|  |
| --- |
| BAN CƠ YẾU CHÍNH PHỦ  **HỌC VIỆN KỸ THUẬT MẬT MÃ**  ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯ |
| ĐỒ ÁN TỐT NGHIỆP  **XÂY DỰNG PHẦN MỀM KÝ SỐ, MÃ MẬT CMS, JSON VÀ XML TRÊN WINDOWS** |
| Ngành: An toàn thông tin  Mã số: 7.48.02.02  *Sinh viên thực hiện*:  **Võ Lê Huy – AT131315**  Lớp: AT13P  *Cán bộ hướng dẫn*:  **Tiến sĩ Lê Quang Huy**  Cục chứng thực số và Bảo mật thông tin |
| Thành phố Hồ Chí Minh, 2021 |
| BAN CƠ YẾU CHÍNH PHỦ  **HỌC VIỆN KỸ THUẬT MẬT MÃ**  ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯ |
| ĐỒ ÁN TỐT NGHIỆP  **XÂY DỰNG PHẦN MỀM KÝ SỐ, MÃ MẬT CMS, JSON VÀ XML TRÊN WINDOWS** |
| Ngành: An toàn thông tin  Mã số: 7.48.02.02  *Sinh viên thực hiện*:  **Võ Lê Huy – AT131315**  Lớp: AT13P  *Cán bộ hướng dẫn*:  **Tiến sĩ Lê Quang Huy**  Cục chứng thực số và Bảo mật thông tin |
| Thành phố Hồ Chí Minh, 2021 |

**Mục lục**

[CHƯƠNG 1: HẠ TẦNG MẬT MÃ KHÓA CÔNG KHAI 1](#_Toc90216124)

[**1.1.** **Một số khái niệm về PKI** 1](#_Toc90216125)

[**1.2.** **Sự cần thiết của PKI** 1](#_Toc90216126)

[**1.3.** **Kiến trúc hạ tầng PKI** 4](#_Toc90216127)

[**1.4.** **Các thành phần, dịch vụ** 4](#_Toc90216128)

[1.4.1. Các thành phần của PKI 4](#_Toc90216129)

[1.4.2. Certification Authority (CA) 7](#_Toc90216130)

[1.4.3. Registration Authority (RA) 9](#_Toc90216131)

[1.4.4. Validation Authority (VA) 10](#_Toc90216132)

[1.4.5. Thực thể cuối 10](#_Toc90216133)

[**1.5.** **Chứng thư số và CRL** 11](#_Toc90216134)

[1.5.1. Chứng thư số 11](#_Toc90216135)

[1.5.2. Danh sách thu hồi chứng thư số CLR 22](#_Toc90216136)

[**1.6.** **Hoạt động của hệ thống PKI** 26](#_Toc90216137)

[1.6.1. Chứng thực cặp khóa 26](#_Toc90216138)

[1.6.2. Sử dụng cặp khóa 26](#_Toc90216139)

[**1.6.** **Ứng dụng của PKI** 29](#_Toc90216140)

[**1.7.** **Miền tin cậy và mở rộng miền tin cậy** 29](#_Toc90216141)

[**1.8.** **Các mô hình hoạt động của PKI** 30](#_Toc90216142)

[1.8.1. Hierarchical PKI 30](#_Toc90216143)

[1.8.2. Mesh PKI 31](#_Toc90216144)

[1.8.3. Single CA 32](#_Toc90216145)

[**1.9.** **Chuẩn, đặc tả, luật pháp và chính sách** 32](#_Toc90216146)

[CHƯƠNG 2: CẤU TRÚC DỮ LIỆU KÝ SỐ, MÃ MẬT CMS, XML JSON 33](#_Toc90216147)

[**2.1.** **Cấu trúc dữ liệu ký số, mã mật CMS** 33](#_Toc90216148)

[2.1.1. Giới thiệu chung 33](#_Toc90216149)

[2.1.2. Các chuẩn về CMS 34](#_Toc90216150)

[2.1.3. Cấu trúc dữ liệu CMS 46](#_Toc90216151)

[**2.2.** **Cấu trúc dữ liệu ký số, mã mật XML** 48](#_Toc90216152)

[2.2.1. Giới thiệu chung 48](#_Toc90216153)

[2.2.2. Các chuẩn về ký số, mã mật XML 51](#_Toc90216154)

[2.2.3. Cấu trúc dữ liệu XML 57](#_Toc90216155)

[**2.3.** **Cấu trúc dữ liệu ký số, mã mật JSON** 59](#_Toc90216156)

[2.3.1. Giới thiệu chung 59](#_Toc90216157)

[2.3.2. Các chuẩn về JSON Web Token 62](#_Toc90216158)

[2.3.3. Cấu trúc dữ liệu JSON 71](#_Toc90216159)

[CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG DEMO 73](#_Toc90216160)

[**3.1.** **Một số thư viện mật mã** 73](#_Toc90216161)

[3.1.1. Thư viện mật mã Bouncy Castle 73](#_Toc90216162)

[3.1.2. Thư viện mật mã Microsoft Crypto API 82](#_Toc90216163)

[**3.2.** **Phân tích thiết kế chương trình** 84](#_Toc90216164)

[3.2.1. Giới thiệu chung về ứng dụng 84](#_Toc90216165)

[3.2.2. Phân tích 85](#_Toc90216166)

[3.2.3. Thiết kế 86](#_Toc90216167)

[Kết luận 89](#_Toc90216168)

[TÀI LIỆU THAM KHẢO 90](#_Toc90216169)

**LỜI CẢM ƠN**

Lời nói đầu tiên, em xin gửi lời cảm ơn đến quý thầy cô Học Viện Kỹ Thuật Mật Mã cũng như quý thầy cô khoa an toàn thông tin đã truyền đạt kiến thức, kinh nghiệm quý báu, quan tâm, hỗ trợ em trong suốt thời gian học tại trường.

Đặc biệt, em xin chân thành cảm ơn Tiến sĩ Lê Quang Huy, người đã hướng dẫn, tạo điều kiện thuận lợi cho em thực hiện khóa luận này.

Mặc dù, em đã cố gắng hoàn thành khóa luận nhưng do hạn chế về thời gian và kiến thức nên không tránh khỏi những sai sót. Em kính mong nhận được sự thông cảm và ý kiến đóng góp từ quý thầy cô để em có thể hoàn thiện và phát triển khóa luận này.

Em xin chân thành cảm ơn.

Thành phố Hồ Chí Minh, 2021

# **CHƯƠNG 1: HẠ TẦNG MẬT MÃ KHÓA CÔNG KHAI**

## **Một số khái niệm về PKI**

Cơ sở hạ tầng: các bộ phận thiết bị phụ thuộc tạo thành một cơ sở của một hệ thống hay các thiết bị cơ bản cố định của một nhà nước. Phân biệt khái niệm cơ sở hạ tầng trong triết học.

Đặc điểm hạ tầng:

* Lan tỏa rộng khắp (mạng điện, đường…), có thể sử dụng bất cứ khi nào.
* Hỗ trợ các ứng dụng (Trong suốt với người dùng, dễ nhận biết và sử dụng).

Hạ tầng an ninh, an toàn mạng: các bộ phận phụ thuộc tạo thành cơ sở của hệ thống an toàn mạng.

Cơ sở hạ tầng khóa công khai (Public Key Infrastructure – PKI) là một hạ tầng an toàn, an ninh mạng, sử dụng các kỹ thuật mật mã khoá công khai nhằm cung cấp các dịch vụ đảm bảo an toàn cho các giao dịch điện tử trên mạng.

Hạ tầng mật mã hóa công khai: là một hệ thống hỗ trợ cho việc áp dụng mật mã khóa công khai vào thực tiễn, nhằm đảm bảo an toàn cho các giao dịch điện tử trong môi trường mạng.

Mục tiêu chính của hạ tầng mật mã khóa công khai: cung cấp các dịch vụ nhằm đảm bảo an toàn cho các giao dịch điện tử.

## **Sự cần thiết của PKI**

Xã hội phát triển, mạng Internet được sử dụng rộng rãi, dẫn tới các giao dịch điện tử được thực hiện ngày càng nhiều, kéo theo các vấn đề về an toàn thông tin.

Các vấn đề về an toàn thông tin cho các giao dịch điện tử: xác thực, bí mật, toàn vẹn, chống chối bỏ.

Các giải pháp đã được áp dụng để đảm bảo an toàn cho các giao dịch như sau:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Các giải pháp/dịch vụ | | Xác thực | Toàn vẹn | Bí mật | Chống chối bỏ |
| Tên/mật khẩu | | Trung bình | Không | Không | Không |
| Sinh trắc | | Mạnh | Không | Không | Không |
| Các kỹ thuật mật mã | Đối xứng | Không | Mạnh | Mạnh | Không |
| **Bất đối xứng** | **Mạnh** | **Mạnh** | **Mạnh** | **Mạnh** |

Theo như bảng thống kê trên thì giải pháp tốt nhất là ứng dụng mật mã, mật mã khóa công khai. Tuy nhiên mật mã khóa công khai khi ứng dụng gặp một số vấn đề như sau:

* Xác thực cặp khóa: ai là chủ sở hữu cặp khóa đã thực hiện ký, mã dữ liệu.
* Chối bỏ hành động: chủ sở hữu cặp khóa có thể chối bỏ việc mình đã thực hiện một hành động (ký lên một tài liệu).
* Thu hồi cặp khóa: làm mất hiệu lực một cặp khóa và thông báo với những người khác có quan hệ giao dịch với mình.

Mật mã khóa công khai khi ứng dụng gặp một số vấn đề. Và cần có hạ tầng để giải quyết các vấn đề đó. PKI là một phần quan trọng của xương sống chiến lược CNTT. PKI rất quan trọng vì công nghệ dựa trên chứng chỉ giúp các tổ chức thiết lập chữ ký, mã hóa và danh tính đáng tin cậy giữa con người, hệ thống và mọi thứ.

Với việc các mô hình kinh doanh đang phát triển ngày càng phụ thuộc nhiều hơn vào các giao dịch điện tử và tài liệu kỹ thuật số, và với nhiều thiết bị nhận biết Internet hơn được kết nối với mạng công ty, vai trò của cơ sở hạ tầng khóa công khai không còn giới hạn ở các hệ thống biệt lập như email an toàn, thẻ thông minh để truy cập vật lý hoặc lưu lượng truy cập web được mã hóa. PKI ngày nay được kỳ vọng sẽ hỗ trợ số lượng lớn hơn các ứng dụng, người dùng và thiết bị trên các hệ sinh thái phức tạp. Và với các quy định nghiêm ngặt hơn về bảo mật dữ liệu của chính phủ và ngành, các hệ điều hành chính thống và các ứng dụng kinh doanh đang trở nên phụ thuộc hơn bao giờ hết vào một tổ chức PKI để đảm bảo sự tin cậy.

## **Kiến trúc hạ tầng PKI**

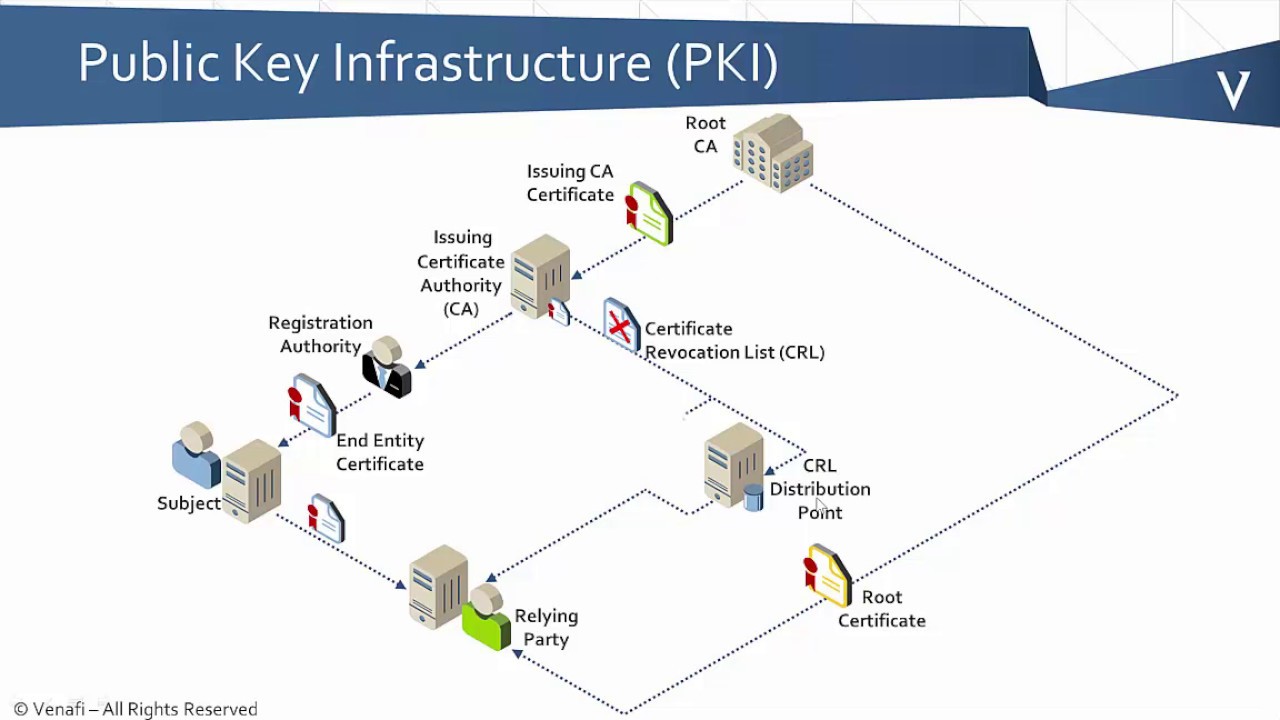


Figure 1. Kiến trúc hạ tần khóa công khai (PKI).

Kiến trúc được chia thành 2 phần riêng biệt: khung pháp lý (văn bản luật, văn bản dưới luật, hệ thông SP, CP và CPS) và hạ tầng kỹ thuật (hạ tầng kỹ thuật, phần cứng, phần mềm, con người, kỹ thuật mật mã).

## **Các thành phần, dịch vụ**

### *Các thành phần của PKI*

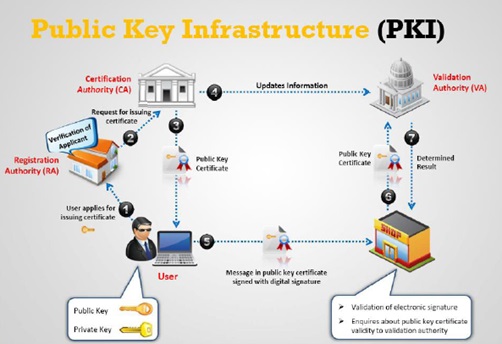


Figure 2. Các thành phần trong PKI.

Mô hình hoạt động: Client – Server

* Hạ tầng - cung cấp dịch vụ: CA, RA, VA.
* Client - sử dụng dịch vụ: người dùng, thiết bị.

Thẩm quyền chứng thực (Certification Authority - CA) bao gồm chức năng và nhiệm vụ.

Chức năng của thẩm quyền chứng thực:

* Chứng thực cặp khóa thuộc hoặc không thuộc một thuê bao.
* Quản lý vòng đời của chứng thư số (cặp khóa) sau khi phát hành.

Nhiệm vụ của thẩm quyền chứng thực: Phát hành, gia hạn, tạm dừng/kích hoạt, thu hồi chứng thư số, khôi phục cặp khóa, phát hành CRL.

Thẩm quyền đăng ký (Registration Authority - RA) bao gồm xác định/xác thực và hỗ trợ người dùng cuối.

Xác định và xác thực của thẩm quyền đăng ký: công việc quản lý (tiếp nhận, kiểm tra và phê duyệt các yêu cầu cấp, thu hồi, gia hạn, tạm dừng, khôi phục … chứng thư số, cặp khóa).

Hỗ trợ người dùng cuối của thẩm quyền đăng ký: tạo cặp khóa quản lý người dùng, bàn giao kết quả.

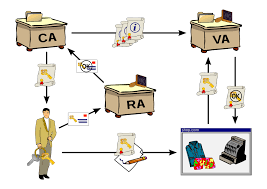


Figure 3. Mối quan hệ giữa VA-CA-RA.

Thẩm quyền xác nhận (Validation Authority - VA):

* Mục tiêu: Khẳng định tính hợp lệ và tin cậy của chứng thư số (cặp khóa).
* Kho công cộng chứa chứng thư số và CRL (LDAP), OCSP…
* Dịch vụ dấu thời gian: TimeStamp, …

Thực thể cuối (client) – người dùng, thiết bị:

* Người dùng, thiết bị, sử dụng chứng thư số - relying party.
* Chủ thể sở hữu chứng thư số (thuê bao) – subscriber.

Thẩm quyền chứng thực (Certification Authority – CA).

Thẩm quyền đăng ký (Registration Authority - RA).

Thực thể cuối (client) – người dùng, thiết bị.

Chứng thực: chứng thực cặp khóa thuộc về một chủ thể (CTS) và chứng thực một cặp khóa không thuộc về một chủ thể (CRL).

Khẳng định tính hợp lệ và tin cậy của các cặp khóa đã chứng thực:

* Kho chứng thư số: chứa tất cả các CTS, CRL do tổ chức chứng thực tạo ra, điểm tin cậy của hệ thống.
* Dịch vụ kiểm tra trạng thái chứng thư trực tuyến OCSP.

Dấu thời gian an toàn: tạo dấu thời gian gắn kèm với dữ liệu thỏa mãn các tính chất chính xác, xác thực và toàn vẹn.

### *Certification Authority (CA)*

#### Vai trò

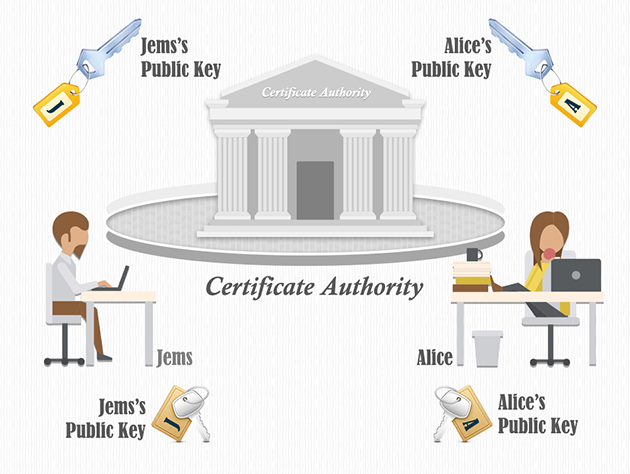


Figure 4. Vai trò của Certification Authority (CA).

Thẩm quyền chứng thực (Certification Authority - CA) là thẩm quyền được tin cậy có nhiệm vụ chứng nhận một cặp khóa thuộc/không thuộc về một thực thể xác định (Thiết lập được một thực thể tin cậy). Đây là thành phần quan trọng nhất của hệ thống.

#### Chức năng

Chức năng chứng nhận một cặp khóa thuộc/không thuộc về một chủ thể (chứng thực).

Quản lý vòng đời chứng thư số (cặp khóa) sau khi chứng thực (tạo chứng thư số).

Tuân thủ CP và CPS.

#### Nhiệm vụ



Figure 5. Nhiệm vụ của CA.

Tạo mới, thay cặp khóa, gia hạn, đổi thông tin (chứng thực thuộc): Tạo chứng thư số.

Tạm dừng/kích hoạt, thu hồi chứng thư số (Chứng thực không thuộc) Tạo danh sách thu hồi chứng thư số (CRL).

Lưu trữ, khôi phục chứng thư số/cặp khóa.

### *Registration Authority (RA)*

#### Vai trò

Thẩm quyền đăng ký (Registration Authority - RA) là:

* Thành phần đảm nhiệm vai trò quản lý hỗ trợ cho thẩm quyền chứng thực CA.
* Trung gian, cầu nối giữa CA và thực thể cuối.

#### Chức năng

Xác định và xác thực.

Hỗ trợ người dùng cuối.

#### Nhiệm vụ

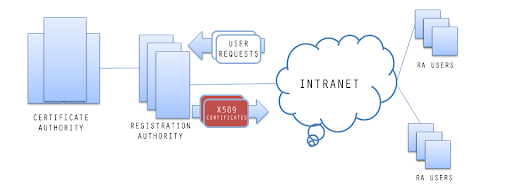


Figure 6. Nhiệm vụ của RA.

Tiếp nhận các yêu cầu chứng thực từ thuê bao.

Kiểm tra và phê duyệt/từ chối các yêu cầu chứng thực (đảm bảo tính xác thực về thông tin của người dùng gửi yêu cầu). Chuyển các yêu cầu đã phê duyệt tới CA.

Tiếp nhận kết quả chứng thực (khóa, chứng thư số) từ CA và chuyển tới thuê bao.

Hỗ trợ thuê bao sinh cặp khóa.

Quản lý thuê bao, yêu cầu, thiết bị lưu khóa (các thông tin về thuê bao).

### *Validation Authority (VA)*

#### Vai trò

Thẩm quyền xác nhận (Verification Authority - VA) khẳng định tính tin cậy và hợp lệ của các cặp khóa đã chứng thực thuộc/không thuộc đối với các giao dịch.

#### Chức năng

Công bố gốc tin cậy của hệ thống: Công bố chứng thư số gốc, các chứng thư chéo, chứng thư cầu nối, danh sách tin cậy, …

Công bố thông tin chứng thực cặp khóa: Công bố chứng thư số CA trung gian, chứng thư số người dùng cuối. Công bố CARL và CRL.

Cung cấp các dịch vụ hỗ trợ đảm bảo an toàn cho các giao dịch điện tử.

#### Nhiệm vụ

Duy trì kho chứng thư số: chứa các chứng thư số gốc, chứng thư số CA, chứng thư số chéo, cầu nối, danh sách tin cậy… chứng thư số thực thể cuối; CARL và CRL.

Cung cấp dịch vụ kiểm tra trạng thái chứng thư trực tuyến OCSP, SCVP.

Cung cấp dịch vụ cấp dấu thời gian cho mỗi giao dịch.

#### Thành phần

Kho chứng thư số: chứa tất cả các chứng thư số, CARL, CRL do tổ chức chứng thực tạo ra, điểm tin cậy của hệ thống.

Các dịch vụ hỗ trợ kiểm tra trạng thái chứng thư số: OCSP, SCVP...

Dịch vụ dấu thời gian an toàn: tạo dấu thời gian gắn kèm với dữ liệu thỏa mãn các tính chất xác thực và toàn vẹn.

### *Thực thể cuối*

Thực thể cuối là những đối tượng (như người dùng, thiết bị, phần mềm) sử dụng cặp khóa đã chứng thực để đảm bảo an toàn cho các giao dịch điện tử mình tham gia.

* Gửi các yêu cầu chứng thực (cấp mới, thay khóa, thu hồi, tạm dừng, treo…) tới RA và nhận kết quả yêu cầu từ RA.
* Sử dụng chứng thư số trong các giao dịch.

Phân loại thực thể cuối

* Chủ thể tin cậy và sử dụng chứng thư số - relying party.
* Chủ thể sở hữu chứng thư số (thuê bao) - subscriber.
* Thực thể cuối giao tiếp với hạ tầng (các dịch vụ của hạ tầng cung cấp) thông qua phần mềm người dùng cuối (client-side software) còn gọi là PKI-client (PKI-enable) software.

## **Chứng thư số và CRL**

### *Chứng thư số*

#### 1.5.1.1. Khái niệm

Chứng thư số là một dạng chứng thư điện tử được cấp bởi tổ chức cung cấp dịch vụ chứng thực chữ ký số. Chứng thư số có thể coi như giấy chứng minh nhân nhân (CMND) được sử dụng trong thế giới điện tử và mạng internet.

Loại chứng thư này được dùng như một công cụ điện tử giúp nhận diện cá nhân, máy chủ hoặc một số đối tượng khác. Nó gắn định danh đối tượng đó với một “khóa công khai” được cấp bởi tổ chức có thẩm quyền.

Chứng thư số chứa khóa công khai (public key), trong khi đó chữ ký số chứa khóa bí mật (private key). Chứng thư số và chữ ký số kết hợp lại sẽ tạo thành một cặp khóa. Ta có thể sử dụng cặp khóa này để ký số. Khóa bí mật của chữ ký số được lưu trữ trong 1 USB (gọi là Token USB hoặc SmartCard) giúp các khóa này tránh bị sao chép hoặc bị tấn công bởi virus khiến hỏng hóc và mất dữ liệu). Chứng thư số chứa public key và các thông tin người dùng theo chuẩn X.509.

Trong chứng minh thư luôn có những thông tin như họ tên, ngày tháng năm sinh, quê quán, hộ khẩu thường trú… Chứng thư số cũng vậy, tuy nhiên dữ liệu trên chứng thư số không phải những thông tin như trên mà bao gồm các nội dung dữ liệu sau đây:

* Tên của người dùng ở định dạng tên phân biệt (DN). DN chỉ định tên của người dùng và bất kỳ thuộc tính bổ sung nào cần thiết để nhận dạng duy nhất người dùng (ví dụ: DN có thể chứa số nhân viên của người dùng).
* Khóa công khai của người dùng. Khóa công khai là bắt buộc để những người khác có thể mã hóa cho người dùng hoặc xác minh chữ ký số của người dùng.
* Thời hạn hiệu lực (hoặc thời gian tồn tại) của chứng chỉ: ngày bắt đầu và ngày kết thúc.
* Các hoạt động cụ thể mà khóa công khai sẽ được sử dụng để mã hóa dữ liệu, xác minh chữ ký số hay cả hai.

Chữ ký của CA trên chứng chỉ có nghĩa là mọi hành vi giả mạo nội dung của chứng chỉ sẽ dễ dàng bị phát hiện. Chữ ký của CA trên chứng chỉ giống như con dấu phát hiện giả mạo trên một chai thuốc - bất kỳ hành vi giả mạo nào đối với nội dung của chứng chỉ đều dễ dàng bị phát hiện. Miễn là có thể xác minh được chữ ký của CA trên chứng chỉ thì chứng chỉ đó có tính toàn vẹn. Vì tính toàn vẹn của chứng chỉ có thể được xác định bằng cách xác minh chữ ký của CA, chứng chỉ vốn đã an toàn và có thể được phân phối theo cách hoàn toàn công khai (ví dụ: thông qua hệ thống thư mục có thể truy cập công khai).Tất cả những thông tin này sẽ được sử dụng để kê khai thuế qua mạng, khai báo hải quan, dịch vụ công, phát hành hóa đơn điện tử,… và thực hiện các giao dịch trực tuyến khác.

#### Đặc điểm

Chứng thư số được CA ký số, nên chúng tự bảo vệ đối với tính toàn vẹn và chỉ chứa những thông tin công khai nên có thể được phát tán một cách tự do.

Chứng thư số chứa (gắn kết) tên thuê bao và khóa công khai, nó giải quyết được hai vấn đề của mật mã khóa công khai khi áp dụng vào thực tiễn (xác thực cặp khóa, chống chối bỏ).

#### Phân loại

Căn cứ vào đối tượng của chứng thư số có thể phân loại chứng thư số thực thể cuối và chứng thư số CA.

* Chứng thư số thực thể cuối: là loại chứng thư số do một CA phát hành cho thực thể cuối. Chứng thư thực thể cuối này không dùng để phát hành chứng thư khác.
* Chứng thư số CA: là chứng thư số do một CA phát hành cho một CA được phân biệt bởi trường Basic constraint. Chứng thư số CA này có thể dùng để phát hành chứng thư khác. Chứng thư số CA có thể là
  + Chứng thư số tự phát hành: là một loại chứng thư số đặc biệt trong đó chủ thể phát hành (issuer) và chủ thể sở hữu (subject) là một (giống nhau). Loại chứng thư số này được tạo ra cho những mục đích đặc biệt như kiểm tra hợp lệ cặp khóa mới khi khóa của CA hết hạn.
  + Chứng thư số tự ký - chứng thư số của RootCA: là một loại chứng thư số tự phát hành. Trong đó khóa riêng ký chứng thư số tương ứng với khóa công khai trong chứng thư số.
  + Chứng thư số chéo: là một loại chứng thư số mà chủ thể phát hành và chủ thể sở hữu là các CA khác nhau. Chứng thư số chéo dùng để xây dựng mối quan hệ tin cậy giữa các CA.

Căn cứ vào mục đích sử dụng, chứng thư số còn có thể được phân thành chứng thư số khóa công khai, chứng thư số đủ điều kiện, và chứng thư số thuộc tính, CVC.

* Chứng thư số khóa công khai (Public Key Certificate)
* Chứng thư số thuộc tính (Attribute Certificate): chứng thư số chứa các thông tin như: thành viên nhóm, vai trò-role, mức an toàn … và các thông tin về sự cho phép kết hợp với thông tin về chủ sở hữu chứng thư. Chứng thư số thuộc tính không chứa khóa công khai và dùng để cấp phép thực hiện hành vi đối với mỗi chủ thể (Authorization).
* Chứng thư số đủ điều kiện (Qualified Certificate): là một loại chứng thư số được dùng để xác định một người với mức an toàn nhất định. Chứng thư số đủ điều kiện thường gồm các thông tin xác định chủ thể như, họ tên, ngày sinh, ,.. thông tin sinh trắc (ảnh, vân tay..).
* Card Verifiable Certificates (CVC): được thiết kế để có thể xử lý bởi các thiết bị có khả năng tính toán hạn chế như thẻ thông minh. Sử dụng cấu trúc (TLV) với các trường cố định. (mỗi trường trong chứng chỉ có độ dài cố định hoặc tối đa và mỗi trường theo thứ tự được xác định rõ).

#### Chứng thư X509

##### *Giới thiệu*

Chứng nhận X.509 là chứng nhận khóa công khai phổ biến nhất. Hiệp hội Viễn thông quốc tế (International Telecommunications Union – ITU) đã chỉ định chuẩn X.509 vào năm 1988. Đây là định dạng phiên bản 1 của chuẩn X.509. Vào năm 1993, phiên bản 2 của chuẩn X.509 được phát hành với 2 trường tên nhận dạng duy nhất được bổ sung. Phiên bản 3 của chuẩn X.509 được bổ sung thêm trường mở rộng đã phát hành vào năm 1997.

Một chứng nhận khóa công khai kết buộc một khóa công khai với sự nhận diện của một người (hoặc một thiết bị). Khóa công khai và tên thực thể sở hữu khóa này là hai mục quan trọng trong một chứng nhận.

Về tổng quan ta đúc kết được các thông tin X509 gồm có các thành phần sau:

1. Version: Chỉ định phiên bản của chứng nhận X.509.
2. Serial Number: Số loạt phát hành được gán bởi CA. Mỗi CA nên gán một mã số loạt duy nhất cho mỗi giấy chứng nhận mà nó phát hành.
3. Signature Algorithm: Thuật toán chữ ký chỉ rõ thuật toán mã hóa được CA sử dụng để ký giấy chứng nhận. Trong chứng nhận X.509 thường là sự kết hợp giữa thuật toán băm (chẳng hạn như MD5 hoặc SHA-1) và thuật toán khóa công khai (chẳng hạn như RSA).
4. Issuer Name: Tên tổ chức CA phát hành giấy chứng nhận, đây là một tên phân biệt theo chuẩn X.500 (xem Phụ lục A). Hai CA không được sử dụng cùng một tên phát hành.
5. Validity Period: Trường này bao gồm 2 giá trị chỉ định khoảng thời gian mà giấy chứng nhận có hiệu lực. Hai phần của trường này là not-before và not-after, các giá trị thời gian này được đo theo chuẩn thời gian Quốc tế, chính xác đến từng giây.
6. Not-before chỉ định thời gian mà chứng nhận này bắt đầu có hiệu lực.
7. Not-after chỉ định thời gian mà chứng nhận hết hiệu lực.
8. Subject Name: là một X.500 DN, xác định đối tượng sở hữu giấy chứng nhận mà cũng là sở hữu của khóa công khai. Một CA không thể phát hành 2 giấy chứng nhận có cùng một Subject Name.
9. Public key: Xác định thuật toán của khóa công khai (như RSA) và chứa khóa công khai được định dạng tùy vào kiểu của nó.
10. Issuer Unique ID và Subject Unique ID: Hai trường này được giới thiệu trong X.509 phiên bản 2, được dùng để xác định hai tổ chức CA hoặc hai chủ thể khi chúng có cùng DN. RFC 2459 đề nghị không nên sử dụng 2 trường này.
11. Extensions: Chứa các thông tin bổ sung cần thiết mà người thao tác CA muốn đặt vào chứng nhận. Trường này được giới thiệu trong X.509 phiên bản 3.
12. Signature: Đây là chữ ký điện tử được tổ chức CA áp dụng. Tổ chức CA sử dụng khóa bí mật có kiểu quy định trong trường thuật toán chữ ký. Chữ ký bao gồm tất cả các phần khác trong giấy chứng nhận. Do đó, tổ chức CA chứng nhận cho tất cả các thông tin khác trong giấy chứng nhận chứ không chỉ cho tên chủ thể và khóa công khai. Những phần mở rộng của tên tập tin phổ biến cho chứng nhận X.509 bao gồm:
    1. Phần mở rộng tệp tin cer: chứng nhận được mã hóa theo luật mã hóa tiêu chuẩn (Canonical Encoding Rules – CER).
    2. Phần mở rộng tệp tin der: chứng nhận được mã hóa theo luật mã hóa phân biệt (Distinguished Encoding Rules – DER).
    3. Phần mở rộng tệp tin pem (Privacy-Enhanced Electronic Mail): định dạng mã hóa được sử dụng để lưu trữ các chứng nhận và khóa. Một tập tin được định dạng với chuẩn này có thể chứa các khóa bí mật (RSA và DSA), khóa công khai (RSA và DSA) và các chứng nhận X509. Định dạng này lưu trữ dữ liệu ở định dạng DER được mã hóa cơ sở 64, nằm giữa "-----BEGIN CERTIFICATE-----" và "-----END CERTIFICATE-----", phù hợp cho việc trao đổi ở dạng văn bản giữa các hệ thống.
    4. Phần mở rộng tệp tin p7b và p7c: PKCS #7 là một định dạng mã hóa cho việc lưu trữ một chứng nhận số và chuỗi chứng nhận của nó dưới dạng các ký tự ASCII. Định dạng này được sử dụng bởi CA để trả về các chứng nhận được phát hành cùng với chuỗi chứng nhận. Định dạng này có có thể được sử dụng như đầu vào cho yêu cầu gia hạn chứng nhận đến một CA.
    5. Phần mở rộng tệp tin pfx và p12: PKCS #12 là một định dạng mã hóa cho việc lưu trữ một chứng nhận số và kết hợp với khóa bí mật dưới dạng các ký tự ASCII. Định dạng này luôn luôn được trả về bởi CA khi CA phát sinh các khóa và phát hành chứng nhận đồng thời.

Đi sâu vào chi tiết có 3 phiên bản của chứng chỉ số được dùng trong một hệ tầng PKI là version 1,2, 3.

##### *Chứng thư X509 phiên bản 1*

Được định nghĩa vào năm 1988, X.509 version 1 giờ đây hầu như không còn được sử dụng nữa. Định dạng của loại chứng chỉ này được thể hiện như sau.

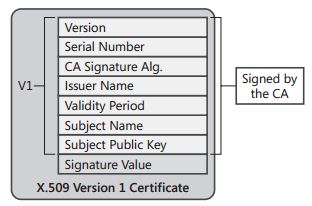


Figure 7. Chứng thư số X.509 phiên bản 1.

Một chứng chỉ X.509 version 1 bao gồm các trường sau:

1. Version: chứa giá trị cho biết đây là chứng chỉ X.509 version 1.
2. Serial Number: cung cấp một mã số nhận dạng duy nhất cho mỗi chứng chỉ được phát hành bởi CA.
3. CA Signature Algorithm: tên của thuật toán mà CA sử dụng để ký lên nội dung của chứng chỉ số.
4. Issuer Name: tên phân biệt (distinguished name) của CA phát hành chứng chỉ. Thường thì tên phân biệt này được biểu diễn theo chuẩn X.500 hoặc định dạng theo đặc tả của X.509 và RFC 3280.
5. Validity Period: khoảng thời gian mà chứng chỉ được xem là còn hiệu lực, bao gồm 2 trường là: Valid From và Valid To.
6. Subject Name: tên của máy tính, người dùng, thiết bị mạng sở hữu chứng chỉ. Thường thì tên chủ thể này được biểu diễn theo chuẩn X.500 hoặc định dạng theo đặc tả của X.509, nhưng cũng có thể bao gồm các định dạng tên khác như được mô tả trong RFC 822.
7. Subject Public Key Info: khóa công khai của đối tượng nắm giữ chứng chỉ. Khóa công khai này được gửi tới CA trong một thông điệp yêu cầu cấp chứng chỉ (certificate request) và cũng được bao gồm trong nội dung của chứng chỉ được phát hành sau đó. Trường này cũng chứa nhận dạng của thuật toán được dùng để tạo cặp khóa công khai và khóa bí mật được liên kết với chứng chỉ.
8. Signature Value: chứa giá trị của chữ ký.

Các trường Issuer Name và Subject Name được cấu trúc để các chứng chỉ có thể được tổ chức thành một chuỗi các chứng chỉ mà bắt đầu bằng chứng chỉ được cấp cho người dùng, máy tính, thiết bị mạng, hoặc dịch vụ và kết thúc bằng chứng chỉ gốc của CA.

##### *Chứng thư X509 phiên bản 2*

Mặc dù chứng chỉ X.509 version 1 cung cấp khá đầy đủ những thông tin cơ bản về người nắm giữa chứng chỉ nhưng nó lại có ít thông tin về tổ chức cấp phát chứng chỉ khi chỉ bao gồm Issuer Name, CA Signature Algorithm và Signature Value. Điều này không giúp dự phòng trong trường hợp CA được thay mới.

Khi chứng chỉ của CA được thay mới, trường Issuer Name trong cả 2 chứng chỉ mới và cũ đều như nhau. Tương tự, có thể có một tổ chức khác muốn tạo một CA có trường Issuer Name trong chứng chỉ giống như vậy. Giải quyết vấn đề này để có thể sử dụng lại Issuer Name thì chứng chỉ X.509 version 2 đã được giới thiệu vào năm 1993. Trong định dạng của nó có thêm 2 trường mới như được thể hiện trong hình sau đây.

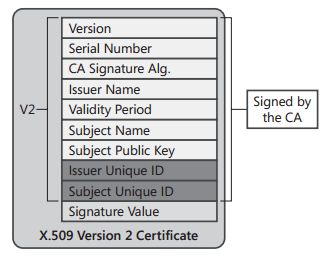


Figure 8. Chứng thư số X.509 phiên bản 2.

Hai trường mới được bổ sung là

* Issuer Unique ID: là một trường không bắt buộc, chứa chuỗi giá trị ở hệ 16, mang tính duy nhất và dành để nhận dạng CA. Khi CA thay mới chứng chỉ của chính nó, một Issuer Unique ID mới được khởi tạo cho chứng chỉ đó.
* Subject Unique ID: là một trường không bắt buộc, chứa chuỗi giá trị ở hệ 16, mang tính duy nhất và dùng để nhận dạng chủ thể của chứng chỉ. Nếu chủ thể này cũng chính là CA thì trường này sẽ giống với Issuer Unique ID.

Ngoài việc đưa vào 2 trường mới ở trên thì trường Version trong chứng chỉ X.509 version 2 có giá trị là 2 để chỉ ra phiên bản của chứng chỉ.

Các trường Issuer Unique ID và Subject Unique ID đã cải tiến quá trình xâu chuỗi chứng chỉ. Giờ đây việc tìm kiếm chứng chỉ của của CA sẽ là so khớp Issuer Name trong chứng chỉ được cấp phát với Subject Name trong chứng chỉ của CA và thực hiện thêm một bước kiểm tra thứ hai là so khớp Issuer Unique ID trong chứng chỉ được cấp phát với Subject Unique ID trong chứng chỉ của CA.

Bước so khớp thứ hai này cho phép phân biệt giữa các chứng chỉ của cùng một CA khi CA đó làm mới lại chứng chỉ của chính nó. Cách này cũng giúp phân biệt giữa các CA khác nhau nhưng trùng Subject Name.

Mặc dù định dạng X.509 version có cải tiến hơn version 1 nhưng chuẩn này cũng không còn được áp dụng rộng rãi. Và thực tế thì trong RFC 3280 đã khuyến cáo là bỏ qua việc sử dụng 2 trường mới trên của X.509 version 2 do lo ngại có thể có sự xung đột xảy ra nếu như hai chứng chỉ có cùng Subject Name và Subject Unique ID.

##### *Chứng thư X509 phiên bản 3*

Được ra đời vào năm 1996, định dạng X.509 version 3 được bổ sung thêm các phần mở rộng (extension) để khắc phục các vấn đề liên quan tới việc so khớp Issuer Unique ID và Subject Unique ID cũng như là các vấn đề về xác thực chứng chỉ. Một chứng chỉ X.509 version 3 co thể chứa một hoặc nhiều extension, như được thể hiện trong hình dưới đây:

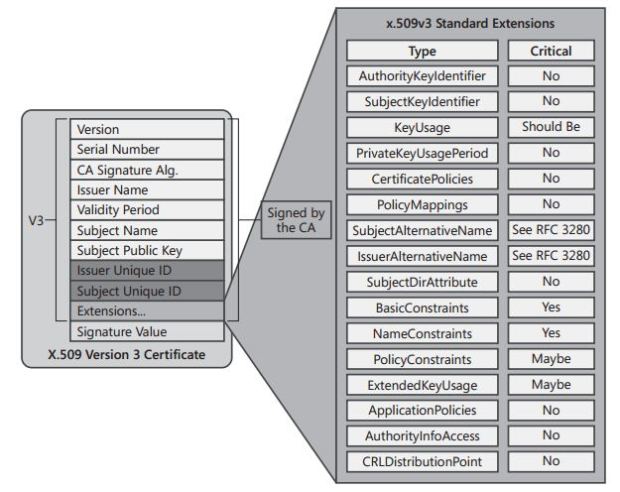


Figure 9. Chứng thư số X.509 phiên bản 3.

Mỗi extension trong chứng chỉ X.509 version 3 gồm 3 phần:

* Extension Identifier: là một mã nhận dạng đối tượng (Object Identifier – OID) cho biết kiểu định dạng và các định nghĩa của extension.
* Criticality Flag: là một dấu hiệu cho biết thông tin trong extension có quan trọng (critical) hay không. Nếu một ứng dụng không thể nhận diện được trạng thái critical của extension hoặc extension không hề chứa giá trị nào thì chứng chỉ đó không thể được chấp nhận hoặc được sử dụng. Nếu mục criticality flag này không được thiết lập thì một có thể sử dụng chứng chỉ ngay cả khi ứng dụng đó không nhận diện được extension.
* Extension Value: là giá trị được gán cho extension. Nó phụ thuộc vào từng extension cụ thể.

Trong một chứng chỉ X.509 version 3 thì các extension sau có thể có là:

1. Authority Key Identifier: extension này có thể chứa một hoặc hai giá trị, chúng có thể là:

* Subject Name của CA và Serial Number của chứng chỉ của CA mà đã cấp phát chứng chỉ này.
* Giá trị băm của khóa công khai của chứng chỉ của CA mà đã cấp phát chứng chỉ này.

1. Subject Key Identifier: extension này chứa giá trị băm của khóa công khai của chứng chỉ.
2. Key Usage: một CA, người dùng, máy tính, thiết bị mạng hoặc dịch vụ có thể sở hữu nhiều hơn một chứng chỉ. Extension này định nghĩa các dịch vụ bảo mật mà một chứng chỉ có thể cung cấp như:
   * Digital Signature: khóa công khai có thể được dùng để kiểm tra chữ ký. Khóa này cũng được sử dụng để xác thực máy khách và xác minh nguồn gốc của dữ liệu.
   * Non-Repudiation: khóa công khai có thể được dùng để xác minh nhận dạng của người ký, ngăn chặn người ký này từ chối rằng họ không hề ký lên thông điệp hoặc đối tượng nào đó.
   * Key Encipherment: khóa công khai có thể được dùng để trao đổi khóa, vú dụ như đối xứng (hoặc khóa phiên). Giá trị này được dùng khi một khóa RSA được dùng cho việc quản lý khóa.
   * Data Encipherment: khóa công khai có thể được dùng để mã hóa dữ liệu một cách trực tiếp thay vì phải trao đổi một khóa đối xứng (hay khóa phiên) để mã hóa dữ liệu.
   * Key Agreement: khóa công khai có thể được dùng để trao đổi khóa, ví dự như khóa đối xứng. Giá trị này được dùng khi một khóa Diffie-Hellman được dùng cho việc quản lý khóa.
   * Key Cert Sign: khóa công khai có thể được dùng để kiểm tra chữ ký của chứng chỉ số.
   * CRL Sign: khóa công khai có thể được dùng để kiểm tra chữ ký của CRL (danh sách chứa các chứng chỉ bị thu hồi).
   * Encipher Only: giá trị này được dùng kết hợp với các extension Key Agreement và Key Usage. Kết quả là khóa đối xứng chỉ có thể được dùng để mã hóa dữ liệu.
   * Decipher Only: giá trị này được dùng kết hợp với các extension Key Agreement và Key Usage. Kết quả là khóa đối xứng chỉ có thể được dùng để mã hóa dữ liệu.
3. Private Key Usage Period: extension này cho phép khóa bí mật có khoảng thời gian hiệu lực khác so với khoảng thời gian hiệu lực của chứng chỉ. Giá trị này có thể được đặt ngắn hơn so với khoảng thời gian hiệu lực của chứng chỉ. Điều này giúp khóa bí mật có thể được dùng để ký lên các tài liệu trong một khoảng thời gian ngắn (ví dụ, một năm) trong khi khóa công khai có thể được dùng để xác minh chữ ký trong khoảng thời gian hiệu lực của chứng chỉ là 5 năm.
4. Certificate Policies: extension này mô tả các chính sách và thủ tục được dùng để xác minh chủ thể của chứng chỉ trước khi chứng chỉ được cấp phát. Các chính sách chứng chỉ được đại diện bởi các OID. Ngoài ra, một chính sách chứng chỉ có thể bao gồm một đường dẫn (URL) tới trang web mô tả nội dung của chính sách và thủ tục.
5. Policy Mappings: extension này cho phép chuyển dịch thông tin về chính sách giữa hai tổ chức. Ví dụ, thử tưởng tượng rằng một tổ chức định nghĩa một chính sách chứng chỉ có tên là Management Signing mà trong đó các chứng chỉ được dùng để ký lên một lượng lớn các đơn đặt hàng. Một tổ chức khác có thể có một chính sách chứng chỉ tên là Large Orders mà cũng được dùng để ký lên một lượng lớn các đơn đặt hàng. Khi đó, Policy Mapping cho phép hai chính sách chứng chỉ này được đánh giá ngang nhau.
6. Subject Alternative Name: extension này cung cấp một danh sách các tên thay thế cho chủ thể của chứng chỉ. Trong khi định dạng cho Subject Name thường tuân theo chuẩn X.500 thì Subject Alternative Name cho phép thể hiện theo các dạng khác như User Principal Name (UPN), địa chỉ email, địa chỉ IP hoặc tên miền (DNS).
7. Issuer Alternative Name: extension này cung cấp một danh sách các tên thay thế cho CA. Mặc dù thường không được áp dụng nhưng extension này có thể chứa địa chỉ email của CA.
8. Subject Dir Attribute: extension này có thể bao gồm bất kỳ thuộc tính nào từ danh mục LDAP hoặc X.500 của tổ chức, ví dụ, thuộc tính country. Extension này có thể chứa nhiều thuộc tinh và với mỗi thuộc tính phải gồm OID và giá trị tương ứng của nó.
9. Basic Constraints: extension này cho biết chứng chỉ có phải của CA hay của các chủ thể như người dùng, máy tính, thiết bị, dịch vụ. Ngoài ra, extension này còn bao gồm một rằng buộc về độ dài của đường dẫn mà giới hạn số lượng các CA thứ cấp (subordinate CA) có thể tồn tại bên dưới CA mà cấp phát chứng chỉ này.
10. Name Constraints: extension này cho phép một tổ chức chỉ định không gian tên (namespace) nào được phép hoặc không được phép sử dụng trong chứng chỉ.
11. Policy Constraints: extension này có thể có trong các chứng chỉ của CA. Nó có thể ngăn cấm Policy Mapping giữa các CA hoặc yêu cầu mỗi chứng chỉ trong chuỗi chứng chỉ phải bao gồm một OID của chính sách chứng chỉ.
12. Enhanced Key Usage: extension này cho biết khóa công khai của chứng chỉ có thể được sử dụng như thế nào. Những cái này không có trong extension Key Usage. Ví dụ, Client Authentication (có OID là 1.3.6.1.5.5.7.3.2), Server Authentication (có OID là 1.3.6.1.5.5.7.3.1), và Secure E-mail (có OID là 1.3.6.1.5.5.7.3.4). Khi ứng dụng nhận được một chứng chỉ, nó có thể yêu cầu sự có mặt của một OID trong các OID kể trên.
13. CRL Distribution Points: extension này chứa một hoặc nhiều URL dẫn tới tập tin chứa danh sách các chứng chỉ đã bị thu hồi (CRL) được phát hành bởi CA. Nếu việc kiểm tra trạng thái thu hồi của chứng chỉ được cho phép thì một ứng dụng sẽ sử dụng các URL này để tải về phiên bản cập nhật của CRL. Các URL có thể sử dụng một trong các giao thức như HTTP, LDAP, FTP, File.
14. Authority Information Access: extension này có thể chứa một hoặc nhiều URL dẫn tới chứng chỉ của CA. Một ứng dụng sử dụng URL này để tải về chứng chỉ của CA khi xây dựng chuỗi chứng chỉ nếu như nó không có sẵn trong bộ nhớ đệm của ứng dụng.
15. Freshest CRL: extension này chứa một hoặc nhiều URL dẫn tới delta CRL do CA phát hành. Delta CRL chỉ chứa các chứng chỉ bị thu hồi kể từ lần cuối base CRl được phát hành. Nếu việc kiểm tra trạng thái thu hồi của chứng chỉ được cho phép thì một ứng dụng sẽ sử dụng các URL này để tải về phiên bản cập nhật của delta CRL. Các URL có thể sử dụng một trong các giao thức như HTTP, LDAP, FTP, File.
16. Subject Information Access: extension này chứa thông tin cho biết cách thức để truy cập tới các các chi tiết khác về chủ thể của chứng chỉ. Nếu đây là chứng chỉ của CA thì thông tin này có thể bao gồm các chi tiết về các dịch vụ xác minh chứng chỉ hay chính sách của CA. Nếu chứng chỉ được cấp cho người dùng, máy tính, thiết bị mạng, hoặc dịch vụ thì extension này có thể chứa thông tin về các dịch vụ được các chủ thể này cung cấp và cách thức để truy cập tới các dịch vụ đó.

### *Danh sách thu hồi chứng thư số CLR*

#### Khái niệm thu hồi chứng thư

Thu hồi chứng thư là việc làm mất hiệu lực (không còn hợp lệ) của chứng thư trước khi chứng thư hết hạn tự nhiên.

#### Lý do thu hồi chứng thư

Mỗi chứng thư khi phát hành đều được thiết lập một khoảng thời gian hợp lệ. Tuy nhiên, nhiều tình huống nảy sinh sau khi phát hành chứng thư làm cho chứng thư đã phát hành trở nên không hợp lệ nữa, kể cả khi chứng thư chưa hết hạn. Một số tình huống thường gặp:

* Thay đổi trạng thái của chủ thể (nghỉ việc…).
* Thay đổi các tham số chứng thư số (chuyển công tác, gia hạn).
* Nghi ngờ lộ khoá riêng...

Cần có các phương pháp hiệu quả và tin cậy để thu hồi một chứng thư số trước khi nó hết hạn tự nhiên.

#### Sự cần thiết thu hồi chứng thư số

Để đảm bảo an toàn cho các giao dịch, mỗi chứng thư số phải trải qua một quá trình kiểm tra tính hợp lệ trước khi có thể được sử dụng. Một bước trong quá trình kiểm tra là nhằm xác định xem liệu chứng thư số cần đánh giá đã bị thu hồi hay chưa.

Trách nhiệm tạo và công bố thông tin thu hồi chứng thư số:

* CA có trách nhiệm tạo ra thông tin thu hồi chứng thư số.
* Việc công bố thông tin thu hồi chứng thư số có thể do CA hoặc một bên thứ 3 tin cậy thực hiện ở một dạng này hay một dạng khác.

#### Khái niệm danh sách thu hồi chứng thư số CLR

Danh sách thu hồi chứng thư số (CRL) là một cấu trúc dữ liệu chứa danh sách các chứng thư số đã bị thu hồi. CRL thường được ký và phát hành bởi chủ thể phát hành các chứng thư số (đã bị thu hồi), được liệt kê trong CRL.

CRL chỉ chứa các chứng thư số chưa hết hạn nhưng bị thu hồi. Các chứng thư số hết hạn, đã bị thu hồi sẽ bị loại khỏi CRL.

CRL có thể chứa các chứng thư số bị thu hồi của một hay nhiều CA.

CRL thường do CA phát hành chứng thư số tạo ra. Nếu CA phát hành chứng thư số ủy quyền cho một bên tin cậy thứ ba (trust third party) phát hành CRL thì CRL này gọi là Indirect CRL (gián tiếp).

Mỗi CRL có một phạm vi (mục đích) nhất định. Phạm vi CRL định nghĩa tập các chứng thư số có thể nằm trong một CRL nhất định.

Ví dụ CRL chứa tất cả các chứng thư số do một CA phát hành, CRL chứa tất cả các chứng thư số bị thu hồi vì một lý do xác định.

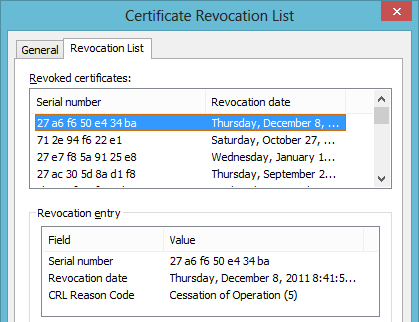


Figure 10. Ảnh minh họa CRL.

Mỗi một mục (entry) trong CRL tương ứng với một certificate và thường gồm 3 thông tin sau:

* Serial number của certificate.
* Thời điểm bị thu hồi.
* Lý do thu hồi.

#### Phân loại CRL

Căn cứ vào nội dung CRL có thể phân thành 2 loại sau:

* CARL (Certification Authority Revocation List) là một loại danh sách thu hồi chứng thư chứa tất cả các chứng thư số cấp cho CA đã bị thu hồi của một CA nhất định. Chủ thể phát hành CARL là một CA cấp cao hơn. CARL không chứa thông tin thu hồi chứng thư của thực thể cuối. CARL không áp dụng với các CA tự ký (rootCA).
* EPRL – (End-entity Public-key Certificate Revocation List) là một loại danh sách thu hồi chứng thư số chỉ chứa các chứng thư đã thu hồi của thực thể cuối. Một EPRL có thể chứa tất cả thông tin thu hồi của một miền PKI, hoặc nó có thể được phân chia theo nhiều cách khác nhau.

#### Cấu trúc CLR

Các tùy chọn cho mỗi mục trong danh sách thu hồi (per entry)

* Reason Code (mã lý do): mã nguyên nhân chứng thư số bị thu hồi: lộ khoá bí mật, thay đổi thông tin chủ thể sở hữu ....
* Certificate Issuer (chủ thể phát hành chứng thư số): là tên của chủ thể phát hành chứng thư số (đối với các CRL gián tiếp).
* Hold Instruction Code (mã chỉ thị treo): được sử dụng để treo tạm thời một chứng thư số. Chứng thư số bị treo sau đó có thể được cài đặt lại hoặc thu hồi.
* Ngày không hợp lệ (Invalidity Date): xác định thời điểm chứng thư không còn hợp lệ.

Các tùy chọn mở rộng của danh sách thu hồi (CRL extention):

* Authority Key Identifier (Định danh khoá của thẩm quyền): dùng để xác định duy nhất khoá kiểm tra chữ ký trên CRL. Giá trị này được sử dụng khi chủ thể phát hành CRL có nhiều khóa ký cùng tồn tại. (CRL Number + isuer name)
* Issue Alternative Name (Tên khác của chủ thể phát hành): chứa một danh sách các dạng tên khác nhau dùng để xác định chủ thể phát hành CRL (vd, địa chỉ e-mail, địa chỉ IP, URI...).
* CRL Number (số CRL): số thứ tự duy nhất cho mỗi CRL.
* Issuing Distribution Point (Điểm phân phối phát hành) chứa các thông tin về một CRL Distribution Point (URI), scope và các kiểu chứng thư chứa trong CRL (ví dụ, chỉ có chứng thư của người dùng cuối, chỉ có các chứng thư của CA, hoặc chỉ có các chứng thư được thu hồi theo một nguyên nhân đặc biệt). Trường này nếu chứa thông tin về một CRL Distribution Point (URI) thì nó phải giống với một trong các trường ở CRL Distribution Point trong chứng thư số. Nếu không chứa giá trị nào thì bản thân CRL phải chứa các chứng thư bị thu hồi.

#### Simple CLR

Simple (Complete) CRL là CRL có thể chứa tất cả các chứng thư số đã thu hồi của 1 CA nhất định.

Ưu điểm: đơn giản, dễ cài đặt.

Nhược điểm: Kích thước của CRL có thể rất lớn (tốn băng thông mạng, không phù hợp với các ứng dụng nhỏ-mobilephone). Khó công bố tức thời.

#### Partioned CLR

Partitioned CRL là việc chia đưa các thông tin thu hồi chứng thư của 1 CA nhất định vào nhiều CRL.

Việc phân chia thông tin thu hồi vào nhiều CRL được thực hiện thông qua trường CRL Distribution Points trong chứng thư số và trường Issuing Distribution Point. Mỗi một CRL Distribution Point chứa một con trỏ tới vị trí của CRL partition.

Ưu điểm: thay đổi kích thước lớn nhất của CRL, chia các thông tin thu hồi thành các nhóm xác định.

Nhược điểm: Khi đã xác định các phần của CRL không thể thay đổi được (kém linh hoạt)

#### Delta CLR

Delta CRL là một loại CRL được thiết kế với mục tiêu giảm kích thước CRL mà các phần mềm client cần phải tải về.

Đầu tiên và định kỳ sau một khoảng thời gian xác định, toàn bộ các chứng thư bị thu hồi được đặt vào Base CRL (Simple CRL).

Định kỳ sau một khoảng thời gian ngắn hơn, các thông tin về thu hồi chứng thư tính từ thời điểm phát hành Base CRL được đưa vào Delta CRL.

Base CRL + Delta CRL là toàn bộ chứng thư đã bị thu hồi. Cả hai phải có chung scope.

Giá trị Base CRL number phải nhỏ hơn giá trị Delta CRL number.

## **Hoạt động của hệ thống PKI**

### *1.6.1. Chứng thực cặp khóa*

* Phát hành, tạo mới (Enrollments): người dùng gửi yêu cầu đăng ký, CA cấp chứng thư số cho thuê bao.
* Gia hạn (Renewal): thay đổi thời hạn sử dụng của chứng thư khi chứng thư hết hạn.
* Tạm dừng/khôi phục(Suspend/Active): treo/kích hoạt (tạm dừng/khôi phục) hiệu lực của chứng thư trong một khoảng thời gian nhất định: tạm thời làm mất hiệu lực của chứng thư số.
* Thu hồi(Revocation): làm mất hiệu lực chứng thư trước khi hết hạn tự nhiên

### *Sử dụng cặp khóa*

Người dùng sử dụng chứng thư số để ký số, dùng để mã mật, dùng để xác thực, dùng để chống chối bỏ.

#### Dùng ký số

Khi người dùng muốn xác nhận với một cá nhân, nhóm người hoặc lớn hơn là một tổ chức biết rằng bản thân đủ tỉnh táo để hiểu thông tin mà chính bản thân muốn mang lại cho đối phương thì người dùng sẽ sử dụng khóa bí mật để ký vào dữ liệu. Quá trình ký số được tiến triển một cách trình tự là băm dữ liệu cho kết quả đầu ra cố định để phù hợp cho bước mã hóa kế tiếp.

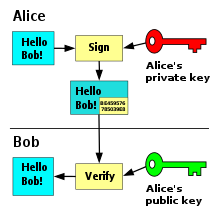


Figure 11. Mô hình ký số và xác minh truyền thống.

Thuật toán phổ biến được nhắc đến trong ký số là RSA sử dụng các khóa có kích thước 1024, 2048, 4096, ..., 16384 bit. RSA cũng hỗ trợ các khóa dài hơn (ví dụ: 65536 bit), nhưng hiệu suất quá chậm để sử dụng thực tế (một số hoạt động có thể mất vài phút hoặc thậm chí hàng giờ). Đối với mức bảo mật 128 bit, cần có khóa 3072 bit.

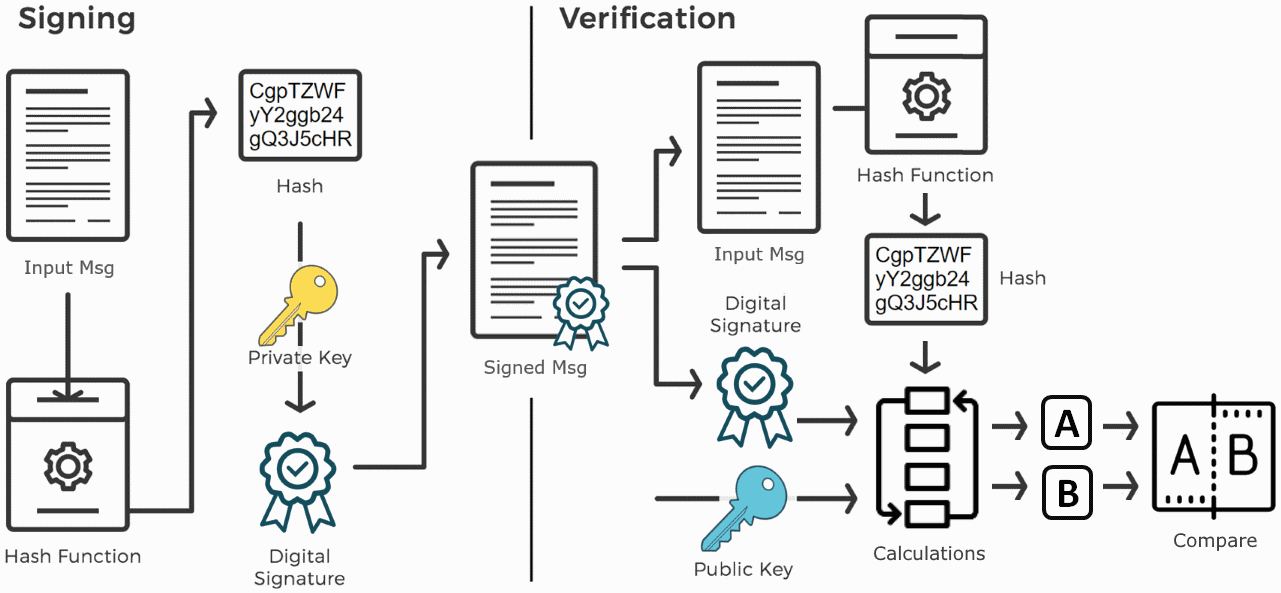


Figure 12. Mô tả chi tiết mô hình ký số và xác minh.

#### Dùng xác thực

Chứng thực (authentication) là quá trình một server xác định xem đối phương là ai. Phương pháp chứng thực đơn giản nhất mà người dùng sử dụng hằng ngày được gọi là “đăng nhập” (log in). Hằng ngày, người dùng nhập một mật khẩu tĩnh (không thay đổi trong vài tháng, hoặc cả đời), và máy chủ sẽ dò xem mật khẩu của người dùng có khớp với mật khẩu được lưu hay không. Tuy nhiên, một số người đặt mật khẩu rất dễ đoán, và mật khẩu có khả năng bị đánh cắp rất cao bằng những thủ thuật đơn giản. Do đó, trong một số các ứng dụng, một “mật khẩu điện tử” sẽ giúp việc chứng thực trở nên an toàn hơn. Phương thức chứng thực này thường được sử dụng trong SSH để đăng nhập vào máy khác (thường là máy chủ) mà không cần phải nhập mật khẩu.

Việc xác thực được tiến triển tuần tự thông qua bước giải mã dữ liệu được bảng băm và tiến hành so sánh kết quả đó với dữ liệu băm mà thông điệp người gửi trao cho người nhận.

Ghi chú: Máy chủ không lưu trực tiếp mật khẩu của người dùng, mà lưu bản hash của nó, để nếu khi hash bị lộ thì mật khẩu vẫn tương đối an toàn.

Thay vì một mật khẩu bình thường, người đăng nhập sẽ sinh ra một ra một bộ khóa công khai và bí mật. Sau đó, người đăng nhập sẽ đăng ký khóa công khai lên Máy chủ (khóa bí mật không bao giờ bị tiết lộ ra ngoài). Mỗi khi cần đăng nhập vào Máy chủ, các bước sau đây sẽ xảy ra:

Máy chủ sinh ra một số ngẫu nhiên a và gửi cho người đăng nhập.

Người đăng nhập sử dụng khóa bí mật để tạo mật văn b gửi lại cho máy chủ.

Máy chủ sử dụng khóa công khai (được đăng ký trước đó) để giải mã. Nếu kết quả sau khi giải mã lại bằng a, đó là bằng chứng cho việc người đăng nhập sở hữu đúng khóa bí mật.

Nếu một kẻ mạo nhận muốn đăng nhập vào máy chủ, sau khi được nhận a từ Máy chủ, hắn không có đúng khóa bí mật và gửi lại mật văn giả b’, thì khi server giải mã ra lại sẽ có a’ khác a và do đó hắn không thể đăng nhập. Mặt khác, hắn không thể đoán hay tái sử dụng mật văn b, bởi vì a là ngẫu nhiên và sẽ thay đổi cho mỗi lần đăng nhập khác nhau.

Một điều bất tiện khi sử dụng “mật khẩu điện tử” này là khóa bí mật chỉ hợp lệ trên một máy duy nhất (người nào đó có thể sao chép sang máy khác, nhưng điều đó làm khóa bí mật của người dùng gặp nhiều rủi ro bị tiết lộ hơn). Tuy nhiên, người dùng hoàn toàn có thể tạo một bộ khóa riêng cho mỗi máy và đăng ký nhiều khóa công khai cho một tài khoản, như vậy người dùng có thể đăng nhập ở bất kỳ máy nào.

Tuy nhiên, nếu kẻ tấn công nghe lén những gói tin của người dùng, mặc dù hắn không biết được mật khẩu, hắn vẫn có thể biết được những thông tin người dùng truyền sau khi người dùng đăng nhập, ví dụ lịch sử giao dịch ngân hàng của người dùng. Do đó, người dùng cần phải có một cách để bảo vệ đường truyền của người dùng.

#### Dùng để mã mật

Việc mã hóa và giải mã trong mã mật tương tự với ký và xác thực ở trên nhưng hoán đổi vai trò của khóa bí mật và khóa công khai.

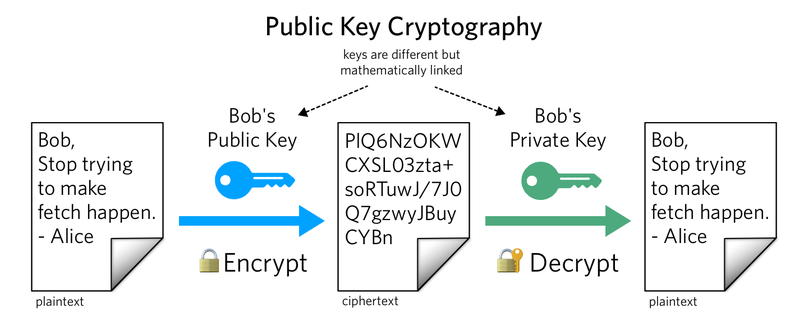


Figure 13. Mô hình sử dụng mã mật khóa công khai.

Việc mã hóa sẽ được thực hiện bởi người gửi với sự tham gia của khóa công khai, người nhận sẽ giải mã bằng khóa bí mật.

Nhưng có một vấn đề tồn tại rằng mật mã khóa công khai chỉ cho ra hiệu xuất tốc độ, tối ưu thời gian với lượng dữ liệu nhỏ, vì vậy trong thực tế chúng được sử dụng kết hợp với mật mã khóa bí mật, chúng đóng vai trò trong việc bảo vệ khóa đối xứng khi trung chuyển trong môi trường mạng.

## **Ứng dụng của PKI**

Nhóm các dịch vụ chính phủ điện tử eGovernment:

* Các dịch vụ - G2G: hệ thống trao đổi tài liệu và văn bản điện tử, điều hành tác nghiệp, hỗ trợ ra quyết định, hệ thống lưu trữ …
* Các dịch vụ - G2B: Hóa đơn , thuế, hải quan, cấp phép điện tử.
* Các dịch vụ - G2C: Bầu cử điện tử, hộ chiếu điện tử, chứng minh điện tử,…

Nhóm các dịch vụ của doanh nghiệp - B2C, B2B, B2G:

* Ngân hàng trực tuyến (Online Banking), Thanh toán trực tuyến, Tiền điện tử, ví điện tử, Kinh doanh chứng khoán trực tuyến, Đấu thầu trực tuyến, Bảo hiểm trực tuyến, Y tế, Giáo dục trực tuyến ….

## **Miền tin cậy và mở rộng miền tin cậy**

Khái niệm tin cậy (trust): là một phương diện trong mối quan hệ giữa hai thực thể A và B. A tin cậy B khi A giả sử rằng B sẽ hành động đúng như A mong đợi (X509 2000).

Miền tin cậy (trust domain): là một cộng đồng hoạt động dưới một điều khiển chung, cùng tiêu chí, tuân theo cùng quy tắc, chính sách trong đó, các cá thể thiết lập được các mối quan hệ, và mô hình hoạt động với một mức tin cậy cao nhất.

Miền tin cậy PKI là tập hợp người dung tin cậy vào tổ chức chứng thực.

Miền tin cậy PKI có thể được mở rộng bằng cách thiết lập các mối quan hệ tin cậy với nhau thông qua các phương pháp mở rộng miền tin cậy.

## **Các mô hình hoạt động của PKI**

### *Hierarchical PKI*

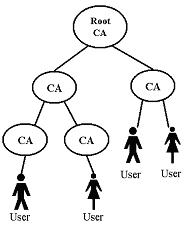


Figure 14. Mô hình Hiierarchical.

Đây là mô hình PKI được áp dụng rộng rãi trong các tổ chức lớn. Có một CA nằm ở cấp trên cùng gọi là root CA, tất cả các CA còn lại là các Subordinate CA (gọi tắt là sub. CA) và hoạt động bên dưới root CA. Ngoại trừ root CA thì các CA còn lại trong đều có duy nhất một CA khác là cấp trên của nó. Hệ thống tên miền DNS trên Internet cũng có cấu trúc tương tự mô hình này.

Tất cả các thực thể (như người dùng, máy tính) trong tổ chức đều phải tin cậy cùng một root CA. Sau đó các trust relationship được thiết lập giữa các sub. CA và cấp trên của chúng thông qua việc CA cấp trên sẽ cấp các chứng chỉ cho các sub. CA ngay bên dưới nó. Lưu ý, root CA không trực tiếp cấp chứng chỉ số cho các thực thể mà chúng sẽ được cấp bởi các sub. CA. Các CA mới có thể được thêm ngay dưới root CA hoặc các sub. CA cấp thấp hơn để phù hợp với sự thay đổi trong cấu trúc của tổ chức. Sẽ có các mức độ tổn thương khác nhau nếu một CA nào đó trong mô hình này bị xâm hại.

Trường hợp một sub. CA bị thỏa hiệp thì CA cấp trên của nó sẽ thu hồi chứng chỉ đã cấp cho nó và chỉ khi sub. CA đó được khôi phục thì nó mới có thể cấp lại các chứng chỉ mới cho người dùng của nó. Cuối cùng, CA cấp trên sẽ cấp lại cho nó một chứng chỉ mới.

Nếu root CA bị xâm hại thì đó là một vấn đề hoàn toàn khác, toàn bộ hệ thống PKI sẽ chịu ảnh hưởng. Khi đó tất cả các thực thể cần được thông báo về sự cố và cho đến khi root CA được phục hồi và các chứng chỉ mới được cấp lại thì không một phiên truyền thông nào là an toàn cả. Vì thế, cũng như single CA, root CA phải được bảo vệ an toàn ở mức cao nhất để đảm bảo điều đó không xảy ra và thậm chí root CA có thể ở trạng thái offline – bị tắt và không được kết nối vào mạng.

### *Mesh PKI*

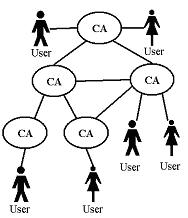


Figure 15. Mô hình Mesh.

Nổi lên như một sự thay thế chính cho mô hình Hierarchical PKI truyền thống, thiết kế của Mesh PKI giống với kiến trúc Web-of-Trust trong đó không có một CA nào làm root CA và các CA sẽ có vai trò ngang nhau trong việc cung cấp dịch vụ. Tất cả người dùng trong mạng lưới có thể tin cậy chỉ một CA bất kỳ, không nhất thiết hai hay nhiều người dùng phải cùng tin một CA nào đó và người dùng tin cậy CA nào thì sẽ nhận chứng chỉ do CA đó cấp.

Các CA trong mô hình này sau đó sẽ cấp các chứng chỉ cho nhau. Khi hai CA cấp chứng chỉ cho nhau thì một sự tin cậy hai chiều được thiết lập giữa hai CA đó. Các CA mới có thể được thêm vào bằng cách tạo các mối tin cậy hai chiều giữa chúng với các CA còn lại trong mạng lưới.

Vì không có một CA duy nhất làm cấp cao nhất nên sự tổn hại khi tấn công vào mô hình này có khác so với hai mô hình trước đó. Hệ thống PKI không thể bị đánh sập khi chỉ một CA bị thỏa hiệp. Các CA còn lại sẽ thu hồi chứng chỉ mà chúng đã cấp cho CA bị xâm hại và chỉ khi CA đó khôi phục hoạt động thì nó mới có khả năng cấp mới các chứng chỉ cho người dùng rồi thiết lập trust với các CA còn lại trong mạng lưới.

### *Single CA*

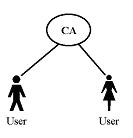


Figure 16. Mô hình Single CA.

Đây là mô hình PKI cơ bản nhất phù hợp với các tổ chức nhỏ trong đó chỉ có một CA cung cấp dịch vụ cho toàn hệ thống và tất cả người dùng đặt sự tin cậy vào CA này. Mọi thực thể muốn tham gia vào PKI và xin cấp chứng chỉ đều phải thông qua CA duy nhất này. Mô hình này dễ thiết kế và triển khai nhưng cũng có các hạn chế riêng. Thứ nhất là ở khả năng co giãn – khi quy mô tổ chức được mở rộng, chỉ một CA thì khó mà quản lý và đáp ứng tốt các dịch vụ. Hạn chế thứ hai là CA này sẽ là điểm chịu lỗi duy nhất, nếu nó ngưng hoạt động thì dịch vụ bị ngưng trệ. Cuối cùng, nếu nó bị xâm hại thì nguy hại tới độ tin cậy của toàn bộ hệ thống và tất cả các chứng chỉ số phải được cấp lại một khi CA này được phục hồi.

## **Chuẩn, đặc tả, luật pháp và chính sách**

Các tổ chức chuẩn hóa về PKI chính trên thế giới gồm:

* ITU-T recommendations.
* PKIX - Internet RFCs
* PKCS.

Các lĩnh vực chuẩn hóa về PKI gồm:

* Chuẩn về mật mã.
* Khuôn dạng chứng thư số và danh sách thu hồi chứng thư (CRL).
* Các giao thức hoạt động: LDAP, HTTP, FTP, X500.
* Các giao thức quản lý: CMP, CMS.
* Các chính sách: Chính sách chứng thư - Certificate Policy (CP) và quy chế chứng thực - Certification Practice Statement (CPS).
* Các dịch vụ cấp dấu thời gian và chứng thực dữ liệu.

# **CHƯƠNG 2: CẤU TRÚC DỮ LIỆU KÝ SỐ, MÃ MẬT CMS, XML JSON**

## **Cấu trúc dữ liệu ký số, mã mật CMS**

### *Giới thiệu chung*

#### Khái niệm

Cấu trúc CMS có tên đầy đủ tiếng Anh là Cryptographic Message Syntax, được dùng trong việc ký số, tóm tắt nội dung, xác thực hoặc mã hóa dữ liệu.

CMS cho phép đóng gói lồng nhau, tương tự như vậy, nó cho phép vừa có thể ký điện tử với một số dữ liệu được đóng gói trước đó. CMS cũng cho phép tùy chọn các thuộc tính, chẳng hạn như thời gian ký, …

#### Lịch sử phát triển

CMS có nguồn gốc từ PKCS#7 version 1.5, được ghi lại trong RFC 2315.

PKCS#7 version 1.5 được phát triển bên ngoài IETF; ban đầu nó được xuất bản dưới dạng RSA Laboratories Technical Note vào tháng 11 năm 1993. Kể từ thời điểm đó, IETF chịu trách nhiệm phát triển và duy trì CMS. Ngày nay, một số giao thức IETF Standards-Track quan trọng đã và đang sử dụng CMS. Phần này mô tả những thay đổi mà IETF đã thực hiện đối với CMS trong mỗi phiên bản đã xuất bản.

RFC-2630 là phiên bản đầu tiên của CMS trên IETF Standards-Track. Bất cứ khi nào PKCS#7 phiên bản 1.5 cũng tương thích được với các phiên bản sau. Tuy nhiên, các thay đổi đã được thực hiện đối với điều chỉnh chuyển chứng chỉ thuộc tính version 1 và hỗ trợ quản lý khóa độc lập theo thuật toán. PKCS#7 version 1.5 chỉ hỗ trợ cho việc trao đổi khóa. RFC-2630 bổ sung hỗ trợ cho thỏa thuận khóa và các kỹ thuật khóa mã hóa với mật mã khóa đối xứng được phân phối trước đó.

Mỗi cấu trúc dữ liệu chính bao gồm một số phiên bản như mục đầu tiên trong cấu trúc dữ liệu. Số phiên bản nhằm tránh lỗi giải mã ASN.1. Một số triển khai không kiểm tra số phiên bản trước khi thử giải mã và nếu xảy ra lỗi giải mã, thì số phiên bản được kiểm tra như một phần của quy trình xử lý lỗi. Đây là một cách tiếp cận hợp lý; nó đặt xử lý lỗi bên ngoài đường dẫn nhanh. Cách tiếp cận này cũng có thể tha thứ khi người gửi sử dụng số phiên bản không chính xác. Hầu hết các số phiên bản ban đầu đã được chỉ định trong PKCS # 7 phiên bản 1.5. Những người khác đã được chỉ định khi cấu trúc được tạo ban đầu. Bất cứ khi nào cấu trúc được cập nhật, số phiên bản cao hơn sẽ được chỉ định. Tuy nhiên, để đảm bảo khả năng tương tác tối đa, số phiên bản cao hơn chỉ được sử dụng khi tính năng cú pháp mới được sử dụng. Đó là, số phiên bản thấp nhất hỗ trợ cú pháp đã tạo được sử dụng.

### *Các chuẩn về CMS*

#### Tiêu chuẩn PKCS#7

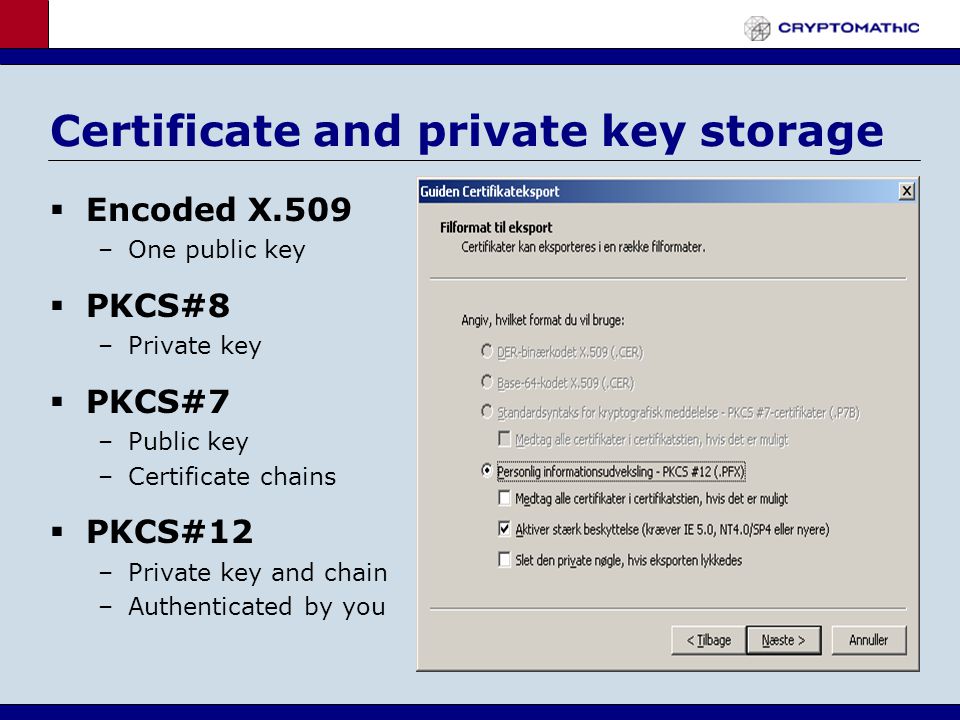


Figure 17. Tổng quan về các chuẩn chứng thư số.

PKCS#7 là một định dạng mã hóa cho việc lưu trữ một chứng chỉ số và chuỗi chứng nhận của nó dưới dạng các ký tự ASCII. Nó chủ yếu đươc sử dụng trên các nền tảng của Window và Java Tomcat. Hiện nay, CMS (Cryptographic Message Syntax) đang được sử dụng rất phổ biến, cũng giống như với SSL và TLS, PKCS#7 là tên thông thường mà tất cả chúng ta sử dụng và đang được thay thế. PKSC#7 có 2 định dạng mở rộng là: p7b và p7c. Không giống như PEM, PKCS#7 không thể lưu trữ khóa bảo mật (private key), nó chỉ có thể lưu trữ chứng chỉ gốc và chứng chỉ trung gian.

#### Tiêu chuẩn RFC-5652

##### *Giới thiệu*

Tài liệu này mô tả Cú pháp Thông điệp Mật mã (CMS). Cú pháp này được sử dụng để ký điện tử, thông báo, xác thực hoặc mã hóa nội dung tin nhắn tùy ý.

CMS mô tả cú pháp đóng gói để bảo vệ dữ liệu. Nó hỗ trợ chữ ký điện tử và mã hóa. Cú pháp cho phép nhiều gói; một phong bì đóng gói có thể được lồng vào bên trong một phong bì khác. Tương tự như vậy, một bên có thể ký điện tử một số dữ liệu đã được đóng gói trước đó. Nó cũng cho phép các thuộc tính tùy ý, chẳng hạn như thời gian ký, được ký cùng với nội dung thư và nó cung cấp cho các thuộc tính khác như ký tự đối lập được liên kết với một chữ ký.

CMS có thể hỗ trợ nhiều kiến ​​trúc khác nhau để quản lý khóa dựa trên chứng chỉ, chẳng hạn như kiến ​​trúc được xác định bởi nhóm làm việc PKIX (Cơ sở hạ tầng khóa công khai sử dụng X.509).

Các giá trị CMS được tạo bằng ASN.1, sử dụng mã hóa BER- (Quy tắc mã hóa cơ bản). Các giá trị thường được biểu diễn dưới dạng chuỗi octet. Trong khi nhiều hệ thống có khả năng truyền các chuỗi octet tùy ý một cách đáng tin cậy, thì ai cũng biết rằng nhiều hệ thống thư điện tử thì không. Tài liệu này không đề cập đến các cơ chế mã hóa chuỗi octet để truyền đáng tin cậy trong các môi trường như vậy.

##### *Phiên bản*

Mỗi cấu trúc dữ liệu chính bao gồm một số phiên bản là mục đầu tiên trong cấu trúc dữ liệu. Số phiên bản nhằm tránh lỗi giải mã ASN.1. Một số triển khai không kiểm tra số phiên bản trước khi thử giải mã và nếu xảy ra lỗi giải mã, thì số phiên bản được kiểm tra như một phần của quy trình xử lý lỗi. Đây là một cách tiếp cận hợp lý; nó đặt xử lý lỗi bên ngoài đường dẫn nhanh. Cách tiếp cận này cũng có thể tha thứ khi người gửi sử dụng số phiên bản không chính xác.

Hầu hết các số phiên bản ban đầu đã được chỉ định trong PKCS#7 phiên bản 1.5. Những người khác đã được chỉ định khi cấu trúc được tạo ban đầu. Bất cứ khi nào cấu trúc được cập nhật, số phiên bản cao hơn sẽ được chỉ định.

Tuy nhiên, để đảm bảo khả năng tương tác tối đa, số phiên bản cao hơn chỉ được sử dụng khi tính năng cú pháp mới được sử dụng. Đó là, số phiên bản thấp nhất hỗ trợ cú pháp đã tạo được sử dụng.

##### *Data Content Type*

Kiểu nội dung dữ liệu nhằm tham chiếu đến các chuỗi octet tùy ý, chẳng hạn như tệp văn bản ASCII; việc giải thích được để lại cho ứng dụng. Các chuỗi như vậy không cần có bất kỳ cấu trúc bên trong nào (mặc dù chúng có thể có định nghĩa ASN.1 riêng hoặc cấu trúc khác).

S/MIME sử dụng dữ liệu id để xác định nội dung được mã hóa MIME. Việc sử dụng mã nhận dạng nội dung này được chỉ định trong RFC 2311 cho S/MIME v2, RFC 2633 cho S/MIME v3 và RFC 3851 cho S/MIME v3.1.

##### Signed-data Content Type

Kiểu nội dung dữ liệu đã ký bao gồm nội dung thuộc bất kỳ kiểu nào và không hoặc nhiều giá trị chữ ký. Bất kỳ số người ký song song có thể ký bất kỳ loại nội dung nào.

Ứng dụng điển hình của kiểu nội dung dữ liệu đã ký thể hiện chữ ký số của một người ký trên nội dung của kiểu nội dung dữ liệu. Một ứng dụng điển hình khác phổ biến chứng chỉ và danh sách thu hồi chứng chỉ (CRL).

###### *SignedData Type*

Những tham số được đề cập trong loại cấu trúc này là:

* DigestAlgo là một tập hợp các mã định danh thuật toán thông báo thông báo. CÓ THỂ có bất kỳ số phần tử nào trong tập hợp, kể cả số không. Mỗi phần tử xác định thuật toán thông báo thông báo, cùng với bất kỳ tham số liên quan nào, được sử dụng bởi một hoặc nhiều người ký. Bộ sưu tập nhằm mục đích liệt kê các thuật toán thông báo thông báo được sử dụng bởi tất cả những người ký, theo bất kỳ thứ tự nào, để tạo điều kiện xác minh chữ ký một lần. Việc triển khai CÓ THỂ không xác thực được chữ ký sử dụng thuật toán thông báo không có trong tập hợp này.
* EncapContentInfo là nội dung đã ký, bao gồm một mã định danh loại nội dung và chính nội dung đó.
* SignerInfos là một tập hợp thông tin về mỗi người ký. CÓ THỂ có bất kỳ số phần tử nào trong tập hợp, kể cả số không. Khi bộ sưu tập đại diện cho nhiều hơn một chữ ký, việc xác thực thành công một chữ ký từ một người ký nhất định phải được coi là một chữ ký thành công của người ký đó. Tuy nhiên, có một số môi trường ứng dụng cần các quy tắc khác. Chi tiết về kiểu SignerInfo được thảo luận trong Phần 5.3. Vì mỗi người ký có thể sử dụng một kỹ thuật chữ ký số khác nhau và các thông số kỹ thuật trong tương lai có thể cập nhật cú pháp, nên tất cả việc triển khai PHẢI xử lý một cách duyên dáng các phiên bản chưa hoàn thiện của SignerInfo. Hơn nữa, vì tất cả các triển khai sẽ không hỗ trợ hết các thuật toán chữ ký có thể có, nên tất cả các triển khai PHẢI xử lý các thuật toán chữ ký một cách linh hoạt khi chúng gặp phải.

###### *EncapsulatedContentInfo Type*

Những tham số được đề cập trong loại cấu trúc này là:

* eContentType là một định danh đối tượng. Định danh đối tượng chỉ định duy nhất kiểu nội dung.
* eContent là chính nội dung, được mang dưới dạng một chuỗi octet. EContent không cần được mã hóa DER.

###### *SignerInfo Type*

Những tham số được đề cập trong loại cấu trúc này là:

* Version là số phiên bản cú pháp. Nếu SignerIdentifier là nhà phát hành CHOICEAndSerialNumber, thì phiên bản PHẢI là 1. Nếu SignerIdentifier là subjectKeyIdentifier, thì phiên bản phải là 3.
* SID chỉ định chứng chỉ của người ký (và do đó là khóa công khai của người ký). Người nhận cần khóa công khai của người ký để xác minh chữ ký.
* SignAttrs là một tập hợp các thuộc tính được ký.
* SignatureAlgorithm xác định thuật toán chữ ký và bất kỳ tham số liên quan nào, được người ký sử dụng để tạo chữ ký điện tử.
* Signature là kết quả của việc tạo chữ ký điện tử, sử dụng bản tóm tắt thông báo và khóa riêng của người ký.

###### *Quá trình tính toán bảng tóm tắt (Message digest)*

Quá trình tính toán thông báo thông báo tính toán thông báo tóm tắt về nội dung được ký hoặc nội dung cùng với các thuộc tính đã ký. Trong cả hai trường hợp, đầu vào ban đầu cho quá trình tính toán thông báo thông báo là "giá trị" của nội dung được đóng gói đang được ký. Cụ thể, đầu vào ban đầu là encapContentInfo eContent OCTET STRING mà quá trình ký được áp dụng. Chỉ các octet bao gồm giá trị của eContent OCTET STRING mới được đầu vào cho thuật toán thông báo thông báo, không phải thẻ hoặc các octet độ dài.

Kết quả của quá trình tính toán thông báo thông báo phụ thuộc vào việc trường signAttrs có xuất hiện hay không. Khi trường không có, kết quả chỉ là thông báo tóm tắt nội dung như được mô tả ở trên. Tuy nhiên, khi trường hiện diện, kết quả là bản tóm tắt thông báo về mã hóa DER hoàn chỉnh của giá trị SignedAttrs có trong trường SignAttrs. Vì giá trị SignedAttrs, khi xuất hiện, phải chứa loại nội dung và thuộc tính thông báo-thông báo, những giá trị đó được đưa vào kết quả một cách gián tiếp. Thuộc tính loại nội dung KHÔNG ĐƯỢC đưa vào thuộc tính không dấu của bộ đếm như được định nghĩa trong Phần 11.4. Một mã hóa riêng của trường signAttrs được thực hiện để tính toán thông báo thông báo. Thẻ IMPLICIT [0] trong signAttrs không được sử dụng cho mã hóa DER, thay vào đó thẻ EXPLICIT SET OF được sử dụng. Nghĩa là, mã hóa DER của thẻ EXPLICIT SET OF, chứ không phải của thẻ IMPLICIT, PHẢI được đưa vào phép tính thông báo thông báo cùng với các octet độ dài và nội dung của giá trị SignedAttributes. Khi không có trường signAttrs, chỉ các octet bao gồm giá trị của SignedData eContent OCTET STRING (ví dụ: nội dung của tệp) được nhập vào phép tính thông báo thông báo. Điều này có ưu điểm là độ dài của nội dung được ký không cần phải biết trước trong quá trình tạo chữ ký.

Mặc dù thẻ encapContentInfo eContent OCTET STRING và các octet độ dài không được bao gồm trong tính toán thông báo thông báo, chúng được bảo vệ bằng các phương tiện khác. Các octet độ dài được bảo vệ bởi bản chất của thuật toán tổng hợp thông báo vì nó không khả thi về mặt tính toán để tìm bất kỳ hai nội dung thông báo riêng biệt nào có độ dài bất kỳ có cùng một thông báo tóm tắt.

###### *Quá trình tạo chữ ký số*

Đầu vào cho quá trình tạo chữ ký bao gồm kết quả của quá trình tính toán bản tóm tắt thông báo và khóa riêng của người ký. Các chi tiết của việc tạo chữ ký phụ thuộc vào thuật toán chữ ký được sử dụng. Định danh đối tượng, cùng với bất kỳ tham số nào, chỉ định thuật toán chữ ký mà người ký sử dụng được mang trong trường signatureAlgorithm. Giá trị chữ ký được tạo bởi người ký PHẢI được mã hóa dưới dạng OCTET STRING và được mang trong trường chữ ký.

###### *Quá trình xác thực chữ ký số*

Đầu vào cho quá trình xác minh chữ ký bao gồm kết quả của quá trình tính toán bản tóm tắt thông điệp và khóa công khai của người ký. Người nhận CÓ THỂ lấy được khóa công khai chính xác cho người ký bằng bất kỳ phương tiện nào, nhưng phương pháp ưu tiên là từ chứng chỉ lấy được từ trường chứng chỉ SignedData. Việc lựa chọn và xác nhận khóa công khai của người ký CÓ THỂ dựa trên xác nhận đường dẫn chứng nhận cũng như bối cảnh bên ngoài khác, nhưng nằm ngoài phạm vi của tài liệu này. Các chi tiết của xác minh chữ ký phụ thuộc vào thuật toán chữ ký được sử dụng.

Người nhận KHÔNG PHẢI dựa vào bất kỳ giá trị thông báo thư nào của người khởi tạo. Nếu người ký SignedDataInfo bao gồm các Thuộc tính đã ký, thì thông báo nội dung PHẢI được. Để chữ ký hợp lệ, giá trị thông báo tin nhắn do người nhận tính phải PHẢI giống với giá trị của thuộc tính messageDigest có trong các Thuộc tính đã ký của Người ký dữ liệu đã ký SignedDataInfo.

Nếu SignedData signerInfo bao gồm các thuộc tính signAttributes, thì giá trị thuộc tính kiểu nội dung PHẢI khớp với giá trị SignedData encapContentInfo eContentType.

##### *Enveloped-Data Content Type*

###### *EnvelopedData Type*

Cấu trúc này được cấu tạo như sau:

* Khóa mã hóa nội dung cho một thuật toán mã hóa nội dung cụ thể được tạo ngẫu nhiên.
* Khóa mã hóa nội dung được mã hóa cho từng người nhận. Các chi tiết của mã hóa này phụ thuộc vào thuật toán quản lý khóa được sử dụng, nhưng bốn kỹ thuật chung được hỗ trợ:
  + Key transport: khóa mã hóa nội dung được mã hóa trong khóa công khai của người nhận.
  + Key agreement: khóa công khai của người nhận và khóa riêng của người gửi được sử dụng để tạo khóa đối xứng theo cặp, sau đó khóa mã hóa nội dung được mã hóa trong khóa đối xứng theo cặp.
  + Symmetric key-encryption keys: khóa mã hóa nội dung được mã hóa trong khóa mã hóa khóa đối xứng được phân phối trước đó.
  + Password: khóa mã hóa nội dung được mã hóa bằng khóa mã hóa khóa có nguồn gốc từ mật khẩu hoặc giá trị bí mật được chia sẻ khác.
* Đối với mỗi người nhận, khóa mã hóa nội dung được mã hóa và thông tin dành riêng cho người nhận khác được thu thập vào một giá trị RecipientInfo.
* Nội dung được mã hóa bằng khóa mã hóa nội dung. Mã hóa nội dung có thể yêu cầu nội dung được đệm vào nhiều kích thước khối nào đó.
* Các giá trị RecipientInfo cho tất cả người nhận được thu thập cùng với nội dung được mã hóa để tạo thành giá trị EnvelopedData như đã xác định.

###### *RecipientInfo Type*

Thông tin về mỗi người nhận được thể hiện trong kiểu RecipientInfo. RecipientInfo có một định dạng khác nhau cho từng kỹ thuật quản lý khóa được hỗ trợ. Bất kỳ kỹ thuật quản lý khóa nào cũng có thể được sử dụng cho mỗi người nhận cùng một nội dung được mã hóa. Trong mọi trường hợp, khóa mã hóa nội dung được mã hóa được chuyển đến một hoặc nhiều người nhận.

Vì tất cả các triển khai sẽ không hỗ trợ mọi thuật toán quản lý khóa có thể có, nên tất cả các triển khai PHẢI xử lý các thuật toán chưa hoàn thiện một cách uyển chuyển khi chúng gặp phải. Ví dụ: nếu người nhận nhận được khóa mã hóa nội dung được mã hóa trong khóa công khai RSA của họ bằng cách sử dụng RSA-OAEP (Lớp đệm mã hóa bất đối xứng tối ưu) và việc triển khai chỉ hỗ trợ RSA PKCS # 1 v1.5, thì một lỗi duyên dáng phải được thực hiện.

Việc triển khai PHẢI hỗ trợ truyền khóa, thỏa thuận khóa và các khóa mã hóa khóa đối xứng được phân phối trước đó, như được đại diện bởi ktri, kari và kekri, tương ứng. Việc triển khai CÓ THỂ hỗ trợ quản lý khóa dựa trên mật khẩu như được đại diện bởi pwri. Việc triển khai CÓ THỂ hỗ trợ bất kỳ kỹ thuật quản lý khóa nào khác được đại diện bởi ori. Vì mỗi người nhận có thể sử dụng một kỹ thuật quản lý khóa khác nhau và các thông số kỹ thuật trong tương lai có thể xác định các kỹ thuật quản lý khóa bổ sung, tất cả các triển khai PHẢI xử lý một cách duyên dáng các lựa chọn thay thế chưa hoàn thiện trong RecipientInfo CHOICE, tất cả các triển khai PHẢI xử lý một cách duyên dáng các phiên bản chưa hoàn thiện của các lựa chọn thay thế được hỗ trợ khác trong RecipientInfo CHOICE, và tất cả triển khai PHẢI xử lý một cách duyên dáng các giải pháp thay thế ori chưa hoàn thành hoặc chưa biết.

###### *KEKRecipientInfo Type*

Những tham số được đề cập trong loại cấu trúc này là:

* Versopn là số phiên bản cú pháp. Nó PHẢI luôn luôn là 4.
* Kekid chỉ định một khóa mã hóa khóa đối xứng đã được phân phối trước đó cho người gửi và một hoặc nhiều người nhận.
* KeyEncryptionAlgorithm xác định thuật toán mã hóa khóa và bất kỳ tham số liên quan nào, được sử dụng để mã hóa khóa mã hóa nội dung bằng khóa mã hóa khóa. Quá trình mã hóa khóa được mô tả.
* EncryptedKey là kết quả của việc mã hóa khóa mã hóa nội dung trong khóa mã hóa khóa.

###### *Quá trình mã hóa nội dung*

Khóa mã hóa nội dung cho thuật toán mã hóa nội dung mong muốn được tạo ngẫu nhiên. Dữ liệu cần được bảo vệ được đệm như mô tả bên dưới, sau đó dữ liệu được đệm được mã hóa bằng khóa mã hóa nội dung. Thao tác mã hóa ánh xạ một chuỗi octet tùy ý (dữ liệu) sang một chuỗi octet khác (theciphertext) dưới sự kiểm soát của khóa mã hóa nội dung. Dữ liệu được mã hóa được bao gồm trong EnvelopedData đã mã hóaContentInfo được mã hóaContent OCTET STRING.

###### *Quá trình mã hóa khóa đối xứng*

Đầu vào cho quá trình mã hóa khóa - giá trị được cung cấp cho thuật toán mã hóa khóa của người nhận - chỉ là "giá trị" của khóa mã hóa nội dung.

Bất kỳ kỹ thuật quản lý khóa nào đã nói ở trên đều có thể được sử dụng cho mỗi người nhận cùng một nội dung được mã hóa.

#### Tiêu chuẩn ETSI TS 101 733

##### *Khái niệm*

Có hai tiêu chuẩn phổ biến được đề cập ở ETSI TS 101 733 là phiên bản v1.8.1 (v010801p) và v2.2.1 (020201p).

Thế giới hiện nay thương mại điện tử đang nổi lên như một phương thức kinh doanh tương lai giữa các công ty ở địa phương, quốc gia và toàn cầu. Sự tin tưởng vào cách kinh doanh này là điều cần thiết cho sự thành công và tiếp tục phát triển của thương mại điện tử. Do đó, điều quan trọng là các công ty sử dụng phương tiện kinh doanh điện tử này phải có các cơ chế và kiểm soát an ninh phù hợp để bảo vệ các giao dịch của họ và đảm bảo sự tin cậy và niềm tin với các đối tác kinh doanh của họ.

Về mặt này, chữ ký điện tử là một thành phần bảo mật quan trọng có thể được sử dụng để bảo vệ thông tin và cung cấp sự tin cậy trong kinh doanh điện tử. Điều này bao gồm bằng chứng về tính hợp lệ của nó ngay cả khi người ký hoặc bên xác minh cố gắng từ chối tính hợp lệ của chữ ký. Tiêu chuẩn này hiện tại được áp đụng với rất nhiều môi trường; nó có thể được áp dụng cho bất kỳ môi trường nào, ví dụ: thẻ thông minh, thẻ SIM GSM, các chương trình đặc biệt cho chữ ký điện tử, v.v.

Châu Âu định nghĩa chữ ký điện tử là dữ liệu ở dạng điện tử được đính kèm hoặc liên kết hợp lý với dữ liệu điện tử khác và được dùng như một phương pháp xác thực.

##### *Các định dạng chữ ký điện tử*

###### *Chữ ký điện tử cơ bản CAdES (CAdES-BES)*

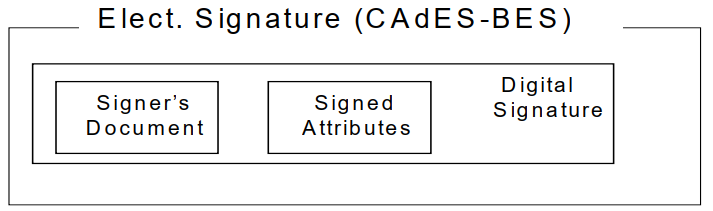


Figure 18. Cấu trúc CAdES-BES.

Chữ ký điện tử cơ bản bao gồm:

* Thông tin người ký.
* Một tập hợp các thuộc tính bắt buộc trong ký số.
* Các thuộc tính bổ sung.
* Giá trị chữ ký điện tử được tính toán trên dữ liệu người dùng.

Các thuộc tính bắt buộc trong ký số là:

* Content type: loại nội dung, được chứa trong cấu trúc thông tin đóng gói (encapsulated).
* Message digest: bảng tóm tắt dữ liệu.
* Chứng chỉ để ký ESS signing certificate chỉ cho phép sử dụng thuật toán SHA1.

Các thuộc tính không bắt buộc được thêm vào CadES-BES là:

* Signing time: thời gian ký.
* Content hints: gợi ý của nội dung, cơ bản cho biết nội dung có được gói trong một lớp nội dung khác.
* Signer location: mô tả vị trí địa lý người ký.
* Content time stamp: cho biết khoảng thời gian đến thời điểm hiện tại kể từ lúc ký.

###### *Chữ ký điện tử CadES dựa trên chính sách minh bạch (CadES-EPES)*

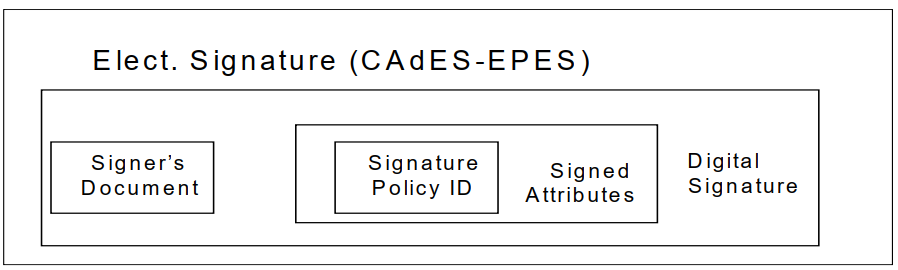


Figure 19. Kiến trúc CAdES-EPES.

Chữ ký điện tử CadES-EPES mở rộng định nghĩa về chữ ký điện tử của chữ ký điện tử cơ bnar CadES-BES. CadES-EPES kết hợp với thuộc tính sigPoliciID để xác định về chính sách chữ ký số.

##### *Các định dạng chữ ký điện tử với dữ liệu xác thực*

###### *CAdES-T*

Chữ ký điện tử với thời gian (Electronic Signature with Time), loại chữ ký điện tử này chủ yếu dựa vào sự tin cậy thời gian.

Thời gian đáng tin cậy có thể được cung cấp bởi:

* Một thuộc tính tem thời gian là một thuộc tính được thêm vào ES.
* Dấu thời gian của ES do Trusted Service Provider.

Time-stamp do Trusted Service Provider sẽ có tác dụng tương tự như thuộc tính chữ ký time-stamp, nhưng trong trường hợp này, không có thuộc tính nào được thêm vào ES, vì TSP có trách nhiệm cung cấp bằng chứng về thời gian - đánh dấu khi được yêu cầu. Việc quản lý các mốc thời gian thật sự phức tạp và được đảm nhiệm bởi các nhóm chuyên gia riêng. Thời gian đáng tin cậy cung cấp các bước ban đầu hướng tới việc cung cấp giá trị lâu dài. Chữ ký điện tử có thuộc tính dấu thời gian hoặc BES / EPES được đánh dấu thời gian tạo thành CAdES-T.

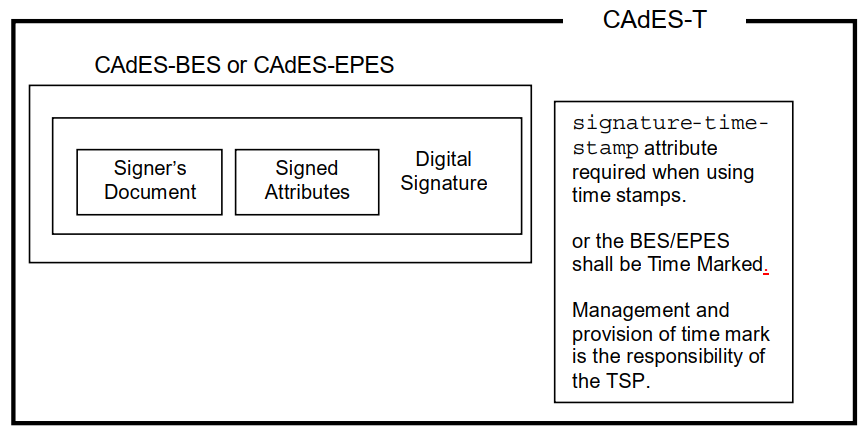


Figure 20. Cấu trúc CAdES-T.

###### *CAdES-C*

Chữ ký điện tử với các tham chiếu dữ liệu xác thực hoàn chỉnh (CAdES-C), phù hợp với tài liệu hiện tại, bổ sung vào CAdES-T các tham chiếu đầy đủ-chứng chỉ và các thuộc tính hoàn chỉnh, như được định nghĩa trong tài liệu hiện tại. Chứng chỉ hoàn chỉnh-thuộc tính chứa các tham chiếu đến tất cả các chứng chỉ có trong đường dẫn được sử dụng để xác minh chữ ký. Thuộc tính hoàn chỉnh-thu hồi-tham chiếu chứa các tham chiếu đến các phản hồi CRL hoặc OCSP được sử dụng để xác minh chữ ký. Việc lưu trữ các tham chiếu cho phép các giá trị của đường dẫn chứng nhận và các phản hồi CRL hoặc OCSP được lưu trữ ở nơi khác, với kích thước của định dạng chữ ký điện tử được lưu trữ.

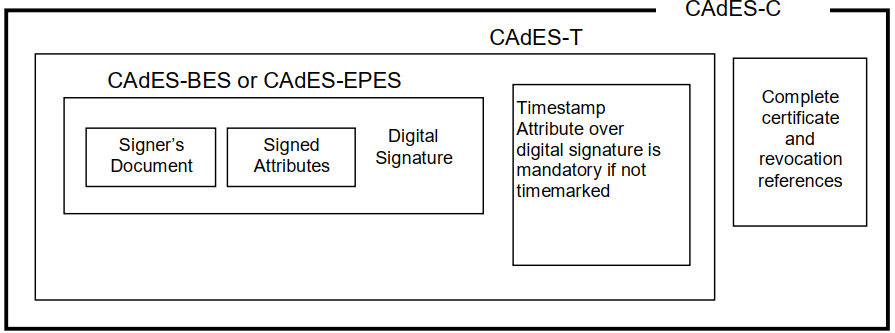


Figure 21. Cấu trúc CAdES-C

###### *CAdES-A*

Tài liệu hiện tại xác định hai biểu mẫu lưu trữ (CAdES-A) cho chữ ký CAdES, đó là: biểu mẫu CAdES-A có thuộc tính lưu trữ-thời gian-đóng dấu (ATSv2) và biểu mẫu CAdES-A có lưu trữ-thời gian-tem-v3 ( Thuộc tính ATSv3). Các hệ thống tuyên bố tuân thủ CAdES-A, như được định nghĩa trong tài liệu này, sẽ tạo ra các dấu thời gian lưu trữ sử dụng ATSv3. ATSv2 được bao gồm để tương thích ngược. Hệ thống có thể xác minh các chữ ký được tạo bằng ATSv2 và kéo dài thời gian tồn tại của chúng bằng ATSv3. ATSv3 được xây dựng dựa trên các khái niệm được định nghĩa trong ATSv2 được định nghĩa trong 6.4.1 và thuộc tính xác thực dài hạn được xác định trong điều khoản 6.5, đảm bảo xử lý rõ ràng các cấu trúc bên trong chữ ký và mang lại sự linh hoạt hơn cho việc kết hợp các thuộc tính không dấu (ví dụ: chữ ký ) hoặc chữ ký song song, sau khi chữ ký đã được đóng dấu thời gian lưu trữ.

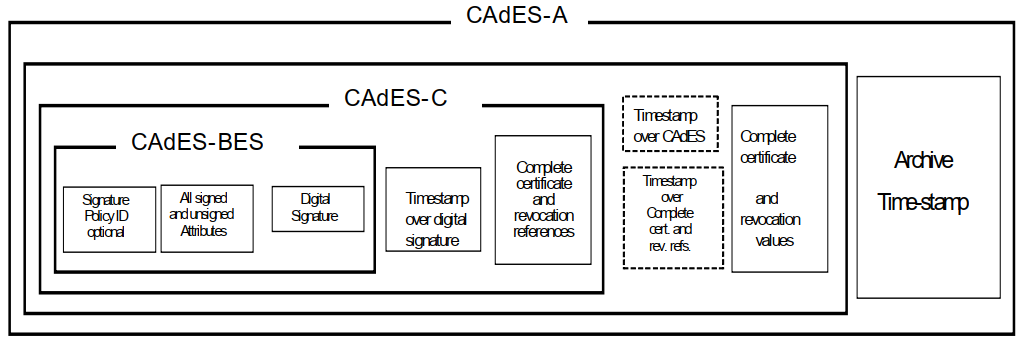


Figure 22. Cấu trúc CAdES-A

###### *CAdES-LT*

Biểu mẫu dài hạn (CAdES-LT), phù hợp với tài liệu hiện tại được xây dựng trên bất kỳ định dạng nào trong số CAdES-T, CAdES-C, CAdES-X Dài, CAdES-X Dài Loại 1 hoặc 2 hoặc CAdES-A bằng cách thêm một hoặc hơn thuộc tính xác nhận dài hạn. Biểu mẫu này được sử dụng để lưu trữ các chữ ký dài hạn. Một thuộc tính như vậy có thể được sử dụng trước khi gửi chữ ký đến dịch vụ bảo quản hoặc các dấu thời gian liên tiếp có thể được sử dụng để bảo vệ toàn bộ tài liệu khỏi các thuật toán băm dễ bị tấn công hoặc việc phá vỡ tài liệu hoặc thuật toán mật mã. Điều khoản 6.5 bao gồm các chi tiết kỹ thuật. Điều C.4.5 và C.4.8 cung cấp cơ sở lý luận. Thuộc tính này giống với các dạng CAdES-X Long và CAdES-A. Tuy nhiên, nó đơn giản hơn để thực hiện và linh hoạt hơn những thứ này. Việc sử dụng biểu mẫu CAdES-LT không được chấp nhận cho bất kỳ chữ ký nào chưa áp dụng thuộc tính xác thực dài hạn.

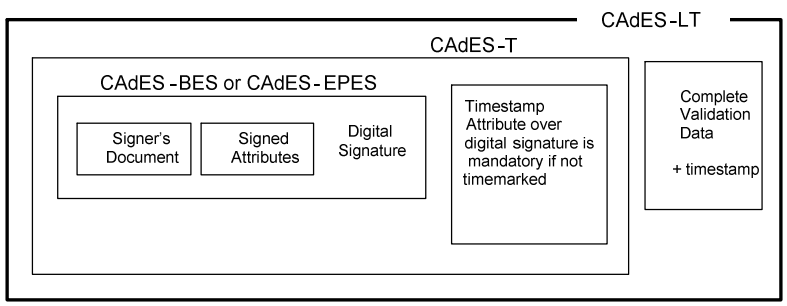


Figure 23. Cấu trúc CAdES-LT

### *Cấu trúc dữ liệu CMS*

#### Cấu trúc ký số

##### Cấu trúc nội dung Signed Data

Kiểu nội dung dữ liệu đã ký bao gồm nội dung thuộc bất kỳ kiểu nào và không hoặc nhiều giá trị chữ ký. Bất kỳ số người ký song song có thể ký bất kỳ loại nội dung nào. Ứng dụng điển hình của kiểu nội dung dữ liệu đã ký thể hiện chữ ký số của một người ký trên nội dung của kiểu nội dung dữ liệu. Một ứng dụng điển hình khác phổ biến chứng chỉ và danh sách thu hồi chứng chỉ (CRL).

##### Quá trình hình thành Signed Data

Quá trình Signed-data Content được xây dựng liên quan đến các bước sau

* Đối với mỗi người ký, bản tóm tắt dữ liệu, bang hoặc bản băm, được tính toán trên nội dung bằng thuật toán tóm tắt dữ liệu được chỉ định bởi người ký. Nếu người ký đang ký bất kỳ thông tin nào ngoài nội dung, thì bản tóm tắt nội dung và thông tin khác sẽ được tóm tắt lại bằng thuật toán thông báo thông báo của người.
* Đối với mỗi người ký, bản tóm tắt nội dung được ký bằng khóa riêng của người ký.
* Đối với mỗi người ký, giá trị chữ ký và thông tin khác của người ký thu thập thành giá trị gọi là SignerInfo.
* Các thuật toán tóm tắt nội dung và giá trị SignerInfo được thu thập cùng với nội dung trở thành một giá trị SignedData.

##### Cấu trúc Signed Data

Loại thông tin này gồm có các trường sau

|  |
| --- |
| // mã nguồn BouncyCastle v1.9.0  public class SignedData : Asn1Encodable  {  private static readonly DerInteger Version1 = new DerInteger(1);  private static readonly DerInteger Version3 = new DerInteger(3);  private static readonly DerInteger Version4 = new DerInteger(4);  private static readonly DerInteger Version5 = new DerInteger(5);  private readonly DerInteger version;  private readonly Asn1Set digestAlgorithms;  private readonly ContentInfo contentInfo;  private readonly Asn1Set certificates;  private readonly Asn1Set crls;  private readonly Asn1Set signerInfos;  private readonly bool certsBer;  private readonly bool crlsBer;  // …  } |

* Version: tương ứng với CMSVersion trong tiêu chuẩn RFC-5652, tức phiên bản CMS.
* DigestAlgo: Thuật toán tóm tắt.
* ContentInfo: tương ứng với EncapContentInfo trong tiêu chuẩn RFC-5652, tức nội dung được ký.
* Certificates: Tập các chứng chỉ.
* Clrs: Các danh sách thu hồi.
* SignerInfo: Cacs thông tin người ký.

##### Cấu trúc Encapsulated Content Info

Loại thông tin này gồm có các trường sau:

|  |
| --- |
| // mã nguồn BouncyCastle v1.9.0  public class ContentInfo : Asn1Encodable  {  private readonly DerObjectIdentifier contentType;  private readonly Asn1Encodable content;  // … |

* ContentType: loại nội dung
* Content: đối tượng nội dung chính, dữ liệu này không cần mã hóa DER

##### Cấu trúc Signer Info

Mỗi thông tin người ký sẽ được đại diện thông qua SignerInfo, cấu trúc SignerInfo bao gồm các trường sau:

|  |
| --- |
| // mã nguồn BouncyCastle v1.9.0  public class SignerInfo : Asn1Encodable  {  private DerInteger version;  private SignerIdentifier sid;  private AlgorithmIdentifier digAlgorithm;  private Asn1Set authenticatedAttributes;  private AlgorithmIdentifier digEncryptionAlgorithm;  private Asn1OctetString encryptedDigest;  private Asn1Set unauthenticatedAttributes;  // …  } |

* Version: tương ứng với CMSVersion trong tiêu chuẩn RFC-5652, tức phiên bản CMS.
* Sid: ID của người ký, khi người nhận bóc tách dữ liệu ở trường này sẽ nhận được 2 thông tin dùng để xác minh chữ ký:
* IssuerAndSerialNumber: định danh chứng chỉ thông qua tên người phát hành và số serial
* SubjectKeyId: định danh chứng cỉ thông qua subject key
* DigAlgorithm: thuật toán tóm tắt
* AuthenticatedAttributes: tương ứng với SignedAttr trong tiêu chuẩn RFC-5652, tức các thuộc tính được ký.
* DigEncryptionAlgorithm: tương ứng với SignatureAlgo trong tiêu chuẩn RFC-5652, tức thuật toán ký.
* EncryptedDigest: tương ứng với Signature trong tiêu chuẩn RFC-5652, tức chữ ký số.
* UnauthenticatedAttributes: tương ứng với UnsignedAttr trong tiêu chuẩn RFC-5652, tức các thuộc tính không ký.

#### Cấu trúc mã mật

##### Cấu trúc Enveloped Data

Cấu trúc thông tin này được dùng cho việc mã hóa, các trường thông tin quan trọng trong cấu trúc gồm có:

|  |
| --- |
| // mã nguồn BouncyCastle v1.9.0  public class EnvelopedData : Asn1Encodable  {  private DerInteger version;  private OriginatorInfo originatorInfo;  private Asn1Set recipientInfos;  private EncryptedContentInfo encryptedContentInfo;  private Asn1Set unprotectedAttrs;  // …  } |

* Version: tương ứng với CMSVersion trong tiêu chuẩn RFC-5652, tức phiên bản CMS;
* OriginatorInfo: thông tin người khởi tạo, bao gồm tập các chứng chỉ và các CRL
* EncryptedContentInfo: thông tin nội dung được mã hóa, bao gồm 3 trường nhỏ:
  + ContentType: loại nội dung.
  + ContentEncryptionAlgorithm: thuật toán mã hóa nội dung.
  + EncryptedContent: bản mã.

##### Cấu trúc KekRecipient Info

Loại cấu trúc được dùng trong việc hỗ trợ mã hóa, gồm có các trường sau:

|  |
| --- |
| // mã nguồn BouncyCastle v1.9.0  public class KekRecipientInfo : Asn1Encodable  {  private DerInteger version;  private KekIdentifier kekID;  private AlgorithmIdentifier keyEncryptionAlgorithm;  private Asn1OctetString encryptedKey;  // …  } |

* Version: tương ứng với CMSVersion tiêu chuẩn RFC-5652, tức phiên bản CMS.
* KekID: định danh của Key Encryption Key (khóa dùng cho việc bảo vệ khóa mã hóa).
* KeyEncryptionAlgorithm: thuật toán mã hóa của khóa.
* EncryptedKey: khóa mã hóa nội dung hoặc khóa được bảo vệ.

## **Cấu trúc dữ liệu ký số, mã mật XML**

### *Giới thiệu chung*

#### Khái niệm

XML được viết tắt từ tên eXtensible Markup Language, có nghĩa là ngôn ngữ đánh dấu mở rộng. Ngôn ngữ này được World Wide Web Consortium (W3C) kiến nghị tạo nên để góp phần xây dựng các API Service. XML có khả năng truyền dữ liệu và đọc dịch nhiều loại dữ liệu khác nhau. API sẽ trả kết quả về dạng XML để các hệ thống khác nhau có thể giao tiếp với nhau.

Nếu mô tả một cách đơn giản hơn thì XML giống như một máy phiên dịch ngôn ngữ giữa các hệ thống với nhau vậy. Ngoài chức năng thông dịch, nó còn giúp đơn giản hóa dữ liệu giữa các platform và hệ thống khác nhau. XML cũng có thể được dùng như một kho lưu trữ các dữ liệu đã trao đổi.

#### Đặc điểm

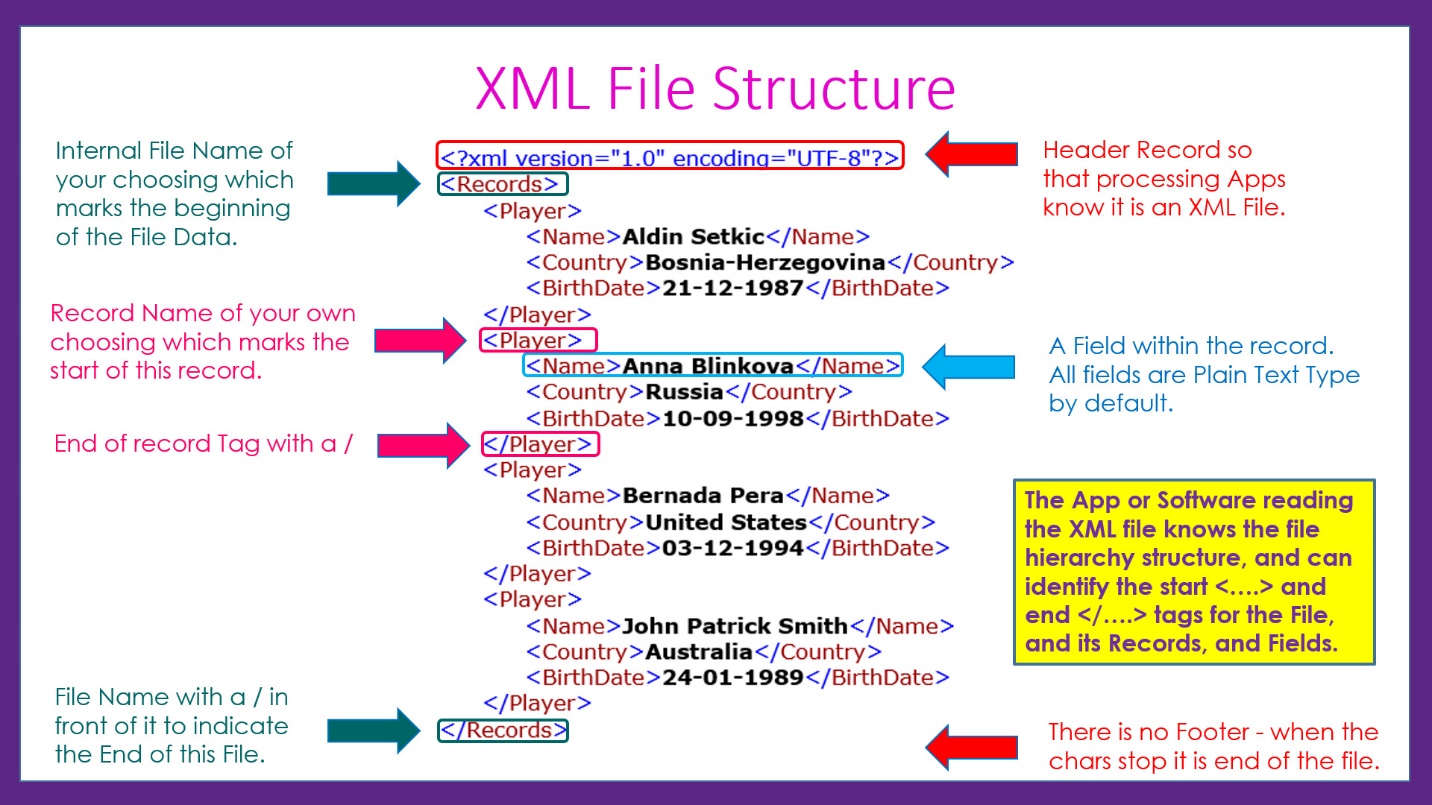


Figure 24. Cấu trúc tệp tin XML.

XML được dùng cho những loại dữ liệu có cấu trúc. XML có cấu tạo trực quan khá giống với HTML, nhưng vẫn khác nhau ở nhiều điểm. Có thể nói, XML là cầu nối để đưa HTML đến với XHTML.

Tuy XML tồn tại như một dạng văn bản nhưng nó không dùng để đọc, giúp các hệ thống khác nhau có thể thấu hiểu lẫn nhau. Khi ta nhắc đến định nghĩa đọc XML trong Java thì có nghĩa là dùng XML để mã hóa Java theo ngôn ngữ đánh dấu.

Một số định dạng phổ biến:

* RSS và ATOM: Chúng đều dùng để mô tả cách làm thế nào mà ứng dụng đọc xử lý được nguồn cấp web.
* Microsoft .NET: Hệ thống này sẽ sử dụng XML cho các tệp tin cấu hình của nó.
* Các phiên bản kể từ Microsoft Office 2007: Chúng sử dụng XML làm cơ sở cho cấu trúc tài liệu. Ký hiệu “X” trong định dạng tài liệu Word .DOCX cũng xuất phát từ đây.
* Ngoài ra, XML cũng được ứng dụng trong Excel (tệp tin XLSX) và PowerPoint (tệp tin PPTX).

#### Sự khác nhau giữa XML và HTML

|  |  |
| --- | --- |
| **XML** | **HTML** |
| Do người dùng định nghĩa | Được định nghĩa trước và người dùng phải tuân thủ |
| Được thiết kế để chuyển tải và lưu trữ dữ liệu, tập trung vào "what data are" | Được thiết kế để hiển thị dữ liệu, tập trung vào "how data look" |

#### Cấu trúc và thành phần của tệp tin XML

Cấu trúc XML gồm có 5 thành phần chính:

* Header là thẻ đầu tiên trong tệp tin XML, được dùng để thông báo version (thường là 1.0) và còn có tác dụng lưu trữ thông tin về việc mã hóa ký tự. Giá trị encoding của Header tệp tin sẽ thuộc một trong những dạng sau: UTF-8, UTF-16, ISO-10646-UCS-2, ISO-10646-UCS-4, ISO-2022-JP. Còn về encoding là gì thì nó là một dạng mã hóa ký tự.
* Ví dụ về thẻ Header tệp tin: **<?xml version=”1.0″ encoding=”utf-8″ ?>**. UTF-8 là viết tắt của 8-bit Unicode Transformation Format có nghĩa là định dạng chuyển đổi Unicode 8-bit. Nó được sử dụng như một bộ mã hóa ký tự với chiều rộng biến thiên dành cho Unicode.
* Root node: là phần tử bên ngoài cùng hay còn được gọi là phần tử gốc. Nó thường là tên gọi hoặc đại diện cho thông tin lưu trữ bên trong XML.
* Element: là một dạng tên gọi nhưng sẽ dùng để chỉ các tệp con nằm trong Root node.
* Attribute: là thuộc tính của thẻ, được dùng để phân biệt đặc tính giữa 2 element trùng tên.

#### Namespace

Cấu trúc một tài liệu XML được xây dựng bởi các lập trình viên, do đó họ có thể tự đặt tên thẻ XML dẫn đến xung đột nếu trong một tệp tin bị đặt trùng tên dẫn đến không phân biệt được thẻ nào dùng cho ứng dụng nào. Vì vậy để giải quyết vấn đề này thì người ta sử dụng XML Namespace.

Cách khai báo:

* Đưa thêm thuộc tính xmlns:prefix vào bên trong phần tử gốc.
* “prefix” là đường dẫn URL của namespace, có thể là một địa chỉ nhưng phải đảm bảo duy nhất trong tài liệu XML.

#### Khái niệm về chữ ký XML

Chữ ký XML (còn được gọi là XMLDSig, XML-DSig, XML-Sig) định nghĩa một cú pháp XML cho chữ ký điện tử và được định nghĩa trong Đề xuất W3C Cú pháp và Xử lý Chữ ký XML. Về mặt chức năng, nó có nhiều điểm chung với PKCS # 7 nhưng có thể mở rộng hơn và hướng tới việc ký các tài liệu XML. Nó được sử dụng bởi các công nghệ Web khác nhau như SOAP, SAML và các công nghệ khác.

Chữ ký XML có thể được sử dụng để ký dữ liệu – tài nguyên – thuộc bất kỳ loại nào, điển hình là tài liệu XML, nhưng bất kỳ thứ gì có thể truy cập thông qua URL đều có thể được ký. Chữ ký XML được sử dụng để ký một thông tin bên ngoài tài liệu XML chứa nó được gọi là chữ ký tách rời; nếu nó được sử dụng để ký một số phần của tài liệu chứa nó, nó được gọi là chữ ký bao thư; nếu nó chứa dữ liệu đã ký bên trong nó thì nó được gọi là chữ ký được bao bọc

### *Các chuẩn về ký số, mã mật XML*

#### Tiêu chuẩn RFC-3275

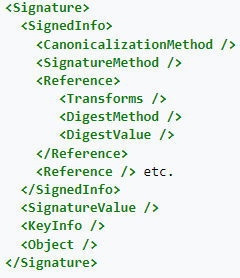


Figure 25. Cấu trúc chữ ký số của XML.

Phần tử SignedInfo chứa hoặc tham chiếu đến dữ liệu đã ký và chỉ định thuật toán nào được sử dụng.

* Phần tử SignatureMethod và CanonicalizationMethod được phần tử SignatureValue sử dụng và được đưa vào SignedInfo để bảo vệ chúng khỏi bị giả mạo.
  + Một hoặc nhiều phần tử Tham chiếu chỉ định thông tin đang được ký bởi tham chiếu URI và bất kỳ biến đổi nào sẽ được áp dụng cho thông tin trước khi ký.
  + Chuyển đổi chứa các biến đổi được áp dụng cho thông tin trước khi ký.
    - Một phép biến đổi có thể là một biểu thức xPath chọn một tập con đã xác định của cây tài liệu.
    - DigestMethod chỉ định thuật toán băm trước khi áp dụng băm.
    - DigestValue chứa kết quả được mã hóa Base64 của việc áp dụng thuật toán băm cho dữ liệu đã chuyển đổi được xác định trong các thuộc tính phần tử Tham chiếu.
* Phần tử SignatureValue chứa kết quả chữ ký được encode với thuật toán Base64 - chữ ký được tạo bằng các tham số được chỉ định trong phần tử SignatureMethod - của phần tử SignedInfo sau khi áp dụng thuật toán được chỉ định bởi CanonicalizationMethod.
* Phần tử KeyInfo tùy chọn cho phép người ký cung cấp cho người nhận khóa xác thực chữ ký, thường ở dạng một hoặc nhiều chứng chỉ số X.509. Bên phụ thuộc phải xác định khóa từ ngữ cảnh nếu KeyInfo không có mặt.

##### *Thành phần SignedInfo*

Cấu trúc của SignedInfo bao gồm thuật toán chuẩn hóa, thuật toán chữ ký và một hoặc nhiều tham chiếu. Phần tử SignedInfo có thể chứa một thuộc tính ID tùy chọn cho phép nó được tham chiếu bởi các chữ ký và đối tượng khác.

SignedInfo không bao gồm các thuộc tính chữ ký hoặc thông báo rõ ràng (chẳng hạn như thời gian tính toán, số sê-ri thiết bị mật mã, v.v.). Nếu một ứng dụng cần liên kết các thuộc tính với chữ ký hoặc thông báo, nó có thể bao gồm thông tin đó trong phần tử SignatureProperties bên trong phần tử Đối tượng.

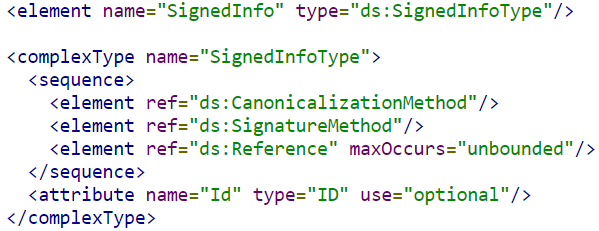


Figure 26. Cấu trúc SignedInfo.

##### *Thành phần SignatureValue*

Phần tử SignatureValue chứa giá trị thực của chữ ký điện tử và được mã hóa bằng thuật toán base64.

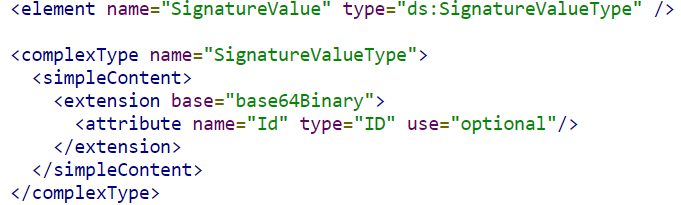


Figure 27. Cấu trúc SignatureValue.

##### *Thành phần SignatureMethod*

SignatureMethod là một phần tử bắt buộc chỉ định thuật toán được sử dụng để tạo và xác thực chữ ký. Thuật toán này xác định tất cả các hàm mật mã liên quan đến hoạt động chữ ký (ví dụ: băm, thuật toán khóa công khai, MAC, đệm, v.v.). Phần tử này sử dụng cấu trúc chung ở đây cho các thuật toán được mô tả trong phần 6.1 Số nhận dạng thuật toán và Yêu cầu triển khai. Mặc dù có một số nhận dạng duy nhất, nhưng số nhận dạng đó có thể chỉ định định dạng chứa nhiều giá trị chữ ký riêng biệt.

##### *Thành phần CanonicalizationMethod*

CanonicalizationMethod là một phần tử bắt buộc chỉ định thuật toán chuẩn hóa được áp dụng cho phần tử SignedInfo trước khi thực hiện các phép tính chữ ký. Phần tử này sử dụng cấu trúc chung cho các thuật toán.

##### *Thành phần DigestValue*

DigestValue là một phần tử chứa giá trị được mã hóa của thông báo. Thông báo luôn được mã hóa bằng thuật toán base64.

##### *Thành phần DigestMethod*

DigestMethod là một phần tử bắt buộc xác định thuật toán thông báo được áp dụng cho đối tượng đã ký. Phần tử này sử dụng cấu trúc chung ở đây cho các thuật toán được chỉ định.

Nếu kết quả của việc bỏ qua URI và ứng dụng của Transforms là một tập hợp nút XPath (hoặc sự thay thế đủ chức năng được ứng dụng thực hiện) thì nó phải được chuyển đổi.Nếu kết quả của việc bỏ tham chiếu URI và áp dụng các phép biến đổi là một luồng octet, thì không có chuyển đổi nào xảy ra (các chú thích có thể xuất hiện nếu XML Canonical có Chú thích được chỉ định trong Chuyển đổi). Thuật toán thông báo được áp dụng cho các octet dữ liệu của luồng octet kết quả.

##### *Thành phần KeyInfo*

KeyInfo là một phần tử tùy chọn cho phép (các) người nhận lấy khóa cần thiết để xác thực chữ ký. KeyInfo có thể chứa khóa, tên, chứng chỉ và thông tin quản lý khóa công khai khác, chẳng hạn như phân phối khóa trong băng tần hoặc dữ liệu thỏa thuận khóa. Đặc tả này xác định một số kiểu đơn giản nhưng các ứng dụng có thể mở rộng các kiểu đó hoặc tất cả cùng thay thế chúng bằng nhận dạng khóa riêng và ngữ nghĩa trao đổi bằng cách sử dụng cơ sở không gian tên XML [XML-NAMES]. Tuy nhiên, các câu hỏi về độ tin cậy của thông tin quan trọng như (ví dụ: tính xác thực hoặc độ mạnh của nó) nằm ngoài phạm vi của đặc điểm kỹ thuật này và để lại cho ứng dụng.

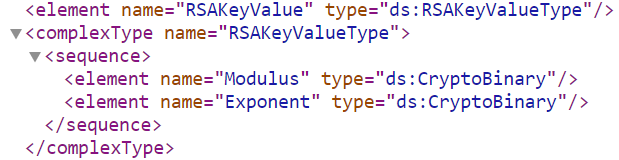


Figure 28. Thành phần KeyInfo.

#### Tiêu chuẩn RFC-6931

##### *Mã hóa XML Element*

Số thẻ tín dụng của Smith là thông tin nhạy cảm! Nếu ứng dụng muốn giữ bí mật thông tin đó, ứng dụng có thể mã hóa phần tử CreditCard:



Figure 29. Cấu trúc dữ liệu sau khi mã hóa.

##### *Mã hóa dữ liệu với khóa đối xứng*



Figure 30. Cấu trúc dữ liệu mã hóa.

* + [s1] Loại dữ liệu được mã hóa có thể được biểu diễn dưới dạng giá trị thuộc tính để hỗ trợ quá trình giải mã và xử lý tiếp theo. Trong trường hợp này, dữ liệu được mã hóa là một "phần tử". Các lựa chọn thay thế khác bao gồm 'nội dung' của một phần tử hoặc một chuỗi octet bên ngoài cũng có thể được xác định thông qua các thuộc tính MimeType và Encoding.
  + [s2] Đây (3DES CBC) là mật mã khóa đối xứng.
  + [s4] Khóa đối xứng có tên "John Smith".
  + [s6] CipherData chứa CipherValue, là một chuỗi octet được mã hóa base64. Ngoài ra, nó có thể chứa một CipherReference, là một tham chiếu URI cùng với các phép biến đổi cần thiết để có được dữ liệu được mã hóa dưới dạng một chuỗi octet

##### *Mã hóa khóa đối xứng*

Cấu trúc EncryptedData sau đây rất giống với cấu trúc ở trên, ngoại trừ lần này khóa được tham chiếu bằng cách sử dụng ds: RetrievalMethod:

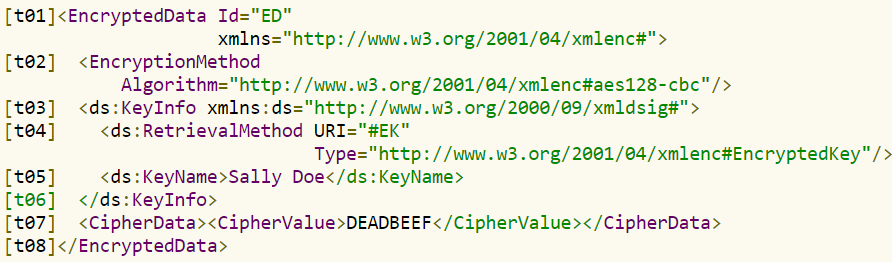


Figure 31. Cấu trúc dữ liệu khóa được mã hóa.

* [t02] Đây (AES-128-CBC) là mật mã khóa đối xứng.
* [t04] ds: RetrievalMethod được sử dụng để chỉ ra vị trí của một khóa có kiểu xenc: EncryptedKey. Khóa (AES) nằm ở '#EK'.
* [t05] ds: KeyName cung cấp một phương pháp thay thế để xác định khóa cần thiết để giải mã CipherData. Có thể sử dụng cả hai hoặc ds: KeyName và ds: KeyRetrievalMethod để xác định cùng một khóa.

#### Tiêu chuẩn XadES

Chữ ký điện tử XadES được viết tắt của XML Advanced Electronic Signatures, là một tập hợp các phần mở rộng của XML-DSig, làm cho nó phù hợp với các chữ ký điện tử nâng cao. W3C và ETSI cùng nhau duy trì và cập nhật XAdES.

Mặc dù XML-DSig là một framework chung cho các tài liệu ký điện tử, XAdES chỉ định các cấu hình chính xác của XML-DSig tuân thủ quy định eIDAS của Châu Âu (Quy định về dịch vụ nhận dạng điện tử và ủy thác cho các giao dịch điện tử trong thị trường nội bộ). Quy định eIDAS nâng cao và bãi bỏ Chữ ký Điện tử 1999/93/EC. EIDAS có giá trị pháp lý ràng buộc ở tất cả các quốc gia thành viên của Châu Âu kể từ tháng 7 năm 2014. Chữ ký điện tử được tạo ra tuân theo eIDAS có giá trị pháp lý tương tự như chữ ký viết tay.

Chữ ký điện tử, được triển khai kỹ thuật dựa trên XAdES có trạng thái của một chữ ký điện tử nâng cao. Điều này có nghĩa rằng:

* Được liên kết duy nhất với người ký kết.
* Có khả năng xác định người ký kết.
* Chỉ người ký mới có quyền kiểm soát dữ liệu được sử dụng để tạo chữ ký.
* Có thể được xác định nếu dữ liệu đính kèm với chữ ký đã được thay đổi sau khi ký.

Một thuộc tính kết quả của XAdES là các tài liệu được ký điện tử có thể vẫn có giá trị trong thời gian dài, ngay cả khi các thuật toán mật mã cơ bản bị hỏng.

Tuy nhiên, các tòa án không có nghĩa vụ phải chấp nhận chữ ký điện tử dựa trên XAdES làm bằng chứng trong quá trình tố tụng của họ; ít nhất ở EU, điều này chỉ bắt buộc đối với những chữ ký đủ tiêu chuẩn. Chữ ký điện tử đủ điều kiện cần được gắn với chứng thư số, được mã hóa bằng thiết bị tạo chữ ký bảo mật và danh tính của chủ sở hữu phải được xác minh theo mức độ đảm bảo nghiêm ngặt của quy định eIDAS.

XAdES định nghĩa ra 4 chuẩn khác nhau về mức độ bảo vệ được cung cấp:

* XAdES-B-B (Chữ ký điện tử cơ bản), phiên bản thấp nhất và đơn giản nhất chỉ chứa SignedInfo, SignatureValue, KeyInfo và SignedProperties. Biểu mẫu này mở rộng định nghĩa về chữ ký điện tử để phù hợp với chính sách chữ ký đã được xác định.
* XAdES-B-T (Chữ ký có time-stamp), liên quan đến thời điểm ký được thêm vào để chống chối bỏ.
* XAdES-B-LT (Chữ ký với Long term), chứng thư số và dữ liệu thu hồi được nhúng vào để cho phép xác minh trong tương lai ngay cả khi nguồn gốc ban đầu không có sẵn.
* XAdES-B-LTA (Chữ ký với Long term và time stamp archive), bằng cách sử dụng time stamp định kỳ (ví dụ: hàng năm), việc xâm phạm được ngăn chặn có thể do làm suy yếu các chữ ký trước đó trong thời gian lưu trữ dài hạn.

### *Cấu trúc dữ liệu XML*

#### Cấu trúc ký số

Cấu trúc để ký số cho tài liệu XML là tiêu chuẩn RFC-3275. Hạn chế của tiêu chuẩn này là mập mờ trong việc chọn chứng thư số chứa khóa công khai để giải mã chữ ký hoặc gọi là xác thực tài liệu. Sẽ rõ ràng và chính xác hơn khi RFC-3275 được kết hợp với RFC-7517 (JWK) và RFC-7516 (JWE) để người nhận có thể tin tưởng và tự động hóa hoàn toàn việc giải mã mà không phải thủ công chọn chứng thư số chứa khóa công khai.

Để làm được điều này cần chuẩn hóa thêm chữ ký Signature, sau khi ký cần thêm một thành phần con trong khóa Signature và đặt tên là SignatureVerifycation, thành phần con này được người nhận quan tâm, trích xuất và giải mã.

Cấu trúc SignatureVerification bao gồm một số thuộc tính hay còn gọi là thành phần con:

* + Use: mục đích khi sử dụng cấu trúc SignatureVerification.
  + Kty: xác định thuật toán được sử dụng để giải mã chữ ký.
  + E: giá trị exponent là một thuộc tính của khóa công khai.
  + N: giá trị n là thuộc tính của khóa công khai.

Sau khi ký số lên tài liệu XML thì ví dụ kết quả sẽ được như sau:



Quá trình xác thực

Sau khi nhận được tài liệu XML đã được ký số, người dùng sẽ lần lượt làm 3 công đoạn để xác thực:

* Trích xuất khóa công khai RSA:
  + Use: Kiểm tra mục đích sử dụng có phải là ký số.
  + Kty: kiểm tra thuật toán được sử dụng khi ký số.
  + Trích xuất các giá trị của khóa công khai exponent và n.
* Loại bỏ thành phần SignatureVerifycation để trở lại tài liệu ký nguyên thủy.
* Từ giá trị của exponent và n sẽ khởi tạo khóa công khai RSA, sau đó xác thực.

#### Cấu trúc mã mật

XML Encryption là một đặc tả, được điều chỉnh bởi khuyến nghị W3C, xác định cách mã hóa nội dung của một phần tử XML.

Mặc dù XML Encryption có thể được sử dụng để mã hóa bất kỳ loại dữ liệu nào, nhưng nó vẫn được gọi là XML Encryption bởi vì một phần tử XML (hoặc là phần tử EncryptedData hoặc EncryptedKey) chứa hoặc tham chiếu đến văn bản mật mã, thông tin khóa và thuật toán.

Cả XML Signature và XML Encryption đều sử dụng phần tử KeyInfo, phần tử này xuất hiện dưới dạng phần tử con của phần tử SignedInfo, EncryptedData hoặc EncryptedKey và cung cấp thông tin cho người nhận về tài liệu khóa nào sẽ sử dụng để xác thực chữ ký hoặc giải mã dữ liệu được mã hóa.

Phần tử KeyInfo là tùy chọn: nó có thể được đính kèm trong tin nhắn hoặc được gửi qua một kênh bảo mật.

Mã hóa XML khác và không liên quan đến Bảo mật tầng truyền tải, được sử dụng để gửi các tin nhắn được mã hóa (bao gồm cả nội dung xml, cả nội dung được mã hóa và theo cách khác) qua internet.

Mỗi thành phần sau khi được mã hóa gồm có:

* EncryptedData: trường cha sẽ bao bọc các trường, giá trị, mô tả. Chứa các thuộc tính url chỉ ra cấu trúc mã hóa theo tiêu chuẩn của W3C.
* CipherData: nêu, chỉ định ra thuật toán được sử dụng để mã hóa.
* CipherValue: nội dung của tag được mã hóa.

## **Cấu trúc dữ liệu ký số, mã mật JSON**

### *Giới thiệu chung*

#### Khái niệm

JSON là tên viết tắt của cụm từ JavaScript Object Notation, nó là một kiểu định dạng dữ liệu tuân theo một quy tắc nhất định mà hầu hết mọi ngôn ngữ lập trình hiện nay đều có thể đọc được. JSON là một trong những tiêu chuẩn mở giúp trao đổi dữ liệu có trên website.

Nói cách khách, JSON là định dạng giúp lưu trữ các thông tin có cấu trúc và nó chủ yếu được dùng để truyền tải dữ liệu giữa server và client.

#### Đặc điểm

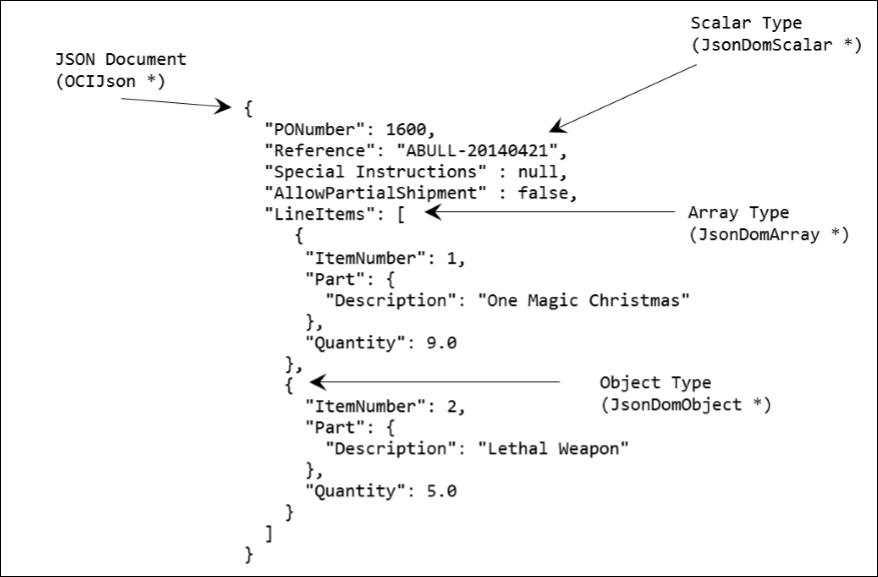


Figure 32. Đặc điểm tệp tin JSON.

JSON là sẽ sử dụng các cặp key-value để dữ liệu có thể sử dụng. Khi đó, nó sẽ hỗ trợ cho những cấu trúc dữ liệu dạng đối tượng và mảng.

#### Ưu và nhược điểm

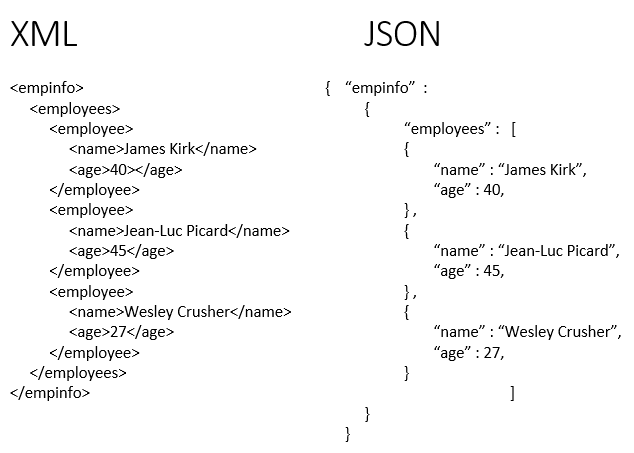


Figure 33. So sánh cấu trúc của XML và JSON.

JSON đang rất được yêu thích hiện nay bởi vì nó sở hữu những tính năng nổi bật như:

* JSON là định dạng trao đổi dữ liệu của văn bản với dung lượng nhẹ.
* JSON là một ngôn ngữ hoàn toàn độc lập.
* JSON thường có thể tự mô tả được và rất dễ hiểu.
* Tốc độ phân tích cú pháp trong JSON nhanh hơn XML.
* Vượt trội trong việc xử lý API cho các ứng dụng.

Nhược điểm: định dạng viết khá khó hiểu, trái ngược với XML, có cú pháp tương tự như HTML, giúp dễ học hơn. Ngôn ngữ ký hiệu JavaScript này dễ bị tấn công hơn, đặc biệt là trong trang web mà chưa được tin tưởng không xử lý lỗi trên yêu cầu.

### *Các chuẩn về JSON Web Token*

#### Tiêu chuẩn RFC-7515

##### *Khái niệm*

JSON Web Signature (JWS) đại diện cho nội dung được bảo mật bằng chữ ký số hoặc Mã xác thực tin nhắn (MAC) bằng cách sử dụng cấu trúc dữ liệu dựa trên JSON. Các thuật toán và số nhận dạng mật mã để sử dụng với đặc điểm kỹ thuật này được mô tả trong thông số kỹ thuật Thuật toán web JSON (JWA) riêng biệt và sổ đăng ký IANA được xác định bởi thông số kỹ thuật đó. Các khả năng mã hóa liên quan được mô tả trong thông số kỹ thuật Mã hóa web JSON (JWE) riêng biệt.

##### *Cấu trúc*

JWS đại diện cho nội dung được ký số hoặc MACed bằng cách sử dụng cấu trúc dữ liệu JSON và mã hóa base64url. Các cấu trúc dữ liệu JSON này chứa khoảng trắng và / hoặc ngắt dòng trước hoặc sau bất kỳ JSON hoặc ký tự cấu trúc nào:

* JOSE Header.
* JWS Payload.
* JWS Signature.

Các thành viên JOSE Header là sự kết hợp của các thành viên của những giá trị này:

* JWS Protected Header.
* JWS Unprotected Header.

Kết quả cho ra của quá trình serialization phép nối bởi dấu chấm các các chuỗi mã base64 là:

* BASE64URL(UTF8(JWS Protected Header)).
* BASE64URL(JWS Payload).
* BASE64URL(JWS Signature).

##### *Cấu trúc JOSE Header*

Trong cấu trúc này gồm có:

* Alg: xác định thuật toán mật mã được sử dụng để bảo mật JWS. Giá trị Chữ ký JWS không hợp lệ nếu giá trị "alg" không đại diện cho thuật toán được hỗ trợ hoặc nếu không có khóa để sử dụng với thuật toán đó được liên kết với bên đã ký kỹ thuật số hoặc MACed nội dung. các giá trị "alg" phải được đăng ký trong sổ đăng ký "Thuật toán mã hóa và chữ ký web JSON" của IANA do [JWA] thiết lập hoặc là giá trị chứa Tên chống va chạm. Giá trị "alg" là một chuỗi ASCII phân biệt chữ hoa chữ thường chứa giá trị StringOrURI. Tham số Header này PHẢI có mặt và PHẢI được hiểu và xử lý bởi các triển khai.
* Jku: là một URI [RFC3986] đề cập đến một tài nguyên cho một tập hợp các khóa công khai được mã hóa JSON, một trong số đó tương ứng với khóa được sử dụng để ký kỹ thuật số JWS. Các khóa PHẢI được mã hóa dưới dạng Bộ JWK [JWK]. Giao thức được sử dụng để có được tài nguyên PHẢI cung cấp khả năng bảo vệ toàn vẹn; một yêu cầu HTTP GET để truy xuất Bộ JWK PHẢI sử dụng Bảo mật lớp truyền tải.
* Jwk: là tiêu đề "jwk" (JSON Web Key) là khóa công khai tương ứng với khóa được sử dụng để ký kỹ thuật số JWS. Khóa này được biểu diễn dưới dạng Khóa web JSON [JWK]. Việc sử dụng Tham số Tiêu đề này là TÙY CHỌN.
* X5c: là URI đề cập đến tài nguyên cho chứng chỉ khóa công khai X.509 hoặc chuỗi chứng chỉ tương ứng với khóa được sử dụng để ký kỹ thuật số JWS.
* X5u: chứa chứng chỉ khóa công khai X.509 hoặc chuỗi chứng chỉ tương ứng với khóa được sử dụng để ký kỹ thuật số JWS.
* X5t: là dấu hiệu thu nhỏ SHA-1 được mã hóa base64url (còn gọi là thông báo kỹ thuật số) của mã hóa DER của chứng chỉ X.509 tương ứng với khóa được sử dụng để ký kỹ thuật số JWS.
* Typ: được các ứng dụng JWS sử dụng để khai báo loại phương tiện [IANA.MediaTypes] của JWS hoàn chỉnh này.

#### Tiêu chuẩn RFC-7516

##### *Khái niệm*

Mã hóa web JSON (JWE) đại diện cho nội dung được mã hóa bằng cách sử dụng cấu trúc dữ liệu dựa trên JSON. Các thuật toán mật mã và để sử dụng với đặc điểm kỹ thuật này được mô tả trong thông số kỹ thuật Web JSON (JWA) riêng biệt và các đăng ký IANA được xác định bởi thông số kỹ thuật đó. Các khả năng liên quan đến chữ ký điện tử và Mã xác thực thư (MAC) được mô tả trong thông số kỹ thuật Chữ ký Web JSON (JWS) riêng biệt.

Tổng quan, JWE đại diện cho nội dung được mã hóa bằng cách sử dụng cấu trúc dữ liệu JSON và mã hóa Base64. Các cấu trúc dữ liệu JSON này có thể chứa khoảng trắng và / hoặc ngắt dòng trước hoặc sau bất kỳ giá trị hoặc ký tự cấu trúc JSON. JWE đại diện cho các giá trị logic này:

* JOSE Header: với giá trị JWE Per-Recipient Unprotected Header.
* JWE Encrypted Key: với giá trị BASE64URL (Khoá mã hoá JWE).
* JWE Initialization Vector: với giá trị BASE64URL (Vectơ khởi tạo JWE).
* JWE AAD: với giá trị BASE64URL (JWE AAD).
* JWE Ciphertext: với giá trị BASE64URL (JWE Ciphertext).
* JWE Authentication Tag: với giá trị BASE64URL (Thẻ xác thực JWE).

##### *Tạo JWE*

Trong JWE Compact Serialization, JWE được đại diện với các thành phần sau được nối lại với dấu chấm “.”:

* BASE64URL(UTF8(JWE Protected Header)).
* BASE64URL(JWE Encrypted Key).
* BASE64URL(JWE Initialization Vector).
* BASE64URL(JWE Ciphertext).
* BASE64URL(JWE Authentication Tag).

Ví dụ: Content Encryption Key và IV là khóa của thuật toán đối xứng mã hóa dữ liệu và được bảo vệ bằng mật mã khóa công khai RSAES-OAEP.

#### Tiêu chuẩn RFC-7517

##### *Khái niệm*

JSON Web Key (JWK) là cấu trúc dữ liệu Ký hiệu đối tượng JavaScript (JSON) đại diện cho một khóa mật mã. Đặc tả này cũng xác định cấu trúc dữ liệu JWK Set JSON đại diện cho một tập hợp các JWK. Các thuật toán và số nhận dạng mật mã để sử dụng với đặc điểm kỹ thuật này được mô tả trong thông số kỹ thuật Thuật toán web JSON (JWA) riêng biệt và các sổ đăng ký IANA được thiết lập bởi thông số kỹ thuật đó.

##### *Định đạng JWK*

Các thuộc tính nằm trong cấu trúc JWK như sau:

* Kty: xác định họ thuật toán mật mã được sử dụng với khóa, chẳng hạn như "RSA" hoặc "EC".
* Use: xác định mục đích sử dụng của khóa công khai. Tham số "use" được sử dụng để cho biết liệu khóa công khai có được sử dụng để mã hóa dữ liệu hoặc xác minh chữ ký trên dữ liệu hay không. giá trị được xác định bởi thông số kỹ thuật này là:
  + sig (chữ ký).
  + enc (mã hóa).
* Alg: xác định thuật toán được thiết kế để sử dụng với khóa.

##### *Ví dụ khóa công khai*

Ví dụ sau đây JWK Set chứa hai khóa công khai được biểu diễn dưới dạng JWK: một khóa sử dụng thuật toán Đường cong Elliptic và khóa thứ hai sử dụng thuật toán RSA. Đầu tiên chỉ định rằng khóa sẽ được sử dụng để mã hóa. Điều thứ hai chỉ định rằng khóa sẽ được sử dụng với thuật toán "RS256". Cả hai đều cung cấp một ID khóa cho các mục đích khớp khóa. Trong cả hai trường hợp, các số nguyên được biểu diễn bằng cách sử dụng mã hóa base64url của các đại diện big-endian của chúng.

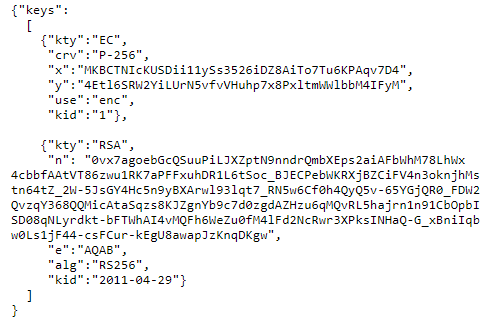


Figure 34. Minh họa cấu trúc token khóa công khai.

##### *Ví dụ khóa bí mật (mã hóa công khai)*

Ví dụ sau đây JWK Set chứa hai khóa được biểu diễn dưới dạng JWK chứa cả giá trị khóa công khai và khóa riêng: một khóa sử dụng thuật toán Đường cong Elliptic và khóa thứ hai sử dụng thuật toán RSA. Ví dụ này mở rộng ví dụ trong phần trước, thêm các giá trị khóa riêng.



Figure 35. Minh họa cấu trúc token khóa bí mật.

##### *Ví dụ khóa bí mật (mật mã đối xứng)*

Ví dụ sau JWK Set chứa hai khóa đối xứng được biểu diễn dưới dạng JWK: một khóa được chỉ định để sử dụng với thuật toán AES Key Wrap và khóa thứ hai là khóa HMAC.

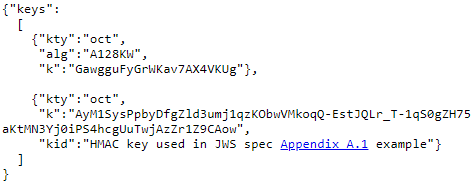


Figure 36. Minh họa cấu trúc khóa bí mật (mật mã đối xứng).

#### Tiêu chuẩn RFC-7518

##### *Khái niệm*

JWA - thông số kỹ thuật này đăng ký các thuật toán và mã nhận dạng mật mã được sử dụng với các thông số kỹ thuật Chữ ký web JSON (JWS), Mã hóa web JSON (JWE) và Khóa web JSON (JWK). Nó xác định một số đăng ký IANA cho những số nhận dạng này.

xác định một số đăng ký IANA cho những số nhận dạng này. Tất cả các thông số kỹ thuật này sử dụng cấu trúc dữ liệu [RFC7159] dựa trên JSON. Đặc tả này cũng mô tả ngữ nghĩa và hoạt động cụ thể cho các thuật toán và loại khóa này.

Việc đăng ký các thuật toán và mã nhận dạng tại đây, thay vì trong các thông số kỹ thuật JWS, JWE và JWK, nhằm mục đích cho phép chúng không thay đổi khi đối mặt với những thay đổi trong tập hợp các thuật toán Bắt buộc, Khuyến nghị, Tùy chọn và Không được chấp nhận theo thời gian. Điều này cũng cho phép thay đổi các thông số kỹ thuật JWS, JWE và JWK mà không làm thay đổi tài liệu này.

Các tên được xác định theo đặc tả này là ngắn vì mục tiêu cốt lõi là làm cho các biểu diễn thu được phải nhỏ gọn.

##### *Cấu trúc*

Các thuộc tính được sử dụng trong header JWA là:

* Alg: Bảng dưới đây là tập hợp các giá trị Tham số tiêu đề "alg" (thuật toán) được xác định bởi đặc tả này để sử dụng với JWS, mỗi giá trị được giải thích chi tiết hơn trong các phần sau:

|  |  |  |
| --- | --- | --- |
| HS256 | HMAC sử dụng SHA-256 | Required |
| HS384 | HMAC sử dụng SHA-384 | Optional |
| HS512 | HMAC sử dụng SHA-512 | Optional |
| RS256 | RSASSA-PKCS1-v1\_5 sử dụng sha256 | Recommended |
| RS384 | RSASSA-PKCS1-v1\_5 sử dụng sha384 | Optional |
| RS512 | RSASSA-PKCS1-v1\_5 sử dụng sha512 | Optional |
| ES256 | ECDSA sử dụng P-256 và SHA-256 | Recommended+ |
| ES384 | ECDSA sử dụng P-256 và SHA-384 | Optional |
| ES512 | ECDSA sử dụng P-256 và SHA-512 | Optional |
| PS256 | RSASSA-PSS sử dụng SHA-256 và MGF1 với SHA-256 | Optional |
| PS384 | RSASSA-PSS sử dụng SHA-256 và MGF1 với SHA-384 | Optional |
| PS512 | RSASSA-PSS sử dụng SHA-256 và MGF1 với SHA-512 | Optional |

* IV: là biểu diễn được mã hóa base64url của giá trị 96-bit IV được sử dụng cho hoạt động mã hóa khóa. Tham số Header này PHẢI có mặt và PHẢI được hiểu và xử lý bởi các triển khai khi các thuật toán này được sử dụng.

#### Tiêu chuẩn RFC-7519

JSON Web Token (JWT) là một phương tiện URL an toàn, nhỏ gọn để đại diện cho các tuyên bố được chuyển giữa hai bên. Các xác nhận quyền sở hữu trong JWT được mã hóa dưới dạng đối tượng JSON được sử dụng làm trọng tải của cấu trúc Chữ ký web JSON (JWS) hoặc như bản rõ của cấu trúc Mã hóa web JSON (JWE), cho phép các xác nhận quyền sở hữu được ký điện tử hoặc được bảo vệ toàn vẹn với Mã xác thực tin nhắn (MAC) và / hoặc được mã hóa.

##### *Cấu trúc*

JWT bao gồm 3 thành phần:

* Header.
* Payload.
* Signature.

Header: là một cấu trúc JSON bao gồm hai phần chính, sẽ được mã hóa Base64 trở thành một chuỗi:

* Typ: loại token (mặc định là JWT – cho biết đây là một Token JWT)
* Alg: thuật toán đã dùng để mã hóa (HMAC SHA256 – HS256 hoặc RSA).

Payload: là nơi chứa các nội dung của thông tin (claim). Thông tin truyền đi có thể là mô tả của 1 thực thể (ví dụ như người dùng) hoặc cũng có thể là các thông tin bổ sung thêm cho phần Header. Chúng được chia làm 3 loại: reserved, public và private:

* Reserved: là những thông tin đã được quy định ở trong IANA JSON Web Token Claims registry. Những thông tin này không có cái nào là bắt buộc cả. Tuy nhiên tùy vào từng ưng dụng triển khai mà hãy ràng buộc yêu cầu bắt buộc đối với những thông tin cần thiết:
  + iss (issuer): tổ chức phát hành token (không bắt buộc).
  + sub (subject): chủ đề của token (không bắt buộc).
  + aud (audience): đối tượng sử dụng token (không bắt buộc).
  + exp (expired time): thời điểm token sẽ hết hạn (không bắt buộc).
  + nbf (not before time): token sẽ chưa hợp lệ trước thời điểm này.
  + iat (issued at): thời điểm token được phát hành, tính theo UNIX time.
  + jti: JWT ID.
* Public: Khóa có thể define tùy theo ý muốn của người sử dụng JWT. Tuy nhiên để tránh trùng lặp, khó nên được quy định ở trong IANA JSON Web Token Registry hoặc là 1 URI có chứa không gian tên không bị trùng lặp.
* Private: Phần thông tin thêm dùng để truyền qua giữa các client.

Signature: là phần chữ ký được tạo bằng cách kết hợp 2 phần Header + Payload, rồi mã hóa nó lại bằng 1 giải thuật encode bất kỳ.

##### *Luồng sử dụng JWT*

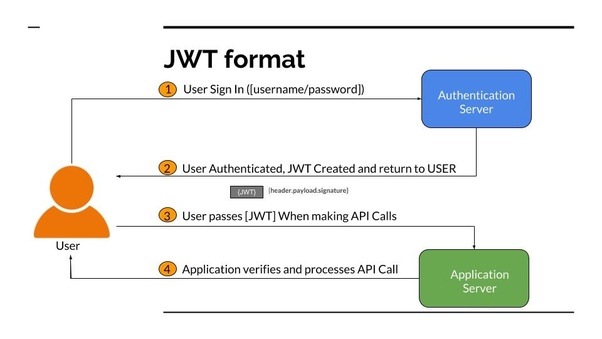


Figure 37. Luồng xử lý của JWT.

* Bước 1: user thực hiện login bằng cách gửi id/password hay sử dụng các tài khoản mạng xã hội lên phía Authentication Server (Server xác thực)
* Bước 2: authentication Server tiếp nhận các dữ liệu mà User gửi lên để phục vụ cho việc xác thực người dùng. Trong trường hợp thành công, Authentication Server sẽ tạo một JWT và trả về cho người dùng thông qua response.
* Bước 3: người dùng nhận được JWT do Authentication Server vừa mới trả về làm “chìa khóa” để thực hiện các “lệnh” tiếp theo đối với Application Server.
* Bước 4: application Server trước khi thực hiện yêu cầu được gọi từ phía User, sẽ verify JWT gửi lên. Nếu OK, tiếp tục thực hiện yêu cầu được gọi.

### *Cấu trúc dữ liệu JSON*

#### Cấu trúc ký số

Vấn đề phát sinh khi máy chủ gửi token về cho máy khách, thông tin này được truyền qua mạng máy tính và có thể bị giả mạo hoặc can thiệp bởi kẻ tấn công.

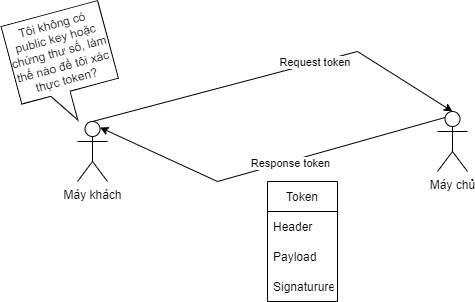


Figure 38. Vấn đề gặp phải trong xác thực.

Sử dụng tiêu chuẩn RFC-7519 để tạo ra JSON Web Token, nhưng sẽ gặp một bất cặp rằng không rõ ràng và chính xác trong việc chọn chứng thư số để giải mã chữ ký hoặc gọi là xác thực, để khác phục được nhược điểm này, RFC-7517 đã trang bị thêm một số thuộc tính trong token header nhằm giúp chính xác trong việc xác minh cũng như tự động hóa nhờ vào thông tin và tính chất của khóa công khai được lưu trong header.

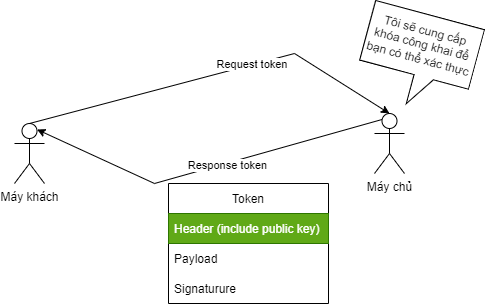


Figure 39. Khắc phục vấn đề xác thực minh bạch.

Cấu trúc của header bao gồm:

* alg: thuật toán được chỉ định cho việc ký số.
* typ: loại token JWT chính là tiêu chuẩn RFC-7519.
* use: thuộc tính được nhắc đến trong tiêu chuẩn RFC-7517, chỉ định mục đích.
* tty: chỉ ra thuật toán của khóa để xác thực.
* e: giá trị exponent của thuật toán RSA để xác thực.
* n: giá trị n của thuật toán RSA để xác thực.

|  |
| --- |
| header = new {  alg = "RS256",  typ = "JWT",  // RFC7517  use = "sig",  kty ="RSA",  e = publicKey.Exponent.ToString(),  n = publicKey.Modulus.ToString() }; |

Sau khi người nhận nhận được token từ phía máy chủ gửi về, người dùng có thể dễ dàng trích xuất khóa công khai RSA để xác thực nội dung của máy chủ, việc mô tả rõ ràng trong header sẽ bảo đảm được sự an toàn và chính xác trong vấn đề chọn đối tượng để xác thực.

#### Cấu trúc để mã mật

Xuất phát từ nhu cầu mã hóa các thông tin quan trọng ví dụ như các thông tin trong thẻ tín dụng và sử dụng sự tiện lợi của cấu trúc lưu trữ, truyền tải JSON, … các nhu cầu này đã tạo nên tiêu chuẩn và áp dụng JWE – tiêu chuẩn RFC-7516.

Vấn đề phát sinh khi dữ liệu được mã hóa bằng mật mã khóa bí mật, tức khóa đối xứng nhưng không có bất cứ lá chắn bảo vệ nào.

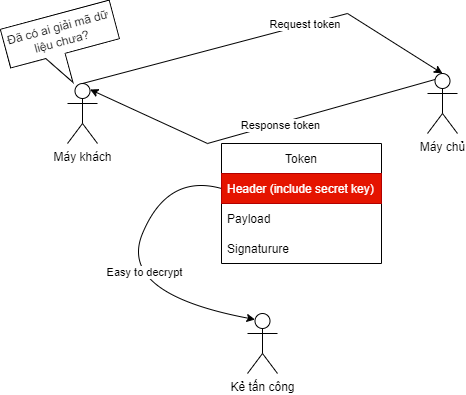


Figure 40. Vấn đề khi mã hóa JSON.

Dữ liệu vẫn sẽ được mã hóa nhưng cần có thêm lớp bảo vệ khóa đối xứng, trong việc này mật mã khóa công khai RSA đảm nhiệm vai trò truyền tải khóa được mã hóa thông qua kết nối mạng mà vẫn đảm bảo được tính bí mật của khóa đối xứng.

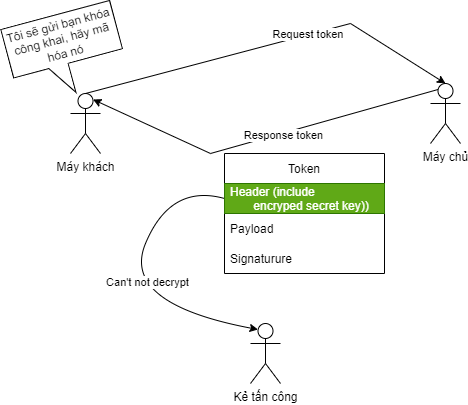


Figure 41. Khắc phục vấn đề mã hóa JSON.

Phối hợp với tiêu chuẩn RFC-7516 với sự bảo vệ khóa đối xứng bởi khóa công khai RSA-OAEP.

|  |
| --- |
| headerObject = new  {  alg="RSA-OAEP",  enc="AES256",  key= encryptedAesKeyBase64,  iv= encryptedAesIvBase64  }; |

Sau khi nhận được token từ phía người gửi, người nhận sẽ sử dụng khóa bí mật năm trong chứng thư số PKCS#12 hoặc trong windows personal certificate để giải mã có được khóa AES.

Sau khi có được khóa đối xứng, người nhận tiến hành giải mã nội dung đã được mã hóa và serialize thành cấu trúc trong lập trình, từ đó có thể truy vấn hoặc cài đặt theo các tính năng, nhu cầu mong muôn.

# **CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG DEMO**

## **Một số thư viện mật mã**

### *Thư viện mật mã Bouncy Castle*

#### Giới thiệu chung

Thư viện Bouncy Castle Crypto API là một tập hợp các triển khai các khái niệm về thuật toán mật mã. Các API của Bouncy Castle được phát triển trên cả hai ngôn ngữ lâp trình bậc cao Java và C#. Các API này có thể được tải xuống miễn phí từ trang chủ của Bouncy Castle.



Figure 42. Logo thư viện BouncyCastle.

BC là thư viện mã nguồn mở theo mô hình cấp phép linh hoạt và được sở hữu bởi một tổ chức từ thiện phần mềm có trụ sở tại Úc, Legion of the Bouncy Castle Inc. Đội ngũ phát triển BC có gần 20 năm kinh nghiệm trong lĩnh vực mật mã và họ đều là nhân viên của PrimeKey. Mã hóa và các dịch vụ bảo mật tương ứng như việc sử dụng chứng chỉ kỹ thuật số và TLS là nền tảng cho các ứng dụng và quy trình kinh doanh được kết nối ngày nay. Có thể thực hiện và duy trì các chi phí dịch vụ bảo vệ này một cách hiệu quả và an toàn theo thời gian là điều quan trọng để có được sự tin tưởng lâu dài. BC được thiết kế cho các lập trình viên sử dụng, cung cấp mã nguồn mở với các hướng dẫn triển khai đầy đủ thông qua các API hỗ trợ của nó.

#### Danh mục các lớp

Các lớp hay cấu trúc thông dụng được dùng bởi ứng dụng demo và các lập trình viên toàn cầu.

##### *DotNetUtilities*

Cấu trúc có những tính năng cho phép chuyển đổi, trích xuất dữ liệu ở định dạng giữa Microsoft và BouncyCastle:

ToX509Certificate: nhận vào là một cấu trúc chứng thư số của BouncyCastle và sẽ được chuyển đổi đầu ra là cấu trúc chứng thư số của Microsoft.

|  |
| --- |
| // mã nguồn BouncyCastle  public static SystemX509.X509Certificate ToX509Certificate( X509Certificate x509Cert)  {  return new SystemX509.X509Certificate(x509Cert.GetEncoded());  } |

FromX509Certificate: ngược lại với ToX509Certificate, đầu vào là chứng thư số của Microsoft và cho ra chứng thư số theo định dạng cấu trúc của BouncyCastle.

|  |
| --- |
| // mã nguồn BouncyCastle  public static X509Certificate FromX509Certificate(SystemX509.X509Certificate x509Cert)  {  return new X509CertificateParser().ReadCertificate(x509Cert.GetRawCertData());  } |

ToRSA: nhận một cấu trúc RSA của thư viện BouncyCastle và cho ra cấu trúc RSA của Microsoft.

|  |
| --- |
| // mã nguồn BouncyCastle  public static RSA ToRSA(RsaPrivateCrtKeyParameters privKey)  {  return CreateRSAProvider(ToRSAParameters(privKey));  }  // Cách sử dụng  DotNetUtilities.ToRSA(pkcs12Store.GetKey(keyAlias).Key as RsaPrivateCrtKeyParameters); |

GetRsaKeyPair: chuyển đổi cặp khóa RSA của Microsoft sang cặp khóa theo định dạng BouncyCastle.

|  |
| --- |
| RSACryptoServiceProvider rsa = new RSACryptoServiceProvider(512);  RSAParameters rp = rsa.ExportParameters(true);  AsymmetricCipherKeyPair kp = DotNetUtilities.GetRsaKeyPair(rp);  DotNetUtilities.ToRSA((RsaKeyParameters)kp.Public);  DotNetUtilities.ToRSA((RsaPrivateCrtKeyParameters)kp.Private); |

GetRsaPublicKey: trích xuất khóa công khai từ định dạng của Microsoft sang BouncyCastle.

|  |
| --- |
| // mã nguồn BouncyCastle  public static RsaKeyParameters GetRsaPublicKey(RSA rsa)  {  return GetRsaPublicKey(rsa.ExportParameters(false));  }  // mã nguồn BouncyCastle  public static RsaKeyParameters GetRsaPublicKey(RSAParameters rp)  {  return new RsaKeyParameters(false,  new BigInteger(1, rp.Modulus),  new BigInteger(1, rp.Exponent));  } |

##### *GeneratorUtilities*

Cấu trúc tiện ích thường được sử dụng để gọi thuật toán tạo khóa của mật mã đối xứng và công khai.

GetKeyGenerator: gọi thuật toán sinh khóa đối xứng

|  |
| --- |
| CipherKeyGenerator aesKg = GeneratorUtilities.GetKeyGenerator("AES");  aesKg.Init(new KeyGenerationParameters(new SecureRandom(), keySize));  return ParameterUtilities.CreateKeyParameter("AES", aesKg.GenerateKey()); |

GetKeyPairGenerator: gọi thuật toán sinh khóa mật mã công khai.

|  |
| --- |
| var kpGen = GeneratorUtilities.GetKeyPairGenerator("RSA");  kpGen.Init(new KeyGenerationParameters(new SecureRandom(), 1024));  AsymmetricKeyParameter privKey = kpGen.GenerateKeyPair().Private; |

##### *X509Certificate*

Cấu trúc cho biết các thông tin của chứng thư số:

IsValid, CheckValidity: kiểm tra hiệu lực của chứng chỉ thông qua thời gian.

|  |
| --- |
| // mã nguồn BouncyCastle  public virtual bool IsValid(DateTime time)  {  return time.CompareTo(NotBefore) >= 0 && time.CompareTo(NotAfter) <= 0;  } |

|  |
| --- |
| // mã nguồn BouncyCastle  public virtual void CheckValidity(DateTime time)  {  if (time.CompareTo(NotAfter) > 0)  throw new CertificateExpiredException("certificate expired on " + c.EndDate.GetTime());  if (time.CompareTo(NotBefore) < 0)  throw new CertificateNotYetValidException("certificate not valid until " + c.StartDate.GetTime());  } |

Tổng hợp tất cả thông tin về chứng thư số bao gồm: version, serial number, issuer, subject, signature, …

GetPublicKey: trích xuất khóa công khai.

|  |
| --- |
| pkcs12Store.GetCertificate(keyAlias).Certificate.GetPublicKey(); |

Verify: kiểm tra chữ ký chứng thư với đầu vào khóa công khai.

|  |
| --- |
| // mã nguồn BouncyCastle  public virtual void Verify(AsymmetricKeyParameter key)  {  CheckSignature(new Asn1VerifierFactory(c.SignatureAlgorithm, key));  } |

##### *Pkcs12Store*

Cấu trúc này cho phép xuất và nhập các chuẩn PKCS#12.

Hàm tạo hoặc Load: chỉ định đầu vào là tên chứng thư và mật khẩu để tải thông tin.

|  |
| --- |
| public static Pkcs12Store GetPkcs12Store(string pfxPath, string pwd)  {  Pkcs12Store pkcs12Store = null;  using (var keyStream = new FileStream(pfxPath, FileMode.Open, FileAccess.Read))  {  pkcs12Store = new Pkcs12Store();  pkcs12Store.Load(keyStream, pwd.ToCharArray());  }  return pkcs12Store;  } |

Alias: liệt kê các bí danh của chứng thư.

|  |
| --- |
| public static string GetAliasFromPkcs12Store(Pkcs12Store pkcs12Store)  {  return pkcs12Store.Aliases.Cast<string>().FirstOrDefault(p => pkcs12Store.IsKeyEntry(p));  } |

GetKey: sử dụng alias để lấy ra khóa bí mật.

|  |
| --- |
| AsymmetricKeyParameter privateKey = pkcs12Store.GetKey(keyAlias).Key; |

GetCertificate: sử dụng alias để lấy ra cấu trúc chứng thư X.509.

|  |
| --- |
| X509Certificate cert = pkcs12Store.GetCertificate(keyAlias).Certificate; |

##### *CmsSignedDataStreamGenerator*

Cấu trúc cho phép ký số mã mật CMS ở chế độ Stream phù hợp yêu cầu tốc độ và dữ liệu lớn.

* AddSigner: nhận khóa công khai, thông tin chứng thư X.509 và thuật toán tóm tắt thông điệp.
* Open: trả về một stream ký số.

|  |
| --- |
| using (FileStream signedStream = new FileStream(signedFile, mode: FileMode.Create, access: FileAccess.Write))  using (FileStream originDataStream = new FileStream(originalFile, FileMode.Open, access: FileAccess.Read))  {  var pkcs12Store = …// load PKCS#12  var keyAlias = …// get alias  privateKey = …// get private key  bouncycastleCert = pkcs12Store.GetCertificate(keyAlias).Certificate;  var gen = new CmsSignedDataStreamGenerator();  gen.AddSigner(privateKey: privateKey, cert: bouncycastle\_cert, CmsSignedDataGenerator.DigestSha256);  using (Stream signingStream = gen.Open(signedStream, true))  {  // sign  originDataStream.CopyTo(signingStream);  }  } |

##### *CmsSignedDataParser*

Cấu trúc chứa thông tin dữ liệu đã được ký:

* Cấu trúc này cần khởi tạo 2 stream: CmsTypedStream là tham số đầu vào với dữ liệu gốc. Stream là tham số thứ hai với dữ liệu được tạo.
* GetSignerInfos: liệt kê ra thông tin người ký.

##### *SignerInformation*

Cấu trúc chứa thông tin người ký, nhờ vào thông tin này để chứng thực chữ ký, nội dung ký.

* Verify với khóa công khai: xác minh với khóa công khai, tính năng này làm tròn nhiệm vụ xác minh nhưng vẫn còn thiếu xót.
* Verify với chứng thư X.509: xác minh với chứng thư số, tính năng rất tốt ngoài việc xác minh còn kiểm tra hiệu lực của chứng thư.

|  |
| --- |
| SignerInformation signer = signers.FirstOrDefault();  if (signer.Verify(bouncycastleCert))  {  // signature ok  byte[] expectedDigest = signer.GetContentDigest();  if (Org.BouncyCastle.Utilities.Arrays.AreEqual(digest, expectedDigest))  {  // data ok  result = true;  }  else  {  // fake data  throw new Exception("Verify data fail");  }  } |

##### *CmsEnvelopedDataStreamGenerator*

Cấu trúc này dùng cho việc mã hóa dữ liệu mã mật CMS với mật mã khóa bí mật tức khóa đối xứng. Trong cấu trúc mã hóa CMS Bouncy Castle đã khởi tạo sẵn khóa bí mật để mã hóa nhưng yêu cầu thêm một số biện pháp bảo vệ khóa bí mật này. Cấu trúc này cho phép sử dụng stream nhằm phục vụ nhu cầu tốc độ nhanh và dữ liệu lớn.

* AddKekRecipient: sử dụng khái niệm Kek (Key Encryption Key) để bảo vệ khóa đối xứng, tính năng này yêu cầu một khóa đảm nhiệm vai trò Kek.
* AddKeyTransRecipient: sử dụng chứng thứ số X.509 cấu trúc BouncyCastle để bảo vệ khóa đối xứng và khóa bí mật khi cần giải mã.
* Open: khởi tạo stream để mã hóa.

|  |
| --- |
| var aesKey = ParameterUtilities.CreateKeyParameter("AES", aesKeyRaw);  byte[] kekId = new byte[] { 1, 2, 3, 4, 5 };  var cmsEncryptor = new CmsEnvelopedDataStreamGenerator();  cmsEncryptor.AddKekRecipient("AES128", aesKey, kekId);  using (var originalDataStream = new FileStream(rawFilePath, FileMode.Open, FileAccess.Read))  using (var encryptedStream = new FileStream(cipherFilePath, FileMode.Create, FileAccess.Write))  {  // encrypt  var encryptingStream = cmsEncryptor.Open(encryptedStream, CmsEnvelopedDataGenerator.Aes128Cbc);  originalDataStream.CopyTo(encryptingStream);  encryptingStream.Close();  } |

##### *CmsEnvelopedDataParser*

Cấu trúc dùng để tải dữ liệu đã mã hóa, trích xuất các thông tin và phương thức để giải mã. Với dữ liệu đầu vào là stream dữ liệu đã mã hóa.

* GetRecipientInfos: lấy ra danh sách người nhận. Từ kết quả này có thể trích xuất chi tiết mỗi người nhận và giải mã.
  + GetRecipients.
  + RecipientInformation: chứa phương thức GetContentStream trả về một stream giải mã.

|  |
| --- |
| KeyParameter aesKey = ParameterUtilities.CreateKeyParameter("AES", aesKeyRaw);  using (FileStream cipherStream = new FileStream(cipherFilePath, FileMode.Open, FileAccess.Read))  using (FileStream decryptedStream = new FileStream(decryptedFilePath, FileMode.Create, FileAccess.Write))  {  // import cipher  var cmsEnvelopedDataParser = new CmsEnvelopedDataParser(cipherStream);  RecipientInformationStore recipientInformationStore = cmsEnvelopedDataParser.GetRecipientInfos();  ICollection recipients = recipientInformationStore.GetRecipients();  foreach (RecipientInformation recipient in recipients)  {  // decrypt  CmsTypedStream decryptingStream = recipient.GetContentStream(aesKey);  decryptingStream.ContentStream.CopyTo(decryptedStream);  decryptingStream.ContentStream.Close();  }  } |

### *Thư viện mật mã Microsoft Crypto API*

#### Giới thiệu chung

Thư viện CryptoAPI của Microsoft cho phép lập trình viên bảo mật ứng dụng thông qua mật mã, hỗ trợ các tính năng như mã hóa, giải mã dữ liệu và ký số, xác minh dữ liệu.

Thư viện với bản gốc được thể hiện thông qua ngôn ngữ C và các tính năng của Win32.

* Sự tiện lợi là tốc độ cao nhưng với không phổ biến vì làm việc với ngôn ngữ bậc thấp và một số cú pháp khó chịu của Microsoft WinAPI.
* Ngôn ngữ Microsoft C# đã hỗ trợ chuyển thể CryptoAPI để dễ dàng tiếp cận và sử dụng cho mọi lập trình viên.

#### Danh mục các lớp

##### *SignedXml*

Cấu trúc này được xây dựng bởi Microsoft với phương thức để tính toán cho ra chữ ký số, các thông tin quan trọng cần được chú ý tới trong cấu trúc SignedXml như sau:

* Thuộc tính SigningKey: nhận khóa bí mật để ký số.
* Phương thức ComputeSignature: kích hoạt tính toán nội thân để cho ra chữ ký số.
* Phương thức GetXml: lấy ra chữ ký số.

|  |
| --- |
| SignedXml signedXml = new SignedXml(xmlDoc);  signedXml.SigningKey = privateKey;  signedXml.AddReference(reference);  signedXml.ComputeSignature();  XmlElement xmlDigitalSignature = signedXml.GetXml(); |

##### *X509Certificate2UI*

Cấu trúc này cho phép người dùng thao tác với chứng thư số thông qua một số giao diện được quy định sẵn bởi Microsoft.

* Phương thức SelectFromCollection: trình diễn giao diện cho người dùng chọn chứng thư số.
* Phương thức DisplayCertificate: trình diễn giao diện thuộc tính của một chứng thư số.

|  |
| --- |
| var store = new X509Certificates.X509Store("MY", StoreLocation.CurrentUser);  store.Open(OpenFlags.ReadOnly | SOpenFlags.OpenExistingOnly);  var collection = store.Certificates;  collection = X509Certificate2UI.SelectFromCollection(collection, "Select", "Select a certificate", X509SelectionFlag.MultiSelection);  var microsoftCert = collection[0]; |

##### *EncryptedXml*

Cấu trúc gồm những chức năng mã hóa tài liệu XML thông qua khóa đối xứng.

* Phương thức EncryptData: dùng để mã hóa thành phần trong tài liệu XML.
* Phương thức ReplaceElement: được dùng để thay thế thành phần hay dữ liệu gốc bằng dữ liệu đã mã hóa.

|  |
| --- |
| byte[] encryptedBytes = xmlEncryptor.EncryptData(elementToEncrypt, symAlgo, false);  var encryptedDataElem = new EncryptedData();  encryptedDataElem.Type = EncryptedXml.XmlEncElementUrl;  encryptedDataElem.EncryptionMethod = new EncryptionMethod(algo);  encryptedDataElem.CipherData.CipherValue = Encoding.UTF8.GetBytes(Convert.ToBase64String(encryptedBytes));  var target = Tuple.Create(elementToEncrypt, encryptedDataElem);  EncryptedXml.ReplaceElement(target.Item1, target.Item2, false); |

## **Phân tích thiết kế chương trình**

### *Giới thiệu chung về ứng dụng*

Chương trình ký số nhằm đáp ứng nhu cầu đóng dấu chữ ký một cách tiện lợi, có thể áp dụng cùng được nhiều tệp tin cùng một lúc, ngoài ra chương trình cho phép mã hóa và giải mã tài liệu.

Chương trình được viết bởi ngôn ngữ C# và công nghệ giao diện Windows Form, ngoài ra module hạt nhân được phân tách riêng biệt thành các tệp tin thư viện DLL giúp dễ dàng cho việc quản lý, nâng cấp hoặc sửa đổi mà không ảnh hưởng giữa các giao diện, vì thế có thể dễ dàng nâng cấp từ công nghệ Winform sang công nghệ mới nhất hiện nay của Microsoft là Windows Presentation Foundation (WPF).

Chương trình ký số sử dụng mã mật CMS, JSON và XML là một trong những cấu trúc được sử dụng phổ biến trên thế giới, cho phép tùy chọn các chứng thư số được liệt kê trong kho tài nguyên chính chủ của Microsoft. Ngoài ra chương trình còn áp dụng tiêu chuẩn quốc tế RFC để giải quyết các vấn đề nhập nhằng trong logic giữa truyền và nhận dữ liệu, nhằm đáp ứng hay tiền đề cho việc áp dụng giao tiếp xuyên suốt giữa ứng dụng người dùng và máy chủ một cách an toàn.

### *Phân tích*

#### Xác định các tác nhân và trường hợp sử dụng

##### *Tác nhân (Actor)*

###### *Khái niệm*



Figure 43. Minh họa tác nhân (actor).

Là đối tượng bên ngoài tương tác với hệ thống theo 3 hình thức:

* Tương tác trao đổi thông tin với hệ thống hoặc sử dụng chức năng.
* Cung cấp đầu vào hoặc nhận thông tin đầu ra từ hệ thống.
* Không điều khiển hoạt động của hệ thống.

###### Xác định tác nhân

Tác nhân trong chương trình ký số là người dùng sử dụng các tính năng trong chương trình: mở tệp tin, đóng chương trình, ký số, xác thực, mã hóa, giải mã, …



Figure 44. Xác định tác nhân trong chương trình.

##### *Trường hợp sử dụng (UseCase)*

###### *Khái niệm*

UC được xem là chức năng của hệ thống cung cấp từ quan điểm của người dùng.

UC dùng để mô tả hệ thống mới về mặt chức năng, mỗi một chức năng sẽ được biểu diễn như một hoặc nhiều UC.

Không phải là thiết kế, cài đặt mà là một phần của vấn đề cần giải quyết.



Figure 45. Xác định trường hợp sử dụng với tính năng chính.

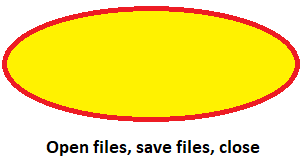


Figure 46. Xác định trường hợp sử dụng với tính năng phụ trợ.

###### *Xác định UseCase*

Những usecase trong chương trình gồm:

* Mở tệp tin, lưu tệp tin, đóng ứng dụng, xem giới thiệu phần mềm (about), hướng dẫn sử dụng.
* Chọn hành động: ký số, xác thực, mã hóa, giải mã.
* Chọn chế độ mã mật: CMS, JSON, XML
* Chốt hành động: ký số, xác thực, mã hóa, giải mã.

#### Xác định các lớp và mối quan hệ

### *Thiết kế*

#### Thiết kế chức năng

##### *Mở tệp tin*

Chức năng mở tệp tin được đặt ở thanh menu bar truyền thống phù hợp với thói quen của tất cả mọi người.

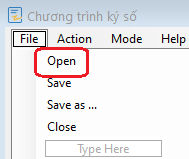


Figure 47. Giao diện tính năng mở các tệp tin.

Chức năng cho phép mở một hoặc nhiều tệp tin cùng lúc để ký số, xác thực, mã hóa hoặc giải mã.

##### *Đóng ứng dụng*

Chức năng đóng ứng dụng được đặt ở thanh menu bar truyền thống phù hợp với thói quen của tất cả mọi người.

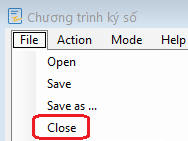


Figure 48. Giao diện tính năng thoát chương trình.

##### *Ký số mã mật CMS, XML, JSON*

Đầu vào được yêu cầu thỏa mãn để thực hiện chức năng ký số mã mật CMS như sau:

* Chấp nhận tất cả phần mở rộng cho tệp tin được ký với chế độ CMS, đối với cấu trúc XML cần có đúng định dạng.
* Đối tượng được ký có thể là một hoặc hơn.
* Tệp tin tiêu chuẩn PKCS#12 (pfx hoặc p12) và mật khẩu.
* Trình diễn chi tiết các thông tin của tệp tin PKCS#12 bao gồm:
* Phiên bản hiện hành.
* Ngày hiệu lực và ngày hết hạn
* Số Serial.
* Thumprint.
* Thuật toán của chữ ký.

Sử dụng ý tưởng của tiêu chuẩn quốc tế RFC-7517 áp dụng cho XML và JSON.

Sau khi nhân nút khởi tạo, đầu ra tệp tin đã ký tương ứng với tệp tin đầu vào và thêm phần đuôi mở rộng “.sig”.

##### *Xác thực mã mật CMS*

Tải lên chứng thư số từ kho tài nguyên chính chủ của windows bằng thư viện hệ thống mà Microsoft cung cấp cho ngôn ngữ lập trình C#, đồng thời trình diễn chi tiết thông tin chứng thư số để người dùng dễ dàng nắm bắt hoàn cảnh đang sử dụng.

##### *Xác thực mã mật XML, JSON*

Việc xác thực ngược lại với ký số ở chỗ không có đầu ra, chỉ thông báo xác minh thành công hoặc thất bại.

* Đầu vào tệp tin cần xác minh.
* Trích xuất khóa công khai đã có trong phần header để xác minh tài liệu, không tùy chọn chứng thư nhu CMS nhằm đáp ứng nhu cầu truyền và nhận dữ liệu liên tục giữa máy chủ và ứng dụng máy khách một cách nhanh, xuyên suốt và liên tục.

##### *Mã hóa và giải mã mã mật CMS, XML*

Đầu vào bao gồm:

* Tệp tin dữ liệu cần mã hóa không giới hạn đuôi mở rộng, đối với tệp tin cần giải mã phải có đuôi mở rộng nhận biết “.enc”.
* Mật khẩu mã hóa và tùy chọn kích thước khóa đối xứng AES.

Đầu ra là tệp tin dữ liệu tương ứng với đầu vào và thêm đuôi mở rộng “.enc” với trường hợp giải mã, ngược lại giải mã loại bỏ đuôi mở rộng “.enc”.

##### *Mã hóa và giải mã mã mật JSON*

Mô phỏng trao đổi dữ liệu xuyên suốt giữa máy chủ và máy khách theo cách cơ bản nhất, cách này có thể được bóc tách làm một module xử lý trong máy chủ.

Áp dụng tiêu chuẩn quốc tế RFC-7516, mỗi bên sẽ có cặp khóa bất đối xứng trong chứng thư số của mình, mỗi khi gửi request thì mỗi bên đính kèm một khóa công khai để bên nhận sử dụng nó để mã hóa khóa đối xứng đã mã hóa nội dung mà bên gửi yêu cầu.

Chu trình được lặp đi lặp lại một cách liên tục mà vẫn đảm bảo được độ an toàn.

# **Kết luận**

Chữ ký điện tử và chữ ký số ngày nay đang được sử dụng khá phổ biến trong tất cả các giao dịch điển tử với những lợi ích đặc biệt mà chúng mang lại. Việc dùng chữ ký số để cam kết lời hứa của mình và điều đó không thể rút lại được giúp nhận dạng người ký vào bản cam kết.

Con người đã sử dụng các hợp đồng dưới dạng điện tử từ hơn 100 năm nay với việc sử dụng mã Morse và điện tín. Vào năm 1889, tòa án tối cao bang New Hampshire (Hoa kỳ) đã phê chuẩn tính hiệu lực của chữ ký điện tử. Tuy nhiên, chỉ với những phát triển của khoa học kỹ thuật gần đây thì chữ ký điện tử mới đi vào cuộc sống một cách rộng rãi. Vào thập niên 1980, các công ty và một số cá nhân bắt đầu sử dụng máy fax để truyền đi các tài liệu quan trọng. Mặc dù chữ ký trên các tài liệu này vẫn thể hiện trên giấy nhưng quá trình truyền và nhận chúng hoàn toàn dựa trên tín hiệu điện tử.

Tuy nhiên các mã mật lạc hậu thời xưa nói riêng và giải pháp ký số nói chung thiếu các đảm bảo mật mã học về nhận dạng người tạo ra và thiếu các kiểm tra tính toàn vẹn của dữ liệu.

Phần mềm ký số hiện nay được thiết kế rất chặt chẽ, ứng dụng rất nhiều mã mật nói chung và tiêu chuẩn quốc tế RFC nói riêng, các việc chọn và sử dụng mã mật để đáp ứng cho ra thành phẩm được bảo vệ một cách tốt nhất là rất cần thiết để tránh khỏi các tác hại rò rỉ thông tin, mạo danh người ký hoặc thay đổi dữ liệu.

Bên cạnh các kết quả mà em nghiên cứu được thì vẫn còn các vấn đề tồn tại:

* Chỉ tập trung vào phần mềm với các thao tác từ người dùng nên chưa mô hình hóa rõ nét được sự trao đổi thông tin của mã mật XML/JSON trong mô hình máy chủ và máy khách.
* Chưa sử dụng nâng cao với thiết bị ngoại vi USB Token.
* Chưa nắm hết được các yếu tố bảo mật tùy chọn của thư viện BouncyCastle.
* Chưa tìm hiểu được một vài trường hợp api của BouncyCastle bị ngăn chặn khi sử dụng đồng thời với các phần mềm chống virus của các hãng lớn như: Norton của Mỹ, Kastersky của Nga, …

# **TÀI LIỆU THAM KHẢO**

1. Nghiên cứu Hạ tầng khóa công khai (PKI) tại trang chủ Microsoft: <https://docs.microsoft.com/en-us/windows/win32/seccertenroll/public-key-infrastructure>
2. Tìm hiểu thư viện mã mật BouncyCastle tại trang chủ: <https://www.bouncycastle.org/csharp/>
3. Nghiên cứu cách sử dụng thư viện mã mật BouncyCastle C# tại github được cung cấp bởi trang chủ: <https://github.com/bcgit/bc-csharp>
4. Nghiên cứu thư viện mã mật Windows Crypto Win32: <https://docs.microsoft.com/en-us/windows/win32/seccrypto/cryptography-portal>
5. Nghiên cứu thư viện mật mã Windows Crypto C# .NET API: <https://docs.microsoft.com/en-us/dotnet/api/?view=dotnet-plat-ext-6.0>
6. Nghiên cứu tiêu chuẩn quốc tế XML RFC-3275: <https://datatracker.ietf.org/doc/html/rfc3275>
7. Nghiên cứu tiêu chuẩn quốc tế CMS RFC-5652: <https://datatracker.ietf.org/doc/html/rfc5652>
8. Nghiên cứu tiêu chuẩn quốc tế XML RFC-6931: <https://datatracker.ietf.org/doc/html/rfc6931>
9. Nghiên cứu tiêu chuẩn quốc tế JSON RFC-7515: <https://datatracker.ietf.org/doc/html/rfc7515>
10. Nghiên cứu tiêu chuẩn quốc tế JSON RFC-7516: <https://datatracker.ietf.org/doc/html/rfc7516>
11. Nghiên cứu tiêu chuẩn quốc tế JSON RFC-7517: <https://datatracker.ietf.org/doc/html/rfc7517>
12. Nghiên cứu tiêu chuẩn quốc tế JSON RFC-7518: <https://datatracker.ietf.org/doc/html/rfc7518>
13. Nghiên cứu tiêu chuẩn JSON RFC-7519: <https://datatracker.ietf.org/doc/html/rfc7519>
14. Nghiên cứu chuẩn PKCS#7 tại trang chủ Microsoft: <https://docs.microsoft.com/en-us/windows/win32/seccrypto/pkcs--7-concepts>
15. Nghiên cứu tiêu chuẩn Viện tiêu chuẩn Viễn thông Âu châu: <https://www.etsi.org/standards#Pre-defined%20Collections>
16. Nghiên cứu chuẩn PKCS#12: <https://en.wikipedia.org/wiki/PKCS_12>