_i = \perp$
  end if
end for

Algorithm 3 SVT in Roth's 2011 Lecture Notes [15].

Input: D, Q, Δ, T, c.
$\rho = \mathsf{Lap}\left(\frac{2\Delta}{\epsilon}\right)$, count = 0
for each query $q_i \in Q$ **do**
  $\nu_i = \mathsf{Lap}\left(\frac{2c\Delta}{\epsilon}\right)$
  if $q_i(D) + \nu_i \ge T + \rho$ **then**
    Output $a_i = q_i(D) + \nu_i$
    count = count + 1, **Abort** if count $\ge c$.
  else
    Output $a_i = \perp$
  end if
end for

Algorithm 4 SVT in Lee and Clifton 2014 [13].

Input: D, Q, Δ, T, c.
$\rho = \mathsf{Lap}\left(\frac{4\Delta}{\epsilon}\right)$, count = 0
for each query $q_i \in Q$ **do**
  $\nu_i = \mathsf{Lap}\left(\frac{4\Delta}{3\epsilon}\right)$
  if $q_i(D) + \nu_i \ge T + \rho$ **then**
    Output $a_i = \top$
    count = count + 1, **Abort** if count $\ge c$.
  else
    Output $a_i = \perp$
  end if
end for

Algorithm 5 SVT in Stoddard et al. 2014 [18].

Input: D, Q, Δ, T.
$\rho = \mathsf{Lap}\left(\frac{2\Delta}{\epsilon}\right)$
for each query $q_i \in Q$ **do**
  $\nu_i = 0$
  if $q_i(D) + \nu_i \ge T + \rho$ **then**
    Output $a_i = \top$

  else
    Output $a_i = \perp$
  end if
end for

Algorithm 6 SVT in Chen et al. 2015 [1].

Input: $D, Q, \Delta, \mathbf{T} = T_1, T_2, \cdots$.
$\rho = \mathsf{Lap}\left(\frac{2\Delta}{\epsilon}\right)$
for each query $q_i \in Q$ **do**
  $\nu_i = \mathsf{Lap}\left(\frac{2\Delta}{\epsilon}\right)$
  if $q_i(D) + \nu_i \ge T_i + \rho$ **then**
    Output $a_i = \top$

  else
    Output $a_i = \perp$
  end if
end for

	Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5	Alg. 6
Scale of threshold noise ρ	$2\Delta/\epsilon$	$2c\Delta/\epsilon$	$2\Delta/\epsilon$	$4\Delta/\epsilon$	$2\Delta/\epsilon$	$2\Delta/\epsilon$
Reset ρ after each output of $\top$	No	Yes	No	No	No	No
Scale of query noise ν_i	$4c\Delta/\epsilon$	$4c\Delta/\epsilon$	$2c\Delta/\epsilon$	$4\Delta/3\epsilon$	0	$2\Delta/\epsilon$
Outputting $q_i + \nu_i$ instead of $\top$	No	No	Yes	No	No	No
Stop after outputting c $\top$'s	Yes	Yes	Yes	Yes	No	No
Privacy Property	ϵ-DP	ϵ-DP	∞-DP **	$(\frac{1+6c}{4}\epsilon)$-DP	∞-DP	∞-DP

Figure 2: Differences among Algorithms 1-6.

* Algorithms 1 and 6 use a sequence of thresholds $\mathbf{T} = T_1, T_2, \cdots$, allowing different thresholds for different queries. The other algorithms use the same threshold T for all queries. We point out that this difference is mostly syntactical. In fact, having an SVT where the threshold always equals 0 suffices. Given a sequence of queries $q_1, q_2, \cdots$, and a sequence of thresholds $\mathbf{T} = T_1, T_2, \cdots$, we can define a new sequence of queries $r_i = q_i - T_i$, and apply the SVT to r_i using 0 as the threshold to obtain the same result. In this paper, we decide to use thresholds to be consistent with the existing papers.

** ∞-DP means that an algorithm doesn't satisfy ϵ'-DP for any finite privacy budget ϵ'.

Approximate couplings

- Let μ_1 and μ_2 are distributions over A
- Let μ_L and μ_R be distributions over $A \times A$
- (μ_L, μ_R) is an (ϵ, δ)-coupling of (μ_1, μ_2) if
  - $\pi_1(\mu_L) = \mu_1$ and $\pi_2(\mu_R) = \mu_2$
  - $\Delta_\epsilon(\mu_L, \mu_R) \leq \delta$

Benefits

- (Almost) no probabilistic reasoning
- Mechanizable
- Composition theorems extend
- New proof techniques
- Extend to f-divergences

Approximate probabilistic Relational Hoare Logic

- Judgment

$$\vDash \{P\}\ c_1\ \sim_{\epsilon,\delta}\ c_2\ \{Q\}$$

- Validity

$$\forall m_1, m_2.\ (m_1, m_2) \vDash P \implies (\llbracket c_1 \rrbracket\ m_1, \llbracket c_2 \rrbracket\ m_2) \vDash Q^{\sharp(\epsilon,\delta)}$$

- c is (ϵ, δ)-differentially private wrt Φ iff

$$\vDash \{\Phi\}\ c\ \sim_{\epsilon,\delta}\ c\ \{\equiv\}$$

Proof rules

$$\frac{\begin{array}{c}\models \{P\}\ c_1 \ \sim_{\epsilon,\delta}\ c_2\ \{Q\} \\ \models \{Q\}\ c_1' \ \sim_{\epsilon',\delta'}\ c_2'\ \{R\}\end{array}}{\models \{P\}\ c_1; c_1' \ \sim_{\epsilon+\epsilon',\delta+\delta'}\ c_2; c_2'\ \{R\}}$$

$$\frac{\begin{array}{c}\models \{P \wedge e\langle 1\rangle\}\ c_1 \ \sim_{\epsilon,\delta}\ c\ \{Q\} \\ \models \{P \wedge \neg e\langle 1\rangle\}\ c_2 \ \sim_{\epsilon,\delta}\ c\ \{Q\}\end{array}}{\models \{P\}\ \text{if } e \text{ then } c_1 \text{ else } c_2 \ \sim_{\epsilon,\delta}\ c\ \{Q\}}$$

$$\frac{\begin{array}{c}\models \{P \wedge e\langle 1\rangle\}\ c_1 \ \sim_{\epsilon,\delta}\ c_1'\ \{Q\} \\ \models \{P \wedge \neg e\langle 1\rangle\}\ c_2 \ \sim_{\epsilon,\delta}\ c_2'\ \{Q\} \\ P \rightarrow e\langle 1\rangle = e'\langle 2\rangle\end{array}}{\models \{P\}\ \text{if } e \text{ then } c_1 \text{ else } c_2 \ \sim_{\epsilon,\delta}\ \text{if } e' \text{ then } c_1' \text{ else } c_2'\ \{Q\}}$$

Proof principles for Laplace mechanism

Making different things look equal

$$\frac{\Phi \triangleq |e_1\langle 1\rangle - e_2\langle 2\rangle| \leq k'}{\vDash \{\Phi\}\ y_1 \xleftarrow{\$} \mathrm{Lap}_\epsilon(e_1)\ \sim_{k'\cdot\epsilon,0}\ y_2 \xleftarrow{\$} \mathrm{Lap}_\epsilon(e_2)\ \{y_1\langle 1\rangle = y_2\langle 2\rangle\}}$$

Making equal things look different

$$\frac{\Phi \triangleq e_1\langle 1\rangle = e_2\langle 2\rangle}{\vDash \{\Phi\}\ y_1 \xleftarrow{\$} \mathrm{Lap}_\epsilon(e_1)\ \sim_{k\cdot\epsilon,0}\ y_2 \xleftarrow{\$} \mathrm{Lap}_\epsilon(e_2)\ \{y_1\langle 1\rangle + k = y_2\langle 2\rangle\}}$$

Keeping things the same, at no cost

$$\frac{y_1 \notin FV(e_1) \qquad y_2 \notin FV(e_2) \qquad \Psi \triangleq y_1\langle 1\rangle - y_2\langle 2\rangle = e_1\langle 1\rangle - e_2\langle 2\rangle}{\vDash \{\top\}\ y_1 \xleftarrow{\$} \mathrm{Lap}_\epsilon(e_1)\ \sim_{0,0}\ y_2 \xleftarrow{\$} \mathrm{Lap}_\epsilon(e_2)\ \{\Psi\}}$$

Pointwise equality

$$\frac{\forall i. \vDash \{\Phi\}\ c_1\ \sim_{\epsilon,0}\ c_2\ \{x\langle 1\rangle = i \Rightarrow x\langle 2\rangle = i\}}{\vDash \{\Phi\}\ c_1\ \sim_{\epsilon,0}\ c_2\ \{x\langle 1\rangle = x\langle 2\rangle\}}$$

Coupling proof of sparse vector

Case $b = +\infty$ and $M = 1$

- (Cost ϵ): set $A\langle 1\rangle + 1 = A\langle 2\rangle$
- By pointwise equality, must prove for all k

$$I\langle 1\rangle = k \Rightarrow I\langle 2\rangle = k$$

- (Cost ϵ) critical iteration $i = k$: set $S\langle 1\rangle + 1 = S\langle 2\rangle$, and hence left test succeeds iff right test succeeds
- (Cost 0) iterations $i < k$: by sensitivity, $|S\langle 1\rangle - S\langle 2\rangle| \leq 1$, therefore right test succeeds implies left test succeeds
- (Cost 0) iterations $i > k$: similar

General case

- Use new optimal subset coupling for critical iterations
- Use accuracy to ensure that noisy intervals are non-empty

Accuracy via union bound logics

- Judgment $\vDash_\beta \{\Phi\}\, c\, \{\Psi\}$
- Validity: for every m, $m \models \Phi$ implies $\Pr_{[\![c]\!](m)}[\neg\Psi] \leq \beta$

Selected rules

$$\frac{\vDash_\beta \{\Phi\}\, c\, \{\Phi'\} \qquad \vDash_{\beta'} \{\Phi'\}\, c'\, \{\Phi''\}}{\vDash_{\beta+\beta'} \{\Phi\}\, c;\ c'\, \{\Phi''\}}$$

$$\frac{\vDash_\beta \{\Phi \wedge e\}\, c\, \{\Psi\} \qquad \vDash_\beta \{\Phi \wedge \neg e\}\, c'\, \{\Psi\}}{\vDash_\beta \{\Phi\}\, \text{if } e \text{ then } c \text{ else } c'\, \{\Psi\}}$$

$$\frac{\beta \in (0,1) \qquad \gamma = \frac{1}{\epsilon} \log\left(\frac{1}{\beta}\right)}{\vDash_\beta \{\top\}\, x \xleftarrow{\$} \mathrm{Lap}_\epsilon(e)\, \{|x - e| \leq \gamma\}}$$

Formalization

isv.ec

```
 + while true (N - i); last by auto => /#.
 move=> z; wp; rnd predT; call adv_ll; auto; smt(lap_ll).
+ while true (N - i); last by auto => /#.
  move=> z; wp; rnd predT; call adv_ll; auto; smt(lap_ll).
+ by move=> /#.
move => [L GA] => /=.
case (size L <= M).
conseq <[ bigi predT (fun (x : int) =>
if mem L (N - 1 - x) then 2%r * eps else 0%r) 0 N) & 0%r ]>.
  move=> &1 &2 /> _ lt_LM; rewrite addrK -big_mkcond.
  rewrite (partition_big (fun i => N - 1 - i) _ predT _ _ (undup L)) /=.
  + by rewrite undup_uniq.
  + by move=> /> y _; rewrite mem_undup.
  apply/(ler_trans (big predT (fun _ => 2%r * eps) (undup L))); last first.
    rewrite Bigreal.sumr_const mulrAC (mulrC (count _ _)%r).
    rewrite ler_wpmul2l ?mulr_ge0 // count_predT le_fromint.
    by apply/(IntOrder.ler_trans _ _ _ (size_undup _)).
  apply/Bigreal.ler_sum=> i _ /=; rewrite big_mkcond /=.
  case: (0 <= N - 1 - i < N) => h; last first.
    rewrite -big_mkcond big_seq_cond big_pred0 ?mulr_ge0 //=.
    by move=> j; rewrite mem_range /#.
  rewrite (bigD1 _ _ (N - 1 - i)) ?mem_range ?range_uniq //=.
  by rewrite big1 1:/# /=; case: (_ /\ _) => //=; rewrite mulr_ge0.
awhile [ (fun x => if (mem L (N - 1 - x)) then 2%r * eps else 0%r)
& (fun _ => 0%r) ] N [N - i - 1] (
  adj d{1} d{2} /\ ={i} /\ t0{1} = t0{2} + 1 /\
(suffix l{2} L => ={l, glob A})) => //; first 4 by smt(ge0_eps).
by rewrite sumr_const intmulr.
move => v.
case (suffix (i{2}::l{2}) L).
conseq  <[ (2%r * eps) & 0%r ]>.
smt(suffix_cons).
seq 1 1: (adj d{1} d{2} /\
  ={i, l, q, glob A} /\
  t0{1} = t0{2} + 1 /\ suffix (i{2}::l{2}) L /\ v = N - i{1} - 1).
toequiv.
call ( _ : true); skip; smt(suffixI suffix_cons).
seq 1 1: (adj d{1} d{2} /\
   ={i, l, q, glob A} /\
   t0{1} = t0{2} + 1 /\ suffix (i{2}::l{2}) L /\ v = N - i{1} - 1 /\
  suffix (i{2} :: l{2}) L /\ x{1}=x{2}+1) <[ (2%r * eps) & 0%r ]>.
lap (-1) 2 => />; smt(one_sens).
toequiv; auto.
smt.
toequiv.
smt(ge0_eps).
case (suffix l{2} L).
seq >> 1 1: (adj d{1} d{2} /\
     ={i, q } /\
     t0{1} = t0{2} + 1 /\
     suffix l{2} L /\ ={l, glob A} /\
    i{1} < N /\ v = N - i{1} - 1 /\
   ! suffix (i{2} :: l{2}) L).
call ( _ : true); skip; smt(suffixI suffix_cons).
wp.
 exists* (evalQ q d){1}; elim* => e1;
```

-:--- isv.ec 51% L130 Git:master (EasyCrypt script Scripting)

```
Current goal (remaining: 2)

Type variables: <none>

L : int list
GA: (glob A)
------------------------------------------------------------------
forall (k : int),
  aequiv[[if mem L (N - 1 - k) then 2%r * eps else 0%r & 0%r] M(A).main.q <@
            A.adv(M(A).main.l{hr});; ...; M(A).main.i <-
            M(A).main.i{hr} + 1 ~ M(A).main.q <@ A.adv(M(A).main.l{hr});;
          ...; M(A).main.i <- M(A).main.i{hr} + 1 :
          (adj d{1} d{2} /\
           ={i} /\ t0{1} = t0{2} + 1 /\ (suffix l{2} L => ={l, glob A})) /\
          i{1} < N /\ i{2} < N /\ k = N - i{1} - 1 /\ N - i{1} - 1 <= N ==>
          (adj d{1} d{2} /\
           ={i} /\ t0{1} = t0{2} + 1 /\ (suffix l{2} L => ={l, glob A})) /\
          i{1} < N = i{2} < N /\ N - i{1} - 1 < k]
```

U:%%- *goals* All L1 (EasyCrypt goals)

```
+ >> Copyright (c) - 2012-2015 - IMDEA Software Institute and INRIA

| >> Distributed under the terms of the CeCILL-C license

| >> Standard Library (theories/**/*.ec):

| >>    Distributed under the terms of the CeCILL-B license

| >>

| >> GIT hash: cb24e5da044aef7da56d1359d7c4b97ed6fe0669
```

U:%%- *response* All L12 (EasyCrypt response)

tool-bar goto

Tools

- EasyPriv: interactive proof assistant for probabilistic programs (variant of EasyCrypt)
- Hoare2: higher-order language

Case studies

- statistics
- combinatorial optimization
- (computational) differential privacy
- mechanism design

Conclusion

- Approximate couplings naturally capture differential privacy
- Advanced concepts can be handled
- Future: variants of differential privacy