

# fn cdp\_delta

Michael Shoemate

January 26, 2026

Proves soundness of `fn cdp_delta` in `cdp_delta.rs` at commit `0b8f4222` (outdated<sup>1</sup>). This proof is an adaptation of subsection 2.3 of [CKS20].

## 1 Bound Derivation

**Definition 1.1.** (Privacy Loss Random Variable). Let  $M : \mathcal{X}^n \rightarrow \mathbb{Y}$  be a randomized algorithm. Let  $x, x' \in \mathcal{X}^n$  be neighboring inputs. Define  $f : \mathcal{Y} \rightarrow \mathbb{R}$  by  $f(y) = \log\left(\frac{\mathbb{P}[M(x)=y]}{\mathbb{P}[M(x')=y]}\right)$ . Let  $Z = f(M(x))$ , the privacy loss random variable, denoted  $Z \leftarrow \text{PrivLoss}(M(x)||M(x'))$ .

**Lemma 1.2.** [CKS20] Let  $\epsilon, \delta \geq 0$ . Let  $M : \mathcal{X}^n \rightarrow \mathcal{Y}$  be a randomized algorithm. Then  $M$  satisfies  $(\epsilon, \delta)$ -differential privacy if and only if

$$\delta \geq \mathbb{E}_{Z \leftarrow \text{PrivLoss}(M(x)||M(x'))} [\max(0, 1 - e^{\epsilon - Z})] \quad (1)$$

(2)

for all  $x, x' \in \mathcal{X}^n$  differing on a single element.

*Proof.* Fix neighboring inputs  $x, x' \in \mathcal{X}^n$ . Let  $f : \mathcal{Y} \rightarrow \mathbb{R}$  be as in 1.1. For notational simplicity, let  $Y = M(x)$ ,  $Y' = M(x')$ ,  $Z = f(Y)$  and  $Z' = -f(Y')$ . This is equivalent to  $Z \leftarrow \text{PrivLoss}(M(x)||M(x'))$ . Our first goal is to prove that

$$\sup_{E \subset \mathcal{Y}} \mathbb{P}[Y \in E] - e^\epsilon \mathbb{P}[Y' \in E] = \mathbb{E}[\max\{0, 1 - e^{\epsilon - Z}\}]. \quad (3)$$

For any  $E \subset \mathcal{Y}$ , we have

$$\mathbb{P}[Y' \in E] = \mathbb{E}[\mathbb{I}[Y' \in E]] = \mathbb{E}[\mathbb{I}[Y \in E]e^{-f(Y)}]. \quad (4)$$

This is because  $e^{-f(y)} = \frac{\mathbb{P}[Y=y]}{\mathbb{P}[Y'=y]}$ .

Thus, for all  $E \subset \mathcal{Y}$ , we have

$$\mathbb{P}[Y \in E] - e^\epsilon \mathbb{P}[Y' \in E] = \mathbb{E}[\mathbb{I}[Y \in E](1 - e^{\epsilon - f(Y)})] \quad (5)$$

Now it is easy to identify the worst event as  $E = \{y \in \mathcal{Y} : 1 - e^{\epsilon - f(y)} > 0\}$ . Thus

$$\sup_{E \subset \mathcal{Y}} \mathbb{P}[Y \in E] - e^\epsilon \mathbb{P}[Y' \in E] = \mathbb{E}[\mathbb{I}[1 - e^{\epsilon - f(Y)} > 0](1 - e^{\epsilon - f(Y)})] = \mathbb{E}[\max\{0, 1 - e^{\epsilon - Z}\}] \quad (6)$$

□

---

<sup>1</sup>See new changes with `git diff 0b8f4222..`

`rust/src/combinators/measure_cast/zCDP_to_approxDP/cdp_delta/cdp_delta.rs`

**Theorem 1.3.** [CKS20] Let  $M : \mathcal{X}^n \rightarrow \mathcal{Y}$  be a randomized algorithm. Let  $\alpha \in (1, \infty)$  and  $\epsilon \geq 0$ . Suppose  $D_\alpha(M(x) || M(x')) \leq \tau$  for all  $x, x' \in \mathcal{X}^n$  differing in a single entry.<sup>2</sup> Then  $M$  is  $(\epsilon, \delta)$ -differentially private for

$$\delta = \frac{e^{(\alpha-1)(\tau-\epsilon)}}{\alpha-1} \left(1 - \frac{1}{\alpha}\right)^\alpha \quad (7)$$

*Proof.* Fix neighboring  $x, x' \in \mathcal{X}^n$  and let  $Z \leftarrow \text{PrivLoss}(M(x) || M(x'))$ . We have

$$\mathbb{E}[e^{(\alpha-1)Z}] = e^{(\alpha-1)D_\alpha(M(x) || M(x'))} \leq e^{(\alpha-1)\tau} \quad (8)$$

By 1.2, our goal is to prove that  $\delta \geq \mathbb{E}[\max\{0, 1 - e^{\epsilon-Z}\}]$ . Our approach is to pick  $c > 0$  such that  $\max\{0, 1 - e^{\epsilon-Z}\} \leq ce^{(\alpha-1)z}$  for all  $z \in \mathbb{R}$ . Then

$$\mathbb{E}[\max\{0, 1 - e^{\epsilon-Z}\}] \leq \mathbb{E}[ce^{(\alpha-1)z}] \leq ce^{(\alpha-1)\tau}. \quad (9)$$

We identify the smallest possible value of  $c$ :

$$c = \sup_{z \in \mathbb{R}} \frac{\max\{0, 1 - e^{\epsilon-z}\}}{e^{(\alpha-1)z}} = \sup_{z \in \mathbb{R}} e^{z-\alpha z} - e^{\epsilon-\alpha z} = \sup_{z \in \mathbb{R}} f(z) \quad (10)$$

where  $f(z) = e^{z-\alpha z} - e^{\epsilon-\alpha z}$ . We have

$$f'(z) = e^{z-\alpha z}(1-\alpha) - e^{\epsilon-\alpha z}(-\alpha) = e^{-\alpha z}(\alpha e^\epsilon - (\alpha-1)e^z) \quad (11)$$

Clearly  $f'(z) = 0 \iff e^z = \frac{\alpha}{\alpha-1}e^\epsilon \iff z = \epsilon - \log(1-1/\alpha)$ . Thus

$$c = f(\epsilon - \log(1-1/\alpha)) \quad (12)$$

$$= \left(\frac{\alpha}{\alpha-1}e^\epsilon\right)^{1-\alpha} - e^\epsilon \left(\frac{\alpha}{\alpha-1}e^\epsilon\right)^{-\alpha} \quad (13)$$

$$= \left(\frac{\alpha}{\alpha-1}e^\epsilon - e^\epsilon\right) \left(\frac{\alpha}{\alpha-1}e^{-\epsilon}\right)^\alpha \quad (14)$$

$$= \frac{e^\epsilon}{\alpha-1} \left(1 - \frac{1}{\alpha}\right)^\alpha e^{-\alpha\epsilon}. \quad (15)$$

Thus

$$\mathbb{E}[\max\{0, 1 - e^{\epsilon-Z}\}] \leq \frac{e^\epsilon}{\alpha-1} \left(1 - \frac{1}{\alpha}\right)^\alpha e^{-\alpha\epsilon} e^{(\alpha-1)\tau} = \frac{e^{(\alpha-1)(\tau-\epsilon)}}{\alpha-1} \left(1 - \frac{1}{\alpha}\right)^\alpha = \delta \quad (16)$$

□

**Corollary 1.** [CKS20] Let  $M : \mathcal{X}^n \rightarrow \mathcal{Y}$  be a randomized algorithm. Let  $\alpha \in (1, \infty)$  and  $\epsilon \geq 0$ . Suppose  $D_\alpha(M(x) || M(x')) \leq \tau$  for all  $x, x' \in \mathcal{X}^n$  differing in a single entry. Then  $M$  is  $(\epsilon, \delta)$ -differentially private for

$$\epsilon = \tau + \frac{\ln(1/\delta) + (\alpha-1)\ln(1-1/\alpha) - \ln(\alpha)}{\alpha-1} \quad (17)$$

*Proof.* This follows by rearranging 1.3. □

**Corollary 2.** Let  $M : \mathcal{X}^n \rightarrow \mathcal{Y}$  be a randomized algorithm satisfying  $\rho$ -concentrated differential privacy. Then  $M$  is  $(\epsilon, \delta)$ -differentially private for any  $0 < \delta \leq 1$  and

$$\epsilon = \inf_{\alpha \in (1, \infty)} \alpha \rho + \frac{\ln(1/\delta) + (\alpha-1)\ln(1-1/\alpha) - \ln(\alpha)}{\alpha-1} \quad (18)$$

*Proof.* This follows from 1 by taking the infimum over all divergence parameters  $\alpha$ . □

<sup>2</sup>This is the definition of  $(\alpha, \tau)$ -Rényi differential privacy.

## 2 Pseudocode

### Precondition

None.

### Implementation

```
1 def cdp_delta(rho: float, eps: float) -> float:
2     """The Rust code may be easier to follow due to more commenting."""
3     if rho.is_sign_negative():
4         raise ValueError(f"rho ({rho}) must be non-negative")
5
6     if eps.is_sign_negative():
7         raise ValueError(f"epsilon ({eps}) must be non-negative")
8
9     if rho.is_zero() or eps.is_infinite():
10        return 0.0
11
12    if rho.is_infinite():
13        return 1.0
14
15    a_max = eps.inf_add(1.0).inf_div((2.0).neg_inf_mul(rho)).inf_add(2.0)
16
17    a_min = 1.01
18
19    while True:
20        diff = a_max - a_min
21
22        a_mid = a_min + diff / 2.0
23
24        if a_mid == a_max or a_mid == a_min:
25            break
26
27        # calculate derivative
28        deriv = (2.0 * a_mid - 1.0) * rho - eps + a_mid.recip().neg().ln_1p()
29
30        if deriv.is_sign_negative():
31            a_min = a_mid
32        else:
33            a_max = a_mid
34
35        # calculate delta
36        a_1 = a_max.inf_sub(1.0)
37        ar_e = a_max.inf_mul(rho).inf_sub(eps)
38
39    try:
40        t1 = a_1.inf_mul(ar_e)
41
42    except OpenDPEException:
43
44        # if t1 is negative, then handle negative overflow by making t1 larger: the most
45        # negative finite float
46        # making t1 larger makes delta larger, so it's still a valid upper bound
47        if a_1.is_sign_negative() != ar_e.is_sign_negative():
48            t1 = 1.7976931348623157e308 # f64::MIN
49        else:
50            raise
51
52    t2 = a_max.inf_mul(a_max.recip().neg().inf_ln_1p())
53
54    delta = t1.inf_add(t2).inf_exp().inf_div((a_max.inf_sub(1.0)))
```

```

55     # delta is always <= 1
56     delta.min(1.0)

```

### Postcondition

**Theorem 2.1.** For any possible setting of  $\rho$  and  $\epsilon$ , `cdp_delta` either returns an error, or a  $\delta$  such that any  $\rho$ -differentially private measurement is also  $(\epsilon, \delta)$ -differentially private.

## 3 Proof

*Proof.* The code always finds an  $\alpha_* \approx \mathbf{a\_max} \geq 1.01$ . Since  $\mathbf{a\_max} \in (1, \infty)$ , then by 1, any  $\rho$ -differentially private measurement is also  $(\epsilon(\mathbf{a\_max}), \delta)$ -differentially private. Define  $\delta_{cons}(\alpha)$  as a “conservative” function for computing  $\delta(\epsilon)$ , where floating-point arithmetic is computed with conservative rounding such that  $\delta_{cons}(\alpha) \geq \delta(\alpha)$  for  $\forall \alpha \in (1, \infty)$ . Since  $\mathbf{delta} = \delta_{cons}(\mathbf{a\_max}) \geq \delta(\mathbf{a\_max})$ , then any  $(\epsilon, \delta(\mathbf{a\_max}))$ -differentially private measurement is also  $(\epsilon, \delta)$ -differentially private.  $\square$

## References

- [CKS20] Clément L. Canonne, Gautam Kamath, and Thomas Steinke. The discrete gaussian for differential privacy. *CoRR*, abs/2004.00010, 2020.