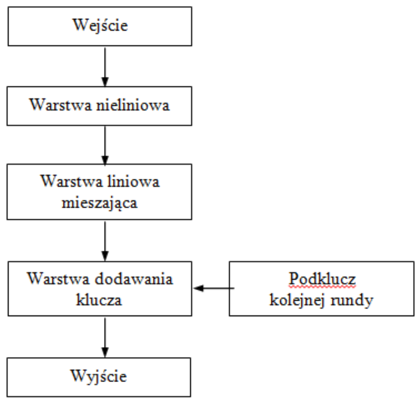
1. usługi kryptograficzne
    - poufność - utrzymywanie informacji w tajemnicy, dla wszystkich poza uprawnionymi (np. AES)
    - integralność – zapewnienie niezmienności informacji w nieuprawniony, lub nieznany sposób (Funkcje skrótu SHA1)
    - Uwierzytelnianie – potwierdzenie tożsamości strony / źródła informacji / oryginalności danych (podpis cyfrowy)
2. AES
    - *Advanced Encryption Standard* - standard symetrycznego szyfru blokowego używany przez rząd U.S.A.
    - Możliwe jest w nim użycie kluczy o długościach 128, 192 i 256 bitów i operuje on na blokach danych o długości 128 bitów
    - AES wykonuje 10 (klucz 128 bitów), 12 lub 14 rund szyfrujących

    <u>DZIAŁANIE:</u>



Rys. 4.1. Schemat blokowy jednej rundy.

- Tekst jawny dzielony na tablice 4x4 kazda komórka tabeli -> 1 bajt
-
- Poszczególne postaci tekstu jawnego po zastosowaniu kolejnych transformacji algorytmu – <u>stany</u>
- przed wykonaniem pierwszej rundy algorytmu wykonujemy operację dodania klucza do bloku wejściowego, -> runda zerowa.
- Rundy:
    - *ByteStub* - Jest to jedyna nieliniowa transformacja stosowana w algorytmie. Działa ona niezależnie na każdy bajt stanu i ma postać skrzynki podstawieniowo – przestawieniowej o wymiarach 8×8.
    - *ShiftRow – przesuniecie wiersza tabeli (pierwszego o 0, drugiego o 1, trzeciego o 2 czwartego, 3), ostatni elemnt wiersza na 1 pozycję*
    - *MixColumn - W transformacji tej kolumny danego stanu rozpatrywane są jako wielomiany (elementy kolumny są współczynnikami wielomia-nu) i mnożone są przez ustalony wielomian*

$$b(x) = c(x) \otimes a(x),$$

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

    - *AddRoundKey* - Dla każdej kolejnej rundy generowany jest podklucz o tej długości co długość przekształcanych stanów. Na koniec każdej rundy podklucz ten jest dodawany (operacja EXOR)

$$\begin{array}{|c|c|c|c|}\hline a_{00} & a_{01} & a_{02} & a_{03} \\ \hline a_{10} & a_{11} & a_{12} & a_{13} \\ \hline a_{20} & a_{21} & a_{22} & a_{23} \\ \hline a_{30} & a_{31} & a_{32} & a_{33} \\ \hline \end{array} \oplus \begin{array}{|c|c|c|c|}\hline k_{00} & k_{01} & k_{02} & k_{03} \\ \hline k_{10} & k_{11} & k_{12} & k_{13} \\ \hline k_{20} & k_{21} & k_{22} & k_{23} \\ \hline k_{30} & k_{31} & k_{32} & k_{33} \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline b_{00} & b_{01} & b_{02} & b_{03} \\ \hline b_{10} & b_{11} & b_{12} & b_{13} \\ \hline b_{20} & b_{21} & b_{22} & b_{23} \\ \hline b_{30} & b_{31} & b_{32} & b_{33} \\ \hline \end{array}$$

Przekształcenia *ShiftRow* (przesunięcie wierszy) i *MixColumn* (mieszanie kolumn) stanowią warstwę liniową mieszającą

- W ten sposób algorytm Rijndael składa się z 11 rund numerowanych od 0 do 10; rundy od 1 do 9 są identyczne. Natomiast runda 10 nie zawiera transformacji *MixColumn*. Do każdej rundy potrzebny jest 128 bitowy podklucz
- podkluczye, generowane są przy pomocy oddzielnego algorytmu (ang. *key schedule*) na podstawie 128-bitowego klucza głównego stosowanego w algorytmie.

3. Protokół Diffie-Hellmana.
   - Protokół Diffie-Hellmana.

7.2.1. ALGORYTM DIFFIEGO-HELLMANA

- oparty na trudności obliczania logarytmów dyskretnych w ciałach skończonych;
- wykorzystywany do dystrybucji kluczy (nie do szyfrowania i deszyfrowania).

Alicja i Bob uzgadniają ze sobą w sposób jawny wybór dwóch dużych liczb całkowitych p i g, $1 < g < p$.

Warunki bezpieczeństwa:

- p i $(p-1)/2$ - liczby pierwsze długości 512 (lub lepiej 1024) bitów;
- liczba g - pierwiastek pierwotny modulo p.

1. Alicja wybiera losowo dużą liczbę całkowitą x (tajną) i oblicza:
$$X = g^x \bmod p.$$
2. Bob wybiera losowo dużą liczbę całkowitą y (tajną) i oblicza:
$$Y = g^y \bmod p.$$
3. Alicja wysyła X do Boba, a Bob wysyła Y do Alicji.
4. Alicja oblicza $k = Y^x \bmod p$.
5. Bob oblicza $k' = X^y \bmod p$.
$$k = g^{xy} \bmod p = k'.$$

Podsłuchujący zna tylko p, g, X, Y.

Aby znaleźć k musi wyznaczyć x lub y obliczając logarytm dyskretny
$$x = \log_g X \quad \text{lub} \quad y = \log_g Y.$$

Bezpieczeństwo protokołu D-H

Przeciwnik zna parametry p, g oraz przesyłane liczby A, B, nie zna natomiast liczb a, b na podstawie których mógłby obliczyć liczbę k. W tym celu musi obliczyć jeden z logarytmów dyskretnych $log_g(A)$ lub $log_b(B)$ w grupie Z_p^*, jeśli p jest odpowiednio duże jest to zadanie niewykonalne w realnym czasie. Bezpieczeństwo protokołu Diffie-Hellmana opiera się na trudnym obliczeniowo problemie logarytmu dyskretnego. Aktualnie zalecaną wielkością p jest minimum 1024 bity, czyli około 10^{300}.

- Najpierw p potem ge (xD)

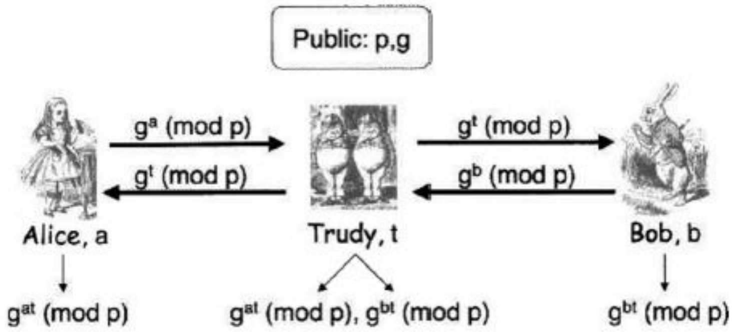
4. Atak metodą 'człowiek w środku'.
    - **Ogólnie ocb**

<u>Atak na protokół D-H metodą Man-in-the-Middle</u>

Załóżmy, że Trudy, która jest aktywnym przeciwnikiem, chce czytać tajną pocztę przesyłaną między Alicją i Bolkiem.

Strategia Trudy jest taka, że w czasie wykonywania protokołu D-H przez Alicję i Bolka przejmuje ona wysyłane przez nich liczby A i B, a następnie w ich miejsce podstawia odpowiednie inne, swoje wielkości.

Alicja i Bolek są nieświadomi tego, co dzieje się między nimi i sądzą, że ustalili wspólny, tajny klucz.



Atak typu Man-in-the-Middle na protokół Diffie-Hellmana

1. Trudy przejmuje wartości A i B wysłane przez Alicję i Bolka.
2. Trudy wybiera wykładnik t, oblicza $T = g^t$ i wysyła T do Alicji i Bolka.
3. Alicja wierzy, że T pochodzi od Bolka, zaś Bolek wierzy, że T pochodzi od Alicji.
4. Trudy oblicza $k_A = A^t = g^{at}$ oraz $k_B = B^t = g^{bt}$.
5. Alicja, nie wiedząc, że Trudy jest w środku, postępując zgodnie z protokołem D-H oblicza k_A.
6. Podobnie Bolek oblicza k_B.
7. Teraz, gdy Alicja wysyła wiadomość do Bolka (zaszyfrowaną kluczem k_A), deszyfruje ją i szyfruje ponownie (lub szyfruje inną wiadomość) kluczem k_B przed wysłaniem jej do Bolka.
8. W ten sposób Trudy może czytać (i jeśli chce zmieniać) wiadomości przesyłane między Alicją a Bolkiem, którzy są nieświadomi tego, co sie dzieje.
9. Atak „człowiek w środku” może być udaremniony przez odpowiednie uwierzytelnienie obu stron protokołu D-H, na przykład przez zastosowanie podpisu cyfrowego.

5. Kryptosystemy klucza publicznego.

W dotychczas rozważanych symetrycznych kryptosystemach znajomość przekształcenia szyfrującego e_k, przy ustalonym kluczu $k \in K$, równoważna była znajomości przekształcenia deszyfrującego d_k. Strony porozumiewające się A i B muszą dokonać wcześniej poufnej wymiany klucza.

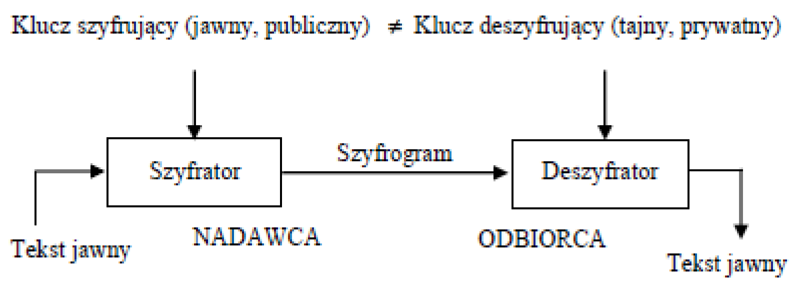
Podstawową ideą kryptografii klucza publicznego jest założenie, że obliczenie d_k na podstawie znajomości e_k jest problemem trudnym obliczeniowo. Wtedy znajomość e_k można uczynić jawną podając ją w „książce telefonicznej" zawierającej tego typu „adresy" poszczególnych uczestników kryptosystemu. Jeśli Alicja chce wysłać zaszyfrowaną wiadomość do Boba to wybiera z książki adresowej „klucz publiczny" Boba i szyfruje nim wiadomość przeznaczoną dla niego. Bob po otrzymaniu tej wiadomości deszyfruje ją używając swojego „klucza prywatnego". W tego typu asymetrycznych kryptosystemach nie jest konieczna poufna wymiana kluczy.

- Koncepcja, Oznaczenia, jak wygląda

Idea kryptosystemu klucza publicznego - 1976 r.
Whitfield DIFFIE , Martin HELLMAN , Ralph MERKLE

Pierwsza realizacja asymetrycznych kryptosystemów należy do Rivesta, Shamira i Adlemana w roku 1977. Kryptosystem RSA jest najpowszechniej stosowanym systemem klucza publicznego; stosowany jest między innymi w protokołach kryptograficznych wymiany kluczy i podpisach cyfrowych.

Od tego czasu zaproponowano szereg kryptosystemów klucza publicznego opartych na problemach trudnych obliczeniowo; na problemie pakowania plecaka, na algebraicznej teorii kodowania i wielu wersjach problemu obliczania logarytmu dyskretnego w ciałach skończonych i na krzywych eliptycznych. Podstawową uwagą jest, że kryptosystemy klucza publicznego nie zapewniają bezwarunkowego bezpieczeństwa. Ich bezpieczeństwo oparte jest na aktualnej wiedzy teoretycznej i możliwościach technologicznych dotyczących rozwiązania danego problemu obliczeniowego. Jeśli przestrzeń wiadomości jawnych nie jest zbyt duża to bezpośrednią metodą przeprowadzenia ataku ze znanym szyfrogramem y jest sprawdzanie wiadomości jawnych $x \in P$, aż znajdziemy taką, że $y = e_k(x)$; klucz publiczny k, a zatem przekształcenie szyfrujące e_k są znane.



Rys. 7.1

Każdy uczestnik wymiany informacji wytwarza dwa klucze:

- klucz szyfrujący (jawny),
  który udostępnia publicznie wszystkim pozostałym uczestnikom;
- klucz deszyfrujący (tajny),
  który utrzymuje w ścisłej tajemnicy.

Każdy może wysłać wiadomość, ale tylko odbiorca może ją odszyfrować.

Wartości funkcji szyfrującej e_k gdy jej argumentem są wiadomości jawne $x \in P$ powinny być łatwe do obliczenia. Natomiast obliczenie funkcji odwrotnej; tzn. obliczenie x gdy znany jest $y = e_k(x)$ musi być problemem trudnym obliczeniowo. Funkcje o tej własności nazywamy jednokierunkowymi. Funkcje jednokierunkowe odgrywają podstawową rolę w kryptografii z kluczem publicznym. Istnieje wiele funkcji o których „wierzy się", że są jednokierunkowe, natomiast nie udowodniono teoretycznie, że istnieje choć jedna funkcja jednokierunkowa. Podstawowym przykładem jest funkcja służąca do konstrukcji algorytmu RSA.

Niech $n = pq$, gdzie p i q są odpowiednio dużymi liczbami pierwszymi. Rozważmy funkcję:

$$f : \mathbf{Z}_n \to \mathbf{Z}_n$$

daną wzorem

$$f(x) = x^b \bmod n,$$

gdzie wykładnik b jest odpowiednio dobrany względem n; f jest funkcją szyfrującą w systemie RSA. Funkcja f jest używana przy szyfrowaniu wiadomości wysyłanych do określonego odbiorcy, np. Boba. Z kolei Bob powinien mieć możliwość szybkiego odszyfrowania odebranej wiadomości, czyli szybkiego odwrócenia funkcji f.

W tym celu stosuje się tzw. funkcje jednokierunkowe z zapadką, są to funkcje jednokierunkowe, które można szybko odwrócić przy posiadaniu dodatkowej informacji; jest to właśnie „klucz tajny" Boba służący do deszyfrowania.

6. Kryptosystem RSA. Problem faktoryzacji.
    - **Konkretny algorytm przedstawić wzorami**

Podstawy matematyczne

Twierdzenie (Euklides)

Jest nieskończenie wiele liczb pierwszych

$$p_1 < p_2 < \ \dots$$

Twierdzenie to nie podaje jak wyrażają się kolejne liczby pierwsze.

Istnieją tzw. testy pierwszości pozwalające sprawdzać, czy dana liczba jest pierwsza; mają one złożoność wielomianową,

tak że nie jest trudno generować kilkuset-cyfrowe liczby pierwsze.

Twierdzenie (Euklides) Podstawowe Twierdzenie Arytmetyki

Każdą liczbę naturalną $n > 1$ można jednoznacznie przedstawić jako iloczyn liczb pierwszych

$$n = p_1^{k_1} p_2^{k_2} \dots p_s^{k_s},$$

gdzie liczby pierwsze $p_1 < \cdots < p_s$ są jednoznacznie określone, zaś $k_1 \geq 1, \dots, k_s \geq 1$ są jednoznacznie określonymi potęgami naturalnymi.

Dowód tego twierdzenia jest nie-konstruktywny; nie znamy efektywnej metody (tzn. o złożoności wielomianowej) rozkładu (faktoryzacji) liczb naturalnych na czynniki pierwsze.

Konstrukcja kryptosystemu RSA

Generacja kluczy

1. Strona **A** wybiera dwie, odpowiednio duże liczby pierwsze p, q.
2. **A** oblicza iloczyn $n = pq$ oraz
$$\varphi = (p-1)(q-1).$$
3. **A** wybiera losowo liczbę e taką że $1 < e < \varphi$ oraz $NWD(\varphi, e) = 1$.
   Para $e_A = (n, e)$ stanowi klucz prywatny strony **A**.
4. **A** oblicza $d = e^{-1} mod \varphi$, tzn. $ed \equiv 1 mod \varphi$, d jest odwrotnością modularną e modulo φ.
   Para $d_A = (n, d)$ jest kluczem prywatnym strony **A**, d jest tajne.
5. Liczby p, q, φ są kasowane.

Etap poufnej wymiany wiadomości.

M – wiadomość, liczba $0 \leq M < n$

Szyfrowanie: $C = M^e mod\ n$,

wykonuje nadawca **B** wiadomości do **A**.

Deszyfrowanie: $C^d mod\ n = M$,

wykonuje odbiorca **A** szyfrogramu C.

- Bezpieczeństwo ma trudność obliczeniową **faktoryzacji**

7.1.2. BEZPIECZEŃSTWO RSA

Bezpieczeństwo algorytmu RSA zależy od długości (wielkości) modułu n oraz mocy obliczeniowych współczesnych komputerów i ich możliwości faktoryzacji modułu n. Przy czym moc obliczeniowa maszyn cyfrowych zależy oczywiście od poniesionych nakładów. Poniżej przedstawiono dane obrazujące współczesne możliwości faktoryzacji dużych liczb złożonych.

Bezpieczne długości modułu n:

664 bity (200 cyfr), 1024 bity (308 cyfr), 2048 bitów (616 cyfr).

7. Usługa integralności. Funkcje skrótu.
   - Integralność -> zapewnienie integralności

A wysyła do **B** wiadomość M wraz z jej skrótem (cyfrowym odciskiem – *message digist*) $h(M)$.

B po otrzymaniu wiadomości M' (zakładamy, że M w czasie transmisji mogła ulec zmianie) oblicza skrót $h(M')$.
Następnie sprawdza, czy

$h(M') = h(M)$. Jeśli TAK, to wnioskuje że $M' = M$, jeśli NIE, to wnioskuje że $M' \neq M$, tzn. wiadomość M uległa zmianie w czasie jej transmisji. Wnioskowanie oparte jest na własnościach zastosowanej funkcji skrótu h , która służy do sprawdzania integralności danych.

Jednokierunkowe funkcje skrótu

$$h : \{0,1\}^* \rightarrow \{0,1\}^n$$

$\{0,1\}^*$ - ciągi bitowe dowolnej, skończonej długości (zbiór wszystkich wiadomości).

$\{0,1\}^n$ - ciągi bitowe o długości n bitów, n jest długością skrótu.

W praktyce n = 128, 160, 256, 512 bitów.

- Warunki na funkcje skrótu

Warunki nakładane na funkcje skrótu

1. Obliczanie skrótu $h(x)$ dla $x \in X = \{0,1\}^*$ jest efektywne (złożoność wielomianowa).
2. Dla $y \in Y = \{0,1\}^n$ trudne obliczeniowo jest znalezienie $x \in X$, takiego że $h(x) = y$, (złożoność $O(2^n)$).
3. Dla $x \in X$ trudne obliczeniowo jest znalezienie $x' \in X, x \neq x'$, takiego że $h(x) = h(x')$, (złożoność $O(2^n)$).
4. Trudno obliczeniowo jest znaleźć $x, x' \epsilon X, x \neq x'$, takie że $h(x) = h(x')$ - tzw. odporność na kolizje, (złożoność $O(2^{\frac{n}{2}})$ - na podstawie paradoksu dnia urodzin).

Funkcję skrótu uważamy za „złamaną”, jeśli w jednym z punktów 2 , 3 lub 4 , wykorzystując budowę wewnętrzną funkcji skrótu, uzyskamy złożoność mniejszą niż wskazana, wynikająca z warunków ogólnych.

Stosowane jednokierunkowe funkcje skrótu

a) MD4 – skrót 128 bitów, złamana, aktualnie nie stosowana.
b) MD5 – skrót 128 bitów, złamana ze względu na znajdowanie kolizji, była powszechnie stosowana w protokole internetowym *SSL*.
c) SHA-1 – skrót 160 bitów, standard USA, występuje w standardzie podpisu cyfrowego, atak teoretyczny na znajdowanie kolizji o złożoności $O(2^{64})$.

d) Ogłoszony konkurs (przez *NIST* – Nationai Institute of Standards and Technology w USA) na opracowanie nowych funkcji skrótu.

Warunki bezpieczeństwa 2 i 3 są wystarczające w zastosowaniu funkcji skrótu w usłudze integralności.

Warunek 4 (odporność funkcji skrótu na kolizje) jest ważna w zastosowaniu tych funkcji w schemacie podpisu cyfrowego, aby strona generująca podpis cyfrowy dokumentu nie mogła później podstawić dokumentu o innej treści, ale mającego ten sam skrót.

8. Podpis Cyfrowy

Schemat ogólny: