Wykład 6

Temat: Inne algorytmy z kluczem publicznym [Diffiego-Hellmana (zasada działania, bezpieczeństwo), ElGamala, systemy oparte na zagadnieniu krzywych eliptycznych, system Rabina, system podpisów cyfrowych DSS (wprowadzenie do podpisów cyfrowych, system DSS, protokoły wiedzy zerowej)]. Systemy kryptograficzne z kluczem publicznym [wprowadzenie, projekt NESSIE, projekt CRYPTREC, nowoczesne systemy kryptograficzne klucza publicznego (RSAES-OAEP, RSA-PSS)].

6.i. Inne algorytmy z kluczem publicznym

6.1.1. System Diffiego-Hellmana

Spotkania na corocznych konferencjach Crypto i Eurocrypt poświęconych kryptografii zaowocowały koniecznością zaproponowania wielu systemów kryptograficznych. Przed RSA w roku 1976 Whitfield Diffie i Martin Hellman zaproponowali system tylko do wymiany klucza dla systemów symetrycznych, który jest wykorzystywany do dziś, często wymieniany w skrócie: D-H. System ten jest niemal równie bezpieczny jak RSA, ale wykorzystuje bardziej złożone mechanizmy.

6.1.1.1. Zasada działania

Algorytm opiera się na problemie wyznaczania logarytmów dyskretnych. Rozwiązanie tego problemu wymaga znalezienia wartości x z wyrażenia:

$$a^x \equiv b \ (mod\ p), \tag{6.1}$$

dla zadanych wartości a, b i p. Nie wszystkie logarytmy dyskretne mają rozwiązanie, ponieważ rozwiązaniem może być tylko liczba całkowita.

Problem bazuje na strukturach matematycznych zwanych grupami i opisuje się go następująco: Zakłada się że dana jest skończona cykliczna grupa multiplikatywna (tzn. między elementami grupy zachodzą relacje mnożenia) liczb całkowitych G rzędu n; o grupie G mówi się że jest cykliczna jeśli posiada pierwotny element g, czyli jeśli $ord(g)=|G|$ (operator ord określa rząd, przykładowo $ord(x)$ określa rząd x, czyli liczbę elementów w x).

Przyjmując inny element z $G - a$, trzeba znaleźć z wyrażenia:

$$g^x = a \tag{6.2}$$

wartość [illegible]. Wartość x nazywa się logarytmem liczby a o podstawie g i zapisuje się jako:

$$x = log_g(a) \tag{6.3}$$

Logarytmy dyskretne można obliczać w różnych grupach skończonych, stąd wynika ich wszechstronne zastosowanie. W algorytmie Diffiego-Hellmana zastosowano multiplikatywną grupę liczb całkowitych modulo liczba pierwsza (zapis: Z^*_p), później okazało się że można wykorzystać jakąkolwiek inną grupę multiplikatywną – najbardziej popularne są obecnie następujące grupy multiplikatywne w polu skończonym (czyli tzw. polu Galoisa): $GF(p)$ oraz $GF(2^n)$.

Operacje arytmetyczne wykonywane są w polu Galoisa określonego rzędu (czyli w polu o określonej liczbie elementów); zazwyczaj rząd określony jest liczbą pierwszą, co zapisuje się jako: $GF(p)$ (co jest równoznaczne z zapisem Z^*_p – przykładowo pole takie stosuje się w algorytmie D-H) albo potęgą liczby pierwszej: $GF(2^n)$ (inaczej Z^*_p). Dla każdej liczby pierwszej lub jej potęgi istnieje dokładnie jedno pole skończone Galoisa. W kryptografii stosuje się pola Galoisa $GF(p)$ o dużej liczbie p.

W systemach kryptograficznych z kluczem publicznym bazujących na logarytmach dyskretnych, grupa osób może współdzielić pewne parametry (tzw. parametry systemowe) wykorzystywane w danym systemie – powodem takiego postępowania jest tworzenie prostszych implementacji systemów, ale może to stanowić zachętę dla osób które chciałyby złamać klucz.

Wracając do rozważań na temat systemu Diffiego-Hellmana, należy wspomnieć, że nie można go bezpośrednio wykorzystać do szyfrowania danych. Algorytm D-H został zaprojektowany z myślą o wygenerowaniu klucza tajnego (sesyjnego) u obu stron uczestniczących w wymianie poufnych danych za pomocą uzgodnionego klucza publicznego.

Wiadomości z arytmetyki modularnej i kongruencji

Niech reszta po dzieleniu liczb $a = q^x$ i m na liczbę p jest ta sama. Wtedy kongruencje lub przystawanie liczb $a = q^x$ i m według modułu p (modulo p) oznaczamy wzorem

$$q^x \equiv m \pmod{p},$$

gdzie q - pierwiastek pierwotny modulo p i $p \nmid m$, oraz x - logarytm dyskretny albo indeks liczby m o podstawie q i modulo p. W tym przypadku zapisujemy $x = \mathrm{ind}(m, q, p)$ czyli $x = \mathrm{ind}_q m$, jeśli modulo p jest znany, przy czym $0 \le \mathrm{ind}_q m < p - 1$. Na przykład:

$\mathrm{ind}(1, 5, 23) = 0$	$\mathrm{ind}(3, 5, 23) = 16$	$\mathrm{ind}(5, 5, 23) = 1$
$\mathrm{ind}(2, 5, 23) = 2$	$\mathrm{ind}(4, 5, 23) = 4$	$\mathrm{ind}(6, 5, 23) = 18$

Algorytm Diffiego-Hellmana przebiega następująco:

Dwie strony, A i B, uzgadniają dwie możliwie jak największe wartości (parametry) publiczne: u i p, będące liczbami pierwszymi. Następnie każda strona wybiera inną, tajną wartość – przykładowo użytkownik A wybiera wartość a, natomiast użytkownik B - wartość b. Użytkownik A wysyła wartość:

$$x = u^a \bmod p \tag{6.4}$$

do użytkownika B, który z kolei wysyła użytkownikowi A wartość

$$y = u^b \bmod p. \tag{6.5}$$

Po otrzymaniu wartości przez obie strony, użytkownik A oblicza

$$k = y^a \bmod p, \tag{6.6}$$

natomiast użytkownik B wyznacza

$$k' = x^b \bmod p. \tag{6.7}$$

Można pokazać, że zarówno k jak i k' równe są $u^{ab} \bmod p$. Uważa się za niemożliwe wyznaczenie liczb a i b, czyli również wartości $u^{ab} \bmod p$ dla osoby nieupoważnionej ponieważ ktoś podsłuchujący kanał mógłby pozyskać tylko wartości x, y oraz u i p. O ile intruz nie będzie w stanie wyznaczyć wartości a i b w oparciu o logarytmy dyskretne, metoda pozostaje bezpieczna. Trudność w wyznaczaniu klucza tajnego zwiększa się oczywiście ze wzrostem wartości liczb u i p.

W wyniku przebiegu powyższego algorytmu powstaje klucz sesyjny równy wartości wyrażenia $u^{ab} \bmod p$, znanej tylko użytkownikom A i B.

Jeśli w wymianie informacji poufnej biorą udział więcej niż dwie osoby, metoda może być w prosty sposób rozwinięta. Podobnie przyjmuje się jawne wartości *u* i *p*, a klucz sesyjny, ***przykładowo dla trzech osób, A, B i C***, wyznacza się następująco:

- użytkownik A przyjmuje dużą liczbę a i wysyła do użytkownika B wartość:

$$x = u^a \bmod p; \tag{6.8}$$

- użytkownik B przyjmuje dużą liczbę b i wysyła do użytkownika C wartość:

$$y = u^b \bmod p; \tag{6.9}$$

- użytkownik C wybiera dużą liczbę c i wysyła do użytkownika A wartość:

$$z = u^c \bmod p; \tag{6.10}$$

- użytkownik A dysponując wartością z pochodzącą od użytkownika C, wysyła użytkownikowi B wartość:

$$z' = z^a \bmod p; \tag{6.11}$$

- użytkownik B dysponując wartością x pochodzącą od użytkownika A, wysyła użytkownikowi C wartość:

$$x' = x^b \bmod p; \tag{6.12}$$

- użytkownik C dysponując wartością y pochodzącą od użytkownika B, wysyła użytkownikowi A wartość:

$$y' = y^c \bmod p; \tag{6.13}$$

- użytkownik A po otrzymaniu wartości y' od użytkownika C oblicza klucz sesyjny:

$$k = y'^a \bmod p; \tag{6.14}$$

- podobnie użytkownik B korzystając wartości z' od użytkownika A oblicza klucz sesyjny

$$k = z'^y \bmod p \tag{6.15}$$

oraz użytkownik C wykorzystując wartość x' pochodzącą od użytkownika C wyznacza również klucz sesyjny

$$k = x'^c \bmod p. \tag{6.16}$$

Tajny klucz k równy jest wyrażeniu $u^{xyz} \bmod p$, które nie może być wyznaczone przez nikogo podsłuchującego publiczny kanał przesyłania danych.

Wygenerowany z wykorzystaniem metody tajny klucz sesyjny może zostać zastosowany w systemach klucza tajnego do operacji szyfrowania i odszyfrowania. Z systemu Diffiego-Hellmana korzysta się powszechnie – jest równie popularny jak system RSA, szczególnie w zastosowaniu w sieciach rozległych, np. w relacji klient-serwer.

6.1.1.2. Bezpieczeństwo algorytmu Diffiego-Hellmana

Jak już wcześniej wspomniano, bezpieczeństwo systemu Diffiego-Hellmana zależy od trudności wyznaczania logarytmów dyskretnych, generalnie jednak system ten można uważać za bezpieczny jeśli zastosowane są odpowiednio długie klucze, wygenerowane w oparciu o dobry generator liczb pseudolosowych.

Z drugiej strony - o tym że metoda bazująca na problemie logarytmów dyskretnych może okazać się niewystarczająco bezpieczna mówią badania przeprowadzone przez Briana LaMachia i Andrew Odlyzko (zamieszczone w książce p.t. "*Computation of Discrete Logarithms in Prime Fields, Designs, Codes and Cryptography*"). Wykazano bowiem, że przy zastosowaniu odpowiednich obliczeń w odniesieniu do pewnej unikalnej liczby pierwszej obliczenie logarytmu dyskretnego nie tylko jest możliwe ale i bardzo efektywne – wystarczy tu porównać że wkład potrzebny na wyznaczenie rozwiązania dla tego problemu jest porównywalny z wkładem potrzebnym na faktoryzację liczby tego samego rzędu wielkości.

System Diffiego-Hellmanaa może być narażony również na inne formy ataku np. na tzw. „Atak człowiek w środku" stosowany w odniesieniu do protokołów wymiany klucza. Atakujący w tym przypadku ustawia się na kanale komunikacyjnym pomiędzy nadawcą i odbiorcą, przeprowadzając odrębną procedurę wymiany klucza z każdą ze stron. W rezultacie intruz zna klucze należące do obu ze stron. Również nowa forma ataku - na czas wykonania, bazująca na powtarzaniu mierzenia dokładnego czasu wykonania operacji potęgowania może być użyta do złamania wielu implementacji Diffiego-Hellmana.

Jednym z zastosowań tego systemu jest wykorzystanie go w protokole tunelingu warstwy 3 modelu OSI – IPsec (IP Security) do wymiany kluczy między dwoma stacjami biorącymi udział w komunikacji, do podpisywania wymienianych kluczy i weryfikacji użytkowników biorących udział w dwustronnym połączeniu.

Szyfrowanie kluczem publicznym sprowadza się tu do zaszyfrowania losowo wygenerowanej liczby przez obie strony, kluczem publicznym strony przeciwnej. Uwierzytelnienie kończy się powodzeniem, gdy odszyfrowana kluczem prywatnym liczba, odesłana do nadawcy zgadza się z liczbą przed zaszyfrowaniem – w tym przypadku można stosować system RSA. Uwierzytelnienie może się tu również odbywać za pomocą podpisu cyfrowego; każda strona podpisuje blok danych i przesyła ją drugiej stronie – w tym przypadku używa się obecnie DSS i RSA.

Z kolei klucz sesyjny stosowany do tej wymiany ustala się z wykorzystaniem algorytmu Diffiego–Hellmana. Poprawne uwierzytelnienie implikuje proces uzgadniania klucza stosowanego w dalszej komunikacji – klucz ten może być identyczny z uprzednio wygenerowanym kluczem sesyjnym, co zapewni większą szybkość. Jeżeli najistotniejsza jest kwestia bezpieczeństwa, powinno się utworzyć nowy klucz.

6.1.2. System ElGamala

W roku 1985 Taher ElGamal zaprojektował wariant algorytmu Diffiego–Hellmana. Rozszerzył go aby umożliwić szyfrowanie i otrzymał algorytm, który mógł szyfrować i jeden, który mógł uwierzytelniać. Ponieważ opiera się on na algorytmie Diffiego–Hellmana, bezpieczeństwo informacji zależy od trudności obliczenia logarytmów dyskretnych. Algorytm ElGamala był pierwszym systemem który oprócz szyfrowania posłużył również do podpisu cyfrowego – stał on się przez to podstawą rozwoju innych systemów tego typu, takich jak DSS. W algorytmie ElGamala stosuje się pole *GF(p)*.

W procesie ***szyfrowania*** tym algorytmem wykorzystuje się za każdym razem inną, losowo wybraną liczbę – powoduje to że szyfr może mieć różną postać dla tej samej wiadomości jawnej. W systemie ElGamala klucz publiczny służy do szyfrowania a prywatny do odszyfrowania – role kluczy nie mogą ulec zmianie jak w przypadku RSA.

Do wygenerowania pary kluczy wybiera się najpierw liczbę pierwszą p oraz losowe liczby g i x dla których zachodzi: $g < p$ *i* $x < p$.

Warunki dla wyboru liczby p:

- p musi być wystarczająco duża (aby wyznaczenie dla niej logarytmu było niemożliwe);
- wartość $p - 1$ musi mieć co najmniej jeden duży czynnik pierwszy.

Liczba g to element pierwotny pola *GF(p)*, zwany inaczej generatorem. Obie liczby – g i p to parametry systemowe które mogą być wymieniane w danej grupie korespondencyjnej użytkowników.

Pozostaje jeszcze dobranie liczby y w wyrażeniu

$$y = g^x \bmod p \tag{6.17}$$

i otrzymuje się klucze: publiczny składający się z trójki liczb *(y, g, p)* oraz prywatny – liczba x.

Następnie dobiera się liczbę losową k względnie pierwszą do $p - 1$. Szyfrowana wiadomość M musi mieć długość mniejszą od liczby p.

Szyfrowanie przebiega w następujący sposób:

$$a = g^k \bmod p, \tag{6.18}$$

$$b = y^k M \bmod p. \tag{6.19}$$

Para (a, b) stanowi szyfrogram.

Odszyfrowanie przeprowadza się następująco:

$$M = \frac{b}{a^x} \bmod p \quad . \tag{6.20}$$

Powyższe równanie jest prawdziwe, ponieważ skoro

$$a^x = g^{kx} \ (mod\ p), \tag{6.21}$$

to:

$$\frac{b}{a^x} \equiv \frac{y^k \ M \ (\bmod p)}{a^x} \equiv \frac{g^{xk} \ M \ (\bmod p)}{g^{xk}} \equiv M \bmod p \quad . \tag{6.22}$$

Wadą algorytmu ElGamala jest długość uzyskiwanego szyfrogramu – dwukrotnie większa od długości wiadomości jawnej.

W procesie ***podpisywania i weryfikacji podpisu*** stosuje się pewną modyfikację algorytmu:

podobnie wybiera się liczbę k względnie pierwszą do $p - 1$, obliczane jest

$$a = g^k \ mod\ p. \tag{6.23}$$

Następnie wykorzystuje się rozszerzony algorytm Euklidesa do wyznaczenia wartości b z równania:

$$M = (xa + kb)\ mod\ (p - 1). \tag{6.24}$$

Weryfikacja podpisu sprowadza się do potwierdzenia czy zachodzi równość:

$$y^a a^b \ mod\ p = g^M \ mod\ p. \tag{6.25}$$

System ElGamala wykorzystuje się z powodzeniem w aplikacjach, choć jest wolniejszy w szyfrowaniu i weryfikacji niż RSA, a podpisy wygenerowane z jego wykorzystaniem są dłuższe niż w przypadku RSA.

6.1.3. Systemy oparte na zagadnieniu krzywych eliptycznych

Systemy kryptograficzne bazujące na operacjach matematycznych na krzywych eliptycznych zostały zaproponowane w latach '80 jako systemy kryptograficzne bazujące na dyskretnej eksponencie w ciele skończonym(polu Galoisa) o charakterystyce 2: $GF(2^n)$.

W tym miejscu należałoby wytłumaczyć pojęcie charakterystyki pola Galoisa, mianowicie:

Istnieje najmniejsza dodatnia wartość całkowita n która spełnia warunek: $\sum^{n} e = 0$ dla pewnego elementu *e* pola. Wartość tę nazywa się charakterystyką pola i jest to liczba pierwsza dla każdego skończonego pola.

Systemy kryptograficzne oparte o krzywe eliptyczne zyskały ostatnimi laty zainteresowanie ze względu na zapewnienie podobnego poziomu bezpieczeństwa jak inne, znane systemy kryptograficzne ale przy zastosowaniu mniejszych długości klucza. Również rozwój implementacji – jak ulepszona metoda generacji krzywych eliptycznych sprawia że system zyskuje na znaczeniu w obecnych zastosowaniach.

Krzywe eliptyczne to złożone struktury matematyczne; wykazano że mogą posłużyć do takich procesów jak np. test pierwszości.

Obecnie wykorzystywane systemy kryptograficzne oparte na krzywych eliptycznych mogą być w ten sposób zdefiniowane aby stanowiły bliskie odpowiedniki znanych systemów, bazujących zarówno na problemie logarytmów dyskretnych jak i tych bazujących na problemie faktoryzacji.

Systemy kryptograficzne oparte o krzywe eliptyczne, zdefiniowane jako odpowiednik logarytmów dyskretnych bazują na przedefiniowaniu problemu logarytmów dyskretnych na krzywe eliptyczne. Zaletą systemów opartych o krzywe eliptyczne jest znacznie dłuższy czas potrzebny na ich złamanie niż w przypadku logarytmów dyskretnych, dlatego stosuje się to podejście – mówi się o tzw. problemie logarytmów dyskretnych z arytmetyką opartą o krzywe eliptyczne. Najczęściej systemy te rozważa się jako odpowiedniki DSA, nazywając je Elliptic Curve DSA (w skrócie ECDSA) lub jako odpowiedniki ElGamala – tzw. Elliptic Curve Encryption Schemes (ECES).

W przypadku próby zdefiniowania odpowiednika systemów bazujących na problemie faktoryzacji, jak np. RSA, okazało się że taki

system nie wykazuje żadnych zalet w stosunku do RSA wobec czego wariantu tego się nie rozważa.

Przystępując do zaprojektowania systemu kryptograficznego ECDSA albo ECES, konieczne jest ustalenie pewnych parametrów systemowych (dla porównania, system RSA nie wymaga takich parametrów) – związane jest to z wyborem odpowiedniej krzywej eliptycznej wraz z punktem należącym do tej krzywej, zwanym generatorem (oznacza się go literą *G*). Ze względu na cechy matematyczne krzywych eliptycznych (skomplikowanie jak i pewne ograniczenia), wybór krzywych eliptycznych powinien być uzasadniony i opierać się o wiedzę matematyczną w tym kierunku.

Po ustaleniu parametrów systemowych (z których korzysta wiele użytkowników w danej grupie korespondencyjnej) wygenerowanie pary kluczy: prywatnego i publicznego nie stanowi już problemu. Klucz prywatny stanowi tajna, losowa wartość *k* natomiast publiczny – *k*-ta wielokrotność generatora *G*: *kG*. Zakłada się że trudno jest wyznaczyć wartość *k* dysponując złożeniem *kG*.

W przypadku kryptoanalizy systemów RSA, rozmiar problemu zależy od wielkości czynnika *n*; odnośnie systemów opartych o krzywe eliptyczne, trudność związana jest z liczbą punktów wybranych na krzywej eliptycznej – zakłada się z kolei że liczba tych punktów jest równa rzędowi wybranego pola (liczbie elementów w tym polu). Dla przykładowego 160-bitowego pola $GF(2^{160})$, system oparty o krzywe eliptyczne będzie tak samo odporny na atak jak RSA z 1024-bitowym czynnikiem *n*.

Wykorzystanie mniejszych rozmiarów kluczy w systemie opartym o krzywe eliptyczne w stosunku do RSA ma zaletę nie tylko w przypadku problemu przechowywania samych kluczy ale i skutkuje podniesieniem wydajności algorytmu. Obecnie rozważa się również, że rozwiązywanie problemów logarytmów dyskretnych opartych o arytmetykę krzywych eliptycznych jest trudniejsze od faktoryzacji.

Metoda krzywych eliptycznych uznawana jest za bardzo szybką jeśli zaimplementuje się ją sprzętowo. Występują jednak implementacyjne problemy z bezpieczeństwem, ze względu na efektywne wyznaczanie klucza.

6.1.4. System Rabina

Rabin zaproponował system w którym można udowodnić, że znajdowanie klucza prywatnego i fałszerstwo jest równie trudne jak proces faktoryzacji. System ten ma w tym przypadku przewagę nad RSA, w którym takiego dowodu nie można przeprowadzić, czyli teoretycznie możliwe byłoby przeprowadzenie prostszej operacji w celu złamania metody. Złamanie metody Rabina wymaga pomyślnego procesu faktoryzacji.

Kluczem prywatnym w algorytmie Rabina jest para liczb pierwszych *(p, q)* natomiast kluczem publicznym – czynnik $n = p*q$. W algorytmie Rabina w wyniku procesu odszyfrowania uzyskuje się cztery wersje wiadomości jawnej z których tylko jedna jest poprawna. W przypadku gdy szyfrowane były dane binarne, w rozpoznaniu prawidłowej wiadomości posłużyć może dodatkowy nagłówek dołączony do wiadomości.

Metoda Rabina jest jednak podatna na specyficzny rodzaj ataku, w którym fałszerz podsuwa stronie wysyłającej wiadomość specjalnie spreparowany formularz.

6.1.5. System podpisów cyfrowych DSS

6.1.5.1. Wprowadzenie do podpisów cyfrowych

Podpisy cyfrowe wykorzystane są do identyfikacji źródła wiadomości z którego została nadana oraz integralności wiadomości wykorzystując cyfrowy odcisk palca; najmniejsza modyfikacja wiadomości powoduje natychmiastowe jej wykrycie przez mechanizmy hashujące.

Obecnie podpis cyfrowy w USA jest równoprawny z podpisem odręcznym. Pomysł wspierany jest przez międzynarodową organizację Verisign która podjęła się zadania dostarczania certyfikatów kluczy. W architekturze certyfikacji kluczy – PKI (Public Key (certificates) Infrastructure), tworzonej przez urzędy certyfikacji, łączy się klucze wykorzystane do podpisów cyfrowych z informacją osobistą.

Proces certyfikacji wymaga istnienia, dla obu korespondujących stron, zaufanej trzeciej strony (tzw. „trusted authority") komunikującej się z nimi, a więc instytucji która wydaje certyfikat (urząd certyfikacji) stwierdzający autentyczność klucza publicznego i prawo wykorzystania go przez konkretnego użytkownika.

Jednemu podpisowi cyfrowemu może towarzyszyć jeden lub więcej certyfikatów.

Certyfikacja kluczy daje gwarancję, że osoba, która użyła klucza jest tą, za którą się podaje, ponieważ pozwala na weryfikację czy złożony podpis cyfrowy jest autentyczny. Oto dane jakie zawiera certyfikat:

- dane osobiste podmiotu dla którego wystawiany jest certyfikat;
- klucz publiczny posiadacza certyfikatu;
- dane urzędu certyfikującego.

Podpis cyfrowy może posłużyć do uwierzytelnienia użytkowników w przypadku gdy wymaga się od nich aby w domenie publicznej złożyli podpis w celu uzyskania dostępu do pewnych zasobów sieciowych. Architektura uwierzytelniania użytkowników zapewnia poufność nie tylko użytkownikom Internetu, ponieważ może być również zrealizowana np. na nośnikach typu smart card.

Jeśli w przyszłości architektura PKI ma rozwinąć system nadzoru, ważne jest aby uwzględnić w niej przede wszystkim aspekt nienaruszalności prywatności. System certyfikacji kluczy wiąże się z podjęciem pewnego ryzyka, jeśli osoba nieuprawniona mająca dostęp do danych może analizować ruch między użytkownikami.

6.1.5.2. System DSS

Standardem cyfrowego uwierzytelniania wykorzystywanym przez Rząd Stanów Zjednoczonych stał się w 1994 roku DSS (Digital Signature Standard), który określa algorytm DSA (Digital Signature Algorithm) ustanowiony przez NIST (National Institute of Standards and Technology), wspólnie z NSA.

DSA bazuje na trudności rozwiązywania problemu logarytmów dyskretnych a więc związany jest z metodami zaproponowanymi przez ElGamala czy Schnorra i jest systemem służącym wyłącznie do uwierzytelniania.

DSA posiada mechanizmy umożliwiające zarówno generację jak i weryfikację podpisów. Spotyka się różne sposoby implementacji DSA – sprzętowe lub programowe. DSA wykorzystuje się m.in. w procesach rozprowadzania oprogramowania, składowania i cyfrowej wymiany danych, w usługach takich jak poczta elektroniczna.

Podpis cyfrowy wygenerowany w wyniku przebiegu algorytmu DSA jest parą dużych liczb. W procesie generacji podpisu wykorzystuje się klucz prywatny, natomiast w procesie weryfikacji – klucz publiczny danego użytkownika.

W DSA wykorzystuje się SHA jako mechanizm wytworzenia skrótu wiadomości. W algorytmie wykorzystuje się następujące parametry:

- p – liczba pierwsza o długości od 512 do 1024 bitów z krokiem co 64 bity;
- q – 160-bitowa liczba pierwsza będąca dzielnikiem wartości $p-1$;
- g, które ma postać:

$$g = h^{\frac{(p-1)}{q}} \bmod p \quad , \tag{6.26}$$

gdzie h - dowolna liczba mniejsza od $p-1$, dla której zachodzi:

$$h^{\frac{(p-1)}{q}} \bmod p > 1 \quad . \tag{6.27}$$

Powyższe wartości to parametry systemowe, które mogą być rozpowszechnione w obrębie danej grupy korespondencyjnej użytkowników. Ponadto określa się:

- x, k – dowolne liczby całkowite dodatnie, mniejsze od q;
- $y = g^x \bmod p.$ (6.28)

Liczba x jest kluczem prywatnym a y – publicznym; k musi być generowana przy każdej generacji podpisu na nowo.

Podpis wiadomości M tworzą dwie liczby: r i s, wyznaczone w następujący sposób:

$$r = (g^k \bmod p) \bmod q; \tag{6.29}$$

$$s = (k^{-1} (SHA(M)) + xr)) \bmod q. \tag{6.30}$$

k^{-1} w równaniu (6.30) jest odwrotnością multiplikatywną k modulo q, czyli

$$(k^{-1} * k) = 1 \bmod q; \tag{6.31}$$

SHA(M) to funkcja wykonująca skrót wiadomości *M*.

Podpis wysyłany jest wraz z wiadomością do odbiorcy. Ponieważ odbiorca nie ma jeszcze pewności co do prawdziwości podpisu czy integralności danych, zakłada się że otrzymany podpis składa się z wartości *s'*, *r'*, a otrzymaną wiadomość oznacza się przez *M'*. Odbiorca najpierw sprawdza czy $r', s' \in [0, q]$. Jeśli ten warunek nie jest spełniony, wynik weryfikacji jest negatywny i wiadomość powinna być odrzucona. W przeciwnym wypadku, odbiorca oblicza najpierw:

$$w = (s')^{-1} \bmod q; \tag{6.32}$$

$$u_1 = (SHA(M') * w) \bmod q; \tag{6.33}$$

$$u_2 = (r'w) \bmod q; \tag{6.34}$$

$$v = ((g^{u_1} \times y^{u_2}) \bmod p) \bmod q \quad . \tag{6.35}$$

Weryfikacja ma pozytywny wynik jeśli v = r.

Podobnie jak w większości systemów kryptograficznych z kluczem publicznym, w systemie DSS duże znaczenie ma wybór „silnej” liczby pierwszej. Aby istniała pewność, że liczby pierwsze zostały wybrane w sposób losowy, system DSS zawiera mechanizmy sprawdzające poprawność wybranych parametrów *p* czy *q*. W tym celu korzysta się z funkcji hashującej w odniesieniu do wartości *s*. Jeśli wartość wynikowa funkcji, przykładowo *p*, spełnia narzucone kryteria (jak np. długość liczby pierwszej), wówczas parametr *p* zostaje użyty w algorytmie, natomiast wartość *s* zachowywana jest jako dowód na losowość liczby *p*. Osoba która chce się przekonać że *p* zostało wybrane w sposób losowy, wykonuje tę samą funkcję hashującą na wartości *s* i sprawdza czy zgadza się z parametrem systemowym *p*.

Wykorzystanie funkcji hashującej zezwala na większą tolerancję przy wyborze parametrów systemowych przez użytkownika. Ponieważ wybór klucza publicznego jest uzależniony bezpośrednio od klucza prywatnego, odpowiednie wymagania co do wyboru nakłada się na właśnie na klucz prywatny.

Jako najczęściej stosowany do uwierzytelniania, system RSA był początkowo przyczyną małej popularności DSS (liczono, że ten pierwszy zostanie uznany za oficjalny standard), a wręcz krytycyzmu tego systemu: zwrócono uwagę na brak możliwości zmiany długości klucza (z początku była to ustalona wartość równa 512 bitów, jak się szybko okazało – zbyt mała), na to iż jest stosunkowo nowy aby już podlegał standaryzacji, ponieważ nie zapewnia wystarczającego bezpieczeństwa które wynikałoby z badań i testów.

Ponadto postulowano, że DSS jako drugi zaimplementowany system obok istniejącego już RSA może stanowić dodatkową niewygodę. Również krytykuje się utajniony sposób w jaki NIST wybrało DSS, w wyniku dużego wpływu NSA na tę decyzję.

W praktyce, w systemie DSS generacja podpisu jest szybsza niż jego weryfikacja, podczas gdy w RSA jest na odwrót. W DSA można jeszcze podnieść efektywność procesu podpisywania wiadomości wykonując wstępne obliczenia wartości dotyczących klucza. Gdyby się jednak zastanowić nad zastosowaniem podpisu cyfrowego – wiadomość jest podpisywana tylko raz, a sprawdzana wiele razy – dochodzi się do wniosku, że znacznie lepiej byłoby przyspieszyć proces weryfikacji, czego próby również podjęto.

Ostatecznie, postulaty porównawcze NIST odnośnie DSS i RSA zaowocowały zainteresowaniem się DSS.

6.1.5.3. Protokoły wiedzy zerowej

Niektóre metody identyfikacji personalnej, jak metoda zaproponowana przez Amosa Fiata i Adi Shamira, bazująca na tzw. interaktywnych protokołach wiedzy zerowej (zero-knowledge protocols) może być przystosowana do wykorzystania w podpisywaniu dokumentów.

Protokoły wiedzy zerowej stosuje się do realizacji następującego problemu:

- pewna osoba uważa że zna tajną informację;
- osoba ta chce przekonać innych (osoby które chcą zweryfikować pogłoskę) że zna tajną informację;
- osoba ta udowadnia fakt znajomości tajnej informacji bez przekazywania tej wiadomości osobie weryfikującej.

Protokoły wiedzy zerowej mają następujące własności:

- prawdopodobieństwo że wiadomość jest fałszywa wynosi $\left(\frac{1}{2}\right)^n$ w przypadku wykonania *n* sprawdzeń autentyczności wiadomości;
- dowód powiedzie się tylko wtedy gdy tajna informacja rzeczywiście jest autentyczna i osoba ją znająca jest w stanie udowodnić jej autentyczność;
- dowód powiedzie się z bardzo małym prawdopodobieństwem jeśli rzekomo tajna informacja nie jest autentyczna albo jeśli osoba proklamująca jej znajomość nie jest w stanie udowodnić jej autentyczności;
- dowód może przeprowadzić osoba weryfikująca, której nie został przedstawiony dowód na autentyczność pogłoski.

Oprócz dowodu autentyczności wiadomości, protokoły wiedzy zerowej stosuje się również do:

- podpisywania dokumentów, tak jak w przypadku metody Fiata-Shamira;
- uwierzytelniania:

    a) bezpieczne logowanie,

    b) sprawdzenie tożsamości przed rozpoczęciem procesu wymiany kluczy,

    c) zastosowanie w kartach typu smart cards lub systemach wbudowanych.

W odniesieniu do zasady działania protokołów wiedzy zerowej, przystosowanie metody Fiata-Schamira do podpisów cyfrowych wymaga przekształcenia mechanizmu sprawdzającego autentyczność wiadomości do funkcji hashującej. Bezpieczeństwo algorytmu zależy od prawdopodobieństwa określonego potęgą $\left(\frac{1}{2}\right)$ i od trudności faktoryzacji czynnika *n*.

Zaletą metody Fiata-Shamira jako systemu podpisu cyfrowego w stosunku do metody RSA jest szybkość wykonywania operacji; jak podaje literatura, metoda Fiata-Schamira wymaga zaledwie co najwyżej 4% mnożeń modularnych systemu RSA.

6.2. Systemy kryptograficzne z kluczem publicznym

6.2.1. Wprowadzenie

Systemu z kluczem publicznym powinien zawierać kluczowe zagadnienia pozwalające na stworzenie wygodnego, bezpiecznego i wydajnego systemu.

Należy sobie uzmysłowić jakie usługi bezpieczeństwa oferują konkretne algorytmy kryptograficzne i czy spełniają one określone wymagania. Oczekuje się, aby algorytmy klucza publicznego pozwalały np. sprawdzić autentyczność pochodzenia danych czy uwzględniały proces ustalania kluczy przez strony biorące udział w korespondencji. W tym celu wykorzystać można różnego rodzaju algorytmy – metody cyfrowych podpisów, szyfrowania czy uzgadniania kluczy.

Oprócz podstawowych zagadnień funkcjonalnych systemu - kwestia generacji i zarządzania kluczem, procedury szyfrowania, podpisywania, weryfikacji i hashowania realizowane przez algorytmy kryptograficzne – czyli zapewnienie poufności, autentyczności czy integralności, należy położyć nacisk na:

- ***efektywność.*** Własność tą rozważa się zawsze pod kątem przeznaczenia systemu. Do osiągnięcia zamierzonego efektu bierze się pod uwagę zarówno szybkość wykonywanych operacji jak i rozmiar danych które wykorzystywane będą w algorytmach. Może się również okazać, że rozmiar nie tyle strumienia przetwarzanych danych co samych parametrów systemowych stanowić będzie przeszkodę w osiągnięciu oczekiwanej wydajności;
- ***bezpieczeństwo.*** W tym przypadku mowa jest nie tyle o bezpieczeństwie wybranego algorytmu, wynikającego z idei jego działania ale głównie o solidnej jego implementacji – dotyczy to jakości kodu, który może okazać się podatny na ataki, a wręcz może zawierać błędy, które nie zostały odpowiednio wcześnie wykryte;
- ***zaufanie do wybranej metody.*** Miarę pewności w stosunku do algorytmu realizującego określone czynności kryptograficzne określają długoletnie badania nad jego bezpieczeństwem. Wynika to z faktu że algorytmy kryptograficzne bazują na problemach trudnych których pewności nie da się dowieść wprost.

W dalszej części wykładu poświęcono uwagę nowoczesnym algorytmom: szyfrowania RSA-OAEP i stosowanemu do podpisu cyfrowego RSA-PSS, wspieranym przez projekty NESSIE i CRYPTREC.

6.2.2. Projekt NESSIE

Projekt NESSIE (New European for Signature, Integrity and Encryption) gromadzi partnerów z siedmiu państw Europy – Belgii, Francji, Włoch, Wielkiej Brytanii, Niemiec, Izraela i Norwegii.

Zasadniczym celem projektu NESSIE jest wyeksponowanie znaczenia obiecujących algorytmów kryptograficznych, uzyskanych w drodze otwartego konkursu oraz otwartego procesu oceny rozwoju tych algorytmów, jak i rozwój samych technik testowania algorytmów. Wynikiem takich działań jest ostatecznie stworzenie systemu bezpiecznego i wydajnego.

NESSIE obejmuje zarówno algorytmy służące do uwierzytelniania jak i zachowania poufności. Oprócz algorytmów klucza publicznego - szyfrowania i podpisu cyfrowego, NESSIE obejmuje również szyfry blokowe i strumieniowe, funkcje hashujące, algorytmy MAC i funkcje pseudolosowe.

Poprzednikiem NESSIE był projekt RIPE, realizowany na przełomie lat '80 i '90. W wyniku zainteresowania projektem, powstały metody kryptograficzne spośród których wiele standaryzowano – np. algorytm hashujący RIPEMD.

W przeciwieństwie do japońskiego CRYPTREC, który zostanie omówiony później, rezultaty NESSIE nie mają znaleźć zastosowania w celach rządowych, choć same algorytmy podlegać mogą standaryzacji.

W NESSIE niewątpliwie interesująca jest faza testowania algorytmów, dzięki której algorytmy badane są pod kątem zapewnienia długoterminowego bezpieczeństwa, pod kątem wymagań rynku, efektywności i stopnia funkcjonalności, w tym przystosowania do różnego rodzaju środowiska komputerowego w którym pracują. Fakt przeprowadzania testów służy rozwojowi tego typu oprogramowania, aby w przyszłości móc lepiej i szybciej testować nowo powstałe algorytmy. Projekt NESSIE obejmuje zagadnienia bezpieczeństwa, wydajności i własności intelektualnej.

Analiza algorytmów w NESSIE przebiegała w następujących krokach:

- wstępne sprawdzenie poszczególnych kandydujących algorytmów (ich spójności i niezależności);
- dokładne sprawdzenie poszczególnych algorytmów (w przypadku stwierdzenia wad dany algorytm wymaga optymalizacji);
- poddanie algorytmów analizie bezpieczeństwa i wydajności w celu stworzenia raportu określającego na podstawie jakich założeń zapewnione są dane aspekty bezpieczeństwa; szacowanie bezpieczeństwa czy tworzenie modelów bezpieczeństwa;
- analiza wydajnościowa obejmuje różne zastosowania algorytmów testowanych na różnych platformach sprzętowych – m.in. z wykorzystaniem układów FPGA czy kart smart cards. W zależności od zastosowanych platform sprzętowych, pod główną ocenę podlegają takie aspekty jak: wykorzystanie pamięci operacyjnej, szybkość, rozmiar kodu, przepustowość czy opóźnienia;
- badanie podatności na różnego rodzaju ataki;
- stwierdzenie różnic w operacjach szyfrowania i odszyfrowania, podpisu i weryfikacji.

6.2.3. Projekt CRYPTREC

Japoński rząd również wyraził inicjatywę rozwoju bezpieczeństwa sieci komunikacyjnych w Japonii. W ramach strategii e-Japan, mającej na celu stworzenie infrastruktury dla elektronicznego rządu, powstał projekt podobny do europejskiego NESSIE pod nazwą CRYPTREC, którego celem było zgromadzenie metod kryptograficznych, ich przetestowanie pod względem wydajności i bezpieczeństwa i ostatecznie opublikowanie rezultatów w raporcie technicznym.

Projekt CRYPTREC oprócz algorytmów klucza publicznego obejmuje również metody symetryczne, funkcje hashujące i generatory liczb pseudolosowych.

Względem algorytmów asymetrycznych wymaga się następujących aspektów: poufność, autentyczność, proces uzgadniania klucza.

Ocena bezpieczeństwa algorytmów w projekcie CRYPTREC określana jest przez analizę zastosowanych założeń matematycznych (np. faktoryzacji, problemu logarytmów dyskretnych), funkcji pomocniczych, implementacji metody, ilości danych przekazywanych między korespondentami. Algorytmy poddawane są weryfikacji z uwzględnieniem argumentów które mają przemawiać za bezpieczeństwem metody. Wymaga się aby metoda nie miała zbyt dużego zapotrzebowania na zasoby sprzętowe.

Istnienie takich projektów jak NESSIE czy CRYPTREC niesie wiele korzyści zarówno dla grup badawczych zajmujących się kryptografią jak i osób zajmujących się implementacją metod kryptograficznych czy użytkowników. Projektant metod kryptograficznych podejmuje się optymalizacji wydajności i rozważa dalsze praktyczne możliwości ich zastosowania. Użytkownik jest w stanie określić zalety dobrze zdefiniowanych algorytmów.

6.2.4. Nowoczesne systemy kryptograficzne klucza publicznego

6.2.4.1. RSAES-OAEP

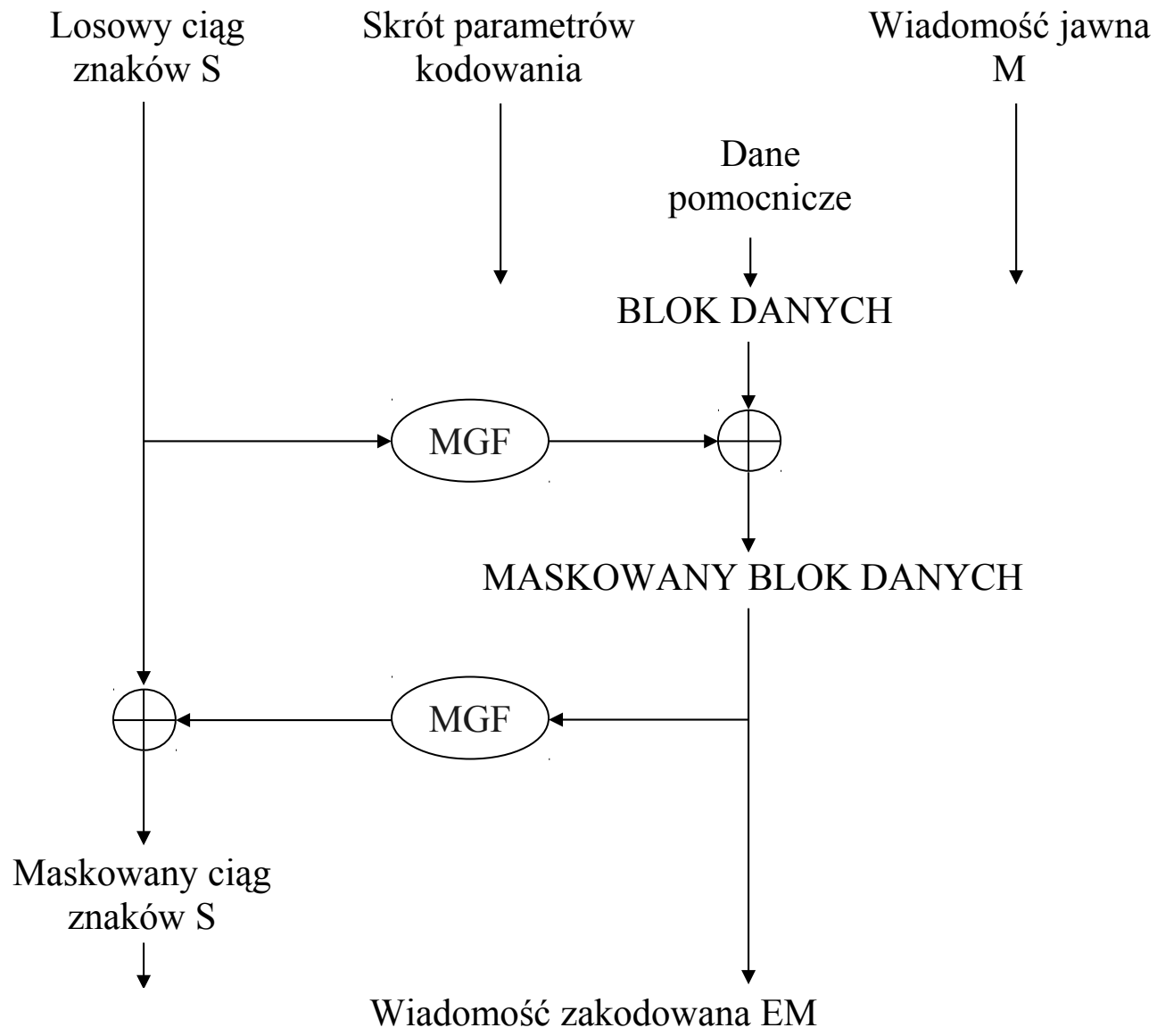
RSAES-OAEP (Rivest Shamir Adelman Encryption Scheme - Optimal Asymmetric Encryption Padding) jest metodą klucza publicznego do szyfrowania wiadomości, zalecaną do wykorzystania w nowych zastosowaniach. System łączy w sobie metodę szyfrowania OAEP i zasadę o którą opiera się RSA – dla wiadomości jawnej *m* uzyskiwana jest wiadomość tajna *c* według wzoru $c = m^e \ (mod\ n)$, natomiast postać jawna z zaszyfrowanej wiadomości *c* uzyskiwana jest według wzoru $m=c^d (mod\ n)$.

W skrócie, metoda polega na przekształceniu wiadomości jawnej w zaszyfrowaną z wykorzystaniem metody OAEP, a następnie rezultat podlega zaszyfrowaniu metodą RSA z wykorzystaniem klucza publicznego. Odszyfrowanie odbywa się z zastosowaniem metod w odwrotnej kolejności.

Twórcami metody OAEP są Mihir Bellare i Phillip Rogaway. Operacja kodowania (OAEP-ENCODE) przekształca jawną wiadomość *M* do danych zakodowanych *EM* określonej długości, zadanej przez parametry kodowania - funkcję skrótu oraz funkcję generacji maski (MGF). Operacja odkodowania (OAEP-DECODE), która odwzorowuje kod *EM* na wiadomość jawną *M* jest funkcją odwrotną do kodowania.

Funkcja generacji maski (MGF) bazuje na funkcji skrótu. Na wejściu pobiera zmiennej długości dane generując ciąg danych określonej długości jako rezultat. Dane wyjściowe są ściśle powiązane z danymi wejściowymi. Wymaga się, aby generowane dane były pseudolosowe – w istocie poziom bezpieczeństwa funkcji MGF bazuje na naturze funkcji skrótu.

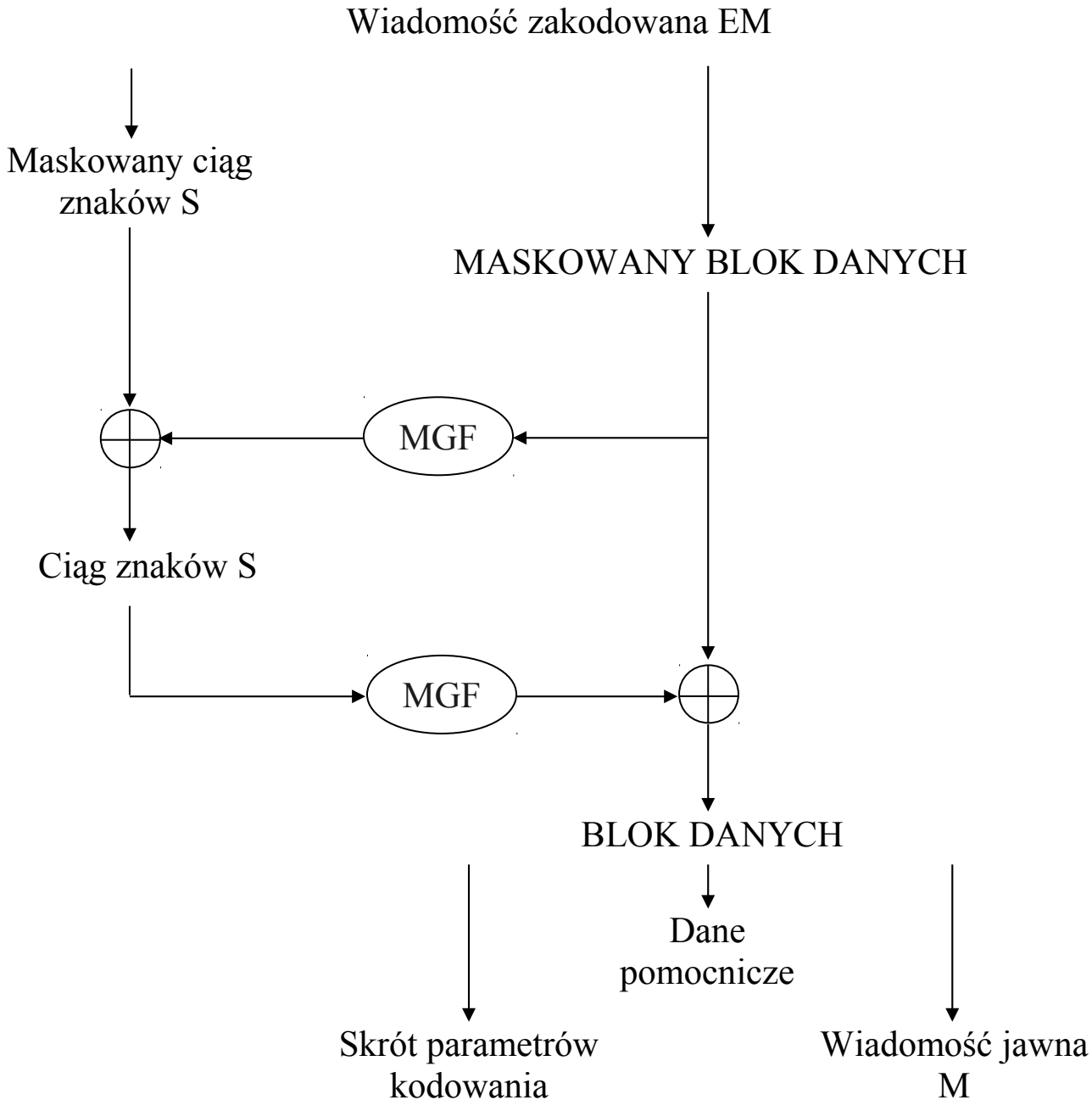
Zakodowana wiadomość posiada strukturę która jest weryfikowalna przez operację dekodowania – dzięki tej operacji z zakodowanej wiadomości można uzyskać skrót parametrów kodowania. Poniżej przedstawiony jest schemat kodowania. Dane pomocnicze stanowi oktet o wartości 1.



Rys. 6.1. Kodowanie wiadomości metodą OAEP (OAEP-ENCODE):

| symbol oznacza operację XOR

W procedurze odkodowania funkcja OAEP-DECODE przyjmuje za parametr wiadomość zakodowaną oraz inne parametry kodowania ale tym razem bez długości wiadomości wyjściowej. Odkodowanie zaprezentowane jest na rys. 6.2.



Rys. 6.2. Odkodowanie wiadomości metodą OAEP (OAEP-DECODE): symbol ⊕ oznacza operację XOR

W oparciu o podstawy matematyczne RSA i powyżej przedstawione kodowanie OAEP, ***szyfrowanie wiadomości M w metodzie RSAES-OAEP realizowane jest zgodnie z algorytmem:***

1) wyznaczenie kodu *EM* na podstawie wiadomości *M*:

EM = OAEP-ENCODE (M, parametry kodowania, długość wyjścia),

długość wyjścia określa długość wiadomości *EM* w bajtach;

2) zastosowanie metody RSA z kluczem publicznym *(n, e)* na wiadomości *em* (liczbowa reprezentacja ciągu znaków *EM*) i uzyskanie szyfru *C*.

W procedurze odszyfrowania potrzebne jest podanie poprawnego klucza prywatnego *K*. ***Odszyfrowanie wiadomości C metodą RSAES-OAEP wygląda następująco:***

1) zastosowanie odszyfrowania metodą RSA z kluczem *K* na wejściowym szyfrogramie *c* (liczbowa reprezentacja szyfru *C*) w celu uzyskania liczby całkowitej *em;*
2) odkodowanie wiadomości *EM* (reprezentacja wartości *em* w postaci ciągu znaków):
   M = OAEP-DECODE (EM, parametry kodowania).

Dowiedziono, że jeśli obliczenie odwrotności funkcji jednostronnej zastosowanej w RSA jest trudne bez wykorzystania klucza prywatnego, wówczas za pomocą odpowiedniej metody szyfrowania, opartej o OAEP, kryptoanalityk nie jest w stanie uzyskać poprawnej wiadomości zaszyfrowanej bez znajomości wiadomości jawnej. Pod tym względem metoda odporna jest na atak przeciwko wybranemu szyfrogramowi.

Niestety słabość systemu leży w samej metodzie RSA – jeśli możliwe jest częściowe obliczenie odwrotności funkcji jednostronnej (mówi się o tzw. funkcji częściowo jednostronnej), ale nie da się w całości obliczyć odwrotności, możliwe staje się złamanie całego systemu. Funkcja jest częściowo jednostronna gdy pierwsze *n* bitów wiadomości nie może być wyznaczone z odwrotności tej funkcji.

6.2.4.2. RSA-PSS

System klucza publicznego RSA-PSS, podobnie jak RSAES-OAEP, jest kombinacją dwóch metod kryptograficznych – algorytmu RSA i PSS (Probabilistic Signature Scheme), choć ma on posłużyć do składania cyfrowych podpisów i ich weryfikacji. PSS wynaleziony został przez Mihira Bellare i Phillipa Rogawaya.

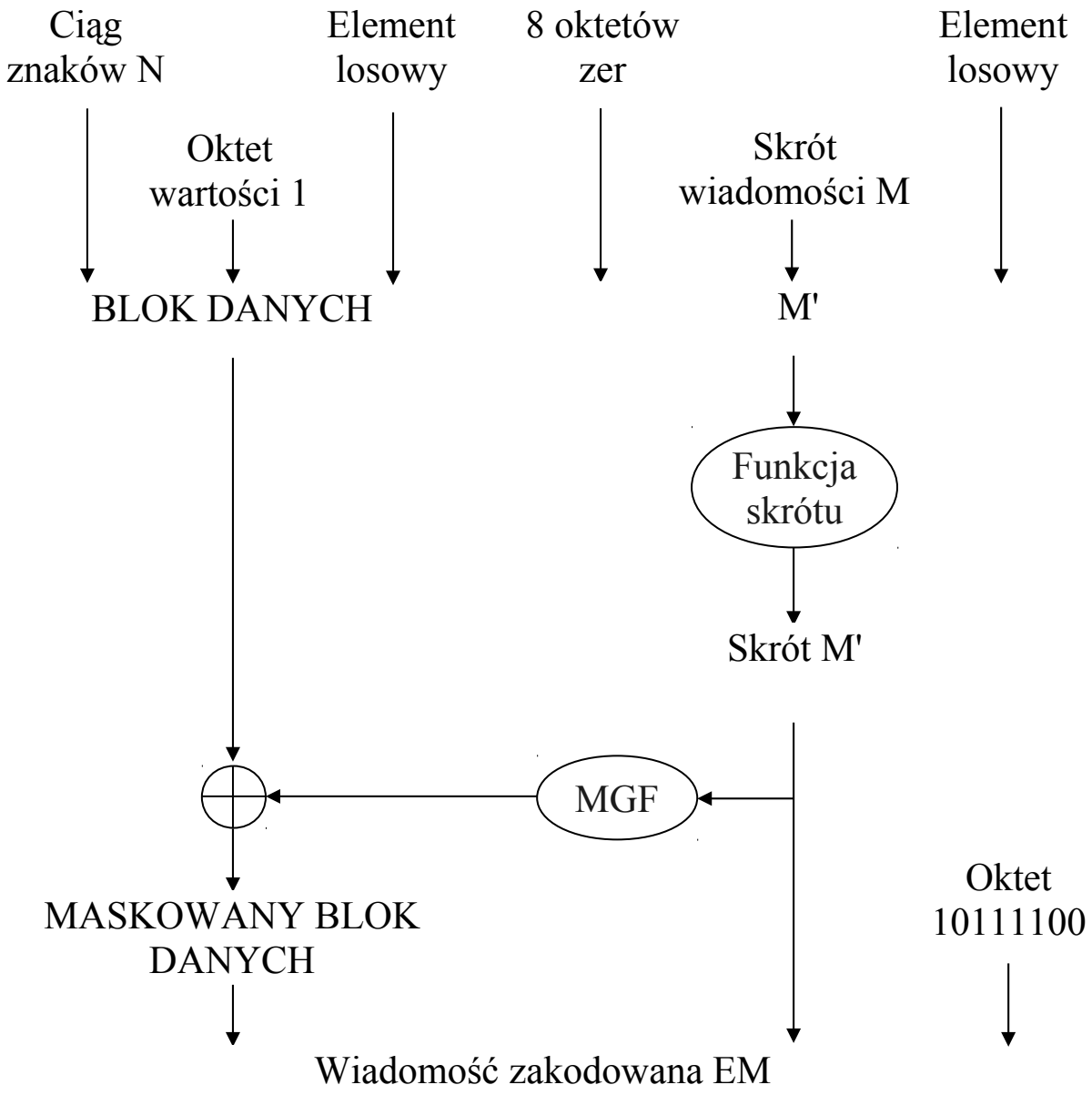
Metoda RSA wykorzystana jest tu do generacji podpisu na podstawie klucza prywatnego i zakodowanej wiadomości oraz do weryfikacji podpisu z wykorzystaniem klucza publicznego.

Funkcja odpowiedzialna za kodowanie wiadomości (PSS-ENCODE) oblicza skrót wiadomości a następnie przekształca uzyskany rezultat za pomocą funkcji generacji maski (opisanej w p. 6.2.4.1) do zakodowanej postaci określonej długości. W operacji odwrotnej (PSS-DECODE), z zakodowanej wiadomości odzyskiwany jest skrót, który jest następnie porównywany ze skrótem oryginalnej wiadomości.

W metodzie PSS funkcja generacji maski wykorzystuje bezpośrednio funkcję skrótów do zakodowania wiadomości, przy czym wykorzystana funkcja hashująca nie jest dołączana do zakodowanej wiadomości, tylko jej rezultat (skrót). Jeśli ktoś podjąłby próbę wykorzystania innej funkcji hashującej do weryfikacji podpisu to próba ta zakończyłaby się niepowodzeniem, ponieważ funkcja generacji maski zależy tu nie od samej wartości skrótu lecz od użytej funkcji skrótu.

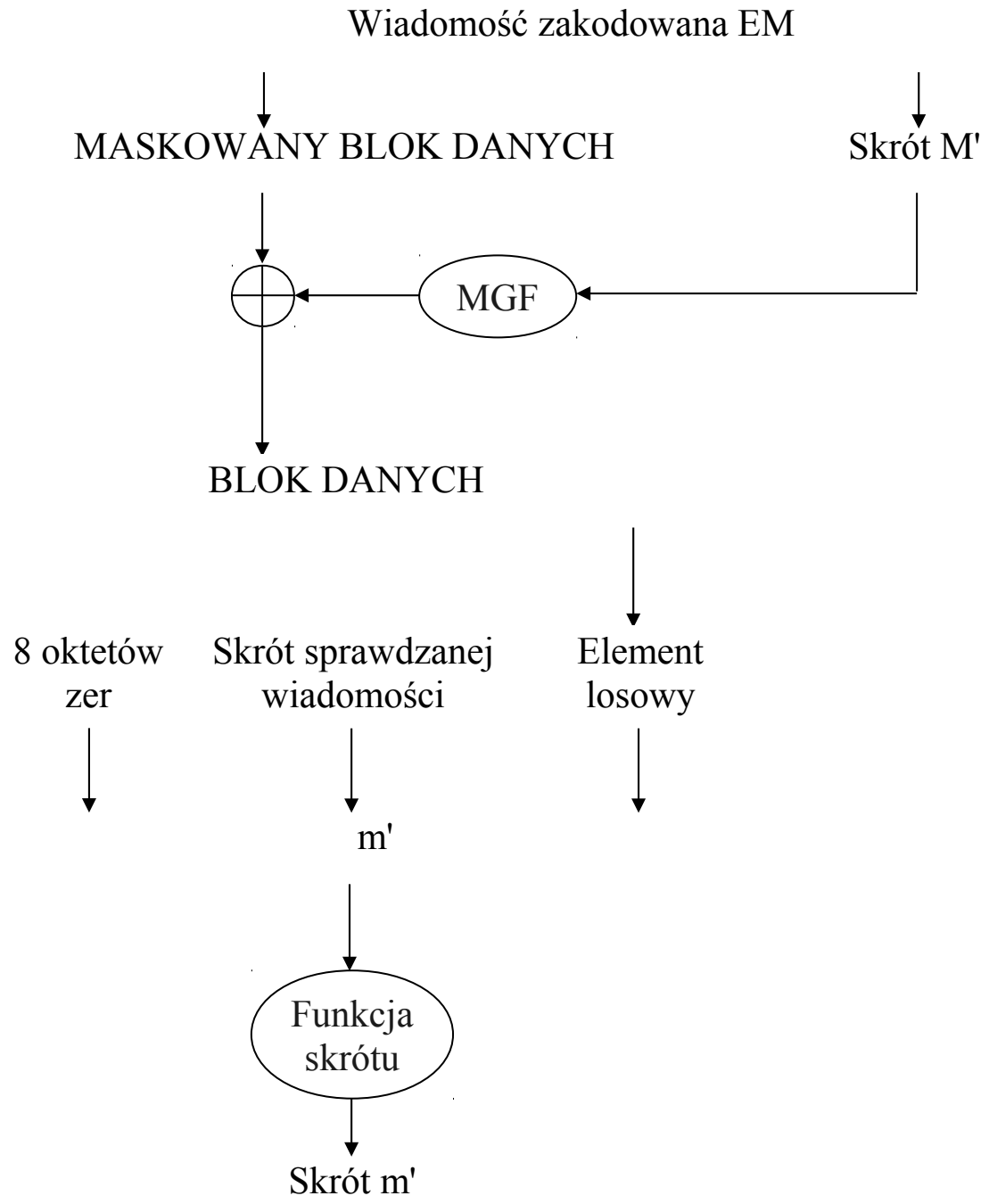
PSS jest probabilistyczną metodą kodowania ponieważ wykorzystuje, oprócz innych danych wejściowych, losowy element – zapewnia to wyższy stopień bezpieczeństwa bez nakładania szczególnych wymagań na funkcję skrótów – nawet znalezienie identycznego skrótu dla dwóch różnych wiadomości nie oznacza że podpis mógłby zostać podrobiony, ponieważ atakującemu nie znana jest wartość losowa.

W procesie ***kodowania*** na podstawie jawnej wiadomości M, tworzony jest skrót i powstaje ciąg danych M', składający się z ośmiu oktetów zer oraz ze skrótu wiadomości M i elementu losowego. Jednocześnie tworzony jest blok danych składający się z wygenerowanego dowolnie ciągu znaków z określoną liczbą oktetów zer (oznaczony jako N), jednego oktetu wartości 1 i elementu losowego. Operacja kodowania przedstawiona jest na rys. 6.3.



Rys. 6.3. Kodowanie wiadomości metodą PSS(PSS-ENCODE):
symbol ⊕ oznacza operację XOR.

Podczas przebiegu operacji ***odkodowania***, pozyskany z wiadomości zakodowanej skrót *M′* porównywany jest ze skrótem otrzymanego rezultatu (*m′*) i na tej podstawie odbywa się weryfikacja podpisu. Do uzyskania ciągu danych *m′* potrzebny jest skrót wiadomości otrzymanej przez odbiorcę stąd jednym z parametrów dekodowania jest właśnie ta wiadomość, która podlegać będzie weryfikacji.



Rys. 6.4. Odkodowanie wiadomości metodą PSS (PSS-DECODE):

symbol ⊕ oznacza operację XOR

Przebieg operacji złożenia podpisu cyfrowego S wiadomości M z wykorzystaniem klucza prywatnego K metody RSA w systemie RSA-PSS wygląda następująco:

1) zakodowanie wiadomości *M* do postaci *EM* za pomocą PSS-ENCODE;
2) wygenerowanie podpisu *s* (reprezentacja liczbowa ciągu znaków *S*) z wykorzystaniem metody RSA z parametrami –- klucz prywatny *K* i *em* – reprezentacja liczbowa *EM*.

Operacja weryfikacji podpisu w RSA-PSS wymaga podania klucza publicznego (n, e) wykorzystanego przez metodę RSA oraz sprawdzanej wiadomości M na podstawie podpisu S:

1) z wykorzystaniem liczbowej reprezentacji podpisu – *s*, obliczana jest wiadomość

$$m = s^e \bmod n; \tag{6.36}$$

2) w celu przeprowadzenia operacji odkodowania PSS-DECODE, należy najpierw uzyskać ciąg znaków *EM* z wartości *m*.

W RSA-PSS o poziomie bezpieczeństwa decydują matematyczne operacje podpisu i weryfikacji w RSA oraz zasadnicze cechy metody PSS:

- wystąpienie dodatkowego oktetu (10111100), co powoduje znaczne zwiększenie liczby iteracji potrzebnych do uzyskania poprawnej wiadomości w przypadku próby ataku;
- wystąpienie ośmiu oktetów zer i skrótu podpisywanej wiadomości w połączeniu z pewnym elementem losowym sprawia, że słabe punkty funkcji skrótu nie stanowią tu problemu.

Oba omówione systemy – RSAES-OAEP i RSA-PSS charakteryzują się dużym bezpieczeństwem przy rozsądnej szybkości. Zarówno metoda RSAES-OAEP i RSA-PSS wykazują jednak dużą zmienność pod względem efektywności; zaobserwowano, że koszt operacji w metodach OAEP i PSS jest nieznaczny w porównaniu z operacjami metody RSA.

Wymagana jest bezpieczna implementacja systemu – w środowisku projektowym błędy implementacji mogą zostać wyeliminowane przez solidną praktykę projektową, np. wykonywanie odpowiednich testów w cyklu rozwojowym systemu, ochronę informacji służących do generacji klucza prywatnego i liczb pseudolosowych, bezpieczne zarządzanie pamięcią i reakcja na ewentualne błędy.

Musi być również zapewniona ochrona przed atakami – dzięki naturze metod OAEP i PSS cały system zyskuje dużą odporność na różnego rodzaju ataki – wliczając analizę czasu wykonywania czy analizę błędów a także wrażliwość na nieprawidłowe klucze które intruz usiłuje wykorzystać. Oczywiście najlepszym podejściem byłoby zapewnienie ochrony przed atakami już na etapie systemowym a nie algorytmicznym.