Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica

Implementación y análisis de algoritmos de encriptación en un FPGA.

Por:

Alejandro León Torres

Ciudad Universitaria "Rodrigo Facio", Costa Rica

Noviembre de 2015

Implementación y análisis de algoritmos de encriptación en un FPGA.

Por:

Alejandro León Torres

IE-0499 Proyecto eléctrico

Aprobado por el Tribunal:

M.Sc. Diego Valverde Garro
Profesor guía

M.Sc. Carlos Duarte Martínez
Profesor lector

M.Sc. Enrique Coen Alfaro
Profesor lector

Dedicatoria

Dedico este proyecto a las siguientes personas:

- asdas

Reconocimientos

asdasd

Resumen

El proyecto busca investigar sobre la teoría de criptografía y como la misma es implementada en la computación para el resguardo de datos, específicamente en hardware haciendo uso de FPGAs y el lenguaje de descripción de hardware Verilog.

Como punto de partida se elegirán dos algoritmos de encriptación comúnmente empleados, para llevar a cabo un análisis comparativo de una serie de parámetros que son relevantes para su implementación en una plataforma de FPGA. Estos parámetros consideran tres de las mayores limitantes para implementación de algoritmos en FPGA, como son el consumo de celdas, la cantidad de memoria interna utilizada y finalmente el uso de celdas especializadas de aritmética o DSP.

Índice general

Índice de figuras

Índice de tablas

1 Introducción

1.1 Justificación

La encriptación de datos por seguridad es una necesidad en la actualidad. Por ejemplo cuando a un empleado le roban una computadora corporativa que contiene datos sensibles para la empresa en la que labora, la encriptación evita que se pueda acceder a los mismos y que así se protejan de terceros.

En el caso de la encriptación de software trae consigo problemáticas como lo son la necesidad de actualizaciones y el desempeño del computador. Este último aspecto se debe a que una encriptación a nivel de software debe dedicar recursos del sistema a encriptar/desencriptar información.

En el caso de hardware se tienen los beneficios de que es muy confiable, rápido y conveniente. A diferencia de la encriptación de software se tiene una parte de hardware dedicada al proceso de encriptación/descriptación y por tanto el desempeño de la computadora no se ve tan afectado, también al no haber forma de actualizar hardware ocurre una disminución en costos (ACA FALTA LA CITA). Otra de las grandes ventajas de la encriptación basada en hardware es la facilidad de configuración, ya que gran parte de esto proceso es eliminado debido a que el hardware se encarga directamente de esto. (CITA DE DRIVETRUST).

Debido a lo expuesto anteriormente se buscó trabajar en algoritmos de encriptación en hardware ya que es un área de trabajo que se puede explotar para investigar y mejorar.

Se escogió como plataforma de trabajo un FPGA

1.2 Alcances y limitaciones del proyecto

Se implementarán 2 algoritmos de encriptación de datos en un FPGA haciendo uso de Verilog como lenguaje de descripción de hardware.

Posteriomente y mediante las herramientas de síntesis de Xilinx se realizará un análisis de métricas críticas en el desarrollo de aplicaciones en FPGAs como los son la cantidad de compuertas o celdas, el consumo de memoria interna del FPGA así como la cantidad de bloques aritméticos o de DSP que son usados por cada algoritmo.

Inicialmente se va a llevar a cabo un análisis individual de cada algoritmo, haciendo uso de las métricas anteriormente descritas y variando parámetros

comunes de los algoritmos de encriptación como lo son el tamaño de la llave y la cantidad de rondas de encriptación (este último en algoritmos de tipo Feistel).

Como segunda parte del proyecto se va a realizar un análisis comparativo entre ambos algoritmos eligiendo parámetros fijos para ambos.

Los análisis individuales y comparativos anteriomente mencionados no abarcarán ningún tipo de criptoanálisis de algún algoritmo con respecto otro ni de cuál sería la escogencia de los parámetros ideal para realizar un análisis comparativo entre ambos. Sino que a partir del análisis individual realizado se va a efectuar una escogencia de los parámetros de ambos algoritmos para realizar su comparación implementando las métricas descritas.

Para la escogencia de estos dos algoritmos se realizará a partir de una identificación de las principales ramas del cifrado para así elegir los dos algoritmos de dos de estas ramas abarcando de esta manera un tema más amplio para el análisis y discusión.

Se limitará a realizar una escogencia de esta manera sin la necesidad de realizar un criptoanálisis de los algoritmos, y más bien se simplificará a buscar algoritmos que sean implementables, de manera relativamente sencilla, en un FPGA conociendo desde un principio las limitantes estáticas del hardware.

1.3 Objetivos

Objetivo General

Implementar dos algoritmos de encriptación en un FPGA y realizar un análisis comparativo de la implementación de ambos algoritmos, empleando una serie de parámetros previamente seleccionados.

Objetivos Específicos

1. Implementar dos algoritmos de encriptación comúnmente empleados en un FPGA utilizando el lenguaje de descripción de hardware Verilog.

2. Realizar un análisis individual para cada uno de los algoritmos individuales, variando alguno de sus parámetros (por ejemplo el tamaño de la llave) para comparar haciendo uso de métricas como la cantidad de compuertas, el consumo de memoria interna del FPGA así como la cantidad de bloques aritméticos o de DSP que se van a ir utilizando conforme se varíe el parámetro del algoritmo elegido.

3. Realizar un análisis comparativo de los dos algoritmos implementados, utilizando como métricas la cantidad de compuertas o celdas, el con-

sumo de memoria interna del FPGA así como la cantidad de bloques aritméticos o de DSP que son usados por cada algoritmo.

1.4 Metodología

La metodología que se siguió para la realización del proyecto es la siguiente:

1. Estudios bibliográficos de:
   - Criptografía: importancia y como la misma se implementa en la computación.
   - Algoritmos de cifrado: Identificación de ramas y subramas.
   - Código e implementación de algoritmos de encriptación en diferentes lenguajes de programación.
2. Escogencia de los algoritmos de encriptación a implementar.
3. Implementación de los algoritmos de cifrado en un FPGA Xilinx MODELO????.
4. Realización del análisis individual de cada uno de los algoritmos.
5. Realización del análisis comparativo entre ambos algoritmos.
6. Realización de las conclusiones y Recomendaciones.

1.5 Desarrollo

El presente informe se estructura para el lector de la siguiente manera:

1. Capítulo I: Introducción.
2. Capítulo II: Antecedentes y Marco Teórico.
3. Capítulo III: Implementación de los algoritmos de encriptación en el FPGA.
4. Capítulo IV: Resultados de los análisis individuales y comparativos de los 2 algoritmos implementados en el FPGA.
5. Capítulo V: Conclusiones y recomendaciones.

2 Marco Teórico

Este capítulo va a poner en contexto al lector con respecto a conceptos básicos de criptografía para posteriormente explicar los algoritmos que van a ser implementados en el FPGA.[1]

2.1 Conceptos Básicos

Según la Real Academia Española **?** la criptografía se define como

Arte de escribir con clave secreta o de un modo enigmático.

El mensaje que se desea transmitir es usualmente llamado *texto plano* o simplemente *mensaje*. Este mensaje pasa por un proceso donde se disfraza el texto plano en un *texto cifrado*, el proceso es llamado *cifrado*. El proceso inverso donde se toma un texto cifrado en un texto plano se denomina *descifrado*.

El texto plano o mensaje se denota usualmente por la letra M o P, el texto cifrado se denota usualmente por la letra C, la función o algoritmo que cifra se denota por E y la que descrifa se denota por D. Un algoritmo criptográfico corresponde a la función matemática para cifrar y descrifrar.
Se muestra en las Ecuaciones (2.1) y (2.2) las relaciones entre estas notaciones. Note como al aplicarle la función de cifrado al texto plano se obtiene el texto cifrado y como al aplicarle la función de descifrado al texto cifrado se obtiene el texto plano. Finalmente se debe cumplir la identidad que describe la Ecuación (2.3). **?**

$$E(M) = C \tag{2.1}$$

$$D(C) = M \tag{2.2}$$

$$D(E(M)) = M \tag{2.3}$$

La importancia de criptografía transciende más allá de brindar la confidencialidad en la comunicación la criptografía también cumple con la siguientes tareas:

- Autenticación: El receptor del mensaje debe de poder conocer y asegurar el emisor del mensaje, esto para que un tercero no pueda adjudicarse la identidad del emisor.

[1] Según el ISO 7498-2 los términos correctos para encriptar y desencriptar son cifrar y descifrar respectivamente.

- Integridad: El receptor tiene que poder asegurarse que el mensaje no fue cambiado en el transito del mismo. Esto para que un tercero no pueda cambiar el contenido enviado por el emisor sin que el receptor lo sepa.
- *Non-repudiation*: El emisor del mensaje no puede negar que el mensaje fue enviado por él.

2.2 Sistema criptográfico (*criptosistema*)

Cuando la seguridad del algoritmo se basa en como procede el algoritmo, se denomina *algoritmo restringido*. Este tipo de algoritmos son poco utilizados en la actualidad debido al gran problema que presentan. Tomemos de ejemplo que un grupo de usuarios decide utilizar un algoritmo de cifrado restringido para sus comunicaciones, se tendrá una comunicación segura hasta que alguno de los miembros decida salirse del grupo, ya que el usuario al no pertenecer más al grupo, no le importa mantener en secreto el algoritmo y puede distribuirlo para que terceros intercepen las comunicaciones. Así cada vez que un miembro deja el grupo, el grupo debe proceder a cambiarse a todo un nuevo algoritmo lo cual puede tornarse una labor complicada.

En cambio, la criptografía moderna **?** agrega el concepto de *llave* donde se tiene un algoritmo el cual toma como parámetro de entrada una llave y el texto plano o texto cifrado y cifra o descifra el mismo de forma correcta únicamente si se tiene la llave correcta. Retomando el ejemplo anterior, el grupo solamente necesitaría cambiar de llave cuando un miembro se va, facilitando el uso del cifrado y manteniendo las comunicaciones secretas.

Este concepto anterior viene a definir lo que actualmente se conoce como sistema criptográfico o *criptosistema*. Según **?** un criptosistema cuenta con 5 componentes:

- Un espacio de textos planos o mensajes (M)
- Un espacio de textos cifrados (C)
- Un espacio de llaves (k)
- Una familia de *transformaciones de cifrado*: $E_K : M \to C$ donde $K \epsilon k$
- Una familia de *transformaciones de descrifrado*: $C_K : C \to M$ donde $K \epsilon k$

Se entiende como *espacio* el conjunto de posibles valores para la variable dada, sea esta M, C o K. Y una familia de transformaciones corresponde a todos los posibles mapeos que se pueden realizar de un espacio a otro (de M a C o viceversa) con todos los valores contenidos en el espacio k.

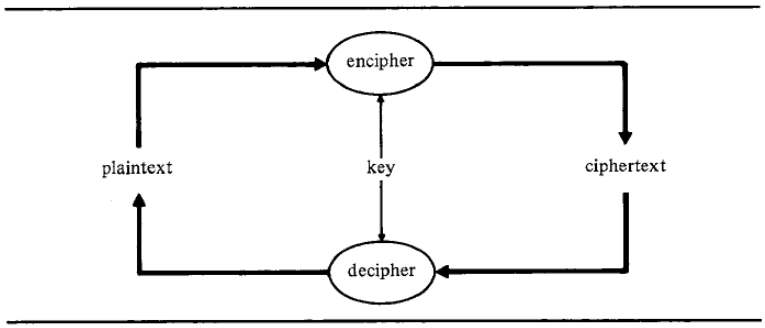

Figura 2.1: Descripción gráfica de un sistema criptográfico.

En la actualidad se trabaja la criptografía sobre computadoras, es decir, cifrando y descrifrando bits, los cuales pueden tener diferentes significados, ya sea una imagen, un texto, un programa, etc. Esto significa que en la actualidad no se trabaja sobre caracteres del alfabeto o símbolos, sino más bien sobre 1's y 0's. Esto toma relevancia ya que al tener solo dos símbolos para cifrar, los algoritmos se vuelven más complejos por la falta de alternativas para sustituir un símbolo por otro.

Así antiguamente la criptografía se basaba en caracteres que eran sustituidos o traspuestos por otros caracteres. Esto corresponde a cifrados de sustitución y de transposición, los cuales continúan siendo la base de la criptografía pero basado los 2 símbolos del sistema binario.

Como se explicó anteriormente este tipo de cifrado se basa en tomar un caracter del texto plano y sustituirlo por otro caracter. Para descrifrar el texto cifrado simplemente se sustituyen de vuelta los caracteres y listo.

Según **?** en la criptografía clásica existen 4 tipos de cifrado por sustitución:

- Cifrado de sustitución simple: Una sustitución de uno a uno entre cada caracter del texto plano y el texto cifrado. Ejemplos de este tipo de sustitución son el famoso cifrado de César y el ROT13 utilizado en UNIX.

- Cifrado de sustitución homofónico: Una sustición de uno a muchos. Un caracter del texto plano, por ejemplo A, puede ser sustituido por varios caracteres en el texto crifado, por ejemplo "5", "13", "43". Observe la Figura 2.2 donde se presenta una serie de posibles asignaciones de números a las letras del mensaje PLAIN PILOT y un posible texto cifrado haciendo uso de este tipo de cifrado.

- Cifrado de sustitución de poligrama: Una sustitución por bloques en donde se toma un bloque de caracteres del texto plano y se sustituye por su bloque equivalente en el texto cifrado. Por ejemplo si en el texto plano se tiene "ABC" se sustituye por "SLL" en el texto cifrado.

Letter	Homophones
A	17 19 34 41 56 60 67 83
I	08 22 53 65 88 90
L	03 44 76
N	02 09 15 27 32 40 59
O	01 11 23 28 42 54 70 80
P	33 91
T	05 10 20 29 45 58 64 78 99

One possible encipherment of the message is:

M =	P	L	A	I	N	P	I	L	O	T
C =	91	44	56	65	59	33	08	76	28	78

Figura 2.2: Ejemplo de cifrado homofónico.

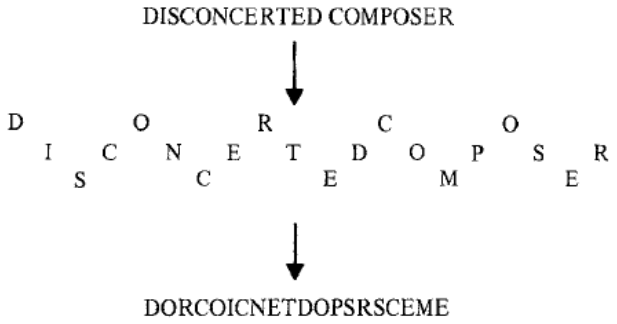

Figura 2.3: Ejemplo de cifrado de transposición.

- Cifrado de sustitución polialfabético: ?????

La otra variedad de algoritmos son los de tranposición, en este tipo de algoritmos de cifrado el texto plano se convierte en texto cifrado cuando el orden de los caracteres es cambiado bajo alguna norma. Un ejemplo moderno de este tipo de algoritmos es el "rail-fence"donde el texto plano se reacomoda con la forma de una cerca como se observa en la Figura 2.3. En este caso la llave del algoritmos sería la profundidad de la cerca, para efectos de este ejemplo es de 3.

Actualmente en los algoritmos que se implementan en computadoras se combina tanto la tranposición como la sustitución. Por ejemplo se tiene el algoritmo RC5, el cual se desarrollará más adelante en este proyecto en donde se utiliza corrimientos o rotaciones a bits (tranposición) y sumas o XOR's (sustituciones) para cifrar el texto plano. Los algoritmos que se implementan en la criptografía se dividen en 2 categorías principales: simétricos y de llave

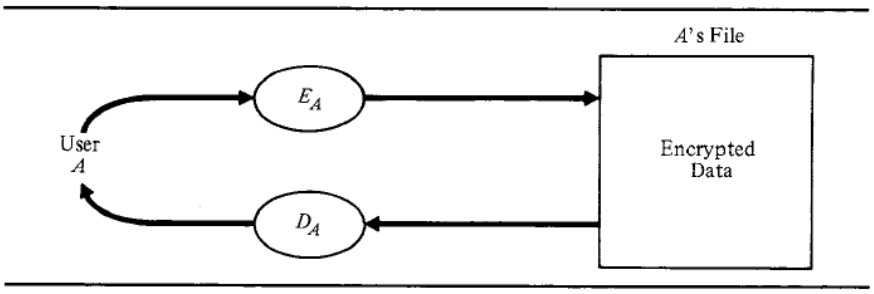

Figura 2.4: Descripción gráfica de una algoritmo de llave simétrica.

pública (también llamados asimétricos **?**).

2.3 Algoritmos de cifrado

Algoritmos simétricos

? da una muy buena analogía para explicar el concepto de un algoritmo simétrico. Piense en el algoritmo como una caja fuerte. La combinación de la caja fuerte vendría a ser la *llave* del algoritmo. Note como en una caja fuerte cualquier persona con la combinación puede llegar, abrir la caja y poner o sacar documentos de la misma. En el caso de no conocer la combinación, se debe proceder a forzar la caja o probando todas las combinaciones posibles hasta hallar la correcta. Es decir en un algoritmo simétrico se cuenta con una llave única que funciona tanto para cifrar como para descrifrar los mensajes como se muestra en la Figura 2.4. La notación para el cifrado y descrifado en estos algoritmos se muestra en las Ecuaciones (2.4) y (2.5).

$$E_K(M) = C \tag{2.4}$$

$$D_K(C) = M \tag{2.5}$$

Los algoritmos simétricos se dividen en 2 categorías (**?**):

- Cifrado de bloque: Se cifra en bloques de bits ya sea bytes, words, etc. Es decir cuando se va a proceder a cifrar un texto plano, se segmenta el texto en grupos de bits y estos son cifrados de manera independiente. Se puede tomar como ejemplo el algoritmo *Data Encryption Standart* (DES) el cual cifra sobre bloques de 64 bits. Ejemplos de estos algoritmos pueden ser:
  - Data Encryption Standart (DES).

  - Lucifer.
  - LOKI.
  - 3-way.
  - RC5.
  - GOST.
  - IDEA.

- Cifrado de *Stream*: Es cuando se trabaja sobre un bit únicamente. Estos no son muy usuales en la actualidad ya que al trabajar en lenguaje binario solo se cuenta con 2 símbolos y si se cifra únicamente un bit no existen muchas posibilidades para sustituir o transposicionar. Ejemplos de estos algoritmos pueden ser:

  - RC4.
  - SEAL.
  - WAKE.

Según (**?**) los criptosistemas simétricos en la red afrontan los siguientes problemas

- Distribución de la llave: La llave se debe mantener en secreto. Esto en la actualidad en una tarea demasiado díficil de lograr porque la llave debe ser conocida para el cifrado y descifrado (emisores y receptores) entonces para establecer una comunicación segura el primer paso debe ser entregar la llave de forma segura, lo cual en una red de computadoras se puede tornar una tarea prácticamente imposible de realizar. La única solución sería entregar las llaves mediantes un servicio de *courier* o similares e igualmente se corren riesgos.

- Compromiso de seguridad: Si la llave es conocida por un tercero, si este intercepta el tráfico de información, todas las comunicaciones serán descifradas fácilmente.

- Comunicaciones aisladas: En el caso de que cada usuario de una red se desee comunicar secretamente por separado con todos los otros usuarios del misma haciendo uso del mismo criptosistema, se debe utilizar una llave diferente para cada comunicación. Así para una red de N usuarios se requieren $N(N-1)/2$ llaves. A primera vista esto no parece tan importante pero por ejemplo para 10 usuarios, se necesitan 45 llaves lo cual está bien, pero para 100 usuarios se necesitarían 4950 llaves.

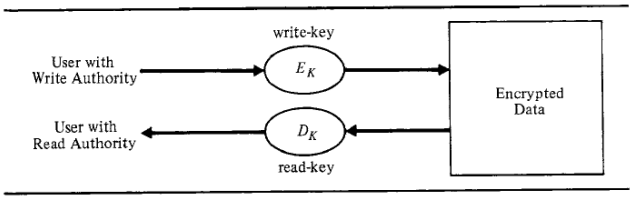

Figura 2.5: Descripción gráfica de una algoritmo de llave pública.

Algoritmos de llave pública

Nuevamente **?** nos ofrece una excelente analogía para explicar en este caso los algoritmos de llave pública. Tenemos un *mailbox*, donde cualquier persona puede poner un mensaje adentro pero ÚNICAMENTE el dueño puede abrirlo para sacar y leer los mensajes.

En los algoritmos de llave pública los emisores hacen uso de una llave para cifrar los mensajes que desean enviar pero estos mensajes pueden ser descrifrados únicamente si se tiene la llave para descifrar mensajes que es diferente de la llave para cifrar, la cual la tendrá el receptor bien resguardada. En la Figura 2.5 se muestra el diagrama básico de un algoritmo de llave pública. Ejemplos de estos algoritmos pueden ser:

- RSA
- Pohlig-Hellman
- Rabin
- ElGamal

Estos algoritmos sientan su base matemática en funciones llamadas *one-way* en donde al aplicarle una función a una variable, la variable no puede retornar a su valor original de ninguna manera, es decir la función no tiene una inversa.

Según **?** existen funciones *one-way* con una "puerta trasera" donde se puede retornar a la variable original, este tipo de funciones son las que se implementan en algoritmos de llave pública. para lograr este objetivo los algoritmos dominantes de esta rama se basan en la dificultad de factorizar números grandes que son el resultado de multiplicar dos números primos grandes como también se basan en el *Discrete Logarithm Problem*.

En estos algoritmos no es posible a partir de la llave para cifrar obtener la llave para descrifrar. Esto permite que la llave para cifrar se pueda hacer pública, por lo cual recibe el nombre de llave pública y la llave para descrifrar

se denomina llave privada. La notación para estos algoritmos corresponde a la de las Ecuaciones (2.6) y (2.7)

$$E_{K_x}(M) = C \tag{2.6}$$

$$D_{K_y}(C) = M \tag{2.7}$$

El objetivo de utilizar cifrado de llave pública se basa en:

- Receptor y emisor acuerdan un sistema de cifrado.
- El receptor entrega al emisor su llave pública.
- El emisor cifra el texto plano haciendo uso de la llave pública entregada y el sistema acordado.
- El emisor envía el texto cifrado.
- El receptor descrifa el texto cifrado haciendo uso de la llave privada.

De esta manera no hay forma que fisgones logren descrifrar el mensaje aunque obtengan la llave pública, así el receptor se asegura que las comunicaciones van a ser mucho más seguras ya que el emisor deja de tener la llave para descifrar y no puede brindarse a nadie o que sea robada a este. Para la implementación de un criptsistema que hace uso de algoritmo de llave pública, lo que se hace es que en la red se tiene una base de datos donde se registra el usuario y su respectiva llave pública, así cuando un usuario se desea comunicar con otro, va a la base de datos, busca al usuario y su llave pública, cifra el mensaje con la misma y se la envía. Esto solventa los problemas que presentaba anteriormente los algoritmos simétricos, ya que no es necesario transmitir de forma secreta llaves para poder realizar comunicaciones y para que diferentes usuarios se comuniquen de forma secreta entre si no es necesario el uso de diferentes llaves por cada enlace de comunicación.

Criptosistemas híbridos

Estos son grandes beneficios pero se pasan a un precio muy caro: tiempo de procesamiento. Según **?** el tiempo de procesamiento del RSA con respecto al del DES es de alrededor de 1000 mil veces más lento.

En un mundo donde la velocidad es una clave fundamental en las comunicaciones se propuso la siguiente solución. Ya que los algoritmos simétricos tienen la debilidad de comunicar la llave antes de comenzar la comunicación pero son mucho más rápidos, y que los algoritmos de llave pública ostentan una mejor sistema para comunicarse pero son muy lentos, se decidió utilizar ambos. La llave del algoritmo simétrico es encriptada con una algoritmo de

llave pública para transmistirse en la red sin comprometerla y posterior a esto se realizan las comunicaciones con el algoritmo simétrico para obtener una mejor velocidad de comunicación. Se puede ver el protocolo de la siguiente manera (**?**):

- El receptor envía su llave pública al emisor.
- El emisor genera una llave de sesión[2] la cifra y se la envía al receptor usando la llave pública que le fue dada.
- El receptor descrifra la llave de sesión usando su llave privada.
- Ahora ambos puede comunicarse de forma secreta con la llave de sesión con un criptosistema simétrico.

2.4 Seguridad en algoritmos de cifrado

Tipos de ataques

Public-key cryptosystems are vulnerable to chosen-plaintext attacks. If C = E(P), when P is one plaintext out of a set of n possible plaintexts, then a cryptanalyst only has to encrypt all n possible plaintexts and compare the results with C (remember, the encryption key is public). He won't be able to recover the decryption key this way, but he will be able to determine P.

Tamaño de la llave

Como se mencionó anteriormente la seguridad de un buen algoritmo depende del tamaño de su llave.

Asumiendo que se tiene una seguridad del algoritmo perfecta, la única forma de quebrar el criptosistema sería mediante un ataque de fuerza bruta el cual es un tipo de ataque *known-plaintext* donde sería solamente necesario unos 64 bits de texto plano y texto cifrado para ejecutarlo. Ahora tomemos en consideración el tamaño de la llave, si se tuviera una llave de 2^8 bits el atacante solamente tendría que probar 256 posibilidades para obtener la llave lo que le tomaría a un atacante unos cuantos segundos en una computadora, en cambio para 2^{64} bits tenemos $1{,}8446744^{19}$ posibles llaves que el atacante debe probar lo cual tomaría con los recursos computacionales de una supercomputadora alrededor de 585,000 años. Y para una longitud de llave de 2048, con un billón

[2] A common cryptographic technique is to encrypt each individual conversation with a separate key. This is called a session key, because it is used for only one particular communications session. As discussed in Section 8.5, session keys are useful because they only exist for the duration of the communication. How this common session key gets into the hands of the conversants can be a complicated matter.

Symmetric Key Length	Public-key Key Length
56 bits	384 bits
64 bits	512 bits
80 bits	768 bits
112 bits	1792 bits
128 bits	2304 bits

Figura 2.6: Comparación del tamaño de llaves en algoritmos simétricos y asimétricos.

de intentos por segundo en computadoras en paralelo se necesitarían 10597 años para encontrar la llave.

Comparar el nivel de seguridad antes un ataque de fuerza bruta de un algoritmo simétrico y uno de llave pública con llaves del mismo tamaño no es posible. La Figura 2.6 muestra una tabla con equivalentes realizados empíricamente por **?** sobre tamaños de llaves que dan un nivel de seguridad equivalente para los 2 tipso de algoritmos basado en los algoritmos más populares de cada rama.

Manejo de las llaves

Se puede tener un algoritmo extremadamente robusto, veloz y con un tamaño de llave perfecto, pero si un atacante puede obtener la llave, ya sea chantajeando, interceptando, extorsionando o pagando por la misma se pierde todo. De ahí la gran importancia de como manejar las llaves.

Al ser esto tan importante existen ciertas maneras de generar llaves:

- Ingresada por el usuario: El usuario elige una llave y esa es la que se va a utilizar. Este método es sumamente inseguro debido a que gran

cantidad de contraseñas se repiten, o son datos personales del usuario como su número de celular o similares. Por tanto se pueden realizar *ataques de diccionario* donde las primeras llaves que se prueban son las mencionadas anteriormente.

- Random keys: Es un muy buen método para generar llaves, consiste en utilizar un programa que genere la llave, es ventajoso ya que es robusto ante ataques de diccionario pero presenta el problema que la llave es difícil de recordar y posiblemente se olvide.

- pass phrases: Es una combinación de ambos, el usuario escribe una contraseña fácil de recordar y después hace uso de un one-way has function para convertir esta llave en una llave random de tamaño arbitrario.

2.5 Escogencia de los algoritmos a implementar

En un principio el objetivo de este proyecto era realizar una comparación con las métricas descritas en la Introducción sobre un algoritmo simétrico y un algoritmo de llave pública, posterior a la realización del marco teórico se llegó a la conclusión de que realizar este análisis no iba a ser para nada justo debido mayoritariamente a tres razones:

- El tiempo de procesamiento: Como se menciona anteriormente los algoritmos de llave pública consumen mucho más tiempo realizando el cifrado y descifrado que un algoritmo de llave simétrica.

- La formas en las que estos algoritmos son diseñados producen falencias en seguridad distintas por lo cual analizar comparativamente no sería tan interesante.

- Los algoritmos realizan el proceso de cifrado de maneras muy diferentes: En el caso de los algoritmos simétricos son basados en transposición y sustitución que trabajan en su mayoría con XOR's, sumas y corrimientos. En cambio los algoritmos de llave pública trabajan basados en la multiplicación de números primos y *Discrete Logarithm Problem* donde ocurren muchas multiplicaciones y sumas. Esto produce un consumo de recursos muy diferente entre ambos tipos.

Bajo el criterio que ambos algoritmos tuvieran características similares para poder ser analizados comparativamente y que fueran fáciles de implementar a nivel de hardware se optó por comparar dos algoritmos simétricos, específicamente el RC5 y el IDEA los cuales están basados en redes de Feistel[3] y

[3]Feistel Networks Most block algorithms are Feistel networks. This idea dates from the early 1970s [552,553]. Take a block of length n and divide it into two halves of length n/2: L

por tanto comparten características de diseño para que la comparación sea un poco más justa.

2.6 Descripción de los algoritmos a implementar

RC5

IDEA o LUCIFER o FEAL

2.7 Características de un FPGA

and R. Of course, n must be even. You can define an iterated block cipher where the output of the ith round is determined from the output of the previous round: Li = Ri - 1 Ri = Li - 1 f(Ri - 1,Ki) Ki is the subkey used in the ith round and f is an arbitrary round function. You've seen this concept in DES, Lucifer, FEAL, Khufu, Khafre, LOKI, GOST, CAST, Blowfish, and others. Why is it such a big deal? The function is guaranteed to be reversible. Because XOR is used to combine the left half with the output of the round function, it is necessarily true that Li - 1 f(Ri - 1,Ki) f(Ri - 1,Ki) = Li - 1 A cipher that uses this construction is guaranteed to be invertible as long as the inputs to f in each round can be reconstructed. It doesn't matter what f is; f need not be invertible. We can design f to be as complicated as we please, and we don't have to implement two different algorithms—one for encryption and another for decryption. The structure of a Feistel network takes care of all this automatically.