EÖTVÖS LÓRÁND TUDOMÁNYEGYETEM

TERMÉSZETTUDOMÁNYI KAR

Dan Brown Digitális erődje és a nyilvános kulcsú titkosítás

BSc Szakdolgozat

Készítette:
Fekete Ildikó
Elemző matematika szakos hallgató

Témavezető:
Korándi József
adjunktus

Budapest
2014

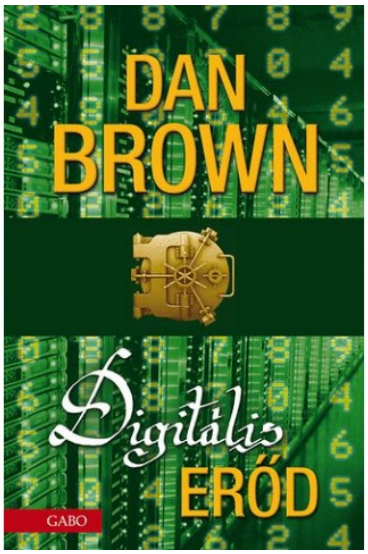
Tartalomjegyzék

Bevezetés

A Digitális erőd egy világszerte ismert sikerkönyv, melyben az izgalmas események hátterében a matematika áll. Ez ugyanis a titkok őrzője. Arról van szó, hogy az Amerikában található NSA szervezetet kódfeltörés céljából hozták létre. Itt a legjobb informatikusok és matematikusok dolgoznak és a legmodernebb technika is a rendelkezésükre áll. Így bármilyen kódolt üzenetet néhány perc alatt képesek megfejteni, azokat is, melyeket ma biztonságosnak ítélünk, mint például az RSA kódot. Ám egyszer csak valaki a feltörhetetlen algoritmus létrehozásával próbálkozik.

Szakdolgozatomban azt szeretném bemutatni, hogyan működik mindez a valóságban. Megmutatom, hogyan kell létrehozni az RSA kódot, és hogy miért tekintjük ezt biztonságos eljárásnak.

Az első fejezetben bemutatom a régebben használt főbb titkosítási eljárásokat. Ezt követően számba veszem, hogy a Digitális erődben miként jelenik meg a matematika.



1. ábra. Dan Brown: Digitális erőd című könyve

Dan Brown könyve a modern titkosításról szól. Mivel napjaink leggyakrabban használt módszere az RSA algoritmus, a továbbiakban ezzel foglalkozom részletesen. Néhány tétel és definíció közlése után bemutatom egy általános RSA rendszer létrehozását, majd erre adok is konkrét példát, az egyiket a Maple program alkalmazásával.

Mivel az RSA-ban fontos szerepe van a nagy prímszámoknak, a következő fejezetben arra is kitérek, hogyan lehet ilyeneket találni. Ezt követően bemutatom az RSA általános feltörésére vonatkozó sikertelen próbálkozásokat, majd azokat a speciális eseteket, amikor a nem megfelelő alkalmazás miatt válik feltörhetővé a kód.

Ezt követően szó esik a különféle problémaosztályokról is, hiszen ezáltal válik érthetővé, hogy napjainkban miért tekintjük biztonságosnak ezt a titkosítást és hogy a jövőben esetleg miért nem lesz már feltörhetetlen. Végül leírom, hogy ez a mószer hogy jelenik meg a gyakorlatban.

1. fejezet

A titkosítás rövid története

Titkokra mindig is szükség volt. Ám két ember általában nem volt olyan fizikai közelségben, hogy a titkaikat élő szóban meg tudták volna beszélni, így kellett egy harmadik személy, aki továbbította az üzeneteket. Annak érdekében, hogy se a hírvivő, se pedig más ne tudhassa, mi áll a levélben, titkosítani kellett, méghozzá oly módon, hogy csak a feladó és a címzett érthesse az üzenetet, mindenki más számára megfejthetetlen legyen.

Az üzenet biztonságos célba juttatásának egy másik módja, ha magát a szöveget tüntetjük el. Tehát ha nem lehet tudni, hogy van egyáltalán üzenet. Ezt szteganográfiának nevezik. Ez megoldható például szobahőmérsékleten láthatatlan, de hőre sötétedő tintával vagy egy képbe ügyesen beleírt szöveggel, azonban az ilyen eljárásokra nem térek ki a dolgozatban.

Tudomásunk szerint az első rejtjelezett üzenetek több mint kétezer évesek. Ezeknek két csoportját különböztethetjük meg, a felcserélő és a behelyettesítő titkosításokat.

1.1. A felcserélő titkosítás

Lényege, hogy az eredeti szöveg betűit valamilyen algoritmus alapján felcserélik egymással. Ezt anagrammaírásnak is nevezzük. Egy Digitális erődbeli példát használva legyen az eredeti üzenet: $AKRIPTOGRAFUSNONEVE : SUSAN$. Első lépésként vegyük ki a szövegből a szóközöket és a központozást, majd írjuk le az üzenetet két sorba oly módon, hogy minden páratlanodik betű az első sorba és minden párosodik betű a második sorba kerüljön. Majd a második sort írjuk rögtön az első után.

Ezzel kész a titkosított szöveg: $ARPORFSOEEUAKITGAUNNVSSN$.

Olvasásához a kódoló algoritmus fordítottját kell alkalmazni. Ez így még elég könnyen megfejthető, de lehet nehezíteni azzal, hogy nem két, hanem több sorba osztjuk szét az eredeti szöveg betűit. Azonban már ezt a rövid szöveget is nagyon sokféleképpen lehet megadni ($\frac{24!}{3!\cdot 2!\cdot 2!\cdot 2!\cdot 3!\cdot 3!\cdot 2!} \approx 3{,}59\cdot 10^{20}$ féle). Így a kód csak algoritmus segítségével fejthető meg, túl sokáig tartana felírni az összes variációs lehetőséget.

1.2. A behelyettesítő titkosítás

Lényege, hogy minden betűt egy másik karakterrel jelölünk a kódolt szövegben. Az ilyen fajta eljárásokat monoalfabetikus behelyettesítő rejtjeleknek is nevezik. Ennek a legrégebbi ismert formája a Caesar rejtjel. Az algoritmus egyszerű, az abc betűit tolja el valamennyivel. Tehát ha például a 26 betűs abc-vel dolgozunk, akkor az eredeti szöveget 25 féleképpen lehet kódolni. (A 26-tal való eltolás már önmagába viszi a szöveget.)

Azonban ez sem elég biztonságos, hiszen könnyűszerrel feltörhető. Ennek egy nehezebb változata az, amikor az eredeti abc betűit véletlenszerűen helyettesítjük egymással. (Ezzel a módszerrel $26! \approx 4 \cdot 10^{26}$ féleképpen írható le a szöveg. Ha a szóközöket is belevesszük, akkor a lehetőségek száma tovább nő.)

Például:
$ABCDEFGHIJKLMNOPQRSTUVWYZ =$
$= HPFGJRZAENQWYBLSCOUKDTIVMX.$

Ekkor, ha az eredeti üzenet: $AKRIPTOGRAFUSNONEVE : SUSAN$, akkor a kódolt változat: $HQOESLZOHRDUBLBJTJUDUHB$. Ennek a véletlenszerű behelyettesítésnek azonban az az egyik hátránya, hogy a kulcs nehezen megjegyezhető. Azonban a levelező felek megegyezhetnek egy közös kulcsban, ami könnyen észben tartható. Például legyen a kulcs az, hogy $ESIKAHO$.

Ekkor az algoritmus a következő:
$ABCDEFGHIJKLMNOPQRSTUVWYZ =$
$= ESIKAHOBCDFGIJKLMNPQRSTUVXYZ.$

Tehát a kulcsot előre írjuk, aztán mögé az abc betűit sorba, kihagyva azokat, amik szerepelnek a kulcsban. A módszer annál biztonságosabb, minél hosszabb a kulcs, hiszen annál inkább véletlenszerű.

Az eljárás látszólag tovább bonyolítható azzal, ha számokkal, jelekkel vagy akár ábrákkal egészítjük ki a titkosító készletünket. Azonban, ha az eredeti szöveg egy adott betűjét a kódban mindig egy adott karakterrel helyettesítjük, akkor minél hosszabb az üzenet, annál kevésbé biztonságos ez az eljárás. Az így titkosított szöveg ugyanis gyakoriságvizsgálatnak vethető alá. Ez azon alapszik, hogy minden nyelvnek megvannak a maga sajátságai, tehát megadható, melyek azok a betűk vagy betűkapcsolatok, amik a leggyakrabban előfordulnak az adott nyelvben. (A magyarban például az E, A, T és az O a leggyakrabban használtak között van.) Így tehát egy hosszú kódolt szöveg esetén elég megvizsgálni a karakterek százalékos előfordulását. Ez jó támpontot nyújt arra, hogy mivel kell helyettesíteni a kódban lévő karaktereket. Hátránya azonban az, hogy előre tudnunk kell, hogy a feltörni kívánt szöveg milyen nyelven íródott, emelett az ismert gyakoriságok csak a köznyelvre igazak, egy szakszöveg ettől lényegesen eltérhet.

Az ilyenfajta feltörést nehezíteni lehet azzal, ha egy adott betűt a kódban több karakterrel is helyettesítünk, így egy adott kódbeli karakter gyakorisága is csökken. (Ezt polialfabetikus kódrendszernek nevezzük.)

1.3. Polialfabetikus kódrendszerek

Az egyik gyakran használt polialfabetikus kód a Vigenére tábla. Ennek alapja a Caesar rejtjel. Kiindulásként tekintsük a 26 betűs abc-t. Írjuk egymás alá az abc eltoltjait növekvő sorrendben. Ekkor a következő 26×26-os táblát kapjuk:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
2	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
3	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
4	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
5	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
6	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
7	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
8	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
9	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
10	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
11	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
12	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
13	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
14	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
15	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
16	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
17	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
18	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
19	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
20	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
21	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
22	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
23	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
24	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
25	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
26	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

1.1. ábra. Vigenére tábla

A polialfabetikus rendszer úgy épül fel, hogy a kódhoz más-más sorokat használunk. Hogy egyértelmű legyen a titkosítás illetve a visszafejtés, a feleknek itt is meg kell állapodni egy bizonyos kulcsban, mely betű- vagy számkombináció is lehet. A kulcs azt adja meg, hogy a kódoláskor éppen melyik sort használják. Tehát ha a kulcs: $ESIKAHO$ és a rejtjelezni kívánt szöveg: $AKRIPTOGRAFUSNONEVE : SUSAN$, akkor a kódolás a következő módon történik:

A	K	R	I	P	T	O	G	R	A	F	U	S	N	O	N	E	V	E	S	U	S	A	N
E	S	I	K	A	H	O	E	S	I	K	A	H	O	E	S	I	K	A	H	O	E	S	I	
E	C	Z	S	P	A	C	K	J	I	P	U	Z	B	S	F	M	X	E	Z	I	W	S	V	

1.2. ábra

Vagyis az A kódolásához megkeressük az E-vel kezdődő sort és megnézzük, hogy az eredeti abc-hez képest mi áll az A helyén. Ez az E lesz. Ennél az első karakternél nem olyan látványos a titkosítás, mint a rá következőnél, a K-nál. Ott megkeressük az S-el kezdődő sort és megnézzük, hogy az eredeti abc-ben lévő K helyén itt mi áll. Ez a C lesz, tehát a második helyen álló K-t C fogja jelölni a titkosított szövegben, és így tovább. Jól látható, hogy a rendszer tényleg polialfabetikus, például az A-t háromféle karakter jelöli a szövegben. Azonban ez a kódolási technika sem bizonyult megfejthetetlennek. Friedrich Kasiski 1863-ban publikálta azt az algoritmust, amellyel ez megfejthető, a tényleges dekódoláshoz azonban számítógépek szükségesek.

2. fejezet

Matematika a Digitális erőd című könyvben

A könyv megértéséhez nincs szükség mélyebb matematikai vagy informatikai ismeretekre. A fontosabb fogalmak közérthetően le vannak írva, látszik, hogy Dan Brown utánanézett ezeknek vagy -állítása szerint- igénybe vette néhány volt NSA dolgozó segítségét. Azonban akad néhány olyan tétel vagy fogalom melyek valójában nem léteznek. Ilyen például a "Bergofsky elv". Emellett az író a szakkifejezéseket néhol csak arra használja, hogy képet fessen az olvasónak a matematikusokról. Például a 18. oldalon: "... Valami érthetetlen nyelven hadováltak. Folyamatos titkosítókat, rekurzív generátorokat, különféle pakolási problémákat, nemfeltáró protokollokat és unicitási pontokat emlegettek." Ekkor csak hangulatfestésnek használja a szavakat, az ilyen jellegű matematikával a továbbiakban nem kívánok foglalkozni.

Az első algoritmus, amiről részletesen olvashatunk az a 27. oldalon "Julius Caesar teljes négyzet táblája". (Én nem találtam forrást arról, hogy ezt a konkrét algoritmust is Caesar találta volna ki.) Ez lényegében egy felcserélő titkosírás. Itt az üzenetben lévő betűk száma mindig egy négyzetszám. Ha például hatvannégy, akkor ez azt jelenti, hogy a szöveget nyolc sorba bontjuk szét, majd írjuk az így kapott betűcsoportokat egymás után. Megfejtéséhez elég egy 8×8-as táblába beleírni a kódolt szöveget (balról jobbra) majd fentről lefelé olvasni.

Az író nemcsak a szereplőivel használtatja ezt, ő maga is elrejtett egy kódot a könyv utolsó oldalán, mely a következő képpen néz ki: "$118 - 112 - 62 - 36 - 59 - 61 - 2 - - 18 - 48 - 124 - 118 - 118 - 16 - 95 - 71 - 32$". Minden egyes szám a könyv azonos számú fejezetének első betűjét jelöli. Ezeket behelyettesítve az alábbi szöveget kapjuk: $IRAK - SJTA - MUII - EKTD$. Majd Caesar táblába írva és megfelelően olvasva megkapjuk az eredeti szöveget: $ISMERJUKATITKAID$.

Ezek mind a régebben használt titkosírások, melyek azon alapulnak, hogy a titkosító algoritmus ismerete nélkül nagyon nehézzé válik a szöveg megfejtése, de ha mégis ismerjük az algoritmust, annak a fordítottja alkalmazásával az üzenet könnyen visszafejthető.

A mai, nyilvános kulcsú algoritmusok abban térnek el, hogy az algoritmus ismerete nem elég a kód feltöréséhez, ezért azt nem kell titokban tartani, csupán a titkosításhoz használt privát kulcsot.

Mivel a könyv fő témája a modern titkosítás, szó esik a prímszámokról és azok tulajdonságairól is, hiszen ezek az alapjai a mai eljárásoknak. Ilyen állítás például, hogy a prímek csak eggyel és önmagukkal oszthatók vagy az, hogy végtelen sok van belőlük. Habár ezek az állítások igazak, nem elegendőek a ma használt algoritmusok megismeréséhez. (A regénynek nem is ez a célja.)

A könyv olvasása során többször találkozhatunk a "nyilvános kulcs" vagy a "nyilvános titkosító" kifejezésekkel. Ezek egyértelműen a mai, nyilvános kulcsú titkosírásokra utalnak, az azonban nem derül ki, hogy pontosan melyikről van szó. Ilyen titkosító például a PGP, melyet Philip R. Zimmermann fejlesztett ki. De mivel napjainkban az RSA a legelterjedtebb ilyen algoritmus, a későbbiekben ezzel foglalkozom részletesebben.

A könyvben szó esik még egy magyar matematikusról, Harne Józsefről is. Azt olvashatjuk, hogy 1987-ben publikált egy cikket, melyben bemutatta saját, feltörhetetlen algoritmusát. Ennek az a lényege, hogy ha a számítógép ki is találja a megfelelő kulcsot, ezt akkor sem fogja megtudni, mert a titkosított szöveg karakterei egy időváltozó szerint folyamatosan módosulnak, így nem fog a számítógép azonosítható (értelmes) karakterláncokat találni. Internetes kutatásaim során sajnos ahol felbukkant Harne József neve vagy ez az elgondolás mindenhol azt találtam, hogy sem a matematikus sem a publikáció nem létezik.

3. fejezet

Az RSA séma bemutatása

A nyilvános kulcsú titkosítás alapgondolata Diffietől és Hellmantól származik, melyet 1975-ben vetettek fel. A konkrét algoritmus azonban Ron Rivestről, Adi Shamirtól és Leonard Aldemantól származik, melyet a Scientific American című folyóirat 1977-es augusztusi számában publikáltak. Az RSA elnevezés feltalálóinak vezetéknevének kezdőbetűiből ered.

Az RSA biztonsága abban rejlik, hogy két nagy számot számítógéppel gyorsan össze tudunk szorozni, de egy több száz jegyű szám prímtényezős felbontását megtalálni a ma ismert algoritmusokkal nagyon sok időbe telik. (Erről a későbbiekben írok részletesebben.) Ezért ha összeszorzunk például két darab kétszáz jegyű számot és a szorzatot nyilvánosságra hozzuk, rajtunk kívül senki nem fogja tudni, hogy mely számokat szoroztuk össze.

Az algoritmus leírásához az alábbi tételeket illetve definíciókat használjuk: (Ezek tanulmányaim folyamán szerepeltek, ezért a tételek bizonyításától eltekintek.)

3.1. Definíció. ***Euler-féle φ függvény:*** *$\varphi(n) = 1,2,\ldots,n$ számok közül az n-hez relatív prímek száma. (Ez egyenlő a $\pmod n$ redukált maradékosztályok számával. Ha p prím, akkor nyilvánvalóan $\varphi(p) = p - 1$.)*

3.2. Tétel. ***Kis Fermat tétel:*** *ha p prím és nem osztója a-nak, akkor $a^{p-1} \equiv 1 \pmod p$ (vagy $a^p \equiv a \pmod p$). (Ez az Euler-Fermat tétel speciális esete.)*

3.3. Tétel. ***Euler-Fermat tétel:*** *ha $(a, m) = 1$, akkor $a^{\varphi(m)} \equiv 1 \pmod m$.*

3.4. Tétel. *Tegyük fel, hogy p és q különböző prímszámok és $m = pq$. Ekkor minden olyan a egész számra, mely relatív prím m-hez, teljesül, hogy $a^{(p-1)(q-1)} \equiv 1 \pmod m$. (Euler-Fermat tételből egyszerűen következik.)*

Az RSA használatához első lépésként válasszunk két nagy prímet, p-t és q-t, és szorozzuk össze őket. Ekkor $pq = n$. Ezután meg kell határoznunk $\varphi(n)$-t, mely nyilvánvalóan $(p - 1)(q - 1)$. Ekkor keresnünk kell egy olyan e számot, mely relatív prím $(p - 1)$-hez és $(q - 1)$-hez is. Majd meg kell határoznunk az e szám $(p - 1)(q - 1)$ modulusra

vonatkozó inverzét, mely az f szám lesz. (Tehát azt az f számot keressük, amire teljesül, hogy $ef \equiv 1 \pmod{(p-1)(q-1)}$. Ezt megtehetjük az euklideszi algoritmus segítségével, melyet később részletesen bemutatok.) Ekkor tetszőleges a számra igaz lesz, hogy $a^{ef} \equiv a \pmod{pq}$. Végezetül, közzétesszük az n és az e számokat, f-et pedig titokban tartjuk. (A kiindulási prímeket akár el is felejthetjük, a továbbiakban nincs rájuk szükség.)

Tegyük fel, hogy valaki (például Tamás) egy RSA-val titkosított üzenetet szeretne küldeni nekünk. Ekkor a szöveget valamilyen módon először számmá kell tennie. Erre alkalmas lehet például az $A = 01, B = 02, \ldots, Z = 26$ átírás. Ekkor Tamásnak az üzenetet legfeljebb $n-1$ nagyságú részekre kell darabolnia, az így kapott blokkok legyenek x_1, x_2, …, x_k. Utána egyesével ki kell számolnia az $x_1^e, x_2^e, \ldots, x_k^e \pmod{n}$ maradékokat, melyeket nevezzünk $c_1, c_2, \ldots, c_k$-nak. Ezek azok az üzenetek, melyeket el fog küldeni nekünk. (Teljesül rájuk, hogy $x_1^e \equiv c_1, x_2^e \equiv c_2, \ldots, x_k^e \equiv c_k \pmod{n}$).

Ha megkaptuk az üzenetet, akkor az f dekódoló kulcsunk segítségével kiszámoljuk $c_1^f, c_2^f, \ldots, c_k^f \pmod{n}$ maradékokat. Ezáltal visszakapjuk az eredeti üzenetet, mert $ef \equiv$ $\equiv 1 \pmod{(p-1)(q-1)}$ miatt $c_1^f \equiv x_1^{ef} \equiv x_1, c_2^f \equiv x_2^{ef} \equiv x_2, \ldots, c_k^f \equiv x_k^{ef} \equiv x_k \pmod{n}$. Mivel x_i-k kisebbek, mint n, a visszafejtés egyértelmű.

(Fontos, hogy x_i-k relatív prímek legyenek n-hez, különben az $x_i^e \equiv 1 \pmod{n}$ kongruencia nem teljesül. Tehát x_i-k nem lehetnek oszthatók sem p-vel, sem q-val. A valóságban azonban olyan kicsi annak a valószínűsége, hogy valamely x_i ilyen lenne, hogy ezt a feltételt el is hanyagolhatjuk.)

3.1. Gyakorlati példa az RSA-kódolásra

Legyen $p = 59$, $q = 97$. Ekkor $n = 5723$ és $\varphi(n) = 5568$. Válasszuk e-t például 73-nak. Ezekkel a kiindulási értékekkel f meghatározása euklideszi algoritmussal a következő módon történik:

$$5568 = 73 \cdot 76 + 20$$

$$73 = 20 \cdot 3 + 13$$

$$20 = 13 \cdot 1 + 7$$

$$13 = 7 \cdot 1 + 6$$

$$7 = 6 \cdot 1 + 1$$

$$6 = 1 \cdot 6 + 0.$$

(A hányadost és a maradékot kiszámíthatjuk $iquo(5568{,}73), irem(5568{,}73)$; Maple parancsokkal is.) Ezzel egyben ellenőriztük is, hogy $(e, \varphi(n)) = 1$, de ezekre a részszámításokra szükség van a továbbiakban, ugyanis minden osztási maradékot ki kell fejezni $\varphi(n)$-nel és e-vel.

$$20 = \varphi(n) - 76e$$
$$13 = e - 3 \cdot 20 = -3\varphi(n) + 229e$$
$$7 = 20 - 13 = 4\varphi(n) - 305e$$
$$6 = 13 - 7 = -7\varphi(n) + 534e$$
$$1 = 7 - 6 = 11\varphi(n) - 839e.$$

Ebből következik, hogy $1 \equiv -839e \pmod{\varphi(n)}$. Ezért $f = -839 = 4729 \pmod{\varphi(n)}$. (Maple-ben f gyors kiszámítására használhatjuk az $msolve(73 * f = 1{,}5568)$; parancsot.)

Tehát a nyilvánosságra hozott adataink: $n = 5723$ és $e = 73$.

Tamás azt az üzenetet szeretné nekünk küldeni, hogy: $A\ KRIPTOGRAFUS\ NO\ NEVE\ SUSAN$. A 26 betűs abc-t és az $A = 01, B = 02, \ldots, Z = 26$ (szóköz$= 00$) átírást használva a következő számsort kapta:
$0100111809162015070106211900141500140522050019211901 14$.

Mivel n négy hosszú, a számsort legfeljebb három hosszú részekre kell tagolni, ekkor kapjuk meg az x_i-ket. Tamás ezután minden x_i-t az e-edik hatványra emeli és veszi a mo-dulo n maradékukat, ezáltal megkapva a c_i-ket, melyeket elküld nekünk. A Maple prog-ram segítségével c_1-et az alábbi módon határozhatjuk meg: $solve(c1 = 010 \wedge \wedge 73 mod 5723, c1)$;. Az eredmény:

$c_1 = 996$	$c_2 = 280$	$c_3 = 2054$	$c_4 = 4699$	$c_5 = 316$	$c_6 = 4096$
$c_7 = 996$	$c_8 = 3283$	$c_9 = 1160$	$c_{10} = 4154$	$c_{11} = 2275$	$c_{12} = 4154$
$c_{13} = 4288$	$c_{14} = 3384$	$c_{15} = 1$	$c_{16} = 5481$	$c_{17} = 1160$	$c_{18} = 887.$

A titkosított üzenetet az f kulcsunk segítségével visszafejthetjük: $c_1^f = 996^{4729} \equiv x_1 \pmod{5723}$

$x_1 = 10$	$x_2 = 11$	$x_3 = 180$	$x_4 = 916$	$x_5 = 201$	$x_6 = 507$
$x_7 = 10$	$x_8 = 621$	$x_9 = 190$	$x_{10} = 14$	$x_{11} = 150$	$x_{12} = 14$
$x_{13} = 52$	$x_{14} = 205$	$x_{15} = 1$	$x_{16} = 921$	$x_{17} = 190$	$x_{18} = 114.$

Ezek után az egyjegyű számok elé két darab 0-át, a kétjegyű számok elé egy darab 0-át írva megkaptuk az eredeti üzenetet, melyet már csak vissza kell helyettesítenünk betűkké.

Az eljárás hátránya azonban az, hogy nagyon időigényes egy hosszabb szöveg ilyen módú titkosítása. Általában magát a szöveget valamilyen más, gyorsabb eljárással szokás titkosítani és csak a kulcsot küldik el RSA-val. A biztonság érdekében egy kulccsal csak 2-3 üzenetet titkosítanak, majd lecserélik ezt egy másik kulcsra.

3.2. Digitális aláírás

Az RSA eljárás nem csak a szöveg titkosítására használható, segítségével az üzenet feladója is egyértelműen azonosítható. Ez azért fontos, mert így biztosak lehetünk benne, hogy valóban Tamás írt nekünk levelet, nem pedig valaki más az ő nevében. Ez az alábbi módon kivitelezhető: tegyük fel, hogy Tamás is keres két nagy p' és q' prímeket. Hozzánk hasonlóan, ő is kiszámítja az n', $\varphi(n')$, e' és f' számokat majd n'-t és e'-t nyilvánosságra hozza, ez lesz az ő nyilvános kulcsa. Ekkor az elküldeni kívánt üzenetet először a saját f' kulcsával kódolja majd a mi nyilvános kulcsunkkal és így küldi el az üzenetet. (Ezt megteheti, hiszen az algoritmusban e' és f' szerepe felcserélhető, mert $(x_i^{e'})^{f'}$ kommutatív.) Mi tudjuk a saját f-ünket, Tamás nyilvános kulcsa pedig mindenki számára rendelkezésre áll, ezáltal az üzenet meg tudjuk fejteni és biztosak lehetünk benne, hogy az üzenet tényleg Tamástól származik. Valójában az is elegendő, ha Tamás nem a teljes üzenetet, hanem annak csak egy részét titkosítja saját f'-ével, ugyanúgy tudni fogjuk, hogy tőle származik az üzenet, mégis kevesebb számolással jár.

Tegyük fel, hogy az ő számai: $n' = 4033$, $e' = 53$ a titkos f' kulcs pedig 2861. Az üzenet legyen ugyanaz, mint az előbb, csak írja oda a végére, hogy $TAMAS$. Ez a számsorrá írt változatban 002001130119-et jelent. Csak ezt a részt titkosítja f'-vel is. Ez a következő képpen néz ki: $c_{19} = (0022861 \pmod{4033})^{73} \pmod{5723}$.

$$c_{19} = 4536 \quad c_{20} = 1 \quad c_{21} = 2134 \quad c_{22} = 35$$

Mi c_{19}-et az alábbi módon fejthetjük vissza: $x_{19} = (45364729 \pmod{5723})^{53} \pmod{4033}$

$$x_{19} = 2 \quad x_{20} = 1 \quad x_{21} = 130 \quad x_{22} = 119$$

Itt is, az előzőekhez hasonlóan, az egyjegyű számok elé két darab 0-át, a kétjegyű számok elé egy darab 0-át kell írni, a kapott számsort pedig már csak betűkké kell visszaalakítani. Azt a részt, melyet Tamás a saját privát kulcsával is letitkosított, az ő digitális aláírásának nevezzük.

3.3. RSA titkosítás a Maple programcsomag segítségével

Természetesen a Maple programot nem csak a részszámításokhoz használhatjuk, hanem a kódolás teljes folyamatához is. Első lépésként meghatározunk két darab 10^{50} nagyságrendű számot. (Megpróbáltam 10^{100} nagyságrendű számokkal dolgozni, de néhány lépés után a Maple már nem tudott számolni ezekkel, mert túl nagy számok adódtak a műveletek végrehajtása során. Vagyis, ha 100 vagy többjegyű prímeket szeretnénk használni, olyan szoftverre van szükség, ami ezeket tudja kezelni.)

Az alábbi példát a Cryptologia című folyóirat 2007 októberi számában találtak alapján

készítettem.

Első lépésként a véletlenszám generátor segítségével meghatározunk két 10^{50} nagyságrendű számot, ezek lesznek A és B.

```
> A:=rand(10^50)(); B:=rand(10^50)();
A:58163789757207574331898388261357089318697759637186
B:26381882362109684030884479093509230642380228696825
```

A következő lépésben megkeressük A és B után következő p és q kiindulási prímeket.

```
>p:=nextprime(A); q:=nextprime(B);
p:58163789757207574331898388261357089318697759637233
q:26381882362109684030884479093509230642380228696943
```

Az így kapott prímekből meghatározzuk n-et és $\varphi(n)$-t.

```
>n:=p*q; phi:=(p-1)*(q-1);
n:153447025910913040655160192758728804323476030721848\
947478673450822717134900616256048610452397607871 9
phi(n):153447025910913040655160192758728804323476030 7\
218404929114615190968808566138807694166143445987744544
```

Ezekhez pedig meghatározunk egy e számot, szintén a véletlenszám generátort használva.

```
> e:=rand(10^50)();
e:91813353875673723219171606589178335466578961281289
```

Majd megnézzük, hogy e relatív prím-e n-hez. (Ha nem, akkor e-t annyiszor generáljuk újra, míg megfelelő nem lesz.) Az alábbi parancs e és n legnagyobb közös osztóját adja meg. Ha ez 1, akkor relatív prímek.

```
>igcd(e,n);
1
```

Ekkor kiszámoljuk f-et.

```
>f:=1/e mod phi;
f:11862274053288917870339819964025547756492461787434 4\
28377955407749522986140928366232569181521431367 29
```

Megadjuk a titkosítani kívánt szöveget.

```
>plaintext:="Dan Brown Digitalis erod cimu konyveben a
kriptografus no neve Susan."
```

Ekkor a szöveget számokká kell alakítani. Ezt megtehetjük az alábbi parancsal. Itt a kapott szám mindig az aktuális betű byte értéke lesz.

```
>enchiper1:=convert(plaintext,bytes);
enchiper1:[32,68,97,110,32,66,114,111,119,110,32,68,105,103,
105,116,97,108,105,115,32,101,114,111,100,32,99,105,109,117,
32,107,111,110,121,118,101,98,101,110,32,97,32,107,114,105,
112,116,111,103,114,97,102,117,115,32,110,111,32,110,101,118,
101,32,83,117,115,97,110,46]
```

Az így kapott számsort átírhatjuk egy másik számrendszerbe. Ezt azért érdemes megtenni, hogy a további műveletek során kevesebb számmal keljen dolgozni. Ekkor a sok kicsi szám helyett néhány nagyobb keletkezik (esetünkben 2) és így a hatványozásokat illetve a modulus képzést csak ezekre kell elvégezni.

```
>enchiper2:=convert(enchiper1,base,257,n);
enchiper2: [22547315970689493355435492958140659716400\
68640600922696724432440024474176242937702565432251481713 29,
58374202019634864261776973591382100803743522206149246\
0849594173922494]
```

Ekkor elkészíthetjük azt a függvényt, ami a hatványozást és a $\pmod n$-ek kiszámolását fogja végezni a következő lépésben. (A Power parancs csak a számolás gyorsítását segíti.)

```
>encoder:=(x,e,n) ->Power(x,e) mod n;
```

Végül alkalmazzuk az iménti függvényt.

```
>chipertext:=map(encoder,chiper2,e,n);
chipertext:[22547315970689493355435492958140659716400\
68640600922696724432440024474176242937702565432251481713 29,
58374202019634864261776973591382100803743522206149246\
0849594173922494]
```

A visszafejtés az alábbi módon történik: először létre kell hoznunk a dekódoló függvényt, mely az f hatványra emelést és a $\pmod n$ képzést fogja végezni,

```
>decoder:=(x,f,n) ->Power(x,f) mod n;
```

Majd a függvényt alkalmazzuk a kódolt üzenetre.

```
>dechiper1:=map(decoder,chipertext,f,n);
dechiper1: [225473159706894933554354929581406597164 00\
6864060092269672443244002447417624293770256543225148171329,
583742020196348642617769735913821008037435222061492 46\
0849594173922494]
```

Majd az így kapott számsort vissza kell alakítani abba a számrendszerbe, amiben eredetileg dolgoztunk.

```
>dechiper2:=convert(dechiper1,base,n,257);
dechiper2: [32,68,97,110,32,66,114,111,119,110,32,68,105,
103,105,116,97,108,105,115,32,101,114,111,100,32,99,105,
109,117,32,107,111,110,121,118,101,98,101,110,32,97,32,107,
114,105,112,116,111,103,114,97,102,117,115,32,110,111,32,
110,101,118,101,32,83,117,115,97,110,46]
```

Utolsó lépésként vissza kell alakítani a számsort karakterekké.

```
>dechiper3:=convert(dechiper2,bytes);
dechiper3: "Dan Brown Digitalis erodjeben a
kriptografus no neve Susan."
```

4. fejezet

Nagy prímek megtalálása

Ahhoz, hogy a kódunkat ne lehessen feltörni, nagy prímekre van szükség. De hogyan lehet prímeket találni? Erre több eljárás is létezik.

4.1. Eratoszthenész szitája

A legrégebbi ismert módszer Eratoszthenész szitája. Ez egy adott n számig az összes prímet megtalálja. Először kihúzzuk a listából az 1-et, hiszen az nem prím. A 2 viszont igen, tehát az bent marad és bekarikázzuk. Majd kihúzzuk 2 összes többszörösét, mindaddig, míg el nem érünk n-ig. A 2 után a 3 az első olyan szám, melyet eddig nem húztunk ki, tehát a 3 prím. Ekkor bekarikázzuk a 3-at és kihúzzuk 3 összes többszörösét. Az eljárást egészen $\sqrt{n}$-ig kell folytatni ahhoz, hogy megtaláljuk az összes prímet n-ig, hiszen ekkor minden ki nem húzott szám ($\sqrt{n}$ után is) prím lesz. (Az alábbi weboldalon ezt be is mutatják lépésről lépésre: `http://archives.math.utk.edu/mathtech/prime--2/Movie1.html`) A végeredményt 100-ig az alábbi kép mutatja:

4.1. ábra

Ez egy determinisztikus prímteszt. A kifejezés azt jelenti, hogyha a módszer egy számot prímnek nyilvánít, akkor valóban az is. Kis n-ek esetén gyorsan meg lehet találni a prímeket ezzel a módszerrel, ám ez az algoritmus nem a leghatékonyabb, mert belátható, hogy a lépésszám az input hosszának exponenciális függvénye. (Arról, hogy ez miért nem előnyös, a továbbiakban részletesebben is szó lesz.)

A prímkeresés egy másik fajtája a valószínűségi prímteszt. Mint ahogy a neve is mutatja, ezek biztosan nem, csak nagy valószínűséggel állapítják meg egy számról, hogy prím-e. Előnyük azonban az, hogy könnyen és gyorsan lehet velük számolni.

4.2. Kis Fermat tétel megfordítása

Egy ilyen módszer a kis Fermat tétel megfordítását használja. Ez úgy szól, hogyha p nem osztója a-nak és $a^{p-1} \equiv 1 \pmod{p}$ teljesül, akkor p prím. Ezzel csupán az a probléma, hogy nem igaz. Ha a kongruencia nem teljesül, akkor a p szám biztosan összetett, de ha teljesül, abból még nem derül ki, hogy a p szám prím-e. Léteznek ugyanis olyan számok, amelyek noha nem prímek, a kis Fermat tétel szempontjából úgy viselkednek, mintha azok lennének. Ezeket (a alapú) álprímeknek vagy pszeudo-prímeknek nevezik. A 341 például kettes alapú álprím, mert $2^{340} \equiv 1 \pmod{341}$ teljesül, azonban a 341 összetett szám. $(341 = 11 \cdot 31)$ Úgy "leplezhetjük le" az álprímeket, ha más alapot használva belátható, hogy a kongruencia nem teljesül. Azonban ez sem mindig segít, mert léteznek univerzális álprímek, amelyek minden p-hez relatív prím a alapnál prímként viselkednek, ilyen például az 1729. (Az univerzális álprímeket Carmichael számoknak is nevezik.) 1992-ben sikerült bebizonyítani, hogy noha az univerzális álprímek nagyon ritkák, mégis végtelen sok van belőlük. Így tehát a kis Fermat tétel megfordításával biztosan nem, csak nagy valószínűséggel tudjuk megmondani egy p számról, hogy prím-e.

4.1. Tétel. *Ha p nem pszeudo-prím, akkor a mudulo p prímitív maradékosztályok (azok a maradékosztályok, amik elemei p-hez relatív prímek) legfeljebb felére teljesül, hogy $1 \leq$ $\leq a \leq p-1$ esetén $a^{p-1}-1$ osztható p-vel.*

A tétel ismeretében annak eldöntésére, hogy egy p szám prím-e, használhatjuk az alábbi módszert: véletlenszerűen kiválasztunk egy egész a számot 1 és p között. Erre leellenőrizzük, hogy $a^{p-1}-1$ osztható-e p-vel. Ha nem, akkor tudjuk, hogy p nem prím. Ha igen, akkor megismételjük az eljárást. Ha például százszori ismétlés után is azt kapjuk, hogy p prím, akkor annak a valószínűsége, hogy mégis összetett kisebb, mint $(\frac{1}{2})^{100}$. Sajnos azonban az eljárás az álprímekre nem működik, azokat nagy valószínűséggel prímnek találja. (Az álprímek azért lehetnek veszélyesek, mert ha egy ilyet választunk kiindulási prímnek, akkor megkönnyítjük a támadók munkáját n faktorizálásánál. Ugyanis, ha sorban megnézi, hogy n melyik prímmel osztható, könnyebben meg fogja találni az álprímek osztóit, mert azok kisebbek, mint az álprím, ezáltal fel tudja törni az üzeneteinket.)

4.3. Miller-Rabin prímteszt

Erősebb feltételeket is lehet szabni, melyeket a prímszámok teljesítenek. Adjuk meg a $p-1$ számot $2^s \cdot d$ (ahol d páratlan) alakban. Azt mondjuk, hogy egy a szám megsérti a Meiller-Rabin-feltételt, ha $a^d-1, a^d+1, a^{2d}+1, a^{4d}+1, \ldots, a^{(2s-1)\cdot d}+1$ számok egyike sem osztható p-vel.

4.2. Tétel. *Akkor és csak akkor elégíti ki minden $1 \leq a \leq p-1$ egész szám a Miller-Rabin-feltételt, ha p prím.*

Bizonyítás. Ha p összetett, akkor bármelyik valódi osztója megsérti a Miller-Rabin-feltételt. (Ez egyszerűen látszik.) Tegyük fel, hogy p prím. Az $(a^d-1)\cdot(a^d+1)\cdot(a^{2d}+1)\cdot(a^{4d}+ +1)\cdot\ldots\cdot(a^{(2s-1)d}+1) = a^{p-1}-1$. Ekkor a kis Fermat tételt felhasználva azt mondhatjuk, hogy $a^{p-1} \equiv 1 \pmod{p}$. Ezért bármely $1 \leq a \leq p-1$ számra $a^{p-1}-1$ osztható lesz p-vel. □

A Miller-Rabin-teszten megbuknak az álprímek, tehát jobb módszer a kis Fermat tétel megfordításánál. (Léteznek olyan összetett számok, amik valamilyen a alapra kielégítik a fenti feltételt. Ezeket dörzsölt álprímeknek hívjuk.)

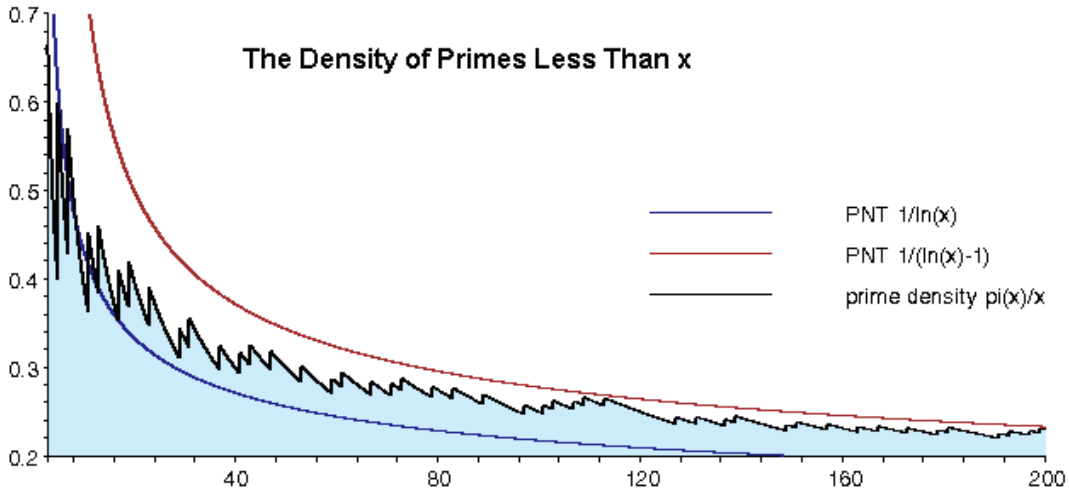
4.3. Tétel. *Ha p összetett szám, akkor a modulo p primitív maradékosztályok legalább fele megsérti a Miller-Rabin-feltételt.*

Tehát minden újabb a alap kipróbálásánál annak a valószínűsége, hogy p összetett az $\frac{1}{2}$-ére csökken, amennyiben a teszt p-t prímnek minősítette. Így annak a valószínűsége, hogy p mégis összetett, tetszőlegesen kicsire csökkenthető. Finomabb módszerekkel az is megmutatható, hogyha $1 \leq a \leq p-1$, akkor p az a alapok legfeljebb $\frac{1}{4}$-ére lehet dörzsölt álprím.

Nem biztos, hogy az első próbálkozásra már prímet fogunk találni. De a prímszámtétel segítségével megtudhatjuk, hogy körül-belül hányszor kell próbálkozni ahhoz, hogy a kívánt nagyságrendben prímet találjunk.

4.4. Tétel. *Jelölje $\pi(x)$ az x-ig terjedő prímszámok számát, ahol x pozitív egész. Ekkor $\lim_{x\to\infty} \frac{\pi(x)}{\frac{x}{lnx}} = 1$ Vagyis $\pi(x)$ és $\frac{x}{lnx}$ aszimptotikusan egyenlők.*

Az alábbi képen a prímszámok előfordulása látható 200-ig.



4.2. ábra. prímek gyakorisága 200-ig

Látható, hogy x növekedésével egyre ritkábban találunk prímszámokat. Ha $100 - -200$ jegyű prímeket szeretnénk találni, akkor nagyjából minden 4600-adik próbálkozásunk lesz sikeres. Ezt az értéket a következő képpen kaptam:

$$\frac{10^{201} - 10^{99}}{\frac{10^{200}}{ln(10^{200})} - \frac{10^{100}}{ln(10^{100})}} = 4605{,}170186.$$

Számítógéppel előállíthatunk nagy prímeket, például a Maple program segítségével, mely a Miller-Rabin tesztet alkalmazza. Első lépésként határozzuk meg, hogy hány jegyű legyen. Ezután a véletlenszámgenetáror segítségével alkossunk egy ilyen számot, ezt nevezzük el g-nek. Ezt megtehetjük a $g := rand(10 \wedge 100)()$; parancsal. Majd ezután a $nextprime(g)$; parancsal megadhatjuk a g után következő első prímszámot. Végül ugyanezt az eljárást alkalmazva megkaphatjuk a másik szükséges prímet is az RSA kódhoz.

5. fejezet

Az RSA titkosító eljárás feltörése

Vajon feltörhetőek-e az RSA-val titkosított üzenetek és ha igen, hogyan? Az algoritmus ismeretében több támadási lehetőség is adódik.

A nyilvános kulcsban $pq = n$ szorzat ismert. Ha valaki n birtokában képes megadni annak prímtényezős felbontását, akkor lényegében már fel is törte a titkosított üzenetet. Ugyanis, ha sikerült kitalálni p-t és q-t, akkor abból már $\varphi(n) = (p-1)(q-1)$ azonnal kiszámítható. Ebből pedig f dekódoló kulcs is gyorsan meghatározható, hiszen e ismert. (Nyilván n páratlan szám, hiszen ha páros lenne osztható lenne 2-vel és így a másik osztója könnyen meghatározható lenne.) Azonban a valóságban a nagyon nagy n-ek faktorizálása igen nehéz feladat. Ez többféleképpen is történhet.

5.1. Próbaosztások

A legnyilvánvalóbb módszer az, ha n-et szépen sorban megpróbáljuk elosztani $\sqrt{n}$-ig az összes prímszámmal. Azonban nagy n esetén ez túl sok időt vesz igénybe, hiszen $\sqrt{n}$-ig még meg is kell találni a prímeket. Kevesebb lépés kell, ha egy "butább" módszert alkalmazunk, vagyis prímek helyett a páratlan számokkal próbálunk meg osztani. Ekkor ugyan több osztást kell elvégezni, de nem kell azt vizsgálni, hogy az osztó prím-e. Azonban az RSA-ban használt számok olyan nagyok, hogy n osztóinak megtalálása még ezzel a módszerrel is túlságosan hosszadalmas lenne.

5.2. Fermat-eljárás

Mivel a kódban használt n két prím szorzataként áll elő, érdemesebb lehet az úgynevezett Fermat-eljárást használni. (Ez S.W. Golomb algoritmus néven is ismert.) Ennek ugyanis nem az a lényege, hogy megtalálja a szám összes prímtényezőjét, hanem az, hogy n-et felbontsa két egész szám szorzatára. Ez az alábbi módon történik: n-et felírjuk két négyzetszám különbségeként: $n = c^2 - d^2 = (c+d)(c-d)$. Ehhez először kiszá-

moljuk $\sqrt{n}$-et. Ez számítógéppel gyorsan elvégezhető. Ha ez egész, akkor $n = p^2$, ekkor a kódtörő szinte azonnal meg is fejti titkosított üzenetünket, hiszen már minden információ a rendelkezésére áll. Ezért tehát p és q választásánál nem érdemes $p = q$-t használni. Ha $\sqrt{n}$ nem egész, akkor vesszük ennek a felső egész részét, legyen ez c. Ezt követően megnézzük, hogy $\sqrt{c^2 - n} = k$ egész-e. Ha igen, akkor átrendezve azt kapjuk, hogy $c^2 - -n = k^2 \to n = c^2 - k^2$. Ekkor kész is vagyunk, hiszen $k^2 = d^2$. Ha ez sem egész, akkor c értékét minden lépésben növeljük eggyel, egészen addig, míg k egész lesz. (Abban az esetben, ha n prím, akkor nem fogunk találni ilyen k-t a triviális megoldáson kívül.)

5.1. Állítás. *Azok a számok írhatók fel két négyzetszám különbségeként, amiknek van azonos paritású osztópárjuk. A megoldásszám az azonos paritású osztópárok száma.*

Bizonyítás. Legyen $c - d = s$. Ekkor $c + d = \frac{n}{s}$. Adjuk össze $(c - d)$-t és $(c + d)$-t. Ekkor $2c = d + \frac{n}{d}$. Tehát $c = \frac{d + \frac{n}{d}}{2}$. c csak akkor lesz egész, ha d és $\frac{n}{d}$ azonos paritásúak. Hasonló eredményt kapunk, ha $(c - d)$-t és $(c + d)$-t kivonjuk egymásból □

Ha n páratlan, akkor következik, hogy minden osztópár előállítható így, mivel minden osztópár páratlan, tehát azonos paritású. Mivel az RSA algoritmusban n két prím (tehát két páratlan szám) szorzataként jön létre, az iménti eljárás meg fogja találni n osztóit. A nyilvánvaló megoldáson kívül (1 és n) csak egy megoldást fogunk kapni. Ez az eljárás is eléggé lassú, azonban, ha a két választott p és q szám közel van egymáshoz, ez az algoritmus könnyen rájuk bukkanhat. Ezért érdemes kellően távoli prímeket használni, ekkor ez a módszer sem fogja őket megtalálni.

5.3. Komplementer prímszita

Egy másik lehetséges módszer n faktorizálására komplementer prímszita nevű eljárás. Ez az alábbi tételre épít:

5.2. Tétel. *Minden olyan prímszám, mely* 3*-nál nagyobb, vagy* $6k + 1$ *vagy* $6k - 1$ *alakú, ahol* $k = 1,2,3,\ldots$ *természetes számok.*

5.3. Megjegyzés. A tétel nem azt mondja, hogy ez az alak minden k számra prímeket állít elő!

Bizonyítás. Tegyük fel, hogy van olyan prím, ami nem ilyen alakú. Ekkor az a következő alakokban állhat elő:

$6k \pm 2 = 2(3k \pm 1)$, de ez páros, tehát nem prím,

$6k \pm 3 = 3(2k \pm 1)$, ez osztható 3-mal tehát nem prím,

$6k \pm 4 = 2(3k \pm 2)$, ez páros, tehát nem prím,

$6k \pm 5$ az ilyen számok azonban $6k \pm 1$ alakba írhatók. Így ellentmondásra jutottunk. □

Ezt használja fel Dénes Tamás tétele:

5.4. Tétel. *Legyenek n, k, u, v természetes számok és $u, v \geq 1$. Ekkor az $n = 6k + 1$ összetettszám, ha $k = 6uv + u + v$ vagy $k = 6uv - u - v$. Az $n = 6k - 1$ összetettszám, ha $k = 6uv - u + v$ vagy $k = 6uv + u - v$.*

Bizonyítás. Először tekintsük $n = 6k + 1$ és $k = 6uv + u + v$ esetet. Tehát $n = 36uv + +6u+6v+1 = (6u+1)(6v+1)$. Ebből következik, hogy összetett, hiszen felírtuk szorzat alakban. Ez biztosan nem a triviális $1 \cdot n$ alak, mivel u és v legalább 1, így mindkét tényező nagyobb 1-nél. (A többire ugyanígy.) □

Ezek alapján az algoritmus a következő: Először eldöntjük az adott n számról, hogy $6k+1$ vagy $6k-1$ alakú-e. Ha például $6k+1$alakú, akkor legyen $y_0 = \frac{n-1}{6} i = 1,2,3, \ldots$. Ha $y_0 - i \equiv 0 \pmod{6i+1}$, akkor megvan az egyik prím, ez a $p = 6i + 1$, ebből pedig egyértelműen következik a másik osztó is. Ha a kongruencia nem teljesül, meg kell próbálni a következő i-vel. A többi esetre is hasonlóképpen kell alkalmazni az algoritmust. Ez azért jó, mert könnyen és gyorsan végrehajtható műveleteket tartalmaz, azonban nagy n esetén előfordulhat, hogy túl sok i-t kell kipróbálni, ekkor viszont sajnos ez is túlságosan időigényes.

Ezeken kívül még számos eljárás ismert a prímfaktorizációra, ám ilyen nagy számoknál, melyeket az RSA használ, egyik sem vezetett lényeges eredményre. Ezért úgy lehetünk biztosak abban, hogy jól választottuk meg p-t és q-t, hogy megpróbáljuk n-re alkalmazni az összes faktorizációs eljárást. Ha egyik sem vezet eredményre, akkor a prímek megfelelőek. Ez az ellenőrzés azonban szintén időigényes.

Esetleg gondolhatnánk arra, hogy $pq = n$ és $(p-1)(q-1) = \varphi(n)$ biztos közel vannak egymáshoz, ezért próbálgatással rábukkannánk $\varphi(n)$-re, melyből e ismeretében meg lehetne határozni f-et az euklideszi algoritmussal. Mivel $pq = (p-1)(q-1)+p+q- -1 = \varphi(n)+p+q-1$, látszik, hogy a keresett $\varphi(n)$ mégsincsen annyira közel, ezért ismét ott tartunk, hogy szükség lenne n faktorizációjára, hiszen ekkora számoknál a távolság is olyan nagy, hogy lehetetlen rövid időn belül kipróbálni ennyi $\varphi(n)$-t majd mindhez f-et konstruálni, végül minden f-el kiszámolni, hogy vajon megoldja-e a kongruenciát.

Egy másik módja lehet az RSA feltörésének, ha az f kulcsot próbálnánk meg kitalálni. Ugyanis nem tudjuk, hogy f az euklideszi algoritmuson kívül még hányféleképpen állítható elő, illetve azt sem, hogy csak egy olyan f érték létezik-e, mely teljesíti a kívánt kongruenciát.

Sokak szerint az RSA feltörése és a prímfaktorizáció egyenértékű feladat. Amíg nem találunk általános, gyors algoritmust egy szám osztóinak felírásához, addig az RSA is biztonságos marad. Az persze lehetséges, hogy a számítógépek teljesítménye nő, de ekkor elegendő még nagyobb prímeket használni a kódoláshoz, mert a prímkeresés gyorsabb

eljárás a faktorizációnál. Bizonyítani azonban mindmáig nem sikerült a séma feltörhetetlen mivoltát, sőt, volt már rá példa, hogy sikerült megfejteni egy konkrét üzenetet, holott a prímek jól voltak megválasztva. Ekkor több számítógép egyszerre dolgozott a faktorizáción, megosztva a számítási munkálatokat, de még így is hónapokig tartott a nyílt szöveg feltárása. A Digitális erődben is olvashatunk hasonló technikáról. Itt 3000000 processzort kapcsoltak össze, így egy óriási teljesítményű gépet létrehozva, mellyel csupán néhány perc bármely RSA kód feltörése. Ezt a gépet a könyvben TRANSLTR-nek hívják, az eljárást pedig "nyers erőnek".

6. fejezet

Az RSA nem megfelelő alkalmazásából adódó feltörhetőség

6.1. Túl közeli prímek

Mint ahogy az az imént kiderült a Fermat-eljárásból, érdemes a prímeket "jól" megválasztani.

Azonban nem csak a Fermat eljárás miatt érdemes távoli prímeket alkalmazni. Fermat egy másik tétele ugyanis kimondja, hogy m tetszőleges egész számot ismételten szorozva önmagával és egy adott n modulust használva, véges sok lépésben visszakapjuk m-et. Ez akkor is igaz, ha m hatványaira alkalmazzuk a beszorzást, illetve ha a számot ismételten hatványaira emeljük. Így előfordulhat, hogy az üzenet csupán néhány lépésben megfejthetővé válik, a kiindulási prímek ismerete nélkül is. Az alábbi példát az interneten találtam, Dénes Tamás matematikus oldalán. (`http://www.titoktan.hu/_raktar/_e_vilagi_gondolatok/HTRSA.htm`)
Azért szeretném ezt itt felhasználni, mert jól mutatja azt, hogy nagy számokra is hatékony lehet az RSA feltörése, ha túl közeli prímeket választunk.

"60 jegyű RSA modulussal dolgozunk, mégis 6 lépésben megvan a megoldás.
Legyen p és q két 30 decimális jegyű prímszám.

$p = 586203142714212103540772438083$, $q = 797648851083268082237297393713$,
valamint $n = 467584263287392317254879360844851439325139765393920565972179$,
$e = 227104576216343623581376514176931506454984828057257408953697$.

Tehát az RSA publikus kulcs (n, e). Ezt ismerheti bárki. Vegyük az m üzenetet.

$m = 200805001301200805130120090301120009142005121209070514030518$.
$c = m^e \pmod{n} \equiv 558899877578499654114721338354996634436912630687583541630 1$

A kapott c-t emeljük többször az e hatványra és tekintsük n-el vett maradékát.

$c_1 = c = 55889987757849965411472133835499663443691263068475835416301$
$c_2 = c_1^e \pmod n = 22375083207688556707559653392152057143266981783101556784702 9$
$c_3 = c_2^e \pmod n = 12557068219229988475528558884549133456288937978998761750574 0$
$c_4 = c_3^e \pmod n = 39190579928879931623006653552442837298453037272553084845891 94$
$c_5 = c_4^e \pmod n = 44320621574466035847763721403724856989333600141396740167 14$
$c_6 = c_5^e \pmod n = 20080500130120080513012009030112000914200512120907051403051 8$
$c_7 = c_6^e \pmod n = 55889987757849965411472133835499663443691263068475835416301$

$c_7 = c_1 \rightarrow c_6^e \pmod n = m^e \pmod n$ tehát adódik, hogy $c_6 = m$, azaz sikerült megfejteni a nyílt üzenetet a titkos kulcs ismerete nélkül, és ez mindössze 6 lépést igényelt."

Az ilyen fajta feltörés úgy védhető ki, ha távoli prímeket használunk, ezáltal a ciklus elegendően hosszú lesz ahhoz, hogy ez az eljárás se vezessen eredményre rövid időn belül.

6.2. Kicsi f használata

Egy rendszer a számítások gyorsítása érdekében generálhat kis f-et is. Azonban ha f egy adott értéknél kisebb, akkor az M. Wiener tétel alapján az RSA feltörhető.

6.1. Tétel. *Legyen $n = pq$ úgy, hogy $q < p < 2q$ és legyen $f < \frac{1}{3}n^{\frac{1}{4}}$. Ha adott n, e úgy, hogy $ef \equiv 1 \pmod{\varphi(n)}$, akkor f hatékonyan megfejthető.*

Mivel manapság n általában 1024 bites szokott lenni, a tétel alapján f legalább 256 bites kell, hogy legyen. A tételt egyébként sikerült tovább erősíteni, ez Boneh és Durfee nevéhez fűződik, az általuk már biztonságosnak vélt korlát: $f > n^{0.292}$.

6.3. Ugyanaz az e használata

Üzeneteink megfejthetővé válnak, ha kiindulásként ugyan más prímeket használunk, de minden n-hez ugyanazt az e-t alkalmazzuk. Ez a kínai maradéktételből következik.

6.2. Tétel. *Ha $(n_1, n_2) \mid c_2 - c_1$, akkor az*

$$x \equiv c_1 \pmod{n_1}$$

$$x \equiv c_2 \pmod{n_2}$$

szimultán kongruenciarendszer bármilyen c_1 és c_2 egész szám esetén megoldható és ha x_0 megoldás, akkor $x_1 \equiv x_0 \pmod{[n_1, n_2]}$

Bizonyítás. $x = n_1y_1+c_1$ és $x = n_2y_2+c_2 \to n_1y_1+c_1 = n_2y_2+c_2 \to n_1y_1-n_2y_2 = c_2-$ $- c_1$ ez pedig egy lineáris diofantikus egyenlet, ami megoldható, ha $(n_1, n_2) \mid c_2 - c_1$. Ha x_0 megoldás, akkor $x_0 \equiv c_1 \pmod{n_1}$ és $x_0 \equiv c_2 \pmod{n_2}$. Tehát x_1 csak akkor megoldás, ha $x_1 \equiv c_1 \pmod{n_1}$ és $x_1 \equiv c_2 \pmod{n_2}$. Ebből következik, hogy $x_1 \equiv$ $\equiv x_0 \pmod{n_1}$, tehát $n_1 \mid x_1 - x_0$ és $x_1 \equiv x_0 \pmod{n_2}$, tehát $n_2 \mid x_1 - x_0$. Ekkor $[n_1, n_2] \mid x_1 - x_0$. Ez pedig csak akkor van, ha $x_1 \equiv x_0 \pmod{[n_1, n_2]}$. (Speciális eset, ha $(n1, n2) = 1$, ekkor mindig megoldható és a megoldások egyetlen maradékosztályt alkotnak modulo n_1n_2.) □

Ezt használja a kínai maradéktétel, melyet Szun Cu kínai matematikus már időszámításunk előtt körül-belül 2000 ével is ismert.

6.3. Tétel. *Legyenek $n_1, n_2, \ldots, n_k$ páronként relatív prímek. Ekkor az*

$$\begin{aligned} x &\equiv c_1 \pmod{n_1} \\ x &\equiv c_2 \pmod{n_2} \\ &\ldots \\ x &\equiv c_k \pmod{n_k} \end{aligned}$$

szimultán kongruencia rendszer bármilyen $c_1, \ldots, c_k$ egészek esetén megoldható és a megoldások egyetlen maradékosztályt alkotnak modulo $n_1n_2 \ldots n_k$.

Bizonyítás. Az előző tételből k szerinti teljes indukcióval adódik. $k = 2$ esetén ez maga az előző tétel. Tegyük fel, hogy a tétel igaz $k-1$ kongruenciára. Ekkor tekintsük a k kongruenciából álló rendszert. Az iterációs feltételből tudjuk, hogy a $k-1$ kongruenciából álló rendszert kielégítő számok egyetlen maradékosztályt alkotnak modulo $n_1n_2 \ldots n_{k-1}$. Ezáltal az első $k-1$ darab kongruenciát kicserélhetjük egyetlen darabra. Ekkor az alábbiakat kapjuk:

$$\begin{aligned} x &\equiv c \pmod{n_1n_2 \ldots n_{k-1}} \\ x &\equiv c_k \pmod{n_k} \end{aligned}$$

Ebből pedig visszakaptuk az előző tételt. □

A tétel alkalmazásánál elég a $k = 2$ esetet megfigyelni. Tehát

$$\begin{aligned} x &\equiv c_1 \pmod{n_1} \\ x &\equiv c_2 \pmod{n_2}. \end{aligned}$$

Az előzőekből tudjuk, hogy van olyan y_1, y_2 egész szám, amire $c_2 - c_1 = n_1y_1 - n_2y_2$. Ez nem határozza meg egyértelműen y_1, y_2 számokat, de az is elég, ha az euklideszi algoritmussal találunk olyan y_1', y_2' számokat, amire $c_2 - c_1 = n_1y_1' - n_2y_2'$. Ilyen számokat pedig az euklideszei algoritmus ad ha $(n_1, n_2) = 1$.

Az RSA algoritmusban ugyan nem x-el, hanem x^e-vel dolgozunk, de könnyen látható, hogy ez nem befolyásolja a támadókat abban, hogy üzenetünket feltörhessék.

6.4. Több felhasználó közös modulussal

Tegyük fel, hogy Tamás is ugyanazokat a prímeket választotta, mint mi. Ekkor a mi nyilvános kulcsunkban ismert n és e. Mivel ugyanazok volta a kiindulási számaink, Tamás tudni fogja n prímtényezős felbontását és $\varphi(n)$-t is. Ezek alapján a nyilvános e kulcsunk felhasználásával már könnyedén ki tudja számítani az f-ünket, ezáltal meg tudja fejteni az üzeneteinket. Persze kicsi annak a valószínűsége, hogy két ember véletlenül ugyanazokat a prímeket válassza. Azonban van olyan, hogy egy adott rendszer minden felhasználónak generál egy kulcsot. Ekkor a számítások csökkentése érdekében előfordul, hogy meghatároz egy $n = pq$ számot és ehhez ad minden felhasználónak egy saját e-t és f-et. A prímeket, illetve $\varphi(n)$-t ugyan nem közli a felhasználóval, de a Simmons tétel alapján, mely kimondja, hogy n faktorizációja végrehajtható e és f ismeretében, az azonos modulussal rendelkező felhasználók fel tudják törni egymás üzeneteit.

Ez a következő módon történhet: minden felhasználó tudja a saját e és f értéket. Ismert továbbá, hogy $ef \equiv 1 \pmod{\varphi(n)}$. Legyen $k = ef - 1$, tehát k többszöröse $\varphi(n)$-nek. k-t felírhatjuk a következő alakban: $k = 2s \cdot d$, ahol d páratlan és $s \geq 1$. (Mivel $\varphi(n)$ páros, biztos hogy a 2 osztója.) Az Euler-Fermat tétel alapján tudjuk, bármely g-re, ahol $(g, n) = 1$ teljesül $g^k \equiv 1 \pmod{n}$. Ekkor $g^{\frac{k}{2}} = \sqrt{g^k}$. A kínai maradéktétel alapján 1-nek négy négyzetgyöke van modulo n. Ebből kettő a ± 1. A másik kettő a $\pm x$, ahol $x \equiv 1 \pmod{p}$ és $x \equiv -1 \pmod{q}$. Ebből például $x - 1$ és n legnagyobb közös osztója meg fogja adni $p - t$, így már q is meghatározható.

6.5. Azonos üzenetek, de különböző f-ekkel titkosítva

Tegyük fel, hogy Tamás rajtunk kívül még mással is levelezésben áll és van olyan üzenet, melyet mindegyikőnknek el akar küldeni. Ekkor, ha ugyanazzal a modulussal, de más (e-kel és) f-ekkel dolgozik (melyek egymáshoz relatív prímek), ezt az üzenetet, melyet többen is megkaptunk, a támadók meg tudják fejteni.

Legyen $x^{f_1} \equiv c_1$ és $x^{f_2} \equiv c_2 \pmod{n}$. Ekkor megadható olyan y_1, y_2 szám, amire $y_1 f_1 + y_2 f_2 \equiv 1 \pmod{n}$. Ekkor $c_1^{y_1} c_2^{y_2} \equiv x^{y_1 f_1 + y_2 f_2} \equiv x \pmod{n}$, ahol x a titkosítatlan üzenet.

6.6. Informatikai támadások

A támadások egy másik csoportja az, amely inkább informatikai jellegű. Egy ilyen módszert dolgozott ki Adi Shamir Nicki von Sommerennel. Ez a módszer a pirvát kulcsot akarja megtalálni. A számítógépek ezt az adatot általában a merevlemezen tárolják, olyan fájlokban, amikhez csak nagyon kevés felhasználónak van hozzáférési joga és maga a privát kulcs is titkosított formában van. Ezt csak a célalkalmazás tudja dekódolni, ekkor

azonban a kulcs átkerül a memóriába kódolatlan formában. Ha a támadó valamilyen módon itt hozzá tud férni, akkor meg tudja szerezni a privát komponenst. Azt azonban hozzá kell tenni, hogy a memóriában akár több száz megabytnyi adat is lehet, melyben nem egyszerű egy 1024 bites kulcs megtalálása. Egyéb implementáció elleni támadás lehet még a számítási idő vagy az áramfelvétel mérése. Azonban ezek alkalmazásához magas számitástechnikai tudásra van szükség, ezért erre nem térek ki.

7. fejezet

Az RSA biztonságossága

Egy matematikai feladat esetén nemcsak az a fontos, hogy legyen algoritmus annak megoldására, hanem az is, hogy a lehető leggyorsabban jussunk eredményhez, tehát, hogy az algoritmus lépésszáma a lehető legkisebb legyen. (Lépésszám alatt azt értem, a hogy szükséges műveletek elvégzéséhez mennyi lépés kell. Például összeadjuk n és m számokat. Ha feltesszük, hogy n a nagyobb abszolút értékű, akkor a műveletet legfeljebb $(n$ mérete$) \cdot 2$ lépésben elvégezhetjük.)

Tekintsük az alábbi példát: kiszínezhetőek-e egy adott gráf csúcsai két színnel úgy, hogy az azonos színűeket ne kösse össze él? Erre legalább kétféleképpen adhatunk megoldást. Nézzük végig az összes lehetséges két színnel való színezést, és ha találunk olyat, ami megfelelő, akkor létezik ilyen. Minden csúcsot két színnel színezhetünk, tehát egy n csúcsú gráf esetén $2n$-féle színezés lehetséges. (Ahol n a csúcsok száma.) Már $n = 100$-ra is olyan nagy szám adódik, hogy túl hosszú ideig tartana mindezt végignézni, ezért ezt a megoldást, noha eredményre vezet, mégsem szokás alkalmazni.

A másik eljárás az, ha az egyik csúcsot találomra kiszínezzük valamelyik színnel. Majd azokat a csúcsokat, melyeket ezzel él köt össze, az ellenkező színre színezzük. Így haladunk egészen addig, míg van színtelen csúcs. Ha az eljárás közben nem találunk olyan csúcsot, melyet azonos színűre kellene színezni valamely szomszédjával, akkor a színezés lehetséges. Ennek a lépésszáma jóval kisebb, csupán $n\frac{n-1}{2}$. (Itt n a csúcsok számát jelöli a gráfban. Ekkor az iménti képlet az élek maximális számát adja meg, ha a gráfban nincs hurokél. Mivel a feladat eldöntéséhez elég minden élet megvizsgálni, a lépésszám az élszámmal egyenlő.) Míg az előzőnél a lépésszám n méretével exponenciálisan nő, addig itt csupán négyzetesen, ezért a második megoldás tekinthető gyakorlati szempontból is használhatónak.

Általános esetben olyan algoritmusokat szeretnénk használni, ahol a lépésszám korlátozható egy olyan n^k függvénnyel, ahol k egy konstans, n pedig a bemeneti adatok száma. Ekkor az algoritmus polinomiális. Tudjuk, hogy például az összeadás, kivonás, szorzás, hatványozás a modulo m maradékosztályok körében vagy az m-hez relatív prímmel való osztás polinomiális.

7.1. Definíció. *Eldöntendő problémának hívjuk az olyan problémákat, ahol az input egy olyan kérdés, amire az output válasz igen vagy nem.*

7.2. Definíció. *P-belinek nevezzük azokat az eldöntési problémákat, amik az input méretének polinomiális függvényével felülről becsülhető időben megoldhatók.*

Tehát, hogy egy adott gráf kiszínezhető-e két színnel, az P-beli probléma, hiszen $n\frac{n-1}{2} < n^2$. Vannak azonban olyan problémák, melyek eldöntésére még nem ismerünk polinom idejű algoritmust. Ilyen például az, hogy egy adott gráf kiszínezhető-e három színnel úgy, hogy az azonos színű pontok között nem fut él, vagy az, hogy egy adott gráfban létezik-e Hamilton-kör.

7.3. Definíció. *NP-belinek nevezzük azokat az eldöntendő problémákat, ahol minden olyan I inputhoz, melyre a válasz igenlő, létezik olyan T tanú, hogy T hossza felülről becsülhető I hosszának egy polinomjával és I és T ismeretében polinom időben ellenőrizhető, hogy a válasz igenlő. (NP jelentése: nemdeterminisztikusan polinomiális.)*

Ilyen például annak az eldöntése, hogy egy gráf síkbarajzolható-e.

Ha p_2 probléma visszavezethető p_1-re és p_1-ről tudjuk, hogy P-beli, akkor p_2 is P-beli. Nyilvánvaló, hogy $P \subseteq NP$, azt azonban a matematikusok máig nem tudják, hogy $P = NP$ teljesül-e. (Bár sokuk szerint nem.)

Egy probléma NP-nehéz, ha minden NP-beli probléma visszavezethető rá. Ha ő maga is NP-beli, akkor a problémát NP-teljesnek hívjuk. Ha egy ilyen problémát polinom időben meg tudnánk oldani, akkor az összes NP-beli probléma is polinom időben megoldható lenne. (Az, hogy egy gráf tartalmaz-e Hamilton-kört, NP-teljes probléma.)

7.4. Definíció. *$co-NP$-belinek nevezzük azokat az eldöntendő problémákat, ahol minden olyan I inputhoz, melyre a válasz nemleges, létezik olyan T tanú, hogy T hossza felülről becsülhető I hosszának egy polinomjával és I és T ismeretében polinom időben ellenőrizhető, hogy a válasz nemleges.*

Például az, hogy igaz-e, hogy egy adott gráfban nincs Hamilton-kör, $co-NP$-beli probléma.

Létezik olyan probléma is, ami NP-$co-NP$-ben van. Ilyen például az, hogy egy páros gráfban van-e teljes párosítás. A sejtés az, hogy minden ilyen probléma P-beli lehet, mert sokról belátták hogy az, de egyikről sem tudták belátni, hogy nem az.

Azért taglaltam ilyen részletesen a problémaosztályokat, hogy jobban tudjam szemléltetni azt, hogy mit használ ki az RSA algoritmus. Az e és f kulcs ismeretében a szöveget gyorsan lehet kódolni illetve visszafejteni, mert az ehhez szükséges lépések polinomiális időben végrehajthatók. A támadóknak azonban már nincs ilyen könnyű dolga. Ha adott egy n szám, akkor az, hogy prím-e polinom időben eldönthető. De ezt is csak 2002-ben sikerült bebizonyítani, ami Agrawal, Kayan és Saxena nevéhez fűződik.

Ezt követően, ha már tudjuk, hogy n összetett, akkor szeretnénk megadni egy valódi osztóját. Ebben a formában ez még nem egy eldöntendő probléma, de átfogalmazással azzá tehető: n számnak létezik-e k-nál nem nagyobb valódi osztója? Ez viszont már egy NP-beli probléma. Látható, hogy ezzel meg tudnánk adni n prímfelbontását, azonban még nem ismert ennek megoldására polinom idejű algoritmus. Jelenleg az az eljárás ismert, hogy végigpróbálgatjuk a prímeket, hátha valamelyik osztja n-t, így egyszer biztosan megkapjuk a felbontását. A baj az, hogy ez n méretével exponenciálisan növő lépésszámú. A Digitális erődben Dan Brown erre használja a "Bergofsky-elv"'elnevezést. Ez azt jelenti, hogy véges számú lehetőség esetén, ha elkezdjük sorban kipróbálni, hogy egy prím osztója-e n-nek, akkor előbb-utóbb biztosan megtaláljuk az osztókat. Ezért az RSA biztonságát nem az adja, hogy nem lehet kitalálni a kulcsot, hanem az, hogy senkinek nincs elég ideje illetve megfelelő technikai háttere hozzá. A "Bergofsky-elv" egyébként igaz, de a valóságban ez az elnevezés nem létezik.

Annak ismeretében, hogy a prímség elsöntése P-beli, nem zárható ki, hogy n faktorizációja is az. Sőt, A. Lenstra, H. Lenstra és Lovász bebizonyították egy ehhez hasonló problémáról, hogy P-beli. Ez az egész együtthatós polinomok irreducibilis tényezők szorzatára bontása volt. Ha tényleg létezik olyan algoritmus, mely polinomiális időben tudja faktorizálni n-t, akkor minden RSA-val titkosított üzenet gyorsan feltörhetővé válik. Egyelőre azonban nem ismerünk ilyen algoritmust.

Az egyik elgondolás az NP-beli problémák polinomiális időben való megoldására a kvantumszámítógépek használata. Ezek párhuzamosan több művelet elvégzésére lennének képesek, így ha nincs jobb algoritmus annál, mint végignézni az összes lehetséges esetet, akkor sem lenne semmi baj, mert bármilyen véges számú lehetőséget gyorsan meg tudna vizsgálni. Történtek már kísérletek ilyen gépek kifejlesztésére, létezik olyan, mely egyszerre három bittel tud dolgozni, sokan azonban hitetlenek a tényleges megvalósítást illetően. Mindenesetre már létezik az az algoritmus, ami kvantumszámítógéppel polinom időben tudja faktorizálni n-et. (Ez négyzetes növekményű végrehajtási időt igényel.) Ezt 1994-ben publikálta Peter Shor. (Gyöngyösi László: Kvantum-prímfaktorizáció című pdf-jében az olvasható hogy egy ilyen számítógéppel egy RSA-kód feltörése csupán néhány másodpercet vett igénybe. A megfogalmazás azt sugallja, az algoritmus kipróbálásra került egy kvantumszámítógépen, tehát ilyen gépek már léteznek.) A Digitális erőd ezt az új technikát is említi. Ebben a TRANSLTR nemcsak több millió processzor együttese, hanem kvantum számítógépeké is. (Bár az előzőek alapján ha egy kvantumszámítógép néhány másodperc alatt képes feltörni az RSA-t, teljesen felesleges még hárommillió processzort összehangolni vele.)

Megjegyzem, hogy attól még, hogy egy algoritmus exponenciális idejű, még nem biztos, hogy abszolút alkalmazhatatlan. Ha például egy algoritmus általános esetben exponenenciális, könnyen lehet, hogy speciális esetben polinomiális. Ilyen egy hálózatban a maximális kapacitású vágás meghatározására létezik polinom idejű algoritmus, ha a gráf síkbarajzolható.

8. fejezet

Az RSA gyakorlati alkalmazása

Az RSA-val kapcsolatos forrásokat tanulmányozva az olvasónak úgy tűnhet, hogy ezt a titkosítási eljárást biztos csak matematikusok használják. Ez azonban nem igaz. Valójában minden ember használja, aki szokott internetezni. RSA-t alkalmazzák ugyanis az SSL-rendszerek (Secure Socket Layer) ezeket pedig a biztonságos weboldalak használják. Onnan tudhatjuk, hogy ilyen oldalon vagyunk, hogy az oldal címe https-el kezdődik. (Ami nem titkosított, abban csak http van.) Ilyen biztonságos oldal a facebook, a különböző e-mail rendszerek vagy e-bank oldalak. Így biztosítják azt, hogy például a bankkártyánk adataihoz senki ne férhessen hozzá egy esetleges kártyahasználat alkalmával. (A Digitális erődben pedig azért kellet létrehozni a TRANSLTR-t, a kódfeltörő gépet, hogy az RSA-val titkosított e-maileket is el tudják olvasni.) Amikor ilyen oldalra látogatunk, akkor a számítógépünk azonnal generál nekünk egy nyilvános és egy titkos kulcsot is. Ebből a felhasználó maximum annyit vesz észre, hogy az adott oldallal történő kapcsolatfelvétel egy kicsit tovább tart a megszokottnál.

Manapság a 100 jegyű prímeken alapuló titkosítók már nem tekinthetők biztonságosnak, ezért az e-kereskedelem körül-belül 200 jegyű, míg a katonaság 400 jegyű prímekkel dolgozik.

Irodalomjegyzék

[1] Judy A. Holdener, Eric J. Holdener, A Cryptographic Scavenger Hunt. *Cryptologia*, (2007), 31:316-323.

[2] Dan Brown, *Digitális erőd*, Gabo Kiadó, Budapest, 2005.

[3] Paul Lunde, *Titkos kódok*, Kossuth Kiadó, Kína, 2010.

[4] Freud Róbert, Gyarmati Edit, *Számelkélet*, Nemzeti Tankönyvkiadó Rt., Budapest, 2000.

[5] Katona Gyula Y., Recski András, Szabó Csaba, *A számítástudomány alapjai*, Typotex Kiadó, Budapest, 2002.

[6] Lovász László, Pelikán József, Vesztergombi Katalin, *Diszktét matematika*, Typotex Kiadó, Budapest, 2010.

[7] Megyesi Zoltán, *Titkosírások*, Szalay Könyvkiadó és Kereskedőház Kft., Kisújszállás

[8] Dan Boneh, *Twenty years of attacks on the RSA cryptosytem*,
`https://crypto.stanford.edu/~dabo/papers/RSA-survey.pdf`

[9] Lovász László, *Algoritmusok bonyolultsága*,
`http://www.cs.elte.hu/~kiraly/alg.pdf`

[10] Demetrivics János, Katona Gyula, *Az algoritmusok bonyolultsága*,
`http://www.termeszetvilaga.hu/kulonsz/k002/algoritmus.html`

[11] Varga Zsolt, *Modern Prímfaktorizáció*,
`https://www.cs.elte.hu/blobs/diplomamunkak/bsc_alkmat/2012/varga_zsolt.pdf`

[12] Gyöngyösi László, *Kvantum-prímfaktorizáció*,
`http://www.mcl.hu/quantum/foliak/kvantum_primfakt1.pdf`

[13] Dénes Tamás,
http://www.titoktan.hu/_raktar/_e_vilagi_gondolatok/HTRSA.htm

[14] http://hu.wikipedia.org/wiki/RSA-elj%C3%A1r%C3%A1s

[15] http://titanic.nyme.hu/~gmsi/gmphd_minta.pdf

[16] http://home.sandiego.edu/~dhoffoss/teaching/cryptography/10--Rabin-Miller.pdf

[17] https://www.mozaweb.hu/Lecke-MAT-Sokszinu_matematika_6--7_Primszamok_osszetett_szamok-105639

[18] https://primes.utm.edu/howmany.html

[19] http://archives.math.utk.edu/mathtech/prime-2/Movie1.html