PROTOTIPO DE APLICACIÓN SEGURA PARA LA TRANSFERENCIA REMOTA DE DINERO EN ENTORNOS WEB, BASADO EN CRIPTOGRAFÍA DE CURVA ELÍPTICA

LUIS EDUARDO ZAMBRANO FERNÁNDEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2007

PROTOTIPO DE APLICACIÓN SEGURA PARA LA TRANSFERENCIA REMOTA DE DINERO EN ENTORNOS WEB, BASADO EN CRIPTOGRAFÍA DE CURVA ELÍPTICA

LUIS EDUARDO ZAMBRANO FERNÁNDEZ

Trabajo de grado para optar al título de
Ingeniero de Sistemas

Director
MANUEL GUILLERMO FLÓREZ
Ingeniero de Sistemas, UIS.

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2007

A mis padres, Enith y Fernando quienes siempre confiaron en que alcanzaría este logro.

AGRADECIMIENTOS

El autor expresa sus gratitudes a la Universidad Industrial de Santander, en especial a sus docentes por su constante aporte a la construcción del saber, al ingeniero Manuel Guillermo Flórez por su orientación profesional, a Manuel Pancorbo Castro y Julio López Hernández por permitirme compartir conocimientos, al profesor Edilberto Reyes González por su disposición, a mis compañeros de trabajo por la socialización profesional, y a mi familia por su apoyo y aliento en los momentos difíciles.

CONTENIDO

pág.

LISTA DE TABLAS

pág.

LISTA DE FIGURAS

pág.

LISTA DE ANEXOS

TITULO: **PROTOTIPO DE APLICACIÓN SEGURA PARA LA TRANSFERENCIA REMOTA DE DINERO EN ENTORNOS WEB, BASADO EN CRIPTOGRAFÍA DE CURVA ELÍPTICA.**[*]

AUTOR: **LUIS EDUARDO ZAMBRANO FERNÁNDEZ.**[**]

Palabras Claves: Criptosistema, Curvas Elípticas, Dinero, Seguridad Informática, Criptografía, Cifrado, Firma Digital, ECC, CCE.

RESUMEN

En este trabajo de grado se ha diseñado e implementado un criptosistema como prototipo de aplicación segura, en donde se genera, almacena, organiza y se transmite información financiera en entornos Web haciendo uso de algoritmos de curva elíptica. La transferencia de datos entre dos entidades financieras se hace bajo el marco de un protocolo de alto nivel diseñado sobre el protocolo HTTP. Los documentos se envían cifrados y firmados digitalmente garantizando la confidencialidad, autenticidad, integridad y no repudio de la información; la gestión de claves públicas se hace a través de un tercero confiable que actúa como entidad de certificación.

Inicialmente, se hace una introducción teórica a los conceptos básicos necesarios para entender los criptosistemas de curvas elípticas, se dan algunos conceptos de teoría de números, álgebra abstracta, teoría de la información y complejidad algorítmica. Se describen las curvas elípticas sobre campos finitos, su definición y sus operaciones, se evalúa cuáles son las curvas aptas para aplicaciones criptográficas (curvas no anómalas y no singulares) y cómo éstas pueden ser aplicadas a la criptografía; se define el problema del logaritmo discreto sobre curvas elípticas (PLDCE).

Finalmente, se describe en detalle el criptosistema desarrollado, la terminología utilizada y su manual de usuario, el modelo de datos y las reglas y estados que definen el protocolo de comunicación entre las entidades financieras y entre éstas y un tercero confiable que gestiona las claves públicas.

---

[*] Trabajo de Grado.

[**] Facultad de Ingenierías Fisicomecánicas, Escuela de Ingeniería de Sistemas.
Director: Manuel Guillermo Flórez Becerra.

TITULO: **PROTOTIPO DE APLICACIÓN SEGURA PARA LA TRANSFERENCIA REMOTA DE DINERO EN ENTORNOS WEB, BASADO EN CRIPTOGRAFÍA DE CURVA ELÍPTICA.**[*]

AUTOR: **LUIS EDUARDO ZAMBRANO FERNÁNDEZ.**[**]

Key words: Cryptosystem, Elliptic Curve, Money, Computer Security, Cryptography, Encrypt, Digital Signature, ECC, CCE.

ABSTRACT

In this degree project, a cryptosystem as a prototype of a safe application has been designed and performed within which financial information in Web environment is generated, stored, organized and transmitted by using elliptic curve algorithms. Data transfer between two financial entities is done under a high level protocol designed over the HTTP protocol. Documents are sent encrypted and digitally signed up guarantying confidentiality, authenticity, integrity, and non rejection of the information; management of public keys is done by a reliable third party who acts as a certification authority.

Initially, a theoretical introduction to basic concepts that is necessary for the understanding of elliptic curve cryptosystems is done. Some concepts of the number theory, abstract algebra, the theory of information, and algorithmic complexity are given. Elliptic curves over finite fields; their definitions and operations are described. The appropriate curves for cryptographic applications (not anomalous curves and nonsingular) are evaluated and how they can be applied to cryptography; the elliptic curve discrete logarithm problem (ECDLP) is also defined.

Finally it is defined in detail the developed cryptosystem, the terminology used and its user manual, data model, rules and states that define the communication protocol between financial entities and between them and a reliable third party that manages public keys.

---

[*] Trabajo de Grado.

[**] Facultad de Ingenierías Fisicomecánicas, Escuela de Ingeniería de Sistemas.
Director: Manuel Guillermo Flórez Becerra.

INTRODUCCIÓN.

La transferencia de información a sufrido un cambio cualitativo desde la aparición de la Internet, la naturaleza misma de los entornos Web hacen de las redes un medio inseguro para transferir datos; casos de suplantación, alterabilidad y hurto de la información han costado en Colombia cerca de 6,6 billones de pesos a empresas y más de 349.000 millones de pesos a personas naturales[1]. Este flagelo en nuestro país a tenido un crecimiento acelerado en los últimos 5 años y se ha observado que a medida que crecen las nuevas tecnologías también lo hacen proporcionalmente los problemas asociados a ellas.

Desde la universidad, debemos aportar a la solución de esta problemática, investigar, innovar e implantar soluciones que ayuden a reducir los riesgos a los que se somete la información y diseñar políticas de seguridad viables y fiables que garanticen su protección.

Una vez más, la matemática muestra una solución. La matemática como ciencia es inherente al desarrollado de la humanidad, su penetración en todas las áreas del conocimiento es a veces muy directa como se percibe en el uso de las ecuaciones diferenciales y la probabilidad y otras veces indirecta como la teoría de números y el álgebra abstracta. Hacer uso de recursos matemáticos y aplicarlos a la solución de la problemática de nuestro país es lo que se sigue en este trabajo de grado, en particular, el estudio de las curvas elípticas en criptografía y su aplicación a la transferencia de información financiera en entornos Web.

La criptografía de curva elíptica (ECC por sus siglas en inglés) es un método criptográfico de clave pública (o asimétrico) en donde cada usuario que toma parte en el intercambio de datos posee un par de claves, una pública que es compartida a todos y otra privada que solo el propietario tiene acceso a ella. Estas dos claves, más un conjunto de constantes predefinidas (parámetros del dominio) y ciertas operaciones asociadas a ellas permiten el cifrado y el descifrado de los datos[2].

La ECC ha ganado gran interés dentro de la comunidad académica en los últimos años, principalmente porque presentan ventajas significativas frente a otros métodos de clave pública como RSA[3] y DSA[4] en cuanto a eficiencia, tamaño de claves y consumo de recursos computacionales, lo que las hace ideales para aplicaciones en ambientes Web.

---

1 Según datos de la DIJIN publicados al inicio de la *Semana de la Seguridad Informática*. Bogotá, D.C. Mayo de 2007.

2 En varios libros de criptografía se usan los términos *encriptar* y *desencriptar* derivados del verbo inglés *encrypt*, pero estos términos no están recogidos en el diccionario de la Real Academia Española por lo que se evitan y se consideran erróneos; en vez de ellos se usarán los verbos *cifrar* y *descifrar.*

3 RSA: Algoritmo de cifrado de clave pública desarrollado por Rivest, Shamir y Adelman.

4 Digital Signature Algoritm o Algoritmo de Firma Digital, es un estándar del gobierno de U.S para firmas digitales.

Como ejemplo, una clave ECC de 160 bits brinda una seguridad comparable con una clave RSA de 1024 bits (longitud de claves RSA más usadas hoy en día) y una clave ECC de 224 bits es considera al menos tan segura como una clave RSA de 2048 bits[5] (recomendadas para garantizar seguridad a largo plazo); además, para garantizar la seguridad de una clave RSA de 7680 bits (muy lenta y considerada de "paranoia militar"), se necesita una clave ECC de 384 bits y un tiempo de computo equivalente al RSA de 2048 bits.

[5] Fuente: Certicom Research, *Sec 1: Elliptic curve cryptography, Version 1.0*, Standards for Efficient Cryptography, http://www.secg.org, September 2000.

1 PRESENTACIÓN DEL PROYECTO

1.1 TÍTULO DEL PROYECTO

PROTOTIPO DE APLICACIÓN SEGURA PARA LA TRANSFERENCIA REMOTA DE DINERO EN ENTORNOS WEB, BASADO EN CRIPTOGRAFÍA DE CURVA ELÍPTICA.

1.2 OBJETIVO GENERAL

Diseñar e implementar un prototipo de aplicación segura basada en el cifrado de curva elíptica que permita el almacenamiento, organización, tratado y transferencia de información financiera en entornos Web donde los recursos de memoria, procesamiento y ancho de banda sean reducidos.

1.3 OBJETIVOS ESPECÍFICOS

- Investigar los diferentes tipos de curvas elípticas y evaluar aquellos que se hacen útiles para aplicaciones criptográficas.

- Implementar un prototipo de aplicación fiable con algoritmos computacionales de cifrado y descifrado de curva elíptica que permitan ofrecer confidencialidad de la información financiera en transferencias sobre redes inseguras.

- Establecer un sistema de firma digital basado en criptografía de curva elíptica que garantice a corto y mediano plazo, tanto la integridad de los datos como la autenticidad del emisor en cada transacción.

- Diseñar e implantar un modelo de datos para el almacenamiento de los datos relevantes al proceso de transferencia remota de dinero.

- Diseñar e implementar un sistema de copias de respaldo (*backup*) para la base de datos asociada al proceso de transferencia remota de dinero.

- Diseñar una política de seguridad en el sistema que permita el cifrado sobre la base de datos de campos y registros relevantes de modo que la confidencialidad no se vea comprometida aun en casos de ingreso a los datos por personal no autorizado.

- Implementar en software un criptosistema que integre estos algoritmos de cifrado de curva elíptica y que permita el tratado y la transferencia de información financiera en entornos Web con recursos limitados de memoria, procesamiento y ancho de banda.

1.4 ENTIDADES INTERESADAS EN EL PROYECTO

Todas aquellas entidades financieras cuya razón social les enfrenta con la necesidad de la protección de la información en sus operaciones transaccionales sobre redes inseguras.

Así mismo, cualquier entidad de cualquier índole en donde la integridad, la confidencialidad y la autenticidad de los datos hagan parte de sus objetivos y/o donde los recursos técnicos de máquinas de cómputo, ancho de banda y memoria sean limitados.

1.5 JUSTIFICACIÓN

Podemos medir el valor de la información que se desea transferir y esta evaluación demanda diferentes grados de seguridad. Son muchas y variadas las situaciones donde cabe garantizar la *privacidad*, la *integridad* y la *autenticidad* de los datos, la información financiera es por su naturaleza, valiosa; los datos que representan dinero desde un emisor a un receptor requieren garantizar estos tres elementos de la seguridad, lo que la hace una buena elección para este trabajo de grado como el objeto a proteger.

El concepto de criptografía ha experimentado un cambio radical desde el advenimiento de la informática y las grandes redes de computadores, los criptosistemas y los algoritmos empleados en estos aumentaron repentinamente y con ellos su complejidad. Todos estos algoritmos basan su seguridad en algún problema matemático que, por su gran magnitud, es casi imposible de resolver en la práctica. Estos problemas han sido tratados desde hace mucho tiempo por matemáticos como Euclides, Fermat o Euler, pero estos estudios no tenían como fin su aplicación en criptografía, pues en ese entonces no existía la criptografía de clave pública[6], ni siquiera existían las computadoras, por lo que estudiar algoritmos en donde los números involucrados eran "enormes" simplemente no era de interés.

En este sentido, podemos decir que los problemas matemáticos subyacentes a la criptografía son aun jóvenes, al menos desde que son concebidos como aplicables a la criptografía, esto se vislumbra cada vez que se descubren nuevas propiedades y algoritmos para "romper" o descifrar los criptosistemas en un tiempo menor al que se

[6] La criptografía de clave pública fue inventada en 1976 por W. Diffie y M. Hellman aunque algunos autores consideran que fue inventada por J. Ellis en 1969, otros argumentan que algunos conceptos de ésta ya habían sido propuestos por M. Wilkes en 1968 y manejados por G. Purdy en 1974. Pero en realidad el concepto fundamental de criptografía de clave pública fue desarrollado por Merkle en 1974, quien envió un articulo para publicar en *Communications of the ACM;* pero el articulo no fue publicado sino hasta 1978.

Fuente: Menezes, Van Oorschot, and Vanstone, *Handbook of applied cryptography,* CRC Press 1997.

había calculado inicialmente. La aparición en los últimos años de estos métodos y algoritmos de criptoanálisis (como el algoritmo ρ de Pollard, el algoritmo de Gaudry, etc) de tiempo sub-exponencial, ha creado la necesidad de agrandar el espacio de claves para "emparchar" estos criptosistemas.

Como una opción, Neil Koblitz y Victor Miller (independientemente) en 1985 propusieron el *Elliptic Curve Cryptosystem* (ECC), o Criptosistema de Curva Elíptica que en vez de utilizar números enteros como los símbolos del alfabeto de un mensaje a cifrar, usan puntos en un objeto matemático llamado *Curva Elíptica.*

Hasta el momento, no se conoce ataque alguno cuyo tiempo de ejecución esperado sea sub-exponencial para poder romper los ECC, esto hace que para obtener el mismo nivel de seguridad que brindan los otros sistemas, el espacio de claves de los ECC sea mucho más pequeño, lo que los hace una tecnología adecuada para utilizar en ambientes restringidos en recursos (memoria, costo, velocidad, ancho de banda, etc.). Por ejemplo, es notable el hecho de que una clave ECC de 512 bits brinda una seguridad comparable con una de 15.360 bits de un sistema DSA o RSA[7].

Los ECC han alcanzado un grado de madurez suficiente que los hace confiables, si bien recientemente se creo un estándar de los algoritmo basados en ECC en U.S, en Colombia, pese a que existe la ley 527 de 1999 que define y reglamenta el acceso y uso de los mensajes de datos, el uso de los ECC es muy escaso o quizás nulo.

1.6 DEFINICIÓN DEL PROBLEMA

Durante el lanzamiento de la Semana de la Seguridad[8], la DIJIN reveló que hasta abril del 2007 los casos denunciados por fraude electrónico en Colombia han costado más de $ 349.000 millones de pesos a personas naturales y cerca de 6,6 billones de pesos a empresas. Del hurto a personas naturales, más de $ 311.000 millones se han sustraído de cuentas bancarias.

La Dirección de Investigación Criminal señaló que solo en los primeros 4 meses del 2007 se habían registrado 310 casos por delitos informáticos superando el 71 por ciento de los 433 casos registrados en todo el 2006.

Más recientemente, se informó que durante todo el 2006 se hurtaron $ 3.900 millones por Internet, mientras que a septiembre de 2007 la cifra ya supera los $ 6.000 millones.

Estas cifras confirman que el fraude electrónico es uno de los principales problemas actuales en nuestro país, los casos de suplantación, alterabilidad, copia, hurto y perdida

---

[7] Fuente: Certicom Research, *Sec 1: Elliptic curve cryptography, Version 1.0*, Standards for Efficient Cryptography, http://www.secg.org, September 2000.

[8] Bogotá, D.C. 04 al 12 de Mayo del 2007.

parcial o total de la información han aumentado en más del 70% en los últimos años[9] según informes de la Unidad de Delitos Informáticos de la DIJIN, estos ataques tienen causas varias, entre ellas la no protección física de los datos, inexistencias de procedimientos de cifrado de la información y copias de respaldo y en ocasiones, ausencia casi total de una política de seguridad.

Aunque las entidades financieras prefieren ocultar estos desfalcos para evitar problemas de imagen, la problemática está hoy presente y latente.

Por otra parte, existen pequeñas y hasta medianas empresas que además de estar expuestas a estos mismos problemas, presentan recursos limitados en cuanto a equipos de procesamiento, memoria y ancho de banda.

En este sentido, este trabajo de grado pretende brindar una alternativa viable a seguir para entidades financieras grandes o pequeñas en donde exista la necesidad de aplicar una política de seguridad de cifrado eficiente, confiable y que involucre un sistema de copias de respaldo de la base de datos y/o en entidades donde los recursos de costos, equipos potentes y ancho de banda sean limitados.

1.7 IMPACTO ESPERADO

Se espera que este trabajo de investigación permita ayudar a solucionar la problemática existente de seguridad en las pequeñas, medianas y grandes entidades financieras que requieran nuevas políticas de seguridad.

Existen expectativas sobre su uso inicialmente en entidades de la región y posteriormente se espera que su uso trascienda al ámbito nacional como prototipo funcional de cifrado de curva elíptica y como una herramienta con una política de seguridad definida.

En el ámbito de investigación, este trabajo de grado ayuda a sentar los cimientos para futuros proyectos de ingeniería que busquen un uso mas generalizado de las curvas elípticas como objetos matemáticos ventajosos en criptografía, así como aquellos que busquen aportar soluciones a problemáticas de seguridad en los diferentes sistemas de información.

[9] Según el director de la oficina de delitos informáticos de la DIJÍN, Mayor Freddy Bautista, la cifra viene en acelerado crecimiento desde el 2002 y según estudios de ASOBANCARIA, el número de transacciones bancarias que se realizan en Colombia a aumentado cerca de 1.800 por ciento en los últimos 5 años.

2. PRELIMINARES

2.1 CONCEPTOS DE LA SEGURIDAD INFORMÁTICA.

Se detallan a continuación algunos conceptos básicos concernientes a seguridad informática que preparan al lector a comprender mejor todo el contenido de este documento, si bien no es una descripción exhaustiva, se ha intentado recopilar los conceptos más importantes de la seguridad informática.

2.1.1 Criptografía.

Pese a que el Diccionario de la Real Academia Española (RAE) la define como *"Arte de escribir con clave secreta o de un modo enigmático"*, la criptografía, (de los términos griegos "κryptos" (oculto) y "graphos" (escritura)), es más bien *el conjunto de técnicas y recursos lógicos utilizados para garantizar la seguridad de la información*.

Además de los recursos en software y hardware, la criptografía hace uso principalmente (y no exclusivamente) de los recursos matemáticos que aportan la Teoría de Números, el Álgebra Abstracta, Teoría de la Información y la Complejidad Algorítmica.

2.1.2 Criptoanálisis.

Las técnicas y recursos utilizados en la criptografía están orientados al cifrado, es decir, solo a la transformación de la información en códigos ilegibles de manera que solo puedan ser descifrados por quien(es) esté(n) autorizado(s) para observarla.

El estudio de métodos para obtener el sentido de una información cifrada sin poseer la clave que la descifra, o bien, el estudio de métodos para deducir dicha clave secreta y en general, el estudio de las debilidades de los algoritmos criptográficos se le conoce como *criptoanálisis.*

Cabe anotar que el criptoanálisis excluye los métodos que se alejan del estudio del algoritmo o del protocolo criptográfico empleado, como el soborno, la coerción, el hurto, etc.

Muchos autores coinciden en que los ataques de *fuerza bruta*[10] tampoco deben ser considerados como auténtico criptoanálisis y en general se denomina *ataque* a cualquier técnica que permita recuperar un mensaje cifrado empleando menos esfuerzo computacional que el que se usaría por fuerza bruta.

[10] Los ataques de fuerza bruta consisten en aplicar el algoritmo de descifrado empleando cada una de todas las claves posibles hasta que la salida *"tenga sentido"*.

2.1.3 Criptología.

Derivado del término inglés *cryptology*, y aun no recogido en el Diccionario RAE, el término criptología es empleado para agrupar las técnicas de criptografía y criptoanálisis y en general el estudio de los criptosistemas.

2.1.4 Criptosistemas[11].

Un criptosistema o sistema criptográfico se suele definir como la quíntupla (*M, C, K, E, D*) donde:

- *M* representa el conjunto de todos los mensajes sin cifrar (lo que se denomina texto claro).
- *C* representa el conjunto de todos los posibles mensajes cifrados, o criptogramas.
- *K* representa el conjunto de claves que se pueden emplear en el criptosistema.
- *E* es el conjunto de transformaciones de cifrado o familia de funciones que se aplica a cada elemento de M para obtener un elemento de C. Existe una transformación diferente E_k para cada valor posible de la clave k.
- *D es el conjunto de transformaciones de descifrado, análogo a E.*

2.1.5 Criptosistemas Simétricos.

Los criptosistemas simétricos (o de clave secreta compartida) son aquellos que emplean la misma clave *k* tanto para cifrar como para descifrar. La clave secreta *k* debe ser conocida por el receptor y el emisor por lo que su seguridad reside en mantener dicha clave bien asegurada.

En todo criptosistema simétrico se cumple la siguiente condición:

$$D_k(E_k(m)) = m.$$

es decir, que si tenemos un mensaje m, lo ciframos empleando la clave k y luego lo desciframos empleando la misma clave, obtenemos de nuevo el mensaje original m.

---

[11] Extraído de: Manuel J. Lucena López, *Criptografía y Seguridad en Computadores, 4a Edición.* Libro electrónico disponible en http://wwwdi.ujaen.es/mlucena/lcripto.html

La principal ventaja de los criptosistemas simétricos es que son computacionalmente menos costosos que los criptosistemas Asimétricos pero presentan ciertos inconvenientes:

i) Se necesitan *n(n-1)/2* claves secretas por cada *n* entidades involucradas en el sistema.[12]
ii) Para iniciar la comunicación, una nueva clave secreta generada no puede ser compartida por un canal inseguro entre emisor y receptor.
iii) No se pueden implementar las firmas digitales.

2.1.6 Criptosistemas Asimétricos.

Los criptosistemas Asimétricos (o de clave pública) son aquellos que emplean dos claves (*k*, *P*) por cada entidad involucrada en la comunicación, una clave es *pública* (*P*) y puede ser conocida por todos y otra clave es *privada* (*k*) y solo conocida por su propietario.

Si nos permitiéramos mirar el cifrado como una función matemática que transforma una variable del dominio (mensaje claro) en otro valor del rango (mensaje cifrado), se podría decir que la clave pública y la clave privada son funciones inversas entre sí, es decir, lo que la clave pública cifra, la clave privada lo descifra y viceversa[13].

$$D_k(E_P(m)) = m \quad \wedge \quad D_P(E_k(m)) = m$$

La seguridad de los criptosistemas asimétricos reside en la dificultad práctica de descubrir la clave privada a partir de la clave pública.

Su principal desventaja frente a los criptosistemas simétricos es que son computacionalmente más costosos, aunque presentas variadas ventajas frente a estos.

i) Solo se necesitan *n* pares de claves por cada *n* entidades involucradas en el sistema.
ii) Resuelven el problema del intercambio de claves, pues solo la clave pública necesita ser compartida.
iii) Permiten la implementación de firmas digitales.

En los criptosistemas asimétricos, el emisor cifra la información que desea transmitir con la clave pública del receptor, de modo que solo este último pueda descifrarlo con su clave privada, así, se garantiza la *confidencialidad* de la información. Para el firmado digital, el emisor hace uso de su clave privada de modo que cualquiera que tenga la clave pública de

---

[12] El lector podrá notar que la primer entidad necesita compartir (n-1) claves diferentes, la segunda (n-2) claves, la tercera (n-3) y así sucesivamente hasta la penúltima que solo comparte una (la última no necesita compartir, ya todos le han compartido su clave). Al sumar de *i=1* a *i=n* la expresión *(n - i),* obtenemos *n(n-1)/2* claves totales.

[13] En realidad, para muchos criptosistemas, estas claves son inversas dentro de un cuerpo finito pero no es así en los criptosistemas de curva elíptica donde, estrictamente hablando, la clave pública es un punto de la curva *E* y la clave privada es una constante k.

éste pueda verificar su firma, garantizando así la *autenticidad* y el *no repudio* por cuanto solo quien tiene la clave privada puede generar esa firma y la *integridad* por cuanto todo documento alterado haría fallar la verificación de la firma.

2.1.7 Criptosistemas Híbridos.

Para explotar tanto las ventajas de los criptosistemas simétricos como las ventajas de los criptosistemas asimétricos, se hace uso de la criptografía de clave pública para intercambiar de forma segura las claves secretas[14] para cifrado simétrico, de modo que después del intercambio de la clave, se utiliza la clave compartida para cifrado simétrico y la criptografía de clave pública solo para firmado digital.

Este tipo de criptosistemas es muy recomendable sobre todo cuando se necesita intercambiar mensajes muy extensos o grandes volúmenes de datos; en estos casos, se hace muy significativa la ganancia del rendimiento obtenido.

2.1.8 Seguridad Informática.

"*Ningún sistema es totalmente seguro*" es la premisa que parece derivarse en la práctica cuando se habla de seguridad informática, en este sentido, se define la seguridad informática como la *probabilidad* que tiene un sistema de proteger la integridad y confidencialidad de los objetos que lo forman.

En este documento debe entenderse la palabra *seguridad* como la *fiabilidad* del criptosistema, excluyéndose arbitrariamente el conjunto de factores físicos como sistemas de vigilancia de personal, incendios, desastres naturales, etc. ya que se salen de los objetivos de este trabajo y nos centraremos en los procedimientos lógicos que sigue el criptosistema. A continuación se describen los principales conceptos que debe garantizar un criptosistema *seguro.*

2.1.9 Confidencialidad.

La confidencialidad se refiere a la capacidad de mantener un documento electrónico inaccesible a todos, excepto a una o varias personas autorizadas para accederlo, es decir, que solo las personas autorizadas tengan acceso a la información.

2.1.10 Integridad.

La integridad se refiere a la seguridad de que la información no ha sido alterada, borrada o reordenada, bien durante el proceso de transmisión o en un tiempo posterior a ésta. La integridad garantiza que el documento recibido coincide con el documento emitido sin posibilidad alguna de cambio.

---

[14] Comúnmente, se genera una clave por cada nueva comunicación por lo que son llamadas *claves de sesión*.

2.1.11 Autenticidad.

La autenticidad se refiere a la capacidad de determinar si una o varias personas han establecido su reconocimiento y/o compromiso sobre el contenido del un documento electrónico. En otras palabras, es la capacidad de saber que la información ha provenido de una fuente fidedigna.

2.1.12 No Repudio.

El no repudio es la condición en la que quien envía el mensaje no puede negar la validez del resultado del proceso que se utilizó para autenticar la información[15], es decir, la imposibilidad de negar la autoría de un mensaje.

2.1.13 Disponibilidad.

La disponibilidad de la información se refiere a la seguridad que la información pueda ser recuperada en el momento que se necesite, esto es, evitar su pérdida o bloqueo, bien sea por ataque doloso, mala operación accidental o situaciones fortuitas o de fuerza mayor.[16]

También se define como la *aptitud* de un elemento para hallarse en estado de realizar una función requerida en un tiempo determinado o en cualquier instante de un intervalo dado, suponiendo que se facilitan, si es necesario, los recursos externos.[17]

2.1.14 Funciones Resumen

Las funciones resumen (o funciones *hash*) son algoritmos que representan de manera "casi" unívoca a un documento de tamaño cualquiera, estos algoritmos generan una secuencia de bits (de tamaño fijo) que identifican probabilísticamente un documento en particular.

Si *r* es una función resumen, *m* un mensaje cualquiera y *r(m)* el resultado de aplicar el algoritmo de la función resumen *r* al mensaje *m,* entonces *r* deben cumplir con las siguientes propiedades:

- *Unidireccionalidad:* conocido *r(m),* debe ser computacionalmente imposible encontrar *m.*
- *Compresión:* El número de bits de *r(m)* debe ser fijo para cualquier tamaño del mensaje *m.*
- *Facilidad de Cálculo:* debe ser eficiente calcular *r(m)* para cualquier mensaje *m*.

---

15 Tomado de: www.hidcorp.com/espanol/page.php

16 Tomado de: www.bradanovic.cl/pcasual/ayuda3.html

17 Tomado de: www.euromaya.com/glosario/D_GLOSARIO.htm

- *Difusión:* el resumen *r(m)* debe ser una función compleja de todos los bits del mensaje *m*, esto es, si se modifica un solo bit del mensaje *m*, debe esperarse que cambie en promedio el 50% de todos los bits de *r(m)*.

- *Resistencia débil a colisiones:* conocido m, debe ser computacionalmente imposible encontrar otro mensaje m' tal que r(m) = r(m').

- *Resistencia fuerte a colisiones:* debe ser computacionalmente *muy difícil* encontrar dos mensajes cualesquiera m y m' tales que r(m) = r(m').

Como se puede notar, para un buen algoritmo de funciones resumen, las propiedades de *resistencia a colisiones* están relacionadas con el tamaño de *r(m)*. Pero ¿Qué tan pequeño puede ser el tamaño de *r(m)?.*

Para responder a esa pregunta, debemos explicar la erróneamente llamada *"paradoja de cumpleaños"*; digo *"erróneamente llamada"*, porque en realidad no es una paradoja.

¿Cuál es la cantidad *n* de personas que deben estar en una habitación para que la probabilidad P de que el cumpleaños de una de ellas sea el mismo día que el mío supere el 50%?[18]. Sabemos que cuando n = 1, P = 1/365. Cuando n = 2, la probabilidad de que ningún cumpleaños coincida con el nuestro es el producto de la probabilidad de que no coincida el primero, por la probabilidad de que no coincida el segundo, luego:

$$P = 1 - (364/365)*(364/365)$$

En el caso general, $P = 1 - (364/365)^n$

Para que P > 0,5, n debe ser al menos igual a 253. Sin embargo, ¿Cuál sería la cantidad de gente necesaria para que la probabilidad Q de que dos personas cualesquiera cumplan años el mismo día supere el 50%? Las dos primeras personas (o sea, cuando $n = 2$) tienen una probabilidad 364/365 de no compartir el cumpleaños; una tercera, supuesto que las dos primeras no lo comparten, tiene una probabilidad 363/365 de no compartirlo con las otras dos, por lo que tenemos (364/365) * (363/365) , y así sucesivamente. En el caso general nos queda:

$$Q = 1 - ((364*363*...*(365 - n + 1)) / (365^{n-1})) \quad \text{con } n > 1.$$

Calculando, vemos que Q > 0,5 si $n > 22$, una cantidad *"sorprendentemente pequeña".*

Con esto se vislumbra que las dos propiedades de *resistencia a colisiones* de las funciones resúmenes son muy distintas; aunque resulte muy difícil para un atacante calcular un m' dado un m tal que r(m) = r(m'), es computacionalmente menos costoso generar muchos valores aleatorios de mensajes en *M* y posteriormente buscar una pareja cualquiera (m, m') tal que r(m) = r(m').

[18] Supondremos años de 365 días.

En general, para llevar a cabo un ataque de cumpleaños para un r(m) de *n* bits, el atacante debe probar aproximadamente $1{,}2 \cdot \sqrt{n}$ mensajes aleatorios para que la probabilidad de encontrar una pareja (m, m') tal que r(m) = r(m') sea del 50%. Como ejemplo, para un *r (m)* de 64 bits, dado un *m* necesitaríamos 2^{64} mensajes para obtener el m', pero bastaría con generar aproximadamente 2^{32} mensajes para que aparecieran dos (m, m') tal que r(m) = r(m'). Con una computadora que realice un millón de operaciones por segundo, el primer caso le tomaría más de 584.000 años, mientras que el segundo ataque le tomaría aproximadamente solo una hora y 12 minutos.

Con el poder computacional actual, se recomiendan funciones resúmenes de al menos 128 bits.

Como ejemplo de funciones resúmenes más empleadas, están el algoritmo MD5 (*Message-Digest Algorithm 5)* de 128 bits desarrollado por R. Rivest basado en el MD4 (del mismo autor); el SHA-1 (S*ecure Hash Algorithm*) basado en MD5, es un algoritmo de 160 bits publicado por el NIST[19] y la familia SHA-2 compuesta por los algoritmos SHA-224, SHA-256, SHA-384, y SHA-512 con longitudes explícitas en sus nombres.

En el 2004 un grupo de criptoanalistas Chinos de la Universidad Shandong, lograron romper el algoritmo MD5 y demostraron ser capaces de romper el SHA-1 unas 2000 veces más rápido que un ataque de fuerza bruta, por lo que a futuro, se prevé que los algoritmos derivados del MD5 tienen sus "*días contados*".[20]

Algunos expertos sugieren buscar reemplazos para una nueva función estandarizada, los algoritmos que se mencionan como reemplazo son *Tiger* y *WHIRLPOOL.* En este trabajo de grado se hace uso del algoritmo *Tiger* de 192 bits como función resumen en el cifrado y firmado de curva elíptica.

2.1.15 Firmas Digitales

La firma digital de un documento *m*, básicamente es el cifrado con la clave privada de el valor *r(m)*, junto con otros atributos como la fecha y hora en que se firma. Por supuesto esta no es una definición precisa, pero es apropiada para entender el principio básico del firmado digital. Este procedimiento es simple, pero de gran importancia: garantiza la *integridad* del documento *m* puesto que cualquier cambio en *m* por un atacante o por perdida o modificación de bits al transmitirlo hace que al comparar el resumen en la firma (obtenido al descifrarla) con un nuevo resumen del documento alterado (recalculado por el receptor) difieran, si esto ocurre, el documento debe ser rechazado, en otro caso el documento está *íntegro*. También garantiza la *autenticidad* y el *no repudio* dado que el

---

19 *National Institute of Standards and Technology* (NIST).

20 Para una descripción de estos ataques visite los siguientes enlaces:
http://en.epochtimes.com/news/7-1-11/50336.html
http://csrc.nist.gov/hash_standards_comments.pdf
http://www.schneier.com/blog/archives/2005/02/sha1_broken.html

valor numérico de la firma[21] del remitente solo puede ser generado por quien posea su clave privada, en principio, él mismo.

En nuestro país, la ley 527 de 1999 define la firma digital como "*un valor numérico que se adhiere a un mensaje de datos y que, utilizando un procedimiento matemático conocido, vinculado a la clave del iniciador y al texto del mensaje permite determinar que este valor se ha obtenido exclusivamente con la clave del iniciador y que el mensaje inicial no ha sido modificado después de efectuada la transformación*".

Finalmente, cabe mencionar algunas características del firmado digital:

- Firmar un documento digitalmente **no** implica cifrarlo, es decir, una firma digital puede ir anexa a un documento sin que éste esté necesariamente cifrado, por ejemplo si quisiésemos hacerlo público.

- La firma varía según el mensaje firmado y la clave firmante, es decir, el valor numérico de la firma digital cambia cuando firmamos diferentes mensajes.

- En la práctica, el valor numérico de la firma digital varía aun cuando el mismo documento es firmado con la misma clave privada; esto es así porque la fecha y hora de firmado hacen parte de la firma y es poco probable que el documento sea firmado dos veces en el mismo instante de tiempo.

- Con un algoritmo fiable, las firmas digitales son más difíciles de falsificar que las firmas manuscritas por lo que se consideran más seguras.

2.1.16 Autoridades Certificadoras y Certificados Digitales.

Ya hemos visto que los criptosistemas de clave pública solucionan el problema del intercambio de claves inherente a los criptosistemas simétricos puesto que solo es necesario transmitir la clave pública por medio del canal inseguro. Pero, surge entonces otro inconveniente, ¿cómo estamos seguros de que una nueva clave publicada pertenece a quien dice pertenecer?; si en una comunicación siempre admitiéramos que una clave pública recibida pertenece a quien dice ser, estaríamos expuestos a que un atacante nos envíe una clave pública haciéndose pasar por otra persona, el atacante recibiría y podría leer entonces los mensajes destinados a la persona suplantada, lo cual es un problema grave en la seguridad.

Si pensamos en el intercambio personal de claves públicas cada vez que se involucre una nueva entidad, volveríamos al mismo problema de los criptosistemas simétricos. Para solucionar este inconveniente, surge la necesidad de un tercero de confianza, a quien conozcamos de antemano y de quien estemos seguros poseer su clave pública; este

[21] Entiéndase *valor numérico de la firma* como la secuencia de bits que se produce al firmar el documento.

tercero es llamado una *Autoridad Certificadora* (AC) y se encargará de *certificar* que cualquier nueva clave pública generada, efectivamente pertenezca a quien dice pertenecer. Pero, ¿Cómo la certifica?, El poseedor de la nueva clave debe entregar su clave personalmente a la AC[22], la AC firma la nueva clave pública y forma un documento que además de contener dicha firma, contenga la clave pública firmada, la clave pública firmante, la fecha en que fue certificada (firmada) la fecha en que caduca la firma (si aplica) y otros datos relevantes; a este documento se le conoce como *Certificado Digital.*

De este modo, la nueva entidad no necesita presentarse personalmente ante las otras entidades, es suficiente con demostrar a través del Certificado Digital que su clave está firmada por la Autoridad Certificadora en quien las otras entidades confían.

Las ACs tiene otras funciones como revocar certificados digitales cuando estos caduquen o cuando se sepa que la clave privada asociada la clave pública firmada haya sido comprometida, deben tener listas siempre disponibles y actualizadas de todos los certificados revocados, entre otras funciones. En cada país están reglamentadas y vigiladas jurídicamente, y siguen una estandarización mundial para la generación de certificados.

En Colombia, la ley 527 de 1999 y el decreto 1747 del 11 de septiembre del 2000 reglamentan y le dan validez jurídica a las ACs, y son supervisadas por la *Superintendencia de Industria y Comercio,* sin embargo, hasta ahora la única Autoridad Certificadora en Colombia es Certicamara S.A.[23] pero ésta, no implementa sistemas de certificación con curvas elípticas. Se espera que a futuro podamos contar en el país con más Autoridades Certificadoras y que en lo posible, hagan uso de las ventajas que ofrecen los esquemas de curva elíptica.

En cuanto a los Certificados Digitales, el estándar seguido es el X.509 v3, publicado oficialmente en 1988; actualmente un certificado X.509 maneja la siguiente estructura:

- Certificado
    - Versión
    - Número de serie
    - ID del algoritmo
    - Emisor
    - Validez
        - No antes de
        - No después de
    - Tema
    - Tema información de clave pública
        - Algoritmo de clave pública
        - Tema clave pública

[22] La ley 527 de 1999 permite a las *Autoridades Certificadoras* delegar el procedimiento de presentación personal a un tercero llamado *Autoridad de Registro*. Esta misma Ley permite a los Notarios y Cónsules actuar como tales.

[23] http://www.certicamara.com.co

    - Identificador único de emisor (opcional)
    - Identificador único de tema (opcional)
    - Extensiones (opcional)
        - ...
- Algoritmo de certificado de firma
- Certificado de firma.

Para agrupar al conjunto de Autoridades Certificadoras, Certificados Digitales, políticas de seguridad, software, hardware, listas de revocación, suscriptores de certificados, tecnología y procedimientos que garantizan las operaciones criptográficas se utiliza el término genérico *Infraestructura de Clave Pública (PKI por sus siglas en Inglés).*

2.2 Conceptos de Teoría de Números y Álgebra Abstracta.

El lector familiarizado con los conceptos básicos de congruencia, aritmética modular, teoría de grupos y tests de primalidad podrá prescindir de esta sección.

2.2.1 División Euclídea.

Para cualesquiera enteros *a* y *b*, con *b* no nulo, existen enteros únicos *c* y *r*, llamados *cociente* y *residuo* respectivamente, tales que:

$$a = dc + r \text{ y } 0 \leq r < |d|.$$

2.2.2 Congruencia.

Sean *a, b, p* números naturales, se dice que *a* es *congruente con b módulo p,* y se denota

$$a \equiv b \text{ (mód } p)$$

si existe algún entero *k tal que:*

$$a = b + kp.$$

es decir, si los residuos de dividir *a* por *p* y *b* por *p* son iguales.

Ejemplo: $37 \equiv 5$ (mód 8), porque $37 = 5 + 4*8$.

2.2.3 Pequeño Teorema de Fermat.

Si *p* es un número primo, entonces, para todo número natural *a,*

$$a^p \equiv a \text{ (mód } p).$$

Ejemplo: Si el MCD {a,p} = 1, se tiene $a^{p-1} \equiv 1$ (mód *p).*

2.2.4 Función φ de Euler.

Si *n* es un entero positivo, entonces φ*(n)* se define como el número de enteros positivos menores o iguales a *n* y coprimos con *n*.

Ejemplo: Si *p* es un número primo, φ*(p) = p – 1*

2.2.5 Teorema de Euler. *(Generalización del Pequeño Teorema de Fermat)*

Sean *a, n* enteros positivos, con a y n coprimos, entonces

$$a^{\varphi(n)} \equiv 1 \ (\text{mód } n).$$

2.2.6 Operación Binaria.

Una operación binaria * sobre un conjunto *S* es un mapeo de *S*x*S* a *S*. Esto es, * es una regla que asigna a cada par ordenado de elementos de *S* un elemento de *S*.

Ejemplo: La operación + en **Z**. Dado a,b ∈ **Z**, (a+b) ∈ **Z**

2.2.7 Grupos.

Un *grupo* (*G, **) consiste de un conjunto *G* con una operación binaria * sobre *G* satisfaciendo los siguientes tres axiomas.

(i) La operación del grupo es *asociativa*. Esto es, $a * (b * c) = (a * b) * c$ para todo *a, b* ∈ *G*.

(ii) Existe un elemento 1 ∈ *G*, llamado el *elemento identidad*, tal que $a * 1 = 1 * a = a$ para todo *a* ∈ *G*.

(iii) Para cada *a* ∈ *G* existe un elemento $a^{-1} \in G$, llamado el *inverso* de *a*, tal que $a * a^{-1} = a^{-1} * a = 1$.

Un grupo *G* es *abeliano* (o *conmutativo*) si además,
(i) Para todo *a, b* ∈ *G, a * b = b * a*.

Ejemplo: Sea $\mathbf{Z_n}=\{0,1,2,...,n-1\}$, entonces ($\mathbf{Z_n}$, +) es un grupo, puesto que la suma módulo *n* entre enteros es conmutativa, existe el número 0 que es la identidad, y el inverso de *a* es *–a* (mód *n*). También si $\mathbf{Z}^*_p=\{1,2,...,p-1\}$, entonces ($\mathbf{Z}^*_p$, ***) con *p* primo es un grupo con la operación de multiplicación módulo *p*, donde la identidad es el número 1, y el inverso de *a* siempre existe dado que *p* es primo.

2.2.8 Subgrupos.

Un subconjunto no vacío *H* de un grupo *G* es un *subgrupo* de *G* si *H* es en si mismo un grupo con respecto a la operación de *G*. Si *H* es un subgrupo de *G* y $H \neq G$, entonces *H* es llamado un subgrupo *propio* de *G*.

Ejemplo Si consideramos el grupo $\mathbf{Z}^*_{19}$, el conjunto $H = \{1, 7, 11\}$ es un subgrupo de $\mathbf{Z}^*_{19}$, ya que todas las combinaciones de cálculos módulo 19 con elementos de *H* da resultados en *H*: 1*1 = 1; 1*7 = 7; 1*11 = 11; 7*7 = 11; 7*11 = 1; 11*11 = 7.

2.2.9 Grupo Cíclico.

Un grupo $(G, *)$ es *cíclico* si existe un elemento $\alpha \in G$ tal que para cada $b \in G$ existe un número entero *i* con $b = \alpha^i$. Tal elemento α es llamado un *generador* de $(G, *)$.

Ejemplo: El conjunto $\mathbf{Z}^*_p$ con *p* primo es un grupo cíclico ya que por el *Pequeño Teorema de Fermat*, $a^{p-1} = 1$ (mód *p*), multiplicando por *a* cada miembro queda $a^p = a$ (mód *p*).

De aquí en adelante, omitiremos la operación del grupo, si ésta es irrelevante o sobreentendida, de acuerdo al contexto en que se encuentre.

2.2.10 Subgrupo Generado.

Si G es un grupo y $a \in G$, entonces el conjunto de todas las potencias de a forma un subgrupo cíclico de G, llamado el subgrupo generado por a, y denotado por <a>.

Ejemplo: El conjunto $H = \{1, 7, 11\}$ es un subgrupo cíclico de $\mathbf{Z}^*_{19}$. Además $H = \langle 7 \rangle = \langle 11 \rangle$.

2.2.11 Orden de un Elemento del Grupo.

Sea *G* un grupo y $a \in G$. El *orden* de *a* es definido como el menor entero positivo *t* tal que $a^t = 1$, si es que tal entero existe. Si tal *t* no existe, entonces el orden de *a* es definido como *1*.

Ejemplo: En el conjunto $\mathbf{Z}^*_{16}$,el orden del número 3 es $t = 4$, ya que: 3^1 mód 16 = 3, 3^2 mód 16 = 9, 3^3 mód 16 = 11, 3^4 mód 16 = 1.

2.2.12 Grupos Finitos.

Un grupo *G* es *finito* si $|G|$ es finito. El *orden* de *G* es el número de elementos de *G*, si éste es finito.

Ejemplo: $\mathbf{Z}_n$ tiene orden *n*, mientras que $\mathbf{Z}^*_p$ tiene orden $p - 1$ si *p* es primo.

2.2.13 Orden del Subgrupo Generado.

Sea G un grupo, sea a ∈ *G un elemento de orden finito t. Entonces* |<*a*>| = *t (el orden de a es la cantidad de elementos de <a>.)*

Ejemplo: En $\mathbf{Z}^*_{19}$, el orden del elemento 7 es |<7>| = 3.

2.2.14 Teorema de Lagrange.

Si G es un grupo finito y H es un subgrupo de G, entonces |*H*| *divide a* |*G*|*. Por lo tanto, si a* ∈ *G, el orden de a divide a* |*G*|*.*

Ejemplo: $\mathbf{Z}^*_{19}$ es un grupo de orden $18 = 2*3^2$, luego si $a \in \mathbf{Z}^*_{19}$, entonces |<*a*>| ∈ {1, 2, 3, 6, 9, 18}.

2.2.15 Anillos.

Un *anillo* (*R,* +, *) consiste de un conjunto *R* con dos operaciones binarias denotadas arbitrariamente + (adición) y * (multiplicación) sobre *R*, satisfaciendo los siguientes axiomas.

(i) (*R,* +) es un grupo conmutativo con identidad denotada 0.

(ii) La operación * es asociativa. Esto es, $a * (b * c) = (a * b) * c$ para todo *a, b, c* ∈ *R*.

(iii) Existe una identidad multiplicativa denotada 1, con 1 ≠ 0, tal que $1 * a = a * 1 = a$ para todo *a* ∈ *R*.

(iv) La operación * es *distributiva* sobre +. Esto es, $a * (b+c) = (a*b) + (a*c)$ y $(b+c) * a = (b*a) + (c*a)$ para todo *a, b, c* ∈ *R*.

El anillo es un *anillo conmutativo* si $a * b = b * a$ para todo *a, b* ∈ *R*.

Ejemplo: $\mathbf{Z}_n$ con la suma y multiplicación módulo *n* es un anillo conmutativo.

2.2.16 Elemento Unidad.

Un elemento *a* de un anillo *R* es llamado una *unidad* o un *elemento invertible* si existe un elemento *b* ∈ *R* tal que $a * b = 1$.

Ejemplo: En $\mathbf{Z}_n$, existe el inverso multiplicativo de *a* ≠ 0 si el máximo común divisor MCD(*a, n*) = 1.

2.2.17 Campos.

Un *campo* ***F*** es un anillo conmutativo en el cual todos los elementos no nulos tienen inversos multiplicativos.

Ejemplo: Si p es primo, $\mathbf{Z}_p$ es un campo.

2.2.18 Característica de un Campo *F*

La *característica* de un campo ***F*** es el menor entero positivo p tal que:

$$p \cdot 1 = 0$$

donde $p \cdot 1$ quiere decir 1 + 1 + ... + 1, p-veces. Si no existe tal p, la característica del campo es cero. Se puede demostrar que si p existe, entonces p es un número primo.

2.2.19 Subcampos.

Un subconjunto F de un campo E es un *subcampo* de E si F es en si mismo un campo con respecto a las operaciones de E. Si este es el caso, E se dice que es un *campo extensión* de F.

2.2.20 Campos Finitos.

Un *campo finito* es un campo F el cual contiene un número finito de elementos. El *orden* de F es el número de elementos de F.

Ejemplo: Si p es primo, $\mathbf{Z}_p$ es un campo finito de orden p.

2.2.21 Logaritmo Discreto.

Sea G un grupo cíclico finito de orden n. Sea α un generador de G, y sea $\beta \in G$. El *logaritmo discreto de* β *a la base* α, denotado $\log_\alpha \beta$, es el único entero x, $0 \leq x \leq n - 1$, tal que $\beta = \alpha^x$.

2.2.22 Existencia y Unicidad de los Campos Finitos.

(i) *Si F es un campo finito, entonces F contiene p^m elementos para algún número primo p y entero $m \geq 1$.*

(ii) *Para todo orden de potencia prima p^m, existe un único campo finito (salvo isomorfismos) de orden p^m. Este campo es denotado por $F_{p\hat{}m}$, o a veces por $GF(p^m)$.*[24]

[24] La *F* viene de la palabra en inglés field (campo). GF viene de Galois Field.

En otras palabras, dos campos del mismo orden son *isomorfos*, es decir, son estructuralmente el mismo, aunque la representación de sus elementos del campo pudieran ser diferentes. Note que si p es un número primo entonces $\mathbf{Z}_p$ es un campo, y por lo tanto todo campo de orden p es isomorfo a $\mathbf{Z}_p$. A menos que se indique explícitamente, el campo finito F_p será identificado con $\mathbf{Z}_p$.

2.2.23 Subcampo de un Campo Finito.

Sea F_q *un campo finito de orden* $q = p^m$. *Entonces todo subcampo de* F_q *tiene orden* p^n, *para algún n que es un divisor positivo de m. Y viceversa, si n es un divisor positivo de m, entonces existe exactamente un subcampo de* F_q *de orden* p^n; *un elemento* $a \in F_q$ *está en el subcampo* F_{p^n} *si y solo si* $a^{p^n} = a$.

Ejemplo: Ya que $1|m$, donde p es un número primo y $m \geq 1$, entonces F_{p^m} contiene una copia de $\mathbf{Z}_p$ como un subcampo. Por lo tanto F_{p^m} puede ser visto como un campo extensión de $\mathbf{Z}_p$ de grado m.

2.2.24 Grupo Multiplicativo.

Los elementos no nulos de F_q forman un grupo bajo la operación de multiplicación llamado el *grupo multiplicativo* de F_q, denotado por F^*_q.

Ejemplo: Si p es primo, el grupo multiplicativo de $\mathbf{Z}_p$ es $\mathbf{Z}^*_p = \mathbf{Z}_p - \{0\}$.

2.2.25 Teorema del Grupo Cíclico.

Si F^*_q *es un grupo cíclico de orden* $q - 1$. *Luego,* $a^q = a$ *para todo* $a \in F^*_q$

Ejemplo: En $\mathbf{Z}^*_q$ con $q = p^m$, para $m = 1$ tenemos el Pequeño Teorema de Fermat.

2.2.26 Anillo Polinomial.

Si R es un anillo conmutativo, el *anillo polinomial* $R[x]$ es el anillo formado por el conjunto de todos los polinomios en la indeterminada x teniendo coeficientes en R. Las dos operaciones son la suma y multiplicación estándar de polinomios, con coeficientes aritméticos que operan en el anillo R.

Ejemplo: $f(x) = x^3 + x + 1$ y $g(x) = x^2 + x$ son elementos del anillo polinomial $\mathbf{Z}_2[x]$, donde los coeficientes pueden ser 0 o 1, y las operaciones son módulo 2. $f(x) + g(x) = x^3 + x^2 + 1$ y $f(x)*g(x) = x^5 + x^4 + x^3 + x$.

Sea $\mathbf{Z}_p$ el campo finito de orden p, con p primo. La Teoría de Números se puede llevar al anillo polinomial $\mathbf{Z}_p[x]$, y más generalmente al anillo polinomial $F[x]$, donde F es cualquier campo. En otras palabras, los polinomios se pueden factorizar, dividir, calcular el MCD,

calcular el algoritmo de Euclídes, definir congruencias, etc. de manera similar a los elementos de $\mathbf{Z}_p$.

2.2.27 Polinomios Irreducibles.

Sea $f(x) \in F[x]$ un polinomio de grado al menos 1. Entonces $f(x)$ se dice que es *irreducible sobre F* si este no puede ser escrito como el producto de dos polinomios en $F[x]$, cada uno de grado positivo.

Ejemplo: $f(x) = x^2 + x + 1$ es irreducible sobre $\mathbf{Z}_2$.

2.2.28 Número de Términos de un Polinomio Irreducible.

Un polinomio $f(x)$ *irreducible sobre* F_{2^m} *tiene un número impar de términos.*

Demostración. Esto es así puesto que si tuviera un número par de términos, entonces debe ocurrir alguna de dos posibilidades:

1. Si $a_0 = 0$, entonces $f(x) = x^{k1}+x^{k2}+ \ldots +x^{k2t}$. Luego $f(0) = 0+0+ \ldots +0 = 0$, por lo tanto $x|f(x)$.
2. Si $a_0 = 1$, entonces $f(x) = x^{k1} + x^{k2} + \ldots + x^{k2t-1} + 1$. Luego $f(1) =1 + 1 + \ldots + 1 + 1 = 0$ puesto que hay un número par de 1's, por lo tanto $(x + 1)|f(x)$.

Entonces debe ser que un polinomio irreducible sobre F_{2^m} tiene un número impar de términos.

Como se mencionó, en $F[x]$ se pueden definir congruencias módulo un polinomio $f(x)$. Al igual que en $\mathbf{Z}_n$, esta congruencia define una relación de equivalencia en $F[x]$, y así $F[x]$ queda particionado en clases de equivalencia.

2.2.29 Clases de Equivalencia en Anillos Polinomiales.

$F[x]/(f(x))$ denota el conjunto de *clases de equivalencia* de los polinomios en $F[x]$ de grado menor que n = grado$f(x)$. La suma y multiplicación se ejecutan módulo $f(x)$.

Teorema: $F[x]/(f(x))$ es un anillo conmutativo.

Teorema: Si $f(x)$ *es irreducible sobre F, entonces* $F[x]/(f(x))$ *es un campo.*

Ejemplo: $f(x) = x^2 + x + 1$ es irreducible sobre $\mathbf{Z}_2$. Luego $\mathbf{Z}_2[x]/(f(x))$ es un campo, donde las operaciones son módulo $f(x)$.

Una representación comúnmente usada para los elementos de un campo finito F_q, donde $q = p^m$ y p es un número primo, es una *representación de base polinomial*. Si $m = 1$, entonces F_q es simplemente $\mathbf{Z}_p$ y la aritmética es ejecutada módulo p.

2.2.30 Representación en Base Polinomial.

Para cada $m \geq 1$, existe un polinomio monico irreducible de grado m sobre $\mathbf{Z}_p$. Por lo tanto, todo campo finito tiene una representación de base polinomial.

Entonces, los elementos de un campo finito $F_{p\hat{}m}$ serán representados por polinomios en $\mathbf{Z}_p[x]$ de grado $< m$, y con coeficientes en $\mathbf{Z}_p$.

Ejemplo: El campo finito $F_{2\hat{}4}$ se puede representar como el conjunto de polinomios sobre F_2 de grado < 4. Esto es,

$$F_{2\hat{}4} = \{a_3x^3 + a_2x^2 + a_1x + a_0 \mid a_i \in \{0, 1\}\}$$

Por conveniencia, el polinomio $a_3x^3 + a_2x^2 + a_1x + a_0$ es representado por el vector $(a_3a_2a_1a_0)$ de longitud 4, y así

$$F_{2\hat{}4} = \{(a_3a_2a_1a_0) \mid a_i \in \{0, 1\}\}$$

Lo cual es muy adecuado para representación en una computadora. También el polinomio $f(x) = x^4 + x + 1$ es irreducible sobre $\mathbf{Z}_2$, entonces las operaciones se hacen módulo $f(x)$.

A veces es conveniente utilizar un polinomio primitivo para definir un campo finito:

2.2.31 Polinomios Primitivos.

Un polinomio irreducible $f(x) \in \mathbf{Z}_p[x]$ de grado m es llamado un *polinomio primitivo* si x es un generador de $F^*_{p\hat{}m}$, el grupo multiplicativo de todos los elementos no nulos en $F_{p\hat{}m}$.

Ejemplo: $f(x) = x^4 + x + 1$ es un polinomio primitivo, o equivalentemente, el elemento $x = (0010)$ del campo es un generador de $F^*_{2\hat{}4}$. Es decir que $F^*_{2\hat{}4} = \{x^i \bmod f(x) \mid 0 \leq i < 2^4 - 1\}$.

2.2.32 Problema de la Primalidad.

Determinar si un natural n es un número primo es conocido como el *problema de la primalidad* y es un aspecto muy importante de la criptografía actual, la seguridad de muchos criptosistemas asimétricos actuales está en función de que se tenga certeza de haber generado números primos "enormes". Como ejemplo, la generación de claves del esquema RSA depende de la intratabilidad de *Problema de la Factorización Entera (PFE)*[25] que involucra números primos.

Para determinar si un número es primo, se han ideado varios algoritmos, unos son determinísticos, llamados *pruebas de primalidad,* que al ser ejecutados determinan con certeza si el número dado es primo o no, y otros probabilísticos, llamados *test de primalidad* que, dado un número compuesto, existe muy baja probabilidad de que el

[25] El *Problema de la Factorización Entera (PFE)* consiste en encontrar un divisor no trivial de un número compuesto formado generalmente como el producto de dos primos de muchas cifras elegidos al azar.

algoritmo lo identifique como número primo (para algunos de estos test, la probabilidad decrece por cada iteración ejecutada en el algoritmo).

Mas formalmente, "un *test de primalidad* es un algoritmo que, dado un número de entrada *n*, no consigue verificar la hipótesis de un teorema cuya conclusión es que *n* es compuesto" y "una *prueba de primalidad* (o test verdadero de primalidad) es un algoritmo determinístico que, dado un número de entrada *n*, verifica la hipótesis de un teorema cuya conclusión es que *n* es primo"[26].

Desafortunadamente, las pruebas de primalidad son computacionalmente mucho más costosas que los tests de primalidad, y además, la probabilidad de "equivocación" de los tests de primalidad son significativamente bajas, por lo que, en la practica se usan estos últimos.

Algunos ejemplos de test y pruebas de primalidad son:

- *Test de Fermat.* Basado en el Pequeño Teorema de Fermat, más concretamente en el recíproco lógico de este teorema, y parte del hecho de que si *p* es un número compuesto, entonces $a^{(p-1)}$ es poco probable que sea congruente con 1 módulo *p* para un valor arbitrario de *a*. El algoritmo consiste entonces en, dado el parámetro de entrada *p*, probar valores de *a* desde 2 hasta *p* - 2 y evaluar la expresión $a^{(p-1)}$ (mód *p*), si el residuo es diferente de 1 para algún *a,* entonces *p* es compuesto y el algoritmo "para", en otro caso, el número *p* es probablemente primo.

  La ventaja de éste algoritmo es que con pocos valores *a* se alcanza una muy baja probabilidad de obtener residuo 1 siendo *p* compuesto; la gran desventaja es que existen números que siendo compuestos pasan el test para todos los valores de *a,* estos son los llamados *números de Carmichael.*

- Test de *Solovay-Strassen* (SS). Basado en el criterio de Euler, este test tiene la particularidad de que en cada iteración la probabilidad de que un número compuesto pase como primo es ½ , o equivalentemente, la probabilidad de que un número compuesto pase *n* iteraciones es $½^n$.

- *Test de Miller-Rabin.* Basado en la hipótesis generalizada de Riemann (aun no demostrada), este test mejora (baja) la probabilidad de que un número compuesto sea declarado primo a menos de ¼ por cada iteración (menos de $¼^n$ por cada n iteraciones) y además, al usar exponenciación binaria es mucho más eficiente que el test SS por lo que se ha convertido en el test más implementado en el desarrollo de criptosistemas.

[26] Extraído de: http://es.wikipedia.org

- *Prueba de primalidad de Fibonacci.* Es un algoritmo determinista muy simple y extremadamente ineficiente, consiste en verificar que ningún otro número primo menor a $\sqrt{p}$ divide a *p.*

- *Prueba LKL (o prueba de Lucas-Kraitchik-Lehmer*). Es una prueba que solo se puede aplicar a los números de Mersenne[27]; debido a esta limitación, es utilizada principalmente en la generación de los llamados *primos de Mersenne*.

- *PRIMES is in P.* Es el nombre de un reciente artículo publicado por los académicos hindúes Manindra Agrawal, Neeraj Kayal y Nitin Saxena, donde describen un algoritmo determinista incondicionado de tiempo polinomial que determina si un número dado es primo o compuesto.[28]

2.3 Conceptos de Teoría de la Complejidad Algorítmica.

2.3.1 Principio de Invarianza.

Si dos implementaciones del mismo algoritmo consumen $t_1(n)$ y $t_2(n)$ segundos respectivamente, siendo *n* el tamaño de los datos de entrada, entonces existe una constante positiva *c* tal que $t_1(n) \leq c \cdot t_2(n)$, para algún *n* suficientemente grande.

Esto quiere decir que, aunque se usen computadoras más poderosas que otras, la evolución del tiempo de ejecución de un algoritmo en función del tamaño del problema permanecerá constante.

2.3.2 Cota Superior Asintótica O*(g(x))* .

La cota superior asintótica O(*g(x)*) es el conjunto de funciones acotadas superiormente por $c \cdot g(x)$, a partir de un valor $x_0 > 0$, para algún $c > 0$.

Formalmente,

$$O(g(x)) = \left\{ \begin{array}{c} f(x) : \text{existen } c, x_0 > 0 \text{ tales que} \\ \forall x \geq x_0 : 0 \leq |f(x)| \leq c|g(x)| \end{array} \right\}$$

Notación: Se acostumbra escribir $f(x) = O(g(x))$ en vez de $f(x) \in O(g(x))$.

2.3.3 Cota Inferior Asintótica Ω(*g(x)*).

La cota inferior asintótica Ω(*g(x)*) es el conjunto de funciones acotadas inferiormente por $c \cdot g(x)$, a partir de un valor $x_0 > 0$, para algún $c > 0$.

---

27 Los números de Mersenne son números de la forma $2^p - 1$, donde *p* es un número primo.

28 Annals of Mathematics, 160 (2004), 781-793.

Formalmente,

$$\Omega(g(x)) = \left\{ \begin{matrix} f(x) : \text{existen } c, x_0 \text{ constantes positivas tales que} \\ \forall x : x_0 \leq x : 0 \leq cg(x) \leq f(x) \end{matrix} \right\}$$

Notación: Se acostumbra escribir $f(x) = \Omega(g(x))$ en vez de $f(x) \in \Omega(g(x))$.

2.3.4 Cota Ajustada Asintótica Θ(*g(x)*).

La cota ajustada asintótica Θ(*g(x)*) es el conjunto de funciones acotadas inferiormente por $c_1 \cdot g(x)$, y acotada superiormente por $c_2 \cdot g(x)$, a partir de un valor x_0, con $c_1, c_2, x_0 > 0$.

Formalmente,

$$\Theta(g(x)) = \left\{ \begin{matrix} f(x) : \text{existen } c_1, c_2, x_0 \text{ constantes positivas tales que} \\ \forall x : x_0 \leq x : 0 \leq c_1 g(x) \leq f(x) \leq c_2 g(x) \end{matrix} \right\}$$

Teorema: $f(x) \in \Theta(g(x))$ si y solo si $f(x) \in O(g(x))$ y $f(x) \in \Omega(g(x))$
Notación: Se acostumbra escribir $f(x) = \Theta(g(x))$ en vez de $f(x) \in \Theta(g(x))$.

2.3.5 Notación-L.

La notación-L es usada ampliamente para expresar la complejidad de un algoritmo. Por definición:

$$L_n[\alpha, c] = O\left(e^{(c+o(1))(\ln n)^{\alpha}(\ln \ln n)^{1-\alpha}}\right)$$

donde *c* es una constante positiva y α es una constante $0 \leq \alpha \leq 1$.

Cuando α = 0, entonces $L_n[0,c]$ es una función polinomial de *ln(n)* (logaritmo natural de *n*).
Cuando α = 1, entonces $L_n[0,c]$ es una función completamente exponencial de *ln(n).*

3 CURVAS ELÍPTICAS.

3.1 Definición. Una *Curva Elíptica* sobre un campo ***F*** queda definida por la ecuación de Weierstrass:

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

donde los $a_1 ... a_6 \in \boldsymbol{F}$.

Comentario (regla mnemotécnica): Podemos pensar que cada coeficiente a_i tiene asociado un peso *i*, que las indeterminadas *x* e *y* tienen pesos 2 y 3 respectivamente y que el peso de un monomio es la suma de los pesos de sus factores. Entonces los subíndices están elegidos para que todos los monomios tengan peso 6.

Un *punto* de una curva elíptica, es una pareja (x,y), con x, y $\in \boldsymbol{F}$, que satisfacen la anterior ecuación. La ecuación de la curva elíptica se denota con *E* y el conjunto de puntos en ***F*** que satisfacen *E* se denota *E(**F**).*

Para que las curvas elípticas sean criptográficamente seguras, las ecuaciones admiten transformaciones dependiendo del campo sobre el cual se están trabajando. Si la característica del campo ***F*** es distinta de 2 y de 3 (ejemplo: el campo formado por los reales ***F***=***R***), entonces existe un cambio de coordenadas racional tal que la ecuación de curva elíptica *E* tiene una *ecuación reducida de Weierstrass* de la forma:[29]

$$y^2 = x^3 + ax + b, \quad a, b \in \boldsymbol{F}$$

Para que éstas curvas sean criptográficamente útiles, no se deben admitir factores repetidos, o equivalentemente que $4a^3 + 27b^2 \neq 0$.

Más adelante veremos, que a partir del conjunto de puntos *E*(***F***), y definiendo una operación binaria y un elemento identidad, podemos formar un *grupo*.

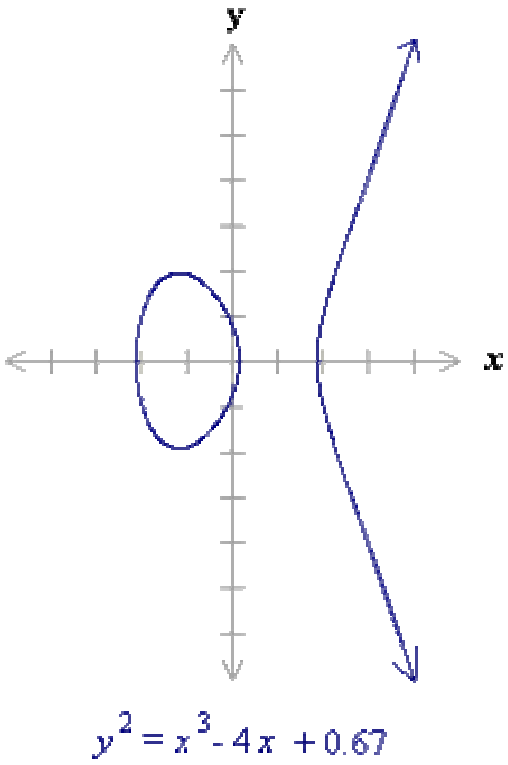
Ejemplo (curva elíptica en los reales):

$$y^2 = x^3 - 4x + 0.67$$

En la Figura 1 se muestra el gráfico resultante, note que la curva se extenderá hacia la derecha hasta el infinito.

[29] La existencia de tal cambio de coordenadas (para curvas de género 1 en general, con un punto racional dado) es una consecuencia del teorema de Riemann-Roch.

Figura 1. Gráfico de una curva elíptica



Si la característica de ***F*** es 2, usando también transformaciones lineales se obtiene una de las siguientes expresiones:

$$y^2 + cy = x^3 + ax + b, \quad a, b, c \in \boldsymbol{F}, \quad \text{con } c \neq 0.$$

o bien,

$$y^2 + xy = x^3 + ax + b, \quad a, b \in \boldsymbol{F}, \quad \text{con } b \neq 0.$$

Si el campo es de característica 3, E puede transformarse en la ecuación:

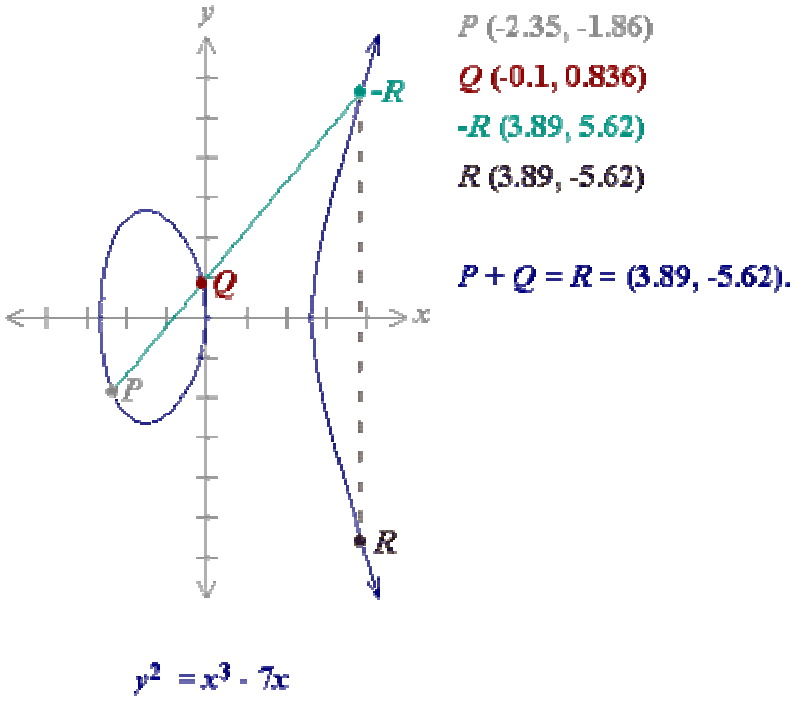
$$y^2 = x^3 + ax^2 + bx + c, \quad a, b, c \in \boldsymbol{F}$$

3.2 Descripción Geométrica de la Suma de Puntos

Inicialmente, se mostrará una explicación geométrica de la suma de puntos en *E, p*ara ello, definiremos una curva elíptica sobre el campo de los números reales **R**, posteriormente, cuando tratemos las curvas elípticas sobre los campos finitos $\boldsymbol{Z}_p$ *y* F_{2^m} se presentará una explicación analítica.

Considere el gráfico de la ecuación en ***R,*** $y^2 = x^3 - 7x$

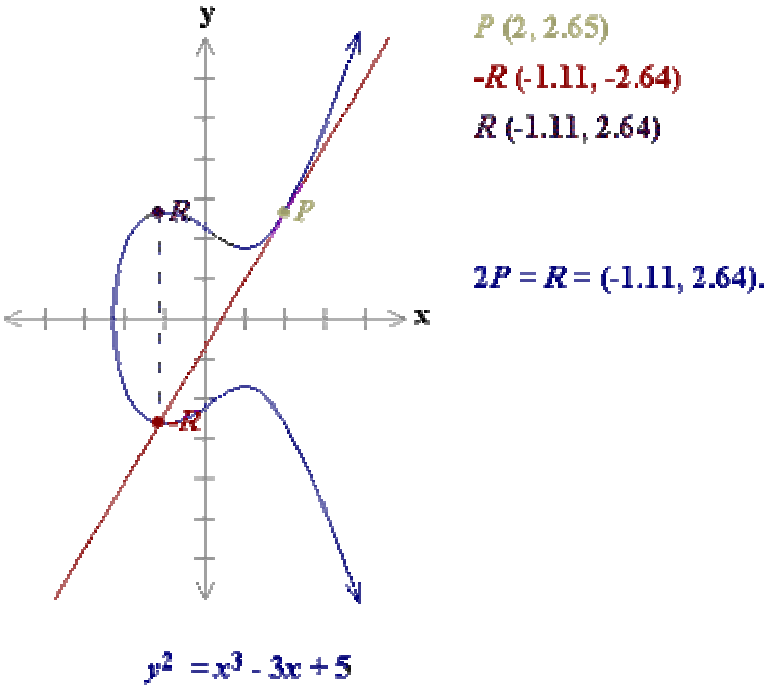
Figura 2. Suma de puntos de una Curva Elíptica.



Tome dos puntos *P* y *Q* de la curva, y trace la línea que pasa por ambos puntos. En el caso general esta línea siempre tiene un punto de intersección con la curva. Ahora tome este tercer punto en la intersección de la línea con la curva y trace una línea vertical. El otro punto *R* de intersección con la curva de esta línea vertical se define como la *suma* de *P* y *Q*, en símbolos, *R* = *P* + *Q*. El opuesto del punto R es el punto -*R* = (*x, -y*).

Si $P_1 = P_2$ y queremos obtener $R = P + P$ (ver Figura 3), entonces la línea a ser construida en el primer paso es la tangente a la curva, el cual otra vez tiene otro punto de intersección con la curva. Note que el grupo es conmutativo.

Figura 3. Suma del mismo punto *P*.

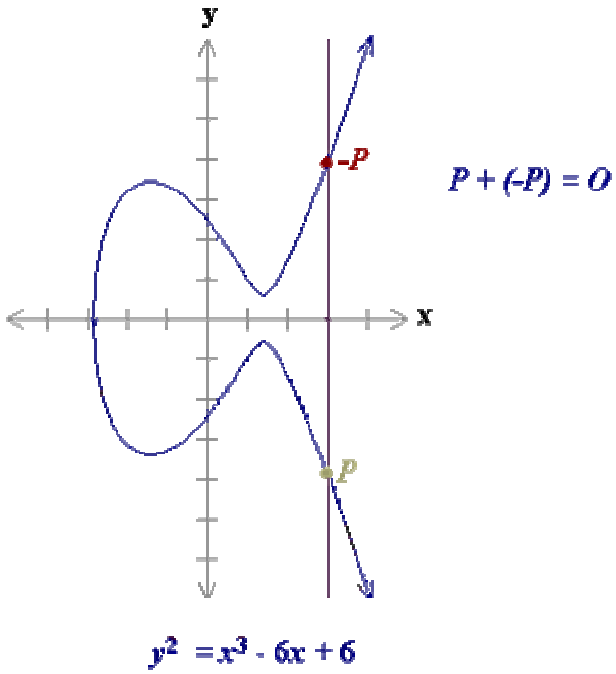


3.3 Punto en el Infinito O.

Para poder construir el grupo de la curva elíptica, agregaremos un punto extra ubicado en lugar infinitamente lejos sobre el eje vertical. Este punto extra se llama *point at infinity* o *punto en el infinito*, y se lo designa con *O*. El punto *O* es la identidad del grupo de la curva elíptica.

Por ejemplo, observe el punto *P* en Figura 4. Dada la Definición del elemento identidad (o elemento nulo), $P + (-P) = O$.

Figura 4 Suma del punto *P* con su opuesto.



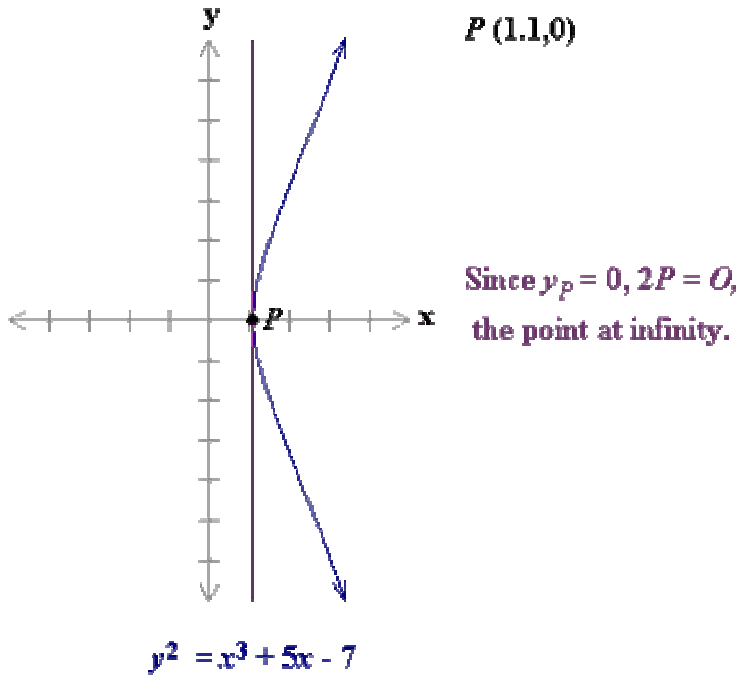
Si *k* es un entero positivo, entonces $kP = \sum_{i=1}^{k} P$ denota el punto obtenido de sumar *k*-veces el punto *P* consigo mismo. El proceso de calcular *kP* a partir de *P* y *k* es llamado *multiplicación escalar*.

El *orden* de una curva elíptica *E* definida sobre un campo finito ***F*** es el número de puntos sobre la curva elíptica *E*, incluyendo O. Esto es denotado por $\#E(F_q)$.

El *orden de un punto P* es el entero positivo más pequeño *n* tal que *nP* = *O* (el punto en el infinito).

Observe la Figura 5, el caso especial en el que queremos obtener el punto *P* + *P* = 2*P*, donde la recta tangente a la curva en el punto *P* es la recta vertical (lo cual sucede solo cuando *P* = (*x, y*) con *y* = 0); en este caso también obtenemos 2*P* = *O*.

Figura 5. Suma del punto P=(x,y) con si mismo, donde y=0.



Puede demostrarse que la operación de suma es una operación binaria de grupo bien definida, aunque algunos de los requerimientos como la asociatividad no son nada fáciles de probar.

Notaremos que en la practica, no podríamos considerar las curvas elípticas sobre cualquier campo en general porque no todos son criptográficamente convenientes, por ejemplo, al trabajar con el campo de los **R** tenemos problemas de truncamiento o redondeo. Debemos definir entonces, las curvas elípticas sobre campos finitos.

En adelante, consideraremos solo las curvas elípticas definidas sobre $\mathbf{Z}_p$ donde p es un número primo impar y especial atención le daremos a las curvas elípticas definidas sobre campos finitos de la forma F_{2^m} ($m \geq 1$), éstos últimos, producen las implementaciones más eficientes de la aritmética de curva elíptica por lo que fueron elegidos para la implementación de la aplicación software de este trabajo de grado.

3.4 CURVAS ELÍPTICAS SOBRE Z_p.

Definición Una *curva elíptica E* sobre $\mathbf{Z}_p$, denotada $E(\mathbf{Z}_p)$ es definida por la ecuación de la forma

$$y^2 = x^3 + ax + b \pmod{p}$$

donde $a, b \in \mathbf{Z}_p$, y $4a^3 + 27b^2 \neq 0 \pmod{p}$, junto con el *punto en el infinito O*.

El conjunto $E(\mathbf{Z}_p)$ consiste de todos los puntos (x, y), $x, y \in \mathbf{Z}_p$, los cuales satisfacen la anterior ecuación.

Ejemplo: Si $p = 23$, y consideramos la curva elíptica $E: y^2 = x^3 + x + 1$ definida sobre $\mathbf{Z}_{23}$ donde las constantes usadas fueron $a = b = 1$. Note que $4a^3 + 27b^2 = 4 + 4 = 8 \neq 0$ (mód 23), de manera que en realidad es una curva elíptica.

3.4.1 Teorema de Hasse en $E(\mathbf{Z}_p)$.

Sea $E(\mathbf{Z}_p)$ *una curva elíptica, con p primo impar, entonces* $p + 1 - 2\sqrt{p} \leq \#E(\mathbf{Z}_p) \leq p + 1 + 2\sqrt{p}$

En $E(\mathbf{Z}_p)$ es posible encontrar una fórmula algebraica para la suma de puntos $P_s = P_1 + P_2$, es decir, encontrar las coordenadas (x_S, y_S) del punto P_s, en función de las coordenadas (x_1, y_1) y (x_2, y_2) de los puntos P_1 y P_2 respectivamente.

3.4.2 Descripción Algebraica de la Suma de Puntos en $E(\mathbf{Z}_p)$.

En $E(\mathbf{Z}_p)$, la operación de *suma* de dos puntos de la curva se define según las siguientes reglas:

1. $O + O = O$.

2. $P + O = O + P = P$ para todo $P \in E(\mathbf{Z}_p)$.

3. Si $P = (x, y) \in E(\mathbf{Z}_p)$, entonces $(x, y) + (x, -y) = O$. (El punto $(x, -y)$ es denotado $-P$, y es llamado el *negativo* de P; observe que $-P$ es un punto de la curva.)

4. Sea $P_1 = (x_1, y_1) \in E(\mathbf{Z}_p)$ y $P_2 = (x_2, y_2) \in E(\mathbf{Z}_p)$, donde $x_1 \neq x_2$. Entonces $P_1 + P_2 = (x_S, y_S)$ es:
   $x_S = \lambda^2 - x_1 - x^2 \pmod{p}$
   $y_S = \lambda (x_1 - x_S) - y_1 \pmod{p}$, donde
   $\lambda = (y_2 - y_1) / (x_2 - x_1) \pmod{p}$

5. Sea $P_1 = (x_1 , y_1) \in E(\mathbf{Z}_p)$ con $y_1 \neq 0$. Entonces $P_1 + P_1 = (x_S , y_S)$, donde:
$x_S = \lambda^2 - 2x_1 \pmod p$
$y_S = \lambda (x_1 - x_S) - y_1 \pmod p$, donde
$\lambda = (3x_1^2 + a) / (2y_1) \pmod p$

Ejemplo: Considere la curva elíptica E : $y^2 = x^3 + x + 1$ definida sobre $\mathbf{Z}_{23}$ (p=23).

Sumemos los puntos P = (3, 10) y Q = (9, 7) (note que $P,Q \in E(\mathbf{Z}_{23})$.) Entonces $P + Q = (x_S , y_S)$ se calcula como sigue:

$\lambda = (7 - 10) / (9 - 3) = -3 /6 = -1/2 = 11 \in \mathbf{Z}_{23}$,
$x_S = 11^2 - 3 - 9 = 6 - 3 - 9 = -6 = 17 \pmod{23}$,
$y_S = 11(3 - (-6)) - 10 = 11(9) - 10 = 89 = 20 \pmod{23}$.

Por lo tanto, $P + Q$ = (17, 20).

Ahora calculemos $2P = P + P = (x_S, y_S)$.

$\lambda = (3(3^2) + 1) / 20 = 5 /20 = 1 / 4 = 6 \in \mathbf{Z}_{23}$,
$x_S = 6^2 - 6 = 30 = 7 \pmod{23}$,
$y_S = 6(3 - 7) - 10 = -24 - 10 = -11 = 12 \pmod{23}$:

Por lo tanto, $2P$ = (7, 12).

3.5 CURVAS ELÍPTICAS SOBRE F_{2^m}

F_{2^m} es el conjunto de todos los polinomios de grado < m con coeficientes en F_2.

$$F_{2^m} = \{a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + ... + a_1x + a_0 \mid a_i \in \{0, 1\}\}$$

A esta representación se la llama *representación polinomial*.[30]

Puesto que los coeficientes de los elementos en F_{2^m} son *ceros* o *unos*, cada elemento en F_{2^m} puede ser representado en una computadora como una palabra de m bits ($a_{m-1}a_{m-2} ... a_1a_0$).[31]

$$F_{2^m} = \{(a_{m-1}a_{m-2} ... a_1a_0) \mid a_i \in \{0, 1\}\}$$

$E(F_{2^m})$ es el conjunto de todas las soluciones (x,y), x,y $\in F_{2^m}$ que satisfacen la ecuación:

$$y^2 + xy = x^3 + ax^2 + b$$

[30] Existe otra representación, conocida como *representación de base optima*, la cual es más eficiente realizando multiplicaciones pero más difícil de implementar y revisar por terceros.

[31] Para las implementaciones en software, esta cadena de bits suele escribirse en notación hexadecimal, una forma más compacta que facilita su lectura.

donde a,b $\in F_{2^m}$, $b \neq 0$ (no supersingular).

En la representación con palabras de bits, el bit menos significativo a_0 en la primera posición de derecha a izquierda es el coeficiente de $x^0=1$ hasta el bit más significativo a_{m-1} que es el coeficiente de x^{m-1}. Esta forma de representación posicional para los polinomios hacen que la aritmética sobre puntos en $E(F_{2^m})$ sea más eficiente.

El lector podrá notar que en contraste a $E(\mathbf{Z}_p)$, $E(F_{2^m})$ es el conjunto de *parejas de polinomios* (x,y) con x,y $\in F_{2^m}$, donde las multiplicaciones y el inverso multiplicativo se hacen módulo un polinomio irreducible $f(x)$ de grado m.

En F_{2^m} la suma y la resta son operaciones equivalentes y se realizan haciendo XOR bit a bit de dos palabras de datos, la identidad aditiva es el elemento (0 ... 00), la identidad multiplicativa es el elemento (0 ... 01) y cada elemento es el inverso aditivo de sí mismo $(a_{m-1} \ldots a_1a_0) + (a_{m-1} \ldots a_1a_0) = (0 \ldots 00)$.

La multiplicación de dos polinomios $(a_{m-1} \ldots a_1a_0)$ $(b_{m-1} \ldots b_1b_0) = (r_{m-1} \ldots r_1r_0)$ donde $r_{m-1}x_{m-1} + \ldots + r_1x + r_0$ es residuo obtenido cuando el producto $(a_{m-1}x_{m-1} + \ldots + a_1x + a_0)$ $(b_{m-1}x_{m-1} + \ldots + b_1x + b_0)$ es dividido por el polinomio f(x) sobre F_2.

La exponenciación $(a_{m-1} \ldots a_1a_0)^e$ es calculada multiplicando e copias del polinomio $(a_{m-1} \ldots a_1a_0)$. En la multiplicación (y exponenciación) todos los coeficientes de los polinomios son reducidos módulo 2.

Ejemplo: Con representación polinomial, en F_{2^4} las 16 palabras de bits son:

(0001), (0010), (0100), (1000), (0011), (0110),

(1100), (1011), (0101), (1010), (0111), (1110),

(1111), (1101), (1001), (0001).

El polinomio irreducible usado para las operaciones será $f(x) = X^4 + x + 1$. Mostraremos algunos ejemplos de las operaciones aritméticas.

- **Suma.**

  (0110) + (0101) = (0011). ← XOR bit a bit.

- **Multiplicación.**

  (1101)(1001)
  $= (x^3 + x^2 + 1)(x^3 + 1)$ mód f(x)
  $= x^6 + x^5 + 2x^3 + x^2 + 1$ mód f(x)

$= x^6 + x^5 + x^2 + 1$ mód f(x) ← Los coeficientes son reducidos módulo 2.
$= (x^4 + x + 1)(x^2 + x) + (x^3 + x^2 + x + 1)$ mód f(x)
$= (x^3 + x^2 + x + 1)$ ← Se toma el residuo de la división polinomial.
$= (1111)$

- **Exponenciación**

$(0010)^5$ ← Se calcula de esta forma $((0010)^2)^2$ (0010) $= (0010)^4 (0010)$.
$(0010)^2$
$= (0010)(0010)$
$=$ x x mód f(x)
$= x^2$ mód f(x)
$= (x^4 + x + 1)(0) + (x^2)$ mód f(x)
$= x^2$
$= (0100)$
$(0010)^4$
$= (0010)^2 (0010)^2$
$= (0100)$ (0100)
$= x^2 x^2$ mód f(x)
$= (x^4 + x + 1)(1) + (x + 1)$ mód f(x)
$= x + 1$
$= (0011)$
$(0010)^5$
$= (0010)^4 (0010)$
$= (0011)$ (0010)
$= (x + 1)$ (x) mód f(x)
$= (x^2 + x)$ mód f(x)
$= (x^4 + x + 1)(0) + (x^2 + x)$ mód f(x)
$= x^2 + x$
$= (0110)$

También se puede usar una notación muy conveniente que evita tener que lidiar con $f(x)$ cada vez que se quieran realizar multiplicaciones y divisiones. Cada elemento de $F_{2^m}^*$ puede escribirse como el polinomio $g^i = x^i$ mód $f(x)$, $0 \leq i < 2^m - 1$, si $f(x)$ es irreducible primitivo. Luego, dados números i, j tenemos que:

$$g^i g^j = g^{(i+j) \text{ mód } (2^m - 1)} \text{ y } (g^i)^j = g^{ij} = g^{ij \text{ mód } (2^m - 1)}.$$

En F_{2^m} existe al menos un elemento g tal que los elementos no nulos en F_{2^m} pueden ser expresados como una potencia de g. Este elemento es llamado el generador del campo F_{2^m}. El inverso multiplicativo de un elemento $a = g^i$ es $a^{-1} = g^{-i \text{ mód } (2^m - 1)}$.

Ejemplo: Considere el campo F_{2^4}, que definiremos usando la representación polinomial, las operaciones se harán módulo $f(x) = X^4 + x + 1$ (irreducible).[32]

El elemento g = (0010) es el *generador* del campo. Las potencias de *g* son:

g^0=(0001), g^1=(0010), g^2=(0100), g^3=(1000), g^4=(0011), g^5=(0110),

g^6=(1100), g^7=(1011), g^8=(0101), g^9=(1010), g^{10}=(0111), g^{11}=(1110),

g^{12}=(1111), g^{13}=(1101), g^{14}=(1001), g^{15}=(0001).

La *identidad multiplicativa* es g^0 = (0001). El **inverso multiplicativo** de g^7=(1011) es g^{-7} $^{\text{mód } 15}$ = g^8 = (0101) . Para verificar esto note que:

$g^7\ g^{-7}$
= (1011) (0101)
= $(x^3 + x + 1)\ (x^2 + 1)$ mód f(x)
= $x^5 + x^2 + x + 1$ mód f(x)
= $(x^4 + x + 1)(x) + 1$ mód f(x)
= 1
= (0001) ← La identidad multiplicativa.

La notación basada en el generador g^e (en vez de usar las cadenas de bits), también permite la multiplicación sin referirse al polinomio irreducible $f(x) = X^4 + x + 1$.

Ejemplo: Considere la curva $y^2 + xy = x^3 + g^4x^2 + 1$. Donde $a = g^4$ y $b = g^0 = 1$. Verifiquemos que el punto (g^5, g^3) satisface ésta ecuación sobre F_{2^m} :

$$y^2 + xy = x^3 + g^4x^2 + 1$$

$$(g^3)^2 + g^5g^3 = (g^5)^3 + g^4g^{10} + 1$$

$$g^6 + g^8 = g^{15} + g^{14} + 1$$

$$(1100) + (0101) = (0001) + (1001) + (0001)$$

$$(1001) = (1001)$$

De igual forma, puede verificarse que en total los 15 puntos que satisfacen esta ecuación son:

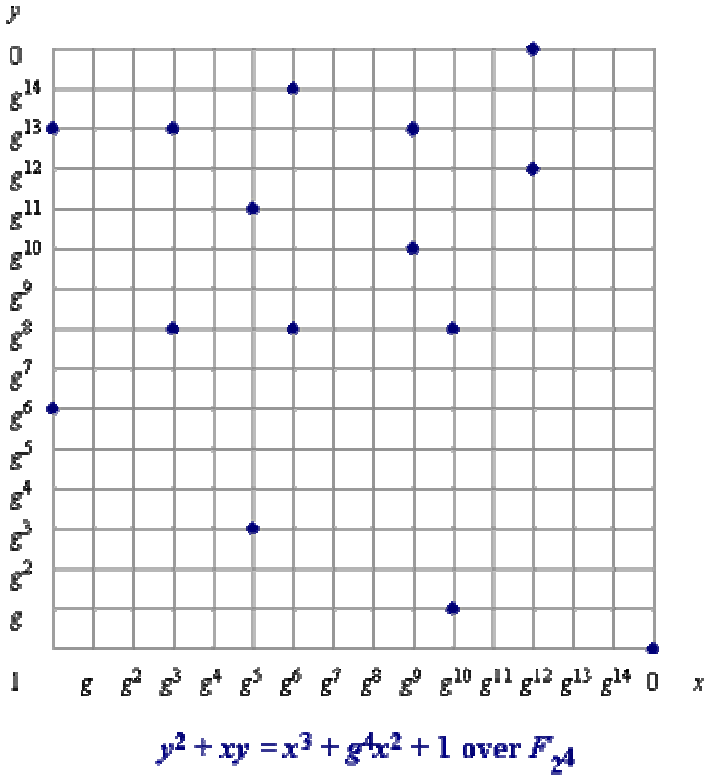
[32] Por supuesto, en un criptosistema real el parámetro *m* debe ser más grande, m=160 es una buena elección, pero *m=4* servirá para nuestro ejemplo.

$(1, g^{13})$ (g^3, g^{13}) (g^5, g^{11}) (g^6, g^{14}) (g^9, g^{13}) (g^{10}, g^8) (g^{12}, g^{12})

$(1, g^6)$ (g^3, g^8) (g^5, g^3) (g^6, g^8) (g^9, g^{10}) (g^{10}, g) $(g^{12}, 0)$ $(0, 1)$

La Figura 6 muestra la gráfica de estos puntos:

Figura 6. Gráfico de una curva elíptica sobre F_{2^4}



3.5.1 Curva Elíptica No Supersingular.

Sea F_{2^m} un campo finito, y sean $a, b \in F_{2^m}$ satisfaciendo $b \neq 0$ en F_{2^m}. Entonces una curva elíptica (no supersingular) $E(F_{2^m})$ sobre F_2 definida por los parámetros $a, b \in F_{2^m}$ consiste del conjunto de soluciones o puntos $P = (x, y)$ para $x, y \in F_{2^m}$ a la ecuación:

$$y^2 + xy = x^3 + ax^2 + b \text{ en } F_{2^m}$$

junto con un punto en el infinito O.

Conviene decir que, las únicas curvas elípticas sobre F_{2^m} criptográficamente útiles son las curvas elípticas no supersingulares.

Ejemplo: Podemos definir una curva elíptica $E(F_{2^4})$, donde $a, b \in F_{2^4}$. Es decir que $a = (a_3a_2a_1a_0)$ y $b = (b_3b_2b_1b_0) \neq (0000)$ son palabras binarias de 4 bits. La curva E la definen las palabras $x, y \in F_{2^4}$ que satisfagan la ecuación $y^2 + xy = x^3 + ax^2 + b$, donde las operaciones (salvo la suma) se hacen módulo el polinomio irreducible en F_{2^4} de grado 4 $f(x) = x^4 + x + 1$.

3.5.2 Curva de Koblitz.

Una *Curva de Koblitz* es una curva elíptica sobre F_{2^m} con $a, b \in \{0, 1\}$.

Ejemplo: Si el campo es F_{2^4}, $a = (0)_{10} = (0000)_2$, $b = (1)_{10} = (0001)_2$, $f(x) = x^4 + x + 1$, entonces $E: y^2 + xy = x^3 + 1$ es una curva elíptica de Koblitz no supersingular.

3.5.3 Teorema de Hasse en $E(F_{2^m})$.

Sea $E(F_{2^m})$ *una curva elíptica, con* $m \geq 1$, *entonces* $2^m + 1 - 2\sqrt{(2^m)} \leq \#E(F_{2^m}) \leq 2^m + 1 + 2\sqrt{(2^m)}$

3.5.4 Descripción Algebraica de la Suma de Puntos en $E(F_{2^m})$.

La *suma* de puntos en $E(F_{2^m})$ se define como:

1. $O + O = O$.

2. $P + O = O + P = P$ para todo $P \in E(F_{2^m})$.

3. Si $P = (x, y) \in E(F_{2^m})$, entonces $(x, y) + (x, x + y) = O$. (El punto $(x, x + y)$ es denotado -P, y es llamado el *negativo* de P; observe nuevamente que $-P$ es un punto de la curva).

4. Sea $P_1 = (x_1, y_1) \in E(F_{2^m})$ y $P_2 = (x_2, y_2) \in E(F_{2^m})$, donde $x_1 \neq x_2$.
   Entonces $P_1 + P_2 = (x_S, y_S)$ es:
   $x_S = \lambda^2 + \lambda + x_1 + x_2 + a$ en F_{2^m}
   $y_S = \lambda (x_1 + x_S) + x_S + y_1$ en F_{2^m}, donde
   $\lambda = (y_1 + y_2) / (x_1 + x_2)$ en F_{2^m}

5. Sea $P_1 = (x_1, y_1) \in E(F_{2^m})$ con $x_1 \neq 0$. Entonces $P_1 + P_1 = (x_S, y_S)$, donde:
   $x_S = \lambda^2 + \lambda + a$ en F_{2^m}
   $y_S = x_1^2 + (\lambda + 1)x_S$ en F_{2^m}, donde
   $\lambda = x_1 + (y_1 / x_1)$ en F_{2^m}:

3.6 EL PROBLEMA DEL LOGARITMO DISCRETO GENERALIZADO (PLDG)

En general, la seguridad de los criptosistemas asimétricos, descansa en la intratabilidad de un problema matemático que, por su gran magnitud, es computacionalmente inviable de resolver.

El esquema de cifrado RSA por ejemplo, basa su seguridad el el *Problema de la Factorización Entera (PFE),* que básicamente consiste en encontrar un divisor no trivial de un número compuesto. Este problema que para números pequeños es muy fácil, para números enteros grandes es muy difícil de resolver. Sin embargo, recientemente se han encontrado algoritmos que resuelven el problema en tiempo subexponencial, estos ataques lo describiremos más adelante.

Otros esquemas de cifrado y firma digital como el DSA y los criptosistemas de curva elíptica (ECC) basan su seguridad en una "instancia" del *problema del logaritmo discreto generalizado* que definiremos a continuación:

Dados un grupo cíclico finito G de orden n, un generador $\alpha \in G$, y un elemento $\beta \in G$, encontrar el entero x, $0 \leq x \leq n - 1$, tal que $\alpha^x = \beta$ es llamado el *problema del logaritmo discreto generalizado* (PLDG).

En el PLDG, el único requerimiento sobre G para que tenga un uso práctico en criptografía es que el PLDG sea un problema difícil de resolver en G, mientras que la exponenciación sea una operación eficiente de ejecutar.

Se cree que al *generalizar* la definición del PLDG para uso con *cualquier* grupo G y elementos $\alpha, \beta \in G$, donde el problema es encontrar un entero x tal que $\alpha^x = \beta$, si es que tal entero existe, sin que se requiera que G sea cíclico, ni tampoco que α sea un generador de G, es un problema mucho mas dificil de resolver.

En el grupo multiplicativo $\mathbf{Z}_p^* = \{1, 2, \ldots, p - 1\}$, donde p es un número primo y el orden de $\mathbf{Z}_p^*$ es $p - 1$, el *problema del logaritmo discreto (PLD)* es: dado un generador $\alpha \in \mathbf{Z}_p^*$, y un elemento $\beta \in Z_p^*$, encontrar el entero x, $0 \leq x \leq p - 2$, tal que $\alpha^x = \beta$ (mód p). La operación del grupo $\mathbf{Z}_p^*$ es la multiplicación módulo p.

Ejemplo: En el protocolo Difie-Hellman básico[33], dos partes A y B quieren compartir un secreto. A genera un primo p y generador $\alpha \in \mathbf{Z}_p^*$, $(2 \leq \alpha \leq p - 2)$, genera un número aleatorio (secreto) x, tal que $1 \leq x \leq p - 2$, calcula α^x mód p, y envía a B el mensaje (p, α, α^x mód p).

[33] El protocolo básico no provee ni autenticación de las partes involucradas en la comunicación, ni autenticación de los secretos generados.

Una vez que B recibió el mensaje, elige un número aleatorio y, $1 \leq y \leq p - 2$, y envía el mensaje (α^y mód p) hacia A. B luego calcula el secreto compartido haciendo $K = (\alpha^x)^y$ mód p.

Por otra parte, A recibe el mensaje de B, y calcula el secreto compartido haciendo $K = (\alpha^y)^x$ mód p.

$K = \alpha^{xy} \ (mód\ p)$ es el secreto compartido por ambos y en la practica, nadie que conozca $(p, \alpha, \alpha^x, \alpha^y)$ podría calcularlo, para hacerlo debe poder romper el PLD.

Otros criptosistemas han sido propuestos cuya seguridad se basa en el PLD, entre ellos están:

- Los esquemas de acuerdo de claves derivados del Difie-Hellman, tal como el de ElGamal, la familia de protocolos MTI y el protocolo STS (Station-to-Station).
- El esquema de encriptación ElGamal.
- El esquema de firma digital ElGamal y sus variantes, como DSA (Digital Signature Algorithm), el esquema de firma de Schnorr, y el esquema ElGamal con recuperación de mensaje de Nyberg-Rueppel.

3.7 PROBLEMA DEL LOGARITMO DISCRETO DE CURVA ELÍPTICA (PLDCE)

Definición (PLDCE): Sea E una curva elíptica definida sobre un campo finito F_q, y sea $G \in E(F_q)$ un punto sobre E de orden n (número primo y grande). El PLDCE es, dados E, G, y un múltiplo escalar Q de G, determinar un entero k tal que $Q = kG$.

k es llamado el *logaritmo discreto* de Q en base G.

Ejemplo: Considere la curva elíptica: $y^2 = x^3 + 9x + 17$ sobre F_{23}

¿Cuál es el logaritmo discreto k de Q = (4,5) en base G =(16,5) ?

Una forma de encontrar k es computando los múltiplos de G hasta encontrar Q. Los primeros múltiplos de G son:

G=(16,5) 2G=(20,20) 3G=(14,14) 4G=(19,20) 5G=(13,10) 6G=(7,3) 7G=(8,7)

8G=(12,17) 9G=(4,5)

En k=9, encontramos que 9G=(4,5)=Q, por lo que el logaritmo discreto de Q en base G es k=9.

En la práctica, k debe ser un número más grande tal que sea inviable encontrar k de esta manera.

Una vez definido el PLDCE, los esquemas de cifrado y firma digital basados en el PLD pueden ser adaptados utilizando curvas elípticas. En la actualidad, estas adaptaciones han tenido estandarizaciones como ECDSA, EC Difie-Hellman, encriptación EC ElGamal, etc.

Cabe recalcar aquí, que la razón por la que es conveniente utilizar los campos definidos sobre curvas elípticas y atenidos al PLDCE en vez de los campos sobre números enteros atenidos al PFE o al PLD, son los algoritmos encontrados para atacar el mismo problema para distintos grupos.

Además, hay que decir que, los criptosistemas basados en curvas elípticas no proveen ninguna ventaja en seguridad cuando los métodos basados en el PFE usan como grupo las curvas elípticas.

La importancia entonces está en los avances en criptoanálisis, por lo que presentaremos a continuación algunos de estos avances.

3.8 CRIPTOANÁLISIS AL PROBLEMA DE FACTORIZACION ENTERA (PFE).

Recientemente, un intento por factorizar un número de 200 digitos (665 bits) propuesto como el *Reto de Factorización RSA* tardó 18 meses utilizando computación en paralelo y consumió el equivalente a 75 años de tiempo de cómputo para un único procesador AMD Opteron de 2.2 GHz.[34]

RSA-200 = 27997833911221327870829467638722601621070446786955428537560009929326128400107609345671052955360856061822351910951365788637105954482006576775098580557613579098734950144178863178946295187237869221823983

RSA-200 = 35324619344027701212726049781984643686711974001976250236493034687761212536794232000585479565280883491 (primo)
*
7925869954478333033347085841480059687737975857364219960734330341455767872818152135381409304740185467 (primo)

Ya hemos mencionado que RSA utiliza $\mathbf{Z}_n^*$ como grupo, donde $n = pq$ con p y q números primos grandes elegidos al azar. Existen algoritmos de orden sub-exponencial que resuelven el problema de la factorización de n.

Estos algoritmos explotan la propiedad conocida como *smoothness* de los número enteros. Un número n es B-smooth si todos sus factores primos son ≤ B. Note que esta propiedad es de los números enteros, no del criptosistema RSA.

Mencionaremos algunos de los algoritmos para la factorización de números enteros:

- Algoritmo ρ de Pollard.

[34] F. Bahr, M. Boehm, J. Franke, and T. Kleinjung, Mayo 9 de 2005.

- Factorización por Curvas Elípticas de Lenstra
- Algoritmo p-1 de Pollard.
- Algoritmo p+1 de Williams
- Quadratic Sieve (QS)
- General Number Field Sieve (GNFS).
- Algoritmo de Shor (para computadores cuánticos)[35]

Además de utilizar la propiedad smoothness, algunos algoritmo utilizan una base de datos de números primos, llamada base de factores (*factor base*), esta base de datos hace que aumente la probabilidad de encontrar factores del número a factorizar (el módulo *n* en RSA). En especial, GNFS utiliza dos bases de factores y es fácil de paralelizar.

Actualmente, el GNFS es el algoritmo conocido más eficiente para factorizar enteros enormes utilizando computación "no cuántica" y la complejidad para factorizar un número *n* es:

$$O\left(e^{(c+o(1))(\log n)^{\frac{1}{3}}(\log\log n)^{\frac{2}{3}}}\right) = L_n\left[1/3, c\right]$$

para alguna constante *c*.

La siguiente tabla resume los *Retos de Factorización RSA* y la fecha en que cada uno ha sido roto (en su mayoría usando el algoritmo GNFS).[36]

Tabla 1. Retos de factorización RSA.

Número	***dígitos***	***bits***	***Factorizado***
RSA-100	100	333	Abr. 1991
RSA-110	110	366	Abr. 1992
RSA-120	120	399	Jun. 1993
RSA-129	129	429	Abr. 1994
RSA-130	130	432	Abr. 1996
RSA-140	140	466	Feb. 1999
RSA-155	150	499	Abr. 2004
RSA-160	160	532	Ago. 1999
RSA-200	200	665	Abr. 2003
RSA-576	174	576	Dic. 2003
RSA-640	193	640	Nov. 2005
RSA-704	212	704	--*sin factorizar*--
RSA-768	232	768	--*sin factorizar*--
RSA-896	270	896	--*sin factorizar*--
RSA-1024	309	1024	--*sin factorizar*--
RSA-1536	463	1536	--*sin factorizar*--
RSA-2048	617	2048	--*sin factorizar*--

[35] El algoritmo de Shor resuelve el problema en tiempo polinómico $O((\log n)^3)$. En 2001, la primer computadora cuántica de 7 cubits fue la primera en correr este algoritmo. Factorizó el número 15.

[36] Fuente: http://mathworld.wolfram.com/

3.9 CRIPTOANÁLISIS AL PROBLEMA DEL LOGARITMO DISCRETO (PLD)

El criptoanálisis al PLD sobre $\mathbf{Z}_p^*$ y $F_{p\wedge m}^*$ está inspirado en el criptoanálisis al PFE. Muchos de los algoritmos utilizados para atacar el PFE han sido adaptados para atacar el PLD sobre $\mathbf{Z}_p^*$ y $F_{p\wedge m}^*$ como por ejemplo el algoritmo GNFS. Esto quiere decir que los avances en criptoanálisis para un problema (PFE o PLD) implican avances en el criptoanálisis para el otro sobre los mismos grupos.

Un algoritmo muy conocido para el PLD es el Pollard-ρ (Pollard's rho) que no es sub-exponencial y tiene un tiempo esperado de ejecución de $O(\sqrt{n})$ operaciones, donde n es el orden del grupo cíclico. Otros algoritmos están condicionados como el algoritmo de Pohlig-Hellman que es eficiente solo cuando el orden n del grupo es smooth.

El algoritmo más eficiente no condicionado a las característica de los elementos del grupo es el index-calculus. En este método, una base de datos de primos pequeños y sus logaritmos correspondientes es calculada, esto sirve para poder calcular eficientemente logaritmos de elementos arbitrarios del grupo. El algoritmo index-calculus y todas sus variantes, como el algoritmo de Coppersmith y el Number Field Sieve (adaptado para logaritmos) tienen un tiempo de ejecución sub-exponencial y son fácilmente paralelizables.

3.10 CRIPTOANÁLISIS AL PROBLEMA DEL LOGARITMO DISCRETO DE CURVA ELÍPTICA (PLDCE)

Hasta el momento de escribir este documento, no se conocen algoritmos sub-exponenciales no condicionados para resolver el PLDCE. Tampoco se conoce el concepto de smoothness de un punto sobre una curva elíptica.

Aunque no se han encontrado algoritmos sub-exponenciales para el criptoanálisis del PLDCE, existen tres clases de curvas que son susceptibles a ataques:

1. **Curvas Supersingulares.**
   Las curvas elípticas E sobre F_q con n (el orden del punto base G) dividiendo a $q^B - 1$ para B pequeño. Estas curvas permiten reducir eficientemente el PLDCE al PLD sobre una extensión de grado pequeño de Fq (campo numérico).

2. **Curvas Anómalas.**
   Las curvas elípticas E sobre F_q con $\#E(F_q) = q$, permiten mapear eficientemente la curva elíptica E dentro del grupo aditivo de F_q.

3. Las curvas elípticas E sobre $F_{2\wedge m}$ cuando m es un número compuesto.

Estas tres clases débiles de curvas son fácilmente detectables y en la práctica no son utilizadas en la implementación de criptosistemas.

Dentro de los algoritmos no condicionados destacan los basados en el Pollard-ρ y el Pollard-λ. El Pollard-ρ toma alrededor de $\sqrt{(\pi n / 2)}$ pasos (cada paso representa la suma de dos puntos en una curva elíptica), y el Pollard-λ toma alrededor de $2\sqrt{n}$ pasos. Ambos métodos pueden ser paralelizados.[37]

Recientemente, Gallant, Lambert, Vanstone, Weiner y Zuccherato mostraron que el Pollard-ρ puede ser acelerado (*Pollard-ρ mejorado*) en un factor de $\sqrt{2}$, es decir una mejora que toma $\sqrt{(\pi n)} / 2$ pasos; también mostraron que condicionando el Pollard-ρ a curvas elípticas sobre F_{2^m} cuando *m* no es primo, este tipo de curvas permiten acelerar el Pollard-ρ en un factor de $\sqrt{(2d)}$ cuando $m=pd$ con *p* primo.

En la Tabla 2.[38] se ilustra la dificultad para resolver el PLDCE utilizando el algoritmo Pollard-ρ mejorado. Esta tabla contiene una estimación en MIPS-años. Un año MIPS es el tiempo de cómputo de un año por un computadora capaz de ejecutar un *Millón de Instrucciones Por Segundo* (MIPS). Para ubicar esta medida en un contexto, Odlyzko ha estimado que el 0,1% del poder computacional del todo el mundo trabajando durante todo un año equivale a 10^8 MIPS-años en el año 2004 y a 10^{10} o 10^{11} MIPS-años en el 2014.

Tabla 2. Poder computacional requerido para resolver el PLDCE.

Tamaño de n (en bits)	$\sqrt{(\pi n)} / 2$	***MIPS-años***
160	2^{80}	$8,5 \times 10^{11}$
186	2^{93}	$7,0 \times 10^{15}$
234	2^{117}	$1,2 \times 10^{23}$
354	2^{177}	$1,3 \times 10^{41}$
426	2^{213}	$9,2 \times 10^{51}$

3.11 PARÁMETROS DEL DOMINIO DE CURVAS ELÍPTICAS

Como hemos visto, no todas las curvas elípticas definidas sobre algún campo finito son convenientes en la construcción de los criptosistemas, algunas curvas deben excluirse. En contraste con otros criptosistemas, la seguridad en ECC no solo depende de la longitud de la clave sino también de los *parámetros* de la curva elíptica, de sus características.

Por otra parte, en el diseño de un criptosistema, las partes que intercambiarán información deben compartir estos parámetros. Los algoritmos de cifrado y descifrado, y los algoritmos de firma y verificación de firmas deben operar con los mismos parámetros del dominio para que el modelo PKI funcione.

---

[37] Certicom, Standards for Efficient Cryptography, *SEC 1: Elliptic Curve Cryptography, Version 1.0*, September 2000. http://www.secg.org/download/aid-385/sec1_final.pdf

[38] Extraída de: ANSI X9.62-1998: *Public Key Cryptography for the Financial Services Industry: the Elliptic Curve Digital Signature Algorithm (ECDSA)*. American Bankers Association, 1999.

Para garantizar la interoperabilidad entre criptosistemas, varios organismos han emitido estándares sobre los parámetros del dominio de curva elíptica. La Tabla 3. muestra algunos estándares para criptosistemas basados en ECC.

Tabla 3. Estándares ECC

Organismo	***Estándar***
IEEE	1363-2000 * 1363a * 1363.2
CEN	TC331 WG3 (DPM)
NESSIE	ECDSA * "PSEC"
SECG	SEC1 * SEC2
ANSI X9F	X9.24 Key management * X9.37 Check Image Exchange * X9.57 Cert management * X9.59 Payment * X9.62 ECDSA * X9.63 Key establishment * X9.68 Compressed certificates * X9.73 CMS * X9.84 Biometrics * X9.90 IRD * X9.92 ECPVS * X9.95 Time stamps * X9.96 XML CMS
FIPS	FIPS 186-2 Signatures (ECDSA) * SP 800-56 Key establishment * SP 800-57 Key Management.
FAA Security	Security * Next generation ATN * Secure ACARS
ISO	14888 * 15946 * 9796 * 18033 * ...
IETF	PKIX * S/MIME * IPSec (IKE) * TLS
CE	1394 * Consumer Electronics DTCP
OMA	WTLS * WPKI * WML Scripts * ...

3.11.1 Parámetros de dominio para curvas sobre F_p

Los parámetros de dominio de las curvas elípticas sobre el campo F_p son la séxtupla:

$$T = (p, a, b, G, n, h)$$

donde

p es un número entero primo p que especifica el campo finito F_p,
a,b son dos elementos $a, b \in F_p$ que determinan la curva elíptica

$$E: y^2 = x^3 + ax + b \pmod{p},$$

$G = (x_G, y_G) \in E(F_p)$ es un punto base sobre la curva de orden primo (análogo del generador a en el PLD).
n es un número primo grande el cual es el orden de G (n es tal que $nG=\mathbf{O}$)
h es un número entero que es el cofactor $h = \#E(F_p)/n$.

La condición de que n sea grande se debe a que hay que garantizar que sea inviable buscar todos los múltiplos $G, 2G, ..., (n\text{-}1)G$ de G. El cofactor h es un divisor de $\#E(F_p)$, puesto que $\#E(F_p) = nh$.

El *Standards for Efficient Cryptography Group (SECG)* en su documento *SEC2: Recommended Elliptic Curve Domain Parameters* sugiere generar la séxtupla $T = (p, a, b, G, n, h)$ con el siguiente proceso:

1. Elegir el nivel de seguridad aproximado (en bits) para los parámetros de dominio de curva elíptica. Este debe ser un entero $t \in \{56, 64, 80, 96, 112, 128, 192, 256\}$ de manera que calcular logaritmos sobre la curva generada tome aproximadamente 2^t operaciones. Luego, generar los parámetros de dominio de la curva elíptica como sigue:
2. Seleccionar un número primo p tal que $[\log_2 p\ ] = 2t$ si $t \neq 256$ y tal que $[\log_2 p] = 521$ si $t = 256$ para determinar el campo finito F_p.
3. Seleccionar elementos $a, b \in F_p$ para determinar la curva $E(F_p)$ definida por la ecuación $E : y^2 = x^3 + ax + b$ (mód p), un punto base $G = (x_G, y_G)$ sobre $E(F_p)$, un número primo n el cual es el orden de G, y un cofactor entero $h = \#E(Fp)/n$, sujeto a las siguientes restricciones:
   (i) $4a^3 + 27b^2 \neq 0$ (mód p).
   (ii) $\#E(F_p) \neq p$.
   (iii) $p^\beta \neq 1$ (mód n) para todo $1 \leq B < 20$.
   (iv) $h \leq 4$.
4. retornar (p, a, b, G, n, h).

Comentario. Según el nivel de seguridad deseado, el número primo p de $\mathbf{Z}_p$ se suele elegir de manera que:

$$[\log_2 p\ ] \in \{112, 128, 160, 192, 224, 256, 384, 521\}$$

Esto representa el tamaño en bits de p. Note la aparición del número 521, en vez de 512, esto es un requerimiento que aparece en parámetros estándares de seguridad del gobierno de U.S.

Con respecto a las restricciones que deben cumplir los demás parámetros, (i) se pide para que se cumpla de definición que dimos de curvas elípticas sobre F_p. La condición (ii) se pide para evitar las curvas anómalas, y la (iii) para evitar las curvas supersingulares (a esta condición se la conoce como *condición MOV* debido a sus descubridores: Menezes, Okamoto, Vanstone). La condición (iv) pide que $h \leq 4$, esto es para que $\#E(F_p)$ sea un número "cerca" de un primo grande.

3.11.2 Parámetros del dominio para curvas sobre F_{2^m}

Los parámetros de dominio de una curva elíptica sobre F_{2^m} son una séptupla

$$T = (m, f(x), a, b, G, n, h)$$

Donde,

m es un número entero que define el campo finito F_{2^m}
$f(x)$ es un polinomio binario irreducible de grado m
$a,b \in F_{2^m}$ dos elementos que definen la curva elíptica

$$E:\ y^2 + xy = x^3 + ax^2 + b$$

$G \in E(F_{2^m})$ es un punto base de la curva.
n es un número primo que corresponde al orden de G
h es un número entero cofactor. $h = \#E(F_{2^m})/n$.

Comentario. Para respetar la definición que hemos dados para curvas elípticas sobre F_{2^m} se presenta m como un número entero (arbitrario), pero ya hemos visto que se debe evitar elegir m como un número compuesto; las curvas sobre estos campos son débiles y susceptibles a ataques.

Nuevamente, en aras de garantizar la interoperabilidad entre distintos criptosistemas, el *SECG* sugiere generar una séptupla $T = (m, f(x), a, b, G, n, h)$ con el siguiente proceso:

1. Seleccionar el nivel de seguridad (en bits) apropiado para los parámetros de curva elíptica (este debe ser un número entero $t \in \{ 56, 64, 80, 96, 112, 128, 192, 256 \}$ de manera que calcular logaritmos sobre la curva elíptica asociada tome aproximadamente 2^t operaciones.
2. Sea t' el número entero más pequeño que sea mayor que t en el conjunto { 64, 80, 96, 112, 128, 192, 256, 512 }. Seleccionar $m \in \{ 113, 131, 163, 193, 233, 239, 283, 409, 571 \}$ tal que $2t < m < 2t'$ para determinar el campo finito F_{2^m}.
3. Seleccionar un polinomio binario irreducible $f(x)$ de grado m de la Tabla 4 para determinar la representación de F_{2^m}.

Tabla 4. Polinomio irreducible para F_{2^m}.

Campo	***Polinomio irreducible***
F_{2^113}	$f(x) = x^{113} + x^9 + 1$
F_{2^131}	$f(x) = x^{131} + x^8 + x^3 + x^2 + 1$
F_{2^163}	$f(x) = x^{163} + x^7 + x^6 + x^3 + 1$
F_{2^193}	$f(x) = x^{193} + x^{15} + 1$
F_{2^233}	$f(x) = x^{233} + x^{74} + 1$
F_{2^239}	$f(x) = x^{239} + x^{36} + 1$ ó $f(x) = x^{239} + x^{158} + 1$
F_{2^283}	$f(x) = x^{283} + x^{12} + x^7 + x^5 + 1$
F_{2^409}	$f(x) = x^{409} + x^{87} + 1$
F_{2^571}	$f(x) = x^{571} + x^{10} + x^5 + x^2 + 1$

4. Seleccionar elementos $a, b \in F_{2^m}$ para determinar la curva elíptica $E(F_{2^m})$ definida por la ecuación $E : y^2 + xy = x^3 + ax^2 + b$ en F_{2^m}, un punto base $G = (x_G, y_G) \in E(F_{2^m})$, un número primo n el cual es el orden de G, y un entero h, que es el cofactor $h = \#E(F_{2^m})/n$, sujetos a las siguientes restricciones:
   (i) $b \neq 0$ en F_{2^m}.
   (ii) $\#E(F_{2^m}) \neq 2^m$.

(iii) $2^{mB} \neq 1$ (mód n) para todo $1 \leq B < 20$.

(iv) $h \leq 4$.

5. retornar (m, $f(x)$, a, b, G, n, h).

Comentario. La regla para elegir al polinomio irreducible $f(x)$ es elegir un trinomio de la forma $f(x) = x^m + x^k + 1$, con $m > k \geq 1$, y k tan pequeño como sea posible.

Si un trinomio irreducible no existe, se opta por un pentanomio de la forma $f(x) = x^m + x^{k3} + x^{k2} + x^{k1} + 1$, $m > k_3 > k_2 > k_1 \geq 1$, con:

a) k_3 tan pequeño como sea posible
b) k_2 tan pequeño como sea posible dado k_3
c) k_1 tan pequeño como sea posible dado k_3 y k_2.

Estos polinomios habilitan el cálculo eficiente de las operaciones del campo, ya que son los polinomios irreducibles sobre F_{2^m} de menor cantidad de términos posibles (ver preliminares 2.2.28).

Con respecto a las restricciones que deben cumplir los demás parámetros, (i) se pide para que se cumpla le definición que dimos de curvas elípticas sobre F_{2^m}. La condición (ii) se pide para evitar las curvas anómalas, y la (iii) para evitar las curvas supersingulares. La condición (iv) pide que $h \leq 4$, esto es para que $\#E(F_{2^m})$ sea un número "cerca" de un primo grande.

4. DESARROLLO DEL PROYECTO.

4.1 METODOLOGÍA

La metodología que se siguió para el desarrollo del software fue la del Proceso Unificado de Desarrollo de Software, creada por Ivar Jacobson, Grady Booch y James Rumbaugh, miembros de Rational Software Corporation.

Esta metodología busca guiar de manera lógica y ordenada todas las actividades necesarias para lograr el desarrollo óptimo de un sistema. El Proceso Unificado está fundamentado en los siguientes tres aspectos: Dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental.

Dirigido por casos de uso: El desarrollo del software se centra en la importancia del desarrollo para el usuario y no en términos de funciones que debe cumplir el sistema. Los casos de uso dirigen el proceso durante todos los flujos de trabajo de las distintas fases.

Un caso de uso es una descripción de un conjunto de secuencias de acciones que un sistema lleva a cabo, un fragmento de su funcionalidad y que proporciona a un resultado interés para un actor determinado, donde un actor puede ser un usuario, un sistema o un rol.

Centrado en la arquitectura: Al describir la arquitectura se obtiene una mayor comprensión del sistema, se organiza el desarrollo y se fomenta la reutilización. Esta arquitectura abarca la organización del sistema software, los elementos estructurales que compondrán el sistema y sus interfaces, así como su comportamiento y colaboraciones entre elementos.

Es iterativo e incremental: Un proceso iterativo permite una comprensión creciente de los requerimientos, a la vez que se va haciendo crecer el sistema abordando las tareas más riesgosas primero. El trabajo de desarrollo se divide de manera planeada en partes más pequeñas llamadas iteraciones lo cual genera progresivamente un incremento en el proyecto total.

En cada iteración, se identifican y especifican los casos de uso relevantes, se crea un diseño utilizando la arquitectura seleccionada como guía, se implementa el diseño mediante componentes, y se verifican que los componentes satisfacen los casos de uso. Si una iteración cumple sus objetivos el desarrollo continúa con la siguiente iteración, en caso contrario, se revisan las decisiones previas y se prueba un nuevo enfoque.

Un desarrollo iterativo, guiado por los casos de uso y centrado en la arquitectura, construye un software mediante pequeños incrementos, y añade cada incremento a la acumulación previa de incrementos de tal forma que siempre se tenga una construcción

ejecutable. La arquitectura proporciona la estructura sobre la cual guiar las iteraciones mientras que los casos de uso definen los objetivos y dirigen el trabajo de cada iteración.

De esta manera el proceso reduce el riesgo de grandes retrasos en la entrega de un producto, se fijan metas más inmediatas por lo cual se puede controlar mejor el avance del proyecto.

Para construir una versión del producto siguiendo esta metodología se realizan cuatro fases, cada una de las cuales se divide en iteraciones o mini-proyectos, cada iteración pasa por cinco flujos de trabajo como se puede observar en la Figura 7.

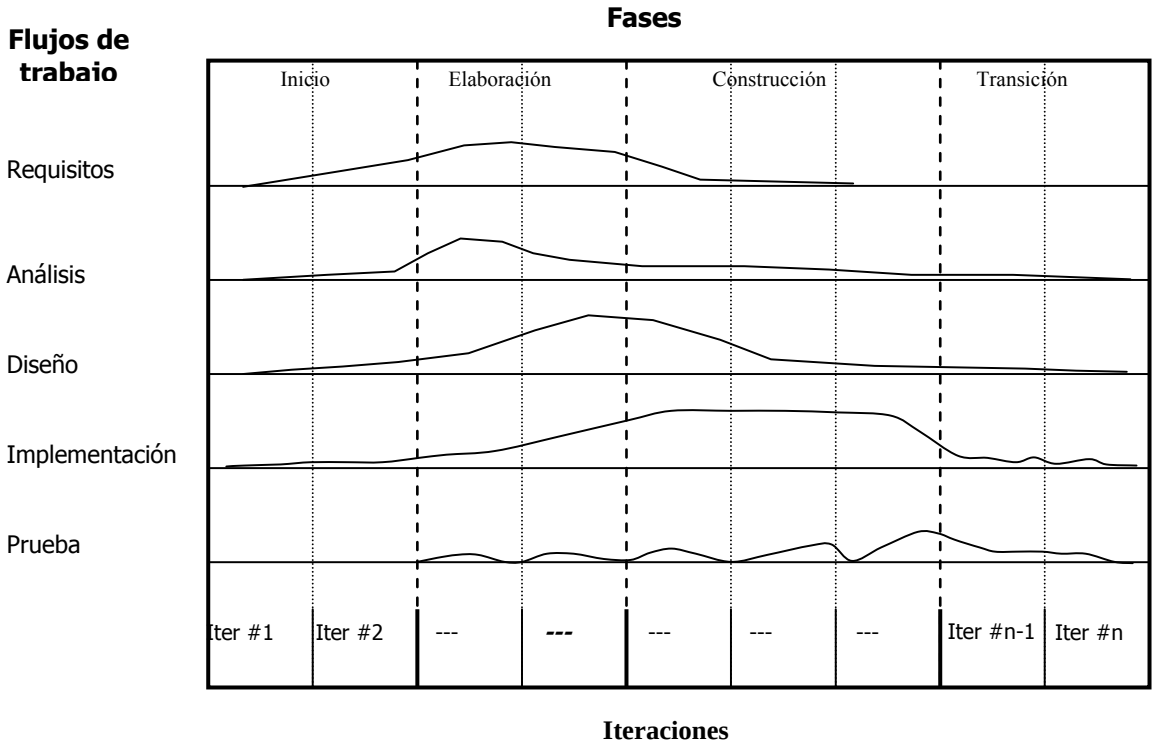


Figura 7. Fases y flujos de trabajo

Requisitos: su propósito es describir los requisitos del sistema, es decir, lo que el sistema debe hacer de acuerdo a lo discutido entre el usuario y el desarrollador.

Análisis: Su objetivo es lograr una comprensión más precisa de los requisitos y obtener una descripción de éstos que permita estructurar el sistema de una manera más fácil.

Diseño: Su propósito fundamental es formular modelos que se centran en los requisitos no funcionales y el dominio de la solución.

Implementación: Su propósito es la construcción del sistema, es decir, la elaboración del código fuente, interfaces, ejecutables, etc.

Prueba: Su objetivo es comprobar el resultado de la implementación mediante pruebas.

Las fases de un ciclo son:

- **Inicio:** En esta fase se desarrolla una descripción del producto final a partir del modelo de casos de uso y un esbozo de la arquitectura provisional del sistema. Además, se planifica en detalle la fase de elaboración.

- **Elaboración:** Se especifican en detalle la mayoría de los casos de uso y se diseña la arquitectura del sistema. Durante esta fase se realizan los casos de uso más críticos detectados en la fase de inicio. Además se planifican las actividades y se estiman los recursos necesarios para terminar el proyecto.

- **Construcción:** Se consolida la arquitectura del sistema, lo que permite construir el sistema de acuerdo a los casos de uso que se habían establecido.

- **Transición:** se verifica si realmente se cumplió con los objetivos propuestos, de no ser así, se hacen las respectivas correcciones para poder entregar el producto final.

4.2 PLAN DE TRABAJO

De acuerdo a la metodología explicada en el punto anterior, se especifican a continuación cada una de las actividades realizadas en las diferentes fases.

1. Inicio. En esta fase correspondió recopilar la información necesaria para desarrollar el proyecto. Las actividades seguidas fueron:

- Recopilar la información bibliográfica necesaria para el desarrollo del proyecto.
- Realizar un estudio matemático extenso para comprender los conceptos básicos y generales de las curvas elípticas.
- Elaborar el marco teórico en el cual se especifica el problema y se visualizan las posibles soluciones.
- Realizar un estudio de los posibles lenguajes de programación y la plataforma a utilizar para el desarrollo del proyecto, y escoger el que mejor se ajuste a los requerimientos del sistema.
- Esbozar los casos de uso funcionales más relevantes para el modelo.

2. Elaboración. Las actividades a realizar en esta fase son:

- Estudiar los diferentes modelos matemáticos subyacentes a los ECC y seleccionar aquellos que se hacen apropiados para criptografía.

- Diseñar el modelo de datos que se utilizará para el almacenamiento de la información financiera y especificar de los campos y registros que serán objeto de cifrado.
- Establecer un procedimiento confiable que permita emular una entidad certificadora para un proceso de firma digital.
- Diseñar una política de copias de respaldo a la base de datos.
- Hacer la evaluación de costos de implementación y determinar el más conveniente desde el punto de vista costo/beneficio.
- Evaluar el recurso hardware necesario de forma que posibilite elaborar comparativos de transferencias en entornos con diferentes condiciones de recursos.
- Especificar en mayor detalle los casos de uso.

3. Construcción. En esta fase se desarrollará la codificación del modelo, estableciendo las siguientes actividades:

- Implantar el modelo de datos diseñado en la etapa anterior.
- Elaborar los algoritmos de cifrado y descifrado de curva elíptica que serán utilizados en los diferentes procesos del sistema.
- Construir en software el procedimiento automático de copias de respaldo.
- Implementar el sistema de firma digital que garantice la integridad y la autenticidad de los datos que serán transferidos.
- Implementar las diferentes interfaces de usuario, un módulo de mantenimientos de tablas soportes, variables de entornos y las diferentes entradas requeridas por el sistema para el almacenamiento y tratado de los datos.
- Implementar cada uno de los casos de usos establecidos previamente.
- Construir el prototipo de aplicación que integre los aspectos anteriores y que permita la transferencia de la información en forma segura.
- Elaborar el manual del usuario de la aplicación.

4. Transición. Con el fin de verificar las acciones del sistema. Las actividades a ejecutar son:

- Efectuar las pruebas de verificación de la interfaz.
- Validar los algoritmos implementados con al menos uno de los métodos conocidos de criptoanálisis y verificar su funcionalidad.
- Hacer el comparativo de eficiencia, consumo de memoria, etc; en al menos dos ambientes con diferentes recursos.

4.3 DISEÑO E IMPLEMENTACIÓN DE LOS CRIPTOSISTEMAS.

A continuación describiremos el prototipo de aplicación desarrollado, se presentarán los diagramas modulares de los criptosistemas desarrollados, se explicará el modelo de datos que soportan toda la estructura lógica del almacenamiento y tratado de la información, el cómo se genera dicha información, el mantenimiento de los datos, la gestión de perfiles de usuario para acceso restringido al sistema, la elección de los parámetros del dominio de curva elíptica sobre F_{2^m}, el *modus operandi* en el cifrado, descifrado, firma y verificación de firma que garantizarán los principios básicos de la seguridad informática, a saber, la *integridad*, *confidencialidad*, *autenticidad*, *no repudio* y *disponibilidad;* se presentará un

protocolo de alto nivel diseñado sobre el protocolo HTTP que permitirán el intercambio de información financiera entre las entidades involucradas en el modelo y entre éstas y un tercero de confianza que emula una *Autoridad Certificadora* encargada de la gestión de las claves públicas, y finalmente se describirán las políticas de seguridad seguidas.

Antes de continuar, se recomienda revisar el anexo B., un glosario que recoge los términos propios de los criptosistemas desarrollados y otros referentes al modelo de negocios financieros que permitirán un mejor seguimiento de las siguientes secciones.

4.3.1 FILOSOFÍA DE SERVICIO.

Pensando en solucionar un problema real y palpable sobre la transferencia de información financiera sobre redes inseguras, en este trabajo de grado se ha diseñado un modelo de transferencia de giros, en donde cada entidad involucrada en el sistema es un *punto de giro o sede (o simplemente una entidad financiera)* que maneja un par de claves en un esquema de criptografía asimétrica. La *infraestructura de clave pública* (PKI) está conformada por un número arbitrario de estas entidades, un tercero de confianza que actúa como Autoridad Certificadora, un esquema criptográfico basado en curvas elípticas y un protocolo de comunicación definido.

La elección de *remisión de giros* como proceso central del sistema no es arbitraria, y obedece a la importancia que en los últimos años han adquirido para la economía del país los rubros como *giros nacionales y remesas*[39] y al hecho de que distintas entidades financieras demandan los mismos niveles de seguridad pero los recursos de pequeñas entidades (como microempresas dedicadas a giros nacionales) no les permiten adquirir costosos sistemas criptográficos para la protección de la información. Se presenta aquí un prototipo de aplicación fiable, funcional, a bajo costo, implementado con herramientas de software libre, capaz de operar sobre redes inseguras y cubriendo los principales requerimientos de un sistema seguro.

Si bien no hay un informe reciente sobre *giros nacionales*, desde el punto de vista de la seguridad informática no perderemos generalidad al mostrar algunas estadísticas de *remesas,* es claro que el flagelo no hace distintivos entre estos tipos de rubros.

En Colombia, el flujo de remesas pasó de 1578 millones de dólares en el 2000 a 3890 millones de dólares en el 2006. Es una cifra tan significativa para la economía del país que los ingresos representados por este rubro superan a los ingresos generados por las exportaciones de café y carbón.

---

[39] La distinción entre giros nacionales y remesas se hace de acuerdo a la definición del *Manual de Balanza de Pagos del Fondo Monetario Internacional*, se consideran remesas a tres conceptos contables diferentes: Worker's remittances (remesas de trabajadores), compensation of employees (compensación de empleados) y migrant's transfers (transferencia de emigrantes); las tres están relacionadas directamente con el envío de dinero desde el exterior.

En el año anterior (2006), este rubro representó el 2,9% del PIB, 2,7 veces más que las exportaciones de café, 0,6 veces las exportaciones de petróleo y derivados, 1,3 veces las de carbón y 0,3 veces las exportaciones de productos no tradicionales. La Tabla 5 recoge las estadísticas de los últimos 7 años.[40]

Tabla 5. Flujo de remesas en Colombia.

	2000	***2001***	***2002***	***2003***	***2004***	***2005***	***2006***
Remesas en Millones de USD.	**1.578**	**2.021**	**2.454**	**3.060**	**3.170**	**3.314**	**3.890**
Como % del PIB	1,9%	2,5%	3,0%	3,8%	3,2%	2,7%	2,9%
Como % de los ingresos corrientes – Balanza de pagos.	8,4%	10,9%	13,7%	15,4%	13,1%	11,1%	11,1%
Como % de las exportaciones de bienes.	11,5%	15,7%	19,9%	22,1%	18,4%	15,2%	15,4%
Como número de veces de las exportaciones de:							
Café	1,5	2,6	3,2	3,8	3,3	2,3	2,7
Petróleo y derivados	0,3	0,6	0,7	0,9	0,7	0,6	0,6
Carbón	1,8	1,7	2,5	2,2	1,7	1,3	1,3
No tradicionales	0,2	0,3	0,4	0,4	0,4	0,3	0,3
Como proporción de:							
Ingresos por Inversión extranjera discreta (%)	65,9%	80,1%	116,0%	169,9%	101,7%	32,3%	61,8%

4.3.2 DIAGRAMAS MODULARES DE LOS CRIPTOSISTEMAS.

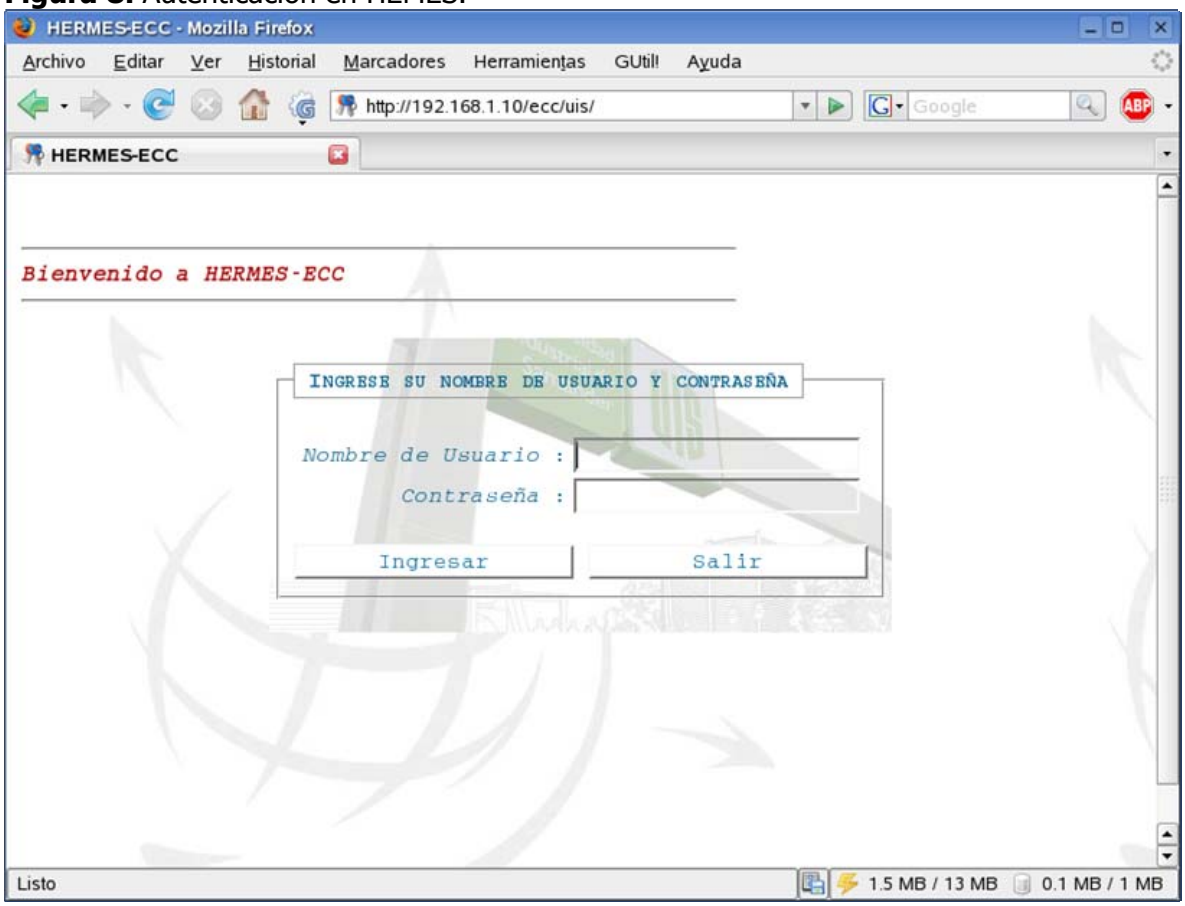
Aunque el sistema debe ser visto como un todo, se ha querido mostrar estructuralmente los principales procesos que componen el sistema, esto permitirá al lector hacerse una idea global del funcionamiento general del modelo desarrollado.

El nombre dado a la aplicación software que opera en cada entidad financiera es HERMES y el software implementado para la entidad financiera que actúa como Autoridad Certificadora es REPOSITORIO-ECC. A manera de comentario, diremos que la palabra HERMES está relacionada al dios griego del comercio (mitología), mensajero de los dioses y traductor de significados ocultos.

[40] Fuente: Centro de Estudios Monetarios Latinoamericanos, Fondo Multilateral de Inversiones, Banco Interamericano de Desarrollo. *Remesas Internacionales en Colombia,* Marzo 2007.

- **Criptosistema HERMES.**

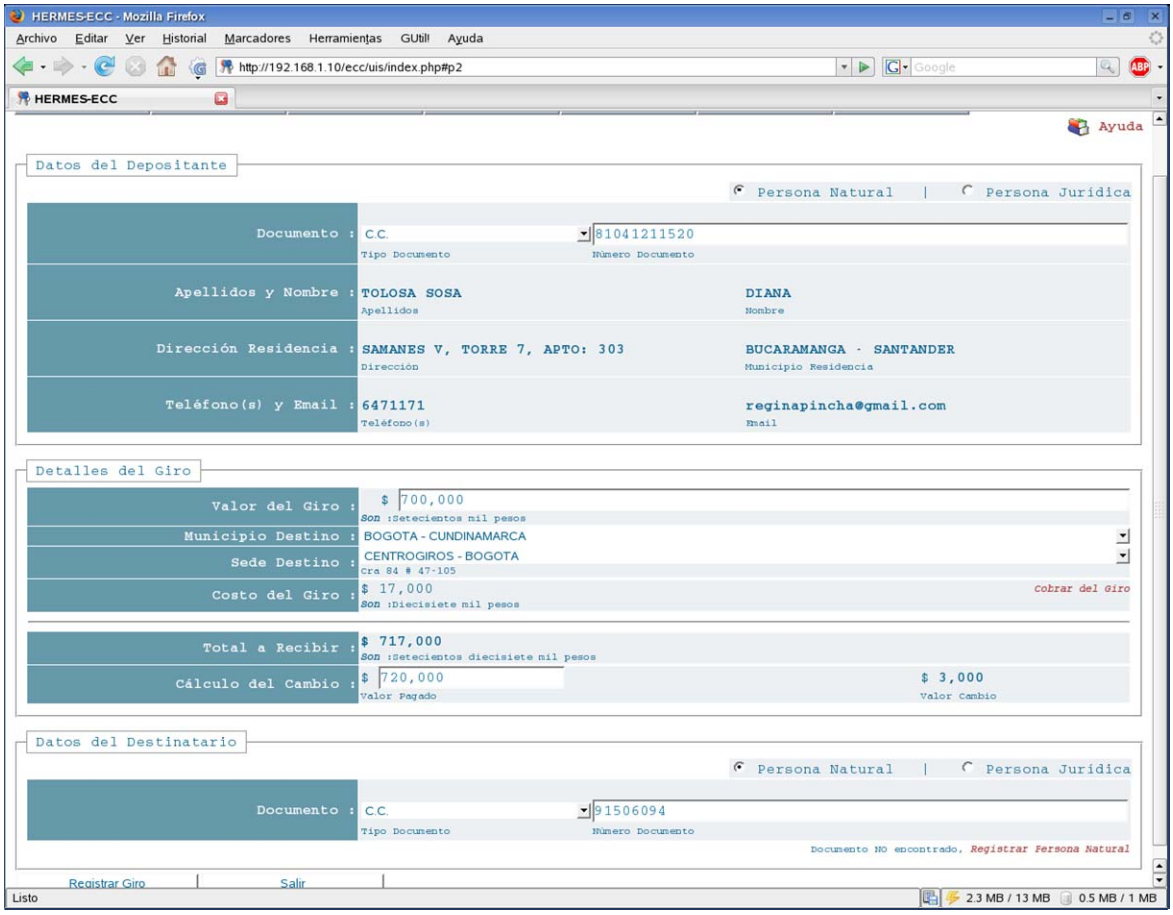
Figura 8. Autenticación en HEMES.



El sistema HERMES se ha pensado como un sistema completo, operando en redes inseguras, con puntos de giros en ciudades distantes y conectadas principalmente a través de la Internet, los principales procesos que se realizan en el sistema son:

1. Generar la información a proteger. En nuestro caso, la información a proteger es el conjunto de datos que un cliente deposita en el evento de remisión de un giro desde una ciudad a otra.

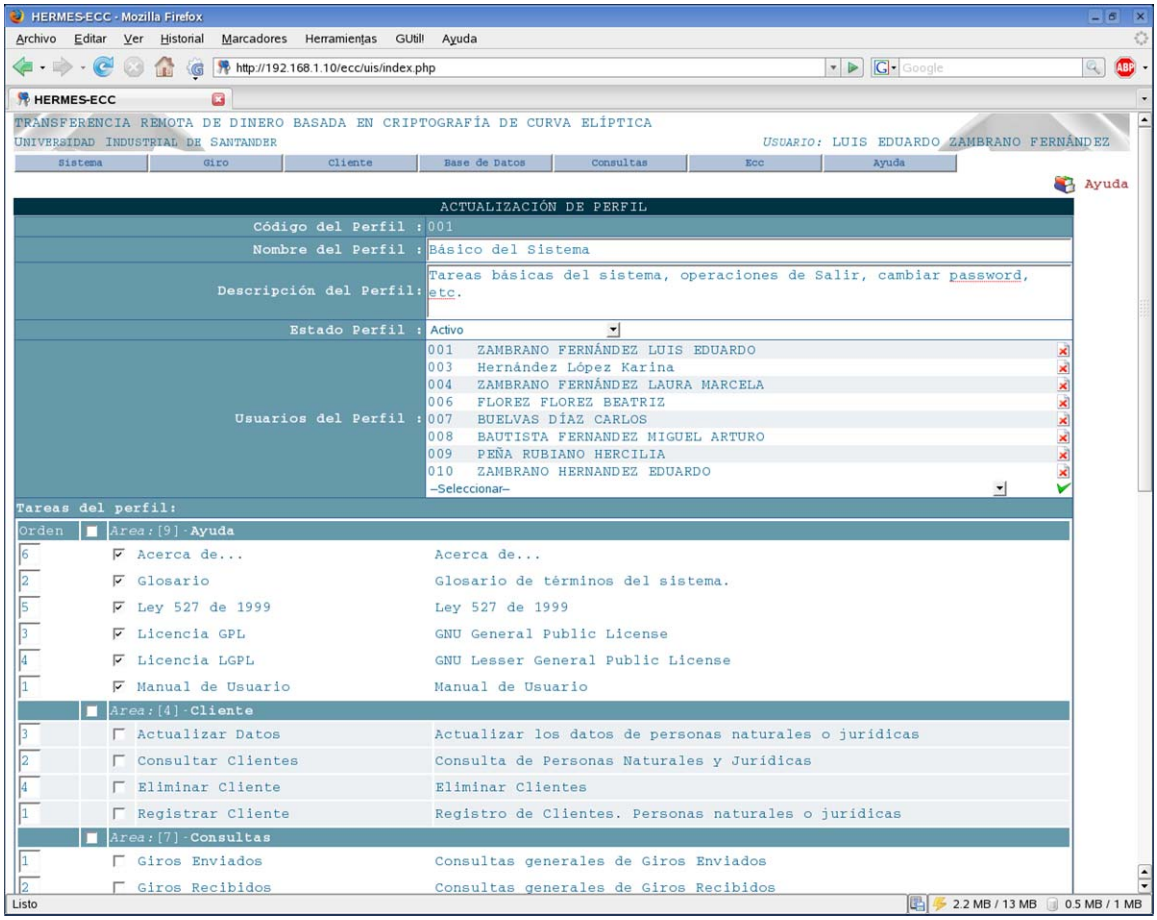
Figura 9. Generación de la información.



2. Cada entidad financiera o punto de giro, maneja un par de claves de cifrado asimétrico. Estas claves son las que se utilizan para cifrado, descifrado, firmado digital y verificación de firmas en cada transacción.

3. Gestionar *cuentas y perfiles de usuario* que permitan ofrecer distintos roles para usuarios administradores y usuarios comunes del sistema. De modo que se ha ajustado un menú *dinámico* (cambiante según el usuario que inicia sesión en el sistema).

Figura 10. Actualización de un Perfil.



4. Un módulo de acceso restringido al usuario administrador que permite manejar toda o parcialmente la información de la *base de datos*, como tablas de variables de entorno, registro de nuevos puntos de giro y mantenimiento de cualquier tabla en general (ver Figura 11).

Figura 11. Menú de Mantenimiento de la Base de Datos.

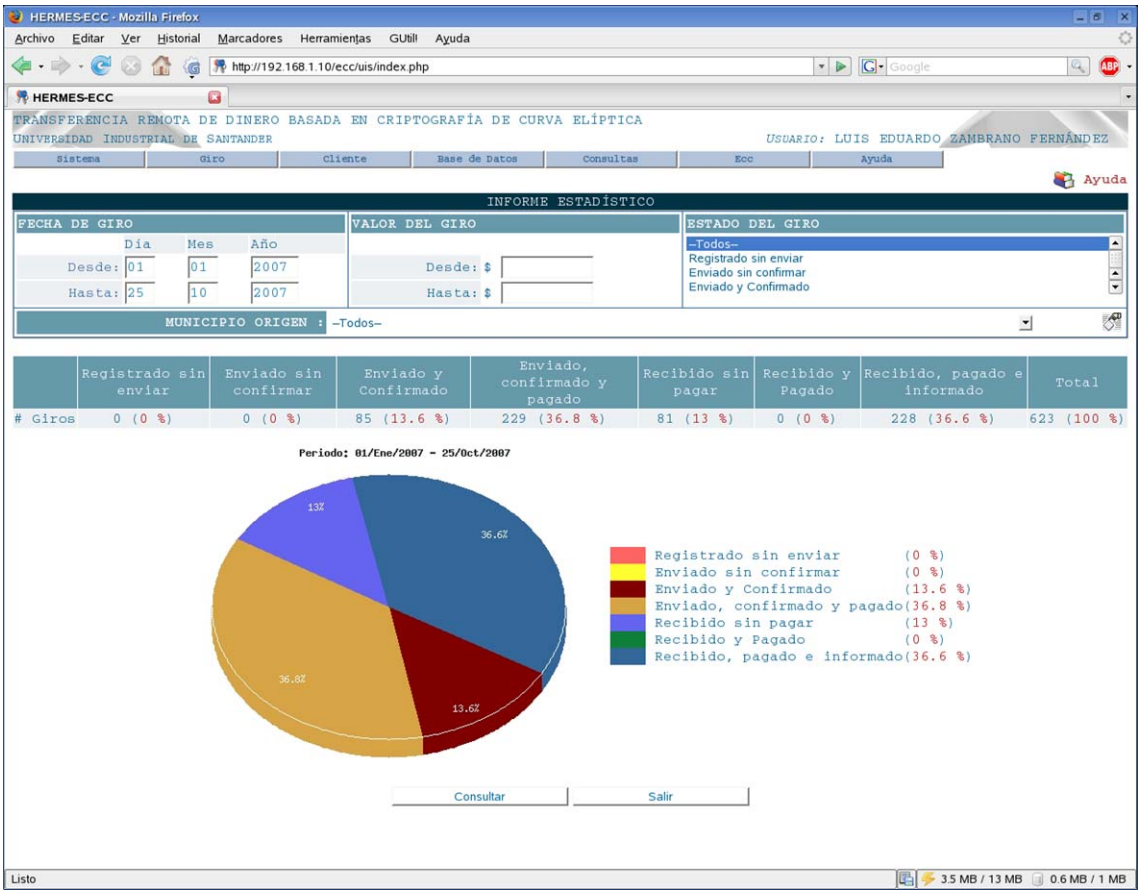
5. Llevar un proceso de auditoría permanente de cada usuario y sus procedimientos en el sistema que implique alterabilidad, acceso o eliminación de registros en la base de datos.

6. Un módulo de *clientes*, que son personas que se acercan a los puntos de giros a enviar dinero de una ciudad a otra y que exigen y demandan que sus datos sean protegidos en todo momento.

Figura 12. Módulo de Clientes.

7. Un módulo de consultas e informes estadísticos que permiten observar el estado actual de la información en todo momento (disponibilidad).

Figura 13. Informe estadístico.



8. Ofrecer una "granja ECC" de algoritmos de curva elíptica que además de soportar la seguridad de la transferencia de la información financiera, le permite a cada usuario del sistema cifrar o firmar cualquier documento que desee proteger o autenticar, o bien, descifrar o verificar la firma de cualquier documento protegido que le sea enviado.

Figura 14. Menú ECC.

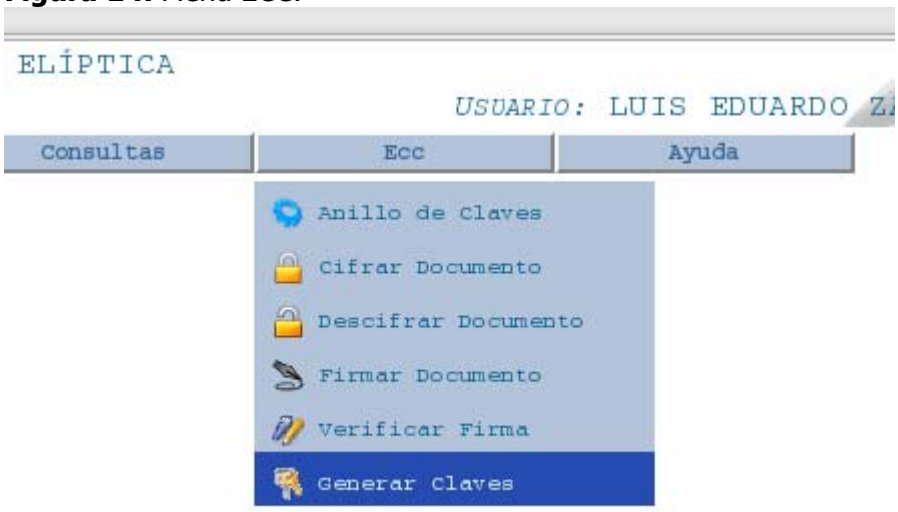
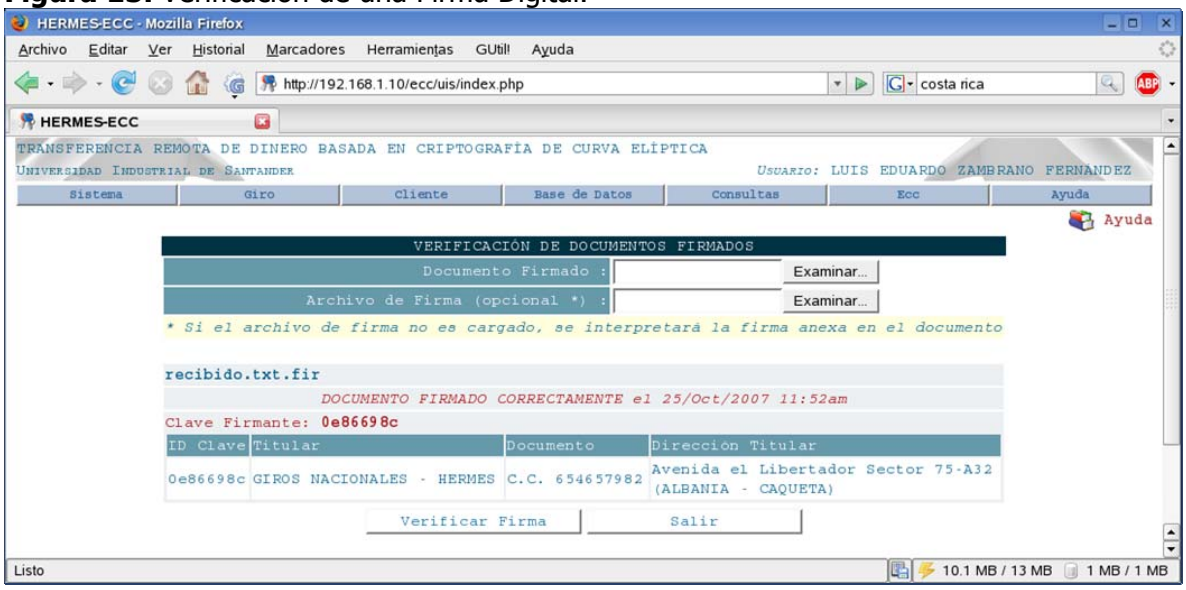
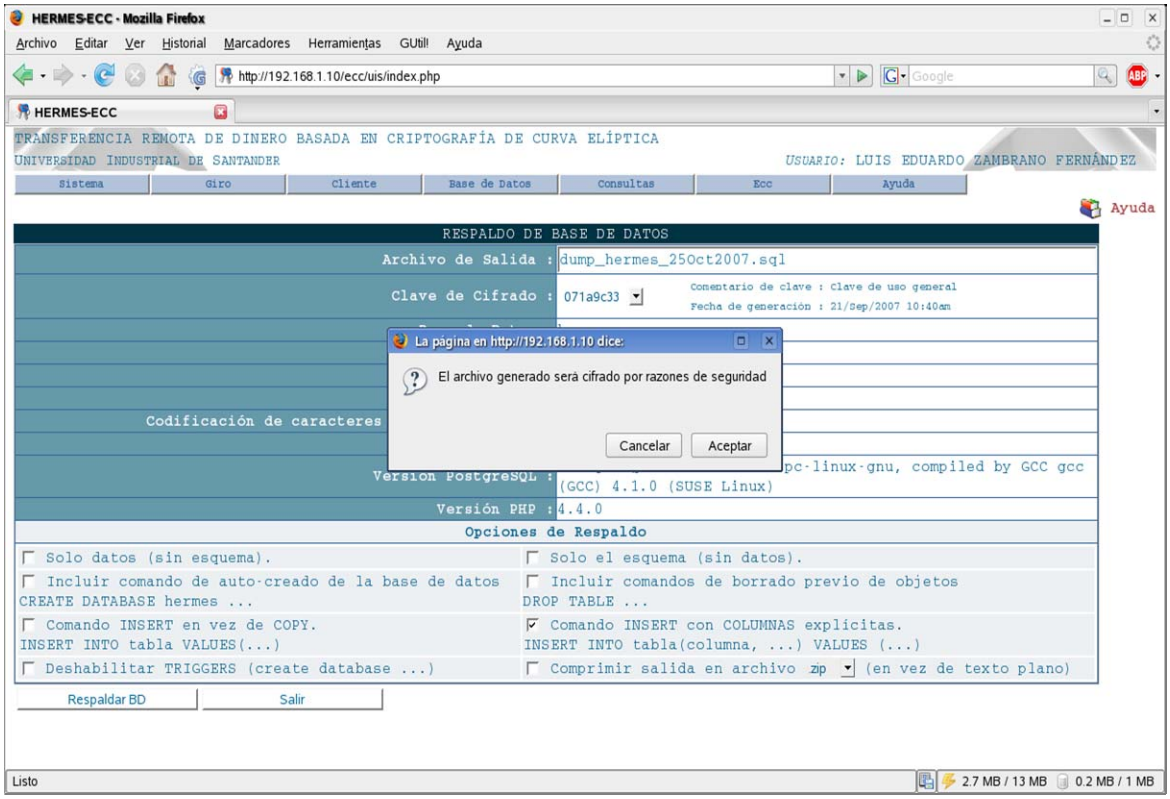


Figura 15. Verificación de una Firma Digital.



9. Una opción de respaldo de la información que permite obtener de forma cifrada un documento con todo el contenido de la base de datos de forma que aun en caso de hurto o copia de este documento por persona no autorizada, la información no se verá comprometida. Solo quien conozca la clave privada (que nunca es almacenada) podrá descifrar el contenido del documento.

Figura 16. Respaldo Cifrado de la Base de Datos.



10. Un centro de informe y reporte de giros que le permiten a cada entidad financiera remitir a otras ciudades los giros que depositan los clientes remitentes y a su vez, informar los pagos de aquellos giros que ya han sido retirados por los clientes destinatarios. Tanto los reportes de giros como los informes de pagos se realizan en documentos cifrados y firmados digitalmente.

Figura 17. Centro de Reporte de Giros e Informe de Pagos.

11. Un módulo de *ayuda en línea* que permite explicar el funcionamiento de cada interfaz del sistema justo en el momento de ser utilizada, también agrupa un manual de usuario de todo el sistema, un glosario de términos importantes y otros documentos de interés.

Figura 18. Ayuda en Línea.

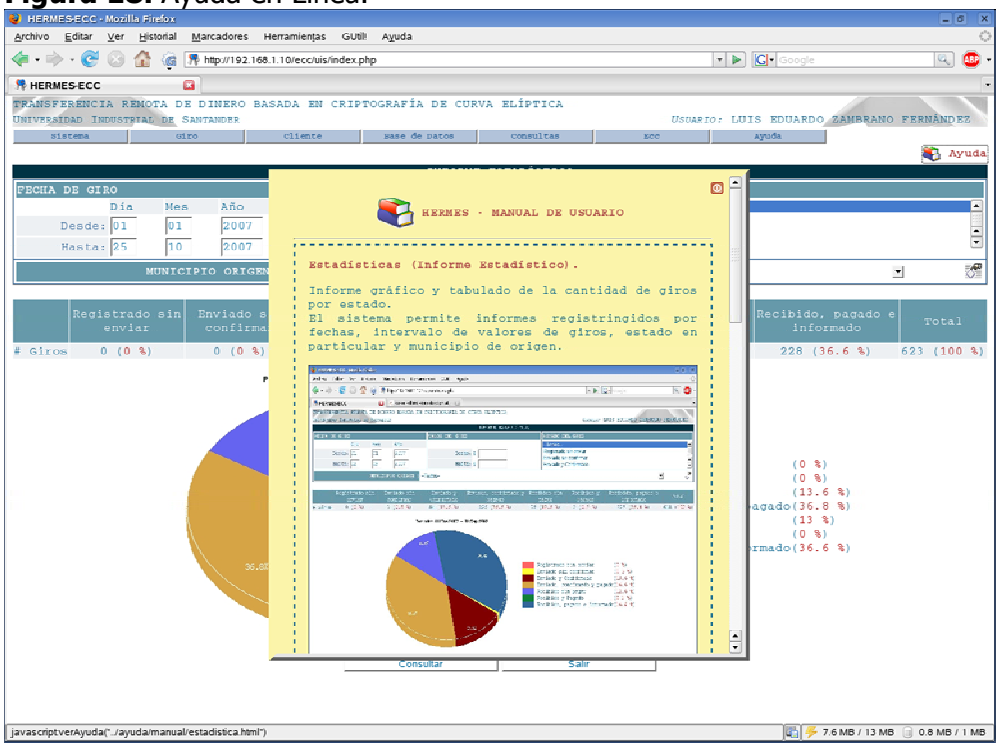
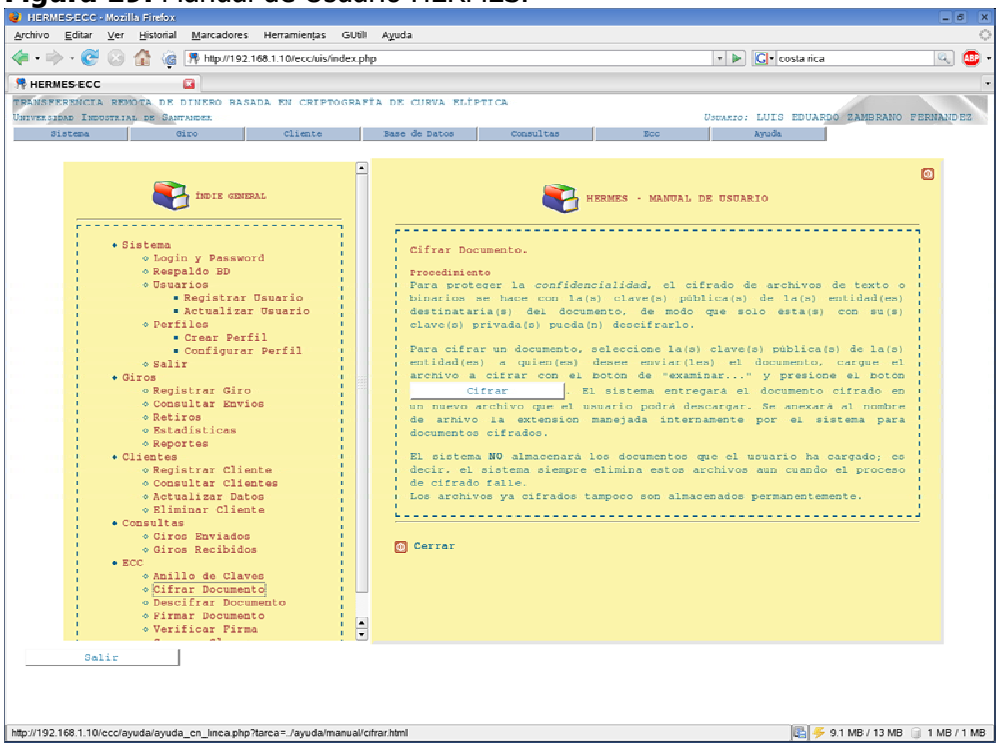
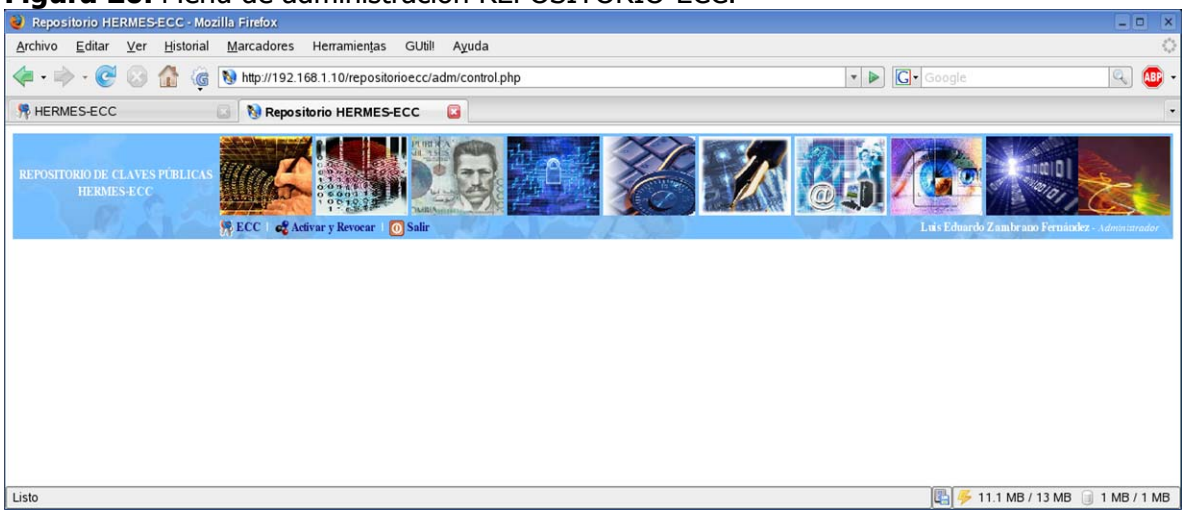


Figura 19. Manual de Usuario HERMES.



- **Criptosistema REPOSITORIO-ECC**

Figura 20. Menú de administración REPOSITORIO-ECC.

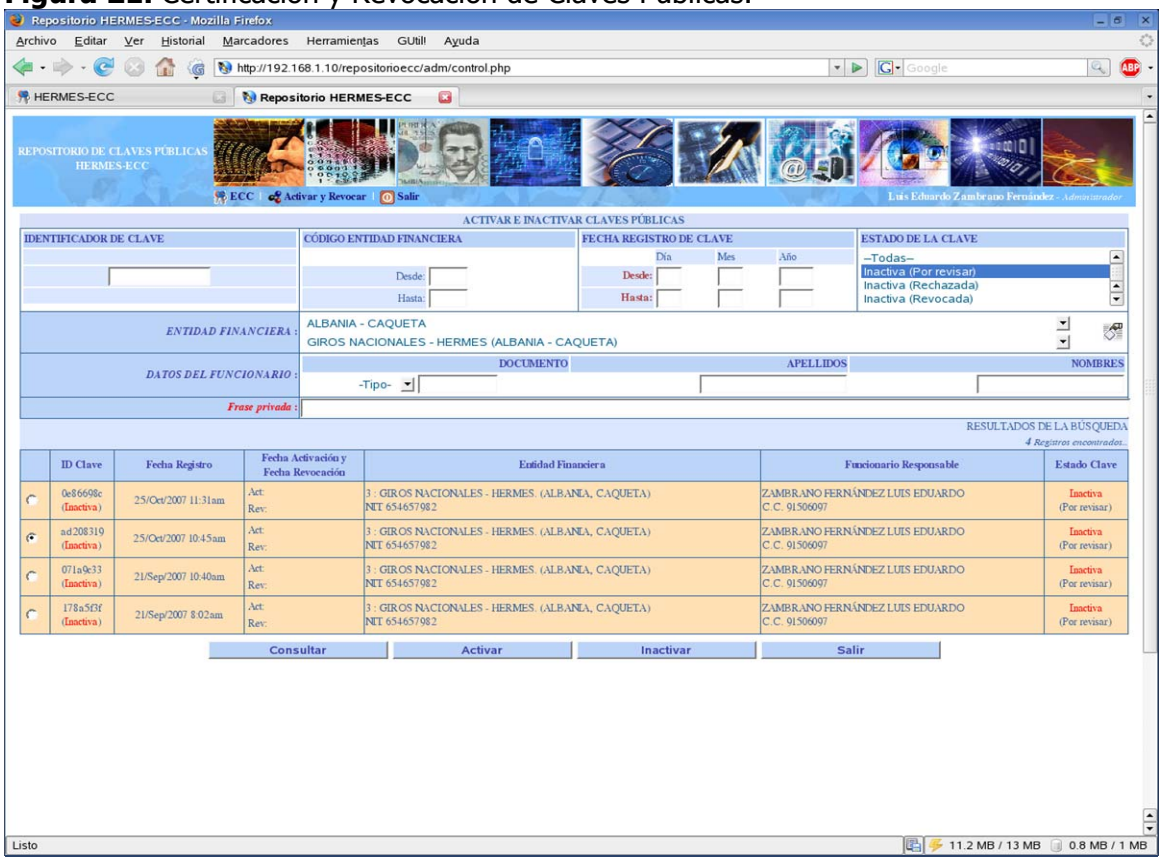


Como parte fundamental de la *infraestructura de clave pública* (PKI) se ha implementado un segundo criptosistema llamado REPOSITORIO-ECC que como hemos dicho actuará como tercero de confianza y se encargará de certificar las claves públicas de las entidades financieras involucradas en el modelo. Los procedimientos que realiza el REPOSITORIO-ECC son:

1. Certificar a través de su firma digital la validez de una clave pública.

2. Revocar las claves públicas cuando estas caduquen o cuando se conozca que la clave privada vinculada a dicha clave pública haya sido comprometida.

Figura 21. Certificación y Revocación de Claves Públicas.



3. Hacer público y de libre acceso a todas las entidades financieras un listado actualizado de las claves públicas revocadas y de las nuevas claves públicas certificadas.

4. Permitir actualizaciones automáticas de su clave en el evento en que el repositorio cambie su clave privada, y actualizaciones automáticas de claves revocadas y nuevas claves activas.

Figura 22. Sincronizaciones Automatizadas con el Repositorio.

5. Autofirmar su propia clave pública de modo que permita a las entidades financieras verificar la validez de dicha clave en todo momento.

Para permitir la transferencia de la información financiera en los *envíos de giros* y los *informes de pagos*, la PKI opera sobre el PROTOCOLO-HERMES, un protocolo que describe los estados y acciones de los distintos procedimientos en la comunicación.

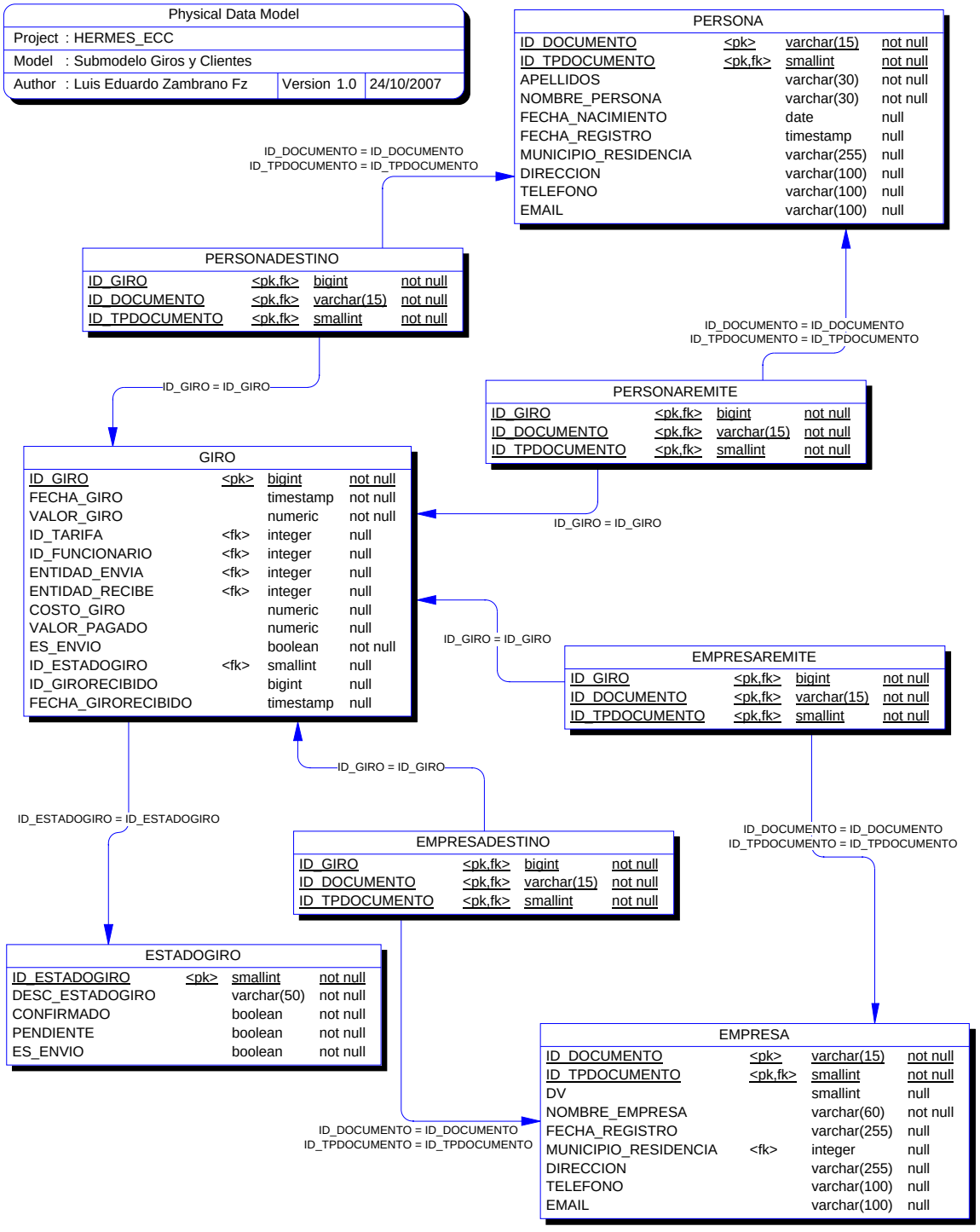
4.3.3 MODELOS DE DATOS

Tanto para el criptosistema HERMES como para el Repositorio-ECC se ha diseñado e implementado un *modelo de datos relacional.* La base de datos para HERMES consta de 37 tablas en total, 47 índices (incluyendo llaves primarias), 2 secuencias, 1 esquema, 1 grupo y 1 usuario. La base de datos para el Repositorio tiene 13 tablas, 14 índices, 1 esquema, 1 grupo, 1 usuario y una secuencia. El *sistema gestor de bases de datos relacional* (RDBMS) para ambos modelos es PostgreSQL.

✓ **Modelo de datos HERMES**

Dado el gran número de tablas del modelo de datos HERMES, se ha divido en varios sub-modelos que facilitan su comprensión. Cada tabla en un sub-modelo contiene cuatro columnas, la primera columna corresponde a los nombres de los campos o atributo de la tabla; la segunda columna muestra el tipo de llave (si aplica), la etiqueta <pk> indica que el campo es una llave primaria (*primary key*) y la etiqueta <fk> indica una llave foránea (*foreign key*); la tercer columna muestra el tipo de de dato y la última columna muestra la condición de nulidad de cada campo. Algunas *tablas padres* a las que son referidas algunas llaves foráneas no se presentan cuando no son relevantes según lo que se desea ilustrar.

Figura 23. Hermes: Submodelo Giros y Clientes



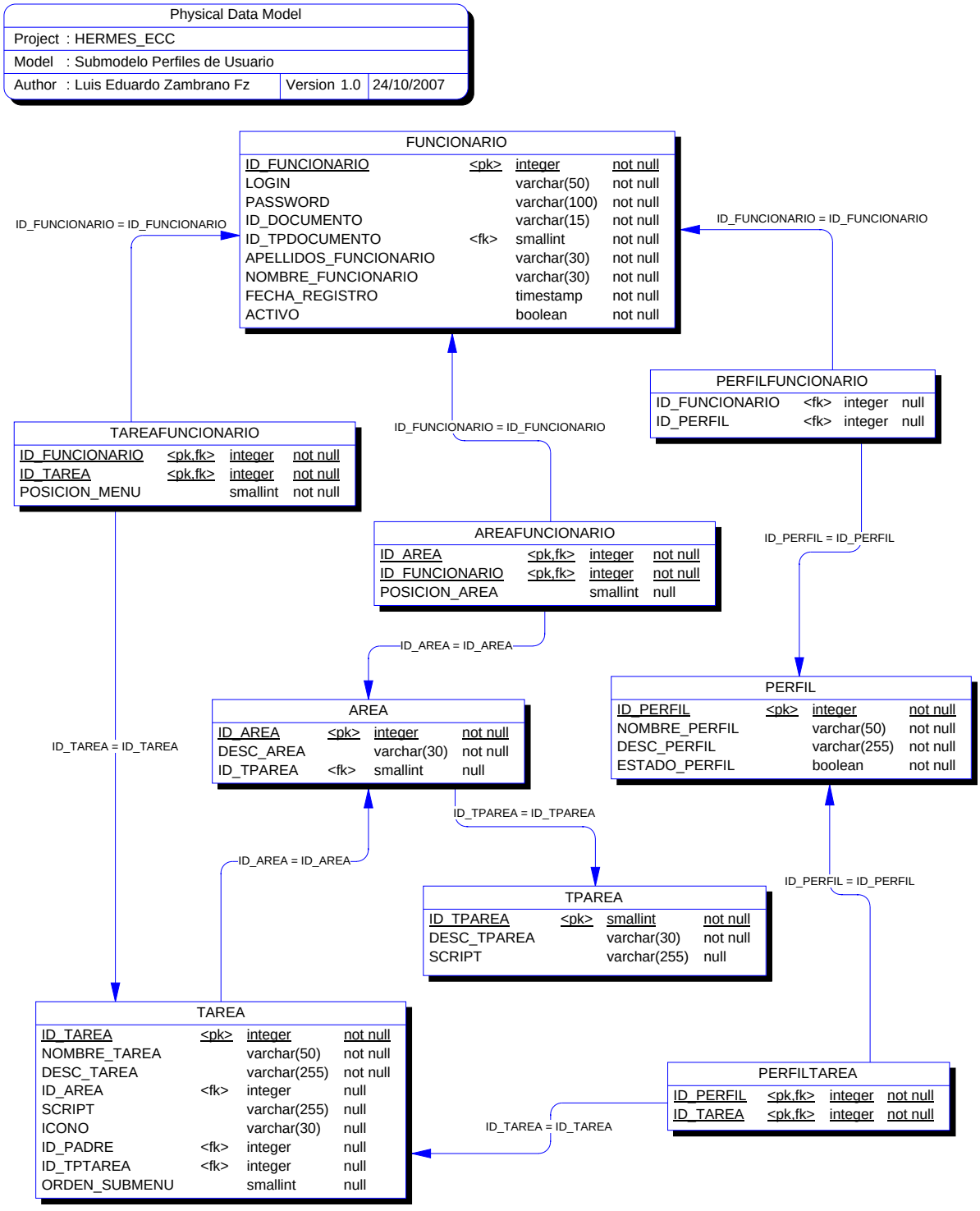
El submodelo *Giros y Clientes* (ver Figura 23.) muestra la relación entre las tablas que almacenan la información de los giros y las tablas que almacenan la información de los clientes. Note que tanto los giros enviados como los giros recibidos por cada entidad

financiera son almacenadas en una misma tabla (tabla GIRO). En la práctica, puede ocurrir que se esté registrando un giro para enviar a otra ciudad destino y en el mismo instante de tiempo se este recibiendo un giro enviado desde una ciudad remota, es decir, puede ocurrir un intento de dos registros simultáneos en la misma tabla. Para manejar este evento de concurrencias, se ha creado una *secuencia*, esta secuencia se asigna como valor por defecto al campo **id_giro** (llave primaria) de la tabla *GIRO* de modo que en cada inserción a la tabla, la secuencia arrojará un valor diferente y consecutivo para cada giro. Con este diseño simple, pero ventajoso a toda vista, se ha superado el problema de control de concurrencias y además, permite a HERMES ser un sistema multiusuario.

Definición. En el sistema, una *tarea* (o *submódulo)* es cada uno de los procedimientos individuales que se realizan para llevar a cabo un caso de uso en particular. Ejemplos de tareas son: *cifrar documento, verificar firma, registrar giro*, etc. Las tareas están agrupadas en áreas (o módulos), ejemplos de áreas son: *Sistema, Giros, ECC, Ayuda*, etc.

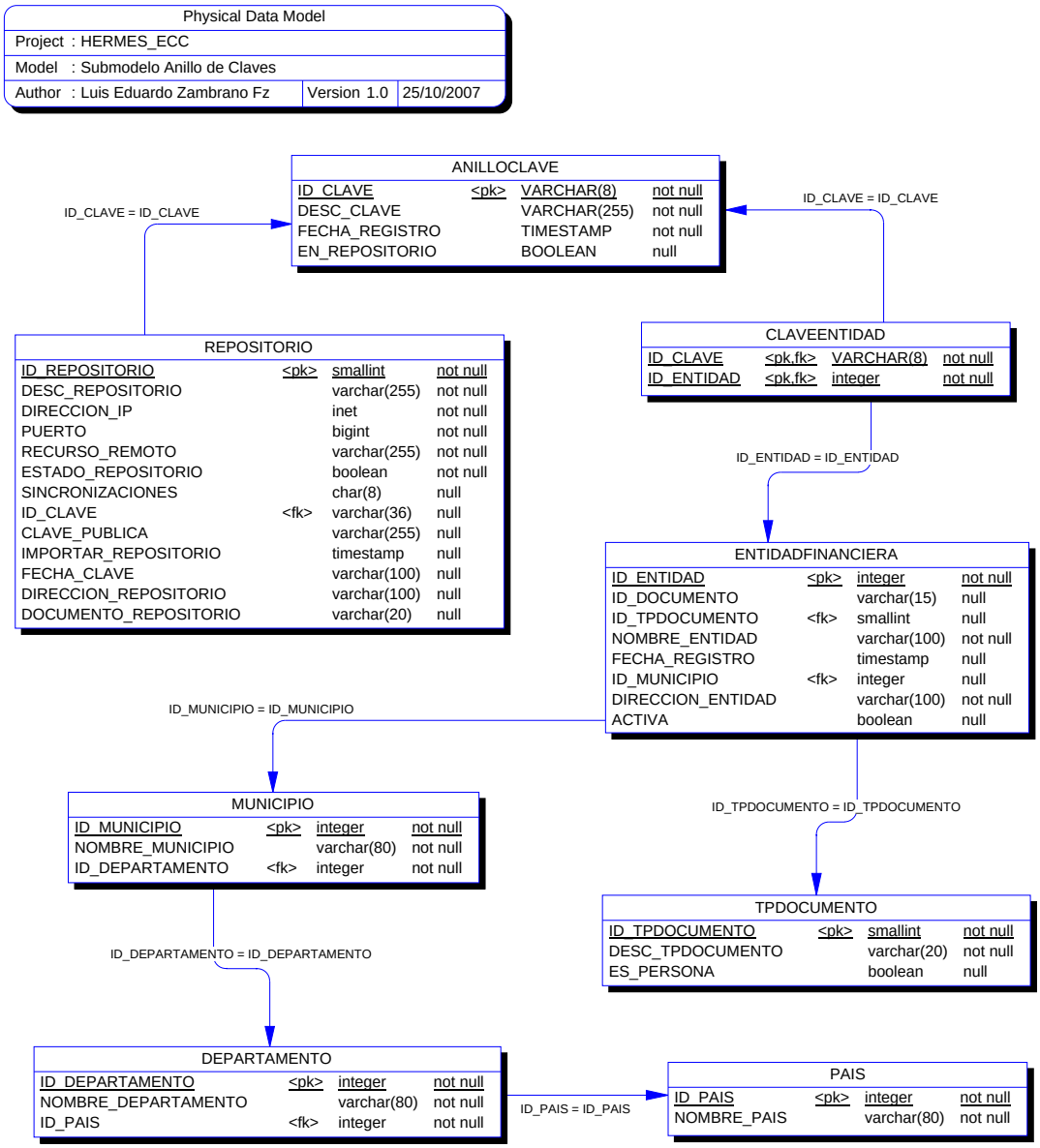
La Figura 24 muestra el modelo entidad-relación de las tablas que soportan el manejo de los perfiles de usuario.

Figura 24. Hermes: Submodelo Perfiles de Usuario.



A través de los perfiles de usuario es que se maneja el control de acceso a los diferentes módulos y submódulos del sistema. Un usuario podrá acceder a una tarea si éste posee algún perfil que agrupe esta tarea (ver tablas PERFILTAREA, PERFILFUNCIONARIO y sus relaciones). Note que, además de la relación por perfiles, existe una relación directa entre las tareas y los funcionarios, y entre éstos y las áreas; esto permite al administrador del sistema hacer excepciones en el acceso a un modulo o submódulo para uno o varios funcionarios, por ejemplo, el administrador podría asignar el mismo perfil de 4 tareas a 6 funcionarios y exceptuar (negar acceso) una tarea a uno de ellos simplemente cortando la relación entre el funcionario y la tarea sin necesidad de excluir la tarea del perfil; lo mismo si se quiere exceptuar un área.

Figura 25. Hermes: Submodelo Anillo de Claves.



El submodelo *Anillo de Claves* de la Figura 25 ilustra la forma de almacenamiento de las claves públicas de las entidades financieras y la información del Repositorio de claves. La relación uno a mucho entre el anillo de claves y las entidades financieras (ver tablas ANILLOCLAVE, ENTIDADFINANCIERA y CLAVEENTIDAD) permite a una entidad financiera manejar más de un par de claves asimétricas (aunque una clave sólo pertenece a una entidad), por ejemplo, podría ocurrir que para cifrar los respaldos de la base de datos, una entidad quiera utilizar una clave distinta a la que utiliza para transferencia de giros; pero no está permitido manejar más de una clave pública para transferencia de giros, el repositorio de claves solo admitirá una clave activa por cada entidad financiera.

La Figura 26 muestra el submodelo de auditoría; se registra auditoría (ver tabla AUDITORIA) a todo evento que realice un usuario del sistema (ver tabla FUNCIONARIO) que implique borrado, inserción o alterabilidad de la información financiera; nuevamente hemos asignado al campo **id_auditoria** de la tabla *AUDITORIA* una secuencia como valor por defecto que permitirá auditar múltiples procesos simultáneos en un ambiente multiusuario. También se auditan los registros de giros recibidos que son procesos automatizados por el sistema, estos registros de giros recibidos se manejan en archivos *log* apartes dado que son alterados por el servidor Web Apache y no por un usuario del sistema.

Figura 26. Hermes: Submodelo Auditoría

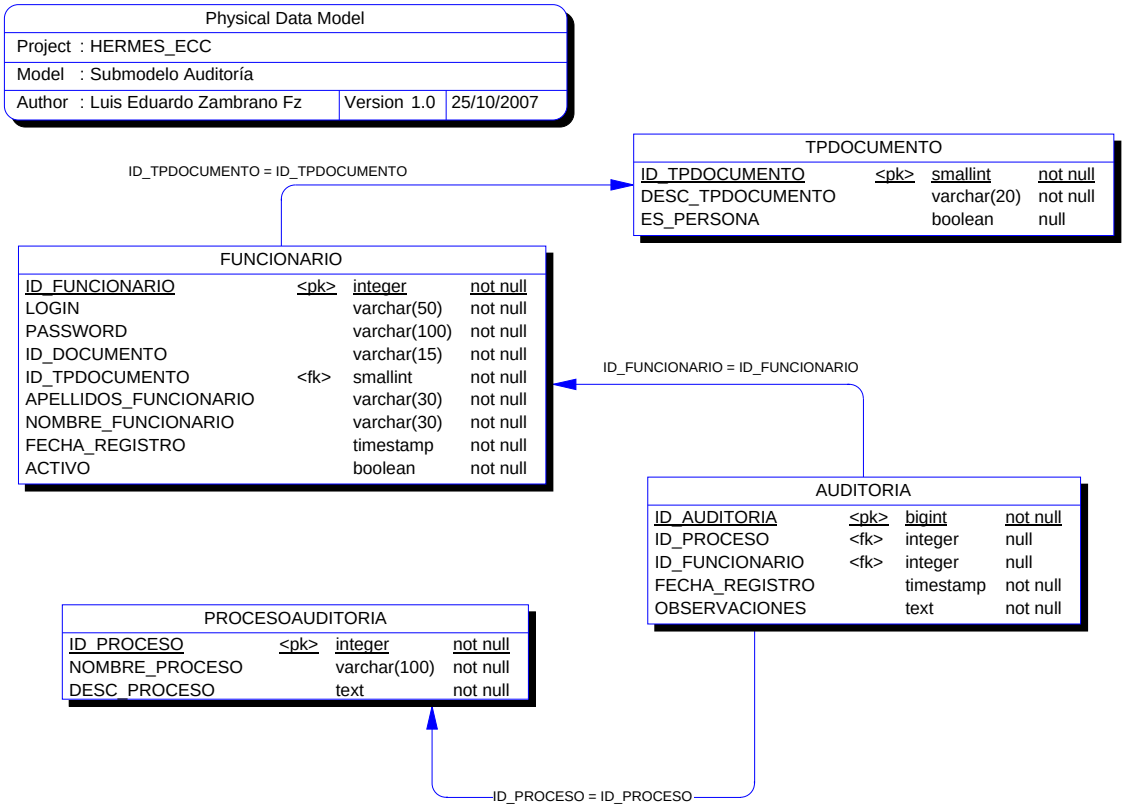
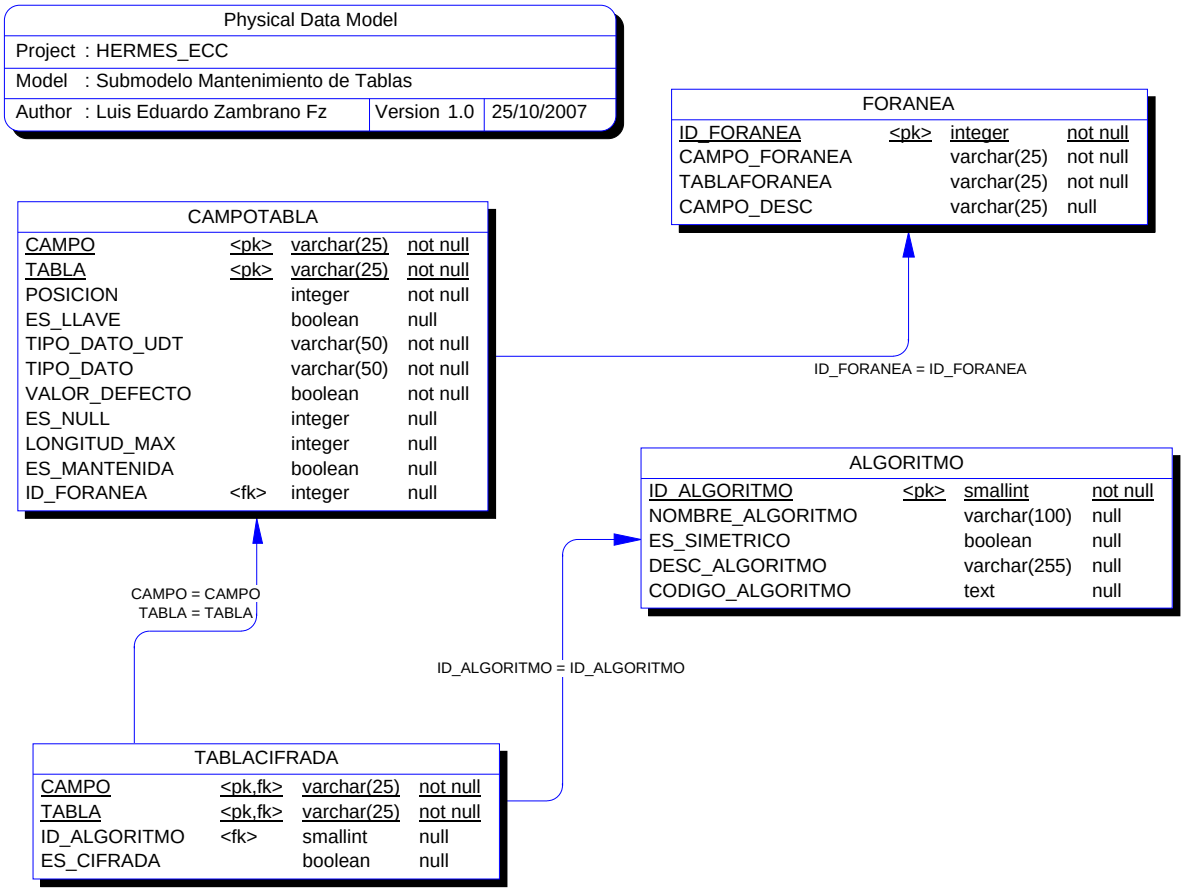


Figura 27. Hermes: Submodelo Mantenimiento de Tablas.



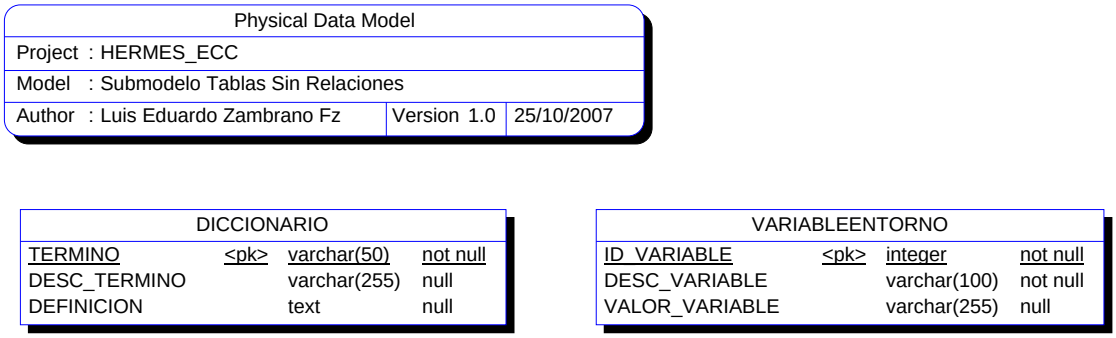
El diagrama entidad relación presentado en la Figura 27, es el submodelo que permite soportar el mantenimiento de todas las tablas del sistema. La tabla CAMPOTABLA contendrá todos los campos de todas las tablas del sistema (inclusive a si misma[41]), note que la llave primaria es la pareja (CAMPO, TABLA) lo que permite almacenar campos homónimos de distintas tablas. El campo booleano **es_mantenida** de esta misma tabla controla si se hace o no mantenimiento de un campo cualquiera, esto permitirá excluir del mantenimiento aquellas tablas que no lo requieran, por ejemplo la tabla GIRO o la tabla AUDITORIA que son alteradas por procesos automáticos del sistema.

[41] N. del A.:Un conjunto *anormal* como diría Bertrand Russell

Aunque el submodelo de datos mostrado en la Figura 28 no es un diagrama entidad-relación, contiene dos tablas muy importantes dentro del sistema. La tabla DICCIONARIO almacena el glosario de términos necesarios para la comprensión de todo el sistema, este glosario de términos es una tarea que se muestra en el menú de ayuda y debe ser consultado por todo funcionario antes de usar HERMES por primera vez.

Como su nombre lo indica, la tabla VARIABLEENTORNO almacena las variables cuyos valores son utilizados por el sistema en el entorno donde opera. Una variable de entorno consta de un identificador único, un nombre de variable y su valor. Ejemplos de valores que se almacenan en las variables de entornos son: el nombre del sistema, el municipio donde está la entidad financiera, las extensiones de archivos que usa el criptosistema en la generación de los documentos cifrados o firmados, etc.

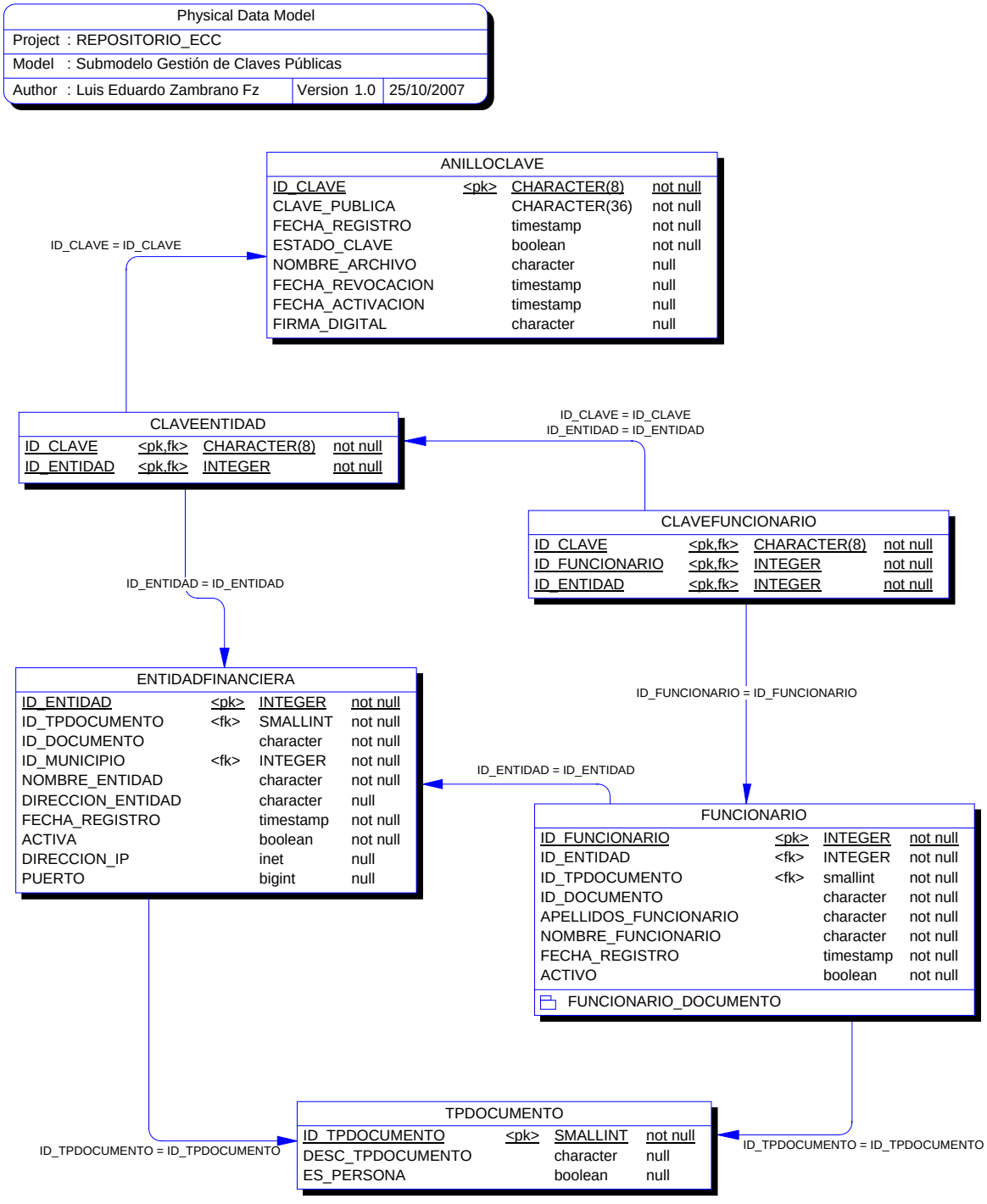
Figura 28. Hermes: Submodelo Tablas Sin Relaciones.



✓ Modelo de datos REPOSITORIO-ECC

El modelo de datos del REPOSITORIO-ECC lo presentaremos en 2 submodelos, en la Figura 29 mostraremos el diagrama entidad relación que soporta la gestión de las claves públicas en el repositorio, y en la Figura 30 presentamos el diagrama entidad relación que soporta el proceso de auditoria.

Figura 29. Repositorio: Submodelo Gestión de Claves Públicas.

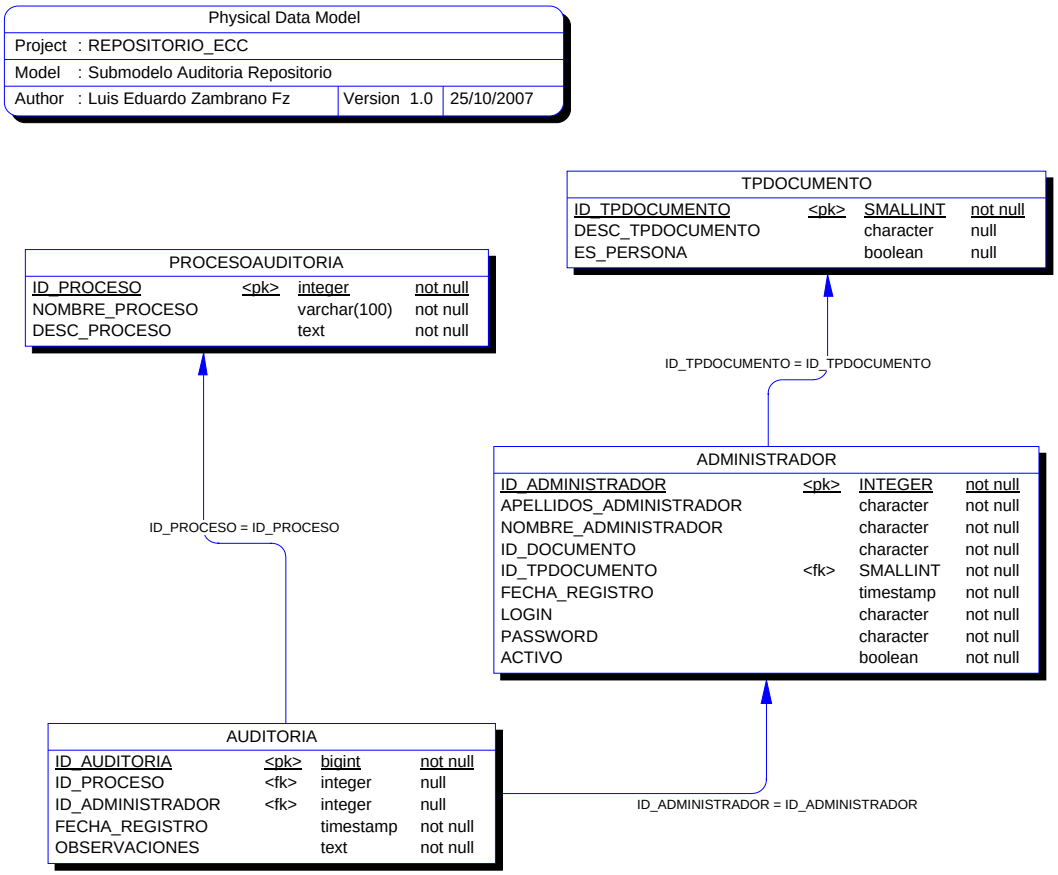


Observe que se relaciona tanto la entidad financiera con su clave pública como el funcionario que reporta dicha clave, esto es una medida de control que permite determinar responsabilidades en el proceso y a la vez es necesaria para la certificación de las claves, pues es con el funcionario con quien se debe constatar que la clave pública pertenece a una entidad financiera, bien con un procedimiento de presentación personal, o bien con el envío de la clave previamente y una verificación posterior con cualquier otro medio *razonablemente* seguro.

Observe también que en contraste al anillo de claves del criptosistema HERMES, el anillo de claves en el repositorio maneja el estado de las claves y la firma del repositorio que certifica las claves activas (ver campo **estado_clave** y **firma_digital** de la tabla ANILLOCLAVE), también se manejan las fechas de activación y revocación datos muy importantes en las listas de claves activadas y revocadas que se consultan por todas las entidades financieras.

Por último, en la Figura 30 se ilustra el diagrama entidad-relación del proceso de auditoría. El modelo de datos soporta un ambiente multiusuario pero para el repositorio no se recomienda esta práctica, el tercero de confianza debe tener un único administrador responsable, como un notario electrónico.

Figura 30. Repositorio: Submodelo Auditoría Repositorio



4.3.4 PARÁMETROS DEL DOMINIO DE CURVA ELÍPTICA.

Los parámetros del dominio elegidos para la implementación los criptosistema es la séptupla:

$$T = (m, f(x), a, b, G, n, h)$$

Donde,

$m = 191$ (primo) que define el campo finito $F_{2^{191}}$

$f(x) = x^{191} + x^9 + 1$ (un polinomio binario irreducible en F_2 de grado 191).

$a = 1$. Note que $a \in F_{2^{191}}$

$b = (36F815F10D48986590ECD1E78C5616DF3C540ACA70BBA6A0)_{16}$
Note que $b \in F_{2^{191}}$ es un polinomio escrito en representación hexadecimal para una mejor lectura. Note además que a y b definen la curva elíptica:

$$E: y^2 + xy = x^3 + ax^2 + b$$

$G = (G_x, G_y) \in E(F_{2^m})$ es un punto base de la curva. Las coordenadas de G son polinomios que se presentan en base hexadecimal:

$G_x = (4BFE41F4DE99C616B89A0D3D9C70AB4230E8A2CFCCFAC421)_{16}$
$G_y = (677278B6405D3AC85D5134AB6EB6A3702BC90728BB3BE851)_{16}$

$n =$ 1569275433846670190958947355765639058464984869317920351161
n es un número primo (base decimal) que corresponde al orden de G.

$h = 2$ es el cofactor de $\#E(F_{2^m})$, $\#E(F_{2^m}) = n \cdot h$.

Comentario. Los parámetros del dominio se han elegido de tal forma que cumplan las condiciones que hemos visto para curvas elípticas apropiadas para fines criptográficos. Así, observe que estos parámetros del dominio cumplen que:

(i) $b \neq 0$ en $F_{2^{191}}$.

(ii) $\#E(F_{2^{191}}) =$ 3138550867693340381917894711531278116929969738635840702322

$\neq$

2^{191} = 3138550867693340381917894711603833208051177722232017256448.

(iii) $2^{191.B} \neq 1$ (mód n) verificado para todo $1 \leq B \leq 45$ (Lo mínimo es hasta 20).

1. $h = 2 \leq 4$.

Además, se ha encontrado un trinomio como polinomio primitivo $f(x)$ que harán más eficiente la aritmética sobre puntos de la curva (comparado con un pentanomio).

Cabe mencionar que se ha hecho caso omiso a las sugerencias hechas por los organismos emisores de estándares con respecto a la elección del polinomio primitivo y el número primo m que debe definir F_{2^m}, pues en nuestro caso, tanto en el criptosistema HERMES como en el REPOSITORIO-ECC manejarán los mismos parámetros del dominio con lo que la interoperabilidad está garantizada.

4.3.5 GENERACIÓN DE CLAVES PARA CURVAS ELÍPTICAS

Dado los parámetros de dominio de curva elíptica $T = (p, a, b, G, n, h)$ o $T = (m, f(x), a, b, G, n, h)$, un par de claves de curva elíptica (k, P) asociado con T consiste de una clave secreta de curva elíptica k la cual es un entero en el intervalo $[1, n-1]$, y una clave pública de curva elíptica $P = (P_x, P_y)$ la cual es el punto $P = k \cdot G$.

Para generar un par de claves, una entidad procede como sigue, dados los parámetros (válidos) de dominio $T = (p, a, b, G, n, h)$ o $T = (m, f(x), a, b, G, n, h)$ según corresponda:

1. Aleatoriamente o pseudo aleatoriamente seleccionar un número entero k en el intervalo $[1, n - 1]$.
2. Calcular $P = k \cdot G$.
3. Retornar (k, P).

4.3.6 Validación de Claves Públicas

Definición. Una clave pública de curva elíptica P se dice que es *parcialmente valida* si P es un punto sobre la curva elíptica asociada pero no necesariamente es cierto que $P = kG$ para algún entero k.

Para comprobar que la clave pública que una entidad quiere utilizar es válida, puede proceder a ejecutar una serie de tests sobre la clave pública (usando los parámetros de dominio) para validar que la clave pública efectivamente fue derivada correctamente.

En nuestro sistema, una clave pública estará firmada por el REPOSITORIO-ECC que actúa como Autoridad Certificadora confiable, por lo que la validación de las claves públicas consiste en verificar la firma digital que el repositorio emite para cada clave.

Sin embargo, en ausencia de Autoridades Certificadoras, se podría implementar la *validación parcial* de claves públicas, este procedimiento consiste en chequear (1) que P = (P_x, P_y) ≠ *O*, (2) que P_x, P_y ∈ F_{2^m} sean polinomios de grado a lo sumo *m* - 1 y (3) que P ∈ *E*(F_{2^m}).

4.3.7 Primitivas Difie-Hellman de Curvas Elípticas

Las primitivas Difie-Hellman de curvas elípticas son la base para operación del ECAES (Elliptic Curve Augmented Encryption Scheme), y el Esquema Difie-Hellman de curvas elípticas.

Se especifican dos primitivas: la *primitiva Difie-Hellman de curva elíptica* y la *primitiva Difie-Hellman de cofactor de curva elíptica*. La idea básica de ambos esquemas es la misma -generar un valor secreto compartido a partir de una clave privada adueñada por una entidad *U* y una clave pública adueñada por otra entidad *V* de manera que si ambas entidades ejecutan la primitiva simultáneamente con las claves correspondientes, podrán recuperar el mismo valor secreto compartido-. Sin embargo, las dos primitivas son sutilmente diferentes: la primitiva Difie-Hellman de curva elíptica es el análogo directo del bien conocido protocolo de acuerdo de claves Difie-Hellman (*definida mas adelante*), mientras que la primitiva Difie-Hellman de cofactor de curva elíptica incorpora el cofactor *h* en el calculo del valor secreto compartido para proveer una resistencia eficiente contra ataques de subgrupo pequeño.

A partir de estas definiciones, podemos ver que la construcción de un protocolo de acuerdo de claves Difie-Hellman basado en ECC es directa:

4.3.7.1 Primitiva Difie-Hellman de Curva Elíptica: Para que una entidad *U* pueda calcular un valor secreto compartido con la entidad *V* usando la primitiva Difie-Hellman de curva elíptica, se debe ejecutar el proceso siguiente, que toma como entrada parámetros validados de curva elíptica *T* = (*p, a, b, G, n, h*) o *T* = (*m, f*(*x*)*, a, b, G, n, h*), la clave privada d_U asociada con *T* perteneciente a *U*, y una clave pública Q_V (al menos parcialmente validada) asociada con *T* perteneciente a *V*.

1. Calcular el punto $P = (x_P , y_P) = d_U Q_V$ de la curva elíptica.
2. Chequear que *P* ≠ *O*. Si *P* = *O*, retornar "invalido" y parar.
3. retornar $z = x_P$, el elemento del campo finito que es secreto y compartido.

4.3.7.2 Primitiva Difie-Hellman de Cofactor de Curva Elíptica: El cálculo del valor secreto para esta primitiva es esencialmente el mismo que para la primitiva Difie-Hellman descripta anteriormente. La única diferencia radica en el paso de la multiplicación escalar

(paso 1). En vez de solo multiplicar por la clave privada d_U, la entidad U ahora debe calcular el punto $P = (x_P , y_P) = hd_UQ_V$.

4.3.8 ESQUEMAS DE FIRMA DIGITAL

Para el diseño de nuestros criptosistemas hemos intentado seguir los estándares para firmado digital, en particular las recomendaciones del NIST para el estándar de DSS (Digital Signature Standard), que describe ahora tanto el algoritmo de firma digital DSA como también el ECDSA. Sin embargo, se han hecho leves modificaciones, como el cambio de la función resumen SHA-1 por Tiger.

El algoritmo DSA es una variante del esquema de firmas de ElGamal. Este explota pequeños subgrupos en Z^*_p de manera de reducir el tamaño de las firmas producidas. A continuación se describen los procedimientos de generación de claves, generación de firma y verificación para el algoritmo DSA:

4.3.8.1 Generación de claves DSA. Cada entidad A hace lo siguiente:

1. Seleccionar un número primo q tal que $2^{159} < q < 2^{160}$. (q es un primo de 160 bits.)
2. Seleccionar un número primo p de 1024 bits con la propiedad que $q \mid p - 1$. (El estándar DSS obliga a que p sea un primo tal que $2^{511+64t} < p < 2^{512+64t}$ donde $0 \leq t \leq 8$. Si $t = 8$ entonces p es un primo de 1024 bits.)
3. Seleccionar un elemento $h \in Zp^*$ y calcular $g = h^{(p-1)/q}$ mód p, repetir hasta que $g \neq 1$. (g es un generador del único subgrupo cíclico de orden q en Z^*_p.)
4. Seleccionar aleatoriamente un entero x en el intervalo $[1, q - 1]$.
5. Calcular $y = g^x$ mód p.
6. La clave pública de A es (p, q, g, y). La clave privada de A es x.

4.3.8.2 Generación de una firma en DSA. Para firmar un mensaje m, A hace los siguiente:

1. Seleccionar aleatoriamente un entero k en el intervalo $[1, q - 1]$.
2. Calcular $r = (g^k$ mód $p)$ mód q.
3. Calcular k^{-1} mód q.
4. Calcular $s = k^{-1}\{h(m) + xr\}$ mód q, donde h es el Secure Hash Algorithm (SHA-1).
5. Si $s = 0$ entonces volver al paso 1. (Si $s = 0$, entonces s^{-1} mód q no existe, s^{-1} es requerido en el paso 2 de la verificación de la firma.)
6. El firma para el mensaje m es el par de enteros (r, s).

4.3.8.3 Verificación de una firma en DSA. Para verificar la firma (r, s) de A sobre el mensaje m, B debería hacer lo siguiente:

1. Obtener una copia autentica de la clave pública (p, q, g, y) de A.
2. Verificar que r y s sean enteros en el intervalo $[1, q - 1]$.
3. Calcular $w = s^{-1}$ mód q y $h(m)$. (h es SHA-1.)
4. Calcular $u_1 = h(m)w$ mód q y $u_2 = rw$ mód q.
5. Calcular $v = (g^{u1}y^{u2}$ mód $p)$ mód q.

6. Aceptar la firma si y solo si $v = r$.

Ya que r y s son ambos enteros menores que q, las firmas DSA son de 320 bits de longitud. La seguridad de DSA descansa sobre dos problemas de logaritmos discretos distintos, pero relacionados. Uno es el PLD donde el algoritmo Number Field Sieve (NFS) es aplicable (este algoritmo tiene un tiempo sub-exponencial). Si p es un primo de 1024 bits, entonces DSA no es vulnerable a un ataque con el NFS.

El segundo problema de logaritmo discreto trabaja a la base g: Dados p, q, g e y, encontrar x tal que $y = g^x$ (mód p). Para p grande (por ej. 1024 bits), el mejor algoritmo conocido para este problema es el Pollard-ρ, y toma alrededor de $\sqrt{(\pi q / 2)}$ pasos. Si $q \approx 2^{160}$ entonces $\sqrt{(\pi q / 2)}$ representa un monto de computación inviable, luego el DSA no es vulnerable a este ataque.

4.3.9 ELLIPTIC CURVE DSA (ECDSA): ECDSA es el análogo sobre curvas elípticas al DSA. Esto es, en vez de trabajar sobre el subgrupo de orden q en $\mathbf{Z}^*_p$, se trabaja en un grupo de curva elíptica $E(\mathbf{Z}_p)$. Como dijimos anteriormente, el ECDSA fue estandarizado por el NIST, pero también están en este proceso los comités de estándares ANSI X9F1 e IEEE P1363.

En concreto, el ECDSA es un esquema de firma basado en ECC. Está diseñado para ser existencialmente infalsificable, aún ante la presencia de un adversario capaz de lanzar ataques de mensajes elegidos por él (es decir, un chosenmessage attack).

Definición: Un esquema de firma es *existencialmente infalsificable* si es inviable para un adversario falsificar una firma sobre algún mensaje que no ha sido previamente firmado por el usuario legítimo del esquema.

Existe un interés especial en este algoritmo, puesto que su estructura es muy similar al DSA, el cual ha sido muy estudiado estos últimos años. Menos interés ha habido en estandarizar un análogo sobre curvas elípticas del esquema de Schnorr, o el de Nyberg-Rueppel, y es por eso que lo omitimos.

A continuación se dan los procedimientos para generar los pares claves, generar una firma y la verificación de una firma usando ECDSA:

4.3.9.1 Generación de claves ECDSA: Cada entidad A hace lo siguiente:

1. Seleccionar una curva elíptica E definida sobre Zp*. El número de puntos en $E(Zp)$ debería ser divisible por un número primo grande n.
2. Seleccionar un punto $P \in E(Zp)$ de orden n.
3. Seleccionar un entero d estadísticamente único e impredecible en el intervalo [1, n _ 1].
4. Calcular $Q = dP$.
5. La clave pública de A es (E, P, n,Q), la clave privada de A es d.

4.3.9.2 Generación de una firma ECDSA: Para firmar un mensaje *m*, *A* hace lo siguiente:

1. Seleccionar un entero *k* estadísticamente único e impredecible en el intervalo [1*, n* - 1].
2. Calcular $kP = (x_1, y_1)$ y $r = x_1$ mód *n*. (Aquí x_1 es tratado como a un número entero, por ejemplo por conversión de su representación binaria). Si $r = 0$, entonces volver al paso 1. (Esta es una condición de seguridad: si $r = 0$, entonces la ecuación de firma $s = k^{-1}\{h(m) + dr\}$ mód *n* no involucra a la clave privada *d*!)
3. Calcular k^{-1} mód *n*.
4. Calcular $s = k^{-1}\{h(m) + dr\}$ mód *n*, donde *h* es el Secure Hash Algorithm (SHA-1).
5. Si $s = 0$, entonces volver al paso 1. (Si $s = 0$ entonces s^{-1} mód *n* no existe, s^{-1} es requerido en el paso 2 de la verificación de firma.)
6. La firma para el mensaje *m* es el par de enteros (*r, s*).

4.3.9.3 Verificación de firma ECDSA: Para verificar la firma (*r, s*) de *A* sobre el mensaje *m*, *B* debería hacer:

1. Obtener una copia autentica de la clave pública (*E, P, n,Q*) de *A*.
2. Verificar que *r* y *s* sean enteros en el intervalo [1*, n* - 1]. Si alguno no esta en el intervalo [1*, n* - 1] retornar "invalida" y parar.
3. Calcular $w = s^{-1}$ mód *n* y *h*(*m*).
4. Calcular $u_1 = h(m)w$ mód *n* y $u_2 = rw$ mód *n*.
5. Calcular $R = u_1P + u_2Q = (x_0, y_0)$ y $v = x_0$ mód *n*. Si $R = O$, retornar "invalida" y parar.
6. Aceptar la firma si y solo si $v = r$. Si $v \neq r$ retornar "invalida".

El estándar ANSI X9.62 obliga a que $n > 2^{160}$. Para obtener un nivel de seguridad similar al de DSA (con *q* de 160 bits y *p* de 1024 bits), el parámetro *n* debería tener alrededor de 160 bits. En este caso, las firmas producidas por DSA y ECDSA tienen la misma longitud de 320 bits.

En vez de que cada entidad genere su propia curva elíptica, las entidades podrán elegir usar la misma curva *E* sobre $\mathbf{Z}_p$, y punto *P* de orden *n*; estas cantidades luego son llamadas *parámetros del sistema*. (En DSA, los parámetros de sistema análogos serian *p*, *q* y *g*.) En este caso, la clave pública de una entidad consiste solamente del punto *Q*. Esto resulta en claves públicas de menor tamaño.

ECDSA tiene un número de similitudes con DSA, de las cuales las más importantes son:

1. Ambos algoritmos están basados en el esquema de firmas de ElGamal y usan la misma ecuación de firma: $s = k^{-1}\{h(m) + dr\}$ mód *n*.
2. En ambos algoritmos, los valores que son difíciles de generar son los parámetros del sistema (*p*, *q* y *g* para DSA; *p*, *E*, *P* y *n* para ECDSA) los cuales son públicos; su generación puede ser auditada y chequeada por validez en forma independiente. Esto ayuda a mostrar que no fueron producidos para reunir ningún criterio secreto (por Ej. proveer a los parámetros con alguna propiedad "trapdoor"). Generar una clave privada, dado un conjunto de parámetros de sistema es relativamente simple y generar la clave pública asociada es directo. Contraste esto con el algoritmo RSA,

donde los valores que son difíciles de generar (los primos p y q) deben ser mantenidos secretos.

3. En su versión actual, tanto DSA como ECDSA usan SHA-1 como única opción para la función de hash. Sin embargo, esta planeado el soporte para SHA-2 a medida que SHA-2 pase por los procesos de estandarización de ANSI y NIST.

No obstante, también existen algunas diferencias entre DSA y ECDSA, a saber:

1. La clave privada d y el valor efímero k generado por cada firma en ECDSA están definidos para que sean *estadísticamente únicos e impredecibles* en vez de sencillamente *aleatorio* como en DSA. Esta es una clarificación importante y es una mejor definición de los requerimientos de seguridad. Si k puede ser determinado o si k se repite entonces un adversario puede recuperar d, la clave privada. Por supuesto, el uso de un valor aleatorio esta explícitamente siendo permitido; no obstante arquitecturalmente es preferible definir los requerimientos en vez de decir una forma en particular de reunir los requerimientos. Por Ej., dar los requerimientos permite a una implementación de alta seguridad filtrar los valores k para asegurar que no se repitan. Esta posibilidad no esta permitida si k es requerido que sea aleatorio. También, definir los requerimientos da más guía a los implementadores y usuarios respecto a qué constituye una preocupación de seguridad.
2. En ECDSA, un método llamado *compresión de punto* permite a un punto sobre la curva elíptica (por ej., una clave pública Q) ser representado en forma compacta por un elemento del campo y un bit adicional, en vez de dos elementos del campo[42]. Luego, por Ej., si $p \approx 2^{160}$ (de manera que los elementos en $\mathbf{Z}_p$ son cadenas de 160 bits), las claves públicas pueden ser representadas como cadenas de 161 bits. Esto puede llevar a una reducción sustancial en el tamaño de un certificado de clave pública en el orden de un 25% cuando es comparado con otros algoritmos asimétricos.
3. En el DSA, hay un chequeo de límites opcional durante la generación de la firma sobre los componentes de la firma digital (r y s). Esto resulta en una probabilidad extremadamente baja que un sistema conformista el cual no haga el chequeo de límites genere una firma que no se pudiera verificar. Sin embargo, esto significa que código escrito genéricamente (es decir, código que no tome en cuenta conocimiento de la implementación subyacente de DSA) debe verificar la firma que generó el mismo, si es que debe haber un 100% de garantía que la firma se pueda verificar. El chequeo de límites análogo ha sido hecho obligatorio en ECDSA; el 100% de las firmas digitales que son generadas se pueden verificar. Note, no obstante, que aún podría haber situaciones donde es una sabia decisión verificar explícitamente una firma recién generada, tal como cuando se sabe que la firma será distribuida ampliamente. La verificación explícita ayudará a detectar implementaciones dudosas y errores de aplicación.
4. La prueba de primalidad en DSA es probabilística. Como la tecnología de curva elíptica tiene la capacidad para una prueba de primalidad determinística, tal prueba fue incluida como una opción en el estándar X9.62, mientras que también retiene la prueba probabilística. Esto le permite a una aplicación con altos requerimientos de

[42] Esto se puede hacer debido a que toda curva elíptica es simétrica con respecto al eje x, entonces solo se representa la coordenada x del punto; la coordenada y correspondiente es calculada en vez de almacenada. El bit extra es para indicar si el punto correcto está del lado positivo del eje y, o del lado negativo del eje y.

seguridad (por Ej., un Certificate Authority) verificar que los valores primos presentados son en realidad primos, aunque a un costo computacional adicional.

5. ECDSA especifica los pasos de un procedimiento para verificar los parámetros del sistema generados, mientras que DSA no. En un escenario común, uno recibiría los parámetros del sistema DSA de una Tercer Parte Confiable (TTP) donde no hay necesidad de validar los parámetros del sistema. Sin embargo, otro escenario útil es donde una entidad ha generado sus propios parámetros personales del sistema. En el último escenario uno podría querer verificar que los parámetros del sistema provisto realmente reúnan todos los requerimientos de seguridad antes de usarlos. Tal procedimiento de validación de parámetros del sistema podría ser agregado a DSA.
6. Se han demostrado debilidades teóricas en DSA basadas en el hecho que la función de hash realmente usada en DSA es SHA-1 mód *q*, no simplemente SHA-1, donde *q* es el número primo de 160 bits. Esta debilidad permite la falsificación de un mensaje si el adversario puede seleccionar los parámetros del sistema. Esta debilidad no existe en DSA si los parámetros del sistema son seleccionados como se especifica en el estándar X9.30. El análogo de *q* en ECDSA es *n*, el orden del punto base *P*. Si $n > 2160$, entonces no existen colisiones y tal ataque no es posible.
7. DSA especifica que la integridad del dato firmado es dependiente de la prevención de la revelación no autorizada, modificación, sustitución, inserción y borrado de la clave privada o el valor *k* (particular) de una firma. Sin embargo, mientras quizás sea una implicación, el uso no autorizado no es explícitamente prohibido. En ECDSA, el requerimiento para uso autorizado de una clave privada ha sido hecho explícito.

4.3.10 ESTRUCTURA DE LOS SCRIPTS

Con el objetivo de facilitar los procesos de mantenimiento del software, presentamos en esta sección la estructura de directorios de cada criptosistema, sus contenidos y en general lo relacionado con los scripts PHP que contienen el código fuente de la aplicación.

El código fuente de los criptosistema lo hemos agrupado en un único directorio `ecc_uis`, este directorio raíz agrupa los directorios `ecc` y `repositorioecc` que contienen todo el código fuente del los criptosistemas HERMES y REPOSITORIO-ECC respectivamente.

Se debe aclarar que esta agrupación no es estrictamente necesaria, cada criptosistema es independiente el uno del otro, no comparten ningún script y de hecho, en la práctica, es más probable que el Repositorio y el criptosistema HERMES estén instalados en computadores distintos, aunque se ha diseñado para que ambos puedan coexistir en una misma computadora: sus anillos de claves, archivos *logs* y directorios temporales son disjuntos.

El contenido de los directorios `ecc` y `repositorioecc` están relacionados con las áreas o menús en cada criptosistema. La Figura 31 muestra la estructura de árbol expandido; daremos una breve explicación de los subdirectorios más relevantes. La Figura 32 muestra el contenido de algunos de los subdirectorios de ambos criptosistemas.

Figura 31. Estructura de Directorios de los Criptosistemas.

✓ Contenido del directorio *ecc*

Los subdirectorios: `ayuda`, `cliente`, `consulta`, `ecc`, `giro`, y `sistema` están relacionados con las áreas de los mismos nombres que se muestran en el menú principal del criptosistema HERMES. El directorio `mantenimiento` está relacionado con el área *Base de Datos* por razones "históricas" (y lógicas).

El subdirectorio `biblioteca` contiene el código fuente en C de las librerías[43] de cifrado, en los archivos de cabeceras *.h están definidos los parámetros del dominio de curva elíptica; también en este subdirectorio están algunas librerías de compresión de archivos que utilizamos en la generación del respaldo de la base de datos.

El directorio `demonio` contiene los scripts PHP que trabajan como "`demonios`" en el sentido de los procesos Unix que no presentan una interfaz gráfica de usuario y trabajan en segundo plano; éstos script están en "escucha" permanente, trabajan basados en las reglas del protocolo HERMES y son los encargados de recibir y darle un tratado adecuado a los giros que las otras entidades envían.

[43] El término librería es un error en la traducción del término inglés *library,* el término correcto es *biblioteca;* pese a esto, aceptaremos el término librería debido a su amplio uso.

El subdirectorio `publico` contiene librerías PHP y JavaScript que son utilizadas por casi todos los scripts del sistema.

El subdirectorio `scp` (*secure copy*) es un directorio con permisos especiales donde se almacenan temporalmente archivos utilizados en los procesos críticos del intercambio de datos.

El subdirectorio `uis` es la puerta de entrada al sistema y es el único directorio que el usuario del sistema ve en la URL.

- ✓ **Contenido del directorio `repositorio-ecc`**

El criptosistema REPOSITORIO-ECC agrupa dos sitios Web, uno administrativo de acceso restringido donde se desarrolla toda la gestión de claves públicas, el otro de acceso libre a las entidades financieras donde se publican las listas de claves revocadas y nuevas claves certificadas (firmadas).

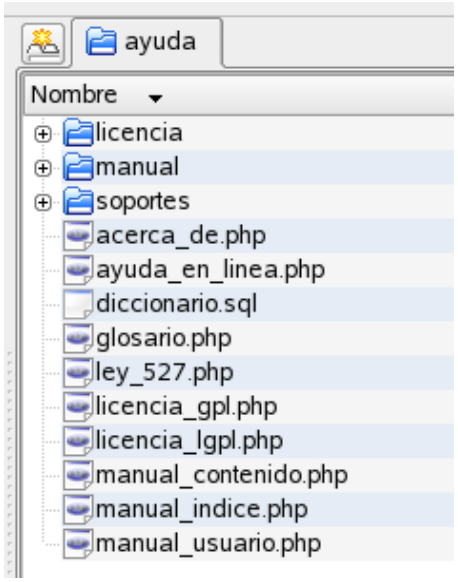
El subdirectorio `adm` del directorio `repositorio-ecc` contiene los script encargados de la gestión de claves por el administrador del sistema, el subdirectorio `src` contiene las bibliotecas donde se definen los parámetros del dominio de curva elíptica y otras librerías de uso general.

Los subdirectorios `importaciones` y `sincronizaciones` contienen respectivamente los *demonios* encargados del tratamiento de las nuevas claves importadas y las actualizaciones automáticas del repositorio con las entidades.

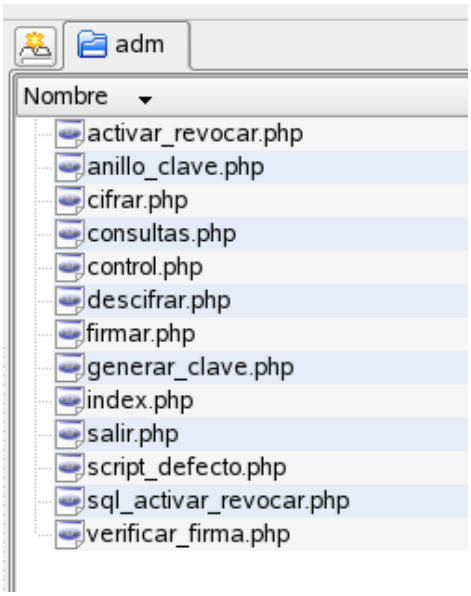
El subdirectorio `uis` contiene el sitio Web donde se publican las listas de claves públicas revocadas y certificadas.

Figura 32. Algunos Scripts de los Criptosistemas.

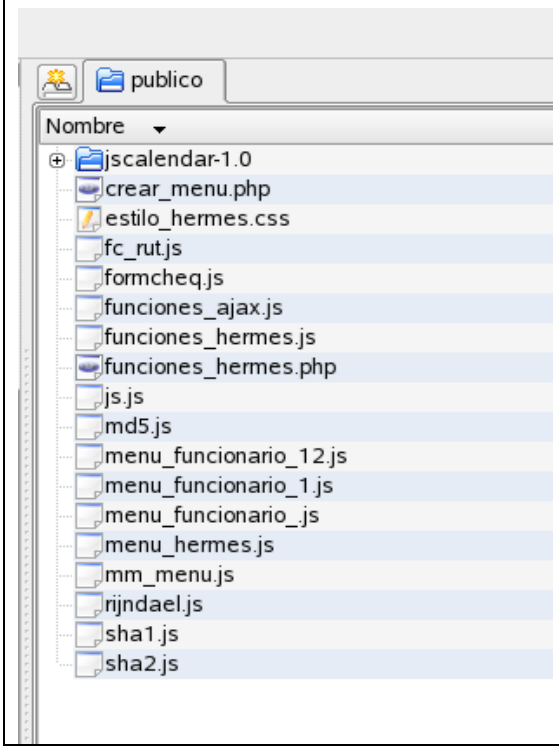
ecc_uis/ecc/ayuda



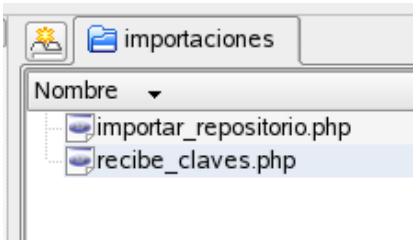
ecc_uis/repositorioecc/adm



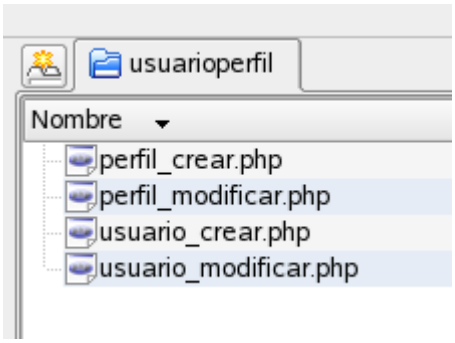
ecc_uis/ecc/publico



ecc_uis/repositorioecc/importaciones



ecc_uis/ecc/usuarioperfil



4.3.11 INSTALADORES

Para ambos sistemas se ha programado un *Shell Script* instalador, esto facilita enormemente el proceso de puesta en marcha del sistema. Los dos instaladores siguen procesos similares, pero varían en el usuario y nombre de la base de datos, en el ajuste de los permisos, y otros detalles pequeños.

Cada instalador se encarga de crear (y borra previamente si existe) el usuario y la base de datos **hermes** (o **repositorioecc**), copia el árbol de directorios del sistema **ecc** (o **repositorioecc**) en el directorio (document_root) de apache, compila las bibliotecas, establece los permisos adecuados, etc. en fin, instala todo el sistema si las variables de instalación son especificadas correctamente.

```
#!/bin/sh
#TRANSFERENCIA REMOTA DE DINERO BASADA EN CRIPTOGRAFÍA DE CURVA ELIPTICA
#LUIS EDUARDO ZAMBRANO FERNANDEZ.
#UNIVERSIDAD INDUSTRIAL DE SANTANDER
#INGENIERIA DE SISTEMAS

###ESPECIFIQUE A CONTINUACION LAS 5 VARIABLES DE INSTALACION

#Valor del DocumentRoot de apache.
#puede verlo en su sistema en los archivos:
#/etc/httpd/conf/httpd2.conf (RedHat)
# o /etc/apache2/default-server.conf (Suse).
# o /etc/apache2/apache2.conf (Versiones recientes).



DIRECTORIO_APACHE=/var/www/html    # <-- RedHat
#DIRECTORIO_APACHE=/srv/www/htdocs # <-- SuSe
#DIRECTORIO_APACHE=/usr/local/apache/htdocs # <-- UIS
#DIRECTORIO_APACHE= # <-- Otro, cual?


#El nombre de usuario unix para apache. (RedHat=apache, SuSe=wwwrun)
UAPACHE=apache    # <-- RedHat
#UAPACHE=wwwrun   # <-- SuSe
#UAPACHE= # <-- Otro, cual?


#El nombre de grupo unix para apache. (RedHat=apache, SuSe=www)
GAPACHE=apache   # <-- RedHat
#GAPACHE=www    # <-- SuSe
#GAPACHE= # <-- Otro, cual?

#El navegador que desea usar. Se ha probado en
# Konqueror (vers>=3.4), Firefox, Mozilla e Iceweasel
# KDE=konqueror, Gnome y otros={firefox | mozilla}
#NAVEGADOR=mozilla
```

```
NAVEGADOR=firefox
#NAVEGADOR=konqueror
#NAVEGADOR=iceweasel
#NAVEGADOR= # <-- Otro, cual?

#El directorio del ejecutable psql
#DIRECTORIO_PSQL=/usr/local/pgsql/bin
DIRECTORIO_PSQL=/usr/bin
#DIRECTORIO_PSQL= # <-- Otro, cual?



###INSTALADOR  (no modificar)

if [ $UID != 0 ]; then
      echo "Debes ser el ROOT para ejecutar este comando"
      exit
fi

if [ -z $DIRECTORIO_APACHE ]; then
      echo "Debes editar \"instalar-ecc.sh\" y descomentar la variable
DIRECTORIO_APACHE para tu sistema"
      exit
fi

if [ ! -d $DIRECTORIO_APACHE ]; then
      echo "El directorio espcificado por la variable DIRECTORIO_APACHE
($DIRECTORIO_APACHE) no existe."
      exit
fi

if [ -z $GAPACHE -o -z $UAPACHE ]; then
      echo "Debes editar \"instalar-ecc.sh\" y descomentar la variable UAPACHE
y GAPACHE para tu sistema"
      exit
fi

fPASSWD=/etc/passwd
if [ -f $fPASSWD ]; then
      TMPGREP=$(grep $UAPACHE $fPASSWD)
      if [ -z "$TMPGREP" ]; then
            echo "El USUARIO especificado en UAPACHE no existe en el sistema"
            exit
      fi
fi

fGROUP=/etc/group
if [ -f $fGROUP ]; then
      TMPGREP=$(grep $GAPACHE $fGROUP)
      if [ -z "$TMPGREP" ]; then
            echo "El GRUPO especificado en GAPACHE no existe en el sistema"
            exit
      fi
```

```
fi

$DIRECTORIO_PSQL/psql -U postgres -d template1 -c "drop database hermes;" &>
/dev/null
$DIRECTORIO_PSQL/psql -U postgres -d template1 -c "drop user hermes;" &>
/dev/null
$DIRECTORIO_PSQL/psql -U postgres -d template1 -c "create user hermes encrypted
password 'luisz' createdb createuser;"
if [ $? -ne 0 ]; then
      echo -e "No se pudo crear el usuario hermes para el sistema
ECC\nVerifique que no se esté usando la template1 en este momento\no bien que
el usuario postgres exista como un usuario administrador"
      exit
fi
$DIRECTORIO_PSQL/createdb -U hermes --encoding LATIN1 hermes
$DIRECTORIO_PSQL/psql -U hermes -d hermes -f hermes.sql
rm -rf $DIRECTORIO_APACHE/ecc &> /dev/null
cp -r ./ecc $DIRECTORIO_APACHE/
if [ "$DIRECTORIO_APACHE" = "/var/www/html" ]; then
      sed "s/\/srv\/www\/htdocs/\/var\/www\/html/g" ./ecc/biblioteca/Makefile >
./ecc/biblioteca/Makefile.lnx
      if [ -s ./ecc/biblioteca/Makefile.lnx ]; then
            mv ./ecc/biblioteca/Makefile.lnx
$DIRECTORIO_APACHE/ecc/biblioteca/Makefile
      fi
fi
if [ "$DIRECTORIO_APACHE" = "/usr/local/apache/htdocs" ]; then
      sed "s/\/srv\/www\/htdocs/\/usr\/local\/apache\/htdocs/g"
./ecc/biblioteca/Makefile > ./ecc/biblioteca/Makefile.lnx
      if [ -s ./ecc/biblioteca/Makefile.lnx ]; then
            mv ./ecc/biblioteca/Makefile.lnx
$DIRECTORIO_APACHE/ecc/biblioteca/Makefile
      fi
fi
ARCH=$(arch)
if [ -z $ARCH ]; then
      ARCH=$CPU
fi

if [ "$ARCH" != "" -a "$ARCH" != "i686" ]; then
      sed "s/ARCH=i686/ARCH=$ARCH/g" $DIRECTORIO_APACHE/ecc/biblioteca/Makefile
> ./ecc/biblioteca/Makefile.lnx
      if [ -s ./ecc/biblioteca/Makefile.lnx ]; then
            mv ./ecc/biblioteca/Makefile.lnx
$DIRECTORIO_APACHE/ecc/biblioteca/Makefile
      fi
fi
rm -f $DIRECTORIO_APACHE/ecc/ecc/sks &> /dev/null
rm -rf $DIRECTORIO_APACHE/ecc/tmp &> /dev/null
rm -rf $DIRECTORIO_APACHE/ecc/log &> /dev/null
rm -rf $DIRECTORIO_APACHE/ecc/scp &> /dev/null
$DIRECTORIO_PSQL/psql -U hermes -d hermes -c "delete from claveentidad;delete
from anilloclave;"
```

```
chmod 755 $DIRECTORIO_APACHE/ecc -R
cd $DIRECTORIO_APACHE/ecc/biblioteca/
      make clean; make; make install &> /dev/null
chown $UAPACHE:$GAPACHE $DIRECTORIO_APACHE/ecc –R
chmod 700 $DIRECTORIO_APACHE/ecc/scp -R
if [ "$NAVEGADOR" != "" ]; then
      $NAVEGADOR http://127.0.0.1/ecc/ &
fi
```

- ✓ **Proceso de instalación.**

El instalador asume lo siguiente:

a) Usted tiene instalado y corriendo en su sistema el Apache, (se espera, aunque es opcional, que el modulo SSL/TLS de apache esté instalado)

b) Usted tiene el PostgreSQL instalado y corriendo.

c) Está disponible el terminal interactivo de Postgres: **psql**

d) Existe el nombre de usuario **postgres** como administrador PostgreSQL.

e) No se está usando la base de datos template1 en el momento de la instalación del sistema.

f) El navegador (agente de usuario) debe permitir ventanas emergentes.

g) El directorio `ecc_uis` está ubicado en un servidor ftp o http del cual usted lo descarga en formato `zip`.

El proceso de instalación es el siguiente:

1. Descargar el `zip` adjunto **ecc-uis.zip**

2. En una terminal, descomprime el `zip` (por ejemplo, con el siguiente comando)
   `unzip ecc-uis.zip`

3. Pasar al directorio donde descomprimió el `zip`.
   `cd ecc-uis/`

4. Edite el archivo **instalar-ecc.sh** y ajuste las 5 variables de instalación.

DIRECTORIO_APACHE, UAPACHE, GAPACHE, DIRECTORIO_PSQL y NAVEGADOR.

Estas variables, corresponden a los valores en su sistema del DocumentRoot de apache, el nombre de usuario de apache, el nombre de grupo de apache, la ubicación del terminal psql y el navegador que desea usar respectivamente. Si su distribución es RedHat quizás los valores por defecto son los adecuados.

5. Puesto que el instalador copia el árbol al directorio de Apache, debe *autenticarse* como root para que este tenga los permisos adecuados.

```
su root
```

6. De permisos de ejecución al instalador.

```
chmod a+x instalar-ecc.sh
```

7. Como root, y en el directorio actual donde descomprimió, ejecute el instalador.

```
sh ./instalar-ecc.sh
```

8. El instalador, al finalizar debe lanzar el navegador con la dirección http://127.0.0.1/ecc que muestra la interfaz de ingreso al sistema, use los datos:

Nombre de Usuario : **admin**

Contraseña : **123456**

para ingreso por primera vez, por razones obvias de seguridad, usted debe cambiar estos datos cuando ingrese al sistema.

9. El primer paso al ingresar al sistema es crear el par de claves de cifrado/firmado por el menú *ECC > Generar Claves*.

10. Asegúrese que su navegador permita ventanas emergentes.

11. Lea el glosario de términos y el manual de usuario que se suministra con la ayuda y ajuste sus variables de entorno, las direcciones IP, puertos de otras entidades, etc. Use el sistema.

Si ocurre algún problema durante la instalación, por ejemplo si el navegador no abre durante los próximos segundos de terminar la instalación, puede intentarlo directamente

en su navegador con la dirección `http://127.0.0.1/ecc` Si por algún motivo el sistema no muestra la interfaz de registro, una lista de posibles problemas pueden ser.

- Las variables de instalación en el archivo **instalar-ecc.sh** no son las adecuadas para su sistema y debe cambiarlas.
- Apache, aunque puede estar instalado, no está corriendo. Debe iniciar el apache `/etc/init.d/httpd2 start` o `/etc/init.d/apache2 start`
- El postgres, aunque puede estar instalado, no está corriendo. Debe iniciarlo `/etc/init.d/postgresql start` (si están como servicios)
- Se está usando la base de datos template1 por algún usuario. Debe salir y volver a ejecutar el instalador (paso 7)
- Si definitivamente no es posible la instalación, envíeme un correo a lezambranof@gmail.com para intentar solucionarlo.

El proceso de instalación del repositorio es análogo, pero debe hacerse con el Shell Script `intalar-repositorioecc.sh` y para el ingreso por primera vez use los datos:

Nombre de Usuario : **repositorio**

Contraseña : **9876543210**

4.3.12 VENTAJAS DE LOS CRIPTOSISTEMAS IMPLEMENTADOS.

✓ **SEGURIDAD:**

En la Tabla 6 vemos una comparación del tamaño necesario de los parámetros de varios esquemas para alcanzar un nivel comparable de seguridad. En la primera columna se presenta el tamaño de un parámetro en bits. La segunda columna lista los tamaños de claves en un esquema simétrico "ideal", luego los tamaños de un esquema basado en ECC, y después para los esquemas RSA y DSA (utilizan el PFE y PLD respectivamente.) Cada fila representa un nivel de seguridad comparable entre los distintos métodos. En los ECC el parámetro de seguridad es el tamaño de n (el orden del punto base G).

Tabla 6. Tamaños de clave comparables (en bits).

Nivel de Seguridad	Esquema Simétrico (tamaño de clave)	Esquema Basado en ECC (tamaño de *n*)	DSA/RSA (tamaño del módulo)
56	56	112	512
80	80	160	1024
112	112	224	2048
128	128	256	3072
192	192	384	7680
256	256	512	15360

- ✓ **EFICIENCIA:** Cuando se habla sobre eficiencia de un criptosistema de clave pública, hay que tener en cuenta tres factores:

  - Sobrecarga en cálculos: Cuanta computación se requiere para ejecutar las transformaciones de clave privada y clave pública.
  - Tamaño de clave: Cuantos bits se requieren para almacenar el par de claves y algunos otros parámetros del sistema.
  - Ancho de banda: Cuantos bits deben ser comunicados para transferir un mensaje encriptado o una firma.

Efectuaremos las comparaciones de eficiencia de un criptosistema ECC de 160 bits con un RSA y DSA de 1024 bits, ya que ambos proveen un nivel comparable de seguridad.

- ✓ **SOBRECARGA DE CALCULOS:** En RSA, un exponente público corto puede ser empleado (aunque esto presenta algunos riesgos de seguridad) para acelerar la encriptación y la verificación de firma (por ej., suele emplearse el exponente $e = 3$ o $e = 216 + 1$).[44] En DSA y ECC, una gran proporción de la generación de una firma y transformaciones de encriptación pueden ser precomputada. También, varias bases especiales para los campos finitos F_{2^m} pueden ser empleadas para ejecutar más rápidamente la aritmética modular involucrada en la operación de ECC, como por ejemplo usar una representación normal optima o cambiar los puntos de la curva a coordenadas proyectivas. Buenas implementaciones de los sistemas muestran que con todas estas eficiencias incorporadas, ECC es un orden de magnitud más rápido que RSA o DSA.

- ✓ **TAMAÑO DE CLAVE:** En la Tabla 7 se muestran los tamaños de clave pública y privada, y de parámetros del sistema. Es claro que los de ECC son más cortos que los de RSA o DSA.

[44] Note que la representación binaria de *e* tiene solo dos 1's, lo cual hace que el algoritmo square-and-multiply sea muy eficiente.

Tabla 7. Tamaño de los parámetros del sistema y par de claves (en bits).

	Parámetro del Sistema	Clave Pública	Clave Privada
RSA	n/a	1088	2048
DSA	2208	1024	160
ECC	481	161	160

- ✓ **ANCHO DE BANDA:** Aquí consideramos solo el caso cuando mensajes cortos están siendo transformados, ya que los criptosistemas de clave pública son a menudo empleados para transmitir mensajes cortos, como claves de sesión para algoritmos e encriptación de clave simétrica. Para hacer una comparación concreta, supongamos que cada sistema esta siendo usado para firmar un mensaje de 2000 bits, o para cifrar un mensaje de 100 bits. Tablas 8 y 9 comparan las longitudes de las firmas y mensajes cifrados respectivamente (observe que en Tabla 9, se compara el cifrado con el método de ElGamal, en vez de DSA. Esta comparación tiene sentido puesto que ambos se basan en el mismo algoritmo). Se ve que ECC ofrece un ahorro de ancho de banda considerable sobre los otros tipos de sistemas de clave pública.

Tabla 8. Tamaños de firma (en bits).

	Tamaño de Firma
RSA	1024
ElGamal	320
ECC	320

Tabla 9. Tamaños de mensajes cifrados (en bits).

	Tamaño mensaje cifrado
RSA	1024
ElGamal	2084
ECC	321

Por lo tanto, todos estos ahorros redundan en velocidades más altas, menor consumo de energía, y reducciones en el tamaño del código.

4.3.13 PROTOCOLO DE COMUNICACIÓN HERMES

La interoperabilidad entre algoritmos de cifrado y firmado en un esquema asimétrico queda garantizada cuando las partes usan los mismos parámetros del dominio, en nuestro caso, las entidades financieras y el repositorio de claves comparten los parámetros del dominio de curva elíptica definidos en 4.3.4, sin embargo, esta interoperabilidad está limitada a los procesos criptográficos y no contempla el medio y la forma en que los datos son transmitidos, el formato de los documentos, qué tipos de problemas se pueden presentar durante el intercambio, qué lógica se debe seguir cuando ocurran estos problemas, etc.

La descripción formal de todas las reglas de tratado criptográfico, formato y transmisión de la información financiera entre las entidades involucradas en nuestro modelo PKI conforman el PROTOCOLO HERMES.

✓ **Diseño.**

Dado que nuestro objetivo es la transferencia de información financiera en entornos Web, el protocolo HERMES se ha diseñado como un protocolo de nivel superior construido sobre el protocolo HTTP, éste a su vez basado en y/o parte de la familia de protocolos TCP/IP de más bajo nivel (transporte y red). Este diseño nos permite hacer uso de toda la sintaxis y semántica XHTML, de los métodos de transmisión HTTP, en especial el método POST.

✓ **Caso de Uso.**

1. Instalación, ajuste de variables de entorno, establecimiento de dirección IP y puerto del criptosistema REPOSITORIO-ECC.
2. Creación del par de claves del esquema asimétrico en el repositorio.
3. Instalación del criptosistema HERMES para dos entidades financieras ubicadas en dos ciudades distantes A y B.
4. Ajuste de variables de entorno, direcciones IP y puertos en las entidades financieras.
5. Registro de los datos del repositorio en las entidades financieras: dirección IP, puerto, recursos de sincronizaciones y test de conexiones HTTP.
6. Creación de pares de claves públicas en las entidades financieras y envío de solitud de certificación al repositorio de dichas claves.

   Comentario. El sistema permite el envío de la clave pública al repositorio como un proceso automatizado en la creación de las claves, pero puede ser enviada posteriormente o presentada personalmente.

7. El repositorio de claves certifica ambas claves previa comprobación y certeza de que las claves pertenecen a las entidades financieras.

8. La entidad A descarga del repositorio la clave pública certificada de la entidad B y viceversa.

   Comentario. El proceso puede hacerse descargando la clave pública de las listas de claves certificadas y revocadas, o bien, puede hacerse en un proceso automatizado por los demonios en HERMES que constantemente verifican e informan la aparición de nuevas claves certificadas (o revocadas).

9. El cliente se acerca a un punto de giro en la ciudad A para enviar dinero a otra persona en la ciudad B.

   Comentario. El sistema admite personas naturales o personas jurídicas como remitentes o destinatarias de giros.

10. El funcionario de la entidad financiera A registra la información del cliente en el sistema HERMES, la información es almacenada en la base de datos y se asigna un código único de giro, el estado inicial del giro es 1: *Registrado sin enviar*.

    Comentario. Cuando un cliente se acerca por primera vez, se registra en el sistema todos los datos del cliente, nombres, apellidos, etc. cuando el cliente envíe otro giro posteriormente, solo se necesitará su documento de identidad, el sistema rastreará y presentara en la interfaz toda la información del cliente.

11. El funcionario administrador de la entidad A (remitente) ingresa al submódulo de "Reporte de giros", digita la frase privada para firmado digital y presiona el botón "Reportar Giro" para enviar el giro a la entidad en la ciudad B (destinataria).

12. El sistema valida que la frase privada se corresponda con la clave pública de la entidad financiera A. Si la frase privada es correcta, el sistema forma un documento con un formato definido, lo cifra con la clave pública de la entidad B y lo firma con la clave privada de la entidad A.

13. Un procedimiento del sistema abre un socket remoto en la entidad B especificando 3 argumentos: la dirección IP de la entidad A, el puerto de escucha del servidor Web (generalmente el 80 pero puede ser cualquier otro) y un recurso HTTP que recibirá el documento cifrado (un script demonio en PHP).

14. Un proceso en la entidad A envía a la entidad B el documento cifrado y firmado en 12 a través del socket abierto en 13 y pasa el giro al estado 2: *Enviado sin confirmar*.

    Comentario. Una lista extensa de los problemas que pueden presentarse durante la transmisión del documento son mostrados en la Tabla 10.

15. Un demonio en la entidad B verifica que el documento proviene de una IP perteneciente a una entidad conocida, si es así el documento es almacenado en un

espacio de memora temporal, en otro caso el documento es rechazado y el proceso termina.

16. Un proceso en B verifica la firma del documento recibido con la clave pública de A. Si la firma no se verifica, el documento es rechazado, se informa a la entidad A y el proceso termina.

17. El usuario administrador en B descifra el documento recibido utilizando la clave privada.

18. Un procedimiento en B verifica que el archivo descifrado en 17 tenga el formato adecuado, si no es así, el documento es rechazado, se informa a la entidad A y el proceso termina.

19. Un procedimiento en B guarda el contenido del archivo descifrado, asigna un código único de giro (distinto al asignado en A), establece el estado del giro recibido a 5: *Recibido sin pagar* e informa a la entidad A con un mensaje de recibido.

20. Un proceso en la entidad A captura el mensaje de recibido y pasa el giro al estado 3: *Enviado y confirmado*.

21. La persona destinataria en la ciudad B, se acerca al punto de giro B y se identifica ante un funcionario.

22. El funcionario en B entrega el dinero del valor del giro a la persona destinataria y registra el pago del giro en el sistema; el registro del pago pasa el giro en B al estado 6: *Recibido y pagado*.

23. El usuario administrador en B realiza un informe de pagos y notifica a la entidad A que se ha realizado el pago del giro. El giro en B pasa al estado 7: *Recibido, pagado e informado*. En el mismo proceso de informe de pagos un demonio en A recibe la notificación de B y pasa el giro (en B) al estado 4: *Enviado, confirmado y pagado*.

    Comentario: Note que los informes de pagos son procedimientos atómicos y el cambio del giro a los estados 4 y 7 en A y B respectivamente es "simultáneo".

- ✓ **Estados de Transferencia Sobre HTTP**

En los diferentes eventos de la comunicación entre los puntos de giro y el repositorio se pueden presentar diferentes problemas, unos inherentes al protocolo HTTP, otros asociados a interrupciones de la conexión, e inclusive otros asociados a errores en los scripts (*bugs*). El protocolo HERMES define tanto el tratado de la información transmitida

como el conjunto de problemas que se pueden presentar durante dicha transmisión. La Tabla 10 muestra algunos de estos estados para los procesos principales.

Los códigos de errores están categorizados en intervalos, los códigos de estados en el intervalo [100, 299] corresponden a procesos de envíos de giros, los códigos en el intervalo [300, 499] corresponden a reporte de pagos, la exportación de claves públicas desde los puntos de giros al repositorio están agrupados en el intervalo [600, 699], las sincronizaciones de datos desde el repositorio a las entidades y las descargas de claves públicas certificadas están agrupadas en los intervalos [800, 899] y [900,999] respectivamente. Los estados 1000, 1001, 1002, 1003, etc. son estados que indican transferencias exitosas en los diferentes procesos.

Tabla 10. Estados de transferencia del protocolo HERMES

Proceso /Actor	**Código Estado**	**Descripción**
Envío de Giro. Entidad A.	100	Intento de envío de giros por usuario no autorizado.
	101	El código de giro especificado no existe en la base de datos.
	102	No se pudo construir el documento a transferir: No se cuenta con permisos de escritura en el directorio scp.
	103	El estado del giro no permite enviarlo: el giro debe estar en “Registrado sin enviar” o “Enviado sin confirmar”.
	104	No existen giros pendientes por reportar.
	105	Fallo de verificación: la información del archivo creado difiere a la información en la base de datos, posible archivo corrupto.
	106	La clave pública de la entidad destino no existe en el anillo de claves.
	107	Falló el cifrado del documento.
	106	Falló el firmado del documento.
	107	La apertura del socket FALLÓ (IP=$direccion_ip, puerto=$puerto, recurso_remoto=$recurso_remoto)
	...	
	116	Imposible cambiar el estado del giro a “Enviado sin confirmar”
	117	Imposible registrar la auditoría.

	1000	Envío exitoso.
Recepción de Giros. Entidad B.	200	Se recibió una petición HTTP sin un archivo de giro. Posible acceso no autorizado.
	201.	Archivo de giro sin firma adjunta.
	202.	El nombre de archivo no se corresponde con el formato adecuado.
	203.	El archivo recibido no pudo ser cargado a un espacio de memoria temporal.
	204.	La firma no pudo ser cargada a un espacio de memoria temporal.
	…	
	211	Falló la verificación de la firma.
	…	
	1001	Giro recibido exitosamente.
Reporte de Pagos. Entidad B.	300	Imposible informar pagos, se recibió una variable sin su par requerido (código de giro + envidad remitente)
	301	El estado del giro impide informar su pago.
	302	No existen giros pendientes por informar.
	303	Imposible crear el archivo para informe de pagos: Permiso denegado en el directorio scp

	304	La clave pública de la entidad remitente fue eliminada del anillo de claves.
	305	Falló el firmado del archivo para informe de pagos.
	306	Falló el cifrado del archivo para informe de pagos.
	…	
	313	La apertura del socket falló (IP=$direccion_ip, puerto=$puerto, recurso_remoto=$recurso_remoto).
	314	Imposible cambiar el estado del giro a “7-Recibido, pagado e informado”
	1002	Informe de pagos realizado exitosamente.
Recibir informes de pagos. Entidad A.	400	Se recibió una petición HTTP sin envío de archivos de informe de pagos.
	401	Se recibió un archivo de informe de pagos sin firma adjunta.
	402	El archivo recibido no pudo ser cargado a un espacio de memoria temporal.
	403	La firma no pudo ser cargada a un espacio de memoria temporal.
	…	
	410	Verificar la firma del archivo recibido.
	412	Imposible descifrar el archivo recibido

	413	El formato del contenido del archivo recibido no es válido.
	415	El contenido del archivo recibido no coincide con algún giro enviado, confirmado y pendiente por informe de pago.
	…	
	1003	Informe de pagos recibido exitosamente.
	500-599	No usado
Envío de claves públicas al repositorio. Todas las Entidades Financieras.	600	No se ha definido un repositorio por defecto en la variable de entorno 16.
	601	Imposible crear el archivo para exportar la clave. Permiso denegado en scp.
	602	Imposible abrir un socket en el repositorio (IP=$ip_repositorio, puerto=$puerto_repositorio, recurso=$recibe_claves).
	603	Falló la verificación: la clave en el archivo a enviar no coincide con la clave en el anillo.
	…	
	1004	Clave exportada al repositorio exitosamente.
Recibir nuevas claves públicas para certificar. Repositorio.	700	Se recibió una petición HTTP sin envío de clave pública
	701	No se pudo mover la clave pública recibida a un espacio de memoria temporal.
	702	La clave pública no corresponde con un formado adecuado: la clave está fuera de rango.

	…	
	1005	Clave pública recibida exitosamente: pendiente la verificación de la identidad del remitente.
Sincronización de datos con el repositorio. Todas las entidades.	800	No se ha definido un repositorio por defecto en la variable de entorno 16.
	801	Imposible abrir un socket en el repositorio (IP=$ip_repositorio, puerto=$puerto_repositorio, recurso=$sincronizaciones).
	...	
	807	Imposible consultar los datos del repositorio: la petición fue denegada.
	1006	Datos del repositorio confirmado exitosamente.
Sincronización de datos con las entidades financieras. Repositorio.	900	La petición de datos del repositorio no tiene el formato adecuado: la petición será denegada.
	901	No se aceptan peticiones de funcionarios inactivos.
	902	No se aceptan peticiones de entidades financieras inactivas.
	905	La clave del repositorio no ha sido generada aún.

	906	La clave del repositorio no está disponible: no se pudo leer el anillo de claves.
	..	
	910	Imposible crear el documento con la clave pública del repositorio.
	913	No se pudo Autofirmar la clave pública del repositorio.
	...	
	1007	Petición respondida, proceso completado.

5. ANÁLISIS DE LA BIBLIOGRAFÍA

5.1 CURVAS ELÍPTICAS EN CRIPTOGRAFÍA.

ATKIN A.O.L., MORAIN F. Elliptic Curves and Primality Proving. Mathematics of Computation, 1993.

En este libro se encuentra la base matemática de las curvas elípticas, describe analíticamente las curvas singulares, definición de la suma de puntos, el punto al infinito, entre otros aspectos. Se necesita una buena base matemática para un buen seguimiento.

WASHINGTON Lawrence C. Elliptic Curves. CRC Press. 2003. 428 p.

Escrito por un teórico bien conocido sobre la teoría de números, este libro presenta un fondo criptográfico extenso, entreteje la teoría de curvas elípticas con sus usos, particularmente en la criptografía. Un tratamiento cuidadoso se da para el problema del logaritmo discreto para las curvas elípticas.

MENEZES Alfred J. Elliptic Curve Public Key Cryptosystems. Springer. 1993. 144 p.

Especial comparativa nos muestra el autor sobre la seguridad equivalente que presentan los ECC con respecto a los esquemas públicos existentes, pero con longitudes de claves más cortas. El libro examina las variadas situaciones que se presentan en la puesta en práctica segura y eficiente de los ECC.

BLAKE A., SEROUSSI G., SMART N. Elliptic Curves in Cryptography. London Mathematical Society, Cambridge University Press, 1999. 220 p.

Este trabajo resume el conocimiento recolectado en Hewlett-Packard sobre un número de años y explica las matemáticas detrás de la puesta en práctica de los sistemas de curva elípticos.

ESPINOSA F.J., HERNÁNDEZ L. Una Revisión de los Criptosistemas de Clave Pública Sobre Curvas Elípticas e Hiperelípticas. VIII RECSI, 2004.

En este libro se hace el comparativo entre diferentes tipos de curvas elípticas, se estudia el grupo de puntos de una curva elíptica definida sobre un cuerpo finito y el Jacobiano de una curva hiperelíptica (sin puntos singulares) definida sobre un cuerpo finito.

HANKERSON D., MENEZES A., VANSTONE S. Guide to Elliptic Curve Cryptography, Springer, 2004. 311 p.

Anclado por un tratamiento comprensivo de los aspectos prácticos de la criptografía de curva elíptica, esta guía explica las matemáticas básicas, describe métodos avanzados de la puesta en práctica, y presenta los protocolos estandarizados para el cifrado de llave pública, las firmas digitales, etc. Además, cabe resaltar la inclusión de referencias útiles, una lista de algoritmos, y apéndices en parámetros de la muestra, estándares de ECC, y herramientas software de fácil comprensión, esta referencia altamente enfocada es un recurso útil e imprescindible para los investigadores en informática, profesionales del diseño de red y profesionales en la seguridad de datos de la red.

5.2 CRIPTOGRAFÍA Y SEGURIDAD

PASTOR José, SARASA Miguel Angel. CRIPTOGRAFÍA DIGITAL. FUNDAMENTOS Y APLICACIONES. Prensas Universitarias de Zaragoza. 1998. 597 p.

Extenso y completo texto sobre técnicas criptográficas modernas, con una buena cantidad de ejemplos y profusión de tablas con datos de interés. Destacan los capítulos de cifra con clave secreta, en donde se estudian más de una docena de criptosistemas, los de clave pública y su aplicación en firmas digitales y un capítulo dedicado a protocolos criptográficos. En particular, están muy bien tratados los apéndices con temas matemáticos así como los algoritmos de factorización y del logaritmo discreto, algo no muy común y que se agradece. Para el buen seguimiento es necesario contar con una buena base matemática.

FÚSTER Amparo, DE LA GUÍA Dolores, HERNÁNDEZ Luis, MONTOYA Fausto, MUÑOZ Jaime. Técnicas Criptográficas de Protección de Datos. TERCERA EDICIÓN. Editorial Ra-Ma. 2004. 416 p.

Detallado y actualizado resumen de las técnicas de cifra modernas, profundizando en los sistemas de clave secreta con cifra en flujo y en bloque, de clave pública y sus aplicaciones en redes. De especial interés resulta el capítulo dedicado a protocolos criptográficos y sus apéndices en donde explica los métodos matemáticos usados en criptografía y nociones sobre complejidad computacional. La edición incluye además una interesante colección de problemas y sus soluciones en un CD ROM.

CABALLERO Pino. Introducción a La Criptografía. SEGUNDA EDICIÓN. Editorial Ra-Ma, Textos Universitarios, Madrid 2002. 160 p.

Libro de introducción a las técnicas criptográficas que presenta estos temas bajo una orientación matemática clara y precisa. Actualización de la primera edición de 1996 en la que trata los temas de criptografía teórica, criptografía de clave secreta y pública, problemas de autenticación y accesos y algunas aplicaciones criptográficas. Para un buen seguimiento de la lectura, en algunos apartados es recomendable contar con una base de conocimientos en matemáticas a nivel universitario.

STALLINGS William. Fundamentos de Seguridad en Redes. Aplicaciones y Estándares. Prentice-Hall Inc. 2003. 456 p.

Corresponde a la traducción del inglés al español del libro que se comenta más adelante. Manteniendo la misma filosofía que su edición en inglés, en este caso contiene 225 páginas menos. Presenta temas de cifra simétrica y asimétrica, algoritmos, firmas, hash, autenticación y seguridad en redes. Está más centrada en la seguridad en Internet, profundizando en temas como correo seguro, protocolos de redes, IP seguro, seguridad en Web, intrusiones, cortafuegos, etc. Un excelente libro para el estudiante, si bien para el profesor es recomendable el original en inglés.

MORANT R. J.L., RIBAGORDA G. A., SANCHO R. J., Seguridad Y Protección de la Información. Colección de Informática, Centro de Estudios Ramón Areces S.A., Madrid, 1994. 388 p.

Libro que trata, además de los temas genéricos de la criptografía clásica y moderna, aspectos de seguridad en sistemas operativos, en bases de datos y en redes de computadores. Buen texto descriptivo que profundiza en ciertos aspectos matemáticos y hace un buen estudio de la gestión de claves. Tiene además como característica ser el primer libro con formato universitario sobre criptografía y seguridad informática en España, y seguramente de lengua española.

STALLINGS William. Cryptography and Network Security. Principles and Practice. Third Edition. Prentice-Hall Inc. 2003. 681 p.

Con una centena más de páginas que la segunda edición de (1999), el texto está estructurado de una forma óptima que permite una agradable lectura. Además de cifra simétrica y asimétrica, algoritmos, firmas, hash, autenticación y seguridad en redes, esta edición se centra en la seguridad en Internet, profundizando en temas como correo seguro, protocolos de redes, IP seguro, seguridad en Web, intrusiones, cortafuegos, etc. Incluye algunos ejemplos y ejercicios. Junto al de A. Menezes, es probablemente el mejor libro de criptografía y seguridad informática en la actualidad.

MENEZES Alfred, OORSSCHOF Paul, VANSTONE Scott. Handbook of Applied Cryptography. CRC Press Inc. 1997. 780 p.

Interesante y completo libro dedicado al estudio de los algoritmos con una visión matemática de alto nivel. Sus capítulos están orientados a bases matemáticas para la criptografía, sistemas de claves secretas y públicas, cifradores de flujo y de bloque, hash, autenticación, firma digital y gestión de claves. Obra imprescindible para el estudiante universitario que desea profundizar en el análisis de los algoritmos, si bien el seguimiento del mismo puede resultar algo complejo por el nivel matemático comentado. Junto al de W. Stallings, es probablemente el mejor libro de criptografía en la actualidad.

SEBERRY Jennifer, PIEPRZYK Josef. Cryptography An Introduction To Computer Security. Prentice-Hall, New York, 1989. 375 p.
Además de los temas propios de criptografía clásica y moderna, incluye un capítulo de introducción a la aritmética modular bien estructurado. Trata también la seguridad informática en bases de datos, en sistemas operativos y en redes. Como lector se agradece en especial la gran cantidad de ejercicios propuestos y resueltos en cada capítulo, incluyendo el código en PASCAL.

5.3 TÉCNICOS

JACOBSON, Ivar. BOOCH, Grady. RUMBAUGH, James. Proceso Unificado de Desarrollo de Software. Addison Wesley. 2000.

Es un libro imprescindible, proporciona una amplia revisión sobre el proceso unificado de desarrollo de software, poniendo especial énfasis en el modelado practico con UML.

JACOBSON, Ivar. BOOCH, Grady. RUMBAUGH, James. El Lenguaje Unificado de Modelado. Addison Wesley. 1999

Este libro presenta el Lenguaje que utiliza la metodología que planeamos utilizar para el desarrollo del proyecto UML.

GRECH, Pablo. Introducción a la Ingeniería. Prentice Hall. 2001.

Libro orientado a desarrollar las habilidades de los estudiantes de ingeniería en la solución de problemas.

PRESSMAN, Roger S. Ingeniería del Software. Ed McGraw Hill. Quinta edición 2002.

Este libro presenta temas importantes a tener en cuenta cuando se inicia un desarrollo software como lo son los métodos formales de desarrollo, patrones de desarrollo, métricas entre otros.

KORTH, Fundamentos de Bases de Datos. Ed McGraw-Hill. 1998.

Ofrece un marco completo de los fundamentos del diseño, lenguajes de acceso e implementación de bases de datos.

5.4 TRABAJOS DE GRADO

INTRODUCCIÓN A LOS CRIPTOSISTEMAS DE CURVA ELÍPTICA.
Universidad Industrial de Santander. Facultad de Ciencias, Bucaramanga, 2001.

Autor : Alonso Sepúlveda Castellanos
Director : Edilberto Reyes González

SEGURIDAD EN EL COMERCIO ELECTRÓNICO.
Pontificia Universidad Javeriana. Facultad de Derecho, Bogotá D.C, 2004.

Autor : Héctor José García Santiago
Director : Alvaro Andrés Motta Navas

5.5 ENLACES WEB

http://citeseer.ist.psu.edu/hankerson00software.html Darrel Hankerson, Julio López Hernández, Alfred Menezes, *Software Implementation of Elliptic Curve Cryptography over Binary Fields,* 2000.

http://citeseer.ist.psu.edu/lopez00highspeed.html Julio López Hernández, Ricardo Dahab, *High-Speed Software Multiplication in F(2 ^{m}),* 2000.

http://www.cacr.math.uwaterloo.ca/ecc/ Darrel Hankerson, Alfred Menezes, and Scott Vanstone. *Guide to Elliptic Curve Cryptography. Sample chapter: Finite Field Arithmetic,* March 2005.

http://www.secg.org/download/aid-385/sec1_final.pdf Certicom, Standards for Efficient Cryptography, *SEC 1: Elliptic Curve Cryptography, Version 1.0,* September 2000.

_____, Standards for Efficient Cryptography, *SEC 2: Recommended Elliptic Curve Domain Parameters, Version 1.0,* September 2000. http://www.secg.org/download/aid-386/sec2_final.pdf

_____, Certicom ecc tutorials, http://www.certicom.com/ecc/enter/index.htm.

_____, Current public-key cryptographic systems, Whitepaper 2, Certicom Corp., http://www.certicom.com, April 1997.

_____, Remarks on the security of elliptic curve cryptosystem, Whitepaper 3, Certicom Corp., http://www.certicom.com, September 1997.

_____, The elliptic curve cryptosystem for smart cards, Whitepaper 7, Certicom Corp., http://www.certicom.com, May 1998.

http://sks.merseine.nu/ Manuel Pancorbo Castro, *SKS Versión 0,92b.*

http://es.wikipedia.org Wikipedia, Enciclopedia en Línea.

http://www.cr0.net:8040/code/crypto/ Código en C de: AES, RC4, 3DES, MD5, SHA1, SHA2.

http://www.cs.eku.edu/faculty/styer/460/Encrypt/JS-SHA1.html A Javascript SHA-1 calculator showing intermediate values in the calculation.

http://www.gnupg.org/ GnuPG.

http://www.pgp.com/ PGP Corportation.

http://www.pgpi.org/ PGP International.

http://www.openpgp.org/ OpenPGP Alliance.

http://www.ietf.org/html.charters/secsh-charter.html Protocolo SSH.

http://www.openssl.org/ OpenSSL.

http://www.openssh.org/ OpenSSH.

http://www.nist.gov/ National Institute of Standards and Technology (NIST).

http://www.itl.nist.gov/fipspubs Federal Information Processing Standard.

http://www.ansi.org American National Standards Institute.

http://www.iso.org/ International Organization for Standardization.

http://www.ieee.org/ Institute of Electrical and Electronics Engineers.

http://www.ietf.org/ietf_chairs_year.html Internet Engineering Task Force.

http://www.nsa.gov/ National Security Agency.

http://www.gchq.gov.uk/ Government Communications Headquarters.

http://www.cse-cst.gc.ca/ Communications Security Establishment.

http://www.cryptonessie.org/ NESSIE.

http://www.ipa.go.jp/security/enc/CRYPTREC/index-e.html Cryptography Research and Evaluation Committee.

http://www.cryptool.org/ CryptTool.

6. RECONOCIMIENTOS.

Se hace difícil poder relacionar aquí todas las personas que colaboraron directa e indirectamente en la realización de este proyecto, el software utilizado es en su mayoría el esfuerzo conjunto de muchas personas, que ningún libro sería los suficientemente completo para contenerlas. Se ha utilizado principalmente software de libre distribución bajo la las licencias GPL y LGPL, algunos de estos recursos queremos presentar:

- El menú que se presenta en la interfaz principal de HERMES-ECC está basado en el JavaScript *mm_menu.js* Versión 6.0 por Andy Finnell de Macromedia, Inc. Éste a su vez, se basó en el script *menu.js* por Gary Smith de Netscape Communications Corp. Las modificaciones que se hicieron para este trabajo de grado básicamente fueron el soporte para iconos y la generación dinámica (menú cambiante según perfiles de usuario).

- El calendario en las interfaces de consulta es una adaptación del script *calendar.js* Versión 1.51 de Mihai Bazon, se hicieron modificaciones solo en las hojas de estilo.

- Los principales algoritmos de cifrado fueron tomados de la librería *LibTomCrypt* (*Modular Cryptographic Library*) por Tom St Denis. *LibTomCryp* es una librería que provee varios algoritmos de cifrado en un medio altamente modular y flexible.

- Se han adaptado las implementaciones en JavaScript de los algoritmos para *MD5* y *SHA-1* de Paul Johnston & Compañía. y la implementación de *SHA-2* por Angel Marin. El algoritmo de resumen *Tiger* fue una adaptación del algoritmo desarrollado por Ross Anderson y Eli Biham.

- Los algoritmos para el cálculo con números grandes y para las operaciones polinómicas fueron tomados de la librería *LibTomMath* también de Tom St Denis.

- La construcción de la aplicación en Linux que integra los algoritmos de cifrado y la gestión de los anillos de claves están basados en la filosofía y diseño de *SKS* (*Simpla Kriptografia Sistemo*) del profesor Manuel Pancorbo Castro, éste a su vez se basó en el diseño de *pegwit* de George Barwood. Ligeras modificaciones se hicieron para hacer posible la interacción e intercambio de datos entre la aplicación Linux y el PHP (ambos en el mismos servidor).

- Para la generación de los parámetros del dominio y la validación de los mismos se utilizó *Elliptic Curve Builder, Ellipsa-ECB,* una aplicación disponible para entornos Windows sobre plataforma de 32 bits.

7. CONCLUSIONES

Pese a su complejidad para estudiarlas, las curvas elípticas ganan cada vez más interés por parte de las comunidades de matemáticos e informáticos; su utilidad práctica está más que demostrada, desde el estudio para calcular perímetros de elipses (de donde toman su nombre), hasta nuestros tiempos donde han sido útiles en la construcción de aplicaciones criptográficas, en la factorización de números enteros y en la demostración[45] del *Último Teorema de Fermat* (en particular la demostración de la *conjetura de Taniyama-Shimura* que sugiere que cada curva elíptica puede asociarse unívocamente con un objeto matemático denominado *forma modular*).

Sin embargo, aunque en criptografía las curvas elípticas hayan alcanzado un alto nivel de confianza, su seguridad está basada en que no se han encontrado ataques efectivos al PLDCE. Al igual que otros esquemas asimétricos, esta confianza irá en aumento a medida que pase el tiempo y que las curvas elípticas se muestren prometedoras ante nuevos ataques.

En general, las curvas elípticas presentan variadas ventajas sobre los actuales y más utilizados criptosistemas RSA y DSA: iguales niveles de seguridad con longitudes de claves significativamente más cortas, menor costo de cómputo y reducido consumo de ancho de banda por lo que se espera un completo reemplazo de estos a medida que se vayan adoptando los nuevos estándares ECC.

En la implementación de criptosistemas en software, las curvas elípticas definidas sobre F_{2^m} son ventajosas respecto a las definidas sobre Z_p dada la facilidad de representación de los polinomios binarios como palabras de bits, lo que incrementa la eficiencia en la aritmética de puntos. Sin embargo, cualquiera que sea el campo donde deseen definirse, existen curvas criptográficamente débiles que deben ser evitadas.

En Colombia, el problema de inseguridad de la información financiera es de relevancia y naturaleza comparable con el flujo de dinero por medios electrónicos. Esta relación lógicamente opuesta en conveniencia, ha mostrado un significativo aumento en los últimos años y nos insta a ofrecer soluciones desde la universidad. Se ha demostrado con la implementación de este criptosistema que es posible ofrecer una solución fiable, de implementación a bajo costo que garantiza los principales requerimientos de la seguridad informática haciendo uso de uno de los mejores recursos disponibles en la actualidad: la criptografía de curva elíptica.

[45] WILES, Andrew. *Modular elliptic curves and Fermat's last theorem*. Annals of Mathematics. (2) 141 (1995), no. 3, 443—551.

8. RECOMENDACIONES

Los criptosistemas y en general todo software relacionado con la seguridad debe estar sometido a revisión permanente por la comunidad académica. La seguridad del cifrado debe estar siempre en función de la clave privada y no en *secretismos* en el código fuente o en el diseño. Invitamos a la comunidad académica a revisar los criptosistemas desarrollados, a validarlos y en lo posible a contribuir con nuevas implementaciones que ayuden a fortalecer los aspectos de la seguridad informática o mejorar las implementaciones existentes, esto es condición necesaria para garantizar la madurez del software en el futuro.

Tres aspectos particulares se desean recomendar como punto de partida:

1. La implementación de nuevos algoritmos que soporten coordenadas proyectivas o la representación en base normal optima sobre los campos finitos F_{2^m}, esto permitirá ganar aun más eficiencia en la aritmética de puntos y por ende un mayor rendimiento en los procesos criptográficos.

2. Dado que el sistema es modular y extensible se pueden implementar nuevos módulos que permitan la cohesión o acoplamiento con otras aplicaciones que también operen en ambientes Web, de tal forma que éstas hagan uso de los algoritmos de cifrado y firmado sin que necesariamente intervengan en el modelo de transferencia de información financiera, esto implicará una ampliación del protocolo HERMES y la exigencia de una gestión de memoria temporal de forma segura.

3. Por último y no menos importante, hacer uso de la *biometría* como forma única de autenticación en todo el sistema, en especial la identificación biométrica basada en huellas dactilares puesto que los dispositivos escáneres para toma de huellas son más asequibles y accesibles que los dispositivos para reconocimiento del iris o los de reconocimientos de patrones de venas y arterias en la palma de la mano, entre otros; también porque dado el estado del arte actual, la huella dactilar es la segunda característica humana conocida más singular[46] (después del ADN) lo que garantizaría un nivel muy alto en la seguridad si utilizáramos la secuencia de bits de las *minucias* de la huella como clave privada en nuestro esquema de cifrado de curva elíptica.

[46] Se ha calculado que la probabilidad de que dos personas tengan la misma huella dactilar es de 1 / 67 billones.

Anexo A. CRONOGRAMA

TIEMPO	Mes 01				Mes 02				Mes 03				Mes 04				Mes 05			
FASES	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20
INICIO																				
Recopilar la información bibliográfica.																				
Estudio matemático extenso sobre los conceptos de curvas elípticas.																				
Consecución del equipo hardware necesario para la elaboración del proyecto.																				
Estudio de la plataforma y el lenguaje de programación a utilizar.																				
ANALISIS																				
Especificar el problema y visualizar soluciones a dichos estudios.																				
Evaluar costo/beneficio de implementación y concretar el software a utilizar.																				
Análisis de una política de seguridad general.																				
DISEÑO																				
Especificación de requisitos del sistema																				
Selección de los casos de usos funcionales.																				
Diseño del modelo de Datos.																				
Diseño de política de copias de respaldo a la base de datos.																				
Esbozo del sistema de firma digital.																				
Especificación de requisitos del software																				
Diseño diagrama Entidad-Relación y diagrama de Estados.																				
Diseño de las diferentes interfaces de usuario.																				
Diseño detallado de funciones, clases y librerías a utilizar.																				
IMPLEMENTACION																				
Implantación de la base de datos.																				
Implementación en software de las copias de respaldo.																				
Implementar las clases y librerías.																				
Creación del procedimiento de firma digital y emulación de entidad certificadora.																				
Desarrollo del modulo de mantenimiento de tablas y variables de entornos.																				
Implementar casos de usos establecidos previamente.																				
Elaboración del manual de usuario.																				
TRANSICION																				
Efectuar las pruebas de verificación de la interfaz.																				
Evaluar los algoritmos implementados.																				
Evaluación general de un prototipo funcional.																				

Anexo B. GLOSARIO.

- **Anillo (Anillo de Claves):** En el sistema, el anillo de claves es el conjunto de todas las claves públicas (inclusive las propias) que podemos utilizar para cifrar un documento. Por restricciones del sistema, las claves existentes en el anillo deben estar asociadas solo a entidades financieras o al repositorio.

- **Área (o Módulo):** En el sistema, un área o módulo es la *agrupación de tareas* generalmente comunes o relacionadas entre si. Cada entrada en el menú principal del criptosistema HERMES corresponde a un área.

  Ejemplos de áreas son:
  - Sistema
  - Giro
  - Ayuda

- **Auditoría** (**Auditoría informática**): Es el conjunto de actividades encaminadas a la validación y verificación de los sistemas, procesos y resultados en los que se utilicen tecnologías automatizadas, ya sea en cumplimiento de la legislación, como garantía de la integridad y correctitud de la información aportada por un sistema o por alineamiento con determinados estándares relacionados con el buen uso (best practices) de los sistemas.

- **Autenticación** (**Autentificación**): Autenticación (Griego: αυθεντικός = *verdadero o genuino*, de "los authentes" = *el autor*) es el acto de establecimiento o confirmación de algo (o alguien) como auténtico, es decir que reclama hecho por o sobre la cosa son verdadero. La autenticación de un objeto puede significar (pensar) la confirmación de su procedencia, mientras que la autenticación de una persona a menudo consiste en verificar su identidad.
  Tomado de http://es.wikipedia.org/wiki/Autenticación

- **Base 36** (**Sistema de numeración en base 36**): El sistema de numeración *Base 36*, sistema *hexatridecimal*, sistema *sexatrigesimal*, sistema *hexatrigesimal* o simplemente sistema *alfadecimal* (éste último un neologismo) es un sistema de numeración posicional que utiliza 36 como base. En informática, la selección de 36 como base es conveniente puesto que los dígitos pueden ser representados usando los diez números arábigos (0-9) y las 26 letras del alfabeto latino (A-Z). Desde el punto de vista matemático, la selección es conveniente puesto que 36 es divisible por 2 y 3 (y sus múltiplos 4, 6, 9, 12 y 18).

- **Caso de Uso:** Es una descripción narrativa textual de la secuencia de eventos y acciones que ocurren en un sistema durante un proceso significativo.

- **Cliente** (**Clientes del Sistema**): En el sistema, un cliente es cualquier particular que actúa como remitente o destinatario de un giro.

- **Entidad Financiera** (**Punto de Giro**): Una Entidad Financiera o Punto de Giro es cada una de los lugares físicos o lógicos que actuando como intermediario o

"mensajero" permite a un tercero (cliente) enviar un valor en dinero a otro.
Cada entidad financiera tiene instalado y funcionando el sistema HERMES, a generado al menos un par de claves de cifrado asimétrico de curvas elípticas, posee una dirección IP y un puerto específico que permiten a las otras entidades financieras abrir un socket para la comunicación e intercambio de datos basado en el PROTOCOLO-HERMES.

- **Estado de Giro:** Un estados de giro es la condición o situación en la que se encuentra un giro desde que es registrado en el sistema hasta que es retirado por el destinatario. Los estados de giros manejados en el PROTOCOLO-HERMES son:

  1. Registrado sin enviar.
  2. Enviado sin confirmar.
  3. Enviado y confirmado.
  4. Enviado, confirmado y pagado.
  5. Recibido sin pagar.
  6. Recibido y pagado.
  7. Recibido, pagado e informado.

- **Giro** :
  - En el sistema, los giros son cada una de las ordenes que una persona o empresa actuando como remitente envía a otra persona o empresa destinataria a través de una entidad financiera.
    Los giros constituyen la información financiera objeto a proteger y por su naturaleza, requieren que se garantice los principales pilares de la seguridad informática (confidencialidad, autenticidad, integridad y no repudio).
  - Giro, "*es una orden escrita e incondicional que una persona dirige a otra. Va firmado por la persona que lo extiende y pide al destinatario de la misma que proceda el pago de una determinada suma a la vista en fecha futura que se determina, a la orden de una determinada persona, o bien al portador*".
    Tomado del glosario contable BusinessCol http://www.businesscol.com

- **Hermes** :
- Nombre del criptosistema desarrollado en este trabajo de grado.
- En la mitología griega Hermes es el dios de las fronteras y los viajeros que las cruzan, de los pastores y las vacadas, de los oradores, literatos y poetas, del atletismo, de los pesos y medidas y los inventos y el comercio en general, de los mentirosos y de la astucia de los ladrones. Como traductor, es el mensajero entre los dioses y los humanos. Un hallazgo afortunado era un *hermaion*. Un intérprete que cruza las fronteras con extraños es un *hermeneus*. De Hermes procede la palabra «*hermenéutica*» para el arte de interpretar los significados ocultos.
  Tomado de http://es.wikipedia.org/wiki/Hermes

- **HTML:** Acrónimo de *HyperText Markup Language* (Lenguaje de Marcado de Hipertexto), es el lenguaje utilizado para la elaboración de los documentos (*páginas web*) que se visualizan en la WWW.

- **HTTP:** *Hypertex Transfer Protocol* o protocolo de transferencia de hipertexto, es un conjunto de estándares que permiten la comunicación entre aplicaciones cliente y servidores en la WWW.

- **Informes de Pagos**: En el sistema, un informe de pagos es el proceso por el cual, la entidad local, actuando como destinataria, "avisa" o hace saber a las entidades financieras remitentes que los giros que éstas enviaron han sido pagados completamente, es decir, que ya se hizo el retiro del dinero por parte de la persona a quien iba dirigido el valor del giro.

- **Modelo de datos:**

  - El modelo de datos es la abstracción de una base de datos.

  - **"***Un modelo de datos es un sistema formal y abstracto que permite describir los datos de acuerdo con reglas y convenios predefinidos. Es formal pues los objetos del sistema se manipulan siguiendo reglas perfectamente definidas y utilizando exclusivamente los operadores definidos en el sistema, independientemente de lo que estos objetos y operadores puedan significar'.*[47]

- **Perfil de Usuario**: en el sistema, un perfil de usuario es la agrupación de una, ninguna o varias tareas, generalmente, relacionadas entre si. Los perfiles están diseñados para facilitar el control de acceso de los usuarios al sistema. Un usuario que tenga asociado un perfil en particular podrá acceder a todas las tareas que dicho perfil agrupa, esto evita tener que asignar una a una las tareas a cada usuario.

- **PHP**: Acrónimo de *Hypertext Preprocessor*, es un lenguaje *Open Source* interpretado de alto nivel, rápido, especialmente diseñado para desarrollos Web. La mayoría de su sintaxis es similar a C, Java y Perl. Es un lenguaje que se ejecuta en el servidor, cuenta con un gran número de funciones para acceso a bases de datos, conexiones de red, módulos para manejo de números grandes, compresión, etc.

- **Protocolo HERMES:** El protocolo HERMES es cada una de las reglas bien definidas, secuenciales y ordenadas que se sigue en este sistema para el intercambio de datos entre dos entidades financieras y entré estas y el repositorio.

  EL protocolo HERMES es un protocolo de nivel superior construido sobre el protocolo HTTP, éste a su vez basado en y/o parte de la familia de protocolos TCP/IP de más bajo nivel (transporte y red).

---

[47] Ullam, Jeffrey y Widom, Jennifer, *Introducción a los Sistemas de Bases de Datos.* Prentice Hall, Mexico 1999.

- **Reporte de Giro**: Es el proceso mediante el cual la entidad financiera local, actuando como remitente, genera un documento con la información de cada giro en los estados "Registrado sin enviar" y/o "Enviado Sin Confirmar", los cifra con la clave pública de la entidad destinataria, los firma con su clave privada y los envía utilizando un protocolo definido.

- **Script**: Es una secuencia de código sin compilar que puede ejecutar acciones mediante un software que la interpreta en lugar de ser procesado por un compilador.

- **Socket** (**Socket de Internet**): Socket designa un concepto abstracto por el cual dos programas (posiblemente situados en computadoras distintas) pueden intercambiarse cualquier flujo de datos, generalmente de manera fiable y ordenada. Un socket queda definido por una dirección IP, un protocolo y un número de puerto.
  Tomado de http://es.wikipedia.org/wiki/Socket_de_Internet

- **Tarea** (**Submódulo**): En el sistema, una *tarea* o *submódulo* es cada uno de los procedimientos individuales que se realizan para llevar a cabo un caso de uso en particular. Las tareas están agrupadas en áreas, cada entrada en el menú principal corresponde a un área y cada opción del menú principal corresponde a una tarea.

  Ejemplos de tareas son:
  - Cifrar Documento
  - Verificar Firma
  - Salir

- **Usuario del Sistema**: Un usuario del sistema o funcionario es cualquier persona con un nombre de usuario y contraseña válidos para acceso al sistema HERMES.

- **Usuario vs Cliente** (**Usuarios vs Clientes**): La principal diferencia entre un usuario y un cliente es que el cliente NO tiene acceso al sistema mientras que el usuario si. Un cliente es cualquier particular que actúa como remitente o destinatario de un giro; el cliente NO tiene un nombre de usuario y contraseña. El usuario en cambio, hace uso del sistema, puede (si tiene los perfiles adecuados) consultar giros, registrar giros, cifrar documentos, firmar documentos, etc.

- **WWW:** Acrónimo de *World Wide Web,* es un sistema de información distribuido que hace parte de la red de internet y es utilizada para el intercambio de documentos de hipertexto a través del protocolo HTTP.