Brief: Shor-Style ECC Demo on IBM Quantum (ROM Oracle)

Author: Joseph “Joey” Fredrickson • **Email:** <your.email@domain> • **Date:** 2025-11-08

1. Objective

Demonstrate a verified recovery of an ECC private key on real quantum hardware using a Shor-style pipeline on tiny curves, with a compact oracle and robust post-processing tailored to hardware constraints.

2. Method Overview

- **State prep.** Uniform superposition over two address registers `(a,b)` sized to `ceil(log2 n)` where `n` is the group order of `G`.
- **Oracle.** A ROM-style oracle emits packed residues of the x-coordinate of `aG + bQ` modulo small integers (e.g., `{32,7}`) across a single horizon row; output is measured.
- **Fourier step.** Apply an approximate QFT (degree 1) on both `(a,b)` to reduce two-qubit depth; measure `(r,s)`.
- **Post-processing.** From `(r,s)`, recover `d` via voting across **four** linear congruences to resolve sign/wiring ambiguities:
- `r + s·d ≡ 0 (mod n)`, `r - s·d ≡ 0 (mod n)`,
- `s + r·d ≡ 0 (mod n)`, `s - r·d ≡ 0 (mod n)`.

Discard samples with `r≡0` or `s≡0 (mod n)` to avoid the known `2^k→n` zero-frequency bias.

3. Implementation Notes

- **Circuit depth.** Keep `horizon=1`, small moduli, and QFT approximation degree 1; transpile at optimization level 1.
- **Runtime.** Use IBM Runtime Sampler (Batch mode) to submit and retrieve results cleanly; report job IDs.
- **Verification.** On tiny curves, verify correctness by computing `dG` and checking equality with `Q` exactly on-curve.

4. Results (Hardware)

- **Curve[1]** (p=43, G=(34,3), Q=(21,25), order n=31): recovered `d=18`, verified

`dG==Q` True, shots 2048 (backend `ibm_torino`).

- **Curve[0]** (p=13, G=(11,5), Q=(11,8), order n=7): recovered `d=6`, verified `dG==Q` True, shots 1024 (backend `ibm_torino`).

5. Requirements & Drawbacks

- **Requirements.** IBM Quantum account; `qiskit`, `qiskit-aer`, `qiskit-ibm-runtime`; Python 3.10+ recommended.
- **Drawbacks.** Scaling to larger `n` increases address width and QFT cost; error rates limit reliable recovery at bigger sizes. Embedding `Z_n` in `Z_{2^k}` introduces aliasing; mitigated by the zero-sample filter and multi-relation voting.

6. Reproduction (Windows CMD)

```
py qday_ibm_oracle_demo.py --mode shor --backend ibm_torino --curve-index 1 --moduli 32,7
 --horizon 1 --shots 2048 --print-params --save-report run_curve1_2048.json
py qday_ibm_oracle_demo.py --mode shor --backend ibm_torino --curve-index 0 --moduli 32,5
 --horizon 1 --shots 1024 --print-params --save-report run_curve0_1024.json
```