essence, the data are created in the native form that corresponds to the type specified by the media type. ■ Canonical form: The entire body, including out-of-band information such as record lengths and possibly file attribute information, is converted to a universal canonical form. The specific media type of the body as well as its associated attributes dictates the nature of the canonical form that is used. Conversion to the proper canonical form may involve character set conversion, transformation of audio data, compression, or various other operations specific to the various media types. 8.3 E-mail Threats and Comprehensive E-mail Security For both organizations and individuals, e-mail is both pervasive and especially vulnerable to a wide range of security threats. In general terms, e-mail security threats can be classified as follows: ■ Authenticity-related threats: Could result in unauthorized access to an enterprise's e-mail system. ■ Integrity-related threats: Could result in unauthorized modification of e-mail content. ■ Confidentiality-related threats: Could result in unauthorized disclosure of sensitive information. ■ Availability-related threats: Could prevent end users from being able to send or receive e-mail. A useful list of specific e-mail threats, together with approaches to mitigation, is provided in SP 800-177 (Trustworthy E-mail, September 2015) and is shown in Table 8.3. SP 800-177 recommends use of a variety of standardized protocols as a means for countering these threats. These include: ■ STARTTLS: An SMTP security extension that provides authentication, integrity, non-repudiation (via digital signatures) and confidentiality (via encryption) for the entire SMTP message by running SMTP over TLS. ■ S/MIME: Provides authentication, integrity, non-repudiation (via digital signatures) and confidentiality (via encryption) of the message body carried in SMTP messages. ■ DNS Security Extensions (DNSSEC): Provides authentication and integrity protection of DNS data, and is an underlying tool used by various e-mail security protocols. ■ DNS-based Authentication of Named Entities (DANE): Is designed to overcome problems in the certificate authority (CA) system by providing an alternative channel for authenticating public keys based on DNSSEC, with the 8.3 / E-mail Threats and Comprehensive E-mail Security 267 Threat Impact on Purported Sender Impact on Receiver Mitigation E-mail sent by unauthorized MTA in enterprise (e.g., malware botnet) Loss of reputation, valid e-mail from enterprise may be blocked as possible spam/phishing attack. UBE and/or e-mail containing malicious links may be delivered into user inboxes. Deployment of domainbased authentication techniques. Use of digital signatures over e-mail. E-mail message sent using spoofed or unregistered sending domain Loss of reputation, valid e-mail from enterprise may be blocked as possible spam/phishing attack. UBE and/or e-mail containing malicious links may be delivered into user inboxes. Deployment of domainbased authentication techniques. Use of digital signatures over e-mail. E-mail message sent using forged sending address or e-mail address (i.e., phishing, spear phishing) Loss of reputation, valid e-mail from enterprise may be blocked as possible spam/phishing attack. UBE and/or e-mail containing malicious links may be delivered. Users may inadvertently divulge sensitive information or PII. Deployment of domainbased authentication techniques. Use of digital signatures over e-mail. E-mail modified in transit Leak of sensitive information or PII. Leak of sensitive information, altered message may contain malicious information. Use of TLS to encrypt e-mail transfer between servers. Use of end-toend e-mail encryption. Disclosure of sensitive information (e.g., PII) via monitoring and capturing of e-mail traffic Leak of sensitive information or PII. Leak of sensitive information, altered message may contain malicious information. Use of TLS to encrypt e-mail transfer between servers. Use of end-toend e-mail encryption. Unsolicited Bulk E-mail (UBE) (i.e., spam) None, unless purported sender is spoofed. UBE and/or e-

mail containing malicious links may be delivered into user inboxes. Techniques to address UBE. DoS/DDoS attack against an enterprises' e-mail servers Inability to send e-mail. Inability to receive e-mail. Multiple mail servers, use of cloud-based e-mail providers. Table 8.3 E-mail Threats and Mitigations result that the same trust relationships used to certify IP addresses are used to certify servers operating on those addresses. ■ Sender Policy Framework (SPF): Uses the Domain Name System (DNS) to allow domain owners to create records that associate the domain name with a specific IP address range of authorized message senders. It is a simple matter for receivers to check the SPF TXT record in the DNS to confirm that the purported sender of a message is permitted to use that source address and reject mail that does not come from an authorized IP address. ■ DomainKeys Identified Mail (DKIM): Enables an MTA to sign selected headers and the body of a message. This validates the source domain of the mail and provides message body integrity. ■ Domain-based Message Authentication, Reporting, and Conformance (DMARC): Lets senders know the proportionate effectiveness of their SPF and DKIM policies, and signals to receivers what action should be taken in various individual and bulk attack scenarios. 268 chapter 8 / Electronic Mail Security Figure 8.4 shows how these components interact to provide message authenticity and integrity. Not shown, for simplicity, is that S/MIME also provides message confidentiality by encrypting messages. 8.4 S/MIME Secure/Multipurpose Internet Mail Extension (S/MIME) is a security enhancement to the MIME Internet e-mail format standard based on technology from RSA Data Security. S/MIME is a complex capability that is defined in a number of documents. The most important documents relevant to S/MIME include the following: ■ RFC 5750, S/MIME Version 3.2 Certificate Handling: Specifies conventions for X.509 certificate usage by (S/MIME) v3.2. Figure 8.4 The Interrelationship of DNSSEC, SPF, DKIM, DMARC, DANE, and S/MIME for Assuring Message Authenticity and Integrity msg msg sig msg sig msg sig Sender MUA Sender's S/MIME signing key (private key) DKIM signature DKIM TXT RR provides sending MTA's public key to receiving MTA DMARC TXT tells receiving MTA that sender uses DKIM and SPF DANE TLSA RR specifies SMTP TLS certificate Receiver MUA verifies S/MIME signature DNSSEC secured DNSSEC secured MTA's DKIM signing key DANE = DNS-based Authentication of Named Entities DKIM = DomainKeys Identified Mail DMARC = Domain-based Message Authentication, Reporting, and Conformance DNSSEC = Domain Name System Security Extensions SPF = Sender Policy Framework S/MIME = Secure Multi-Purpose Internet Mail Extensions TLSA RR = Transport Layer Security Authentication Resource Record SPF TXT specfies sender's IP address Sender DNS Receiver DNS Receiver MUA Sending MTA Receiving MTA 8.4 / S/MIME 269 ■ RFC 5751, S/MIME) Version 3.2 Message Specification: The principal defining document for S/MIME message creation and processing. ■ RFC 4134, Examples of S/MIME Messages: Gives examples of message bodies formatted using S/MIME. ■ RFC 2634, Enhanced Security Services for S/MIME: Describes four optional security service extensions for S/MIME. ■ RFC 5652, Cryptographic Message Syntax (CMS): Describes the Cryptographic Message Syntax (CMS). This syntax is used to digitally sign, digest, authenticate, or encrypt arbitrary message content. ■ RFC 3370, CMS Algorithms: Describes the conventions for using several cryptographic algorithms with the CMS. ■ RFC 5752, Multiple Signatures in CMS: Describes the use of multiple, parallel signatures for a message. ■ RFC 1847, Security Multiparts for MIME—Multipart/Signed and Multipart/ Encrypted: Defines a framework within which security services may be applied to MIME body parts. The use of a digital signature is relevant to S/MIME, as explained subsequently. Operational Description S/MIME provides for four message-related services: authentication, confidentiality,

compression, and e-mail compatibility (Table 8.4). This subsection provides an overview. We then look in more detail at this capability by examining message formats and message preparation. Authentication Authentication is provided by means of a digital signature, using the general scheme discussed in Chapter 3 and illustrated in Figure 3.15. Most commonly RSA with SHA-256 is used. The sequence is as follows: 1. The sender creates a message. 2. SHA-256 is used to generate a 256-bit message digest of the message. Function Typical Algorithm Typical Action Digital signature RSA/SHA-256 A hash code of a message is created using SHA-256. This message digest is encrypted using SHA-256 with the sender's private key and included with the message. Message encryption AES-128 with CBC A message is encrypted using AES-128 with CBC with a one-time session key generated by the sender. The session key is encrypted using RSA with the recipient's public key and included with the message. Compression unspecified A message may be compressed for storage or transmission. E-mail compatibility Radix-64 conversion To provide transparency for e-mail applications, an encrypted message may be converted to an ASCII string using radix-64 conversion. Table 8.4 Summary of S/MIME Services 270 chapter 8 / Electronic Mail Security 3. The message digest is encrypted with RSA using the sender's private key, and the result is appended to the message. Also appended is identifying information for the signer, which will enable the receiver to retrieve the signer's public key. 4. The receiver uses RSA with the sender's public key to decrypt and recover the message digest. 5. The receiver generates a new message digest for the message and compares it with the decrypted hash code. If the two match, the message is accepted as authentic. The combination of SHA-256 and RSA provides an effective digital signature scheme. Because of the strength of RSA, the recipient is assured that only the possessor of the matching private key can generate the signature. Because of the strength of SHA-256, the recipient is assured that no one else could generate a new message that matches the hash code and, hence, the signature of the original message. Although signatures normally are found attached to the message or file that they sign, this is not always the case: Detached signatures are supported. A detached signature may be stored and transmitted separately from the message it signs. This is useful in several contexts. A user may wish to maintain a separate signature log of all messages sent or received. A detached signature of an executable program can detect subsequent virus infection. Finally, detached signatures can be used when more than one party must sign a document, such as a legal contract. Each person's signature is independent and therefore is applied only to the document. Otherwise, signatures would have to be nested, with the second signer signing both the document and the first signature, and so on. Confidentiality S/MIME provides confidentiality by encrypting messages. Most commonly AES with a 128-bit key is used, with the cipher block chaining (CBC) mode. The key itself is also encrypted, typically with RSA, as explained below. As always, one must address the problem of key distribution. In S/MIME, each symmetric key, referred to as a content-encryption key, is used only once. That is, a new key is generated as a random number for each message. Because it is to be used only once, the content-encryption key is bound to the message and transmitted with it. To protect the key, it is encrypted with the receiver's public key. The sequence can be described as follows: 1. The sender generates a message and a random 128-bit number to be used as a content-encryption key for this message only. 2. The message is encrypted using the content-encryption key. 3. The content-encryption key is encrypted with RSA using the recipient's public key and is attached to the message. 4. The receiver uses RSA with its private key to decrypt and recover the content-encryption key. 5. The content-encryption key is used to decrypt the message. Several observations

may be made. First, to reduce encryption time, the combination of symmetric and public-key encryption is used in preference to simply using public-key encryption to encrypt the message directly: Symmetric algorithms 8.4 / S/MIME 271 are substantially faster than asymmetric ones for a large block of content. Second, the use of the public-key algorithm solves the session-key distribution problem, because only the recipient is able to recover the session key that is bound to the message. Note that we do not need a session-key exchange protocol of the type discussed in Chapter 4, because we are not beginning an ongoing session. Rather, each message is a one-time independent event with its own key. Furthermore, given the store-and-forward nature of electronic mail, the use of handshaking to assure that both sides have the same session key is not practical. Finally, the use of onetime symmetric keys strengthens what is already a strong symmetric encryption approach. Only a small amount of plaintext is encrypted with each key, and there is no relationship among the keys. Thus, to the extent that the public-key algorithm is secure, the entire scheme is secure. Confidentiality and Authentication As Figure 8.5 illustrates, both confidentiality and encryption may be used for the same message. The figure shows a sequence in which a signature is generated for the plaintext message and appended to the message. Then the plaintext message and signature are encrypted as a single block using symmetric encryption and the symmetric encryption key is encrypted using public-key encryption. S/MIME allows the signing and message encryption operations to be performed in either order. If signing is done first, the identity of the signer is hidden by the encryption. Plus, it is generally more convenient to store a signature with a plaintext version of a message. Furthermore, for purposes of third-party verification, if the signature is performed first, a third party need not be concerned with the symmetric key when verifying the signature. If encryption is done first, it is possible to verify a signature without exposing the message content. This can be useful in a context in which automatic signature verification is desired, as no private key material is required to verify a signature. However, in this case the recipient cannot determine any relationship between the signer and the unencrypted content of the message. E-mail Compatibility When S/MIME is used, at least part of the block to be transmitted is encrypted. If only the signature service is used, then the message digest is encrypted (with the sender's private key). If the confidentiality service is used, the message plus signature (if present) are encrypted (with a one-time symmetric key). Thus, part or all of the resulting block consists of a stream of arbitrary 8-bit octets. However, many electronic mail systems only permit the use of blocks consisting of ASCII text. To accommodate this restriction, S/MIME provides the service of converting the raw 8-bit binary stream to a stream of printable ASCII characters, a process referred to as 7-bit encoding. The scheme typically used for this purpose is Base64 conversion. Each group of three octets of binary data is mapped into four ASCII characters. See Appendix K for a description. One noteworthy aspect of the Base64 algorithm is that it blindly converts the input stream to Base64 format regardless of content, even if the input happens to be ASCII text. Thus, if a message is signed but not encrypted and the conversion is applied to the entire block, the output will be unreadable to the casual observer, which provides a certain level of confidentiality. 272 chapter 8 / Electronic Mail Security RFC 5751 also recommends that even if outer 7-bit encoding is not used, the original MIME content should be 7-bit encoded. The reason for this is that it allows the MIME entity to be handled in any environment without changing it. For example, a trusted gateway might remove the encryption, but not the signature, of a message, and then forward the signed message on to the end recipient so that they can verify the signatures directly. If the transport internal to the site is not 8-bit clean, such as on a wide

area network with a single mail gateway, verifying the signature will not be possible unless the original MIME entity was only 7-bit data. Compression S/MIME also offers the ability to compress a message. This has the benefit of saving space both for e-mail transmission and for file storage. Figure 8.5 Simplified S/MIME Functional Flow Sign (e.g., RSA/ SHA-256) Sender's private key (a) Sender signs, then encrypts message (b) Receiver decrypts message, then veri•es sender's signature One-time secret key Encrypt (e.g, AES-128/ CBC Encrypt (e.g., RSA) msg msg sig sig sig msg sig Receiver's public key Sender's public key Decrypt (e.g., RSA) Receiver's private key Secret key generated by sender Decrypt (e.g, AES-128/ CBC Verify signature (e.g., RSA/ SHA-256) msg msg 8.4 / S/MIME 273 Compression can be applied in any order with respect to the signing and message encryption operations. RFC 5751 provides the following guidelines: ■ Compression of binary encoded encrypted data is discouraged, since it will not yield significant compression. Base64 encrypted data could very well benefit, however. ■ If a lossy compression algorithm is used with signing, you will need to compress first, then sign. S/MIME Message Content Types S/MIME uses the following message content types, which are defined in RFC 5652, Cryptographic Message Syntax: ■ Data: Refers to the inner MIME-encoded message content, which may then be encapsulated in a SignedData, EnvelopedData, or CompressedData content type. ■ SignedData: Used to apply a digital signature to a message. ■ EnvelopedData: This consists of encrypted content of any type and encryptedcontent encryption keys for one or more recipients. ■ CompressedData: Used to apply data compression to a message. The Data content type is also used for a procedure known as clear signing. For clear signing, a digital signature is calculated for a MIME-encoded message and the two parts, the message and signature, form a multipart MIME message. Unlike SignedData, which involves encapsulating the message and signature in a special format, clear-signed messages can be read and their signatures verified by e-mail entities that do not implement S/MIME. Approved Cryptographic Algorithms Table 8.5 summarizes the cryptographic algorithms used in S/MIME. S/MIME uses the following terminology taken from RFC 2119 (Key Words for use in RFCs to Indicate Requirement Levels, March 1997) to specify the requirement level: ■ MUST: The definition is an absolute requirement of the specification. An implementation must include this feature or function to be in conformance with the specification. ■ SHOULD: There may exist valid reasons in particular circumstances to ignore this feature or function, but it is recommended that an implementation include the feature or function. The S/MIME specification includes a discussion of the procedure for deciding which content encryption algorithm to use. In essence, a sending agent has two decisions to make. First, the sending agent must determine if the receiving agent is capable of decrypting using a given encryption algorithm. Second, if the receiving agent is only capable of accepting weakly encrypted content, the sending agent must decide if it is acceptable to send using weak encryption. To support this decision process, a sending agent may announce its decrypting capabilities in order of preference for any message that it sends out. A receiving agent may store that information for future use. 274 chapter 8 / Electronic Mail Security The following rules, in the following order, should be followed by a sending agent. 1. If the sending agent has a list of preferred decrypting capabilities from an intended recipient, it SHOULD choose the first (highest preference) capability on the list that it is capable of using. 2. If the sending agent has no such list of capabilities from an intended recipient but has received one or more messages from the recipient, then the outgoing message SHOULD use the same encryption algorithm as was used on the last signed and encrypted message received from that intended recipient. 3. If the sending agent has no

knowledge about the decryption capabilities of the intended recipient and is willing to risk that the recipient may not be able to decrypt the message, then the sending agent SHOULD use triple DES. 4. If the sending agent has no knowledge about the decryption capabilities of the intended recipient and is not willing to risk that the recipient may not be able to decrypt the message, then the sending agent MUST use RC2/40. If a message is to be sent to multiple recipients and a common encryption algorithm cannot be selected for all, then the sending agent will need to send two messages. However, in that case, it is important to note that the security of the message is made vulnerable by the transmission of one copy with lower security. S/MIME Messages S/MIME makes use of a number of new MIME content types. All of the new application types use the designation PKCS. This refers to a set of public-key cryptography specifications issued by RSA Laboratories and made available for the S/MIME effort. Function Requirement Create a message digest to be used in forming a digital signature. MUST support SHA-256 SHOULD support SHA-1 Receiver SHOULD support MD5 for backward compatibility Use message digest to form a digital signature. MUST support RSA with SHA-256 SHOULD support —DSA with SHA-256 —RSASSA-PSS with SHA-256 —RSA with SHA-1 —DSA with SHA-1 —RSA with MD5 Encrypt session key for transmission with a message. MUST support RSA encryption SHOULD support —RSAES-OAEP —Diffie–Hellman ephemeral-static mode Encrypt message for transmission with a one-time session key. MUST support AES-128 with CBC SHOULD support —AES-192 CBC and AES-256 CBC —Triple DES CBC Table 8.5 Cryptographic Algorithms Used in S/MIME 8.4 / S/MIME 275 We examine each of these in turn after first looking at the general procedures for S/MIME message preparation. Securing a Mime Entity S/MIME secures a MIME entity with a signature, encryption, or both. A MIME entity may be an entire message (except for the RFC 5322 headers), or if the MIME content type is multipart, then a MIME entity is one or more of the subparts of the message. The MIME entity is prepared according to the normal rules for MIME message preparation. Then the MIME entity plus some security-related data, such as algorithm identifiers and certificates, are processed by S/MIME to produce what is known as a PKCS object. A PKCS object is then treated as message content and wrapped in MIME (provided with appropriate MIME headers). This process should become clear as we look at specific objects and provide examples. In all cases, the message to be sent is converted to canonical form. In particular, for a given type and subtype, the appropriate canonical form is used for the message content. For a multipart message, the appropriate canonical form is used for each subpart. The use of transfer encoding requires special attention. For most cases, the result of applying the security algorithm will be to produce an object that is partially or totally represented in arbitrary binary data. This will then be wrapped in an outer MIME message and transfer encoding can be applied at that point, typically base64. However, in the case of a multipart signed message (described in more detail later), the message content in one of the subparts is unchanged by the security process. Unless that content is 7 bit, it should be transfer encoded using base64 or quotedprintable so that there is no danger of altering the content to which the signature was applied. We now look at each of the S/MIME content types. EnvelopedData An application/pkcs7-mime subtype is used for one of four categories of S/MIME processing, each with a unique smime-type parameter. In all cases, the resulting entity, (referred to as an object) is represented in a form known as Basic Encoding Rules (BER), which is defined in ITU-T Recommendation X.209. The BER format consists of arbitrary octet strings and is therefore binary data. Such an object should be transfer encoded with base64 in the outer MIME message. We first look at envelopedData. The steps for preparing an envelopedData MIME entity are: 1.

Generate a pseudorandom session key for a particular symmetric encryption algorithm (RC2/40 or triple DES). 2. For each recipient, encrypt the session key with the recipient's public RSA key. 3. For each recipient, prepare a block known as RecipientInfo that contains an identifier of the recipient's public-key certificate,1 an identifier of the algorithm used to encrypt the session key, and the encrypted session key. 4. Encrypt the message content with the session key. 1 This is an X.509 certificate, discussed later in this section. 276 chapter 8 / Electronic Mail Security The RecipientInfo blocks followed by the encrypted content constitute the envelopedData. This information is then encoded into base64. A sample message (excluding the RFC 5322 headers) is given below. Content-Type: application/pkcs7-mime; smime-type=envelopeddata; name=smime.p7m Content-Transfer-Encoding: base64 Content-Disposition: attachment; filename=smime.p7m
rfvbnj756tbBghyHhHUujhJhjH77n8HHGT9HG4VQpfyF467GhIGfHfYT6
7n8HHGghyHhHUujhJh4VQpfyF467GhIGfHfYGTrfvbnjT6jH7756tbB9H
f8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
0GhIGfHfQbnj756YT64V To recover the encrypted message, the recipient first strips off the base64 encoding. Then the recipient's private key is used to recover the session key. Finally, the message content is decrypted with the session key. SignedData The signedData smime-type can be used with one or more signers. For clarity, we confine our description to the case of a single digital signature. The steps for preparing a signedData MIME entity are as follows. 1. Select a message digest algorithm (SHA or MD5). 2. Compute the message digest (hash function) of the content to be signed. 3. Encrypt the message digest with the signer's private key. 4. Prepare a block known as SignerInfo that contains the signer's public-key certificate, an identifier of the message digest algorithm, an identifier of the algorithm used to encrypt the message digest, and the encrypted message digest. The signedData entity consists of a series of blocks, including a message digest algorithm identifier, the message being signed, and SignerInfo. The signedData entity may also include a set of public-key certificates sufficient to constitute a chain from a recognized root or top-level certification authority to the signer. This information is then encoded into base64. A sample message (excluding the RFC 5322 headers) is the following. Content-Type: application/pkcs7-mime; smime-type=signeddata; name=smime.p7m Content-Transfer-Encoding: base64 Content-Disposition: attachment; filename=smime.p7m
567GhIGfHfYT6ghyHhHUujpfyF4f8HHGTrfvhJhjH776tbB9HG4VQbnj7
77n8HHGT9HG4VQpfyF467GhIGfHfYT6rfvbnj756tbBghyHhHUujhJhjH
HUujhJh4VQpfyF467GhIGfHfYGTrfvbnjT6jH7756tbB9H7n8HHGghyHh
6YT64V0GhIGfHfQbnj75 8.4 / S/MIME 277 To recover the signed message and verify the signature, the recipient first strips off the base64 encoding. Then the signer's public key is used to decrypt the message digest. The recipient independently computes the message digest and compares it to the decrypted message digest to verify the signature. Clear Signing Clear signing is achieved using the multipart content type with a signed subtype. As was mentioned, this signing process does not involve transforming the message to be signed, so that the message is sent "in the clear." Thus, recipients with MIME capability but not S/MIME capability are able to read the incoming message. A multipart/signed message has two parts. The first part can be any MIME type but must be prepared so that it will not be altered during transfer from source to destination. This means that if the first part is not 7 bit, then it needs to be encoded using base64 or quoted-printable. Then this part is processed in the same manner as signedData, but in this case an object with signedData format is created that has an empty message content field. This object is a

detached signature. It is then transfer encoded using base64 to become the second part of the multipart/signed message. This second part has a MIME content type of application and a subtype of pkcs7-signature. Here is a sample message: Content-Type: multipart/signed; protocol="application/pkcs7-signature"; micalg=sha1; boundary=boundary42 —boundary42 Content-Type: text/plain This is a clear-signed message. —boundary42 Content-Type: application/pkcs7-signature; name=smime.p7s Content-Transfer-Encoding: base64 Content-Disposition: attachment; filename=smime.p7s ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6 4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4 7GhIGfHfYT64VQbnj756 —boundary42— The protocol parameter indicates that this is a two-part clear-signed entity. The micalg parameter indicates the type of message digest used. The receiver can verify the signature by taking the message digest of the first part and comparing this to the message digest recovered from the signature in the second part. Registration Request Typically, an application or user will apply to a certification authority for a public-key certificate. The application/pkcs10 S/MIME 278 chapter 8 / Electronic Mail Security entity is used to transfer a certification request. The certification request includes certificationRequestInfo block, followed by an identifier of the public-key encryption algorithm, followed by the signature of the certificationRequestInfo block, made using the sender's private key. The certificationRequestInfo block includes a name of the certificate subject (the entity whose public key is to be certified) and a bit-string representation of the user's public key. Certificates-Only Message A message containing only certificates or a certificate revocation list (CRL) can be sent in response to a registration request. The message is an application/pkcs7-mime type/subtype with an smime-type parameter of degenerate. The steps involved are the same as those for creating a signedData message, except that there is no message content and the signerInfo field is empty. S/MIME Certificate Processing S/MIME uses public-key certificates that conform to version 3 of X.509 (see Chapter 4). S/MIME managers and/or users must configure each client with a list of trusted keys and with certificate revocation lists. That is, the responsibility is local for maintaining the certificates needed to verify incoming signatures and to encrypt outgoing messages. On the other hand, the certificates are signed by certification authorities. User Agent Role An S/MIME user has several key management functions to perform. ■ Key generation: The user of some related administrative utility (e.g., one associated with LAN management) MUST be capable of generating separate Diffie–Hellman and DSS key pairs and SHOULD be capable of generating RSA key pairs. Each key pair MUST be generated from a good source of nondeterministic random input and be protected in a secure fashion. A user agent SHOULD generate RSA key pairs with a length in the range of 768 to 1024 bits and MUST NOT generate a length of less than 512 bits. ■ Registration: A user's public key must be registered with a certification authority in order to receive an X.509 public-key certificate. ■ Certificate storage and retrieval: A user requires access to a local list of certificates in order to verify incoming signatures and to encrypt outgoing messages. Such a list could be maintained by the user or by some local administrative entity on behalf of a number of users. Enhanced Security Services RFC 2634 defines four enhanced security services for S/MIME: ■ Signed receipts: A signed receipt may be requested in a SignedData object. Returning a signed receipt provides proof of delivery to the originator of a message and allows the originator to demonstrate to a third party that the recipient received the message. In essence, the recipient signs the entire original message plus the original (sender's) signature and appends the new signature to

form a new S/MIME message. 8.5 / Pretty Good Privacy 279 ■ Security labels: A security label may be included in the authenticated attributes of a SignedData object. A security label is a set of security information regarding the sensitivity of the content that is protected by S/MIME encapsulation. The labels may be used for access control, by indicating which users are permitted access to an object. Other uses include priority (secret, confidential, restricted, and so on) or role based, describing which kind of people can see the information (e.g., patient's health-care team, medical billing agents). ■ Secure mailing lists: When a user sends a message to multiple recipients, a certain amount of per-recipient processing is required, including the use of each recipient's public key. The user can be relieved of this work by employing the services of an S/MIME Mail List Agent (MLA). An MLA can take a single incoming message, perform the recipient-specific encryption for each recipient, and forward the message. The originator of a message need only send the message to the MLA with encryption performed using the MLA's public key. ■ Signing certificates: This service is used to securely bind a sender's certificate to their signature through a signing certificate attribute. 8.5 Pretty Good Privacy An alternative e-mail security protocol is Pretty Good Privacy (PGP), which has essentially the same functionality as S/MIME. PGP was created by Phil Zimmerman and implemented as a product first released in 1991. It was made available free of charge and became quite popular for personal use. The initial PGP protocol was proprietary and used some encryption algorithms with intellectual property restrictions. In 1996, version 5.x of PGP was defined in IETF RFC 1991, PGP Message Exchange Formats. Subsequently, OpenPGP was developed as a new standard protocol based on PGP version 5.x. OpenPGP is defined in RFC 4880 (OpenPGP Message Format, November 2007) and RFC 3156 (MIME Security with OpenPGP, August 2001). There are two significant differences between S/MIME and OpenPGP: ■ Key Certification: S/MIME uses X.509 certificates that are issued by Certificate Authorities (or local agencies that have been delegated authority by a CA to issue certificates). In OpenPGP, users generate their own OpenPGP public and private keys and then solicit signatures for their public keys from individuals or organizations to which they are known. Whereas X.509 certificates are trusted if there is a valid PKIX chain to a trusted root, an OpenPGP public key is trusted if it is signed by another OpenPGP public key that is trusted by the recipient. This is called the Web-of-Trust. ■ Key Distribution: OpenPGP does not include the sender's public key with each message, so it is necessary for recipients of OpenPGP messages to separately obtain the sender's public key in order to verify the message. Many organizations post OpenPGP keys on TLS-protected websites: People who wish to verify digital signatures or send these organizations encrypted mail 280 chapter 8 / Electronic Mail Security need to manually download these keys and add them to their OpenPGP clients. Keys may also be registered with the OpenPGP public key servers, which are servers that maintain a database of PGP public keys organized by e-mail address. Anyone may post a public key to the OpenPGP key servers, and that public key may contain any e-mail address. There is no vetting of OpenPGP keys, so users must use the Web-of-Trust to decide whether to trust a given public key. SP 800-177 recommends the use of S/MIME rather than PGP because of the greater confidence in the CA system of verifying public keys. Appendix H provides an overview of PGP. 8.6 DNSSEC DNS Security Extensions (DNSSEC) are used by several protocols that provide e-mail security. This section provides a brief overview of the Domain Name System (DNS) and then looks at DNSSEC. Domain Name System DNS is a directory lookup service that provides a mapping between the name of a host on the Internet and its numeric IP address. DNS is essential to the functioning of the Internet. The

DNS is used by MUAs and MTAs to find the address of the next hop server for mail delivery. Sending MTAs query DNS for the Mail Exchange Resource Record (MX RR) of the recipient's domain (the right hand side of the "@" symbol) in order to find the receiving MTA to contact. Four elements comprise the DNS: ■ Domain name space: DNS uses a tree-structured name space to identify resources on the Internet. ■ DNS database: Conceptually, each node and leaf in the name space tree structure names a set of information (e.g., IP address, name server for this domain name) that is contained in resource record. The collection of all RRs is organized into a distributed database. ■ Name servers: These are server programs that hold information about a portion of the domain name tree structure and the associated RRs. ■ Resolvers: These are programs that extract information from name servers in response to client requests. A typical client request is for an IP address corresponding to a given domain name. The DNS Database DNS is based on a hierarchical database containing resource records (RRs) that include the name, IP address, and other information about hosts. The key features of the database are as follows: ■ Variable-depth hierarchy for names: DNS allows essentially unlimited levels and uses the period (.) as the level delimiter in printed names, as described earlier. 8.6 / DNSSEC 281 ■ Distributed database: The database resides in DNS servers scattered throughout the Internet. ■ Distribution controlled by the database: The DNS database is divided into thousands of separately managed zones, which are managed by separate administrators. Distribution and update of records is controlled by the database software. Using this database, DNS servers provide a name-to-address directory service for network applications that need to locate specific servers. For example, every time an e-mail message is sent or a Web page is accessed, there must be a DNS name lookup to determine the IP address of the e-mail server or Web server. Table 8.6 lists the various types of resource records. DNS Operation DNS operation typically includes the following steps (Figure 8.6): 1. A user program requests an IP address for a domain name. 2. A resolver module in the local host or local ISP queries a local name server in the same domain as the resolver. 3. The local name server checks to see if the name is in its local database or cache, and, if so, returns the IP address to the requestor. Otherwise, the name server queries other available name servers, if necessary going to the root server, as explained subsequently. 4. When a response is received at the local name server, it stores the name/ address mapping in its local cache and may maintain this entry for the amount of time specified in the time-to-live field of the retrieved RR. 5. The user program is given the IP address or an error message. Type Description A A host address. This RR type maps the name of a system to its IPv4 address. Some systems (e.g., routers) have multiple addresses, and there is a separate RR for each. AAAA Similar to A type, but for IPv6 addresses. CNAME Canonical name. Specifies an alias name for a host and maps this to the canonical (true) name. HINFO Host information. Designates the processor and operating system used by the host. MINFO Mailbox or mail list information. Maps a mailbox or mail list name to a host name. MX Mail exchange. Identifies the system(s) via which mail to the queried domain name should be relayed. NS Authoritative name server for this domain. PTR Domain name pointer. Points to another part of the domain name space. SOA Start of a zone of authority (which part of naming hierarchy is implemented). Includes parameters related to this zone. SRV For a given service provides name of server or servers in domain that provide that service. TXT Arbitrary text. Provides a way to add text comments to the database. WKS Well-known services. May list the application services available at this host. Table 8.6 Resource Record Types 282 chapter 8 / Electronic Mail Security The distributed DNS database that supports the DNS functionality must be

updated frequently because of the rapid and continued growth of the Internet. Further, the DNS must cope with dynamic assignment of IP addresses, such as is done for home DSL users by their ISP. Accordingly, dynamic updating functions for DNS have been defined. In essence, DNS name servers automatically send out updates to other relevant name servers as conditions warrant. DNS Security Extensions DNSSEC provides end-to-end protection through the use of digital signatures that are created by responding zone administrators and verified by a recipient's resolver software. In particular, DNSSEC avoids the need to trust intermediate name servers and resolvers that cache or route the DNS records originating from the responding zone administrator before they reach the source of the query. DNSSEC consists of a set of new resource record types and modifications to the existing DNS protocol, and is defined in the following documents: ■ RFC 4033, DNS Security Introduction and Requirements: Introduces the DNS security extensions and describes their capabilities and limitations. The document also discusses the services that the DNS security extensions do and do not provide. ■ RFC 4034, Resource Records for the DNS Security Extensions: Defines four new resource records that provide security for DNS. Figure 8.6 DNS Name Resolution User program User system Internet user query query query user response response response Name resolver Cache Name server Cache Database Database Foreign name server Cache 8.6 / DNSSEC 283 ■ RFC 4035, Protocol Modifications for the DNS Security Extensions: Defines the concept of a signed zone, along with the requirements for serving and resolving by using DNSSEC. These techniques allow a security-aware resolver to authenticate both DNS resource records and authoritative DNS error indications. DNSSEC Operation In essence, DNSSEC is designed to protect DNS clients from accepting forged or altered DNS resource records. It does this by using digital signatures to provide: ■ Data origin authentication: Ensures that data has originated from the correct source. ■ Data integrity verification: Ensures that the content of a RR has not been modified. The DNS zone administrator digitally signs every Resource Record set (RRset) in the zone, and publishes this collection of digital signatures, along with the zone administrator's public key, in the DNS itself. In DNSSEC, trust in the public key (for signature verification) of the source is established not by going to a third party or a chain of third parties (as in public key infrastructure [PKI] chaining), but by starting from a trusted zone (such as the root zone) and establishing the chain of trust down to the current source of response through successive verifications of signature of the public key of a child by its parent. The public key of the trusted zone is called the trust anchor. Resource Records for DNSSEC RFC 4034 defines four new DNS resource records: ■ DNSKEY: Contains a public key. ■ RRSIG: A resource record digital signature. ■ NSEC: Authenticated denial of existence record. ■ DS: Delegation signer. An RRSIG is associated with each RRset, where an RRset is the set of resource records that have the same label, class, and type. When a client requests data, an RRset is returned, together with the associated digital signature in an RRSIG record. The client obtains the relevant DNSKEY public key and verifies the signature for this RRset. DNSSEC depends on establishing the authenticity of the DNS hierarchy leading to the domain name in question, and thus its operation depends on beginning the use of cryptographic digital signatures in the root zone. The DS resource record facilitates key signing and authentication between DNS zones to create an authentication chain, or trusted sequence of signed data, from the root of the DNS tree down to a specific domain name. To secure all DNS lookups, including those for non-existent domain names and record types, DNSSEC uses the NSEC resource record to authenticate negative responses to queries. NSEC is used to identify the 284 chapter 8 / Electronic Mail Security range of DNS names or resource record types that do

not exist among the sequence of domain names in a zone. 8.7 DNS-Based Authentication of Named Entities DANE is a protocol to allow X.509 certificates, commonly used for Transport Layer Security (TLS), to be bound to DNS names using DNSSEC. It is proposed in RFC 6698 as a way to authenticate TLS client and server entities without a certificate authority (CA). The rationale for DANE is the vulnerability of the use of CAs in a global PKI system. Every browser developer and operating system supplier maintains a list of CA root certificates as trust anchors. These are called the software's root certificates and are stored in its root certificate store. The PKIX procedure allows a certificate recipient to trace a certificate back to the root. So long as the root certificate remains trustworthy, and the authentication concludes successfully, the client can proceed with the connection. However, if any of the hundreds of CAs operating on the Internet is compromised, the effects can be widespread. The attacker can obtain the CA's private key, get issued certificates under a false name, or introduce new bogus root certificates into a root certificate store. There is no limitation of scope for the global PKI and a compromise of a single CA damages the integrity of the entire PKI system. In addition, some CAs have engaged in poor security practices. For example, some CAs have issued wildcard certificates that allow the holder to issue sub-certificates for any domain or entity, anywhere in the world. The purpose of DANE is to replace reliance on the security of the CA system with reliance on the security provided by DNSSEC. Given that the DNS administrator for a domain name is authorized to give identifying information about the zone, it makes sense to allow that administrator to also make an authoritative binding between the domain name and a certificate that might be used by a host at that domain name. TLSA Record DANE defines a new DNS record type, TLSA, that can be used for a secure method of authenticating SSL/TLS certificates. The TLSA provides for: ■ Specifying constraints on which CA can vouch for a certificate, or which specific PKIX end-entity certificate is valid. ■ Specifying that a service certificate or a CA can be directly authenticated in the DNS itself. The TLSA RR enables certificate issue and delivery to be tied to a given domain. A server domain owner creates a TLSA resource record that identifies the certificate and its public key. When a client receives an X.509 certificate in the TLS negotiation, it looks up the TLSA RR for that domain and matches the TLSA data against the certificate as part of the client's certificate validation procedure. 8.7 / DNS-Based Authentication of Named Entities 285 Figure 8.7 shows the format of a TLSA RR as it is transmitted to a requesting entity. It contains four fields. The Certificate Usage field defines four different usage models, to accommodate users who require different forms of authentication. The usage models are: ■ PKIX-TA (CA constraint): Specifies which CA should be trusted to authenticate the certificate for the service. This usage model limits which CA can be used to issue certificates for a given service on a host. The server certificate chain must pass PKIX validation that terminates with a trusted root certificate stored in the client. ■ PKIX-EE (service certificate constraint): Defines which specific end entity service certificate should be trusted for the service. This usage model limits which end entity certificate can be used by a given service on a host. The server certificate chain must pass PKIX validation that terminates with a trusted root certificate stored in the client. ■ DANE-TA (trust anchor assertion): Specifies a domain-operated CA to be used as a trust anchor. This usage model allows a domain name administrator to specify a new trust anchor—for example, if the domain issues its own certificates under its own CA that is not expected to be in the end users' collection of trust anchors. The server certificate chain is self-issued and does not need to verify against a trusted root stored in the client. ■ DANE-EE (domain-issued certificate): Specifies a domain-operated CA to be used as a trust

anchor. This certificate usage allows a domain name administrator to issue certificates for a domain without involving a third-party CA. The server certificate chain is self-issued and does not need to verify against a trusted root stored in the client. The first two usage models are designed to co-exist with and strengthen the public CA system. The final two usage models operate without the use of public CAs. The Selector field indicates whether the full certificate will be matched or just the value of the public key. The match is made between the certificate presented in TLS negotiation and the certificate in the TLSA RR. The Matching Type field indicates how the match of the certificate is made. The options are exact match, SHA-256 hash match, or SHA-512 hash match. The Certificate Association Data is the raw certificate data in hex format. Figure 8.7 TLSA RR Transmission Format Certi cate usage Selector Matching type Certi cate association data 0Bit: 8 16 24 31 286 chapter 8 / Electronic Mail Security Use of DANE for SMTP DANE can be used in conjunction with SMTP over TLS, as provided by STARTTLS, to more fully secure e-mail delivery. DANE can authenticate the certificate of the SMTP submission server that the user's mail client (MUA) communicates with. It can also authenticate the TLS connections between SMTP servers (MTAs). The use of DANE with SMTP is documented in an Internet Draft (SMTP Security via Opportunistic DANE TLS, draft-ietf-dane-smtp-with-dane-19, May 29, 2015). As discussed in Section 8.1, SMTP can use the STARTTLS extension to run SMTP over TLS, so that the entire e-mail message plus SMTP envelope are encrypted. This is done opportunistically, that is, if both sides support STARTTLS. Even when TLS is used to provide confidentiality, it is vulnerable to attack in the following ways: ■ Attackers can strip away the TLS capability advertisement and downgrade the connection to not use TLS. ■ TLS connections are often unauthenticated (e.g., the use of self-signed certificates as well as mismatched certificates is common). DANE can address both these vulnerabilities. A domain can use the presence of the TLSA RR as an indicator that encryption must be performed, thus preventing malicious downgrade. A domain can authenticate the certificate used in the TLS connection setup using a DNSSEC-signed TLSA RR. Use of DNSSEC for S/MIME DNSSEC can be used in conjunction with S/MIME to more fully secure e-mail delivery, in a manner similar to the DANE functionality. This use is documented in an Internet Draft (Using Secure DNS to Associate Certificates with Domain Names for S/MIME, draft-ietf-dane-smime-09, August 27, 2015), which proposes a new SMIMEA DNS RR. The purpose of the SMIMEA RR is to associate certificates with DNS domain names. As discussed in Section 8.4, S/MIME messages often contain certificates that can assist in authenticating the message sender and can be used in encrypting messages sent in reply. This feature requires that the receiving MUA validate the certificate associated with the purported sender. SMIMEA RRs can provide a secure means of doing this validation. In essence, the SMIMEA RR will have the same format and content as the TLSA RR, with the same functionality. The difference is that it is geared to the needs of MUAs in dealing with domain names as specified in e-mail addresses in the message body, rather than domain names specified in the outer SMTP envelope. 8.8 Sender Policy Framework SPF is the standardized way for a sending domain to identify and assert the mail senders for a given domain. The problem that SPF addresses is the following: With the current e-mail infrastructure, any host can use any domain name for each of the 8.8 / Sender Policy Framework 287 various identifiers in the mail header, not just the domain name where the host is located. Two major drawbacks of this freedom are: ■ It is a major obstacle to reducing unsolicited bulk e-mail (UBE), also known as spam. It makes it difficult for mail handlers to filter out e-mails on the basis of known UBE sources. ■ ADMDs (see Section 8.1) are understandably concerned about the ease with which other

entities can make use of their domain names, often with malicious intent. RFC 7208 defines the SPF. It provides a protocol by which ADMDs can authorize hosts to use their domain names in the “MAIL FROM” or “HELO” identities. Compliant ADMDs publish Sender Policy Framework (SPF) records in the DNS specifying which hosts are permitted to use their names, and compliant mail receivers use the published SPF records to test the authorization of sending Mail Transfer Agents (MTAs) using a given “HELO” or “MAIL FROM” identity during a mail transaction. SPF works by checking a sender’s IP address against the policy encoded in any SPF record found at the sending domain. The sending domain is the domain used in the SMTP connection, not the domain indicated in the message header as displayed in the MUA. This means that SPF checks can be applied before the message content is received from the sender. Figure 8.8 is an example in which SPF would come into play. Assume that the sender’s IP address is 192.168.0.1. The message arrives from the MTA with domain mta.example.net. The sender uses the MAIL FROM tag of alice@example.org, indicating that the message originates in the example.org domain. But the message header specifies alice.sender@example.net. The receiver uses SPF to query for the SPF RR that corresponds to example.com to check if the IP address 192.168.0.1 is S: 220 foo.com Simple Mail Transfer Service Ready C: HELO mta.example.net S: 250 OK C: MAIL FROM: S: 250 OK C: RCPT TO: S: 250 OK C: DATA S: 354 Start mail input; end with . C: To: bob@foo.com C: From: alice.sender@example.net C: Date: Today C: Subject: Meeting Today . . . Figure 8.8 Example in which SMTP Envelope Header Does Not Match Message Header 288 chapter 8 / Electronic Mail Security Tag Description ip4 Specifies an IPv4 address or range of addresses that are authorized senders for a domain. ip6 Specifies an IPv6 address or range of addresses that are authorized senders for a domain. mx Asserts that the listed hosts for the Mail Exchange RRs are also valid senders for the domain. include Lists another domain where the receiver should look for an SPF RR for further senders. This can be useful for large organizations with many domains or sub-domains that have a single set of shared senders. The include mechanism is recursive, in that the SPF check in the record found is tested in its entirety before proceeding. It is not simply a concatenation of the checks. all Matches every IP address that has not otherwise been matched. (a) SPF Mechanisms Modifier Description + The given mechanism check must pass. This is the default mechanism and does not need to be explicitly listed. - The given mechanism is not allowed to send e-mail on behalf of the domain. ~ The given mechanism is in transition and if an e-mail is seen from the listed host/IP address, then it should be accepted but marked for closer inspection. ? The SPF RR explicitly states nothing about the mechanism. In this case, the default behavior is to accept the e-mail. (This makes it equivalent to = + > unless some sort of discrete or aggregate message review is conducted.) (b) SPF Mechanism Modifiers Table 8.7 Common SPF Mechanisms and Modifiers listed as a valid sender, and then takes appropriate action based on the results of checking the RR. SPF on the Sender Side A sending domain needs to identify all the senders for a given domain and add that information into the DNS as a separate resource record. Next, the sending domain encodes the appropriate policy for each sender using the SPF syntax. The encoding is done in a TXT DNS resource record as a list of mechanisms and modifiers. Mechanisms are used to define an IP address or range of addresses to be matched, and modifiers indicate the policy for a given match. Table 8.7 lists the most important mechanisms and modifiers used in SPF. The SPF syntax is fairly complex and can express complex relationships between senders. For more detail, see RFC 7208. SPF on the Receiver Side If SPF is implemented at a receiver, the SPF entity uses the SMTP envelope MAIL FROM: address

domain and the IP address of the sender to query an SPF TXT RR. The SPF checks can be started before the body of the e-mail message is received, 8.9 / Domainkeys Identified Mail 289 which may result in blocking the transmission of the e-mail content. Alternatively, the entire message can be absorbed and buffered until all the checks are finished. In either case, checks must be completed before the mail message is sent to the end user's inbox. The checking involves the following rules: 1. If no SPF TXT RR is returned, the default behavior is to accept the message. 2. If the SPF TXT RR has formatting errors, the default behavior is to accept the message. 3. Otherwise the mechanisms and modifiers in the RR are used to determine disposition of the e-mail message. Figure 8.9 illustrates SPF operation. 8.9 Domainkeys Identified Mail DomainKeys Identified Mail (DKIM) is a specification for cryptographically signing e-mail messages, permitting a signing domain to claim responsibility for a message in the mail stream. Message recipients (or agents acting in their behalf) can verify the signature by querying the signer's domain directly to retrieve the appropriate public key and thereby can confirm that the message was attested to by a party in possession of the private key for the signing domain. DKIM is an Internet Standard (RFC 6376: DomainKeys Identified Mail (DKIM) Signatures). DKIM has been widely adopted by a range of e-mail providers, including corporations, government agencies, gmail, Yahoo!, and many Internet Service Providers (ISPs). Figure 8.9 Sender Policy Framework Operation Sender Inbound mail server SPF record lookup Authorization pass/fail Further policy checks Inbox Junk e-mail Quarantine Block/delete DNS Internet 290 chapter 8 / Electronic Mail Security E-mail Threats RFC 4686 (Analysis of Threats Motivating DomainKeys Identified Mail) describes the threats being addressed by DKIM in terms of the characteristics, capabilities, and location of potential attackers. Characteristics RFC 4686 characterizes the range of attackers on a spectrum of three levels of threat. 1. At the low end are attackers who simply want to send e-mail that a recipient does not want to receive. The attacker can use one of a number of commercially available tools that allow the sender to falsify the origin address of messages. This makes it difficult for the receiver to filter spam on the basis of originating address or domain. 2. At the next level are professional senders of bulk spam mail. These attackers often operate as commercial enterprises and send messages on behalf of third parties. They employ more comprehensive tools for attack, including Mail Transfer Agents (MTAs) and registered domains and networks of compromised computers (zombies), to send messages and (in some cases) to harvest addresses to which to send. 3. The most sophisticated and financially motivated senders of messages are those who stand to receive substantial financial benefit, such as from an e-mail-based fraud scheme. These attackers can be expected to employ all of the above mechanisms and additionally may attack the Internet infrastructure itself, including DNS cache-poisoning attacks and IP routing attacks. Capabilities RFC 4686 lists the following as capabilities that an attacker might have. 1. Submit messages to MTAs and Message Submission Agents (MSAs) at multiple locations in the Internet. 2. Construct arbitrary Message Header fields, including those claiming to be mailing lists, resenders, and other mail agents. 3. Sign messages on behalf of domains under their control. 4. Generate substantial numbers of either unsigned or apparently signed messages that might be used to attempt a denial-of-service attack. 5. Resend messages that may have been previously signed by the domain. 6. Transmit messages using any envelope information desired. 7. Act as an authorized submitter for messages from a compromised computer. 8. Manipulation of IP routing. This could be used to submit messages from specific IP addresses or difficult-to-trace addresses, or to cause diversion of messages to a specific domain. 9. Limited influence over portions of DNS using

mechanisms such as cache poisoning. This might be used to influence message routing or to falsify advertisements of DNS-based keys or signing practices. 8.9 / Domainkeys Identified Mail 291 10. Access to significant computing resources, for example, through the conscription of worm-infected "zombie" computers. This could allow the "bad actor" to perform various types of brute-force attacks. 11. Ability to eavesdrop on existing traffic, perhaps from a wireless network. Location DKIM focuses primarily on attackers located outside of the administrative units of the claimed originator and the recipient. These administrative units frequently correspond to the protected portions of the network adjacent to the originator and recipient. It is in this area that the trust relationships required for authenticated message submission do not exist and do not scale adequately to be practical. Conversely, within these administrative units, there are other mechanisms (such as authenticated message submission) that are easier to deploy and more likely to be used than DKIM. External bad actors are usually attempting to exploit the "any-toany" nature of e-mail that motivates most recipient MTAs to accept messages from anywhere for delivery to their local domain. They may generate messages without signatures, with incorrect signatures, or with correct signatures from domains with little traceability. They may also pose as mailing lists, greeting cards, or other agents that legitimately send or resend messages on behalf of others. DKIM Strategy DKIM is designed to provide an e-mail authentication technique that is transparent to the end user. In essence, a user's e-mail message is signed by a private key of the administrative domain from which the e-mail originates. The signature covers all of the content of the message and some of the RFC 5322 message headers. At the receiving end, the MDA can access the corresponding public key via a DNS and verify the signature, thus authenticating that the message comes from the claimed administrative domain. Thus, mail that originates from somewhere else but claims to come from a given domain will not pass the authentication test and can be rejected. This approach differs from that of S/MIME and PGP, which use the originator's private key to sign the content of the message. The motivation for DKIM is based on the following reasoning:2 1. S/MIME depends on both the sending and receiving users employing S/MIME. For almost all users, the bulk of incoming mail does not use S/MIME, and the bulk of the mail the user wants to send is to recipients not using S/MIME. 2. S/MIME signs only the message content. Thus, RFC 5322 header information concerning origin can be compromised. 3. DKIM is not implemented in client programs (MUAs) and is therefore transparent to the user; the user need not take any action. 4. DKIM applies to all mail from cooperating domains. 5. DKIM allows good senders to prove that they did send a particular message and to prevent forgers from masquerading as good senders. 2 The reasoning is expressed in terms of the use of S/MIME. The same argument applies to PGP. 292 chapter 8 / Electronic Mail Security Figure 8.10 Simple Example of DKIM Deployment Mail origination network Mail delivery network DNS Public key query/response DNS = Domain Name System MDA = Mail Delivery Agent MSA = Mail Submission Agent MTA = Message Transfer Agent MUA = Message User Agent SMTP MUA MUA SMTP SMTP Signer Veri•er SMTP POP, IMAP MTA MSA MTA DNS MDA Figure 8.10 is a simple example of the operation of DKIM. We begin with a message generated by a user and transmitted into the MHS to an MSA that is within the user's administrative domain. An e-mail message is generated by an e-mail client program. The content of the message, plus selected RFC 5322 headers, is signed by the e-mail provider using the provider's private key. The signer is associated with a domain, which could be a corporate local network, an ISP, or a public e-mail facility such as gmail. The signed message then passes through the Internet via a sequence of MTAs. At the destination, the MDA retrieves the public key for the incoming

signature and verifies the signature before passing the message on to the destination e-mail client. The default signing algorithm is RSA with SHA-256. RSA with SHA-1 also may be used. DKIM Functional Flow Figure 8.11 provides a more detailed look at the elements of DKIM operation. Basic message processing is divided between a signing Administrative Management Domain (ADMD) and a verifying ADMD. At its simplest, this is between the originating ADMD and the delivering ADMD, but it can involve other ADMDs in the handling path. Signing is performed by an authorized module within the signing ADMD and uses private information from a Key Store. Within the originating ADMD, 8.9 / Domainkeys Identified Mail 293 this might be performed by the MUA, MSA, or an MTA. Verifying is performed by an authorized module within the verifying ADMD. Within a delivering ADMD, verifying might be performed by an MTA, MDA or MUA. The module verifies the signature or determines whether a particular signature was required. Verifying the signature uses public information from the Key Store. If the signature passes, reputation information is used to assess the signer and that information is passed to the message filtering system. If the signature fails or there is no signature using the author's domain, information about signing practices related to the author can be retrieved remotely and/or locally, and that information is passed to the message filtering system. For example, if the sender (e.g., gmail) uses DKIM but no DKIM signature is present, then the message may be considered fraudulent. Figure 8.11 DKIM Functional Flow Originating or relaying ADMD: Sign message with SDID RFC 5322 message yes pass fail no Relaying or delivering ADMD: Message signed? Verify signature Private key store (paired) Public key store Remote sender practices Local info on sender practices Reputation/ accreditation information Assessments Message ltering engine Check signing practices Internet 294 chapter 8 / Electronic Mail Security The signature is inserted into the RFC 5322 message as an additional header entry, starting with the keyword Dkim-Signature. You can view examples from your own incoming mail by using the View Long Headers (or similar wording) option for an incoming message. Here is an example: Dkim-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed; d=gmail.com; s=gamma; h=domainkey- signature:mime-version:received:date: message-id:subject :from:to:content-type: content-transfer-encoding; bh=5mZvQDyCRuyLb1Y28K4zgS2MPOemFToDBgvbJ 7GO90s=; b=PcUvPSDygb4ya5Dyj1rbZGp/VyRiScuaz7TTG J5qW5slM+klzv6kcfYdGDHzEVJW+Z FetuPfF1ETOVhELtwH0zjSccOyPkEiblOf6gILO bm3DDRm3Ys1/FVrbhVOlA+/jH9Aei uIlw/5iFnRbSH6qPDVv/beDQqAWQfA/wF7O5k= Before a message is signed, a process known as canonicalization is performed on both the header and body of the RFC 5322 message. Canonicalization is necessary to deal with the possibility of minor changes in the message made en route, including character encoding, treatment of trailing white space in message lines, and the "folding" and "unfolding" of header lines. The intent of canonicalization is to make a minimal transformation of the message (for the purpose of signing; the message itself is not changed, so the canonicalization must be performed again by the verifier) that will give it its best chance of producing the same canonical value at the receiving end. DKIM defines two header canonicalization algorithms ("simple" and "relaxed") and two for the body (with the same names). The simple algorithm tolerates almost no modification, while the relaxed algorithm tolerates common modifications. The signature includes a number of fields. Each field begins with a tag consisting of a tag code followed by an equals sign and ends with a semicolon. The fields include the following: ■ v= DKIM version/ ■ a= Algorithm used to generate the signature; must be either rsa-sha1 or rsa-sha256 ■ c= Canonicalization method used on the header and the body. ■ d= A domain name used as an identifier to refer to the identity of a responsible person or organization.

In DKIM, this identifier is called the Signing Domain IDentifier (SDID). In our example, this field indicates that the sender is using a gmail address. ■ s= In order that different keys may be used in different circumstances for the same signing domain (allowing expiration of old keys, separate departmental signing, or the like), DKIM defines a selector (a name associated with a key) that is used by the verifier to retrieve the proper key during signature verification. 8.10 / Domain-Based Message Authentication 295 ■ h= Signed Header fields. A colon-separated list of header field names that identify the header fields presented to the signing algorithm. Note that in our example above, the signature covers the domainkey-signature field. This refers to an older algorithm (since replaced by DKIM) that is still in use. ■ bh= The hash of the canonicalized body part of the message. This provides additional information for diagnosing signature verification failures. ■ b= The signature data in base64 format; this is the encrypted hash code. 8.10 Domain-Based Message Authentication, Reporting, and Conformance Domain-Based Message Authentication, Reporting, and Conformance (DMARC) allows e-mail senders to specify policy on how their mail should be handled, the types of reports that receivers can send back, and the frequency those reports should be sent. It is defined in RFC 7489 (Domain-based Message Authentication, Reporting, and Conformance, March 2015). DMARC works with SPF and DKIM. SPF and DKM enable senders to advise receivers, via DNS, whether mail purporting to come from the sender is valid, and whether it should be delivered, flagged, or discarded. However, neither SPF nor DKIM include a mechanism to tell receivers if SPF or DKIM are in use, nor do they have feedback mechanism to inform senders of the effectiveness of the anti-spam techniques. For example, if a message arrives at a receiver without a DKIM signature, DKIM provides no mechanism to allow the receiver to learn if the message is authentic but was sent from a sender that did not implement DKIM, or if the message is a spoof. DMARC addresses these issues essentially by standardizing how e-mail receivers perform e-mail authentication using SPF and DKIM mechanisms. Identifier Alignment DKIM, SPF, and DMARC authenticate various aspects of an individual message. DKIM authenticates the domain that affixed a signature to the message. SPF focuses on the SMTP envelope, defined in RFC 5321. It can authenticate either the domain that appears in the MAIL FROM portion of the SMTP envelope or the HELO domain, or both. These may be different domains, and they are typically not visible to the end user. DMARC authentication deals with the From domain in the message header, as defined in RFC 5322. This field is used as the central identity of the DMARC mechanism because it is a required message header field and therefore guaranteed to be present in compliant messages, and most MUAs represent the RFC 5322 From field as the originator of the message and render some or all of this header field's content to end users. The e-mail address in this field is the one used by end users to identify the source of the message and therefore is a prime target for abuse. DMARC requires that From address match (be aligned with) an Authenticated Identifier from DKIM or SPF. In the case of DKIM, the match is made between the DKIM signing domain and the From domain. In the case of SPF, the match is between the SPF-authenticated domain and the From domain. 296 chapter 8 / Electronic Mail Security DMARC on the Sender Side A mail sender that uses DMARC must also use SPF or DKIM, or both. The sender posts a DMARC policy in the DNS that advises receivers on how to treat messages that purport to originate from the sender's domain. The policy is in the form of a DNS TXT resource record. The sender also needs to establish e-mail addresses to receive aggregate and forensic reports. As these e-mail addresses are published unencrypted in the DNS TXT RR, they are easily discovered, leaving the poster subject to unsolicited bulk e-mail. Thus, the poster of the DNS TXT RR

needs to employ some kind of abuse countermeasures. Similar to SPF and DKIM, the DMARC policy in the TXT RR is encoded in a series of tag=value pairs separated by semicolons. Table 8.8 describes the common tags. Once the DMARC RR is posted, messages from the sender are typically processed as follows: 1. The domain owner constructs an SPF policy and publishes it in its DNS database. The domain owner also configures its system for DKIM signing. Finally, the domain owner publishes via the DNS a DMARC message-handling policy. 2. The author generates a message and hands the message to the domain owner's designated mail submission service. 3. The submission service passes relevant details to the DKIM signing module in order to generate a DKIM signature to be applied to the message. 4. The submission service relays the now-signed message to its designated transport service for routing to its intended recipient(s). DMARC on the Receiver Side A message generated on the sender side may pass through other relays but eventually arrives at a receiver's transport service. The typical processing order for DMARC on the receiving side is the following: 1. The receiver performs standard validation tests, such as checking against IP blocklists and domain reputation lists, as well as enforcing rate limits from a particular source. 2. The receiver extracts the RFC 5322 From address from the message. This must contain a single, valid address or else the mail is refused as an error. 3. The receiver queries for the DMARC DNS record based on the sending domain. If none exists, terminate DMARC processing. 4. The receiver performs DKIM signature checks. If more than one DKIM signature exists in the message, one must verify. 5. The receiver queries for the sending domain's SPF record and performs SPF validation checks. 6. The receiver conducts Identifier Alignment checks between the RFC 5321 From and the results of the SPF and DKIM records (if present). 8.10 / Domain-Based Message Authentication 297 Tag (Name) Description v= (Version) Version field that must be present as the first element. By default the value is always DMARC1. p= (Policy) Mandatory policy field. May take values none or quarantine or reject. This allows for a gradually tightening policy where the sender domain recommends no specific action on mail that fails DMARC checks (p= none), through treating failed mail as suspicious (p= quarantine), to rejecting all failed mail (p= reject), preferably at the SMTP transaction stage. aspf= (SPF Policy) Values are r (default) for relaxed and s for strict SPF domain enforcement. Strict alignment requires an exact match between the From address domain and the (passing) SPF check must exactly match the MailFrom address (HELO address). Relaxed requires that only the From and MailFrom address domains be in alignment. For example, the MailFrom address domain smtp.example.org and the From address announce@example.org are in alignment, but not a strict match. adkim= (DKIM Policy) Optional. Values are r (default) for relaxed and s for strict DKIM domain enforcement. Strict alignment requires an exact match between the From domain in the message header and the DKIM domain presented in the (d= DKIM), tag. Relaxed requires only that the domain part is in alignment (as in aspf). fo= (Failure reporting options) Optional. Ignore if a ruf argument is not also present. Value 0 indicates the receiver should generate a DMARC failure report if all underlying mechanisms fail to produce an aligned pass result. Value 1 means generate a DMARC failure report if any underlying mechanism produces something other than an aligned pass result. Other possible values are d (generate a DKIM failure report if a signature failed evaluation), and s (generate an SPF failure report if the message failed SPF evaluation). These values are not exclusive and may be combined. ruf= Optional, but requires the fo argument to be present. Lists a series of URIs (currently just mailto:) that list where to send forensic feedback reports. This is for reports on message-specific failures. rua= Optional list of URIs (like in ruf= , using the mailto: URI) listing where

to send aggregate feedback back to the sender. These reports are sent based on the interval requested using the ri= option with a default of 86400 seconds if not listed. ri= (Reporting interval) Optional with the default value of 86400 seconds. The value listed is the reporting interval desired by the sender. pct= (Percent) Optional with the default value of 100. Expresses the percentage of a sender's mail that should be subject to the given DMARC policy. This allows senders to ramp up their policy enforcement gradually and prevent having to commit to a rigorous policy before getting feedback on their existing policy. sp= (Receiver Policy) Optional with a default value of none. Other values include the same range of values as the p= argument. This is the policy to be applied to mail from all identified subdomains of the given DMARC RR. Table 8.8 DMARC Tag and Value Descriptions 7. The results of these steps are passed to the DMARC module along with the Author's domain. The DMARC module attempts to retrieve a policy from the DNS for that domain. If none is found, the DMARC module determines the organizational domain and repeats the attempt to retrieve a policy from the DNS. 8. If a policy is found, it is combined with the Author's domain and the SPF and DKIM results to produce a DMARC policy result (a "pass" or "fail") and can optionally cause one of two kinds of reports to be generated. 298 chapter 8 / Electronic Mail Security Figure 8.12 DMARC Functional Flow DKIM DKIM SPF SPF Failure report Block Pass Sender Receiver Fail Quarantine Author composes and sends e-mail Standard processing (including antispam) Sending mail server attaches DKIM signature Standard validation tests at receiver (including IP blocklists, reputation, rate limits, etc) Retrieve veried DKIM domains Retrieve "envelope from" via SPF Update periodic aggregate report to be sent to sender Apply DMARC policy 9. Recipient transport service either delivers the message to the recipient inbox or takes other local policy action based on the DMARC result. 10. When requested, Recipient transport service collects data from the message delivery session to be used in providing feedback. Figure 8.12, based on one at DMARC.org, summarizes the sending and receiving functional flow. 8.10 / Domain-Based Message Authentication 299 DMARC Reports DMARC reporting provides the sender's feedback on their SPF, DKIM, Identifier Alignment, and message disposition policies, which enable the sender to make these policies more effective. Two types of reports are sent: aggregate reports and forensic reports. Aggregate reports are sent by receivers periodically and include aggregate figures for successful and unsuccessful message authentications, including: ■ The sender's DMARC policy for that interval. ■ The message disposition by the receiver (i.e., delivered, quarantined, rejected). ■ SPF result for a given SPF identifier. ■ DKIM result for a given DKIM identifier. ■ Whether identifiers are in alignment or not. ■ Results classified by sender subdomain. ■ The sending and receiving domain pair. ■ The policy applied, and whether this is different from the policy requested. ■ The number of successful authentications. ■ Totals for all messages received. This information enables the sender to identify gaps in e-mail infrastructure and policy. SP 800-177 recommends that a sending domain begin by setting a DMARC policy of p= none, so that the ultimate disposition of a message that fails some check is determined by the receiver's local policy. As DMARC aggregate reports are collected, the sender will have a quantitatively better assessment of the extent to which the sender's e-mail is authenticated by outside receivers, and will be able to set a policy of p = reject, indicating that any message that fails the SPF, DKIM, and alignment checks really should be rejected. From their own traffic analysis, receivers can develop a determination of whether a sender's p = reject policy is sufficiently trustworthy to act on. A forensic report helps the sender refine the component SPF and DKIM mechanisms as well as alerting the sender that their domain is being used as part of a phishing/spam campaign. Forensic