Benchmarks

Verificatum JavaScript Cryptographic Library
Version: 1.1.0
Date: 2016-08-31
Browser: Chrome 1+ (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	10.7
modp1024	19.3
modp1536	47.0
modp2048	104.0
modp3072	329.3
modp4096	711.0
modp6144	2214.0
modp8192	4991.3

Standard Elliptic Curves

Group	ms / exp
prime192v1	6.3
prime192v2	6.7
prime192v3	6.3
prime256v1	14.3
prime239v1	9.0

prime239v3	10.3
secp192k1	7.7
secp192r1	6.7
secp224k1	9.0
secp224r1	7.7
secp256k1	13.0
secp256r1	10.3
secp384r1	25.0
secp521r1	48.0
brainpoolp192r1	9.0
brainpoolp224r1	10.3
brainpoolp256r1	15.7
brainpoolp320r1	22.0
brainpoolp384r1	31.0
brainpoolp512r1	61.0
P-192	8.0
P-224	9.0
P-256	11.7
P-384	25.0
P-521	48.0

Fixed-basis Exponentiation for Selected Groups

Here zero gives plain exponentiation for easy reference.

Group \ Exps	0	1	2	4	8	16	32
modp3072	329.3	413.0	245.3	162.5	117.0	89.3	71.5
modp4096	703.0	896.0	634.8	385.5	273.0	202.0	164.5
modp6144	2221.7	2819.0	1750.0	1138.5	806.9	594.6	485.2

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	569.0	1148.7	1726.7	2313.7
modp4096	1302.3	2590.0	3905.7	5164.7
modp6144	4095.7	8288.7	12466.7	16845.7
P-256	26.0	47.0	69.0	89.7

secp384r1	52.0	96.7	149.7	192.7
P-521	91.0	183.7	286.7	359.3

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	866.0	1728.0	2594.0	3462.7
modp4096	1966.7	3892.3	5819.7	7791.7
modp6144	6101.0	12117.7	18134.0	24192.0
P-256	39.0	70.3	113.0	140.7
secp384r1	74.3	144.3	230.3	290.3
P-521	136.7	285.3	423.0	555.0

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	1509.3	3016.0	4598.0	6051.3
modp4096	3420.0	6874.7	10234.7	13633.0
modp6144	10580.7	21143.3	31712.3	42117.3
P-256	59.7	121.0	179.7	239.7
secp384r1	122.3	255.3	381.3	504.0
P-521	245.0	492.0	717.7	955.7