INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA

SECCIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

UNIDAD CULHUACAN

"DISEÑO DE ARQUITECTURAS ARITMÉTICAS DE CAMPO FINITO EN VHDL CON APLICACIONES EN SISTEMAS DE CRIPTOGRAFÍA DE CURVA ELÍPTICA"

T E S I S

QUE PARA OBTENER EL GRADO DE

MAESTRO EN INGENIERÍA EN SEGURIDAD Y TECNOLOGÍAS DE LA INFORMACIÓN

PRESENTA

JOSÉ ANTONIO FLORES ESCOBAR

ASESORES:

DR. MOISÉS SALINAS ROSALES

DR. JOSÉ VELÁZQUEZ LÓPEZ

MÉXICO D.F. MARZO 2012

SIP-14-BIS

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D. F. siendo las 16:00 horas del día 1° del mes de diciembre del 2011 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación de SEPI ESIME CULHUACAN para examinar la tesis titulada:

"Diseño de Arquitecturas Aritméticas de Campo Finito en VHDL con Aplicaciones en Sistemas de Criptografía de Curva Elíptica"

Presentada por el alumno:

Flores	**Escobar**	**José Antonio**
Apellido paterno	Apellido materno	Nombre(s)

Con registro: A 1 0 0 6 8 0

aspirante de:

MAESTRÍA EN INGENIERÍA EN SEGURIDAD Y TECNOLOGÍAS DE LA INFORMACIÓN

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Directores de tesis

Dr. Moisés Salinas Rosales

Dr. José Velázquez López

Dr. Juan Carlos Sánchez García

Dr. Gonzalo Isaac Duchen Sánchez

Dra. María de Lourdes López García

INSTITUTO POLITECNICO NACIONAL
ESTADOS UNIDOS MEXICANOS
S.E.P.
SECCION DE ESTUDIOS DE POSGRADO E INVESTIGACION
ESIME CULHUACAN

PRESIDENTE DEL COLEGIO DE PROFESORES

Dr. Gonzalo Isaac Duchen Sánchez

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de _México, D.F._ el día _1_ del mes _de Diciembre_ del año _2011_, el (la) que suscribe _José Antonio Flores Escobar_ alumno (a) del Programa de _Maestría en Ingeniería en Seguridad y Tecnologías de la Información_ con número de registro _A100680_, adscrito a _la Sección de Estudios de Posgrado e Investigación de la ESIME Unidad Culhuacan_, manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección de _Dr. Moisés Salinas Rosales y Dr. José Velázquez López_ y cede los derechos del trabajo intitulado _**"Diseño de Arquitecturas Aritméticas de Campo Finito en VHDL con aplicaciones en Sistemas de Criptografía de Curva Elíptica"**_, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección _jflorese0902@ipn.mx, msalinasr@ipn.mx, jvelazquezl@ipn.mx_. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

José Antonio Flores Escobar

Nombre y firma

A mis padres, porque su apoyo durante este proyecto fue total e incondicional, porque su ejemplo ha sido decisivo para lograr esta meta.

A mis hermanas cuyos logros han sido una inspiración para concluir esta etapa de mi vida, por su apoyo ¡muchas gracias!

A Clio por su comprensión, su tiempo, sus consejos y por siempre estar a mi lado.

AGRADECIMIENTOS

Al Dr. Moisés Salinas Rosales por haberme aceptado en el Grupo de Sistemas Criptográficos, por ser mi guía durante este proyecto y por todo su apoyo incondicional a lo largo de mi estancia en el IPN.

Al Dr. José Velázquez López por su paciencia en las materias, sus comentarios y sus valiosas sugerencias, por toda la ayuda brindada para concluir este proyecto.

Al Instituto Politécnico Nacional por darme la oportunidad de estudiar la maestría en sus instalaciones.

A mi comité revisor integrado por el Dr. Juan Carlos Sánchez García, Dr. Gonzalo Isaac Duchen Sánchez y Dra. María de Lourdes López García.

A Agus, Ede, Gus, Jorge (citados alfabéticamente para evitar malos entendidos) por su valiosa amistad durante estos dos años y medio, por las tardes de café, por todas las celebraciones, por los viajes... y por todas las dudas resueltas en el transcurso de la maestría.

A Hito, Luis, Orrala, Wolf, Ilce, Arthur, Doris y Sergio por su invaluable amistad y por su apoyo en todo, gracias chav@s.

Al Dr. Gualberto por su apoyo en PIFI y a la Dra. Gina por su ayuda, y a toda la comunidad MISTI, por apoyar sin condiciones y siempre compartir su conocimiento.

El presente trabajo ha sido realizado en el Laboratorio de Sistemas Digitales de la Sección de Estudios de Posgrado e Investigación, en la Escuela Superior de Ingeniería Mecánica y Eléctrica Unidad Culhuacan, del Instituto Politécnico Nacional, bajo el asesoramiento del Dr. Moisés Salinas Rosales y del Dr. José Velázquez López, con el apoyo de CONACyT bajo el número de CVU 347497.

RESUMEN

En esta tesis se presenta un multiplicador polinomial que opera elementos del campo GF(2^{192}). Este multiplicador ha sido basado en dos esquemas que funcionan de manera conjunta: torres de campos y multiplicadores simétricos de bloques.

Para lograr dicho multiplicador, se han propuesto 4 multiplicadores base, de los cuales, en base a las directrices de diseño además de los requerimientos de seguridad, ha sido seleccionado un multiplicador que sirva como base para implementar un multiplicador GF($((2^8)^6)^4$) equivalente a un multiplicador GF(2^{192}).

Las directrices de diseño propuestas para este multiplicador son la eficiencia y el rendimiento, así mismo, los requerimientos de seguridad se basan en el ANSI x9.62 el cual dicta que una aplicación es considerada segura si opera elementos de un campo finito mayor GF(2^{160}) representados con una base como lo es la polinomial.

Los resultados obtenidos en esta tesis se encuentran dentro de los parámetros esperados, alcanzando el objetivo que se planteó al inicio del desarrollo.

ABSTRACT

In this thesis a polynomial multiplier, that operates elements of GF(2^{192}), is presented. The multiplier is based in two schemes working together: tower fields and symmetric multipliers.

In order to achieve the multiplier, four base multipliers have been proposed, one of them, and based in some design guidelines besides the security requirements, has been selected as a base multiplier for implementing the multiplier GF(((2^8)^6)^4) which is equivalent to GF(2^{192}).

The design guidelines for the proposed multiplier are efficiency and throughput; also, the security requirements are based in ANSI x9.62 which states that an application is secure if operates elements of a finite field bigger than GF(2^{192}), with a standard basis, such as a polynomial basis.

The results obtained in this thesis are within the expected parameters, reaching the main objective proposed at the beginning of this work.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Actualmente, y a partir del uso masivo de computadoras, telefonía celular, etc., la comunicación sobre redes públicas y el almacenamiento digital de información juegan un rol importante en la vida diaria, los sistemas de comunicación ayudan a tener una mayor disponibilidad de la información, de todo tipo, ya que su objetivo principal es la transmisión de la información. La seguridad de la información es sumamente importante para cualquier organización, y para evitar que esta información sea obtenida por personas u organizaciones no autorizadas se desarrollan e implementan sistemas criptográficos [1].

La criptografía es una disciplina que trata sobre el diseño y análisis de técnicas matemáticas, que ayudan a establecer comunicaciones seguras sobre canales inseguros, es decir, la criptografía provee herramientas eficientes para mantener segura la información, lo cual es su principal objetivo [2]. La criptografía busca proveer objetivos de seguridad tales como la confidencialidad, integridad, autenticación y no repudio [3]. La criptografía, de manera general, puede clasificarse en dos clases: criptografía simétrica y criptografía asimétrica, también llamada criptografía de clave pública (PKC, por sus siglas en inglés) [1]. La simétrica esencialmente se basa en compartir una clave secreta entre las entidades que desean comunicarse, en la asimétrica las entidades poseen un par de claves, que consiste en una clave privada y una clave pública [4].

A partir de la propuesta del uso de Curvas Elípticas en PKC [5,6], surge la Criptografía de Curva Elíptica (ECC, por sus siglas en inglés). A diferencia de otros sistemas criptográficos de clave pública, ECC ha demostrado, bit a bit, un mejor balance entre seguridad y eficiencia, los sistemas de ECC basan su seguridad en la intratabilidad del Problema del Logaritmo Discreto de Curva Elíptica (ECDLP, por sus siglas en inglés), con

lo cual ofrece mayor seguridad incluso con una clave de longitud menor, comparada con otros sistemas criptográficos de clave pública [7]. La aritmética de ECC está construida en términos de un campo finito, y es por esta razón que, en los últimos años, los campos finitos han tomado un papel sumamente importante en las implementaciones de ECC.

Los campos finitos, también llamados campos de Galois, por sus propiedades, tienen numerosas aplicaciones en implementaciones criptográficas, teoría de códigos y procesamiento digital de señales [8]. Esencialmente, existen 3 tipos de campos finitos que son especialmente importantes para una implementación eficiente de sistemas de curva elíptica: campos binarios, campos primos y extensiones óptimas del campo [1].

Los campos finitos binarios, o de característica 2, son muy atractivos para aplicaciones prácticas. En el caso de implementaciones en hardware han sido propuestos diferentes algoritmos, los cuales reducen la complejidad de las operaciones aritméticas del campo, permitiendo una eficiente operación aritmética del campo, tal como la multiplicación y la inversión [2].

Los componentes aritméticos de campos finitos implementados en hardware, para aplicaciones en ECC, a diferencia de los implementados en software, ofrecen una mayor velocidad y ancho de banda, posibilitando el cifrado en tiempo real. Existen dos alternativas para implementar dichos componentes aritméticos: FPGA y VLSI [2]. Los FPGA's ofrecen beneficios como flexibilidad y un bajo costo, además de ofrecer ventajas como el ser reconfigurables y poder ser reprogramados incluso en tiempo de ejecución, otra de sus ventajas es el que virtualmente cualquier circuito digital puede ser implementado en ellos [9].

PROBLEMÁTICA

Existen diversas amenazas para la protección de la información, y debido a esas amenazas la seguridad informática ha tomado un gran auge. Una medida de protección es la seguridad lógica, que tiene como función principal proteger los datos de cualquier entidad, y además puede ser implementada tanto en software como en hardware. Una de las vertientes principales es el desarrollo sistemas criptográficos que ayuden a mantener una completa confidencialidad, integridad, autenticación y no repudio.

La posibilidad de interconexión en casi cualquier parte del mundo ha abierto nuevos horizontes, pero a su vez ha hecho surgir nuevas amenazas para los sistemas y la información que resguardan. Gracias al avance conjunto de diferentes áreas, los dispositivos con los que se realizan interconexiones móviles son cada vez más pequeños, se debe recordar que el procesamiento se vuelve más poderoso, más barato y además cuenta con una mayor disponibilidad, y por ende, las amenazas son más latentes, así que los requerimientos de seguridad se deben volver más estrictos.

En un procesador de ECC los componentes aritméticos de campo finito son cruciales, es muy importante lograr una relación área/tiempo eficiente, sobre todo en escenarios donde los recursos y el consumo de energía están restringidos, porque es entonces cuando esos recursos se vuelven muy valiosos. ECC usa llaves considerablemente más pequeñas ofreciendo el mismo nivel de seguridad que otros sistemas criptográficos de clave pública, lo cual se resume en un mejor balance entre seguridad/eficiencia.

JUSTIFICACIÓN

Resulta necesario proponer el diseño de componentes aritméticos de alta eficiencia, con aplicaciones para criptografía de curva elíptica, que se ajusten a los requerimientos de entornos restringidos, que ayuden a mejorar los tiempos y la eficiencia en el cifrado y descifrado, que ofrezcan la mayor seguridad con los mínimos requerimientos y que además sea viable una implementación futura.

En el caso de ambientes con recursos restringidos es necesario contar con componentes que usen la menor cantidad energía, que tengan el mayor rendimiento y además puedan ser implementados en áreas pequeñas.

OBJETIVO GENERAL

Diseñar un multiplicador de campo finito, para auxiliar el desarrollo de operaciones de criptografía de curva elíptica en ambientes de recursos restringidos, por medio de tecnología de hardware reconfigurable.

OBJETIVOS ESPECÍFICOS

- Elegir el campo finito base, sobre el cual se definirá la aritmética y su técnica de representación, con base en las especificaciones del NIST, para lograr la mayor eficiencia y seguridad posible.
- Seleccionar algoritmos, a través de revisión bibliográfica, para el desarrollo de la multiplicación en campo finito.
- Diseñar e implementar la arquitectura de hardware, en VHDL, para desarrollar la multiplicación en campo finito.
- Evaluar el desempeño del la arquitectura propuesta, a partir de pruebas de ejecución, para identificar posibles mejoras.

CAPÍTULO 1.

SEGURIDAD INFORMÁTICA

El objetivo de la seguridad informática es brindar servicios como la confidencialidad, integridad, autenticación y no repudio, servicios que guardan una estrecha relación entre ellos [2]. La criptografía, mediante técnicas matemáticas, ayuda a proporcionar dichos servicios [1]. Por tal motivo, al programar aplicaciones seguras, se busca que al menos en una de sus capas se implemente criptografía.

Debido a las ventajas que ofrece, con respecto a otros sistemas criptográficos, la aplicación de las arquitecturas aritméticas se enfocará a Criptografía de Curva Elíptica, haciendo énfasis en que de acuerdo al modelo jerárquico para Criptografía de Curva Elíptica, presentado en este capítulo, la capa base está soportada por la aritmética de campo finito.

1.1. NOCIONES BÁSICAS DE SEGURIDAD

Normalmente, la comunicación entre las entidades, sucede en un canal inseguro, lo que implica que la información que se intercambia no cuente con garantías de privacidad ni de integridad, por esta razón, es necesario proveer mecanismos y/o servicios que brinden seguridad a la información, estos mecanismos y/o servicios de seguridad son construidos mediante herramientas criptográficas [1]. Debido al creciente uso de la redes, sin excepción, todas las computadoras y, en general, cualquier dispositivo conectado a cualquier red debe tener al menos una capa de criptografía implementada, así mismo ese dispositivo será capaz de acceder a funciones criptográficas con el fin de proveer seguridad [2].

El esquema básico de comunicación se presenta en la fig. 1, supongamos que si A desea comunicarse con B, sobre un canal inseguro, se tiene que asumir que toda la comunicación se establece en presencia de un adversario E, que tiene como objetivo violar cualquier servicio de seguridad provisto por A a B.

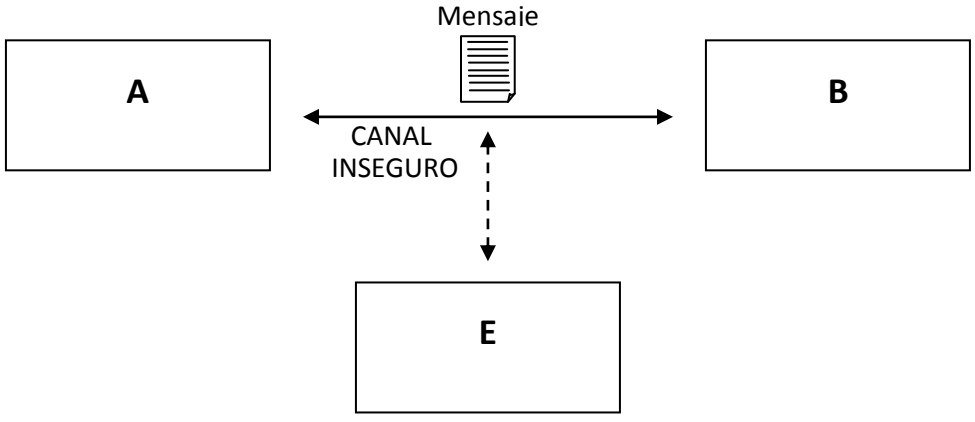

Figura 1: Esquema básico de comunicación.

Existe una relación estrecha entre el desarrollo de sistemas computacionales de alto rendimiento y la criptografía. A medida que los

sistemas criptográficos se vuelven más complejos, existe la necesidad de desarrollar métodos de criptoanálisis más eficientes, ayudados por el rápido crecimiento del poder de cómputo [2].

De manera general, la seguridad busca proveer los siguientes servicios, los cuales están interrelacionados entre ellos [3]:

1. *Confidencialidad*: Se debe mantener el contenido de la información en secreto excepto para aquellos que están autorizados a tener esa información.
2. *Integridad*: Es la propiedad de mantener la información sin alteraciones por personas no autorizadas.
3. *Autenticación*: Poder corroborar la identidad de una persona.
4. *No Repudio*: Prevenir que una entidad niegue compromisos o acciones anteriores.
5. *Control de Acceso*: Proteger los recursos del sistema.

La fig. 2 presenta un modelo jerárquico de 6 capas para aplicaciones de seguridad de la información [2]. En la capa 6 se muestran aplicaciones de seguridad, las cuales es muy común usar todos los días, tales como: monedero electrónico, correo electrónico seguro, corta fuegos, voto electrónico, por nombrar algunas de las más populares; estas aplicaciones dependen de la correcta implementación de protocolos de autenticación, por ejemplo: SSL/TLS, IPSec, IEEE 802.11, etc, los cuales se encuentran en la capa 5; sin embargo, estos protocolos no son seguros si no se implementa la capa 4, que trata acerca de los servicios de seguridad que han sido descritos con anterioridad: confidencialidad, integridad, autenticación y no repudio; la infraestructura básica para dichos servicios se soporta por las primitivas criptográficas: cifrar/descifrar y firmar/verificar, que son descritas en la capa

3; las primitivas criptográficas pueden ser implementadas mediante algoritmos de clave pública (RSA, DSA, ECC, emparejamientos, etc.) y/o clave privada (DES, AES, RC4, Camellia, etc.) que se encuentran en la capa 2; por último, cuando el objetivo es obtener un alto rendimiento/desempeño de los algoritmos criptográficos es crucial contar con implementaciones eficientes de operaciones aritméticas tales como la suma, resta, multiplicación, inversión, etc., como se muestra en la capa 1.

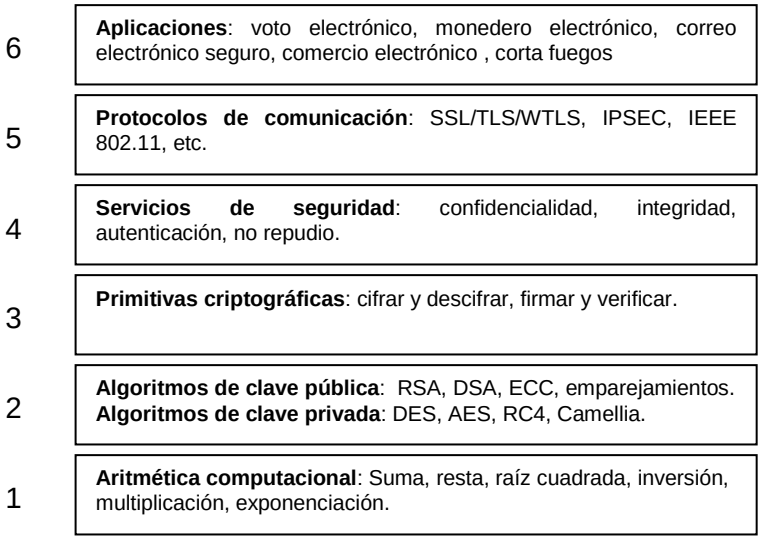

Figura 2. Esquema jerárquico para aplicaciones de seguridad informática

1.2. CRIPTOGRAFÍA

La criptografía es la disciplina que estudia el diseño y el análisis de técnicas matemáticas que nos ayudan a establecer comunicaciones inseguras incluso en presencia de adversarios maliciosos; donde existe el ideal de que aún cuando esos adversarios maliciosos tuvieran a su disposición todo el poder de cómputo de la Tierra, serían incapaces de obtener la información resguardada con criptografía [1].

La palabra Criptografía se deriva de las palabras griegas kryptos, que significa escondido, y graphien, la cual significa escribir. La criptografía busca cumplir con los objetivos que persigue la seguridad: Confidencialidad, Integridad, Autenticación y No Repudio [3].

Un sistema criptográfico, o criptosistema, se forma por la quíntupla ($\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D}$), donde se mantienen las siguientes condiciones [3]:

1. $\mathcal{P}$ es un conjunto finito de posibles textos claros
2. $\mathcal{C}$ es un conjunto finito de posibles textos cifrados
3. $\mathcal{K}$ es el espacio de claves, es un conjunto finito de posibles claves
4. Para cada $K \in \mathcal{K}$ existe una regla de cifrado $e_K \in \mathcal{E}$ y una regla de descifrado correspondiente $d_K \in \mathcal{D}$. Cada $e_K : \mathcal{P} \rightarrow \mathcal{C}$ y $d_K : \mathcal{C} \rightarrow \mathcal{P}$ son funciones tales que $d_K(e_K(x))=x$ para cada elemento $x \in \mathcal{P}$.

La primitiva de cifrado $e_K : \mathcal{P} \rightarrow \mathcal{C}$ consiste en que a partir de un texto claro, se aplica un algoritmo criptográfico, y se obtiene un texto cifrado. Este algoritmo criptográfico funciona como una transformación *E*, para obtener el texto claro a partir del texto cifrado d_K: $\mathcal{C} \rightarrow \mathcal{P}$, que se conoce como descifrado, simplemente se aplica la transformación inversa E^{-1}, el esquema básico de cifrado se muestra en la fig. 3. Un sistema criptográfico puede ocultar la información cifrándola antes de enviarla o antes de almacenarla.

Figura 3: Esquema básico general de cifrado.

Criptografía Clásica

Desde que el hombre aprendió a comunicarse, ha tenido presente la necesidad de proteger su información, como consecuencia, está en una búsqueda constante de herramientas o métodos para transformar o cifrar la información a enviar, con la finalidad de mantenerla en secreto o de manera confidencial. Así es como nace la criptografía, que a grandes rasgos se refiere al arte de ocultar información, a través de símbolos o caracteres desconocidos.

Los primeros resultados de dicha búsqueda datan de hace más de 4500 años, y pueden verse con el uso de los jeroglíficos tallados en monumentos del Antiguo Egipto. La *Escitala*, un *método de transposición*, es el primer método utilizado para comunicarse de una forma segura, fue utilizado en la Grecia del siglo VI A.C., en este método, tanto emisor como receptor debían de tener la misma clave para poder comunicarse [10]. Posteriormente, en Roma, durante el siglo I A.C., fue utilizado el *Cifrado César*, este *método de sustitución*, era considerado uno de los más simples, se puede apreciar en la fig. 4, donde también emisor y receptor compartían la misma clave [11].

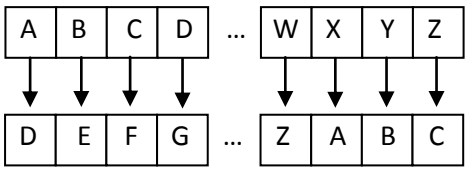

Figura 4: Cifrado César

Siglos más tarde, durante la Edad Media y hasta el siglo XIII no dejaron de utilizarse los *métodos de sustitución*, durante este siglo, la escritura oculta fue practicada en varias ciudades italianas, y mayormente

por la iglesia católica, sin embargo, es hasta los siglos XIV y XV cuando se dieron grandes progresos para proteger la información [12].

En el siglo XVI, Blaise de Vigenère desarrolló una importante teoría sobre las transformaciones polialfabéticas, lo que hoy se conoce como el *Tablero de Vigenère* [13], el cual tuvo una vigencia hasta el siglo XIX, y es durante este siglo cuando Auguste Kerckhoffs, en su artículo *La Criptografía Militar*, estableció importantes principios de la criptografía moderna [14].

A principios del siglo XX, y debido a la 1ª y 2ª Guerra Mundial, el uso de la criptografía en las transmisiones de información cobra una mayor importancia, originando esto un gran auge tanto en las técnicas como de las máquinas de cifrar; en Alemania, se hizo gran uso de diversas variantes de una máquina de rotores electromecánica llamada Enigma [15], como la que se muestra en la fig. 5, es entonces cuando da inicio el criptoanálisis, el cual consiste en descifrar la información en clave, rompiendo así el sistema criptográfico; en Japón se introdujo una familia de máquinas para cifrar las cuales implementaron sustituciones polialfabéticas. A cada una de estas máquinas se les asignaba un color, siendo la máquina *RED*, de color rojo, la primera de esta familia [16].

Figura 5: Máquina Enigma de 3 rotores

Criptografía Moderna

Tras la conclusión de la segunda guerra mundial, la criptografía tiene un desarrollo teórico importante, porque es cuando Claude Shannon, en 1948, publica *A Mathematical Theory of Communication*, donde formaliza el estudio de la *Teoría de la Información*, y es en este artículo donde se demuestra que un texto cifrado no da ninguna información acerca del texto claro y que a mayor longitud de la clave, mayor seguridad tendrá el criptosistema [17], por tal motivo, Shannon es conocido como el padre de la *Teoría de la Información* [18].

En 1966, Horst Fiestel describió el papel que juega la Criptografía en proporcionar la privacidad en los Sistemas Computacionales, sugirió una "plantilla" de diseño para la elaboración de algoritmos de cifrado, todo esto dio inicio a un gran cambio que años mas tarde terminaría en la criptografía asimétrica [16].

Posteriormente, en el año de 1976, Whitfield Diffie y Martin Hellman, publican el artículo *New Directions in Cryptography*, el cual es un estudio realizado que trata sobre la aplicación de funciones matemáticas de un solo sentido a un modelo de cifrado, denominado cifrado con clave pública [19], dejando atrás el método de clave compartida, que también es llamado criptografía simétrica.

1.2.1. CRIPTOGRAFÍA SIMÉTRICA

En la criptografía simétrica existe solo una única clave, llamada secreta, la cual deben compartir emisor y receptor, con esta única clave se cifra y se descifra. La seguridad de la criptografía simétrica reside en el hecho de mantener la clave en secreto.

La mayor ventaja de la Criptografía Simétrica es la alta eficiencia, aunque tenga desventajas como el problema de la distribución de claves,

donde, es altamente recomendable, distribuirlas mediante un canal seguro y autenticado, otra desventaja es la administración de claves, en una red de *N* entidades, cada entidad debe tener una clave diferente para cada una de las otras *N-1* entidades, y por último, la Criptografía Simétrica no resulta ideal para los Esquemas de Firma Digital que proveen el No Repudio [1].

Los sistemas criptográficos simétricos pueden dividirse, de acuerdo a sus formas de operar, en *cifradores de bloque* y *cifradores de flujo*.

A. Cifradores de Bloque

Un esquema de cifrado por bloques opera con un grupo de bits de cierta longitud, llamado bloque, en conjunto con una transformación que no cambia [3].

Como ejemplo, la entrada de un cifrador de bloque puede ser un texto claro de 128 bits de longitud, la salida corresponde a un texto cifrado de la misma longitud en bits, donde en la transformación existe la figura de la clave, que puede tener la misma longitud del texto en claro, o puede ser más grande, siendo que a mayor longitud de la clave, más seguridad tendrá el criptosistema.

Un mensaje que sea de una longitud mayor que la longitud del bloque de entrada puede cifrarse si el mensaje se divide en *n* bloques, cada bloque deberá cifrarse individualmente, esto se conoce como Modos de Operación [20].

Los cifradores de bloque vigentes, de acuerdo al Instituto Nacional de Estándares y Tecnología (NIST, por sus siglas en inglés), son AES, Triple DES y Skipjack.

- AES (Advanced Encryption Standard): Es un estándar de cifrado, fue publicado el 26 de noviembre de 2001, y aceptado en el 2002, aunque

es hasta el 2006 cuando se vuelve el estándar de cifrado con mayor uso. Funciona con boques de 128 bits, y opera con claves de 128, 192 o 256 bits [21]. Algunos de sus usos más populares son Soluciones Empresariales de BlackBerry [22], cajeros automáticos [23], además de ser el estándar de cifrado para el Gobierno de los Estados Unidos.

- Triple DES (Triple Data Encryption Standard): Es un algoritmo que hace tres veces el cifrado DES. Triple DES es ampliamente usado y es reconocido como seguro, ya que hace uso de 2 o 3 claves [24]. Este algoritmo se hizo estándar en 2004. Es ampliamente usado, en Tarjetas de Crédito y algunos otros sistemas de pago electrónico, a pesar de su lentitud.

- Skipjack: Es un algoritmo de cifrado cuyo diseño mucho tiempo se mantuvo en secreto. Las primeras referencias aparecieron en el FIPS 185 en Febrero de 1984, y en Mayo de 1998 el NIST lo hizo público. Skipjack, normalmente, está implementado en hardware de propósito específico [25].

Existen más cifradores de bloque en uso, que si bien no son estándares del NIST, se usan en demasiadas aplicaciones, entre ellos se encuentran MISTY y Camellia, por mencionar algunos.

B. Cifradores de Flujo

Un algoritmo de cifrado de flujo convierte el texto en claro en texto cifrado bit a bit, esto es, que una secuencia de texto claro $m_0m_1…m_n$ se cifra en un texto cifrado $c_0c_1…c_n$.

Los cifradores de flujo tienen diferentes ventajas que los hacen más adecuados para algunas aplicaciones, usualmente son más rápidos, además de tener, en hardware, una complejidad más baja que los cifradores de bloque, son adecuados cuando el buffer es limitado, y en algunos casos, no

se ven afectados por errores de propagación [26], las llamadas telefónicas son un ejemplo de aplicación de cifradores de flujo.

Algunos de los cifradores de flujo más conocidos y usados son A5/2, RC4.

- RC4: Fue desarrollado por RSA Data Security. RC4 se usa en protocolos como Secure Socket Layer (SSL) con el fin de proteger el tráfico de internet, en Wired Equivalent Privacy (WEP) para asegurar redes inalámbricas. RC4 es simple además de ser veloz en implementaciones en Hardware [26].

- A5/2: Es un cifrador de flujo que proporcionó seguridad en llamadas de telefonía celular con el protocolo GSM, a partir de Julio de 2006, los teléfonos móviles no soportan A5/2, debido a su debilidad, pues fue comprobado que con pocos recursos e incluso en tiempo real, este cifrador podía se roto [27].

1.2.2. CRIPTOGRAFÍA ASIMÉTRICA

A mediados de 1970 Diffie y Hellman imaginaron un mundo en el que los sistemas de cifrado utilizaran una clave para cifrar y una diferente para descifrar. Es aquí donde surge la criptografía de clave pública (PKC), la cual ha proporcionado una serie de servicios que con la criptografía simétrica eran inalcanzables [1].

PKC es un tipo de criptografía que se utiliza para desarrollar diferentes primitivas criptográficas como es el cifrado y la firma digital. En ambos casos la PKC proporciona los algoritmos que ayudan a cifrar información, mediante el uso de dos claves, una pública y una privada. La clave pública como su nombre lo indica se da a conocer públicamente y la clave privada se mantiene en resguardo del propietario del par de claves. El hecho que la clave pública sea conocida, no implica que se vaya a obtener la clave

privada, debido a que los algoritmos de clave pública basan su funcionamiento en funciones matemáticas de una solo vía [7]. El par de claves se generada por cada usuario que así lo requiera, utilizando un algoritmo de generación de claves asimétrico, dicho algoritmo es particular para cada sistema criptográfico a utilizarse.

La seguridad de la criptografía asimétrica, o de clave pública, reside en la dificultad computacional de descubrir la clave privada a partir del conocimiento de la pública [7].

En la primitiva criptográfica de cifrado, la clave pública se utiliza para cifrar la información, mientras que la clave privada es utilizada para descifrarla, garantizando de este modo que la información se mantenga en secreto, para aquellas personas no autorizadas, como se muestra en la fig. 6. Dicha transformación de cifrado se lleva a cabo mediante el algoritmo de cifrado del sistema criptográfico que se esté utilizando. En ella la información, también conocida como texto claro es transformada o cifrada en una forma incomprensible, llamada texto cifrado, y para poder regresar a su estado original, necesita de un proceso inverso ó de descifrado, donde transforma el texto cifrado a texto claro.

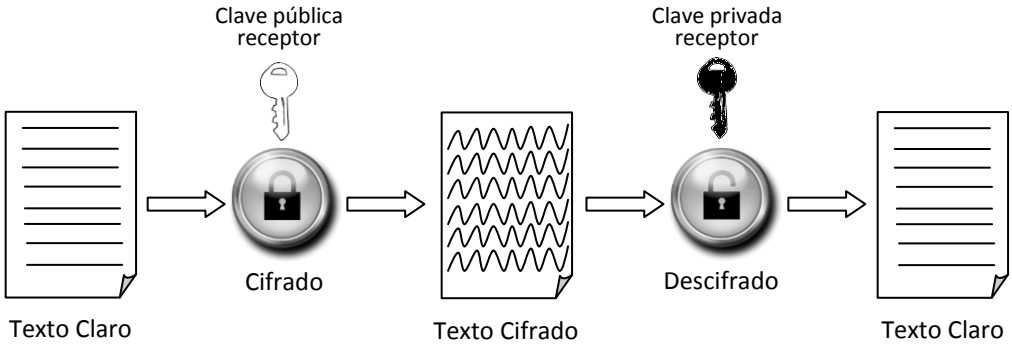

Figura 6. Esquema básico de Cifrado y Descifrado en PKC.

Entre los sistemas criptográficos de clave pública más usados se encuentran:

- RSA: Fue desarrollado en 1978 por Ronald Rivest, Adi Shamir y Leonard Adleman [28]. RSA es muy utilizado, tanto para cifrar como para firmar digitalmente y su seguridad se basa en el problema para factorizar números enteros grandes[11]. RSA seguirá siendo seguro mientras no se conozcan formas rápidas o viables para factorizar un número suficientemente grande en productos primos.

- ElGamal: Fue desarrollado por Taher ElGamal en 1984, es un esquema de cifrado basado en el intercambio de llaves Diffie-Hellman. Es un esquema probabilistico, lo que significa que a partir de un texto claro se pueden obtener diferentes textos cifrados [29].

- ECC (Elliptic Curve Cryptography): Miller y Koblitz propusieron el uso de curvas elípticas en PKC, y a partir de ese momento surge la ECC [5,6], que basa su seguridad en el problema matemático del logaritmo discreto de curva elíptica, lo que significa, como consecuencia, la misma seguridad que otros criptosistemas asiméticos con una clave de menor longitud [30].

RSA (www.rsa.com) y ECC (www.certicom.com), comercialmente, son los únicos criptosistemas asimétricos viables, aunque se espera que ECC reemplace a RSA, por sus ventajas tecnológicas, por ejemplo, su mayor eficiencia y aprovechamiento de los recursos computacionales [31].

El problema central en el uso de la criptografía de clave pública reside en la certeza de saber que, realmente, una clave pública pertenece a la persona o entidad de quién dice ser. De manera general sólo se menciona que el enfoque habitual a este problema consiste en utilizar una Infraestructura de Clave Pública, PKI por sus siglas en inglés, en donde uno de sus componentes principales son las Autoridades Certificadoras, CA. No obstante, el problema con el uso de una PKI, es que en ocasiones la CA llega a saturarse con la solicitud de certificados, razón por la cual es aquí

donde surge el uso de la Criptografía de Clave Publica basada en Identidad mejor conocida como ID-PKC, donde elimina el uso de certificados.

1.2.2.1. CRIPTOGRAFÍA DE CURVA ELÍPTICA

A partir de la propuesta de Victor Miller y Neal Koblitz, en 1985 ambas propuestas como trabajos independientes, acerca del uso de las curvas elípticas en la criptografía de clave pública, es como surge la Criptografía de Curva Elíptica (ECC por sus siglas en inglés) [5,6]. Las curvas elípticas han sido propuestas debido a sus propiedades de grupo y a que aún no se conoce un ataque eficiente contra su problema de logaritmo discreto.

Las implementaciones en ECC son más pequeñas y con mayor eficiencia que otras implementaciones [30]. ECC puede usar una llave pequeña ofreciendo la misma seguridad que otros algoritmos asimétricos que usan llaves más grandes, como se muestra en la Tabla 1.

Tabla 1: Tamaño de llaves de ECC comparado con RSA

Tamaño de llave – ECC (bits)	Tamaño de llave – RSA (bits)	Proporción de Llaves	Bits de Seguridad
163	1024	1 : 6	80
256	3072	1 : 12	128
384	7680	1 : 20	192
512	15360	1 : 30	256

En la fig. 7 puede verse un esquema jerárquico de la ECC [2], en la capa 6 se encuentran las aplicaciones de ECC, en la capa 5 se encuentran los protocolos de ECC, en la capa 4 las primitivas de ECC, como lo son la firma/verificación y el cifrado/descifrado, en la capa 3 se encuentran las operaciones principales de ECC, en la capa 2 las operaciones de Curva Elíptica, tales como la suma de puntos y el doblado de puntos, la forma en que estas operaciones son definidas ayuda en gran medida a que ECC ofrezca más seguridad incluso con longitud de llaves más pequeñas, en la

capa 1 se encuentra la aritmética de campo finito, una buena implementación de la aritmética de campo finito es crucial para que las capas superiores ofrezcan una mejor relación costo/eficiencia.

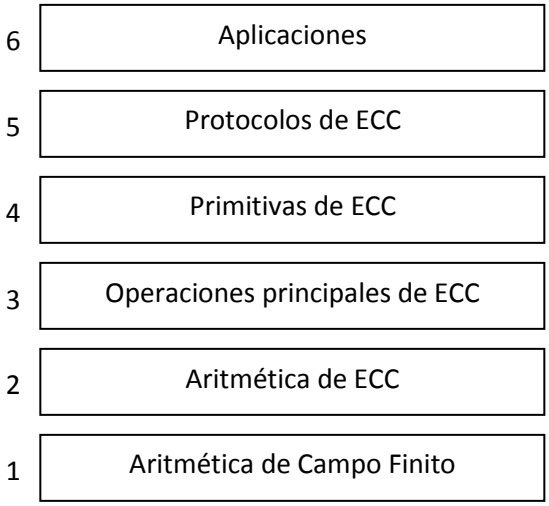

Figura 7: Esquema jerárquico de Criptografía de Curva Elíptica

CAPITULO 2.

FUNDAMENTOS MATEMÁTICOS

Como se ha descrito anteriormente, la seguridad de la criptografía asimétrica o de clave pública se base en la dificultad de resolver un problema matemático, así mismo, la base de una buena implementación de aplicaciones de criptografía de curva elíptica se da en términos de la aritmética del campo finito base, por este motivo es mandatorio contar con un buen fundamento matemático como el que se presenta en este capítulo.

2.1. CONCEPTOS ELEMENTALES DE TEORÍA DE NÚMEROS

La teoría de números es una herramienta muy usada para desarrollar y estudiar algoritmos criptográficos. Un buen fundamento matemático ayuda en gran medida a obtener los resultados deseados en los diseños o implementaciones que se pretenden lograr. La sección 2.1. tiene como fundamento [1, 3, 33].

2.1.1. NOCIONES BÁSICAS

Números enteros: Se define como números enteros al conjunto de números $Z = \{..., -2, -1, 0, 1, 2,...\}$. Dentro de este conjunto se tiene al subconjunto de de los números naturales $N = \{1, 2, 3, 4,...\}$, el cual se define como el conjunto de todos los positivos, o números mayores a cero.

Divisibilidad: Sean a y b dos enteros con $a \neq 0$. Se dice que a divide a b ($a|b$) si existe un entero $k \in Z$ tal que $b = a \cdot k$. Si $a|b$ se dice que a es un divisor o factor de b, b es un múltiplo de a, o que b es divisible entre a. Si no existe divisibilidad entonces se denota de la forma $a \nmid b$.

Máximo común divisor: Sean dos enteros $a \neq 0$ y $b \neq 0$, entonces el entero $d > 1$ es el máximo común divisor (*gcd* por sus siglas en inglés) de a y b si $d|a$, $d|b$ y para cualquier entero c tal que $c|a$, $c|b$ y $c|d$.

Números primos: Un entero $p > 1$ es un número primo si sólo es divisible entre 1 y él mismo.

Primos relativos: Dos enteros a y b son primos relativos si se cumple $gcd(a, b) = 1$.

Números compuestos: Si un entero q no es un número primo entonces es un número compuesto.

2.1.2. ARITMÉTICA MODULAR

El conjunto Z_m se define como el conjunto de todos los enteros positivos modulo m, el cual está compuesto de la forma $Z_m = \{0, 1, 2, 3,..., m-1\}$.

Congruencia: Si $m \in Z$, $m > 1$, entonces a, $b \in Z$ son congruentes módulo m si y sólo si $m|(a-b)$, y representa como $a \equiv b \bmod m$, donde m es el módulo de la congruencia. Si m divide a $(a - b)$ entonces a y b tienen el mismo residuo cuando se dividen entre m.

Adición modular: Si a, $b \in Z_m$, entonces el resultado de la adición modular $a + b \bmod m$ es un elemento que también pertenece al conjunto Z_m. Las propiedades de la adición modular son las siguientes:

- *conmutatividad*, a + b mod m = b + a mod m.
- *asociatividad*, (a + b) + c mod m = a + (b + c) mod m.
- *elemento neutro* (0) tal que a + 0 = a mod m.
- $\forall$ a, b $\in$ Z_m existe un *elemento único* x $\in$ Z_m tal que a + x = b mod m.

Multiplicación modular: Si a, $b \in Z_m$, entonces el resultado de la multiplicación modular $a \cdot b \bmod c$ es un elemento que también pertenece al conjunto Z_m. Las propiedades de la multiplicación modular son las siguientes:

- *conmutatividad*, a · b mod m = b · a mod m.
- *asociatividad*, (a · b) · c mod m = a · (b · c) mod m.
- *elemento neutro* (1) tal que a · 1 = a mod m.
- si gcd(m, c) = 1 y a · c ≡ b · c mod m, entonces a ≡ b mod m. Si m es un número primo, esta propiedad siempre se mantiene.

Inverso multiplicativo: Un número entero a tiene un inverso multiplicativo *m* si existe un entero *b* tal que $1 = a \cdot b \bmod m$. Entonces el entero *b* es el inverso de a y se denota de la forma a^{-1}. El inverso de un número *a mod m* existe si y solo si existen dos números *x* y *y* tales que $a \cdot x + m \cdot y = 1$ y esos números existen si $gcd(a, m) = 1$.

División modular: Si $a, b \in Z_p$ y *p* es un número primo, entonces la división de *a* entre *b* puede realizarse por medio de la operación $a \cdot b^{-1} \bmod m$, donde b^{-1} es el ***inverso multiplicativo*** de *b mod p*.

2.2. ESTRUCTURAS ALGEBRAICAS ELEMENTALES

La buena implementación de componentes aritméticos de campo finito es un pre-requisito para realizar implementaciones de criptografía de curva elíptica, debido a que las operaciones aritméticas de curva elíptica se realizan en el campo finito base.

2.2.1. GRUPOS

Un grupo es un conjunto G asociado a una operación binaria * en G tal que las siguientes propiedades se mantienen [33]:

1. * es asociativa: $\forall$ a, b, c $\in$ R, (a * b) * c = a * (b * c)
2. existe un elemento identidad e en G tal que: $\forall$ a $\in$ G, a * e = e * a = a
3. $\forall$ a $\in$ G existe un elemento inverso a^{-1} en G tal que: $a * a^{-1} = a^{-1} * a = e$

Si el grupo satisface

4. $\forall$ a, b $\in$ G se tiene que a * b = b * a

entonces se dice que el grupo es abeliano o conmutativo.

2.2.2. ANILLO

Un anillo <R, +, ·> es un conjunto R asociado a dos operaciones binarias: adición denotada por "+" y multiplicación de notada por "·", tal que:

1. R es un grupo abeliano con respecto a +.
2. · es asociativa: $\forall$ a, b, c $\in$ R, (a · b) · c = a · (b · c)
3. La ley distributiva se mantiene: $\forall$ a, b, c $\in$ R, a · (b + c) = a · b + a · c y (b · c) · a = b · a + c · a.

Además, los anillos pueden ser clasificados de acuerdo a las siguientes definiciones:

i. Un anillo es llamado anillo con identidad si el anillo tiene identidad en la multiplicación, esto es, si existe un elemento e tal que $\forall$ a $\in$ R, a · e = e · a = a

ii. Un anillo es llamado conmutativo si · es conmutativa .

iii. Un anillo es llamado dominio de integridad si es un anillo conmutativo con identidad e ≠ 0 en lo cual a · b = 0 implica que a = 0 o b = 0.

iv. Un anillo es llamado anillo de división (skew ring) si los elementos no nulos de R forman un grupo bajo ·.

v. Un anillo conmutativo de división es llamado campo.

2.2.3. CAMPO

Un campo <F, +, ·> es un conjunto F asociado a dos operaciones binarias: adición denotada por "+" y multiplicación denotada por "·". Tal que:

1. F es un grupo abeliano con respecto a +.

2. F/{0} es un grupo abeliano con respecto a ·.
3. La ley distributiva de los anillos se mantiene.

La resta de elementos del campo se da en términos de la adición, y la división se da en términos de la multiplicación:

- $\forall\, a, b \in F$, $(a - b) = a + (-b)$ donde $-b$ es llamado el *negativo* de b
- $\forall\, a, b \in F$, $a / b = a \cdot b^{-1}$ donde b^{-1} es el único elemento en F tal que $b \cdot b^{-1} = 1$, b^{-1} es llamado el inverso de b

Si el número de elementos de un campo es finito, entonces se dice que el campo es un campo finito.

2.2.4. CAMPO FINITO

Debido a que los *campos finitos* gozan de las propiedades esenciales mencionadas anteriormente, tienen numerosas aplicaciones tales como *Implementaciones Criptográficas* [34, 35], la *Teoría de Códigos* [36] y el *Procesamiento Digital de Señales* [37]. En los últimos años, los *campos finitos* han tomado un papel muy importante en la Criptografía de *Curvas Elípticas* e *Híper-elípticas*, debido a que la aritmética de criptografía de curva elíptica se realiza en el campo finito base [1].

Un campo finito F, también llamado Campo de Galöis y denotado GF [7], es una estructura algebraica que consiste en un anillo conmutativo, en el cual todos los elementos, excepto el elemento cero, tienen elemento inverso. Un campo finito puede ser visto como una abstracción de sistemas de números, por ejemplo racionales, reales y complejos, con sus propiedades esenciales., tales como elemento identidad, elemento inverso, ley distributiva, ley asociativa y ley de la cerradura [1].

Un campo finito F(q) donde $q = p^m$, es un conjunto con característica p y con q elementos; el campo finito $F(p^m)$, de orden p^m existe si y solo si p^m es potencia de un número primo [33], donde *p* es *número primo* y *m* es un *número entero positivo*. En criptografía los dos casos más estudiados son q = p, donde p es un primo y $q = 2^m$. El primer caso F(p) se conoce como campo primo; el segundo caso $F(2^m)$ se conoce como campo con característica dos o campo finito binario, y también puede denotarse de la forma $GF(2^m)$.

2.2.5. EXTENSIONES DE CAMPOS

Una extensión de campo puede ser generalizada de la siguiente forma: sea p un número primo y m un entero mayor a 1, entonces el campo *F(p)[x]* denota el conjunto de polinomios en la variable x con coeficientes en *F(p)*. De esta manera, los elementos del campo $F(p^m)$ son los polinomios en *F(p)[x]* de grado máximo m-1 [33].

2.2.6. TORRES DE CAMPOS

Una torre de campo es una secuencia finita de extensiones de campo [32]. Esto es, la secuencia:

$$F_0 \subseteq F_1 \subseteq F_2 \subseteq \ldots \subseteq F_n$$

es una torre de campo, la cual obtiene su nombre al poderse expresar de la forma

F_n
|
...
|
F_2
|
F_1
|
F_0

En la fig. 8 se presenta un ejemplo de torre de campo, que es equivalente a un campo con grado mayor. A partir de la figura es posible notar que una torre de campo es un campo obtenido mediante la extensión repetida de un campo base mediante un polinomio irreducible del mismo grado.

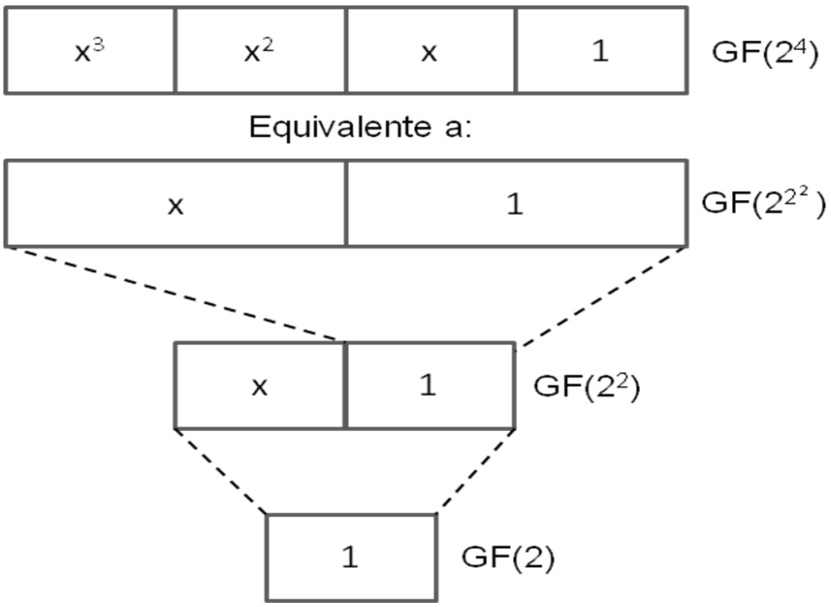

Figura 8: Ejemplo de una Torre de Campo

2.2.7. CAMPOS FINITOS BINARIOS

Un campo finito binario o de característica dos tiene un orden 2^m. Una forma de representar un campo finito binario es con una representación polinomial [39]. La base polinomial es la representación estándar para denotar los elementos de un campo finito binario, en esta notación se considera a cada elemento del campo como un polinomio de grado m-1, el cual se construye sobre una indeterminada, por ejemplo x, cuyos coeficientes pertenecen al campo base, como puede verse de la siguiente manera en la ec. 1:

$$A(x) = \{a_{m-1}x^{m-1} + \ldots + a_2x^2 + a_1x + a_0\} \qquad \text{Ec. } 1$$

De manera general, el campo GF(2^m) está construido como una extensión del campo GF(2). El conjunto {1, x, x^2,…,x^{m-1}} forma una base polinomial sobre el campo GF(2), donde cualquier elemento arbitrario A(x) $\in$ GF(2^m) puede ser expresado como en la ec. 2:

$$A(x) = \sum_{i=0}^{m-1} a_i x^i \qquad \text{Ec. 2}$$

Los elementos del campo también pueden ser denotados como un vector de bits de longitud m, el elemento A(x) $\in$ GF(2^m) se denota como sigue, en la ec. 3:

$$A = \{a_{m-1}, ...a_2, a_1, a_0\} \qquad \text{Ec. 3}$$

Un polinomio irreducible P(x) [1] siempre existe para cualquier grado m, este polinomio irreducible no puede descomponerse como producto de polinomios binarios de grado menor a m. Este polinomio irreducible P(x) se representa en la ec. 4:

$$P(x) = \{p_m x^m + p_{m-1} x^{m-1} + ... + p_2 x^2 + p_1 x + p_0\} \qquad \text{Ec. 4}$$

Existen algunas clases especiales de polinomios irreducibles, los cuales resultan más convenientes para implementaciones eficientes de aritmética de campo finito binario, tales como los trinomios y pentanomios [2]. Los trinomios son polinomios con tres coeficientes diferentes a cero, y tienen la forma que se presenta en la ec. 5:

$$P(x) = x^k + x^n + 1 \qquad \text{Ec. 5}$$

Los pentanomios tienen cinco coeficientes diferentes a cero, y tienen la forma presentada en la ec. 6:

$$P(x) = x^k + x^{n_2} + x^{n_1} + x^{n_0} + 1 \qquad \text{Ec. 6}$$

2.3. BASES DE CAMPOS FINITOS

En esta sección se describen diferentes bases del campo GF(2^m) construido sobre GF(2), cada una de estas bases presenta ventajas y desventajas cuando se realizan implementaciones de su aritmética.

2.3.1. BASES POLINOMIALES

Las *Bases Polinomiales* son las representaciones más utilizadas para denotar los elementos de un *campo finito* construido como una *extensión* de un campo más pequeño [32, 33, 38], denominado *campo base*, por ejemplo, el campo GF(2^m) se construye como una *extensión* de *orden m* sobre el campo GF(2). En esta notación se considera a cada *elemento* del campo como un *polinomio* [3], el cual se construye sobre una *indeterminada*, por ejemplo *x*, y cuyos coeficientes pertenecen al campo base. En esta representación, el *elemento identidad* de la *multiplicación*, se representa como el *polinomio uno* (1), el cual se forma con todos los coeficientes en 0 excepto el coeficiente de menor orden, el *elemento identidad* de la *suma* es representado por el *polinomio cero* (0), el cual se forma con todos los coeficientes en 0, estos *polinomios* se pueden representar como *vectores de bits* [40], donde el orden de los bits se arregla de la forma $a_{m-1}\ldots a_1 a_0$.

Las *operaciones* entre elementos del campo se realizan siguiendo las reglas del *algebra convencional*, donde las operaciones entre coeficientes se suscriben en la *aritmética convencional del campo base*. La *suma* de *elementos* del campo GF(2^m) se realiza como una *suma* de *polinomios* con coeficientes aritméticos módulo 2. Por ejemplo, la suma de elementos del campo puede realizarse como una *XOR bit a bit* realizada entre los coeficientes correspondientes [40]. En el caso de la *multiplicación* de elementos del campo existe la figura de un *polinomio irreducible* de grado *m* (siempre existe polinomio irreducible para cualquier grado *m*) [11], y esta

multiplicación de elementos del campo se comporta como la operación multiplicación algebraica módulo dicho *polinomio irreducible* [1]. Cabe hacer notar que un polinomio es *irreducible* si no se puede factorizar como un producto de *polinomios binarios* de grado menor que *m*.

2.3.2. BASES NORMALES

Otra forma para representar un elemento de un campo de la forma GF(2^m) construido sobre GF(2) es usar *Bases Normales*. Una *Base Normal* del campo GF(2^m) es una base de la forma $N = \{\beta, \beta^2, \beta^{2^2}, \ldots, \beta^{2^{m-1}}\}$ [41], donde β genera la *Base Normal N* y β es llamado e*lemento normal* del campo GF(2^m) [42].

Cuando se representa un elemento del campo de la forma GF(2^m) mediante una *Base Normal*, los elementos se pueden representar como *vectores de bits* [43], el orden de los bits se ordena de la forma $a_0a_1\ldots a_{m-1}$ [39], se puede notar que el orden de los bits es *diferente* al de una *Base Polinomial*. En una *Base Normal*, el *elemento identidad* en la *multiplicación* se representa como un vector de bits 1's de longitud m, y el *elemento cero* se representa como un vector de bits 0's de longitud m [40].

Una ventaja de usar *Bases Normales*, es que las *potencias* múltiplos de 2 de un elemento del campo se logran con una simple *rotación de bits* en su vector de representación [44]. Las *Bases Normales* no son usadas en implementaciones de hardware, e incluso, las multiplicaciones de software en *Bases Normales* suelen ser demasiado lentas, en comparación a multiplicaciones de *Base Polinomial* [1], debido a estas desventajas, han sido definidas las *Bases Normales Óptimas* y *Bases Normales de Baja Complejidad*, descritas más adelante, las cuales *disminuyen* la *complejidad* en implementaciones de aritmética de campo finito en hardware y/o software.

2.3.2.1. BASES NORMALES ÓPTIMAS

Para el campo de la forma GF(2^m) sobre GF(2) existen *Bases Normales Óptimas Tipo I* y *Tipo II* [45], una *Base Normal* es llamada *Base Normal Óptima* si cumple con los siguientes atributos:

I. *Base Normal Óptima Tipo I*, si *m+1* es un *primo* y 2 es un *elemento generador* en el campo F_{m+1}, entonces el *orden multiplicativo* de 2 módulo *m+1* es *m* [44, 46].
II. *Base Normal Óptima Tipo II*, si *2m+1* es un *primo* y también 2 es un *elemento generador* en el campo F_{2m+1} ó *2m+1* es *idéntico* a 3 módulo 4 y el *orden multiplicativo* de 2 módulo *2m+1* es *m* [42, 47].

Cabe señalar que en ambos casos existe al menos un *elemento generador* en un campo de la forma GF(2^m), y todos sus elementos diferentes a cero pueden expresarse como potencias de ese *elemento generador*, el cual también es llamado *elemento primitivo*.

Las *Bases Normales Óptimas* son usadas en la práctica con el fin de reducir la complejidad en el hardware para efectos de multiplicación entre *elementos normales* del campo en GF(2^m).

Aunque las *Bases Normales Óptimas* no muestran ventajas significativas sobre *Bases Polinomiales* para implementaciones en software, se puede decir que son atractivas para implementaciones en hardware, porque conservan las ventajas de una *representación polinomal* [48].

2.3.2.2. BASES NORMALES GAUSSIANAS

Al hablar de una *Base Normal Gaussiana Tipo t* para un campo de la forma GF(2^m), y para valores específicos de *m* y *t* [49], cuando *m* no es divisible entre 8 [50] y si *t* es un entero positivo, entonces se dice que GF(2^m) tiene al menos una *Base Normal Gaussiana Tipo t* [40]. El *tipo* (*t*) de una

Base Normal Gaussiana es un entero positivo que mide la complejidad de la multiplicación con respecto a esa base [51], de manera general, entre más pequeño sea el *tipo*, más eficiente es la multiplicación. Con los casos específicos de *t=1* y *t=2*, las *Bases Normales Gaussianas* se convierten en *Bases Normales Óptimas Tipo I* y *II* [52]. La selección de la *Base Normal Gaussiana*, que debe utilizarse para representar el campo GF(2^m), se realiza de acuerdo a tres condiciones [39]:

I. Si existe una *Base Normal Gaussiana Tipo 2* entonces debe usarse.

II. Si no existe una *Base Normal Gaussiana Tipo 2* pero si existe una *Tipo 1* entonces la *Tipo 1* debe usarse.

III. Si no existe *Base Normal Gaussiana Tipo* 1 y tampoco existe *Tipo 2* entonces la *Base Normal Gaussiana* con el menor tipo debe usarse.

2.3.3. DOMINIO DE LA FRECUENCIA

Un *elemento* del campo de la forma GF(2^m) puede representarse en el *Domino de la Frecuencia*, y para convertir un elemento de GF(2^m) dentro de una representación de *Dominio de la Frecuencia* se usa la *Transformada Discreta de Fourier* sobre un *campo finito* [8]. La *Transformada Discreta de Fourier* sobre un *campo finito* también es conocida como la *Transformada Teórica de Números* sobre un *anillo*, y fue dada a conocer por *Pollard* en 1971 [54]. Cuando se necesita desarrollar una multiplicación modular en el *Dominio de la Frecuencia*, primero se deben transformar los *operandos*, *elementos del campo*, en el *Dominio de la Frecuencia*, a pesar de que existen diversos *algoritmos* para *multiplicación* de *elementos del campo,* todos ellos tienen un grado de complejidad de acuerdo a las *sumas* y *multiplicaciones* de sus *coeficientes* [55].

La *Transformada Discreta de Fourier* fue originalmente un método propuesto para la *multiplicación* de *enteros*, y se ha demostrado que es un

método muy *eficiente*, reduciendo la complejidad en la *multiplicación de polinomios* de grado *m-1*.

Aunque la multiplicación de elementos de un *campo finito* se logra de manera más eficiente en el *dominio de la frecuencia* que en el *dominio del tiempo*, la *transformación* de un elemento del d*ominio del tiempo* al *dominio de la frecuencia* es *costosa*, debido a esta transformación la *Transformada Discreta de Fourier era* un método *no práctico* para *operar* elementos pequeños, por ejemplo de longitud *menor* a *1000 bits*, como se usan en la mayoría de las aplicaciones [56], una de las *primeras* implementaciones operaba elementos *muy largos*, superiores a *4060 bits* [57].

Implementaciones *recientes* han demostrado que es posible operar, de una manera *eficiente*, elementos de longitud relevante para *ECC*, por ejemplo *160 bits*, usando la *Transformada Teórica de Números* basada en aritmética de *campos finitos* [8], estas implementaciones resultan útiles especialmente en plataformas computacionales con *espacio restringido*.

2.4. COMPARATIVA Y SELECCIÓN DE LA BASE

La importancia de seleccionar y *construir* una base para representar elementos del *campo finito* viene dada por las *ventajas* y/o *desventajas* que esta base ofrece, al implementar, en *hardware* o *software*, componentes aritméticos de campos finitos con aplicaciones como la *ECC*.

Independientemente de la base, en la suma de elementos no hay acarreos, puede ser realizada fácilmente mediante una *XOR bit a bit*, y por consiguiente la suma puede ser implementada en *hardware* o *software* con más facilidad en *campos finitos binarios* que en *campos finitos primos*.

Reducir la complejidad de la *multiplicación* entre elementos del campo es el principal *objetivo* de las diferentes *técnicas de representación*, para efectos de implementaciones de componentes de campos finitos en

hardware o *software*. Algunas características de las técnicas de representación son mencionadas a continuación:

Base Polinomial.

- La *base polinomial* es usada como representación estándar.
- La forma trivial de realizar la multiplicación incluye una *reducción polinomial*.
- La complejidad de la multiplicación cambia de acuerdo a la selección de *polinomio irreducible*, por ejemplo, si el polinomio irreducible es un *trinomio*, la *reducción polinomial* se realiza con mayor eficiencia.

Base Normal

- La *multiplicación* entre elementos del campo es *complicada*, por lo que una implementación en *hardware* ó *software* no es eficiente.
- El elevar al cuadrado un elemento del campo se implementa *eficientemente* en *hardware*, porque la operación sólo se realiza mediante un corrimiento cíclico de su vector de representación.
- Se puede calcular eficientemente una *raíz cuadrada*, lo que resulta útil para la recuperación de puntos, en *Curva Elíptica*.

Base Normal Gaussiana

- Es una *base normal* cumpliendo la característica de que m no es divisible entre 8.
- Entre más bajo sea el tipo de base, *menos complejidad* tiene la multiplicación, esto significa menor complejidad en la implementación de componentes aritméticos de campo finito en *hardware* o *software*.

Base Normal Óptima

- Es una *base normal* que cumple ciertas condiciones, *reducen* la *complejidad* de la *multiplicación* entre elementos en una base normal.

- Es posible lograr *implementaciones eficientes* de aritmética de campo, en *velocidad* y *complejidad* de *hardware*.

Dominio de la Frecuencia

- La *multiplicación* tiene *menor complejidad* en el *dominio de la frecuencia* que en el *dominio del tiempo*.
- *Transformar* un *elemento* del campo del *dominio del tiempo* al *dominio de la frecuencia* es *costoso*.

Debido a que es la representación estándar de los campos finitos, y a que existen diferentes algoritmos de multiplicación, además de su facilidad de implementación tanto en hardware como software, se ha seleccionado la **base polinomial** como la base para representar los elementos del campo finito de los multiplicadores propuestos

2.5. ARITMÉTICA DE CAMPOS FINITOS BINARIOS

Se ha definido que cada elemento perteneciente al campo puede ser representado como un polinomio de grado máximo m-1, y que este polinomio a su vez puede representarse como un vector de bits de longitud m.

2.5.1. ADICIÓN

La adición de dos elementos A(x), B(x) $\in$ GF se realiza simplemente como una adición de polinomios donde cada uno de los coeficientes pertece al campo GF(2), de forma equivalente si dichos elementos se representan como vector de bits, entonces la adición corresponde a una operación XOR bit a bit del elemento A(x) y del elemento B(x). A manera de ejemplo: si los vectores A(x) y B(x) $\in$ GF se denotan como vectores de bits, la adición C(x) = A(x) + B(x) se representa de la forma $(c_{m-1},\ldots,c_1,c_0) = (a_{m-1},\ldots,a_1,a_0) + (b_{m-1},\ldots,b_1,b_0)$, donde $c_i=a_i \oplus b_i$.

2.5.2. MULTIPLICACIÓN

La multiplicación de dos elementos A(x) y B(x) $\in$ GF es una multiplicación de polinomios C(x)=A(x) · B(x) módulo P(x). Si se tiene (r_{m-1},…,r_2,r_1,r_0) = (a_{m-1},…,a_2,a_1,a_0) · (b_{m-1},…,b_2,b_1,b_0), entonces (r_{m-1},…,r_2,r_1,r_0) es el polinomio residuo cuando se realiza la multiplicación R(x) = A(x) · B(x) y el resultado se divide entre (p_m,p_{m-1},… ,p_2,p_1,p_0), a esta operación R(x) módulo P(x) se le llama reducción polinomial. La multiplicación entre elementos del campo puede ser realizada como se presenta en la ec. 7:

$$c_i = \sum_{j=0}^{m-1} a_j b_{i-j(\mathrm{mod}\, m)} \qquad \text{Ec. 7}$$

donde d=2^m-1 para i $\in$ {0,…,d-1}.

Esta operación puede realizarse en dos pasos, en el primer paso se calcula el polinomio R(x) = A(x) · B(x), que tiene como grado máximo 2m-2, como se muestra en la ec. 8, esta operación puede ser realizada con una técnica de suma y desplazamiento, por medio de una matriz de multiplicación, mediante el algoritmo de Karatsuba-Ofman, etc.

$$R(x) = (\sum_{i=0}^{m-1} a_i x^i) \cdot (\sum_{i=0}^{m-1} b_i x^i) \qquad \text{Ec. 8}$$

En el segundo paso, se obtiene el polinomio C(x) de grado máximo m-1, como se muestra a continuación en la ec. 9, al igual que el paso anterior, esta operación de reducción puede realizarse con una técnica de suma y desplazamiento o por medio de una Reducción de Barret, etc.

$$P(x) = R(x) \bmod P(x) \qquad \text{Ec. 9}$$

2.6. ARITMÉTICA DE CURVA ELÍPTICA

Como se vio en la figura 7, la aritmética de curva elíptica está soportada por la aritmética de campo finito. Las operaciones de aritmética de curva elíptica son la suma y el doblado de puntos, las cuales se definirán más adelante en esta sección, estas operaciones aritméticas se apoyan de diferentes operaciones de campo finito, como lo son la suma de elementos de campo finito, la multiplicación y la inversión, por nombrar las más importantes. De ahí que resulte importante proponer un multiplicador eficiente de campo finito binario con aplicaciones en sistemas de criptografía de curva elíptica.

2.6.1. CURVAS ELÍPTICAS

Una curva elíptica E es un lugar geométrico que se define mediante una ecuación de tercer grado [7], ésta a su vez puede definirse sobre cualquier cuerpo K, como se presenta en la ec. 10:

$$E/K : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \qquad \text{Ec. } 10$$

donde cada coeficiente a_1, a_2, a_3, a_4, $a_6 \in K$, y a K se le conoce como campo base, así también si la curva E está definida sobre el cuerpo K entonces la curva puede ser definida sobre cualquier extensión de K, a esta ecuación se le conoce como la ecuación general de Weierstrass [53], donde generalmente K es un campo, por ejemplo números reales R, complejos C, racionales Q o campos finitos GF.

A manera de ejemplo se presenta la gráfica de una curva elíptica, fig. 9, la cual está definida sobre el conjunto R.

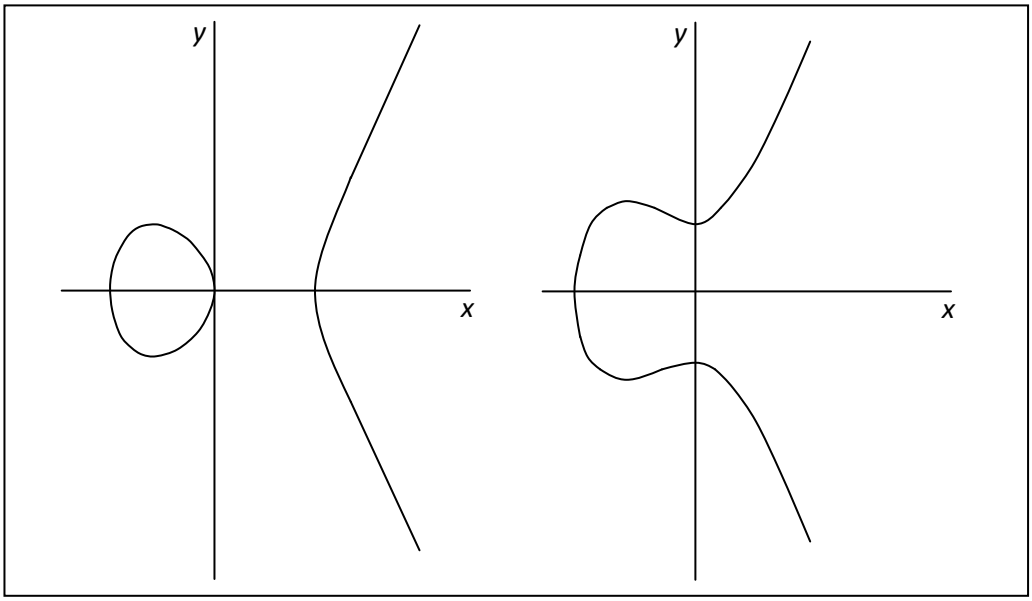

Figura 9: Curvas Elípticas usadas en Criptografía de Curva Elíptica

2.6.2. ARITMÉTICA DE PUNTOS

Sea E una curva elíptica definida sobre el campo K, entonces existe una regla para sumar dos de E(K) y así obtener un tercer punto que también debe pertenecer a E(K).

Existen dos operaciones con los puntos de una curva, la adición de puntos y el doblado de puntos [7]. Las aritmética de curva elíptica está soportada por la aritmética de campo finito, es decir, para la suma y para el doblado de puntos necesitamos operar elementos que pertenecen al campo finito, cada ordenada es un elemento del campo base. Estas operaciones se definen a continuación.

2.6.2.1. ADICIÓN DE PUNTOS

Sean $P=(x_1, y_1)$ y $Q=(x_2, y_2)$ dos puntos diferentes en una curva E, entonces la suma de puntos P + Q = R se define como el lugar geométrico correspondiente a trazar una línea recta entre los puntos P y Q, donde esta

línea debe intersectar la curva elíptica en un tercer punto, cuyo reflejo con respecto al eje de las x corresponderá al punto R, siendo este tercer punto el resultante de la adición. El lugar geométrico correspondiente a la adición de dos puntos se describe en la fig. 10, y las ecuaciones involucradas en su obtención son $x_3 = \lambda^2 + \lambda + x_1 + x_2 + a$, $y_3 = \lambda(x_1 + x_3) + x_3 + y_1$ y por último la ecuación $\lambda = (y_1 + y_2)/(x_1 + x_2)$ donde λ equivale al cálculo de la pendiente, x_3 equivale al punto de intersección entre la curva y la recta PQ y finalmente y_3 describe el valor de "y" para la curva E(x), estas ecuaciones también son descritas en la fig. 10.

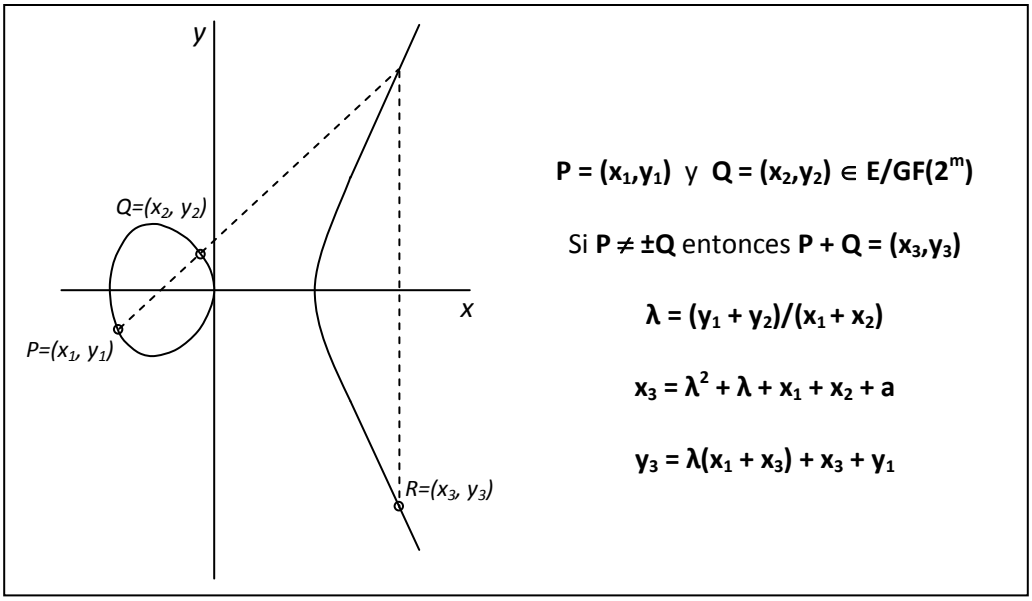

Figura 10: Adición de Puntos para una curva definida sobre GF(2^m)

2.6.2.2. DOBLADO DE PUNTOS

Sea P=(X_1, y_1) un punto en una curva, entonces la suma P + P = R, que se conoce como doblado de puntos, se define como sigue el lugar geométrico correspondiente a trazar una línea recta tangente del punto P hacia la curva elíptica, esta línea debe intersectar la curva elíptica en un segundo punto, cuyo reflejo con respecto al eje de las x corresponderá al punto R, siendo este segundo punto el resultado del doblado de puntos. El

lugar geométrico correspondiente al doblado de puntos es descrito en la fig. 11, y las ecuaciones involucradas en su obtención son $x_3 = \lambda^2 + \lambda + a$, $y_3 = x^2_1 + \lambda x_3 + x_3$ y por último la ecuación $\lambda = x_1 + (y_1 / x_1)$ donde λ equivale al cálculo de la pendiente, x_3 equivale a el punto de intersección entre la curva y la línea tangente al punto P y finalmente y_3 describe el valor de "y" para la curva E(x), estas ecuaciones también son descritas en la fig. 11.

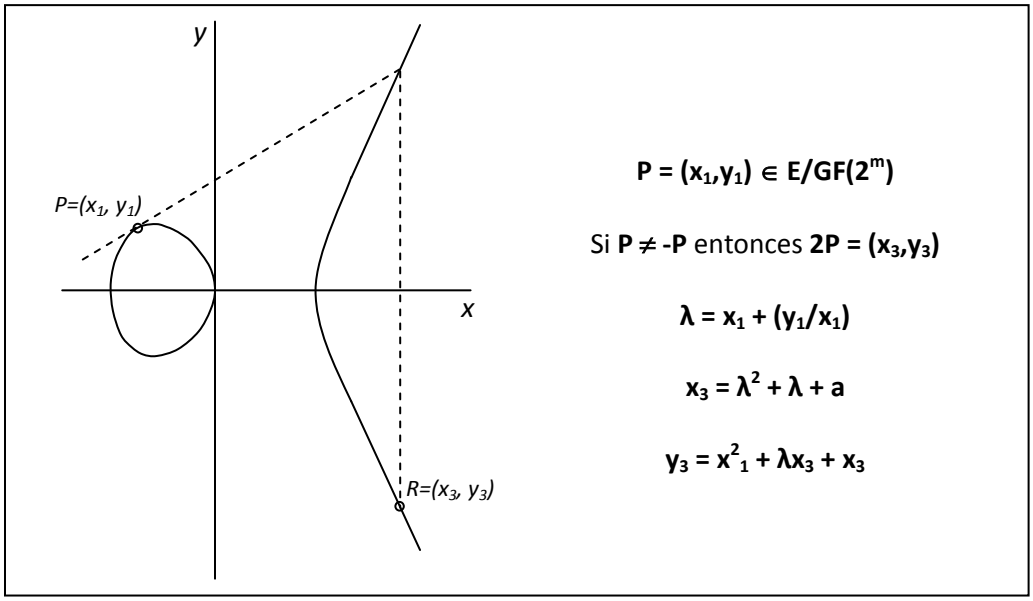

Figura 11: Doblado de Puntos para una curva definida sobre GF(2^m)

CAPITULO 3.

HARDWARE RECONFIGURABLE

De manera general, existen dos alternativas para la implementación de componentes aritméticos de campos finitos: los FPGA's y los VLSI.

Un VLSI pueden procesar grandes cantidades de datos, pero el ciclo de diseño toma demasiado tiempo, además de que un cambio o modificación en el diseño es muy complicado, incluso puede no ser posible. Debido a estos inconvenientes la opción de diseño se basa en el hardware reconfigurable.

El hardware reconfigurable se refiere a circuitos prefabricados cuya funcionalidad es reprogramable, es decir, estos circuitos pueden ser personalizados desde el exterior mediante diversas técnicas de programación. Por sus características y ventajas, que son descritas más adelante, el desarrollo de esta tesis fue realizado usando hardware reconfigurable, como lo es un FPGA.

3.1. ALTERNATIVAS PARA EL DISEÑO.

Para describir las diferencias más notables entre un VLSI y un FPGA a continuación se presenta una tabla comparativa [2], tabla 2, entre VLSI y FPGA.

Tabla 2: Comparativa entre VLSI y FPGA

	VLSI	FPGA
Tamaño	Grande	Pequeño
Costo	Alto	Bajo
Velocidad	Muy alto	Alto
Memoria	Alto	Alto
Flexibilidad	No	Sí - Muy alto
Consumo de energía	Bajo	Medio
Pruebas/Verificaciones	Difícil	Fácil
Configuración en tiempo de ejecución	No	Sí

3.2. FPGA

Un Arreglo de Compuertas Programables en Campo ó FPGA (Field Programmable Gate Array) es un dispositivo semiconductor que contiene bloques de lógica cuya interconexión y funcionalidad se puede programar mediante un Lenguaje de Descripción de Hardware, en este caso VHDL [9]. Su mayor ventaja es la reconfiguración, pueden ser reprogramados incluso en tiempo de ejecución, además de proveer un ciclo de diseño más corto porque ofrecen pruebas de funcionalidad rápidas y exactas. Dada la conectividad interna en un FPGA, es posible desarrollar, de una forma mucho más fácil y económica, circuitos integrados de aplicación específica [58]. De esta forma, es posible decir que virtualmente cualquier circuito digital puede ser implementado en un FPGA.

Los FPGAs pueden ser usados para implementar algoritmos en hardware, y tomando en cuenta sus capacidades, se identifica como uno de sus usos potenciales las aplicaciones con estructuras iterativas y altamente regulares con valores de entrada de igual o diferente longitud, como lo es en el caso de la criptografía [2].

Así mismo, es importante hacer énfasis en que un FPGA supera las dificultades de un diseño en VLSI, también permitiendo procesar una gran cantidad de datos.

3.2.1. XILINX SPARTAN-3 XC3S200

La arquitectura de la familia Spartan-3 consiste en 5 elementos funcionales programables fundamentales [59]:

- *CLBs*: Bloques Lógicos Configurables (Configurable Logic Blocks) que contienen *LUTs* (Tablas de Búsqueda o Look-Up Tables) basados en *RAM* para implementar la lógica y el almacenamiento de elementos que podrían usarse como flip-flops. Los *CLBs* pueden programarse para desarrollar diversas operaciones lógicas, así como también almacenar datos. Cabe mencionar que cada *CLB* equivale a 4 slices, y contiene 8 *LUTs*.

- *IOBs*: Los Bloques de Entrada/Salida (Input/Output Blocks) controlan el flujo de datos entre los pins de entrada/salida y la lógica interna del dispositivo. Cada *IOB* es capaz de soportar el flujo de datos bidireccional.

- Bloques de *RAM*: Proveen el almacenamiento de datos en forma de bloques de puerto dual de 18-Kbits.

- Multiplicadores: Los multiplicadores de bloque aceptan como entrada dos operandos de 18 bits, y calcula el producto.

- *DCMs*: Bloques de Gestión de Reloj Digital (Digital Clock Manager) que proporciona auto-calibración, soluciones totalmente digitales para la distribución, retardo, multiplicación, división.

Los anteriores elementos se encuentran distribuidos como se muestra en la fig. 12.

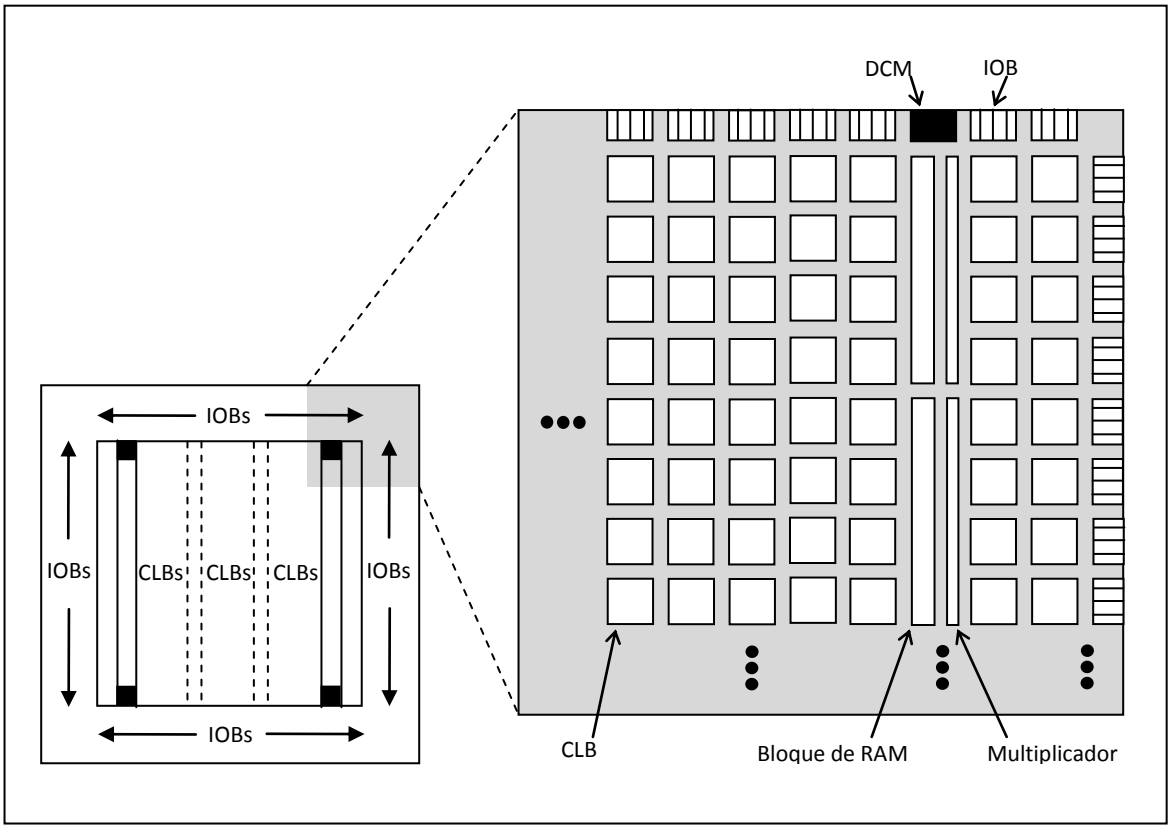

Figura 12: Arquitectura Interna de la Familia Spartan-3

Así mismo, los atributos del Spartan-3 XC3S200 se muestran en la siguiente tabla:

Tabla 3: Atributos del Spartan-3 XC3S200

Array CLB			IOBs	Bloques de RAM	Multiplicadores dedicados	DCMs
Renglones	Columnas	CLBs Totales				
24	20	480	173	216k	12	4

Otra característica importante de los FPGA de la familia Spartan-3 es que cuentan con ocho entradas globales de reloj, denominadas GCLK0 - GCLK7, estas entradas proporcionan acceso a baja capacitancia.

3.3. VHDL

El lenguaje VHDL (Very high speed integrated circuits Hardware Description Languaje) es una herramienta de diseño y programación que proporciona diferentes ventajas en la planeación y en el diseño de sistemas electrónicos digitales [9]. Es decir, mediante el leguaje de programación VHDL se puede describir la estructura y el comportamiento de sistemas electrónicos digitales, además el diseñador sabe cómo son mapeadas las instrucciones dentro de los componentes del FPGA [2].

La aparición de VHDL fue auspiciada por la IEEE, y adoptado en el estándar 1076 de la IEEE, y al igual que todos los demás estándares, es revisado periódicamente [58].

El diseño de VHDL ayuda en el proceso de diseño debido a las siguientes razones:

- permite la descripción de la estructura de un sistema, de una manera más general, permite saber cómo un sistema puede ser descompuesto en subsistemas y además, como esos subsistemas están interconectados entre sí.
- permite especificar cómo funciona un sistema mediante formas conocidas de lenguaje de programación.
- permite que el diseño sea simulado antes del proceso de manufactura, de esta forma los diseñadores pueden comparar, de manera rápida, los resultados de la simulación con los resultados de otras alternativas, de

manera general los diseñadores realizan las pruebas sin la necesidad de los costos y tiempos que generan los prototipos en hardware.

Algunas de las características de VHDL es el que se pueden describir actividades que ocurren en forma simultánea o concurrente, se permiten describir módulos con acciones que serán evaluadas en forma secuencial o procedural donde cada uno de esos módulos es visto de una forma concurrente, así mismo se posibilita realizar una construcción jerárquica donde se pueden combinar descripciones estructurales y de flujo de datos con descripciones de comportamiento.

Si durante el proceso de diseño de sistemas electrónicos digitales se usa VHDL, entonces es posible obtener las siguientes ventajas:

- Como herramienta de especificación ofrece:
  - ✓ puede ser usado para especificar el sistema de una manera general en nivel de hardware.
  - ✓ el hardware puede ser descrito en un nivel de sistema, de subsistema o de componentes.
- Como herramienta de diseño ofrece:
  - ✓ se puede tener una mejor documentación, así también existe una facilidad en la reutilización de componentes.
  - ✓ se facilita la portabilidad, es decir existe una independencia tecnológica.
- Como herramienta de simulación ofrece:
  - ✓ permite el uso de modelos normalizados de diversos componentes que han sido desarrollados por diferentes fabricantes.

- ✓ existe una facilidad para la generación de vectores de pruebas.

El uso de un Lenguaje de Descripción de Hardware, como lo es VHDL, para el diseño de hardware es variado [9], como se muestra en la fig. 13, puede ser usado en un ASIC, FPGA, PLD o en partes estándar, esta tesis se enfoca al diseño de componentes aritméticos en un FPGA.

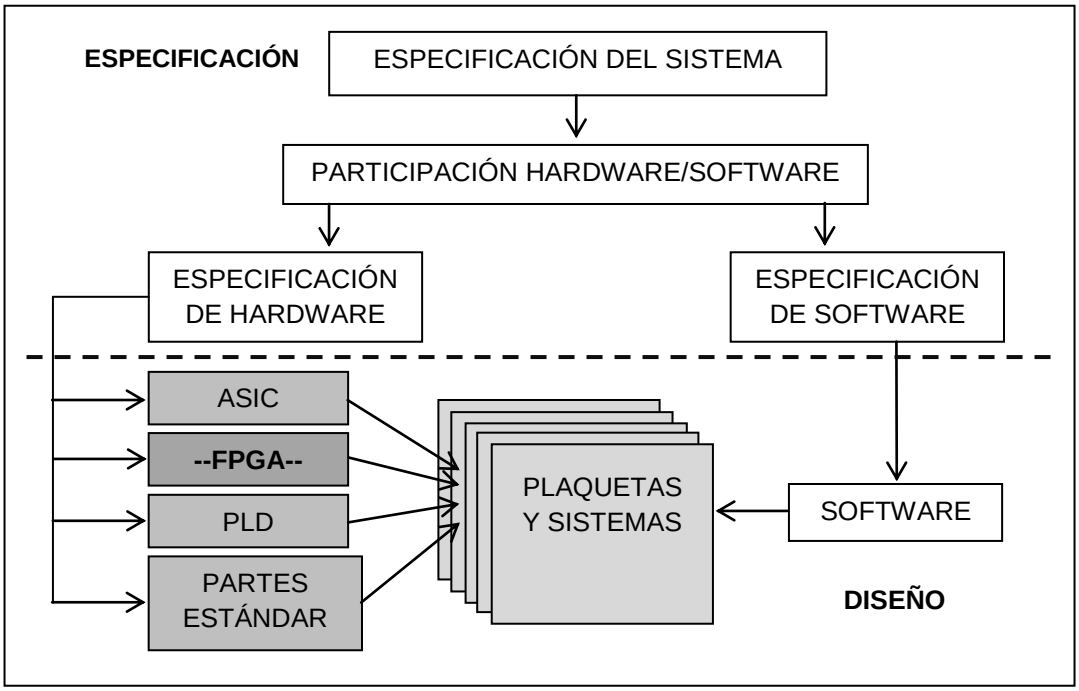

Figura 13: Posibles usos de un Lenguaje de Descripción de Hardware

CAPITULO 4.

DISEÑO E IMPLEMENTACIÓN DE ARQUITECTURAS BASE DE COMPONENTE ADITIVO Y MULTIPLICATIVO

El diseño de la arquitectura del multiplicador propuesto se ha basado en las siguientes directrices de diseño:

- Eficiencia - De una manera general, la eficiencia mide los recursos usados, y tiene como métrica el espacio, el número de procesos o el tiempo. Para medir la eficiencia de los diseños propuestos se usa como métrica el espacio. Mediante la eficiencia se busca obtener un buen balance de costo/beneficio.
- Rendimiento - El rendimiento se refiere al número de ciclos de reloj desde que hay una entrada, esta entrada se procesa y así exista una salida.

Estas directrices de diseño en hardware fueron consideradas en la propuesta de diseño debido a su gran importancia, además ayudan en gran medida a lograr el objetivo principal que se ha planteado, que es diseñar componentes de alta velocidad ocupando poca área en el dispositivo.

Cabe destacar que además de estas directrices de diseño, también se deben tener en cuenta los requerimientos de seguridad. El documento X9.62, estándar del Algoritmo de Firma Digital para Curva Elíptica (ECDSA por sus siglas en inglés) para la industria de servicios financieros, realizado por el Instituto Nacional Estadounidense de Estándares (ANSI por sus siglas en inglés) dicta que una implementación es segura si hace uso de una clave de longitud mayor a 160 bits, y si también se hace uso de una base polinomial, base normal óptima o una base normal gaussiana, para representar los elementos del campo finito binario sobre el cual se define la curva elíptica. Los multiplicadores de campo finito propuestos en el presente trabajo operan con elementos de 192 bits, representados mediante una base polinomial.

Es importante remarcar que la importancia de proponer un multiplicador eficiente de campo finito binario con aplicaciones en sistemas de criptografía de curva elíptica reside en que la aritmética de curva elíptica tiene como base a la aritmética de campo finito. Lo anterior se describe de forma gráfica en la fig. 7 y así mismo ha sido explicado en la sección 2.6.

4.1. DIAGRAMA GENERAL DE BLOQUES

El diagrama general de bloques propuesto hace referencia a una arquitectura de multiplicador de 192 bits, es decir, el multiplicador operará elementos que pertenezcan al campo GF(2^{192}), de esta manera se tiene que m = 192. El diagrama está compuesto de la siguiente manera: el bloque principal consiste en los componentes MULT y ADD(XOR), componente multiplicativo y componente aditivo respectivamente, a su vez este bloque principal tiene dos entradas y una salida. Las entradas A(x) y B(x) en realidad son polinomios, y como se ha visto anteriormente, el grado de los polinomios es m-1, aunque estos polinomios también pueden ser vistos cada uno como un vector de bits de longitud m, el caso de la salida C(x) es similar, ya que es también un polinomio de grado m-1. Este diagrama general de bloques se muestra en la fig. 14. Los componentes MULT y ADD(XOR) son explicados a continuación.

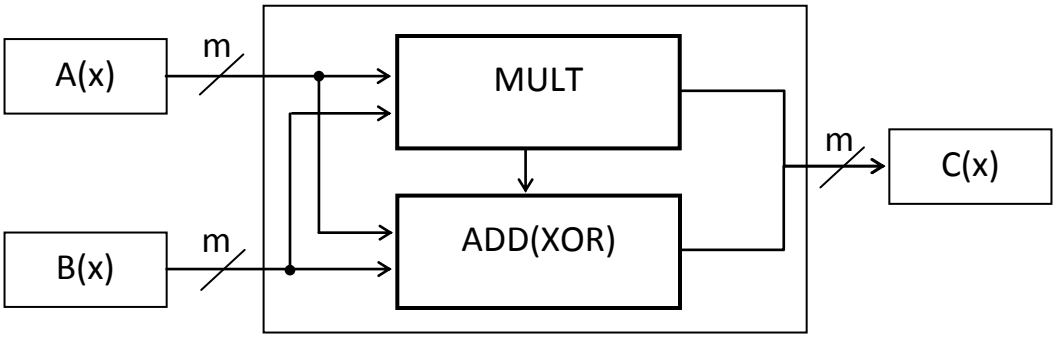

Figura 14: Diagrama general de bloques para los componentes propuestos

4.1.1. COMPONENTE ADITIVO

El diagrama de bloques del componente aditivo se propone de la siguiente forma: un bloque principal ADD(XOR) que realiza la operación adición, debe recordarse que esta adición aritmética es realizada módulo dos, la cual es equivalente a la operación lógica xor, el componente aditivo

también cuenta con dos entradas A(x) y B(x) y con una salida C(x), las cuales son polinomios de grado m-1. Este componente aditivo se muestra en la fig. 15, debe tenerse en cuenta que el valor de m = 192.

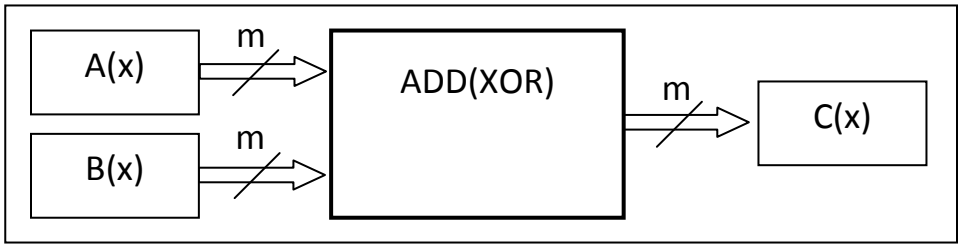

Figura 15: Diagrama de bloques del componente aditivo

El componente principal ADD(XOR) es propuesto como un arreglo de m compuertas xor, las cuales se encuentran alineadas de forma paralela, es decir, cada una de las m compuertas procesa de manera simultánea los datos de entrada A(i) y B(i) y así proporcionar una salida C(i). Una ventaja del diseño en paralelo es que al ejecutar el procesamiento de forma concurrente es posible conseguir un menor tiempo de ejecución. Este componente aditivo, fig. 14, realiza la operación C(x)=A(x) + B(x).

4.1.2. COMPONENTE MULTIPLICATIVO

El diagrama de bloques del componente multiplicativo, que se muestra en la fig. 16, está propuesto de la siguiente manera: dos bloques principales MULT y MOD, los cuales realizan la operación multiplicación y reducción respectivamente, de manera general se tienen dos polinomios de entrada A(x) y B(x), ambos polinomios de grado m-1, y se tiene un polinomio de salida C(x) de grado m-1. El valor de m = 192.

El bloque MULT realiza la operación R(x) = A(x) · B(x), esta operación es una multiplicación de polinomios y puede ser realizada mediante diferentes algoritmos de multiplicación, a manera de ejemplo se mencionan algunos: suma y desplazamiento, matrices, Karatsuba-Ofman, etc. Las

entradas de este bloque son los polinomios de grado m-1 A(x) y B(x), y se tiene como salida un polinomio de grado máximo 2·m-2 R(x).

El bloque MOD realiza la operación C(x) = R(x) mod P(x), esta operación es llamada reducción polinomial, y puede ser realizada por diferentes algoritmos de reducción, por ejemplo: suma y desplazamiento, reducción de Barret, etc. El grado máximo del polinomio R(x) es 2·m-2 y su vector de bits de representación tiene una longitud de 2m, el polinomio irreducible P(x) es de grado m, y el polinomio C(x), que es la salida final del multiplicador tiene grado m-1, y su vector de bits de representación es de longitud m.

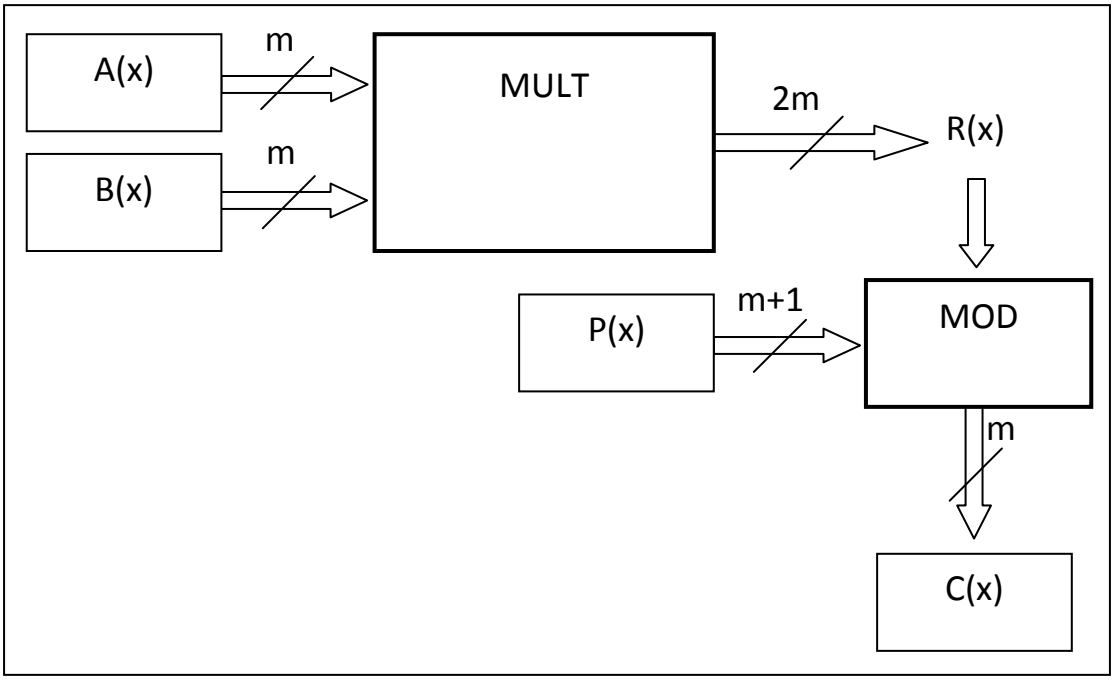

Figura 16: Diagrama de bloques del componente aditivo

4.2. DISEÑO DE COMPONENTES BASE

Para la implementación de la arquitectura del multiplicador de 192 bits se hace uso de diseños de arquitecturas de multiplicadores base, las cuales

son de menor longitud, es decir, operan elementos de campos finitos más pequeños. Estas arquitecturas base se deben integrar de manera conjunta para lograr la implementación de una arquitectura de 192 bits. Este esquema de integración se presenta más adelante.

Para efectos de diseño se hace uso de los mismos componentes descritos con anterioridad, sólo cambia el valor de m dependiendo de la arquitectura y/o algoritmo que se busca implementar. Así mismo se tienen que implementar diseños de multiplicadores base, los cuales se apoyan de un sumador base. Las implementaciones del sumador base y de distintos multiplicadores base son descritas a continuación.

4.3. IMPLEMENTACIÓN DE SUMADOR BASE

Este sumador base ha sido implementado usando un dispositivo FPGA XC3S200 de la familia Spartan-3. Las características de la tarjeta como su descripción general han sido abordados ampliamente con anterioridad en el punto 3.2.1.

Se propone un sumador base que recibe dos vectores de entrada de longitud m y que proporciona una salida de la misma longitud m. Como se mencionó anteriormente, este sumador realiza la operación xor de los dos vectores de entrada de forma simultánea, ya que cuenta con un arreglo de compuertas xor alineadas de forma paralela, es decir, cada compuerta xor realiza la operación cx(i) = ax(i) xor bx(i).

La simulación del sumador de longitud 16 se presenta en la fig. 17. El tiempo desde que existe una entrada hasta que existe una salida es de 2 ps., de manera que en 1 ps. se realiza la carga de datos y en 1 ps. se realiza la operación. A partir de estos resultados se puede concluir que el rendimiento del componente es bueno. Este sumador no depende de ninguna señal de reloj, así que la simulación de la operación fue llevada en el tiempo por

defecto de la tarjeta. Este componente es usado en todos los algoritmos de multiplicación que han sido implementados en esta tesis, la diferencia notable es que dependiendo de la longitud del multiplicador cambia la longitud del sumador.

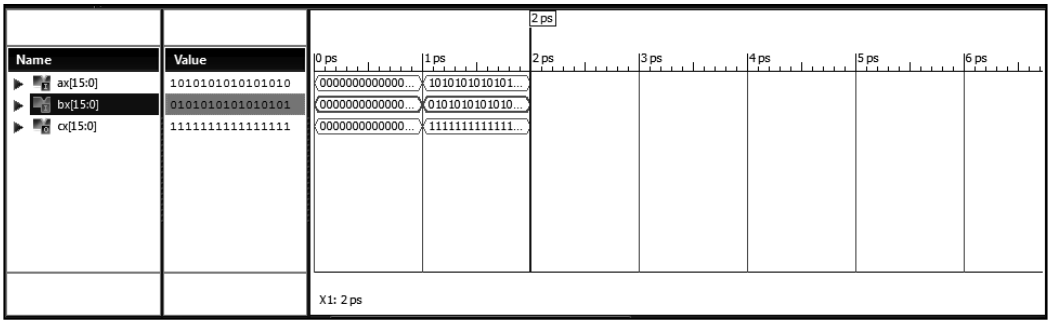

Figura 17: Simulación de un sumador de 16 bits

El área total que ocupa el diseño del sumador de 16 bits se reporta en la tabla 4, puede notarse que de manera general el sumador ocupa pocos recursos del dispositivo FPGA. El número de slices ocupados por el sumador son 9 de un total de 1920, el número de LUTs usadas es de 16 de un total de 3840, el número total de IOBs usados es de 48 de un total de 173, lo que representa el mayor consumo de recursos. De manera general se puede concluir que de las 480 CLBs con que cuenta el dispositivo han sido usadas solamente 2, lo que representa un porcentaje de uso del 0.42%. De acuerdo a lo anterior se puede concluir que el sumador es eficiente.

Tabla 4: Cantidad de uso del dispositivo para un sumador de 16 bits

Cantidad de uso del dispositivo			
Uso lógico	**Usado**	**Disponible**	**Porcentaje Usado**
Número de slices	9	1920	0%
Número de LUTs de 4 entradas	16	3840	0%
Número de IOBs	48	173	27%

4.4. IMPLEMENTACIÓN DE MULTIPLICADORES BASE

Como propuesta general se implementaron diferentes algoritmos de multiplicación, los cuales son descritos más adelante, estos multiplicadores base tienen diferente longitud, y en base a la eficiencia y rendimiento de cada uno de ellos, tomando como métricas lo descrito con anterioridad, se ha realizado una selección de un multiplicador base para proponer un multiplicador de 192 bits y así diseñar un componente que tenga aplicaciones en sistemas de Criptografía de Curva Elíptica.

Como se ha mencionado anteriormente, la implementación de los multiplicadores base ha sido realizada utilizando el FPGA XC3S200 de la familia Spartan-3 [59]. Una descripción detallada de esta tarjeta puede encontrarse en la sección 3.2.1, así también sus características generales pueden revisarse en la anterior Tabla 4.

4.4.1. PROPUESTA 1 - MULTIPLICADOR BASE DE 16 BITS

Un multiplicador de suma y desplazamiento se realiza de la siguiente forma, a manera de ejemplo, en la fig. 18 se presenta de forma gráfica un multiplicador de 9 bits realizado por medio de sumas y desplazamientos. Se tienen dos vectores operadores de longitud m=9, el algoritmo se repite desde i=1 hasta m, si el multiplicador en la posición i es 0 entonces no se realiza ninguna operación, y a continuación se realiza un desplazamiento de un bit a la izquierda, si el multiplicador en la posición es 1 entonces se hace una operación xor entre el multiplicador y un polinomio acumulador, y a continuación se realiza un desplazamiento de un bit a la izquierda. El algoritmo termina una vez que valor de i sea mayor a m, y es entonces cuando se tiene un resultado, el cual será un vector de longitud 2m.

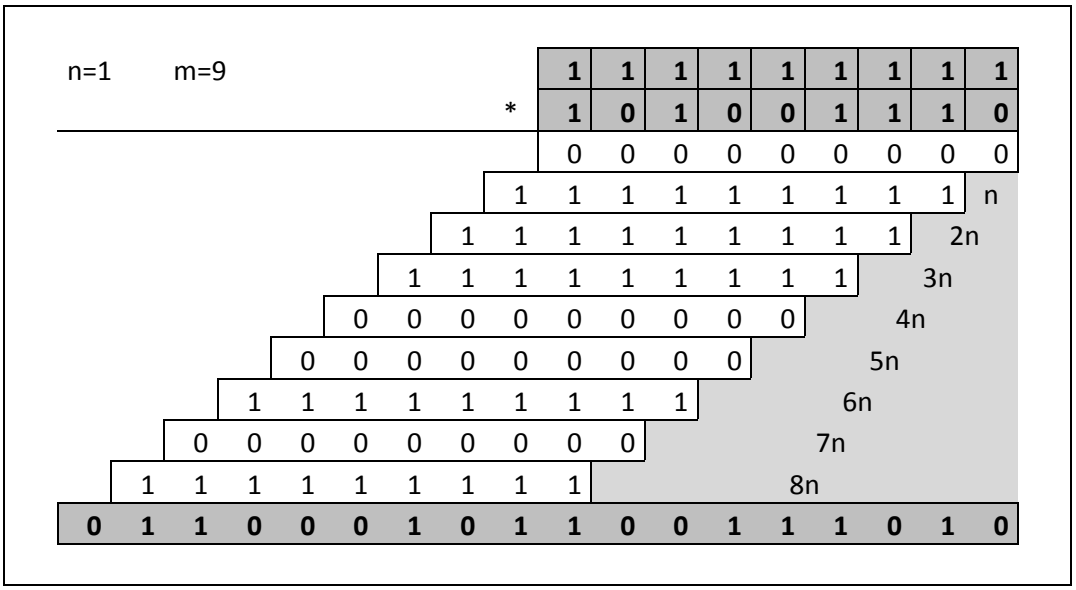

Figura 18: Ejemplo- Algoritmo gráfico de un multiplicador de suma y desplazamiento

La forma en la cual se ha desarrollado este algoritmo de suma y desplazamiento ha sido mediante una Máquina de Estados Finitos o *FSM* (por sus siglas en inglés). Se tienen 4 estados: pad, orx, shift y finish, la máquina de estados entra por defecto al estado pad, y después permanece un cambio entre los estados orx y shift hasta que se cumpla la condición i mayor a m, en ese momento cambia al estado finish y se mantiene en ese estado finalizando así el multiplicador.

La simulación del multiplicador de suma y desplazamiento de 16 bits se presenta en la fig. 19, se tienen dos vectores entradas de longitud 16, se realiza el algoritmo de suma y desplazamiento y se obtiene un vector de salida de 32 bits. Para realizar la simulación se han tomado los siguientes valores: el periodo de la señal de reloj se ha definido en 2 *ps*., con un flanco de subida en 1 y un flanco de bajada en 0, con un ciclo de trabajo del 50%.

El tiempo en el cual se ha llevado a cabo la multiplicación de dos polinomios de 16 bits es de 64 *ps*., lo cual es un buen tiempo para este tipo de multiplicadores, pero es lento comparado con otros algoritmos de

multiplicación. Se podría decir que este multiplicador tarda 4 ciclos de reloj en sumar un polinomio y hacer un corrimiento a la izquierda.

A partir de lo anterior se puede deducir que los 64 *ps*., que es tiempo total para la multiplicación, es el resultado de multiplicar 4 * 16, donde 4 ciclos de reloj es el tiempo que se tarda en realizar una "suma más un desplazamiento" y 16 es el número de veces que debe hacer "suma más un desplazamiento".

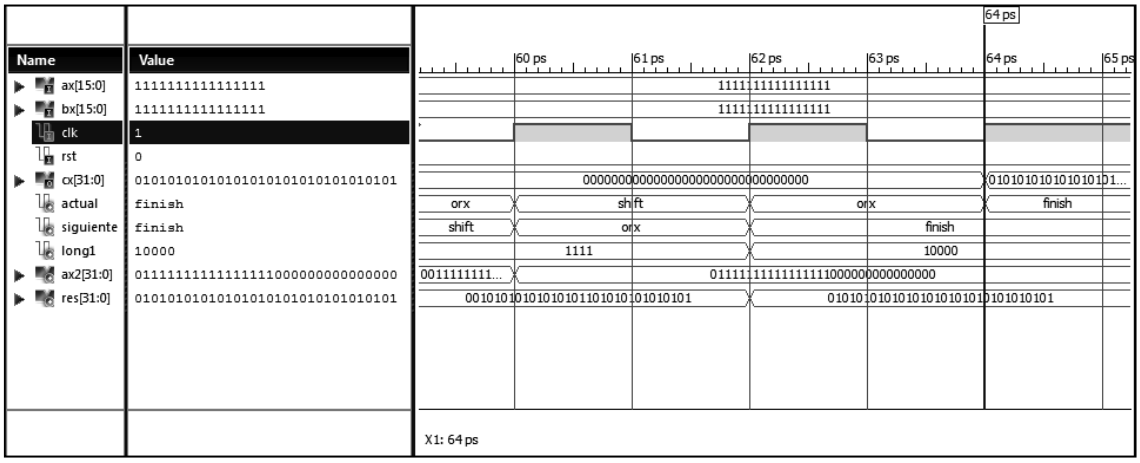

Figura 19: Simulación del multiplicador de 16 bits por medio de una Máquina de Estados Finitos

El área total que ocupa el diseño del multiplicador de 16 bits se reporta en la tabla 5. El número de slices ocupados por el diseño son 68 de un total de 1920, el número de slices de Flip/Flops es de 100 de un total de 3840, las *LUTs* de 4 entradas usadas son 100 de un total de 3840, el número de *IOBs* es de 66 de un total de 173, el número de *GCLKs* usados es de 3 de un total de 8. Teniendo en cuenta los resultados obtenidos se puede concluir que la eficiencia de este multiplicador no supera los resultados esperados.

Tabla 5: Cantidad de uso del dispositivo para un multiplicador de 16 bits

Cantidad de uso del dispositivo			
Uso lógico	**Usado**	**Disponible**	**Porcentaje Usado**
Número de slices	68	1920	3%
Número de slices de Flip/Flops	100	3840	2%
Número de LUTs de 4 entradas	102	3840	2%
Número de IOBs	66	173	38%
Número de GCLKs	3	8	37%

4.4.2. PROPUESTA 2 - MULTIPLICADOR BASE DE 8 BITS

Este multiplicador de 8 bits ha sido implementado utilizando programación por componentes. Este multiplicador tiene dos vectores de entrada de 8 bits de longitud, y se realiza una multiplicación usando componentes de suma y desplazamiento, y de esta manera obtenemos un vector de salida de 16 bits de longitud. Para realizar la simulación de este multiplicador se ha ajustado el tiempo de resolución en 1 *ps*.

Como se muestra en la fig. 20, el tiempo de simulación en el cual se lleva a cabo la multiplicación de dos entradas de 8 bits es de 2 *ps*., es posible ver que en 1 *ps*. se realiza la carga de datos y a continuación, en 1 *ps*. se realiza la multiplicación. En base a estos resultados se puede asumir que el rendimiento del componente es muy bueno.

Es importante señalar que aunque este multiplicador no depende de ninguna señal de reloj se debe tener muy en cuenta que en conjunto los componentes deben estar sincronizados, con el fin de obtener los resultados esperados, así también se debe mencionar que este multiplicador por componentes hace uso de 16 sumadores iguales al sumador base de 16 bits explicado con anterioridad.

.

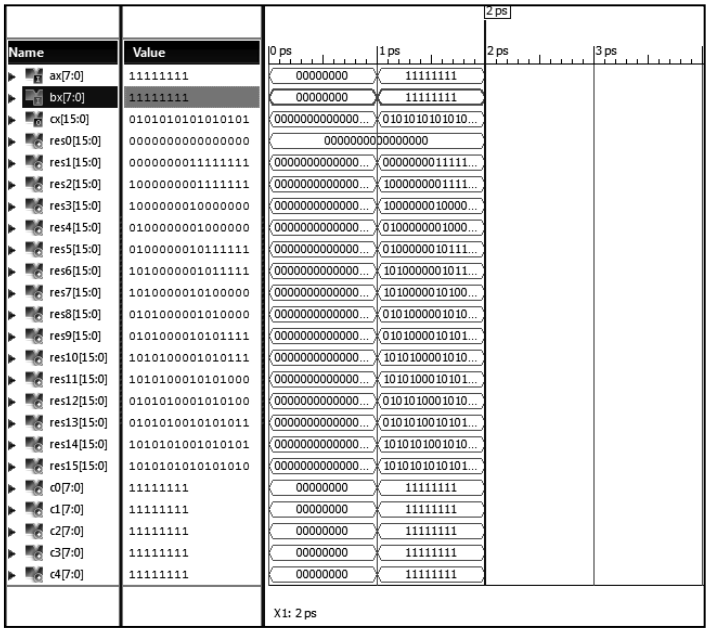

Figura 20: Simulación de un multiplicador de 8 bits

El uso de área que ocupa la implementación del multiplicador de 8 bits se presenta a continuación en la tabla 5. El número de slices usados es de 28 de un total de 1920, lo cual representa sólo el 1%; del total de 3840 *LUTs* de 4 entradas disponibles se han ocupado 102, lo cual representa el 2%; y por último se han usado 32 *IOBs* de un total de 173 disponibles, esto representa el 18% del dispositivo. A partir de los resultados mostrados en la tabla 5 se puede concluir que la propuesta de este multiplicador base de 8 bits es eficiente.

Tabla 6: Cantidad de uso del dispositivo para un multiplicador de 8 bits

Cantidad de uso del dispositivo			
Uso lógico	**Usado**	**Disponible**	**Porcentaje Usado**
Número de slices	28	1920	1%
Número de LUTs de 4 entradas	102	3840	2%
Número de IOBs	32	173	18%

4.4.3. PROPUESTA 3 - MULTIPLICADOR BASE DE 12 BITS

La propuesta de un multiplicador base de 12 bits se ha hecho por medio de componentes. Este multiplicador base tiene dos vectores de entrada de 12 bits de longitud, una vez que se han guardado las entradas, se realiza una multiplicación ayudándose de sumas y desplazamientos, una vez realizada esta operación, se obtiene un vector de salida de 24 bits de longitud. Para efectos de simulación de este multiplicador base de 12 bits, el tiempo de resolución del reloj ha sido a 1 *ps*.

En la figura 21 se presenta la simulación del multiplicador base de 12 bits, el tiempo total desde que existen dos vectores de entrada, estos vectores se multiplican y se obtiene una salida es de 2 *ps*., si tenemos en cuenta que la carga de datos se realiza en 1 *ps*., entonces la multiplicación y por consiguiente la obtención de un resultado tarda sólo 1 *ps*. Algo que se debe tener en cuenta acerca de este multiplicador es al no depender de ninguna señal de reloj, es un multiplicador asíncrono. Y para obtener los resultados deseados los componentes internos de este multiplicador deben estar en sincronía entre ellos. Al igual que los otros multiplicadores propuestos, este multiplicador se apoya en el sumador base presentado con anterioridad.

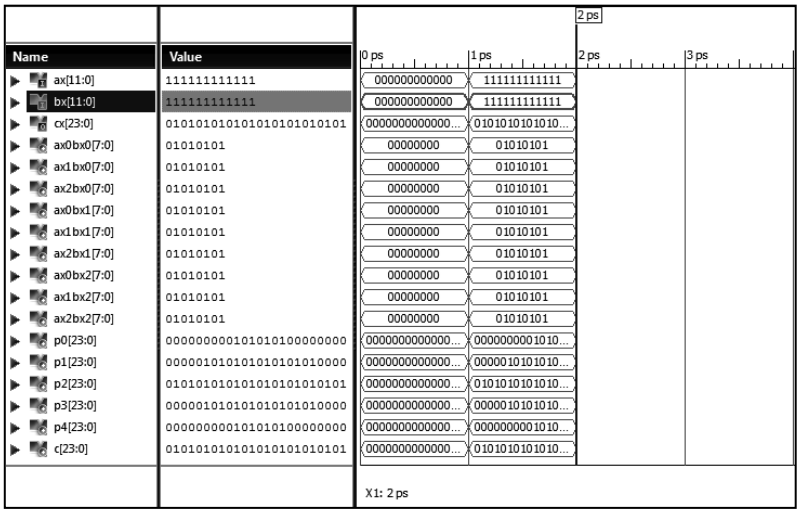

Figura 21: Simulación de un multiplicador de 12 bits

El espacio total que ocupa la simulación del multiplicador base de 12 bits se presenta en la tabla 7. De un total de 1920 slices disponibles se ocupan 56, lo que representa un 2% de uso; de 3840 *LUTs* de 4 entradas con las que cuenta el dispositivo se ocupan un total de 98, que significa el 2%; y por último, de un total de 173 *IOBs* con los que cuenta el dispositivo se usan 48, para dar un porcentaje de 27. El espacio total que ocupa el multiplicador de 12 bits tiene una buena relación con el tiempo en el cual se realiza la operación.

Tabla 7: Cantidad de uso del dispositivo para un multiplicador de 12 bits

Cantidad de uso del dispositivo			
Uso lógico	**Usado**	**Disponible**	**Porcentaje Usado**
Número de slices	56	1920	2%
Número de LUTs de 4 entradas	98	3840	2%
Número de IOBs	48	173	27%

4.4.4. PROPUESTA 4 - MULTIPLICADOR BASE DE 16 BITS

La simulación de un multiplicador de 16 bits programado por componentes se muestra en la fig. 22, el tiempo total que se necesita para dicha simulación es de 2 *ps*., de manera específica el tiempo en que se realiza la carga de datos es de 1 *ps*., en este tiempo se tienen dos vectores de entrada de 16 bits de longitud; en el siguiente *ps*. se realiza la multiplicación de esos polinomios de entrada y se obtiene como resultado un vector de salida de 32 bits de longitud.

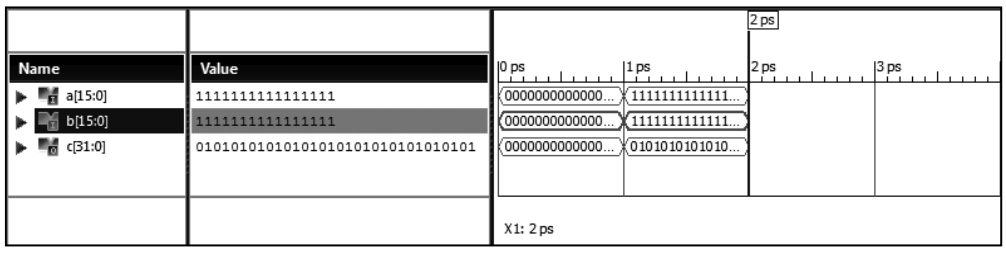

Figura 22: Simulación de un multiplicador de 16 bits

Para realizar la simulación de este multiplicador de 16 bits se ha ajustado el tiempo de resolución del reloj en 1 *ps*. Aunque este diseño no hace uso directamente de una señal de reloj, es importante tener en cuenta la sincronización de cada una de los componentes para que estos funcionen de manera conjunta y se obtengan los resultados esperados. Al igual que las propuestas anteriores, este multiplicador se apoya del sumador base que se ha explicado anteriormente.

En lo que respecta al área que consume el diseño de un multiplicador de 16 bits se obtuvieron los siguientes resultados: de un total de 1920 slices disponibles se han ocupado 98, lo cual representa un 5% de uso; las *LUTs* que se han ocupado son 171 de un total de 3840 disponibles, que es el 4% de uso del dispositivo; referente al número de *IOBs* que se han ocupado en el diseño se tiene 64 de un total de 173 del total disponible, que significa un uso del 36% de la tarjeta FPGA. Los resultados anteriores demuestran que el diseño del multiplicador base actual es viable para ocuparse en un esquema de torres de campos y así proponer una solución al objetivo planteado.

Tabla 8: Cantidad de uso del dispositivo para un multiplicador de 16 bits

Cantidad de uso del dispositivo			
Uso lógico	**Usado**	**Disponible**	**Porcentaje Usado**
Número de slices	98	1920	5%
Número de LUTs de 4 entradas	171	3840	4%
Número de IOBs	64	173	36%

4.4.5. SELECCIÓN DE MULTIPLICADOR BASE

Se debe seleccionar un multiplicador que sirva de base para la integración de un multiplicador que opere elementos de 192 bits de longitud, para realizar la selección del multiplicador base se tomaron en cuenta las directrices de diseño propuestas y descritas con anterioridad: eficiencia y rendimiento. Estas directrices de diseño determinan en mayor medida la

selección del diseño propuesto del multiplicador base, aunque de la misma manera se tomaron en cuenta los requerimientos de seguridad con base en especificaciones que dicta el NIST, como lo es la base para representar elementos del campo finito y la longitud en bits.

Es importante mencionar que las propuestas de multiplicadores base son en esencia diferentes, además de operar elementos con diferente longitud de bits, han sido diseñados mediante diferentes esquemas de diseño como lo son maquina de estados finitos o por componentes. En la tabla 9 se presenta una comparativa entre los diferentes multiplicadores base propuestos.

Tabla 9: Comparativa de uso del dispositivo en multiplicadores base propuestos

Cantidad de uso del dispositivo					
	Usado				
Uso lógico	Prop. 1 m = 16	Prop. 2 m = 8	Prop. 3 m = 12	Prop. 4 m = 16	Disponible
Número de slices	68	28	56	98	1920
Número de slices de Flip/Flops	100	-------	-------	-------	3840
Número de LUTs de 4 entradas	102	102	98	171	3840
Número de IOBs	66	32	48	64	173
Número de GCLKs	3	-------	-------	-------	8

En base a los resultados de simulación obtenidos y mostrados en la tabla anterior, y de manera gráfica en la figura 23, se puede apreciar lo siguiente:

- Respecto al número de slices ocupados, la **propuesta 2** ocupa menos recursos; puede apreciarse incluso que la propuesta 3 ocupa el doble de slices que la **propuesta 2**; y que la **propuesta 1**, la cual tiene operandos con una longitud del doble de la **propuesta 2** ocupa más del 140%; la **propuesta 4** ocupa más del 40% que la propuesta 1, a pesar de operar elementos de la misma longitud.

- Respecto al número de slices de flip/flops la **propuesta 1** es la única que hace uso de estos recursos, manteniéndose las demás propuestas en 0% de uso.

- Respecto al número de *LUTs* de 4 entradas la **propuesta 3** es la que menos recursos consume, incluso operando elementos con longitud mayor en 50% a la **propuesta 2**; la **propuesta 1** es la que tiene una mejor relación entre la longitud de los operandos y el uso de *LUTs*, al utilizar los mismos recursos que la **propuesta 2** operando elementos del doble de longitud, usando 4% más de recursos que la propuesta 3 operando elementos con una longitud mayor en 25%, y por último, usando cerca del 60% menos de recursos operando elementos con la misma longitud.

- Respecto al número de *IOBs* la **propuesta 2** es la que ocupa menos recursos, pero esto se debe únicamente a que opera elementos de una menor longitud, en este caso en especial puede decirse que por cada bit de longitud se ocupan cuatro *IOBs*, manteniendo así una relación lineal, es decir el de 8 bits de longitud usa 32 *IOBs*, el de 12 usa 48, el de 16 usa 64; el diseño de la **propuesta 1** usa sólo dos *IOBs* extras. De esta manera ningún diseño posee ventajas sobre los demás diseños.

- Respecto al número de *GCLKs* la **propuesta 1** es la única que hace uso de estos recursos, manteniéndose las demás propuestas en 0% de uso.

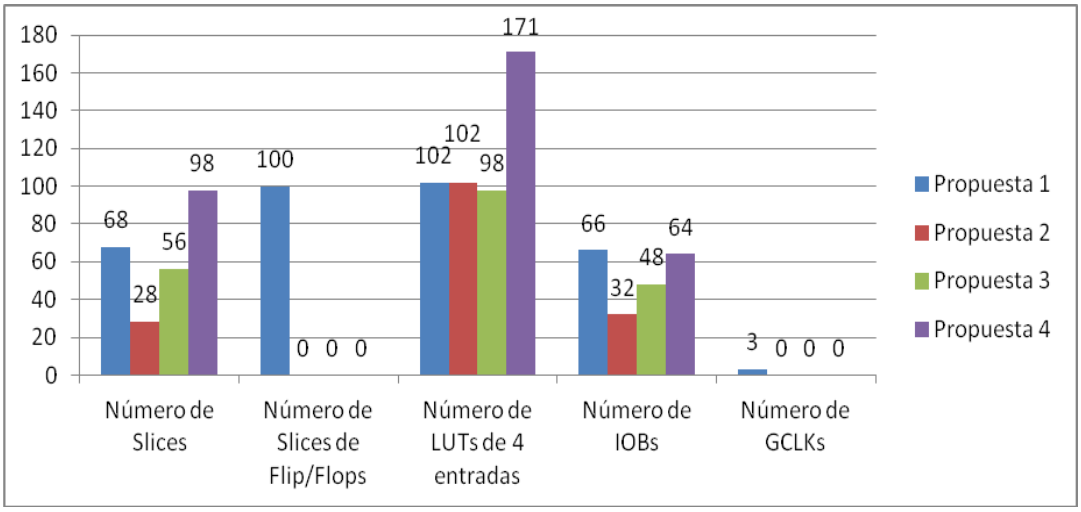

Figura 23: Gráfica comparativa de uso en multiplicadores base propuestos

Los tiempos de simulación de los multiplicadores propuestos pueden verse en la tabla 10 y a manera de gráfica en la fig. 24. Las propuestas 2, 3 y 4 tienen un tiempo de simulación de 2 *ps*., en este tiempo se logra leer una entrada/s, esta se procesa y existe una salida. La propuesta 1 tiene un tiempo de simulación de 64 *ps*., puede apreciarse que para procesar cada bit del operando (16 bits en el caso de la propuesta 1) se necesitan 4 *ps*.

De manera general las propuestas 2, 3 y 4 no representan ventajas referentes al tiempo de simulación, en cambio la propuesta 1 tarda demasiado tiempo en proporcionar una salida.

Tabla 10: Tiempo de simulación de los multiplicadores base propuestos

Tiempo de Simulación de Multiplicadores Propuestos			
Propuesta 1 m = 16	**Propuesta 2 m = 8**	**Propuesta 3 m = 12**	**Propuesta 4 m = 16**
64 ps.	2 ps.	2 ps.	2 ps.

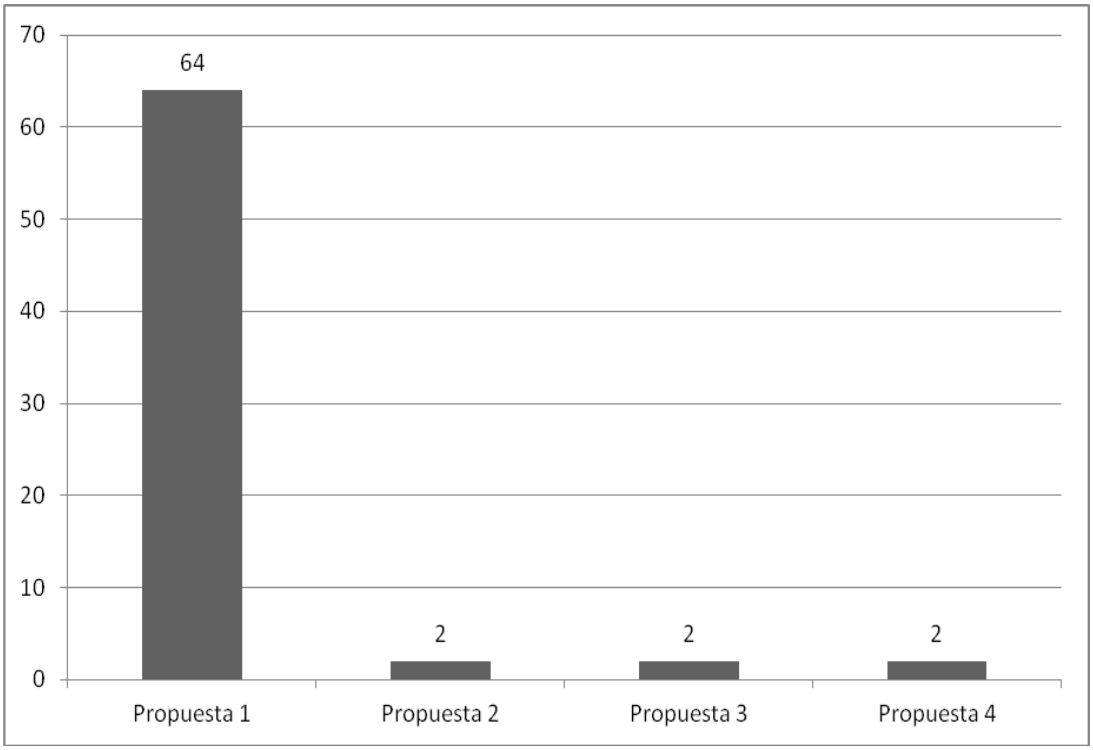

Figura 24: Gráfica de tiempo de simulación de los multiplicadores base propuestos

De acuerdo al análisis anterior se puede concluir que las propuestas ofrecen diferentes ventajas en las diferentes características analizadas, aunque es importante remarcar que cada característica es muy importante para realizar la selección del multiplicador base.

En relación al número de slices y al número de *LUTs*, que son las características en las cuales se aprecia una diferencia notable entre todas las propuestas de multiplicadores base, se realiza la selección del multiplicador. Las propuestas 2 y 3 son las que ocupan menos recursos de la tarjeta, como se ha visto en el análisis anterior ambas tienen una ventaja y una desventaja con respecto a la otra:

- ✓ Propuesta 2: Ocupa un 4% más de *LUTs* que la propuesta 1 operando entradas menores en un 33%.

- ✓ Propuesta 3: Ocupa un 100% más de slices que la propuesta 2 operando entradas mayores sólo en un 50%.

Por lo anterior, y en relación a las ventajas y desventajas de cada multiplicador base, se ha seleccionado la **propuesta 2** como multiplicador base para proponer un multiplicador polinomial que opere elementos en el campo GF(2^{192}).

CAPITULO 5.

MULTIPLICADOR POLINOMIAL DE 192 BITS

La propuesta del multiplicador de 192 bits se realiza usando un esquema de torres de campos mediante multiplicadores compuestos simétricos. Los resultados obtenidos por el multiplicador propuesto se encuentran en un rango competitivo comparado con otros trabajos recientes.

De acuerdo al estándar ANSI x9.62 se considera una aplicación segura cuando opera tamaños de llave mayor a los 160 bits, en este caso el multiplicador propuesto en este capítulo opera elementos de 192 bits.

La aplicación para el multiplicador propuesto es la ECC, aunque de una manera general, podría tener aplicaciones en cada una de las áreas que tiene como apoyo a los campos finitos binarios, como el procesamiento digital de señales, la teoría general de códigos, y por supuesto las aplicaciones criptográficas.

5.1. DISEÑO DEL MULTIPLICADOR

Componer un multiplicador a partir de multiplicadores más pequeños es similar a realizar una multiplicación por medio de un método de escuela, es decir, realizar la multiplicación "a mano", fig. 18.

Para realizar una multiplicación muy grande, una buena estrategia es auxiliarse mediante bloques de multiplicadores más pequeños, como se muestra en la fig. 25. En esta figura se describe un multiplicador simétrico por bloques [60], donde cada bloque es un multiplicador más pequeño, este multiplicador más pequeño opera elementos de longitud m, es decir realiza la operación m * m. A manera de ejemplo, hay que suponer que A0 es multiplicador de 8 bits, es decir 8 bits * 8 bits, entonces el multiplicador simétrico por bloques es un multiplicador de 32 bits, es decir opera 32 bits * 32 bits. El número total de multiplicadores más pequeños requeridos para componer un multiplicador simétrico por bloques es n * n, que es la multiplicación de los bloques requeridos para componer el multiplicando por el multiplicador (en este caso el mismo número), para este ejemplo en específico se tiene 4 * 4 = 16.

Puede notarse que el número de resultados intermedios es igual al número de multiplicadores p1,...,p16, que se ordenan como se muestra en la fig. 25, estos resultados intermedios pueden verse como un sumador de árbol con una profundidad de 16, para obtener el resultado de esta suma podrían usarse como mínimo 16 unidades de tiempo, teniendo en cuenta que al no existir acarreos la suma de cada resultado intermedio es una operación lógica xor.

A diferencia de realizar la multiplicación "a mano", mediante este esquema se pueden realizar todas las multiplicaciones de manera simultánea, al no depender de resultados anteriores, e incluso al poseer la

ventaja de carecer de acarreos, como lo es el caso de los campos finitos binarios. Esto ha sido basado en el esquema de "divide y vencerás".

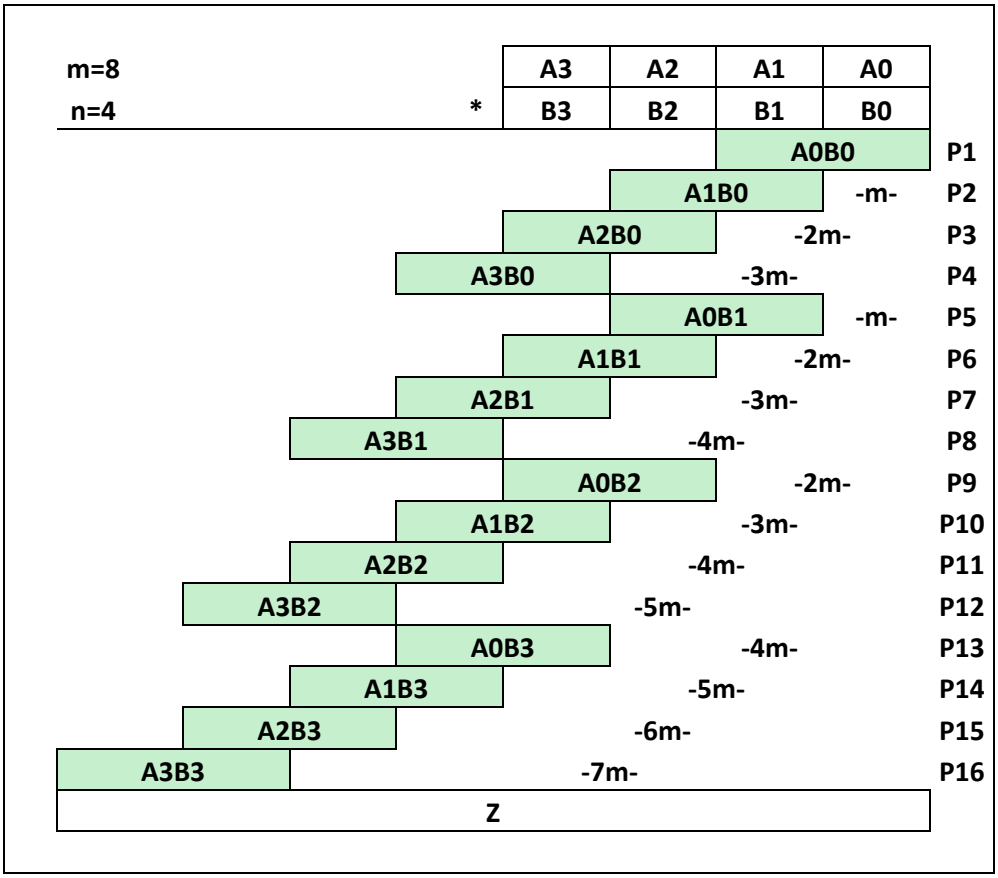

Figura 25: Multiplicador simétrico por bloques con profundidad de n * n

La profundidad del sumador de árbol puede ser reducida simplemente reagrupando los resultados intermedios que no se sobreponen con otros resultados. Esto se puede llevar a cabo de una manera eficiente por medio de una concatenación, y no mediante una suma, lo cual significa que al reagrupar no se genera latencia.

En la fig. 26 se presenta cómo pueden reagruparse los resultados para reducir la profundidad del sumador [60]. Puede notarse que no es posible reagrupar todos los resultados, porque existen algunos que se

sobreponen con los demás. Al realizar este reagrupamiento se crean en total (2 * n) - 1 resultados intermedios, lo cual significa que como mínimo la suma puede ser llevada a cabo en 7 unidades de tiempo, que representa una reducción de 9 unidades de tiempo.

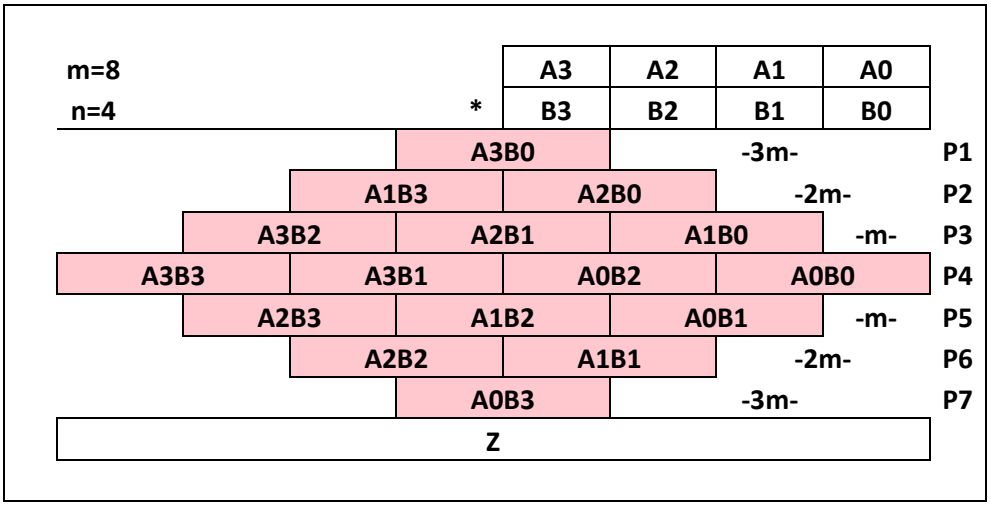

Figura 26: Multiplicador simétrico por bloques reagrupado con profundidad de (2 * n) - 1

Cabe hacer énfasis en que el diseño del multiplicador polinomial de 192 bits está basado en un esquema de torres de campos y un esquema de multiplicador simétrico por bloques. De esta manera el multiplicador del campo $GF(2^{192})$ en un esquema de torres de campos puede representase de la forma $GF(((2^8)^6)^4)$, usando el esquema de multiplicador simétrico por bloques se tendría:

- En la etapa $GF(2^8)$ se tendría el multiplicador base diseñado e implementado que se ha presentado en el capitulo anterior.
- En la etapa $GF((2^8)^6)$ se tendría un multiplicador simétrico por bloques con valores de m = 8 y n = 6, lo cual representa un multiplicador de 48 bits * 48 bits usando 36 multiplicadores más pequeños señalados en la etapa anterior, en decir 36 multiplicadores base de 8 bits * 8 bits.

- En la etapa GF(((2^8)^6)^4) se tiene un multiplicador simétrico por bloques con valores de m = 6 y n = 4, lo cual, si se toma en cuenta la etapa anterior, resulta en un multiplicador de 192 bits (8 * 6 * 4). Esto significa que se tienen 16 multiplicadores de 48*48 bits, los cuales a su vez están conformados por multiplicadores base.

La implementación de estos dos esquemas trabajando en conjunto se presenta a continuación.

5.2. SIMULACIÓN DEL MULTIPLICADOR

La implementación del multiplicador puede verse en la fig. 27. Se presenta la imagen de la simulación, los resultados obtenidos se presentan en la siguiente sección.

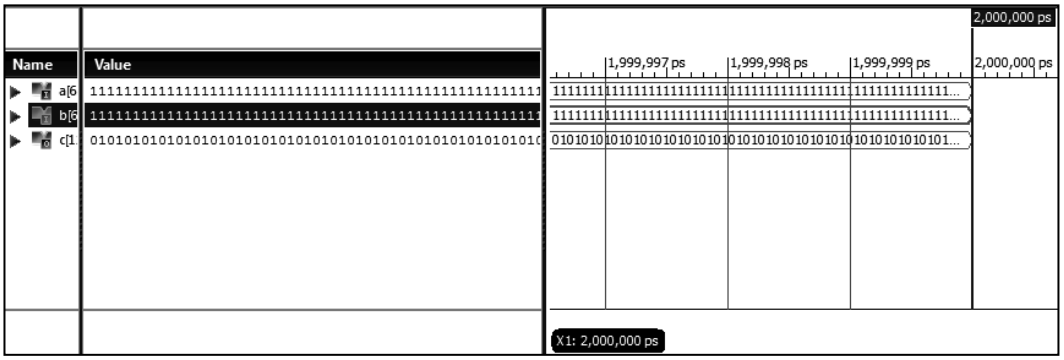

Figura 27: Simulación del multiplicador propuesto

Al igual que algunos de los multiplicadores base propuestos, este multiplicador no hace uso de una señal de reloj, aunque es muy importante tener en cuenta que cada uno de las etapas del multiplicador deben tener sincronía para obtener los resultados esperados.

5.3. RESULTADOS Y ANÁLISIS

Los resultados obtenidos de la simulación del multiplicador son presentados a continuación:

El tiempo de ejecución y el uso de memoria en el cual se lleva a cabo una multiplicación en el campo GF(2^{192}) está descrita en la fig. 28. Cabe mencionar que estos resultados son arrojados en el ISim de Xilinx una vez terminada la simulación del multiplicador.

Uso de Memoria: 17188 KB
Uso de CPU: 233 ns

Figura 28: Resultados obtenidos por medio de la simulación del multiplicador

El área que ocupa el proyecto puede verse en la tabla 11, el número total de slices ocupadas es de 1602 que representa un total de uso del 83%, el número de LUTs usadas en la implementación es de 2786 que representa un total de uso del 72%, finalmente, los IOBs ocupados son 256 de 173, lo que representa un 147% del total. Es importante hacer notar que si bien el número de IOBs que el dispositivo le asigna al usuario es de 173, internamente cuenta con casi el doble, por este motivo el dispositivo fue capaz de llevar a cabo la simulación del multiplicador.

Tabla 11: Uso total del dispositivo para el multiplicador propuesto

Cantidad de uso del dispositivo			
Uso lógico	**Usado**	**Disponible**	**Porcentaje Usado**
Número de slices	1602	1920	83%
Número de LUTs de 4 entradas	2786	3840	72%
Número de IOBs	256	173	147%

5.4. COMPARATIVA

En esta sección se muestran los resultados obtenidos en la implementación en comparación a trabajos relacionados, esta comparativa puede verse en la tabla 12:

Tabla 12: Comparativa entre la propuesta propia y trabajos relacionados

Comparativa				
	Propuesto	**Shieh, 2009 [61]**	**Kitsos, 2003 [62]**	**Chávez, 2011 [63]**
Longitud de m	**192**	163	192	256
Número de slices	**1602**	2490	---	4745
Tiempo	**233 ns.**	2.55 ms.	10.4 µs	15 ns.
Algoritmo	**Simétrico**	MMA	MSB-first, bit-serial	Asimétrico
Dispositivo	**XC3S200**	XCV1000E	XCV1000E	Virtex 5

Como se ha mencionado anteriormente, el multiplicador es el elemento que consume más recursos en el diseño. Por esta razón, siempre se busca una reducción en el consumo de área o una reducción en el tiempo en el cual se obtienen resultados. La longitud de m representa la longitud de los elementos a operar, el número de slices representa al total de slices que el diseño ha usado del dispositivo, el tiempo se refiere al tiempo que transcurre desde que existe una entrada hasta que se termina de obtener el resultado, el algoritmo se refiere al algoritmo de multiplicación usado para la realización del diseño, y por último, el dispositivo se refiere al dispositivo en el cual se ha implementado el diseño del multiplicador.

En la tabla anterior puede apreciarse que el multiplicador [61] ocupa un área mayor además de necesitar de mucho más tiempo para arrojar resultados, los resultados se muestran en milisegundos, en comparación a los resultados obtenidos en el presente trabajo, que son reportados en nanosegundos, además usa una longitud de m menor a la propuesta realizada en este trabajo, el algoritmo que ocupa es un algoritmo de

multiplicación modificado y el dispositivo en el cual ha sido implementado dicho diseño es una tarjeta XCV1000E.

En [62] el tamaño de m es el mismo que en el presente trabajo, no se da información referente al área total que ocupa el diseño, pero sí hay información referente al tiempo, el cual es un resultado intermedio entre la propuesta [61] y la propuesta del presente trabajo, pero incluso el tiempo es demasiado elevado al reportar resultados en microsegundos, el algoritmo de multiplicación usado en dicha propuesta es Most Significant Bit- first bit-serial y el dispositivo usado para realizar la implementación es, al igual que el trabajo analizado anteriormente, una tarjeta XCV1000E.

En [63] se implementa un multiplicador que opera elementos más grandes, el área total del dispositivo que ocupa es mayor a la usada por la propuesta de este trabajo, pero sacrifica el área a razón del tiempo, que es reportado en 15 nanosegundos, lo cual presenta una ventaja sobre el diseño propuesto, el algoritmo de multiplicación utiliza una estrategia similar a la de esta propuesta ya que utiliza también torres de campos, sin embargo difiere en que ocupa un esquema asimétrico en lugar de un esquema simétrico. A diferencia del presente trabajo el dispositivo que se ocupa para implementar el diseño es una tarjeta con mejores prestaciones (Virtex 5).

En lo que se refiere a espacio/tiempo se buscó encontrar un buen balance donde no se tuviera que sacrificar ninguna a costa de la otra, el resultado es un multiplicador que cumple con los requerimientos de seguridad y diseño que se plantearon al inicio del presente trabajo.

Las ventajas que el presente multiplicador tiene sobre las demás propuestas es que en todos los casos el área que ocupa es mucho menor, el tiempo total en el cual se obtienen resultados es muy competitivo al presentarse resultados en nanosegundos, al juntar estas dos ventajas espacio/tiempo es posible decir que el presente multiplicador es capaz de

adaptarse a un sistema en ambientes de recursos restringidos donde se necesite, a manera de ejemplo, cifrado en tiempo real. La tarjeta donde se ha implementado dicho trabajo es una tarjeta con prestaciones que están restringidas, lo cual ha estado alineado con los requerimientos planteados al inicio del presente trabajo.

CONCLUSIONES

El estándar ANSI x9.62 contempla aplicaciones bancarias seguras basadas en Criptografía de Curva Elíptica, mayores a 160 bits representados mediante una de las bases propuestas en el estándar, a partir de esto se hizo una comparativa entre las bases de campos finitos propuestas, y a partir de ahí se realizó la selección de la base tomando como referencia las ventajas y desventajas que las bases presentaban en relación a las otras.

Al utilizar un esquema de torres de campos, se tuvo que proponer un multiplicador base, para realizar esta propuesta se implementaron cuatro multiplicadores base, de los cuales, y en base a las directrices de diseño requeridas en este trabajo, se seleccionó uno de ellos para implementar el multiplicador de 192 bits.

Al inicio del presente trabajo se esperaba obtener resultados que estuvieran dentro de los parámetros del estado del arte actual, objetivo que se ha cumplido ya que se ha logrado un multiplicador que utiliza poca área del dispositivo, además de haber logrado un diseño eficiente para un multiplicador que opera entradas de 192 bits.

Los resultados obtenidos en este multiplicador y de acuerdo al análisis realizado son competitivos, aunque realizando diversos ajustes pueden mejorarse estos resultados. Los resultados pudieran variar si se tiene en cuenta que la estrategia de diseño fue realizar $GF(((2^8)^6)^4)$ y pudo realizarse de la forma $GF(((2^{16})^4)^3)$.

En el camino se ha aprendido que para un correcto diseño y una buena implementación de la arquitectura que se desea proponer, se debe tener en cuenta el dispositivo en el cual se desea implementar, con el fin de

aprovechar todas las ventajas y bondades que esta tecnología de hardware reconfigurable ofrece.

En todos los diseños siempre se busca la mejor relación espacio/tiempo, aunque en la mayoría de las veces se debe optar por la que tenga más peso en el objetivo que se persigue.

En base a que se han cumplido los objetivos propuestos al inicio del proyecto y se han cubierto los requerimientos de seguridad y diseño, además de tener en cuenta la comparativa de la arquitectura propuesta con trabajos relacionados, se concluye que el **diseño es bueno**.

TRABAJO A FUTURO

En este trabajo no se han abarcado todos los algoritmos de multiplicación, por esta razón el diseñar multiplicadores que operen de acuerdo a diferentes algoritmos de multiplicación es visto como trabajo a futuro.

Así mismo, diseñar multiplicadores que operen con elementos de campos finitos representados con diferente base, las cuales son descritas en el ANSI x9.62.

En el presente trabajo, de manera general, se han diseñado un sumador y un multiplicador, entonces como trabajo a futuro se tiene el diseñar componentes para realizar otras operaciones, tales como la inversión, la exponenciación, la raíz, etc.

Se pretende en un futuro utilizar un dispositivo de nueva generación, para la implementación y la simulación de las arquitecturas diseñadas y a diseñar.

Así mismo se pretende realizar un multiplicador que opere elementos mayores a 192 bits, para obtener mayor seguridad. También, con miras a trabajo futuro, se considera implementar un multiplicador que opere elementos de un campo finito primo.

Se busca la mejora en área/tiempo usando diferentes técnicas de diseño, diferentes algoritmos de multiplicación y diferentes dispositivos.

Por último, proponer un diseño en el cual todos los componentes aritméticos propuestos trabajen de manera conjunta con el objetivo de realizar el diseño, la implementación y la simulación para así proponer un procesador criptográfico.

REFERENCIAS

[1] D. Hankerson, A. Menezes y S. Vanstone, *Guide to Elliptic Curve Cryptography*. New York: Springer-Verlag, 2004, pp. 1-69.

[2] F. Rodríguez-Henríquez, N. A. Saqib, A. Díaz-Pérez y Ç. K. Koç, *Cryptographic Algorithms on Reconfigurable Hardware*. New York: Springer, 2006, pp. 7-186.

[3] A. Menezes, P. van Oorschot y S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996, pp. 1-86.

[4] M. Morales, *Notas sobre Criptografía*, INAOE, Tech. Rep. [Online]. Disponible: http://ccc.inaoep.mx/~mmorales/documents/Criptograf.pdf

[5] V. S. Miller, “Use of Elliptic Curves in Cryptography”, Lecture Notes in Computer Science, vol. 218, pp. 417-426. 1986.

[6] N. Koblitz, “Elliptic Curve Cryptosystems”, Mathematics of Computation, vol. 48, no. 177, pp. 203-209, Enero 1987.

[7] H. Cohen y G. Frey, *Handbook of Elliptic and Hyperelliptic Curve Cryptograhy*. Boca Raton: Chapman & Hall/CRC, 2006, pp. 201-238.

[8] S. Baktir, S. Kumar, C. Paar y B. Sunar, “A State-of-the-art Elliptic Curve Cryptographic Processor Operating in the Frequency Domain”, Mobile Networks and Applications, vol. 12, no. 4, pp. 259-270, Septiembre 2007.

[9] J. Ashenden y J. Lewis, *The designer’s guide to VHDL*. San Francisco: Morgan-Kaufmann Publishers, 1995, pp. 1-64.

[10] Kahn, D. *The Codebreakers.* New York: New Amercan Library, 1973, pp. 1-200.

[11] D. R. Stinson, *Cryptography Theory and Practice*. Boca Raton: Chapman & Hall/CRC, 2006, pp. 250-254.

[12] Galende-Díaz, J. La Criptografía Medieval: El libro del tesoro. Madrid: Universidad Complutense de Madrid, 2003.

[13] Fernández, S. La Criptografía Clásica. pp. 1-24, Abril 2004.

[14] Kerckhoffs, A. La Cryptographie Militaire. *Journal des sciences militaires*, pp. 5-38, 1883

[15] Wesolkowski, S. The Invention of Enigma and How the Polish Broke It Before the Start of WWII. *University of Waterloo, 2009.*

[16] Konheim, A. *Computer Security and Cryptography.* New Jersey: Wiley, 2007.

[17] Shannon, C. A Mathematical Theory of Communication. *The Bell System Technical Journal , 27*, pp. 379-423, 1948.

[18] Alcatel-Lucent. *Bell Labs Advances Intelligent Networks* [online]. Disponible: http://www.alcatel-lucent.com/wps/portal/!ut/p/kcxml/04_Sj9SPykssy0xPLMnMz0vM0Y_QjzKLd4w39w3RL8h2VAQAGOJBYA!!?LMSG_CABINET=Bell_Labs&LMSG_CONTENT_FILE=News_Features/News_Feature_Detail_000025.

[19] Diffie, W., y Hellman, M. New Directions in Cryptography. *IEEE Transactions on Information Theory , 22* (6), pp. 644-654, 1976.

[20] National Institute of Standards and Technology, Recommendation for Block Cipher Modes of Operation. *Methods and Techniques* . Gaithersburg, Estados Unidos de América, Diciembre de 2001.

[21] FIPS 197. Announcing the ADVANCED ENCRYPTION STANDARD (AES). Estados Unidos de América, noviembre de 2001.

[22] BlackBerry. *How the BlackBerry Enterprise Solution uses an AES encryption algorithm* [online]. Disponible: http://docs.blackberry.com/en/admin/deliverables/12873/How_BESolution_uses_AES_186609_11.jsp

[23] Radmin. *Solución para bancos* [online]. Dispoible: http://www.radmin.es/solutions/business/banks_and_atm_machines_administration.php

[24] National Institute of Standards and Technology. Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher. Gaithersburg, Estados Unidos de América, mayo de 2004.

[25] National Institute of Standards and Technology. *Computer Security Resource Center* [online]. Disponible: http://csrc.nist.gov/groups/ST/toolkit/documents/skipjack/skipjack.pdf

[26] van Tilborg, H. *Encyclopedia of Cryptography and Security.* New York: Springer, 2005.

[27] 3GPP. *Prohibiting A5/2 in mobile stations and other clarifications regarding A5 algorithm support* [online]. Disponible: http://www.3gpp.org/ftp/tsg_sa/TSG_SA/TSGS_37/Docs/SP-070671.zip

[28] Rivest, R., Shamir, A., & Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of ACM , 21* (2), pp. 120-126, 1978.

[29] ElGamal, T. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory , 31* (04), 469-472, 1985.

[30] Certicom. *An Elliptic Curve Cryptography (ECC) Primer* [online]. Disponible: http://www.certicom.com/images/pdfs/WP-ECCprimer.pdf

[31] Vanstone, S. *Cryptography thrown an Elliptic Curve*, 21 de Agosto de 2008.

[32] Pierre, J. *Galois Theory*, Graduate text in Mathematics, vol. 44, sec. 4.1.4. Springer-Verlag., 1991.

[33] R. Lidl y H. Niederreiter, *Introduction to finite fields and their applications*. Cambridge: Cambridge University Press, 1986, pp. 43-159.

[34] C. Paar, "Efficient VLSI Architectures for Bit-Parallel Computation in Galois Fields", Tesis doctoral, Instituto para Matemáticas Experimentales, Univ. of Essen, Essen, Alemania, 1994.

[35] I. F. Blake, X. Gao, R. C. Mullin, S. A. Vanstone y T. Yaghoobian, *Applications of Finite Fields*. Norwell: Kluwer Academic Publishers, 1993, pp. 1-15.

[36] R. C. Agarwal y C. S. Burrus, "Fast Digital Convolution Using Fermat Transforms", en Proc. 1973 IEEE Southwest Conf., pp 538-543.

[37] R. E. Blahut, *Fast Algorithms for Digital Signal Processing*. New York, Cambridge University Press, 2010, pp. 286-339.

[38] J. Guajardo, S. S. Kumar, C. Paar y J. Pelzl, “Efficient Software-Implementation of Finite Fields with Applications to Cryptography”, Act. App. Math.: An International Survey, vol. 93, no. 1-3, pp. 3-32, Septiembre 2006.

[39] ANSI, *Public Key Cryptography for the Financial Services Industry- The Elliptic Curve Digital Signature Algorithm (ECDSA)*, Draft, ANSI X9.62, 1998.

[40] D. Johnson, A. Menezes y S. Vanstone, “The Elliptic Curve Digital Signature Algorithm”, Tech. Rep. COOR 99-34, Depto. of C&O, Univ. de Waterloo, Agosto 1999.

[41] A. Reyhani-Masoleh y M. A. Hasan, "Fast Normal Basis Multiplication Using General Purpose Proccesors", IEEE Transactions on Computers, vol. 52, no. 11, pp. 1379-1390,

[42] S. Gao, "Normal Bases over Finite Fields", Tesis doctoral, Univ. de Waterloo, Ontario, Canada, 1993.

[43] S. Pastore, “Normal bases in finite fields and Efficient Compact Subgroup Trace Representation”, Tesis, Universita Degli Studi Roma Tree, Roma, Italia, 2007.

[44] A. Reyhani-Masoleh y M. A. Hasan, “Efficient Digit-Serial Normal Bases Multipliers over Binary Extension Fields”, ACM Transactions on Embedded Computing Systems, vol. 3, no. 3, pp. 575-592, Agosto 2004.

[45] J. Omura y J. Massey, “Computational method and apparatus for finite field arithmetic”, Patente U.S. 4 587 627, Mayo 1986.

[46] A. Reyhani-Masoleh y M. A. Hasan, “Efficient Multiplication Beyond Optimal Normal Bases”, IEEE Transactions on Computers, vol. 52, no. 4, pp. 428-439.

[47] B. Sunar y Ç. K. Koç, “An Efficient Optimal Normal Basis Type II Multiplier”, IEEE Transactions on Computers, vol. 50, no. 1, pp. 83-87.

[48] I. F. Blake, R. M. Roth y G. Seroussi, “Efficient Arithmetic in GF(2^n) through Palindromic Representation”, Hewlett Packard, Tech. Rep. [Online]. Disponible: http://www.hpl.hp.com/techreports/98/HPL-98-134.pdf

[49] J. H. Park, J. Y. Park y S. G. Hahn, “Elliptic curve point counting over finite fields with Gaussian normal basis”, Proceedings of the Japan Academy, Series A. vol. 79, no. 1, pp. 5-8.

[50] D. W. Ash, I. F. Blake y S. A. Vanstone, “Low complexity normal bases”, vol. 25, no. 3, pp. 191-210, Noviembre 1989.

[51] Z. Wang y S. Fan, “New Montgomery-based Semi-systolic Multiplier for Even-type GNB of GF(2^m)”, Cryptology ePrint Archive: Report 2010/218 [Online]. Disponible: http://eprint.iacr.org/2010/218.pdf

[52] R. Dahab, D. Hankerson, F. Hu, M. Long, J. López y A. Menezes, “Software Multiplication using Gaussian Normal Bases”, IEEE, Transactions on Computers, vol. 55, no. 8, pp. 974-984.

[53] Washington, L. *Elliptic Curves Number Theory and Cryptography*, CRC Press, University of Maryland, 2008.

[54] J. M. Pollard, "The Fast Fourier Transform in a Finite Field". Mathematics of Computation, vol. 25, no. 114, pp. 365-374, Abril 1971.

[55] S. Baktir y B. Sunar, “Finite Field Polynomial Multiplication in the Frequency Domain with Application to Elliptic Curve Cryptography”, Computer and Information Sciences-ISCIS 2006, vol.4263/2006, pp 991-1001.

[56] S. Baktir, “Frequency Domain Finite Field Arithmetic for Elliptic Curve Cryptography”, Tesis Doctoral, Worcester Polytechnic Institute, Worcester, MA, USA, Abril 2008.

[57] Kalach K. y David J. P., “Hardware Implementation of large number multiplication by FTT with modular reduction”, Proceedings of the 3er international IEEE-NEWCAS conference, pp. 267-270, Junio 2005.

[58] Maxinez, D. y Alcalá J., *VHDL, El arte de programar sistemas digitales*, ITESM, CEM, 2002.

[59] Xilinx, *Spartan-3 FPG Family: Complete Data Sheet,* Enero de 2005.

[60] Srinath, S., *"Automatic generation of high performance multipliers for FPGA with Asymmetric multiplier blocks"*, Tesis de Maestría, University of Wisconsin-Madison, 2010.

[61] Shieh, M., Chen, J., Ling, W. y Wu, C. *An efficient Multiplier/Divider Design for Elliptic Curve Cryptosystem over GF(2^m)*. en Journal of Information Science and Engineering, vol. 25, pp. 1555-1573. 2009.

[62] Kitsos, P., Thoedoridis, G. y Koufopavlou, O. *An efficient reconfigurable architecture for Galois field GF(2^m)*, en Microelectronics journal, vol. 34, pp. 975-980. 2003.

[63] Chavez, C. y Rodríguez. F. *Efficient Implementation of a 256 bit Modular Multiplier on a FPGA Device*. en 9no Coloquio Nacional de Teoría de Códigos, Criptografía y Áreas Relacionadas, Ciudad de México, 2011.

ANEXOS

Artículos Publicados en Memorias de Congreso Nacional

Comparativa de Técnicas de Representación para Campos Finitos

José Antonio Flores Escobar, Moisés Salinas Rosales, Gualberto Aguilar Torres, Gina Gallegos García

ROC&C 2010

Artículos Publicados en Memorias de Congreso Internacional

Finite Field Polynomial 16-bit Multiplier for Power Constrained Devices

José Antonio Flores Escobar, Moisés Salinas Rosales, José Velázquez López

CONIELECOMP 2012