passes: *806* failures: *0* duration: *75.84*s 100%

Unit Tests

Crypto

Cipher

- ✓ Blowfish cipher test with test vectors from https://www.schneier.com/code/vectors.txt
- ✓ CAST-128 cipher test with test vectors from RFC2144
- ✓ Twofish with test vectors from https://www.schneier.com/code/ecb_ival.txt

AES Rijndael cipher test with test vectors from ecb_tbl.txt

- ✓ 128 bit key
- ✓ 192 bit key
- ✓ 256 bit key

TripleDES (EDE) cipher test with test vectors from NIST SP 800-20

- ✓ 3DES EDE test vectors
- ✓ DES encrypt/decrypt padding tests

Hash

- ✓ MD5 with test vectors from RFC 1321
- ✓ RIPE-MD 160 bits with test vectors from https://homes.esat.kuleuven.be/~bosselae/ripemd160.html
- ✓ SHA* with test vectors from NIST FIPS 180-2

Random Buffer

- ✓ Throw error if not initialized
- ✓ Initialization
- ✓ Set Method
- ✓ Get Method

API functional testing

Sign and verify

- ✓ RSA
- ✓ DSA

Encrypt and decrypt

- ✓ Symmetric with OpenPGP CFB
- ✓ Asymmetric using RSA with eme_pkcs1 padding
- ✓ Asymmetric using Elgamal with eme_pkcs1 padding 47ms

Elliptic Curve Cryptography @lightweight

Basic Operations

- ✓ Creating curve from name or oid
- ✓ Creating KeyPair 478ms
- ✓ Signature verification
- ✓ Invalid signature
- ✓ Signature generation

ECDSA signature

- ✓ Invalid curve oid
- ✓ Invalid public key
- ✓ Invalid point
- ✓ Invalid signature
- ✓ Valid signature
- ✓ Sign and verify message

ECDH key exchange @lightweight

- ✓ Invalid curve oid

- ✓ Invalid ephemeral key
- ✓ Invalid elliptic public key
- ✓ Invalid key data integrity

passes: *806* failures: *0* duration: *75.84*s

ECDHE key generation

- ✓ Invalid curve
- ✓ Different keys
- ✓ Invalid fingerprint
- ✓ Successful exchange curve25519
- ✓ NIST p256 - Successful exchange
- ✓ NIST p384 - Successful exchange
- ✓ NIST p521 - Successful exchange

Comparing decrypting with and without native crypto

- ✓ NIST p256 60ms
- ✓ NIST p384 153ms
- ✓ NIST p521 300ms

PKCS5 padding

- ✓ Add and remove padding

AES Key Wrap and Unwrap

- ✓ 128 bits of Key Data with a 128-bit KEK
- ✓ 128 bits of Key Data with a 192-bit KEK
- ✓ 128 bits of Key Data with a 256-bit KEK
- ✓ 192 bits of Key Data with a 192-bit KEK
- ✓ 192 bits of Key Data with a 256-bit KEK
- ✓ 256 bits of Key Data with a 256-bit KEK

Symmetric AES-GCM (experimental)

Symmetric AES-GCM (native)

- ✓ aes128
- ✓ aes192
- ✓ aes256

Symmetric AES-GCM (asm.js fallback)

- ✓ aes128
- ✓ aes192
- ✓ aes256

Symmetric AES-GCM (native encrypt, asm.js decrypt)

- ✓ aes128
- ✓ aes192
- ✓ aes256

Symmetric AES-EAX

Symmetric AES-EAX (native)

- ✓ Passes all test vectors

Symmetric AES-EAX (asm.js fallback)

- ✓ Passes all test vectors

Symmetric AES-OCB

- ✓ Passes all test vectors
- ✓ Different key size test vectors 57ms

basic RSA cryptography

- ✓ generate rsa key
- ✓ generate rsa key - without native crypto 425ms
- ✓ sign and verify using generated key params
- ✓ encrypt and decrypt using generated key params

- ◦ decrypt nodeCrypto by bnCrypto and vice versa
- ✓ compare native crypto and bnSign
- ✓ compare native crypto and bnVerify

passes: *806* failures: *0* duration: *75.84*s 100%

EdDSA parameter validation

- ✓ EdDSA params should be valid
- ✓ detect invalid edDSA Q

ECC curve validation

- ✓ EdDSA params are not valid for ECDH
- ✓ EdDSA params are not valid for ECDSA
- ✓ ECDH x25519 params are not valid for ECDSA
- ✓ ECDSA params are not valid for EdDSA
- ✓ ECDH x25519 params are not valid for EdDSA

ECC curve25519 parameter validation

- ◦ ECDSA curve25519 params should be valid
- ◦ ECDSA curve25519 - detect invalid Q
- ✓ ECDH curve25519 params should be valid
- ✓ ECDH curve25519 - detect invalid Q

ECC p256 parameter validation

- ✓ ECDSA p256 params should be valid
- ✓ ECDSA p256 - detect invalid Q
- ✓ ECDH p256 params should be valid 109ms
- ✓ ECDH p256 - detect invalid Q

ECC p384 parameter validation

- ✓ ECDSA p384 params should be valid
- ✓ ECDSA p384 - detect invalid Q
- ✓ ECDH p384 params should be valid 131ms
- ✓ ECDH p384 - detect invalid Q

ECC p521 parameter validation

- ✓ ECDSA p521 params should be valid
- ✓ ECDSA p521 - detect invalid Q 236ms
- ✓ ECDH p521 params should be valid 46ms
- ✓ ECDH p521 - detect invalid Q

ECC secp256k1 parameter validation

- ✓ ECDSA secp256k1 params should be valid
- ✓ ECDSA secp256k1 - detect invalid Q
- ✓ ECDH secp256k1 params should be valid
- ✓ ECDH secp256k1 - detect invalid Q

ECC brainpoolP256r1 parameter validation

- ✓ ECDSA brainpoolP256r1 params should be valid
- ✓ ECDSA brainpoolP256r1 - detect invalid Q
- ✓ ECDH brainpoolP256r1 params should be valid
- ✓ ECDH brainpoolP256r1 - detect invalid Q

ECC brainpoolP384r1 parameter validation

- ✓ ECDSA brainpoolP384r1 params should be valid
- ✓ ECDSA brainpoolP384r1 - detect invalid Q
- ✓ ECDH brainpoolP384r1 params should be valid
- ✓ ECDH brainpoolP384r1 - detect invalid Q

ECC brainpoolP512r1 parameter validation

- ✓ ECDSA brainpoolP512r1 params should be valid 50ms
- ✓ ECDSA brainpoolP512r1 - detect invalid Q
- ✓ ECDH brainpoolP512r1 params should be valid 49ms

- ✓ ECDH brainpoolP512r1 - detect invalid Q

passes: *806* failures: *0* duration: *75.84*s 100%

RSA parameter validation

- ✓ generated RSA params are valid
- ✓ detect invalid RSA n
- ✓ detect invalid RSA e

DSA parameter validation

- ✓ DSA params should be valid
- ✓ detect invalid DSA p
- ✓ detect invalid DSA y
- ✓ detect invalid DSA g

ElGamal parameter validation

- ✓ params should be valid 197ms
- ✓ detect invalid p
- ✓ detect invalid y 187ms
- ✓ detect invalid g 191ms
- ✓ detect g with small order 143ms

General

Util unit tests

isString

- ✓ should return true for type "string"
- ✓ should return true for type String
- ✓ should return true for inherited type of String
- ✓ should return true for empty string
- ✓ should return false for undefined
- ✓ should return false for Object

isArray

- ✓ should return true for []
- ✓ should return true for type Array
- ✓ should return true for inherited type of Array
- ✓ should return false for undefined
- ✓ should return false for Object

isUint8Array

- ✓ should return true for type Uint8Array
- ✓ should return true for inherited type of Uint8Array
- ✓ should return false for undefined
- ✓ should return false for Object

leftPad

- ✓ should not change the input if the length is correct
- ✓ should add leading zeros to input array

uint8ArrayToMPI

- ✓ should strip leading zeros
- ✓ should throw on array of all zeros

isEmailAddress

- ✓ should return true for valid email address
- ✓ should return true for valid email address
- ✓ should return false for invalid email address
- ✓ should return false for invalid email address
- ✓ should return false for invalid email address
- ✓ should return false for empty string
- ✓ should return false for undefined

✓ should return false for Object

constant time select

passes: *806* failures: *0* duration: *75.84*s

100%

✓ selectUint8Array should work for arrays of equal length

✓ selectUint8Array should work for arrays of different length

✓ selectUint8 should return the expected value based on condition

Misc.

✓ util.readNumber should not overflow until full range of uint32

OpenPGP.js webcrypt public api tests

WebCrypt general - unit tests

✓ Status test 1702ms

```
async function () {
      await Webcrypt_Logout$1(statusCallback);
      await Webcrypt_FactoryReset$1(statusCallback);
      const res = await WEBCRYPT_STATUS$1(statusCallback);
      expect$m(res.UNLOCKED).to.be.false;
      expect$m(res).to.have.any.keys('UNLOCKED', 'VERSION', 'ATTEMPTS');
      console.log('Webcrypt status output, including version', {
        res,
        version: new TextDecoder().decode(hexStringToByte$1(res.VERSION_STR
      });
      return true;
    }
```

✓ plugin based key generation 5369ms

```
async function () {
      await plugin.init();
      console.log('test plugin based key generation');
      const { privateKey: lwebcrypt_privateKey, publicKey: lwebcrypt_publick
        curve: 'webcrypt_p256',
        userIDs: [{ name: 'Jon Smith', email: 'jon@example.com' }],
        format: 'object',
        date: plugin.date(),
        plugin: plugin
      });
      console.log({ lwebcrypt_privateKey, lwebcrypt_publicKey });
      webcrypt_privateKey = lwebcrypt_privateKey;
      webcrypt_publicKey = lwebcrypt_publicKey;
      return true;
    }
```

✓ Check cache

```
async function () {
      console.log({ webcrypt_privateKey, webcrypt_publicKey });
      expect$m(webcrypt_privateKey).to.be.ok;
      expect$m(webcrypt_publicKey).to.be.ok;
    }
```

✓ Check generated public key

```
async function () {
      expect$m(webcrypt_publicKey.getFingerprint()).to.be.ok;
      return true;
    }
```

passes: *806* failures: *0* duration: *75.84*s 100%

✓ Encrypting and decrypting message against Webcrypt key 878ms

```
async function () {
      const plaintext = 'Hello, World!';
      const encrypted = await openpgp$8.encrypt({
        message: await openpgp$8.createMessage({ text: plaintext }),
        encryptionKeys: webcrypt_publicKey,
        format: 'binary'
      });
      expect$m(encrypted).to.be.a('Uint8Array');
      const message = await openpgp$8.readMessage({
        binaryMessage: encrypted
      });
      const { data: decrypted, signatures } = await openpgp$8.decrypt({
        message,
        decryptionKeys: webcrypt_privateKey,
        plugin: plugin
      });
      expect$m(decrypted).to.be.equal(plaintext);
      return true;
    }
```

✓ Signing message 872ms

```
async function () {
      const message = await openpgp$8.createMessage({ text: 'Hello, World!'
      const detachedSignature = await openpgp$8.sign({
        message,
        signingKeys: webcrypt_privateKey,
        plugin: plugin,
        detached: true
      });
      console.log({ detachedSignature });
      expect$m(detachedSignature).to.be.ok;
      const signature = await openpgp$8.readSignature({
        armoredSignature: detachedSignature
      });
      const verificationResult = await openpgp$8.verify({
        message,
        signature,
        verificationKeys: webcrypt_publicKey
      });
      const { verified, keyID } = verificationResult.signatures[0];
      await verified; // throws on invalid signature
      console.log('webcrypt detached signature, signed by key id ' + keyID.
      expect$m(keyID.toHex()).to.be.equal(webcrypt_publicKey.keyPacket.keyID
    }
```

✓ Signing message webcrypt non-detached 866ms

passes: *806* failures: *0* duration: *75.84*s 100%

```
async function () {
      const unsignedMessage = await openpgp$8.createCleartextMessage({ text
      const cleartextMessage = await openpgp$8.sign({
        message: unsignedMessage, // CleartextMessage or Message object
        signingKeys: webcrypt_privateKey,
        plugin: plugin
      });
      // console.log('after signing', { cleartextMessage }); // '-----BEGIN
      expect$m(cleartextMessage).to.be.ok;
      expect$m(cleartextMessage).to.have.string('BEGIN PGP SIGNED MESSAGE')
      const signedMessage = await openpgp$8.readCleartextMessage({
        cleartextMessage // parse armored message
      });
      const verificationResult = await openpgp$8.verify({
        message: signedMessage,
        verificationKeys: webcrypt_publicKey
      });
      const { verified, keyID } = verificationResult.signatures[0];
      await verified; // throws on invalid signature
      expect$m(keyID.toHex()).to.be.equal(webcrypt_publicKey.keyPacket.keyI
    }
```

✓ Signing big message webcrypt non-detached 5073ms

```
async function () {
      const clearText = 'Hello, World!'.padEnd(980, '='); // 980, 900, 500 \
      const unsignedMessage = await openpgp$8.createCleartextMessage({ text
      const cleartextMessage = await openpgp$8.sign({
        message: unsignedMessage,
        signingKeys: webcrypt_privateKey,
        plugin: plugin
      });
      const signedMessage = await openpgp$8.readCleartextMessage({
        cleartextMessage // parse armored message
      });
      const verificationResult = await openpgp$8.verify({
        message: signedMessage,
        verificationKeys: webcrypt_publicKey
      });
      const { verified, keyID } = verificationResult.signatures[0];
      await verified; // throws on invalid signature
      expect$m(keyID.toHex()).to.be.equal(webcrypt_publicKey.keyPacket.keyI
    }
```

✓ OpenPGPjs key import to WebCrypt 2281ms

```
async function () {
      // software key
      const { privateKey, publicKey } = await openpgp$8.generateKey({
        curve: 'p256',
        userIDs: [{ name: 'Jon Smith', email: 'jon@example.com' }],
```

```
        format: 'object'
      });
      await WEBCRYPT_OPENPGP_IMPORT$1(statusCallback, {
        sign_privkey: privateKey.keyPacket.privateParams.d,
        encr_privkey: privateKey.subkeys[0].keyPacket.privateParams.d,
        date: privateKey.getCreationTime()
      });
      const webcrypt_openpgp_keys_current = await WEBCRYPT_OPENPGP_INFO$1(s
      console.log({ sw: publicKey.keyPacket.publicParams.Q, wc: webcrypt_ope
      expect$m(publicKey.keyPacket.publicParams.Q, 'Main key public key chec
      expect$m(publicKey.subkeys[0].keyPacket.publicParams.Q, 'Subkey public
    }
```

passes: *806* failures: *0* duration: *75.84*s 100%

✓ OpenPGPjs imported key import to WebCrypt 4275ms

```
async function () {
      // reset plugin cached info
      await plugin.init();
      const { privateKey: lwebcrypt_privateKey, publicKey: lwebcrypt_publicK
        curve: 'webcrypt_p256',
        userIDs: [{ name: 'Jon Smith', email: 'jon@example.com' }],
        format: 'object',
        date: plugin.date(),
        plugin: plugin
      });
      console.log({ lwebcrypt_privateKey, lwebcrypt_publicKey });
      webcrypt_privateKey = lwebcrypt_privateKey;
      webcrypt_publicKey = lwebcrypt_publicKey;
    }
```

✓ WebCrypt OpenPGP factory reset 2632ms

```
async function () {
      const webcrypt_openpgp_keys_before = await WEBCRYPT_OPENPGP_INFO$1(sta
      await WEBCRYPT_OPENPGP_GENERATE$1(statusCallback);
      const webcrypt_openpgp_keys_current = await WEBCRYPT_OPENPGP_INFO$1(s
      console.log('Current webcrypt keys and the regenerated key after WEBCR
        webcrypt_openpgp_keys_before,
        webcrypt_openpgp_keys_current
      });
      expect$m(webcrypt_openpgp_keys_before).to.be.not.equal(webcrypt_openpg
      await plugin.init();
    }
```

BigInteger interface

- ✓ constructor throws on undefined input
- ✓ constructor supports strings
- ✓ constructor supports Uint8Arrays
- ✓ conditional operators are correct
- ✓ bitLength is correct
- ✓ byteLength is correct
- ✓ toUint8Array is correct
- ✓ binary operators are consistent

- ✓ unary operators are consistent
- ✓ modExp is correct (large values)
- ✓ gcd is correct
- ✓ modular inversion is correct
- ✓ getBit is correct

passes: *806* failures: *0* duration: *75.84*s 100%

ASCII armor

- ✓ Parse cleartext signed message
- ✓ Exception if mismatch in armor header and signature
- ✓ Exception if no header and non-MD5 signature
- ✓ Exception if unknown hash algorithm
- ✓ Multiple hash values
- ✓ Multiple hash header lines
- ✓ Non-hash header line throws exception
- ✓ Multiple wrong hash values
- ✓ Multiple wrong hash values
- ✓ Filter whitespace in blank line
- ✓ Exception if improperly formatted armor header - plaintext section
- ✓ Exception if improperly formatted armor header - signature section
- ✓ Exception if improperly formatted armor footer
- ✓ Ignore unknown armor header - signature section
- ✓ Exception if wrong armor header type
- ✓ Armor checksum validation - mismatch
- ✓ Armor checksum validation - valid
- ✓ Armor checksum validation - missing
- ✓ Armor checksum validation - missing - trailing newline
- ✓ Accept header with trailing whitespace
- ✓ Do not filter blank lines after header
- ✓ Do not add extraneous blank line when base64 ends on line break

Packet

- ✓ Symmetrically encrypted packet without integrity protection - allow decryption
- ✓ Symmetrically encrypted packet without integrity protection - disallow decryption by default
- ✓ Sym. encrypted integrity protected packet
- ✓ Sym. encrypted AEAD protected packet
- ✓ Sym. encrypted AEAD protected packet is encrypted in parallel (AEAD, GCM)
- ✓ Sym. encrypted AEAD protected packet test vector (AEAD)
- ✓ Sym. encrypted session key with a compressed packet
- ✓ Public key encrypted symmetric key packet
- ✓ Secret key packet (reading, unencrypted)
- ✓ Public key encrypted packet (reading, GPG)
- ✓ Sym. encrypted session key reading/writing (CFB)
- ✓ Sym. encrypted session key reading/writing (AEAD)
- ✓ Sym. encrypted session key reading/writing test vector (EAX, AEAD)
- ✓ Sym. encrypted session key reading/writing test vector (AEAD, OCB)
- ✓ Secret key encryption/decryption test
- ✓ Secret key reading with signature verification.
- ✓ Reading a signed, encrypted message.
- ✓ Reading signersUserID from armored signature
- ✓ Reading notations from armored key
- ✓ Writing and encryption of a secret key packet (AEAD)
- ✓ Writing of unencrypted v5 secret key packet
- ✓ Writing and encryption of a secret key packet (CFB)
- ✓ Writing and verification of a signature packet

PacketList parsing

- ✓ Ignores unknown packet version with `config.ignoreUnsupportedPackets` enabled
- ✓ Throws on unknown packet version with `config.ignoreUnsupportedPackets` disabled
- ✓ Ignores unknown signature algorithm only with `config.ignoreUnsupportedPackets` enabled
- ✓ Ignores unknown key algorithm only with `config.ignoreUnsupportedPackets` enabled
- ✓ Ignores unknown PKESK algorithm only with `config.ignoreUnsupportedPackets` enabled
- ✓ Throws on disallowed packet even with tolerant mode enabled
- ✓ Throws on parsing errors `config.ignoreMalformedPackets` disabled

passes: *806* failures: *0* duration: *75.84*s

Signature

- ✓ Retrieve the issuer Key ID of a signature
- ✓ Throws when reading a signature missing the creation time
- ✓ Ignores marker packets when verifying signatures
- ✓ Testing signature checking on CAST5-enciphered message
- ✓ Consider signature expired at the expiration time
- ✓ Signing fails if primary key is expired
- ✓ Signing fails if the signing date is before the key creation date
- ✓ Verification fails if primary key binding signature is expired
- ✓ Verification fails if signing key's self-sig were created after the time of signing, unless config allows it
- ✓ Verification fails if signing key was already expired at the time of signing (one-pass signature, streamed)
- ✓ Verification fails if signing key was already expired at the time of signing (standard signature)
- ✓ Verification succeeds if an expired signing key was valid at the time of signing (with streaming)
- ✓ Verification succeeds if an expired signing key was valid at the time of signing (without streaming)
- ✓ Supports non-human-readable notations
- ✓ Verify V4 signature. Hash: SHA1. PK: RSA. Signature Type: 0x00 (binary document)
- ✓ Verify signature of signed and encrypted message from GPG2 with openpgp.decrypt
- ✓ Verify signed message with two one pass signatures
- ✓ Verify fails with signed message with critical notations
- ✓ Verify succeeds with known signed message with critical notations
- ✓ Verify cleartext signed message with two signatures with openpgp.verify
- ✓ Verify latin-1 signed message
- ✓ Verify cleartext signed message with trailing spaces from GPG
- ✓ Verify signed message with trailing spaces from GPG
- ✓ Streaming verify signed message with trailing spaces from GPG 1708ms
- ✓ Verify signed message with missing signature packet
- ✓ Streaming verify signed message with missing signature packet 1002ms
- ✓ Sign text with openpgp.sign and verify with openpgp.verify leads to same string cleartext and valid signatures
- ✓ Sign text with openpgp.sign and verify with openpgp.verify leads to same string cleartext and valid signatures -- escape armored message 38ms
- ✓ Sign text with openpgp.sign and verify with openpgp.verify leads to same string cleartext and valid signatures -- trailing spaces
- ✓ Sign text with openpgp.sign and verify with openpgp.verify leads to same bytes cleartext and valid signatures - armored
- ✓ Sign text with openpgp.sign and verify with openpgp.verify leads to same bytes cleartext and valid signatures - not armored
- ✓ Should verify cleartext message correctly when using a detached cleartext signature and binary literal data
- ✓ Should verify cleartext message correctly when using a detached binary signature and text literal data
- ✓ Should verify encrypted cleartext message correctly when encrypting binary literal data with a canonical text signature
- ✓ Verify primary key revocation signatures
- ✓ Verify subkey revocation signatures
- ✓ Verify key expiration date
- ✓ Write unhashed subpackets
- ✓ Write V4 signatures

- ✓ Verify a detached signature using appendSignature
- ✓ Detached signature signing and verification 57ms
- ✓ Sign message with key without password
- ✓ Verify signed key
- ✓ Verify signed UserIDs and User Attributes
- ✓ should verify a shorter RSA signature
- ✓ should verify a shorter EdDSA signature
- ✓ should verify a shorter ECDSA signature

passes: *806* failures: *0* duration: *75.84*s

100%

Accept SHA-1 signatures

- ✓ Verify signed message with trailing spaces from GPG
- ✓ Streaming verify signed message with trailing spaces from GPG 1701ms
- ✓ Verify signed message with missing signature packet
- ✓ Streaming verify signed message with missing signature packet 1003ms

Key

- ✓ Parsing armored text with RSA key and ECC subkey
- ✓ Parsing armored text with two keys
- ✓ Parsing armored key with an authorized revocation key in a User ID self-signature
- ✓ Parsing armored key with an authorized revocation key in a direct-key signature
- ✓ Parsing V5 public key packet
- ✓ Testing key ID and fingerprint for V4 keys
- ✓ Create new key ID with fromID()
- ✓ Testing key method getSubkeys
- ✓ Verify status of revoked primary key
- ✓ Verify status of revoked subkey
- ✓ Verify status of key with non-self revocation signature
- ✓ Verify primary key with authorized revocation key in a direct-key signature
- ✓ Verify certificate of key with future creation date
- ✓ Evaluate key flags to find valid encryption key packet
- ✓ should pad an ECDSA P-521 key with shorter secret key
- ✓ should not decrypt using a sign-only RSA key, unless explicitly configured
- ✓ Key.getExpirationTime()
- ✓ Key.getExpirationTime() - expired key
- ✓ SubKey.getExpirationTime()
- ✓ Key.getExpirationTime() - never expiring key
- ✓ Key.getExpirationTime() - key expiration in direct-key signature
- ✓ decryptKey() - throw if key parameters don't correspond
- ✓ validate() - don't throw if key parameters correspond
- ✓ validate() - throw if all-gnu-dummy key
- ✓ validate() - gnu-dummy primary key with signing subkey
- ✓ validate() - gnu-dummy primary key with encryption subkey 220ms
- ✓ validate() - curve ed25519 (eddsa) cannot be used for ecdsa
- ✓ isDecrypted() - should reflect whether all (sub)keys are encrypted
- ✓ isDecrypted() - gnu-dummy primary key
- ✓ isDecrypted() - all-gnu-dummy key
- ✓ makeDummy() - the converted key can be parsed
- ✓ makeDummy() - the converted key can be encrypted and decrypted
- ✓ makeDummy() - the converted key is valid but can no longer sign
- ✓ makeDummy() - subkeys of the converted key can still sign
- ✓ makeDummy() - should work for encrypted keys
- ✓ clearPrivateParams() - check that private key can no longer be used
- ✓ clearPrivateParams() - detect that private key parameters were removed
- ✓ clearPrivateParams() - detect that private key parameters were zeroed out
- ✓ update() - throw error if fingerprints not equal

passes: *806* failures: *0* duration: *75.84*s 100%

- ✓ update() - merge revocation signatures
- ✓ update() - merge user
- ✓ update() - merge user - other and certification revocation signatures
- ✓ update() - merge subkey
- ✓ update() - merge subkey - revocation signature
- ✓ update() - merge private key into public key
- ✓ update() - merge private key into public key - no subkeys
- ✓ update() - merge private key into public key - mismatch throws error
- ✓ update() - merge subkey binding signatures
- ✓ update() - merge multiple subkey binding signatures
- ✓ revoke() - primary key
- ✓ revoke() - subkey
- ✓ applyRevocationCertificate() should produce the same revoked key as GnuPG
- ✓ getRevocationCertificate() should produce the same revocation certificate as GnuPG
- ✓ getRevocationCertificate() should have an appropriate comment
- ✓ getPreferredAlgo('symmetric') - one key
- ✓ getPreferredAlgo('symmetric') - two key
- ✓ getPreferredAlgo('symmetric') - two key - one without pref
- ✓ getPreferredAlgo('aead') - one key - OCB
- ✓ getPreferredAlgo('aead') - two key - one without pref
- ✓ getPreferredAlgo('aead') - two key - one with no support
- ✓ User attribute packet read & write
- ✓ getPrimaryUser()
- ✓ getPrimaryUser() should throw if no UserIDs are bound
- ✓ Generate session key - latest created user
- ✓ Generate session key - primary user
- ✓ Generate session key - specific user
- ✓ Fails to encrypt to User ID-less key
- ✓ Sign - specific user 41ms
- ✓ Find a valid subkey binding signature among many invalid ones
- ✓ Selects the most recent subkey binding signature
- ✓ Selects the most recent non-expired subkey binding signature
- ✓ Selects the most recent valid subkey binding signature
- ✓ Handles a key with no valid subkey binding signatures gracefully
- ✓ Reject encryption with revoked primary user
- ✓ Reject encryption with revoked subkey
- ✓ Reject encryption with key revoked with appended revocation cert
- ✓ Merge key with another key with non-ID user attributes
- ✓ Should throw when trying to encrypt a key that's already encrypted
- ✓ Subkey.verify returns the latest valid signature

V4

- ✓ Preferences of generated key
- ✓ Preferences of generated key - with config values
- ✓ Generated key is not unlocked by default
- ✓ Generate key - single userID
- ✓ Generate key - single userID (all empty)
- ✓ Generate key - single userID (empty email)
- ✓ Generate key - single userID (empty comment)
- ✓ Generate key - setting date to the past
- ✓ Generate key - setting date to the future
- ✓ Generate key - multi userID
- ✓ Generate key - default values
- ✓ Generate key - two subkeys with default values
- ✓ Generate RSA key - two subkeys with default values

passes: *806* failures: *0* duration: *75.84*s

100%

- ✓ Generate key - one signing subkey
- ✓ Reformat key - one signing subkey 73ms
- ✓ Generate key - override main RSA key options for subkey
- ✓ Generate key - ensure keyExpirationTime works
- ✓ Sign and verify key - primary user
- ✓ Sign key and verify with wrong key - primary user
- ✓ Sign and verify key - all users
- ✓ Sign key and verify with wrong key - all users
- ✓ Reformat and encrypt key with no subkey
- ✓ Reformat key with one subkey 39ms
- ✓ Reformat key with no subkey
- ✓ Reformat and encrypt key
- ✓ Sign and encrypt with reformatted key
- ✓ Reject with user-friendly error when reformatting encrypted key
- ✓ Revoke generated key with revocation certificate
- ✓ Revoke generated key with private key
- ✓ Revoke reformatted key with revocation certificate
- ✓ Parses V5 sample key

V5

- ✓ Preferences of generated key
- ✓ Preferences of generated key - with config values
- ✓ Generated key is not unlocked by default
- ✓ Generate key - single userID
- ✓ Generate key - single userID (all empty)
- ✓ Generate key - single userID (empty email)
- ✓ Generate key - single userID (empty comment)
- ✓ Generate key - setting date to the past
- ✓ Generate key - setting date to the future
- ✓ Generate key - multi userID
- ✓ Generate key - default values
- ✓ Generate key - two subkeys with default values
- ✓ Generate RSA key - two subkeys with default values
- ✓ Generate key - one signing subkey
- ✓ Reformat key - one signing subkey 63ms
- ✓ Generate key - override main RSA key options for subkey
- ✓ Generate key - ensure keyExpirationTime works
- ✓ Sign and verify key - primary user 40ms
- ✓ Sign key and verify with wrong key - primary user
- ✓ Sign and verify key - all users
- ✓ Sign key and verify with wrong key - all users
- ✓ Reformat and encrypt key with no subkey
- ✓ Reformat key with one subkey
- ✓ Reformat key with no subkey
- ✓ Reformat and encrypt key
- ✓ Sign and encrypt with reformatted key 46ms
- ✓ Reject with user-friendly error when reformatting encrypted key
- ✓ Revoke generated key with revocation certificate
- ✓ Revoke generated key with private key
- ✓ Revoke reformatted key with revocation certificate
- ✓ Parses V5 sample key

addSubkey functionality testing

- ✓ create and add a new rsa subkey to stored rsa key 41ms
- ✓ Add a new default subkey to an rsaSign key

passes: *806* failures: *0* duration: *75.84*s 100%

- ✓ Add a new default subkey to an ecc key
- ✓ Add a new default subkey to a dsa key 57ms
- ✓ should throw when trying to add a new default subkey to an ecc key that uses a blacklisted curve (brainpool)
- ✓ should throw when trying to encrypt a subkey separately from key
- ✓ encrypt and decrypt key with added subkey 40ms
- ✓ create and add a new eddsa subkey to a eddsa key 46ms
- ✓ create and add a new ecdsa subkey to a eddsa key
- ✓ create and add a new ecc subkey to a rsa key
- ✓ create and add a new rsa subkey to a ecc key 223ms
- ✓ create and add a new rsa subkey to a dsa key 49ms
- ✓ sign/verify data with the new subkey correctly using curve25519 57ms
- ✓ encrypt/decrypt data with the new subkey correctly using curve25519
- ✓ sign/verify data with the new subkey correctly using rsa 43ms
- ✓ encrypt/decrypt data with the new subkey correctly using rsa 47ms

OpenPGP.js public api tests

readKey(s) and readPrivateKey(s) - unit tests

- ✓ readKey and readPrivateKey should create equal private keys
- ✓ readPrivateKeys and readKeys should create equal private keys
- ✓ readPrivateKey should throw on armored public key
- ✓ readPrivateKeys should throw on armored public keys

generateKey - validate user ids

- ✓ should fail for invalid user name
- ✓ should fail for invalid user email address
- ✓ should fail for invalid user email address
- ✓ should fail for string user ID
- ✓ should work for valid single user ID object
- ✓ should work for array of user ID objects
- ✓ should work for undefined name
- ✓ should work for an undefined email address

generateKey - unit tests

- ✓ should have default params set
- ✓ should output keypair with expected format 50ms

reformatKey - unit tests

- ✓ should output keypair with expected format 44ms

revokeKey - unit tests

- ✓ should output key with expected format

decryptKey - unit tests

- ✓ should work for correct passphrase
- ✓ should work with multiple passphrases 39ms
- ✓ should fail for incorrect passphrase
- ✓ should fail for corrupted key

encryptKey - unit tests

- ✓ should not change original key
- ✓ encrypted key can be decrypted
- ✓ should throw on empty passphrase
- ✓ should support multiple passphrases
- ✓ should encrypt gnu-dummy key

decrypt - unit tests

- ✓ Calling decrypt with encrypted key leads to exception
- ✓ decrypt/verify should succeed with valid signature (expectSigned=true)

- ✓ decrypt/verify should throw on missing public keys (expectSigned=true) 43ms
- ✓ decrypt/verify should throw on missing signature (expectSigned=true)
- ✓ decrypt/verify should throw on invalid signature (expectSigned=true) 41ms
- ✓ decrypt/verify should succeed with valid signature (expectSigned=true, with streaming)
- ✓ decrypt/verify should throw on missing public keys (expectSigned=true, with streaming) 42ms
- ✓ decrypt/verify should throw on missing signature (expectSigned=true, with streaming)
- ✓ decrypt/verify should throw on invalid signature (expectSigned=true, with streaming)
- ✓ Supports decrypting with GnuPG dummy key 145ms
- ✓ decrypt with `config.constantTimePKCS1Decryption` option should succeed 62ms
- ✓ decrypt with `config.constantTimePKCS1Decryption` option should succeed (with streaming) 68ms
- ✓ decrypt with `config.constantTimePKCS1Decryption` option should fail if session key algo support is disabled 51ms

verify - unit tests

message

- ✓ verify should succeed with valid signature (expectSigned=true) 43ms
- ✓ verify should throw on missing signature (expectSigned=true)
- ✓ verify should throw on invalid signature (expectSigned=true) 39ms
- ✓ verify should succeed with valid signature (expectSigned=true, with streaming)
- ✓ verify should throw on missing signature (expectSigned=true, with streaming)
- ✓ verify should throw on invalid signature (expectSigned=true, with streaming)
- ✓ verify should fail if the signature is re-used with a different message

cleartext message

- ✓ verify should succeed with valid signature (expectSigned=true)
- ✓ verify should throw on missing signature (expectSigned=true)
- ✓ verify should throw on invalid signature (expectSigned=true)

sign - unit tests

- ✓ Supports signing with GnuPG dummy key
- ✓ Calling sign with no signing key leads to exception
- ✓ should output cleartext message of expected format
- ✓ should output message of expected format
- ✓ should output message of expected format
- ✓ should output message of expected format (with streaming) 66ms
- ✓ should output message of expected format (detached)
- ✓ should output message of expected format (detached, with streaming) 55ms

encrypt - unit tests

- ✓ Does not encrypt to expired key (expiration time subpacket on a direct-key signature)
- ✓ should output message of expected format
- ✓ should output message of expected format (with streaming) 46ms

encryptSessionKey - unit tests

- ✓ should output message of expected format
- ✓ passing no encryption keys or passwords leads to exception

encrypt, decrypt, sign, verify - integration tests

- ✓ Configuration
- ✓ should fail to decrypt a message containing a literal packet (and no session key)
- ✓ should fail to decrypt a message containing a literal packet (and a session key)
- ✓ should fail to decrypt non-integrity-protected message by default
- ✓ should allow decrypting non-integrity-protected message when enabled
- ✓ should allow stream-decrypting non-integrity-protected message when enabled

CFB mode (asm.js)

encryptSessionKey, decryptSessionKeys

- ✓ should encrypt with public key

passes: *806* failures: *0* duration: *75.84*s 100%

- ✓ should encrypt with password
- ✓ should not decrypt with a key without binding signatures
- ✓ roundtrip workflow: encrypt, decryptSessionKeys, decrypt with pgp key pair
- ✓ roundtrip workflow: encrypt, decryptSessionKeys, decrypt with pgp key pair -- trailing spaces
- ✓ roundtrip workflow: encrypt, decryptSessionKeys, decrypt with password
- ✓ roundtrip workflow: encrypt with multiple passwords, decryptSessionKeys, decrypt with multiple passwords
- ✓ roundtrip workflow: encrypt twice with one password, decryptSessionKeys, only one session key

AES / RSA encrypt, decrypt, sign, verify

- ✓ should encrypt then decrypt
- ✓ should encrypt then decrypt with multiple private keys 43ms
- ✓ should encrypt then decrypt with wildcard
- ✓ should encrypt then decrypt with wildcard with multiple private keys 51ms
- ✓ should encrypt then decrypt using returned session key
- ✓ should encrypt using custom session key and decrypt using session key
- ✓ should encrypt using custom session key and decrypt using private key
- ✓ should encrypt/sign and decrypt/verify
- ✓ should encrypt/sign and decrypt/verify (expectSigned=true)
- ✓ should encrypt/sign and decrypt/verify (no AEAD support)
- ✓ should encrypt/sign and decrypt/verify with generated key 53ms
- ✓ should encrypt/sign and decrypt/verify with generated key and detached signatures 59ms
- ✓ should encrypt/sign and decrypt/verify with null string input
- ✓ should encrypt/sign and decrypt/verify with detached signatures
- ✓ should encrypt and decrypt/verify with detached signature as input for encryption 76ms
- ✓ should fail to encrypt and decrypt/verify with detached signature as input for encryption with wrong public key
- ✓ should fail to verify decrypted data with wrong public pgp key
- ✓ should fail to verify decrypted null string with wrong public pgp key
- ✓ should successfully decrypt signed message without public keys to verify
- ✓ should fail to verify decrypted data with wrong public pgp key with detached signatures
- ✓ should encrypt and decrypt/verify both signatures when signed with two private keys 77ms
- ✓ should fail to decrypt modified message 210ms
- ✓ should fail to decrypt unarmored message with garbage data appended

ELG / DSA encrypt, decrypt, sign, verify

- ✓ round trip test 129ms

3DES decrypt

- ✓ Decrypt message

AES encrypt, decrypt

- ✓ should encrypt and decrypt with one password
- ✓ should encrypt and decrypt with two passwords
- ✓ should encrypt and decrypt with password and not ascii armor
- ✓ should encrypt and decrypt with binary data

Encrypt, decrypt with compression

compression - uncompressed

- ✓ should encrypt and decrypt with one password
- ✓ Streaming encrypt and decrypt small message roundtrip

compression - zip

- ✓ should encrypt and decrypt with one password
- ✓ Streaming encrypt and decrypt small message roundtrip

compression - zlib

- ✓ should encrypt and decrypt with one password

- ✓ Streaming encrypt and decrypt small message roundtrip

GCM mode (V5 keys)

passes: *806* failures: *0* duration: *75.84*s 100%

encryptSessionKey, decryptSessionKeys

- ✓ should encrypt with public key
- ✓ should encrypt with password
- ✓ should not decrypt with a key without binding signatures
- ✓ roundtrip workflow: encrypt, decryptSessionKeys, decrypt with pgp key pair
- ✓ roundtrip workflow: encrypt, decryptSessionKeys, decrypt with pgp key pair -- trailing spaces
- ✓ roundtrip workflow: encrypt, decryptSessionKeys, decrypt with password
- ✓ roundtrip workflow: encrypt with multiple passwords, decryptSessionKeys, decrypt with multiple passwords
- ✓ roundtrip workflow: encrypt twice with one password, decryptSessionKeys, only one session key

AES / RSA encrypt, decrypt, sign, verify

- ✓ should encrypt then decrypt
- ✓ should encrypt then decrypt with multiple private keys 45ms
- ✓ should encrypt then decrypt with wildcard
- ✓ should encrypt then decrypt with wildcard with multiple private keys 50ms
- ✓ should encrypt then decrypt using returned session key
- ✓ should encrypt using custom session key and decrypt using session key
- ✓ should encrypt using custom session key and decrypt using private key
- ✓ should encrypt/sign and decrypt/verify
- ✓ should encrypt/sign and decrypt/verify (expectSigned=true)
- ✓ should encrypt/sign and decrypt/verify (no AEAD support)
- ✓ should encrypt/sign and decrypt/verify with generated key 62ms
- ✓ should encrypt/sign and decrypt/verify with generated key and detached signatures 54ms
- ✓ should encrypt/sign and decrypt/verify with null string input
- ✓ should encrypt/sign and decrypt/verify with detached signatures
- ✓ should encrypt and decrypt/verify with detached signature as input for encryption 66ms
- ✓ should fail to encrypt and decrypt/verify with detached signature as input for encryption with wrong public key
- ✓ should fail to verify decrypted data with wrong public pgp key
- ✓ should fail to verify decrypted null string with wrong public pgp key
- ✓ should successfully decrypt signed message without public keys to verify
- ✓ should fail to verify decrypted data with wrong public pgp key with detached signatures
- ✓ should encrypt and decrypt/verify both signatures when signed with two private keys 75ms
- ✓ should fail to decrypt modified message 195ms
- ✓ should fail to decrypt unarmored message with garbage data appended

ELG / DSA encrypt, decrypt, sign, verify

- ✓ round trip test 123ms

3DES decrypt

- ✓ Decrypt message

AES encrypt, decrypt

- ✓ should encrypt and decrypt with one password
- ✓ should encrypt and decrypt with two passwords
- ✓ should encrypt and decrypt with password and not ascii armor
- ✓ should encrypt and decrypt with binary data

Encrypt, decrypt with compression

compression - uncompressed

- ✓ should encrypt and decrypt with one password
- ✓ Streaming encrypt and decrypt small message roundtrip

compression - zip

✓ should encrypt and decrypt with one password
✓ Streaming encrypt and decrypt small message roundtrip

passes: *806* failures: *0* duration: *75.84*s 100%

compression - zlib

✓ should encrypt and decrypt with one password
✓ Streaming encrypt and decrypt small message roundtrip

EAX mode (small chunk size)

encryptSessionKey, decryptSessionKeys

✓ should encrypt with public key
✓ should encrypt with password
✓ should not decrypt with a key without binding signatures
✓ roundtrip workflow: encrypt, decryptSessionKeys, decrypt with pgp key pair
✓ roundtrip workflow: encrypt, decryptSessionKeys, decrypt with pgp key pair -- trailing spaces
✓ roundtrip workflow: encrypt, decryptSessionKeys, decrypt with password
✓ roundtrip workflow: encrypt with multiple passwords, decryptSessionKeys, decrypt with multiple passwords
✓ roundtrip workflow: encrypt twice with one password, decryptSessionKeys, only one session key

AES / RSA encrypt, decrypt, sign, verify

✓ should encrypt then decrypt
✓ should encrypt then decrypt with multiple private keys 43ms
✓ should encrypt then decrypt with wildcard
✓ should encrypt then decrypt with wildcard with multiple private keys 54ms
✓ should encrypt then decrypt using returned session key
✓ should encrypt using custom session key and decrypt using session key
✓ should encrypt using custom session key and decrypt using private key
✓ should encrypt/sign and decrypt/verify
✓ should encrypt/sign and decrypt/verify (expectSigned=true)
✓ should encrypt/sign and decrypt/verify (no AEAD support)
✓ should encrypt/sign and decrypt/verify with generated key 61ms
✓ should encrypt/sign and decrypt/verify with generated key and detached signatures 60ms
✓ should encrypt/sign and decrypt/verify with null string input
✓ should encrypt/sign and decrypt/verify with detached signatures
✓ should encrypt and decrypt/verify with detached signature as input for encryption 68ms
✓ should fail to encrypt and decrypt/verify with detached signature as input for encryption with wrong public key
✓ should fail to verify decrypted data with wrong public pgp key
✓ should fail to verify decrypted null string with wrong public pgp key
✓ should successfully decrypt signed message without public keys to verify
✓ should fail to verify decrypted data with wrong public pgp key with detached signatures
✓ should encrypt and decrypt/verify both signatures when signed with two private keys 79ms
✓ should fail to decrypt modified message 215ms
✓ should fail to decrypt unarmored message with garbage data appended

ELG / DSA encrypt, decrypt, sign, verify

✓ round trip test 126ms

3DES decrypt

✓ Decrypt message

AES encrypt, decrypt

✓ should encrypt and decrypt with one password
✓ should encrypt and decrypt with two passwords
✓ should encrypt and decrypt with password and not ascii armor
✓ should encrypt and decrypt with binary data

Encrypt, decrypt with compression

compression - uncompressed

- ✓ should encrypt and decrypt with one password passes: *806* failures: *0* duration: *75.84*s
- ✓ Streaming encrypt and decrypt small message roundtrip

compression - zip

- ✓ should encrypt and decrypt with one password
- ✓ Streaming encrypt and decrypt small message roundtrip

compression - zlib

- ✓ should encrypt and decrypt with one password
- ✓ Streaming encrypt and decrypt small message roundtrip

OCB mode

encryptSessionKey, decryptSessionKeys

- ✓ should encrypt with public key
- ✓ should encrypt with password
- ✓ should not decrypt with a key without binding signatures
- ✓ roundtrip workflow: encrypt, decryptSessionKeys, decrypt with pgp key pair
- ✓ roundtrip workflow: encrypt, decryptSessionKeys, decrypt with pgp key pair -- trailing spaces
- ✓ roundtrip workflow: encrypt, decryptSessionKeys, decrypt with password
- ✓ roundtrip workflow: encrypt with multiple passwords, decryptSessionKeys, decrypt with multiple passwords
- ✓ roundtrip workflow: encrypt twice with one password, decryptSessionKeys, only one session key

AES / RSA encrypt, decrypt, sign, verify

- ✓ should encrypt then decrypt
- ✓ should encrypt then decrypt with multiple private keys 47ms
- ✓ should encrypt then decrypt with wildcard
- ✓ should encrypt then decrypt with wildcard with multiple private keys 54ms
- ✓ should encrypt then decrypt using returned session key
- ✓ should encrypt using custom session key and decrypt using session key
- ✓ should encrypt using custom session key and decrypt using private key
- ✓ should encrypt/sign and decrypt/verify
- ✓ should encrypt/sign and decrypt/verify (expectSigned=true)
- ✓ should encrypt/sign and decrypt/verify (no AEAD support)
- ✓ should encrypt/sign and decrypt/verify with generated key 53ms
- ✓ should encrypt/sign and decrypt/verify with generated key and detached signatures 52ms
- ✓ should encrypt/sign and decrypt/verify with null string input
- ✓ should encrypt/sign and decrypt/verify with detached signatures
- ✓ should encrypt and decrypt/verify with detached signature as input for encryption 64ms
- ✓ should fail to encrypt and decrypt/verify with detached signature as input for encryption with wrong public key
- ✓ should fail to verify decrypted data with wrong public pgp key
- ✓ should fail to verify decrypted null string with wrong public pgp key
- ✓ should successfully decrypt signed message without public keys to verify
- ✓ should fail to verify decrypted data with wrong public pgp key with detached signatures
- ✓ should encrypt and decrypt/verify both signatures when signed with two private keys 79ms
- ✓ should fail to decrypt modified message 193ms
- ✓ should fail to decrypt unarmored message with garbage data appended

ELG / DSA encrypt, decrypt, sign, verify

- ✓ round trip test 126ms

3DES decrypt

- ✓ Decrypt message

AES encrypt, decrypt

- ✓ should encrypt and decrypt with one password

- ✓ should encrypt and decrypt with two passwords
- ✓ should encrypt and decrypt with password and not ascii armor
- ✓ should encrypt and decrypt with binary data

passes: *806* failures: *0* duration: *75.84*s 100%

Encrypt, decrypt with compression

compression - uncompressed

- ✓ should encrypt and decrypt with one password
- ✓ Streaming encrypt and decrypt small message roundtrip

compression - zip

- ✓ should encrypt and decrypt with one password
- ✓ Streaming encrypt and decrypt small message roundtrip

compression - zlib

- ✓ should encrypt and decrypt with one password
- ✓ Streaming encrypt and decrypt small message roundtrip

AES / RSA encrypt, decrypt, sign, verify

- ✓ should sign and verify cleartext message
- ✓ should sign and verify cleartext message with multiple private keys 54ms
- ✓ should sign and verify data with detached signatures
- ✓ should sign and fail to verify cleartext message with wrong public pgp key
- ✓ should sign and fail to verify data with wrong public pgp key with detached signature
- ✓ should sign and verify data and not armor
- ✓ should sign and verify data and not armor with detached signatures
- ✓ should sign and verify data with a date in the past
- ✓ should sign and verify binary data with a date in the future
- ✓ should sign and verify binary data without one-pass signature
- ✓ should streaming sign and verify binary data without one-pass signature
- ✓ should encrypt and decrypt data with a date in the future
- ✓ should encrypt and decrypt binary data with a date in the past
- ✓ should sign, encrypt and decrypt, verify data with a date in the past
- ✓ should sign, encrypt and decrypt, verify binary data with a date in the future
- ✓ should sign, encrypt and decrypt, verify mime data with a date in the future
- ✓ should fail to encrypt with revoked key
- ✓ should fail to encrypt with revoked subkey 46ms
- ✓ should decrypt with revoked subkey 102ms
- ✓ should not decrypt with corrupted subkey 94ms
- ✓ RSA decryption with PKCS1 padding of wrong length should fail
- ✓ should decrypt with two passwords message which GPG fails on
- ✓ should decrypt with three passwords
- ✓ should decrypt broken ECC message from old OpenPGP.js
- ✓ should decrypt broken ECC message from old go crypto
- ✓ should decrypt Blowfish message
- ✓ should normalize newlines in encrypted text message

Sign and verify with each curve

- ✓ sign/verify with secp256k1
- ✓ sign/verify with p256
- ✓ sign/verify with p384
- ✓ sign/verify with p521
- ✓ sign/verify with curve25519
- ✓ sign/verify with brainpoolP256r1
- ✓ sign/verify with brainpoolP384r1 69ms
- ✓ sign/verify with brainpoolP512r1 141ms

Errors

- ✓ Error message should contain the original error message

passes: *806* failures: *0* duration: *75.84*s 100%

Specific encryption/signing key testing

- ✓ Encrypt message with a specific encryption key id
- ✓ Sign message with a specific signing key id
- ✓ Encrypt and sign with specific encryption/signing key ids

Custom configuration

- ✓ openpgp.readMessage
- ✓ openpgp.readSignature
- ✓ openpgp.readKey
- ✓ openpgp.readKeys
- ✓ openpgp.generateKey
- ✓ openpgp.reformatKey 40ms
- ✓ openpgp.revokeKey
- ✓ openpgp.decryptKey
- ✓ openpgp.encryptKey 56ms
- ✓ openpgp.encrypt
- ✓ openpgp.decrypt 157ms
- ✓ openpgp.sign
- ✓ openpgp.verify 67ms

detects unknown config property

- ✓ openpgp.generateKey
- ✓ openpgp.encryptKey
- ✓ openpgp.decryptKey
- ✓ openpgp.reformatKey
- ✓ openpgp.revokeKey
- ✓ openpgp.sign
- ✓ openpgp.encrypt
- ✓ openpgp.verify
- ✓ openpgp.decrypt
- ✓ openpgp.generateSessionKey
- ✓ openpgp.encryptSessionKey
- ✓ openpgp.decryptSessionKeys

Oid tests

- ✓ Constructing
- ✓ Reading and writing

Elliptic Curve Cryptography for NIST P-256,P-384,P-521 curves @lightweight

- ✓ Omnibus NIST P-256 Test
- ✓ Sign message
- ✓ Encrypt and sign message

Elliptic Curve Cryptography for secp256k1 curve @lightweight

- ✓ Load public key
- ✓ Load private key
- ✓ Verify clear signed message
- ✓ Sign message
- ✓ Decrypt and verify message
- ✓ Encrypt and sign message
- ✓ Generate key

X25519 Cryptography

- ✓ Load public key
- ✓ Load private key 156ms
- ✓ Verify clear signed message
- ✓ Sign message
- ✓ Decrypt and verify message

- ✓ Encrypt and sign message

Ed25519 Test Vectors from RFC8032

passes: *806* failures: *0* duration: *75.84*s 100%

- ✓ Signature of empty string
- ✓ Signature of single byte
- ✓ Signature of two bytes
- ✓ Signature of 1023 bytes
- ✓ Signature of SHA(abc)

X25519 Omnibus Tests

- ✓ Omnibus Ed25519/Curve25519 Test 63ms

Brainpool Cryptography @lightweight

- ✓ Load public key
- ✓ Load private key 129ms
- ✓ Verify clear signed message
- ✓ Sign message 171ms
- ✓ Decrypt and verify message 49ms
- ✓ Decrypt and verify message with leading zero in hash 70ms
- ✓ Decrypt and verify message with leading zero in hash signed with old elliptic algorithm 70ms
- ✓ Encrypt and sign message 77ms

Brainpool Omnibus Tests @lightweight

- ✓ Omnibus BrainpoolP256r1 Test 67ms

Decrypt and decompress message tests

- ✓ Decrypts message compressed with zip 64ms
- ✓ Decrypts message compressed with zlib 70ms
- ✓ Decrypts message compressed with bzip2 64ms

Streaming

- ✓ Encrypt small message
- ✓ Encrypt larger message 444ms
- ✓ Input stream should be canceled when canceling encrypted stream
- ✓ Sign: Input stream should be canceled when canceling encrypted stream
- ✓ Encrypt and decrypt larger message roundtrip 3456ms
- ✓ Encrypt and decrypt larger message roundtrip (allowUnauthenticatedStream=true) 444ms
- ✓ Encrypt and decrypt larger message roundtrip using public keys (allowUnauthenticatedStream=true) 464ms
- ✓ Encrypt and decrypt larger message roundtrip using curve x25519 (allowUnauthenticatedStream=true) 538ms
- ✓ Encrypt and decrypt larger message roundtrip using curve brainpool (allowUnauthenticatedStream=true) 752ms
- ✓ Detect MDC modifications (allowUnauthenticatedStream=true) 437ms
- ✓ Detect armor checksum error (allowUnauthenticatedStream=true) 469ms
- ✓ Detect armor checksum error when not passing public keys (allowUnauthenticatedStream=true) 463ms
- ✓ Sign/verify: Detect armor checksum error 451ms
- ✓ stream.transformPair()
- ✓ Sign/verify: Input stream should be canceled when canceling verified stream
- ✓ Don't pull entire input stream when we're not pulling encrypted stream 3005ms
- ✓ Sign: Don't pull entire input stream when we're not pulling signed stream 3015ms
- ✓ Sign/verify: Don't pull entire input stream when we're not pulling verified stream 3017ms
- ✓ Detached sign small message
- ✓ Detached sign small message using brainpool curve keys 257ms
- ✓ Detached sign small message using x25519 curve keys 113ms
- ✓ Detached sign is expected to pull entire input stream when we're not pulling signed stream 3002ms
- ✓ Detached sign: Input stream should be canceled when canceling signed stream

AEAD

- ✓ Encrypt and decrypt larger message roundtrip (AEAD) 445ms
- ✓ Encrypt and decrypt larger text message roundtrip (AEAD) 119ms
- ✓ Don't pull entire input stream when we're not pulling decrypted stream (AEAD) 3046ms
- ✓ Input stream should be canceled when canceling decrypted stream (AEAD)

passes: *806* failures: *0* duration: *75.84*s

100%

Web Worker

Application Worker

- ✓ Should support loading OpenPGP.js from inside a Web Worker 184ms

Security

- ✓ Does not accept non-binary/text signatures
- ✓ Does not accept unsigned subpackets
- ✓ Does not trust subkeys without Primary Key Binding Signature 330ms
- ✓ Does not accept message encrypted with algo not mentioned in preferred algorithms