Piotr KAPCIA*

KRYPTOSYSTEMY OPARTE NA PROBLEMACH TRUDNYCH OBLICZENIOWO Z WYSZCZEGÓLNIENIEM PROBLEMU FAKTORYZACJI LICZB CAŁKOWITYCH

STRESZCZENIE

Publikacja przedstawia analizę złożoności obliczeniowej problemów matematycznych, na których opiera się konstrukcja, a zarazem bezpieczeństwo kryptosystemów klucza publicznego. Jednocześnie, na podstawie aktualnej wiedzy z obliczeniowej teorii liczb oraz informacji o możliwościach nowoczesnej techniki obliczeniowej, przedyskutowana została skuteczność algorytmów wykorzystywanych do rozwiązywania tych problemów.

***Słowa kluczowe**: bezpieczeństwo kryptosystemów klucza publicznego, złożoność obliczeniowa, faktoryzacja liczb całkoitych, logarytm dyskretny, logarytm dyskretny na krzywych eliptycznych*

CRYPTOSYSTEMS BASED ON DIFFICULT COMPUTATIONAL PROBLEMS WITH SPECIFICATION OF THE PRIME FACTORIZATION PROBLEM

This paper presents analysis of computational complexity of mathematical problems on which construction and security of the public key cryptosystems could be based. Additionally on the basis of current knowledge sourcing from computational number and information theory, regarding possibilities of modern computational techniques, it has been discussed effectiveness of algorithms solving these problems.

***Keywords**: security of public key cryptosystems, computational complexity, prime factorization, discrete logarithm, elliptic curve discrete logarithm*

1. WSTĘP

Istotą kryptosystemu z **kluczem publicznym** (inaczej **kluczem jawnym**) jest zastosowanie takiego algorytmu kryptograficznego, dla którego proces ustalenia **klucza prywatnego** i przełamania szyfrogramu, przy znanym schemacie szyfrowania, powinien być procesem obliczeniowo niewykonalnym. Przy takim założeniu schemat i klucz do szyfrowania można ujawnić, choćby publikując go w książce telefonicznej, gazecie czy na stronie internetowej, gdzie staje się publicznie dostępny. Od 1977 roku (realizacji kryptosystemu RSA, którego autorami są: Rivest, Shamir, Adleman) wymyślono kilka systemów szyfrujących z kluczem publicznym, których bezpieczeństwo, wbrew pozornie zagmatwanym procedurom szyfrowania, opiera się tylko na kilku prostych do zdefiniowania, ale ciągle nie rozwiązanych tzw. jednokierunkowych problemach matematycznych. Wspólną cechą takich problemów jest ich duża złożoność obliczeniowa. Dla przykładu, dowolną liczbę naturalną $n > 1$ można wyrazić jednoznacznie w postaci $n = p_1 p_2 ... p_t$, gdzie $p_1 \le p_2 \le ... \le p_t$, a każde p_k jest liczbą pierwszą [9]. Wymnożenie czynników p_k jest zadaniem banalnie prostym dla zwykłego komputera domowego. Niestety, ani do znalezienia tego rozkładu liczby n na czynniki pierwsze (szczególnie, gdy $n = p_1 \cdot p_2 > 10^{200}$, a p_1 i p_2 są podobnego rzędu), ani do rozstrzygnięcia, czy n jest liczbą pierwszą, czy też nie jest, nie istnieją algorytmy komputerowe zwracające żądaną odpowiedź w sensownym czasie. Nie może tu przyjść nam z pomocą nawet nowoczesna technika obliczeniowa dużej skali, w której wykorzystuje się wyspecjalizowane superkomputery wielkich mocy obliczeniowych, zdolne do przetwarzania równoległego oraz miliony rozproszonych komputerów podłączonych do Internetu [35]. Należy tu przede wszystkim wziąć pod uwagę, iż zwiększająca się gęstość przechowywania danych i czas niezbędny do ich przetwarzania ciągle napotyka na ograniczenia natury fizycznej, których nie da się obejść, niezależnie od włożonego w to wysiłku. Na przykład według obecnej wiedzy pamięć o pojemności 10^{60} bitów musiałaby mieć masę całego Układu Słonecznego, a wykonanie 10^{70} operacji wymagałoby więcej czasu, niż upłynęło od Wielkiego Wybuchu (który miał miejsce około 10^{18} sekund temu), nawet jeśli każda operacja trwałaby nie dłużej niż 10^{-43} s [1]. Jest to tzw. czas Plancka, czyli najkrótszy, zdaniem współczesnej fizyki, sensowny przedział czasu.

Podane wyżej rzędy wielkości liczb – jak w dalszych rozdziałach się okaże – są ściśle związane z algorytmami klucza publicznego, które klasyfikuje się właśnie w zależności od ich obliczeniowej złożoności czasowej lub przestrzennej, wyrażonej jako funkcja argumentu n (gdzie n jest rzędem używanych liczb). Za przykład może posłużyć klasyczny problem znalezienia funkcji odwrotnej do **funkcji jednokierunkowej** (jak się zaraz okaże, odgrywającej zasadniczą rolę w kryptografii), którego nie udaje się rozwiązać z powodu braku czasu i przestrzeni do składowania danych, tzn. z powodu złożoności czasowej i złożoności danych. Łatwo więc tu zauważyć, iż kryptosystemy klucza publicznego nie zapewniają bezwarunkowego bezpieczeństwa, które opiera się przede wszystkim na aktualnej wie-

* Doktorant na Wydziale EAIiE AGH

dzy z matematyki teoretycznej (zwłaszcza teorii liczb) i możliwościach technologicznych dotyczących rozwiązania danego problemu obliczeniowego.

Postęp technologiczny przesuwa ciągle linię graniczną oddzielającą rzeczy „oporne" od „efektywnych". Obecnie mniej więcej co dwa lata podwaja się szybkość najszybszego komputera i mniej więcej co 15 miesięcy ceny komputerów dwukrotnie maleją [1]. Ten drugi trend pozwala na upowszechnianie przetwarzania równoległego, jak również rozproszonego. Postępowi technologicznemu kryptologia przeciwstawia się jednak bardzo łatwo, zwiększając odpowiednio parametry szyfrowania, które zapobiegają przejściu przez bariery kryptoanalityczne. Na przykład pewna metoda odwracania kryptograficznej funkcji jednokierunkowej polega na znalezieniu rozkładu danej liczby n na czynniki pierwsze. Zadanie to (wyżej już przedstawione) jest niesłychanie pracochłonne w porównaniu z zagadnieniem wymnożenia danych czynników tworzących liczbę n. Wystarczy więc zwiększyć parametr n przez zwiększenie rzędu wielkości czynników go tworzących, aby wzrósł czas obliczeń „odwracających", który to jest głównym kryterium oceny bezpieczeństwa algorytmu szyfrującego pod względem odporności na kryptoanalizę.

Od co najmniej kilku lat zaczęto pokładać nadzieję w komputerach kwantowych, przy których użyciu rozwiązywanie problemów obliczeniowo trudnych, stosowanych m.in. w algorytmach kryptograficznych z kluczem publicznym, można by przeprowadzać w czasie zaledwie kilku minut, a nawet sekund [4, 24]. Również pojawiły się propozycje alternatywnych rozwiązań do komputerów kwantowych. Od 2002 roku zaczęły ukazywać się publikacje poruszające temat sprzętowej faktoryzacji, która pozwalałaby znacząco skracać czas łamania kluczy kryptosystemu RSA. Podstawą rozważań stał się, opublikowany pod koniec 2001 roku, artykuł naukowy, w którym zostało przeanalizowane zagadnienie faktoryzacji za pomocą masowej równoległości wykorzystywanej w tzw. „maszynie Bernsteina" – urządzeniu, którego teoretyczne podstawy opracował Daniel Bernstein [2, 4]. Pomysł ten zainspirował dwóch innych naukowców, Adiego Shamira (jednego z twórców algorytmu RSA) oraz Erana Tromera. Wykorzystując wyniki prac nad poprzednimi projektami (w tym nad optoelektronicznym urządzeniem TWINKLE) oraz możliwości nowoczesnej technologii półprzewodnikowej, opracowali oni projekt maszyny TWIRL (skrót od nazwy The Weizmann Institute Relation Locator), która wykonywałaby najtrudniejszy obliczeniowo fragment najefektywniejszego znanego dziś algorytmu rozkładania liczb na czynniki „ogólnego sita ciała liczbowego" (wykorzystanego do złamania w 1999 r. klucza 512-bitowego w konkursie RSA) [22, 26, 27, 37]. Jednak z artykułów tych wynika, iż pomysły te są bardzo dalekie od realizacji, a szansa na skonstruowanie komputera kwantowego lub alternatywnej maszyny tego typu pojawi się nie wcześniej niż za kilkadziesiąt lat.

Wychodząc naprzeciw niedostatecznie wydajnej elektronicznej technice obliczeniowej, zaczęła prężnie rozwijać się nowa gałąź matematyki, a dokładniej teorii liczb, która nazywa się obliczeniową teorią liczb, a wraz z nią również teoria złożoności algorytmów. Głównym celem tej nowej gałęzi matematyki stało się znajdywanie oraz implementacja efektywnych algorytmów komputerowych w celu rozwiązywania różnych teorioliczbowych problemów, szczególnie stosowanych w nowoczesnej kryptologii. Zarysowany wyżej problem faktoryzacji, czyli rozkładania liczb całkowitych na czynniki pierwsze, zastosowany w konstrukcji pierwszego algorytmu asymetrycznego RSA, jest tylko jednym z kilku takich problemów trudnych obliczeniowo wykorzystywanych w kryptografii z kluczem publicznym. W celu pełnego zobrazowania aktualnego stanu wiedzy na temat bezpieczeństwa kryptosystemów obliczeniowo trudnych, w pracy zostały również przedstawione najważniejsze osiągnięcia obliczeniowej teorii liczb.

Do takich osiągnięć zalicza się przede wszystkim algorytmy, które stosowane są do rozwiązywania wszystkich problemów obliczeniowych, wykorzystywanych w kryptosystemach asymetrycznych, w których skład wchodzą:

- faktoryzacja liczb całkowitych,
- obliczanie logarytmu dyskretnego w ciałach skończonych,
- obliczanie logarytmu dyskretnego na krzywych eliptycznych.

2. ZŁOŻONOŚĆ OBLICZENIOWA

Powszechnie używaną miarą efektywności dowolnego algorytmu jest złożoność czasowa. Określamy ją jako funkcję $f(n)$, jeżeli dla każdego n i każdych danych wejściowych o długości n wykonanie algorytmu odbywa się najwyżej w n krokach. Dla danego rozmiaru danych wejściowych i danej szybkości procesora złożoność czasowa jest czynnikiem ograniczającym czas wykonania algorytmu. Funkcja $f(n)$ jest najczęściej nazywana **rzędem wielkości** wyrażenia $O(g(n))$ (zwanego notacją **wielkiego** O), przy czym $f(n) = O(g(n))$ oznacza, że istnieją takie stałe c i n_0, że $|f(n)| \le c|g(n)|$ dla $n \ge n_0$ [21, 29]. Mamy tu jednak do czynienia z kilkoma dwuznacznościami.

Po pierwsze definicja kroku nie jest precyzyjna. Krok może być pojedynczą instrukcją komputera jednoprocesorowego lub pojedynczą instrukcją języka wysokiego poziomu itd. Te różne definicje kroku są ze sobą powiązane multiplikatywną stałą c. Dla bardzo dużych wartości n stałe te nie są istotne. Istotna jest szybkość wzrostu względnego czasu wykonania. Na przykład, jeżeli zastanawiamy się, czy dla kryptosystemu RSA zastosować klucz 300-cyfrowy czy 350-cyfrowy, nie musimy (jest to zresztą niemożliwe) wiedzieć dokładnie, ile czasu zajęłoby złamanie klucza o danym rozmiarze. Interesują nas raczej szacunkowe dane na temat liczby prób oraz o ile więcej prób potrzeba dla większego rozmiaru klucza.

Po drugie, ogólnie rzecz biorąc, nie potrafimy określić dokładnej formuły na $f(n)$. Potrafimy uzyskać jej przybliżenie. Jesteśmy jednak przede wszystkim zainteresowani szybkością zmian $f(n)$, gdy n staje się bardzo duże.

W celu analizy algorytmów stosowanych w kryptosystemach opartych na problemach trudnych obliczeniowo należy wprowadzić niezbędne definicje oraz przeprowadzić klasyfikację tych problemów. Definiowane pojęcia zostały zaczerpnięte z prac [10, 31, 32, 33].

Definicja 2.1

Problem decyzyjny P należy do klasy NP, jeśli dla każdego przypadku problemu P osoba dysponująca nieograniczoną mocą obliczeniową może nie tylko odpowiedzieć na pytanie będące treścią problemu, ale w przypadku, gdy odpowiedź brzmi „tak", może dostarczyć dowodu, którego inna osoba może użyć do sprawdzenia poprawności odpowiedzi w czasie wielomianowym. Taki dowód poprawności odpowiedzi „tak" nazywamy **certyfikatem o czasie wielomianowym**.

Prawie na pewno klasa NP jest znaczenie większa niż klasa P (zawierająca problemy rozwiązywalne w sposób deterministyczny za pomocą algorytmów w czasie wielomianowym), ale nie zostało to udowodnione. Stwierdzenie $P \neq NP$ jest najsłynniejszą hipotezą z teoretycznych podstaw informatyki.

Do klasy P należy na przykład problem potęgowania modularnego oraz problem wyznaczania pierwiastków kwadratowych modulo liczba pierwsza, a do problemów klasy NP należy m.in. problem logarytmowania oraz pierwiastkowania dyskretnego.

Mówi się, że jeśli problem Q jest z klasy NP, to można go rozwiązać w sposób deterministyczny w czasie wykładniczym, czyli istnieje wielomian $p(n)$ taki, że problem Q można rozwiązać za pomocą algorytmu deterministycznego w czasie $O(c^{p(n)})$, przy czym $c > 1$ jest pewną stałą.

Należy jednak mieć świadomość tego, iż algorytm z czasem działania n^{100}, gdzie n jest długością danych, jest wolniejszy od algorytmu z czasem działania $e^{n/1000}$, dopóty n jest mniejsze od około dziesięciu milionów. Mimo to pierwszy algorytm ma czas wielomianowy, podczas gdy drugi ma czas wykładniczy.

Definicja 2.2

Problem decyzyjny P nazywamy problemem klasy co-NP, jeśli P spełnia warunek analogiczny do powyższego, w którym odpowiedź „tak" zamieniono na „nie". Oznacza to, że dla każdego przypadku P, dla którego odpowiedź brzmi „nie", istniej certyfikat o czasie wielomianowym dowodzący poprawności odpowiedzi „nie".

Dla przykładu rozpatrzmy problem faktoryzacji.

Dane: liczby naturalne N i k.

Pytanie: czy N ma dzielnik w przedziale $[2, k]$?

Problem ten prawie na pewno nie jest problemem klasy P. Jest on jednak w klasie NP. Przypuśćmy mianowicie, że odpowiedni superkomputer rozkłada N na czynniki pierwsze i znajduje dzielniki $M \in [2, k]$. Następnie informuje nas, że właściwą odpowiedzią jest „tak" i dostarcza nam M. Mając tę informację, możemy sprawdzić poprawność odpowiedzi w czasie wielomianowym, po prostu wykonując dzielenie N przez M.

Problem faktoryzacji jest również klasy co-NP, chociaż stwierdzenie tego wymaga bliższego przyjrzenia się sprawie. Jeśli odpowiedź brzmi „nie", to superkomputer daje nam pełen rozkład N na czynniki pierwsze, z którego możemy natychmiast zobaczyć, że nie ma czynnika pierwszego $\leq k$. Wraz z rozkładem musi nam dostarczyć certyfikat, który pozwoli nam sprawdzić w czasie wielomianowym, że każdy czynnik jest rzeczywiście liczbą pierwszą.

Można więc powiedzieć, iż problem rozkładu liczb na czynniki pierwsze należy do zbioru NP $\cup$ co-NP.

Definicja 2.3

Problem decyzyjny P klasy NP nazywamy problemem NP-zupełnym, jeśli każdy inny problem Q klasy NP można zredukować do problemu P w czasie wielomianowym.

Inaczej, gdyby ktoś znalazł algorytm wielomianowy dla pewnego problemu NP-zupełnego P, to istniałyby też algorytmy wielomianowe dla wszystkich problemów Q klasy NP. Oznaczałoby to, że klasa NP pokrywa się z klasą P i hipoteza $P \neq NP$ jest fałszywa. Z tego powodu nikt nie oczekuje pojawienia się algorytmu wielomianowego dla jakiegokolwiek problemu NP-zupełnego. W pewnym sensie problemy NP-zupełne są problemami najtrudniejszymi w klasie NP. Jednak stwierdzenie to należy potraktować ostrożnie. Nie jest wykluczone, że można podać efektywny algorytm – nawet algorytm wielomianowy – który rozwiązuje większość przypadków pewnego problemu NP-zupełnego. Nie przeczy to jednak hipotezie $P \neq NP$.

W praktyce jednak zazwyczaj bardzo trudno rozwiązać większe przypadki problemu NP-zupełnego. Często wraz ze wzrostem długości danych czas obliczeń wszystkich znanych algorytmów rośnie wykładniczo.

Problemem NP-zupełnym jest na przykład problem wyznaczania w grafie cyklu Hamiltona oraz problem plecakowy, zwany problemem upakowania.

Z kryptograficznego punktu widzenia bardzo ważną klasą, jest klasa UP. Jest to klasa problemów niedeterministycznych, rozwiązywalnych za pomocą takich algorytmów wielomianowych, że dla każdej danej wejściowej istnieje co najwyżej jedno rozwiązanie. Dla ścisłego zapisu należy wprowadzić definicję tej klasy.

Definicja 2.4

Klasa UP (*unique P*) składa się z problemów klasy NP o tej własności, że dla każdego przypadku, w którym odpowiedź jest twierdząca, istniej przepis na jednoznacznie wyznaczony certyfikat wielomianowy.

Każda funkcja różnowartościowa $f: X \rightarrow Y$, dla której wartości można obliczać w czasie wielomianowym, dostarcza odpowiedniego problemu klasy UP.

Dane: $y \in Y$.

Pytanie: czy istnieje $x \in X$ takie, że $f(x) = y$?

Certyfikat dla odpowiedzi „tak" stanowi po prostu jedyne x, dla którego $f(x) = y$. Można pokazać, że **jednokierunkowa funkcja** szyfrująca w kryptografii z kluczem publicznym istniej wtedy i tylko wtedy, gdy klasa UP jest istotnie większa od klasy P, czyli musi zachodzić $P \neq UP$.

Jest jasne, że P $\subset$ UP $\subset$ NP, ale o żadnej z tych inkluzji nie udowodniono, że jest właściwa. Oczywiście dowód nierówności P $\neq$ UP lub UP $\neq$ NP pociągałby za sobą dowód fundamentalnej hipotezy P $\neq$ NP.

W następnym rozdziale obszerniej zostanie opisana funkcja jednokierunkowa, która odgrywa zasadniczą rolę w określeniu publicznego systemu kryptograficznego. Jak dotąd nie udowodniono, że istnieje choćby jedna funkcja jednokierunkowa, chociaż co do niektórych, jak mnożenie liczb pierwszych czy potęgowanie modularne, zakłada się, że tak jest.

3. PROBLEMY TRUDNE OBLICZENIOWO STOSOWANE W KRYPTOSYSTEMACH ASYMETRYCZNYCH

Własnością, jakiej oczekuje się od szyfru, jest dowód, że jego złamanie jest tak trudne jak rozwiązanie pewnego problemu obliczeniowego, który ogólnie uważany jest za trudny, np. jak problem odwracania funkcji jednokierunkowej, związany z problem faktoryzacji oraz problemem logarytmowania dyskretnego. W celu dokładnego zrozumienia tych zagadnień należy wprowadzić kolejne pojęcia i definicje.

Definicja 3.1

Szyfr obliczeniowo bezpieczny to taki szyfr, którego kryptoanaliza w sposób deterministyczny jest obliczeniowo trudna, a więc możliwa wyłącznie w czasie wykładniczym, a niewykonalna w czasie wielomianowym. Czyli problem kryptoanalizy jest z klasy NP [12].

Należy jednak podkreślić, że złożoność obliczeniowa, tak jak się ją rozumie powyżej, jest złożonością pesymistyczną, tj. rozpatrywaną dla najgorszego przypadku. Nawet gdyby rozpatrywać złożoność przeciętną, to w zbiorze rozpatrywanych przypadków mogą znaleźć się przypadki łatwe obliczeniowo. Wynika stąd, że aby problem był odpowiedni do wykorzystania w konstrukcji szyfru, to powinien być trudny dla prawie wszystkich przypadków. Jednak doświadczenie uczy, że system kryptograficzny zbudowany w oparciu o problem nawet tak trudny jak NP-zupełny, może okazać się możliwy do złamania w czasie wielomianowym [32].

Definicja 3.2

Funkcja różnowartościowa $f: X \rightarrow Y$ nazywana jest **funkcją ściśle jednokierunkową**, jeśli spełniony jest poniższy warunek.

Istnieje efektywna metoda obliczeniowa $f(x)$ przy dowolnym $x \in X$, ale nie istnieje żadna efektywna metoda wyznaczania x z zależności $y = f(x)$ przy dowolnym znanym $y \in f[X]$. Efektywna metoda, tzn. taka, której pracochłonność zależy wielomianowo od $\log |X|$ [1, 21].

Dobrym obrazowym przykładem jest rozbicie talerza na kawałki. Z jednej strony zadanie jest bardzo łatwe, ponieważ rozbicie talerza na tysiące malutkich części jest zadaniem bardzo prostym. Z drugiej strony złożenie tych mikroskopijnych części w cały talerz jest zadaniem znacznie trudniejszym. Dowód istnienia funkcji ściśle jednokierunkowych jest ograniczony brakiem dostatecznie dobrych dolnych oszacowań dla znanych metod. Mimo to pewne funkcje bazujące na mnożeniu liczb pierwszych i potęgowaniu modularnym są dobrymi „kandydatami" na funkcje jednokierunkowe.

Przykładem funkcji jednokierunkowej bez zapadki (termin ten zostanie wyjaśniony w dalszej części) jest już wspominane mnożenie liczb pierwszych. Jest rzeczą względnie prostą pomnożenie dwóch liczb o dziesięciu tysiącach cyfr, a więc także dwóch liczb pierwszych o tej samej wielkości – na komputerze domowym trwa to maksymalnie kilka sekund. Jednak w chwili obecnej nie istnieje (publicznie) znana metoda efektywnego rozkładu 300-cyfrowej liczby dziesiętnej na czynniki pierwsze (poza szczególnymi przypadkami) [22].

Czyli w zapisie formalnym mamy: Niech x_1, x_2 są liczbami pierwszymi i $K \leq x_1 \leq x_2$ dla dostatecznie dużego K, funkcja zdefiniowana jako $f(x_1, x_2) = x_1 \cdot x_2$ jest wówczas funkcją jednokierunkową. Nie są dla niej znane żadne zapadki [1].

Kolejnym ww. przykładem funkcji jednokierunkowej bez zapadki jest potęgowanie w arytmetyce modularnej. Niech p będzie liczbą pierwszą. Dla ustalonego g funkcja g-wykładnicza w ciele Galoisa GF(p) zdefiniowana jako

$$f(n) = g^n \bmod p$$

jest, dla dostatecznie dużych p i g, funkcją jednokierunkową [1, 10, 11].

Inną funkcją jednokierunkową, która była kiedyś interesująca z kryptograficznego punktu widzenia, jest funkcja, która nie pochodzi z arytmetyki klas residualnych, ale związana jest z problemem plecakowym, zaliczającym się do zagadnień programowania całkowitego [12].

Funkcje ściśle jednokierunkowe nie mogą być jednak użyte bezpośrednio w charakterze funkcji (algorytmów) szyfrujących, ponieważ uprawniony odbiorca wiadomości nie mógłby jej zdeszyfrować. Narzędziem odpowiedniejszym dla tego celu jest funkcja zapadkowa.

Definicja 3.3

Funkcja różnowartościowa $f: X \rightarrow Y$ jest nazywana **funkcją jednokierunkową z zapadką**, jeśli spełnione są następujące warunki:

- istnieje efektywna metoda obliczenia $f(x)$ przy dowolnym $x \in X$,
- istnieje efektywna metoda obliczenia $f^{-1}(y)$ przy dowolnym $y \in f[X]$, ale nie da się jej efektywnie otrzymać z zależności $y = f(x)$ dla każdego $y \in f[X]$, potrzebna jest tajna dodatkowa informacja, nosząca nazwę zapadki [1, 30].

Problem zapadki pyta o odwrócenie konstrukcji matematycznej, na której jest oparta zapadka kryptosystemu z kluczem publicznym. Jeżeli znajdziemy efektywny algorytm dla problemu zapadki, to będziemy również dysponować efektywnym algorytmem dla problemu złamania kryptosystemu [10]. Implementacja odwrotna niekoniecznie musi być prawdziwa (a nawet jeśli jest prawdziwa, to może być

trudna do udowodnienia). Oznacza to, że mogą być inne sposoby złamania systemu niż bezpośrednie badanie funkcji zapadkowej.

Zagadnienie funkcji jednokierunkowych ma również ścisły związek z ideą **szyfru wykładniczego**, a klasycznym jego przykładem jest kryptosystem RSA.

Definicja 3.4

Szyfr nazywamy **wykładniczym**, jeśli wiadomość jawna $m \in \{0, 1,..., n-1\}$ jest szyfrowana w taki sposób, że $c = m^e \bmod n$, a deszyfrowana dla $c \in \{0, 1,..., n-1\}$, zgodnie z zasadą $m = c^d \bmod n$, gdzie $k_e = (e, n)$ jest kluczem szyfrującym, a $k_d = (d, n)$ jest kluczem deszyfrującym.

Jeżeli n jest iloczynem dwóch liczb pierwszych, to zasada deszyfrowania posiada zapadkę, którą jest rozkład n na czynniki pierwsze [8, 17].

W procedurze szyfrowania RSA liczbę n, będącą modułem, wybiera się w taki sposób, aby była ona iloczynem dwóch dużych liczb pierwszych p oraz q. Bezpieczeństwo systemu RSA opiera się na założeniu, że funkcja szyfrująca jest jednokierunkowa, a więc zadanie deszyfrowania tekstu zaszyfrowanego jest obliczeniowo niewykonalne, dla kogokolwiek oprócz odbiorcy, który (do jej „odwrócenia") posiada tajną informację, czyli klucz deszyfrujący k_d. Tak więc długość (rząd wielkości) modułu n gwarantuje bezpieczeństwo RSA. Powszechnie uznaje się, że złamanie kryptosystemu RSA jest tak trudne jak faktoryzacja modułu n, jednak formalnego dowodu na to nie ma [3, 10, 11]. Można więc powiedzieć, że nie ma dowodu na to, że problem rozwiązywalny za pomocą algorytmu RSA leży w klasie NP\P [31].

Rodzimy do RSA szyfr Rabina był pierwszym przykładem szyfru asymetrycznego, w którego przypadku dowiedziono, że jest bezpieczny: znalezienie przez intruza tekstu jawnego na podstawie znajomości szyfrogramu jest obliczeniowo równoważne faktoryzacji. Problem ten sprowadza się do wyznaczenia pierwiastka kwadratowego modulo n i jest on obliczeniowo równoważny do problemu faktoryzacji n, gdyż są to problemy wzajemnie redukowalne [18, 31, 32].

Według obecnej wiedzy bezpieczeństwo zdefiniowanego szyfru wykładniczego opiera się na dużej złożoności wyznaczania tzw. **logarytmu dyskretnego**. Implikuje to stwierdzenie, iż prawdopodobnie odwrotną do funkcji wykładniczej w arytmetyce modularnej jest jedynie funkcja logarytmowania dyskretnego. Również tu nie wiadomo, czy złożoność obliczeniowa eksponencjalnego sposobu szyfrowania jest równoważna złożoności obliczeniowej logarytmowania dyskretnego [3, 31, 32]. Do grupy algorytmów kryptograficznych z kluczem publicznym, które wykorzystują przy konstrukcji szyfru problem **logarytmu dyskretnego** w ciałach skończonych, należą: DSA (*Digital Signature Algorithm*), algorytm Diffiego – Hellmana oraz ElGamala [20].

Na podstawie przytoczonego wyżej przykładu funkcji jednokierunkowej, czyli potęgowania w arytmetyce modularnej, sformułujemy teraz pojęcie **problemu logarytmu dyskretnego** w ciałach skończonych. Aby zrozumieć na czym polega ten problem, należy się jemu trochę dokładniej przyglądnąć. Obserwacje te będą również potrzebne później, przy definiowaniu problemu obliczania **logarytmu dyskretnego na krzywych eliptycznych**.

Po pierwsze pierwiastek liczby pierwszej p to taki, który podniesiony do potęgi da wszystkie liczby całkowite od 1 do p–1. To znaczy, jeżeli g jest pierwiastkiem liczby pierwszej p, to liczby $g \bmod p$, $g^2 \bmod p$, ... , $g^{p-1} \bmod p$ są różne i są liczbami całkowitymi od 1 do p–1 w pewnej permutacji. Dla dowolnej liczby całkowitej y i pierwiastka g liczby pierwszej p można znaleźć jeden wykładnik x taki, że $y = g^x \bmod p$, gdzie $0 \le x \le p-1$. Wykładnik x nazywamy logarytmem dyskretnym przy podstawie $g \bmod p$ [29].

Na tej podstawie można zdefiniować problem obliczenia **logarytmu dyskretnego**.

Definicja 3.5

Niech $F(p) = \{1, 2, ..., p-1\}$ będzie grupą multiplikatywną liczb całkowitych modulo liczba pierwsza p oraz $g \in F(p)$ będzie ustalonym elementem (naszą „podstawą"). Problem **logarytmu dyskretnego** w $F(p)$ przy podstawie g nazywamy zadanie wyznaczenia dla danego $y \in F(p)$ takiej liczby naturalnej x, że $y = g^x \bmod p$ (jeżeli x istnieje, w przeciwnym wypadku powinniśmy otrzymać jako wynik stwierdzenie, że y nie należy do podgrupy generowanej przez g) [10, 11, 13, 19].

Jak już wiemy, w arytmetyce modularnej potęgowanie jest funkcją jednokierunkową. Znaczy to, że o ile łatwo jest obliczyć jakąś liczbę $y = g^x \bmod p$ dla tajnej wartości x, o tyle znacznie trudniej jest obliczyć x z y, jeśli liczby te są wystarczająco duże, powiedzmy mają długość kilkuset cyfr (zakładamy że g oraz p są niewiadome). Problem ten określa się więc jako trudność obliczenia logarytmu dyskretnego w ciele skończonym, ponieważ x jest logarytmem y o podstawie $g \pmod p$ i są to liczby skończone oraz całkowite (bez ułamków zwykłych lub dziesiętnych). Inaczej mówiąc x jest logarytmem dyskretnym z y przy podstawie g, w ciele reszt modulo p [19].

Efektywność algorytmu DSA, Diffiego–Hellmana oraz ElGamala zależy od stopnia trudności obliczenia logarytmów dyskretny, czyli od rzędu wielkości używanych kluczy w procesie szyfrowania, analogicznie jak w przypadku systemów RSA i Rabina. Obliczenie wartości funkcji wykładniczej, nawet dla wykładników o dużych wartościach, nie jest trudne [36]. Nie istnieje jednak publicznie znany algorytm obliczania wartości logarytmu dyskretnego (dla dużych liczb) w rozsądnym czasie, nawet przy wykorzystaniu obecnie dostępnej całej mocy obliczeniowej komputerów na świecie. Istnieje, co prawda, efektywny algorytm, który wylicza logarytm dyskretny, lecz działa tylko wtedy, gdy $(p-1)/2$ nie jest liczbą pierwszą [37]. Ma to szczególne znaczenie na przykład dla algorytmu ElGamala, w którym przy generowaniu pary kluczy (jawnego i tajnego) musimy wylosować liczbę pierwszą p taką, aby $(p-1)/2$ też była pierwszą, wykluczając w ten sposób przeprowadzenie łatwego ataku na szyfrogram.

Istnieją warianty wymienionych systemów DSA, Diffiego–Hellmana oraz ElGamal oparte na **krzywych eliptycznych** ECM (*Elliptic Curve Method*), które mogą być nawet trudniejsze do złamania niż system bazujące na problemach opisanych powyżej. Zamiast stosować obliczenia na liczbach całkowitych mod *n*, systemy ECC (*Elliptic Curve Cryptosystem*) operują punktami na pewnej krzywej eliptycznej. Precyzyjniej mówiąc, metoda ta wykorzystuje teorię pewnych krzywych algebraicznych trzeciego stopnia (krzywych eliptycznych) na płaszczyźnie rzutowej nad skończonym ciałem K, w szczególności nad ciałem $F(p^k)$. Jest bardzo prawdopodobne, że stosowanie w kryptosystemach metody krzywych eliptycznych daje wyższy poziom zabezpieczeń, przy użyciu zde-cydowanie mniejszych kluczy [3, 7, 11]. Szczególnie interesujące są krzywe eliptyczne w $F(2^k)$, gdyż konstrukcja procesorów arytmetycznych, przeznaczonych do prowadzenia rachunków w odpowiednim ciele, jest łatwa dla dużych *n* [1].

Bezpieczeństwo systemu kryptograficznego zbudowanego na podstawie krzywej eliptycznej zależy od trudności obliczeniowych wyznaczania logarytmów dyskretnych w grupie (E, +) punktów krzywej eliptycznej E. Jest to problem analogiczny do powyższego problemu logarytmu dyskretnego w grupie multiplikatywnej ciała skończonego. Tak samo można zrobić w grupie addytywnej punktów krzywej eliptycznej E zdefiniowanej nad ciałem skończonym F_q [3, 7, 10].

Definicja 3.6

Jeśli E jest krzywą eliptyczną nad ciałem F_q i *B* jest punktem tej krzywej, to problem **logarytmu dyskretnego na krzywej** E (przy podstawie $B \in E$) polega na znalezieniu dla danego punktu $P \in E$ liczby całkowitej *x*, takiej że $x \cdot B = P$, jeśli taka liczba istnieje.

Dzisiejszy stan wiedzy wskazuje, że problem ten jest obliczeniowo trudniejszy od problemu rozkładu liczb złożonych na czynniki pierwsze i od problemu logarytmowania dyskretnego w ciałach skończonych [3, 7, 10]. Z tej przyczyny asymetryczny system kryptograficzny zbudowany na podstawie krzywej eliptycznej korzysta z istotnie mniejszych liczb naturalnych niż jest to w przypadku systemów opartych na faktoryzacji liczb lub logarytmowaniu w ciele skończonym.

4. ANALIZA BEZPIECZEŃSTWA I PORÓWNANIE SZYFRÓW BEZPIECZNYCH OBLICZENIOWO

Jak już wiadomo, kryptosystemy klucza publicznego nie zapewniają bezwarunkowego bezpieczeństwa, które opiera się jedynie na aktualnej wiedzy teoretycznej i dostępnych mocach obliczeniowych. Ściślej mówiąc – zgodnie z twierdzeniem – problem kryptoanalizy jest zagadnieniem z klasy NP [12]. Jest to na pewno poważne ostrzeżenie przed użyciem algorytmów asymetrycznych. Nigdy nie osiągniemy dla nich takiego bezpieczeństwa (doskonałego) jak na przykład dla metody kryptograficznej *one-time pad*, w której szyfrowanie – przy użyciu funkcji *XOR* – odbywa się każdorazowo z innym, wygenerowanym losowo kluczem. Można by więc sądzić, że aktualnie stosowane algorytmy asymetryczne (tab. 1) są słabe. Argumenty te jednak nie są w pełni rozstrzygające. Aby je przeanalizować, należy przedstawić najnowsze wyniki badań, które pokazują, w jaki sposób kryptologia wykorzystuje dostępną wiedzę z obliczeniowej teorii liczb i teorii algorytmów, by przeciwstawić się postępowi technologicznemu oraz zapobiec jego „przejściu" przez bariery kryptoanalityczne.

Analizę zaczniemy od metod stosowanych do rozwiązywania problemu faktoryzacji liczb całkowitych, ponieważ jest to obszar wiedzy najdłużej badany przez matematyków i kryptologów. Analiza złożoności obliczeniowej rozkładania liczb na czynniki pierwsze ułatwi nam później porównania z problemami logarytmowania dyskretnego.

Jeden z najszybszych znanych algorytmów, zwanych **sitem kwadratowym** QS (*Quadratic Sieve*) charakteryzuje złożoność podwykładnicza. Istniej również modyfikacja tej metody nazywana **wielokrotnym wielomianowym sitem kwadratowym** MPQS (*Multiple Polynomial Quadratic Sieve*) i jest najszybszą wersją tego algorytmu. Ani ona, ani metoda krzywych eliptycznych (ECM) oraz nawet najlepsza **ogólna metoda sita ciała liczbowego** GNFS (*General Number Field Sieve*) nie są efektywne w sensie podanym dla definicji funkcji jednokierunkowej (def. 3.2) [14, 23, 30].

W celu zestawienia i porównania tych algorytmów należy zdefiniować funkcję L_x, wykorzystywaną w analizie złożoności asymptotycznej problemów obliczeniowych stosowanych w kryptologii [3, 13, 17]

$$L_x(v,c) = \exp(c(\ln x)^v (\ln \ln x)^{(1-v)}).$$

Tabela 1. Porównanie algorytmów asymetrycznych stosowanych w systemach publicznych

Podstawa bezpieczeństwa system publicznego	Matematyczny problem	Przykład systemu
Faktoryzacja liczb całkowitych	Dane: liczba $n = p \cdot q$ Szukane: czynniki pierwsze *p* i *q*	RSA, Rabin
Logarytmowanie dyskretne w ciele skończonym	Dane: liczba pierwsza *p* oraz liczby *g* i *y* Szukane: *x* takie, że $y = g^x \bmod p = g \cdot g \cdot g \cdot \ldots \cdot g$ *x* razy	Diffie–Hellman, DSA, ElGamal
Logarytmowanie dyskretne na krzywej eliptycznej	Dane: krzywa eliptyczna E i punkty *B* i $P \in E$ Szukane: *x* takie, że $P = x \cdot B = B + B + \ldots + B$ *x* razy	ECC: EC DSA, EC Diffie–Hellman

Tabela 2. Najefektywniejsze metody faktoryzaji liczb całkowitych

Metoda	Wymaga asymptotycznie liczby operacji rzędu	Czas przebiegu
Sito kwadratowe (wersja MPQS)	$e^{(1+O(1))\sqrt{(\ln n)(\ln\ln n)}}$	Podwykładniczy
Krzywe eliptyczne (ECM)	$e^{(1+O(1))\sqrt{2(\ln p)(\ln\ln p)}}$	Podwykładniczy
Sito ciała liczbowego (GNFS)	$e^{(1.92+O(1))(\ln n)^{1/3}(\ln\ln n)^{2/3}}$	Podwykładniczy

Dla $v = 1$ funkcja L_x zależy wykładniczo od $\ln x$, podczas gdy dla $v = 0$ zależy od $\ln x$ wielomianowo. Zatem widać, iż dla $0 < v < 1$ wzrost funkcji jest szybszy niż wielomianowy, ale wolniejszy niż wykładniczy i dlatego nazywamy go wzrostem podwykładniczym. W przypadku faktoryzacji $x = n$, czyli jest rozmiarem modułu n.

Zgodnie z takim nazewnictwem oraz na podstawie aktualnej wiedzy została stworzona tabela 2, która zawiera zestawienie najefektywniejszych algorytmów używanych do rozkładania liczb na czynniki pierwsze [14, 23, 30].

Symbol $O(1)$ oznacza – w teorii złożoności – funkcję zmiennej n, która dąży do 0, gdy $n \to \infty$, natomiast p jest najmniejszym czynnikiem pierwszym liczby n. W najgorszym przypadku p jest w przybliżeniu równe pierwiastkowi z liczby n, a więc można przyjąć, że asymptotyczne czasy działania algorytmów sita kwadratowego i krzywych eliptycznych są równe, jednak w takich sytuacjach sito kwadratowe daje lepsze rezultaty [17, 23, 30]. Z kolei metoda krzywych eliptycznych jest bardziej przydatna, gdy czynniki pierwsze liczby n różnią się znacząco wielkością [8, 11]. Natomiast jeśli w przypadku najwydajniejszego algorytmu GNFS udałoby się ograniczyć multiplikatywną stałą 1,92 do 1,5, to już dziś realna stałaby się faktoryzacja liczb 1024-bitowych przewidywanych – jeszcze w 2002 r. (tab. 4) – jako bezpieczne [37]. Ale nie udało się jak dotąd tego dokonać.

Wielu naukowców prowadzących badania stara się szacować bezpieczne długości modułu n, jednakże praktyka pokazuje, iż oszacowania te mogą być mało sprawdzalne [1, 24, 25]. Można tu przytoczyć następujący przykład.

Dla modułu $n = 10^{70}$ oraz użytego algorytmu sita kwadratowego, w 1984 roku rozkład na czynniki pierwsze liczby $(10^{71} – 1)/9$ wymagał 9,5 godziny pracy komputera CRAY X-MP. Autor przykładu (zaczerpniętego z pracy [1]), na podstawie tych danych, przeprowadził ekstrapolację i oszacowania, przy założeniu, że co dwa lata podwaja się maksymalna osiągalna prędkość pracy pojedynczego komputera. Zaznacza również, iż uzyskane oszacowania należy traktować z rozwagą. Otrzymany wynik, według którego w 2004 roku dla modułu n rzędu 10^{200} oszacowanie czasu rozkładu na czynniki pierwsze miało wynosić $4{,}8 \cdot 10^4$ lat, jest w chwili obecnej rzeczywiście oszacowaniem niezbyt dokładnym. Przemawia za tym fakt, iż obecnie (tj. ledwie pół roku później) został sfaktoryzowany moduł n tego samego rzędu w czasie niecałych 20 miesięcy [22]. Osiągnięcie to nie zaskakuje aż tak bardzo. W 1994 roku sensację wzbudziła udana próba rozkładu 129-cyfrowej liczby dziesiętnej na dwa czynniki pierwsze po 65 cyfr każdy, której sfaktoryzowanie (zdaniem Rivesta, twórcy RSA) miało zająć 40 kwadrylionów lat, zaś według oszacowań, podanych przez autora przytoczonego przykładu, pojedynczy komputer potrzebowałby na to 3330 dni. W rzeczywistości jednak całą pracę – rozdzieloną między 1600 mniej wydajnych komputerów połączonych przez Internet – zakończono w czasie 8 miesięcy. W latach późniejszych były przełamywane kolejne moduły n, aż do aktualnego osiągnięcia. Więcej szczegółów na ich temat, można uzyskać na stronie firmy RSA [22].

Dla przejrzystości analizy, porównamy teraz ze sobą wszystkie poruszane w publikacji problemy obliczeniowe.

Problem logarytmu dyskretnego w grupie multiplikatywnej dobrze wybranego ciała wydaje się, że w praktyce wymaga takiego samego czasu obliczeń jak rozkład na czynniki liczby całkowitej wielkości porównywalnej z liczbą elementów w ciele. Tak jak w przypadku faktoryzacji, istnieją różne algorytmy rozwiązania tego problemu o czasie podwykładniczym. Wystarczy teraz porównać implementację wykorzystującą grupę punktów $E(F_q)$ odpowiednio wybranej krzywej eliptycznej nad ciałem skończonym z implementacją wykorzystującą grupę multiplikatywną ciała skończonego $F(p)$. Przy tym porównaniu F_q i $F(p)$ mogą nie być tymi samymi ciałami. Kluczową obserwacją jest fakt, że dla dobrze dobranej krzywej najlepsze znane metody rozwiązywania problemu logarytmu dyskretnego w $E(F_q)$ mają złożoność zależną wykładniczo od rozmiaru elementu ciała, czyli od $n = \lceil \log_2 q \rceil$, podczas gdy dla problemu logarytmu dyskretnego w grupie $F(p)$ dostępne są algorytmy podwykładnicze ze względu na $n = \lceil \log_2 p \rceil$ [3, 7, 13]. Parametr n można interpretować jako wyrażony w bitach rozmiar klucza dla odpowiednich kryptosystemów.

Ściślej ujmując, logarytmy dyskretne w ciele skończonym $F(p)$ można znajdować w czasie proporcjonalnym do $L_x(1/3, c)$, gdzie $x = p$, a $c \approx 1{,}92$, korzystając z metody GNFS [3, 13]. Przyjmuje się więc, że jest to metoda o mniej więcej tej samej złożoności asymptotycznej jak w przypadku algorytmu faktoryzacji. Dlatego dalsze rozważania i porównania (tab. 3) stosuje się również do „konwencjonalnych" kryptosystemów, a więc na przykład do systemu RSA czy DSA i Diffiego–Hellmana.

Tabela 3. Zestawienie najefektywniejszych algorytmów faktoryzacji oraz logarytmowania dyskretnego

Problem obliczeniowy	Najlepsza metoda rozwiązująca problem	Czas przebiegu
Faktoryzacja liczb całkowitych	Ogólna metoda sita ciała liczbowego (GNFS) $e^{(1.92+O(1))(\ln n)^{1/3}(\ln\ln n)^{2/3}}$	Podwykładniczy
Logarytmowanie dyskretne w ciele skończonym	Ogólna metoda sita ciała liczbowego (GNFS) $e^{(1.92+O(1))(\ln n)^{1/3}(\ln\ln n)^{2/3}}$	Podwykładniczy
Logarytmowanie dyskretne na krzywej eliptycznej	Pollard-rho $\sqrt{n}$	Wykładniczy

Dla krzywych eliptycznych nie jest znany żaden algorytm o czasie podwykładniczym (z wyjątkiem krzywych supersingularnych) [3, 7, 10, 11]. Jedynymi dostępnymi metodami znajdowania logarytmów na krzywych eliptycznych są metody, które stosuje się do dowolnych grup. Złożoność najlepszych znanych algorytmów ogólnych dla problemu logarytmu dyskretnego na krzywej eliptycznej jest proporcjonalna do pierwiastka kwadratowego z n. Dlatego, na obecnym etapie rozwoju wiedzy o algorytmach, przy porównywalnej sile kryptograficznej rozmiar klucza n w kryptosystemie opartym na krzywych eliptycznych rośnie nieco szybciej niż pierwiastek sześcienny z rozmiaru klucza systemu „konwencjonalnego" [3, 15]. Pomaga to wyjaśnić współczesne trendy i zainteresowanie kryptografią opartą na krzywych eliptycznych, jako tańszą alternatywą dla systemów konwencjonalnych.

Na koniec tego rozdziału należy zadać pytanie, jaka długość klucza jest uważana za odpowiednią dla poszczególnych algorytmów, ponieważ jest ono powtarzane zawsze wtedy, gdy przekroczona zostaje kolejna granica w rozkładzie liczb na czynniki pierwsze oraz padają kolejne rekordy dotyczące problemu obliczania logarytmów dyskretnych [5, 16, 22]. Wychodząc od postawienia dobrze ugruntowanych założeń w powiązaniu z aktualnym stanem wiedzy, Lenstra i Verheul przedstawili zalecenia odnośnie do prognozowanej minimalnej wielkości kluczy kryptograficznych w przyszłości [15, 34]. Określili, jakiej długości klucze asymetryczne (a co za tym idzie – jakiej wielkości liczby pierwsze) należy stosować, aby algorytmy były bezpieczne.

Tabela 4 zawiera takie zestawienie.

5. WNIOSKI

Na podstawie przeprowadzonej analizy można stwierdzić, iż wykorzystywane algorytmy w kryptografii z kluczem jawnym (tab. 1) są tego rodzaju, że można złamać klucze prywatne, odwracając zastosowaną funkcję jednokierunkową. Jest to zadanie o wiele szybsze niż proces pełnego przeglądu przestrzeni klucza, ale na tyle pracochłonne, że przy wykorzystaniu obecnie dostępnych mocy obliczeniowych jest niewykonalne w czasie opisywanym przez niskiego stopnia wielomian zmiennej n (tab. 3). Implikuje to niebezpieczne stwierdzenie, iż pozory bezpieczeństwa oferowane użytkownikowi takich kryptosystemów opierają się jedynie na złożoności kombinatorycznej danego problemu. Niestety należy wziąć tu pod uwagę jeszcze fakt, iż mimo istnienia wielu takich funkcji, o których sądzi się, że są jednokierunkowe, nie udało się do dziś udowodnić o żadnej z nich, iż ma tę własność. Dodatkowym zagrożeniem szyfrowania z kluczem publicznym jest to, że nikt nie wie na pewno, czy rozkładanie na czynniki pierwsze lub obliczanie logarytmów dyskretnych musi być dla większości kluczy powolnym procesem. Należy tu bardzo uważać, ponieważ jak wykazano, pewne klucze (czyli liczby pierwsze) oraz pewne krzywe eliptyczne nie nadają się do konstrukcji bezpiecznego kryptosystemu klucza publicznego ze względu na małą odporność kryptoanalityczną [3, 6, 7, 10, 11, 24].

Powiązane ze sobą problemy logarytmowania dyskretnego i faktoryzacji liczby całkowitych niosą ze sobą jeszcze jedno zagrożenie.

Tabela 4. Bezpieczne długości asymetrycznych kluczy kryptograficznych

Rok	Długość w bitach / liczba cyfr w systemie dziesiętnym	
	Systemy oparte na problemie faktoryzacji i logarytmowania dyskretnego w ciałach skończonych	Systemy oparte na problemie logarytmowania dyskretnego na krzywych eliptycznych
2000	952 / 287	132 / 40
2002	1028 / 310	139 / 42
2005	1149 / 346	147 / 45
2010	1369 / 413	160 / 49
2015	1613 / 486	173 / 52
2020	1881 / 567	188 / 57

Jeśli ktoś będzie potrafił obliczyć logarytm dyskretny, to posiadał też będzie algorytm faktoryzowania dużych liczb. Równocześnie, jeśli znamy rozkład modułu *n* na czynniki pierwsze, to wyznaczenie logarytmu dyskretnego w ciele skończonym nie jest trudne [3, 7, 33, 37]. Wystarczy teraz wziąć pod uwagę fakt, iż w praktyce klucze publiczne są dość rzadko zmieniane, a każdej parze kluczy odpowiada niewiarygodna liczba zaszyfrowanych wiadomości. Gdyby ktokolwiek przechwytywał i zachowywał zaszyfrowane korespondencje, a za kilka lat okaże się, że problemy te doczekają się rozwiązań o przebiegu wielomianowym, oznaczałoby to kompletną klęskę. Można by wtedy odczytać od ręki wszystkie wiadomości: bieżące i również te zapomniane, ponieważ łatwo będzie złamać także wszystkie klucze sesyjne szyfrowanych połączeń.

Aby zapewnić wystarczający margines bezpieczeństwa w krytycznych zastosowaniach, długość klucza nie powinna być mniejsza niż wynika z oszacowań Lenstry i Verheula (tab. 4), ponieważ ze względu na postępujące sukcesy obliczeniowe, jesteśmy coraz bliżej wartości granicznych [5, 16, 22]. Jednakże nie możemy przesadzać z wyborem odpowiednio dużych kluczy, ponieważ należy pamiętać o tym, że koszt obliczeń wzrasta ze wzrostem długości klucza. Tak więc długość klucza powinna być wystarczająco bezpieczna, ale również wystarczająco mała, by była użyteczna pod względem obliczeniowym. W praktyce bowiem krótsze klucze mogą przełożyć się na szybsze implementacje, mniejsze zużycie energii, mniejszą powierzchnię krzemu itd., a takie argumenty zdecydowanie przemawiają na korzyść kryptosystemów ECC. Tu jednak też należy mieć pewne obawy, że zasadniczy problem, będący podstawą systemów opartych na krzywych eliptycznych, nie został tak szczegółowo przebadany jak np. problem faktoryzacji liczb całkowitych.

6. PODSUMOWANIE

Jednak mimo wszystko te ogromne liczby mogą stać się złudne. Teoria złożoności w swojej obecnej postaci niewiele może tu pomóc, gdyż zazwyczaj daje ona jedynie górne oszacowanie pracochłonności danej metody. Nie ma również dowodu na brak algorytmów dużo szybszych niż przytoczone w niniejszej pracy. Również trudno udowodnić nieistnienie innych zapadek niż te, które już znamy. Nowa zapadka może zagrozić bezpieczeństwu kryptosystemu tak samo jak bezpośredni atak deszyfrujący, omijający całkowicie trudności związane z wyznaczeniem funkcji odwrotnej. Niebezpieczeństwo to stanowi podstawowe ryzyko stosowania metod asymetrycznych. Jest ono na tyle poważne, że stawia pod znakiem zapytania ich zastosowanie w szczególnie istotnych dziedzinach.

Może ktoś dokona „wielkiego przełomu", choć możliwe, że taki przełom jest nierealny, za czym stoi bardzo silna hipoteza informatyki teoretycznej, że $P \neq NP$. Przypuszczalnie komputery kwantowe lub równie efektywne – wspomniane we wstępie publikacji – rozwiązania sprzętowe potrafiłyby złamać znacznie dłuższe klucze niż obecnie stosowane, ale takich technologii należy jednak oczekiwać dopiero za kilkadziesiąt lat (jeżeli kiedykolwiek będziemy z nich korzystać). Na razie można chyba uznać, że coraz większe rozmiary klucza będą równoważyły wszelki ulepszenia w zakresie rozkładania na czynniki pierwsze oraz logarytmowania dyskretnego.

Literatura

[1] Bauer F.: *Sekrety kryptografii*. Gliwice, Helion 2003, ISBN 83-7197-960-6
[2] Bernstein D.: *Circuits for integer factorization: a proposal*. Excerpted from DMS-0140542 grant proposal, 2001. http://cr.yp.to/papers/nfscircuit.pdf
[3] Blake I., Seroussi G., Smart N.: *Krzywe eliptyczne w kryptografii*. Warszawa, WNT 2004, ISBN 83-204-2951-X
[4] Borzyszkowski A., Srebrny M.: *Faktoryzacja za pomocą masowej równoległości?* Artykuł z archiwum Centrum Certyfikacji Signet, 2002. http://www.signet.pl/archiwum/93.html
[5] Certicom: *ECC Challenge*. http://www.certicom.com/index.php?action=ecc,ecc_challenge
[6] Denning D.: *Wojna informacyjna i bezpieczeństwo informacji*. Warszawa, WNT 2002, ISBN 83-204-2687-1
[7] Gawinecki J., Szmidt J.: *Zastosowanie ciał skończonych i krzywych eliptycznych w kryptografii*. Warszawa, Wyd. WAT 2002, ISBN 83-88442-40-6
[8] Goldwasser S., Bellare M.: *Lecture Notes on Cryptography*. MIT 2001. http://www.cs.ucsd.edu/users/mihir/papers/gb.pdf
[9] Knuth D.: *Sztuka programowania. Tom 2. Algorytmy seminumeryczne*. Warszawa, WNT 2002, ISBN 83-204-2553-0
[10] Koblitz N.: *Algebraiczne aspekty kryptografii*. Warszawa, WNT 2000, ISBN 83-204-2418-6
[11] Koblitz N.: *Wykład z teorii liczb i kryptografii*. Warszawa, WNT 1995, ISBN 83-204-1836-4
[12] Kutyłowski M., Strothmann W.: *Kryptografia. Teoria i praktyka zabezpieczania systemów komputerowych*. Warszawa, Read Me 1999, ISBN 83-7147-092-4
[13] Lenstra A.: *Computational Methods in Public Key Cryptology (lecture notes)*. http://www.win.tue.nl/~klenstra/notes.pdf
[14] Lenstra A.: *Factoring estimates for a 1024-bit RSA modulus*. Asiacrypt 2003. http://www.win.tue.nl/~klenstra/fac_1024RSA.pdf
[15] Lenstra A., Verheul E.: *Selecting Cryptographic Key Sizes*. Journal of Cryptology 14, 2001, 255-293. http://www.win.tue.nl/~klenstra/key.pdf
[16] Lercier R.: *Discrete logarithms in finite fields*. http://www.medicis.polytechnique.fr/~lercier/english/dlog.html
[17] Menezes A., Oorschot P., Vanstone S.: *Handbook of Applied Cryptography*. CRC Press 1996. http://www.cacr.math.uwaterloo.ca/hac/
[18] Ogiela M. R.: *Bezpieczeństwo systemów komputerowych*. Kraków, UWND AGH 2002, ISBN 83-86408-57-7
[19] Ogiela M. R.: *Podstawy kryptografii*. Kraków, UWND AGH 2000, ISBN 83-88408-31-3
[20] Ogiela M. R.: *Systemy utajniania informacji*. Kraków, UWND AGH 2003, ISBN 83-89388-26-X
[21] Papadimitriou Ch.: *Złożoność obliczeniowa*. Warszawa, WNT 2002, ISBN 83-204-2659-6
[22] RSA Lab.: *Factoring Challenge*. http://www.rsasecurity.com/rsalabs/node.asp?id=2092
[23] RSA Lab.: *Frequently Asked Questions About Today's Cryptography*. http://www.rsasecurity.com/rsalabs/faq/files/rsalabs_faq41.pdf
[24] Schneier B.: *Kryptografia dla praktyków. Protokoły, algorytmy i programy źródłowe w języku C*. Warszawa, WNT 2002, ISBN 83-204-2678-2

[25] Schneier B.: *Ochrona poczty elektronicznej*. Warszawa, WNT 1996, ISBN 83-204-1979-4
[26] Shamir A.: *Factoring large numbers with the TWINKLE device (extended abstract)*. Springer-Verlag, 1999 http://cryptome.org/twinkle.zip
[27] Shamir A., Tromer E.: *Factoring large numbers with the TWIRL device*. Springer-Verlag, 2003 http://www.wisdom.weizmann.ac.il/~tromer/papers/twirl.pdf
[28] Sierpiński W.: *Teoria liczb*. Warszawa, PWN 1950
[29] Stallings W.: *Ochrona danych w sieci i intersieci. W teorii i praktyce*. Warszawa, WNT 1997, ISBN 83-204-2184-5
[30] Stinson D.: *Kryptografia. W teorii i w praktyce.* Warszawa, WNT 2005, ISBN 83-204-2982-X
[31] Stokłosa J.: *Algorytmy kryptograficzne*. Poznań, OWN 1994, ISBN 83-85481-65-6
[32] Stokłosa J., Bilski T., Pankowski T.: *Bezpieczeństwo danych w systemach informatycznych*. Warszawa, PWN 2001, ISBN 83-01-13452-6
[33] Stokłosa J.: *Kryptograficzna ochrona danych w systemach komputerowych*. Poznań, Nakom 1994, ISBN 83-85060-66-9
[34] Stokłosa J.: O*chrona danych i zabezpieczenia w systemach teleinformatycznych*. Poznań, Wyd. Politechniki Poznańskiej 2003, ISBN 83-7143-478-2
[35] TOP500 Supercomputer List. http://top500.org
[36] Welschenbach M.: *Kryptografia w C i C++*. Warszawa, Mikom 2002, ISBN 83-7279-247-X
[37] Wobst R.: *Kryptologia. Budowa i łamanie zabezpieczeń*. Warszawa, RM 2002, ISBN 83-7243-068-3

Wpłynęło: 13.06.2005

Piotr KAPCIA

Urodzony 21 czerwca 1974 roku w Brzesku. Studia na Wydziale Inżynierii Elektrycznej i Komputerowej Politechniki Krakowskiej ukończył w 1999 roku. Został nagrodzony dyplomem Stowarzyszenia Elektryków Polskich za najlepszą pracę magisterską z zakresu automatyki. Obecnie jest doktorantem Wydziału Elektrotechniki, Automatyki, Informatyki i Elektroniki Akademii Górniczo-Hutniczej w Krakowie i zajmuje się obliczeniową teorią liczb oraz algebraicznymi aspektami kryptografii asymetrycznej.

e-mail: pika@agh.edu.pl