Hushmail

WHITE PAPER

Authentication and Encryption Design

Table of Contents

Introduction

Hushmail provides a range of email services and applications that offer an enhanced level of security, particularly with regard to the use of encryption. This document provides technical information on how authentication and encryption is handled in those services and applications.

The security architecture is described in enough detail for a security analysis to determine whether Hushmail is appropriate for various use-cases. This document can also serve as a starting point for a security audit of the Hushmail system. This document is an overview; as such, specific implementation details, such as information on how the architecture is realized in code, are generally not provided.

This document refers to a simplified model where the term “server” encompasses the web server, data storage, firewalls, etc. In reality, there is a network of servers with elaborate security controls between them. In particular, servers that directly handle connections from the Internet do not store persistent data, and allow access to data on backend systems only through controlled and authenticated channels. The details of the Hushmail server-side architecture are outside the scope of this document.

Applications and Services

Hushmail’s offerings can be broadly grouped into three areas.

2.1 Webmail

Hushmail offers a webmail service, with a mobile-friendly web option. Webmail is accessible at https://www.hushmail.com. Webmail offers server-side OpenPGP encryption.

2.2 Hushmail for iPhone

Hushmail offers an iPhone app. Hushmail for iPhone features client-side OpenPGP encryption.

2.3 IMAP/POP/SMTP Access

When accessing email over IMAP, POP and SMTP using email clients that do not support OpenPGP encryption, OpenPGP encryption and decryption can be performed on the server side.

2.4 Hush Secure Forms

Forms hosted either by Hushmail or on a customer’s website are submitted to the Hushmail server, encrypted using OpenPGP, and delivered to the recipient’s mailbox.

Account Creation

A Hushmail account is built around a unique email address and associated OpenPGP keys.

3.1 Where Account Creation Occurs

Account creation occurs server-side in a web application. The email address, passphrase, and other values needed for account creation are sent from the web browser to the server.

3.2 Determining the Right to Create an Account

Anyone can sign up for an account at hushmail.com, hush.com, hush.ai, hushmail.me or mac.hush.com. There are two modes for determining the right to create an email account on a different email domain hosted by Hushmail.

The simpler mode requires that the customer ID be supplied. The customer ID is a semi-private value associated with a customer who may run one or more email domains.

The more secure mode requires that an account be reserved by an administrator and associate with a reservation code. The reservation code must then be supplied to create the Hushmail account.

3.3 Key Generation

Key generation occurs servers-side in the web application. The user must supply a passphrase before the keys are generated. By default, the keys generated are RSA 2048-bit keys, including a main key and an encryption subkey. The main key is signed with a signature that makes the Hushmail Certificate Authority key a designated revoker.

The private keys, once generated, are stored in a symmetrically encrypted OpenPGP message that is encrypted with the chosen passphrase. The default algorithm for encryption is AES256. The passphrase is processed by OpenPGP S2K conversion with a random salt and an iteration count of 2^{20}. The default digest algorithm used is SHA256.

Once the keys are generated, they are stored on the server, indexed by the full email address with which they are associated. A hashed passphrase value is stored with the keys for future authentication. That hashed value is produced by applying the OpenPGP S2K conversion algorithm with an iteration count of 2^{20} and the full email address plus a newline as salt.

In order to store the keys, the appropriate value must be provided to prove the right to create the account, as described previously. When the public keys are stored on the server, the main key is signed by the Hushmail Certificate Authority.

3.3.1 Passphrase Stretching

The large iteration counts used for private key encryption and generation of the hashed passphrase value provide protection in case the passphrase database or private key database is compromised and an attempt is made to crack passphrases.

3.4 Email Account Creation

When a Hushmail account is initially created, it is assigned an email server where emails will be stored. As soon as emails begin to arrive, either via SMTP or IMAP, they will be stored on that email server.

3.5 Associated Metadata

Various metadata is associated with a Hushmail account. This includes things such as configuration, contacts, and allow and block lists. A particularly important value is the email password. This is a randomly generated password that can be used to access the email store of the account over IMAP or POP without granting access to the decryption keys. Metadata is not encrypted by the user's OpenPGP key.

Two-step Verification

Two-step verification can be set up after account creation.

Two-step verification uses a time-based algorithm[1]. When performing authentication, the user obtains a code that is valid for a limited amount of time and must provide that code to the server.

On enabling two-step verification, the user must choose at least two of three possible methods for receiving a code.

Two methods involve the server sending a code to the user. This may be done by email to an alternate email address, or by SMS. The third method is by an app such as Google Authenticator or Duo Mobile, which can scan a QR code presented to the user and then generate time-based codes as needed.

On setup, each method is validated by requiring the user to present a valid code. The key is reset with each method validation.

After two-step verification is enabled or disabled, all existing user sessions must be reauthenticated.

4.1 Trusted Device Cookie

During any authentication, a trusted device cookie is stored on the client. The unique identifier associated with the cookie is stored on the server and associated with the user account. That cookie is called a "trusted device cookie." Whenever two-step verification is achieved successfully, that trusted cookie is flagged on the server so that two-step verification will not be required again so long as that valid cookie is presented by the client. Trusted device cookies are reset whenever a security event, such as a passphrase change or the enabling or disabling of two-step verification, occurs on the account. An account can only have a limited number of device cookies, and use of more devices will rotate out the oldest cookies. Trusted device cookies expire after one year.

Authentication

5.1 Authentication to the Webmail and Other Web Applications

[1] *https://tools.ietf.org/html/rfc6238*

5.1.1 Confirmation of Correct Passphrase and Access Rights

The browser sends the email address and passphrase to the server. The server determines the right of that email address to access webmail, and verifies that the passphrase is correct by hashing the passphrase according to the definition above and checking it against the stored value.

If the passphrase is correct, the private keys are decrypted with the passphrase and held in memory on the server for future decryption and signing operations. A digital signature of a random value is performed with the private key and checked against the public key.

5.1.2 Two-step Verification

If two-step verification is required, the user must enter a two-step verification code which is sent to the server from the browser. If phone or SMS codes are enabled, the server may send a code to the user by that method.

If a trusted device cookie is present, indicating that two-step verification has previously been performed, two-step verification can be skipped.

5.1.3 Establishment of Authenticated Session

If the above checks pass successfully, then the session is flagged as authenticated for the account, and webmail can be used.

5.2 Authentication from Hushmail for iPhone

Authentication with Hushmail for iPhone involves several steps, performed in the following order.

5.2.1 Confirmation of Correct Passphrase and Access Rights

A hashed passphrase value must be generated in order to verify access rights. The client requests that the server provide information on the mechanism used. This does not require any authentication. Based on that information the hashed passphrase value can be generated.

The email address and the hashed passphrase value, generated according to the definition above, are sent to the server. If the hashed passphrase validates successfully, the response indicates whether the user has permission to use Hushmail for iPhone, and whether two-step verification is required.

5.2.2 Two-step Verification

This step is performed only if needed. If a trusted device cookie is present, indicating that two-step verification has previously been performed, two-step verification can be skipped.

A request is sent to the server to determine if the authenticating account requires two-step

verification. If so, the user can choose from multiple methods to receive the code.

5.2.3 Retrieval of Public and Private Keys for the Authenticating User

The email address and the hashed passphrase value, generated in the same way as the earlier step, are sent to the server. If the hashed passphrase validates successfully, the public and private keys associated with the email account are returned. The client can decrypt and use the private keys.

If two-step verification is enabled, a valid trusted device cookie or two-step verification code is required. Rate limiting is applied independently to checks of the two-step verification code and the hashed passphrase value.

5.2.4 Establishment of an Authenticated API Session

This step cannot proceed unless all previous steps have passed successfully.

The client requests a nonce from the server. The server returns the nonce. The client then digitally signs the nonce and returns it to the server along with the email address to authenticate. If two-step verification is enabled, a valid trusted device cookie or two-step verification code is required. Rate limiting is applied independently to checks of the two-step verification code and the signature.

If the signature verifies successfully, the server marks the session as authenticated with the given email address.

If a correct two-step verification code was sent with the authentication request, a trusted device cookie is returned that can be stored on the device and applied for future authentications.

5.2.5 Settings Retrieval

This step is specific to Hushmail for iPhone.

The client can then request settings specific to Hushmail for iPhone. This includes the email password that can be used for IMAP and SMTP authentication.

5.2.6 Authentication to IMAP and SMTP within Hushmail for iPhone

Using the email password retrieved in a previous step, authenticated IMAP and SMTP sessions can be established as needed.

If the email account is also allowed to send from other email addresses (email aliases and domain forwarded addresses) authentication can be performed for those addresses as well, using the same email password.

If IMAP or SMTP authentication fails, it triggers a connection to the API to retrieve an updated authentication value. Typically, though, this would fail and trigger a full authentication dialog.

5.2.7 Initial Authentication

Authentication is initially performed when an email account is added to Hushmail for iPhone. At that

time, a trusted device identifier is cached so that two-step verification does not need to be done for subsequent logins on the same device.

5.2.8 Routine Reauthentication to the API

Normal reauthentication with the API server occurs as needed whenever the API is used and the existing session is not valid. As long as the trusted device cookie is still valid and the private key has not changed, this is a seamless process. A new authentication nonce is requested from the server, it is signed, and the session continues.

5.3 Standard Authentication to POP/IMAP/SMTP

When the user authenticates to IMAP or POP to retrieve methods, the only supported mechanism is PLAIN over TLS/SSL. Direct SSL/TLS to ports 993 and 995 are supported as well as STARTTLS on ports 143 and 110.

The password used for authentication is the same passphrase that protects the private key. This allows the server to decrypt the private key and retain it in memory for the lifetime of the connection. This allows OpenPGP encrypted emails to be seamlessly decrypted as they are downloaded to the IMAP or POP client.

Note that this decryption of the private key does not occur when Hushmail for iPhone is used as described previously, as in that case the passphrase is not sent to the server.

SMTP authentication is performed using the LOGIN mechanism, and TLS/SSL is required. Direct TLS/SSL to port 465 is supported, along with STARTTLS to ports 25 and 587.

In the case of SMTP, the private key is not decrypted, as there is no need to decrypt any encrypted email.

During IMAP, POP and SMTP authentication, the passphrase is hashed and compared against the hash stored in the the server.

5.3.1 Two-Step Verification

IMAP, POP and SMTP clients do not provide integration with time-based two-factor authentication. Thus, when two-step verification is enabled, POP, IMAP and SMTP authentication requires that a random string, which can be retrieved from the webmail settings screen, be appended to the passphrase in order to authenticate.

5.4 Use of Cookies

In the above authentication processes, cookies are used where applicable to maintain state between requests and responses.

5.5 Rate Limiting

Attempts to access accounts by passphrase are rate-limited to 60 tries in 24 hours by default. Attempts to validate a two-step verification code are limited to 10 tries in 24 hours by default.

Passphrase Management

All passphrase management operations are performed in web applications, and operations on the passphrases are performed on the server.

6.1 Passphrase Change by User

When the user changes the passphrase, a new set of OpenPGP keys is generated. The old keys are decrypted with the old passphrase, and then all the keys, new and old, are encrypted with the new passphrase and stored on the server. All passphrase processing is the same as in the original account creation step.

A revocation signature is placed on each of the old public keys by the Hushmail Certificate Authority. The old keys can no longer be used for authentication by digital signature.

Decryption of messages encrypted with the old keys is performed seamlessly.

When the passphrase is changed, two-step verification must be performed again on all devices.

6.2 Passphrase Recovery and Change by Administrator

Passphrase recovery can be enabled on Hushmail Business domains. In such cases, when a passphrase is created or changed, a line interpolation algorithm is used to split the passphrase into three components, any two of which can recover the passphrase[2].

One component is stored on the server in a database, without OpenPGP encryption. One or both of the other components are stored in designated email accounts on the domain. By default there is a single account, which is the administrator account for the domain. These are stored in OpenPGP encrypted messages. By authenticating to an administrative website, the administrator can combine the passphrase component stored on the server with the passphrase component stored in the encrypted email to retrieve the original passphrase. The rest of the passphrase change is performed as if the user were changing the passphrase.

Email Message Encryption

7.1 Standard OpenPGP Encryption

Email bodies and attachments are encrypted with OpenPGP encryption as described in RFC4880[3].

The format used for MIME messages is based on OpenPGP Partitioned Format[4]. In this format, each MIME part and subpart has OpenPGP encryption applied to it directly.

7.2 Encryption to Hushmail Recipients

---

[2] *https://en.wikipedia.org/wiki/Shamir%27s_Secret_Sharing*
[3] *https://tools.ietf.org/html/rfc6238*
[4] *http://www.ietf.org/mail-archive/web/openpgp/current/msg01811.html*

If both sender and recipient are Hushmail users, the email message is PGP encrypted using the public keys of both the sender and the recipients when it is sent.

The public key is retrieved and verified against the Hushmail Certificate Authority key.

7.2.1 Archive Support

Hushmail supports archive accounts. Archive accounts can be configured on a given email domain by the administrator of the domain.

Whenever a public key is retrieved for a given email address (a sender or a recipient), the public keys for any archive accounts associated with the same domain as that email address are retrieved as well. Any data encrypted with the key for the email address will also be encrypted with the archive key.

When an email for which the sender or recipient domain has an archive account configured passes through a Hushmail SMTP relay, a copy is made and placed in that archive account.

7.2.2 Hush Secure Forms

Each form in Hush Secure Forms is associated with a web address to which it will be submitted by HTTP POST, protected by TLS/SSL encryption. That web address is associated with a Hushmail email account that has a public key. When such a form is submitted to the Hushmail server it is encrypted using the public key associated with that email account, and is delivered to that email account. The email is not digitally signed, as there is no private key available at the time of encryption. One Hushmail email account may have any number of associated forms with unique web addresses.

7.3 Encryption to non-Hushmail Recipients

In the case where a user needs to send encrypted mail to a recipient without a public key, there is the option to apply symmetric OpenPGP encryption.

In this case, the encrypted message is not sent directly to the recipient. It is stored on the server, and the recipient receives an email with a link to a web application that allows the retrieval of the message.

7.3.1 Encryption Process With No Public Key

Some value is considered to be the “answer”. Depending on the context this may be the answer to a security question, or a generated value.

The answer is canonicalized by lowercasing it, and removing whitespace and the following characters: [ ] . , -

A random value called the “answer salt” is generated.

That answer is prefixed by the answer salt followed by a colon, hashed with SHA-256, and hex-encoded high-nybble first. This value is used as input to OpenPGP symmetric encryption. The purpose of the answer salt is so that in cases where the key needs to be cached in memory, such as to facilitate the decryption of multiple attachments, the actual answer need not be cached.

A value called the “answer hash” is computed by passing the above symmetric key through SHA-256 and hex encoding the result. This value is used to verify correct attempts to access the message without actually attempting to perform decryption.

The answer salt and answer hash are stored with the encrypted message on the server. The following methods are used to provide the source of an answer.

• Onboarding message: The answer is generated randomly, stored with the message, and used to decrypt the message whenever the URL that refers to it is accessed. This provides no additional security over storing plain text since the key is stored with the message, but it is used by default as an onboarding process to set up an encrypted communication channel by requiring the recipient to generate OpenPGP keys for future messages.

• Answer to a security question: The sender provides a security question and its answer. The answer is used to encrypt the message. The question is stored on the server along with the message. The recipient sees the question as a password hint, and provides the answer.

• Server-generated answer: The sender need not provide a question or answer. The answer is randomly generated, and sent via encrypted email to a predefined second recipient. The recipient sees a reference code, and contacts the predefined second recipient to retrieve the answer through some pre-established outside channel.

Passwordless message: There is no question, and the answer is generated randomly. In this case, the answer is stored with the message, and used to decrypt the message whenever the URL that refers to it is accessed. This provides no additional security over sending a standard email, but it is commonly used as an onboarding process to set up an encrypted communication channel by requiring the recipient to generate OpenPGP keys for future messages.

7.3.2 Encryption Process With a Public Key

When retrieving an email by any of the above methods, the non-Hushmail recipient can generate an OpenPGP key that will be stored on the server and associated with their email address for future secure communication. This key is generated and stored with the private key protected by a passphrase, using the same process described previously for account creation.

When an email is sent from a Hushmail sender to a non-Hushmail recipient with a public key, the message is OpenPGP encrypted and stored on the server. When the recipient follows the link in the email that they receive, they are prompted for their passphrase, which is used to decrypt their private key and then decrypt the message.

7.3.3 Webmail

In webmail, the encryption operations are performed on the server. The encrypted email message, along with the answer salt, answer hash and other needed values, are stored on the server.

7.3.4 Hushmail for iPhone

In Hushmail for iPhone, the app interacts with the server to determine which operation to perform on the message. When the message is sent, the answer salt, answer hash, and other values that must be stored on the server are included with the email message as message headers, which are processed when the message is stored on the server.

7.3.5 Security Restrictions on Message Retrieval

The ciphertext of the message is never exposed outside of the Hushmail network. Attempts to decrypt the message via web application are limited, and the message is locked after the number of allowed attempts is exceeded.

SSL/TLS

All interactions over the Internet using HTTP, IMAP and POP are protected with TLS/SSL. SMTP traffic is protected whenever possible, although it is still permitted to send emails to and receive emails from servers that don't support SSL/TLS if the customer's configuration allows it.

The Hushmail TLS/SSL configuration supports perfect forward secrecy to help ensure that captured traffic cannot be read if private keys are compromised in the future.

On all websites, HTTP Strict Transport Security is enabled, to help protect users from being maliciously redirected to a non-HTTP site.

SSL Certificates require 2048-bit RSA and SHA-256.

On Hushmail for iPhone, TLS/SSL pinning is used to eliminate reliance on certificate authorities.

Security Models

9.1 OpenPGP Encrypted Data

PGP encryption is applied to the bodies and attachments of email messages sent between recipients that support it. This is the higher level of security in the Hushmail system.

9.1.1 Webmail and IMAP/POP/SMTP Access

For webmail and IMAP/POP/SMTP access, the security model requires that a trusted TLS/SSL connection be established to the server. This secure connection protects data in transit between the user and the server.

The OpenPGP encryption that is performed on the server side uses keys that are unique to each user. There is no global system key for decrypting that data. The messages are decrypted only when they are needed by the user, and the user's passphrase is needed to perform that decryption operation. The passphrase is transmitted to the server when needed over TLS/SSL, but it is not stored on the server. Since the passphrase is not stored on the system and the messages are encrypted, an attacker with only the ability to exfiltrate stored data will not be able to access the messages. This model, however, does not protect against an attacker that is able to compromise the server over a long period of time and alter its functionality, as that will make passphrase interception possible.

Under this model, the user's OpenPGP-encrypted data is best protected if these conditions are met:

1. The user's computer or device is not compromised
2. A secure TLS/SSL connection can be established
3. The attacker is not able to guess the user's passphrase, or the user is protected by two-step verification
4. An attacker is not able to persistently compromise the server

This model protects best against the following threats:

1. Interception of data in transit
2. Exfiltration or loss of data from the server (applies to OpenPGP-encrypted data where the user's passphrase is sufficiently strong)

9.1.2 Hushmail for iPhone

Hushmail for iPhone offers a stronger security model. First, certificate pinning eliminates the necessity to trust certificate authorities. Second, the OpenPGP encryption is applied before the data is transmitted over TLS/SSL to the server, providing better protection against an attack in which the attacker is able to persistently compromise the server. Since the passphrase is never sent to the server, an attacker that has compromised the server will not be able to access it.

Under this model, the user's OpenPGP-encrypted data is best protected if these conditions are met:

1. The user's computer or device is not compromised
2. The attacker is not able to guess the user's passphrase, or the user is protected by two-step verification

This model protects best against the following threats:

1. Interception of data in transit
2. Compromised certificate authority
3. Exfiltration or loss of data from the server (applies to OpenPGP-encrypted data where the user's passphrase is sufficiently strong)
4. Persistent compromise of the server (applies to OpenPGP-encrypted data where the user's passphrase is sufficiently strong)

It is still possible for persistent and complete compromise of the server to result in the interception and decryption of new messages sent by the user, as the attacker could replace the public key used for encryption.

9.1.3 Encryption to non-Hushmail Recipients

The security model for sending messages to non-Hushmail recipients varies depending on the method used to encrypt the message.

In the case where the non-Hushmail recipient has generated OpenPGP keys, the security model is essentially the same as the security model for webmail. See the conditions described in the previous section.

In the case where the answer to a security question is used, the security model is comparable to webmail as well, but it is weakened by the fact that the answer value is likely to be much easier to guess than a passphrase. If a generated answer is used, the security level is higher.

Under this model, the non-Hushmail recipient's OpenPGP-encrypted data is best protected if these conditions are met:

1. The recipient's computer or device is not compromised
2. A secure TLS/SSL connection can be established
3. The attacker is not able to guess the answer value within the number of tries allowed
4. An attacker is not able to persistently compromise the server

This model protects best against the following threats:

1. Interception of data in transit
2. Exfiltration or loss of data from the server, if the answer is automatically generated or sufficiently strong

9.2 Other Data

Data that is stored on the server but is not OpenPGP-encrypted is protected by TLS/SSL encryption while being transferred to the server.

This data is best protected if the following conditions hold:

1. The server is not compromised
2. A secure TLS/SSL connection can be established, requiring a trusted CA

Hushmail for iPhone Local Device Security

In addition to the email protection described above, Hushmail for iPhone provides additional protection against unauthorized access to the data on the local device. This protection is incremental, as a fully compromised device may ultimately be vulnerable. These protections are intended to provide some risk mitigation where practical.

10.1 Data Storage on the Device

10.1.1 File Protection For All App Data

All data files created by Hushmail for iPhone are protected with the flag NSFileProtectionComplete. This means that these files are all encrypted with the device's own protection, such as the device

password, and are not available in unencrypted form when the device is locked or booting.

10.1.2 Data that is OpenPGP Encrypted

In addition to the encryption provided by NSFileProtectionComplete, certain sensitive data is stored OpenPGP encrypted. This includes the bodies and attachments of email messages that were encrypted in the user's email account originally, cached private keys, the passphrases that encrypt those private keys, authenticated session cookies, and the IMAP/SMTP password needed to access the user's email. (Attachments, however, are cached during app use on the file system with NSFileProtectionComplete and no OpenPGP encryption.)

In contrast to the built-in encryption provided by NSFileProtectionComplete, which applies only when the device is locked or powered off, the OpenPGP encryption renders the data unreadable even if the device is unlocked, unless the key to decrypt the data is available. Thus, the data has additional protection, especially when the device is unlocked and Hushmail for iPhone is not running.

The key that provides the encryption for the OpenPGP-encrypted data is the Master Password. That password encrypts the passphrase, the authenticated session cookies, and the IMAP/SMTP password for each account. For each account, the passphrase is needed to decrypt the cached private key which is needed to decrypt OpenPGP-encrypted email message bodies and attachments.

By default, when there is a single email account set up in the app, the Master Password is the passphrase of that single account. If a second account is added, the user is prompted to choose a Master Password.

The Master Password is used as the key for standard OpenPGP symmetric encryption. The OpenPGP S2K algorithm is used to generate the encryption key, with random salt, with SHA256 as the hash, and with an iteration count of 2^{20}. The encryption algorithm used is AES256.

The user may choose to cache the Master Password in the iPhone keychain by enabling Touch ID. If that is done, the Master Password is subject to the protection afforded by the keychain.

The Master Password is stored in the keychain using the kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly and kSecAccessControlTouchIDCurrentSet flags. This means the Master Password will not be stored to iCloud or migrated to new devices. It also means that the Master Password will be wiped if a finger is added to or removed from Touch ID, and that the Master Password will be wiped if the device passcode is reset.

10.2 Locked Mode

Hushmail for iPhone has a locked mode that is independent from the device's locked mode. This provides protection in the case that the user is concerned about another person picking up the device while it is unlocked and browsing encrypted emails.

Hushmail for iPhone enters locked mode if the user explicitly locks the app in the Accounts screen, or if the app is resumed after being idle for 10 minutes.

When Hushmail for iPhone is in locked mode, the user must enter the Master Password to access the app. If Touch ID is enabled, the app may be unlocked that way.