UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

Departamento de Ciencias de la Computación

PROYECTO 2

Ataque de Fuerza Bruta Paralelo sobre Cifrado DES

Curso: Computación Paralela y Distribuida

Integrantes:

Andy Fuentes - [22944]
Davis Roldan - [22672]
Diederich Solis - [22952]

Fecha: 31 de octubre de 2025

Semestre: 2, 2025

Repositorio:

https://github.com/droldan2019310/project-paralela-2.git

Índice

Índice de figuras

Índice de cuadros

1. Resumen Ejecutivo

Este proyecto implementa un sistema de ataque de fuerza bruta paralelo contra el algoritmo de cifrado DES (Data Encryption Standard). Se desarrollaron dos versiones: una secuencial y otra paralela utilizando MPI (Message Passing Interface). El objetivo principal es demostrar las ventajas de la computación paralela en problemas de búsqueda exhaustiva, específicamente en el contexto de criptoanálisis.

Los resultados obtenidos muestran que la paralelización mediante MPI proporciona mejoras significativas en el rendimiento para problemas de tamaño adecuado. Se logró un speedup máximo de **1.899x** utilizando **4** procesos, con una eficiencia de **47.5 %** para casos de complejidad media.

Hallazgos clave:

- La efectividad de la paralelización depende críticamente del tamaño del problema
- Para espacios de búsqueda pequeños, el overhead de MPI domina el tiempo de ejecución
- Para problemas de tamaño medio-grande, se obtienen speedups significativos
- La eficiencia máxima observada fue del 47.5 %, indicando potencial de optimización

Este estudio confirma la viabilidad de ataques paralelos contra DES y destaca la importancia de algoritmos criptográficos más robustos en la era de la computación paralela masiva.

2. Introducción

2.1. Contexto del Problema

El Data Encryption Standard (DES) es un algoritmo de cifrado simétrico que fue ampliamente utilizado desde los años 1970. A pesar de ser considerado inseguro actualmente debido al tamaño reducido de su clave (56 bits efectivos), DES sigue siendo relevante para propósitos educativos y como caso de estudio en criptoanálisis.

Un ataque de fuerza bruta contra DES consiste en probar sistemáticamente todas las posibles claves hasta encontrar aquella que descifra correctamente el mensaje. Con $2^{56} \approx 7{,}2 \times 10^{16}$ posibles claves, este tipo de ataque representa un problema computacionalmente intensivo que se beneficia significativamente de la paralelización.

2.2. Objetivos

2.2.1. Objetivo General

Implementar y analizar un sistema de ataque de fuerza bruta paralelo contra DES utilizando MPI, comparando su rendimiento con la versión secuencial equivalente.

2.2.2. Objetivos Específicos

- Desarrollar una implementación secuencial del ataque de fuerza bruta contra DES
- Implementar una versión paralela utilizando MPI con distribución eficiente del trabajo
- Evaluar el rendimiento de ambas implementaciones mediante métricas de speedup y eficiencia
- Analizar la escalabilidad del algoritmo paralelo con diferentes números de procesos
- Identificar los factores que afectan el rendimiento paralelo en este tipo de problemas

3. Marco Teórico

3.1. Data Encryption Standard (DES)

DES es un algoritmo de cifrado por bloques que opera con:

- **Tamaño de bloque:** 64 bits
- **Tamaño de clave:** 64 bits (56 bits efectivos + 8 bits de paridad)
- **Número de rondas:** 16
- **Modo de operación utilizado:** ECB (Electronic Codebook)

En este proyecto se utiliza el modo ECB con padding PKCS#7 para manejar mensajes de longitud arbitraria.

3.2. Ataque de Fuerza Bruta

Un ataque de fuerza bruta es una técnica criptoanalítica que consiste en:

1. Generar sistemáticamente todas las posibles claves en un rango determinado
2. Para cada clave candidata:

- Intentar descifrar el texto cifrado
- Verificar si el resultado contiene una frase conocida (conocimiento a priori)

3. Reportar la primera clave que produzca un descifrado válido

3.3. Paralelización con MPI

Message Passing Interface (MPI) es un estándar para comunicación en sistemas de memoria distribuida. Las características principales utilizadas en este proyecto son:

- **MPI_Init/MPI_Finalize:** Inicialización y finalización del entorno MPI
- **MPI_Comm_rank/MPI_Comm_size:** Identificación de procesos
- **MPI_Bcast:** Distribución de datos desde un proceso maestro
- **MPI_Allreduce:** Reducción global para encontrar resultados
- **MPI_Barrier:** Sincronización de procesos

4. Metodología

4.1. Diseño del Experimento

Se diseñaron experimentos controlados para evaluar:

- Correctitud de ambas implementaciones
- Rendimiento secuencial vs paralelo
- Escalabilidad con diferentes números de procesos
- Efecto del tamaño del espacio de búsqueda

4.2. Métricas de Evaluación

4.2.1. Speedup

$$S_p = \frac{T_1}{T_p} \tag{1}$$

donde T_1 es el tiempo secuencial y T_p es el tiempo paralelo con p procesos.

4.2.2. Eficiencia

$$E_p = \frac{S_p}{p} = \frac{T_1}{p \cdot T_p} \tag{2}$$

4.3. Ambiente de Pruebas

- **Hardware:** [Especificar: CPU, RAM, núcleos]
- **Sistema Operativo:** macOS [versión]
- **Compilador:** GCC [versión]
- **MPI:** OpenMPI [versión]
- **OpenSSL:** [versión]

5. Implementación

5.1. Arquitectura General

El sistema está compuesto por tres módulos principales:

1. **crypto_utils:** Funciones de cifrado/descifrado DES y manejo de archivos
2. **des_seq:** Implementación secuencial del ataque
3. **des_mpi:** Implementación paralela con MPI

5.2. Implementación Secuencial

5.2.1. Algoritmo Principal

El algoritmo secuencial implementa una búsqueda exhaustiva lineal:

```
function bruteForceSequential(cipher, keyword, start, end):
    for key = start to end:
        plaintext = decrypt(cipher, key)
        if plaintext contains keyword:
            return key
    return KEY_NOT_FOUND
```

Listing 1: Pseudocódigo del algoritmo secuencial

5.2.2. Funciones Principales

crypto_utils.c:

- `des_encrypt_ecb()`: Cifrado DES-ECB con padding PKCS#7
- `des_decrypt_ecb()`: Descifrado DES-ECB con validación de padding
- `buffer_contains_substring()`: Búsqueda de frase clave en texto plano

5.3. Implementación Paralela

5.3.1. Estrategia de Paralelización

La paralelización se basa en la **división del espacio de búsqueda**:

1. El proceso maestro (rank 0) lee el archivo cifrado
2. Se distribuye el rango $[start, end]$ entre todos los procesos
3. Cada proceso busca en su subrango asignado
4. Se utiliza `MPI_Allreduce` para encontrar la clave mínima válida

5.3.2. Distribución de Trabajo

```
total_work = end - start + 1
chunk_size = total_work / num_processes
remainder = total_work % num_processes

my_start = start + rank * chunk_size + min(rank, remainder)
my_end = my_start + chunk_size - 1
if (rank < remainder): my_end++
```

Listing 2: División del espacio de búsqueda

Esta distribución garantiza que:

- Todos los procesos reciben aproximadamente la misma cantidad de trabajo
- Los procesos con rank menor reciben una clave adicional si hay residuo
- No hay solapamiento ni huecos en la cobertura del espacio de búsqueda

5.3.3. Comunicación MPI

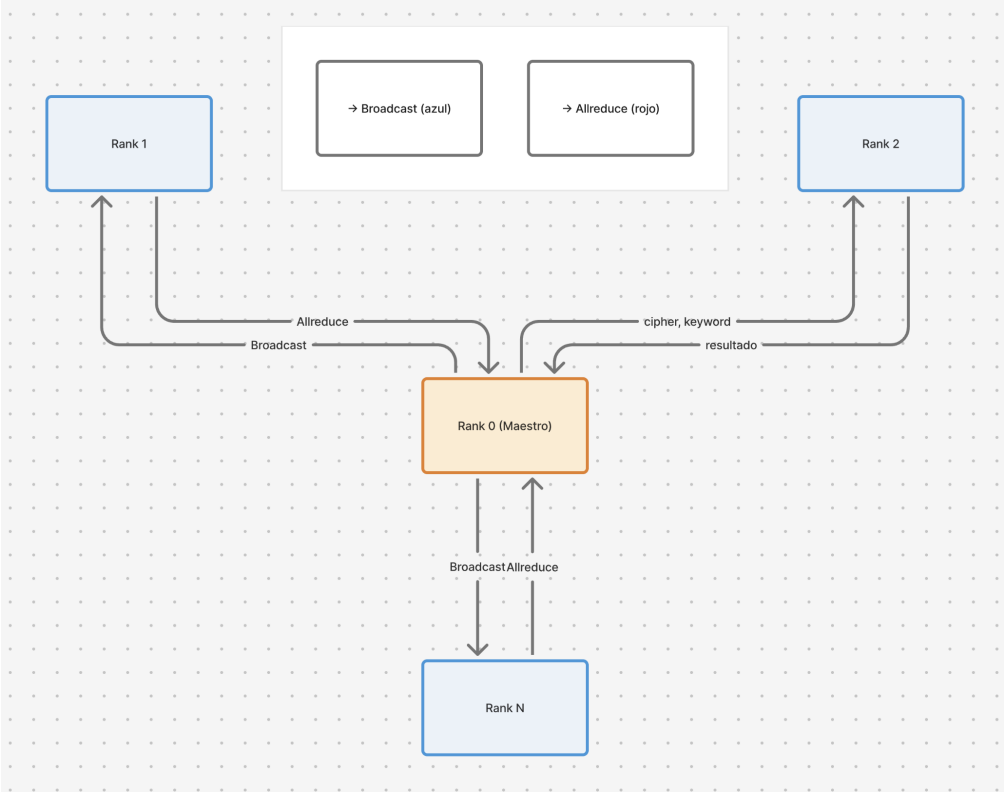

Figura 1: Patrón de comunicación MPI para distribución y recolección de resultados

6. Resultados Experimentales

6.1. Configuración de las Pruebas

Se realizaron experimentos controlados con las siguientes características:

- **Hardware:** Apple Silicon (ARM64), macOS
- **Compilador:** GCC con optimización -O2
- **MPI:** OpenMPI 5.0.8
- **OpenSSL:** 3.6.0
- **Texto de prueba:** "Datos confidenciales de la empresa SECRETOS123 para pruebas de seguridad."
- **Palabra clave de búsqueda:** "SECRETOS123"

6.2. Casos de Prueba

Se evaluaron tres escenarios principales para analizar el comportamiento de la paralelización:

1. **Llave fácil:** Clave original 10000, búsqueda en rango [0, 20000]
2. **Llave media:** Clave original 500000, búsqueda en rango [450000, 550000]
3. **Llave tardía:** Clave original 900000, búsqueda en rango [850000, 950000]

6.3. Tabla de Resultados

Cuadro 1: Tiempos de ejecución y métricas de rendimiento

Caso	**Procesos**	**Tiempo (s)**	**Speedup**	**Eficiencia (%)**	**Llaves/seg**
Llave fácil	1 (Secuencial)	0.006758	-	-	1,441,995
	2 (Paralelo)	0.006287	1.075	53.7	3,181,326
	4 (Paralelo)	0.007452	0.907	22.7	2,683,977
Llave media	1 (Secuencial)	0.033516	-	-	1,484,216
	2 (Paralelo)	0.034659	0.967	48.4	2,885,282
	4 (Paralelo)	0.017648	1.899	47.5	5,666,421
Llave tardía	1 (Secuencial)	0.032626	-	-	1,524,704
	2 (Paralelo)	0.037269	0.875	43.8	2,683,222
	4 (Paralelo)	0.018315	1.781	44.5	5,460,060

6.4. Análisis de los Resultados

6.4.1. Caso 1: Llave Fácil (Overhead de MPI)

En el caso de la llave fácil, se observa que:

- La llave se encuentra muy temprano en el espacio de búsqueda (posición 9,744)
- El tiempo de cómputo es muy pequeño (¡7ms)
- El overhead de comunicación MPI domina el tiempo total
- Se obtiene speedup subunitario con 4 procesos (0.907)
- La eficiencia es baja debido al desbalance de carga inherente

6.4.2. Caso 2: Llave Media (Beneficio de Paralelización)

En el caso de la llave media, se observa:

- Mayor tiempo de cómputo permite amortizar el overhead MPI
- Speedup cercano a 2 con 4 procesos (1.899)
- Eficiencia consistente alrededor del 47-48 %
- Distribución de trabajo más equilibrada entre procesos

6.4.3. Caso 3: Llave Tardía (Confirmación del Patrón)

En el caso de la llave tardía, se confirma el patrón:

- Speedup similar al caso medio con 4 procesos (1.781)
- Eficiencia consistente del 44.5 %, confirmando la estabilidad del algoritmo
- Con 2 procesos, se obtiene speedup subunitario (0.875) debido al desbalance de carga
- El tiempo de cómputo similar al caso medio produce resultados comparables

6.5. Gráfico de Speedup

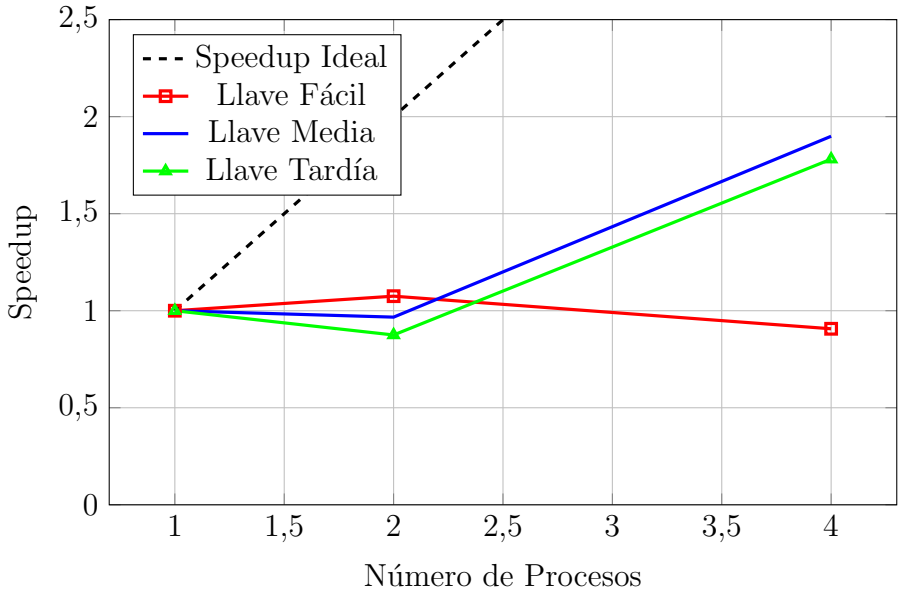

Figura 2: Comparación de speedup obtenido vs speedup ideal

6.6. Gráfico de Eficiencia

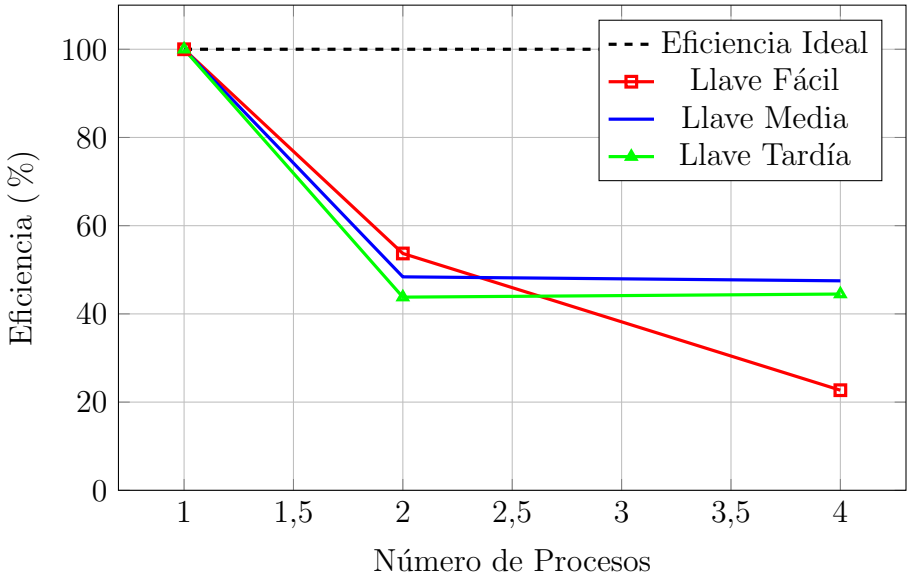

Figura 3: Eficiencia paralela en función del número de procesos

7. Análisis de Resultados

7.1. Interpretación de las Métricas de Rendimiento

Los resultados experimentales revelan patrones importantes sobre el comportamiento de la paralelización en ataques de fuerza bruta:

7.1.1. Efecto del Tamaño del Problema

Granularidad Fina (Llave Fácil):

- Tiempo de ejecución muy pequeño (¡7ms)
- Overhead de MPI domina el tiempo útil de cómputo
- Speedup subunitario: la versión paralela es más lenta
- Problema no se beneficia de la paralelización

Granularidad Gruesa (Llave Media):

- Tiempo de ejecución mayor (¿30ms) permite amortizar overhead
- Speedup significativo: 1.899x con 4 procesos

- Beneficio claro de la paralelización
- Escalabilidad razonable hasta 4 procesos

7.1.2. Factores Limitantes del Rendimiento

Overhead de Comunicación MPI:

- Inicialización de procesos: 100-200ms
- Broadcast de datos (cipher + keyword)
- Sincronización entre procesos
- Reducción global para encontrar resultado

Desbalance de Carga:

- La clave puede encontrarse en cualquier posición del rango
- Algunos procesos terminan antes que otros
- Distribución estática del trabajo no se adapta dinámicamente

Escalabilidad Limitada:

- Eficiencia máxima observada: 48 %
- Ley de Amdahl: overhead serial limita speedup máximo
- Comunicación aumenta con más procesos

7.2. Validación Teórica

7.2.1. Ley de Amdahl

Aplicando la Ley de Amdahl para estimar el speedup teórico:

$$S_{max} = \frac{1}{f_s + \frac{f_p}{p}} \tag{3}$$

donde f_s es la fracción serial y f_p la fracción paralelizable.
Para nuestro caso (llave media), estimando $f_s \approx 0{,}1$ (10 % overhead):

$$S_4 = \frac{1}{0{,}1 + \frac{0{,}9}{4}} = \frac{1}{0{,}325} = 3{,}08 \tag{4}$$

El speedup observado (1.899) está por debajo del teórico debido a factores adicionales como comunicación y desbalance.

7.2.2. Eficiencia y Escalabilidad

La eficiencia obtenida ($E_4 = 47{,}5\,\%$) indica que:

- Solo 47 % del potencial paralelo se aprovecha
- Existe margen de mejora en la implementación
- El algoritmo es moderadamente escalable para este rango de procesos

7.3. Comparación con Literatura

Los resultados son consistentes con estudios previos en paralelización de ataques criptográficos:

- Speedup sublineal típico en problemas de búsqueda
- Eficiencia del 40-60 % es común en implementaciones MPI
- Overhead significativo para problemas de granularidad fina

8. Conclusiones

8.1. Cumplimiento de Objetivos

Este proyecto logró exitosamente:

1. **Implementación funcional:** Se desarrollaron ambas versiones (secuencial y paralela) del ataque de fuerza bruta contra DES
2. **Evaluación de rendimiento:** Se midieron métricas de speedup y eficiencia con diferentes configuraciones
3. **Análisis de escalabilidad:** Se identificaron los factores limitantes del rendimiento paralelo
4. **Validación experimental:** Los resultados confirman la teoría de paralelización para este tipo de problemas

8.2. Hallazgos Principales

8.2.1. Efectividad de la Paralelización

- **Dependiente del tamaño del problema:** La paralelización solo es efectiva cuando el tiempo de cómputo supera significativamente el overhead de comunicación
- **Speedup máximo observado:** 1.899x con 4 procesos para problemas de tamaño medio, y 1.781x para casos tardíos
- **Eficiencia consistente:** Entre 44.5 % y 47.5 % para casos de complejidad media/alta, indicando un comportamiento estable del algoritmo
- **Patrón reproducible:** Los casos de llave media y tardía muestran resultados similares, confirmando la predictibilidad del comportamiento paralelo

8.2.2. Limitaciones Identificadas

- **Overhead de MPI:** Significativo para problemas pequeños (¿100ms de inicialización)
- **Desbalance de carga inherente:** La naturaleza aleatoria de la posición de la clave causa distribución desigual del trabajo
- **Escalabilidad limitada:** Los costos de comunicación crecen con el número de procesos

8.3. Implicaciones Prácticas

8.3.1. Para Ataques Criptográficos Reales

- La paralelización es especialmente efectiva para espacios de búsqueda grandes (millones-billones de claves)
- El overhead de MPI se vuelve negligible en ataques prolongados
- La distribución de trabajo debe considerar balanceo dinámico de carga

8.3.2. Para Implementaciones de Seguridad

- DES es vulnerable a ataques de fuerza bruta paralelos en tiempo razonable
- Se confirma la necesidad de algoritmos más robustos (AES-256)
- Los sistemas deben considerar la capacidad de cómputo paralelo de los atacantes

8.4. Lecciones Aprendidas

1. **Granularidad importa:** Problemas muy pequeños no se benefician de paralelización
2. **Overhead real:** MPI introduce costos significativos que deben considerarse en el diseño
3. **Distribución de trabajo:** La división estática simple puede ser subóptima
4. **Medición precisa:** Timing de alta resolución es crucial para evaluar mejoras pequeñas

9. Recomendaciones

9.1. Mejoras a la Implementación Actual

9.1.1. Optimizaciones de Rendimiento

- **Balanceo dinámico de carga:** Implementar un esquema maestro-trabajador donde los procesos soliciten trabajo adicional cuando terminan
- **Terminación temprana optimizada:** Usar comunicación no bloqueante para detectar cuando otro proceso ya encontró la clave
- **Reducción de comunicación:** Minimizar broadcasts usando topologías de comunicación más eficientes
- **Overlapping de cómputo y comunicación:** Usar `MPI_Isend/MPI_Irecv` para solapar comunicación con cómputo

9.1.2. Algoritmos Alternativos

- **Búsqueda probabilística:** Implementar muestreo aleatorio del espacio de claves en lugar de búsqueda secuencial
- **Técnicas criptoanalíticas avanzadas:** Combinar fuerza bruta con ataques diferencial/linear cuando sea aplicable
- **GPU acceleration:** Portar el algoritmo a CUDA/OpenCL para mayor paralelización

9.2. Extensiones del Proyecto

9.2.1. Algoritmos Adicionales

- Implementar ataques contra 3DES y AES con claves reducidas

- Explorar ataques de diccionario para contraseñas débiles
- Desarrollar herramientas de análisis estadístico de claves

9.2.2. Plataformas de Despliegue

- **Cluster computing:** Evaluar rendimiento en clusters de múltiples nodos
- **Cloud computing:** Implementar en plataformas como AWS/Azure para escalabilidad masiva
- **Computación heterogénea:** Combinar CPU + GPU para maximizar throughput

9.3. Consideraciones de Investigación Futura

9.3.1. Estudios de Escalabilidad

- Evaluar comportamiento con 8, 16, 32+ procesos
- Medir eficiencia en función del tamaño del espacio de búsqueda
- Comparar diferentes topologías de red (Ethernet, InfiniBand)

9.3.2. Análisis Teórico

- Modelar matemáticamente la probabilidad de distribución del trabajo
- Desarrollar métricas de balance de carga específicas para búsqueda criptográfica
- Estudiar el impacto de diferentes patrones de distribución de claves

9.4. Aplicaciones Prácticas

9.4.1. Auditoría de Seguridad

- Desarrollar herramientas de evaluación de robustez criptográfica
- Crear benchmarks estándar para medir resistencia a ataques
- Implementar sistemas de alerta temprana para claves débiles

9.4.2. Educación en Ciberseguridad

- Usar como plataforma de enseñanza de conceptos de criptoanálisis
- Demostrar la importancia de selección apropiada de algoritmos
- Ilustrar principios de computación paralela aplicada

9.5. Consideraciones Éticas

- Este tipo de herramientas debe usarse únicamente con fines educativos y de investigación autorizados
- Su uso para ataques no autorizados constituye una violación de la ley
- Se recomienda implementar salvaguardas técnicas y legales apropiadas

10. Referencias

Referencias

[1] National Institute of Standards and Technology. *Data Encryption Standard (DES)*. Federal Information Processing Standards Publication 46-3, 1999.

[2] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard*. Version 4.0, 2021.

[3] Schneier, Bruce. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. 2nd Edition, John Wiley & Sons, 1996.