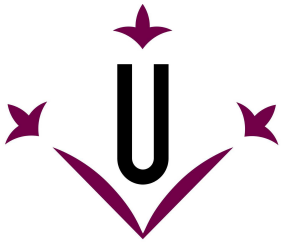
Universitat de Lleida

Protocolos de seguridad para sistemas de identificación por radiofrecuencia

Santi Martínez Rodríguez



Universitat de Lleida
Escola Politècnica Superior
Departament de Matemàtica

PROTOCOLOS DE SEGURIDAD PARA SISTEMAS DE IDENTIFICACIÓN POR RADIOFRECUENCIA

Memoria presentada para aspirar al grado
de Doctor por la Universitat de Lleida

Programa de doctorat en Enginyeria

Santi Martínez Rodríguez

Directoras:
Concepció Roig Mateu
Magda Valls Marsal

Marzo de 2011

«Es preciso que te percates de todo: tanto del corazón sin temblor de la redonda verdad como de los pareceres de los mortales, en los que no hay verdadera solidez.»

Parménides de Elea – *Sobre la naturaleza*

«Die lebensrettende Funktion der Verdrängung in der Kindheit verwandelt sich später beim Erwachsenen in eine lebenszerstörende Macht.»

Alice Miller – *Das verbannte Wissen*

«Intelligence is the ability to avoid doing work, yet getting the work done.»

Linus Torvalds

Agradecimientos

A mis directoras de tesis: Concepció Roig y Magda Valls, por su inestimable ayuda durante todo este tiempo, y por su enorme paciencia ante mi propia parsimonia.

A Francesc Giné, Josep M. Miret y Ramiro Moreno, por sus aportaciones a esta tesis, como coautores de algunos de mis artículos.

A Jordi Pujolàs, Francesc Sebé, Francesc Solsona y al resto de miembros del seminario de criptoparalelismo, por las interesantes sesiones de lluvia de ideas.

A Daniel Sadornil y Juan Tena por sus frecuentes y productivas visitas a la Universitat de Lleida.

Al resto de compañeros de la EPS, tanto profesores como PAS, especialmente a los de los departamentos de matemática e informática.

A mi mujer, Rosana, por su paciencia, comprensión y por los ánimos brindados en los momentos de máximo estrés.

A mis padres, familia y familia política por sobrellevar que haya sido como un mueble estos últimos meses.

A todos los amigos que he atendido menos de lo que merecen, especialmente a Ioannis y Rubén.

A todos los profesores que me hicieron amar la ciencia y también a aquellos que me hicieron aborrecer las letras, porque entre todos me han guiado hasta este momento.

Al gato Tuno, por ayudarme a liberar el estrés.

Y a todos aquellos que cuando imprima esta tesis me percate de que he olvidado incluirlos.

Resumen

En el campo de la identificación automática de objetos existe un interés creciente, en el sector industrial, por la implantación de sistemas de identificación remota por radiofrecuencia. La identificación por radiofrecuencia RFID (*Radio-Frequency IDentification*) es un método automático de identificación donde los objetos están provistos de unas etiquetas que disponen de circuito integrado y antena, que permiten su identificación a distancia.

Las consultas son generadas por vía inalámbrica por parte de un dispositivo llamado lector RFID. Debido a que estas consultas son inalámbricas, nos encontramos con el problema de que cualquiera podría intentar espiar el canal de comunicación, o introducir en él sus propios mensajes. Además, se debe considerar la posibilidad de que alguna de las etiquetas pueda caer en malas manos, por lo que una etiqueta nunca deberá contener información (en claro) que pueda comprometer al sistema.

Así pues, para hacer uso de la gran potencialidad que ofrece la identificación por radiofrecuencia, es necesario dotar a los sistemas RFID de los mecanismos de seguridad apropiados que eviten los problemas señalados.

En esta tesis, se proponen protocolos de seguridad, aplicados a RFID, que actúan en dos niveles: la capa de aplicación y la capa de comunicación.

Para la capa de aplicación, se propone un protocolo seguro para la identificación de etiquetas, basado en el uso de criptografía de curvas elípticas y en un protocolo de conocimiento nulo.

Con el fin de facilitar el cálculo de curvas elípticas útiles, se desarrolla un mecanismo paralelo y automático que genera dichas curvas en un tiempo eficiente.

Finalmente, para la capa de comunicación, se propone un protocolo de control de acceso al medio que evita que un atacante pueda contar el número de etiquetas presentes en el entorno. Este problema tiene sentido en entornos en los que se incluyan etiquetas RFID en productos clave, como podrían ser los billetes de curso legal.

Abstract

The use of remote object identification, based on radio-frequency technology, has received an increasing attention from the industrial sector. Radio-Frequency IDentification (or simply RFID) is an automatic identification method where an RFID tag is attached to each object. These tags are equipped with integrated circuit and antenna.

Queries are generated over the air by a device called an RFID reader. Since these queries are wireless, it arises the problem that anyone might try to eavesdrop on the communication channel, or insert his own messages in it. In addition, the possibility that some of the tags may fall into the wrong hands must be considered. Thus, tags should never contain information (in clear) that could compromise the system.

Therefore, in order to make use of the great potential offered by radio-frequency identification, it is necessary to provide RFID systems with the appropriate security mechanisms to avoid these problems.

In this thesis, security protocols applied to RFID are proposed. These protocols operate at two levels: the application layer and the communication layer.

For the application layer, a secure protocol for tag identification is proposed, based on the use of elliptic curve cryptography and a zero knowledge protocol.

To facilitate the computation of useful elliptic curves, a parallel and automatic mechanism that generates these curves in efficient time is developed.

Finally, a Medium Access Control protocol for the communication layer is presented. This protocol prevents an attacker from counting the number of tags in the environment. This problem makes sense in environments that include RFID tags in key products, such as legal banknotes.

Resum

En el camp de la identificació automàtica d'objectes existeix un interès creixent, en el sector industrial, per la implantació de sistemes d'identificació remota per radiofreqüència. La identificació per radiofreqüència RFID (*Radio-Frequency IDentification*) és un mètode automàtic d'identificació on els objectes estan proveïts d'unes etiquetes que disposen de circuit integrat i antena, que permeten la seva identificació a distància.

Les consultes són generades per via sense fils per part d'un dispositiu anomenat lector RFID. Degut a que aquestes consultes són sense fils, ens trobem amb el problema que qualsevol podria intentar espiar el canal de comunicació, o introduir en ell els seus propis missatges. A més, s'ha de considerar la possibilitat que alguna de les etiquetes pugui caure en males mans, per la qual cosa una etiqueta mai haurà de contindre informació (en clar) que pugui comprometre al sistema.

Així doncs, per fer ús de la gran potencialitat que ofereix la identificació per radiofreqüència, és necessari dotar als sistemes RFID dels mecanismes de seguretat apropiats que evitin els problemes assenyalats.

En aquesta tesi, es proposen protocols de seguretat, aplicats a RFID, que actuen en dos nivells: la capa d'aplicació i la capa de comunicació.

Per a la capa d'aplicació, es proposa un protocol segur per a la identificació d'etiquetes, basat en l'ús de criptografia de corbes el·líptiques i en un protocol de coneixement nul.

A fi de facilitar el càlcul de corbes el·líptiques útils, es desenvolupa un mecanisme paral·lel i automàtic que genera aquestes corbes en un temps eficient.

Finalment, per a la capa de comunicació, es proposa un protocol de control d'accés al medi que evita que un atacant pugui comptar el nombre d'etiquetes presents en l'entorn. Aquest problema té sentit en entorns en els quals s'incloguin etiquetes RFID en productes clau, com podrien ser els bitllets de curs legal.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Hoy en día la sociedad ha evolucionado acostumbrada a un uso cada vez más intensivo de las nuevas tecnologías y las comodidades que éstas nos aportan. Dentro de las múltiples novedades que nos ofrece la tecnología, tenemos la posibilidad de detectar e identificar objetos remotamente. Esto es especialmente útil para el mundo industrial, que se puede beneficiar de una mayor automatización, y no está exento de ventajas para la sociedad actual.

Un sistema de identificación por radiofrecuencia (RFID: *Radio-Frequency IDentification*) es un método de identificación automático que permite identificar objetos a los que previamente se les ha adjuntado una etiqueta especial, llamada etiqueta RFID.

Una etiqueta RFID es un objeto que puede ser incorporado en un producto, con el fin de identificarlo a distancia, mediante ondas de radio. Estas etiquetas constan de un circuito integrado y una pequeña antena. El circuito les permite almacenar y procesar información, así como modular y demodular la señal de radiofrecuencia, que será transmitida o recibida por la antena.

Para que a una empresa o industria le compense el uso de un sistema RFID, las etiquetas deben ser muy baratas en comparación con el precio del producto etiquetado, de forma que, aunque productos muy caros podrían beneficiarse del uso de etiquetas 'menos baratas', en general éstas son dispositivos muy limitados, tanto en capacidad de cómputo como en memoria, que cuestan pocos céntimos de euro.

La escasa memoria de las etiquetas RFID les permite almacenar únicamente

un identificador de producto y algunos bytes más que requiera el protocolo de lectura. Debido a esta limitación, se requiere una base de datos de respaldo en el sistema RFID que asocie los identificadores con la información completa de los productos etiquetados.

El esquema básico de un sistema RFID puede verse en la Figura 1.1; éste consta de tres componentes:

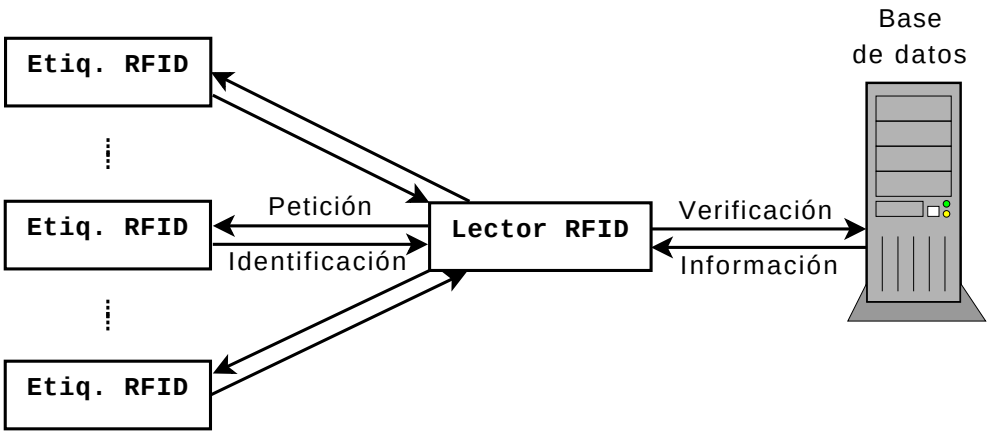


Figura 1.1: Sistema RFID.

- Etiquetas RFID, incorporadas a los productos (e.g. cajas de medicinas), que enviarán su identificador (*ID*) cuando se les solicite.
- Lector RFID, que se comunicará tanto con las etiquetas (para requerirles su identificador) como con la base de datos centralizada (para obtener la información completa del producto).
- Base de datos, que contiene la información de los objetos etiquetados (e.g. nombre de la medicina, componentes químicos, ...). Los lectores RFID consultarán la base de datos para identificar un objeto y para obtener su información asociada.

En cierto aspecto, las etiquetas RFID son similares a un código de barras, pero, en realidad, son mucho más versátiles, ya que el identificador permite distinguir entre diferentes unidades de un mismo tipo de producto (e.g. un

brick concreto de leche, en comparación con sólo la marca y el tipo de leche, en el caso de los códigos de barras). Esta distinción entre productos individuales permite dotar a un sistema RFID de interesantes funcionalidades, ya que la base de datos puede incorporar información individualizada (información de seguimiento del producto, fecha de empaquetado y de caducidad, etc).

Dependiendo de la fuente de energía de las etiquetas, éstas pueden ser clasificadas como etiquetas pasivas, etiquetas semipasivas (o semiactivas) o etiquetas activas.

Las etiquetas pasivas no tienen batería. Este tipo de etiquetas obtienen su energía a partir de la señal del lector. Las etiquetas semipasivas tienen una pequeña batería que aporta energía a su circuito cuando éstas son interrogadas. La batería se recarga siempre que la etiqueta reciba la señal del lector. Finalmente, las etiquetas activas tienen una batería más potente que aporta energía para realizar las transmisiones. Éstas etiquetas son más caras, por lo que sólo se usan en entornos donde su coste esté justificado. De entre los diversos tipos de etiquetas RFID, sólo las activas pueden iniciar una comunicación [SWE03].

Esta tesis está enfocada hacia los problemas que plantean los sistemas con etiquetas pasivas, ya que, al ser más baratas, son el tipo más ampliamente usado de etiquetas RFID [Jue06]. Sin embargo, su simplicidad implica importantes restricciones que deben ser respetadas: el coste no debería sobrepasar unos pocos céntimos de euro, el número de puertas lógicas debería ser inferior a unas 15000 [HLS09] (aunque no todas se pueden destinar a seguridad) y la tasa de transmisión debe ser de unos 400 kbit/s para los mensajes del lector a la etiqueta y unos 100 kbit/s para los mensajes de la etiqueta al lector.

Hoy en día, los sistemas RFID se usan en múltiples aplicaciones [MW04, NLLP04, Ang05, ASTP10], entre otras:

(a) Aplicaciones cuyo objetivo principal es la detección del objeto etiquetado en un punto de control:

- Identificación de personas (o animales).
- Control de inventario de bombonas de gas.
- Control de acceso.
- Control de *stock* de mercancía.

(b) Aplicaciones que requieren trazabilidad:

- Gestión de cadenas de suministro.
- Identificación de tubos de petroleo.
- Control de producción alimentaria.
- Identificación de análisis de sangre.
- Monitorización del vuelo de las palomas.
- Identificación de vehículos robados.
- Monitorización de cadenas de montaje.
- Identificación de partes.
- Monitorización de residuos tóxicos.

(c) Redes de sensores:

- Análisis de aguas.
- Monitorización de aparcamientos.
- Identificación de pneumáticos con sensores de presión.

(d) Aplicaciones que requieren ocultar la presencia de la etiqueta ante un atacante:

- Identificación de objetos de valor.
- Monitorización de guardias de seguridad.

Además de estas aplicaciones, cabe destacar, como dato significativo que, desde el año 2006, todos los pasaportes que se expiden dentro del territorio español corresponden al llamado pasaporte electrónico (*e-passport*) [HHJ+06, AKQ08], el cual incorpora un chip incrustado en su contraportada que contiene datos biométricos relativos a la imagen facial del titular del documento, además de los datos personales. Desde 2009 se incluyen también las huellas dactilares de los dedos índice de ambas manos.

Figura 1.2: Símbolo del pasaporte electrónico.

Este sistema ha sido adoptado por casi todos los países de la Unión Europea. La ventaja de este tipo de pasaportes es que son mucho más difíciles de falsificar y que permiten acelerar el proceso de identificación en las aduanas. En la Figura 1.2 se muestra el símbolo para los pasaportes biométricos, que suele estar impreso en la portada del pasaporte.

Cabe mencionar también que el Banco Central Europeo (BCE) estudia incorporar chips RFID a los billetes [YY03], para evitar la falsificación, el blanqueo de dinero y la evasión de divisas.

Todas estas aplicaciones exponen la gran potencialidad que pueden proporcionar las etiquetas RFID en todo tipo de procesos de identificación, pero a la vez hacen entrever una gran problemática de seguridad, que debe ser resuelta antes de llegar al uso masivo de las mismas. Debido a esto, se plantean una serie de retos para el colectivo científico, principalmente: evitar que se revele información sensible, evitar que podamos ser seguidos a través de las etiquetas que llevemos encima, evitar la falsificación de etiquetas y asegurar que el sistema RFID funcione correctamente en todo momento. La presente tesis se centra en resolver algunas de las problemáticas de seguridad que se plantean.

1.1. Evolución histórica de la tecnología RFID

El uso de los sistemas RFID se ha implantado en la última década gracias a la evolución de las tecnologías implicadas y a la miniaturización de los componentes electrónicos.

Antiguamente, la comprensión y el uso del electromagnetismo por parte de la humanidad eran muy limitados. Se observaban descargas electromagnéticas a simple vista (rayos), se entendían las propiedades de los imanes, pero poco más.

El progreso científico fue lento hasta el siglo XVI. A partir del XVII hubo una explosión de conocimiento sobre electromagnetismo y observaciones matemáticas relacionadas, y el siglo XIX marcó el inicio del entendimiento fundamental de la energía electromagnética.

En 1846, Faraday propuso que tanto la luz como las ondas de radio son formas de energía electromagnética. En 1864, Maxwell publicó su teoría sobre

electromagnetismo. En 1887, Hertz confirmó la teoría de Maxwell e hizo la primera transmisión de ondas de radio de la historia. En 1896, Marconi realizó la primera transmisión telegráfica transatlántica.

A principios del siglo XX se inició la comunicación por radio moderna, con un control sobre todos los aspectos de las ondas de radio. Además, es en ésta misma época cuando se considera que nació el radar. Esto último es fundamental, ya que una forma de RFID combina el radar con la tecnología de radiodifusión [Tro06].

El primer trabajo en la historia que exploró la idea de lo que acabaría dando lugar a la tecnología RFID es el artículo *Communication by Means of Reflected Power* [Sto48] publicado por Stockman en 1948. Durante esa misma década el radar recibió un gran impulso debido a la segunda guerra mundial.

En los años 50 del siglo XX hubo las primeras investigaciones sobre RFID. Varias tecnologías fueron exploradas como los sistemas de largo alcance para aviones "identificación, amigo, o enemigo" (IFF: *Identification, Friend, or Foe*).

Durante los años 60 se desarrolló la teoría electromagnética relacionada con RFID, y se iniciaron pruebas sobre posibles campos de aplicación. A finales de esta década, se fundaron las compañías *Sensormatic* y *Checkpoint*, las cuales, junto con otras como *Knogo*, desarrollaron la vigilancia electrónica de artículos (EAS: *Electronic Article Surveillance*) para evitar los robos de mercancías [Wik].

En los años 70, hubo una explosión del desarrollo sobre RFID, con un incremento de las pruebas y algunas implementaciones tempranas. Un desarrollo muy importante fue el presentado en el artículo *Short-Range Radio-Telemetry for Electronic Identification Using Modulated Backscatter* [KDF75] de Koelle, Depp y Freyman. Este trabajo marcó al inicio de las etiquetas completamente pasivas con un rango operacional de decenas de metros.

En la década de los 80 las aplicaciones comerciales basadas en RFID se utilizan de forma más extensa en el mundo industrial. En Europa, el principal interés estaba en los sistemas de corto alcance para animales y en las aplicaciones industriales y empresariales. En España, por ejemplo, los peajes de las autopistas de pago se equiparon con RFID (la VIA-T).

En los 90, la tecnología RFID se convierte en parte de la vida cotidiana.

Algunas compañías europeas que formaron parte activa en el desarrollo de los sistemas RFID, fueron *Microdesign*, *CGA*, *Alcatel*, *Bosch*, y los *spin-offs* de *Philips*: *Combitech*, *Baumer* y *Tagmaster*. Empezaron a aparecer los primeros libros específicos sobre RFID [HH88, Fin98].

A partir del año 2000 la explosión RFID no ha hecho más que continuar. El avance en el estudio de nuevos materiales ha permitido que en la actualidad las etiquetas se puedan construir como pegatinas, que pueden ser fijadas a casi cualquier objeto. La reducción de tamaño de las etiquetas está limitada por el tamaño de la antena, no por el del circuito integrado.

En el *Massachusetts Institute of Technology* (MIT) se creó el *Auto-ID Center* [Aut] para unir a fabricantes, investigadores y usuarios de sistemas RFID, con el fin de desarrollar estándares, investigar y compartir información. Los estándares para las aplicaciones de gestión de cadenas de suministro son desarrollados por *EPC Global*, mientras que para muchas otras aplicaciones se ocupa directamente la *International Standards Organization* (ISO).

El futuro de la tecnología RFID es prometedor, pero aún se necesitan avances en muchas áreas: infraestructuras de diseño, instalación y mantenimiento de sistemas RFID, software y aplicaciones, aspectos legales, y sobre todo, políticas de privacidad y seguridad.

1.2. Problemas de seguridad

Debido a que esta tesis tiene como objetivo dotar de mayor seguridad a las aplicaciones que hacen uso de sistemas RFID, se ha necesitado, en primer lugar, identificar los problemas que pueden derivarse del uso de éstos. Surgen tres problemas básicos del uso de tecnología RFID sin buenos mecanismos de seguridad:

- Revelado de la información de las etiquetas. En sistemas sin buenas medidas de seguridad cualquiera podría saber qué objetos etiquetados tenemos (e.g., el caso de los medicamentos etiquetados, lo que sería una grave invasión de la privacidad).

- Seguimiento del comportamiento a partir del seguimiento de alguna (o

varias) de las etiquetas que siempre llevemos encima (e.g. teléfonos móviles etiquetados).

- Recuento del número de etiquetas que hay en un entorno o que alguien lleva encima (e.g. contar el número de billetes que lleva alguien, si finalmente se introducen etiquetas RFID en los billetes, con insuficientes medidas de seguridad).

Cualquier sistema que sufra el problema del revelado de información padecerá también el problema del seguimiento, ya que cada vez que leamos una etiqueta el valor devuelto va a ser siempre el mismo, i.e. su identificador. Por tanto, podremos hacer el seguimiento de cualquier persona, a condición de que conozcamos el identificador de al menos una de las etiquetas que lleva siempre consigo. El atacante solo debe asegurarse de hacer una primera lectura cerca del objetivo a seguir con el fin de obtener sus identificadores, y a partir de ese momento el seguimiento podrá ser automatizado fácilmente.

Nótese que proteger un sistema del revelado de información no lo protege automáticamente del seguimiento, ya que la solución trivial de cifrar los identificadores antes de copiarlos a las etiquetas, evita que se revele información (el atacante no sabrá descifrar lo que lea) y sin embargo no evita el seguimiento: basta con seguir las etiquetas que envíen unos criptogramas en concreto, a pesar de no entenderlos. Por lo tanto, deberán usarse alternativas que tengan en consideración ambos problemas.

En resumen, los identificadores han de estar cifrados, para impedir el revelado de información sensible. Pero se puede hacer el seguimiento de una etiqueta que siempre responde el mismo valor (cifrado, o no), por lo tanto, se ha de cambiar frecuentemente el valor devuelto.

Un problema adicional, ignorado en muchos sistemas RFID, es el hecho de que un atacante pueda contar cuantas etiquetas hay en un entorno. Esto es un problema porque en algunos casos el simple hecho de conocer el número de etiquetas ya supone una información valiosa para el atacante. Si, tal como está planeado, el BCE acaba incluyendo chips RFID en los billetes de Euro, obviamente no querremos que cualquiera con un lector fraudulento sea capaz de deducir cuantos billetes llevamos encima (aunque no sepa la denominación o el

número de serie de ninguno de ellos). En el fondo, podemos ver este problema, como una forma del problema de revelado de información; en este caso, la información revelada por cada una de las etiquetas sería su propia presencia en el entorno. Además, tal como se ha mencionado previamente, cualquier sistema que revele información es susceptible de sufrir seguimiento. Incluso en este caso, imaginemos que cada persona llevara siempre encima un número de etiquetas invariante en el tiempo, y que la cantidad de etiquetas que lleva consigo la persona objetivo no coincide con ninguna otra. Un atacante podría usar esta información para hacer el seguimiento.

1.3. Objetivos de la tesis

Como se ha visto, los sistemas RFID tienen múltiples campos de aplicación y eso plantea diversos problemas de seguridad.

El objetivo de esta tesis se centra en proponer nuevos protocolos de seguridad, a distintos niveles, para las aplicaciones que hacen uso de sistemas RFID. Para ello, una vez identificados los problemas de seguridad, nos encontramos con la necesidad de diseñar un sistema que solucione cada uno de ellos y que actúe en las distintas capas que conforman el sistema RFID.

Para los problemas de revelado de información y seguimiento de las etiquetas, el objetivo se centra en la investigación de nuevos protocolos de lectura para la capa de aplicación, basados en la criptografía de curvas elípticas. La investigación basada en curvas elípticas se justifica en el entorno RFID, debido a que encaja mejor con las restricciones computacionales y de memoria de las etiquetas RFID, que no los sistemas criptográficos convencionales de clave pública.

Puesto que el objetivo es hacer uso de la criptografía de curvas elípticas, la necesidad de disponer de curvas seguras resulta primordial para el buen funcionamiento del sistema. La obtención de estas curvas es, por tanto, otro de los objetivos de la tesis.

Para solucionar el problema de la generación de las curvas, se propone la paralelización de un método para la obtención de gran cantidad de curvas criptográficamente útiles. Este método proporciona, a partir de una curva inicial,

otras curvas con la misma seguridad teórica que la original.

Finalmente, para el problema del recuento del número de etiquetas, nos proponemos la definición de un nuevo protocolo de control de acceso al medio que evite el recuento de etiquetas por parte de un atacante. Éste se basa en un procedimiento de respuesta aleatorio, y dinámico, por parte de las etiquetas, de manera que se regule automáticamente el flujo de respuestas al lector, a lo largo de la ejecución del protocolo.

1.4. Contribuciones realizadas

Durante el proceso de desarrollo de la tesis, se han aportado soluciones en la dirección de cada uno de los tres objetivos indicados anteriormente.

El protocolo de identificación seguro, perteneciente a la capa de aplicación, que proponemos, cubre el objetivo de tener un protocolo de lectura que evite el revelado de información y el seguimiento de las etiquetas. Este protocolo ha dado lugar a dos artículos: *An Elliptic Curve and Zero Knowledge Based Forward Secure RFID Protocol* [MVR+07] y *A Secure Elliptic Curve-Based RFID Protocol* [MVR+09].

Para la generación de curvas elípticas se ha propuesto un algoritmo paralelo que solventa el problema de obtener las curvas seguras necesarias para poder ir reemplazándolas con cierta regularidad. Se han realizado dos artículos exponiendo este método: *Paralelización del cálculo de volcanes para usos criptográficos* [MTR+05] y *Parallel Calculation of Volcanoes for Cryptographic Uses* [MTR+06].

Finalmente, el protocolo que proponemos para el control de acceso al medio (capa de comunicación) evita el problema del recuento del número de etiquetas. Esto permite obtener un sistema RFID mucho más seguro. El protocolo se ha presentado en el artículo: *Securing the Use of RFID-Enabled Banknotes* [MRV10], en el que, además, se plantea su necesidad, debido a la futura inclusión de etiquetas RFID en los billetes.

1.5. Estructura de la tesis

El contenido de la tesis está estructurado en los siguientes capítulos:

Capítulo 1: Introducción.

Se explica qué es un sistema de identificación por radiofrecuencia y su evolución a lo largo de la historia. Se plantean los objetivos de esta tesis y se exponen las contribuciones que se han aportado.

Capítulo 2: Seguridad en sistemas RFID: estado de la cuestión.

Se presentan algunas de las soluciones existentes en la literatura para aportar seguridad a los sistemas RFID.

Capítulo 3: Fundamentos criptográficos.

Se introducen algunos conceptos previos necesarios para esta tesis: criptografía sobre curvas elípticas, isogenias de curvas elípticas, protocolos de autenticación de conocimiento nulo y mecanismos de control de acceso al medio en entornos inalámbricos.

Capítulo 4: Protocolo seguro para la capa de aplicación.

Se presenta el protocolo de identificación seguro, basado en autenticación de conocimiento nulo sobre curvas elípticas, que se ha diseñado para la capa de aplicación de un sistema RFID.

Capítulo 5: Algoritmo paralelo para la generación de curvas elípticas.

Se expone el método paralelo que se ha implementado para la generación de curvas elípticas criptográficamente útiles, necesarias para el protocolo propuesto para la capa de aplicación.

Capítulo 6: Protocolo seguro para la capa de comunicación.

Se presenta un nuevo protocolo de control de acceso al medio eficiente y seguro que evita el recuento de etiquetas por parte de un atacante.

Capítulo 7: Conclusiones y trabajos futuros.

Se exponen las conclusiones y posibles trabajos futuros que podrían continuar el trabajo de esta tesis.

Capítulo 2

Seguridad en sistemas RFID: estado de la cuestión

Este capítulo da, a modo de resumen, una relación de las posibles formas de atacar la seguridad de un sistema RFID, junto con diversas soluciones aportadas por la literatura para detectar y contrarrestar cada ataque. A partir del análisis de estas soluciones previas y de las vulnerabilidades que no han sido solventadas satisfactoriamente, la aportación de esta tesis se ha basado en una alternativa eficaz para proporcionar seguridad a los sistemas RFID.

Un sistema de comunicación basado en tecnología RFID consta de tres capas [SWE03]:

(a) Capa de aplicación: trata la información del usuario. Es en esta capa en donde se definen los protocolos de identificación y autenticación (si se necesita). Requiere de los servicios de las capas inferiores, para transmitir la información necesaria.

(b) Capa de comunicación: controla las comunicaciones entre lector y etiquetas. En esta capa se encuentra el protocolo de control de acceso al medio (MAC: *Medium Access Control*). En telecomunicaciones, el MAC es el conjunto de mecanismos y protocolos que permite a los dispositivos compartir un mismo canal de comunicación.

(c) Capa física: define la interfaz física aérea, las características eléctricas (tipo de modulación, etc) que se usarán en la transmisión, y gestiona

el establecimiento, mantenimiento y liberación del enlace. Básicamente, esta capa se encarga de transmitir el flujo de información a través del medio.

La seguridad global del sistema recae en cómo se asegura cada capa individualmente [AO05b], pues cada capa tiene sus propias vulnerabilidades asociadas. Vamos a ver, en este capítulo, algunos de los problemas de seguridad en los entornos RFID y las principales soluciones que se han propuesto para ellos en la literatura.

2.1. Ataques tipo al sistema RFID

De acuerdo con la literatura existente [MRT08, MRT09], hay una serie de ataques básicos que deben ser considerados a la hora de evaluar la seguridad de un sistema RFID.

Una primera distinción la podemos realizar entre ataques pasivos y activos. En un ataque pasivo el adversario trata de obtener información sin interferir en el funcionamiento normal del sistema. Su capacidad es bastante limitada (y por tanto, es inútil ante sistemas que cumplan unos mínimos requisitos de seguridad), sin embargo tienen la ventaja de ser prácticamente indetectables. Muchos ataques necesitan obtener algún tipo de información adicional del sistema, o de las etiquetas, para poder surtir efecto. Debido a ello, minimizar las fugas de información sensible es crucial en cualquier sistema.

En un ataque activo, en cambio, el adversario participa de alguna manera en el protocolo, enviando él mismo información, capturando etiquetas, generando ruido, o cualquier otra acción que implique algo más que limitarse a escuchar los canales de comunicación. Si bien estos ataques pueden llegar a ser más peligrosos, las acciones del adversario podrían ser detectadas, ya que existen mecanismos diseñados para ello, como por ejemplo el *RFID Guardian*[1] [RCT05, RGC+06].

[1] *RFID Guardian* es un dispositivo electrónico portátil que se puede llevar encima. De hecho, éste podría llegar a integrarse en PDAs y *smartphones*, en el futuro. Su función es buscar, recordar y mostrar todas las etiquetas RFID presentes en su proximidad, así como las lecturas que se realicen; administrar claves RFID, autenticar a lectores cercanos y bloquear el acceso a las etiquetas del usuario por parte de lectores no autorizados.

En la literatura, se han detallado distintos tipos de ataques, atendiendo a los objetivos del atacante o a las técnicas empleadas. Un sistema RFID seguro ha de disponer de mecanismos que le permitan defenderse de los ataques que se relacionan a continuación:

(a) *Sniffing*

En un ataque de *sniffing*, el atacante escucha las comunicaciones entre un lector y una etiqueta, intentando obtener información útil. Éste es, por tanto, un ataque inherentemente pasivo.

Una forma habitual de defenderse contra este tipo de ataques es utilizar el cifrado de la información.

(b) Seguimiento de las etiquetas

El ataque de seguimiento (*tracking*) de las etiquetas persigue conocer el comportamiento del usuario de una etiqueta, o de un conjunto de etiquetas, a partir del conocimiento de la localización de las mismas en diversos instantes de tiempo. Por ejemplo, si alguien tiene una etiqueta RFID en su teléfono móvil, el seguimiento de esta etiqueta facilita el seguimiento de su comportamiento. Así, un atacante podría tener lectores clandestinos en diversos puntos clave, de manera que supiera cuando sus víctimas pasan por cada uno de ellos.

La clave para evitar este tipo de ataques consiste en que la etiqueta no envíe siempre la misma información como respuesta. En caso contrario, si la respuesta de una etiqueta fuese siempre la misma, o fuera previsible, incluso a pesar de que ésta estuviera cifrada, al atacante le bastaría con rastrear los lugares en los que se obtiene esa respuesta (a pesar de que no pudiera entenderla) para seguir al dueño de la etiqueta.

(c) Suplantación

Un ataque de suplantación (*spoofing*), trata de suplantar la identidad de alguna de las entidades de un sistema RFID. Se deben considerar dos versiones diferentes de este ataque, en función de la entidad suplantada:

- Suplantación de un lector (por ejemplo, para realizar identificaciones no autorizadas).
- Suplantación de una etiqueta (por ejemplo, para la falsificación de productos).

Una forma clásica de evitar este tipo de ataques es mediante la utilización de métodos de identificación y autenticación de las partes implicadas.

(d) Denegación de servicio

Un ataque de denegación de servicio (DoS: *Denial of Service*), consiste en la incapacitación temporal, o permanente (e.g. destrucción de una etiqueta), del sistema o de parte de él.

Debido al hecho de que los sistemas RFID son inalámbricos, hay ataques DoS, como los causados por la presencia de fuentes no controladas de ruido electromagnético en la frecuencia de radio de un sistema, que nunca pueden ser evitados (aunque sí detectados).

Existen, sin embargo, otros tipos de ataques DoS más elaborados que pueden ser evitados por algunos sistemas. En general, los sistemas más complejos suelen ser más fáciles de atacar, pues dependen de más requisitos para funcionar, e incluso ciertos sistemas pueden favorecer ataques DoS que normalmente no serían factibles.

(e) Ataque de reutilización

Un ataque de reutilización (*replay attack*) consiste en que un atacante reenvíe información que haya capturado previamente, por ejemplo, escuchando una sesión de lectura anterior.

Este tipo de ataque es un medio, más que un fin. Puede ser utilizado para realizar una suplantación, un ataque DoS, obtener información, etc.

(f) *Forward attack*

En un *forward attack*, el atacante va capturando y almacenando información que ha sido enviada cifrada por la etiqueta, con la esperanza de poderla descifrar en el futuro, vía la obtención de información secreta

interna de la etiqueta. Esto suele requerir realizar un ataque físico, el cual se verá más adelante.

Este tipo de ataque tiene sentido cuando la información capturada no está contenida en la información secreta obtenida de la etiqueta. Como ejemplo de esta situación tenemos las redes de sensores, en las que las etiquetas tienen algún tipo de sensor (e.g. identificación de pneumáticos con sensor de presión, redes de análisis de aguas, ...). Suponiendo que las lecturas del sensor se envíen cifradas de alguna forma, un sistema es vulnerable a un *forward attack* si la obtención de información secreta de la etiqueta permite descifrar la información cifrada anteriormente por la etiqueta.

La propiedad de *forward security* garantiza que un sistema es resistente a este tipo de ataques.

(g) Ataque de canal lateral

Los ataques de canal lateral (SCA: *Side Channel Attacks*) intentan obtener información a partir de la implementación física del sistema RFID, en lugar de a partir de vulnerabilidades teóricas del diseño del propio protocolo.

A partir de la capa física se pueden obtener diversas fuentes adicionales de información a ser explotada (los llamados canales laterales). Como ejemplos tenemos: la información del tiempo de cómputo, la monitorización del consumo de energía, las fugas electromagnéticas, las huellas de radiofrecuencia, el análisis diferencial de errores, la información sonora, e incluso la observación con un telescopio del reflejo de un monitor en las gafas de un usuario.

También existen ataques SCA específicos al uso de criptografía sobre curvas elípticas, como son los basados en los llamados puntos de valor cero (ZVP: *Zero-Value Points*) [MST+09, MST+10].

Los ataques SCA dependen de la implementación elegida para el sistema RFID. Por tanto, dos protocolos iguales desde un punto de vista teórico pueden tener una gran diferencia en seguridad una vez implementados.

(h) Ataque físico

En un ataque físico, el atacante tiene acceso al *hardware* de una (o varias) etiquetas, de manera que puede aplicar la ingeniería inversa para obtener información interna almacenada en la etiqueta. Esto posibilita, en la mayoría de los sistemas (sean RFID o de otro tipo), la realización de muchos otros ataques.

Conseguir que un sistema mantenga unos requisitos mínimos de seguridad, aun contando con que un atacante pueda tener acceso físico a las etiquetas, resulta bastante complejo.

La solución ideal sería poder utilizar *hardware* resistente a manipulación (*tamper resistant*), pero éste es excesivamente costoso, por lo que no es apropiado para entornos RFID. Así pues, se han de diseñar los sistemas tratando de minimizar las consecuencias de este ataque.

Al plantear un sistema, los desarrolladores deben decidir a qué ataques hacer frente y a cuales no.

2.2. Soluciones de seguridad

La literatura existente sobre RFID ha propuesto, durante los últimos años, diversas alternativas para solucionar los problemas de privacidad y seguridad.

No es el objetivo de esta tesis exponerlas todas en detalle, sino dar unas pinceladas de las ideas generales empleadas en las distintas soluciones, pues existen similitudes conceptuales en varias de ellas. Algunas no basan su seguridad en la criptografía y las que lo hacen suelen preferir la criptografía ligera. Sin embargo, aportan conceptos claves que se pueden aplicar en múltiples escenarios.

Para facilitar la comprensión de las soluciones expuestas, éstas se agruparán según la capa (aplicación, comunicación o física) a la que están dirigidas.

2.2.1. Propuestas para la capa de aplicación

Un primer grupo de soluciones lo constituyen los esquemas basados en la desactivación temporal o permanente de las etiquetas. Estas propuestas se enfocan sólo en proporcionar un nivel muy básico de seguridad, representando un incremento de coste mínimo para el sistema.

(a) *Tag Killing scheme*

El más básico de ellos probablemente sea el *Tag Killing scheme* [Aut02]. En éste, las etiquetas poseen un comando protegido por un PIN que permite desactivarlas ("matarlas") para proteger la privacidad del usuario.

Cuando una etiqueta recibe dicho comando, con el PIN correspondiente, se vuelve inoperativa de manera permanente. Éste es, evidentemente, un esquema muy sencillo de implementar, pero esta aproximación no es apropiada para sistemas que necesiten beneficiarse continuamente de la identificación RFID.

Además, toda la información (tanto identificadores como comandos) se envía en claro, así que las suplantaciones son triviales y, de hecho, cualquier atacante podría matar etiquetas realizando, por tanto, una denegación de servicio.

(b) *Sleep/Wake modes*

Es una evolución del *Tag Killing scheme*, en la que las etiquetas no se desactivan de forma permanente. En el presente esquema, las etiquetas tienen dos comandos protegidos por PIN, *sleep* y *wake* (dormir y despertar). Una etiqueta se desactiva cuando recibe el comando *sleep*, y se reactiva cuando recibe el comando *wake*.

Igual que en el esquema anterior, el usuario debe elegir entre preservar su privacidad o poder usar las funcionalidades RFID, puesto que las dos opciones no son compatibles al mismo tiempo.

El siguiente grupo de soluciones basa su mecanismo de seguridad en no revelar el identificador del objeto etiquetado, puesto que la etiqueta no almacena su identificador real. Esto aporta cierta dosis de privacidad al sistema, ya que

un atacante cuyos únicos medios consistan en espiar las comunicaciones entre lector y etiqueta, no podrá relacionar la información obtenida con un producto concreto.

(c) *Anonymous ID*

En este esquema [KHK+03], lo que retorna la etiqueta es un identificador anónimo que fue generado previamente mediante criptografía o como un valor aleatorio. Cabe remarcar que, sin embargo, permite el seguimiento de las etiquetas, puesto que el identificador es fijo.

(d) *External Encryption*

En éste [JP03], los datos de la etiqueta son cifrados cada cierto tiempo por una unidad externa usando criptografía de clave pública.

Un atacante que haya almacenado el valor devuelto por una etiqueta podrá realizar el seguimiento de la misma (aunque no pueda descifrar el identificador) durante un tiempo, hasta que ésta vuelva a ser cifrada por la unidad externa, en cuyo caso el atacante no podrá asociar el nuevo valor con el anterior y, por lo tanto, "perderá la pista" de la etiqueta.

Otras propuestas basan su mecanismo de seguridad en el uso de funciones *hash*, para la autenticación del lector o la etiqueta.

(e) *Hash Lock*

En este esquema [WSRE03], el lector tiene una clave k para cada etiqueta; cada etiqueta almacena $metaID = hash(k)$ (además del identificador real). En una operación de lectura, la etiqueta envía su $metaID$, el lector le envía su clave, la etiqueta verifica el *hash* y, si es correcto, envía su identificador real.

Como se puede observar, un atacante que no posea las claves puede hacer un seguimiento trivialmente, gracias a que para una misma etiqueta el $metaID$ es siempre constante. Además, un espía que esté escuchando una sesión con un lector válido, obtendrá tanto la clave como el identificador real de la etiqueta, puesto que éstos se envían en claro.

(f) *Random Hash Lock*

El esquema *Random Hash Lock* [WSRE03] soluciona el problema de la privacidad y el seguimiento de las etiquetas, porque éstas envían identificadores generados de forma pseudoaleatoria.

En una operación de lectura, la etiqueta calcula $c = hash(id|r)$, donde r es un valor aleatorio, y envía (c, r). La base de datos calcula el *hash* de cada uno de los identificadores concatenados con r, hasta encontrar el que dé como resultado c; el identificador correcto es retornado al lector.

Como el identificador nunca se llega a enviar en claro, éste podría usarse para cifrar datos, por ejemplo, procedentes de algún tipo de sensor. Sin embargo, en caso de producirse un ataque físico, el atacante puede obtener el identificador de la etiqueta, y usándolo, descifrar cualquier información que haya sido cifrada con él. Por tanto, este esquema no proporciona *forward security*.

Además, esta solución tiene un problema de escalabilidad, ya que requiere que el lector compruebe todas las posibles claves, lo cual puede no ser factible en sistemas con un gran número de etiquetas.

Un tipo de esquemas más eficientes son los basados en árbol de claves, como los dos que se describen a continuación. En estos sistemas, la búsqueda de claves que el lector debe hacer para identificar una etiqueta es logarítmica en el máximo número de etiquetas (depende de la profundidad del árbol).

(g) Molnar y Wagner

En este esquema [MW04], cada nodo del árbol tiene una clave, y cada etiqueta está asociada a una hoja del árbol de claves, y además recibe todas las claves correspondientes al camino desde la raíz hasta su hoja.

Los esquemas basados en árbol pueden ser vulnerables a un problema de filtración de datos por un ataque específico, llamado ataque de compromiso: si un atacante captura un grupo de etiquetas, consigue acceso a todas las claves desde la raíz hasta sus hojas correspondientes, comprometiendo, por tanto, algunas de las claves de las etiquetas no capturadas.

(h) Protocolo *SPA*

En [LLH+07], Lu *et al.* propusieron el protocolo *SPA*, que considera que cada nodo del árbol almacena dos (o en general, un número fijo) de claves, de las cuales una etiqueta sólo almacena una para cada nodo de su camino desde la raíz. Cuando una etiqueta T ha sido identificada, las claves pueden actualizarse, pero como la base de datos mantiene las antiguas, otras etiquetas que compartan nodos con T todavía se podrán identificar correctamente.

Este protocolo es capaz de defenderse contra el mencionado ataque de compromiso, reduciendo altamente la probabilidad de exposición de claves cuando se capturan etiquetas.

Otro tipo de protocolos son los que están basados en el uso de etiquetas especiales en el entorno.

(i) *Noisy Tags*

En [CA06], se proporciona seguridad mediante "etiquetas ruidosas", que comparten un secreto común con el lector. Cuando las etiquetas respondan a una petición del lector, habrá una colisión entre las respuestas de las *noisy tags* y las de las etiquetas normales. El lector restará todas las respuestas de las *noisy tags* y obtendrá la información enviada por la etiqueta genuina (si sólo respondió una).

El principal peligro de éste método proviene de la posible detección por parte de un atacante de la posición física de una *noisy tag*. En tal caso, el atacante podría destruir la *noisy tag* o, en caso de que ésta se encontrase protegida, simplemente apantallar o reflejar su señal de manera que no se reciba su secuencia aleatoria.

Nótese que, como el lector resta la respuesta que espera de dicha *noisy tag*, esto puede conllevar una denegación de servicio.

Si el atacante consigue destruir o bloquear la señal de todas las *noisy tags* del entorno, el resto de etiquetas estarán desprotegidas, pues su información viajará en claro por el aire.

(j) *Blocker Tag*

La idea de usar etiquetas extra como método para proporcionar seguridad también se usa en el *blocker tag protocol* [JRS03]. De todas formas, el objetivo es completamente diferente: prevenir la identificación de una o más etiquetas por parte de cualquier lector. Esta etiqueta bloqueadora confundirá al lector haciéndose pasar por la etiqueta (o conjunto de etiquetas) que queramos proteger.

El principal problema de este método es que, mientras la *blocker tag* esté actuando, las etiquetas protegidas no pueden ser detectadas ni siquiera por un detector genuino; y si se desactiva la *blocker tag* para poderlas detectar, se pierde la seguridad del sistema, puesto que entonces también las podría detectar un adversario. Por tanto, este método sólo tiene sentido en situaciones donde las etiquetas ya han finalizado su uso en el entorno RFID, por ejemplo, tras comprar un producto.

Hay otras propuestas que pueden cubrir requisitos de seguridad más estrictos. Como son, por ejemplo, los esquemas de cadenas de *hashes*. En estos esquemas, la etiqueta modifica su identificador tras cada lectura mediante la aplicación de una función de *hash*. Esto crea una cadena de identificadores que el lector también debe generar para identificar la etiqueta.

(k) *Hash Chains*

Ohkubo, Suzuki y Kinoshita propusieron en [OSK03] un esquema en el que cada etiqueta mantiene un secreto que puede ser usado para enviar datos cifrados. Cuando el lector envía una petición de identificación, la etiqueta genera el identificador a enviar mediante la aplicación de una función de *hash* a su secreto, e inmediatamente modifica el secreto usando una segunda función de *hash*.

En la Figura 2.1, podemos ver esquematizado el funcionamiento del protocolo. En el instante k, el secreto de la etiqueta i es s_i^k; la lectura del sensor produce el valor u^k. El identificador a enviar, r_i^k se genera mediante la función de *hash* G aplicada a s_i^k. El dato del sensor, u^k, se

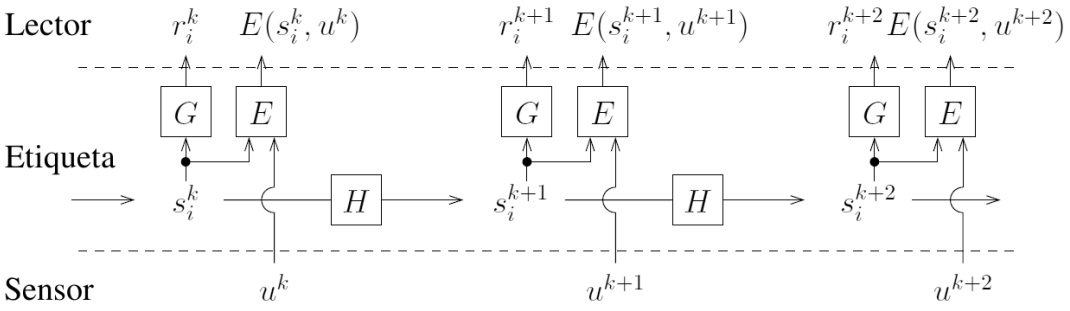


Figura 2.1: Esquema *Hash Chains* (figura extraída de [AO05a]).

cifra mediante la función E, usando s_i^k como clave. Finalmente, el nuevo secreto, s_i^{k+1}, se obtiene vía la aplicación de la función de *hash* H a s_i^k.

Esta aproximación resuelve los problemas de filtración de dados, evita el seguimiento de las etiquetas y proporciona *forward security*. Desgraciadamente, presenta una alta complejidad computacional para la base de datos durante la fase de identificación, pues ésta debe calcular todos los posibles *hashes* hasta que encuentra una coincidencia. Esto puede ser aprovechado por un atacante, puesto que si éste fuerza muchas veces la lectura de una misma etiqueta, cuando un lector legítimo intente identificarla, la base de datos se verá obligada a generar cadenas de *hashes* mucho más largas para todas las posibles etiquetas, hasta identificar la etiqueta atacada. Esto puede ralentizar el sistema, por tanto, este protocolo es vulnerable a la denegación de servicio, y además no es escalable.

(l) *Hash Chains with time-memory tradeoff*

El problema de la complejidad del esquema previo se soluciona parcialmente en el esquema de Avoine y Oechslin [AO05a] realizando una compensación tiempo-memoria (*time-memory tradeoff*), en la que algunos identificadores (no consecutivos) son precalculados y almacenados.

El problema principal de esta aproximación recae en la escalabilidad, ya que el número de identificadores almacenados aumenta demasiado cuando el número de etiquetas del sistema es muy elevado.

La razón que crea la necesidad de recursos extraordinarios en los dos protocolos anteriores es que a cualquier lector (incluso a los maliciosos) se le permite leer las etiquetas. Las lecturas inesperadas de los lectores maliciosos causarán la modificación del secreto interno de la etiqueta, de tal manera que la base de datos no sabrá cual es el valor a retornar esperado para cada etiqueta en un cierto momento. Esta incertidumbre es la responsable de los cálculos extra en el esquema de Ohkubo *et al.* y de la memoria extra en el de Avoine y Oechslin, ya que tienen que verificar todas las posibilidades hasta encontrar la correcta. Esto hace a estos dos protocolos seguros, aunque no sean escalables.

En el Capítulo 4 de esta tesis, proponemos un protocolo para la capa de aplicación, que resumimos a continuación, en el que basándonos en algunas de las propuestas presentadas, tratamos de dar solución a los problemas mencionados en la Sección 2.1.

(m) *Elliptic Point Chains*

En [MVR+07, MVR+09] propusimos un esquema en el que, con el fin de que sólo los lectores válidos puedan leer las etiquetas, éstos tienen que autenticarse. Con esto, se consigue la misma seguridad que las dos propuestas anteriores, basadas en cadenas de *hashes*, pero ofreciendo una alternativa que mejora su principal debilidad: la baja escalabilidad.

Debido a la inseguridad intrínseca del canal de comunicación entre lectores y etiquetas, se usa un protocolo de conocimiento nulo (ZKP, de *Zero Knowledge Protocol*) para llevar a cabo la autenticación de los lectores. Un ZKP permite al lector demostrar el conocimiento de un secreto sin revelar ninguna información relacionada con él. Así, las autenticaciones previas entre lectores y etiquetas no se pueden reutilizar por parte de lectores maliciosos que intenten suplantar a uno válido.

Este protocolo, propuesto en la presente tesis, está basado en criptografía de curvas elípticas, porque permite usar menos bits que la criptografía de clave pública convencional, haciendo su implementación más factible. Además, este protocolo mejora a los anteriores (como se aprecia en la Tabla 2.1) pues consigue dar solución a todos los problemas de seguridad considerados, manteniendo, además, la escalabilidad.

Tabla 2.1: Comparativa de soluciones de la capa de aplicación.

	¿Evita filtrar datos?	¿Evita el seguimiento?	¿Evita suplantar un lector?	¿Evita suplantar etiquetas?	¿Evita la DoS?	*Forward Security*	Comentarios
Tag Killing scheme	Sólo si está muerta	Sólo si está muerta	No	No	No	N/A	Hay que escoger entre seguridad o usar RFID
Sleep/Wake modes	Sólo si está dormida	Sólo si está dormida	No	No	Sí	N/A	Si un atacante duerme a una etiqueta, ésta se puede volver a despertar
Anonymous ID	Sí	No	No	No	Sí	N/A	No existe relación entre identificador y producto
External Encryption	Sí	Se dificulta	No	No	Sí	N/A	Si se conoce el *ID*, se puede hacer seguimiento hasta que se cambie
Hash Lock	No	No	Sólo con etiquetas no leídas	Sólo las no leídas	Sí	N/A	Claves e identificadores se envían en claro
Random Hash Lock	Sí	Sí	No, pero es inútil hacerlo	Sí	Sí	No	Se deben comprobar todas las posibles claves
Molnar y Wagner	Ataque de compromiso	Ataque de compromiso	Sí	Sí	Sí	N/A	El ataque de compromiso requiere capturar un grupo de etiquetas
Protocolo *SPA*	Minimiza el compromiso	Minimiza el compromiso	Sí	Sí	No	N/A	Riesgo de que una etiqueta se quede con un secreto obsoleto
Noisy Tags	Sí	Sí	No, pero es inútil hacerlo	Sí	Sí	Depende del generador aleatorio	Si se silencian las *noisy tags* se pierde la seguridad
Blocker Tag	Se minimiza	Se minimiza	No	No	No	N/A	Para rangos pequeños se revela información
Hash Chains	Sí	Sí	No	Sí	No	Sí	La BD debe calcular todos los posibles *hashes*
Hash C. with time-memory tradeoff	Sí	Sí	No	Sí	Sí	Sí	Importante consumo de memoria
Elliptic Point Chains	Sí	Sí	Sí	Sí	Sí	Sí	Véase el Capítulo 4

Dada la diversidad de tipos de ataques y soluciones de seguridad que existen para la capa de aplicación de los sistemas RFID, se ha considerado oportuno elaborar un resumen a modo de tabla (Tabla 2.1), donde se relacionan los protocolos de seguridad con los ataques a los que son (o no) resistentes.

Las vulnerabilidades consideradas en la Tabla 2.1 son: la filtración de datos, el seguimiento de las etiquetas, la suplantación de un lector y de las etiquetas, y la denegación de servicio. También se menciona si el esquema proporciona *forward security*.

Cabe mencionar que, exceptuando el *forward attack*, para el resto de ataques se considera que el atacante no tiene acceso físico a las etiquetas, en el sentido de que no puede leer su memoria interna ni destruirlas (un ataque físico posibilita la mayoría de los ataques trivialmente).

Respecto a la denegación de servicio, con un generador de ruido electromagnético se puede inhabilitar cualquier sistema. Debido a ello, sólo consideramos los ataques capaces de inhabilitar un sistema de manera indefinida, sin que haya presentes fuentes de ruido.

Finalmente, los sistemas en los que las etiquetas no almacenan ningún dato secreto compartido con el lector, no tienen la posibilidad de cifrar la información, por lo que en éstos, la propiedad de *forward security* no es aplicable (indicado en la tabla como 'N/A').

2.2.2. Propuestas para la capa de comunicación

En la capa de comunicación de un sistema RFID se encuentra, principalmente, el protocolo MAC. En un sistema RFID, el protocolo MAC hace posible que varias etiquetas se comuniquen en un entorno común con el lector RFID.

Como se ha dicho al principio de éste capítulo, cada una de las tres capas tiene que ser segura; pero a la hora de diseñar un protocolo MAC para RFID, la principal preocupación que suele ser tenida en cuenta es su rendimiento: identificar la mayor cantidad de etiquetas en el mínimo tiempo. Éste es el caso de varios protocolos MAC para RFID existentes, como el *Framed Aloha* [FW06] y sus variaciones [FW06, KYLS06], y el *Colorwave* [WES03], entre otros.

En el protocolo *slotted Aloha* básico [Abr70], el acceso al canal de comuni-

cación está dividido en varias ranuras de tiempo. Al inicio de cada petición de lectura, el lector escoge el número de ranuras e informa a las etiquetas de ello. Cada etiqueta escoge una ranura aleatoriamente, para responder. Si dos o más etiquetas escogen la misma ranura (lo que siempre pasará si hay más etiquetas que ranuras) entonces se produce una colisión. Si se da el caso, el lector deberá iniciar otra sesión para poder identificar las etiquetas que colisionaron. Para facilitar el proceso, puede indicar a las etiquetas que fueron identificadas que no respondan durante esa sesión, mediante el envío de sus identificadores o de la ranura que usaron la sesión anterior (obviamente, éste último método es más seguro) y, si es necesario, se puede escoger una cantidad diferente de ranuras.

En general, estos protocolos tienden a degradar el tiempo de respuesta cuando el número de etiquetas presentes en el entorno es grande, así que, proporcionar escalabilidad también es un asunto importante.

En entornos específicos hay un problema adicional que es el recuento de etiquetas. Por ejemplo, en posibles escenarios futuros como sería tener etiquetas RFID incrustadas en los billetes [JP03, RCT06], prevenir que un atacante cuente el número de etiquetas será tan importante como evitar el revelado de sus datos (la denominación del billete, por ejemplo).

Casi ningún protocolo, aunque se aplique a sistemas que se consideran seguros, soluciona el problema de que un atacante cuente el número de etiquetas. De hecho, este problema no suele ser tenido en cuenta, probablemente porque en muchos entornos esto no se considera una amenaza.

En [CLL07] se propuso un protocolo para proporcionar privacidad de localización al propietario de un billete en sistemas de banca basados en RFID, pero tampoco tenía como objetivo el problema del recuento de etiquetas.

En esta tesis, en el Capítulo 6, proponemos un protocolo MAC para RFID que aporta: (a) escalabilidad en el funcionamiento del sistema, adaptando la frecuencia de respuesta de las etiquetas al número de ellas que se encuentran en el entorno, y (b) privacidad sobre el número de etiquetas presentes, mediante el uso de una *noisy tag* para enmascarar el número de respuestas a las peticiones del lector. De esta manera, un atacante no puede saber cuando ha ocurrido una respuesta o una colisión de etiquetas no ruidosas.

2.2.3. Propuestas para la capa física

En el caso de la capa física, básicamente, se deben evitar las filtraciones de información debidas a las huellas de radiofrecuencia [TK95, AO05b] y a los *side channel attacks*.

Respecto a los problemas con las huellas de radiofrecuencia, éstos se pueden evitar usando un estándar común y asegurando un proceso de fabricación común para todas las etiquetas [AO05b]. Esto se puede conseguir fácilmente si un único fabricante produce todas las etiquetas.

Para evitar las filtraciones por *side channel attacks*, se deben tomar contramedidas específicas que pueden variar de un protocolo a otro, pero que, en general, tienen un objetivo común: igualar lo más posible el consumo de energía y tiempo de las operaciones de la etiqueta, independientemente de los datos con los que trabajen. En [MP10] se presentan algunas contramedidas para el ataque *side channel* conocido como *Differential Power Analysis* (DPA).

La protección contra los ataques *side channel* así como el resto de ataques físicos escapa el ámbito de estudio de esta tesis, debido a que son cuestiones que dependen de la implementación concreta de cada sistema. Sin embargo, un sistema teóricamente seguro podría ser vulnerable a estos ataques, por lo que es un aspecto importante a considerar en el momento de realizar cualquier implementación.

Capítulo 3

Fundamentos criptográficos

En esta tesis se proponen soluciones de seguridad para sistemas RFID que requieren el uso de ciertas técnicas criptográficas. Por ello, se ha decidido incluir este capítulo con el fin de introducir los fundamentos criptográficos necesarios para la posterior comprensión de los protocolos planteados.

Se comenzará por una breve introducción a la criptografía clásica (Sección 3.1), para luego pasar a la criptografía sobre curvas elípticas (Sección 3.2) e isogenias de curvas elípticas (Sección 3.2.1), los protocolos de autenticación de conocimiento nulo (Sección 3.3), con el protocolo Schnorr elíptico (Sección 3.3.1) como ejemplo.

3.1. Introducción a la criptografía

La criptografía es el arte de cifrar y descifrar información, de manera que sólo pueda entenderla quien tenga los medios (claves) para descifrarla [MvOV96, Sta10]. Junto con el criptoanálisis, que estudia métodos para recuperar la información sin utilizar las claves, constituyen la criptología, que es la ciencia que engloba a ambos [Sin99].

La información a proteger se denomina *texto en claro* o *texto llano*, que una vez cifrado se convierte en un *criptograma*. El algoritmo de descifrado requiere normalmente una clave que no deben conocer aquellos a los que no vaya dirigida la información.

En los llamados criptosistemas simétricos o de *clave privada*, esta clave

es compartida entre emisor y receptor. En los criptosistemas asimétricos o de *clave pública*, se usa una clave para cifrar y otra distinta para descifrar, de manera que la primera puede hacerse pública sin problemas.

Existe un criptosistema simétrico que, si se utiliza bien, es imposible de romper: la *libreta de un solo uso* (*one-time pad*). Cada carácter (o bit) del texto llano se cifra mediante una suma modular con un carácter (o bit) de una clave aleatoria de la misma longitud que el texto en claro. Originariamente, este criptosistema proporcionaba las claves en libretas físicas, de manera que se fueran destruyendo las páginas a medida que se fueran utilizando.

Si la clave es realmente aleatoria, igual de larga, o más, que el mensaje, y se usa una única vez, entonces el criptograma será imposible de descifrar sin conocer la clave. Esta propiedad se conoce como *secreto perfecto*[1].

Sin embargo, la libreta de un solo uso tiene ciertos inconvenientes. En primer lugar, requiere una secuencia perfectamente aleatoria (no basta que sea pseudoaleatoria), lo que no es fácil de conseguir. En segundo lugar, ningún dato de la secuencia puede ser reutilizado jamás, ni en un mismo criptograma (de ahí la necesidad de que la libreta sea tan larga como el mensaje) ni tampoco en mensajes posteriores, lo que plantea serios problemas para la gestión y almacenamiento de claves. Cualquier fallo en el cumplimiento de estos requisitos elimina la propiedad de *secreto perfecto*.

Debido a esto, normalmente se utilizan otros criptosistemas simétricos menos exigentes. Algunos de los más conocidos son el DES (*Data Encryption Standard*) que usaba claves de sólo 56 bits, y que, por ello, hoy en día se considera inseguro, el 3DES (Triple DES) que hace triple cifrado DES y tiene una seguridad equivalente a una cifra de 112 bits, y el AES (*Advanced Encryption Standard*, también conocido como *Rijndael*), que con sus claves de 128, 192 o 256 bits, está reemplazando al 3DES y puede ser hasta seis veces más rápido.

El principal inconveniente de los sistemas simétricos consiste en el problema del intercambio de claves. Si en un grupo de n personas todos quieren comunicarse con todos, habrá un total de $\binom{n}{2} = \frac{n^2-n}{2}$ claves diferentes (una por cada

[1]La propiedad de *secreto perfecto* implica que el criptograma no aporta ninguna información sobre el texto llano. Es decir, la probabilidad *a priori* de que el texto llano sea un cierto mensaje M es la misma que la probabilidad *a posteriori* de que el texto sea M, conociendo el criptograma correspondiente (pero no la clave).

posible emparejamiento), de las cuales cada participante deberá obtener $n-1$ para comunicarse con cada una de las otras personas y ahí es donde aparece el problema: ¿qué canal de comunicación seguro utilizan para compartirlas? Si dos personas tienen un canal para comunicarse a distancia de manera segura, bien podrían utilizar este canal para todos los mensajes en lugar de sólo para compartir la clave (sin necesidad de usar criptografía), y si no lo tuvieran, la situación es aún peor, pues se verían obligadas a encontrarse físicamente para compartir la clave, lo que no siempre es posible.

La criptografía de clave pública no presenta este problema. Para un grupo de n personas que se quieran comunicar entre ellos, se requieren un total de $2n$ claves diferentes (cada persona genera su propia clave privada y la correspondiente clave pública), de las cuales cada participante deberá almacenar su propio par de claves y las $n-1$ claves públicas restantes. De todas ellas, sólo necesita almacenar de forma segura su clave privada, por tanto, las claves públicas no necesitan ser intercambiadas de forma segura, pues, como su nombre indica, pueden hacerse públicas sin comprometer la seguridad del criptosistema, así que, en este caso, no aparece el problema del intercambio de claves.

De esta manera, si Alicia quiere enviar un mensaje cifrado a Benito, sólo necesita obtener su clave pública y utilizarla para cifrar el mensaje en cuestión. La única forma de descifrar el criptograma obtenido será mediante la clave privada de Benito (que sólo él posee).

Otra ventaja de los sistemas de clave pública es que también pueden ser usados para "firmar" un mensaje. Puesto que si un mensaje se cifra con la clave privada (en lugar de con la pública), cualquiera que tenga la clave pública podrá descifrarlo. Por tanto, si Benito quiere firmar un mensaje, sólo necesita cifrarlo con su clave privada y, como nadie más posee dicha clave, resultará evidente que él ha sido quien lo ha firmado.

La mayoría de criptosistemas de clave pública están basados en problemas matemáticos duros, de tal forma que la clave privada proporcione la información necesaria para resolverlos. Por ejemplo, tomar dos números primos grandes (de varios centenares de dígitos) y multiplicarlos es trivial, sin embargo factorizar el resultado obtenido es prácticamente imposible. Éste es el problema en el que basa su seguridad el criptosistema RSA (de Rivest, Shamir

y Adleman [RSA78]).

Otro problema muy utilizado en criptografía de clave pública es el problema del logaritmo discreto o DLP (*Discrete Logarithm Problem*). En este caso, realizar una exponenciación módulo un cierto número primo es fácil, pero encontrar a qué exponente hay que elevar un número para que dé otro, también módulo un cierto primo, es un problema intratable (equivalente al problema de la factorización).

Un inconveniente del DLP (así como de la factorización) es el tamaño de las claves que generan, ya que se necesitan miles de bits para alcanzar la seguridad de los sistemas simétricos. Como veremos, esto no es un problema para la criptografía utilizada en ordenadores, pero es un importante impedimento para usarlos en sistemas RFID. La versión para curvas elípticas del DLP no presenta este inconveniente, pues admite claves mucho más cortas, para mismos niveles de seguridad.

El problema de los sistemas asimétricos, en comparación con los simétricos, es su lentitud. Pues, al estar basados en problemas matemáticos, las operaciones que necesitan hacer suelen ser de mayor dificultad que las usadas en criptosistemas simétricos (que suelen basarse en la aplicación reiterada de operaciones muy sencillas a nivel de bit).

Para solucionar esto, una alternativa muy usada es combinar ambos sistemas: Si Alicia quiere enviar un mensaje de forma segura a Benito, primero escogerá una clave para algún sistema simétrico (llamada clave de sesión), cifrará el mensaje usando esa clave, y luego cifrará la clave utilizada con la clave pública de Benito (lo que resultará sencillo, puesto que la clave de sesión suele ser muy pequeña). Lo que se enviará a Benito será el mensaje cifrado con la clave de sesión y la clave de sesión cifrada con su clave pública. Para descifrar el mensaje, Benito tendrá que descifrar la clave de sesión (usando su clave privada) y usar dicha clave para descifrar el mensaje. Nótese que, si se desea compartir más de un mensaje en una misma sesión se puede seguir usando la misma clave.

Una idea similar se puede aplicar a las firmas para evitar tener que cifrar el mensaje entero con la clave privada. Basta con obtener un resumen del mensaje (mediante una función *hash*, que cumpla ciertos requisitos de seguridad), firmar

sólo el resumen, y luego enviar el mensaje junto con la firma del resumen. Otra ventaja de este método es que alguien que no necesite verificar la firma puede leer el mensaje directamente.

3.2. Introducción a la criptografía sobre curvas elípticas

Una curva elíptica [Kob84] es una curva algebraica de género uno. Toda curva elíptica se puede escribir como una ecuación cúbica que no tenga ni vértices ni autointersecciones. A pesar de su nombre, las curvas elípticas no son elipses. La gráfica de una curva elíptica sobre los números reales puede tener uno o dos componentes, dependiendo de si su discriminante es positivo o negativo.

Las curvas elípticas sobre los cuerpos de característica 0 (los racionales y sus extensiones) han tenido una gran importancia en las matemáticas, especialmente en la teoría de números. Como ejemplo, el teorema de Fermat-Wiles[2] (mal llamado "último teorema de Fermat", pues Fermat nunca mostró su demostración y hoy en día se considera muy poco probable que, en caso de existir, estuviera libre de errores) hace uso de cierto tipo de curvas elípticas definidas sobre $\mathbb{Q}$, como podemos ver en su extenso artículo [Wil95].

El motivo por el que este tipo de cuerpos no se utilizan en criptografía es que, al ser infinitos, no se podría garantizar que todos los parámetros involucrados en el proceso criptográfico tuvieran un tamaño acotado.

En 1985, Neal Koblitz [Kob87] y Victor Miller [Mil86] introdujeron la criptografía de curvas elípticas [BSS99]. En el ámbito de la criptografía, las curvas elípticas se toman sobre un cuerpo finito $\mathbb{F}_q$, siendo éste, casi siempre, primo ($\mathbb{F}_p$) o binario ($\mathbb{F}_{2^n}$). En algunos criptosistemas también se toman curvas definidas sobre un anillo u otras estructuras algebraicas. Además, las curvas elípticas sobre cuerpos finitos también pueden utilizarse para la factorización de enteros [Len87].

[2]El teorema de Fermat-Wiles dice que no existen tres enteros positivos a, b y c que satisfagan la ecuación $a^n + b^n = c^n$ para ningún valor de n mayor que dos.

Una curva elíptica E definida sobre un cuerpo $\mathbb{F}_q$ consiste en el conjunto de todos los puntos de la forma $P = (x, y) \in \mathbb{F}_q \times \mathbb{F}_q$ que satisfacen una ecuación del tipo

$$E/\mathbb{F}_q : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \text{ con } a_i \in \mathbb{F}_q, \tag{3.1}$$

y tal que el discriminante sea no nulo (para que no tenga puntos singulares), junto con el punto del infinito $\mathcal{O}$.

Si la característica del cuerpo es distinta de 2 y 3 se pueden realizar una serie de cambios de variable hasta obtener la llamada forma reducida de Weierstraß:

$$E/\mathbb{F}_q : y^2 = x^3 + ax + b, \text{ con } a, b \in \mathbb{F}_q, \tag{3.2}$$

el discriminante de la cual se puede calcular como $\Delta = -16(27b^2 + 4a^3)$.

Es posible definir una operación binaria entre puntos, llamada suma de puntos, cuyo elemento neutro es $\mathcal{O}$. El conjunto de puntos con esta operación forma un grupo abeliano.

La suma de puntos se define a partir de los métodos de la cuerda y la tangente, los cuales para el caso de curvas elípticas definidas sobre el cuerpo de los reales ($\mathbb{R}$) puede ser fácilmente visualizado, como se muestra en la Figura 3.1.

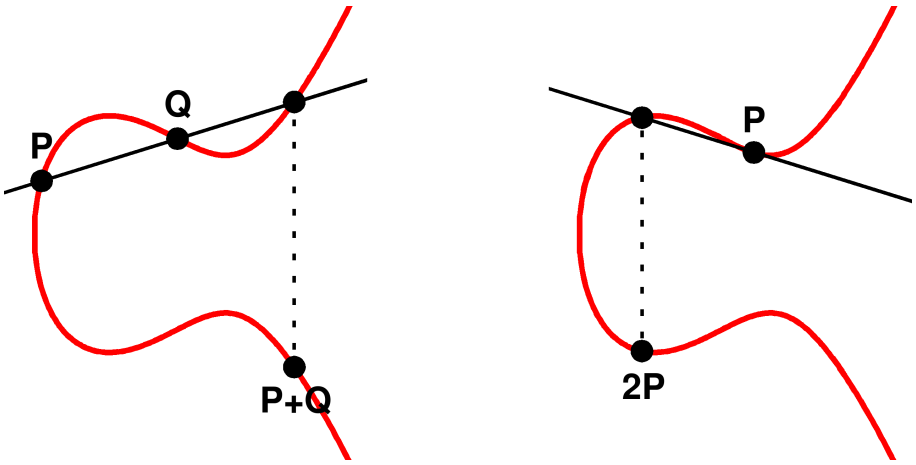


Figura 3.1: Métodos de la cuerda y la tangente.

Para sumar dos puntos (distintos), se traza una recta que pase por ambos

puntos. Esta recta, al ser la curva elíptica una curva cúbica pasará forzosamente por un tercer punto. El simétrico de este tercer punto, respecto el eje de abscisas, es el resultado de la suma. Para sumar dos puntos iguales (doblado de puntos), se traza una tangente al punto y se continua como en el caso anterior.

Dados dos puntos de la curva $P = (x, y)$, y $Q = (x, -y)$, la cuerda que pasa por ambos puntos (o la tangente a la curva, si ambos puntos coinciden) no corta la curva en ningún otro punto afín, sino que la corta en el punto del infinito $\mathcal{O}$. Su suma, por tanto, es el elemento neutro. Esto quiere decir que Q es el inverso de P, i.e. $Q = -P$.

Estos cálculos se pueden realizar fácilmente de forma algebraica. Para una curva en forma reducida de Weierstraß, sean $P = (x_1, y_1)$ y $Q = (x_2, y_2)$. Si $Q = -P$, entonces $P + Q = \mathcal{O}$, en caso contrario, $P + Q = (x_3, y_3)$, donde:

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= \lambda(x_1 - x_3) - y_1 \end{aligned}$$

y donde

$$\begin{aligned} \lambda &= \frac{y_2 - y_1}{x_2 - x_1}, \quad \text{si } P \neq Q \\ \lambda &= \frac{3x_1^2 + a}{2y_1}, \quad \text{si } P = Q \end{aligned} \tag{3.3}$$

Por ejemplo, dada la curva $E/\mathbb{F}_{13} : y^2 = x^3 + x + 1$ y los puntos $P = (0, 1)$ y $Q = (1, 4)$, se tiene: $P + Q = (8, 1)$ y $2P = P + P = (10, 7)$.

Sin embargo, las Ecuaciones 3.3 de suma y doblado tienen un pequeño inconveniente: requieren calcular un inverso en el cuerpo finito (para la división en la obtención de λ), y esa operación es considerablemente más lenta.

Existen otros sistemas de coordenadas que no tienen este problema. Son los sistemas de coordenadas proyectivas. Las coordenadas proyectivas utilizan una tercera variable[3]: se pasa de (x, y) a $(X : Y : Z)$, de manera que cada punto afín se pueda representar de diversas maneras.

Por ejemplo, en coordenadas homogéneas, $(x, y) = (kx : ky : k)$, para todo

[3]Algunos sistemas emplean más de tres: el sistema Jacobiano modificado usa cuatro coordenadas y el sistema Jacobiano Chudnovsky utiliza cinco.

$k \in \mathbb{F}_q^*$; así que, el punto $(X : Y : Z)$ corresponde al punto afín $(\frac{X}{Z}, \frac{Y}{Z})$. Los puntos con $Z = 0$, que, por razones obvias, no corresponden a ningún punto afín, son los llamados puntos del infinito, salvo el $(0 : 0 : 0)$, que no se considera un punto proyectivo válido. El punto del infinito de todas las curvas elípticas es precisamente el $(0 : 1 : 0)$.

En la Ecuación 3.4 se muestra la obtención de la forma reducida de Weierstraß en coordenadas homogéneas.

$$\begin{aligned} E/\mathbb{F}_q : y^2 &= x^3 + ax + b, \quad \text{se sustituye } x \text{ por } \frac{X}{Z} \text{ e } y \text{ por } \frac{Y}{Z} \\ \frac{Y^2}{Z^2} &= \frac{X^3}{Z^3} + a\frac{X}{Z} + b, \quad \text{se multiplica por } Z^3 \\ Y^2Z &= X^3 + aXZ^2 + bZ^3 \end{aligned} \tag{3.4}$$

Tabla 3.1: Operaciones básicas necesarias para la suma y el doblado de puntos.

	Suma de puntos			Doblado de puntos		
Sistema de coordenadas	Mult.	Cuad.	Sumas	Mult.	Cuad.	Sumas
Proyectivas Jacobianas	15	5	7	5	5	4
López-Dahab	4	1	2	2	5	1
López-Dahab modificado	6	1	2	3	5	1

En la Tabla 3.1 se muestran el número de multiplicaciones, cuadrados y sumas que hay que realizar en el cuerpo finito, para el cómputo de la suma y el doblado de puntos, en coordenadas proyectivas Jacobianas [CC86], coordenadas de López-Dahab [LD99] y coordenadas de López-Dahab modificado [TB06]. Para algunas curvas estos costes pueden ser aún menores.

Las implementaciones de los criptosistemas basados en curvas elípticas suelen utilizar coordenadas proyectivas, dejando al implementador la decisión de que sistema es el más adecuado según la situación.

Una vez definida la suma, un punto $Q \in E(\mathbb{F}_q)$ se puede sumar consigo mismo varias veces (i.e. multiplicarlo por un escalar):

$$\underbrace{Q + \ldots + Q}_{e \text{ veces}} = eQ = P. \tag{3.5}$$

El conjunto de puntos que pueden ser obtenidos multiplicando Q por un escalar forman un subgrupo, del que Q es un generador.

El problema inverso (i.e. dados dos puntos P y Q, encontrar un e tal que $P = eQ$), llamado problema del logaritmo discreto sobre curvas elípticas o ECDLP (de *Elliptic Curve Discrete Logarithm Problem*), resulta ser computacionalmente difícil de resolver. Hay varios criptosistemas [Mil86, Kob87] cuya seguridad se basa en la intratabilidad del problema ECDLP.

El principal interés del ECDLP, comparado con el problema del logaritmo discreto para el grupo multiplicativo de un cuerpo finito (DLP), es que existen algoritmos subexponenciales, para resolver el DLP en grupos multiplicativos, pero estos no pueden ser usados para resolver el ECDLP.

El mejor algoritmo para resolver el DLP en un cuerpo finito $\mathbb{F}_q$ es el *index calculus* [AD94]. Su coste computacional es subexponencial, de orden $L_q[1/2, c] = O(e^{(c+o(1))(\ln q)^{1/2}(\ln \ln q)^{1/2}})$, para algún $(c > 0)$. Para cuerpos binarios $\mathbb{F}_{2^m}$ existe una mejora, hecha por Don Coppersmith, cuyo coste es $L_{2^m}[1/3, c] = O(e^{(c+o(1))(\ln 2^m)^{1/3}(\ln \ln 2^m)^{2/3}})$, para $0 < c < 1{,}587$.

En cambio, el algoritmo más rápido para resolver el ECDLP es el *baby-step giant-step* [Sha72], que es un algoritmo genérico para resolver el DLP en cualquier grupo cíclico (otros algoritmos pueden ser mejores para casos específicos, como son el algoritmo *rho* de Pollard o el de Pohlig-Hellman). Tiene un coste computacional exponencial de $O(e^{\frac{\ln n}{2}})$, donde n es el orden del grupo (en el caso elíptico, se refiere al grupo de puntos generado por Q, la "base" del logaritmo). De hecho, el tamaño de n estará en torno al tamaño del cuerpo, q (teorema de Hasse [Has33]). Debido a esto, el ECDLP resulta ser un problema más difícil.

Desde un punto de vista práctico, este hecho se traduce en que si se usa el ECDLP como base para hacer criptografía se pueden utilizar claves más cortas dando la misma seguridad que el DLP.

La Tabla 3.2 muestra el número de bits necesarios para alcanzar los mismos niveles de seguridad en criptosistemas que basan su seguridad en el DLP, com-

parados con aquellos que están basados en el ECDLP, y con los criptosistemas simétricos (como por ejemplo el estándar actual AES).

Tabla 3.2: Bits necesarios para los mismos niveles de seguridad.

DLP	ECDLP	Simétrico
512 bits	112 bits	56 bits
1024 bits	160 bits	80 bits
2048 bits	224 bits	112 bits
3072 bits	276 bits	138 bits
4096 bits	320 bits	160 bits
6144 bits	390 bits	195 bits
8192 bits	448 bits	224 bits
12288 bits	552 bits	276 bits
16384 bits	640 bits	320 bits

En la tabla podemos ver que los criptosistemas de clave pública clásica necesitan claves mucho mayores que aquellos basados en el logaritmo discreto sobre curvas elípticas o los criptosistemas simétricos. Esa diferencia se acentúa a medida que crecen las claves (lo que probablemente hará la diferencia abismal en el futuro), pues al doblar las claves de un criptosistema público clásico se obtiene una seguridad similar a la correspondiente a multiplicar la longitud de uno basado en el ECDLP por $\sqrt{2}$.

Por su parte, los criptosistemas basados en el ECDLP necesitan claves de "sólo" el doble de bits que un criptosistema simétrico con la misma seguridad.

Para sistemas RFID es impensable (al menos actualmente) utilizar criptografía pública clásica, pues sus enormes claves sobrepasan las capacidades de los chips de las etiquetas. Y la criptografía simétrica necesita que el lector y las etiquetas compartan una clave común, lo que implica que, en caso de que un enemigo obtenga la clave de una etiqueta, se comprometa todo el sistema.

Por otra parte, los sistemas RFID basados en criptografía ligera (cuyas primitivas criptográficas suelen ser operaciones lógicas muy sencillas o funciones *hash*), suelen tener una seguridad muy limitada o problemas de escalabilidad.

Así pues, en la presente tesis planteamos el uso de criptosistemas basados en el logaritmo discreto sobre curvas elípticas, analizando su viabilidad y

seguridad.

En la actualidad, la seguridad óptima para sistemas convencionales se considera alrededor de 2048 bits para la criptografía de clave pública clásica, 224 bits para el logaritmo discreto sobre curvas elípticas y 112 bits para los criptosistemas simétricos. Sin embargo, dadas las fuertes restricciones de las etiquetas, en sistemas RFID nos vemos obligados a rebajar las exigencias de seguridad lo máximo posible, con el fin de balancear nivel de seguridad y eficiencia de la implementación. Como se verá en el Capítulo 4, nosotros trabajaremos en el cuerpo $\mathbb{F}_{2^{137}}$.

De todas formas, no todas las curvas de un mismo cuerpo son igual de seguras. Para que el logaritmo discreto no sea fácil de resolver en una curva, se requiere trabajar en un subgrupo de puntos cuyo orden (cantidad de puntos del subgrupo) sea un primo grande, i.e. con una magnitud similar al tamaño del cuerpo. Por tanto, se requiere que la curva tenga un cardinal (número de puntos de la curva) que factorice en, como mínimo, ese primo grande.

Una curva que cumpla ese requisito tendrá un cardinal de la forma $\#E = h \cdot p$, donde p es el mencionado primo grande y h (no necesariamente primo) es muy pequeño. Para obtener un subgrupo de puntos de orden p, se empieza por elegir un punto aleatorio P y se calcula el punto $Q = hP$. Si $Q \neq \mathcal{O}$ entonces Q es un punto de orden p, en caso contrario, basta con repetir el procedimiento con otro punto aleatorio, tantas veces como sea necesario (lo normal es obtenerlo con muy pocos intentos).

Visto que la seguridad criptográfica de una curva elíptica está directamente relacionada con su cardinal, se plantea el problema adicional de asegurarse de que la curva sobre la que se trabajará tenga un buen cardinal. No se puede escoger una curva de manera que tenga un cardinal concreto (a no ser que partamos de una que ya lo tenga, como se verá en la siguiente sección), por tanto, la alternativa consiste en escoger curvas aleatoriamente y comprobar si su cardinal es adecuado.

Sin embargo, calcular el cardinal de una curva elíptica no es tarea fácil. El mejor algoritmo hasta la fecha, el SEA, que fue creado por Schoof [Sch95] y mejorado por Elkies y luego por Atkin, tiene un coste computacional de $O(\log^4 q)$, donde q es el tamaño del cuerpo en el que está definida la curva.

A esto hay que sumar el hecho de que probablemente haya que repetir este procedimiento bastantes veces, hasta dar con una curva cuyo cardinal factorice en un primo grande.

Para solucionar esto, el SECG[4] publicó en el año 2000 [Cer00] una lista de curvas precalculadas que cumplen con todos los requisitos de seguridad para diversos tamaños de cuerpos finitos, tanto primos como binarios. En la última versión del documento, que data del 2010 [Cer10], han eliminado las curvas correspondientes a los cuerpos más pequeños.

3.2.1. Isogenias de curvas elípticas

Supongamos que un atacante está intentando romper el logaritmo discreto en una curva elíptica para obtener la clave secreta. Si se cambia de clave y de curva, el atacante, para obtener la nueva clave, se verá obligado a empezar todo el trabajo desde cero (aunque nada le impide seguir intentando obtener la clave antigua, si así lo desea). En cambio, si sólo se cambia la clave, el atacante podrá aprovechar parte del trabajo realizado para obtener la clave anterior para la obtención de la nueva.

Así, es conveniente cambiar cada cierto tiempo, no sólo de clave secreta, sino también de curva elíptica, para que los atacantes no se puedan beneficiar de los intentos fallidos.

Sin embargo, la cantidad de curvas proporcionadas por el SECG es muy pequeña: sólo hay entre una y tres curvas para cada tamaño de cuerpo primo y otras tantas para cada tamaño de cuerpo binario.

Existe una solución, cuyo único requisito consiste en tener al menos una curva inicial buena: las isogenias de curvas elípticas.

Una isogenia entre dos curvas elípticas E y E' es un morfismo racional $\mathcal{I} : E \to E'$, de manera que se cumpla que $\mathcal{I}(\mathcal{O}_E) = \mathcal{O}_{E'}$ [BSS99]. Entonces, una l-isogenia es una isogenia para la que se cumple que el conjunto de puntos cuya imagen es el neutro (denominado *kernel*) es un subgrupo de orden l.

[4]El SECG (*Standards for Efficient Cryptography Group*) es un consorcio internacional fundado en 1998 por Certicom al que también pertenecen Entrust, Fujitsu, el NIST, Pitney Bowes, Unisys y Visa. El grupo desarrolla estándares comerciales para hacer la criptografía de curvas elípticas interoperable y eficiente.

Una propiedad muy interesante de las isogenias es que dos curvas isógenas siempre tienen el mismo cardinal. La utilidad de esta propiedad consiste en que, como se ha dicho, la seguridad de la curva depende del cardinal; por tanto, si tenemos una curva con un cardinal criptográficamente útil, cualquiera de sus isógenas será igual de segura (desde el punto de vista teórico). Además, dadas dos curvas –definidas sobre el mismo cuerpo– que tengan el mismo cardinal, siempre existirá una isogenia entre ellas. Esto significa que a partir de una curva criptográficamente útil deberíamos ser capaces de obtener todas las otras curvas con el mismo cardinal, aplicando isogenias reiteradamente.

En particular, dos curvas son llamadas l-isógenas (para un l específico) si existe una l-isogenia entre ellas. Además, todas las curvas elípticas alcanzables mediante l-isogenias (para un l primo fijo) pueden ser clasificadas fácilmente en un grafo llamado l-volcán [FM02, MMS+06], donde los nodos representan clases de isomorfía de curvas elípticas y las aristas las l-isogenias entre ellas.

Este grafo, presenta una jerarquía de niveles de manera que las l-isogenias pueden ser horizontales, ascendentes o descendentes dependiendo de si, partiendo de una curva elíptica, llegan a otra del mismo nivel, del nivel superior o del inferior, respectivamente.

Como una de las aportaciones de esta tesis es un protocolo que utiliza curvas elípticas, y que, al tratarse de un sistema para RFID, se considera necesario utilizar claves relativamente pequeñas (por las restricciones físicas de las etiquetas), se plantea la necesidad de disponer de un listado de curvas elípticas seguras para poder cambiarlas con cierta regularidad.

Debido a ello, en el Capítulo 5 también se propone un método para obtener dichas curvas más rápidamente mediante la paralelización de un algoritmo de generación de volcanes.

3.3. Protocolos de autenticación de conocimiento nulo

Un protocolo de conocimiento nulo (o demostración de conocimiento nulo) es un método interactivo que permite que una entidad demuestre a otra que

una afirmación es cierta, sin revelar nada, salvo la veracidad de la propia afirmación. La entidad que pretende demostrar esta afirmación es el probador y quien la comprobará es el verificador. Por comodidad, al probador se le llamará Peggy (o simplemente P) y al verificador, Víctor (o V).

Existe un ejemplo muy sencillo [QGB90] de protocolo de conocimiento nulo, que sirve para ilustrar el funcionamiento típico de estos protocolos, y que podemos ver en la Figura 3.2.

En este ejemplo, Peggy sabe la palabra secreta para abrir una puerta mágica en el interior de una cueva con forma de anillo. La entrada a la cueva está en el lado opuesto a la puerta mágica. Víctor está dispuesto a pagar por la palabra secreta pero sólo si está seguro de que Peggy realmente la conoce. Peggy, a su vez, está dispuesta a revelar la palabra, pero no antes de haber cobrado por ella.

Idean un método para que Peggy pueda demostrar que conoce la palabra sin revelarla: Peggy entra en la cueva y escoge el camino izquierdo o derecho sin que Víctor lo vea. Luego Víctor grita el camino por el que quiere ver salir a Peggy. Si éste coincide con el camino por el que Peggy entró, ella se limitará a salir por donde entró; si es el otro camino, deberá usar la palabra secreta para cruzar la puerta mágica del interior y así poder salir por el camino solicitado.

En el caso de que Peggy hubiese mentido y no conociese la palabra, sólo podría volver por el camino elegido por Víctor si fuera el mismo por el que ella entró. Considerando que las dos elecciones de camino son aleatorias, Peggy tiene un 50 % de posibilidades de acertar el camino por el que Víctor le pedirá que salga. Sin embargo, si este procedimiento se repite varias veces, la probabilidad de acertar todas las elecciones decrece considerablemente (2^{-n}, donde n es el número de veces que se repite el procedimiento). Así que, si Peggy aparece todas las veces por el camino correcto, muy probablemente conozca la palabra secreta.

La principal utilidad de los protocolos de conocimiento nulo es que pueden ser usados como mecanismo de autenticación: el probador P es el poseedor de un secreto s cuyo conocimiento tiene que ser demostrado al verificador V. Como P es el único capaz de realizar dicha demostración, V dará por buena la identidad de P. Además, el protocolo no revela ninguna información útil,

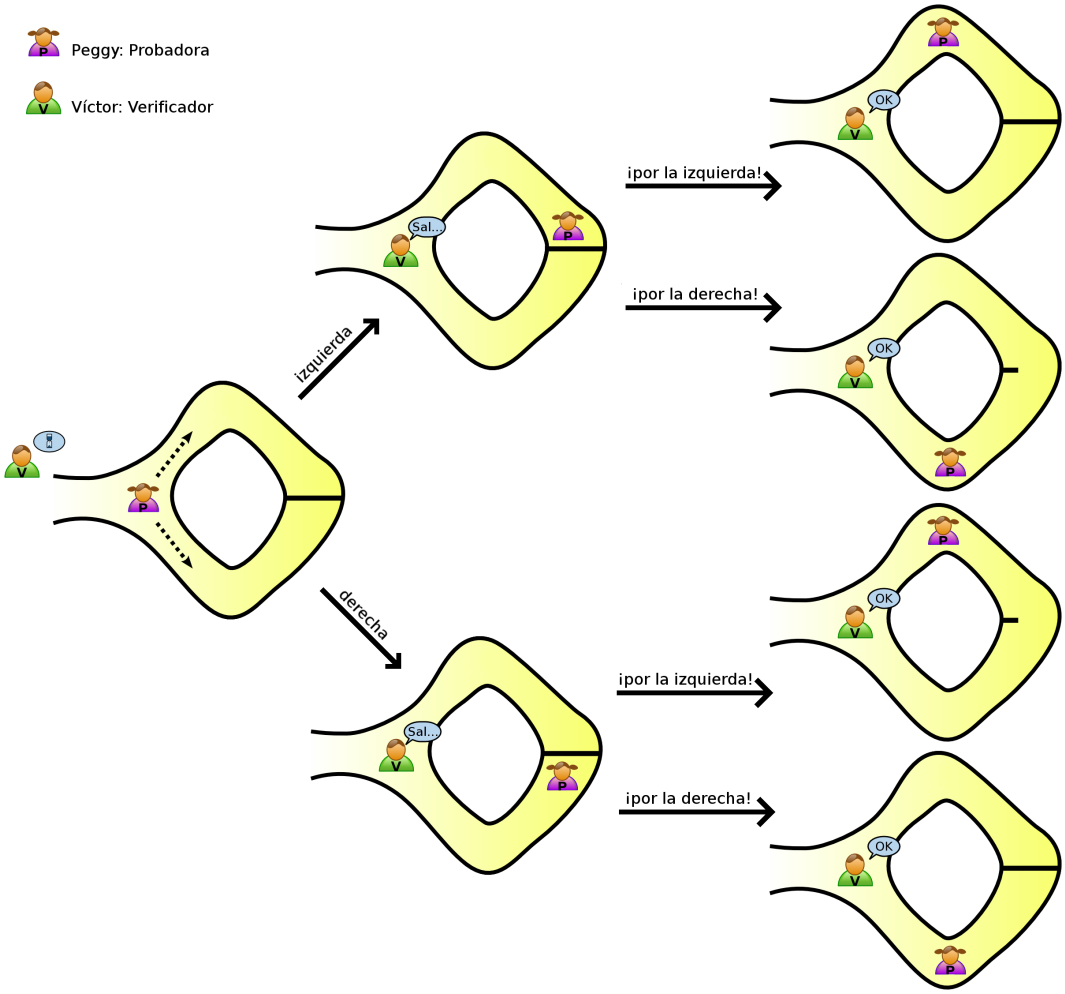


Figura 3.2: Ejemplo de protocolo de conocimiento nulo.

excepto que P sabe el secreto s, así que V o un posible atacante no obtendrán ninguna información adicional de s y, debido a ello, nunca serán capaces de hacerse pasar por P.

El procedimiento habitual consiste en que P envía a V un valor, llamado testigo, que define un rango de posibles preguntas (o retos) relacionadas con s, que solamente el poseedor de s (P) puede responder, pero de tal forma que la respuesta a cualquiera de ellas no revele ni el secreto ni parte del mismo. Luego, V envía uno de estos retos a P, y éste, enviará la respuesta a V.

Para hacer esto, normalmente P y V intercambian mensajes dependientes

de números aleatorios, durante una o más rondas, tras las cuales, si la respuesta –o respuestas– de P son correctas, V aceptará, con cierta probabilidad de estar en lo cierto, la prueba de posesión de s por parte de P. En cambio, si hay alguna respuesta inválida, la rechazará. Esta probabilidad proviene del hecho de que un atacante podría hacerse pasar por P si fuera capaz de acertar qué se le va a preguntar.

Un protocolo de conocimiento nulo debe satisfacer las siguientes tres propiedades [MvOV96]:

1. **Completitud**: un probador honesto, que conozca el secreto, será capaz de demostrar a un verificador honesto que lo conoce.

2. **Solidez**: un probador tramposo, que no conozca el secreto, sólo tendrá una pequeña probabilidad de convencer al verificador de que lo conoce.

3. **Conocimiento nulo**: un verificador tramposo, que escuche una demostración válida, no aprenderá nada más, a parte de que el probador es propietario del secreto.

La propiedad de conocimiento nulo equivale a que un verificador puede implementar un simulador con el que, a partir de la afirmación a demostrar, y sin acceso a ningún probador, puede producir una transcripción que *parece* una interacción válida con un probador honesto.

Se tiene un conocimiento nulo perfecto cuando las distribuciones producidas por el simulador y el protocolo real son exactamente las mismas. El conocimiento nulo estadístico significa que ambas distribuciones son estadísticamente cercanas, i.e. su diferencia estadística es una función despreciable. Finalmente, el conocimiento nulo computacional implica que ningún algoritmo eficiente[5] puede distinguir las dos distribuciones (son computacionalmente indistinguibles) [MvOV96].

En criptografía, es suficiente el conocimiento nulo computacional, puesto que el secreto a demostrar se podría obtener igualmente mediante algoritmos no eficientes, por tanto un conocimiento nulo más estricto no aporta mucho.

[5]Específicamente, un algoritmo probabilístico de tiempo polinómico con una probabilidad de error menor a $\frac{1}{2}$ para todas las instancias (i.e. perteneciente a la clase de complejidad PP).

Existen varios protocolos de conocimiento nulo. De hecho, se podrían usar para ello todos los problemas NP-completos (como por ejemplo demostrar el conocimiento de un ciclo Hamiltoniano en un grafo grande), y también algunos problemas intratables, como el logaritmo discreto [CEvdGP87, Sch91] o la factorización y la obtención de raíces cuadradas modulares [FFS87].

El protocolo de conocimiento nulo que se utiliza en el Capítulo 4 de esta tesis, es el protocolo de Schnorr [Sch91], que se expone en la siguiente subsección. El motivo de la elección de este protocolo es que es fácilmente adaptable al uso de curvas elípticas. Nótese que, en su versión elíptica, su seguridad se basa en el problema ECDLP, por lo que se podrán utilizar claves pequeñas, lo que permitirá encajar en las restricciones de implementación de las etiquetas RFID.

3.3.1. Protocolo de Schnorr elíptico

El protocolo Schnorr elíptico utiliza los siguientes parámetros públicos: un cuerpo finito $\mathbb{F}_q$, una curva $E/\mathbb{F}_q$ y un generador Q de orden d. El secreto del probador es $s \in [2, d-1]$. Y su clave pública es $P \in E(\mathbb{F}_q)$ y se obtiene como $P = sQ$.

Durante el proceso de prueba, el probador elige aleatoriamente un valor $r \in [2, d-1]$, llamado compromiso, a partir del cual calcula el testigo $W = rQ$, que envía al verificador. Éste elige otro valor aleatorio $c \geq 1$, como reto y lo envía al probador. Finalmente, el probador calcula la respuesta $a = r + cs$ y la envía al verificador.

La prueba de conocimiento nulo se aceptará como válida si y sólo si se verifica $aQ - cP = W$.

Veamos que, efectivamente, cumple las tres propiedades requeridas en un protocolo de conocimiento nulo:

Completitud. Como se ha dicho, la prueba requiere que se verifique la igualdad $aQ - cP = W$. Nótese que $aQ - cP = (r + cs)Q - c(sQ) = rQ + csQ - csQ = rQ = W$, por lo que si la respuesta es correcta, la igualdad se cumplirá. Por otra parte, para ser capaz de calcular la respuesta $a = r + cs$, el probador debe conocer realmente el secreto s y

el logaritmo elíptico base Q de W (que es r), pues no sabe qué reto c le enviará el verificador hasta después de haberle enviado W. Por tanto, si el probador envía la respuesta correcta al verificador, éste aceptará que el probador conoce el secreto; i.e. el protocolo es completo.

Solidez. Un probador que no conozca el secreto puede intentar adivinar el reto. Sea c' su predicción, no tiene más que inventarse la respuesta a enviar a' y calcular W como $a'Q - c'P$ y enviarlo. Si acierta, y el reto enviado c es igual a c', entonces puede enviar su respuesta a' y la verificación será un éxito. La probabilidad de que acierte es 2^{-n}, donde n es la longitud del reto; así que, si éste es suficientemente grande, el protocolo es sólido.

Conocimiento nulo. La respuesta, a, del probador no revela información: si el verificador conociera r, podría calcular s como $\frac{a-r}{c}$, pero para obtener r debería resolver el logaritmo de W. Por otra parte, cualquiera puede realizar un simulador del protocolo, pues como se ha visto, basta con escoger un reto c, una respuesta a y calcular $W = aQ - cP$ (que es precisamente la igualdad a verificar), y se obtendrá un triplete (W, c, a) válido, sin necesidad de conocer el secreto. Por tanto, el protocolo es de conocimiento nulo.

Dado que el protocolo Schnorr elíptico cumple las características mencionadas, éste se ha utilizado como mecanismo de autenticación del lector a las etiquetas para la capa de aplicación de un sistema RFID (expuesto en el Capítulo 4 de esta tesis).

Capítulo 4

Protocolo seguro para la capa de aplicación

Como se ha dicho en el Capítulo 2, la seguridad de un sistema RFID depende de la seguridad de cada una de sus capas. En este capítulo se propone un método para proporcionar seguridad a la capa de aplicación de un sistema RFID. Este método se avanzó en el Capítulo 2 con el nombre de *Elliptic Point Chains*.

La capa de aplicación requiere un protocolo capaz de garantizar el envío seguro de, al menos, un identificador pequeño (el número de bits necesario dependerá del sistema concreto). Además, para que el sistema sea útil para redes de sensores se necesita también poder enviar cifrados los datos obtenidos por el sensor.

Basándonos en estos requisitos, proponemos un protocolo que hace uso de la criptografía de curvas elípticas, para proteger el identificador de las etiquetas, y de un mecanismo de autenticación de conocimiento nulo, para establecer el diálogo entre las etiquetas y el lector. Esto proporciona un protocolo que evita el filtrado de datos, el seguimiento y las suplantaciones (tanto del lector como de las etiquetas), es resistente a los ataques de denegación de servicio, y proporciona *forward security*.

A lo largo del presente capítulo, se expondrá una descripción del protocolo propuesto para la capa de aplicación (Sección 4.1), los detalles de implementación del mismo (Sección 4.2) y un análisis de seguridad (Sección 4.3).

Este protocolo dio lugar a dos publicaciones científicas: *An Elliptic Curve and Zero Knowledge Based Forward Secure RFID Protocol* [MVR+07] y *A Secure Elliptic Curve-Based RFID Protocol* [MVR+09]. Este último expone un extenso análisis de seguridad del protocolo.

4.1. Descripción del protocolo

En el protocolo que se propone intervienen tres elementos: un conjunto de etiquetas RFID, un lector RFID y una base de datos. En un entorno real lo más probable es que haya más de una etiqueta, sin embargo, este protocolo pertenece a la capa de aplicación y por tanto considera solucionado el problema de la singulación (i.e. escoger con cual de todas las etiquetas se realiza la comunicación) mediante algún mecanismo de control de acceso al medio en la capa de comunicación.

El protocolo, cuyo diagrama puede verse en la Figura 4.1, usa la variante sobre curvas elípticas del protocolo de Schnorr combinada con un mecanismo para actualizar el secreto de la etiqueta. La idea clave es que el lector se autentique ante la etiqueta antes de recibir la identificación de la misma, el cual es calculado a partir de un secreto que se cambia después de cada operación de lectura.

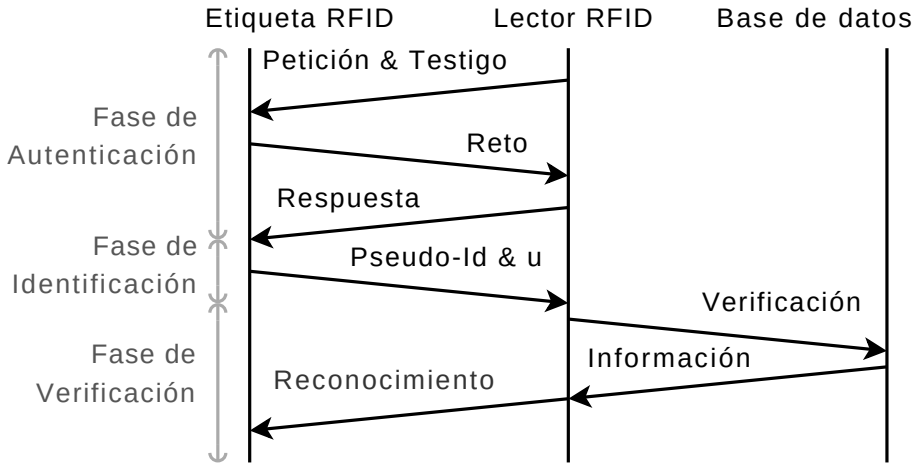


Figura 4.1: Diagrama del protocolo.

El protocolo consta de las siguientes cuatro fases: (a) fase de *setup*, (b) fase de autenticación del lector, (c) fase de identificación de la etiqueta y, (d) fase de verificación de la etiqueta.

(a) Fase de *setup*

En el *setup* del sistema se generan los siguientes parámetros públicos y secretos:

- Un cuerpo finito $\mathbb{F}_q$ y una curva elíptica $E/\mathbb{F}_q$ definida sobre este cuerpo finito.
- Un generador Q de un subgrupo cíclico de puntos de la curva elíptica, de orden d, en el cual resolver el ECDLP no sea factible.
- Un secreto $s \in [2, d-1]$, elegido como secreto del lector. El conocimiento de este valor deberá ser demostrado a las etiquetas durante la autenticación.
- La clave pública $P \in E(\mathbb{F}_q)$ del lector, calculada como $P = sQ$.
- Un punto secreto inicial $K_1^i \in E(\mathbb{F}_q)$ para cada etiqueta i, a partir del cual se calcularán los pseudo-identificadores. Este punto se cambiará al finalizar cada lectura.

Los tamaños específicos de los parámetros del protocolo serán discutidos en la Sección 4.2.1.

(b) Fase de autenticación del lector

Para prevenir la suplantación de un lector válido, se realiza una fase de autenticación, basada en el protocolo Schnorr elíptico (Sección 3.3.1), antes de la identificación de la etiqueta. Esta fase consta de los tres pasos siguientes:

1. El lector elige un valor aleatorio r (el *Compromiso*), con $2 \leq r \leq (d-1)$, calcula $W = rQ$ (el *Testigo*), y envía W a la etiqueta, junto con la señal de *Petición* de lectura.

2. La etiqueta elige un valor aleatorio, $c \geq 1$ (el *Reto*), de l bits de longitud, y lo envía al lector.

3. El lector calcula $a = r + cs$ (la *Respuesta*) y la envía a la etiqueta.

En este momento, la etiqueta aceptará que el lector es válido si y sólo si $aQ - cP = W$ (nótese que, de hecho, $aQ - cP = (r+cs)Q - c(sQ) = rQ = W$).

(c) Fase de identificación de la etiqueta

Este proceso tendrá lugar cada vez que un lector quiera leer una etiqueta, y haya sido autenticado con éxito. Cada etiqueta T_i mantiene un punto secreto K_j^i (para la lectura j-ésima), perteneciente a la curva elíptica $E(\mathbb{F}_q)$, el cual irá variando a lo largo del tiempo. Este punto será recalculado cada vez que la etiqueta sea identificada con éxito para prevenir que dos lecturas de la misma etiqueta puedan ser relacionadas entre ellas por parte de un adversario. Cada vez que una etiqueta se lee, ésta envía su *Pseudo-Id* actual, id_j^i, correspondiente a la etiqueta T_i y a la lectura j-ésima (el identificador real no se envía y tampoco se almacena en la etiqueta).

El proceso de identificación de la etiqueta i en la lectura j-ésima consta de los siguientes tres pasos:

1. La etiqueta calcula su *Pseudo-Id* como, $id_j^i = LB(x(K_j^i)) * LB(y(K_{j-1}^i))$, donde $x(K_j^i)$ e $y(K_{j-1}^i)$ son la abscisa y la ordenada del punto secreto actual y del previo, respectivamente, $LB(\cdot)$ devuelve algunos de los últimos bits de su entrada, y $*$ es una operación que no sea algebraica sobre $\mathbb{F}_q$, con el fin de dificultar el uso posterior de esta información por parte de un atacante. Esta operación podría ser una suma modular en el caso de un cuerpo binario o un *xor* bit a bit en el caso de un cuerpo primo. Para la primera lectura, se necesita una semilla que actúe como $y(K_0^i)$.

2. La etiqueta calcula su siguiente punto secreto $K_{j+1}^i = zQ$, donde z es el resultado de aplicar una cierta función f a la abscisa de K_j^i (i.e. $z = f(x(K_j^i))$). Esta función f debe permitir facilitar el cálculo del nuevo punto sin comprometer la seguridad.

3. La etiqueta envía su *Pseudo-Id* actual, id_j^i, al lector.

A pesar de haberlo calculado, la etiqueta aún no actualiza su punto secreto, pues para ello, por razones de seguridad, debe esperar a un mensaje de reconocimiento que se envía durante la fase de verificación.

(d) Fase de verificación de la etiqueta

En este momento, el lector ya ha recibido id_j^i, así que debe acceder a la base de datos para verificar la identidad de la etiqueta, así como para obtener la información asociada a ella. Para una identificación exitosa la base de datos de respaldo debe almacenar el siguiente identificador esperado para cada una de las etiquetas, i.e. todos los id_j^i para $i \in \{1, \ldots, n\}$ donde n es el número de etiquetas del sistema. El valor de j corresponderá al número de la siguiente lectura de cada etiqueta, por tanto, podrá diferir entre etiquetas. La base de datos también necesita almacenar los puntos secretos correspondientes K_j^i. Estos valores deberían ser guardados en una tabla *hash* para tener fácil acceso.

Así que, cuando un lector válido obtiene un *Pseudo-Id*, se procede de la siguiente manera:

1. El lector reenvía el *Pseudo-Id* a la base de datos (mensaje de *Verificación*).

2. La base de datos busca el *Pseudo-Id* en la tabla *hash*, cambia el punto secreto correspondiente de la misma forma en que lo hizo la etiqueta, elimina el *Pseudo-Id* viejo de la tabla *hash* e inserta el que será obtenido durante la siguiente lectura.

3. Al mismo tiempo, la etiqueta calcula el valor de reconocimiento esperado, $expected\_ack_j^i = MB(x(K_j^i)) * MB(y(K_{j+1}^i))$, donde $MB(\cdot)$ devolverá algunos de los bits centrales, evitando los bits previamente usados para la computación del *Pseudo-Id*, ya que un solapamiento podría, potencialmente, dar información a un atacante.

4. La base de datos calcula el valor de reconocimiento, ack_j^i, de la misma manera en que lo ha hecho la etiqueta y envía al lector un mensaje de *Información* que contiene el valor ack_j^i, además de la información del producto.

5. El lector envía el mensaje de *Reconocimiento* (que contiene ack_j^i) a la etiqueta.

En este momento, la etiqueta guardará el nuevo punto secreto K_{j+1}^i (calculado en la fase anterior) si y sólo si el reconocimiento recibido (ack_j^i) coincide con el que se esperaba ($expected\_ack_j^i$).

El motivo por el que se espera a actualizar el nuevo punto secreto a la llegada de un mensaje de *Reconocimiento* es, que en entornos donde sea posible que el lector no reciba el *Pseudo-Id* (porque haya ruido o atacantes bloqueando la señal), las etiquetas podrían haber sido actualizadas sin la correspondiente actualización de la base de datos.

Además, un atacante que interceptara y bloqueara el mensaje con el *Pseudo-Id* no podría calcular el reconocimiento correcto (pues éste depende de puntos secretos de la etiqueta), por lo que la etiqueta sólo aceptará un *Reconocimiento* recibido de un lector válido.

Obviamente, este mensaje también es susceptible de no llegar, pero esto sería un problema poco relevante, ya que en tal caso sería la etiqueta la que requeriría ser actualizada. La única consecuencia de este hecho es que la siguiente lectura de la etiqueta retornaría un *Pseudo-Id* viejo, por lo que la operación de lectura debería ser repetida hasta que la etiqueta alcanzase el *Pseudo-Id* que está almacenado en la base de datos.

En resumen, el *Reconocimiento* impide una actualización del secreto de la etiqueta sin la correspondiente actualización de la base de datos, lo cual sí sería un problema importante. De todas formas, en entornos en los que sea imposible para un atacante bloquear el mensaje con el *Pseudo-Id*, se puede prescindir de todo el proceso de reconocimiento, ya que en tal caso, la base de datos siempre se actualizará correctamente.

4.1.1. Extensión del protocolo para redes de sensores

En diversos escenarios la simple identificación no es suficiente y se necesita dotar a las etiquetas de algún tipo de sensor [CJ08, OF09]. Son las llamadas redes de sensores. Por ejemplo, Michelin decidió usar etiquetas RFID en sus ruedas para monitorizar la presión. En este ejemplo, si pensamos en carreras

de alta competición, no interesa que cualquiera pueda saber si llevamos las ruedas más o menos hinchadas (o incluso pinchadas) sino que este dato debería revelarse únicamente al ordenador de a bordo del vehículo.

El protocolo presentado admite la extensión de enviar estos datos provenientes de un sensor de forma segura. Esto lo hace conveniente para ser usado en redes de sensores. Este protocolo puede ser adaptado para proveer a la etiqueta la habilidad de enviar al lector pequeños datos cifrados además del *Pseudo-Id*.

Esto se puede conseguir con una forma muy simple de cifrado, usando el secreto de la etiqueta, K_j^i, como clave. De este modo, los datos adicionales a enviar, v_j^i, se cifran usando un *xor* bit a bit con los primeros bits de la ordenada del punto secreto: $u_j^i = v_j^i \; xor \; FB(y(K_j^i))$, donde $FB(\cdot)$ devolverá algunos de los bits iniciales. Como estos primeros bits no serán usados en el cálculo del *Pseudo-Id*, y tampoco deben usarse en el *Reconocimiento*, esto puede ser considerado como una *libreta de un solo uso* (*one-time pad*), y, por lo tanto, muy seguro. Los datos cifrados u_j^i se envían en el mismo mensaje que el *Pseudo-Id*, tal como se muestra en la Figura 4.1.

4.2. Aspectos de implementación

Tal como se mencionó en el Capítulo 1, en la implementación del protocolo se tienen que considerar las fuertes restricciones de memoria y cómputo de las etiquetas RFID pasivas, ya que estamos tratando con dispositivos muy pequeños y baratos.

Con el fin de conseguir una implementación factible y eficiente, en esta sección vamos a analizar dos aspectos básicos: la selección de parámetros apropiada para el protocolo propuesto y la factibilidad de la implementación en los sistemas RFID actuales.

4.2.1. Selección de parámetros

Hay algunos parámetros que necesitan ser seleccionados cuidadosamente para conseguir un buen balance entre el grado deseado de seguridad y un

buen funcionamiento. En esta sección se indica una selección específica para la longitud de los parámetros a ser usados durante el protocolo propuesto, así como para la función f que interviene en la fase de identificación.

(a) El cuerpo finito, la curva elíptica y el generador

Para el cuerpo finito $\mathbb{F}_q$, donde $q = p^m$ con p primo, se recomienda el uso de un cuerpo finito binario $\mathbb{F}_{2^m}$. De hecho, se obtiene una seguridad similar tanto con los cuerpos finitos binarios como con los primos, pero la aritmética de un cuerpo finito binario se puede implementar en *hardware* más fácilmente que la de uno primo [BGK+06a].

Por lo que respecta al grado de extensión del cuerpo, m, se deberían elegir valores primos, por razones de seguridad. Esto es debido a que las curvas pertenecientes a cuerpos cuyo grado de extensión es un número compuesto son vulnerables a ataques basados en el descenso de Weil [GS99, GHS02, MT06].

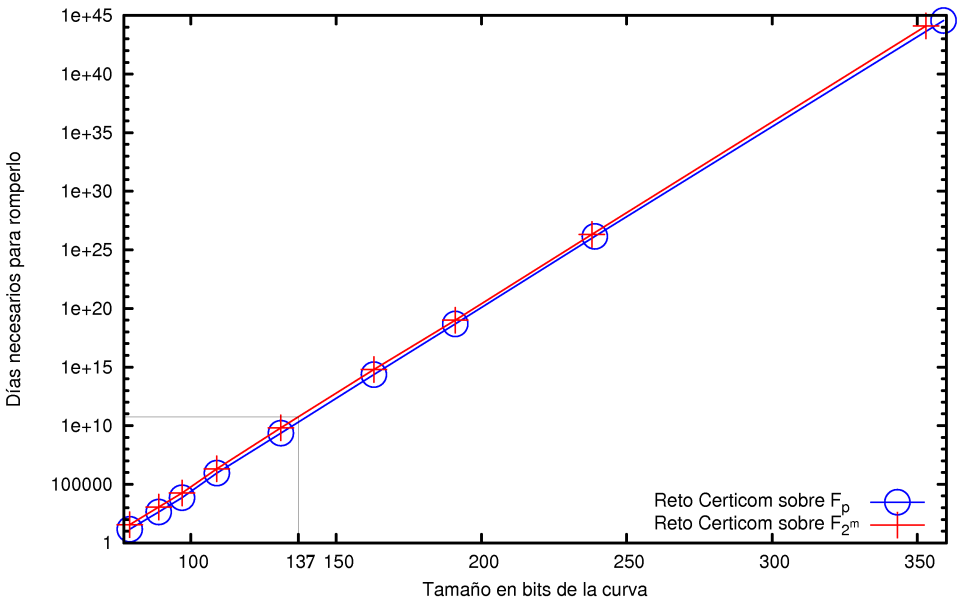


Figura 4.2: Días necesarios para resolver los retos de Certicom.

El último reto ECC[1] resuelto es el ECC2-109, que está definido sobre $\mathbb{F}_{2^{109}}$.

[1]El reto ECC (*Certicom ECC Challenge*) fue introducido por Certicom en 1997 para evaluar la dificultad del problema del logaritmo discreto sobre curvas elípticas.

En la Figura 4.2 se muestra el tiempo necesario estimado (en días) para resolver los retos Certicom del tipo ECCp-m y ECC2-m, donde m es el tamaño del cuerpo en bits (i.e. el tamaño del primo en los ECCp-m y el grado de la extensión en los ECC2-m) en un único procesador de 1GHz[2].

Debido a estos tiempos, proponemos el uso del cuerpo $\mathbb{F}_{2^{137}}$, cuyo logaritmo, tal como se ilustra en el gráfico, requiere más de 10^{10} días para ser resuelto en un único procesador de 1GHz. Esto implica que las claves usadas en el sistema tendrán 137 bits. Éste es, actualmente, un nivel aceptable de seguridad para un entorno RFID, aunque se podrían utilizar cuerpos mayores en sistemas que necesiten seguridad a muy largo plazo o que sean especialmente sensibles.

Finalmente, de acuerdo a los estándares propuestos por Certicom [Cer00], la curva elíptica debe ser escogida de manera que tenga un cardinal de la forma $\#E(\mathbb{F}_{2^{137}}) = h \cdot p$, donde p es un primo grande y el cofactor h es menor o igual a cuatro [MTR+06, MMS+06]. Un punto Q de orden del primo p se ha de escoger como generador.

(b) Longitud del *Reto*: l (número de bits de c)

Para la longitud, l, del *Reto*, hay que tener en cuenta dos consideraciones: En primer lugar, l debe ser suficientemente grande para hacer despreciable la probabilidad de adivinar el *Reto* probando valores al azar. En segundo lugar, está demostrado que el protocolo no sería de conocimiento nulo para valores de l demasiado grandes, porque mediante la interacción, el verificador obtendría soluciones (W, a, c) a la ecuación $W = aQ - cP$ que, de otra manera, podría no ser capaz de calcular (por ejemplo si c se escogiera para ser dependiente de W) [MvOV96].

Estas dos consideraciones llevan a que $2^{-l} \approx 0$ y, $q > 2^{2l}$. Por ello, para $q = 2^{137}$ se recomienda el uso de 40 bits, lo que da una probabilidad insignificante de aproximadamente 10^{-12} de acertar el *Reto*, y permite seguir manteniendo

[2]Es una práctica habitual dar los tiempos de cómputo para una máquina de un núcleo con una velocidad concreta (habitualmente una potencia de 10), esto permite escalar los datos en función de la velocidad (y el número de núcleos) de forma muy sencilla. En esta tesis se toma como referencia un procesador de 1GHz, de forma que, para un ordenador con 4 procesadores de 3GHz cada uno, basta con dividir los tiempos entre 3 y luego (suponiendo que el algoritmo empleado escale bien al ser paralelizado) entre 4.

la propiedad de conocimiento nulo.

(c) Longitud del *Pseudo-Id* id_j^i

Para el *Pseudo-Id*, id_j^i, que calculan las etiquetas, el objetivo es minimizar la probabilidad de que dos etiquetas tengan el mismo *Pseudo-Id*. Por eso, su número de bits debe depender de n, el número de etiquetas en el sistema. La longitud del *Pseudo-Id* debería ser mayor que $2\log_2(n)$ para evitar las colisiones debidas a la paradoja del cumpleaños [MvOV96]; nótese que si la longitud es muy cercana a este valor, entonces la probabilidad de tener dos etiquetas con el mismo *Pseudo-Id* se acerca al 50 %.

Así que, por ejemplo, en un sistema con un millón de etiquetas ($\approx 2^{20}$), la longitud del *Pseudo-Id* debería ser mayor a $2\log_2(1000000) \approx 40$. Escogiendo una longitud de 48 bits, se obtendría una probabilidad de colisión por debajo del 1 %.

De todas formas, incluso en el raro caso de que una operación de lectura retornara un *Pseudo-Id* que fuera el mismo para dos (o más) etiquetas, uno podría simplemente releer las etiquetas. Entonces, los valores deberían ser diferentes.

(d) La función f para obtener el valor z

Tal como se mencionó en la fase de identificación de la etiqueta en la Sección 4.1 (c), una etiqueta calcula su siguiente punto secreto K_{j+1}^i multiplicando el generador Q por un factor $z = f(x(K_j^i))$. Así, obtener el secreto previo a partir del actual resulta ser computacionalmente difícil, ya que implica el cálculo de un logaritmo discreto elíptico. Este hecho da la propiedad de *forward security* a nuestro sistema (como se demostrará en la Sección 4.3.1 f).

Con el fin de obtener una buena eficiencia, la función f debe ser elegida de tal forma que se eviten grandes pesos de Hamming para z, asegurando que la computación de zQ sea más rápida sin comprometer la seguridad. En la Figura 4.3 podemos ver una comparativa de los diferentes niveles de complejidad en función del peso de Hamming.

Los datos mostrados corresponden al número de días estimado que se re-

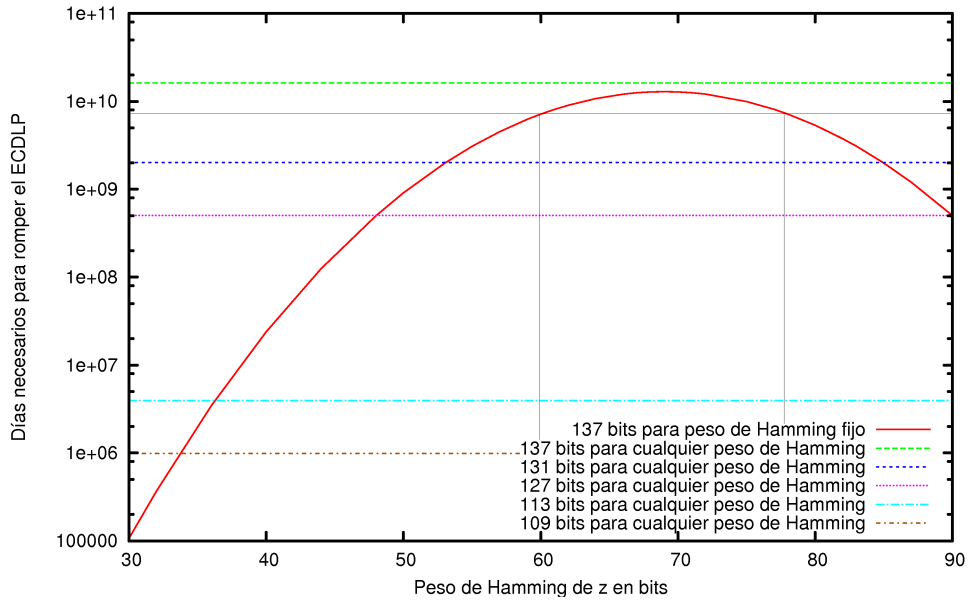


Figura 4.3: Comparativa de complejidad para distintos pesos de Hamming.

querirían para resolver un logaritmo discreto sobre un subgrupo de puntos del tamaño indicado, en un único procesador de 1GHz.

Las cotas horizontales corresponden a logaritmos sin restricción para el peso de Hamming de z. La curva corresponde a la complejidad de resolver el ECDLP, para diversos pesos de Hamming fijos, en un grupo de puntos del orden de unos 137 bits.

Como se puede observar, es recomendable fijar el peso de Hamming a un valor entre 60 y 75 (aproximadamente) para maximizar la seguridad.

Una buena función f para la generación de estos valores es una que asegure la generación de una z con un peso máximo de Hamming en torno a los 66 bits. De las múltiples opciones, nosotros proponemos la siguiente: partir la abscisa de K_j^i en varios fragmentos solapados de 7 bits y usar cada uno de estos fragmentos para elegir qué bits de z activar (es decir, valdrán 1).

Si alguno de los fragmentos corresponde a un bit previamente activado, este bit puede ser desactivado, de esta manera el número de fragmentos será una cota máxima para el peso de Hamming de z. Esto ayudará a decrementar los cálculos realizados por la etiqueta; además de aumentar el espacio de valores

posibles para z, en comparación con funciones que den un peso de Hamming exacto, por lo que la seguridad del sistema también resultará beneficiada.

Para ilustrar el cálculo de z, considérese el siguiente ejemplo con el cuerpo finito pequeño $\mathbb{F}_{2^{16}}$, y un punto de curva elíptica hipotético de coordenadas $(54321, 9876)$, o lo que es lo mismo $(1101010000110001_2, 10011010010100_2)$ en binario. Si se necesita un factor z de peso máximo de Hamming 7, se puede conseguir de la siguiente manera: generando siete fragmentos de la abscisa, de 4 bits cada uno (desplazando dos bits cada vez). Para cada fragmento se calcula su valor binario correspondiente (bit), lo que implicará la activación del valor 2^{bit} en z. Esto se muestra en la Tabla 4.1; en este caso, el valor final de z será $12347 = 8192 + 32 + 16 + 1 + 8 + 4096 + 2$, o 11000000111011_2 en binario.

Tabla 4.1: Generación de z para un ejemplo pequeño.

Fragmento	bit	2^{bit}
1101_2	13	8192
0101_2	5	32
0100_2	4	16
0000_2	0	1
0011_2	3	8
1100_2	12	4096
0001_2	1	2

De forma similar, para el cuerpo $\mathbb{F}_{2^{137}}$, generar el factor z con fragmentos de 7 bits, desplazando dos bits cada vez, produce factores con un peso de Hamming máximo de 66, que, por lo tanto, pueden considerarse seguros.

4.2.2. Factibilidad de la implementación

En este punto, consideraremos la factibilidad de implementar nuestra solución bajo los estándares de implementación actuales para etiquetas RFID.

Hay cinco aspectos principales que se deben considerar:

(a) Complejidad espacial: el número de puertas lógicas.

(b) Complejidad temporal: tiempo requerido para ejecutar el protocolo.

(b.1) Cálculos de la etiqueta: tiempo necesario para realizar los cálculos de la etiqueta.

(b.2) Comunicaciones: tiempo necesario para transferir la información entre etiqueta y lector.

(c) Eficiencia de la BD: la organización de la información de la base de datos de respaldo para que se pueda realizar la fase de verificación en tiempo mínimo.

(d) Escalabilidad: volumen de datos a almacenar en la BD para implementar el sistema propuesto.

El análisis de estos aspectos se hará usando los parámetros propuestos en la Sección 4.2.1 para el protocolo y su extensión para enviar datos capturados por algún sensor.

(a) Complejidad espacial

De acuerdo a los requisitos de las implementaciones de procesadores criptográficos para curvas elípticas que se reportan en la literatura [BGK+06a, BGK+06b, BGK+07], la implementación de las primitivas criptográficas elípticas (suma/doblado de puntos y producto por un escalar), para el cuerpo $\mathbb{F}_{2^{137}}$, se puede conseguir usando menos de 10000 puertas lógicas. Teniendo en cuenta que las etiquetas pasivas actuales pueden tener alrededor de 15000 puertas lógicas [HLS09], se puede asegurar que el protocolo propuesto es factible en términos de complejidad espacial.

(b) Complejidad temporal

Dentro de este apartado se contabilizará el tiempo necesario para realizar los cálculos de la etiqueta y el empleado en transferir la información entre etiquetas y lector. Todo el proceso no debería durar mucho más de un segundo.

(b.1) Cálculos de la etiqueta

Las operaciones más costosas que una etiqueta tendrá que realizar son la verificación de la identidad del lector y la generación del siguiente punto secreto, requiriendo tres multiplicaciones de punto de curva elíptica: La primera es aQ, la segunda es cP, cuyo factor escalar, c, está restringido al rango $1 \leq c < 2^{40}$, y la tercera es zQ, donde z tiene un peso de Hamming máximo de 66.

Considerando las restricciones sobre el espacio de valores de los escalares, el tiempo necesario esperado para las tres multiplicaciones es cercano a $0{,}77 + 0{,}22 + 0{,}54 = 1{,}53$ segundos, usando los procesadores criptográficos para curvas elípticas de [BGK+06a, BGK+06b, BGK+07]. El resto de los cálculos de la etiqueta: comparaciones, *xor*s bit a bit, etc. son trivialmente rápidos.

A pesar de que este tiempo es aceptable, se puede reducir si los puntos Q^{2^e} para $1 \leq e < 137$, para la primera y la tercera multiplicación, y los puntos P^{2^t} para $1 \leq t < 40$, para la segunda multiplicación, se guardan en la memoria de la etiqueta (suponiendo que tenga suficiente espacio). Esto permitiría reducir el tiempo en, aproximadamente, 0,87 s, dejando un tiempo de cómputo reducido a 0,66 segundos.

En caso de no poder almacenar los puntos antes mencionados, el algoritmo de *suma y doblado* para la multiplicación de un punto por un escalar podría ser modificado, para que aQ y zQ se calculasen al mismo tiempo (ya que éstas multiplican escalares por un punto común Q). Esto reduciría el tiempo para estas multiplicaciones en aproximadamente 0,38 s, dejando un tiempo de cómputo de 1,15 segundos, que aún puede considerarse aceptable.

(b.2) Comunicaciones

Para la evaluación de la comunicación hay que considerar las longitudes de los mensajes intercambiados entre lector y etiqueta durante el protocolo (véase la Figura 4.1).

Los primeros tres mensajes corresponden a la fase de autenticación:

El primer mensaje contiene el *Testigo* W. Nótese que para identificar un punto, es suficiente la abscisa y un bit para designar una de las dos posibles ordenadas, por lo que se necesitará enviar $137+1 = 138$ bits. El segundo mensaje contiene el *Reto* c de 40 bits. El tercer mensaje contiene la *Respuesta* a de 137 bits.

El siguiente mensaje, enviado por la etiqueta en la fase de identificación, contiene (id_j^i, u_j^i). Considerando que los datos a cifrar, tomados por el sensor, tienen 64 bits de longitud (el tamaño de un número en coma flotante de doble precisión o de dos números enteros), la longitud del mensaje será de $48 + 64 = 112$ bits. Finalmente, en la fase de verificación el lector envía el mensaje de *Reconocimiento*, ack_j^i, que debe contener no más de 25 bits (nótese que, usar 25 bits o menos para este mensaje, permite garantizar que las funciones FB, MB y LB no se solaparán).

En el estándar *ISO/IEC 18000-3 MODE 2*[3], la tasa de transmisión para los mensajes del lector a la etiqueta es de 423,75 kbit/s, mientras que para los mensajes de la etiqueta al lector es de 105,9375 kbit/s.

Los mensajes enviados por el lector (W, a y ack_j^i) suman $138 + 137 + 25 = 300$ bits, que a una velocidad de 423,75 kbit/s, requieren 0,708 ms. Los enviados por la etiqueta (c y id_j^i, u_j^i) suman $40 + 112 = 152$ bits, que a una velocidad de 105,9375 kbit/s, requieren 1,435 ms. Por tanto, considerando todos los mensajes, el tiempo total invertido en comunicaciones es de 2,143 ms, lo que es insignificante en comparación con el tiempo empleado en cálculos.

En resumen, el tiempo necesario para las interacciones entre lector y etiqueta está determinado principalmente por los cálculos que hace la etiqueta, puesto que las comunicaciones son considerablemente más rápidas. De igual forma, tanto las comunicaciones entre lector y base de datos, como los cálculos que se realizarán para localizar el identificador

[3] *ISO/IEC 18000-3 MODE 2*, cuya última revisión data del 2010, es un estándar para RFID con etiquetas pasivas. Utiliza la frecuencia $13{,}56MHz$ y es de medio alcance (hasta poco más de un metro).

en ella, son insignificantes comparados con el tiempo de cómputo de la etiqueta. Por tanto, el tiempo total del protocolo estará entre los 0,66 s y los 1,53 s, dependiendo de si se implementan o no las mejoras mencionadas.

(c) Eficiencia de la base de datos

La eficiencia se analizará en términos de tiempo de acceso para dar servicio a un sistema RFID como el descrito a lo largo del capítulo.

La base de datos de respaldo tiene que guardar un registro para cada etiqueta, con el siguiente *Pseudo-Id* esperado (48 bits) y el punto secreto actual, que tiene $137 + 1$ bits (abscisa y un bit para determinar la ordenada), junto con algunos datos adicionales, como el tipo de producto, la caducidad, o similares. Estos registros se guardarán en una tabla *hash* indexada por el *Pseudo-Id* para acceder fácilmente. Como es bien sabido, las tablas *hash* [ML75] se usan ampliamente, ya que permiten los accesos y las inserciones en tiempo constante, sin depender del número de elementos almacenados (identificadores de etiquetas en este caso).

Por tanto, la fase de verificación es altamente eficiente, pues es constante para cualquier número de etiquetas.

(d) Escalabilidad

La escalabilidad se analiza en función del incremento de información que requiere la base de datos para implementar el protocolo al aumentar el número de etiquetas.

Como se observó en el Capítulo 2, en aquellos protocolos en los que el número de cambios de los identificadores de las etiquetas no está limitado, con la intención de no comprometer la propiedad de *forward security* del sistema [OSK03, AO05a], la base de datos perdía el control sobre el identificador actual de cada etiqueta (ya que un lector malicioso podría obligar a las etiquetas a refrescar sus valores innecesariamente). Debido a ello, la base de datos tenía que mantener información de largas cadenas de posibles identificadores para cada etiqueta. La ventaja de nuestra

propuesta es precisamente que no necesitamos calcular ni mantener estas cadenas de identificadores, sino solamente el siguiente identificador esperado y el secreto actual.

Considerando sólo el punto y el *Pseudo-Id*, pero ningún otro dato adicional no relacionado con el protocolo en si mismo, la memoria necesaria para nuestra solución, para n etiquetas, es de orden $O(n \log n)$ bits, porque deben guardarse n *Pseudo-Id*s cada uno de ellos con aproximadamente $2 \log_2 n$ bits, y los n puntos secretos (cuya longitud debe asegurar que las funciones FB, MB y LB no se solapen). Por lo tanto, este coste espacial es polinómico en el número de etiquetas.

Como ejemplo, un sistema con un millón de etiquetas RFID necesitará $1000000 \cdot (48 + 137 + 1)$ bits, lo que es sólo alrededor de 22 MiB (perfectamente razonable en sistemas actuales). Además, nuestro sistema no incrementa las necesidades de memoria en función del número de operaciones de lectura. Todo esto implica que nuestro sistema es altamente escalable.

4.3. Análisis de seguridad

En esta sección, se presenta el análisis de seguridad del sistema. Para hacer esto, resultará útil recordar la información pública y secreta implicada en el desarrollo del protocolo.

Información pública permanente:

- $\mathbb{F}_q$: Cuerpo finito.
- $E/\mathbb{F}_q$: Curva elíptica sobre $\mathbb{F}_q$.
- $Q \in E(\mathbb{F}_q)$: Generador de un subgrupo de puntos.
- $P \in \langle Q \rangle$: Clave pública del lector.

Información secreta permanente:

- $s \in [2, d-1]$: Secreto del lector.

Información secreta por sesión de lectura:

- $r \in [2, d-1]$: El *Compromiso* aleatorio, escogido por el lector.
- $K_j^i \in \langle Q \rangle$: Secreto de la etiqueta T_i, en el instante j.
- $LB(y(K_{j-1}^i))$: Necesario para el cálculo del *Pseudo-Id*.
- $z = f(x(K_j^i))$: El factor de cambio de secreto de la etiqueta.
- v_j^i: El valor capturado por el sensor de la etiqueta.

Información pública por sesión de lectura:

- $W = rQ$: El *Testigo* calculado, enviado desde el lector a la etiqueta.
- c: El *Reto* aleatorio, enviado desde la etiqueta al lector.
- $a = r + cs$: La *Respuesta*, enviada desde el lector a la etiqueta.
- $id_j^i = LB(x(K_j^i)) * LB(y(K_{j-1}^i))$: El *Pseudo-Id*, enviado desde la etiqueta al lector.
- $u_j^i = v_j^i \; xor \; FB(y(K_j^i))$: El valor, capturado por el sensor, cifrado, enviado desde la etiqueta al lector.
- $ack_j^i = MB(x(K_j^i)) * MB(y(K_{j+1}^i))$: El *Reconocimiento*, enviado desde el lector a la etiqueta.

Con respecto a la seguridad, el típico escenario que se considera en la literatura [Jue06] para sistemas RFID, asume dos zonas (véase la Figura 4.4): (a) la zona segura, formada por la base de datos de respaldo y el lector RFID, cuyas comunicaciones se suponen cifradas, y (b) la zona insegura, donde están las etiquetas RFID y sus canales de comunicación con el lector. En algunos casos (por ejemplo, para verificar si el sistema tiene *forward security*), se considera que un atacante puede leer las etiquetas físicamente.

En las siguientes subsecciones, se analizarán los tipos básicos de ataque contra este protocolo, en las comunicaciones que se establecen en la zona insegura, que deben ser analizados [RCT06]. Luego, también se evaluará la propiedad de *forward security*.

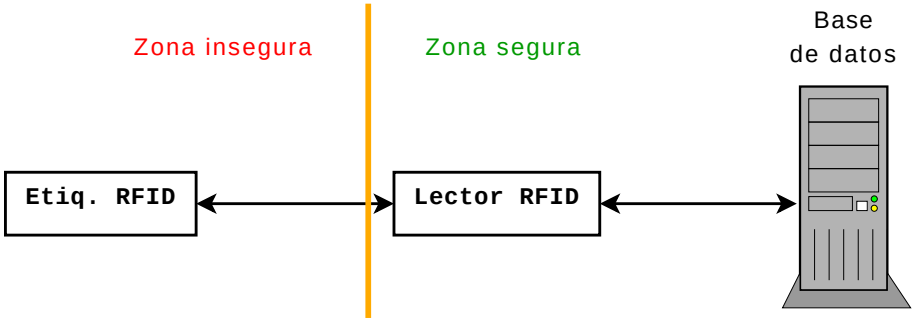


Figura 4.4: Zonas de seguridad de un sistema RFID.

4.3.1. Tipos de ataques

Tal como se ha explicado en el Capítulo 2, los ataques que deben ser considerados pueden clasificarse en diversos tipos básicos. De entre todos ellos, el *sniffing*, el seguimiento de las etiquetas, la suplantación de un lector o de una etiqueta, la denegación de servicio, los ataques de reutilización, y el *forward attack* se analizan en la presente sección. Los *side channel attacks* y los ataques físicos escapan el ámbito de esta tesis, pues dependen de la implementación que se realice (un mismo sistema puede ser vulnerable o no, según la implementación).

(a) *Sniffing*

En un ataque de *sniffing*, el atacante espía las comunicaciones entre un lector y una etiqueta, intentando obtener información útil.

La única información pública es la necesaria para el *setup* del protocolo, la clave pública del lector, la información enviada durante la fase de autenticación del lector (el *Testigo*, el *Reto* y la *Respuesta*), la información enviada en la fase de identificación de la etiqueta (el *Pseudo-Id* y, opcionalmente, los datos del sensor cifrados) y la enviada durante la fase de verificación de la etiqueta (el *Reconocimiento*).

Con estos datos, si un atacante intenta adivinar el secreto de una etiqueta, K_j^i, la única información relacionada con ello es el *Pseudo-Id* (y el *Reconocimiento*). Pero los bits del *Pseudo-Id* ($id_j^i = LB(x(K_j^i)) *$

$LB(y(K_{j-1}^{i})))$ son el resultado de una operación, que no es algebraica, realizada con los últimos bits de la abscisa y la ordenada de dos puntos secretos diferentes, así que no sería computacionalmente factible obtener el secreto a partir del *Pseudo-Id* (un razonamiento similar se puede aplicar al *Reconocimiento*).

Si el atacante quiere el secreto del lector, s, la única información relacionada con ello es la *Respuesta*. Pero, en ese caso, el secreto estará multiplicado por el *Reto* y sumado al *Compromiso* aleatorio (r), que sólo puede ser obtenido calculando el logaritmo discreto elíptico del *Testigo*, lo cual no es computacionalmente factible.

De hecho, formalmente, cualquiera podría realizar una simulación de una autenticación hipotética, eligiendo *Reto* (c) y *Respuesta* (a) de antemano, y calculando el *Testigo* correspondiente como $W = aQ - cP$. De esta manera, se pueden obtener tripletes (W, c, a) válidos sin necesidad de que intervenga el lector. Como se dijo en el Capítulo 3, la posibilidad de tal simulación demuestra que el protocolo es de conocimiento nulo.

Por tanto, un ataque de *sniffing* es inútil debido al uso de un protocolo de conocimiento nulo en combinación con el uso de un *Pseudo-Id*, que no puede ser relacionado con el secreto de la etiqueta, y el cifrado de los datos del sensor.

(b) Seguimiento de las etiquetas

El ataque de seguimiento de las etiquetas permite hacer un seguimiento del comportamiento del propietario de una etiqueta. Por ejemplo, si alguien tiene una etiqueta RFID en su teléfono móvil, el seguimiento de esa etiqueta permite realizar el seguimiento de su comportamiento.

En este caso, la única información que debe ser considerada es el *Pseudo-Id* (y los datos opcionales del sensor), ya que el *Reto* es aleatorio y el resto de datos son enviados por el lector. Un *Pseudo-Id sniffado* en un cierto momento, no puede ser relacionado con la información obtenida antes (ni con la que se vaya a obtener después), porque se genera usando el secreto de una etiqueta, que varía con cada operación de lectura, de una

forma no reversible, asumiendo la dificultad computacional del logaritmo discreto elíptico. El mismo razonamiento se aplica a los datos cifrados del sensor.

(c) Suplantación

En un ataque de suplantación, se intenta suplantar alguna de las entidades del sistema. No tiene sentido plantear la suplantación de la base de datos, debido a los mecanismos de seguridad empleados en las comunicaciones de ésta con el lector. Por lo tanto, se tienen que considerar dos tipos diferentes de este ataque:

(c.1) Suplantación de un lector

Debido al uso del protocolo de Schnorr, cuya seguridad ha sido demostrada, como protocolo de autenticación de conocimiento nulo, la probabilidad de suplantar con éxito a un lector es despreciable.

(c.2) Suplantación de una etiqueta

Un atacante con intención de suplantar a una etiqueta necesita ser capaz de generar el *Pseudo-Id* ($id_j^i = LB(x(K_j^i)) * LB(y(K_{j-1}^i))$), que se calcula a partir de los secretos actual y anterior de la etiqueta a suplantar.

La única información transferida, relativa a los puntos secretos, es el *Pseudo-Id*, el *Reconocimiento* ($ack_j^i = MB(x(K_j^i)) * MB(y(K_{j+1}^i))$) y los datos cifrados ($u_j^i = v_j^i \ xor \ FB(y(K_j^i))$). Cada uno de estos datos utiliza sólo una porción de la abscisa y la ordenada de dos puntos secretos (sólo uno en el caso de los datos cifrados), de manera que no se solapen los bits usados en cada uno de ellos.

Nótese que, las longitudes recomendadas (48, 25 y 64 bits, respectivamente) permiten escoger los bits centrales a usar de forma que se evite dicho solapamiento, pues $48 + 25 + 64 = 137$, que es, precisamente, el tamaño de las coordenadas de los puntos secretos. Es decir, no existe información redundante entre estos tres parámetros, lo que impide posibles operaciones entre ellos, para intentar reconstruir alguno de los fragmentos de las coordenadas.

Por otra parte, la obtención de uno de estos fragmentos tampoco sería suficiente, pues sería imposible saber a qué punto secreto corresponde (alrededor de 2^{73} puntos tendrán iguales los primeros 64 bits).

Así, los puntos secretos no pueden obtenerse a partir de la información pública o las comunicaciones establecidas durante la ejecución del protocolo, por lo que un atacante no puede generar el siguiente *Pseudo-Id*.

Debe tenerse en cuenta que, si un atacante obtiene físicamente el secreto de una etiqueta, pero retorna la etiqueta al sistema sin alterarla, y suplanta la etiqueta antes de que un lector válido lea la etiqueta real, entonces, el secreto de la etiqueta real estará obsoleto, ya que la base de datos cambiará el *Pseudo-Id* esperado. Esto permite una denegación de servicio para esa etiqueta, pero considerando que el atacante necesita acceso físico a la etiqueta para hacer esto, él también podría haber destruido la etiqueta robada directamente.

En ambos casos, para los ataques de suplantación, el adversario debe antes obtener el secreto del elemento a suplantar.

(d) Denegación de servicio

Una denegación de servicio consiste en una incapacitación temporal (o permanente) del sistema o de una parte del mismo.

Como una etiqueta sólo cambia su secreto K_j^i si un lector se ha autenticado exitosamente, no hay peligro de que un adversario realice un ataque de denegación de servicio haciendo múltiples peticiones de lectura, como pasaba en los protocolos de cadenas de *hashes*.

Como se dijo en la fase de verificación de la etiqueta en la Sección 4.1 (d), un mensaje adicional de *Reconocimiento* puede ser necesario en algunos entornos, para evitar el problema de tener la base de datos obsoleta.

Obviamente, como en cualquier sistema inalámbrico, un lector malicioso generando múltiples peticiones de lectura (o simplemente ruido electromagnético) podría provocar que las peticiones provenientes del lector vá-

lido fueran denegadas. En general, los sistemas inalámbricos no se pueden proteger contra esto.

(e) Ataque de reutilización

Un ataque de reutilización consiste en que el atacante reenvíe información que haya capturado previamente, espiando una sesión previa.

Por una parte, es inútil reutilizar el *Testigo*, el *Reto* o la *Respuesta*, ya que es un protocolo de autenticación de conocimiento nulo. Por otra parte, un *Pseudo-Id* no puede ser reutilizado, porque la base de datos esperará el siguiente *Pseudo-Id* de cualquier etiqueta, así que si un atacante reutiliza un *Pseudo-Id*, la base de datos no podrá realizar la verificación de la etiqueta, por lo que el lector no reconocerá ese valor como válido.

(f) *Forward attack*

La propiedad de *forward security* asegura que el revelado de información secreta de la etiqueta no pondrá en peligro la seguridad de la información enviada previamente.

Se considerará que la etiqueta tiene algún tipo de sensor, y que los datos del sensor $v_1^i,\ v_2^i,\ ...,\ v_{j-1}^i$ han sido enviados de forma segura en operaciones de lectura anteriores. También se considerará que el atacante conoce todos los parámetros públicos como antes, y que ha obtenido todos los valores $u_1^i,\ u_2^i,\ ...,\ u_{j-1}^i$, los *Pseudo-Id*s y todos los demás parámetros enviados durante la ejecución del protocolo.

Como hemos visto que un sistema RFID no puede protegerse del todo ante un ataque físico, vamos a suponer que, en el peor de los casos, el adversario es capaz de atacar físicamente la etiqueta T_i y obtener su secreto actual K_j^i.

Entonces, para descifrar los valores u, necesita los secretos previos $K_{j-1}^i,\ K_{j-2}^i,\ ...,\ K_1^i$, y éstos no pueden obtenerse fácilmente porque eso implica resolver un logaritmo discreto de curva elíptica para cada punto secreto que quiera obtener (e invertir la función no biyectiva f).

Como prueba de la propiedad de *forward security*, se estudiará el problema de obtener el secreto previo K_{j-1}^i a partir del actual.

Asumiendo que el ECDLP es difícil, veremos que un atacante que pueda romper el sistema sería capaz de resolver el ECDLP. Como esto último se considera imposible, se deduce que el sistema es seguro por reducción al absurdo.

Considérese un atacante con conocimiento de todos los parámetros públicos y de los parámetros secretos actuales de una etiqueta robada, en particular: el cuerpo $\mathbb{F}_q$, la curva elíptica $E/\mathbb{F}_q$, el generador $Q \in E(\mathbb{F}_q)$, y el secreto de la etiqueta $K_j^i \in E(\mathbb{F}_q)$. Recuérdese que, $K_j^i = zQ = f(x(K_{j-1}^i))Q$. Se puede demostrar que si este atacante tiene acceso a un oráculo que retorne K_{j-1}^i a partir de estos datos, podría calcular fácilmente logaritmos discretos sobre una curva elíptica, por tanto, obtener el secreto previo a partir del actual no es más fácil que calcular logaritmos discretos sobre una curva elíptica

Supongamos que el atacante quiere calcular el factor $z = \log_Q T$. Para obtenerlo, primero utiliza el oráculo, que retorna un punto K, correspondiente a un punto secreto previo hipotético de T, o sea, $T = zQ = f(x(K))Q$. Entonces, el atacante puede calcular fácilmente, a partir de K, $z = f(x(K))$, que es el logaritmo discreto deseado.

Así, esta comunicación tiene la propiedad de *forward security*, i.e. para un atacante que intente descifrar algunos de los valores que pueda haber espiado, es inútil obtener el secreto actual de la etiqueta, porque obtener las claves secretas previas proveería de una solución para el ECDLP, el cual no se puede resolver eficientemente.

A partir del análisis realizado, puede verse que el protocolo propuesto proporciona un canal seguro de comunicación entre lector y etiquetas de un sistema RFID. El protocolo permite defenderse de todos los ataques considerados, mejorando así al resto de protocolos presentados en el Capítulo 2, que son vulnerables a algunos de ellos. Además, su seguridad no repercute negativamente en la escalabilidad del sistema. Esto marca también diferencia con los protocolos existentes, puesto que la mayoría de sistemas que ofrecen la propiedad de *forward security*, se ven obligados a renunciar a la escalabilidad en favor de la seguridad.

Capítulo 5

Algoritmo paralelo para la generación de curvas elípticas

En el Capítulo 4 se ha visto la necesidad de disponer de curvas elípticas, sobre un cuerpo $\mathbb{F}_q$, criptográficamente útiles, i.e. con un cardinal que factorice en un número primo grande y un cofactor pequeño, donde la seguridad vendrá dada, precisamente, por el tamaño de ese primo.

Ante la necesidad de disponer de curvas útiles, obtenerlas mediante la generación de curvas al azar y verificar si su cardinal es aceptable, no es una buena alternativa, ya que esta comprobación es lenta y sólo un pequeño porcentaje de las curvas generadas de esta manera resultarían adecuadas para su uso en criptografía.

Sin embargo, una vez se consigue una curva criptográficamente útil, disponemos de un método para generar nuevas curvas con una seguridad similar basado en el cálculo de las curvas isógenas a una dada. Estas isogenias se organizan en estructuras denominadas volcanes de curvas elípticas, y nos permiten obtener una gran cantidad de curvas elípticas seguras a partir de una dada.

El objetivo de este capítulo es dar un soporte aplicado al protocolo de seguridad definido en el Capítulo 4, proporcionando un mecanismo automático y eficiente para la generación de estos volcanes mediante técnicas de cómputo paralelo. Para ello, se ha desarrollado una aplicación paralela que optimiza el tiempo de generación de dichos volcanes a partir del cálculo simultáneo de las curvas elípticas que lo componen.

A lo largo de este capítulo se expondrá en que consisten estos grafos volcán (Sección 5.1), el método utilizado para paralelizar su generación (Sección 5.2) y los resultados de rendimiento obtenidos en el proceso (Sección 5.3)

Este método dio lugar a dos publicaciones científicas: *Paralelización del cálculo de volcanes para usos criptográficos* [MTR+05] y *Parallel Calculation of Volcanoes for Cryptographic Uses* [MTR+06]. Este último estudia analíticamente la parametrización adecuada de la aplicación.

5.1. Volcanes de curvas elípticas

En esta sección, se expone la estructura del grafo volcán junto con sus principales características y sus propiedades [FM02].

Dado un número primo ℓ, un ℓ-volcán se define como un grafo que contiene un único ciclo, al que llamaremos *cráter*, y tal que de cada uno de los nodos de este ciclo penden $\ell - 1$ árboles ℓ-arios con la misma altura h. El conjunto de las hojas de todos estos árboles es lo que denotamos como el *suelo* del ℓ-volcán, mientras que el resto de nodos de los árboles configuran la *ladera* del ℓ-volcán.

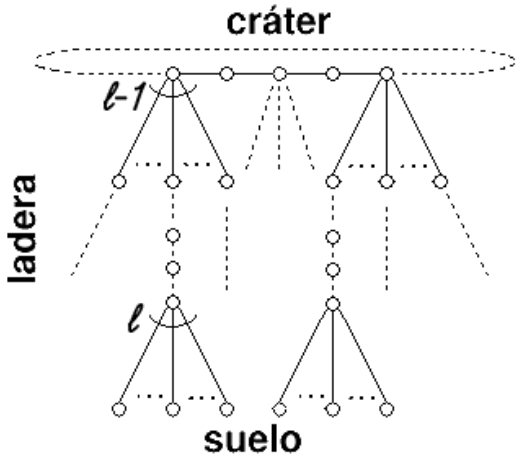


Figura 5.1: Estructura del grafo volcán.

La altura del volcán se define como la altura de los árboles de la ladera, h. Cada nodo del ℓ-volcán, excepto los del suelo, tiene $\ell + 1$ aristas. En los nodos de la ladera, una arista asciende y las otras ℓ descienden. En el cráter,

dos aristas son transversales y $\ell - 1$ son descendientes. La estructura general de un grafo ℓ-volcán está esbozada en la Figura 5.1.

Este grafo volcán puede usarse para representar clases de isomorfía de curvas elípticas sobre un mismo cuerpo finito $\mathbb{F}_q$ y con el mismo cardinal. Cada curva elíptica puede localizarse en un único nodo de su ℓ-volcán. Entonces, los nodos adyacentes de una curva dada corresponden a sus curvas ℓ-isógenas.

Como se ha dicho en el Capítulo 3, la seguridad criptográfica de una curva elíptica está relacionada directamente con su cardinal (número de puntos de la curva). Como las curvas isógenas tienen el mismo cardinal, si una curva dada cumple los requisitos deseados sobre seguridad criptográfica, las curvas obtenidas visitando los nodos de su ℓ-volcán tendrán las mismas propiedades que la original.

La estructura de estos grafos volcán fue analizada por primera vez por Mireille Fouquet [Fou01]. En general, el grafo ℓ-volcán suele ser achatado, puesto que mientras la ladera del ℓ-volcán tiene pocos niveles [MMS+08] (de hecho, la distancia entre un nodo del cráter y un nodo del suelo es, normalmente, inferior a diez), tenemos que, por contra, el cráter tiene un gran número de nodos. En las curvas que se usan en aplicaciones criptográficas, el cráter puede llegar a ser de millones de nodos [MMS+02]. En la Figura 5.2 podemos ver, como ejemplo, un grafo 3-volcán de altura tres, donde se esquematiza la estructura achatada.

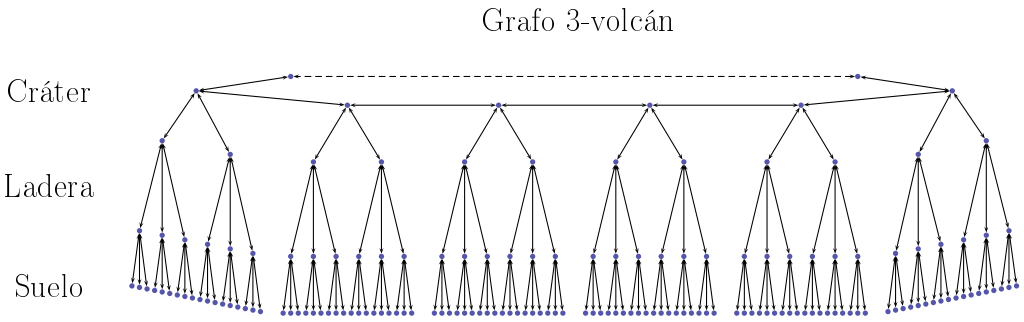


Figura 5.2: Estructura de un 3-volcán.

Para la generación de cada una de las curvas elípticas que componen un ℓ-volcán es necesario visitar las curvas que corresponden a los nodos vecinos de cada curva. Es importante destacar que el cálculo para la obtención de los

nodos adyacentes a partir de un nodo dado es muy costoso, puesto que obtener cada nodo supone el cálculo de raíces cuadradas modulares para $\ell = 2$ o la resolución de un polinomio de grado $\frac{\ell^2-1}{2}$ en el cuerpo finito $\mathbb{F}_q$ para $\ell > 2$. Debido a ello, y al gran número de nodos que componen un ℓ-volcán, el cálculo secuencial de todos los nodos resulta ser computacionalmente costoso [MST+04, MMS+06].

Por ello, proponemos una aproximación que consiste en un algoritmo paralelo que genere todas las curvas elípticas del volcán superponiendo la generación de las diferentes partes del grafo volcán. Evidentemente, dicha superposición en el cálculo debe respetar el orden que marca la generación de cada curva elíptica a partir de uno de sus nodos vecinos. Por lo tanto, este orden, junto con la forma achatada habitual de los volcanes, determinan la estrategia de paralelización, así como el máximo rendimiento que se podrá obtener para la aplicación desarrollada. Esta estrategia se expone a continuación.

5.2. Paralelización del problema

En esta sección se expondrá la estrategia de paralelización seguida para el recorrido y la generación del grafo volcán.

Dada una curva elíptica *buena* (i.e. criptográficamente segura), el proceso que genera los nodos de su grafo volcán distingue tres estadios principales:

1. Encontrar un camino hacia el cráter: el procedimiento asciende, partiendo del nodo en el que se encuentra la curva (del que se desconoce, en general, su nivel o distancia al suelo), hasta alcanzar un nodo del cráter del volcán. Con esto se obtiene también la altura del volcán.

2. Recorrer todos los nodos del cráter.

3. Explorar todos los árboles que penden de los nodos del cráter (la ladera).

En la Figura 5.3, se muestra el orden en que se irían generando los nodos del 2-volcán correspondiente a una curva, etiquetada con el número 1, que está en la ladera del mismo. El número indica en que momento se podría generar cada curva, atendiendo a las dependencias de datos existentes entre ellas.

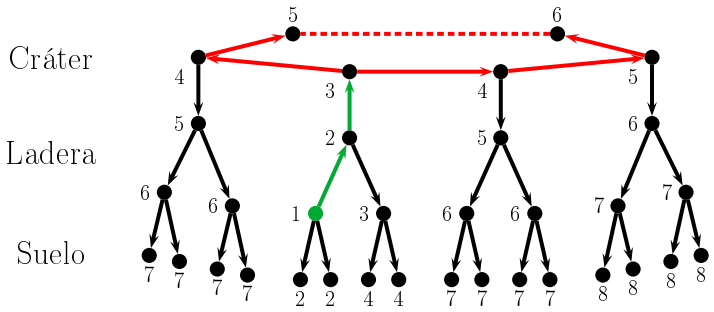


Figura 5.3: Recorrido de un 2-volcán.

Respecto al coste computacional de los pasos de generación del grafo volcán, tenemos que, por una parte, alcanzar el cráter es una tarea computacionalmente sencilla, debido a que los ℓ-volcanes que nos encontramos habitualmente son de escasa altura. Por otra parte, el recorrido del cráter y el posterior descenso de los árboles de su ladera puede llegar a suponer un cálculo del orden de millones de nodos; hecho que conlleva un elevado coste computacional. Debido a ello, la paralelización que se ha llevado a cabo para la aplicación afecta a los recorridos del cráter y la ladera del volcán, pero no al ascenso inicial.

5.2.1. Algoritmo paralelo

El algoritmo paralelo fue desarrollado mediante el paradigma de paso de mensajes, con la librería MPI (*Message Passing Interface*) [MPI94], en su implementación LAM 7.1.

La estrategia de paralelización seguida se basa en el mecanismo *master-worker* con un aprovechamiento de paralelismo mixto: tanto funcional como de datos [SV00].

El mecanismo implementado en el algoritmo consiste en ir obteniendo los nodos del cráter y, de forma concurrente, ir generando el árbol de la ladera que pende de cada uno de estos nodos. Para hacer esto, la aplicación consta de los tres tipos de tareas siguientes:

(a) Una Tarea Maestra (T_M) que coordina el funcionamiento global de la aplicación.

(b) Dos Tareas Cráter (T_C^1 y T_C^2) que obtienen los nodos del cráter. El motivo

por el que se necesitan dos tareas cráter es que, una vez alcanzado el cráter, al ser éste un ciclo, el recorrido se puede efectuar paralelamente en dos direcciones opuestas.

(c) n Tareas Ladera (T_L^1 a T_L^n) que generan los nodos de la ladera del grafo volcán. El número apropiado de tareas ladera T_L^j a generar depende de las características de cada grafo volcán. La elección *a priori* de un número adecuado de tareas es crucial para obtener una paralelización apropiadamente equilibrada, ya que éste es uno de los parámetros que afecta a la granularidad. En la Sección 5.2.2 se proporciona un estudio analítico de esta granularidad.

Estas tareas tienen una estructura de interacción como la mostrada en la Figura 5.4. Como se puede observar, la comunicación de cada una de las tareas se efectuará siempre con la tarea maestra, tal como corresponde al paradigma *master-worker*.

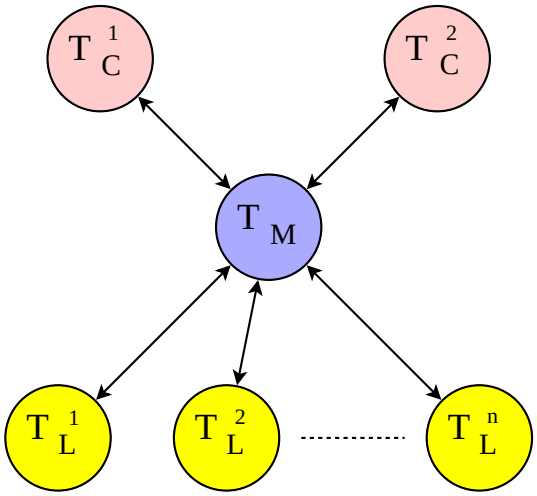


Figura 5.4: Estructura de interacción de las tareas.

A continuación se expone la funcionalidad de cada una de las tareas.

(a) Tarea maestra (T_M)

Es la encargada de hallar los dos primeros nodos del cráter y coordinar todo el recorrido del volcán. Para ello, se encarga de enviar la información a

procesar al resto de tareas y de recomponer todo el grafo volcán a partir de los nodos generados por las mismas. En el Algoritmo 1 se muestra el pseudocódigo que corresponde a la tarea T_M.

Algoritmo 1 : Pseudocódigo de la tarea maestra (T_M).

```
   Entrada : Curva elíptica c_ini
   Salida  : Volcán de curvas
1  (c_d, c_i, h) := Ascender_al_Cráter (c_ini);
2  enviar ((c_i, c_d, h), T_C^1);
3  enviar ((c_d, c_i, h), T_C^2);
4  enviar (h, T_L^*)  (* se envía a todas las tareas ladera *);
5  mientras no esté todo el cráter explorado bucle
6  |  recibir (paquete_nodos, origen);
7  |  si origen == T_C^i entonces
8  |  |  T_L^j := Elegir_Tarea_Ladera_Destino;
9  |  |  enviar (paquete_nodos, T_L^j);
10 |  si no  // origen == T_L^j
11 |  |  Almacenar (paquete_nodos);
12 |  fin si
13 fin bucle
14 enviar (Finalización, T_C^1);
15 enviar (Finalización, T_C^2);
16 mientras queden T_L^j activas bucle
17 |  recibir (paquete_nodos, T_L^j);
18 |  Almacenar (paquete_nodos);
19 fin bucle
20 enviar (Finalización, T_L^*)  (* se envía a todas las tareas ladera *);
```

A partir de un nodo que corresponde a la curva inicial c_{ini}, el algoritmo empieza ejecutando la función `Ascender_al_Cráter`, que va generando los nodos adyacentes a c_{ini}, hasta encontrar el cráter (lo que detecta mediante la determinación de la longitud de un camino de cada nodo hacia el suelo). Esta función retorna el primer nodo del cráter que se encontró, c_d, un nodo adyacente a éste, c_i, que también pertenezca al cráter y la altura, h, del volcán (que corresponde a la longitud de un camino desde el cráter hasta el suelo). Luego, envía la información a las dos tareas cráter T_C^i, para que puedan empezar sus cómputos, y a las tareas ladera T_L^j. Nótese que los dos primeros nodos del

cráter se envían en diferente orden a las tareas cráter, para que cada una genere el cráter en un sentido distinto. A partir de aquí, se entra en un bucle en el que, a medida que se van recibiendo nodos del cráter, estos se van reenviando a las tareas ladera, para que computen el árbol que pende de cada nodo del cráter, y a medida que se van recibiendo nodos de la ladera, estos se van almacenando.

Una vez el recorrido del cráter ha finalizado, la tarea maestra activa la finalización de las tareas T_C^i y continúa hasta recoger todos los nodos calculados de la ladera. Hecho esto, activa la finalización de las tareas T_L^j.

(b) Tarea cráter (T_C^i)

Cada una de las dos tareas cráter aborda el recorrido del cráter en un sentido (al que, por conveniencia, denotamos como derecha o izquierda). Para ello, se parte de los dos nodos del cráter iniciales, recibidos de la tarea maestra, en la dirección que va del primer nodo recibido, hacia el segundo. Además de estos dos nodos iniciales, también se recibe la altura del volcán, h, pues se necesita para el resto de cálculos.

Cabe destacar que el cálculo de cada nodo depende del anterior, por lo tanto la creación de más tareas cráter no aportaría ningún beneficio en el rendimiento de la aplicación. Debido a esto, se contempla únicamente la creación de dos tareas cráter: T_C^1 y T_C^2. En el Algoritmo 2 se muestra el pseudocódigo que corresponde a ambas tareas T_C^i.

El pseudocódigo muestra que cada tarea T_C^i calcula un conjunto de nodos del cráter que se almacena en *paquete_nodos* y luego los envía a la tarea maestra. La longitud de cada paquete de nodos del cráter (P_Lon) es configurable, y es crucial para el funcionamiento de la aplicación ya que define uno de los factores que afecta a la granularidad: la relación entre cómputo y comunicación de las tareas.

En la Sección 5.2.2 se discute el número adecuado de nodos para diferentes valores de ℓ. Tal como se verá en el la Sección 5.3.1, para la implementación del generador de 2-volcanes se ha apostado por calcular dichos nodos en paquetes de doscientos, para dotar a la aplicación de una granularidad adecuada para ser ejecutada en un entorno *cluster*.

La función más importante que está involucrada en la tarea cráter es

Algoritmo 2 : Pseudocódigo de las tareas cráter (T_C^i).

```
1  recibir ((c_0, c_1, h), T_M);
2  c_pre := c_0;
3  c_act := c_1;
4  Añadir (c_act, paquete_nodos);
5  mientras no se reciba la finalización bucle
6  |  para n ∈ {1 .. P_Lon} bucle
7  |  |  c_tmp := Calcular_Nuevo_Nodo (c_pre, c_act);
8  |  |  c_pre := c_act;
9  |  |  c_act := c_tmp;
10 |  |  Añadir (c_act, paquete_nodos);
11 |  fin bucle
12 |  enviar (paquete_nodos, T_M);
13 |  vaciar (paquete_nodos);
14 fin bucle
```

`Calcular_Nuevo_Nodo`. Esta función toma como entrada los nodos del cráter previo y actual, y retorna como salida el siguiente nodo del cráter a ser visitado (siguiendo el recorrido del cráter en la dirección que va del primer nodo al segundo). Nótese que el nodo actual posee ℓ nodos adyacentes que aún no hayan sido visitados, sólo uno de los cuales está en el cráter. Distinguir el nodo del cráter respecto a los otros no es inmediato. Para tal propósito, la función genera ℓ caminos de longitud h, cada uno de los cuales empieza en uno de los nodos adyacentes nuevos, visitando aleatoriamente nodos adyacentes. El camino cuyo nodo inicial pertenece al cráter será aquel que no haya alcanzado el suelo. De este modo, el nodo del cráter podrá ser diferenciado del resto.

Puesto que el cráter es un ciclo, el proceso de recorrido del cráter acabará cuando ambas tareas T_C^1 y T_C^2 se crucen (i.e. cuando hayan calculado algún nodo en común). Esta situación será detectada por la tarea T_M, en cuyo caso activará la finalización de las tareas T_C^i.

(c) Tarea ladera (T_L^j)

Las tareas ladera son las encargadas de generar los árboles de la ladera del volcán.

Inicialmente, se recibe la altura del volcán de la tarea maestra. Luego, se

entra en un bucle en el que, en cada iteración, se recibe de la tarea maestra un paquete de P_Lon nodos del cráter (calculados por una tarea cráter) y se calcula el árbol de nodos que pende de cada nodo del paquete recibido.

La función principal de la tarea ladera es, precisamente, `Construir_Árbol`. Ésta función recibe un nodo y retorna el conjunto de todos los nodos que descienden de éste. Después, estos nodos serán añadidos al paquete de nodos de ladera.

Una vez hayan sido calculados los P_Lon árboles, se mandan todos los nodos a la tarea T_M. En el Algoritmo 3 se muestra el pseudocódigo que corresponde a las tareas T_L^j.

Algoritmo 3 : Pseudocódigo de las tareas ladera (T_L^j).

```
1  recibir (h, T_M);
2  mientras no se reciba la finalización bucle
3  |  recibir (nodos_cráter, T_M);
4  |  para cada nodo ∈ nodos_cráter bucle
5  |  |  árbol := Construir_Árbol (nodo);
6  |  |  Añadir (árbol, paquete_nodos);
7  |  fin bucle
8  |  enviar (paquete_nodos, T_M);
9  |  vaciar (paquete_nodos);
10 fin bucle
```

La tarea T_M detectará la situación en que se han calculado todos los árboles de la ladera del volcán, en cuyo caso activará la finalización de las tareas T_L^j.

Se crearán varias tareas ladera. Nótese que la implementación paralela propuesta no fija el número de tareas ladera T_L^j que deben ser creadas. El número de tareas ladera debe ser establecido por el programador dependiendo de las características del grafo volcán. Una vez más, la adaptación adecuada de este parámetro revertirá en una mejor eficiencia. Por tanto, en la Sección 5.2.2 se llevará a cabo un estudio analítico con el fin de encontrar, para cada volcán y previamente a la ejecución, el número de tareas ladera que proporcionarán una granularidad de computación *versus* comunicación adecuada, que garantice una ejecución con un buen balance de carga.

5.2.2. Análisis de granularidad

Desde un punto de vista práctico, la aplicación paralela no sólo debe tener un diseño funcional correcto de sus tareas, sino que, además, su granularidad debe ser la apropiada para conseguir una ejecución eficiente. Por un lado, aumentar el cómputo de las tareas en sobremedida, limitará el paralelismo, ya que se asignará más trabajo secuencial a una misma tarea, que podría haber sido distribuido en varias. Por otro lado, cada tarea debe comunicarse con la tarea maestra. Dicha comunicación se establecerá a través de la red, cosa que consumirá tiempo. Esto hace que una generación demasiado agresiva de tareas pueda provocar una excesiva sobrecarga de comunicación e impedir un avance eficiente en la ejecución paralela. Por lo tanto, es necesario crear el número de tareas adecuado que consiga un buen compromiso entre cómputo y comunicación.

En el caso de la aplicación desarrollada, la granularidad depende de la longitud de los paquetes de nodos que crean las tareas cráter y del número de tareas ladera T_L^j que serán generadas.

Ambos parámetros deben ser establecidos al inicio de la ejecución. Puesto que ℓ y h son los únicos parámetros del volcán que se conocen *a priori*, el criterio para determinar la longitud del paquete y el número de tareas ladera debe depender sólo de ellos.

Longitud del paquete (P_Lon)

La longitud apropiada para el paquete de nodos del cráter se escogerá atendiendo al número de nodos que colgarán de cada uno de ellos. Como es lógico, cuanto mayor sea dicho número, más pequeños se deberán tomar los paquetes.

El número de nodos que cuelgan de cada nodo del cráter es $\ell^h - 1$ (ver la Tabla 5.1). Se puede observar que las cantidades crecen desmesuradamente al incrementar ℓ o h. Sin embargo, a pesar de que dicha cantidad sea exponencial en h, como las alturas suelen ser muy pequeñas, el factor que tiene más incidencia es ℓ. Por tanto, la longitud del paquete ha sido seleccionada de acuerdo al número máximo esperado de nodos en la ladera del volcán para cada valor

de ℓ. La experimentación previa sugirió que una cantidad adecuada de nodos a generar por cada tarea ladera debería estar entre 10^6 y 10^7. Por ejemplo, para $\ell = 5$ el número máximo esperado de nodos en la ladera del volcán es $390624 \cdot P\_Lon$, así que tomando $P\_Lon = 20$ el número está dentro de los umbrales mencionados.

Tabla 5.1: Número de nodos que cuelgan de cada nodo del cráter y longitud de paquete recomendada.

h	$\ell = 2$	$\ell = 3$	$\ell = 5$	$\ell = 7$	$\ell = 11$
3	7	26	124	342	1330
4	15	80	624	2400	14640
5	31	242	3124	16806	161050
6	63	728	15624	117648	1771560
7	127	2186	78124	823542	19487170
8	255	6560	390624	5764800	214358880
P_Lon	200	200	20	2	2

De todas formas, los detalles de implementación sugieren un valor mínimo y máximo para P_Lon. Por una parte, se considera un mínimo de 2 nodos, ya que un paquete de longitud 1 necesita algo de trabajo extra para discriminar las isogenias del cráter de las que bajan hacia el suelo. Por otra parte, se establece un máximo de 200 nodos, porque si la longitud del paquete fuera mayor podría haber más solapamiento en la intersección de las dos tareas cráter al computar los últimos nodos del cráter. En la última fila de la tabla se muestra el valor adecuado para P_Lon, siguiendo los criterios propuestos.

Número de tareas ladera

El objetivo es crear un número N_T_L de tareas ladera de tal forma que el trabajo a llevar a cabo por cada tarea cráter y cada tarea ladera esté equilibrado. Esto facilitará un mejor balanceo del cómputo.

El análisis de los cálculos que deben ser realizados para la generación de los grafos volcán nos permitió derivar una expresión analítica para calcular de antemano el número apropiado de tareas ladera a crear, de manera que se consiga la eficiencia máxima en la ejecución. La unidad de cómputo de

referencia es el trabajo necesario para la generación de los nodos adyacentes de un nodo dado.

Por una parte, recuérdese que, como se comentó antes, determinar el siguiente nodo del cráter implica visitar ℓ caminos de longitud h. Por tanto, se deben generar $h\ell$ nodos en cada paso. Luego, teniendo en cuenta que hay dos tareas cráter, cada una de ellas visitando $c/2$ nodos del cráter, la cantidad total de trabajo asignado a cada tarea T_C^i es $h\ell\frac{c}{2}$, donde c es el número de nodos que tiene el cráter.

Por otra parte, la cantidad de trabajo realizado por una tarea ladera debe ser evaluada. Primeramente, nótese que visitar los nodos que cuelgan de cada nodo del cráter implicará el cálculo de los nodos adyacentes de ℓ^{h-1} nodos. Así, el trabajo total será $\ell^{h-1}c$. Por tanto, una distribución equilibrada de la computación entre las tareas ladera sugiere que la cantidad de cálculo que llevará a cabo cada tarea T_L^j es $\ell^{h-1}\frac{c}{N\_T_L}$.

Así pues, intentando equilibrar la cantidad de trabajo asignado a las tareas cráter y a las tareas ladera, obtenemos que el número adecuado de tareas ladera debe cumplir la siguiente expresión, de la cual se deduce N_T_L:

$$\begin{aligned} h\ell\frac{c}{2} &= \ell^{h-1}\frac{c}{N\_T_L} \\ N\_T_L &= \frac{2\ell^{h-2}}{h} \end{aligned} \tag{5.1}$$

Como se puede observar, el valor N_T_L depende sólo de ℓ y h, tal como se requería, pues son dos parámetros que se conocen de antemano.

La experimentación confirmó la idoneidad de éste número de tareas ladera obtenido analíticamente, como se verá más adelante.

5.3. Evaluación de rendimiento

De cara a evaluar el rendimiento de la aplicación paralela se ha realizado un proceso de experimentación para analizar la ganancia que se obtiene en el tiempo de ejecución de la aplicación paralela respecto la secuencial (*speedup*), así como para analizar la granularidad idónea de la aplicación.

Dicha experimentación se ha realizado en dos entornos diferentes:

(a) Plataforma de paso de mensajes con la aplicación implementada para el caso específico de $\ell = 2$.

(b) Entorno de simulación usando la aplicación paralela modelada para el cálculo de volcanes con valores de ℓ mayores que 2, con la finalidad de estudiar la escalabilidad de la aproximación propuesta.

En las siguientes subsecciones, se muestran los resultados experimentales que se obtuvieron en ambos casos.

5.3.1. Plataforma de paso de mensajes

El sistema utilizado para las ejecuciones ha sido un *cluster* con 16 nodos, donde cada nodo consta de un procesador Pentium IV a 3 GHz con 2 GB de memoria RAM. La red de interconexión entre nodos es una Gigabit Ethernet.

La aplicación paralela que se ha implementado genera grafos 2-*volcán* de diferentes alturas. De todos los nodos salen tres aristas exceptuando los del suelo de los que sale sólo una. Mediante esta aplicación se ha generado un conjunto de grafos volcán de distintas características cuya descripción se da en la Tabla 5.2 con los siguientes parámetros:

- Altura del volcán (h). Número de aristas desde el cráter hasta el suelo.
- Número de nodos del cráter (c).
- Número total de nodos de la ladera del volcán, incluyendo el suelo (N_Lad).
- Tiempo secuencial (T_Sec), que corresponde al tiempo obtenido, en minutos, al ejecutar la aplicación en un único procesador.

Con el fin de generar el número de tareas ladera adecuado, y evaluar su idoneidad, se han realizado distintas ejecuciones, para cada volcán, variando el número de tareas ladera T_L^j utilizadas. El número total de tareas que ha usado la aplicación en cada caso corresponde al número de tareas ladera más tres,

Tabla 5.2: Descripción de los volcanes usados.

Nombre del volcán	h	c	N_Lad	T_Sec (min)
V_tres	3	27104	189728	7,489
V_cuatro	4	11121	166815	5,209
V_cinco	5	13164	408084	9,516
V_seis	6	5457	343791	5,748
V_siete	7	41052	5213604	60,370
V_ocho	8	788	200940	1,601

puesto que hay que contemplar la existencia de la tarea maestra y dos tareas cráter. Cada ejecución se ha realizado asignando una tarea por procesador y se han evaluado los dos siguientes parámetros:

Speedup. Relación entre el tiempo de ejecución secuencial y el paralelo:

$$Speedup = \frac{\text{tiempo secuencial}}{\text{tiempo paralelo}} \tag{5.2}$$

donde el tiempo secuencial se ha obtenido ejecutando la aplicación en un único procesador y el tiempo paralelo se ha obtenido con una tarea por procesador.

Eficiencia. Relación entre el *speedup* obtenido y el máximo posible:

$$Eficiencia = \frac{speedup \text{ obtenido}}{speedup \text{ máximo}} \tag{5.3}$$

La eficiencia nos da, por lo tanto, un valor entre 0 y 1 que corresponden a la mínima y máxima eficiencia respectivamente. En el caso que nos ocupa, una eficiencia de 1 significaría que el valor de *speedup* obtenido es igual al número de procesadores usados (máximo speedup obtenible), y que, por lo tanto, la aplicación se ha ejecutado con una concurrencia máxima entre sus tareas.

En este caso, se escogió usar una longitud para el paquete del cráter, P_Lon, de doscientos nodos (tal como se mostró en la Tabla 5.1) para proveer a la aplicación de una granularidad adecuada a la ejecución en un *cluster*.

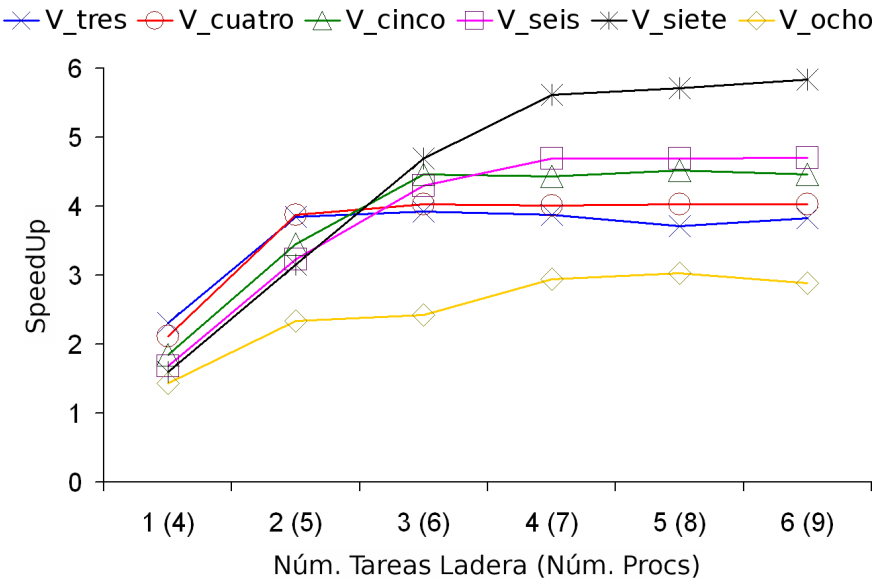


Figura 5.5: *SpeedUp* del generador paralelo de 2-volcanes.

Teniendo en cuenta estos datos se ha realizado la Figura 5.5, que muestra la evolución del *speedup* en función del número de tareas ladera que se crearon, y para cada uno de los volcanes de prueba. En la Figura 5.6 se muestra la eficiencia obtenida.

Lo primero que se puede observar es que, en general, hasta una cierta cota en el número de procesadores, el *speedup* se incrementa, para luego estabilizarse. El punto de estabilización depende de las características del volcán, de manera que, tanto una altura mayor, como un cráter más largo, favorecen la obtención de mejores resultados.

El volcán que ofrece mayor *speedup* es V_siete, puesto que tiene una altura y longitud de cráter considerables; le siguen los volcanes V_seis a V_tres en estricto orden de altura, a pesar de que V_cinco tiene más nodos que V_seis, y V_tres tiene más nodos que V_cuatro. Sin embargo, esto no quiere decir que sea la altura el único factor determinante del *speedup* máximo; como contraejemplo tenemos V_ocho, el cual, a pesar de ser el de mayor altura, ofrece la paralelización más pobre, debido a su cráter inusualmente pequeño.

Esta estabilización del *speedup* se corresponde con bastante exactitud a un pico de eficiencia máxima, tal como podemos ver en la Figura 5.6.

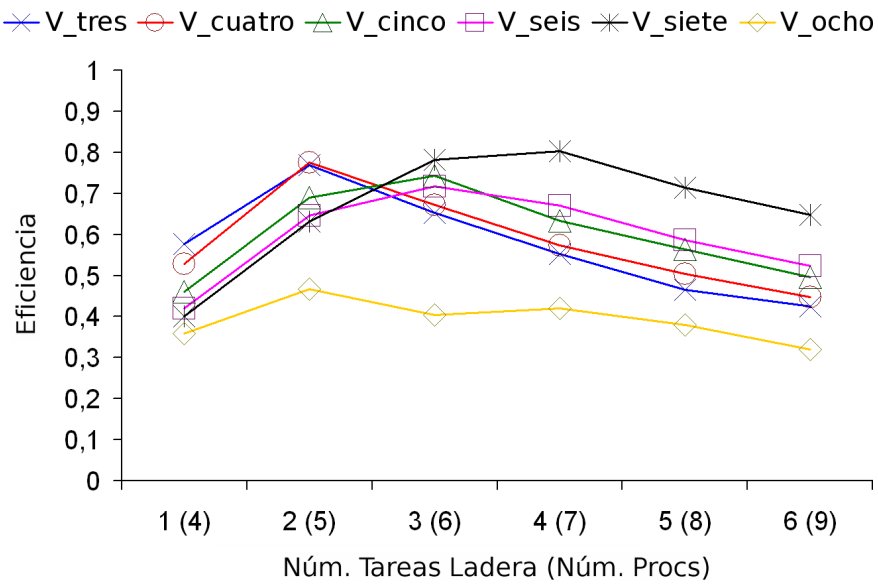


Figura 5.6: Eficiencia del generador paralelo de 2-volcanes.

Otra cosa que podemos observar en el gráfico de eficiencia es que la máxima se alcanza con un número mayor de procesadores cuanto mayor es la altura. V_ocho es una excepción a esta regla debido a su pequeño cráter en comparación con los otros volcanes que hemos calculado.

Así, tenemos que los volcanes V_tres, V_cuatro y V_ocho tienen el pico en 5 procesos, es decir, en dos procesos ladera. En V_cinco y V_seis el pico está en 6 procesos, aunque los picos que nos encontramos con estos volcanes no son tan abruptos como en el caso de V_tres y V_cuatro. Y, finalmente, el pico de eficiencia de V_siete lo encontramos en 7 procesos.

Cabe decir, también, que a medida que la altura del volcán aumenta, la gráfica de eficiencia suaviza su pendiente. Así, mientras vemos el caso de V_tres y V_cuatro donde el pico está perfectamente definido y hay unas pendientes considerables a ambos lados del pico de eficiencia, en los casos de V_seis, V_siete y V_ocho hay unas pendientes más suaves; es más, en el caso de V_ocho se podría hablar de dos picos de eficiencia.

Así, podemos deducir de estas observaciones que para tamaños de cráter similares, se obtiene que el pico de eficiencia máximo se encuentra con un número mayor de procesos a medida que la altura del volcán es más alta. Ade-

más, en este caso, las eficiencias obtenidas con número de procesos similares son más parecidas entre si, pues añadir o suprimir una única tarea tiene menos influencia en la eficiencia. Por contra, cuanto menor sea la altura nos encontraremos el pico con menos procesos y la diferencia de eficiencia con números similares de procesos puede llegar a ser importante, dibujando así las gráficas tan abruptas que encontramos.

A pesar de esto, un cráter demasiado pequeño puede mermar la eficiencia que podemos llegar a obtener, como se ve claramente en el volcán de altura 8, V_ocho. Esto se debe al hecho de que un cráter demasiado pequeño (no mucho mayor que el tamaño del paquete de nodos de las tareas cráter) es más vulnerable al problema de solapamiento comentado anteriormente en la Sección 5.2.2.

Sin embargo, la longitud del cráter no es un factor que podamos conocer *a priori*, por lo que la elección adecuada de la cantidad de tareas ladera, y por consiguiente de procesadores, debería basarse en la altura de cada volcán, como se propuso en el estudio analítico de la sección anterior.

El número de tareas ladera, N_T_L, que proporcionaron los mejores resultados experimentales de *speedup* y eficiencia para cada valor de h ha sido comparado con el valor analítico de N_T_L de la Ecuación 5.1. En la Tabla 5.3 se muestran ambos valores para cada altura.

Tabla 5.3: Valores analíticos y experimentales de N_T_L para $\ell = 2$.

	$h=3$	$h=4$	$h=5$	$h=6$	$h=7$	$h=8$
Analítico	1,33	2	3,2	5,33	9,14	16
Experimental	2	2	3	4	6	5

Se puede observar que el valor analítico es bastante preciso para alturas variando entre 3 y 6. Si la altura es mayor (cosa muy poco probable), el valor analítico de tareas ladera a generar se incrementa significativamente, pero, por contra, experimentalmente, el número de tareas ladera que proporciona la mayor eficiencia aumenta en menor medida. Esto es debido a que hay factores que influencian más a la ejecución, como son las contenciones en las comunicaciones, los cambios de contexto, etc., que no se tienen en cuenta en la expresión

analítica. Debido a ello, como se podría esperar, en estos casos el valor analítico de N_T_L empieza a diferir respecto al valor apropiado de N_T_L obtenido en la ejecución real.

5.3.2. Entorno de simulación

El cálculo de los nodos adyacentes en la generación de los grafos volcán implica el cálculo de raíces cuadradas modulares para $\ell = 2$ o la resolución de un polinomio de grado $\frac{\ell^2-1}{2}$ en el cuerpo finito $\mathbb{F}_q$ para $\ell > 2$. En la práctica, diferentes valores de ℓ necesitan diferentes implementaciones de la aplicación.

Antes de invertir un esfuerzo de tal magnitud para las futuras implementaciones, sería valioso estudiar la escalabilidad del rendimiento de la aplicación cuando el valor de ℓ crece. Por eso, llevamos a cabo una experimentación, basada en una simulación, que se presenta a continuación.

La ejecución de cada aplicación se llevó a cabo con el entorno de simulación *ESPPADA* [GRRL04], que funciona con aplicaciones de paso de mensajes. Las aplicaciones a simular fueron modeladas basándose en el conocimiento de los periodos de los tiempos de computación de las tareas y en los volúmenes de comunicación a transferir entre tareas, lo que se puede obtener de la implementación real de la aplicación de generación de 2-volcanes, adaptando los datos al caso de volcanes con diferentes valores de ℓ. El sistema subyacente fue modelado en *ESPPADA* definiendo un conjunto de nodos de computación homogéneos, con las mismas características que los usados en el *cluster* real de la subsección anterior.

Los volcanes probados tienen un valor ℓ de 3, 5, 7 y 11, y una altura h comprendida entre 3 y 7. Hay que señalar que cuando el valor ℓ crece, la altura de los volcanes tiende a ser más baja. Por eso, para cada valor ℓ, hemos probado solamente las alturas que podían ser razonablemente esperadas. La longitud del paquete a transferir entre las tareas cráter y las tareas ladera se estableció, en cada caso, basándose en el estudio mostrado en la Sección 5.2.2.

Los resultados del *speedup* obtenidos a partir de la simulación se muestran en las Figuras 5.7, 5.8, 5.9 y 5.10. Cada gráfico corresponde a un valor de ℓ diferente, y refleja el comportamiento del *speedup* para cada uno de los valores

de h probados para esos ℓ-volcanes.

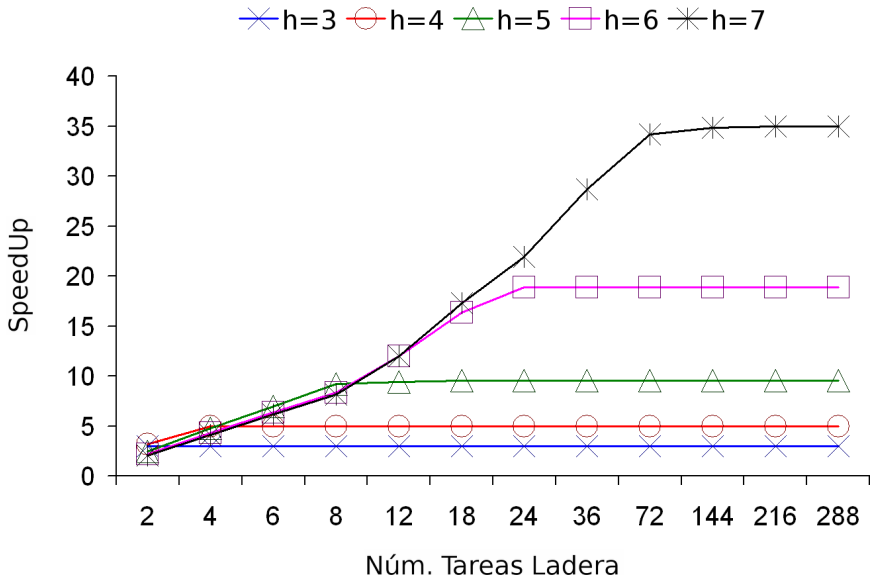


Figura 5.7: *SpeedUp* para $\ell = 3$.

Como se puede ver en la Figura 5.7, cuando ℓ es igual a 3, para las alturas 3, 4, 5, 6 y 7 el *speedup* se estabiliza en 2, 4, 9, 24 y 65 tareas ladera respectivamente, por tanto, el punto de estabilización es mayor cuando el volcán es más alto. Recordemos que también se observó un fenómeno de estabilización similar en el caso de los grafos 2-volcán, así que esto es consistente con los resultados experimentales obtenidos.

Para $\ell = 5$ (Figura 5.8), la altura $h = 3$ estabiliza en 3 tareas ladera, las alturas $h = 4$ y $h = 5$ estabilizan en 12 y 45 tareas ladera respectivamente, y la altura $h = 6$ estabiliza en 200 tareas ladera.

Para $\ell = 7$ (Figura 5.9), la estabilización para la altura $h = 3$ se alcanzó en 4 tareas ladera, para la altura $h = 4$ se alcanzó en 24 tareas ladera, y para la altura $h = 5$ se alcanzó en 135 tareas ladera. Esto no es sorprendente, porque no sólo una gran altura, sino también una ℓ elevada, proveen al volcán de una ladera mayor, lo que proporciona mejores resultados de paralelización.

Para $\ell = 11$ (Figura 5.10), el fenómeno de estabilización también es observable para la altura $h = 3$, que estabiliza en 7 tareas ladera, y para la altura $h = 4$, que estabiliza en 55 tareas ladera.

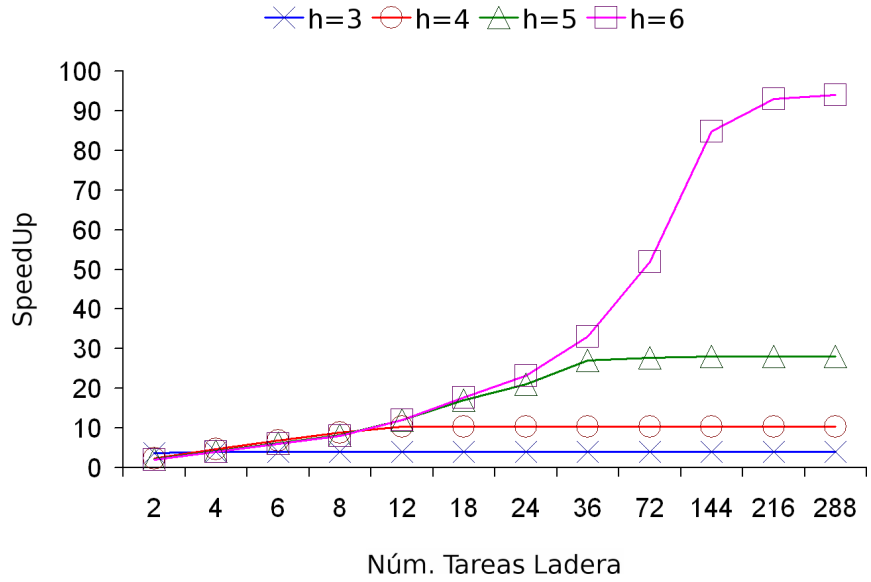


Figura 5.8: *SpeedUp* para $\ell = 5$.

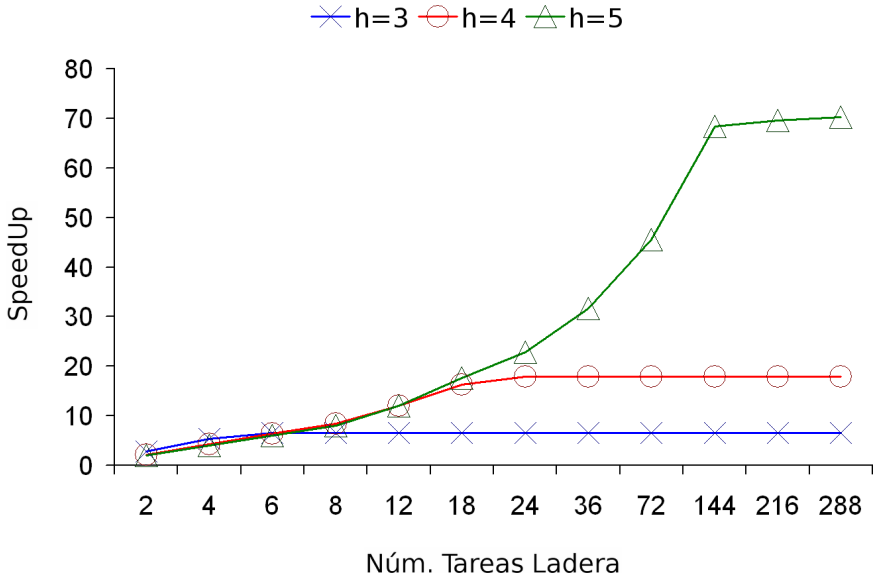


Figura 5.9: *SpeedUp* para $\ell = 7$.

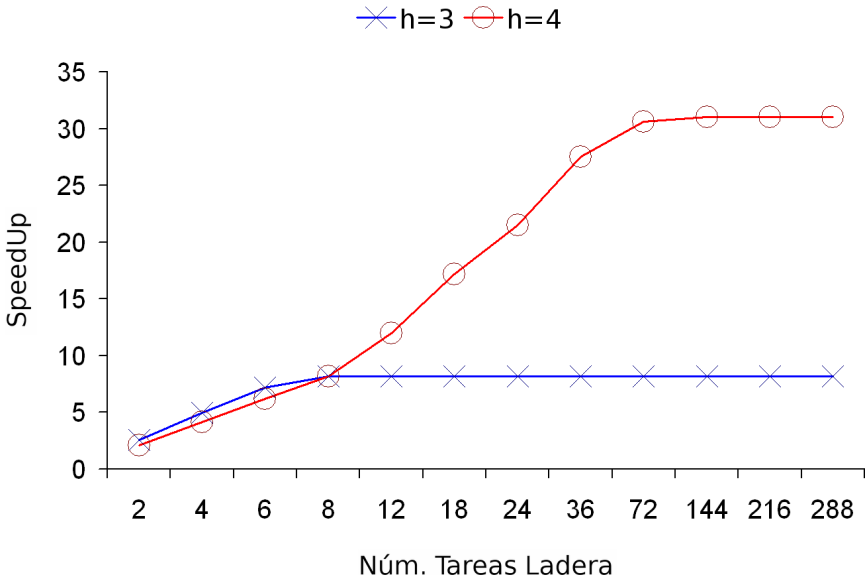


Figura 5.10: *SpeedUp* para $\ell = 11$.

Otro hecho que se puede deducir a partir de los gráficos es que para grandes alturas y valores de ℓ elevados, el *speedup* tiende al número de tareas ladera (número de procesadores menos tres) hasta cierto punto, a partir del cual se estabiliza. Esto es debido al hecho de que, en estos casos, la ladera del volcán es tan grande que las tareas cráter tienen una cantidad despreciable de trabajo a realizar comparado con el de las tareas ladera.

La evolución del *speedup* obtenido en este proceso de simulación es coherente con los resultados para el caso de $\ell = 2$, que se mostraron en la Sección 5.3.1. Los valores analíticos de N_T_L son cercanos a los puntos de estabilización obtenidos para alturas o valores de ℓ pequeños, como se puede ver en la Tabla 5.4. Para alturas o valores de ℓ mayores, los paquetes de nodos de la ladera del volcán se hacen más grandes, así que la sobrecarga del sistema y de las comunicaciones resultan en un *speedup* menor, lo que se traduce en puntos de estabilización más bajos. El valor experimental de N_T_L es siempre menor que el analítico, porque el análisis no tiene en cuenta las sobrecargas mencionadas.

Podemos concluir, a partir de los resultados obtenidos, que el problema de la obtención de curvas elípticas buenas admite una solución paralelizable y que

Tabla 5.4: Valores analíticos y experimentales de N_T_L.

$\ell = 3$		$h = 3$	$h = 4$	$h = 5$	$h = 6$	$h = 7$
	Analítico	2	4,5	10,8	27	69,4
	Experimental	2	4	9	24	65
$\ell = 5$		$h = 3$	$h = 4$	$h = 5$	$h = 6$	
	Analítico	3,33	12,5	50	208,3	
	Experimental	3	12	45	200	
$\ell = 7$		$h = 3$	$h = 4$	$h = 5$		
	Analítico	4,67	24,5	137,2		
	Experimental	4	24	135		
$\ell = 11$		$h = 3$	$h = 4$			
	Analítico	7,33	60,5			
	Experimental	7	55			

dicha paralelización presenta un buen rendimiento si se escogen los parámetros adecuados para las variables implicadas.

Capítulo 6

Protocolo seguro para la capa de comunicación

En el presente capítulo se propone un protocolo de control de acceso al medio (MAC: *Medium Access Control*) para la capa de comunicación de un sistema RFID, que evite el revelado de información a un adversario acerca de la cantidad de etiquetas presentes en el entorno.

Se define un protocolo capaz de garantizar no sólo los requerimientos de seguridad sino también un buen rendimiento y escalabilidad. Para ello, se utiliza un algoritmo probabilístico, capaz de adaptarse a diferentes escenarios.

A lo largo del capítulo se expone detalladamente el protocolo MAC propuesto para la capa de comunicación (Sección 6.1), su correspondiente análisis de seguridad (Sección 6.2) y un extenso análisis de rendimiento vía simulación (Sección 6.3).

Este protocolo ha dado lugar a la siguiente publicación científica: *Securing the Use of RFID-Enabled Banknotes* [MRV10].

6.1. El protocolo MAC propuesto

En este capítulo, consideramos un escenario RFID típico, en el que tenemos diversas etiquetas, un lector autorizado y una base de datos que contiene la información de las etiquetas [Jue06]. También habrá una "etiqueta ruidosa" (*noisy tag*) presente físicamente en el entorno y reconocida por el lector.

El objetivo principal del protocolo presentado en este capítulo es proporcionar un mecanismo MAC (control de acceso al medio), que permita una comunicación segura entre lectores autorizados y etiquetas, de manera que el número de etiquetas que haya presentes en el entorno no sea revelado a posibles atacantes que estén espiando el canal de comunicación. Estos atacantes estarían, supuestamente, interesados en conocer la cantidad de etiquetas del entorno, incluso aunque no pudieran descifrar nada de la información enviada por las etiquetas. En el caso particular de que se añadan etiquetas RFID a los billetes, ésta sería una información muy útil para el adversario.

En este capítulo, consideramos que la capa de aplicación del sistema RFID es segura desde el punto de vista de la protección de datos, i.e. que la información se envía de forma segura y sólo los lectores autorizados son capaces de entenderla [OSK03, AO05a, MVR+07, MVR+09]. En el ejemplo de los billetes, consideramos que el identificador de la etiqueta del billete y cualquier información adicional (que contendrán, probablemente, el número de serie del billete y su denominación) se envían de forma segura al lector RFID. Así pues, un atacante capaz de leer esta información no podrá entenderla. También consideramos una capa física segura [AO05b]. Basándose en estas consideraciones, el protocolo funciona, en la capa de comunicación, con los siguientes dos objetivos:

(a) Permitir a los lectores RFID autorizados identificar la mayor cantidad de etiquetas en un tiempo mínimo, y detectar aquellas que son falsas.

(b) Evitar que los lectores RFID no autorizados adivinen el número de etiquetas con sólo escuchar el canal de comunicación.

La idea principal de nuestro protocolo es que las etiquetas puedan responder de tal manera que el número de respuestas recibidas por el lector durante la operación de lectura sea independiente del número de etiquetas del sistema. A parte de esto, el lector se comunicará con la base de datos de respaldo para verificar la validez de las etiquetas que han sido correctamente identificadas, siguiendo el protocolo definido en la capa de aplicación.

6.1.1. Operación de lectura

El protocolo procede siguiendo varios *time slots* sucesivos (denominados *intervalos* durante el resto del capítulo). En cada uno de estos intervalos las etiquetas pueden responder enviando un mensaje al lector o esperar silenciosamente hasta el siguiente intervalo. Cada etiqueta responde o espera bajo una cierta probabilidad. Las probabilidades de que las etiquetas respondan durante cada intervalo se irán modificando durante la ejecución del protocolo, de forma que las etiquetas tiendan a probabilidades que aseguren la misma distribución de respuestas por intervalo, independientemente del número de etiquetas en el entorno. Esto hace al protocolo escalable cuando el número de etiquetas es grande.

Claramente, si en el entorno hay pocas etiquetas, cada etiqueta tiene que responder con mayor frecuencia, para acelerar la identificación de todas las etiquetas; y si en el entorno hay una gran cantidad de etiquetas, la frecuencia de respuesta de las etiquetas debe ser menor, de forma que se minimicen las colisiones. Para conseguir esto, el lector tiene que colaborar con las etiquetas, informándoles sobre el número de respuestas que recibe.

Por su parte, la etiqueta ruidosa enviará un mensaje al lector en cada intervalo, usando ruido generado pseudoaleatoriamente a partir de un secreto compartido con el lector, de manera que sólo éste pueda restarlo [VV83]. Esto se hace para engañar a los posibles espías, haciendoles creer que en todos los intervalos hay una colisión de respuestas[1]. Este secreto se comparte en el *setup* del sistema.

Al inicio de cada intervalo, el lector envía una señal de intervalo, con información sobre la cantidad de respuestas que se recibieron durante el intervalo anterior. Esto debe hacerse de forma segura, usando el protocolo de la capa de aplicación, de manera que un atacante no pueda obtener esta información.

La cantidad de respuestas recibidas en cada intervalo responde a tres posibles situaciones que conllevan una actuación distinta por parte del protocolo:

[1] Las respuestas de las etiquetas contienen algún tipo de suma de comprobación, de forma que, en caso de que haya dos o más respuestas simultáneas (colisión), el mensaje resultante no tenga una suma de comprobación válida y se detecte la colisión. Cuando responda la etiqueta ruidosa, al enviar un mensaje pseudoaleatorio cuya suma de comprobación no cuadrará, parecerá que ha habido una colisión incluso aunque ninguna otra etiqueta haya respondido.

Cero respuestas: Ninguna etiquetas ha respondido, únicamente la etiqueta ruidosa lo ha hecho. Se recibe una autenticación falsa de la etiqueta ruidosa (simula la autenticación de una etiqueta).

Una única respuesta: Una etiqueta no ruidosa ha respondido. Se ejecuta el protocolo de la capa de aplicación para autenticar a la etiqueta que respondió en ese intervalo previo, mientras el resto de las etiquetas esperan en silencio durante este proceso.

Múltiples respuestas (i.e. colisión): Varias etiquetas (además de la ruidosa) han respondido. Se recibe una autenticación falsa de la etiqueta ruidosa.

El motivo de que se requiera la autenticación falsa de la etiqueta ruidosa en los intervalos de cero o múltiples respuestas es que, en caso contrario, los atacantes podrían discernir entre los intervalos de respuesta única y los otros dos tipos (si los intervalos de una respuesta fueran los únicos en incluir esta parte de autenticación).

Durante cada intervalo, las etiquetas modificarán sus probabilidades de respuesta teniendo en cuenta la cantidad de respuestas del intervalo previo. Obviamente, la señal del primer intervalo no contiene esta información, por lo que las probabilidades son establecidas a un valor por defecto del protocolo.

Basándose en estas tres situaciones, el lector y las etiquetas procederán de acuerdo a los algoritmos expuestos en las siguientes subsecciones. Por simplicidad, primero se explicarán los algoritmos del lector, de la etiqueta y de la etiqueta ruidosa, y posteriormente, se discutirá sobre los valores apropiados para los parámetros que toman parte en estos algoritmos.

6.1.2. Algoritmo del lector

En esta sección, se expone el algoritmo que corresponde a la operación de lectura del lector, cuyo pseudocódigo se muestra en el Algoritmo 4.

El algoritmo recibe como entrada el valor MAX_INTER, que determina el número de intervalos a ejecutar. La cantidad apropiada se discutirá en la Sección 6.1.5.

Como salida se retornará *Conj_IDs*, el conjunto de los identificadores de las etiquetas autenticadas.

Algoritmo 4 : Operación de lectura del lector.

```
   Entrada  : MAX_INTER: Entero positivo
   Salida   : Conj_IDs: Conjunto de identificadores
1  Enviar_Señal_Primer_Intervalo;
2  para Inter ∈ {1 .. MAX_INTER} bucle
3  |  Esperar_Respuestas;
4  |  Restar_Ruido;
5  |  si no hubo respuestas entonces
6  |  |  Enviar_Señal_Intervalo (0);
7  |  |  Obtener_Id_Etiqueta  (* de la etiqueta ruidosa *);
8  |  sino si hubo una respuesta entonces
9  |  |  Enviar_Señal_Intervalo (1);
10 |  |  Id := Obtener_Id_Etiqueta;
11 |  |  Verificar_Validez (Id);
12 |  |  si Id es válido entonces
13 |  |  |  Añadir (Id, Conj_IDs);
14 |  |  si no  // no es válido
15 |  |  |  Notificar que el Id es incorrecto;
16 |  |  fin si
17 |  si no  // hubo colisión
18 |  |  Enviar_Señal_Intervalo (2);
19 |  |  Obtener_Id_Etiqueta  (* de la etiqueta ruidosa *);
20 |  fin si
21 fin bucle
```

El protocolo empieza con el lector enviando la primera señal de intervalo. Después de esto, comienza el bucle principal con el lector esperando respuestas de las etiquetas durante el resto del intervalo. Entonces, se resta el ruido pseudoaleatorio de la etiqueta ruidosa, y el lector envía de forma segura una nueva señal de intervalo, con información sobre la situación de la cantidad de etiquetas que respondieron durante el intervalo anterior (con el valor 0, 1 o 2, correspondiendo a las tres situaciones consideradas de cero respuestas, una respuesta o colisión, respectivamente).

Si hubo una única respuesta, el identificador de la etiqueta es obtenido y verificado usando el protocolo de la capa de aplicación. Si el identificador es

válido (y la etiqueta aún no había sido identificada), éste se añade al conjunto de identificadores, *Conj_IDs*. Si hubo una colisión o un intervalo sin respuestas, se obtiene un identificador espurio de la etiqueta ruidosa, que se descarta automáticamente. Entonces, si hay que ejecutar más intervalos, se inicia una nueva iteración del bucle.

6.1.3. Algoritmo de la etiqueta

El pseudocódigo correspondiente al algoritmo de las etiquetas se muestra en el Algoritmo 5.

Una etiqueta responderá a la señal de intervalo del lector bajo una cierta probabilidad, *Prob_Act*. Así, el factor clave del protocolo es el cambio de esta probabilidad, que se modificará dinámicamente durante la ejecución del protocolo. Esta modificación se lleva a cabo de acuerdo a los siguientes parámetros:

- Una estimación, hecha por la etiqueta, del número de etiquetas en el entorno, *Est_Etiq*. La probabilidad se modifica directamente en función del inverso de este valor. Este parámetro depende, a su vez, de los dos siguientes.

- El número de intervalos procesados, *Inter*. La modificación de la estimación actual, con respecto a la previa, se hace más suave cuando el número de intervalos crece, con el fin de asegurar que la etiqueta estabilizará su probabilidad de responder.

- El número de intervalos consecutivos con cero (*Cero_Consec*) o con múltiples (*Cols_Consec*) respuestas. Un gran número de éstas es un indicio de una probabilidad inadecuada, por lo que la modificación de la estimación debería ser más drástica.

Además de estas variables, el algoritmo almacenará en *Respuestas* la situación (0, 1 o 2) sobre la cantidad de respuestas producidas durante el intervalo anterior.

El algoritmo empieza cuando el lector envía la primera señal de intervalo. Inicialmente, la etiqueta usa una estimación del número de etiquetas de 64

Algoritmo 5 : Operación de lectura de la etiqueta.

```
1  Esperar_Señal_Primer_Intervalo;
2  Inter := 1;
3  Cero_Consec := 0; Cols_Consec := 0;
4  Est_Etiq := 64; Prob_Act := 1/64;
5  repite
6  |  Enviar_Respuesta (Prob_Act);
7  |  Respuestas := Recibir_Señal_Intervalo;
8  |  Inter ++;
9  |  según Respuestas haz
10 |  |  caso 0  // cero respuestas
11 |  |  |  Cero_Consec ++; Cols_Consec := 0;
12 |  |  |  Est_Etiq := Est_Etiq / Factor_Mod (Inter, Cero_Consec);
13 |  |  |  si Prob_Act < 1/Est_Etiq entonces
14 |  |  |  |  Prob_Act := (Prob_Act+1/Est_Etiq)/2;
15 |  |  |  si no
16 |  |  |  |  Prob_Act := Incremento (Prob_Act, 1/Est_Etiq);
17 |  |  |  fin si
18 |  |  |  Esperar mientras se autentica la etiqueta ruidosa;
19 |  |  caso 1  // una respuesta
20 |  |  |  si esta etiqueta fue la que respondió entonces
21 |  |  |  |  Autenticarse al lector;
22 |  |  |  si no
23 |  |  |  |  Cero_Consec := 0; Cols_Consec := 0;
24 |  |  |  |  Est_Etiq −−;
25 |  |  |  |  Prob_Act := Incremento (Prob_Act, 1/Est_Etiq^2);
26 |  |  |  |  Esperar mientras se autentica otra etiqueta;
27 |  |  |  fin si
28 |  |  caso 2  // colisión
29 |  |  |  Cero_Consec := 0; Cols_Consec ++;
30 |  |  |  Est_Etiq := Est_Etiq * Factor_Mod (Inter, Cols_Consec);
31 |  |  |  si Prob_Act > 1/Est_Etiq entonces
32 |  |  |  |  Prob_Act := (Prob_Act+1/Est_Etiq)/2;
33 |  |  |  si no
34 |  |  |  |  Prob_Act := Decremento (Prob_Act, 1/Est_Etiq);
35 |  |  |  fin si
36 |  |  |  Esperar mientras se autentica la etiqueta ruidosa;
37 |  |  fin
38 |  fin
39 hasta que esta etiqueta haya sido identificada ;
```

(esta estimación inicial puede ajustarse dependiendo del número esperado de etiquetas en el entorno, véase la Sección 6.1.5 para mayor detalle), y la probabilidad inicial de que una etiqueta responda durante el intervalo es su inverso (1/64). Entonces, entra en el bucle principal.

Cada iteración del bucle comienza con la etiqueta eligiendo si responder o no durante el intervalo, basándose en su probabilidad actual. A continuación, se espera a la señal que da comienzo al nuevo intervalo y que lleva información sobre las tres posibles situaciones (0, 1 o 2) del intervalo anterior. Dependiendo de la situación, la etiqueta modificará el número estimado de etiquetas y la probabilidad de una de las tres maneras siguientes:

caso 0: $Respuestas = 0$

No hubo ninguna respuesta en el intervalo anterior, así que la etiqueta incrementa el contador de ceros consecutivos y restablece a 0 el contador de colisiones consecutivas. Esta situación significa que la probabilidad actual de responder de la etiqueta es demasiado baja, debido a que la estimación del número de etiquetas es significativamente superior a la real.

Para solucionar esto, la etiqueta realiza las siguientes acciones:

(a) Decrementa el número estimado de etiquetas dividiéndolo por un factor modificador (función *Factor_Mod*) que dependerá del número de intervalos iniciados y del número de ceros consecutivos.

(b) Incrementa la probabilidad de respuesta actual estableciéndola a la media aritmética entre el inverso de la nueva estimación y la probabilidad anterior, salvo en el caso de que este inverso fuera menor (o igual) que la probabilidad anterior, puesto que en esta situación, hacer la media decrementaría la probabilidad (o la dejaría inalterada). En vez de esto, la probabilidad debe ser incrementada usando la función *Incremento*, que depende de la probabilidad actual y del inverso de la estimación del número de etiquetas.

En la Sección 6.1.5 hablaremos sobre la selección de las funciones *Factor_Mod* e *Incremento*.

caso 1: *Respuestas* = 1

Hubo una única respuesta en el intervalo anterior. Si esta etiqueta fue la que respondió durante el intervalo previo, se autentica al lector usando el protocolo de la capa de aplicación.

En caso contrario, el contador de ceros consecutivos y el contador de colisiones consecutivas son restablecidos a 0, y el número estimado de etiquetas se decrementa en uno (puesto que la etiqueta que respondió no lo hará más en la sesión actual). Entonces, la etiqueta incrementa su probabilidad actual en función del inverso de la estimación al cuadrado, de forma que la probabilidad global de que el lector obtenga una sola respuesta no se vea modificada (como si la probabilidad de la etiqueta que respondió se repartiera entre el resto). Finalmente, se espera a que la etiqueta que respondió se autentique.

caso 2: *Respuestas* = 2

Hubo una colisión de respuestas en el intervalo anterior, así que la etiqueta restablece a 0 el contador de ceros consecutivos e incrementa el contador de colisiones consecutivas. Esta situación significa que la probabilidad actual de responder de la etiqueta es demasiado alta, debido a que la estimación del número de etiquetas es significativamente inferior a la real.

Para solucionar esto, la etiqueta realiza las siguientes acciones:

(a) Incrementa el número estimado de etiquetas multiplicándolo por un factor modificador que dependerá del número de intervalos iniciados y del número de colisiones consecutivas.

(b) Decrementa la probabilidad de respuesta actual estableciéndola a la media aritmética entre el inverso de la nueva estimación y la probabilidad anterior, salvo en el caso de que este inverso fuera mayor (o igual) que la probabilidad anterior, puesto que en esta situación, hacer la media incrementaría la probabilidad (o la dejaría inalterada). En vez de esto, la probabilidad debe ser decrementada usando la función *Decremento*, que depende de la probabilidad actual y del inverso de la estimación del número de etiquetas.

En la Sección 6.1.5 hablaremos sobre la selección de la función *Decremento*.

Una vez estudiados los tres casos posibles, nótese que, en caso que *Respuestas* sea 0 o 2, la etiqueta esperará durante la autenticación falsa de la etiqueta ruidosa. Finalmente, si la etiqueta aún no ha sido identificada, comienza una nueva iteración. En caso contrario, se desactiva para el resto de la sesión.

Aunque no se mencione, la estimación del número de etiquetas, *Est_Etiq*, nunca se decrementa por debajo de 1, ya que cada etiqueta está (bastante) convencida de su propia existencia. Nótese que un valor inferior a 1, daría problemas con diversas partes del algoritmo que necesitan que el inverso de *Est_Etiq* pertenezca a $[0, 1]$.

Además, el algoritmo admite otra mejora que no fue expuesta en el pseudocódigo para favorecer la comprensión. Ésta permitirá alcanzar la probabilidad apropiada incluso más rápido cuando el número de etiquetas es demasiado grande o demasiado pequeño. Consiste en modificar la probabilidad de manera diferente cuando el protocolo está en sus fases iniciales. Una etiqueta considera que el protocolo está en sus fases iniciales hasta que ha habido un intervalo de cada uno de los tres tipos (con cero, una y múltiples respuestas). En estas fases iniciales, la probabilidad se establecerá al inverso de *Est_Etiq*.

Vale la pena remarcar que el protocolo descrito podría ser implementado fácilmente en la circuitería de una etiqueta, porque las operaciones involucradas en el algoritmo son muy sencillas: básicamente operaciones aritméticas elementales y comparaciones.

6.1.4. Algoritmo de la etiqueta ruidosa

El pseudocódigo correspondiente al comportamiento de la etiqueta ruidosa se muestra en el Algoritmo 6.

A partir de la recepción de la primera señal de intervalo, la etiqueta ruidosa entra en un bucle en el que envía ruido para ocultar posibles respuestas del resto de etiquetas. Luego, como el resto de etiquetas, espera la señal de intervalo con información sobre la cantidad de respuestas que recibió el lector. Si no hubo respuesta única (i.e. no hubo ninguna o hubo colisión), entonces la etiqueta ruidosa hará su autenticación falsa, con el fin de despistar a un posible espía del

Algoritmo 6 : Operación de lectura de la etiqueta ruidosa.

```
1  Esperar_Señal_Primer_Intervalo;
2  para siempre bucle
3  |  Enviar_Ruido;
4  |  Respuestas := Recibir_Señal_Intervalo;
5  |  si Respuestas ≠ 1 entonces
6  |  |  Autenticarse al lector;
7  |  si no
8  |  |  Esperar mientras se autentica una etiqueta;
9  |  fin si
10 fin bucle
```

canal. En caso contrario esperará a que se autentique la etiqueta que respondió. Después de esto empezará una nueva iteración.

6.1.5. Selección de parámetros

Algunos de los parámetros y funciones que intervienen en el protocolo definido se deben ajustar para un funcionamiento apropiado del sistema: la estimación inicial del número de etiquetas, la función para la modificación del número estimado de etiquetas en el sistema, las funciones para la modificación de la probabilidad de respuesta de la etiqueta y el número de intervalos que el protocolo debería ejecutar.

Estimación inicial del número de etiquetas

Para la inicialización de *Est_Etiq*, sugerimos un valor de 64, pues éste funciona razonablemente bien para entornos con un número de etiquetas entre 2 y 2000 (aunque se pueden cubrir valores incluso más grandes), que son los escenarios de ejemplo que hemos considerado. Nótese que, si al principio hubiera tres intervalos consecutivos sin respuestas, la evolución de la estimación sería, aproximadamente, $64, 32, 9, 2$, y si hubiera tres colisiones consecutivas, la evolución sería $64, 128, 450, 2100$, por lo que las etiquetas pueden adaptar su estimación al valor correcto en breve tiempo, tanto si en el entorno hay un gran número de etiquetas como si hay muy pocas. Esta estimación se podría

adaptar al valor apropiado para otros escenarios, dependiendo del rango de posibles cantidades de etiquetas.

Modificación del número estimado de etiquetas

El objetivo de la función *Factor_Mod* es ajustar dinámicamente el valor de *Est_Etiq*, para acercarse al número correcto de etiquetas en el entorno. El factor retornado por la función será utilizado para decrementar el valor de *Est_Etiq*, cuando la estimación es demasiado alta (en caso que *Respuestas* = 0), dividiendo por él, o para incrementarlo, cuando la estimación es demasiado baja (en caso que *Respuestas* = 2), multiplicando por él. Por lo tanto, la función tiene que retornar un valor mayor que 1.

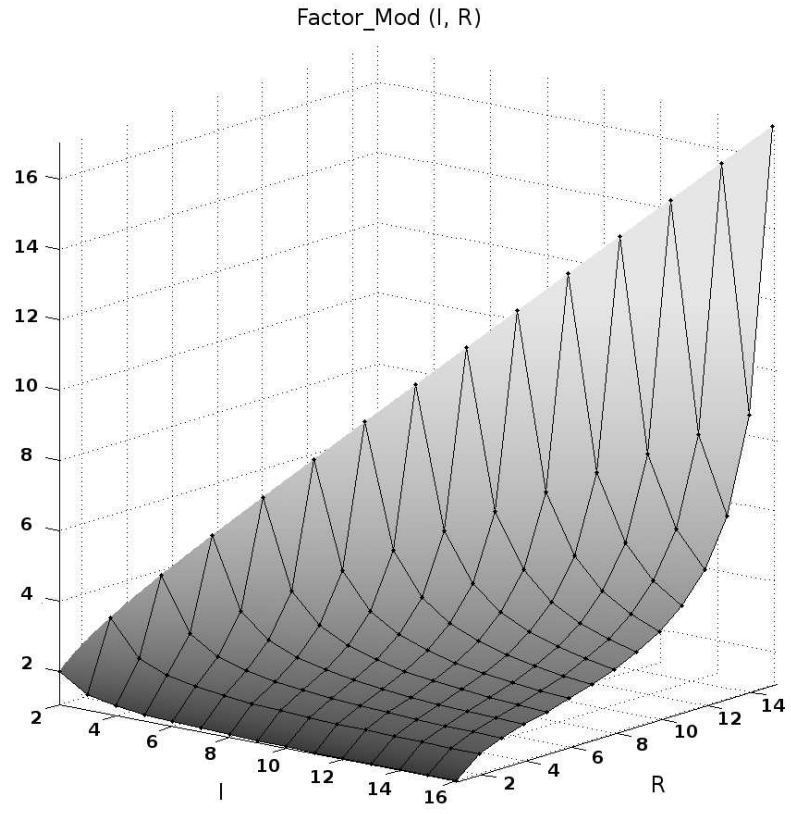


Figura 6.1: Función $Factor\_Mod(I, R)$.

La función *Factor_Mod* recibe dos argumentos: I es el número de in-

tervalos iniciados, y R es el contador de ceros ($Cero\_Consec$) o colisiones ($Cols\_Consec$) consecutivas, dependiendo de la situación. De acuerdo con esto, se propone la función que corresponde a la siguiente expresión:

$$Factor\_Mod(I,R) = 1 + \frac{R}{I-R} + \frac{R-1}{R} \tag{6.1}$$

La Figura 6.1 muestra la representación gráfica de la función $Factor\_Mod$ para los primeros valores de I y R. Como se puede observar, para un mismo valor de I, el resultado de la función crece a medida que se incrementa R (i.e. cuantos más ceros/colisiones consecutivas haya habido, más alto será el factor por el que dividir/multiplicar Est_Etiq). Por otra parte, para un mismo valor de R, el resultado de la función decrece a medida que se incrementa I (así, Est_Etiq cambiará de manera suave). Esto cumplirá con el objetivo de decrementar el valor de Est_Etiq en caso que no haya respuestas e incrementarlo en caso de colisión.

Tabla 6.1: Diversos valores de $Factor\_Mod(I,R)$.

I \ R	1	2	3	4	5	6	7	8	9	10	11	12
2	2,00											
3	1,50	3,50										
4	1,33	2,50	4,67									
5	1,25	2,17	3,17	5,75								
6	1,20	2,00	2,67	3,75	6,80							
7	1,17	1,90	2,42	3,08	4,30	7,83						
8	1,14	1,83	2,27	2,75	3,47	4,83	8,86					
9	1,13	1,79	2,17	2,55	3,05	3,83	5,36	9,88				
10	1,11	1,75	2,10	2,42	2,80	3,33	4,19	5,88	10,9			
11	1,10	1,72	2,04	2,32	2,63	3,03	3,61	4,54	6,39	11,9		
12	1,09	1,70	2,00	2,25	2,51	2,83	3,26	3,88	4,89	6,90	12,9	
13	1,08	1,68	1,97	2,19	2,42	2,69	3,02	3,47	4,14	5,23	7,41	13,9

La Tabla 6.1 muestra, como ejemplo, el valor específico de $Factor\_Mod(I,R)$ para los primeros doce valores de I y R. Los valores mostrados en la tabla corresponden a aquellos que cumplen la condición $I > R$, porque el número de ceros o colisiones consecutivas es siempre menor que el número de intervalos iniciados.

El valor de $Factor\_Mod(I+1, R+1)$ siempre es mayor que el de $Factor\_Mod(I,R)$, pues, por ejemplo, tras i intervalos, con los últimos r

siendo colisiones, una nueva colisión causaría que ambos argumentos se incrementaran en uno, y queremos factores más grandes en estos casos.

Cuando los valores de I y R son muy cercanos, la función retorna valores más grandes, porque esto significa que la estimación está lejos del número real de etiquetas y necesita ser modificada más drásticamente.

También se puede observar en la tabla, que $Factor\_Mod(I, R)$ siempre retorna un valor mayor que 1 (tal como se requería), y que tiende a 1 cuando I crece y R decrece, pues en tal caso la estimación es buena y no necesita ser modificada.

Modificación de la probabilidad de respuesta

Las funciones *Incremento* y *Decremento* han sido diseñadas para modificar la probabilidad actual de respuesta, $Prob\_Act$. Como su salida se guardará como la nueva probabilidad, el resultado debe pertenecer a $[0, 1]$.

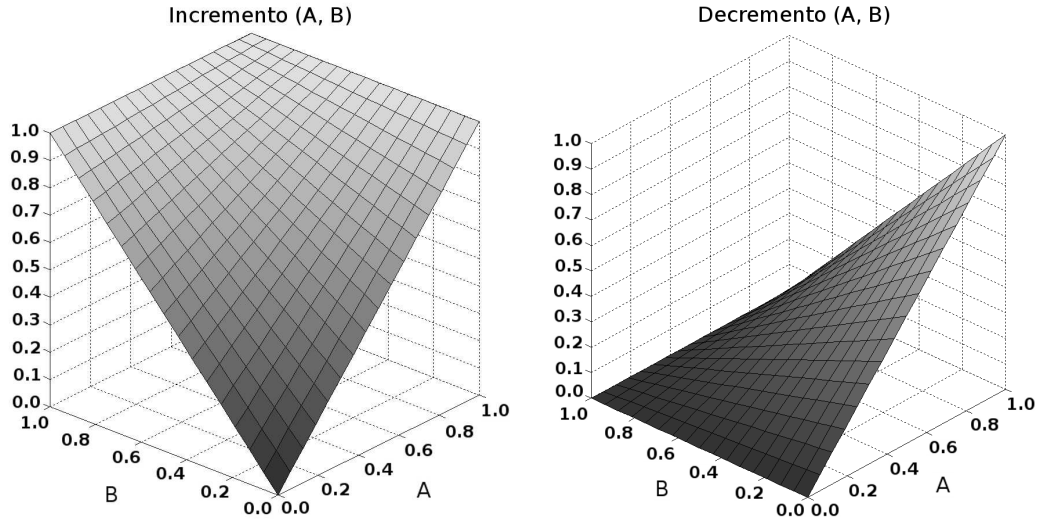


Figura 6.2: Funciones $Incremento(A, B)$ y $Decremento(A, B)$.

Ambas funciones reciben dos parámetros A y B, cuyos valores pertenecen a $[0, 1]$, y corresponden a la probabilidad actual y a la cantidad de modificación, respectivamente. Una cantidad de modificación de 0 corresponde a una modificación nula, por lo que se retornará la probabilidad actual inalterada. Si

es 1, en cambio, la modificación será la máxima posible, retornando, por tanto, una probabilidad de 0 en el caso de la función *Decremento* y una probabilidad de 1 en el caso de *Incremento*. De acuerdo a estos parámetros, las expresiones que proponemos para calcular ambas funciones son las siguientes:

$$\begin{aligned} Incremento(A, B) &= A + B \cdot (1 - A) \\ Decremento(A, B) &= A - B \cdot A \end{aligned} \tag{6.2}$$

Como se puede observar en la Figura 6.2, *Incremento* se comporta como una suma saturada $A + B$; retorna un valor mayor o igual que A, y el incremento depende de B. Similarmente, *Decremento* corresponde a una resta saturada $A - B$; retorna un valor menor o igual que A, y el decremento depende de B. En ambos casos, si B es mayor, la modificación es más importante.

Tabla 6.2: Diversos valores de $Incremento(A, B)$ y $Decremento(A, B)$.

Incremento

A \ B	0	0,2	0,4	0,6	0,8	1
0	0	0,2	0,4	0,6	0,8	1
0,2	0,2	0,36	0,52	0,68	0,84	1
0,4	0,4	0,52	0,64	0,76	0,88	1
0,6	0,6	0,68	0,76	0,84	0,92	1
0,8	0,8	0,84	0,88	0,92	0,96	1
1	1	1	1	1	1	1

Decremento

A \ B	0	0,2	0,4	0,6	0,8	1
0	0	0	0	0	0	0
0,2	0,2	0,16	0,12	0,08	0,04	0
0,4	0,4	0,32	0,24	0,16	0,08	0
0,6	0,6	0,48	0,36	0,24	0,12	0
0,8	0,8	0,64	0,48	0,32	0,16	0
1	1	0,8	0,6	0,4	0,2	0

La Tabla 6.2 muestra, como ejemplo, el valor concreto de $Incremento(A, B)$ y $Decremento(A, B)$ para diversos valores de A y B. En ella se puede observar, numéricamente, el comportamiento descrito previamente.

En el algoritmo, la cantidad de modificación irá en función del inverso de la estimación del número de etiquetas.

Número de intervalos a ejecutar

Para el número de intervalos a ejecutar por el protocolo, *MAX_INTER*, el criterio recomendado es ejecutar siempre un número fijo de intervalos, con el fin de no proporcionar ninguna pista sobre el número de etiquetas presentes. Así pues, decidir el número de intervalos debe hacerse de antemano.

Como se verá en la Sección 6.3.1, el número de intervalos con una única respuesta tiende a un tercio del número total de intervalos. Debido a ello, *MAX_ INTER* debería ser ligeramente mayor que el triple del número máximo esperado de etiquetas. Esto permitirá identificar todas las etiquetas.

6.1.6. Ejemplo de ejecución del protocolo

Para ilustrar el funcionamiento del protocolo, consideramos como ejemplo de ejecución, un pequeño entorno con cinco etiquetas, del que mostramos los intervalos necesarios para identificarlas todas.

Tabla 6.3: Ejemplo de ejecución del protocolo con 5 etiquetas.

I	0_C	C_C	F_{Mod}	Est_E	P_{act}	Et. 1	Et. 2	Et. 3	Et. 4	Et. 5
1	0	0		64	0,016	0,029	0,324	0,869	0,406	0,955
2	1	0	/2	32	0,031	0,711	0,255	0,283	0,828	0,859
3	2	0	/3,5	9,14	0,109	0,285	0,938	0,910	0,480	0,369
4	3	0	/4,67	1,96	0,510	0,890	**0,156**	0,698	0,726	**0,267**
5	0	1	∗1,25	2,45	0,408	**0,129**	**0,210**	0,809	**0,351**	**0,379**
6	0	2	∗2	4,90	0,204	0,527	0,554	**0,096**	0,638	0,665
7	0	0	−−	3,90	0,257	0,333	0,361		0,903	0,930
8	1	0	/1,14	3,41	0,275	0,472	**0,142**		0,280	0,336
9	0	0	−−	2,41	0,400	**0,368**			**0,395**	0,937
10	0	1	∗1,11	2,68	0,386	0,913			0,455	0,483
11	1	0	/1,1	2,44	0,399	0,431			0,487	**0,028**
12	0	0	−−	1,44	0,691	**0,491**			**0,168**	
13	0	1	∗1,08	1,55	0,667	**0,301**			0,866	
14	0	0	−−	1	1				**0,777**	

La Tabla 6.3 ilustra el valor de las variables que intervienen en el protocolo durante los 14 intervalos de ejecución que han sido necesarios. Para cada intervalo, se muestra la siguiente información:

I: Número de intervalo (*Inter*).

0_C: Número de intervalos consecutivos con cero respuestas (*Cero_ Consec*).

C_C: Número de intervalos consecutivos con colisión (*Cols_ Consec*).

F_{Mod}: Factor de modificación (*Factor_Mod*).

Est_E: Número estimado de etiquetas (*Est_Etiq*).

P_{act}: Probabilidad actual (*Prob_Act*).

Et. 1 a **5:** Valores aleatorios generados por cada una de las etiquetas.

Las seis primeras columnas son comunes a todas las etiquetas, ya que estamos suponiendo que todas ellas estaban presentes en la vecindad del lector cuando el protocolo empezó. Si una etiqueta se incorporase más tarde, tendría valores diferentes; obviamente el valor de *Inter* sería menor y, por esa misma razón, sus factores de modificación serían mayores, asegurando que se alcanzase una probabilidad adecuada más rápido.

Las cinco columnas restantes corresponden a los valores aleatorios (i.e. los valores generados por *Enviar_Respuesta (Prob_Act)* al decidir si responder o no) de cada una de las etiquetas. Básicamente, la función *Enviar_Respuesta (Prob_Act)* genera un valor aleatorio entre 0 y 1, y envía el identificador de la etiqueta (cifrado por el protocolo de la capa de aplicación) si ese valor es menor que *Prob_Act*. Estos valores aleatorios fueron generados por ordenador y truncados a tres dígitos para este ejemplo.

Los valores en negrita en estas columnas corresponden a aquellos que son inferiores a *Prob_Act*, lo que significa que la etiqueta respondió durante ese intervalo. Una fila con más de un valor en negrita corresponde a un intervalo con una colisión. Filas sin ninguno de estos valores corresponden a intervalos con cero respuestas.

El factor de modificación no se usa en el primer intervalo, ni en los siguientes a uno de respuesta única (indicado con dos guiones), en los que la estimación simplemente se decrementará en 1.

Hasta el sexto intervalo, el protocolo está en sus fases iniciales (aún no había habido un intervalo de cada uno de los tres tipos), por tanto, en esos intervalos la probabilidad *Prob_Act* se establece directamente al inverso de *Est_Etiq*. A partir de entonces, se sigue el algoritmo normal, de manera que las modificaciones de la probabilidad se hacen más suaves (aunque siempre aproximándose al inverso de la estimación).

Como ejemplo, la cuarta fila corresponde a un intervalo posterior a tres intervalos sin respuesta, por lo que se dividirá la estimación por el valor $Factor\_Mod(I,R) = Factor\_Mod(4,3) = 1 + \frac{3}{4-3} + \frac{3-1}{3} = 4 + \frac{2}{3} = 4{,}67$, pasando de la estimación del intervalo anterior, $Est\_Etiq = 9{,}14$, a la nueva, $Est\_Etiq = 9{,}14/4{,}67 = 1{,}96$. Al estar en las fases iniciales, $Prob\_Act$ se establece al inverso de la estimación: $1/1{,}96 = 0{,}51$. Finalmente, las etiquetas generan sus valores aleatorios y dos de ellas (la segunda y la quinta) deciden responder, por lo que se producirá una colisión.

En el noveno intervalo, que corresponde a un intervalo posterior a uno de respuesta única, se decrementa Est_Etiq en 1 (pasando de 3,41 a 2,41) y se establece $Prob\_Act$ en función de la probabilidad anterior y el inverso de la estimación al cuadrado: $Incremento(Prob\_Act, \frac{1}{Est\_Etiq^2}) = Incremento(0{,}275, \frac{1}{2{,}41^2}) = Incremento(0{,}275, 0{,}172) = 0{,}275 + 0{,}172 \cdot (1 - 0{,}275) = 0{,}275 + 0{,}1248 = 0{,}4$. Finalmente, la primera y cuarta etiquetas deciden responder.

Como se puede observar, el valor de Est_Etiq se aproxima, progresivamente, al número de etiquetas pendientes de identificar, mientras que el valor de $Factor\_Mod$ tiende a 1, en los intervalos en los que está definido.

Finalmente, en el último intervalo, la estimación debería haberse decrementado en 1, sin embargo eso daría un valor de 0,55, por lo que, en lugar de ello, ésta se establece a 1.

6.2. Seguridad del protocolo

El protocolo descrito en este capítulo es un MAC (perteneciente a la capa de comunicación). Como la seguridad de las otras capas se considera garantizada, lo único que se necesita asegurar por parte del presente protocolo es precisamente la dificultad de deducir el número de etiquetas en el entorno a partir del número de respuestas. Podemos considerar tres tipos de adversarios, atendiendo a sus capacidades y objetivos.

(a) Atacante pasivo espiando el canal de comunicación

Su objetivo es obtener el número de etiquetas.

Para evitar que un espía pueda distinguir entre los tres tipos de intervalos (sin respuestas, una respuesta o colisión), una etiqueta ruidosa responderá con ruido en cada intervalo, de forma que, desde el punto de vista de un atacante, en todos los intervalos parecerá que ha habido una colisión. Además, después de una colisión o de un intervalo sin respuestas, la etiqueta ruidosa realizará una autenticación falsa, por lo que los atacantes no notarán diferencia entre intervalos de respuesta única y los otros dos tipos. Obviamente, el mensaje de la señal de intervalo, conteniendo el valor 0, 1 o 2, también debería ser cifrado por el protocolo de la capa de aplicación. En resumen, el atacante sólo verá un número fijo de iteraciones –determinado por el valor de *MAX_INTER* fijado en el protocolo– una colisión aparente y una autenticación en cada intervalo, así que será incapaz de deducir el número de etiquetas a partir de esta información.

(b.1) Atacante activo controlando todas las etiquetas del entorno durante un periodo de tiempo

Su objetivo es predecir respuestas futuras de la etiqueta ruidosa.

El ruido pseudoaleatorio debería generarse de manera que un atacante no lo pueda predecir. En caso contrario, un atacante que controlase todas las etiquetas del entorno (excepto la etiqueta ruidosa), podría saber las respuestas de todas las etiquetas y, por tanto, restarlas para obtener las respuestas de la etiqueta ruidosa. Si además pudiera predecir la secuencia de ruido de la etiqueta ruidosa a partir de estas respuestas, podría restar futuras respuestas de la etiqueta ruidosa cuando entrasen nuevas etiquetas (de una víctima potencial) en el entorno.

Un método seguro posible para generar la secuencia pseudoaleatoria es calcular una secuencia base inicial a partir de una semilla, y entonces aplicar una función *hash* criptográficamente segura (o, en general, cualquier función *one-way*) a cada valor de la secuencia. Así, si la etiqueta ruidosa tiene la semilla s_i, devolverá $h(s_i)$ (donde $h(\cdot)$ es la función *hash*), luego ésta generará el siguiente valor de la secuencia base $s_{i+1} = f(s_i)$ usando alguna función f apropiada. Otros métodos serían igualmente posibles [VV83].

(b.2) Atacante activo controlando etiquetas ruidosas adicionales

Su objetivo es realizar un ataque de denegación de servicio.

La presencia de etiquetas ruidosas adicionales generaría un ataque del tipo denegación de servicio. Pero este no es un problema que pueda ser controlado, ya que todos los protocolos están afectados por este tipo de ataque ante la presencia de fuentes de ruido electromagnético no controladas en su frecuencia de radio.

6.3. Análisis de rendimiento

En este apartado se realiza un análisis de rendimiento que tiene por objetivo mostrar que el mecanismo que implementa el protocolo propuesto escala al aumentar el número de etiquetas presentes en el sistema. Para ello, el protocolo se ha simulado usando un programa multitarea, con varias tareas actuando como las etiquetas y otra actuando como el lector. Este entorno ha sido programado usando el lenguaje Ada [TDB+06], puesto que es especialmente apropiado para la programación de sistemas multitarea. Mediante simulación, se muestra que se pueden identificar todas las etiquetas fijando un valor apropiado para el número máximo de intervalos, MAX_INTER.

6.3.1. Identificando todas las etiquetas

Los resultados experimentales se muestran en la Tabla 6.4, que colecta los datos para simulaciones correspondientes a cantidades de etiquetas de la forma 2^n, para $0 \leq n \leq 11$. Para cada cantidad de etiquetas, se han hecho 100 simulaciones. Obtuvimos los siguientes datos:

Intervalos: Media de intervalos necesarios para identificar todas las etiquetas.

Cero respuestas: Media y porcentaje de intervalos de cero respuestas.

Una respuesta: Media y porcentaje de intervalos de una respuesta.

Colisión: Media y porcentaje de intervalos de colisión.

Tabla 6.4: Estadísticas de la simulación.

Etiquetas	Intervalos	Cero respuestas		Una respuesta		Colisión	
1	4,2	3,2	76,2 %	1	23,8 %	0	0,0 %
2	6,8	4,4	64,7 %	2	29,4 %	0,4	5,9 %
4	13,4	6,0	44,8 %	4	29,9 %	3,4	25,4 %
8	21,2	8,2	38,7 %	8	37,7 %	5,0	23,6 %
16	42,6	15,0	35,2 %	16	37,6 %	11,6	27,2 %
32	93,6	32,7	34,9 %	32	34,2 %	28,9	30,9 %
64	185,0	64,2	34,7 %	64	34,6 %	56,8	30,7 %
128	365,0	123,4	33,8 %	128	35,1 %	113,6	31,1 %
256	751,4	251,0	33,4 %	256	34,1 %	244,4	32,5 %
512	1524,1	505,5	33,2 %	512	33,6 %	506,6	33,2 %
1024	3033,5	1002,3	33,0 %	1024	33,8 %	1007,2	33,2 %
2048	6067,0	2004,5	33,0 %	2048	33,8 %	2014,5	33,2 %

A partir de los resultados obtenidos vemos que las proporciones de intervalos con cero, una o múltiples respuestas convergen todas ellas a un tercio del número total de intervalos. Pero esto no significa que el protocolo tenga que ser ejecutado un número variable de intervalos, ya que eso proporcionaría una información muy valiosa al adversario (que podría deducir el número de etiquetas a partir del número de intervalos). La finalidad de esta primera simulación es demostrar que el protocolo escala bien cuando el número de etiquetas crece.

La Figura 6.3 muestra el gráfico del número total de intervalos, ejecutados por el lector, con respecto al número de etiquetas, comparado con la línea recta $y = 3x$ (indicada con línea de puntos). La simulación se paró cuando se identificaron todas las etiquetas. En la gráfica puede verse que la recta es una buena aproximación, especialmente cuando el número de etiquetas es mayor que 30.

Estos resultados se pueden tener en cuenta para el problema de decidir el número apropiado de intervalos que el lector debería ejecutar para identificar todas las etiquetas, suponiendo que uno sepa la máxima cantidad esperada. Como se propuso en la Sección 6.1.5, este número debería ser ligeramente mayor que el triple de la cantidad máxima esperada de etiquetas.

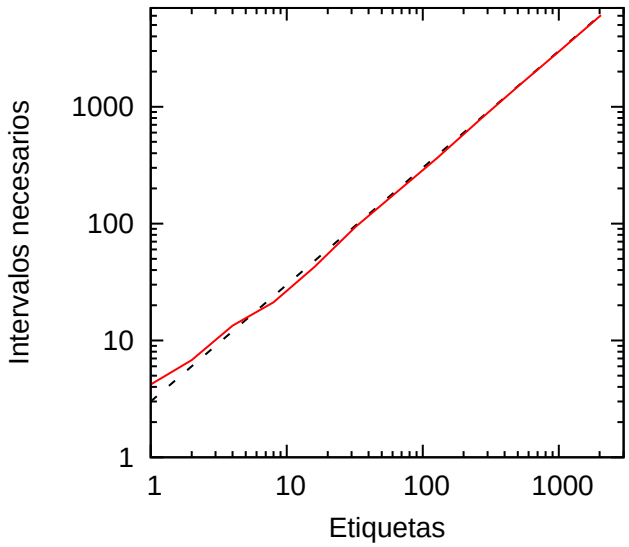


Figura 6.3: Intervalos necesarios según la cantidad de etiquetas.

Nótese que, mientras hay sistemas en los que la cantidad máxima de etiquetas puede conocerse *a priori* con bastante exactitud, como es el caso de las cadenas de montaje, esto no siempre será así. Por ejemplo, si se implanta un sistema de detección de billetes etiquetados en un aeropuerto, es posible que un pasajero lleve muchos más billetes de lo previsto por el sistema. Sin embargo, aunque eso implique que el lector no los identificará todos, sí que detectará que ha identificado una cantidad anormalmente alta, por lo que se podría mandar un aviso de seguridad para realizar un registro más a fondo del pasajero en cuestión.

6.3.2. Eficiencia del protocolo

Para evaluar la eficiencia del protocolo propuesto, se ha comprobado experimentalmente la cantidad de etiquetas que se pueden identificar en un número fijo de intervalos, incrementando el número de etiquetas hasta sobrepasar $MAX\_INTER/3$. Para hacer esto, se han hecho varias simulaciones para cantidades de etiquetas de la forma 2^n, para $0 \leq n \leq 11$, con MAX_INTER fijado a 500. Igual que antes, se han realizado 100 simulaciones para cada cantidad

de etiquetas. La Tabla 6.5 recoge los datos medios para estas simulaciones. El número de etiquetas identificadas corresponde al número de intervalos con una sola respuesta.

Tabla 6.5: Simulación de 500 intervalos.

Etiquetas	Cero respuestas		Una respuesta		Colisión	
1	499	99,8 %	1	0,2 %	0	0,0 %
2	497,7	99,5 %	2	0,4 %	0,3	0,1 %
4	492,9	98,6 %	4	0,8 %	3,1	0,6 %
8	486,7	97,3 %	8	1,6 %	5,3	1,1 %
16	470,9	94,2 %	16	3,2 %	13,1	2,6 %
32	439,2	87,8 %	32	6,4 %	28,8	5,8 %
64	378,5	75,7 %	64	12,8 %	57,5	11,5 %
128	258,2	51,6 %	128	25,6 %	113,8	22,8 %
256	169,4	33,9 %	165,6	33,1 %	165,0	33,0 %
512	164,4	32,9 %	168,4	33,7 %	167,2	33,4 %
1024	163,4	32,7 %	164,8	33,0 %	171,8	34,3 %
2048	162,8	32,5 %	164,3	32,9 %	172,9	34,6 %

Podemos ver que para cantidades de etiquetas por debajo de un tercio del número de intervalos (hasta las 128 etiquetas), todas ellas han sido identificadas. Como cada etiqueta autenticada deja de participar en el protocolo, el número de intervalos sin respuestas es mayor en estos casos.

Por otra parte, para grandes cantidades de etiquetas (a partir de 256), el número de etiquetas identificadas es cercano a un tercio del número de intervalos (i.e. aproximadamente 166). Lo mismo sucede con la proporción de intervalos sin respuestas o con colisión. Esto sigue siendo válido para cualquier cantidad de etiquetas por encima de ese nivel. Nótese que esto no es normalmente cierto para otros protocolos MAC, que tienden a degradar la frecuencia de respuesta cuando el número de participantes crece [Abr70].

Con respecto al tiempo necesario para identificar todas las etiquetas, éste se determina multiplicando el valor de MAX_INTER por el tiempo necesario para identificar una etiqueta, ya que el resto del protocolo no representa una gran sobrecarga. Así, el factor clave para el coste total es el tiempo que necesita el protocolo de la capa de aplicación.

A partir de los resultados mostrados, puede verse que el protocolo MAC propuesto en el presente capítulo cumple con su doble objetivo. Por una parte, da solución al problema del recuento de etiquetas, evitando que un atacante pueda distinguir cuantas etiquetas responden en cada operación de lectura. Por otra parte, permite la correcta identificación de todas las etiquetas presentes en el entorno si se conoce *a priori* el número máximo que puede haber.

Capítulo 7

Conclusiones y trabajos futuros

La presente tesis se centra en el estudio de mecanismos de seguridad para los sistemas de identificación por radiofrecuencia, RFID. Las aportaciones de la tesis se enmarcan tanto en la capa de aplicación como en la de comunicación.

En cuanto a la capa de aplicación, se ha propuesto un protocolo que permite la identificación segura de las etiquetas frente al lector. Para ello, el protocolo combina técnicas criptográficas con curvas elípticas: el protocolo de conocimiento nulo de Schnorr para autenticar al lector, y un procedimiento de actualización segura del secreto de la etiqueta, que permite prevenir el seguimiento de las mismas.

El uso de criptografía elíptica nos ha permitido ajustarnos a las limitaciones computacionales y de memoria de las etiquetas RFID y, a la vez, dotar el sistema de niveles de seguridad considerables. De este modo, el sistema propuesto provee un canal seguro entre etiquetas y lector que, además, puede resultar útil en el caso que las etiquetas precisaran transmitir de forma segura pequeños valores de sus posibles sensores.

En cuanto a la implementabilidad del sistema, se han analizado los requerimientos técnicos para estudiar y proponer una selección adecuada de los parámetros involucrados en el protocolo, y se ha demostrado que el protocolo es escalable cuando el número de etiquetas crece. En el análisis de seguridad, se ha visto que el sistema es resistente ante los ataques de *sniffing*, seguimiento de etiquetas, suplantación, reutilización y *forward attack*.

Para la implementación de este protocolo para RFID, resulta imprescindible

disponer de curvas elípticas criptográficamente útiles. Con el fin de agilizar la generación de tales curvas, hemos planteado la paralelización de un algoritmo de construcción de volcanes de isogenias. Dos curvas pertenecientes al mismo grafo volcán van a ser isógenas y, por consiguiente, tendrán el mismo cardinal. Entonces, van a poder ofrecer niveles de seguridad criptográfica similares.

La paralelización del algoritmo de construcción del volcán divide la aplicación en tres tipos de tareas: una tarea maestra que coordina el funcionamiento, dos tareas cráter que construyen el cráter del volcán en direcciones opuestas, y una cantidad configurable de tareas ladera que construyen los árboles que penden de los nodos del cráter.

Desde un punto de vista de rendimiento, es crucial determinar el número adecuado de tareas ladera a ejecutar, para obtener los mejores resultados. Como dicho número debe elegirse *a priori*, hemos realizado un análisis teórico de la aplicación que nos permite encontrar una expresión analítica para este valor.

En último lugar, hemos propuesto un protocolo MAC para la capa de comunicación, que es escalable y evita el revelado ante un adversario del número de etiquetas RFID presentes en el entorno. Este protocolo puede ser especialmente apropiado en sistemas donde el revelado de tal información resulte perjudicial para la seguridad. Por ejemplo, el Banco Central Europeo ha estado estudiando la posibilidad de añadir etiquetas RFID a los billetes de euro de curso legal. Sin duda, revelar la cantidad de billetes que un ciudadano lleva encima, supondría una vulneración a su intimidad y un grave problema de seguridad.

La idea principal del protocolo radica en que las etiquetas respondan de modo que el número de respuestas recibidas por el lector, durante la operación de lectura, sea independiente del número de etiquetas presentes en el entorno. Por un lado, se utiliza un algoritmo probabilístico capaz de adaptarse a diferentes escenarios y, por otro lado, se sitúa una etiqueta ruidosa en el entorno, de manera que un atacante no pueda obtener ninguna información al escuchar el canal de comunicación.

El trabajo realizado en la presente tesis ha dejado cuestiones abiertas que no hemos podido abordar hasta el momento, y que se plantean como cuestiones

a tratar en trabajos futuros.

Dado que la simulación que se ha realizado de la paralelización del algoritmo de generación de grafos ℓ-volcán, con $\ell > 2$, ha mostrado que es factible alcanzar una buena eficiencia, un posible trabajo futuro sería, precisamente, la implementación real del mismo para estos casos.

Por otro lado, los protocolos propuestos para las capas de aplicación y comunicación se han planteado y analizado de forma independiente. Se debería abordar su integración, y estudiar su implementabilidad conjunta. Incluso se podría estudiar la seguridad de la capa física, y abordar una implementación real del sistema.

Bibliografía

[Abr70] Norman Abramson. The ALOHA System: Another Alternative for Computer Communications. In *Fall Joint Computer Conference. International Workshop on Managing Requirements Knowledge*, AFIPS, pages 281–285. ACM, 1970.

[AD94] Leonard M. Adleman and Jonathan DeMarrais. A Subexponential Algorithm for Discrete Logarithms over All Finite Fields. In *Advances in Cryptology – CRYPTO'93*, volume 773 of *LNCS*, pages 147–158. Springer, 1994.

[AKQ08] Gildas Avoine, Kassem Kalach, and Jean-Jacques Quisquater. ePassport: Securing International Contacts with Contactless Chips. In *Financial Cryptography and Data Security – FC*, volume 5143 of *LNCS*, pages 141–155. IFCA, Springer, 2008.

[Ang05] Rebecca Angeles. RFID Technologies: Supply-Chain Applications and Implementation Issues. *Information Systems Management*, 22:51–65, 2005.

[AO05a] Gildas Avoine and Philippe Oechslin. A Scalable and Provably Secure Hash-Based RFID Protocol. In *International Workshop on Pervasive Computing and Communication Security – PerSec*, pages 110–114. IEEE Computer Society, 2005.

[AO05b] Gildas Avoine and Philippe Oechslin. RFID Traceability: A Multilayer Problem. In *Financial Cryptography – FC*, volume 3570 of *LNCS*, pages 125–140. IFCA, Springer, 2005.

[ASTP10] Guillermo Azuara, José Luis Salazar, José Luis Tornos, and Joan Josep Piles. Reliable Food Traceability Using RFID Tagging. In *Financial Cryptography and Data Security (FC), First International Workshop on Lightweight Cryptography for Resource-Constrained Devices (WLC)*, volume 6054 of *LNCS*, pages 57–67. IFCA, Springer, 2010.

[Aut] Auto-ID Labs. http://www.autoidlabs.org/page.html. formerly MIT Auto-ID Center.

[Aut02] Auto-ID Center. 860MHz-960MHz Class I Radio Frequency Identification Tag Radio Frequency and Logical Communication Interface Specification Proposed Recommendation Version 1.0.0. Technical Report MIT-AUTOID-TR-007, 2002.

[BC06] Julien Bringer and Hervé Chabanne. On the Wiretap Channel Induced by Noisy Tags. In *European Workshop on Security and Privacy in Ad hoc and Sensor Networks – ESAS*, volume 4357 of *LNCS*, pages 113–120. Springer, 2006.

[BGK+06a] Lejla Batina, Jorge Guajardo, Tim Kerins, Nele Mentens, Pim Tuyls, and Ingrid Verbauwhede. An Elliptic Curve Processor Suitable For RFID-Tags. Cryptology ePrint Archive, Report 2006/227, 2006.

[BGK+06b] Lejla Batina, Jorge Guajardo, Tim Kerins, Nele Mentens, Pim Tuyls, and Ingrid Verbauwhede. Public-Key Cryptography for RFID-Tags. In *Workshop on RFID Security – RFIDSec*, pages 61–76. Ecrypt, 2006.

[BGK+07] Lejla Batina, Jorge Guajardo, Tim Kerins, Nele Mentens, Pim Tuyls, and Ingrid Verbauwhede. Public-Key Cryptography for RFID-Tags. In *International Workshop on Pervasive Computing and Communication Security – PerSec*, pages 217–222. IEEE Computer Society, 2007.

[BSNP95] Sasan Bakhtiari, Rei Safavi-Naini, and Josef Pieprzyk. Cryptographic Hash Function: A Survey. Technical Report 95–09, 1995.

[BSS99] Ian F. Blake, Gadiel Seroussi, and Nigel P. Smart. *Elliptic Curves in Cryptography*, volume 265 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, 1999.

[CA06] Claude Castelluccia and Gildas Avoine. Noisy Tags: A Pretty Good Key Exchange Protocol for RFID Tags. In *International Conference on Smart Card Research and Advanced Applications – CARDIS*, volume 3928 of *LNCS*, pages 289–299. IFIP, Springer, 2006.

[CC86] David V. Chudnovsky and Gregory V. Chudnovsky. Sequences of numbers generated by addition in formal groups and new primality and factorization tests. *Advances in Applied Mathematics*, 7(4):385–434, 1986.

[Cer00] Certicom Corp. SECG. SEC 2: Recommended Elliptic Curve Domain Parameters *(v1.0)*. Standards for Efficient Cryptography Group, 2000. *http://www.secg.org/*.

[Cer10] Certicom Corp. SECG. SEC 2: Recommended Elliptic Curve Domain Parameters *(v2.0)*. Standards for Efficient Cryptography Group, 2010. *http://www.secg.org/*.

[CEvdGP87] David Chaum, Jan-Hendrik Evertse, Jeroen van de Graaf, and René Peralta. Demonstrating Possession of a Discrete Logarithm Without Revealing it. In *Advances in Cryptology – CRYPTO'86*, volume 263 of *LNCS*, pages 200–212. Springer, 1987.

[CJ08] Soo-Hyun Choi and You-Hyeon Jeong. A Secure and Scalable Transaction Protocol for Ubiquitous Sensor Network using RFID Systems. In *10th International Conference on Advanced Communication Technology – ICACT*, volume 3, pages 1781–1784. IEEE Computer Society, 2008.

[CLL07] Eun Young Choi, Su Mi Lee, and Dong Hoon Lee. Self-updating: Strong Privacy Protection Protocol for RFID-Tagged Banknotes. In *Ubiquitous Intelligence and Computing*, volume 4611 of *LNCS*, pages 1171–1180. Springer, 2007.

[Cor98] Certicom Corp. The Elliptic Curve Cryptosystem For Smart Cards. *http://www.comms.scitech.susx.ac.uk/fft/ crypto/ECC_SC.pdf, Certicom White Paper*, The seventh in a series of ECC white papers, 1998.

[DS98] Michael W. David and Kouichi Sakurai. Security Issues for Contactless Smart Cards. In *Public Key Cryptography: First International Workshop on Practice and Theory in Public Key Cryptography – PKC*, volume 1431 of *LNCS*, pages 247–252. Springer, 1998.

[EFG+10] Daniel Engels, Xinxin Fan, Guang Gong, Honggang Hu, and Eric M. Smith. Hummingbird: Ultra-Lightweight Cryptography for Resource-Constrained Devices. In *Financial Cryptography and Data Security (FC), First International Workshop on Lightweight Cryptography for Resource-Constrained Devices (WLC)*, volume 6054 of *LNCS*, pages 3–18. IFCA, Springer, 2010.

[FFS87] Uriel Feige, Amos Fiat, and Adi Shamir. Zero Knowledge Proofs of Identity. In *Symposium on Theory of Computing*, pages 210–217. ACM, 1987.

[Fin98] Klaus Finkenzeller. *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication.* John Wiley & Sons, Ltd, 1998.

[FM02] Mireille Fouquet and François Morain. Isogeny Volcanoes and the SEA Algorithm. In *Algorithmic Number Theory – ANTS-V*, volume 2369 of *LNCS*, pages 47–62. Springer, 2002.

[Fou01] Mireille Fouquet. *Anneau d'endomorphismes et cardinalité des courbes elliptiques: aspects algorithmiques.* PhD thesis, École Polytechnique, 2001.

[FW06] Christian Floerkemeier and Matthias Wille. Comparison of Transmission Schemes for Framed ALOHA based RFID Protocols. In *International Symposium on Applications and the Internet Workshops – SAINTW*, pages 92–97. IEEE Computer Society, 2006.

[GHS02] Pierrick Gaudry, Florian Hess, and Nigel P. Smart. Constructive and destructive facets of Weil descent on elliptic curves. *Journal of Cryptology*, 15:19–46, 2002.

[GRRL04] Fernando Guirado, Ana Ripoll, Concepció Roig, and Emilio Luque. Performance Prediction Using an Application Oriented Mapping Tool. *Proc. Euromicro Conf. on Parallel, Distributed and Network-based Processing*, pages 184–191, 2004.

[GS99] Steven D. Galbraith and Nigel P. Smart. A Cryptographic Application of Weil Descent. In *Cryptography and Coding*, volume 1746 of *LNCS*, pages 191–200. Springer, 1999.

[Has33] Helmut Hasse. Beweis des Analogons der Riemannschen Vermutung für die Artinschen und F. K. Schmidtschen Kongruenzzetafunktionen in gewissen elliptischen Fällen, Vorläufie Mitteilung. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen I*, 42(3):253–262, 1933.

[HH88] Peter J. Harrop and Teresa Henry. *Chipless Smart Labels: The Ultimate RFID.* IDTechEx Ltd., 1988.

[HHJ+06] Jaap-Henk Hoepman, Engelbert Hubbers, Bart Jacobs, Martijn Oostdijk, and Ronny Wichers Schreur. Crossing Borders: Security and Privacy Issues of the European e-Passport. In *Advances in Information and Computer Security, First International*

Workshop on Security – IWSEC, volume 4266 of *LNCS*, pages 152–167. Springer, 2006.

[HLS09] Henry Holtzman, Sanghoon Lee, and Daniel Shen. OpenTag: Privacy Protection for RFID. *IEEE Pervasive Computing*, 8(2):71–77, 2009.

[JP03] Ari Juels and Ravikanth Pappu. Squealing Euros: Privacy Protection in RFID-Enabled Banknotes. In *Financial Cryptography – FC*, volume 2742 of *LNCS*, pages 103–121. IFCA, Springer, 2003.

[JRS03] Ari Juels, Ronald Rivest, and Michael Szydlo. The Blocker Tag: Selective Blocking of RFID Tags for Consumer Privacy. In *Conference on Computer and Communications Security – ACM CCS*, pages 103–111. ACM, 2003.

[Jue06] Ari Juels. RFID Security and Privacy: A Research Survey. *IEEE Journal on Selected Areas in Communications (J-SAC)*, 24(2):381–395, 2006.

[KDF75] Alfred R. Koelle, Steven W. Depp, and Robert W. Freyman. Short-Range Radio-Telemetry for Electronic Identification Using Modulated Backscatter. *Proceedings of the IEEE*, 63(8):1260–1261, 1975.

[KHK+03] Shingo Kinoshita, Fumitaka Hoshino, Tomoyuki Komuro, Akiko Fujimura, and Miyako Ohkubo. Nonidentifiable Anonymous-ID Scheme for RFID Privacy Protection. *Joho Shori Gakkai Shinpojiumu Ronbunshu CSS (in Japanese)*, 15:497–502, 2003.

[Kob84] Neal Koblitz. *Introduction to elliptic curves and modular forms.* Springer, 1984.

[Kob87] Neal Koblitz. Elliptic Curve Cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.

[KYLS06] Girish Khandelwal, Aylin Yener, Kyounghwan Lee, and Semih Serbetli. ASAP: A MAC Protocol for Dense and Time-Constrained RFID Systems. In *IEEE International Conference on Communications – ICC*, volume 9, pages 4028–4033. IEEE Computer Society, 2006.

[LD99] Julio López and Ricardo Dahab. Fast Multiplication on Elliptic Curves Over $GF(2^m)$ without precomputation. In *Cryptographic Hardware and Embedded Systems – CHES'99*, volume 1717 of *LNCS*, pages 316–327. Springer, 1999.

[Len87] Hendrik Willem Lenstra, Jr. Factoring Integers with Elliptic Curves. *The Annals of Mathematics, Second Series*, 126(3):649–673, 1987.

[LK06] Chae Hoon Lim and Tymur Korkishko. mCrypton - A Lightweight Block Cipher For Security of Low-Cost RFID Tags and Sensors. In *Workshop on Information Security Applications – WISA'05*, volume 3786 of *LNCS*, pages 243–258. Springer, 2006.

[LLH+07] Li Lu, Yunhao Liu, Lei Hu, Jinsong Han, and Lionel M. Ni. A Dynamic Key-Updating Private Authentication Protocol for RFID Systems. In *International Conference on Pervasive Computing and Communications – PerCom*, pages 13–22. IEEE Computer Society, 2007.

[Mil86] Victor S. Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology – CRYPTO'85*, volume 218 of *LNCS*, pages 417–426. Springer, 1986.

[ML75] Ward Douglas Maurer and Theodore Gyle Lewis. Hash Table Methods. *ACM Computing Surveys – CSUR*, 7(1):5–19, 1975.

[MMS+02] Josep M. Miret, Ramiro Moreno, Daniel Sadornil, Juan Tena, and Magda Valls. Construcción de curvas isógenas criptográficamente útiles. In *VII Reunión Española sobre Criptología y Seguridad de la Información – RECSI*, pages 579–587, 2002.

[MMS+06] Josep M. Miret, Ramiro Moreno, Daniel Sadornil, Juan Tena, and Magda Valls. An algorithm to compute volcanoes of 2-isogenies of elliptic curves over finite fields. *Applied Mathematics and Computation*, 176(2):739–750, 2006.

[MMS+08] Josep M. Miret, Ramiro Moreno, Daniel Sadornil, Juan Tena, and Magda Valls. Computing the height of volcanoes of ℓ-isogenies of elliptic curves over finite fields. *Applied Mathematics and Computation*, 196(1):67–76, 2008.

[MP10] Amir Moradi and Axel Poschmann. Lightweight Cryptography and DPA Countermeasures: A Survey. In *Financial Cryptography and Data Security (FC), First International Workshop on Lightweight Cryptography for Resource-Constrained Devices (WLC)*, volume 6054 of *LNCS*, pages 68–79. IFCA, Springer, 2010.

[MPI94] MPI (Message Passing Interface) Forum. MPI: A Message-passing Interface Standard. *Journal of Supercomputer Applications*, 8(3/4), 1994.

[MRT08] Aikaterini Mitrokotsa, Melanie R. Rieback, and Andrew S. Tanenbaum. Classification of RFID Attacks. In *2nd International Workshop on RFID Technology – IWRT*, pages 73–86, 2008.

[MRT09] Aikaterini Mitrokotsa, Melanie R. Rieback, and Andrew S. Tanenbaum. Classifying RFID attacks and defenses. *Information Systems Frontiers*, 12:491–505, 2009.

[MRV10] Santi Martínez, Concepció Roig, and Magda Valls. Securing the Use of RFID-Enabled Banknotes. In *Financial Cryptography and Data Security (FC), First International Workshop on Lightweight Cryptography for Resource-Constrained Devices (WLC)*, volume 6054 of *LNCS*, pages 80–93. IFCA, Springer, 2010.

[MST+04] Ramiro Moreno, Daniel Sadornil, Juan Tena, Magda Valls, and Francisco Romero. Enumeración de curvas elípticas mediante

grafos de 2-isogenias. *IV Jornadas de Matemática Discreta y Algorítmica*, pages 187–194, 2004.

[MST+07] Josep M. Miret, Daniel Sadornil, Juan Tena, Rosana Tomàs, and Magda Valls. Volcanoes of ℓ–isogenies of elliptic curves over finite fields: the case $\ell = 3$. *Publicacions Matemàtiques*, 1:165–180, 2007.

[MST+08] Josep M. Miret, Daniel Sadornil, Juan Tena, Rosana Tomàs, and Magda Valls. Exploiting Isogeny Cordillera Structure to Obtain Cryptographically Good Elliptic Curves. *Journal of Research and Practice in Information Technology*, 40(4):255–265, 2008.

[MST+09] Josep M. Miret, Daniel Sadornil, Juan Tena, Rosana Tomàs, and Magda Valls. On Avoiding ZVP–Attacks Using Isogeny Volcanoes. In *Workshop on Information Security Applications – WISA'08*, volume 5379 of *LNCS*, pages 266–277. Springer, 2009.

[MST+10] Santi Martínez, Daniel Sadornil, Juan Tena, Rosana Tomàs, and Magda Valls. Curvas de Edwards y ataques basados en puntos de valor cero (ZVP). In *XI Reunión Española sobre Criptología y Seguridad de la Información – RECSI*, pages 49–53, 2010.

[MT06] Alfred J. Menezes and Edlyn Teske. Cryptographic implications of Hess' generalized GHS attack. *Applicable Algebra in Engineering, Communication and Computing*, 16:439–460, 2006.

[MTR+05] Santi Martínez, Rosana Tomàs, Concepció Roig, Magda Valls, and Francesc Giné. Paralelización del cálculo de volcanes para usos criptográficos. In *I Congreso Español de Informática (CEDI), XVI Jornadas de Paralelismo*, pages 615–622, 2005.

[MTR+06] Santi Martínez, Rosana Tomàs, Concepció Roig, Magda Valls, and Ramiro Moreno. Parallel Calculation of Volcanoes for Cryptographic Uses. In *20th IEEE International Parallel & Distributed Processing Symposium (IPDPS), Workshop on Parallel*

and Distributed Scientific and Engineering Computing (PDSEC), page 307. IEEE Computer Society, 2006.

[MvOV96] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[MVR+07] Santi Martínez, Magda Valls, Concepció Roig, Francesc Giné, and Josep M. Miret. An Elliptic Curve and Zero Knowledge Based Forward Secure RFID Protocol. In *Conference on RFID Security – RFIDSec*, pages 129–140. Ecrypt, 2007.

[MVR+09] Santi Martínez, Magda Valls, Concepció Roig, Josep M. Miret, and Francesc Giné. A Secure Elliptic Curve-Based RFID Protocol. *Journal of Computer Science and Technology*, 24(2):309–318, 2009.

[MW04] David Molnar and David Wagner. Privacy and Security in Library RFID: Issues, Practices, and Architectures. In *Conference on Computer and Communications Security – ACM CCS*, pages 210–219. ACM, 2004.

[NLLP04] Lionel M. Ni, Yunhao Liu, Yiu Cho Lau, and Abhishek P. Patil. LANDMARC: Indoor Location Sensing Using Active RFID. *Wireless Networks*, 10:701–710, 2004.

[OF09] Yossef Oren and Martin Feldhofer. A Low-Resource Public-Key Identification Scheme for RFID Tags and Sensor Nodes. In *Conference on Wireless Network Security – WiSec*, pages 59–68. ACM, 2009.

[OSK03] Miyako Ohkubo, Koutarou Suzuki, and Shingo Kinoshita. Cryptographic Approach to "Privacy-Friendly" Tags. In *RFID Privacy Workshop*, pages 1–9, 2003.

[QGB90] Jean-Jacques, Myriam, Muriel and Michaël Quisquater, Louis C., Marie Annick, Gaïd, Anna, Gwenolé and Soazig Guillou, and Thomas A. Berson. How to Explain Zero-Knowledge Protocols

to Your Children. In *Advances in Cryptology – CRYPTO'89*, volume 435 of *LNCS*, pages 628–631. Springer, 1990.

[RCT05] Melanie Rieback, Bruno Crispo, and Andrew Tanenbaum. RFID Guardian: A Battery-Powered Mobile Device for RFID Privacy Management. In *Australasian Conference on Information Security and Privacy – ACISP*, volume 3574 of *LNCS*, pages 184–194. Springer, 2005.

[RCT06] Melanie R. Rieback, Bruno Crispo, and Andrew S. Tanenbaum. The Evolution of RFID Security. *IEEE Pervasive Computing*, 5(1):62–69, 2006.

[RGC+06] Melanie R. Rieback, Georgi Gaydadjiev, Bruno Crispo, Rutger F. H. Hofman, and Andrew S. Tanenbaum. A Platform for RFID Security and Privacy Administration. In *SAGE conference on Large Installation System Administration – LISA*, pages 8–8. USENIX Association, 2006.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[Sch95] René Schoof. Counting points on elliptic curves over finite fields. *Journal de théorie des nombres de Bordeaux*, 7(1):219–254, 1995.

[Sha72] Daniel Shanks. Five number-theoretic algorithms. In *2nd Manitoba Conference on Numerical Mathematics*, Congressus Numerantium VII, pages 51–70, 1972.

[Sin99] Simon Singh. *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography.* Fourth Estate, 1999.

[Sta10] William Stallings. *Cryptography and Network Security: Principles and Practice, Fifth Edition.* Prentice Hall, 2010.

[Sto48] Harry Stockman. Communication by Means of Reflected Power. *Proceedings of the IRE (Institute of Radio Engineers)*, 36(10):1196–1204, 1948.

[SV00] Jaspal Subhlok and Gary Vondran. Optimal Use of Mixed Task and Data Parallelism for Pipelined Computations. *Journal of Parallel and Distributed Computing*, 60(3):297–319, 2000.

[SWE03] Sanjay Sarma, Stephen Weis, and Daniel Engels. RFID Systems and Security and Privacy Implications. In *Cryptographic Hardware and Embedded Systems – CHES'02*, volume 2523 of *LNCS*, pages 454–469. Springer, 2003.

[TB06] Pim Tuyls and Lejla Batina. RFID-Tags for Anti-counterfeiting. In *Topics in Cryptology - CT-RSA, The Cryptographers' Track at the RSA Conference*, volume 3860 of *LNCS*, pages 115–131. Springer, 2006.

[TDB+06] S. Tucker Taft, Robert A. Duff, Randall L. Brukardt, Erhard Ploedereder, and Pascal Leroy, editors. *Ada 2005 Reference Manual: Language and Standard Libraries. International Standard ISO/IEC 8652/1995 (E) with Technical Corrigendum 1 and Amendment 1*, volume 4348 of *LNCS*. Springer, 2006.

[TK95] Jason Paul Toonstra and Witold Kinsner. Transient analysis and genetic algorithms for classification. In *IEEE Western Canada Conference and Exhibition – WESCANEX*, volume 2 of *Communications, Power, and Computing*, pages 432–437. IEEE Computer Society, 1995.

[TM99] Neil T. Trask and Michael Victor Meyerstein. Smart Cards in Electronic Commerce. *BT Technology Journal*, 17(3):57–66, 1999.

[Tro06] Trolley Scan (Pty) Ltd. Issues on range and accuracy of RFID-radar™ system. Matching the radar to the physical situation.

http://rfid-radar.com, 2006. Paper on measurement accuracy of RFID-radar.

[VV83] Umesh V. Vazirani and Vijay V. Vazirani. Trapdoor pseudo-random number generators, with applications to protocol design. *Annual IEEE Symposium on Foundations of Computer Science*, 0:23–30, 1983.

[WES03] James Waldrop, Daniel W. Engels, and Sanjay E. Sarma. Colorwave: A MAC for RFID Reader Networks. *Wireless Communications and Networking (WCNC)*, 3:1701–1704, 2003.

[Wik] Wikipedia, the free encyclopedia. Electronic article surveillance. en.wikipedia.org/wiki/Electronic%5Farticle%5Fsurveillance. Last check: December 2010.

[Wil95] Andrew John Wiles. Modular elliptic curves and Fermat's Last Theorem. *Annals of Mathematics*, 141:443–551, 1995.

[WLY07] Jianping Wang, Huiyun Li, and Fengqi Yu. Design of Secure and Low-Cost RFID Tag Baseband. In *International Conference on Wireless Communications, Networking and Mobile Computing (WiCom)*, pages 2066–2069, 2007.

[WSRE03] Stephen A. Weis, Sanjay E. Sarma, Ronald L. Rivest, and Daniel W. Engels. Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. In *First International Conference on Security in Pervasive Computing – SPC*, volume 2802 of *LNCS*, pages 454–469. Springer, 2003.

[YY03] Kim Yong-Young. Radio ID chips may track banknotes. http://news.cnet.com/2100-1017-1009155.html, 2003. CNET News.com.