Casino

writeup by manf

Challenge

We are given the following values:

```
challenge = rng.randint(0, 1)

a, b, z = rng.randint(1, prime-1), rng.randint(1, prime-1), rng.randint(1, prime-1)

A, B, C, Z = pow(generator, a, prime), pow(generator, b, prime),
             pow(generator, a*b, prime), pow(generator, z, prime)

print(f"""Guess the random bit I have coosen!
Commitment: {A}, {B}, {C if challenge == 1 else Z}""")
```

and have to guess the value of `challenge`. If we guess correctly, our money (initially equal to `100`) goes up by one, if we guess incorrectly, it is decreased by one. The goal is to reach a balance of `200` (without ever going into debt).

Intended solution

It is easy to find out (by factoring $p-1$) that the given generator g has full order $p-1$ modulo the used prime p - i.e., $p^a \neq 1 \mod p$ for any \$1 < a < p \$. Therefore, we are are actually working in the *cyclic group* $\mathbb{Z}/(p-1)\mathbb{Z}$, which is isomorphic to our multiplicative group $(\mathbb{Z}/p\mathbb{Z})^*$.

For group-based cryptographic schemes, it is always important to understand the actual group structure of the group one is using, and to think about any ways it can be used to "move" the problem to a "smaller one". For example, the famous Pohlig-Hellman algorithm is based on this fact: if $p-1$ has many small factors, any discrete logarithm problem can be moved via the right homomorphism to a set of much smaller groups and solved there.

In this case, $p-1$ factors as $p-1 = 2 \cdot q$ with q a large prime - large enough that computing discrete logarithms is clearly out of reach. However, we aren't actually interested in the discrete logarithm: The problem as posed is a *decisional Diffie-Hellman* game which asks us to distinguish triples (g^a, g^b, g^{ab}) from ones of the form (g^a, g^b, g^c) with (a, b, c) being chosen indepently and uniformly random. It's a fundamental cryptographic assumption that this game is hard in a well-chosen group, like a modular multiplicative group of large prime order, or an elliptic curve.

However, the group from the problem is not one of those. Going back to the group structure, we can clearly factor the group as $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/q\mathbb{Q}$. While this group doesn't factor neatly into small factors, it does at least have *one* very small factor of order 2. We also get an easily computable projection homomorphism into this factor: Simply taking the map $f : x \mapsto x^q$ sends each element to one of order at most $(p-1)/q = 2$; it is easily seen that this is a non-trivial homomorphism $(\mathbb{Z}/p\mathbb{Z})^* \mapsto (\{1, -1\}, *)$ (for example, $f(-1)$ is clearly -1). It follows by a basic theorem of group theory that f sends exactly half of the values to $+1$ and half to -1.

But now f sends Diffie-Hellman-Triples (triples of the form (g^a, g^b, g^{ab})) to Diffie-Hellman-Triples: Clearly, $f(g^a) = f(g)^a$, $f(g^b) = f(g)^b$, $f(g^{ab}) = f(g)^{ab}$. Therefore, the problem is now reduced to $\mathbb{Z}/2\mathbb{Z}$, where it can be solved by simple enumeration: Only 4 of the 8 possible triples are Diffie-Hellman-Triples; if we get any other, we can confidently answer with "No". If we get one of those 4, we can answer with "Yes". This means we have a 100% success rate if the real answer is "Yes" (as we will always err on the side of possibly being a DH-triple), and a 50% success rate if the real answer is "No" (50% of those will randomly be valid triples), giving an overall success rate of 75%. The resulting expected value of +0.5 in every round is enough to finish the game quite quickly.

All of this can also be written - perhabs a bit more succintly - in terms of quadractic (non)residues or the legendre symbol. However, I much prefer this way of thinking about subgroups and homomorphisms: It reveals a much more general framework to tackle problems like this, which still works if the small factor is not just equal to 2.

Trivial solution

The challenge had an unintended solution as well. Consider the case of simply sending random bits (or alternatively, just 0s or just 1s), giving us a win chance of 50% at every point. This means that our balance forms an unbiased, one-dimensional random walk: If A_n is the available money at step n, then cleary $P[A_{n+1} = A_n + 1] = P[A_{n+1} = A_n - 1] = 0.5$. This random walk will continue until our money either reaches 0 or 200.

First, let's consider the chances of this scheme eventually working. The easiest approach is to consider symmetry: As we start out with $100 = \frac{0+200}{2}$ money, the entire situation is symmetrical when reflecting at 100. It easily follows that the probability of stopping at 200 is equal to the one of stopping at 0 - exactly 1/2.

In a slightly more general setting, assume that we want to reach a value of $k \cdot 100$. As there is no house edge, every gamble is expectation value neutral - that is, $E(A_{n+1}|A_1, A_2, \dots, A_n) = A_n$. In other words, the process forms a *martingale*. The optional stopping theorem applies: If τ is the time where we first either run out of money or reach our target, then $E(A_\tau) = E(A_1) = 100$. As clearly $E(A_\tau) = p \cdot (k \cdot 100) + (1-p) \cdot 0$ where p is our success probability, it follows that $p = \frac{1}{k}$.

But the probability of success is only one part of the story - let's now estimate the running time of this solution. First, notice that we can write A_n as $\sum_{i=1}^n X_i$ with each X_i independantly and fairly sampled from $\{1, -1\}$. By the central limit theorem, for n large enough A_n will be distributed approximately as a normal distribution with mean $\mu = E(A_n) = E(A_1) = 100$ and variance $\sigma^2 = \sqrt{n} \cdot 1$.

It is well-known, that for a normally distributed variable X with mean μ and variance σ^2 we have $P[\mu - \sigma \leq X \leq \mu + \sigma] \approx 0.68$; thefore, if we choose $n \geq 100^2 = 10000$, we have at least roughly a 68% chance of stopping in the first n steps (this somewhat undercounts the possibilities for stopping, as every path that takes us outside $[0, 200]$ *and then back in again* before time n will also lead to stopping).

In total, this proves that we will need in expectation only about $\frac{1}{0.68 \cdot 0.5} \approx 3$ tries of sending 10000 random bits before winning the game. An analysis for the asymmetric case can be done similarly.