RSArmageddon v.1.0 athena

User manual

Vittorio Mignini aka M1gnus Nalin Dhingra aka Lotus

September 4, 2020

Contents

Part I

Premise and contact

1 Premise

The goal of the software is to manipulate RSA cryptosystem and attack RSA public keys. The power of this software is given by SageMath, a truly powerful math framework whose functions is used to perform heavy operations on numbers in the attack scripts. Please feel free to contact us if you found anything that can be improved or problems that should be corrected.

2 Contacts

M1GNUS
`email:` m1gnus@protonmail.com
`telegram:` @m1gnus
`site:` https://pgiatasti.it

Part II

Data types

3 list

`string[,string[,...]]`
List of string. Is possible to insert one of more string separated by a comma.

4 plaintext

`string[:type]`
Plaintext used during the encryption process, is possible to specify the plaintext type, that can be choosen from the following:
`-str`
`-dec`
`-hex`
`-oct`
`-bin`
if type is not chosen from this ones or type is not specified, then the type is set to `str` by default.

5 ciphertext

`string[:type]`
Ciphertext used during the decryption process, is possible to specify the ciphertext type, that can be choosen from the following:
`-dec`
`-hex`
`-oct`
`-bin`
if type is not chosen from this ones or type is not specified, then the type is set to `dec` by default.

6 int

`string[:base]`
Integer value. Is possible to use the standard notation for hexadecimal (0x...), octal (0...) and binary (0b...) representations of integers.
Is also possible to specify an integer `0 < base < 10`. If `base` <= 0 then base is ignored, if `base` > 9 then a `ValueError` exception is raised.

Part III

Options

7 General features

7.1 factor

`--factor <int>`
`factor` takes an `int` value and returns its factors. This feature is a wrapper to Sage's `factor` function. *"Note that Sage's general factor command does nothing but call Pari's factor C library function."* – from doc.sagemath.org.

7.2 ecm

`--ecm <int>`
`ecm` takes an `int` value and returns its factors. This feature is a wrapper to Sage's `ecm.factor` method. ECM factorization is the fastest way to factorize a composite integer if one of its factor is relatively small (25 digits / 80 bit). See doc.sagemath.org for more informations about Sage's `ecm.factor` method. See wikipedia for more informations about ECM factorization method.

7.3 qs

`--qs <int>`
`qs` takes an `int` value (40 or more digits) and returns its factors. This feature is a wrapper to Sage's `qsieve` method. The Quadrartic Sieve factorization is the fastest method for integers which have less than 100 digits and the second fastest method known (the fastest is general number field sieve). See doc.sagemath.org for more informations about Sage's `qsieve` method. See wikipedia for more informations about Quadratic Sieve factorization.

7.4 isprime

`--isprime <int>`
`isprime` takes an `int` value and say if it's prime or not. This feature is a wrapper to Sage's `is_prime` method.

7.5 show-attacks

`--show-attacks`
`show-attacks` shows implemented attacks.

7.6 credits

`--credits`
`credits` shows credits.

7.7 version

`--version`
`version` shows version number and version name.

8 pem

8.1 key

`--key <string>`
`key` takes a `string` which represent a path to a public/private key file.

8.2 n

`-n <int>`
n takes an `int` value which represent RSA public modulus.

8.3 e

`-e <int>`
`e` takes an `int` value which represent RSA public exponent.

8.4 d

`-d <int>`
`d` takes an `int` value which represent RSA private exponent.

8.5 p

`-p <int>`
`p` takes an `int` value which represent the first factor of RSA modulus.

8.6 q

`-q <int>`
`q` takes an `int` value which represent the second factor of RSA modulus.

8.7 output-priv

`-output-priv <string>`
`output-priv` takes a `string` which represent the path of the private key that will be created if `--createpriv` is specified.

8.8 output-pub

`-output-pub <string>`
`output-pub` takes a `string` which represent the path of the public key that will be created if `--createpub` is specified.

8.9 dumpvalues

`--dumpvalues`
`dumpvalues` shows the numeric values of the key specified by `--key`.

8.10 createpub

`--createpub`
`createpub` will build a public key from the values of n and e given by the user. If a path is not provided with `--output-pub` then the public key will be printed to stdout. is possible to specify a format for the key file using `--format`, the choice is between `PEM`, `DER`, `OpenSSH`. If none of the three choices is chosen then the format will be setted to `PEM` by default.

8.11 createpriv

`--createpriv`
`createpriv` will build a private key from the provided numeric values. If a path is not provided with `--output-priv` then the public key will be printed to stdout. is possible to specify a format for the key file using `--format`, the choice is between `PEM`, `DER`, `OpenSSH`. If none of the three choices is chosen then the format will be setted to `PEM` by default. In order to have success

in the creation of a private key the user must provide at least one of the following set of values (/ divide two alternatives):
`n, p/q, e/d`
`p, q, e/d`.

8.12 format

`--format`
`format` takes a `string` which specify the output files format, the choose is between `PEM`, `DER`, `OpenSSH`. If none of the three choices is chosen then the format will be setted to `PEM` by default.

8.13 generate

`--generate`
`generate` specifies will build a new key pair (2048 bits) using a method of pycryptodome RSA object: RSA.generate(). If `e` is specified by the user then the public exponent of the new key will be `e`, otherwise the default value of the public exponent will be 65537. if `--output-pub` is setted then the public key will be saved in the specified path, otherwise it will be printed to stdout. Same for the private key.

9 ciphertool

9.1 cipher

9.1.1 n

`-n <int>`
n takes an `int` value which represent RSA public modulus.

9.1.2 e

`-e <int>`
`e` takes an `int` value which represent RSA public exponent.

9.1.3 key

`--key <string>`
`key` takes a `string` which specify the path for a public/private key that will be used to encrypt the plaintext/plaintext-file. If you don't provide a `key`, then you must provide `n` and `e`.

9.1.4 plaintext

`--plaintext <plaintext>`
`plaintext` takes a `plaintext` which specify the plaintext to encrypt with the provided key. The ciphertext will be printed in decimal, hexadecimal and raw format.

9.1.5 plaintext-file

`--plaintext-file <list>`
`plaintext-file` takes a `list` which specify the path for some files that will be encrypted with the provided key.

9.1.6 output-file

`--output-file <list>`
`output-file` takes a `list` which specify some names for the encrypted files. If there are more files than names, then the name of the encrypted file will be `file_name.enc` by default.

9.1.7 padding

`--padding <str>`
`padding` takes a `string` which represent the type of the padding which will be used to pad the plaintext, you can choose one of the following:
`pkcs7`
`iso7816`
`x923`.

9.1.8 file-padding

`--file-padding <str>`
`file-padding` takes a string which represent the type of the padding which will be used during the encryption process for the plaintext file, you can choose one of the following:
`raw`
`pkcs7`
`ssl`
`oaep`
`x931`
If padding is not provided, then the default value will be pkcs.

9.2 uncipher

9.2.1 n

`-n <int>`
n takes an `int` value which represent RSA public modulus.

9.2.2 e

`-e <int>`
`e` takes an `int` value which represent RSA public exponent.

9.2.3 d

`-d <int>`
`d` takes an `int` value which represent RSA private exponent.

9.2.4 p

`-p <int>`
`p` takes an `int` value which represent RSA first modulus factor.

9.2.5 q

`-q <int>`
`q` takes an `int` value which represent RSA second modulus factor.

9.2.6 key

`--key <string>`
`key` takes a `string` which specify the path for a private key that will be used to decrypt the ciphertext/ciphertext-file. If you don't provide a `key`, then you must provide at least the needed argument to build a private key or the needed arguments to perform the decryption. so one of the following set of values (`/` divide two alternatives):
`n, p/q, e/d`
`p, q, e/d`
`phi, e, n`
`n, d`.

9.2.7 ciphertext

`--plaintext <ciphertext>`
`ciphertext` takes a `ciphertext` which specify the ciphertext to decrypt with the provided key. The plaintext will be printed in decimal, hexadecimal and raw format.

9.2.8 ciphertext-file <list>

`--ciphertext-file <list>`
`ciphertext-file` takes a `list` which specify the path for some files that will be decrypted with the provided key.

9.2.9 padding

`--padding <str>`
`padding` takes a string which represent the type of the padding which will be used during the process in the plaintext, you can choose one of the following:
`pkcs7`
`iso7816`
`x923`.

9.2.10 file-padding

`--file-padding <str>`
`file-padding` takes a string which represent the type of the padding which will be used during the process in the plaintext file, you can choose one of the following:
`raw`
`pkcs7`
`ssl`
`oaep`
`x931`
If padding is not provided, then the default value will be pkcs.

9.2.11 output-file

`--output-file <list>`
`output-file` takes a `list` which specify some names for the decrypted files. If there are more files than names, then by default the name of the decrypted file will be `file_name.dec`.

10 attack

10.1 publickey

`--publickey <str>`
`publickey` takes a `string` which represent a path to a public key file.

10.2 publickeydir

`--publickeydir <str>`
`publickeydir` takes a `string` which represent a path to a directory which contain public key files with extension specified in `ext` (.pem by default).

10.3 attack

`--attack <list>`
`attack` takes a list with the names of the attacks that will be performed on the public key(s), in order to see what attacks are implemented please use –show-attacks flag.

10.4 n

`-n <list>`
`n` takes a list of int with some public key modulus.

10.5 e

`-e <list>`
`e` takes a list of int with some public key exponent.

10.6 n-e-file

`--n-e-file <str>`
`n-e-file` takes a `string` which represent a path to a file on which every line is formatted as follows: `n:e` or `n`.

10.7 ext

`--ext <str>`
`ext` takes a `string` which specify the extension of public keys in the directory specified by `--publickeydir`.

10.8 private

`--private`
`private` will create a private key file if the private key values is recovered, if `output-private` is not specified then the private key file will be prompted to stdout.

10.9 output-private

`--output-private <str>`
`output-private` takes a `string` which represent the path where the recovered private key file will be saved if `--private` flag is setted.

10.10 uncipher

`--uncipher <ciphertext>`
`uncipher` takes a `ciphertext` that will be decrypted if private key values will be recovered.

10.11 uncipher-file

`--uncipher-file <str>`
`uncipher-file` takes a `string` which represent a ciphertext-file that will be decrypted if private key values will be recovered, the decrypted file will be saved in the path specified by `output-file`.

10.12 output-file

`--output-file <str>`
`output-file` takes a `string` which represent the path where the decrypted ciphertext-file specified by `uncipher-file` will be saved.

10.13 output-dir

`--output-dir <str>`
`output-dir` takes a `string` which represent the path where the private key files recovered from the public keys in `publickeydir` will be saved, if `--output-dir` is not specified then the private key files will be printed to stdout.

10.14 timeout

`--timeout <int>`
`timeout` takes an `int` which represent the max attacks running time. if `timeout` is not specified then an attack can run indefinitely.