fn make_randomized_response_bool

Vicki Xu, Hanwen Zhang, Zachary Ratliff

October 21, 2024

This implementation contains partial mitigations for timing side-channel vulnerabilities. Timing side-channel vulnerabilities may still be exploitable due to branching, compiler- and instruction-level optimizations, caching, and so on.

Proves soundness of `make_randomized_response_bool` in `mod.rs` at commit f5bb719 (outdated[1]).

`make_randomized_response_bool` accepts a float parameter `prob` and a bool parameter `mitigate_timing`. The function on the resulting measurement takes in a boolean data point `arg` and returns the truthful value `arg` with probability `prob`, or the complement `!arg` with probability $1 -$ `prob`. The measurement function makes mitigations against timing channels if `mitigate_timing` is set.

1 Hoare Triple

Preconditions

Compiler-verified

- Argument `prob` must be of type `f64`
- Argument `mitigate_timing` must be of type `bool`

Human-verified None

Pseudocode

```python
def make_randomized_response_bool(prob: f64, mitigate_timing: bool):
    input_domain = AtomDomain(bool)
    output_domain = AtomDomain(bool)
    input_metric = DiscreteMetric()
    output_measure = MaxDivergence()

    if (prob < 0.5 or prob >= 1):  #
        raise Exception("probability must be in [0.5, 1)")

    c = prob.inf_div((1).neg_inf_sub(prob)).inf_ln()
    def privacy_map(d_in: u32) -> f64:  #
        if d_in == 0:
            return 0
        else:
            return c

    def function(arg: bool) -> bool:  #
        sample = not sample_bernoulli_float(prob, mitigate_timing)
```

[1]See new changes with `git diff f5bb719..2c38b790 rust/src/measurements/randomized_response/mod.rs`

```
        return arg ^ sample

    return Measurement(input_domain, output_domain, function, input_metric, output_measure,
    privacy_map)
```

Postcondition

Theorem 1.1. For every setting of the input parameters (`prob, mitigate_timing`) to `make_randomized_response_bool` such that the given preconditions hold, `make_randomized_response_bool` raises an exception (at compile time or run time) or returns a valid measurement. A valid measurement has the following property:

1. (Privacy guarantee). For every pair of elements x, x' in `input_domain` and for every pair (`d_in`, `d_out`), where `d_in` has the associated type for `input_metric` and `d_out` has the associated type for `output_measure`, if x, x' are `d_in`-close under `input_metric`, `privacy_map`(`d_in`) does not raise an exception, and `privacy_map`(`d_in`) $\leq$ `d_out`, then `function`(x), `function`(x') are `d_out`-close under `output_measure`.

2 Proof

Proof. **(Privacy guarantee.)**

> **Note 1.** The following proof makes use of the following lemma that asserts the correctness of a Bernoulli sampler function.
>
> **Lemma 2.1.** `sample_bernoulli_float` satisfies its postcondition.

`sample_bernoulli` can only fail due to lack of system entropy. This is usually related to the computer's physical environment and not the dataset. The rest of this proof is conditioned on the assumption that `sample_bernoulli` does not raise an exception.

Let x and x' be datasets in the input domain (either $\top$ or $\bot$) that are `d_in`-close with respect to `input_metric`. Here, the metric is `DiscreteMetric` which enforces that `d_in` ≥ 1 if $x \neq x'$ and `d_in` $= 0$ if $x = x'$. If $x = x'$, then the output distributions on x and x' are identical, and therefore the max-divergence is 0. Consider $x \neq x'$ and assume without loss of generality that $x = \top$ and $x' = \bot$. For shorthand, we let p represent `prob`, the probability that `sample_bernoulli_float` returns $\top$. Observe that $p = [0.5, 1.0)$ otherwise `make_randomized_response_bool` raises an error.

We now consider the max-divergence $D_\infty(Y||Y')$ over the random variables $Y =$ `function`(x) and $Y' =$ `function`(x').

$$
\begin{aligned}
\max_{x\sim x'} D_\infty(Y||Y') &= \max_{x\sim x'} \max_{S\subseteq Supp(Y)} \ln\left(\frac{\Pr[Y\in S]}{\Pr[Y'\in S]}\right) \\
&\leq \max_{x\sim x'} \max_{y\in Supp(Y)} \ln\left(\frac{\Pr[Y=y]}{\Pr[Y'=y]}\right) && \text{Lemma 3.3 [1]} \\
&= \max_{x\sim x'} \max\left(\ln\left(\frac{\Pr[Y=\top]}{\Pr[Y'=\top]}\right), \ln\left(\frac{\Pr[Y=\bot]}{\Pr[Y'=\bot]}\right)\right) \\
&= \max\left(\ln\left(\frac{p}{1-p}\right), \ln\left(\frac{1-p}{p}\right)\right) \\
&= \ln\left(\frac{p}{1-p}\right)
\end{aligned}
$$

We let $c =$ `privacy_map`(`d_in`) $=$ `prob.inf_div(1.neg_inf_sub(prob)).inf_ln()`. The computation of `c` rounds upward in the presence of floating point rounding errors. This is because `1.neg_inf_sub(prob)` appears in the denominator, and to ensure that the bound holds even in the presence of rounding errors, the conservative choice is to round down (so the quantity as a whole is bounded above). Similarly, `inf_div` and `inf_ln` round up.

When `d_in` > 0 and no exception is raised in computing `c` $=$ `privacy_map`(`d_in`), then $\ln\left(\frac{p}{1-p}\right) \leq$ `c`.

Therefore we've shown that for every pair of elements $x, x' \in \{\bot, \top\}$ and every $d_{DM}(x, x') \leq$ `d_in` with `d_in` ≥ 0, if x, x' are `d_in`-close then `function`(x), `function`$(x') \in \{\bot, \top\}$ are `privacy_map`(`d_in`)-close under `output_measure` (the Max-Divergence). $\square$

References

[1] Shiva P. Kasiviswanathan and Adam Smith. On the "semantics" of differential privacy: A bayesian formulation. *Journal of Privacy and Confidentiality*, 6(1), June 2014.