wrapper

September 18, 2021

[ ]:
```
import random
from string import ascii_letters, digits
from hashlib import sha256
import numpy as np
import plotly.express as px
import pandas as pd
```

[ ]:
```
ALPHANUM: str = ascii_letters + digits


def hex2bin(data: hex) -> bin:
    return bin(int(data, 16))


def truncateHex(data: hex, bits: int = 256) -> hex:
    return hex(int(hex2bin(data)[0:bits+2], 2))


def ranString(length: int = 16) -> str:
    return ''.join(random.choice(ALPHANUM) for i in range(length))


def sha256w(data: str = "", bits: int = 256) -> str:

    sha = sha256(data.encode())
    enc = truncateHex(sha.hexdigest(), bits)
    return enc
```

[ ]:
```
def collision() -> dict[int, list[int]]:
    bits = [8, 10, 12, 14, 16, 18, 20, 22]
    hashes = []
    bits = {size: [] for size in bits}
    tries = 0

    for size in bits.keys():
        for i in range(75):
            base = ranString(25)
```

```
            res = sha256w(base, size)
            hashes = [res]

            while True:
                tries += 1

                alt = ranString(25)
                test = sha256w(alt, size)
                if test in hashes:
                    break
                else:
                    hashes.append(test)

            bits[size].append(tries)
            tries = 0
    return bits

col_trials = collision()
```

[ ]:
```
def preImage() -> dict[int, list[int]]:
    bits = [8, 10, 12, 14, 16, 18, 20, 22]
    bits = {size: [] for size in bits}
    tries = 0

    for size in bits.keys():
        for i in range(75):
            base = ranString(25)
            res = sha256w(base, size)
            while True:
                tries += 1
                alt = ranString(25)
                test = sha256w(alt, size)
                if res == test:
                    break
            # print(
            #     f"Pre-image attack with bit length {size} took {tries} tries")
            bits[size].append(tries)
            tries = 0

    return bits



pre_trials = preImage()
```

[ ]:
```
# print(pre_trials)
pre_expected = {
```

```
    x: 2**x for x in range(8, 23, 2)
}
print(pre_expected)

col_expected = {
    x: 2**(int(x/2)) for x in range(8, 23, 2)
}
print(col_expected)
```

```
{8: 256, 10: 1024, 12: 4096, 14: 16384, 16: 65536, 18: 262144, 20: 1048576, 22:
4194304}
{8: 16, 10: 32, 12: 64, 14: 128, 16: 256, 18: 512, 20: 1024, 22: 2048}
```

[ ]:
```
import plotly.graph_objects as go

preDF = pd.DataFrame.from_dict(pre_trials)

fig = px.box(preDF, points='outliers', log_y=True, range_y=(1, 20000000),␣
↪title="Pre-Image Attack Iteration Results",
                    labels={'variable': 'Bit Sizes', 'value': '# of␣
↪Iterations'}, )
fig.add_scatter(x=list(pre_expected.keys()), y=[np.mean(data) for data in␣
↪pre_trials.values()], name='Avg Iterations')
fig.add_scatter(x=list(pre_expected.keys()), y=list(pre_expected.values()),␣
↪name='Expected Iterations')
fig.show()
```

[ ]:
```
colDF = pd.DataFrame(col_trials)

fig = px.box(colDF, points='outliers', log_y=True, range_y=(1, 5000),␣
↪title="Collision Attack Iteration Results",
                    labels={'variable': 'Bit Sizes', 'value': '# of␣
↪Iterations'}, )
fig.add_scatter(x=list(col_expected.keys()), y=[np.mean(data) for data in␣
↪col_trials.values()], name='Avg Iterations')
fig.add_scatter(x=list(col_expected.keys()), y=list(col_expected.values()),␣
↪name='Expected Iterations')
fig.show()
```