Verificatum

Complexity Analysis
# of the Verificatum Mix-Net

VMN Version 3.1.0
2022-09-10

The Verificatum mix-net is an implementation of a provably secure El Gamal-based mix-net. This document provides several benchmarks as well as a heuristic complexity analysis based on these benchmarks. For additional information about the mix-net, please visit `http://www.verificatum.org`.

Contents

1 Introduction

It is important to understand the computational complexity of any protocol, in particular protocols used in practice. Unfortunately, mix-nets are quite complex which makes it difficult to perform a rigorous and precise theoretical complexity analysis. However, we have a good grasp of the general complexity of El Gamal-based mix-nets, and of the Verificatum mix-net (VMN) in particular. A comprehensive set of benchmarks can therefore be used to give a robust heuristic complexity analysis.

In this document we motivate why a complexity analysis is important, i.e., how it can help in applications, briefly mention some of the optimization techniques used, and provide concrete estimates of the running time in terms of the main parameters. These estimates are based on the benchmarks provided in Appendix A.

2 Motivation

When confronted with benchmarks for the running time of VMN people react positively, but some ask: Why is speed is so important? After all, many elections do not have many eligible voters within each voting district or even many eligible voters in total. There are several answers.

2.1 User Demand

The most obvious answer is that voting authorities and the public expect any computation performed by computers to finish quickly, since their experience is that this is possible in most other domains and the inability of doing this is deemed to be a sign of poor quality.

This is perhaps not a convincing argument when the input is small, since the running time is small even for a relatively naive implementation. However, if the input is large, then small constant factors matter. An optimized implementation handles both settings of course.

It is also important that external parties can implement efficient verifiers used by auditors to check that there was no tampering during the election, but in most settings the requirements on such verifiers are not as demanding since they are only executed after the completion of an election to ensure that the result is trusted to be correct.

2.2 Larger Security Parameters

History shows that the privacy of voters in sensitive elections should be protected for at least a life time. A breach of privacy may have catastrophic consequences for some individuals, but more importantly, many voters may abstain to vote if they do not trust that this is the case. Thus, the security parameters must be chosen conservatively to meet the security goals. This implies large security parameters which can have a major impact on the running time of an unoptimized implementation. We elaborate on this below and derive the data in this section from [2].

It is expected that a symmetric cryptosystem such as AES [1] remains secure against brute-force attacks for at least 30 year if the key size is at least 128 bits. To remain secure for the foreseeable future the key size must be increased to 256 bits. This assumes that there is no mathematical breakthrough in the cryptanalysis of ciphers.

We stress that if there is no such breakthrough, then the estimates are conservative and derived by groups of leading experts in cryptography, mathematics, and hardware design. Thus, it is safe to assume that more precise estimates can not be given by anybody today based on public information.

In particular, conflicting claims made by companies, voting authorities, or individual researchers should at best be considered fringe opinions, and at worst deliberate attempts to undermine the security of the system or market substandard software that is not sufficiently fast to be used with secure parameters.

The underlying cryptosystem of VMN is El Gamal which can be based on any prime order group for which messages can be encoded into and decoded from group elements, but the two most common and most trusted alternatives are: prime order subgroups of multiplicative groups modulo primes and prime order subgroups of carefully chosen elliptic curves. To determine a suitable security level for this cryptosystem we must translate the key sizes for ciphers above into the sizes of groups that give the same expected security for the El Gamal cryptosystem.

This is non-trivial to do in a convincing way and inevitably involves some hand-waiving, but conservative estimates have been derived by extrapolating the state of the art in hardware design and protocol analysis [2]. These estimates assume that it remains infeasible to construct a quantum computer with sufficiently many quantum bits for the foreseeable future.

Security	**Prime modulus**	**Subgroup**	**EC**
112	2432	224	224
128	3248	256	256
160	5312	320	320
192	7936	384	384
256	15424	512	512

Table 1: Excerpt from Table 7.2 in [2]. The first column shows the security level, which corresponds to performing a key recovery brute force attack on a cipher with the given number of bits in its key. The second is the bit size of the prime modulus for the discrete logarithm problem for the corresponding security and the third is the logarithm of the size of the smallest subgroup that can be used without decreasing the hardness of the discrete logarithm problem. The fourth column is the logarithm of the order of a suitably chosen elliptic curve with the corresponding security level. This is typically also the order of the underlying field.

Unfortunately, there is no efficient way to embed more than a few bits in an invertible and efficient way into a small subgroup of the multiplicative group modulo a prime. Thus, such groups are rarely used with the El Gamal cryptosystem and are cumbersome to use in most voting applications. In other words, to use a multiplicative group modulo a prime we need to make the subgroup actually used almost as large as the complete multiplicative group.

We stress that all of the above assumes that the proof of security of the mix-net is tight, since otherwise we need even larger security parameters.

We conclude that the consensus in the cryptographic community is that multiplicative groups need at least 3248 bits and that the logarithm of the size of elliptic curves used must be at least 256 bits.

2.3 Secure Design and Implementation of Electronic Voting Systems

A more technical answer is that, combined with the flexibility provided by VMN, many situations can be handled in a uniform way that would otherwise require the design and implementation of custom-made protocols

Recall that in an electronic voting system the mix-net is used to tally. Although the encryption of the votes submitted by voters is merely a front-end to the mix-net, it matters how this encryption is performed. Depending on the type of election it may be non-trivial to design such a scheme and provide protocols used by clients to prove that this was done correctly. In the worst case a small mistake means that the result is not secure.

Furthermore, the use of tailor-made clients implies that the output must be processed in an additional step, which must also be verified during auditing. For some proposed submission schemes this involves performing operations in the underlying group, so this is not merely a matter of formatting strings, e.g., in some schemes this involve taking discrete logarithms.

With a sufficiently fast mix-net that can handle multiple keys and ciphertexts a much more relaxed approach can be taken in the construction of a full electronic voting system. Clients can embed a vote into multiple ciphertexts processed as one in the mix-net and use more straightforward proofs that this was done correctly. The use of more standard proofs reduces the risk that there is security flaw on the client side.

Example 1. In a ranked election with 10 candidates the candidates can be encrypted in separate ciphertexts and the resulting tuple processed as if it was a single ciphertext. A common misunderstanding is that this increases the running time by a factor of 10, but properly implemented as in VMN, this only increases the running time by a factor of 3 (and this decreases if the width increases). For example, if $1,000,000$ simple ciphertexts can be processed in reasonable time, then at least $300,000$ votes with 10 ranked candidates can be handled.

Example 2. Consider an election with multiple relatively small voting districts. Instead of processing each district separately, multiple districts can be pooled and processed together and the final result sorted with respect to the district. This is of course possible in any scheme, but only at the expense of introducing additional complexity in the proof that a ciphertext is well-formed. The client must also behave differently depending on to which voting district the voter belongs. In VMN the mix-nets can simply append an encryption of the voting district, formed using fixed and known randomness, to each ciphertext submitted by a voter. Then the resulting pairs of ciphertexts are processed. The result is a list of plaintexts consisting of an index of a voting district and a vote within that index. Sorting with respect to the voting district is now trivial.

2.4 Economy of Development and Deployment

A scheme that allows simpler clients drastically reduces the time needed for developing and auditing such clients. This means that the solution can be deployed more quickly, but perhaps more importantly that the costs for development are reduced. Indeed, in contrast to many other types of software, the client must be free of bugs to remain secure, and developing such code is expensive.

A smaller running time also allows using slower servers. This is of course important in poor countries. However, it may also be important in other countries where the total cost of maintaining a server in a secure location mainly consists of the work involved and not buying the server hardware: with a fast mix-net there is likely no need to choose, acquire, and install new servers.

3 Optimization Techniques

The expensive operations throughout the execution are exponentiations in the underlying group. Most papers in the literature simply count the exponentiations and derive estimates from this. Sometimes these values are used to compare the running times of different protocols. In particular proofs of shuffles. Unfortunately, such approximations are far from precise and may result in wrong conclusions about the relative speeds of the corresponding optimized implementations. Below we briefly describe some optimization techniques that are used in VMN.

3.1 Exponentiation Algorithms

Not all exponentiations are equally costly. There are techniques to compute multiple fixed-base exponentiations efficiently, i.e., to compute g^{x_i} for $i = 1, \ldots, k$ given $x_1, \ldots, x_k$. When the number k of exponentiations to be computed is large, the cost of each exponentiations is drastically reduced. For medium-sized k, other techniques allow computing multi-exponentiations such as $\prod_{i=1}^{k} g_i^{x_i}$ given $g_1, \ldots, g_k$ and $x_1, \ldots, x_k$ much faster than simply computing $g_i^{x_i}$ for $i = 1, \ldots, k$ and then taking the product. These methods are described in, e.g., [4].

When the number k of basis elements is large additional techniques can be applied in the case of multi-exponentiations that reduces the running time almost by an additional factor of $1/2$.

These techniques may be applied naively to a protocol from the literature. However, a careful analysis typically reveals that expressions can be written in different ways and the techniques mentioned above can be applied to different extents for these.

3.2 Threading

Today computers have multiple cores. Some operations lend themselves well to threading whereas others do not. Again, in some cases expressing things in a different way makes it possible to thread operations that are seemingly not possible to thread.

3.3 Parallelism

The complexity estimates found in the literature typically assumes that the parties perform their computations in the order they are communicated in a protocols. Careful planning of the software allows computing many values in parallel. Optimizing this is non-trivial, e.g., it is sometimes possible to rewrite expressions that are more costly to compute on their own, but can be parallelized to a greater extent which gives a faster execution overall.

4 Costs of Optimization

Almost any piece of software can be optimized, but the cost to do so in terms of increased development time and code complexity is not necessarily worth the effort. Knuth [3] summarizes this wisely.

> Programmers waste enormous amounts of time thinking about, or worrying about, the speed of noncritical parts of their programs, and these attempts at efficiency actually have a strong negative impact when debugging and maintenance are considered. We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil. Yet we should not pass up our opportunities in that critical 3%.

This applies not only to the code itself, but also to the abstract protocols implemented. In particular, the simplicity of the so-called “proofs of shuffles” used to prove that the execution is correct is an important parameter to consider when deploying a protocol.

There is a subtle interplay between: the choices of protocols, security proofs and resulting security parameters, rewriting for exponentiations techniques, threading, and parallelism, and code complexity. Finding a good tradeoff is difficult and requires a deep understanding of cryptography, arithmetic algorithms, and software engineering.

Broadly speaking the approach taken in VMN is to optimize aggressively at a low level only when it matters, and strive to keep the application code as clean and simple as possible, even in cases where more complex application code could give minor speed improvements.

5 Complexity Estimates

Due to the flexibility of VMN it is infeasible to provide an explicit formula that includes all of its parameters. However, it is possible to give explicit benchmarks for natural parameters and we can derive formulas that relate the main parameters. It is not hard to see that if an El Gamal based mix-net is implemented correctly, then its running time should be determined by the running times of:

1. A distributed key generation protocol.
2. A distributed protocol for shuffling a list of ciphertexts, including mutual proofs of correctness and verification of these.
3. A distributed protocol for decrypting a list of ciphertexts, including mutual proofs of correctness and verification of these.
4. A program for verifying a proof of correct shuffling consisting of components provided by the parties in a distributed computation.
5. A program for verifying a proof of correct decryption consisting of components provided by the parties in a distributed computation.

The distributed key generation protocol is not optimized aggressively, since it is not expected to be executed frequently. It is simply fast enough and takes a couple of seconds. The running times of the shuffling phase and the decryption phase add up to roughly the running time of the execution of a complete mixing session as expected.

The remaining estimates depend mainly on the underlying group, security parameters, the number of ciphertexts N, the number k of parties, the threshold number λ of parties needed for decryption, the key width κ, and the width ω of ciphertexts.

If all parameters are fixed except the number N of ciphertexts, then the running time of shuffling, decryption, and hence mixing, can be expressed as $aN + b$, where b is small. This is what is expected.

If all parameters are fixed except the threshold number λ of parties, then the running time of shuffling may be expressed as $a\lambda + b$, where b is small. The situation for decryption is quite different. In this case the running time can be described as as $a\lambda + b$, where a is nearly zero. Although the former estimate is expected, the second may come as a surprise. The explanation is that the computations needed for distributed decryption are performed in parallel and the proof of correct decryption is parallelized.

If all parameters are fixed except the key width κ and the width ω of ciphertexts, then the running time can be expressed as $a\kappa\omega + b$. The bilinear relation is natural due to how ciphertexts are represented. It is worth noticing that b is relatively large compared to a, e.g., increasing the width of ciphertexts from one to two only increases the running time slightly. The reason is that the most costly part of the proof of a shuffle used does not involve the ciphertexts at all.

The claims above are supported by the benchmarks in Appendix A. These benchmarks are interesting in their own right, but combining them with the insight into the implementation also gives a fair idea of how the performance of VMN depends on all the main parameters together.

Another important parameter to study is the amount of communication and the sizes of proofs of correctness. We do not derive explicit expressions in the benchmarks below, since all values are relatively small, but note that they scale similarly with the exception of decryption for which the communication and size of proofs grow linearly with the number of parties.

The soundness of the proofs of correctness is determined by a separate security parameter that is constant in all benchmarks, but this has been chosen very conservatively to 256 bits which is typically considered to give, loosely speaking, at least 128-bit security. This is a natural choice, since soundness on the one hand is perhaps the most important property, but on the other hand does not need to hold for as long as privacy.

The parameters of the signature scheme and cryptosystem used to implement secure authenticated broadcast have a negligible impact on the running times, but they have been chosen very conservatively at a 256-bit security level. The latter is strictly speaking necessary for the benchmarks for the 256-bit security parameters to be valid.

References

[1] J. Daemen and V. Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer Verlag, 2002.

[2] European Network of Excellence in Cryptology II. Ecrypt2 yearly report on algorithms and keysizes (2011-2012). September 2012.

[3] D. E. Knuth. Structured programming with go to statements. *ACM Comput. Surv.*, 6(4):261–301, 1974.

[4] A. Menezes, P. Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

A Benchmarks

The following benchmarks do not take network time into consideration. The problem with communication in benchmarks is that without dedicated channels between the parties, it is typically asymmetric in nature and introduces too much variance in measurements. Note however that the amount of communicated data is specified and that the benchmarks are computed by a simple script that is easy to run for any concrete setting.

We have been unable to run the benchmarks on dedicated machines with identical hardware for the full duration of the benchmarks, and have only had access to three machines that were chosen to be as similar as possible. This explains relative errors for our estimates of a few percent in some benchmarks. The machines have been run remotely at different physical locations over the Internet and not in a lab environment.

A.1 NIST Standard Elliptic Curve P-256

```
Party01:
   host: dog@178.78.255.115
    CPU: Intel(R) Core(TM) i7-4790K CPU (X4 x2) (4.00GHz)
     OS: GNU/Linux 3.13.0-68-generic (Ubuntu 14.04.3 LTS)
Party02:
   host: dog@130.237.224.123
    CPU: AMD Phenom(tm) II X6 1100T Processor (X6) (3.31GHz)
     OS: GNU/Linux 3.13.0-68-generic (Ubuntu 14.04.3 LTS)
Party03:
   host: dog@130.237.224.227
    CPU: AMD Phenom(tm) II X4 955 Processor (X4) (3.20GHz)
     OS: FreeBSD 10.1-RELEASE
Group is:
ECqPGroup(P-256)
```

```
======================================================================

Key generation, 2.84 seconds.

======================================================================

Shows that the running times of shuffling and decryption adds to
approximately the running time of mixing. Furthermore, the
communication of shuffling and decryption adds approximately to the
communication of mixing. Similarly, the sizes of the proofs of
shuffling and decryption adds approximately to the size of the proof
of mixing.

          Secs:   Operation (1000000 ciphertexts):
        1072.49   Shuffling
         136.03   Decryption
        1208.52   Shuffling + Decryption
        1198.18   Mixing

         460.28   Verifying shuffling
          84.54   Verifying decryption
         544.82   Verifying shuffling + Verifying decryption
         543.26   Verifying mixing

 Communication:   Operation (1000000 ciphertexts):
         1.4 GB   Shuffling
       324.0 MB   Decryption
         1.8 GB   Shuffling + Decryption
         1.8 GB   Mixing

    Proof size:   Operation (1000000 ciphertexts):
         1.3 GB   Shuffling
       486.0 MB   Decryption
         1.8 GB   Shuffling + Decryption
         1.4 GB   Mixing

======================================================================

Shows that the running times E(N) and V(N) for mixing and for
verifying mixing, respectively, grow linearly with the number N
of ciphertexts, and that the number of bytes of data communicated
C(N), and the the sizes of Fiat-Shamir proofs P(N) grow similarly.

Approximations of the functions E(N), V(N), C(N), and P(N):

    e(N) = 1.198 * N  +  9176.250 (ms)
    v(N) = 0.539 * N  +  4188.100 (ms)

    c(N) = 1767.000 * N  +  27782.000 (bytes)
    p(N) = 1448.000 * N  +  2237.000 (bytes)


MIXING       | Time (sec)             | Time/ciphertext (ms)
Ciphertexts: | Executing:  Verifying: | Executing:  Verifying:
-------------+------------------------+----------------------
```

```
     200,000 |     247.38      111.81 |          1           1
     400,000 |     490.68      219.41 |          1           1
     600,000 |     731.38      327.81 |          1           1
     800,000 |     965.11      435.95 |          1           1
   1,000,000 |    1205.94      543.02 |          1           1
```

Recall that E(N) and V(N) denote the execution time and verification time respectively in milliseconds on N ciphertexts. We bound the error of our approximations compared to the benchmark by:

```
    |e(N) - E(N)| / E(N) <= 0.01
    |v(N) - V(N)| / V(N) <= 0.00
```

```
MIXING       | Data
Ciphertexts: | Communicated:         Proof:
-------------+----------------------------
     200,000 |      353.4 MB       289.6 MB
     400,000 |      706.8 MB       579.2 MB
     600,000 |        1.1 GB       868.8 MB
     800,000 |        1.4 GB         1.2 GB
   1,000,000 |        1.8 GB         1.4 GB
```

Recall that C(N) denotes the number of bytes communicated, and P(N) denotes the size in bytes of the Fiat-Shamir proof of correctness. We bound the error of our approximations compared to the benchmark by:

```
    |c(N) - C(N)| / C(N) <= 0.00
    |p(N) - P(N)| / P(N) <= 0.00
```

```
======================================================================
```

Shows that the running times E(N) and V(N) for shuffling and for verifying shuffling, respectively, grow linearly with the number N of ciphertexts, and that the number of bytes of data communicated C(N), and the the sizes of Fiat-Shamir proofs P(N) grow similarly.

Approximations of the functions E(N), V(N), C(N), and P(N):

```
    e(N) = 1.060 * N  +  5413.950 (ms)
    v(N) = 0.458 * N  +  3354.950 (ms)

    c(N) = 1443.000 * N  +  17546.000 (bytes)
    p(N) = 1286.000 * N  +  1611.000 (bytes)
```

```
SHUFFLING    | Time (sec)             | Time/ciphertext (ms)
Ciphertexts: | Executing:  Verifying: | Executing:  Verifying:
-------------+------------------------+-----------------------
     200,000 |     217.50       94.73 |          1           0
     400,000 |     429.09      186.02 |          1           0
     600,000 |     644.49      278.53 |          1           0
     800,000 |     850.98      369.35 |          1           0
   1,000,000 |    1065.67      460.77 |          1           0
```

Recall that E(N) and V(N) denote the execution time and verification

```
time respectively in milliseconds on N ciphertexts. We bound the error
of our approximations compared to the benchmark by:

    |e(N) - E(N)| / E(N) <= 0.00
    |v(N) - V(N)| / V(N) <= 0.00


SHUFFLING    | Data
Ciphertexts: | Communicated:         Proof:
-------------+----------------------------
     200,000 |      288.6 MB       257.2 MB
     400,000 |      577.2 MB       514.4 MB
     600,000 |      865.8 MB       771.6 MB
     800,000 |        1.2 GB         1.0 GB
   1,000,000 |        1.4 GB         1.3 GB

Recall that C(N) denotes the number of bytes communicated, and P(N)
denotes the size in bytes of the Fiat-Shamir proof of correctness. We
bound the error of our approximations compared to the benchmark by:

    |c(N) - C(N)| / C(N) <= 0.00
    |p(N) - P(N)| / P(N) <= 0.00

======================================================================

Shows that the running times E(N) and V(N) for decryption and for
verifying decryption, respectively, grow linearly with the number N
of ciphertexts, and that the number of bytes of data communicated
C(N), and the the sizes of Fiat-Shamir proofs P(N) grow similarly.

Approximations of the functions E(N), V(N), C(N), and P(N):

    e(N) = 0.131 * N  +  3934.050 (ms)
    v(N) = 0.083 * N  +  1492.750 (ms)

    c(N) = 324.000 * N  +  17594.000 (bytes)
    p(N) = 486.000 * N  +  1017.000 (bytes)


DECRYPTION   | Time (sec)                | Time/ciphertext (ms)
Ciphertexts: | Executing:  Verifying:  | Executing:  Verifying:
-------------+-------------------------+-----------------------
     200,000 |      29.92       18.33  |          0           0
     400,000 |      56.84       34.51  |          0           0
     600,000 |      82.14       51.24  |          0           0
     800,000 |     110.00       67.62  |          0           0
   1,000,000 |     135.06       84.72  |          0           0

Recall that E(N) and V(N) denote the execution time and verification
time respectively in milliseconds on N ciphertexts. We bound the error
of our approximations compared to the benchmark by:

    |e(N) - E(N)| / E(N) <= 0.01
    |v(N) - V(N)| / V(N) <= 0.01
```

```
DECRYPTION   | Data
Ciphertexts: | Communicated:         Proof:
-------------+----------------------------
     200,000 |        64.8 MB        97.2 MB
     400,000 |       129.6 MB       194.4 MB
     600,000 |       194.4 MB       291.6 MB
     800,000 |       259.2 MB       388.8 MB
   1,000,000 |       324.0 MB       486.0 MB

Recall that C(N) denotes the number of bytes communicated, and P(N)
denotes the size in bytes of the Fiat-Shamir proof of correctness. We
bound the error of our approximations compared to the benchmark by:

    |c(N) - C(N)| / C(N) <= 0.00
    |p(N) - P(N)| / P(N) <= 0.00

======================================================================

Shows how the running times E(lambda) and V(lambda) for shuffling
and for verifying shuffling, respectively, grow with the threshold
number lambda of parties, and similarly for the number of bytes of
data communicated C(lambda), and the the size of Fiat-Shamir proofs
P(lambda).

Approximations of the functions E(lambda), V(lambda), C(lambda),
and P(lambda):

    e(lambda) = 766985.000 * lambda  +  -443177.333 (ms)
    v(lambda) = 219928.000 * lambda  +  21547.667 (ms)

    c(lambda) = 481004788.000 * lambda  +  481007970.000 (bytes)
    p(lambda) = 481000693.000 * lambda  +  324000225.000 (bytes)


SHUFFLING    | Time (sec)              | Data
Parties:     | Executing:  Verifying:  | Communication:      Proof:
-------------+-------------------------+---------------------------
           1 |     336.14      241.67  |        962.0 MB    805.0 MB
           2 |    1066.13      461.02  |          1.4 GB      1.3 GB
           3 |    1870.11      681.52  |          1.9 GB      1.8 GB

Recall that E(lambda) and V(lambda) denote the execution time and
verification time respectively in milliseconds on lambda
ciphertexts. We bound the error of our approximations compared to the
benchmark by:

    |e(lambda) - E(lambda)| / E(lambda) <= 0.04
    |v(lambda) - V(lambda)| / V(lambda) <= 0.00

Furthermore, C(lambda) denotes the number of bytes communicated, and
P(lambda) denotes the size in bytes of the proof of correctness. We
bound the error of our approximations compared to the benchmark by:

    |c(lambda) - C(lambda)| / C(lambda) <= 0.00
    |p(lambda) - P(lambda)| / P(lambda) <= 0.00
```

The fact that the servers run on different hardware explains why the approximation of the execution time is not precise, i.e., when the threshold is increased machines with different hardwares are activated.

```
=================================================================
```

Shows how the running times E(lambda) and V(lambda) for decryption and for verifying decryption, respectively, grow with the threshold number lambda of parties, and similarly for the number of bytes of data communicated C(lambda), and the the size of Fiat-Shamir proofs P(lambda).

Approximations of the functions E(lambda), V(lambda), C(lambda), and P(lambda):

```
    e(lambda) = 15165.000 * lambda  +  105249.667 (ms)
    v(lambda) = 14294.500 * lambda  +  54122.667 (ms)

    c(lambda) = 0.000 * lambda  +  324017594.000 (bytes)
    p(lambda) = 81.000 * lambda  +  486000855.000 (bytes)


DECRYPTION  | Time (sec)             | Data
Parties:    | Executing:  Verifying: | Communication:      Proof:
------------+------------------------+--------------------------
          1 |     119.51       67.67 |       324.0 MB    486.0 MB
          2 |     137.40       84.21 |       324.0 MB    486.0 MB
          3 |     149.84       96.26 |       324.0 MB    486.0 MB
```

Recall that E(lambda) and V(lambda) denote the execution time and verification time respectively in milliseconds on lambda ciphertexts. We bound the error of our approximations compared to the benchmark by:

```
    |e(lambda) - E(lambda)| / E(lambda) <= 0.01
    |v(lambda) - V(lambda)| / V(lambda) <= 0.02
```

Furthermore, C(lambda) denotes the number of bytes communicated, and P(lambda) denotes the size in bytes of the proof of correctness. We bound the error of our approximations compared to the benchmark by:

```
    |c(lambda) - C(lambda)| / C(lambda) <= 0.00
    |p(lambda) - P(lambda)| / P(lambda) <= 0.00

=================================================================
```

Shows that the running times E(kappa,omega) and V(kappa,omega) for shuffling of 20000 ciphertexts grow linearly both with
the key width kappa and the width omega of ciphertexts, and that bytes of data communicated C(kappa,omega) and Fiat-Shamir proof size P(kappa,omega) grow linearly with both kappa and width.

Approximations of the functions E(kappa,omega), V(kappa,omega), C(kappa,omega), and P(kappa,omega):

```
if (kappa = 1 or omega = 1) {

    e(kappa,omega) = 6825.000 * kappa * omega  +  19241.200 (ms)
    v(kappa,omega) = 2564.500 * kappa * omega  +  8102.500 (ms)

} else {

    e(kappa,omega) = 6950.625 * kappa * omega  +  18487.250 (ms)
    v(kappa,omega) = 2579.000 * kappa * omega  +  8036.600 (ms)
}
```

The need for two bilinear maps is due to how group elements are represented. The two cases kappa = 1 or omega = 1 are handled with trees of depth 1, whereas kappa,omega > 1 requires a tree of depth 2.

```
    c(kappa,omega) = 9720.7 * kappa * omega  +  19156.920 (KB)
    p(kappa,omega) = 12960.5 * kappa * omega  +  12761.519 (KB)
```

```
Execution (ms)
    |          1           2           3           4           5
----+-----------------------------------------------------------
  1 |      26.08       33.44       39.29       46.38       53.38
  2 |      32.22       46.38       60.22       74.30       87.83
  3 |      39.54       60.41       81.09      103.14      123.86
  4 |      46.80       74.31      103.49      132.33      159.96
  5 |      53.69       88.45      126.44      160.13      195.46

Verification (ms)
    |          1           2           3           4           5
----+-----------------------------------------------------------
  1 |      10.69       13.19       15.80       18.35       20.95
  2 |      13.20       18.33       23.55       28.64       33.84
  3 |      15.82       23.54       31.18       38.93       46.51
  4 |      18.40       28.68       38.98       49.13       59.37
  5 |      20.92       33.61       46.53       59.44       72.34
```

Recall that E(kappa,omega) and V(kappa,omega) denote the actual execution time and verification time respectively in milliseconds on 20000 ciphertexts. We bound the error of our approximations compared to the benchmark by:

```
    |e(kappa,omega) - E(kappa,omega)| / E(kappa,omega) <= 0.03
    |v(kappa,omega) - V(kappa,omega)| / V(kappa,omega) <= 0.01
```

```
Communication
    |          1           2           3           4           5
----+-----------------------------------------------------------
  1 |    28.9 MB     38.6 MB     48.3 MB     58.0 MB     67.8 MB
  2 |    38.6 MB     58.0 MB     77.5 MB     96.9 MB    116.4 MB
  3 |    48.3 MB     77.5 MB    106.6 MB    135.8 MB    165.0 MB
  4 |    58.0 MB     96.9 MB    135.8 MB    174.7 MB    213.6 MB
  5 |    67.8 MB    116.4 MB    165.0 MB    213.6 MB    262.2 MB

Proof size
    |          1           2           3           4           5
```

```
----+--------------------------------------------------
  1 |    25.7 MB    38.7 MB    51.6 MB    64.6 MB    77.6 MB
  2 |    38.7 MB    64.6 MB    90.5 MB   116.4 MB   142.4 MB
  3 |    51.6 MB    90.5 MB   129.4 MB   168.3 MB   207.2 MB
  4 |    64.6 MB   116.4 MB   168.3 MB   220.1 MB   272.0 MB
  5 |    77.6 MB   142.4 MB   207.2 MB   272.0 MB   336.8 MB

Recall that C(kappa,omega) denotes the number of bytes communicated,
and P(kappa,omega) denotes the size in bytes of the Fiat-Shamir proof
of correctness, on 20000 ciphertexts. We bound the error
of our approximations compared to the benchmark by:

    |c(kappa,omega) - C(kappa,omega)| / C(kappa,omega) <= 0.00
    |p(kappa,omega) - P(kappa,omega)| / P(kappa,omega) <= 0.00

===================================================================

Shows that the running times E(kappa,omega) and V(kappa,omega) for
decryption of 20000 ciphertexts grow linearly both with
the key width kappa and the width omega of ciphertexts, and that bytes
of data communicated C(kappa,omega) and Fiat-Shamir proof size
P(kappa,omega) grow linearly with both kappa and width.

Approximations of the functions E(kappa,omega), V(kappa,omega),
C(kappa,omega), and P(kappa,omega):

if (kappa = 1 or omega = 1) {

    e(kappa,omega) = 2978.250 * kappa * omega  +  2856.650 (ms)
    v(kappa,omega) = 1979.500 * kappa * omega  +  585.300 (ms)

} else {

    e(kappa,omega) = 2973.625 * kappa * omega  +  2820.650 (ms)
    v(kappa,omega) = 1895.875 * kappa * omega  +  806.350 (ms)
}
The need for two bilinear maps is due to how group elements are
represented. The two cases kappa = 1 or omega = 1 are handled with
trees of depth 1, whereas kappa,omega > 1 requires a tree of depth 2.

    c(kappa,omega) = 6480.4 * kappa * omega  +  17.768 (KB)
    p(kappa,omega) = 9720.3 * kappa * omega  +  1.477 (KB)


Execution (ms)
    |          1          2          3          4          5
----+--------------------------------------------------
  1 |       5.84       8.78      11.87      14.71      17.75
  2 |       8.77      14.53      20.43      27.02      32.56
  3 |      11.81      20.97      29.83      38.52      47.83
  4 |      15.79      26.70      39.23      51.22      62.66
  5 |      17.39      32.80      47.92      62.66      78.08

Verification (ms)
    |          1          2          3          4          5
----+--------------------------------------------------
```

```
  1 |       2.58        4.62        6.49        8.44       10.50
  2 |       4.58        8.40       12.20       15.97       19.75
  3 |       6.64       12.27       17.83       23.60       29.33
  4 |       8.37       16.04       23.78       31.22       39.02
  5 |      10.48       19.74       29.27       38.86       48.24

Recall that E(kappa,omega) and V(kappa,omega) denote the actual
execution time and verification time respectively in milliseconds on
20000 ciphertexts. We bound the error of our
approximations compared to the benchmark by:

    |e(kappa,omega) - E(kappa,omega)| / E(kappa,omega) <= 0.06
    |v(kappa,omega) - V(kappa,omega)| / V(kappa,omega) <= 0.02


Communication
    |          1           2           3           4           5
----+------------------------------------------------------------
  1 |     6.5 MB     13.0 MB     19.5 MB     25.9 MB     32.4 MB
  2 |    13.0 MB     25.9 MB     38.9 MB     51.9 MB     64.8 MB
  3 |    19.5 MB     38.9 MB     58.3 MB     77.8 MB     97.2 MB
  4 |    25.9 MB     51.9 MB     77.8 MB    103.7 MB    129.6 MB
  5 |    32.4 MB     64.8 MB     97.2 MB    129.6 MB    162.0 MB

Proof size
    |          1           2           3           4           5
----+------------------------------------------------------------
  1 |     9.7 MB     19.4 MB     29.2 MB     38.9 MB     48.6 MB
  2 |    19.4 MB     38.9 MB     58.3 MB     77.8 MB     97.2 MB
  3 |    29.2 MB     58.3 MB     87.5 MB    116.6 MB    145.8 MB
  4 |    38.9 MB     77.8 MB    116.6 MB    155.5 MB    194.4 MB
  5 |    48.6 MB     97.2 MB    145.8 MB    194.4 MB    243.0 MB

Recall that C(kappa,omega) denotes the number of bytes communicated,
and P(kappa,omega) denotes the size in bytes of the Fiat-Shamir proof
of correctness, on 20000 ciphertexts. We bound the error
of our approximations compared to the benchmark by:

    |c(kappa,omega) - C(kappa,omega)| / C(kappa,omega) <= 0.00
    |p(kappa,omega) - P(kappa,omega)| / P(kappa,omega) <= 0.00
```

A.2 NIST Standard Elliptic Curve P-521

```
Party01:
   host: dog@178.78.255.115
    CPU: Intel(R) Core(TM) i7-4790K CPU (X4 x2) (4.00GHz)
     OS: GNU/Linux 3.13.0-71-generic (Ubuntu 14.04.3 LTS)
Party02:
   host: dog@130.237.224.123
    CPU: AMD Phenom(tm) II X6 1100T Processor (X6) (3.31GHz)
     OS: GNU/Linux 3.13.0-71-generic (Ubuntu 14.04.3 LTS)
Party03:
   host: dog@130.237.224.227
    CPU: AMD Phenom(tm) II X4 955 Processor (X4) (3.20GHz)
```

```
     OS: FreeBSD 10.1-RELEASE
Group is:
ECqPGroup(P-521)

======================================================================

Key generation, 2.85 seconds.

======================================================================

Shows that the running times of shuffling and decryption adds to
approximately the running time of mixing. Furthermore, the
communication of shuffling and decryption adds approximately to the
communication of mixing. Similarly, the sizes of the proofs of
shuffling and decryption adds approximately to the size of the proof
of mixing.

          Secs:   Operation (500000 ciphertexts):
        1209.41   Shuffling
         161.94   Decryption
        1371.35   Shuffling + Decryption
        1366.16   Mixing

         552.10   Verifying shuffling
          91.81   Verifying decryption
         643.91   Verifying shuffling + Verifying decryption
         638.90   Verifying mixing

 Communication:   Operation (500000 ciphertexts):
         1.3 GB   Shuffling
       294.0 MB   Decryption
         1.6 GB   Shuffling + Decryption
         1.6 GB   Mixing

    Proof size:   Operation (500000 ciphertexts):
         1.2 GB   Shuffling
       441.0 MB   Decryption
         1.6 GB   Shuffling + Decryption
         1.3 GB   Mixing

======================================================================

Shows that the running times E(N) and V(N) for mixing and for
verifying mixing, respectively, grow linearly with the number N
of ciphertexts, and that the number of bytes of data communicated
C(N), and the the sizes of Fiat-Shamir proofs P(N) grow similarly.

Approximations of the functions E(N), V(N), C(N), and P(N):

    e(N) = 2.717 * N  +  15819.550 (ms)
    v(N) = 1.258 * N  +  9907.900 (ms)

    c(N) = 3219.000 * N  +  29828.000 (bytes)
    p(N) = 2636.000 * N  +  3920.000 (bytes)
```

```
MIXING       | Time (sec)              | Time/ciphertext (ms)
Ciphertexts: | Executing:  Verifying:  | Executing:  Verifying:
-------------+-------------------------+-----------------------
     100,000 |     287.44      136.05  |          3           1
     200,000 |     559.38      261.87  |          3           1
     300,000 |     833.46      387.26  |          3           1
     400,000 |    1099.93      512.00  |          3           1
     500,000 |    1374.19      639.20  |          3           1

Recall that E(N) and V(N) denote the execution time and verification
time respectively in milliseconds on N ciphertexts. We bound the error
of our approximations compared to the benchmark by:

    |e(N) - E(N)| / E(N) <= 0.00
    |v(N) - V(N)| / V(N) <= 0.00


MIXING       | Data
Ciphertexts: | Communicated:         Proof:
-------------+-----------------------------
     100,000 |      321.9 MB       263.6 MB
     200,000 |      643.8 MB       527.2 MB
     300,000 |      965.7 MB       790.8 MB
     400,000 |        1.3 GB         1.1 GB
     500,000 |        1.6 GB         1.3 GB

Recall that C(N) denotes the number of bytes communicated, and P(N)
denotes the size in bytes of the Fiat-Shamir proof of correctness. We
bound the error of our approximations compared to the benchmark by:

    |c(N) - C(N)| / C(N) <= 0.00
    |p(N) - P(N)| / P(N) <= 0.00

======================================================================

Shows that the running times E(N) and V(N) for shuffling and for
verifying shuffling, respectively, grow linearly with the number N
of ciphertexts, and that the number of bytes of data communicated
C(N), and the the sizes of Fiat-Shamir proofs P(N) grow similarly.

Approximations of the functions E(N), V(N), C(N), and P(N):

    e(N) = 2.374 * N  +  17364.950 (ms)
    v(N) = 1.095 * N  +  6484.850 (ms)

    c(N) = 2631.000 * N  +  18932.000 (bytes)
    p(N) = 2342.000 * N  +  2799.000 (bytes)


SHUFFLING    | Time (sec)              | Time/ciphertext (ms)
Ciphertexts: | Executing:  Verifying:  | Executing:  Verifying:
-------------+-------------------------+-----------------------
     100,000 |     253.63      116.48  |          3           1
     200,000 |     493.49      225.62  |          2           1
     300,000 |     733.20      335.14  |          2           1
     400,000 |     964.17      443.07  |          2           1
     500,000 |    1203.19      554.43 |          2           1

Recall that E(N) and V(N) denote the execution time and verification
time respectively in milliseconds on N ciphertexts. We bound the error
of our approximations compared to the benchmark by:

    |e(N) - E(N)| / E(N) <= 0.00
    |v(N) - V(N)| / V(N) <= 0.00


SHUFFLING    | Data
Ciphertexts: | Communicated:         Proof:
-------------+-----------------------------
     100,000 |      263.1 MB       234.2 MB
     200,000 |      526.2 MB       468.4 MB
     300,000 |      789.3 MB       702.6 MB
     400,000 |        1.1 GB       936.8 MB
     500,000 |        1.3 GB         1.2 GB

Recall that C(N) denotes the number of bytes communicated, and P(N)
denotes the size in bytes of the Fiat-Shamir proof of correctness. We
bound the error of our approximations compared to the benchmark by:

    |c(N) - C(N)| / C(N) <= 0.00
    |p(N) - P(N)| / P(N) <= 0.00

======================================================================

Shows that the running times E(N) and V(N) for decryption and for
verifying decryption, respectively, grow linearly with the number N
of ciphertexts, and that the number of bytes of data communicated
C(N), and the the sizes of Fiat-Shamir proofs P(N) grow similarly.

Approximations of the functions E(N), V(N), C(N), and P(N):

    e(N) = 0.311 * N  +  4868.950 (ms)
    v(N) = 0.179 * N  +  2336.750 (ms)

    c(N) = 588.000 * N  +  18254.000 (bytes)
    p(N) = 882.000 * N  +  1776.000 (bytes)


DECRYPTION   | Time (sec)              | Time/ciphertext (ms)
Ciphertexts: | Executing:  Verifying: | Executing:  Verifying:
-------------+------------------------+-----------------------
     100,000 |      36.44       20.23 |          0           0
     200,000 |      67.61       38.16 |          0           0
     300,000 |      97.26       55.66 |          0           0
     400,000 |     129.08       74.25 |          0           0
     500,000 |     160.97       91.82 |          0           0

Recall that E(N) and V(N) denote the execution time and verification
time respectively in milliseconds on N ciphertexts. We bound the error
of our approximations compared to the benchmark by:

    |e(N) - E(N)| / E(N) <= 0.01
    |v(N) - V(N)| / V(N) <= 0.01


DECRYPTION    | Data
Ciphertexts: | Communicated:         Proof:
-------------+-----------------------------
     100,000 |       58.8 MB        88.2 MB
     200,000 |      117.6 MB       176.4 MB
     300,000 |      176.4 MB       264.6 MB
     400,000 |      235.2 MB       352.8 MB
     500,000 |      294.0 MB       441.0 MB

Recall that C(N) denotes the number of bytes communicated, and P(N)
denotes the size in bytes of the Fiat-Shamir proof of correctness. We
bound the error of our approximations compared to the benchmark by:

    |c(N) - C(N)| / C(N) <= 0.00
    |p(N) - P(N)| / P(N) <= 0.00

======================================================================

Shows how the running times E(lambda) and V(lambda) for shuffling
and for verifying shuffling, respectively, grow with the threshold
number lambda of parties, and similarly for the number of bytes of
data communicated C(lambda), and the the size of Fiat-Shamir proofs
P(lambda).

Approximations of the functions E(lambda), V(lambda), C(lambda),
and P(lambda):

    e(lambda) = 935576.000 * lambda  +  -637409.667 (ms)
    v(lambda) = 259681.000 * lambda  +  33007.000 (ms)

    c(lambda) = 438505250.000 * lambda  +  438508432.000 (bytes)
    p(lambda) = 438501221.000 * lambda  +  294000357.000 (bytes)


SHUFFLING    | Time (sec)             | Data
Parties:     | Executing:  Verifying: | Communication:      Proof:
-------------+------------------------+---------------------------
           1 |     310.35      292.19 |       877.0 MB    732.5 MB
           2 |    1209.37      553.36 |         1.3 GB      1.2 GB
           3 |    2181.50      811.56 |         1.8 GB      1.6 GB

Recall that E(lambda) and V(lambda) denote the execution time and
verification time respectively in milliseconds on lambda
ciphertexts. We bound the error of our approximations compared to the
benchmark by:

    |e(lambda) - E(lambda)| / E(lambda) <= 0.04
    |v(lambda) - V(lambda)| / V(lambda) <= 0.00

Furthermore, C(lambda) denotes the number of bytes communicated, and
P(lambda) denotes the size in bytes of the proof of correctness. We
bound the error of our approximations compared to the benchmark by:
    |c(lambda) - C(lambda)| / C(lambda) <= 0.00
    |p(lambda) - P(lambda)| / P(lambda) <= 0.00

The fact that the servers run on different hardware explains why the
approximation of the execution time is not precise, i.e., when the
threshold is increased machines with different hardwares are
activated.

======================================================================

Shows how the running times E(lambda) and V(lambda) for decryption
and for verifying decryption, respectively, grow with the threshold
number lambda of parties, and similarly for the number of bytes of
data communicated C(lambda), and the the size of Fiat-Shamir proofs
P(lambda).

Approximations of the functions E(lambda), V(lambda), C(lambda),
and P(lambda):

    e(lambda) = 17356.500 * lambda  +  124684.000 (ms)
    v(lambda) = 16603.000 * lambda  +  56928.667 (ms)

    c(lambda) = 0.000 * lambda  +  294018254.000 (bytes)
    p(lambda) = 147.000 * lambda  +  441001482.000 (bytes)


DECRYPTION   | Time (sec)              | Data
Parties:     | Executing:  Verifying: | Communication:      Proof:
-------------+------------------------+--------------------------
           1 |     141.39       73.12 |       294.0 MB    441.0 MB
           2 |     160.70       90.95 |       294.0 MB    441.0 MB
           3 |     176.10      106.33 |       294.0 MB    441.0 MB

Recall that E(lambda) and V(lambda) denote the execution time and
verification time respectively in milliseconds on lambda
ciphertexts. We bound the error of our approximations compared to the
benchmark by:

    |e(lambda) - E(lambda)| / E(lambda) <= 0.01
    |v(lambda) - V(lambda)| / V(lambda) <= 0.01

Furthermore, C(lambda) denotes the number of bytes communicated, and
P(lambda) denotes the size in bytes of the proof of correctness. We
bound the error of our approximations compared to the benchmark by:

    |c(lambda) - C(lambda)| / C(lambda) <= 0.00
    |p(lambda) - P(lambda)| / P(lambda) <= 0.00

======================================================================

Shows that the running times E(kappa,omega) and V(kappa,omega) for
shuffling of 10000 ciphertexts grow linearly both with
the key width kappa and the width omega of ciphertexts, and that bytes
of data communicated C(kappa,omega) and Fiat-Shamir proof size
P(kappa,omega) grow linearly with both kappa and width.
```

Approximations of the functions E(kappa,omega), V(kappa,omega), C(kappa,omega), and P(kappa,omega):

```
if (kappa = 1 or omega = 1) {

    e(kappa,omega) = 8102.000 * kappa * omega  +  22205.800 (ms)
    v(kappa,omega) = 3211.000 * kappa * omega  +  9485.000 (ms)

} else {

    e(kappa,omega) = 8237.000 * kappa * omega  +  21536.800 (ms)
    v(kappa,omega) = 3212.750 * kappa * omega  +  9447.300 (ms)
}
```

The need for two bilinear maps is due to how group elements are represented. The two cases kappa = 1 or omega = 1 are handled with trees of depth 1, whereas kappa,omega > 1 requires a tree of depth 2.

```
    c(kappa,omega) = 8821.2 * kappa * omega  +  17507.811 (KB)
    p(kappa,omega) = 11760.8 * kappa * omega  +  11662.640 (KB)
```

```
Execution (ms)
    |          1          2          3          4          5
----+-------------------------------------------------------
  1 |      30.45      38.29      46.31      54.64      62.86
  2 |      38.19      54.43      70.98      87.10     104.09
  3 |      46.44      71.42      95.85     120.26     144.95
  4 |      53.42      87.47     120.13     153.27     186.64
  5 |      63.90     104.04     146.26     186.73     228.59
```

```
Verification (ms)
    |          1          2          3          4          5
----+-------------------------------------------------------
  1 |      12.69      15.93      19.14      22.29      25.54
  2 |      15.86      22.39      28.71      35.11      41.56
  3 |      19.14      28.83      38.57      47.91      57.48
  4 |      22.32      35.31      48.13      60.87      73.40
  5 |      25.59      41.51      57.78      73.74      89.64
```

Recall that E(kappa,omega) and V(kappa,omega) denote the actual execution time and verification time respectively in milliseconds on 10000 ciphertexts. We bound the error of our approximations compared to the benchmark by:

```
    |e(kappa,omega) - E(kappa,omega)| / E(kappa,omega) <= 0.02
    |v(kappa,omega) - V(kappa,omega)| / V(kappa,omega) <= 0.01
```

```
Communication
    |          1          2          3          4          5
----+-------------------------------------------------------
  1 |    26.3 MB    35.2 MB    44.0 MB    52.8 MB    61.6 MB
  2 |    35.2 MB    52.8 MB    70.4 MB    88.1 MB   105.7 MB
  3 |    44.0 MB    70.4 MB    96.9 MB   123.4 MB   149.8 MB
  4 |    52.8 MB    88.1 MB   123.4 MB   158.6 MB   193.9 MB
  5 |    61.6 MB   105.7 MB   149.8 MB   193.9 MB   238.0 MB
```

```
Proof size
    |          1          2          3          4          5
----+-------------------------------------------------------
  1 |    23.4 MB    35.2 MB    46.9 MB    58.7 MB    70.5 MB
  2 |    35.2 MB    58.7 MB    82.2 MB   105.7 MB   129.3 MB
  3 |    46.9 MB    82.2 MB   117.5 MB   152.8 MB   188.1 MB
  4 |    58.7 MB   105.8 MB   152.8 MB   199.8 MB   246.9 MB
  5 |    70.5 MB   129.3 MB   188.1 MB   246.9 MB   305.7 MB

Recall that C(kappa,omega) denotes the number of bytes communicated,
and P(kappa,omega) denotes the size in bytes of the Fiat-Shamir proof
of correctness, on 10000 ciphertexts. We bound the error
of our approximations compared to the benchmark by:

    |c(kappa,omega) - C(kappa,omega)| / C(kappa,omega) <= 0.00
    |p(kappa,omega) - P(kappa,omega)| / P(kappa,omega) <= 0.00

======================================================================

Shows that the running times E(kappa,omega) and V(kappa,omega) for
decryption of 10000 ciphertexts grow linearly both with
the key width kappa and the width omega of ciphertexts, and that bytes
of data communicated C(kappa,omega) and Fiat-Shamir proof size
P(kappa,omega) grow linearly with both kappa and width.

Approximations of the functions E(kappa,omega), V(kappa,omega),
C(kappa,omega), and P(kappa,omega):

if (kappa = 1 or omega = 1) {

    e(kappa,omega) = 3530.750 * kappa * omega  +  2876.550 (ms)
    v(kappa,omega) = 2110.500 * kappa * omega  +  589.100 (ms)

} else {

    e(kappa,omega) = 3586.125 * kappa * omega  +  2701.850 (ms)
    v(kappa,omega) = 2084.625 * kappa * omega  +  700.050 (ms)
}
The need for two bilinear maps is due to how group elements are
represented. The two cases kappa = 1 or omega = 1 are handled with
trees of depth 1, whereas kappa,omega > 1 requires a tree of depth 2.

    c(kappa,omega) = 5880.6 * kappa * omega  +  18.560 (KB)
    p(kappa,omega) = 8820.5 * kappa * omega  +  2.599 (KB)


Execution (ms)
    |          1          2          3          4          5
----+-------------------------------------------------------
  1 |       6.35      10.09      13.45      16.99      20.47
  2 |       9.85      17.15      24.08      31.48      38.53
  3 |      13.45      24.01      35.41      44.90      56.29
  4 |      17.04      30.77      45.51      59.98      73.97
  5 |      21.20      38.72      56.23      73.62      90.93
```

```
Verification (ms)
    |          1          2          3          4          5
----+-------------------------------------------------------
  1 |       2.69       4.84       6.89       9.05      11.13
  2 |       4.85       8.96      13.24      17.46      21.53
  3 |       7.00      13.16      19.40      25.82      31.69
  4 |       9.11      17.07      25.35      33.45      42.09
  5 |      11.12      21.28      31.56      41.86      51.90

Recall that E(kappa,omega) and V(kappa,omega) denote the actual
execution time and verification time respectively in milliseconds on
10000 ciphertexts. We bound the error of our
approximations compared to the benchmark by:

    |e(kappa,omega) - E(kappa,omega)| / E(kappa,omega) <= 0.03
    |v(kappa,omega) - V(kappa,omega)| / V(kappa,omega) <= 0.02


Communication
    |          1          2          3          4          5
----+-------------------------------------------------------
  1 |     5.9 MB    11.8 MB    17.7 MB    23.5 MB    29.4 MB
  2 |    11.8 MB    23.5 MB    35.3 MB    47.1 MB    58.8 MB
  3 |    17.7 MB    35.3 MB    52.9 MB    70.6 MB    88.2 MB
  4 |    23.5 MB    47.1 MB    70.6 MB    94.1 MB   117.6 MB
  5 |    29.4 MB    58.8 MB    88.2 MB   117.6 MB   147.0 MB

Proof size
    |          1          2          3          4          5
----+-------------------------------------------------------
  1 |     8.8 MB    17.6 MB    26.5 MB    35.3 MB    44.1 MB
  2 |    17.6 MB    35.3 MB    52.9 MB    70.6 MB    88.2 MB
  3 |    26.5 MB    52.9 MB    79.4 MB   105.8 MB   132.3 MB
  4 |    35.3 MB    70.6 MB   105.9 MB   141.1 MB   176.4 MB
  5 |    44.1 MB    88.2 MB   132.3 MB   176.4 MB   220.5 MB

Recall that C(kappa,omega) denotes the number of bytes communicated,
and P(kappa,omega) denotes the size in bytes of the Fiat-Shamir proof
of correctness, on 10000 ciphertexts. We bound the error
of our approximations compared to the benchmark by:

    |c(kappa,omega) - C(kappa,omega)| / C(kappa,omega) <= 0.00
    |p(kappa,omega) - P(kappa,omega)| / P(kappa,omega) <= 0.00
```

A.3 Multiplicative Group for 3248-bit Safe-prime Modulus

```
Party01:
   host: dog@178.78.255.115
    CPU: Intel(R) Core(TM) i7-4790K CPU (X4 x2) (4.00GHz)
     OS: GNU/Linux 3.13.0-71-generic (Ubuntu 14.04.3 LTS)
Party02:
   host: dog@130.237.224.123
    CPU: AMD Phenom(tm) II X6 1100T Processor (X6) (3.31GHz)
     OS: GNU/Linux 3.13.0-71-generic (Ubuntu 14.04.3 LTS)
```

```
Party03:
   host: dog@130.237.224.227
    CPU: AMD Phenom(tm) II X4 955 Processor (X4) (3.20GHz)
     OS: FreeBSD 10.1-RELEASE
Group is:
ModPGroup(safe-prime modulus=2*order+1. order bit-length = 3247)

=======================================================================

Key generation, 3.24 seconds.

=======================================================================

Shows that the running times of shuffling and decryption adds to
approximately the running time of mixing. Furthermore, the
communication of shuffling and decryption adds approximately to the
communication of mixing. Similarly, the sizes of the proofs of
shuffling and decryption adds approximately to the size of the proof
of mixing.

          Secs:   Operation (100000 ciphertexts):
         721.01   Shuffling
         249.98   Decryption
         970.99   Shuffling + Decryption
         972.23   Mixing

         275.22   Verifying shuffling
          37.11   Verifying decryption
         312.33   Verifying shuffling + Verifying decryption
         310.00   Verifying mixing

 Communication:   Operation (100000 ciphertexts):
       864.6 MB   Shuffling
       164.8 MB   Decryption
         1.0 GB   Shuffling + Decryption
         1.0 GB   Mixing

    Proof size:   Operation (100000 ciphertexts):
       741.2 MB   Shuffling
       247.2 MB   Decryption
       988.4 MB   Shuffling + Decryption
       823.6 MB   Mixing

=======================================================================

Shows that the running times E(N) and V(N) for mixing and for
verifying mixing, respectively, grow linearly with the number N
of ciphertexts, and that the number of bytes of data communicated
C(N), and the the sizes of Fiat-Shamir proofs P(N) grow similarly.

Approximations of the functions E(N), V(N), C(N), and P(N):

    e(N) = 10.266 * N  +  -18450.700 (ms)
    v(N) = 3.090 * N  +  632.600 (ms)

    c(N) = 10294.000 * N  +  41363.000 (bytes)
    p(N) = 8236.000 * N  +  12960.000 (bytes)


MIXING        | Time (sec)              | Time/ciphertext (ms)
Ciphertexts:  | Executing:  Verifying:  | Executing:  Verifying:
-------------+------------------------+-----------------------
      20,000 |     202.04       63.08 |         10           3
      40,000 |     390.41      123.66 |         10           3
      60,000 |     582.56      185.50 |         10           3
      80,000 |     789.17      247.63 |         10           3
     100,000 |    1023.31      310.28 |         10           3

Recall that E(N) and V(N) denote the execution time and verification
time respectively in milliseconds on N ciphertexts. We bound the error
of our approximations compared to the benchmark by:

    |e(N) - E(N)| / E(N) <= 0.08
    |v(N) - V(N)| / V(N) <= 0.01


MIXING        | Data
Ciphertexts:  | Communicated:         Proof:
-------------+----------------------------
      20,000 |      205.9 MB       164.7 MB
      40,000 |      411.8 MB       329.5 MB
      60,000 |      617.7 MB       494.2 MB
      80,000 |      823.6 MB       658.9 MB
     100,000 |        1.0 GB       823.6 MB

Recall that C(N) denotes the number of bytes communicated, and P(N)
denotes the size in bytes of the Fiat-Shamir proof of correctness. We
bound the error of our approximations compared to the benchmark by:

    |c(N) - C(N)| / C(N) <= 0.00
    |p(N) - P(N)| / P(N) <= 0.00

======================================================================

Shows that the running times E(N) and V(N) for shuffling and for
verifying shuffling, respectively, grow linearly with the number N
of ciphertexts, and that the number of bytes of data communicated
C(N), and the the sizes of Fiat-Shamir proofs P(N) grow similarly.

Approximations of the functions E(N), V(N), C(N), and P(N):

    e(N) = 7.559 * N  +  5435.850 (ms)
    v(N) = 2.738 * N  +  988.300 (ms)

    c(N) = 8646.000 * N  +  26987.000 (bytes)
    p(N) = 7412.000 * N  +  9229.000 (bytes)


SHUFFLING     | Time (sec)              | Time/ciphertext (ms)
Ciphertexts:  | Executing:  Verifying:  | Executing:  Verifying:
-------------+------------------------+-----------------------
      20,000 |     155.46       56.10 |          8           3
      40,000 |     314.52      110.24 |          8           3
      60,000 |     468.77      164.70 |          8           3
      80,000 |     596.05      220.26 |          7           3
     100,000 |     760.21      275.18 |          8           3

Recall that E(N) and V(N) denote the execution time and verification
time respectively in milliseconds on N ciphertexts. We bound the error
of our approximations compared to the benchmark by:

    |e(N) - E(N)| / E(N) <= 0.02
    |v(N) - V(N)| / V(N) <= 0.01


SHUFFLING    | Data
Ciphertexts: | Communicated:         Proof:
-------------+-----------------------------
      20,000 |      172.9 MB       148.2 MB
      40,000 |      345.9 MB       296.5 MB
      60,000 |      518.8 MB       444.7 MB
      80,000 |      691.7 MB       593.0 MB
     100,000 |      864.6 MB       741.2 MB

Recall that C(N) denotes the number of bytes communicated, and P(N)
denotes the size in bytes of the Fiat-Shamir proof of correctness. We
bound the error of our approximations compared to the benchmark by:

    |c(N) - C(N)| / C(N) <= 0.00
    |p(N) - P(N)| / P(N) <= 0.00

======================================================================

Shows that the running times E(N) and V(N) for decryption and for
verifying decryption, respectively, grow linearly with the number N
of ciphertexts, and that the number of bytes of data communicated
C(N), and the the sizes of Fiat-Shamir proofs P(N) grow similarly.

Approximations of the functions E(N), V(N), C(N), and P(N):

    e(N) = 2.478 * N  +  2127.350 (ms)
    v(N) = 0.365 * N  +  741.950 (ms)

    c(N) = 1648.000 * N  +  21734.000 (bytes)
    p(N) = 2472.000 * N  +  5446.000 (bytes)


DECRYPTION   | Time (sec)              | Time/ciphertext (ms)
Ciphertexts: | Executing:  Verifying: | Executing:  Verifying:
-------------+------------------------+-----------------------
      20,000 |      52.18        8.13 |          3           0
      40,000 |     101.63       15.36 |          3           0
      60,000 |     150.41       22.55 |          3           0
      80,000 |     199.51       29.86 |          2           0
     100,000 |     250.46       37.34 |          3           0

Recall that E(N) and V(N) denote the execution time and verification
time respectively in milliseconds on N ciphertexts. We bound the error
```

```
of our approximations compared to the benchmark by:

    |e(N) - E(N)| / E(N) <= 0.01
    |v(N) - V(N)| / V(N) <= 0.01


DECRYPTION   | Data
Ciphertexts: | Communicated:         Proof:
-------------+----------------------------
      20,000 |       33.0 MB        49.4 MB
      40,000 |       65.9 MB        98.9 MB
      60,000 |       98.9 MB       148.3 MB
      80,000 |      131.9 MB       197.8 MB
     100,000 |      164.8 MB       247.2 MB

Recall that C(N) denotes the number of bytes communicated, and P(N)
denotes the size in bytes of the Fiat-Shamir proof of correctness. We
bound the error of our approximations compared to the benchmark by:

    |c(N) - C(N)| / C(N) <= 0.00
    |p(N) - P(N)| / P(N) <= 0.00

======================================================================

Shows how the running times E(lambda) and V(lambda) for shuffling
and for verifying shuffling, respectively, grow with the threshold
number lambda of parties, and similarly for the number of bytes of
data communicated C(lambda), and the the size of Fiat-Shamir proofs
P(lambda).

Approximations of the functions E(lambda), V(lambda), C(lambda),
and P(lambda):

    e(lambda) = 480353.500 * lambda  +  -221619.333 (ms)
    v(lambda) = 134957.500 * lambda  +  5327.000 (ms)

    c(lambda) = 288207935.000 * lambda  +  288211117.000 (bytes)
    p(lambda) = 288204171.000 * lambda  +  164800887.000 (bytes)


SHUFFLING   | Time (sec)              | Data
Parties:    | Executing:  Verifying:  | Communication:      Proof:
------------+-------------------------+---------------------------
          1 |     266.68      140.25 |       576.4 MB    453.0 MB
          2 |     723.20      275.31 |       864.6 MB    741.2 MB
          3 |    1227.38      410.17 |         1.2 GB      1.0 GB

Recall that E(lambda) and V(lambda) denote the execution time and
verification time respectively in milliseconds on lambda
ciphertexts. We bound the error of our approximations compared to the
benchmark by:

    |e(lambda) - E(lambda)| / E(lambda) <= 0.03
    |v(lambda) - V(lambda)| / V(lambda) <= 0.00

Furthermore, C(lambda) denotes the number of bytes communicated, and
```

```
P(lambda) denotes the size in bytes of the proof of correctness. We
bound the error of our approximations compared to the benchmark by:

    |c(lambda) - C(lambda)| / C(lambda) <= 0.00
    |p(lambda) - P(lambda)| / P(lambda) <= 0.00

The fact that the servers run on different hardware explains why the
approximation of the execution time is not precise, i.e., when the
threshold is increased machines with different hardwares are
activated.

======================================================================

Shows how the running times E(lambda) and V(lambda) for decryption
and for verifying decryption, respectively, grow with the threshold
number lambda of parties, and similarly for the number of bytes of
data communicated C(lambda), and the the size of Fiat-Shamir proofs
P(lambda).

Approximations of the functions E(lambda), V(lambda), C(lambda),
and P(lambda):

    e(lambda) = 10908.500 * lambda  +  222166.667 (ms)
    v(lambda) = 10528.500 * lambda  +  10497.667 (ms)

    c(lambda) = 0.000 * lambda  +  164821734.000 (bytes)
    p(lambda) = 412.000 * lambda  +  247204622.000 (bytes)


DECRYPTION  | Time (sec)               | Data
Parties:    | Executing:  Verifying: | Communication:      Proof:
------------+------------------------+---------------------------
          1 |     230.89       18.19 |       164.8 MB    247.2 MB
          2 |     248.35       37.23 |       164.8 MB    247.2 MB
          3 |     252.71       39.24 |       164.8 MB    247.2 MB

Recall that E(lambda) and V(lambda) denote the execution time and
verification time respectively in milliseconds on lambda
ciphertexts. We bound the error of our approximations compared to the
benchmark by:

    |e(lambda) - E(lambda)| / E(lambda) <= 0.02
    |v(lambda) - V(lambda)| / V(lambda) <= 0.16

Furthermore, C(lambda) denotes the number of bytes communicated, and
P(lambda) denotes the size in bytes of the proof of correctness. We
bound the error of our approximations compared to the benchmark by:

    |c(lambda) - C(lambda)| / C(lambda) <= 0.00
    |p(lambda) - P(lambda)| / P(lambda) <= 0.00

======================================================================

Shows that the running times E(kappa,omega) and V(kappa,omega) for
shuffling of 2000 ciphertexts grow linearly both with
the key width kappa and the width omega of ciphertexts, and that bytes
```

of data communicated C(kappa,omega) and Fiat-Shamir proof size P(kappa,omega) grow linearly with both kappa and width.

Approximations of the functions E(kappa,omega), V(kappa,omega), C(kappa,omega), and P(kappa,omega):

```
if (kappa = 1 or omega = 1) {

    e(kappa,omega) = 3890.750 * kappa * omega  +  19515.150 (ms)
    v(kappa,omega) = 1196.250 * kappa * omega  +  6347.050 (ms)

} else {

    e(kappa,omega) = 3884.375 * kappa * omega  +  19505.950 (ms)
    v(kappa,omega) = 1180.250 * kappa * omega  +  6442.300 (ms)
}
```

The need for two bilinear maps is due to how group elements are represented. The two cases kappa = 1 or omega = 1 are handled with trees of depth 1, whereas kappa,omega > 1 requires a tree of depth 2.

```
    c(kappa,omega) = 4947.8 * kappa * omega  +  12371.256 (KB)
    p(kappa,omega) = 6594.6 * kappa * omega  +  8240.390 (KB)
```

Execution (ms)

	1	2	3	4	5
1	23.48	27.27	31.21	34.94	39.04
2	27.35	34.94	42.67	50.69	58.42
3	30.22	42.37	54.06	66.10	78.17
4	34.38	50.21	66.39	81.75	97.89
5	39.04	58.32	77.85	97.65	117.78

Verification (ms)

	1	2	3	4	5
1	7.56	8.74	9.92	11.12	12.34
2	8.80	11.14	13.46	15.96	18.25
3	9.96	13.54	17.04	20.63	24.22
4	11.19	15.89	20.63	25.34	30.03
5	12.36	18.24	24.09	29.96	35.98

Recall that E(kappa,omega) and V(kappa,omega) denote the actual execution time and verification time respectively in milliseconds on 2000 ciphertexts. We bound the error of our approximations compared to the benchmark by:

```
    |e(kappa,omega) - E(kappa,omega)| / E(kappa,omega) <= 0.03
    |v(kappa,omega) - V(kappa,omega)| / V(kappa,omega) <= 0.01
```

Communication

	1	2	3	4	5
1	17.3 MB	22.3 MB	27.2 MB	32.2 MB	37.1 MB
2	22.3 MB	32.2 MB	42.1 MB	52.0 MB	61.8 MB

```
  3 |    27.2 MB    42.1 MB    56.9 MB    71.7 MB    86.6 MB
  4 |    32.2 MB    52.0 MB    71.7 MB    91.5 MB   111.3 MB
  5 |    37.1 MB    61.8 MB    86.6 MB   111.3 MB   136.1 MB

Proof size
    |          1          2          3          4          5
----+------------------------------------------------------
  1 |    14.8 MB    21.4 MB    28.0 MB    34.6 MB    41.2 MB
  2 |    21.4 MB    34.6 MB    47.8 MB    61.0 MB    74.2 MB
  3 |    28.0 MB    47.8 MB    67.6 MB    87.4 MB   107.2 MB
  4 |    34.6 MB    61.0 MB    87.4 MB   113.8 MB   140.1 MB
  5 |    41.2 MB    74.2 MB   107.2 MB   140.1 MB   173.1 MB

Recall that C(kappa,omega) denotes the number of bytes communicated,
and P(kappa,omega) denotes the size in bytes of the Fiat-Shamir proof
of correctness, on 2000 ciphertexts. We bound the error
of our approximations compared to the benchmark by:

    |c(kappa,omega) - C(kappa,omega)| / C(kappa,omega) <= 0.00
    |p(kappa,omega) - P(kappa,omega)| / P(kappa,omega) <= 0.00

======================================================================

Shows that the running times E(kappa,omega) and V(kappa,omega) for
decryption of 2000 ciphertexts grow linearly both with
the key width kappa and the width omega of ciphertexts, and that bytes
of data communicated C(kappa,omega) and Fiat-Shamir proof size
P(kappa,omega) grow linearly with both kappa and width.

Approximations of the functions E(kappa,omega), V(kappa,omega),
C(kappa,omega), and P(kappa,omega):

if (kappa = 1 or omega = 1) {

    e(kappa,omega) = 5405.500 * kappa * omega  +  2631.300 (ms)
    v(kappa,omega) = 995.500 * kappa * omega  +  567.500 (ms)

} else {

    e(kappa,omega) = 5333.625 * kappa * omega  +  3439.650 (ms)
    v(kappa,omega) = 976.375 * kappa * omega  +  616.750 (ms)
}
The need for two bilinear maps is due to how group elements are
represented. The two cases kappa = 1 or omega = 1 are handled with
trees of depth 1, whereas kappa,omega > 1 requires a tree of depth 2.

    c(kappa,omega) = 3297.7 * kappa * omega  +  23.400 (KB)
    p(kappa,omega) = 4945.3 * kappa * omega  +  8.349 (KB)


Execution (ms)
    |          1          2          3          4          5
----+------------------------------------------------------
  1 |       7.96      13.49      18.67      24.54      29.58
  2 |      13.59      26.32      35.19      45.85      56.26
  3 |      18.70      34.49      50.42      66.16      82.70
  4 |      23.80      45.08      66.06      87.61     109.16
  5 |      29.07      55.93      82.55     109.36     135.55

Verification (ms)
    |          1          2          3          4          5
----+-------------------------------------------------------
  1 |       1.55       2.58       3.56       4.55       5.53
  2 |       2.57       4.55       6.48       8.39      10.38
  3 |       3.59       6.50       9.48      12.21      15.16
  4 |       4.57       8.36      12.32      16.23      19.96
  5 |       5.61      10.50      15.25      19.90      24.73

Recall that E(kappa,omega) and V(kappa,omega) denote the actual
execution time and verification time respectively in milliseconds on
2000 ciphertexts. We bound the error of our
approximations compared to the benchmark by:

    |e(kappa,omega) - E(kappa,omega)| / E(kappa,omega) <= 0.06
    |v(kappa,omega) - V(kappa,omega)| / V(kappa,omega) <= 0.01


Communication
    |          1          2          3          4          5
----+-------------------------------------------------------
  1 |     3.3 MB     6.6 MB     9.9 MB    13.2 MB    16.5 MB
  2 |     6.6 MB    13.2 MB    19.8 MB    26.4 MB    33.0 MB
  3 |     9.9 MB    19.8 MB    29.7 MB    39.6 MB    49.5 MB
  4 |    13.2 MB    26.4 MB    39.6 MB    52.8 MB    66.0 MB
  5 |    16.5 MB    33.0 MB    49.5 MB    66.0 MB    82.5 MB

Proof size
    |          1          2          3          4          5
----+-------------------------------------------------------
  1 |     4.9 MB     9.9 MB    14.8 MB    19.8 MB    24.7 MB
  2 |     9.9 MB    19.8 MB    29.7 MB    39.6 MB    49.5 MB
  3 |    14.8 MB    29.7 MB    44.5 MB    59.4 MB    74.2 MB
  4 |    19.8 MB    39.6 MB    59.4 MB    79.1 MB    98.9 MB
  5 |    24.7 MB    49.5 MB    74.2 MB    98.9 MB   123.7 MB

Recall that C(kappa,omega) denotes the number of bytes communicated,
and P(kappa,omega) denotes the size in bytes of the Fiat-Shamir proof
of correctness, on 2000 ciphertexts. We bound the error
of our approximations compared to the benchmark by:

    |c(kappa,omega) - C(kappa,omega)| / C(kappa,omega) <= 0.00
    |p(kappa,omega) - P(kappa,omega)| / P(kappa,omega) <= 0.00
```

A.4 Multiplicative Group for 15492-bit Safe-prime Modulus

```
Party01:
   host: dog@178.78.255.115
    CPU: Intel(R) Core(TM) i7-4790K CPU (X4 x2) (4.00GHz)
     OS: GNU/Linux 3.13.0-71-generic (Ubuntu 14.04.3 LTS)
Party02:
   host: dog@130.237.224.123
    CPU: AMD Phenom(tm) II X6 1100T Processor (X6) (3.31GHz)
     OS: GNU/Linux 3.13.0-71-generic (Ubuntu 14.04.3 LTS)
Party03:
   host: dog@130.237.224.227
    CPU: AMD Phenom(tm) II X4 955 Processor (X4) (3.20GHz)
     OS: FreeBSD 10.1-RELEASE
Group is:

======================================================================

Key generation, 10.01 seconds.

======================================================================

Shows that the running times of shuffling and decryption adds to
approximately the running time of mixing. Furthermore, the
communication of shuffling and decryption adds approximately to the
communication of mixing. Similarly, the sizes of the proofs of
shuffling and decryption adds approximately to the size of the proof
of mixing.

          Secs:   Operation (2000 ciphertexts):
         583.68   Shuffling
         251.89   Decryption
         835.57   Shuffling + Decryption
         832.59   Mixing

         144.05   Verifying shuffling
          23.26   Verifying decryption
         167.31   Verifying shuffling + Verifying decryption
         155.43   Verifying mixing

 Communication:   Operation (2000 ciphertexts):
        81.6 MB   Shuffling
        15.6 MB   Decryption
        97.2 MB   Shuffling + Decryption
        97.2 MB   Mixing

    Proof size:   Operation (2000 ciphertexts):
        70.0 MB   Shuffling
        23.3 MB   Decryption
        93.3 MB   Shuffling + Decryption
        77.7 MB   Mixing

======================================================================

Shows that the running times E(N) and V(N) for mixing and for
verifying mixing, respectively, grow linearly with the number N
of ciphertexts, and that the number of bytes of data communicated
C(N), and the the sizes of Fiat-Shamir proofs P(N) grow similarly.

Approximations of the functions E(N), V(N), C(N), and P(N):

    e(N) = 381.723 * N  +  70146.050 (ms)
    v(N) = 64.305 * N  +  26873.000 (ms)
    c(N) = 48550.000 * N  +  101049.000 (bytes)
    p(N) = 38840.000 * N  +  60401.000 (bytes)


MIXING       | Time (sec)             | Time/ciphertext (ms)
Ciphertexts: | Executing:  Verifying: | Executing:  Verifying:
-------------+------------------------+-----------------------
         400 |     222.96       52.47 |        557         131
         800 |     375.11       78.28 |        469          98
       1,200 |     527.86      104.38 |        440          87
       1,600 |     681.42      129.70 |        426          81
       2,000 |     833.72      155.36 |        417          78

Recall that E(N) and V(N) denote the execution time and verification
time respectively in milliseconds on N ciphertexts. We bound the error
of our approximations compared to the benchmark by:

    |e(N) - E(N)| / E(N) <= 0.00
    |v(N) - V(N)| / V(N) <= 0.00


MIXING       | Data
Ciphertexts: | Communicated:         Proof:
-------------+-----------------------------
         400 |       19.5 MB       15.6 MB
         800 |       38.9 MB       31.1 MB
       1,200 |       58.4 MB       46.7 MB
       1,600 |       77.8 MB       62.2 MB
       2,000 |       97.2 MB       77.7 MB

Recall that C(N) denotes the number of bytes communicated, and P(N)
denotes the size in bytes of the Fiat-Shamir proof of correctness. We
bound the error of our approximations compared to the benchmark by:

    |c(N) - C(N)| / C(N) <= 0.00
    |p(N) - P(N)| / P(N) <= 0.00

======================================================================

Shows that the running times E(N) and V(N) for shuffling and for
verifying shuffling, respectively, grow linearly with the number N
of ciphertexts, and that the number of bytes of data communicated
C(N), and the the sizes of Fiat-Shamir proofs P(N) grow similarly.

Approximations of the functions E(N), V(N), C(N), and P(N):

    e(N) = 260.128 * N  +  63147.450 (ms)
    v(N) = 59.695 * N  +  24595.000 (ms)

    c(N) = 40782.000 * N  +  68309.000 (bytes)
    p(N) = 34956.000 * N  +  42897.000 (bytes)


SHUFFLING    | Time (sec)             | Time/ciphertext (ms)
Ciphertexts: | Executing:  Verifying: | Executing:  Verifying:
```

```
-------------+-----------------------+----------------------
          400 |     167.73       48.46 |        419        121
          800 |     270.59       72.37 |        338         90
        1,200 |     374.36       96.22 |        312         80
        1,600 |     479.88      120.12 |        300         75
        2,000 |     583.94      143.97 |        292         72

Recall that E(N) and V(N) denote the execution time and verification
time respectively in milliseconds on N ciphertexts. We bound the error
of our approximations compared to the benchmark by:

    |e(N) - E(N)| / E(N) <= 0.00
    |v(N) - V(N)| / V(N) <= 0.00


SHUFFLING    | Data
Ciphertexts: | Communicated:         Proof:
-------------+----------------------------
          400 |       16.4 MB        14.0 MB
          800 |       32.7 MB        28.0 MB
        1,200 |       49.0 MB        42.0 MB
        1,600 |       65.3 MB        56.0 MB
        2,000 |       81.6 MB        70.0 MB

Recall that C(N) denotes the number of bytes communicated, and P(N)
denotes the size in bytes of the Fiat-Shamir proof of correctness. We
bound the error of our approximations compared to the benchmark by:

    |c(N) - C(N)| / C(N) <= 0.00
    |p(N) - P(N)| / P(N) <= 0.00

======================================================================

Shows that the running times E(N) and V(N) for decryption and for
verifying decryption, respectively, grow linearly with the number N
of ciphertexts, and that the number of bytes of data communicated
C(N), and the the sizes of Fiat-Shamir proofs P(N) grow similarly.

Approximations of the functions E(N), V(N), C(N), and P(N):

    e(N) = 121.757 * N  +  8467.400 (ms)
    v(N) = 4.766 * N  +  13750.450 (ms)

    c(N) = 7768.000 * N  +  40098.000 (bytes)
    p(N) = 11652.000 * N  +  25339.000 (bytes)


DECRYPTION   | Time (sec)             | Time/ciphertext (ms)
Ciphertexts: | Executing:  Verifying: | Executing:  Verifying:
-------------+-----------------------+----------------------
          400 |      57.22       15.65 |        143         39
          800 |     105.80       17.59 |        132         22
        1,200 |     154.61       19.46 |        129         16
        1,600 |     203.21       21.36 |        127         13
        2,000 |     252.03       23.28 |        126         12
```

```
Recall that E(N) and V(N) denote the execution time and verification
time respectively in milliseconds on N ciphertexts. We bound the error
of our approximations compared to the benchmark by:

    |e(N) - E(N)| / E(N) <= 0.00
    |v(N) - V(N)| / V(N) <= 0.00


DECRYPTION   | Data
Ciphertexts: | Communicated:          Proof:
-------------+----------------------------
         400 |          3.1 MB         4.7 MB
         800 |          6.3 MB         9.3 MB
       1,200 |          9.4 MB        14.0 MB
       1,600 |         12.5 MB        18.7 MB
       2,000 |         15.6 MB        23.3 MB

Recall that C(N) denotes the number of bytes communicated, and P(N)
denotes the size in bytes of the Fiat-Shamir proof of correctness. We
bound the error of our approximations compared to the benchmark by:

    |c(N) - C(N)| / C(N) <= 0.00
    |p(N) - P(N)| / P(N) <= 0.00

======================================================================

Shows how the running times E(lambda) and V(lambda) for shuffling
and for verifying shuffling, respectively, grow with the threshold
number lambda of parties, and similarly for the number of bytes of
data communicated C(lambda), and the the size of Fiat-Shamir proofs
P(lambda).

Approximations of the functions E(lambda), V(lambda), C(lambda),
and P(lambda):

    e(lambda) = 338119.500 * lambda  +  -71766.000 (ms)
    v(lambda) = 65715.500 * lambda  +  12520.667 (ms)

    c(lambda) = 27209709.000 * lambda  +  27212891.000 (bytes)
    p(lambda) = 27207475.000 * lambda  +  15539947.000 (bytes)


SHUFFLING    | Time (sec)                | Data
Parties:     | Executing:  Verifying:  | Communication:      Proof:
-------------+-------------------------+---------------------------
           1 |      276.99       78.21 |          54.4 MB     42.7 MB
           2 |      583.20      144.00 |          81.6 MB     70.0 MB
           3 |      953.23      209.64 |         108.8 MB     97.2 MB

Recall that E(lambda) and V(lambda) denote the execution time and
verification time respectively in milliseconds on lambda
ciphertexts. We bound the error of our approximations compared to the
benchmark by:

    |e(lambda) - E(lambda)| / E(lambda) <= 0.04
    |v(lambda) - V(lambda)| / V(lambda) <= 0.00
```

Furthermore, C(lambda) denotes the number of bytes communicated, and P(lambda) denotes the size in bytes of the proof of correctness. We bound the error of our approximations compared to the benchmark by:

```
    |c(lambda) - C(lambda)| / C(lambda) <= 0.00
    |p(lambda) - P(lambda)| / P(lambda) <= 0.00
```

The fact that the servers run on different hardware explains why the approximation of the execution time is not precise, i.e., when the threshold is increased machines with different hardwares are activated.

```
===============================================================
```

Shows how the running times E(lambda) and V(lambda) for decryption and for verifying decryption, respectively, grow with the threshold number lambda of parties, and similarly for the number of bytes of data communicated C(lambda), and the the size of Fiat-Shamir proofs P(lambda).

Approximations of the functions E(lambda), V(lambda), C(lambda), and P(lambda):

```
    e(lambda) = 4099.000 * lambda  +  241696.667 (ms)
    v(lambda) = 3987.000 * lambda  +  13080.667 (ms)

    c(lambda) = 0.000 * lambda  +  15576098.000 (bytes)
    p(lambda) = 1942.000 * lambda  +  23325455.000 (bytes)


DECRYPTION  | Time (sec)              | Data
Parties:    | Executing:  Verifying: | Communication:      Proof:
------------+------------------------+--------------------------
          1 |     244.57       15.96 |         15.6 MB     23.3 MB
          2 |     252.35       23.27 |         15.6 MB     23.3 MB
          3 |     252.77       23.93 |         15.6 MB     23.3 MB
```

Recall that E(lambda) and V(lambda) denote the execution time and verification time respectively in milliseconds on lambda ciphertexts. We bound the error of our approximations compared to the benchmark by:

```
    |e(lambda) - E(lambda)| / E(lambda) <= 0.01
    |v(lambda) - V(lambda)| / V(lambda) <= 0.10
```

Furthermore, C(lambda) denotes the number of bytes communicated, and P(lambda) denotes the size in bytes of the proof of correctness. We bound the error of our approximations compared to the benchmark by:

```
    |c(lambda) - C(lambda)| / C(lambda) <= 0.00
    |p(lambda) - P(lambda)| / P(lambda) <= 0.00

===============================================================
```

Shows that the running times E(kappa,omega) and V(kappa,omega) for

shuffling of 40 ciphertexts grow linearly both with
the key width kappa and the width omega of ciphertexts, and that bytes
of data communicated C(kappa,omega) and Fiat-Shamir proof size
P(kappa,omega) grow linearly with both kappa and width.

Approximations of the functions E(kappa,omega), V(kappa,omega),
C(kappa,omega), and P(kappa,omega):

```
if (kappa = 1 or omega = 1) {

    e(kappa,omega) = 8755.500 * kappa * omega  +  31632.300 (ms)
    v(kappa,omega) = 2485.250 * kappa * omega  +  19852.250 (ms)

} else {

    e(kappa,omega) = 8792.125 * kappa * omega  +  31561.250 (ms)
    v(kappa,omega) = 2457.500 * kappa * omega  +  19966.400 (ms)
}
```

The need for two bilinear maps is due to how group elements are
represented. The two cases kappa = 1 or omega = 1 are handled with
trees of depth 1, whereas kappa,omega > 1 requires a tree of depth 2.

```
    c(kappa,omega) = 483.6 * kappa * omega  +  1216.005 (KB)
    p(kappa,omega) = 633.2 * kappa * omega  +  815.797 (KB)
```

```
Execution (ms)
    |         1          2          3          4          5
----+--------------------------------------------------------
  1 |     40.38      49.17      57.79      66.76      75.40
  2 |     49.11      66.76      84.35     101.90     119.45
  3 |     57.93      84.34     111.05     137.22     163.70
  4 |     66.84     102.01     137.09     172.38     207.53
  5 |     75.33     119.53     163.59     207.84     251.88

Verification (ms)
    |         1          2          3          4          5
----+--------------------------------------------------------
  1 |     22.35      24.86      27.26      29.79      32.29
  2 |     24.87      29.82      34.71      39.63      44.53
  3 |     27.34      34.69      42.08      49.47      56.93
  4 |     29.76      39.62      49.47      59.37      69.19
  5 |     32.24      44.54      56.99      69.30      81.47
```

Recall that E(kappa,omega) and V(kappa,omega) denote the actual
execution time and verification time respectively in milliseconds on
40 ciphertexts. We bound the error of our
approximations compared to the benchmark by:

```
    |e(kappa,omega) - E(kappa,omega)| / E(kappa,omega) <= 0.00
    |v(kappa,omega) - V(kappa,omega)| / V(kappa,omega) <= 0.00
```

```
Communication
    |         1          2          3          4          5
----+--------------------------------------------------------
```

```
  1 |     1.7 MB     2.2 MB     2.7 MB     3.2 MB     3.6 MB
  2 |     2.2 MB     3.2 MB     4.1 MB     5.1 MB     6.1 MB
  3 |     2.7 MB     4.1 MB     5.6 MB     7.0 MB     8.5 MB
  4 |     3.2 MB     5.1 MB     7.0 MB     9.0 MB    10.9 MB
  5 |     3.6 MB     6.1 MB     8.5 MB    10.9 MB    13.3 MB

Proof size
    |          1          2          3          4          5
----+-------------------------------------------------------
  1 |     1.4 MB     2.1 MB     2.7 MB     3.3 MB     4.0 MB
  2 |     2.1 MB     3.3 MB     4.6 MB     5.9 MB     7.1 MB
  3 |     2.7 MB     4.6 MB     6.5 MB     8.4 MB    10.3 MB
  4 |     3.4 MB     5.9 MB     8.4 MB    11.0 MB    13.5 MB
  5 |     4.0 MB     7.2 MB    10.3 MB    13.5 MB    16.7 MB

Recall that C(kappa,omega) denotes the number of bytes communicated,
and P(kappa,omega) denotes the size in bytes of the Fiat-Shamir proof
of correctness, on 40 ciphertexts. We bound the error
of our approximations compared to the benchmark by:

    |c(kappa,omega) - C(kappa,omega)| / C(kappa,omega) <= 0.00
    |p(kappa,omega) - P(kappa,omega)| / P(kappa,omega) <= 0.01

======================================================================

Shows that the running times E(kappa,omega) and V(kappa,omega) for
decryption of 40 ciphertexts grow linearly both with
the key width kappa and the width omega of ciphertexts, and that bytes
of data communicated C(kappa,omega) and Fiat-Shamir proof size
P(kappa,omega) grow linearly with both kappa and width.

Approximations of the functions E(kappa,omega), V(kappa,omega),
C(kappa,omega), and P(kappa,omega):

if (kappa = 1 or omega = 1) {

    e(kappa,omega) = 8068.250 * kappa * omega  +  4907.250 (ms)
    v(kappa,omega) = 1407.750 * kappa * omega  +  12452.550 (ms)

} else {

    e(kappa,omega) = 8089.250 * kappa * omega  +  7581.300 (ms)
    v(kappa,omega) = 1391.750 * kappa * omega  +  13469.300 (ms)
}
The need for two bilinear maps is due to how group elements are
represented. The two cases kappa = 1 or omega = 1 are handled with
trees of depth 1, whereas kappa,omega > 1 requires a tree of depth 2.

    c(kappa,omega) = 318.5 * kappa * omega  +  47.888 (KB)
    p(kappa,omega) = 472.0 * kappa * omega  +  38.955 (KB)


Execution (ms)
    |          1          2          3          4          5
----+-------------------------------------------------------
  1 |      12.89      21.05      29.18      37.28      45.16
  2 |      23.73      39.97      56.24      72.19      88.45
  3 |      34.60      58.83      83.18     107.17     131.27
  4 |      45.65      78.00     110.28     142.16     174.23
  5 |      56.73      96.72     137.10     177.32     217.44

Verification (ms)
    |          1          2          3          4          5
----+-------------------------------------------------------
  1 |      13.85      15.28      16.69      18.08      19.48
  2 |      16.23      19.01      21.84      24.66      27.36
  3 |      18.56      22.72      26.87      31.10      35.34
  4 |      20.91      26.44      32.08      37.55      43.10
  5 |      24.73      30.25      37.17      44.14      51.08

Recall that E(kappa,omega) and V(kappa,omega) denote the actual
execution time and verification time respectively in milliseconds on
40 ciphertexts. We bound the error of our
approximations compared to the benchmark by:

    |e(kappa,omega) - E(kappa,omega)| / E(kappa,omega) <= 0.20
    |v(kappa,omega) - V(kappa,omega)| / V(kappa,omega) <= 0.21


Communication
    |          1          2          3          4          5
----+-------------------------------------------------------
  1 |   350.8 KB   669.4 KB   987.9 KB     1.3 MB     1.6 MB
  2 |   684.9 KB     1.3 MB     2.0 MB     2.6 MB     3.2 MB
  3 |     1.0 MB     2.0 MB     2.9 MB     3.9 MB     4.8 MB
  4 |     1.4 MB     2.6 MB     3.9 MB     5.2 MB     6.4 MB
  5 |     1.7 MB     3.3 MB     4.9 MB     6.5 MB     8.1 MB

Proof size
    |          1          2          3          4          5
----+-------------------------------------------------------
  1 |   491.4 KB   963.4 KB     1.4 MB     1.9 MB     2.4 MB
  2 |   982.9 KB     1.9 MB     2.9 MB     3.8 MB     4.8 MB
  3 |     1.5 MB     2.9 MB     4.3 MB     5.7 MB     7.1 MB
  4 |     2.0 MB     3.9 MB     5.7 MB     7.6 MB     9.5 MB
  5 |     2.5 MB     4.8 MB     7.2 MB     9.5 MB    11.9 MB

Recall that C(kappa,omega) denotes the number of bytes communicated,
and P(kappa,omega) denotes the size in bytes of the Fiat-Shamir proof
of correctness, on 40 ciphertexts. We bound the error
of our approximations compared to the benchmark by:

    |c(kappa,omega) - C(kappa,omega)| / C(kappa,omega) <= 0.04
    |p(kappa,omega) - P(kappa,omega)| / P(kappa,omega) <= 0.03
```