Benchmarks

Verificatum JavaScript Cryptographic Library
Version: 1.1.1
Date: 2020-04-02
Browser: Unable to detect (best effort detection)

Each benchmark computes at least 3 samples of the measured quantity and takes the average. The running time of the layout engine is not included in the running time. Note that benchmarks give different results using different JavaScript engines and that other factors may influence the results. In particular, the last measurements may not reflect actual running time (it would be faster), since the garbage collection is poor on some platforms. Thus, interpret the results with care and investigate the particular case you are interested in before drawing any hard conclusions.

The library is designed with pre-computation in mind all the way up to the highest abstraction layer. Combined with a web worker as in this benchmark, such computations can be done in the background.

Exponentiation

The running times include the cost of generating random exponents, which gives an upper bound of the running time of the actual exponentiation.

The running time of modular exponentiation is increased by almost factor of 8 when the bit size of the modulus is doubled as expected from a relatively naive implementation. A similar behavior can be seen in the elliptic curves with growing field size, but with a slightly smaller factor.

Standard Multiplicative Groups

Group	ms / exp
modp768	8.7
modp1024	12.7
modp1536	33.7
modp2048	73.0
modp3072	223.0
modp4096	499.0
modp6144	1560.7
modp8192	3562.3

Standard Elliptic Curves

Group	ms / exp
prime192v1	6.7
prime192v2	4.0
prime192v3	4.0
prime256v1	9.3
prime239v1	6.3
prime239v3	6.3
secp192k1	6.3
secp192r1	4.3

secp224k1	6.3
secp224r1	5.3
secp256k1	8.7
secp256r1	7.0
secp384r1	15.7
secp521r1	31.3
brainpoolp192r1	6.0
brainpoolp224r1	6.3
brainpoolp256r1	9.7
brainpoolp320r1	14.7
brainpoolp384r1	20.0
brainpoolp512r1	39.0
P-192	4.7
P-224	5.0
P-256	7.7
P-384	16.7
P-521	30.7

Fixed-basis Exponentiation for Selected Groups

Here zero gives plain exponentiation for easy reference.

Group \ Exps	0	1	2	4	8	16	32
modp3072	217.7	273.0	171.2	111.7	79.8	61.5	49.0
modp4096	493.7	650.0	451.7	277.4	190.9	143.7	116.6
modp6144	1559.7	2012.3	1257.8	817.0	575.7	424.9	342.4

Encryption over Selected Groups

The running time of encryption grows linearly with the width, so the values for greater widths are readily extrapolated from the given numbers.

El Gamal Encryption (ms / ciphertext)

This is only benchmarked for the purpose of comparison. It is not CCA2 secure or even non-malleable, and should therefore not be used unless other equivalent mechanisms are in place.

Group \ Width	1	2	3	4
modp3072	399.7	801.0	1181.0	1579.7
modp4096	903.0	1807.7	2742.3	3633.0
modp6144	2875.0	5902.0	8810.7	11734.7
P-256	17.0	29.0	43.0	58.7
secp384r1	30.7	60.3	89.7	120.7
P-521	58.0	115.3	174.7	237.3

El Gamal Encryption with Label and ZKPoK (ms / ciphertext)

This is the simplest cryptosystem, which is non-malleable in a standard heuristic sense, i.e., it is the El Gamal cryptosystem with proof of knowledge of the randomness turned non-interactive using the Fiat-

Shamir heuristic. This is not provably secure in the random oracle model, but likely to be secure.

Group \ Width	1	2	3	4
modp3072	596.0	1196.3	1794.0	2391.3
modp4096	1320.7	2672.3	4036.7	5337.0
modp6144	4262.0	8533.7	12749.3	16786.3
P-256	25.7	46.7	66.3	89.3
secp384r1	47.0	93.3	138.7	187.3
P-521	89.3	176.7	268.7	354.7

Naor-Yung with Label and ZKPoK (ms / ciphertext)

This is the Naor-Yung cryptosystem, which is provably CCA2 secure with the Fiat-Shamir heuristic in the random oracle model.

Group \ Width	1	2	3	4
modp3072	1006.7	2076.0	3046.7	4030.7
modp4096	2310.7	4630.7	6996.7	9288.0
modp6144	7314.3	14610.0	21907.3	29405.7
P-256	39.7	76.3	114.3	153.0
secp384r1	78.0	159.0	237.7	317.7
P-521	152.0	306.0	456.7	612.7