CHƯƠNG 8 - SƠ ĐỒ CHỮ KÝ

Ngày 13 tháng 10 năm 2023

Trong chương này, ta sẽ tìm hiểu về các sơ đồ chữ ký, hay còn được gọi là các chữ ký số. Ta sẽ bao hàm các sơ đồ chữ ký dựa trên các vấn đề về phân tích nhân tử và giải thuật rời rạc, bao gồm cả tiêu chuẩn về chữ ký số (Digital Signature Standard)

8.1 Giới thiệu

Chữ ký viết tay "thông thường" đính kèm vào tài liệu được sử dụng để chỉ định người chịu trách nhiệm về nó. Chữ ký được sử dụng trong các tình huống hàng ngày như viết thư, rút tiền ngân hàng, ký hợp đồng, v.v.

Sơ đồ chữ ký là phương pháp ký một thông điệp được lưu trữ dưới dạng điện tử. Như vậy, một tin nhắn đã ký có thể được truyền qua mạng máy tính. Trong chương này, chúng ta sẽ nghiên cứu một số các sơ đồ chữ ký phổ biến, tuy nhiên trước đó ta sẽ thảo luận về một số khác biệt cơ bản giữa chữ ký thông thường và chữ ký số.

Đầu tiên là vấn đề ký một văn bản. Với chữ ký thông thường, chữ ký là một phần của tài liệu vật lý được ký. Tuy nhiên, chữ ký điện tử không được gắn vật lý vào tin nhắn được ký, vì vậy thuật toán được sử dụng phải bằng cách nào đó "che giấu" chữ ký với bản tin.

Thứ hai là vấn đề xác minh. Chữ ký thông thường được xác minh bằng cách so sánh nó với các chữ ký xác thực khác. Ví dụ: nếu ai đó ký để mua hàng bằng thẻ tín dụng (ngày nay không còn phổ biến do sự phổ biến của công nghệ chip-pin), nhân viên bán hàng phải so sánh chữ ký trên phiếu bán hàng với chữ ký ở mặt sau của thẻ tín dụng để xác minh chữ ký. Tất nhiên, đây không phải là một phương pháp an toàn vì việc giả mạo chữ ký của người khác là không khó. Mặt khác, chữ ký số có thể được xác minh bằng thuật toán xác minh được biết đến rộng rãi. Vì vậy, "bất kỳ ai" cũng có thể xác minh chữ ký số. Việc sử dụng sơ đồ chữ ký vì thế sẽ ngăn ngừa khả năng giả mạo.

Một điểm khác biệt cơ bản giữa chữ ký thông thường và chữ ký số là "bản sao" của tin nhắn kỹ thuật số được ký giống hệt với bản gốc. Trong khi đó, bản sao của tài liệu giấy đã được ký thường có thể được phân biệt với bản gốc. Tính năng này có nghĩa rằng, chúng ta sẽ phải cẩn thận để tránh việc sử dụng lại bản tin kỹ thuật số đã ký. Ví dụ: nếu Alice ký một tin nhắn kỹ thuật số cho phép Bob rút 100 đô la từ tài khoản ngân hàng của cô ấy, cô ấy muốn Bob chỉ có thể làm như vậy một lần. Vì vậy, bản thân tin nhắn phải chứa thông tin, chẳng hạn như ngày tháng, để ngăn không cho nó được sử dụng lại.

Một sơ đồ chữ ký bao gồm hai thành phần: thuật toán mã hoá và thuật toán giải mã. Alice có thể mã hoá một bản tin x bằng thuật toán ký (riêng tư) sig_K phụ thuộc vào khóa riêng K. Chữ ký kết quả $sig_K(x)$ sau đó có thể được xác minh bằng thuật toán giải mã công khai verK. Cho một cặp (x, y), trong đó x là một bản tin và y là chữ ký có mục đích trên x, thuật toán xác minh trả về câu trả lời đúng hay sai tùy thuộc vào việc y có phải là chữ ký hợp lệ cho thông báo x hay không.

Dưới đây là định nghĩa đầy đủ về một sơ đồ chữ ký

Định nghĩa 8.1: Một số đồ chữ ký là một bộ năm (P, C, K, E, D), với các điều kiện sau:

- P là không gian bản tin
- C là không gian các chữ ký
- K là không gian khoá, gồm danh sách các khoá có thể sử dụng
- Với mỗi khoá $k \in K$, sẽ có một hàm mã hoá $sig_K \in E$, tương ứng với đó là một hàm giải mã $ver_K \in D$. Mọi bản tin x và mọi chữ ký y đều phải thoả mãn:

$$ver_K(x, y) = \begin{cases} true & \text{if } \ y = sig_K(x) \\ false & \text{if } \ y \neq sig_K(x) \end{cases}$$

Một cặp (x, y) với $x \in P$ và $y \in C$ được gọi là một bản tin được mã hoá (signed message)

Với mọi $k \in K$, các hàm sig_K và ver_K phải là các hàm thời gian đa thức (polynomial-time functions). Thuật toán xác minh ver_K sẽ được công khai và thuật toán ký sig_K sẽ ở chế độ riêng tư. Với một thông báo x, không ai khác ngoài Alice có thể tính toán chữ ký y sao cho $ver_K(x, y) = true$ (và lưu ý rằng có thể có nhiều hơn một y như vậy cho một x cho trước, tùy thuộc vào cách thức chức năng ver được xác định). Nếu Oscar có thể tính toán một cặp (x, y) sao cho $ver_K(x, y) = true$ và x chưa được ký bởi Alice trước đó thì chữ ký y được gọi là giả mạo. Một cách không chính thức, chữ ký giả mạo là chữ ký hợp lệ được tạo bởi người khác không phải Alice.

8.1.1 Sơ đồ chữ ký RSA

Đây là ví dụ đầu tiên mà ta sẽ tìm hiểu, quan sát xem hệ mật RSA được sử dụng để cung cấp chữ ký số như thế nào. Dưới đây là định nghĩa của hệ mật RSA

Cho $n = pq$, với p và q là hai số nguyên tố đủ lớn. Xét $P = A = Z_n$, ta có định nghĩa

$$K = \{(n, p, q, a, b) : n = pq, ab \equiv 1(mod\phi(n))\} \quad (1)$$

Ta có n và b là khoá công khai, còn p, q, và a là khoá bí mật

Với $K = (n, p, q, a, b)$, ta định nghĩa hàm mã hoá như sau

$$sig_K(x) = x^a mod n \quad (2)$$

và hàm giải mã có dạng như sau

$$ver_K(x, y) = true \Leftrightarrow x \equiv y^b(mod n) \quad (3)$$

Với $x, y \in Z_n$

Lấy ví dụ, Alice tiến hành mã hoá một tin nhắn x bằng cách sử dụng hàm giải mã RSA e_K. Alice là người duy nhất có thể tạo chữ ký vì $e_K = sig_K$ là khoá bí mật. Thuật toán giả mã sử dụng quy tắc giải hóa RSA d_K. Bất kỳ ai cũng có thể giải mã được chữ ký vì d_K được công khai.

Lưu ý rằng bất kỳ ai cũng có thể giả mạo chữ ký RSA của Alice bằng cách chọn ngẫu nhiên y và tính $x = e_K(y)$; thì $y = sig_K(x)$ là chữ ký hợp lệ trên thông điệp x. (Tuy nhiên, lưu ý rằng dường như không có cách rõ ràng nào để chọn x trước rồi tính chữ ký y tương ứng; nếu điều này có thể thực hiện được thì Hệ thống mật mã RSA sẽ không an toàn.) Một cách để ngăn chặn những cuộc tấn công này là yêu cầu các tin nhắn chứa đủ độ dư thừa để chữ ký giả mạo này không tương ứng với tin nhắn x "có ý nghĩa", ngoại trừ với một xác suất rất nhỏ. Ngoài ra, việc sử dụng hàm băm kết hợp với sơ đồ chữ ký sẽ loại bỏ phương pháp giả mạo này (hàm băm mật mã đã được thảo luận trong Chương 5). Ta sẽ tìm hiểu kĩ về cách tiếp cận này trong phần tiếp theo.

Phần còn lại của chương 8 bao gồm như sau: Phần 8.2 giới thiệu khái niệm về bảo mật cho sơ đồ chữ ký và cách sử dụng hàm băm cùng với sơ đồ chữ ký. Phần 8.3 trình bày về Sơ đồ chữ ký ElGamal và thảo luận về tính bảo mật của nó. Phần 8.4 sẽ đề cập đến ba sơ đồ chữ ký quan trọng được phát triển từ hệ mật ElGamal, cụ thể là sơ đồ chữ ký Schnorr, thuật toán chữ ký số và thuật toán chữ ký số đường cong Elliptic. Sơ đồ chữ ký có độ an toàn có thể chứng minh được gọi là Full Domain Hash là nội dung của Phần 8.5 và các chứng chỉ sẽ được thảo luận trong Phần 8.6. Cuối cùng, một số phương pháp kết hợp sơ đồ chữ ký với sơ đồ mã hóa sẽ được thảo luận và xem xét trong Phần 8.7.

8.2 Yêu cầu về bảo mật cho các sơ đồ chữ ký

Ở phần này, ta sẽ thảo luận về việc một hệ mật như thế nào thì được coi là bảo mật. Ta sẽ cần phải chị định mô hình tấn công, mục đích của kẻ tấn công, và dạng bảo mật được cung cấp bởi sơ đồ chữ ký.

Nhắc lại một chút ở phần 2.2, một mô hình tấn công sẽ được thực hiện dựa theo những thông tin công khai. Đối với trường hợp của sơ đồ chữ ký, các dạng tấn công phổ biến bao gồm:

Tấn công chỉ bằng khoá công khai

Ví dụ, Oscar có được khoá hay chữ ký công khai của Alice thông qua hàm giải mã e_K hay ver_K

Tấn công bằng bản tin đã biết

Ví dụ, Oscar đã có được danh sách một số bản tin trước đã được Alice mã hoá như là

$$(x_1, y_1), (x_2, y_2), ...,$$

Tấn công với các bản tin được chọn trước

Ví dụ, Oscar yêu cầu các chữ ký của Alice đối với danh sách một số bản tin nhất định. Từ đó, anh ấy sẽ chọn các bản tin $x_1, x_2, ...$, và Alice sẽ cung cấp các chữ ký của cô ấy lần lượt là $y_i = sig_K(x_i), i = 1, 2, ...$

Ta sẽ xem xét một vài mục đích của việc tấn công

Bẻ khoá toàn bộ (total broken)

Oscar có thể xác định được khoá bí mật của Alice hay là hàm mã hoá sig_K. Từ đó anh ấy có thể tạo ra các chữ ký khác với bất kì bản tin nào.

Giả mạo có chọn lọc (selective forgery)

Với một số xác suất không thể bỏ qua, Oscar có thể tạo một chữ ký hợp lệ trên một tin nhắn do người khác chọn. Nói cách khác, nếu Oscar nhận được một bản tin x thì anh ta có thể xác định (với xác suất nào đó) chữ ký y sao cho $ver_K(x, y) = true$. Tin nhắn x không được là tin nhắn đã được Alice ký trước đó.

Giả mạo hiện sinh (existential forgery)

Oscar có thể tạo chữ ký hợp lệ cho ít nhất một tin nhắn. Nói cách khác, Oscar có thể tạo một cặp (x, y), trong đó x là một thông điệp và $ver_K(x, y) =$

true, còn bản tin x không phải là bản tin đã được Alice ký trước đó.

Một sơ đồ chữ ký sẽ không thể có đủ độ bảo mật, nếu như Oscar có thể kiểm tra một chữ ký khả dụng $y \in A$ với một bản tin x được cho trước bằng cách sử dụng hàm giải mà ver_K, cho đến khi anh ấy tìm ra một chữ ký có nghĩa. Vậy với khoảng thời gian đủ lâu thì Oscar sẽ luôn tìm ra được chữ ký của Alice cho mọi bản tin. Vì thế, với các hệ mật có khoá công khai, mục tiêu của chúng ta sẽ là tìm ra các sơ dồ chữ ký có khả năng tính toán lớn và đảm bảo về bảo mật.

Lưu ý rằng các định nghĩa trên có một số điểm tương đồng với các cuộc tấn công vào MAC mà chúng ta đã xem xét trong Phần 5.5. Trong cài đặt MAC, không có thứ gọi là khóa chung, vì vậy sẽ không có ý nghĩa gì khi nói về một cuộc tấn công chỉ bằng khóa (và tất nhiên, MAC không có chức năng ký và xác minh riêng biệt). Các cuộc tấn công trong Phần 5.5 là các cuộc tấn công giả mạo hiện hữu bằng cách sử dụng các cuộc tấn công tin nhắn đã chọn.

Ta minh họa các khái niệm được mô tả ở trên bằng một số cuộc tấn công vào Lược đồ Chữ ký RSA. Trong Phần 8.1, chúng ta đã quan sát thấy Oscar có thể xây dựng một thông điệp được ký hợp lệ bằng cách chọn chữ ký y và sau đó tính x sao cho $ver_K(x, y) = true$. Đây sẽ là một sự giả mạo bằng cách sử dụng một cuộc tấn công chỉ bằng khóa công khai. Một kiểu tấn công khác dựa trên đặc tính nhân của RSA mà ta đã đề cập trong Phần 6.9.1. Giả sử $y_1 = sig_K(x_1)$ và $y_2 = sig_K(x_2)$ là hai thông điệp bất kỳ được Alice ký trước đó. Sau đó

$$ver_K(x_1x_2 mod n, y_1y_2 mod n) = true,$$

và do đó Oscar có thể tạo chữ ký hợp lệ $y_1y_2 \ mod n$ trên bản tin $x_1x_2 \ mod n$. Đây là một ví dụ về sự giả mạo hiện hữu bằng cách sử dụng một cuộc tấn công tin nhắn đã biết.

Sau đây là một dạng khác. Giả sử Oscar muốn giả mạo chữ ký trên tin nhắn x, trong đó x có thể đã được người khác chọn. Việc tìm $x1, x2 \in Z_n$ sao cho

$$x \equiv x_1x_2 (mod n)$$

là một vấn đề đơn giản. Bây giờ, giả sử anh ta yêu cầu Alice cho chữ ký của cô trên bản tin x_1 và x_2, chúng ta ký hiệu lần lượt là y_1 và y_2. Sau đó, như trong cuộc tấn công trước, $y_1y_2 \ mod n$ là chữ ký cho bản tin $x = x_1x_2 mod n$. Đây là hành vi giả mạo có chọn lọc bằng cách sử dụng một cuộc tấn công bằng tin nhắn đã chọn.

8.2.1 Chữ ký và hàm băm

Sơ đồ chữ ký hầu như luôn được sử dụng cùng với hàm băm mật mã (công khai) an toàn. Hàm băm $h : \{0, 1\}^* \to Z$ sẽ nhận một thông báo có độ dài tùy ý và tạo ra một bản tóm tắt thông báo có kích thước xác định (224 bit là lựa chọn phổ biến). Sau đó, bản tóm tắt thông điệp sẽ được ký bằng sơ đồ chữ ký (P, C, K, S, V), trong đó $Z \subseteq P$. Việc sử dụng hàm băm và sơ đồ chữ ký này

được mô tả dưới dạng sơ đồ trong Hình 8.1.

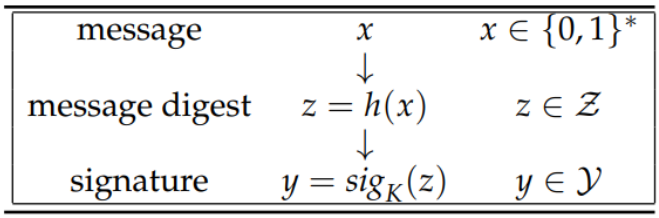

FIGURE 8.1: Signing a message digest

Giả sử Alice muốn ký một tin nhắn x, là một chuỗi bit có độ dài tùy ý. Đầu tiên cô ấy xây dựng bản tóm tắt thông điệp $z = h(x)$, sau đó tính chữ ký trên z, cụ thể là $y = sig_K(z)$. Sau đó cô ấy truyền cặp thứ tự (x, y) qua kênh. Bây giờ, việc xác minh có thể được thực hiện (bởi bất kỳ ai) bằng cách trước tiên xây dựng lại bản tóm tắt thông báo $z = h(x)$ bằng cách sử dụng hàm băm công khai h, sau đó kiểm tra xem $ver_K(z, y) = true$.

Ta sẽ phải cẩn thận rằng việc sử dụng hàm băm h không làm suy yếu tính bảo mật của sơ đồ chữ ký, vì đó là bản tóm tắt thông điệp được ký chứ không phải thông điệp. h sẽ cần phải đáp ứng một số đặc tính nhất định để ngăn chặn các cuộc tấn công khác nhau. Các thuộc tính mong muốn của hàm băm là những thuộc tính đã được thảo luận trong Phần 5.2.

Kiểu tấn công rõ ràng nhất là Oscar bắt đầu bằng một tin nhắn được ký hợp lệ (x, y), trong đó $y = sig_K(h(x))$ (Cặp (x, y) có thể là bất kỳ thông điệp nào được Alice ký trước đó.). Sau đó, anh ta tính $z = h(x)$ và cố gắng tìm $x' \neq x$ sao cho $h(x') = h(x)$. Nếu Oscar có thể làm điều này, (x', y) sẽ là một tin nhắn được ký hợp lệ, vì vậy y là chữ ký giả mạo cho tin nhắn x' . Đây là một sự giả mạo hiện hữu bằng cách sử dụng một cuộc tấn công tin nhắn đã biết. Để ngăn chặn kiểu tấn công này, ta yêu cầu h phải có khả năng chống lại hình ảnh thứ hai.

Một cuộc tấn công khác có thể xảy ra như sau: Đầu tiên Oscar tìm thấy hai tin nhắn $x \neq x'$ sao cho $h(x) = h(x')$. Sau đó, Oscar đưa x cho Alice và thuyết phục cô ký vào bản tóm tắt thông điệp $h(x)$, thu được y. Khi đó (x', y) là một tin nhắn được ký hợp lệ và y là chữ ký giả mạo cho tin nhắn x.

Đây là một hành vi giả mạo hiện sinh bằng cách sử dụng một cuộc tấn công tin nhắn đã chọn; nó có thể được ngăn chặn nếu h có khả năng chống va chạm. Đây là kiểu tấn công thứ ba. Thông thường, với một số lược đồ chữ ký nhất định, có thể giả mạo chữ ký trên các bản tóm tắt thông báo ngẫu nhiên z (ta đã quan sát thấy rằng điều này có thể được thực hiện với sơ đồ chữ ký RSA). Nghĩa là, giả định rằng sơ đồ chữ ký (không có hàm băm) có thể bị giả mạo

bằng cách sử dụng một cuộc tấn công chỉ bằng khóa công khai. Bây giờ, giả sử Oscar tính toán một chữ ký trên bản tóm tắt thông điệp z nào đó, và sau đó anh ta tìm thấy một thông điệp x sao cho $z = h(x)$. Nếu anh ta có thể làm điều này thì (x, y) là một thông điệp được ký hợp lệ và y là một chữ ký giả mạo cho thông điệp x. Đây là một sự giả mạo tồn tại trong sơ đồ chữ ký bằng cách sử dụng một cuộc tấn công chỉ bằng khóa. Để ngăn chặn cuộc tấn công này, chúng tôi mong muốn h là hàm băm kháng tiền ảnh.

8.3. Cơ chế chữ ký ElGamal (ElGamal Signature Scheme)

Trong phần này, chúng tôi trình bày về Cơ chế Chữ ký ElGamal, cơ chế này đã được mô tả trong một bài báo vào năm 1985. Một bản tinh chỉnh của cơ chế này được sử dụng trong Thuật toán Chữ ký số (Digital Signature Algorithm - DSA) bởi Viện Tiêu chuẩn và Công nghệ Quốc gia (National Institute of Standards and Technology – NIST). DSA cũng kết hợp một số ý tưởng được sử dụng trong một cơ chế gọi là cơ chế chữ ký Schnorr. Tất cả các cơ chế này được thiết kế chuyên dụng để sử dụng cho các chữ ký, không giống với hệ mật RSA - nó có thể được sử dụng đồng thời như là một hệ mật mã hóa khóa công khai và một cơ chế chữ ký.

Hệ mật 8.2: Cơ chế chữ ký ElGamal

Giả sử p là một số nguyên tố sao cho bài toán logarit rời rạc trong Z_p là bất trị, giả sử $\alpha \in Z_p^*$ là một phần tử nguyên thủy. Giả sử $P = Z_p^*$, $A = Z_p^* \times Z_{p-1}$, và định nghĩa:

$$\mathcal{K} = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod{p}\}.$$

Các giá trị p, α và β là các khóa công khai (public key), và a là khóa riêng tư (private key).

Với K = (p, α, a, β) và với một số ngẫu nhiên (bí mật) $k \in Z_{p-1}^*$, định nghĩa:

$$\mathbf{sig}_K(x, k) = (\gamma, \delta),$$

trong đó:

$$\gamma = \alpha^k \bmod p$$

và:

$$\delta = (x - a\gamma)k^{-1} \bmod (p - 1).$$

Với x, ɣ $\in Z_p^*$ và $\delta \in Z_{p-1}$, định nghĩa:

$$\mathbf{ver}_K(x,(\gamma,\delta)) = true \Leftrightarrow \beta^\gamma \gamma^\delta \equiv \alpha^x \pmod{p}.$$

Cơ chế chữ ký Elgamal là ngẫu nhiên (Hệ mật khóa công khai Elgamal cũng là ngẫu nhiên). Điều này có nghĩa là có nhiều chữ ký hợp lệ cho mỗi thông điệp, và thuật toán xác thực phải có khả năng chấp nhận bất cứ chữ ký hợp lệ nào trong số đó là chính xác. Mô tả cho cơ chế chữ ký Elgamal được đưa ra trong hệ mật 8.2.

Chúng ta hãy bắt đầu với một số quan sát dự trù. Một chữ ký Elgamal bao gồm 2 thành phần, được ký hiệu là ɣ và δ. Thành phần đầu tiên, ɣ, có được bằng cách lấy lũy thừa α theo một cơ số modulo ngẫu nhiên p; nó không phụ thuộc vào thông điệp (được đặt tên là x) đang được ký. Thành phần thứ hai, δ, phụ thuộc vào thông điệp x và khóa riêng tư a. Việc xác thực chữ ký được thực hiện bằng cách kiểm tra xem một đồng dư cụ thể chứa modulo p; đồng dư này tất nhiên không chứa khóa riêng tư.

Chúng ta chỉ ra rằng, khi chữ ký được xây dựng đúng cách, quá trình xác thực sẽ diễn ra thành công. Điều này được suy diễn một cách dễ dàng từ đồng dư sau đây:

$$\begin{aligned} \beta^\gamma \gamma^\delta &\equiv \alpha^{a\gamma}\alpha^{k\delta} \pmod{p} \\ &\equiv \alpha^x \pmod{p}, \end{aligned}$$

với

$$a\gamma + k\delta \equiv x \pmod{p-1}.$$

Thật ra, bắt đầu với phương trình xác thực sẽ dễ nắm bắt hơn, và sau đó suy ra hàm ký tương ứng. Giả sử chúng ta có đồng dư:

$$\alpha^x \equiv \beta^\gamma \gamma^\delta \pmod{p}. \tag{8.1}$$

Sau đó chúng ta có các phép thế

$$\gamma \equiv \alpha^k \pmod{p}$$

và

$$\beta \equiv \alpha^a \pmod{p},$$

nhưng chúng ta không thay thế ɣ trong lũy thừa của (8.1). Ta có:

$$\alpha^x \equiv \alpha^{a\gamma + k\delta} \pmod{p}.$$

Bây giờ, α là một phần tử nguyên thủy modulo p; do đó đồng dư này là đúng khi và chỉ khi các lũy thừa là đồng dư p - 1, cụ thể là, khi và chỉ khi:

$$x \equiv a\gamma + k\delta \pmod{p-1}.$$

Cho x, a, γ, và k, đồng dư này có thể được giải để tìm ra δ, tạo ra công thức được sử dụng trong hàm ký của Hệ mật 8.2.

Alice tính toán một chữ ký sử dụng cả khóa riêng tư, a, và một số ngẫu nhiên, k, (được sử dụng để ký một thông điệp, x). Việc xác thực có thể được thực hiện mà chỉ sử dụng thông tin công khai.

Hãy cùng làm một ví dụ nhỏ để minh họa việc tính toán:

Ví dụ 8.1 Giả sử chúng ta có p = 467, $\alpha = 2$, $a = 127$; ta có

$$\begin{aligned} \beta &= \alpha^a \bmod p \\ &= 2^{127} \bmod 467 \\ &= 132. \end{aligned}$$

Giả sử Alice muốn ký thông điệp $x = 100$ và cô ấy chọn số ngẫu nhiên là $k = 213$ (chú ý rằng gcd(213, 466) = 1 và $213^{-1} \bmod 466 = 431$). Ta có

$$\gamma = 2^{213} \bmod 467 = 29$$

và

$$\delta = (100 - 127 \times 29)\ 431 \bmod 466 = 51$$

Người khác có thể xác thực chữ ký (29,51) bằng cách kiểm tra rằng

$$132^{29} 29^{51} \equiv 189 \pmod{467}$$

và

$$2^{100} \equiv 189 \pmod{467}$$

Do đó, chữ ký là hợp lệ.

8.3.1 Tính bảo mật của Cơ chế chữ ký ElGamal

Cùng xem tính bảo mật của Cơ chế chữ ký ElGamal. Giả sử Oscar muốn giả mạo một chữ ký cho thông điệp x, trong khi không biết a. Nếu Oscar chọn một giá trị γ và sau đó cố gắng tìm δ tương ứng, anh ta phải tính logarit rời rạc $\log_{\gamma}\alpha^{x}\beta^{-\gamma}$. Mặt khác, nếu trước tiên anh ta chọn δ và sau đó cố gắng tìm γ, nghĩa là anh ta đang cố "giải" phương trình

$$\beta^{\gamma}\gamma^{\delta} \equiv \alpha^{x} \pmod{p}$$

để tìm ra γ "chưa biết". Đây là một bài toán chưa có lời giải khả thi; tuy nhiên, nó dường như không có liên quan đến bất cứ một bài toán nổi tiếng nào ví dụ như bài toán **Logarit Rời rạc.** Vẫn còn lại khả năng tồn tại một cách nào đó để có thể tính đồng thời cả γ và δ sao cho (γ,δ) là một chữ ký. Hiện chưa có ai tìm ra cách để làm điều này, nhưng ngược lại, chưa có ai chứng minh được là nó là bất khả thi.

Nếu Oscar chọn γ và δ và sau đó cố gắng giải để tìm ra x, anh ta lại một lần nữa phải đối mặt với một bài toán Logarit Rời rạc, cụ thể là phép tính $\log_{\alpha}\beta^{\gamma}\gamma^{\delta}$. Do đó, Oscar không thể ký một thông điệp x cho trước sử dụng hướng đi này.

Tuy nhiên, có một phương pháp Oscar có thể sử dụng để ký một tin nhắn ngẫu nhiên bằng cách chọn γ, δ, và x đồng thời. Do đó một giả mạo tồn tại (existential forgery) có thể thực hiện dưới một cuộc tấn công chỉ sử dụng khóa (giả sử rằng một hàm băm không được sử dụng). Chúng tôi sẽ mô tả cách làm việc này bây giờ.

Giả sử i và j là các số nguyên sao cho $0 \le i \le p - 2$, $0 \le j \le p - 2$, và giả sử chúng ta biểu diễn γ theo dạng $\gamma = \alpha^{i}\beta^{j} \bmod p$. Điều kiện xác thực có dạng

$$\alpha^x \equiv \beta^\gamma (\alpha^i \beta^j)^\delta \pmod{p}.$$

Tương đương với

$$\alpha^{x-i\delta} \equiv \beta^{\gamma+j\delta} \pmod{p}.$$

Biểu thức đồng dư thứ hai này sẽ được thỏa mãn nếu

$$x - i\delta \equiv 0 \pmod{p-1}$$

và

$$\gamma + j\delta \equiv 0 \pmod{p-1}.$$

Cho i và j, chúng ta có thể dễ dàng giải hai đồng dư modulo p - 1 để tìm ra δ và x, biết rằng gcd(j, p - 1) = 1. Ta có:

$$\begin{aligned} \gamma &= \alpha^i \beta^j \bmod p, \\ \delta &= -\gamma j^{-1} \bmod (p-1), \quad \text{and} \\ x &= -\gamma i j^{-1} \bmod (p-1). \end{aligned}$$

Thông qua cách chúng ta đã xây dựng (γ,δ), đây rõ ràng là một chữ ký hợp lệ cho thông điệp x.

Chúng ta hãy minh họa nó bằng một ví dụ.

Ví dụ 8.2 Như trong ví dụ trước, giả sử p = 467, α = 2, và β = 132. Giả sử Oscar chọn i = 99 và j = 179; ta có j^{-1} mod (p - 1) = 151. Anh ta sẽ tính như sau:

$$\begin{aligned} \gamma &= 2^{99}132^{179} \bmod 467 &&= 117 \\ \delta &= -117 \times 151 \bmod 466 &&= 41 \\ x &= 99 \times 41 \bmod 466 &&= 331. \end{aligned}$$

Ta được (117,41) là một chữ ký hợp lệ cho thông điệp 331, có thể được xác thực bằng cách kiểm tra:

$$132^{117}117^{41} \equiv 303 \pmod{467}$$

và

$$2^{331} \equiv 303 \pmod{467}.$$

Trong dạng giả mạo thứ hai, Oscar bắt đầu với một thông điệp được ký trước đó bởi Alice. Đây là một cuộc tấn công giả mạo tồn tại (existential forgery) với thông điệp được biết trước. Giả sử (γ,δ) là một chữ ký hợp lệ cho thông điệp x. Oscar có khả năng ký nhiều thông điệp khác. Giả sử h, i, và j là các số nguyên, $0 \leq h,i,j \leq p - 2$, và $\gcd(h\gamma - j\delta, p - 1) = 1$. Ta cần tính:

$$\begin{aligned} \lambda &= \gamma^h \alpha^i \beta^j \bmod p \\ \mu &= \delta\lambda(h\gamma - j\delta)^{-1} \bmod (p-1), \quad \text{and} \\ x' &= \lambda(hx + i\delta)(h\gamma - j\delta)^{-1} \bmod (p-1). \end{aligned}$$

Sau đó, sẽ khá mất thời gian nhưng nó là dễ tiếp cận để kiểm tra điều kiện xác thực

$$\beta^\lambda \lambda^\mu \equiv \alpha^{x'} \pmod{p}$$

là đúng. Do đó, (λ, μ) là chữ ký hợp lệ cho x'.

Cả hai phương pháp này là giả mạo tồn tại (existential forgery), nhưng chúng khó có khả năng có thể được sửa đổi để tạo ra giả mạo lựa chọn (selective forgery). Do đó, chúng dường như không đem lại mối đe dọa nào cho tính báo mật của Cơ chế chữ ký ElGamal, với điều kiện một hàm băm bảo mật được sử dụng như đã mô tả ở **phần 8.2.1**.

Chúng tôi cũng đề cập đến một số cách Cơ chế chữ ký ElGamal có thể bị phá nếu được sử dụng không cẩn thận (đó là các trường hợp rộng hơn của lỗi giao thức (protocol failures), được giới thiệu trong các bài tập của **chương 6**). Đầu tiên, giá trị ngẫu nhiên k được sử dụng trong việc tính toán chữ ký không nên bị để lộ. Vì, nếu k được biết và $\gcd(\gamma, p - 1) = 1$, thì nó sẽ là rất đơn giản để tính toán

$$a = (x - k\delta)\gamma^{-1} \bmod (p-1).$$

Một khi a được biết thì hệ thống đã hoàn toàn bị phá hỏng và Oscar có thể tự do giả mạo chữ ký.

Một cách sử dụng hệ thống sai khác là sử dụng cùng một giá trị k để ký hai thông điệp khác nhau. Việc này sẽ dẫn đến một giá trị γ trùng lặp, và nó cũng cho phép Oscar có thể dễ dàng tính a và do đó phá hỏng hệ thống. Việc này có thể được làm như sau. Giả sử (γ, δ_1) là một chữ ký của x_1 và (γ, δ_2) là một chữ ký của x_2. Ta có

$$\beta^{\gamma}\gamma^{\delta_1} \equiv \alpha^{x_1} \pmod{p}$$

và

$$\beta^{\gamma}\gamma^{\delta_2} \equiv \alpha^{x_2} \pmod{p}.$$

Nên

$$\alpha^{x_1-x_2} \equiv \gamma^{\delta_1-\delta_2} \pmod{p}.$$

Viết $\gamma = \alpha^k$, ta có phương trình sau với k chưa biết:

$$\alpha^{x_1-x_2} \equiv \alpha^{k(\delta_1-\delta_2)} \pmod{p},$$

tương đương với

$$x_1 - x_2 \equiv k(\delta_1 - \delta_2) \pmod{p-1}. \tag{8.2}$$

Đặt $d = \gcd(\delta_1 - \delta_2, p - 1)$. Nếu $d = 1$, chúng ta có thể ngay lập tức giải (8.2), ta được

$$k = (x_1 - x_2)(\delta_1 - \delta_2)^{-1} \pmod{p-1}.$$

Tuy nhiên, ngay cả khi $d > 1$, chúng ta vẫn có khả năng xác định giá trị của k, với điều kiện là d không quá lớn. Vì $d \mid (p - 1)$ và $d \mid (\delta_1 - \delta_2)$, ta có $d \mid (x_1 - x_2)$. Định nghĩa

$$\begin{aligned} x' &= \frac{x_1 - x_2}{d} \\ \delta' &= \frac{\delta_1 - \delta_2}{d} \\ p' &= \frac{p-1}{d}. \end{aligned}$$

Đồng dư (8.2) trở thành:

$$x' \equiv k\delta' \pmod{p'}.$$

Vì gcd(δ', p') = 1, chúng ta có thể tính

$$\epsilon = (\delta')^{-1} \bmod p'.$$

Giá trị của k là

$$k = x'\epsilon \bmod p'.$$

k có *d* giá trị khả thi:

$$k = x'\epsilon + ip' \bmod (p-1)$$

với một vài giá trị *i*, $0 \le i \le d$ - 1. Trong số *d* giá trị khả thi đó, giá trị (duy nhất) đúng có thể được xác định bằng cách thử điều kiện

$$\gamma \equiv \alpha^k \pmod{p}.$$

8.4

Trong nhiều trường hợp, một message có thể chỉ được mã hóa và giải mã một lần, do đó, chỉ cần sử dụng bất kỳ hệ thống mật mã nào được biết là an toàn tại thời điểm tin nhắn được mã hóa là đủ. Mặt khác, một signed message có thể hoạt động như một tài liệu pháp lý như hợp đồng hoặc di chúc; vì vậy rất có thể cần phải xác minh chữ ký nhiều năm sau khi thông điệp được ký. Do đó, điều quan trọng là phải thực hiện nhiều biện pháp phòng ngừa liên quan đến tính bảo mật của sơ đồ chữ ký thay vì hệ thống mật mã. Vì Lược đồ chữ ký ElGamal không bảo mật hơn Discrete Logarithm problem nên điều này đòi hỏi phải sử dụng mô đun p lớn. Hầu hết mọi người bây giờ cho rằng độ dài của p phải ít nhất là 2048 bit để bảo mật ngày nay và thậm chí lớn hơn để bảo mật trong tương lai gần

Mô-đun 2048 bit dẫn đến chữ ký ElGamal có 4096 bit. Đối với các ứng dụng tiềm năng, trong đó có nhiều ứng dụng liên quan đến việc sử dụng thẻ thông minh và cần chữ ký ngắn hơn. Năm 1989, Schnorr đề xuất một sơ đồ chữ ký có thể được xem như một biến thể của Sơ đồ chữ ký ElGamal trong đó kích thước chữ ký được giảm đi đáng kể. Thuật toán chữ ký số (hoặc DSA) là một sửa đổi khác của Lược đồ chữ ký ElGamal, kết hợp một số ý tưởng được sử dụng trong Lược đồ chữ ký Schnorr. DSA được công bố trong Đăng ký Liên bang vào ngày 19 tháng 5 năm 1994 và được thông qua như một tiêu chuẩn vào ngày 1 tháng 12 năm 1994 (tuy nhiên, nó được đề xuất lần đầu tiên vào tháng 8 năm 1991). Chúng tôi mô tả Lược đồ chữ ký Schnorr, DSA và sửa đổi DSA thành các đường cong elip (được gọi là Thuật toán chữ ký số đường cong Elliptic, hoặc ECDSA) trong các phần phụ tiếp theo.

8.4.1. Sơ đồ chữ ký Schnorr

Giả sử p và q là các số nguyên tố sao cho $p - 1 \equiv 0 \ (mod\ q)$. Thông thường $p \sim 2^{2048}$ và $q \sim 2^{224}$. Lược đồ chữ ký Schnorr sửa đổi Lược đồ chữ ký El-Gamal một cách khéo léo sao cho bản tóm tắt (message digest) có độ dài $log_2 q$ - bit được ký bằng chữ ký $2log_2 q$ - bit, nhưng việc tính toán được thực hiện trong Z_p. Cách thực hiện điều này là làm việc trong một nhóm con Z_p^* có kích thước q. Độ an toàn giả định của lược đồ dựa trên niềm tin rằng việc tìm các logarit rời rạc trong nhóm con xác định này của Z_p^* là an toàn.

Gọi α là căn bậc q của p. Để xác định α, gọi α_0 là phần tử nguyên thủy của Z_p, khi đó $\alpha = \alpha_0^{(p-1)/q} \ mod\ p$. Khóa trong Lược đồ chữ ký Schnorr tương tự như khóa trong Lược đồ chữ ký ElGamal ở các khía cạnh khác. Tuy nhiên, Sơ đồ Chữ ký Schnorr tích hợp một hàm băm trực tiếp vào thuật toán ký. Chúng ta sẽ giả sử rằng $h: \{0, 1\}^* \longrightarrow Z_q$ là hàm băm an toàn. Một mô tả đầy đủ về Lược đồ Chữ ký Schnorr được đưa ra dưới dạng Hệ thống mật mã 8.3.

Cryptosystem 8.3: Schnorr Signature Scheme

Cho p là một số nguyên tố sao cho bài toán logarit rời rạc trong Z_p^* là một bài toán khó giải, và gọi số nguyên tố q là ước của p-1. Gọi $\alpha \in Z_p^*$ là căn bậc q của p. Gọi $P = \{0, 1\}^*$ và $A = Z_q \ x \ Z_q$ và định nghĩa $K = \{(p, q, \alpha, a, \beta): \beta \equiv \alpha^a (mod\ p)\}$, trong đó $0 \leq a \leq q - 1$. Các giá trị p, q, α, và β là khóa công khai, a là khóa bí mật. Cuối cùng, đặt $h: \{0, 1\}^* \longrightarrow Z_q$ là hàm băm an toàn.

Với $K = \{(p, q, \alpha, a, \beta)$ và số bí mật ngẫu nhiên k, $1 \le k \le q - 1$, định nghĩa:

$$sig_K(x, k) = (\gamma, \delta)$$

Trong đó $\gamma = h(x \,||\, a^k \bmod p)$ và $\delta = k + a\gamma \bmod q$.

Với $x \in \{0,1\}^*$ và $\gamma, \delta \in Z_q$, việc xác minh được thực hiện bằng cách thực hiện các tính toán sau:

$$ver_K(x, (\gamma, \delta)) = true \Leftrightarrow h(x \,||\, \alpha^{\delta}\beta^{-\gamma} \bmod p) = \gamma.$$

Quan sát thấy mỗi thành phần trong số hai thành phần của chữ ký Schnorr là một phần tử của Z_q. Dễ dàng kiểm tra rằng $\alpha^{\delta}\beta^{-\gamma} \equiv \alpha^k \bmod p$ và do đó chữ ký Schnorr sẽ được xác minh.

Example 8.3

8.4.2 Thuật toán chữ ký số

Chúng ta sẽ trình bày các thay đổi được thực hiện trong hàm xác minh của Lược đồ chữ ký ElGamal trong đặc tả của DSA. DSA sử dụng một nhóm con có thứ tự q thuộc Z_p^*, cũng giống như Lược đồ chữ ký Schnorr. Trong DSA, khuyến nghị q là một số nguyên tố 224-bit và p là một số nguyên tố 2048-bit. Khóa trong DSA có cùng dạng như trong Lược đồ chữ ký Schnorr. Giả định rằng message sẽ được băm sử dụng SHA3-224 trước khi ký. Kết quả là một bản tóm tắt 224-bit được ký bằng một chữ ký 448-bit, và các tính toán được thực hiện trong Z_p và Z_q.

Trong Sơ đồ chữ ký ElGamal, giả sử chúng ta thay đổi "-" thành "+" trong định nghĩa của δ, do đó: $\delta = (x + a\gamma)k^{-1} \bmod (p - 1)$.

Dễ dàng nhận thấy rằng điều này thay đổi điều kiện xác minh như sau:

$$\alpha^x\beta^{\gamma} \equiv \gamma^{\delta} \pmod p \quad (8.3)$$

Bây giờ, α có bậc q, và β và γ là lũy thừa của α, nên chúng cũng có bậc q. Điều này có nghĩa là tất cả số mũ trong (8.3) có thể được rút gọn theo modulo q mà không ảnh hưởng đến tính đúng đắn của đồng dư. Vì x sẽ được thay thế bằng bản tóm tắt 224 bit trong DSA nên chúng ta sẽ giả sử rằng $x \in Zq$. Hơn nữa, chúng ta sẽ thay đổi định nghĩa của δ, sao cho $\delta \in Zq$, như sau: $\delta = (x + a\gamma)k^{-1} \bmod q$.

Coi $\gamma = \alpha^k \bmod p$. Giả sử chúng ta tạm thời xác định $\gamma' = \gamma \bmod q = (\alpha^k \bmod p) \bmod q$. Để ý $\delta = (x + a\gamma)k^{-1} \bmod q$ nên δ không đổi. Chúng ta có thể viết phương trình xác minh như sau: $\alpha^x\beta^{\gamma'} \equiv \gamma^{\delta} \pmod p$. (8.4)

Cryptosystem 8.4: Digital Signature Algorithm

Cho p là một số nguyên tố 2048-bit sao cho bài toán logarit rời rạc trong Z_p^* là một bài toán khó giải, và gọi số nguyên tố 224-bit q là ước của p-1. Gọi $\alpha \in Z_p^*$ là căn bậc q của p. Gọi $P = \{0, 1\}^*$ và $A = Z_q \, x \, Z_q$ và định nghĩa $K = \{(p, q, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod p\}$, trong đó $0 \le a \le q - 1$. Các giá trị p, q, α, và β là khóa công khai, a là khóa bí mật.

Với $K = \{(p, q, \alpha, a, \beta)$ và số bí mật ngẫu nhiên k, $1 \le k \le q - 1$, định nghĩa:

$$sig_K(x, k) = (\gamma, \delta)$$

Trong đó $\gamma = (a^k \bmod p) \bmod q$ và $\delta = (SHA3\_224(x) + a\gamma)k^{-1} \bmod q$. (Nếu $\gamma = 0$ hoặc $\delta = 0$, giá trị ngẫu nhiên k sẽ được chọn lại.

Với $x \in \{0,1\}^*$ và $\gamma, \delta \in Z_q^*$, việc xác minh được thực hiện bằng cách thực hiện các tính toán sau:

$$e_1 = SHA3\_224(x)\,\delta^{-1} \bmod q,\ e_2 = \gamma\delta^{-1} \bmod q$$

$$ver_K(x, (\gamma, \delta)) = true \Leftrightarrow (\alpha^{e1}\beta^{e2} \bmod p) \bmod q = \gamma$$

Lưu ý rằng chúng ta không thể thay thế sự xuất hiện còn lại của γ bằng γ'. Bây giờ chúng ta tiến hành viết lại (8.4), bằng cách tăng cả hai vế lên lũy thừa δ^{-1} mod q (điều này đòi hỏi $\delta^{-1} \neq 0$). Chúng tôi có được những điều sau đây: $\alpha^{x\delta^{-1}}\beta^{\gamma'\delta^{-1}} \equiv \gamma \pmod p$ (8.5)

Bây giờ chúng ta có thể rút gọn cả hai vế của (8.5) cho modulo q, thu được kết quả sau:

$(\alpha^{x\delta^{-1}}\beta^{\gamma'\delta^{-1}} \bmod p) \bmod q \equiv \gamma'$ (8.6).

Mô tả đầy đủ về DSA được đưa ra dưới dạng Cryptosystem 8.4, trong đó chúng tôi đổi tên γ' thành γ và thay thế x bằng bản tóm tắt SHA3-224 (x). Lưu ý rằng nếu Alice tính giá trị $\delta \equiv 0 \pmod q$ trong thuật toán ký DSA, cô ấy nên tạo chữ ký mới với k ngẫu nhiên mới. Chúng ta nên chỉ ra rằng điều này khó có thể gây ra vấn đề trong thực tế: xác suất $\delta \equiv 0 \pmod q$ có thể ở mức 2^{-224}; vì vậy, xét về mọi ý định và mục đích, điều đó sẽ không bao giờ xảy ra.

Example 8.4

Khi DSA được đề xuất vào năm 1991, đã có một số lời chỉ trích được đưa ra. Một khiếu nại là quy trình lựa chọn của NIST không được công khai. Tiêu chuẩn này được Cơ quan An ninh Quốc gia (NSA) phát triển mà không có sự tham gia của ngành công nghiệp Hoa Kỳ. Bất kể giá trị của kế hoạch đạt được là gì, nhiều người vẫn phẫn nộ với cách tiếp cận "đóng cửa".

Trong số những lời chỉ trích kỹ thuật được đưa ra, nghiêm trọng nhất là kích thước của mô đun p ban đầu được cố định ở mức 512 bit. Nhiều người cho rằng không nên cố định kích thước mô đun mà có thể sử dụng kích thước mô đun lớn hơn nếu muốn. Để đáp lại những nhận xét này, NIST đã thay đổi mô tả tiêu chuẩn để cho phép có nhiều kích cỡ mô đun khác nhau.

8.4.3. Đường cong Elliptic DSA

Năm 2000, Thuật toán chữ ký số đường cong Elliptic (ECDSA) đã được phê duyệt là FIPS 186-2. Sơ đồ chữ ký này có thể được xem như một sự sửa đổi của DSA đối với việc thiết lập các đường cong elliptic. Chúng ta có hai điểm A và B trên đường cong elliptic xác định trên Z_p với một số nguyên tố p. Logarit rời rạc $m = \log_A B$ là khóa bí mật. (Điều này tương tự với quan hệ $\beta \equiv \alpha^a \pmod p$ trong DSA, trong đó a là khóa bí mật.) Cấp của A là số nguyên tố lớn q. Việc tính toán chữ ký trước tiên bao gồm việc chọn một giá trị ngẫu nhiên k và tính toán kA (điều này tương tự như việc tính toán α^k trong DSA).

Đây là điểm khác biệt chính giữa DSA và ECDSA. Trong DSA, giá trị $\alpha^k \pmod p$ được giảm modulo q để tạo ra giá trị γ là thành phần đầu tiên của chữ ký (γ, δ). Trong ECDSA, giá trị tương tự là r, là tọa độ x của điểm kA trên đường cong elliptic, rút gọn modulo q. Giá trị *r* này là thành phần đầu tiên của chữ ký *(r, s)*.

Cuối cùng, trong ECDSA, giá trị *s* được tính từ *r, m, k* và message *x* theo cách giống hệt như δ được tính từ γ, *a, k* và message *x* trong DSA. Bây giờ chúng tôi trình bày mô tả đầy đủ về ECDSA dưới dạng Hệ thống mật mã 8.5.

Cryptosystem 8.5: Elliptic Curve Digital Signature Algorithm

Cho p là một số nguyên tố lớn và Ɛ là đường cong Elliptic định nghĩa trong Z_p. Gọi A là một điểm thuộc Ɛ có bậc q nguyên tố, sao cho bài toán Logarit rời rạc trong <A> không thể giải được. Gọi $P = \{0, 1\}^*$ và $A = Z_q \, x \, Z_q$ và định nghĩa

$K = \{(p, q, Ɛ, A, m, B) : B = mA$, trong đó $0 \leq m \leq q - 1$.

Các giá trị p, q, Ɛ, A, B là khóa công khai, m là khóa bí mật.

Với $K = \{(p, q, Ɛ, A, m, B)$ và số bí mật ngẫu nhiên $k, 1 \leq k \leq q - 1$, định nghĩa:

$$sig_K(x, k) = (r, s)$$

Trong đó $kA = (u, v), r = u \bmod q$ và $s = k^{-1}(SHA3\_224(x) + mr) \bmod q$. (Nếu $r = 0$ hoặc $s = 0$, giá trị ngẫu nhiên k sẽ được chọn lại.

Với $x \in \{0, 1\}^*$ và $r, s \in Z_q^*$, việc xác minh được thực hiện bằng cách thực hiện các tính toán sau:

$$w = s^{-1} \bmod q$$
$$i = w \times SHA3\_224(x) \bmod q$$
$$j = wr \bmod q$$
$$(u, v) = iA + jB$$
$$ver_K(x, (r, s)) = true \Leftrightarrow u \bmod q = r$$

Chú thích:
oracle model = mô hình máy băm
message digest = bản tóm tắt thông điệp
trapdoor one-way permutations = phép biến đổi một chiều với hàm trapdoor
signature scheme = hệ thống chữ ký
padding scheme = mô hình đệm

8.5. Full Domain Hash

Trong phần 6.9.2, chúng ta đã chỉ ra cách xây dựng hệ thống mã hóa bảo mật khóa công khai từ các phép biến đổi một chiều với hàm trapdoor (trong mô hình máy băm ngẫu nhiên). Các ứng dụng thực tế của các hệ thống này dựa trên Hệ thống mật mã hóa RSA và thay thế máy băm ngẫu nhiên bằng một hàm băm như SHA3-224. Trong phần này, chúng ta sử dụng một phép biến đổi một chiều với hàm trapdoor để xây dựng một hệ thống chữ ký bảo mật trong mô hình máy băm ngẫu nhiên. Hệ thống mà chúng ta trình bày được gọi là Full Domain Hash. Tên của hệ thống này xuất phát từ yêu cầu rằng phạm vi của máy băm ngẫu nhiên phải giống với miền của phép biến đổi một chiều với trapdoor được sử dụng trong hệ thống. Hệ thống này được trình bày như là Cryptosystem 8.6.

Cryptosystem 8.6: *Full Domain Hash*
Gọi k là số nguyên dương, F là tập các phép biến đổi một chiều trapdoor thỏa mãn $f\colon \{0,1\}^k \to \{0,1\}^k$ với mọi $f \in F$ và $G\colon \{0,1\}^* \to \{0,1\}^k$ là hàm ngẫu nhiên. $P = \{0,1\}^*$ và $A = \{0,1\}^k$ và định nghĩa

$$K = \{(f, f^{-1}, G)\colon f \in F\}$$

Cho khóa $K = (f, f^{-1}, G)$, f^{-1} là private key, (f, G) là public key.

Cho $K = (f, f^{-1}, G)$ và $x \in \{0,1\}^*$ và định nghĩa

$$sig_K(x) = f^{-1}(G(x))$$

Chữ ký $y = (y_1, y_2, \dots, y_k) \in \{0,1\}^k$ trên thông điệp x sẽ được xác nhận như sau:

$$ver_K(x, y) = true \Leftrightarrow f(y) = G(x)$$

Full Domain Hash sử dụng phương pháp băm-để-ký (hash-then-sign method) quen thuộc. G(x) là bản tóm tắt thông điệp (message digest) được tạo ra bởi

máy băm ngẫu nhiên G. Hàm f^{-1} được sử dụng để ký tóm tắt thông điệp, và f được sử dụng để xác minh nó.

Hãy tưởng tượng một cài đặt dựa trên RSA cho hệ thống này. Hàm f^{-1} sẽ là hàm ký RSA (tức là, giải mã), và f sẽ là hàm xác minh RSA (tức là, mã hóa). Ví dụ để đảm bảo tính bảo mật, chúng ta cần chọn k = 2048. Giả sử máy băm ngẫu nhiên G được thay thế bằng hàm băm SHA3-224. Hàm băm này tạo ra một bản tóm tắt thông điệp 224 bit, vì vậy phạm vi của hàm băm, tức $\{0,1\}^{224}$, là một tập con rất nhỏ của $\{0,1\}^{k} = \{0,1\}^{2048}$. Trong thực tế, cần phải xác định một mô hình đệm (padding scheme) cụ thể để mở rộng một thông điệp 224 bit thành 2048 bit trước khi áp dụng hàm f^{-1}. Điều này thường được thực hiện bằng cách sử dụng một mô hình đệm cố định (xác định trước).

Chúng ta tiếp tục đến phần chứng minh về tính bảo mật, trong đó chúng ta giả định rằng F là một tập hợp các phép biến đổi một chiều với trapdoor và G là một máy băm ngẫu nhiên "miền đầy đủ". (Lưu ý rằng các chứng minh về tính bảo mật chúng ta sẽ trình bày không áp dụng khi máy băm ngẫu nhiên được thay thế bằng một hàm băm được xác định hoàn toàn như SHA3-224.) Có thể chứng minh rằng Full Domain Hash an toàn khi chống lại việc làm giả tồn tại (existential forgery) bằng việc tấn công những message được chọn; tuy nhiên, chúng ta sẽ chỉ chứng minh kết quả dễ dàng hơn là Full Domain Hash an toàn khi chống lại việc làm giả tồn tại (existential forgery) bằng cuộc tấn công chỉ vào khóa (key-only attack).

Như thường lệ, chứng minh tính bảo mật là một loại giảm thiểu (type of reduction). Chúng ta giả định rằng có một kẻ thù (adversary) (ví dụ, một thuật toán ngẫu nhiên, được ký hiệu bởi FDH-FORGE) có khả năng làm giả chữ ký (với một xác suất cụ thể) khi nó được cung cấp khóa công khai và truy cập vào máy băm ngẫu nhiên (nhớ rằng nó có thể truy vấn máy băm ngẫu nhiên để lấy giá trị G(x), nhưng không có thuật toán cụ thể được chỉ định để đánh giá hàm G). FDH-FORGE thực hiện một số lần truy vấn máy băm, ví dụ như q_h. Cuối cùng, FDH-FORGE xuất ra một giả mạo hợp lệ với một xác suất cụ thể, được ký hiệu là e.

Algorithm 8.1: FDH-INVERT(z_0, q_h)

```
external f
procedure SIMG(x)
        if j > qh
                then return ("failure")
                else if j = j0
                then z ← z0
                else let z ∈ {0, 1} k be chosen at random
         j ← j + 1
        return (z)

main
        choose j0 ∈ {1, . . . , qh} at random
        j ← 1
        insert the code for FDH-FORGE(f) here
        if FDH-FORGE(f) = (x, y)
        then {
                if f(y) = z0
                then return (y)
                else return ("failure")}
```

Chúng ta xây dựng một thuật toán, FDH-INVERT, cố gắng đảo ngược các phần tử được chọn ngẫu nhiên $z_0 \in \{0,1\}^k$. Tức là, cho trước $z_0 \in \{0,1\}^k$, chúng ta hy vọng
là FDH-INVERT(z_0) =$f^{-1}(z_0)$. Bây giờ chúng ta trình bày FDH-INVERT như là Thuật toán 8.1.

Thuật toán 8.1 khá đơn giản. Nó chủ yếu bao gồm việc chạy kẻ thù (running the adversary), FDH-FORGE. Các truy vấn băm được thực hiện bởi FDH-FORGE được xử lý bởi hàm SIMG, một mô phỏng của máy băm ngẫu nhiên. Chúng ta đã giả định rằng FDH-FORGE sẽ thực hiện q_h truy vấn băm, ví dụ như $x_1, \ldots, x_{qh}$. Để cho đơn giản, chúng ta giả định rằng các x_1 đều là riêng biệt. (Nếu chúng không phải là riêng biệt, thì chúng ta cần đảm bảo rằng SIMG(x_i) = SIMG(x_j) khi $x_i = x_j$. Điều này không khó thực hiện; chỉ đòi hỏi một số công việc ghi chép, như đã thực hiện trong Thuật toán 6.14.) Chúng ta ngẫu nhiên

chọn một truy vấn, ví dụ như truy vấn thứ j_0, và định nghĩa SIMG(x_{j_0}) = z_0 (z_0 là giá trị mà chúng ta đang đảo ngược). Đối với tất cả các truy vấn khác, giá trị SIMG(x_j) được chọn là một số ngẫu nhiên. Bởi vì z_0 cũng là ngẫu nhiên, nên dễ thấy rằng SIMG không thể phân biệt được so với một máy băm ngẫu nhiên thực sự. Do đó, kết quả là FDH-FORGE xuất ra một thông điệp và một chữ ký giả mạo hợp lệ, được ký hiệu là (x, y), với xác suất e. Sau đó, chúng ta kiểm tra xem f(y) = z_0; nếu có, thì y = $f^{-1}(z_0)$ và chúng ta đã thành công trong việc đảo ngược z_0.

Nhiệm vụ chính của chúng ta là phân tích xác suất thành công của thuật toán FDH-INVERT dưới dạng một hàm của xác suất thành công e của FDH-FORGE. Chúng ta sẽ giả định rằng e > 2^{-k}, bởi vì một lựa chọn ngẫu nhiên của y sẽ là một chữ ký hợp lệ cho một thông điệp x với xác suất 2^{-k}, và chúng ta chỉ quan tâm đến những adversaries có khả năng thành công cao hơn đoán ngẫu nhiên.

Giống như chúng ta đã làm ở trên, chúng ta ký hiệu các truy vấn băm được thực hiện bởi FDH-FORGE bằng $x_1, \ldots, x_{q_h}$, trong đó x_j là truy vấn băm thứ j, $1 \leq j \leq q_h$.

Chúng ta bắt đầu bằng cách viết điều kiện cho xác suất thành công, e, dựa trên việc x có $\in$ {x1, . . . , xqh} hay không:

$$e = \mathbf{Pr}[\text{FDH-FORGE thành công} \wedge (x \in \{x_1, \ldots, x_{q_h}\})] + \mathbf{Pr}[\text{FDH-FORGE thành công} \wedge (x \notin \{x_1, \ldots, x_{q_h}\})]. \quad (8.7)$$

Không khó để thấy rằng:

$$\mathbf{Pr}[\text{FDH-FORGE thành công} \wedge (x \notin \{x_1, \ldots, x_{q_h}\})] = 2^{-k}.$$

Điều này xảy ra vì giá trị (chưa xác định) SIMG(x) có khả năng như nhau để nhận bất kỳ giá trị cụ thể nào trong $\{0, 1\}^k$, và do đó xác suất SIMG(x) = f(y) là

2^{-k}. (Đây là nơi chúng ta sử dụng giả thiết rằng hàm băm là một "miền đầy đủ".) Thay thế vào (8.7), chúng ta có:

$$\textbf{Pr}[\text{FDH-FORGE thành công} \wedge (x \in \{x_1, \ldots, x_{q_h}\})] \geq e - 2^{-k}. \quad (8.8)$$

Bây giờ chúng ta xem xét xác suất thành công của FDH-INVERT. Bất đẳng thức tiếp theo là rõ ràng:

$$\textbf{Pr}[\text{FDH-INVERT thành công}] \geq \textbf{Pr}[\text{FDH-FORGE thành công} \wedge (x = x_{j_0})]. \quad (8.9)$$

Quan sát cuối cùng của chúng ta là:

$$\textbf{Pr}[\text{FDH-FORGE thành công} \wedge (x = x_{j_0})] = \frac{1}{x_{q_h}} \times \textbf{Pr}[\text{FDH-FORGE thành công} \wedge (x \in \{x_1, \ldots, x_{q_h}\})]. \quad (8.10)$$

Lưu ý rằng phương trình (8.10) đúng vì có xác suất $\frac{1}{q_h}$ rằng x = x_{j_0}, trong trường hợp x $\in \{x_1, \ldots, x_{q_h}\}$. Bây giờ, nếu chúng ta kết hợp (8.8), (8.9), và (8.10), chúng ta sẽ có được giới hạn sau đây:

$$\textbf{Pr}[\text{FDH-INVERT thành công}] \geq \frac{e - 2^{-k}}{q_h}. \quad (8.11)$$

Vì vậy, chúng ta đã có được một giới hạn dưới cụ thể về xác suất thành công của FDH-INVERT. Chúng ta đã chứng minh kết quả sau đây.

THEOREM 8.1: Giả sử tồn tại một thuật toán FDH-FORGE có khả năng xuất ra một giả mạo tồn tại cho Full Domain Hash với xác suất e > 2^{-k} sau khi thực hiện q_h truy vấn tới máy băm ngẫu nhiên, sử dụng cuộc tấn công chỉ có khóa. Sau đó, tồn tại một thuật toán FDH-INVERT có khả năng tìm ra các phần tử ngẫu nhiên nghịch đảo $z_0 \in \{0, 1\}^k$ với xác suất ít nhất là $\frac{e - 2^{-k}}{q_h}$.

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].