Sieci komputerowe

Wykład 10: Kodowanie i szyfrowanie

Marcin Bieńkowski

10.1 Suma kontrolna CRC

10.1.1 Wielomiany

Sumy kontrolne CRC bazują na dzieleniu w pierścieniu wielomianów nad ciałem $\mathcal{F}_2$ (zbiór $\{0,1\}$ z działaniami modulo 2). W prostszych słowach podstawą CRC są działania wykonywane na wielomianach, których współczynniki są ze zbioru $\{0,1\}$, a działania na tych współczynnikach są wykonywane modulo 2. Poniżej przedstawimy krótkie przypomnienie własności takich wielomianów.

Weźmy przykładowo $A(x) = x^5 + x^3 + x^2 + 1$ i $B(x) = x^3 + x$. Ich sumą jest $A(x) + B(x) = x^5 + x^2 + x + 1$. Zauważmy, ze $A(x) + A(x) \equiv 0$ a zatem $A(x) = -A(x)$ i dlatego odejmowanie jest tym samym co dodawanie: $B(x) - A(x) = B(x) + A(x)$. Wielomiany można też mnożyć (tak jak mnożymy „zwykłe" wielomiany nad ciałem liczb rzeczywistych) pamiętając, że współczynniki są z ciała $\mathcal{F}_2$. Czyli przykładowo $(x+1) \cdot (x+1) = x^2 + x + x + 1 = x^2 + 1$.

Można też wielomiany dzielić (np. pisemnie) z resztą. Weźmy dwa dowolne wielomiany $A(x)$, $B(x) \not\equiv 0$ i niech k będzie stopniem wielomianu $B(x)$. Wtedy istnieje dokładnie jedna para wielomianów $Q(x)$ i $R(x)$, taka że

$$A(x) = Q(x) \cdot B(x) + R(x) \ ,$$

gdzie $R(x)$ jest wielomianem stopnia co najwyżej $k-1$. Przykładowo jeśli $A(x) = x^{10} + x^8 + x^3$ a $B(x) = x^3 + x^2 + 1$, to $Q(x) = x^7 + x^6 + x^4 + x$ a $R(x) = x$.

10.1.2 Liczenie CRC

Aby obliczyć sumę CRC zamieniamy ciąg bitów przesyłanej wiadomości $\overline{m}$ na wielomian $M(x)$, tzn. przykładowo $10100001 \to x^7 + x^5 + 1$. Załóżmy, że chcemy wygenerować r-bitową sumę kontrolną. Będziemy w tym celu potrzebowali wielomianu $G(x)$ stopnia r (znanego nadawcy i odbiorcy)[1] o stopniu r. Naszym celem jest utworzenie takiej r-bitowej sumy kontrolnej $\overline{s}$, żeby wielomian odpowiadający ciągowi bitów $\overline{ms}$ był podzielny przez $G(x)$. Następnie wysyłamy wiadomość $\overline{ms}$ i odbiorca sprawdza, czy jest ona podzielna przez $G(x)$. Jeśli nie jest, to w transmisji musiało wystąpić przekłamanie. Jeśli wiadomość jest podzielna, to istnieje nadal mała szansa, że dane zostały przekłamane, lecz zakładamy, że zostały przesłane poprawnie.

Jak obliczyć $\overline{s}$? Niech $S(x)$ będzie wielomianem (stopnia $r-1$) odpowiadającym ciągowi bitów $\overline{s}$. Wtedy ciągowi $\overline{ms}$ odpowiada $x^r \cdot M(x) + S(x)$. Przypomnijmy, że wymagamy, żeby

$$G(x) \mid x^r \cdot M(x) + S(x) \tag{1}$$

Niech $R(x)$ będzie resztą z dzielenia wielomianu $x^r \cdot M(x)$ przez $G(x)$, tj. istnieje taki wielomian $Q(x)$, że $x^r \cdot M(x) = Q(x) \cdot G(x) + R(x)$, gdzie $R(x)$ jest wielomianem stopnia co najwyżej $r-1$. Używając tych oznaczeń wymagamy, żeby $G(x) | Q(x) \cdot G(x) + R(x) + S(x)$, co jest równoważne $G(x) | R(x) + S(x)$. Zauważmy, że $R(x) + S(x)$ jest wielomianem stopnia co najwyżej $r-1$, a zatem $G(x)$ może go dzielić tylko wtedy, jeśli jest on tożsamościowo równy zeru, tj. jeśli $S(x) = R(x)$.

Przykładowo jeśli chcemy wysłać wiadomość $\overline{m} = 10100001 \to x^7 + x^5 + 1$, a wielomianem CRC jest $x^3 + x^2 + 1$, to suma kontrolna będzie miała 3 bity, tj. $r = 3$. Dlatego $x^r \cdot M(x) = x^{10} + x^8 + x^3 = (x^7 + x^6 + x^4 + x) \cdot G(x) + x$. Reszta z dzielenia, $S(x) = x$, odpowiada 3-bitowemu ciągowi $\overline{s} = 010$, który zostaje dołączony na końcu wysyłanej wiadomości.

[1] W rzeczywistych zastosowaniach te wielomiany określa standard. Na przykład w Ethernecie mamy 32-bitową sumę (CRC-32), której wielomian to $G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$.

10.1.3 Wykrywanie błędów

Jak wspomnieliśmy wyżej, odbiorca dzieli wielomian odpowiadający otrzymanej wiadomości przez $G(x)$ i w ten sposób sprawdza, czy wiadomość została zakłócona. To jakie typy zakłóceń zostają wychwycone zależy od wyboru wielomianu $G(x)$.

Załóżmy, że oryginalnie przesyłaną wiadomością (wraz z sumą CRC) jest $\bar{t}$, czyli $G(x)|T(x)$. W trakcie transmisji zostają zmienione wartości niektórych bitów, co odpowiada dodaniu do $T(x)$ wielomianu $E(x)$ nazywanego *wielomianem błędu*. Odbiorca uzna wiadomość za poprawną jeśli $G(x) \,|\, T(x) + E(x) \Leftrightarrow G(x) \,|\, E(x)$.

Przykład 10.1. *W przypadku przekłamania jednego bitu, $E(x) = x^j$. Błąd ten zostanie wykryty jeśli $G(x)$ jest sumą co najmniej dwóch jednomianów.*

Przykład 10.2. *CRC oparte o wielomian $G(x) = x^2 + x + 1$ wykryje błąd polegający na zamianie pięciu kolejnych bitów.*

Dowód. Załóżmy, że ostatnim bitem na którym wystąpił błąd jest bit nr i; wielomianem błędu jest zatem $E(x) = x^{i+4} + x^{i+3} + x^{i+2} + x^{i+1} + x^i = x^i \cdot (x^4 + x^3 + x^2 + x + 1)$. Musimy pokazać, że dla dowolnego i zachodzi $G(x) \not| E(x)$.

Po pierwsze zauważmy, że $G(x)$ nie dzieli $x^4 + x^3 + x^2 + x + 1$, bo $x^4 + x^3 + x^2 + x + 1 = x^2 \cdot (x^2 + x + 1) + (x + 1)$. Po drugie $G(x) \perp x^i$ ($G(x)$ jest względnie pierwsze z x^i) bo jedynym dzielnikiem pierwszym x^i jest x.[2] Zatem $G(x)$ nie dzieli $x^i \cdot (x^4 + x^3 + x^2 + x + 1)$. ■

10.2 Algorytm szyfrowania RSA

Na początku generujemy dla siebię kluczy (publiczny i prywatny) w następujący sposób.

1. Wybieramy $p \neq q$: duże liczby pierwsze.
2. Obliczamy $n = p \cdot q$.
3. Znajdujemy dużą liczbę d względnie pierwszą z $(p-1) \cdot (q-1)$.
4. Znajdujemy takie e, że $d \cdot e \mod (p-1) \cdot (q-1) = 1$ (za pomocą rozszerzonego algorytmu Euklidesa).
5. Para (e, n) to nasz klucz publiczny, a (d, n) to nasz klucz prywatny.

Jak teraz szyfrujemy daną wiadomość? Zapisujemy ją bitowo i dzielimy na kawałki, których długość jest nie większa od $\log n$. Dzięki temu, każdy z kawałków jest liczbą z zakresu $[0, n)$. Każdą z liczb będziemy szyfrować osobno.[3]

Załóżmy zatem, że chcemy zaszyfrować liczbę $m \in [0, n)$. Obliczamy liczbę

$$E(m) = m^e \mod n \ ,$$

i wysyłamy ją jako szyfrogram s odbiorcy. Odbiorca otrzymuje szyfrogram s i odszyfrowuje go obliczając

$$D(s) = s^d \mod n \ .$$

[2] Nie wystarczy powiedzieć, że $G(x) \not| x^i$, bo z $A \not| B$ i $A \not| C$ nie wynika $A \not| (B \cdot C)$

[3] Takie naiwne podejście prowadzi do tego, że takie same kawałki byłyby szyfrowane w ten sam sposób. W praktyce stosuje się różne obejścia tego problemu, np. dołączanie losowego ciągu.

10.2.1 Dlaczego to działa?

Musimy pokazać, że dla dowolnego $m \in [0, n)$ zachodzi $D(E(m)) = m$. Pokażemy to dla m większych od zera i względnie pierwszych z n. Udowodnienie tego dla pozostałych wartości m pozostawiam jako ćwiczenie. Mamy

$$\begin{aligned}
D(E(m)) &= (m^e \mod n)^d \mod n \\
&= (m^e)^d \mod n && \text{(z własności modulo)} \\
&= m^{k\cdot(p-1)\cdot(q-1)+1} \mod n && (k \in \mathbb{N} \cup \{0\}) \\
&= 1^k \cdot m^1 \mod n && \text{(z Twierdzenia Eulera)} \\
&= m
\end{aligned}$$

Twierdzenie 10.3 (Twierdzenie Eulera)**.** *Dla dowolnej dodatniej liczby naturalnej n, niech $\mathbb{Z}_n^* = (\{a : 1 \leqslant a \leqslant n \wedge a \perp n\}, \cdot \mod n)$ będzie grupą, której elementami są liczby względnie pierwsze z n, zaś działaniem mnożenie modulo n. Niech $\phi(n)$ będzie liczbą elementów takiej grupy. Wtedy dla $m \in \mathbb{Z}_n^*$ zachodzi $m^{\phi(n)} \equiv 1 \mod n$.*