Security Design Review of the Ciphire System

Russ Housley, Vigil Security, LLC
Niels Ferguson, MacFergus BV

27th July 2004

Abstract

Ciphire is an email security system. Like other email security solutions, Ciphire includes provisions to authenticate and encrypt messages. Yet unlike many email security systems available today, Ciphire automates most of the key management, eliminating prevalent problems related to enrollment, certificate distribution, and revocation.

Table of Contents

1 Introduction

Email has become a vital communications link. Nearly everyone uses email today for private and sensitive business communication. Unfortunately, almost all email messages are sent without any form of security. A moderately competent attacker – or anyone who has downloaded the right tools – can intercept and read email messages, modify and delete them, or even generate entirely bogus email messages. Identity theft can happen to your email identity too. This poses a serious threat to all email users – that is, all of us. To date, we have seen relatively few attacks on the email system, other than flooding it with spam. However, the very high value of some messages makes the email system an irresistible target. Email will gain more and more attacker attention in the future.

The Ciphire system is not the first attempt to create cryptographic[1] security protocols to secure email. Previous attempts include PEM[2], PGP[3], and S/MIME[4]. But none of these solutions are widely used, mostly because their implementations impose significant additional burdens on users.

The Ciphire system[5] aims to solve this problem. It uses advanced cryptography to encrypt and authenticate email messages in much the same way as S/MIME and PGP, but it requires far less user interaction, making it much easier to use.

1.1 The user experience

In normal use, Ciphire works seamlessly behind the scenes. Every email message that the user sends is checked. If the recipient participates in the Ciphire system, then the email message is automatically encrypted and authenticated. Only email messages addressed to non-Ciphire users are sent without additional security. This situation is unfortunate, but it is necessary to maintain backward compatibility with the existing email system. However, in this situation, a digital signature may be added based on the user's client configuration.

When a Ciphire user receives a secure email, it is automatically decrypted and the authentication is automatically checked. Since the security processing happens automatically behind the scenes, all the user sees is a small notice indicating that the email message was received securely.

1.2 The Ciphire system

The Ciphire Mail client is cryptographic software that resides between the email client and the email server, automatically intercepting, encrypting and decrypting email communication.

Most Ciphire Mail client operations are performed in the background. By keeping the number of manual operations to a bare minimum, the risk of potential security breaches by incorrect handling are minimized. This approach makes the Ciphire Mail client easy to use; it is nearly transparent.

The interaction of Ciphire with the user's mail applications does not change the look and feel of the user's email environment. Supported email protocols include SMTP[6], POP3[7], IMAP4[8], as well as the Microsoft Exchange and Lotus Notes proprietary email systems.

A Ciphire client requires access to central services such as the certification authority and the directory service. The central infrastructure running these services is provided as a managed service by Ciphire Labs.

Only the Ciphire client has to be installed locally by users. Interaction with the infrastructure, such as creation and downloading of certificates is automated and handled by the Ciphire client.

2 Certificates

Modern cryptographic tools make it easy to encrypt an email message – provided the originator knows the public key of the recipient. Of course, knowing which public key to use is the most difficult problem facing successful deployment of cryptography for email security. Public key cryptography ensures that only a person with the corresponding private key can decrypt the email message. However, if you want to send an encrypted email message to Uncle Bob, you need to know Uncle Bob's public key. Linking public keys to people is very difficult. Many previous attempts to solve this hard problem have resulted in systems that are too difficult to actually use[9]. Either the owner of the private key is faced with an onerous, error-prone enrollment process or message originators have difficulty determining the appropriate public key for a particular email message recipient.

The classic solution to this problem is a public key infrastructure (PKI), and the Ciphire system uses a PKI too. However, the Ciphire system employs several innovations that make the PKI easy and straightforward for all concerned. These innovations include:

- Certificate binds public keys to an identity, which is either an email address, a host name, or a domain name;
- The private key owner controls certificate creation, renewal, and revocation;
- Only one active certificate exists for a specific identity at a given time;
- Certificate can contain multiple public keys, permitting one certificate to support more that one cryptographic system, such as RSA[10], DSA[11], and ElGamal[12];
- A dedicated revocation certificate replaces a certificate when it is revoked; and
- The Ciphire Fingerprint System permits automated certificate validation.

The Ciphire certificate format includes unique features that improve security and certificate location efficiency. These innovations include:

- Multiple certificate issuer signatures;
- Multiple private key owner signatures; and
- Replacement certificates contain pointers to superceded certificates.

2.1 Email addresses

A certificate has a simple function. Given a person that holds a private key, the certificate binds the corresponding public key to the identity of the person that holds the private key. It is surprisingly difficult to reliably issue certificates in a worldwide system. The first problem is naming all of the people unambiguously.

Ciphire avoids the unique name problem by binding public keys to email addresses. No other information about the identity of the private key holder, such as first and last name, is required for the certification process. By using the email address as the only form of identity, the enrollment process is kept simple. Also, the email address is easy for the certification authority to validate since it is used to communicate with the private key owner during the enrollment process.

2.2 Getting a certificate

By getting a Ciphire certificate, the private key owner becomes a secure email user.

An email-based certification process is automated, and it is initiated by the user. After the client software has created all required public/private key pairs, an email message containing a certification signing request (CSR) is sent to the Ciphire Certification Authority (CA). The Ciphire CA responds with an email message containing a challenge; a valid response provides proof that the email address belongs to a person who has access to the private keys that corresponds to the public keys in the CSR. After successful verification of the email address and the response, the Ciphire CA issues a certificate, binding the user's public key and the user's email address together. This message flow is shown in Figure 2.1.

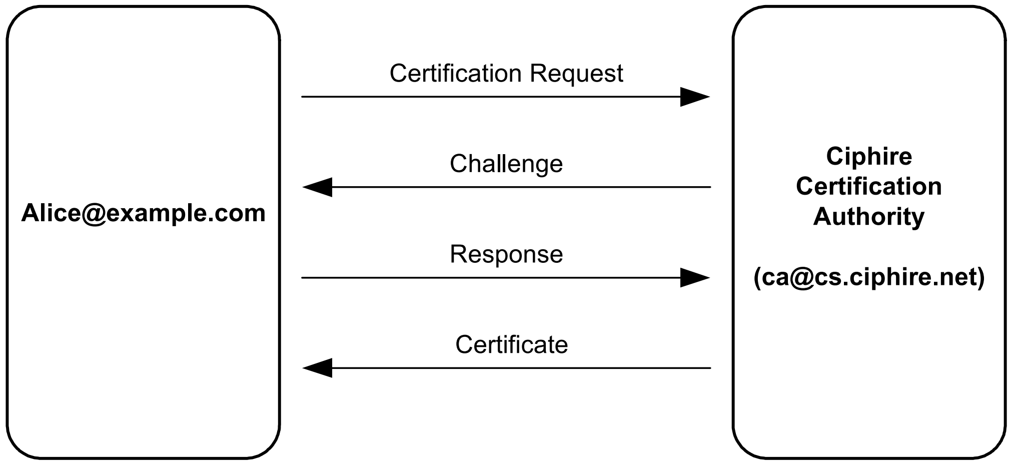

Figure 2.1. Messages exchanged to obtain a certificate.

Unlike other PKI enrollment processes, the user does not need to reveal anything except an email address and the public keys. This is the only information an email message originator needs to send an encrypted email message. Privacy advocates will be pleased that the certification authority does not demand additional information.

This whole process is easy, and most of the steps are automated. It only takes a few minutes. Yet, the Ciphire CA makes sure that impersonation has not taken place. An attacker can easily generate an email message with a bogus source address. To overcome this situation, the Ciphire CA ensures that the requestor can also receive messages sent to that email address. At a minimum, this requires the user to know the password to the email account. At the same time, the response ensures that the user has access to the claimed private keys. The Ciphire client software must be able to perform a private key operation using the appropriate algorithm to generate an acceptable response. In this way, bogus enrollment requests are detected and rejected.

2.3 Finding certificates for other people

As part of the certificate issuing process, the Ciphire CA provides the certificate to the requesting user. More importantly, the certificate is also made available to everyone else by publishing it in a central certificate database. This database is called the Ciphire Certificate Directory (CCD). Certificates are usually retrieved based on email address or a unique certificate identifier (CID).

Whenever a message originator needs a public key for a specific email address, the corresponding certificate is automatically retrieved from the CCD. In the case of a digitally signed message, the certificate is retrieved from the CCD based on its CID. To ensure that the CCD can serve a huge number of clients, access is provided by a network of proxy servers. This is similar to the manner in which large web sites provide their content to many browsers simultaneously. Performance is increased by each proxy server caching responses from the CCD for a period of time.

2.4 Validating certificates

In a conventional PKI, users must have confidence in trust anchors, which are sometimes called Root Certification Authorities. Typically, several trust anchors are configured. A certificate issued by the trust anchor is validated by verifying the digital signature with the trust anchor's public key. For the overall system to be secure, all of the trust anchor operators must be well behaved.

To avoid this complex trust relationship in the Ciphire system, certification actions are publicly auditable. The Ciphire Fingerprint System (see chapter 7 for further details) uses hash values and hash chaining techniques[13] to create a log that is easily verified by any Ciphire client. Hash chaining is a method for cryptographically linking data; it ensures that previous log entries cannot be changed without invalidating later ones. In this way, all Ciphire CA actions are open to scrutiny by all Ciphire clients.

2.5 Revocation

When a particular email address is no longer being used or when a user's private key is lost or disclosed, the associated certificate needs to be revoked. By revoking the certificate, other users are notified that the binding between the email address and the public key is no longer valid.

The Ciphire system provides the user with two revocation choices: normal revocation and emergency revocation. If a user chooses normal revocation, the client software creates an email message containing a revocation request and sends it to the Ciphire CA. This request is digitally signed by the private key associated with the to-be-revoked certificate. If the private key is lost or otherwise unavailable, an emergency revocation can be performed. The emergency revocation system uses the World Wide Web, not email. A web browser is required. Authorization is based on the email address in the certificate and the user's previously chosen revocation passphrase. One can think of the email address as a web site user name.

After processing the revocation request, the Ciphire CA issues either a revocation certificate or an emergency revocation certificate. From this point, attempts to fetch a certificate for that user from the CCD will result in the revocation certificate or emergency revocation certificate, which tells the user that a valid public key for that particular email address is not available.

2.6 Updating keys

Once created, public/private key pairs are commonly used for a long time – often for several years. However, it is advisable to replace a public/private key pair more often, especially when there is not continuous control over the computer where the private keys are stored. The Ciphire system allows a user to update the public/private key pairs and corresponding certificate at any time. Updates can be performed manually or automatically at a user-specified update interval.

In the Ciphire system, the public/private key pair and certificate replacement process is called renewal. Certificate renewal is accomplished by a combination of a normal revocation request, public/private key generation, and certification request creation mechanisms. During renewal the old certificate is revoked, new public/private key pairs are created, and a new certificate is issued.

2.7 Gateways

The Ciphire system can employ an encryption gateway, where an email server takes care of all encryption and decryption tasks. This approach is especially popular in an enterprise because the email server administrator can set policies that apply to all of the enterprise users. Server configuration is much easier than adjusting the configuration of each email client in the enterprise. However, there is a significant limitation that prevents server-based email encryption solutions from becoming a complete security solution. While administration is significantly easier, messages sent between the email client and the server are not encrypted. Within a secure enterprise environment, the communication path between the email client and the server is probably adequately protected. But when a user is traveling and checking email from a hotel room, the communication path is vulnerable unless other security techniques are employed, such as a virtual private network (VPN).

Gateways act on behalf of every user that has an account on the email server. To support this environment, the Ciphire system uses a special gateway certificate that binds the gateway's public key a host name or a domain name. In this way, a single certificate represents all of the user accounts on the email server, which could be an entire enterprise or a department within an enterprise.

Example host names and domain names include:

- example.com
- hamburg.example.com
- sales.example.com

3 Certificate details

The previous chapter looked at Ciphire certificates. Recall that a certificate binds together the Ciphire user's public key and email address. The digital signatures on the certificate provide confidence that the certificate has not been altered and that its contents are correct. The process used to generate the certificate ensures that the email address belongs to the user who has access to the private keys that correspond to the public keys in the certificate. This chapter builds on these foundational concepts, discussing several important details.

3.1 Email address

When a certificate is issued for a single user, the certificate contains that user's email address. No other information about the identity of the private key holder, such as first and last name, is required for the certification process. Use of the user's email address is the typical situation, but this identifier does not support a gateway solution. In this case, the identity in the certificate is the fully-qualified host name or the domain name supported by the gateway.

Unlike other certificate-based systems, the directory within the Ciphire system ensures that only one certificate at a time is active for any given identity. The CCD provides the active certificate to requestors. For efficiency, the Ciphire client software caches certificates. This permits a small period of time where a message originator uses an "old" certificate. This does not cause a problem because the corresponding private keys remain available to the recipient. Further, the next time the client interacts with the CCD, the client software can obtain the revocation certificate associated with the "old" certificate as well as the valid "new" certificate. This facilitates a graceful transition from an "old" certificate to a "new" one. This is a routine transition that occurs as a result of certificate renewal.

When more than one person has access to one email account, it is not possible for the Ciphire CA to distinguish the normal user normal from anyone else that can access the account. A request from any such users will be honored. However, the normal user will learn about the previous certificate as soon as they try to obtain a certificate.

3.2 Multiple public keys

Ciphire certificates may contain multiple public keys. Thus, a single certificate can provide different types of public keys as well as different sizes for public keys of the same type. Each public key in the certificate is flagged with specific purposes, such as digital signing, authentication, and encryption.

By default, every Ciphire certificate contains three 2048-bit public keys:

- an RSA public key for digital signing, authentication, and encryption;
- a DSA public key for digital signing and authentication; and
- an ElGamal public key for encryption.

3.3 Ciphire certificate signatures

All Ciphire certificates are signed by two parties: the Ciphire CA and the owner of the certificate.

The Ciphire CA issues all certificates. Prior to signing the requested certificate, the challenge/response described in the previous chapter is used to validate the email address and confirm that the requesting user has access to the necessary private keys. Once the email address and private key access are confirmed, the Ciphire CA digitally signs the certificate with all of the signing keys in the certification authority certificate.

The Ciphire CA certificate is known to all Ciphire clients; after all, it has already signed their certificates. The Ciphire CA signatures of each certificate allow a Ciphire client to verify that the certificate of another user has gone through the same validation process and that it has not been tampered. Even the owner of the certificate, who has access to the private keys that correspond to the public keys in the certificate, does not have the capability to change the certificate content once it has been issued.

In addition to the Ciphire CA signatures, every Ciphire certificate also contains self-signatures. These signatures are generated with the signing keys of the certificate owner. The self-signatures of each certificate allow each Ciphire client to verify that the owner of the certificate participated in the certificate generation process and that the certificate has not been tampered with.

Since a Ciphire certificate contains the signatures of the Ciphire CA and the owner of the certificate, both parties must collaborate to issue a certificate. This technical measure increases the confidence Ciphire certificates over traditional certificates.

3.4 Revocation

A Ciphire certificate and the public keys that are contained in the Ciphire certificate are revoked by issuing a special revocation certificate. Except for the certificate type, a revocation certificate is identical to the original Ciphire certificate. Also, the validity period is adjusted to indicate the date and time of the revocation. The certificate owner begins the revocation process. Revocation may be performed, if the email address is no longer in use or if the user suspects that his private keys have been compromised. As with the original Ciphire certificate, the Ciphire CA and the owner of the certificate must collaborate to issue a revocation certificate.

Since a self-signature is needed on the revocation certificate, performing a revocation requires use of the corresponding private keys. If these private keys are lost, an emergency revocation must be performed to revoke the original Ciphire certificate. Emergency revocation only requires the user to submit the emergency revocation passphrase and email address to the Ciphire CA via an SSL/TLS-enabled web browser. This can be accomplished without the Ciphire client software.

On the emergency revocation web site, the user enters the email address associated with his certificate. The emergency revocation system uses the CCD to locate the appropriate certificate. Then, the user enters the passphrase to confirm the requested revocation. The passphrase is used to generate the key needed to decrypt the revocation certificate signing request (CSR) that was stored in the emergency revocation system when the certificate was originally created. The decrypted CSR is sent to the Ciphire CA and the revocation certificate is created and posted in the CCD.

3.5 Renewal

The Ciphire client software automatically renews Ciphire certificates at regular intervals. However, a user may instruct the Ciphire client software to renew his certificate at any time. This might be done if the user suspects a compromise of his private keys since new public/private key pairs are generated as part of the renewal process.

The certificate renewal process is simply a combination of the Ciphire certificate creation and revocation processes performed at the same time. Thus, new public/private key pairs and a new certificate are created, and the old certificate is revoked.

Certificates created via a renewal include additional security properties: successor signatures in the revocation certificate and certificate chain references in the new certificate.

3.5.1 Successor signatures

Revocation certificates can include successor signatures, which are very similar to self-signatures. Successor signatures are created only if the revocation certificate is created as part of certificate renewal. The signing private keys associated with the new certificate are used to create the successor signatures, providing additional authentication for the revocation certificate. These signatures can be validated with the public keys contained in the new certificate, providing confidence that the new certificate is the intended replacement for the revoked certificate.

3.5.2 Certificate chain

When certificate renewal is used to create a new certificate, it includes a reference to the revocation certificate for the old certificate. The successor signature can be thought of as a forward pointer; this reference is the corresponding backward pointer. As a reference, the serial number of the certificate is used. By traversing the set of references, the result is a chain of all certificates for a particular email address.

3.6 Validity

The Ciphire certificates include a validity period, which is the time interval during which the certificate is intended to be used. This includes two date and time values. The first one indicates the date on which the certificate validity period begins, and the date on which the certificate validity period ends.

4 Certificate data structure

Figure 4-1 shows the Ciphire certificate data structure. This chapter provides a brief description for each of the fields in the structure. The field names and syntax are similar to X.509 certificates[14]. The context for the brief description is provided in the preceding chapters.

Certificate Field Name	Purpose of the Certificate Field
Certificate	The Ciphire Certificate.
tbsCertificate	The part of the certificate that is digitally signed by the Ciphire CA.
version	Identifies the certificate syntax, and allows for future changes.
serialNumber	An integer assigned by the Ciphire CA to uniquely identify the certificate.
issuer	The name of the Ciphire CA.
subject	The email address of the Ciphire user; or host or domain name of Gateway.
validity	The notBefore and notAfter dates assigned by the Ciphire CA.
userValidity	The notBefore and notAfter dates assigned by the certificate owner.
certificateTypes	Distinguishes certificates and revocation certificates.
certificateChain	Identify predecessor and successor certificates.
subjectPublicKeysInfo	The collection of public keys in the certificate.
subjectPublicKey	The information about one public key. (Multiple.)
algorithmIdentifier	The algorithm associated with the public key, such as RSA, DSA, and ElGamal.
key	The public key.
keyUsage	The acceptable uses of the key, such as encryption and digital signature.
daughterPublicKeysInfo	The collection of associated certificates. (Not supported yet.)
successorSignatures	The collection of successor signatures.
selfSignatures	The collection of user certificate signatures.
signatures	The collection of Ciphire CA certificate signatures.

Figure 4.1. Ciphire Certificate Structure

Each of the fields shown in Figure 4-1 is described briefly in the following paragraphs.

Certificate is the entire Ciphire certificate. It has two primary parts. The first part contains the information that is signed by the Ciphire CA, and the second part contains the Ciphire CA signatures.

tbsCertificate is the portion of the certificate that is signed by the Ciphire CA. It is comprised of the version, serialNumber, issuer, subject, validity, userValidity, certificateTypes, certificateChain, subjectPublicKeysInfo, daughterPublicKeysInfo, successorSignatures, and selfSignatures fields.

version is the version number of the Ciphire certificate syntax. This document describes version 1. This field allows future versions to be specified without causing interoperability problems.

serialNumber is the unique identifier assigned by the Ciphire CA. Each Ciphire certificate is assigned a unique integer value before the Cipher CA signs it.

issuer is the name of the Ciphire CA. The email address of the Ciphire CA is ca@cs.ciphire.net.

subject is the email address of the Ciphire user, the host name or the domain name (Fully-Qualified Domain Name, FQDN) of a mail gateway. Subject names have two components: common name (CN) and email address (E). They are used as shown in Table 4-1.

Certificate Subject	**Common Name**	**Email Address**
Single User	CN=<Email address>	E=<Email Address>
Mail Gateway	CN=@<FQDN>	E=postmaster@<FQDN>

Table 4-1. Common name and email address conventions.

validity contains the notBefore and notAfter dates assigned by the Ciphire CA. The notBefore date, which is certificate creation date, is also used by clients to locate the appropriate fingerprint list for the certificate. The dates listed in this field and the corresponding dates listed in the userValidity field must not differ by more than 24 hours. The only exception is the notAfter date in emergency revocation certificates.

userValidity contains the notBefore and notAfter dates assigned by the Ciphire user. The certificate signing request (CSR) communicates these dates from the Ciphire client software to the Ciphire CA. This field, not the validity field, is covered by the self-signature since Ciphire CA generates the validity field after the self-signature is generated. Also, the Ciphire Mail client software uses this validity period, not the one in the Ciphire CA assigned validity field, when making use of a certificate.

certificateTypes is a list of identifiers that indicate the type of the Ciphire certificate. The types include user, mail server, host, gateway, and proxy certificates. Further, a dedicated identifier is used in authority certificates, such as the Ciphire CA or Ciphire Root CA. For revocation certificates, a revocation identifier is added to the list of identifiers that appears in the original certificate. The certificateTypes field tells clients how the certificate ought to be used.

certificateChain contains a predecessor certificate identifier (CID) and a successor CID if the Ciphire certificate was generated as a result of certificate renewal. If the successor CID is present, the successorSignatures field must also be present, which only occurs in revocation certificates.

subjectPublicKeysInfo contains a collection of entries, one for each public key in the certificate.

subjectPublicKey contains the information about one public key. It is comprised of the algorithmIdentifier, key, and keyUsage fields.

algorithmIdentifier indicates the algorithm associated with the public key. RSA, DSA, and ElGamal are currently supported. Other asymmetric algorithms are easily accommodated.

key is the public key.

keyUsage is the manner in which the public key can be used. One or more of the following flags is set. The *encrypt* flag is set if the public key can be used for data encryption. The *decrypt* flag is set if the public key can be used for data decryption. The *authenticate* flag is set if the public key can be used to verify authentication. The *sign* flag is set if the public key can be used to verify a digital signature. The *certselfsign* flag is set if the public key can be used to verify certificate self-signatures. The *certsign* flag is set if the public key can be used to verify certificate signatures.

daughterPublicKeysInfo is not presently implemented. When it is implemented, it will contain references to OpenPGP and X.509 certificates for the same user.

successorSignatures is a collection of successor signatures, which are only present in revocation certificates. The successor CID in the certificateChain field is used to obtain the certificate needed to validate these signatures. The successorSignatures field contains a signatureAlgorithmIdentifier, which names the digital signature algorithm used. It is an index, which identifies the public key within the successor certificate that is needed to validate the signature, and the signature value.

selfSignatures is a collection of self-signatures. The self-signatures are generated over the following certificate fields: version, serialNumber, issuer, subject, userValidity, certificateType, certificateChain, subjectPublicKeysInfo, daughterPublicKeysInfo, and successorSignatures fields. The selfSignatures field contains a signatureAlgorithmIdentifier, which names the digital signature algorithm used. It is an index, which identifies the public key within the certificate that is needed to validate the signature, and the signature value. The Ciphire CA provides the serialNumber to the client software during certificate enrollment as part of the challenge, and the self-signature is returned as part of the response. In this way, the self-signature is bound to the certificate in which it appears.

signatures contains the Ciphire CA signatures.

5 Securing Email Messages

5.1 Processing email messages

As mentioned in the introduction, the Ciphire client sits between the mail client (mail user agent, MUA) and the mail server (mail transfer agent, MTA). The Ciphire client transparently processes the communication (SMTP, POP3, IMAP4) between the MUA and MTA. Direct integration with an MUA is therefore not required, except in Microsoft Outlook/Exchange (MAPI) and Lotus Notes, where the appropriate plug-ins are provided.

This approach allows the user to continue using their preferred MUA. Also, the user does not need access to recipient certificates at the time an email message is created, but only at the time the message is sent, which naturally involves being online. Internet access to obtain recipient certificates from the CCD is only needed at the time when email messages are actually being sent and received. At first view this sounds like a minor difference, but it has a great impact on the user. With many other secure email applications, an email message cannot be encrypted (or marked for encryption prior to sending) until the recipient’s certificate is fetched. There are situations where network access is unavailable, for example, when a user creates an email message during an airplane flight. The Ciphire client avoids these problems by processing email messages as they are transmitted to (or received from) a mail server. These benefits outweigh the possibility of a slightly noticeable delay when the user transfers many email messages or a very large email message.

As a result of this architectural choice, the decision to sign or encrypt a particular email message, as well as which keys are used, is made at the time the email message is transmitted to the mail server. There are no additional *Encrypt* or *Sign* buttons the user selects when composing an email message. Instead, the user controls the behavior of the Ciphire client through the configuration of security options.

5.2 Encryption and Signing Options

When a Ciphire user is sending an email message, the Ciphire client automatically asks the CCD if an active certificate is available for the recipient's email address. If so, the certificate is validated and the public keys are used to encrypt the message. Otherwise, the message may be sent without encryption.

The user configures the Ciphire client to select the cases when a signature is applied.

Encryption and signing options are globally defined. To allow further refinement, a user may configure recipient settings for specific email addresses, host names and domain names.

5.2.1 Encryption Options

- Refuse unencrypted: If a recipient has no Ciphire encryption capabilities, the mail message is not sent and an error message is returned to the MUA.
- Warn if unencrypted: If a recipient has no Ciphire encryption capabilities, the user is asked via a pop-up message whether the message should be sent unencrypted or not. If the user stops the transmission, an error message is returned to the MUA.
- Try to encrypt: If a recipient has no Ciphire encryption capabilities, the message is sent unencrypted.
- Never encrypt: This turns of the encryption functionality of the Ciphire client and all email messages are sent unencrypted.

5.2.2 Signing Options

- Always sign: All email messages, whether encrypted or unencrypted are digitally signed.
- Never sign: This turns of the signing functionality of the Ciphire client.

5.2.3 Overriding Encryption and Signing Options

In addition to the configured encryption and signing options, advanced users can override the normal configuration setting for a particular email message. This is done by annotating the email subject with short commands that override a particular setting. For example, to force an email message to be signed, the subject is prefixed with the command 's!'.

The Ciphire client supports the following override commands:

- **s!** sign (reject message if signing is not possible)
- **n!** do not sign message
- **e!** encrypt message (reject message if encryption for recipient is not possible)
- **u!** do not encrypt message

The signature-related commands can be combined with the encryption-related commands. For example, to force the sending of an unencrypted and unsigned email message, the command "un!" is used. Also, when requesting encryption of an email message, the command "ef!" can be used to force a CCD lookup, ignoring any certificates that might be in the Ciphire client's certificate cache.

The Ciphire client will strip the override command out of the subject before sending the email message. A recipient of such a message will not be able to determine that override commands were used.

5.2.4 Multiple recipients

When sending mail to a group of recipients, the Ciphire client does one of the following:

- creates a dedicated (encrypted) email message for each recipient that has Ciphire encryption capabilities;
- creates a dedicated (encrypted) email message for each Ciphire recipient, and sends a single plaintext message to the recipients that do not have Ciphire encryption capabilities; or
- sends a single plaintext message, if any of the recipients does not have Ciphire encryption capabilities.

5.3 Encrypting an email message

When the Ciphire client finds a valid certificate for a recipient, the public keys marked with the *encrypt* keyUsage flag are used to encrypt the message for the recipient.

With every message the Ciphire client can create multiple layers of encryption, each using different asymmetric and symmetric algorithms. By default, two encryption layers are created using the 2048-bit RSA key (inner layer) and the 2048-bit ElGamal key (outer layer) from the Ciphire certificate. The inner layer is symmetrically encrypted using AES[15] in CBC-HMAC[16] mode and the outer layer is symmetrically encrypted using Twofish[17] in CCM[18] mode. For both algorithms a dedicated 256-bit session key is used.

5.3.1 Message Format

The complete content of an email message, including its headers such as Subject, From and To, is encrypted to create a secure email message[19]. The encrypted contents are encapsulated and Base64 encoded.

Even though the From and To headers are part of the encrypted contents, the envelope addresses from the original message are used. Mail servers need to know the address of the intended email message recipient as well as the address of the sender in case the message cannot be delivered.

To support mail clients that allow a user to retrieve only headers of an email message before retrieving the complete message, the secure email message contains an encrypted Subject. This allows a receiving Ciphire client to decrypt the Subject separately from the rest of the secure email message, allowing MUAs to properly display subject information.

The following is an excerpt of a format structure from a secured email message:

```
Subject: uAIAAAAAAAACAb0AAAQCgnM+OnFVOixoYioYQ12z5oUyU1UYWf3Bcp4N
Zx5AAgAAcd8Baf4dpqQinAai+hcIiZFMwl/sphy/a29/WtFnziFnNP5GR4LyNUhk
[... several lines of encryption ...]
WnNOKrYsItF4g1YXScEVJ8N5WclqepcEtpbrM1kQ7ORNGFqfNlQBOC+ig+3dNKP1
Nzgc62Uo6xw=
To: bob@example.com
From: alice@example.com

-----Begin Ciphire Message-----
uAIAAAAAAAACAb0AAAEAAQAAAQCgnM+OnFVOixoYioYQ12z5oUyU1UYWf3Bcp4N2
Zx5AAgAAcd8Baf4dpqqinAai+hcIiZFMwl/sphy/a29/WtFnziFnNP5GR4LyNUhk
[... several lines of encryption ...]
BvdJH+fruZn4Hj5OnzFUYOhiYIlI8pFAhj0AM7Z51TvfWbXWvJsJuAh8cnTYEBF4
ojrNbcH4efVUDFHvwenezdWoEEToLQde2cu19ZznGpnEUtOAJsoLCA==
-----End Ciphire Message-----
```

A receiving Ciphire client decrypts the message and restores the original message. Headers that have been added while the message was in transit, such as Received headers are merged into the header of the decrypted message.

5.4 Digitally signing an email message

The Ciphire system uses a unique approach when signing an email message. It is different from other secure email technologies. Ciphire does not distinguish between signing a plaintext message versus signing and encrypting a message. Like most other approaches, if a message is to be signed and encrypted, it is first signed as a plaintext message and then encrypted. The body and, if present, other MIME[20] parts such as attachments, are each signed individually.

The first text/plain and text/html MIME part of an email message are signed with an inline signature, and one additional MIME part is created to carry the signatures of all other MIME parts.

Each MIME part is individually signed, which allows the signature on each attachment to be checked, even if one attachment is missing or corrupted. This can happen when anti-virus scanners check attachments. Anti-virus scanning could remove an attachment, perhaps based on its MIME type or file name. Anti-virus scanning can also modify an attachment, perhaps to remove macros from a Microsoft Word document.

The following is an example of a signature for an attached document:

```
---------------------[ Ciphire Signature ]----------------------
From: alice@example.com signed "contract.pdf" (664223 bytes)
Date: on 27 July 2004 at 13:23:42 GMT
To:   bob@example.com, carl@example.net
----------------------------------------------------------------
```

```
00fAAAAAEAAAD+WZ9ABAEAALgCAAIAAgACACAxtN4blwNOgpZbT2j9Gm84OPspAO
COTr17U+wzYy8P7QEAd64CrcECnu6qeOQRHlgGd+wrPwq99XEn/3sgO4Twmnpztu
q1wP6ioxV5kn3WLy7lMWdTx2IvlVujFeifEe18/A==
-----------------[ End Ciphire Signed Message ]----------------
```

The From, To, and Date lines in the signature representation are part of the actual signature, though the header of the email message itself is not included in any of the signatures.

5.4.1 Signature Chaining

If a Ciphire user forwards a signed message, the Ciphire client verifies the old signatures and leaves valid signatures in the message. The forwarded message, including its original signatures, is then signed again when it is sent to the new recipient.

5.5 Authenticating an email message

Signing outgoing email messages allows recipients to verify the origin of the message, and it allows the recipient to identify forged email messages, which has unfortunately become very common practice. A digital signature effectively authenticates an email message, and digital signatures also allow the recipient to prove to a third party that a particular message and its content came from a particular originator. This property is often referred to as “non-repudiation” since it prevents the originator from denying having sent the message. In many jurisdictions the legal standing of such signatures is unknown. Even though such a digital signature may not be legally binding, there are cases in which the sender may find it undesirable to create such a signature.

With encryption solutions using PGP or S/MIME, an unsigned email message allows an attacker to forge the originator’s identity even if the message is encrypted. The recipient cannot easily detect the change in the originator. However, in the Ciphire system, encryption includes authentication information. The session key used to encrypt the email message is digitally signed by the sender for every layer of encryption.

This is quite different from a traditional signature as only the session key is signed. The recipient has proof that the session key came from the originator, but the recipient has no proof that can be shared with a third party about the origin of the contents. However, the recipient can be assured who sent the message; apart from the recipient itself, the sender is the only party that knows the session key.

The receiving Ciphire client automatically validates the signature on the session key to authenticate the source of the encrypted message. If it is invalid or missing, the message is treated as invalid, and it is not decrypted. The client requires the signer’s certificate in order to verify the signature on the session key. If the client was not able to retrieve the certificate, perhaps due to network problems, the email message is still decrypted, but a warning is displayed to the user.

5.6 Using a gateway

Centralized solutions where an encryption application is deployed on a mail gateway are becoming more and more common. The gateway performs all cryptographic functions, avoiding the need for any client installations. However, centralized solutions are less secure than individual client solutions. The gateway has access to all private keys of all users for which it handles email (which is bad in its own right) and thus, becomes an attractive target for attackers.

The Ciphire system can be used on a central mail gateway, either by itself or in combination with individual client installations. On the gateway, the Ciphire system typically uses certificates where the private key holder is identified by a host or domain name. In this case, a sending Ciphire client will not find a certificate for a particular email address, for example, bob@example.net. Instead, it finds a mail gateway certificate for the group of users that share the mail domain, such as @example.net.

Using the Ciphire system in conjunction with individual client software provides enhanced security. Consider the example where Alice (alice@example.com) is sending an email message to Bob (bob@example.net). The following scenarios are possible:

1. Bob has an individual certificate (and individual client software), and he is also using a mail gateway (@example.net).
   Alice' Ciphire client encrypts the message using the public keys from Bob's certificate (bob@example.net). In addition, the certificate for @example.net is used to add an additional layer of encryption for the mail gateway. The receiving Ciphire mail gateway decrypts the outer layer and forwards the message to Bob, where his Ciphire client performs the final decryption.
2. Bob and Alice both have individual certificates (and individual client software) and both are using their own mail gateways, which have an appropriate certificates for their domains.
   Alice's Ciphire client encrypts the message for bob@example.net and submits it to her mail gateway. The mail gateway adds another layer of encryption for Bob's mail gateway (@example.net). The receiving Ciphire mail gateway decrypts the outer layer and forwards the message to Bob, where his Ciphire client performs the final decryption.

In the second scenario, an attacker who is eavesdropping on the communication between the two mail gateways will not be able to determine that Alice is sending an email message to Bob. Instead, the attacker can only observe that one mail gateway is sending a message to the other mail gateway. This configuration provides some protection against traffic analysis.

5.7 Time-Stamping

Validity times in certificates are very important. Enforcing a validity period is impossible, if the user can use the wrong date and time setting on his computer, which is an easy attack known to almost all computer users.

A correct date and time setting is important to ensure that:

- replay attacks are not possible (e.g., when communicating with a proxy);
- a signature contains a proper timestamp; and
- a CSR contains proper userValidity values.

In the Ciphire system, a client can request a current timestamp from the Ciphire Time-Stamping Authority (TSA). This is done regularly by a client, and it is also done in certain special cases when time sensitive operations are performed, like certificate renewal. If a client is about to create the first certificate, a normal timestamp lookup is not possible, so in this case the time is retrieved from a proxy with a plaintext request.

By periodically obtaining the current time from the TSA, the Ciphire system ensures that all clients have a closely synchronized view of the current date and time. The TSA uses the UTC time zone (also known as GMT and ZULU clock time). The synchronization mechanism makes no attempt to be as precise as the Network Time Protocol (NTP)[21]. This level of precision is not needed for email security.

The TSA time is kept internally by the Ciphire client. It is not used to set the clock of the operating system or the hardware clock.

6 Comparison with X.509 and PGP

The Ciphire system employs a new custom certificate format. Why? In short, there are shortcomings with the X.509 and OpenPGP certificate formats. Ciphire certificates include the best aspects of both X.509 and OpenPGP certificates, and the Ciphire system adds some unique features.

Like X.509 certificates, Ciphire certificates are defined using ASN.1[22]. This approach is much more compact than other alternatives such as XML[23].

Like OpenPGP key files, which are often called OpenPGP certificates, Ciphire certificates contain more than one public key and are signed by more than one party. A Ciphire certificate always includes at least three public keys (RSA, DSA, and ElGamal), while the Ciphire CA and the owner of the private key are always the certificate signers.

6.1 Advantages and Disadvantages

The X.509 and OpenPGP certificate formats are well known and a lot of third-party software supports them. For this reason, adopting one of these certificate formats would have improved interoperability between the Ciphire system and other systems. This benefit was sacrificed to attain other benefits.

In the Ciphire system, a set of private keys and one Ciphire certificate is associated with a single email address. That is, a dedicated set of private keys is generated for each email address. If a user has multiple email addresses, a single Ciphire Mail client can handle all of them by creating the appropriate number of public/private key pairs and certificates. Similarly, a gateway uses a dedicated set of private keys generated for each host name and domain name.

This scheme cannot be easily accomplished with X.509 certificates, since they contain a single public key. X.509 certificates allow multiple email addresses to be associated with the same public key, but this is undesirable.

X.509 certificates support a rich set of naming alternatives. However, security requires that the name form used by the application be employed directly in the certificate. Many X.509 certificates do not include an email address. As a result, the application software must map the name in the certificate to an email address. This is often done poorly, leading to overall system vulnerability.

X.509 certificates support many trust models, including hierarchies and meshes. While this flexibility is sometimes desirable, it also adds complexity. The Ciphire system has selected a single trust model (more about that in the next chapter), and avoids much of the potential complexity.

Only the issuer signs the X.509 certificate. The replying party must have faith that the policies, practices, and procedures of the certificate issuer are followed. This is true in the Ciphire system, but the amount of faith that a relying party must have in the Cipher CA is much less. The self-signatures in the Ciphire certificate ensure that the private key owner is involved in certificate creation. Further, the Ciphire Fingerprint System (described in the next chapter) gives system users the ability to audit the certificate creation operations of the Ciphire CA. Together self-signatures and the Fingerprint system greatly reduce the potential of abuse or error by the Ciphire CA operators.

This brief discussion of the disadvantages of the X.509 certificate format leads us to the conclusion that a new custom certificate format is preferable to using the X.509 certificate format. The OpenPGP key file format has other disadvantages, which lead to a similar conclusion.

The OpenPGP key file format supports multiple public keys, and it supports multiple signers. However, the owner of the public key is usually the issuer of the certificate. There is no control from a third-party that ensures that appropriate email addresses (or real names) are associated with the public keys. Thus, a user can generate arbitrary OpenPGP key files. An OpenPGP public key file can contain signatures from third parties, such as other users, which certify that the identified subject owns the corresponding private key. All of the signers are equal peers. None of them assumes the responsibility for generation and distribution of revocation information. Revocation is the responsibility of the private key owner, but in practice, OpenPGP key files are not revoked when a private key is compromised or otherwise becomes obsolete.

The OpenPGP key file format supports a single trust model, which avoids much complexity. However, without an authority to take responsibility generation and distribution of revocation information, maintaining confidence in the overall system is difficult.

The OpenPGP key file format does not allow custom extensions, making it impossible to adapt it to other uses, such as those envisioned by the Ciphire system.

6.2 Compatibility with OpenPGP and X.509

An X.509 daughter certificate or an OpenPGP daughter key file is used when compatibility with another system is needed. For instance, an X.509 certificate is often needed for S/MIME email security, and an OpenPGP key file is needed for PGP email or file security.

While this feature is not yet implemented, in the future the Ciphire system will be able to provide these certificates in addition to Ciphire certificates. A Ciphire certificate will, in the future, contain references to daughter certificates. Ciphire certificates will reference an associated X.509 daughter certificate and an OpenPGP daughter key file. The daughterPublicKeysInfos field in the Ciphire certificate contains up to 256 references, which are comprised of the certificate identifier (CID), a hash algorithm identifier, and a hash value.

For an X.509 certificate, the Ciphire-style CID is calculated over the serial number and issuer data, and the hash is computed on the entire X.509 certificate.

For an OpenPGP key file, the Ciphire-style CID is composed of the PGP Key Identifier value followed by the string "PGPID" and an appropriate number of zeros to make the result the size of a normal CID, and the hash is the PGP key fingerprint.

7 Trust model

The Ciphire system employs a hybrid trust model. It combines elements of a hierarchical trust model and elements of a distributed trust model.

7.1 Hierarchical trust model elements

Ciphire employs a strict hierarchical trust model. The Ciphire system has a single Root CA and may use multiple CAs for issuing user certificates. At present, a single CA system is being used. The Ciphire Root CA issues CA certificates, and the Ciphire CA issues all other certificates within the Ciphire system.

7.1.1 Certificate Signatures

Certificates issued by the Ciphire CA are signed with the CA's RSA and DSA keys. In addition, the certificate chaining feature (as described in the certificate details chapter) is used with the Ciphire Root CA and Ciphire CA certificates.

Each client knows the Ciphire Root CA and Ciphire CA certificates as they are part of every client distribution. Updates of these certificates are handled automatically. If the client encounters a new authority certificate, the new one is retrieved from the CCD and validated. The validation includes a check of the certificate's chain, which must lead to a previously known authority certificate.

In addition to the Ciphire CA signatures, every certificate contains self-signatures. Also, successor signatures are included in renewed certificates. (See the certificate details chapter for the details.).

7.1.2 Authorization

A certificate is only deemed valid if all of the self-signatures and the Ciphire CA signatures are valid and the certification path leads to the Ciphire Root CA. A certificate validated with the public key in a normal user certificate is never considered to be valid.

Since the certificate includes both Ciphire CA signatures and self-signatures, the Ciphire CA and the private key owner must work together to create a valid certificate. A user cannot issue a certificate without the Ciphire CA, and the Ciphire CA cannot issue a certificate without the user. However, there are still certain malicious actions that could be performed by the Ciphire CA which are prevented with the distributed trust model elements described in the following section.

7.2 *Distributed trust model elements*

A hash value of a certificate is called a certificate fingerprint. This term was established by PGP, where the certificate fingerprint is used to authenticate the delivery of a certificate. The fingerprint is used o ensure that one certificate is not swapped for another by an attacker. This kind of checking should be done if a public key does not contain a signature from a trusted introducer. The Ciphire CA is a trusted introducer. With PGP the fingerprint check has to be performed manually by the user, which is a very tedious task. In the Ciphire system, a fingerprint system is used which makes it possible for a Ciphire client to perform a fingerprint check automatically, providing automatic certificate authentication.

The system consists of two parts, the fingerprint system itself and a mechanism to automatically exchange certain fingerprint data between clients.

7.2.1 Protection

The system protects against the following (intentional or unintentional) malicious actions that could be performed by the Ciphire CA or related authority systems:

- Malicious replacement of the public keys in a certificate (e.g., to allow man-in-the-middle attacks); and
- Malicious changes to one or more certificate fields, such as the validity dates or email address in the subject of the certificate.

7.2.2 The Ciphire Fingerprint System

The fingerprint system uses hash values and hash-chaining techniques to create a trusted log of certificate and summary hashes. In this case, the term “trusted” means that old entries in the log cannot be changed at a later time without invalidating newer entries.

Concept

Whenever the CA issues a new certificate or revokes an old certificate, a hash value H(C) over the certificate is calculated. In addition, a new summary hash H(S) is calculated. The summary hash is calculated over the new certificate and all previously generated hash values, including a creation timestamp T for each certificate hash.

The values H(C), H(S), and T are stored as an entry in a log, and the log is made available to all clients. As described above, the hash value H(C) is the certificate fingerprint. In the case of a certificate renewal, two fingerprints are created, one for the revocation certificate and one for the new certificate.

All fingerprint data is made available to clients, and the clients use it to authenticate certificates. The client calculates the certificate fingerprint and then compares it with the entry provided by the Ciphire CA. If they match, the certificate is considered valid.

Implementation Details

The actual implementation of the Ciphire fingerprint system is not based on a single log of fingerprints; rather the data is split in smaller parts forming a tree-like structure. This approach provides performance and scalability. Based on user configuration choices, the client software will either download all fingerprint data or just the portion that is required to authenticate a specific certificate. Since use of the fingerprint system is purely optional, a user may also disable it altogether if desired.

Fingerprint format

A single fingerprint consists of the following values:

- H(AID): The hash of the certificate's address ID (e.g., email address);
- H(CID): The hash of the certificate ID (serial number);
- H(C): The hash of the entire certificate; and
- M: A 2-byte metadata field, which tells if the Ciphire CA action is related to certificate creation, renewal, or revocation..

SHA_d-256 is used as the one-way hash function to compute each of these values, resulting in a total size of 98 byte per fingerprint.

Fingerprint Lists

Fingerprint data is partitioned into multiple lists of fingerprints for particular time intervals.

For each interval a tree structure of multiple fingerprint lists (FPLs) is generated. The leaves of the tree, called a "branch FPL," contain the actual certificate fingerprints. The summary hashes of each branch FPL are collected in section FPLs. Finally, all summary hashes of the section FPLs are collected in a single master FPL which represents the root of the tree for a particular time interval.

Each FPL type contains the same header and footer.

Header values are:

- Type: denotes the type of the FPL: either branch, section, or master;
- Version: the version number of the FPL format;
- Interval Start: timestamp denoting the start time of the FPL;
- Branch Total: the total number of available branches; and
- Branch Current: the current branch number for the FPL.

Footer values are:

- Interval End: timestamp denoting the end time of the FPL;
- Carry-over Hash: a hash calculated over the complete FPL of the previous and current time interval; and
- FPL Hash: a hash calculated over the complete contents of the FPL header.

At present, the Ciphire system uses an interval of 1 hour where such a set of fingerprint lists is generated. The number of branch FPLs is 4 per interval, but depending on the number of certificates created per interval. This number can be increased to ensure each branch FPL is a reasonable size.

From each master FPL a summary hash is derived. These summary hashes are collected in a single list which grows over time: the cross FPL list. Each entry in this list contains a timestamp, a hash calculated over the master FPL for the given interval and all previous entries of the cross FPL. This cross FPL hash is distributed between clients as described in the below.

7.2.3 Distribution of fingerprint data

How are fingerprints distributed to Ciphire clients? The client may not know if the fingerprint data provided by the Ciphire CA is really the originally created fingerprint data. The design ensures that the Ciphire CA is unable to provide fake data to a particular client.

For a client to be sure that it has the same fingerprint data as any other client, the most current summary hash is exchanged with other clients. When the user sends a secure email message to another Ciphire user, the client automatically includes the summary hash and timestamp in the email message. The receiving client extracts the hash and compares it with the corresponding hash in its local copy of the fingerprint data. If the hash values do not match, either the sending client or the receiving client has wrong fingerprint data.

The distribution of summary hashes is called cross-client verification, and it effectively creates a link between each Ciphire system user.

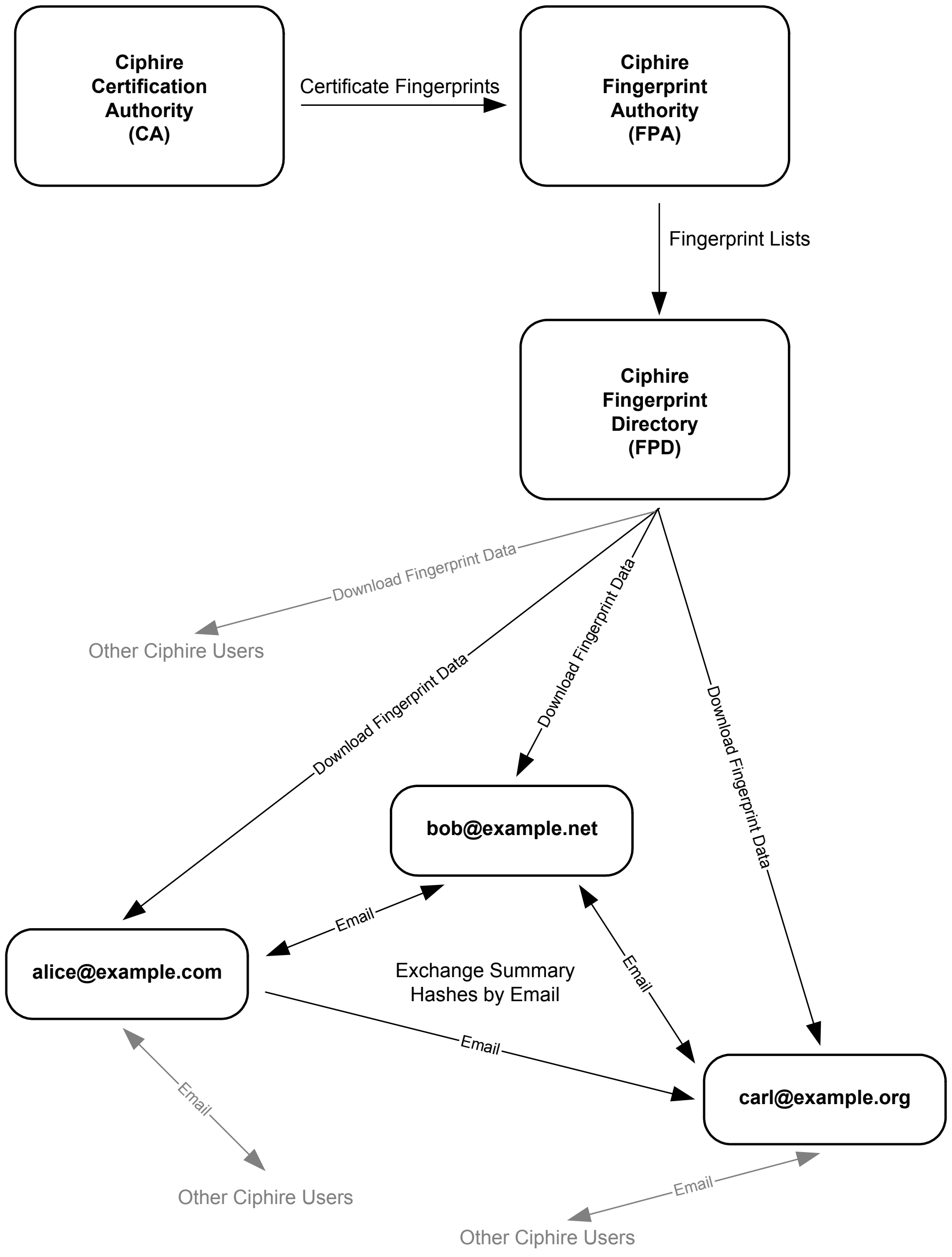
Ciphire Certification Authority (CA)
Certificate Fingerprints
Ciphire Fingerprint Authority (FPA)
Fingerprint Lists
Ciphire Fingerprint Directory (FPD)
Download Fingerprint Data
Other Ciphire Users
Download Fingerprint Data
Download Fingerprint Data
Download Fingerprint Data
bob@example.net
Email
alice@example.com
Exchange Summary Hashes by Email
Email
Email
carl@example.org
Email
Other Ciphire Users
Email
Other Ciphire Users

Figure 7.1. Distribution of fingerprint data.

There is a single case where a client does not need such a link to make an authoritative decision about entry validity: if the entry is for one of its own certificates.

Each client checks its own certificates against the fingerprint data and shares the summary hash with each communication partner. This practice makes it very hard to perform any of the potential CA attacks described above without alerting many users that something is wrong.

7.3 Trust Model Summary

The hybrid trust model employed by the Ciphire system reduces the authority of the Ciphire CA when compared with traditional hierarchical trust models. The community as of whole becomes a certification authority, providing the security and confidence in the system. The fingerprint system does not prevent the Ciphire CA – or anyone with control over it – from performing malicious actions; however, the fingerprint system does minimize the chances of such an action remaining undiscovered.

However, there are two things where a user simply has to trust the Ciphire CA:

- The Ciphire CA may just create a certificate for an arbitrary email address if no active certificate exists for that address. In combination with a man-in-the-middle attack, this bogus certificate could trick a user into sending data he would have never without encryption. The threat is minor. Most importantly, the owner of the email address will find out about the bogus certificate if he tries to create a legitimate certificate.
- The fingerprint system prevents the Ciphire CA from denying the existence of a certificate. If the CCD fails to return a certificate for an email address, there are two possibilities: either there really is no certificate for that address or the CCD pretends that it has been removed. However, there are several possibilities for a client to find out that something is wrong:
    - The client may have already seen a certificate for that address. In this case, at least a revocation certificate must exist, which cannot be generated by the Ciphire CA alone.
    - Unless a man-in-the-middle attack is being performed, the recipient of an email message may observe a plaintext signature and discover that the sender has a certificate.
    - Removal of a certificate from the CCD will stop the reception of all encrypted mail for that email address, which is likely to be noticed by the certificate owner.

Even considering these two minor issues, the hybrid trust model offers only the slightest probability of undetected compromise.

7.4 Proxy Communication

While proxy communication is not directly related to the trust model, the communication between the Ciphire client and the CCD is relevant because it is handled by one or more proxies. To avoid abuse and unauthorized use, a Ciphire client must authenticate itself to a proxy before sending a lookup. This is called *proxy logon*. This authentication means that communications with the CCD are not anonymous, potentially allowing the CCD and proxy to perform traffic analysis on the user's email communication.

This kind of traffic analysis is always possible for the Internet Service Provider (ISP) or the email service provider. However, the Ciphire system tries to minimize this risk using the mechanisms discussed below.

7.4.1 Lookup Cache

Every Ciphire client uses a lookup cache. The cache includes, but is not limited, to the caching of certificates as mentioned in an earlier chapter. The Ciphire client caches every query with its response, including negative responses. A user may configure its client to cache certificate query responses for several days. The Ciphire client will only use the local copy, and not send a query to the proxy, until the cached response expires.

7.4.2 Encrypted Communication

To prevent other parties from performing the same traffic analysis by watching the traffic between the Ciphire client and a proxy, communication is encrypted using a 128-bit cipher. By default the AES and Twofish algorithms are used: AES in CBC-HMAC mode with RSA for proxy logons and Twofish in CCM mode for lookups.

7.4.3 Hashed Lookup arguments

Lookup arguments, such as email addresses, are not included as plaintext values in the lookup. Instead a hash is calculated and used as argument. Only if the lookup yields a certificate will the proxy be able to determine the email address or certificate information of interest. Otherwise, the proxy is not able to derive any useful information from the lookup arguments.

7.4.4 Daily Logons

As previously stated, the Ciphire client is required to logon to a proxy before using it. During the logon, a digital signature is used to authenticate the Ciphire client to the proxy. If the digital signature and the corresponding certificate are valid, the proxy assigns a 128-bit session key and a 128-bit session token to the Ciphire client. This is done in a way that does not require the proxy to cache session keys or tokens. The proxy saves a 256-bit random value, and a new one is generated every 24 hours. The 128-bit token value and the 256-bit random value are hashed to compute the session key whenever it is needed.

Each lookup is encrypted with the session key. The plaintext session token and the encrypted lookup are sent to the proxy. The proxy uses the session token and its secret 256-bit random value to calculate the session key. If decryption fails, the Ciphire client must logon again.

Unless the proxy caches or stores any details from a client's logon, the data from a lookup cannot be used to derive the identity of the client. This makes it cumbersome for the proxy to correlate individual CCD queries and the requestor's identity.

7.4.5 Primary Certificate

The certificate used for the proxy logon is not necessarily the certificate for the account that is being used as the sender of an email message. One Ciphire client can manage multiple certificates and the corresponding private keys, but only one of these certificates is the *primary certificate*, which is used for all proxy logons.

By combining the CCD queries associated with many email addresses, details about email traffic are hidden from the CCD.

7.4.6 Local or third-party proxies

When a proxy successfully decrypts a lookup, the content is forwarded to another proxy or to a root proxy, which in turn communicates with the CCD. The proxy must logon to the next proxy, just like a Ciphire client, and the new lookup contains only the lookup arguments from the original request. Therefore the receiving proxy does not have any identity information about the original source of the lookup.

Using a proxy that is not operated by Ciphire Labs, like a local proxy operated by a company or the user's ISP, effectively hides the Ciphire client's identity, severely limiting the amount of information available for traffic analysis.

8 Cryptographic functions

For additional robustness the Ciphire system uses two or more different cryptographic algorithms for each function. This means that even if one algorithm is broken, the Ciphire system will remain secure.

Ciphire certificates and software are algorithms independent, and they accommodate any key size. Below is a summary of algorithms that are currently being used.

8.1 Asymmetric algorithms

The Ciphire system uses three asymmetric algorithms:

- RSA for digital signatures and encryption (the default key size is 2048 bits);
- ElGamal for encryption (the default key size is 2048 bits); and
- DSA-2k for digital signatures (the default key size is 2048 bits). The current DSA standard supports only keys up to 1024 bits in length. The Ciphire system uses the DSA algorithm with a 2048-bit prime and a 256-bit group order. To highlight this extension to the DSA standard, the signature algorithm with the larger key sizes is called "DSA-2k".

8.2 Hash algorithms

The Ciphire system supports various hash functions, including SHA_d-256, SHA_d-512[24], and Whirlpool_d (512 bit)[25]. Currently, the default hash algorithms are SHA_d-256 and Whirlpool_d.

The "$_d$" denotes that the hash function is used in double-hashing mode. In this mode, the output of the hashing function is again used as input to the hashing function, i.e., H(H(M)) is being computed instead of just H(M). The second hash computation destroys all information about the internal state after hashing M, eliminating any possibility of length extension attacks (which plague almost all hash functions).

8.3 Digital Signatures

Digital signatures created by the Ciphire system are created using one of the following:

1. DSA with SHA_d-256 (used for normal signatures on email messages and lookup responses from Ciphire infrastructure systems, like CCD),
2. RSA with SHA_d-256 (used for signing fingerprint lists), and
3. RSA with $Whirlpool_d$ (used to authenticate email message session keys).

Certificates are always signed with all of the signing keys available to the CA and user. Currently, DSA and RSA are both used to sign certificates.

8.4 Symmetric algorithms

The Ciphire system supports most common encryption algorithms. Currently the preferred encryption algorithms are AES and Twofish. Three modes of operation are supported: CBC-HMAC, CCM (Counter with CBC-MAC), and CTR (Counter).

8.5 Pseudo-random number generation

The Fortuna[26] pseudo-random number generator (PRNG) is used by the Ciphire client for all operations requiring random numbers. The Ciphire implementation of Fortuna uses the Twofish cipher in counter mode with multiple OS-specific entropy sources.

8.6 SSL-specific cryptography

The Ciphire client supports SSL/TLS to protect communications between mail clients and mail servers using the SMTP, POP3, or IMAP4 protocols. Supported versions include SSL v3[27], SSL v2[28], and TLS v1[29].

Ciphire-specific communication is not based on SSL or TLS.

9 Cryptographic Protocols and Formats

Ciphire applications use a single common cryptographic layer, including the Ciphire Packet Format (CPF), which is used for all internal communication between the Ciphire system components. CPF protects non-interactive protocols like the Ciphire mail protocol, as well as interactive protocols like the proxy communication protocol. The CPF allows two peers to communicate securely over an arbitrary transport protocol, and the resulting secure communication channel is called a *tunnel*. The initiator of a tunnel establishes a *forward tunnel*, and in the case of an interactive communication, the responder may establish a *backward tunnel*. Data sent over a tunnel is split into discrete messages, called *packets*, thus the name Ciphire Packet Format. These packets can contain either control or content data.

The CPF is used in three general operational modes:

- Key mode: Encryption and authentication operations using asymmetric encryption to establish session keys;

- Token mode: Encryption and authentication operations with previously-defined session keys (symmetric encryption only); and
- Signature mode: Digital signature operations.

Key mode uses the first packet to establish the session keys that will be used to encrypt the remaining data. Using asymmetric cryptography, the sender encrypts the session keys using one or more public keys from the receiver's certificate.

When a session key is available, asymmetric encryption is not required. A session token may be defined in the first packet of a session. Despite this difference, token and key modes are very similar, encrypting the remaining data in the same manner.

Signature mode is not directly related to the exchange of data between two peers; rather, it provides signature operations. These operations may be combined with the encryption operations of key or token mode.

The primary aspects of CPF are:

- The data in a single communication session data can be encrypted with multiple different keys using different algorithms;
- Symmetric and asymmetric authentication;
- The sender identity and its authentication information is always encrypted;
- Arbitrary cryptographic algorithms can be used;
- Arbitrary packet length is supported;
- Every packet contains a timestamp; and
- Accommodates both interactive and store-and-forward transport protocols.

9.1 Packet Types and Structure

CPF includes three packet types: Init, Data, and Control.

The Init packet contains information about the encryption and keys being used, including authentication information, a session identifier, and other metadata. In a communication, the Init packet is always the first packet sent by the initiator.

The Data packet contains the actual content data, authentication data, a session identifier, timestamp, counter, and other metadata. In a communication, an arbitrary number of data packets may be exchanged.

The Control packet is similar to the data packet; it contains predefined control information.

9.1.1 Init Packet

An Init packet is always the first packet sent by the initiator of the tunnel, and it starts a transfer session of data between two peers.

Init packets serve the following purposes:

- They define a unique 256-bit session identifier that is used in every packet that is part of the session. The session identifier is encrypted in the init packet, but the session identifier is plaintext in data and control packets. A single init packet may define multiple session identifiers.

- Selects the cryptographic algorithms. The encryption algorithms and generated (key mode) or pre-defined (token mode) session keys for a particular session are selected. An Init packet may contain an arbitrary number of such algorithm definitions. The asymmetric algorithm and one-way hash function being used to create authentication information are selected as well.
- Establishes the recipient of the protected data by including the public key identifier of the from the recipient's certificate. The plaintext key identifier is included, allowing a receiver to select the appropriate private key for decryption.

9.1.2 Data and Control Packets

Data and control packets are nearly identical, but control packets only include control commands. These packets have the same format for both, key mode and token mode.

Each data and control packet contains a header, authentication information, and the content data. The header contains the packet length, counter, session identifier, and content length. These values (apart from the content length) are the only part of the packet that is not encrypted.

A data packet payload may be of any size, but it is typically less than 16 Kbytes.

9.1.3 Signature Packets

The plaintext content data is signed, and then it is encrypted and transferred using data packets. The signature data is added to the plaintext content, and it becomes part of the payload of a data packet.

A single data packet can include multiple signatures created using different algorithms and private keys.

10 Conclusions

The Ciphire system provides authentication, integrity, and confidentiality of email messages. Unlike other email security systems available today, the Ciphire system automates most of the key management and certificate management. These are the areas that make other email security solutions too cumbersome to be used by most email users. By eliminating prevalent problems related to enrollment, certificate distribution, and revocation, the designers of the Ciphire system have created an email security solution that can be used by anyone--even the most novice computer user. Yet, the Ciphire system includes robust security mechanisms, including safeguards against flaws being discovered in any one of the cryptographic algorithms. The use of multiple layers of protection ensures that the discovery of such a flaw will not result in the compromise of a single email message or a single certificate in the Ciphire infrastructure. While providing robust security, the Ciphire-specific protocols include features to protect the privacy of all email users.

11 Author Information

Russ Housley is the founder of Vigil Security, LLC, and he is coauthor of "Planning for PKI" published by John Wiley & Sons. He has over 20 years of communications and computer security experience. His expertise is in security protocols, system engineering, system security architectures, and product definition. He is Security Area Director for the Internet Engineering Task Force (IETF). He is the author of the Cryptographic Message Syntax (CMS), the security foundation for S/MIME. He is one of the authors of the Internet X.509 Certificate Profile (RFC 3280), commonly called PKIX Part 1. He is one of the authors of the SDNS Message Security Protocol (MSP), the security cornerstone of the U.S. Defense Message System (DMS). He is one of the authors of the IEEE series of LAN/MAN security standards (IEEE 802.10), and he helped improve the security of IEEE 802.11 Wireless LANs with his contributions to IEEE 802.11i.

Niels Ferguson is a cryptographic engineer and consultant. He has extensive experience in the design and implementation of cryptographic algorithms, protocols, and large-scale security infrastructures. Previously, Ferguson was a cryptographer for DigiCash and CWI, the National Research Institute for Mathematics and Computer Science in the Netherlands, and he worked closely with Bruce Schneier at Counterpane Internet Security where he helped to design the block cipher Twofish for the Advanced Encryption Standard (AES) competition. He has published numerous scientific papers. Currently, he is working as a cryptographic engineer for Microsoft.

References

[1] W. Diffie, M. E. Hellmann: “New Directions in Cryptography”, IEEE Transactions on Information Theory, 1976.

[2] J. Linn: “Privacy-Enhanced Electronic Mail”, RFC 1421, February 1993.

[3] J. Callas, et al: “OpenPGP Message Format”, RFC 2440, November 1998.

[4] B. Ramsdell, et al: “S/MIME Version 3 Message Specification”, RFC 2633, June 1999.

[5] L. Eilebrecht: “Ciphire - Technical Product Description”, Ciphire Labs, unpublished.

[6] J. Klensin, et al: “Simple Mail Transfer Protocol”, RFC 2821, April 2001.

[7] J. Myers, M. Rose: “Post Office Protocol - Version 3”, RFC 1939, May 1996.

[8] M. Crispin: “Internet Message Access Protocol - Version 4rev1”, RFC 2060, December 1996.

[9] P. Gutmann: “Why isn't the Internet Secure Yet, Dammit!”, http://www.cs.auckland.ac.nz/~pgut001/pubs/dammit.pdf, 2004.

[10] B. Kaliski; “PKCS #1: RSA Cryptography Specifications Version 2.0”, RFC 2437, March 1998.

[11] NIST: “Digital Signature Standard (DSS)”, FIPS 186-2, January 2000.

[12] T. ElGamal: “A public-key cryptosystem and a signature scheme based on discrete logarithms.”, IEEE Transactions on Information Theory, IT-31: 469-472, 1985.

[13] S. Haber, W. S. Stornetta: “How to Time-Stamp a Digital Document”, Journal of Cryptography, 1991.

[14] R. Housley, et al: “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL)”, RFC 3280, April 2002.

[15] NIST: “Advanced Encryption Standard (AES)”, FIPS-192, November 2001.

[16] H. Krawczyk, et al: “HMAC: Keyed-Hashing for Message Authentication”, RFC 2104, February 1997.

[17] B. Schneier, et al: “Twofish: A 128-bit Block Cipher”, http://www.schneier.com/paper-twofish-paper.pdf, June 1998.

[18] D. Whitting, R. Housley, N. Ferguson: “Counter with CBC-MAC (CCM)”, http://www.macfergus.com/pub/ccm.html, 2002.

[19] P. Resnick: “Internet Message Format”, RFC 2822, April 2001.

[20] N. Freed: “Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies”, RFC 2045, November 1996.

[21] D. L. Mills: “Network Time Protocol (Version 3) Specification, Implementation”, RFC 1305, March 1992.

[22] ITU: “Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation”, ITU-T Recommendation X.680 (ISO/IEC 8824-1:2002), 2002.

[23] T. Bray, et al: “Extensible Markup Language (XML) 1.0”, W3C Recommendation, February 1998.

[24] NIST: “Specifications for the Secure Hash Standard”, FIPS 180-2, August 2002.

[25] V. Rijmen, P. S. L. M. Barreto: “The Whirlpool Hash Function”, http://planeta.terra.com.br/informatica/paulobarreto/WhirlpoolPage.html, 2001.

[26] N. Ferguson, B. Schneier: “Practical Cryptography”, Wiley, 2003.

[27] A. Frier, P. Karlton, P. Kocher, “The SSL 3.0 Protocol”, Netscape Communications Corp., November1996.

[28] Hickman, Kipp: “The SSL Protocol”, Netscape Communications Corp., February 1995.

[29] T. Dierks, C. Allen: “The TLS Protocol Version 1.0”, RFC 2246, January 1999.