Group and attack: Auditing differential privacy

Johan Lokna **Anouk Paradis** Dimitar I. Dimitrov Martin Vechev

ETH Zurich, Switzerland

lokjoh@gmail.com, {anouk.paradis, dimitar.iliev.dimitrov, martin.vechev}@inf.ethz.ch

Why Audit Differential Privacy?

Definition

M is (ϵ, δ)-DP iff for all $(a, a') \in \mathcal{N}$ and S:

$$\Pr[M(a) \in S] \leq \exp(\epsilon)\Pr[M(a') \in S] + \delta$$

Why Audit Differential Privacy?

Definition

M is (ϵ, δ)-DP iff for all $(a, a') \in \mathcal{N}$ and S:

$$\Pr[M(a) \in S] \leq \exp(\epsilon)\Pr[M(a') \in S] + \delta$$

Privacy decreases with increasing (ϵ, δ)

Why Audit Differential Privacy?

Definition

M is (ϵ, δ)-DP iff for all $(a, a') \in \mathcal{N}$ and S:

$$\Pr[M(a) \in S] \leq \exp(\epsilon)\Pr[M(a') \in S] + \delta$$

Theory - Gauss Mechanism

$$\begin{array}{l} M_{\epsilon,\delta}(a): \\ \quad \sigma = f(\epsilon, \delta) \\ \quad \eta \sim \mathcal{N}(0, \sigma) \\ \quad \text{return count}(a) + \eta \end{array}$$

Why Audit Differential Privacy?

Definition

M is (ϵ, δ)-DP iff for all $(a, a') \in \mathcal{N}$ and S:

$$\Pr[M(a) \in S] \leq \exp(\epsilon)\Pr[M(a') \in S] + \delta$$

Theory - Gauss Mechanism

$$\begin{aligned} & M_{\epsilon,\delta}(a): \\ & \quad \sigma = f(\epsilon, \delta) \\ & \quad \eta \sim \mathcal{N}(0, \sigma) \\ & \quad \text{return count}(a) + \eta \end{aligned}$$

Implementation

```python
def dp_count(count):
    std = compute_std(epsilon, delta)
    noise = np.random.normal(scale=std)
    return count + noise

```

Why Audit Differential Privacy?

Definition

M is (ϵ, δ)-DP iff for all $(a, a') \in \mathcal{N}$ and S:

$$\Pr[M(a) \in S] \leq \exp(\epsilon)\Pr[M(a') \in S] + \delta$$

Theory - Gauss Mechanism

$$\begin{array}{l} M_{\epsilon,\delta}(a): \\ \quad \left| \begin{array}{l} \sigma = f(\epsilon, \delta) \\ \eta \sim \mathcal{N}(0, \sigma) \\ \text{return count}(a) + \eta \end{array} \right. \end{array}$$

Implementation

```python
def dp_count(count):
    std = compute_std(epsilon, delta)
    noise = np.random.normal(scale=std)
    return count + noise

```

Jiankai Jin et al. “Are we there yet? timing and floating-point attacks on differential privacy systems.” (2021)

Why Audit Differential Privacy?

Definition

M is (ϵ, δ)-DP iff for all $(a, a') \in \mathcal{N}$ and S:

$$\Pr[M(a) \in S] \leq \exp(\epsilon)\Pr[M(a') \in S] + \delta$$

Theory - Gauss Mechanism

$$M_{\epsilon,\delta}(a):$$

$$\sigma = f(\epsilon, \delta)$$

$$\eta \sim \mathcal{N}(0, \sigma)$$

$$\text{return count}(a) + \eta$$

Implementation

```python
def dp_count(count):
    std = compute_std(epsilon, delta)
    noise = np.random.normal(scale=std)
    return count + noise

```

Auditing

There exists $(a, a') \in \mathcal{N}$ and S:

$$\Pr[M(a) \in S] > \exp(\epsilon)\Pr[M(a') \in S] + \delta$$

Jiankai Jin et al. “Are we there yet? timing and floating-point attacks on differential privacy systems.” (2021)

Why Audit Differential Privacy?

Definition

M is (ϵ, δ)-DP iff for all $(a, a') \in \mathcal{N}$ and S:

$$\Pr[M(a) \in S] \leq \exp(\epsilon)\Pr[M(a') \in S] + \delta$$

Theory - Gauss Mechanism

$$M_{\epsilon,\delta}(a):$$

$$\sigma = f(\epsilon, \delta)$$
$$\eta \sim \mathcal{N}(0, \sigma)$$
$$\text{return count}(a) + \eta$$

Implementation

```
def dp_count(count):
    std = compute_std(epsilon, delta)
    noise = np.random.normal(scale=std)
    return count + noise

```

Auditing with Delta-Siege

There exists $(a, a') \in \mathcal{N}$ and S:

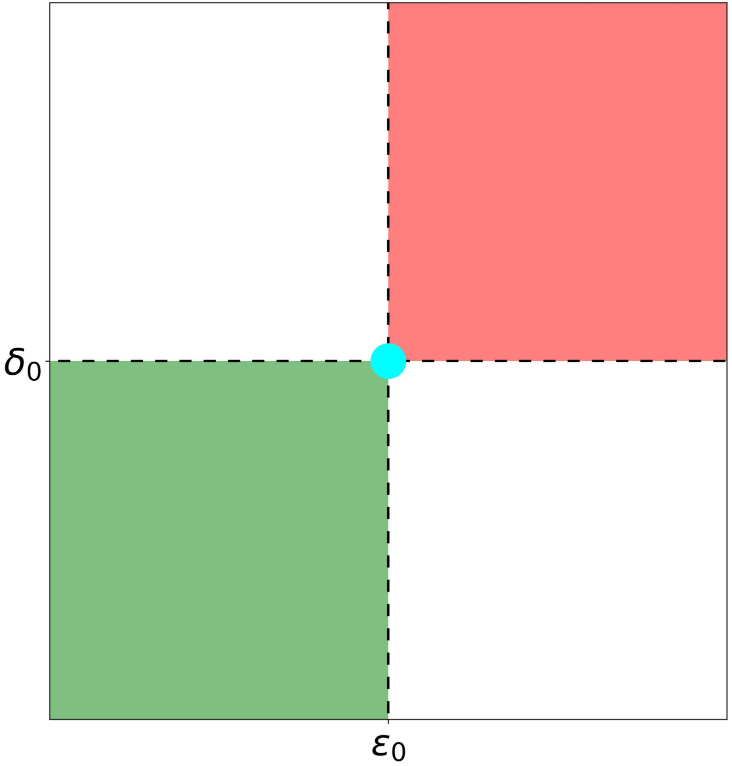
$$\Pr[M(a) \in S] > \exp(\epsilon)\Pr[M(a') \in S] + \delta$$

Jiankai Jin et al. "Are we there yet? timing and floating-point attacks on differential privacy systems." (2021)

Characterizing a Violation

Mechanism ( )

```python
import numpy as np

epsilon = 1e-1
delta = 1e-6

def dp_count(count):
    std = compute_std(epsilon, delta)
    noise = np.random.normal(scale=std)
    return count + noise
```



(ϵ_0, δ_0)

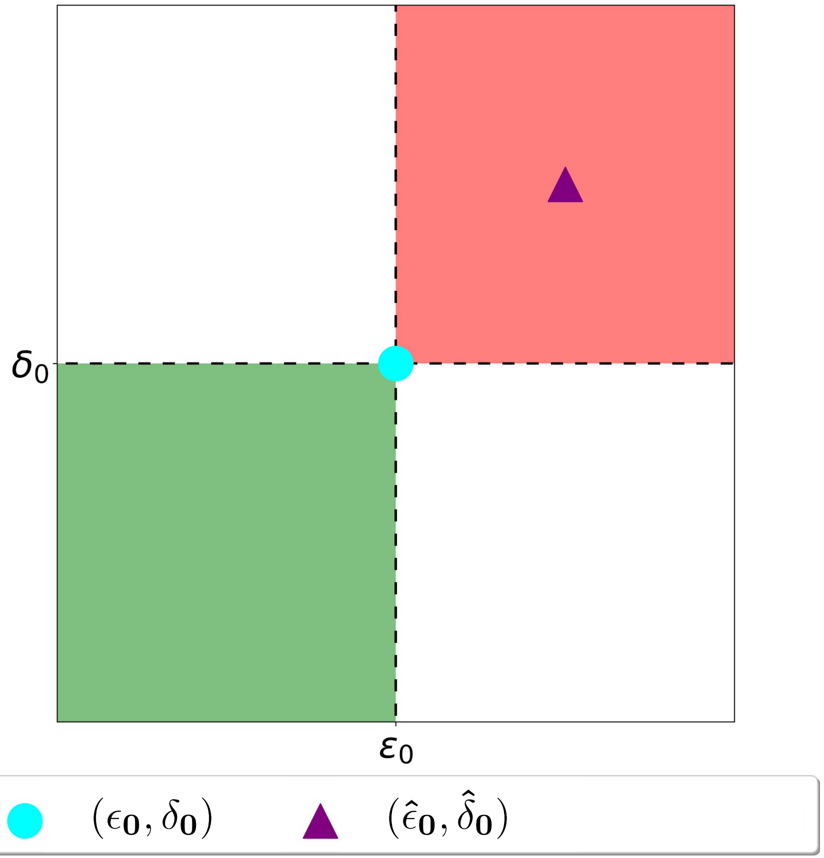
Characterizing a Violation

Mechanism (●)

```python
import numpy as np

epsilon = 1e-1
delta = 1e-6

def dp_count(count):
    std = compute_std(epsilon, delta)
    noise = np.random.normal(scale=std)
    return count + noise
```

Empirical Estimate

▲ $(\hat{\epsilon}_0, \hat{\delta}_0)$



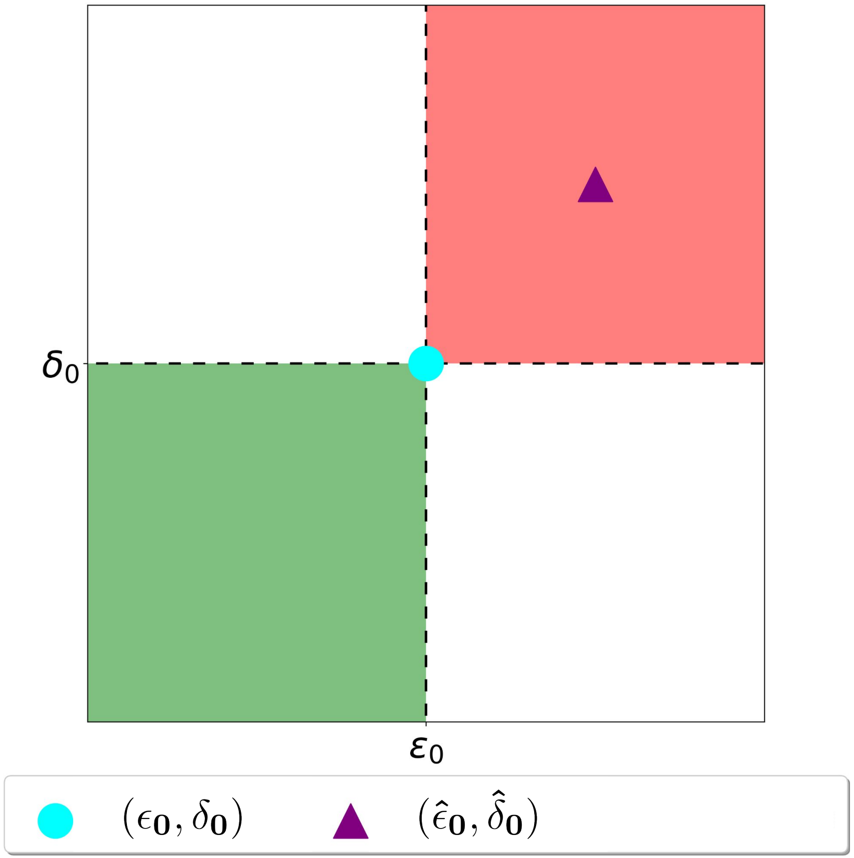
Characterizing a Violation

Mechanism ( )

```
import numpy as np

epsilon = 1e-1
delta = 1e-6

def dp_count(count):
    std = compute_std(epsilon, delta)
    noise = np.random.normal(scale=std)
    return count + noise
```

Empirical Estimate

$(\hat{\epsilon}_0, \hat{\delta}_0)$



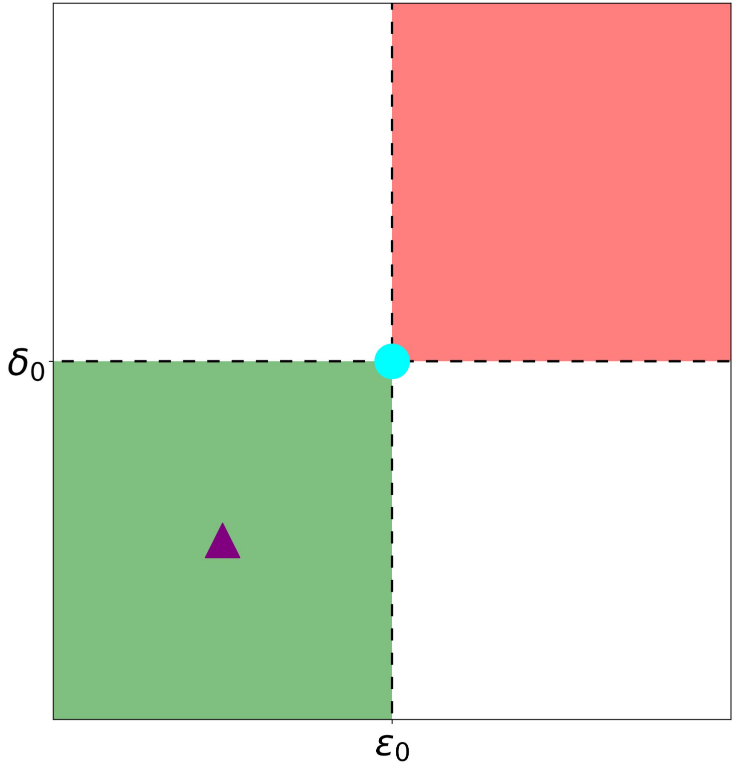
Characterizing a Violation

Mechanism ( )

```
import numpy as np

epsilon = 1e-1
delta = 1e-6

def dp_count(count):
    std = compute_std(epsilon, delta)
    noise = np.random.normal(scale=std)
    return count + noise
```

Empirical Estimate

$(\hat{\epsilon}_{\mathbf{0}}, \hat{\delta}_{\mathbf{0}})$



$(\epsilon_{\mathbf{0}}, \delta_{\mathbf{0}})$ $(\hat{\epsilon}_{\mathbf{0}}, \hat{\delta}_{\mathbf{0}})$

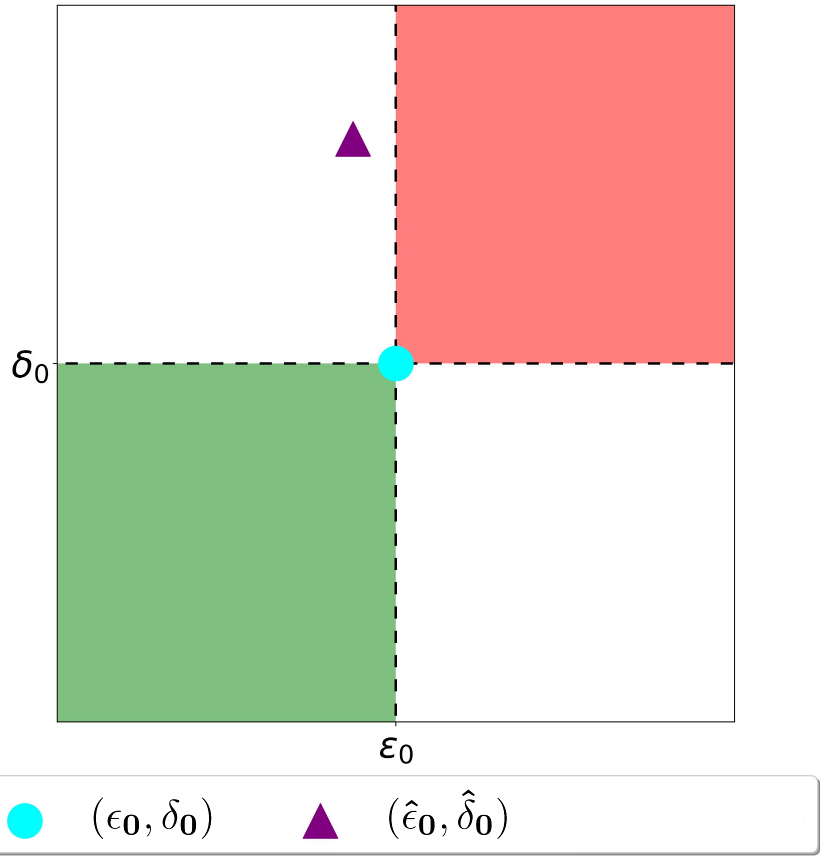
Characterizing a Violation

Mechanism (●)

```
import numpy as np

epsilon = 1e-1
delta = 1e-6

def dp_count(count):
    std = compute_std(epsilon, delta)
    noise = np.random.normal(scale=std)
    return count + noise

```

Empirical Estimate

▲ $(\hat{\epsilon}_0, \hat{\delta}_0)$



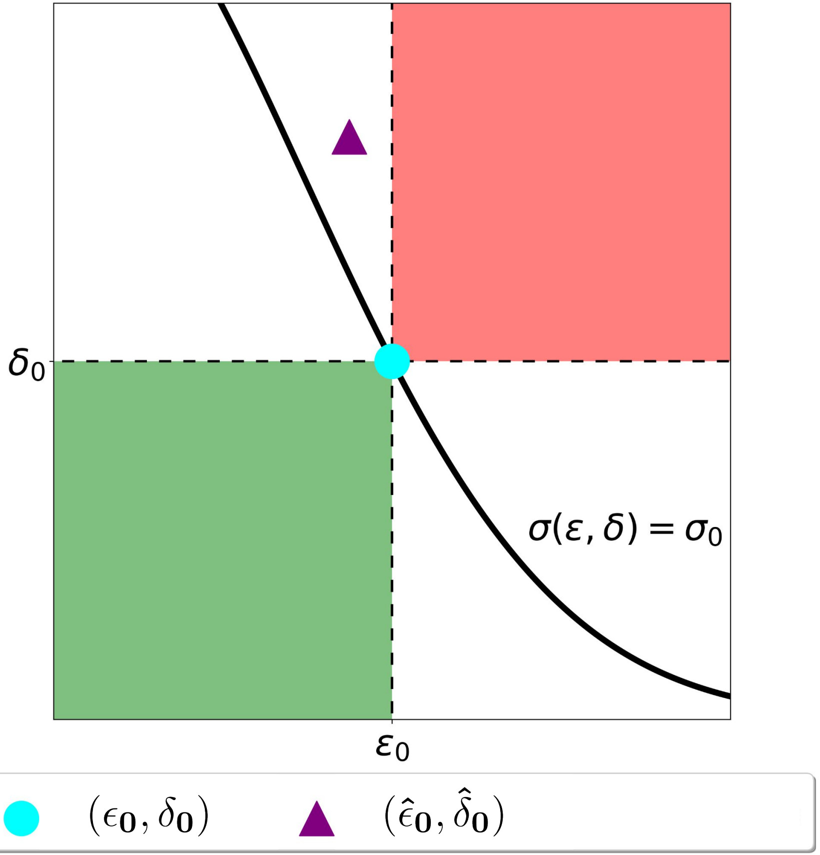
Mechanism

```python
import numpy as np

epsilon = 1e-1
delta = 1e-6

def dp_count(count):
    std = compute_std(epsilon, delta)
    noise = np.random.normal(scale=std)
    return count + noise

```

Mechanism

```
import numpy as np

epsilon = 1e-1
delta = 1e-6

def dp_count(count):
    std = compute_std(epsilon, delta)
    noise = np.random.normal(scale=std)
    return count + noise

```

Characterizing a Violation



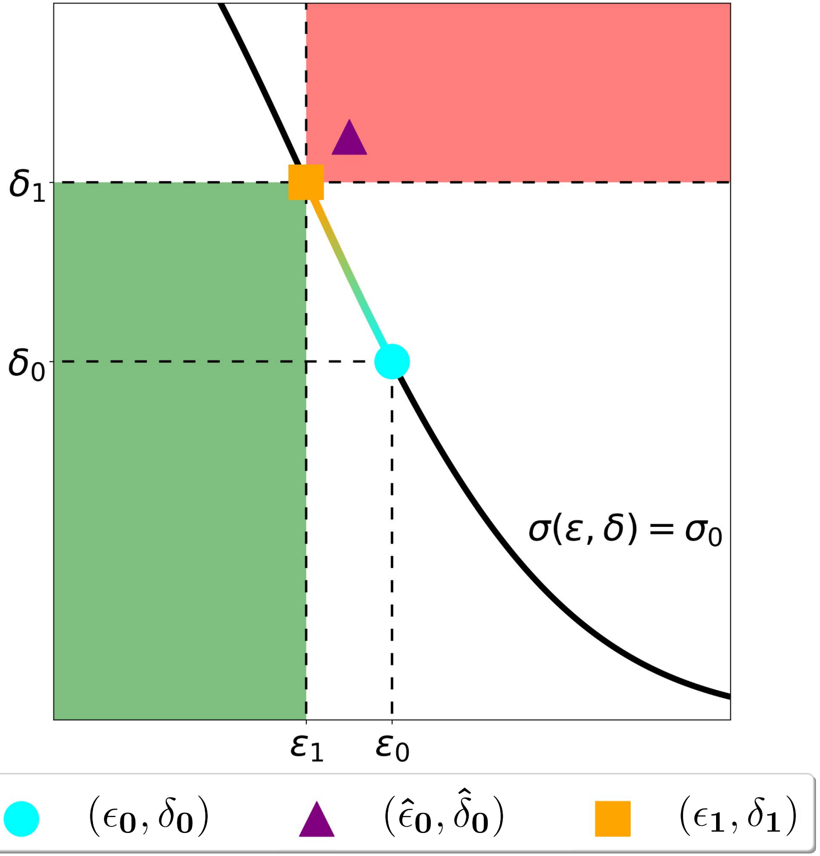
Characterizing a Violation

Mechanism ( )

```python
import numpy as np

epsilon = 1e-1
delta = 1e-6

def dp_count(count):
    std = compute_std(epsilon, delta)
    noise = np.random.normal(scale=std)
    return count + noise
```

Empirical Estimate

▲ $(\hat{\epsilon}_{\mathbf{0}}, \hat{\delta}_{\mathbf{0}})$



Characterizing a Violation

Mechanism (●)

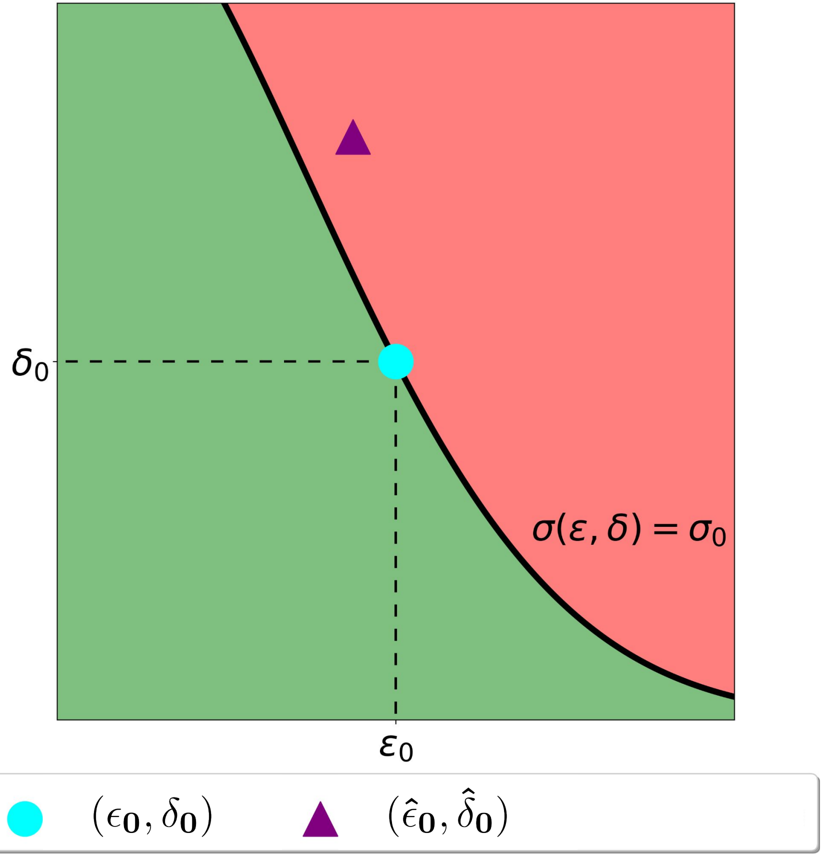
```python
import numpy as np

epsilon = 1e-1
delta = 1e-6

def dp_count(count):
    std = compute_std(epsilon, delta)
    noise = np.random.normal(scale=std)
    return count + noise

```

Empirical Estimate

▲ $(\hat{\epsilon}_{\mathbf{0}}, \hat{\delta}_{\mathbf{0}})$

Violation Found If

$\sigma(\blacktriangle) < \sigma(\bullet)$



Finding ▲

Goal: ▲ such that $\Pr[M(a) \in S] >> \Pr[M(a') \in S]$

Finding ▲

Goal: ▲ such that Pr[M(a)∈S]>>Pr[M(a')∈S]

Bichsel, Benjamin, et al. "Dp-sniper: Black-box discovery of differential privacy violations using classifiers."(2021)

Finding ▲

Goal: ▲ such that $\Pr[M(a) \in S] >> \Pr[M(a') \in S]$

Bichsel, Benjamin, et al. "Dp-sniper: Black-box discovery of differential privacy violations using classifiers."(2021)

0. Pick a, a’

Finding $\blacktriangle$

Goal: $\blacktriangle$ such that $\Pr[M(a) \in S] >> \Pr[M(a') \in S]$

Bichsel, Benjamin, et al. "Dp-sniper: Black-box discovery of differential privacy violations using classifiers."(2021)

0. Pick a, a'

1. Train a classifier $p(b) \simeq \Pr[A=a \mid M(A)=b]$

Finding ▲

Goal: ▲ such that Pr[M(a)∈S]>>Pr[M(a’)∈S]

Bichsel, Benjamin, et al. "Dp-sniper: Black-box discovery of differential privacy violations using classifiers."(2021)

0. Pick a, a’

1. Train a classifier p(b) ≃ Pr[A=a | M(A)=b]

2. Define S = {b | p(b) > 0.99}

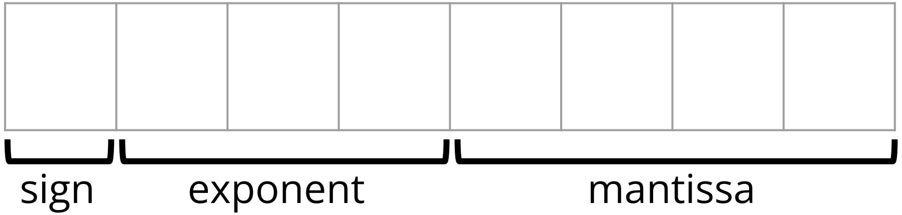
Finding the Root Cause

```
noise = np.random.normal(scale=std)
return count + noise
```

count(a) = 0, count(a’) = 1

p ← linear regression

Finding the Root Cause

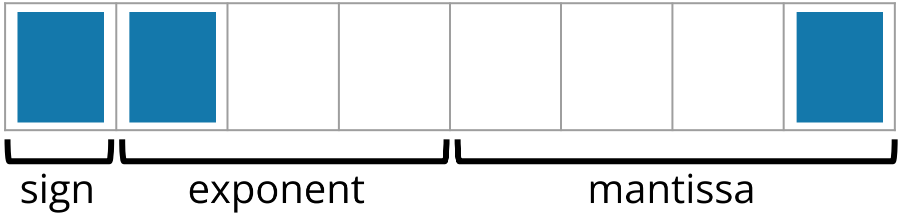



```python
noise = np.random.normal(scale=std)
return count + noise
```

count(a) = 0, count(a’) = 1

p ← linear regression

Finding the Root Cause




```
noise = np.random.normal(scale=std)
return count + noise
```

count(a) = 0, count(a’) = 1

p ← linear regression

Finding the Root Cause

1	0						1
sign	exponent			mantissa			

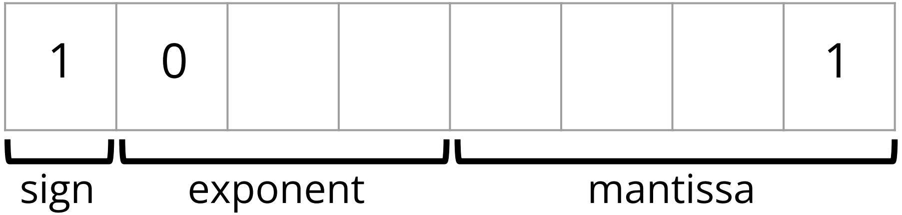
$$b = -\,1.xx1e^{\leq 0}$$

```python
noise = np.random.normal(scale=std)
return count + noise
```

count(a) = 0, count(a’) = 1

p ← linear regression

Finding the Root Cause



$$b = -\ 1.xx1e^{\leq 0}$$

Case a':

```
noise = np.random.normal(scale=std)
return count + noise
```

count(a) = 0, count(a') = 1

p ← linear regression

Finding the Root Cause

```python
noise = np.random.normal(scale=std)
return count + noise
```

count(a) = 0, count(a’) = 1

p ← linear regression

1	0						1

sign exponent mantissa

$$b = -\ 1.xx1e^{\leq 0}$$

Case a’:

Example noise = $-\ 1.1111e^{0}$

Finding the Root Cause

1	0						1

sign exponent mantissa

$$b = -\,1.xx1e^{\leq 0}$$

Case a':

Example noise = $-\,1.1111e^{0}$

$$\begin{array}{lll} & \text{count(a')} & 1.0000e^{0} \\ + & \text{noise} & \underline{-\,1.1111e^{0}} \end{array}$$

```
noise = np.random.normal(scale=std)
return count + noise
```

count(a) = 0, count(a') = 1

p ← linear regression

Finding the Root Cause

1	0						1

sign exponent mantissa

$$b = -\,1.xx1e^{\leq 0}$$

Case a':

Example noise = $-\,1.1111e^{0}$

$$\begin{array}{lll} & \text{count(a')} & 1.0000e^{0} \\ + & \text{noise} & \underline{-\,1.1111e^{0}} \\ & & -\,0.1111e^{0} \end{array}$$

```
noise = np.random.normal(scale=std)
return count + noise
```

count(a) = 0, count(a') = 1

p ← linear regression

Finding the Root Cause

1	0						1

sign exponent mantissa

$$b = -\,1.xx1e^{\leq 0}$$

```
noise = np.random.normal(scale=std)
return count + noise
```

count(a) = 0, count(a') = 1

p ← linear regression

Case a':

Example noise = $-\,1.1111e^{0}$

$$\begin{array}{lll} & \text{count(a')} & 1.0000e^{0} \\ + & \text{noise} & \underline{-\,1.1111e^{0}} \\ & & -\,0.1111e^{0} \\ & & -\,1.1110e^{-1} \end{array}$$

Finding the Root Cause

```
noise = np.random.normal(scale=std)
return count + noise
```

count(a) = 0, count(a’) = 1

p ← linear regression

1	0						1

$\underbrace{\hspace{2em}}_{\text{sign}}\ \underbrace{\hspace{6em}}_{\text{exponent}}\ \underbrace{\hspace{8em}}_{\text{mantissa}}$

$$b = -\ 1.xx1e^{\leq 0}$$

Case a’:

Example noise = $-\ 1.1111e^{0}$

$$\begin{array}{lll} & \text{count(a')} & 1.0000e^{0} \\ + & \text{noise} & \underline{-\ 1.1111e^{0}} \\ & & -\ 0.1111e^{0} \\ & & -\ 1.1110e^{-1} \end{array}$$

Finding the Root Cause

```python
noise = np.random.normal(scale=std)
return count + noise
```

count(a) = 0, count(a’) = 1

p ← linear regression

1	0						1
sign	exponent			mantissa			

$b = -\,1.xx1e^{\leq 0}$

Case a’:

Example noise = $-\,1.1111e^{0}$

$$
\begin{array}{lll}
 & \text{count(a')} & 1.0000e^{0} \\
+ & \text{noise} & \underline{-\,1.1111e^{0}} \\
 & & -\,0.1111e^{0} \\
 & & -\,1.1110e^{1}
\end{array}
$$

Case a:

Finding the Root Cause

```
noise = np.random.normal(scale=std)
return count + noise
```

count(a) = 0, count(a’) = 1

p ← linear regression

1	0						1

$\underbrace{\quad}_{\text{sign}}\underbrace{\qquad\qquad}_{\text{exponent}}\underbrace{\qquad\qquad\qquad}_{\text{mantissa}}$

$$b = -\,1.xx1e^{\leq 0}$$

Case a’:

Example noise = $-\,1.1111e^{0}$

$$\begin{array}{lll} & \text{count(a’)} & 1.0000e^{0} \\ + & \text{noise} & \underline{-\,1.1111e^{0}} \\ & & -\,0.1111e^{0} \\ & & -\,1.1110e^{-1} \end{array}$$

Case a:

$$\begin{array}{lll} & \text{count(a’)} & 0.0000e^{0} \\ + & \text{noise} & \underline{-\,1.1111e^{0}} \\ & & -\,1.1111e^{0} \end{array}$$

Finding the Root Cause

```
noise = np.random.normal(scale=std)
return count + noise
```

count(a) = 0, count(a') = 1

p ← linear regression

1	0						1

$\underbrace{}_{\text{sign}}\ \underbrace{}_{\text{exponent}}\ \underbrace{}_{\text{mantissa}}$

$$b = -\,1.xx1e^{\leq 0}$$

Case a':

Example noise = $-\,1.1111e^{0}$

$$\begin{array}{lll} & \text{count(a')} & 1.0000e^{0} \\ + & \text{noise} & \underline{-\,1.1111e^{0}} \\ & & -\,0.1111e^{0} \\ & & -\,1.1110e^{-1} \end{array}$$

Case a:

$$\begin{array}{lll} & \text{count(a')} & 0.0000e^{0} \\ + & \text{noise} & \underline{-\,1.1111e^{0}} \\ & & -\,1.1111e^{0} \end{array}$$

Finding the Root Cause

```
noise = np.random.normal(scale=std)
return count + noise
```

count(a) = 0, count(a’) = 1

p ← linear regression

1	0						1

sign | exponent | mantissa

$$b = -\,1.xx1e^{\leq 0}$$

Case a’:

Example noise = $-\,1.1111e^{0}$

$$\begin{array}{lll} & \text{count(a’)} & 1.0000e^{0} \\ + & \text{noise} & \underline{-\,1.1111e^{0}} \\ & & -\,0.1111e^{0} \\ & & -\,1.1110e^{-1} \end{array}$$

Case a:

$$\begin{array}{lll} & \text{count(a’)} & 0.0000e^{0} \\ + & \text{noise} & \underline{-\,1.1111e^{0}} \\ & & -\,1.1111e^{0} \end{array}$$

There are fixes!

Desfontaines, Damien, and Samuel Haney. "How to Break, Then Fix, Differential Privacy on Finite Computers." (2023).

Auditing Results

Mechanism	
OpenDP	Laplace Gauss
DiffPrivLib (IBM)	Laplace Float Gauss Analytic Float Gauss Discrete Gauss
PyDP	Laplace Gauss
Opacus (Fb)	Gauss
MST	
AIM	

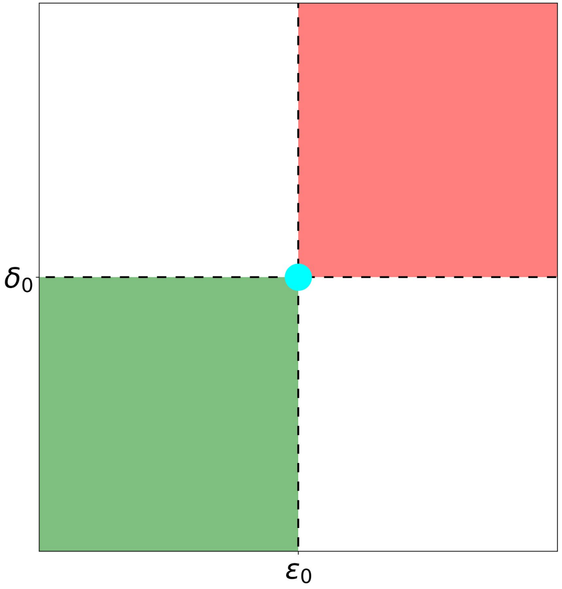
Auditing Results

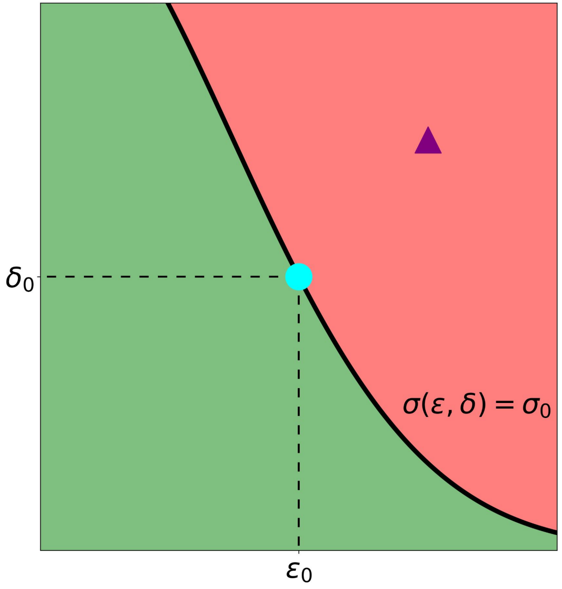
Mechanism		Violation?
OpenDP	Laplace	
	Gauss	
DiffPrivLib (IBM)	Laplace	✓
	Float Gauss	✓
	Analytic Float Gauss	✓
	Discrete Gauss	
PyDP	Laplace	
	Gauss	✓
Opacus (Fb)	Gauss	✓
MST		✓
AIM		✓

Auditing Results

Mechanism		Violation?
OpenDP	Laplace	
	Gauss	
DiffPrivLib (IBM)	Laplace	✓
	Float Gauss	✓
	Analytic Float Gauss	✓
	Discrete Gauss	
PyDP	Laplace	
	Gauss	$\boxed{\checkmark}$
Opacus (Fb)	Gauss	$\boxed{\checkmark}$
MST		$\boxed{\checkmark}$
AIM		$\boxed{\checkmark}$

Summary





New vulnerabilities found

Root cause analysis

eth-sri/Delta-Siege