GUÍA/NORMA DE SEGURIDAD DE LAS TIC (CCN-STIC-807)

CRIPTOLOGÍA DE EMPLEO EN EL ESQUEMA NACIONAL DE SEGURIDAD

SEPTIEMBRE 2011

Edita:

NIPO: 075-11-053-3

Tirada: 1000 ejemplares
Fecha de Edición: septiembre de 2011

Security Wisdom ha participado en la elaboración y modificación del presente documento y sus anexos.
El Ministerio de Política Territorial y Administración Pública ha financiado el desarrollo del presente documento y sus anexos.

PRÓLOGO

El uso masivo de las tecnologías de la información y las telecomunicaciones (TIC), en todos los ámbitos de la sociedad, ha creado un nuevo espacio, el ciberespacio, donde se producirán conflictos y agresiones, y donde existen ciberamenazas que atentarán contra la seguridad nacional, el estado de derecho, la prosperidad económica, el estado de bienestar y el normal funcionamiento de la sociedad y de las administraciones públicas.

La Ley 11/2002, de 6 de mayo, reguladora del Centro Nacional de Inteligencia, encomienda al Centro Nacional de Inteligencia el ejercicio de las funciones relativas a la seguridad de las tecnologías de la información en su artículo 4.e), y de protección de la información clasificada en su artículo 4.f), a la vez que confiere a su Secretario de Estado Director la responsabilidad de dirigir el Centro Criptológico Nacional en su artículo 9.2.f).

Partiendo del conocimiento y la experiencia del CNI sobre amenazas y vulnerabilidades en materia de riesgos emergentes, el Centro realiza, a través de su 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, a la aplicación de políticas y procedimientos de seguridad, y al empleo de tecnologías de seguridad adecuadas.

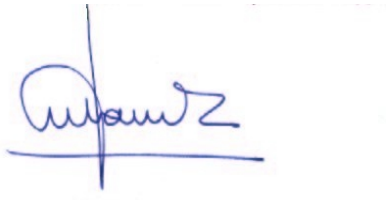
Una de las funciones más destacables del Centro Criptológico Nacional es la de elaborar y difundir normas, instrucciones, guías y recomendaciones para garantizar la seguridad de los sistemas de las tecnologías de la información y las comunicaciones de la Administración, materializada en la existencia de la serie de documentos CCN-STIC.

Disponer de un marco de referencia que establezca las condiciones necesarias de confianza en el uso de los medios electrónicos es, además, uno de los principios que establece la ley 11/2007, de 22 de junio, de acceso electrónico de los ciudadanos a los servicios públicos, en su artículo 42.2 sobre el Esquema Nacional de Seguridad (ENS).

Precisamente el Real Decreto 3/2010 de 8 de Enero de desarrollo del Esquema Nacional de Seguridad fija los principios básicos y requisitos mínimos así como las medidas de protección a implantar en los sistemas de la Administración, y promueve la elaboración y difusión de guías de seguridad de las tecnologías de la información y las comunicaciones por parte de CCN para facilitar un mejor cumplimiento de dichos requisitos mínimos.

En definitiva, la serie de documentos CCN-STIC se elabora para dar cumplimiento a los cometidos del Centro Criptológico Nacional y a lo reflejado en el Esquema Nacional de Seguridad, 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 su difícil, y en ocasiones, ingrata tarea de proporcionar seguridad a los sistemas de las TIC bajo su responsabilidad.

Enero de 2011



Félix Sanz Roldán
Secretario de Estado
Director del Centro Criptológico Nacional

ÍNDICE

ANEXOS

1. INTRODUCCIÓN

1. El desarrollo de la sociedad actual repercute en la administración del estado al tener que actualizar ésta los sistemas electrónicos con el fin de proporcionar a los ciudadanos los mismos servicios electrónicos que ofrece de forma local. Tal actualización supone una remodelación la seguridad de los sistemas para su adaptación a los nuevos requisitos y posibles amenazas.
2. El Centro Criptológico Nacional (CCN) lleva años trabajando en la securización, evaluación y acreditación de los sistemas de información de la administración del estado, además de la formación del personal de dicha administración.
3. Dentro de la estructura del CCN se encuentra el Organismo de Certificación (OC) del Centro Criptológico Nacional que comprende a las entidades públicas o privadas que deseen ejercer como laboratorios de evaluación de la seguridad de las Tecnologías de la Información (TI) en el marco del Esquema Nacional de Evaluación y Certificación de la Seguridad de las Tecnologías de la Información (ENECSTI). También incluye a las entidades públicas o privadas que sean fabricantes de productos o sistemas de TI que pretendan certificar la seguridad de sus productos en el marco del ENECSTI (véase http://www.oc.ccn.cni.es/index_es.html para mayor información). Los certificados "Common Criteria" emitidos por el Organismo de Certificación están reconocidos internacionalmente por más de veinte países. Además, el OC está acreditado por la Entidad Nacional de Acreditación, conforme a los criterios recogidos en la Norma UNE-EN 45011:1998 para la certificación de productos.
4. Por esta razón los sistemas, productos y equipos evaluados y certificados por el CCN cumplen los requisitos de funcionalidad que tales productos afirman verificar en la declaración de seguridad.

1.1. ORGANISMOS DE ESTANDARIZACIÓN

5. Existen diferentes organismos internacionales que se encargan de establecer como «estándares» determinados sistemas, productos y equipos, entre los que también se encuentran algoritmos y protocolos criptográficos. Estos organismos son los que determinan la calidad y fiabilidad de los diferentes sistemas y productos para su uso comercial. La mayor parte de los organismos son entidades independientes (aunque otras pertenezcan a organismos gubernamentales).
6. Los organismos de estandarización internacionales más importantes son los siguientes:
7. ANSI (American National Standards Institute): el Instituto Nacional Americano de Estándares (http://www.ansi.org/) es una organización norteamericana que supervisa el desarrollo de estándares para productos, servicios, procesos y sistemas. Es miembro de ISO y de IEC. También se encarga de la coordinación entre los estándares norteamericanos e internacionales.
8. IEC (International Electrotechnical Commission): la Comisión Electrotécnica Internacional (http://www.iec.ch/) es un organismo de estandarización en los campos eléctrico, electrónico y de otras tecnologías relacionadas con ellos. Muchas normas se desarrollan conjuntamente con ISO, por lo que muchas de ellas se conocen como normas ISO/IEC.

9. IEEE (Institute of Electrical and Electronics Engineers): el Instituto de Ingenieros Eléctricos y Electrónicos (http://www.ieee.org/index.html) es una asociación mundial de carácter técnico-profesional dedicada a la estandarización de tecnologías derivadas de la electricidad: ingeniería computacional, tecnología biomédica y aeroespacial, energía eléctrica, telecomunicaciones, etc.

10. ISO (International Organization for Standardization): la Organización Internacional para la Estandarización (http://www.iso.org/iso/home.html) es el organismo encargado de desarrollar normas internacionales de fabricación, comercio y comunicación en todas las ramas de la industria (salvo las relativas a la industria eléctrica y electrónica), especialmente en los temas relacionados con las normas de los productos y la seguridad.

11. NIST (National Institute of Standards and Technology): el Instituto Nacional de Estándares y Tecnología norteamericano (http://www.nist.gov/index.html) es una agencia del Departamento de Comercio de los Estados Unidos. Se encarga de promocionar la innovación y la competencia industrial en Estados Unidos mediante avances en las normas aplicadas y en la propia tecnología. Sus principales áreas de actuación son biotecnología, nanotecnología y tecnologías de la información.

12. SECG (Standards for Efficient Cryptography Group): el Grupo de Estándares para la Criptografía Eficiente (http://www.secg.org/) es un consorcio internacional cuyo principal objetivo es promover el uso de la criptografía basada en curvas elípticas. Entre sus miembros destacan Certicom, Entrust, Fujitsu y Visa.

2. OBJETIVO

13. En la presente guía se presentan los algoritmos criptográficos que han sido acreditados para el uso únicamente en el Esquema Nacional de Seguridad, cuando sus características y requerimientos se consideren necesarios.

3. ALGORITMOS ACREDITADOS

14. La siguiente relación de algoritmos y protocolos criptográficos se consideran acreditados por el CCN para su uso dentro del Esquema Nacional de Seguridad (ENS), siempre que se realice una implementación correcta de los mismos según las especificaciones adjuntas:

3.1. CIFRADO SIMÉTRICO

15. TDEA (Triple Data Encryption Algorithm, Triple Algoritmo de Cifrado de Datos): SP 800-20, SP800-38B y SP 800-67 del NIST ([NIST, SP800-20], [NIST, SP800-38B], [NIST, SP800-67]).

16. AES (Advanced Encryption Standard, Cifrado de Datos Avanzado): FIPS 197 y SP800-38B del NIST ([NIST, FIPS197], [NIST, SP800-38B]) y la Suite B de la NSA ([NSA, SuiteB]).

3.2. PROTOCOLOS DE ACUERDO DE CLAVE

17. DH o DHKA (Diffie-Hellman Key Agreement, Acuerdo de Clave de Diffie-Hellman): ANSI X9.42 ([ANSI, X9.42]) y PKCS #3 de los laboratorios RSA ([RSALab, 1993]).
18. MQV (Menezes-Qu-Vanstone Key Agreement, Acuerdo de Clave de Menezes-Qu-Vanstone): ANSI X9.42 ([ANSI, X9.42]), ANSI X9.63 ([ANSI, X9.63]) e IEEE 1363 [IEEE, 1363].
19. ECDH (Elliptic Curve Diffie-Hellman, Acuerdo de Clave de Diffie-Hellman con Curvas Elípticas): ANSI X9.63 ([ANSI, X9.63]), IEEE1363 ([IEEE, 1363]), IEEE1363a ([IEEE, 1363a]) y la Suite B de la NSA ([NSA, SuiteB]).
20. ECMQV (Elliptic Curve Menezes-Qu-Vanstone, Acuerdo de Clave de Menezes-Qu-Vanstone con Curvas Elípticas): Suite B de la NSA ([NSA, SuiteB]) y SEC 1 del SECG ([SECG, SEC1]).

3.3. ALGORITMOS ASIMÉTRICOS

21. DSA (Digital Signature Algorithm, Algoritmo de Firma Digital): ANSI X9.30 ([ANSI, X9.30-1]), FIPS 186-2 ([NIST, FIPS186-2]) y FIPS 186-3 ([NIST, FIPS186-3]).
22. ECDSA (Elliptic Curve Digital Signature Algorithm, Algoritmo de Firma Digital con Curvas Elípticas): ANSI X9.62 ([ANSI, X9.62]), FIPS 186-2 ([NIST, FIPS186-2]), SP 800-57A del NIST ([NIST, SP800-57A]), la Suite B de la NSA ([NSA, SuiteB]) y SEC 1 del SECG ([SECG, SEC1]).
23. RSA (Criptosistema RSA): ANSI X9.44 ([ANSI, X9.44]), FIPS 186-2 ([NIST, FIPS186-2]) y PKCS #1 de los laboratorios RSA ([RSALab, 2002]).
24. ECIES (Elliptic Curve Integrated Encryption Scheme, Esquema de Cifrado Integrado con Curvas Elípticas): ANSI X9.63 ([ANSI, X9.63]), IEEE1363a ([IEEE, 1363a]) e ISO 18033-2 ([ISOIEC, 18033-2]).

3.4. FUNCIONES RESUMEN

25. SHA (Secure Hash Algorithm, Algoritmo Resumen Seguro): FIPS180-1 ([NIST, FIPS180-1]), la Suite B de la NSA ([NSA, SuiteB]) y FIPS180-2 ([NIST, FIPS180-2]).
26. HMAC (Hash Message Authentication Code, Código de Autenticación de Mensaje con Resumen): ANSI X9.71 ([ANSI, X9.71]) y FIPS 198 ([NIST, FIPS198]).

4. PRODUCTOS CERTIFICADOS

27. Se entiende por productos certificados todos aquellos que hayan sido evaluados conforme a normas europeas o internacionales y que estén certificados por entidades independientes de reconocida solvencia.
28. Tendrán la consideración de normas europeas o internacionales, ISO/IEC 15408, u otras de naturaleza y calidad análogas.
29. Tendrán la consideración de entidades independientes de reconocida solvencia las recogidas en los acuerdos o arreglos internacionales de reconocimiento mutuo de los certificados de la seguridad.

30. Dado que la lista de los productos certificados es muy extensa y podría quedar obsoleta por la certificación de nuevos productos, se recomienda consultar las listas actualizadas de productos certificados de los organismos de acreditación y certificación anteriores. En particular, son de especial interés la página web del Organismo de Certificación del Centro Criptológico Nacional (http://www.oc.ccn.cni.es/ProdCert_es.html) y el portal de Common Criteria (http://www.commoncriteriaportal.org/products/).

31. En cualquier caso, debe tenerse en cuenta que si la implementación de determinado algoritmo no ha sido conveniente y correctamente verificada, no es posible asegurar que tal algoritmo sea seguro. Debilidades en la generación de números aleatorios, espacios de claves reducidas, además se conocen ataques a determinadas implementaciones prácticas de algoritmos mediante los llamados ataques por canal lateral (en inglés, side-channel attacks) que explotan la falta de medidas y contramedidas de seguridad en tales implementaciones, como por ejemplo, fallos de programación, defectos en la configuración, errores de arquitectura, etc.

5. MEDIDAS DE SEGURIDAD

32. A continuación se listan cada una de las medidas de seguridad especificadas en el ENS que utilizan algoritmos criptológicos acreditados, determinando la fortaleza criptológica mínima necesaria para cada dimensión.

5.1. MECANISMOS DE IDENTIFICACIÓN

33. Se entiende por identificación la comprobación de la identidad de una entidad, ya sea ésta una persona, un terminal, un proceso, una tarjeta, etc.

34. Con el fin de que toda entidad sea convenientemente identificada, cada una de ellas tendrá asignado un identificador único de modo que en cada momento se pueda tener conocimiento de quién ha hecho qué cosa. Por ejemplo, pueden servir de identificadores únicos, el número del DNI, el número del pasaporte, una secuencia de caracteres numéricos o alfanuméricos, un certificado digital, etc.

35. Si se emplea un certificado digital, éste tendrá que utilizar en su firma digital una función resumen cuya seguridad sea mayor o igual a la función SHA-1 (ver la sección 3 del ANEXO C) o, preferiblemente, cualquiera de la serie SHA-2 (ver la sección 5 del ANEXO C).

36. En el caso de que los caracteres numéricos o alfanuméricos a emplear sean obtenidos de forma aleatoria, éstos deberán ser generados con la suficiente seguridad como para evitar repeticiones o hipótesis acerca de su posible valor (ver el ANEXO E).

37. Los requerimientos para los tres niveles de seguridad de los mecanismos de identificación son los mismos.

5.1.1. NIVEL BAJO

38. Se aplica la sección 5.1.3.

5.1.2. NIVEL MEDIO

39. Se aplica la sección 5.1.3.

5.1.3. NIVEL ALTO

40. Los mecanismos de identificación serán todos aquellos que garanticen de forma fehaciente la identidad de la entidad de que se trate. Así, para la identificación de personas se pueden considerar como tales el DNI, el pasaporte, el carnet de conducir, tarjetas de identidad emitidas por el organismo que controle el acceso, etc. También se considerarán mecanismos válidos las tarjetas numeradas de visitantes, siempre que la persona poseedora de la misma haya pasado previamente por un control de acceso donde haya sido identificada mediante alguno de los mecanismos ya citados.

41. Para la verificación de los derechos que posee la entidad, deberá existir una base de datos segura que permita validar en todo momento dichos derechos.

5.2. MECANISMOS DE AUTENTICACIÓN

42. Un mecanismo de autenticación (o autentificación) es un proceso por el que una parte se asegura, mediante la obtención de una evidencia, de la identidad de una segunda parte que está implicada en un protocolo y en la que dicha segunda parte ha participado, es decir, la segunda parte participa activamente en el momento preciso o justo antes de que se adquiera la evidencia (ver sección 8 del ANEXO C).

43. En general, los mecanismos de autenticación se basan en el uso de tres posibles propiedades o características de la parte que ha de ser autenticada: algo que sabe (por ejemplo, una clave), algo que se tiene (una tarjeta inteligente o un dispositivo físico son ejemplos de esta característica) y algo que se es (un rasgo o propiedad biométrica, fisonomía facial, la huella digital, el patrón de iris, etc.).

44. Para los mecanismos de autenticación se consideran tres niveles de seguridad: bajo, medio y alto.

5.2.1. NIVEL BAJO

45. Para el nivel bajo se admite cualquier mecanismo de autenticación, ya sean claves concertadas (se entiende por clave concertada una ristra de bits que no hay forma de memorizar), contraseñas (se entiende por contraseña números de identificación personal –PIN– de al menos 4 dígitos o caracteres), dispositivos físicos (en inglés, tokens, dongles), dispositivos software como certificados digitales y procesos basados en biometría.

46. Con el fin de evitar posibles ataques por fuerza bruta, se recomienda la utilización de políticas de bloqueo de contraseñas, de modo que después de determinado número de intentos fallidos (se recomiendan hasta tres intentos) el acceso mediante contraseña quede bloqueado o métodos de retardo de solicitud de contraseña.

5.2.2. NIVEL MEDIO

47. Los mecanismos de autenticación admitidos para el nivel medio son, también, dispositivos físicos, dispositivos software como certificados digitales y procesos basados en biometría.

48. No está aconsejado el uso de claves concertadas, en el caso de utilización se permitirán aquellas que estén formadas de al menos 8 caracteres alfanuméricos. Si estos últimos son generados de forma aleatoria o pseudoaleatoria, deberán poseer la suficiente seguridad como para evitar repeticiones o hipótesis acerca de su posible valor (ver el ANEXO E).

49. De nuevo, para evitar posibles ataques por fuerza bruta, se recomienda la utilización de políticas de bloqueo de contraseñas, de modo que después de determinado número de intentos fallidos (se recomiendan hasta tres intentos) el acceso mediante contraseña quede bloqueado o métodos de retardo de solicitud de contraseña.

5.2.3. NIVEL ALTO

50. Para el nivel alto, los mecanismos de autenticación se basarán en dispositivos físicos personalizados o mediante dispositivos que hagan uso de patrones biométricos. En el caso de utilizar un patrón biométrico se requerirá la utilización de un segundo factor de autenticación token o clave segura (generada de forma aleatoria y con una longitud de al menos 8 caracteres alfanuméricos).

51. No se permite, para este nivel, el uso de claves concertadas.

52. A modo de ejemplo, además de los sistemas de autenticación mencionados en la sección 8 del ANEXO C, también son aceptables los siguientes sistemas de autenticación: RADIUS (en inglés, Remote Authentication Dial-In User Server), TLS (en inglés, Transport Layer Security), EAP (en inglés, Extensible Authentication Protocol), WPA (en inglés, Wi-Fi Protected Access).

53. A fin de evitar posibles ataques por fuerza bruta, se recomienda la utilización de políticas de bloqueo de contraseñas, de modo que después de determinado número de intentos fallidos (se recomiendan hasta tres intentos) el acceso mediante contraseña quede bloqueado o métodos de retardo de solicitud de contraseña.

5.3. PROTECCIÓN DE LA CONFIDENCIALIDAD

54. La confidencialidad de una información consiste en mantener dicha información secreta para todos salvo para los autorizados a conocerla.

5.3.1. NIVEL BAJO

55. No se aplica.

5.3.2. NIVEL MEDIO

56. Para mantener la confidencialidad de la información en el nivel medio se utilizarán redes privadas virtuales, en particular se hará uso de IPsec (en inglés, Internet Protocol security), SSL (en inglés, Secure Sockets Layer) y TLS.

57. IPsec es un protocolo, que posibilita proteger las comunicaciones sobre una red IP, de modo que cada uno de los paquetes de datos que se transmite es cifrado y autenticado. Dado que IPsec incluye protocolos para el establecimiento de claves de cifrado, éstas deberán garantizar, para este nivel medio, una seguridad equivalente a 112 bits (ver el ANEXO A).

58. Por su parte, SSL y su sucesor TLS, son protocolos criptográficos que proporcionan autenticidad y privacidad en una red, es decir, comunicaciones seguras, haciendo uso de métodos criptográficos. En general, en estos protocolos únicamente se autentica el servidor, quedando el cliente sin autenticar. Al igual que con IPsec, las claves que se utilicen en estos protocolos deberán tener un nivel de seguridad de 112 bits.

59. Un nivel de seguridad de 112 bits se traduce en claves de 112 bits (o superiores) para los sistemas de cifrado simétrico TDEA y AES (ver secciones 7 y 8 del ANEXO A), en claves de longitud 2048 bits (o superiores) para el criptosistema RSA (ver sección 3 del ANEXO B) y en claves de longitudes comprendidas entre los 224 y 255 bits para criptosistemas basados en curvas elípticas (ver sección 6 del ANEXO B).

5.3.3. NIVEL ALTO

60. En el nivel alto de protección de la información se emplearán, preferentemente, dispositivos hardware para el establecimiento y uso de la red privada virtual.

61. Además, en este caso las claves a utilizar tienen que garantizar un nivel de seguridad de 128 bits.

62. Un nivel de seguridad de 128 bits supone el uso de claves de 128 bits (o superiores) para el sistema de cifrado simétrico AES (ver secciones 7 y 8 del ANEXO A) y de claves de entre 256 y 283 bits para los sistemas basados en curvas elípticas (ver sección 6 del ANEXO B).

63. Para el caso particular del criptosistema RSA (ver sección 3 del ANEXO B), se permiten claves de 2048 bits, si bien se recomienda que la longitud de las claves sea mayor.

5.4. PROTECCIÓN DE LA AUTENTICIDAD Y DE LA INTEGRIDAD

64. Se entiende por autenticidad de una información la corroboración de la fuente de la información, es decir, la verificación de que quien la elaboró o quien dice ser el remitente de la misma es quien dice ser. Por su parte, la integridad de una información hace referencia a la comprobación de que la información recibida no ha sido alterada por entidades no autorizadas o por medios no conocidos.

5.4.1. NIVEL BAJO

65. En el nivel bajo, se asegurará, al menos, la autenticidad del otro extremo de la comunicación antes de proceder al intercambio de información (ver sección 8 del ANEXO C), de modo que se prevengan posibles ataques activos, que serán, como mínimo, detectados.

66. Se consideran ataques activos (por contraposición a los ataques pasivos en los que sólo se monitoriza la comunicación con el fin de obtener información de la misma o de lo intercambiado en ella) a aquellos ataques en los que se altere la información transmitida, se inserte información engañosa, o se secuestre la comunicación.

5.4.2. NIVEL MEDIO

67. En el nivel medio se emplearán, al igual que en la protección de la confidencialidad (ver sección 5.3.2), redes privadas virtuales que proporcionen una seguridad equivalente a 112 bits.

68. Esto es, se utilizarán claves cuya longitud sea de 112 bits (o superiores) para los sistemas de cifrado simétrico TDEA y AES (ver secciones 7 y 8 del ANEXO A), claves de 2048 bits (o superiores) para el criptosistema RSA (ver sección 3 del ANEXO B) y claves de

longitudes comprendidas entre los 224 y 255 bits para criptosistemas basados en curvas elípticas (ver sección 6 del ANEXO B).

5.4.3. NIVEL ALTO

69. De manera análoga a como se menciona para el nivel alto en la protección de la confidencialidad (ver sección 5.3.3), en este nivel alto también se emplearán redes privadas virtuales que garanticen una seguridad equivalente a 128 bit, con la salvedad ya señalada del criptosistema RSA. Además, se recomienda el uso de dispositivos hardware para el establecimiento y uso de las redes privadas virtuales.

70. Es decir, se emplearán claves de 128 bits (o mayores) para los criptosistemas de cifrado simétrico TDEA y AES (ver secciones 7 y 8 del ANEXO A) y claves de entre 256 y 283 bits para los criptosistemas basados en curvas elípticas (ver sección 6 del ANEXO B). Para el criptosistema RSA (ver sección 3 del ANEXO B), se permiten claves de 2048 bits, si bien se recomienda que la longitud de las claves sea mayor.

5.5. CIFRADO DE LA INFORMACIÓN

71. Cifrar una información consiste en transformarla de modo que pase a ser ilegible para todos salvo para las entidades autorizadas a acceder a dicha información. En general, el acceso a la información original a partir de su versión cifrada se lleva a cabo mediante el uso de claves y algoritmos.

72. El cifrado será aplicable a la información cuya confidencialidad sea considerada de nivel alto. Así, no se hará distinción acerca del soporte en el que la misma esté almacenada. En este sentido, se incluyen tanto los dispositivos fijos (disco duro, etc.), como los dispositivos removibles, es decir, CDs, DVDs, discos USB, etc.,

5.5.1. NIVEL BAJO

73. No se aplica.

5.5.2. NIVEL MEDIO

74. No se aplica.

5.5.3. NIVEL ALTO

75. Para el cifrado de información considerada de nivel alto en confidencialidad, ya sea en tránsito o mientras esté almacenada, se utilizarán sistemas de cifrado seguros.

76. Se entienden por sistemas de cifrado seguros los que garantizan una seguridad de, al menos, 128 bits. Es decir, se emplearán claves para el cifrado/descifrado de información de 128 bits (o mayores) para los criptosistemas de cifrado simétrico TDEA y AES (ver secciones 7 y 8 del ANEXO A) y claves de entre 256 y 283 bits para los criptosistemas basados en curvas elípticas (ver sección 6 del ANEXO B). Para el caso particular del criptosistema RSA (ver sección 3 del ANEXO B), se permiten claves de 2048 bits, si bien se recomienda que dicha longitud sea mayor en la medida de lo posible.

5.6. PROTECCIÓN DE CLAVES CRIPTOGRÁFICAS

77. Las claves criptográficas, independientemente de la seguridad que ofrezcan, estarán protegidas durante todo su ciclo de vida. Ello significa que se deberán arbitrar las medidas de seguridad necesarias tanto en el proceso de generación de las claves, como en su transporte al punto de explotación, en su custodia durante el tiempo que estén en uso, y en su posterior almacenamiento después de su vida activa hasta su destrucción final.

78. En la medida de lo posible, se velará porque los dispositivos portátiles no almacenen claves de acceso remoto a los diferentes organismos. Se entienden por claves de acceso remoto aquellas que permiten acceder a los equipos del organismo del que se depende o de otros organismos de naturaleza similar.

79. Se recomienda que, en caso de que sea necesario almacenar claves en ordenadores portátiles u otros dispositivos removibles, éstas estén a su vez cifradas por otras claves que sólo el propietario del hardware que las tiene almacenadas sea capaz de generar y utilizar.

5.6.1. NIVEL BAJO

80. Para proteger las claves en el nivel bajo, los procesos de generación de las mismas deberán estar aislados y no conectados a ninguna red. De igual modo, las claves archivadas por haber sido retiradas y en espera de ser destruidas, también deberán estar almacenadas en dispositivos aislados.

81. Además, el acceso a los procesos tanto de generación de claves, como de transporte, custodia y almacenamiento estarán protegidos con una seguridad equivalente a 112 bits.

5.6.2. NIVEL MEDIO

82. Para el nivel medio, se considerarán las mismas protecciones que las señaladas en la sección 5.6.1, si bien, en este caso las claves proporcionarán una seguridad equivalente a 128 bits.

5.6.3. NIVEL ALTO

83. Se aplica la sección 5.6.2.

5.7. FIRMA ELECTRÓNICA

84. Una firma electrónica o digital es la versión análoga a una firma manuscrita pero en formato electrónico y tiene como fin enlazar de forma robusta una información con una entidad, esto es, el firmante de la información. De este modo, la firma electrónica previene el hecho de que un firmante pueda repudiar ser el autor de la información firmada, además de que la misma permite garantizar la integridad del contenido firmado.

5.7.1. NIVEL BAJO

85. Se podrá utilizar cualquier medio de firma electrónica de los reconocidos por la legislación vigente. En este sentido, los protocolos de firma electrónica harán uso de

certificados digitales reconocidos (por ejemplo, los emitidos por la Fábrica Nacional de Moneda y Timbre) con claves RSA (del firmante) de, al menos, 1024 bits (ver sección 3 del ANEXO B), o claves de 224-255 bits si se emplean curvas elípticas (ver sección 6 del ANEXO B).Se aplica la sección 5.7.2.

86. No se admitirá el uso de certificados cuya función resumen (en inglés, hash) sea la función MD5 u otra de seguridad inferior (ver ANEXO C). Esto es, la función resumen deberá tener una seguridad mínima equiparable a la de la función SHA-1 o la RIPEMD-160 (ver secciones 3 y 4 del ANEXO C).

5.7.2. NIVEL MEDIO

87. Se podrá utilizar cualquier medio de firma electrónica de los reconocidos por la legislación vigente. En este sentido, los protocolos de firma electrónica harán uso de certificados digitales reconocidos (por ejemplo, los emitidos por la Fábrica Nacional de Moneda y Timbre) con claves RSA (del firmante) de, al menos, 1024 bits (ver sección 3 del ANEXO B), o claves de 224-255 bits si se emplean curvas elípticas (ver sección 6 del ANEXO B).

88. No se admitirá el uso de certificados cuya función resumen (en inglés, hash) sea la función MD5 u otra de seguridad inferior (ver ANEXO C). Esto es, la función resumen deberá tener una seguridad mínima equiparable a la de la función SHA-1 o la RIPEMD-160 (ver secciones 3 y 4 del ANEXO C).

89. En cualquier caso, se recomienda el uso de certificados digitales reconocidos (por ejemplo, los emitidos por la Fábrica Nacional de Moneda y Timbre o por la Dirección General de la Policía y Guardia Civil) cuya clave pública RSA sea de 2048 bits (o superior), o si se emplean curvas elípticas, las claves deberían tener una clave cuya longitud esté comprendida entre 255 y 283 bits. Se recomienda que la función resumen a utilizar sea, preferiblemente, cualquiera de la serie SHA-2 (ver la sección 5 del ANEXO C).

90. Además, las firmas se protegerán con sellos de tiempo (ver sección 5.8).

5.7.3. NIVEL ALTO

91. Al igual que en el caso del nivel medio, se podrá utilizar cualquier medio de firma electrónica de los reconocidos por la legislación vigente siempre que su clave RSA sea de, al menos, 2048 bits (ver sección 3 del ANEXO B) y utilice como función resumen la función SHA-1, la RIPEMD-160 u otra de seguridad equivalente, aunque se recomienda el uso de cualquier función resumen de las incluidas en la serie SHA-2 (ver la sección 5 del ANEXO C).

92. Se recomienda el uso de certificados digitales reconocidos cuya clave pública RSA sea mayor de 2048 bits y cuya función resumen sea SHA-1, RIPEMD-160 o, preferiblemente, cualquiera de la serie SHA-2 (ver la sección 5 del ANEXO C).

93. Si se emplean certificados digitales reconocidos basados en criptosistemas de clave pública que empleen curvas elípticas, deberán tener una clave cuya longitud esté entre los 256 y los 283 bits y deberán hacer uso de la función SHA-1, RIPEMD-160 o, preferiblemente, de cualquiera de las funciones incluidas en la serie SHA-2.

94. También en este nivel, las firmas se protegerán con sellos de tiempo (ver sección 5.8).

5.8. SELLOS DE TIEMPO

95. Se entiende por sellado de tiempo al almacenamiento o grabación del momento temporal en que se creó o se tiene constancia de una información, de modo que será imposible repudiar dicha información posteriormente al momento en que se selló.

5.8.1. NIVEL BAJO

96. No se aplica.

5.8.2. NIVEL MEDIO

97. No se aplica.

5.8.3. NIVEL ALTO

98. Se hará uso de sistemas de sellado de tiempo que utilicen una Autoridad de Sellado de Tiempo (en inglés, Time Stamping Authority o TSA) (sección 6 del ANEXO D) siendo de los denominados esquemas simples o esquemas enlazados.

99. En los esquemas simples la TSA recibe el documento a sellar, le añade el tiempo actual y lleva a cabo un proceso de firma electrónica mediante un criptosistema asimétrico.

100. La firma electrónica de la TSA para el sellado de tiempo se hará mediante una clave RSA de, al menos, 3072 bits (ver sección 3 del ANEXO B), o curvas elípticas con claves de, al menos, 284 bits, y una función resumen de las incluidas en la serie SHA-2 con una seguridad mayor o igual que la SHA-256 (ver la sección 5 del ANEXO C). Es decir, no se aceptarán sellados de tiempo con firmas electrónicas realizadas por la TSA con longitudes de claves RSA menores a 3072 bits, o con claves de menos de 284 bits para ECC, ni con funciones resumen cuya seguridad sea menor que la ofrecida por la función SHA-256 de la serie SHA-2.

101. Para los esquemas enlazados, la seguridad recae en la función resumen empleada, por tanto, se empleará cualquiera de las funciones de la serie SHA-2 con una seguridad mayor o igual que la SHA-256 (ver la sección 5 del ANEXO C).

ANEXO A. CRIPTOGRAFÍA DE CLAVE SIMÉTRICA: CIFRADORES EN FLUJO

1. SISTEMAS CRIPTOGRÁFICOS: CONCEPTOS BÁSICOS

102. Etimológicamente la palabra Criptología proviene del griego (criptos = oculto y logos = ciencia, tratado) y denota de forma genérica dos disciplinas opuestas pero a su vez complementarias: la Criptografía y el Criptoanálisis.

103. La Criptografía diseña procedimientos para cifrar, es decir, para ocultar o enmascarar una determinada información confidencial.

104. El Criptoanálisis se ocupa de atacar dichos procedimientos de cifrado para así recuperar la información original.

105. La Criptografía se puede considerar la parte «constructiva» de este proceso mientras que el Criptoanálisis es claramente la parte «destructiva». Sin embargo, ambas disciplinas siempre se han desarrollado de forma conjunta puesto que cualquier procedimiento de cifrado da lugar a un criptoanálisis o, al menos, a un intento de criptoanálisis.

106. En la Figura A.1 se ilustra el esquema general de un proceso criptográfico (cifrado/descifrado) que puede desglosarse de la siguiente manera:

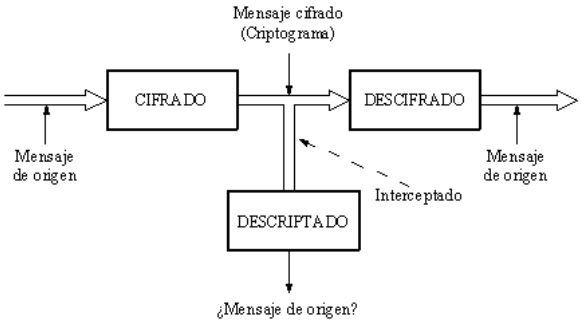


Figura A.1. Esquema general de un proceso criptográfico (cifrado/descifrado).

107. Sean A y B el emisor y receptor, respectivamente, de un determinado mensaje, que en términos criptográficos se denomina texto claro, texto fuente o mensaje original.

108. En emisión, A transforma el mensaje original, mediante un proceso de cifrado controlado por una clave, en un mensaje cifrado llamado criptograma. El criptograma se envía al receptor por un canal abierto sin ningún tipo de protección puesto que ya está cifrado.

109. En recepción, B (con conocimiento de la clave) aplica la transformación inversa sobre el criptograma reconvirtiéndolo en el texto fuente y recuperando así la información original.

110. En el proceso de transmisión, un enemigo o atacante criptoanalista puede interceptar ese criptograma y llevar a cabo una labor de descriptado. Es decir, intenta a partir del criptograma y sin conocimiento de la clave recuperar el mensaje original.

111. Un buen sistema criptográfico es pues aquel que ofrece un descifrado sencillo para el legítimo receptor pero un descriptado imposible para el enemigo criptoanalista.

112. Según sean las claves utilizadas en el proceso de cifrado/descifrado, existe una primera clasificación de métodos criptográficos:

113. MÉTODOS CRIPTOGRÁFICOS DE CLAVE SIMÉTRICA: son aquellos en los que la clave de cifrado coincide con la clave de descifrado. Esta clave única tiene que permanecer secreta y ser conocida exclusivamente por A y B. Este hecho presupone que emisor y receptor se han puesto previamente de acuerdo en la determinación de la misma o que existe un centro de distribución de claves, que por un canal seguro, la hace llegar a ambos comunicantes.

114. Estos métodos tienen pues que solventar el difícil problema de la distribución de claves.

115. MÉTODOS CRIPTOGRÁFICOS DE CLAVE ASIMÉTRICA: son aquellos en los que la clave de cifrado es diferente a la de descifrado. En general la clave de cifrado es conocida libremente por el público, mientras que la de descifrado es conocida únicamente por el usuario sin tener que compartirla con nadie.

116. Estos métodos claramente evitan el problema de la distribución de claves.

117. La Criptografía simétrica también se denomina Criptografía clásica o Criptografía de clave secreta, mientras que los métodos asimétricos se corresponden con la llamada Criptografía de clave pública, introducida por Diffie y Hellman en 1976 ([Diffie and Hellman, 1976]).

118. En los anexos subsiguientes se ahondará en cada uno de estos métodos criptográficos.

119. Conceptualmente hablando, los procedimientos de clave asimétrica resultan mucho más satisfactorios puesto que evitan el complejo entramado de la gestión de claves. Sin embargo, a la hora de implementarlos, son procedimientos muy lentos especialmente cuando se comparan con los de clave simétrica. Esto los hace menos eficientes cuando hay que proteger grandes cantidades de información.

120. En la práctica se tiende a una solución ecléctica: se utiliza la Criptografía de clave asimétrica para la distribución de claves, puesto que éstas tiene una longitud corta. Una vez que emisor y receptor disponen de una misma clave secreta, entonces ya se utiliza un método criptográfico simétrico.

121. La Criptografía de clave simétrica se divide a su vez en dos grupos fundamentales:

122. MÉTODOS DE CIFRADO EN FLUJO: son aquellos en los que la transformación de cifrado se aplica sobre cada símbolo del alfabeto en el que está escrito el mensaje original. Actualmente toda la información está codificada en un alfabeto binario, luego la transformación de cifrado se aplica sobre cada bit del texto claro.

123. MÉTODOS DE CIFRADO EN BLOQUE: son aquellos en los que la transformación de cifrado se aplica sobre un grupo (bloque) de símbolos del alfabeto en el que está escrito el mensaje original dando lugar a un bloque de texto cifrado. En la práctica estos bloques pueden ser de 64, 128 ó 256 bits del mensaje original dependiendo del método de cifrado utilizado.

124. Tradicionalmente la Criptografía se ha asociado con el concepto de confidencialidad, es decir, que la información transmitida sea incomprensible para todos con excepción de los dos comunicantes A y B.

125. Hoy en día la confidencialidad es tan sólo un elemento más entre otros muchos aspectos a considerar en cuestiones de seguridad tales como:

- Autenticidad tanto del criptograma (*integridad*) como del par emisor/receptor (que se trate realmente del legítimo emisor/receptor).
- No-repudio o negación de la recepción de un mensaje por parte del receptor.
- Control de acceso a unos determinados dispositivos o servicios.
- Protocolos criptográficos para establecer una comunicación, etc.

126. Todos ellos conforman ese concepto más amplio y general que actualmente se denomina seguridad en comunicaciones.

1.1. PRINCIPIOS DE SUSTITUCIÓN Y TRANSPOSICIÓN

127. Dentro de la Criptografía de clave simétrica aparecen dos procedimientos de cifrado básicos que se han ido repitiendo sistemáticamente a lo largo de los siglos hasta llegar a nuestros días. Son los principios de sustitución y transposición.

128. PRINCIPIO DE SUSTITUCIÓN: consiste en establecer una correspondencia entre las letras del alfabeto en el que está escrito el mensaje original y los elementos de otro conjunto, que puede ser el mismo o distinto alfabeto. De esta forma cada letra del texto claro se «sustituye» por su símbolo correspondiente en la elaboración del criptograma. En recepción, el legítimo receptor, que conoce también la correspondencia establecida, sustituye cada símbolo del criptograma por el símbolo correspondiente del alfabeto original, recuperando así la información inicial.

129. PRINCIPIO DE TRANSPOSICIÓN: consiste en «barajar» los símbolos del mensaje original colocándolos en un orden distinto, de manera que el criptograma contenga los mismos elementos del texto claro, pero colocados de tal forma que resulten incomprensibles. En recepción, el legítimo receptor, con conocimiento de la transposición, recoloca los símbolos desordenados del criptograma devolviéndolos a su posición original.

130. Sustitución y transposición son dos ideas muy naturales e intuitivas; se puede decir que es lo que se le ocurre al ser humano cuando tiene que desarrollar un procedimiento de cifrado.

131. Todos los procedimientos criptográficos de clave simétrica están basados en sustituciones, transposiciones o bien en una combinación de ambas. Por ejemplo, el cifrado en flujo es un caso límite de sustitución variable sobre los símbolos de un alfabeto binario, mientras que el DES (Data Encryption Standard, el más emblemático de los procedimientos de cifrado en bloque, véase ANEXO A) no es más que una combinación afortunada de sustituciones y transposiciones.

1.2. CONDICIONES DE SECRETO PERFECTO

132. En 1949, C. Shannon ([Shannon, 1949]) definió sus condiciones de secreto perfecto que permitieron establecer las bases científicas de la Criptografía. Dichas condiciones parten de dos hipótesis distintas:

133. La clave secreta se utilizará solamente una vez, a diferencia de lo que sucedía en los métodos de cifrado antiguos en los que la clave era fija.

134. El enemigo criptoanalista tiene acceso sólo al criptograma, por tanto está limitado a un ataque sobre texto cifrado únicamente.

135. Basándose en estas dos hipótesis, Shannon enunció sus condiciones de secreto perfecto que se definen tal y como sigue:

136. Un sistema criptográfico verifica las condiciones de secreto perfecto si el texto claro X es estadísticamente independiente del criptograma Y, lo que en lenguaje probabilístico puede expresarse como

$$P(X = x \mid Y = y) = P(X = x)$$

para todos los posibles textos fuentes x y todos los posibles criptogramas y. Es decir la probabilidad de que la variable aleatoria X tome el valor x es la misma con o sin conocimiento del valor tomado por la variable aleatoria Y.

137. En términos más sencillos, esto equivale a decir que la información sobre el texto claro aportada por el criptograma es nula. Por tanto, el enemigo criptoanalista no puede hacer una mejor estimación de X con conocimiento de Y que la que haría sin su conocimiento, independientemente del tiempo y recursos computacionales de los que disponga para el procesado del criptograma.

138. Una vez establecidas las condiciones de secreto perfecto, la pregunta natural que uno puede hacerse es: ¿existen cifradores perfectos?

139. La respuesta es afirmativa y a continuación se presenta una familia de cifradores que las verifica.

140. Se considera un método de cifrado en el que tanto texto claro como criptograma y clave tomen valores en un alfabeto de L elementos, esto es $\{0, 1, ..., L–1\}$ y en el que la longitud de la clave, del criptograma y del texto claro sean iguales entre sí e iguales a M. En este caso el número de posibles textos claros, criptogramas y claves son iguales entre sí e iguales a LM.

141. Se supone que:

142. La clave Z se elige de forma completamente aleatoria, es decir

$$P(Z = z) = L^{-M}.$$

143. La transformación de cifrado es

$$Y_i = X_i \oplus Z_i \, , \, (i = 1, ..., M),$$

donde $\oplus$ denota la adición módulo[1] L del i-ésimo elemento de texto claro X_i con el i-ésimo elemento de la clave Z_i para dar lugar al i-ésimo elemento de criptograma Y_i.

144. Fijado un texto claro X = x, a cada posible valor de la clave Z = zj, (j = 1, ..., LM), le corresponde unívocamente un criptograma Y = yj, (j = 1, ..., LM). Entonces, de acuerdo con las condiciones anteriores, es fácil ver que a un mismo texto claro X = x le puede corresponder con igual probabilidad cualquiera de los LM posibles criptogramas.

145. Luego,

$$P(Y = y) = P(Y = y \mid X = x) = L^{-M}.$$

[1] Una operación *módulo un número dado* consiste en realizar la operación y luego considerar como resultado de la misma al resto de la división entre el resultado de la operación y el número dado.

146. Por tanto la información aportada por el criptograma sobre el texto claro es nula, X e Y son estadísticamente independientes y la familia de cifradores basados en la suma módulo L verifica las condiciones de secreto perfecto.

147. Hay que hacer notar que este tipo de cifrado módulo L ofrece una total seguridad respecto a la estadística del texto claro. Se trata de una cualidad muy deseable, puesto que sería extraordinariamente peligroso que la seguridad de un método de cifrado dependiera de la naturaleza estadística del lenguaje utilizado en el mensaje a cifrar.

148. Cuando L = 2, resulta el cifrado Vernam.

1.3. CIFRADO DE VERNAM

149. Este tipo de cifrado apareció en 1917 y su nombre se debe a su inventor, el ingeniero estadounidense G.S. Vernam ([Kahn, 1967]).

150. Tanto el texto claro como el criptograma y la clave utilizan un alfabeto binario pues inicialmente se utilizó para comunicaciones telegráficas.

151. La clave es una secuencia binaria perfectamente aleatoria de la misma longitud que el texto claro.

152. La operación de cifrado es una suma módulo 2 bit a bit (operación lógica OR-exclusiva) entre los dígitos del texto claro y los dígitos de la clave, dando como resultado los correspondientes dígitos del criptograma:

$$Y_i = X_i \oplus Z_i, \ (i = 1, ..., M),$$

donde $\oplus$ denota ahora la adición módulo 2 y M es la longitud del texto claro, clave y criptograma.

153. La operación de descifrado es una suma módulo 2 bit a bit (operación lógica OR-exclusiva) entre los dígitos del criptograma y los dígitos de la clave, dando como resultado los correspondientes dígitos del texto claro:

$$Y_i \oplus Z_i = (X_i \oplus Z_i) \oplus Z_i = X_i, \ (i = 1, ..., M).$$

154. La originalidad del procedimiento Vernam radica en que la clave se utiliza solamente una vez, de ahí que en inglés este procedimiento de cifrado se denomine one-time pad cipher o cifrado con cinta de un único uso.

155. Dado que el cifrado Vernam utiliza una clave de la misma longitud que el texto claro, que ésta es una secuencia perfectamente aleatoria y que además se utiliza solamente una vez, puede concluirse que este tipo de cifrado verifica las condiciones de secreto perfecto.

156. Es decir el cifrado Vernam es un procedimiento incondicionalmente seguro o con una seguridad matemáticamente demostrable. En la actualidad es el único procedimiento de cifrado para el que se puede demostrar esta seguridad incondicional.

2. CIFRADO EN FLUJO

157. Aunque el cifrado Vernam ofrece teóricamente las máximas garantías de seguridad, en la práctica presenta un inconveniente bastante evidente: requiere un dígito de clave secreta por cada dígito de texto claro. Al mismo tiempo y para su implementación, hay que hacer llegar por un canal seguro toda esa cantidad de clave tanto al emisor como al receptor.

158. Teniendo en cuenta la necesidad actual de cifrar grandísimas cantidades de información, el método resulta poco factible para su aplicación generalizada. Queda más bien reservado para aquellas circunstancias en las que se requieren unas condiciones máximas de seguridad con un mínimo de información a proteger (por ejemplo, el teléfono rojo Washington–Moscú en la época de la guerra fría, véase [Sgarro, 1990]).

159. Se trata, por tanto, de un procedimiento de cifrado incondicionalmente seguro pero poco práctico a la hora de su implementación real. Así pues, debe ser modificado hasta convertirlo en algo viable.

160. La estrategia para su modificación es la siguiente: en vez de usar como clave una secuencia binaria perfectamente aleatoria se utilizan las secuencias generadas por generadores pseudoaleatorios; es decir, algoritmos determinísticos que, a partir de una clave corta elegida aleatoriamente (semilla del generador) y conocida únicamente por A y B, generen simultáneamente en emisión y recepción una misma secuencia con la longitud deseada.

161. Esta será la llamada secuencia cifrante que se sumará módulo 2 con el mensaje original (en emisión) o con el criptograma (en recepción) y que hará las veces de clave en el cifrado Vernam.

162. Esta versión modificada del cifrado Vernam es lo que se conoce como cifrado en flujo.

163. El aspecto más favorable es que un cifrado en flujo es ya implementable y susceptible de ser utilizado masivamente, puesto que la clave que hay que hacer llegar a ambos comunicantes es corta (128–256 bits).

164. El aspecto menos favorable es que estas secuencias cifrantes provienen de un algoritmo determinístico, luego nunca serán auténticas secuencias aleatorias. Como mucho serán secuencias pseudoaleatorias o secuencias que se asemejan a las aleatorias pero sin llegar a serlo. Por tanto, dichas secuencias no verificarán propiamente las condiciones de secreto perfecto de Shannon.

165. En cualquier caso, cuando el generador de secuencia pseudoaleatoria está bien diseñado, el cifrado en flujo ofrece un nivel de seguridad suficientemente satisfactorio como para permitir su uso generalizado.

166. En la Figura A.2 se ilustra el esquema general de un procedimiento de cifrado en flujo.

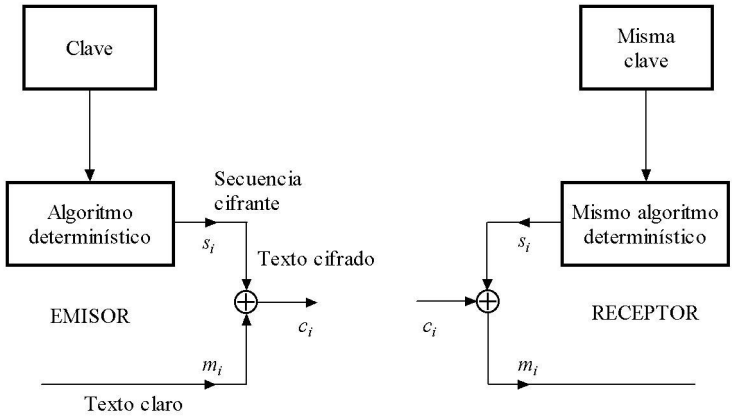


Figura A.2. Esquema general de un procedimiento de cifrado en flujo.

167. El cifrado en flujo es sencillo y rápido, de hecho es el más rápido entre todos los sistemas de cifrado que existen en la actualidad. En resumen, el cifrado en flujo es una aproximación al cifrado Vernam tanto más segura cuanto más se aproxime la secuencia cifrante a una auténtica secuencia aleatoria.

2.1. SECUENCIAS CIFRANTES: CARACTERÍSTICAS GENERALES

168. A continuación se considerarán una serie de características y propiedades que deben verificar las secuencias binarias candidatas a secuencias cifrantes.

169. No es fácil evaluar cuándo una secuencia binaria es lo suficientemente segura para su utilización en Criptografía ya que no existe un criterio global que lo garantice. Sin embargo, sí se pueden señalar ciertas características que toda secuencia cifrante ha de verificar para su correcta aplicación en el procedimiento de cifrado en flujo.

170. Hay que señalar que las siguientes condiciones son necesarias pero nunca suficientes. Si una candidata a secuencia cifrante no las verificase, el generador de secuencia que la produce debería ser desechado. Si por el contrario las verificase, tampoco podríamos asegurar que se tratara de un generador seguro de secuencia cifrante. Estas condiciones son:

171. PERIODO: el periodo de la secuencia cifrante ha de ser al menos tan largo como la longitud de la secuencia a cifrar. Sin embargo, en la práctica es fácil generar secuencias con periodos del orden de 1050 bits o incluso superiores que cumplen sobradamente este requisito criptográfico.

172. DISTRIBUCION DE CEROS Y UNOS: En una secuencia aleatoria, diferentes muestras de una determinada longitud han de estar uniformemente distribuidas a lo largo de toda ella. Previamente a esta caracterización, se introduce una cierta terminología.

173. En una secuencia binaria se denomina racha de longitud k a la sucesión de k dígitos iguales entre dos dígitos distintos. En la bibliografía especializada, las rachas de ceros se denominan gaps y las de unos blocks.

174. La función de auto-correlación AC de una secuencia binaria periódica de período T se define como

$$AC(k) = (A - D)/T$$

donde A y D representan, respectivamente, el número de coincidencias (Agreements) y no coincidencias (Disagreements) entre la secuencia considerada y ella misma desplazada cíclicamente k posiciones.

175. En [Golomb, 1982], Golomb formula tres postulados que toda secuencia binaria finita debe satisfacer para poder ser denominada secuencia pseudoaleatoria:

- *Primer postulado:* En cada periodo de la secuencia considerada, el número de unos tiene que ser igual al número de ceros. Más concretamente, la diferencia entre uno y otro no debe exceder la unidad.
- *Segundo postulado:* En cada periodo de la secuencia considerada, la mitad de las rachas del número total de rachas observadas tiene que tener longitud 1, una cuarta parte longitud 2, una octava parte longitud 3, etc. Al mismo tiempo, para cada una de las longitudes anteriores habrá el mismo número de rachas de ceros que de unos.
- *Tercer postulado:* La auto-correlación $AC(k)$ tiene que ser constante para todo valor de k.

176. Una secuencia binaria finita que verifique estos tres postulados se denomina PN-secuencia y verifica todas las propiedades de una secuencia binaria con distribución uniforme.

177. El primer postulado establece que ceros y unos deben aparecer a lo largo de la secuencia con igual probabilidad.

178. El segundo postulado comprueba que a lo largo de la secuencia las distintas muestras de n dígitos consecutivos (n-gramas) ocurran con la probabilidad correcta. Véase para más detalles el capítulo 2 de [Tilborg, 1988].

179. El tercer postulado comprueba que el cómputo de coincidencias entre una secuencia y su versión desplazada no aporte ninguna información sobre el periodo de la misma, a menos que ésta se desplace sobre sí misma un múltiplo de dicho periodo.

180. Toda secuencia binaria candidata a secuencia cifrante en un proceso de cifrado en flujo debe cumplir necesariamente los tres postulados de pseudoaleatoriedad de Golomb.

181. IMPREVISIBILIDAD: La secuencia cifrante ha de ser imprevisible, quiere esto decir que dada una porción de secuencia de cualquier longitud, un criptoanalista no debería predecir el siguiente dígito con una probabilidad de acierto superior a ½.

182. Una medida de la imprevisibilidad de una secuencia es su complejidad lineal que denota la cantidad de secuencia necesaria que hay que conocer para poder determinar el resto de la misma. En términos criptográficos, la complejidad ha de ser tan grande como sea posible pues eso quiere decir que se necesitaría una enorme cantidad de secuencia cifrante interceptada para poder deducir los restantes bits.

183. Un algoritmo para calcular la complejidad lineal de una secuencia es el algoritmo de Massey-Berlekamp ([Massey, 1969]) que corre en tiempo polinómico, es decir, que emplea un corto periodo de tiempo para calcular la salida correspondiente a una entrada dada.

184. EFECTO AVALANCHA: La secuencia cifrante producida por el generador pseudoaleatorio tiene que depender de todos y cada uno de los bits de la clave o semilla. Es decir, una variación de un único bit de la clave se ha de traducir en una nueva secuencia donde al menos, en promedio, la mitad de los bits estén invertidos con relación a la secuencia generada antes de la modificación de la clave. En caso contrario esto

significaría que algunos de los bits de clave son redundantes, lo que disminuiría el tamaño del espacio de claves.

185. FACILIDAD DE IMPLEMENTACION: La secuencia tiene que ser fácil de generar con medios electrónicos para su total aplicabilidad en el proceso de cifrado/descifrado. En este apartado se incluyen una serie de aspectos técnicos: velocidad de generación (adecuada para comunicaciones de banda ancha), coste de la implementación, tamaño del hardware que la soporta, escalabilidad, consumo, facilidad para recuperar el sincronismo entre emisión y recepción en caso de que se pierda, etc. Todos ellos han de tenerse en cuenta a la hora de implementar el generador de secuencia cifrante.

3. GENERADORES DE SECUENCIA CIFRANTE

186. En este apartado se describen algunos de los generadores más conocidos para la obtención de secuencias pseudoaleatorias. Se presenta inicialmente una estructura básica común a todos ellos.

3.1. REGISTROS DE DESPLAZAMIENTO REALIMENTADOS LINEALMENTE (LFSRS)

187. Del inglés Linear Feedback Shift Register, los LFSRs constituyen el elemento fundamental para la generación de secuencias cifrantes.

188. Un LFSR es un dispositivo electrónico que consta de un número L de celdas de memoria (flip-flops) de contenido binario interconectadas entre sí y numeradas 1, 2, …, L de derecha a izquierda, véase la Figura A.3.

189. A cada pulso de reloj, en el LFSR se realizan las siguientes operaciones:

190. El contenido binario de la celda L sale al exterior y constituye el primer dígito de la secuencia generada.

191. El contenido binario de la i-ésima celda se desplaza a la celda contigua i+1-ésima para todo $1 \leq i < L$.

192. El nuevo contenido binario de la celda 1 será la suma módulo 2 (operación lineal) de las celdas incluidas en el lazo de realimentación y determinadas por el polinomio de realimentación del registro.

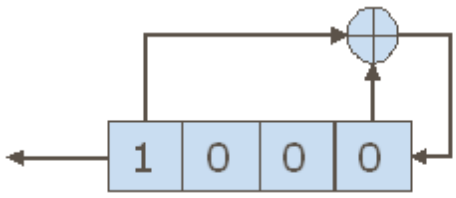


Figura A.3. Registro de desplazamiento realimentado linealmente (LFSR).

193. En el ejemplo de la Figura A.3., el polinomio de realimentación es $P(x) = x4 + x + 1$, pues son las celdas 4 y 1 las entradas a la puerta lógica OR-exclusiva, el 1 (término independiente del polinomio) representa simbólicamente la realimentación a la primera celda.

194. Si dicho polinomio es primitivo (véase [Golomb, 1982]), entonces la secuencia producida es una PN-secuencia o secuencia de período máximo de valor T = 2L – 1.

195. Se denomina estado del registro al contenido de las etapas entre dos pulsos de reloj. El estado inicial corresponde al contenido de las etapas en el momento de empezar el proceso. En el caso representado en la figura este contenido inicial sería (1 0 0 0).

196. La sucesión de estados por los que va pasando el registro de la figura es: (1,0,0,0), (0,0,0,1), (0,0,1,1), (0,1,1,1), … , (0,1,0,0). Es decir, la sucesión ordenada de todas las configuraciones binarias de 4 bits a excepción del estado (0,0,0,0).

197. La PN-secuencia generada sería:1,0,0,0,1,1,1,1,0,1,0,1,1,0,0 de período T = 15, pues en este caso el polinomio de realimentación es primitivo.

198. Las secuencias generadas por LFSRs con polinomios primitivos cumplen perfectamente las propiedades exigibles a una secuencia cifrante (periodo largo, buena distribución estadística de ceros y unos, buena correlación, facilidad de implementación, efecto avalancha, etc.) pero la condición que no verifican es la referente a la imprevisibilidad.

199. Las PN-secuencias son fácilmente previsibles ya que conociendo L bits de la secuencia de salida y el polinomio característico se puede deducir, mediante la resolución de un sistema lineal de L ecuaciones con L incógnitas, el resto de la misma.

200. El problema de la previsibilidad de la secuencia de salida de los LFSRs es inherente al hecho de que son estructuras lineales.

201. En Criptografía todo lo que sea lineal es susceptible de ser criptoanalizado. Sin embargo, las restantes propiedades de las PN-secuencias son excelentes para su uso como secuencias cifrantes.

202. Se trata por tanto de utilizar los LFSRs como estructuras básicas de los generadores pero introduciendo algún elemento de no linealidad. Según sea esta manera de introducir no linealidades se tendrán diversas familias de generadores de secuencia cifrante basadas en LFSRs.

3.2. EJEMPLOS DE GENERADORES DE SECUENCIA CIFRANTE

203. A continuación se exponen diferentes ejemplos de generadores basados en diferentes formas de introducir la no linealidad. En todos ellos la clave constituye el estado inicial de los LFSRs.

204. GENERADOR DE BETH-PIPER: el generador de Beth-Piper ([Beth and Piper, 1984]) forma parte de una familia de generadores de secuencia que introducen desplazamientos irregulares en algunos de sus registros componentes.

205. Tal y como se representa en la Figura A.4., la entrada al reloj del LFSR2 está controlada por la salida del LFSR1, de forma que el LFSR2 se desplaza en el instante t sólo si el bit de salida del LFSR1 en dicho instante es 1.

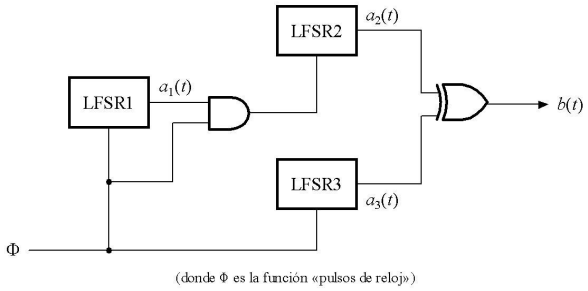


Figura A.4. Generador de Beth-Piper

206. Por tanto, LFSR1 y LFSR3 se desplazan simultáneamente a cada golpe de reloj, mientras que LFSR2 lo hace según le vayan marcando los bits de LFSR1. El bit de salida del generador en un instante t es la suma módulo 2 de los bits de salida de LFSR2 y LFSR3.

207. Si las longitudes de los tres registros son L1, L2 y L3 y sus polinomios de realimentación primitivos, entonces el periodo de la secuencia de salida viene dado por:

$$T = (2^{L1} - 1)\,(2^{L2} - 1)(2^{L3} - 1),$$

y su complejidad lineal es:

$$LC = (2^{L1} - 1)\,L2 + L3.$$

208. Nótese que esta manera de introducir la no linealidad mediante desplazamientos no síncronos de los LFSRs provoca una complejidad lineal exponencial en la longitud de uno de los registros, lo que supone una característica muy deseable desde el punto de vista criptográfico.

209. Así y todo este tipo de generador sí presenta alguna vulnerabilidad a nivel de correlación ya que la probabilidad de encontrar correlaciones entre valores consecutivos de la secuencia de salida y de la secuencia generada por el LFSR3 es alta (véase [Brickell and Odlyzko, 1988]).

210. GENERADOR SHRINKING: el generador shrinking ([Coppersmith, 1994]) es un generador de secuencia cifrante constituido por sólo 2 registros de desplazamiento LFSR1 y LFSR2.

211. Su creador ahondó en la idea de cómo con dos únicos LFSRs se puede diseñar un generador sencillo, rápido y presumiblemente seguro desde un punto de vista criptográfico.

212. El generador shrinking obedece al siguiente esquema, véase la Figura A.5.

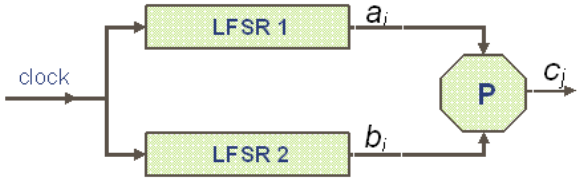


Figura A.5. Generador Shrinking

213. En este caso la no linealidad se introduce mediante la decimación (eliminación) de algunos de los bits generados por uno de los LFSRs.

214. En concreto, la secuencia producida por el LFSR1, notada {an}, décima la secuencia producida por el LFSR2, notada {bn}, según una ley de decimación P. La secuencia de salida se denota por {cn}.

215. Más explícitamente, la regla de decimación se define como:

> Si $a_j = 1$, el bit del registro LFSR2 corresponde al bit de salida, esto es $c_i = b_j$.
> Si $a_j = 0$, el bit del registro LFSR2 se desecha.

216. En resumen, la secuencia de salida de este generador es una decimación irregular de la PN-secuencia producida por el LFSR2 según le vayan marcando los bits de salida del LFSR1. De ahí el nombre de generador shrinking o generador que encoge la secuencia de salida producida por el segundo LFSR.

217. En la literatura, existen métodos criptoanalíticos desarrollados para romper este generador ya sean con una complejidad computacional de tipo exponencial ([Simpson et al., 1998]) o bien con el conocimiento de unos pocos bits específicos (bits estratégicos) de la secuencia cifrante ([Fúster and Caballero, 2008]).

218. GENERADOR A5/1: es un generador de secuencia cifrante que protege la confidencialidad de la conversación entre móvil/estación base y viceversa en telefonía móvil GSM (Global System for Mobile Communications).

219. El A5/1 genera una secuencia cifrante bajo control de una clave de sesión KC de 64 bits.

220. Una conversación GSM puede visualizarse como una sucesión de tramas donde cada una de ellas contiene 114 bits que representan la comunicación digitalizada entre móvil/estación base y otros 114 bits que representan la comunicación digitalizada en sentido contrario.

221. Una vez inicializado, el generador de secuencia cifrante produce 228 bits que se suman módulo 2 con los 228 bits de conversación en claro para producir los 228 bits de conversación cifrada. El procedimiento se repite para cada trama.

222. El esquema general del generador A5/1 aparece representado en la Figura A.6.

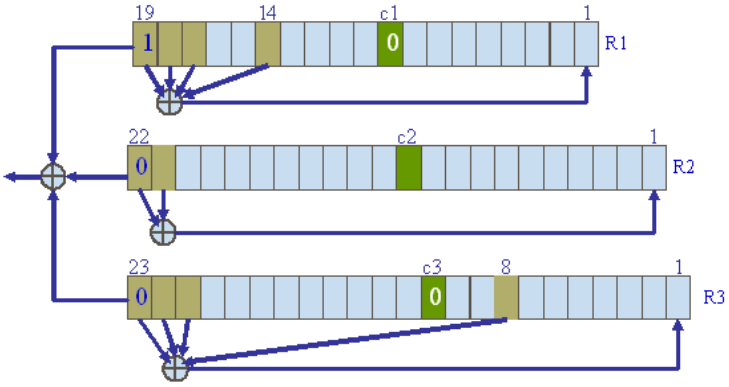


Figura A.6. Generador A5/1 para telefonía GSM

223. Consta de tres registros de desplazamiento realimentados linealmente denotados respectivamente por R1, R2 y R3.

224. El registro R1 tiene una longitud de 19 etapas numeradas de derecha a izquierda y polinomio de realimentación primitivo P1(x) = x19 + x18 + x17 + x14 + 1. La salida del registro se toma como el contenido binario de la etapa 19. El contenido de la etapa 9 (bit

central de R1, notado c1) es una de las entradas a la función que controla el desplazamiento de dicho registro.

225. El registro R2 tiene una longitud de 22 etapas y polinomio de realimentación primitivo P2(x) = x22 + x21 + 1. La salida del registro se toma como el contenido binario de la etapa 22. El contenido de la etapa 11 (bit central de R2, notado c2) es una de las entradas a la función que controla el desplazamiento de dicho registro.

226. El registro R3 tiene una longitud de 23 etapas y polinomio de realimentación primitivo P3(x) = x23 + x22 + x21 + x8 + 1. La salida del registro se toma como el contenido binario de la etapa 23. El contenido de la etapa 11 (bit central de R3, notado c3) es una de las entradas a la función que controla el desplazamiento de dicho registro.

227. La función F (función mayoría) que controla el desplazamiento de los registros tiene como entradas a los bits centrales de cada uno de los registros (c1, c2, c3). Si entre esos tres bits hay mayoría por ejemplo de ceros, entonces sólo aquellos registros cuyo bit central sea cero se desplazarán. En el ejemplo de la Figura A.6. el bit mayoría es el 0, luego los registros R1 y R3 se desplazan, R2 permanece quieto.

228. A cada instante de tiempo t, el bit de salida del generador es la suma módulo 2 de los bits de salida de cada registro.

229. En la actualidad el algoritmo A5/1 sí ha sido criptoanalizado tal y como aparece en [Biryukov et al., 2000].

230. Los requerimientos para su criptoanálisis fueron: 2 minutos de secuencia de salida del generador interceptada (215 tramas), una tabla pre-computada y un PC.

231. La idea consiste en llegar a identificar un estado intermedio de una trama, para a partir del él llegar al estado inicial. El conocimiento del estado inicial nos permite calcular la clave procediendo en orden inverso al procedimiento de inicialización.

232. Hoy día existe un hardware especializado que lleva a cabo este criptoanálisis en tiempo real.

233. GENERADOR E0 (BLUETOOTH): el cifrado de la información transmitida mediante tecnología Bluetooth se lleva a cabo mediante un procedimiento de cifrado en flujo cuyo generador de secuencia cifrante aparece representado en la Figura A.7.

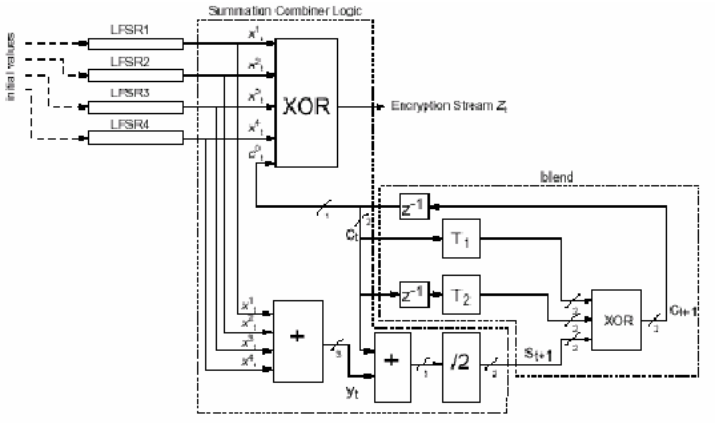


Figura A.7. Esquema general del generador E0 de uso en tecnología Bluetooth

234. La función E0 constituye el generador de secuencia cifrante para el proceso de cifrado/descifrado en este tipo de tecnología para intercambio de ficheros entre dispositivos electrónicos.

235. Consta de 4 registros de desplazamiento LFSR1, LFSR2, LFSR3 y LFSR4 de longitudes respectivamente 25, 31, 33 y 39 etapas o celdas de memoria. La longitud de clave es, por tanto, de 128 bits.

236. La salida de cada uno de estos registros junto con la salida de una función no lineal se suman módulo 2 para producir el correspondiente bit de secuencia cifrante.

237. La función no lineal incluye productos lógicos (AND) y una división aritmética.

238. La inicialización de este generador requiere el contenido inicial de los 4 LFSRs más 4 bits de memoria (nonces).

239. Al igual que la tecnología GSM, Bluetooth también funciona a nivel de un sistema de tramas que se van cifrando sucesivamente. La longitud de trama es ahora de 2745 bits. La diferencia con GSM es que la tecnología Bluetooth cifra cada trama con una clave distinta.

240. Esto implica que el criptoanalista dispone solamente de 2746 bits para desarrollar su criptoanálisis lo cual, en términos criptográficos, es demasiado poco. De ahí que ninguno de los ataques criptoanalíticos desarrollados por vía algebraica o por correlación haya resultado fructífero, véase como más representativo ([Armknecht, 2002]).

241. La inseguridad achacable al Bluetooth es más debida a un mal método de inicialización y cambio de clave que a una debilidad detectada en el diseño del generador de secuencia cifrante.

242. Hasta ahora se han enumerado algunos de los ejemplos más representativos de generadores de secuencia cifrante encontrados en la literatura especializada. En la siguiente sección se presentan las propuestas más recientes de cifradores en flujo englobadas dentro del proyecto eSTREAM.

4. HACIA UN ESTÁNDAR DE CIFRADO EN FLUJO: THE ESTREAM PROJECT

243. El Proyecto eSTREAM surgió en Noviembre de 2004 a iniciativa de la Universidad de Lovaina (Bélgica) para seleccionar un algoritmo de cifrado en flujo que pudiera considerarse como el «estándar» de cifrado en flujo.

244. Sin embargo, los promotores de esta idea nunca utilizaron la palabra estándar, sino que más bien lo definieron como un algoritmo de amplia difusión y uso generalizado. Toda la información correspondiente a este evento puede encontrarse en [eSTREAM, 2008] y [ECRYPT II, 2009].

245. La intención de la convocatoria eSTREAM fue la de estimular trabajos en el área del cifrado en flujo, ya que en los últimos años esta temática había sufrido una constante pérdida de interés.

246. El llamamiento del eSTREAM fue masivamente secundado por la comunidad criptográfica internacional y, tras 6 meses de convocatoria, se presentaron más de 30 algoritmos de cifrado en flujo procedentes de muy diversas nacionalidades y grupos criptográficos, para más detalles véase [Fúster and Pazo-Robles, 2009].

247. Al mismo tiempo una comisión de expertos de reconocido prestigio internacional se encargó de evaluar los algoritmos presentados.

248. Inicialmente se presentaron 34 propuestas diferentes de cifradores en flujo, todos ellos provistos de la documentación pertinente: descripción del generador de secuencia cifrante, implementación software/hardware, resultados de los test estadísticos aplicados, posibles criptoanálisis, modificaciones, análisis y opiniones de la comunidad criptográfica etc. Las características de cada una de estas propuestas pueden encontrarse en [eSTREAM, 2008].

249. En líneas generales los criterios exigidos a estas propuestas eran:

- *Criterios de seguridad*: cualquier ataque criptoanalítico de recuperación de la clave tiene que ser al menos tan costoso como la búsqueda exhaustiva.
- *Criterios de implementación*: una longitud de clave de al menos 128 bits más un vector de inicialización IV de entre 64–128 bits. Cualquiera de las propuestas tenía que tener unas prestaciones superiores a las del AES (estándar criptográfico de cifrado en bloque).
- *Criterios de mercado*: flexibilidad, eficiencia y posibilidad de un uso generalizado a nivel mundial.

250. Las propuestas recibidas se agrupaban en dos perfiles bien definidos:

- *Perfil software*: cifradores en flujo para aplicaciones software con altas prestaciones de salida de secuencia cifrante.
- *Perfil hardware*: cifradores en flujo para aplicaciones hardware con limitaciones en lo que respecta a memoria utilizada, número de puertas y consumo de potencia. En este caso la longitud de la clave se reduce a 80 bits.

251. Tras una primera y segunda fase de evaluación, 16 propuestas pasaron a la tercera fase, 8 orientadas a su implementación software y otras tantas a su implementación hardware (véase [eSTREAM, 2008], Phase 3 candidates).

252. Finalmente, la Comisión encargada de seleccionar un candidato de cada perfil se decantó por un conjunto de cuatro algoritmos software Profile 1 (SW) y 3 hardware Profile 2 (HW), que son los que aparecen en la Cuadro A.1.

Cuadro A.1. Algoritmos ganadores en la fase final

Perfil 1 (Software)	Perfil 2 (Hardware)
HC-128	Grain v1
Rabbit	MICKEY v2
Salsa20/12	Trivium
SOSEMANUK	

253. De esta manera la convocatoria eSTREAM no se redujo a un único candidato de cada tipo, sino que se seleccionó un conjunto de varios para su uso y amplia difusión entre la comunidad criptográfica.

254. A continuación se expone en detalle un ejemplo representativo del perfil hardware.

4.1. GENERADOR TRIVIUM (PERFIL HARDWARE)

255. El algoritmo Trivium fue diseñado por C. De Cannière y B. Preneel como un generador de secuencia cifrante orientado al hardware ([Trivium, 2008]). Se trata de un generador que busca el equilibrio entre simplicidad, seguridad y velocidad.

256. Este generador tiene una clave K de 80 bits y un estado inicial IV de otros 80 bits, pudiendo generar hasta 264 bits consecutivos de secuencia cifrante.

257. El esquema general del generador Trivium aparece representado en la Figura A.8.

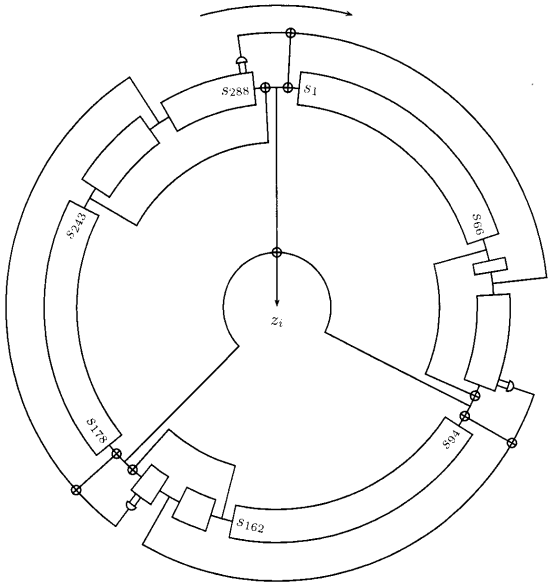


Figura A.8. Generador TRIVIUM

258. El generador Trivium contiene un estado interno de 288 bits, notados (s1, s2, …, s288), y divididos en tres bloques de 93, 84 y 111 bits respectivamente. El primer bloque incluye las etapas (s1, s2, …, s93), el segundo las etapas (s94, s95, …, s177) y el tercero las restantes etapas (s178, s179, …, s288). Cada bloque está conectado con el bloque contiguo dando lugar a un registro circular. A su vez, cada bloque se puede considerar como un registro de desplazamiento no lineal puesto que la realimentación incluye no sólo sumas lógicas sino también productos.

259. La generación de secuencia se realiza mediante un proceso iterativo que extrae los valores de 15 bits del estado interno y los utiliza para actualizar otros 3 bits (t1, t2, t3) con los que, finalmente, calcula el correspondiente bit zi de secuencia cifrante.

260. Los bits de estado son posteriormente rotados y el proceso se repite hasta generar los 264 posibles bits de secuencia de salida correspondientes a una misma clave.

261. En concreto, el procedimiento puede describirse tal y como sigue:

$$t_1 \leftarrow s_{66} \oplus s_{93}$$

$$t_2 \leftarrow s_{162} \oplus s_{177}$$

$$t_3 \leftarrow s_{243} \oplus s_{288}$$

$$z_i \leftarrow t_1 \oplus t_2 \oplus t_3$$

$$t_1 \leftarrow t_1 \oplus s_{91} \cdot s_{92} \oplus s_{171}$$

$$t_2 \leftarrow t_2 \oplus s_{175} \cdot s_{176} \oplus s_{264}$$

$$t_3 \leftarrow t_3 \oplus s_{286} \cdot s_{287} \oplus s_{69}$$

$$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$$

$$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$$

$$(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$$

262. Aquí los símbolos $\oplus$ y $\cdot$ representan respectivamente las operaciones lógicas OR-exclusiva y AND.

263. Nótese que el número que se repite sistemáticamente en este generador es el número 3. Primero se calculan 3 variables (t1, t2, t3) para determinar el correspondiente bit de salida, después se actualizan esos 3 parámetros y finalmente se produce la rotación de los 3 bloques. De ahí, a decir de los autores, el nombre de Trivium.

264. El generador Trivium es un diseño orientado al hardware que pretende ser: compacto en entornos con restricciones sobre el número de puertas lógicas, eficiente en implementaciones con recursos computacionales limitados y rápidos en aplicaciones que requieran una alta velocidad de cifrado.

265. Este algoritmo reúne todas estas características y, hasta el momento, es el más rápido de todos los presentados al eSTREAM.

266. A día de hoy, se considera inmune a todos los ataques del tipo: correlación, guess and determine y ataques algebraicos conocidos. Esto tiene todavía mayor mérito teniendo en cuenta que, dada su simplicidad y velocidad, Trivium ha estado y seguirá estando en el punto de mira de la comunidad criptoanalítica internacional.

267. Criptografía de clave simétrica: Cifradores en bloque

5. CIFRADORES EN BLOQUE

268. Se denomina «Cifrado en bloque» a aquél en que se cifra el mensaje original agrupando los símbolos en grupos (bloques) de dos o más. Algunos sistemas de cifra como el cifrado poligráfico y el cifrado por transposición, son ejemplos elementales de cifrado en bloque.

269. Los cifrados en bloque pertenecen a la categoría de los cifradores de clave secreta, también denominada clave simétrica. Es decir que la clave es única y se emplea tanto para cifrar como para descifrar. La clave ha de ser distribuida mediante un mecanismo seguro a los dos corresponsales que realizan el cifrado, bien mediante un correo seguro o un sistema alternativo de cifrado.

5.1. PROPIEDADES DEL CIFRADO EN BLOQUE

270. Todos los cifrados en bloque tienen las siguientes propiedades:

- *Dependencia entre símbolos*: en cada bloque cada bit del texto cifrado es una función compleja de TODOS los bits de la clave y TODOS los bits del bloque del texto original.

- *Cambio de los bits de entrada*: un cambio de un bit en el bloque del mensaje original produce el cambio del 50%, aproximadamente, de los bits del bloque del mensaje cifrado.
- *Cambio de los bits de clave*: un cambio en un bit de la clave produce, aproximadamente, el cambio de la mitad de los bits del mensaje cifrado.
- *Error de transmisión*: un error en la transmisión de un texto cifrado, se «propaga» a todo el bloque del que forma parte, produciendo un conjunto de errores, en promedio, después del descifrado del 50% de los bits del bloque afectado.

5.2. ARQUITECTURA DEL CIFRADO EN BLOQUE

271. Todos los cifrados en bloque se componen de cuatro elementos:

- Una transformación inicial,
- Una función criptográficamente débil iterada *r* veces, o «vueltas»,
- Una transformación final,
- Un algoritmo de expansión de clave.

272. En la Figura B.1 aparece la estructura general de un cifrado en bloque

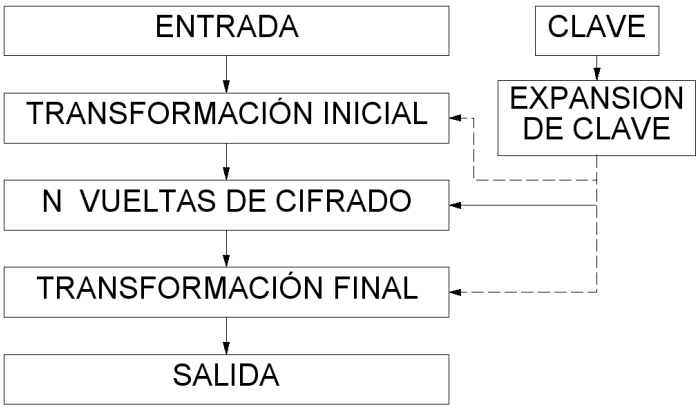


Figura B.1. Estructura del cifrado en bloque.

273. La transformación inicial en algunos sistemas como el «Algoritmo de Cifrado de Datos» (en inglés, Data Encryption Algorithm, DEA) carece de significado criptográfico y su función consiste en aleatorizar simplemente los datos de entrada; pero en otros como el «Estándar de Cifrado Avanzado» (en inglés, Advanced Encryption Standard, AES), puede tener significado criptográfico para entorpecer ataques por análisis lineal o diferencial, en estos últimos casos es función de la clave.

274. Las vueltas intermedias consisten en una función no lineal, complicada, de los datos y de la clave, que puede ser unidireccional (DEA) o no (AES). La función no lineal puede estar formada por una sola operación muy compleja o por la sucesión de varias transformaciones simples.

275. Las vueltas intermedias no han de tener la estructura matemática de grupo, para que el conjunto de varias pasadas sucesivas con sus sub-claves correspondientes no sean equivalentes a una pasada única con una sub-clave diferente, lo que sería equivalente a reducir el número de pasadas y debilitar el algoritmo.

276. La transformación final en algunos sistemas como en el DEA carece de significado criptográfico y su función consiste en invertir la transformación inicial. Pero en otros como el AES, puede tener significado criptográfico.

277. El algoritmo de expansión de clave tiene por objeto convertir la clave de usuario, normalmente de longitud comprendida entre unos 56 a 256 bits, en un conjunto de sub-claves que pueden estar constituidas por varios cientos de bits en total. Conviene que sea unidireccional y que el conocimiento de una, o varias, sub-claves intermedias no permita deducir las sub-claves anteriores o siguientes. Además, se ha de cuidar que las sub-claves producidas no sean un pequeño subconjunto monótono de todas las posibles sub-claves.

5.3. REDES DE FEISTEL

278. Se denominan así a los criptosistemas de cifrado en bloque que constan de varias vueltas de cifrado similares repetidas; las vueltas de cifrado se diferencian entre sí nada más que en que son función de claves diferentes Ki. Inicialmente, el bloque de datos claro se divide en dos mitades y en cada vuelta de cifrado intermedia se transforma, alternadamente, cada una de las dos mitades, tal como se ilustra en la Figura B.2.

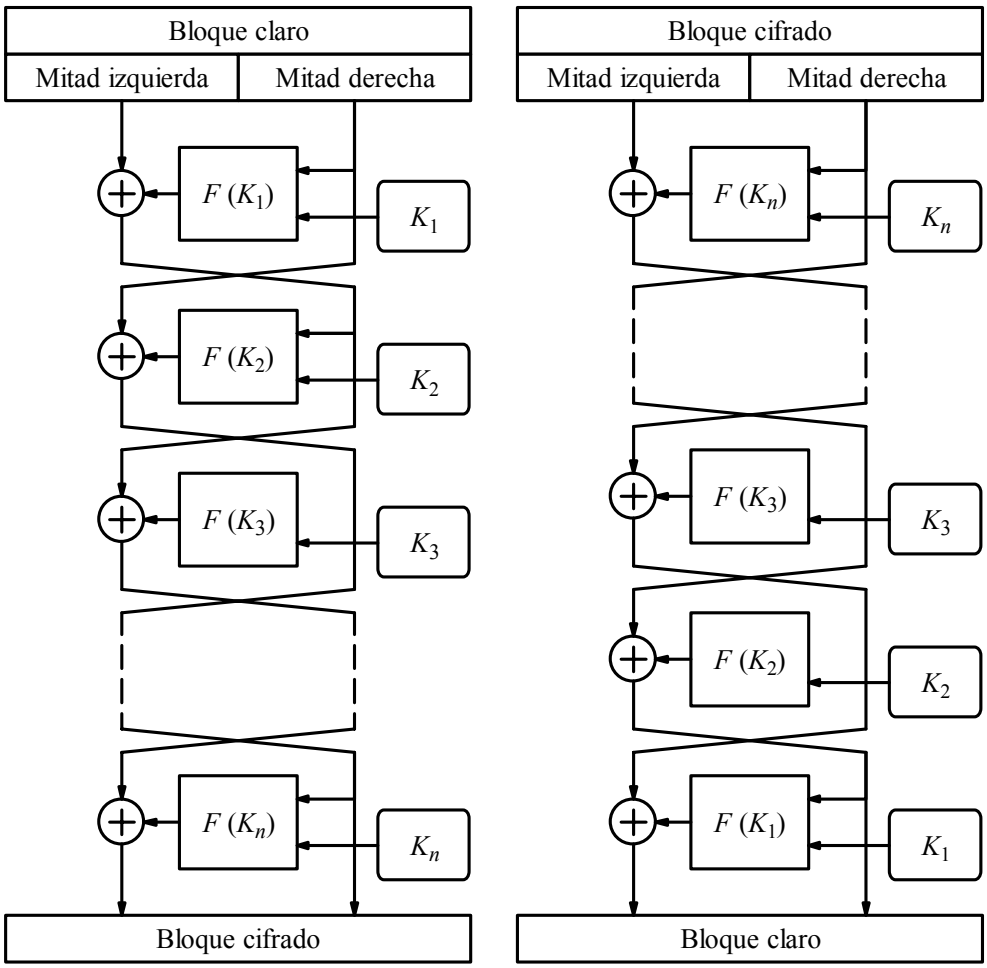


Figura B.2. Redes de Feistel de cifrado (izquierda) y descifrado (derecha).

279. Obsérvese que, en cada vuelta de cifrado i, el sub-bloque anterior derecho pasa al lugar izquierdo posterior sin ninguna modificación; mientras que el sub-bloque anterior izquierdo se suma módulo 2 (operación XOR) con la función F(Ki) del sub-bloque derecho anterior y de la clave Ki de la correspondiente vuelta i, para constituir el sub-bloque posterior derecho. Es decir, que cada vuelta (menos la última) termina con un cruce como se muestra en la Figura B.2. Debido a este esquema, el número de transformaciones que sufre cada bloque, a lo largo del algoritmo, es la mitad del número de vueltas.

280. Es preciso destacar que dos vueltas impares sucesivas, o dos vueltas pares sucesivas, i e i+2, que utilicen la misma clave de vuelta Ki = Ki+2, se neutralizan mutuamente; el resultado neto es equivalente a no haber realizado ninguna de las dos transformaciones. La razón es que la repetición de la operación de la adición módulo 2 de cualquier cantidad tiene por resultado una identidad. A este fenómeno se le denomina «involución». Esta propiedad permite realizar las operaciones de cifrado y descifrado repitiendo idénticamente el algoritmo de principio a fin, con la única excepción del orden de uso de las claves, que en la operación de descifrado ha de ser inverso al que se usó para cifrar.

281. El peligro evidente es que varias claves de vuelta sean idénticas; por tanto una precaución que hay que adoptar es la verificación de que todas las claves utilizadas sean diferentes.

282. Pertenecen a este tipo los criptosistemas DEA, TDEA (Triple DEA), Camellia, MISTI1 y CAST-128.

6. DES Y DEA

283. En 1977 el NBS (National Bureau of Standards, USA) publicó una norma especificando un sistema de cifrado en bloque Data Encryption Standard (Norma o Estándar de Cifrado de Datos), abreviadamente llamado DES. La aprobación y modificación de la propuesta se hizo bajo la supervisión de la NSA (National Security Agency, USA). Este sistema tenía que realizarse obligatoriamente con un microcircuito electrónico que se describió en la Norma FIPS 46-1 ([NIST, FIPS46-1]).

284. En Diciembre de 1993 se revisó la norma y se publicó una nueva, la FIPS 46-2 ([NIST, FIPS46-2]), que incluía la posibilidad de realizar el DES en software, firmware, hardware o una combinación de ellos. El sistema, así realizado, pasó a denominarse Data Encryption Algorithm (DEA).

285. El DEA es un algoritmo de cifrado en bloque cuya longitud de bloque es de 64 bits (ocho símbolos ASCII). La longitud de la clave es de 64 bits; pero 8 de ellos se reservan para paridad, quedando solamente 56 bits de clave útiles, lo que equivale a que existan 256 claves diferentes.

6.1. RETIRADA DEL DES Y DEA

286. Tanto el DES como el DEA han quedado obsoletos debido a su falta de seguridad a causa de la insuficiente longitud de su clave. Se estima que el tiempo requerido en 2010 para romper el sistema mediante un ataque de «prueba exhaustiva de claves» —también llamado «ataque de fuerza bruta»— es de tan sólo 10 horas. En 2005 se han retirado las últimas normas que hacían uso de ellos. En la literatura actual se siguen utilizando los nombres DES y DEA, dependiendo de los autores, mientras que en las normas del

National Institute of Standards and Technology (NIST) se le denomina exclusivamente DEA.

287. En este Anexo se incluye una descripción detallada de DEA tanto por razones históricas como por su facilidad de comprensión.

288. En lugar del DEA —y como sistema transitorio— se ha definido otro algoritmo llamado TDEA (Triple Data Encription Algoritm), que hace uso de tres algoritmos DEA encadenados, de forma que su fortaleza sea suficiente para aplicaciones de seguridad baja y media.

289. La norma que define el TDEA es la [NIST, SP800-67], de mayo de 2004. En ella se redefine nuevamente el DEA como un componente del TDEA, su especificación coincide enteramente con la de las anteriores normas retiradas. El TDEA se explica más adelante, pero como sus componentes consisten en tres DEA, para comprenderlo es preciso estudiar previamente el diseño del DEA.

6.2. ESTRUCTURA DEL DEA

290. El DEA responde a la estructura general del cifrado en bloque representada en la Figura B.1. Las vueltas intermedias constituyen una red de Feistel. En la Figura B.2 aparece el esquema de cifrado del DEA; se parte de un bloque en claro de 64 bits X0 = x0(1), x0(2), …, x0(64).

291. La primera operación es la «transformación inicial», consiste en una permutación fija PF —y, por tanto, sin significación criptográfica— cuya finalidad es lograr una difusión inicial de los bits del bloque, en previsión de que este esté formado por rachas de varios bits 0 seguidas de rachas de bits 1.

292. Después de la transformación inicial se divide el bloque de 64 bits en dos sub-bloques de 32 bits: el derecho D0 = d0(1), . . . , d0(32) y el izquierdo I0 = i0(1), …, i0(32), en donde d0(j) e i0(j) son los bits de los sub-bloques derecho e izquierdo.

293. A continuación tienen lugar 16 vueltas de cifrado todas iguales. Se utilizan 16 sub-claves diferentes: K1, K2, …, K16, construidas mediante un procedimiento de expansión a partir de la clave K, tal como se describe en el apartado 2.4. La salida de cada vuelta responde a las fórmulas:

$$I_n = D_{n-1}, \quad (1)$$
$$D_n = I_{n-1} \oplus F(K_n, D_{n-1}), \quad (2)$$

en donde n es el número de vuelta, Kn es la sub-clave de la vuelta n y el símbolo ⊕ representa la suma módulo 2 bit a bit (también llamada operación XOR bit a bit).

294. Obsérvese que en cada vuelta el sub-bloque de entrada derecho Dn−1 pasa al lugar de salida izquierdo In = Dn-1 sin ninguna modificación; mientras que el sub-bloque de entrada izquierdo In−1 se suma módulo 2 con la función del sub-bloque de entrada derecho y la sub-clave Kn, para constituir el sub-bloque de salida derecho Dn. Es decir que cada vuelta termina con un cruce como se muestra claramente en la Figura B.3. Debido a este esquema, el número de modificaciones que sufre cada bloque a lo largo del algoritmo es la mitad del número de vueltas.

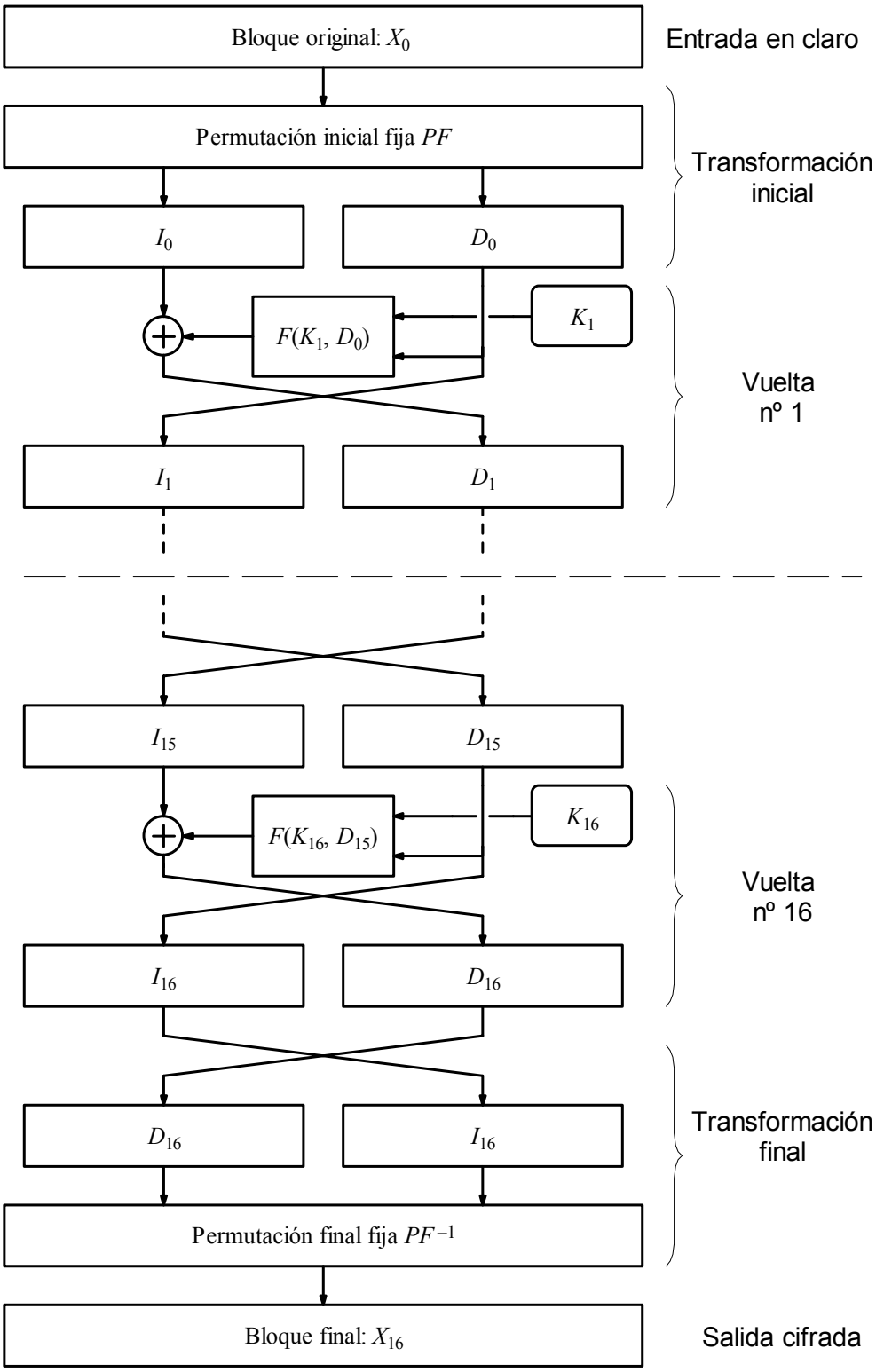


Figura B.3. Estructura del DEA.

295. La transformación final tiene dos etapas, un cruce y una permutación. El cruce consiste en el intercambio de la posición de los sub-bloques de la salida de la vuelta 16, D16 y I16, que pasan a ser D16 = I16 y I16 = D16; este nuevo cruce cancela el cruce final de la vuelta número 16. La arquitectura inicial del DEA, exige que haya dos tipos de vueltas:

las 15 primeras con un cruce y la última sin cruce; pero por economía, para no construir o programar dos tipos de vueltas diferentes, se llama 16 veces a una única rutina de ejecución de vuelta con cruce, siendo entonces necesario el añadir el cruce de la transformación final para anular el efecto del cruce de la última vuelta.

296. Después de la transformación final se obtiene el bloque cifrado:

$$X_{16} = x_{16}(1), x_{16}(2), \ldots, x_{16}(64).$$

6.3. DESCIFRADO E INVOLUCIÓN EN EL DEA

297. Para recuperar un texto claro que ha sido cifrado con DEA es suficiente repetir la misma operación, partiendo del texto cifrado; pero utilizando las sub-claves en orden inverso al que se usó para cifrar: K16, K15, . . . , K1.

298. En la Figura B.4 se ilustra el esquema de descifrado, representando las dos últimas vueltas de cifrado y las dos primeras de descifrado de un DEA. El esquema puede simplificarse ya que las permutaciones fijas final PF−1 y la inicial PF se anulan mutuamente e igualmente se anulan los dos cruces anteriores a la permutación final.

299. La propiedad fundamental de los cifrados de Feistel —y por tanto del DEA— es la «involución», es decir, la ejecución repetida de dos veces de la suma módulo 2 con la función del sub-bloque de entrada derecho y la sub-clave Ki es una operación nula, gracias a que la suma módulo 2 repetida dos veces es una operación nula. Dicho de otro modo, la 1ª vuelta de descifrado cancela la vuelta 16ª de cifrado, la vuelta 2a de descifrado cancela la vuelta 15a de cifrado, etc.

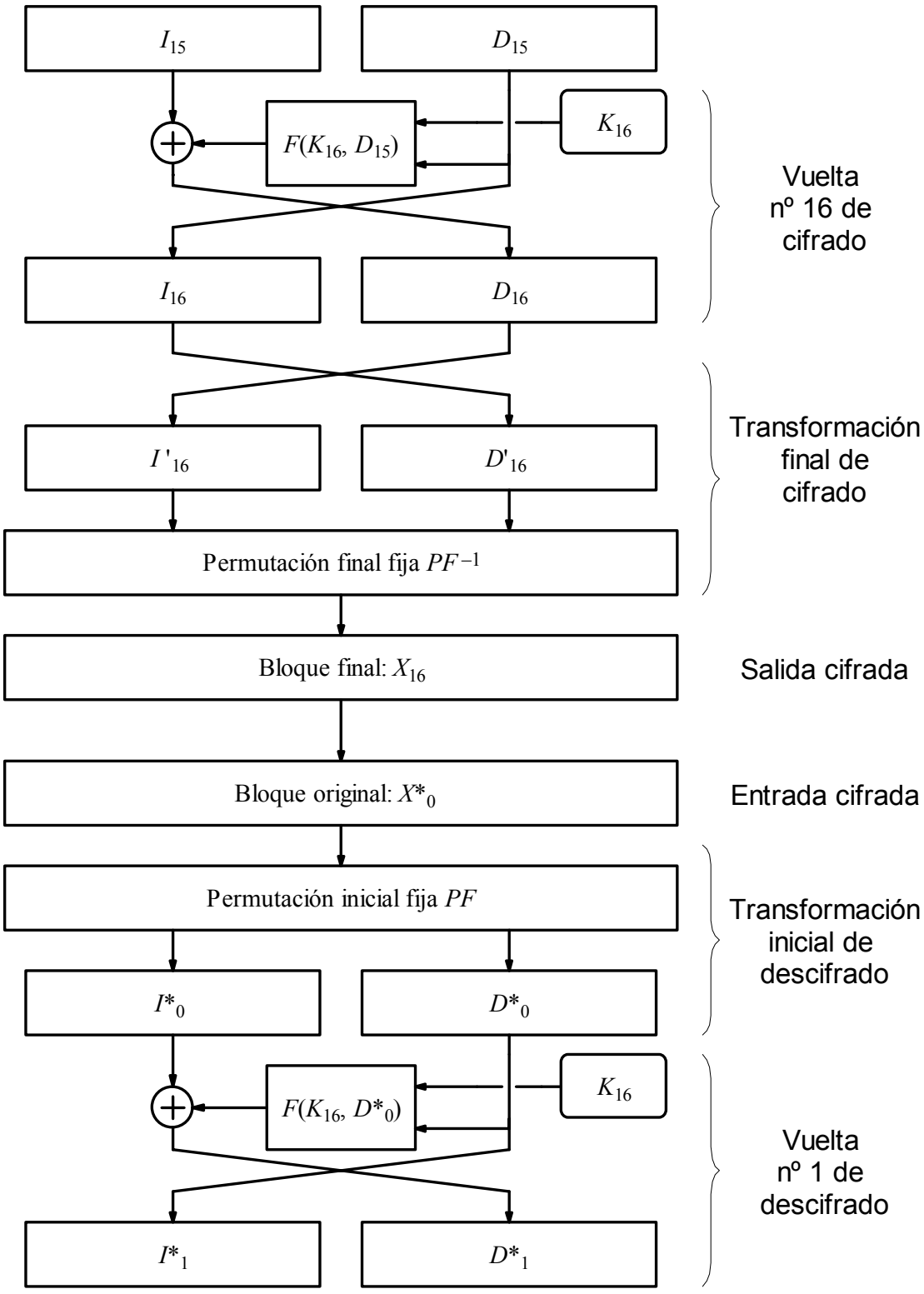


Figura B.4. Descifrado e involución en el DEA.

6.4. EXPANSIÓN DE CLAVES EN EL DEA

300. Como ya se ha dicho, DES tiene una clave de 64 bits de longitud, de los cuales 56 bits son efectivos, mientras que los 8 restantes son de paridad, es decir corresponden a un código detector de errores que garantiza que las claves mal transmitidas se reduzcan al mínimo. Como el DEA necesita 48 bits en la sub-clave de cada vuelta y tiene 16 vueltas, la cantidad total de bits que necesita es 768.

301. Para alcanzar esta cantidad de bits se sigue un procedimiento de expansión, que tiene las siguientes etapas:

- Se realiza una permutación fija de los 56 bits útiles de clave.
- Se dividen los 56 bits permutados en dos mitades de 28 bits.
- Cada mitad se rota a la izquierda uno o dos bits (incrementalmente respecto de la posición en la vuelta anterior) dependiendo del número de vuelta a que corresponda. Haciéndose en total 16 vueltas de rotación a la izquierda.
- Se vuelven a juntar las mitades rotadas de cada vuelta obteniéndose 16 claves de 56 bits.
- Se seleccionan 48 bits determinados de cada una de las 16 claves de 56 bits, siempre de la misma posición (se toman 24 bits de la izquierda y 24 bits de la derecha).
- Se permutan los 48 bits de cada sub-clave, siempre en el mismo orden.

302. Como los desplazamientos han sido diferentes para cada vuelta, las 16 claves de 56 bits del paso d son una permutación diferente de los 56 bits de entrada, pero el mecanismo de selection del paso e, que reduce el número de bits a 48, implica que se utiliza un conjunto diferente de bits en cada sub-clave; cada bit se usa aproximadamente en 14 de las 16 sub-claves.

6.5. PROPIEDAD DE COMPLEMENTACIÓN

303. Sea x el complemento bit a bit de un texto claro x y sea EK la operación de cifrado en DEA con la clave K; entonces tendremos que y = EK(x).

304. El DEA tiene una propiedad denominada de «complementación» que implica: y = EK(x) —donde las variables subrayadas indican la negación de las variables sin subrayar—. Es decir que si se cifra el complemento de un determinado texto claro con una clave complementaria de la que se usó originalmente con el texto claro, se obtiene un texto cifrado complementario del texto cifrado original.

6.6. CLAVES DÉBILES Y SEMI-DÉBILES DEL DES

305. La propiedad de involución conduce a la existencia de las llamadas «claves débiles», son aquellas para las que —una vez ejecutado el algoritmo de expansión de clave— las sub-claves obtenidas son todas iguales entre si. Naturalmente, cada sub-clave Kn anula la transformación que hizo la sub-clave Kn−2 y el resultado es que no hay cifrado. Existen cuatro claves débiles que son las siguientes (en hexadecimal):

```
0101 0101 0101 0101
fefe fefe fefe fefe
1f1f 1f1f 1f1f 1f1f
e0e0 e0e0 e0e0 e0e0
```

306. También existen 12 pares de claves «semi-débiles». Un par de claves semi-débiles K y K' es aquel que cumple EK(EK'(x)) = x, es decir que el cifrado con una de las claves del par semi-débil es equivalente al descifrado con la otra clave.

307. Cuando se elige una clave para cifrar con cualquier criptosistema es preciso asegurarse de que no se está empleando una clave débil o semi-débil. Pero cuando las claves débiles son conocidas y muy escasas en comparación con el número total de claves posibles, la generación de claves aleatorias asegura que la probabilidad de la obtención de una clave débil sea muy pequeña; este es el caso del DEA y hay autores que opinan que se puede prescindir de dicha comprobación. Aunque lo prudente y recomendable es asegurarse de que no se han producido claves débiles ni semidébiles.

308. Las claves débiles y semi-débiles del DEA no han de considerarse como un fallo de seguridad del sistema, sino una particularidad de su estructura, pues eran conocidas desde que se definió el DES y el DEA.

6.7. SEGURIDAD DEL DEA

309. No existe ninguna prueba que garantice que un algoritmo de cifrado sea prácticamente indescifrable sin conocer la claves secreta utilizada (sí existe un algoritmo teóricamente indescriptable: el de Vernam), lo único que existen son demostraciones de que ciertos algoritmos son vulnerables.

310. Hasta hoy nadie ha demostrado ser capaz de «romper» el DEA. Se ha especulado con la posibilidad de la existencia de una «puerta oculta» para descriptar el DEA; pero hasta el momento no hay ninguna evidencia de ello.

311. La opinión generalizada es que el DEA fue un excelente sistema de cifrado. El único problema que presenta es que su espacio de claves resulta excesivamente reducido para el actual estado del arte de la tecnología electrónica. Una clave de 56 bits es claramente insuficiente frente a la potencia de los actuales ordenadores y las posibilidades de integración a gran escala de la tecnología microelectrónica.

312. El primer ataque especializado para el DEA ha sido el criptoanálisis diferencial, descrito en [Biham and Shamir, 1993]. Mediante esta técnica se consigue recuperar la clave del DEA a cambio de un considerable esfuerzo computacional que obliga al análisis de una cantidad ingente de parejas de textos claros y sus correspondientes textos cifrados. La economía frente al esfuerzo necesario para un ataque por fuerza bruta es moderada, porque los diseñadores del DEA ya habían previsto la eventualidad de un ataque de este género.

313. Hoy día, el sistema de ataque al DEA más eficaz lo constituye la prueba exhaustiva de las 256 claves (ataque por fuerza bruta), mediante una máquina masivamente paralela. La existencia de la primera máquina de este género se anunció públicamente en Mayo de 1998 ([EFF, 1998]) y era capaz de probar todas las claves del DEA en 9 días, o lo que es equivalente, el tiempo medio que requería para encontrar una clave era de 4 días y medio. En 2010 se estima que el tiempo necesario para la rotura del DEA por fuerza bruta es de tan solo 10 horas. Por tanto, actualmente se considera que el DEA es un sistema de cifrado carente de seguridad.

7. TRIPLE DEA «TDEA»

314. Al considerarse que el DEA ya no resulta seguro, debido a la escasa longitud de su clave, como medida transitoria, para poder seguir utilizando los millones de circuitos DES y algoritmos DEA del mundo, se decidió definir un nuevo algoritmo, el TDEA, consistente en la concatenación de tres algoritmos DEA, o la realización de tres operaciones sucesivas con un circuito integrado DES.

7.1. CIFRADO MÚLTIPLE

315. Un procedimiento para aumentar el espacio de claves de un cifrado en bloque consiste en hacer un cifrado múltiple o cifrado producto. En la Figura B.5, se ilustra un cifrado múltiple con cualquier sistema de cifrado en bloque. Como puede verse consiste en una repetición del cifrado n veces usando n claves independientes Kn, cada una de longitud L. En principio, puede parecer que la seguridad aumenta proporcionalmente a n, pero se puede demostrar que la longitud efectiva de la clave en bits es tan solo unos $l = L \cdot \lceil n/2 \rceil$ bits, en vez de $L \cdot n$, es decir si $n = 2$ la longitud de clave frente a un ataque por prueba exhaustiva de claves será solamente de L bits, sin ningún beneficio.

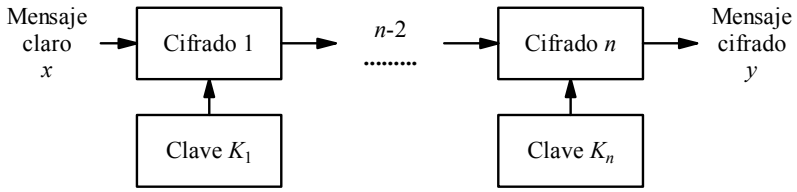


Figura B.5. Cifrado múltiple.

7.2. ESTRUCTURA DEL TDEA

316. El TDEA se describió inicialmente en la norma (ahora retirada) [NIST, FIPS-46-3], a la que sustituyen las normas [NIST, SP800-67] y [ISOIEC, 18033-3R]. La configuración aprobada es la que se ilustra en la Figura B.6, consistente en la concatenación de una operación de cifrado-DEA con clave K1, con una de descifrado-DEA con clave K2 y otra de cifrado-DEA con clave K3. Naturalmente, cuando se descifra con TDEA las operaciones son las opuestas: descifrado-DEA con clave K3, cifrado-DEA con clave K2 y descifrado-DEA con clave K1.

317. Hay tres variantes del TDEA, en función de las claves que se utilicen:

- 1TDEA, variante donde $K_1 = K_2 = K_3$, las tres claves son iguales. La seguridad efectiva es de 56 bits y la longitud real de clave es 56 bits (más 8 bits de paridad). Se utiliza solamente como medio de compatibilidad con el DES y el DEA sencillos, para descifrar documentos antiguos cifrados con DES o DEA.
- 2TDEA, variante en la que $K_1 = K_3 \neq K_2$, la primera y tercera claves son iguales mientras que la segunda es diferente e independiente de las otras. La seguridad efectiva frente a un ataque es de unos 80 bits y la longitud real de clave es 112 bits (más 16 bits de paridad). Se utiliza cuando el nivel de seguridad requerido es mínimo.
- 3TDEA, variante en la que $K_1 \neq K_2 \neq K_3 \neq K_1$, las tres claves son diferentes e independientes entre sí. La seguridad efectiva frente a un ataque es de 112 bits,

mientras que la longitud real de clave es 168 bits (más 24 bits de paridad). Se utiliza cuando el nivel de seguridad requerido es medio.

318. Nótese que ninguna de las variantes descritas del TDEA alcanza nivel de seguridad elevado. En caso de necesidad de un nivel de seguridad elevado ha de recurrirse a otros criptosistemas con claves de seguridad efectiva comprendida entre 128 y 256 bits.

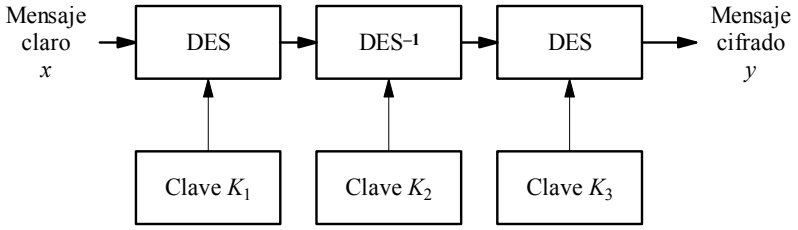


Figura B.6. TDEA.

8. AES Y RIJNDAEL

319. En 1996, el Instituto Nacional de Estándares y Tecnología (National Institute of Standards and Technology, NIST) dio los primeros pasos para la creación de un Estándar de Cifrado Avanzado (Advanced Encryption Standard, AES). Su objetivo fue desarrollar una especificación para encontrar un algoritmo de cifrado que sustituyera al anticuado DES, de manera que el nuevo algoritmo fuese capaz de proteger la información sensible de los ciudadanos y del gobierno hasta bien entrado el siglo XXI. El algoritmo seleccionado será utilizado por el Gobierno de Estados Unidos para proteger información sensible no clasificada y voluntariamente por el sector privado. Por extensión, presumiblemente, sería adoptado por el resto de países del mundo.

320. En septiembre de 1997, se realizó una convocatoria para la presentación de algoritmos, los propuestos tenían que soportar obligatoriamente una longitud de bloque de 128 bits y una longitud de clave de 128, 192 y 256 bits, al margen de cualesquiera otras longitudes posibles.

321. A la citada convocatoria acudieron diversos diseñadores. La selección se llevó a cabo en varias etapas eliminatorias; se hizo de forma pública basándose en las críticas, evaluación y criptoanálisis realizados por la comunidad científica internacional.

322. Finalmente, el 2 de octubre de 2000, el NIST anunció el algoritmo ganador, denominado Rijndael por sus autores Vincent Rijmen y Joan Daemen.

323. Los criterios considerados en la selección fueron los siguientes:

- Seguridad (es decir, el esfuerzo necesario para criptoanalizarlos).
- Eficiencia computacional.
- Requisitos de memoria.
- Adecuación hardware y software.
- Simplicidad de diseño.
- Flexibilidad.
- Requisitos de licencia (no patentado, ni patentable).

324. El 26 de noviembre de 2001 el NIST publicó la norma definitiva, la FIP 197 ([NIST, FIPS197]), que entró en funcionamiento el 26 de mayo de 2002. El texto completo se puede localizar en http://csrc.nist.gov/publications/.

325. El Rijndael es un cifrador en bloque, que opera longitudes de Nb palabras de 32 bits; los valores de Nb pueden ser 4, 6 y 8; la longitud de clave es de Nk palabras de 32 bits; los valores de Nk pueden ser 4, 6 y 8. Es decir que tanto las longitudes de clave y como de bloque pueden ser de 128, 192 ó 256 bits. El Rijndael es fácilmente adaptable a cualquier longitud de bloque y/o clave múltiplo de 32 bits.

326. Este Anexo se limita a describir la versión finalmente aprobada y publicada por el NIST, que restringe la longitud de bloque a 128 bits (Nb = 4), mientras que las claves pueden ser de 128, 192 ó 256 bits.

8.1. ESTRUCTURA DEL AES

327. El AES es un cifrador iterado, relativamente sencillo, que emplea funciones invertibles y opera con bloques enteros, a diferencia de los cifradores de tipo Feistel que lo hacen alternativamente con mitades de bloque.

328. Al resultado obtenido en cada paso del algoritmo se le denomina «Estado», siendo éste un conjunto de tantos bits como la longitud del bloque. Los bits adyacentes se agrupan de 8 en 8 formando bytes y estos en una tabla cuadrada de 4 filas y 4 columnas.

329. Al principio del algoritmo se copian los bytes de entrada eni, —numerados correlativamente en vertical y de izquierda a derecha—, en la tabla de Estado, como se indica en el Cuadro A.1. Los bytes de salida se numeran de igual manera que los de entrada. Los bytes de entrada se convierten en bytes de estado y se numeran con dos subíndices, el primero corresponde a la fila ocupada y el segundo a la columna; aunque después de cada operación de cifrado pueden cambiar de lugar y valor.

Cuadro A.1. Cuadro de estados del AES, con entrada y salida.

Bytes de entrada	Estado	Bytes de salida
en_0 en_4 en_8 en_{12}	$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$	sal_0 sal_4 sal_8 sal_{12}
en_1 en_5 en_9 en_{13}	$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$	sal_1 sal_5 sal_9 sal_{13}
en_2 en_6 en_{10} en_{14}	$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$	sal_2 sal_6 sal_{10} sal_{14}
en_3 en_7 en_{11} en_{15}	$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$	sal_3 sal_7 sal_{11} sal_{15}

330. En la Figura B.7 se ilustra el proceso de cifrado completo, que consta de tres etapas:

- Una transformación inicial,
- *Nr*–1 vueltas regulares,
- Una vuelta final.

331. La transformación inicial consiste en una suma módulo 2 bit a bit (XOR bit a bit) de los primeros 128 bits de la clave con el mensaje claro.

332. El número de vueltas Nr es función de la longitud de la clave: Nr = 10 para clave de 128 bits, Nr = 12 para clave de 192 bits y Nr = 14 para clave de 256 bits.

333. La transformación que tiene lugar en cada vuelta regular de cifrado está compuesta a su vez por cuatro transformaciones diferentes:

- `SubBytes()`: sustitución no lineal de bytes.
- `ShiftRows()`: desplazamiento de las filas del Estado cíclicamente, con diferentes saltos.
- `MixColumns()`: mezcla de columnas.
- `AddRoundKey()`: suma módulo 2 (XOR) con la subclave de vuelta correspondiente.

334. La vuelta final es casi igual a las vueltas regulares; la diferencia consiste en la no existencia de la tercera transformación.

335. La inversión del cifrado (el descifrado) se realiza efectuando las operaciones inversas de las empleadas en el cifrado, en orden inverso al del utilizado en el cifrado.

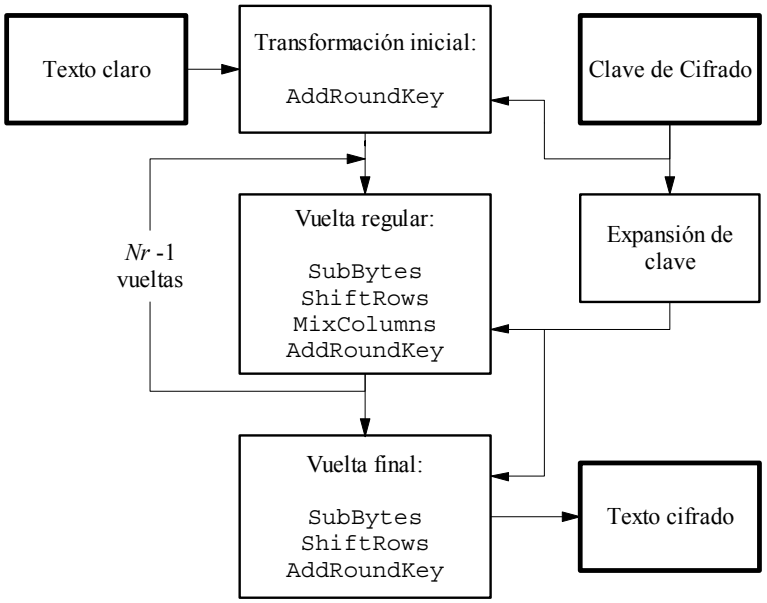


Figura B.7. Esquema general del AES.

8.2. ESQUEMA DE CLAVE EN EL AES

336. La longitud de clave en el AES puede ser de 128, 192 ó 256 bits, siendo necesario generar a partir de ella muchas subclaves: la subclave inicial más Nr subclaves de vuelta, cada una de ellas de 4 palabras de 32 bits. Las subclaves requeridas se generan por derivación de la clave de cifrado mediante un procedimiento de expansión cuyo esquema varía en función de la longitud de la clave de cifrado.

337. Dada una clave de cifrado K, la expansión de clave en el AES se realiza construyendo una secuencia W, de 4(Nr + 1) palabras Wi de 32 bits. Es decir, que con la clave de 128 bits se generarán 44 palabras de 32 bits, con la clave de 192 bits se generarán 52 palabras de 32 bits y con la clave de 256 bits se generarán 60 palabras de 32 bits.

338. En todos los casos, las Nk palabras de la clave constituyen las primeras Nk palabras de la clave expandida W0 ... WNk−1.

339. En la Figura B.8 se muestra el principio del esquema de bloques de la expansión de claves para una clave de 128 bits, es decir Nk = 4, siendo la cantidad total de palabras de 32 bits de la clave expandida 4(Nr + 1) = 44. Si las palabras de la clave expandida se alinean en filas de longitud Nk = 4, podemos observar que las nuevas palabras se fabrican

a partir de la suma módulo 2, bit a bit, de la palabra anterior y la palabra situada inmediatamente encima, con la excepción de la primera columna. Las palabras de la primera columna se obtienen a partir de la suma módulo 2, bit a bit, palabra situada inmediatamente encima y de una transformación compleja de la palabra anterior (es decir la última de la fila anterior). Esta transformación consistente en la aplicación sucesiva de las tres transformaciones elementales RotWord(), SubWord() y Rcn=Rcon[n].

340. En la Figura B.8 sólo se presentan las 12 primeras palabras de las 44 totales. En ella se usan las siguientes abreviaturas para los bloques: RW = RotWord(), SW=SubWord()y Rcn=Rcon[n].

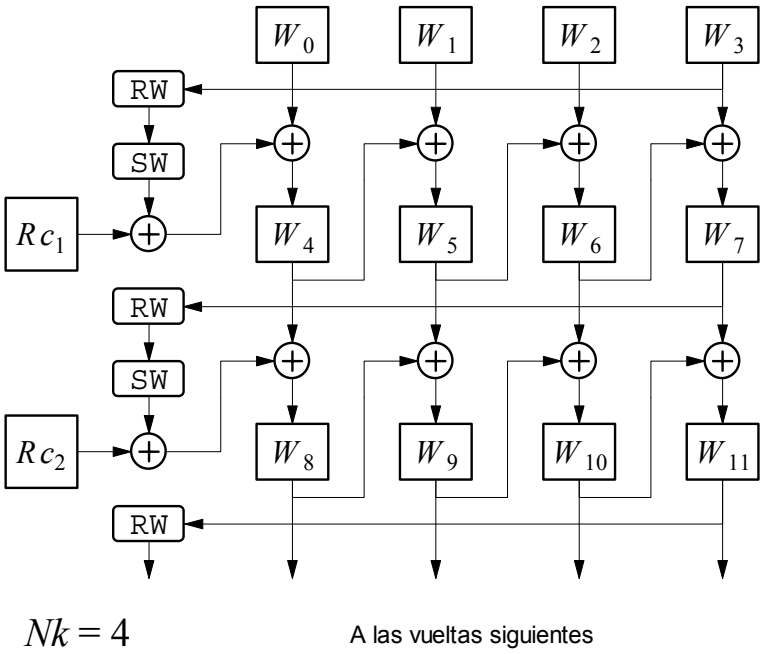


Figura B.8. Principio del esquema de bloques de la expansión de claves del AES, para una clave de 128 bits. Se muestran las 12 primeras palabras de las 44 totales.

341. Procesos análogos se llevan a cabo para la expansión de claves de 192 y 256 bits.

9. MODOS DE USO COMUNES AL DES, DEA, TRIPLE DES Y AES

342. Los sistemas de cifrado en bloque usados directamente solo deben operar sobre conjuntos de información muy reducidos, por lo que son adecuados para cifrar pequeños mensajes, como otras claves, identificaciones, firmas, contraseñas etc. Pero son totalmente inadecuados para el cifrado de grandes cantidades de datos tales como textos muy formateados, listados, programas, tablas y formularios, ya que la estructura del documento se detecta fácilmente; mucho peor aún resulta su aplicación al cifrado de gráficos.

343. Por ejemplo un bloque del TDEA son 64 bits, equivalentes a 8 símbolos ASCII, resulta que cada coincidencia repetida de un conjunto de 8 símbolos quedará cifrada de manera idéntica. Si el mensaje es muy largo —piénsese en imágenes de Mbytes, o videos de

Gbytes— se producirá una cantidad ingente de repeticiones, que darán lugar a un análisis estadístico por distinción y conducirán al compromiso total o parcial del sistema, pues aunque no se recupere la clave, se puede llegar a estimar el contenido aproximado del mensaje, en función de su distribución estadística, lo que es totalmente inaceptable en un sistema de cifrado.

344. Para el cifrado de mensajes grandes el único sistema de cifrado admisible es el cifrado en flujo, porque en él, a cualquier cantidad de datos idénticos en claro siempre le corresponderán diferentes datos cifrados, imposibilitando cualquier análisis estadístico.

345. Una posible solución para cifrar grandes cantidades de datos con un algoritmo de cifrado en bloque es la construcción de un cifrador en flujo, o algo que se le parezca, mediante una arquitectura apropiada que incluya un cifrador en bloque como elemento básico.

346. Se ha convenido en denominar al uso directo de cualquier cifrador en bloque como modo de «libro electrónico de códigos». En la norma [NIST, SP 800-38A] se definen los cinco modos de uso para el cifrado siguientes:

- Modo Libro Electrónico de Códigos (en inglés, Electronic Codebook, ECB).
- Modo Encadenamiento de Bloques Cifrados (en inglés, Cipher Block Chaining, CBC).
- Modo Realimentación del Texto Cifrado (en inglés, Cipher Feedback, CFB).
- Modo Realimentación de la Salida (en inglés, Output Feedback, OFB).
- Modo Contador (en inglés, Counter Mode, CTR).

347. En los apartados siguientes se describen los diferentes modos de uso de los cifradores en bloque. La nomenclatura utilizada es la siguiente: K es la clave que utiliza el cifrador; CIFK significa la operación de cifrado en bloque utilizando la clave K; CIFK–1 significa la operación inversa, con la misma clave K; b es la longitud del bloque correspondiente al algoritmo de cifrado (en TDEA b = 64, en AES b = 128). Cuando el mensaje a cifrar tenga más de un bloque se designará el número de bloques como N.

9.1. MODO LIBRO ELECTRÓNICO DE CÓDIGOS (ECB)

348. El modo ECB de un cifrador en bloque ha de reservarse para conjuntos pequeños de datos, que no ocupen más allá de unos cuantos bloques, jamás debe usarse para cifrar documentos largos. Sus propiedades son:

349. Cada bloque cifrado es solamente función del bloque claro correspondiente y de la clave.

350. Cada bloque descifrado es solamente función de la clave y del bloque cifrado correspondiente. Por ello cada error de transmisión afecta únicamente a un bloque de bits; pero se produce una multiplicación del error, pues un solo bit erróneo afecta a todo el bloque (aproximadamente la mitad de los bits del bloque cambian de valor). Así en el caso del TDEA un error, o grupo de errores, en la transmisión de un bloque de 64 bits produce un bloque erróneo de 64 bits; en el caso del AES el error afectaría a 128 bits.

351. Se puede cifrar y descifrar de forma paralelizada —si se dispone de varios módulos de cifrado o descifrado que se puedan utilizar simultáneamente— pues cada bloque depende únicamente de sí mismo.

352. Dado un texto cifrado, se puede descifrar una parte de sus bloques sin necesidad de descifrar los bloques anteriores ni posteriores.

353. Si el texto claro tiene varios bloques idénticos, el texto cifrado tendrá idénticos todos los bloques correspondientes —por supuesto diferentes de los bloques del texto claro—.

354. Solamente se pueden cifrar bloques completos, de manera que los textos claros han de tener un número de bits exactamente múltiplo de la longitud de bloque del cifrador utilizado. En caso contrario se complementará el último trozo del texto claro con tantos ceros como sea necesario, hasta alcanzar el tamaño de bloque cifrador utilizado.

9.2. MODO ENCADENAMIENTO DE BLOQUES CIFRADOS (CBC)

355. En este caso, se divide el mensaje claro en N bloques de b de bits; si el último bloque N tiene menos de b bits se completa con ceros hasta alcanzar b bits. Cada bloque claro se suma módulo 2 bit a bit con el bloque cifrado anterior; menos el primer bloque que se suma módulo 2 bit a bit (XOR) con un vector inicial.

356. El primer bloque en claro se suma módulo 2 bit a bit con un «vector inicial». No importa que el vector inicial no sea secreto, pero sí es obligatorio que sea elegido de forma aleatoria para evitar ataques por repetición. El vector inicial ha de tener la misma longitud del bloque claro, es decir b bits.

357. Las propiedades del modo CBC son:

- Cada bloque cifrado es función de todos los bloques claros anteriores, del vector inicial y de la clave.
- Cada bloque descifrado es solamente función de la clave, del bloque cifrado correspondiente y del bloque cifrado anterior (excepto el primer bloque). Por ello cada error de transmisión afecta a dos bloques, es decir que cada bit de error aislado produce 2O*b* bits consecutivos erróneos (128 en el TDEA y 256 en el AES).
- Como consecuencia de la propiedad anterior, este modo se comporta como un cifrado autosincronizante, pudiéndose empezar a descifrar a partir de cualquier punto.
- Se puede hacer que cifre mensajes iguales de forma diferente con solo cambiar cada vez el vector inicial.
- No cambia el tamaño del espacio de claves.
- No se puede cifrar de forma paralelizada, pues para cifrar cualquier bloque es preciso disponer previamente del bloque cifrado anterior.
- Se puede descifrar de forma paralelizada: si se dispone de *N* descifradores trabajando en paralelo se pueden descifrar *N* bloques a la vez, disminuyendo notablemente el tiempo requerido para descifrar

9.3. MODO REALIMENTACIÓN DEL CIFRADO (CFB)

358. La principal característica del modo de uso de un cifrador en bloque por realimentación del texto cifrado (CFB) es que realimenta los sucesivos segmentos cifrados en los bloques de entrada del cifrador de la etapa siguiente, cuya salida se suma módulo 2 bit a bit con el texto claro, para producir el texto cifrado y viceversa.

359. En este esquema no se divide el mensaje claro en bloques de b bits exactamente, sino en segmentos de longitud s, tal que $1 \leq s \leq b$. Los tamaños recomendados para s son: 1, 8, 64 y 128 bits.

360. El primer bloque de salida claro se obtiene cifrando en bloque un vector inicial de longitud b bits. No importa que el vector inicial no sea secreto, pero sí es obligatorio que sea elegido de forma aleatoria para evitar ataques por repetición.

361. Del bloque de salida de b bits, así obtenido, se seleccionan los s bits más significativos, que se suman módulo 2 bit a bit con el primer segmento de s bits del texto claro, obteniendo el primer segmento cifrado de s bits.

362. 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 texto cifrado anterior.

363. Las propiedades del modo CFB son:

- Cada bloque cifrado es función de todos los segmentos claros anteriores, del vector inicial y de la clave.
- Cada bloque descifrado es solamente función de la clave, del segmento cifrado correspondiente y del segmento cifrado anterior (excepto el primer bloque). Cada error de transmisión se propaga a los b/s segmentos siguientes, resultando que cada bit de error aislado produce $b+s$ bits consecutivos erróneos.
- Como consecuencia de la propiedad anterior se comporta como un cifrado autosincronizante, pudiéndose empezar a descifrar a partir de cualquier punto.
- Se puede hacer que cifre mensajes iguales de forma diferente con solo cambiar cada vez el vector inicial.
- No cambia el tamaño del espacio de claves.
- No se puede cifrar de forma paralelizada, pues para cifrar cualquier bloque es preciso disponer previamente del bloque cifrado anterior.
- Se puede descifrar de forma paralelizada: si se dispone de N descifradores trabajando en paralelo se pueden descifrar N bloques a la vez, disminuyendo notablemente el tiempo requerido para descifrar.

9.4. MODO REALIMENTACIÓN DE LA SALIDA (OFB)

364. El esquema operativo del modo Realimentación de la Salida convierte un cifrador en bloque (por ejemplo, TDEA o AES) en un auténtico cifrador en flujo. Para empezar se toma un vector inicial que se cifra de forma repetida, obteniéndose N bloques de b bits, que constituyen una secuencia cifrante. El mensaje cifrado es la secuencia de los bloques del mensaje en claro sumados módulo 2 bit a bit con los bloques de la secuencia cifrante. El vector inicial debe de ser secreto, aleatorio y de uso único para cada mensaje.

365. Las propiedades de este modo de cifrado son:

- Cada bloque cifrado es función únicamente del vector inicial, de la clave y del bloque de texto claro correspondiente.
- Cada bloque descifrado es solamente función del vector inicial, de la clave y del bloque cifrado correspondiente.
- Cada error de transmisión afecta a un solo bit, es decir que cada bit de error aislado produce únicamente 1 bit erróneo en el texto claro.
- No es autosincronizante.
- Se puede hacer que cifre mensajes iguales de forma diferente con solo cambiar cada vez el vector inicial.

- No cambia el tamaño del espacio de claves.
- No se puede cifrar ni descifrar de forma paralelizada, pues para cifrar o descifrar cualquier bloque es preciso disponer previamente de todos los bloques anteriores de la secuencia cifrante.
- Para evitar ataques por distinción, la cantidad de bloques máxima del texto a cifrar, debe ser inferior a la raíz cuadrada del número de estados posibles del cifrador en bloque usado, es decir para el TDEA debe ser inferior a 2^{32} bloques y para el AES inferior a 2^{64} bloques.

9.5. MODO CONTADOR (CRT)

366. El Modo Contador de uso de los cifradores en bloque es equivalente a un cifrado en flujo, pues se crea una serie cifrante bloque a bloque cifrando con el cifrador en bloque los diferentes contadores, que luego se suma módulo 2 bit a bit con los sucesivos bloques del mensaje claro o del cifrado. El proceso de cifrado de los bloques se puede hacer con anterioridad a disponer del mensaje en claro o en cifrado; de esta forma se ahorra tiempo, pues al disponerse del mensaje solo es necesario hacer la operación de suma módulo 2, que es sumamente rápida.

367. Para cada clave distinta que se use, el contenido de cada contador ha de ser diferente y no reutilizarse; es decir si se usa una «clave de sesión» y cada sesión incluye el cifrado de varios documentos, los contadores han de ser diferentes para cada documento que se cifre. Una forma sencilla de conseguir este fin es construir de forma aleatoria un «número de uso único» (en inglés, number used once: nonce) como cabecera del mensaje cifrado. El contenido del contador nº1 sería el propio nonce, los contenidos de los contadores serían los sucesivos incrementos del primer contador. El nonce cumple el mismo papel que el vector inicial en otros modos.

368. Las propiedades del modo CTR de cifrado en bloque son:

- Cada bloque cifrado es función del nonce, del incremento del contador, de la clave y del correspondiente bloque de claro.
- Cada bloque descifrado es función del nonce, del incremento del contador, de la clave y del bloque cifrado correspondiente.
- Cada error de transmisión afecta a un solo bit, es decir que cada bit de error aislado produce únicamente 1 bit erróneo en el texto claro.
- No es autosincronizante.
- Se puede hacer que cifre mensajes iguales de forma diferente con solo cambiar cada vez el nonce.
- No cambia el tamaño del espacio de claves.
- Se puede cifrar y descifrar de forma paralelizada.
- Para evitar ataques por distinción, la cantidad de bloques máxima del texto a cifrar, debe ser inferior a la raíz cuadrada del número de estados posibles del cifrador en bloque usado, es decir para el TDEA debe ser inferior a 2^{32} bloques y para el AES inferior a 2^{64} bloques.

10. CÓDIGO DE AUTENTICACIÓN DE MENSAJE (CMAC)

369. Un Código de Autenticación de Mensaje (en inglés, Message Authentication Code, MAC) consiste en un bloque de bits que actúa como resumen de un mensaje y permite verificar la autenticidad e integridad de éste (ver sección 7 del ANEXO C). Funciona de tal manera que la variación de un solo bit del mensaje original produce un cambio total e imprevisible en los bits del MAC. Cuando se desea garantizar que un mensaje no ha sido manipulado, se calcula su MAC (con una clave secreta) y se guarda con el mensaje; posteriormente se puede comprobar la autenticidad e integridad realizando otro MAC con la misma clave, si ambos MACs coinciden se concluye que el mensaje es auténtico e íntegro.

370. Tradicionalmente se ha usado como MAC el último bloque de un cifrado en modo CBC; consecuentemente el MAC depende también de la clave usada en cifrador en bloque, que ha de permanecer secreta y solo compartida por el creador y el verificador del MAC.

371. La norma [NIST, SP800-38B] define un MAC denominado CMAC, para usarse con los cifradores en bloque AES y TDEA, aprobados en las normas [NIST, FIPS 197] y [NIST, SP800-67].

372. El CMAC, además de la clave de cifrado de mensaje acostumbrada K, utiliza dos subclaves, K1 y K2 que se derivan de la clave K mediante un algoritmo definido en [NIST, SP800-38B]. El uso de una u otra subclave depende de que el mensaje tenga una longitud exactamente igual o diferente de b bits, siendo b el tamaño de bloque del algoritmo que se esté usando: b = 64 en TDEA y b = 128 en AES.

373. Cuando el mensaje tiene un tamaño igual a un múltiplo del tamaño de bloque del cifrador empleado se utiliza la primera subclave K1. En caso contrario se utiliza la segunda subclave K2. En este caso, se completa el último bloque del mensaje con un bit «1» seguido de tantos bits «0» como sea necesario hasta completar los bits necesarios.

374. En la Figura B.9 (a) se ilustra el diagrama de bloques del CMAC, en el caso en que el mensaje tenga un tamaño igual a un múltiplo exacto de b bits. El mensaje se divide en N bloques de b bits y se cifra de modo CBC durante los N–1 primeros bloques del mensaje; pero el bloque MN se suma modulo 2 bit a bit con la clave K1 previamente a su cifrado, una vez realizado este, se obtiene una palabra de b bits; de los cuales se retienen los T bits más significativos MSBTlen, siendo $T \leq b$. La elección adecuada de T depende de la seguridad esperada del MAC y del número de bloques N del mensaje original.

375. En la Figura B.9 (b) se ilustra el diagrama de bloques del CMAC cuando el mensaje no tenga un tamaño igual a un múltiplo exacto de b bits.

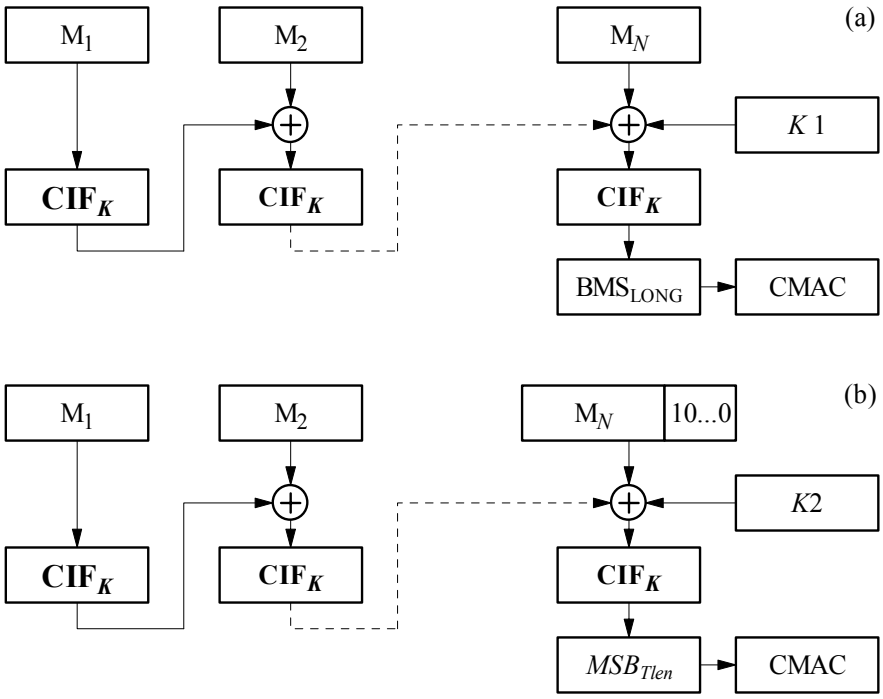


Figura B.9. Modo de Autenticación de Mensaje CMAC. (a) El mensaje tiene un tamaño igual a un múltiplo de *b* bits. (b) El mensaje tiene un tamaño no múltiplo de *b* bits.

10.1. MODO DE AUTENTICACIÓN Y CONFIDENCIALIDAD (CCM) DEL AES

376. Se denomina modo de «Autenticación y Confidencialidad» (CCM) (Counter with Cipher Block Chaining-Message Authentication Code, o Counter with CBC-MAC, CCM) del AES ([NIST, SP800-38C]) al uso conjunto de los modos CBC-MAC y CTR. Ambos se aplican al mismo mensaje, el primero para autenticar mediante un MAC y el segundo para cifrar. En los dos procesos se usa la misma clave.

377. El CBC-MAC es un MAC simple, similar al CMAC, pero en el que no existen las claves K1 ni K2.

378. Nótese que solamente está aprobado para algoritmos de cifrado en bloque con 128 bits de longitud de bloque, como el AES, no pudiéndose aplicar al DEA ni al TDEA.

10.2. MODO DE CONTADOR DE GALOIS (GCM) DEL AES

379. Otro modo de uso del AES es el «Contador de Galois» (Galois Counter Mode, GCM). Combina el modo de contador con un nuevo esquema de autenticación con operaciones en un cuerpo de Galois. La ventaja es que las operaciones de autenticación en el cuerpo de Galois se pueden realizar fácilmente en paralelo, gracias a que los bloques se procesan por separado y luego se hace una operación de combinación, consiguiendo un rendimiento mucho mayor en las operaciones de autenticación que las que se efectuaban mediante cifrados encadenados.

380. Su definición data finales del año 2007 y se recoge en la publicación [NIST, SP800-38D].

381. Una propiedad interesante es que se puede autenticar un mensaje entero y cifrar solamente parte de él, esto es muy práctico cuando el mensaje a transmitir de forma secreta lleva cabeceras que no se pueden cifrar; como direcciones IP, puertos, numero de protocolo y otros, a estos datos se les denomina «Datos de Autenticación Auxiliares» (en inglés, Additional Authenticated Data, AAD).

382. Una consecuencia de que la autenticación se haga en paralelo es que cuando se disponga de la autenticación de un mensaje y se cambie parte de él, para autenticarlo de nuevo no resulta imprescindible volver a procesar todos los bloques, sino solamente los afectados por la modificación, para después combinarlos de nuevo.

383. Su uso para cifrar y autenticar se denomina GCM, mientras que el uso para autenticar, exclusivamente, se denomina «Código de Autenticación de Mensaje de Galois» (en inglés, Galois Message Authentication Code, GMAC).

10.3. MODO DE CONFIDENCIALIDAD DEL AES EN MEDIOS DE ALMACENAMIENTO (XTS-AES)

384. El algoritmo XTS-AES es un modo de uso de confidencialidad del AES, para utilizarse en medios de almacenamiento de datos, sin aportar autenticación. Está especialmente diseñado para cifrar con AES los datos de un disco duro en el que el índice que indica el sector del disco forma parte del proceso de cifrado, también se aplica al cifrado de las memorias flash.

385. Su descripción completa puede encontrarse en [NIST, SP800-38E] de enero de 2010.

386. La sigla XTS es el acrónimo de XEX Tweakable Block Cipher with Ciphertext Stealing; y a su vez la sigla XES es el acrónimo de XOR Encrypt XOR.

11. SEGURIDAD DEL AES

387. Hasta el momento de redacción de esta guía, no ha aparecido ningún ataque que comprometa la seguridad del AES. Se ha de considerar por tanto que el AES es un cifrador seguro para cualquier tamaño de clave.

388. El ataque más simple contra un algoritmo de cifrado consiste en la prueba exhaustiva de claves, que con los medios informáticos actuales resulta absolutamente imposible contra el AES. Si en unas cuantas décadas de investigación se llegase a desarrollar un ordenador cuántico —en lo que se trabaja con ahínco—, la seguridad de las claves de 128 y 192 bits quedaría comprometida, pero la seguridad de las claves de 256 bits seguiría intacta. En previsión de esta remota posibilidad, los archivos cifrados que deban de mantenerse seguros durante varias décadas deberían cifrarse con AES de 256 bits.

389. El método más común de ataque contra un cifrador en bloque consiste en ataques sobre versiones del cifrador con un número menor de vueltas del establecido. El AES tiene 10 vueltas para claves de 128 bits, 12 vueltas para claves de 192 bits, y 14 vueltas para claves de 256 bits. Los mejores ataques conocidos ([Ferguson et al., 2000]) son sobre versiones reducidas a 7, 8 y 9 vueltas respectivamente, para los tamaños de clave anteriores. Estos ataques no son efectivos contra el AES de tamaño estándar.

390. Recientemente se ha publicado un ataque ([Biryukov et al. 2009]) contra el AES de 256 bits consistente en el uso de claves afines —claves que comparten una cantidad de bits idénticos muy superior a la mitad de su longitud— con una versión del AES reducida a 11 vueltas. Este ataque es puramente académico, e inútil en el mundo real, además de no ser aplicable al AES de 128 bits.

391. Otros ataques posibles contra el AES son los denominados de canal lateral. Estos ataques se pueden realizar contra cualquier algoritmo de cifrado y consisten en violar físicamente las implementaciones en microcircuito de un algoritmo de cifrado cualquiera, que presente defectos de medidas de seguridad contra intrusión física. Estos ataques permitirían, por ejemplo, leer directamente las claves o el mensaje en claro. Pero tales debilidades no conciernen al algoritmo sino a implementaciones defectuosas.

12. CUADRO RESUMEN

Cuadro A.2. Resumen de longitudes de claves de criptosistemas de cifrado en bloque para el ENS

Cifrado simétrico: TDEA y AES	**Nivel Bajo**	**Nivel Medio**	**Nivel Alto**
2.5. Protección de la confidencialidad	No se aplica	Permitido Claves ≥ 112 bits	Permitido Claves ≥ 128 bits
2.6. Protección de la autenticidad y de la integridad	No se aplica	Permitido Claves ≥ 112 bits	Permitido Claves ≥ 128 bits
2.7. Cifrado de la información	No se aplica	No se aplica	Permitido Claves ≥ 128 bits
2.8. Protección de las claves criptográficas	Permitido Claves ≥ 112 bits	Permitido Claves ≥ 128 bits	Permitido Claves ≥ 128 bits

ANEXO B. CRIPTOGRAFÍA DE CLAVE ASIMÉTRICA BASADA EN LA FACTORIZACIÓN

1. ACUERDO DE CLAVE DE DIFFIE-HELLMAN: DHKA

392. La criptografía de clave pública tiene su origen en la aparición del artículo [Diffie and Hellman, 1976], donde se describe un protocolo que permite intercambiar de modo seguro informaciones secretas entre dos personas a través de un canal inseguro.

393. El «establecimiento de clave» es el proceso por el cual dos (o más) partes establecen una clave secreta compartida, llamada «clave de sesión». Esta clave es usada posteriormente para alcanzar algún objetivo criptográfico, tal como la privacidad en la transmisión.

394. Antes de explicar el «Acuerdo de Clave Diffie-Hellman» (en inglés, Diffie-Hellman Key Agreement, DH o DHKA), recordemos que un grupo G es un conjunto provisto de una operación asociativa, con elemento neutro y elemento inverso; usando notación multiplicativa, esto significa: $(x \cdot y) \cdot z = x \cdot (y \cdot z)$, $1 \cdot x = x$, $x \cdot 1 = x$, $x^{-1} \cdot x = 1$, $x \cdot x^{-1} = 1$, para todo sistema de elementos x, y, z de G.

395. El subgrupo generado por un elemento g en un grupo finito G es el conjunto de sus potencias: g^i, i = 0, 1, 2, 3, …

396. Como G es finito, existe un entero mínimo n tal que: $g^n = 1$. Dicho número es el «orden» de g. El subgrupo generado por g está entonces formado por las n primeras potencias, esto es, $1 = g^0, g = g^1, g^2, g^3, \ldots, g^{n-1}$.

397. El ejemplo básico que se considera es el del grupo multiplicativo de los números enteros módulo un primo p, es decir, el conjunto de los posibles restos que resultan al dividir un entero entre p, esto es, $Z_p = \{0, 1, \ldots, p-1\}$. Sin embargo, el acuerdo de clave de Diffie-Hellman es válido en cualquier grupo finito.

398. Dos usuarios A y B que quieran intercambiar una información común mediante el DHKA proceden como sigue:

- Seleccionan públicamente un grupo finito *G* y eligen un elemento *g* de *G* que genere un subgrupo de orden suficientemente grande (en la práctica, se supone que *g* genera el propio grupo *G*).
- El usuario *A* (respectivamente *B*) genera un número entero aleatorio *a* (resp. *b*), calcula la potencia g^a (resp. g^b) en el grupo *G* y envía el resultado a *B* (resp. *A*).
- *A* recibe g^b y calcula $(g^b)^a = g^{ab}$. Análogamente, *B* recibe g^a y calcula $(g^a)^b = g^{ab}$.
- Por tanto, *A* y *B* poseen un elemento común g^{ab} del grupo *G*, que es secreto, y puede ser usado como clave secreta compartida por *A* y *B*.

399. Se han propuesto varias clases de grupos específicos para aplicar el protocolo de acuerdo de clave Diffie-Hellman. Entre ellos destacan los siguientes:

- Los grupos multiplicativos de cuerpos finitos grandes (cuerpos primos en [Diffie and Hellman, 1976] o extensiones finitas de los mismos),
- Los grupos multiplicativos de clases de restos de números enteros módulo un número compuesto ([McCurley, 1988]),
- Las curvas elípticas sobre cuerpos finitos ([Miller, 1986], [Koblitz, 1987]),

- El jacobiano de una curva hiperelíptica sobre un cuerpo finito ([Koblitz, 1989]),
- El grupo de clases de cuerpos cuadráticos imaginarios ([Buchmann and Williams, 1988]).

400. El «Problema de Diffie-Helman» (en inglés, Diffie-Hellman Problem, DHP) con respecto al generador g del grupo G, consiste en calcular gab conociendo g, ga y gb.

401. Este problema está íntimamente relacionado con el «Problema del Logaritmo Discreto» (en inglés, Discrete Logarithm Problem, DLP). Este problema consiste en calcular a conociendo un generador g del grupo G y ga. De hecho, si el problema del logaritmo discreto puede resolverse, entonces obviamente el DHP también; pero el recíproco no es del todo exacto. Ahora bien, se conocen condiciones generales sobre el grupo G (que básicamente afectan el tamaño de los factores primos del orden del grupo) bajo las cuales el DHP y el DLP son equivalentes en tiempo polinómico probabilístico; para los detalles de esta teoría pueden consultarse [Maurer, 1994], [Maurer and Wolf, 1996], [Maurer and Wolf, 1999], [Maurer and Wolf, 2000].

402. Por otra parte, el «Problema de Decisión de Diffie-Hellman» (en inglés Diffie-Hellman Decisional Problem, DHDP) consiste en lo siguiente: sea g un generador de G y supóngase que ga, gb y gc se han elegido independiente y aleatoriamente en G según una distribución uniforme. Dados los triples (ga, gb, gab) y (ga, gb, gc) en orden aleatorio, se trata de decidir con probabilidad sustancialmente mayor que ½ cuál de los triples es el correcto DHKA (ga, gb, gab).

403. El DHDP parece ser más fácil de resolver que el DHP en general. El DHP puede ser difícil si el orden del grupo contiene al menos un factor primo grande, mientras que el DHDP puede ser difícil sólo si el orden de G no tiene ningún factor primo pequeño.

404. Un protocolo de establecimiento de clave se dice que suministra «autenticación de clave» si un usuario está seguro de que ningún otro usuario, aparte del destinatario legítimo, puede conocer el valor de la clave de sesión.

405. Un protocolo de establecimiento de clave se dice que suministra «confirmación de clave» si un usuario está seguro de que el destinatario legítimo tiene en realidad posesión de una particular clave de sesión.

406. Si se proporciona la autenticación de clave o la confirmación de clave a ambos usuarios implicados en el protocolo, entonces la autenticación se dice que es «mutua»; mientras que si se proporciona a un solo usuario, la autenticación es «unilateral».

407. En términos generales, hay dos clases de protocolos de establecimiento de clave: protocolos de «transferencia de clave» en los que se crea una clave por un usuario y se transmite de modo seguro a un segundo usuario, y protocolos de «acuerdo de clave» en los cuales ambas partes contribuyen información que establece conjuntamente la clave secreta compartida. Para más detalles, véanse [Rueppel and Oorschot, 1994] y el capítulo 12 de [Menezes et al., 1997].

408. Existen tres protocolos de acuerdo de clave definidos en [Matsumoto et al., 1986] y modificados en [Menezes et al., 1995] para evitar ciertos ataques, conocidos como MTI/A0, MTI/B0 y MTI/C0, que son variantes del protocolo original de acuerdo de clave de Diffie-Hellman.

409. También existe un protocolo para dos usuarios A y B que establece una clave de sesión k transmitiendo solamente un mensaje de A a B; véanse [Nyberg and Rueppel, 1995] y [Menezes et al., 1995].

410. En la sección 4 de [Menezes et al., 1995] se presentan dos protocolos de acuerdo de clave. Difieren de los anteriores en que son simétricos, no interactivos y no requieren ni funciones resumen (hash) ni cifrado.

411. El primero es esencialmente un acuerdo de clave de Diffie-Hellman y tiene la desventaja de no ofrecer «secreto avanzado perfecto» en el sentido de [Diffie et al., 1992]; esto es, si un adversario conoce la clave privada de largo plazo de un usuario A, entonces puede deducir todas las claves pasadas de sesión de A.

412. El segundo protocolo resuelve estas deficiencias y puede verse como una modificación del protocolo MTI/A0.

413. Por último, en la sección 5 de [Menezes et al., 1995] se describe otro protocolo de acuerdo de clave cuyo propósito es establecer una clave de sesión entre dos usuarios A y B teniendo que transmitir sólo un mensaje de A a B, siempre y cuando se suponga que A tiene a priori una copia auténtica de la clave pública de B. Este protocolo puede verse como una extensión de la variante de ElGamal del cambio de clave de Diffie-Hellman.

2. CRIPTOGRAFÍA DE CLAVE ASIMÉTRICA

414. La «criptografía asimétrica» o «criptografía de clave pública» nació para resolver algunos de los inconvenientes de la criptografía clave secreta. En un criptosistema de clave pública cada usuario elige y maneja, en realidad, dos claves: una es la clave pública que el usuario pone a disposición del resto de usuarios del sistema; otra es la clave privada, sólo conocida por él. Dado que la clave de cifrado y la de descifrado son diferentes, estos criptosistemas también se denominan de «clave asimétrica», a diferencia de los criptosistemas simétricos o de clave secreta, en los que se emplea la misma clave tanto para cifrar como para descifrar información.

2.1. DEFINICIONES

415. Dado un criptosistema de clave pública, si un usuario A quiere enviar al usuario B un mensaje cifrado, tiene que seguir los pasos siguientes:

- *A* selecciona en el directorio de claves públicas la clave pública correspondiente a *B*.
- *A* cifra su mensaje aplicando la clave de *B* y se lo envía.

416. Los pasos del usuario B son:

- *B* recibe el mensaje cifrado.
- *B* descifra el mensaje aplicando su clave privada, sólo conocida por él.

417. El sistema basa su seguridad en la dificultad que representa para cualquier usuario distinto de B —desprovisto, por lo tanto, de la clave privada— descifrar el mensaje. Cuanto mayor sea esa dificultad, más seguro se puede considerar el sistema y más difícil será el trabajo del criptoanalista o atacante.

418. De forma más precisa, los criptosistemas de clave asimétrica pueden definirse a partir de una función invertible entre el conjunto de todos los mensajes en claro, M, que se pueden intercambiar dos usuarios y el conjunto de todos los textos cifrados, C, f: M→C, de modo que sea «fácil» calcular el cifrado de un mensaje, f(m) = c, mientras que sea «difícil»

computar el descifrado del mismo, f□¹(c) = m, a no ser que se posea de determinada información específica, como por ejemplo, la clave de descifrado.

419. Los términos «fácil» y «difícil» deben entenderse como que dicho cómputo requiere poco o mucho tiempo de computación, respectivamente.

420. Una función que verifique las condiciones anteriores se denomina «Función Unidireccional con Trampilla» (en inglés, Trapdoor One-Way Function, TOWF). La información adicional extra es lo que se conoce como «trampilla».

421. En general, un «Criptosistema de Clave Pública» se puede definir como una familia de funciones unidireccionales con trampilla, {fk}, de modo que para cada clave k∈ K se debe poder describir un algoritmo eficiente que permita calcular fk, pero de modo que sea intratable, computacionalmente hablando, determinar tanto la clave k como $(fk)^{-1}(\cdot)$ sin conocer el valor de k.

422. Hay dos funciones que, tradicionalmente, se emplean como funciones unidireccionales con trampilla. La primera de ellas es el producto de números enteros, cuya inversa es la factorización del número obtenido, y la segunda es la exponenciación discreta, cuya inversa es el logaritmo discreto. Las dos funciones son fáciles de computar, mientras que no lo son sus inversas. Es decir, dado un número compuesto n, es difícil determinar su descomposición en factores primos y, por otra parte, dados a e y, es difícil calcular x de modo que ax = y, en un grupo cíclico finito.

423. Estas dos funciones permiten implementar los dos criptosistemas de clave asimétrica más extendidos en la actualidad, el criptosistemas RSA y el de ElGamal (y sus derivados).

424. Para simplificar la notación, se considera que e es la clave pública de un usuario A y que d es su clave privada. Además, la función de cifrado utilizando una clave genérica k es Ck, mientras que la función de descifrado con la misma clave sería Dk. Sea cual sea la función unidireccional con trampilla que se emplee, un criptosistema de clave pública para cifrar y descifrar un mensaje m sigue los siguientes pasos:

- El usuario *B* desea enviar un mensaje cifrado al usuario *A*, para lo que obtiene su clave pública, *e*.
- *B* cifra el mensaje *m* que quiere transmitir haciendo uso de esta clave pública: $C_e(m) = c$.
- *B* envía a *A* el mensaje cifrado: *c*.

425. Para que A descifre el criptograma c sólo tiene que utilizar su clave privada *d* y recuperar el mensaje original *m*:

$$D_d(c) = D_d(C_e(m)) = m.$$

426. Los criptosistemas de clave pública resuelven algunos de los principales inconvenientes de los sistemas de clave secreta, pero plantean, sin embargo, otros problemas. Uno muy inmediato está relacionado con la autentificación de la clave pública. En efecto, el remitente debe estar seguro de que está cifrando con la clave pública «auténtica» del destinatario, pues esa clave ha sido quizá publicada u obtenida a través de un canal inseguro, lo que permite un ataque activo, es decir, un ataque en el que se puede proceder como sigue:

- *A* quiere enviar a *B* un mensaje, para lo cual solicita al directorio la clave pública *k* de *B*.

- Un criptoanalista o atacante al sistema *C* interfiere la comunicación y envía maliciosamente a *A* una clave pública errónea, *k'*, de *B*.
- *A* envía un mensaje a *B* usando *k'*.
- *C* lo intercepta y lo descifra usando su propia clave privada.
- *C* lo cifra de nuevo con la clave pública *k* de *B* y lo envía a *B*.

427. Este problema se resuelve en la práctica con un protocolo de certificado de clave pública que se basa en la capacidad de firma disponible en los criptosistemas de clave pública y en la autoridad de un «Tercero de Confianza» (en inglés, Trusted Third Party, TTP). Pueden verse más detalles, por ejemplo, en [Menezes et al., 1997].

428. El gran desarrollo de la criptografía ha hecho que ésta no se limite únicamente a los procesos de cifrado y descifrado de datos. Hoy en día, no puede entenderse esta ciencia sin una nueva componente como es «Autenticación» o «Autentificación». En la mayoría de las comunicaciones (ya sean electrónicas o no) es imprescindible la garantía de que quien dice ser el remitente de una información sea realmente quien dice serlo. Esta necesidad queda garantizada mediante los procesos de autenticación que proporciona la criptografía de clave asimétrica, a través de los procedimientos de firma digital.

429. Por otra parte, mediante el uso de determinadas herramientas criptográficas, es posible elaborar esquemas y protocolos que permitan algunas de las actuales aplicaciones a través de Internet y comunicación con o sin cable, como son el pago con dinero electrónico, votaciones en redes de ordenadores, demostrar que se posee determinada información sin revelar parte alguna de la misma, distribuir un secreto entre varias personas de modo que sea necesario que varias de estas personas se pongan de acuerdo para recuperar dicho secreto, etc.

2.2. PROTOCOLO DE ENVOLTURA DIGITAL

430. Debido a que los criptosistemas de clave secreta son más rápidos que los de clave pública a la hora de cifrar y descifrar mensajes (los primeros pueden llegar a ser 1.000 veces más rápidos, en hardware, que los segundos), se suele recurrir a un protocolo que usa ambos tipos de criptosistemas.

431. Este protocolo se conoce como «protocolo de envoltura digital» y permite que dos usuarios A y B se intercambien mensajes secretos utilizando un criptosistema de clave secreta, (Sk, Tk), y otro de clave pública, (Ee, Dd), sobre los que se han puesto de acuerdo de antemano. El protocolo para cifrar mediante la envoltura digital consiste en lo siguiente:

- El usuario *B* desea enviar un mensaje cifrado al usuario *A*, para lo que obtiene su clave pública, *e*.
- *B* elige aleatoriamente una clave *k* para utilizar con el criptosistema de clave secreta, que ambos han acordado de antemano.
- *B* cifra el mensaje *m* a enviar a *A* mediante el criptosistema de clave secreta elegido, usando la clave *k*: $S_k(m) = M$.
- *B* cifra la clave secreta *k* mediante el criptosistema de clave pública elegido, usando la clave pública de *A*: $E_e(k) = K$.
- *B* envía a *A* el par formado por *M* y por *K*.

432. El protocolo de descifrado sigue los siguientes pasos:

- *A* obtiene la clave secreta *k*, descifrando *K* mediante su clave privada correspondiente al criptosistema de clave pública:

$$D_d(K) = D_d(E_e(k)) = k.$$

- *A* descifra el mensaje mediante el criptosistema de clave secreta haciendo uso de la clave que acaba de obtener:

$$T_k(m) = T_k(S_k(m)) = m.$$

3. CRIPTOSISTEMA RSA

433. El criptosistema RSA fue introducido por Rivest, Shamir y Adleman en [Rivest et al., 1978] (ver también [Durán et al., 2005], [Menezes et al., 1997]) y actualmente es el criptosistema de clave pública más conocido y más ampliamente usado. La seguridad de este criptosistema se basa en la dificultad que supone resolver el «Problema de la Factorización Entera» (en inglés, Integer Factorization Problem, IFP). Dicho problema puede plantearse en los siguientes términos: dado un número entero compuesto (no primo), determinar los factores primos que lo dividen, es decir, calcular su factorización como producto de números primos elevados a potencias.

434. Como todo criptosistema de clave pública, el protocolo del criptosistema RSA tiene tres partes:

- Generación de claves,
- Cifrado de mensajes,
- Descifrado de mensajes.

3.1. GENERACIÓN DE CLAVES

435. Cada usuario A produce su clave pública y su correspondiente clave privada, procediendo como sigue:

- El usuario *A* genera dos primos «grandes»[(2)] *p* y *q*, usando alguno de los algoritmos existentes para ello, por ejemplo, véanse [Menezes et al., 1997], [Pollard, 1974], [Rivest et al., 1978], [Salomaa, 1990], entre otros.
- El usuario *A* calcula el producto $n = p \cdot q$ y su indicador de Euler, esto es, $\varphi(n) = (p-1)(q-1)$.
- El usuario *A* elige también aleatoriamente un entero *e* que verifique las dos siguientes propiedades:
- El entero *e* debe ser mayor que la unidad y menor que el indicador de Euler de *n*, es decir: $1 < e < \varphi(n)$,
- Los enteros *e* y $\varphi(n)$ deben ser primos entre sí, es decir, no deben poseer factores comunes, o equivalentemente, su máximo común divisor debe ser igual a la unidad: $\text{mcd}(e,\varphi(n)) = 1$.

(2) La palabra «grandes» debe entenderse en el sentido de que en cada momento los métodos de factorización no permitan factorizar el producto $p \cdot q$ (véase la sección 3.4 del ANEXO B para más detalles). En la actualidad, se recomienda que los primos *p* y *q* tengan alrededor de 512 bits cada uno.

- Usando el Algoritmo de Euclides Extendido (puede consultarse en la sección 2.107 de la página 67 de [Menezes et al., 1997]), *A* calcula el inverso de *e* módulo φ(*n*); esto es, *A* calcula el único entero *d* que verifica las dos siguientes propiedades:
- El entero *d* debe ser mayor que la unidad y menor que el indicador de Euler de *n*, es decir: $1 < d < \varphi(n)$.
- El resto de efectuar la división entera del producto $e \cdot d$ entre φ(*n*) ha de ser igual a la unidad, lo cual se escribe: $e \cdot d = 1 \pmod{\varphi(n)}$.
- Finalmente, *A* publica su clave pública (*n*,*e*), mientras que mantiene en secreto su clave privada, que es *d*.

436. Los valores de p, q y φ(n) también deben mantenerse en secreto, aunque no forman parte propiamente de la clave privada.

437. Los exponentes e y d reciben el nombre de «exponente de cifrado» y «exponente de descifrado», respectivamente, y n recibe el nombre de «módulo» del criptosistema. Un «mensaje en claro» es un número entero comprendido entre 1 y n, o equivalentemente, es un entero módulo n.

3.2. CIFRADO DE MENSAJES

438. Si el usuario B quiere enviar un mensaje cifrado al usuario A, procede como sigue:

- En primer lugar debe obtener la clave pública de *A*, que ha sido previamente publicada, es decir, (*n*, *e*),
- Luego calcula el mensaje cifrado *c* mediante la operación de elevar el mensaje *m* al exponente de cifrado *e* y luego tomar el resto del resultado de dividir esta potencia entre *n*, esto es, el mensaje cifrado se obtiene calculando:

$$c = m^e \pmod{n}.$$

3.3. DESCIFRADO DE MENSAJES

439. Si el usuario A quiere recuperar el texto en claro que le ha enviado cifrado el usuario B, hace lo siguiente:

440. Usa su clave privada d para elevar el mensaje cifrado c a dicha clave y dividir el resultado entre el módulo n, quedándose con el resto de tal división; esto es, cd (mod n).

441. De este modo se recupera el mensaje m; esto es, se demuestra (la demostración puede verse en la sección 8.2.1 de [Menezes et al., 1997] o bien en el Lemma 4.1 de [Salomaa, 1990]) que el resultado anterior coincide con el mensaje en claro, en otras palabras se verifica:

$$m = c^d \pmod{n}.$$

3.4. SEGURIDAD

442. Los ataques a la seguridad se dividen en las siguientes categorías: relativos a la factorización del módulo, exponentes de cifrado pequeños, exponentes de descifrado pequeños, ataques cíclicos y sus generalizaciones, y mensajes inocultables.

443. ATAQUES RELATIVOS A LA FACTORIZACIÓN DEL MÓDULO. El primer ataque contra el criptosistema RSA consiste en tratar de factorizar el módulo n del criptosistema. Este tipo de ataque ha dado lugar a un gran número de publicaciones.

444. El Teorema Fundamental de la Aritmética establece que todo número entero puede factorizarse como producto de números primos, cada uno con una cierta multiplicidad, y que los factores primos y sus multiplicidades son únicos.

445. Ahora bien, obtener la factorización de un número entero es computacionalmente difícil en general si el tamaño del número es grande.

446. Los mejores algoritmos para resolver el problema de la factorización de números enteros tienen un tiempo de ejecución subexponencial; véanse por ejemplo: [Bach and Shallit, 1996], [Boneh, 1999], [Buhler et al., 1993], [Kleinjung et al., 2010], [Lenstra, 1993], [Lenstra et al., 1993], [Pollard, 1974], [Sarkar and Maitra, 2009a].

447. Por tanto, este problema se considera muy difícil computacionalmente en la actualidad; pero existen ciertas clases especiales de números enteros que sí pueden factorizarse eficientemente por medio de algoritmos específicos; véanse, por ejemplo, los artículos [Pollard, 1974] y [Williams, 1982].

448. Si un criptoanalista puede factorizar el módulo n y obtener sus factores primos p y q, entonces la seguridad del sistema queda totalmente comprometida, porque el criptoanalista puede calcular directamente el indicador de Euler $\varphi(n) = (p-1)(q-1)$ y, puesto que el exponente de cifrado e es público, también puede calcular su inverso d módulo $\varphi(n)$, empleando el Algoritmo de Euclides Extendido, de modo que el criptoanalista puede conocer la clave privada d del usuario.

449. De hecho, se sabe (véanse [Rivest et al., 1978], [Miller, 1976], la sección 8.2.2 de [Menezes et al., 1997] y [Coron and May, 2007] para el resultado definitivo) que el problema de computar el exponente de descifrado d conociendo la clave pública (n, e), y el problema de factorizar n, son computacionalmente equivalentes.

450. Para evitar el ataque por factorización deben seguirse las siguientes recomendaciones:

- Los primos p y q deben ser de la misma longitud, pero no deben estar demasiado cercanos; véase [de Weger, 2002], [Menezes et al., 1997] y [Salomaa, 1990] para más detalles. De modo preciso, si k es el número de bits de n y se verifica: $\log_2|q-p| < (k+5)/4$, entonces existe un algoritmo eficiente que permite factorizar n; véase [Khadir, 2008].
- El máximo común divisor de $p-1$ y $q-1$ debe ser pequeño, típicamente: $\text{mcd}(p-1, q-1) = 2$. (Obsérvese que el máximo común divisor de $p-1$ y $q-1$ ha de ser por lo menos 2, pues $p-1$ y $q-1$ son pares.) Esta condición se consigue si p y q son primos 1-seguros. Se dice que un número primo p es «1-seguro» si el entero $(p-1)/2$ también es primo.
- Los primos p y q deben ser robustos. Un primo impar p se dice que es «robusto» si verifica las tres siguientes propiedades:
- $p-1$ contiene un factor primo grande r,
- $p+1$ también contiene un factor primo grande,
- $r-1$ también contiene un factor primo grande.

451. Las dos primeras propiedades previenen de los ataques basados en los algoritmos de Pollard ([Pollard, 1974]) y Williams ([Williams, 1982]), mientras que la tercera está destinada a evitar los ataques cíclicos, que se describirán más adelante.

452. ATAQUES A EXPONENTES DE CIFRADO PEQUEÑOS. En general, para facilitar y hacer más eficiente el proceso de cifrado del criptosistema RSA se eligen exponentes e de cifrado pequeños. Pero bajo ciertas circunstancias esta práctica puede comprometer potencialmente la seguridad del sistema. El problema surge cuando se envían e mensajes cifrados del mismo mensaje en claro m con el mismo exponente de cifrado e (pequeño) a usuarios cuyos módulos ni son primos entre sí dos a dos, o sea, mcd(ni, nj) = 1 para i < j, pues en este caso, el algoritmo de Gauss (véase la sección 2.121 de [Menezes et al., 1997]; véanse también [Hastad, 1988] y [Coppersmith et al., 1996]) permite recuperar el mensaje m.

453. Este ataque permite obtener de nuevo el texto en claro bajo ciertas circunstancias, como hemos explicado, pero no compromete la seguridad global del sistema, ya que no permite averiguar el exponente de descifrado d de ningún usuario ni factorizar su módulo.

454. ATAQUES A EXPONENTES DE DESCIFRADO PEQUEÑOS. Estos ataques explotan el hecho de que el tamaño de d sea menor que una cierta cota superior que depende del ataque considerado, lo cual supone una debilidad del sistema.

455. Los más importantes son los siguientes:

456. El ataque de la fracción continua de Wiener, desarrollado en [Wiener, 1990]. Usando el desarrollo en fracción continua de e/n, Wiener diseñó un algoritmo que permite recuperar la clave privada si ésta verifica: $d < n^{0.25}$; esto es, si el número de bits de d es inferior a la cuarta parte del número de bits de n. Además, en este caso, el algoritmo que calcula d se ejecuta en tiempo polinómico.

457. Extensiones y refinamientos de este resultado pueden consultarse en [Verheul and van Tilborg, 1997], [Chen et al., 2004], [Bleichenbacher and May, 2006], [Jochemsz and May, 2007], [Djuella, 2009] y [Luo et al., 2009].

458. El ataque de Boneh y Durfee desarrollado en [Boneh and Durfee, 1999], [Boneh and Durfee, 2000], [Durfee and Nguyen, 2000] (véase también [Boneh, 1999]), que se basa en el «Algoritmo de Lenstra-Lenstra-Lovász» (en inglés LLL Algorithm o L3 Algorithm) y permite calcular d eficientemente siempre y cuando $d < n^{0.292}$, lo cual mejora el resultado de Wiener.

459. El ataque de Blömer y May ha sido desarrollado básicamente en [Blömer and May, 2001], [May, 2002] y [Maitra and Sarkar, 2008]. Este ataque es un refinamiento muy técnico del de Boneh-Durfee.

460. Recientemente, se han efectuado algunos ataques al criptosistema RSA suponiendo que se conocen, o bien los bits más significativos o bien los menos significativos de la clave privada d; véanse [Sarkar and Maitra, 2009a], [Sarkar and Maitra, 2009b]. Otros ataques suponen la existencia de dos exponentes de descifrado (por ejemplo, [Sarkar and Maitra, 2010]). Estos ataques pueden ser interesantes bajo ciertas circunstancias particulares, pero no son de propósito general.

461. En resumen: todos los ataques anteriores consiguen recuperar la clave privada d a partir de la clave pública (n, e) exclusivamente, suponiendo que el tamaño de d sea suficientemente pequeño. Por tanto, en tales ataques la seguridad del criptosistema queda completamente comprometida.

462. Sin embargo, todos estos ataques pueden evitarse tomando d suficientemente grande, según indique la cota superior para d de cada uno de los ataques mencionados.

463. ATAQUES CÍCLICOS Y SUS GENERALIZACIONES. Si c = me (mod n) es un texto cifrado, entonces se demuestra que existe un cierto entero k tal que verifica:

464. ce^k = c (mod n)(3).

465. Pueden consultarse [Cohen, 1993], [Menezes et al., 1997], [Gysin and Seberry, 1999]. En este caso, se tiene:

466. ce^(k–1) = m (mod n).

467. Por tanto, si se conoce k se puede recuperar el mensaje en claro m.

468. Los ataques cíclicos permiten recuperar el texto en claro, pero no rompen el criptosistema. Ahora bien, los ataques cíclicos generalizados sí pueden comprometer la seguridad del criptosistema RSA, aunque, dada la dificultad de la factorización de números enteros, no suponen una amenaza real, siempre y cuando los factores p y q se elijan cumpliendo las recomendaciones explicadas.

469. MENSAJES INOCULTABLES. Este ataque tiene éxito siempre y cuando la potencia e-ésima del texto en claro coincida con el propio texto en claro, lo cual puede formularse con precisión como sigue: un mensaje m (0 < m < n) se dice que es «inocultable» si se verifica: me = m (mod n). Véase [Blakley and Borosh, 1979], [Menezes et al., 1997], [Chmielowiec, 2010].

470. Siempre hay por lo menos 9 mensajes inocultables pero su número es muy pequeño. Tales mensajes no pueden ser cifrados. Por otra parte, existen ejemplos de módulos RSA para los cuales todos los mensajes son inocultables. Un ejemplo famoso es el siguiente: p = 97, q = 109, n = 10573, e = 865, d = 9505.

471. Este ataque se aplica sólo a valores excepcionales del módulo n; si p y q se toman de modo aleatorio y, además, e y φ(n) son primos entre sí (es decir, mcd(e, φ(n)) = 1), entonces la probabilidad de hallar un mensaje inocultable es menor que K(ln n)2/n, siendo K una constante.

3.5. COMENTARIOS

472. En las implementaciones actuales del criptosistema RSA en tarjetas inteligentes, se usa el «Teorema Chino del Resto» (en inglés, Chinese Remainder Theorem, CRT; véase [Bach and Shallit, 1996] o [Menezes et al., 1997]) para que los cálculos sean más rápidos, especialmente el proceso de descifrado; véanse [Quisquater and Couvreur, 1982], [Boneh and Shacham, 2002], [Sun and Wu, 2005], [Sun et al., 2005], [Okeya and Takagi, 2006]).

473. En esta situación, el criptosistema RSA recibe el nombre de RSA-CRT.

474. Los procesos de cifrado y descifrado en este sistema modificado son distintos de los correspondientes procesos en el sistema RSA estándar. En RSA-CRT,

- Cada usuario A genera dos primos «grandes» p y q, y calcula los productos $n = p \cdot q$, $\varphi(n) = (p-1)(q-1)$, como antes.
- Después, toma dos enteros aleatorios d_p, d_q que verifiquen las siguientes propiedades:
- El máximo común divisor de d_p y $p-1$ es la unidad, es decir, mcd(d_p, p–1) = 1.
- Análogamente, el máximo común divisor de d_q y $q-1$ es la unidad, o sea, mcd(d_q, q–1) = 1.
- Los enteros d_p y d_q tienen la misma paridad, es decir, $d_p = d_q$ (mod 2).

(3) La expresión *e^k* representa la potencia *k*-ésima del número entero *e*, es decir, $e\text{^}k = e^k$.

- Usando el Teorema Chino del Resto se determina un entero d que tiene el mismo resto que d_p al ser dividido por p–1 y el mismo resto que d_q al ser dividido por q–1, esto es, $d = d_p \pmod{p-1}$ y $d = d_q \pmod{q-1}$, y se calcula el inverso e de d módulo $\varphi(n)$; es decir: $e \cdot d = 1 \pmod{\varphi(n)}$.

475. Para recuperar el texto en claro,

- A usa su clave privada d para calcular

$$m_p = c^{d\_p} \pmod{p} = c^d \pmod{p} \text{ y } m_q = c^{d\_q} \pmod{q} = c^d \pmod{q}^{(4)}.$$

- Usando de nuevo el Teorema Chino del Resto, determina una solución común a las dos ecuaciones siguientes: $m = m_p \pmod{p}$, $m = m_q \pmod{q}$.

476. Recientemente (por ejemplo, véase [Hinek, 2008]), se ha propuesto también una generalización del criptosistema RSA estándar, llamada «RSA multi-primo», en la cual el módulo del sistema contiene más de dos factores primos. Cifrando y descifrando módulo cada factor primo del módulo y usando el Teorema Chino del Resto en el criptosistema RSA multi-primo, el coste computacional de estos procesos se reduce considerablemente.

477. Existen también ataques al exponente de descifrado pequeño para el criptosistema RSA-CRT (pueden verse [Bleichenbacher and May, 2006], [Jochemsz and May, 2007]) y ataques al criptosistema multi-primo ([Hinek, 2008], [Sarkar and Maitra, 2009b], [Sarkar and Maitra, 2010]).

478. El mínimo común múltiplo de p–1 y q–1 recibe el nombre de «función lambda de Charmichael» y se designa por λ(n) = mcm(p–1, q–1) . El entero λ(n) puede ser usado en vez del indicador de Euler φ(n). Tiene la ventaja de poseer, en principio, un exponente de descifrado d de tamaño menor, lo cual redunda en la rapidez del proceso de descifrado. Ahora bien, si p y q se eligen «realmente» de modo aleatorio, entonces el máximo común divisor de los enteros p–1 y q–1 es pequeño, de modo que λ(n) y φ(n) tienen un tamaño muy parecido.

4. CUADRO RESUMEN

Cuadro C.1. Resumen de longitudes de claves del criptosistema RSA para el ENS

Cifrado simétrico: RSA	**Nivel Bajo**	**Nivel Medio**	**Nivel Alto**
2.5. Protección de la confidencialidad	No se aplica	Permitido Claves ≥ 2048 bits	Permitido Claves ≥ 2048 bits
2.6. Protección de la autenticidad y de la integridad	No se aplica	Permitido Claves ≥ 2048 bits	Permitido Claves ≥ 2048 bits
2.7. Cifrado de la información	No se aplica	No se aplica	Permitido Claves ≥ 2048 bits
2.8. Protección de las claves criptográficas	Permitido Claves ≥ 2048 bits	Permitido Claves ≥ 2048 bits	Permitido Claves ≥ 2048 bits

(4) Las expresiones d_p y d_q denotan subíndices, es decir, $d\_p = d_p$ y $d\_q = d_q$.

479. Criptografía de clave Asimétrica basada en el logaritmo discreto: Criptosistemas de ElGamal y de Curvas Elípticas

5. CRIPTOSISTEMA DE ELGAMAL

480. El Criptosistema de Taher ElGamal fue publicado en 1985 y es de gran relevancia hoy en día ([ElGamal, 1985], [Fúster et al., 2004], [Menezes et al., 1997]). Por sí mismo, este criptosistema no ha sido muy utilizado en implementaciones prácticas pero es importante por cuanto es la base sobre la que se fundamenta uno de los criptosistemas más prometedores: el basado en determinada clase de curvas elípticas.

481. La seguridad de criptosistema de ElGamal estriba en la dificultad computacional que supone resolver el problema del logaritmo discreto, uno de los problemas matemáticos cuya resolución requiere tiempos de computación muy elevados (conocidos como tiempo exponencial o subexponencial, ver la sección A.3 de [Durán et al., 2005]) que son considerados como problemas muy difíciles, de forma análoga a como lo es el problema de la factorización de números enteros, utilizado con el criptosistema RSA.

482. El Problema del Logaritmo Discreto (DLP), en su forma más sencilla (ver la sección 3.6 de [Menezes et al., 1997] para una descripción más completa de este problema y su expresión en grupos matemáticos concretos), consiste en calcular el logaritmo de un número en una base dada pero en un conjunto finito de números (de ahí lo de discreto), $\log_g L = x$.

483. Dicho de otro modo, se trata de determinar el exponente, x, al que hay que elevar una base dada, g, para que dicha potencia proporcione otro número conocido de antemano, L: $g^x = L$. Definido de esta manera, el problema podría parecer sencillo, pero el conjunto finito de números se elige de tal forma que calcular tal potencia requiere un tiempo de computación, al menos, subexponencial.

484. A modo de ejemplo se considera el conjunto de los posibles restos de la división entre 23, salvo el 0, denotado por $Z23^* = \{1, 2, \ldots, 21, 22\}$. La operación que se utiliza entre dos números dados del conjunto, a y b, es la de su producto módulo 23, que se representa como $a \cdot b \pmod{23}$. Como ya se sabe, esta operación consiste en multiplicar los dos números dados y luego tomar como resultado el resto de la división entera de tal producto entre 23. Así, si se consideran los números 13 y 17, por ejemplo, se tiene que

485. $13 \cdot 17 \pmod 7 = 221 \pmod{23} = (23 \cdot 9 + 14) \pmod{23} = 14$,

486. dado que 14 es el resto de la división entera de 221 entre 23. La ventaja de esta operación estriba en que al multiplicar dos números cualesquiera del conjunto, el resultado es siempre otro número del mismo conjunto.

487. En este ejemplo, se verifica, además, que todos los elementos de dicho conjunto se pueden obtener como potencias de uno de ellos, llamado generador. Así, se verifica que un generador de $Z23^*$ es el número 5. En efecto, se tiene que todas las potencias de 5 módulo 23 recorren en conjunto completo:

488. $5^1 \pmod{23} = 5$, $5^2 \pmod{23} = 2$, $5^3 \pmod{23} = 10$, …, $5^{22} \pmod{23} = 1$.

489. En esta situación, un problema particular de logaritmo discreto se puede plantear como sigue: resolver el siguiente logaritmo: $x = \log_5 21$. Se trata de calcular el valor de x de modo que se verifique: $5^x = 21 \pmod{23}$. En este caso, debido a la magnitud de los

números implicados, es inmediato que x = 13, por tanto, se puede afirmar que log5 21 = 13 en Z23*.

490. No obstante, la dificultad del DLP se pone de manifiesto cuando los números que se emplean en lugar de ser de una o dos cifras, son de 300 ó 400 cifras.

491. Volviendo al criptosistema de ElGamal, como en todo sistema de clave pública, se tienen tres fases:

- Generación de claves,
- Cifrado de mensajes,
- Descifrado de mensajes.

5.1. GENERACIÓN DE CLAVES

492. Para que el usuario A genere sus claves para este criptosistema, debe seguir los siguientes pasos:

- *A* selecciona un número primo grande *p* (por ejemplo de 1024 bits, es decir, de alrededor de 300-310 dígitos).
- *A* elige un generador, *g*, del grupo multiplicativo de las unidades de los enteros módulo *p*, es decir, del conjunto de números dado por

$$\mathbf{Z}_p^* = \{1, 2, \ldots, p-2, p-1\},$$

de tal manera que todos los elementos de $\mathbf{Z}_p^*$ se pueden obtener como potencias de elemento *g*, es decir,

$$\mathbf{Z}_p^* = \{g^i \pmod p: i = 1, 2, \ldots, p-1\}.$$

- *A* genera un número aleatorio *a*, mayor que 1 y menor que *p*–2, $1 < a < p-2$, y considera el resto de la división de la potencia de *g* elevado a *a* y luego dividido por *p*, es decir, calcula el valor de $g^a \pmod p$.
- La clave pública de *A* es el trío formado por los números (p, g, g^a); mientras que su clave privada es el número *a*.

493. En este caso, la mínima longitud recomendada de la clave pública, esto es, de p, es de 1024 bits; la misma longitud que la recomendada para el RSA. No obstante, debe recordarse que para el criptosistema de ElGamal, p es un único primo mientras que para el criptosistema RSA, n era el producto de dos primos, cada uno de ellos de longitud la mitad.

5.2. CIFRADO DE MENSAJES

494. Si el usuario B, desea cifrar un mensaje al usuario A, debe hacer lo siguiente:

- *B* obtiene la clave pública de *A*, es decir, conoce los valores de (p, g, g^a).
- *B* representa el mensaje *m* como un elemento de $\mathbf{Z}_p^*$; es decir, como un número entero dentro del intervalo entre 1 y *p*–1: $m \in [1, p-1]$.
- *B* genera un número aleatorio *x* mayor que 1 y menor que *p*–2, $1 < x < p-2$, calcula los valores de $u = g^x \pmod p$ y de $v = m \cdot (g^a)^x \pmod p$, y envía a *A* el criptograma o mensaje cifrado, que es el formato por el par $c = (u, v)$.

5.3. DESCIFRADO DE MENSAJES

495. Una vez que el destinatario, A, recibe el criptograma c = (u, v), para recuperar el mensaje original, m, lleva a cabo los siguientes pasos:

- *A* utiliza su clave privada, *a*, y calcula $k = u^{p-1-a} \pmod p$.
- *A* obtiene el mensaje original multiplicando el valor de *k* por *v*: $k \cdot v$.

496. En efecto, el producto anterior permite obtener m dado que se tiene lo siguiente:

$$k \cdot v \pmod p = u^{p-1-a} \cdot v \pmod p = g^{x \cdot (p-1-a)} \cdot m \cdot g^{a \cdot x} \pmod p$$
$$= g^{x \cdot (p-1) - x \cdot a + a \cdot x} \cdot m \pmod p = m.$$

5.4. SEGURIDAD

497. Un atacante al sistema o criptoanalista, C, podría conocer las claves públicas de A y de B, es decir, podría conocer el conjunto de números que se emplea, Zp*, y los valores de p, g, ga y gb, dado que todos estos valores son públicos.

498. Por otra parte, C podría haber interceptado la comunicación entre A y B y conocer los valores de u y de v.

499. Para romper el criptosistema, es decir, para determinar cualquiera de las claves privadas de los usuarios A o B, a o b, o para determinar el valor del parámetro secreto x que permitió cifrar el mensaje, tendría que resolver un caso del problema del logaritmo discreto, es decir, calcular a, b o x a partir de ga, gb o u = gx, lo que hoy por hoy, es un problema computacionalmente inabordable, con los tamaños recomendados para las claves.

500. Por esta razón, se dice que la seguridad del criptosistema de ElGamal es equivalente a la del logaritmo discreto.

501. Los algoritmos que calculan logaritmos discretos, logg L = x, es decir, que determinan el valor de x de modo que gx = L, se pueden clasificar de la siguiente manera. Los algoritmos que trabajan en grupos arbitrarios, como la búsqueda exhaustiva o fuerza bruta, el paso gigante-paso enano y el algoritmo rho de Pollard. Existen algoritmos que funcionan en grupos arbitrarios pero que son especialmente eficientes si el tamaño (orden) del grupo tiene factores o divisores pequeños, como por ejemplo, el algoritmo de Pohlig-Hellman, y por último, los algoritmos de cálculo del índice, que solo son eficientes en determinados grupos especiales (ver la sección 3.6 de [Menezes et al., 1997]).

502. BÚSQUEDA EXHAUSTIVA O FUERZA BRUTA. El algoritmo más elemental para determinar logaritmos discretos consiste en calcular las sucesivas potencias del generador del grupo considerado hasta obtener el valor que se busca. Sin embargo, este algoritmo requiere un tiempo de computación excesivamente elevado (de hecho el tiempo es exponencial), por lo que es ineficiente si el tamaño del conjunto (grupo) considerado es elevado, como sucede en los casos de interés en criptografía.

503. PASO GIGANTE-PASO ENANO. Este algoritmo es una mejora de la búsqueda exhaustiva. Consiste en calcular una lista de los pares formados por un índice, i, y la potencia del generador, g, elevada a dicho índice, gi, donde el índice recorre los valores de 1 hasta en número entero más cercano por defecto a la raíz cuadrada del orden n del

grupo con el que se está trabajando, sea $m = \lfloor \sqrt{n} \rfloor$. Una vez determinada la lista, se ordena por la segunda componente del par. Luego se determinan los productos de los inversos de estas segundas componentes, ya ordenadas, por el valor dado, L, cuyo logaritmo se busca, $x = \log_g L$, es decir, se determinan $L \cdot g^{-m \cdot j}$, $j = 1, \ldots, m-1$. Finalmente, se compara cada uno de estos valores calculados con las segundas componentes de la lista original ordenada. Si se encuentra una coincidencia, por ejemplo, $L \cdot g^{-m \cdot j} = g^i$, resulta que despejando de la igualdad obtenida es: $L = g^{i+m \cdot j}$, de donde se tiene el logaritmo buscado:

504. $x = \log_g L = \log g^{i+m \cdot j} = i+m \cdot j$.

505. Sin embargo, este algoritmo tampoco es eficiente dado que calcular las entradas de la tabla, ordenar la tabla y hacer la comparación requiere de un tiempo también exponencial.

506. Una actualización muy reciente de este algoritmo se debe a Cheon ([Cheon, 2010]). La modificación rebaja los tiempos de computación del logaritmo discreto cuando se trabaja sobre un grupo de orden primo p, de modo que además de los valores g y g^a, se conocen otras entradas auxiliares $(g^a)^i$, para valores $i = 1, \ldots, d$, siendo d un divisor de $p-1$.

507. ALGORITMO RHO DE POLLARD. Este algoritmo es aleatorio y requiere el mismo tiempo de computación que el del paso gigante-paso enano. Sin embargo, es más eficiente porque necesita menores recursos de memoria. Por tal motivo es preferible al anterior.

508. ALGORITMO DE POHLIG-HELLMAN. Este algoritmo es especialmente eficiente en grupos para los que se conoce la factorización de su orden y ésta tiene factores o divisores pequeños ([Pohlig and Hellman, 1978]). En este caso, se tiene en cuenta y se explota el hecho de que se conozca la factorización del orden del grupo sobre el que se trabaja. Así, se resuelve el problema del logaritmo discreto planteado para cada uno de los factores que dividen al orden y luego las soluciones particulares se unen haciendo uso del algoritmo de Gauss, que proporciona una solución al Teorema Chino del Resto, ya comentado (ver la sección 2.4.3 de [Menezes et al., 1997]).

509. ALGORITMO DE CÁLCULO DEL ÍNDICE. Este algoritmo es uno de los mejores para calcular logaritmos discretos, si bien su tiempo de computación es subexponencial. El algoritmo precisa de la selección de un subconjunto relativamente pequeño de elementos del grupo considerado (llamado base de factores), de modo que una significativa parte de los elementos del grupo puedan expresarse de forma eficiente como producto de elementos de la base de factores.

6. CRIPTOSISTEMA DE CURVAS ELÍPTICAS: ECC

510. El criptosistema de ElGamal presentado anteriormente se ha definido considerando el conjunto de los números enteros módulo un número primo p y dotado de la operación de multiplicación. La seguridad de este criptosistema se basa en la dificultad de resolver el problema del logaritmo discreto, como ya se ha indicado.

511. Sin embargo, este criptosistema puede definirse sobre otro conjunto diferente, no necesariamente sobre los enteros módulo un número dado. De hecho, hay otras alternativas propuestas en la literatura para el conjunto sobre el que llevan a cabo las operaciones, pero el más importante de todos ellos es el conjunto de los puntos de una curva elíptica con la operación de la suma de puntos (ver [Hankerson et al., 2004], [Koblitz, 1987], [Menezes, 1993], [Miller, 1986]). Este conjunto permite definir

«Criptosistemas basados en Curvas Elípticas» (en inglés, Elliptic Curve Cryptosystem, ECC).

512. La principal razón para considerar este conjunto es porque permite reducir considerablemente el tamaño de las claves, sin por ello disminuir la seguridad de los criptosistemas empleados, a pesar de que la computación requiere un conocimiento matemático más elevado.

513. Por otra parte, el protocolo de cifrado basado en curvas elípticas más extendido es el denominado «Esquema de Cifrado Integrado con Curvas Elípticas» (Elliptic Curve Integrated Encryption Scheme, ECIES), que emplea el protocolo de la envoltura digital para el intercambio de información cifrada ([ANSI, X9.63], [IEEE, 1363a], [ISOIEC, 18033-2]).

514. Con el fin de comprender los criptosistemas basados en curvas elípticas, sin profundizar excesivamente en los conceptos matemáticos implicados, se introducirán las curvas elípticas para determinados casos especiales, de más fácil comprensión. Para un análisis y estudio más profundo remitimos al lector a la amplia literatura publicada sobre este tema (ver [Blake et al., 1999], [Blake et al., 2005], [BSI, 2009], [Cohen et al., 2006], [Enge, 1999], [Silverman, 1994], [Silverman, 2009]).

6.1. CURVAS ELÍPTICAS SOBRE UN CUERPO FINITO

515. Una curva elíptica sobre un cuerpo finito es una curva dada por una ecuación como la siguiente, conocida como ecuación de Weierstrass:

$$E: y^2+a_1 \cdot x \cdot y+a_3 \cdot y = x^3+a_2 \cdot x^2+a_4 \cdot x+a_6,$$

de modo que los valores de los números a_1, a_3, a_2, a_4 y a_6 verifican determinadas condiciones y pertenecen a un conjunto de llamado «cuerpo finito» (ver [Menezes et al., 1993a]). Un ejemplo de cuerpo finito es el conjunto de los restos de la división entera entre un número primo p, con las operaciones de suma y producto cuyo orden es p:

$$\mathbf{F}_p = \mathbf{Z}_p = \{0, 1, 2, \ldots, p-2, p-1\}.$$

516. Los puntos que pertenecen a la curva son los pares de elementos (x, y) que, perteneciendo al cuerpo finito, verifican la ecuación de Weierstrass anterior.

517. Así por ejemplo, la curva elíptica de ecuación y2 = x3+11·x+20 definida sobre el cuerpo finito de 23 elementos, representado como F23 = {0, 1, …, 22}, es la que se presenta en la Figura D.1.

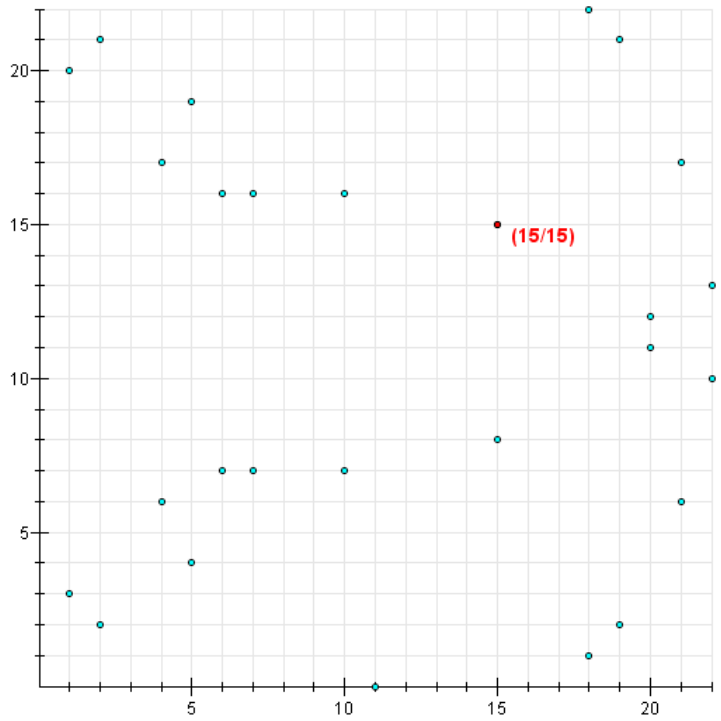


Figura D.1. Ejemplo de curva elíptica sobre un cuerpo finito.

518. Algunos de los puntos de esta curva son los siguientes:

(1, 3), (1, 20), (2, 2), (2, 21), (4, 6), (4, 17), …, (21, 10), (21, 13).

519. Como se puede ver, todos ellos verifican la ecuación que define a la curva considerando la operación definida en el conjunto F23, que consiste en multiplicar los números y luego tomar como resultado el resto de la división del producto hallado entre 23:

$$3^2 = 9 \pmod{23},$$

$$1^3+11\cdot 1+20 = 1+11+20 = 32 = 23+9 = 9 \pmod{23},$$

$$2^2 = 4 \pmod{23},$$

$$2^3+11\cdot 2+20 = 8+22+20 = 50 = 23\cdot 2+4 = 4 \pmod{23},$$

$$6^2 = 36 = 23+13 = 13 \pmod{23},$$

$$4^3+11\cdot 4+20 = 64+44+20 = 128 = 23\cdot 5+13 = 13 \pmod{23}.$$

520. Una de las principales ventajas de las curvas elípticas es que es posible definir una operación de suma de puntos que proporciona como resultado otro punto de la curva. Esta operación es la análoga a la operación de multiplicar y hacer módulo considerada en el caso del criptosistema de ElGamal.

521. En el caso de un cuerpo finito, la suma de dos puntos de una curva elíptica cuyas coordenadas sean P = (x1, y1) y Q = (x2, y2), es otro punto de la curva cuyas coordenadas son: R = (x3, y3), dadas por:

$$x_3 = z^2 - x_1 - x_2,$$

$$y_3 = z\cdot(x_1 - x_3) - y_1,$$

donde

$$z = (x_1^2 + a)/(2\cdot y_1), \text{ si } P = Q,$$

$$z = (y_2 - y_1)/(x_2 - x_1), \text{ si } P \neq Q.$$

522. Por ejemplo, la suma de los puntos cuyas coordenadas son P = (6, 16) y Q = (15, 8) de la curva elíptica anterior es el punto R = (15, 15) (ver Figura D.2).

523. En efecto, como P = (6, 16) ≠ Q = (15, 8), resulta que, modulo 23, es

$$z = (y_2 - y_1)/(x_2 - x_1) = (8 - 16)/(15 - 6) = -8/9 = 15\cdot 18 = 17,$$

$$x_3 = z^2 - x_1 - x_2 = 15^2 - 6 - 15 = 15,$$

$$y_3 = z\cdot(x_1 - x_3) - y_1 = 17(6 - 15) - 16 = 15.$$

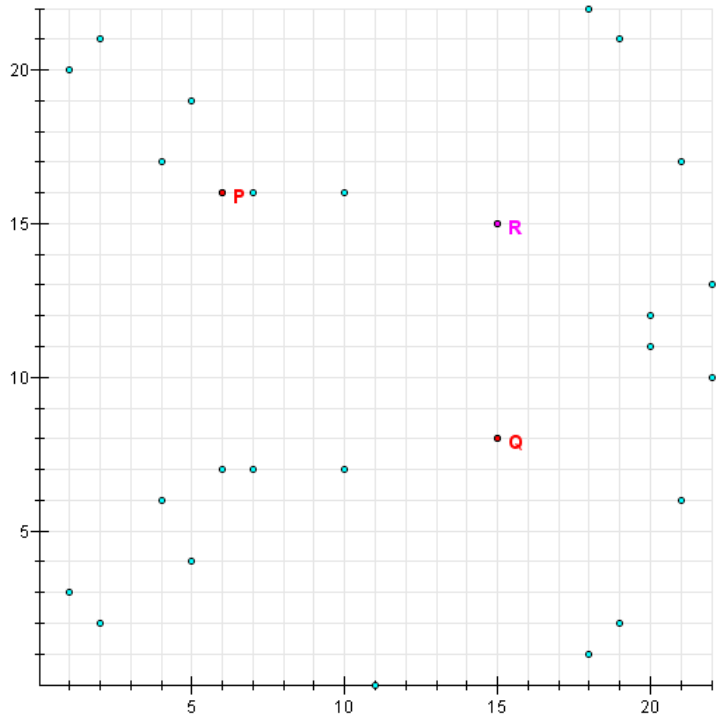


Figura D.2. Ejemplo de suma de puntos en una curva elíptica.

6.2. ACUERDO DE CLAVE DE DIFFIE-HELLMAN CON CURVAS ELÍPTICAS: ECDHKA

524. Al igual que sucede con el acuerdo de clave de Diffie-Hellman y con el criptosistema de ElGamal, los criptosistemas de curvas elípticas basan su seguridad en el problema del logaritmo discreto que se define en el grupo sobre el que se trabaja. En este caso, como el

grupo de puntos de una curva elíptica es un grupo aditivo, este problema se define de la siguiente manera.

525. El «Problema del Logaritmo Discreto sobre una Curva Elíptica» o «Problema del Logaritmo Elíptico» (en inglés, Elliptic Discrete Logarithm Problem, EDLP), es el siguiente: dado el grupo de puntos de una curva elíptica sobre un cuerpo finito, E, generada por el punto G y dado un punto de la curva $P \in E$, se trata de determinar el entero x tal que $x \cdot P = G$.

526. Debe tenerse en cuenta que como la notación en este caso es aditiva, todos los puntos de la curva se obtienen como un múltiplo del generador de la misma. Dicho de otro modo, el producto y la potencia en Zp* se transforman en una curva elíptica en suma y producto, respectivamente.

527. El Acuerdo de Clave de Diffie-Hellman (DHKA) presentado en la sección 1 del ANEXO B tiene una versión para el caso en que se utilice el grupo de puntos de una curva elíptica sobre un cuerpo finito, E, en lugar del grupo Zp*. En este caso, se conoce como «Acuerdo de Clave de Diffie-Hellman con Curvas Elípticas» (en inglés, Elliptic Curve Diffie-Hellman Key Agreement, ECDH o ECDHKA).

528. Con el fin de considerar una de las situaciones más sencillas, se supondrá que el grupo de puntos de la curva elíptica E es cíclico, es decir, que posee un generador, G, y que tiene orden primo (si no fuera así, se elegiría un subgrupo cíclico de la curva generado por un punto con un orden primo, cercano al orden de la curva).

529. Dos usuarios A y B que quieran intercambiar una información común mediante el ECDHKA proceden como sigue:

- Seleccionan públicamente una curva elíptica E sobre un cuerpo finito de q elementos $\mathbf{F}_q$ y eligen un elemento G de E que genere un subgrupo de orden primo suficientemente grande (en la práctica, se supone que G genera la propia curva E).
- El usuario A (respectivamente B) genera un número entero aleatorio a (resp. b), calcula el punto de la curva dado por $a \cdot G$ (resp. $b \cdot G$) en E y envía el resultado a B (resp. A).
- A recibe $b \cdot G$ y calcula $a \cdot (b \cdot G) = a \cdot b \cdot G$. Análogamente, B recibe $a \cdot G$ y calcula $b \cdot (a \cdot G) = b \cdot a \cdot G$.
- Por tanto, A y B poseen un punto en común de la curva elíptica: $a \cdot b \cdot G = b \cdot a \cdot G$, que es secreto, y puede ser usado como clave secreta compartida por A y B.

6.3. CRIPTOSISTEMA DE CURVAS ELÍPTICAS

530. Como ya hemos dicho, utilizando como referencia el criptosistema de ElGamal, presentado en la sección 5 del ANEXO B, es posible definir un criptosistema de curvas elípticas utilizando como grupo base el definido por los puntos de una curva elíptica.

531. Las ventajas de los criptosistemas elípticos son las siguientes:

- La ley del grupo es muy fácil de implementar. De hecho, la suma de puntos sólo requiere unas pocas operaciones aritméticas en el cuerpo finito base, que supondremos tiene q elementos, $\mathbf{F}_q$.
- El problema del logaritmo elíptico (ECDLP) es más difícil de resolver que el problema del logaritmo discreto (DLP) en $\mathbf{Z}_p^*$.

- El grupo de puntos de la curva tiene, aproximadamente, q elementos, es decir, $\#E = q$; de modo que si el mayor factor primo de q es n, entonces el mejor algoritmo conocido para resolver el ECDLP requiere tiempo exponencial en n.
- Se puede utilizar una curva elíptica sobre el cuerpo finito $\mathbf{F}_q$, siendo $q \approx 2^{160}$ y conseguir la misma seguridad que con el criptosistema de ElGamal sobre el grupo $\mathbf{Z}_p^*$, siendo $p \approx 2^{1024}$.
- Es decir, la seguridad que proporciona el criptosistema de ElGamal con claves de 1024 bits no es mayor que la que proporciona una curva elíptica con claves de 160 bits.

532. En el Cuadro D.1 se presentan las longitudes en bits de las claves que proporcionan seguridades equivalentes para los diferentes criptosistemas ([Hankerson et al., 2005], [NIST, SP800-57]).

Cuadro D.1. Compación de niveles de seguridad y longitudes de claves.

Nivel de seguridad (bits)	Longitud de clave para RSA	Longitud de clave para ECC
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

533. De forma análoga a otros criptosistemas de clave pública, los ECC constan de tres fases:

- Generación de claves,
- Cifrado de mensajes,
- Descifrado de mensajes.

6.4. GENERACIÓN DE CLAVES

534. Si el usuario A desea generar sus claves para un criptosistema de curvas elípticas, debe seguir los siguientes pasos:

- A genera una curva elíptica, E, sobre el cuerpo finito $\mathbf{F}_q$ de modo que la representación binaria de q tenga, al menos, 160 bits.
- A elige un generador de la curva, G, cuyo orden sea primo, p, con $p \approx q$.
- A genera un número aleatorio a, con $1 < a < p–2$, y calcula el valor del punto de la curva $a \cdot G$.
- La clave pública de A es el conjunto $(\mathbf{F}_p, E, G, a \cdot G)$ y su clave privada es el número a.

6.5. CIFRADO DE MENSAJES

535. Si el usuario B desea cifrar un mensaje m para A seguirá el siguiente protocolo:

- B obtiene la clave pública de A: $(\mathbf{F}_p, E, G, a \cdot G)$.
- B representa el mensaje m como un punto de la curva E, es decir, transforma m en $M = (m_1, m_2) \in E$, con $m_1, m_2 \in \mathbf{F}_p$.
- B genera un número aleatorio x, $1 < x < p–2$, calcula los puntos de la curva dados por

$$P = x \cdot G,$$

$$Q = M + x \cdot (a \cdot G),$$

y envía a A el criptograma, que está formado por $c = (P, Q)$.

536. En este caso, el factor de expansión del criptosistema de curvas elípticas es 2, dado que cada criptograma está formado por dos puntos de la curva, mientras que lo que se cifra es sólo un punto. Sin embargo, su tamaño en bits es menor que en el caso de ElGamal, dado que p es sólo de 160 bits.

537. En general, para ahorrar tiempo en las comunicaciones, se suele recomendar el uso de una representación comprimida de los puntos de la curva. En efecto, como la ecuación de la curva es conocida, para definir un punto Z = (z1, z2), basta con dar su coordenada z1 y un bit de la coordenada z2, no todo el valor de z2. De hecho, si la representación de un punto de la curva es Z = (z1, 0), la representación de su punto opuesto, –Z, es –Z = (z1, 1). De esta manera para representar un punto de la curva sólo son necesarios p+1 bits.

6.6. DESCIFRADO DE MENSAJES

538. Una vez que el destinatario A recibe el criptograma c = (P, Q), para recuperar el mensaje original, m, realiza los siguientes pasos:

- A utiliza su clave privada, a, para calcular

$$R = a \cdot P = a(x \cdot G) = a \cdot x \cdot G.$$

- A obtiene el punto de la curva original sumando al segundo punto del criptograma, el punto opuesto del obtenido anteriormente:

$$Q + (-R) = M + x \cdot (a \cdot G) + (-a \cdot x \cdot G) = M + x \cdot a \cdot G - a \cdot x \cdot G = M.$$

6.7. SEGURIDAD

539. Dado que el problema matemático sobre el que se fundamentan los ECC es el logaritmo discreto sobre curvas elípticas, los mejores ataques contra este tipo de criptosistemas son los mismos que los ya presentados para el caso del criptosistema de ElGamal en la sección 5 de este ANEXO B. Sin embargo, existen otros ataques que son específicos de las curvas elípticas, dado que éstas pueden elegirse sin las suficientes condiciones de seguridad.

540. A continuación se presentan los principales ataques contra los ECC.

541. ATAQUE POR FUERZA BRUTA. Este ataque contra el ECDLP consiste en probar todos los posibles valores de un entero k hasta que se obtenga la solución buscada k = a, es decir, hasta obtener el valor conocido de a·G, lo que permitiría conocer la clave privada del usuario al que se ataca. Este ataque puede evitarse si los parámetros del criptosistema se eligen con los tamaños recomendados para las claves, según los diferentes niveles de seguridad requeridos, de modo que el tiempo de computación necesario sea lo suficientemente elevado.

542. ATAQUE DE POHLIG-HELLMAN ([Pohlig and Hellman, 1978]). El ataque trata de resolver el ECDLP en subgrupos de orden primo pequeño del grupo original, de modo

que las soluciones parciales pueden ser luego complementadas mediante el Teorema Chino del Resto para obtener la solución global. Con el fin de evitar este ataque, se recomienda que el orden de la curva elíptica definida sobre el cuerpo finito, #E, tenga un divisor primo grande, por ejemplo p, de modo que éste sea mayor que el máximo entre 2160 y 22·s–1, donde s es el nivel de seguridad deseado, en bits, es decir, uno de los siguientes valores: 80, 112, 128, 192, 256, etc. Dicho de otro modo, si se cumple que

$$p > \max\{2^{160}, 2^{2 \cdot s-1}\}, \text{ donde } s \in \{80, 112, 128, 192, 256, \ldots\}.$$

543. ATAQUE DE FREY Y RÜCK ([Frey y Rück, 1994]). Este ataque generaliza el ataque de Menezes, Okamoto y Vanstone ([Menezes et al., 1993b]) generalmente conocido como ATAQUE MOV. El ataque reduce el ECDLP definido sobre una curva elíptica a un DLP sobre conjuntos donde este problema es algo más sencillo de resolver. Ambos ataques pueden ser evitados si la curva elítpica se define sobre el cuerpo Fq, de modo que se verifique la siguiente condición: qi ≠ 1 (mod n), para todo i, con 1 ≤ i ≤ 100.

544. ATAQUES A CURVAS ANÓMALAS. Se dice que una curva elíptica definida sobre un cuerpo finito Fq es «anómala» si se cumple que el orden de la curva verifica #E = q. Semaev ([Semaev, 1998]), Smart ([Smart, 1999]) y Satoh y Araki ([Satoh and Araki, 1998]) mostraron que el ECDLP en curvas anómalas se puede resolver de modo eficiente. Por tal motivo, dichas curvas deben ser evitadas. Los ataques a curvas anómalas se puede prevenir si se comprueba que se verifica lo siguiente:

$$\#E \neq q.$$

545. Existen otros ataques más sofisticados que los presentados anteriormente y que requieren de conceptos matemáticos más complejos, por lo que remitimos al lector interesado a las siguientes referencias: [Frey, 2001], [Gaudry et al., 2002], [Jacobson et al., 2001], [Maurer et al., 2002] y [Menezes and Qu, 2001].

7. CUADRO RESUMEN

Cuadro D.2. Resumen de longitudes de claves de criptosistemas basados en curvas elípticas para el ENS

Cifrado simétrico:	**Nivel Bajo**	**Nivel Medio**	**Nivel Alto**
ECC			
2.5. Protección de la confidencialidad	No se aplica	Permitido Claves: 224-255 bits	Permitido Claves: ≥ 256 bits
2.6. Protección de la autenticidad y de la integridad	No se aplica	Permitido Claves: 224-255 bits	Permitido Claves: ≥ 256 bits
2.7. Cifrado de la información	No se aplica	No se aplica	Permitido Claves: ≥ 256 bits
2.8. Protección de las claves criptográficas	Permitido Claves: 224-255 bits	Permitido Claves: 224-255 bits	Permitido Claves: ≥ 256 bits

ANEXO C. FUNCIONES RESUMEN

1. FUNCIONES RESUMEN

546. En general, los protocolos que calculan la firma electrónica de un documento electrónico son bastante lentos, tanto en software como en hardware. Con el fin de ahorrar tiempo de computación, lo que se suele hacer es firmar una especie de resumen del documento en lugar de firmar el documento completo. La ventaja de este procedimiento es que el resumen del documento se limita a unos cientos de bits, sin importar cómo de grande sea el documento original.

547. Para calcular esta especie de resumen del documento se utiliza un tipo especial de función, denominada «Función Resumen» (en inglés, Hash Function). Para más información sobre este tipo de funciones, véase [Fúster et al., 2004], [Katz and Lindell, 2008], [Menezes et al., 1997], [Stinson, 2006].

548. Una función resumen, por ejemplo, h, puede definirse como una función que asocia a cualquier documento electrónico de cualquier tamaño, m, un resumen suyo, m, de longitud fija, es decir, h(m) = m. Así, si los resúmenes que proporciona una función resumen determinada son, por ejemplo, de 256 bits, resulta que para cualquier documento o mensaje, sin importar su tamaño, siempre proporcionará resúmenes de 256 bits.

549. Sin embargo, no sirve cualquier función, las funciones resumen para ser utilizadas en los protocolos criptográficos y de firma electrónica han de cumplir determinadas condiciones.

550. En primer lugar, han de ser fácilmente computables, es decir, el cálculo del resumen de un documento dado debe requerir muy poco tiempo de computación. Además, las funciones resumen deben conocerse públicamente.

551. Por otra parte, si los resúmenes de los documentos para una determinada función resumen tienen siempre una longitud fija, para cualquier longitud de los documentos, se deduce que el número de posibles resúmenes es mucho menor que el de posibles documentos. Dicho de otra manera, el número de mensajes de cualquier longitud es infinito, mientras que el número de resúmenes de mensajes diferentes de, por ejemplo, 256 bits, es «sólo» de 2^{256}.

552. Además, como una de las principales aplicaciones de las funciones resumen es para la firma electrónica de documentos, otra propiedad que deben cumplir estas funciones es que no debe ser fácil encontrar dos documentos diferentes cuyo resumen sea el mismo, porque en ese caso habría discrepancias acerca de cuál de los dos documentos es el que se ha utilizado. Por todo ello, las funciones resumen deben cumplir las siguientes propiedades:

- *Dependencia de bits*: el resumen de un mensaje o documento, $h(m) = \underline{m}$, debe ser una función compleja dependiente de todos los bits del mensaje, de tal manera que si se modifica un único bit del mensaje, su resumen debería cambiar, aproximadamente, la mitad de sus bits.
- *Resistencia a la preimagen*: dado un resumen $\underline{m}$, debe ser computacionalmente difícil obtener un documento m de modo que $h(m) = \underline{m}$.
- *Resistencia a la segunda preimagen:* dado un documento m_1, debe ser computacionalmente difícil encontrar otro documento m_2 diferente del anterior,

$m_1 \neq m_2$, de modo que ambos tengan el mismo resumen, $\underline{m}$, es decir, tales que $h(m_1) = h(m_2) = \underline{m}$.

- *Resistencia a colisiones*: debe ser computacionalmente difícil encontrar dos mensajes distintos cualesquiera m_1, m_2, $m_1 \neq m_2$, de modo que $h(m_1) = h(m_2) = \underline{m}$.

553. Las propiedades C y D anteriores pueden parecer la misma, pero no lo son en absoluto. Debe notarse que la propiedad D es menos restrictiva que la C. En efecto, en la propiedad C se supone que el documento m1 es conocido y debe encontrarse otro documento m2 con la condición señalada; mientras que en la propiedad D no hay condiciones previas sobre ninguno de los dos documentos, de hecho, ni siquiera tendrían por qué tener sentido, gramaticalmente hablando.

554. Si no se cumplieran las propiedades anteriores, las funciones resumen serían vulnerables, es decir, podrían ser rotas por el ataque conocido como de la «Paradoja del cumpleaños». Esta paradoja procede de un problema matemático que puede plantearse en los siguientes términos: «Si un año tiene n = 365 días, ¿cuántas personas debe haber en una sala para que la probabilidad de que dos de ellas celebren su cumpleaños el mismo día sea mayor del 50%?» Otra forma de plantearla sería la siguiente: «Determinar la confianza (es decir, que la probabilidad sea mayor de 0,5) de que en una sala con k personas, dos de ellas celebren su cumpleaños el mismo día».

555. La solución a este problema es sorprendente (de ahí el nombre de paradoja). En efecto, basta con que haya k = 23 personas en una sala para tal probabilidad sea mayor de ½. De hecho la probabilidad para 23 personas es 0,5072972343. Debe tenerse en cuenta que el enunciado del problema anterior no especifica qué día concreto debe celebrarse el cumpleaños, sólo exige que se celebre el mismo día, esto es, uno cualquiera de los 365 días del año.

556. La paradoja anterior se traduce para las funciones resumen equiparando los mensajes con las personas y los resúmenes con los cumpleaños.

557. En las secciones siguientes se presentan las principales funciones resumen que se utilizan en la actualidad.

2. MD5

558. La función resumen conocida como MD5 sigue siendo utilizada hoy en día a pesar de que se han hallado algunas debilidades ([Wang and Yu, 2005]) y su uso no está permitido para aplicaciones de seguridad. No obstante se ha incluido en esta guía dado que algunos certificados digitales emitidos por determinadas autoridades de certificación siguen empleándola. En cualquier caso, debe quedar claro que el uso de esta función no está permitido en el ENS.

559. La función resumen MD5 fue propuesta por Rivest ([Rivest, 1992]) y proporciona resúmenes de 128 bits, es decir, dado un mensaje o documento, m, de cualquier longitud, la función MD5 proporciona un resumen del mismo, MD5(m) = m, de sólo 128 bits. Debe tenerse en cuenta que la longitud del documento puede ser de 10 bits, 1 Kbits, 100 Gigabits o cualquier otra longitud. En todos los casos, el resumen de cada mensaje siempre tendrá 128 bits.

560. Las iniciales MD de la función MD5 proceden de los apellidos de los dos primeros autores que propusieron este tipo de funciones: Merckle ([Merkle, 1990]) y Damgård ([Damgård, 1990]).

561. El esquema general de cómo funciona la función resumen MD5 es el siguiente:

- El documento a resumir, *m*, se expande de modo que su longitud sea múltiplo de 512 bits, añadiendo, si fuera necesario, bits al final del mismo.
- A continuación se seleccionan cuatro vectores, *A*, *B*, *C* y *D*, cada uno de los cuales tiene 32 bits de longitud. Una vez que estos cuatro vectores se concatenan, es decir, se colocan uno tras de otro, se obtiene un vector de 128 bits ($32 \cdot 4 = 128$), llamado *IV* (Vector de Inicialización o en inglés, Inicialization Vector).
- Se considera el primer bloque de 512 bits del documento *m* y se llevan a cabo diferentes operaciones lógicas con dicho bloque y con los 128 bits del *IV* = *ABCD*. La salida de esta operación tiene 128 bits (ver [Menezes et al., 1997]).
- La salida anterior se considera como un nuevo *IV* y se hace operar la misma con el segundo bloque de 512 bits del documento *m*.
- Este proceso se itera hasta agotar el mensaje extendido.
- La salida del algoritmo MD5 es el resumen que corresponde a los 128 bits de la última de las iteraciones anteriores.

2.1. EJEMPLOS

562. A modo de ejemplo, el resumen MD5 del mensaje «Funciones resumen», que tiene sólo unos pocos bits, es, en hexadecimal, el siguiente:

```
95c576534dd748c1dae464146aeffd30;
```

mientras que el resumen MD5 del fichero «A-2010-1330.pdf» (disponible en https://www.aecid.gob.es/export/sites/default/sede_electronica/galerias/descargas/leyes_sede_electronica/Real_Decreto_3_2010_de_8_de_enero.pdf) que contiene el Real Decreto 3/2010, de 8 de enero, por el que se regula el Esquema Nacional de Seguridad en el ámbito de la Administración Electrónica, que consta de 50 páginas, es este otro:

```
df69635e7a18c7021cf79671088a8902.
```

3. SHA-1

563. La función resumen SHA-1, cuyo nombre procede de las iniciales Secure Hash Algorithm (Algoritmo Resumen Seguro) es una función resumen propuesta a raíz de una modificación de la función MD5. SHA-1 fue adoptada por el National Institute for Standard and Technology americano (NIST) como función resumen estándar ([NIST, FIPS180-1]) y proporciona resúmenes de 160 bits. SHA-1 es más segura que MD5, debido, fundamentalmente, a que sus resúmenes son más largos que los de la función MD5.

564. SHA-1 es actualmente la función resumen más utilizada, a pesar de que se han encontrado algunas debilidades. De hecho, se han encontrado colisiones con menos de 2^{69} operaciones, cuando los ataques por fuerza bruta necesitaban de 2^{80} operaciones ([Wang et al., 2005]). En cualquier caso, el número de operaciones necesario para encontrar colisiones es tan elevado que puede considerarse como una función segura a corto plazo.

565. El diseño de esta función es muy similar al de la función MD5, si bien se deben tener en cuenta las siguientes diferencias:

- SHA-1 tiene 5 vectores iniciales: *A*, *B*, *C*, *D* y *E*, en lugar de los cuatro de MD5. Como cada uno de ellos sigue siendo de 32 bits, el vector de inicialización *IV* en este caso tiene 160 bits y así, el resumen de SHA-1 es de 128+32 = 160 bits.
- A cada bloque de 512 bits del documento se le aplican 80 vueltas o iteraciones, mientras que MD5 sólo lleva a cabo 64 vueltas.

3.1. EJEMPLOS

566. Utilizando los mismos documentos que los empleados en el caso de la función MD5, el resumen SHA-1 del mensaje «Funciones resumen», en hexadecimal, es el siguiente:

```
38d689b6ca9a19f84029fc2970a5134faa5889d5;
```

mientras que el resumen SHA-1 del fichero «A-2010-1330.pdf» que contiene el Real Decreto 3/2010 (50 páginas) tiene la misma longitud que el anterior:

```
615a5de5a8de7f22c25348ce514817cd4de5bf04.
```

4. RIPEMD-160

567. La función RIPEMD-160 (Primitivas de Integridad de Resumen de Mensaje, o en inglés, RACE Integrity Primitives Evaluation Message Digest) es una función resumen que proporciona un resumen de 160 bits y fue desarrollada en 1996 ([Dobbertin et al., 1996]).

568. Existen versiones de esta función que tienen como salida resúmenes de 128, 256 y 320 bits, llamadas RIPEMD-128, RIPEMD-256 y RIPEMD-320, respectivamente.

569. La versión RIPEMD-160 es una versión mejorada de la primitiva RIPEMD, que estaba basada en la MD4 función (predecesora de la MD5).

570. Dado que la longitud de su resumen es de 160 bits, su seguridad es similar a la de SHA-1, si bien su uso está mucho menos extendido, por lo que ha sido menos analizada. De todos modos, en 2004 se publicaron algunas debilidades para la versión de 128 bits de RIPEMD que no afectan a las otras versiones ([Wang et al, 2004]).

571. La función de compresión general de RIPEMD-160 aplica entradas de 21 palabras (variables encadenadas de 5 palabras más 16 bloques del mensaje, con palabras de 32 bits) en salidas 5 palabras. Cada bloque de entrada se procesa en paralelo con versiones distintas de la función de compresión.

4.1. EJEMPLOS

572. Calculando los resúmenes RIPEMD-160 de los mismos documentos ya empleados en otros ejemplos, se tiene que el resumen del mensaje «Funciones resumen», en hexadecimal, es el siguiente:

```
811ec2c5694c7750b9a2d815fcdc150ebdc2ee91;
```

mientras que el resumen del fichero «A-2010-1330.pdf» es:

```
70183a62a52e01e9478eac1c47267d67b6e4fc52.
```

5. SHA-2

573. La función resumen SHA-2 es la sucesora de la SHA-1 y ha sido propuesta después de conocerse que se pueden encontrar colisiones para la SHA-1 con 263 operaciones (resultado debido a Shamir, no publicado).

574. El algoritmo de la función resumen SHA-2 contiene los subalgoritmos SHA-224, SHA-256, SHA-384 y SHA-512, que proporcionan resúmenes de documentos de 224, 256, 384 y 512 bits, respectivamente, lo que aumenta considerablemente la seguridad del algoritmo ([NIST, FIPS180-2]).

575. SHA-256 y SHA-512 se calculan con palabras de 32 y 64 bits, respectivamente. Ambas usan diferentes constantes y número de rondas, pero su estructura es, básicamente, la misma.

576. La función SHA-224 se definió para que coincidiera con la longitud de dos claves de Triple DES (TDES). Ésta función y la SHA-384, son versiones truncadas de las funciones SHA-256 y SHA-512, respectivamente, además de que se calculan con vectores iniciales diferentes.

577. SHA-2 no se utiliza en la actualidad tanto como la SHA-1 a pesar de que proporciona una seguridad mucho mayor. Las razones hay que buscarlas en la falta de compatibilidad con protocolos, implementaciones y dispositivos ya existentes en el mercado.

578. Conviene destacar que el DNIe incluye dentro de sus protocolos criptográficos para la firma electrónica de documentos la posibilidad de calcular resúmenes mediante las funciones SHA-1 y SHA-2, si bien la segunda de ellas no está activada por la razón apuntada en el párrafo anterior

5.1. EJEMPLOS

579. Las funciones SHA-256, SHA-384 y SHA-512 proporcionan los siguientes resúmenes (en hexadecimal), respectivamente, del siguiente mensaje: «Funciones resumen»:

SHA-256: `1bae916bd6fd589ceda7fcb9414d1e66`

`5de44f5316d1363ffd91a204962bd71d,`

SHA-384: `53fcb71d121ba6c858fa774b0521f9c8`

`25c6de6710a4ea3b67eca611eeaad534`

`2553c8b12ab243d4acb09ec70192604b,`

SHA-512: `634ead292f05389b6942f3d2eea5d33c`

`ffa16e7fbe194e206a56dad8c909fe1c`

`71400986d417d9a57680de75b24c3463`

`735bf9f37c67b341bbf780a39871f51a.`

580. Los correspondientes resúmenes del fichero «A-2010-1330.pdf» son, respectivamente:

SHA-256: `51be21833a1db2a2c5ae018de49ccb03`

`6cb651cca0a3b30ae6db5d0e191d4c23`,

SHA-384: `fc5b0a75b8ad3c2c7a8e8b236391a548`

`e4fa6e45a2d1588fda1f73b8d8903d71`

`030dd825abcfb532657b44f57f2da9e3`,

SHA-512: `1135651068d86010ff10b4a02be8647e`

`442ce01bef834c695efebc277fa03518`

`80a2ea8993d04e16c82fa6be7e66c909`

`be731ee041559238fc17f818334baa38`.

6. SHA-3

581. En noviembre de 2007, el NIST anunció formalmente el inicio de una competición internacional para elegir una nueva función resumen, para ser considerada como estándar y que será llamada función resumen SHA-3 ([NIST, SHA-3]). La presentación de funciones candidatas se terminó el 31 de octubre de 2008 y se espera que la función ganadora se dé a conocer en 2012.

582. El NIST recibió 64 propuestas iniciales de funciones candidatas a ser SHA-3. Después de una primera revisión, se seleccionaron las 51 que cumplían los requisitos solicitados en la convocatoria original. Las 51 candidatas forman la lista de funciones que se ha dado en llamar la Ronda 1.

583. En Febrero de 2009, el NIST celebró el congreso «First SHA-3 Candidate Conference» en la Universidad Católica de Lovaina (Bélgica) para que los autores de las 51 funciones candidatas admitidas en la Ronda 1 presentaran y defendieran sus algoritmos. Después de este congreso, el NIST ha recibido comentarios a los algoritmos propuestos y ha elaborado una segunda lista de funciones candidatas que son las que forman la llamada Ronda 2. La lista de las 14 funciones candidatas a convertirse en la nueva SHA-3, que configuran la Ronda 2, son las listadas en el Cuadro E.1.

Cuadro E.1. Lista de funciones resumen de la Ronda 2.

Algoritmo	Autor principal
BLAKE	Jean-Philippe Aumasson
Blue Midnight Wish	Svein Johan Knapskog
CubeHash	D. J. Bernstein
ECHO	Henri Gilbert
Fugue	Charanjit S. Jutla
Grøstl	Lars Ramkilde Knudsen
Hamsi	Ozgul Kucuk

Algoritmo	Autor principal
JH	Hongjun Wu
Keccak	Joan Daemen
Luffa	Dai Watanabe
Shabal	Jean-Francois Misarsky
SHAvite-3	Orr Dunkelman
SIMD	Gaetan Leurent
Skein	Bruce Schneier

584. La competición continuó con la celebración del «Second SHA-3 Candidate Conference» en la Universidad de Los Ángeles en Santa Bárbara (California, USA) durante los días 23 y 24 de Agosto de 2010. Los resultados de esta conferencia se han publicado el 9 de diciembre de 2010 y han dado como resultado una lista de 5 algoritmos, que son los que constituyen la Ronda final, de entre los que saldrá el futuro SHA-3. La lista de los 5 candidatos finales es: BLAKE, Grøstl, JH, Keccak y Skein.

7. HMAC

585. Los sistemas de cifrado de documentos y mensajes garantizan la confidencialidad de los mismos, es decir, que su contenido se mantiene en secreto. Sin embargo, no aseguran que tales documentos sean auténticos. Para garantizar esta propiedad, es necesarios hacer uso de los llamados «Códigos de Autenticación de Mensajes» (en inglés, Authentication Message Code, MAC).

586. En general, un MAC (ver [Menezes et al., 1997]) se calcula a partir de la última parte del criptograma de un mensaje cuando el tipo de sistema de cifrado utilizado tenga la propiedad de que todos los bits cifrados sean función de todos los bits del mensaje original. Por ejemplo, en los cifradores en bloque de tipo TDES, el MAC de un mensaje podría ser el último bloque de 64 bits del criptograma ([Katz and Lindell, 2008], [Stinson, 2006]).

587. Un tipo especial de códigos de autenticación de mensajes se obtiene cuando se considera, además, una función resumen. En este caso se habla de «Código de Autenticación de Mensaje con Resumen» (en inglés, Hash Message Authentication Code, HMAC). En el caso particular en que tal código permita, además, el uso de claves, se conoce como «Código de Autenticación de Mensaje con Resumen y Clave» (en inglés, Keyed-Hash Message Authentication Code, KHMAC).

588. La versión más extendida de HMAC es la adoptada como estándar por el NIST ([ANSI, X9.71], [NIST, FIPS198]) y usa una clave de 512 bits. La seguridad de este MAC depende, obviamente, de la seguridad de la función resumen empleada.

589. El operador HMAC con una clave k sobre un mensaje m se define como sigue:

$$\mathrm{HMAC}_k(m) = h((k \oplus opad) \| h((k \oplus ipad) \| m)),$$

donde h es la función resumen (por ejemplo, SHA-1, SHA-2, etc.), $\|$ es la operación de concatenación, $opad$ = 5c5c5c···5c5c e $ipad$ = 363636···3636.

7.1. EJEMPLOS

590. Los valores HMAC basados en las funciones resumen MD5, SHA-1, SHA-256, SHA-384 y SHA-512 del mensaje «Funciones resumen», con la clave «Hash» son, respectivamente, los siguientes:

HMAC-MD5: `f864dc3dced79b1b4b9940af8a1dfe48,`

HMAC-SHA-1: `2cb93aabff961bae6fa3f73fb9700e1b8ed3b984,`

HMAC-SHA256: `1054690d7dfeecca19e9008b0fc66c44`
`08f74a90a6a57fcf4ab2f0eee087f3ea,`

HMAC-SHA384: `d07374fcf65b354d17ce973f119d380f`
`ce50cfbe4a73b69d461ac98756bd2cd6`
`3aa2c9f8c5e0e10b34f598bb02a19042,`

HMAC-SHA512: `3a12502ae8f06b97821bcd2b36da06ca`
`284a4fbdb80f93b9cac0cba5c979bdbb`
`50a168791c3b8bb625c417f91a085edc`
`24f5194b102fd519be1a6409edda52a4.`

591. Por su parte, los correspondientes HMAC para el fichero «A-2010-1330.pdf» proporcionados por las mismas funciones, con la clave «RD 3/2010» son, respecivamente:

HMAC-MD5: `de65e1ca6489452418070c02de331c86,`

HMAC-SHA-1: `1a4f91dd71be565edf49953b19403c9f97fb0604,`

HMAC-SHA256: `b5148d6eeaa25b84007d6093fd899a49`
`3dcd30d40b9bf02f4e5959da6debedc5,`

HMAC-SHA384: `92790c0bd169a8a30097365099af5348`
`3c3898644bce509c981de16309cc79f6`
`bea82e35dd12d2d15c024b710e613790,`

HMAC-SHA512: `c3c30530a0ba5594e6108876cb9f4c83`
`0ccb08cce836b492e1b8d5ab724b9d98`
`05ca2bcd4cbeb5fc1d0fb41be2520d31`

38fce5885fdb1ac1db6f3578480845f4.

8. AUTENTICACIÓN DE USUARIOS

592. La posesión de claves privadas compartidas entre varios usuarios permite la creación de protocolos de autenticación, para demostrar la personalidad del usuario ante otro usuario o ante un equipo servidor.

593. Hay dos clases de protocolos, según el número de participantes. El más simple es aquel en el que solo participan una pareja de usuarios que comparten una clave simétrica secreta. El más complejo es aquel que además de la pareja de usuarios se requiere una tercera parte confiable o tercero de confianza que actúa como servidor de seguridad.

594. El primer tipo de protocolo es adecuado para redes con pocos usuarios (del orden del centenar, o menor), ya que cada uno puede almacenar y gestionar fácilmente la clave a usar con cada uno de sus corresponsales. En cambio, cuando la cantidad de posibles corresponsales es muy amplia (del orden de miles, o mayor) el único tipo manejable es el segundo; en este caso cada usuario solo dispone de una clave que comparte con el servidor de seguridad, mientras que el servidor de seguridad tiene un archivo con la clave de cada uno de sus usuarios clientes.

595. Los protocolos de autenticación emplean una serie de valores conocidos y permanentes, como los identificadores de los terminales de la red y otros que son números de un solo uso, generalmente aleatorios. A continuación se listan las abreviaturas de los elementos que se utilizan en ellos:

A: Identificador de Alicia, un usuario legítimo del sistema.

B: Identificador de Benito, un usuario legítimo del sistema.

T: Identificación de Teresa, una tercera parte confiable.

E_K: Algoritmo de cifrado de clave simétrica K.

K: Clave simétrica secreta.

r_A, r_B: Números aleatorios, generados por Alicia o Benito.

t_A, t_B: Sellos temporales, generados por Alicia o Benito (ver sección 6 del ANEXO D).

n_A, n_B: Números de uso único, generados por Alicia o Benito, también llamado «nonce» (*number used once*).

8.1. PROTOCOLO DE AUTENTICACIÓN CON SELLO TEMPORAL

596. Sean dos corresponsales Alicia y Benito, que comparten la clave simétrica secreta K y poseen el algoritmo de cifrado E. En el Cuadro E.2 se presenta un esquema de protocolo de autenticación fuerte unilateral, que hace uso de un sello temporal según la norma [ISOIEC, 9798-2].

Cuadro E.2. Autenticación con sello temporal.

Paso	Dato	Operación	Flujo de información
1	$E_K(t_A,B)$	Alicia envía un sello temporal cifrado a Benito.	$A \rightarrow B$

597. Benito comprueba, tras descifrar el mensaje, que el sello temporal no ha caducado, que el identificador es el suyo propio y que Alicia conoce la clave; así Alicia queda autenticada frente a Benito. El sello temporal tA evita un ataque por reenvío retardado contra Benito. El indicador de Benito B evita un ataque inmediato contra Alicia por reflexión. En este protocolo Benito reconoce a Alicia, pero Alicia no sabe si realmente habla con Benito; si se quiere asegurar ese extremo, ha de repetirse el protocolo en sentido contrario.

598. Como se puede observar, este método es sumamente económico, pues se consigue la identificación segura con un solo mensaje. La desventaja que presenta es que se requiere que los relojes locales sean seguros, exactos y estén en sincronismo con un reloj central. El mecanismo de sincronización entre relojes se puede conseguir fácilmente mediante protocolos adicionales de comunicación entre el servidor y los clientes; pero éstos deben a su vez ser también seguros, lo que conduce a una nueva complicación del sistema, que desvirtúa de alguna forma su supuesta sencillez.

8.2. PROTOCOLO DE AUTENTICACIÓN CON NÚMEROS ALEATORIOS

599. Sean dos corresponsales Alicia y Benito, que comparten la clave simétrica secreta K y poseen el algoritmo de cifrado E. En el Cuadro E.3 se muestra un ejemplo de identificación bilateral que hace uso de números aleatorios junto con los identificadores de Alicia y Benito ([ISOIEC, 9798-2]):

Cuadro E.3. Identificación por desafío-respuesta seguro.

Paso	Dato	Operación	Flujo de información
1	r_A	Alicia genera un numero aleatorio r_A.	$A \rightarrow B$
2	$E_K(r_A,r_B,A)$	Benito genera r_B, lo concatena con r_A y A, lo cifra con el algoritmo E, bajo la clave K y lo envía a Alicia; entonces Alicia reconoce a Benito y recupera r_B.	$A \leftarrow B$
3	$E_K(r_A,r_B,B)$	Benito recupera r_A, r_B y B, reconociendo a Alicia.	$A \rightarrow B$

600. Es parecido al método de desafío respuesta, pero perfeccionado. En este caso se añaden sendos indicativos de Alicia y Benito. Los indicativos evitan el ataque por reflexión, pues las respuestas son diferentes según a quién vayan dirigidas.

601. Para una identificación unilateral, hubiesen bastado los dos primeros pasos del protocolo y no hubiese sido necesario el número rB.

602. Este método precisa un generador de números aleatorios, que ha de ser criptográficamente seguro.

8.3. PROTOCOLO DE AUTENTICACIÓN CON FUNCIONES UNIDIRECCIONALES (SKID3)

603. En ciertos casos no es deseable utilizar un algoritmo de cifrado y descifrado E, de modo que éste se puede sustituir por una función unidireccional dependiente de una clave K; esa función podría ser un MAC. La comprobación de identidad se lleva a cabo calculando el MAC en recepción a partir de los datos recibidos y/o conocidos, que debe coincidir con el MAC recibido. En este caso ha de enviarse el número rB en claro en el segundo paso ([ISOIEC, 9798-4]). Un ejemplo de este tipo de protocolo de este tipo es el denominado SKID3, descrito en el Cuadro E.4.

Cuadro E.4. Identificación con MAC (SKID3).

Paso	Dato	Operación	Flujo de información
1	r_A	Alicia genera un numero aleatorio r_A.	$A \rightarrow B$
2	r_B, MAC_K (r_A,r_B,A)	Benito genera otro número aleatorio r_B y fabrica un MAC_K, que envía a Alicia.	$A \leftarrow B$
3	MAC_K (r_A,r_B,B)	Alicia fabrica un MAC_K, que compara con el recibido; comprueba la identidad de Benito y fabrica otro MAC_K que envia a benito para que este la reconozca.	$A \rightarrow B$

8.4. PROTOCOLO WIDE MOUTH FROG

604. Sean dos corresponsales Alicia y Benito, que comparten sendas claves KAT y KBT con una tercera parte confiable, Teresa, y poseen el algoritmo de cifrado E. Este protocolo, llamado en inglés Wide mouth frog por sus autores ([Burrows et al., 1993]), permite, además de la autenticación, el establecimiento de una clave de sesión. Se resume en el Cuadro E.5.

Cuadro E.5: Protocolo *Wide mouth frog*.

Paso	Dato	Operación	Flujo de información
1	A, E_{KAT} (t_A,B,K)	Alicia genera una clave K e inicia el protocolo.	$A \rightarrow T$
2	A, E_{KBT} (t_T,A,K)	Benito recupera la clave K y el indicativo de Alicia	$T \rightarrow B$

605. En el primer paso, Alicia contacta con una tercera parte confiable, Teresa, con la que comparte la clave KAT. Le envía su identificación junto con un mensaje cifrado con dicha clave; el mensaje incluye un sello temporal tA, el indicador de Benito y la clave que quiere usar en la sesión de trabajo K. Teresa sabe que es Alicia porque demuestra conocer su clave común.

606. En el segundo paso, Teresa envía a Benito, cifrado con la clave KBT, otro sello temporal, el indicativo de Alicia y la clave de sesión K. Benito reconoce a Teresa porque ésta demuestra conocer su clave común y acepta la clave de sesión que le traslada para dialogar con Alicia.

607. Los sellos temporales tA y tT impiden los ataques por repetición.

608. La desventaja de este protocolo, aparentemente tan sencillo, es que se obliga a Teresa a realizar el trabajo de abrir una sesión de comunicación con Benito. Otro inconveniente es que ha de garantizarse que Alicia sepa desempeñar la tarea —no simple— de generar

claves aleatorias K, criptográficamente seguras, con todas las garantías necesarias. Y, por supuesto, hay que contar con la exigencia de mantener relojes seguros sincronizados.

8.5. PROTOCOLO DE NEEDHAM-SCROEDER

609. Sean dos corresponsales Alicia y Benito, que comparten sendas claves KAT y KBT con una tercera parte confiable, Teresa, y poseen el algoritmo de cifrado E. El protocolo Needham-Scroeder ([Needham and Schroeder, 1987]) permite la autenticación fuerte del usuario Alicia frente a Benito y el establecimiento de una clave secreta de sesión K, con la ayuda de Teresa, que es el servidor de autenticación (ver Cuadro E.6).

Cuadro E.6. Protocolo Needham-Scroeder.

Paso	Dato	Operación	Flujo de información
1	A, B, r_A	Alicia inicia un protocolo con Teresa.	$A \rightarrow T$
2	$E_{KAT}(r_A,B,K,E_{KBT}(K,A))$	Alicia recupera la clave K y una credencial para Benito $E_{KBT}(K,A)$.	$A \leftarrow T$
3	$E_{KBT}(K,A)$	Alicia retransmite la credencial a Benito, que recupera K y A.	$A \rightarrow B$
4	$E_K(r_B)$	Alicia reconoce a Benito.	$A \leftarrow B$
5	$E_K(r_B-1)$	Benito reconoce a Alicia.	$A \rightarrow B$

610. Todo el trabajo de generación de claves seguras queda localizado en Teresa, liberando a los corresponsales de tal responsabilidad.

611. Alicia contacta directamente con Teresa, que le proporciona la credencial adecuada, y posteriormente con Benito; éste se comunica únicamente con Alicia. Se supone que, previamente, Alicia dispone de una clave secreta que comparte con Teresa, KAT, que también comparte una clave secreta, KBT, con Benito.

612. Los números rA, rB y rB−1 sirven para evitar ataques por repetición.

613. En el segundo paso, Alicia verifica que Teresa es legítima y que no está contestando a un protocolo antiguo gracias a rA. En el cuarto paso, Alicia reconoce a Benito. En el quinto paso, Benito hace el reconocimiento de Alicia y se asegura de que ésta no está reproduciendo un protocolo realizado anteriormente. Un mérito del protocolo es que libera del mantenimiento de un reloj seguro sincronizado. Pero, a cambio, presenta el problema de que las claves de sesiones anteriores son válidas siempre y se podrían reutilizar si hubiesen sido capturadas por un atacante hostil.

8.6. PROTOCOLO DE OTWAY-RESS MODIFICADO

614. Sean dos corresponsales Alicia y Benito, que comparten sendas claves KAT y KBT con una tercera parte confiable, Teresa, y poseen el algoritmo de cifrado E. El protocolo de Otway-Rees modificado ([Otway and Rees, 1987]) subsana las debilidades del protocolo Needham-Schroeder. Además, permite compartir una clave secreta de sesión K generada por Teresa —un servidor de claves confiable— y lograr una identificación recíproca. Alicia comparte una clave secreta con Teresa KAT, mientras que Benito comparte la clave KBT con Teresa. M, rA y rB son tres números aleatorios empleados para garantizar la frescura (ver Cuadro E.7).

Cuadro E.7. Protocolo Otway-Rees modificado

Paso	Dato	Operación	Flujo de información
1	$M, A, B, E_{KAT}(r_A,M,A,B)$	Alicia inicia un protocolo con Benito.	$A \rightarrow B$
2	$M, A, B, E_{KAT}(r_A,M,A,B)$, $E_{KBT}(r_B,M,A,B)$	Teresa comprueba que: $(M,A,B) = (M,A,B) = (M,A,B)$.	$B \rightarrow T$
3	$E_{KAT}(r_A,K)$, $E_{KBT}(r_B,K)$	Teresa genera una clave de sesión K, para Alicia y Benito, que envía cifradas a Benito.	$T \rightarrow B$
3b	$E_K(A,r_B)$	Benito comprueba que $r_B = r_B$, reconociendo a Teresa; recupera K y construye $E_K(A,r_B)$.	B
4	$E_{KAT}(r_A,K)$, $E_K(A,r_B)$	Alicia comprueba que $r_A = r_A$, recupera K y construye $E_K(r_B)$	$A \leftarrow B$
5	$E_K(r_B)$	Benito reconoce a Alicia.	$A \rightarrow B$

615. Alicia se pone en contacto únicamente con Benito y le envía una solicitud de inicio de sesión. Benito se encarga de contactar al servidor. El servidor reconoce a los solicitantes gracias a que demuestran la posesión de sus claves particulares. Benito reconoce al servidor porque éste demuestra conocer su clave particular. Alicia reconoce al servidor porque éste conoce su clave particular, mientras que a Benito le reconoce de forma implícita porque le suministra una información válida EKAT(rA,K) y además explícitamente, porque le suministra EK(A,rB). Benito sabe que Alicia fue identificada por el servidor en el paso 3; pero en el paso 5 obtiene una confirmación porque Alicia demuestra estar en posesión de su clave particular.

8.7. PROTOCOLO KERBEROS

616. El protocolo Kerberos permite la autenticación del usuario Alicia frente a un servidor de información B y el establecimiento de una clave secreta de sesión, con la intervención y respaldo de una tercera parte confiable T, que es el servidor de autenticación. Sus diseñadores lo denominaron Kerberos y fue diseñado en el Massachusetts Institute of Technology (MIT) (http://www.kerberos.org/), dentro del proyecto Athena.

617. Este protocolo también es similar al de Needham-Schroeder; pero incluye un número l de tiempo de vida de la clave y un sello temporal de Alicia tA, que evitan un ataque por repetición de una clave antigua (ver Cuadro E.8).

Cuadro E.8. Protocolo Kerberos.

Paso	Dato	Operación	Flujo de información
1	A, B, r_A	Alicia inicia un protocolo con Kerberos.	$A \rightarrow Kerberos$
1 b	$Tique = E_{KBT}(K, A, l)$	Kerberos fabrica un tique para el servidor B.	$Kerberos$
2	$E_{KAT}(r_A, K, l, B)$	Alicia recupera la clave de sesión K.	$Kerberos \rightarrow A$
2b	$E_K(A, r_B)$	Alicia fabrica un autenticador.	A
3	Tique + Autenticador	El servidor B recupera K y valida a Alicia.	$A \rightarrow B$
4	$E_K(t_A)$	Alicia verifica la frescura del sello t_A.	$A \leftarrow B$

618. Alicia contacta directamente con Kerberos, que le proporciona la credencial adecuada, y posteriormente se pone en contacto con Benito. Benito se comunica únicamente con Alicia. Se supone que, previamente, Alicia dispone de una clave secreta que comparte con Kerberos, KAT. El servidor también comparte una clave secreta KBT con Kerberos. Tiene varias versiones, aquí se presenta la versión 5 en su forma básica.

619. Alicia se identifica, solicita conexión con el servidor Benito, y envía un número rA a Kerberos.

620. Kerberos, sin molestarse en saber si Alicia es auténtica o miente, le contesta y le devuelve un «Tique» para el servidor B, junto con un paquete cifrado con la clave de Alicia, que contiene una clave secreta de sesión K, y el número rA, el período de vida de la clave l y el indicador del servidor B. Si Alicia es auténtica, puede seguir el protocolo, recuperando rA (que le sirve para autenticar a Kerberos), K, l, y B.

621. A continuación, Alicia envía al servidor el Tique junto con un «Autenticador», que se confecciona cifrando con la clave K el identificador A y un sello temporal tA. Entonces, el servidor realiza la autenticación de Alicia comparando la información encerrada en el Tique y el Autenticador.

622. Finalmente, el servidor devuelve a Alicia el sello temporal cifrado, para que Alicia tenga la confirmación de la identidad del servidor B y de que no se trata de la repetición de un protocolo antiguo.

623. En algunas versiones de este protocolo se sustituye el número aleatorio rA por un sello temporal, elaborado por Alicia.

ANEXO D. ESQUEMAS DE FIRMA ELECTRÓNICA

1. FIRMAS ELECTRÓNICAS

624. Los criptosistemas de clave pública, además de permitir la confidencialidad de los datos cifrados, facilitan el diseño de otros protocolos, entre los que destacan los relacionados con los de la firma electrónica de mensajes o documentos electrónicos.

625. La «firma electrónica» o «firma digital» de un mensaje o documento es un protocolo electrónico cuyo objetivo es análogo al de la firma manuscrita, es decir, garantizar que quien remite un mensaje es realmente quien dice serlo. Este protocolo electrónico puede llevarse a cabo sin importar que la comunicación sea confidencial o no.

626. Dicho de otra forma, un protocolo de firma electrónica puede utilizarse bien con el único fin de garantizar la propiedad de un documento público mediante la firma electrónica de su autor, bien para garantizar la procedencia de una comunicación confidencial.

627. En general, ambos protocolos son diferentes, si bien en determinados casos, tal diferencia es mínima y sólo afecta al proceso de verificación y no al de elaboración de la firma.

628. La firma se calcula para cada mensaje o documento y depende tanto del mensaje como del remitente. Al contrario de lo que sucede con la firma manuscrita, que es siempre la misma para cualquier documento y sólo varía cuando cambia el firmante, la firma electrónica es única para cada mensaje. Además, las firmas deben ser fáciles de calcular, verificar y difíciles de falsificar.

629. Las firmas digitales pueden clasificarse de la siguiente manera:

- *Implícita*: la firma electrónica calculada se incluye en el propio fichero que contiene el documento que se firma.
- *Explícita*: la firma se añade al documento a firmar pero no forman parte de él, es decir, se almacena en un fichero diferente.
- *Privada*: la firma permite identificar al firmante sólo en el caso en que el destinatario comparta una información confidencial o secreta con el firmante.
- *Pública*: la firma identifica al firmante utilizando información públicamente conocida del propio firmante.
- *Revocable*: la firma no impide que el firmante pueda negar que fuera él quien firmó del documento.
- *Irrevocable*: la firma impide que el firmante pueda negar que la firma le pertenece.

630. En general, los protocolos de firma electrónica más extendidos son aquellos en los que la firma calculada es explícita, pública e irrevocable.

631. Existen otros tipos de firma, como las firmas ciegas (o a ciegas), las delegadas, las que se elaboran en nombre de un grupo, las múltiples, las autocertificadas, las basadas en la identidad del firmante, etc., si bien su uso está menos extendido. De hecho, estos protocolos de firma no se consideran estándares, por lo que no serán tratados en esta guía.

632. Un protocolo de firma electrónica consta de dos partes, supuesto que cada participante en el protocolo posee su par de claves pública y privada. La primera fase del protocolo consiste en la elaboración de la firma en sí y lo lleva a cabo el firmante del documento haciendo uso de su clave privada; mientras que la segunda parte es la verificación de la

firma y es responsabilidad del destinatario o verificador del documento ([Fúster et al., 2004], [Menezes et al., 1997], [Stinson, 2006]).

633. En los dos protocolos que siguen, la firma es explícita, puesto que la misma se añade al mensaje y no forma parte del mismo; es pública dado que puede ser verificada a partir de información del firmante conocida públicamente; y, finalmente, es irrevocable dado que la única persona que puede llevarla a cabo es el propietario de la clave privada.

1.1. PROTOCOLO DE FIRMA ELECTRÓNICA DE UN DOCUMENTO PÚBLICO

634. Se consideran dos participantes en un protocolo de firma electrónica, F y V, de modo que F es el firmante y V es el verificador de la firma de F. Por simplicidad, tanto F como V utilizan un protocolo de clave asimétrica dado por sus correspondientes procesos de cifrado, E, y de descifrado, D; siendo (A, a) las claves pública y privada de F, respectivamente; mientras que las correspondientes claves de V son (B, b). Además del criptosistema asimétrico establecido por F y V, es necesario que ambos se pongan de acuerdo en una función resumen segura, para lo cual basta con elegir una función conveniente de las alguna de las mencionadas en el ANEXO C, que se denotará por h. Estas funciones deben ser SHA-1, RIPMED-160 y aconsejablemente SHA-2.

635. El protocolo estándar para elaborar la firma electrónica de un documento que no requiera proteger su confidencialidad, es decir, público, m, es el siguiente:

- *F* determina el resumen del documento mediante la función resumen establecida, es decir, halla el valor de

$$h(m) = m_{\mathrm{F}}.$$

- *F* calcula lo que se denomina su rúbrica o firma para el mensaje *m*, para lo que debe cifrar el resumen del documento mediante su clave privada, es decir, determina

$$r = E_a(m_{\mathrm{F}}).$$

- *F* hace público o envía a *V* el par formado por el documento y su rúbrica: (*m*, *r*).

636. Para que el verificador V pueda comprobar la firma electrónica de F para el documento m, lleva a cabo los siguientes pasos:

- *V* determina el resumen del mensaje que *F* le envió mediante la función resumen acordada:

$$h(m) = m_{\mathrm{V}}.$$

- *V* calcula el resumen del mensaje original descifrando con la clave pública de *F*, *A*, la rúbrica recibida:

$$D_A(r) = D_A(E_a(m_{\mathrm{F}})) = m_{\mathrm{F}}.$$

- *V* comprueba si los dos resúmenes calculados coinciden:

$$m_{\mathrm{V}} = m_{\mathrm{F}}.$$

637. En el caso en que ambos resúmenes coincidan, V considerará que la firma del documento m recibido ha sido realmente hecha por F, dado que sólo F es capaz de lleva a cabo el paso a. de la elaboración de la firma, dado que es el único que conoce su clave privada, a.

Además, la igualdad de los resúmenes permite afirmar que el documento original no ha sido manipulado.

638. Si ambos resúmenes no coinciden, V no aceptará la firma como válida porque o bien se ha firmado el documento con una clave privada diferente de la de F, o bien se ha manipulado el documento original.

1.2. PROTOCOLO DE FIRMA ELECTRÓNICA DE UN DOCUMENTO NO PÚBLICO

639. En el caso en que el documento a firmar requiera proteger su confidencialidad y no pueda darse a conocer públicamente, el protocolo para la elaboración de la firma es ligeramente diferente al presentado anteriormente. En este caso, se supone que el firmante cifra y envía al verificador el documento cifrado, sea c el criptograma, y luego procede como sigue:

- *F* determina el resumen del documento mediante la función resumen establecida, es decir, halla el valor de

$$h(m) = m_{\mathrm{F}}.$$

- *F* calcula su rúbrica para el mensaje *m*, por lo que debe cifrar el resumen del documento mediante su clave privada, es decir, determina

$$r = E_a(m_{\mathrm{F}}).$$

- *F* computa su firma electrónica para el mensaje sin más que cifrar con la clave pública del verificador, *B*, la rúbrica calculada en el paso anterior, es decir, determina

$$s = E_B(r) = E_B(E_a(m_{\mathrm{F}})).$$

- *F* envía a *V* el su firma electrónica para el documento *m*: *s*.

640. Para que V verifique la firma electrónica de F para el documento m, sigue los siguientes pasos:

- *V* recupera el documento original, *m*, descifrando el criptograma recibido, *c*, mediante su clave de descifrado correspondiente y determina su resumen con la función resumen acordada:

$$h(m) = m_{\mathrm{V}}.$$

- *V recupera la rúbrica, r, calculada por F para el documento, descifrando la firma electrónica con su clave privada, b:*

$$D_b(s) = D_b(E_B(r)) = r.$$

- *V obtiene el documento resumen del documento original, m, descifrando la rúbrica mediante la clave pública de F:*

$$D_B(r) = D_B(E_b(m_{\mathrm{F}})) = m_{\mathrm{F}}.$$

- *V comprueba si el resumen del documento obtenido en el paso anterior, mF, coincide con el resumen del documento, mV, que calculó en el paso a. de este protocolo:*

$$m_{\mathrm{V}} = m_{\mathrm{F}}.$$

641. Si la verificación es incorrecta, tanto la firma como el propio documento se rechazan. Las posibles razones son:

- *Que el documento, ha sido modificado.*
- *Que el paquete de firma ha sido modificado.*
- *Que la firma y el documento no están relacionados.*
- *Que no se ha empleado la clave pública de verificación adecuada.*
- *Que no se ha empleado la clave pública de cifrado adecuada.*

642. Conviene señalar que en determinadas ocasiones se ha propuesto que dado que se firma un resumen del documento y no el documento en sí, no es necesario ejecutar el paso c. en el protocolo de elaboración de la firma anterior, ni, en consecuencia, el paso b. en el protocolo de verificación. En estas ocasiones se aduce que como a las funciones resumen se les exige ser seguras frente al ataque por colisiones, es decir, que obtener un documento a partir del conocimiento de su resumen es computacionalmente muy difícil, basta con cifrar el resumen con la clave privada del firmante, reduciendo la firma electrónica a sólo la rúbrica.

643. Sin embargo, esta propuesta no es segura y no deben admitirse protocolos que se implementen de esta forma. En efecto, si los protocolos de elaboración y verificación anteriores se redujeran tal y como se ha dicho, cualquier usuario podría calcular la rúbrica a partir de la clave pública del firmante y obtener el resumen de un documento que se desea mantener en secreto.

644. Con este procedimiento, la seguridad del documento sólo dependería de la seguridad de la función resumen empleada, puesto que invertir el proceso de dicha función daría el mensaje original. Sin embargo, es importante señalar que aunque las funciones resumen son seguras frente a ataques contra la pre-imagen y frente a colisiones, no emplean ningún tipo de clave, por lo que el secreto del documento no dependería, paradójicamente, de ninguna clave, a pesar de estar empleando criptosistemas de clave pública.

2. FIRMA ELECTRÓNICA RSA

645. El protocolo de firma digital presentado en la sección 1 del presente Anexo es un protocolo genérico de firma electrónica basado en un criptosistema de clave pública general. A continuación se presenta cómo se concreta este proceso cuando se emplea el criptosistema RSA.

646. Por otra parte, se distinguirá si el documento que se firma es confidencial o no, dado que los protocolos de elaboración de la firma y de verificación de la misma son diferentes en ambos casos.

647. En cualquiera de los dos supuestos anteriores, se supondrá que el firmante F posee sus claves pública y privada, (nA, eA) y dA, respectivamente; mientras que las claves correspondientes del verificador V son (nB, eB) y dB, respectivamente. Además, la función resumen acordada por ambas partes será h(m) = m.

2.1. FIRMA ELECTRÓNICA DE UN DOCUMENTO PÚBLICO CON RSA

648. En este protocolo, el firmante F firma un documento m que es conocido por cualquier verificador, dado que se supone que el documento es público.

- *F determina el resumen del documento a firmar, m, mediante la función resumen acordada, es decir, calcula h(m) = m.*
- *F calcula el valor de su firma electrónica para el resumen m mediante su exponente de descifrado, dA, es decir, determina el valor de(5)*

$$s = \underline{m}^{d_A} \pmod{n_A}.$$

- *F envía al verificador su firma s junto con el mensaje m.*

649. El protocolo para la verificación de la firma digital del firmante llevado a cabo por el verificador cuando el documento es público es el siguiente:

- *V calcula el resumen del documento recibido descifrando la firma electrónica mediante la clave pública del firmante:*

$$s^{e_A} \pmod{n_A} = (\underline{m}^{d_A})^{e_A} \pmod{n_A} = \underline{m}^{(d_A \cdot e_A)} \pmod{n_A} = \underline{\underline{m}}.$$

- *V* determina el resumen del mensaje *m*, $h(m) = \underline{m}$, que *conoce por ser público y verifica si tal resumen coincide con el obtenido en el paso anterior*:

$$\underline{m} = \underline{\underline{m}}.$$

- *Sólo si ambos resúmenes coinciden la firma es correcta.*

2.2. FIRMA ELECTRÓNICA DE UN DOCUMENTO NO PÚBLICO CON RSA

650. En este caso se supone que la firma se lleva a cabo para un documento que requiere confidencialidad, es decir, que sólo será conocido por el verificador cuando descifre el criptograma recibido.

- *F determina el resumen del documento a firmar, m, mediante la función resumen acordada, es decir, calcula h(m) = m.*
- *F calcula el valor de su rúbrica para el resumen m mediante su exponente de descifrado, dA, es decir, determina el valor de*

$$r = m^{d_A} \pmod{n_A}.$$

- *A continuación determina su firma electrónica para el documento m cifrando la rúbrica anterior con la clave pública del verificador:*

$$s = r^{e_B} \pmod{n_B}.$$

- *F* envía al verificador su firma *s* junto con el mensaje cifrado *c*.

651. El protocolo de verificación de la firma digital del firmante que lleva a cabo el verificador en el caso en que el documento no sea público es el siguiente:

- *V obtiene la rúbrica de F para el mensaje m descifrando la firma recibida haciendo uso de su clave privada:*

$$s^{d_B} \pmod{n_B} = (r^{e_B})^{d_B} \pmod{n_B} = r^{(e_B \cdot d_B)} \pmod{n_B} = r.$$

(5) La expresión genérica *a^b* representa la potencia *b*-ésima de *a*, mientras que *(a^b)^c* representa el valor de *a* elevado a la potencia *b* y todo ello elevado a *c*.

- *V* calcula el resumen del mensaje descifrando la rúbrica mediante la clave pública del firmante:

$$r\text{^}e_A \pmod{n_A} = (\underline{m}\text{^}d_A)\text{^}e_A \pmod{n_A} = \underline{m}\text{^}(d_A \cdot e_A) \pmod{n_A} = \underline{m}.$$

- *V* determina el resumen $h(m) = \underline{m}$, del mensaje *m*, descifrando el criptograma recibido y verifica si tal resumen coincide con el obtenido en el paso anterior:

$$\underline{m} = \underline{m}.$$

- Sólo si ambos resúmenes coinciden, *V* da por válida y correcta la firma.

652. Los ataques contra el protocolo de firma digital RSA son los mismos que contra el propio criptosistema dado que ambos protocolos se basan en el mismo problema matemático.

3. FIRMA ELECTRÓNICA ELGAMAL

653. Para el caso particular del protocolo de firma según el sistema de ElGamal, se puede suponer que las claves pública y privada del firmante, F, son, respectivamente, (p, g, A = ga) y a; mientras que las correspondientes del verificador, V, son (p, g, B = gb) y b. Además, la función resumen acordada por ambas partes será h(m) = m.

654. Para este criptosistema, los protocolos para el caso en que el documento sea público o no, no son muy diferentes, por lo que no se señalará la distinción entre ambos, sin necesidad de considerar dos protocolos diferentes.

655. Tanto si el documento es público como si no lo es, el proceso para la elaboración de la firma es el mismo y es el siguiente:

- El firmante *F* calcula el resumen del documento confidencial:

$$h(m) = \underline{m}.$$

- *F* genera una clave de sesión, esto es, un número aleatorio *x* entre 1 y *p*–2, $1 \leq x \leq p-2$, primo con *p*–1, es decir, tal que

$$\text{mcd}\,(x, p-1) = 1.$$

- *F* calcula su rúbrica:

$$r = g^x \pmod{p}.$$

- *F* resuelve en la incógnita *s* la ecuación siguiente:

$$\underline{m} = a \cdot r + x \cdot s \pmod{p-1}$$

- para lo cual calcula $x^{-1} \pmod{p-1}$ y determina

$$s = x^{-1}\,(\underline{m} - a \cdot r) \pmod{p-1}.$$

- *F* envía a *V* su firma electrónica para el documento *m*:

$$(r, s).$$

656. Para verificar la firma de F, el verificador, V, lleva a cabo el siguiente protocolo:

- *V calcula el resumen del mensaje. En el caso en que el documento sea público, lo hará utilizando dicho documento, mientras que si el documento es no público, deberá*

descifrar el criptograma correspondiente antes de calcular el resumen de dicho documento. En ambos casos, sea el resumen

$$\underline{m} = h(m).$$

- *V* calcula los siguientes dos valores
  $u = A^r \pmod p$, donde A es parte de la clave pública del firmante.
  $v = r^s \pmod p$.
- *V* comprueba si

$$u \cdot v \pmod p = g^{\underline{m}} \pmod p.$$

- Si la comprobación es correcta, se considera que la firma es auténtica.

657. En efecto, basta con tener en cuenta que

$$u \cdot v \pmod p = A^r \pmod p \cdot r^s \pmod p$$

$$= (g^a)\hat{}r \cdot (g^x)\hat{}s \pmod p$$

$$= g^{a \cdot r + x \cdot s} \pmod p$$

$$= g^{\underline{m}} \pmod p.$$

3.1. FIRMA ESTÁNDAR DEL NIST: DSS Y DSA

658. El Instituto de Estándares y Tecnología norteamericano (NIST) ha publicado una norma para la firma digital de documentos ([NIST, FIPS186-2], [NIST, FIPS186-3]) que es obligatoria para todas las entidades que deseen mantener comunicaciones con el gobierno americano.

659. Esta norma se conoce como Estándar de Firma Digital (DSS, Digital Signature Standard) y consta de tres partes: Generación de claves, elaboración de la firma y verificación de la firma.

660. Para generar las claves que se van a utilizar en el DSS se sigue el siguiente protocolo:

- El firmante genera un número primo, p, con L bits[6], es decir, de modo que

$$2^{L-1} < p < 2^L.$$

- A continuación, determina un número primo, q, que sea divisor de p–1, y con N bits[7], esto es,

$$q \mid p-1 \text{ y } 2^{N-1} < q < 2^N.$$

- Calcula un generador, g, con $1 < g < p$, del único subgrupo cíclico de Z_p^* de orden q.
- Genera un entero, x, con $0 < x < q$, como su clave privada y determina su clave pública, y, calculando

$$y = g^x \pmod p.$$

- Finalmente determina aleatoriamente un entero k, único para cada mensaje, de modo que $1 < k < q$.

[6] El valor de L no es único. Las posibles elecciones de este número se mencionan más abajo.
[7] Tampoco el valor de N es único. Sus posibles valores, según el valor elegido para L, se incluyen posteriormente.

661. Para llevar a cabo la firma digital DSS de un mensaje, m, se ejecuta el protocolo siguiente:

- El firmante calcula el resumen del mensaje a firmar:

$$h(m) = \underline{m}.$$

- Luego selecciona un entero aleatorio k, con $0 < k < q$ y calcula

$$r = (g^k \pmod p) \pmod q.$$

- Más tarde resuelve en la incógnita s la congruencia

$$\underline{m} = -a \cdot r + k \cdot s \pmod q,$$

- calculando $k^{-1} \pmod q$ y computando

$$s = k^{-1} (\underline{m} + a \cdot r) \pmod q.$$

- La firma digital es el par (r, s).

662. Los posibles valores de L y N, es decir, las longitudes en bits recomendadas por el NIST en 2009 ([NIST, FIPS186-3]) para los tamaños de p y q son los siguientes:

$L = 1024, N = 160.$
$L = 2048, N = 224.$
$L = 2048, N = 256.$
$L = 3072, N = 256.$

663. El protocolo para la verificación de la firma digital DSS, es el siguiente:

El verificador determina

$$w = s^{-1} \pmod q.$$

Luego calcula los dos valores siguientes

$$u_1 = \underline{m} \cdot w \pmod q,$$
$$u_2 = r \cdot w \pmod q.$$

A continuación determina

$$v = (g\text{^}u_1 \cdot y\text{^}u_2 \pmod p) \pmod q.$$

Finalmente comprueba si $v = r$, teniendo en cuenta que se verifica lo siguiente:

$$v = (g\text{^}u_1 \cdot y\text{^}u_2 \pmod p) \pmod q$$
$$= (g^{\underline{m} \cdot w \pmod q} \cdot g^{a \cdot r \cdot w \pmod q} \pmod p) \pmod q$$
$$= (g^{w \cdot (\underline{m} + a \cdot r) \pmod q} \pmod p) \pmod q$$
$$= (g^k \pmod p) \pmod q$$
$$= r.$$

664. Debido a que la generación de las claves de la norma DSS anterior no es tan inmediata como pudiera parecer, sobre todo porque no es fácil determinar la factorización de p–1 ni tampoco localizar de forma rápida un divisor q de p–1, lo que se hace en la práctica es utilizar un algoritmo que se conoce como Algoritmo de Firma Digital (DSA, Digital Signature Algorithm).

665. El algoritmo determina los parámetros del protocolo DSS de modo más eficiente de la siguiente forma:

En primer lugar, el firmante selecciona un número primo q de N bits, es decir, con $2N-1 < q < 2N$. Para ello repite la generación aleatoria de un entero impar, q, hasta que sea primo, haciendo uso del test de pseudo-primalidad de Miller-Rabin.
A continuación selecciona un primo p de L bits, es decir, con $2L-1 < p < 2L$. Para ello genera aleatoriamente un entero n de modo que

$$(2^{L-1}-1)/(2\cdot q) < n < (2^{L}-1)/(2\cdot q)$$

hasta que $p = 2\cdot n\cdot q+1$ sea primo.

Finalmente, obtiene un elemento g de Zp* que tenga orden q. Este generador se consigue sin más que generar aleatoriamente un entero z, con $1 < z < p-1$, hasta que se verifique que

$$z^{(p-1)/q} \pmod p = g \pmod p \neq 1.$$

666. Una vez generados los parámetros del protocolo, se procede como se ha indicado antes para la elaboración y verificación de la firma digital de un documento.

4. FIRMA ELECTRÓNICA CON CURVA ELÍPTICA: ECDSA

667. El protocolo de firma digital para curvas elípticas se basa en el de ElGamal y es análogo al presentado para el DSA (ver la sección 3 de este Anexo), pero en notación aditiva. La forma habitual de llevar a cabo este protocolo se conoce como Algoritmo de Firma Digital con Curva Elíptica (ECDSA, Elliptic Curve Digital Signature Algorithm) según las normas [ANSI, X9.62] y [ANSI, X9.63].

668. En este caso, se supone dada una curva elíptica dada, E, definida sobre un cuerpo finito F de q elementos, de modo que G es el punto base de la curva elíptica (o generador), cuyo orden es n¸ y de modo que su cofactor, es decir, el cociente entre el orden de la curva y n es c. Se supone que las claves pública y privada del firmante, F, son, respectivamente, (G, A = a·G) y a; mientras que las correspondientes del verificador, V, son (G, B = b·G) y b. Además, la función resumen acordada por ambas partes es h(m) = m.

669. La forma concreta de llevar a cabo este protocolo de firma es la siguiente:

El firmante calcula el resumen del mensaje:

$$h(m) = \underline{m}.$$

Genera una clave de sesión, es decir, un número aleatorio k con $1 < k < q-1$, de modo que sea primo con q–1, es decir, tal que mcd(k, q–1) = 1 y calcula

$$k\cdot G = (x_1, y_1),$$

$$r = x_1 \pmod q.$$

Si $r = 0$, el firmante selecciona otro valor para k.
Calcula $k^{-1} \pmod q$ y determina

$$s = k^{-1}\cdot(\underline{m}+a\cdot r) \pmod q,$$

La firma digital es el par (r, s).

670. Para verificar la firma del firmante, el verificador lleva a cabo el siguiente protocolo:

Calcula el resumen del mensaje, ya sea porque conoce el mensaje al ser éste público o porque lo ha descifrado si era no público:

$$\underline{m} = h(m).$$

Comprueba que los enteros r y s están en el intervalo $[1, q-1]$.
Calcula

$$w = s^{-1} \pmod{q}.$$

Calcula

$$u_1 = \underline{m} \cdot w \pmod{q},$$

$$u_2 = r \cdot w \pmod{q}.$$

Calcula

$$u_1 \cdot G + u_2 \cdot A = (x_0, y_0),$$

$$v = x_0 \pmod{q}.$$

El verificador acepta la firma como válida si y sólo si se cumple que $v = r$.

671. Las longitudes en bits recomendadas por el NIST en 2009 ([NIST, FIPS186-3]) para los tamaños del orden de G, denotado por |n|; de su cofactor, c; del tamaño del cuerpo primo, es decir, si el cuerpo tiene un número primo p de elementos; y de los cuerpos binarios, es decir, cuerpos con 2m elementos, son los siguientes:

$161 \leq |n| \leq 223$, $c < 2^{10}$, $|p| = 192$, $|m| = 163$.
$224 \leq |n| \leq 255$, $c < 2^{14}$, $|p| = 224$, $|m| = 233$.
$256 \leq |n| \leq 383$, $c < 2^{16}$, $|p| = 256$, $|m| = 283$.
$384 \leq |n| \leq 511$, $c < 224$, $|p| = 384$, $|m| = 409$.
$|n| \geq 512$, $c < 2^{32}$, $|p| = 521$, $|m| = 571$.

672. Por otra parte, las curvas elípticas cuyo uso recomienda el NIST están publicadas en el Apéndice D de [NIST, FIPS186-3]. Existen tres tipos de curvas:

Curvas sobre cuerpos primos: P-192, P-224, P-256, P-384 y P-521.
Curvas sobre cuerpos binarios: K-163, B-163, K-233, B-233, K-283, B-283, K-409, B-409, K-571 y B-571.

673. El significado de los elementos utilizados en los identificadores de las curvas NIST es el siguiente:

P: identifica las curvas sobre cuerpos primos.
B: indica que se trata de una curva sobre cuerpos binarios.
K: indica que se trata de una curva de Koblitz (definida sobre cuerpos binarios).
Cadena de dígitos: expresa el tamaño en bits del cuerpo sobre el que está definida la curva.

674. Finalmente, las curvas elípticas recomendadas por Brainpool (ver [Brainpool, 2005], [Lochter and Merkle, 2010]) son las siguientes: P160r1, P192r1, P224r1, P256r1, P320r1, P384r1 y P512r1; donde es importante señalar que Brainpool no propone curvas sobre cuerpos binarios.

675. El significado de los elementos utilizados en los identificadores de las curvas Brainpool es el siguiente:

- P: identifica las curvas sobre cuerpos primos.
- Primera cadena de dígitos: expresa el tamaño en bits del cuerpo sobre el que está definida la curva.
- r : informa que los coeficientes de la curva se han generado aleatoriamente (random) según el procedimiento descrito en [Brainpool, 2005].
- Dígito final: permite identificar diferentes curvas donde el resto de los parámetros utilizados por esta notación coinciden.

5. COMENTARIOS SOBRE DSA Y ECDSA

676. Como se ha visto, los protocolos de firma DSA y ECDSA son bastante similares, sin embargo, existen algunas diferencias que es importante destacar.

677. En el DSA, la generación de r se hace calculando el valor aleatorio de (gk (mod p)) (mod q), obteniéndose un entero en el intervalo [1, q–1], mientras que en el ECDSA se genera el entero r en el intervalo [1, q–1] considerando la coordenada x del múltiplo aleatorio k·G y reduciéndolo módulo q.

678. En DSA, el primo q es un divisor primo de p–1 de N bits y g es un elemento de orden q; mientras que en ECDSA, n es el orden primo del punto G, con n próximo al tamaño del cuerpo.

679. En el ECDSA, el cuerpo finito F puede fijarse como parámetro común para todos los usuarios, mientras que la curva elíptica, E, y el punto base, G, pueden ser elegidos libremente por cada uno de los usuarios. En este caso, el punto G y su orden, n, deben formar parte de la clave pública del firmante. Si F es fijo, entonces el hardware y el software pueden construirse de modo que optimicen los cálculos en el cuerpo base, teniendo en cuenta que hay una enorme cantidad de posibles elecciones para la curva elíptica sobre un cuerpo base fijado de antemano.

6. SELLOS DE TIEMPO

680. El sellado de tiempo es una técnica criptográfica que se usa para proporcionar información temporal sobre los documentos electrónicos. Permite a otras partes validar el instante en el que un documento ha sido creado y comprobar que dicho documento no ha sido alterado desde entonces.

681. Al hacer posible la trazabilidad temporal de la firma de documentos, el sellado de tiempo ayuda significativamente a aumentar el nivel de confianza que se requiere hoy día en una infraestructura de clave pública. En muchos casos el sellado de tiempo llega a ser la principal prueba a la hora de determinar el estado de un documento. Ejemplos típicos de uso de los sellos de tiempo pueden ser la validación de firmas electrónicas, solicitudes de patentes, inicios de sesión en ordenadores (evaluación de aspectos de seguridad y

funcionamiento en sistemas y redes), suscripciones por ordenador (garantizando la revocación de las suscripciones), servicios de notariado electrónico, voto electrónico, órdenes de venta y recibos o factura electrónica, protección de la propiedad intelectual, sellado de contenidos, etc.

682. La importancia de los sellos de tiempo resulta evidente cuando pensamos en la necesidad de dar validez legal a los documentos electrónicos durante un largo periodo de tiempo ([Buldas et al., 1998]).

6.1. SISTEMAS DE SELLADO DE TIEMPO

683. Las técnicas de sellado de tiempo se agrupan en dos tipos: unas están basadas en el concepto de la confianza distribuida y otras necesitan de una tercera parte de confianza.

684. La técnica basada en la confianza distribuida consiste en generar documentos fechados y firmados por un número de personas muy elevado. El protocolo sería el siguiente:

- Cuando un usuario quiere sellar un documento D con un protocolo distribuido, primero utiliza un generador pseudoaleatorio, lo inicializa con D, y calcula k valores V1, V2,…, Vk.
- El usuario interpreta estos k valores como la identificación de k personas, a las que les envía D.
- Cada una de esas personas añade fecha y hora al documento antes de firmarlo y enviarlo de vuelta al remitente.
- El interesado guarda esas k firmas como sello de tiempo de su documento.

685. Es preciso que k sea suficientemente grande para que el usuario quede persuadido de que no es posible haber corrompido a tantas personas simultáneamente. Esta técnica tiene pues la enorme desventaja de que requiere mucha cooperación. Otra desventaja es el tiempo de vida de las firmas que se envían de vuelta al remitente, un problema recurrente siempre que se usan firmas.

686. Las técnicas que se basan en la existencia de una tercera parte de confianza dependen de la imparcialidad de la entidad que está al cargo de emitir el sello de tiempo, la Autoridad de Sellado de Tiempo (TSA, acrónimo inglés de Time Stamping Authority), que garantiza la existencia de unos determinados datos electrónicos en una fecha y hora concretos ([Pinkas et al., 2003]). Se suelen dividir en dos tipos: esquemas simples y esquemas enlazados ([ISOIEC, 18014]).

687. En los esquemas simples la función de la TSA después de recibir el documento a sellar es añadirle el tiempo actual (un sello de tiempo seguro) y firmar el resultado electrónicamente por medio de un cifrado asimétrico basado en infraestructura de clave pública.

688. El término «sello de tiempo» hace referencia al certificado digital que contiene el código resumen del archivo que contiene el documento, más la indicación de tiempo. Nadie salvo la propia TSA puede alterar el sello de tiempo así obtenido o fechar ningún dato del documento con anterioridad. Este esquema se utiliza en el estándar de sellado de tiempo propuesto para Internet ([Adams et al., 2001]) y en las especificaciones técnicas [ETSI, TS-102023] y [ETSI, TS-101733] del Instituto Europeo de Normas de Telecomunicaciones (European Telecommunications Standards Institute, ETSI).

689. Entre sus ventajas destacan que son compactos y fáciles de calcular, no dependen de otros sellos de tiempo, el protocolo requiere un único paso y es posible comparar sellos de tiempo emitidos por distintas TSAs.

690. El principal inconveniente de estos sistemas es que es preciso confiar absolutamente en la TSA, la cual puede emitir de manera indetectable sellos de tiempo atrasados. Existe otro inconveniente relacionado con las firmas criptográficas, cuyo tiempo de vida puede ser menor que el del documento. Además, todos los sellos de tiempo quedan invalidados cuando la clave privada de la TSA resulta comprometida.

691. El sellado de tiempo basado en esquemas enlazados proporciona autenticación temporal relativa ([Massias and Quisquater, 1997], [Massias et al., 1999]). En ellos lo que se verifica es el orden relativo de emisión entre dos sellos de tiempo cualesquiera. Utilizan funciones resumen, pero no claves, es decir no usan información secreta. Se han propuesto varios tipos de esquemas enlazados. En todos ellos, un certificado de tiempo de un sello emitido en un instante depende de sellos emitidos con anterioridad.

692. Los esquemas enlazados en cadena enlazan, utilizando funciones resumen unidireccionales, cada documento con el que la TSA firmó previamente, construyendo así una cadena cronológica inmodificable ([Haber and Stornetta, 1991]). Resultan poco útiles en la práctica, puesto que la longitud de cada certificado aumenta de modo lineal con el número de sellos de tiempo emitidos hasta ese momento (el número de pasos necesarios para comparar dos sellos depende del número de sellos que hay entre ellos).

693. En los esquemas enlazados en árbol todos los documentos sellados durante un periodo corto (por ejemplo un segundo), se organizan como hojas de un árbol binario ([Merkle, 1979]).

694. El sellado de tiempo, para una petición individual procesada durante una determinada ronda, consiste en información que permite a cada uno comprobar que su petición es parte del proceso de acumulación que produjo el correspondiente valor de la ronda.

695. Los valores calculados en sucesivas rondas se enlazan unos a otros por medio de una función resumen, creando así una cadena temporal infalsificable.

696. Periódicamente, uno de los valores de la ronda se publica en un medio inalterable y ampliamente expuesto a testigos (por ejemplo, un periódico). Estos valores de ronda especiales son la base de la confianza en todos los sellos de tiempo emitidos. Es preciso confiar en esos valores, así como en el tiempo asociado a ellos, lo cual es un requisito razonable puesto que tales valores han sido puestos a la vista de un gran número de testigos. El tiempo absoluto en el que los potenciales verificadores confían es el tiempo indicado en esos medios inalterables.

697. En la Figura F.1 se representa la construcción del árbol correspondiente a una ronda.

698. Cada petición de sellado consiste en un valor resumen de un documento dado. Las hojas del árbol son cada uno de esos valores resumen, representadas por H1, …, H8 en la Figura F.1. Los valores de las hojas se van concatenando de dos en dos, aplicando nuevas funciones resumen y así sucesivamente hasta obtener el valor de la ronda actual.

699. Ese valor superior del árbol de cada ronda, L15, se concatena con el valor obtenido en la ronda precedente Pi-1 y, por último, se aplica la función resumen para obtener el valor real de la ronda, Pi.

700. El sello de tiempo solicitado durante la ronda actual contiene todos los valores necesarios para reconstruir su correspondiente rama del árbol, hasta la raíz. En el ejemplo de la Figura F.1, el sello de tiempo para H4 contiene H3, L9, L14 y Pi-1.

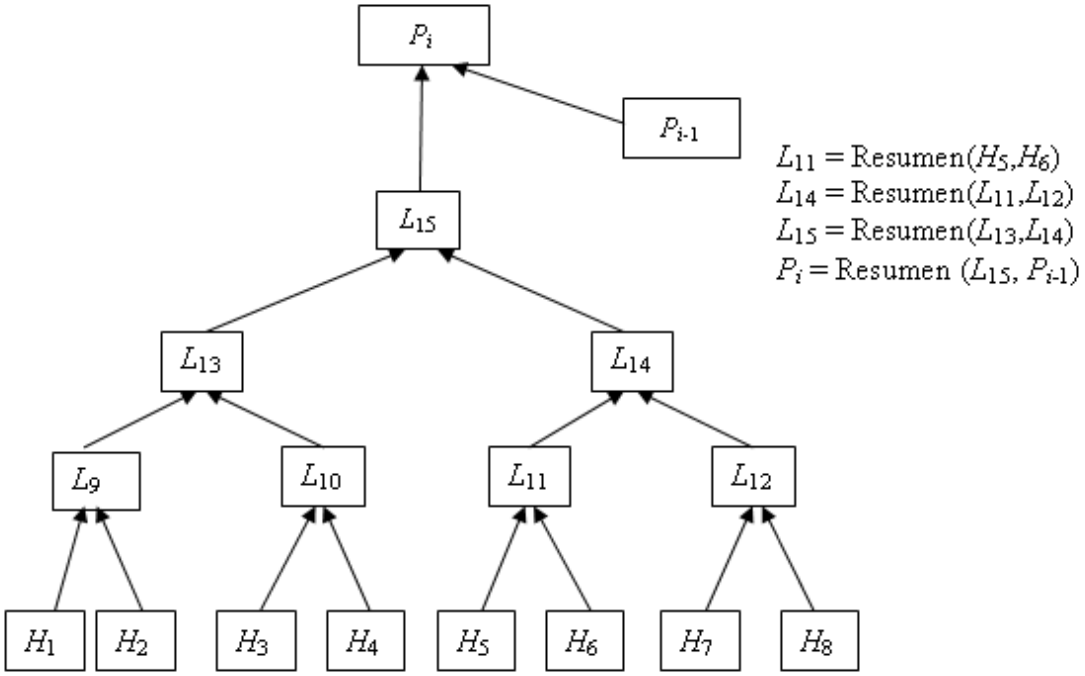


Figura F.1. Representación gráfica de una ronda construida usando un árbol binario.

701. El tamaño de los sellos de tiempo aumenta de modo logarítmico con el número de peticiones procesadas en una ronda.

702. El proceso de verificación consiste en reconstruir la rama del árbol y la cadena enlazada de valores de ronda hasta que se recalcule un valor en el que el verificador confía ([Haber and Stornetta, 1991], [Massias and Quisquater, 1997]).

703. La seguridad se basa en la función resumen utilizada: si es inviable encontrar dos entradas que conduzcan al mismo valor resumen (esto es, si la función no presenta colisiones), entonces es imposible falsificar el árbol binario o la cadena de enlace.

704. Se podrían construir dos árboles para cada ronda, utilizando dos funciones resumen diferentes, de modo que el sistema permaneciese seguro en caso de ruptura de una de las funciones resumen.

705. Otro requisito de seguridad es forzar a los clientes a comprobar los sellos de tiempo tan pronto como los reciban. De ese modo el proceso es auditado continuamente y la TSA no podrá maniobrar de modo poco fiable.

706. Un modo de alargar la vida de los sellos de tiempo consiste en resellar tanto el documento como el sello original antes de que se consiga romper la función resumen ([Haber and Stornetta, 1991]).

707. Estos sistemas tienen la ventaja de que su seguridad es independiente de la clave privada de la TSA, es imposible emitir sellos atrasados y la verificación es muy rápida. Necesita menos requisitos de almacenamiento y aumenta el número de partes interesadas que sirven como testigos.

708. Sin embargo, se trata de un protocolo muy complejo y es difícil comparar sellos de tiempo de diferentes TSAs.

709. En los esquemas de acumulador unidireccional ([Benaloh and de Mare, 1993]) se recogen todas las peticiones recibidas durante una ronda yi, 1 <= i <= m, que son nuevamente funciones resumen de documentos individuales. La acumulación se realiza por medio de la operación de exponenciación modular:

710. Zm = Z0^(Π yi) (mod N), para 1 <= i <= m, con N un modulo tipo RSA.

711. Para verificar el sello de tiempo para una petición particular yi es preciso comprobar que se verifica la siguiente ecuación:

712. Zm = Zj^(yi) (mod N), con Zj = Z0^(Π yi) (mod N), para 1 <= i <= m e i ≠ j.

713. Por tanto un sello de tiempo individual para la petición yi consiste en los valores Zj y Zm. El tamaño de los sellos es independiente del número de peticiones procesadas en la ronda.

714. La desventaja de este método es que la operación de exponenciación modular es menos eficiente que la función resumen.

715. La seguridad de este método se basa en la seguridad del RSA, de modo que si no se conoce la factorización del módulo N es inviable producir un sello de tiempo falso.

7. CUADRO RESUMEN

Cuadro F.1. Resumen de longitudes de claves de criptosistemas asimétricos y tipos funciones resumen en esquemas de firma electrónica para el ENS

2.9. Firma electrónica	Nivel Bajo	Nivel Medio	Nivel Alto
RSA	Permitido Claves ≥ 1024 bits	Permitido (a corto plazo) Claves ≥ 1024 bits	Permitido Claves ≥ 2048 bits
ECC	Permitido Claves: 224-255 bits	Permitido (a corto plazo) Claves: 224-255 bits	Permitido Claves: 256-283 bits
MD5	No permitido	No permitido	No permitido
SHA-1	Permitido (a corto plazo)	Permitido (a corto plazo)	Permitido (a corto plazo)
RIPEMD-160	Permitido (a corto plazo)	Permitido (a corto plazo)	Permitido (a corto plazo)
SHA-2	Permitido	Permitido	Permitido

Cuadro F.2. Resumen de longitudes de claves de criptosistemas asimétricos y tipos funciones resumen en sellos de tiempo para el ENS

2.10. Sellos de tiempo (Nivel Alto)	Esquemas simples	Esquemas enlazados
RSA	Permitido Claves ≥ 3072 bits	No se aplica
ECC	Permitido Claves ≥ 284 bits	No se aplica
MD5	No permitido	No permitido
SHA-1	No permitido	No permitido
RIPEMD-160	No permitido	No permitido
SHA-2	Permitido SHA-256 o superior	Permitido SHA-256 o superior

ANEXO E. GENERADORES DE NÚMEROS PSEUDOALEATORIOS

1. GENERADORES DE SECUENCIAS PSEUDOALEATORIAS

716. Los generadores de secuencias aleatorias (RNG por su nombre inglés: Random Number Generator) y pseudoaleatorios (PRNG por su nombre inglés: PseudoRandom Number Generator) son importantes en matemáticas y en toda clase de tecnologías, pues sirven para simular procesos estadísticos, como en los métodos de Montecarlo etc. Es necesario destacar que estos generadores son prácticamente imprescindibles en criptografía porque es muy difícil imaginar una aplicación criptográfica bien diseñada que no utilice números aleatorios o pseudoaleatorios.

717. Los números aleatorios son necesarios para generar las claves de sesión, los vectores de inicialización, en la generación de claves públicas, generar los parámetros que se emplean una sola vez, en las operaciones de firma digital (nonces), en diferentes protocolos criptográficos y en muchas otras aplicaciones. Si los números aleatorios generados no son seguros, toda la aplicación donde se utilizan resulta insegura.

718. Existen dos tipos básicos de generadores de secuencias: los generadores de números aleatorios (RNG), y los generadores pseudoaleatorios (PRNG). Para aplicaciones criptográficas, ambos generadores producirán una secuencia de ceros y unos que pueden dividirse en subsecuencias o bloques de números aleatorios.

2. GENERACIÓN DE BITS GENUINAMENTE ALEATORIOS (TRNG)

719. Todos los generadores genuinamente aleatorios son el resultado de un proceso físico imprevisible, tal como el ruido térmico en una resistencia, la desintegración de un átomo radioactivo o un proceso óptico cuántico. Todos ellos tienen como característica fundamental la irrepetibilidad, lo que es a la vez una ventaja y un inconveniente.

720. La ventaja es que una vez generada una secuencia, jamás podrá recrearla nadie, aunque disponga del mismo dispositivo que la creó por primera vez. Esta característica es muy conveniente por ejemplo, para proteger el secreto de las comunicaciones. Cuando se ha generado la secuencia, ésta se almacena y se distribuye únicamente a dos corresponsales, que pueden utilizarla para comunicarse secretamente mediante el cifrado en flujo.

721. El gran inconveniente radica en la dificultad de distribuir la clave de cifrado a los dos corresponsales utilizando un canal seguro, que normalmente es un correo humano y que en un futuro —no cercano— puede ser una comunicación óptica cuántica. Otro inconveniente es la necesidad de disponer de un elemento físico más o menos voluminoso, caro y difícil de fabricar. Por todas estas razones en la práctica se utilizan los denominados «generadores pseudoaleatorios».

722. Un generador de números aleatorios verdadero requiere de una fuente natural de términos aleatorios. En general utiliza una fuente no determinista (es decir, una fuente de entropía), junto con alguna función de procesamiento (un proceso de destilación de la entropía) para producir aleatoriedad.

723. La fuente de entropía consiste típicamente de una magnitud física, como el ruido en un circuito eléctrico, la sincronización de los procesos de usuario (por ejemplo, pulsaciones de teclas o movimientos del ratón), o de los efectos cuánticos en un semiconductor, aunque generalmente se utilizan varias combinaciones de estas entradas. De cualquier modo es muy difícil diseñar un dispositivo o un programa que explote esta aleatoriedad y produzca una secuencia de bits que esté libre de sesgos y correlaciones.

724. El uso de un proceso de destilación es necesaria para superar cualquiera de las debilidades en la fuente de entropía. La destilación suele consistir en hacer algunas operaciones lógicas sobre la secuencia obtenida.

725. Las secuencias de salida de los generadores aleatorios se pueden utilizar directamente como secuencias aleatorias o pueden utilizarse como entrada de un generador pseudoaleatorio. En cualquier caso, cuando se utilice directamente como secuencia aleatoria (es decir, sin transformación), ésta deberá al menos satisfacer estrictamente todos los criterios de aleatoriedad que se analizan mediante pruebas estadísticas, que serán explicadas más adelante. Para la mayoría de las aplicaciones criptográficas el generador no deberá estar sujeto a observación o manipulación por parte de un adversario, de modo que el dispositivo debe estar a buen recaudo.

2.1. GENERADORES BASADOS EN HARDWARE

726. Estos generadores explotan la aleatoriedad de algunos fenómenos que ocurren en dispositivos físicos. Entre los ejemplos de estos fenómenos físicos tenemos: tiempo transcurrido entre las emisiones de partículas durante una desintegración radioactiva, ruido térmico producido por diodos semiconductores o resistencias, inestabilidad de la frecuencia del funcionamiento libre de un oscilador, movimientos de un ratón de ordenador, fluctuaciones aleatorias en la velocidad de las unidades de disco provocadas por la turbulencia del aire dentro de la misma, el sonido desde un micrófono o entrada de vídeo de una cámara.

2.2. GENERADORES BASADOS EN SOFTWARE

727. Diseñar un generador de bits aleatorios por software es más difícil que el caso anterior. Entre los procesos en los que pueden basarse se encuentran: el reloj del sistema del ordenador, el contenido de los buffers de entrada o salida, entradas de usuarios, valores del sistema operativo tal como la carga del sistema y las estadísticas de la red, tiempo transcurrido entre pulsaciones de un operador humano al teclear un texto en un ordenador, etc.

2.3. TÉCNICAS PARA ELIMINAR EL SESGO Y LA CORRELACIÓN

728. En la mayoría de los casos, los bits obtenidos de las fuentes aleatorias están sesgados, es decir, suele haber más unos que ceros o viceversa (la probabilidad de que la fuente emita un 1 no es igual a ½) o correlacionados —la probabilidad de que la fuente emita un 1 depende de los bits emitidos con anterioridad—. Esta situación no es aceptable, puesto que se necesita una fuente aleatoria no sesgada, que presente igual probabilidad tanto para el 0 como para el 1. Existen varias técnicas para generar secuencias de bits realmente aleatorias a partir de la salida de bits de tales generadores defectuosos; estas técnicas se denominan de-skewing (des-sesgado).

3. GENERADORES DE NÚMEROS PSEUDOALEATORIOS (PRNG)

729. Los tipos básicos de generadores de números pseudoaleatorios generalmente se basan en la utilización de registros de desplazamiento realimentados lineales y múltiples combinaciones de éstos (ya sea lineal o no linealmente); en algunos problemas matemáticos difíciles de resolver, por ejemplo, el problema de la factorización, el problema de los residuos cuadráticos y el problema del logaritmo discreto, y en las típicas funciones o primitivas criptográficas por ejemplo el AES, DES, SHA-x.

730. Los generadores de secuencias pseudoaleatorias más utilizados, para propósitos de simulación y para la implementación de algoritmos aleatorios, incluyen el Generador Congruencial Lineal (abreviado como LCG). Sin embargo la salida de un LCG es predecible por lo que no es apropiado para aplicaciones criptográficas.

731. La construcción más simple de un PRNG está basada en los registros de desplazamiento lineales realimentados (Linear Feedback Shift Registers o LFSR). Estos generadores se pueden implementar eficientemente en hardware y son capaces de generar secuencias pseudoaleatorias muy largas con una alta calidad en su distribución estadística. Aunque el generador en su forma simple no es muy seguro, debido a su linealidad, éste se utiliza típicamente como bloque básico en la construcción de generadores más complicados. Los diseños más avanzados de estos generadores, están basados en la combinación no lineal de varios generadores LFSR o mediante la utilización de uno o más LFSR que sirvan de reloj a otros LFSR (o a una combinación de éstos).

732. Otra clase de generadores pseudoaleatorios totalmente diferente son generadores pseudoaleatorios criptográficamente seguros basados en problemas difíciles de la teoría de números y la teoría de la complejidad. Típicamente utilizan la aritmética modular, lo que hace que estos generadores sean muy lentos, aunque esto puede ser parcialmente solucionado mediante la aceleración del hardware.

733. A menudo se utilizan funciones de una dirección (One Way Function o OWF) para generar secuencias de bits pseudoaleatorios. Generalmente se selecciona una semilla aleatoria s, y luego se aplica dicha función a la secuencia de valores s, s+1, s+2, … La secuencia de salida será f(s), f(s+1), f(s+2), … Dependiendo de las propiedades de la función unidireccional utilizada se necesitará guardar sólo unos pocos bits de los valores de salida a fin de eliminar las posibles correlaciones entre los valores sucesivos. Entre las funciones unidireccionales adecuadas para este tipo de generadores destacan la función SHA-2, o un sistema de cifrado en bloque como AES con clave secreta.

4. ATAQUES ESPECÍFICOS A LOS PRNG

734. Ataque criptoanalítico directo. Este ataque se lleva a cabo cuando la secuencia generada por el PRNG no es completamente indistinguible de una secuencia verdaderamente aleatoria. En este caso, un atacante, basándose en las particularidades observadas, puede deducir qué tipo de PRNG se ha utilizado y, en ciertos casos, cuál es la clave que lo gobierna. Este ataque solo es factible cuando se puede observar directamente cierta cantidad de los números de la secuencia generada, o bien se puede averiguar la secuencia indirectamente, por ejemplo cuando se hace un ataque por texto conocido a un cifrado en flujo y se recupera la secuencia pseudoaleatoria. Si el PRNG se utilizase exclusivamente

para generar claves de otros algoritmos de cifrado seguros, como podrían ser el TDEA o el AES, no sería posible deducir estas claves y por tanto seria imposible atacar al PRNG que las generó.

735. Ataques basados en la entrada. Un ataque a la entrada del PRNG se produce cuando un atacante es capaz de usar el conocimiento acerca de la secuencia de entrada al PRNG para criptoanalizarla, es decir, distinguir entre la producción del PRNG y valores aleatorios. Este ataque puede implementarse de varias formas: ataque por entrada conocida, ataque por entrada repetida y ataque por entrada elegida. Los ataques de entrada elegida pueden ser prácticos contra tarjetas inteligentes y otras manipulaciones durante un ataque criptoanalítico/físico; también pueden ser prácticos para las aplicaciones donde se usan las contraseñas de usuario, las estadísticas de red, etc. El ataque por repetición de entrada es igualmente práctico en las mismas situaciones, pero requiere menos control y un poco menos de sofisticación por parte del atacante.

736. Ataques por extensión de un estado comprometido. Este ataque intenta extender las ventajas de un ataque previo exitoso donde se ha recobrado un estado S del generador. En la práctica, es prudente asumir que un estado S del generador puede estar comprometido en un momento dado, por lo que para preservar la solidez del sistema, el PRNG debe diseñarse de modo que pueda resistir este ataque.

4.1. CONSIDERACIONES A TENER EN CUENTA CUANDO SE UTILIZAN PRNG

737. Utilizar una función resumen. Si se sospecha que un PRNG es vulnerable a ataques criptoanalíticos directos, la salida del PRNG debe ser preprocesada con una función resumen criptográfica. Esto podrá aumentar la seguridad de dicha secuencia, aunque disminuye la rapidez del generador.

738. Aplicar una función resumen a la entrada con un contador o marca de hora antes de usarla. Esto ayuda a prevenir la mayoría de los ataques por entrada elegida. Las entradas deben de concatenarse o mezclarse módulo 2 bit a bit con una marca de tiempo (time stamping) para después hacer un resumen, antes de cargarla en el PRNG.

739. Generar de vez en cuando un nuevo estado de entrada del PRNG. Para generadores PRNG como ANSI X9.17, que dejan una gran parte de su estado inalterable una vez inicializado, es conveniente generar un nuevo estado de entrada a partir del actual en el PRNG. Esto asegurará que cualquier PRNG se pueda reinicializar a sí mismo, con suficiente tiempo y entropía de entrada.

740. Se debe prestar especial atención a los puntos de inicio y semillas de los PRNG. La mejor manera de resistir todos los ataques por extensión de estado comprometido es simplemente asegurarse que el estado del generador nunca esté comprometido. Los diseñadores de sistemas deben iniciar su PRNG desde un punto inimaginable, manipulando las semillas de los PRNG inteligentemente.

4.2. ALEATORIEDAD

741. Actualmente no se dispone de ninguna prueba matemática que determine de forma categórica la propiedad de aleatoriedad de una secuencia de bits dada. El estudio y análisis de la aleatoriedad de una secuencia cualquiera se lleva a cabo mediante diferentes tests estadísticos. Cada test determina si la secuencia posee, o no, alguna propiedad que

permita considerarla como aparentemente aleatoria. Las conclusiones de los diferentes tests nunca son definitivas, pero sí probabilísticas. Si la secuencia estudiada supera todos los tests estadísticos a los que ha sido sometida, entonces se dice que la secuencia de bits es aceptada como aleatoria (aceptada porque no ha podido ser rechazada). Desde el punto de vista criptográfico es imperativo someter las secuencias pseudoaleatorias generadas a todos los análisis estadísticos posibles como paso previo a su aceptación como parte de cualquier aplicación criptográfica.

742. Un bit de secuencia aleatoria se puede interpretar como el resultado del «lanzamiento de una moneda», con lados etiquetados con 0 y 1, cada uno con una probabilidad de ½ para producir un «cero» o un «uno». Como los lanzamientos y las caras son independientes, el resultado de cualquier lanzamiento de la moneda no deberá afectar el resultado futuro. Esto es lo que se considera un generador de bits aleatorios perfecto: los valores 0 y el 1 se distribuirán aleatoria y uniformemente y todos los elementos de la secuencia se generan independientemente unos de otros. Además, el valor del próximo elemento en la secuencia no se puede predecir a pesar de la cantidad de elementos que se hayan producido hasta ese momento. Obviamente el uso de este sistema para propósitos criptográficos es inviable. Sin embargo la salida hipotética de tal generador de verdadera secuencia aleatoria idealizado sirve como referencia para la evaluación de los generadores aleatorios.

4.3. IMPREDECIBILIDAD

743. Los números pseudoaleatorios y aleatorios generados para aplicaciones criptográficas deben ser impredecibles. En el caso de los generadores pseudoaleatorios, si se desconoce la semilla, la salida del próximo número en la secuencia tendrá que ser impredecible a pesar de conocer cualquier número generado previamente. Esta propiedad se conoce como «impredecibilidad hacia adelante». Tampoco debe ser posible determinar la semilla del generador a partir del conocimiento de algunos valores generados («impredecibilidad hacia atrás»). No debe existir correlación entre la semilla y cualquier valor generado a partir de ella, es decir, cada elemento de la secuencia debe aparecer como la salida de un nuevo evento aleatorio independiente, cuya probabilidad es ½.

744. Para asegurar la impredecibilidad hacia adelante se debe ser muy cuidadoso a la hora de elegir la semilla. Los valores generados por un PRNG son completamente predecibles si se conoce la semilla y el algoritmo de generación. Como en muchos casos los algoritmos de generación son públicamente conocidos, la seguridad recae en mantener en secreto la semilla, la cual no se podrá intuir a partir de la secuencia pseudoaleatoria que ésta produce. Por tanto, la semilla por sí misma debe ser impredecible. Cuando se realizan pruebas de aleatoriedad a secuencias binarias aleatorias se asume lo siguiente:

- *Uniformidad*: en cualquier punto de la generación de una secuencia aleatoria o bits pseudo aleatorios, la ocurrencia de un cero o un uno es igualmente probable, es decir, la probabilidad de cada uno es exactamente ½. El número esperado de ceros (o unos) es $n/2$, donde n es la longitud de la secuencia.
- *Escalabilidad*: cualquier prueba aplicable a una secuencia, podrá ser aplicada también a subsecuencias extraídas aleatoriamente. Si la secuencia es aleatoria, entonces cada subsecuencia extraída también será aleatoria, por tanto cualquier subsecuencia extraída deberá también pasar el test de aleatoriedad.
- *Consistencia*: el comportamiento de un generador será consistente con el valor inicial, es decir con la semilla. Lo cual quiere decir que si la semilla es un

parámetro débil, la secuencia generada también lo será; por tanto es inadecuado comprobar un generador de secuencia pseudoaleatorio basados en la utilización de una mala semilla o a un generador aleatorio sobre la base de una salida producida a partir de una salida física poco aleatoria.

4.4. PRUEBAS ESTADÍSTICAS DE ALEATORIEDAD

745. Las aplicaciones criptográficas exigen que las secuencias pseudoaleatorias sean indistinguibles de las verdaderamente aleatorias. La suposición subyacente es que los números en la secuencia deberán estar uniformemente distribuidos y ser independientes. Para determinar esta propiedad existe un número infinito de posibilidades de pruebas estadísticas. Cada una valora la presencia o ausencia de patrones y comprueba si las estadísticas recogidas se encuentran dentro de los rangos razonables para decidir si la secuencia aparenta ser aleatoria.

746. La publicación especial del National Institute of Standards and Technology (NIST) SP 800-22 Rev. 1a, de abril de 2010 ([NIST, SP800-22]) recomienda una serie pruebas estadísticas para los generadores de números aleatorios en el campo de la criptografía. En ella se explican las pruebas de aleatoriedad especificadas para los generadores de números aleatorios y pseudoaleatorios que se pueden utilizar para muchos propósitos, incluyendo la modelización y la simulación de aplicaciones, aunque el objetivo fundamental para los que han sido diseñados y estudiados son las aplicaciones criptográficas.

747. La SP 800-22 Rev. 1a recomienda 15 pruebas estadísticas. Este conjunto de pruebas no es adecuado cuando se desconoce el generador. Todos estos tests son recomendados también por el European Telecommunications Standards Institute.

748. Existen otros tests más exigentes que la anterior normativa SP 800-22 Rev. 1a, la cual es apropiada para analizar secuencias de longitud moderada (menos de un millón de bits). Para secuencias de longitud superior hay otros tests más exigentes y apropiados, estos son los siguientes:

- Test DIEHARD: es un conjunto de 16 pruebas, desarrollado por George Marsaglia ([Marsaglia, 2002]), apropiado para secuencias de longitud de cien millones de bits o más, se puede obtener en: http://www.stat.fsu.edu/pub/diehard/.
- Tuftest ([Marsaglia and Tsang, 2002]): equivalente al anterior que con solo tres pruebas detecta todos los generadores que hubiesen fallado alguna prueba del DIEHARD, pero es aún más exigente. Se puede obtener en: http://www.jstatsoft.org/v07/i03.
- TestU01: diseñado por Pierre L'Ecuyer y Richard Simard ([L'Ecuyer and Simard, 2007]) comprende 14 pruebas. Es apropiado para analizar secuencias de más de mil millones de bits. El paquete de software está redactado en ANSI C, e incluye la implementación varios tipos de generadores de números aleatorios en forma genérica, así como muchos generadores específicos propuestos en la literatura o que se encuentran en muchos paquetes de software utilizados ampliamente. Esta batería también proporciona implementaciones generales de las pruebas clásicas de estadística para los generadores de números aleatorios, así como otras propuestas en la literatura, y algunas pruebas originales del autor. Las pruebas se pueden aplicar a los generadores predefinidos en la biblioteca y a los generadores definidos por el usuario. También se proporciona una suite de pruebas específicas, tanto para las secuencias de números aleatorios uniformes en [0, 1] como para secuencias de bits.

También se proporcionan herramientas básicas para trazar algunos gráficos de vectores de puntos producidos por los generadores. Se puede obtener en: http://www.iro.umontreal.ca/~simardr/testu01/tu01.html. o bien: http://www.iro.umontreal.ca/~lecuyer/.

ANEXO F. REFERENCIAS

[Adams et al., 2001] C. Adams, P. Cain, D. Pinkas, and R. Zuccherato, Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP), Request for Comments RFC 3161, August 2001

[Armknecht, 2002] F. Armknecht, A Linealization Attack on the Bluetooth Key Stream Generator. *IACR e-print archive*, 2002/191.

[ANSI, X9.30-1] American National Standards Institute, *Public key cryptography using irreversible algorithms-Part 1: The Digital Signature Algorithm (DSA)*, ANSI X9.30-1, 1997.

[ANSI, X9.42] American National Standards Institute, *Public key cryptography for the financial services industry: Agreement of symmetric keys using discrete logarithm cryptography*, ANSI X9.42, 2003.

[ANSI, X9.44] American National Standards Institute, *Key establishment using integer factorization cryptography*, ANSI X9.44, 2007.

[ANSI, X9.62] American National Standards Institute, *Public key cryptography for the financial services industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*, ANSI X9.62, 2005.

[ANSI, X9.63] American National Standards Institute, *Public key cryptography for the financial services industry: Key agreement and key transport using elliptic curve cryptography*, ANSI X9.63, 2001.

[ANSI, X9.71] American National Standards Institute, *Keyed hash message authentication code*, ANSI X9.71, 2001.

[Anderson and Biham, 1996] R. Anderson and E. Biham, Tiger: A fast new hash function, *Lecture Notes in Comput. Sci.,*1039 (1996), 89-97.

[Bach and Shallit, 1996] E. Bach and J. Shallit, *Algorithmic number theory, Volume I: Efficient algorithms*, The MIT Press, Cambridge, MA, 1996.

[Benaloh and de Mare, 1993] J. Benaloh and M. de Mare, One-way accumulators: A decentralized alternative to digital signatures, *Lecture Notes in Comput. Sci*., 765 (1993), 274-285.

[Beth and Piper, 1984] T. Beth and F.C. Piper, The Stop-and-Go Generator, *Lecture Notes in Comput. Sci.,* 209 (1984), 88-92.

[Biham and Shamir, 1993] E. Biham and A. Shamir, *Differential cryptanalysis of the full* 16-*round DES*, Proc. of Crypto'92, Springer Verlag, 1993.

[Biryukov et al., 2000] A. Biryukov, A. Shamir and D. Wagner, Real-Time Cryptanalysis of A5/1 on a PC, *Lecture Notes in Comput. Sci.,* 1978 (2000), 127-139.

[Biryukov et al. 2009] A. Biryukov, D. Khovratovich, and I. Nikoli, Distinguisher and related-key attack on the full AES-256, *Lecture Notes in Comput. Sci*., 5677 (2009), 231-249.

[Blake et al., 1999] I. Blake, G. Seroussi, and N. Smart, *Elliptic curves in cryptography*, Cambridge University Press, Cambridge, 1999.

[Blake et al., 2005] I. Blake, G. Seroussi, and N. Smart, *Advances in elliptic curve cryptography*., Cambridge University Press, Cambridge, 2005.

[Blakley and Borosh, 1979] G.R. Blakley and I. Borosh, Rivest-Shamir-Adleman public key cryptosystems do not always conceal messages, *Comp. Math. Appl.*, 5 (1979), no. 3, 169-178.

[Bleichenbacher and May, 2006] D. Bleichenbacher and A. May, New attacks on RSA with small secret CRT-exponents, *Lecture Notes in Comput. Sci.*, 3958 (2006), 1-13.

[Blömer and May, 2001] J. Blömer and A. May, Low secret exponent RSA revisited, *Lecture Notes in Comput. Sci.*, 2146 (2001), 4-19.

[Boneh, 1999] D. Boneh, Twenty years of attacks on the RSA cryptosystem, *Notices Amer. Math. Soc.*, 46 (1999), no. 2, 203-213.

[Boneh and Durfee, 1999] D. Boneh and G. Durfee, Cryptanalysis of RSA with private key d less tan $n^{0.292}$, *Lecture Notes in Comput. Sci.*, 1592 (1999), 1-11.

[Boneh and Durfee, 2000] D. Boneh and G. Durfee, Cryptanalysis of RSA with private key d less tan $n^{0.292}$. *IEEE Trans. Inform. Theory*, 46 (2000), no. 4, 1339-1349.

[Boneh and Shacham, 2002] D. Boneh and H. Shacham, Fast variants of RSA, *CryptoBytes*, 5 (2002), no. 1, 1-9.

[Brainpool, 2005] Brainpool, ECC Brainpool standard curves and curve generation. Ver. 1.0. 2005, http://www.ecc-brainpool.org/download/Domain-parameters.pdf

[Brickell and Odlyzko, 1988] E.F. Brickell and A.M. Odlyzko, Cryptanalysis: A Survey of Recent Results, *Proc. IEEE,* 76 (1988), 578-593.

[Buchmann and Williams, 1988] J. Buchmann and H.C. Williams, A key-exchange system based on imaginary quadratic fields, *J. Cryptology* 1 (1988), no. 2, 107-118.

[Buhler et al., 1993] J.P. Buhler, H.W. Lenstra, Jr., Carl Pomerance, Factoring integers with the number field sieve, The development of the number field sieve, *Lecture Notes in Math.*, 1554 (1993), 50-94.

[Buldas et al., 1998] A. Buldas, P. Laud, H. Lipmaa, and J. Villemson, Time-stamping with binary linking schemes, *Lecture Notes in Comput. Sci.*, 1462 (1998), 486-501.

[Burrows et al., 1993] M. Burrows, M. Abadi, and R. Needham, A logic Authentication, *ACM Transactions on Computer Systems*, 8 (1990), 18-36.

[BSI, 2009] Bundesamt Für Sicherheit in der Informationstechnik (BSI), *Elliptic curve cryptography*, version 1.11, TR-03111, 2009.

[Chen et al., 2004] C.Y. Chen, C.Y. Ku, and D. C. Yen, Cryptanalysis of short secret exponents modulo RSA primes, *Inform. Sci.*, 160 (2004), no. 1-4, 225-233.

[Cheon, 2010], J.H. Cheon, Discrete logarithm problems with auxiliary inputs, *J. Cryptology*, 23 (2010), 457-476.

[Chmielowiec, 2010] A. Chmielowiec, Fixed points of the RSA encryption algorithm, *Theoret. Comput. Sci.*, 411 (2010), 288-292.

[Cohen, 1993] H. Cohen, *A course in computational algebraic number theory*, Springer-Verlag, Berlin, 1993.

[Cohen et al., 2006] H. Cohen, G. Frey, et al., *Handbook of elliptic and hyperelliptic curve cryptography*, Chapman & Hall/CRC, Boca Raton, FL, 2006.

[Coppersmith, 1994] D. Coppersmith, The Shrinking Generator, *Lecture Notes in Comput. Sci.,*773 (1994), 22-39.

[Coppersmith et al., 1996] D. Coppersmith, M. Franklin, J. Patarin, and M. Reiter, Low-exponent RSA with related messages, *Lecture Notes in Comput. Sci.*, 1070 (1996), 1-9.

[Coron and May, 2007] J.S. Coron, A. May, Deterministic polynomial-time equivalence of computing the RSA secret key and factoring, *J. Cryptology*, 20 (2007), no. 1, 39-50.

[Daemen and Assche, 2007] J. Daemen and G. van Assche, Producing collisions for Panama, instantaneously, *Lecture Notes in Comput. Sci.*, 4593 (2007), 1-18.

[Daemen and Clapp, 1998] J. Daemen and C. Clapp, Fast hashing and stream encryption with Panama, *Lecture Notes in Comput. Sci.*, 1372 (1998), 60-74.

[Damgård, 1990] I. Damgård, A design principle for hash functions, *Lecture Notes in Comput. Sci.*, 435 (1990), 416-427.

[de Weger, 2002] B. de Weger, Cryptanalysis of RSA with small prime difference, *Appl. Algebra Engrg. Comm. Comput.*, 13 (2002), no. 1, 17-28.

[Diffie and Hellman, 1976] W. Diffie and M.E. Hellman, New directions in cryptography, *IEEE Trans. Information Theory* IT-22 (1976), no. 6, 644-654.

[Diffie et al., 1992] W. Diffie, P. van Oorschot, and M. Wiener, Authentication and authenticated key exchanges, *Des. Codes Cryptography*, 2 (1992), 107-125.

[Dobbertin et al., 1996] H. Dobbertin, A. Bosselaers, and B. Preneel, RIPEMD-160: A strengthened version of RIPEMD, Lecture Notes in Computer Science 1039 (1996), 71–82.

[Dujella, 2009] A. Dujella, A variant of Wiener's attack on RSA, *Computing*, 85 (2009), 77-83.

[Durfee and Nguyen, 2000] G. Durfee and P.Q. Nguyen, Cryptanalysis of the RSA schemes with short secret exponent from Asiacrypt '99, *Lecture Notes in Comput. Sci.*, 1976 (2000), 14-29.

[Durán et al., 2005] R. Durán Díaz, L. Hernández Encinas y J. Muñoz Masqué, *El Criptosistema RSA*, RA-MA, Madrid, 2005.

[ECRYPT II, 2009] ECRYPT II - The eSTREAM Portfolio, 2009 Annual Update, 2009. http://www.ecrypt.eu.org/stream/D.SYM.3-v1.1.pdf

[EFF, 1998] Electronic Frontier Foundation. *Cracking DES*, O Reily, 1998.

[ElGamal, 1985] T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Trans. Inform. Theory,* 31 (1985), 469-472.

[Enge, 1999] A. Enge, *Elliptic curves and their applications to cryptography: An introduction*, Kluwer Academic Publishers, Boston, MA, 1999.

[eSTREAM, 2008] eSTREAM-The ECRYPT Stream Cipher Project, last updated September 8, 2008. http://www.ecrypt.eu.org/stream/

[ETSI, TS-102023] European Telecommunications Standards Institute, Technical Specification TS 102 023, V1.1.1, *Policy requirements for Time-Stamping Authorities*, April 2002.

[ETSI, TS-101733] European Telecommunications Standards Institute, Technical Specification TS 101 733, V1.5.1, *Electronic Signatures and Infrastructures (ESI), Electronic Signature Formats*, December 2003.

[Ferguson et al., 2000] N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner, and D. Whiting, Improved Cryptanalysis of Rijndael, *Lecture Notes in Comput. Sci*., 1978 (200), 213-230.

[Frey, 2001] G. Frey, *Applications of arithmetical geometry to cryptographic constructions*, Proc. of the Fifth International Conference on Finite Fields and Applications, 128-161, Springer, 2001.

[Frey and Rück, 1994] G. Frey and H. Ruck, A remark concerning *m*-divisibility and the discrete logarithm in the divisor class group of curves, *Math. Comp.*, 62 (1994), 865-874.

[Fúster and Caballero, 2008] A. Fúster and P. Caballero, Strategic Attack on the Shrinking Generator, *Theoretical Computer Science*, 409, 3, (2008), 530-536.

[Fúster and Pazo-Robles, 2009] A. Fúster and M.E. Pazo-Robles, Tendencias actuales en cifrado en flujo: The eSTREAM Project, Nuevos Avances en Criptografía y Seguridad de la Información, Sesión especial del Congreso de la Real Sociedad Matemática Española 2009, Oviedo, Febrero 2009. *Ediciones y Publicaciones de la Universitat de Lleida*, 53-62.

[Fúster et al., 2004] A. Fúster Sabater, D. de la Guía Martínez, L. Hernández Encinas, F. Montoya Vitini y J. Muñoz Masqué, *Técnicas criptográficas de protección de datos*, RA-MA, 3ª ed., Madrid, 2004.

[Gaudry et al., 2002] P. Gaudry, F. Hess, and N. Smart, Constructive and destructive facets of Weil descent on elliptic curves, *J. Cryptology*, 15 (2002), 19-46.

[Golomb, 1982] S.W. Golomb, *Shift Register Sequences*, Aegean Park Press, Laguna Hills, California, 1982.

[Gysin and Seberry, 1999] M. Gysin, and J. Seberry, Generalized cycling attacks on RSA and strong RSA primes, *Lecture Notes in Comput. Sci.*, 1587 (1999), 149-163.

[Haber and Stornetta, 1991] S. Haber and W.S. Stornetta, How to time-stamp a digital document, *Journal of Cryptology* 3, No. 2 (1991), 99-111.

[Hankerson et al., 2004] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to elliptic curve cryptography*, Springer-Verlag, New York, 2004.

[Hastad, 1988] J. Hastad, Solving simultaneous modular equations of low degree, *SIAM J. Computing*, 17 (1988), no. 2, 336-341.

[Hinek, 2008] M.J. Hinek, On the security of multi-prime RSA, *J. Math. Cryptol.*, 2 (2008), no. 2, 117-147.

[IEEE, 1363] Institute of Electrical and Electronics Engineers, *Standard specifications for public key cryptography*, IEEE 1363, 2000.

[IEEE, 1363a] Institute of Electrical and Electronics Engineers, *Standard specifications for public key cryptography, Amendment 1: Additional techniques*, IEEE 1363a, 2004.

[ISOIEC, 9798-2] International Organization for Standardization/International Electrotechnical Commission, *Information technology-Security techniques-Entity authentication-Part 2: Mechanisms using symmetric encipherment algorithms*. ISO/IEC 9798-2, 1999.

[ISOIEC, 9798-4] International Organization for Standardization/International Electrotechnical Commission, *Information technology-Security techniques-Entity authentication-Part 4: Mechanisms using a cryptographic check function.* ISO/IEC 9798-4, 1999.

[ISOIEC, 18014] International Organization for Standardization/International Electrotechnical Commission, *Information technology-Security techniques-Time-stamping services*, ISO/IEC 18014, 2009.

[ISOIEC, 18033-2] International Organization for Standardization/International Electrotechnical Commission, *Information technology-Security techniques-Encryption algorithms-Part 2: Asymmetric ciphers*, ISO/IEC 18033-2, 2006.

[ISOIEC, 18033-3] International Organization for Standardization/International Electrotechnical Commission, *Information technology-Security techniques-Encryption algorithms-Part 4: Stream ciphers*, ISO/IEC 18033-3, 2005.

[ISOIEC, 18033-3R] International Organization for Standardization/International Electrotechnical Commission, *Information technology-Security techniques-Encryption algorithms-Part 3: Block ciphers*. ISO/IEC 18033-3, Rev-2010.

[Jacobson et al., 2001] M. Jacobson, A. Menezes, and A. Stein, Solving elliptic curve discrete logarithm problems using Weil descent, *J. Ramanujan Math. Soc.*, 16 (2001), 231-260.

[Jochemsz and May, 2007] E. Jochemsz and A. May, A polynomial time attack on RSA with private CRT-exponents smaller than $n^{0.073}$, *Lecture Notes in Comput. Sci.*, 4622 (2007), 395-411.

[Kahn, 1967] D. Khan, *The Codebreakers: The Story of Secret Writing*, Macmillan Pub. Co., New York, 1967.

[Katz and Lindell, 2008] J. Katz and Y. Lindell, *Introduction to modern cryptography*, Chapman & Hall/CRC, Boca Raton, FL, 2008.

[Khadir, 2008] O. Khadir, Algorithm for factoring some RSA and Rabin moduli, *J. Discrete Math. Sci. Cryptogr.*, 11 (2008), no. 5, 537-543.

[Kleinjung et al., 2010] T. Kleinjung, K. Aoki, J. Franke, A.K. Lenstra, E. Thomé, J.W. Bos, P. Gaudry, A. Kruppa, P.L. Montgomery, D.A. Osvik, H. te Riele, A. Timofeev, and P. Zimmermann, Factorization of a 768-bit RSA modulus, version 1.3, http://eprint.iacr.org/2010/006.pdf.

[Koblitz, 1987] N. Koblitz, Elliptic curve cryptosystems, *Math. Comp.* 48 (1987), no. 177, 203-209.

[Koblitz, 1989] N. Koblitz, Hyperelliptic cryptosystems, *J. Cryptology* 1 (1989), no. 3, 139-150.

[L'Ecuyer and Simard, 2007] P. L'Ecuyer and R. Simard, Testu01: A C library for empirical testing of random number generators, *ACM Transactions on Mathematical Software*, 2007.

[Lenstra, 1993] H.W. Lenstra, Jr., The number field sieve: An annotated bibliography, The development of the number field sieve, *Lecture Notes in Math.*, 1554 (1993), 1-3.

[Lenstra et al., 1993] A.K. Lenstra, H. W., Lenstra, Jr., M. S. Manasse, and J. M. Pollard, The number field sieve, The development of the number field sieve, *Lecture Notes in Math.*, 1554 (1993), 11-42.

[Lochter and Merkle, 2010] M. Lochter and J. Merkle, RFC 5639-Elliptic Curve Cryptography (ECC) Brainpool Standard. RFC Archives, 2010. http://www.faqs.org/rfcs/rfc5639.html.

[Luo et al., 2009] P. Luo, H. Zhou, D. Wang, and Y. Dai, Cryptanalysis of RSA for special case with $d > e$, *Sci. China Ser F-Inf. Sci.*, 52 (2009), no. 4, 809-616.

[Maitra and Sarkar, 2008] S. Maitra and S. Sarkar, A new class of weak encryption exponents in RSA, *Lecture Notes in Comput. Sci.*, 5365 (2008), 337-349.

[Marsaglia, 2002] G. Marsaglia, Diehard, a battery of tests for RNGS. http://stat.fsu.edu/geo/diehard.html, 2002.

[Marsaglia and Tsang, 2002] G. Marsaglia and W.W. Tsang, Some difficult-to-pass tests of randomness, *Journal of Statistical Software*, 7, 3 (2002), 1-9.

[Massey, 1969] J.L. Massey, Shift-register synthesis and BCH decoding, *IEEE Trans. on Informat Theory*, 15 (1969), 122-127.

[Massias and Quisquater, 1997] H. Massias and J. Quisquater, Time and cryptography, Technical report, TIMESEC project, Université catholique de Louvain, 1-19, 1997.

[Massias et al., 1999] H. Massias, X. Serret, and J. Quisquater, Timestamps: Main issues on their use and implementation, *In Proceedings of IEEE 8th International Workshops on enabling Technologies* (1999), 178-183.

[Matsumoto et al., 1986] T. Matsumoto, Y. Takashima, and H. Imai, On seeking smart public-key distribution systems, *The Transactions of the IECE of Japan*, E69 (1986), 99-106.

[Maurer, 1994] U.M. Maurer, Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms, *Lecture Notes in Comput. Sci.*, 839 (1994), 271-281.

[Maurer and Wolf, 1996] U.M. Maurer and S. Wolf, Diffie-Hellman oracles, *Lecture Notes in Comput. Sci.*, 1109 (1996), 268-282.

[Maurer and Wolf, 1999] U.M. Maurer and S. Wolf, The relationship between breaking the Diffie-Hellman protocol and computing discrete logarithms, *SIAM J. Comput.*, 28 (1999), no. 5, 1689-1721.

[Maurer and Wolf, 2000] U.M. Maurer and S. Wolf, The Diffie-Hellman protocol. Towards a quarter-century of public key cryptography, *Des. Codes Cryptography*, 19 (2000), no. 2-3, 147-171.

[Maurer et al., 2002] M. Maurer, A. Menezes, and E. Teske, Analysis of the GHS Weil descent attack on the ECDLP over characteristic two finite fields of composite degree, *LMS J. Comput. Math.*, 5 (2002), 127-174.

[May, 2002] A. May, Cryptanalysis of unbalanced RSA with small CRT-exponent, *Lecture Notes in Comput. Sci.*, 2442 (2002), 242-256.

[McCurley, 1988] K.S. McCurley, A key distribution system equivalent to factoring, *J. Cryptology* 1 (1988), no. 2, 95-105.

[Menezes, 1993] A.J. Menezes, *Elliptic curve public key cryptosystems*, Kluwer Academic Publishers, Boston, 1993.

[Menezes and Qu, 2001] A. Menezes and M. Qu, Analysis of the Weil descent attack of Gaudry, Hess and Smart, *Lecture Notes in Comput. Sci.*, 2020 (2001), 308-318.

[Menezes et al., 1993a] A.J. Menezes (editor), *Applications of finite fields*, Kluwer Academic Publishers, Boston, 1993.

[Menezes et al., 1993b] A. Menezes, W. Okamoto, and S. Vanstone, Reducing elliptic curve logarithms to logarithms in a finite field, *IEEE Trans. Inform. Theory*, 39 (1993), 1639-1646.

[Menezes et al., 1995] A. Menezes, M. Qu, and S. Vanstone, *Some new key agreement protocols providing implicit authentication*, Proc. of Workshops on Selected Areas in Cryptography (1995), 22-32.

[Menezes et al., 1997] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of applied cryptography*, CRC Press, Boca Raton, FL, 1997.

[Merkle, 1979] R. Merkle, Secrecy, authentication, and public key systems, Ph.D. dissertation, Dept. of Electrical Engineering, Stanford University, 1979.

[Merkle, 1990] R.C. Merkle, One way hash functions and DES, *Lecture Notes in Comput. Sci.,*435 (1990), 428-446.

[Miller, 1986] V.S. Miller, Use of elliptic curves in cryptography, *Lecture Notes in Comput. Sci.*, 218 (1986), 417-426.

[Miller, 1976] G.L. Miller, Riemann's hypothesis and tests for primality, Working papers presented at the ACM-SIGACT Symposium on the Theory of Computing (Albuquerque, N.M., 1975), *J. Comput. System Sci.*, 13 (1976), no. 3, 300-317.

[Needham and Schroeder, 1987] R. Needham and M. Schroeder, Using Encryption for Authentication in Large Networks of Computers, *Communications of the ACM*, 21 (1987), 7.

[NIST, FIPS46-1] National Institute of Standards and Technology, *Data encryption standard*, Federal Information Processing Standard Publication, FIPS 46-1, 1988.

[NIST, FIPS46-2] National Institute of Standards and Technology, *Secure hash standard*, Federal Information Processing Standard Publication, FIPS 46-2, 1993.

[NIST, FIPS46-3] National Institute of Standards and Technology, *Data encription standard reafirmed*, Federal Information Processing Standard Publication, FIPS 46-3, 1999.

[NIST, FIPS180-1] National Institute of Standards and Technology, *Secure hash standard*, Federal Information Processing Standard Publication, FIPS 180-1, 1995.

[NIST, FIPS180-2] National Institute of Standards and Technology, *Secure hash standard (SHS)*, Federal Information Processing Standard Publication, FIPS 180-2, 2002.

[NIST, FIPS186-2] National Institute of Standards and Technology, *Digital signature standard (DSS)*, Federal Information Processing Standard Publication, FIPS 186-2, 2000.

[NIST, FIPS186-3] National Institute of Standards and Technology, *Digital signature standard (DSS)*, Federal Information Processing Standard Publication, FIPS 186-3, 2009.

[NIST, FIPS197] National Institute of Standards and Technology, *Advanced Encryption Standard (AES)*, Federal Information Processing Standard Publication 197, 2001.

[NIST, FIPS198] National Institute of Standards and Technology, *The keyed-hash message authentication code (HMAC)*, Federal Information Processing Standard Publication, FIPS 198, 2002.

[NIST, SP800-20] National Institute of Standards and Technology, *Modes of operation validation system for the triple data encryption algorithm (TMOVS): Requirements and procedures*, Special Publication, SP 800-20, 2000.

[NIST, SP800-22] National Institute of Standards and Technology, *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*, Special Publication SP 800-22 Revision 1a, 2010.

[NIST, SP800-38A] National Institute of Standards and Technology, *Recommendation for block cipher modes of operation: Methods and techniques*, Special Publication, SP 800-38A, 2001.

[NIST, SP800-38B] National Institute of Standards and Technology, *Recommendation for block cipher modes of operation: The CMAC mode for authentication*, Special Publication, SP 800-38B, 2005.

[NIST, SP800-38C] National Institute of Standards and Technology, *Recommendation for block cipher modes of operation: The CCM mode for authentication and confidentiality,* Special Publication, SP 800-38C, 2004.

[NIST, SP800-38D] National Institute of Standards and Technology, *Recommendation for block cipher modes of operation: Galois/counter mode (GCM) and GMAC,* Special Publication, SP 800-38D, 2007.

[NIST, SP800-38E] National Institute of Standards and Technology, *Recommendation for block cipher modes of operation: The XTS-AES mode for confidentiality on storage devices,* Special Publication, SP 800-38E, 2010.

[NIST, SP800-57] National Institute of Standards and Technology, *Recommendation for key management–Part 1: General (Revised)*, Special Publication, SP 800-57, 2007.

[NIST, SP800-57A] National Institute of Standards and Technology, *Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography (Revised)*, Special Publication, SP 800-57A, 2007.

[NIST, SP800-67] National Institute of Standards and Technology, *Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher*, Special Publication, SP 800-67 Version 1.1, 2008.

[NIST, SHA-3] National Institute of Standards and Technology, *Cryptographic hash algorithm competition*, http://csrc.nist.gov/groups/ST/hash/sha-3/index.html

[NSA, SuiteB] National Security Agency, *Suite B cryptography*, 2005.

[Nyberg and Rueppel, 1995] K. Nyberg and R. Rueppel, Message recovery for signature schemes based on the discrete logarithm problem, *Lecture Notes in Comput. Sci.*, 950 (1995), 182-193.

[Okeya and Takagi, 2006] K. Okeya and T. Takagi, Security analysis of CRT-based cryptosystems, *Int. J. Inf. Secur.*, 5 (2006), no. 3, 177-185.

[Otway and Rees, 1987] D. Otway and O. Rees, Efficient and Timely Mutual Authentication, *Operating Systems Review*, 21 (1987), 8-10.

[Peterson and Brown, 1961] W.W. Peterson and D.T. Brown, Cyclic codes for error detection, *Proceedings of the IRE*, 49 (1961), no. 1, 228-235.

[Pinkas et al., 2003] D. Pinkas, N. Pope, and J. Ross, Requirements for Time-Stamping Authorities, Request for Comments RFC 3628, 2003.

[Pohlig and Hellman, 1978] S.C. Pohlig and M.E. Hellman, An improved algorithm for computing logarithms over *GF*(*p*) and its cryptographic significance, *IEEE Trans. Inform. Theory*, IT-24 (1978), 106-110

[Pollard, 1974] J.M. Pollard, Theorems on factorization and primality testing, *Math. Proc. Cambridge Philos. Soc.*, 76 (1974), 521-528.

[Quisquater and Couvreur, 1982] J.J. Quisquater and J.M. Couvreur, Fast decipherment algorithm for RSA public-key cryptosystem, *Electron. Lett.*, 17 (1982), no. 21, 905-907.

[Rijmen et al., 2002] V. Rijmen, B. van Rompay, B. Preneel, and J. Vandewalle, Producing collisions for Panama, *Lecture Notes in Comput. Sci.*, 2355 (2002), 259-275.

[Rivest, 1991] R.L. Rivest, The MD4 message digest algorithm, *Lecture Notes in Comput. Sci.*, 537 (1991), 303-311.

[Rivest, 1992] R.L. Rivest, *RFC 1321: The MD5 message-digest algorithm*, Internet Request for Comments 1321, April, 1992, Rump session of Crypto'91.

[Rivest et al., 1978] R.L. Rivest, A. Shamir, and L.M. Adleman, A method for obtaining digital signatures and public-key cryptosystems, *Commun. ACM*, 21 (1978), no. 2, 120-126.

[RSALab, 1993] RSA Laboratories, *PKCS #3, Diffie-Hellman key agreement standard*, version 1.4, 1993. http://www.rsa.com/rsalabs/node.asp?id=2126

[RSALab, 2002] RSA Laboratories, *PKCS #3, RSA cryptography standard*, version 2.1, 2002. http://www.rsa.com/rsalabs/node.asp?id=2125.

[Rueppel and Oorschot, 1994] R. Rueppel and P. van Oorschot, Modern key agreement techniques, *Comput. Commun.*, 17 (1994), 458-465.

[Salomaa, 1990] A. Salomaa, *Public-key cryptography*, Springer-Verlag, EATCS Monographs on Theoretical Computer Science, 23, Springer-Verlag, Berlin, 1990.

[Sarkar and Maitra, 2009a] S. Sarkar and S. Maitra, Further results on implicit factoring in polynomial time, *Adv. Math. Commun.*, 3 (2009), no. 2, 205-217.

[Sarkar and Maitra, 2009b] S. Sarkar and S. Maitra, Partial key exposure attacks on RSA and its variant by guessing a few bits of one of the prime factors, *Bull. Korean Math. Soc.*, 46 (2009), no. 4, 721-741.

[Sarkar and Maitra, 2010] S. Sarkar and S. Maitra, Cryptanalysis of RSA with two encryption exponents, *Inf. Processing Lett.*, 10 (2010), 178-181.

[Satoh and Araki, 1998] T. Satoh and K. Araki, Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves, *Comment. Math. Univ. St. Paul*, 47 (1998), 81-92.

[Semaev, 1998] I.A. Semaev, Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p, *Math. Comp.*, 67 (1998), 353-356.

[Sgarro, 1990] A. Sgarro, *Códigos Secretos*, Editorial Pirámide, Madrid, 1990.

[Shannon, 1949] C.E. Shannon, Communication theory of secrecy systems, *Bell. Syst. Tech. J.*, 28 (1949), 656-715.

[Silverman, 1994] J.H. Silverman, *Advanced topics in the arithmetic of elliptic curves*, Springer-Verlag, New York, 1994.

[Silverman, 2009] J.H. Silverman, *The arithmetic of elliptic curves*, 2nd ed., Springer-Verlag, New York, 2009.

[Smart, 1999] N.P. Smart, The discrete logarithm problem on elliptic curves of trace one, *J. Cryptology*, 12 (1999), 193-196.

[Simpson et al, 1998] L. Simpson, J. Golic and E. Dawson, Correlation Attack on the Shrinking Generator, *Lecture Notes in Comput. Sci.,*1438 (1998), 147-158.

[SECG, SEC1] Standards for Efficient Cryptography Group, *SEC 1: Elliptic curve cryptography*, version 2.0, 2009. http://www.secg.org/download/aid-773/sec1_1point9.pdf

[Stinson, 2006] D.R. Stinson, *Cryptography: Theory and practice*, Chapman & Hall/CRC, 3rd ed., Boca Raton, FL, 2006.

[Sun and Wu, 2005] H.M. Sun and M.E. Wu, *Design of rebalanced RSA-CRT for fast encryption*, Proc. of Information Security Conference, 2005, pp. 16-27.

[Sun et al., 2005] H.M. Sun, J. Hinek, and M.E. Wu, *On the design of rebalanced RSA-CRT*, Technical Report CACR 2005-35, University of Waterloo.

[Tilborg, 1988] H.C. Tilborg, *An Introduction to Cryptology*, Kluwer Academic Publisher, Boston, 1988.

[Trivium, 2008] The eSTREAM Project, eSTREAM Phase 3 Stream Cipher, Trivium, 2008. http://www.ecrypt.eu.org/stream/triviump3.html

[Verheul and van Tilborg, 1997] E.R. Verheul and H.C.A. van Tilborg, Cryptanalysis of "less short" RSA secret exponents, *Appl. Algebra Engrg. Comm. Comput.*, 8 (1997), no. 5, 425-435.

[Wang et al., 2004] X. Wang, D. Feng, X. Lai, and H. Yu, *Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD*, http://eprint.iacr.org/2004/199.pdf, 2004.

[Wang et al., 2005] X. Wang, Y.L. Yin, and H. Yu, Finding collisions in the full SHA-1, *Lecture Notes in Comput. Sci.*, 3621 (2005), 17-36.

[Wang and Yu, 2005] X. Wang and H. Yu, How to break MD5 and other hash functions, *Lecture Notes in Comput. Sci.*, 3494 (2005), 19-35.

[Wang et al., 2004] X. Wang, D. Feng, X. Lai, and H. Yu, Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD http://eprint.iacr.org/2004/199.pdf, 2004.

[Wiener, 1990] M.J. Wiener, Cryptanalysis of short RSA secret exponents, *IEEE Trans. Inform. Theory*, 36 (1990), no. 3, 553-558.

[Williams, 1982] H.C. Williams, A p+1 method of factoring, *Math. Comp.*, 39 (1982), no. 159, 225-234.