

trait impl PSRN for TulapPSRN

Yu-Ju Ku, Jordan Awan, Aishwarya Ramasethu, Michael Shoemate

August 12, 2024

This proof resides in “**contrib**” because it has not completed the vetting process.

Proves soundness of **TulapPSRN**. **edge** accepts parameter **self**, containing the state of the Tulap sampler and **R** specifying the rounding mode.

This implementation is susceptible to floating-point vulnerabilities.

Warning 1 (Code is not constant-time). The implementation of **edge** uses procedures that are vulnerable to timing attacks.

PR History

- [Pull Request #1126](#)

1 Hoare Triple

Preconditions

- Variable **self** is of type **TulapPSRN**.
- Generic **R** denotes the rounding mode, one of "up" or "down".

Pseudocode

```
1
2 class TulapPSRN(object):
3     def __init__(self, shift, epsilon, delta) -> None:
4         self.shift = shift
5         self.exp_eps = Fraction(epsilon.neg_inf_exp())
6         self.exp_neg_eps = Fraction((-epsilon).inf_exp())
7         self.c = (1 - delta) / (1 + self.exp_eps)
8         self.delta = delta
9         self.uniform = UniformPSRN()
10
11         if c >= 0.5:
12             raise ValueError("c must be less than 1/2")
13
14     def q_cnd(self, unif) -> Fraction | None: # CND quantile function for f
15         if unif < c:
16             return self.q_cnd(1 - self.f(unif)) - 1
17         elif unif <= 1 - self.c: # the linear function
```

```

18         num = unif - 1 / 2
19         den = 1 - 2 * self.c
20         if den.is_zero():
21             return
22         return num / den
23     else:
24         return self.q_cnd(self.f(1 - unif)) + 1
25
26     def f(self, unif):
27         t1 = 1 - self.delta - self.exp_eps * unif
28         t2 = self.exp_neg_eps * (1 - self.delta - unif)
29         return max(t1, t2, 0)
30
31     def edge(self, R):
32         return self.q_cnd(self.uniform.edge(R)) + self.shift
33
34     def refine(self):
35         self.uniform.refine()
36
37     def refinements(self):
38         return self.uniform.refinements()

```

Postcondition

`edge` returns an upper or lower bound for the true Tulap sample, a distribution with CDF defined in `make_tulap`.

2 Proof

Proof.

Proposition 1. *The quantile function $F_f^{-1} : (0, 1) \rightarrow \mathbb{R}$ for F_f can be expressed as*

$$F_f^{-1}(u) = \begin{cases} F_f^{-1}(1 - f(u)) - 1 & u < c \\ \frac{u-1/2}{1-2c} & c \leq u \leq 1 - c \\ F_f^{-1}(f(1 - u)) + 1 & u > 1 - c, \end{cases}$$

where c is the unique fixed point of f . Furthermore, for any $u \in (0, 1)$, the expression $Q_f(u)$ takes a finite number of recursive steps to evaluate. Thus, if $U \sim U(0, 1)$, then $F_f^{-1}(U) \sim F_f$.

The cdf of $\text{Tulap}(0, b, q)$ is

$$F_N(x) = \begin{cases} 0 & F_{N_0}(x) < q/2 \\ \frac{F_{N_0}(x) - q/2}{1 - q} & q/2 \leq F_{N_0}(x) \leq 1 - q/2 \\ 1 & F_{N_0}(x) > 1 - q/2. \end{cases}$$

By inspection, the fixed point of $f_{\epsilon, \delta}$ is $c = \frac{1-\delta}{1+\epsilon}$. It is easy to verify that $F_N(x) = c(1/2 - x) + (1 - c)(x + 1/2)$ for $x \in (-1/2, 1/2)$.

The function then uses the inverse transform of a sample of a uniform RV to sample a Tulap RV centered at zero. Arbitrarily precise estimates of the lower and upper bound of the uniform sample can be retrieved, by the postcondition of `UniformPSRN`. $F_N(x)$ is computed conservatively, where the values of b and q are computed according to privacy parameters that are no greater than ϵ, δ .

The computation of F_f^{-1} is handled exactly via fractional arithmetic, as it involves no transcendental functions.

The function then returns the outcome, shifted by `self.shift`, a sample from $\text{Tulap}(\text{shift}, b, q)$, where $b = \exp(-\epsilon)$ and $q = \frac{2\delta b}{1 - b + 2\delta b}$.

□