Benchmarking class groups

Class groups, for which finding their order is considered hard under standard assumptions, are important building blocks to many cryptographic primitives including RSA accumulators, time lock puzzles, polynomial commitments, etc.
For evaluating our implementation of class group we attempted to compare to three other class group implementations written in rust (which are the only public libraries we found), where two of which, Cambrian's accumulator class group , and classy groups , failed to run and threw the (exact same) exception:

```
| ^^^^^^^^^^^ expected struct NonNull, found *-ptr
```

Hence, we benchmarked our class group and the VDF's class group using our machine (2.2 GHz Intel Core i7-4770HQ). Specifically we tested exponentiation in this group. Benchmarking VDF's class group gave odd results; on the one hand, for the very specific discriminant (that they chose), generated by a 2048 bit number, their implementation outperform ours (as can be seen in the table), on the other hand, for **any other** discriminant, the running time is extremely slow, even for very small exponents.
The following table presents the statistics of 10 samples of exponentiation in this group by comparing the running time of computing the term b^e for various sizes of the exponents e (e is sampled randomly, although its size is fixed) where the base b is a fixed element in the group, of size 2048 bits. The description of the benchmark output can be found in the Criterion documentation.

exponent size (bits)	**ZenGo-X's class group**	**VDF's class group**
1000	time: [496.88 ms **498.78 ms** 501.77 ms] mean [496.88 ms 501.77 ms] std. dev. [584.56 us 6.8414 ms]	time: [500.70 ms **502.58 ms** 504.62 ms] mean [500.70 ms 504.62 ms] std. dev. [7.2704 ms 12.602 ms]
2000	time: [1.0059 s **1.0116 s** 1.0210 s] mean [1.0059 s 1.0210 s] std. dev. [1.3614 ms 21.771 ms]	time: [1.0213 s **1.0292 s** 1.0381 s] mean [1.0213 s 1.0381 s] std. dev. [29.303 ms 55.157 ms]
3000	time: [2.1033 s **2.1141 s** 2.1256 s] mean [2.1033 s 2.1256 s] std. dev. [9.0197 ms 25.800 ms]	time: [1.5160 s **1.7578 s** 2.1290 s] mean [1.5160 s 2.1290 s] std. dev. [8.1550 ms 878.99 ms]

4000	time: [3.0620 s **3.1031 s** 3.1568 s] mean [3.0620 s 3.1568 s] std. dev. [17.054 ms 114.38 ms]	time: [2.2157 s **2.3876 s** 2.5741 s] mean [2.2157 s 2.5741 s] td. dev. [144.69 ms 359.00 ms]
5000	time: [4.2546 s **4.2694 s** 4.2851 s] mean [4.2546 s 4.2851 s] std. dev. [12.302 ms 33.993 ms]	time: [3.0478 s **3.0685 s** 3.0889 s] mean [3.0478 s 3.0889 s] std. dev. [20.192 ms 43.748 ms]
6000	time: [5.5715 s **5.5889 s** 5.6114 s] mean [5.5715 s 5.6114 s] std. dev. [10.508 ms 50.110 ms]	time: [3.0478 s **3.0685 s** 3.0889 s] mean [3.0478 s 3.0889 s] std. dev. [20.192 ms 43.748 ms]

The following table presents the benchmark results for various bases and various *fixed* exponents. The last three rows present the performance of exponentiation of a base different from the one chosen in the VDF's implementation. Since benchmarking VDF in this setting takes extremely long time, the corresponding results are omitted.

Base size (bits)	**exp size (bits)**	**ZenGo-X's class group**	**VDF's class group**
2048	**3000**	**time**: [581.75 ms **583.68 ms** 585.67 ms] mean [581.75 ms 585.67 ms] std. dev. [2.4093 ms 6.1236 ms]	time: [493.87 ms **496.61 ms** 499.59 ms] mean [493.87 ms 499.59 ms] std. dev. [11.590 ms 17.317 ms]
2048	**4500**	**time: [880.70 ms 884.56 ms 889.56 ms]** mean [880.70 ms 889.56 ms] std. dev. [3.6273 ms 15.778 ms]	**time: [743.41 ms 747.73 ms 753.65 ms]** mean [743.41 ms 753.65 ms] std. dev. [12.485 ms 41.665 ms]s]
2048	**6000**	**time: [1.1609 s 1.1743 s 1.1925 s]** mean [1.1609 s 1.1925 s] std. dev. [14.249 ms 57.719 ms]	**time: [994.98 ms 999.32 ms 1.0039 s]** mean [994.98 ms 1.0039 s] std. dev. [19.377 ms 26.191 ms]
3000	**3000**	**time**: [1.0271 s **1.0479 s** 1.0707 s] mean [1.0271 s 1.0707 s] std. dev. [34.438 ms 61.229 ms]	
4000	**3000**	**time**: [1.5157 s **1.5285 s** 1.5407 s] mean [1.5157 s 1.5407 s] std. dev. [22.234 ms 33.631 ms]	
5000	**3000**	**time**: [2.0682 s **2.0890 s** 2.1100 s] mean [2.0682 s 2.1100 s] std. dev. [37.118 ms 56.502 ms]	