Kryptografia

z elementami kryptografii kwantowej

Ryszard Tanaś

http://zon8.physd.amu.edu.pl/~tanas

Wykład 11

Spis treści

16 Zarządzanie kluczami

16.1 Generowanie kluczy

- Do generowania kluczy najlepiej nadają się generatory ciągów losowych.
- Przykład: standard ANSI X9.17
  (Financial Institution Key Management)
  Niech $EDE_{K_1,K_2}(X)$ oznacza szyfrowanie 3-DES z kluczami K_1, K_2 liczby X. Niech V_0 będzie tajną liczbą 64 bitową, a T znacznikiem czasu, wtedy klucz losowy R_i generuje się w następujący sposób:
  $R_i = EDE_{K_1,K_2}(EDE_{K_1,K_2}(T_i) \oplus V_i)$
  $V_{i+1} = EDE_{K_1,K_2}(EDE_{K_1,K_2}(T_i) \oplus R_i)$

16 Zarządzanie kluczami

16.1 Generowanie kluczy

- Do generowania kluczy najlepiej nadają się generatory ciągów losowych.
- Przykład: standard ANSI X9.17
  (Financial Institution Key Management)
  Niech $EDE_{K_1,K_2}(X)$ oznacza szyfrowanie 3-DES z kluczami K_1, K_2 liczby X. Niech V_0 będzie tajną liczbą 64 bitową, a T znacznikiem czasu, wtedy klucz losowy R_i generuje się w następujący sposób:
  $R_i = EDE_{K_1,K_2}(EDE_{K_1,K_2}(T_i) \oplus V_i)$
  $V_{i+1} = EDE_{K_1,K_2}(EDE_{K_1,K_2}(T_i) \oplus R_i)$

16 Zarządzanie kluczami

16.1 Generowanie kluczy

- Do generowania kluczy najlepiej nadają się generatory ciągów losowych.
- Przykład: standard ANSI X9.17
  (Financial Institution Key Management)
  Niech $EDE_{K_1,K_2}(X)$ oznacza szyfrowanie 3-DES z kluczami K_1, K_2 liczby X. Niech V_0 będzie tajną liczbą 64 bitową, a T znacznikiem czasu, wtedy klucz losowy R_i generuje się w następujący sposób:
  $R_i = EDE_{K_1,K_2}(EDE_{K_1,K_2}(T_i) \oplus V_i)$
  $V_{i+1} = EDE_{K_1,K_2}(EDE_{K_1,K_2}(T_i) \oplus R_i)$

16.2 Przesyłanie kluczy

- Jeśli Alicja i Bolek zamierzają posługiwać się symetrycznym algorytmem kryptograficznym, to potrzebują tego samego klucza.
- Alicja może wygenerować taki klucz używając generatora ciągów losowych, ale pozostaje problem jak w bezpieczny sposób przekazać ten klucz Bolkowi.

16.2 Przesyłanie kluczy

- Jeśli Alicja i Bolek zamierzają posługiwać się symetrycznym algorytmem kryptograficznym, to potrzebują tego samego klucza.
- Alicja może wygenerować taki klucz używając generatora ciągów losowych, ale pozostaje problem jak w bezpieczny sposób przekazać ten klucz Bolkowi.

16.2 Przesyłanie kluczy

- Jeśli Alicja i Bolek zamierzają posługiwać się symetrycznym algorytmem kryptograficznym, to potrzebują tego samego klucza.
- Alicja może wygenerować taki klucz używając generatora ciągów losowych, ale pozostaje problem jak w bezpieczny sposób przekazać ten klucz Bolkowi.

16.3 Przechowywanie kluczy

- Najbezpieczniejszym sposobem przechowywania klucza jest zapamiętanie go przez Alicję. Niestety sposób ten ma tę wadę, że Alicja może zapomnieć taki klucz.
- Klucze mogą być przechowywane w pamięci ROM. Klucz taki może być rozdzielony na połowy, z których jedna jest przechowywana w terminalu a druga w pamięci ROM.
- W wielu sytuacjach istnieje konieczność przechowywania kopii zapasowych klucza. Do tego celu wykorzystuje się np. karty inteligentne.
- Klucze na ogół mają strukturę hierarchiczną

16.3 Przechowywanie kluczy

- Najbezpieczniejszym sposobem przechowywania klucza jest zapamiętanie go przez Alicję. Niestety sposób ten ma tę wadę, że Alicja może zapomnieć taki klucz.
- Klucze mogą być przechowywane w pamięci ROM. Klucz taki może być rozdzielony na połowy, z których jedna jest przechowywana w terminalu a druga w pamięci ROM.
- W wielu sytuacjach istnieje konieczność przechowywania kopii zapasowych klucza. Do tego celu wykorzystuje się np. karty inteligentne.
- Klucze na ogół mają strukturę hierarchiczną

16.3 Przechowywanie kluczy

- Najbezpieczniejszym sposobem przechowywania klucza jest zapamiętanie go przez Alicję. Niestety sposób ten ma tę wadę, że Alicja może zapomnieć taki klucz.
- Klucze mogą być przechowywane w pamięci ROM. Klucz taki może być rozdzielony na połowy, z których jedna jest przechowywana w terminalu a druga w pamięci ROM.
- W wielu sytuacjach istnieje konieczność przechowywania kopii zapasowych klucza. Do tego celu wykorzystuje się np. karty inteligentne.
- Klucze na ogół mają strukturę hierarchiczną

16.3 Przechowywanie kluczy

- Najbezpieczniejszym sposobem przechowywania klucza jest zapamiętanie go przez Alicję. Niestety sposób ten ma tę wadę, że Alicja może zapomnieć taki klucz.
- Klucze mogą być przechowywane w pamięci ROM. Klucz taki może być rozdzielony na połowy, z których jedna jest przechowywana w terminalu a druga w pamięci ROM.
- W wielu sytuacjach istnieje konieczność przechowywania kopii zapasowych klucza. Do tego celu wykorzystuje się np. karty inteligentne.
- Klucze na ogół mają strukturę hierarchiczną

16.3 Przechowywanie kluczy

- Najbezpieczniejszym sposobem przechowywania klucza jest zapamiętanie go przez Alicję. Niestety sposób ten ma tę wadę, że Alicja może zapomnieć taki klucz.
- Klucze mogą być przechowywane w pamięci ROM. Klucz taki może być rozdzielony na połowy, z których jedna jest przechowywana w terminalu a druga w pamięci ROM.
- W wielu sytuacjach istnieje konieczność przechowywania kopii zapasowych klucza. Do tego celu wykorzystuje się np. karty inteligentne.
- Klucze na ogół mają strukturę hierarchiczną

- master keys

  klucze znajdujące się najwyżej w hierarchii, takie klucze nigdy nie są zmieniane. Taki klucz jest zwykle tylko zapamiętywany przez użytkownika lub zapisany w urządzeniu kryptograficznym. Może on być częściowo zapisany na kartach inteligentnych a częściowo zapamiętywany przez użytkownika. Master key służy do zabezpieczania innych kluczy.

- klucze do szyfrowania kluczy (keys-encrypting keys)

  klucze wykorzystywane w protokołach do uzgadniania (przesyłania) kluczy.

- master keys

  klucze znajdujące się najwyżej w hierarchii, takie klucze nigdy nie są zmieniane. Taki klucz jest zwykle tylko zapamiętywany przez użytkownika lub zapisany w urządzeniu kryptograficznym. Może on być częściowo zapisany na kartach inteligentnych a częściowo zapamiętywany przez użytkownika. Master key służy do zabezpieczania innych kluczy.

- klucze do szyfrowania kluczy (keys-encrypting keys)

  klucze wykorzystywane w protokołach do uzgadniania (przesyłania) kluczy.

- klucze do szyfrowania danych (data keys)

  są to zwykle klucze o krótkim czasie ważności (np. klucze sesyjne) służące do szyfrowania danych

17 Uzgadnianie kluczy

17.1 Algorytm Diffiego-Hellmana

- Alicja i Bolek uzgadniają dwie liczby: dużą liczbę pierwszą p oraz liczbę g, która jest generatorem $\mathbb{Z}_p$
- Alicja wybiera losowo dużą liczbę całkowitą $a < p - 1$ i przesyła Bolkowi $x = g^a \pmod p$
- Bolek wybiera losowo dużą liczbę całkowitą $b < p - 1$ i wysyła Alicji $y = g^b \pmod p$
- Alicja oblicza $K = y^a \pmod p$
- Bolek oblicza $K = x^b \pmod p$
- Uzasadnienie:
  $K = y^a = (g^b)^a = g^{ab} \pmod p$
  $K = x^b = (g^a)^b = g^{ab} \pmod p$

17 Uzgadnianie kluczy

17.1 Algorytm Diffiego-Hellmana

- Alicja i Bolek uzgadniają dwie liczby: dużą liczbę pierwszą p oraz liczbę g, która jest generatorem $\mathbb{Z}_p$
- Alicja wybiera losowo dużą liczbę całkowitą $a < p - 1$ i przesyła Bolkowi $x = g^a \pmod{p}$
- Bolek wybiera losowo dużą liczbę całkowitą $b < p - 1$ i wysyła Alicji $y = g^b \pmod{p}$
- Alicja oblicza $K = y^a \pmod{p}$
- Bolek oblicza $K = x^b \pmod{p}$
- Uzasadnienie:
  $K = y^a = (g^b)^a = g^{ab} \pmod{p}$
  $K = x^b = (g^a)^b = g^{ab} \pmod{p}$

17 Uzgadnianie kluczy

17.1 Algorytm Diffiego-Hellmana

- Alicja i Bolek uzgadniają dwie liczby: dużą liczbę pierwszą p oraz liczbę g, która jest generatorem $\mathbb{Z}_p$
- Alicja wybiera losowo dużą liczbę całkowitą $a < p - 1$ i przesyła Bolkowi $x = g^a \pmod{p}$
- Bolek wybiera losowo dużą liczbę całkowitą $b < p - 1$ i wysyła Alicji $y = g^b \pmod{p}$
- Alicja oblicza $K = y^a \pmod{p}$
- Bolek oblicza $K = x^b \pmod{p}$
- Uzasadnienie:
  $K = y^a = (g^b)^a = g^{ab} \pmod{p}$
  $K = x^b = (g^a)^b = g^{ab} \pmod{p}$

17 Uzgadnianie kluczy

17.1 Algorytm Diffiego-Hellmana

- Alicja i Bolek uzgadniają dwie liczby: dużą liczbę pierwszą p oraz liczbę g, która jest generatorem $\mathbb{Z}_p$
- Alicja wybiera losowo dużą liczbę całkowitą $a < p - 1$ i przesyła Bolkowi $x = g^a \pmod{p}$
- Bolek wybiera losowo dużą liczbę całkowitą $b < p - 1$ i wysyła Alicji $y = g^b \pmod{p}$
- Alicja oblicza $K = y^a \pmod{p}$
- Bolek oblicza $K = x^b \pmod{p}$
- Uzasadnienie:

  $K = y^a = (g^b)^a = g^{ab} \pmod{p}$

  $K = x^b = (g^a)^b = g^{ab} \pmod{p}$

17 Uzgadnianie kluczy

17.1 Algorytm Diffiego-Hellmana

- Alicja i Bolek uzgadniają dwie liczby: dużą liczbę pierwszą p oraz liczbę g, która jest generatorem $\mathbb{Z}_p$
- Alicja wybiera losowo dużą liczbę całkowitą $a < p - 1$ i przesyła Bolkowi $x = g^a \pmod p$
- Bolek wybiera losowo dużą liczbę całkowitą $b < p - 1$ i wysyła Alicji $y = g^b \pmod p$
- Alicja oblicza $K = y^a \pmod p$
- Bolek oblicza $K = x^b \pmod p$
- Uzasadnienie:

  $K = y^a = (g^b)^a = g^{ab} \pmod p$

  $K = x^b = (g^a)^b = g^{ab} \pmod p$

17 Uzgadnianie kluczy

17.1 Algorytm Diffiego-Hellmana

- Alicja i Bolek uzgadniają dwie liczby: dużą liczbę pierwszą p oraz liczbę g, która jest generatorem $\mathbb{Z}_p$
- Alicja wybiera losowo dużą liczbę całkowitą $a < p - 1$ i przesyła Bolkowi $x = g^a \pmod{p}$
- Bolek wybiera losowo dużą liczbę całkowitą $b < p - 1$ i wysyła Alicji $y = g^b \pmod{p}$
- Alicja oblicza $K = y^a \pmod{p}$
- Bolek oblicza $K = x^b \pmod{p}$
- Uzasadnienie:

  $K = y^a = (g^b)^a = g^{ab} \pmod{p}$

  $K = x^b = (g^a)^b = g^{ab} \pmod{p}$

17 Uzgadnianie kluczy

17.1 Algorytm Diffiego-Hellmana

- Alicja i Bolek uzgadniają dwie liczby: dużą liczbę pierwszą p oraz liczbę g, która jest generatorem $\mathbb{Z}_p$
- Alicja wybiera losowo dużą liczbę całkowitą $a < p - 1$ i przesyła Bolkowi $x = g^a \pmod{p}$
- Bolek wybiera losowo dużą liczbę całkowitą $b < p - 1$ i wysyła Alicji $y = g^b \pmod{p}$
- Alicja oblicza $K = y^a \pmod{p}$
- Bolek oblicza $K = x^b \pmod{p}$
- Uzasadnienie:

  $K = y^a = (g^b)^a = g^{ab} \pmod{p}$

  $K = x^b = (g^a)^b = g^{ab} \pmod{p}$

17.2 Algorytm ElGamala

- Bolek wybiera liczbę pierwszą p oraz generator g w $\mathbb{Z}_p$, a następnie wybiera losowo liczbę $0 < b < p - 1$, oblicza $g^b \pmod p$ oraz ogłasza swój klucz publiczny $\{p, g, g^b\}$
- Alicja pobiera klucz publiczny Bolka, wybiera losowo liczbę $0 < a < p - 1$ i wysyła Bolkowi $g^a \pmod p$ obliczając jednocześnie klucz $K = (g^b)^a \pmod p$
- Bolek oblicza ten sam klucz $K = (g^a)^b \pmod p$

17.2 Algorytm ElGamala

- Bolek wybiera liczbę pierwszą p oraz generator g w $\mathbb{Z}_p$, a następnie wybiera losowo liczbę $0 < b < p - 1$, oblicza $g^b \pmod{p}$ oraz ogłasza swój klucz publiczny $\{p, g, g^b\}$
- Alicja pobiera klucz publiczny Bolka, wybiera losowo liczbę $0 < a < p - 1$ i wysyła Bolkowi $g^a \pmod{p}$ obliczając jednocześnie klucz $K = (g^b)^a \pmod{p}$
- Bolek oblicza ten sam klucz $K = (g^a)^b \pmod{p}$

17.2 Algorytm ElGamala

- Bolek wybiera liczbę pierwszą p oraz generator g w $\mathbb{Z}_p$, a następnie wybiera losowo liczbę $0 < b < p - 1$, oblicza $g^b \pmod{p}$ oraz ogłasza swój klucz publiczny $\{p, g, g^b\}$
- Alicja pobiera klucz publiczny Bolka, wybiera losowo liczbę $0 < a < p - 1$ i wysyła Bolkowi $g^a \pmod{p}$ obliczając jednocześnie klucz $K = (g^b)^a \pmod{p}$
- Bolek oblicza ten sam klucz $K = (g^a)^b \pmod{p}$

17.2 Algorytm ElGamala

- Bolek wybiera liczbę pierwszą p oraz generator g w $\mathbb{Z}_p$, a następnie wybiera losowo liczbę $0 < b < p - 1$, oblicza $g^b \pmod{p}$ oraz ogłasza swój klucz publiczny $\{p, g, g^b\}$
- Alicja pobiera klucz publiczny Bolka, wybiera losowo liczbę $0 < a < p - 1$ i wysyła Bolkowi $g^a \pmod{p}$ obliczając jednocześnie klucz $K = (g^b)^a \pmod{p}$
- Bolek oblicza ten sam klucz $K = (g^a)^b \pmod{p}$

17.3 Station-to Station protocol (STS)

- W tym protokole przyjmuje się, że Alicja ma certyfikat klucza publicznego Bolka i na odwrót Bolek ma certyfikat klucza publicznego Alicji.
- Niech E oznacza symetryczny algorytm szyfrujący, $S_A(M)$ oznacza podpis Alicji pod wiadomością M: $S_A(M) = (H(M))^{d_A} \pmod{n_A}$ (RSA na wartości funkcji hashującej). Każda ze stron wybiera odpowiednią liczbę pierwszą p oraz generator g w $\mathbb{Z}_p$. Każda ze stron wybiera klucze RSA do podpisu.
- Alicja wybiera losowo liczbę a i wysyła Bolkowi $g^a \pmod{p}$
- Bolek wybiera losowo liczbę b i oblicza klucz $K = (g^a)^b \pmod{p}$

17.3 Station-to Station protocol (STS)

- W tym protokole przyjmuje się, że Alicja ma certyfikat klucza publicznego Bolka i na odwrót Bolek ma certyfikat klucza publicznego Alicji.
- Niech E oznacza symetryczny algorytm szyfrujący, $S_A(M)$ oznacza podpis Alicji pod wiadomością M: $S_A(M) = (H(M))^{d_A} \pmod{n_A}$ (RSA na wartości funkcji hashującej). Każda ze stron wybiera odpowiednią liczbę pierwszą p oraz generator g w $\mathbb{Z}_p$. Każda ze stron wybiera klucze RSA do podpisu.
- Alicja wybiera losowo liczbę a i wysyła Bolkowi $g^a \pmod{p}$
- Bolek wybiera losowo liczbę b i oblicza klucz $K = (g^a)^b \pmod{p}$

17.3 Station-to Station protocol (STS)

- W tym protokole przyjmuje się, że Alicja ma certyfikat klucza publicznego Bolka i na odwrót Bolek ma certyfikat klucza publicznego Alicji.
- Niech E oznacza symetryczny algorytm szyfrujący, $S_A(M)$ oznacza podpis Alicji pod wiadomością M: $S_A(M) = (H(M))^{d_A} \pmod{n_A}$ (RSA na wartości funkcji hashującej). Każda ze stron wybiera odpowiednią liczbę pierwszą p oraz generator g w $\mathbb{Z}_p$. Każda ze stron wybiera klucze RSA do podpisu.
- Alicja wybiera losowo liczbę a i wysyła Bolkowi $g^a \pmod{p}$
- Bolek wybiera losowo liczbę b i oblicza klucz $K = (g^a)^b \pmod{p}$

17.3 Station-to Station protocol (STS)

- W tym protokole przyjmuje się, że Alicja ma certyfikat klucza publicznego Bolka i na odwrót Bolek ma certyfikat klucza publicznego Alicji.
- Niech E oznacza symetryczny algorytm szyfrujący, $S_A(M)$ oznacza podpis Alicji pod wiadomością M: $S_A(M) = (H(M))^{d_A} \pmod{n_A}$ (RSA na wartości funkcji hashującej). Każda ze stron wybiera odpowiednią liczbę pierwszą p oraz generator g w $\mathbb{Z}_p$. Każda ze stron wybiera klucze RSA do podpisu.
- Alicja wybiera losowo liczbę a i wysyła Bolkowi $g^a \pmod{p}$
- Bolek wybiera losowo liczbę b i oblicza klucz $K = (g^a)^b \pmod{p}$

17.3 Station-to Station protocol (STS)

- W tym protokole przyjmuje się, że Alicja ma certyfikat klucza publicznego Bolka i na odwrót Bolek ma certyfikat klucza publicznego Alicji.
- Niech E oznacza symetryczny algorytm szyfrujący, $S_A(M)$ oznacza podpis Alicji pod wiadomością M: $S_A(M) = (H(M))^{d_A} \pmod{n_A}$ (RSA na wartości funkcji hashującej). Każda ze stron wybiera odpowiednią liczbę pierwszą p oraz generator g w $\mathbb{Z}_p$. Każda ze stron wybiera klucze RSA do podpisu.
- Alicja wybiera losowo liczbę a i wysyła Bolkowi $g^a \pmod{p}$
- Bolek wybiera losowo liczbę b i oblicza klucz $K = (g^a)^b \pmod{p}$

- Bolek podpisuje konkatenację g^b, g^a, szyfruje podpis kluczem K i wysyła Alicji
  $g^b \pmod p$ oraz $E_K(S_B(g^b, g^a))$
- Alicja oblicza klucz $K = (g^b)^a \pmod p$, deszyfruje otrzymane dane oraz używa klucza publicznego Bolka do sprawdzenia podpisu pod wartością funkcji hashującej z konkatenacji obu eksponensów. Jeśli wartość funkcji hashującej zgadza się z otrzymaną przez nią wartością, Alicja akceptuje klucz K i wysyła Bolkowi
  $E_K(S_A(g^a, g^b))$
- Bolek deszyfruje otrzymaną wiadomość, weryfikuje podpis Alicji i jeśli wynik jest pozytywny, akceptuje klucz K

- Bolek podpisuje konkatenację g^b, g^a, szyfruje podpis kluczem K i wysyła Alicji
  $g^b \pmod p$ oraz $E_K(S_B(g^b, g^a))$
- Alicja oblicza klucz $K = (g^b)^a \pmod p$, deszyfruje otrzymane dane oraz używa klucza publicznego Bolka do sprawdzenia podpisu pod wartością funkcji hashującej z konkatenacji obu eksponensów. Jeśli wartość funkcji hashującej zgadza się z otrzymaną przez nią wartością, Alicja akceptuje klucz K i wysyła Bolkowi
  $E_K(S_A(g^a, g^b))$
- Bolek deszyfruje otrzymaną wiadomość, weryfikuje podpis Alicji i jeśli wynik jest pozytywny, akceptuje klucz K

- Bolek podpisuje konkatenację g^b, g^a, szyfruje podpis kluczem K i wysyła Alicji
  $g^b \pmod p$ oraz $E_K(S_B(g^b, g^a))$
- Alicja oblicza klucz $K = (g^b)^a \pmod p$, deszyfruje otrzymane dane oraz używa klucza publicznego Bolka do sprawdzenia podpisu pod wartością funkcji hashującej z konkatenacji obu eksponensów. Jeśli wartość funkcji hashującej zgadza się z otrzymaną przez nią wartością, Alicja akceptuje klucz K i wysyła Bolkowi
  $E_K(S_A(g^a, g^b))$
- Bolek deszyfruje otrzymaną wiadomość, weryfikuje podpis Alicji i jeśli wynik jest pozytywny, akceptuje klucz K

17.4 Uzgadnianie klucza z szyfrowaniem

- Zakładamy, że Alicja i Bolek (użytkownik i komputer) znają hasło P.
- Alicja generuje losowo parę kluczy (publiczny i prywatny), szyfruje klucz publiczny K' używając algorytmu symetrycznego wykorzystującego hasło P jako klucz i wysyła do Bolka $E_P(K')$
- Bolek deszyfruje wiadomość otrzymaną od Alicji otrzymując klucz K', następnie generuje klucz sesyjny K, szyfruje ten klucz kluczem publicznym K' oraz kluczem P i wysyła Alicji $E_P(E_{K'}(K))$

17.4 Uzgadnianie klucza z szyfrowaniem

- Zakładamy, że Alicja i Bolek (użytkownik i komputer) znają hasło P.
- Alicja generuje losowo parę kluczy (publiczny i prywatny), szyfruje klucz publiczny K' używając algorytmu symetrycznego wykorzystującego hasło P jako klucz i wysyła do Bolka $E_P(K')$
- Bolek deszyfruje wiadomość otrzymaną od Alicji otrzymując klucz K', następnie generuje klucz sesyjny K, szyfruje ten klucz kluczem publicznym K' oraz kluczem P i wysyła Alicji $E_P(E_{K'}(K))$

17.4 Uzgadnianie klucza z szyfrowaniem

- Zakładamy, że Alicja i Bolek (użytkownik i komputer) znają hasło P.
- Alicja generuje losowo parę kluczy (publiczny i prywatny), szyfruje klucz publiczny K' używając algorytmu symetrycznego wykorzystującego hasło P jako klucz i wysyła do Bolka $E_P(K')$
- Bolek deszyfruje wiadomość otrzymaną od Alicji otrzymując klucz K', następnie generuje klucz sesyjny K, szyfruje ten klucz kluczem publicznym K' oraz kluczem P i wysyła Alicji $E_P(E_{K'}(K))$

17.4 Uzgadnianie klucza z szyfrowaniem

- Zakładamy, że Alicja i Bolek (użytkownik i komputer) znają hasło P.
- Alicja generuje losowo parę kluczy (publiczny i prywatny), szyfruje klucz publiczny K' używając algorytmu symetrycznego wykorzystującego hasło P jako klucz i wysyła do Bolka $E_P(K')$
- Bolek deszyfruje wiadomość otrzymaną od Alicji otrzymując klucz K', następnie generuje klucz sesyjny K, szyfruje ten klucz kluczem publicznym K' oraz kluczem P i wysyła Alicji $E_P(E_{K'}(K))$

- Alicja deszyfruje wiadomość otrzymaną od Bolka uzyskując klucz K. Następnie generuje ciąg losowy r_A, szyfruje go kluczem K i wysyła do Bolka

  $E_K(r_A)$

- Bolek deszyfruje wiadomość otrzymaną od Alicji otrzymując ciąg r_A, generuje własny ciąg losowy r_B, szyfruje obydwa ciągi kluczem K i wysyła Alicji

  $E_K(r_A, r_B)$

- Alicja deszyfruje wiadomość uzyskując r_A i r_B. Jeśli r_A jest prawidłowe, szyfruje r_B i wysyła do Bolka

  $E_K(r_B)$

- Bolek deszyfruje wiadomość i jeśli r_B jest prawidłowe klucz K zostaje zaakceptowany jako klucz sesyjny.

- Alicja deszyfruje wiadomość otrzymaną od Bolka uzyskując klucz K. Następnie generuje ciąg losowy r_A, szyfruje go kluczem K i wysyła do Bolka

  $E_K(r_A)$

- Bolek deszyfruje wiadomość otrzymaną od Alicji otrzymując ciąg r_A, generuje własny ciąg losowy r_B, szyfruje obydwa ciągi kluczem K i wysyła Alicji

  $E_K(r_A, r_B)$

- Alicja deszyfruje wiadomość uzyskując r_A i r_B. Jeśli r_A jest prawidłowe, szyfruje r_B i wysyła do Bolka

  $E_K(r_B)$

- Bolek deszyfruje wiadomość i jeśli r_B jest prawidłowe klucz K zostaje zaakceptowany jako klucz sesyjny.

- Alicja deszyfruje wiadomość otrzymaną od Bolka uzyskując klucz K. Następnie generuje ciąg losowy r_A, szyfruje go kluczem K i wysyła do Bolka

  $E_K(r_A)$

- Bolek deszyfruje wiadomość otrzymaną od Alicji otrzymując ciąg r_A, generuje własny ciąg losowy r_B, szyfruje obydwa ciągi kluczem K i wysyła Alicji

  $E_K(r_A, r_B)$

- Alicja deszyfruje wiadomość uzyskując r_A i r_B. Jeśli r_A jest prawidłowe, szyfruje r_B i wysyła do Bolka

  $E_K(r_B)$

- Bolek deszyfruje wiadomość i jeśli r_B jest prawidłowe klucz K zostaje zaakceptowany jako klucz sesyjny.

- Alicja deszyfruje wiadomość otrzymaną od Bolka uzyskując klucz K. Następnie generuje ciąg losowy r_A, szyfruje go kluczem K i wysyła do Bolka

  $E_K(r_A)$

- Bolek deszyfruje wiadomość otrzymaną od Alicji otrzymując ciąg r_A, generuje własny ciąg losowy r_B, szyfruje obydwa ciągi kluczem K i wysyła Alicji

  $E_K(r_A, r_B)$

- Alicja deszyfruje wiadomość uzyskując r_A i r_B. Jeśli r_A jest prawidłowe, szyfruje r_B i wysyła do Bolka

  $E_K(r_B)$

- Bolek deszyfruje wiadomość i jeśli r_B jest prawidłowe klucz K zostaje zaakceptowany jako klucz sesyjny.

- Istnieją różne implementacje tego protokołu, z wykorzystaniem algorytmu RSA czy ElGamala. Protokół ten wzmacnia bezpieczeństwo przy uzgadnianiu klucza sesyjnego.

- Istnieją różne implementacje tego protokołu, z wykorzystaniem algorytmu RSA czy ElGamala. Protokół ten wzmacnia bezpieczeństwo przy uzgadnianiu klucza sesyjnego.

17.5 ssh (secure shell)

- Protokół umożliwiający bezpieczne logowanie się do komputerów w sieci stworzony przez Tatu Ylönena, który skutecznie zapobiega takim metodom ataku jak IP Spoofing i DNS Spoofing, czy też podsłuchiwaniu haseł lub transmitowanych danych.
- Obecnie rozwijana jest wersja OpenSSH, na licencji GNU.

17.5 ssh (secure shell)

- Protokół umożliwiający bezpieczne logowanie się do komputerów w sieci stworzony przez Tatu Ylönena, który skutecznie zapobiega takim metodom ataku jak IP Spoofing i DNS Spoofing, czy też podsłuchiwaniu haseł lub transmitowanych danych.
- Obecnie rozwijana jest wersja OpenSSH, na licencji GNU.

17.5 ssh (secure shell)

- Protokół umożliwiający bezpieczne logowanie się do komputerów w sieci stworzony przez Tatu Ylönena, który skutecznie zapobiega takim metodom ataku jak IP Spoofing i DNS Spoofing, czy też podsłuchiwaniu haseł lub transmitowanych danych.
- Obecnie rozwijana jest wersja OpenSSH, na licencji GNU.

- przy instalacji programu generowana jest para kluczy algorytmu asymetrycznego przynależna danemu komputerowi — klucz publiczny komputera
  — host key
- przy uruchomieniu demona sshd generowana jest następna para kluczy — server keys. Publiczny jest dostępny do czytania, a prywatny jest przechowywany w pamięci komputera (nie jest zapisywany na dysku). Co godzinę para tych kluczy jest zmieniana.
- każdy użytkownik generuje kolejną parę kluczy (ssh-keygen), które służą do uwierzytelniania użytkownika. Klucz prywatny jest szyfrowany.

- przy instalacji programu generowana jest para kluczy algorytmu asymetrycznego przynależna danemu komputerowi — klucz publiczny komputera
  — host key
- przy uruchomieniu demona sshd generowana jest następna para kluczy — server keys. Publiczny jest dostępny do czytania, a prywatny jest przechowywany w pamięci komputera (nie jest zapisywany na dysku). Co godzinę para tych kluczy jest zmieniana.
- każdy użytkownik generuje kolejną parę kluczy (ssh-keygen), które służą do uwierzytelniania użytkownika. Klucz prywatny jest szyfrowany.

- przy instalacji programu generowana jest para kluczy algorytmu asymetrycznego przynależna danemu komputerowi — klucz publiczny komputera
  — host key
- przy uruchomieniu demona sshd generowana jest następna para kluczy — server keys. Publiczny jest dostępny do czytania, a prywatny jest przechowywany w pamięci komputera (nie jest zapisywany na dysku). Co godzinę para tych kluczy jest zmieniana.
- każdy użytkownik generuje kolejną parę kluczy (ssh-keygen), które służą do uwierzytelniania użytkownika. Klucz prywatny jest szyfrowany.

- przy instalacji programu generowana jest para kluczy algorytmu asymetrycznego przynależna danemu komputerowi — klucz publiczny komputera
  — host key
- przy uruchomieniu demona sshd generowana jest następna para kluczy — server keys. Publiczny jest dostępny do czytania, a prywatny jest przechowywany w pamięci komputera (nie jest zapisywany na dysku). Co godzinę para tych kluczy jest zmieniana.
- każdy użytkownik generuje kolejną parę kluczy (ssh-keygen), które służą do uwierzytelniania użytkownika. Klucz prywatny jest szyfrowany.

- Kiedy Alicja próbuje zalogować się na komputer Bolka, to komputer ten wysyła jej swoje dwa klucze publiczne host key i server key. Komputer Alicji sprawdza czy host key zgadza się z kluczem zapisanym w lokalnym pliku known-hosts.
- Jeśli wszystko się zgadza to Alicja generuje losowy klucz sesji i szyfruje go po kolei obydwoma kluczami publicznymi komputera Bolka i tak uzyskany kryptogram przesyła do Bolka.
- Bolek potrzebuje dwóch kluczy prywatnych do odszyfrowania klucza sesyjnego
- Bolek przesyła Alicji liczbę losową r_B zaszyfrowaną kluczem publicznym Alicji.

- Kiedy Alicja próbuje zalogować się na komputer Bolka, to komputer ten wysyła jej swoje dwa klucze publiczne host key i server key. Komputer Alicji sprawdza czy host key zgadza się z kluczem zapisanym w lokalnym pliku known-hosts.
- Jeśli wszystko się zgadza to Alicja generuje losowy klucz sesji i szyfruje go po kolei obydwoma kluczami publicznymi komputera Bolka i tak uzyskany kryptogram przesyła do Bolka.
- Bolek potrzebuje dwóch kluczy prywatnych do odszyfrowania klucza sesyjnego
- Bolek przesyła Alicji liczbę losową r_B zaszyfrowaną kluczem publicznym Alicji.

- Kiedy Alicja próbuje zalogować się na komputer Bolka, to komputer ten wysyła jej swoje dwa klucze publiczne host key i server key. Komputer Alicji sprawdza czy host key zgadza się z kluczem zapisanym w lokalnym pliku known-hosts.
- Jeśli wszystko się zgadza to Alicja generuje losowy klucz sesji i szyfruje go po kolei obydwoma kluczami publicznymi komputera Bolka i tak uzyskany kryptogram przesyła do Bolka.
- Bolek potrzebuje dwóch kluczy prywatnych do odszyfrowania klucza sesyjnego
- Bolek przesyła Alicji liczbę losową r_B zaszyfrowaną kluczem publicznym Alicji.

- Kiedy Alicja próbuje zalogować się na komputer Bolka, to komputer ten wysyła jej swoje dwa klucze publiczne host key i server key. Komputer Alicji sprawdza czy host key zgadza się z kluczem zapisanym w lokalnym pliku known-hosts.
- Jeśli wszystko się zgadza to Alicja generuje losowy klucz sesji i szyfruje go po kolei obydwoma kluczami publicznymi komputera Bolka i tak uzyskany kryptogram przesyła do Bolka.
- Bolek potrzebuje dwóch kluczy prywatnych do odszyfrowania klucza sesyjnego
- Bolek przesyła Alicji liczbę losową r_B zaszyfrowaną kluczem publicznym Alicji.

- Alicja deszyfruje otrzymany kryptogram i odsyła Bolkowi $H(r_B)$ udowadniając mu swoją tożsamość.

  W ten sposób zostaje ustanowiony bezpieczny kanał komunikacyjny.