Algorytmiczne aspekty kryptografii

Notatki do wykładu

Damian Niwiński

Luty 2007

Przez wiele stuleci kryptografia dotyczyła jednego zagadnienia: szyfrowania tekstów w celu bezpiecznego przesyłania i z drugiej strony łamania szyfrów. Współcześnie (od lat 1970), w związku z rozwojem komunikacji elektronicznej, kryptografia objęła szereg innych problemów, jak podpisywanie dokumentów, uwierzytelnienie, identyfikacja, wybory elektroniczne, pieniądze elektroniczne itd.; lista ta będzie się zapewne wciąż wydłużać.

Do realizacji tych celów służą *protokoły kryptograficzne*, które z kolei oparte są na algorytmach, czy ogólniej zjawiskach algorytmicznych, odkrytych stosunkowo niedawno w związku z rozwojem teorii złożoności, jak *funkcje jedno-kierunkowe, silne generatory pseudolosowe, dowody z zerową wiedzą* i in. Zjawiska te są ciekawe same w sobie i można je studiowaać niezaleznie od potencjalnych zastosowań. Znammieny jest podział przyjęty przez Odeda Goldericha w dziele *Foundations of Cryptography*: *Volume I Basic Tools* [1] i *Volume II Basic Applications* [2]. Jednak większość wykładów wprowadza narzędzia, ukazując jednocześnie ich zastosowania i na odwrót, pokazuje, jak problemy praktyczne stymulują rozwój algorytmiki. My również pójdziemy tą drogą, nie zapominając jednak o tym podstawowym rozróżnieniu – na poziom pierwotnych pojęć algorytmicznych i poziom protokołów. Tym bardziej, że zagadnienie bezpieczeństwa – absolutnie centralne w kryptografii – dotyczy obu tych poziomów (a także dalszych, jak implementacja, również w aspekcie fizycznym).

Innym podstawowym rozróżnieniem jest podział na *kryptografię*, która zajmuje się tworzeniem bezpiecznych protokołów i *kryptoanalizę*, która przeciwnie, zajmuje się ich łamaniem. (Obie te dziedziny wzięte razem nazywa się zwykle *kryptologią*[1].) Oczywiście, *per saldo*, kryptoanalityk oddaje przysługę kryptografowi, wykazując, że jego protokół jednak nie był bezpieczny.

Często wykład z kryptografii rozpoczyna się od przeciwstawienia legalnych uczestników protokołu – zwanych zwykle Alicją i Bobem oraz Ewy (*Eve*, od ang. *enemy*), określanej jako przeciwnik, nieprzyjaciel, intruz itp. Nie powinniśmy jednak zbytnio przywiązywać się do moralnych etykietek nadawanych poszczególnym charakterom, nie zapominając, że w największym historycznie sukcesie polskiej kryptologii, jakim było niewątpliwie rozszyfrowanie Enigmy (w 1933 r.), polscy matematycy występowali właśnie jako Ewa :)

[1] Jakkolwiek wiele osób używa w tym znaczeniu terminu *kryptografia*. To niegroźne zamieszanie terminologiczne dotyczy zarówno języka polskiego jak i określeń *cryptography* i *cryptology* w języku angielskim.

1 Szyfrowanie

1.1 Doskonała tajność

System szyfrujący (lub kryptosystem) jest układem obejmującym skończone zbiory
$\mathcal{M}$ — *wiadomości*,
$\mathcal{K}$ — *kluczy*,
$\mathcal{C}$ — *szyfrogramów*,
funkcję *szyfrowania*

$$E : \mathcal{M} \times \mathcal{K} \to \mathcal{C}$$

oraz funkcję *deszyfrowania*

$$D : \mathcal{C} \times \mathcal{K} \to \mathcal{M}$$

spełniające warunek:

$$(\forall k_1)\,(\exists k_2)\,(\forall m)\; D(E(m, k_1), k_2) = m. \tag{1}$$

Warunek (1) implikuje, że

$$(\forall k)\;\text{ funkcja } E(\cdot, k) : m \mapsto E(m, k) \text{ jest różnowartościową funkcją z } \mathcal{M} \text{ w } \mathcal{C}. \tag{2}$$

Na odwrót, jeśli funkcja E spełnia (2), to zawsze można określić funkcję deszyfrującą D', na przykład

$$D'(c, k) = \begin{cases} m & \text{jeśli } c = E(m, k) \quad \text{dla pewnego } m \\ \text{jakkolwiek} & \text{w pp.} \end{cases}$$

Funkcja ta spełnia warunek mocniejszy niż (1), mianowicie

$$(\forall k)\,(\forall m)\; D'(E(m, k), k) = m. \tag{3}$$

(w (1) można dodatkowo założyć, że $k_2 = k_1$). Taki kryptosystem nazywamy *symetrycznym*: wiadomość zaszyfrowaną kluczem k odszyfrowujemy tym samym kluczem k.

Wykazaliśmy więc, że każdy system można „poprawić" do symetrycznego, modyfikując co najwyżej funkcję D.

Jednak to, co jest banalne z punktu widzenia teorii mnogości, nie musi takie być z punktu widzenia algorytmicznego. Co więcej, czasem może być celowe stosowanie systemu *asymetrycznego*, tzn. spełniającego (1), ale nie (3). W istocie idea ta otworzyła drogę do tzw. kryptografii klucza publicznego, o której będziemy mówić w dalszych wykładach. Na razie ograniczymy się jednak do systemów symetrycznych.

Kiedy system szyfrujący uznamy za bezpieczny?

W pierwszym przybliżeniu, jeśli *na podstawie szyfrogramu nie można powiedzieć nic sensownego o wiadomości*. Stwierdzenie „nie można nic powiedzieć" wydaje się trudne do zmatematyzowania, dlatego ujmiemy to trochę inaczej: *na podstawie szyfrogramu można o wiadomości powiedzieć dokładnie tyle, co bez znajmości szyfogramu*.

Przykład Załóżmy, że wiadomościami są słowa nad alfabetem $\Sigma = \{a, b, \ldots, z\}$, a kluczami permutacje $\varphi : \Sigma \to \Sigma$. Załóżmy, że szyfrowanie polega na permutacji liter zgodnie z kluczem:

$$E(w_1 w_2 \ldots w_n, \varphi) = \varphi(w_1)\varphi(w_2)\ldots\varphi(w_n).$$

Oczywiście, deszyfrowanie polega na zastosowaniu funkcji odwrotnej:

$$D(v_1 v_2 \ldots v_n, \varphi) = \varphi^{-1}(v_1)\varphi^{-1}(v_2)\ldots\varphi^{-1}(v_n).$$

Jeżeli wiadomości są tekstami w języku naturalnym (np. polskim), to takie szyfrowanie jest podatne na atak wykorzystujący różne częstotliwości występowania poszczególnych liter. Jest to zagadnienie ogólnie znane i nie będziemy go tutaj omawiać.

Przypuśćmy jednak, że wiadomościami mogą być dowolne słowa ustalonej długości N, każde z tym samym prawdopodobieństwem (możemy mysleć, że wiadomościami są np. liczby w systemie $|\Sigma|$-arnym). Czy wtedy nasz system jest bezpieczny?

Szyfrogram $v_1 v_2 \ldots v_N$ może pochodzić z $|\Sigma|!$ wiadomości. Jeśli tylko $|\Sigma|! \leq |\Sigma|^N$, to jest to właściwy podzbiór $\mathcal{M}$, tak więc znajomość $E(w, \varphi)$, nawet bez znajomości klucza φ, daje nam nietrywialną informację o w.

Określimy teraz postulat doskonałej tajności, a następnie udowodnimy warunek konieczny, pozwalający łatwo identyfikować systemy, które nie są doskonale tajne.

Rozważmy zmienne losowe

M o wartościach w $\mathcal{M}$,

K o wartościach w $\mathcal{K}$.

Zakładamy ponadto, że M i K są niezależne oraz K ma rozkład jednorodny, tzn. $p(K = k) = \frac{1}{|\mathcal{K}|}$, dla każdego k. Okreslmy teraz zmienna losową C o wartościach w $\mathcal{C}$,

$$C = E(M, K).$$

System nazywamy *doskonale tajnym*, jeśli przy powyższych założeniach zmienne C i M są niezależne, innymi słowy

$$p(M = m | C = c) = p(M = m), \tag{4}$$

o ile tylko powyższe prawdopodobieństwo warunkowe jest określone (tzn. $p(C = c) > 0$).

Intuicja mówi nam, że choć funkcja szyfrowania przy ustalonym kluczu jest 1:1, to poza tym musi sporo „mieszać". W skrajnym przypadku, gdyby E była funkcją różnowartościową, to wiadomość m można by odtworzyć z szyfrogramu $c = E(m, k)$, nawet bez znajmości klucza. A zatem szyfrogramów nie może być „za wiele" w porównaniu z liczbą kluczy.

Twierdzenie 1 *W systemie doskonale tajnym*

$$|\mathcal{M}| \leq |\mathcal{C}| \leq |\mathcal{K}|. \tag{5}$$

Dowód. Pierwsza z nierówności wynika natychmiast z (2). Z kolei rozważmy takie wartości m i c, że $p(M = m) > 0$ i $p(C = c) > 0$.

Notacja. W dalszym ciągu, o ile nie będzie to prowadziło do nieporozumień, będziemy upraszczać notację dotyczącą zmiennych losowych, pisząc np. $p(m)$ zamiast $p(M = m)$. Podobnie $p(m_1 \wedge c_2|k')$ jest skrótem $p(M = m_1 \wedge C = c_2|K = k')$ itp.

Mamy (tzw. wzór Bayesa)

$$p(m \wedge c) = p(m|c) \cdot p(c) = p(c|m) \cdot p(m).$$

Skoro $p(m|c) = p(m)$ (założenie (4)), to

$$p(c|m) = p(c) \tag{6}$$

Mamy, z definicji prawdopodobieństwa warunkowego,

$$p(c|m) = \frac{p(c \wedge m)}{p(m)},$$

gdzie

$$p(c \wedge m) = \sum_{k:E(m,k)=c} p(k) \cdot p(m) \tag{7}$$

(pamiętamy o niezależności M i K), a zatem

$$p(c) = p(c|m) = \sum_{k:E(m,k)=c} p(k). \tag{8}$$

Powyższa równość zachodzi dla każdego c, dla którego $p(C = c) > 0$ (przy ustalonym m). Zauważmy jednak, że w równaniach odpowiadających różnym c muszą wystąpić różne klucze k (w przeciwnym razie E nie byłoby funkcją). A zatem kluczy jest co najmniej tyle co tych $c \in \mathcal{C}$, dla których $p(C = c) > 0$. Ponieważ dla pewnej zmiennej losowej M ten zbiór jest całym $\mathcal{C}$ (np. dla M o rozkładzie jednorodnym), otrzymujemy nierówność $|\mathcal{C}| \leq |\mathcal{K}|$, jak chcieliśmy. □

Uwaga. Powyższe twierdzenie znane jest jako *Pesymistyczne Twierdzenie Shannona*. Często wyraża się je w języku teorii informacji: warunek niezależności zmiennych M i C oznacza, że ich wzajemna informacja $I(M; K)$ jest równa zeru, a konkluzja twierdzenia implikuje, że *długość klucza* musi być co najmniej taka, jak długość wiadomości (jeśli $\mathcal{M}$ i $\mathcal{K}$ są zbiorami ciągów nad tym samym alfabetem; zobacz
`http://wazniak.mimuw.edu.pl/index.php?title=Teoria_informacji`, Wykład 6).

Na pocieszenie dodajmy od razu, że systemy doskonale bezpieczne rzeczywiście istnieją, a nawet bardzo łatwo jest je skonstruować.

Twierdzenie 2 *Przypuśćmy, że $|\mathcal{M}| = |\mathcal{C}| = |\mathcal{K}|$ i zmienna K ma rozkład jednorodny. Załóżmy dalej, że*

$$(\forall m)\,(\forall c)\,(\exists k)\; c = E(m, k). \tag{9}$$

Wtedy system jest doskonale tajny.

Dowód. Zauważmy najpierw, że (9) można wzmocnić do

$$(\forall m)\,(\forall c)\,(\exists!k)\; c = E(m,k), \tag{10}$$

tzn. można żądać, że odpowiednie k jest jedyne. Istotnie, przy ustalonym m, (9) implikuje istnienie pewnej funkcji $\varphi : c \mapsto k$, takiej, że $c = E(m, \varphi(c))$. Funkcja ta musi być różnowartościowa (w przeciwnym razie E nie byłoby funkcją), a wobec tego że $|\mathcal{C}| = |\mathcal{K}|$, jest także na $\mathcal{K}$. Gdyby więc zachodziło $c = E(m, \varphi(c)) = E(m, k')$, dla jakiegoś $k' \neq \varphi(c)$, to owo k' byłoby wartością $k' = \varphi(c')$, dla jakiegoś $c' \neq c$. I to już sprzeczność, bo założyliśmy właśnie, że $E(m, \varphi(c')) = E(m, \varphi(c))$.

Dalej przechodzimy podobne rozumowanie jak w dowodzie Twierdzenia 1, tylko w odwrotnej kolejności. (7) daje nam

$$p(c|m) = p(\varphi(c)) = \frac{1}{|\mathcal{K}|} \tag{11}$$

a z drugiej strony (z wzoru na pradopodobieństwo całkowite)

$$p(c) = \sum_{m':p(m')>0} p(c|m') \cdot p(m') = \frac{1}{|\mathcal{K}|} \cdot \underbrace{\sum_{m':p(m')>0} p(m')}_{1}. \tag{12}$$

Z równości $p(c|m) = p(c)$ wnioskujemy[2] $p(m|c) = p(m)$, co daje doskonałą tajność systemu. □

Uwaga. Nietrudno jest sprawdzić, że przy założeniu $|\mathcal{M}| = |\mathcal{C}| = |\mathcal{K}|$, warunek (9) jest jednocześnie warunkiem koniecznym dla doskonałej tajności.

Funkcję szyfrującą można ogólnie przedstawić jako macierz $|\mathcal{M}|$ na $|\mathcal{K}|$ o wartościach w $\mathcal{C}$, $E(m,k)$. Warunek różnowartościowości funkcji $E(\cdot,k)$ (dla każdego k) oznacza, że w każdej kolumnie tej macierzy wartości są różne. Jeśli $|\mathcal{M}| = |\mathcal{C}| = |\mathcal{K}| = n$, to macierz ta jest kwadratowa $n \times n$. Warunek (9) oznacza z kolei, że w każdym wierszu tej macierzy muszą pojawić się wszystkie wartości z $\mathcal{C}$, a zatem wszystkie są różne. Oczywiście, dla każdego n można łatwo znaleźć macierz $n \times n$ przyjmującą n wartości i taką, że w każdym wierszu i odpowiednio w każdej kolumnie wszystkie wartości są różne. Na przykład macierz

$$\begin{pmatrix} c_1 & c_2 & \dots & \dots & c_n \\ c_2 & c_3 & \dots & c_n & c_1 \\ \dots & \dots & \dots & \dots & \dots \\ c_n & c_1 & c_2 & \dots & c_{n-1} \end{pmatrix}$$

ma tę własność. Każda taka macierz wyznacza system doskonale tajny.

Z powodów historycznych, a także ze względu na łatwość obliczenia, najważniejszym przykładem jest tzw. *szyfr Vernama*, znany też jako *One-Time-Pad*.

[2]Ściśle biorąc, zakładamy przy tym, że $p(c), p(m) > 0$, ale jeśli jedno z tych prawdopodobieństw jest 0, wtedy równość $p(c \wedge m) = p(c) \cdot p(m)$ zachodzi trywialnie.

W systemie tym $\mathcal{M} = \mathcal{K} = \mathcal{C} = \{0,1\}^n$, dla pewnego $n \in \mathbb{N}$ i

$$E(m,k) \quad = \quad m \oplus k,$$

gdzie $\oplus$ jest operacją *xor* (dodawania *modulo* 2) po współrzędnych (np. $101101 \oplus 110110 = 011011$). Jako funkcję deszyfrującą (symetryczną) można przyjąć $D = E$, mamy bowiem

$$(m \oplus k) \oplus k = m.$$

One-Time-Pad spełnia oczywiście założenia Twierdzenia 2, a zatem jest szyfrem doskonale tajnym.

Sposób, w jaki określiliśmy *One-Time-Pad*, stosując po współrzędnych funkcję $\oplus$, można zresztą uogolnić. Zauważmy, że jeśli $\mathcal{M}_i, \mathcal{K}_i, \mathcal{M}_i, E_i$, wyznaczają kryptosystem spełniający założenia Twierdzenia 2, dla $i = 1, \ldots, p$, to spelnia je również system wyznaczony przez

$$\prod_{i=1}^{p} \mathcal{M}_i = \prod_{i=1}^{p} \mathcal{K}_i = \prod_{i=1}^{p} \mathcal{C}_i$$

i funkcję E

$$E(m_1 \ldots m_p, k_1 \ldots k_p) \quad = \quad (E_1(m_1,k_1), \ldots, E_p(m_p,k_p)) \tag{13}$$

Ćwiczenie. Dla zmiennej losowej X o wartościach w $\mathcal{X}$ przyjmijmy oznaczenie $Im(X) = \{x \in \mathcal{X} : p(X = x) > 0\}$. Dowieść, że jeśli w systemie doskonale tajnym $Im(C) = |\mathcal{K}|$, to zmienna C ma rozkład jednostajny (niezależnie od rozkładu M). Pokazać przykład, że w ogólności tak być nie musi.

1.2 Bezpieczeństwo

Jakkolwiek powyżej używaliśmy określenia „doskonała tajność", nie należy zapominać, że jest to tylko skrót dla pewnych związków matematycznych. Powinniśmy być bardzo ostrożni z interpretacją tego pojęcia i w szegółności Twierdzenia 2 jako gwarancji bezpieczeństwa.

W ogólności pojęcie *bezpieczeństwa* nie ma satysfakcjonującej matematycznej definicji i prawdopodobnie nigdy nie będzie zmatematyzowane w takim stopniu jak pojęcie *obliczalności* lub *poprawności* programu. Parafrazując tezę Churcha można jedynie powiedzieć, że nie należy uważać za bezpieczne czegoś, co w intuicyjnym sensie z jakichś powodów takim nie jest.

Zwróćmy uwagę na dwie cechy systemu *one time pad*.

Po pierwsze – jak sama nazwa nas poucza – system ten jest tajny pod warunkiem, że klucz jest używany dokładnie raz. Jeśli ten sam klucz $k \in \{0,1\}^n$ zastosujemy np. dwukrotnie, to w istocie mamy do czynienia z szyfrowaniem

$$E : \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^{2n}$$
$$E(m_1 m_2, k) = (m_1 \oplus k)\ (m_2 \oplus k)$$

(gdzie $|m_1| = |m_2|$). Mamy więc $|\mathcal{C}| = 2^{2n}$, podczas gdy

$$|\mathcal{K}| = 2^n = \sqrt{|\mathcal{C}|},$$

system nie jest więc oczywiście doskonale tajny (Twierdzenie 1).

W istocie, dwukrotne użycie tego samego klucza umożliwia atak ze znanym tekstem jawnym: jeśli znamy m_1 i $c_1 = m_1 \oplus k$, to możemy już obliczyć $k = m_1 \oplus c_1$ i wtedy znając c_2 znajdziemy już $m_2 = c_2 \oplus k$. Ale nawet jeśli nie znamy żadnego tekstu jawnego, sama znajomość $m_1 \oplus k$ i $m_2 \oplus k$ daje nam $m_1 \oplus m_2$. Jak bardzo istotna może być to informacja łatwo jest sobie wyobrazić na przykładzie *kryptografii wizualnej*, gdzie przesyłana wiadomość jest mapą bitową jakiegoś obrazu. Historycznie, słynnym skutkiem powtarzania klucza było rozszyfrowanie przez Amerykanów w latach 1950, w ramach tzw. projektu *Venona*, wiadomości przesyłanych przez szpiegów KGB na temat technologii wytworzenia broni atomowej.

Innym założeniem doskonałej tajności systemu *one time pad*, o wiele trudniejszym do spełnienia niż niepowtarzanie klucza, jest, że klucze są losowane z jednorodnym rozkładem pradwopodobieństwa. W istocie, gdyby klucze produkowane były z wykorzystaniem jakiegoś algorytmu, przeciwnik, nawet bez znajomości wszystkich szczegółów, mógłby z pewnym powodzeniem zawęzić zbiór mozliwych kluczy i w ten sposób nierówność $|\mathcal{C}| \leq |\mathcal{K}|$ przestałaby zachodzić.

Z drugiej strony musimy jednak podkreślić, że sam fakt, że system nie jest doskonale tajny nie daje nam jeszcze do ręki metody jego łamania. W tym miejscu powinniśmy uściślić nieco pojęcie *ataku*. Ogólnie, atak na system szyfrujący jest algorytmem, który na podstawie szyfrogramu $c = E(m, k)$ znajduje wiadomość m, w tym kontekście zwaną też *tekstem jawnym* lub *otwartym* (ang. *plaintext* lub *cleartext*).

Zakłada się przy tym, że konstruktor algorytmu atakującego zna wszystkie elementy kryptosystemu, w tym funkcje szyfrowania E i deszyfrowania D. Jest to tzw.

Zasada Kerkchoffsa Bezpieczeństwo kryptosystemu opiera się jedynie na tajności kluczy [3].

Typologia ataków zależy od tego, czego może on używać jako wejścia: czy jest to tylko c, czy też dodatkowo ma do dyspozycji ciąg szyfrogramów zaszyfrowany tym samym (nieznanym) kluczem, $c_1 = E(m_1, k), \ldots, c_\ell = E(m_\ell, k)$, lub nawet ciąg par $(m_1, c_1), \ldots, (m_\ell, c_\ell)$. W jednej z najsilniejszych wersji, przeciwnik może niejako „używać" funkcji

$$E(\cdot, k) : m \mapsto E(m, k) = c$$

(bez znajmości klucza k!). Jest to tzw. *atak z wybranym tekstem jawnym* (ang. *chosen plaintext attack*), którego historycznym przykładem było przygotowanie do bitwy o Midway (1942): Amerykanie celowo użyli niezaszyfrowanej nazwy geograficznej (*cleartext* = *Midway*), która natychmiast pojawiła się jako szyfrogram w wielu depeszach Japończyków.

Dokładne omowienie typów ataków można znaleźć w [7], Rozdz. 1.1, str. 33–34.

1.3 Szyfry strumieniowe

Jak wspomnieliśmy, jedną z głównych trudności przy stosowaniu szyfru *One time pad* jest długość klucza. Przypuśćmy, że A i B posiadają losowo wygenerowany i uzgodniony wcześniej, bezpiecznie przechowywany wspólny klucz k, który jednak jest krótki (nic nie jest doskonałe). Przypuśćmy dalej, że A chce wysłać wiadomość m, $|k| \ll |m|$. Podzielenie m na bloki długości $|k|$ i szyfrowanie ich tym samym kluczem k jest, jak już zauważyliśmy, metodą bardzo ryzykowną. Idea jest, by „rozciągnąć" k do klucza k', $|k'| = |m|$, ale w sensowniejszy sposób. Oczywiście, jeśli metoda „rozciągania" jest deterministyczna, to Twierdzenie 1 ostrzega nas, że nigdy nie otrzymamy w ten sposób doskonałej tajności. Jednak celem kryptografii nie jest w ogólności doskonałe bezpieczeństwo, lecz raczej rozsądny kompromis pomiędzy bezpieczeństwem a efektywnością.

W szyfrowaniu strumieniowym zwykle nie ustala się z góry długości wiadomości m, lecz podaje metodę, która z krótkiego klucza k – zwanego w tym kontekście zarodkiem lub ziarnem (ang. *seed*) – generuje *a priori* nieskończony ciąg („strumień") bitów. W zależności od potrzeby wybiera się z tego ciągu odpowiednio długi prefiks, by zastosować *One time pad*.

Schematy rekursji liniowej (LFSR)

Omówimy jedną z podstawowych metod w tej dziedzinie, jaką jest rekursja liniowa, identyfikowalna pod skrótem LFSR, od nazwy jej maszynowej reprezentacji: ang. *linear feedback shift register*, czyli *liniowy rejestr przesuwający ze sprzężeniem zwrotnym*[3].

Liniowy schemat rekursji z m współczynnikami, lub krótko m-schemat zadany jest przez m bitów $c_1, \ldots, c_m$. Schemat ten opisuje funkcję liniową nad ciałem $\mathbb{Z}_2$:

$$c(z_1, \ldots, z_m) \;=\; c_1 \cdot z_1 + \ldots + c_m \cdot z_m \tag{14}$$

W dalszym ciągu tego punktu zakładamy, że wszystkie działania są wykonywane w ciele $\mathbb{Z}_2$ (tzn. $\bmod 2$).

Dla dowolnego ciągu bitów $w_0, w_1, \ldots, w_{m-1}$ (zwanego w tym kontekście *ziarnem*), m-schemat wyznacza nieskończony ciąg bitów

$$W \;=\; w_0, w_1, \ldots, w_{m-1}, w_m, w_{m+1}, \ldots$$

gdzie

$$w_m \;=\; c_1 \cdot w_{m-1} + \ldots + c_m \cdot w_0$$

i ogólnie

$$w_t \;=\; c_1 \cdot w_{t-1} + \ldots + c_m \cdot w_{t-m}, \tag{15}$$

dla $t \geq m$.

[3] W tłumaczeniu W. Bartola [8].

Liniowy schemat często przedstawia się przy pomocy diagramu – wspomnianego wyżej LFSR – ilustrującego przepływ informacji przy obliczaniu[4] kolejnego bitu ciągu W.

Zauważmy, że dla $t \geq m$, bit w_t ciągu W jest całkowicie zdeterminowany przez układ poprzedzających go m bitów, a zatem ciąg W jest od pewnego miejsca okresowy, tzn. istnieją m_0 i d, że $(\forall t \geq m_0)\, w_{t+d} = w_t$.

Fakt 1 *Jeśli w m-schemacie $c_m = 1$, to ciąg W jest okresowy z okresem $\leq 2^m - 1$. W ogólności, ciąg W jest okresowy począwszy od $m - k$ z okresem $\leq 2^k - 1$, gdzie k jest największe takie, że $c_k = 1$ (lub 0, gdy nie ma takich c_k).*

Dowód. Załóżmy najpierw $c_m = 1$. Rozważmy ciąg słów skończonych

$$\begin{array}{lcccc} W_0 & = & w_{m-1} & \ldots & w_0 \\ W_1 & = & w_m & \ldots & w_1 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ W_k & = & w_{k+m-1} & \ldots & w_k \\ \ldots & \ldots & \ldots & \ldots & \ldots \end{array}$$

Zauważmy, że słowo W_{k+1} powstaje ze słowa W_k w wyniku transformacji

$$x_{m-1} \ldots x_1 x_0 \quad \longrightarrow \quad c(x_{m-1} \ldots x_1 x_0)\, x_{m-1} \ldots x_1 \tag{16}$$

Niech i będzie najmniejsze takie, że dla pewnego $j > i$, $W_i = W_j$. Wykażemy, że $i = 0$.

Przypuśćmy przeciwnie, wtedy

$$\begin{array}{ccc} W_{i-1} & \longrightarrow & W_i \\ & & \| \\ W_{j-1} & \longrightarrow & W_j \end{array}$$

Z postaci transformacji (16) wynika, że W_{i-1} i W_{j-1} mogą się różnić co najwyżej ostatnim bitem, jednak gdyby tak było, to pierwsze bity W_i i W_j byłyby różne (bo kiedy $c_m = 1$, ostatni bit wpływa na wartość funkcji c). A zatem $W_{i-1} = W_{j-1}$, wbrew minimalności i.

Niech więc j będzie najmniejsze takie, że $W_0 = W_j$. A zatem słowa $W_0, \ldots, W_{j-1}$ są różne. Zauważmy jednak, że gdyby jedno z nich było 0^m, to cały ciąg W składałby się z samych zer, w szczególności byłby okresowy o okresie 1. Jeśli 0^m nie występuje, to $j \leq 2^m - 1$.

A zatem ciąg W_k jest okresowy o okresie $\leq 2^m - 1$ i ciąg W ma również ten okres[5], bo oczywiscie

$$W \quad = \quad \mathrm{last}(W_0)\, \mathrm{last}(W_1)\, \mathrm{last}(W_2) \ldots$$

gdzie $\mathrm{last}(V)$ oznacza ostatni bit słowa V.

Załóżmy teraz, że $c_m = 0$. Jeśli wszystkie c_i są zerami, to $W = w_0 w_1 \ldots w_{m-1} 000 \ldots$, a zatem, począwszy od m, jest okresowy z okresem 1. W przeciwnym razie niech k

[4] Dynamiczną ilustrację można znaleźć na `http://en.wikipedia.org/wiki/Linear_feedback_shift_register`.

[5] Nie wykluczamy w tym miejscu, że może mieć także okresy mniejsze, choć w rzeczywistości nie jest to możliwe.

największe takie, że $c_k = 1$ i niech W' będzie ciągiem generowanym przez k-schemat $c_1, \ldots, c_k$ z ziarna $w_{m-k} w_{m-k+1} \ldots w_{m-1}$. Jak już wiemy, ciąg ten jest okresowy z okresem $\leq 2^k - 1$. Nietrudno sprawdzić, że

$$W \quad = \quad w_0 w_1 \ldots w_{m-k-1} W'.$$

□

Uwaga Wiadomo, że okres $2^m - 1$ jest osiągany i znane jest kryterium pozwalające definiować schematy o maksymalnym okresie. Okazuje się, że własności okresu powiązane są z własnościami wielomianu

$$1 + c_1 \cdot x + c_2 \cdot x^2 + \ldots + c_m \cdot x^m$$

Jeśli wielomian ten jest *pierwotny* tzn. jest nierozkładalny i nie dzieli wielomianu $x^d + 1$ dla żadnego $d < 2^m - 1$, to okres jest maksymalny (zob. [5]).

Oczywiście, dany ciąg W może być generowany przez wiele, niekoniecznie optymalnych, schematów. Z powyższego Faktu możemy jednak wysnuć ogólną zależność.

Wniosek 1 *Ciąg W jest generowany przez pewien m-schemat (LFSR) wtedy i tylko wtedy, gdy jest od pewnego miejsca okresowy.*

Dowód. Jeśli ciąg W i $m_0 \geq 0$ spełniają $(\forall t \geq m_0)\, w_{t+d} = w_t$, to W można wygenerować na przykład z ziarna $w_0 w_1 \ldots w_{m_0-1}$ przez schemat $c_1, \ldots, c_{d+m_0}$, gdzie $c_d = 1$, a pozostałe c_i są zerami. □

Jednak własności kryptograficzne ciągów okresowych są słabe.

Fakt 2 *Jeśli ciąg $W = w_0 w_1 \ldots$ jest generowany przez m-schemat $c_1, \ldots, c_m$, gdzie $c_m = 1$ i nie jest generowany przez żaden k-schemat dla $k < m$, to na podstawie dowolnego podsłowa W długości $2m$ można jednoznacznie wyznaczyć współczynniki $c_1, \ldots, c_m$.*

Dowód. Z Faktu 1 ciąg W jest okresowy, dlatego każdy nieskończony sufiks $w_i w_{i+1} \ldots$ ciągu W ma te same własności, które zakładamy o W. Bez zmniejszenia ogólności wystarczy więc rozważyć początkowy fragment $w_0 w_1 \ldots w_{2m-1}$. Zależność drugiej połowy tego ciągu od pierwszej można zapisać macierzowo

$$\begin{pmatrix} w_m \\ w_{m+1} \\ \ldots \\ w_{2m-1} \end{pmatrix} = \begin{pmatrix} w_{m-1} & w_{m-2} & \ldots & w_0 \\ w_m & w_{m-1} & \ldots & w_1 \\ \ldots & \ldots & \ldots & \ldots \\ w_{2m-2} & w_{2m-3} & \ldots & w_{m-1} \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ c_2 \\ \ldots \\ c_m \end{pmatrix}$$

Dla jednoznacznego wyznaczenia ciągu $c_1, \ldots, c_m$ wystarczy wykazać, że macierz występująca po prawej stronie rownania jest odwracalna (nad $\mathbb{Z}_2$). Przypuśćmy, że jest przeciwnie i oznaczmy jej kolejne wiersze przez $W_0, \ldots, W_{m-1}$. Mamy więc

$$b_0 \cdot W_0 + \ldots + b_{m-1} \cdot W_{m-1} = \mathbf{0}$$

gdzie $b_0, \ldots, b_{m-1}$ nie są wszystkie zerami. Niech k największe takie, że $b_k = 1$. Gdyby $k = 0$ mielibyśmy $W_0 = \mathbf{0}$, a więc ciąg W składałby się wyłącznie z zer i byłby okresowy o okresie 1, skąd $m = 1$, a wiemy, że $c_m = 1$ z założenia. W przeciwnym razie, z własności ciała $\mathbb{Z}_2$, mamy

$$W_k \;=\; \sum_{i=0}^{k-1} b_i \cdot W_i \tag{17}$$

Zauważmy, że $W_0, \ldots, W_{m-1}$ są początkowymi wyrazami ciągu, jaki określiliśmy w dowodzie Faktu 1, wyraz $W_{\ell+1}$ tego ciągu powstaje z W_ℓ w wyniku transformacji (16)

$$x_{m-1} \ldots x_1 x_0 \;\longrightarrow\; c(x_{m-1} \ldots x_1 x_0)\, x_{m-1} \ldots x_1$$

Transformacja ta jest oczywiście przekształceniem liniowym przestrzeni liniowej $\{0,1\}^m$ nad ciałem $\mathbb{Z}_2$, w notacji macierzowej

$$\begin{pmatrix} c(\mathbf{x}) \\ x_{m-1} \\ \ldots \\ \ldots \\ x_1 \end{pmatrix} \;=\; \begin{pmatrix} c_1 & c_2 & \ldots & \ldots & c_m \\ 1 & & & & \\ & 1 & & & \\ & & \ldots & & \\ & & & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} x_{m-1} \\ x_{m-2} \\ \ldots \\ \ldots \\ x_0 \end{pmatrix}$$

A zatem (17) implikuje, że

$$W_{k+1} \;\; \textstyle\sum_{i=0}^{k-1} b_i \cdot W_{i+1}$$

i ogólnie

$$W_{k+t} \;\; \textstyle\sum_{i=0}^{k-1} b_i \cdot W_{i+t}$$

Patrząc na ostatnią współrzędną wektorów W_{k+t} otrzymujemy, że

$$w_{k+t} \;\; \textstyle\sum_{i=0}^{k-1} b_i \cdot w_{i+t}$$

czyli, dla $\ell \geq k$ mamy

$$w_\ell \;=\; \sum_{i=0}^{k-1} b_i \cdot w_{\ell-(k-i)}$$

Tak więc ciąg W jest generowany przez k-schemat dla $k < m$, wbrew założeniu o minimalności m. □

Powróćmy do scenariusza opisanego na początku tej sekcji. Przypuśćmy, że klucz $w_0 w_1 \ldots w_n$ wykorzystany w systemie *One time pad* został wygenerowany z (być może znacznie krótszego) ciągu $w_0 w_1 \ldots w_{m-1}$ przy pomocy m-schematu liniowej rekursji (LFSR). Fakt 2 wraz z Wnioskiem 1 pokazują, że istnieje $m' \leq m$, takie że znajomość $2m'$ bitów klucza (ewentualnie powyżej progu niewiększego od m) wystarcza dla wyznaczenia całego klucza (ewentualnie bez kilku początkowych bitów). Szyfr ten jest zatem w szczególności podatny na atak, w którym znamy $2m'$ kolejnych bitów tekstu jawnego i szyfrogramu.

Złożoność liniowa

Najmniejszą liczbę m taką, że nieskończony ciąg $W = w_0, w_1, \ldots$ można wygenerować przez pewien m-schemat nazywamy *złożonością liniową* ciągu W. Jeśli w ogóle nie można, złożoność jest ∞. Pojęcie to stosuje się również do ciągów skończonych: złożonością liniową ciągu $V = v_0, w_1, \ldots v_{m-1}$ jest minimalna złożoność rozszerzenia V do ciągu nieskończonego (oczywiście, liczba ta nie przekracza m).

Oczywiście, dla skończonego ciągu $w_0, w_1, \ldots w_{m-1}$ złożoność nie przekracza m.

Fakt 3 *Istnieje algorytm wyznaczający złożoność liniową ciągu $w_0, w_1, \ldots w_{n-1}$ w czasie $\mathcal{O}(n^3 \log n)$.*

Dowód. Zauważmy najpierw, że dla $m < n$, możemy sprawdzić, czy ciąg ma złożoność $\leq m$, rozwiązując układ równań

$$\begin{pmatrix} w_m \\ w_{m+1} \\ \ldots \\ w_{n-1} \end{pmatrix} = \begin{pmatrix} w_{m-1} & w_{m-2} & \ldots & w_0 \\ w_m & w_{m-1} & \ldots & w_1 \\ \ldots & \ldots & \ldots & \ldots \\ w_{n-2} & w_{n-3} & \ldots & w_{n-m-1} \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ c_2 \\ \ldots \\ c_m \end{pmatrix}$$

w czasie $\mathcal{O}(m^3)$. Właściwe m znajdziemy stosując binarne wyszukiwanie. □

Uwaga Powyższy algorytm nie jest optymalny. Algorytm Berlekampa-Massey'a, wykorzystujący własności ciał skończonych, znajduje złozoność liniową w czasie $\mathcal{O}(n^2)$ [5].

1.4 System DES i kryptoazaliza różnicowa

Do uzupełnienia

2 Algorytmy szyfrowania asymetrycznego

2.1 Funkcje jednokierunkowe

Wielomianowa obliczalność

Mówiąc o *funkcji obliczalnej* mamy na myśli funkcję $f : \Sigma^* \to \Gamma^*$, gdzie Σ i Γ są skończonymi alfabetami, zazwyczaj wystarczy nam $\Sigma = \Gamma = \{0, 1\}$. W ten sposób traktujemy też funkcje na liczbach naturalnych, $f : \mathbb{N} \to \mathbb{N}$, utożsamiając liczbę n z jej binarną reprezentacją (czasem z początkowym prefiks zer). Często dopuszczamy, by funkcja miała więcej argumentów, ogólnie $f : (\{0,1\}^*)^k \to \{0,1\}^*$; sytuację tę można łatwo sprowadzić do funkcji jednoargumentowej stosując jakąś funkcję pary, np.

$$\langle x, y \rangle \;=\; l_1 0 l_2 0 \ldots l_{k-1} 0 l_k 1 x y$$

gdzie $l_1 l_2 \ldots l_k$ jest binarnym przedstawieniem długości x.

W podobny sposób mówimy o *problemie obliczalnym*, który jest reprezentowany jako język $L \subseteq \Sigma^*$ i który możemy utożsamiać z funkcją obliczalną $\chi_L : \Sigma^* \to \{0,1\}^*$ o wartościach w $\{0,1\}$:

$$\chi_L(w) = 1 \iff w \in L.$$

Zakładamy, że Czytelnik ma wyrobioną intuicję wielomianowej obliczalności. *Funkcję wielomianowo obliczalną* możemy ściśle zdefiniować jako funkcję obliczaną przez pewną deterministyczną maszynę Turinga w czasie ograniczonym przez wielomian[6] od długości wejścia. Maszyna może być jedno lub wielotaśmowa, możemy też zamiast maszyny Turinga użyć maszyny *RAM*; wybory te wpływają na wykładnik wielomianu, ale nie naruszają wielomianowości. Dlatego intuicyjnie wystarczy myśleć po prostu o *algorytmie* [działającym w czasie] *wielomianowym* — zakladamy, że Czytelnik zna już wiele takich algorytmów. Należy jednak pamiętać, że złożoność funkcji liczbowych jest mierzona względem *długości* reprezentacji binarnej liczby, a nie jej wartości. A więc na przyklad algorytm Euklidesa jest algorytmem wielomianowym, ale sito Eratostenesa wykładniczym.

Oczywiście, postawienie znaku równości pomiędzy wielomianową obliczalnością a obliczalnością realizowalną w praktyce jest pewnym uproszczeniem. Na przykład algorytmy o złożoności n^{100} lub $10^{10^{10}}n$ nie będą zapewne nadzwyczajnie przydatne. Jednak nie dysponujemy lepszym teoretycznym przybliżeniem praktycznej obliczalności, a poza tym, na ogół dowód wielomianowej obliczalności prowadzi w końcu do algorytmu o jakimś rozsądnym wykładniku i stałej.

Ale również na odwrót — możliwy jest praktycznie szybko działający algorytm w sytuacji, gdy nie potrafimy udowodnić wielomianowej obliczalności problemu, a może wcale jej nie ma. Po pierwsze złożoność średnia może być o wiele niższa od pesymistycznej. Po drugie, możemy poprawić złożoność czasową korzystając z losowości lub dostępu do dodatkowej informacji.

Te sytuacje musimy mieć na uwadze kiedy, jak to ma miejsce w kryptografii, szacujemy moc obliczeniową przeciwnika.

Algorytmy probabilistyczne i niejednorodne

W naszych rozważaniach algorytm będzie często parametryzowany wielomianem; będziemy wtedy stosować tę samą literę, wielką i małą.

Wielomianowy algorytm probabilistyczny A, parametryzowany wielomianem $a(n)$, może być przedstawiony jako deterministyczny algorytm wielomianowy, przyjmujący dwa argumenty $x, r \in \{0,1\}^*$. Zakładamy, że $|r| = a(|x|)$; w przeciwnym razie algorytm sygnalizuje błąd (oczywiście, można to zapewnić w czasie wielomianowym). Wynik działania algorytmu A na parze $\langle x, r\rangle$ oznaczamy $A(x,r)$. Z punktu widzenia deterministycznego algorytmu, obie współrzędne pary mają tę samą naturę, ale my traktujemy je inaczej. Część x nazywamy *właściwym wejściem* („inputem") algorytmu, a część r *bitami losowymi*. Przyjmując jednostajny rozkład prawdopodobieństwa na $\{0,1\}^{a(|x|)}$, traktujemy

[6] Mówiąc *wielomian* mamy zawsze na myśli wielomian $p(n)$ o współczynnikach naturalnych. Równoważnie można powiedzieć „w czasie $n^{\mathcal{O}(1)}$", ale często wygodnie jest odwoływać się do wielomianu $p(n)$.

wynik algorytmu (przy ustalonym x), jako zmienną losową oznaczaną $A(x)$; dokładniej, przyjmujemy

$$p(A(x) = y) = \frac{\#\{r \in \{0,1\}^{a(|x|)} : A(x,r) = y\}}{2^{|a(|x|)}} \tag{18}$$

W literaturze mówi się czasem, że algorytm „rzuca monetą", czyli losuje $|r|$ kolejnych bitów (niezależnie, z prawdopodobieństwem $\frac{1}{2}$). Pamiętajmy jednak, że jest to tylko nasz sposób interpretacji zależności wyniku od danych; w sposobie obliczenia maszyny Turinga nic się nie zmieniło.

Uwaga Jeśli $p(A(x))$ przyjmuje wartości ułamkowe, to w ogólności $A(x)$ nie jest funkcją. Jeśli mimo to używamy algorytmu A do obliczania funkcji (np. funkcji charakterystycznej), to dopuszczamy, że popełnia on błędy – w stosunku do specyfikacji. Jest to normalne założenie w algorytmach probabilistycznych (zob. 11 rozdział [6]); ryzyko błędu jest ceną jaką płacimy za przyśpieszenie algorytmu. Warto jednak wspomnieć, że w pewnych sytuacjach możemy wykorzystać bity losowe bez wprowadzenia błędów, tzn. $p(A(x) = y)$ przyjmuje tylko wartości 0 lub 1 (jak w algorytmie deterministycznym). Czytelnik zaznajomiony z algorytmami rozpoznającymi liczby pierwsze wie zapewne, że składając test Millera-Rabina — który bezbłędnie rozpoznaje liczby pierwsze, ale złożoną może czasem uznać za pierwszą — z testem Adlemana opartym na krzywych eliptycznych (zob. np. [4]) — który bezbłędnie rozpoznaje liczby złożone, ale może czasem pierwszą uznać za złożoną, otrzymamy test typu *Las Vegas*, który nie daje odpowiedzi błędnych, ale może dać odpowiedź „nie wiem". Jeśli teraz zapuścimy ten test równolegle z deterministycznym testem Agrawala, Saxeny i Kayala (który zawsze daje odpowiedź w czasie $\mathcal{O}n^6$) to otrzymamy algorytm bezbłędny i średnio działający szybciej niż AKS[7].

Z kolei możemy „uzmiennić" również x. Dla ustalonego $n \in \mathbb{N}$, niech U_n oznacza zmienną losową przyjmującą wartości w $\{0,1\}^n$ z rozkładem jednorodnym. Przyjmując, że wybór $x \in \{0,1\}^n$ jest niezależny od wyboru $r \in \{0,1\}^{a(n)}$, otrzymujemy zmienną losową $A(U_n)$, gdzie

$$p(A(U_n) = y) = \frac{\#\{\langle x,r\rangle \in \{0,1\}^n \times \{0,1\}^{a(n)} : A(x,r) = y\}}{2^{n+a(n)}} \tag{19}$$

$$= \frac{1}{2^n} \sum_{x \in \{0,1\}^n} p(A(x) = y) \tag{20}$$

Uwaga Powyższa definicja $A(U_n)$ ma również sens, kiedy A jest „zwykłym" algorytmem deterministycznym (bez żadnych r); prowadzi do analizy średniego zachowania A. Możemy na przykład określić prawdopodobieństwo, że liczba jedynek w $A(U_n)$ jest parzysta, albo że algorytm wykona więcej niż n^2 kroków (przy ustalonym modelu obliczeń).

[7] Przyśpieszenie bierze się jednak raczej tylko z testu Millera-Rabina, bo test Adlemana i jego warianty mają również wysoką złożoność wielomianową.

Wielomianowy algorytm niejednorodny (ang. *non-uniform*) C parametryzowany wielomianem $c(n)$ może być przedstawiony jako deterministyczny algorytm wielomianowy od dwóch argumentów x, y, dany wraz ciągiem słów $(d_n)_{n\in\mathbb{N}}$, gdzie $|d_n| = c(n)$. Wynik działania algorytmu C na słowie x określamy jako $C(x, d_{|x|})$. Słowo $d_{|x|}$ jest czasem nazywane *podpowiedzią* (ang. *advice*). Zauważmy, że – oprócz rozmiaru – nie kładziemy żadnego ograniczenia na ciąg d_n; w szczególności nie musi on być nawet obliczalny. Stąd właśnie określenie „niejednorodny". Intuicyjnie, możemy myśleć, że dla każdego n mamy inny algorytm – dostosowany do wejść x długości n, jednak czas działania pozostaje ograniczony przez wspólny wielomian od x (bo jest to czas działania algorytmu C na wejściu $\langle x, d_{|x|}\rangle$, gdzie $|d_{|x|}| = c(|x|)$, a C jest algorytmem wielomianowym).

W kryptografii rozsądne jest zakładanie, że przeciwnik moze dysponować algorytmem niejednorodnym. Nawet jeśli dla jakiegoś problemu nie jest znany deterministyczny algorytm wielomianowy, istnienie podpowiedzi, przy której problem staje się łatwy dla wszystkich wejść danej długości, stanowi realne zagrożenie (bo przeciwnik mogł znaleźć tę podpowiedź np. w wyniku długotrwałych zmasowanych poszukiwań). Ponadto – choć nie jest to oczywiste – algorytmy niejednorodne okazują się silniejsze niż probabilistyczne (zob. Twierdzenie 11.6 w [6]).

Uwaga Klasę problemów (języków) rozpoznawanych przez niejednorodne algorytmy oznacza się $P/poly$ (*poly* odnosi się do faktu, że podpowiedzi mają długość wielomianową). Nietrudno jest pokazać, że klasa ta jest nieprzeliczalna i zawiera języki nieobliczalne; z drugiej strony przypuszcza się, że problemy NP-zupełne nie należą do $P/poly$. Zauważmy różnicę w definicji tej klasy i klasy NP, obejmującej języki postaci

$$L \;=\; \{x : (\exists y)\, R(x, y)\}$$

gdzie R jest relacją obliczalną przez deterministyczny algorytm wielomianowy i (można założyć) $R(x, y) \Rightarrow |y| = r(|x|)$, dla pewnego wielomianu $r(n)$. W przypadku NP, każde $x \in L$ może mieć swoją własną „podpowiedź" y (zazwyczaj wiele podpowiedzi), natomiast w przypadku $P/poly$ jest jedna podpowiedź dobra dla wszystkich słów danej długości.

Bardzo naturalnym modelem obliczeń dla klasy $P/poly$ są sieci (obwody) logiczne o rozmiarze wielomianowym. Język $L \subseteq \{0,1\}^*$ z tej klasy jest rozpoznawany przez ciąg obwodów $(C_n)_{n\in\mathbb{N}}$, gdzie obwód C_n przyjmuje jako wejście wektory Boole'owskie $x_1 \ldots x_n$ i tym samym rozpoznaje fragment $L \cap \{0,1\}^n$. Zakładamy przy tym, że $|C_n| \leq c(n)$, dla pewnego wielomianu c, ale poza tym obwody C_n nie są ze sobą powiązane (stąd niejednorodność).

Podobnie jak wyżej, uzmienniając x, otrzymujemy

$$p(C(U_n, d_n) = y) \;=\; \frac{\#\{x \in \{0,1\}^n : C(x, d_n) = y\}}{2^n} \tag{21}$$

Zaniedbywalność

Powiemy, że funkcja $w : \mathbb{N} \to \mathbb{R}_+$ jest *zaniedbywalna*, jeśli, dla każdego wielomianu $p(n)$

$$w(n) \;\leq\; \frac{1}{p(n)},$$

dla prawie wszystkich n.

Zazwyczaj będzie nas interesowała zaniedbywalność prawdopodobieństwa, że algorytm probabilistyczny osiągnie jakiś efekt (np. złamie szyfr). Z tego punktu widzenia warto zauważyć, że powtarzanie algorytmu wielomianową liczbę razy nie może zmienić zaniedbywalności.

Dokładniej, przypuśćmy, że A jest algorytmem probabilistycznym, a φ jest własnością taką, że prawdopodobieństwo $p(\varphi(U_n, A(U_n))$ jest zaniedbywalne (jako funkcja n). Niech $s(n)$ będzie wielomianem. Rozważmy algorytm A':

Input $x \in \{0,1\}^n$.

For i := 1 **to** $S(n)$ **do** $A(x)$; **reset**; **endFor**.

Oczywiście, dla deterministycznego A powtarzanie obliczeń na tym samym wejściu nie miałoby większego sensu, ale w algorytmie probabilistycznym przebiegi moga być rózne ze względu na wciąż nowe bity losowe. Niech $A_i(x)$ oznacza wynik po i-tej rundzie. Wtedy

$$p\left((\exists i \leq S(n))\, \varphi(U_n, A_i(U_n))\right) \text{ jest zaniedbywalne.} \tag{22}$$

Istotnie, niech $p(n)$ bedzie dowolnym wielomianem. Będziemy chcieli pokazać, że prawdopodobieństwo, że φ nie zajdzie po żadnej rundzie jest $\geq 1 - \frac{1}{p(n)}$. Z elementarnej analizy łatwo wynika, ze możemy znaleźć wielomian $p_1(n)$, taki że

$$\left(1 - \frac{1}{p_1(n)}\right)^{S(n)} \quad \geq \quad 1 - \frac{1}{p(n)}$$

dla prawie wszystkich n.

Z założenia $p\left(\neg\varphi(U_n, A(U_n))\right) \geq 1 - \frac{1}{p_1(n)}$, dla prawie wszystkich n. Wtedy

$$\begin{aligned}
p\left((\forall i \leq S(n))\, \neg\varphi(U_n, A_i(U_n))\right) &= \frac{1}{2^n} \sum_{x \in \{0,1\}^n} p(\neg\varphi(x, A(x)))^{S(n)} \\
&\geq \left(\frac{1}{2^n} \sum_{x \in \{0,1\}^n} p(\neg\varphi(x, A(x)))\right)^{S(n)} \\
&= \left(p\left(\neg\varphi(U_n, A(U_n))\right)\right)^{S(n)} \\
&\geq \left(1 - \frac{1}{p_1(n)}\right)^{S(n)} \\
&\geq 1 - \frac{1}{p(n)}.
\end{aligned}$$

Zauważmy, że skorzystaliśmy tu ze znanej nierowności $\frac{1}{N}\sum_{i=1}^N a_i^k \geq \left(\sum_{i=1}^N a_i\right)^k$, jaka wynika z wypukłości funkcji x^k.

Funkcje jednokierunkowe

Podstawowa intuicja funkcji jednokierunkowej jest: *łatwo obliczalna, ale trudno odwracalna*, tzn. na podstawie $f(x)$ trudno jest znaleźć (jakiekolwiek) x' takie, że $f(x') = f(x)$.

Jednak w tym ostatnim punkcie nie zadowala nas trudność ze względu na algorytmy deterministyczne; oczekujemy jej również dla algorytmów probabilistycznych, a nawet niejednorodnych. Z drugiej strony nie możemy wykluczyć, że algorytm probabilistyczny, dla danego $f(x)$, wylosuje dokładnie x; możemy jednak wymagać, by prawdopodobieństwo takiego zdarzenia było małe.

Funkcja $f : \{0,1\}^* \to \{0,1\}^*$ jest *jednokierunkowa* (ang. *one way*), jeśli spełnia następujące warunki.

1. f jest obliczalna przez deterministyczny algorytm wielomianowy.

2. Dla każdego probabilistycznego algorytmu wielomianowego A, prawdopodobieństwo

$$p\left(A(f(U_n), 1^n) \in f^{-1}(f(U_n))\right) \tag{23}$$

jest zaniedbywalne.

W silniejszym wariancie definicji, warunek 2 zastępuje się

2′. Dla każdego niejednorodnego algorytmu $D(., c_n)$, prawdopodobieństwo

$$p\left(D(f(U_n), 1^n, c_n) \in f^{-1}(f(U_n))\right) \tag{24}$$

jest zaniedbywalne.

Uwaga W powyższej definicji, algorytm A (podobnie algorytm D) przyjmuje jako wejście właściwe parę, przy czym interesuje nas sytucja, gdy pierwszą współrzędną jest $f(x)$, dla pewnego $x \in \{0,1\}^n$, podczas gdy druga, 1^n, niejako „podpowiada" algorytmowi, jakiej długości ma być poszukiwany argument. Nie powinno nas to jednak zmylić; dokładna „podpowiedź" długości nie jest istotna — równie dobrze algorytm mógłby znać tylko górne ograniczenie i probować dla każdej potencjalnej długości $|x|$ (czas pozostałby wielomianowy). Warunek ten jest dodany ze względow technicznych, żeby uniknąć sytuacji, gdy trudność odwrócenia $f(x)$ wynika jedynie z faktu, że $f(x)$ jest o wiele krótsze od x. Alternatywnie zakłada się czasem, że funkcja f nie może „za bardzo" skracać, tzn. $|f(x)| \geq |x|^{\frac{1}{k}}$, dla pewnego k. (Czytelnik może sprawdzić, że w takim przypadku nie potrzeba dodawać argumentu 1^n w definicji funkcji jednokierunkowej.)

Dla lepszego zaznajomienia się z powyższymi definicjami udowodnimy, że jeśli funkcja jest jednokierunkowa ze względu na „ataki" niejednorodne, to tym bardziej jest odporna na ataki probabilistyczne.

Fakt 4 *Jeśli f spełnia warunek 2′, to spełnia warunek 2.*

Dowód. Przypuśćmy, że jest przeciwnie i niech q będzie takim wielomianem, że

$$p\left(A(f(U_n, 1^n) \in f^{-1}f(U_n)\right) \quad \geq \quad \frac{1}{q(n)}, \tag{25}$$

dla nieskończenie wielu n. Lewą stronę (25) możemy przedstawić jako

$$\begin{aligned}\frac{1}{2^n}\sum_{x\in\{0,1\}^n} p\left(A(f(x),1^n)\in f^{-1}f(x)\right) &= \frac{\#\{\langle x,r\rangle\in\{0,1\}^{n+a(n)} : A(f(x),1^n,r)\in f^{-1}f(x)\}}{2^{n+a(n)}}\\ &= \frac{1}{2^{a(n)}}\sum_{r\in\{0,1\}^{a(n)}}\frac{\#\{x\in\{0,1\}^n : A(f(x),1^n,r)\in f^{-1}f(x)\}}{2^n}\end{aligned}$$

Jeśli więc zachodzi (25), to możemy wskazać konkretne $r=r_n$, dla którego

$$\frac{\#\{x\in\{0,1\}^n : A(f(x),1^n,r_n)\in f^{-1}f(x)\}}{2^n} = p\left(A(U_n,1^n,r_n)\in f^{-1}f(U_n)\right)\geq\frac{1}{q(n)}.$$

A zatem algorytm niejednorodny $A(.,r_n)$ (gdzie dla pozostałych n, r_n jest dowolnym słowem długości $a(n)$), przeczy warunkowi $2'$, wbrew założeniu. □

Nietrudno jest zauważyć, że jesli $P=NP$, to funkcje jednokierunkowe nie istnieją (implikcja przeciwna nie jest znana), nic więc dziwnego, ze o żadnej funkcji nie udowodniono jeszcze, ze jest jednokierunkowa. Istnieje jednak wiele naturalnych kandydatek, w szczególności przypuszcza się, że funkcja mnożenia

$$\langle m,n\rangle \;\mapsto\; m\cdot n$$

(gdzie wszystkie liczby są dane w postaci binarnej) jest jednokierunkowa.

Często podawany jest przykład potęgowania *modulo* liczba pierwsza:

$$\langle p,g,x\rangle \;\mapsto\; g^x \bmod p$$

gdzie p jest liczbą pierwszą, a g jest generatorem grupy multiplikatywnej $\mathbb{Z}_p^*$. Funkcja odwrotna nosi wtedy nazwe *logarytmu dyskretnego*. W definicji tej jest jednak pewien problem. Ze względu na zastosowania, chcielibyśmy móc najpierw stwierdzić, że p i g rzeczywiście spełniają powyższy warunek; czy jednak potrafimy to zrobić w wielomianowym czasie? Od 2002 r., algorytm AKS zapewnia deterministyczne testowanie pierwszości, ale co ze sprawdzeniem, że g jest generatorem? Na szczęście, od dawna znany jest sposób krótkiego certyfikowania tej własności (podany przez Vaughana Pratta), który poniżej omówimy. Właściwa definicja potęgowania powinna więc raczej brzmieć

$$\langle p,g,C(p,g),x\rangle \;\mapsto\; g^x \bmod p$$

gdzie $C(p,g)$ jest certyfikatem faktu, że p jest pierwsza, a g jest generatorem grupy multiplikatywnej $\mathbb{Z}_p^*$ (zdefiniowanym poniżej). Warnek ten można sprawdzić w czasie wielomianowym, jeżeli nie jest spelniony – algorytm sygnalizuje błąd.

Certyfikaty pierwszości

Jeśli $p=2$, to jedynym generatorem $\mathbb{Z}_p^*$ jest $g=1$. Jeśli $p>2$, to naiwny sposob sprawdzenia, że g jest generatorem $\mathbb{Z}_p^*$ polegałby na sprawdzeniu, że $g^{p-1}=1 \bmod p$, podczas gdy $g^i\neq 1 \bmod p$, dla $1\leq i<p-1$. Zauważmy, ze ten ostatni warunek

implikuje również pierwszość p, oznacza bowiem że $\mathbb{Z}_p^*$ ma $p-1$ elementów. Istotnie, $g^{p-1} = 1 \bmod p$ daje nam $g \perp p$ (g względnie pierwsze z p). Gdyby $g^i = g^j \bmod p$, dla pewnych $1 \leq i < j \leq p-1$, to $p | g^j - g^i = g \cdot (g^{i-1} - g^{j-1})$, a zatem $p | g^{i-1} - g^{j-1}$. A więc minimalne i o tej własności musi być 1, ale wtedy $p | 1 - g^{j-1}$, czyli $1 = g^{j-1} \bmod p$, wbrew założeniu.

Jednak sprawdzenie wszystkich i nie byłoby wielomianowe (względem długości p). Zauważmy jednak, że jeśli $g^i = 1 \bmod p$, dla pewnego $1 \leq i < p-1$, to również $g^j = 1$, dla pewnego właściwego dzielnika $j | p-1$, tzn. $1 < j < p-1$. Istotnie, jeśli i jest *najmniejsze* o tej własności i przedstawimy $p - 1 = \alpha \cdot i + i_1$, to

$$1 = g^{p-1} = \underbrace{g^{\alpha \cdot i}}_{1} \cdot g^{i_1} \bmod p$$

a więc również $g^{i_1} = 1 \bmod p$ i wobec minimalności i, $i_1 = 0$. A zatem wystarczy sprawdzić, że warunek $g^i \neq 1 \bmod p$, zachodzi dla właściwych dzielników $p-1$. Ponieważ $g^i = 1 \bmod p$ implikuje $g^{\alpha \cdot i} = 1 \bmod p$, dla dowolnego α, wystarczy dalej ograniczyć się do „maksymalnych" dzielników (ze względu na relację podzielności). Reasumując, jeżeli $p - 1 = p_1^{\alpha_1} \cdot \ldots \cdot p_k^{\alpha_k}$ jest faktoryzacją $p-1$ na czynniki pierwsze, wystarczy sprawdzić, że

$$g^{\frac{p-1}{p_i}} \quad \neq \quad 1 \bmod p$$

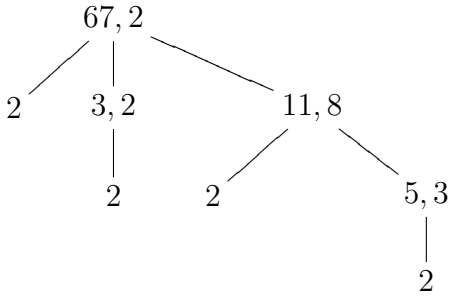
dla $i = 1, \ldots, k$.

Powyższe obserwacje prowadzą do rekurencyjnej definicji certyfikatu.

$$C(p,q) = \begin{cases} 2 & \text{gdy } p = 2 \wedge q = 1 \\ p_1, \ldots, p_k,\ \alpha_1, \ldots, \alpha_k & \\ C(p_1, q_1), \ldots, C(p_k, q_k) & \text{w pp.} \end{cases}$$

gdzie ponadto $p - 1 = p_1^{\alpha_1} \cdot \ldots \cdot p_k^{\alpha_k}$ i $g^{\frac{p-1}{p_i}} \neq 1 \bmod p$, dla $i = 1, \ldots, k$.

Certyfikat ten można przedstawić w postaci drzewa, gdzie dzieci wierzchołka $C(p,q)$ są etykietowane $C(p_1, q_1), \ldots, C(p_k, q_k)$, a liśćmi są certyfikaty pierwszości dwójki. Oto przykładowy certyfikat pierwszości liczby 67.



Wobec nierówności $p_1 \cdot \ldots \cdot p_k < p$, nietrudno widzieć, że iloczyn wszystkich liści jest mniejszy niż p, a zatem liczba liści jest $\leq \log_2 p$. Każdy wierzchołek z wyjątkiem liści i rodziców liści ma co najmniej 2 następniki (2 i coś jeszcze). To daje nam oczywiste oszacowanie na liczbę wszystkich wierzchołków w drzewie ($\leq 3 \cdot \log_2 p$) i tym samym na rozmiar certyfikatu.

2.2 Kryptosystem klucza publicznego

Systemy szyfrujące (lub kryptosystemy) wprowadziliśmy na samym początku wykładu (punkt 1.1). Przypomnijmy, że system asymetryczny to taki, który spełnia (1), choć niekoniecznie (3). Jaki może być pożytek z takich systemów?

Najważniejsza okaże się tu własność obliczeniowa. Jeżeli na podstawie szyfrogramu, nawet przy znajomości klucza szyfrującego, bardzo trudno jest odtworzyć tekst jawny (a więc tym bardziej klucz deszyfrujący), to bez obawy można przesłać ten klucz drogą jawną (np. oglosić w prasie), przynajmniej na użytek korespondencji w jedną stronę. Rozwiązuje to podstawowy w kryptografii symetrycznej problem wcześniejszej dystrybucji kluczy. Istnienie – przynajmniej teoretycznie – takiej możliwości zostało odkryte przez Diffiego i Hellmana w 1975 r., a wsparte konkretnym algorytmem o żądanej własności (RSA) przez Rivesta, Shamira i Adlemana w 1977 r. Jak ogłoszono w 1997 r., brytyjscy matematycy pracujący w Cambridge dla wojska brytyjskiego (Ellis, Cocks, Williamson) mieli podobne odkrycia nieco wcześniej – już ok. 1973 r.

Rozpoczniemy od przykładów. Zgodnie z tradycją, będziemy zakładać, ze Alicja chce przesłać zaszyfrowaną wiadomość do Boba. Inaczej niż w punkcie 1.1, gdzie badaliśmy jeden „oderwany" kryptosystem, będziemy teraz rozważać algorytm (probabilistyczny), który dla danego $\ell \in \mathbb{N}$ konstruuje kryptosystem stosowny do szyfrowania wiadomości w $\{0,1\}^\ell$. Nasz obecny scenariusz zakłada asymetrię: Alicja będzie znała tylko klucz szyfrujący, który mogą poznać wszyscy – dlatego jest nazywany *kluczem publicznym*, podczas gdy Bob zna zarówno klucz szyfrujący jak i deszyfrujący, zwany także *kluczem prywatnym*. Dlatego wygodnie jest myśleć, że w pierwszym kroku algorytmu, to właśnie Bob (na podstawie ℓ) wybiera oba klucze. Ten krok będziemy nazywali z angielska *setup*.

System Ellisa–Cocksa

Setup Dane ℓ. Bob generuje liczby pierwsze p i q tak by $|p \cdot q| \approx \ell$. Wymagamy ponadto, żeby $p \perp q-1$ i $q \perp p-1$. **Kluczem publicznym** zostaje $n = p \cdot q$; **kluczem prywatnym** są liczby p i q. Zakładamy, że

$$\mathcal{M} = \mathcal{C} = \{0, 1, \ldots, n-1\}$$

przy czym – na potrzeby algorytmu – elementy tego zbioru są reprezentowane binarnie z początkowymi zerami, tak że wszystkie mają długość ℓ.

Dla potrzeb przyszłego deszyfrowania, Bob przy pomocy algorytmu Euklidesa znajduje wzajemne odwrotności, tzn. r, s, u, v, takie że

$$\begin{aligned} pr &= 1 \bmod q-1 \\ qs &= 1 \bmod p-1 \\ pu &= 1 \bmod q \\ qv &= 1 \bmod p \end{aligned}$$

Szyfrowanie Chcąc wysłać wiadomość $M \in \mathcal{M}$, Alicja oblicza i wysyła do Boba

$$E(M,n) \;=\; M^n \bmod n$$

Deszyfrowanie Bob deszyfruje otrzymany szyfrogram C według wzoru

$$D(C, \langle p, q\rangle) \quad = \quad up\,(C^r \bmod q) + vq\,(C^s \bmod p) \bmod n$$

(występujące tu stałe r, s, u, v są jednoznacznie wyznaczone i łatwo obliczalne z p i q).

Pozostaje sprawdzić, że

$$D(E(M, n), \langle p, q\rangle) \quad = \quad M.$$

W tym celu wystarczy sprawdzić, że

$$n \,|\, upM^{nr} + vqM^{ns} - M$$

ale ze względu na pierwszość p i q (oraz symetrię), wystarczy sprawdzić, że

$$q \,|\, upM^{nr} - M$$

Korzystając dwukrotnie z Małego Twierdzenia Fermata ($M^{\alpha\cdot(q-1)+1} = M \bmod q$), otrzymujemy że

$$M^{q\cdot p\cdot r} = M^{p\cdot r} = M \bmod q$$

wystarczy więc sprawdzić, że

$$q \,|\, M(up - 1)$$

co zachodzi, z wyboru u.

Własności dobrego systemu

Używając podobnej analizy jak dla funkcji jednokierunkowych, możemy sformułować postulaty dobrego systemu. (Wygodnie będzie dopuścić wartość *error*.)

Istnieją wielomianowe algorytmy: probabilistyczny I, deterministyczny lub probabilistyczny E, oraz deterministyczny D o następujących własnościach.

1. $p(I(1^\ell) = \mathit{error})$ jest zaniedbywalne.

   Jeśli $I(1^\ell, r) \neq \mathit{error}$ (gdzie r oznacza ciąg bitów losowych), to

   $$I(1^\ell, r) \quad = \quad \langle k_1^\ell, k_2^\ell\rangle$$

   Słowa k_1^ℓ i k_2^ℓ stanowią klucz publiczny i prywatny, odpowiednio. Dla danego ℓ, możliwych par k_1^ℓ, k_2^ℓ może – a nawet powinno[8]– być wiele, ale możemy założyć, że dla różnych ℓ, wartości na żadnej ze współrzędnych się nie pokrywają, tzn. $k_i^\ell \neq k_i^{\ell'}$, dla $\ell \neq \ell'$. W dalszym ciągu, notacja k_i^ℓ oznacza klucz wygenerowany z 1^ℓ.

[8] Por. warunek doskonałej tajności w Twierdzeniu 1.

2. $E(m, k_1^\ell) \neq error$, dla co najmniej połowy[9] ciągów $m \in \{0,1\}^\ell$.

   Jeśli E jest algorytmem probabilistycznym, wymagamy, żeby $p(E(m, k_1^\ell) = error) = 0$, dla co najmniej połowy $m \in \{0,1\}^\ell$.

   Co więcej, dla $m_1 \neq m_2$

   $$p(E(m_1, k_1^\ell) = E(m_2, k_1^\ell)) = 0$$

   (jednoznaczność szyfrowania).

3. Dla każdego wielomianowego probabilistycznego algorytmu A, prawdopodobieństwo

   $$p\left(A(E(m, k_1^\ell), k_1^\ell, 1^\ell) = m\right)$$

   jest zaniedbywalne. Intuicyjnie: żaden wielomianowy algorytm probabilistyczny, nawet znając k_1^ℓ, nie potrafi z $E(m, k_1^\ell)$ odtworzyć m.

4. Możemy to natomiast łatwo zrobić, znając k_2^ℓ:

   $$D(E(m, k_1^\ell), k_2^\ell) \quad = \quad m.$$

Ciąg dalszy nastąpi

Literatura

[1] Oded Golderich, Foundations of Cryptography. Basic Tools, Cambridge University Press, 2001.

[2] Oded Golderich, Foundations of Cryptography. Basic Applications, Cambridge University Press, 2004.

[3] Auguste Kerckhoffs, *La cryptographie militaire*, Journal des sciences militaires, vol. IX, pp. 5–83, jan. 1883, pp. 161–191, févr. 1883.

[4] Neal Koblitz, Wykład z teorii liczb i kryptografii, WNT, 2006.

[5] Rudolf Lidl and Harald Niederreiter, Introduction to Finite Fields and their Applications, Cambridge University Press, 1986.

[6] Christos P. Papadimitriou, Złożoność obliczeniowa, WNT, 2002.

[7] Bruce Schneier, Kryptografia dla praktyków, WNT, 2002.

[8] Douglas R. Stinson, Kryptografia. W teorii i w praktyce, WNT, 2005.

[9] Jaohn Talbot and Dominic Welsh, Complexity and Cryptography, Cambridge University Press, 2006.

[9] Oczywiście, najlepiej byłoby powiedzieć: dla wszystkich $m \in \{0,1\}^{(\ell)}$. Ale ani szyfr Ellisa-Cooksa ani RSA nie spełniają tej własności: ciągi o długości $|n|$ reprezentujące liczby większe niż n nie będą zaszyfrowane. Stąd to – mało eleganckie – osłabienie.