Universidad de los Andes
Ingeniería de Sistemas y Computación
ISIS 2203 Infraestructura Computacional
Semestre 2023-2

Caso de estudio # 3
Seguridad

Objetivos

- Comprender el alcance y limitaciones en la aplicación de códigos criptográficos de hash para el manejo de confidencialidad e integridad de datos.
- Construir un prototipo a escala de una herramienta que evalúa el esquema para manejo de integridad usado en la tecnología bitcoin.

Problemática:

Bitcoin registra las transacciones en bloques de un tamaño definido y encadena los códigos criptográficos de hash de dichos bloques para garantizar la integridad de los datos. Adicionalmente, bitcoin incluye un procedimiento conocido como "minado"; este procedimiento se debe ejecutar por cada bloque y se encarga de buscar un valor (v) que concatenado con las transacciones que ya están en el bloque, permite generar un código criptográfico de hash (ch) que cumple una condición específica; usualmente la condición se especifica como el número de ceros que el código generado (ch) debe tener al comienzo. Por las propiedades de los algoritmos de generación de códigos criptográficos de hash, el proceso de minado debe usar una aproximación de ensayo y error, intentando diferentes valores hasta encontrar uno que cumpla con la condición buscada. El costo del minado, el uso de firmas digitales y la replicación de la información permiten garantizar la integridad de los datos almacenados.

Nota: en el artículo que propone bitcoin originalmente puede encontrar información adicional. La referencia se encuentra al final del enunciado.

A. [90%] Implementación del Prototipo.

Para comprender el costo del proceso de minado (y por qué ayuda a garantizar la integridad de la información almacenada) vamos a implementar un prototipo simplificado de las operaciones involucradas. Su tarea consiste en escribir un programa en Java que:

- Recibe como datos de entrada: (i) un algoritmo de generación de código criptográfico de hash, (ii) una cadena C que representa los datos de una transacción, (iii) un entero que indica el número buscado de bits en cero y (iv) un entero que indica el número de threads que debe correr: los valores posibles son 1 y 2.
- Explora un espacio de búsqueda, buscando un valor v tal que el código criptográfico de hash de la cadena de entrada concatenada con v tenga en su comienzo el número buscado de ceros. Si el número de threads es 2 el programa debe distribuir el espacio de búsqueda entre los dos threads (observe que la distribución debería ser equitativa para maximizar el beneficio).

Su programa debe cumplir con las siguientes condiciones:

- No puede usar las instrucciones break, ni continue. Vamos a contribuir al debate break, continue, goto vs. programación estructurada.
- No use librerías especiales, solo las librerías estándar de java (java.security). Si tiene alguna duda sobre una librería específica consulte con los profesores.
- El programa debe reportar la cadena de entrada usada (C), el valor v que permite cumplir la condición definida y el tiempo que tomó hacer la búsqueda.
- El programa debe terminar si encuentra una respuesta. (Observe que en el caso de dos threads esto significa que, si uno de los dos encuentra una respuesta entonces los dos threads deben terminar).
- No puede usar stop, sleep, ni interrupt.

Para simplificar un poco el problema:

- Escoja una cadena C con datos de entrada con una longitud de entre 16 y 20 caracteres.
- Los algoritmos para generación de código criptográfico de hash que usaremos son SHA256 y SHA512.

- El número de ceros buscado puede ser: 20, 24, 28, 32 o 36. Este número corresponde a la cantidad de bits en cero que deben aparecer al comienzo del código hash generado.
- El espacio de búsqueda para el valor v debe considerar cadenas con una longitud entre 1 y 7 caracteres (longitud máxima de 7 caracteres) y solamente consideraremos letras minúsculas (a-z). Tenga en cuenta que, en el caso de bitcoin, y otras aplicaciones que usan la técnica de minado, se trabaja con todas las combinaciones posibles de bits, no solo con los códigos correspondiente a letras.
- Adicionalmente, por limitar el tamaño del espacio de búsqueda, es posible que después de explorar todos los valores en el espacio no se encuentre una respuesta. Si no se encuentra respuesta, entonces el tiempo reportado debería corresponder al tiempo de revisión de todo el espacio de búsqueda. Tenga en cuenta que, en un contexto real sería necesario continuar generando valores hasta encontrar uno que permita cumplir con la condición.

Adicionalmente, resuelva las siguientes tareas:

1. Corra su programa en diferentes escenarios construidos a partir de las variaciones en los datos de entrada: variando algoritmo, sobre dos cadenas diferentes, para número de ceros 20, 24, 28, 32 o 36 y con 1 o 2 threads. Construya una tabla con los datos recopilados.
2. Construya dos gráficas:
    a. Una gráfica que ilustre el comportamiento para el programa monothread (sobre todas las variaciones – Usando los datos del punto anterior).
    b. Una gráfica que ilustre el comportamiento del programa con dos threads (sobre todas las variaciones – Usando los datos del punto anterior).
3. Identifique la velocidad de su procesador, y estime cuántos ciclos de procesador toma, en promedio, generar y evaluar un valor para determinar si cumple o no con la condición buscada. Escriba todos sus cálculos.
4. Con base en los cálculos del punto anterior, calcule cuánto tiempo tomaría un programa monothread, en el peor caso (explorar todo el espacio de búsqueda).

B. [10%] Análisis y Entendimiento del Problema.

Responda las siguientes preguntas.

Nota: No copie contenido de otros autores, construya sus propias respuestas con base en las referencias consultadas (e incluya dichas referencias en el informe).

1. Busque información adicional sobre los algoritmos de generación de códigos criptográficos de hash y responda las siguientes preguntas: (i) ¿cuáles se usan hoy día?, (ii) ¿por qué dejamos de usar aquellos que se consideran obsoletos?, (iii) ¿qué referencias bibliográficas usó para responder esta pregunta?, (iv) ¿por qué esas referencias tienen autoridad sobre este tema?
2. La tecnología blockchain se construyó a partir de la propuesta de bitcoin. Busque información sobre blockchain y presente un caso de uso en el contexto de la Universidad de los Andes donde sea útil la tecnología blockchain. Justifique su respuesta con argumentos concretos, en particular, responda las siguientes preguntas: (i) ¿cuál (o cuáles) de los cuatro problemas de seguridad, de los estudiados en clase, resuelve blockchain en el caso presentado? y (ii) ¿cómo los resuelve? (es decir, divida la tecnología en componentes e identifique qué parte, o partes, de la tecnología están involucradas en la resolución del problema y cómo lo resuelven).

Entrega:

- Cada grupo debe entregar un zip de un proyecto Java con la implementación del prototipo. Además, en el subdirectorio docs debe haber un archivo que incluya el informe con las respuestas a los puntos A y B de este enunciado. **Al comienzo del informe, deben estar los nombres y carnés de los integrantes del grupo**. Si un integrante no aparece en el documento entregado, el grupo podrá informarlo posteriormente. Sin embargo, habrá una penalización: la calificación asignada será distribuida (dividida de forma equitativa) entre los integrantes del grupo.
- Recuerde incluir en su informe **todas las referencias** que use para resolver este proyecto.
- El trabajo se realiza en los grupos asignados. No debe haber consultas entre grupos.
- El grupo responde solidariamente por el contenido de todo el trabajo, y lo elabora conjuntamente (no es trabajo en grupo repartirse puntos o trabajos diferentes). Se puede solicitar una sustentación a cualquier miembro del grupo sobre cualquier parte del trabajo. Dicha sustentación será parte de la calificación de todos los miembros.
- El proyecto debe ser entregado por Bloqueneon por uno solo de los integrantes del grupo.

- **La fecha límite de entrega es noviembre 8 a las 23:50 p.m.**

Referencias:

- *Cryptography and network security*, W. Stallings, Ed. Prentice Hall, 2003
- *Computer Networks*. Andrew S. Tanenbaum. Cuarta edición. Prentice Hall 2003, Capítulos 7, 8.
- *To use or not to use the goto statement*: Programming styles viewed from Hoare Logic. Hidetaka Kondoh, Kokichi Futatsugi. Science of Computer Programming. Volume 60, issue 1. 2006
- *Bitcoin: A Peer-to-Peer Electronic Cash System*. Satoshi Nakamoto. 2009

Cronograma Propuesto:

- 24 de octubre: Publicación del enunciado
- 25 de octubre: Comunicarse con los compañeros de grupo (tenga en cuenta que este paso es importante para establecer si alguien del grupo piensa retirar el curso)
- 24 a 26 de octubre: Leer el enunciado completo
- 27 a 30 de octubre: Implementar y probar el prototipo (con longitudes cortas para validar su funcionamiento)
- 31 octubre a 5 de noviembre: Correr el prototipo para todos los escenarios requeridos con longitudes mayores y recopilar los datos
- 6 a 8 de noviembre: Informe y entrega
- 9 y 10 de noviembre: Coevaluación