Experimental Results

Dmitar Zvonimir Mitev

1 General Results

The source code for the tests can be found in the `scheme.cpp` file. To perform the tests, remove the comments where it says "PERFORMING TESTS" and comment out the "EXAMPLE USAGE" part.

From the computed times we can deduce that:

1. The Setup and Encrypt functions require approximately the same amount of time for execution, provided that the safe prime is generated in advance. This is a consequence of their nature: they are similar in the sense that both perform many multiplications and exponentiations of big integers.

2. Decrypt requires the most resources out of all four functions. This is a consequence of the computation of the final discrete logarithm which takes $O(\sqrt{\ell \cdot B^2})$ time and space.

3. Keygen is a very fast function, even for large parameters ℓ and B. This is also expected, as it only needs to compute an inner product.

4. Increasing the size of bits of the key drastically increases the time of execution.

2 Size of the Parameters

We discuss the time needed in terms of the size of the parameters. The computed times can be found in the files "l=x.pdf", where 'x' is the size of ℓ. The following was concluded:

1. For $\ell \leq 100$ the scheme is useful even for moderately big $B \leq 1\,000\,000$.

2. For $\ell = 1000, B = 100\,000$ the scheme is still useful, as even for bits $= 4096$ the function Decrypt requires 35 seconds. By increasing B the scheme starts to have problems, as seen in "l=1000.pdf".

3. For $\ell = 10\,000$ we have to bound the number of bits to 2048, as otherwise the scheme takes too long to execute even Setup. When $B = 1\,000\,000$ the process was 'killed', as there was not enough RAM to execute Decrypt.

4. For $\ell = 100\,000$, we must bound B to 2048, as otherwise even Setup required 1499 seconds to execute.

function/avg. time(s)	Python impl.	C++ impl.
Setup	1.518	0.593
Encrypt	0.292	0.032
Keygen	0.00004	0.0001
Decrypt	33.880	0.258

Table 1: Comparison between the two implementations for parameters $\ell = 100, B = 10\,000, \text{bits} = 1024$

3 Final Thoughts

At the beginning of the semester, I stated that the new implementation must work much faster than my previous implementation[1] of the same scheme, which was developed as part of my bachelor's thesis. In Table 1 I present a direct comparison of the execution time required for all four algorithms of both schemes, using parameters $\ell = 100, B = 10\,000, \text{bits} = 1024$.

Note that in the Python implementation, Setup also generates a safe prime. According to the file "generating_safe_primes/times_for_safe_prime_generation.pdf" generating a 1024-bit safe prime requires, on average, 0.56 seconds. Therefore, for a fair comparison, we add 0.56 seconds to the execution time of the Setup function in the C++ implementation.

All four algorithms execute much faster in the new implementation, with Decrypt showing the most significant improvement – a factor of 131.

[1]The Python implementation can be found on https://github.com/dzmitev/InnerProductDDHScheme