Contents

Iteration 2 Report - Leonid Lygin

Intro

- **Student name**: Leonid Lygin.
- **Topic**: "nice" EC param design.
- **Supervisor**: Phillip Braun.
- **Iteration number**: 2.

Initial plan

Initially, the topic was different, and the plan for it was to compare different implementations of brute-force functions.

Actual work done

The main point of this iteration was to implement an environment which can generate parameters, which `openssl` would understand, and also to create a very simple degenerate curve, as a small proof-of-concept for custom curve generation.

The actual degenerate curve that was implemented in this iteration, is based on the named `secp192k1` curve, and then slightly modified, to produce predictable results.

In the initial parameters the generator point G was some large number, with a large cyclic subgroup attached to it (cofactor $h = 1$), and to make the "attack", this generator was changed to a small point $G = (0, 2)$.

In order to make this actually work, the generator must be somewhere on the specified curve $y^2 = x^3 + ax + b$, and, substituting the coordinates for the generator point, we get $2^2 = 0^3 + a \cdot 0 + b$, for which the only working b is $b = 4$, and choice of a is irrelevant.

That set of parameters gives us some cyclic group, let's try to calculate it.

$$(0,2) + (0,2) = -(0,2)$$
$$(0,2) - (0,2) = \mathcal{O}$$
$$(0,2) + \mathcal{O} = (0,2)$$

As we can see, the generated group has order 2, which is unacceptable for a normal cryptographic environment, but for our case this is actually perfect.

We can try actually generating the keys to check that, and about third of the time, we actually get an error from `openssl`, because the ideal cannot be a public key. Other two choices we can get are $(0, 2)$, and $(0, p - 2)$, where p is the order of the prime field.

This can be tested using the `scripts/generate_keys.sh` script in the source repo.

References to results

All results can be found in the github repo: https://github.com/ionagamed/iu-bs3-project.

Current issues

The degenerate curve generated by the script can result in incorrect private keys, because private key for ECC is usually any non-zero bit string, and in some cases, the degenerate curve tries to divide by 0 when calculating the public key.

Plan for the next iteration

Refine the parameter generator, try some well-known attacks from wikipedia.