Podstawy systemów kryptograficznych z kluczem jawnym RSA

RSA – nazwa pochodząca od nazwisk twórców systemu (Rivest, Shamir, Adleman)

Systemów z kluczem jawnym można używać do szyfrowania operacji przesyłanych między dwiema porozumiewającymi się stronami tak, aby ktoś, kto podsłucha informację, nie był w stanie jej zdekodować.

Ponadto, system taki umożliwia dołączenie na końcu elektronicznie przesyłanej informacji niemożliwy do podrobienia podpis cyfrowy.

Podpis cyfrowy traci ważność, gdy zmieniony zostanie choćby bit przesyłanej informacji, może służyć zatem zarówno do potwierdzenia tożsamości nadawcy jak i treści przesyłanej wiadomości.

Ogólna zasada systemów RSA: łatwo odnaleźć dużą liczbę pierwszą, ale trudno rozłożyć na czynniki iloczyn dwóch dużych liczb pierwszych.

Klucz jawny (public key), klucz tajny (secret key)

Klucz to pewien zestaw informacji. W systemie RSA każdy klucz składa się z pary liczb naturalnych.

Każdy użytkownik tworzy dwa własne klucze: jawny oraz tajny.
Klucz tajny jest trzymany w ukryciu.
Klucz jawny można np. opublikować w ogólnie dostępnym katalogu – tak, żeby dowolny użytkownik mógł bez trudu uzyskać do niego dostęp.

Klucze definiują funkcje, które można stosować do dowolnej wiadomości.
Zakładamy, że funkcje te dają się efektywnie obliczać, gdy znany jest klucz.

„Alicja” – klucz publiczny P_A i odpowiadająca mu funkcja $P_A( )$;
klucz prywatny S_A i odpowiadająca mu funkcja $S_A( )$
„Bob” – klucz publiczny P_B, funkcja $P_B( )$;
klucz prywatny S_B, funkcja $S_B( )$;

Niech D oznacza zbiór dopuszczalnych wiadomości, np. zbiór wszystkich skończonych ciągów bitów.
Funkcje $P_A()$, $S_A()$ itp. są zatem permutacjami zbioru D.

Klucze jawny i tajny każdego użytkownika określają funkcje, które są nawzajem swoimi odwrotnościami:

$$M = S_A(P_A(M))$$
$$M = P_A(S_A(M))$$

dla dowolnej wiadomości M.

Podstawową właściwością systemów z kluczem jawnym jest to, że nikt oprócz właściciela nie jest w stanie obliczyć funkcji $S()$ w rozsądnym czasie. Np. zakładamy, że jedynie Alicja potrafi obliczać $S_A()$ i dlatego trzyma klucz S_A w tajemnicy.

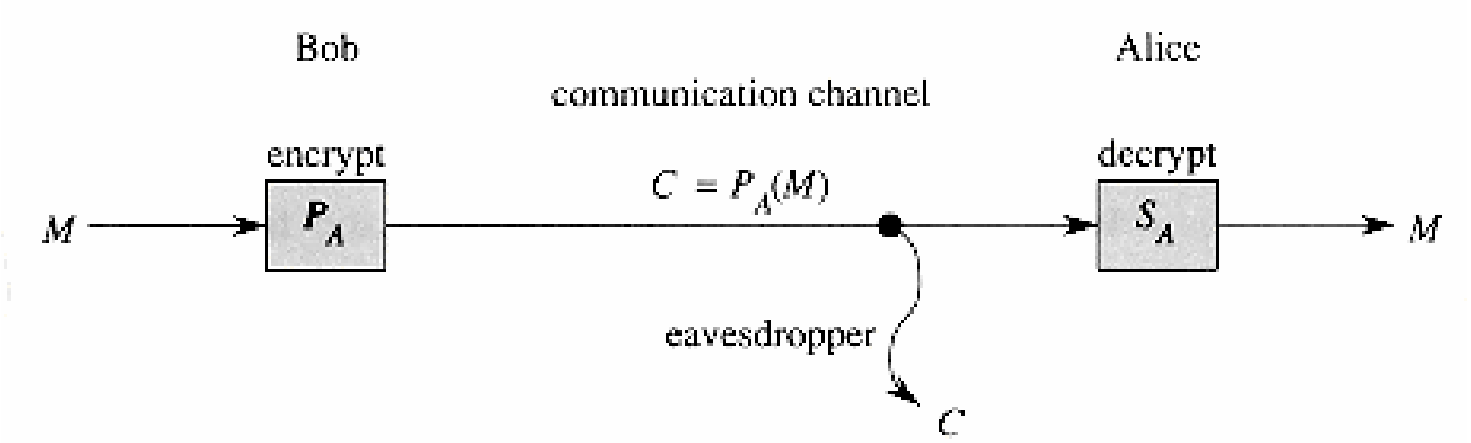
Założenie, że jedynie Alicja potrafi obliczać $S_A(\ )$ musi być spełnione, pomimo, że każdy zna P_A i może efektywnie obliczać $P_A(\ )$ – funkcję odwrotną do $S_A(\ )$.

Główny problem polega więc na tym, że musimy stworzyć system, w którym można ujawnić przekształcenie $P_A(\ )$, bez zdradzania, jak obliczać przekształcenie odwrotne $S_A(\ )$.

Szyfrowanie wiadomości

Przypuśćmy, że Bob chce wysłać Alicji wiadomość M zaszyfrowaną tak, żeby dla osoby podsłuchującej była niezrozumiała.

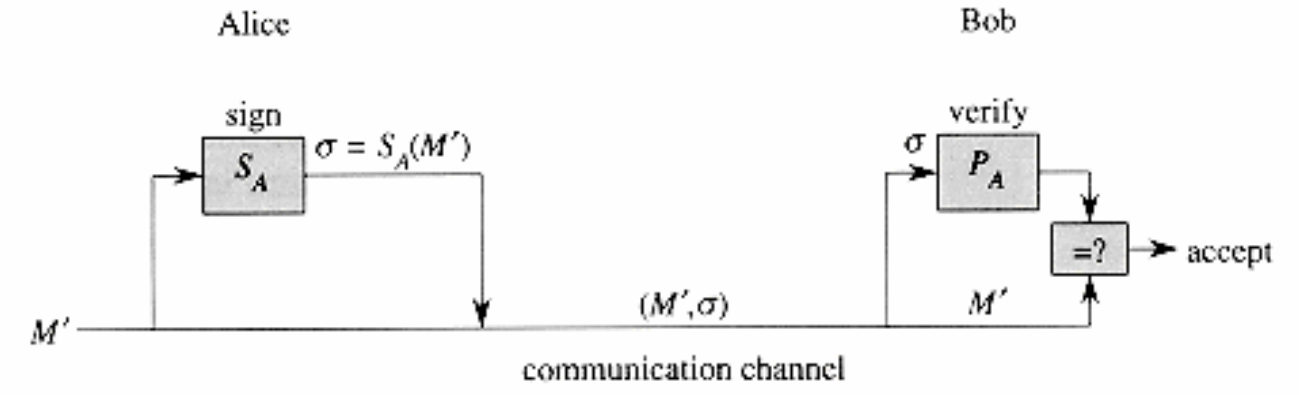
1. Bob pobiera klucz jawny Alicji P_A.
2. Bob oblicza tekst zaszyfrowany $C=P_A(M)$ odpowiadający wiadomości M i przesyła go Alicji.
3. Alicja po otrzymaniu zaszyfrowanego tekstu C, stosuje do niego swój klucz tajny, otrzymując wiadomość pierwotną $M=S_A(C)$.



Podpis cyfrowy

Załóżmy, że Alicja chce wysłać Bobowi podpisaną cyfrowo wiadomość M'.

1. Alicja oblicza swój podpis cyfrowy $\sigma = S_A(M')$ dla wiadomości M'.
2. Alicja wysyła do Boba parę wiadomość/podpis (M', σ).
3. Bob, po otrzymaniu pary (M', σ) może sprawdzić, czy naprawdę pochodzi ona od Alicji, korzystając z klucza jawnego Alicji i sprawdzając równość $M' = P_A(\sigma)$ (założyliśmy, że Bob będzie wiedział jakiego klucza użyć – np. wiadomość M' jest podpisana). Jeżeli równość zachodzi, wiadomość pochodzi od Alicji, w przeciwnym wypadku została wysłana przez kogoś innego, albo też wiadomość lub podpis zostały uszkodzone przy transmisji.



Jak widać, cyfrowy podpis zapewnia zarówno potwierdzenie tożsamości nadawcy jak i autentyczność treści wiadomości.
Inną własnością podpisu cyfrowego jest to, że jego autentyczność może być zweryfikowana przez dowolną osobę, która ma dostęp do klucza jawnego osoby podpisującej.

Przykład: wiadomość może być elektronicznym czekiem Alicji wystawionym na Boba. Bob może sprawdzić autentyczność czeku i np. przekazać go do banku, który ponownie go sprawdzi i dokona przelewu. Bob nie będzie miał możliwości np. zmiany kwoty na czeku, bo przestanie się zgadzać podpis cyfrowy.

W przykładzie podpisana wiadomość była nie zaszyfrowana.

<u>Procedura wysyłania zaszyfrowanej wiadomości z podpisem:</u>

Podpisujący najpierw dołącza swój podpis cyfrowy do wiadomości a następnie szyfruje parę wiadomość/podpis kluczem jawnym adresata.
Odbiorca odszyfrowuje wiadomość / podpis stosując swój klucz tajny, następnie weryfikuje podpis kluczem jawnym nadawcy.

System kryptograficzny RSA

Użytkownicy systemu tworzą klucze jawne i tajne w następujący sposób:

1. Wybierz losowo dwie duże (np. 1024 lub 2048-bitowe) liczby pierwsze `p` i `q`.

2. Oblicz `n = pq`.

3. Wybierz niewielką nieparzystą liczbę `e`, względnie pierwszą (nie mającą wspólnego dzielnika) z wartością `t=(p-1)(q-1)`.

4. Oblicz `d` będące odwrotnością `e` modulo `t`. Na mocy odpowiednich twierdzeń `d` istnieje i jest wyznaczone jednoznacznie.

Para `P=(e,n)` stanowi klucz jawny RSA.
Para `S=(d,n)` stanowi klucz tajny RSA.

Przekształceniem wiadomości M odpowiadającemu kluczowi jawnemu $P=(e,n)$ jest:

$$P(M)=M^e(\mathrm{mod}\ n)$$

Przekształceniem tekstu zaszyfrowanego C odpowiadającego kluczowi tajnemu $S=(d,n)$ jest:

$$S(C)=C^d(\mathrm{mod}\ n)$$

Przekształcenia te są wzajemnie odwrotne.

Bezpieczeństwo systemu RSA opiera się na trudności rozkładu na czynniki dużych liczb. Jeżeli osoba podsłuchująca potrafiłaby to zrobić (tzn. liczbę n rozłożyć na iloczyn pq), system zostałby złamany.

Przykład (dla małych liczb pierwszych)

1. Wybieramy $p=53$, $q=71$.

2. Wyliczamy iloczyn $n = pq = 3763$.

3. Wyliczamy $t=(p-1)(q-1)=52*70=3640$.

4. Wybieramy e takie, żeby było względnie pierwsze z t. Jest to np. liczba $e=3$.

5. Obliczamy element odwrotny do $e=3$ modulo $t=3640$, tzn taki, żeby $ed = 1(\mathrm{mod}\ t)$. Wynosi on $d=2427$.
Sprawdzamy: $3*2427(\mathrm{mod}\ 3640) = 7281(\mathrm{mod}\ 3640) = 1$

Otrzymaliśmy klucz publiczny $(e,n)=(3,3763)$ oraz klucz prywatny $(d,n)=(2427,3763)$.

Szyfrujemy wiadomość, która jest np. liczbą $M=1000$.

$C = M^e (\mathrm{mod}\ n) = 1000^3 (\mathrm{mod}\ 3763) = \mathbf{1565}$

Odszyfrowujemy wiadomość:

$M = C^d (\mathrm{mod}\ n) = 1565^{2427} (\mathrm{mod}\ 3640) = \mathbf{1000}$

Długość szyfrowanego tekstu nie może przekraczać n. Długie wiadomości można więc dzielić na mniejsze lub szyfrować sam klucz do wiadomości.

Dodatkowo, gdy oryginalna wiadomość podniesiona do potęgi e (a e jest małe) nie przekroczy n, to nie zostanie ona zaszyfrowana, gdyż nie nastąpi skrócenie modulo.

Przykład: chcemy zaszyfrować $M=12$.

$C = M^e (\mathrm{mod}\ n) = 12^3 (\mathrm{mod}\ 3763) = \mathbf{1728}$

Ponieważ $e=3$ jest znane (należy do klucza publicznego), można wyliczyć pierwiastek trzeciego stopnia z C i odzyskać M.

Sprawdzanie, czy liczba n jest pierwsza

1. Algorytm siłowy

Dzielimy do skutku sprawdzaną liczbę n przez kolejne liczby 2, 3, ..., √n. Liczba n jest liczbą pierwszą, jeżeli nie dzieli się całkowicie przez każdą z nich.
Czas – wykładniczy w stosunku do długości liczby n. Większość innych algorytmów wymaga czasu wykładniczego.

2. Probabilistyczny test Millera-Rabina

Algorytm sprawdza na podstawie twierdzenia Fermata, czy liczba jest kandydatem na liczbę pierwszą, wykonując odpowiednio dobrane testy. Dla wykonanych s-testów, prawdopodobieństwo błędu nie przekracza 2^{s-1}. Przy odpowiednio dużej liczbie testów, prawdopodobieństwo pomyłki jest niewielkie.

3. Algorytm Agrawala, Kayala, i Saxena (2004 r.)

Bezwarunkowy, deterministyczny test, działający w czasie wielomianowym. Oparty na uogólnionym małym twierdzeniu Fermata.

Jeżeli liczby a i $n \geq 2$ są względnie pierwsze, to n jest liczbą pierwszą wtedy i tylko wtedy, gdy

$$(X + a)^n = X^n + a \pmod{n}$$

Źródło: Annals of Mathematics, 160 (2004), 781–793
http://www.math.princeton.edu/~annals/issues/2004/Sept2004/Agrawal.pdf

Rozkład na czynniki (faktoryzacja) liczby n.

1. Heurystyka „ro” Pollarda

Dzieląc do skutku przez wszystkie liczby aż do B, rozłożymy całkowicie na czynniki każdą liczbę aż do B^2.
Metoda efektywna dla szukania małych czynników. Potrafi znaleźć czynnik p w średnim czasie proporcjonalnym do $\sqrt{p}$. Dla dużych czynników jej czas działania mocno się wydłuża, aż do wykładniczego.

2. Algorytm Shora (1994 r.)

Algorytm możliwy do zaimplementowania na komputerze kwantowym. Czas działania proporcjonalny do $\log^3 n$, a więc algorytm jest bardzo szybki. Udało się go zaimplementować na prostym komputerze kwantowym i rozłożyć za jego pomocą liczbę 15 na czynniki.