Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ciencias de la Computación

CC6908: Introducción al Trabajo de Título

Implementación de un algoritmo distribuido de generación de llaves RSA

Profesor Guía

Profesor Co-Guía

Alumno Memorista

Profesor Guía: Javier Bustos
Profesor Co-Guía: Alejandro Hevia
Alumno: Caterina Muñoz
Carrera: Ingeniería Civil en Computación
Rut: 18.025.979-1
Email: caterina@niclabs.cl
Telefono: +56984799379
Fecha: 19 de enero de 2016

Índice

1. Introducción

2. Motivación

3. Marco Teórico

3.1. Criptografía Asimétrica

3.2. Criptografía Umbral

3.3. Llaves RSA

4. Objetivos

4.1. Objetivo general

El objetivo principal de esta memoria es implementar el algoritmo de generación distribuida de llaves RSA propuesto en [2]. La razón de la elección de este algoritmo es

explicar por qué este y no otro

Además, se debe evaluar la implementación de una o más de las optimizaciones propuestas en [2, 3]

buscar otros papers con más optimizaciones

y se deben realizar pruebas para comprobar el correcto funcionamiento del algoritmo imple-

mentado.

4.2. Objetivos específicos

Para conseguir el objetivo principal de la memoria se pueden detallar varios objetivos específicos expuestos a continuación.

5. Algoritmo Propuesto

A continuación se explica el algoritmo propuesto por Boneh y Franklin en [2]. Dado que este algoritmo tiene como objetivo la generación distribuida de llaves RSA, al final de su ejecución se debe tener:

- Un módulo RSA $N = p \cdot q$, donde p y q son primos primos de al menos n bits.
- Un par de llaves público - privada e, d, donde $e \cdot d = 1 \mod \varphi(N)$.
- N y e son públicos
- y d está repartido entre los k nodos.

  falta explicar t

El algoritmo sigue los siguientes pasos:

1) **Elección de Candidatos**

   Cada uno de los k nodos elige dos números aleatorios de n bits: q_i y p_i que mantiene en secreto.

2) Computación de N usando BGW

Usando una versioń simplificada del método BGW [1], los nodos calculan $N = (p_1 + p_2 + \ldots + p_k) \cdot (q_1 + q_2 + \ldots + q_k)$. Al final de este paso, N es público.

3) Test de Primalidad

Los nodos ocupan un test de primalidad distribuido para determinar si N es el producto de dos primos. En el caso de que no lo sea, el algoritmo vuelve a empezar.

4) Generación de d

Luego de obtener N, los nodos deben computar partes aditivas d_i tal que $\sum_{i=1}^{k} d_i = d$. Así, al final de este paso, cada nodo queda con una parte d_i de d.

5) Repartición de d

Que cada nodo tenga una de las partes d_i generadas en el paso anterior no es suficiente para realizar criptografía umbral. Esto, porque se quiere poder realizar una operación criptográfica con un subconjunto cualquiera de t nodos. Para lograr esto, cada d_i se debe repartir de manera redundante entre los k nodos de manera tal que un subconjunto cualquiera de t nodos pueda "reconstruir" el d_i en cuestión. Con esto, se logra que mismo efecto para d.

6. Diseño del sistema

7. Trabajo Realizado

8. Plan de trabajo

9. Bibliografía

[1] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault tolerant distributed computation. In *Proc. STOC*, 1988.

[2] Dan Boneh and Matthew Franklin. Efficient generation of shared rsa keys. *Journal of the ACM (JACM)*, 48(4):702–722, July 2001.

[3] Pierre-Alain Fouque and Jaques Stern. Fully distributed threshold rsa under standard assumptions. In *ASIACRYPT*, 2001.