Webmail Using the Hush Encryption Engine

Introduction

The webmail service offered by Hushmail and Privacy Professional (referred to as Hushmail) utilizes a basic PHP-based webmail system that uses Cyrus IMAP servers for email storage. Embedding the Hush Encryption Engine (Engine) in a frame in the web browser, and integrating it into the authentication and email transfer process adds end-to-end security to the webmail.

Terms in this Document

Client – This is the processing device utilized by the end-user to perform key generation, encryption, or signature operations using the Engine.
Engine – Shortened reference to the Hush Encryption Engine, contained in the appropriate wrapper. See *Hush Encryption Engine White Paper* for more information.
Key Server – This is the processing device to which the Engine connects to store and retrieve public keys and encrypted private keys.
Alias – A string used to identify a user (an owner of a public key) on the key server. It is usually in the form of an email address, such as "username@domain.com".
Passphrase – A secret string, known only to the end-user and never revealed to any other party, which is used to a) generate the value needed to retrieve encrypted private keys from the Key Server and b) decrypt said encrypted private keys.
Web Server – A web server on which PHP is installed. It connects to IMAP servers to retrieve email information, and uses SMTP to send out email.

Pre-activation – A process by which an administrator on a domain allocates an alias to be associated with keys and certificates, without actually creating the keys and certiricates.

Requirements

Hushmail fulfills the following requirements:

1. Hushmail must provide the capability to encrypt, decrypt, sign and verify email messages and attachments using a public key infrastructure based on OpenPGP.
2. Hushmail must provide a user experience comparable to that of webmail systems that do not offer public key based security.
3. Private keys and private data may only be decrypted on the client computer, never on any server.

Architecture

Figure 1 illustrates the interaction between the Engine and the web-based email system.

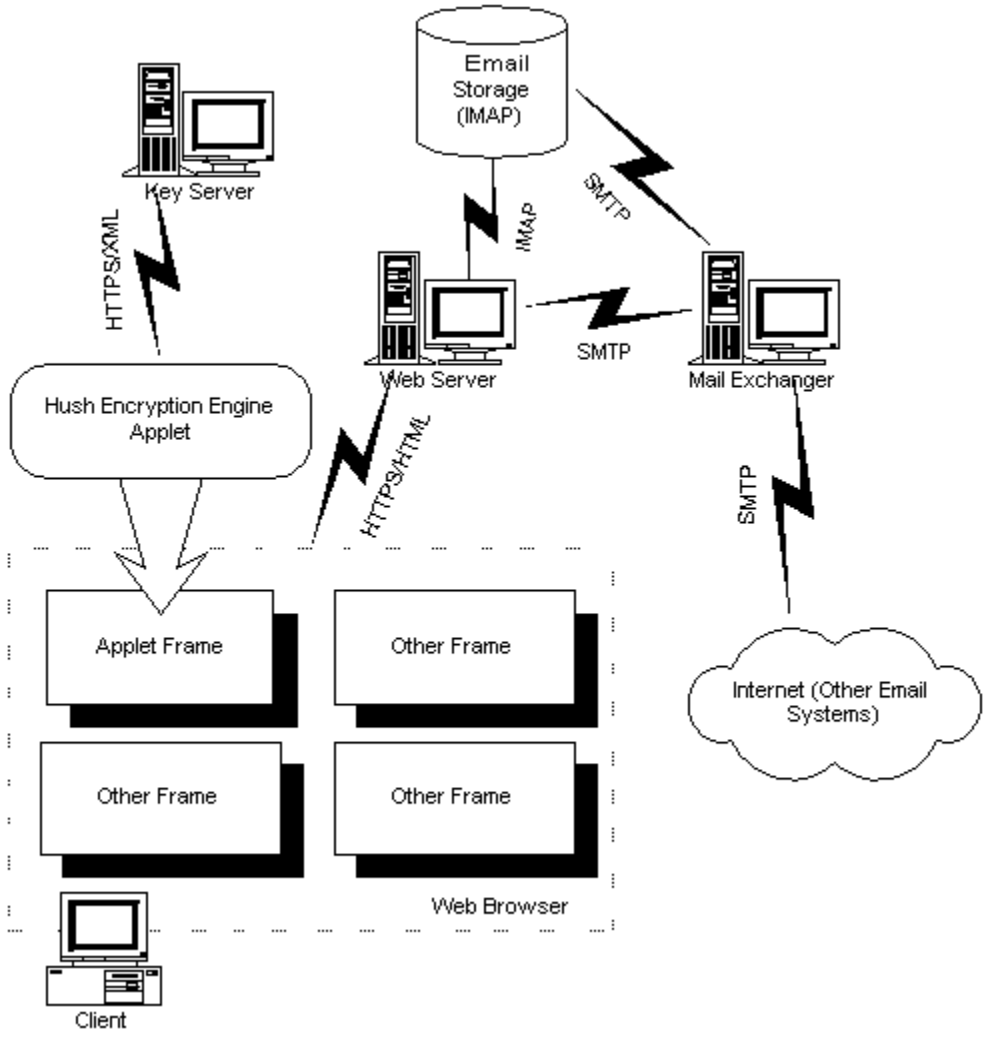

Figure 1

The Engine is loaded at the start of the session into the top left frame. That frame never reloads during the entire course of the session, allowing the Engine to maintain state throughout. Other frames are reloaded with various content during email activity. When necessary, these other frames use JavaScript to access the Engine to perform encryption or digital signature operations on the content that they contain.

Authentication

The Role of the Session

Hushmail uses a unique session ID to identify information about the session that is retained on the server. This includes information about whether or not the user is authenticated. The session is created as soon as the user enters Hushmail, and is passed from page to page and frame to frame as part of the URL.

It is important that the session be destroyed when the user leaves the website. Clicking the Quit button can explicitly trigger this, or the Engine can automatically delete the session by accessing a particular URL when the `stop` method of the Engine applet is called. However, automatic session deletion is not necessarily reliable, especially if the browser process is abnormally terminated, so users are encouraged to always explicitly quit their sessions.

Steps

There are two main steps in the Hushmail authentication process.

Private Key Retrieval

When the user initially accesses Hushmail, the Alias, is passed in via HTTP GET or POST. The Engine is loaded in the top left frame. Once the Engine is loaded, the user is prompted to enter the passphrase in an HTML form. Submitting this passphrase results in a JavaScript call to the Engine that activates a private key lookup. See *Hush Encryption Engine White Paper* for details on this process. Note that the passphrase is not submitted to the server, but processed by the Engine.

Authentication with the IMAP Server

Once the private key has been retrieved, the IMAP username, password, and hostname can be retrieved through the Engine interface. (Also see *Hush Encryption Engine White Paper*.) The Engine transparently decrypts the IMAP password using the private key. These values are placed by JavaScript into an HTML form, and posted to the web-server where they are verified by an attempt to log in to the IMAP server, and then they are stored with the session to be used for future IMAP accesses.

Once both the private key and the IMAP credentials have been retrieved and verified, the user is transferred to a page where the IMAP folders and contents of the Inbox are listed, and the email session can proceed.

Account Creation

Account creation follows a four -step process, in which control is transferred between HTML, JavaScript, the Engine, and server-side processes related to the IMAP account. Figure 2 illustrates the process as a flowchart.

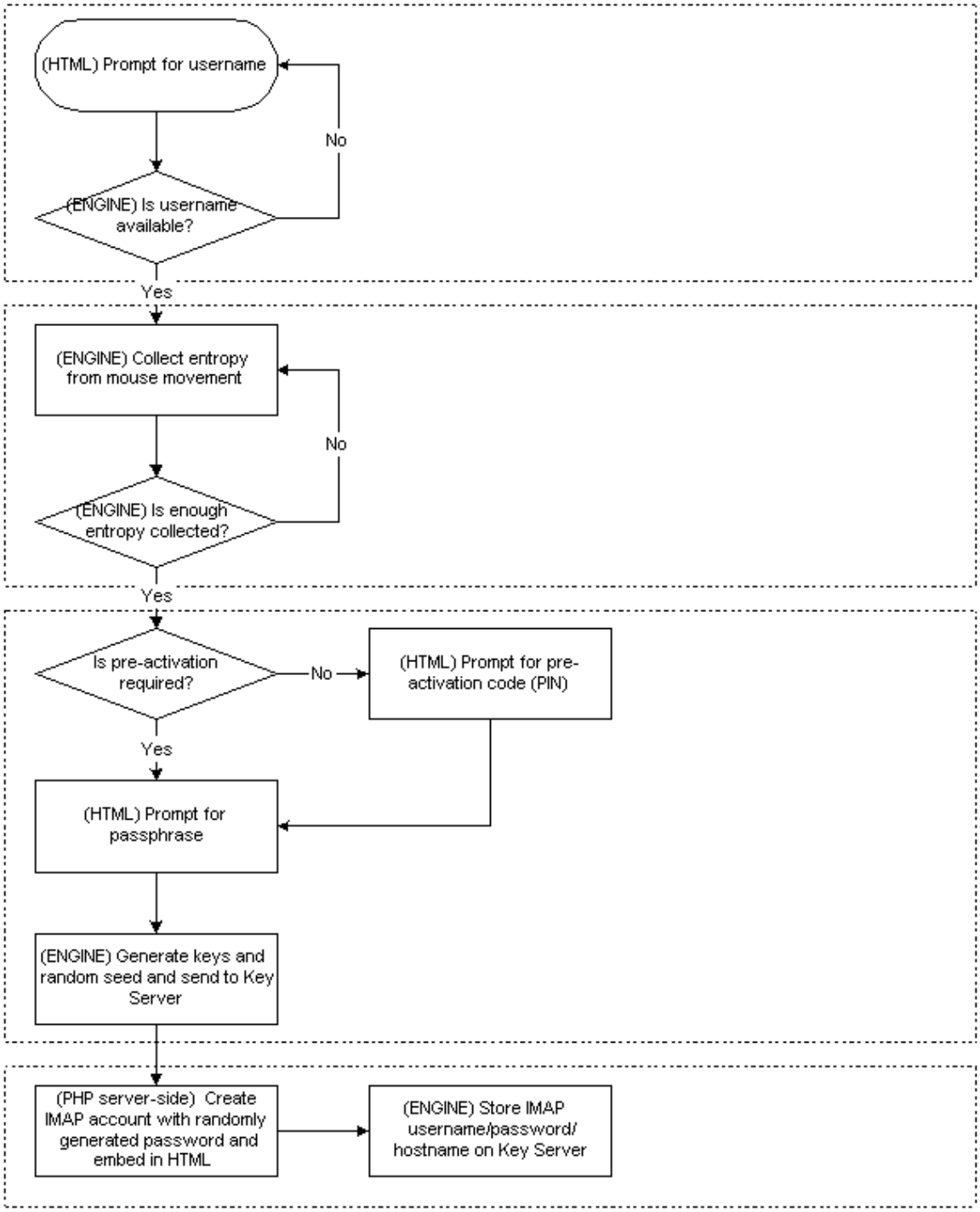

Figure 2

Step by Step Account Creation Process

Administrator Steps

A webmail domain may be configured to require pre-activation.

Reading Email

PHP retrieves a read message from the IMAP server, and writes it out in JavaScript in the HTML response to the client. The client-side JavaScript then interprets the message as encrypted/unencrypted/signed/unsigned, and makes appropriate calls to the Engine (in the other frame). Figure 3 shows the process in a flowchart.

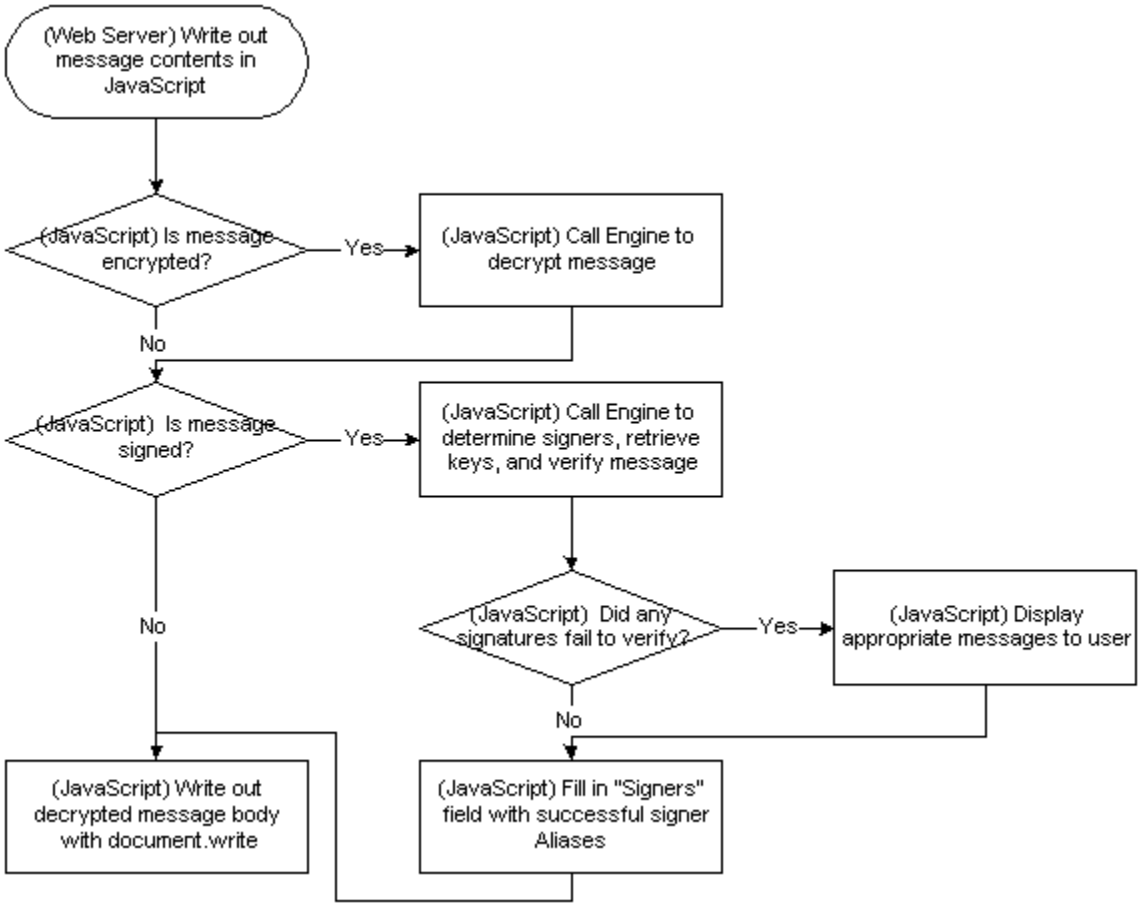

Figure 3

Receiving Attachments

When a message is displayed, links to download each attachment are displayed, and any signatures on the attachments are stored in the JavaScript. Decryption and signature verification on attachments occurs automatically when the user chooses to download the attachment. Figure 4 illustrates the process.

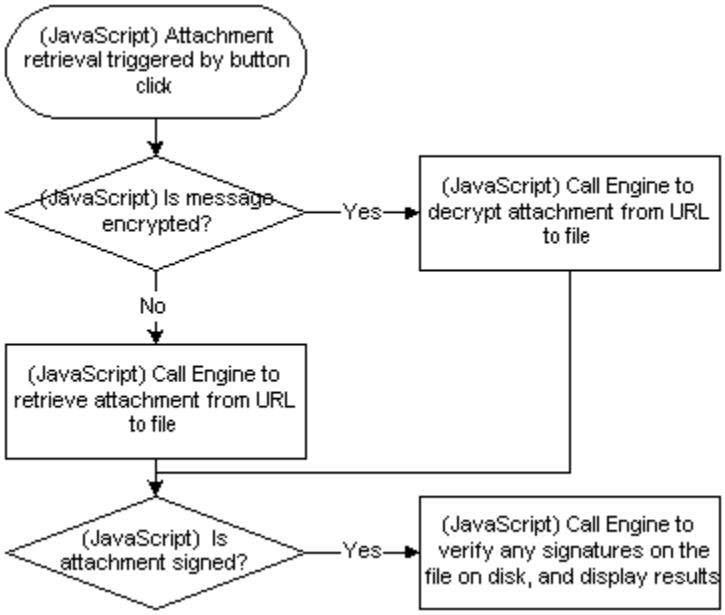

Figure 4

Sending Email

When an email message is sent (or saved) a JavaScript function is triggered which checks to see if encryption or signing of the message is required. Figure 5 shows the process as a flowchart.

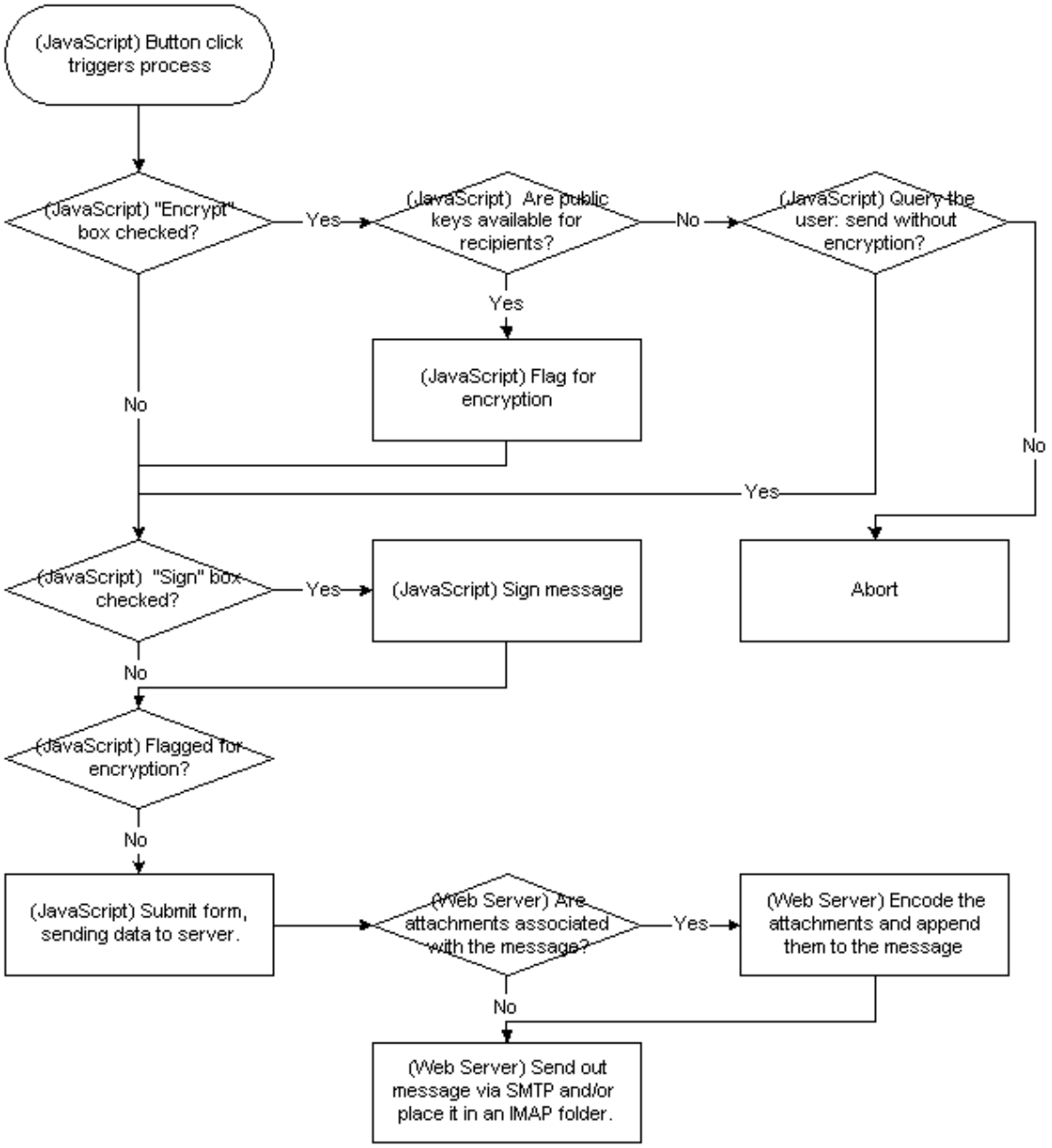

Figure 5

Sending Attachments

When composition of a message begins, the Web Server assigns a unique message identifier, which is then used to associate attachments with the message. Figure 6 illustrates the process of attaching a document to a message.

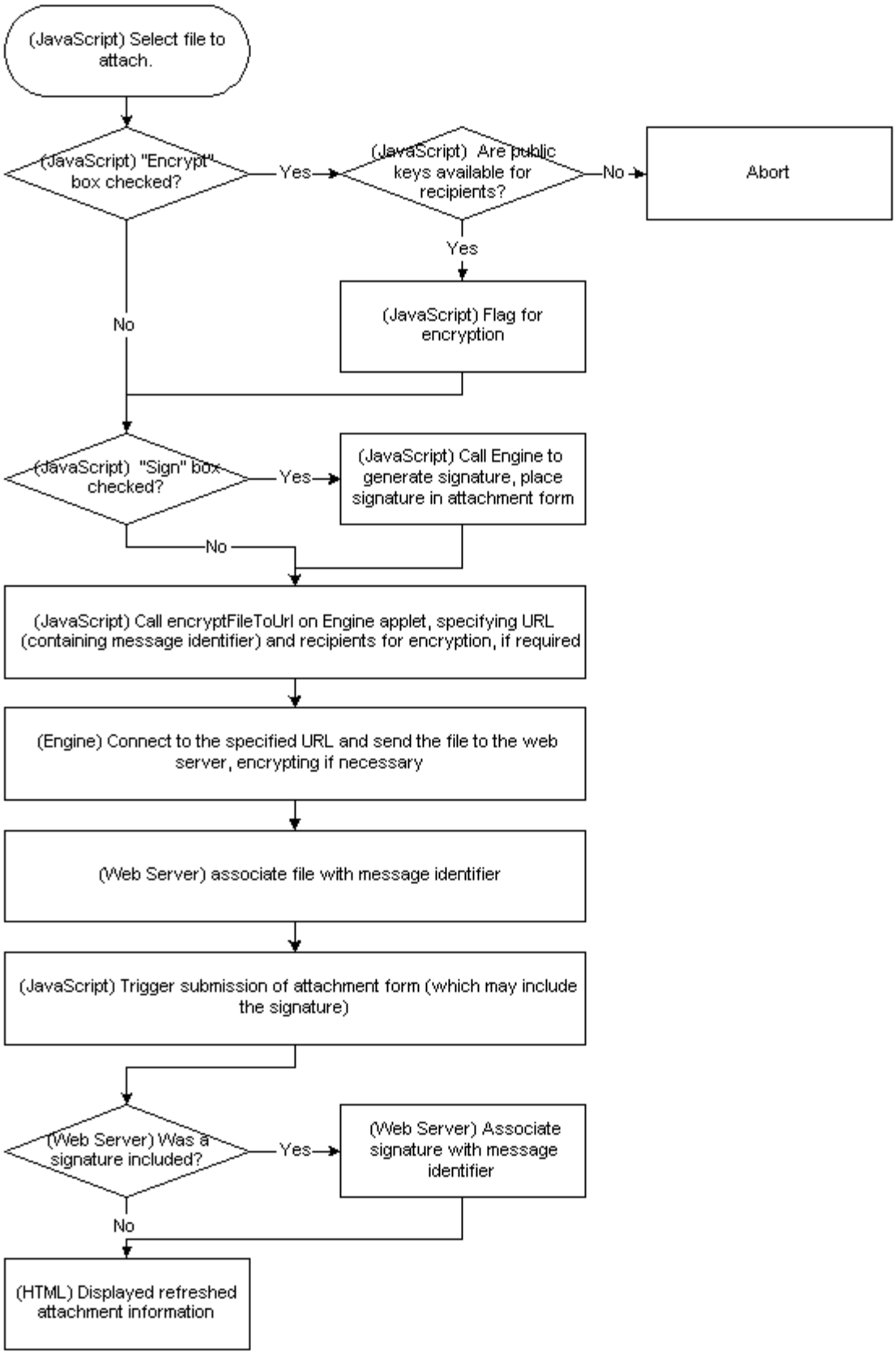
(JavaScript) Select file to attach.
(JavaScript) "Encrypt" box checked?
Yes
(JavaScript) Are public keys available for recipients?
No
Abort
Yes
(JavaScript) Flag for encryption
No
(JavaScript) "Sign" box checked?
Yes
(JavaScript) Call Engine to generate signature, place signature in attachment form
No
(JavaScript) Call encryptFileToUrl on Engine applet, specifying URL (containing message identifier) and recipients for encryption, if required
(Engine) Connect to the specified URL and send the file to the web server, encrypting if necessary
(Web Server) associate file with message identifier
(JavaScript) Trigger submission of attachment form (which may include the signature)
(Web Server) Was a signature included?
Yes
(Web Server) Associate signature with message identifier
No
(HTML) Displayed refreshed attachment information

Figure 6

Security Levels of Various Information

SSL encryption protects all information transferred by Hushmail between web browser and web server. The Engine provides OpenPGP encryption and digital signatures where appropriate. The Engine should not encrypt all data, because that data encrypted by the Engine is inaccessible to the Web Server. For example, email headers are needed to route the message to its destination, and so should not be hidden from the server.

The following table details the encryption and signatures applied to various information transferred by Hushmail.

Feature	**SSL between web browser and web server**	**OpenPGP Encryption**	**OpenPGP Signatures**
Email message bodies	Yes	Yes	Yes
Email headers	Yes	No	No
Email attachments	Yes	Yes	Yes
Stored files (non-attachment)	Yes	Yes	Yes
Contact list	Yes	Yes	No
Read receipts	Yes	No	Yes
Preferences	Yes	No	No