Szyfrowanie z GnuPG

Bartosz Chodorowski

<chomzee@ethernet.pl>

7 stycznia 2010

Po co nam to?

- Rozważmy dwoje ludzi – Alicję i Boba, którzy chcą bezpiecznie komunikować się przez internet

Po co nam to?

- Rozważmy dwoje ludzi – Alicję i Boba, którzy chcą bezpiecznie komunikować się przez internet
- Internet nie jest bezpiecznym kanałem komunikacyjnym. Alicja i Bob narażeni są na podsłuch, celowe złośliwe zniekształcanie przesyłanych danych oraz próby podszycia się pod kogoś z nich w celu uzyskania poufnych informacji

Po co nam to?

- Rozważmy dwoje ludzi – Alicję i Boba, którzy chcą bezpiecznie komunikować się przez internet
- Internet nie jest bezpiecznym kanałem komunikacyjnym. Alicja i Bob narażeni są na podsłuch, celowe złośliwe zniekształcanie przesyłanych danych oraz próby podszycia się pod kogoś z nich w celu uzyskania poufnych informacji
- Dobry kryptosystem umożliwia:

Po co nam to?

- Rozważmy dwoje ludzi – Alicję i Boba, którzy chcą bezpiecznie komunikować się przez internet
- Internet nie jest bezpiecznym kanałem komunikacyjnym. Alicja i Bob narażeni są na podsłuch, celowe złośliwe zniekształcanie przesyłanych danych oraz próby podszycia się pod kogoś z nich w celu uzyskania poufnych informacji
- Dobry kryptosystem umożliwia:
  - szyfrowanie – wiadomość wysłana do Alicji może być odczytana tylko przez Alicję

Po co nam to?

- Rozważmy dwoje ludzi – Alicję i Boba, którzy chcą bezpiecznie komunikować się przez internet
- Internet nie jest bezpiecznym kanałem komunikacyjnym. Alicja i Bob narażeni są na podsłuch, celowe złośliwe zniekształcanie przesyłanych danych oraz próby podszycia się pod kogoś z nich w celu uzyskania poufnych informacji
- Dobry kryptosystem umożliwia:
  - szyfrowanie – wiadomość wysłana do Alicji może być odczytana tylko przez Alicję
  - podpis cyfrowy – Bob otrzymawszy wiadomość od Alicji ma pewność, że dane które otrzymał rzeczywiście wysłała Alicja i nie zostały one po drodze zniekształcone

Po co nam to?

- Rozważmy dwoje ludzi – Alicję i Boba, którzy chcą bezpiecznie komunikować się przez internet
- Internet nie jest bezpiecznym kanałem komunikacyjnym. Alicja i Bob narażeni są na podsłuch, celowe złośliwe zniekształcanie przesyłanych danych oraz próby podszycia się pod kogoś z nich w celu uzyskania poufnych informacji
- Dobry kryptosystem umożliwia:
  - szyfrowanie – wiadomość wysłana do Alicji może być odczytana tylko przez Alicję
  - podpis cyfrowy – Bob otrzymawszy wiadomość od Alicji ma pewność, że dane które otrzymał rzeczywiście wysłała Alicja i nie zostały one po drodze zniekształcone
- Przypadek szczególny: zastosowanie kryptografii przy dystrybucji oprogramowania

Kodowanie

- **Kod** jako sposób zapisu informacji

Kodowanie

- **Kod** jako sposób zapisu informacji
  na przykład:
  - kod Morse'a
  - ASCII, Unicode
  - kod binarny
  - kod Graya
  - kod genetyczny
  - BASE64

Kodowanie

- **Kod** jako sposób zapisu informacji
  na przykład:
  - kod Morse'a
  - ASCII, Unicode
  - kod binarny
  - kod Graya
  - kod genetyczny
  - BASE64
- **Kodowanie** – przekształcenie informacji z jednej formy w inną

Kodowanie

- **Kod** jako sposób zapisu informacji
  na przykład:
  - kod Morse'a
  - ASCII, Unicode
  - kod binarny
  - kod Graya
  - kod genetyczny
  - BASE64
- **Kodowanie** – przekształcenie informacji z jednej formy w inną
- Kod korekcyjny – kod nadmiarowy, który umożliwia wykrycie błędu w jednostce kodowania i jego automatyczne usunięcie

Kodowanie

- **Kod** jako sposób zapisu informacji
  na przykład:
  - kod Morse'a
  - ASCII, Unicode
  - kod binarny
  - kod Graya
  - kod genetyczny
  - BASE64
- **Kodowanie** – przekształcenie informacji z jednej formy w inną
- Kod korekcyjny – kod nadmiarowy, który umożliwia wykrycie błędu w jednostce kodowania i jego automatyczne usunięcie
  na przykład:
  - kod Hamminga
  - kod kontroli parzystości

Szyfrowanie

- **Szyfrowanie** – rodzaj kodowania mający na celu utajnienie informacji tak, aby była możliwa do odczytania jedynie dla osoby, która posiada odpowiedni **klucz**

Szyfrowanie

- **Szyfrowanie** – rodzaj kodowania mający na celu utajnienie informacji tak, aby była możliwa do odczytania jedynie dla osoby, która posiada odpowiedni **klucz**
- Wiadomość przed zaszyfrowaniem nazywamy **tekstem jawnym** *(ang. plaintext)*, wiadomość zaszyfrowaną **szyfrogramem** lub **tekstem zaszyfrowanym** (ang. ciphertext)

Szyfrowanie

- **Szyfrowanie** – rodzaj kodowania mający na celu utajnienie informacji tak, aby była możliwa do odczytania jedynie dla osoby, która posiada odpowiedni **klucz**
- Wiadomość przed zaszyfrowaniem nazywamy **tekstem jawnym** *(ang. plaintext)*, wiadomość zaszyfrowaną **szyfrogramem** lub **tekstem zaszyfrowanym** (ang. ciphertext)
- Podział pierwszy:
  - szyfry symetryczne
  - szyfry asymetryczne

Szyfrowanie

- **Szyfrowanie** – rodzaj kodowania mający na celu utajnienie informacji tak, aby była możliwa do odczytania jedynie dla osoby, która posiada odpowiedni **klucz**
- Wiadomość przed zaszyfrowaniem nazywamy **tekstem jawnym** *(ang. plaintext)*, wiadomość zaszyfrowaną **szyfrogramem** lub **tekstem zaszyfrowanym** (ang. ciphertext)
- Podział pierwszy:
  - szyfry symetryczne
  - szyfry asymetryczne
- Podział drugi:
  - szyfry blokowe
  - szyfry strumieniowe

Funkcje haszujące (1/2)

- **Funkcja haszująca** – funkcja, która przyporządkowuje dowolnie dużej liczbie (wiadomości) krótką, zwykle posiadającą stały rozmiar wartość (skrót wiadomości)

Funkcje haszujące (1/2)

- **Funkcja haszująca** – funkcja, która przyporządkowuje dowolnie dużej liczbie (wiadomości) krótką, zwykle posiadającą stały rozmiar wartość (skrót wiadomości)
- Łatwo zauważyć, że taka funkcja nie może być różnowartościowa!

Funkcje haszujące (1/2)

- **Funkcja haszująca** – funkcja, która przyporządkowuje dowolnie dużej liczbie (wiadomości) krótką, zwykle posiadającą stały rozmiar wartość (skrót wiadomości)
- Łatwo zauważyć, że taka funkcja nie może być różnowartościowa!
- Własności „dobrych” funkcji haszujących:

Funkcje haszujące (1/2)

- **Funkcja haszująca** – funkcja, która przyporządkowuje dowolnie dużej liczbie (wiadomości) krótką, zwykle posiadającą stały rozmiar wartość (skrót wiadomości)
- Łatwo zauważyć, że taka funkcja nie może być różnowartościowa!
- Własności „dobrych” funkcji haszujących:
  - brak praktycznej możliwości wygenerowania wiadomości o takim samym skrócie jak zadana wiadomość

Funkcje haszujące (1/2)

- **Funkcja haszująca** – funkcja, która przyporządkowuje dowolnie dużej liczbie (wiadomości) krótką, zwykle posiadającą stały rozmiar wartość (skrót wiadomości)
- Łatwo zauważyć, że taka funkcja nie może być różnowartościowa!
- Własności „dobrych” funkcji haszujących:
  - brak praktycznej możliwości wygenerowania wiadomości o takim samym skrócie jak zadana wiadomość
  - brak praktycznej możliwości wygenerowanie dwóch wiadomości o takim samym skrócie

Funkcje haszujące (1/2)

- **Funkcja haszująca** – funkcja, która przyporządkowuje dowolnie dużej liczbie (wiadomości) krótką, zwykle posiadającą stały rozmiar wartość (skrót wiadomości)
- Łatwo zauważyć, że taka funkcja nie może być różnowartościowa!
- Własności „dobrych” funkcji haszujących:
  - brak praktycznej możliwości wygenerowania wiadomości o takim samym skrócie jak zadana wiadomość
  - brak praktycznej możliwości wygenerowanie dwóch wiadomości o takim samym skrócie
  - brak możliwości wnioskowania o wiadomości wejściowej na podstawie wartości skrótu

Funkcje haszujące (1/2)

- **Funkcja haszująca** – funkcja, która przyporządkowuje dowolnie dużej liczbie (wiadomości) krótką, zwykle posiadającą stały rozmiar wartość (skrót wiadomości)
- Łatwo zauważyć, że taka funkcja nie może być różnowartościowa!
- Własności „dobrych” funkcji haszujących:
  - brak praktycznej możliwości wygenerowania wiadomości o takim samym skrócie jak zadana wiadomość
  - brak praktycznej możliwości wygenerowanie dwóch wiadomości o takim samym skrócie
  - brak możliwości wnioskowania o wiadomości wejściowej na podstawie wartości skrótu
  - zmiana dowolnego pojedynczego bitu wiadomości powinna zmieniać średnio połowę bitów skrótu w sposób, który nie jest istotnie podatny na kryptoanalizę różnicową

Funkcje haszujące (2/2)

- Przykłady funkcji:

Funkcje haszujące (2/2)

- Przykłady funkcji:
  - MD2, MD4, MD5

Funkcje haszujące (2/2)

- Przykłady funkcji:
  - MD2, MD4, MD5
  - SHA-1

Funkcje haszujące (2/2)

- Przykłady funkcji:
  - MD2, MD4, MD5
  - SHA-1
  - SHA-224, SHA-256, SHA-384, SHA-512

Funkcje haszujące (2/2)

- Przykłady funkcji:
  - MD2, MD4, MD5
  - SHA-1
  - SHA-224, SHA-256, SHA-384, SHA-512
  - GHOST

Funkcje haszujące (2/2)

- Przykłady funkcji:
  - MD2, MD4, MD5
  - SHA-1
  - SHA-224, SHA-256, SHA-384, SHA-512
  - GHOST
- Przykład użycia (MD5):

```
(~) >> echo 'Witaj, świecie!' | md5sum
6c3bc181831142cbd15836b1c20033bf  -
(~) >> echo 'Witaj, Świecie!' | md5sum
f6904a9dc59c9b254e815a0225038f51  -
```

Funkcje haszujące (2/2)

- Przykłady funkcji:
  - MD2, MD4, MD5
  - SHA-1
  - SHA-224, SHA-256, SHA-384, SHA-512
  - GHOST
- Przykład użycia (MD5):

```
(~) >> echo 'Witaj, świecie!' | md5sum
6c3bc181831142cbd15836b1c20033bf  -
(~) >> echo 'Witaj, Świecie!' | md5sum
f6904a9dc59c9b254e815a0225038f51  -
```

- Zastosowanie funkcji haszujących do zapewnienia integralności danych

Szyfrowanie symetryczne

- M – wiadomość (tekst jawny)
  C – szyfrogram
  f – funkcja szyfrująca
  g – funkcja deszyfrująca
  K – tajny klucz

Szyfrowanie symetryczne

- M – wiadomość (tekst jawny)
  C – szyfrogram
  f – funkcja szyfrująca
  g – funkcja deszyfrująca
  K – tajny klucz
-

$$f(M, K) = C, \qquad g(C, K) = M \tag{1}$$

Przykład: XOR

- Niech p, q będą pojedynczymi bitami. Wówczas:

p	q	$p \oplus q$
0	0	0
0	1	1
1	0	1
1	1	0

Przykład: XOR

- Niech p, q będą pojedynczymi bitami. Wówczas:

p	q	$p \oplus q$
0	0	0
0	1	1
1	0	1
1	1	0

- Niech $P = (p_1, p_2, ...)$ oraz $Q = (q_1, q_2, ...)$. Operację „xorowania” ciągów bitów definiujemy w następujący sposób: $P \oplus Q = (p_1 \oplus q_1, p_2 \oplus q_2, \ldots)$

Przykład: XOR

- Niech p, q będą pojedynczymi bitami. Wówczas:

p	q	$p \oplus q$
0	0	0
0	1	1
1	0	1
1	1	0

- Niech $P = (p_1, p_2, ...)$ oraz $Q = (q_1, q_2, ...)$. Operację „xorowania” ciągów bitów definiujemy w następujący sposób: $P \oplus Q = (p_1 \oplus q_1, p_2 \oplus q_2, \ldots)$
- Można pokazać, że funkcja $g(X, Y) = f(X, Y) = X \oplus Y$ spełnia warunek (1)

Inne szyfry symetryczne

- DES

Inne szyfry symetryczne

- DES
- AES (Rijndael)

Inne szyfry symetryczne

- DES
- AES (Rijndael)
- Blowfish

Inne szyfry symetryczne

- DES
- AES (Rijndael)
- Blowfish
- IDEA

Inne szyfry symetryczne

- DES
- AES (Rijndael)
- Blowfish
- IDEA
- RC4

Inne szyfry symetryczne

- DES
- AES (Rijndael)
- Blowfish
- IDEA
- RC4 (pwn3d)

Szyfrowanie asymetryczne

- M – wiadomość (tekst jawny)
  C – szyfrogram
  f – funkcja szyfrująca
  g – funkcja deszyfrująca
  S – klucz prywatny
  P – klucz publiczny

Szyfrowanie asymetryczne

- M – wiadomość (tekst jawny)
  C – szyfrogram
  f – funkcja szyfrująca
  g – funkcja deszyfrująca
  S – klucz prywatny
  P – klucz publiczny
-

$$f(M, P) = C, \qquad g(C, S) = M \tag{2}$$

Szyfrowanie asymetryczne

- M – wiadomość (tekst jawny)
  C – szyfrogram
  f – funkcja szyfrująca
  g – funkcja deszyfrująca
  S – klucz prywatny
  P – klucz publiczny
- $$f(M, P) = C, \qquad g(C, S) = M \tag{2}$$
- Nie istnieje łatwo obliczalny algorytm ζ taki, że $\zeta(P) = S$

Podpisywanie wiadomości

- M – wiadomość lub jej hasz
  G – podpis
  φ – funkcja podpisująca
  ψ – funkcja weryfikująca
  S – klucz prywatny
  P – klucz publiczny

Podpisywanie wiadomości

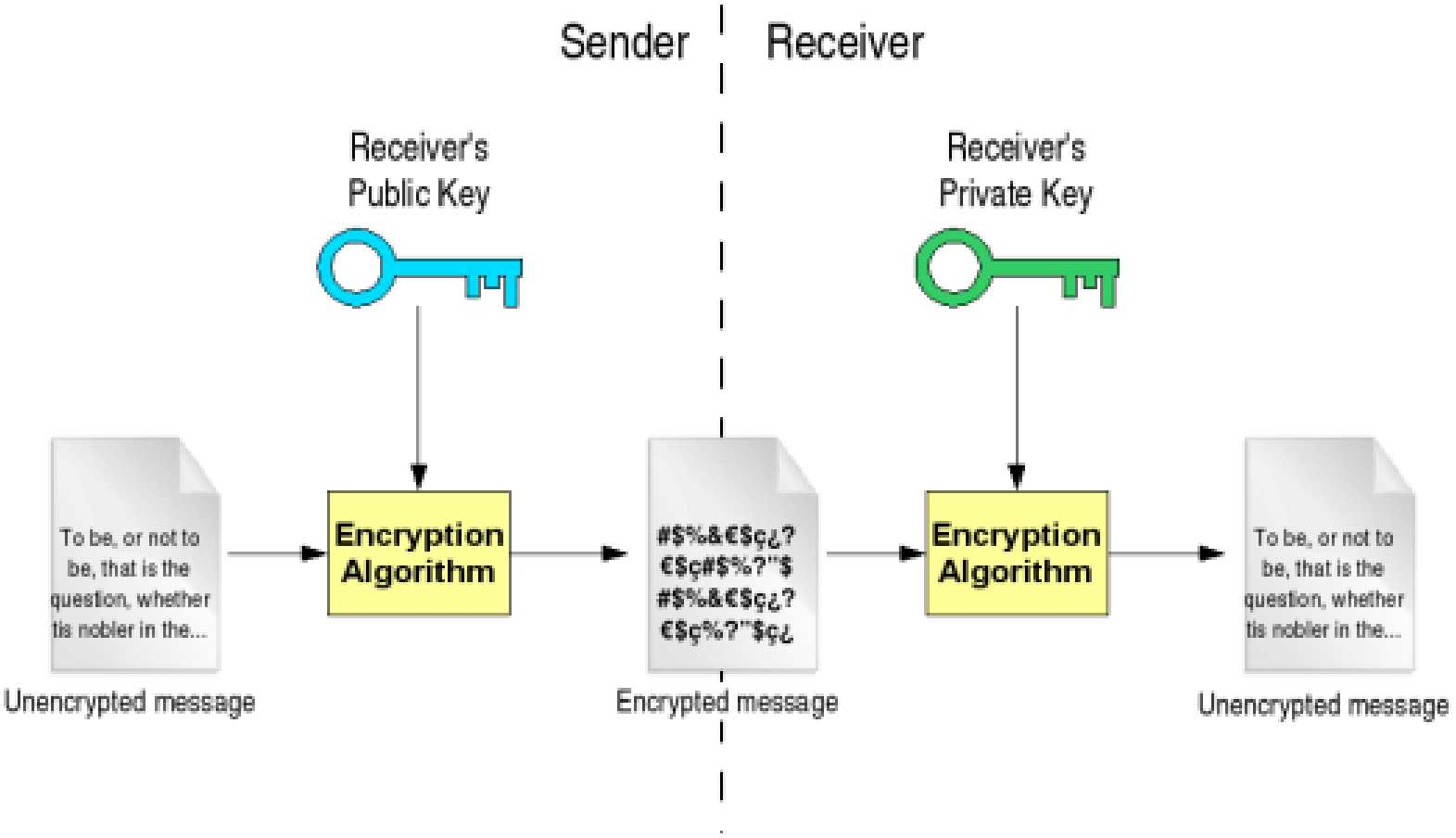
- M – wiadomość lub jej hasz
  G – podpis
  φ – funkcja podpisująca
  ψ – funkcja weryfikująca
  S – klucz prywatny
  P – klucz publiczny
-

$$\varphi(M, S) = G, \qquad \psi(G, P) = M \tag{3}$$

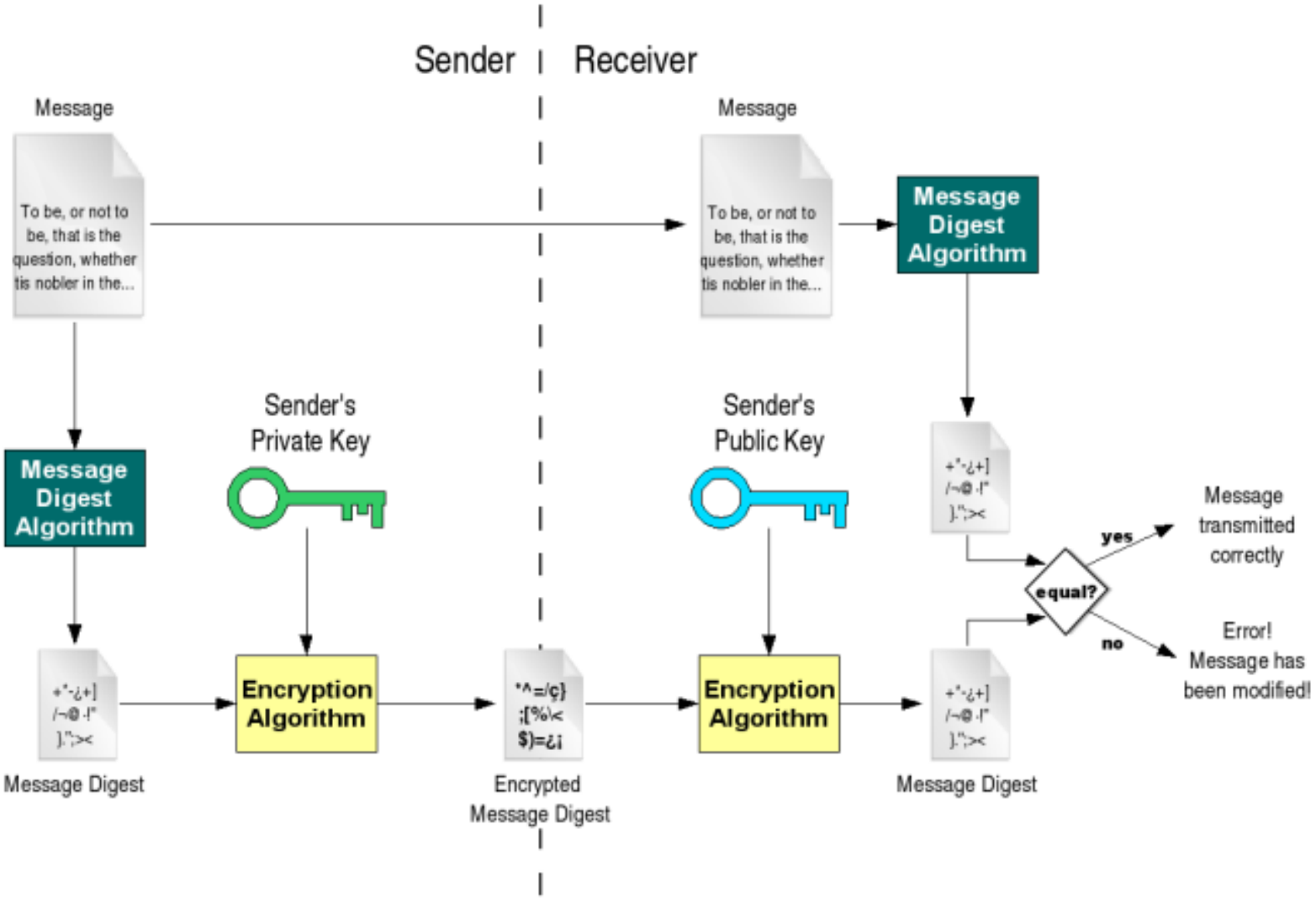
Podpisywanie wiadomości

- M – wiadomość lub jej hasz
  G – podpis
  φ – funkcja podpisująca
  ψ – funkcja weryfikująca
  S – klucz prywatny
  P – klucz publiczny
- $$\varphi(M, S) = G, \qquad \psi(G, P) = M \tag{3}$$
- Nie istnieje łatwo obliczalny algorytm ζ taki, że $\zeta(G) = S$

Szyfrowanie asymetryczne – rysunek



Podpis cyfrowy – rysunek



Przykład: RSA – generowanie klucza

- Alicja losuje duże liczby pierwsze p, q

Przykład: RSA – generowanie klucza

- Alicja losuje duże liczby pierwsze p, q
- Oblicza $n = pq$

Przykład: RSA – generowanie klucza

- Alicja losuje duże liczby pierwsze p, q
- Oblicza $n = pq$
- Oblicza $\varphi(pq) = (p-1)(q-1)$

Przykład: RSA – generowanie klucza

- Alicja losuje duże liczby pierwsze p, q
- Oblicza $n = pq$
- Oblicza $\varphi(pq) = (p-1)(q-1)$
- Wybiera e takie, że $1 < e < \varphi(pq)$ oraz e względnie pierwsze z $\varphi(pq)$

Przykład: RSA – generowanie klucza

- Alicja losuje duże liczby pierwsze p, q
- Oblicza $n = pq$
- Oblicza $\varphi(pq) = (p-1)(q-1)$
- Wybiera e takie, że $1 < e < \varphi(pq)$ oraz e względnie pierwsze z $\varphi(pq)$
- Za pomocą rozszerzonego algorytmu Euklidesa oblicza d, które spełnia kongruencję:

$$de \equiv 1 \mod \varphi(pq)$$

Przykład: RSA – generowanie klucza

- Alicja losuje duże liczby pierwsze p, q
- Oblicza $n = pq$
- Oblicza $\varphi(pq) = (p-1)(q-1)$
- Wybiera e takie, że $1 < e < \varphi(pq)$ oraz e względnie pierwsze z $\varphi(pq)$
- Za pomocą rozszerzonego algorytmu Euklidesa oblicza d, które spełnia kongruencję:

$$de \equiv 1 \mod \varphi(pq)$$

- Kluczem prywatnym jest para (d, n)

Przykład: RSA – generowanie klucza

- Alicja losuje duże liczby pierwsze p, q
- Oblicza $n = pq$
- Oblicza $\varphi(pq) = (p-1)(q-1)$
- Wybiera e takie, że $1 < e < \varphi(pq)$ oraz e względnie pierwsze z $\varphi(pq)$
- Za pomocą rozszerzonego algorytmu Euklidesa oblicza d, które spełnia kongruencję:

$$de \equiv 1 \mod \varphi(pq)$$

- Kluczem prywatnym jest para (d, n)
- Kluczem publicznym jest para (e, n)

Przykład: RSA – generowanie klucza

- Alicja losuje duże liczby pierwsze p, q
- Oblicza $n = pq$
- Oblicza $\varphi(pq) = (p-1)(q-1)$
- Wybiera e takie, że $1 < e < \varphi(pq)$ oraz e względnie pierwsze z $\varphi(pq)$
- Za pomocą rozszerzonego algorytmu Euklidesa oblicza d, które spełnia kongruencję:

$$de \equiv 1 \mod \varphi(pq)$$

- Kluczem prywatnym jest para (d, n)
- Kluczem publicznym jest para (e, n)
- Liczby p oraz q są „niszczone”

Przykład: RSA – Szyfrowanie

- Bob posiada parę (e, n) – klucz publiczny Alicji. Chce wysłać wiadomość M – dodatnią liczbę mniejszą od n

Przykład: RSA – Szyfrowanie

- Bob posiada parę (e, n) – klucz publiczny Alicji. Chce wysłać wiadomość M – dodatnią liczbę mniejszą od n
- Bob za pomocą algorytmu szybkiego potęgowania oblicza C takie, że:

$$M^e \equiv C \mod n$$

Przykład: RSA – Szyfrowanie

- Bob posiada parę (e, n) – klucz publiczny Alicji. Chce wysłać wiadomość M – dodatnią liczbę mniejszą od n
- Bob za pomocą algorytmu szybkiego potęgowania oblicza C takie, że:

$$M^e \equiv C \mod n$$

- C jest wysłanym szyfrogramem

Przykład: RSA – Deszyfrowanie

- Alicja odtwarza M korzystając z tej samej metody co Bob oraz poniższej własności:

$$C^d \equiv M \mod n$$

Przykład: RSA – Podpis cyfrowy

- Alicja chce podpisać wiadomość M

Przykład: RSA – Podpis cyfrowy

- Alicja chce podpisać wiadomość M
- Oblicza $h(M)$, gdzie h jest funkcją haszującą

Przykład: RSA – Podpis cyfrowy

- Alicja chce podpisać wiadomość M
- Oblicza $h(M)$, gdzie h jest funkcją haszującą
- Oblicza podpis $G = h(M)^d \mod n$

Przykład: RSA – Podpis cyfrowy

- Alicja chce podpisać wiadomość M
- Oblicza $h(M)$, gdzie h jest funkcją haszującą
- Oblicza podpis $G = h(M)^d \mod n$
- Bob weryfikuje podpis obliczając $G' = h(M)^e \mod n$

Przykład: RSA – Podpis cyfrowy

- Alicja chce podpisać wiadomość M
- Oblicza $h(M)$, gdzie h jest funkcją haszującą
- Oblicza podpis $G = h(M)^d \mod n$
- Bob weryfikuje podpis obliczając $G' = h(M)^e \mod n$
- Na mocy magii teorii liczb: $G = G'$

Przykład: RSA – Podpis cyfrowy

- Alicja chce podpisać wiadomość M
- Oblicza $h(M)$, gdzie h jest funkcją haszującą
- Oblicza podpis $G = h(M)^d \mod n$
- Bob weryfikuje podpis obliczając $G' = h(M)^e \mod n$
- Na mocy magii teorii liczb: $G = G'$ (chyba, że źli ludzie zakłócili komunikację)

Inne szyfry asymetryczne

- DSA

Inne szyfry asymetryczne

- DSA
- ElGamal

Inne szyfry asymetryczne

- DSA
- ElGamal – oparty na trudności problemu logarytmu dyskretnego w ciele liczb całkowitych modulo duża liczba pierwsza

PGP

- **Pretty Good Privacy (PGP)** – program kryptograficzny, jedno z najpopularniejszych narzędzi do szyfrowania poczty elektronicznej

PGP

- **Pretty Good Privacy (PGP)** – program kryptograficzny, jedno z najpopularniejszych narzędzi do szyfrowania poczty elektronicznej
- Pierwsza wersja została napisana przez Phila Zimmermanna w 1991 roku

PGP

- **Pretty Good Privacy (PGP)** – program kryptograficzny, jedno z najpopularniejszych narzędzi do szyfrowania poczty elektronicznej
- Pierwsza wersja została napisana przez Phila Zimmermanna w 1991 roku
- Licencja freeware

PGP

- **Pretty Good Privacy (PGP)** – program kryptograficzny, jedno z najpopularniejszych narzędzi do szyfrowania poczty elektronicznej
- Pierwsza wersja została napisana przez Phila Zimmermanna w 1991 roku
- Licencja freeware
- Kryptografia asymetryczna

PGP

- **Pretty Good Privacy (PGP)** – program kryptograficzny, jedno z najpopularniejszych narzędzi do szyfrowania poczty elektronicznej
- Pierwsza wersja została napisana przez Phila Zimmermanna w 1991 roku
- Licencja freeware
- Kryptografia asymetryczna
- Nowatorskie uwierzytelnianie siecią zaufania *(ang. web of trust)*

OpenPGP

- Zimmermann w 1997 powołał **OpenPGP Alliance**

OpenPGP

- Zimmermann w 1997 powołał **OpenPGP Alliance**
- ... jako grupę roboczą Internet Engineering Task Force (IETF)

OpenPGP

- Zimmermann w 1997 powołał **OpenPGP Alliance**
- ... jako grupę roboczą Internet Engineering Task Force (IETF)
- Organizacja ta zajmuje się definiowaniem i opieką nad standardem **OpenPGP**

OpenPGP

- Zimmermann w 1997 powołał **OpenPGP Alliance**
- ... jako grupę roboczą Internet Engineering Task Force (IETF)
- Organizacja ta zajmuje się definiowaniem i opieką nad standardem **OpenPGP**
- Obecna, specyfikacja: `http://tools.ietf.org/html/rfc4880`

GnuPG

- **GNU Privacy Guard (GnuPG** lub **GPG)**

GnuPG

- **GNU Privacy Guard** (**GnuPG** lub **GPG**)
- Wolne oprogramowanie (GNU GPL), część projektu GNU

GnuPG

- **GNU Privacy Guard** (**GnuPG** lub **GPG**)
- Wolne oprogramowanie (GNU GPL), część projektu GNU
- Alternatywa PGP, zgodna z OpenPGP

GnuPG

- **GNU Privacy Guard** (**GnuPG** lub **GPG**)
- Wolne oprogramowanie (GNU GPL), część projektu GNU
- Alternatywa PGP, zgodna z OpenPGP
- CLI, liczne nakładki graficzne

GnuPG

- **GNU Privacy Guard** (**GnuPG** lub **GPG**)
- Wolne oprogramowanie (GNU GPL), część projektu GNU
- Alternatywa PGP, zgodna z OpenPGP
- CLI, liczne nakładki graficzne
- Wsparcie ze strony wielu klientów poczty elektronicznej

GnuPG

- **GNU Privacy Guard** (**GnuPG** lub **GPG**)
- Wolne oprogramowanie (GNU GPL), część projektu GNU
- Alternatywa PGP, zgodna z OpenPGP
- CLI, liczne nakładki graficzne
- Wsparcie ze strony wielu klientów poczty elektronicznej
- Nie używa żadnych opatentowanych algorytmów

GnuPG

- **GNU Privacy Guard** (**GnuPG** lub **GPG**)
- Wolne oprogramowanie (GNU GPL), część projektu GNU
- Alternatywa PGP, zgodna z OpenPGP
- CLI, liczne nakładki graficzne
- Wsparcie ze strony wielu klientów poczty elektronicznej
- Nie używa żadnych opatentowanych algorytmów
- Wsparcie dla ElGamal, DSA, RSA, AES, 3DES, Blowfish, Twofish, CAST5, MD5, SHA-1, RIPE-MD-160 oraz TIGER

GnuPG

- **GNU Privacy Guard** (**GnuPG** lub **GPG**)
- Wolne oprogramowanie (GNU GPL), część projektu GNU
- Alternatywa PGP, zgodna z OpenPGP
- CLI, liczne nakładki graficzne
- Wsparcie ze strony wielu klientów poczty elektronicznej
- Nie używa żadnych opatentowanych algorytmów
- Wsparcie dla ElGamal, DSA, RSA, AES, 3DES, Blowfish, Twofish, CAST5, MD5, SHA-1, RIPE-MD-160 oraz TIGER
- Idealny do szyfrowania poczty, podpisywania plików. Nie nadaje się do szyfrowania całych partycji (dm_crypt + LUKS)

Pierwsze uruchomienie

```
(~) >> gpg
gpg: katalog ,,/home/alice/.gnupg'' utworzony
gpg: nowy plik ustawień ,,/home/alice/.gnupg/gpg.conf'' został utworzon
gpg: zbiór kluczy ,,/home/alice/.gnupg/secring.gpg'' został utworzony
gpg: zbiór kluczy ,,/home/alice/.gnupg/pubring.gpg'' został utworzony
gpg: Wpisz tutaj swoją wiadomość ...
```

Generowanie pary kluczy (1/5)

```
(~) >> gpg --gen-key
gpg (GnuPG) 2.0.11; Copyright (C) 2009 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Proszę wybrać rodzaj klucza:
   (1) Para kluczy dla algorytmów DSA i Elgamala (domyślne)
   (2) DSA (tylko do podpisywania)
   (5) RSA (tylko do podpisywania)
Twój wybór? 1
Para kluczy DSA będzie miała 1024 bitów długości.
Klucze ELG będą miały od 1024 do 4096 bitów długości.
Jakiej długości klucz wygenerować? (2048)
Żądana długość klucza to 2048 bitów.
```

Generowanie pary kluczy, (2/5)

```
Okres ważności klucza.
         0 = klucz nie ma określonego terminu ważności
       <n>  = termin ważności klucza upływa za n dni
       <n>w = termin ważności klucza upływa za n tygodni
       <n>m = termin ważności klucza upływa za n miesięcy
       <n>y = termin ważności klucza upływa za n lat
Okres ważności klucza? (0) 1y
Klucz traci ważność śro, 29 gru 2010, 00:11:26 CET
Czy wszystko się zgadza (t/N)? t
```

Generowanie pary kluczy, (3/5)

```
You need a user ID to identify your key; the software constructs the us
from the Real Name, Comment and Email Address in this form:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Imię i nazwisko: Alice Nowak
Adres poczty elektronicznej: AliceNowak.TestGPG@gmail.com
Komentarz:
Twój identyfikator użytkownika będzie wyglądał tak:
    "Alice Nowak <AliceNowak.TestGPG@gmail.com>"

Zmienić (I)mię/nazwisko, (K)omentarz, adres (E)mail, przejść (D)alej,
czy (W)yjść z programu? D
```

Generowanie pary kluczy, (4/5)

```
Musisz podać długie, skomplikowane hasło aby ochronić swój klucz tajny.

Musimy wygenerować dużo losowych bajtów. Dobrym pomysłem aby pomóc
komputerowi podczas generowania liczb pierwszych jest wykonywanie
w tym czasie innych działań (pisanie na klawiaturze, poruszanie
myszką, odwołanie się do dysków); dzięki temu generator liczb
losowych ma możliwość zebrania odpowiedniej ilości entropii.
```

Generowanie pary kluczy, (5/5)

```
gpg: klucz 5BE5CAFB został oznaczony jako obdarzony absolutnym zaufanie
klucz publiczny i prywatny (tajny) zostały utworzone i podpisane.

gpg: sprawdzanie bazy zaufania
gpg: potrzeba 3 marginalnych, 1 pełnych, model zaufania PGP
gpg: poziom: 0 poprawnych:   1 podpisanych:   0 zaufanie: 0-,0q,0n,0m,0
gpg: następne sprawdzanie bazy odbędzie się 2010-12-28
pub   1024D/5BE5CAFB 2009-12-28 [wygasa: 2010-12-28]
      Odcisk klucza = C053 F7BA 7028 AE86 8F9F  AD76 D7B3 6EEF 5BE5 CA
uid                  Alice Nowak <AliceNowak.TestGPG@gmail.com>
sub   2048g/218D0FF1 2009-12-28 [wygasa: 2010-12-28]
```

Eksportowanie klucza publicznego

```
(~) >> gpg -a --export
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.11 (GNU/Linux)

mQGiBEs5Oz8RBACVd3m34dPOKN3/So+E35Z4U7I9xvSUPGseE6LOI7eA3xasTJOj
lgVxPXZe2qHyq5WJDMKv4qYNLTtruYKtMrKDyhcgt1TDknh/wyfyiohEP6OZxdzA
ApoVBcXM9P7K6B1784sScS6pgCWz7BotxMbcFZHFFuwVbbBGtsLq3j+kxwCg53Xw
...
DwUCSzk7PwIbDAUJAeEzgAAKCRDXs27vW+XK+8KtAJ9Rc4hN9HnIt+iFvMyHYt+W
CE+d1wCdEd94ZFfrFtundn2U5g3AUzcMZjw=
=SytO
-----END PGP PUBLIC KEY BLOCK-----
(~) >> gpg -a --export > alice.asc
```

Importowanie klucza publicznego

```
(~) >> file bob.asc
bob.asc: PGP public key block
(~) >> gpg bob.asc
pub  2048R/BF1B1D9E 2009-12-28 Bob Kowalski <BobKowalski.TestGPG@gmail.
sub  2048R/69FFADFC 2009-12-28 [wygasa: 2014-12-27]
(~) >> gpg --import bob.asc
gpg: klucz BF1B1D9E: klucz publiczny ,,Bob Kowalski
  <BobKowalski.TestGPG@gmail.com>'' wczytano do zbioru
gpg: Ogółem przetworzonych kluczy: 1
gpg:           dołączono do zbioru: 1  (RSA: 1)
(~) >> gpg --list-keys
/home/alice/.gnupg/pubring.gpg
------------------------------
pub   1024D/5BE5CAFB 2009-12-28 [wygasa: 2010-12-28]
uid                  Alice Nowak <AliceNowak.TestGPG@gmail.com>
sub   2048g/218D0FF1 2009-12-28 [wygasa: 2010-12-28]

pub   2048R/BF1B1D9E 2009-12-28 [wygasa: 2014-12-27]
uid                  Bob Kowalski <BobKowalski.TestGPG@gmail.com>
sub   2048R/69FFADFC 2009-12-28 [wygasa: 2014-12-27]
```

Ustawianie zaufania (1/2)

```
(~) >> gpg --edit-key bob trust
pub  2048R/BF1B1D9E  utworzono: 2009-12-28  wygasa: 2014-12-27  użycie:
  zaufanie: nieznany       poprawność: nieznany
sub  2048R/69FFADFC  utworzono: 2009-12-28  wygasa: 2014-12-27  użycie:
  [    nieznane   ] (1). Bob Kowalski <BobKowalski.TestGPG@gmail.com>

Zastanów się jak bardzo ufasz temu użytkownikowi w kwestii sprawdzania
tożsamości innych użytkowników (czy sprawdzi on odciski kluczy pobrane
z różnych źródeł, dokumenty potwierdzające tożsamość, itd.).

  1 = nie wiem albo nie powiem
  2 = NIE ufam
  3 = mam ograniczone zaufanie
  4 = mam pełne zaufanie
  5 = ufam absolutnie
  m = powrót do głównego menu
```

Ustawianie zaufania (2/2)

```
Twoja decyzja? 5
Czy na pewno chcesz przypisać absolutne zaufanie temu kluczowi? (t/N) t

pub  2048R/BF1B1D9E  utworzono: 2009-12-28  wygasa: 2014-12-27  użycie:
  zaufanie: absolutne      poprawność: nieznany
sub  2048R/69FFADFC  utworzono: 2009-12-28  wygasa: 2014-12-27  użycie:
  [     nieznane   ] (1). Bob Kowalski <BobKowalski.TestGPG@gmail.com>
```

Szyfrowanie

```
(~) >> cat wiadomosc.txt
Ściśle tajna wiadomość :)
(~) >> gpg -a -e -r bob wiadomosc.txt
(~) >> cat wiadomosc.txt.asc
-----BEGIN PGP MESSAGE-----
Version: GnuPG v2.0.11 (GNU/Linux)

hQEMA8LAI31p/638AQgAhNx7jyAdoKz3KZQj9cPoVR7ZKC2xSqGBMlx5XpM/tIOY
1bDPJjstmNqV8M+5uMHPIzEz9IwKJ1ZqVaOR07rg9+5ZjZ9U68JNpmEoENYOmoEo
Y75Y7nWZP2N8o/YZonfvwVsjOQIFAsARlBMO+xxMUZLu1vRs8gXjWwjEW1G4hrDb
g/u0g/TfCGPf2Ku7fWPByylyPsxITL5MVU2tm8v5oskeye7gtp4QQXfhXJ9uhPJD
F57YlEJCrKfVHBXGh66lp95LK5Aih22mrVfUq/9ke76UZgV+4VIYMw5pNKTncpvd
hfPj1q7TycRo1oZSO7YY4hY7yQ+WHZ9diBVjJrgaZNJdAZnEqJFQsncoPIZ+cQPU
QQ30Pjbrv3MlksObSgzS77hlHWcjQvJR05KZ0eCJCB//Zms3cOrjppR5Z65Dgf2w
78Zy6hblrMVQFwes27omdFhSoNpmuOWD5pieJcAX
=D556
-----END PGP MESSAGE-----
```

Deszyfrowanie (1/2)

```
(~) >> cat odpowiedz.txt.asc
-----BEGIN PGP MESSAGE-----
Version: GnuPG v2.0.11 (GNU/Linux)

hQIOA0u70DUhjQ/xEAf/crOFZ/pNEGxF47BSe7znqpIsmu3YCDvL7FRjDEtO7Pcc
ivLtlwivcCpqXHU/n/QtX9o9Iqomuh7vQLejU1dwb+DRXNyd+T9bCgEdu3xmO3I7
FSNOgyJCsAyWaB6sCNWa9/+mFDaIEZODCgifSb56hdjYtncnH8olmEJSNXkG5gOk
6WQKBTpROOM2yvWCXjvdykAtUeYWd1FS+Xh47FiJ66fq2CVX4CERdhjTeVHRVmbL
sWfmM5plZHa5gFnWzmrKZN5EkHRHndSkfxrJ2vRjE3RoxyMRE6MXqK3j1uKeKJfU
woE5+VLuVKwz3Tp5JNUM7OV0vpRYjFQsT9+I9Hj/Hwf9EKu7n+LAkfFi/K78eZER
FOzKfsJv/a6XX3T19MUPaOs9/NT/1w+XUqS9NeZ7B5UeCM4CrNPJNW/Axt2MS8bf
ELtU91xlEx5BR/4f4ZfX6c8mL5YlMxVn3So9d2ZLYZRjOHQXVK+fHWMlI1XiQjp5
cAV53UEkghmWf5pYPJb4KRrFKJXGPrNsjLZ9GABvFFlm5wtXQTRVmzoCxZhzrWn5
UVKwVTWEMYrvPMOtHw+ib8tU97OEGKVhpex0MnTDOTZQPg2JNWiOCpV9JUYZ4orJ
KKiQcKE/Fp2mcMTGB+sqVMWBPteMaxX/QXipgU1l5RXbBVquatbZ1xQaHq/kLIfz
HtJ9AetOfodpBILnZArW1i4idLqUVFaifQz7FbbhRaj4pb/YXWGshY27FWh4rDgl
sAGx8K/+3e0dgCETqg0ERIyla+kDTfrclPJ4/xr4DOePzAzpv0gPmH3bGV0NFWAm
mCZWfjK7LB8zK8L2bfCgnDx6RzGt6l7S2HqikH79Xss=
=LvCz
-----END PGP MESSAGE-----
```

Deszyfrowanie (2/2)

```
(~) >> gpg odpowiedz.txt.asc

Musisz podać hasło aby odbezpieczyć klucz prywatny użytkownika:
,,Alice Nowak <AliceNowak.TestGPG@gmail.com>''
długość 2048 bitów, typ ELG, numer 218D0FF1, stworzony
  2009-12-28 (ID głównego klucza 5BE5CAFB)

gpg: zaszyfrowano 2048-bitowym kluczem ELG o identyfikatorze
  218D0FF1, stworzonym 2009-12-28
      ,,Alice Nowak <AliceNowak.TestGPG@gmail.com>''
(~) >> cat odpowiedz.txt
Dziękuję za ściśle tajną wiadomość, odpowiadam podobną. :)
```

Podpisywanie (1/2)

```
(~) >> cat dokument.txt
Niniejszy dokument jest bardzo ważny, dlatego
złożę na nim swój podpis.

Oświadczam, iż bardzo lubię Boba.

Z poważaniem,
Alicja
(~) >> gpg --clearsign dokument.txt

Musisz podać hasło aby odbezpieczyć klucz prywatny użytkownika:
,,Alice Nowak <AliceNowak.TestGPG@gmail.com>''
długość 1024 bitów, typ DSA, numer 5BE5CAFB, stworzony 2009-12-28
```

Podpisywanie (2/2)

```
(~) >> cat dokument.txt.asc
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Niniejszy dokument jest bardzo ważny, dlatego
złożę na nim swój podpis.

Oświadczam, iż bardzo lubię Boba.

Z poważaniem,
Alicja
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v2.0.11 (GNU/Linux)

iEYEARECAAYFAks55Z4ACgkQ17Nu71vlyvsVeQCgwXiRpO8iuuwV7qHDD74QV5Tv
i/UAnipzTujnCNnOgVAHC03PGkkQWI/K
=+fsr
-----END PGP SIGNATURE-----
```

Sprawdzanie podpisu

```
(~) >> gpg --verify dokument.txt.asc
gpg: Podpisano w wto, 29 gru 2009, 12:18:54 CET kluczem DSA
  o numerze 5BE5CAFB
gpg: Poprawny podpis złożony przez
  ,,Alice Nowak <AliceNowak.TestGPG@gmail.com>''
```

Podpisywanie do innego pliku

```
(~) >> rm dokument.txt.asc
(~) >> gpg -a --detach-sign dokument.txt

Musisz podać hasło aby odbezpieczyć klucz prywatny użytkownika:
,,Alice Nowak <AliceNowak.TestGPG@gmail.com>''
długość 1024 bitów, typ DSA, numer 5BE5CAFB, stworzony 2009-12-28

(~) >> cat dokument.txt.asc
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v2.0.11 (GNU/Linux)

iEYEABECAAYFAks55kgACgkQ17Nu71vlyvvAoQCgjbMiFLn/yzm2BVEOPYqcBjsP
B3EAnA/EGwk316lq7OOMA90KULRM5OtF
=HRqD
-----END PGP SIGNATURE-----
```

Weryfikacja podpisu

```
(~) >> gpg --verify dokument.txt.asc
gpg: Podpisano w wto, 29 gru 2009, 12:21:44 CET kluczem DSA
  o numerze 5BE5CAFB
gpg: Poprawny podpis złożony przez
  ,,Alice Nowak <AliceNowak.TestGPG@gmail.com>''
(~) >> gpg dokument.txt.asc
gpg: Podpisano w wto, 29 gru 2009, 12:21:44 CET kluczem DSA
  o numerze 5BE5CAFB
gpg: Poprawny podpis złożony przez
  ,,Alice Nowak <AliceNowak.TestGPG@gmail.com>''
```

Ingerencja w podpisany dokument

```
(~) >> echo Linia dodana na próbę >> dokument.txt
(~) >> gpg dokument.txt.asc
gpg: Podpisano w wto, 29 gru 2009, 12:21:44 CET kluczem DSA
  o numerze 5BE5CAFB
gpg: NIEPOPRAWNY podpis złożony przez
  ,,Alice Nowak <AliceNowak.TestGPG@gmail.com>''
```

Unieważnianie klucza (1/4)

```
(~) >> gpg --gen-revoke alice

sec  1024D/5BE5CAFB 2009-12-28 Alice Nowak <AliceNowak.TestGPG@gmail.co

Stworzyć certyfikat unieważnienia tego klucza? (t/N) t
Proszę wybrać powód unieważnienia:
  0 = nie podano przyczyny
  1 = klucz został skompromitowany
  2 = klucz został zastąpiony
  3 = klucz nie jest już używany
  Q = Anuluj

(Prawdopodobnie chcesz tu wybrać 1)
Twoja decyzja? 1
Wprowadź opis (nieobowiązkowy) i zakończ go pustą linią:
> Eve ukradła klucz z mojego komputera
>
```

Unieważnianie klucza (2/4)

```
Powód unieważnienia: klucz został skompromitowany
Eve ukradła klucz z mojego komputera
Informacje poprawne? (t/N) t

Musisz podać hasło aby odbezpieczyć klucz prywatny użytkownika:
,,Alice Nowak <AliceNowak.TestGPG@gmail.com>''
długość 1024 bitów, typ DSA, numer 5BE5CAFB, stworzony 2009-12-28

Certyfikat unieważnienia został utworzony.
```

Uniewaznianie klucza (3/4)

```
Należy przenieść go na nośnik który można bezpiecznie ukryć; jeśli źli
ludzie dostaną ten certyfikat w swoje ręce, mogą użyć go do uczynienia
klucza nieużytecznym.

Niezłym pomysłem jest wydrukowanie certyfikatu unieważnienia i schowani
wydruku w bezpiecznym miejscu, na wypadek gdyby nośnik z certyfikatem
stał się nieczytelny. Ale należy zachować ostrożność, systemy drukowani
różnych komputerów mogą zachować treść wydruku i udostępnić ją osobom
nieupoważnionym.
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.11 (GNU/Linux)
Comment: A revocation certificate should follow

iG4EIBECAC4FAks58AgnHQJFdmUgdWtyYWTFgmEga2x1Y3ogeiBtb2plZ28ga29t
cHV0ZXJhAAoJENezbu9b5cr7wHgAoMnU7x+Sbuw/8GRAAUk6BGC8/Q1HAJ9p1Z0M
q3ebx4VOF8jkVG9D034bXg==
=FI/z
-----END PGP PUBLIC KEY BLOCK-----
```

Unieważnianie klucza (4/4)

```
(~) >> gpg revoke.asc
gpg: oddzielony podpis klasy 0x20.
gpg: Podpisano w wto, 29 gru 2009, 13:03:20 CET kluczem DSA
  o numerze 5BE5CAFB
gpg: osobny certyfikat unieważnienia - użyj ,,gpg --import''
  aby go wczytać
```

Inne przydatne polecenia

- `sign-key` – złożenie podpisu na kluczu

Inne przydatne polecenia

- `sign-key` – złożenie podpisu na kluczu
- `send-keys` – eksport kluczy do serwera kluczy

Inne przydatne polecenia

- `sign-key` – złożenie podpisu na kluczu
- `send-keys` – eksport kluczy do serwera kluczy
- `search-keys` – szukanie kluczy na serwerze

Inne przydatne polecenia

- `sign-key` – złożenie podpisu na kluczu
- `send-keys` – eksport kluczy do serwera kluczy
- `search-keys` – szukanie kluczy na serwerze
- `recv-keys` – import kluczy z serwera kluczy

Inne przydatne polecenia

- `sign-key` – złożenie podpisu na kluczu
- `send-keys` – eksport kluczy do serwera kluczy
- `search-keys` – szukanie kluczy na serwerze
- `recv-keys` – import kluczy z serwera kluczy
- `refresh-keys` – odświeżenie wszystkich kluczy z serwera

Inne przydatne polecenia

- `sign-key` – złożenie podpisu na kluczu
- `send-keys` – eksport kluczy do serwera kluczy
- `search-keys` – szukanie kluczy na serwerze
- `recv-keys` – import kluczy z serwera kluczy
- `refresh-keys` – odświeżenie wszystkich kluczy z serwera
- `help` – :-)

Szyfrowanie poczty

GnuPG + Thunderbird

- Enigmail – wtyczka do programu Thunderbird

GnuPG + Thunderbird

- Enigmail – wtyczka do programu Thunderbird
- Banalnie prosta w instalacji

GnuPG + Thunderbird

- Enigmail – wtyczka do programu Thunderbird
- Banalnie prosta w instalacji
- `http://enigmail.mozdev.org/`

GnuPG + mutt

- `mutt` jest przystosowany do współpracy z GnuPG

GnuPG + mutt

- `mutt` jest przystosowany do współpracy z GnuPG
- Wymagana konfiguracja, przykład w `/usr/share/doc/`

GnuPG + mutt

- `mutt` jest przystosowany do współpracy z GnuPG
- Wymagana konfiguracja, przykład w `/usr/share/doc/`
- Dodatkową konfigurację umieszczamy w `~/.gpg.rc`

GnuPG + mutt

- `mutt` jest przystosowany do współpracy z GnuPG
- Wymagana konfiguracja, przykład w `/usr/share/doc/`
- Dodatkową konfigurację umieszczamy w `~/.gpg.rc`
- W pliku `~/.muttrc` dopisujemy linię:
  `source "~/.gpg.rc"`

Pytania?

Bibliografia

- http://www.gnupg.org/ – strona domowa GnuPG
- http://www.openpgp.org/ – strona domowa OpenPGP
- http://www.ietf.org/rfc/rfc4880.txt – OpenPGP Message Format
- http://gdp.globus.org/gt4-tutorial/ – rysunki
- http://wikipedia.org/

Dziękuję za uwagę.