





Catálogo de Publicaciones de la Administración General del Estado  
<https://cpage.mpr.gob.es>

cpage.mpr.gob.es

Edita:



Pº de la Castellana 109, 28046 Madrid  
© Centro Criptológico Nacional, 2023

NIPO: 083-23-083-8.

Fecha de Edición: marzo de 2023

#### LIMITACIÓN DE RESPONSABILIDAD

El presente documento se proporciona de acuerdo con los términos en él recogidos, rechazando expresamente cualquier tipo de garantía implícita que se pueda encontrar relacionada. En ningún caso, el Centro Criptológico Nacional puede ser considerado responsable del daño directo, indirecto, fortuito o extraordinario derivado de la utilización de la información y software que se indican incluso cuando se advierta de tal posibilidad.

#### AVISO LEGAL

Quedan rigurosamente prohibidas, sin la autorización escrita del Centro Criptológico Nacional, bajo las sanciones establecidas en las leyes, la reproducción parcial o total de este documento por cualquier medio o procedimiento, comprendidos la reprografía y el tratamiento informático, y la distribución de ejemplares del mismo mediante alquiler o préstamo públicos.

## **PRÓLOGO**

En un mundo cada vez más complejo y globalizado, en el que las tecnologías de la información y la comunicación (TIC) desempeñan un papel de suma importancia, hemos de ser conscientes de que la gestión adecuada de la ciberseguridad constituye un reto colectivo al que necesariamente hemos de enfrentar. Resulta necesario garantizar la protección de la capacidad económica, tecnológica y política de nuestro país, máxime cuando la proliferación de ataques dirigidos y el robo de información sensible representan una realidad incontestable.

Por ello, resulta imprescindible estar al día de las amenazas y vulnerabilidades asociadas al uso de las nuevas tecnologías. El conocimiento de los riesgos que se ciernen sobre el ciberespacio ha de servir para implementar con garantías las medidas, tanto procedimentales como técnicas y organizativas, que permitan un entorno seguro y confiable.

La Ley 11/2002, de 6 de mayo, reguladora del Centro Nacional de Inteligencia (CNI), encomienda al Centro Nacional de Inteligencia el ejercicio de las funciones relativas a la seguridad de las tecnologías de la información y de protección de la información clasificada, a la vez que confiere a su Secretario de Estado Director la responsabilidad de dirigir el Centro Criptológico Nacional (CCN)

Partiendo del conocimiento y la experiencia del CNI sobre amenazas y vulnerabilidades en materia de riesgos emergentes, el Centro realiza, a través del Centro Criptológico Nacional, regulado por el Real Decreto 421/2004, de 12 de marzo, diversas actividades directamente relacionadas con la seguridad de las TIC, orientadas a la formación de personal experto, al empleo de tecnologías de seguridad adecuadas y a la aplicación de políticas y procedimientos de seguridad.

Precisamente, esta serie de documentos CCN-STIC es un claro reflejo de la labor que este organismo lleva a cabo en materia de implementación de seguridad, permitiendo la aplicación de políticas y procedimientos, pues las guías han sido elaboradas con un claro objetivo: mejorar el grado de ciberseguridad de las organizaciones, conscientes de la importancia que tiene el establecimiento de un marco de referencia en esta materia que sirva de apoyo para que el personal de la Administración lleve a cabo la difícil tarea de proporcionar seguridad a los sistemas de las TIC bajo su responsabilidad.

Con esta serie de documentos, el Centro Criptológico Nacional, en cumplimiento de sus cometidos y de lo reflejado en el Real Decreto 3/2010 por el que se regula el Esquema Nacional de Seguridad en el ámbito de la Administración electrónica, contribuye a mejorar la ciberseguridad española y mantener las infraestructuras y los sistemas de información de todas las administraciones públicas con unos niveles óptimos de seguridad. Todo ello, con el fin de generar confianza y garantías en el uso de estas tecnologías, protegiendo la confidencialidad de los datos y garantizando su autenticidad, integridad y disponibilidad.

Marzo de 2023

Esperanza Casteleiro Llamazares  
Secretaria de Estado  
Directora del Centro Criptológico Nacional

## ÍNDICE

<b>1</b>	<b>INTRODUCCIÓN</b>	12
1.1	OBJETIVO DE LA GUÍA	12
1.2	MECANISMOS CRIPTOGRÁFICOS	16
1.2.1	ALGORITMOS, PROTOCOLOS Y ESQUEMAS	16
1.2.2	TIPOS DE ATAQUE	18
1.2.3	ESQUEMAS SIMÉTRICOS Y ASIMÉTRICOS	19
1.3	COMPLEJIDAD DE UN ATAQUE Y NIVELES DE SEGURIDAD	20
1.3.1	COMPLEJIDAD DE UN ATAQUE	20
1.3.2	NIVELES DE SEGURIDAD	21
1.4	ORGANIZACIÓN DE LA GUÍA	24
<b>2</b>	<b>MECANISMOS SIMÉTRICOS</b>	25
2.1	PRIMITIVAS SIMÉTRICAS	25
2.1.1	CIFRADORES EN FLUJO	25
2.1.2	CIFRADORES EN BLOQUE	26
2.1.3	FUNCIONES RESUMEN	26
2.1.4	COMPARTICIÓN DE SECRETOS	28
2.2	CONSTRUCCIONES SIMÉTRICAS	28
2.2.1	MODOS DE OPERACIÓN EN EL CIFRADO Y DESCIFRADO	28
2.2.2	CIFRADO DE DISCO DURO	34
2.2.3	CÓDIGOS DE AUTENTICACIÓN DE MENSAJES	36
2.2.4	ESQUEMAS SIMÉTRICOS DE AUTENTICACIÓN DE ENTIDADES	39
2.2.5	CIFRADO AUTENTICADO	40
2.2.6	PROTECCIÓN DE LAS CLAVES	42
2.2.7	FUNCIONES DE DERIVACIÓN DE CLAVES	43
2.2.8	MECANISMOS DE PROTECCIÓN/RESUMEN DE CONTRASEÑAS	44
<b>3</b>	<b>MECANISMOS ASIMÉTRICOS</b>	46
3.1	PRIMITIVAS ASIMÉTRICAS	46
3.1.1	PROBLEMA DE LA FACTORIZACIÓN DE NÚMEROS ENTEROS (RSA)	47
3.1.2	PROBLEMA DEL LOGARITMO DISCRETO MULTIPLICATIVO	48
3.1.3	PROBLEMA DEL LOGARITMO DISCRETO ADITIVO	49
3.1.4	OTROS PROBLEMAS COMPUTACIONALMENTE DIFÍCILES	52
3.2	CONSTRUCCIONES ASIMÉTRICAS	52
3.2.1	ESQUEMAS DE CIFRADO ASIMÉTRICO	53
3.2.2	ESQUEMAS DE CIFRADO HÍBRIDO	55
3.2.3	FIRMAS DIGITALES	59
3.2.3.1	FIRMAS DIGITALES NO RESISTENTES A LA COMPUTACIÓN CUÁNTICA	59
3.2.3.2	FIRMAS DIGITALES POSTCUÁNTICAS	67
3.2.3.3	FIRMAS DIGITALES PARA FIRMWARE	68
3.2.4	ESQUEMAS DE AUTENTICACIÓN DE ENTIDAD ASIMÉTRICA	73
3.2.5	ESTABLECIMIENTO DE CLAVES	73
<b>4</b>	<b>PROTOCOLOS CRIPTOGRÁFICOS</b>	81
4.1	TLS	81

4.1.1	TLS VERSION 1.3	86
4.1.2	TLS VERSION 1.2	88
4.2	SSH	92
4.2.1	ACUERDO DE CLAVE	93
4.2.2	CIFRADO	94
4.2.3	INTEGRIDAD Y AUTENTICIDAD EN ORIGEN	95
4.2.4	AUTENTICACIÓN DEL SERVIDOR Y DEL CLIENTE	96
4.3	IPSEC CON IKEV2	96
4.3.1	ACUERDO DE CLAVES	97
4.3.2	CIFRADO	98
4.3.3	INTEGRIDAD Y AUTENTICACIÓN	98
4.3.4	FUNCIONES PSEUDO-ALEATORIAS	99
4.3.5	MECANISMOS DE AUTENTICACIÓN	99
<b>5</b>	<b>GENERADORES DE NÚMEROS ALEATORIOS</b>	<b>101</b>
5.1	GENERADORES DE NÚMEROS ALEATORIOS	101
5.2	GENERADORES FÍSICOS DE NÚMEROS ALEATORIOS	102
5.3	GENERADORES DE NÚMEROS ALEATORIOS DETERMINISTAS	104
5.4	GENERADORES DE NÚMEROS REALMENTE ALEATORIOS NO FÍSICOS	106
5.5	GENERACIÓN DE NÚMEROS ALEATORIOS CON UNA DISTRIBUCIÓN ESPECÍFICA	108
<b>6</b>	<b>GESTIÓN DE CLAVES</b>	<b>110</b>
6.1	GESTIÓN DE CLAVES	110
6.2	GENERACIÓN DE CLAVES	110
6.3	ALMACENAMIENTO Y TRANSPORTE DE CLAVES	112
6.4	USO DE CLAVES	112
6.5	DESTRUCCIÓN DE CLAVES	113
<b>7</b>	<b>AUTENTICACIÓN DE PERSONAS</b>	<b>114</b>
7.1	AUTENTICACIÓN	114
7.2	PROCEDIMIENTOS DE AUTENTICACIÓN	114
7.2.1	LIMITACIÓN EN EL NÚMERO DE ENSAYOS	115
7.2.2	LIMITACIÓN TEMPORAL EN EL NÚMERO DE ENSAYOS	115
<b>8</b>	<b>GENERACIÓN DE PRIMOS Y CLAVES RSA</b>	<b>117</b>
8.1	GENERACIÓN DE NÚMEROS PRIMOS	117
8.2	TEST DE PRIMALIDAD Y DE PSEUDO-PRIMALIDAD	119
8.2.1	GENERACIÓN DE PRIMOS PROBABLES ( <i>PROBABLE PRIMES</i> )	120
8.3	GENERACIÓN DEL PAR DE CLAVES RSA	125
8.4	ATAQUE ROCA AL RSA	126
<b>9</b>	<b>CRIPTOGRAFÍA POSTCUÁNTICA</b>	<b>128</b>
9.1	LA AMENAZA CUÁNTICA	128
9.1.1	CONVOCATORIA DEL NIST	129
9.1.2	SEGURIDAD	131
9.2	CRIPTOGRAFÍA BASADA EN RETÍCULOS	133
9.2.1	RETÍCULOS	133
9.2.2	CRYSTALS-KYBER	136

9.2.3	FRODOKEM . . . . .	139
9.2.4	CRYSTALS-DILITHIUM . . . . .	140
9.2.5	FALCON . . . . .	143
9.3	FIRMAS DIGITALES BASADAS EN FUNCIONES RESUMEN ( <i>HASH</i> ) . . . . .	145
9.3.1	SPHINCS <sup>+</sup> . . . . .	145
<b>10</b>	<b>TABLAS</b> . . . . .	<b>149</b>
10.1	PRIMITIVAS SIMÉTRICAS . . . . .	149
10.2	CONSTRUCCIONES SIMÉTRICAS . . . . .	149
10.3	PRIMITIVAS ASIMÉTRICAS . . . . .	151
10.4	CONSTRUCCIONES ASIMÉTRICAS . . . . .	152
10.5	TLS . . . . .	153
10.6	SSH . . . . .	157
10.7	IPSEC CON IKEV2 . . . . .	158
10.8	GENERADORES FÍSICOS DE NÚMEROS ALEATORIOS . . . . .	159
10.9	GENERADORES DE NÚMEROS ALEATORIOS DETERMINISTAS . . . . .	159
10.10	GENERADORES DE NÚMEROS REALMENTE ALEATORIOS NO FÍSICOS . . . . .	160
10.11	GENERACIÓN DE NÚMEROS ALEATORIOS CON UNA DISTRIBUCIÓN ESPECÍFICA . . . . .	160
10.12	GENERACIÓN DE CLAVES . . . . .	160
10.13	PROCEDIMIENTOS DE AUTENTICACIÓN . . . . .	161
10.14	GENERACIÓN DE NÚMEROS PRIMOS . . . . .	161
10.15	GENERACIÓN DEL PAR DE CLAVES RSA . . . . .	161
10.16	LA AMENAZA CUÁNTICA . . . . .	162
<b>11</b>	<b>GLOSARIO</b> . . . . .	<b>163</b>
<b>12</b>	<b>REFERENCIAS</b> . . . . .	<b>169</b>

## FIGURAS

Figura 1.	Modo CTR para cifrado y descifrado . . . . .	30
Figura 2.	Modo OFB para cifrado y descifrado . . . . .	31
Figura 3.	Modo CBC para cifrado y descifrado . . . . .	32
Figura 4.	Modo CFB para cifrado y descifrado . . . . .	33
Figura 5.	Modo XTS-AES para cifrado y descifrado . . . . .	35
Figura 6.	Esquema de cifrado mediante el criptosistema híbrido ECIES . . . . .	58
Figura 7.	Esquema de descifrado mediante el criptosistema híbrido ECIES . . . . .	59

## TABLAS

Tabla 2.1.	Tipos autorizados de cifradores en bloque . . . . .	26
Tabla 2.2.	Funciones resumen autorizadas . . . . .	27
Tabla 2.3.	Compartición de secretos autorizada . . . . .	28
Tabla 2.4.	Modos autorizados de cifrado simétrico . . . . .	32
Tabla 2.5.	Modos autorizados de cifrado simétrico para discos duros . . . . .	35
Tabla 2.6.	MAC autorizados basados en cifradores en bloque y funciones resumen . . . . .	38
Tabla 2.7.	Tamaño de los protocolos de desafío-respuesta autorizados . . . . .	40
Tabla 2.8.	Esquemas simétricos de cifrado autenticado autorizados . . . . .	41
Tabla 2.9.	Esquemas de protección de claves autorizados . . . . .	43
Tabla 2.10.	Funciones de derivación de claves autorizadas . . . . .	44
Tabla 2.11.	Mecanismos de resumen de contraseñas autorizados . . . . .	45
Tabla 3.1.	Tamaño de las Primitivas RSA autorizadas . . . . .	48
Tabla 3.2.	Tamaño de las primitivas autorizadas del logaritmo discreto multiplicativo sobre un cuerpo finito . . . . .	49
Tabla 3.3.	Curvas elípticas autorizadas . . . . .	51
Tabla 3.4.	Esquema de cifrado asimétrico autorizado . . . . .	54
Tabla 3.5.	Esquemas de firma digital autorizados . . . . .	71
Tabla 3.6.	Esquemas de establecimiento de claves autorizados . . . . .	79
Tabla 4.1.	Versiones del protocolo TLS autorizadas . . . . .	83
Tabla 4.2.	Suites criptográficas autorizadas para el protocolo TLS 1.3 . . . . .	86
Tabla 4.3.	Modos de clave precompartida recomendados para el protocolo TLS 1.3 . . . . .	86
Tabla 4.4.	Grupos de Diffie-Hellman recomendados para el protocolo TLS 1.3 . . . . .	87
Tabla 4.5.	Algoritmos de firma (cliente/servidor) recomendados para el protocolo TLS 1.3 . . . . .	88
Tabla 4.6.	Algoritmos de firma (en certificados) recomendados para el protocolo TLS 1.3 . . . . .	88
Tabla 4.7.	Suites criptográficas recomendadas para TLS 1.2 con un servidor que disponga de un certificado con la clave pública EC-DSA . . . . .	89
Tabla 4.8.	Suites criptográficas heredadas para TLS 1.2 con un servidor que disponga de un certificado con la clave pública RSA . . . . .	89
Tabla 4.9.	Suites criptográficas recomendadas para TLS 1.2 cuando no hay soporte ECC o modo de cifrado autenticado . . . . .	90
Tabla 4.10.	Suites criptográficas recomendadas para TLS 1.2 con clave precompartida . . . . .	91
Tabla 4.11.	Grupos de Diffie-Hellman recomendados para el protocolo TLS 1.2 . . . . .	92
Tabla 4.12.	Algoritmos de firma recomendados para el protocolo TLS 1.2 . . . . .	92
Tabla 4.13.	Funciones resumen para el protocolo TLS 1.2 . . . . .	92
Tabla 4.14.	Versiones de acuerdos de clave para el protocolo SSH autorizadas . . . . .	93
Tabla 4.15.	Algoritmos de cifrado autorizados para el protocolo . . . . .	95
Tabla 4.16.	Funciones MAC autorizadas para el protocolo SSH . . . . .	95
Tabla 4.17.	Métodos de autenticación del servidor autorizados para el protocolo SSH . . . . .	96

Tabla 4.18.	Versiones autorizadas del protocolo IPsec . . . . .	97
Tabla 4.19.	Grupos de tipo DH y ECDH acordados por IKEv2 . . . . .	98
Tabla 4.20.	Esquemas MAC y HMAC autorizados para IKEv2 y ESP . . . . .	99
Tabla 4.21.	Funciones pseudo-aleatorias autorizadas para IKEv2 . . . . .	99
Tabla 4.22.	Esquemas de firma acordados para IKEv2 . . . . .	100
Tabla 5.1.	Clases de generadores de números aleatorios autorizados . . . . .	103
Tabla 5.2.	Generadores de números aleatorios deterministas autorizados . . . . .	105
Tabla 5.3.	Clases de generadores de números aleatorios deterministas autorizados	105
Tabla 5.4.	Clase de generador de números aleatorios ni físico ni determinista autorizado . . . . .	107
Tabla 5.5.	Esquemas autorizados de generación de números enteros aleatorios módulo un número $q$ , que no es una potencia de 2 . . . . .	109
Tabla 6.1.	Métodos autorizados para la generación de claves genéricas . . . . .	111
Tabla 7.1.	Probabilidades máximas de falsa aceptación . . . . .	115
Tabla 8.1.	Generación de primos por muestreo de rechazo . . . . .	119
Tabla 8.2.	Número de iteraciones del test de Miller-Rabin según la longitud en bits del candidato para $P_{obj} = 2^{-125}$ . . . . .	123
Tabla 8.3.	Test de primalidad probabilístico autorizado . . . . .	123
Tabla 8.4.	Algoritmo autorizado para la generación claves RSA . . . . .	125
Tabla 9.1.	Propuesta KEM seleccionada después de la tercera ronda y Primitiva matemática asociada . . . . .	130
Tabla 9.2.	Propuestas a firmas seleccionadas después de la tercera ronda y Primitivas matemáticas asociadas . . . . .	130
Tabla 9.3.	Candidatos a KEM para ser analizados en la cuarta ronda y Primitivas matemáticas asociadas . . . . .	130
Tabla 9.4.	Propuesta KEM de interés para el CCN . . . . .	131
Tabla 10.1.	Tipos autorizados de cifradores en bloque . . . . .	149
Tabla 10.2.	Funciones resumen autorizadas . . . . .	149
Tabla 10.3.	Compartición de secretos autorizada . . . . .	149
Tabla 10.4.	Modos autorizados de cifrado simétrico . . . . .	149
Tabla 10.5.	Modos autorizados de cifrado simétrico para discos duros . . . . .	150
Tabla 10.6.	MAC autorizados basados en cifradores en bloque y funciones resumen	150
Tabla 10.7.	Tamaño de los protocolos de desafío-respuesta autorizados . . . . .	150
Tabla 10.8.	Esquemas simétricos de cifrado autenticado autorizados . . . . .	150
Tabla 10.9.	Esquemas de protección de claves autorizados . . . . .	151
Tabla 10.10.	Funciones de derivación de claves autorizadas . . . . .	151
Tabla 10.11.	Mecanismos de resumen de contraseñas autorizados . . . . .	151
Tabla 10.12.	Tamaño de las Primitivas RSA autorizadas . . . . .	151
Tabla 10.13.	Tamaño de las primitivas autorizadas del logaritmo discreto multiplicativo sobre un cuerpo finito . . . . .	151
Tabla 10.14.	Curvas elípticas autorizadas . . . . .	152
Tabla 10.15.	Esquema de cifrado asimétrico autorizado . . . . .	152
Tabla 10.16.	Esquemas de firma digital autorizados . . . . .	152
Tabla 10.17.	Esquemas de establecimiento de claves autorizados . . . . .	153

Tabla 10.18.	Versiones del protocolo TLS autorizadas . . . . .	153
Tabla 10.19.	Suites criptográficas autorizadas para el protocolo TLS 1.3 . . . . .	153
Tabla 10.20.	Modos de clave precompartida recomendados para el protocolo TLS 1.3 . . . . .	153
Tabla 10.21.	Grupos de Diffie-Hellman recomendados para el protocolo TLS 1.3 . . . . .	154
Tabla 10.22.	Algoritmos de firma (cliente/servidor) recomendados para el protocolo TLS 1.3 . . . . .	154
Tabla 10.23.	Algoritmos de firma (en certificados) recomendados para el protocolo TLS 1.3 . . . . .	154
Tabla 10.24.	Suites criptográficas recomendadas para TLS 1.2 con un servidor que disponga de un certificado con la clave pública EC-DSA . . . . .	155
Tabla 10.25.	Suites criptográficas heredadas para TLS 1.2 con un servidor que disponga de un certificado con la clave pública RSA . . . . .	155
Tabla 10.26.	Suites criptográficas recomendadas para TLS 1.2 cuando no hay soporte ECC o modo de cifrado autenticado . . . . .	155
Tabla 10.27.	Suites criptográficas recomendadas para TLS 1.2 con clave precompartida . . . . .	156
Tabla 10.28.	Grupos de Diffie-Hellman recomendados para el protocolo TLS 1.2 . . . . .	156
Tabla 10.29.	Algoritmos de firma recomendados para el protocolo TLS 1.2 . . . . .	156
Tabla 10.30.	Funciones resumen para el protocolo TLS 1.2 . . . . .	156
Tabla 10.31.	Versiones de acuerdos de clave para el protocolo SSH autorizadas . . . . .	157
Tabla 10.32.	Algoritmos de cifrado autorizados para el protocolo SSH . . . . .	157
Tabla 10.33.	Funciones MAC autorizadas para el protocolo SSH . . . . .	157
Tabla 10.34.	Métodos de autenticación del servidor autorizados para el protocolo SSH . . . . .	157
Tabla 10.35.	Versiones autorizadas del protocolo IPsec . . . . .	158
Tabla 10.36.	Grupos de tipo DH y ECDH acordados por IKEv2 . . . . .	158
Tabla 10.37.	Esquemas MAC y HMAC autorizados para IKEv2 y ESP . . . . .	158
Tabla 10.38.	Funciones pseudo-aleatorias autorizadas para IKEv2 . . . . .	159
Tabla 10.39.	Esquemas de firma acordados para IKEv2 . . . . .	159
Tabla 10.40.	Clases de generadores de números aleatorios autorizados . . . . .	159
Tabla 10.41.	Generadores de números aleatorios deterministas autorizados . . . . .	159
Tabla 10.42.	Clases de generadores de números aleatorios deterministas autorizados . . . . .	160
Tabla 10.43.	Clase de generador de números aleatorios ni físico ni determinista autorizado . . . . .	160
Tabla 10.44.	Esquemas autorizados de generación de números enteros aleatorios módulo un número $q$ , que no es una potencia de 2 . . . . .	160
Tabla 10.45.	Métodos autorizados para la generación de claves genéricas . . . . .	160
Tabla 10.46.	Probabilidades máximas de falsa aceptación . . . . .	161
Tabla 10.47.	Generación de primos por muestreo de rechazo . . . . .	161
Tabla 10.48.	Número de iteraciones del test de Miller-Rabin según la longitud en bits del candidato para $P_{obj} = 2^{-125}$ . . . . .	161
Tabla 10.49.	Test de primalidad probabilístico autorizado . . . . .	161
Tabla 10.50.	Algoritmo autorizado para la generación claves RSA . . . . .	161

Tabla 10.51.	Propuesta KEM seleccionada después de la tercera ronda y Primitiva matemática asociada . . . . .	162
Tabla 10.52.	Propuestas a firmas seleccionadas después de la tercera ronda y Primitivas matemáticas asociadas . . . . .	162
Tabla 10.53.	Candidatos a KEM para ser analizados en la cuarta ronda y Primitivas matemáticas asociadas . . . . .	162
Tabla 10.54.	Propuesta KEM de interés para el CCN . . . . .	162

## 1. INTRODUCCIÓN

1. En esta sección se presenta una introducción a esta Guía de uso de mecanismo criptográficos a nivel nacional. De este modo, se presenta su objetivo, los mecanismos criptográficos que van a ser considerados para lo que se incluyen las definiciones de los términos más importantes, como algoritmos, protocolos, esquemas, primitivas, etc.; los diferentes tipos de ataque y la distinción entre los esquemas simétricos y asimétricos. Sigue una revisión general sobre los diferentes tipos de complejidad que pueden considerarse cuando se ataca un mecanismo y unos comentarios sobre los niveles de seguridad. Finalmente, se incluye una descripción de cómo está organizada esta Guía y el contenido de los diferentes capítulos que la conforman.

### 1.1. OBJETIVO DE LA GUÍA

2. El principal objetivo de este documento es especificar los mecanismos y dispositivos criptográficos que están reconocidos y aceptados por el Centro Criptológico Nacional (CCN), del Centro Nacional de Inteligencia (CNI). Se trata de determinar de forma clara qué productos son recomendados por el CCN para ser utilizados tanto por desarrolladores como por evaluadores pertenecientes a empresas y organismos nacionales que pretendan implementar sistemas y algoritmos de cifra para su labor diaria.
3. En el caso de que alguna empresa u organismo deseara utilizar un mecanismo o dispositivo no incluido en esta Guía, podrá elevar su consulta al CCN, quien, en última instancia, será quien tome la decisión definitiva con relación al producto en cuestión.
4. También este documento puede ayudar a todos los usuarios que deseen emplear productos criptográficos para fines personales de modo que puedan cubrir sus necesidades en este entorno, obteniendo la confianza de que tales productos son seguros y proporcionan protección en los diferentes aspectos en los que la criptografía tiene su ámbito de actuación: confidencialidad, integridad, autenticación y no repudio de los datos e información.
5. Otros países de nuestro entorno también han publicado documentos o informes similares, con recomendaciones que tienen objetivos similares a los de esta Guía (véanse por ejemplo, [16], [14], [197], [17], [42]). No obstante, en España, quien tiene la responsabilidad y obligación de velar por la seguridad y protección de la información en el entorno criptográfico es el CCN [140], por lo que esta Guía es de obligado cumplimiento. Esto es, los productos criptográficos aquí recomendados tienen prioridad para su aplicación y uso en el territorio nacional, sean cuales sean las recomendaciones y sugerencias que otros países puedan publicar en sus correspondientes ámbitos de actuación.

6. No obstante, el CCN, siendo consciente de la globalización de todos los aspectos relacionados con la seguridad y protección de la información y siendo consciente de que los ataques que los adversarios puedan plantear pueden proceder de cualquier lugar o país, ha tenido en cuenta el estado del arte de esta cuestión, las recomendaciones de otros países, organismos como SOG-IS (*Senior Officials Group-Information Systems Security*<sup>1</sup>) y estándares, incluidos los Criterios Comunes (CC o *Common Criteria*<sup>2</sup>).
7. Este documento pretende, por tanto, ser una guía enfocada a proporcionar las características de los productos que garantizan seguridad contra quienes pretendan atacar y vulnerar la confidencialidad, integridad, autenticación y no repudio de la información que deba ser protegida de accesos no permitidos. Además, el documento contiene consejos y recomendaciones relacionados con la posible implementación de diferentes mecanismos, de modo que puedan ser de utilidad para los desarrolladores y evaluadores nacionales.
8. A lo largo de esta guía se incluirán diferentes cajas de texto, con fondos de diferentes colores, con información relevante y destacada. A continuación se muestra un ejemplo de cada una de ellas señalando el tipo de información que contiene.

9. **[Definición:]** estas cajas de texto con fondo verde contienen la definición del término que aparece en negrita y entre corchetes al inicio del mismo.

10. **[Recomendación:]** las cajas de texto con fondo amarillo contienen recomendaciones a tener en cuenta, pero sin que tengan un carácter obligatorio.

11. **Nota 1 (Consideraciones para desarrolladores y evaluadores.)** Este tipo de caja de texto con fondo gris hace referencia a los aspectos que se deben tener en cuenta a la hora de desarrollar o evaluar los diferentes mecanismos. Cada caja de texto puede considerarse como una nota que va numerada y está referenciada en la columna de «Notas» de las tablas que resumen las características de los mecanismos recomendados o heredados.

12. Por otra parte, es importante señalar que en esta Guía se consideran, inicialmente, dos tipos de productos criptográficos, fundamentalmente relacionados con su perdurabilidad en el tiempo, a tenor de la fortaleza o robustez que ofrecen, en tanto no se conozcan nuevos ataques, vulnerabilidades o debilidades que pongan en entredicho tal fortaleza. Por ello, se llevará a cabo una actualización permanente

<sup>1</sup><https://www.sogis.eu/>

<sup>2</sup><https://commoncriteriaportal.org/>

de esta Guía que tenga en cuenta los posibles ataques que se publiquen, ya sean de tipo criptoanalítico, por canal lateral e inducción de fallos, o por el desarrollo de la computación cuántica.

13. Los dos tipos de productos se clasifican en:

- **Mecanismos recomendados:** son los mecanismos criptográficos que ofrecen una seguridad probada de, al menos, 128 bits. Tal nivel de seguridad ha sido mostrado en publicaciones científicas (o son considerados estándares), de modo que el mismo está respaldado por argumentos científicos. Dicho de otro modo, no se conocen debilidades o amenazas en el estado del arte y la técnica que puedan poner tales argumentos en entredicho, ya sea considerando sus aspectos matemáticos o de ingeniería en seguridad criptográfica.

Si alguno de los mecanismos recomendados que se consideran en esta Guía fuera objeto de algún ataque que pusiera en duda su seguridad, en futuras versiones de esta Guía, tal hecho será tenido en cuenta y se adoptarían las medidas oportunas en función de la calidad del ataque.

- **Mecanismos heredados:** son aquellos mecanismos criptográficos de uso muy extendido que ofrecen un nivel de seguridad de, al menos, 100 bits, de tal manera que su seguridad es solo aceptable a corto plazo. Esto es, son mecanismos que deben dejar de utilizarse en un corto plazo de tiempo porque el estado del arte ha demostrado que su garantía de seguridad se ha visto comprometida. Para estos mecanismos heredados, se ha considerado un periodo de validez. La principal razón para que el uso de estos mecanismos se mantenga durante determinado periodo de tiempo se debe a razones de compatibilidad dado que están implementados a gran escala y necesitan de un tiempo para ser sustituidos por otros más seguros (los recomendados).

Una vez concluido tal periodo de validez, el mecanismo se considerará obsoleto y no podrá ser utilizado al amparo de esta Guía. Así, la notación  $H[2030]$  ( $H$  señala que es «heredado») significa que el periodo de validez concluye en diciembre de 2030, esto es, que tal mecanismo se acepta hasta esa fecha. Por su parte, la expresión  $H[2030+]$  indica que el periodo de validez es mínimo, es decir, que la validez del mecanismo dado será, al menos, hasta finales de 2030; siendo posible que su validez sea ampliada en futuras versiones de esta Guía.

14. [Mecanismos autorizados:] son aquellos que ofrecen un nivel de seguridad criptográfica probada de, al menos, 128 bits. El periodo de validez, por defecto, para los mecanismos heredados se establece en  $H[2030+]$ .

15. [Mecanismos heredados:] son aquellos que ofrecen un nivel de seguridad criptográfica de entre 100 y 128 bits.

16. Dado que no es posible establecer a priori durante cuánto tiempo un mecanismo criptográfico permanecerá siendo seguro, debido a la publicación de nuevos ataques o a la mejora en los tiempos de computación que puedan vulnerar los algoritmos en los que tales mecanismos se basan, esta Guía puede considerarse conservadora en el sentido de que primará la seguridad sobre cualquier otro aspecto. Por ello, los nuevos mecanismos criptográficos que puedan recomendarse en el futuro deberán utilizar los algoritmos y los tamaños de clave que hayan sido probados seguros.
17. Así, si el estado del arte lo aconseja, es posible que un mecanismo recomendado en una versión determinada de esta Guía, pase a ser considerado como heredado en versiones posteriores, si existen razones de peso para ello, como por ejemplo, por compatibilidad con implementaciones o arquitecturas previas. Esta cuestión es de especial relevancia si, por ejemplo, los avances de la computación cuántica son más rápidos de los que se consideran en la literatura.
18. De hecho, es sabido que los dos problemas computacionalmente difíciles en los que se fundamenta la seguridad de la mayoría de los criptosistemas asimétricos (o de clave pública), como son el problema de la factorización de enteros o IFP (*Integer Factorization Problem*) y el problema del logaritmo discreto o DLP (*Discrete Logarithm Problem*) son vulnerables si se desarrolla un ordenador cuántico con la suficiente potencia de cómputo, dicho de otro modo, Shor publicó sendos algoritmos cuánticos que rompen ambos problemas en tiempo polinómico [195]. De modo análogo, es conocido que si se dispusiera de un ordenador cuántico con la suficiente capacidad de cómputo, los criptosistemas de clave simétrica (o de clave secreta) verían restringida su seguridad a la mitad de las longitudes de las claves actuales, dado que el algoritmo propuesto por Grover reduce el tiempo de búsqueda por un ataque de fuerza bruta a la raíz cuadrada del tiempo actual [78]. Además, resultados posteriores al de Grover, basados en la paralelización del algoritmo de Simon, apuntan a que esta seguridad podría reducirse aún más [108], [109].
19. La situación que acaba de comentarse sobre la vulnerabilidad cuántica de los principales mecanismos criptográficos actuales pone sobre la mesa uno de los problemas a los que se enfrenta la protección de la información almacenada en la actualidad. Se trata de tomar medidas de modo que esta información no pueda ser desvelada en el futuro si la computación cuántica (u otros ataques) es capaz de vulnerar los sistemas empleados en la actualidad y que la están protegiendo.
20. Así, por ejemplo, si la criptografía simétrica puede ver reducida su seguridad a la mitad de las longitudes de las claves usadas hoy en día, debería recomendarse el uso de sistemas cuya seguridad equivalente sea considerada segura, esto es, al menos, 256 bits (ya hemos mencionado que los mecanismos recomendados son aquellos que ofrecen una seguridad probada de, al menos, 128 bits).
21. Por su parte, para la criptografía asimétrica debería recomendarse el uso de soluciones híbridas de modo que la combinación de un mecanismo criptográfico determinado con un mecanismo criptográfico que se considere resistente a la

computación cuántica, garantizaría por un lado, su seguridad frente a los ataques precuánticos (actuales), a la vez que proporcionaría determinada seguridad frente a los ataques cuánticos.

22.

[**Protección cuántica:**] para los criptosistemas simétricos debería recomendarse el uso de claves que proporcionen una seguridad de, al menos, 256 bits; mientras que para la criptografía asimétrica debería recomendarse el uso de soluciones híbridas.

## 1.2. MECANISMOS CRIPTOGRÁFICOS

Con el fin de establecer una terminología clara a lo largo de esta Guía, presentamos a continuación los principales conceptos utilizados. Podría considerarse, no sin razón, que las definiciones que siguen que distinguen, por ejemplo, esquemas criptográficos de protocolos criptográficos o primitivas de construcciones, son en cierto modo arbitrarias, dado que los límites entre los diferentes conceptos pueden ser difusos.

### 1.2.1. ALGORITMOS, PROTOCOLOS Y ESQUEMAS

Con el fin de tener una referencia externa, seguimos la propuesta de [190], que está en consonancia con las prácticas comunes de la comunidad criptográfica y parece ser compartida por otros organismos internacionales como [197].

- Un **Algoritmo Criptográfico** es una transformación, esto es, un conjunto de pasos, bien definida de modo que dado un valor de entrada, produce un valor de salida, logrando ciertos objetivos de seguridad.
- Un **Protocolo Criptográfico** es un algoritmo distribuido que describe con precisión las interacciones entre dos o más entidades, logrando ciertos objetivos de seguridad.

Al igual que con los algoritmos, se espera que un protocolo satisfaga ciertos objetivos de seguridad. Los protocolos criptográficos requieren la interacción entre dos o más partes y, por lo tanto, la definición de estos protocolos requiere la definición de los tipos de canales que están disponibles para estos usuarios.

- Un **Esquema Criptográfico** es un conjunto de algoritmos criptográficos y protocolos criptográficos relacionados que logran ciertos objetivos de seguridad.

21.

[**Algoritmo Criptográfico:**] conjunto de pasos que para una entrada dada proporciona una salida verificando determinados objetivos de seguridad.

22. **[Protocolo Criptográfico:]** algoritmo distribuido que describe las interacciones entre dos o más entidades, logrando ciertos objetivos de seguridad.
23. **[Esquema Criptográfico:]** conjunto de algoritmos y protocolos criptográficos relacionados que cumplen ciertos objetivos de seguridad.
24. En general, los mecanismos criptográficos se distinguen según su complejidad. Así, se denominan
- **Primitiva Criptográfica:** es el mecanismo criptográfico más elemental.
  - **Construcción Criptográfica:** es un mecanismo criptográfico que permite lograr objetivos de seguridad de mayor nivel que las primitivas.
  - **Primitiva Atómica:** es un mecanismo criptográfico que no puede describirse como una construcción o cuando tal descomposición no es una práctica habitual en la industria criptográfica.

25. **Nota 2 [Primitivas y Construcciones autorizadas]** Una construcción se considera «autorizada» solo si utiliza primitivas autorizadas, a menos que se indique explícitamente lo contrario. Por otra parte, para que una construcción señalada como «recomendada» dé como resultado un mecanismo recomendado, es necesario que las primitivas subyacentes también sean recomendadas. De hecho, si una construcción está considerada como «recomendada», pero una primitiva subyacente está marcada como «heredada», el mecanismo resultante se considerará «heredado», no recomendado.

26. Debe tenerse en cuenta que es posible que un mecanismo criptográfico dado sea simultáneamente una primitiva y una construcción, dependiendo de si forma parte como primitiva de un protocolo o de si está compuesto por otras primitivas criptográficas. A modo de ejemplo, el algoritmo de firma digital basado en curvas elípticas o ECDSA (*Elliptic Curve Digital Signature Algorithm*) puede considerarse como una construcción, dado que se basa en una función resumen (*hash*) criptográfica y en el logaritmo discreto en un grupo de puntos de una curva elíptica, y, a la vez, puede usarse como una primitiva en un protocolo de intercambio de claves autenticado.
27. Con relación a la dificultad mencionada de esos «límites difusos» en las definiciones anteriores, es importante señalar que, en el diseño de mecanismos criptográficos, la tendencia generalizada consiste en favorecer diseños modulares. Por ejemplo, la función resumen SHA-3 es una función de esponja construida sobre

una permutación con buenas propiedades criptográficas; así, podría considerarse esta función tanto una primitiva como un esquema basado en esta permutación primitiva. También, por ejemplo, un esquema de cifrado implica una comunicación entre dos partes: una que cifra el texto claro y otra que descifra el texto cifrado, por lo que en lugar de ser un esquema podría considerarse como un protocolo elemental donde hay una interacción limitada entre las dos partes.

28. Ejemplos de primitivas atómicas son, por ejemplo, el estándar de cifrado avanzado o AES (*Advanced Encryption Standard*), la función resumen SHA-256, o problemas como el de la factorización de enteros y el del logaritmo discreto sobre cuerpos finitos (ya sea en su versión multiplicativa o aditiva).
29. En este sentido, es importante señalar que la fortaleza o robustez criptográfica de una primitiva atómica es difícil de evaluar dado que, en general, se debe evaluar la dificultad computacional de un problema matemático específico. Además, es muy probable que no se pueda dar una demostración matemática efectiva de tal seguridad. De hecho, la seguridad de estos mecanismos se considera supuesta la dificultad computacional de determinado problema, esto es, de que no existe prueba evidente de que el problema puede resolverse en tiempo polinómico. Así, esta seguridad recae, en parte, en la confianza que el evaluador deposita en los resultados académicos publicados.
30. Dado que las entidades que interactúan en un protocolo criptográfico se intercambian información a través de canales de comunicación, es posible hacer una distinción básica entre los canales punto a punto que unen dos entidades y los canales de transmisión que conectan un remitente a varios receptores.
31. Muchas veces se da por hecho que los canales de comunicación ofrecen garantías de seguridad específicas. Así, suele entenderse que un «canal privado» (o canal seguro) es un canal punto a punto que utiliza alguna forma de cifrado para proteger la información intercambiada contra posibles escuchas y, además, es muy probable que emplee alguna forma de autenticación para evitar la manipulación de los mensajes. Por otra parte, un canal sin protección contra escuchas ilegales y manipulaciones suele denominarse «canal público» (o canal inseguro).

### 1.2.2. TIPOS DE ATAQUE

32. Los «modelos de comunicación» describen los tipos de canales que están disponibles entre diferentes conjuntos de entidades.
33. Por otra parte, también se distinguen los ataques pasivos de los activos, de modo que ambos tipos de ataque se definen en función del adversario o atacante. Si una entidad ha caído bajo el control de un adversario se dice que es «corrupta»; mientras que las entidades restantes se consideran «honestas».

- En un **Ataque Pasivo**, el adversario no interfiere con la ejecución de los algoritmos y protocolos que componen un esquema criptográfico. Un adversario pasivo simplemente escucha la comunicación entre las entidades y registra toda la información a la que tiene acceso, incluida toda la información privada de las entidades corruptas.

Las entidades que son pasivamente corruptas también se conocen como entidades semihonestas u «honestas pero curiosas».

- En un **Ataque Activo**, el adversario, además de actuar pasivamente, puede interferir con la comunicación del esquema criptográfico borrando, inyectando o modificando mensajes. El adversario activo puede hacer que las entidades corruptas se desvíen de su comportamiento prescrito de manera arbitraria.

Las entidades activamente corruptas suelen denominarse entidades «maliciosas» o tramposas.

34. [Ataque Pasivo:] ataque en el que el adversario no interfiere, solo escucha y registra, la comunicación entre las entidades.

35. [Ataque Activo:] ataque donde el adversario, además, interfiere en la comunicación del esquema criptográfico borrando, inyectando o modificando mensajes.

36. A modo de ejemplo, un ataque pasivo a un esquema de cifrado podría corresponderse con el comportamiento clásico de un fisgón; mientras que el conocido ataque del hombre en el medio o MitM (*Man-in-the-Middle*) a un protocolo de intercambio de claves es un caso de ataque activo.

### 1.2.3. ESQUEMAS SIMÉTRICOS Y ASIMÉTRICOS

Finalmente, haremos mención a los conceptos de simetría y asimetría relacionados con los mecanismos criptográficos. Dado que, generalmente, estos mecanismos hacen uso de claves criptográficas y que su seguridad se fundamenta no solo en mantener la confidencialidad e integridad de las claves, sino también en cómo las mismas son empleadas en los diferentes esquemas criptográficos y sus correspondientes operaciones, es tradicional considerar los siguientes dos tipos de esquemas criptográficos:

- **Esquema criptográfico simétrico:** es un esquema en el que la clave utilizada por cada parte es conocida o puede ser derivada por la otra parte. Las claves empleadas son únicas e idénticas para todas las partes que intervienen.

En los esquemas simétricos, los intervinientes suelen llevar a cabo procesos análogos y comparten los parámetros a utilizar, de ahí el calificativo de «simétrico».

- **Esquema criptográfico asimétrico:** es aquel esquema que no es simétrico. Generalmente, estos esquemas reciben este nombre porque las acciones que lleva a cabo o los parámetros que utiliza cada parte son diferentes.

37. Los esquemas criptográficos simétricos y asimétricos siguen diseños muy diferentes y, por extensión, los mecanismos criptográficos se clasifican de forma análoga a los esquemas. Es interesante indicar que las funciones resumen son mecanismos criptográficos que no emplean claves y se consideran primitivas criptográficas simétricas.

### 1.3. COMPLEJIDAD DE UN ATAQUE Y NIVELES DE SEGURIDAD

38. Es costumbre definir el «nivel de seguridad» de un mecanismo criptográfico como el número de operaciones necesarias para que un adversario rompa con éxito la seguridad proporcionada por dicho mecanismo. Este número se expresa como un logaritmo en base 2, de modo que una seguridad de 100 bits significa que son necesarias  $2^{100}$  operaciones para romper el mecanismo.

#### 1.3.1. COMPLEJIDAD DE UN ATAQUE

39. Las diferentes métricas de complejidad que evalúan el coste de ejecutar un ataque son las siguientes:

- La **Complejidad temporal** hace referencia a la cantidad de cálculos fuera de línea (*offline*) requeridos por el ataque, basados en datos extraídos del mecanismo criptográfico. A veces a esta complejidad se la conoce como «complejidad fuera de línea».

Debe tenerse en cuenta que la complejidad temporal no depende de la potencia computacional de que dispone el atacante, de modo que la misma se establece sin considerar el modelo de computación que use el atacante y así es consistente con las mejoras que puedan producirse en la tecnología. Si el ataque es paralelizable, el atacante puede aprovechar esta característica para rebajar el tiempo que necesitaría si no lo fuera, pero esta propiedad no modifica la complejidad temporal porque no reduce la cantidad total de cálculos necesarios para llevar a cabo el ataque.

Por ejemplo, la complejidad temporal de una búsqueda exhaustiva de claves AES-128 es (aproximadamente)  $2^{128}$  operaciones.

- La **Complejidad en memoria** considera la cantidad de almacenamiento que es necesario para realizar el ataque.

Por ejemplo, la implementación de Nohl y Krißler [173] de la compensación de tiempo/memoria de Barkan, Biham y Keller [21] en el algoritmo A5/1 requiere 2 TB de memoria para almacenar las tablas del arco iris.

- La **Complejidad en datos** tiene en cuenta la cantidad de interacciones que el adversario necesita realizar con el mecanismo criptográfico para desarrollar el ataque contra él. Estas interacciones pueden corresponder a la cantidad de datos que el adversario necesita extraer de la ejecución de los mecanismos criptográficos para poder realizar el ataque, o a la cantidad de datos que el adversario debe enviar para lograr un resultado determinado. Esta complejidad suele calificarse como «complejidad en línea».

A modo de ejemplo, el criptoanálisis lineal contra el cifrado de datos estándar o DES (*Data Encryption Standard*) de Matsui [130] requiere  $2^{43}$  textos claros conocidos, y el ataque de oráculo PKCS#1 v1.5 (*Public-Key Cryptography Standard*) de Bleichenbacher [32] al criptosistema RSA propuesto por Rivest, Shamir y Adleman [184] de 1024 bits necesita alrededor de un millón de consultas al oráculo de relleno.

40. [Complejidad temporal:] es la cantidad de cálculos fuera de línea necesarios para realizar con éxito un ataque criptográfico.
41. [Complejidad en memoria:] es la cantidad de almacenamiento necesario para ejecutar exitosamente un ataque.
42. [Complejidad en datos:] es la cantidad de interacciones que el adversario necesita realizar con el mecanismo criptográfico para desarrollar el ataque.
43. Todas estas medidas de complejidad también se pueden combinar compensando unas con otras. Así, es posible disminuir la complejidad temporal de un ataque aumentando la complejidad en memoria o en datos.

### 1.3.2. NIVELES DE SEGURIDAD

44. Al hablar del nivel de seguridad de un mecanismo criptográfico, en general, se entiende que se está considerando como la complejidad temporal del mejor ataque conocido, de modo que las complejidades en memoria y en datos pasan a un segundo término. De hecho, el coste de las operaciones necesarias para la escritura de la información en la memoria y el procesamiento de los datos se supone que ya se

han tenido en cuenta cuando se determinó la complejidad temporal de determinado ataque. Esto es, el nivel de seguridad de un mecanismo no puede ser menor que la complejidad en memoria o la complejidad en datos del mejor ataque conocido al mecanismo.

45. [Nivel de seguridad de un mecanismo criptográfico:] es la complejidad temporal del mejor ataque conocido contra dicho mecanismo. La complejidad en memoria y en datos pasan a un segundo término.
46. Al diseñar o evaluar un mecanismo criptográfico, debe llevarse a cabo un análisis sobre la fortaleza del mismo, de modo que debe haber ciertas garantías de que ningún atacante pueda romper la seguridad del mecanismo. Por ello, al proponer el uso de un mecanismo dado, se deben recomendar, a la vez, determinados parámetros, en función de la seguridad deseada. Así, es importante recomendar, por ejemplo, el tamaño de la clave y el tamaño del estado interno.
47. Algunos principios que deben considerarse en el proceso de evaluación de los mecanismos criptográficos son los siguientes:
- Como ya se ha mencionado anteriormente, los mecanismos recomendados deben proporcionar una seguridad de, al menos, 128 bits contra ataques temporales (fuera de línea). Una seguridad de 100 bits solo es aceptable para los mecanismos heredados, que claramente proporcionan una seguridad más baja.
  - En situaciones muy especiales, se podría aceptar una complejidad temporal menor. Podría considerarse esta situación cuando se utilizan claves de sesión, esto es, claves de un solo uso, o en los mecanismos criptográficos denominados ligeros, como los implementados en etiquetas de identificación por radio frecuencia o RFID (*Radio Frequency Identification*), cuyos recursos computacionales son limitados.
  - Los niveles de seguridad aceptables asociados a la complejidad en datos suelen ser menores. Esto se debe a que adquirir o transmitir datos a un proceso, dispositivo, etc., que utiliza alguna clave secreta precisa comunicarse con dicho dispositivo. En este caso, hay restricciones en la comunicación que vienen determinadas por las limitaciones físicas del dispositivo (tarjeta inteligente, etc.), o que pueden ser impuestas por el mismo (número máximo de intentos de identificación mediante un PIN, tiempo de caducidad de las claves, etc.). Todo ello indica que basta con estudiar la seguridad del mecanismo contra los atacantes cuya complejidad en datos está limitada o restringida.
- Por ejemplo, dado que en una red a 100 GB, el tiempo necesario para intercambiar  $2^{64}$  bloques de AES es de más de setecientos años, los posibles ataques contra AES que requieran acceder a más de  $2^{64}$  bloques de datos no son un problema real.

- Es muy importante tener en cuenta el estado del arte relacionado con la seguridad de los mecanismos criptográficos. Es posible que cuando un mecanismo fuera propuesto y aceptado (incluso considerado como un estándar) para su uso con determinados valores de sus parámetros, sea criptoanalizado con posterioridad. Es decir, cabe en lo posible que investigaciones posteriores hayan determinado que el mecanismo no es seguro para determinado conjunto de parámetros. En este caso, la existencia de posibles ataques invalidan la propuesta original, incluso en el caso de que estos nuevos ataques no tengan un impacto inmediato práctico.
48. Ya hemos mencionado que el concepto de nivel de seguridad de un mecanismo se expresa en términos de la cantidad mínima de cálculos que deben realizarse para romperlo y que ello se traduce en términos del tamaño de las claves. Conviene, por tanto, señalar que cuando mencionamos la noción de «longitud o tamaño de una clave», nos estamos refiriendo a la entropía del mecanismo que genera dicha clave. De forma más concreta, una clave simétrica almacenada en  $k$  bits, elegida uniformemente al azar de entre todas las claves posibles, tienen una longitud de, como máximo,  $k$  bits.
  49. A modo de ejemplo, es sabido que las claves de DES se almacenan en 64 bits y que 8 de esos bits (el último de cada byte) son redundantes, esto es, son los bits de paridad que verifican que cada grupo de 7 bits que precede a cada uno de ellos es correcto. Así pues, la longitud de la clave de DES es, realmente, de solo 56 bits. Como consecuencia, la longitud de la clave de Triple-DES con 2 claves (resp. 3 claves) es de 112 bits (resp. 168 bits), que es estrictamente menor que los 128 bits (resp. 192 bits) utilizados para su almacenamiento.
  50. En el caso de la claves correspondientes a los mecanismos asimétricos, la relación entre la longitud de la clave y el nivel de seguridad no es tan directa y tan sencilla de expresar. De hecho, en general, la seguridad de una clave asimétrica depende de un problema matemático, supuestamente difícil de resolver, y de la complejidad temporal del mejor algoritmo conocido que sea capaz de resolver dicho problema.
  51. A modo de ejemplo, si la clave,  $n$ , de un mecanismo criptográfico asimétrico es producto de dos números primos con, aproximadamente, el mismo tamaño, y tiene una longitud de  $b = 1024$  bits (resp.  $b = 2048$  bits) y el mejor algoritmo conocido para resolver el problema en el que se basa la seguridad de dicho mecanismo requiere  $\mathcal{O}\left(e^{(1,92 \cdot b)^{1/3} \cdot (\log_2(b))^{2/3}}\right)$  operaciones<sup>3</sup>, se deduce que la seguridad proporcionada por esta clave es de  $2^{83,89}$  operaciones (resp.  $2^{112,63}$  operaciones).

<sup>3</sup>Se ha elegido esta función a modo de ejemplo y con fines didácticos, aunque está estrechamente relacionada con el problema de la factorización de enteros, que es la base de la seguridad del criptosistema RSA.

52. En efecto, basta tener en cuenta que

$$\begin{aligned} \mathcal{O}\left(e^{(1,92 \cdot b)^{1/3} \cdot (\log_2 b)^{2/3}}\right) &= \mathcal{O}\left(e^{1966,08^{1/3} \cdot \log_2(1024)^{2/3}}\right) = \mathcal{O}\left(e^{12,624 \cdot 10^{2/3}}\right) \\ &= \mathcal{O}\left(e^{12,624 \cdot 4,651}\right) = \mathcal{O}\left(e^{58,729}\right) \\ &= \mathcal{O}\left(1,791 \cdot 10^{25}\right) \approx \mathcal{O}\left(2^{83,89}\right), \\ \mathcal{O}\left(e^{(1,92 \cdot b)^{1/3} \cdot (\log_2 b)^{2/3}}\right) &= \mathcal{O}\left(e^{3932,16^{1/3} \cdot \log_2(1024)^{2/3}}\right) = \mathcal{O}\left(e^{15,783 \cdot 11^{2/3}}\right) \\ &= \mathcal{O}\left(e^{15,783 \cdot 4,946}\right) = \mathcal{O}\left(e^{78,067}\right) \\ &= \mathcal{O}\left(8,024 \cdot 10^{33}\right) \approx \mathcal{O}\left(2^{112,63}\right), \end{aligned}$$

que sería equivalente a una seguridad de unos 83 bits en el primer caso y de unos 112 en el segundo.

#### 1.4. ORGANIZACIÓN DE LA GUÍA

53. Después de este primer capítulo introductorio, el resto de esta Guía se organiza de la siguiente manera. En el capítulo 2 se presentan las primitivas atómicas y construcciones criptográficas simétricas. El capítulo 3 introduce las primitivas atómicas y construcciones criptográficas asimétricas. El capítulo 4 incluye uno de los principales protocolos criptográficos de comunicación como es el protocolo de seguridad de la capa de transporte o TLS (*Transport Layer Security*), que será complementado en el futuro con otros protocolos. Los generadores de números aleatorios se detallan en el capítulo 5; mientras que los protocolos de gestión de claves se contemplan en el capítulo 6. El capítulo 7 incluye los mecanismos de autenticación e identificación y el capítulo 8 la generación de números primos y de claves para el criptosistema asimétrico RSA, el más utilizado en la actualidad. Finalmente, en el capítulo 9 se tratan los principales fundamentos en los que se basa la criptografía postcuántica, que propone soluciones criptográficas seguras para ser implementadas en ordenadores como los actuales, pero capaces de resistir la potencia de cómputo de los futuros ordenadores cuánticos.
54. Cada una de las primitivas y construcciones autorizadas estará acompañada de una tabla en la que se incluirán sus nombres o denominaciones, los tamaños de los parámetros (si ha lugar), si se considera recomendada o heredada (denotándose cada propiedad como R o H, respectivamente) y otros aspectos que puedan ser de interés.
55. Este documento incluye, además, un Glosario de términos para facilitar su lectura y una Bibliografía que permitirá al lector ampliar determinados aspectos introducidos en esta Guía, en la que no tiene cabida un desarrollo en profundidad de los mismos.

## 2. MECANISMOS SIMÉTRICOS

56. Como ya se ha mencionado en la Sección 1, la seguridad de la criptografía simétrica se basa, fundamentalmente, en mantener la confidencialidad de una clave que comparten los usuarios legítimos que intervienen en un proceso criptográfico dado. En esta sección trataremos los mecanismos que entran dentro del ámbito de los considerados simétricos.

### 2.1. PRIMITIVAS SIMÉTRICAS (§10.1)

57. La criptografía simétrica (o de clave secreta) se caracteriza porque los intervinientes es un proceso de cifrado utilizan una única clave para cifrar y descifrar la información que se intercambian. Dicha clave es la misma para ambos procesos, por lo que debe ser compartida. La seguridad de estos mecanismos se basa, por lo tanto, en mantener en secreto tal clave compartida.

58. En esta sección presentamos las primitivas atómicas simétricas aprobadas, para describir posteriormente las construcciones simétricas construidas sobre estas primitivas.

#### 2.1.1. CIFRADORES EN FLUJO

59. El procedimiento para cifrar en flujo un texto claro de  $L$  bits consiste en generar una secuencia de  $L$  bits, aparentemente aleatorios, a partir de una clave secreta corta de  $k$  bits, que es conocida solo por las dos partes interesadas), y un algoritmo público que genera la secuencia de los  $L$  bits. Esta secuencia generada se denomina secuencia de flujo de claves (*keystream sequence*).

60. Para el cifrado, el remitente realiza la operación XOR bit a bit entre los bits del texto claro y la secuencia de flujo de claves. El resultado es el texto cifrado que se enviará al receptor. Para el descifrado, el receptor genera la misma secuencia de flujo de claves, realiza la misma operación XOR bit a bit entre el texto cifrado recibido y la secuencia generada, recuperando el texto claro original. El objetivo es hacer prácticamente imposible para un adversario la recuperación del texto claro a partir del texto cifrado sin el conocimiento de la clave secreta  $k$ . Los cifrados en flujo se utilizan ampliamente debido a su alta eficiencia y la idoneidad del hardware.

61. En la versión actual de esta Guía no se recomienda ningún cifrado en flujo concreto, por lo que no se presenta ninguna tabla de primitivas de cifrado.

62. Es sabido que en 2004 se lanzó el proyecto eSTREAM como parte de ECRYPT (European Network of Excellence in Cryptology<sup>4</sup>). El proyecto es un esfuerzo de

<sup>4</sup><http://www.ecrypt.eu.org/stream/>

varios años para identificar nuevos cifrados en flujo que podrían ser adecuados para una adopción generalizada. Como resultado, en 2008 se preseleccionaron siete finalistas (cuatro de perfil software y tres de perfil hardware), todos ellos fueron declarados aptos porque todos ellos son algoritmos rápidos y adecuados para el cifrado en flujo, pero no se han convertido en estándares.

### 2.1.2. CIFRADORES EN BLOQUE

63. El cifrado en bloque es un método que permite cifrar cada uno de los bloques en los que se divide un mensaje de texto claro, cada uno de ellos de la misma longitud, sea  $L$  bits, dando como resultado un bloque cifrado, también de  $L$  bits de longitud, la misma que el original. La operación de cifrado viene determinada por una clave secreta de  $r$  bits de longitud, elegida uniformemente al azar. La operación inversa, esto es, el descifrado de cada bloque utiliza la misma clave que para el cifrado y devuelve el bloque del texto claro original. El objetivo de este tipo de cifrado es hacer que sea prácticamente imposible recuperar el texto claro a partir del texto cifrado si no se conoce la clave secreta utilizada.
64. Desde el punto de vista de su implementación, un cifrado en bloque puede entenderse como una familia de permutaciones del conjunto  $\{0, 1\}^n$ , cada una de las cuales se selecciona haciendo uso de un parámetro de  $r$  bits llamado clave. Los valores  $n$  y  $r$  representan, respectivamente, el tamaño en bits del bloque y el tamaño de la clave.
65. En la Tabla 2.1 se listan las primitivas autorizadas, los tamaños de sus parámetros, si se considera Recomendada o Heredada y otros aspectos que puedan ser de interés. La expresión  $\ll H[aaaa] \gg$  indica que la primitiva dejará de considerarse Heredada el año indicado (aaaa) y será eliminada de esta guía criptográfica.

Primitiva	Parámetros (bits)	Rec./Her.	Referencias	Notas
AES	$k = 128$	R	[54], [152], [94]	
	$k = 192$	R		
	$k = 256$	R		

Tabla 2.1: Tipos autorizados de cifradores en bloque

### 2.1.3. FUNCIONES RESUMEN

66. Una función resumen (*hash* en inglés) es una función, sin clave, computacionalmente eficiente,  $h$ , que aplica cadenas binarias de longitud arbitraria en cadenas binarias de longitud fija [135, Cap. 9]. La salida de este tipo de funciones se llama «resumen» o «valor hash». La idea principal que subyace a estas funciones es que su salida o resumen puede usarse como una representación compacta de una cadena de

entrada de longitud arbitraria. Las funciones resumen se emplean, preferentemente, en esquemas de firma digital y para verificar la integridad de datos.

67. Las funciones resumen deben verificar las siguientes cuatro propiedades para ser consideradas seguras:
1. «Resistencia a la preimagen»: para un resumen dado,  $y$ , es computacionalmente imposible encontrar una entrada,  $x$ , cuya salida sea precisamente  $y$ , es decir, es inviable computacionalmente encontrar una preimagen  $x$  tal que  $h(x) = y$ . En otras palabras, debe ser difícil invertir la función  $h$ .
  2. «Resistencia a la segunda preimagen»: dada una entrada  $x_1$ , es computacionalmente imposible encontrar una segunda entrada diferente,  $x_2$ , que tenga la misma salida, es decir, encontrar una segunda preimagen  $x_2 \neq x_1$  tal que  $h(x_1) = h(x_2)$ .
  3. «Resistencia a colisiones»: no es factible, computacionalmente hablando, encontrar dos entradas distintas  $x_1$  y  $x_2$  que tengan la misma salida, es decir, que  $h(x_1) = h(x_2)$ .
  4. «Resistencia cercana a la colisión»: es difícil encontrar dos entradas distintas,  $x_1$  y  $x_2$  de modo que la diferencia entre  $h(x_1)$  y  $h(x_2)$  ( $h(x_1) \oplus h(x_2)$ ) sea pequeña.
68. Es sabido que las funciones *hash* de la familias SHA2 y SHA3 verifican estas condiciones, por lo que se consideran funciones autorizadas y de ahí su inclusión en esta guía.
69. En la Tabla 2.2 se listan las funciones resumen autorizadas, los tamaños de sus resúmenes ( $h$ ) y la denominación de la función resumen correspondiente, si se considera Recomendada o Heredada y con qué plazo de caducidad y las referencias apropiadas.

Primitiva	Parámetros (bits)	Rec./Her.	Referencias	Notas
SHA-2	$h = 256$ (SHA-256)	R	[135], [163], [101]	
	$h = 384$ (SHA-384)	R		
	$h = 512$ (SHA-512)	R		
	$h = 256$ (SHA-512/256)	R		
SHA-3	$h = 256$	R	[164]	
	$h = 384$	R		
	$h = 512$	R		

Tabla 2.2: Funciones resumen autorizadas

70. La función SHA-1 no es una función resumen autorizada; sin embargo, el código de autenticación de mensajes conocido como HMAC-SHA-1, cuya construcción se basa en SHA-1, se acepta como un esquema heredado, como se verá en la sección 2.2.3.

#### 2.1.4. COMPARTICIÓN DE SECRETOS

71. Los esquemas de intercambio, división o compartición de secretos son métodos que distribuyen un secreto entre un grupo de participantes, de modo que cada participante recibe una parte o sombra del secreto. La recuperación del secreto original solo puede llevarse a cabo cuando determinado número de participantes comparten su sombra con el resto.
72. De forma más precisa, se denomina esquema de compartición de secretos  $(k, n)$ -umbral a un protocolo que permite repartir un secreto en varias sombras, una para cada participante, de modo que el conocimiento de, al menos,  $k$  sombras permite recuperar el secreto, pero es imposible hacerlo con  $k - 1$  sombras o menos [194]. Debe tenerse en cuenta que el conocimiento de cualquier sombra no aporta ninguna información sobre el secreto original.
73. La Tabla 2.3 muestra el único esquema de compartición de secretos autorizado, que corresponde a la propuesta de Shamir del año 1979.

Primitiva	Rec./Her.	Referencias	Notas
Compartición de secretos de Shamir	R	[194]	

Tabla 2.3: Compartición de secretos autorizada

## 2.2. CONSTRUCCIONES SIMÉTRICAS (§10.2)

74. Las construcciones simétricas se basan en las primitivas simétricas ya presentadas y permiten tratar una gran variedad de objetivos de seguridad. La seguridad de los esquemas simétricos con clave (secreta) se fundamenta, al igual que las primitivas en las que se basan, en la confidencialidad e integridad de una clave secreta compartida por las partes legítimas que intervienen en el proceso o comunicación. En el capítulo 6 se tratarán los mecanismos para la generación de claves secretas y determinados aspectos sobre su gestión.

### 2.2.1. MODOS DE OPERACIÓN EN EL CIFRADO Y DESCIFRADO

75. Cada cifrador en bloque utiliza diferentes «modos de operación», los cuales permiten gestionar de diferente manera, para los procesos de cifrado y descifrado, los bloques en los que se divide el texto claro o cifrado. El objetivo que persiguen

estos modos de operación es proporcionar diferentes objetivos de seguridad a la hora de cifrar o descifrar un texto. Cada modo describe cómo aplicar de forma iterada una operación de cifrado (descifrado) en un bloque simple para transformar de modo seguro textos claros (cifrados) mayores que un único bloque.

76. La mayoría de los modos de operación utiliza una secuencia binaria única, llamada vector de inicialización o IV (*Initialization Vector*), para cada operación de cifrado. Este IV se utiliza para garantizar que se generan textos cifrados distintos aunque el mismo bloque se cifre varias veces con la misma clave. En algunos modos, el IV se puede elegir aleatoriamente.

77. Los principales modos de operación son los siguientes [71, Cap. 4]:

- Modo contador o CTR (*Counter Mode*): este modo es equivalente a un cifrador en flujo pues se crea una serie cifrante bloque a bloque, cifrando con el cifrador en bloque empleado, que luego se suma módulo 2 (XOR) con los sucesivos bloques del texto claro o del cifrado. El proceso de cifrado de los bloques permite ahorrar tiempo puesto que se puede hacer con anterioridad al disponerse del texto claro o cifrado. Es claro que la longitud de palabra del contador debe ser igual a la del bloque del cifrador que se esté usando.

Para cada clave que se emplee, el contenido de cada contador debe ser diferente y no debe ser reutilizado. Una forma sencilla de conseguir este fin es construir de forma aleatoria un número de uso único (*nonce* o *number used once*) como cabecera del mensaje cifrado. El contenido del primer contador será el propio *nonce*, los contenidos de los siguientes contadores serán los sucesivos incrementos del primer contador. En la Figura 1 se muestra el esquema de este modo, tanto para el proceso de cifrado como para el de descifrado.

- El modo realimentación de la salida u OFB (*Output Feedback*) convierte al cifrador en bloque que se utilice en un auténtico cifrador en flujo. En este caso, se considera un vector inicial que se cifra de forma repetida  $N$  veces, obteniéndose  $N$  bloques de la misma longitud que el bloque del cifrador y que constituyen una secuencia cifrante. El mensaje cifrado es la secuencia de los bloques del mensaje en claro sumados módulo 2 (XOR) con los bloques de la secuencia cifrante. El vector inicial debe ser secreto, aleatorio y de uso único para cada mensaje. En la Figura 2 se muestra el esquema de este modo, primero cuando se cifra y después cuando se descifra.
- El modo de encadenamiento de bloques de cifrado o CBC (*Cipher-Block Chaining*) realimenta los sucesivos bloques a cifrar antes de que estos entren en el cifrador de la etapa siguiente. La salida de cada etapa se suma módulo 2 (XOR) con el bloque del texto claro siguiente, para producir el texto cifrado. En este caso, no es necesario que el IV sea secreto, pero sí ha de tener la misma longitud del bloque claro y tiene que ser elegido aleatoriamente para evitar ataques por repetición. Una desventaja de este modo de cifrado es que

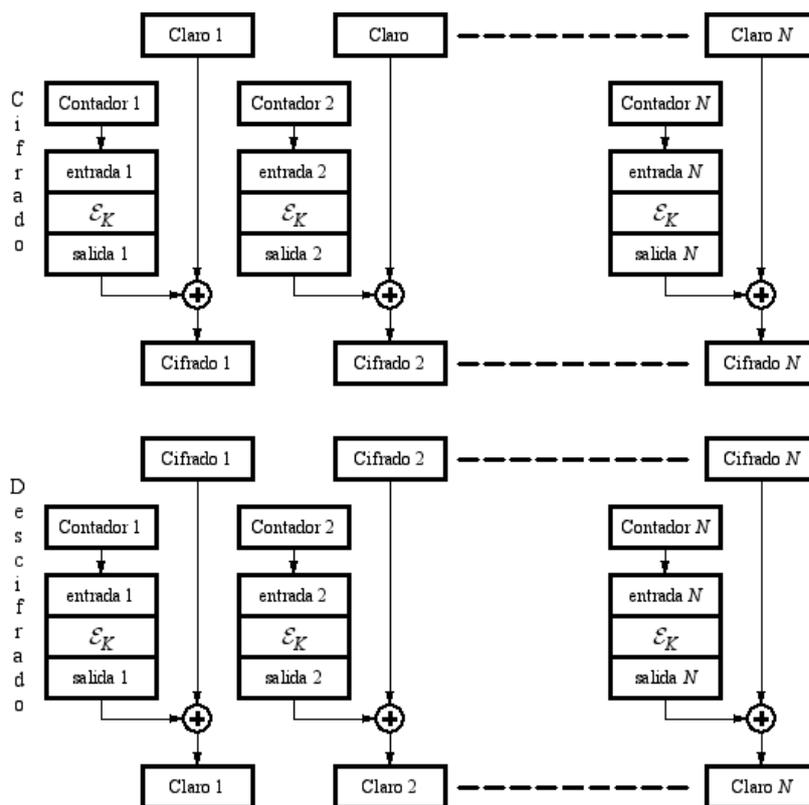


Figura 1: Modo CTR para cifrado y descifrado

debe realizarse de forma secuencial, por lo que no puede ser paralelizado. La Figura 3 muestra los esquemas de cifrado y descifrado para de este modo.

- El modo de encadenamiento de bloques de cifrado con robo de texto cifrado o CBC-CS (*Cipher-Block Chaining-Ciphertext Stealing*) es una forma de cifrado que permite que el último bloque en claro pueda ser cifrado sin necesidad de rellenarlo de forma especial para que tenga el tamaño exigido a todos los bloques según el cifrador que se esté utilizando.

El proceso consiste en modificar solo la forma de operar con los dos últimos bloques del texto claro, de modo que una parte del texto cifrado del penúltimo bloque se «roba» para rellenar el último bloque de texto claro. A continuación, el bloque final rellenado se cifra como los demás. El texto cifrado final, para los dos últimos bloques, consiste en el penúltimo bloque parcial, donde la parte «robada» se omite, más el bloque final completo, que son del mismo tamaño que el texto en claro original. En el proceso de descifrado, es preciso descifrar el último bloque antes que el penúltimo para luego restaurar el texto cifrado robado en el penúltimo bloque, que luego se puede descifrar como el resto.

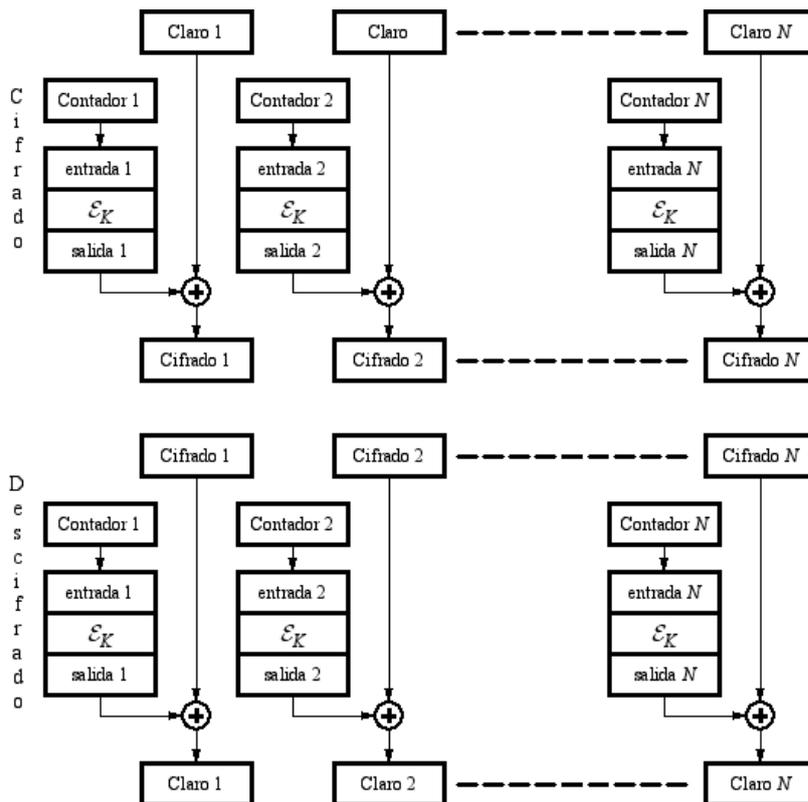


Figura 2: Modo OFB para cifrado y descifrado

- En el modo realimentación del texto cifrado o CFB (*Cipher Feedback Mode*) no se divide el texto claro en bloques de la longitud determinada por el cifrador que se utilice,  $b$ , sino en segmentos de longitud menor  $s$ , con  $1 \leq s \leq b$ . Para el cifrador AES, los tamaños recomendados para  $s$  son 1, 8, 64 y 128 bits.

El primer bloque cifrado se obtiene cifrando un vector inicial de la longitud que señale el cifrador utilizado, sea  $b$ . Del bloque cifrado de  $b$  bits se seleccionan los  $s$  bits más significativos, que se suman módulo 2 (XOR) con el primer segmento de  $s$  bits del bloque claro siguiente, obteniéndose el primer segmento cifrado de  $s$  bits. Los sucesivos tramos de cifrado se efectúan partiendo de unos bloques de entrada en los que los  $b - s$  bits más significativos son los  $b - s$  bits menos significativos del bloque de entrada anterior, completándose el bloque para que alcance la longitud  $b$ , por concatenación con los  $s$  bits del bloque cifrado anterior. No se requiere que el IV sea secreto, pero sí que sea elegido aleatoriamente para evitar ataques por repetición. Los procesos de cifrado y descifrado de este modo se presentan en la Figura 4.

78. En la Tabla 2.4 se muestran los diferentes modos de esquemas de cifrado simétrico con sus principales características.

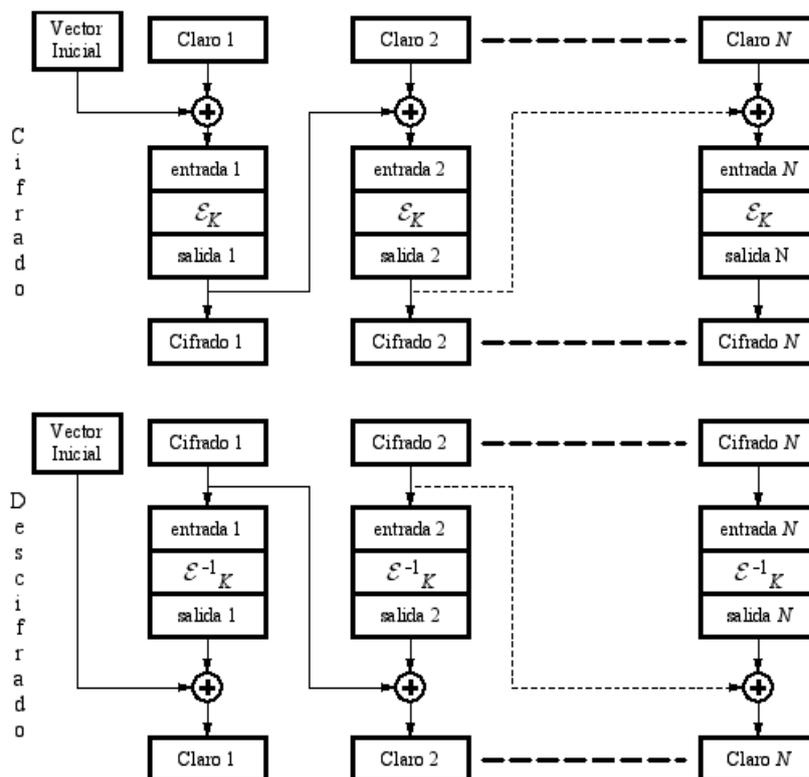


Figura 3: Modo CBC para cifrado y descifrado

Modo de operación	Rec./Her.	Referencias	Notas
CTR	R*	[153], [99]	3, 4, 5
OFB	R*	[153], [99]	3, 4, 5
CBC	R*	[153], [99]	3, 4, 6
CBC-CS	R*	[161]	3, 4
CFB	R*	[153], [99]	3, 4, 6

Tabla 2.4: Modos autorizados de cifrado simétrico

79.

Nota 3 [Tipo de Vector de Inicialización] Con el fin de proporcionar una seguridad en un sentido estricto, el esquema de cifrado considerado debe ser probabilístico y generar un IV aleatorio para iniciar el proceso cifrado, o utilizar una entrada adicional, como un *nonce*. Cada una de las especificaciones de los diferentes modos de operación describen qué se utiliza, si un IV aleatorio o un *nonce*. Las implementaciones de estos modos deberán seguir estas especificaciones.

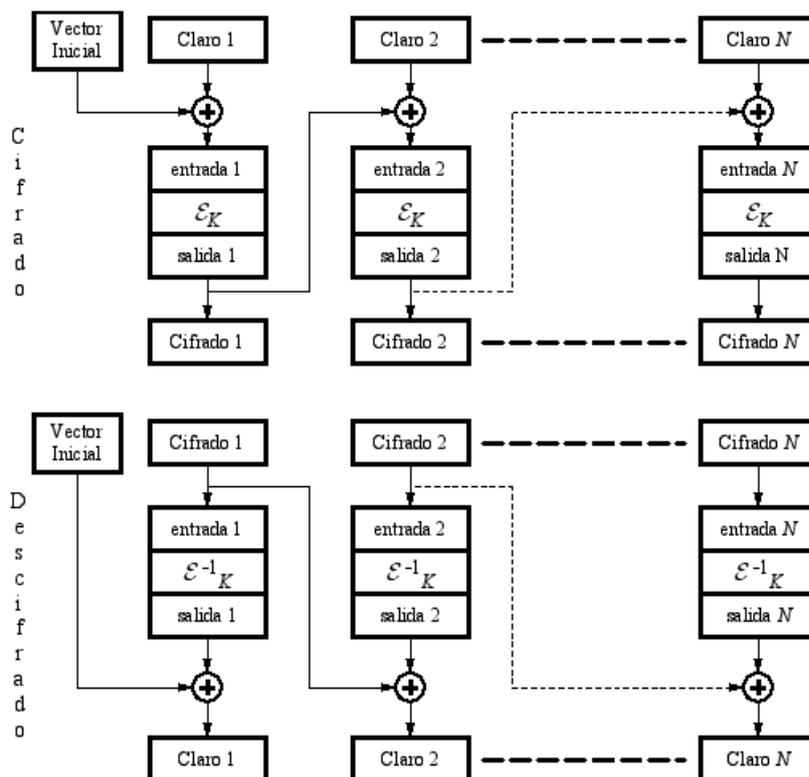


Figura 4: Modo CFB para cifrado y descifrado

80.

Nota 4 [Integridad añadida] Es sabido que los esquemas de cifrado autenticado ofrecen garantías de seguridad de privacidad superiores a las que brindan los mecanismos que solo cifran. Como consecuencia, aunque todos los modos que solo cifran, considerados anteriormente, están recomendados para la construcción de modos, su uso independiente se considera heredado. Esta es la razón de que aparezca la expresión  $R^*$  en la Tabla 2.4.

81.

Nota 5 [Modo en flujo] Algunos esquemas simétricos operan enmascarando el texto claro mediante un flujo de claves generado a partir de la propia clave y del IV. Estos modos de funcionamiento se llaman en flujo y para ellos es muy importante verificar que nunca se superponen dos flujos de claves. Esta propiedad se puede lograr de forma determinista en algunos modos, por ejemplo, en el CTR. En otros modos, como el OFB, la probabilidad de que tal propiedad se cumpla es casi 1, si no se utiliza el mismo par IV-clave (o solo con una probabilidad insignificante) para cifrar dos mensajes con la misma clave.

- Nota 6 [**Padding (Relleno)**] Algunos modos de cifrado no son capaces de tratar el cifrado del último bloque si este está incompleto, es decir, si tiene menos bits que los obligados para cada bloque. En estos casos, se debe llevar a cabo una modificación de tal bloque para que la operación pueda ejecutarse con normalidad; operación que suele denominarse padding. Uno de los procedimientos más utilizados consiste en completar el bloque claro con algunos datos estructurados de modo que se obtenga el tamaño requerido o que lo conviertan en un texto claro con una longitud que sea un múltiplo del tamaño del bloque. Sin embargo, la verificación del formato del padding durante el descifrado puede filtrar información sobre el valor descifrado de forma que podría usarse para descifrar cualquier bloque cifrado. Los desarrolladores y evaluadores deben asegurarse de que la implementación del descifrado no proporcione al atacante ningún tipo de información sobre el padding o su formato. Una alternativa a la aplicación del esquema de padding es el uso del robo de texto cifrado.
- 82.

## 2.2.2. CIFRADO DE DISCO DURO

83. Como ya hemos mencionado en §2.2.1, los modos de cifrado de propósito general permiten cifrar y descifrar datos atendiendo a diferentes características y según diferentes niveles de seguridad. Además, para lograr una seguridad en un sentido estricto, en determinadas ocasiones hace falta expandir los datos de algunos bloques debido al uso de un vector de inicialización o *nonce*. Sin embargo, en algunos entornos, estas modificaciones resultan ser un inconveniente. Este es el caso, por ejemplo, del cifrado del disco duro [71, §4.10]. En estas situaciones se permite el uso de modos de cifrado deterministas, como se muestra a continuación, si bien deben tenerse en cuenta las notas que se incluyen en cada caso.
84. El modo de cifrado en bloque modificable XEX<sup>5</sup> con robo de texto cifrado o XTS (*XEX Tweakable Block Cipher with Ciphertext Stealing*) es un modo de uso de confidencialidad de cifradores en bloque que se utiliza para el cifrado de medios de almacenamiento de datos, sin aportar autenticación. Este modo está especialmente diseñado para cifrar los datos de un disco duro en el que el índice del sector del disco duro forma parte del proceso de cifrado. De esta manera la información queda protegida por dos parámetros: su ubicación y la clave utilizada, de modo que si un adversario cambia de disco o el sector donde se almacena la información, no podrá recuperarla aunque conozca la clave.
85. En la Figura 5 se presenta el esquema de bloques XTS para el caso del AES (XTS-AES), cuando el último bloque que hay que cifrar es un bloque completo de 128 bits. Puede observarse que AES se emplea dos veces, con dos claves distintas

<sup>5</sup>XEX significa XOR *Encrypt* XOR

$K_1$  y  $K_2$ . El parámetro  $i$  corresponde al índice del sector que se está cifrando (en este caso tiene 128 bits); cada sector tiene su retoque (*tweak*), que consiste en un entero positivo de 128 bits, que se va incrementando a medida que se cifra. El parámetro  $j$  es un contador que señala el número de bloque dentro del sector y  $\alpha$  es un elemento primitivo del cuerpo finito  $\mathbb{F}_{2^{128}} \approx GF(2^{128})$ .

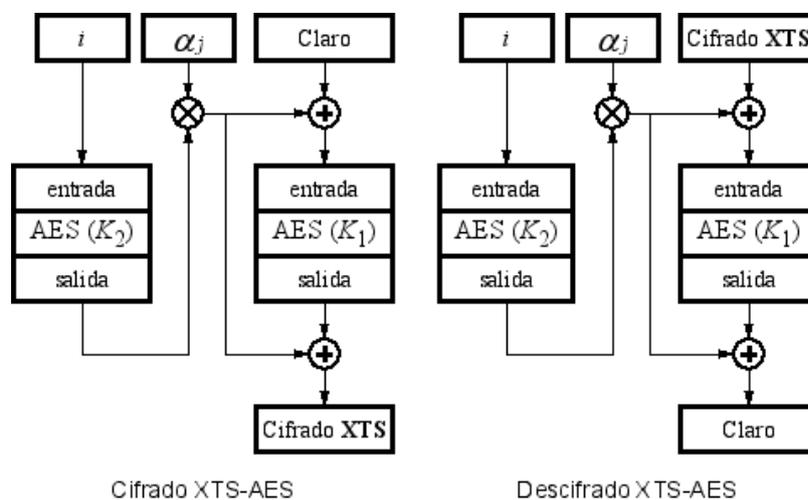


Figura 5: Modo XTS-AES para cifrado y descifrado

86. En la Tabla 2.5 se muestran los diferentes modos de esquemas de cifrado autorizados para discos duros, con sus principales características.

Esquema	Rec./Her.	Referencias	Notas
XTS	R	[158]	7, 8, 9

Tabla 2.5: Modos autorizados de cifrado simétrico para discos duros

87. Nota 7 [Modo de cifrado en flujo de disco] Los modos de cifrado de disco son, por naturaleza, modos de cifrado deterministas, donde el IV se deriva de la ubicación donde se almacenan los datos cifrados. Por ello, los modos de operación de cifrado en flujo son inapropiados, dado que los textos cifrados correspondientes a dos textos claros diferentes almacenados en la misma ubicación filtrarían la diferencia entre los textos en claro.

88. Nota 8 [Retoque único] El valor de retoque que se utilice para cifrar cada posición de bloque (completo o incompleto) en cada sector de disco deber ser único, es decir, un disco o conjunto de discos cifrados con la misma clave nunca deberá contener dos bloques distintos cifrados con la misma clave y el mismo valor de retoque.

89. Nota 9 [Retoque de dirección] Por lo general, el retoque se deriva de la dirección donde se almacena el texto cifrado. Para los dispositivos de almacenamiento o controladores de disco que determinan el retoque utilizado a partir de direcciones lógicas en lugar de direcciones físicas, la unicidad del retoque no está garantizada a priori y se debe tener especial cuidado para evaluar la conformidad con la Nota 8.

### 2.2.3. CÓDIGOS DE AUTENTICACIÓN DE MENSAJES

90. Los códigos de autenticación de mensajes o MAC (*Message Authentication Codes*) ofrecen autenticación de la integridad y el origen de datos basada en un bloque de bits determinado mediante un mecanismo simétrico, en general un cifrado en bloque o una función resumen [71, §4.7]. Un MAC consta de dos funciones, una que genera el código y otra que lo verifica. La primera tiene como entradas una clave secreta y un mensaje y proporciona como salida el código. La segunda tiene como entrada la misma clave secreta, el mismo mensaje y el código, siendo su salida un elemento del conjunto  $\{Verdadero, Falso\}$ .
91. Para ahorrar ancho de banda en las comunicaciones, es costumbre truncar el resultado de un esquema MAC, pero a la vez, para que el esquema sea resistente a los ataques de suposición (*guessing*), en los que un adversario intenta falsificar el MAC mediante un valor aleatorio, la longitud final del MAC no debe ser demasiado corta.
92. El MAC basado en cifrado o CMAC (*Cipher-based MAC*) utiliza dos subclaves que se derivan de la clave secreta mediante un algoritmo definido en [154]. El uso de una u otra subclave depende de que el mensaje tenga una longitud exactamente igual a la del bloque considerado, o no.
93. El MAC basado en funciones resumen se denota por HMAC, como ya se ha indicado. El HMAC más utilizado es el propuesto por la norma el ANSI 9.71 (*American National Standards Institute*) [10] y el NIST (*National Institute for Standards and Technology*) [157], que usa una clave de 512 bits,  $k$ . Si  $M$  es el mensaje que va

a ser autenticado,  $\mathfrak{h}$  la función resumen,  $\oplus$  el operador exclusivo (XOR) y  $opad$  e  $ipad$  son las siguientes constantes de *padding* de 512 bits:

$$opad = 5c5c5c \cdots 5c5c, \quad ipad = 363636 \cdots 3636,$$

entonces el operador  $\text{HMAC}_k(M)$  se define por

$$\text{HMAC}_k(M) = \mathfrak{h}((k \oplus opad) || \mathfrak{h}((k \oplus ipad) || 0M)).$$

La seguridad de este HMAC depende directamente de la seguridad de la función resumen que se utilice. En el caso de que tal función resumen sea la SHA-1, se tiene el HMAC-SHA-1.

94. Un modo de cifrado no mencionado anteriormente en §2.2.1 es el modo contador de Galois o GCM, (*Galois Counter Mode*). Este modo combina el modo contador (CTR) con un nuevo esquema de autenticación con operaciones en un cuerpo finito o de Galois [71, §4.9]. Una propiedad interesante de este modo es que se puede utilizar para autenticar un mensaje entero y cifrar solamente parte de él. Esto es muy práctico cuando el mensaje que hay que transmitir de forma secreta lleva cabeceras que no se pueden cifrar (direcciones IP, puertos, número de protocolo, etc.). El uso de este contador para cifrar y autenticar se denomina, como hemos dicho, GCM, pero si se utiliza solo para autenticar se conoce como MAC de Galois o GMAC (*Galois Message Authentication Code*).
95. SHA-3 y SHAKE (véase §2.2.7) son familias de funciones con muy buenas propiedades de seguridad, pero las posibilidades que ofrece Keccak son aún mayores, por lo que en la publicación especial del NIST [113] se consideran nuevas funciones. Entre ellas, cSHAKE y KMAC.
96. El algoritmo KMAC *Keccak Message Authentication Code* consta de una función pseudo-aleatoria y una función hash con clave basada en Keccak. KMAC proporciona una salida de longitud variable y, a diferencia de SHAKE y cSHAKE, la modificación de la longitud de salida genera una nueva salida no relacionada. KMAC tiene dos variantes: KMAC-128 y KMAC-256, construidas a partir de cSHAKE-128 y cSHAKE-256, respectivamente.
97. En la Tabla 2.6 se incluyen los códigos de autenticación de mensajes basados en cifradores en bloque y en funciones resumen que están autorizados.

Esquema	Tamaño clave (bits)	Rec./Her.	Referencias	Notas
CMAC		R	[154], [95]	10, 11
CBC-MAC		R	[95, Alg. 1, Relleno 2]	10, 11, 12
HMAC	$k \geq 128$ $k \geq 100$	R H	[118], [96]	
HMAC-SHA1	$k \geq 100$	H[2030]	[118], [96], [163]	13
GMAC		R	[155]	14, 15, 16
KMAC-128		R	[113]	
KMAC-256		R	[113]	

Tabla 2.6: MAC autorizados basados en cifradores en bloque y funciones resumen

98. Nota 10 **[MAC truncado a 96 bits]** Se acepta el truncamiento de un MAC a, al menos, 96 bits, generado por un mecanismo MAC autorizado. Esta condición necesaria no tiene por qué ser una condición suficiente para determinados esquemas MAC, como el GMAC (ver Nota 16).

99. Nota 11 **[MAC truncado a 64 bits]** El truncamiento de un MAC a, al menos, 64 bits, generado por un mecanismo de autorizado se considera heredado siempre que el número máximo de verificaciones del MAC realizadas para una clave determinada durante su vida útil pueda limitarse a  $2^{20}$ .

100. Nota 12 **[Longitud de entrada fija]** El CBC-MAC solo se emplea en entornos en los que los tamaños de todas las entradas para las que se calcula el MAC con la misma clave son los mismos. Si se permiten entradas de longitud variable, es posible llevar a cabo falsificaciones de extensión de la longitud.

101. Nota 13 [**HMAC-SHA-1**] HMAC no requiere la resistencia a colisiones de la función hash subyacente y, por ahora, HMAC-SHA-1 se considera un mecanismo heredado aceptable, aunque la función resumen SHA-1 no lo sea. En todo caso, se recomienda eliminar gradualmente el HMAC-SHA-1.

102. Nota 14 [**GMAC-GCM con *nonce***] El IV debe administrarse dentro de las condiciones de seguridad del proceso de cifrado autenticado. Por ejemplo, es crucial asegurarse de que ningún adversario pueda hacer que el mismo IV sea reutilizado para proteger diferentes pares de (texto claro, datos asociados) con la misma clave.

103. Nota 15 [**Opciones para GMAC-GCM**] Solo están autorizadas las siguientes opciones para GCM: la longitud del IV debe ser igual a 96 bits, se debe utilizar el método de construcción determinista para IV dado en [155, §8.2.1], la longitud del MAC debe ser 128 bits.

104. Nota 16 [**Límites para GMAC-GCM**] Como los límites de imposibilidad de falsificación de las opciones autorizadas de GMAC y GCM de la Nota 15 no son óptimos, la longitud del MAC debe ser de, al menos, 128 bits. Por lo tanto, no se debe realizar ningún truncamiento a una longitud MAC final, como 96 bits.

#### 2.2.4. ESQUEMAS SIMÉTRICOS DE AUTENTICACIÓN DE ENTIDADES

105. Los esquemas de autenticación de entidades permiten que una entidad pruebe su identidad ante un verificador demostrando que conoce determinado secreto. Por su estructura, son esquemas interactivos y, en general, utilizan un esquema MAC o un esquema de cifrado con un protocolo de desafío-respuesta aleatorio. Para este tipo de esquemas de autenticación no se proporciona ninguna tabla de esquemas autorizados en concreto, dado que son esquemas con diferentes objetivos de seguridad a los de los MAC aunque puedan basarse en ellos. Así, los modos de integridad y los esquemas de autenticación de entidad simétricos no deben utilizar la misma clave (véase la Nota 89). Una condición necesaria para que se acuerde un esquema basado en un

esquema de cifrado (resp. MAC) es que el cifrado en bloque subyacente (resp. MAC) esté autorizado.

106. Sin embargo, en la Tabla 2.7 sí se muestran los tamaños de los protocolos de desafío-respuesta autorizados.

Tamaño del protocolo desafío-respuesta (bits)	Rec./Her.	Notas
$128 \leq l$	R	17
$96 \leq l < 128$	H	17

Tabla 2.7: Tamaño de los protocolos de desafío-respuesta autorizados

107. Nota 17 [Protocolo desafío-respuesta] El verificador debe asegurarse de que un desafío no puede repetirse, con una probabilidad no despreciable. El desafío podría, por ejemplo, implementarse mediante un valor aleatorio generado por el verificador, cuyo tamaño sea lo suficientemente grande.

### 2.2.5. CIFRADO AUTENTICADO

108. Los esquemas de cifrado autenticado o AE (*Authenticated Encryption*) permiten obtener confidencialidad, integridad y autenticación del origen de los datos o de los mensajes. Un esquema AE consta de una función de cifrado que transforma un texto claro en un texto cifrado mediante una clave determinada, y una función de descifrado que recupera el texto claro a partir del texto cifrado y la clave. La diferencia con los esquemas que solo cifran es que la función de descifrado del esquema AE puede no devolver un valor descifrado si el texto cifrado no respeta algún patrón de redundancia. Debe tenerse en cuenta que, por lo general, alguna parte del texto cifrado actúa como un valor de verificación que se comprueba durante el descifrado.

109. En muchas ocasiones, los AE incorporan una característica adicional que consiste en combinar la autenticación de los datos cifrados con la autenticación de datos adicionales no cifrados. El AE con esa propiedad se conoce como cifrado autenticado con datos asociados o AEAD (*Authenticated Encryption with Associated Data*). Ambos tipos de esquemas se pueden obtener a partir de la combinación de un esquema de cifrado y un MAC.

110. El modo contador con encadenamiento de bloques de cifrado-MAC o CCM (*Counter with Cipher Block Chaining-Message Authentication Code*) garantiza la confidencialidad y autenticidad de los datos y se basa en un algoritmo de cifrado en

bloque de claves simétricas recomendado cuyo tamaño de bloque es de 128 bits (y no menos) [71, §4.8]. Este modo usa conjuntamente los modos CBC-MAC y CTR y ambos se aplican al mismo mensaje; el primero para autenticar el mensaje mediante un MAC y el segundo para cifrarlo. En los dos procesos se usa la misma clave que es preciso establecer previamente entre las partes.

111. El modo cifrar-luego-autenticar-luego-traducir o EAX (*Encrypt-then-Authenticate-then-Translate*) es un modo de AEAD que proporciona autenticación y privacidad del mensaje con un esquema de dos pasadas, una para lograr privacidad y otra para autenticidad para cada bloque.
112. ChaCha20\_Poly1305 [150] es también un modo de AEAD que está compuesto por el cifrador ChaCha20 y el autenticador Poly1305. ChaCha20 es un cifrador de alta velocidad propuesto por Bernstein en [28] y es considerablemente más rápido que AES en implementaciones solo de software. Ello le permite ser tres veces más rápido en plataformas que carecen de hardware AES especializado. Por su parte, Poly1305 es un código de autenticación de mensajes de alta velocidad, también presentado por Bernstein [30].
113. La Tabla 2.8 presenta los cifrados autenticados autorizados con sus principales propiedades.

Esquema	Rec./Her.	Referencias	Notas
Cifrado-y luego-MAC	R	[23]	18, 19
CCM	R	[156], [102]	18, 19
GCM	R	[155], [102]	14, 15, 16, 18, 19, 20
EAX	R	[102]	18, 19
ChaCha20_Poly1305	R	[150], [28], [30]	21, 22

**Tabla 2.8: Esquemas simétricos de cifrado autenticado autorizados**

114. Nota 18 [Esquemas de cifrado autenticado] Debe tenerse en cuenta que los esquemas de cifrado autenticado simétrico autorizados son todas construcciones criptográficas. Algunos de ellos usan como primitiva un único cifrado en bloque, como CCM o GCM; otros se basan en un esquema de cifrado primitivo y en un esquema de autenticación de mensajes primitivo, que describen solo cómo están compuestos; por ejemplo, Cifrado-y luego-MAC. En tales casos, las primitivas deben estar autorizadas (ver Nota 2). Es importante señalar que las Notas 10 y 11 también se aplican a esquemas de cifrado autenticado simétrico.

115. Nota 19 [**Orden de descifrado.**] Si la integridad de los textos cifrados no se comprueba correctamente antes del descifrado, el desarrollador o evaluador debe asegurarse de que la implementación no cree ninguna instancia de padding u otro oráculo erróneo. Esto significa que la implementación no revela ninguna información sobre el padding o el formato del texto claro obtenido a través del descifrado de un texto cifrado arbitrario. Si no fuera así, un adversario podría descifrar un texto cifrado objetivo explotando, por ejemplo, mensajes de error o información derivada de un ataque por canal lateral por análisis temporal. Los textos claros nunca deben enviarse a las aplicaciones que los vayan a utilizar antes de que se haya verificado su integridad.

116. Nota 20 [**Longitud del texto claro GCM**] Por especificación, en cada invocación de GCM, la longitud del texto claro debe ser, como máximo, de  $2^{32} - 2$  bloques del cifrado de bloque subyacente. Es necesario verificar que no se exceda esta longitud máxima en entornos donde esto pudiera suceder.

117. Nota 21 [**Esquema ChaCha20\_Poly1305**] El esquema autorizado usa una variante del cifrador ChaCha20 con 20 rondas y una clave de 256 bits. Es sabido que existen variantes con claves de 128 bits y de entre 8 y 12 rondas, pero no esas otras variantes no están autorizadas.

118. Nota 22 [**No reutilización del vector de inicialización**] El vector de inicialización debe procesarse dentro del perímetro de seguridad del esquema de cifrado autenticado. Así, es esencial asegurarse de que un adversario nunca pueda hacer que se use el mismo valor IV para proteger dos pares diferentes de mensaje y datos asociados con la misma clave. La reutilización de un IV puede afectar la confidencialidad.

## 2.2.6. PROTECCIÓN DE LAS CLAVES

119. Un mecanismo de protección de claves o de envoltura de claves, permite almacenar o transmitir claves de forma segura, lo que garantiza confidencialidad, integridad y autenticación de los datos de origen de la clave.

120. Un vector de inicialización sintético o SIV (*Synthetic Initialization Vector*) es un modo de operación de cifrado en bloque que considera una clave, un texto claro y múltiples cadenas de octetos de longitud variable que se autenticarán pero no se cifrarán. SIV produce un texto cifrado que tiene la misma longitud que el texto claro y un vector de inicialización sintético. Dependiendo de cómo se utilice, SIV logra el objetivo de cifrado autenticado determinista o el objetivo del cifrado autenticado resistente al uso indebido y basado en *nonce*.
121. AES-Keywrap permite la protección de la confidencialidad y la integridad de las claves criptográficas. En particular, KW (*Key Wrap*) y KWP (*Key Wrap with Padding*) son dos modos de operación deterministas de cifrado autenticado del algoritmo AES.
122. La Tabla 2.9 presenta los esquemas de protección de claves autorizados y sus principales propiedades.

Esquema	Rec./Her.	Referencias	Notas
SIV	R	[83]	
AES-Keywrap	R	[160, Alg. KW y KWP]	

Tabla 2.9: Esquemas de protección de claves autorizados

### 2.2.7. FUNCIONES DE DERIVACIÓN DE CLAVES

123. Una función de derivación de claves o KDF (*Key Derivation Function*) permite obtener varias claves a partir de una única clave maestra. En general, la función considera como entrada tres argumentos: un valor secreto  $K$ , un valor (posiblemente) público  $N$  y una longitud  $n$ , y genera  $n$  bits, que pueden dividirse en varias claves que parecen ser independientes. Hay muchas buenas formas para implementar estas funciones. La lista de mecanismos de derivación de claves autorizados se proporciona en la Tabla 2.10. En general, una KDF se considera autorizada si los mecanismos criptográficos que emplea lo están.
124. Los dos primeros grupos de funciones de derivación de claves están estandarizadas por el NIST en los documentos 56A, 56B, 56C y 108 de la familia SP800. Estas KDF se basan en los problemas del logaritmo discreto, de la factorización de enteros y de una extracción seguida de una expansión.
125. La función X9.63-KDF [12] está estandarizada por ANSI y se fundamenta en funciones resumen.
126. La función 2 de derivación de clave basada en contraseña o PBKDF2 (*Password-Based Key Derivation Function 2*) es una KDF con un costo computacional variable, que se utiliza para reducir las vulnerabilidades de los ataques de fuerza bruta [159]. Esta función forma parte de los estándares PKCS de los

laboratorios RSA, específicamente PKCS #5 v2.0 [106], [142]. No debe confundirse esta PBKDF2 con la PBKDF1, dado que la última solo da lugar a claves derivadas de hasta 160 bits. Para la PBKDF2 se añade una nota específica que los desarrolladores y evaluadores deben tener en cuenta.

127. SCRYPT, por su parte, es una KDF basada en contraseñas propuesta en [174] para el servicio de copias de seguridad en línea de Tarsnap. Esta KDF se diseñó para que fuera difícil realizar ataques de hardware personalizados a gran escala, dado que requiere grand cantidad de memoria. De hecho, algunas criptomonedas utilizan una versión simplificada de SCRYPT como esquema de prueba de trabajo.
128. La función HKDF es una función de derivación de clave simple (KDF) basada en el código de autenticación de mensajes HMAC [119]. HKDF sigue el protocolo de extraer y luego expandir, donde el KDF consta de dos etapas. En la primera considera la entrada y extrae una clave pseudoaleatoria de longitud fija y en la segunda la expande en varias claves pseudoaleatorias adicionales, que es su salida.

Esquema	Rec./Her.	Referencias	Notas
NIST SP800-56A/B/C	R	[166], [167], [169]	
NIST SP800-108	R	[172]	
ANSI-X9.63-KDF	R	[12]	
PBKDF2	R	[142], [159]	23
SCRYPT	R	[175]	
HKDF	R	[119]	

Tabla 2.10: Funciones de derivación de claves autorizadas

- Nota 23 [PBKDF2] Esta función de derivación de claves se basa en una función pseudoaleatoria, que se puede concretar usando una función de generación de MAC. La función pseudoaleatoria deberá ser un mecanismo autorizado. En el caso de que se utilice una función HMAC, se debe vigilar la longitud de la clave. De hecho, si la clave HMAC es más larga que la longitud del bloque de mensajes de la función resumen, la clave es resumida. Este prerresumen en HMAC puede reducir la entropía efectiva de la clave derivada.
- 129.

### 2.2.8. MECANISMOS DE PROTECCIÓN/RESUMEN DE CONTRASEÑAS

130. Los mecanismos de resumen de contraseñas asocian a una contraseña un valor de verificación, de forma que los sistemas de autenticación de usuarios que se basan en

contraseñas no almacenan las propias contraseñas, sino estos valores de verificación. De este modo, las contraseñas nunca están disponibles (no se almacenan en claro por lo que no estarán comprometidas), pero es posible verificar que pertenecen a determinado usuarios, comprobando su correspondiente valor de verificación.

131. En la Tabla 2.11 se incluyen las características del mecanismo de resumen de contraseñas autorizado.

Esquema	Rec./Her.	Referencias	Notas
PBKDF2	R	[149]	24, 25

Tabla 2.11: Mecanismos de resumen de contraseñas autorizados

132.

Nota 24 **[Número de iteraciones]** Los mecanismos de resumen de contraseñas permiten seleccionar parámetros que controlan la complejidad de la ejecución del resumen. De esta manera se incrementa el trabajo para realizar posibles ataques de fuerza bruta. De hecho, un uso legítimo del resumen de una contraseña solo necesita una ejecución, mientras que un ataque de fuerza bruta precisa una gran cantidad de ejecuciones. Así pues, para proteger este mecanismo, el número de iteraciones de PBKDF2 debe seleccionarse lo más grande posible.

133.

Nota 25 **[Salar (*Salt*)]** Este proceso consiste en generar un valor aleatorio (*salt*) cuando se registra una contraseña y que se almacena junto con el valor de verificación de dicha contraseña. De este modo, salar (to salt) un mecanismo de resumen de contraseñas permite contrarrestar los ataques por precálculo. La longitud del valor generado debe ser de, al menos, 128 bits.

### 3. MECANISMOS ASIMÉTRICOS

134. Los mecanismos asimétricos son aquellos, como ya se introdujo en la Sección 1, en los que las acciones que llevan a cabo o los parámetros que utiliza cada una de las partes que intervienen son diferentes, esto es, existe una asimetría entre los parámetros y acciones de cada parte. En esta sección se abordarán los mecanismos que corresponden a esta categoría.

#### 3.1. PRIMITIVAS ASIMÉTRICAS (§10.3)

135. La criptografía asimétrica (a veces también llamada de clave pública) tiene como propiedad distintiva que cada usuario emplea dos claves, una para el proceso de cifrado y otra diferente para el de descifrado. La primera de las claves es la «clave pública» que cada usuario da a conocer para que sea utilizada como clave para cifrar los mensajes que se le envíen; mientras que la otra es la «clave privada» (o secreta), que solo conoce dicho usuario y le permite descifrar los mensajes cifrados que recibe.

136. Ambas claves están relacionadas mediante un problema matemático y dado que ambas llevan a cabo procesos inversos (una cifra y la otra descifra), tal problema se elige de modo que el primero de los procesos (cifrado) suponga resolver un problema matemático sencillo, a la vez que el segundo proceso (descifrado) equivalga a determinar la solución de un problema matemático computacionalmente imposible de resolver en un tiempo razonable. Es claro que, como las claves están relacionadas, el problema matemático seleccionado debe garantizar que el conocimiento de la clave pública no permite recuperar la clave privada. Así pues, la seguridad de la criptografía asimétrica se basa en la supuesta dificultad de resolver computacionalmente un problema matemático.

137. Debido a que los mecanismos asimétricos hacen uso de la clave privada de un usuario para, principalmente, descifrar un texto cifrado (esquemas de cifrado) o elaborar una firma válida (esquemas de firma electrónica), no debería ser posible realizar ninguna operación que requiera la clave privada con el conocimiento exclusivo de la clave pública. Esta propiedad debe mantenerse incluso en el caso de que un adversario reciba resultados de operaciones privadas que requieren la clave privada, donde los recursos de estas operaciones privadas son conocidos o elegidos por el adversario.

138. Comenzamos presentando en esta sección las primitivas atómicas asimétricas que están aceptadas y los problemas matemáticos correspondientes. En la siguiente sección se mostrarán las construcciones asimétricas construidas a partir de estas primitivas. Debe tenerse en cuenta que solo se consideran aceptadas las primitivas o construcciones cuyos parámetros satisfagan las condiciones que se muestren en las diferentes tablas que se incluyen.

### 3.1.1. PROBLEMA DE LA FACTORIZACIÓN DE NÚMEROS ENTEROS (RSA)

139. La seguridad de varios esquemas criptográficos asimétricos, incluidos los esquemas de cifrado y firma electrónica basados en el RSA, se basa en la dificultad que supone resolver el problema de la factorización de números enteros (IFP) en comparación con el problema, que puede considerarse su inverso, de la multiplicación de enteros y con el problema de la exponenciación modular. Para los dos últimos existen algoritmos cuyo tiempo es polinómico; mientras que para el primero, el mejor algoritmo conocido requiere de tiempo subexponencial.
140. La primitiva RSA puede considerarse como permutación pública parametrizada por una clave pública y la permutación inversa privada parametrizada por la clave privada asociada. Esta primitiva se utiliza en los esquemas de cifrado y firma de RSA. Tales esquemas especifican, por su parte, aspectos relacionados con la verificación de relleno (*padding*), redundancia, etc. Recordamos que las primitivas por sí solas no debe considerarse como esquemas completos de cifrado o firma, dado que precisan convenciones adicionales para garantizar su seguridad.
141. Sean  $p$  y  $q$  dos números primos grandes, elegidos al azar, y sea su producto  $N = p \cdot q$ , que se denotará como el «módulo RSA». El valor  $n = \lfloor \log_2(N) \rfloor + 1$  es el tamaño de  $N$ , esto es, su longitud en binario. La clave pública es el par formado por el módulo  $N$  junto con un elemento  $e$ , llamado «exponente público», que es un invertible módulo  $\phi = (p - 1)(q - 1)$ , es decir, es primo con  $\phi$ , o lo que es igual,  $\text{mcd}(e, \phi) = 1$ . El inverso de  $e$  módulo  $\text{mcm}(p - 1, q - 1)$ , denotado por  $d$ , se llama «exponente privado». La clave privada está formada por este exponente privado junto con el módulo.
142. La permutación pública mencionada en el párrafo anterior opera sobre los números enteros módulo  $N$ ,  $\mathbb{Z}_N$ , y consiste en la exponenciación de la entrada considerada elevada a la potencia  $e$ , módulo  $N$ . Recuérdese que tanto  $N$  como  $e$  son públicos, por lo que cualquiera que los conozca puede determinar, fácilmente, el valor resultante de la expresión

$$M^e \pmod{N}, \quad \text{donde } M \in \mathbb{Z}_N.$$

Por otra parte, la permutación privada opera sobre el conjunto de los números enteros módulo  $N$ ,  $\mathbb{Z}_N$ , y consiste en la exponenciación de la entrada elevada a la potencia  $d$ , módulo  $N$ , esto es,

$$C^d \pmod{N}, \quad \text{siendo } C \in \mathbb{Z}_N.$$

En este caso, como el valor de  $d$  solo es conocido por el propietario de la clave privada, nadie, salvo él, puede determinar el valor de la expresión anterior.

143. En la Tabla 3.1 se muestran los tamaños de las claves autorizadas para la primitiva basada en el problema de la factorización de números enteros (RSA) y otras características.

Primitiva	Tamaño de los parámetros (bits)	Rec./Her.	Referencias	Notas
RSA	$n \geq 3000, \log_2(e) > 16$ $n \geq 1900, \log_2(e) > 16$	R H[2025]	[184]	26

Tabla 3.1: Tamaño de las Primitivas RSA autorizadas

144. Nota 26 [RSA Heredado] La fecha límite aceptable para el uso heredado de un módulo de tamaño superior a 1900 bits, pero inferior a 3000 bits, se establece en el 31 de diciembre de 2025.

### 3.1.2. PROBLEMA DEL LOGARITMO DISCRETO MULTIPLICATIVO

145. De forma análoga a como sucede con el problema de la factorización de números enteros y de su problema inverso, la multiplicación de enteros (ver §3.1.1), la seguridad de otros esquemas criptográficos asimétricos se basa en la dificultad de resolver el problema del logaritmo discreto (DLP) en el grupo multiplicativo de un cuerpo finito, en contraposición con la facilidad del problema de la exponenciación modular (MODP), también en un cuerpo finito.
146. Es posible elegir diferentes cuerpos finitos en los que implementar este problema, pero la solución más segura y ampliamente utilizada es considerar un cuerpo finito primo,  $\mathbb{F}_p \approx GF(p)$ , siendo  $p$  un número primo grande. En esta guía, y mientras no se diga lo contrario, se supondrá que el cuerpo considerado es de este tipo.
147. Esta primitiva cuya seguridad se basa en el problema del logaritmo discreto en el grupo multiplicativo de  $\mathbb{F}_p$  se utiliza en varios esquemas de acuerdo o intercambio de claves, firmas y cifrado (sobre todo de tipo híbrido). Sea  $g$  un generador de un subgrupo de orden  $q$  del grupo multiplicativo  $\mathbb{F}_p^*$ , donde  $q$  divide a  $p - 1$  dado que  $p - 1 = |\mathbb{F}_p^*|$ , y sea  $r$  el factor primo más grande de  $q$ . La primitiva es la función de exponenciación de base  $g$  en  $\mathbb{F}_p$  de modo que si se considera como entrada el entero  $x$ , en general con  $1 \leq x \leq q - 1$ , proporciona como salida el entero  $y = g^x$ . Según como el esquema considerado utilice esta primitiva,  $x$  e  $y$  pueden representar (una parte de) una clave privada y la clave pública asociada, o pueden representar un exponente efímero de Diffie-Hellman y su valor público asociado, etc.
148. La Tabla 3.2 presenta los tamaños de las claves autorizadas para la primitiva basada en el problema del logaritmo discreto multiplicativo sobre un cuerpo finito primo, cuya primitiva asociada se abrevia como FF-DLOG (*Finite Field-Discrete Logarithm*) y sus características. Por otra parte, para todos los grupos considerados, se tiene que  $r = q = (p - 1)/2$ .

Familia	Tamaño del grupo (bits)	Rec./Her.	Referencias	Notas
MODP	3072 bits	R	[116]	27, 28
	4096 bits	R		
	6144 bits	R		
	8192 bits	R		
	2048 bits	H[2025]		

**Tabla 3.2: Tamaño de las primitivas autorizadas del logaritmo discreto multiplicativo sobre un cuerpo finito**

149.

Nota 27 **[Precomputación]** Los algoritmos de logaritmos discretos implican una fase de precomputación relacionada con el grupo, que es el cuello de botella en términos de complejidad del ataque. Como consecuencia, para los módulos del logaritmo discreto compartidos por muchos usuarios y aplicaciones, se recomienda encarecidamente no utilizar módulos de longitud cercana al límite inferior del rango heredado.

150.

Nota 28 **[DLP Heredado]** La fecha límite de aceptación para el uso heredado de grupos multiplicativos módulo un primo de tamaño superior a 1900 bits, pero inferior a 3000 bits, se establece en el 31 de diciembre de 2025.

### 3.1.3. PROBLEMA DEL LOGARITMO DISCRETO ADITIVO

151. La dificultad del problema del logaritmo discreto también se puede definir en el grupo de puntos racionales de una curva elíptica definida sobre un cuerpo finito. En este caso, el problema se denomina problema del logaritmo discreto sobre curvas elípticas, elíptico, aditivo o ECDLP (*Elliptic Curve Discrete Logarithm Problem*), en contraposición al caso anterior, en el que la operación considerada era la multiplicación en un grupo finito. En este caso, la primitiva se conoce como logaritmo discreto sobre curvas elípticas y de forma abreviada como EC-DLOG (*Elliptic Curve Discrete Logarithm*).
152. Para comprender bien la primitiva asociada a este problema, conviene considerar la siguiente notación y recordar las siguientes propiedades.
153. Sea  $p$  un número primo y  $\mathbb{F}_p$  el cuerpo primo con  $p$  elementos. Sea, además,  $E(\mathbb{F}_p)$  una curva elíptica definida sobre  $\mathbb{F}_p$ , denotada por  $E$ . Dicha curva está definida por

una ecuación general del tipo<sup>6</sup>

$$E: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6,$$

donde  $a_1, a_2, a_3, a_4, a_6 \in \mathbb{F}_p$ .

154. La curva  $E$  está formada por los puntos del plano  $\mathbb{F}_p \times \mathbb{F}_p$  que verifican dicha ecuación y sobre este conjunto se define una operación de suma de puntos que, junto con el punto del infinito,  $\mathcal{O}$ , hace que  $E$  sea un grupo abeliano (para un estudio más exhaustivo de la criptografía basada en curvas elípticas, recomendamos al lector [72]).
155. Sea  $P$  un punto de orden  $q$  de la curva  $E$ , esto es, tal que  $qP = P + q \dots - 2 + P = \mathcal{O}$ . Sea además,  $r$  el factor primo más grande de  $q$ . La primitiva asociada al problema del logaritmo discreto en  $E$  es la multiplicación de puntos de la curva por escalares, de modo que si se considera como entrada el entero  $x$ , con  $1 \leq x \leq q - 1$ , la salida es el punto de la curva  $E$  dado por  $Q = xP$ .
156. El orden de la curva, esto es, el número de puntos de la curva,  $\#E$ , puede ser un número primo o un número compuesto. Al cociente del orden de la curva entre su mayor factor primo se le denomina cofactor, de manera que si  $G$  es un punto de orden  $n$  que genera un subgrupo cíclico de orden primo, y no existe otro factor primo de  $\#E$  mayor que  $n$ , entonces se cumplirá que el cofactor,  $h$ , de la curva será  $h = \#E/n$ . En general se recomienda utilizar curvas cuyo orden sea un número primo o que, como segunda opción, sea el producto de un número primo y un cofactor pequeño (típicamente 2, 3 o 4) [82].
157. El conjunto de parámetros públicos de los esquemas criptográficos donde se usa esta primitiva es  $\{p, E, P, q\}$ . En función del esquema criptográfico que se considere,  $x$  y  $Q$  pueden representar (una parte de) una clave privada y la clave pública asociada, o pueden representar un valor secreto efímero de Diffie-Hellman y su valor público asociado, etc.
158. En estos grupos, el problema del logaritmo discreto también se considera difícil, en comparación con su operación inversa que, como hemos visto, es la multiplicación de puntos de la curva por escalares. Por otra parte, en este caso es posible seleccionar dos parámetros: el cuerpo finito sobre el cual se definirá la curva elíptica y la propia curva elíptica. También, como en el caso del logaritmo discreto multiplicativo, solo se consideran curvas elípticas definidas sobre cuerpos primos.
159. En la literatura se han propuesto numerosas curvas elípticas, buscando, sobre todo, la sencillez de sus ecuaciones, y cuerpos finitos en los que su cardinal, es decir, el primo considerado, tenga una expresión binaria en la que abunden los ceros, de modo que la implementación de la aritmética de la curva sea eficiente.

<sup>6</sup>Según sea el cuerpo considerado y su característica, la expresión de la curva elíptica puede tomar expresiones más sencillas a la dada, de modo que los coeficientes deben verificar determinadas condiciones.

160. En la Tabla 3.3 se muestran las familias de curvas elípticas autorizadas para su implementación, así como el nombre correspondiente de cada una de ellas.

Familia de curvas	Curva	Rec./Her.	Referencias	Notas
Brainpool	BrainpoolP256r1	R	[125]	29, 30, 31
	BrainpoolP384r1	R		
	BrainpoolP512r1	R		
NIST	NIST P-256	R	[162]	29, 30, 31
	NIST P-384	R		
	NIST P-521	R		
FR	FRP256v1	R	[13]	29, 30, 31, 32
Bernstein	B1 (Curve25519)	R	[27]	29, 30, 31
Hamburg	B2 (Curve448)	R	[122]	29, 30, 31

Tabla 3.3: Curvas elípticas autorizadas

161. Nota 29 [**Puntos en la curva**] Es preciso verificar que los puntos considerados están en la curva, es decir, verifican su ecuación.

162. Nota 30 [**Puntos en un subgrupo**] Es preciso verificar que los puntos considerados están en el subgrupo considerado de la curva. Nótese que el uso de subgrupos de la curva tiene lugar cuando el orden de la misma no es un número primo.

163. Nota 31 [**Orden primo**] Si el orden del subgrupo es un número primo, esto es,  $q = r$ , y es tal que  $r^2$  no divide al cardinal de la curva,  $\#E(\mathbb{F}_p)$ , las comprobaciones mencionadas en la Nota 30 se reducen a verificar que los puntos considerados tienen orden precisamente  $r$ .

164. Nota 32 [**Elección del primo**] La especial forma del número primo  $p$  seleccionado para la construcción del cuerpo finito  $\mathbb{F}_p$  hace que los ataques por canal lateral sean más eficientes que con un primo aleatorio.

### 3.1.4. OTROS PROBLEMAS COMPUTACIONALMENTE DIFÍCILES

165. Los tres problemas matemáticos considerados anteriormente (factorización de números enteros, logaritmo discreto multiplicativo y logaritmo discreto aditivo) son los más estudiados y los más utilizados en la criptografía actual. Sin embargo, tal y como se comentará en el Capítulo 9, estos algoritmos podrían ser considerados vulnerables a la computación cuántica si se utilizara el algoritmo de Shor [195] y se dispusiera de un ordenador cuántico con la suficiente capacidad de cómputo.
166. Otro problema matemático de interés es el problema del logaritmo discreto en el grupo de puntos racionales del Jacobiano de una curva hiperelíptica de género 2 o 3 [144].
167. En este sentido es importante mencionar algunos problemas matemáticos que permiten definir construcciones asimétricas para los cuales no se conoce un método genérico de resolución eficiente en general, ni siquiera mediante el uso de ordenadores cuánticos.
168. Estos problemas serán más detallados en el Capítulo 9 y dan lugar a los siguientes tipos de criptografía:
- Basada en códigos correctores de errores.
  - Basadas en funciones resumen (*hash function*).
  - Multivariante cuadrática.
  - Basada en retículos.
  - Basada en isogenias entre curvas elípticas.
169. No obstante, conviene mencionar que estos «nuevos problemas» han sido mucho menos estudiados que los tres mencionados anteriormente por lo que solo las primitivas basadas en estos últimos son las autorizadas. Los restantes problemas mencionados están siendo escrutados por la convocatoria internacional hecha por el NIST [165] en busca de estándares que se pretende sean resistentes a la computación cuántica (*quantum resistant*).

### 3.2. CONSTRUCCIONES ASIMÉTRICAS (§10.4)

170. Determinados problemas matemáticos, computacionalmente difíciles<sup>7</sup>, suelen utilizarse para construir esquemas asimétricos.

---

<sup>7</sup>Se dice que un problema matemático es computacionalmente difícil si o bien no se conoce un algoritmo que lo resuelva, o si existiendo tal algoritmo, el tiempo de computación necesario para obtener la solución requiere de un tiempo (sub)exponencial, utilizando la mejor tecnología disponible y el algoritmo más eficiente.

171. En los esquemas asimétricos, cada usuario posee un par de claves. La primera de ellas es una clave públicamente conocida, denotada por  $p_k$ , y una segunda clave privada, esto es que se mantiene en secreto, designada como  $s_k$ . La seguridad de tal esquema debe basarse en la dificultad computacional de un problema matemático, esto es, que conocida la clave pública, determinar la clave privada asociada sea equivalente a resolver dicho problema matemático. Ello supone que no es posible descifrar los mensajes destinados a un usuario que se hayan cifrado con su clave pública, salvo, claro está, el propietario de la clave privada asociada a tal clave pública. De forma análoga, nadie podrá firmar digitalmente un fichero haciendo uso de la clave privada, aunque sí podrá verificar que dicha firma corresponde a tal usuario para lo que utilizarán su correspondiente clave pública.
172. Al margen de que la seguridad de los esquemas asimétricos se base en la seguridad computacional de un problema matemático, también es posible que tal seguridad dependa de la seguridad de un esquema o primitiva simétrico. En todo caso, para que una construcción criptográfica esté autorizada, debe basarse en primitivas también autorizadas. De forma más concreta, los requisitos sobre los tamaños de los parámetros establecidos en la sección anterior (ver §3.1) también se aplican a los esquemas de esta sección.
173. Al margen de lo ya dicho en los párrafos anteriores, es claro que la seguridad de los esquemas asimétricos con clave también requiere de la confidencialidad e integridad de la clave privada y de la integridad y autenticidad del origen de datos de la clave pública.
174. Para cada esquema asimétrico con clave considerado en esta sección, los aspectos específicos de la generación de pares de claves se tratan en la misma subsección que el resto del esquema.
175. En el capítulo 6 se tratará el tema de la gestión de las claves y de los mecanismos para la generación de pares de claves asimétricas autorizadas.

### 3.2.1. ESQUEMAS DE CIFRADO ASIMÉTRICO

176. Un esquema de cifrado (o criptosistema) asimétrico consta de tres protocolos. El de generación del par de claves; el protocolo de cifrado por el que un mensaje o texto en claro dado,  $m$ , se transforma en un texto cifrado o criptograma mediante la clave pública,  $p_k$ ; y el protocolo de descifrado, que permite recuperar el texto claro,  $m$ , a partir del texto cifrado,  $c$ , mediante la clave privada,  $s_k$ .
177. El *padding* de cifrado asimétrico óptimo u OAEP (*Optimal Asymmetric Encryption Padding*) es un esquema de relleno que permite completar la longitud necesaria (generalmente en bits) de la entrada a un determinado algoritmo. En general, se utiliza junto con el cifrado RSA. Este OAEP fue introducido en [26] y más tarde estandarizado en [185] y en [107].

178. El algoritmo OAEP es un tipo de red Feistel que usa dos oráculos aleatorios para procesar el texto claro antes de proceder a su cifrado. Cuando se combina con una permutación unidireccional con trampa, se puede demostrar que esta forma de proceder da como resultado, en el modelo de oráculo aleatorio, un esquema combinado que es semánticamente seguro bajo el ataque de texto claro elegido (*Indistinguishability under chosen-plaintext attack* o IND-CPA). Cuando se implementa con ciertas permutaciones de trampa (por ejemplo, con el RSA), el OAEP también es seguro contra el ataque de texto cifrado elegido. En este sentido, el OAEP añade un aspecto aleatorio que sirve para convertir un esquema de cifrado determinista (como el RSA) en un esquema probabilístico. Además, previene el descifrado parcial de textos cifrados de modo que un adversario es incapaz de recuperar ninguna parte del texto claro, salvo que pueda invertir la permutación unidireccional con trampa utilizada.
179. Los estándares de criptografía de clave pública PKCS son una colección de estándares (desde el PKCS#1 al PKCS#15) desarrollados y publicados por los laboratorios RSA (<https://www.rsa.com/en-us/index>, <https://web.archive.org/web/20061209135809/http://www.rsasecurity.com/rsalabs/node.asp?id=2124>).
180. En la Tabla 3.4 se incluyen las características del esquema de cifrado asimétrico RSA autorizado.

Primitiva	Esquema	Rec./Her.	Referencias	Notas
RSA	OAEP PKCS#1 v2.1	R	[141], [185]	33, 34
RSA	PKCS#1 v1.5	H	[141], [185]	33, 35

Tabla 3.4: Esquema de cifrado asimétrico autorizado

181. Nota 33 [**Relleno (*Padding*) aleatorio**] Los esquemas de cifrado asimétrico utilizan un *padding* aleatorio que será generado por un generador de bits aleatorios autorizado (ver el capítulo 5).

182. Nota 34 [**Ataque de relleno-OAEP**] En el caso de que el procedimiento de descifrado OAEP no se implemente de modo correcto, es decir, si las comprobaciones realizadas por la decodificación EME-OAEP no se realizan en el orden especificado, el RSA OAEP puede ser vulnerable a ataques de oráculo.

183. Nota 35 [Ataque de *padding*] Si hubiera un oráculo de *padding* disponible, el esquema RSA-PCKS #1v1.5 sería vulnerable a ataques eficientes.

### 3.2.2. ESQUEMAS DE CIFRADO HÍBRIDO

184. Los protocolos de cifrado híbrido son protocolos de cifrado seguros que utilizan las propiedades de los criptosistemas simétricos (§2.1) y de los asimétricos (§3.1) para hacer más efectivo el cifrado y descifrado de datos.

185. Así, el algoritmo de cifrado en origen se lleva a cabo según el Algoritmo 1.

---

**Algoritmo 1** Algoritmo de cifrado híbrido

---

1. Se cifra el mensaje,  $m$ , con una clave secreta (de sesión),  $k$ , correspondiente a un criptosistema de clave simétrica, y se obtiene el criptograma,  $c$ .
  2. Se cifra la clave secreta,  $k$ , con la clave pública del destinatario mediante un criptosistema de clave asimétrica, y se obtiene  $K$ .
  3. Se envía el criptograma,  $c$ , y la clave cifrada,  $K$ .
- 

186. El descifrado en destino sigue los pasos del Algoritmo 2.

---

**Algoritmo 2** Algoritmo de descifrado híbrido

---

1. Se descifra la clave secreta,  $k$ , haciendo uso del criptosistema de clave asimétrica a partir de  $K$  y de la clave privada del usuario.
  2. Se descifra el mensaje  $m$  mediante el criptosistema de clave simétrica haciendo uso de la clave secreta obtenida,  $k$ , y del criptograma,  $c$ .
- 

187. En el caso de utilizar este tipo de criptosistema híbrido, se utilizará como criptosistema simétrico cualquiera de los cifradores en bloque autorizados en la Tabla 2.1 y como asimétricos los cifradores autorizados en las Tablas 3.1, 3.2 y 3.3.

#### 188. Criptosistema ECIES

189. El esquema de cifrado integrado basado en curvas elípticas o ECIES (*Elliptic Curve Integrated Encryption Scheme*) es el esquema de cifrado híbrido basado en curvas elípticas más extendido en la actualidad [73, 129, 74, 72]. Todas sus versiones están incluidas en los estándares ANSI X9.63 [12], IEEE 1363a [89] y SECG SEC 1 [193], y se adaptan a un mismo modelo general, con la siguiente notación:

- Clave privada,  $u$ , y clave pública,  $U$ , temporales del usuario que envía el mensaje,  $\mathcal{U}$ .
- Clave privada,  $v$ , y clave pública,  $V$ , temporales del usuario que recibe el criptograma,  $\mathcal{V}$ .
- $KA$ : función de generación del secreto compartido o de acuerdo de clave (*Key Agreement*).
- $KDF$ : función de derivación de claves.
- $m$ : mensaje en claro a enviar.
- $c$ : mensaje cifrado con el criptosistema simétrico  $ENC$ .
- $tag$ : etiqueta de autenticación generada por el código de autenticación de mensajes  $MAC$ .
- $k_{ENC}$ : clave de cifrado simétrico a utilizar por la función de cifrado simétrico,  $ENC$ .
- $k_{MAC}$ : clave de autenticación a utilizar por la función de autenticación,  $MAC$ .
- $P1$  y  $P2$ : conjuntos de parámetros opcionales utilizados por las funciones  $KDF$  y  $MAC$ , respectivamente.

190. Para enviar un mensaje cifrado con ECIES, es necesario llevar a cabo los pasos que se muestran en el Algoritmo 3 (véase la Figura 6).

---

**Algoritmo 3** ECIES: algoritmo de cifrado

---

1.  $\mathcal{U}$  crea una pareja de claves temporales  $u$  y  $U$ . Para ello, selecciona aleatoriamente un valor  $u$ , de manera que su correspondiente clave pública sea  $U = uG$ .
  2.  $\mathcal{U}$  genera un dato secreto compartido utilizando la función  $KA$  a partir de su clave secreta temporal  $u$ , de la clave pública permanente  $V$  del usuario receptor y opcionalmente del cofactor  $h$ . En caso de utilizar la función DH, el dato secreto se calculará como  $uV = uvG$ . En cambio, si se utiliza la función DH con cofactor o DHC (*Diffie-Hellman with cofactor*), el dato secreto será calculado como  $huV = huvG$ .
  3.  $\mathcal{U}$  utiliza la función  $KDF$  acordada para generar, a partir del dato secreto compartido (y opcionalmente también de los parámetros  $P1$  acordados de forma previa entre emisor y receptor), una cadena binaria que se interpretará como la concatenación de las claves  $k_{MAC}$  y  $k_{ENC}$ .
  4.  $\mathcal{U}$  emplea la función de cifrado simétrico  $ENC$ , junto con el mensaje  $m$  y la clave  $k_{ENC}$ , para generar el mensaje cifrado  $c$ .
  5.  $\mathcal{U}$  utiliza la función  $MAC$  acordada al inicio del proceso y la clave  $k_{MAC}$  para generar el elemento  $tag$ , que es la etiqueta de autenticación correspondiente a la concatenación del mensaje cifrado  $c$  junto con los parámetros opcionales  $P2$ .
  6. Finalmente,  $\mathcal{U}$  envía a  $\mathcal{V}$  el criptograma  $C = (U, c, tag)$  compuesto por la clave pública temporal  $U$ , el mensaje cifrado  $c$  y la etiqueta  $tag$ .
- 

191. Para descifrar un mensaje que ha sido cifrado con ECIES se sigue el Algoritmo 3 (véase la Figura 7).

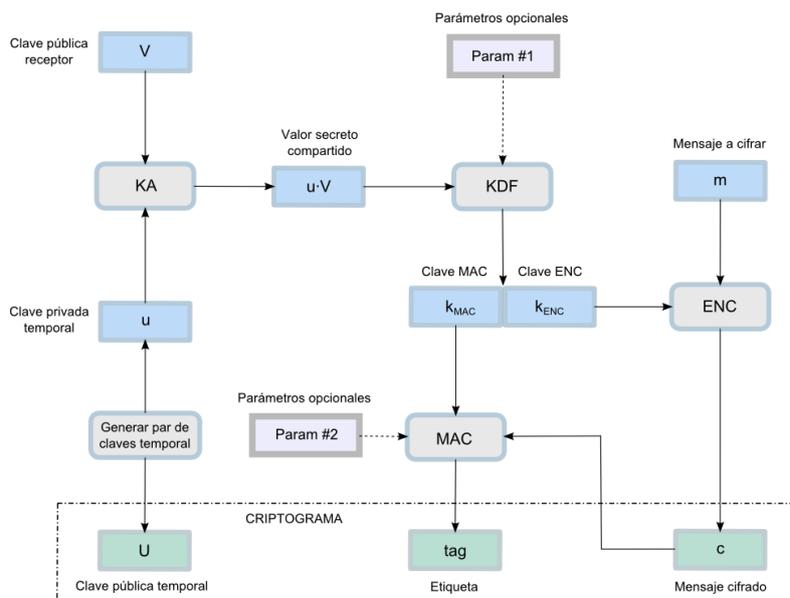


Figura 6: Esquema de cifrado mediante el criptosistema híbrido ECIES

#### Algoritmo 4 ECIES: algoritmo de descifrado

1. A partir del criptograma recibido,  $\mathcal{V}$  identifica y extrae la clave pública temporal  $U$ , el mensaje cifrado  $c$  y la etiqueta  $tag$ .
2. Con la clave pública temporal  $U$ , su propia clave privada permanente  $v$  y opcionalmente el cofactor  $h$ ,  $\mathcal{V}$  multiplica esos elementos para producir el dato secreto compartido, ya que  $vU = vuG = uvG = uV$ .
3. Una vez obtenido el dato secreto compartido,  $\mathcal{V}$  utiliza la misma función  $KDF$ , junto con los mismos parámetros opcionales  $P1$ , para obtener las claves  $k_{MAC}$  y  $k_{ENC}$ .
4. Con la clave  $k_{MAC}$ , el mensaje cifrado  $c$  y los parámetros opcionales  $P2$ ,  $\mathcal{V}$  calcula la etiqueta asociada a dichos datos.
5.  $\mathcal{V}$  compara la etiqueta generada por él con la recibida como parte del criptograma. Si ambos valores coinciden, el usuario puede proseguir con el proceso de descifrado. En caso contrario, debe detener el proceso.
6. Finalmente,  $\mathcal{V}$  recupera el mensaje original  $m$  utilizando la función de descifrado correspondiente a la función de cifrado  $ENC$ , la clave  $k_{ENC}$  y el mensaje cifrado  $c$ .

192. Si se emplea ECIES, la función de cifrado y descifrado  $ENC$  será cualquiera de las correspondientes a los cifradores en bloque autorizados en la Tabla 2.1.

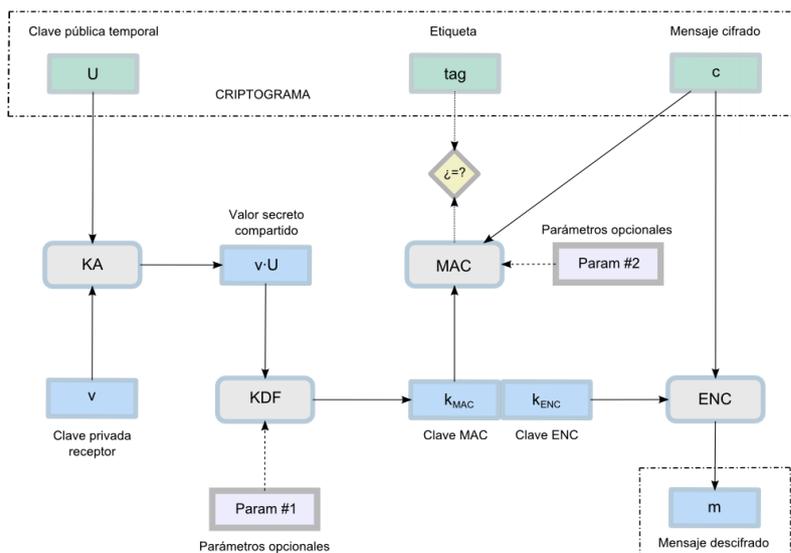


Figura 7: Esquema de descifrado mediante el criptosistema híbrido ECIES

### 3.2.3. FIRMAS DIGITALES

193. Los esquemas de firma digital permiten realizar la firma de un documento y constan de tres protocolos: uno de generación del par de claves (pública y privada); otro de elaboración de la firma, cuyas entradas son la clave privada del firmante y el mensaje a firmar) y cuya salida es la firma del mensaje; y un protocolo de verificación de la firma, cuyas entradas son la clave pública del firmante, el mensaje firmado y la firma, siendo su salida o Verdadero o Falso. Los esquemas de firma digital garantizan la autenticación de los datos y el no repudio.

#### 3.2.3.1. FIRMAS DIGITALES NO RESISTENTES A LA COMPUTACIÓN CUÁNTICA

#### 194. Algoritmos DSA y ECDSA

195. De forma muy resumida, el algoritmo de firma digital (*Digital Signature Algorithm* o DSA) [162] se muestra como Algoritmo 5.

---

#### Algoritmo 5 DSA: generación de parámetros

---

1. Un primo  $p$  y otro primo  $q$ , divisor de  $p - 1$ .
  2. Un generador  $g$  de un subgrupo cíclico de  $\mathbb{Z}_p^*$  de orden  $q$ .
  3. Su clave privada será el entero,  $x$ , con  $0 < x < q$ .
  4. Su clave pública se determina calculando  $y = g^x \pmod{p}$ .
-

196. Para la elaboración de la firma digital DSS de un mensaje,  $m$ , se sigue el Algoritmo 6.

---

**Algoritmo 6** DSA: elaboración de la firma digital

---

1. Se selecciona un entero aleatorio  $k \in \mathbb{Z}_q^*$ .
  2. Se calcula  $r = (g^k \pmod{p}) \pmod{q}$ .
  3. Se resuelve en la incógnita  $s$  la congruencia dada por  $m = -x \cdot r + k \cdot s \pmod{q}$ , calculando  $k^{-1} \pmod{q}$  y  $s = k^{-1}(m + x \cdot r) \pmod{q}$ .
  4. La firma digital es el par  $(r, s)$ .
- 

197. Para verificar la firma digital  $(r, s)$  de  $m$ , se sigue el Algoritmo 7.

---

**Algoritmo 7** DSA: verificación de la firma digital

---

1. Se determina  $w = s^{-1} \pmod{q}$ .
  2. Se calculan  $u_1 = m \cdot w \pmod{q}$  y  $u_2 = r \cdot w \pmod{q}$ .
  3. Se determina  $v = (g^{u_1} y^{u_2} \pmod{p}) \pmod{q}$ .
  4. Se acepta la firma si y solo si  $v = r$ .
- 

198. El algoritmo de firma digital basado en curvas elípticas (*Elliptic Curve Digital Signature Algorithm* o ECDSA) fue propuesto originalmente en 1992 por Scott Vanstone en respuesta a la convocatoria del NIST para el establecimiento de un estándar de firma digital (*Digital Signature Standard* o DSS). Posteriormente, fue aceptado como estándar por ISO [100], ANSI X9.62 [11], IEEE [89] y NIST ([162]).

199. Como todos los esquemas de firma digital, tiene tres fases: 1) generación de claves, 2) elaboración de la firma y 3) verificación de la firma. Los parámetros del ECDSA son una curva elíptica definida sobre un cuerpo finito primo,  $\mathbb{F}_p$ ,  $E(\mathbb{F}_p)$  y un punto de la curva de orden primo  $q$ ,  $G \in E$ , que actúa como generador con  $q \approx p$ .

200. La clave privada del usuario  $\mathcal{A}$  es un entero aleatorio en el intervalo  $[1, q - 1]$ ,  $a$ , y su clave pública es el punto de la curva dado por  $A = aG$ .

201. Para elaborar la firma del documento  $m$ ,  $\mathcal{A}$  sigue el Algoritmo 8.

**Algoritmo 8 ECDSA: elaboración de la firma digital**

1. Genera al azar un entero  $k$ ,  $1 < k < q - 1$ . Calcula  $kG = (x_1, y_1)$  y  $r = x_1 \pmod{q}$ . Si  $r = 0$  elige otro valor de  $k$ .
2. Calcula  $s = k^{-1}(\tilde{m} + a \cdot r) \pmod{q}$ , siendo  $\tilde{m}$  el resumen de  $m$  por una función resumen segura acordada de antemano. Si  $s = 0$ , se elige otro valor para  $k$ .
3. La firma para  $m$  de  $\mathcal{A}$  es el par  $(r, s)$ .

202. Para verificar la firma de  $\mathcal{A}$  para  $m$ , el usuario  $\mathcal{B}$  procede según se indica en el Algoritmo 9.

**Algoritmo 9 ECDSA: verificación de la firma digital**

1. Obtiene los parámetros y la firma de  $\mathcal{A}$ ,  $(r, s)$  para  $m$ .
2. Comprueba que  $r, s \in [1, q - 1]$ .
3. Calcula  $w = s^{-1} \pmod{q}$  y  $\tilde{m}$ .
4. Determina  $u_1 = \tilde{m} \cdot w \pmod{q}$  y  $u_2 = r \cdot w \pmod{q}$ .
5. Calcula  $u_1G + u_2A = (x_0, y_0)$  y  $v = x_0 \pmod{q}$ .
6. Acepta la firma si y solo si se cumple que  $v = r$ .

**203. Algoritmos de firma de Schnorr y EC-Schnorr**

204. La firma propuesta por Schnorr es una variante del esquema de firma de DSA [189]. Este esquema usa el mismo protocolo de generación de claves que el DSA sin restricciones en el tamaño de los parámetros y también emplea un subgrupo de orden  $q$  de  $\mathbb{Z}_p^*$ . Además, se emplea una función resumen  $\mathfrak{h}: \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$ .

205. Los pasos a seguir para la elaboración de la firma para un mensaje  $m$  son los mostrados en el Algoritmo 10.

**Algoritmo 10 Schnorr: elaboración de la firma digital**

1. El firmante genera un entero aleatorio secreto  $k$ , con  $1 \leq k \leq q$ .
2. Calcula  $u = g^k \pmod{p}$ ,  $r = \mathfrak{h}(m||u)$  y  $s = (ar + k) \pmod{q}$ , siendo  $a$  su clave privada.
3. La firma es el par  $(s, r)$ .

206. Para la verificación de la firma se sigue el Algoritmo 11.

**Algoritmo 11** Schnorr: verificación de la firma digital

1. El verificador calcula  $w = g^s A^{-r} \pmod{p}$  y  $\bar{r} = \mathfrak{h}(m||w)$ , donde  $A = g^a \pmod{p}$  es la clave pública del firmante.
2. La firma se acepta si y solo si  $r = \bar{r}$ .

207. La versión de la firma de Schnorr para curvas elípticas se conoce como EC-Schnorr (*Elliptic Curve Based Schnorr Signature Algorithm*) [37].

208. En este algoritmo se considera que la clave privada del firmante,  $\mathcal{A}$  es  $d_A$  y su clave pública es  $A$ , siendo  $(p, a, b, G, n, h)$  los parámetros de la curva elíptica  $E$ , esto es,  $p > 2$  es un primo que define el cuerpo base,  $a, b$  los parámetros de la curva,  $G$  un punto de la curva elíptica de orden  $n$  y  $h$  el cofactor correspondiente. El algoritmo para la elaboración de la firma es el Algoritmo 12.

**Algoritmo 12** EC-Schnorr: elaboración de la firma digital

1. El firmante genera un entero aleatorio secreto  $k$ , con  $1 \leq k \leq n - 1$ .
2. Calcula  $Q = kG = (Q_x, Q_y)$  en  $E$  y  $r = \text{OS2I}(\text{FE2OS}(Q_x)) \pmod{n}$ . Si  $r = 0$ , se elige otro valor para  $k$ .
3. Determina el valor de  $s = (kr - \text{OS2I}(\mathfrak{h}(m))d_A) \pmod{n}$ . Si  $s = 0$ , se elige otro valor para  $k$ .
4. La firma es el par  $(r, s)$ .

209. En el algoritmo anterior, OS2I es una primitiva que convierte cadenas de octetos en cadenas de enteros (*Octet String to Integer Conversion*) como sigue: si  $o_{l-1}o_{l-2} \dots o_2o_1$  es una cadena de  $l$  octetos, cada uno de ellos se puede interpretar como un entero no negativo en base 256, de modo que el bit más significativo es el que está más a la izquierda; en definitiva, se obtiene el entero dado por

$$x = x_{l-1} \cdot 256^{l-1} + x_{l-2} \cdot 256^{l-2} + \dots + x_1 \cdot 256 + x_0, \quad (3.1)$$

con  $0 \leq i \leq l - 1$ . Por su parte, la primitiva FE2OS convierte elementos del cuerpo base en cadenas de octetos (*Field Element to Octet String Conversion Primitive*). En este caso, si  $x \in \mathbb{F}_p$  es un elemento del cuerpo base, se convierte en una cadena de octetos de longitud  $l = \lceil \log_{256} p \rceil$  aplicando la función de conversión I2OS con el parámetro  $l$ , es decir,

$$\text{FE2OS}(x) = \text{I2OS}(x, l).$$

Finalmente, la primitiva que convierte enteros en cadenas de octetos, I2SO (*Integer to Octet String Conversion Primitive*), considera un entero no negativo,  $x$ , con una longitud deseada,  $l$ , y lo convierte en una cadena de octetos. El parámetro  $l$  debe

verificar  $256^l > x$ . La idea es escribir el entero  $x$  en su representación única como un  $l$ -dígito en base 256 como se muestra en la expresión (3.1), con  $x_i < 256$  para  $0 \leq i \leq l-1$ , de modo que la cadena de octetos es  $o_{l-1}o_{l-2} \dots o_2o_1$ .

210. El algoritmo de verificación de la firma de EC-Schnorr es el Algoritmo 13.

---

**Algoritmo 13** EC-Schnorr: verificación de la firma digital

---

1. El verificador comprueba los tamaños de  $r$  y  $s$ , esto es,  $r, s \in \{1, 2, \dots, n-1\}$ .
  2. Calcula  $\bar{r} = r^{-1} \pmod{n}$
  3. Determina los valores de  $u_1 = \bar{r} \cdot \text{OS2I}(\mathfrak{h}(m)) \pmod{n}$  y  $u_2 = \bar{r}s \pmod{n}$ .
  4. Calcula  $Q = (u_1G + u_2A)$  sobre  $E$ . Si  $Q = \mathcal{O}$ , el algoritmo finaliza con error.
  5. Calcula  $v = \text{OS2I}(\text{FE2OS}(Q_x)) \pmod{n}$ .
  6. Acepta la firma si y solo si  $v = r$ .
- 

211. Existen algunas variantes de los algoritmos DSA y ECDSA que pueden verse en [187]. En particular, unas breves descripciones de las variantes KCDSA, EC-KCDSA y EC-GDSA se incluyen a continuación.

### 212. Algoritmos KCDSA y EC-KCDSA

213. En [111, 124], se ha propuesto un algoritmo como estándar para la firma digital coreana basado en certificados (*Korean Certificate-based Digital Signature Algorithm* o KCDSA). En este algoritmo de firma, la clave pública se valida mediante un certificado del tipo X-509 emitido por una autoridad de confianza.

214. Los parámetros del dominio del KCDSA, esto es, los parámetros que comparten un grupo de usuarios son los siguientes:

- Un primo grande,  $p$ , de modo que su tamaño (longitud en bits) varíe entre 512 y 2048 bits, es decir,  $|p| = 512 + 256i$ ,  $0 \leq i \leq 6$ , con incrementos de múltiplos de 256 bits.
- Un factor primo  $q$  de  $p-1$  tal que  $|q| = 128 + 32j$ ,  $1 \leq j \leq 4$ . Esto es, el tamaño de  $q$  puede variar entre 128 y 256 bits, con incrementos de múltiplos de 32 bits. Además, se requiere que  $(p-1)/2q$  sea primo o, al menos, que todos sus factores primos sean mayores que  $q$ . La razón de esta última restricción es evitar los ataques contra los subgrupos de orden pequeño de  $\mathbb{Z}_p^*$ .

- Un elemento base,  $g$ , de orden  $q$  modulo  $p$ , es decir,  $g \neq 1$  and  $g^q \equiv 1 \pmod{p}$ .

215. Por su parte, los parámetros del usuario son  $x, y, z$ , generados de la siguiente manera:

- $x$  es la clave privada del firmante, elegida al azar en  $\mathbb{Z}_q^*$ .
- $y$  es a clave pública del firmante que permite la verificación de su firma y se calcula como  $y = g^{x^{-1}} \pmod{p}$ , siendo  $x^{-1}$  el inverso multiplicativo de  $x$  módulo  $q$ .
- $z$  es un resumen del valor  $C_D$ , es decir,  $z = \mathfrak{h}(C_D)$ . El valor  $C_D$  denota los datos de certificación del firmante que, al menos, debe contener el identificador distinguido del firmante, la clave pública  $y$  y los parámetros del dominio  $\{p, q, g\}$ .

216. El algoritmo para que el firmante,  $\mathcal{A}$ , elabore su firma digital para el mensaje  $m$  es el Algoritmo 14.

---

**Algoritmo 14** KCDSA: elaboración de la firma digital

---

1.  $\mathcal{A}$  genera al azar un entero  $k \in \mathbb{Z}_q^*$  y calcula  $w = g^k \pmod{p}$ .
  2. Calcula  $r = \mathfrak{h}(w)$  y  $e = r \oplus \mathfrak{h}(z||m) \pmod{q}$ .
  3. Determina  $s = x(k - e) \pmod{q}$ .
  4. (opcional) Si  $s = 0$  se elige otro valor de  $k$ , si no es así, la firma para  $m$  es  $(r||s)$ .
- 

217. A pesar de que la probabilidad de que  $s = 0$  sea despreciable, es conveniente verificar si se obtiene ese valor o no, dado que si  $s = 0$ , el resultado de la firma es independiente de la clave privada,  $x$ , del firmante, lo que puede dar lugar a problemas de autenticación y no repudio. En cualquier caso, no hay peligro de que los valores de  $x$  o de  $k$  sean descubiertos por un adversario.

218. También es importante señalar que como el paso que lleva más tiempo de computación es la determinación de  $w$ , es posible calcular el valor del par  $(k, r)$  de forma previa e independientemente del mensaje firmar, lo que puede acelerar los cálculos en línea. En efecto, bastaría entonces con calcular los siguientes dos valores para elaborar la firma digital del mensaje  $m$ :

$$r = \mathfrak{h}(g^k \pmod{p}), \text{ con } k \in_r \mathbb{Z}_q^*,$$

$$s = x(k - r \oplus \mathfrak{h}(z||m) \pmod{q}).$$

219. El algoritmo para que el usuario  $\mathcal{B}$  verifique la firma  $(r||s)$  de  $\mathcal{A}$  para el mensaje  $m$  es el Algoritmo 15.

---

**Algoritmo 15** KCDSA: verificación de la firma digital

---

1. En primer lugar comprueba la validez del certificado del firmante, extrae los datos de certificación,  $C_D$ , del certificado y calcula su resumen:  $z = \mathfrak{h}(C_D)$ .
  2. Comprueba el tamaño de  $r$  y de  $s$ , de modo que  $0 \leq r < 2^{|q|}$  y  $0 < s < q$ .
  3. Calcula el valor de  $e = r\mathfrak{h}(z||m) \pmod{q}$ .
  4. Determina  $w = y^s g^e \pmod{p}$ .
  5. Acepta la firma si y solo si se cumple que  $r = \mathfrak{h}(w)$ .
- 

220. La variante del KCDSA para curvas elípticas se conoce como EC-KCDSA (*Elliptic Curve Korean Certificate-based Digital Signature Algorithm*).

221. Al igual que en el caso del KCDSA, el EC-KCDSA utiliza un certificado del usuario que va a firmar el mensaje, de modo que los datos de certificado se denotan como  $C_D$  y se utilizan para calcular el valor  $z = \mathfrak{h}(C_D)$ , siendo  $\mathfrak{h}$  la función resumen segura seleccionada. En este caso,  $C_D$  debe contener, al menos, el identificador del usuario, su clave pública y los parámetros del dominio.

222. Los parámetros del dominio de EC-KCDSA son los necesarios para definir la curva elíptica sobre el cuerpo que se determine. En este caso, tales parámetros son:

- Un entero positivo,  $m > 1$ , o un número primo  $p > 2$  y un entero  $m$ , que definen el cuerpo finito a considerar,  $\mathbb{F}_{2^m}$  o  $\mathbb{F}_{p^m}$ .
- Los coeficientes  $a, b$  que definen la curva elíptica,  $E$ , dada por su ecuación correspondiente:

$$E: \begin{cases} y^2 + xy = x^3 + ax^2 + b, & \text{para } \mathbb{F}_{2^m} \\ y^2 = x^3 + ax + b, & \text{para } \mathbb{F}_{p^m} \end{cases}$$

- Un primo  $q$  que divida al orden de la curva,  $\#(E)$ . En este caso, la elección de  $q$  debe seguir las especificaciones estándares de modo que su tamaño no ponga en peligro la seguridad del sistema.
- Un punto de la curva de orden  $q$ ,  $G = (G_x, G_y)$ .

223. Por otra parte, los parámetros del usuario son:

- La clave de firma privada del firmante, elegida al azar,  $x \in \mathbb{Z}_q^*$ .

- La clave pública de verificación de la firma,  $Y$ , calculada como el punto de la curva dado por  $Y = x^{-1}G$ , siendo  $x^{-1}$  el inverso multiplicativo de  $x$  módulo  $q$ .
- Los datos de certificación,  $z$ , como en el esquema KCDSA.

224. El algoritmo para la elaboración de la firma es el Algoritmo 16.

---

**Algoritmo 16** ECKCDSA: elaboración de la firma digital

---

1.  $\mathcal{A}$  genera al azar un entero  $k \in \mathbb{Z}_q^*$  y calcula  $W = kG$  sobre  $E$ .
  2. Calcula  $r = \mathfrak{h}(W) = \mathfrak{h}(W_x || W_y)$  y  $e = r \oplus \mathfrak{h}(z || m) \pmod{q}$ .
  3. Determina  $s = x(k - e) \pmod{q}$ .
- 

225. Como en el caso del KCDSA, una parte del algoritmo se puede realizar con antelación (fuera de línea), esto es, es posible calcular los valores de

$$r = \mathfrak{h}(kG), \text{ con } k \in_r \mathbb{Z}_q^*,$$

$$s = x(k - r \oplus \mathfrak{h}(z || m)) \pmod{q}.$$

226. El algoritmo de verificación de la firma se presenta como Algoritmo 17.

---

**Algoritmo 17** ECKCDSA: verificación de la firma digital

---

1. El verificador comprueba la validez del certificado del firmante y calcula su resumen:  $z = \mathfrak{h}(C_D)$ .
  2. Comprueba los tamaños de  $r$  y  $s$ :  $0 \leq r < 2^{|q|}$  y  $0 < s < q$ .
  3. Determina el valor de  $e = r \mathfrak{h}(z || m) \pmod{q}$ .
  4. Calcula  $W = (sY + eG)$  sobre  $E$ .
  5. Acepta la firma si y solo si  $r = \mathfrak{h}(W)$ .
- 

### 227. Algoritmo EC-GDSA

228. El esquema EC-GDSA es el algoritmo de firma digital alemán (*Elliptic Curve German Digital Signature Algorithm* [37, §4.2.2]). Se sabe que una de las desventajas del esquema ECDSA es que requiere calcular inversos en la fase de elaboración de la firma y teniendo en cuenta que esta operación es una de las más caras en la aritmética modular, evitar tener que hacerla reduce costes y tiempo. Por esta razón, en el esquema EC-GDSA el cálculo del inverso se hace en el momento de la

generación de la clave y no cuando se firma un mensaje. Esta modificación tiene en cuenta, además, que la firma se realiza con más frecuencia que la generación de claves, por lo que esta última se mantendrá constante durante un largo periodo de tiempo.

229. Para determinar la clave privada del usuario  $\mathcal{A}$ , se genera un entero aleatorio en el intervalo  $[1, q - 1]$ ,  $a_0$ , y se calcula  $a = a^{-1} \pmod{q}$ . Su clave pública es el punto de la curva dado por  $A = aG$ .
230. En la elaboración de la firma del documento  $m$ ,  $\mathcal{A}$  sigue los pasos señalados en el Algoritmo 18.

---

**Algoritmo 18** ECGDSA: elaboración de la firma digital

---

1. Genera al azar un entero  $k$ ,  $1 < k < q - 1$ . Calcula  $kG = (x_1, y_1)$  y  $r = x_1 \pmod{q}$ . Si  $r = 0$  elige otro valor de  $k$ .
  2. Calcula  $s = (k \cdot r - \tilde{m})a \pmod{q}$ , siendo  $\tilde{m}$  el resumen de  $m$  por una función resumen segura acordada de antemano. Si  $s = 0$ , se elige otro valor para  $k$ .
  3. La firma para  $m$  de  $\mathcal{A}$  es el par  $(r, s)$ .
- 

231. Para verificar la firma de  $\mathcal{A}$  para  $m$ , el usuario  $\mathcal{B}$  procede como se indica en el Algoritmo 19.

---

**Algoritmo 19** ECGDSA: verificación de la firma digital

---

1. Obtiene los parámetros y la firma de  $\mathcal{A}$ ,  $(r, s)$  para  $m$ .
  2. Comprueba que  $r, s \in [1, q - 1]$ .
  3. Calcula  $w = r^{-1} \pmod{q}$  y  $\tilde{m}$ .
  4. Determina  $u_1 = \tilde{m} \cdot w \pmod{q}$  y  $u_2 = s \cdot w \pmod{q}$ .
  5. Calcula  $u_1G + u_2A = (x_0, y_0)$  y  $v = x_0 \pmod{q}$ .
  6. Acepta la firma si y solo si se cumple que  $v = r$ .
- 

### 3.2.3.2. FIRMAS DIGITALES POSTCUÁNTICAS

#### 232. Algoritmo CRYSTALS-Dilithium

233. Esta propuesta de firma digital ha sido seleccionada por el NIST en su informe de 2022, después de la tercera ronda y basa su seguridad en problemas definidos sobre retículos [127]. Una descripción detallada del algoritmo se presenta en el capítulo

dedicado a la criptografía postcuántica (véase §9.2.4). Esta firma está emparejada con el KEM conocido como CRYSTALS-Kyber (véase §9.2.2).

#### 234. Algoritmo Falcon

235. El algoritmo de firma denominado Falcon [178] basa su seguridad en problemas definidos sobre retículos y también ha sido seleccionado en 2022 por el NIST [171] como esquema de firma electrónica. Una descripción más detallada de este algoritmo se muestra en el capítulo donde se aborda la criptografía postcuántica (véase §9.2.5).

#### 236. Algoritmo SPHINCS<sup>+</sup>

237. Las características del algoritmo de firma SPHINCS<sup>+</sup> [88], cuya seguridad se basa en una función *hash* sin estado, pueden analizarse con más detalle en el capítulo dedicado a la criptografía postcuántica (véase §9.3.1). Al igual que los algoritmos anteriores, también ha sido seleccionado por el NIST como algoritmo de firma [171] en 2022.

### 3.2.3.3. FIRMAS DIGITALES PARA FIRMWARE

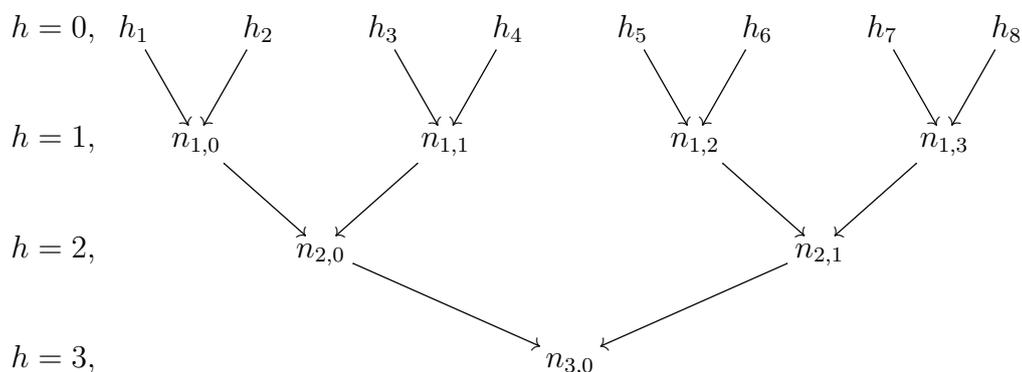
#### 238. Algoritmo XMSS

239. El algoritmo de firma XMSS fue propuesto por Buchmann et al. [46] y desarrollado posteriormente en [87] y [170]. como un algoritmo extendido del esquema de firma de Merkle o MSS (*Merkle Signature Scheme*) [138]. Ambos esquemas se conocen como esquemas de firmas basadas en *hashes* o HBS (*Hash-Based Signatures*). Las HBS son unos de los primeros protocolos criptográficos asimétricos propuestos y se basan en el esquema de firma única de Lamport [120]. Su fundamento son los conocidos árboles de Merkle, que a su vez son un tipo particular de árbol binario.

240. Para comprender el esquema XMSS, presentaremos brevemente el esquema MSS (basado en el esquema de Lamport), que se puede utilizar para firmar un número limitado de mensajes, de hecho, una potencia de 2 ( $N = 2^H$ ).

241. Un árbol binario es una estructura en forma de árbol de modo que cada uno del mismo tiene exactamente tres aristas, dos de las cuales van al nivel más cercano a las hojas y la otra va a la raíz. Las hojas tienen una única arista y la raíz tiene solo dos aristas.

242. Un ejemplo sencillo de árbol binario con  $2^H = 2^3 = 8$  hojas se muestra a continuación:



En general, los nodos se denotan como  $n_{h,k}$ , donde  $h$  es el número del piso y  $k$  es el orden dentro de un piso, de izquierda a derecha, de modo que se tiene  $0 \leq k \leq 2^{H-h} - 1$  para el piso número  $h$ .

243. Para implementar una HBS se pueden usar árboles de Merkle junto con un esquema de firma única u OTS (*One Time Signature*), como la de Lamport. Un OTS es un esquema de firma con una clave privada que se usa para firmar un mensaje y la clave pública correspondiente se usa para verificar dicha firma. El problema es que la clave privada solo se puede usar una vez para firmar un mensaje.

244. La propuesta de Lamport se puede resumir de la siguiente manera [120]: dado un mensaje  $m \in \{0, 1\}^n$  y una función unidireccional  $F: \{0, 1\}^k \rightarrow \{0, 1\}^k$ , el firmante genera una clave privada y la clave pública correspondiente. La clave privada consta de  $2n$  valores aleatorios de  $k$  bits cada uno,  $s_{i,j}$  para  $i \in \{1, \dots, n\}$  y  $j \in \{0, 1\}$ . La clave es el resultado de aplicar la función unidireccional  $F$  a cada valor secreto de modo que se tiene  $p_{i,j} = F(s_{i,j})$  y la clave pública es  $pk = (p_{1,0}, p_{1,1}, \dots, p_{n,0}, p_{n,1})$ . Dado un mensaje,  $m$ , el firmante revela selectivamente los secretos  $s_{i,m_i}$ , siendo  $m_i$  es el  $i$ -ésimo bit de  $m$ . Esta lista de valores componen la firma buscada. Finalmente, dada firma, el verificador debe aplicar  $F$  a cada secreto revelado  $s_{i,m_i}$  de modo que la firma se considera válida si  $F(s_{i,m_i}) = p_{i,m_i}$  para todos los bits de  $m$ .

245. Un árbol HBS es un árbol Merkle cuyas hojas son las claves públicas del OTS, y cada nodo interno del árbol consiste en el *hash* de sus dos hijos. La raíz del árbol es la clave pública de la construcción de Merkle.

246. Dados  $2^H$  pares de claves privadas/públicas,  $(s_i, p_i)$ , con  $0 \leq i \leq 2^H - 1$ , si  $\mathfrak{h}$  es una función *hash*, en el árbol de Merkle se tendría que

$$n_{0,k} = \mathfrak{h}(P_k), \quad 0 \leq k \leq 2^H - 1,$$

$$n_{h,k} = \mathfrak{h}(n_{h-1,2k} \parallel n_{h-1,2k+1}),$$

para  $1 \leq h \leq H$ ,  $0 \leq k \leq 2^{H-h} - 1$ , siendo  $\parallel$  la concatenación. Una vez completado el árbol, el usuario publica como clave pública global el valor del nodo raíz,  $n_{H,0}$ , y sirve como compromiso de todo el conjunto de claves públicas,  $p_i$ .

247. De forma más precisa, el esquema de Merkle utiliza  $2^H$  instancias de OTS, cada una con un par de claves públicas y privadas,  $(p_i, s_i)$ , y construye un árbol de Merkle cuyas hojas son los *hashes* de las claves públicas de las instancias de OTS, y cada nodo es el *hash* de sus dos nodos secundarios. La clave pública consiste en la raíz del árbol de Merkle y la clave privada contiene las claves privadas de todas las instancias de OTS de  $2^H$ . La firma de un mensaje  $m$  consta de un índice  $i$  que especifica una instancia OTS  $(p_i, s_i)$ , la firma única  $\sigma_{OTS}$  en  $m$  bajo la clave  $p_i$ , la clave  $p_i$  y una ruta de autenticación  $A_i$  que se utiliza para verificar la validez de  $p_i$ .
248. La ventaja del esquema MSS es que se considera resistente a los algoritmos cuánticos, de hecho, solo depende de la existencia de funciones *hash* seguras.
249. Desde que Merkle anunció su propuesta, se ha publicado una gran cantidad de trabajos que intentan mejorar diferentes aspectos de MSS [60, 44, 47, 48, 43], etc. En particular, es de destacar la propuesta XMSS de Buchmann [46].
250. El esquema XMSS usa un OTS que solo puede firmar un mensaje con una clave, pero para superar esta limitación, se usa un árbol HBS que permite reducir la autenticidad de muchas claves de verificación OTS a una clave XMSS pública. Además, para minimizar los requisitos de almacenamiento, se utilizan generadores pseudo-aleatorios.
251. Para reducir el tamaño de la clave privada (que consiste en las claves privadas de las  $2^H$  instancias OTS) se utiliza una PRF con una semilla maestra de  $n$  bits con el fin de generar una semilla OTS de  $n$  bits para cada instancia OTS, que a su vez se utiliza para generar la clave privada de esa instancia.
252. El OTS que usa XMSS es una variante de la OTS de Winternitz (W-OTS), llamada W-OTS<sup>+</sup> en [138], [45] y [86], que elimina el requisito de una función *hash* resistente a colisiones.
253. Además de los árboles binarios ya mencionados, también son de interés los árboles binarios no balanceados, llamados L-árboles [18]. Estos se utilizan exclusivamente para los hash de claves públicas W-OTS<sup>+</sup>. Las  $\ell$  hojas de un L-Tree son los elementos de una clave pública W-OTS<sup>+</sup> y el árbol se construye como los ya mencionados, con la salvedad de que el nodo izquierdo que no tiene un hermano derecho se eleva a un nivel superior del L-árbol hasta que se convierte en el hermano derecho de otro nodo. Los L-árboles tienen una altura de  $\lceil \log \ell \rceil$  y, por lo tanto, necesitan  $\lceil \log \ell \rceil$  máscaras.
254. Además, XMSS usa una estructura de L-árbol para reducir la clave pública OTS [29] y utiliza máscaras de cegamiento para las cadenas W-OTS<sup>+</sup> y para cada nodo en HBS y en el L-árbol. Las claves ciegas se generan de forma pseudo-aleatoria para cada nodo del árbol mientras se genera o verifica una firma.
255. Los parámetros públicamente conocidos para generar las claves de firma OTS son los siguientes:

- Los parámetros de seguridad son  $n, w, m, p \in \mathbb{N}$ , que corresponden, respectivamente, a la longitud del *hash* (en bytes), el parámetro Winternitz, la longitud del mensaje en bits y el número de cadenas Winternitz utilizadas en una sola operación OTS.
- Una familia de funciones  $F(n) = \{f_K: \{0, 1\}^n \rightarrow \{0, 1\}^n | K \in \{0, 1\}^n\}$ .
- La altura del árbol,  $H \in \mathbb{N}$  (recuérdese que XMSS permite hacer  $2^H$  firmas usando un par de claves).
- Una función resumen  $h_K$  elegida aleatoriamente con distribución uniforme de la familia  $\mathfrak{H}(n) = \{h_K: \{0, 1\}^{2n} \rightarrow \{0, 1\}^n | K \in \{0, 1\}^n\}$ .
- Una cadena elegida al azar con la distribución uniforme,  $x \in \{0, 1\}^n$ . La cadena  $x$  se usa para construir las claves de verificación única.

256. Para  $K, x \in \{0, 1\}^n$  y  $e \in \mathbb{N}$ , se define  $f_K^e(x)$  como sigue:

$$f_K^0(x) = K, \quad f_K^{e-1}(x) = K' \text{ if } e > 0, \quad \text{y } f_K^e(x) = f_{K'}(x).$$

257. En la Tabla 3.5 se incluyen las características de los esquemas de firma digital autorizados.

Primitiva	Esquema	Rec./Her.	Referencias	Notas
RSA	OAEP PKCS#1v2.1	R	[105], [93], [185]	36,37
FF-DLOG	KCDSA Schnorr DSA	R	[100] [100] [162], [100]	36,37 36,37,38 36,37
EC-DLOG	EC-KCDSA EC-DSA EC-GDSA EC-Schnorr	R	[100] [162], [100] [37] [100]	36,37 36,37,38 36,37 36,37
RSA	PKCS#1v1.5	H	[105], [93], [185]	36,37,39
Hash	XMSS	R	[46], [87], [170]	
Retículo	Dilithium	R	[127]	40, 41
Retículo	Falcon	R	[178]	40, 41
Hash	SPHINCS+	R	[88]	40, 41

Tabla 3.5: Esquemas de firma digital autorizados

258. Nota 36 **[Función resumen]** El esquema estará autorizado siempre que la función resumen subyacente lo esté (ver §2.1.3).

259. Nota 37 **[Problema difícil]** El esquema estará autorizado siempre que el problema matemático subyacente utilice los parámetros autorizados (ver §3.1).

260. Nota 38 **[Aleatoriedad DSA]** En el algoritmo DSA y en sus variantes de curva elíptica, el procedimiento de elaboración de firmas genera un valor aleatorio. La exfiltración de los valores aleatorios por firma utilizados durante la elaboración de las firmas plantea riesgos, a largo plazo, para la confidencialidad de las claves asociadas. Por lo tanto, debe evitarse tal exfiltración. En principio, esto se refiere tanto a la filtración estadística a través de sesgos en el generador de números aleatorios utilizado como a las filtraciones en el valor de bits aleatorios particulares que puede obtener un atacante (por ejemplo, a través del análisis de canales laterales). Por lo tanto, se recomienda utilizar un generador de números aleatorios con un fuerte postprocesamiento criptográfico, seguridad hacia atrás mejorada y reelección de la semilla desde una fuente verdaderamente aleatoria (ver §5.1).

261. Nota 39 **[Verificación de formato PKCS]** Las comprobaciones de formato deben implementarse cuidadosamente para evitar los ataques de Bleichenbacher [32].

262. Nota 40 **[Firmas digitales postcuánticas]** Los mecanismos de firma digital seleccionados por el NIST en 2022 [171] se consideran cuánticamente seguros, esto es, se puede afirmar que, hasta la fecha, no se conocen ataques que puedan vulnerar su seguridad, ni siquiera recurriendo a la potencia de cómputo de ordenadores cuánticos. Por este motivo, estos son los mecanismos especialmente recomendados.

263. Nota 41 **[Implementación de las firmas digitales postcuánticas]** Se recomienda no implementar este algoritmo hasta que el NIST publique el estándar definitivo, dado que hasta entonces, el algoritmo puede sufrir variaciones.

### 3.2.4. ESQUEMAS DE AUTENTICACIÓN DE ENTIDAD ASIMÉTRICA

264. Los esquemas de autenticación de entidades asimétricas permiten que una entidad pruebe su identidad ante otra, demostrando su conocimiento de una clave privada. Estos esquemas son esquemas interactivos por naturaleza y generalmente consisten en utilizar un esquema de firma en un protocolo de desafío-respuesta aleatorio. En esta subsección no se proporciona ningún listado autorizado de estos esquemas porque, aunque pueden basarse en esquemas de firma, son un tipo distinto de esquemas, con diferentes objetivos de seguridad. Por lo tanto, la misma clave no debe ser utilizada por un esquema de firma y por un esquema de autenticación de entidad asimétrica (ver Nota 89).
265. En cuanto a los esquemas de autenticación de entidades simétricas, el desafío debe verificar algunas propiedades como las señaladas en la Nota 17.

### 3.2.5. ESTABLECIMIENTO DE CLAVES

266. Los esquemas de establecimiento de claves permiten que dos o más partes generen un secreto común sin utilizar ningún valor secreto previamente compartido. Por lo general, estos esquemas se combinan con los de autenticación asimétrica o simétrica basada en un par de claves público-privada o una clave secreta compartida. El esquema de establecimiento de claves entre dos partes más utilizado es el de Diffie y Hellman [59] y se basa en el problema del logaritmo discreto, ya sea utilizando la aritmética multiplicativa (grupo multiplicativo de un cuerpo finito  $\mathbb{F}_p^*$ ), en cuyo caso se denota simplemente como DH, o la aditiva (grupo de los puntos de una curva elíptica definida sobre un cuerpo finito,  $E(\mathbb{F}_q)$ ), denotándose entonces como EC-DH (*Elliptic Curve-Diffie-Hellman*). En cualquiera de los casos, se procede de la siguiente manera:
1. Ambos usuarios,  $A$  y  $B$ , acuerdan un grupo cíclico finito,  $G$ , y un generador del mismo  $g$ .
  2. Cada usuario genera un valor aleatorio  $v_i$ ,  $i \in \{A, B\}$  y envía al otro usuario el valor  $g^{v_i}$  en el caso multiplicativo y  $v_i \cdot g$  en el caso aditivo.
  3. Ambos usuarios pueden entonces calcular el elemento común del grupo, que vendrá dado como  $(g^{v_A})^{v_B} = (g^{v_B})^{v_A} = g^{v_A v_B}$  en el caso multiplicativo y como  $v_A(v_B \cdot g) = v_B(v_A \cdot g) = (v_A v_B)g$  en el caso aditivo. Es claro que cada uno de ellos puede calcular dicho valor a partir del propio valor aleatorio y del elemento recibido del otro usuario.
267. Es importante destacar que este protocolo es vulnerable a los ataques del hombre en el medio (MitM). En particular, se deben realizar pasos adicionales y se deben intercambiar datos adicionales para garantizar la autenticación de los usuarios y de los mensajes de establecimiento de claves.

## 268. Criptosistema DLIES

269. En 1997, Mihir Bellare y Philip Rogaway [24] presentaron un criptosistema denominado DLAES (*Discrete Logarithm Augmented Encryption Scheme*), que fue mejorado y renombrado posteriormente como DHAES (*Diffie-Hellman Augmented Encryption Scheme*) en 1998 [1] y como DHIES (*Diffie-Hellman Integrated Encryption Scheme*) en 2001 [2] y [3], para evitar confusiones con el AES. Si bien, en la actualidad es conocido como esquema de cifrado integrado basado en el logaritmo discreto o DLIES (*Discrete Logarithm Integrated Encryption Scheme*). DLIES ha sido estandarizado en ANSI X9.63 [12], IEEE 1363a [89] y en ISO/IEC 18033-2 [92].
270. DLIES es similar a ECIES (véase §3.2.2), con la diferencia de que el primero se basa en la dificultad de resolver el problema de Diffie-Hellman en un adecuado subgrupo de  $\mathbb{F}_p^*$ ; mientras que el segundo lo hace sobre curvas elípticas. DLIES combina un mecanismo de encapsulamiento de claves (para más detalles, véase §3.2.5) o KEM (*Key Encapsulation Mechanism*) con un mecanismo de encapsulamiento de datos (véase §3.2.5) o DEM (*Data Encapsulation Mechanism*).
271. DLIES requiere los siguientes elementos: un criptosistema simétrico,  $\mathcal{E}_K$ , (que puede ser cualquiera de los cifrados en bloque autorizados en esta guía, ver la Tabla 2.1), un código de autenticación de mensajes,  $\text{MAC}_{KM}$ , y una función de derivación de claves,  $H$ , que puede ser sencillamente una función resumen si su salida es, al menos, de la misma longitud que la total del material clave que se derivará. A continuación se detallan las características de DLIES.
272. El algoritmo para la generación de claves de DLIES se muestra como Algoritmo 20. La clave pública es el cuarteto formado por  $(p, g, A, q)$  y la clave privada es el entero  $a$ .

---

### Algoritmo 20 DLIES: generación de las claves

---

1. Se selecciona al azar un número primo de longitud adecuada,  $q$ .
  2. Se genera aleatoriamente un entero  $k$  con un tamaño adecuado para garantizar que  $kq$  tiene la longitud de la clave que se va a generar. Este proceso se repite hasta que  $p = kq + 1$  sea primo.
  3. Se selecciona  $x \in \mathbb{Z}_p^*$  de modo que  $x^k \neq 1$  y se considera  $g = x^k$ , de modo que  $g$  es un elemento de orden  $q$  en  $\mathbb{Z}_p^*$ .
  4. Se selecciona al azar un entero  $a$  con  $2 \leq a < q$  y se calcula  $A = g^a$ .
- 

273. El protocolo de cifrado es como sigue: dado un mensaje  $m \in \{0, 1\}^*$  y una clave pública del destinatario,  $(p, g, A, q)$ , el remitente sigue los pasos del Algoritmo 21.

---

**Algoritmo 21** DLIES: cifrado

---

1. Elige un número aleatorio  $1 \leq b \leq r - 1$  y calcula  $B = g^b$ .
  2. A continuación determina  $X = A^b$  y  $h = H(X)$ .
  3. A partir de  $h$ , se consideran el suficiente número de bits como para que sea posible crear una clave  $k_{ENC}$  para el criptosistema simétrico,  $\mathcal{E}$ , y otra clave,  $k_{MAC}$ , para el MAC.
  4. Se determina el texto cifrado  $c$  haciendo uso del sistema de cifrado  $ENC$ , el mensaje  $m$  y la clave correspondiente  $k_{ENC}$ .
  5. Se calcula una etiqueta de MAC,  $tag$ , con la función  $MAC$ , la clave generada,  $k_{MAC}$ , y el texto cifrado,  $c$ .
  6. Finalmente, se envía el criptograma formado por el trío  $(B, c, tag)$ , al destinatario.
- 

274. Para descifrar un criptograma,  $(B, c, tag)$ , el destinatario del mismo ejecuta el Algoritmo 22.

---

**Algoritmo 22** DLIES: descifrado

---

1. Calcula  $X = B^a$ , siendo  $a$  su clave privada.
  2. Determina  $h = H(X)$ , y las claves para el sistema de cifrado y el MAC, respectivamente,  $k_{ENC}$  y  $k_{MAC}$ .
  3. Calcula la etiqueta MAC correspondiente con los datos determinados,  $t$ , y comprueba si  $t = tag$ . Si no es correcta la verificación, el proceso finaliza con error.
  4. En caso contrario, se recupera el mensaje original,  $m$ , descifrando el texto cifrado,  $c$ , con la clave  $k_{ENC}$  y el inverso del sistema de cifrado  $\mathcal{E}$ .
- 

**275. Mecanismos de encapsulamiento de claves o de datos**

276. De forma muy general, puede decirse que un mecanismo de encapsulamiento de claves (KEM) funciona como un cifrado híbrido (ver §3.2.2), con alguna característica adicional. Como sistema híbrido, un KEM requiere de un algoritmo de cifrado simétrico y de otro asimétrico, a los que se añade un algoritmo para la generación de claves, que puede coincidir con el empleado por el cifrado asimétrico.

277. Por otra parte, dado que la clave simétrica es generalmente corta, se suele requerir un relleno (*padding*) para obtener mayor seguridad. Este relleno suele llevarse a cabo mediante un función de derivación de clave o una función hash. Dicho de otro

modo, un KEM simplifica este proceso al generar un elemento aleatorio en el grupo finito asociado al criptosistema asimétrico. Al margen de que se puedan definir KEM de manera específica, es posible utilizar un criptosistema asimétrico (posiblemente con longitud fija o acotada para el texto claro) para definir un KEM.

278. De forma más precisa, un KEM consta de los siguientes tres algoritmos:

1.  $\text{KEM}.\mathcal{G}()$ : un algoritmo de generación de claves que proporciona como salida un par de claves pública-privada,  $(pk, sk)$ , cuya estructura depende del esquema en particular que se considere.
2.  $\text{KEM}.\mathcal{E}_c(pk)$ : un algoritmo de encapsulado que considera como entrada una clave pública,  $pk$ , y genera como salida el par formado por una clave secreta y un texto cifrado  $(K, C)$ .
3.  $\text{KEM}.\mathcal{D}_c(sk, C)$ : un algoritmo de desencapsulado que toma como entrada una clave privada,  $sk$ , y un texto cifrado,  $c$ , proporcionando como salida una clave secreta,  $K$ .

279. Un KEM incluye como dato la longitud de un entero positivo  $\text{KEM}.\text{KeyLen}$ , que es la longitud de la clave secreta de  $\text{KEM}.\mathcal{E}_c$  y  $\text{KEM}.\mathcal{D}_c$ .

280. El algoritmo que implementa un KEM se presenta como el Algoritmo 23.

---

**Algoritmo 23** Algoritmo de un mecanismo de encapsulamiento de claves

---

1. Se genera un par de claves, pública-privada, mediante el algoritmo de generación de claves, con el parámetro de seguridad,  $\lambda$ , apropiado:

$$\mathcal{G}(\lambda) = (pk, sk).$$

2. Se encapsula la clave pública,  $pk$ , generando el par formado por una clave secreta y un texto cifrado:

$$\mathcal{E}_c(pk) = (K, C).$$

3. Se desencapsula la clave privada,  $sk$ , junto con el texto cifrado,  $C$ , obteniéndose como salida la clave secreta del paso anterior:

$$\mathcal{D}_c(sk, C) = K.$$


---

281. Debe tenerse en cuenta que la única entrada en el algoritmo de encapsulado es la clave pública del destinatario, esto es, no se usan mensajes para producir textos cifrados.

282. Los KEM deben ser correctos en el sentido de que para cualquier par de claves pública-privada,  $(pk, sk)$ , y para cualquier salida  $(K, C)$  del algoritmo  $\mathcal{E}_c(\cdot)$ , el texto cifrado  $C$  debe desencapsularse con  $sk$  para dar lugar a  $K$ . Este requisito puede ser relajado, de modo que se mantenga para todos, excepto para una fracción insignificante de pares de claves pública-privada. Por esta razón, puede darse el caso de que el algoritmo de desencapsulado falle en determinadas circunstancias.

### 283. Algoritmo CRYSTALS-Kyber

284. El algoritmo CRYSTALS-Kyber, o Kyber [191], es un KEM que ha sido seleccionado por el NIST en 2022 [171] como el mecanismo de encapsulamiento de claves que se supone resistente a la computación cuántica, es decir, no se conocen ataques que puedan vulnerarlo. Su seguridad se basa en problemas definidos sobre retículos. Para una información más detallada sobre sus propiedades y característica, remitimos al lector al capítulo donde se trata la criptografía postcuántica, en particular, véase §9.2.2.

285. Debe tenerse en cuenta que el KEM CRYSTALS-Kyber está pendiente de la estandarización definitiva por el NIST; por ello, se autorizará la versión de Kyber que estandarice este instituto.

286. En términos generales, un mecanismo de encapsulamiento de datos (DEM) proporciona un “sobre digital” que protege la confidencialidad y la integridad de los datos mediante técnicas criptográficas simétricas.

287. En concreto, un DEM, además de especificar la longitud de una clave  $DEM.KeyLen$ , está formado por dos algoritmos, uno de encapsulado y otro de desencapsulado:

1.  $DEM.\mathcal{E}_c(K, L, M)$ : el algoritmo de encapsulado considera como entradas una clave secreta,  $K$ , una etiqueta  $L$  y un texto claro,  $M$ , dando como salida un texto cifrado  $C$ . Además, la longitud de  $K$  es  $DEM.KeyLen$  y  $L$  y  $M$  pueden tener una longitud arbitraria. Este algoritmo puede fallar si las longitudes de  $L$  o  $M$  exceden de los límites considerados por la implementación realizada.
2.  $DEM.\mathcal{D}_c(K, L, C)$ : el algoritmo de desencapsulado toma como entrada la clave secreta,  $K$ , una etiqueta,  $L$ , y un texto cifrado, proporcionando como salida un texto claro  $M$ .

288. Los algoritmos  $DEM.\mathcal{E}_c(\cdot)$  y  $DEM.\mathcal{D}_c(\cdot)$  deben ser determinísticos y correctos, es decir, para cualquier clave secreta dada,  $K$ , cualquier etiqueta,  $L$ , cualquier texto claro,  $M$ , de modo que las longitudes de  $L$  y  $M$  no excedan los límites de la implementación considerada, se debe verificar que

$$DEM.\mathcal{D}_c(K, L, DEM.\mathcal{E}_c(K, L, M)) = M.$$

## 289. Algoritmo FrodoKEM

290. El algoritmo basado en retículos no estructurados denominado FrodoKEM [143] se puede considerar como una opción conservadora con relación a su seguridad. Debe tenerse en cuenta que FrodoKEM fue incluido por el NIST en la tercera ronda como un algoritmo alternativo y no como finalista, habiendo sido descartado a partir de la tercera ronda [168].

291. Las razones que el NIST alegó para su decisión se deben, en gran parte, a que su rendimiento es menor que el de otros algoritmos basados en retículos. Este menor rendimiento se debe a que FrodoKEM no emplea ninguna estructura matemática adicional al contrario de lo que sucede con otros algoritmos basados en retículos. Esta falta de estructura subyacente hace que FrodoKEM sea la opción de seguridad más conservadora, de ahí que el CCN, al igual que otros organismos de seguridad europeos, lo mantenga como algoritmo autorizado para KEM.

292. En la actualidad aún no existe un mecanismo definitivamente autorizado de establecimiento de claves que sea resistente a la computación cuántica (ver el capítulo 9). Al igual que en otros esquemas ya comentados anteriormente, esta situación puede ser un motivo de preocupación para las aplicaciones que requieren una confidencialidad a largo plazo, dado que un ordenador cuántico podría romper las sesiones de establecimiento de claves grabadas si dicho dispositivo se implementara antes del final de la vida útil de la información protegida. En tales situaciones, siguiendo los principios establecidos en §1.1, se recomienda combinar un mecanismo de establecimiento de clave autorizado junto con un mecanismo de establecimiento de clave resistente a la computación cuántica que sea uno de los candidatos del NIST o una clave previamente compartida solo por las partes afectadas. Esta acción se llevará a cabo mediante un mecanismo de derivación de claves autorizado en el que las claves establecidas previamente compartidas se utilicen como entrada.

293. **[Acuerdos de clave híbridos:]** Se recomienda el uso de acuerdos de clave híbridos, es decir, acuerdos de clave que combinen el esquema EC-DH con alguno de los algoritmos postcuánticos autorizados (por ejemplo Kyber o FrodoKEM).

294. En la Tabla 3.6 se muestran los esquemas de establecimiento de claves autorizados y sus características más relevantes.

Primitiva	Esquema	Rec./Her.	Referencias	Notas
FF-DLOG	DH DLIES-KEM	R	[98], [166] [92]	42,43,44 42,45,47
EC-DLOG	EC-DH ECIES-KEM	R	[98], [166] [92]	42,43,44 42,46,47
Retículo	CRYSTALS-Kyber FrodoKEM	R	[191] [143]	48, 49 50

Tabla 3.6: Esquemas de establecimiento de claves autorizados

295. Nota 42 [**Problema difícil**] Este esquema se considera autorizado si el problema matemático subyacente utiliza los parámetros adecuados.

296. Nota 43 [**Autenticación DH**] El protocolo DH es un establecimiento de claves no autenticado que puede ser víctima de ataques MitM. Con el fin de garantizar su seguridad, es preciso autenticar a las dos partes y los datos intercambiados durante el esquema de establecimiento de claves (valores públicos acordados e identidades). Esta autenticación requiere secretos a largo plazo.

297. Nota 44 [**Ataques al subgrupo DH/EC-DH**] Tanto si el protocolo Diffie-Hellman se define sobre el grupo multiplicativo de un cuerpo finito (DH) o sobre el grupo de puntos racionales de una curva elíptica definida sobre un cuerpo finito (EC-DH), debe asegurarse que los valores manipulados se encuentren en el subgrupo deseado y tengan un orden lo suficientemente grande (ver Notas 29 y 30).

298. Nota 45 [**Longitud de clave en DLIES**] Un requisito necesario para garantizar la seguridad del DLIES es que sea imposible determinar el logaritmo discreto en el subgrupo generado por  $g$ . Según el actual estado del arte, la dificultad del DLP en  $\mathbb{F}_p^*$  puede reducirse significativamente mediante cálculos previos que solo dependen de  $p$  y no, por ejemplo, del subgrupo elegido o de su generador. Por tanto, ello obliga a requerir que la longitud del número primo  $p$  sea de, al menos, 3000 bits y que la longitud del número primo  $q$  sea de, al menos, 250 bits.

299. Nota 46 [**Longitud de clave en ECIES**] Dado que para ECIES debe verificarse el mismo requisito que el señalado en la Nota 45 con relación a la dificultad de resolver el DLP en el subgrupo generado por el punto de la curva considerado con un orden dado, la longitud de este orden debe ser, al menos, de 250 bits.

300. Nota 47 [**Distribución uniforme para DLIES y ECIES**] Tanto DLIES como ECIES son algoritmos probabilísticos dado que en ambos casos se seleccionan valores aleatorios pertenecientes a determinados intervalos. Por ello, tales valores deben ser elegidos siguiendo una distribución uniforme.

301. Nota 48 [**CRYSTALS-Kyber un KEM postcuántico**] El mecanismo de encapsulamiento de claves Kyber ha sido seleccionados por el NIST en 2022 [171] en la convocatoria de estándares seguros contra la potencia de los ordenadores cuánticos. Por este motivo, este KEM es especialmente recomendado.

302. Nota 49 [**Implementación de CRYSTALS-Kyber**] Se recomienda no implementar este algoritmo hasta que el NIST publique el estándar definitivo, dado que hasta entonces, el algoritmo puede sufrir variaciones.

303. Nota 50 [**FrodoKEM un KEM postcuántico**] El mecanismo de encapsulamiento de claves FrodoKEM fue seleccionado como propuesta alternativa por el NIST en la tercera ronda [168] de su convocatoria y no ha pasado a la cuarta ronda. Sin embargo, el CCN considera que este KEM, basado en retículos no estructurados, debe considerarse como autorizado dado que se encuentra entre los KEM con opciones muy conservadoras en cuanto a seguridad.

## 4. PROTOCOLOS CRIPTOGRÁFICOS

304. El protocolo TLS o protocolo de seguridad de la capa de transporte, tiene como objetivo la protección de las comunicaciones llevadas a cabo a través de Internet mediante la configuración de los canales que emplean cliente y servidor.

### 4.1. TLS (§10.5)

305. El protocolo de seguridad en la capa de transporte o protocolo TLS (*Transport Layer Security*), anteriormente denominado protocolo de la Capa de conexión segura o protocolo SSL<sup>8</sup> (*Secure Socket Layer*), es un protocolo que permite proteger las comunicaciones que se realizan a través de Internet [58], como puede ser, por ejemplo, la conexión a través del protocolo seguro de transferencia de hipertexto o HTTPS (*Hypertext Transfer Protocol Secure*) o el protocolo seguro de transferencia de archivos o FTPS (*File Transfer Protocol Secure*). En definitiva, el protocolo TLS permite configurar y utilizar canales seguros entre las dos partes de una conexión: cliente y servidor [15], [40].

306. Nota 51 [**Versiones de SSL**] Las versiones v2 y v3 de SSL no están recomendadas.

307. Nota 52 [**Versiones de TLS**] Las versiones 1.0 y 1.1 de TLS no están recomendadas.

308. No obstante, antes de que se puedan transmitir los datos, se debe establecer un canal o conexión segura entre el cliente y el servidor. Este proceso se denomina protocolo de enlace o *handshake* y es una parte importante del protocolo TLS. En este protocolo, el cliente y el servidor acuerdan

- Los algoritmos criptográficos que emplearán para el cifrado de datos, la protección de la integridad, el acuerdo de claves y, si es necesario, la autenticación entre las partes, ya sea unilateral (en general, el servidor se autentica ante el cliente) o bilateral (ambas partes se autentican entre sí).
- Un secreto compartido, es decir, un secreto maestro, del que se derivarán las claves de sesión que usarán posteriormente para la protección de la integridad y para el cifrado de datos.

<sup>8</sup>El protocolo SSL no está recomendado por haber quedado obsoleto y ser vulnerable [22]

309. Como hemos mencionado, el protocolo TLS permite conexiones no autenticadas o autenticadas unilateralmente (como sucede habitualmente con las conexiones HTTPS, que solo se autentican en el lado del servidor). En todo caso, son los desarrolladores quienes deben considerar si es necesaria una autenticación adicional en la capa de aplicación (caso del acceso a las cuentas bancarias, por ejemplo). En el caso de conexiones críticas, se recomienda la autenticación mediante dos factores (algo que se conoce y algo que se tiene).
310. Finalmente, el protocolo de registro utiliza el resultado de los datos de sesión establecidos para proteger la confidencialidad e integridad de las comunicaciones posteriores, por ejemplo, los datos de la aplicación o la actualización de los datos de la sesión. Los mensajes transmitidos a través del protocolo TLS se denominan registros.
311. Con relación a las diferentes versiones de los protocolos SSL y TLS disponibles, es importante señalar que ninguna de las versiones del protocolo SSL debe utilizarse, ya sea la v2 [65, 204] o la v3 [67, 22] (la versión v1 no fue publicada). Por su parte, TLS 1.0 es un desarrollo adicional directo de SSL 3.0 [56] por lo que tampoco debe ser utilizado. También están disponibles para el TLS las versiones 1.1 [57], 1.2 [58] y 1.3 [183], pero solo las versiones TLS 1.2 y TLS 1.3 están autorizadas para su uso.
312. Las especificaciones propias de TLS 1.3 han modificado la estructura genérica del protocolo TLS. De hecho, el TLS 1.3 solo requiere un viaje de ida y vuelta (*Round-Trip Time*) para completar el protocolo de enlace, mientras que las versiones anteriores requerían dos. Esta modificación permite al cliente transmitir datos de la aplicación desde el tercer paquete transmitido. La reducción en el número de intercambios requeridos se debe a la eliminación de ciertos mensajes presentes en versiones anteriores. Así, se han eliminado los mensajes de señalización `ChangeCipherSpec` y `ServerHelloDone`, junto con los mensajes `ClientKeyExchange` y `ServerKeyExchange` utilizados para intercambiar valores públicos del protocolo DH. Con estas modificaciones, el protocolo TLS 1.3 es como sigue:
1. El cliente inicia una solicitud enviando un mensaje del tipo `ClientHello`, que contiene las suites criptográficas que admite y sus extensiones.
  2. El servidor responde con un `ServerHello` que contiene la suite y las extensiones seleccionadas necesarias para el intercambio de claves.
  3. El servidor envía un mensaje `EncryptedExtensions`, que contiene las otras extensiones (no necesarias para el intercambio de claves criptográficas) para esta sesión de las que no dependen los parámetros criptográficos.
  4. El servidor envía un mensaje de `Certificate`, que contiene, en particular, su clave pública en un certificado digital.

5. El servidor se autentica ante el cliente transmitiendo CertificateVerify que contiene datos firmados con la clave privada correspondiente a la clave pública anterior.
6. El servidor envía un Finished.
7. El cliente por su parte envía un Finished.

313. ClientHello contiene una lista de parámetros criptográficos que el cliente puede utilizar durante la sesión, de modo que el servidor selecciona los parámetros criptográficos para dicha sesión, después de compararlos con los que él acepta. Esta selección afecta a la forma en que se utilizarán las claves criptográficas para proteger los registros intercambiados después del protocolo de enlace, que transportan los datos de la aplicación. El propio procedimiento de negociación de claves se modifica de acuerdo con los parámetros adoptados. Los mecanismos criptográficos negociados durante el protocolo de enlace son:

- Un mecanismo de intercambio de claves.
- Un mecanismo de autenticación, simétrico o asimétrico, de la fase de intercambio de claves.
- Mecanismos que garantizan la confidencialidad e integridad de los datos intercambiados después del protocolo de enlace:
  - Puede ser mediante un modo de cifrado integrado que ofrece simultáneamente cifrado e integridad.
  - O como la composición de un algoritmo de cifrado y una función resumen utilizada en modo HMAC.
- Una función resumen empleada para la derivación de claves y en los mecanismos de autenticación asimétrica.

314. En la Tabla 4.1 se muestran las versiones recomendadas del protocolo TLS de comunicaciones.

Protocolo	Rec./Her.	Referencias	Notas
TLS 1.3	R	[183]	53
TLS 1.2	R	[58]	53

**Tabla 4.1: Versiones del protocolo TLS autorizadas**

- Nota 53 [Versiones de TLS] Siempre que sea posible, se recomienda utilizar la versión 1.3 de TLS antes que la versión 1.2; no obstante, esta última también está aceptada siempre que se sigan las recomendaciones de esta guía. Así pues, no están permitidas las versiones v2 y v3 de SSL y las versiones 1.0 y 1.1 de TLS. Además, debería preferirse el uso de software que no admita ninguna de estas versiones.
- 315.
316. Se recomienda utilizar una suite criptográfica que ofrezca *Perfect Forward Secrecy* (PFS), es decir, secreto perfecto persistente.
317. PFS es una característica de los protocolos criptográficos que asegura que incluso si la clave privada a largo plazo se ve comprometida, las claves de sesión y las claves de sesión anteriores aún permanecen secretas. En general, esta propiedad se puede obtener mediante el uso de claves públicas efímeras.
318. **Acuerdo de clave:** Con relación a los mecanismos de acuerdo de clave, se debe garantizar la propiedad de confidencialidad persistente, lo que requiere de una suite criptográfica basada en un intercambio Diffie-Hellman con claves efímeras, esto es, claves generadas en cada nueva sesión (denotados como DHE o EC-DHE).
319. Como ya se ha mencionado, el TLS 1.3 ha evolucionado dando lugar a la posibilidad de que el cliente negocie los grupos DHE y EC-DHE. En el caso de DHE, la seguridad del intercambio está ligada, como se sabe, al orden del grupo multiplicativo. De hecho, el ataque [5] mostró la debilidad de grupos de 512 bits por lo que tampoco los grupos de 1024 son recomendables y, así, se sugiere el uso de grupos de 3072 bits o más (se aceptan grupos de 2048 bits para la protección de datos, solo hasta 2030).
320. Los grupos multiplicativos autorizados son los definidos en [76] y se listan en la Tabla 4.4 para el protocolo TLS 1.3 y en la Tabla 4.11 para el protocolo TLS 1.2. En el caso de EC-DHE, se deben utilizar grupos cuyos órdenes sean múltiplos de un número primo mayor de, al menos, 256 bits.
321. **Autenticación:** En versiones anteriores a la 1.3 del TLS, se podían utilizar diferentes mecanismos de intercambio de claves, pero no todos ellos requerían la autenticación del servidor, por lo que podían ser atacados por el ataque del MitM. Para evitar esta debilidad, es indispensable la autenticación del servidor ante el cliente mediante un mecanismo asimétrico. Para la autenticación se prefieren los métodos basados en EC-DSA. Ahora bien, debido a que el algoritmo de firma está vinculado al certificado del servidor, es posible que la autenticación basada en EC-DSA no pueda llevarse a cabo, por lo que en este caso se tolera la autenticación del servidor con el algoritmo RSA. En el proceso de autenticación del servidor se desaconsejan las alternativas anónimas o las basadas en el uso de certificados sin procesar definidos en [209]. Las funciones resumen necesarias en los procesos de autenticación se comentarán más tarde.

322. **Cifrados simétricos:** Los mecanismos de autenticación simétrica son, en general, más complejos de implementar y de mantener por lo que solo se deberían utilizar en entornos controlados.
323. El secreto compartido obtenido en el intercambio de clave permite que ambas partes (cliente y servidor) determinen una clave simétrica con la que proteger la confidencialidad de la información intercambiada que sigue a la fase de negociación. El algoritmo de cifrado a emplear se fija en la selección de una suite criptográfica y el protocolo TLS permite utilizar tanto cifradores en bloque como en flujo. Tradicionalmente, el único modo de cifrado en flujo disponible en TLS era RC4. Sin embargo, después de publicarse algunas debilidades de este sistema de cifrado [7, 205] se ha prohibido su uso en este protocolo [177]. En lugar de emplear RC4, se ha propuesto el algoritmo de cifrado ChaCha20 [121], que se mantiene para TLS 1.3. ChaCha20 emplea claves de cifrado de 256 bits y hasta la fecha no se conoce ningún ataque a este algoritmo. En todo caso, se prefiere el uso de AES, ya sea con una clave de 128 o de 256 bits, dado que, hasta la fecha, no se conocen ataques prácticos que pongan en entredicho su seguridad.
324. **Cifrado de integridad:** Para evitar las debilidades de las versiones de TLS anteriores a la 1.2, TLS 1.2 introdujo la capacidad de utilizar modos de cifrado fuertes, proporcionando una función de cifrado combinada y una función de cálculo de patrón de integridad; por su parte, TLS 1.3 solo ofrece modos de cifrado combinados. Se han estandarizado las suites que ofrecen los modos de funcionamiento GCM y CCM, también se permite el modo ChaCha20\_Poly1305 [150]. Los modos combinados de GCM y ChaCha20\_Poly1305 requieren especial atención al administrar las claves de un solo uso (*nonces*). En estos dos modos, el cifrado de cada registro requiere el uso de un *nonce* único durante la sesión y si no se garantiza la unicidad del mismo, la confidencialidad y la integridad de los datos intercambiados puede quedar comprometida. En el caso del TLS 1.2 no se especifica cómo se debe generar este *nonce*; mientras que en el TLS 1.3 la construcción de estos *nonces* sí se ha incorporado a sus especificaciones.
325. **Funciones resumen:** Como ya se ha mencionado, el protocolo TLS necesita una función resumen para la derivación de la clave secreta, para calcular la firma durante la autenticación asimétrica y para determinar los patrones de integridad cuando se utiliza el modo de cifrado CBC. Las versiones TLS 1.2 y TLS 1.3 usan SHA-256 o SHA-384 dependiendo del paquete criptográfico seleccionado. En el caso de que no se use un modo de cifrado fuerte, el patrón de integridad se calcula mediante el modo HMAC. A pesar de que en las versiones de TLS se permite el uso de diferentes funciones resumen, solo las funciones de la familia SHA-2 y SHA-3 deben ser utilizadas.

#### 4.1.1. TLS VERSION 1.3

326. A continuación se presentan las tablas que contienen los diferentes conjuntos de mecanismos criptográficos autorizados para la versión 1.3 del protocolo TLS.

327. La Tabla 4.2 presenta la suite de cifradores para la versión 1.3 de TLS. La convención que se sigue para esta tabla es TLS\_ENC\_Long\_Mod\_Hash, siendo ENC el sistema de cifrado, Long es la longitud de la clave considerada, Mod es el modo de operación del cifrado y Hash hace referencia a la función resumen considerada.

Código	Suites criptográficas	Rec./Her.	Referencias	Notas
0x1302	TLS_AES_256_GCM_SHA384	R	[183]	
0x1301	TLS_AES_128_GCM_SHA256	R	[183]	
0x1304	TLS_AES_128_CCM_SHA256	R	[183]	
0x1303	TLS_CHACHA20_POLY1305_SHA256	R	[121]	

**Tabla 4.2: Suites criptográficas autorizadas para el protocolo TLS 1.3**

328. Además del acuerdo de clave de Diffie-Hellman sobre cuerpos finitos o curvas elípticas, TLS 1.3 ofrece modos de protocolo de enlace adicionales utilizando claves precompartidas o PSK (*Pre-Shared Key*). En este contexto, las PSK se refieren a claves que se proporcionan fuera de banda o al material de claves que se ha establecido en una sesión anterior a través del mecanismo de ticket de sesión. En la Tabla 4.3 se presentan los modos PSK recomendados para TLS 1.3.

Código	Modo PSK	Rec./Her.	Referencias	Notas
0x0000	psk_ke	H[2026]	[183]	54, 55
0x0001	psk_dhe_ke	R	[183]	55

**Tabla 4.3: Modos de clave precompartida recomendados para el protocolo TLS 1.3**

329. Nota 54 [Modo psk\_ke de PSK] El modo PSK psk\_ke no ofrece secreto perfecto persistente. Este modo solo debe usarse en aplicaciones especiales después de consultar a un experto.

330. Nota 55 [Datos 0-RTT] El protocolo TLS 1.3 ofrece una opción para incluir datos de aplicación ya en el primer mensaje de un protocolo de enlace PSK, son los datos llamados de tiempo cero de ida y vuelta o datos 0-RTT (*zero Round-Trip Time data*). Estos datos no están protegidos contra ataques de reproducción por lo que no se recomienda enviar o aceptar datos de este tipo.

331. Existen algunas extensiones para el protocolo TLS 1.3 que se mostrarán a continuación y que tienen que ver con los grupos que se pueden utilizar, los algoritmos de firma, etc.
332. En el TLS 1.3, el cliente y el servidor pueden usar la extensión “supported\_groups” para informarse mutuamente sobre los grupos Diffie-Hellman que desean usar para (EC)DHE. En la Tabla 4.4 se listan los grupos Diffie-Hellman recomendados.

Código	Grupo DH	Rec./Her.	Referencias	Notas
0x0100	ffdhe2048	H[2025]	[76]	
0x0101	ffdhe3072	R	[76]	
0x0102	ffdhe4096	R	[76]	
0x0017	secp256r1 (P-256)	R	[148]	
0x0018	secp384r1 (P-384)	R	[148]	
0x001F	brainpoolP256r1tls13	R	[36]	
0x0020	brainpoolP384r1tls13	R	[36]	
0x0021	brainpoolP512r1tls13	R	[36]	

**Tabla 4.4: Grupos de Diffie-Hellman recomendados para el protocolo TLS 1.3**

333. En TLS 1.3, el cliente y el servidor pueden usar las extensiones “signature\_algorithms” y “signature\_algorithms\_cert” para informarse mutuamente sobre los algoritmos de firma que desean usar para la autenticación basada en certificados. La extensión “signature\_algorithms” hace referencia a firmas que son generadas por el cliente o servidor para su mensaje CertificateVerify; mientras que la extensión “signature\_algorithms\_cert” se refiere a firmas en certificados. En la Tabla 4.5 se listan los algoritmos de firma recomendados para la extensión “signature\_algorithms” y en la Tabla 4.6 se presentan los algoritmos de firma recomendados para la extensión “signature\_algorithms\_cert”.

Código	Algoritmo de firma	Rec./Her.	Referencias	Notas
0x0804	rsa_pss_rsae_sha256	R	[183]	
0x0805	rsa_pss_rsae_sha384	R	[183]	
0x0806	rsa_pss_rsae_sha512	R	[183]	
0x0809	rsa_pss_pss_sha256	R	[183]	
0x080A	rsa_pss_pss_sha384	R	[183]	
0x080B	rsa_pss_pss_sha512	R	[183]	
0x0403	ecdsa_secp256r1_sha256	R	[183]	
0x0503	ecdsa_secp384r1_sha384	R	[183]	
0x081A	ecdsa_brainpoolP256r1tls13_sha256	R	[36]	
0x081B	ecdsa_brainpoolP384r1tls13_sha384	R	[36]	
0x081C	ecdsa_brainpoolP512r1tls13_sha512	R	[36]	

Tabla 4.5: Algoritmos de firma (cliente/servidor) recomendados para el protocolo TLS 1.3

Código	Funciones resumen	Rec./Her.	Referencias	Notas
0x0401	rsa_pkcs1_sha256	H[2025]	[183]	56
0x0501	rsa_pkcs1_sha384	H[2025]	[183]	56
0x0601	rsa_pkcs1_sha512	H[2025]	[183]	56
0x0804	rsa_pss_rsae_sha256	R	[183]	
0x0805	rsa_pss_rsae_sha384	R	[183]	
0x0806	rsa_pss_rsae_sha512	R	[183]	
0x0809	rsa_pss_pss_sha256	R	[183]	
0x080A	rsa_pss_pss_sha384	R	[183]	
0x080B	rsa_pss_pss_sha512	R	[183]	
0x0403	ecdsa_secp256r1_sha256	R	[183]	
0x0503	ecdsa_secp384r1_sha384	R	[183]	
0x081A	ecdsa_brainpoolP256r1tls13_sha256	R	[36]	
0x081B	ecdsa_brainpoolP384r1tls13_sha384	R	[36]	
0x081C	ecdsa_brainpoolP512r1tls13_sha512	R	[36]	

Tabla 4.6: Algoritmos de firma (en certificados) recomendados para el protocolo TLS 1.3

334. Nota 56 [Firma rsa\_pkcs1\_\*] El uso de los algoritmos de firma rsa\_pkcs1\_\* (códigos 0x0401, 0x0501 y 0x0601) solo se admite hasta 2025, porque utilizan el esquema de relleno PKCS # 1 v1.5.

#### 4.1.2. TLS VERSION 1.2

335. En la versión 1.2 de TLS, los mecanismos criptográficos de una conexión se definen mediante una suite criptográfica que especifica un mecanismo de acuerdo de claves

(con autenticación) para el protocolo de enlace, un algoritmo de cifrado autenticado para el protocolo de registro y una función resumen para el proceso de derivación de claves. Dependiendo de la suite criptográfica, también se debe especificar un grupo para el protocolo de Diffie-Hellman, ya sea en un subgrupo de un cuerpo finito o en una curva elíptica sobre un cuerpo finito, y un algoritmo de firma para el acuerdo de claves.

336. Para estas suites criptográficas se suele utilizar la siguiente notación: `TLS_AKE_WITH_ENC_Hash`, donde AKE denota un mecanismo de acuerdo de clave (con autenticación), ENC es un algoritmo de cifrado que incluye la longitud de su clave y el modo de operación, y Hash hace referencia a la función resumen. La función resumen se utiliza para un HMAC que se emplea en una función pseudo-aleatoria (PRF o *Pseudo-Random Function*), la cual es usada para la derivación de claves. En el caso de que el sistema de cifrado considerado utilice el modo de operación CCM, no se indica ninguna función resumen y se da por hecho que para la PRF usa SHA-256. Si ENC no fuera un algoritmo AEAD, entonces el HMAC también se utiliza para la protección de la integridad en el protocolo de registro.
337. Se muestran a continuación las tablas que contienen las diferentes suites criptográficas autorizadas para el protocolo TLS 1.2.
338. En las Tablas 4.7 y 4.8 se muestran las suites criptográficas recomendadas para la versión 1.2 de TLS en los casos en los que el servidor disponga de un certificado con la clave pública ECDSA o RSA, respectivamente.

Código	Suites criptográficas	Rec./He	Referencias
0xC02C	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	R	[182]
0xC02B	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	R	[182]
0xC0AD	TLS_ECDHE_ECDSA_WITH_AES_256_CCM	R	[134]
0xC0AC	TLS_ECDHE_ECDSA_WITH_AES_128_CCM	R	[134]
0xCCA9	TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256	R	[121]

**Tabla 4.7: Suites criptográficas recomendadas para TLS 1.2 con un servidor que disponga de un certificado con la clave pública EC-DNA**

Código	Suites criptográficas	Rec./Her.	Referencias
0xC030	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	H	[182]
0xC02F	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	H	[182]
0xCCA8	TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	H	[121]

**Tabla 4.8: Suites criptográficas heredadas para TLS 1.2 con un servidor que disponga de un certificado con la clave pública RSA**

339. Cuando una de las dos partes en una comunicación no es la dominante, no siempre es posible negociar una sesión TLS con una de las suites criptográficas anteriores. Si se ha identificado una gran necesidad de compatibilidad, se pueden adoptar otras

suites, con detrimento de la seguridad en las comunicaciones. En este caso, es necesario evaluar el perfil de los servidores o de los clientes interesados y adoptar solo las suites que se consideren esenciales para llevar a cabo las funciones de la aplicación consideradas.

340. La Tabla 4.9 lista las suites criptográficas recomendadas para uso general con TLS 1.2 cuando no hay soporte ECC o modo de cifrado autenticado.

Código	Suites criptográficas	Rec./Her.	Referencias	Notas
0xC024	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	H[2025]	[182]	57
0xC023	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	H[2025]	[182]	57
0xC028	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	H[2025]	[182]	57
0xC027	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	H[2025]	[182]	57
0x009F	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	H	[186]	58
0x009E	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	H	[186]	58
0xC09F	TLS_DHE_RSA_WITH_AES_256_CCM	H	[133]	58
0xC09E	TLS_DHE_RSA_WITH_AES_128_CCM	H	[133]	58
0x006B	TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	H[2025]	[58]	57,58
0x0067	TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	H[2025]	[58]	57,58
0x009D	TLS_RSA_WITH_AES_256_GCM_SHA384	H		59
0x009C	TLS_RSA_WITH_AES_128_GCM_SHA256	H		59
0xC09D	TLS_RSA_WITH_AES_256_CCM	H		59
0xC09C	TLS_RSA_WITH_AES_128_CCM	H		59
0x003D	TLS_RSA_WITH_AES_256_CBC_SHA256	H[2025]		57,59
0x003C	TLS_RSA_WITH_AES_128_CBC_SHA256	H[2025]		57,59
0xCCAA	TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256	H	[121]	
0x006A	TLS_DHE_DSS_WITH_AES_256_CBC_SHA256	H	[58]	
0x00A3	TLS_DHE_DSS_WITH_AES_256_GCM_SHA384	H	[186]	

**Tabla 4.9: Suites criptográficas recomendadas para TLS 1.2 cuando no hay soporte ECC o modo de cifrado autenticado**

341. Nota 57 [TLS Cifrado y luego MAC] Las suites criptográficas TLS cuyos mecanismos de cifrado se basan en CBC deben utilizarse junto con la extensión `encrypt_then_mac`.

342. Nota 58 [TLS con DHE] Cuando se utiliza el intercambio de claves DHE en TLS, el servidor impone los parámetros del grupo al cliente y como la validación de estos parámetros por parte del cliente puede ser problemática, se debe preferir el uso de EC-DHE para el intercambio de claves, dado que en este caso, los parámetros de grupo se negocian en el protocolo de enlace.

343. Nota 59 [TLS con RSA] El intercambio de clave con RSA no ofrece PFS.

344. Si los datos adicionales que se han intercambiado de antemano se van a incorporar en el acuerdo de clave, se pueden utilizar suites criptográficas con una PSK. En general, se recomienda utilizar suites criptográficas para los que se incorporan al acuerdo de claves más claves efímeras o números aleatorios previamente intercambiados, además de la clave precompartida. En el caso del protocolo TLS 1.2 cuando se utilizan suites criptográficas con una clave previamente compartida no se recomienda el uso de suites criptográficas de tipo TLS\_PSK\_\*, es decir, sin claves efímeras o números aleatorios adicionales, porque la seguridad de la conexión se basa únicamente en la entropía y la confidencialidad de las claves previamente compartidas para estos conjuntos de cifrado.

345. En la Tabla 4.10 se incluyen las suites criptográficas con PSK que se recomiendan.

Código	Suites criptográficas	Rec./Her.	Referencias	Notas
0xC037	TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256	R	[81]	
0xC038	TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA384	R	[81]	
0xD001	TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256	R	[131]	
0xD002	TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384	R	[131]	
0xD005	TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256	R	[131]	
0xCCAD	TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256	R	[121]	
0x00B2	TLS_DHE_PSK_WITH_AES_128_CBC_SHA256	R	[19]	
0x00B3	TLS_DHE_PSK_WITH_AES_256_CBC_SHA384	R	[19]	
0x00AA	TLS_DHE_PSK_WITH_AES_128_GCM_SHA256	R	[19]	
0x00AB	TLS_DHE_PSK_WITH_AES_256_GCM_SHA384	R	[19]	
0xC0A6	TLS_DHE_PSK_WITH_AES_128_CCM	R	[133]	
0xC0A7	TLS_DHE_PSK_WITH_AES_256_CCM	R	[133]	

**Tabla 4.10: Suites criptográficas recomendadas para TLS 1.2 con clave precompartida**

346. En cuanto a las extensiones para TLS 1.2, a continuación se presentan las recomendadas para esta versión del protocolo.

347. Se recomienda el uso de la extensión “supported\_groups” para los conjuntos de cifrado TLS\_DHE\_\* tan pronto como estén disponibles las implementaciones correspondientes. Así, en la Tabla 4.11 se listan los grupos Diffie-Hellman recomendados.

Código	Grupo DH	Rec./Her.	Referencias
0x0100	ffdhe2048	H[2025]	[76]
0x0101	ffdhe3072	R	[76]
0x0102	ffdhe4096	R	[76]
0x0017	secp256r1 (P-256)	R	[148]
0x0018	secp384r1 (P-384)	R	[148]
0x001A	brainpoolP256r1	R	[136]
0x001B	brainpoolP384r1	R	[136]
0x001C	brainpoolP512r1	R	[136]

**Tabla 4.11: Grupos de Diffie-Hellman recomendados para el protocolo TLS 1.2**

348. En TLS 1.2, el cliente puede usar la extensión “signature\_algorithms” [58] para informar al servidor sobre los algoritmos de firma que desea usar para acuerdos de clave y certificados. El algoritmo debe especificarse como una combinación de un algoritmo de firma y una función resumen. En la Tabla 4.12 se listan los algoritmos de firma recomendados y en la Tabla 4.13 las funciones resumen para tales algoritmos de firma.

Código	Algoritmo de firma	Rec./Her.	Referencias
0x0001	rsa	H[2025]	[58]
0x0002	dsa	R	[58]
0x0003	ecdsa	R	[58]

**Tabla 4.12: Algoritmos de firma recomendados para el protocolo TLS 1.2**

Código	Funciones resumen	Rec./Her.	Referencias
0x0004	sha256	R	[58]
0x0005	sha384	R	[58]
0x0006	sha512	R	[58]

**Tabla 4.13: Funciones resumen para el protocolo TLS 1.2**

#### 4.2. SSH (§10.6)

349. SSH (*Secure Shell*) es un protocolo criptográfico de seguridad [213] diseñado originalmente para sustituir a los protocolos de conexión remota inseguros como, por ejemplo, Telnet (la página oficial de este protocolo es <https://www.ssh.com/academy/ssh>). Este protocolo se puede utilizar para establecer un canal seguro dentro de una red insegura. Las aplicaciones más comunes del protocolo SSH son

iniciar sesión en un sistema remoto (inicio de sesión de línea de comando remoto) y ejecutar comandos o ejecutar aplicaciones en sistemas remotos [41].

350. La primera versión del protocolo SSH se conoce como SSH-1 [210], pero su uso no está recomendado. La versión que sí está recomendada es la versión 2, denotada por SSH-2 [213].
351. El protocolo SSH está formado por tres subprotocolos: Protocolo de capa de transporte, Protocolo de autenticación de usuario y Protocolo de conexión. El Protocolo de la capa de transporte [214] permite la autenticación del servidor, el cifrado, la protección de la integridad y, opcionalmente, la compresión de datos. Se basa en el protocolo TCP/IP. El Protocolo de autenticación de usuario [211] se utiliza para autenticar al usuario en el servidor y se basa en el Protocolo de la capa de transporte. Finalmente, el Protocolo de conexión [212] es el responsable de crear y administrar canales lógicos dentro del túnel cifrado y se basa en el Protocolo de autenticación de usuarios.

#### 4.2.1. ACUERDO DE CLAVE

352. Cuando se establece una conexión SSH se intercambian las claves con el fin de crear e intercambiar claves de sesión compartidas para la autenticación y el cifrado. El mecanismo de acuerdo de clave del protocolo SSH se basa en el de Diffie-Hellman. De hecho, hay varios grupos recomendados, todos ellos con la función SHA512. Los números primos y el generador de cada grupo están publicados en [116]. En el caso del acuerdo de clave con curvas elípticas, se emplea el mecanismo ECDH [202]. En la Tabla 4.14 se presentan las versiones recomendadas del protocolo SSH.

Protocolo	Rec./Her.	Referencias	Notas
DH-exchange-SHA256	R	[68, §4.2]	60, 61
DH-grupo15-SHA512	R	[116, §4]	61
DH-grupo16-SHA512	R	[116, §5]	61
DH-grupo17-SHA512	R	[116, §6]	61
DH-grupo18-SHA512	R	[116, §7]	61
ECDH-SHA2-*	R	[202, §6.3]	61, 62

Tabla 4.14: Versiones de acuerdos de clave para el protocolo SSH autorizadas

353. Nota 60 [Diffie-Hellman-group-exchange-SHA256] El tamaño del número primo a emplear,  $p$ , debe ser de, al menos, 3000 bits. Además, el orden del generador debe tener un tamaño de  $2^{250}$ , como mínimo. Finalmente,  $p$  debe ser un primo seguro. Recuérdese que un primo seguro,  $p$ , es un primo de la forma  $p = 1 + 2q$ , siendo  $q$  otro primo. Por tanto, el orden del generador  $g$  solo puede ser un divisor de  $p - 1 = 2 \cdot q$ , esto es, o es 2 o  $q$ . Esto significa que el tamaño de  $q$  debe ser mucho mayor que el mínimo recomendado de  $2^{250}$  para el orden de  $g$ .

354. Nota 61 [Renovación de la clave] Con el fin de más dificultar el ataque contra las claves de sesión, se recomienda renovar la clave utilizada en una conexión después de un cierto período de tiempo o una cierta cantidad de datos transmitidos. Con SSH este proceso lo pueden iniciar tanto el cliente como el servidor, para lo que es suficiente con enviar el mensaje SSH\_MSG\_KEXINIT. Así pues, siguiendo la recomendación de [214, Cap. 9], las claves de sesión se deben renovar después de una hora o después de que se haya transmitido un gigabyte (lo que ocurra primero).

355. Nota 62 [ECDH-SHA2-\*] El intercambio de claves ECDH se define mediante una familia de nombres de métodos. Cada nombre de método es la concatenación de la cadena ECDH-SHA2- con el parámetro de dominio de la correspondiente curva elíptica. Las curvas elípticas acordadas son las siguientes: por una parte, nistp256, nistp384 y nistp521 [151] y por otra secp256r1, secp384r1 y secp521r1 [192].

#### 4.2.2. CIFRADO

356. Durante el intercambio de claves, el cliente y el servidor acuerdan un algoritmo de cifrado y una clave de cifrado compartida. Los algoritmos de cifrado recomendados se muestran en la Tabla 4.15.

Protocolo	Rec./Her.	Referencias	Notas
AEAD_AES_128_GCM	R	[90, §6.1]	63, 64
AEAD_AES_256_GCM	R	[90, §6.2]	63, 64
AES128-CTR	R	[25, §4]	64
AES192-CTR	R	[25, §4]	64
AES256-CTR	R	[25, §4]	64

Tabla 4.15: Algoritmos de cifrado autorizados para el protocolo

357. Nota 63 [**Protección MAC**] Los algoritmos AEAD\_AES\_128\_GCM y AEAD\_AES\_256\_GCM ya incluyen la protección MAC en el modo GCM por lo que se deberían utilizar con preferencia a los restantes.

358. Nota 64 [**Declaración de seguridad**] En la medida de lo posible, deben utilizarse los algoritmos AEAD\_AES\_128\_GCM y AEAD\_AES\_256\_GCM puesto que existen declaraciones de seguridad comprobables para el modo GCM con respecto al objetivo de seguridad del cifrado autenticado. Si se escoge cifrado no autenticado (i.e., AES-CTR) será obligatorio utilizar alguna de las opciones recomendadas para proteger la integridad y autenticidad de los mensajes (i.e., HMAC-SHA2).

#### 4.2.3. INTEGRIDAD Y AUTENTICIDAD EN ORIGEN

359. La integridad y la autenticidad en origen se pueden llevar a cabo mediante la protección por MAC, para lo cual se recomiendan las funciones que se muestran en la Tabla 4.16.

Función	Rec./Her.	Referencias	Notas
HMAC-SHA1	H[2030]	[214, §6.4]	
HMAC-SHA2-256	R	[31, §2]	
HMAC-SHA2-512	R	[31, §2]	
AEAD_AES_128_GCM	R	[90]	
AEAD_AES_256_GCM	R	[90]	

Tabla 4.16: Funciones MAC autorizadas para el protocolo SSH

#### 4.2.4. AUTENTICACIÓN DEL SERVIDOR Y DEL CLIENTE

360. La autenticación de servidor SSH se lleva a cabo mediante criptografía asimétrica usando claves públicas o certificados. En la Tabla 4.17 se presentan los métodos recomendados para la autenticación del servidor en el protocolo SSH.

Método	Rec./Her.	Referencias	Notas
ECDSA-SHA2-*	R	[202, §3]	65
x509v3-ECDSA-SHA2-*	R	[91, §3.4]	66

Tabla 4.17: Métodos de autenticación del servidor autorizados para el protocolo SSH

361. Nota 65 [ECDSA-SHA2-\*] El protocolo ECDSA se define mediante una familia de nombres de métodos. Cada nombre de método es la concatenación de la cadena ECDSA-SHA2- con el parámetro de dominio de la correspondiente curva elíptica. Las curvas elípticas acordadas son las siguientes: por una parte, nistp256, nistp384 y nistp521 [151] y por otra secp256r1, secp384r1 y secp521r1 [192].

362. Nota 66 [x509v3-ECDSA-\*] El protocolo x509v3-ECDSA se define mediante una familia de nombres de métodos. Cada nombre de método es la concatenación de la cadena x509v3-ECDSA-SHA2- con el parámetro de dominio de la correspondiente curva elíptica. Las curvas elípticas acordadas son las siguientes: nistp256, nistp384 y nistp521 [151] y secp256r1, secp384r1 y secp521r1 [192].

#### 4.3. IPSEC CON IKEV2 (§10.7)

363. En esta sección se tratará la seguridad del protocolo de Internet o IPsec (*Internet Protocol Security*) [208] y el protocolo de intercambio de claves de Internet o IKE (*Internet Key Exchange*), cuya versión 2 se denota por IKEv2 [110]. En esta guía no se considera la versión 1 de este protocolo (IKEv1) dado que solo está autorizada en modo herencia.

364. IPsec es un estándar que proporciona seguridad a nivel de capa de red del Protocolo de Internet o IP (*Internet Protocol*) en la pila del protocolo TCP/IP (*Transmission Control Protocol/Internet Protocol*). A diferencia de los protocolos TLS (ver §4.1) y SSH (ver §4.2), IPsec proporciona seguridad en las capas superiores, como la de aplicación.

365. El uso más importante de IPsec es el de crear redes privadas virtuales o VPN (*Virtual Private Network*), esto es, establecer canales de comunicación seguros mediante redes IP que no son seguras.
366. IPsec se puede desplegar en el modo túnel y en el de transporte. En el modo túnel, el paquete IP se protege de modo criptográfico al completo y en modo transporte, la cabecera del paquete IP original se conserva y se le añaden algunos campos de seguridad.

367. Nota 67 **[Modo túnel-Modo transporte]** Se puede afirmar que el modo túnel ofrece más seguridad que el modo transporte, por lo que se deberá usar aquella configuración antes que esta. El modo transporte está justificado solo si el tamaño del paquete presenta problemas debido a las restricciones que imponga la red.

368. En la Tabla 4.18 se muestran las versiones del protocolo IPsec autorizadas.

Protocolo	Rec./Her.	Referencias	Notas
IPsec	R	[208]	68
IKEv2	R	[110], [149]	
ESP	R	[114]	68

Tabla 4.18: Versiones autorizadas del protocolo IPsec

369. Nota 68 **[Ataques a IPsec]** Existen ataques conocidos cuando se implementa IPsec en cualquier configuración *MAC-then-Encrypt* (como, por ejemplo, si se usa AH en modo transporte antes de un ESP sólo cifrado en modo túnel). Sin embargo, no se conocen ataques a IPsec si se usa ESP sólo cifrado seguido de AH o ESP con autenticidad e integridad sin que le siga otro AH. Por lo tanto, se recomienda el uso de ESP siempre con las opciones de protección de integridad y autenticidad, además de confidencialidad.

#### 4.3.1. ACUERDO DE CLAVES

370. El mecanismo de acuerdo de clave empleado en el protocolo IKEv2 se basa en el de Diffie-Hellman. Se consideran varios grupos de tipo Diffie-Hellman para ser empleados en IKEv2, ya sean de exponenciación modular o MODP [116], basados en curva elípticas modulo un primo, ECP (curva elípticas modulo un primo) [70] o curvas Brainpool [147].

371. En la Tabla 4.19 se muestran los los grupos de DH y ECDH acordados.

Grupo (EC)DH	Rec./Her.	Referencias	Notas
3072-bit MODP	R	[116]	
4096-bit MODP	R	[116]	
6144-bit MODP	R	[116]	
8192-bit MODP	R	[116]	
256-bit aleatorio ECP	R	[70]	
384-bit aleatorio ECP	R	[70]	
521-bit aleatorio ECP	R	[70]	
brainpoolP256r1	R	[137]	
brainpoolP384r1	R	[137]	
brainpoolP512r1	R	[137]	
x25519	R	[147]	
x448	R	[147]	

Tabla 4.19: Grupos de tipo DH y ECDH acordados por IKEv2

#### 4.3.2. CIFRADO

372. Las propuestas para los esquemas de cifrado IKEv2 y ESP pueden incluir tanto esquemas de cifrado clásico como esquemas AEAD. Atendiendo a las recomendaciones señaladas en las subsecciones 2.2.1 y 2.2.5, se acuerdan los esquemas de cifrado o esquemas de cifrado autenticado que se incluyen en las Tablas 2.4 y 2.8.

#### 4.3.3. INTEGRIDAD Y AUTENTICACIÓN

373. Los protocolos IKEv2 y ESP utilizan mecanismos MAC para la verificación de la integridad y la autenticación de origen. Los mecanismos MAC autorizados para uso recomendado son los basados en esquemas AES-CMAC, AES-GMAC y HMAC-SHA2. Por su parte, los esquemas HMAC-SHA2, cuando se utilizan en IPsec como mecanismos de integridad y autenticidad, la longitud de la clave sera fija en función del tamaño del valor hash de salida, y se realiza un truncado de la salida tal y como se comenta en las notas de la Tabla 4.20.

Esquema (H)MAC	Rec./Her.	Referencias	Notas
HMAC-SHA2-256_128	R	[112]	69
HMAC-SHA2-384_192	R	[112]	69
HMAC-SHA2-512_256	R	[112]	69
AES-CMAC-96	R	[201]	
AES-GMAC	R	[53]	

Tabla 4.20: Esquemas MAC y HMAC autorizados para IKEv2 y ESP

374. Nota 69 [HMAC-SHA-XXX-YYY] Cada uno de estos esquemas utiliza una longitud de clave fija de XXX bits, truncando la salida a YYY bits.

#### 4.3.4. FUNCIONES PSEUDO-ALEATORIAS

375. Como ya se ha mencionado, las claves para el cifrado y la autenticación de mensajes se derivan del secreto compartido obtenido con el intercambio Diffie-Hellman, utilizando la PRF negociada. Estas funciones para IKEv2 son similares a los mecanismos MAC señalados en la Tabla 4.20, con la excepción de que se anula la restricción de claves de tamaño fijo y se elimina el truncado, debido al tamaño de la salida de la función. En la Tabla 4.21 se muestran las PRF autorizadas.

Esquema (H)MAC	Rec./Her.	Referencias	Notas
HMAC-SHA2-256	R	[112]	
HMAC-SHA2-384	R	[112]	
HMAC-SHA2-512	R	[112]	
AES128-CMAC	R	[200]	

Tabla 4.21: Funciones pseudo-aleatorias autorizadas para IKEv2

#### 4.3.5. MECANISMOS DE AUTENTICACIÓN

376. El mecanismo de autenticación para IKEv2 es la firma electrónica, recomendándose el uso de certificados X.509. Los esquemas de firma autorizados para uso con IKEv2 se basan en ECDSA y RSA y se muestran en la Tabla 4.22.

Esquema de firma	Rec./Her.	Referencias	Notas
RSA (RSASSA-PSS)	R	[188]	70
ECDSA-SHA-256 y P-256	R	[69]	71
ECDSA-SHA-384 y P-384	R	[69]	71
ECDSA-SHA-512 y P-521	R	[69]	71
ECDSA-256-BrainpoolP256r1	R	[117]	71
ECDSA-384-BrainpoolP384r1	R	[117]	71
ECDSA-512-BrainpoolP512r1	R	[117]	71
ECGDSA-256-BrainpoolP256r1	R	[117]	
ECGDSA-384-BrainpoolP384r1	R	[117]	
ECGDSA-512-BrainpoolP512r1	R	[117]	

Tabla 4.22: Esquemas de firma acordados para IKEv2

377. Nota 70 (**RSASSA-PSS**) Este esquema solo debe utilizarse con PSS [105, §8 y §9.1] y con una función de la familia SHA-2.

378. Nota 71 (**ECDSA-\***) Cuando se elabora una firma de tipo ECDSA, debe tenerse en cuenta que el *nonce* utilizado en el protocolo se selecciona aleatoriamente y se distribuye uniformemente dentro del intervalo  $[1, q - 1]$ , siendo  $q$  el orden del punto base de la curva elíptica considerada.

## 5. GENERADORES DE NÚMEROS ALEATORIOS

379. Es bien sabido que muchas aplicaciones criptográficas requieren números aleatorios, como por ejemplo la generación de claves, ya sea para ser utilizadas un largo periodo de tiempo (asimétricas), uno corto (simétricas), para una sola vez (claves efímeras y *nonces*); o para generar determinados parámetros del sistema (desafíos, etc.). Por ello, es básico establecer los tipos y propiedades que deben verificar los generadores de números aleatorios.

### 5.1. GENERADORES DE NÚMEROS ALEATORIOS

380. El objetivo a la hora de generar números aleatorios suele ser el de producir bits (0 y 1) de modo que se distribuyan uniformemente en el conjunto  $\{0, 1\}^n$  (ver [14], [197], [17], [42]). Esta generación de bits puede transformarse de modo inmediato a la generación de números. Además, la mayoría de las aplicaciones criptográficas precisa de determinado grado de imprevisibilidad y que los bits o números generados sean secretos. De hecho, a los generadores que se puedan usar se les exige la propiedad de que si un adversario llegara a conocer largas subsecuencias de los números aleatorios generados, no debería poder determinar predecesores o sucesores de la subsecuencia conocida. Dicho de otro modo, conocida determinada subsecuencia de números, la probabilidad de conocer el siguiente o el anterior número de la subsecuencia no debería ser mayor de  $1/2$ . Así pues, en las aplicaciones criptográficas es fundamental utilizar generadores de números aleatorios fuertes y seguros.

381. Dos fuentes de gran interés para la elección y estudio de los generadores de números aleatorios o RNG (*Random Number Generator*)<sup>9</sup> pertenecen al esquema alemán, conocidas como AIS 31 [39] para el caso de los generadores físicos de números aleatorios o PTRNG (*Physical True Random Number Generator*) y AIS 20 [38], para los generadores deterministas de números aleatorios o DRNG (*Deterministic Random Number Generator*). Para ambas fuentes, es de interés el anexo de Killman y Schindler [115], que define la clases de funcionalidad para generadores físicos de números aleatorios PTG.1–PTG.3, para generadores deterministas de números aleatorios DRG.1–DRG.4 y para generadores de números aleatorios no físicos no deterministas NTG.1.

<sup>9</sup>Cuando se habla de generadores de bits aleatorios en lugar de números aleatorios, estos se denotan por RBG (*Random Bit Generator*). Suele ser indiferente hacer referencia a un tipo o a otro de generador puesto que ambas generaciones pueden considerarse equivalentes: todo número se puede transformar en una colección de bits y viceversa.

## 5.2. GENERADORES FÍSICOS DE NÚMEROS ALEATORIOS (§10.8)

382. Los generadores físicos de números aleatorios utilizan hardware dedicado (v.gr. un circuito electrónico) como generador de números realmente aleatorios o TRNG (*True Random Number Generator*), es decir, números aleatorios impredecibles. En general se hace uso del comportamiento impredecible del hardware empleado, de modo que a la postre, la entropía de la señal se debe a un nivel físico o a las influencias ambientales dentro del sistema empleado. En muchos casos, es preciso utilizar un postprocesamiento determinista de los datos del ruido digitalizados tal como se obtienen de la fuente (*raw noise data*) con el fin de eliminar cualquier sesgo o dependencia.
383. Así pues, una fuente realmente aleatoria de números puede entenderse como un procedimiento probabilístico que proporciona bits aleatorios. En general, es muy difícil evaluar la calidad de la salida de una fuente aleatoria y se suelen emplear dos aproximaciones: 1) Mediante pruebas estadísticas a la salida de la fuente y 2) Modelando el proceso probabilístico de la fuente empleada.
384. En el primer caso (pruebas estadísticas a la salida de la fuente), el enfoque se considera es el de caja negra, es decir, no se necesita ningún conocimiento sobre la fuente para realizar las pruebas correspondientes. Este caso tiene dos inconvenientes. El primero, es que las pruebas estadísticas son genéricas y solo se pueden usar para detectar deficiencias de la fuente cuando se compara con una fuente aleatoria ideal. El segundo inconveniente es que las pruebas realizadas no proporcionan ninguna garantía acerca de la distribución de la salida de la fuente aleatoria. En cualquier caso, las pruebas estadísticas son útiles para detectar fallos de la fuente aleatoria, dicho de otro modo, pasar estas pruebas no es una garantía de calidad aleatoria, pero no pasarlas es síntoma inequívoco de mala calidad.
385. En la segunda aproximación (modelado del proceso probabilístico de la fuente) se necesita un gran conocimiento del diseño de fuente aleatoria y, en este caso, se intenta evaluar la calidad de la fuente a partir del estudio de un modelo teórico de la misma. Este enfoque proporciona una mayor seguridad, pero como contrapartida, requiere un alto nivel de experiencia en dominios como la estadística y la física.
386. En términos generales, los generadores de números aleatorios compatibles con PTG.2 o con PTG.3 deben cumplir las siguientes propiedades [42]:
1. Las propiedades estadísticas de los números aleatorios se pueden describir mediante un modelo estocástico y sobre la base de este modelo estocástico, la entropía de los números aleatorios se puede estimar de forma fiable.
  2. El aumento medio de la entropía por bit aleatorio está por encima de un límite mínimo dado (cercano a 1).

3. Las señales de ruido digitalizadas se someten a pruebas estadísticas en línea, que son adecuadas para detectar defectos estadísticos inaceptables, en un período de tiempo razonable.
4. Un fallo total de la fuente de ruido se identifica inmediatamente. Por ello, no se deben admitir números aleatorios que hayan sido generados después de este tipo de fallo.
5. Si se identifica un fallo total de la fuente de ruido o defectos estadísticos inaceptables de los números aleatorios, se provoca una alarma, que va seguida de una respuesta apropiada, como puede ser el apagado de la fuente de ruido.
6. El postprocesamiento criptográfico fuerte para generadores de tipo PTG.3 garantiza que el nivel de seguridad de un DRNG conforme con DRG.3 sigue estando asegurado aunque se produzca un fallo total.

387. En la Tabla 5.1 se muestran las clases de los RNG que están autorizados.

Clase	Rec./Her.	Referencias	Notas
PTG.3	R	[39], [115]	72, 73 74
PTG.2	R	[39], [115]	72, 73 75

**Tabla 5.1: Clases de generadores de números aleatorios autorizados**

388. Nota 72 **[Generadores]** El desarrollo y la evaluación de la seguridad de generadores físicos de números aleatorios requiere una amplia experiencia en este campo por lo que se recomienda el consejo de expertos en este campo en las primeras etapas.

389. Nota 73 **[Generadores sin fuente aleatoria directa]** Debido a la dificultad de evaluar la calidad de una verdadera fuente aleatoria, su uso debería limitarse a la obtención de la semilla y, opcionalmente, resemillar un generador de bits aleatorios determinista. No se acuerdan otros usos de un verdadero generador aleatorio.

390. Nota 74 [Generador PTG.3] Es posible construir un generador de clase PTG.3 a partir de un generador PTG.2 mediante un postprocesado criptográfico de la salida del generador PTG.2 de forma adecuada (puede implementarse en software) [115]. En términos generales, este postprocesamiento debe implementar, además, un generador de números aleatorios determinista compatible con DRG.3 (ver §5.3) de modo que se agregue, al menos, tanta entropía al estado interno del generador de números aleatorios como requiera la aplicación criptográfica.

391. Nota 75 [Generador PTG.2] No está permitido el uso de RNG de clase PTG.2 de forma aislada. Solo se acepta su uso para alimentar un generador determinista y siempre que se lleve a cabo un post-procesado de los datos originales.

### 5.3. GENERADORES DE NÚMEROS ALEATORIOS DETERMINISTAS (§10.9)

392. Los generadores de números aleatorios deterministas, también conocidos como «generadores de números pseudoaleatorios», permiten obtener una secuencia de números pseudoaleatoria de prácticamente cualquier longitud a partir de un valor aleatorio de longitud fija, denominado «semilla». Para lograr esta secuencia, el estado interno del RNG se inicializa con la semilla. En cada paso iterativo se obtiene un número aleatorio (generalmente, una secuencia de bits de longitud fija) a partir del estado interno del RNG y de la salida.
393. Es claro que el estado interno de un DRNG debe protegerse de manera confiable contra lectura y manipulación.
394. Los generadores de números aleatorios deterministas híbridos, por su parte, actualizan el estado interno, de vez en cuando, con valores que son realmente aleatorios (actualizando la semilla). Con relación a esto, es posible utilizar varios esquemas de actualización de semillas, como por ejemplo actualizando la misma a intervalos regulares o a través de la petición de la aplicación.
395. En la Tabla 5.2 se presentan los generadores de números aleatorios deterministas autorizados.

Esquema	Rec./Her.	Referencias	Notas
HMAC-DRBG	R	[38], [97]	76, 77
Hash-DRBG	R	[38], [97]	76, 77
CTR-DRBG	R	[38], [97]	76, 77

Tabla 5.2: Generadores de números aleatorios deterministas autorizados

396. Nota 76 **[Semillas de DRNG]** La seguridad de un DRNG deriva de un adecuado semillado del estado interno de la construcción, partiendo de una fuente aleatoria. Se debe mantener la entropía mínima para la operación de (re)semillado del DRNG impuesta por su especificación. Además, la entropía mínima de la semilla a utilizar será de, al menos, 128 bits.

397. Nota 77 **[Resistencia hacia atrás]** En los sistemas que tienen como objetivo proporcionar un secreto perfecto persistente (PFS), un atacante que haya recuperado el estado actual del RNG empleado en el intercambio de claves anteriores para producir claves efímeras, podrá ser capaz de vulnerar la propiedad PFS si es posible en la práctica calcular salidas anteriores del DRG partiendo del estado actual del DRG. Por lo tanto, solo se deben utilizar los DRNG que no permitan dicho cálculo hacia atrás.

398. En el caso de que se emplee un DRNG, se recomienda utilizar un generador que cumpla con las especificaciones DRG.3 o DRG.4 contra el potencial de ataque alto de acuerdo con AIS 20 [38, 115]. Si se utilizan generadores de números aleatorios de clase DRG.3, es deseable refrescar el flujo de entropía en el estado del RNG, incluso si calidad no es lo suficientemente alta para lograr el cumplimiento de la clase DRG.4.

399. La Tabla 5.3 presenta las diferentes clases de los DRNG autorizados.

Clase	Rec./Her.	Referencias	Notas
DRG.3	R	[38], [115]	78, 79
DRG.4	R	[39], [115]	78, 79, 80

Tabla 5.3: Clases de generadores de números aleatorios deterministas autorizados

400. Nota 78 [Predecesores y Sucesores de DRG.3 y DRG.4] Es prácticamente imposible para un adversario calcular o adivinar predecesores o sucesores de una secuencia de números aleatorios conocida, con una probabilidad significativamente mayor de lo que sería posible sin conocer esta subsecuencia.

401. Nota 79 [Números previamente generados de DRG.3 y DRG.4] Es prácticamente imposible para un adversario calcular o adivinar números aleatorios previamente generados basándose en el conocimiento de un estado interno, con una probabilidad significativamente mayor de lo que sería posible sin conocer el estado interno.

402. Nota 80 [Conformidad con DRG.4] Para ser conforme con la clase DRG.4 se impone la siguiente condición: incluso en el caso de que un adversario conociera el estado interno actual, es prácticamente imposible que pueda calcular o adivinar los números aleatorios que se generan después de la siguiente actualización de la semilla con un probabilidad significativamente mayor de la que sería posible sin conocer el estado interno. Además, los generadores de la clase DRG.4 tienen más ventajas que los de clase DRG.3 con respecto a los ataques a la implementación.

#### 5.4. GENERADORES DE NÚMEROS REALMENTE ALEATORIOS NO FÍSICOS (§10.10)

403. En muchas aplicaciones criptográficas, como en el comercio electrónico o el gobierno electrónico, no se dispone de un generador de números aleatorios físico, dado que la generación de los números que se precisan se lleva a cabo en ordenadores que no disponen de un hardware criptográfico certificado. En su lugar, se utilizan los denominados generadores de números aleatorios no físicos o NPTRNG (*Non-Physical True Random Number Generator*) [42].

404. Como en los TRNG, los NPTRNG también generan números realmente aleatorios, por lo que deben producir la entropía suficiente, pero no utilizan hardware dedicado, sino determinados recursos del sistema (como el tiempo del sistema, el contenido de la RAM, etc.) o interacciones con el usuario (como el movimiento del ratón, cadencia en la entrada del teclado, etc.). Los NPTRNG se utilizan, en general, en ordenadores que no se han desarrollado específicamente para aplicaciones criptográficas, como

los ordenadores domésticos y de ofimática, los portátiles o los smartphones, entre otros.

405. Una forma típica de proceder con los NPTRNG es la siguiente: se generan largas cadenas de bits no deterministas, siendo la entropía por bit generalmente bastante baja. A continuación esta cadena de bits se mezcla con un estado interno. Sobre la base del estado interno, se calculan y se emiten posteriormente los números aleatorios.
406. El más conocido de estos NPTRNG es `/dev/random`. En [115] se define una clase de funcionalidad para tales generadores de números aleatorios, denominada NTG.1. La clase de estos generadores de números aleatorios debe estimar de manera confiable la cantidad de entropía recolectada durante su uso operativo y los datos de salida deben tener una entropía de Shannon  $> 0,997$  bit a bit de salida.
407. La Tabla 5.4 presenta las diferentes clases de los NPTRNG autorizados.

Clase	Rec./Her.	Referencias	Notas
NTG.1	R	[115]	81, 82

**Tabla 5.4: Clase de generador de números aleatorios ni físico ni determinista autorizado**

408. Nota 81 **[Números previamente generados de NPTRNG]** Es virtualmente imposible para un adversario calcular o adivinar los números aleatorios previos basándose en el conocimiento del estado interno y las cadenas de bits aleatorias utilizadas previamente para actualizaciones de semillas, con una probabilidad significativamente mayor de lo que sería posible sin conocer el estado interno.

409. Nota 82 **[Fuentes de entropía para NPTRNG]** Para los NPTRNG es de enorme importancia que las fuentes de entropía utilizadas por el RNG no puedan ser manipuladas por un adversario en términos de reducción de la entropía o que sean predecibles si el adversario está equipado con información precisa sobre el entorno de ejecución.  
Cuando se planea usar un NPTRNG como el único RNG de un sistema dado que va a ser empleado para el procesamiento de datos sensibles, siempre se debe consultar a un experto.

410. Ya se ha mencionado que para inicializar un DRNG se precisa una semilla con una entropía suficientemente alta (ver §5.3). Por ello, la semilla debe generarse con

un generador físico de números aleatorios de las clases de funcionalidad PTG.2 o PTG.3. Como en los ordenadores normales no se dispone de un TRNG o tal RNG no está certificado por una entidad independiente del fabricante, se recomienda el uso de un generador de números aleatorios ni físico ni determinista. Para este propósito, los RNG que cumplen con la clase NTG.1 son adecuados, dado que presentan un alto potencial de ataque.

Nota 83 [Normas generales sobre el uso de RNG] A modo de resumen, conviene considerar unas normas generales a la hora de generar números aleatorios para aplicaciones criptográficas [42]:

- 411.
- Cuando se usa un PTRNG se recomienda usar uno de clase PTG.3. En particular en los casos de generación de claves efímeras para esquemas de firmas digitales y en el protocolo de Diffie-Hellman. En otros contextos se puede construir un generador PTG.3 mediante el postprocesamiento criptográfico de la salida de un generador PTG.2.
  - En general, los generadores PTG.3 y DRG.4 tienen, en comparación con los generadores PTG.2 y DRG.3, la ventaja de que resisten mejor a los ataques de canal lateral y ataques por inducción de fallos.
  - Cuando se utiliza un generador de números aleatorios determinista, se recomienda utilizar un generador DRG.3 o DRG.4, y entonces, es recomendable generar la semilla a partir de un PTRNG de clase PTG.2 o PTG.3. Si no se dispone de un RNG de este tipo, se puede considerar el uso de un generador de números aleatorios no físico y no determinista (ver §5.4).
  - En el caso general, se requiere una entropía mínima de la semilla del RNG de  $n$  bits para una seguridad del sistema de  $n$  bits.
  - Los generadores de números aleatorios híbridos combinan las propiedades de seguridad de los generadores PTRNG y DRNG. Además de una fuerte fuente de ruido, estos generadores deben estar dotados de un potente postprocesamiento criptográfico con memoria. Esto se logra típicamente postprocesando criptográficamente los números aleatorios de un generador de números aleatorios conforme a PTG.2 de una manera apropiada.

## 5.5. GENERACIÓN DE NÚMEROS ALEATORIOS CON UNA DISTRIBUCIÓN ESPECÍFICA (§10.11)

412. Muchos mecanismos criptográficos asimétricos requieren generar números enteros que sigan una distribución específica. Para tales mecanismos, no se puede utilizar

directamente un generador de bits aleatorios, dado que no ofrece directamente una distribución adecuada. Por tanto, los generadores aleatorios que ofrecen distribuciones de salida específicas deben implementarse a partir de generadores de bits aleatorios.

413. En particular, destacan los mecanismos asimétricos que precisan usar números aleatorios generados uniformemente al azar, pero que pertenezcan a un intervalo determinado, de la forma  $[0, q - 1]$ , donde  $q$  no es una potencia de 2. Esto especialmente relevante cuando se trata de generar al azar un elemento dentro de un grupo finito cuyo orden es la potencia de un primo. En estos casos, los generadores autorizados se conocen como de distribución uniforme módulo  $q$ .
414. La Tabla 5.5 muestra los esquemas autorizados para generar números enteros aleatorios módulo un número  $q$  dado, que no es una potencia de 2.

Esquema	Rec./Her.	Referencias	Notas
Técnica de "prueba"	R	[162, Apend. B.1.2]	84
Técnica "extra aleatoria"	R	[162, Apend. B.1.1]	84

**Tabla 5.5: Esquemas autorizados de generación de números enteros aleatorios módulo un número  $q$ , que no es una potencia de 2**

415. La técnica de "prueba" asegura la generación uniforme modulo  $q$  a costa del uso de una cantidad variable de aleatoriedad, posiblemente adicional. Por su parte, la técnica "extra aleatoria" hace que los sesgos sean insignificantes a costa de una pequeña cantidad fija de aleatoriedad adicional.

416. Nota 84 [**Reducción modular aleatoria**] Debe tenerse en cuenta que el método de generación de números enteros que consiste en usar el generador de bits aleatorios subyacente para obtener un número entero aleatoriamente de manera uniforme en un rango de longitud  $2^\ell$ , donde  $\ell = \lfloor \log_2(q) \rfloor$  es el suelo (floor) o  $\ell = \lceil \log_2(q) \rceil$  el techo (ceil) de  $\log_2(q)$ , posiblemente aplicando una reducción modulo  $q$  al resultado, introduce sesgos en la generación que pueden dar lugar a ataques.

## 6. GESTIÓN DE CLAVES

417. En general, la seguridad de los mecanismos criptográficos se basa en la confidencialidad, integridad y autenticidad (CIA) de las claves utilizadas. Comúnmente se acepta que para que el mecanismo empleado sea seguro, es indispensable que los adversarios no sean capaces de comprometer o alterar las claves. En esta sección se presentan los principales aspectos a tener en cuenta en cuanto a la gestión de las claves se refiere.

### 6.1. GESTIÓN DE CLAVES

418. Dado que los mecanismos criptográficos autorizados se consideran robustos, su uso no pone en riesgo la CIA de las claves que utiliza. No obstante, cuando se evalúa un producto que implementa mecanismos criptográficos, se deben considerar todas las formas en las que el producto manipula el material clave y cualquier forma en la que un adversario podría intentar vulnerarlo, de modo que quede asegurado el hecho de que no puede obtener las claves.

419. Como los criptosistemas simétricos suelen estar restringidos a un grupo de cerrado de usuarios, las claves simétricas deben distribuirse entre ellos de modo que nadie externo al grupo cerrado pueda tener conocimiento de las mismas. También es fundamental que el canal de distribución de las claves esté protegido para la autenticidad e integridad.

420. En el caso de los criptosistemas asimétricos, como una de las claves es pública, puede enviarse o compartirse a través de un canal no confidencial, aunque debe protegerse para garantizar su autenticidad e integridad. Por el contrario, como la clave privada se puede generar localmente, debe estar protegida para evitar que pueda acceder a ella cualquier otro usuario que no sea su legítimo propietario.

421. La norma básica en la gestión de claves es la siguiente:

422. Nota 85 [Gestión de claves] La gestión de claves por parte del producto no debería permitir a un atacante recuperar información sobre claves secretas y privadas utilizadas para proteger la información del usuario, ni alterar o inyectar claves públicas utilizadas para proteger identidades.

### 6.2. GENERACIÓN DE CLAVES (§10.12)

423. Para que un adversario no tenga conocimiento a priori de las claves utilizadas por un determinado mecanismo criptográfico, dichas claves deben ser impredecibles.

Además, se requiere que las claves sean lo suficientemente largas como para garantizar la CIA de los protocolos que las utilicen, así como que la distribución de la salida del proceso utilizado para generarlas no se pueda distinguir de una distribución uniforme.

424. En esta sección se presentan los métodos de generación de claves autorizados para los mecanismos criptográficos genéricos que requieran claves. Salvo que se indique lo contrario, las claves utilizadas para los mecanismos criptográficos convenidos de los apartados anteriores se obtendrán truncando una secuencia de bits de salida por un método de generación autorizado y dependiente del tamaño de la clave del mecanismo.
425. En la Tabla 6.1 se presentan los métodos autorizados para la generación de claves genéricas y algunas notas relacionadas.

Método	Notas
Generador de bits aleatorios autorizado	
Mecanismo de establecimiento de claves autorizado	86, 87
Función de derivación clave autorizada	87

**Tabla 6.1: Métodos autorizados para la generación de claves genéricas**

426. Nota 86 [**Establecimiento de clave**] Una clave establecida a través de un procedimiento de acuerdo o establecimiento de claves no debe usarse directamente de la fuente de salida, sino que debe ser posprocesada, en primer lugar, por una función de derivación de claves autorizada (ver la sección 2.2.7).

427. Nota 87 [**Semilla de generación clave**] Algunos mecanismos de generación de claves se basan en secretos preexistentes, como por ejemplo, los exponentes efímeros en el caso del protocolo de establecimiento de claves de Diffie-Hellman, o de un secreto maestro en el caso de la generación de claves del protocolo de registro TLS. En cualquier caso, la entropía de los secretos preexistentes será de, al menos, 125 bits.

428. Para aquellos mecanismos que tengan necesidades específicas a la hora de generar sus claves, o bien se especifica cómo generarlas en el momento en el que son tratados en esta guía, o bien se suele definir un procedimiento específico de generación de claves utilizando un generador de bits aleatorios a modo de caja negra.

### 6.3. ALMACENAMIENTO Y TRANSPORTE DE CLAVES

429. Nota 88 **[Almacenamiento y transporte de claves]** Cuando se transmita o almacene una clave por un medio o en un medio que no sea de confianza, la clave se protegerá con confidencialidad e integridad con un mecanismo de protección de claves autorizado, como se describe en la Sección 2.2.6.

### 6.4. USO DE CLAVES

430. El uso de una misma clave para diferentes mecanismos con el fin de garantizar, por ejemplo, la CIA, es una fuente de errores y también puede abrir vías de ataque que exploten los diversos contextos de uso de dicha clave. Hay que tener en cuenta que el uso de una clave debe entenderse en términos del objetivo de seguridad logrado y no restringido en términos de mecanismos criptográficos. Así, por ejemplo, un contexto de firma digital de un mensaje y un esquema de autenticación asimétrica pueden utilizar el mismo esquema de firma digital, pero los pares de claves utilizados en los dos contextos deben ser diferentes.

431. Nota 89 **[Uso de claves]** Una misma clave no debe utilizarse con diferentes mecanismos.

432. Nota 90 **[Plataforma de confianza]** Una clave solo se utilizará para realizar cálculos en una plataforma confiable. Se garantizará la confidencialidad e integridad de las claves secretas y privadas, y la integridad y autenticidad de las claves públicas.

433. Nota 91 **[Distribution]** La distribución de claves secretas y privadas se limitará al entorno de confianza que haga un uso efectivo de la clave.

## 6.5. DESTRUCCIÓN DE CLAVES

434. Nota 92 [**Destrucción de claves**] Al final del ciclo de vida de una clave, la clave se borrará de forma segura de la plataforma de confianza en la que se utilizó.
435. Ya se ha mencionado que algunos esquemas criptográficos utilizan claves efímeras. En el contexto de los esquemas de establecimiento de claves de Diffie-Hellman, estas claves efímeras son intercambiadas por las partes y se utilizan para derivar un secreto compartido. Siguiendo esta derivación, las claves efímeras ya no son útiles y deben borrarse de forma segura.
436. El proceso de borrado debe adaptarse al entorno y tener en cuenta los problemas de remanencia de la memoria.

## 7. AUTENTICACIÓN DE PERSONAS

437. La autenticación de una persona (o identificación) ante otra consiste en probar de modo fehaciente que se es quien dice ser, de modo que se convenza a la otra parte de tal hecho, esto es, se demuestra la identidad de la primera persona ante la otra entidad, que puede ser una persona, un equipo, etc.

### 7.1. AUTENTICACIÓN

438. En general, la autenticación de una persona ante una entidad se asegura mediante el uso de mecanismos criptográficos. Sin embargo, si la persona debe identificarse ante sí misma en un sistema de información, entonces hay algunas diferencias.

439. La primera de ellas es que dicha persona no puede hacer uso directo de mecanismos criptográficos. La segunda es que lo más probable es que el procedimiento de autenticación se reproduzca, como es el caso, por ejemplo, de las autenticaciones a largo plazo basadas en contraseñas. En tercer lugar, suele suceder que la entropía de los datos usados para la identificación (como contraseñas), es inferior a lo que se esperaría de un sistema criptográfico estándar.

440. Por ello, los procedimientos de autenticación de personas solo se pueden realizar localmente en una plataforma confiable o a través de un canal confiable. Por ejemplo, introduciendo un número de identificación personal o PIN (Personal Identification Number) o proporcionando una *cookie* a un sitio web. Además, estos procedimientos deben requerir una interacción con el sistema porque si los datos de verificación de la identidad pudieran extraerse del sistema, un atacante podría recuperar los datos de identificación mediante el uso de fuerza bruta.

441. En esta guía nos limitaremos a considerar soluciones de autenticación de personas basadas en contraseñas y en PIN.

### 7.2. PROCEDIMIENTOS DE AUTENTICACIÓN (§10.13)

442. El sistema de autenticación de personas debe imponer determinados límites en el procedimiento de autenticación, de modo que una persona no autorizada pueda llevar a cabo infinitas o un número muy elevado de pruebas hasta lograr una autenticación fraudulenta y llegue a suplantar a un usuario legítimo del sistema.

443. En esta guía consideraremos los dos casos que se detallan a continuación.

### 7.2.1. LIMITACIÓN EN EL NÚMERO DE ENSAYOS

444. En el primer caso, el sistema puede imponer un límite determinado en la cantidad de intentos que la persona que realiza la autenticación puede llevar a cabo para introducir la información correcta.
445. En general, este caso se implementa mediante recursos criptográficos, por ejemplo, tarjetas inteligentes o módulos de seguridad hardware (HSM o *Hardware Security Module*), con el fin de desbloquear el acceso a operaciones que afecten a claves almacenadas de forma segura en el recurso.
446. Después de un número determinado de intentos erróneos, la autenticación se desactiva, esto es, se vuelve imposible porque la cuenta personal ya no se puede desbloquear sin recurrir a un procedimiento administrativo. En definitiva, se trata de reducir, en la medida de lo posible, la probabilidad de aceptación falsa, es decir, impedir que una persona que no conoce los datos de identificación pueda lograr autenticarse mediante ensayos aleatorios. En general se recomienda utilizar una probabilidad máxima de falsa aceptación de  $5 \times 10^{-6}$ , correspondiente a un máximo de 5 intentos de un PIN de 6 dígitos. Una probabilidad máxima de aceptación falsa de  $5 \times 10^{-4}$ , como máximo 5 intentos de un PIN de 4 dígitos, es aceptable en aplicaciones heredadas.
447. En la Tabla 7.1 se presentan las probabilidades máxima de falsa aceptación para el caso de los intentos que se citan.

Nº de intentos	Probabilidad de falsa aceptación	Rec./Her.
5	$5 \times 10^{-6}$	R
5	$5 \times 10^{-4}$	H

**Tabla 7.1: Probabilidades máximas de falsa aceptación**

### 7.2.2. LIMITACIÓN TEMPORAL EN EL NÚMERO DE ENSAYOS

448. En el segundo caso, se trataría de limitar el tiempo que el usuario dispondría para llevar a cabo su identificación. Esta limitación no es práctica en aplicaciones reales porque no se discrimina la velocidad con la que un usuario puede llevar a cabo diferentes intentos. Cabría la posibilidad de que el sistema limitara la velocidad a la que la persona que se intenta autenticar puede someterse a tal procedimiento. Dicho de otro modo, este caso proporciona una solución de peor calidad que el caso anterior al limitar la cantidad de intentos de autenticación por unidad de tiempo.
449. La implementación de este caso suele considerarse solo si no es práctico aplicar un mecanismo estricto de bloqueo de cuenta cuando se llega al número máximo de errores de autenticación permitidos. Un ejemplo de dicho mecanismo de limitación sería el de duplicar la demora después de cada intento de autenticación fallido antes

de que sea posible un nuevo intento. Por el momento, esta guía no considera ningún requisito para este segundo caso.

## 8. GENERACIÓN DE PRIMOS Y CLAVES RSA

450. En esta sección se muestran diferentes métodos para la generación de números primos de determinada longitud. Tal generación es básica puesto que la generación de primos de modo seguro es utilizada en varios mecanismos de clave asimétrica, en especial en el RSA.

### 8.1. GENERACIÓN DE NÚMEROS PRIMOS (§10.14)

451. La generación de parámetros para los mecanismos asimétricos y de claves para el RSA necesita generar números primos, cuyo conjunto se denotará por  $\mathbb{P}$ , que sean aleatorios y que, en general, deben mantenerse en secreto. Por ello, esta generación debe ser aleatoria y lo más eficiente posible, de modo que la estrategia utilizada para generar tales números primos es la siguiente: en primer lugar, se genera al azar un número entero candidato del tamaño que se requiera. A continuación se prueba si tal número cumple determinadas propiedades y si es primo. En el caso de que tales pruebas fallen, el número entero candidato se actualiza y se prueba de nuevo con el siguiente candidato.

452. La función de actualización cuando se eligen diferentes números candidatos debe ofrecer un equilibrio entre la cantidad de aleatoriedad extra necesaria y la proximidad de la distribución de números primos generada a la distribución uniforme de números primos. Por otra parte, la prueba de primalidad utilizada puede ser una prueba de pseudo-primalidad, esto, es una prueba de verificación de pseudo-primos o una prueba de primos demostrables o comprobables.

453. En esta sección se muestran dos algoritmos autorizados que permiten generar números primos (véanse los Algoritmos 24 y 25). En ambos casos se supone que se dispone de una primera prueba *Test*, que verifica las condiciones que debe verificar el primo y de una segunda, *TestPrime*, que es una prueba de primalidad.

454. El primer algoritmo para generar números primos (ver Algoritmo 24) no es excesivamente eficiente y utiliza pruebas como las mencionadas anteriormente, esto es, *Test* y *TestPrime*.

455. El segundo algoritmo para generar números primos (ver Algoritmo 25) es más eficiente que el anterior. También en este caso se dispone de pruebas similares a las mencionadas anteriormente, *Test* y *TestPrime*.

456. La Tabla 8.1 muestra dos métodos de generación de primos por muestreo de rechazo autorizados. El segundo de ellos es más eficiente que el primero. El tercer método puede inducir ciertos sesgos estadísticos en la distribución de los primos generados, lo que no es conveniente. Sin embargo, es un método ampliamente utilizado [207, Tabla 1] y no hay indicios de que estos sesgos puedan usarse para atacar el método. Así pues, este tercer método se acepta como un método heredado.

---

**Algoritmo 24** Generación de primos por muestreo de rechazo (método 1)

---

*Entrada:* un intervalo  $I = [a, b] \cap \mathbb{N}$ , en el que se encuentra el primo, una prueba opcional *Test* que codifica propiedades adicionales del primo a generar y una prueba de primalidad *TestPrime*.

*Salida:* un número primo  $p$ .

1. Se selecciona un número  $p$  de acuerdo con la distribución uniforme en  $I$ .
  2. Si  $p$  no es impar, volver al paso 1.
  3. Si  $\text{Test}(p) = \text{Falso}$ , volver al paso 1.
  4. Si  $\text{TestPrime}(p) = \text{Falso}$ , volver al paso 1.
  5. Devolver  $p$ .
- 

---

**Algoritmo 25** Generación de primos por muestreo de rechazo (método 2)

---

*Entrada:* un intervalo  $I = [a, b] \cap \mathbb{N}$ , en el que se encuentra el primo, un entero natural pequeño  $B$  que satisfaga que el producto de los números primos menores que  $B$ , denotado por  $\Pi_B = \prod_{p_i \in \mathbb{P}, p_i < B} p_i$ , sea mucho menor que la amplitud del intervalo  $[a, b]$ ,  $\Pi_B \ll b - a$ , una prueba opcional *Test* que codifica propiedades adicionales del primo a generar y una prueba de primalidad *TestPrime*.

*Salida:* un número primo  $p$ .

1. Se selecciona un número aleatorio  $r$  siguiendo la distribución uniforme en  $[1, \Pi_B]$ .
  2. Si  $\text{mcd}(r, \Pi_B) \neq 1$ , volver al paso 1.
  3. Se elige al azar  $k \in \mathbb{N}$  de modo que  $p = k\Pi_B + r \in I$ . Esto es,  $k$  se debe seleccionar de acuerdo con la distribución uniforme en  $\left[ \left\lceil \frac{a-r}{\Pi_B} \right\rceil, \left\lfloor \frac{b-r}{\Pi_B} \right\rfloor \right]$ .
  4. Si  $\text{Test}(p) = \text{Falso}$ , volver al paso 3.
  5. Si  $\text{TestPrime}(p) = \text{Falso}$ , volver al paso 3.
  6. Devolver  $p$ .
-

Esquema	Rec./Her.	Notas
Algoritmo 24 de generación de primos (Método 1)	R	93
Algoritmo 25 de generación de primos (Método 2)	R	93

**Tabla 8.1: Generación de primos por muestreo de rechazo**

457. Nota 93 (**Ataque ROCA**) El método considerado no es susceptible de ser vulnerado por el ataque ROCA (véase §8.4) [146].

## 8.2. TEST DE PRIMALIDAD Y DE PSEUDO-PRIMALIDAD

458. Existen fundamentalmente dos procedimientos para la generación de primos: los que producen como resultado primos probables (*probable primes*) y aquellos que producen primos demostrables (*provable primes*). Aunque, en términos generales, los primeros son suficientes, en ciertos ambientes, como la generación de curvas elípticas usadas en criptografía, se pueden preferir primos demostrables [132], [64].

459. De manera formal, se tienen las siguientes definiciones.

460. **[Test de primalidad:]** es un algoritmo que determina que un número candidato es un número primo. Si el test de primalidad decide con total seguridad acerca de la primalidad del candidato se denomina determinista.

461. **[Test de composición:]** es un algoritmo que determina que un número candidato es un número compuesto, esto es, un test de composición solo es capaz de asegurar que el candidato es compuesto, de modo que si su salida afirma que no lo es, cabe aún la posibilidad de que lo sea. Por ello, este tipo de test sirve como test de primalidad, pero será un test probabilístico.

462. **[Número primo probable:]** es un número entero que se cree que es primo según una prueba de primalidad probabilística. No debería haber más que una probabilidad despreciable de que el llamado primo probable sea en realidad compuesto.

463. **[Número primo demostrable:]** Es un número entero que se construye para ser primo o se calcula para ser primo utilizando un algoritmo o test de primalidad determinista.

### 8.2.1. GENERACIÓN DE PRIMOS PROBABLES (*PROBABLE PRIMES*)

Es bien conocido que el problema de la primalidad, esto es, decidir si un número dado es primo o no, es un problema que se puede resolver de forma determinista en tiempo polinómico [6]. Si bien desde un punto de vista teórico este resultado es fundamental, el algoritmo requiere un tiempo de ejecución  $\tilde{O}\left(k^{\frac{21}{2}}\right)$ , siendo  $k$  el tamaño del primo generado. Existen variantes de este algoritmo que rebajan ligeramente este tiempo, pero, en todo caso, no es un algoritmo que suela emplearse debido a su escasa velocidad.

Existen otros algoritmos deterministas como los siguientes:

- El test de Lucas-Lehmer que es tiempo polinómico [123], pero solo es aplicable a números de Mersenne [179, Algor. 5.14], es decir, los que se pueden escribir en la forma  $n = 2^s - 1$ .
- El test de las sumas de Jacobi se ejecuta en tiempo subexponencial  $O(k^{\log \log b})$ . [4], [52], [51].
- El test de Goldwasser-Killian, basado en curvas elípticas, se ejecuta en tiempo (probabilísticamente) polinómico  $O(k^{6+\varepsilon})$ .
- El test de Pocklington [176] [179, §5.1.3], en el que se basa el método de generación de primos de Maurer, cuyo tiempo de ejecución está dominado por el coste computacional de la exponenciación.
- El test de primalidad basado en curvas elípticas, debido a Goldwasser y Kilian [77], se basa, en cierto modo, en el test de Pocklington y corre en tiempo polinómico probabilístico para la mayoría de los números primos y conjeturalmente para todos ellos. El test puede probar la primalidad de un número en  $O((\log n)^k)$  operaciones para una determinada constante  $k$  ( $k \approx 6 + \varepsilon$ ).

464. Entre los principales algoritmos probabilísticos destacan los siguientes:

- El test de Fermat que está basado en el teorema (pequeño) que lleva su nombre. Su tiempo de ejecución es The running time is  $O((\log n)^{2+\varepsilon})$ , dependiendo del algoritmo de multiplicación que se utilice. A pesar de que el test de Fermat solo proporcione una primalidad probable, es muy útil dado que computacionalmente muy barato. Por otra parte, es fácil probar que la probabilidad de declarar erróneamente primo un número que es realmente compuesto es menor que  $2^{-\ell}$ , cuando el test se repite  $\ell$  veces para un candidato dado.

- El test de Solovay-Strassen (ver [198], [199] y [179, §5.2.1]) se basa en la siguiente propiedad: si  $n$  es un número primo impar y  $a$  un entero con  $1 \leq a \leq n-1$ , tal que  $\text{mcd}(a, n) = 1$ , entonces  $a^{\frac{n-1}{2}} \equiv \left(\frac{a}{n}\right) \pmod{n}$ , siendo  $\left(\frac{a}{n}\right)$  el símbolo de Jacobi.
- El test de Miller-Rabin (ver [139], [180], [135, Algor. 4.24] y [179, Algor. 5.28]) es el algoritmo probabilístico por excelencia. Se basa en el Teorema (pequeño) de Fermat, que afirma que para cualquier número  $p$  primo, y para cualquier número  $a$  coprimo con  $p$ , esto es, con  $\text{mcd}(a, p) = 1$ , se verifica que  $a^{p-1} \equiv 1 \pmod{p}$ . El tiempo de ejecución esperado para el algoritmo de Miller-Rabin es  $O((\log n)^{2+\varepsilon})$ ,  $\varepsilon \geq 1$ , dependiendo del algoritmo de multiplicación empleado y siendo  $n$  el número candidato.

465. En el Teorema de Fermat, cuando tal número  $a$  existe, se denomina «base» prima con  $p$ . Si se desea comprobar la primalidad de un número candidato  $n$  y se encuentra un valor  $a$ , coprimo con  $n$ , tal que la congruencia anterior no se verifique, entonces se puede asegurar que  $n$  es compuesto. Sin embargo, la afirmación recíproca no es siempre cierta. Dicho de otro modo, el hecho de que no se encuentre base alguna en la que no se verifique dicho teorema no garantiza que el número  $n$  sea primo.

466. Es importante señalar que, aunque la condición que proporciona el Teorema pequeño de Fermat sea sólo una condición necesaria, el número de excepciones es muy pequeño. Tales excepciones se conocen como «pseudoprimos». De hecho, los números  $n$  que satisfacen la congruencia del Teorema pequeño de Fermat,  $a^{n-1} \equiv 1 \pmod{n}$ , para toda base  $a \in [2, n-1]$ , prima con  $n$ , reciben el nombre de números de Carmichael (el más pequeño de tales números es  $n = 3 \cdot 11 \cdot 17 = 561$ ).

467. El algoritmo de pseudo-primalidad de Miller-Rabin (véase el Algoritmo 26) utiliza como entrada los números  $n, t \in \mathbb{N}$ , con  $n \geq 3$  e impar y  $t$  un parámetro de seguridad, y determina si  $n$  es primo con una probabilidad de acierto de  $1 - \frac{1}{2^{2t}} = 1 - 2^{-2t}$ .

468. Si  $P_{obj}$  es el valor objetivo de la probabilidad considerada, entonces se tiene que  $P_{obj} \geq 2^{-2t}$ , es decir, tomando logaritmos y despejando

$$\begin{aligned} \log_2(P_{obj}) &\geq -2t \log_2 2, \\ t &\geq -\frac{1}{2} \log_2(P_{obj}), \end{aligned}$$

es decir,  $1 \leq t \leq \lfloor -\log_2(P_{obj}) \rfloor$ .

469. Se puede estimar la probabilidad,  $P_{k,t}$ , de que un número entero impar de  $k$  bits que pase  $t$  iteraciones del test de Miller-Rabin sea en realidad compuesto. Esta probabilidad se entiende como la relación entre el número de números compuestos impares de  $k$  bits que se puede esperar que pasen  $t$  rondas de pruebas de este test, con bases generadas aleatoriamente, con la suma de ese valor y el número de primos impares enteros de longitud binaria  $k$  [55], [162, App. F.1].

**Algoritmo 26** Algoritmo de pseudo-primalidad de Miller-Rabin

*Entrada:* número entero candidato,  $n$ , y parámetro de seguridad (número de iteraciones),  $t$ .

*Salida:* decisión sobre la primalidad del candidato  $n$ .

1. Se determinan  $s, r \in \mathbb{Z}$  de modo que  $n - 1 = 2^s r$ , con  $r$  impar.

2. **for**  $i$  **from** 1 **to**  $t$  **do**

Se elige al azar un entero  $a$ , con  $2 \leq a \leq n - 2$

$b \leftarrow a^n \pmod{n}$

**if** ( $b \neq 1$  and  $b \neq n - 1$ ) **then do**

$j \leftarrow 1$

**while**  $j \leq s - 1$  and  $b \neq n - 1$  **do**

$b \leftarrow b^2 \pmod{n}$

**if**  $b = 1$  **then** devuelve "Compuesto"

$j \leftarrow j + 1$

**if**  $b \neq n - 1$  **then** devuelve "Compuesto"

3. Devuelve "Primo con probabilidad  $(1 - 2^{-2t})$ ".

470. Esta probabilidad viene dada por la siguiente expresión:

$$P_{k,t} = 2,00743 \cdot \ln(2) \cdot k \cdot 2^{-k} \cdot \left( 2^{k-2-M \cdot t} + \frac{8}{3}(\pi^2 - 6)2^{k-2} \sum_{m=3}^M 2^{m-(m-1)t} \sum_{j=2}^m \frac{1}{2^{j+\frac{k-1}{j}}} \right), \quad (8.1)$$

donde  $3 \leq M \leq \left\lceil 2\sqrt{k-1} - 1 \right\rceil$ . En definitiva, se acepta el valor del número de iteraciones,  $t$ , cuando se cumpla la condición  $P_{k,t} \leq P_{obj}$ .

471. A modo de ejemplo, en la Tabla 8.2 se presenta el número de iteraciones,  $t$ , a realizar con el test de Miller-Rabin<sup>10</sup> en función de la longitud en bits,  $k$ , de la entrada impar generada aleatoriamente para lograr probabilidades menores que  $2^{-125}$ .

<sup>10</sup>El número de iteraciones y las longitudes en bits de los candidatos mostrados en la Tabla 8.2 las hemos computado a la hora de elaborar dicha Tabla. En el caso de  $P_{obj} = 2^{-125}$ , se muestran entre paréntesis los valores publicados en [197]. Suponemos que la diferencia entre los valores calculados y los publicados para este caso, se debe a la distinta precisión numérica empleada a la hora de calcular los valores correspondientes aplicando la fórmula (8.1).

$P_{obj} = 2^{-125}$	
Nº iter., $t$	Long. bits candidato, $k$
6	$950 \leq k < 1036$ (1041)
5	$1036$ (1041) $\leq k < 1289$ (1297)
4	$1289$ (1297) $\leq k < 1720$ (1729)
3	$1720$ (1729) $\leq k < 2607$ (2626)
2	$2607$ (2626) $\leq k < 5672$ (5701)
1	$5672$ (5701) $\leq k$

Tabla 8.2: Número de iteraciones del test de Miller-Rabin según la longitud en bits del candidato para  $P_{obj} = 2^{-125}$

472. Según la Tabla 8.2, obtenida haciendo uso de la fórmula (8.1), se deduce que serán necesarias  $t = 6$  rondas con  $k = 1024$ , o  $t = 3$  rondas con  $k = 2048$  para excluir, con una probabilidad de error de  $2^{-125}$ , que  $p$  es un número compuesto, aunque el algoritmo de Miller-Rabin identifique a  $p$  como un número primo.
473. Dado que el mínimo número de bits exigido para un primo en el caso del criptosistema RSA es de 1536 (el módulo tendrá un tamaño de 3072 bits) y que para el algoritmo de Diffie-Hellman el tamaño del primo es de 3072 bits, siguiendo la expresión (8.1), se tiene que para  $k = 1024$  hacen falta  $t = 6$  iteraciones; para  $k = 1536$ , se necesitan  $t = 4$ , si  $k = 2048$  se precisan  $t = 3$  iteraciones y si  $k = 3072$ , el número de iteraciones es  $t = 2$ .
474. En la Tabla 8.3 se muestra el test autorizado para la determinación de la primalidad de un número candidato.

Esquema	Rec./Her.	Referencias	Notas
Test de Miller-Rabin	R	[139], [180]	94, 95, 96, 97

Tabla 8.3: Test de primalidad probabilístico autorizado

475. Nota 94 (**Primos probables**) En el caso de que la prueba de primalidad utilizada (TestPrime) no pruebe la primalidad del número candidato, la probabilidad de que este número sea compuesto debe ser inferior a  $2^{-125}$ .

- Nota 95 (**Miller-Rabin, peor caso**) Para acotar la probabilidad de que un número dado  $p$  sea identificado como un número primo mediante el algoritmo de Miller-Rabin, aunque sea un número compuesto, el algoritmo debe llamarse 50 veces, cada una de ellas con una base  $a_i \in \{2, 3, \dots, p - 2\}$ ,  $1 \leq i \leq 50$ , elegidas independientemente entre sí con respecto a una distribución uniforme [42, Remark 42(ii)].
476. En esta guía se recomienda realizar una verificación de la propiedad de primalidad con  $t = 50$  rondas para el test de Miller-Rabin en el caso que se generen números primos que deban utilizarse en funciones particularmente críticas para la seguridad en un criptosistema o cuya generación no sea especialmente exigente en el tiempo requerido [162, App. F2]. Esto se aplica, por ejemplo, a los números primos que se generan una vez como parámetros permanentes de un mecanismo criptográfico y no se cambian durante un período de tiempo largo, o posiblemente los utilicen muchos usuarios.

- Nota 96 (**Miller-Rabin, caso promedio**) Con el fin de probar la primalidad de un número impar,  $p \in [2^{k-1}, 2^k - 1]$ , que ha sido elegido aleatoriamente con respecto a la distribución uniforme, se puede considerar un número menor de iteraciones al señalado en la Nota 95 (ver [55], [162, App. F], y [103, Ann. A]). También en este caso, las bases deben elegirse independientemente y con distribución uniforme en  $\{2, 3, \dots, p - 2\}$ . El número específico de iteraciones necesarias depende del tamaño de  $p$ , dado que la densidad de números a los que se aplican las estimaciones del peor de los casos disminuye significativamente al aumentar el tamaño de los números.
- 477.

- Nota 97 (**Miller-Rabin, RNG**) A la hora de utilizar un generador de bits aleatorios con la aleatoriedad requerida, puede emplearse uno de la clase de funcionalidad PTG.3 o DRG.4 (ver Capítulo 5). Si bien desde un punto de vista de la teoría de la información, el uso de un generador de bits aleatorios determinista excluye cualquier posibilidad de generación uniforme de números primos, en este caso, la seguridad no se ve afectada. De hecho, un RNG de la clase de funcionalidad DRG.4 debería generar una distribución de salida que no se distinga de una distribución ideal, por lo que no puede ser atacado de modo práctico por un ataque conocido (no cuántico). Sin embargo, cabe señalar en este contexto que el nivel de seguridad de los módulos RSA generados puede estar limitado por el nivel de seguridad de la generación aleatoria de bits.
- 478.

### 8.3. GENERACIÓN DEL PAR DE CLAVES RSA (§10.15)

479. A continuación se presenta un algoritmo (véase el Algoritmo 27) que permite generar los dos números primos que se emplean para determinar el módulo en el mecanismo asimétrico RSA (véase la sección 3.1.1). El par de números primos aleatorios deben satisfacer un conjunto de restricciones para asegurar el tamaño de la clave, la consistencia del par de claves pública/privada y evitar claves débiles.

---

#### Algoritmo 27 Generación del par de claves para el mecanismo asimétrico RSA

---

*Entrada:* longitud en bits del módulo RSA,  $k$ , y exponente público (cifrado),  $e$ .

*Salida:* dos números primos  $p, q$ .

1. Si  $e$  es par o si  $e \leq 2^{16}$ , muestra *Fallo*.
  2. Usando un método de generación de primos aleatorios autorizado (ver Algoritmos 24 y 25), se genera un primo aleatorio  $p$  en el intervalo  $\left[\frac{1}{\sqrt{2}}2^{\frac{k}{2}}, 2^{\frac{k}{2}}\right]$ , de modo que  $\text{Test}(p): \text{mcd}(p-1, e) = 1$ .
  3. Usando un método de generación de primos aleatorios autorizado (ver Algoritmos 24 y 25), se genera un primo aleatorio  $q$  en el intervalo  $\left[\frac{1}{\sqrt{2}}2^{\frac{k}{2}}, 2^{\frac{k}{2}}\right]$ , de modo que  $\text{Test}(q): \text{mcd}(q-1, e) = 1$  y  $|p-q| \geq 2^{\frac{k}{2}-100}$ .
  4. Se determina  $d = e^{-1} \pmod{\text{mcm}(p-1, q-1)}$ .
  5. Si  $d \leq 2^{\frac{n}{2}}$ , volver al paso 1.
  6. Devolver  $p, q$ .
- 

480. En la Tabla 8.4 se presenta el algoritmo autorizado para la generación del par de claves para el mecanismo asimétrico RSA.

Esquema	Rec./Her.	Notas
Algoritmo 27 de generación de claves para RSA	R	98, 99, 100

**Tabla 8.4: Algoritmo autorizado para la generación claves RSA**

481. **Nota 98 (Generación de claves RSA)** Los números primos  $p$  y  $q$  deben ser dos primos generados aleatoriamente de la misma longitud y cuyo producto (módulo RSA) debe tener la longitud de bits dada. Los dos números primos no deben ser demasiado cercanos para evitar ataques de factorización que exploten una posible distancia pequeña entre los dos factores, por lo que debería verificarse que  $|p-q| \geq 2^{\frac{k}{2}-100}$ , siendo  $k$  el tamaño de dichos números primos.

482. Nota 99 (**Tamaño de la clave privada,  $d$** ) El tamaño de la clave privada,  $d$ , debe ser lo suficientemente grande, es decir, se debe verificar que  $d > 2^{\frac{k}{2}}$ , donde  $k$  la longitud de bits del módulo RSA. Debe tenerse en cuenta que esta condición está garantizado si el exponente público,  $e$ , es pequeño [84].

483. Nota 100 (**Intervalo de los números primos**) Para garantizar la seguridad de los pares de claves pública/privada para los que el módulo RSA se ha obtenido multiplicando dos números primos generados de forma independiente con uno de los métodos autorizados (ver la Tabla 8.1), es importante que el intervalo considerado,  $I = [a, b] \cap \mathbb{N}$  no sea demasiado pequeña. De hecho, si se considera que el módulo tiene una longitud de  $k$  bits, se suele considerar  $I = \left[ \left\lfloor \frac{2^{\frac{n}{2}}}{\sqrt{2}} \right\rfloor, \left\lceil 2^{\frac{n}{2}} \right\rceil \right] \cap \mathbb{N}$ . Las diferentes elecciones del intervalo  $I$  cumplen con esta guía si  $p$  y  $q$  se extraen del mismo intervalo  $I$  y si  $\#(I) \geq 2^{-8}b$ .

#### 8.4. ATAQUE ROCA AL RSA

484. Desde 2016 han aparecido algunas publicaciones que plantean la cuestión de si los bits de una clave pública RSA,  $(n, e)$ , pueden revelar información sobre la elección realizada en el diseño e implementación del algoritmo que genera los números primos  $p$  y  $q$ , es decir, si es posible identificar el origen de los primos conociendo sólo el módulo RSA y no su factorización.
485. De hecho, en [145], [207] y en [206], los autores intentan verificar si los pares de claves RSA generados por cierto software y determinadas tarjetas inteligentes ofrecen la calidad y la seguridad requerida en relación con la aleatoriedad deseada y la resiliencia frente a los ataques más generalizados. Para ello, estudian si es posible determinar el origen de las claves, es decir, identificar qué librería de software o qué tarjeta inteligente es la responsable de generar los números primos asociados a una clave RSA, suponiendo que solo se conoce la clave pública. Así, Švenda et al. [207] demostraron que las opciones de implementación en las bibliotecas criptográficas permiten suponer el origen de las claves RSA públicas.
486. Posteriormente, Nemeč et al. informaron en [146] sobre su descubrimiento de un fallo en el algoritmo en la construcción de números primos para la generación de claves RSA en una biblioteca ampliamente utilizada de un importante fabricante de hardware criptográfico. Este ataque ha sido llamado «ataque ROCA» debido al título del artículo. Los números primos generados por la biblioteca sufren una importante pérdida de entropía, defecto que aprovecharon los autores para proponer un método práctico de factorización para varias longitudes de clave, incluidos 1024

y 2048 bits. Su método no requería información adicional excepto conocer el valor del módulo RSA y no dependía de un generador de números aleatorios débil o defectuoso. Nemeec et al. extendieron el ataque de factorización de Coppersmith utilizando una forma alternativa de los números primos en cuestión. La librería software utilizada se encontraba en dispositivos con certificación CC EAL 5+, que son utilizados en una extensa gama de aplicaciones reales, incluidas tarjetas de identidad, pasaportes y tokens para autenticación o firma de software.

487. Se identificaron, de esta forma, decenas de miles de claves con esta debilidad. Los autores estimaron que la cantidad de dispositivos afectados era del orden de decenas de millones. Además, en el peor de los casos, la factorización de claves de 1024 y 2048 bits precisaba de menos de 3 meses de CPU y 100 años de CPU en un solo núcleo de CPU, respectivamente. Pero además, todas las claves susceptibles de ser atacadas contienen una característica digital que es verificable en microsegundos en un ordenador común, por lo que todas las claves vulnerables se pueden identificar de forma inmediata.
488. Los resultados del ataque ROCA obligó a varios gobiernos europeos a revocar todos los certificados digitales de millones de tarjetas de identificación de sus ciudadanos, ya que tenían claves de 1024 bits y se podía suplantar la identidad de sus ciudadanos. En España se optó por la misma medida de prevención, aunque los DNle españoles no corrían el mismo peligro que los documentos de identidad utilizados en otros países, ya que el DNle español utiliza claves de 2048 bits.

## 9. CRIPTOGRAFÍA POSTCUÁNTICA

489. En esta sección se aborda la criptografía conocida como postcuántica. A pesar de que en el momento de escribir y publicar esta guía no ha concluido el proceso para la determinación de lo que serán, previsiblemente, los nuevos estándares del NIST de la criptografía, resistentes a la amenaza que supone la computación cuántica, parece oportuno señalar cuáles son sus premisas y puntos de partida y cuáles son las propuestas candidatas que optan con mayores posibilidades a constituirse como nuevos estándares. En la fecha de la publicación de esta guía, el NIST ha publicado la lista de los algoritmos seleccionados que han superado la tercera ronda y ha abierto una cuarta ronda. De los algoritmos seleccionados, uno pertenece a la categoría PKE/KEM, se trata de CRYSTALS-Kyber, y los restantes son de la categoría de firmas digitales, son CRYSTALS-Dilithium, Falcon y SPHINCS<sup>+</sup>. Con la apertura de una cuarta ronda, el NIST no ha descartado completamente otras propuestas, de modo que otros cuatro algoritmos deberán seguir siendo analizados. Se trata de BIKE, HQC, Classic McEliece y SIKE. Se hará especial hincapié en los algoritmos seleccionados que han superado la tercera ronda y se tendrán en cuenta los que aún deberán ser analizados en la cuarta. Los primeros ya han sido incluidos en los capítulos anteriores que les corresponden, como mecanismos autorizados.

### 9.1. LA AMENAZA CUÁNTICA (§10.16)

490. En esta sección se hará una breve introducción a las razones que han dado lugar al nacimiento de la denominada criptografía postcuántica (*Post-Quantum Cryptography*, PQC), cuyo principal objetivo es el de resistir a la amenaza real que supone la enorme potencia de los ordenadores cuánticos. Por otra parte, a lo largo de las siguientes secciones se describirán, brevemente, las herramientas matemáticas y los correspondientes problemas matemáticos<sup>11</sup> utilizados para garantizar la seguridad de estos nuevos protocolos.

491. A día de hoy, la comunidad criptográfica ha aceptado que la computación cuántica constituye una seria amenaza para la criptografía actual [50]. De hecho, es bien sabido que los algoritmos cuánticos propuestos por Shor [195] permiten vulnerar los principales mecanismos asimétricos utilizados hoy en día. La cuestión es que si se desarrolla un ordenador cuántico con suficiente capacidad de cómputo, los problemas matemáticos en los que se basa la seguridad de estos mecanismos, como la factorización de números enteros (véase §3.1.1) y el cálculo de logaritmos discretos (véanse §3.1.2 y §3.1.3) podrían resolverse solo en unas pocas horas. De hecho, si un ordenador actual necesita  $\mathcal{O}\left(2^{\sqrt[3]{\log n}}\right)$  operaciones bit para romper un

<sup>11</sup>Consideramos que la escasa divulgación y la falta de conocimiento por no especialistas de estas herramientas y problemas —en algunos casos, más complejos que los empleados hasta ahora— hace necesaria la inclusión de unas definiciones preliminares que no obliguen al lector a tener que recurrir a libros especializados.

algoritmo, un ordenador cuántico, usando el algoritmo de Shor, reduciría ese número de operaciones bit a  $\mathcal{O}(\log^3 n)$  con un almacenamiento de memoria de  $\mathcal{O}(\log n)$  bits.

492. En relación con los mecanismos simétricos (véase §2.1), los algoritmos de Grover [79], [78] y Simon [196] reducirían el tiempo de cálculo necesario para romperlos a la raíz cuadrada del tiempo actual. Esto es, si se desarrolla un ordenador cuántico con la capacidad de cómputo suficiente, la seguridad de los mecanismos simétricos actuales sería equivalente a la de los mismos mecanismos con claves de longitud la mitad. Dicho de otro modo, si un PC actual necesita  $\mathcal{O}(n)$  operaciones bits para romper uno de estos mecanismos, con el algoritmo de Grover este tiempo se reduciría a  $\mathcal{O}(\sqrt{n})$  operaciones bits y requeriría un almacenamiento en memoria de  $\mathcal{O}(\log n)$  bits.

### 9.1.1. CONVOCATORIA DEL NIST

493. Debido a la amenaza de la computación cuántica ya comentada, el NIST lanzó en 2017 una Convocatoria Internacional para seleccionar nuevos algoritmos criptográficos resistentes a esta computación cuántica [165]. Los únicos algoritmos afectados por esta convocatoria son los cifrados asimétricos, los mecanismos de encapsulamiento de clave y las firmas digitales. En definitiva, el objetivo final de este proceso de estandarización para lograr algoritmos resistentes a la computación cuántica es el de actualizar las siguientes publicaciones del NIST: [162], [166] y [167]. Por ello en este capítulo de la guía no se tratarán los mecanismos simétricos que puedan proponerse en un futuro y que sean resistentes a la computación cuántica.
494. La seguridad de los algoritmos considerados en la convocatoria del NIST se fundamenta en determinados problemas matemáticos basados en cinco primitivas:
- Códigos correctores de errores, que dan lugar a la criptografía basada en códigos (*Coded-based cryptography*)
  - Funciones resumen, que proporcionan la criptografía basada en resúmenes (*Hash-based cryptography*)
  - Polinomios multivariantes cuadráticos, que dan lugar a la llamada criptografía multivariante cuadrática (*Multivariate Quadratic Cryptography*, MQC)
  - Retículos, dando pie a la criptografía basada en retículos (*Lattice-based cryptography*)
  - Isogenias definidas sobre curvas elípticas, que proporcionan la criptografía basada en isogenias (*Isogeny-based cryptography*).

495. En julio de 2022, el NIST publicó [171] la lista de algoritmos seleccionados (a expensas de una cuarta ronda) y que listamos en las Tablas 9.1 y 9.2, junto con sus primitivas matemáticas asociadas.

Criptosistema asimétrico y KEM	Primitiva matemática
CRYSTALS-Kyber	Retículo, MLWE

**Tabla 9.1: Propuesta KEM seleccionada después de la tercera ronda y Primitiva matemática asociada**

Firma digital	Primitiva matemática
CRYSTALS-Dilithium	Retículo, MLWE
Falcon	Retículo, SIS
SPHINCS <sup>+</sup>	Hash

**Tabla 9.2: Propuestas a firmas seleccionadas después de la tercera ronda y Primitivas matemáticas asociadas**

496. Como ya se han mencionado, existen cuatro algoritmos que el NIST ha mantenido para ser estudiados en una cuarta ronda. Los mismos se listan en la Tabla 9.3.

Criptosistema asimétrico y KEM	Primitiva matemática
Classic McEliece	Código Goppa
BIKE	Código de densidad moderada cuasi-cíclico
HQC	Código cuasi-cíclico de Hamming
SIKE	Isogenias sobre curvas elípticas

**Tabla 9.3: Candidatos a KEM para ser analizados en la cuarta ronda y Primitivas matemáticas asociadas**

497. En los capítulos anteriores de esta guía hemos incluido los mecanismos criptográficos que a nivel nacional se consideran autorizados, porque ofrecen una seguridad adecuada. Debe notarse que algunos de ellos podrán ser vulnerados si la computación cuántica acaba teniendo la potencia de cálculo necesaria. Por ello, su papel pasará a ser el de los nuevos estándares que se consideren tanto por el NIST como por la comunidad internacional y española.

498. Dado que el NIST está a punto de concluir y publicar los resultados de su convocatoria sobre PQC, en este capítulo solo hemos incluido las propuestas que

se han superado la tercera ronda o que siguen considerándose para ser evaluadas en la cuarta. Por ello, hemos añadido en este capítulo las primitivas matemáticas consideradas en la convocatoria del NIST. No obstante, no se ha incluido la primitiva de isogenias sobre curvas elípticas dado que, aunque en la fecha de la publicación de los candidatos que habían superado la tercera ronda, SIKE era un candidato incluido, todo apunta a que no será tenido en cuenta en el futuro, dado que se ha encontrado un ataque de recuperación de clave eficiente para SIKEp434 (nivel de seguridad 1) utilizando un procesador de un solo núcleo en, aproximadamente, una hora [49]. No obstante, a la anterior consideración se hará una excepción. Se trata de la propuesta FrodoKEM.

499. Para el CCN es de especial importancia el estudio de los algoritmos de acuerdo de clave, por lo que considera, además, del CRYSTALS-Kyber, seleccionado por el NIST y mostrado en la Tabla 9.1, el algoritmo basado en retículos no estructurados FrodoKEM (presentado en la Tabla 9.4), que puede considerarse como una de las opciones más conservadoras con relación a su seguridad.

Criptosistema asimétrico y KEM	Primitiva matemática
FrodoKEM (Round 3)	Retículo, LWE

**Tabla 9.4: Propuesta KEM de interés para el CCN**

### 9.1.2. SEGURIDAD

500. Una de las primeras consideraciones a tener en cuenta para determinar nuevos estándares para la PQC es su seguridad. Esto es, cualquier nuevo mecanismo criptográfico que se proponga debe ser seguro frente a ataques clásicos y a ataques cuánticos.
501. Las «pruebas de reducción» han sido las principales herramientas utilizadas para lograr confianza en un mecanismo criptográfico. Hablando informalmente, estas pruebas de reducción consisten en considerar un problema matemático difícil de resolver (supuestamente), de modo que la afirmación de que un mecanismo criptográfico es demostrablemente seguro frente a una definición de seguridad significa que se puede probar que romper el mecanismo criptográfico implica resolver (en tiempo polinómico) el problema difícil considerado. Dicho de otro modo, la seguridad del mecanismo queda «reducida» a la seguridad del problema matemático.
502. La seguridad clásica de un mecanismo criptográfico, para un determinado conjunto de parámetros y tamaños de clave, se estima en bits. Para los mecanismos asimétricos (criptosistemas, protocolos de acuerdo de clave y firmas digitales) se han introducido diferentes objetivos de seguridad, entre los que destacamos, de forma muy resumida, los siguientes:

- Ataque unidireccional (*One-Way Attack*, OWA): dado un texto cifrado correspondiente a un texto claro aleatorio y su clave pública, un atacante no es capaz de encontrar el texto claro original. Una versión de este ataque es el ataque unidireccional al texto claro elegido (*One-way against Chosen Plaintext Attack*, OW-CPA).
- Indistinguibilidad por ataque al texto claro elegido (*INDistinguishability under Chosen Plaintext Attack*, IND-CPA): dado el cifrado de uno de dos textos claros elegidos por un atacante y su clave pública, el atacante no es capaz de determinar cuál de los textos claros corresponde al texto cifrado dado. Esta definición se aplica a los mecanismos cuando se utilizan claves públicas de un solo uso.
- Indistinguibilidad por ataque (adaptativo) al texto cifrado elegido (*INDistinguishability under (Adaptive) Chosen Ciphertext Attack*, (IND-CCA2) IND-CCA1): el atacante bajo las condiciones del IND-CPA no es capaz de llevar a cabo el ataque con éxito incluso si además se le da acceso a un oráculo de descifrado. El oráculo se puede utilizar en todos los textos cifrados excepto en el texto cifrado del desafío.

En la definición no adaptativa (IND-CCA1), el adversario puede consultar el oráculo solo hasta que recibe el texto cifrado de desafío. En el caso adaptativo (IND-CCA2), el adversario puede continuar consultando el oráculo de descifrado incluso después de haber recibido un texto cifrado de desafío, con la advertencia de que no puede pasar el texto cifrado de desafío para el descifrado. Estas definiciones se aplican a los mecanismos cuando se utilizan claves públicas estáticas.

- Existencialmente infalsificable bajo un ataque adaptativo al mensaje elegido (*Existentially UnForgeable under adaptive Chosen Message Attack*, EUF-CMA): un atacante no puede falsificar nuevas firmas para una clave pública dada, incluso si tiene la capacidad de ver firmas en los mensajes elegidos por el atacante.

503. La seguridad cuántica, por otra parte, mide la seguridad de un mecanismo criptográfico frente a ataques cuánticos. A día de hoy, no está claro cómo estimar la seguridad de un mecanismo criptográfico frente a ataques cuánticos.

504. Lo que se pretende con la criptografía postcuántica es que se utilice allá donde se usa la criptografía asimétrica en la actualidad, esto es, los futuros mecanismos estándares postcuánticos deberían reemplazar directamente a los actuales.

505. Para un mecanismo postcuántico (PQ) es importante considerar el tamaño de la clave, el del texto cifrado o el de la firma. En general, estos tamaños para la PQC son mucho mayores que los empleados en la actualidad, lo que puede llegar a ser un problema en aplicaciones donde la transmisión se realiza a través de canales de

comunicación con ancho de banda limitado. Por otra parte, además de la velocidad de procesamiento, también se debe considerar la velocidad en la generación de claves, en el cifrado y el descifrado, y en la generación de firma y su verificación. Para estos mecanismos también se considera como una propiedad deseada el secreto perfecto persistente y la garantía de seguridad para resistir ataques de canal lateral.

## 9.2. CRIPTOGRAFÍA BASADA EN RETÍCULOS

506. Presentaremos, en primer lugar, unas nociones elementales sobre retículos (*lattices*) y los principales problemas definidos en esta estructura, para posteriormente comentar las propuestas que han superado la tercera ronda del NIST.

### 9.2.1. RETÍCULOS

507. Dada una base de vectores,  $\mathbf{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_m\}$ , un «retículo»,  $\mathcal{L}$ , es un subespacio vectorial formado por las combinaciones lineales enteras (no reales) de los elementos de la base,  $\mathbf{b}_i$ :

$$\mathcal{L} = \left\{ \sum_{i=1}^m a_i \cdot \mathbf{b}_i : a_i \in \mathbb{Z} \right\} = \{\mathbf{B} \cdot \mathbf{a} : \mathbf{a} \in \mathbb{Z}\}.$$

508. El conjunto  $\mathbf{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_n\}$  se denomina base del retículo. El retículo generado por la base  $\mathbf{B}$  se denota por  $\mathcal{L}(\mathbf{B})$ .

509. Como un retículo es la versión discreta de un subespacio vectorial, se puede hablar del elemento no nulo más pequeño del retículo. Se llama mínimo no nulo (distancia mínima) de un retículo a

$$\lambda_1(\mathcal{L}) = \min \{ \|\mathbf{x}\| : \mathbf{x} \in \mathcal{L}, \mathbf{x} \neq 0 \}.$$

510. Los principales problemas, computacionalmente difíciles, que se consideran en retículos se definen en las Notas 101–109.

Nota 101 **[Problema del vector más corto]** (*Shortest Vector Problem, SVP*)

511. Consiste en encontrar un vector no nulo,  $\mathbf{x} \in \mathcal{L}$ , que sea el más corto de  $\mathcal{L}$ , es decir,  $\|\mathbf{x}\| \leq \|\mathbf{y}\|$ , para todo  $\mathbf{y} \in \mathcal{L}$ , esto es,  $\|\mathbf{x}\| = \lambda_1(\mathcal{L})$ .

512. Nota 102 [**Problema del vector más cercano**] (*Closest Vector Problem, CVP*) Dado  $\mathcal{L}$  en  $\mathbb{R}^n$  y un vector  $\mathbf{x} \in \mathbb{R}^n$  de modo que  $\mathbf{x} \notin \mathcal{L}$ , este problema consiste en encontrar un vector  $\mathbf{y} \in \mathcal{L}$  de modo que  $\|\mathbf{x} - \mathbf{y}\| \leq \|\mathbf{x} - \mathbf{z}\|$  para todo  $\mathbf{z} \in \mathcal{L}$ .

513. Nota 103 [**Problema de la solución entera más corta**] (*Short Integer Solution, SIS*) Dados  $m$  vectores uniformemente aleatorios  $\mathbf{a}_i \in \mathbb{Z}_q^n$  que considerados como vectores columnas definen una matriz  $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ , este problema consiste en encontrar un vector no nulo  $\mathbf{z} \in \mathbb{Z}^m$  de norma  $\|\mathbf{z}\| \leq \varepsilon$  tal que

$$\mathbf{A} \cdot \mathbf{z} = \sum_{i=1}^m \mathbf{a}_i \cdot z_i = 0 \in \mathbb{Z}_q^n.$$

Dicho de otro modo, dados muchos elementos uniformemente aleatorios de un cierto grupo aditivo finito grande, el  $\text{SIS}_{n,q,\varepsilon,m}$  trata de encontrar una combinación lineal de ellos, con enteros suficientemente cortos, que sume cero.

514. Nota 104 [**Problema del aprendizaje con errores**] (*Learning With Errors, LWE*) Este problema se parametriza por un entero  $n$ , un número primo  $q \geq 2$  y una distribución de probabilidad  $\chi$  sobre  $\mathbb{Z}_q$ . Típicamente  $\chi$  es una distribución normal de media  $\nu$  y desviación estándar  $\delta$ :

$$\chi = G(x) = \frac{1}{\delta\sqrt{2\pi}} \exp^{-\frac{1}{2}\left(\frac{x-\nu}{\delta}\right)^2}.$$

Una distribución  $A_{\mathbf{s},\chi}$  de un problema LWE sobre  $\mathbb{Z}_q^n \times \mathbb{Z}_q$  se muestrea eligiendo uniforme y aleatoriamente  $\mathbf{a} \leftarrow_{\$} \mathbb{Z}_q^n$ ,  $e \leftarrow_{\$} \mathbb{Z}_q$ , y considerando como salida el par  $(\mathbf{a}, b)$ , siendo  $b = \langle \mathbf{s}, \mathbf{a} \rangle + e \pmod{q}$ .

515. Existen dos versiones del problema LWE: búsqueda y decisión, que se definen a continuación.

Nota 105 [Problema del aprendizaje con errores (búsqueda)] Dadas  $m$  muestras independientes  $(\mathbf{a}_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$  de  $A_{\mathbf{s}, \chi}$  con  $b_i = \langle \mathbf{s}, \mathbf{a}_i \rangle + e_i \pmod{q} \in \mathbb{Z}_q$ , se trata de encontrar el vector secreto  $\mathbf{s} \in \mathbb{Z}_q^n$ , siendo  $m > n$ .

La idea es determinar el valor  $\mathbf{s}$  a partir de  $m$  muestras dadas:

$$516. \quad \begin{aligned} \mathbf{a}_1 &\stackrel{\$}{\leftarrow} \mathbb{Z}_q^n, & b_1 &= \langle \mathbf{s}, \mathbf{a}_1 \rangle + e_1 \pmod{q}, \\ \mathbf{a}_2 &\stackrel{\$}{\leftarrow} \mathbb{Z}_q^n, & b_2 &= \langle \mathbf{s}, \mathbf{a}_2 \rangle + e_2 \pmod{q}, \\ & & & \vdots \\ \mathbf{a}_m &\stackrel{\$}{\leftarrow} \mathbb{Z}_q^n, & b_m &= \langle \mathbf{s}, \mathbf{a}_m \rangle + e_m \pmod{q}. \end{aligned}$$

Nota 106 [Problema del aprendizaje con errores (decisión)] En este problema, el objetivo es distinguir entre dos pares de vectores  $(\mathbf{a}_i, b_i)$  y  $(\bar{\mathbf{a}}_i, \bar{b}_i)$ , donde  $\mathbf{a}_i, \bar{\mathbf{a}}_i \stackrel{\$}{\leftarrow} \mathbb{Z}_q^n$ ,  $b_i = \langle \mathbf{s}, \mathbf{a}_i \rangle + e_i \pmod{q}$  y  $\bar{b}_i \in \mathbb{Z}_q$ .

517. Es decir, se trata de decidir, para un par de vectores dados, si el segundo vector es el producto escalar del primer vector por algún vector secreto,  $\mathbf{s}$ , sumado con algún error, o si es el segundo vector es uniformemente aleatorio.

518. Es posible modificar el problema LWE dotándolo de una estructura de anillo o de módulo [181], definiendo entonces nuevos problemas.

Nota 107 [Problema LWE sobre anillos] (Ring Learning With Errors, RLWE) Se considera el anillo de polinomios,  $R = \mathbb{Z}[x]/(\ell(x))$ , de grado  $n$  sobre  $\mathbb{Z}$ . En general se considera  $\ell(x) = x^n - 1$  si  $n$  es primo o  $\ell(x) = x^n + 1$  si  $n$  es una potencia de 2. Se considera además, el anillo cociente  $R_q = R/qR$ , siendo  $q$  impar y suficientemente grande (hacer módulo  $q$ ).

519. El problema RLWE se parametriza por el anillo  $R$ , de grado  $n$  sobre  $\mathbb{Z}$ , un módulo entero positivo,  $q$ , que define el anillo cociente  $R_q$ , y una distribución de error,  $\chi$ , sobre  $R$ .

Dado un secreto  $s \in R_q$ , una distribución RLWE,  $A_{s, \chi}$ , sobre  $R_q \times R_q$  se muestrea eligiendo uniforme y aleatoriamente  $a \in R_q$ , con  $e \leftarrow \chi$  y considerando como salida  $(a, b = \langle s, a \rangle + e \pmod{q})$ .

Nota 108 [Problema LWE sobre anillos (decisión)] En este caso, se trata de distinguir entre muestras de RLWE y muestras uniformemente aleatorias. El problema se parametriza, además, con el número de muestras disponibles,  $m$ .

520. Este problema consiste en: dadas  $m$  muestras independientes  $(a_i, b_i) \in R_q \times R_q$ , distinguir si cada muestra se distribuye de acuerdo a una de las dos opciones siguientes: (1)  $A_{s,\chi}$  para un  $s \in R_q$  uniformemente aleatorio (fijo para todas las muestras), o (2) una distribución uniforme, sin una ventaja apreciable.

Nota 109 [Problema LWE sobre módulos] (Module Learning With Errors, MLWE) Este problema es análogo al problema RLWE pero en lugar de considerar la estructura subyacente de anillo, se considera la estructura de módulo<sup>12</sup>.

522. El problema RLWE es más compacto que el LWE debido a la estructura de anillo subyacente lo que permite implementaciones más eficientes pero, por el contrario, esta misma estructura lo hace más vulnerable a determinados ataques. Análogamente sucede con el problema MLWE que, al definirse sobre una estructura de módulo es más compacto que el LWE pero menos que el RLWE. Por ello, se considera menos vulnerable que el RLWE pero más propenso a ataques que el LWE.

### 9.2.2. CRYSTALS-KYBER

Bos et al. (véanse [33] y [34]) propusieron un sucesor de NewHope (véanse [8] y [9]) llamado Kyber. La principal modificación de Kyber con relación a otras propuestas basadas en el problema RLWE es el uso del problema MLWE. En resumen, se trabaja en  $R_q^k$  en lugar de en  $R_q$ , siendo  $k = 2, 3, 4$ , y la dimensión  $n$  de  $R_q$  es fija e igual a 256. Dado que la dimensión de  $R_q$  es más pequeña que la de NewHope, se puede usar un módulo más pequeño. En la versión de la Ronda 1 de Kyber, el módulo era  $q = 7681$ , pero en la versión de la Ronda 2, los autores redujeron el módulo a 3329 y lograron una aceleración de la transformada teórica de números (*Number Theoretic Transform*, NTT).

CRYSTALS-Kyber (CRYSTALS viene de *Cryptographic Suite for Algebraic Lattices*) es un protocolo KEM basado en retículos [191] en el que la seguridad del PKE subyacente se basa en la dificultad de resolver el problema MLWE. La propuesta tiene una seguridad IND-CCA2, que está soportada por una prueba de seguridad en el modelo del oráculo aleatorio cuántico (*Quantum Random Oracle Model* o QROM). La estructura del módulo de CRYSTALS-Kyber está definida sobre un anillo ciclotómico potencia de dos, lo que permite cálculos rápidos a través de la NTT. El esquema tiene un excelente rendimiento general para la mayoría de las aplicaciones.

Con el fin de presentar el mecanismo de PKE/KEM Kyber de modo completo, en primer lugar mostramos los algoritmos relativos al PKE (que denotaremos como KyberPKE), para luego presentar los propios del KEM. Los algoritmos de generación de claves, cifrado y descifrado de KyberPKE se muestran como Algoritmo 28, 29 y 30, respectivamente.

Con relación al PKE, los parámetros  $\mu_1$  y  $\mu_2$  (véase el Algoritmo 28) definen los errores y ambos tienen el mismo valor para los conjuntos de parámetros de Kyber-768 y Kyber-1024, pero diferentes para Kyber-512.

---

**Algoritmo 28** KyberPKE: Generación de claves  $\mathcal{G}'$ 

---

1. Genera  $seed_A$
  2.  $A \leftarrow_{PSR} R_q^{k \times k}$
  3.  $s \leftarrow \beta_{\mu_1}(R_q^k)$
  4.  $e \leftarrow \beta_{\mu_1}(R_q^k)$
  5.  $b = As + e$
  6.  $pk = (b || seed_A)$
  7.  $sk = s$
  8. **Devuelve**  $(pk, sk)$
- 

---

**Algoritmo 29** KyberPKE: Cifrado  $\mathcal{E}(pk, m, r)$ 

---

1. Genera  $seed_A$
  2.  $A^T \leftarrow_{PSR} R_q^{k \times k}$
  3.  $r \leftarrow \beta_{\mu_1}(R_q^k)$
  4.  $e_1 \leftarrow \beta_{\mu_2}(R_q^k)$
  5.  $e_2 \leftarrow \beta_{\mu_2}(R_q)$
  6.  $u = A^T r + e_1$
  7.  $v = b^T r + e_2 + m$
  8.  $c = (u || v)$
  9. **Devuelve**  $c$
-

---

**Algoritmo 30** KyberPKE: Descifrado  $\mathcal{D}(sk, c)$ 

---

1.  $m = v - s^T u$
  2. **Devuelve**  $m$
- 

Las funciones hash  $G$ ,  $H$  y la función de derivación de clave  $KDF$  son, en general, las siguientes:  $H$  es SHA3-256,  $G$  es SHA3-512 y  $KDF$  es SHAKE-256 (véase sección 2.2.3).

Los algoritmos de generación de claves, encapsulado y desencapsulado correspondientes a Kyber se muestran como Algoritmos 31, 32 y 33, respectivamente.

---

**Algoritmo 31** Kyber: Generación de claves  $\mathcal{G}$ 

---

1.  $z \leftarrow \{0, 1\}^{256}$
  2.  $(pk, sk') \leftarrow \mathcal{G}'$
  3.  $sk = (sk' || pk || H(pk) || z)$
  4. **Devuelve**  $(pk, sk)$
- 

---

**Algoritmo 32** Kyber: Encapsulado  $\mathcal{E}_c(pk)$ 

---

1.  $m' \leftarrow \{0, 1\}^{256}$
  2.  $m \leftarrow H(m')$
  3.  $(\bar{K}, r) \leftarrow (m || H(pk))$
  4.  $c \leftarrow \mathcal{E}(pk, m, r)$
  5.  $K = KDF(\bar{K} || H(c))$
  6. **Devuelve**  $(c, K)$
-

**Algoritmo 33** Kyber: Desencapsulado  $\mathcal{D}_c(sk, c)$ 

1.  $h = sk + 24 \cdot k \cdot n/8 + 32$
2.  $z = sk + 24 \cdot k \cdot n/8 + 64$
3.  $\tilde{m} \leftarrow \mathcal{D}(sk, c)$
4.  $(\bar{K}', r') = G(\tilde{m} || h)$
5.  $c' \leftarrow \mathcal{E}(pk, \tilde{m}, r')$
6. **if**  $c = c'$  **then Devuelve**  $K = KDF(\bar{K}' || H(c))$
7. **else Devuelve**  $K = KDF(z || H(c))$

**9.2.3. FRODOKEM**

FrodoKEM es un KEM basado en un esquema de cifrado que fundamenta su seguridad en el problema LWE. Este algoritmo se distingue de otros algoritmos basados en retículos en que no utiliza una estructura de anillo o módulo, lo que hace que el algoritmo gane en seguridad, pero pierda en funcionalidad y que tenga longitudes de claves más grandes.

FrodoKEM está diseñado considerando una función hash que toma como entradas un texto claro elegido al azar y el hash de la clave pública, y genera una cadena de bits grande. El hecho de introducir la clave pública en la generación de  $r$  o  $K$  no supone un cambio relevante desde el punto de vista de la seguridad y tiene la ventaja de proporcionar una seguridad multiobjetivo más fuerte [143]. En relación con la seguridad de FrodoKEM, se ha demostrado [35, Th.5.1] que la seguridad IND-CCA del KEM resultante se reduce a la seguridad IND-CPA del PKE subyacente. La prueba de seguridad de FrodoKEM en QROM radica en los resultados de Jian et al. [104].

Los algoritmos de generación de claves, encapsulado y desencapsulado se muestran como Algoritmos 34, 35 y 36, respectivamente. Se puede apreciar que en este caso se emplea un doble hash que involucra a la clave pública para generar la entrada aleatoria del algoritmo de cifrado.

**Algoritmo 34** Frodo: Generación de claves  $\mathcal{G}$ 

1.  $(pk, sk) \leftarrow \mathcal{G}$
2.  $s \leftarrow_R \{0, 1\}^{len_s}$
3.  $phk = H_1(pk)$
4.  $sk' = (sk, s, pk, phk)$
5. **Devuelve**  $(pk, sk')$

**Algoritmo 35** Frodo: Encapsulado  $\mathcal{E}_c(pk)$ 

1.  $m \leftarrow MSPC$
2.  $(r, k) \leftarrow H_2(H_1(pk)||m)$
3.  $c \leftarrow \mathcal{E}(pk, m; r)$
4.  $K = H_3(c||k)$
5. **Devuelve**  $(c, K)$

**Algoritmo 36** Frodo: Desencapsulado  $\mathcal{D}_c(c, sk')$ 

1.  $m' \leftarrow \mathcal{D}(sk, c)$
2.  $(r', k') = H_2(phk||m)$
3.  $K'_0 = H_3(c||k')$
4.  $K'_1 = H_3(c||s)$
5. **if**  $c = \mathcal{E}(m', pk; r')$  **then**  $K' = K'_0$
6. **else**  $K' = K'_1$
7. **Devuelve**  $K'$

**9.2.4. CRYSTALS-DILITHIUM**

CRYSTALS-Dilithium es un esquema de firma digital que sigue las pautas de las firmas de Fiat-Shamir [128]. Fue propuesto por Ducas et al. [62] y se puede considerar como una versión mejorada de los esquemas propuestos en [126], [80], [61] y [20]). La característica principal de Dilithium es que su seguridad se basa en el problema MLWE.

CRYSTALS-Dilithium es uno de los dos esquemas de firma basados en retículos incluidos en la tercera ronda del NIST [127]. Su seguridad se basa en la fortaleza del problema MLWE y del problema en módulos de soluciones enteras cortas (*Module Short Integer Solutions*, MSIS). Dilithium usa el mismo módulo y anillo para todos los conjuntos posibles de parámetros y obtiene las muestras de una distribución uniforme. Estas características hacen que sea relativamente fácil de implementar. Además, tiene un rendimiento equilibrado entre los tamaños de clave y firma y es eficiente con los algoritmos de generación, firma y verificación de claves.

Como en cualquier esquema de firma, el proceso completo consta de tres algoritmos: Generación de claves, Elaboración de la firma digital y verificación de la misma.

El Algoritmo 37 muestra el proceso para la generación de las claves pública,  $pk$ , y privada,  $sk$ . Este algoritmo genera una matriz  $A$  de tamaño  $k \times l$  con entradas en el anillo de polinomios  $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$ , siendo  $q = 2^23 - 2^13 + 1$  y  $n = 256$ . A continuación se determinan dos vectores aleatorios  $s_1$  y  $s_2$ , siendo cada coeficiente de estos vectores un elemento de  $R_q$  con coeficientes pequeños, de tamaño a lo sumo  $\eta$ . Finalmente, se calculan la clave pública,  $pk$ , y la privada,  $sk$ .

---

**Algoritmo 37** Dilithium: Generación de claves

---

1.  $A \leftarrow_R \mathcal{R}_q^{k \times l}$
  2.  $(s_1, s_2) \leftarrow_R \mathcal{S}_\eta^l \times \mathcal{S}_\eta^k$
  3.  $t := As_1 + s_2$
  4. La clave pública es  $pk = (A, t)$  y la clave privada es  $sk = (A, t, s_1, s_2)$
- 

523. Para la elaboración de la firma (véase el Algoritmo 38), en primer lugar se genera un vector de enmascaramiento del polinomio,  $y$ , con coeficientes menores que  $\gamma_1$ . El firmante calcula  $Ay$  y define los bits de orden superior como  $w_1$ . En particular, cada coeficiente  $\alpha$  en  $Ay$  se puede escribir de forma canónica como  $\alpha = w_1 \cdot 2\gamma_2 + w_0$  donde  $|w_0| \leq \gamma_2$ . El desafío  $c$  se crea como el hash del mensaje y  $w_1$ . La salida  $c$  es un polinomio en  $\mathcal{R}_q$  con exactamente  $\tau$  coeficientes iguales a  $\pm 1$  y el resto igual a 0. La razón de esta distribución es que  $c$  tiene una norma pequeña y proviene de un dominio de tamaño  $\log_2 \binom{256}{\tau} + \tau$ , que estaría entre 128 y 256. La firma se calcula como  $z = y + cs_1$ . Como entrada al algoritmo se consideran la clave privada y el mensaje a firmar:  $sk, M$ .

524. Si el algoritmo de firma generara la firma antes de que sea calculada, la clave privada podría quedar expuesta. Para evitar esta debilidad se utiliza el muestreo por rechazo. El parámetro  $\beta$  está configurado para ser el coeficiente máximo posible de  $cs_i$ . Como  $c$  tiene  $\tau$  coeficientes iguales a  $\pm 1$  y el coeficiente máximo de  $s_i$  es  $\eta$ , entonces  $\beta \leq \tau \cdot \eta$ . Si cualquier coeficiente de  $z$  es mayor que  $\gamma_1 - \beta$ , entonces se rechaza la firma y se inicia el procedimiento de firma de nuevo. Además, si algún

coeficiente de los bits de orden inferior de  $Az - ct$  es mayor que  $\gamma_2 - \beta$ , el algoritmo también se reinicia. El primer control es necesario por seguridad, mientras que el segundo es necesario por seguridad y corrección. El algoritmo de verificación de la firma se muestra como Algoritmo 39, que tiene como entradas la clave pública, el mensaje y la firma:  $(pk, M, \sigma = (z, c))$ .

---

**Algoritmo 38** Dilithium: Elaboración de la firma digital
 

---

1.  $z := \perp$ .
  2. while  $z = \perp$ 
    - a)  $y \leftarrow S_{\gamma_1-1}^l$
    - b)  $w_1 := HighBits(Ay, 2\gamma_2)$
    - c)  $c \in B_\tau := H(M||w_1)$
    - d)  $z := y + cs_1$
  3. if  $\|z\|_\infty \geq \gamma_1 - \beta$  or  $\|LowBits(Ay - cs_2, 2\gamma_2)\| \geq \gamma_2 - \beta$  then  $z := \perp$
  4. La firma para el mensaje  $M$  es  $\sigma = (z, c)$ .
- 

525. Por último, en la verificación de la firma (véase el Algoritmo 39) se calcula  $w'_1$  como los bits de orden superior de  $Az - ct$  y se acepta la firma como válida si todos los coeficientes de  $z$  son menores que  $\gamma_1 - \beta$  y si  $c$  es el hash del mensaje y  $w'_1$ . Se debe notar que  $Az - ct = Ay - cs_2$ . Así que todo lo que realmente se necesita mostrar es que

$$HighBits(Ay, 2\gamma_2) = HighBits(A - cs_2, 2\gamma_2)$$

Si los algoritmos funcionan según lo establecido, cualquier firma válida cumple con  $\|LowBits(Ay - cs_2, 2\gamma_2)\|_\infty < \gamma_2 - \beta$ . Dado que los coeficientes de  $cs_2$  son más pequeños que  $\beta$ , es claro que sumar  $cs_2$  no es suficiente para aumentar cualquier coeficiente de orden inferior para que tenga una magnitud de al menos  $\gamma_2$ .

---

**Algoritmo 39** Dilithium: Verificación de la firma digital
 

---

1.  $w'_1 := HighBits(Az - ct, 2\gamma_2)$
  2. La firma se acepta si  $c = H(M||w'_1)$  y los coeficientes de  $z$  son menores que  $\gamma_1 - \beta$ .
- 

526. Cuando el NIST establezca los parámetros recomendados para el uso de Dilithium, se incluirán en esta guía, así como los algoritmos finalmente establecidos.

### 9.2.5. FALCON

527. En la actualidad, Falcon es uno de los tres esquemas de firma digital seleccionado por el NIST después de la tercera ronda [171], basada en retículos [178]. Para un estudio en profundidad de este mecanismo, véase [66]. Su nombre es un acrónimo de *FAst-Fourier Lattice-based COmpact signatures over NTRU*. La seguridad de esta propuesta se basa en el problema SIS en retículos NTRU. Las pruebas de seguridad se consideran tanto en el ROM como en el QROM con reducciones estrictas.
528. Desde un punto de vista técnico, Falcon es el resultado de muchos años de trabajo que incluyen principalmente: el esquema de firma NTRUSign [85], el marco de referencia genérico para construir esquemas de firma seguros basados en el paradigma “*hash and sign*” [75], la reducción de seguridad de NTRU [203], y una implementación práctica de la parte de IBE de la propuesta de Gentry-Peikert-Vaikuntanathan (GPV) sobre retículos NTRU [63].
529. Al igual que los restantes esquemas de firma, también Falcon está compuesto por tres algoritmos: Generación de claves, Elaboración de la firma y verificación de la misma.
530. El Algoritmo 40 muestra el algoritmo de generación de las claves pública,  $pk$ , y privada,  $sk$ . Este algoritmo emplea tres funciones: NTRUGen, FFT y fFLDL\*. La función NTRUGen está diseñada para generar los cuatro polinomios NTRU, es decir, que cumplen la siguiente ecuación

$$fG - gF = 0 \pmod{\phi},$$

siendo  $f, g, F, G \in \mathbb{Z}[x]/\phi$  y  $\phi = x^n + 1$  un polinomio mónico irreducible ciclotómico con  $n = 2^k$ . Además, NTRUGen genera aleatoriamente los polinomios  $g$  y  $f$  comprobando si cumplen las condiciones adecuadas. La primera condición es que se pueda calcular la clave pública  $h = gf^{-1} \pmod{\phi, q}$ . Luego hay una condición sobre  $g$  y  $f$  sobre la longitud de las firmas que se pueden generar con ellas. Después de esto, se usa otra función llamada NTRUSolve para calcular dos polinomios cortos

$G$  y  $F$  tales que, junto con  $g$  y  $f$ , cumplan la ecuación mostrada anteriormente. El algoritmo usa como entradas  $\phi$  y  $q$ .

---

**Algoritmo 40** Falcon: Generación de claves
 

---

1.  $f, g, F, G \leftarrow \text{NTRUGen}(\phi, q)$
  2.  $B \leftarrow \begin{bmatrix} g & -f \\ G & -F \end{bmatrix}$
  3.  $\hat{B} \leftarrow \text{FFT}(B)$
  4.  $G \leftarrow \hat{B} \times \hat{B}^*$
  5.  $T \leftarrow \text{ffLDL}^*(G)$
  6. for each leaf of  $T$  do
  7.   leaf value  $\leftarrow \sigma / \sqrt{\text{leaf value}}$
  8.  $sk \leftarrow (\hat{B}, T)$
  9.  $pk \leftarrow gf^{-1} \pmod{q}$
  10. La clave pública es  $pk$  y la privada es  $sk$
- 

531. El algoritmo para la elaboración de la firma (véase el Algoritmo 41) usa como entradas los valores de la clave privada,  $sk$ , y  $\lfloor \beta^2 \rfloor$ , y el mensaje a firmar,  $M$ , así como las funciones HashToPoint, ffSampling, invFF y Compress.

---

**Algoritmo 41** Falcon: Elaboración de la firma digital
 

---

1.  $r \leftarrow \{0, 1\}^{320}$
  2.  $c \leftarrow \text{HashToPoint}(r || M, q, n)$
  3. Cálculo de un punto del retículo (conocida la base) cercano al elemento  $c$ :  
 $z \leftarrow \text{ffSampling}_n(f, T)$
  4.  $s' = (t - z)\hat{B}$
  5. while  $\|s'\| > \lfloor \beta^2 \rfloor$  do
  6.    $(s_1, s_2) \leftarrow \text{invFF}(s')$
  7.    $s \leftarrow \text{Compress}(s_2, 8 \cdot \text{sbytelen} - 328)$
  8. La firma para el mensaje  $M$  es  $\sigma = (r, s)$
-

532. Para la verificación de la firma se emplean las funciones HashToPoint y Decompress (véase el Algoritmo 42) y como entradas los valores  $\sigma$ ,  $pk$ ,  $\lfloor \beta^2 \rfloor$ , así como el mensaje  $M$ .

---

**Algoritmo 42** Falcon: Verificación de la firma digital

---

1.  $c \leftarrow \text{HashToPoint}(r \| M, q, n)$
  2.  $s_2 \leftarrow \text{Decompress}(s, 8 \cdot \text{sbytelen} - 328)$
  3. if  $s_2 = \perp$  then devuelve "Rechazo"
  4.  $s_1 \leftarrow c - s_2 h \pmod{q}$
  5. if  $\|(s_1, s_2)\|^2 \leq \lfloor \beta^2 \rfloor$  then devuelve "Aceptación"
  6. else devuelve "Rechazo"
- 

533. En el momento que NIST dé a conocer los parámetros concretos para el uso de esta propuesta, serán incluidos en esta guía, a la vez que la descripción de sus oportunos algoritmos.

### 9.3. FIRMAS DIGITALES BASADAS EN FUNCIONES RESUMEN (HASH)

534. Como ya se ha mencionado, solo ha superado la tercera ronda del NIST una propuesta basada en funciones resumen (*hash*): SPHINCS<sup>+</sup> [88]. En esta sección no presentaremos los aspectos básicos de las funciones hash, dado que son ampliamente conocidos.

#### 9.3.1. SPHINCS<sup>+</sup>

535. SPHINCS<sup>+</sup> es un esquema de firma, inicialmente considerado como alternativo por el NIST en la tercera ronda pero ya seleccionado en 2022 después de superar la tercera ronda. Se basa en una función hash sin estado [88]. Su seguridad se fundamenta en la supuesta seguridad de la función hash subyacente considerada, es decir, en SHA-256, SHAKE-256 o en las funciones hash de Haraka. Existe una prueba de seguridad para todas las variantes propuestas de SPHINCS<sup>+</sup> en el modelo de oráculo aleatorio. Además, hay una prueba en el modelo estándar para las variantes robustas basada en suposiciones plausibles pero no estándar con respecto a una función hash modificable construida a partir del hash subyacente.

536. Dado que las funciones hash han sido ampliamente estudiadas, se considera que esta propuesta es la que, entre todos los candidatos a firma, tiene menor probabilidad de ser descifrada por métodos criptoanalíticos. Sin embargo, esto tiene

un costo sustancial en su rendimiento: SPHINCS<sup>+</sup> es más lenta y sus firmas son considerablemente más grandes que la mayoría de los otros esquemas de firma.

537. SPHINCS<sup>+</sup> ha superado la tercera ronda del NIST y en 2022 ha sido seleccionado por esta institución como esquema de firma digital [171]. Pertenece a los esquemas de HBS (véase §3.2.3) por lo que posee sus características generales. En el caso de que el esquema de firma ejecute un algoritmo de firma varias veces para elaborar la firma deseada, se suele denominar firma completa. En particular, una firma completa contiene el índice del par de claves OTS utilizado en el árbol, la clave pública OTS, la firma OTS y la ruta de autenticación. Esta ruta está formada por el conjunto de nodos que permite realizar la ruta desde la clave pública OTS hasta la raíz. En el caso de que se utilice un OTS con Winternitz (W-OTS), la clave pública de OTS se puede calcular a partir de la firma de OTS, en cuyo caso, la clave pública de OTS puede omitirse en la firma completa. Para garantizar que cada par de claves de OTS se use solo una vez, los pares de claves OTS se usan en un orden predefinido, usando las hojas del árbol de izquierda a derecha. Para verificar la firma, se comprueba la firma OTS en el mensaje y se verifica la autenticidad del par de claves OTS comprobando si la clave pública es consistente con la ruta de autenticación y el hash de la clave pública OTS.
538. Esta forma de firma genera firmas pequeñas, claves privadas pequeñas (mediante la generación pseudoaleatoria de las claves secretas OTS) y también claves públicas pequeñas. Sin embargo, la generación de claves y el tiempo de elaboración de la firma son exponenciales en el número de pisos del árbol (su altura), ya que en la generación de claves se tiene que construir el árbol completo. Para una detallada explicación de este mecanismo de firma, remitimos al lector a la documentación original en [18].
539. La ventaja de SPHINCS<sup>+</sup> es que incluye dos nuevas ideas que permiten reducir considerablemente el tamaño de la firma. En primer lugar, para aumentar el nivel de seguridad de la selección de índices aleatorios, SPHINCS<sup>+</sup> reemplaza la hoja OTS con un esquema de firma de poco tiempo o FTS (*Few-Time Signature*) basado en hash. Un FTS es un esquema de firma diseñado para firmar algunos mensajes y en el contexto de SPHINCS<sup>+</sup>, permite algunas colisiones de índices, lo que a su vez facilita una altura de árbol más pequeña para el mismo nivel de seguridad.
540. En segundo lugar, SPHINCS<sup>+</sup> considera la construcción de Goldreich como un hiperárbol con  $h$  pisos de árboles de altura 1 y la generaliza a un hiperárbol con  $d$  capas de árboles de altura  $h/d$ . Esta idea introduce una compensación entre el tamaño de la firma y el tiempo controlado por el número de capas,  $d$ . El tamaño de la firma es  $|\sigma| \approx d|\sigma_{OTS}| + hn$ , considerando que se usa una función hash con salidas de  $n$  bits. Debe recordarse que el tamaño de una firma de un solo uso,  $|\sigma_{OTS}|$ , es aproximadamente  $O(n^2)$ , por lo que al disminuir el número de capas se obtienen firmas completas más pequeñas. Por el contrario, el tiempo de firma aumenta exponencialmente en la disminución de las capas: la firma requiere  $d2^{h/d}$  generaciones de claves OTS y  $d2^{h/d} - d$  cálculos hash.

541. SPHINCS<sup>+</sup> utiliza varios parámetros y funciones. El principal parámetro de seguridad es  $n \in \mathbb{N}$ . Las funciones son las siguientes: dos funciones hash criptográficas de entrada corta  $F: \{0, 1\}^n \rightarrow \{0, 1\}^n$  y  $H: \{0, 1\}^{2n} \rightarrow \{0, 1\}^n$ ; una función hash aleatoria de entrada arbitraria  $\mathcal{H}: \{0, 1\}^n \times \{0, 1\}^* \rightarrow \{0, 1\}^m$ , para  $m = \text{poly}(n)$ ; una familia de generadores pseudoaleatorios  $G_\lambda: \{0, 1\}^n \rightarrow \{0, 1\}^{\lambda n}$  para diferentes valores de  $\lambda$ ; un conjunto de familias de funciones pseudoaleatorias  $\mathcal{F}_\lambda: \{0, 1\}^\lambda \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ , y una familia de funciones pseudoaleatorias  $\mathcal{F}: \{0, 1\}^* \times \{0, 1\}^n \rightarrow \{0, 1\}^{2n}$  que admite longitudes de entrada arbitrarias. Todas estas funciones se pueden construir a partir de una única función hash criptográfica, pero es más natural separar las funciones según sus roles.
542. El mecanismo también usa un hiperárbol (un árbol de árboles) de altura total  $h \in \mathbb{N}$ , siendo  $h$  es un múltiplo de  $d$  y el hiperárbol consta de  $d$  capas de árboles, cada una con altura  $h/d$ . Los componentes de SPHINCS<sup>+</sup> tienen parámetros adicionales que influyen en el rendimiento y en el tamaño de la firma y las claves: la firma de un solo uso de Winternitz, WOTS, permite una compensación de espacio-tiempo utilizando el parámetro de Winternitz  $w \in \mathbb{N}$ , con  $w > 1$ ; el esquema FTS tiene una compensación de espacio-tiempo que está controlada por dos parámetros  $k \in \mathbb{N}$  y  $t = 2^\tau$ , con  $\tau \in \mathbb{N}$  y  $k\tau = m$ . Al igual que la firma XMSS, SPHINCS<sup>+</sup> utiliza W-OTS<sup>+</sup>.
543. En SPHINCS<sup>+</sup>, un árbol binario de hashes de altura  $h$  siempre tiene  $2h$  hojas que son cadenas de  $n$  bits,  $L_i$ , con  $i \in [2^h - 1]$ . Cada nodo  $N_{i,j}$ , para  $0 < j \leq h$ ,  $0 \leq i < 2^{h-j}$ , del árbol almacena una cadena de  $n$  bits. Para construir el árbol, se utilizan máscaras de bits  $h$ ,  $Q_j \in \{0, 1\}^{2n}$ ,  $0 < j \leq h$ . Para los nodos hoja se define  $N_{i,0} = L_i$ . Los valores de los nodos internos  $N_{i,j}$  se calculan como

$$N_{i,j} = H((N_{2i,j-1} || N_{2i+1,j-1}) \oplus Q_j).$$

La raíz del árbol se denota como  $Root = N_{0,h}$ .

544. Una noción importante es la llamada ruta de autenticación de una hoja  $L_i$ ,  $Auth_i = (A_0, \dots, A_{h-1})$ , que son los nodos del árbol necesarios para verificar que  $L_i$  es un nodo del mismo. Así,  $Auth_i$  consta de todos los nodos hermanos de los nodos contenidos en la ruta que van de  $L_i$  a la raíz.
545. SPHINCS<sup>+</sup> puede firmar cientos de mensajes por segundo en una CPU Intel de 4 núcleos a 3,5 GHz [29]. Tanto las claves públicas como las privadas tienen un tamaño de 1 KB y las firmas son de 41 KB. El esquema de firma está diseñado para proporcionar seguridad de  $2^{128}$  a largo plazo incluso contra atacantes con ordenadores cuánticos. A diferencia de la mayoría de los diseños basados en hash, este esquema de firma no tiene estado, por lo que es un buen candidato para reemplazar los esquemas de firma actuales. En [29] se presenta una construcción de SPHINCS-256 como ejemplo de ejecución, considerando los siguientes parámetros:  $n = 256$ ;  $m = 512$ ;  $h = 60$ ;  $d = 12$ ;  $w = 16$ ;  $t = 2^{16}$ ;  $k = 32$ .

546. Una vez que NIST establezca los parámetros recomendados para el uso de SPHINCS<sup>+</sup>, se incluirán en esta guía, así como la descripción de sus correspondientes algoritmos.

## 10. TABLAS

## 10.1. PRIMITIVAS SIMÉTRICAS

Primitiva	Parámetros (bits)	Rec./Her.	Referencias	Notas
AES	$k = 128$	R	[54], [152], [94]	
	$k = 192$	R		
	$k = 256$	R		

Tabla 10.1: Tipos autorizados de cifradores en bloque (§2.1)

Primitiva	Parámetros (bits)	Rec./Her.	Referencias	Notas
SHA-2	$h = 256$ (SHA-256)	R	[135], [163], [101]	
	$h = 384$ (SHA-384)	R		
	$h = 512$ (SHA-512)	R		
	$h = 256$ (SHA-512/256)	R		
SHA-3	$h = 256$	R	[164]	
	$h = 384$	R		
	$h = 512$	R		

Tabla 10.2: Funciones resumen autorizadas (§2.1)

Primitiva	Rec./Her.	Referencias	Notas
Compartición de secretos de Shamir	R	[194]	

Tabla 10.3: Compartición de secretos autorizada (§2.1)

## 10.2. CONSTRUCCIONES SIMÉTRICAS

Modo de operación	Rec./Her.	Referencias	Notas
CTR	R*	[153], [99]	3, 4, 5
OFB	R*	[153], [99]	3, 4, 5
CBC	R*	[153], [99]	3, 4, 6
CBC-CS	R*	[161]	3, 4
CFB	R*	[153], [99]	3, 4, 6

Tabla 10.4: Modos autorizados de cifrado simétrico (§2.2)

Esquema	Rec./Her.	Referencias	Notas
XTS	R	[158]	7, 8, 9

Tabla 10.5: Modos autorizados de cifrado simétrico para discos duros (§2.2)

Esquema	Tamaño clave (bits)	Rec./Her.	Referencias	Notas
CMAC		R	[154], [95]	10, 11
CBC-MAC		R	[95, Alg. 1, Relleno 2]	10, 11, 12
HMAC	$k \geq 128$ $k \geq 100$	R H	[118], [96]	
HMAC-SHA1	$k \geq 100$	H[2030]	[118], [96], [163]	13
GMAC		R	[155]	14, 15, 16
KMAC-128		R	[113]	
KMAC-256		R	[113]	

Tabla 10.6: MAC autorizados basados en cifradores en bloque y funciones resumen (§2.2)

Tamaño del protocolo desafío-respuesta (bits)	Rec./Her.	Notas
$128 \leq l$	R	17
$96 \leq l < 128$	H	17

Tabla 10.7: Tamaño de los protocolos de desafío-respuesta autorizados (§2.2)

Esquema	Rec./Her.	Referencias	Notas
Cifrado-y luego-MAC	R	[23]	18, 19
CCM	R	[156], [102]	18, 19
GCM	R	[155], [102]	14, 15, 16, 18, 19, 20
EAX	R	[102]	18, 19
ChaCha20_Poly1305	R	[150], [28], [30]	21, 22

Tabla 10.8: Esquemas simétricos de cifrado autenticado autorizados (§2.2)

Esquema	Rec./Her.	Referencias	Notas
SIV	R	[83]	
AES-Keywrap	R	[160, Alg. KW y KWP]	

Tabla 10.9: Esquemas de protección de claves autorizados (§2.2)

Esquema	Rec./Her.	Referencias	Notas
NIST SP800-56A/B/C	R	[166], [167], [169]	
NIST SP800-108	R	[172]	
ANSI-X9.63-KDF	R	[12]	
PBKDF2	R	[142], [159]	23
SCRYPT	R	[175]	
HKDF	R	[119]	

Tabla 10.10: Funciones de derivación de claves autorizadas (§2.2)

Esquema	Rec./Her.	Referencias	Notas
PBKDF2	R	[149]	24, 25

Tabla 10.11: Mecanismos de resumen de contraseñas autorizados (§2.2)

### 10.3. PRIMITIVAS ASIMÉTRICAS

Primitiva	Tamaño de los parámetros (bits)	Rec./Her.	Referencias	Notas
RSA	$n \geq 3000, \log_2(e) > 16$ $n \geq 1900, \log_2(e) > 16$	R H[2025]	[184]	26

Tabla 10.12: Tamaño de las Primitivas RSA autorizadas (§3.1)

Familia	Tamaño del grupo (bits)	Rec./Her.	Referencias	Notas
MODP	3072 bits	R	[116]	27, 28
	4096 bits	R		
	6144 bits	R		
	8192 bits	R		
	2048 bits	H[2025]		

Tabla 10.13: Tamaño de las primitivas autorizadas del logaritmo discreto multiplicativo sobre un cuerpo finito (§3.1)

Familia de curvas	Curva	Rec./Her.	Referencias	Notas
Brainpool	BrainpoolP256r1	R	[125]	29, 30, 31
	BrainpoolP384r1	R		
	BrainpoolP512r1	R		
NIST	NIST P-256	R	[162]	29, 30, 31
	NIST P-384	R		
	NIST P-521	R		
FR	FRP256v1	R	[13]	29, 30, 31, 32
Bernstein	B1 (Curve25519)	R	[27]	29, 30, 31
Hamburg	B2 (Curve448)	R	[122]	29, 30, 31

Tabla 10.14: Curvas elípticas autorizadas (§3.1)

#### 10.4. CONSTRUCCIONES ASIMÉTRICAS

Primitiva	Esquema	Rec./Her.	Referencias	Notas
RSA	OAEP PKCS#1 v2.1	R	[141], [185]	33, 34
RSA	PKCS#1 v1.5	H	[141], [185]	33, 35

Tabla 10.15: Esquema de cifrado asimétrico autorizado (§3.2)

Primitiva	Esquema	Rec./Her.	Referencias	Notas
RSA	OAEP PKCS#1v2.1	R	[105], [93], [185]	36,37
FF-DLOG	KCDSA	R	[100]	36,37
	Schnorr		[100]	36,37,38
	DSA		[162], [100]	36,37
EC-DLOG	EC-KCDSA	R	[100]	36,37
	EC-DSA		[162], [100]	36,37,38
	EC-GDSA		[37]	36,37
	EC-Schnorr		[100]	36,37
RSA	PKCS#1v1.5	H	[105], [93], [185]	36,37,39
Hash	XMSS	R	[46], [87], [170]	
Retículo	Dilithium	R	[127]	40, 41
Retículo	Falcon	R	[178]	40, 41
Hash	SPHINCS+	R	[88]	40, 41

Tabla 10.16: Esquemas de firma digital autorizados (§3.2)

Primitiva	Esquema	Rec./Her.	Referencias	Notas
FF-DLOG	DH DLIES-KEM	R	[98], [166] [92]	42,43,44 42,45,47
EC-DLOG	EC-DH ECIES-KEM	R	[98], [166] [92]	42,43,44 42,46,47
Retículo	CRYSTALS-Kyber FrodoKEM	R	[191] [143]	48, 49 50

Tabla 10.17: Esquemas de establecimiento de claves autorizados (§3.2)

## 10.5. TLS

Protocolo	Rec./Her.	Referencias	Notas
TLS 1.3	R	[183]	53
TLS 1.2	R	[58]	53

Tabla 10.18: Versiones del protocolo TLS autorizadas (§4.1)

Código	Suites criptográficas	Rec./Her.	Referencias	Notas
0x1302	TLS_AES_256_GCM_SHA384	R	[183]	
0x1301	TLS_AES_128_GCM_SHA256	R	[183]	
0x1304	TLS_AES_128_CCM_SHA256	R	[183]	
0x1303	TLS_CHACHA20_POLY1305_SHA256	R	[121]	

Tabla 10.19: Suites criptográficas autorizadas para el protocolo TLS 1.3 (§4.1)

Código	Modo PSK	Rec./Her.	Referencias	Notas
0x0000	psk_ke	H[2026]	[183]	54, 55
0x0001	psk_dhe_ke	R	[183]	55

Tabla 10.20: Modos de clave precompartida recomendados para el protocolo TLS 1.3 (§4.1)

Código	Grupo DH	Rec./Her.	Referencias	Notas
0x0100	ffdhe2048	H[2025]	[76]	
0x0101	ffdhe3072	R	[76]	
0x0102	ffdhe4096	R	[76]	
0x0017	secp256r1 (P-256)	R	[148]	
0x0018	secp384r1 (P-384)	R	[148]	
0x001F	brainpoolP256r1tls13	R	[36]	
0x0020	brainpoolP384r1tls13	R	[36]	
0x0021	brainpoolP512r1tls13	R	[36]	

Tabla 10.21: Grupos de Diffie-Hellman recomendados para el protocolo TLS 1.3 (§4.1)

Código	Algoritmo de firma	Rec./Her.	Referencias	Notas
0x0804	rsa_pss_rsae_sha256	R	[183]	
0x0805	rsa_pss_rsae_sha384	R	[183]	
0x0806	rsa_pss_rsae_sha512	R	[183]	
0x0809	rsa_pss_pss_sha256	R	[183]	
0x080A	rsa_pss_pss_sha384	R	[183]	
0x080B	rsa_pss_pss_sha512	R	[183]	
0x0403	ecdsa_secp256r1_sha256	R	[183]	
0x0503	ecdsa_secp384r1_sha384	R	[183]	
0x081A	ecdsa_brainpoolP256r1tls13_sha256	R	[36]	
0x081B	ecdsa_brainpoolP384r1tls13_sha384	R	[36]	
0x081C	ecdsa_brainpoolP512r1tls13_sha512	R	[36]	

Tabla 10.22: Algoritmos de firma (cliente/servidor) recomendados para el protocolo TLS 1.3 (§4.1)

Código	Funciones resumen	Rec./Her.	Referencias	Notas
0x0401	rsa_pkcs1_sha256	H[2025]	[183]	56
0x0501	rsa_pkcs1_sha384	H[2025]	[183]	56
0x0601	rsa_pkcs1_sha512	H[2025]	[183]	56
0x0804	rsa_pss_rsae_sha256	R	[183]	
0x0805	rsa_pss_rsae_sha384	R	[183]	
0x0806	rsa_pss_rsae_sha512	R	[183]	
0x0809	rsa_pss_pss_sha256	R	[183]	
0x080A	rsa_pss_pss_sha384	R	[183]	
0x080B	rsa_pss_pss_sha512	R	[183]	
0x0403	ecdsa_secp256r1_sha256	R	[183]	
0x0503	ecdsa_secp384r1_sha384	R	[183]	
0x081A	ecdsa_brainpoolP256r1tls13_sha256	R	[36]	
0x081B	ecdsa_brainpoolP384r1tls13_sha384	R	[36]	
0x081C	ecdsa_brainpoolP512r1tls13_sha512	R	[36]	

Tabla 10.23: Algoritmos de firma (en certificados) recomendados para el protocolo TLS 1.3 (§4.1)

Código	Suites criptográficas	Rec./He	Referencias
0xC02C	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	R	[182]
0xC02B	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	R	[182]
0xC0AD	TLS_ECDHE_ECDSA_WITH_AES_256_CCM	R	[134]
0xC0AC	TLS_ECDHE_ECDSA_WITH_AES_128_CCM	R	[134]
0xCCA9	TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256	R	[121]

**Tabla 10.24: Suites criptográficas recomendadas para TLS 1.2 con un servidor que disponga de un certificado con la clave pública EC-DSA (§4.1)**

Código	Suites criptográficas	Rec./Her.	Referencias
0xC030	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	H	[182]
0xC02F	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	H	[182]
0xCCA8	TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	H	[121]

**Tabla 10.25: Suites criptográficas heredadas para TLS 1.2 con un servidor que disponga de un certificado con la clave pública RSA (§4.1)**

Código	Suites criptográficas	Rec./Her.	Referencias	Notas
0xC024	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	H[2025]	[182]	57
0xC023	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	H[2025]	[182]	57
0xC028	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	H[2025]	[182]	57
0xC027	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	H[2025]	[182]	57
0x009F	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	H	[186]	58
0x009E	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	H	[186]	58
0xC09F	TLS_DHE_RSA_WITH_AES_256_CCM	H	[133]	58
0xC09E	TLS_DHE_RSA_WITH_AES_128_CCM	H	[133]	58
0x006B	TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	H[2025]	[58]	57,58
0x0067	TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	H[2025]	[58]	57,58
0x009D	TLS_RSA_WITH_AES_256_GCM_SHA384	H		59
0x009C	TLS_RSA_WITH_AES_128_GCM_SHA256	H		59
0xC09D	TLS_RSA_WITH_AES_256_CCM	H		59
0xC09C	TLS_RSA_WITH_AES_128_CCM	H		59
0x003D	TLS_RSA_WITH_AES_256_CBC_SHA256	H[2025]		57,59
0x003C	TLS_RSA_WITH_AES_128_CBC_SHA256	H[2025]		57,59
0xCAA	TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256	H	[121]	
0x006A	TLS_DHE_DSS_WITH_AES_256_CBC_SHA256	H	[58]	
0x00A3	TLS_DHE_DSS_WITH_AES_256_GCM_SHA384	H	[186]	

**Tabla 10.26: Suites criptográficas recomendadas para TLS 1.2 cuando no hay soporte ECC o modo de cifrado autenticado (§4.1)**

Código	Suites criptográficas	Rec./Her.	Referencias	Notas
0xC037	TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256	R	[81]	
0xC038	TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA384	R	[81]	
0xD001	TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256	R	[131]	
0xD002	TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384	R	[131]	
0xD005	TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256	R	[131]	
0xCCAD	TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256	R	[121]	
0x00B2	TLS_DHE_PSK_WITH_AES_128_CBC_SHA256	R	[19]	
0x00B3	TLS_DHE_PSK_WITH_AES_256_CBC_SHA384	R	[19]	
0x00AA	TLS_DHE_PSK_WITH_AES_128_GCM_SHA256	R	[19]	
0x00AB	TLS_DHE_PSK_WITH_AES_256_GCM_SHA384	R	[19]	
0xC0A6	TLS_DHE_PSK_WITH_AES_128_CCM	R	[133]	
0xC0A7	TLS_DHE_PSK_WITH_AES_256_CCM	R	[133]	

**Tabla 10.27: Suites criptográficas recomendadas para TLS 1.2 con clave precompartida (§4.1)**

Código	Grupo DH	Rec./Her.	Referencias
0x0100	ffdhe2048	H[2025]	[76]
0x0101	ffdhe3072	R	[76]
0x0102	ffdhe4096	R	[76]
0x0017	secp256r1 (P-256)	R	[148]
0x0018	secp384r1 (P-384)	R	[148]
0x001A	brainpoolP256r1	R	[136]
0x001B	brainpoolP384r1	R	[136]
0x001C	brainpoolP512r1	R	[136]

**Tabla 10.28: Grupos de Diffie-Hellman recomendados para el protocolo TLS 1.2 (§4.1)**

Código	Algoritmo de firma	Rec./Her.	Referencias
0x0001	rsa	H[2025]	[58]
0x0002	dsa	R	[58]
0x0003	ecdsa	R	[58]

**Tabla 10.29: Algoritmos de firma recomendados para el protocolo TLS 1.2 (§4.1)**

Código	Funciones resumen	Rec./Her.	Referencias
0x0004	sha256	R	[58]
0x0005	sha384	R	[58]
0x0006	sha512	R	[58]

**Tabla 10.30: Funciones resumen para el protocolo TLS 1.2 (§4.1)**

## 10.6. SSH

Protocolo	Rec./Her.	Referencias	Notas
DH-exchange-SHA256	R	[68, §4.2]	60, 61
DH-grupo15-SHA512	R	[116, §4]	61
DH-grupo16-SHA512	R	[116, §5]	61
DH-grupo17-SHA512	R	[116, §6]	61
DH-grupo18-SHA512	R	[116, §7]	61
ECDH-SHA2-*	R	[202, §6.3]	61, 62

Tabla 10.31: Versiones de acuerdos de clave para el protocolo SSH autorizadas (§4.2)

Protocolo	Rec./Her.	Referencias	Notas
AEAD_AES_128_GCM	R	[90, §6.1]	63, 64
AEAD_AES_256_GCM	R	[90, §6.2]	63, 64
AES128-CTR	R	[25, §4]	64
AES192-CTR	R	[25, §4]	64
AES256-CTR	R	[25, §4]	64

Tabla 10.32: Algoritmos de cifrado autorizados para el protocolo SSH (§4.2)

Función	Rec./Her.	Referencias	Notas
HMAC-SHA1	H[2030]	[214, §6.4]	
HMAC-SHA2-256	R	[31, §2]	
HMAC-SHA2-512	R	[31, §2]	
AEAD_AES_128_GCM	R	[90]	
AEAD_AES_256_GCM	R	[90]	

Tabla 10.33: Funciones MAC autorizadas para el protocolo SSH (§4.2)

Método	Rec./Her.	Referencias	Notas
ECDSA-SHA2-*	R	[202, §3]	65
x509v3-ECDSA-SHA2-*	R	[91, §3.4]	66

Tabla 10.34: Métodos de autenticación del servidor autorizados para el protocolo SSH (§4.2)

## 10.7. IPSEC CON IKEV2

Protocolo	Rec./Her.	Referencias	Notas
IPsec	R	[208]	68
IKEv2	R	[110], [149]	
ESP	R	[114]	68

Tabla 10.35: Versiones autorizadas del protocolo IPsec (§4.3)

Grupo (EC)DH	Rec./Her.	Referencias	Notas
3072-bit MODP	R	[116]	
4096-bit MODP	R	[116]	
6144-bit MODP	R	[116]	
8192-bit MODP	R	[116]	
256-bit aleatorio ECP	R	[70]	
384-bit aleatorio ECP	R	[70]	
521-bit aleatorio ECP	R	[70]	
brainpoolP256r1	R	[137]	
brainpoolP384r1	R	[137]	
brainpoolP512r1	R	[137]	
x25519	R	[147]	
x448	R	[147]	

Tabla 10.36: Grupos de tipo DH y ECDH acordados por IKEv2  
Grupos de tipo DH y ECDH acordados por IKEv2 (§4.3)

Esquema (H)MAC	Rec./Her.	Referencias	Notas
HMAC-SHA2-256_128	R	[112]	69
HMAC-SHA2-384_192	R	[112]	69
HMAC-SHA2-512_256	R	[112]	69
AES-CMAC-96	R	[201]	
AES-GMAC	R	[53]	

Tabla 10.37: Esquemas MAC y HMAC autorizados para IKEv2 y ESP (§4.3)

Esquema (H)MAC	Rec./Her.	Referencias	Notas
HMAC-SHA2-256	R	[112]	
HMAC-SHA2-384	R	[112]	
HMAC-SHA2-512	R	[112]	
AES128-CMAC	R	[200]	

Tabla 10.38: Funciones pseudo-aleatorias autorizadas para IKEv2 (§4.3)

Esquema de firma	Rec./Her.	Referencias	Notas
RSA (RSASSA-PSS)	R	[188]	70
ECDSA-SHA-256 y P-256	R	[69]	71
ECDSA-SHA-384 y P-384	R	[69]	71
ECDSA-SHA-512 y P-521	R	[69]	71
ECDSA-256-BrainpoolP256r1	R	[117]	71
ECDSA-384-BrainpoolP384r1	R	[117]	71
ECDSA-512-BrainpoolP512r1	R	[117]	71
ECGDSA-256-BrainpoolP256r1	R	[117]	
ECGDSA-384-BrainpoolP384r1	R	[117]	
ECGDSA-512-BrainpoolP512r1	R	[117]	

Tabla 10.39: Esquemas de firma acordados para IKEv2 (§4.3)

## 10.8. GENERADORES FÍSICOS DE NÚMEROS ALEATORIOS

Clase	Rec./Her.	Referencias	Notas
PTG.3	R	[39], [115]	72, 73 74
PTG.2	R	[39], [115]	72, 73 75

Tabla 10.40: Clases de generadores de números aleatorios autorizados (§5.2)

## 10.9. GENERADORES DE NÚMEROS ALEATORIOS DETERMINISTAS

Esquema	Rec./Her.	Referencias	Notas
HMAC-DRBG	R	[38], [97]	76, 77
Hash-DRBG	R	[38], [97]	76, 77
CTR-DRBG	R	[38], [97]	76, 77

Tabla 10.41: Generadores de números aleatorios deterministas autorizados (§5.3)

Clase	Rec./Her.	Referencias	Notas
DRG.3	R	[38], [115]	78, 79
DRG.4	R	[39], [115]	78, 79, 80

Tabla 10.42: Clases de generadores de números aleatorios deterministas autorizados (§5.3)

#### 10.10. GENERADORES DE NÚMEROS REALMENTE ALEATORIOS NO FÍSICOS

Clase	Rec./Her.	Referencias	Notas
NTG.1	R	[115]	81, 82

Tabla 10.43: Clase de generador de números aleatorios ni físico ni determinista autorizado (§5.4)

#### 10.11. GENERACIÓN DE NÚMEROS ALEATORIOS CON UNA DISTRIBUCIÓN ESPECÍFICA

Esquema	Rec./Her.	Referencias	Notas
Técnica de “prueba”	R	[162, Apend. B.1.2]	84
Técnica “extra aleatoria”	R	[162, Apend. B.1.1]	84

Tabla 10.44: Esquemas autorizados de generación de números enteros aleatorios módulo un número  $q$ , que no es una potencia de 2 (§5.5)

#### 10.12. GENERACIÓN DE CLAVES

Método	Notas
Generador de bits aleatorios autorizado	
Mecanismo de establecimiento de claves autorizado	86, 87
Función de derivación clave autorizada	87

Tabla 10.45: Métodos autorizados para la generación de claves genéricas (§6.2)

## 10.13. PROCEDIMIENTOS DE AUTENTICACIÓN

Nº de intentos	Probabilidad de falsa aceptación	Rec./Her.
5	$5 \times 10^{-6}$	R
5	$5 \times 10^{-4}$	H

Tabla 10.46: Probabilidades máximas de falsa aceptación (§7.2)

## 10.14. GENERACIÓN DE NÚMEROS PRIMOS

Esquema	Rec./Her.	Notas
Algoritmo 24 de generación de primos (Método 1)	R	93
Algoritmo 25 de generación de primos (Método 2)	R	93

Tabla 10.47: Generación de primos por muestreo de rechazo (§8.1)

$P_{obj} = 2^{-125}$	
Nº iter., $t$	Long. bits candidato, $k$
6	$950 \leq k < 1036$ (1041)
5	$1036$ (1041) $\leq k < 1289$ (1297)
4	$1289$ (1297) $\leq k < 1720$ (1729)
3	$1720$ (1729) $\leq k < 2607$ (2626)
2	$2607$ (2626) $\leq k < 5672$ (5701)
1	$5672$ (5701) $\leq k$

Tabla 10.48: Número de iteraciones del test de Miller-Rabin según la longitud en bits del candidato para  $P_{obj} = 2^{-125}$  (§8.1)

Esquema	Rec./Her.	Referencias	Notas
Test de Miller-Rabin	R	[139], [180]	94, 95, 96, 97

Tabla 10.49: Test de primalidad probabilístico autorizado (§8.1)

## 10.15. GENERACIÓN DEL PAR DE CLAVES RSA

Esquema	Rec./Her.	Notas
Algoritmo 27 de generación de claves para RSA	R	98, 99, 100

Tabla 10.50: Algoritmo autorizado para la generación claves RSA (§8.3)

## 10.16. LA AMENAZA CUÁNTICA

Criptosistema asimétrico y KEM	Primitiva matemática
CRYSTALS-Kyber	Retículo, MLWE

Tabla 10.51: Propuesta KEM seleccionada después de la tercera ronda y Primitiva matemática asociada (§9.1)

Firma digital	Primitiva matemática
CRYSTALS-Dilithium	Retículo, MLWE
Falcon	Retículo, SIS
SPHINCS <sup>+</sup>	Hash

Tabla 10.52: Propuestas a firmas seleccionadas después de la tercera ronda y Primitivas matemáticas asociadas (§9.1)

Criptosistema asimétrico y KEM	Primitiva matemática
Classic McEliece	Código Goppa
BIKE	Código de densidad moderada cuasi-cíclico
HQC	Código cuasi-cíclico de Hamming
SIKE	Isogenias sobre curvas elípticas

Tabla 10.53: Candidatos a KEM para ser analizados en la cuarta ronda y Primitivas matemáticas asociadas (§9.1)

Criptosistema asimétrico y KEM	Primitiva matemática
FrodoKEM (Round 3)	Retículo, LWE

Tabla 10.54: Propuesta KEM de interés para el CCN (§9.1)

## 11. GLOSARIO

0-RTT	Datos tiempo cero de ida y vuelta <i>zero Round-Trip Time data</i>
AE	Cifrado autenticado <i>Authenticated Encryption</i>
AEAD	Cifrado autenticado con datos asociados <i>Authenticated Encryption with Associated Data</i>
AES	Estándar de cifrado avanzado <i>Advanced Encryption Standard</i>
AH	Encabezado de autenticación <i>Authentication Header</i>
ANSI	<i>American National Standards Institute</i>
ANSSI	<i>Agence Nationale de la Sécurité des Systèmes d'Information</i>
BIKE	<i>Bit Flipping Key Encapsulation</i>
CBC	Encadenamiento de bloques de cifrado <i>Cipher-Block Chaining</i>
CBC-CS	Encadenamiento de bloques de cifrado con robo de texto cifrado <i>Cipher-Block Chaining-Ciphertext stealing</i>
CC	Criterios Comunes <i>Common Criteria</i>
CCA	Ataque al texto cifrado elegido <i>Chosen Ciphertext Attack</i>
CCM	Contador con encadenamiento de bloques de cifrado-MAC <i>Counter with Cipher Block Chaining-Message Authentication Code</i>
CCN	Centro Criptológico Nacional
CFB	Realimentación del texto cifrado <i>Cipher Feedback Mode</i>
CIA	Confidencialidad, Integridad y Autenticidad
CMAC	MAC basado en cifrado <i>Cipher-based MAC</i>
CNI	Centro Nacional de Inteligencia
CRYSTALS	<i>CRYptographic SuiTe for Algebraic LatticeS</i>
CTR	Contador <i>Counter Mode</i>
CVP	Problema del vector más cercano <i>Closest Vector Problem</i>
DEM	Mecanismo de encapsulamiento de datos <i>Data Encapsulation Mechanism</i>
DES	Cifrado de datos estándar <i>Data Encryption Standard</i>
DH	Diffie-Hellman
DHE	Diffie-Hellman con clave efímera

DHC	Diffie-Hellman con cofactor <i>Diffie-Hellman with cofactor</i>
DHAES	<i>Diffie-Hellman Augmented Encryption Scheme</i>
DHIES	<i>Diffie-Hellman Integrated Encryption Scheme</i>
DLAES	<i>Discrete Logarithm Augmented Encryption Scheme</i>
DLIES	Esquema de cifrado integrado basado en el logaritmo discreto <i>Discrete Logarithm Integrated Encryption Scheme</i>
DLP	Problema del logaritmo discreto <i>Discrete Logarithm Problem</i>
DPKE	Cifrado de clave pública determinístico <i>Deterministic Public Key Encryption</i>
DSA	Algoritmo de firma digital <i>Digital Signature Algorithm</i>
DSS	Estándar de firma digital <i>Digital Signature Standard</i>
DRNG	Generador determinista de números aleatorios <i>Deterministic Random Number Generator</i>
EAX	Cifrar-luego-autenticar-luego-traducir <i>Encrypt-then-Authenticate-then-Translate</i>
EC-DH	Diffie-Hellman basado en curvas elípticas <i>Elliptic Curve-Diffie-Hellman</i>
EC-DHE	Diffie-Hellman basado en curvas elípticas con clave efímera
EC-DLOG	Logaritmo discreto sobre curvas elípticas <i>Elliptic Curve Discrete Logarithm</i>
EC-GDSA	Algoritmo de firma digital alemán <i>Elliptic Curve German Digital Signature Algorithm</i>
EC-KCDSA	Algoritmo de firma digital coreano basado en certificados <i>Elliptic Curve Korean Certificate-based Digital Signature Algorithm</i>
EC-Schnorr	Algoritmo de firma de Schnorr para curvas elípticas <i>Elliptic Curve Based Schnorr Signature Algorithm</i>
ECDLP	Problema del logaritmo discreto sobre curvas elípticas <i>Elliptic Curve Discrete Logarithm Problem</i>
ECDSA	Algoritmo de firma digital basado en curvas elípticas <i>Elliptic Curve Digital Signature Algorithm</i>
ECIES	Esquema de cifrado integrado basado en curvas elípticas <i>Elliptic Curve Integrated Encryption Scheme</i>
ECRYPT	<i>European Network of Excellence in Cryptology</i>
ECP	Curva elípticas modulo un primo <i>Elliptic Curve modulo a Prime</i>
ESP	Carga útil de seguridad encapsulada <i>Encapsulated Security Payload</i>
ESSIV	<i>Encrypted Salt-Sector IV</i>
EUF-CMA	Existencialmente infalsificable bajo un ataque adaptativo al mensaje elegido

	<i>Existentially UnForgeable under adaptive Chosen Message Attack</i>
Falcon	<i>FAst-Fourier Lattice-based COmpact signatures over NTRU</i>
FE2OS	<i>Field Element to Octet String Conversion Primitive</i>
FF-DLOG	Logaritmo discreto multiplicativo sobre un cuerpo finito <i>Finite Field-Discrete Logarithm</i>
FTS	<i>Few-Time Signature</i>
FTPS	Protocolo seguro de transferencia de archivos <i>File Transfer Protocol Secure</i>
GCM	Contador de Galois <i>Galois Counter Mode</i>
GMAC	MAC basado en el modo contador de Galois <i>Galois Message Authentication Code</i>
HBS	Firma basada en <i>hashes</i> <i>Hash-Based Signature</i>
HMAC	MAC basado en función resumen <i>Hash function-based MAC</i>
HQC	<i>Hamming Quasi-Cyclic</i>
HTTPS	Protocolo seguro de transferencia de hipertexto <i>Hypertext Transfer Protocol Secure</i>
HSM	Módulo de seguridad hardware <i>Hardware Security Module</i>
I2SO	<i>Integer to Octet String Conversion Primitive</i>
IFP	Problema de la factorización de números enteros <i>Integer Factorization Problem</i>
IND-CCA1	Indistinguibilidad por ataque al texto cifrado elegido <i>INDistinguishability under Chosen Ciphertext Attack</i>
IND-CCA2	Indistinguibilidad por ataque adaptativo al texto cifrado elegido <i>INDistinguishability under Adaptive Chosen Ciphertext Attack</i>
IND-CPA	Indistinguibilidad por ataque al texto claro elegido <i>INDistinguishability under Chosen-Plaintext Attack</i>
IKE	Intercambio de claves de Internet <i>Internet Key Exchange</i>
IKEv2	Versión 2 del protocolo IKE
IP	Protocolo de Internet <i>Internet Protocol</i>
IPsec	Seguridad del protocolo de Internet <i>Internet Protocol Security</i>
IV	Vector de inicialización <i>Initialization Vector</i>
KA	Acuerdo de clave <i>Key Agreement</i>
KCDSA	Algoritmo de firma digital coreana basado en certificados <i>Korean Certificate-based Digital Signature Algorithm</i>
KDF	Función de derivación de claves

	<i>Key Derivation Function</i>
KEM	Mecanismo de encapsulamiento de claves <i>Key Encapsulation Mechanism</i>
KMAC	<i>Keccak Message Authentication Code</i>
KW	<i>Key Wrap</i>
KWP	<i>Key Wrap with Padding</i>
LWE	Aprendizaje con errores <i>Learning With Errors</i>
MAC	Código de autenticación de mensaje <i>Message Authentication Code</i>
MitM	Hombre en el medio <i>Man-in-the-Middle</i>
MLWE	Aprendizaje con errores sobre módulos <i>Module Learning With Errors</i>
MSS	Esquema de firma de Merkle <i>Merkle Signature Scheme</i>
MODP	Exponenciación modular <i>Modular exponentiation</i>
NIST	<i>National Institute for Standards and Technology</i>
NPTRNG	Generador de números realmente aleatorios no físicos <i>Non-Physical True Random Number Generator</i>
NTRU	<i>N-th degree Truncated polynomial Ring Units</i>
NTT	Transformada teórica de números <i>Number Theoretic Transform</i>
OAEP	Relleno de cifrado asimétrico óptimo <i>Optimal Asymmetric Encryption Padding</i>
OFB	Realimentación de la salida <i>Output Feedback</i>
OID	Identificador del objeto <i>Object Identifier</i>
OS2I	<i>Octet String to Integer Conversion</i>
OTS	Firma única <i>One Time Signature</i>
OWA	Ataque unidireccional <i>One-Way Attack</i>
PFS	Secreto perfecto persistente <i>Perfect Forward Secrecy</i>
PIN	Número de identificación personal <i>Personal Identification Number</i>
PKE	Sistema de clave pública <i>Public Key Encryption</i>
PKCS	<i>Public-Key Cryptography Standard</i>
PPKE	Cifrado de clave pública probabilístico <i>Probabilistic Public Key Encryption</i>

PQ	Postcuántico ( <i>Post-Quantum</i> )
PQC	Criptografía postcuántica <i>Post-Quantum Cryptography</i>
PRF	Función pseudo-aleatoria <i>Pseudo-Random Function</i>
PTRNG	Generador físico de números realmente aleatorios <i>Physical True Random Number Generator</i>
PSK	Clave precompartida <i>Pre-Shared Key</i>
QCSD	<i>Quasi-Cyclic Syndrome Decoding</i>
QROM	Modelo del oráculo aleatorio cuántico <i>Quantum Random Oracle Model</i>
RBG	Generador de bits aleatorios <i>Random Bit Generator</i>
RFID	Etiquetas de identificación por radio frecuencia <i>Radio Frequency Identification</i>
RLWE	Aprendizaje con errores sobre anillos <i>Ring Learning With Errors</i>
RNG	Generador de números aleatorios <i>Random Number Generator</i>
ROM	Modelo del oráculo aleatorio <i>Random Oracle Model</i>
RSA	Rivest, Shamir y Adleman
SA	Asociación de seguridad <i>Security Association</i>
SEC	<i>Standards for Efficient Cryptography</i>
SHAKE	<i>Secure Hash Algorithm and Keccak</i>
SIDH	Intercambio de clave mediante isogenias tipo Diffie-Hellman <i>Supersingular Isogeny Diffie-Hellman key exchange</i>
SIKE	<i>Supersingular Isogeny Key Encapsulation</i>
SIS	Problema de la solución entera más corta <i>Short Integer Solution</i>
SIV	Vector de inicialización sintético <i>Synthetic Initialization Vector</i>
SKE	Sistema de clave secreta ( <i>Secret Key Encryption</i> )
SOG-IS	<i>Senior Officials Group-Information Systems Security</i>
SSCDHP	Problema de Diffie-Hellman computacional supersingular <i>Supersingular Computational Diffie-Hellman Problem</i>
SSIP	Problema de isogenias supersingulares <i>Supersingular Isogeny Problem</i>
SSH	<i>Secure Shell</i>
SSL	Capa de conexión segura

SVP	<i>Secure Socket Layer</i> Problema del vector más corto <i>Shortest Vector Problem</i>
TCP/IP	<i>Transmission Control Protocol/Internet Protocol</i>
TLS	Seguridad de la capa de transporte <i>Transport Layer Security</i>
TRNG	Generador de números realmente aleatorios <i>True Random Number Generator</i>
UOV	Aceite y vinagre desequilibrado <i>Unbalanced Oil and Vinegar</i>
VPN	Red privada virtual <i>Virtual Private Network</i>
W-OTS	<i>Winternitz OTS</i>
XEX	<i>XOR Encrypt XOR</i>
XMSS	Esquema extendido de firma de Merkle <i>eXtended Merkle Signature Scheme</i>
XTS	Cifrado en bloque modificable XEX con robo de texto cifrado <i>XEX Tweakable Block Cipher with Ciphertext Stealing</i>

## 12. REFERENCIAS

- [1] M. Abdalla, M. Bellare, and P. Rogaway. *DHAES: An encryption scheme based on the Diffie-Hellman problem*. IEEE P1363a: Standard Specifications for Public-Key Cryptography, Additional Techniques, 1998. 74
- [2] M. Abdalla, M. Bellare, and P. Rogaway. DHIES: An encryption scheme based on the Diffie-Hellman problem, 2001. <http://web.cs.ucdavis.edu/~rogaway/papers/dhies.pdf>. 74
- [3] M. Abdalla, M. Bellare, and P. Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. *Lecture Notes Comput. Sci.*, 2020:143–158, 2001. [https://doi.org/10.1007/3-540-45353-9\\_12](https://doi.org/10.1007/3-540-45353-9_12). 74
- [4] L. M. Adleman, C. Pomerance, and R. S. Rumely. On distinguishing prime numbers from composite numbers. *Ann. of Math.*, 117:173–206, 1983. <https://doi.org/10.1109/SFCS.1980.28>. 120
- [5] David Adrian, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J. Alex Halderman, Nadia Heninger, Drew Springall, Emmanuel Thomé, Luke Valenta, Benjamin VanderSloot, Eric Wustrow, Santiago Zanella-Béguelink, and Paul Zimmermann. Imperfect forward secrecy: How Diffie-Hellman fails in practice. In *Proc. 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS'15)*, pages 5–17, 2015. <http://dx.doi.org/10.1145/2810103.2813707>. 84
- [6] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in P. *Ann. of Math.*, 160(2):781–793, 2004. <https://doi.org/10.4007/annals.2004.160.781>. 120
- [7] Nadhem AlFardan, Daniel J. Bernstein, Kenneth G. Paterson, Bertram Poettering, and Jacob C.N. Schuldt. On the security of RC4 in TLS and WPA. In *Proc. 22nd USENIX Security Symposium*, pages 305–320, 2013. <https://www.usenix.org/conference/usenixsecurity13/technical-sessions/paper/alFardan>. 85
- [8] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - a new hope. 2015. <http://eprint.iacr.org/2015/1092>. 136
- [9] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange-a new hope. In *25th USENIX Security Symposium*, pages 327–343, 2016. 136
- [10] ANSI. *Keyed Hash Message Authentication Code (MAC)*. American National Standards Institute, ANSI X9.71:2000, 2000. <https://standards.globalspec.com/std/412452/X9.71>. 36

- [11] ANSI. *Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*. American National Standards Institute, ANSI X9.62:2005, 2005. <https://standards.globalspec.com/std/1955141/ANSI20X9.62>. 60
- [12] ANSI. *Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography (R2017)*. American National Standards Institute, ANSI X9.63:2011, 2017. <https://webstore.ansi.org/standards/ascx9/ansix9632011r2017>. 43, 44, 55, 74, 151
- [13] ANSSI. *Avis relatif aux paramètres de courbes elliptiques définis pas l'État français*. Agence Nationale de la Sécurité des Systèmes d'Information, Journal Officiel 0241 (Oct. 2011), p. 17533, 2011. <https://www.legifrance.gouv.fr/jorf/id/JORFTEXT000024668816>. 51, 152
- [14] ANSSI. *Guide des Mécanismes Cryptographiques, Règles et Recommandations Concernant le Choix et le Dimensionnement des Mécanismes Cryptographiques*. Agence Nationale de la Sécurité des Systèmes d'Information, PG-083, 1/1/2020, 2020. [https://www.ssi.gouv.fr/uploads/2021/03/anssi-guide-mecanismes\\_crypto-2.04.pdf](https://www.ssi.gouv.fr/uploads/2021/03/anssi-guide-mecanismes_crypto-2.04.pdf). 12, 101
- [15] ANSSI. *Recommandations de sécurité relatives à TLS*. Agence Nationale de la Sécurité des Systèmes d'Information, 2020. <https://www.ssi.gouv.fr/guide/recommandations-de-securite-relatives-a-tls/>. 81
- [16] ANSSI. *Recommandations pour les Architectures des Systèmes d'Information Sensibles ou Diffusion Restreinte. Version 1.1*. Agence Nationale de la Sécurité des Systèmes d'Information, 2020. <https://www.ssi.gouv.fr/guide/recommandations-pour-les-architectures-des-systemes-d-information-sensibles-ou-diffusion-restreinte/>. 12
- [17] ANSSI. *Guide de Sélection D'Algorithmes Cryptographiques, v1.0*. Agence Nationale de la Sécurité des Systèmes d'Information, PA-079, 8/3/2021, 2021. [https://www.ssi.gouv.fr/uploads/2021/03/anssi-guide-selection\\_crypto-1.0.pdf](https://www.ssi.gouv.fr/uploads/2021/03/anssi-guide-selection_crypto-1.0.pdf). 12, 101
- [18] J.P. Aumasson, D.J. Bernstein, W. Beullens, C. Dobraunig, M. Eichlseder, S. Fluhrer, S.L. Gazdag, A. Hülsing, P. Kampanakis, S. Kölbl, T. Lange, M.M. Lauridsen, F. Mendel, R. Niederhagen, C. Rechberger, J. Rijneveld, P. Schwabe, and B. Westerbaan. SPHINCS+. Submission to the NIST post-quantum project, v.3, 2020. <https://sphincs.org/data/sphincs+-round3-submission-nist.zip>. 146

- [19] M. Badra. *Pre-Shared Key Cipher Suites for TLS with SHA-256/384 and AES Galois Counter Mode*. Internet Engineering Task Force, RFC 5487, 2009. <https://tools.ietf.org/html/rfc5487>. 91, 156
- [20] Shi Bai and Steven D. Galbraith. An improved compression technique for signatures based on learning with errors. In *Proc. Cryptographers' Track at the RSA Conference, Topic in Cryptology - CT-RSA 2014, Lecture Notes Comput. Sci.*, volume 8366, pages 28–47, 2014. [https://doi.org/10.1007/978-3-319-04852-9\\_2](https://doi.org/10.1007/978-3-319-04852-9_2). 140
- [21] Elad Barkan, Eli Biham, and Nathan Keller. Instant ciphertext-only cryptanalysis of GSM encrypted communication. In *Proc. Annual International Cryptology Conference, Advances in Cryptology - CRYPTO 2003, Lecture Notes Comput. Sci.*, volume 2729, pages 600–616, 2003. [https://doi.org/10.1007/978-3-540-45146-4\\_35](https://doi.org/10.1007/978-3-540-45146-4_35). 21
- [22] R. Barnes, M. Thomson, A. Pironti, and A. Langley. *Deprecating Secure Sockets Layer Version 3.0*. Internet Engineering Task Force, RFC 7568, 2015. <https://tools.ietf.org/html/rfc7568>. 81, 82
- [23] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *Proc. International Conference on the Theory and Application of Cryptology and Information Security, Advances in Cryptology - ASIACRYPT 2000, Lecture Notes Comput. Sci.*, volume 1976, pages 531–545, 2000. [https://doi.org/10.1007/3-540-36178-2\\_1](https://doi.org/10.1007/3-540-36178-2_1). 41, 150
- [24] M. Bellare and P. Rogaway. Minimizing the use of random oracles in authenticated encryption schemes. In *Proc. International Conference on Information and Communication Security (ICICS '97), Lecture Notes Comput. Sci.*, volume 1334, pages 1–16, 1997. <https://doi.org/10.1007/BFb0028457>. 74
- [25] M. Bellare, T. Kohno, and C. Namprempre. *The Secure Shell (SSH) Transport Layer Encryption Modes*. Internet Engineering Task Force, RFC 4344, 2006. <https://tools.ietf.org/html/rfc4344>. 95, 157
- [26] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption - How to encrypt with RSA. In *Proc. Annual International Cryptology Conference, Advances in Cryptology - EUROCRYPT 1994, Lecture Notes Comput. Sci.*, volume 950, pages 92–111, 1995. <https://doi.org/10.1007/BFb0053428F>. 53
- [27] Daniel J. Bernstein. Curve25519: New Diffie-Hellman speed records. *Lecture Notes Comput. Sci.*, 3958:207–228, 2006. [https://doi.org/10.1007/11745853\\_14](https://doi.org/10.1007/11745853_14). 51, 152

- [28] Daniel J. Bernstein. ChaCha, a variant of Salsa20. In *Proc. Workshop Record of SASC 2008: The State of the Art in Stream Ciphers*, pages 1–6, 2008. <https://cr.yp.to/papers.html#chacha>. 41, 150
- [29] Daniel J. Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O’Hearn. SPHINCS: practical stateless hash-based signatures. In *Proc. Annual International Cryptology Conference, Advances in Cryptology - EUROCRYPT 2015, Lecture Notes Comput. Sci.*, volume 9056, pages 368–397, 2015. [https://doi.org/10.1007/978-3-662-46800-5\\_15](https://doi.org/10.1007/978-3-662-46800-5_15). 70, 147
- [30] D.J. Bernstein. The Poly1305-AES message-authentication code. In *Proc. International Workshop Fast Software Encryption, FSE’2005, Lecture Notes Comput. Sci.*, 2005. 41, 150
- [31] D. Bider and M. Baushke. *SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol*. Internet Engineering Task Force, RFC 4344, 2012. <https://tools.ietf.org/html/rfc4344>. 95, 157
- [32] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In *Proc. Annual International Cryptology Conference, Advances in Cryptology - CRYPTO 1998, Lecture Notes Comput. Sci.*, volume 14162, pages 1–12, 1998. <https://doi.org/10.1007/BFb0055716>. 21, 72
- [33] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, and Damien Stehlé. Crystals-Kyber: a CCA-secure module-lattice-based KEM. *Cryptology ePrint Archive, Report 2017/634*, 2017. <https://eprint.iacr.org/2017/634>. 136
- [34] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, and Damien Stehlé. Crystals-Kyber: a CCA-secure module-lattice-based KEM. In *Proc. 2018 IEEE Symposium on Security and Privacy*, pages 353–367. IEEE Computer Society Press, 2018. <https://doi.org/10.1109/EuroSP.2018.00032>. 136
- [35] J.W. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan, and D. Stebila. Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE. In *Proc. 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS’16*, pages 1006–1018, 2016. <https://doi.org/10.1145/2976749.2978425>. 139
- [36] L. Bruckert, J. Merkle, and M. Lochter. *Elliptic Curve Cryptography (ECC) Brainpool Curves for Transport Layer Security (TLS) Version 1.3*. Internet Engineering Task Force, RFC 8734, 2020. <https://tools.ietf.org/html/rfc8734>. 87, 88, 154

- [37] BSI. *Elliptic curve cryptography*. Bundesamt für Sicherheit in der Informationstechnik, BSI TR-03111 version 2.0, 2012. [https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03111/BSI-TR-03111\\_pdf.pdf?\\_\\_blob=publicationFile](https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03111/BSI-TR-03111_pdf.pdf?__blob=publicationFile). 62, 66, 71, 152
- [38] BSI. *Functionality Classes and Evaluation Methodology for Deterministic Random Number Generators (AIS-20)*. Bundesamt für Sicherheit in der Informationstechnik, version 3, 2013. [https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS\\_20\\_pdf.pdf](https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_20_pdf.pdf). 101, 105, 159, 160
- [39] BSI. *Functionality Classes and Evaluation Methodology for Physical Random Number Generators (AIS-31)*. Bundesamt für Sicherheit in der Informationstechnik, version 3, 2013. [https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS\\_31\\_pdf.pdf](https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_31_pdf.pdf). 101, 103, 105, 159, 160
- [40] BSI. *Cryptographic Mechanisms: Recommendations and Key Lengths, Part 2 - Use of Transport Layer Security (TLS), version 2022-01*. Bundesamt für Sicherheit in der Informationstechnik, BSI TR-02102-2, 2022/01/24, 2022. [https://www.bsi.bund.de/EN/Service-Navi/Publications/TechnicalGuidelines/tr02102/tr02102\\_node.html](https://www.bsi.bund.de/EN/Service-Navi/Publications/TechnicalGuidelines/tr02102/tr02102_node.html). 81
- [41] BSI. *Cryptographic Mechanisms: Recommendations and Key Lengths, Part 4 - Use of Secure Shell (SSH), version 2022-01*. Bundesamt für Sicherheit in der Informationstechnik, BSI TR-02102-4, 2022/01/24, 2022. [https://www.bsi.bund.de/EN/Service-Navi/Publications/TechnicalGuidelines/tr02102/tr02102\\_node.html](https://www.bsi.bund.de/EN/Service-Navi/Publications/TechnicalGuidelines/tr02102/tr02102_node.html). 93
- [42] BSI. *Cryptographic Mechanisms: Recommendations and Key Lengths, version 2022-01*. Bundesamt für Sicherheit in der Informationstechnik, BSI TR-02102-1, 2022/01/28, 2022. [https://www.bsi.bund.de/EN/Service-Navi/Publications/TechnicalGuidelines/tr02102/tr02102\\_node.html](https://www.bsi.bund.de/EN/Service-Navi/Publications/TechnicalGuidelines/tr02102/tr02102_node.html). 12, 101, 102, 106, 108, 124
- [43] J. Buchmann, E. Dahmen, and M. Szydło. *Post-Quantum Cryptography*, chapter Hash-based Digital Signature Schemes, pages 35–93. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. [https://doi.org/10.1007/978-3-540-88702-7\\_3](https://doi.org/10.1007/978-3-540-88702-7_3). 70
- [44] Johannes Buchmann, Luis Carlos Coronado García, Erik Dahmen, Martin Döring, and Elena Klintsevich. CMSS - an improved Merkle signature scheme. In *Progress in Cryptology - INDOCRYPT 2006, Lecture Notes Comput. Sci.*, volume 4329, pages 349–363, 2006. [https://doi.org/10.1007/11941378\\_25](https://doi.org/10.1007/11941378_25). 70

- [45] Johannes Buchmann, Erik Dahmen, Sarah Ereth, Andreas Hülsing, and Markus Rückert. On the security of the Winternitz one-time signature scheme. In *Proc. Annual International Cryptology Conference in Africa, Progress in Cryptology - AFRICACRYPT 2011, Lecture Notes Comput. Sci.*, volume 6737, pages 363–378, 2011. [https://doi.org/10.1007/978-3-642-21969-6\\_23](https://doi.org/10.1007/978-3-642-21969-6_23). 70
- [46] Johannes Buchmann, Erik Dahmen, and Andreas Hülsing. XMSS - a practical forward secure signature scheme based on minimal security assumptions. In *Proc. Post-Quantum Cryptography (PQCrypto 2011), Lecture Notes Comput. Sci.*, volume 7071, pages 117–129, 2011. [https://doi.org/10.1007/978-3-642-25405-5\\_8](https://doi.org/10.1007/978-3-642-25405-5_8). 68, 70, 71, 152
- [47] Johannes Buchmann, Erik Dahmen, Elena Klintsevich, Katsuyuki Okeya, and Camille Vuillaume. Merkle signatures with virtually unlimited signature capacity. In *Proc. International Conference on Applied Cryptography and Network Security (ACNS 2007), Lecture Notes Comput. Sci.*, volume 4521, pages 31–45, 2007. [https://doi.org/10.1007/978-3-540-72738-5\\_3](https://doi.org/10.1007/978-3-540-72738-5_3). 70
- [48] Johannes Buchmann, Erik Dahmen, and Michael Schneider. Merkle tree traversal revisited. In *Proc. Post-Quantum Cryptography (PQCrypto 2016), Lecture Notes Comput. Sci.*, volume 5299, pages 63–78, 2008. [https://doi.org/10.1007/978-3-540-88403-3\\_5](https://doi.org/10.1007/978-3-540-88403-3_5). 70
- [49] W. Castryck and T. Decru. An efficient key recovery attack on SIDH (preliminary version). *Cryptology ePrint Archive, Report 2022/975*, 2022. <https://eprint.iacr.org/2022/975>. 131
- [50] L. Chen, T. Lange, and S. Moriai (Eds.). *WG 2 SD8 (Post-Quantum Cryptography) – Part 1: General*. International Organization for Standardization, 2020. <https://isotc.iso.org/ecom/livelihood/open/jtc1sc27wg2>. 128
- [51] H. Cohen and A. K. Lenstra. Implementation of a new primality test. *Math. Comp.*, 48(177):103–121, 1987. <https://doi.org/10.1090/S0025-5718-1987-0866102-2>. 120
- [52] H. Cohen and H. W. Lenstra, Jr. Primality testing and Jacobi sums. *Math. Comp.*, 42(165):297–330, 1984. <https://doi.org/10.1090/S0025-5718-1984-0726006-X>. 120
- [53] A. Conta, S. Deering, and M. Gupta. *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*. Internet Engineering Task Force, RFC 4543, 2006. <https://tools.ietf.org/html/rfc4543>. 99, 158

- [54] J. Daemen and V. Rijmen. AES proposal: Rijndael. pages 1–45, 1999. <https://pdfs.semanticscholar.org/4465/73a346acdbd2eb8f0527c5d73fc707f04527.pdf>. 26, 149
- [55] I. Damgård, P. Landrock, and C. Pomerance. Average case error estimates for the strong provable prime test. *Mathematics of Computation*, 61(203):177–194, 1993. <https://doi.org/10.2307/2152945>. 121, 124
- [56] T. Dierks and C. Allen. *The TLS Protocol, Version 1.0*. Internet Engineering Task Force, RFC 2246, 1999. <https://tools.ietf.org/html/rfc2246>. 82
- [57] T. Dierks and E. Rescorla. *The Transport Layer Security (TLS) Protocol, Version 1.1*. Internet Engineering Task Force, RFC 4346, 2006. <https://tools.ietf.org/html/rfc4346>. 82
- [58] T. Dierks and E. Rescorla. *The Transport Layer Security (TLS) Protocol, Version 1.2*. Internet Engineering Task Force, RFC 5246, 2008. <https://tools.ietf.org/html/rfc5246>. 81, 82, 83, 90, 92, 153, 155, 156
- [59] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, 22:644–654, 1976. <https://doi.org/10.1109/TIT.1976.1055638>. 73
- [60] C. Dods, N.P. Smart, and M. Stam. Hash based digital signature schemes. In *Proc. IMA International Conference on Cryptography and Coding, (IMACC 2005), Lecture Notes Comput. Sci.*, volume 3796, pages 96–115, 2005. [https://doi.org/10.1007/11586821\\_8](https://doi.org/10.1007/11586821_8). 70
- [61] Léo Ducas, Alain Durmus, Tancrede Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal Gaussians. In *Proc. Annual International Cryptology Conference, Advances in Cryptology - CRYPTO 2013, Lecture Notes Comput. Sci.*, volume 8042, pages 40–56, 2013. [https://doi.org/10.1007/978-3-642-40041-4\\_3](https://doi.org/10.1007/978-3-642-40041-4_3). 140
- [62] Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: A lattice-based digital signature scheme. In *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 238–268, 2018. <https://doi.org/10.13154/tches.v2018.i1.238-268>. 140
- [63] Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient identity-based encryption over NTRU lattices. In *Proc. International Conference on the Theory and Application of Cryptology and Information Security, Advances in Cryptology - ASIACRYPT 2014, Lecture Notes Comput. Sci.*, volume 8874, pages 22–41, 2014. [https://doi.org/10.1007/978-3-662-45608-8\\_2](https://doi.org/10.1007/978-3-662-45608-8_2). 143

- [64] R. Durán Díaz, V. Gayoso Martínez, and L. Hernández Encinas. Generación de primos demostrables: implementación y resultados. In *Proc. XIV Reunión Española de Criptología y Seguridad de la Información (RECSI 2016)*, pages 58–63, 2016. 119
- [65] Taher ElGamal and Kipp E.B. Hickman. *The SSL Protocol*. Internet Engineering Task Force, 1995. <https://datatracker.ietf.org/doc/html/draft-hickman-netscape-ssl-00>. 82
- [66] P.A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang. Falcon: Fast-Fourier lattice-based compact signatures over NTRU. Specification v1.2 - 01/10/2020, 2020. <https://falcon-sign.info/falcon.pdf>. 143
- [67] Alan O. Freier, Philip Karlton, and Paul C. Kocher. *The Secure Sockets Layer (SSL) Protocol, Version 3.0*. Internet Engineering Task Force, 2011. <https://datatracker.ietf.org/doc/html/draft-hickman-netscape-ssl-00>. 82
- [68] M. Friedl, N. Provos, and W. Simpon. *Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol*. Internet Engineering Task Force, RFC 4419, 2006. <https://tools.ietf.org/html/rfc4419>. 93, 157
- [69] D. Fu and J. Solinas. *IKE and IKEv2 Authentication Using the Elliptic Curve Digital Signature Algorithm (ECDSA)*. Internet Engineering Task Force, RFC 4754, 2007. <https://tools.ietf.org/html/rfc4754>. 100, 159
- [70] D. Fu and J. Solinas. *Elliptic Curve Groups modulo a Prime (ECP Groups) for IKE and IKEv2*. Internet Engineering Task Force, RFC 5903, 2010. <https://tools.ietf.org/html/rfc5903>. 97, 98, 158
- [71] Amparo Fúster Sabater, Luis Hernández Encinas, Agustín Martín Muñoz, Fausto Montoya Vitini, and Jaime Muñoz Masqué. *Criptografía, protección de datos y aplicaciones. Una guía para estudiantes y profesionales*. RA-MA, Madrid, España, 2012. 29, 34, 36, 37, 41
- [72] V. Gayoso Martínez, L. Hernández Encinas, and A. Martín Muñoz. *Criptografía con Curvas Elípticas*. Consejo Superior de Investigaciones Científicas (CSIC), Madrid, Spain, 1<sup>a</sup> edition, 2018. <http://editorial.csic.es/publicaciones/libros/13133/0/criptografia-con-curvas-elipticas.html>. 50, 55
- [73] Víctor Gayoso Martínez, Fernando Hernández Álvarez, Luis Hernández Encinas, and Carmen Sánchez Ávila. A comparison of the standardized versions of ECIES. In *Proc. Sixth International Conference on Information Assurance and Security (IAS 2010)*, volume 4, pages 1–4, 2010. <https://digital.csic.es/handle/10261/32674>. 55

- [74] Víctor Gayoso Martínez, Fernando Hernández Álvarez, Luis Hernández Encinas, and Carmen Sánchez Ávila. Analysis of ECIES and other cryptosystems based on elliptic curves. *Journal of Information Assurance and Security*, 6(4):285–293, 2011. <https://digital.csic.es/handle/10261/42125>. 55
- [75] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proc. 40<sup>th</sup> ACM Symposium on Theory of Computing (STOC'08)*, pages 316–329, 2008. <https://doi.org/10.1145/1374376.1374407>. 143
- [76] D. Gillmor. *Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS)*. Internet Engineering Task Force, RFC 7919, 2016. <https://tools.ietf.org/html/rfc7919>. 84, 87, 92, 154, 156
- [77] Shafi Goldwasser and Joe Kilian. Almost all primes can be quickly certified. In *Proc. Symposium on the Theory of Computing (STOC'86)*, pages 316–329, 1986. <https://doi.org/10.1145/12130.12162>. 120
- [78] L.K. Grover. Quantum mechanics helps in searching for a needle in a haystack. *Phys. Rev. Lett.*, 79(2):325–328, 1997. <https://doi.org/10.1103/PhysRevLett.79.325>. 15, 129
- [79] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proc. 28<sup>th</sup> annual ACM symposium on Theory of Computing (STOC'96)*, pages 212–219, 1996. <https://doi.org/10.1145/800070.802214>. 129
- [80] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In *Proc. International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2012), Lecture Notes Comput. Sci.*, volume 7428, pages 530–547, 2012. [https://doi.org/10.1007/978-3-642-33027-8\\_31](https://doi.org/10.1007/978-3-642-33027-8_31). 140
- [81] I. Hajjeh and M. Badra. *ECDHE\_PSK Cipher Suites for Transport Layer Security (TLS)*. Internet Engineering Task Force, RFC 5489, 2009. <https://tools.ietf.org/html/rfc5489>. 91, 156
- [82] D. Hankerson, A. J. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag, New York, NY, USA, 2004. <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.394.3037&rep=rep1&type=pdf>. 50
- [83] D. Harkins. *Synthetic Initialization Vector (SIV) Authenticated Encryption Using the Advanced Encryption Standard (AES)*. Internet Engineering Task Force, RFC 5297, 2008. <https://tools.ietf.org/html/rfc5297>. 43, 151

- [84] L. Hernández Encinas, J. Muñoz Masqué, and A. Queiruga Dios. Large decryption exponents in RSA. *Applied Mathematics Letters*, 16(3):292–295, 2003. [https://doi.org/10.1016/S0893-9659\(03\)80046-0](https://doi.org/10.1016/S0893-9659(03)80046-0). 126
- [85] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. NTRUSIGN: digital signatures using the NTRU lattice. In *Proc. Cryptographers' Track at the RSA Conference, Topic in Cryptology - CT-RSA 2003, Lecture Notes Comput. Sci.*, volume 2612, pages 122–140, 2003. [https://doi.org/10.1007/3-540-36563-X\\_9](https://doi.org/10.1007/3-540-36563-X_9). 143
- [86] A. Hülsing. W-OTS+ – shorter signatures for hash-based signature schemes. In *Proc. International Conference on Cryptology in Africa – AFRICACRYPT 2013, Lecture Notes Comput. Sci.*, volume 7918, pages 173–188, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. [https://doi.org/10.1007/978-3-642-38553-7\\_10](https://doi.org/10.1007/978-3-642-38553-7_10). 70
- [87] A. Hülsing, D. Butin, S.L. Gazdag, J. Rijneveld, and A. Mohaisen. XMSS: extended Merkle signature scheme. Internet Engineering Task Force, RFC 8391, 2018. <https://tools.ietf.org/html/rfc8391>. 68, 71, 152
- [88] Andreas Hülsing, Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Panos Kampanakis, Stefan Kolbl, Tanja Lange, Martin M Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, and Jean-Philippe Aumasson. SPHINCS+. Online publication, 2020. <https://sphincs.org/>. 68, 71, 145, 152
- [89] IEEE. *Standard Specifications for Public Key Cryptography-Amendment 1: Additional Techniques*. Institute of Electrical and Electronics Engineers, IEEE 1363a, 2004. <https://ieeexplore.ieee.org/document/1335427>. 55, 60, 74
- [90] K. Igoe and J. Solinas. *AES Galois Counter Mode for the Secure Shell Transport Layer Protocol*. Internet Engineering Task Force, RFC 5647, 2009. <https://tools.ietf.org/html/rfc5647>. 95, 157
- [91] K. Igoe and D. Stebila. *X.509v3 Certificates for Secure Shell Authentication*. Internet Engineering Task Force, RFC 6187, 2011. <https://tools.ietf.org/html/rfc6187>. 96, 157
- [92] ISO/IEC. *Information Technology-Security Techniques-Encryption Algorithms-Part 2: Asymmetric Ciphers*. International Organization for Standardization/International Electrotechnical Commission, 2006. <https://www.iso.org/standard/37971.html>. 74, 79, 153
- [93] ISO/IEC. *Information technology – Security techniques – Digital signature schemes giving message recovery, Part 2: Integer factorization based*

- mechanisms, ISO/IEC 9796-2*. International Organization for Standardization, 2010. <https://www.iso.org/standard/54788.html>. 71, 152
- [94] ISO/IEC. *Information Technology – Security Techniques – Encryption Algorithms – Part 3: Block Ciphers*. International Organization for Standardization/International Electrotechnical Commission, 2010. <https://www.iso.org/standard/54531.html>. 26, 149
- [95] ISO/IEC. *Information technology – Security techniques – Message Authentication Codes (MACs), Part 1: Mechanisms using a block cipher, ISO/IEC 9797-1*. International Organization for Standardization/International Electrotechnical Commission, 2011. <https://www.iso.org/standard/50375.html>. 38, 150
- [96] ISO/IEC. *Information technology – Security techniques – Message Authentication Codes (MACs), Part 2: Mechanisms using a dedicated hash-function*. International Organization for Standardization/International Electrotechnical Commission, 2011. <https://www.iso.org/standard/51618.html>. 38, 150
- [97] ISO/IEC. *Information Technology – Security Techniques – Random bit generation, ISO/IEC 18031*. International Organization for Standardization/International Electrotechnical Commission, 2011. <https://www.iso.org/standard/54945.html>. 105, 159
- [98] ISO/IEC. *Information technology – Security techniques – Key management, Part 3: Mechanisms using asymmetric techniques, ISO/IEC 11770-3*. International Organization for Standardization, 2015. <https://www.iso.org/standard/60237.html>. 79, 153
- [99] ISO/IEC. *Information technology – Security techniques – Modes of operation for an n-bit block cipher*. International Organization for Standardization/International Electrotechnical Commission, 2017. <https://www.iso.org/standard/64575.html>. 32, 149
- [100] ISO/IEC. *Information technology – Security techniques – Digital signatures with appendix, Part 3: Discrete logarithm based mechanisms, ISO/IEC 14888-3*. International Organization for Standardization, 2018. <https://www.iso.org/standard/76382.html>. 60, 71, 152
- [101] ISO/IEC. *ISO/IEC 10118-3:2018 - Information technology – Security techniques – Hash-functions – Part 3: Dedicated hash-functions. ISO/IEC10118-3:2018*. International Organization for Standardization/International Electrotechnical Commission, 2018. <https://www.iso.org/standard/67116.html>. 27, 149

- [102] ISO/IEC. *Information technology – Authenticated encryption*. International Organization for Standardization/International Electrotechnical Commission, 2020. <https://www.iso.org/standard/81550.html>. 41, 150
- [103] ISO/IEC. *Information Technology – Security Techniques – Prime number generation, ISO/IEC 18032*. International Organization for Standardization/International Electrotechnical Commission, 2020. <https://www.iso.org/standard/72009.html>. 124
- [104] H. Jiang, Z. Zhang, L. Chen, H. Wang, and Z. Ma. IND-CCA-secure key encapsulation mechanism in the quantum random oracle model, revisited. *Cryptology ePrint Archive, Report 2017-1096*, 2017. <http://eprint.iacr.org/2017/1096>. 139
- [105] K. Moriarty and B. Kaliski J. Jonsson and A. Rusch. *Public-Key Cryptography Standard (PKCS) #1: RSA Cryptography Specifications Version 2.2.*, RFC 8017, 2016. <https://tools.ietf.org/html/rfc8017>. 71, 100, 152
- [106] B. Kaliski. *PKCS #5: Password-Based Cryptography Specification, Version 2.0*. Internet Engineering Task Force, RFC 2898, 2000. <https://datatracker.ietf.org/doc/html/rfc2898>. 44
- [107] B. Kaliski and J. Staddon. *PKCS #1: RSA Cryptography Specifications, Version 2.0*. Internet Engineering Task Force, RFC 2437, 1998. <https://tools.ietf.org/html/rfc2437>. 53
- [108] Marc Kaplan, Gaëtan Leurent, Anthony Leverrier, and María Naya-Plasencia. Breaking symmetric cryptosystems using quantum period finding. In *Proc. Annual International Cryptology Conference, Advances in Cryptology – CRYPTO 2016, Lecture Notes Comput. Sci.*, volume 9815, pages 207–237, 2016. [https://doi.org/10.1007/978-3-662-53008-5\\_](https://doi.org/10.1007/978-3-662-53008-5_). 15
- [109] Marc Kaplan, Gaëtan Leurent, Anthony Leverrier, and María Naya-Plasencia. Quantum differential and linear cryptanalysis. *IACR Transactions on Symmetric Cryptology*, 2:71–94, 2016. <https://doi.org/10.13154/tosc.v2016.i1.71-94>. 15
- [110] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen, and T. Kivinen. *Internet Key Exchange Protocol Version 2 (IKEv2)*. Internet Engineering Task Force, RFC 7296, 2014. <https://tools.ietf.org/html/rfc7296>. 96, 97, 158
- [111] KCDSA Task Force Team. *The Korean Certificate-based Digital Signature Algorithm*. International Organization for Standardization, 1998. <https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.99.8398>. 63
- [112] S. Kelly and S. Frankel. *Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec*. Internet Engineering Task Force, RFC 4868, 2007. <https://tools.ietf.org/html/rfc4868>. 99, 158, 159

- [113] J. Kelsey, S. Chang, and R. Perlner. *SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash, and ParallelHash*. National Institute of Standard and Technology, Special Publication, SP800-185, Rev 1 (November 2014 draft), 2016. <https://doi.org/10.6028/NIST.SP.800-185>. 37, 38, 150
- [114] S. Kent. *IP Encapsulating Security Payload (ESP)*. Internet Engineering Task Force, RFC 4303, 2005. <https://tools.ietf.org/html/rfc4303>. 97, 158
- [115] W. Killmann and W. Schindler. *Functionality Classes and Evaluation Methodology for Physical Random Number Generators*. Bundesamt für Sicherheit in der Informationstechnik, version 2, 2011. [https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS\\_31\\_Functionality\\_classes\\_for\\_random\\_number\\_generators\\_e.pdf](https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_31_Functionality_classes_for_random_number_generators_e.pdf). 101, 103, 104, 105, 107, 159, 160
- [116] T. Kivinen and M. Kojo. *More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)*. Network Working Group, 2003. <https://datatracker.ietf.org/doc/html/rfc3526>. 49, 93, 97, 98, 151, 157, 158
- [117] T. Kivinen and J. Snyder. *Signature Authentication in the Internet Key Exchange Version 2 (IKEv2)*. Internet Engineering Task Force, RFC 7427, 2015. <https://tools.ietf.org/html/rfc7427>. 100, 159
- [118] H. Krawczyk, M. Bellare, and R. Canetti. *HMAC: Keyed-Hashing for Message Authentication*. Internet Engineering Task Force, RFC 2104, 1997. <https://tools.ietf.org/html/rfc2104>. 38, 150
- [119] H. Krawczyk and P. Eronen. *HMAC-based Extract-and-Expand Key Derivation Function (HKDF)*. Internet Engineering Task Force, RFC 5869, 2010. <https://tools.ietf.org/html/rfc5869>. 44, 151
- [120] L. Lamport. *Constructing digital signatures from a one way function*. Technical Report Technical Report SRI-CSL-98, SRI International, Computer Science Laboratory, 1979. <https://www.microsoft.com/en-us/research/publication/constructing-digital-signatures-one-way-function/>. 68, 69
- [121] A. Langley, W. Chang, N. Mavrogiannopoulos, J. Strombergson, and S. Josefsson. *ChaCha20-Poly1305 Cipher Suites for Transport Layer Security (TLS)*. Internet Engineering Task Force, RFC 7905, 2016. <https://tools.ietf.org/html/rfc7905>. 85, 86, 89, 90, 91, 153, 155, 156
- [122] A. Langley, M. Hamburg, and S. Turner. *Elliptic curves for security*. Internet Engineering Task Force, RFC 7748, 2016. <https://tools.ietf.org/html/rfc7748>. 51, 152

- [123] D.H. Lehmer. An extended theory of Lucas' functions. *Ann. of Math.*, 31(3):419–448, 1930. <https://doi.org/10.2307/1968235>. 120
- [124] Chae Hoon Lim and Pil Joong Lee. The Korean certificate-based digital signature algorithm. *Computers and Electrical Engineering*, 25:249–265, 1999. [https://doi.org/10.1016/S0045-7906\(99\)00011-7](https://doi.org/10.1016/S0045-7906(99)00011-7). 63
- [125] M. Locheter and J. Merkle. *Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve generation*. Internet Engineering Task Force, RFC 5639, 2010. <https://tools.ietf.org/html/rfc5639>. 51, 152
- [126] Vadim Lyubashevsky. Lattice signatures without trapdoors. In *Proc. Annual International Conference on the Theory and Applications of Cryptographic Techniques, Advances in Cryptology - EUROCRYPT 2012, Lecture Notes Comput. Sci.*, volume 7237, pages 738–755, 2012. [https://doi.org/10.1007/978-3-642-29011-4\\_43](https://doi.org/10.1007/978-3-642-29011-4_43). 140
- [127] Vadim Lyubashevsky, Leo Ducas, Eike Kiltz, Tancrede Lepoint, Peter Schwabe, Gregor Seiler, and Damien Stehle. CRYSTALS-DILITHIUM. Online publication, 2020. <https://pq-crystals.org/>. 67, 71, 141, 152
- [128] V. Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. *Lecture Notes Comput. Sci.*, 5912:598–616, 2009. [https://doi.org/10.1007/978-3-642-10366-7\\_35](https://doi.org/10.1007/978-3-642-10366-7_35). 140
- [129] V. Gayoso Martínez, L. Hernández Encinas, and C. Sánchez Ávila. A survey of the elliptic curve integrated encryption scheme. *Journal of Computer Science and Engineering*, 2(2):7–13, 2010. <https://digital.csic.es/handle/10261/32671>. 55
- [130] M. Matsui. Linear cryptanalysis method for DES cipher. *Lecture Notes Comput. Sci.*, 765:386–397, 1994. [https://doi.org/10.1007/3-540-48285-7\\_33](https://doi.org/10.1007/3-540-48285-7_33). 21
- [131] J.P. Mattsson and D. Migault. *ECDHE\_PSK with AES-GCM and AES-CCM Cipher Suites for TLS 1.2 and DTLS 1.2*. Internet Engineering Task Force, RFC 8442, 2018. <https://tools.ietf.org/html/rfc8442>. 91, 156
- [132] U.M. Maurer. Fast generation of prime numbers and secure publickey cryptographic parameters. *Journal of Cryptology*, 8(3):123–155, 1995. <https://doi.org/10.1007/BF00202269>. 119
- [133] D. McGrew and D. Bailey. *AES-CCM Cipher Suites for Transport Layer Security TLS*. Internet Engineering Task Force, RFC 6655, 2008. <https://tools.ietf.org/html/rfc6655>. 90, 91, 155, 156

- [134] D. McGrew, D. Bailey, M. Campagna, and R. Dugal. *AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites for TLS*. Internet Engineering Task Force, RFC 7251, 2014. <https://tools.ietf.org/html/rfc7251>. 89, 155
- [135] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1996. <http://cacr.uwaterloo.ca/hac/>. 26, 27, 121, 149
- [136] J. Merkle and M. Lochter. *Elliptic Curve Cryptography (ECC) Brainpool Curves for Transport Layer Security (TLS)*. Internet Engineering Task Force, RFC 7027, 2013. <https://tools.ietf.org/html/rfc7027>. 92, 156
- [137] J. Merkle and M. Lochter. *Using the Elliptic Curve Cryptography (ECC) Brainpool Curves for the Internet Key Exchange Protocol Version 2 (IKEv2)*. Internet Engineering Task Force, RFC 6954, 2013. <https://tools.ietf.org/html/rfc6954>. 98, 158
- [138] Ralph Charles Merkle. A certified digital signature. In *Proc. Annual International Cryptology Conference, Advances in Cryptology - CRYPTO 1989, Lecture Notes Comput. Sci.*, volume 435, pages 218–238, 1989. [https://doi.org/10.1007/0-387-34805-0\\_21](https://doi.org/10.1007/0-387-34805-0_21). 68, 70
- [139] G.L. Miller. Riemann's hypothesis and test for primality. *J. Comput. & System Sci.*, 13(3):300–317, 1976. [https://doi.org/10.1016/S0022-0000\(76\)80043-8](https://doi.org/10.1016/S0022-0000(76)80043-8). 121, 123, 161
- [140] Ministerio de Defensa. *Real Decreto 421/2004, de 12 de marzo, por el que se regula el Centro Criptológico Nacional*. BOE núm. 68, de 19 de marzo de 2004, páginas 12203 a 12204, 2004. <https://www.boe.es/eli/es/rd/2004/03/12/421>. 12
- [141] K. Moriarty, B. Kaliski, J. Jonsson, and A. Rusch. *Public-Key Cryptography Standard (PKCS) #1: RSA Cryptography Specifications, Version 2.2*. Internet Engineering Task Force, RFC 8017, 2016. <https://datatracker.ietf.org/doc/html/rfc8017>. 54, 152
- [142] K. Moriarty, B. Kaliski, and A. Rusch. *PKCS #5: Password-Based Cryptography Specification. Version 2.1*. Internet Engineering Task Force, RFC 8018, 2017. <https://tools.ietf.org/html/rfc8018>. 44, 151
- [143] Michael Naehrig, Erdem Alkim, Joppe Bos, Leo Ducas, Karen Easterbrook, Brian LaMacchia, Patrick Longa, Ilya Mironov, Valeria Nikolaenko, Christopher Peikert, Ananth Raghunathan, and Douglas Stebila. FrodoKEM. Online publication, 2020. <https://frodokem.org/>. 78, 79, 139, 153
- [144] Enric Nart. Counting hyperelliptic curves. *Advances in Mathematics*, 221(3):774–787, 2009. <https://doi.org/10.1016/j.aim.2009.01.001>. 52

- [145] M. Nemeč. The properties of RSA key generation process in software libraries. Technical report, Master Thesis, Faculty of Informatics, Masaryk University, Brno, 2016. <http://www.fi.muni.cz/reports>. 126
- [146] Matus Nemeč, Marek Sys, Petr Švenda, Dusan Klinec, and Vashek Matyas. The return of Coppersmith's attack: Practical factorization of widely used RSA moduli. In *Proc. 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS'17)*, pages 496–499, 2017. <https://doi.org/10.1145/3133956.3133969>. 119, 126
- [147] Y. Nir and S. Josefsson. *Curve25519 and Curve448 for the Internet Key Exchange Protocol Version 2 (IKEv2) Key Agreement*. Internet Engineering Task Force, RFC 8031, 2016. <https://tools.ietf.org/html/rfc8031>. 97, 98, 158
- [148] Y. Nir, S. Josefsson, and M. Pégourié-Gonnard. *Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier*. Internet Engineering Task Force, RFC 8422, 2013. <https://tools.ietf.org/html/rfc8422>. 87, 92, 154, 156
- [149] Y. Nir, T. Kivien, P. Wouters, and D. Migault. *Algorithm Implementation Requirements and Usage Guidance for the Internet Key Exchange Protocol Version 2 (IKEv2)*. Internet Engineering Task Force, RFC 8247, 2017. <https://tools.ietf.org/html/rfc8247>. 45, 97, 151, 158
- [150] Y. Nir and A. Langley. *ChaCha20 and Poly1305 for IETF Protocols*. Internet Engineering Task Force, RFC 8439, 2018. <https://tools.ietf.org/html/rfc8439>. 41, 85, 150
- [151] NIST. *Recommended Elliptic Curves for Federal Government Use*. National Institute of Standards and Technology, 1999. 94, 96
- [152] NIST. *Advanced Encryption Standard*. National Institute of Standard and Technology, Federal Information Processing Standard Publication, FIPS 197, 2001. <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>. 26, 149
- [153] NIST. *Recommendation for Block Cipher Modes of Operation. Methods and Techniques*. National Institute of Standards and Technology, Special Publication SP 800-38A, March 2001. <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf>. 32, 149
- [154] NIST. *Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*. National Institute of Standards and Technology, Special Publication SP 800–38B, May 2005. <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38b.pdf>. 36, 38, 150

- [155] NIST. *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*. National Institute of Standards and Technology, Special Publication SP 800–38D, November 2007. <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf>. 38, 39, 41, 150
- [156] NIST. *Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality*. National Institute of Standards and Technology, Special Publication SP 800–38C, May 2007. <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf>. 41, 150
- [157] NIST. *The Keyed-Hash Message Authentication Code (HMAC)*. National Institute of Standard and Technology, NIST FIPS 198-1, 2008. <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.198-1.pdf>. 36
- [158] NIST. *Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices*. National Institute of Standards and Technology, Special Publication SP 800–38E, November 2010. <https://csrc.nist.gov/publications/detail/sp/800-38e/final>. 35, 150
- [159] NIST. *Recommendation for Password-Based Key Derivation. Part 1: Storage Applications*. National Institute of Standard and Technology, Special Publication, Addendum to SP800-132, 2010. <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf>. 43, 44, 151
- [160] NIST. *Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping*. National Institute of Standards and Technology, Special Publication SP 800–38F, December 2012. <https://csrc.nist.gov/publications/detail/sp/800-38f/final>. 43, 151
- [161] NIST. *Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode*. National Institute of Standard and Technology, Special Publication, Addendum to SP800-38A, 2012. <https://doi.org/10.6028/NIST.SP.800-38A-Add>. 32, 149
- [162] NIST. *Digital Signature Standard (DSS)*. National Institute of Standard and Technology, NIST FIPS 186-4, 2013. <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf>. 51, 59, 60, 71, 109, 121, 124, 129, 152, 160
- [163] NIST. *Secure Hash Standard (SHS)*. National Institute of Standard and Technology, NIST FIPS 180-4, 2015. <https://csrc.nist.gov/publications/detail/fips/180/4/final>. 27, 38, 149, 150

- [164] NIST. *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. National Institute of Standard and Technology, NIST FIPS 202, 2015. [http://csrc.nist.gov/publications/drafts/fips-202/fips\\_202\\_draft.pdf](http://csrc.nist.gov/publications/drafts/fips-202/fips_202_draft.pdf). 27, 149
- [165] NIST. Post-quantum cryptography. On-line publication, 2017. <https://csrc.nist.gov/Projects/Post-Quantum-Cryptography>. 52, 129
- [166] NIST. *Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography*. National Institute of Standard and Technology, Special Publication, SP800-56A, Rev. 3, 2018. <https://doi.org/10.6028/NIST.SP.800-56Ar3>. 44, 79, 129, 151, 153
- [167] NIST. *Recommendation for Pair-Wise Key-Establishment Using Integer Factorization Cryptography*. National Institute of Standard and Technology, Special Publication, SP800-56B, Rev. 2, 2019. <https://doi.org/10.6028/NIST.SP.800-56Br2>. 44, 129, 151
- [168] NIST. PQC standardization process: Third round candidate announcement. Online publication, 2020. <https://csrc.nist.gov/News/2020/pqc-third-round-candidate-announcement>. 78, 80
- [169] NIST. *Recommendation for Key-Derivation Methods in Key-Establishment Schemes*. National Institute of Standard and Technology, Special Publication, SP800-56C, Rev. 2, 2020. <https://doi.org/10.6028/NIST.SP.800-56Cr2>. 44, 151
- [170] NIST. *Recommendation for Stateful Hash-Based Signature Schemes*. National Institute of Standard and Technology, Special Publication, SP800-208, 2020. <https://doi.org/10.6028/NIST.SP.800-208>. 68, 71, 152
- [171] NIST. Post-quantum cryptography. selected algorithms 2022. On-line publication, 2022. <https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022>. 68, 72, 77, 80, 130, 143, 146
- [172] NIST. *Recommendation for Key Derivation Using Pseudorandom Functions*. National Institute of Standard and Technology, Special Publication, Addendum to SP800-108r1, 2022. <https://doi.org/10.6028/NIST.SP.800-108r1>. 44, 151
- [173] K. Nohl and S. Krißler. Subverting the security base of GSM. Technical report, 2009. [https://web.archive.org/web/20110726142029/https://har2009.org/program/attachments/119\\_GSM.A51.Cracking.Nohl.pdf](https://web.archive.org/web/20110726142029/https://har2009.org/program/attachments/119_GSM.A51.Cracking.Nohl.pdf). 21

- [174] C. Percival. Stronger key derivation via sequential memory-hard functions, 2009. <http://www.tarsnap.com/scrypt/scrypt.pdf>. 44
- [175] C. Percival and S. Josefsson. *The SCRYPT Password-Based Key Derivation Function*. Internet Engineering Task Force, RFC 7914, 2016. <https://tools.ietf.org/html/rfc7914>. 44, 151
- [176] H.C. Pocklington. The determination of the prime or composite nature of large numbers by Fermat's theorem. *Math. Proc. Cambridge Philos. Soc.*, 18:29–30, 1914. 120
- [177] A. Popov. *Prohibiting RC4 Cipher Suites*. Internet Engineering Task Force, RFC 7465, 2015. <https://tools.ietf.org/html/rfc7465>. 85
- [178] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Online publication, 2020. <https://falcon-sign.info/>. 68, 71, 143, 152
- [179] L. Hernández Encinas y J. Muñoz Masqué R. Durán Díaz. *El criptosistema RSA*. RA-MA, Madrid, España, 2005. [https://www.ra-ma.es/libro/el-criptosistemarsa\\_48854/](https://www.ra-ma.es/libro/el-criptosistemarsa_48854/). 120, 121
- [180] M.O. Rabin. Probabilistic algorithms for testing primality. *J. Number Theory*, 12(1):128–138, 1980. [https://doi.org/10.1016/0022-314X\(80\)90084-0](https://doi.org/10.1016/0022-314X(80)90084-0). 121, 123, 161
- [181] O. Regev. New lattice-based cryptographic constructions. *Journal of the ACM*, 51(6):899–942, 2004. <https://doi.org/10.1145/1039488.1039490>. 135
- [182] E. Rescorla. *TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)*. Internet Engineering Task Force, RFC 5289, 2008. <https://tools.ietf.org/html/rfc5289>. 89, 90, 155
- [183] E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. Internet Engineering Task Force, RFC 8446, 2018. <https://tools.ietf.org/html/rfc8446>. 82, 83, 86, 88, 153, 154
- [184] Ronald Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978. <https://doi.org/10.1145/359340.359342>. 21, 48, 151
- [185] RSA Labs. *PKCS #1 v2.2: RSA Cryptography Standard*. RSA Laboratories, 2012. [https://mpqs.free.fr/h11300-pkcs-1v2-2-rsa-cryptography-standard-wp-EMC\\_](https://mpqs.free.fr/h11300-pkcs-1v2-2-rsa-cryptography-standard-wp-EMC_)

- [Corporation\\_Public-Key\\_Cryptography\\_Standards\\_\(PKCS\).pdf](#). 53, 54, 71, 152
- [186] J.A. Salowey, D. McGrew, and A. Choudhury. *AES Galois Counter Mode (GCM) Cipher Suites for TLS*. Internet Engineering Task Force, RFC 5288, 2008. <https://tools.ietf.org/html/rfc5288>. 90, 155
- [187] Greeshma Sarath, Devesh Jinwala, and Sankita Patel. A survey on elliptic curve digital signature algorithm and its variants. In *Proc. Second International Conference on Computational Science and Engineering, Computer Science & Information Technology*, volume 4, pages 121–136, 2014. <https://doi.org/10.5121/csit.2014.4411>. 63
- [188] J. Schaad, B. Kaliski, and R. Housley. *Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. Internet Engineering Task Force, RFC 4055, 2005. <https://tools.ietf.org/html/rfc4055>. 100, 159
- [189] Claus Peter Schnorr. Efficient signature generation by smartcards. *Journal of Cryptology*, (4):161–174, 1991. <https://doi.org/10.1007/BF00196725>. 61
- [190] Berry Schoenmakers. *Lecture Notes, Cryptographic Protocols. Version 1.6*. Department of Mathematics and Computer Science, Technical University of Eindhoven, 2021. <https://www.win.tue.nl/~berry/CryptographicProtocols/LectureNotes.pdf>. 16
- [191] Peter Schwabe, Roberto Avanzi, Joppe Bos, Leo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehle. CRYSTALS-KYBER. Online publication, 2020. <https://pq-crystals.org/>. 77, 79, 136, 153
- [192] SEC2. *Recommended Elliptic Curve Domain Parameters*. Standards for Efficient Cryptography Group, 2000. 94, 96
- [193] Standards For Efficient Cryptography (SECG). *SEC 1: Elliptic Curve Cryptography*. Standards for Efficient Cryptography Group, SECG SEC1, 2000. <https://www.secg.org/SEC1-Ver-1.0.pdf>. 55
- [194] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979. <https://doi.org/10.1145/359168.359176>. 28, 149
- [195] P.W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997. <https://doi.org/10.1137/S0097539795293172>. 15, 52, 128

- [196] D.R. Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26(5):1474–1483, 1997. <https://doi.org/10.1137/S0097539796298637>. 129
- [197] SOG-IS. *SOG-IS Crypto Evaluation Scheme. Agreed Cryptographic Mechanisms. Version 1.2*. Senior Officials Group-Information Systems Security, Crypto Working Group, 2020. [https://www.sogis.eu/uk/supporting\\_doc\\_en.html](https://www.sogis.eu/uk/supporting_doc_en.html). 12, 16, 101, 122
- [198] R. Solovay and V. Strassen. A fast Monte-Carlo test for primality. *SIAM J. Computing*, 6:84–85, 1977. <https://doi.org/10.1137/0206006>. 121
- [199] R. Solovay and V. Strassen. Erratum for “A fast Monte-Carlo test for primality”. *SIAM J. Computing*, 7:118, 1978. <https://doi.org/10.1137/0207009>. 121
- [200] J. Song, R. Poovendran, J. Lee, and T. Iwata. *The Advanced Encryption Standard-Cipher-based Message Authentication Code-Pseudo-Random Function-128 (AES-CMAC-PRF-128) Algorithm for the Internet Key Exchange Protocol (IKE)*. Internet Engineering Task Force, RFC 4615, 2006. <https://tools.ietf.org/html/rfc4615>. 99, 159
- [201] J.H. Song and J. Lee. *The AES-CMAC-96 Algorithm and Its Use with IPsec*. Internet Engineering Task Force, RFC 4494, 2006. <https://tools.ietf.org/html/rfc4494>. 99, 158
- [202] D. Stebila and J. Green. *Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer*. Internet Engineering Task Force, RFC 5656, 2009. <https://tools.ietf.org/html/rfc5656>. 93, 96, 157
- [203] Damien Stehlé and Ron Steinfeld. Making NTRU as secure as worst-case problems over ideal lattices. In *Proc. Annual International Conference on the Theory and Applications of Cryptographic Techniques, Advances in Cryptology - EUROCRYPT 2011, Lecture Notes Comput. Sci.*, volume 6632, pages 27–47, 2011. [https://doi.org/10.1007/978-3-642-20465-4\\_4](https://doi.org/10.1007/978-3-642-20465-4_4). 143
- [204] S. Turner and T. Polk. *Prohibiting Secure Sockets Layer (SSL) Version 2.0*. Internet Engineering Task Force, RFC 6176, 2011. <https://tools.ietf.org/html/rfc6176>. 82
- [205] Mathy Vanhoef and Frank Piessens. All your biases belong to us: Breaking RC4 in WPA-TKIP and TLS. In *Proc. 24th USENIX Security Symposium*, pages 97–112, 2015. <https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/vanhoef>. 85

- [206] P. Švenda, M. Nemeč, P. Sekan, R. Kvašňovský, D. Formánek, D. Komárek, and V. Matyáš. The million-key question – Investigating the origins of RSA public keys. Technical report, Faculty of Informatics, Masaryk University, Brno, 2016. <http://www.fi.muni.cz/reports>. 126
- [207] P. Švenda, M. Nemeč, P. Sekan, R. Kvašňovský, D. Formánek, D. Komárek, and V. Matyáš. The million-key question—Investigating the origins of RSA public keys. In *Proc. 25<sup>th</sup> USENIX Security Symposium (USENIX 16)*, pages 893–910, Austin, TX, 2016. USENIX Association. <https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/svenda>. 117, 126
- [208] P. Wouters, D. Migault, J. Marrsson, Y. Nir, and T. Kivien. *Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)*. Internet Engineering Task Force, RFC 8221, 2017. <https://tools.ietf.org/html/rfc8221>. 96, 97, 158
- [209] P. Wouters, H. Tschofenig, J. Gilmore, and T. Kivinen. *Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)*. Internet Engineering Task Force, RFC 7250, 2014. <https://tools.ietf.org/html/rfc7250>. 84
- [210] T. Ylönen. SSH - secure login connections over the internet. In *Proc. 6<sup>th</sup> USENIX Security Symposium (USENIX 96)*, pages 37–42, 1996. [https://www.usenix.org/legacy/publications/library/proceedings/sec96/full\\_papers/yloinen/index.html](https://www.usenix.org/legacy/publications/library/proceedings/sec96/full_papers/yloinen/index.html). 93
- [211] T. Ylonen and C. Lonvick. *The Secure Shell (SSH) Authentication Protocol*. Internet Engineering Task Force, RFC 4252, 2006. <https://tools.ietf.org/html/rfc4252>. 93
- [212] T. Ylonen and C. Lonvick. *The Secure Shell (SSH) Connection Protocol*. Internet Engineering Task Force, RFC 4254, 2006. <https://tools.ietf.org/html/rfc4254>. 93
- [213] T. Ylonen and C. Lonvick. *The Secure Shell (SSH) Protocol Architecture*. Internet Engineering Task Force, RFC 4251, 2006. <https://tools.ietf.org/html/rfc4251>. 92, 93
- [214] T. Ylonen and C. Lonvick. *The Secure Shell (SSH) Transport Layer Protocol*. Internet Engineering Task Force, RFC 4253, 2006. <https://tools.ietf.org/html/rfc4253>. 93, 94, 95, 157

