Kodowanie i szyfrowanie

Sieci komputerowe

Wykład 11

Marcin Bieńkowski

Kodowanie

Kody

Do przesyłanych wiadomości dodajemy dodatkowe bity kontrolne.

- **Kody detekcyjne:** pozwalają **wykryć** niektóre przekłamania transmisji.
- **Kody korekcyjne:** pozwalają **wykryć i poprawić** niektóre przekłamania transmisji.

Skąd się biorą błędy?

Najczęściej: błędy w warstwie fizycznej (bity → analogowy sygnał → bity), bo analogowy sygnał dociera zniekształcony.

- Przekłamania niektórych bitów.
- Przekłamanie ciągu bitów.
- Zgubienie niektórych bitów (rzadziej: wstawienie nieistniejących).

- Rzadziej: błędy urządzeń końcowych lub pośrednich (wadliwy RAM, błędy w oprogramowaniu).

Kody detekcyjne

- **Sumy kontrolne**
  - proste sumy
  - kody CRC
- **Kody MAC**

Proste sumy kontrolne

- Najprostszy wariant kodów detekcyjnych.
- Dodajemy do siebie (16/32-bitowe) słowa w przesyłanej wiadomości
  - Warianty: przeniesienia, negowanie bitów, …
  - Nie wykrywają zamian słów.

- Efektywnie obliczane przez CPU.
- Stosowane w warstwie sieciowej (IP) i transportowej (TCP/UDP).

Sumy kontrolne: bit parzystości

- Prosty wariant sumy kontrolnej.
- Do wiadomości doklejamy dodatkowy bit, ustawiony tak, aby **liczba ustawionych bitów w całości była parzysta**.
- Wykrywa przekłamania nieparzystej liczby bitów.

Kody CRC (*Cyclic Redundancy Check*)

- Oparte na dzieleniu w pierścieniu wielomianów nad ciałem F_2 (zbiór *{0, 1}* z działaniami modulo *2*).
- Efektywnie obliczane sprzętowo.
- Stosowane w warstwie łącza danych.

Przypomnienie: działania na wielomianach nad F_2

Niech $A(x) = x^{10} + x^8 + x^3$ oraz $B(x) = x^3 + x^2 + 1.$

Przypomnienie: działania na wielomianach nad F_2

Niech $A(x) = x^{10} + x^8 + x^3$ oraz $B(x) = x^3 + x^2 + 1$.

Wtedy:

Przypomnienie: działania na wielomianach nad F_2

Niech $A(x) = x^{10} + x^8 + x^3$ oraz $B(x) = x^3 + x^2 + 1.$

Wtedy:

- **Dodawanie:** $A(x) + B(x) = x^{10} + x^8 + x^2 + 1.$

Przypomnienie: działania na wielomianach nad F_2

Niech $A(x) = x^{10} + x^8 + x^3$ oraz $B(x) = x^3 + x^2 + 1$.

Wtedy:

- **Dodawanie:** $A(x) + B(x) = x^{10} + x^8 + x^2 + 1$.
- **Odejmowanie:** $B(x) + B(x) \equiv 0$, a zatem $B(x) = -B(x)$ i stąd $A(x) - B(x) = A(x) + B(x)$.

Przypomnienie: działania na wielomianach nad F_2

Niech $A(x) = x^{10} + x^8 + x^3$ oraz $B(x) = x^3 + x^2 + 1.$

Wtedy:

- **Dodawanie:** $A(x) + B(x) = x^{10} + x^8 + x^2 + 1.$
- **Odejmowanie:** B$(x) + B(x) \equiv 0$, a zatem B(x) = $-B(x)$ i stąd A(x) - $B(x)$ = $A(x) + B(x).$
- **Mnożenie:** jak zwykłe wielomiany, ale współczynniki są z F_2 . Przykładowo: $(x + 1) \cdot (x + 1) = x^2 + x + x + 1 = x^2 + 1.$

Przypomnienie: działania na wielomianach nad F_2

Niech $A(x) = x^{10} + x^8 + x^3$ oraz $B(x) = x^3 + x^2 + 1$.

Wtedy:

- **Dodawanie:** $A(x) + B(x) = x^{10} + x^8 + x^2 + 1$.
- **Odejmowanie:** $B(x) + B(x) \equiv 0$, a zatem $B(x) = -B(x)$ i stąd $A(x) - B(x) = A(x) + B(x)$.
- **Mnożenie:** jak zwykłe wielomiany, ale współczynniki są z F_2 . Przykładowo: $(x + 1) \cdot (x + 1) = x^2 + x + x + 1 = x^2 + 1$.
- **Dzielenie z resztą:** Niech $B(x) \neq 0$ i $k = \mathrm{st}(B)$. Istnieje dokładnie jedna para $Q(x)$ i $R(x)$, taka że $A(x) = Q(x) \cdot B(x) + R(x)$ oraz $\mathrm{st}(R) \leq k\text{-}1$.

Przypomnienie: działania na wielomianach nad F_2

Niech $A(x) = x^{10} + x^8 + x^3$ oraz $B(x) = x^3 + x^2 + 1.$

Wtedy:

- **Dodawanie:** $A(x) + B(x) = x^{10} + x^8 + x^2 + 1.$
- **Odejmowanie:** $B(x) + B(x) \equiv 0$, a zatem $B(x) = -B(x)$ i stąd $A(x) - B(x) = A(x) + B(x).$
- **Mnożenie:** jak zwykłe wielomiany, ale współczynniki są z F_2 . Przykładowo: $(x + 1) \cdot (x + 1) = x^2 + x + x + 1 = x^2 + 1.$
- **Dzielenie z resztą:** Niech $B(x) \neq 0$ i $k = \mathrm{st}(B)$. Istnieje dokładnie jedna para $Q(x)$ i $R(x)$, taka że $A(x) = Q(x) \cdot B(x) + R(x)$ oraz $\mathrm{st}(R) \leq k\text{-}1.$
  - Przykładowo: $A(x) = (x^7 + x^6 + x^4) \cdot B(x) + x.$

Wielomiany a ciągi bitów

Ciąg bitów *m* ↔ wielomian *M(x)*

- $m = 10100001 \leftrightarrow M(x) = x^7 + x^5 + x^0$
- $s = 101 \leftrightarrow S(x) = x^2 + x^0$

Wielomiany a ciągi bitów

Ciąg bitów m ↔ wielomian $M(x)$

- $m = 10100001 \;\leftrightarrow\; \mathrm{M(x)} = x^7 + x^5 + x^0$
- $s = 101 \;\leftrightarrow\; \mathrm{S(x)} = x^2 + x^0$
- $b = m\#s = 10100001101 \;\leftrightarrow\; B(x) = (x^7 + x^5 + x^0) \cdot x^3 + (x^2 + x^0)$

$$= M(x) \cdot x^3 + S(x)$$

konkatenacja napisów

$3 = \mathrm{st}(S)$

CRC

Ustalamy *r* i wielomian *G(x)* stopnia *r* (znany nadawcy i odbiorcy).

- W Ethernecie: $r = 32$, $G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + 1$.

CRC

Ustalamy *r* i wielomian *G(x)* stopnia *r* (znany nadawcy i odbiorcy).

- W Ethernecie: $r = 32$, $G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$.

Generowanie *r*-bitowej sumy kontrolnej *s*:

- Mamy wiadomość m ↔ M(x).
- Wysyłamy ciąg $b = m\#s \;\leftrightarrow\; B(x) = x^r \cdot M(x) + S(x)$,
gdzie **s wybieramy tak, żeby *B(x)* był podzielny przez *G(x)***.

CRC

Ustalamy *r* i wielomian *G(x)* stopnia *r* (znany nadawcy i odbiorcy).

- W Ethernecie: $r = 32$, $G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$.

Generowanie *r*-bitowej sumy kontrolnej *s*:

- Mamy wiadomość m ↔ M(x).
- Wysyłamy ciąg $b = m\#s \;\leftrightarrow\; B(x) = x^r \cdot M(x) + S(x)$,
  gdzie **s wybieramy tak, żeby *B(x)* był podzielny przez *G(x)***.

Odbiorca otrzymuje b′ ↔ B′(x)

- Odbiorca sprawdza, czy G(x) | B′(x).
  - Nie → musiało wystąpić przekłamanie.
  - Tak → **zakładamy**, że dane zostały przesłane poprawnie.

Obliczanie sumy kontrolnej

Jak znaleźć S(x) stopnia ≤ r-1, tak aby G(x) $\mid x^r \cdot M(x) + S(x)$ **?**

Obliczanie sumy kontrolnej

Bo chcemy, żeby s miało r bitów

Jak znaleźć S(x) stopnia ≤ r-1, tak aby G(x) $\mid x^r \cdot M(x) + S(x)$ **?**

Obliczanie sumy kontrolnej

Bo chcemy, żeby s miało r bitów

Jak znaleźć S(x) stopnia ≤ r-1, tak aby G(x) $\mid x^r \cdot M(x) + S(x)$ **?**

- Dzielimy $x^r \cdot M(x)$ przez $G(x)$:
  $x^r \cdot M(x) = Q(x) \cdot G(x) + R(x)$, gdzie st$(R) \leq r\text{-}1$

Obliczanie sumy kontrolnej

Bo chcemy, żeby s miało r bitów

Jak znaleźć S(x) stopnia ≤ r-1, tak aby G(x) | $x^r \cdot M(x) + S(x)$?

- Dzielimy $x^r \cdot M(x)$ przez $G(x)$:
  $x^r \cdot M(x) = Q(x) \cdot G(x) + R(x)$, gdzie st$(R) \leq r\text{-}1$
- Chcemy G(x) | $x^r \cdot M(x) + S(x)$
  $\Leftrightarrow$ G(x) | Q(x) · G(x) + R(x) + S(x)
  $\Leftrightarrow$ G(x) | R(x) + S(x)

Obliczanie sumy kontrolnej

Bo chcemy, żeby s miało r bitów

Jak znaleźć S(x) stopnia ≤ r-1, tak aby G(x) $\mid x^r \cdot M(x) + S(x)$ **?**

- Dzielimy $x^r \cdot M(x)$ przez $G(x)$:
  $x^r \cdot M(x) = Q(x) \cdot G(x) + R(x)$, gdzie st$(R) \leq r\text{-}1$

- Chcemy G(x) $\mid x^r \cdot M(x) + S(x)$
  $\Leftrightarrow$ G(x) $\mid$ Q(x) · G(x) + R(x) + S(x)
  $\Leftrightarrow$ G(x) $\mid$ R(x) + S(x)

- Ale st$(R + S) \leq r\text{-}1$ oraz st(G) = r. A zatem, R(x) + S(x) ≡ 0,
  czyli S(x) = R(x).

Obliczanie sumy kontrolnej

Bo chcemy, żeby s miało r bitów

Jak znaleźć S(x) stopnia ≤ r-1, tak aby G(x) | $x^r \cdot M(x) + S(x)$?

- Dzielimy $x^r \cdot M(x)$ przez $G(x)$:
  $x^r \cdot M(x) = Q(x) \cdot G(x) + R(x)$, gdzie st$(R) \leq r\text{-}1$
- Chcemy G(x) | $x^r \cdot M(x) + S(x)$
  $\Leftrightarrow$ G(x) | Q(x) · G(x) + R(x) + S(x)
  $\Leftrightarrow$ G(x) | R(x) + S(x)
- Ale st$(R + S) \leq r\text{-}1$ oraz st(G) = r. A zatem, R(x) + S(x) ≡ 0,
  czyli S(x) = R(x).

Istnieje dokładnie jedno żądane $S(x)$.

Przykład obliczania sumy kontrolnej

Przykład dla $G(x) = x^3 + x^2 + 1$.

- Chcemy wysłać wiadomość m = 10100001 $\leftrightarrow x^7 + x^5 + 1$.
- Dzielimy $x^r \cdot M(x) = x^{10} + x^8 + x^3$ przez $G(x)$, otrzymując $x^r \cdot M(x) = (x^7 + x^6 + x^4) \cdot G(x) + x$, tzn. $S(x) = x$.
- Suma kontrolna s powinna mieć st$(G) = 3$ bity, czyli $s = 010$.

Wykrywanie błędów transmisji

- Nadawca wysyła $b \leftrightarrow B(x)$.
- Odbiorca otrzymuje $b' \leftrightarrow B'(x) = B(x) + E(x)$.

Wykrywanie błędów transmisji

Zakładamy, że |b| = |b′|

- Nadawca wysyła $b \leftrightarrow B(x)$.
- Odbiorca otrzymuje $b' \leftrightarrow B'(x) = B(x) + E(x)$.

Wykrywanie błędów transmisji

Zakładamy, że $|b| = |b'|$

- Nadawca wysyła $b \leftrightarrow B(x)$.
- Odbiorca otrzymuje $b' \leftrightarrow B'(x) = B(x) + E(x)$.
- Odbiorca sprawdza, czy $G(x) \mid B'(x)$.
- Przekłamanie wykryte gdy $G(x) \nmid B'(x) \iff G(x) \nmid E(x)$.
- **Jakie typy błędów zostaną wykryte?**

Przykład

CRC wykorzystujące wielomian $G(x) = x^2 + x + 1$ wykryje błąd polegający na zamianie pięciu kolejnych bitów.

- Niech j = pozycja ostatniego błędnego bitu. Wtedy $E(x) = x^{j+4} + x^{j+3} + x^{j+2} + x^{j+1} + x^j = x^j \cdot (x^4 + x^3 + x^2 + x^1 + 1)$
- Pokażemy, że $G(x) \nmid E(x)$.

Przykład

CRC wykorzystujące wielomian $G(x) = x^2 + x + 1$ wykryje błąd polegający na zamianie pięciu kolejnych bitów.

- Niech j = pozycja ostatniego błędnego bitu. Wtedy
  $E(x) = x^{j+4} + x^{j+3} + x^{j+2} + x^{j+1} + x^j = x^j \cdot (x^4 + x^3 + x^2 + x^1 + 1)$

- Pokażemy, że $G(x) \nmid E(x)$.

  (1) $G(x) \nmid (x^4 + x^3 + x^2 + x^1 + 1)$,
  bo $(x^4 + x^3 + x^2 + x^1 + 1) = x^2 \cdot G(x) + (x + 1)$.

Przykład

CRC wykorzystujące wielomian $G(x) = x^2 + x + 1$ wykryje błąd polegający na zamianie pięciu kolejnych bitów.

- Niech j = pozycja ostatniego błędnego bitu. Wtedy $E(x) = x^{j+4} + x^{j+3} + x^{j+2} + x^{j+1} + x^j = x^j \cdot (x^4 + x^3 + x^2 + x^1 + 1)$
- Pokażemy, że $G(x) \nmid E(x)$.

(1) $G(x) \nmid (x^4 + x^3 + x^2 + x^1 + 1)$,
bo $(x^4 + x^3 + x^2 + x^1 + 1) = x^2 \cdot G(x) + (x + 1)$.

(2) $G(x)$ jest względnie pierwsze z x^j,
bo nie mają wspólnych dzielników innych niż 1.

Przykład

CRC wykorzystujące wielomian $G(x) = x^2 + x + 1$ wykryje błąd polegający na zamianie pięciu kolejnych bitów.

- Niech j = pozycja ostatniego błędnego bitu. Wtedy $E(x) = x^{j+4} + x^{j+3} + x^{j+2} + x^{j+1} + x^j = x^j \cdot (x^4 + x^3 + x^2 + x^1 + 1)$
- Pokażemy, że $G(x) \nmid E(x)$.

(1) $G(x) \nmid (x^4 + x^3 + x^2 + x^1 + 1)$,
bo $(x^4 + x^3 + x^2 + x^1 + 1) = x^2 \cdot G(x) + (x + 1)$.

(2) $G(x)$ jest względnie pierwsze z x^j,
bo nie mają wspólnych dzielników innych niż 1.

(1) + (2) $\Rightarrow$ $G(x) \nmid x^j \cdot (x^4 + x^3 + x^2 + x^1 + 1)$

CRC w Ethernecie

- Ethernet definiuje wielomian stopnia $n = 32$ równy:
  $G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4}$
  $+ x^{2} + x^{1} + 1.$

- Wykrywa on między innymi:
  - pojedyncze błędy bitów,
  - nieparzystą liczbę pojedynczych błędów bitów,
  - dwa błędy bitów oddalonych o co najwyżej $2^n - 1$,
  - przekłamania ciągu bitów nie dłuższego od n.

Kody MAC (*Message Authentication Code*)

- Kod uwierzytelniający.
- Cel: zapewnienie **integralności** wiadomości: trudno ją celowo zmodyfikować tak, żeby pozostało to niewykryte.

- Dostępne narzędzie: kryptograficzne funkcje haszujące
  - Funkcja h: funkcja haszująca, szybko obliczalna,
  - h: ciąg bitów dowolnej długości → ciąg bitów długości d.
  - Przykładowo d = 160 dla MD5, d = 256 dla SHA-256.
  - Dla dowolnego x znalezienie y, takiego że $h(x) = h(y)$ jest obliczeniowo trudne.

- Funkcję h można wykorzystać do wykrycia błędów w transmisji (np. MD5 podawane wraz z plikiem na stronie).
- **Jak wykorzystać *h* do zapewnienia integralności i uwierzytelnienia nadawcy?**

Kody MAC (*Message Authentication Code*)

- Kod uwierzytelniający.
- Cel: zapewnienie **integralności** wiadomości: trudno ją celowo zmodyfikować tak, żeby pozostało to niewykryte.

- Dostępne narzędzie: kryptograficzne funkcje haszujące
  - Funkcja h: funkcja haszująca, szybko obliczalna,
  - h: ciąg bitów dowolnej długości $\rightarrow$ ciąg bitów długości d.
  - Przykładowo d = 160 dla MD5, d = 256 dla SHA-256.
  - Dla dowolnego x znalezienie y, takiego że $h(x) = h(y)$ jest obliczeniowo trudne.

kolizja funkcji haszującej *h*

- Funkcję h można wykorzystać do wykrycia błędów w transmisji (np. MD5 podawane wraz z plikiem na stronie).
- **Jak wykorzystać *h* do zapewnienia integralności i uwierzytelnienia nadawcy?**

MAC a uwierzytelnianie nadawcy (1)

m = wiadomość

Pomysł 1: wyślij m, $h(m)$

MAC a uwierzytelnianie nadawcy (1)

m = wiadomość

Pomysł 1: wyślij m, $h(m)$

- **Problem:** Atakujący może wysłać m', $h(m')$.

MAC a uwierzytelnianie nadawcy (1)

m = wiadomość

Pomysł 1: wyślij $m, h(m)$

- **Problem:** Atakujący może wysłać $m', h(m')$.

Będziemy potrzebować sekretu s znanego nadawcy i odbiorcy.

MAC a uwierzytelnianie nadawcy (2)

m = wiadomość

s = sekret znany nadawcy i odbiorcy

Pomysł 2: wyślij m, $h(s\#m)$.

MAC a uwierzytelnianie nadawcy (2)

m = wiadomość

s = sekret znany nadawcy i odbiorcy

Pomysł 2: wyślij m, $h(s\#m)$.

- **Problem:** Duża część funkcji h działa w sposób strumieniowy: mając $h(x)$ można obliczyć $h(x\#y)$ nie znając x.
- **Atak przedłużeniowy:**
  - przechwyć oryginalny komunikat m, $h(s\#m)$, wybierz jakieś m'
  - na podstawie $h(s\#m)$ i m' oblicz $h(s\#m\#m')$
  - wyślij odbiorcy $m\#m'$, $h(s\#m\#m')$.

MAC a uwierzytelnianie nadawcy (2)

m = wiadomość

s = sekret znany nadawcy i odbiorcy

Pomysł 3: wyślij $m, h(m\#s)$.

MAC a uwierzytelnianie nadawcy (2)

m = wiadomość

s = sekret znany nadawcy i odbiorcy

Pomysł 3: wyślij $m, h(m\# s)$.

- Jeśli h działa w sposób strumieniowy i atakujący potrafi znaleźć m' takie że $h(m') = h(m)$, to może wysłać $m', h(m\# s)$ nie znając klucza s.
- Bezpieczeństwo takiego MAC jest co najwyżej tak dobre jak trudność znalezienia kolizji funkcji haszującej h.

MAC a uwierzytelnianie nadawcy (2)

m = wiadomość

s = sekret znany nadawcy i odbiorcy

Pomysł 3: wyślij $m, h(m\#s)$.

- Jeśli h działa w sposób strumieniowy i atakujący potrafi znaleźć m' takie że $h(m') = h(m)$, to może wysłać $m', h(m\#s)$ nie znając klucza s.
- Bezpieczeństwo takiego MAC jest co najwyżej tak dobre jak trudność znalezienia kolizji funkcji haszującej h.

pewnie trudne, ale może można lepiej?

Standard HMAC

m = wiadomość

s = sekret znany nadawcy i odbiorcy

- **HMAC:** wyślij m, $h(s\#h(s\#m))$.
- Znalezienie kolizji funkcji haszującej nie implikuje od razu złamania bezpieczeństwa standardu MAC.
- Wykorzystywany w protokołach szyfrujących (TLS, OpenVPN, ...), protokołach routingu dynamicznego, ...

Standard HMAC

m = wiadomość

s = sekret znany nadawcy i odbiorcy

pomijając drobne techniczne szczegóły

- **HMAC:** wyślij m, $h(s\# h(s\# m))$.
- Znalezienie kolizji funkcji haszującej nie implikuje od razu złamania bezpieczeństwa standardu MAC.
- Wykorzystywany w protokołach szyfrujących (TLS, OpenVPN, ...), protokołach routingu dynamicznego, ...

Korekcja błędów

Jak korygować błędy w transmisji?

- Kody detekcyjne + mechanizmy ARQ (wysyłania do skutku).
- Kody korekcyjne.

Kody (ogólnie)

***(a,b)*-kod:** zamienia wiadomość długości b na kod o długości $a \geq b$.

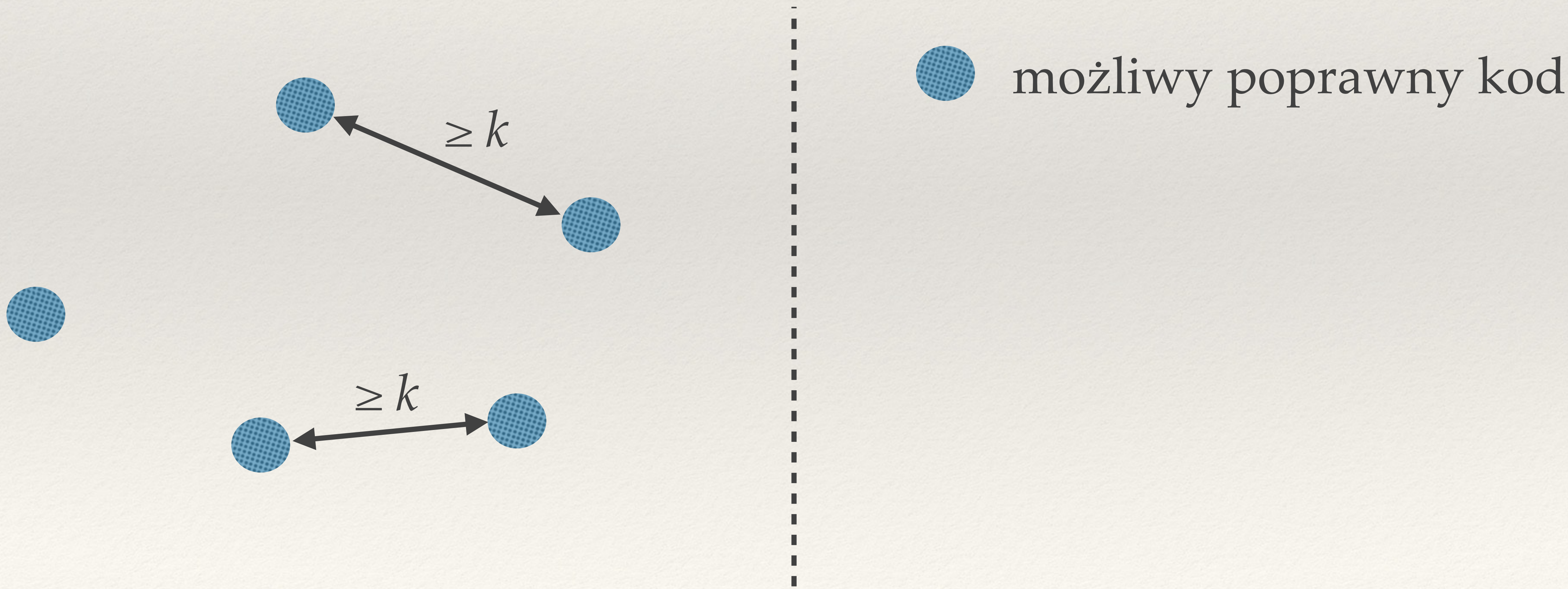
- Przykład: bit parzystości dla ciągów 7-bitowych to *(8,7)*-kod.
- Narzut kodu to a/b.

Odległość Hamminga dwóch kodów = minimalna liczba bitów, które musimy zmienić, żeby zmienić jeden kod w drugi.

Odległość Hamminga

Jeśli mamy kodowanie gwarantujące, że odległość Hamminga między dowolną parą kodów to co najmniej k:

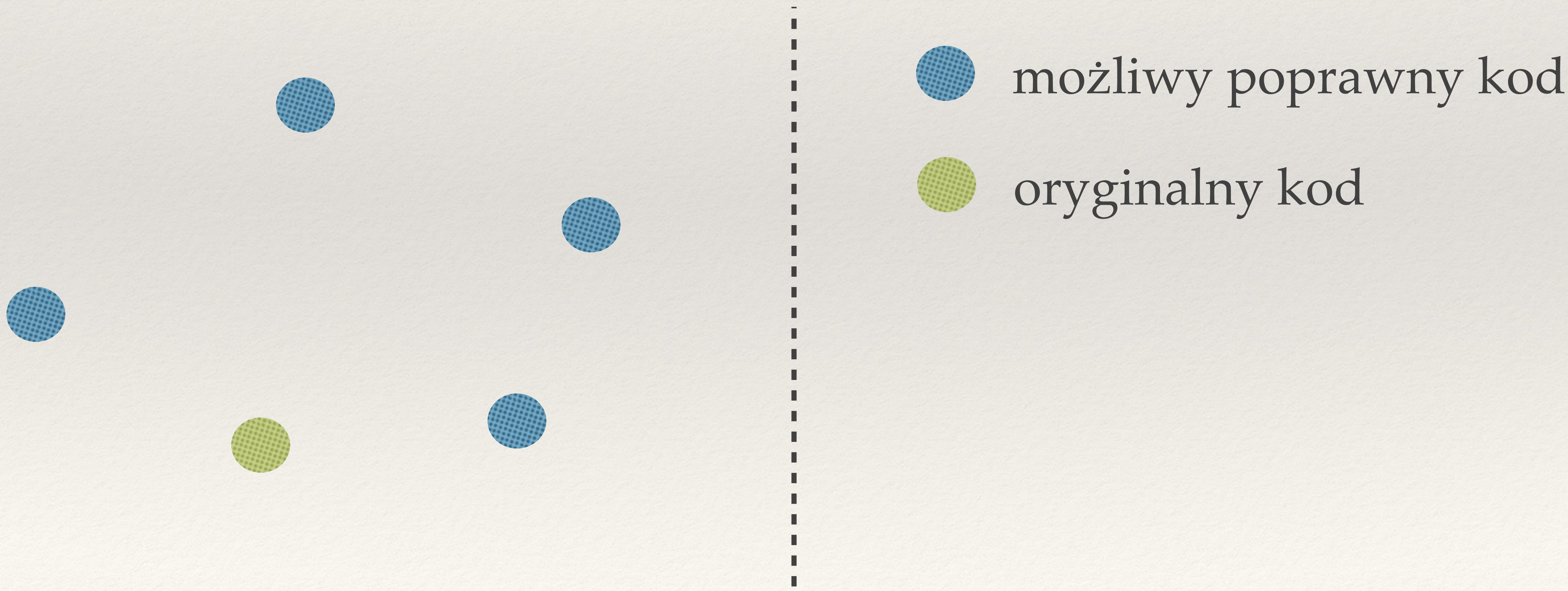
- potrafimy wykryć do *k-1* błędów pojedynczych bitów,
- potrafimy skorygować do *(k-1)*/2 błędów pojedynczych bitów.



Odległość Hamminga

Jeśli mamy kodowanie gwarantujące, że odległość Hamminga między dowolną parą kodów to co najmniej k:

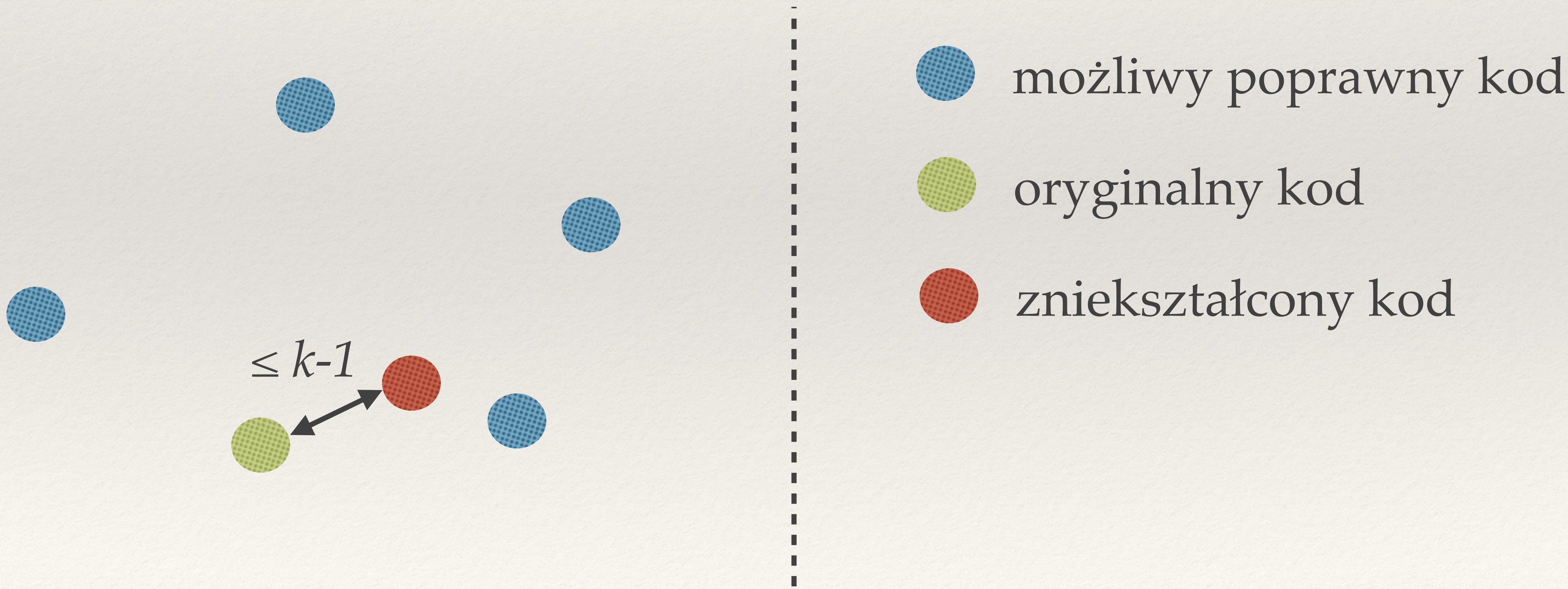
- potrafimy wykryć do *k-1* błędów pojedynczych bitów,
- potrafimy skorygować do *(k-1)*/2 błędów pojedynczych bitów.



Odległość Hamminga

Jeśli mamy kodowanie gwarantujące, że odległość Hamminga między dowolną parą kodów to co najmniej k:

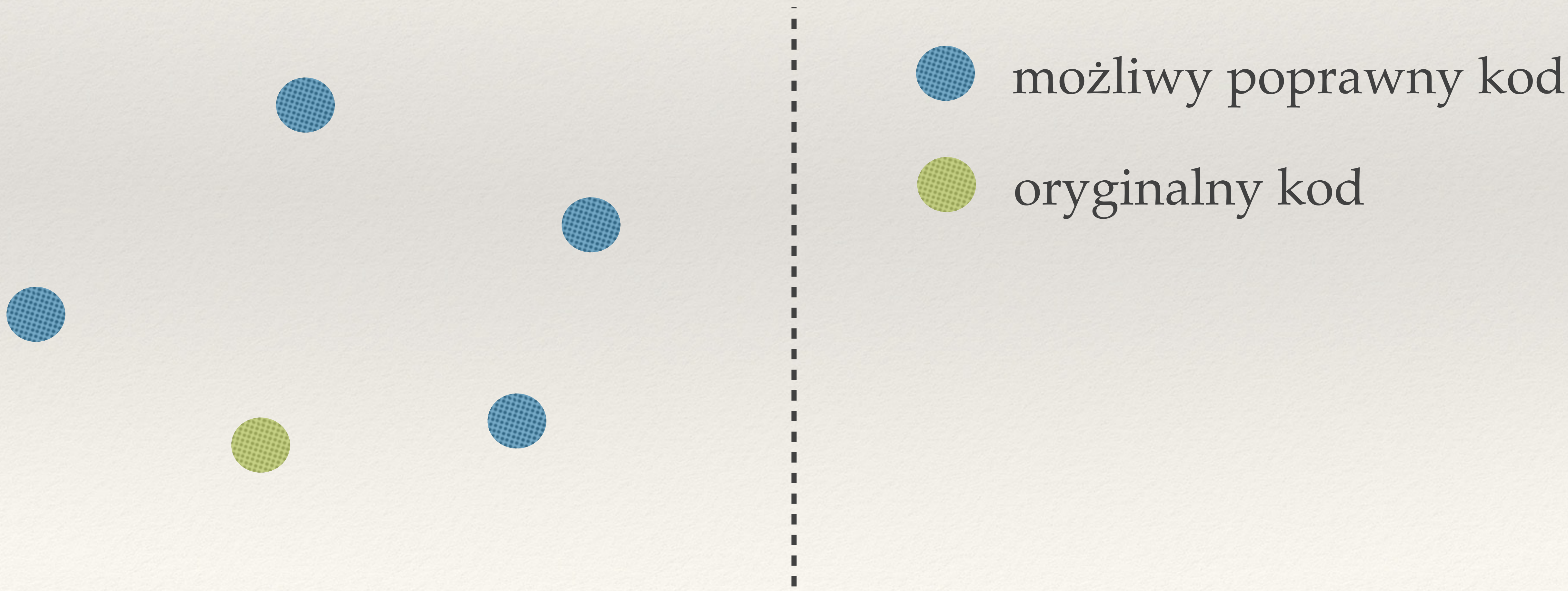
- **potrafimy wykryć do *k-1* błędów pojedynczych bitów,**
- potrafimy skorygować do *(k-1)*/2 błędów pojedynczych bitów.



Odległość Hamminga

Jeśli mamy kodowanie gwarantujące, że odległość Hamminga między dowolną parą kodów to co najmniej k:

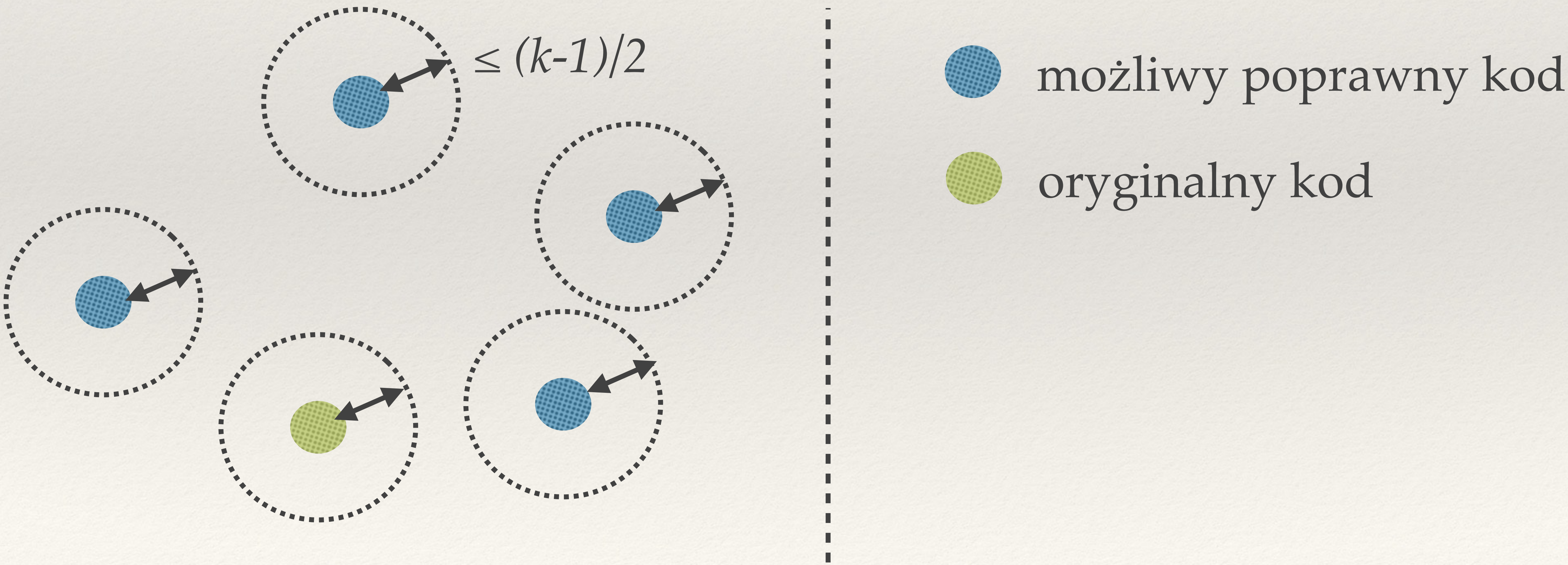
- **potrafimy wykryć do *k-1* błędów pojedynczych bitów,**
- potrafimy skorygować do *(k-1)*/2 błędów pojedynczych bitów.



Odległość Hamminga

Jeśli mamy kodowanie gwarantujące, że odległość Hamminga między dowolną parą kodów to co najmniej k:

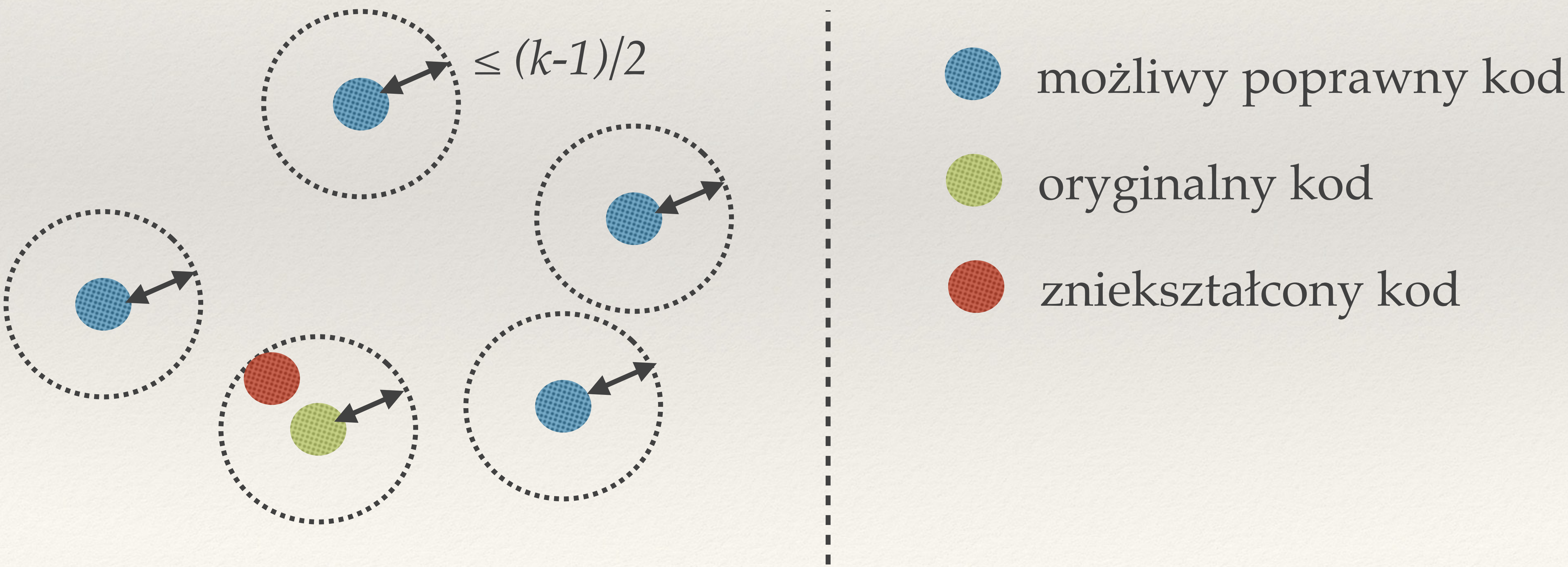
- potrafimy wykryć do *k-1* błędów pojedynczych bitów,
- **potrafimy skorygować do *(k-1)*/2 błędów pojedynczych bitów.**



Odległość Hamminga

Jeśli mamy kodowanie gwarantujące, że odległość Hamminga między dowolną parą kodów to co najmniej k:

- potrafimy wykryć do *k-1* błędów pojedynczych bitów,
- **potrafimy skorygować do *(k-1)*/2 błędów pojedynczych bitów.**



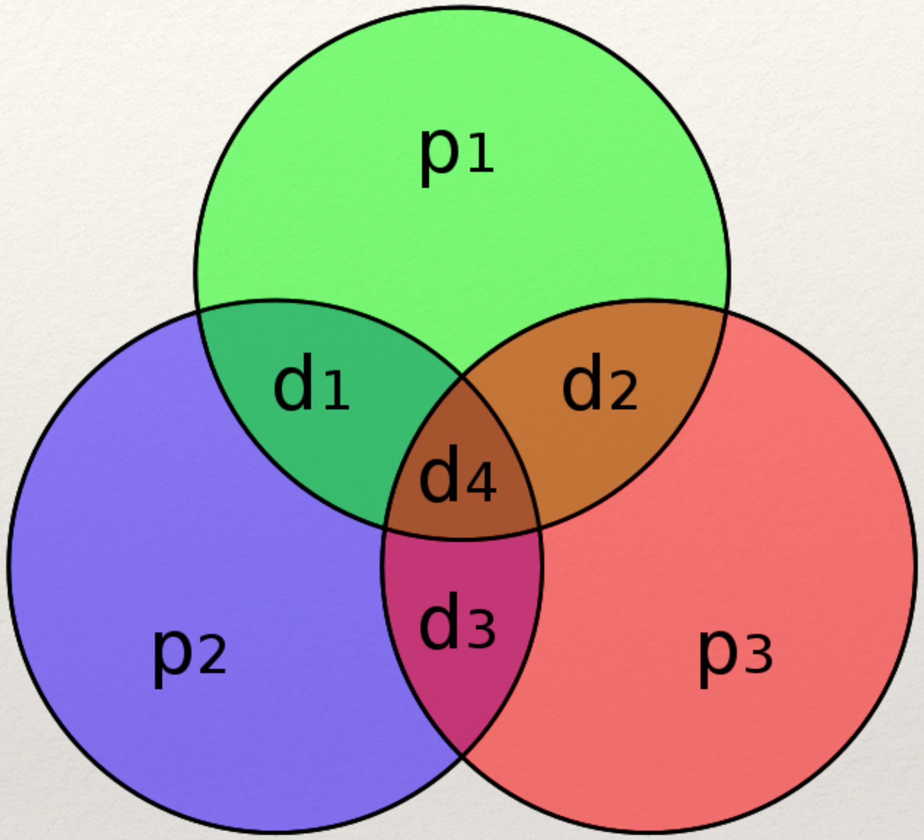
Przykład: (3,1)-kod

Kod o odległości Hamminga ≥ 3

- Wykrywa przekłamanie 2 bitów.
- Koryguje przekłamanie 1 bitu.
- Jak zrobić taki kod?
  - Naiwny pomysł: *(3,1)*-kod, gdzie każdy bit powtarzamy 3 razy.
  - Czy da się lepiej?

Przykład: kodowanie Hamminga (7,4)

- 4 bity danych: d_1, d_2, d_3, d_4
- 3 bity parzystości: p_1, p_2, p_3
(każdy dla innych 3 bitów danych).
- Odległość Hamminga między dowolnymi dwoma kodami ≥ 3 *(ćwiczenie)*.
- Znacznie wyższa efektywność niż naiwny (3,1)-kod.



Obrazek ze strony
`https://en.wikipedia.org/wiki/Hamming(7,4)}`

Szyfrowanie

Bezpieczna komunikacja

Cechy bezpiecznej komunikacji:

- **Poufność** = tylko Alicja i Bob wiedzą, co jest przesyłane.
- **Uwierzytelnianie** = potwierdzanie tożsamości partnera.
- **Integralność** = wykrywanie (złośliwych) zmian w przesyłanej wiadomości.

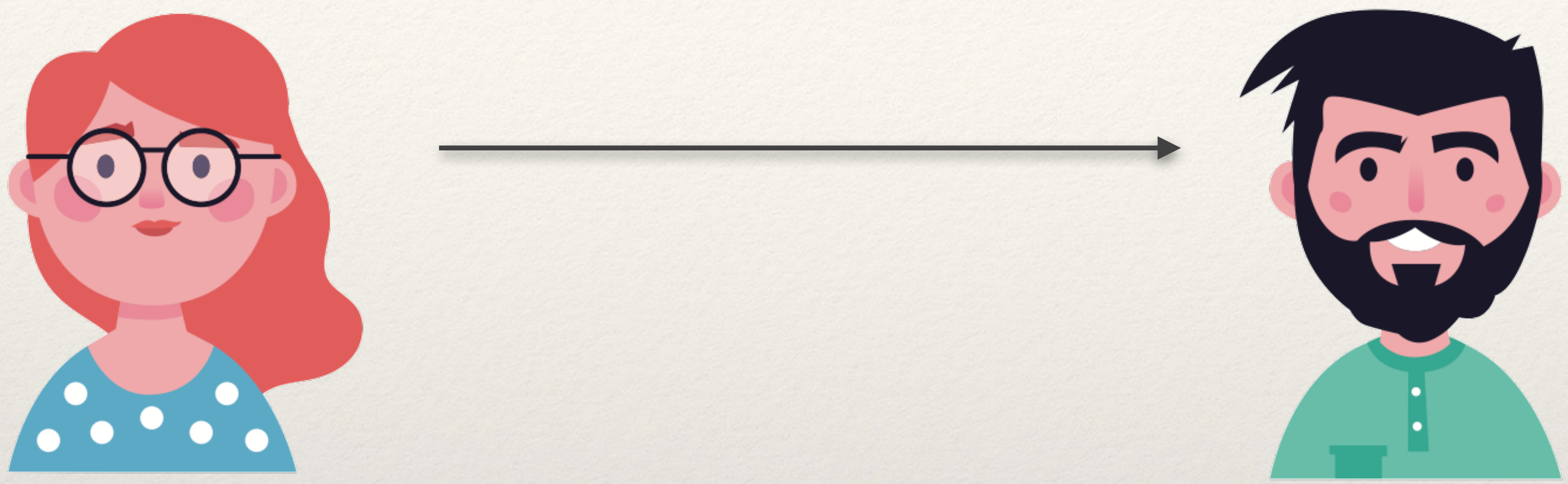
People and hand vectors created by www.freepik.com

Alicja i Bob

Alicja i Bob mogą reprezentować:

- komunikację między dwoma osobami,
- komunikację między fizyczną osobą a usługą (serwerem, bankiem),
- komunikację między dwiema usługami (np. wymiana tablic routingu między routerami).

Szyfrowanie → poufność



Alicja

zna funkcję szyfrującą E

Bob

zna funkcję deszyfrującą $D = E^{-1}$

Szyfrowanie → poufność

Alicja

zna funkcję szyfrującą E

ma tekst jawny m

Bob

zna funkcję deszyfrującą $D = E^{-1}$

Szyfrowanie → poufność

Alicja

zna funkcję szyfrującą E

ma tekst jawny m

szyfrogram

$s = E(m)$

Bob

zna funkcję deszyfrującą $D = E^{-1}$

Szyfrowanie → poufność

Alicja

zna funkcję szyfrującą E

ma tekst jawny m

szyfrogram

$s = E(m)$

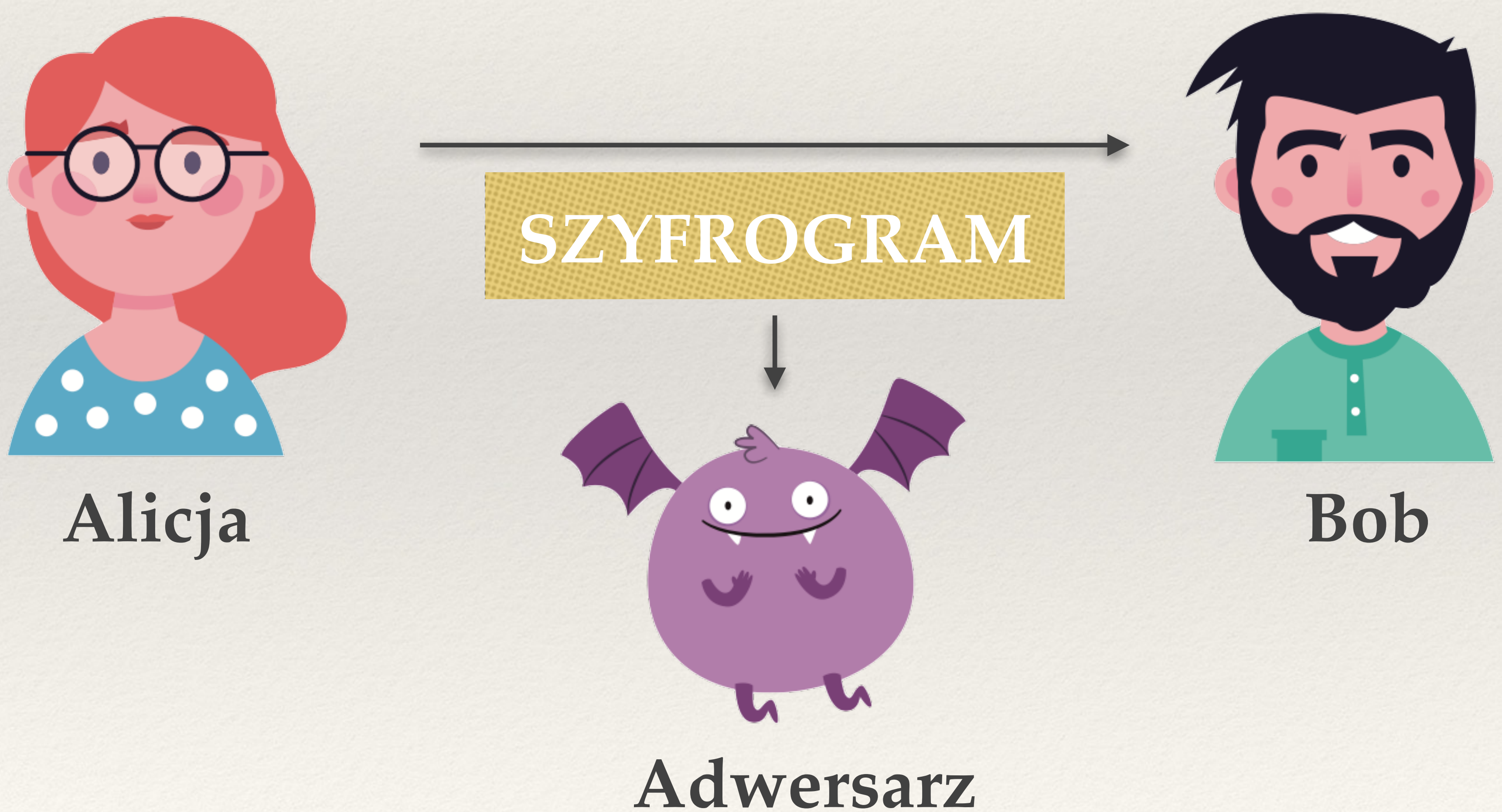
Bob

zna funkcję deszyfrującą $D = E^{-1}$

oblicza $D(s) = E^{-1}(E(m)) = m$

Szyfry monoalfabetyczne (podstawieniowe)

- Funkcja E operuje na pojedynczych literach, przykładowo E zmienia literę `a` na `r`, b na x, …
- Szyfr Cezara: $E(a) = (a + 3) \bmod 26$, ROT-13: $E(a) = (a + 13) \bmod 26$.
- Czy taki szyfr jest bezpieczny?



Typy ataków

- **Atak z wybranym tekstem jawnym:** jeśli adwersarz potrafi zmusić Alicję do wybrania określonego tekstu jawnego (np. "`pchnąć w tę łódź jeża lub ośm skrzyń fig`".
- **Atak ze znanym tekstem jawnym:** jeśli adwersarz potrafi podglądnąć kilka par (*tekst jawny, szyfrogram*).
- **Atak ze znanym szyfrogramem:** jeśli adwersarz widzi tylko szyfrogramy → analiza statystyczna.

People and hand vectors created by `www.freepik.com`

Algorytm i sekret

Szyfrowanie z tajnym algorytmem:

- Bezpieczeństwo szyfru opiera się (również) na tajności algorytmu szyfrowania.

Szyfrowanie z jawnym algorytmem i sekretem (kluczem)

- Algorytm jest jawny
- Tajny jest pewien ciąg bitów (sekret / klucz)
- Analiza poprawności i bezpieczeństwa może być robiona publicznie.

Szyfrowanie symetryczne

- Dany jest pewny publiczny algorytm E (np. DES, 3DES, AES, …) parametryzowany kluczem K.
- Dany jest algorytm deszyfrujący D parametryzowany kluczem K, taki że $D_K(E_K(m)) = m$ dla każdego tekstu jawnego m i klucza K.
- Alicja i Bob ustalają pewien wspólny klucz K.

szyfrogram
$s = E_K(m)$

zna klucz K i tekst jawny m

zna K
oblicza $D_K(s) = D_K(E_K(m)) = m$

Przykład szyfrowania symetrycznego: One-Time Pad

One-Time Pad:

- $E_K(m) = m$ xor K
  (klucz musi być tak samo długi, jak tekst jawny).
- $E_K = D_K$

Bezpieczeństwo:

- Znając szyfrogram *s* ale nie znając *K*, nie dostajemy *żadnej* informacji poza długością tekstu jawnego *m*.
- Ale: znając dowolną parę *(m, s)* możemy obliczyć klucz *K*.
  - Co to w ogóle znaczy, że szyfrowanie jest bezpieczne?

Szyfrowanie symetryczne, cd.

- Algorytm *E* to zazwyczaj złożenie wielu odwracalnych operacji bitowych (xor z częściami klucza, przesunięcia itp.)
- Algorytm *D* to te odwrotności tych operacji wykonane w odwrotnej kolejności.
- Funkcje *E* i *D* są szybko obliczalne.
- Siła kryptograficzna algorytmu zależy głównie od długości klucza (np. 56 bitów w przypadku DES, 128-256 dla AES).

Długość klucza vs. długość wiadomości

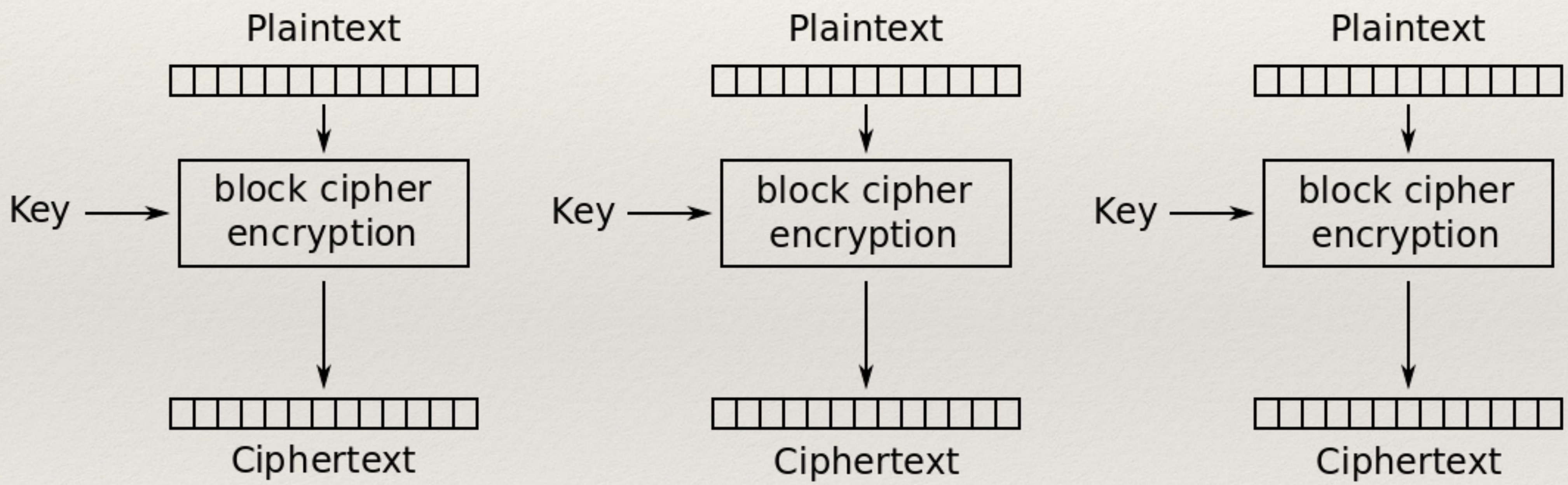
- Algorytm szyfrowania symetrycznego zazwyczaj zakłada, że szyfrowana wiadomość ma określoną długość (DES: 64 bity, AES: 128 bitów).

- Wiadomość dzielona na bloki takiego rozmiaru.
  - Ostatni kawałek wiadomości: dopełniany do długości bloku.
  - Potencjalny problem: jak rozpoznać gdzie zaczyna się wypełnienie?

Wiele bloków (ECB)

ECB (Electronic codebook):

- Każdy blok szyfrowany niezależnie (tym samym kluczem).
- **Problem:** Takie same bloki → takie same kawałki szyfrogramu.

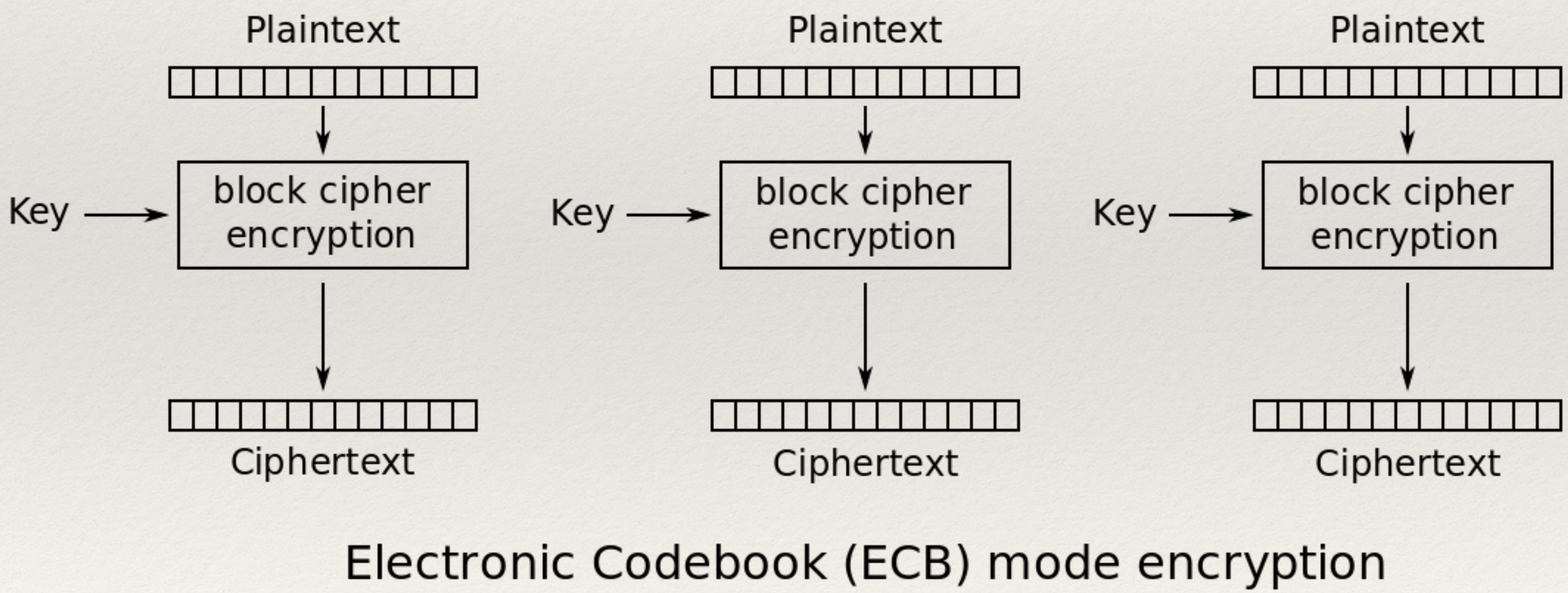


Electronic Codebook (ECB) mode encryption

Obrazek ze strony `https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation`

Wiele bloków (ECB + losowość)

- Każdy blok szyfrowany niezależnie (tym samym kluczem):
- Przed zaszyfrowaniem bloku m_i wylosuj r_i (takie że $|r_i| = |m_i|$).
- Szyfrogram (i-ty blok) $s_i = E_K(m_i \text{ xor } r_i)$.
- Wyślij szyfrogram i wszystkie r_i.
- **Problem:** dwukrotne zwiększenie wysyłanej wiadomości.

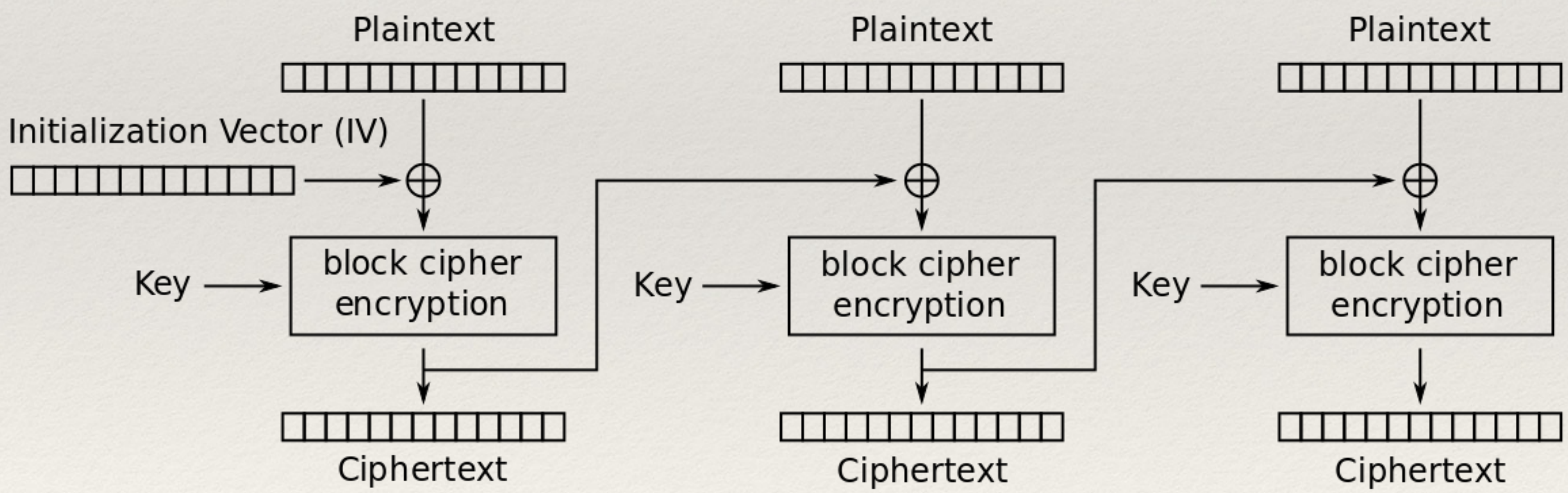


Obrazek ze strony `https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation`

Wiele bloków (CBC)

CBC (Cipher block chaining)

- Wylosuj r_1 = IV = wektor inicjujący.
- Pierwszy blok szyfrogramu jak poprzednio: $s_1 = E_K(\mathrm{m}_1 \text{ xor } r_1)$.
- Kolejne bloki szyfrogramu $s_i = E_K(\mathrm{m}_1 \text{ xor } s_{i-1})$.
- Wyślij szyfrogram i IV



Cipher Block Chaining (CBC) mode encryption

Obrazek ze strony `https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation`

Szyfrowanie symetryczne, cd.

- **Główny problem:** jak ustalić wspólny klucz K?
- Można przesłać innym kanałem (*zabezpieczonym*).
  - Zazwyczaj niepraktyczne lub/i drogie.
- **Inne podejście: szyfrowanie asymetryczne** do przesyłania klucza lub całej wiadomości (*na przyszłym wykładzie*).

Lektura dodatkowa

- Kurose & Ross: rozdział 8.
- Tanenbaum: rozdział 8.

Zagadnienia

- Jakie znasz typy kodów detekcyjnych? Do czego służą i jakie są między nimi różnice?
- Jakie rodzaje błędów mają wykrywać kody detekcyjne? Z czego biorą się błędy przy przesyłaniu danych?
- Jak działa algorytm obliczania sum kontrolnych CRC?
- W jaki sposób działa wykrywanie błędów przy sumie kontrolnej CRC?
- Do czego służą kody MAC? Co to jest HMAC?
- Jakie własności powinna mieć kryptograficzna funkcja skrótu?
- Jakie znasz metody korygowania błędów w transmisji?
- Co to jest (*a*,*b*)-kod? Podaj przykład.
- Co to jest odległość Hamminga? Jak wpływa na możliwość detekcji i korekcji błędów?
- Czym różni się poufność od integralności?
- Co to są szyfry monoalfabetyczne? Dlaczego łatwo je złamać?
- Na czym polegają ataki z wybranym tekstem jawnym, znanym tekstem jawnym i znanym szyfrogramem?
- Czym szyfrowanie symetryczne różni się od asymetrycznego?
- Co to jest szyfrowanie one-time pad?
- Na czym polega szyfrowanie blokowe? Czym różni się tryb ECB od CBC?