NAME

botan - Botan command line util

OUTLINE

The **botan** program is a command line tool for using a broad variety of functions of the Botan library in the shell.

All commands follow the syntax **botan <command> <command-options>**.

If **botan** is run with an unknown command, or without any command, or with the **--help** option, all available commands will be printed. If a particular command is run with the **--help** option (like **botan <command> --help**) some information about the usage of the command is printed.

Starting in version 2.9, commands that take a passphrase (such as **gen_bcrypt** or **pkcs8**) will also accept the literal **-** to mean ask for the passphrase on the terminal. If supported by the operating system, echo will be disabled while reading the passphrase.

Most arguments that take a path to a file will also accept the literal **-** to mean the file content should be read from STDIN instead.

HASH FUNCTION

hash --algo=SHA-256 --buf-size=4096 --no-fsname --format=hex *files

Compute the *algo* digest over the data in any number of *files*. If no files are listed on the command line, the input source defaults to standard input. Unless the **--no-fsname** option is given, the filename is printed alongside the hash, in the style of tools such as **sha256sum**.

PASSWORD HASH

gen_argon2 --mem=65536 --p=1 --t=1 password

Calculate the Argon2 password digest of *password*. *mem* is the amount of memory to use in Kb, *p* the parallelization parameter and *t* the number of iterations to use.

check_argon2 password hash

Checks if the Argon2 hash of the passed *password* equals the passed *hash* value.

gen_bcrypt --work-factor=12 password

Calculate the bcrypt password digest of *password*. *work-factor* is an integer between 4 and 18. A higher *work-factor* value results in a more expensive hash calculation.

check_bcrypt password hash
Checks if the bcrypt hash of the passed *password* equals the passed *hash* value.

pbkdf_tune --algo=Scrypt --max-mem=256 --output-len=32 --check *times
Tunes the PBKDF algorithm specified with **--algo=** for the given *times*.

HMAC

hmac --hash=SHA-256 --buf-size=4096 --no-fsname key files
Compute the HMAC tag with the cryptographic hash function *hash* using the key in file *key* over the data in *files*. *files* defaults to STDIN. Unless the **--no-fsname** option is given, the filename is printed alongside the HMAC value.

ENCRYPTION

encryption --buf-size=4096 --decrypt --mode= --key= --iv= --ad=
Encrypt a given file with the specified *mode*. If **--decrypt** is provided the file is decrypted instead.

PUBLIC KEY CRYPTOGRAPHY

keygen --algo=RSA --params= --passphrase= --pbe= --pbe-millis=300 --provider= --der-out
Generate a PKCS #8 *algo* private key. If *der-out* is passed, the pair is BER encoded. Otherwise, PEM encoding is used. To protect the PKCS #8 formatted key, it is recommended to encrypt it with a provided *passphrase*. *pbe* is the name of the desired encryption algorithm, which uses *pbe-millis* milliseconds to derive the encryption key from the passed *passphrase*. Algorithm specific parameters, as the desired bit length of an RSA key, can be passed with *params*.

- For RSA *params* specifies the bit length of the RSA modulus. It defaults to 3072.
- For DH *params* specifies the DH parameters. It defaults to modp/ietf/2048.
- For DSA *params* specifies the DSA parameters. It defaults to dsa/botan/2048.
- For EC algorithms *params* specifies the elliptic curve. It defaults to secp256r1.

  The default *pbe* algorithm is "PBES2(AES-256/CBC,SHA-256)".

  With PBES2 scheme, you can select any CBC or GCM mode cipher which has an OID defined (such as 3DES, Camellia, SM4, Twofish or Serpent). However most other implementations support only AES or 3DES in CBC mode. You can also

choose Scrypt instead of PBKDF2, by using "Scrypt" instead of the name of a hash function, for example "PBES2(AES-256/CBC,Scrypt)". Scrypt is also supported by some other implementations including OpenSSL.

pkcs8 --pass-in= --pub-out --der-out --pass-out= --pbe= --pbe-millis=300 key

Open a PKCS #8 formatted key at *key*. If *key* is encrypted, the passphrase must be passed as *pass-in*. It is possible to (re)encrypt the read key with the passphrase passed as *pass-out*. The parameters *pbe-millis* and *pbe* work similarly to **keygen**.

sign --der-format --passphrase= --hash=SHA-256 --emsa= --provider= key file

Sign the data in *file* using the PKCS #8 private key *key*. If *key* is encrypted, the used passphrase must be passed as *pass-in*. *emsa* specifies the signature scheme and *hash* the cryptographic hash function used in the scheme.

- For RSA signatures EMSA4 (RSA-PSS) is the default scheme.
- For ECDSA and DSA *emsa* defaults to EMSA1 (signing the hash directly)

For ECDSA and DSA, the option **--der-format** outputs the signature as an ASN.1 encoded blob. Some other tools (including **openssl**) default to this format.

The signature is formatted for your screen using base64.

verify --der-format --hash=SHA-256 --emsa= pubkey file signature

Verify the authenticity of the data in *file* with the provided signature *signature* and the public key *pubkey*. Similarly to the signing process, *emsa* specifies the signature scheme and *hash* the cryptographic hash function used in the scheme.

gen_dl_group --pbits=1024 --qbits=0 --seed= --type=subgroup

Generate ANSI X9.42 encoded Diffie-Hellman group parameters.

- If *type=subgroup* is passed, the size of the prime subgroup q is sampled as a prime of *qbits* length and p is *pbits* long. If *qbits* is not passed, its length is estimated from *pbits* as described in RFC 3766.
- If *type=strong* is passed, p is sampled as a safe prime with length *pbits* and the prime subgroup has size q with *pbits*-1 length.
- If *type=dsa* is used, p and q are generated by the algorithm specified in FIPS 186-4. If the **--seed** parameter is used, it allows to select the seed value, instead of one being

randomly generated. If the seed does not in fact generate a valid DSA group, the command will fail.

dl_group_info --pem name

Print raw Diffie-Hellman parameters (p,g) of the standardized DH group *name*. If *pem* is set, the X9.42 encoded group is printed.

ec_group_info --pem name

Print raw elliptic curve domain parameters of the standardized curve *name*. If *pem* is set, the encoded domain is printed.

pk_encrypt --aead=AES-256/GCM rsa_pubkey datafile

Encrypts **datafile** using the specified AEAD algorithm, under a key protected by the specified RSA public key.

pk_decrypt rsa_privkey datafile

Decrypts a file encrypted with **pk_encrypt**. If the key is encrypted using a password, it will be prompted for on the terminal.

fingerprint --no-fsname --algo=SHA-256 *keys

Calculate the public key fingerprint of the *keys*.

pk_workfactor --type=rsa bits

Provide an estimate of the strength of a public key based on it's size. **--type=** can be "rsa", "dl" or "dl_exp".

X.509

gen_pkcs10 key CN --country= --organization= --ca --path-limit=1 --email= --dns= --ext-ku= --key-pass= --hash=SHA-256 --emsa=

Generate a PKCS #10 certificate signing request (CSR) using the passed PKCS #8 private key *key*. If the private key is encrypted, the decryption passphrase *key-pass* has to be passed.*emsa* specifies the padding scheme to be used when calculating the signature.

- For RSA keys EMSA4 (RSA-PSS) is the default scheme.
- For ECDSA, DSA, ECGDSA, ECKCDSA and GOST-34.10 keys *emsa* defaults to EMSA1.

gen_self_signed key CN --country= --dns= --organization= --email= --path-limit=1 --days=365

--key-pass= --ca --hash=SHA-256 --emsa= --der

Generate a self signed X.509 certificate using the PKCS #8 private key *key*. If the private key is encrypted, the decryption passphrase *key-pass* has to be passed. If *ca* is passed, the certificate is marked for certificate authority (CA) usage. *emsa* specifies the padding scheme to be used when calculating the signature.

- For RSA keys EMSA4 (RSA-PSS) is the default scheme.
- For ECDSA, DSA, ECGDSA, ECKCDSA and GOST-34.10 keys *emsa* defaults to EMSA1.

sign_cert --ca-key-pass= --hash=SHA-256 --duration=365 --emsa= ca_cert ca_key pkcs10_req

Create a CA signed X.509 certificate from the information contained in the PKCS #10 CSR *pkcs10_req*. The CA certificate is passed as *ca_cert* and the respective PKCS #8 private key as *ca_key*. If the private key is encrypted, the decryption passphrase *ca-key-pass* has to be passed. The created certificate has a validity period of *duration* days. *emsa* specifies the padding scheme to be used when calculating the signature. *emsa* defaults to the padding scheme used in the CA certificate.

ocsp_check --timeout=3000 subject issuer

Verify an X.509 certificate against the issuers OCSP responder. Pass the certificate to validate as *subject* and the CA certificate as *issuer*.

cert_info --fingerprint file

Parse X.509 PEM certificate and display data fields. If **--fingerprint** is used, the certificate's fingerprint is also printed.

cert_verify subject *ca_certs

Verify if the provided X.509 certificate *subject* can be successfully validated. The list of trusted CA certificates is passed with *ca_certs*, which is a list of one or more certificates.

trust_roots --dn --dn-only --display

List the certificates in the system trust store.

TLS SERVER/CLIENT

The **--policy=** argument of the TLS commands specifies the TLS policy to use. The policy can be any of the the strings "default", "suiteb_128", "suiteb_192", "bsi", "strict", or "all" to denote built-in policies, or it can name a file from which a policy description will be read.

tls_ciphers --policy=default --version=tls1.2

Prints the list of ciphersuites that will be offered under a particular policy/version.

tls_client host --port=443 --print-certs --policy=default --tls1.0 --tls1.1 --tls1.2 --skip-system-cert-store --trusted-cas= --session-db= --session-db-pass= --next-protocols= --type=tcp

Implements a testing TLS client, which connects to *host* via TCP or UDP on port *port*. The TLS version can be set with the flags *tls1.0*, *tls1.1* and *tls1.2* of which the lowest specified version is automatically chosen. If none of the TLS version flags is set, the latest supported version is chosen. The client honors the TLS policy specified with *policy* and prints all certificates in the chain, if *print-certs* is passed. *next-protocols* is a comma separated list and specifies the protocols to advertise with Application-Layer Protocol Negotiation (ALPN).

tls_server cert key --port=443 --type=tcp --policy=default --dump-traces= --max-clients=0 --socket-id=0

Implements a testing TLS server, which allows TLS clients to connect and which echos any data that is sent to it. Binds to either TCP or UDP on port *port*. The server uses the certificate *cert* and the respective PKCS #8 private key *key*. The server honors the TLS policy specified with *policy*. *socket-id* is only available on FreeBSD and sets the *so_user_cookie* value of the used socket.

tls_http_server cert key --port=443 --policy=default --threads=0 --max-clients=0 --session-db --session-db-pass=

Only available if Boost.Asio support was enabled. Provides a simple HTTP server which replies to all requests with an informational text output. The server honors the TLS policy specified with *policy*.

tls_proxy listen_port target_host target_port server_cert server_key--policy=default --threads=0 --max-clients=0 --session-db= --session-db-pass=

Only available if Boost.Asio support was enabled. Listens on a port and forwards all connects to a target server specified at **target_host** and **target_port**.

tls_client_hello --hex input

Parse and print a TLS client hello message.

NUMBER THEORY

is_prime --prob=56 n

Test if the integer *n* is composite or prime with a Miller-Rabin primality test with *(prob+2)/2* iterations.

factor n

Factor the integer *n* using a combination of trial division by small primes, and Pollard’s Rho

algorithm. It can in reasonable time factor integers up to 110 bits or so.

gen_prime --count=1 bits
Samples *count* primes with a length of *bits* bits.

mod_inverse n mod
Calculates a modular inverse.

PSK DATABASE

The PSK database commands are only available if sqlite3 support was compiled in.

psk_set db db_key name psk
Using the PSK database named db and encrypting under the (hex) key **db_key**, save the provided psk (also hex) under **name**:

```
$ botan psk_set psk.db deadba55 bunny f00fee
```

psk_get db db_key name
Get back a value saved with **psk_set**:

```
$ botan psk_get psk.db deadba55 bunny
f00fee
```

psk_list db db_key
List all values saved to the database under the given key:

```
$ botan psk_list psk.db deadba55
bunny
```

SECRET SHARING

Split a file into several shares.

tss_split M N data_file --id= --share-prefix=share --share-suffix=tss --hash=SHA-256
Split a file into **N** pieces any **M** of which suffices to recover the original input. The ID allows specifying a unique key ID which may be up to 16 bytes long, this ensures that shares can be uniquely matched. If not specified a random 16 byte value is used. A checksum can be appended to the data to help verify correct recovery, this can be disabled using **--hash=None**.

tss_recover *shares
Recover some data split by **tss_split**. If insufficient number of shares are provided an error is

printed.

DATA ENCODING/DECODING

base32_dec file
Encode *file* to Base32.

base32_enc file
Decode Base32 encoded *file*.

base58_enc --check file
Encode *file* to Base58. If **--check** is provided Base58Check is used.

base58_dec --check file
Decode Base58 encoded *file*. If **--check** is provided Base58Check is used.

base64_dec file
Encode *file* to Base64.

base64_enc file
Decode Base64 encoded *file*.

hex_dec file
Encode *file* to Hex.

hex_enc file
Decode Hex encoded *file*.

MISCELLANEOUS COMMANDS

version --full
Print the version number. If option **--full** is provided, additional details are printed.

has_command cmd
Test if the command *cmd* is available.

config info_type
Prints build information, useful for applications which want to build against the library. The **info_type** argument can be any of **prefix**, **cflags**, **ldflags**, or **libs**. This is similar to information provided by the **pkg-config** tool.

cpuid

List available processor flags (AES-NI, SIMD extensions, ...).

cpu_clock --test-duration=500

Estimate the speed of the CPU cycle counter.

asn1print --skip-context-specific --print-limit=4096 --bin-limit=2048 --max-depth=64 --pem file‘

Decode and print *file* with ASN.1 Basic Encoding Rules (BER). If flag **--pem** is used, or the filename ends in **.pem**, then PEM encoding is assumed. Otherwise the input is assumed to be binary DER/BER.

http_get --redirects=1 --timeout=3000 url

Retrieve resource from the passed http *url*.

speed --msec=500 --format=default --ecc-groups= --provider= --buf-size=1024 --clear-cpuid= --cpu-clock-speed=0 --cpu-clock-ratio=1.0 *algos

Measures the speed of the passed *algos*. If no *algos* are passed all available speed tests are executed. *msec* (in milliseconds) sets the period of measurement for each algorithm. The *buf-size* option allows testing the same algorithm on one or more input sizes, for example **speed --buf-size=136,1500 AES-128/GCM** tests the performance of GCM for small and large packet sizes. *format* can be "default", "table" or "json".

timing_test test_type --test-data-file= --test-data-dir=src/tests/data/timing --warmup-runs=1000 --measurement-runs=10000

Run various timing side channel tests.

rng --format=hex --system --rdrand --auto --entropy --drbg --drbg-seed= *bytes

Sample *bytes* random bytes from the specified random number generator. If *system* is set, the system RNG is used. If *rdrand* is set, the hardware RDRAND instruction is used. If *auto* is set, AutoSeeded_RNG is used, seeded with the system RNG if available or the global entropy source otherwise. If *entropy* is set, AutoSeeded_RNG is used, seeded with the global entropy source. If *drbg* is set, HMAC_DRBG is used seeded with *drbg-seed*.

entropy --truncate-at=128 source

Sample a raw entropy source.

cc_encrypt CC passphrase --tweak=

Encrypt the passed valid credit card number *CC* using FPE encryption and the passphrase *passphrase*. The key is derived from the passphrase using PBKDF2 with SHA256. Due to the nature of FPE, the ciphertext is also a credit card number with a valid checksum. *tweak* is public

and parameterizes the encryption function.

cc_decrypt CC passphrase --tweak=
Decrypt the passed valid ciphertext *CC* using FPE decryption with the passphrase *passphrase* and the tweak *tweak*.

roughtime_check --raw-time chain-file
Parse and validate a Roughtime chain file.

roughtime --raw-time --chain-file=roughtime-chain --max-chain-size=128 --check-local-clock=60 --host= --pubkey= --servers-file=
Retrieve time from a Roughtime server and store it in a chain file.

uuid
Generate and print a random UUID.

compress --type=gzip --level=6 --buf-size=8192 file
Compress a given file.

decompress --buf-size=8192 file
Decompress a given compressed archive.