

Index

SYMBOLS

- */ (asterisk, slash) ASN.1 comment suffix, 137**
- ::= (colons, equals sign) ASN.1 tag suffix, 139**
- ! (exclamation mark) SSL message suffix, 373**
- /* (slash, asterisk) ASN.1 comment prefix, 137**

A

Abstract Syntax Notation 1. See ASN.1

accept **method, 371**

AcceptableResponses **structure, 252**

addCertPathChecker **method, 274**

addCertStore **method, 266**

addCRLEntry **method, 242–243**

addExtension **method, 203**

addHandshakeCompletedListener **method, 377, 378**

addIssuer **method, 245**

addIssuerName **method, 245**

addKeyTransRecipient **method, 357**

addProvider **method, 10**

addRequest **method, 254**

AES (Advanced Encryption Standard), 17–18

AESWrapRSAExample **class, 102**

agreement, key, 106, 108

AlgorithmIdentifier **structure, 164–165, 167, 174, 193, 208**

AlgorithmParameterExample **class, 119–120**

AlgorithmParameterGenerator **class**

El Gamal algorithm, using with, 119–120

generateParameters **method, 118**

getInstance **factory pattern, 118**

init **method, 118, 120**

AlgorithmParameters **class**

ASN.1 environment, using in, 165–166, 177, 179–180

getEncryptionAlgorithmParameters **method, 338**

getParameterSpec **method, 34**

symmetric block cipher mode, AlgorithmParameters object creation in, 34

AlgorithmParametersGenerator **object, 118**

AlgorithmParameterSpec **object, 71, 91, 122**

aliases **method, 285, 290**

anchor, trust, 264–265, 267, 278, 306

Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd ed. (Schneier), 24, 429

ARC4 stream cipher, 39–40

ASN.1 (Abstract Syntax Notation 1)

AlgorithmParameters class, using in ASN.1 environment, 165–166, 177, 179–180

BIT STRING type, 141, 145, 154

BMPString type, 142, 145, 154

BOOLEAN type, 141, 145, 154

Bouncy Castle environment

encoding, 153, 162–163

object definition, 156–161

package, 153

tagging, 154, 155–156

type creation, 154–155

certificate extension definition, 195–196, 232

CHOICE type, 148–149

class, tag, 145–146

CLASS type, 149–150

CMS structure, 321–325, 333–335, 345

commenting code, 137

DistinguishedName type, 185

ASN.1 (continued)

ASN.1 (continued)

encoding
 Bouncy Castle environment, 153, 162–163
 IV, converting to ASN.1 binary encoding, 165–166
 key, private, 171, 174–179
 key, public, 171–174, 175
 signature, digital, 166–167, 169–170
 tag value, default, 144
 type annotation, 144

ENUMERATED type, 141, 145, 154

GeneralizedTime type, 141, 145, 154, 195

GeneralString type, 142, 154

GraphicString type, 142

history, 135

IA5String type, 143, 145, 154

INTEGER type, 141, 145, 154

KeyUsage extension definition, 200–201

NULL type, 141, 145, 154

NumericString type, 142, 145, 154

OBJECT IDENTIFIER type, 141, 145, 154, 164, 186

OCTET STRING type, 141, 145, 146, 147, 154

OID, 137, 141, 145, 174

OSI standards, 135

PBE, 177–181

PrintableString type, 142, 145, 154, 186

RDNSequence type, 185

SEQUENCE type, 143, 145, 152, 155, 156

SET type, 143–144, 145, 152–153, 155

standards, 425

syntax, 136–140

tagging, 139, 144–148, 150, 154, 155–156

TeletexString type, 143, 155

type annotation, 144

UniversalString type, 143, 145, 155

UTCTime type, 141, 145, 155, 191

UTF8String type, 143, 145, 155, 186

Utils class, configuring for, 136

VideotexString type, 143

VisibleString type, 143, 145, 155

X509Certificate class based on, 189

ASN.1 Complete (Larmouth), 429

ASN1Dump **class, 162–164**

ASN1Encodable **object, 203**

ASN1EncodableVector **class, 155, 157**

ASN1Set **class, 175**

asterisk, slash (* /) ASN.1 comment suffix, 137

asymmetric key cryptography
 abstraction layer, 88
 interface, key, 85

padding, 93–101
translation, key, 88

Attribute **structure, 165, 209**

AttributeTable **object, 326, 338**

AuthorityKeyIdentifier **extension, 214–215, 218, 239, 243, 301**

AuthorityKeyIdentifierStructure **class, 218**

B

BaseRSAExample **class, 87**

Basic Encoding Rules (BER), 150–151, 152–153, 301

BasicConstraints **extension, 200, 201, 203**

BasicDHExample **class, 107–108**

BasicDSAExample **class, 124–125**

BasicECDHExample **class, 111–112, 114**

BasicECDSAExample **class, 127–128**

BasicOCSPResp **class, 256–258**

BasicOCSPRespGenerator **object, 263**

BasicOCSPResponse **structure, 256, 263**

Bellare, Mihir (“Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols”), 97

BER (Basic Encoding Rules), 150–151, 152–153, 301

BIT STRING **ASN.1 type, 141, 145, 154**

Bleichenbacher, Daniel (Chosen Ciphertext Attacks against Protocols Based on RSA Encryption Standard PKCS #1), 97, 428

Blowfish algorithm, 6, 7

BMPString **ASN.1 type, 142, 145, 154**

BOOLEAN **ASN.1 type, 141, 145, 154**

Bouncy Castle provider
 API JAR files, 316
 ASN.1
 encoding, 153, 162–163
 object definition, 156–161
 package, 153
 tagging, 154, 155–156
 type creation, 154–155
 certificate handling, 191, 193
 cipher, asymmetric, 417
 CMS implementation, 316
 elliptic curve cryptography support, 421–423
 engine class, 3
 installing, 7–10
 java.security file configuration, 8–9
 key agreement algorithm support, 417
 key pair generation using El Gamal algorithm, 118
 keystore object type support, 418
 Lightweight API, 3

MAC algorithm support, 418
 message digest support, 418
 PKIX validation algorithm support, 417
 RSAPrivateKey class in Bouncy Castle environment, 93
 signature algorithm support, 418
 S/MIME implementation, 347, 349
 symmetric block cipher support, 418–419
 symmetric stream cipher support, 419
 Web site, 429
 BouncyCastleProvider **class**, **10**
 bOut **class**, **206**
 build **method**, **275, 276**
 Builder **class**, **295–296, 297**
 buildPath **method**, **318, 331**

C

CA (certificate authority)

KeyStore class JVM CA, 307–308
 password, 209
 requesting certificate from, 208–213
 vendor, 208
 writing certificate provided by, 214–219

cacerts **file**, **308, 313**

CallbackHandler **interface**, **191**

CallbackHandlerProtection **class**, **191**

carriage-return line-feed character (CRLF), **349, 350**

CBC (Cipher Block Chaining) cipher mode, **26–28, 34–35, 56**

Ceiling **function**, **77**

CER (Canonical Encoding Rules), **150**

certGen1 **class**, **194–195**

CertID **structure**, **248–249**

certificate. *See also* **CRL (certificate revocation list)**;

OCSP (Online Certificate Status Protocol)

Bouncy Castle environment, 191, 193
 encoding
 DER, 196, 204, 210
 PEM, 204, 205, 210, 220–221
 end entity, 230, 240
 exception thrown by, 188, 192
 expiration, 234, 238
 extension
 ASN.1 definition, 195–196, 232
 AuthorityKeyIdentifier, 214–215, 218, 239, 243, 301
 BasicConstraints, 200, 201, 203
 CertificateIssuer, 238
 critical, 196, 197, 203

CRL, extension handling in, 232–233, 236–240, 246, 258
 CRLNumber, 239, 246
 DeltaCRLIndicator, 239
 DER encoding, 196
 ExtendedKeyUsage, 200–201, 203, 204
 FreshestCRL, 240
 HoldInstructionCode, 237–238
 InvalidityDate, 238
 IssuerAlternativeName, 239
 IssuerAltName, 200
 IssuingDistributionPoint, 239–240
 KeyUsage, 198–199, 201, 203
 NameConstraints, 264–265
 nonce, 251
 OID, returning, 197
 ReasonCode, 237
 requesting, 209, 211–213
 self-signed certificate, 201–203
 SubjectAltName, 199–200, 201, 213, 218
 SubjectKeyIdentifier, 214, 215, 301
 support information, returning, 197
 type, 196
 value, returning, 197
 version, returning, 190
 X.509 certificate, 188, 189, 190, 195–204
 identifier, unique, 188
 KeyStore class, 283, 288, 290
 path
 carrier class, 219–220
 constraint, specifying when validating, 266, 270
 driver, path validation, 273
 interface, path validation, 265–266
 linkage between certificates in, 219
 output, path validation, 267
 revocation, working with during path validation, 266, 274–275
 scheduling path validation, 266
 status of path validation, returning, 270–271
 store, 266, 275–278
 trust, path validation, 264–265, 269, 278
 type, returning, 267
 profile, 196
 public key, 187–188
 reading, 204–208
 request
 CA, 208–213
 client mode, 218
 creating certificate from certification request, 215–219
 validating, 218

certificate (continued)

- signature
 - self-signed, 193–195, 201–204
 - verifying, 214, 233, 234, 235, 251
- SSL client certificate chain, 369, 378
- store, 219, 246–247, 266, 275–278
- streaming, 204, 205–206, 207, 244–245
- Sun X.509 certificate factory, 311
- trust, 188, 264–265, 269, 278
- Utils class, extending for certificate handling, 228–229
- validation, 214, 229, 264
- writing, 204–208, 214–219

certificate authority. See CA

Certificate **class**, **187–188**

certificate revocation list. See CRL

Certificate **structure**, **193**

CertificateFactory **class**, **204, 207, 243–245, 269**

CertificateFactoryExample **class**, **205**

CertificateID **class**, **249**

CertificateIssuer **extension**, **238**

CertificationRequest **structure**, **208**

CertificationRequestInfo **structure**, **208–209**

CertPath **class**, **197, 220–221**

CertPathBuilder **class**, **275, 276**

CertPathBuilderResult **interface**, **275**

CertPathParameters **interface**, **275**

CertPathValidator **class**, **266–267, 269**

CertPathValidatorExample **class**, **268–269**

CertPathValidatorWithCheckerExample **class**, **273–274**

CertSelector **interface**, **219, 222**

CertStore **class**

- CertPathValidator class, using with, 269
- CertSelector interface, using with, 219
- CMSSignedData class, using with, 327
- getCertificates method, 222
- getInstance factory pattern, 221–222
- retrieving CRL from, 246–247
- X509CertSelector class, using with, 223–224

CertStoreExample **class**, **223–224**

CFB (Cipher Feedback) cipher mode, **38**

check **method**, **271, 273**

checksum, **68**

checkValidity **method**, **192**

Chinese Remainder Theorem (CRT), **92, 93**

CHOICE **ASN.1 type**, **148–149**

Chosen Ciphertext Attacks against Protocols Based on RSA Encryption Standard PKCS #1 (Bleichenbacher), **97, 428**

Cipher Block Chaining (CBC) cipher mode, 26–28, 34–35, 56

Cipher **class**

- DECRYPT_MODE constant, 20
- doFinal method, 20, 23, 54–55
- ENCRYPT_MODE constant, 20
- getBlockSize method, 21, 40
- getInstance factory pattern, 11, 13, 19–20, 28, 34–35
- getIV method, 34
- getOutputSize method, 23, 24
- getParameters method, 34, 166, 179
- init method
 - initializing Cipher class using, 19, 20, 23
 - IV, passing to, 28, 34
 - key, passing to, 19, 40, 47
 - key wrapping, using in, 52
- introduced, 5
- Java runtime key size determination by, 6
- Key object, initialization by, 42
- parameter object, 34
- PRIVATE_KEY constant, 50, 101
- PUBLIC_KEY constant, 50
- SECRET_KEY constant, 50
- unwrap method, 50, 52, 102–103
- UNWRAP_MODE constant, 50
- update method, 19, 20, 23
- wrap method, 50, 102, 179
- WRAP_MODE constant, 50

Cipher Feedback (CFB) cipher mode, **38**

Cipher Text Stealing (CTS) cipher mode, **34–35**

CipherInputStream **class**, **52–55**

CipherMacExample **class**, **72–73**

CipherOutputStream **class**, **52–55, 56**

ciphertext pattern, **25–26, 30, 60**

CLASS **ASN.1 type**, **149–150**

client.p12 **file**, **376**

clone **method**, **220, 245**

close **method**, **206**

CMS (Cryptographic Message Syntax)

- ASN.1 structure, 321–325, 333–335, 345
- Bouncy Castle implementation, 316
- compression, 345–347, 363
- ContentInfo structure, relation to, 318–319, 321
- Data content type, 319
- described, 318
- envelope
 - algorithm, 333
 - ASN.1 Structure, 333–335
 - creating, 338–342

- encoding, 338–340
 - generation process, 332
 - key, using with encrypted, 343–344
 - recipient, 335–337, 338, 340–341, 342
 - signing message, 320–321, 325–332
 - S/MIME classes, 349
 - specification, 318
 - structure, message, 318–319
 - Utils class, configuring for, 316–318
 - CMSCompressedData **class**, 345–346, 347
 - CMSEnvelopedData **class**, 337–338, 340
 - CMSEnvelopedGenerator **object**, 340
 - CMSProcessable **interface**
 - CMSProcessableBodyPart class
 - implementation, 348
 - getContent method, 320
 - write method, 320, 348
 - CMSProcessableBodyPart **class**, 348, 354
 - CMSProcessableBodyPartInbound **class**, 349
 - CMSProcessableBodyPartOutbound **class**, 349
 - CMSProcessableByteArray **class**, 320
 - CMSSignedData **class**
 - CertStore class, using with, 327
 - constructor, 327, 332
 - getCertificatesAndCRLs method, 328
 - getEncoded method, 328
 - getSignedContent method, 328
 - getSignedContentOID method, 328
 - getSignerInfos method, 328
 - replaceSigners method, 328
 - SignedData structure, relation to, 322
 - streaming, 327
 - CMSSignedDataGenerator **class**, 327, 330, 331, 354
 - cofactor**, 114
 - Collection **class**, 245
 - CollectionCertStoreParameters **class**, 224, 245
 - colons, equals sign (::=) ASN.1 tag suffix**, 139
 - CompressedDataExample **class**, 346
 - CompressedMailExample **class**, 362
 - containsAlias **method**, 285
 - ContentInfo **structure**, 298, 299, 318–319, 321, 333
 - Counter (CTR) symmetric block cipher mode**, 35–37, 40–42
 - createChain **method**, 218
 - createCredentials **method**, 318
 - createIssuerAndSerialNumberFor **method**, 348
 - createKeyStore **method**, 289, 294, 303
 - CreateKeyStores **utility**, 368, 373
 - createMimeMessage **method**, 318
 - createMultipartWithSignature **method**, 351, 352, 353, 357, 358
 - createServerSocket **method**, 371
 - createSocket **method**, 370
 - createSSLContext **method**, 385, 387, 389, 394
 - CRL (certificate revocation list)**
 - certificate in CRL, returning, 234
 - client certificate check done on, 231
 - collection, 243, 245
 - date of revocation, returning, 234, 236, 246
 - DER encoding, returning, 236
 - distribution point, 232–233, 239–240
 - downloading, 232
 - exception, 235
 - expiration, 234, 238
 - extension handling, 232–233, 236–240, 246, 258
 - field value, returning, 233–234
 - introduced, 229
 - issuer, adding/returning, 236, 245
 - presence of certificate, verifying, 231
 - reading, 243–245
 - reason subset, 232
 - root certificate, 230
 - scope, 231
 - selector, 245–246
 - serial number, returning, 236
 - signature, returning/verifying, 233, 234, 235
 - store, retrieving from, 246–247
 - type, returning, 231
 - updating certificate, 234
 - version of certificate, returning, 234
 - CRL **class**, 231
 - CRLCertFactoryExample **class**, 244–245
 - CRLCertStoreExample **class**, 246–247
 - CRLF (carriage-return line-feed character)**, 349, 350
 - crlGen **class**, 242–243
 - CRLNumber **extension**, 239, 246
 - CRLSelector **interface**, 245
 - CRT (Chinese Remainder Theorem)**, 92, 93
 - Crypto-Gram newsletter**, 429
 - Cryptographic Message Syntax**. *See* CMS
 - CTR (Counter) symmetric block cipher mode**, 35–37, 40–42
 - CTS (Cipher Text Stealing) cipher mode**, 34–35
- ## D
- Data **CMS content type**, 319
 - dataPart **object**, 357
 - deleteEntry **method**, 285

DeltaCRLIndicator **extension**, 239

DER (Distinguished Encoding Rules)

- BER encoding compared, 152–153
- certificate, DER-encoded, 196, 204, 210
- CRL DER encoding, returning, 236
- extension, 169
- keystore file, 301
- OCSF, 256

derived key, 74

DERSet **object**, 213

DESede (DES-Encrypt-Decrypt-Encrypt), 45, 49

destroy method

- PasswordProtection class, 291
- X500PrivateCredential class, 282

DHGenParameterSpec **class**, 120

DHKey **class**, 110

DHParameterSpec **class**, 109, 118

DHPrivateKey **class**, 110

DHPrivateKeySpec **class**, 109

DHPublicKey **class**, 110

DHPublicKeySpec **class**, 109

dictionary attack, 45

Diffie-Hellman algorithm

- El Gamal algorithm, using with, 118
- elliptic curve cryptography, using with, 110–115
- interface, 110
- JCE support, 109
- key generation, 106, 108, 109, 110–112
- modulus, 106
- specification object, 109
- three-party, 115–116

digest method, 65, 81

DigestInfo **object**, 169

DigestInputStream **class**, 79

DigestOutputStream **class**, 79

digital signature. *See* signature, digital

Digital Signature Algorithm (DSA), 122–128

Digital Signature Standard (DSS), 122

Distinguished Encoding Rules. *See* DER

DistinguishedName **ASN.1 type**, 185

DN (distinguished name)

- DistinguishedName ASN.1 type, 185
- PrintableString type, 186
- RDN, 184–185
- RDNSequence type, 185
- UTF8String type, 186
- X.500 directory structure, place in, 184–185
- X500Principal class, defining using, 186–187

doFinal **method**

- Cipher class, 20, 23, 54–55
- Mac class, 71

DSA (Digital Signature Algorithm), 122–128

DSAKey **class**, 126

DSAParameterSpec **class**, 125–126

DSAPrivateKey **class**, 127

DSAPrivateKeySpec **class**, 126

DSAPublicKey **class**, 127

DSAPublicKeySpec **class**, 126

DSS (Digital Signature Standard), 122

dumpAsString **method**, 162

E

ECB (Electronic Code Book) cipher mode, 25–26, 27–28, 35

ECField **interface**, 112

ECFieldFp **interface**, 112

ECFieldF2m **interface**, 112–113

ECGenParameterSpec **class**, 114–115, 128

ECKeY **interface**, 421

ECNamedCurveParameterSpec **class**, 422–423

ECNamedCurveSpec **class**, 423

ECParameterSpec **class**, 114–115, 423

ECPoint **class**, 113

ECPointEncoder **interface**, 422

ECPrivateKey **interface**, 422

ECPrivateKeySpec **class**, 423

ECPublicKey **interface**, 422

ECPublicKeySpec **class**, 423

El Gamal algorithm, 105, 116–120, 417

Electronic Code Book (ECB) cipher mode, 25–26, 27–28, 35

elliptic curve cryptography, 110–115, 127–128, 421–423

EllipticCurve **class**, 113

EncapsulatedContentInfo **structure**, 324

EncryptedPrivateKeyInfo **class**, 177–181

EncryptedPrivateKeyInfoExample **class**, 178

Entry **interface**, 291–294

entryInstanceOf **method**, 292, 294

ENUMERATED **ASN.1 type**, 141, 145, 154

EnvelopedData **structure**, 333, 340

EnvelopedMailExample **class**, 355–356

EnvelopedSignedMailExample **class**, 358

equals **method**, 186, 219, 336

exclamation mark (!) SSL message suffix, 373

export control, 2, 3

ExtendedKeyUsage **extension**, 200–201, 203, 204

Extension **structure, 196**
 Extensions **structure, 209, 213**

F

fact **class, 207**
factory pattern, 2, 3. See also getInstance **factory pattern**
Ferguson, Niels (Practical Cryptography), 30, 31, 429
 FreshestCRL **extension, 240**

G

GeneralizedTime **ASN.1 type, 141, 145, 154, 195**
 GeneralNames **structure, 199**
 GeneralString **ASN.1 type, 142, 154**
 generate **method, 255, 263, 361**
 generateCertificate **method**
 CertificateFactory class, 204
 fact class, 207
 generateCertificateManagement **method, 354**
 generateCertificates **method, 204, 207**
 generateCertPath **methods, 219, 269**
 generateCRL **method, 243–245**
 generateCRLs **method, 245**
 generateEncapsulated **method, 354**
 generateEndEntityCert **method, 230**
 generateIntermediateCert **method, 228**
 generateKey **method, 43**
 generateKeyPair **method, 91, 120**
 generateOCSPResponse **method, 259–260, 263**
 generateParameters **method, 118**
 generatePrivate **method, 88, 89**
 generatePublic **method, 88**
 generateRootCert **method, 228**
 generateSecret **method, 48**
 GenerateV1Certificate **method, 194**
 genKeyPair **method, 91**
 get **method**
 RecipientInformationStore class, 337
 SignerInformationStore class, 327
 getAffineX **method, 113**
 getAffineY **method, 113**
 getAlgorithm **method**
 CertPathValidator class, 266, 267
 Key interface, 42
 KeyManagerFactory class, 382
 PSource class, 100
 TrustManagerFactory class, 386
 getAlgParameters **method, 177**
 getBasicConstraints **method, 200**

getBlockSize **method, 21, 40**
 getBytes **method, 60**
 getCertID **method, 257**
 getCertificate **method**
 KeyStore class, 285
 PrivateKeyEntry class, 292
 getCertificateAlias **method, 285–286, 342**
 getCertificateChain **method**
 KeyStore class, 286
 PrivateKeyEntry class, 292
 getCertificateIssuer **method, 236**
 getCertificates **method**
 CertPath class, 220
 CertStore class, 222
 OCSPReq class, 251
 getCertificatesAndCRLs **method, 328**
 getCertPath **method, 275**
 getCertPathEncodings **method, 220**
 getCertStatus **method, 257, 258, 264**
 getCipherSuite **method**
 HttpsURLConnection class, 395
 SSLSession interface, 389
 getClientSessionContext **method, 380**
 getCompressedContent **method, 362**
 getContent **method**
 CMSCompressedData class, 345
 CMSProcessable interface, 320
 KeyTransRecipientInformation class, 337
 RecipientInformation class, 336, 342
 SMIMESigned class, 350
 getContentAsMimeMessage **method, 350**
 getContentWithSignature **method, 350**
 getCreationDate **method, 286**
 getCreationTime **method, 389**
 getCriticalExtensionsOIDs **method, 197**
 getCRLs **method, 247**
 getCrtpcoefficient **method**
 RSAPrivateCrtKey class, 93
 RSAPrivateCrtKeySpec class, 93
 getDefault **method, 370, 371**
 getDefaultAlgorithm **method**
 KeyManagerFactory class, 382
 TrustManagerFactory class, 386–387
 getDefaultCipherSuites **method**
 SSLServerSocketFactory class, 371
 SSLSocketFactory class, 370
 getDefaultType **method**
 CertPathValidator class, 266
 KeyStore class, 313
 SSLSocketFactory object, creating using, 370

getDigestAlgOID method

`getDigestAlgOID` **method, 325**
`getDigestAlgorithm` **method, 100**
`getDigestAlgParams` **method, 325**
`getDigestSize` **method, 65**
`getEnabledCipherSuites` **method, 371**
`getEncoded` **method**
 AlgorithmParameters class, 34, 166
 Certificate class, 188
 CertPath class, 220
 CMSCompressedData class, 345–346
 CMSEnvelopedData class, 338
 CMSSignedData class, 328
 EncryptedPrivateKeyInfo class, 177
 Key interface, 42, 52, 102, 172, 176
 X500Principal class, 186, 247
 X509CRL class, 236
`getEncodings` **method, 220**
`getEncryptedContent` **method, 355**
`getEncryptedData` **method, 177**
`getEncryptionAlgOID` **method**
 CMSEnvelopedData class, 338
 SignerInformation class, 326
`getEncryptionAlgorithmParameters` **method, 338**
`getEncryptionAlgParams` **method**
 CMSEnvelopedData class, 338
 SignerInformation class, 326
`getEntry` **method, 291**
`getExtendedKeyUsage` **method, 200–201**
`getExtensions` **method, 263**
`getExtensionValue` **method, 197, 200**
`getFieldSize` **method, 112**
`getFormat` **method, 42, 174**
`getHashAlgOID` **method, 249**
`getHostnameVerifier` **method, 395**
`getId` **method, 390**
`getInstance` **factory pattern**
 AlgorithmParameterGenerator class, 118
 ASN1Set class, 175
 CertStore class, 221–222
 Cipher class, 11, 13, 19–20, 28, 34–35
 KeyAgreement class, 112
 KeyFactory class, 88
 KeyGenerator class, 43
 KeyManagerFactory class, 381
 KeyPairGenerator class, 91
 KeyStore class, 283, 289, 303
 Mac class, 68, 71
 MessageDigest class, 64
 SecretKeyFactory class, 48
 Signature class, 121
 SSLContext class, 380
 TrustManagerFactory class, 386
`getIssuerAlternativeNames` **method, 200**
`getIssuerDN` **method, 191**
`getIssuerKeyHash` **method, 249**
`getIssuerNameHash` **method, 249**
`getIssuerUniqueID` **method, 192**
`getIssuerX500Principal` **method**
 X509Certificate class, 191
 X509CRL class, 234
`getIV` **method, 34**
`getKey` **method, 286**
`getKeyEncryptionAlgOID` **method, 336**
`getKeyEncryptionAlgorithmParameters` **method, 336**
`getKeyEncryptionAlgParams` **method, 336**
`getKeyIdentifier` **method, 342**
`getKeyManagers` **method, 382**
`getKeySpec` **method**
 EncryptedPrivateKeyInfo class, 177
 KeyFactory class, 88
`getKeyStore` **method, 295, 296, 297**
`getKeyUsage` **method, 198–199**
`getKeyUsageForSignature` **method, 331**
`getL` **method, 120**
`getLastAccessedTime` **method, 390**
`getLocalCertificates` **method**
 HttpsURLConnection class, 395
 SSLSession interface, 390
`getLocalPrincipal` **method**
 HttpsURLConnection class, 395
 SSLSession interface, 390
`getModulus` **method**
 RSAPrivateCrtKey class, 92
 RSAPrivateCrtKeySpec class, 92
 RSAPublicKey class, 88
`getName` **method**
 Provider class, 10
 X500Principal class, 186–187
`getNextUpdate` **method**
 SingleResp class, 258
 X509CRL class, 234
`getNonCriticalExtensionOIDs` **method, 197**
`getNotAfter` **method, 191**
`getNotBefore` **method, 191**
`getOutputSize` **method, 23, 24**
`getP` **method, 112**

- getParameters **method**
 - Cipher class, 34, 166, 179
 - Signature class, 122, 170
- getParameterSpec **method, 34**
- getParams **method**
 - DHKey class, 110
 - DSAKey class, 126
- getPassword **method, 291**
- getPeerCertificates **method, 390**
- getPeerHost **method, 390**
- getPeerPort **method, 390**
- getPeerPrincipal **method**
 - HttpsURLConnection class, 395–396
 - SSLSession interface, 390–391
- getPrimeExponentP **method, 93**
- getPrimeExponentQ **method, 93**
- getPrimeP **method, 92**
- getPrimeQ **method, 92**
- getPrivate **method, 90**
- getPrivateExponent **method**
 - RSAPrivateCrtKey class, 92
 - RSAPrivateCrtKeySpec class, 92
 - RSAPrivateKeySpec class, 89
- getPrivateKey **method, 292**
- getProducedAt **method, 257**
- getProtectionParameter **method**
 - Builder class, 295, 296
 - LoadStoreParameter interface, 297
- getProtocol **method**
 - SSLContext class, 380
 - SSLSession interface, 391
- getProvider **method, 10**
- getPublicKey **method**
 - Certificate class, 188
 - PKIXCertPathValidatorResult class, 267
 - SubjectPublicKeyInfo structure, 174
- getRecipientInfos **method, 338**
- getRecipients **method, 337**
- getRequestList **method, 250–251**
- getRequestorName **method, 250, 251**
- getResponderID **method, 257**
- getResponseObject **method, 263**
- getResponses **method, 257–258**
- getRevocationDate **method, 236**
- getRevokedCertificate **method, 235**
- getRevokedCertificates **method, 235**
- getRID **method, 336**
- getSecretKey **method, 292**
- getSerialNumber **method**
 - CertificateID class, 249
 - X509Certificate class, 190–191
 - X509CRLEntry class, 236
- getServerCertificates **method, 396**
- getServerSessionContext **method, 381**
- getServerSocketFactory **method, 381**
- getSessionContext **method, 391**
- getSID **method, 326, 331**
- getSigAlgName **method, 235**
- getSigAlgOID **method**
 - X509Certificate class, 193
 - X509CRL class, 235
- getSigAlgParams **method**
 - X509Certificate class, 193
 - X509CRL class, 235
- getSignature **method**
 - OCSPPReq class, 251
 - SignerInformation class, 326
 - X509Certificate class, 193
 - X509CRL class, 235
- getSignatureAlgOID **method, 251**
- getSignedAttributes **method, 326**
- getSignedContent **method, 328**
- getSignedContentOID **method, 328**
- getSignerInfos **method, 328**
- getSigners **method, 327**
- getSocketFactory **method, 381**
- getSSLSocketFactory **method, 396**
- getStatusMessage **method, 263**
- getSubjectAlternativeNames **method, 199–200**
- getSubjectUniqueID **method, 192**
- getSubjectX500Principal **method, 192**
- getSupportedCipherSuites **method**
 - SSLServerSocketFactory class, 371
 - SSLSocketFactory class, 370
- getSupportedExtensions **method, 271**
- getTBSCertList **method, 233–234**
- getTBSRequest **method, 250**
- getTBSResponseData **method, 256–257**
- getThisUpdate **method**
 - SingleResp class, 257, 258
 - X509CRL class, 234
- getTrustAnchor **method, 267, 278**
- getTrustedCertificate **method, 292**
- getTrustManagers **method, 387**
- getType **method**
 - Certificate class, 187
 - CertPath class, 220
 - CRL class, 231
 - KeyStore class, 286

getUnprotectedAttributes method

getUnprotectedAttributes **method, 338**

getUnsignedAttributes **method, 326**

getValue **method, 100**

getVersion **method**

BasicOCSPResp class, 257

OCSPReq class, 250

X509Certificate class, 190

X509CRL class, 234

getX **method**

DHPrivateKey class, 110

DSAPrivateKey class, 127

getY **method, 110**

GraphicString **ASN.1 type, 142**

H

Handbook of Applied Cryptography (Menezes, van Oorschot, and Vanstone), **124, 429**

handle **method, 191**

HandshakeCompletedEvent **object, 377**

HandshakeCompletedListener **interface, 377–378**

hasExtensions **method, 236**

hash

checksum compared, 68

cryptographic, 57, 68

digital signature based on, 121

HMAC, 68–70, 71

message digest hash calculation, 62, 65

OCSP, returning hash information in, 249

PBE hashing mechanism, 44

hashCode **method, 186, 219, 336**

hasUnsupportedCriticalExtension **method, 197**

HMAC (hash MAC), 68–70, 71

HoldInstructionCode **extension, 237–238**

HostnameVerifier **interface, 396–401**

“How to Implement a Provider for the Java Cryptography Extension”, **4**

HTTPS (Hypertext Transport Protocol Secure), 394–401

HTTPSClientExample **class, 397–400**

HttpsURLConnection **class, 395–396, 397–400**

I

IA5String **ASN.1 type, 143, 145, 154**

IDEA algorithm, 4

IETF (Internet Engineering Task Force) Working Group charter pages, 425–426

init **method**

AlgorithmParameterGenerator class, 118, 120

Cipher class

initializing using init method, 19, 20, 23

IV, passing to, 28, 34

key, passing to, 19, 40, 47

key wrapping, using in, 52

KeyGenerator class, 43

KeyManagerFactory class, 381–382

Mac class, 71

PKIXCertPathChecker class, 270

SSLContext class, 380, 385

TrustManagerFactory class, 386

initialization vector (IV) cipher mode, 28–34, 56, 165–166

Initialize **method, 91, 94, 126**

initSign **method, 121, 125**

initVerify **method, 121, 122, 125**

InlineIvCBCExample **class, 29–30**

InputStream **class, 204, 207, 208**

INTEGER **ASN.1 type, 141, 145, 154**

Internet Engineering Task Force (IETF) Working Group charter pages, 425–426

invalidate **method, 391**

InvalidityDate **extension, 238**

is **method, 266**

isCertificateEntry **method, 286**

isEqual **method, 65**

isForwardCheckingSupported **method, 271**

isKeyEntry **method, 286**

ISO padding, 24

isRevoked **method, 231**

isSigned **method, 251**

IssuerAlternativeName **extension, 239**

IssuerAltName **extension, 200**

IssuerAndSerialNumber **class/structure, 348**

IssuingDistributionPoint **extension, 239–240**

isValid **method, 331, 332, 391**

IV (initialization vector) cipher mode, 28–34, 56, 165–166

IVExample **class, 166**

IvParameterSpec **object, 28, 30**

J

JAF (JavaBeans Activation Framework), 315

JAR (Java archive file)

Bouncy Castle API JAR files, 316

provider installation, 8, 9

signature, digital, 312, 313

jarsigner **tool, 312**

Java Cryptography Architecture (JCA), 2, 3, 248

Java Cryptography Extension. See JCE
“Java Cryptography Extension (JCE) Reference Guide”, 4
Java Development Kit (JDK), 3, 284
Java Native Interface (JNI), 3
Java Secure Socket Extension (JSSE), 365, 366, 369.
 See also **SSL (Secure Sockets Layer)**
Java Virtual Machine (JVM), 307–308, 312–313
JavaBeans Activation Framework (JAF), 315
JavaMail API, 315, 360, 429
 java.policy file, **312–313**
 java.security file, **8–9, 10–11, 14, 313**
java.sun.com Web site, 7, 429
 javax.crypto package, **7**
JCA (Java Cryptography Architecture), 2, 3, 248
JCE (Java Cryptography Extension)
 Diffie-Hellman algorithm support, 109
 downloading, 7
 error message when not installed, 7
 export control, effect on, 2, 3
 JDK shipped with, 3
 JNI interface, built on, 3
 PBE in JCE environment, 45–50
 policy file, unrestricted, 4–7
 provider signing, 4
 Reference Guide, 4
 JCEKSStoreBuilderExample class, **296–297**
 JCEKSStoreEntryExample class, **293–294**
JDK (Java Development Kit), 3, 284
 JKSStoreExample class, **288–289**
JNI (Java Native Interface), 3
JSSE (Java Secure Socket Extension), 365, 366, 369.
 See also **SSL (Secure Sockets Layer)**
JVM (Java Virtual Machine), 307–308, 312–313

K

KEKEnvelopedDataExample class, **343–344**
 KEKRecipientInformation class, **342**
Kerberos cipher suite, 366
Key interface
 getAlgorithm method, 42
 getEncoded method, 42, 52, 102, 172, 176
 getFormat method, 42, 174
key wrapping. See wrapping, key
 KeyAgreement class, **108, 109, 112**
 KeyFactory class, **88, 89**
 KeyGenerator class, **43**
 KeyGeneratorExample class, **41–42**
 KeyManager object, **388**
 KeyManagerFactory class, **381–382, 392**
 KeyPair class, **90**
 KeyPairGenerator class, **91, 96, 120**
 KeyStore class
 aliases method, 285, 290
 BKS type, 284, 295
 Bouncy Castle type, 284–285
 CaseExactJKS type, 284
 certificate, 283, 288, 290
 classes nested in, 290–291
 containsAlias method, 285
 deleteEntry method, 285
 DER, 301
 Entry interface, 291–294
 entryInstanceOf method, 292, 294
 getCertificate method, 285
 getCertificateAlias method, 285–286, 342
 getCertificateChain method, 286
 getCreationDate method, 286
 getDefaultType method, 313
 getEntry method, 291
 getInstance factory pattern, 283, 289, 303
 getKey method, 286
 getType method, 286
 isCertificateEntry method, 286
 isKeyEntry method, 286
 JCEK type, 284
 JDK types, 284
 JKS type, 284, 288–290, 309–310
 JVM CA, 307–308
 load method, 286–287, 288, 290, 295, 297
 PBE, 283, 286–288, 294, 296, 298–299
 PKCS12 type, 284, 285, 298–304, 309–312, 318
 PKIXParameters object, passing to, 265
 policy file entry, 313
 private key entry, 283, 288
 ProtectionParameter interface, 290–291
 setCertificateEntry method, 287
 setEntry method, 291, 292–294, 297
 setKeyEntry method, 287
 size method, 287
 store method, 287–288, 297
 symmetric key entry, 283
 UBER type, 284, 295
 Utils class, configuring for, 281–283
 KeyStoreFileUtility class, **308–309**
 keytool command, **304–307, 308–312**
 KeyTransEnvelopedDataExample class, **339–340**

KeyTransRecipientInformation class/structure

KeyTransRecipientInformation **class/structure**,
336–337

KeyUsage **extension**, **198–199, 201, 203**

L

Larmouth, John (*ASN.1 Complete*), **429**

load **method**, **286–287, 288, 290, 295, 297**

LoadStoreParameter **interface**, **297**

M

MAC **algorithm**, **366, 418**

Mac **class**

cipher, combining with, 70

doFinal **method**, 71

getInstance **factory pattern**, 68, 71

HMAC, 68–70, 71

init **method**, 71

message digest, basing on, 68–71

symmetric cipher, basing on, 72–73

update **method**, 71

man-in-the-middle **attack**, **109**

mark **method**, **204, 207, 208**

match **method**, **222**

Menezes, Alfred J. (*Handbook of Applied Cryptography*),
124, 429

message **digest**

Bouncy Castle support, 418

byte array comparison, 65

cipher, combining with, 65

collision, 62

digital signature based on, 121

feeding data to, 65, 71

function, pseudorandom, 73–79

hash calculation, 62, 65

I/O, 79–81

Mac class, basing on, 68–71

mask generation, 77–79

SHA algorithms, 62–63, 65, 75, 79, 100

size required, determining, 65

tampering, 66–68

MessageDigest **class**

digest **method**, 65, 81

getDigestSize **method**, 65

getInstance **factory pattern**, 64

isEqual **method**, 65

update **method**, 65

MGF1 **function**, **77–79, 100, 101**

MGF1ParameterSpec **class**, **100**

MGFParameterSpec **method**, **100**

“Million Message Attack”, **97**

MimeBodyPart **class**

CMS-compressed message, returning from, 363

enveloping, 359, 361

S/MIME message, returning from, 348, 350–351,
357, 363

MimeMessage **class**, **349**

MimeMultipart **object**, **361**

module structure, **138–140**

MultipleCertificateExample **class**, **206–207**

MyStructure **class**, **157**

MyStructureTest **class**, **160–161**

N

name, distinguished. See DN

Name **structure**, **191**

NameConstraints **extension**, **264–265**

newInstance **method**, **295–296**

NIST (National Institute of Standards and Technology)
publications, **425–426**

nonce **extension**, **251**

nonce (number used once) **value**, **32**

NonceIvCBCExample **class**, **32–33**

NSA (National Security Agency), **3**

NULL **ASN.1 type**, **141, 145, 154**

NumericString **ASN.1 type**, **142, 145, 154**

O

OAEP (Optimal Asymmetric Encryption Padding),
96–99, 100

OAEPpaddedRSAExample **class**, **98–99**

OAEPParameterSpec **class**, **100–101**

Object **class**

clone **method**, 220, 245

equals **method**, 186, 219, 336

hashCode **method**, 186, 219, 336

toString **method**, 191

OBJECT IDENTIFIER **ASN.1 type**, **141, 145, 154,**
164, 186

object identifier (OID), **137, 141, 145, 174, 197**

OCSP (Online Certificate Status Protocol)

CertID structure implementation, 248–249

data structure information, returning, 249

date information, returning, 247

DER encoding, 256

extension, 248, 251–252, 258–264

hash information, returning, 249

- issuer information, returning, 249
 - JCA support, 248
 - request
 - extension, 251–252
 - generation, 253–254, 261
 - information about, returning, 250–251
 - response, 248, 252, 256–264
 - serial number, returning, 249
 - service locator, 252
 - signature information, returning, 249–250, 251
 - version, returning, 250, 257
 - OCSPReq class, 249–251, 263**
 - OCSPResp class, 255, 256**
 - OCSPResponderExample class, 259**
 - OCTET STRING ASN.1 type, 141, 145, 146, 147, 154**
 - OFB (Output Feedback) cipher mode, 37–38**
 - OID (object identifier), 137, 141, 145, 174, 197**
 - Online Certificate Status Protocol. See OCSP**
 - Optimal Asymmetric Encryption Padding (OAEP), 96–99, 100**
 - OSI (Open Systems Interconnection) standards, 135**
- ## P
- Packed Encoding Rules (PER), 150**
 - padding**
 - asymmetric key cryptography, 93–101
 - Initialize method, generating using, 94
 - ISO, 24
 - OAEP, 96–99, 100
 - P parameter, specifying in PSource class, 100
 - PKCS, 21–24, 94–96, 97
 - RSA algorithm, 93–101
 - symmetric key cryptography, 19–20, 21–24, 37, 55
 - TBC, 24
 - ZeroBytePadding, 24
 - PasswordProtection class, 290, 291**
 - PathChecker class, 271–272, 274–275**
 - PBE (password-based encryption)**
 - ASN.1 environment, 177–181
 - bandwidth, 45
 - CA, 209
 - decryption, 44
 - derived key, 74
 - dictionary attack, 45
 - EncryptedPrivateKeyInfo class, using, 177–181
 - hashing mechanism, 44
 - HMAC, 71
 - iteration count, 44, 45
 - JCE environment, in, 45–50
 - key generation, 43–44, 47, 56, 74
 - KeyStore class, 283, 286–288, 294, 296, 298–299
 - keytool command, 305–306, 307
 - parameters passed to, returning, 56
 - PBEWithSHA1AndDES algorithm, 75
 - PKCS, 43, 45, 74–76, 298
 - salt value, 44, 45
 - S/MIME, using with, 43
 - PBEKeySpec class, 46, 47, 48–49, 74**
 - PBEParameterSpec class, 45, 46–47, 48, 74**
 - PBEWithParamsExample class, 46–47**
 - PBEWithSHA1AndDES algorithm, 75**
 - PEM (Privacy Enhanced E-mail) encoding, 204, 205, 210, 220–221**
 - PEMWrite class, 204**
 - PER (Packed Encoding Rules), 150**
 - PFX structure, 298–299, 300**
 - PKCS (Public-Key Cryptography Standard)**
 - ASN.1 environment, 166–169
 - certification request, 208
 - KeyStore class PKCS12 type, 284, 285, 298–304, 309–312, 318
 - padding, 21–24, 94–96, 97
 - PBE, 43, 45, 74–76, 298
 - RSA standards, 426–427
 - signature, digital, 129–130, 167–169
 - PKCS12Attribute structure, 299**
 - PKCS10CertCreateExample class, 216–217**
 - PKCS10CertRequestExample class, 209–210**
 - PKCS8EncodedKeySpec class, 174–176**
 - PKCS10ExtensionExample class, 211–212**
 - PKCS1PaddedRSAExample class, 95–96**
 - PKCS5Scheme1 class, 74–75, 76**
 - PKCS5Scheme1Test class, 75–76**
 - PKCS1SigEncodingExample class, 167–168**
 - PKCS1SignatureExample class, 129–130**
 - PKCS12StoreExample class, 302–303**
 - PKIX validation algorithm, Bouncy Castle support, 417**
 - PKIXBuilderParameters class, 275, 277–278**
 - PKIXCertPathBuilderResult object, 331**
 - PKIXCertPathChecker class, 270, 271, 273**
 - PKIXCertPathValidatorResult class, 267, 278**
 - PKIXParameters class, 265, 266, 274**
 - Practical Cryptography (Ferguson and Schneier), 30, 31, 429**
 - PrecedenceTest class, 11**
 - PrintableString ASN.1 type, 142, 145, 154, 186**
 - Privacy Enhanced E-mail (PEM) encoding, 204, 205, 210, 220–221**

PrivateKey interface

PrivateKey interface, 85
PrivateKeyEntry class, 292
PrivateKeyInfo structure, 174
ProtectionParameter interface, 290–291
provider. See also Bouncy Castle provider
adding, 10
architecture, provider-based, 2
capability, listing, 12–13
installing, 7–10, 14
java.security file setup, 8–9, 10–11
JCE provider signing, 4
precedence, 10–12
property table, 13
returning, 10
Provider class, 10
PSource class, 100
PSpecified class, 100
PSS-based digital signature, 130–133, 169–171
PSSParameterSpec class, 131–132
Public-Key Cryptography Standard. See PKCS
PublicKey interface, 85, 88
putValue method, 391

R

“Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols” (Bellare and Rogaway), 97
RandomKeyElGamalExample class, 117
RandomKeyRSAExample class, 89–90
RDN (relative distinguished name), 184–185
RDNSequence ASN.1 type, 185
ReasonCode extension, 237
RecipientId class, 342
RecipientIdentifier object, 336
RecipientInformation class, 335, 336, 338, 342
RecipientInformationStore class, 337, 338, 344
relative distinguished name (RDN), 184–185
removeHandshakeCompletedListener method, 377, 378
replaceSigners method, 328
replaceUnsignedAttributes method, 327
reset method, 204, 207, 208
ResponseBytes structure, 256
ResponseData structure, 256–257, 263
revokedID object, 263
RFC (Request for Comment) publications, 427–428
Rogaway, Phillip (“Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols”), 97
RosettaNet Web site, 429

RSA (Rivest, Shamir, Adleman) algorithm

ARC4 stream cipher, 39–40
ASN.1 private key, 175
asymmetric key cryptography
abstraction layer, 88
interface, key, 85
padding, 93–101
translation, key, 88
calculation, 86
described, 85–86
digital signature, RSA-based, 122, 128–130
El Gamal algorithm compared, 118
exponent, 86, 89
key
exchange, secret, 103–105
private, 89, 101–103, 128
public, 88–89, 128
random, 89–91
size, 86
Utils class, configuring for RSA key pair
generation, 184
wrapping, 101–103
modulus, 86, 89
padding, 93–101
patent, 3
performance, improving, 91–92
PKCS standards, 426–427
RSAES-OAEP padding, 97
RSAKeyExchangeExample class, 103–105
RSAKeyGenParameterSpec class, 91, 94
RSAMultiPrimePrivateCrtKey interface, 93, 176
RSAMultiPrimePrivateCrtKeySpec class, 93
RSASOtherPrimeInfo class/structure, 93, 176
RSAPrivateCrtKey class, 92–93, 176
RSAPrivateCrtKeySpec class, 92–93
RSAPrivateKey interface/structure, 175, 176
RSAPrivateKeySpec class, 89
RSAPublicKey class, 88
RSAPublicKeySpec class, 88

S

SafeContents structure, 299
salt value, 44, 45
Schneier, Bruce
Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd ed, 24, 429
Crypto-Gram newsletter, 429
Practical Cryptography, 30, 31, 429
SECG (Standards for Efficient Cryptography Group), 428
SecretKey object, 5, 43

- SecretKeyEntry class, 292**
SecretKeyFactory class, 48, 174
SecretKeySpec class, 19, 42, 52
Secure Sockets Layer. See SSL
Secure/Multipurpose Internet Mail Extensions. See S/MIME
SecureRandom class, 31, 43
Security class, 10
Segmented Integer Counter (SIC) cipher mode, 35
SEQUENCE ASN.1 type, 143, 145, 152, 155, 156
server.jks file, 376
ServerKeyExchange message, 369
service provider interface (SPI) class, 2
SET ASN.1 type, 143–144, 145, 152–153, 155
setCertificate method, 222
setCertificateEntry method, 287
setCertStores method, 266
setDate method, 266
setDateAndTime method, 246
setDefaultHostnameVerifier method, 396
setDefaultSSLSocketFactory method, 396
setEnabledCipherSuites method
 SSLServerSocket class, 373
 SSLSocket class, 371
setEnabledProtocols method
 SSLServerSocket class, 373
 SSLSocket class, 372
setEnableSessionCreation method
 SSLServerSocket class, 373
 SSLSocket class, 372
setEntry method, 291, 292–294, 297
setIssuer method, 222
setIssuerDN method
 certGen1 class, 195
 crlGen class, 242
setKeyEntry method, 287
setMaxCRL method, 246
setMaxPathLength method, 275
setMinCRL method, 246
setNeedClientAuth method, 379
setNotAfter method, 195
setNotBefore method, 195
setParameter method, 122, 170
setPublicKey method, 195
setResponseExtensions method, 263
setRevocationEnabled method, 266
setSerialNumber method, 194, 223
setSignatureAlgorithm method, 195
setSubject method, 223
setSubjectDN method, 195
setTargetCertConstraints method, 266
setThisUpdate method, 242
setUseClientMode method
 SSLServerSocket class, 373
 SSLSocket class, 372
setWantClientAuth method, 379
SHA algorithms, 62–63, 65, 75, 79, 100
SIC (Segmented Integer Counter) cipher mode, 35
Signature class
 DSA calculation, using in, 125
 getInstance factory pattern, 121
 getParameters method, 122, 170
 initializing, 125
 initSign method, 121, 125
 initVerify method, 121, 122, 125
 setParameter method, 122, 170
 update method, 125
 Verify method, 122
signature, digital
 algorithm
 DSA-based, 122–128
 DSS-based, 122
 PSS-based, 130–133, 169–171
 RSA-based, 122, 128–130
 ASN.1 environment, 166–167, 169–170
 certificate
 OCSP, returning signature information in,
 249–250, 251
 self-signed, 193–195, 201–204
 verifying, 214, 233, 234, 235, 251
 clear signing technique, 348
 CMS, 320–321, 325–332
 creating, 121, 125
 detached, 322, 328–332
 encapsulation, 322
 external, 322
 introduced, 121
 JAR, 312, 313
 message digest-based, 121
 PKCS, 129–130, 167–169
 S/MIME signed message, 348–354, 357–361
 verifying, 121, 122, 125, 127–128
Signature structure, 250
SignedData structure, 321, 322, 325
SignedDataExample class, 330–331
SignedDataProcessor class, 329, 331
SignedMailExample class, 351, 357
SignedProcessor class, 358

signerConstraints object

- signerConstraints **object**, **331**
- SignerInfo **structure**, **323–324**, **325**, **326**
- SignerInformation **class**, **325**, **326–327**, **331**
- SignerInformationStore **class**, **327**
- SimpleCBCEXample **class**, **27–28**
- SimpleCTREXample **class**, **36–37**
- SimpleECBExample **class**, **25–26**
- SimpleIOExample **class**, **53–54**
- SimplePolicyTest **class**, **5–6**
- SimpleProviderTest **class**, **9**
- SimpleSymmetricExample **class**, **17–18**
- SimpleWrapExample **class**, **50–51**
- SingleResp **class**
 - getCertID method, 257
 - getCertStatus method, 257, 258, 264
 - getNextUpdate method, 258
 - getThisUpdate method, 257, 258
- size **method**
 - KeyStore class, 287
 - RecipientInformationStore class, 337
 - SignerInformationStore class, 327
- slash, asterisk (/*) ASN.1 comment prefix**, **137**
- S/MIME (Secure/Multipurpose Internet Mail Extensions)**
 - Bouncy Castle implementation, 347, 349
 - CMS classes, 349
 - compression, 361–363
 - creating S/MIME-signed message, 350–354
 - CRLF, 349, 350
 - envelope, 348, 354–361
 - MimeBodyPart object, returning from S/MIME message, 348, 350–351, 357, 363
 - PBE, using with, 43
 - signing message, 348–354, 357–361
 - specification, 347
 - Utils class, configuring for, 316–318
 - validating S/MIME-signed message, 350–354, 361
- SMIMECompressed **class**, **361**, **362**
- SMIMEEncryptionKeyPreference **structure**, **353**
- SMIMEEnveloped **class**, **355**
- SMIMEEnvelopedGenerator **object**, **357**, **361**
- SMIMESigned **class**
 - CMSProcessableBodyPartInbound class, using with, 349
 - CMSProcessableBodyPartOutbound class, using with, 349
 - constructor, 350
 - creating during message validation, 354
 - getContent method, 350
 - getContentAsMimeMessage method, 350
 - getContentWithSignature method, 350
- SMIMESignedGenerator **class**, **349**
- SMIMEUtil **class**, **348**, **363**
- specification object**, **109**, **126**
- SPI (service provider interface) class**, **2**
- SSL (Secure Sockets Layer)**
 - cipher suite, 366, 370, 371, 373
 - client
 - certificate chain, 369, 378
 - KeyManagerFactory class, configuring for client-side authentication, 381–382
 - link client end, 374
 - message flow during client-side authentication, 378
 - mode, setting, 372, 373
 - server, interaction with, 369
 - socket factory, default, 377
 - SSLContext class, configuring for client-side authentication, 379–381
 - SSLServerSocket class, configuring for client-side authentication, 378–379
 - SSLSocket class, configuring for client-side authentication, 379
 - TrustManagerFactory class, configuring for client-side authentication, 386–387, 389
 - firewall, 376
 - handshake, 366, 372, 377–378
 - history, 365
 - HTTPS connection, 394–401
 - MAC algorithm, 366
 - server, 369, 374–376, 383–385
 - ServerKeyExchange message, 369
 - session, 372, 389–394
 - socket, creating, 370
 - specification, 366
 - trust store, 377
 - Utils class, configuring for, 366–368
- SSLClientExample **class**, **374–375**
- SSLClientWithClientAuthExample **class**, **383**
- SSLClientWithClientAuthTrustExample **class**, **387–388**
- SSLContext **class**
 - authentication, configuring for client-side, 379–381
 - getClientSessionContext method, 380
 - getInstance factory pattern, 380
 - getProtocol method, 380
 - getServerSessionContext method, 381
 - getServerSocketFactory method, 381
 - getSocketFactory method, 381
 - init method, 380, 385
- SSLServerSocket **class**, **373**, **378–379**
- SSLServerSocketFactory **class**, **370–371**

SSLServerWithClientAuthIdExample class, 392–393
SSLSession interface, 389–394
SSLSocket class
 addHandshakeCompletedListener method, 377, 378
 authentication, configuring for client-side, 379
 removeHandshakeCompletedListener method, 377, 378
 setEnabledCipherSuites method, 371
 setEnabledProtocols method, 372
 setEnableSessionCreation method, 372
 setNeedClientAuth method, 379
 setUseClientMode method, 372
 setWantClientAuth method, 379
 startHandshake method, 372
SSLSocketFactory class, 370, 401
Standards for Efficient Cryptography Group (SECG), 428
 startHandshake **method, 372**
 store **method, 287–288, 297**
streaming
 ARC4 stream cipher, 39–40
 certificate, 204, 205–206, 207, 244–245
 CMSSignedData class, 327
 symmetric stream cipher, 35–40, 419
 tampering with an encrypted stream, 61–62, 66–68
String class, 60
 SubjectAltName **extension, 199–200, 201, 213, 218**
 SubjectKeyIdentifier **extension, 214, 215, 301**
 SubjectKeyIdentifierStructure **object, 218**
 SubjectPublicKeyInfo **structure, 171, 172, 174**
Sun X.509 certificate factory, 311
symmetric cipher
 block mode
 AlgorithmParameters object creation in, 34
 Bouncy Castle support, 418–419
 CBC, 26–28, 34–35, 56
 CFB, 38
 CTR, 35–37, 40–42
 CTS, 34–35
 ECB, 25–26, 27–28, 35
 IV, inline, 28–34, 56
 OFB, 37–38
 padding, 21–24
 SecureRandom object creation in, 31
 streaming, 35–38
 Mac class, basing on, 72–73
 padding, 19–20, 21–24, 37, 55
 stream, 35–38, 39–40
system policy file, 312–313

T

tag-length-value (TLV) convention, 150
 TamperedDigestExample **class, 66–67**
 TamperedExample **class, 61**
 TamperedWithDigestExample **class, 63–64**
 TamperedWithHMacExample **class, 68–69**
TBC (Trailing Bit Complement) padding, 24
 tbsCertificate
 structure, 190, 191, 193, 209, 223
 X509Certificate class field, 189
 TeletexString **ASN.1 type, 143, 155**
three-party Diffie-Hellman algorithm, 115–116
TLS (Transport Layer Security), 366–368. See also SSL (Secure Sockets Layer)
TLV (tag-length-value) convention, 150
 toByteArray **method, 60**
 toHex **method, 16–18, 24**
 toMimeBodyPart **method, 348, 363**
 toString **method**
 Object class, 191
 Utils class, 60
Trailing Bit Complement (TBC) padding, 24
 translateKey **method, 88**
Transport Layer Security (TLS), 366–368. See also SSL (Secure Sockets Layer)
trust anchor, 264–265, 267, 278, 306
 TrustedCertificateEntry **class, 292**
 TrustManagerFactory **class, 386–387, 389, 392**
 trustStore.jks **file, 376**

U

UniversalString **ASN.1 type, 143, 145, 155**
 unwrap **method, 50, 52, 102–103**
 update **method**
 Cipher class, 19, 20, 23
 Mac class, 71
 MessageDigest class, 65
 Signature class, 125
 UTCTime **ASN.1 type, 141, 145, 155, 191**
 UTF8String **ASN.1 type, 143, 145, 155, 186**
Utils class
 ASN.1, configuring for, 136
 buildPath method, 318, 331
 certificate handling, configuring for, 228–229
 CMS, configuring for, 316–318
 createCredentials method, 318
 createMimeMessage method, 318
 extending, 84–85
 generateIntermediateCert method, 228

Utils class (continued)

Utils class (continued)

generateRootCert method, 228
KeyStore object, configuring for, 281–283
RSA key pair generation, configuring for, 184
SecureRandom class, configuring for, 84
S/MIME, configuring for, 316–318
SSL, configuring for, 366–368
TLS, configuring for, 366–368
toByteArray method, 60
toHex method, 16–18, 24
toString method, 60

V

validate method, 266

Van Oorschot, Paul C. (*Handbook of Applied Cryptography*), 124, 429

Vanstone, Scott A. (*Handbook of Applied Cryptography*), 124, 429

verify method

Certificate class, 188
Signature class, 122
SignerInformation class, 326–327
X509CRL class, 235

VideotexString **ASN.1 type, 143**

VisibleString **ASN.1 type, 143, 145, 155**

W

weak key, 19

web of trust certificate approach, 188

wrap method, 50, 102, 179

wrapping, key

Cipher.wrap method, using, 50, 102, 179
purpose-built key-wrapping mechanism, 52
RSA key, 101–103
symmetric, 50–52
unwrapping, 48

write method, 320, 348

X

XER (XML Encoding Rules), 150

X500Principal class, 186–187, 191, 247

X500PrivateCredential class, 282

X509Certificate class

ASN.1, based on, 189

checkValidity method, 192

extension support, 198–201

generateEndEntityCert method, 230

getBasicConstraints method, 200

getExtendedKeyUsage method, 200–201

getIssuerAlternativeNames method, 200

getIssuerDN method, 191

getIssuerUniqueID method, 192

getIssuerX500Principal method, 191

getKeyUsage method, 198–199

getNotAfter method, 191

getNotBefore method, 191

getSerialNumber method, 190–191

getSigAlgOID method, 193

getSigAlgParams method, 193

getSignature method, 193

getSubjectAlternativeNames method, 199–200

getSubjectUniqueID method, 192

getSubjectX500Principal method, 192

getTBSertificate method, 189–190

getVersion method, 190

signatureAlgorithm field, 189

signatureValue field, 189

tbsCertificate field, 189

X509CertSelector class, 222, 223–224

X509CRL class, 233–236

X509CRLEntry class, 236

X509CRLExample class, 241–242

X509CRLSelector class, 245–246

X509EncodedKeySpec class, 171–174

X509EncodedKeySpecExample class, 172–173

X509Extension interface

getCriticalExtensionsOIDs method, 197

getExtensionValue method, 197, 200

getNonCriticalExtensionOIDs method, 197

hasUnsupportedCriticalExtension
method, 197

X509V1CertificateGenerator class, 193, 194

X509V3CertificateGenerator class, 203

X509V1CreateExample class, 193–194

X509V3CreateExample class, 202–203

XML Encoding Rules (XER), 150

Z

ZeroBytePadding mechanism, 24

ZLIB compression algorithm, 345, 347