Wykład 2. Wprowadzenie do kryptografii (cz. 2), RSA

Stefan Dziembowski stefan-mpe@dziembowski.net

Streszczenie. Kontynuujemy wprowadzenie do kryptografii (definiujemy schematy uwierzytelniania i przedstawiamy kryptografię klucza publicznego). Definiujmy szyfr RSA wraz z niezbędnymi podstawami matematycznymi (algebra, teoria liczb).

2 Uwierzytelnianie

W poprzednim rozdziale zajmowaliśmy się wyłącznie *utajnianiem* wiadomości (zakładaliśmy, że Ewa nie jest w stanie zrobić nic ponad podsłuchiwanie kanału łączącego Alicję i Boba). W tym rozdziale zajmiemy się metodami *uwierzytelniania* (inaczej: *zachowania integralności*), które uodporniają uczciwych użytkowników protokołu przed ingerencją Ewy w treść komunikacji. Celem Ewy może być sfarbrykowanie bądź modyfikacja przesyłanych wiadmości. Zauważmy, że samo szyfrowanie nie daje takiej ochrony.

Przykład. Załóżmy, że Alicja i Bob używają szyfru Vernama. Ewa wie (albo domyśla się), że Alicja wysłała Bobowi zaszyfrowaną wiadomość m. Ewa przechwytuje kryptogram $c = m \oplus k$ i wysyła do Boba wiadomość $c \oplus m \oplus m'$ (gdzie m' jest wiadomością wybraną przez Ewę). Wówczas Bob odszyfruje $(c \oplus m \oplus m') \oplus k = m \oplus k \oplus m \oplus m' \oplus k = m'$.

Rozwiązaniem tego problemu jest wyposażenie Alicji i Boba w *schemat uwierzytelniania* $U = (\mathcal{M}, \mathcal{K}, \mathcal{T}, \mathcal{S}, \mathcal{V})$, gdzie:

- $\mathcal{M}$ jest zbiorem *wiadomości* (które Alicja może chcieć wysłać do Boba),
- $\mathcal{K}$ jest zbiorem *kluczy*,
- $\mathcal{T}$ jest zbiorem *oznaczników*,
- $\mathcal{S}$ jest *algorytmem oznaczającym*, który na wejściu pobiera $k \in K$ i $m \in \mathcal{M}$ a na wyjściu zwaraca $t \in T$,
- $\mathcal{V}$ jest *algorytmem weryfikującym*, który na wejściu pobiera $k \in K$, $m \in \mathcal{M}$ i $t \in T$ a ny wyjściu zwraca `tak` albo `nie`

Schematu tego uzywa sie w sposób następujący. Po pierwsze, zakładamy, że Alicja i Bob dysponują kluczem k wybranym losowo ze zbioru $\mathcal{K}$. Jeśli Alicja chce wysłać do Boba wiadomość m, to wraz z nią wysyła oznacznik $\mathcal{S}(k, m)$. Po otrzymaniu pary

(m,t) Bob uruchamia algorytm $\mathcal{V}$ na wejściu (k,m,t) i akceptuje m (jako pochodzącą od Alicji) jeśli algorytm zwrócił `tak`. Oczywiście wymagamy, by schemat był *bezpieczny*, to znaczy by Ewa (nawet jeśli usłyszała komunikat (m,t)) nie potrafiła sama sfabrykować dowolnej pary (m',t') takiej, że $\mathcal{V}(k,m',t') = \texttt{tak}$. Podobnie jak w przypadku szyfrowania, zakładamy, że Ewa ma pełną wiedzę na temat U. Szczegółową definicją bezpieczeństwa zajmiemy się później.

Naturalnie, wymagamy także, by algorytmy $\mathcal{S}$ i $\mathcal{V}$ były wydajne. Często weryfikacja polega po prostu na policzeniu oznacznika dla danego klucza i danej wiadomości i porównaniu z danym oznacznikiem. W tych przypadkach bedziemy pomijali definicję algorytmu $\mathcal{V}$.

2.1 Kody Uwierzytelniające

Podobnie jak w przypadku szyfrowania również tutaj istnieje rozwiązanie które jest idealnie bezpieczne, ale mało praktyczne. W rozwiązaniu tym Alicja i Bob używają następującej metody. Przypuśćmy, że dla pewnej liczby pierwszej q zbiór wiadomości to $\mathcal{M} = Z_q$, zbiór oznaczników $\mathcal{T} = Z_q$, a zbiór kluczy to $\mathcal{K} = Z_q \times Z_q$. Ponadto:

$$\mathcal{S}((k_0,k_1),m) = k_0 \cdot m + k_1 \bmod q.$$

Lemat 1 *Niech K będzie zmienna losową o rozkładzie jednostajnym na $\mathcal{K}$, niech m będzie dowolną wiadomością i niech $\mathcal{A}$ będzie dowolnym algorytmem pobierającym na wejściu m i $s = \mathcal{S}(K,m))$ zwracającym parę (m',s'). Wówczas prawdopodobieństwo, że $s' = \mathcal{S}(K,m')$ wynosi co najwyżej $1/q$.*

Dowód. Jeśli $\mathcal{A}$ otrzymał na wejściu m i s, to zbiór możliwych wartości (k_0,k_1) zmiennej K to

$$\mathcal{K}_{m,s} := \{(k_0,k_1) : k_0 \cdot m + k_1 = s\}.$$

Natomiast podzbiór $\mathcal{K}_{m,s}$ takich kluczy k, że $s' = \mathcal{S}(k,m')$ to

$$\mathcal{K}_{m,s,m',s'} := \left\{ (m_0,m_1) : \begin{array}{l} k_0 \cdot m + k_1 = s \\ k_0 \cdot m' + k_1 = s' \end{array} \right\}$$

(w powyższych wyrażeniach wszystkie operacje liczone są w ciele Z_q). Z algebry liniowej zbiór $\mathcal{K}_{m,s}$ jest warstwą jednowymiarową w przestrzeni $Z_q \times Z_q$, ma więc moc q. Zbiór $\mathcal{K}_{m,s,m',s'}$ jest warstwą zerowymiarową (równania które ją wyznaczają są liniowo niezależne gdyż $m \neq m'$), ma więc moc 1. Zatem szansa, że losowy klucz należący do zbioru $\mathcal{K}_{m,s}$ należy też do $\mathcal{K}_{m,s,m',s'}$ (czyli że $s' = \mathcal{S}(K,m')$) wynosi

$$\frac{|\mathcal{K}_{m,s}|}{|\mathcal{K}_{m,s,m',s'}|} = 1/q.$$

□

Zauważmy, że w dowolnym schemacie uwierzytelniania zawsze szanse przeciwnika (na sfabrykowanie oznacznika) wynoszą co najmniej $|\mathcal{T}|^{-1}$ (bo takie szanse uzyskuje

przeciwnik zgadujący na chybił–trafił). W tym sensie kody uwierzytelniające są optymalnym rozwiązaniem, podobnie jak szyfr Vernama w przypadku tajności. Niestety podobieństwa idą dalej: kody uwierzytelniające nie są stosowane w praktyce ze względu na to, że raz użytego kluczo nie da się użyć ponownie. Łatwo bowiem zauważyć (stosując podobne rozumowanie jak w dowodzie Lematu 1), że znajomość dwóch różnych par (m, s) i (m', s') wyprodukowanych za pomocą tego samego klucza k pozwala obliczyć k.

2.2 Kody Uwierzytelniające Wiadomość

W praktyce używa się kodów, które zapewaniają bezpieczeństwo tylko względem przeciwnika ograniczonego obliczeniowo (w odróżnieniu od kodów uwierzytelniających z Rozdziału 2.1), ale za to dopuszczają wielokrotne używanie tego samego klucza. Nazywa się je *Kodami Uwierzytelniającymi Wiadomość (MAC)*. Przykładem takich kodów jest np. CBC MAC (wykorzystujący ustalony szyfr blokowy, np. AES). Więcej informacji można znaleźć np. w [GB].

2.3 Definicja Schematu Uwierzytalniania

Tak jak powiedzieliśmy na początku tego rozdziału, uznajemy, że celem przeciwnika jest wyprodukowanie dowolnej pary (m', t') takiej, że algorytm werifikujący (który pobrał tę parę jako argument) odpowie `tak`. Zauważmy, że jest to wymaganie dość restrykcyjne, gdyż uznaje ono schemat za złamany nawet jeśli wiadomość m' nie ma sensu (np. nie jest zgodna z określonym formatem). Takie podejście do sprawy jest wyrazem (typowej dla kryptografii) zapobiegliwości.

Przedstawimy teraz (w nieformalny sposób) definicję bezpieczeństwa schematów uwierzytelniania. Przypuśćmy zatem, że Alicja i Bob są w posiadaniu losowego klucza K. Podobnie jak w przypadku szyfrowania musimy określić rodzaj taku jaki przeciwnik może przeprowadzić zanim przystąpi do próby wyprodukowania pary (m', s'). Rozważa się (między innymi) następujace ataki.

1. Atak ze znaną wiadomością — Ewa może poznać dowolną liczbę par

$$(m_1, \mathcal{S}(K, m_1)), \ldots, (m_n, \mathcal{S}(K, m_n)).$$

   W tym przypadku należy jeszcze określić rozkład z jakim wybrane są wiadomości.

2. Atak z wybraną wiadomością — jak w Punkcie 1z tym, że Ewa może wybrać wiadomości $m_1, \ldots, m_n$. Podobnie jak w przypadku szyfrowania rozważa się następujące warianty:

   (a) atak wsadowy — wiadmości muszą być wybrane z góry,

   (b) atak adaptywny — wiadomości mogą być dobrane na podstawie oznaczników na poprzednich wiadomościach.

Tak jak w przypadku szyfrowania atak z Punktu 1 jest łatwy do przeprowadzenia w praktyce. Istnieją też praktyczne scenariusze w których może nastąpić atak z Punktu

2. (Np. gdy Alicja uwierzytelnia wszystkie komunikaty które wysyła do Boba i Ewie udało się nakłonić ją do „sforwardowania" Bobowi jakiejś określonej wiadomości.)

Po tym jak Ewa przeprowadziła jeden z powyższych ataków (dla ustalenia uwagi: atak ze znaną wiadomością) jej zadaniem jest wyprodukowanie pary (m', t'), takiej, że $\mathcal{V}(K, m', t') = \texttt{tak}$ (przy czym: $m \notin \{m_1, \ldots, m_n\}$). Powiemy, że schemat jest *bezpieczny ze względu na atak ze znaną wiadomością*, jeśli szanse dowolnej (wydajnej) Ewy na sukces są zanikome. Analogicznie definiuje się bezpieczeństwo ze względu na (adaptywny/wsadowy) atak ze znana wiadomością.

Aby uzyskać formalną defincję bezpieczeństwa należałoby (podobnie jak w przypadku szyfrów) wprowadzić parametr bezpieczeństwa, pojęcie „wydajnej Ewy" zastąpić „wielomianowym algorytmem probabistycznym" a „znikomy" zastąpić „zaniedbywalnym", etc. Należy też zastąpić „zbiór kluczy" przez „algorytm generacji klucza".

3 Kryptografia klucza publicznego

Dotyczczas rozważaliśmy kryptosystemy (szyfry i schematy uwierzytelaniania) w których klucz do odszyfrowywania (lub weryfkiacji) był identyczny z kluczem szyfrującym (lub oznaczającym). Jest to tak zwana *kryptografia symetryczna*. W latach 70tych rewolujcę w kryptografii spowodowała propozycja [DH76, RSA78] kryptosytemów w których klucze te są rózne. W tym wypadku algorytm generacji klucza tworzy (zamiast pojedynczego klucza) parę: *klucz prywatny* d i *klucz publiczny* e. Zamiast klucza k funkcja $\mathcal{E}$ (odpowiednio: $\mathcal{T}$) pobiera klucz e, natomiast funkcja $\mathcal{D}$ (odpowiednio: $\mathcal{V}$) pobiera klucz d. Wymagamy przy tym, by klucza prywatnego nie dało się (w sposób wydajny) obliczyć na podstawie klucza publicznego. Pozwala to (jak zreszta sugeruje nazwa) na opublikowanie klucza e (np. w gazecie albo w sieci Internet). Zaleta tego podejścia jest jasna: w odróżnieniu od kryptografii symetrycznej nie ma potrzeby tworzenia osobnego klucza dla każdej pary użytkowników. Ktokolwiek zna klucz publiczny Alicji może do niej wysłać kryptogram który tylko ona będzie potrafiła odszyfrować. Podobnie: Alicja może oznaczyć wiadomośc w taki sposób, by każdy mógł zweryfikować to oznaczenie (ale nikt nie mógł oznaczenia podrobić). Dlatego oznaczanie w modelu z kluczem publicznym nazywa się *podpisem elektronicznym*. Osobną kwestią jest to w jaki sposób można upewnić się, że dany klucz e rzeczywiście należy do Alicji[1]. W tym celu tworzy się Infrastrukturę Klucz Publicznego i instytucje w rodzaju Centrów Certyfikacji (więcej na ten temat później).

3.1 Uproszczony obraz

W najprostszym ujęciu szyfr z kluczem publicznym składa się z:

- zbioru wiadomości $\mathcal{M}$.
- zbioru kryptogramów $\mathcal{C}$.
- algorytmu (probabilistycznego) $\mathcal{G}$ generacji klucza, który generuje parę (e, d)

[1] Dokładniej należałoby powiedzieć: Alicja (i tylko ona) zna klucz d który został wygenerowany jednocześni z kluczem e.

- algorytmu szyfrującego $\mathcal{E}$ (deterministycznego), który na wejściu e i $m \in \mathcal{M}$ zwraca kryptogram $\mathcal{E}(e, m)$
- algorytmu odszyfrowującego $\mathcal{D}$, który na wejściu d i c zwraca $\mathcal{D}(d, c) \in \mathcal{M}$.

Oczywiście wymagamy by $\mathcal{D}(d, \mathcal{E}(e, m)) = m$. Schemat podpisu składa się z:

- zbioru wiadomości $\mathcal{W}$,
- zbioru podpisów $\mathcal{T}$,
- algorytmu (probabilistycznego) $\mathcal{G}$ generacji klucza, który generuje parę (e, d)
- algorytmu podpisującego $\mathcal{S}$, który na wejściu d i $m \in \mathcal{W}$ zwraca podpis elektroniczny $\mathcal{S}(d, m)$
- algorytmu veryfikacji $\mathcal{V}$, który na wejściu d, s i m zwraca `tak` albo `nie` (w przypadku gdy $s = \mathcal{S}(e, m)$) algorytm powinien zwrócić `tak`).

(Definicją bezpieczeństwa zajmiemu się za moment.) Jak się okazuje wystarczy skonstruować schemat szyfrowania by uzyskać chemat podpisu. Najprostszy pomysł na uzyskanie schematu podpisu $(\mathcal{W}, \mathcal{T}, \mathcal{G}, \mathcal{S}, \mathcal{V})$ ze schematu szyfrowania z kluczem publicznym $(\mathcal{M}, \mathcal{C}, \mathcal{G}, \mathcal{E}, \mathcal{D})$ jest następujący.

- $\mathcal{W} := \mathcal{M}$,
- $\mathcal{T} := \mathcal{C}$,
- algorytm generacji klucza jest ten sam
- aby uzyskać podpis na wiadomości m traktujemy ją jako szyfrogram i odszyfrowujemy, to znaczy:
$$\mathcal{S}(e, m) := \mathcal{D}(e, m),$$
- Weryfikacja przebiega nastepująco: dla argumentów d, s i m algorytm $\mathcal{V}$ oblicza $\mathcal{E}(s)$. Jeśli wartość ta wynosi m, to algorytm zwraca `tak`, w przeciwnym przypadku zwraca `nie`. Zauważmy, że poprawność tego schamtu opiera się na determinizmie algorytmu szyfrującego oraz na tym, że funkcja którą on wyznacza jest bijekcją[2]

3.2 Dyskusja o bezpieczeństwie

Bezpieczeństwo definiuje się podobnie do przypadku symetrycznego, z tym, że musimy założyć, że Ewa zna klucz pobliczny e. Powstają w związku z tym następujące wątpliwości:

1. Jeśli algorytm szyfrowania jest deterministyczny i jeśli zbiór $\mathcal{X}$ potencjalnych wiadomości jest mały, to Ewa (która zna klucz publiczny) może zaszyfrować wszystkie wiadomości w $\mathcal{X}$ i porównać ich kryptogramy z przesłanym kryptogramem. Poza tym Ewa może dowiedzieć się, że dwa razy zaszyfrowano to samo.

[2]Jeśli tylko założymy, że każdy kryptogram c odpowiada jakiejś wiadomości m.

Powyższe (praktyczne) problemy mają swoje odzwierciedlenie w teorii, mianowicie Ewa bez trudu może wygrać „grę w rozpoznawanie szyfrogramu" (nawet nie przeprowadzając wcześniej żadnego ataku), gdyż sama może policzyć szyfrogramy wybranych przez siebie wiadmości. W zwiazku z tym należy jednak założyć, że $\mathcal{E}$ jest algorytmem probabistycznym (tylko jak go wtedy użyć do podpisu?).

2. Ewa może z ławością sfabrykować parę poprawną parę (wiadomość, podpis): $(\mathcal{E}(e,s),s)$, gdzie s jest dowolnie wybraną wartością. Co prawda $\mathcal{E}(e,s)$ z ogromnym prawdodobieństwem będzie bełkotem, jednak zgonie z definicjami z Rozdziału 2.3 oznacza to, że schemat jest złamany. Rozwiązaniem tego problemu jest wymaganie by wiadomość przed podpisaniem została zapisana w specjalnym formacie. Więcej na ten temat później.

Nie podamy w tej chwili formalnych definicji bezpieczeństwa.

4 Rzut oka na wprowadzone pojecia

Poniższa tabela przedstawia systematyczne zestawienie wprowadzonych pojęć:

	Tajność	Integralność
symetryczne	szyfry	schematy uwierzytelniania
asymetryczne	szyfry z kluczem publicznym	schematy podpisu

Oczywiście każdy szyfr z kluczem publicznym może być użyty jako zwykły szyfr (podobnie ma się rzecz z zapewnianiem integralności). Powodem dla którego szyfry symetryczne (oraz schematy uwierzytelniania) są szeroko stosowane w praktyce jest ich wysoka wydajność w porównaniu z szyframi asymetrycznymi.

5 RSA

Wprowadzimy teraz definicję najbardziej popularnego szyfru z kluczem publicznym. Jest to zarazem historycznie pierwszy opublikowany szyfr tego rodzaju. Pochodzi on z pracy [RSA78]. Wcześniej był on znany niektórym tajnym instytucjom rządowym[3].

5.1 Parę faktów z teorii liczb

Aby pokazać przykład kryptosystemu asymetrycznego potrzebujemy krótkiej powtórki z teorii liczb. Zakładamy przy tym, że uczestnikom wykładu znane jest pojęcie grupy.

Fakt 2 *Istnieją wydajne metody sprawdzania czy dana liczba jest pierwsza.*

Najbardziej wydajnym testem na pierwszość jest (zrandomizowany) algorytm Millera-Rabina (znajdują się one np. [GB], Rozdział C). Ma on niezerowe (choć zaniedbywalne) prawdopodobieństwo błędu. Niedawno pojawił się deterministyczny algorytm [AKS]. Do celów praktycznych jest on nieprzydatny.

[3] Patrz np. http://tinyurl.com/4a668.

Hipoteza 3 *Faktoryzacja iloczynów dużych liczb pierwszych jest trudna obliczeniowo, formalnie mówiąc dla dowolnego WAP A*

$$P\left[A(pq) = p\right]$$

(gdzie p i q są losowymi liczbami pierwszymi długości i), jest zaniedbywalne (jako funkcja i).

Fakt 4 *Dla dowolnych całkowitych a, b istnieją takie całkowite a' i b', że $a'a + b'b = \gcd(a, b)$. Co więcej a' i b' można wydajnie policzyć (uogólnionym algorytmem Euklidesa).*

Dla $n \in \mathbf{N}$ symbol Z_n oznacza zbiór $0, \ldots, n-1$, zaś symbol Z_n^* oznacza zbiór tych elementów z Z_n, które są względnie pierwsze z n. Moc tego zbioru jest równa $\varphi(n)$, gdzie n jest funkcją Eulera. Oczywiście jeśli p jest pierwsza, to $Z_p = Z_p \setminus \{0\}$.

Fakt 5 *Dla dowolnego $n \in \mathbf{N}$ zbiór Z_n^* jest grupą (abelową) z mnożeniem modulo n.*

Twierdzenie 6 (Euler 1736) *Dla dowolnego $n \in \mathbf{N}$ i dowolnego a takiego, że zachodzi $\gcd(n, a) = 1$ mamy $a^{|Z_n^*|} = 1$.*

(Jeśli n jest liczbą pierwszą, to powyższy fakt nosi nazwę *Małego Twierdzenia Fermata.*) Z Faktu 4 dostajemy:

Fakt 7 *Istnieje wydajny algorytm znajdujacy odwrotosci w Z_n^*.*

Dowód. Za pomocą uogólnionego algorytmu Euklidesa możemy obliczyć x' i n' takie, że $x \cdot x' + n \cdot n' = 1$. Jak łatwo widać $x \cdot x' = 1 \pmod{n}$. Zatem $x' \bmod n$ jest odwrotnością x w Z_n^*. □

Fakt 8 *Liczby w Z_n^* można wydajnie podnosić do potęgi. Konkretnie istnieje algorytm podnoszący a do potęgi b modulo n, wydajny ze względu na sumaryczną długość a, b i n.*

Definicja 9 *Niech G i H będą grupami z operacją "$\cdot$". Funkcję $f : G \to H$ nazywamy* homomorfizmem *jeśli dla dowolnych $a, b \in G$ mamy $f(a \cdot b) = f(a) \cdot f(b)$. Grupy G i H są* izomorficzne, *jeśli istnieje między nimi homomorfizm, który jest bijekcją.*

(Intuicyjnie: grupy izomorficzne mają identyczną strukturę.) Dla dowolnych grup G i H (z operacją "$\cdot$" i elementami neutralnymi 1_G i 1_H, odpowiednio), przez $G \times H$ oznaczamy grupę

- z nośnikiem równym iloczynowi kartezjańskiemu nośników G i H
- z operacją "$\cdot$" zdefiniowaną przez operacje grupy G i H wzięte po przekątnych, to znaczy
$$(g, h) \cdot (g', h') = (g \cdot g', h \cdot h')$$
- z elementem neutralnym równym $(1_G, 1_H)$.

Twierdzenie 10 (Chińskie twierdzenie o resztach) *Niech $m_1, \ldots, m_k$ będą parami względnie pierwsze. Niech $m = m_1 \cdot \cdots \cdot m_k$. Wówczas*

1. *grupy Z_m i $Z_{m_1} \times \cdots \times Z_{m_k}$ są izomorficzne.*
2. *Podobnie grupy Z_m^* i $Z_{m_1}^* \times \cdots \times Z_{m_k}^*$.*

Izomorfizm ten jest zdefiniowany przez

$$f(n) := (n \bmod m_1, \ldots, n \bmod m_k)$$

Jest on wydajnie obliczalny w obie strony.

Dowód jest łatwy. Pomijamy go.

5.2 RSA

Schemat szyfrowania RSA zdefiniwowany jest następująco:

Generacja klucza Losujemy dwie różne liczby pierwsze p i q długości i.[4] Niech n (nazywane *modułem RSA*) będzie równe $p \cdot q$.

Bezpieczeństwo szyfru opiera się na trudności rozkładu dużych liczb naturalnych na czynniki pierwsze (Fakt 3). Będziemy działać w grupie Z_n^*. Funkcja szyfrująca którą zdefiniujemy będzie permutacją na Z_n^*, (tzn. bijekcją $Z_n^* \to Z_n^*$). Zarówno szyfrogramy jak i teksty jawne muszą być elementami Z_n^*. (Mamy $|Z_n^*| = (p-1)(q-1)$ zatem szansa, że losowa liczba jest elementem Z_n^* jest zaniedbywalna.)

Losujemy $e \in Z_{\phi(n)}^*$, tzn. takie, że $\gcd(e, \varphi(n)) = 1$ i znajdujemy (uogólnionym algorytmem Euklidesa) d takie, że $ed = 1 \bmod \varphi(n)$,

Kluczem publicznym jest (n, e). Kluczem prywatnym jest (n, d).

Funkcja szyfrująca

$$\mathcal{E}_{n,e}^{\mathrm{RSA}}(x) = x^e \pmod{n}.$$

Funkcja odszyfrowująca

$$\mathcal{D}_{n,d}^{\mathrm{RSA}}(x) = x^d \pmod{n}$$

[4] Takie losowanie może polegać po prostu na losowaniu dowolnych liczb naturalnych długości i aż do znalezienia liczby pierwszej (co może być wydajnie stwierdzone zgodnie z Faktem 2). Oczekiwany czas działania takiej metody jest znośny ponieważ liczby pierwsze występują dość często. Konkretnie: prawdopodobieństwo, że dana liczba długości i jest pierwsza jest (asymptotycznie) odwrotnie proporcjonalne do i (przy czym stała jest rozsądna). Więcej na ten temat można przeczytać np. na stronie mathworld.wolfram.com/PrimeNumberTheorem.html.

Łatwo sprawdzić, że dla dowolnego $x \in Z_n^*$

$$\begin{aligned}
\mathcal{D}_{n,d}^{\mathrm{RSA}}\left(\mathcal{E}_{n,e}^{\mathrm{RSA}}(x)\right) &= (x^e)^d \\
&= x^{ed} \\
&= x^{i\cdot\varphi(n)+1} \\
&= \left(x^i\right)^{\varphi(n)} \cdot x \\
&= 1^i \cdot x \qquad (1) \\
&= x \pmod{n} \qquad (2)
\end{aligned}$$

(dla pewnego i naturalnego), gdzie (1) wynika z Małego Twierdzenia Fermata. W praktyce szyfr RSA stosuje się do szyfrowania dowolnych liczb naturalnych mniejszych od n (czyli Z_n). Jest to bezpieczna praktyka z dwóch powodów. Po pierwsze, szanse, że $x \in Z_n \setminus Z_n^*$ są bardzo małe. Znalezienie takiego x pozwala zresztą na faktoryzację n (gdyż $\gcd(x,n)$ jest wtedy równe jednej z liczb p i q). Po drugie równość (2) zachodzi w gruncie rzeczy dla dowlnego $x \in Z_n$, mamy boweim następujący fakt.

Lemat 11 *Równość* $x^{ed} = x \pmod{n}$ *zachodzi także dla* $x \in Z_n \setminus Z_n^*$.

Dowód.*[Trochę inaczej niż na wykladzie]* Złóżmy (bez straty ogólności), że x jest wielokrotnością p. Wówczas

$$x^{ed} = x \pmod{p} \qquad (3)$$

(bo obie strony są podzielne przez p). Ponadto

$$\begin{aligned}
x^{ed} &= x^{i\cdot\varphi(n)+1} \\
&= \left(x^{i(p-1)}\right)^{q-1} \cdot x \\
&= 1 \cdot x \pmod{q} \qquad (4) \\
&= 1 \pmod{q}. \qquad (5)
\end{aligned}$$

gdzie i jest pewną liczbą naturalną a 4) wynika z małego Twierdzenia Fermata. Z Chińskiego Twierdzenia o Resztach (3) i (5) dają nam: $x^{ed} = x \pmod{n}$ (bo p i q są różnymi liczbami pierwszymi). □

5.3 Związek między trudnością faktoryzacji a bezpieczeństwem RSA

Złamanie RSA jest co najwyżej tak trudne jak faktoryzacja iloczynów dużych liczb pierwszych.

Fakt 12 *Jeśli faktoryzacja iloczynów dużych liczb pierwszych jest łatwa (czyli Hipoteza 3 nie jest prawdziwa), to RSA nie jest bezpieczne.*

Dowód. Znajmość p i q pozwala natychmiast obliczyć $\varphi(n)$. Wówczas na podstawie e możemy obliczyć d. □

Nie jest znany dowód implikacji w drugą stronę. Natomiast można pokazać, że:

Fakt 13 *Jeśli faktoryzacja iloczynów dużych liczb pierwszych jest trudna obliczeniowo, to trudne jest też obliczenie e na podstawie (n,e).*

Dowód przebiega na zasadzie redukcji: pokazujemy, że mając dany wydajny algorytm obliczenia e na podstawie (n,e) potrafimy skonstruować wydajny algorytm faktoryzacji. Ten fakt został pokazany już w oryginalnej pracy [RSA78].

Literatura

[AKS] M. Agrawal, N. Kayal, and N. Saxena. PRIMES is in P. dostępne na stornie `http://www.cse.iitk.ac.in/users/manindra/`.

[DH76] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.

[GB] S. Goldwasser and M. Bellare. Lecture notes in cryptography. dostępne pod adresem `http://www.cs.ucsd.edu/users/mihir/papers/gb.html`.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, February 1978.