



Developer Documentation

GPG4Browsers

An OpenPGP Implementation in JavaScript

Table of Contents

1	Introduction.....	5
2	Architectural Overview.....	6
2.1	Background Page.....	6
2.2	OpenPGP Tab.....	8
2.3	Content Script.....	8
3	Current Limitations and Licensing.....	9
3.1	Contributions.....	9
4	API Reference.....	11
4.1	Main API.....	11
4.1.1	The OpenPGP singleton.....	11
4.1.2	Configuration.....	13
4.1.3	Keyring API.....	14
4.2	Working with OpenPGP objects.....	17
4.2.1	OpenPGP Public Key Object.....	17
4.2.2	OpenPGP Private Key Object.....	19
4.2.3	OpenPGP Message Object.....	19
4.3	OpenPGP Types.....	20
4.3.1	Multi-Precision Integers.....	20
4.3.2	String-to-Key Specifiers.....	22
4.3.3	Key Id.....	22
4.4	OpenPGP Packet Types.....	23
4.4.1	Encrypted Session Key Packet (tag 1, tag 3).....	25
4.4.2	Signature Packet (tag 2).....	26
4.4.3	One-Pass Signature Packet (tag 4).....	27
4.4.4	Key Material Packet (tag 5, tag 6, tag 7, tag 14).....	28
4.4.5	Compressed Data Packet (tag 8).....	30
4.4.6	Symmetrically Encrypted Data Packet (tag 9).....	31
4.4.7	Marker Packet (tag 10).....	32
4.4.8	Literal Data Packet (tag 11).....	32
4.4.9	User ID Packet (tag 13).....	32
4.4.10	User Attribute Packet (tag 17).....	34
4.4.11	Symmetrically Encrypted Integrity Protected Data Packet (tag 18).....	35
4.4.12	Modification Detection Code Packet (tag 19).....	36
4.5	Cryptography.....	36
4.5.1	OpenPGP Cipher Feedback Mode (CFB).....	36
4.5.2	OpenPGP Cryptography API.....	38
4.6	Encoding.....	44
4.6.1	ASCII-Armor and Base64.....	44
4.6.2	PKCS1 Encoding.....	45
4.7	Utility functions.....	47
4.7.1	Printing Messages.....	49
5	Appendix.....	50



5.1 File Reference.....50

Terms and Definitions

Term	Definition
AES	Advanced Encryption Standard
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BSD	Berkeley Software Distribution
CRC	Cyclic Redundancy Check
DES	Data Encryption Standard
DSA	Digital Signature Algorithm
GNU	GNU's Not Unix
GPL	GNU Public License
HTML	Hypertext Markup Language
IV	Initialization Vector
MD5	Message-Digest Algorithm 5
RFC	Request For Comments
RSA	Asymmetric Cryptosystem named after Rivest, Shamir and Adleman
SHA	Secure Hash Algorithm
URL	Uniform Resource Locator

1 Introduction

The OpenPGP JavaScript implementation can be used to make OpenPGP available to Web-based Mail-applications. Especially in environments where code execution or programs on the operating system level cannot be executed this implementation can be used.

The implementation currently implements a Chrome Browser Extension which uses the HTML5 local storage of the extension to store private and public certificates (keys). Chrome Browser Extensions enabling JavaScript applications to run in their own context separated from the Web-site. This is a basic security feature this implementation uses. Since memory-wipe of private data and validation of a secure execution environment cannot be achieved in JavaScript this implementation should not be used in environments where the confidentiality and integrity of the transmitted data is important.

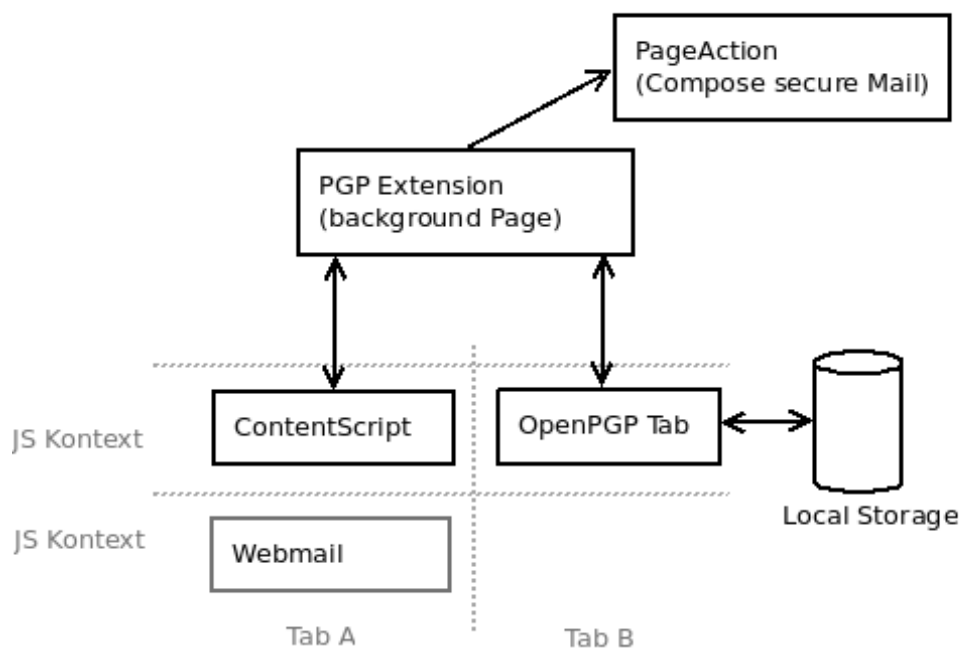
The implementation is in a prototype state.

2 Architectural Overview

The implementation consists of three main components:

- Background Page JavaScript code running in the background exchanging messages between a content script and the extension tab as well as closes and opens tabs.
- Content Script JavaScript code running within the Web-Mail tab or the window which is retrieving the data from the messages allowing them to decrypt / verify signatures using this extension.
- OpenPGP Tab HTML page to perform OpenPGP operations such as encrypt/decrypt data or create/verify signatures.

All three components are running in their own JavaScript context and are exchanging messages through the background page using the `chrome.extension` API.



Drawing 1: Component model and communication

2.1 Background Page

The background page is used to open tabs and retrieve requests from other pages.

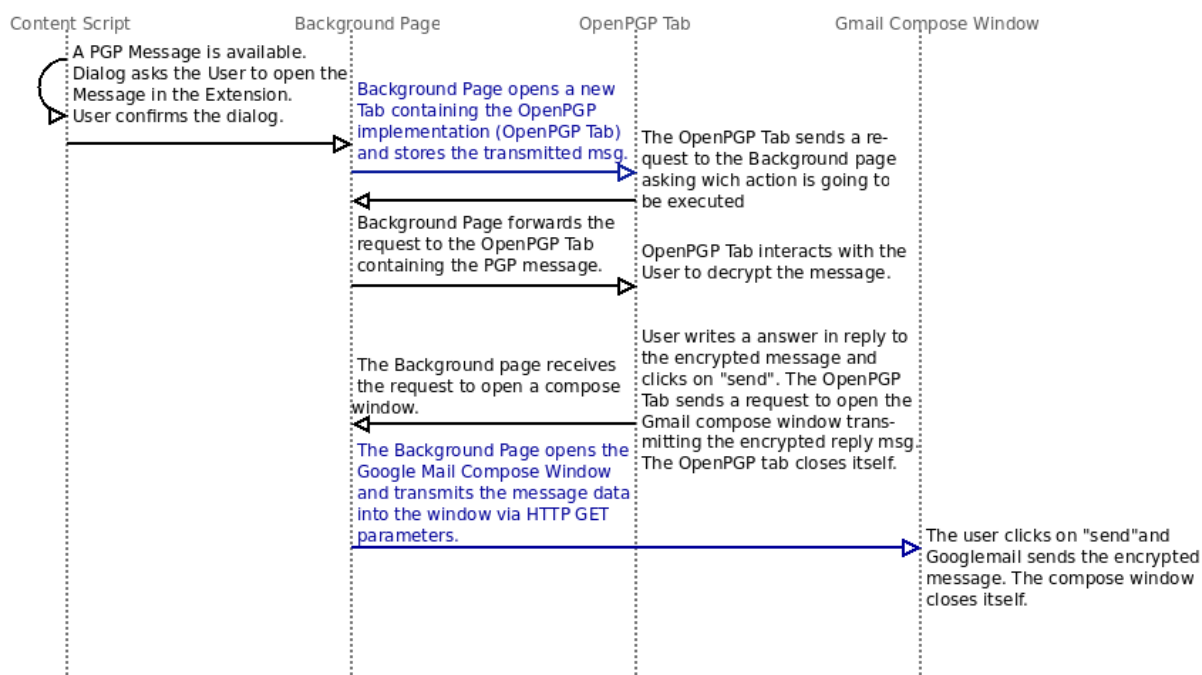
The page registers a request listener and verifies that the calling script URL is within the extension. Every event object has an "action" attribute defining what should be done with this request:

"action" attribute: expected behavior:

- 0** Page Action requests to open the OpenPGP Tab to compose a message
- 1** ContentScript requests to open the OpenPGP Tab to decrypt / verify a message.
- 2** OpenPGP Tab requests to open a Goglemail Mail Compose tab.

Since data must be shared between the Content Script and OpenPGP Tab the background page stores the request sent to open a tab. For each request the tab is sending the "TabId" along with the request. The opened OpenPGP Tab creates a request on load to the background page. If data must be transmitted the background page sends the received request from the Content Script to the OpenPGP Tab.

As an example, the following flowchart shows message exchanged when the user is decrypting a message and encrypting a reply:



Drawing 2: Flowchart of decrypting a message and composing a reply

2.2 OpenPGP Tab

This component holds the OpenPGP implementation and allows to perform OpenPGP operations on messages.

As described above the OpenPGP Tab sends a request on load to the background page to find out which task will be performed (either decrypting or verifying a message: request contains a message; or shows the compose window: request is null).

The following code gives an example on how the OpenPGP implementation can be used to encrypt a message:

```
var privatekey = "-----BEGIN PGP PUBLIC KEY BLOCK-----  
    [...]  
    -----END PGP PUBLIC KEY BLOCK-----";  
var publickey = "-----BEGIN PGP PRIVATE KEY BLOCK-----  
    [...]  
    -----END PGP PRIVATE KEY BLOCK-----";  
  
openpgp.init();  
  
var privatekey_obj = openpgp.read_privateKey(privatekey)[0];  
var publickeys_obj = openpgp.read_publicKey(publickey);  
  
var encryptedmessage = "";  
  
if (privatekey_obj.decryptSecretMPIs("Passwort")) {  
    encryptedmessage = openpgp.write_signed_and_encrypted_message(  
        privatekey_obj,  
        publickeys_obj, "geheimer Text");  
  
    privatekey.lockSecretMPIs();  
}
```

2.3 Content Script

The Content Script implements the interaction with the Web-Mail application and is responsible for:

- reading the message out of the Gmail Web-interface
- showing the user that this message can be opened with the OpenPGP extension
- getting the users data (such as email address)

The interaction with Gmail is poorly implemented and lacks some features. Since Gmail does not provide an API to retrieve message in JavaScript the message parsing is done through retrieving the displayed message in the basic HTML interface and reading the displayed data using X-Path addressing. The implementation for this part is implemented in `gmail.js`.

3 Current Limitations and Licensing

This implementation covers functionality for encryption and signature generation of message in the first place. An implementation of the following functionality is currently missing:

- public / private key generation
- proper validation of certification and signature binding
- compression of messages (all algorithms)

Since the implementation is in a prototype state some validation checks are missing.

3.1 Contributions

Several open source cipher algorithm implementations have been used to create this prototype:

Component	Author	License
AES library	Herbert Hanewinkel	FREE
CAST5 library	pjacobs@xeekr.com with modifications from Herbert Hanewinkel	BSD License
DES library	Paul Tero with modifications from Michael Hayworth	FREE
Blowfish library	nklein software (Patrick Fleckenstein)	FREE
Twofish library	Atsushi Oka	LGPL
SHA library	Brian Turek	BSD License
MD5 library	Henri Torgemane	FREE
RIPEMD/160	Derek Buitenhuis	GNU GPLv2 or later
Base64 encoding library	Herbert Hanewinkel	FREE
JS BigInt library	Tom Wu (tjw@cs.Stanford.EDU)	FREE
OpenPGP CFB	Partially from Herbert Hanewinkel	FREE
JQuery library	See www.jquery.com	MIT or GNU GPL v2





4 API Reference

The following section describes all function in detail. Italic text is copied from RFC 4880 (<http://tools.ietf.org/html/rfc4880.html>) to give a general overview what the implementation is supposed to do here.

4.1 Main API

On load there are three singleton objects for the main usage. To initialize the OpenPGP implementation `openpgp.init()` must be called. This function initializes the configuration parameters and the keyring.

The `openpgp` object holds the main functionality of this implementation as described below.

4.1.1 The OpenPGP singleton

The `openpgp` object serves the following functionality and is implemented in `js/openpgp.js`:

Class	<code>openpgp</code>		
Function	<code>init()</code>		
Description	Initializes the configuration object (<code>openpgp.config</code>) and the keyring.		
Parameters	none		
Return Value	<code>null</code>	<code>undefined</code>	

Class	<code>openpgp_keyring</code>		
Function	<code>read_publicKey(armoredText)</code>		
Description	Reads several <code>publicKey</code> objects from a ASCII armored representation and returns <code>openpgp_msg_publickey</code> packets.		
Parameters	<code>armoredText</code>	<code>String</code>	ASCII armored openpgp public key block message containing at least one key
Return Value	<code>Array[openpgp_msg_publickey]</code>	An array of public key objects parsed; Otherwise an empty array.	

Class	<code>openpgp_keyring</code>		
Function	<code>read_privateKey(armoredText)</code>		
Description	Reads several <code>privateKey</code> objects from a ASCII armored		

	representation and returns openpgp_msg_private packets.		
Parameters	armoredText	String	ASCII armored OpenPGP private key block message containing at least one key
Return Value	Array[openpgp_msg_privatekey]	An array of private key objects parsed. Otherwise an empty array.	

Class	openpgp_keyring		
Function	read_message (armoredText)		
Description	Reads several message objects from a ASCII armored representation an returns openpgp_msg_message packets.		
Parameters	armoredText	String	ASCII armored OpenPGP message block (encrypted and/or signed) message containing at least one message.
Return Value	Array[openpgp_msg_privatekey]	An array of message objects parsed. Otherwise an empty array.	

Class	openpgp_keyring		
Function	write_signed_and_encrypted_message (privatekey, publickeys, messagetext)		
Description	Creates a binary string representation of an encrypted and signed message. The message will be encrypted with the public keys specified and signed with the specified private key.		
Parameters	privateKey	openpgp_msg_privateKey	private key to be used to sign the message
	publicKeys	Array[openpgp_msg_privateKey]	public keys to be used to encrypt the message
	armoredText	String	message text to encrypt and sign
Return Value	[String]	ASCII armored encrypted and signed OpenPGP message block	

Class	openpgp_keyring		
Function	write_encrypted_message (publickeys, messagetext)		
Description	Creates a binary string representation of an encrypted message. The message will be encrypted with the public keys specified.		

Parameters	publicKeys	Array[openpgp_msg_privateKey]	Public keys to be used to encrypt the message
	armoredText	String	Message text to encrypt and sign
Return Value	String	ASCII armored encrypted OpenPGP message block	

Class	openpgp_keyring		
Function	write_signed_message(privatekey, messagetext)		
Description	Creates a binary string representation of a signed message. The message will be signed with the specified private key.		
Parameters	privateKey	openpgp_msg_privateKey	Private key to be used to sign the message
	armoredText	String	Message text to encrypt and sign
Return Value	String	ASCII armored signed OpenPGP message block	

4.1.2 Configuration

This object contains configuration values and implements storing and retrieving configuration from HTML5 local storage. It can be found in `js/openpgp.config.js`.

After calling `openpgp.init()` the configuration object can be accessed using `openpgp.config`. Stored configuration parameters can be accessed using `openpgp.config.config`.

The class contains several predefined attributes:

- The default configuration is written if no configuration was stored before:

```

this.default_config = {
    prefer_hash_algorithm: 2,
    encryption_cipher: 9,
    compression: 1,
    show_version: true,
    integrity_protect: true,
    composition_behavior: 0,
    keyserver: "keyserver.linux.it"
};

```

- The debug switch (if set to "true", debug message will be printed using `util.print_debug();`)

```

this.debug = false;

```

- The version string of GPG4Browsers:

```

this.versionstring = "GPG4Browsers 0.1";

```

The `openpgp_config` class contains the following functions:

Class	<code>openpgp_config</code>	
Function	<code>read()</code>	
Description	Reads the store configuration object out of the HTML5 local storage and initializes the object "config". If "config" is null the default configuration object will be used.	
Parameters	none	
Return Value	<code>null</code>	<code>undefined</code>

Class	<code>openpgp_config</code>	
Function	<code>write()</code>	
Description	Writes the current configuration state to HTML5 local storage.	
Parameters	none	
Return Value	<code>null</code>	<code>undefined</code>

4.1.3 Keyring API

The keyring contains all saved public and private certificates. It is initialized and stored in the background to the HTML5 local storage of the extension. The implementation can be found in `js/openpgp.keyring.js` and has the following functions:

Class	<code>openpgp_keyring</code>	
Function	<code>init()</code>	
Description	Initialization routine for the keyring. This method reads the keyring from the HTML5 local storage and initializes this instance. This method is called by <code>openpgp.init()</code> .	
Parameters	none	
Return Value	<code>null</code>	<code>undefined</code>

Class	<code>openpgp_keyring</code>	
Function	<code>hasPrivateKey()</code>	
Description	Checks if at least one private key is in the keyring.	
Parameters	none	
Return Value	<code>Boolean</code>	"true" if at least one private key is stored in the keyring; "false" otherwise.

Class	openpgp_keyring	
Function	store()	
Description	Saves the current state of the keyring to the HTML5 local storage. The privateKeys array and publicKey array gets "stringified" using JSON.	
Parameters	none	
Return Value	null	undefined

Class	openpgp_keyring	
Function	getPublicKeyForAddress(email_address)	
Description	Searches all public keys in the keyring matching the address or address part of the user ids.	
Parameters	data	String eMail address to search for
Return Value	Array[openpgp_msg_publickey]	An array of public key objects found. Otherwise an empty array.

Class	openpgp_keyring	
Function	getPrivateKeyForAddress(email_address)	
Description	Searches all private keys in the keyring matching the address or address part of the user ids.	
Parameters	data	String eMail address to search for
Return Value	Array[openpgp_msg_privatekey]	An array of private key objects found. Otherwise an empty array.

Class	openpgp_keyring	
Function	getPublicKeysForKeyId(keyId)	
Description	Searches the keyring for public keys having the specified key id.	
Parameters	keyId	String 8 bytes as a string containing the key id to look for.
Return Value	Array[openpgp_msg_publickey]	An array of public key objects found. Otherwise an empty array.

Class	openpgp_keyring	
Function	getPrivateKeysForKeyId(keyId)	

Description	Searches the keyring for private keys having the specified key id.		
Parameters	keyId	String	8 bytes as a string containing the key id to look for.
Return Value	Array[openpgp_msg_privatekey]	An array of private key objects found. Otherwise an empty array.	

Class	openpgp_keyring		
Function	importPublicKey(armored_text)		
Description	Imports a public key from an exported ASCII armored message.		
Parameters	armored_text	String	PUBLIC KEY BLOCK message to read the public key from.
Return Value	null	undefined	

Class	openpgp_keyring		
Function	importPrivateKey(armored_text)		
Description	Imports a private key from an exported ASCII armored message.		
Parameters	armored_text	String	PRIVATE KEY BLOCK message to read the private key from.
Return Value	null	undefined	

Class	openpgp_keyring		
Function	exportPublicKey(index)		
Description	Returns the PUBLIC KEY BLOCK message representation of the public key at public key ring index.		
Parameters	index	Integer	Index of the public key within the publicKeys array.
Return Value	String	The PUBLIC KEY BLOCK message.	

Class	openpgp_keyring		
Function	exportPrivateKey(index)		
Description	Returns the PRIVATE KEY BLOCK message representation of the private key at private key ring index .		
Parameters	index	Integer	Index of the private key within the privateKeys array.

Return Value	String	The PRIVATE KEY BLOCK message.
---------------------	--------	--------------------------------

Class	openpgp_keyring		
Function	removePublicKey(index)		
Description	Removes a public key from the public key keyring at the specified index .		
Parameters	index	Integer	Index of the private key within the publicKeys array.
Return Value	openpgp_msg_publickey	The public key object which has been removed.	

4.2 Working with OpenPGP objects

4.2.1 OpenPGP Public Key Object

The public key object is either retrieved from the keyring or from the operation `openpgp.read_publicKey(text)`. An object instance can be used to validate signatures or encrypt messages (see Main API). The class also provides the following functions:

Class	openpgp_msg_publickey		
Function	getSigningKey()		
Description	Gets the public key material suitable for verifying signatures.		
Parameters	none		
Return Value	openpgp_packet_keymaterial	public key material if the public key is able to validate signatures; otherwise null.	

Class	openpgp_msg_publickey		
Function	getEncryptionKey()		
Description	Gets the public key material suitable for encrypting messages.		
Parameters	none		
Return Value	openpgp_packet_keymaterial	Public key material if the public key is able to encrypt messages; otherwise null.	

Class	openpgp_msg_publickey		
Function	verifyBasicSignatures()		

Description	verifies: <ul style="list-style-type: none"> • revocation certificates directly on key • self signatures • subkey binding and revocation certificates 	
Parameters	none	
Return Value	Boolean	"true" if the basic signatures are all valid.

Class	openpgp_msg_publickey	
Function	verifyCertificationSignatures()	
Description	Verifies all certification signatures on UserIds within the public key. Returns a 2 dimensional array where the first dimension corresponds to the index of the "userIds" attribute index and the second to the result value of openpgp_packet_userid.verifyCertificationSignatures(). Result values for the second array are: 0 = bad signature 1 = signature expired 2 = issuer key not available 3 = revoked 4 = signature valid 5 = signature by key owner expired 6 = signature by key owner revoked	
Parameters	none	
Return Value	Array[Array[Integer]]	A 2 dimensional array. The first dimension corresponds to the UserIds available.

Class	openpgp_msg_publickey	
Function	validate()	
Description	Validates the following properties of the public key (incomplete): <ul style="list-style-type: none"> • check if there are no valid key revocation signatures • search for at least one valid subkey • search for one valid UserId (not expired and valid, signed by owner) 	
Parameters	none	
Return Value	Boolean	Returns "true" if the check was successful; otherwise "false".

The class wraps the following functions of the key material packet: `getKeyId()`,

`getFingerprint()`.

4.2.2 OpenPGP Private Key Object

The private key object is either retrieved from the keyring or from the operation `openpgp.read_privateKey(text)`.

Class	<code>openpgp_msg_privatekey</code>	
Function	<code>getSigningKey()</code>	
Description	Gets the private key material suitable for creating signatures.	
Parameters	none	
Return Value	<code>openpgp_packet_keymaterial</code>	Private key material if the private key has a key for creating signatures; otherwise null.

Class	<code>openpgp_msg_privatekey</code>	
Function	<code>getPreferredSignatureHashAlgorithm()</code>	
Description	Returns the preferred Hash Algorithm (on DSA signatures only a subset of Hash Algorithms is allowed)	
Parameters	none	
Return Value	Integer	Hash Algorithm to use.

The class wraps several functions of the key material packet such as `getKeyId()`, `getFingerprint()` `decryptSecretMPIs()`.

4.2.3 OpenPGP Message Object

An `openpgp` message object is read by `openpgp.read_message` and can be constructed in a string representation by using `openpgp.write_*_message()`.

The object itself, which can only be retrieved by `openpgp.read_message`, has the following methods:

Class	<code>openpgp_msg_message</code>		
Function	<code>decrypt(private_key, sessionkey)</code>		
Description	Decrypts a message and generates a user interface message out of the found signatures. MDC will be verified as well as the message signatures .		
Parameters	<code>private_key</code>	<code>openpgp_msg_privatekey</code>	The private key the message is encrypted with (corresponding to the

			session key).
	sessionkey	openpgp_packet_encryptedsessionkey	The session key to be used to decrypt the message.
Return Value	String	Plain-text of the message or null on error.	

The encrypted session key can be found in the message object in the attribute "sessionKeys".

Class	openpgp_msg_message		
Function	verifySignature()		
Description	Verifies a message signature. This function can be called after read_message if the message was signed only.		
Parameters	none		
Return Value	Boolean	Returns "true" if the signature was correct; otherwise "false".	

4.3 OpenPGP Types

4.3.1 Multi-Precision Integers

Multi-Precision integers (also called MPIs) are unsigned integers used to hold large integers such as the ones used in cryptographic calculations. The type MPI (Multi Precision Integer) is implemented in `js/type/openpgp.type.mpi.js` as a class. An instantiated object can be parsed and written to a binary string using the functions `read()` and `toBin()`.

An MPI consists of two pieces: a two-octet scalar that is the length of the MPI in bits followed by a string of octets that contain the actual integer.

Class	openpgp_type_mpi		
Function	read(input, position, len)		
Description	Parsing function for an MPI (RFC 4880 3.2).		
Parameters	input	String	Payload of MPI data.
	position	Integer	Position to start reading from the input string.
	len	Integer	Length of the packet or the remaining length of input at position.

Return Value	openpgp_type_mpi	(self)
---------------------	------------------	--------

Class	openpgp_type_mpi	
Function	toBin()	
Description	MPI object to a string as specified in RFC4880 3.2	
Parameters	none	
Return Value	String	A binary representation as specified in the OpenPGP standard of the MPI as string.

Class	openpgp_type_mpi	
Function	create(data)	
Description	Creates an MPI from the specified string.	
Parameters	data	String Data to read the MPI from (without the header in front).
Return Value	openpgp_type_mpi	(self)

Class	openpgp_type_mpi	
Function	getByteLength()	
Description	Gets the length of the MPI in bytes.	
Parameters	none	
Return Value	Integer	MPI length in bytes.

Class	openpgp_type_mpi	
Function	getBigInteger()	
Description	Converts the MPI to a BigInteger object.	
Parameters	none	
Return Value	BigInteger	MPI representation as a BigInteger.

Class	openpgp_type_mpi	
Function	toString()	
Description	Generates debug output (pretty print).	
Parameters	none	

Return Value	String	String which provides some information about the MPI.
---------------------	--------	---

4.3.2 String-to-Key Specifiers

String-to-key (S2K) specifiers are used to convert passphrase strings into symmetric-key encryption/decryption keys. They are used in two places, currently: to encrypt the secret part of private keys in the private keyring, and to convert passphrases to encryption keys for symmetrically encrypted messages.

An implementation of this type is defined in `js/type/openpgp.type.s2k.js` with the following functions:

Class	openpgp_type_s2k		
Function	read(input, position)		
Description	Parsing function for a s2k specifier.		
Parameters	input	String	Payload of the string-to-key specifier.
	position	Integer	Position to start reading from the input string.
Return Value	openpgp_type_s2k	(self)	

Class	openpgp_type_s2k		
Function	produce_key(passphrase)		
Description	Produces a key using the specified passphrase and the defined Hash Algorithm (attribute retrieved by read();).		
Parameters	passphrase	String	Passphrase of the key which is used to produce a key for secret key decryption.
Return Value	openpgp_type_s2k	(self)	

Writing of a s2k specifier is currently not implemented, because this implementation cannot create keys.

4.3.3 Key Id

The following functions implement the type key id and can be found in `js/type/openpgp.keyid.js`.

A Key ID is an eight-octet scalar that identifies a key. Implementations SHOULD

NOT assume that Key IDs are unique.

Class	openpgp_type_keyid		
Function	read_packet(input, position)		
Description	Parsing method for a key id. After reading, the attribute "bytes" is set within the object.		
Parameters	input	String	Payload of the key id as string.
	position	Integer	Position to start reading from the input string.
Return Value	openpgp_type_keyid	(self)	

Class	openpgp_type_keyid		
Function	toString()		
Description	Generates debug output (pretty print).		
Parameters	none		
Return Value	String	Debug output for the packet.	

4.4 OpenPGP Packet Types

There is a generic class for parsing single OpenPGP message packets, implemented in `js/packet/openpgp.packet.js`. This class reads the length of the packet header and then invokes the `read_packet` function of the corresponding packet type. If the parsing routine was successful the packet will be returned.

The `openpgp_packet` object has several generic functions for reading packets and writing packet headers. `openpgp_packet.read_packet()` is called by the read functions of the `openpgp` object.

openpgp_packet functions:

Class	openpgp_packet		
Function	write_packet_header(tag_type, length)		
Description	Writes a packet header version 4 with the given tag_type and length to a string.		
Parameters	tag_type	Integer	Integer defining the tag type. (< 32)
	length	Integer	Length of the packet payload which gets appended to the packet header.
Return Value	String	Packet header in string representation.	

Class	openpgp_packet		
Function	write_old_packet_header(tag_type, length)		
Description	Writes a packet header version 3 with the given tag_type and length to a string.		
Parameters	tag_type	Integer	Integer defining the tag type. (< 16)
	length	Integer	Length of the packet payload which gets appended to the packet header.
Return Value	String	Packet header in string representation.	

Class	openpgp_packet		
Function	read_packet(input, position, length)		
Description	Generic packet parser function. Depending on the type read_packet is called by this routine. The return value is one of the openpgp.packet.* types or null if the packet is invalid.		
Parameters	input	String	String to read the packet from.
	position	Integer	Position at input to start reading the packet from.
	length	Integer	Valid length to read (max position – input.length).
Return Value	Object: openpgp_*_packet	A parsed packet with attributes "headerLength" and "packetLength" set.	

Generic openpgp_*_packet functions:

Depending on the type of the tag the packet can have subnotes (tags) which are a property of the tag parsed. Therefore, some packet types introduce the function read_nodes() additionally to the read_packet method to allow the parser to add these packets as sub-packets.

Classes implementing the OpenPGP packets in general have the following functions:

Class	openpgp_*_packet		
Function	read(input, position, len)		
Description	Parsing function for the packet type.		
Parameters	input	String	Payload of packet data.
	position	Integer	Position to start reading from the

			input string.
	len	Integer	Length of the packet or the remaining length of the input at position.
Return Value	openpgp_*_packet	(self)	

Class	openpgp_*_packet		
Function	toString()		
Description	Generates debug output (pretty print).		
Parameters	none		
Return Value	String	Key id as hexadecimal string.	

All other functions are depending on the type.

4.4.1 Encrypted Session Key Packet (tag 1, tag 3)

The `openpgp_packet_encryptedsessionkey` implements two types of tags. Therefore, the `read_packet` function for those 2 types are defined as:

- Tag 1: `read_pub_key_packet(input, position, len)`
- Tag 3: `read_symmetric_key_packet(input, position, len)`

Class	openpgp_packet_encryptedsessionkey		
Function	write_pub_key_packet(publicKeyId, publicMPIs, pubalgo, symmalgo, sessionkey)		
Description	Creates a string representation of a public-key encrypted session key packet.		
Parameters	publicKeyId	String	The public key id corresponding to publicMPIs key as string.
	publicMPIs	Array[openpgp_type_mpi]	Array of MPI objects describing the public key.
	pubalgo	Integer	The corresponding public key algorithm (see RFC4880 9.1).
	symmalgo	Integer	The symmetric cipher algorithm used to encrypt the data within an encrypteddatapacket or encryptedintegrityprotecteddatapacket following this packet (see RFC4880

			9.2).
	sessionkey	String	A string of random bytes representing the session key.
Return Value	String	String representation of a tag 1 packet including the packet header.	

Class	openpgp_packet_encryptedsessionkey		
Function	decrypt(msg, key)		
Description	Decrypts the session key (only for public key encrypted session key packets (tag 1)).		
Parameters	msg	openpgp_msg_message	The message object (with member encryptedData).
	key	openpgp_msg_privatekey	The private key with secMPIs unlocked.
Return Value	String	Decrypted session key.	

Writing or decrypting of tag 3 packets is not implemented.

4.4.2 Signature Packet (tag 2)

The following functions implement writing and verifying an OpenPGP signature. A signature packet describes a binding between a public key and some data. The most common signatures are a signature of a file or a block of text, and a signature that is a certification of a User ID.

The implementation of this type can be found in `js/packet/openpgp.packet.signature.js` and implements the following functions:

Class	openpgp_packet_signature		
Function	write_message_signature(signature_type, data, privatekey)		
Description	Creates a string representation of a version 4 message signature packet (tag 2). Currently this function can only be used to create message text signature (signature type 1).		
Parameters	signature_type	Integer	Should be 1 (one) .
	data	String	Data to be signed.
	privatekey	openpgp_msg_privatekey	Private key used to sign the message (secMPIs MUST be unlocked).
Return Value	String	String representation of a signature packet.	

Class	openpgp_packet_signature		
Function	verify(data, key)		
Description	Verifies the signature packet. Note: no signature types are implemented.		
Parameter	data	String	Data on which the signature applies.
	key	openpgp_msg_publickey	The public key to verify the signature.
Return Value	Boolean	If the signature was correct "true" is returned. Otherwise "false".	

Class	openpgp_packet_signature		
Function	getIssuer()		
Description	Gets the issuer key id of this signature.		
Parameters	none		
Return Value	String	Key id as string.	

Class	openpgp_packet_signature		
Function	getIssuerKey()		
Description	Tries to get the corresponding public key out of the public keyring for the issuer who created this signature		
Parameters	none		
Return Value	{obj: openpgp_msg_publickey, text: String}	If found the public key will be returned; null otherwise.	

4.4.3 One-Pass Signature Packet (tag 4)

The implementation of One-Pass Signatures is located in `js/packet/openpgp_packet.onepasssignature.js`.

The One-Pass Signature packet precedes the signed data and contains enough information to allow the receiver to begin calculating any hashes needed to verify the signature. It allows the Signature packet to be placed at the end of the message, so that the signer can compute the entire signed message in one pass.

For writing packets the function `write_packet` is used:

Class	openpgp_packet_onepasssignature		
Function	write_packet(type, hashalgorithm, privatekey, length, nested)		
Description	Creates a string representation of a One-Pass Signature packet.		
Parameters	type	Integer	Signature types as described in RFC4880 Section 5.2.1.
	hashalgorithm	Integer	The hash algorithm used within the signature.
	privatekey	openpgp_msg_privatekey	The private key used to generate the signature.
	length	Integer	Length of data to be signed.
	nested	Boolean	Boolean showing whether the signature is nested. "true" indicates that the next packet is another One-Pass Signature packet that describes another signature to be applied to the same message data.
Return Value	String	A string representation of a One-Pass Signature packet	

4.4.4 Key Material Packet (tag 5, tag 6, tag 7, tag 14)

The Key Material packet is implemented in one class implementing 4 packet types. The following methods are used to parse the packet:

- Private Key Packet (tag 5)
 - read_tag5(input, position, len) -> read_priv_key
- Private Sub Key Packet (tag 7)
 - read_tag7(input, position, len) -> read_priv_key
- Public Key Packet (tag 6)
 - read_tag6(input, position, len) -> read_pub_key
- Public Sub Key Packet (tag 14)
 - read_tag14(input, position, len) -> read_pub_key

All of them follow the same syntax as described for read_packet above.

The keymaterial implementation is located in

js/packet.openpgp.packet.keymaterial.js. Additionally, the following functions are provided:

Class	openpgp_packet_keymaterial		
Function	decryptSecretMPIs(str_passphrase)		
Description	Decrypts the private key MPIs which are needed to use the key. The attribute hasUnencryptedSecretKeyData of this object instance should be "false" otherwise.		
Parameters	str_passphrase	String	The passphrase for this private key as string.
Return Value	Boolean	"true" if the passphrase was correct; "false" otherwise. The secMPIs attribute is decrypted if this function returned "true".	

Class	openpgp_packet_keymaterial		
Function	read_nodes(parent_node, input, position, len)		
Description	Continue parsing packets belonging to the key material such as signatures.		
Parameters	parent_node	Object	Parent object.
	input	String	Input string to read the packet(s) from.
	position	Integer	Start position for the parser at string input.
	len	Integer	Length of the packet(s) or remaining length of input.
Return Value	String	Length in bytes of the amount read.	

Class	openpgp_packet_keymaterial		
Function	verifyKey()		
Description	Checks the validity for the usage of this (sub)key.		
Parameters	none		
Return Value	Integer	0 = bad key; 1 = expired; 2 = revoked; 3 = valid	

Class	openpgp_packet_keymaterial		
Function	getKeyId()		
Description	Calculates the key id of the key.		

Parameters	none	
Return Value	String	An 8 byte key id as string.

Class	openpgp_packet_keymaterial	
Function	getFingerprint()	
Description	Calculates the fingerprint of the key.	
Parameters	none	
Return Value	String	String containing the fingerprint.

4.4.5 Compressed Data Packet (tag 8)

The Compressed Data packet contains compressed data. Typically, this packet is found as the contents of an encrypted packet, or following a Signature or One-Pass Signature packet, and contains a literal data packet.

The implementation of a compressed data packet can be found in `js/packet/openpgp.packet.compressed.js`. Currently, no compression algorithm is implemented. The generic functions are available. The following functions are implemented as well:

Class	openpgp_packet_compressed	
Function	decompress()	
Description	Decompression method for decompressing the compressed data.	
Parameters	none	
Return Value	String	The decompressed data.

Class	openpgp_packet_compressed		
Function	compress(type, data)		
Description	Compress the packet data available in attribute decompressedData.		
Parameters	type	Integer	Algorithm to be used (see RFC 4880 9.3).
	data	String	Data to be compressed.
Return Value	String	The compressed data.	

Class	openpgp_packet_compressed	
Function	write_packet(algorithm, data)	

Description	Creates a string representation of the packet.		
Parameters	type	Integer	Algorithm to be used (see RFC 4880 9.3).
	data	String	Data to be compressed-
Return Value	String	String representation of the packet.	

4.4.6 Symmetrically Encrypted Data Packet (tag 9)

The Symmetrically Encrypted Data packet contains data encrypted with a symmetric-key algorithm. When it has been decrypted, it contains other packets (usually a literal data packet or compressed data packet, but in theory other Symmetrically Encrypted Data packets or sequences of packets that form whole OpenPGP messages).

The implementation can be found in `js/packet/openpgp.packet.encrypteddata.js`. Next to the generic functions the following functions are available:

Class	openpgp_packet_encrypteddata		
Function	decrypt_sym(symmetric_algorithm_type, key)		
Description	Symmetrically decrypt the packet data.		
Parameters	symmetric_algorithm_type	Integer	Symmetric key algorithm to be used (see RFC4880 9.2).
	key	String	Key as string with the corresponding length to the algorithm.
Return Value	String	The decrypted data.	

Class	openpgp_packet_encrypteddata		
Function	write_packet(algo, key, data)		
Description	Creates a string representation of the symmetrically encrypted data packet. This function symmetrically encrypts the data with the algorithm and the key specified.		
Parameters	algo	Integer	Symmetric key algorithm to be used (see RFC4880 9.2).
	key	String	Key as string with the corresponding length to the algorithm.
	data	String	Data to be encrypted within this packet.

Return Value	String	String representation of the packet.
---------------------	--------	--------------------------------------

4.4.7 Marker Packet (tag 10)

This type is implemented for completeness and is only handled by the parser. The implementation of the parser can be found in `js/packet/openpgp.packet.marker.js`.

An experimental version of PGP used this packet as the Literal packet, but no released version of PGP generated Literal packets with this tag. With PGP 5.x, this packet has been reassigned and is reserved for use as the Marker packet.

Such a packet MUST be ignored when received.

4.4.8 Literal Data Packet (tag 11)

A Literal Data packet contains the body of a message; data that is not to be further interpreted.

The implementation can be found in `js/packet/openpgp.packet.literaldata.js` and implements reading (`read_packet`) and writing (`write_packet`) those packets. The write function is defined as:

Class	openpgp_packet_literaldata		
Function	write_packet(data)		
Description	Creates a string representation of the packet.		
Parameters	data	String	The data to be inserted as body.
Return Value	String	String representation of the packet.	

4.4.9 User ID Packet (tag 13)

A User ID packet consists of UTF-8 text that is intended to represent the name and email address of the key holder. By convention, it includes an RFC 2822 [RFC2822] mail name-addr, but there are no restrictions on its content. The packet length in the header specifies the length of the User ID.

The implementation can be found in `js/packet/openpgp.packet.userid.js` with the following implemented functions:

Class	openpgp_packet_userid		
Function	write_packet(user_id)		
Description	Creates a string representation of the user id packet.		
Parameters	data	String	User id as string ("John Doe <john.doe@mail.us>")

Return Value	String	String representation of the packet.
---------------------	--------	--------------------------------------

Class	openpgp_packet_userid		
Function	read_nodes(parent_node, input, position, len)		
Description	Continue parsing packets belonging to the user id such as certification (revocation) signatures.		
Parameters	parent_node	Object	The parent object.
	input	String	Input string to read the packet(s) from.
	position	Integer	Start position for the parser at string input.
	len	Integer	Length of the packet(s) or remaining length of input.
Return Value	String	Length in bytes of the amount read.	

Class	openpgp_packet_userid		
Function	hasCertificationRevocationSignature(keyId)		
Description	Lookup function to find certification revocation signatures for a given key id.		
Parameters	keyId	String	String containing the key id of the issuer of the signature to lookup.
Return Value	openpgp_packet_signature	A CertificationRevocationSignature if found; otherwise null.	

Class	openpgp_packet_userid		
Function	verifyCertificationSignatures(publicKeyPacket)		
Description	Verifies all certification signatures. This method does not consider possible revocation signatures.		
Parameters	publicKeyPacket	openpgp_packet_keymaterial	publicKeyPacket the top level key material
Return Value	Array[Integer]	An array of integers corresponding to the array of certification signatures. The meaning of each integer is the following: 0 = bad signature 1 = signature expired 2 = issuer key not available	



		3 = revoked 4 = signature valid 5 = signature by key owner expired 6 = signature by key owner revoked
--	--	--

Class	openpgp_packet_userid		
Function	verify(publicKeyPacket)		
Description	Verifies the signatures of the user id.		
Parameters	publicKeyPacket	openpgp_packet_keymaterial	publicKeyPacket the top level key material
Return Value	Integer	0 if the userid is valid; 1 = userid expired; 2 = userid revoked	

The verification of signatures is incomplete.

The following functions are currently not implemented but exist as stubs:

- addCertification(publicKeyPacket, privateKeyPacket)
- revokeCertification(publicKeyPacket, privateKeyPacket)

4.4.10 User Attribute Packet (tag 17)

The User Attribute packet is a variation of the User ID packet. It is capable of storing more types of data than the User ID packet, which is limited to text. Like the User ID packet, a User Attribute packet may be certified by the key owner ("self-signed") or any other key owner who cares to certify it. Except as noted, a User Attribute packet may be used anywhere that a User ID packet may be used.

While User Attribute packets are not a required part of the OpenPGP standard, implementations SHOULD provide at least enough compatibility to properly handle a certification signature on the User Attribute packet. A simple way to do this is by treating the User Attribute packet as a User ID packet with opaque contents, but an implementation may use any method desired.

This packet is currently implemented only on the parser side. No signature verification or writing is implemented. The implementation can be found in `js/packet/openpgp.packet.userattribute.js` and implements next to `read_packet` and `toString()` the `read_nodes()` function.

Class	openpgp_packet_userattribute		
Function	read_nodes(parent_node, input, position, len)		
Description	Continue parsing packets belonging to the user id such as certification		

	(revocation) signatures.		
Parameters	parent_node	Object	The parent object.
	input	String	Input string to read the packet(s) from.
	position	Integer	Start position for the parser at string input.
	len	Integer	Length of the packet(s) or remaining length of input.
Return Value	String	Length in bytes of the amount read.	

4.4.11 Symmetrically Encrypted Integrity Protected Data Packet (tag 18)

The Symmetrically Encrypted Integrity Protected Data packet is a variant of the Symmetrically Encrypted Data packet. It is a new feature created for OpenPGP that addresses the problem of detecting a modification to encrypted data. It is used in combination with a Modification Detection Code packet.

The implementation of this packet is complete and can be found in `js/packet/openpgp.packet.encryptedintegrityprotecteddata.js`.

The following functions are implemented:

Class	openpgp_packet_encryptedintegrityprotecteddata		
Function	decrypt(symmetrical_algorithm_type, key)		
Description	Decrypts the encrypted data contained in this object; <code>read_packet</code> must have been called before.		
Parameters	symmetrical_algorithm	Integer	The selected symmetric encryption algorithm to be used.
	key	String	The key of length depending on the cipher to be used.
Return Value	String	Decrypted data (containing more OpenPGP packets).	

Class	openpgp_packet_encryptedintegrityprotecteddata		
Function	write_packet(symmetrical_algorithm, key, data)		
Description	Creates a string representation of a Symmetric Encrypted Integrity Protected Data Packet (tag 18) (see RFC4880 5.13).		
Parameters	symmetrical_algorithm	Integer	Selected symmetric encryption algorithm to be used .
	key	String	Key of length depending on the cipher to be used.



	data	String	Plain-text data to be encrypted within the packet.
Return Value	String	string representation of the packet.	

4.4.12 Modification Detection Code Packet (tag 19)

The Modification Detection Code packet contains a SHA-1 hash of plaintext data, which is used to detect message modification. It is only used with a Symmetrically Encrypted Integrity Protected Data packet. The Modification Detection Code packet MUST be the last packet in the plaintext data that is encrypted in the Symmetrically Encrypted Integrity Protected Data packet, and MUST appear in no other place.

Writing functionality of this packet is implemented within the `openpgp_packet_encryptedintegrityprotecteddata` class. The implementation can be found in `js/packet/openpgp.packet.modificationdetectioncode.js`. No other functions than the parsing routines are implemented here. The verification of the MDC is also done in the EIPD packet.

4.5 Cryptography

A common interface for using cryptographic algorithms is provided by the cryptographic API. Numbers specifying the algorithms can be found in RFC 4880 Section 9.

All symmetric algorithms and hash functions are contributions by other authors (see Section 3.1). For a more generic access to those implementations, a function has been included in these libraries.

For asymmetric cryptography (DSA, RSA and Elgamal) a BigInteger library has been used.

4.5.1 OpenPGP Cipher Feedback Mode (CFB)

These functions are implementing OpenPGPs Cipher Feedback Mode as described in RFC4880 section 13.9.

Class	none		
Function	<code>openpgp_cfb_encrypt</code> (<code>prefixrandom</code> , <code>blockcipherencryptfn</code> , <code>plaintext</code> , <code>block_size</code> , <code>key</code> , <code>resync</code>)		
Description	This function encrypts given plain-text with the specified <code>prefixrandom</code> using the specified <code>blockcipher</code> to encrypt a message.		
Parameters	<code>prefixrandom</code>	String	Random bytes of <code>block_size</code> length provided as a string to be

			used in prefixing the data.
	blockcipherfn	function(block, key)	The algorithm encrypt function to encrypt data in one block_size encryption. The function must be specified as blockcipherfn([integer_array(integers 0..255)] block, [integer_array(integers 0..255)] key) returning an array of bytes (integers 0..255).
	plaintext	String	Data to be encrypted; provided as a string.
	block_size	Integer	Block size in bytes of the algorithm used.
	key	Array[Integer]	Key to be used to encrypt the data as integer_array(integers 0..255)]. This will be passed to the blockcipherfn.
	resync	Boolean	A boolean value specifying if a resync of the IV should be used or not. The encrypteddatapacket uses the "old" style with a resync. Encryption within an encryptedintegrityprotecteddata packet is not resyncing the IV.
Return Value	String	String with the encrypted data.	

Class	none		
Function	openpgp_cfb_decrypt(blockcipherencryptfn, block_size, key, ciphertext, resync)		
Description	This function decrypts a given plain-text using the specified blockcipher to decrypt a message.		
Parameters	blockcipherfn	function(block, key)	The algorithm encrypt function to encrypt data in one block_size encryption. The function must be specified as blockcipherfn([integer_array(integers 0..255)] block, String key) returning an array of bytes (integers 0..255).

	ciphertext	String	Ciphertext to be decrypted provided as a string.
	block_size	Integer	Block size in bytes of the algorithm used.
	key	String	Key to be used to decrypt the ciphertext as string. This will be passed to the blockcipherfn.
	resync	Boolean	A boolean value specifying if a resync of the IV should be used or not. The encrypteddatapacket uses the "old" style with a resync. Encryption within an encryptedintegrityprotecteddata packet is not resyncing the IV.
Return Value	String	String with the plain-text data.	

Class	none		
Function	openpgp_cfb_mdc(blockcipherencryptfn, block_size, key, ciphertext)		
Description	Decrypts the prefixed data for the Modification Detection Code (MDC) computation.		
Parameters	blockcipherencryptfn	function(block, key)	Cipher function to use.
	block_size	Integer	Block size in bytes of the algorithm used.
	key	String	Key for encryption.
	ciphertext	String	Encrypted data.
Return Value	String	Plain-text data of D(ciphertext) with blocksize length +2	

Additionally, two functions implementing the normal CFB-Mode for decrypting password protected secret key material:

- String normal_cfb_encrypt(blockcipherencryptfn, block_size, key, plaintext, iv) // untested
- String normal_cfb_decrypt(blockcipherencryptfn, block_size, key, ciphertext, iv)

4.5.2 OpenPGP Cryptography API

The following functions are located in `js/ciphers/openpgp.crypto.js` and implement the cryptographic API:

Class	none		
Function	<code>openpgp_crypto_asymmetricEncrypt(algo, publicMPIs, data)</code>		
Description	Encrypts data using the specified public key MPIs and the specified algorithm.		
Parameters	<code>algo</code>	Integer	Algorithm to be used (See RFC4880 9.1).
	<code>publicMPIs</code>	<code>[Array[openpgp_type_mpi]]</code>	Algorithm dependent MPIs.
	<code>data</code>	<code>openpgp_type_mpi</code>	Data to be encrypted as MPI.
Return Value	Object	If RSA: an <code>openpgp_type_mpi</code> ; if elgamal encryption: an array of two <code>openpgp_type_mpi</code> is returned; otherwise null.	

Class	none		
Function	<code>openpgp_crypto_asymmetricDecrypt(algo, publicMPIs, secretMPIs, dataMPIs)</code>		
Description	Decrypts data using the specified public key MPIs of the private key, the specified secretMPIs of the private key and the specified algorithm.		
Parameters	<code>algo</code>	Integer	Algorithm to be used (See RFC4880 9.1).
	<code>publicMPIs</code>	<code>Array[openpgp_type_mpi]</code>	Algorithm dependent MPIs of the public key part of the private key.
	<code>secretMPIs</code>	<code>Array[openpgp_type_mpi]</code>	Algorithm dependent MPIs of the private key used.
	<code>data</code>	<code>openpgp_type_mpi</code>	Data to be decrypted as MPI.
Return Value	<code>BigInteger</code>	Returns a big integer containing the decrypted data; otherwise null.	

Class	none
--------------	------

Function	<code>openpgp_crypto_symmetricEncrypt(prefixrandom, algo, key, data, openpgp_cfb)</code>		
Description	Symmetrically encrypts data using <code>prefixrandom</code> , a key with length depending on the algorithm in <code>openpgp_cfb</code> mode with or without <code>resync</code> (MDC style).		
Parameters	<code>prefixrandom</code>	String	Random bytes of <code>block_size</code> length provided as a string to be used in prefixing the data.
	<code>algo</code>	Integer	Algorithm to use (see RFC4880 9.2).
	<code>key</code>	String	Key as a string. Length is depending on the algorithm used.
	<code>data</code>	String	Data to encrypt.
	<code>openpgp_cfb</code>	boolean	A boolean value specifying if a <code>resync</code> of the IV should be used or not. The <code>encrypteddatapacket</code> uses the "old" style with a <code>resync</code> . Encryption within an <code>encryptedintegrityprotecteddata</code> packet is not <code>resyncing</code> the IV.
Return Value	String	Encrypted data.	

Class	none		
Function	<code>openpgp_crypto_symmetricDecrypt(algo, key, data, openpgp_cfb)</code>		
Description	Symmetrically decrypts data using a key with length depending on the algorithm in <code>openpgp_cfb</code> mode with or without <code>resync</code> (MDC style).		
Parameters	<code>algo</code>	Integer	Algorithm to use (see RFC4880 9.2).
	<code>key</code>	String	Key as a string. Length is depending on the algorithm used.
	<code>data</code>	String	Data to decrypt.
	<code>openpgp_cfb</code>	boolean	A boolean value specifying if a <code>resync</code> of the IV should be used or not. The <code>encrypteddatapacket</code> uses the "old" style with a <code>resync</code> . Encryption within an <code>encryptedintegrityprotecteddata</code>

			packet is not resyncing the IV.
Return Value	String	Plain-text data.	

Class	none		
Function	openpgp_crypto_MDCSystemBytes(algo, key, data)		
Description	Retrieve the MDC prefixed bytes by decrypting them.		
Parameters	algo	Integer	Algorithm to use (see RFC4880 9.2).
	key	String	Key as a string. Length is depending on the algorithm used.
	data	String	Encrypted data where the prefix is decrypted from.
Return Value	String	Plain-text data of the prefixed data.	

Class	none		
Function	openpgp_crypto_generateSessionKey(algo)		
Description	Generating a session key for the specified symmetric algorithm.		
Parameters	algo	Integer	Algorithm to use (see RFC4880 9.2).
Return Value	String	Secure random bytes as a string to be used as a key.	

Class	none		
Function	openpgp_crypto_verifySignature(algo, hash_algo, msg_MPIs, publickey_MPIs, data)		
Description	Verifies a signature on given data using the specified algorithm.		
Parameters	algo	Integer	Asymmetric cipher algorithm to use (See RFC4880 9.1).
	hash_algo	Integer	Hash algorithm to use (See RFC4880 9.4).
	msg_MPIs	Array[openpgp_type_mpi]	Signature MPIs
	publickey_MPI	Array[openpgp_	Public key MPIs

	s	type_mpi]	
	data	String	Data on where the signature was computed on.
Return Value	String	"true" if signature verification was successful (signature data data was equal to data over hash)	

Class	none		
Function	openpgp_crypto_signData(hash_algo, algo, publicMPIs, secretMPIs, data)		
Description	Create a signature on data using the specified algorithm.		
Parameters	hash_algo	Integer	Hash algorithm to use (See RFC4880 9.4).
	algo	Integer	Asymmetric cipher algorithm to use (See RFC4880 9.1).
	publickey_MP Is	Array[openpgp_ type_mpi]	Public key MPIs of the private key.
	secretkey_MP Is	Array[openpgp_ type_mpi]	Private key MPIs which is used to sign the data.
	data	String	Data to be signed.
Return Value	String or openpgp_type_ mpi		

Class	none		
Function	openpgp_crypto_hashData(algo, data)		
Description	Create a hash on the specified data using the specified algorithm.		
Parameters	algo	Integer	Hash algorithm to use (See RFC4880 9.4).
	data	String	Data to be hashed.
Return Value	String	Hash value.	

Class	none		
Function	openpgp_crypto_getHashByteLength(algo)		
Description	Calculates the hash size of the specified hash algorithm type in bytes.		

Parameters	algo	Integer	Hash algorithm to use (See RFC4880 9.4).
Return Value	String	Size in bytes of the resulting hash.	

Class	none		
Function	openpgp_crypto_getRandomBytes(length)		
Description	Retrieve secure random byte string of the specified length.		
Parameters	length	Integer	Length in bytes to generate.
Return Value	String	Random byte string.	

Class	none		
Function	openpgp_crypto_getPseudoRandom(from, to)		
Description	Return a pseudo-random number in the specified range.		
Parameters	from	Integer	Minimum of the random number.
	to	Integer	Maximum of the random number (max 32bit).
Return Value	Integer	Pseudo random number.	

Class	none		
Function	openpgp_crypto_getSecureRandom(from, to)		
Description	Return a secure random number in the specified range.		
Parameters	from	Integer	Minimum of the random number.
	to	Integer	Maximum of the random number (max 32bit).
Return Value	Integer	Secure random number within the specified range.	

Class	none		
Function	openpgp_crypto_getRandomBigInteger(bits)		
Description	Create a secure random big integer of specified bits length.		
Parameters	bits	Integer	bit length of the MPI to create.

Return Value	BigInteger	Resulting big integer.
---------------------	------------	------------------------

4.6 Encoding

4.6.1 ASCII-Armor and Base64

The following functions are located in file `js/encoding/openpgp.encoding.asciarmor.js` and are using the base64 encoding routines in `js/encoding/base64.js`.

Class	none		
Function	<code>openpgp_encoding_deArmor(text)</code>		
Description	De-armor an OpenPGP armored block.		
Parameters	text	String	OpenPGP armored message
Return Value	Object	Either the bytes of the decoded message or an object with attribute "text" containing the message text and an attribute "openpgp" containing the bytes.	

Class	none		
Function	<code>openpgp_encoding_armor(messageType, data, partindex, parttotal)</code>		
Description	Armor an OpenPGP binary packet block.		
Parameters	messageType	Integer	Message type (as returned by <code>getPGPMessageType</code>).
	data	String	String of binary OpenPGP Messages to armor.
	partindex	Integer	If the message is partially armored the number of the part to armor.
	parttotal	Integer	If the message is partially armored the total number of the parts to be armored.
Return Value	String	Armored OpenPGP message.	

Class	none		
Function	<code>getChecksum(data)</code>		

Description	Calculates a CRC-24 checksum over the given data and returns it base64 encoded.		
Parameters	data	String	data to create a CRC-24 checksum for
Return Value	String	base64 encoded checksum	

Class	none		
Function	verifyCheckSum(data, checksum)		
Description	Calculates the checksum over the given data and compares it with the given base64 encoded checksum.		
Parameters	data	String	Data to create a CRC-24 checksum for.
	checksum	String	The checksum as a string in base64 to compare the calculated checksum against.
Return Value	Boolean	"true" if the given checksum is correct; otherwise "false".	

Class	none		
Function	createcrc24(data)		
Description	Internal function to calculate a CRC-24 checksum over a given string (data).		
Parameters	data	String	Data to create a CRC-24 checksum for.
Return Value	Integer	CRC-24 checksum as number.	

4.6.2 PKCS1 Encoding

The following functions are located in file `js/encoding/openpgp.encoding.js`:

Class	none		
Function	openpgp_encoding_base64_encode(message)		
Description	Wrapper function for the base64 codec. This function encodes a String (message) in base64 (radix-64)		
Parameters	message	String	Message to encode.

Return Value	String	base64 encoded data.
---------------------	--------	----------------------

Class	none	
Function	openpgp_encoding_base64_decode(message)	
Description	Wrapper function for the base64 codec. This function decodes a String(message) in base64 (radix-64).	
Parameters	message	String Base64 encoded data
Return Value	String	Raw data after decoding.

Class	none	
Function	openpgp_encoding_html_encode(message)	
Description	Wrapper function for jquery library. This function escapes HTML characters within a string. This is used to prevent XSS.	
Parameters	message	String Message to escape.
Return Value	String	HTML encoded string.

Class	none	
Function	openpgp_encoding_eme_pkcs1_encode(message, length)	
Description	Creates a EME-PKCS1-v1_5 padding (See RFC4880 13.1.1).	
Parameters	message	String Message to be padded.
	length	Integer Length to the resulting message.
Return Value	String	EME-PKCS1 padded message.

Class	none	
Function	openpgp_encoding_emsa_pkcs1_encode(algo, data, keylength)	
Description	Creates a EMSA-PKCS1-v1_5 padding (See RFC4880 13.1.3).	
Parameters	algo	Integer Hash algorithm type used.
	data	String Data to be hashed within the padding.
	keylength	Integer Key size of the public MPI in bytes.

Return Value	String	EME-PKCS1 padded message.
---------------------	--------	---------------------------

Class	none		
Function	openpgp_encoding_emsa_pkcs1_encode(algo, data, keylength)		
Description	Creates a EMSA-PKCS1-v1_5 padding (See RFC4880 13.1.3).		
Parameters	algo	Integer	Hash algorithm type used.
	data	String	Data to be hashed within the padding.
	keylength	Integer	Key size of the public MPI in bytes.
Return Value	String	EMSA-PKCS1 padded message.	

Class	none		
Function	openpgp_encoding_emsa_pkcs1_decode(algo, data)		
Description	Extracts the hash out of an EMSA-PKCS1-v1.5 padding (See RFC4880 13.1.3) .		
Parameters	algo	Integer	Hash algorithm type used.
	data	String	Hash in pkcs1 encoding.
Return Value	String	Hash as String.	

This file also contains ASN1 object identifiers for hashes (See RFC4880 5.2.2).

4.7 Utility functions

The file `js/util.js` provides functions for data conversion and string output and is instantiated on load.

Class	util		
Function	hexstrdump(str)		
Description	Create hexdump from a binary string.		
Parameters	str	String	String to convert
	Return Value	String	String containing the hexadecimal values.

Class	util		
Function	hexidump(str)		
Description	Create hexdump from an array of integers containing values from 0 to 255.		
Parameters	str	Array[Integer]	Array to convert.
Return Value	String	String containing the hexadecimal values.	

Class	util		
Function	str2bin(str)		
Description	Convert a string to an array of integers (0 to 255).		
Parameters	str	String	String to convert.
Return Value	Array[Integer]	Array of integers.	

Class	util		
Function	bin2str(bin)		
Description	Convert an array of integers (0 to 255) to a string.		
Parameters	str	Array[Integer]	Array to convert.
Return Value	String	String representation of the array.	

Class	util		
Function	calc_checksum(text)		
Description	Calculates a 16bit sum of a string by adding each character codes modulus 65535.		
Parameters	text	String	String to create a sum of.
Return Value	Integer	Integer containing the sum of all character codes modulo 65535.	

Class	util		
Function	shiftRight(value, bitcount)		
Description	Shifting a string to n bits right .		
Parameters	value	String	String to shift.

	bitcount	Integer	Amount of bits to shift (MUST be smaller than 9).
Return Value	String	Shifted string.	

Class	util		
Function	get_hashAlgorithmString(algo)		
Description	Return the algorithm type as string.		
Parameters	algo	Integer	Hash Algorithm type.
Return Value	String	String representing the hash (e.g. 2 == "SHA-1").	

4.7.1 Printing Messages

For messages printing containing debug-, error- or info-messages the following functions are provided within `js/util.js`. The page including the code MUST define a `"showMessage(text)"` function. An HTML "tt" entity containing a paragraph with a style attribute where the error-message is HTML encoded in is provided to this function.

Class	util		
Function	print_debug(str)		
Description	Helper function to print a debug-message. Debug-messages are only printed if <code>openpgp.config.debug</code> is set to "true". Line feeds ('\n') are automatically converted to HTML line feeds ' '.		
Parameters	text	String	String containing the message to display.
Return Value	none	undefined	

Class	util		
Function	print_error(str)		
Description	Helper function to print an error-message. Line feeds ('\n') are automatically converted to HTML line feeds ' '.		
Parameters	text	String	String containing the message to display.
Return Value	none	undefined	



Functions `print_warning(str)` and `print_info(str)` are equivalent to `print_error(str)`.

5 Appendix

5.1 File Reference

File	Description	Chapter / Page
js/openpgp.config.js	Storing / Retrieving / Accessing the Configuration	4.1.2 / 13
js/type/openpgp.type.keyid.js	OpenPGP Type KeyID	4.3.3 / 22
js/type/openpgp.type.s2k.js	OpenPGP Type String-2-Key Specifier	4.3.2 / 22
js/type/openpgp.type.mpi.js	OpenPGP Type Multi-precision Integer	4.3.1 / 20
js/ciphers/asymmetricencryption/jsbn.js	JavaScript Bignum library	3.1 / 9
js/ciphers/asymmetricencryption/jsbn2.js		3.1 / 9
js/ciphers/asymmetricencryption/rsa.js	encrypt, decrypt, sign and verify using RSA Algorithm	
js/ciphers/asymmetricencryption/dsa.js	sign and verify using DSA Algorithm	
js/ciphers/asymmetricencryption/elgamal.js	encrypt and decrypt using Elgamal	
js/ciphers/hash/md5.js	MD5 Hash Algorithm	3.1 / 9
js/ciphers/hash/ripe-md.js	RIPE/MD-160 Hash Algorithm	3.1 / 9
js/ciphers/hash/sha.js	SHA (-1,224,256,384,512) Hash Algorithm	3.1 / 9
js/ciphers/symmetricencryption/twofish.js	Twofish Symmetric Cipher Algorithm	3.1 / 9
js/ciphers/symmetricencryption/aes.js	AES Symmetric Cipher Algorithm	3.1 / 9
js/ciphers/symmetricencryption/blowfish.js	Blowfish Symmetric Cipher Algorithm	3.1 / 9
js/ciphers/symmetricencryption/cast5.js	CAST 5 Symmetric Cipher Algorithm	3.1 / 9
js/ciphers/symmetricencryption/dessrc.js	Triple DES EDE Symmetric Cipher Algorithm	3.1 / 9
js/ciphers/openpgp.crypto.js	API for using Hash, Symmetric - and Asymmetric Cipher algorithms	4.5.2 / 39

js/ciphers/openpgp.cfb.js	Implementation of OpenPGP CFB mode	3.1 / 9 or 4.5.1 / 36
js/openpgp.js	The Main API	4.1.1 / 11
js/packet/ openpgp.packet.encrypteddata.js	OpenPGP Message tag 18	4.4.11 / 35
js/packet/ openpgp.packet.keymaterial.js	OpenPGP Message tag 5,6,7,14	4.4.4 / 28
js/packet/ openpgp.packet.onepasssignature.js	OpenPGP Message tag 4	4.4.3 / 27
js/packet/openpgp.packet.marker.js	OpenPGP Message tag 10	4.4.7 / 32
js/packet/ openpgp.packet.compressed.js	OpenPGP Message tag 8	4.4.5 / 30
js/packet/ openpgp.packet.modificationdetection code.js	OpenPGP Message tag 19	4.4.12 / 36
js/packet/openpgp.packet.js	Generic OpenPGP Message Packet Implementation	4.4 / 23
js/packet/ openpgp.packet.literaldata.js	OpenPGP Message tag 11	4.4.8 / 32
js/packet/openpgp.packet.encryptedin tegrityprotecteddata.js	OpenPGP Message tag 18	4.4.11 / 35
js/packet/openpgp.packet.signature.js	OpenPGP Message tag 2	4.4.2 / 26
js/packet/openpgp.packet.userid.js	OpenPGP Message tag 13	4.4.9 / 32
js/packet/ openpgp.packet.userattribute.js	OpenPGP Message tag 17	4.4.10 / 34
js/packet/ openpgp.packet.encryptedsessionkey .js	OpenPGP Message tag 1, 3	4.4.1 / 25
js/encoding/base64.js	Base64 (Radix 64) Impl.	3.1 / 9
js/encoding/openpgp.encoding.js	OpenPGP PKCS1 encoding	4.6.2 / 45
js/encoding/ openpgp.encoding.asciarmor.js	OpenPGP Ascii-Armor Impl.	4.6.1 / 44
js/util.js	Utility Functions (Debug, data conversion)	4.7 / 47
gmail-api.js	Google Mail Integration	2.3 / 8
manifest.json	Extension Manifest	



jquery.min.js	JQuery Library	3.1 / 9
background.html	Background Page (component)	2.1 / 6
contentscripts.js	Content Script (component)	2.3 / 8