## 8.6 Chứng chỉ (Certificate)

Giả sử Alice và Bob là thành viên của một mạng lớn trong đó mỗi người tham gia đều có khóa công khai và khóa bí mật cho hệ thống mật mã được chỉ định trước và/hoặc lược đồ chữ kí số. Trong tổ chức như vậy, nó là luôn luôn cần thiết để cung cấp một loại phương pháp để xác thực khóa công khai của những người khác trong hệ thống mạng. Điều này yêu cầu một loại cơ sở hạ tầng khóa công khai (được viết tắt là PKI). Nói chung, chúng ta giả định rằng có một nhà cung cấp chứng thực số tin cậy, còn đc biết đến là CA, là nhười kí các khóa công khai cho tất cả mọi người trong mạng. Khóa chứng thực (công khai) của CA, viết tắt là $ver_{CA}$, được giả định là được biết "bằng phép thuật" đối với những người trong mạng. Sự cài đặt đơn giản này có thể không hoàn toàn thực tiễn, nhưng nó cho phép chúng ta tập trung vào thiết kế của phương pháp.

Chứng chỉ cho một người nào đó trong mạng sẽ bao gồm một vài thông tin nhận dạng cho người đó (ví dụ, tên của họ, địa chỉ, v.v), khóa công khai của họ, và chữ kí của nhà cung cấp chứng thực số (CA) về thông tin đó. Chứng chỉ cho phép người dùng mạng nhận dạng tính xác thực của khóa lẫn nhau.

Giả sử, ví dụ, rằng Alice muốn nhận chứng chỉ từ bên CA mà chứa một bản sau của khóa chứng thực công khai của Alice cho phương pháp chữ kí. Khi đó các bước ở giao thức 8.1 (Protocol 8.1) sẽ được thi triển.

Chúng ta đang không chỉ rõ là làm thế nào để Alice nhận diện bản thân với bên CA, và cũng không chỉ rõ định dạng cụ thể của ID(Alice), cũng như định dạng khóa công khai và khóa bí mật của Alice. Nói chung, những chi tiết về việc thực hiện có thể thay đổi từ PKI này đến PKI khác.

Với những người biết khóa chứng thực của CA, $ver_{CA}$, họ có thể chứng thực chứng chỉ của người khác. Giả sử Bob muốn đảm bảo là khóa công khai của Alice là khóa chuẩn. Alice có thể đưa chứng chỉ của cô ấy cho Bob. Bob khi đó có thể xác thực chữ kí của CA bằng cách kiểm tra:

$$\mathbf{ver}_{CA}(\text{ID(Alice)} \,||\, \mathbf{ver}_{\text{Alice}}, s) = \text{true}.$$

Bảo mật của chữ kí đi theo lập tức từ bảo mật của phương pháp chữ kí được sử dụng bởi CA.

Như đã đề cập bên trên, mục đích xác thực chứng chỉ là để xác thực khóa công khai của một người. Bản thân chứng chỉ không cung cấp bất kì chứng minh về định danh, vì chứng chỉ chỉ cung cấp thông tin công khai. Chứng chỉ có thể phân phối và phân phối lại cho bất kì người nào, và việc sở hữu chứng chỉ không bao hàm sự sở hữu lên nó.

Một ví dụ về việc sử dụng chứng chỉ một cách minh bạch là trong trình duyệt web. Hầu hết các trình duyệt web đều được cấu hình sẵn với một tập hợp các CA (Certificate Authority) "độc lập". Có thể có khoảng 100 CA như vậy trong một trình duyệt web thông thường. Người dùng ngầm tin tưởng nhà cung cấp trình duyệt web chỉ bao gồm các CA hợp lệ trong trình duyệt.

Bất cứ khi nào người dùng kết nối đến một trang web "an toàn", trình duyệt web của người dùng sẽ tự động xác minh chứng chỉ của trang web bằng khóa công khai thích hợp đã được tải vào trình duyệt web. Đây là một trong những chức năng của giao thức Bảo mật Lớp Vận chuyển (TLS), không yêu cầu bất kỳ hành động nào từ phía người dùng. (TLS, được mô tả chi tiết hơn trong Mục 12.1.1 (Section 12.1.1), được sử dụng để thiết lập các khóa an toàn giữa trình duyệt web của người dùng và trang web.)

## 8.7. Chữ kí và Giải mã (Signing and Encrypting)

Trong phần này, chúng ta xem xét cách mà chúng ta có thể kết hợp chữ ký và mã hóa khóa công khai một cách an toàn. Theo một nghĩa nào đó, đây là phương pháp tương đương với mã hóa xác thực, một chủ đề mà chúng tôi đã đề cập trong Mục 5.5.3 (Section 5.5.3).

Có lẽ phương pháp được khuyến nghị thường xuyên nhất là gọi là **ký rồi mã hóa** (**sign-then-encrypt**). Giả sử Alice muốn gửi một tin nhắn được ký và mã hóa cho Bob (hoặc Bob có thể muốn gửi tin nhắn cho Alice). Cho một văn bản x, Alice sẽ tính toán chữ ký của mình y = $\mathbf{sig}_{Alice}(x)$, và sau đó mã hóa cả x và y bằng hàm mã hóa công khai của Bob $e_{Bob}$, thu được $z = e_{Bob}(x, y)$. Bản mã z này sẽ được truyền cho Bob. Khi Bob nhận được z, anh ta trước tiên giải mã nó bằng hàm giải mã của mình $d_{Bob}$ để lấy (x, y). Sau đó anh ta sử dụng hàm xác minh công khai của Alice để kiểm tra rằng $\mathbf{ver}_{Alice}(x, y) = true$.

Tuy nhiên, có một vấn đề nho nhỏ với cách tiếp cận này. Giả sử Bob nhận được một tin nhắn được ký và mã hóa từ Alice. Bob có thể giải mã bản mã để khôi phục tin nhắn được ký bởi Alice, tức là (x, y). Sau đó Bob xấu tính có thể mã hóa lại tin nhắn này bằng khóa công khai của người khác. Ví dụ, Bob có thể tính toán $z' = e_{Carol}(x, y)$ và gửi z' cho Carol. Khi Carol nhận được z' , cô ấy sẽ giải mã nó

Giao thức 8.2: KÝ RỒI MÃ HÓA:

Trong phần sau, ID(Alice) và ID(Bob) là các chuỗi ID công khai có độ dài cố định cho Alice và Bob, tương ứng. Giả sử Alice muốn gửi một tin nhắn được ký và mã hóa cho Bob.

1. Alice tính toán chữ ký của mình $y = \mathbf{sig}_{Alice}(x, ID(Bob))$.

2. Alice tính toán bản mã $z = e_{Bob}(x, y, ID(Alice))$, mà cô ấy gửi cho Bob.

Khi Bob nhận được z, anh ta thực hiện các bước sau:

1. Bob sử dụng khóa giải mã riêng của mình $d_{Bob}$ để giải mã bản mã z, thu được x, y và ID(Alice)).

2. Bob lấy khóa xác minh công khai của Alice $\mathbf{ver}_{Alice}$ và kiểm tra:

$$\mathbf{ver}_{Alice}((x, ID(Bob)), y) = true$$

thu được (x, y), đây là một tin nhắn hợp lệ đã được ký bởi Alice. Sự khó khăn của kịch bản này là Carol có thể tin rằng cô ấy là người nhận dự định của tin nhắn của Alice, trong khi tin nhắn thực sự dành cho

Bob. Alice cũng có thể cho rằng không ai có quyền truy cập vào văn bản x. Nhưng đây là một giả định sai lầm, vì Bob cũng biết văn bản x.

Một cách tiếp cận thay thế là cho Alice trước tiên mã hóa x, và sau đó ký kết quả (quá trình này được gọi là mã hóa rồi ký). Alice sẽ tính toán

$$z = e_{Bob}(x) \text{ and } y = \mathbf{sig}_{Alice}(z)$$

và sau đó truyền cặp (z, y) cho Bob. Bob sẽ trước tiên xác minh chữ ký y trên z bằng verAlice. Nếu chữ ký là hợp lệ, Bob sẽ giải mã z, thu được x. Tuy nhiên, giả sử Oscar chặn lại tin nhắn này và anh ta thay thế chữ ký của Alice y bằng chữ ký của riêng anh ta

$$y' = \mathbf{sig}_{Oscar}(z).$$

(Lưu ý rằng Oscar có thể ký bản mã $z = e_{Bob}(x)$ mặc dù anh ta không biết giá trị của văn bản x.) Sau đó, nếu Oscar truyền (z, y') cho Bob, chữ ký của Oscar sẽ được xác minh bởi Bob sử dụng verOscar, và Bob có thể suy ra rằng văn bản x xuất phát từ Oscar. Tất nhiên văn bản thực sự được tạo ra bởi Alice. Chúng tôi đã chỉ ra các cuộc tấn công tiềm năng chống lại cả ký rồi mã hóa và mã hóa rồi ký. Hóa ra là có thể sửa chữa cả hai phương pháp này, bằng cách sử dụng hai quy tắc sau:

1. Trước khi mã hóa một tin nhắn, nối thông tin nhận dạng cho người gửi, và

Giao thức 8.3: MÃ HÓA RỒI KÝ

Trong phần sau, ID(Alice) và ID(Bob) là các chuỗi ID công khai có độ dài cố định cho Alice và Bob, tương ứng.

Giả sử Alice muốn gửi một tin nhắn được ký và mã hóa cho Bob.

1. Alice tính toán bản mã $z = e_{Bob}(x, ID(Alice))$.
2. Alice tính toán chữ ký của mình $y = sig_{Alice}(z, ID(Bob))$ và cô ấy gửi (z, y, ID(Alice)) cho Bob.

Khi Bob nhận được (z, y, ID(Alice)), anh ta thực hiện các bước sau.

1. Bob lấy khóa xác minh công khai của Alice $ver_{Alice}$ và kiểm tra rằng $ver_{Alice}((z, ID(Bob)), y) = true$.
2. Bob sử dụng khóa giải mã riêng của mình $d_{Bob}$ để giải mã bản mã z, thu được x và ID(Alice).
3. Anh ta sau đó kiểm tra xem ID(Alice), như được tính toán ở bước 2, có khớp với giá trị ban đầu mà anh ta nhận được trong (z, y, ID(Alice)) hay không.

2. Trước khi ký một tin nhắn, nối thông tin nhận dạng cho người nhận.

Quá trình ký rồi mã hóa được sửa đổi được chi tiết trong Giao thức 8.2. Việc xây dựng một thuật toán mã hóa rồi ký sửa đổi cũng khá đơn giản; xem Giao thức 8.3.

Cả hai Giao thức 8.2 và 8.3 đều có thể được chứng minh là an toàn dưới các giả định thích hợp liên quan đến sự an toàn của các phương pháp mã hóa và chữ ký cơ bản.

Chúng ta cũng nên đề cập rằng có những ví dụ về các phương pháp chuyên biệt, được gọi là các phương pháp signcryption, kết hợp các phương pháp chữ ký và mã hóa, nhưng làm điều đó một cách hiệu quả hơn về mặt tính toán so với ký rồi mã hóa (sign-then-encrypt) hoặc mã hóa rồi ký (encrypt-then-sign).

## 8.8. Ghi chú và Tài liệu tham khảo (Notes and References)

Để có một bài khảo sát hay (nhưng đã lỗi thời) về các phương pháp chữ ký, chúng tôi khuyến nghị Mitchell, Piper và Wild [140]. Bài báo này cũng chứa hai phương pháp giả mạo chữ ký ElGamal mà chúng tôi đã trình bày trong Mục 8.3.

Phương pháp chữ ký ElGamal (*The ElGamal Signature Scheme*) được trình bày bởi ElGamal [80], và Phương pháp chữ ký Schnorr (*The Schnorr Signature Scheme*) do Schnorr [174] đưa ra. Một mô tả hoàn chỉnh về ECDSA được tìm thấy trong Johnson, Menezes và Vanstone [100]. Phương pháp chữ ký số (*The*

*Dgigtal Signature Algorithm*) được công bố lần đầu tiên bởi NIST vào tháng 8 năm 1991, và nó được áp dụng làm FIPS 186 vào tháng 12 năm 1994.

Tiêu chuẩn chữ ký số (*The Digital Signature Standard*) bao gồm RSA, DSA và ECDSA. Nó đã được cập nhật nhiều lần. Phiên bản hiện tại là FIPS 186-4 [147], được ban hành vào tháng 7 năm 2013.

Full Domain Hash do Bellare và Rogaway [22, 21] đưa ra. Bài báo [21] cũng bao gồm một biến thể hiệu quả hơn, được gọi là Phương pháp chữ ký xác suất (*Probabilistic Signature Scheme*) (PSS). Các phương pháp chữ ký ElGamal có tính an toàn có thể được chứng minh cũng đã được nghiên cứu; xem ví dụ Pointcheval và Stern [164].

Chứng chỉ được đề xuất lần đầu tiên như một phương pháp xác thực khóa công khai trong một Luận văn Cử nhân của Kohnfelder vào năm 1978 [116]. Để có một bài viết hay về cơ sở hạ tầng khóa công khai nói chung, chúng tôi khuyến nghị Adams và Lloyd [1].

Smith [186] và An, Dodis và Rabin [4] đưa ra những bài viết chi tiết về ký rồi mã hóa và mã hóa rồi ký. Các phương pháp Signcryption được phát minh bởi Zheng [207].

Một số bài tập chỉ ra những vấn đề bảo mật với các phương pháp chữ ký kiểu ElGamal nếu các giá trị "k" được sử dụng lại hoặc sinh ra theo một cách có thể dự đoán được. Có bây giờ nhiều công trình nghiên cứu theo đuổi chủ đề này; xem ví dụ Bellare, Goldwasser và Micciancio [14] và Nguyen và Shparlinski [156].