ĐỀ XUẤT ĐỀ TÀI

Học phần: Nhập môn an toàn thông tin

Thuật toán Shamir's Secret Sharing và ứng dụng

Thành viên nhóm

Bùi Thanh Lâm

Phạm Văn Nam

Đậu Thị Vân Anh

Hà Nội, tháng 6 năm 2021

TÓM TẮT: Nhóm chúng em đề xuất công việc sẽ thực hiện bao gồm: (1) cài đặt và đánh giá thuật toán chia sẻ riêng tư Shamir (Shamir's Secret Sharing – SSS); (2) ứng dụng SSS vào một bài toán cụ thể - bài toán "thư mục chung an toàn" - với giao thức chúng em tự phát triển, gọi là PVSS-FS.

MỘT VÍ DỤ VUI ĐẶT VẤN ĐỀ: Phòng trọ của Alice có 4 người. Có 4 chìa khóa giống nhau phát cho mỗi người 1 chiếc. Như vậy ai cũng có thể tự mình ra vào phòng thoải mái.

Alice không muốn mọi người tự ý đưa người yêu của họ vào phòng mà không xin phép ai cả. Vì thế Alice mua một ổ khóa đặc biệt:

- Ổ có 2 chỗ cắm, muốn mở cửa phải có đủ 2 chìa bất kỳ (các chìa là khác nhau)
- Có 1 chìa cũng không giúp việc mở cửa dễ dàng hơn là không có gì (i.e: chỉ có 1 người thì vẫn phải phá khóa)

Từ đó, mỗi khi muốn mở phòng, cần phải có hai người; nói cách khác, một người chỉ có thể vào phòng dưới sự cho phép của một người bất kỳ khác. Mọi người vẫn có quyền ngang nhau và không phụ thuộc vào bên thứ ba đáng tin cậy (e.g: chủ trọ, bác bảo vệ, hàng xóm).

1. CƠ SỞ LÝ THUYẾT

1.1. THUẬT TOÁN SHAMIR'S SECRET SHARING

Đây là một thuật toán an toàn dùng để chia sẻ khóa thành phần cho n tổ chức để giữ bí mật cho một thông điệp K, sao cho phải có đủ t tổ chức ($t < n$) cùng có mặt mới có thể khôi phục được thông điệp K.

Lưu ý: Không nên nhầm lẫn thuật ngữ "khóa thành phần" với khóa. Trên thực tế, các k_i không phải là một phần của khóa K, dù chúng có mối liên hệ mật thiết với nhau và với K về toán học.

Kí hiệu: $x_1, x_2, \ldots, x_n$ là định danh công khai của n tổ chức, hàm sinh số ngẫu nhiên G, Z_p là trường số nguyên với module p (p là một số nguyên tố lớn).

Thuật toán có các bước lần lượt sau đây:

Thuật toán 1: Shamir's Secret Sharing

Phase 1: Sinh các khóa thành phần

Input: K

Output: n khóa thành phần $k_1, k_2, \ldots, k_n$

- Bước 1: Trong Z_p, chọn t-1 số $a_1, a_2, \ldots, a_{t-1}$ ngẫu nhiên bởi hàm G một cách bí mật và $a_0 = K$
- Bước 2: Dựng đa thức $P(x) = a_0 + (a_1x + a_2x^2 + a_{t-1}x^{t-1}) \bmod p$
- Bước 3: Lặp lại với i chạy từ 1 đến n: tính toán các giá trị $k_i = P(x_i)$

Phase 2: Tái tạo lại thông điệp K từ t khóa thành phần

Input: t khóa thành phần $k_1, k_2, \ldots, k_t$

Output: K

(Sử dụng hệ phương trình tuyến tính)

Ta có hệ phương trình sau:

$$\begin{cases} a_0 + a_1 x_1 + \cdots + a_{t-1} x_1^{t-1} = k_1 \\ a_0 + a_1 x_2 + \cdots + a_{t-1} x_2^{t-1} = k_2 \\ \cdots \\ a_0 + a_1 x_2 + \cdots + a_{t-1} x_{t-1}^{t-1} = k_t \end{cases}$$

Hay viết lại dưới dạng ma trận Aa = k:

$$\begin{pmatrix} 1\ x_1 \ldots x_1^{t-1} \\ 1\ x_1 \ldots x_1^{t-1} \\ \ldots \ldots \ldots \ldots \\ 1\ x_t \ldots x_t^{t-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \ldots \\ a_{t-1} \end{pmatrix} = \begin{pmatrix} k_1 \\ k_2 \\ \ldots \\ k_t \end{pmatrix} (*)$$

Đây là hệ t phương trình tuyến tính t ẩn, vì thế chúng có nghiệm $\{a_0, a_1, \ldots, a_{t-1}\}$ duy nhất. Thật vậy, định lý Rouché-Capelli đã chứng minh điều này.

Định lý 1: Định lý Rouché-Capelli

Hệ phương trình Ax = B có nghiệm duy nhất khi và chỉ khi det(A) ≠ 0

Định lý 2: Định lý về ma trận Vandermonde

Ma trận A là ma trận Vandermonde nếu có mọi hàng (hoặc cột) có giá trị đầu tiên bằng 1, các giá trị tiếp theo là một cấp số nhân.

Cho A = $\begin{pmatrix} 1\ a_1 \ldots a_m^n \\ \ldots \\ 1\ a_m \ldots a_m^n \end{pmatrix}$ là một ma trận Vandermonde bậc n.

Ta có:

$$\det(A) = \prod_{1 \le i \le j \le n} (a_j - a_i)$$

Chứng minh tính hoàn chỉnh của thuật toán SSS:

Hệ (*) có ma trận hệ số A là một ma trận Vandermonde bậc t-1. Do đó, theo Định lý 2:

$$\det(A) = \prod_{1 \le k \le j \le t-1} (x_j - x_k)\ mod\ p$$

Vì x_i là các định danh nên chúng đôi một khác nhau. Do đó, det(A) ≠ 0 và (*) có nghiệm duy nhất theo Định lý 1.

Có nhiều cách tìm nghiệm của (*) (e.g: khử Gauss, Lagrange interpolation); chúng em sẽ không trình bày ở đây.

Chúng em sẽ cài đặt thuật toán SSS trên trường số nguyên hữu hạn Z_p.

1.2. PEDERSEN'S VERIFIABLE SECRET SHARING

Trong thuật toán SSS trên, ta chưa có cơ chế để kiểm tra các k_i của các tổ chức gửi lên trong phase 2 có đúng là các k_i họ đã được nhận trong phase 1 hay không. Kẻ gian (có thể nằm trong các tổ chức) có thể cố tình gửi sai k_i của mình hòng phá bĩnh hoặc hy vọng tìm ra thêm thông tin.
Để khắc phục điều này, ta sẽ sử dụng một cơ chế là commitment scheme (CS), kết hợp với SSS, tạo ra một phiên bản cải tiến mang tên là Pedersen's Verifirable Secret Sharing (PVSS).

Thuật toán 2: Commitment Scheme

Ý tưởng của thuật toán này là thay vì gửi toàn bộ thông điệp s, người gửi chỉ gửi một lời cam kết (commitment, c) với thông điệp đó. Sau khi người gửi gửi thêm s và t, thông điệp này mới được tiết lộ. Thuật toán sẽ kiểm tra xem s, t, c có thỏa mãn đẳng thức hay không.
(Người gửi là A, người nhận là B)
Phase 1: Gửi commitment (thực hiện phía A)
- Bước 1: A chọn $g \in Z_p^*$. B chọn $h \in Z_p^*$ và gửi cho A
- Bước 2: A tính $c = g^s h^t \bmod p$ (*)
- Bước 3: A gửi c cho B

Phase 2: Kiểm tra commitment (thực hiện phía B)
- Bước 1: Nhận s, t từ phía A
- Bước 2: Kiểm tra c có bằng $g^s h^t$ hay không

Thuật toán này được đảm bảo bởi tính an toàn của bài toán logarit rời rạc. A không thể tạo ra một cặp s', t' khác vẫn thỏa mãn điều kiện (*); đồng nghĩa với việc không thể gửi đi một thông điệp s' khác.

Thuật toán 3: Pedersen's Verifiable Secret Sharing

Phase 1: Sinh các khóa thành phần

Input: K

Output: n khóa thành phần $s_1, s_2, \ldots, s_n$

- Bước 1: Trong Z_p, chọn t-1 số $a_1, a_2, \ldots, a_{t-1}$ ngẫu nhiên bởi hàm G một cách bí mật và a_0=K. Chọn thêm t số $b_0, b_1, \ldots, b_{t-1}$ ngẫu nhiên bởi hàm G một cách bí mật
- Bước 2: Dựng đa thức $P(x) = a_0 + (a_1x + a_2x^2 + a_{t-1}x^{t-1}) \bmod p$
- Bước 3: Dựng đa thức $Q(x) = b_0 + (b_1x + b_2x^2 + b_{t-1}x^{t-1}) \bmod p$
- Bước 4: Lặp lại với i chạy từ 1 đến n: tính toán các giá trị $k_i = P(x_i)$, $t_i = Q(x_i)$
- Bước 5: Tính $c_i = g^{a_i} h^{b_i}$ (i = 0, 1, …, t-1) và lưu lại các c_i này
- Bước 6: Gửi các $s_i = (k_i, t_i)$

Phase 2: Tái tạo lại thông điệp K từ t khóa thành phần

Input: t khóa thành phần $s_1, s_2, \ldots, s_t$

Output: K

- Bước 1: Kiểm tra tính hợp lệ của các s_i bằng cách kiểm tra đẳng thức: (thuật toán 2)

$$g^{k_i} h^{t_i} = \prod_{j=0}^{t-1} c_j^{x_i^j}$$

- Bước 2: Tính K bằng cách giải hệ phương trình tuyến tính (thuật toán 1)

2. BÀI TOÁN ỨNG DỤNG

Phát biểu bài toán: Công ty nọ có n người cùng sở hữu tài liệu D, lưu tại máy chủ dữ liệu. Mỗi khi công việc tạm xong, họ cần lưu trữ nguyên hiện trạng của D cho đến khi có đủ người cùng tiếp tục công việc.

Vấn đề: Các phương pháp thường dùng hiện nay là phân quyền cho nhiều người ở cấp các độ khác nhau. Tuy nhiên, tệp tin D vẫn lưu ở một nơi nào đó (có thể là máy cá nhân hoặc cloud storage). Như vậy nếu có ai đó tìm cách tấn công vào nơi chứa D, rất có thể họ sẽ tự ý xem và chỉnh sửa D. Ngoài ra, cách làm như vậy luôn tồn tại một người có quyền cao nhất; D có thể bị chỉnh sửa nếu người đó muốn hoặc bị ai đó tấn công.

Phương án: Mã khóa D bằng một hệ mật an toàn với khóa bí mật K. Chia sẻ khóa K bằng một *giao thức chia sẻ khóa riêng tư* tới nhiều người, trong đó mỗi người chỉ nhận được một khóa thành phần. Chỉ khi có đủ t khóa thành phần, khóa K mới được tái tạo lại và có thể dùng để mở tài liệu D.

Chúng em sẽ thực hiện một mô phỏng như sau: cài đặt chương trình "chia sẻ tệp tin an toàn" trên tất cả các máy tính của công ty. Chi tiết chương trình sẽ được giới thiệu ở phần tiếp theo.

3. CHƯƠNG TRÌNH THỬ NGHIỆM

3.1. CÁC CÔNG NGHỆ SỬ DỤNG

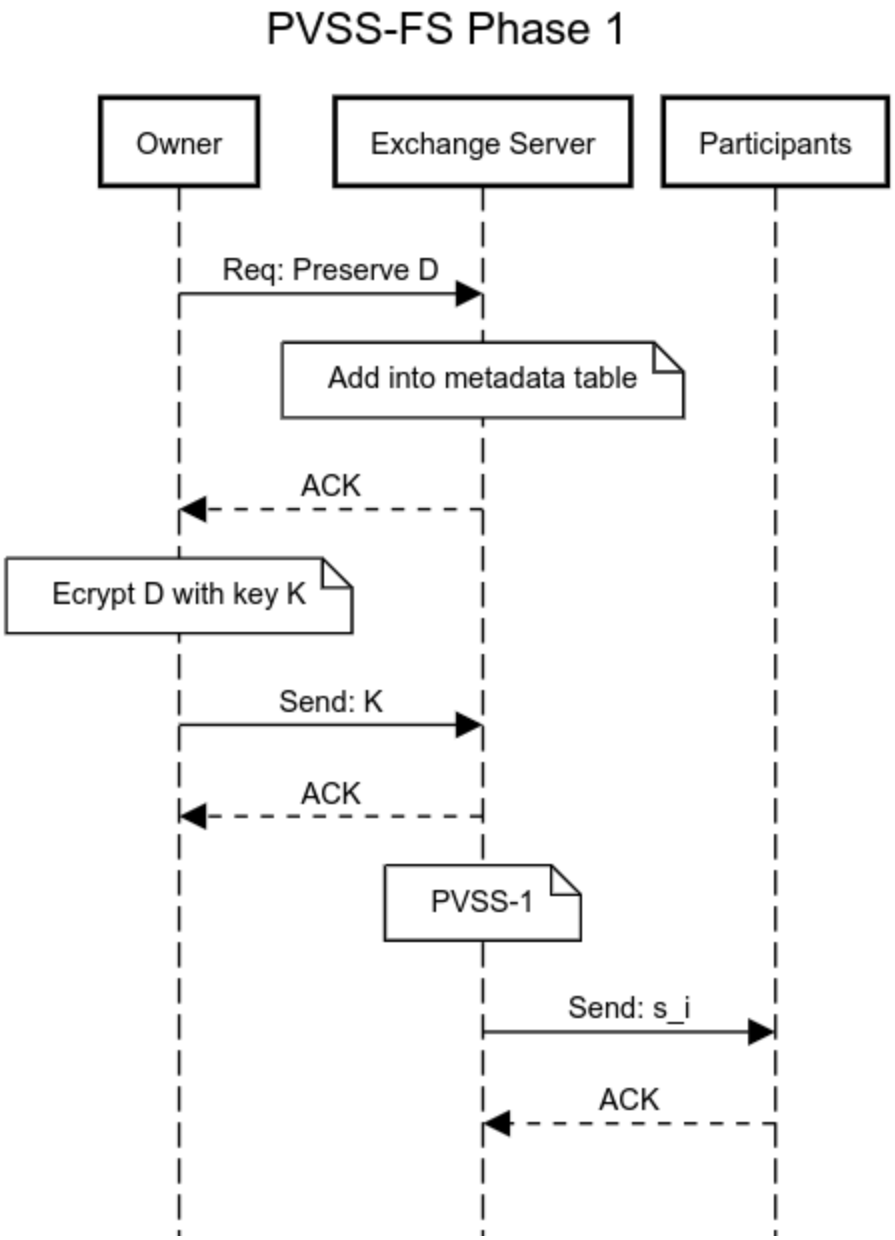
Nhóm chúng em dự kiến lập trình bằng Python và C++. Trong đó, Python dùng để viết phần giao tiếp thông tin giữa các bên (sử dụng thư viện built-in là socket và ssl) và giao diện chương trình; C++ dùng để viết các hàm toán học, tính toán với số lớn và trong trường Z_p.

Để thuận tiện demo, chúng em dự kiến sử dụng một máy tính cá nhân đóng vai trò là exchange server, được exposed lên public qua ngrok. Trên thực tế, code sẽ được execute trên một cloud server có tính bảo mật cao hơn (chẳng hạn Google Cloud, AWS,...). Client là các máy tính cá nhân, giao tiếp với server qua giao thức chúng em đã thiết kế. Phía client sẽ có một giao diện đồ họa đơn giản, viết bằng thư viện kivy của Python (có thể chạy trên mobile).

3.2. CHI TIẾT VỀ CHƯƠNG TRÌNH

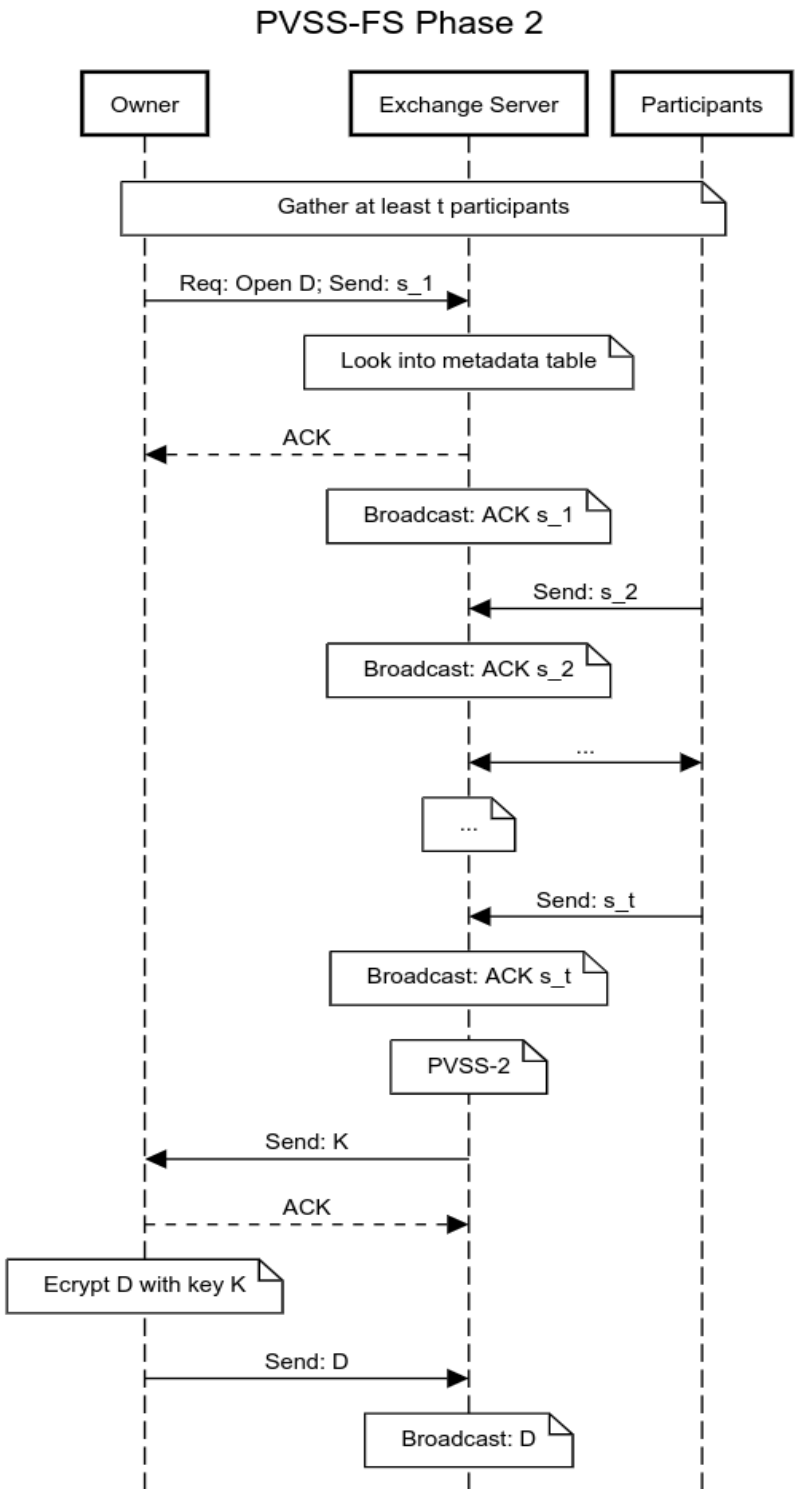
Chúng em sử dụng các ký hiệu như sau:

- Exchange server (S): có trách nhiệm khởi tạo, phân phối, tập hợp và tái tạo khóa.
- Participants (P): là các clients muốn xem/chỉnh sửa tệp tin D_j của người khác, hoặc chia sẻ tệp tin D_i của mình. Người muốn chia sẻ tệp tin còn gọi là owner (O).



Hình 1. Sơ đồ luồng dữ liệu giữa các thành phần của hệ thống – Phase 1

Khi O yêu cầu bảo toàn tệp tin D_o từ máy mình, thuật toán mã hóa AES sẽ chạy với key K trên máy của O và D_o sẽ bị xóa. O gửi K đến S. Ở S, thuật toán PVSS phase 1 sẽ chạy và gửi các s_i cho P_i. Kể từ lúc này (gọi là giai đoạn safe period), tệp tin D_o đã tạm thời bị đóng băng, không ai có thể đơn phương xem/chỉnh sửa, ngay cả chính O hoặc tình huống có kẻ gian xâm nhập vào máy của O.



Hình 2. Sơ đồ luồng dữ liệu giữa các thành phần của hệ thống – Phase 2

Đến thời điểm thích hợp, ta sẽ tập hợp ít nhất t thành viên O, P_1,..., P_{t-1} để cùng nhau mở D_o. Khi O yêu cầu mở tệp tin từ máy mình, S sẽ khởi tạo một hàng đợi có phiên hữu hạn. Tiếp đó, các P_i sẽ gửi s_i của mình đến S. Khi tập hợp đủ s, thuật toán PVSS phase 2 sẽ chạy trên S để tái tạo lại K.

Nếu thành công, S sẽ gửi trả lại K đến O. O có thể dùng K để giải mã D_o bằng AES tại máy O. Sau đó, O gửi D_o lên S và cho phép tất cả các P_i có thể kết nối đến và xem, chỉnh sửa và tải về máy của mình. Ở giai đoạn này (gọi là public period), tập tin D_o không còn bảo mật nữa; mọi người có thể dùng với mục đích nào đó cho đến khi O lại gửi yêu cầu bảo toàn cho D_o.

Nếu trong phiên, S chưa nhận đủ t khóa thành phần, hàng đợi sẽ đóng lại và O phải gửi lại yêu cầu mở D_o. Nếu nhận đủ t khóa thành phần, nhưng vì lý do nào đó không thể tái tạo được K, hoặc K bị sai khiến không thể giải mã D_o đúng cách, S sẽ có cơ chế detect được s_i nào đã hỏng, và đóng phiên nhận khóa thành phần.

Khi triển khai chương trình, các máy tính gửi và nhận thông điệp thông qua giao thức TCP/TLS. Đây là giao thức bảo mật, được sử dụng phổ biến, dựa trên các hệ mật an toàn nhất hiện nay như RSA, ECDH,...

Để quản lý các D_o đang trong safe period, phía S sẽ có một bảng lưu metadata của D bao gồm: tên file, tên owner, ngày bắt đầu bảo toàn,...

Phía S sẽ không lưu lại K trong safe period, cũng như không lưu D khi hết public period. Các P tuy có thể tải về D trong public period, tuy nhiên giao thức không đề cao bảo mật nội dung của D mà chủ yếu dùng để chống chỉnh sửa D trong safe period.

Nhận xét: Giao thức PVSS - FS có những ưu điểm sau:

- Cho trước m khóa thành phần ($m < t$), việc tái tạo lại khóa K là một bài toán khó tương đương với không có bất kỳ khóa thành phần nào
- Cho phép n' tùy ý người mới được phép tham gia vào giữa hai phase mà không ảnh hưởng gì. Người mới vẫn được cấp các khóa thành phần của riêng minh và có quyền tương đương người khác (i.e: threshold t không đổi)
- Cho phép tối đa n-t người rời bỏ mà không ảnh hưởng gì
- Nếu ai đó gửi k sai, PVSS có thể phát hiện là do ai