EÖTVÖS LORÁND TUDOMÁNYEGYETEM
TERMÉSZETTUDOMÁNYI KAR

Az RSA kriptográfiai algoritmus paraméter-követelményeinek változása

BSc Szakdolgozat

Készítette: Szalkai Tamás
Matematika BSc, Matematikai elemző szakirány

Témavezető: Szabó István
Eötvös Loránd Tudományegyetem, Természettudományi Kar
Valószínűségelméleti és Statisztika Tanszék

Budapest, 2021

NYILATKOZAT

Név: Szalkai Tamás

ELTE Természettudományi Kar, szak: Matematika BSc

NEPTUN azonosító: A4T1I7

Szakdolgozat címe:
Az RSA kriptográfiai algoritmus paraméter-követelményeinek változása

A **szakdolgozat** szerzőjeként fegyelmi felelősségem tudatában kijelentem, hogy a dolgozatom önálló szellemi alkotásom, abban a hivatkozások és idézések standard szabályait következetesen alkalmaztam, mások által írt részeket a megfelelő idézés nélkül nem használtam fel.

Budapest, 2021.05.14

Szalkai Tamás

a hallgató aláírása

Köszönetnyilvánítás

Szeretnék köszönetet mondani Szabó István témavezetőmnek, a szakdolgozatom témájának megválasztásától kezdődően nyújtott minden segítségéért, a kérdéseimre adott gyors válaszaiért, türelméért és megértéséért.
Továbbá szeretném megköszönni a családomnak, hogy az egyetemi éveim alatt végig mellettem álltak és támogattak.

Tartalomjegyzék

1. Bevezetés **1**

2. A nyilvános kulcsú titkosítás **2**

3. Szabványok **3**

4. Rivest, Shamir és Adleman publikációja **5**

4.1. RSA algoritmus 5

4.2. Kezdeti ajánlások a paraméterek megválasztására 6

5. RSA követelményeinek a változásai **8**

5.1. Az 1988-as X.509 szabvány 8

5.2. Követelmények változásai néhány korábbi ETSI szabvány bemutatásával 11

5.2.1. ETSI 2003-03 11

5.2.2. ETSI 2005-2011 15

5.2.3. ETSI 2014 16

5.2.4. ETSI 2017-től 17

5.3. NIST szabvány bemutatása 18

5.3.1. Követelmények 2000-től 18

5.3.2. Követelmények 2005-től 19

5.3.3. Követelmények 2009-től 20

5.3.4. A 2011-es NIST SP 800-131A publikáció 24

5.3.5. Követelmények 2014-től 24

5.4. Paraméterek hosszának időbeni változásai 26

5.5. Lehetséges változások a jövőben 30

6. Jelölések és felhasznált matematikai alapismeretek **32**

7. Zárszó **34**

8. Függelék **37**

9. Irodalomjegyzék **40**

1. Bevezetés

A titkosítás története több ezer évre nyúlik vissza. Szerepe volt fontosabb történelmi események alakulásában is. Az idők során több módszert is kidolgoztak, az egyik legnagyobb áttörést viszont a XX. században sikerült elérni, amikor először publikálták az aszimmetrikus rejtjelezés alapgondolatát. Ennek első technikai megoldása Ron Rivest, Adi Shamir és Leonard Adleman nevéhez kötődik, módszerüket ezért nevezik RSA algoritmusnak.

A szerzők, a kezdeti publikációban a munkájuk ismertetése után ajánlást adnak arra is, hogy milyen paraméterekkel érdemes az algoritmust használni a megfelelő biztonság érdekében. Ezután az RSA elterjedésével számos kutatási eredmény született, melyekkel egyes gyenge pontjaira világítanak rá. Emellett a számítógépek teljesítménye is rohamos tempóban fejlődik, ezért a paraméterekkel szembeni követelményeket többször újra kellett gondolni.

A szakdolgozatomban először a nyilvános kulcsú titkosítás alapjairól írok, majd röviden kitérek néhány szabványügyi szervezet bemutatására. Az ezt követő fejezetben, az RSA alapötletéről és a kezdeti publikációban ajánlott paraméter választásról lesz szó. Majd a bemutatott szervezetek néhány korábbi, RSA-ra is vonatkozó szabványa alapján ismertetem a paraméterekkel szembeni követelmények változásait, néhány esetben a szükségességének indoklásával. Utána pár tényezőt megemlítek, melyek a jövőben hatással lehetnek az algoritmus biztonságos használatára.

A legfőbb célom, hogy elsősorban matematikai szemszögből bemutassam, hogy a szükséges biztonság elérése érdekében a paraméterek helyes megválasztására mennyi minden hatással volt az algoritmus publikálása óta. Mivel a témakör rendkívül szerteágazó, így rengeteg publikáció jelent meg róla, melyek pontos megértéséhez is nagyon mély, összetett matematikai ismeretekre van szükség. Ezért a téma feldolgozása során ezek közül néhány esetben van lehetőségem a pontos ismertetésre. Igyekszem átfogó képet mutatni, a témával kapcsolatban mélyebben érdeklődők viszont az irodalomjegyzékben megtalálhatják néhány, általam csak röviden ismertetett kutatási eredmény forrását is, ezekre a szakdolgozatomban az adott helyeken hivatkozom.

2. A nyilvános kulcsú titkosítás

A nyilvános kulcsú titkosítás előnye, hogy a két félnek nincs szüksége egy közös információ (titok) ismeretére, ehelyett a fogadó rendelkezik egy titkos kulccsal (Secret Key), melyet csak ő ismer, illetve egy nyilvános kulccsal (Public Key), amelyhez bárki hozzáférhet, ezt küldi el annak, akitől a titkosított üzenetet várja. Az előbbit sk-val, az utóbbit pk-val jelöljük.
Az üzenetküldő, a fogadó nyilvános kulcsának segítségével egy polinom időben futó véletlen algoritmussal (E) kódolja az üzenetet (a rejtjelezett üzenet c), melyet már nem biztonságos csatornán keresztül továbbíthat, majd a fogadó titkos kulcsával lehet megfejteni a D polinom időben futó determinisztikus algoritmus segítségével. A kulcspárt $\mathcal{G}$ véletlen algoritmus generálja. A lehetséges szövegek egy véges halmazát $\mathcal{M}$-el jelöljük, melynek egy elemét szeretnénk rejtjelezni.
2.1. definíció[2] (Asszimetrikus kulcsú titkosítás) Adott k biztonsági paraméter mellett a $(\mathcal{G}, D, E)$ hármasról azt mondjuk, hogy aszimmetrikus (más néven nyilvános) kulcsú titkosítás, ha az alábbi tulajdonságok teljesülnek:

1) $(pk, sk) \leftarrow \mathcal{G}(1^k)$
2) $c \leftarrow \mathrm{E}_{pk}(m)$, $m \in \mathcal{M}$
3) $m = \mathrm{D}_{sk}(c)$
4) Minden k, $\mathcal{G}$ által generált (sk, pk) és tetszőleges m $\in \mathcal{M}$ esetén fennáll, hogy $\mathrm{P}[\mathrm{D}_{sk}(\mathrm{E}_{pk}(m)) = m] = 1 - \nu(n)$, ahol ν elhanyagolható.

A titkos kulcs ismerete nélküli megfejthetetlenséget olyan matematikai algoritmus biztosítja, mely lehetővé teszi a c kódolt üzenet hatékony kiszámolását, ezzel szemben az inverz meghatározására jelenlegi ismereteink szerint nem létezik politomiális időben futó algoritmus. Éppen ezért, a biztonság érdekében mindig figyelembe kell venni az aktuális matematikai eredményeket és az informatikai eszközök fejlődését.
Fontos, hogy a kulcsok generálása valóban véletlenszerű legyen. Nem megfelelő $\mathcal{G}$ algoritmus esetén a titkos kulcs kitalálhatóvá válik, ekkor viszont a titkosítás nem biztonságos. A gyakorlatban valódi véletlen előállítása nehéz feladat, ezért ennek ellenőrzése miatt a kapott eredményeknek bizonyos kritériumoknak meg kell felelniük.

3. Szabványok

3.1. definíció (Szabványosítás:) olyan tevékenység, amely általános és ismételten alkalmazható megoldásokat ad fennálló vagy várható problémákra azzal a céllal, hogy a rendező hatás az adott feltételek között a legkedvezőbb legyen.[3]

A leggyengébb láncszem elve szerint egy rendszer annyira biztonságos, amennyire a leggyengébb eleme az. Ezért logikus szabványokat használni, melyek megadják a minimális követelményeket a rendszer minden elemére.
Ezeket a szabványokat különböző szervezetek publikálják, két fő szempont szerint csoportosítható: az elfogadottsági szintje, illetve a használatának témaköre szerint. Az első alapján megkülönböztetjük a nemzetközi, regionális és országos szervezeteket. Számos különböző témaköre lehet. Magasabb szinteken elterjedt a távközlési, elektrotechnikai és általános szabványügyi szervezet. Ezek közül néhányat összefoglaltam az alábbi táblázatban:

	Általános	Távközlési	Elektrotechznikai
Nemzetközi	ISO	ITU	IEC
Regionális	CEN	ETSI	CENELEC
Országos	MSZT		

1.táblázat

Az informatikai szabványosítás esetén ez összetettebb, mivel számos nemzetközileg elismert civil szervezet és kormányzati intézmény készít de facto szabványokat.

Közülük néhány:
ITU (International Telecommunication Union) távközlési együttműködés segítésére szakosodott nemzetközi szervezet. 1992 előtt CCITT néven volt ismert.

ETSI (European Telecommunications Standards Institute) nonprofit szabványosító testület, mely infokommunikációs szabványokat állít ki. Az EU elismeri európai szabványügyi szervezetnek.

NIST (National Institute of Standards and Technology) a szabványosítási feladatok szempontjából az ISO-nak alárendelt amerikai szervezet, mely szintén kiad informatikában alkalmazható protokollokat (így kriptográfiai algoritmusokat érintőeket is).

ANSI (American National Standards Institute) elsősorban Amerikai Egyesült Államok számára készít szabványokat. Tagja az ISO és az IEC szervezeteknek. Azonban szabványai nem csak az

USA-ban mérvadóak, a szervezethez köthető többek közt az ANSII karakterkészlet kidolgozása is, melyet számos országban alkalmaznak.

A továbbiakban bemutatom az RSA alapötletét, és a módszert publikálók kezdeti feltételeit a paraméterek helyes megválasztására, melyek közül néhány elvárásra magyarázatot adok.
Majd az IEC (International Telecommunication Union), korábbi nevén CCITT által 1988-ban kiadott X.509 szabványa, az ETSI (European Telecommunications Standards Institute) és a NIST szabványügyi szervezetek néhány RSA-ra vonatkozó szabványai alapján, bemutatom a paraméterekkel szembeni követelmények időbeni változásait, melyek az algoritmus biztonságának fenntartása érdekében történtek. Néhány esetben indoklást adok a szükségességére.

4. Rivest, Shamir és Adleman publikációja

4.1. RSA algoritmus

Az először 1977-ben publikált algoritmus a **2.1. definíció**ban használt jelölésekkel a következő:

1. A $\mathcal{G}(1^k)$ választ 2 véletlen k bitből álló prímszámot p-t és q-t, majd kiszámolja $n = pq$ értéket. Választ egy $d \in \mathbb{N}$ értéket is, melyre $1 < d < \varphi(n)$ és $lnko(d, \varphi(n)) = 1$ teljesül. Meghatározza e-t a $de \equiv 1 \ (mod\ \varphi(n))$ összefüggésből. Majd visszatér a (pk, sk) kulcspárral, ahol $pk = (e, n)$ és $sk = (d, n)$.

2. $\mathrm{E}_{pk}(m) := m^e \ (mod\ n)$

3. $\mathrm{D}_{sk}(c) := c^d \ (mod\ n)$

Ahhoz, hogy a fent leírt módszer aszimmetrikus kulcsú titkosítás legyen, teljesülnie kell, hogy:
$\mathrm{P}[\mathrm{D}_{sk}(\mathrm{E}_{pk}(m)) \equiv m \ (mod\ n)] = 1 - \nu(n)$
Ez belátható, hiszen $de \equiv 1 \ (mod\ \varphi(n)) \Rightarrow \exists k \in \mathbb{Z} : de = k\varphi(n) + 1$, ezért $\mathrm{D}_{sk}(\mathrm{E}_{pk}(m)) \equiv m^{de} \equiv m^{k\varphi(n)+1} \equiv m \ (mod\ n)$. Az utolsó lépésben az Euler-Fermat tétel lett alkalmazva.

Az n és az m üzenet csak abban az esetben nem relatív prímek, amikor $m = p$ vagy $m = q$. A $\mathrm{D}_{sk}(\mathrm{E}_{pk}(m)) \equiv m \ (mod\ n)$ ezekben az esetekben is fennáll, de ezt nem részletezem.

Az RSA titkosítás alapötlete, hogy adott p és q esetén $n = pq$ gyorsan meghatározható, ezzel szemben nem ismert olyan matematikai módszer, mellyel bármilyen n prímtényezőkre bontható lenne polinomiális időben.
Mivel $de \equiv 1 \ (mod\ \varphi(n))$ kongruencia és ebből e és n értéke ismert mindenki számára, a támadhatóság szempontjából elengedhetetlen fontosságú a következő tétel:
4.1.1. tétel[2] Minden p, q (támadó számára ismeretlen) prím és $n = pq$ esetén n prímtényezőkre bontása és a $\varphi(n)$ meghatározása ekvivalens nehézségű probléma.

Ez a tétel biztosítja, hogy nem csak az n faktorizálásának, de $\varphi(n)$ meghatározának nehézsége is védelmet nyújt, hiszen e két probléma ekvivalens nehézségű.
Fontos megemlíteni, hogy azonos méretű számok prímtényezőre bontása és a különböző nagyságrendű d és e esetén m üzenet visszanyerése nem egyenlően nehéz feladat. Ezért nem megfelelő paraméterek megválasztásával az RSA támadható.
Moduláris gyökvonásra szintén nem ismert polinomiális időben futó algoritmus, így megfelelő paraméterek mellett $m \leftarrow E_{pk}^{-1}(c)$ segítségével sem fejthető meg a titkosított üzenet.

A továbbiakban bemutatom a paraméterekkel szembeni követelményeket, annak okait, majd a következő fejezetben a változásait, melyek célja a feltörés elkerülése volt.

4.2. Kezdeti ajánlások a paraméterek megválasztására

1. A p és a q legyen decimálisan 100 számjegyből álló véletlenszerűen választott prímszám.
2. $u = (p-1)$-nek és a $v = (q-1)$-nek legyen nagy prímtényezője.
3. Nagyobb biztonság elérése érdekében ajánlott, hogy $(u-1)$-nek és $(v-1)$-nek is legyen nagy prímtényezője.
4. d-re megfelel bármely $max(p,q)$-nál nagyobb prím választása.
5. Legyen $e > log_2(n)$.

Paraméter követelmények okai:
1. Elegendően nagy prímszámokat kell választani, hogy a prímfaktorizáció nehéz legyen. Az $n = 200$ döntést azzal indokolták, hogy az akkor ismert leggyorsabbnak számító módszerrel is $3.8 \cdot 10^9$ évbe telne p és q visszafejtése. Ez elképzelhetetlenül sok időnek tűnik, viszont amiatt, mert más módszerek is születhetnek faktorizálásra és a számítógépek teljesítménye idővel növekszik, indokolt lehet a biztonság érdekében. Adott esetben, amikor a titkosítás sebessége számít, akkor ennél kisebb n is választható, de ez csak mérsékeltebb biztonságot nyújt a publikáció szerint. Erre a későbbiekben még visszatérek az 5.4 fejezetben.

2. A következő algoritmus lehetővé teszi olyan n-ek prímtényezőkre bontását, melyekre $(p-1)$ vagy $(q-1)$ kizárólag kis prímtényezőkkel rendelkezik:
4.2.1. algoritmus (Pollard $(p-1)$ faktorizáció)

1. Legyen $a = 2$ és $r \in \mathbb{N}$, melyről azt sejtjük, hogy $(p-1)|r$.
2. Számoljuk ki $b \equiv a^r - 1 \ (mod\ n)$-t. Ha $b = 0$, akkor $a := a + 1$, majd számoljuk újra b-t. Legyen $p = lnko(b, n)$.
3. Ha $p \neq 1$ és $p|n$ teljesül, akkor megtaláltuk n prímtényezőre bontását. Ha nem teljesül, akkor válasszunk újabb r-et, majd térjünk vissza a 2. ponthoz.

Az algoritmus működését a következő tétel szolgáltatja:
4.2.2. tétel Legyen p és q prím, $n = pq$. Ha $(p-1)|r$, $1 < a < (p-1)$ és $n \not| \ (a^r - 1)$, akkor $p = lnko(a^r - 1, n)$.
Bizonyítás.[5]
$(p-1)|r \Rightarrow \exists j \in \mathbb{Z} : r = (p-1)j$, ezért $a^r = a^{(p-1)j} = (a^{(p-1)})^j \equiv 1^j \equiv 1 \ (mod\ p)$ az Euler-Fermat tételt alkalmazva. A kongruencia definíciójából következik, hogy $p|(a^r - 1)$ és mivel $p|n$, ezért $p|lnko(a^r - 1, n)$. Az n-nek csak p és q a prímtényezői, így $n \not| \ (a^r - 1)$ esetén $p = lnko(a^r - 1, n)$.

$(p-1)$ természetesen nem ismert, ezért $(p-1)|r$ biztosítása csak abban a speciális esetben lehetséges, amikor $(p-1)$ vagy $(q-1)$ összes prímtényezője "kicsi", ekkor r például a legnagyobb prímtényező faktoriálisaként teljesítheti a feltételt. Erre a későbbiekben még kitérek.

4. d-re teljesülnie kell $lnko(d, \varphi(n))$-nek, a publikáció szerint erre egy kézenfekvő gyors megoldás, ha bármely $max(p, q)$ -nál nagyobb prímet választjuk. Ez megfelelő, mert $\varphi(n) = (p-1)(q-1)$, tehát $\varphi(n)$ prímtényezős felbontásában valóban nem szerepelhet $max(p, q)$-nál nagyobb prím, ezért az így választott d relatív prím lesz vele. Ezáltal $d > \sqrt{n}$-nél is teljesül, aminek a szerepére a később szintén kitérek majd. Fontos, hogy d kellően nagy halmazból legyen választva, hogy a brute force-támadás elkerülhető legyen.

5. Célszerű lenne, ha a titkosítás során minden m üzenet átesne a "wrap-around"-on (mod n). Ez $m = 0$ és $m = 1$ esetén lehetetlen, viszont $m > 1$-re a feltétel ezt teljesíti, hiszen ekkor $\mathrm{E}_{pk}(m) = m^e \geq 2^e > 2^{log_2(n)} = n \Rightarrow m^e > n$. Ha $e > log_2(n)$ nem áll fenn, új d-t kell választani. Ez azt is jelenti, hogy az RSA titkosítás bináris üzenetek titkosítására alkalmatlan (ekkor $c = m$), az üzenetet $m \in [0, n-1]$ intervallumból kell megválasztani.

Ajánlás megfelelő kulcsgenerálására:

Annak biztosítása érdekében, hogy $p-1$ mindenképp rendelkezzen nagy prímtényezővel ezt a módszert javasolják:

Először is válasszunk egy kellően nagy u prímet, majd $i = 2, 4, ..$-re ellenőrízzük $iu + 1$ számról, hogy prím-e. Ha igen, akkor készen vagyunk, hiszen ekkor $p - 1 = ui$ rendelkezik u-val, mint nagy prímtényező.

5. RSA követelményeinek a változásai

5.1. Az 1988-as X.509 szabvány

Az X.509 (1988) szabvány p és q követelményei$_{[S1]}$:

a) véletlenszerűen kell kiválasztani őket;

b) nagynak kell lenniük;

c) prímszámnak kell lenniük;

d) $|p-q|$-nek nagynak kell lennie;

e) $(p+1)$-nek nagy prímtényezővel kell rendelkeznie;

f) $(q+1)$-nek nagy prímtényezővel kell rendelkeznie;

g) $(p-1)$-nek nagy prímtényezővel kell rendelkeznie, ez legyen r;

h) $(q-1)$-nek nagy prímtényezővel kell rendelkeznie, ez legyen s;

i) $(r-1)$-nek nagy prímtényezővel kell rendelkeznie;

j) $(s-1)$-nek nagy prímtényezővel kell rendelkeznie;

A változást az RSA kezdeti publikációjához képest a b), d), e) és f) pontok adják:

b) Az n hosszát 512 bit-ben határozza meg, ez decimálisan a korábbi 200 számjegyes ajánláshoz képest kisebb értéket jelent, hiszen $log_{10}(2^{512})$=154.13. Ennek lehetséges okáról az 5.4 fejezetben írok.

d) Fermat 1643-ban kidolgozott egy módszert $n = pq$ faktorizálására, aminek alapja, hogy ilyen n-ekhez bijektíven hozzárendelhető egy x, y számpár, melyre $n = x^2 - y^2$. Ezért $y^2 = x^2 - n$, így $x = \lceil\sqrt{n}\rceil, \lceil\sqrt{n}\rceil + 1, ..$ próbálkozásokkal elegendő azt vizsgálni, hogy $x^2 - n$ négyzetszám-e, majd $n = x^2 - y^2 = (x-y)(y+x)$ azonosság segítségével n könnyedén prímtényezőkre bontható. Az évek során több hatékonyságot javító módosítás is született, az alábbiakban az egyik ilyen módszert mutatom be:

Legyen $4n = x^2 - y^2$, ekkor $x = \lceil 2\sqrt{n}\rceil, \lceil 2\sqrt{n}\rceil + 1, ..$-re kell vizsgálni, hogy $x^2 - 4n$ négyzetszám-e. Abban az esetben amikor teljesül, az n prímtényezőit a $p = \frac{1}{2}(x+y)$, $q = \frac{1}{2}(x-y)$ összefüggésekkel megkapjuk. A következő lemmával igazolható, hogy ez a módszer nagyon hatékony nagy számok esetén is, ha a két prím elegendően közel van egymáshoz:

5.1.1. lemma$_{[8]}$ Legyen $n = pq$, $p > q$ és $\Delta = p - q$, ekkor $0 < p + q - 2\sqrt{n} < \frac{\Delta^2}{4\sqrt{n}}$.

Bizonyítás.$_{[8]}$ $\Delta^2 = (p-q)^2 = (p+q)^2 - 4n = (p+q-2\sqrt{n})(p+q+2\sqrt{n})$. Mivel $\Delta^2 > 0$ és $p + q + 2\sqrt{n} > 0$, ezért $p + q - 2\sqrt{n} > 0$. Ebből következik, hogy $p + q - 2\sqrt{n} = \frac{\Delta^2}{p+q+2\sqrt{n}}$. Tudjuk, hogy $p + q > 2\sqrt{n}$, így $0 < p + q - 2\sqrt{n} < \frac{\Delta^2}{4\sqrt{n}}$, ami a lemma állítása volt.

Az algoritmus során $x - \lceil 2\sqrt{n}\rceil \approx x - 2\sqrt{n}$-szer próbálkozunk. Végül ha megtaláltuk p-t és q-t, akkor $p = \frac{1}{2}(x+y)$ és $q = \frac{1}{2}(x-y)$, tehát $p + q = x$. Ebből és a fent leírt lemmából

következik, hogy $x - 2\sqrt{n} < \frac{\Delta^2}{4\sqrt{n}}$, tehát legfeljebb $\frac{(p-q)^2}{4\sqrt{n}}$ lépésben eredményre jutunk. Ezért nem érdemes egymáshoz túl közel eső prímeket megválasztani.

e) és f) Csak kis prímtényezővel rendelkező $(p+1)$ vagy $(q+1)$ esetén, Williams 1982-ben publikált $(p+1)$ módszere alkalmazható n faktorizálására, a módszer bemutatásához szükséges néhány matematikai fogalom definiálása is a 6. fejezetben található alapismereteken túl:

5.1.2. definíció[6] (Lucas sorozat) Legyen $P, Q \in \mathbb{Z}$ és α, β az $x^2 - Px + Q$ polinom gyökei. Ekkor Lucas sorozatok a következők:

$$U_n(P, Q) = \frac{(\alpha^n - \beta^n)}{(\alpha - \beta)},$$
$$V_n(P, Q) = \alpha^n + \beta^n.$$

5.1.3. tétel[6] (Lehmer) Ha p egy páratlan prím, $p \nmid Q$ és $\epsilon = \left(\frac{P^2-4Q}{p}\right)$ Legendre szimbólum, akkor:

$$V_{(p-\epsilon)m}(P, Q) \equiv 2Q^{m(1-\epsilon)/2} \ (mod\ p)$$

Williams módszerének az alapját Lehmer 1930-ban publikált tétele adja, ugyanis $P \in \mathbb{Z}$ és $Q = 1$ megválasztásával abban az esetben, ha $\epsilon = -1$ a következő alakját kapjuk:

$$V_{(p+1)m}(P, 1) \equiv 2 \ (mod\ p)$$
$$p \mid V_{(p+1)m}(P, 1) - 2$$
$$lnko(\, V_{(p+1)m}(P, 1) - 2, n) = p, \text{ ha } n \nmid V_{(p+1)m}(P, 1) - 2$$

Tehát Pollard módszeréhez hasonlóan, ha találunk egy r számot melyre teljesül, hogy $(p+1)|r$ vagy $(q+1)|r$, akkor a Lucas sorozat r-edik tagjának kiszámolásával meghatározhatjuk a prímtényezőt. (Gyakorlatban annak az esélye, hogy $V_r(P, 1) - 2$ véletlenül n többszöröse legyen, elhanyagolhatóan kicsi.[6]) Sajnos $\left(\frac{P^2-4Q}{p}\right)$-t nem ismerjük, ezért a számolást több P-vel is érdemes elvégezni, hogy véletlenszerűen $\epsilon = -1$ értéket kapjunk. Lehmer tételéből látszik, hogy $\epsilon = 1$ esetén analóg módon egy Pollardénál bonyolultabb $(p-1)$ módszerhez jutunk. Ha megsejtünk egy r-t, ami teljesíti a kritériumot, akkor a $V_r(P, 1)$ kiszámolását rekurzív módon a következő tétel segítségével tehetjük meg:

5.1.4. tétel[6] Legyen $V_k(P)$ a $Q = 1$ és $P \neq \pm 2$ paraméterekhez tartozó Lucas sorozat tagjai. Ekkor:

$$V_{2f-1}(P) \equiv V_f(P)\, V_{f-1}(P) - P,$$
$$V_{2f}(P) \equiv V_f^2(P) - 2,$$
$$V_{2f+1}(P) \equiv PV_f^2(P) - V_f(P)\, V_{f-1}(P) - P \ (mod\ n).$$

Látható, hogy Pollard és Williams módszerének is az a hátránya, hogy kell találni egy megfelelő r-et. Mint korábban írtam, erre $p \pm 1$ legnagyobb prímtényezőjének a faktoriálisa megoldást

adhat, azonban Stirling-formulával becsülhető, hogy relatív kicsi értékek esetén is óriási számról van szó. Ennél létezik hatékonyabb módszer, mivel a faktoriális legkisebb prímtényezőinek feleslegesen nagy a hatványkitevője. Ahhoz, hogy r pl. $p-1$ többszöröse legyen elegendő $p-1$ legnagyobb prímtényezőjéig az összes prímszámot $\lceil log_{p_j}(n)\rceil$-edig hatványon összeszorozni. Pollard módszere esetén ez $2^{(\prod p_j^{\lceil log_{p_j}(n)\rceil})}$ moduláris hatványozás kiszámolását jelenti. Kis átalakítással $(((2^{\lceil log_{p_1}(n)\rceil})^{\lceil log_{p_2}(n)\rceil})^{\cdot\cdot})\ (mod\ n)$, ennek egy implementációja a függelékben megtalálható. Akár 2^{80} nagyságrendű n-ek esetén is gyorsan eredményre lehet jutni, viszont a szabványban megkövetelt 512bit-es n-ek esetén egyszer sem jártam sikerrel az otthoni számítógépemmel.
Összességében elmondható, hogy biztos r meghatározásához mindenképp szükséges $p \pm 1$ tippelt legnagyobb prímtényezőjéig az összes prímszám ismerete, és az ezekkel történő szükséges műveletek elvégzése, mivel nem tudhatjuk mely prímek szerepelnek a prímtényezős felbontásban és melyek nem.

5.1.5. definíció$_{[S2]}$ p prímet erős prímnek nevezzük, ha $(p-1)$-nek van nagy prímtényezője, ez legyen u. Emellett $(u-1)$ és $(p+1)$ is rendelkezik nagy prímtényezővel.

Az X.509 (1988) szabvány ajánlása e választására:$_{[S1]}$:
A kezdeti publikációhoz képest változás a $\mathcal{G}(1^k)$ kulcsgeneráló algoritmusban, hogy nem d, hanem e értéke van megválasztva, és a d értékét az e-ből kell kiszámolni. Az e-vel szembeni követelmény továbbra is az, hogy $e > log_2(n)$. Amellett hogy ezt teljesíti, a tárolási kapacitást és a számítási sebességet is figyelembe kell venni. Ezért mivel e úgyis publikus, a megosztásra kerülő nyilvános kulcs hosszának a csökkentése érdekében egy fix értéket ajánl: az F_4 Fermat-számot. Ennek a használata azért kedvező, mert $m^{2^k}(mod\ n)$ kiszámolása összesen k moduláris szorzást jelent, mivel $m^{2^k} \equiv m^{2\cdot 2\cdot 2\cdot\cdot} \equiv (((m^2)^2)^2)..\ (mod\ n)$. Ez lecsökkenti a kódoláshoz szükséges időt. Ráadásul F_4 prímszám, így a kulcsok generálásakor szükséges $lnko(e, \varphi(n)) = 1$ feltétel helyett elegendő $e \not| p-1$ és $e \not| q-1$ vizsgálata.

5.2. Követelmények változásai néhány korábbi ETSI szabvány bemutatásával

Az RSA nem csak titkosításra alkalmas, hanem digitális aláírások készítésére is, amivel biztonságosan igazolni lehet, hogy egy dokumentumot valóban a megnevezett készítője írta. Ez egy csatolt bitsorozatot jelent, ami titkosítottan tartalmazza a dokumentum egy ellenőrző összegét, a készítőjének nevét vagy azonosítóját, és egy dátumot. Ekkor az algoritmusnak lényegében az inverzét kell végrehajtani, hiszen a cél, hogy az aláírást csakis egy titkos kulcs birtokában lehessen generálni, az így generált bitsorozatról pedig a nyilvános kulccsal bárki ellenőrizni tudja, hogy valóban hiteles. Ez az RSA esetén a következőképpen néz ki[21]:
-A kulcsgeneráló algoritmus azonos a korábbiakkal.
-Bellare és Rogaway 1996-os javaslata alapján először egy nyilvános, egyirányú függvénnyel ($H : \{0,1\}^* \to \mathbb{Z}_n^*$) lenyomatot kell készíteni.
-Majd ki kell számítani az $s \equiv H(m)^d \ (mod\ n)$ aláírást.
-Ezután ellenőrzéskor az (m, s) alapján ki kell számolni $H(m)$ lenyomatot és $s^e \ (mod\ n)$ értéket. Abban az esetben, ha $H(m) \equiv s^e \ (mod\ n)$ fennáll, akkor az aláírás érvényes, ha nem, akkor az aláírás érvénytelen.

A digitális aláírás módszerét számos területen használják, így erre vonatkozóan is számos RSA-val kapcsolatos szabvány jelent meg az idők során, ebben a fejezetben az ETSI, digitális aláírásokra vonatkozó szabványsorozata alapján mutatom be az RSA paraméterkövetelményeit.

5.2.1. ETSI 2003-03

Paraméter követelmények[S2]:
1) $n = pq$ legyen legalább MinModLen=1020 bit hosszú.
2) p és q legyen közel egyenlő hosszú prím: $0.5 < |log_2(p) - log_2(q)| < 30$.
3) Az elfogadott kulcsgeneráló algoritmus használata.

Paraméter követelmények magyarázata:
1) A MinModLen vagy más néven ModLen a szabvány szerint 1020bit, melyet aláíráshoz 2001.01.01 dátummal fogadtak el, a publikáció szerint az 1999-ben kiadott ISO/IEC 14888-3 digitális aláírások függeléke alapján. Ennek érvényessége 2005.12.31-ig tart.

2) A feltétel célja, hogy p és q távolsága ne legyen se túl kicsi, se túl nagy.

3) Kulcsgeneráló algoritmus[S2]:
A feltételnek megfelelő k bit hosszú n-hez p és q előállítására az ajánlás a következő[S2]:

1. Válasszunk egy véletlenszerű $p \in [2^{k/2-15}, 2^{k/2+15}]$ prímszámot.

2. Válasszunk egy véletlenszerű $q \in [2^{k-1}/p, 2^k/p]$ prímszámot.

3. Ellenőrízzük, hogy teljesül-e a $0.5 < |log_2(p) - log_2(q)| < 30$ feltétel, ha nem, akkor kezdjük az első lépéstől.

4. Legyen $n = pq$.

A véletlen választást EntropyBits paraméterrel trueran vagy SeedLen paraméterrel pseuran módszerrel kell végrehajtani. (Melyekről később részletesen írok.)
Az e és d megválasztása:

1. e-t a $[3, n-1]$ halmazból kell választani úgy, hogy teljesítse a $lnko(e, lkkt(p-1, q-1)) = 1$ feltételt.

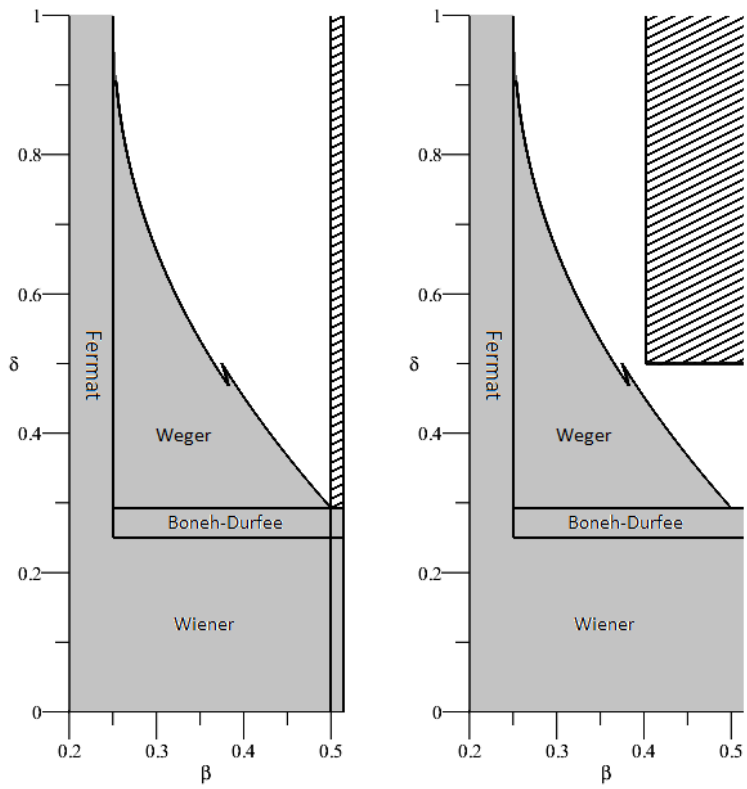
2. d-t az $ed \equiv 1 \ (mod \ lkkt(p-1, q-1))$ egyenletből kell kiszámolni.

Jelentős változás, hogy $\varphi(n)$ helyett $lkkt(p-1, q-1)$ modulust kell választani, erre a NIST szabványról szóló 5.3.1-es fejezetben részletesebben kitérek.

Választható az a módszer is, hogy először $d \in [3, n-1]$ privát kulcs van meghatározva. Majd ebből lesz kiszámolva e. Konzervatív esetben viszont, figyelembe kell venni a kis d értékkel kapcsolatos legújabb publikációkat és ez alapján kell választani egy minimum $\sqrt{n}$ nagyságú intervallumot.
Wiener 1990-ben bebizonyította, hogy a privát kulcs visszanyerhető, ha $d < n^{\frac{1}{4}}$, majd 9 évre rá Boneh és Durfee még erősebb feltételt adott rá: $d < n^{1-\frac{\sqrt{2}}{2}}$. 2002-ben Weger kiterjesztette a két publikációt függővé téve p és q távolságától. [8]Ha $\Delta = |p-q|$, $\Delta = n^{\beta}$ ($\beta \in [\frac{1}{4}, \frac{1}{2}]$) és $d = n^{\delta}$, akkor d értéke hatékonyan visszanyerhető a következő esetekben:

$$2 - 4\beta < \delta < 1 - \sqrt{2\beta - \frac{1}{2}} \text{ vagy}$$
$$\delta < \frac{1}{6}(4\beta + 5) - \frac{1}{3}\sqrt{(4\beta + 5)(4\beta - 1))}$$

Bár az utóbbi eredmény ETSI szabványt alig érinti, hiszen ajánlás szerint kiszámolható, hogy $\beta > 0.4985$, viszont Weger munkája és további javaslatai az akkor érvényben lévő amerikai ANSI X9.31 szabványhoz képest nem elhanyagolható, hiszen jóval enyhébb feltételeket szab a generált prímek távolságára. Az alábbi ábra összefoglalja az akkor ismert kis d és kis Δ támadásainak módszereit β és δ függvényében, bal oldalon az ETSI, jobb oldalon az ANSI paraméterekkel szembeni követelményei láthatóak (sraffozott terület)[8]:



1. ábra.

Ennek ellenére az ETSI szabvány megengedi a $d \in [3, n-1]$ intervallumból történő véletlen választást is, mivel annak a valószínűsége, hogy így feltörhető értéket kapunk elenyészően kicsi: $P[d < (2^{1020})^{0.293}, d \in [3, 2^{1020}]] \approx \frac{2^{298.9}}{2^{1020}} \approx 2^{-721.1}$.

Másik jelentős változás, hogy a szabvány nem követeli meg az erős prímek használatát, mert bebizonyosodott, hogy kicsi az esélye ekkora nagyságrendű számok esetén, hogy egy véletlen prím nem erős:

5.2.1.1. definíció[7] $k \in \mathbb{N}$ B-sima, ha k összes prímfaktora kisebb, mint B.

5.2.1.2. definíció Azt a függvényt, mely megoldása az $u\rho'(u)+\rho(u-1) = 0$ differenciálegyenletnek, $0 \leq u \leq 1 : \rho(u) = 1$ kezdeti értékekkel Dickman-de Bruijn-féle $\rho(u)$ függvénynek nevezzük.

Annak a valószínűsége, hogy egy $k \in \mathbb{N}$ szám $k^{\frac{1}{u}}$ sima legyen becsülhető $\rho(u)$-val.[7] A függvény egy közelítése $\rho(u) \approx u^{-u}$. Később több publikáció is megjelent, ami alapján egy szám B-sima tulajdonságának esélyét lehet becsülni, ezek jobb közelítést adtak, de nagy számok esetén nagyságrendileg az előbb említett becslést adják.

Korábban rámutattam, hogy Pollard és Williams módszere a gyakorlatban mennyire számításigényes. Az alábbi táblázat viszont magyarázza, miért nem jártam sikerrel 512 bit hosszú n-ek esetén. Illetve szemlélteti, hogy a 2003-as szabvány 1020 bit-es n mudulusa esetén, szinte lehetetlen eredményre jutni még egy nagyobb teljesítményű számítógéppel is.

$log_2(p)$	$log_2(B)$	$P[(p-1)\ B$-sima] becslés
256	20	$\approx 6.7 \cdot 10^{-15}$
510	50	$\approx 5.1 \cdot 10^{-11}$
510	100	$\approx 2.4 \cdot 10^{-4}$
510	150	$\approx 1.6 \cdot 10^{-2}$

2.táblázat

A prímszámtétellel becsülhető, hogy B-ig körülbelül hány darab prímszám van. Ez $B = 2^{100}$ esetén $\frac{2^{100}}{ln(2^{100})} \approx 1.8 \cdot 10^{28}$, ennyi ciklus elvégzése után is csupán $1 - (1 - 2.4 \cdot 10^{-4})^4 \approx 9.6 \cdot 10^{-4}$ eséllyel lenne sikeres Pollard vagy Williams faktorizációja.

Véletlen generátálása trueran-nal:

A módszer valódi véletlen generálását jelenti, mely egy fizikai zajforráson, és a zaj utókezelésén alapul. A zaj véletlenszerűségét statisztikai tesztekkel kell ellenőrízni.

Ilyen forrás lehet például egy kvantumfolyamat, amit speciális eszközökkel lehet mérni, majd feldolgozni erre szakosodott számítógépekkel, ezzel hatékonyan lehet előállítani valódi véletlent. Hátránya, hogy ehhez olyan eszköz szükséges, amihez nem mindenki fér hozzá, viszont vannak más források is, ilyenek a számítógép használatából eredő folyamatok, például egérkattintás, billentyűzet leütés, melyek kombinációiból véletlen nyerhető ki. Viszont ezek nem teljesen véletlenszerűek, ezért utókezelésre van szükség, illetve a véletlen statisztikai tesztelésére. A következő átalakítások használhatóak:

-A bit stringek titkosítása, mert néhány titkosítási módszer képes elrejteni az esetleges ismétlődéseket, anélkül, hogy a bithossz változna.

-Más adatok (számlálók vagy egyedi azonosító) értékének hozzáadása.

Véletlenszerű bit stingeknek minden esetben egyedi forrásból kell származniuk, például egy matematikai vagy fizikai állandó számjegyei akkor sem használhatóak, ha kimutatható a véletlenszerűségük, hiszen nem tekinthetőek titkosnak. A generált kulcs kitalálásának a nehézsége minimum egyenértékű kell hogy legyen, egy EntropyBits hosszú igazi véletlen érték kitalálásával, melyet az entrópia becslésével lehet biztosítani. Fontos, hogy az entrópiát nem a generált kulcsból, hanem forrás valószínűség-eloszlásából kell kiszámolni.

Ilyen elven véletlent előállító algoritmusokat nevezzük TRNGk-nek, vagy más néven NRNGk-nek.

Véletlen generálása pseuran-nal:

A módszer álvéletlen számok generálását jelenti egy determinisztikus algoritmus segítségével, melynek bemeneti paraméterét seed-nek nevezzük, amit egy valódi véletlen forrás ad, hosszát pedig SeedLen paraméter határozza meg. Az algoritmus egy k bit hosszú seed esetén $l \gg k$ bit

hosszú számot állít elő. A kimenet nem véletlenszerű, a cél, hogy egy rövid valódi véletlen seed-et oly módon bővítsen nagyobb méretűvé, hogy az ne legyen könnyen megkülönböztethető egy valódi egyenletes eloszlásból származó véletlen számtól. Az ilyen algoritmusokat PRNGk-nek nevezzük. Önmagában nem használhatóak kulcsok generálására, ehhez további kritériumoknak is meg kell felelniük, hogy kriptográfiailag biztonságosnak tekinthessük (CS), mely fogalmat Blum és Micali az 1984-es publikációjában így határozta meg:

5.2.1.3. definíció[9] Azt mondjuk, hogy egy álvéletlen generátor sikeresen teljesíti a next-bit tesztet, ha nem létezik olyan polinomiális időben futó algoritmus, ami a kimenet első l bitjének ismeretében $\frac{1}{2}$-nél lényegesen nagyobb valószínűséggel meg tudja jósolni az $(l+1)$-edik bit értékét.

5.2.1.4. definíció[9] Egy álvéletlen generátort, ami sikeresen teljesíti a next-bit tesztet, kriptográfiailag biztonságos álvéletlen generátornak (CSPRNG) nevezzük.

Az alapját az adta, hogy bebizonyították, hogy [9]egy álvéletlen generátor akkor és csak akkor CSPRNG, ha egyetlen polinom időben futó algoritmus sem képes $\frac{1}{2}$-nél nagyobb valószínűséggel különbséget tenni a generátor kimenete és a valóban véletlen, azonos hosszúságú értékek között.

Fontos megjegyezni, hogy a megfelelő védelem érdekében a paraméterek megválasztásakor nem elegendő a kritériumok betartása, és önmagában az sem, ha véletlenszerűen választották. Ezért minden esetben ajánlott a paraméterek megválasztásához hiteles véletlen generátorokat használni. A módszert pedig statisztikailag elemezni kell, ennek jelentőssége később az 5.2.3 fejezetben be is igazolódik. A hiteles véletlen generáló algoritmusok közül néhányat az ETSI feldolgozott szabványai, vagy a NIST esetén a 800-90A szabványai sorolják fel.

5.2.2. ETSI 2005-2011

Az ETSI a digitális aláírásokról szóló 2005$_{[S3]}$, 2007$_{[S4]}$ és 2011$_{[S5]}$-es szabványában a 2003$_{[S2]}$-as szabványhoz képest nagyon kevés változást eszközölt a paraméterekre vonatkozóan. Mindössze a biztonságosnak tekinthető modulus méreten változtattak és a p, q paraméterek távolságával kapcsolatos elváráson. Így az 5.2.1. fejezetben található 2) paraméter-követelmény $0.1 < |log_2(p) - log_2(q)| < 30$-ra módosult. A legalább 3 évig érvényes aláírás esetén megkövetelt $MinModLen$ pedig ≥ 1536-ra, illetve ≥ 2048-ra módosult, a modulus hosszakkal kapcsolatos változásokat az 5.4-es fejezetben látványosabban összefoglalom, ezért itt nem térek ki rá bővebben.

A p, q-val szembeni követelmény elsődleges célja a korábban leírtak szerint, hogy $\Delta = |p - q|$ ne legyen túl kicsi. A 0.5-ről 0.1-re történő változtatás látszólagosan enyhítés csak, hiszen eközben a modulus hossza, így a kulcsok hossza is változott. Emiatt p és q még távolabb került egymástól: Legyen $2^{x_n} = n = pq = 2^{x_p} \cdot 2^{x_q}$ és $A < log_2(p) - log_2(q)$, ahol $p > q$. Ekkor:

$2^A < 2^{log_2(\frac{p}{q})} = \frac{p}{q}$, tehát $q \cdot 2^A < p$. Δ akkor veszi fel a minimum értékét, ha $q \cdot 2^A = p$, így $2^{x_q} \cdot 2^A = 2^{x_p}$ (*). Az $n = pq$ miatt $2^A \cdot 2^{x_q} \cdot 2^{x_q} = 2^{x_n}$, tehát $A + 2x_q = x_n$, $x_q = \frac{x_n - A}{2}$. Ebből és a (*)-ból következik, hogy $x_p = \frac{x_n + A}{2}$.
$x_n \approx MinModLen$, $\Delta_{min} \approx 2^{\frac{MinModLen+A}{2}} - 2^{\frac{MinModLen-A}{2}}$
Különböző A és $MinModLen$ választások esetén az $A < log_2(p) - log_2(q)$-ból következtethető Δ:
$MinModLen = 1024, A = 0.5$: $\Delta > 2^{510.48}$
$MinModLen = 1536, A = 0.1$: $\Delta > 2^{764.15}$
$MinModLen = 2048, A = 0.1$: $\Delta > 2^{1020.15}$
Ez jól mutatja, hogy a modulus nagyságának a változása miatt, a 0.5-ről történő 0.1-re módosítás csak látszólagos enyhítés, valójában p és q távolabb kerültek egymástól. Ezért n-nek, a Fermat módszerrel történő faktorizálása, a korábbinál is műveletigényesebb feladat lenne.

5.2.3. ETSI 2014

A korábbi szabványhoz képest három jelentős változás is van a 2014[S6]-es dokumentumban a modulus méretén kívül:
1) $2^{16} < e < 2^{256}$
2) A korábbi p, q-val szembeni $0.1 < |log_2(p) - log_2(q)| < 30$ követelmény helyett a következő is választható:

$$\sqrt{2} \cdot 2^{nBits/2-1} \leq p, q \leq 2^{nBits/2} - 1 \text{ és}$$
$$|p - q| > 2^{nBits/2-100}, \text{ (ahol } nBits \text{ az n bithossza).}$$

3) Fontos változás, hogy a p és q véletlen választására nem használható a Dual_EC_DRBG véletlenszám-generáló algoritmus. Az NSA által 2006-ban publikált, NIST szabványba is bekerült CSRNG algoritmusról kiderült, hogy nem egyenletes eloszlással állít elő véletlen számokat, így *backdoor*-ként is használható, ezért már nem tekinthető biztonságosnak.[15]

A paraméterek véletlen választásának az eljárása, az első feltétel választása esetén megegyezik a 2003-as szabványnál leírtakkal. A másik követelmény esetén viszont a FIPS 186-4 publikációban leírtak szerint lehet p-t és q-t választani[S25]:
Egy *nBits* és egy e értéket vár bemenetnek, és $(status, p, q)$ kimeneteket ad. A *status* jelzi, hogy az eljárás sikeresen eredményt adott-e vagy hiba történt. Az algoritmus:

(1) Ha *nBits* nem 2048 vagy 3072, akkor kilép a következővel: (FAILURE, 0, 0).

(2) Ha $(e \leq 2^{16})$ vagy $(e \geq 2^{256})$ vagy (e páros), akkor kilép (FAILURE, 0, 0).

(3) Legyen a *security_strength* érték a 3. táblázatnak megfelelően választott.

(4) A p generálása:

(4.1) $i = 0$.

(4.2) $nBits/2$ hosszú p értéket állítsa elő egy RBG algoritmussal *security_strength*-nek megfelelően.

(4.3) Ha (p páros), akkor $p = p + 1$.

(4.4) Ha ($p < (\sqrt{2})(2^{(nBits/2)-1})$, akkor lépjen a 4.2-re.

(4.5) Ha ($lkkt(p-1, e) = 1$), akkor

(4.5.1) Tesztelje p-t Miller-Rabin prímteszttel.

(4.5.2) Ha p a prímteszt szerint prím, akkor lépjen az 5.-re.

(4.6) $i = i + 1$.

(4.7) Ha ($i \geq 5(nBits/2)$), akkor lépjen ki a (FAILURE, 0, 0) értékekkel. Egyébként lépjen a 4.2-re.

(5) A q generálása:

(5.1) $i = 0$.

(5.2) $nBits/2$ hosszú q értéket állítsa elő egy RBG algoritmussal *security_strength*-nek megfelelően.

(5.3) Ha (q páros), akkor $q = q + 1$.

(5.4) Ha ($|p - q| \leq 2^{(nBits/2)-1}$), akkor lépjen az 5.2-re.

(5.5) Ha ($q < (\sqrt{2})(2^{(nBits/2)-1}$), akkor lépjen a 5.2-re.

(5.5.1) Tesztelje q-t Miller-Rabin prímteszttel.

(5.5.2) Ha q a prímteszt szerint prím, akkor kilép (SUCCESS, p , q).

(5.6) $i = i + 1$

(5.7) Ha ($i \geq 5(nBits/2)$), akkor lépjen ki a (FAILURE, 0, 0) értékekkel. Egyébként lépjen a 5.2-re.

5.2.4. ETSI 2017-től

2014-et követően az ETSI az Electronic Signatures and Infrastructures (ESI); Cryptographic Suites sorozatban további 3 szabványt tett közzé: 2017-ben$_{[S7]}$, 2018-ban$_{[S8]}$ és 2019-ben$_{[S9]}$ a digitális aláírásra vonatkozóan. Ezekben a publikációkban a SOG-IS$_{[S10][S11][S12]}$ szerint meghatározottakat követelik meg az RSA paraméter választásaira. Ezek a követelmények a modulus méretén felül, lényegében megfelelnek a NIST legújabb szabványainak, melyeket a következő fejezetben ismertetek.

5.3. NIST szabvány bemutatása

A 2000-es évek elején a NIST megkezdte a kriptográfiai algoritmusok így az RSA paremétereivel és biztonságos használatával szembeni követelményeinek kidolgozását. Az SP 800-57 Part1 dokumentumok tartalmazzák az intézmény saját ajánlásait a kulcsok kezelésére, az elsőt 2005. augusztusában publikálták, majd később több átdolgozását is kiadták. 2009-től kulcsok létrehozásához az SP 800-56B publikációkat tették közzé, melyek elsősorban az ANSI X.9.31 és a PKCS#1 szabványokra támaszkodnak és a digitális aláírásokkal szembeni irányelveit a NIST a FIPS 186-ban határozza meg. Az újabb elvárásokra való áttérésre az SP 800-131A-ban adnak útmutatást.

5.3.1. Követelmények 2000-től

A legelső RSA-ra vonatkozó elvárást a FIPS 186-2$_{[S23]}$-ben találtam, amit 2000. január 27.-én adtak ki, azonban ekkor még csak a modulus elfogadható hosszát határozták meg, illetve az X9.31 szabványban leírtakat követelik meg[i]. Ezek alapján:
1) $nLen \geq 1024$, $pLen, qLen \approx \frac{nLen}{2}$ (ahol $nLen$, $pLen$, $qLen$ az n, p és a q bithossza)
2) $p, q \in [\sqrt{2} \cdot 2^{nLen/2-1}, 2^{nLen/2} - 1]$
3) p és a q az első 100bitjének legalább egyikében térjen el egymástól
4) $\frac{p}{q}$-nak ne legyen a közelében kis számokból álló $\frac{a}{b}$ racionális szám
5) $p - q$ rendelkezzen nagy prímfaktorral
6) $lnko(p-1, q-1)$ legyen kicsi
7) $p^-, p^+, q^-, q^+ \in [2^{100}, 2^{120}]$, ahol p^- a $p-1$, a p^+ a $p+1$ legnagyobb prímfaktora (q^-, q^+ analóg módon q-ra érvényes)

Magyarázat néhány követelményekre:
2) Ha $p \neq q$ a megadott intervallumból van választva, akkor bármely véletlen választás esetén $nLen$ bithosszú pq-t kapható, hiszen $nLen - 1 = log_2(\sqrt{2} \cdot 2^{nLen/2-1})^2 < log_2(pq) < log_2(2^{nLen/2} - 1)^2 < log_2(2^{nLen/2})^2 = nLen$. Emellett p és q is pontosan $nLen/2$ bit hosszú, ami sokkal szigorúbb, mint ami az ETSI követelménye volt 2003-tól 2014-ig, ahol a kulcsok bithosszának eltérését maximum 30 bitnek határozták meg. A nagyságrendben megegyező bithosszú kulcsok választásának a lehetséges oka, hogy megjelentek speciális típusú faktorizációs algoritmusok, melyek számítási igénye a n kisebb prímtényezőjétől függnek[ii].

3) A $|p - q|$-nek a támadások elkerülése érdekében megfelelően nagynak kell lennie, ennek matematikai okát az 5.1. fejezetben írtam le, illetve az 5.2.1 fejezetben az 1.ábrán összehasonlítottam

[i] Az ANSI X9.31 szabványa nem elérhető ingyenesen bárki számára, ezért a benne szereplő követelményeket későbbi, RSA-ról szóló publikáció alapján foglalom össze.[7]

[ii] Az 5.4 fejezetben említett ECM $exp((1 + o(1))\sqrt{ln(p) \cdot ln(ln(p))})$ műveletigényű, ahol $p < q, n = pq$.

az ETSI és a NIST által elvárt X9.31 szabványban szereplő elvárást.

4) Lehman tovább fejlesztette a Fermat módszert[7]: abban az esetben, ha $\frac{p}{q}$ közel van egy $\frac{r}{s}$ racionális számhoz, ahol r és s is kis egész szám, akkor a támadó próbálkozhat nrs Fermat faktorizációjával, hiszen ekkor $\frac{p}{q} \cdot \frac{s}{r} \approx 1$, tehát ps és qr közel van $\sqrt{nrs}$, így a Fermat faktorizáció könnyen megtalálja ps-t és qr-t. Ekkor mivel $n = pq$, ahol p és q prímek, ezért $lnko(n, ps) = p$ és $lnko(n, qr) = q$ lesz. Lehman módszert adott ilyen racionális számok keresésére is.

6) 7) A d titkos kulcs meghatározásakor a $de \equiv 1 \ (mod \ \varphi(n))$ lineáris kongruenciának maximum 1 megoldása lehet, hiszen a megoldhatóságról szóló tétel szerint csak akkor oldható meg, ha $lnko(e, \varphi(n)) = 1$ és ebben az esetben pontosan 1 megoldást kapunk, mivel a megoldások száma pontosan $lnko(e, \varphi(n))$. Az üzenet visszanyerése viszont az $m^{ed} \equiv m \ (mod \ n) \ (m \in \mathbb{Z}_n^*)$ összefüggésen alapszik, ennek ellenére létezhet $d' \neq d$ mellyel az üzenet szintén visszakapható. Ezt alátámasztja a következő tétel, melynek bizonyítása a megjelölt forrásban található:
5.3.1.1. tétel[14] Legyen $n = pq$ két prímszám szorzata és $lnko(e, \varphi(n)) = 1$. Ekkor $m^{ed} \equiv m \ (mod \ n) \ (m \in \mathbb{Z}_n^*) \Leftrightarrow ed \equiv 1 \ (mod \ \lambda(n))$, ahol $\lambda(n) = lkkt(p-1, q-1)$.
Ezért létezik d-vel nem feltétlenül egyenlő $d' < \lambda(n)$, így abban az esetben, ha $\lambda(n)$ "kicsi", a támadó próbálkozhat egy d' egyszerű keresésével. Ennek elkerülése érdekében érdemes olyan p-t és q-t megválasztani, melyre $p-1$ és $q-1$ rendelkezik nagy és különböző prímtényezővel, illetve az $lkkt(p-1, q-1) = \frac{(p-1)(q-1)}{lnko(p-1,q-1)}$ összefüggés miatt célszerű, ha $lnko(p-1, q-1)$ kicsi.

5.3.2. Követelmények 2005-től

A NIST[S13] továbbra is az X9.31 és a PKCS#1 dokumentumokban foglaltakat követeli meg az RSA-ra vonatkozóan. Melyre a publikáció szerint a FIPS 186-3-ban[S24] adnak majd saját útmutatást digitális aláírásokra, azonban ezt csak később 2009-ben adták ki.

A változás a modulus hosszával szembeni követelményben van, az elvárás[S13][S14][S15]:
-2010-ig legalább 1024bit hosszú n
-2010 és 2030 között legalább 2048bit hosszú n
-2030 után legalább 3072bit hosszú n használata
Amiket bit-ben kifejezett biztonsági erősségekkel határoztak meg, amelynek célja a különböző eljárások biztonságának összehasonlíthatósága. Az RSA biztonságát helyes kulcsválasztás esetén az n faktorizálásának a nehézsége jelenti, így a biztonsági erőssége vélhetően az ismert leggyorsabb faktorizációs módszer (GNFS) műveletigényéből számolható. A GNFS[12] műveletigénye:
$exp\left(\left(\sqrt[3]{\frac{64}{9}} + o(1)\right)(ln(n)^{\frac{1}{3}} \cdot (ln(ln(n)))^{\frac{2}{3}}\right)$, ahol $\left(\sqrt[3]{\frac{64}{9}} \approx 1.923\right)$
Ami valóban egyezést mutat a NIST módszerével, ahol a biztonsági erősséget a következő módon

számolják$_{[S21]}$:
-$E(nBits) = \frac{1.923 \cdot \sqrt[3]{nBits \cdot ln(2)} \cdot \sqrt[3]{(ln(nBits \cdot ln(2))^2} - 4.69}{ln(2)}$, ahol $nBits$-el jelölik n bithosszát.
-$S(nBits) = E(nBits)$-hez legközelebb eső 8 többszörös.
Az S-et tekintik a biztonsági erősségnek. A kerekítésre a más algoritmusokhoz való könnyebb nagyságrendbeli összehasonlítás miatt van szükség. A NIST dokumentumai ettől kezdve a modulus hosszát nem bit-ben, hanem a biztonsági erősségükben határozza meg. Az RSA-ra vonatkozóan a leggyakrabban előforduló értékek:

nBits	E	biztonsági erősség
1024	80	80
2048	110.12	112
3072	131.97	128
4096	149.73	152

3.táblázat

5.3.3. Követelmények 2009-től

A NIST először adta ki az SP 800-56B$_{[S19]}$ publikációt, melyben részletes útmutatást ad a megfelelő kulcsok generálásához. A szabvány elvárásai:
1) $nBits = 1024$ vagy 2048. 2010 után az $nBits = 1024$ nem elfogadott.
2) $65537 \leq e < 2^{256}$
3) $lkkt(p-1, q-1) > e$
4) $p, q \in [\sqrt{2} \cdot 2^{nBits/2-1}, 2^{nBits/2} - 1]$
5) $|p - q| > 2^{(nBits/2)-100}$
6) $2^{nBits/2} < d < lkkt(p-1, q-1),\ ed \equiv 1\ (mod\ lkkt(p-1, q-1))$

Magyarázat néhány követelményre:
1) 4) 5) Követelmény ekvivalens a korábbi évek elvárásaival.

6) Ennek betartásával a 2000-es szabvány 6) pontjában szereplő 5.3.1.1. tétel szerint sem létezhet $d' < d$, illetve $lkkt(p-1, q-1)$ elegendően nagy, ezért ellenáll az ott leírt támadással szemben.

Jelentős változás, hogy a 2009-es NIST 800-56B szabvány nem várja el, hogy a $\frac{p}{q}$ közelében ne legyen kis számlálóval és nevezővel rendelkező $\frac{a}{b}$ racionális szám. Ez azzal magyarázható, hogy ekkora számok esetén, az ilyen jellegű támadás gyakorlati kivitelezésére nagyon kicsi esély van$_{[7]}$. Illetve azt sem várja el, hogy $|p - q|$ rendelkezzen nagy prímtényezővel. A dokumentum szerint a korábbi p^-, p^+, q^- és q^+-al szembeni követelményeknek sem kell megfelelni, ezzel

szemben a digitális aláírásokhoz 2009-ben kiadott FIPS 186-3$_{[S24]}$ $nBits = 1024$ esetén elvárja, $nBits > 1024$ esetén viszont választhatónak tekinti az alábbi elvárásokat:

nBits	p^-, p^+, q^- és q^+ minimális bithossza	$len(p^-) + len(p^+)$ és $len(q^-) + len(q^+)$ maximális hossza	
		p, q valószínű prímek	p, q biztos prímek
1024	> 100 bit	< 496 bit	< 239 bit
2048	> 140 bit	< 1007 bit	< 494 bit
3072	> 170 bit	< 1518 bit	< 750 bit

4.táblázat

Az $len(p^-) + len(p^+)$ és $len(q^-) + len(q^+)$[iii] maximális hosszával szembeni követelményekre a FIPS 186-3-ban található, ezekkel a segédprímekkel történő véletlen p-t és q-t generáló algoritmusoknak a megfelelő működése miatt van szükség.[iv]

A paraméterek véletlen választásának eljárásai:
A 800-56B két függvénycsaládot ad meg a korábban leírt követelmények teljesítésére, az egyik fix e esetén használható, a másik pedig az e-t is véletlenszerűen generálja. Mindkét függvénycsalád 3 függvényből áll, melyek fix e esetén $(s, nBits, e)$, véletlen e esetén pedig $(s, nBits, eBits)$ bemenettel (ahol az s az elvárt biztonsági erősség, az eBits pedig az e bithossza) a következő kimeneteket adják vissza:
-basic: (n, e) és (n, d)
-prime-factor: (p, q, d)
-crt: $(n, e, d, p, q, dP, dQ, qInv)$[v]
Fix e esetén az eljárás a következő (rsakpg1-basic)$_{[S19]}$:

(1) Az intervallumok ellenőrzése:

(1.1) Ha $s \notin \{80, 112\}$, akkor a kimenetben jelezze, hogy a biztonsági erősség nem megfelelő és lépjen ki.

(1.2) Ha $nBits \notin \{1024, 2048\}$, vagy $nBits$ kisebb, mint amit a biztonsági erősségnek s paraméterrel elvárna. (3. táblázat szerint), akkor a kimenet jelezze, hogy a modulus hossza nem megfelelő és lépjen ki.

(1.3) Ha e nem egy páratlan egész a $65537 \leq e < 2^{256}$ intervallumból, akkor kimenetben jelezze, hogy a publikus exponens nem megfelelő és lépjen ki.

[iii] $len(p^-)$ a p^- bithosszát jelöli; p^+, q^- és q^+-ra ugyan így

[iv] Ezekre az algoritmusokra nem térek ki a szakdolgozatomban, mert az SP 800-56B alapvetően el sem várja a segédprímek használatát. Megtalálhatóak a FIPS 186-3$_{[S24]}$ B.3.4, B.3.5, B.3.6 fejezeteiben.

[v] Kínai maradéktétel formátum

(2) Generálja a p-t és q-t a FIPS 186-3-ban meghatározottak szerint.

(3) Határozza meg a d privát exponenst $ed \equiv 1 \ (mod\ lkkt(p-1, q-1))$ szerint. Ha $d \leq 2^{nBits/2}$, akkor törölje d-t és lépjen a 2. ponthoz.

(4) Határozza meg az n modulust $n = pq$ képlettel.

(5) Vizsgálja meg néhány $1 < k < n-1$ esetén, hogy $k \equiv (k^e)^d \ (mod\ n)$ teljesül-e, abban az esetben, ha bármelyik tesztelt k-ra nem teljesül, akkor a kimenetben jelezze, hogy inkonzisztencia van és lépjen ki.

(6) Térjen vissza (n, e) publikus és (n, d) privát kulcsokkal.

Az 5. pontban a vizsgálatot nem szükséges nagyon sok esetre elvégezni, mivel ha a paraméterek nincsenek megfelelő összefüggésben egymással, akkor nagyon pici eséllyel lehet visszakapni az eredeti üzenetet.
A rsakpg1-prime-factor algoritmus (fix e esetén), annyiban tér el a basic függvénytől, hogy a 6. pontban (n, e) és (n, d) kimenet helyett (n, e) publikus és (p, q, d) privát kulccsal tér vissza.
Az rsakpg1-crt algoritmus (szintén fix e esetén) viszont a 4. pontban n modulus kiszámolása mellett a Kínai maradéktételhez szükséges paramétereket is kiszámolja (mellyel az RSA használatát lehet jelentősen gyorsítani), majd ezekkel tér vissza. Az eltérés a basic-hoz képest:

(4) Határozza meg az n, dP, dQ és a $qInv$ értékeket:

- **(4.1)** $n = pq$
- **(4.2)** $dP \equiv d \ (mod\ p-1)$
- **(4.3)** $dQ \equiv d \ (mod\ q-1)$
- **(4.4)** $qInv \equiv q^{-1} \ (mod\ p)$

(6) Térjen vissza az (n, e) publikus és az $(n, e, d, p, q, dP, dQ, qInv)$ privát kulcsokkal.

A nem fix e bemenetet váró függvénycsalád hasonlóképpen épül fel, annyi eltéréssel, hogy az rsakpg2-basic függvénye az rsakpg1-basic-al ellentétben az 1.3 pontban $eBits$ bemenetet ellenőrzi $17 \leq eBits \leq 256$ feltétellel. Illetve a 2. pontban p és q generálása előtt választ egy véletlen e értéket a $[2^{eBits-1}+1, 2^{eBits}-1]$ intervallumból egy a NIST SP 800-90-ben is elfogadott Random Bit Generátor segítségével.

A 2. pontban történő p és q generáláshoz a FIPS 186-3 alapvetően 5 módszert javasol, melyből 3 a 4. táblázatban szereplő kritériumokat is teljesíti, de ezeket az SP 800-56B nem várja el. A segédprímekre vonatkozó feltételek nélkül 2 algoritmus használható, az egyik valószínű prímeket, a másik pedig bizonyított prímeket generál. Erről részletesebben az 5.2.3 fejezetben írtam[vi].

[vi] A FIPS 186-3-ban$_{[S24]}$ és a FIPS 186-4-ben$_{[S25]}$ található algoritmusok nem térnek el egymástól

A paramétereket tesztelni kell, hogy megfelelnek-e az elvárt követelménynek, ezt az 5. lépés után és a 6. lépés előtt célszerű megtenni, viszont a tesztelés nem mindig a p-t és a q-t generáló algoritmus közben történik, ezért szükséges előtte (n, d, e) inputtal a (p, q) meghatározása is, ez a RecoverPrimeFactors függvénnyel történik[S19]:

(1) Legyen $m = de - 1$. Ha m páratlan, akkor lépjen a 4 pontra.

(2) Felírja m-et $m = 2^t r$ alakban, ahol az r a legnagyobb páratlan egész szám, ami osztója m-nek és $t \geq 1$.

(3) For $i = 1$ to 100 do:

- **(3.1)** Generál egy véletlen $g \in [0, n-1]$ egész számot.
- **(3.2)** Legyen $y = g^r \ (mod\ n)$.
- **(3.3)** Ha $y = 1$ vagy $y = n - 1$, akkor lépjen a 3.7 pontra.
- **(3.4)** For $j = 1$ to $t - 1$ do:
  - **(3.4.1)** Legyen $x = y^2 \ (mod\ n)$.
  - **(3.4.2)** Ha $x = 1$, akkor lépjen az 5 pontra.
  - **(3.4.3)** Ha $x = n - 1$, akkor lépjen a 3.7 pontra.
  - **(3.4.4)** Legyen $y = x$.
- **(3.5)** Legyen $x = y^2 \ (mod\ n)$.
- **(3.6)** Ha $x = 1$, lépjen az 5. pontra.
- **(3.7)** Új ciklus.

(4) Térjen vissza "p és q nem található" üzenettel és lépjen ki.

(5) Legyen $p = lnko(y - 1, n)$ és legyen $q = n/p$.

(6) Térjen vissza a (p, q) kimenettel.

Boneh 1999-es publikációja szerint[S19][A3] a 3-as pontban egyetlen ciklus 1/2 valószínűséggel jár sikerrel, így az algoritmus sikeressége az ismétlésszám halmazán értelmezve geometriai eloszlást követ. A (p, q) sikeres visszanyerése várhatóan 2 lépés alatt megtörténik, 100 ciklus elvégzése után a sikertelenség valószínűsége szinte elhanyagolható 2^{-100}, ezért ez esetben arra következtethetünk, hogy az RSA paraméterei valamilyen alapvető feltételt nem teljesítenek.
A teljes tesztelő algoritmus megtalálható a NIST hivatkozott szabványában[19], az ebben elvárt követelményekről korábban írtam, ezért ezt nem részletezem újból.

5.3.4. A 2011-es NIST SP 800-131A publikáció

A 2005-ös publikáció szerint$_{[S13]}$ a $80bit$-es biztonsági erősség[vii] 2010. végéig használható. 2011. januárjában adta ki a NIST$_{[S22]}$ az első ajánlását arra vonatkozóan, hogyan kell más kulcshosszakra áttérni, ez alapján a $80bit$-es biztonsági erősséget 2013. végéig elfogadhatóvá tették. Ennek oka, hogy a vártnál kisebb mértékű javulást tudtak elérni az RSA-t veszélyeztető faktorizáló módszerekben, és a számítógépek számítási teljesítménye továbbra sem teszi lehetővé a korábbi módszerek sikeres alkalmazását ilyen biztonság mellett. Az akkori becslések szerint valószínűleg további 6-7 évre lehet szükség, mire az $1024bit$-es modulusú, a követelményeknek megfelelő paramétereket használó RSA támadhatóvá válik. Ezt mutatja az is, hogy 2011-ig a legnagyobb n modulus, amit faktorizálni tudtak $768bit$-es volt és ehhez is 6 hónapos számolásra volt szükség.[viii]

5.3.5. Követelmények 2014-től

Fontos változás, amiről az 5.2.3. fejezetben is említést tettem, hogy a NIST az elfogadható véletlen generáló algoritmusokat tartalmazó SP 800-90A$_{[S26]}$ szabványából eltávolította a Dual_EC_DRBG-t, melynek okáról korábban írtam.

Ezt követően a paraméterekkel szembeni követelmények a NIST SP 800-56B, az SP 800-57 Part1, az SP 800-131A és a FIPS 186 publikáció sorozatokban csak a modulus bithosszában változtak, amit a következő fejezetben összefoglalok a 5. táblázatban.

A paraméterek teszteléséhez szükséges (p, q) visszaállításhoz viszont a 2019-ben kiadott 800-56B Rev.2 szabvány egy determinisztikus módszert is javasol$_{[S21]}$:
$RecoverPrimeFactors(n, e, d) = (p, q)$ (,ahol $p > q$):

(1) Legyen $a = (de - 1) \cdot lkkt(n - 1, de - 1)$.

(2) Legyen $m = \left\lfloor \frac{a}{n} \right\rfloor$ és $r = a - mn$.

(3) Legyen $b = \frac{n - r}{m + 1} + 1$. Ha $b \notin \mathbb{N}$ vagy $b^2 \leq 4n$, akkor hibajelzéssel kilép.

(4) Legyen $\Upsilon = \sqrt{b^2 - 4n}$, ahol $\Upsilon > 0$. Ha $\Upsilon \notin \mathbb{N}$, akkor hibajelzéssel kilép.

(5) Legyen $p = \frac{b + \Upsilon}{2}$ és $q = \frac{b - \Upsilon}{2}$.

(6) Visszatér a (p, q) kimenettel.

[vii] Az ennek megfelelő nBits a 3. táblázatban található.
[viii] A sikeres RSA modulus faktorizálásokat a 4. ábrán szemléltetem a 5.4. fejezetben.

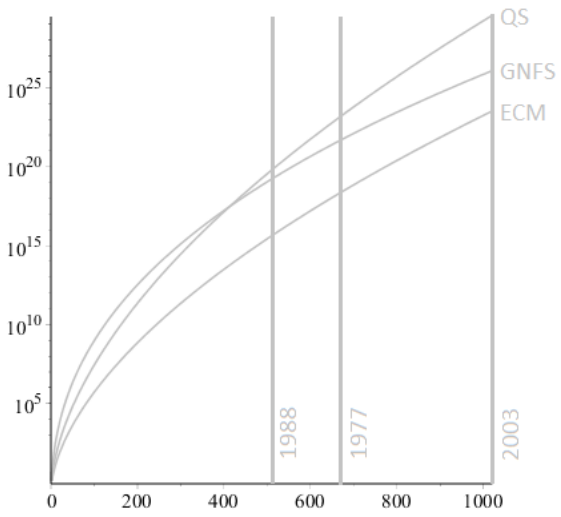
A módszer 3. lépésében meghatározott b-nek $p + q$-val kell egyenlőnek lennie, a 4. lépésében meghatározott Υ-nak pedig $p - q$-val. Majd ezekből könnyen kiszámolható p és q az 5. lépésben. Abban az esetben, ha b és Υ nem a megfelelő értékeket adja, az RSA-hoz használt n, e, d, p és q paraméterek valamelyike nem megfelelően lett generálva és így a kulcspár nem felel meg a követelményeknek.

Boneh 1999-ben publikált algoritmusával szemben, itt a futás nem függ a véletlentől, a (p, q) visszaállításának számításigénye a paraméter méret alapján meghatározható.

5.4. Paraméterek hosszának időbeni változásai

Korábbi fejezetekben bemutattam, mennyire fontos az RSA paramétereinek helyes megválasztása és idővel hogyan változtak az elvárások, annak érdekében, hogy egyes támadási módszereket el lehessen kerülni. Viszont a számítástechnika és a faktorizációs algoritmusok fejlődése lehetővé teszi, hogy egyre nagyobb számokat tudjunk prímtényezőkre bontani, aminek nehézsége a titkosítás biztonságának az alapját képzi.
Ezért ebben a fejezetben szemléletesen szeretném összefoglalni, hogy az n modulus bit hosszával szembeni elvárás az idő során hogyan változott, illetve ezt mi indokolhatta.

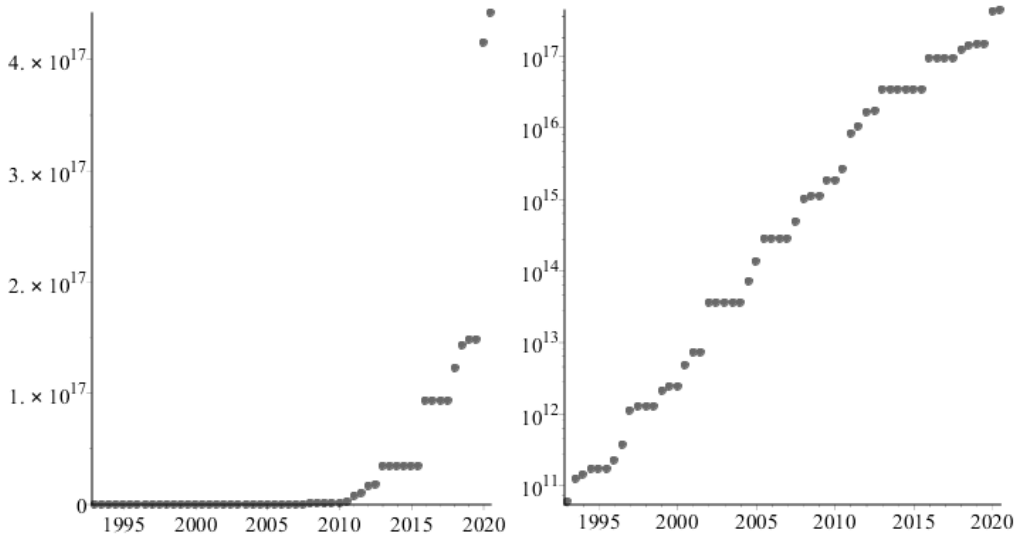
Kezdetben 1977-ben Rivest, Shamir és Adleman az n 200 decimális számjegyűnek ($log_2 10^{200} \approx$ 664 bit hosszúnak) megválasztását, az akkor ismert leggyorsabbnak számító Richard Schroeppeltől származó prímfaktorizációval indokolta[4]. Amivel nagyságrendben $exp(\sqrt{ln(n) \cdot ln(ln(n))}$ műveletigényel számoltak a publikációjukban. Ahol ln a természetes alapú logaritmus.
1981-ben publikálta Carl Pomerance a kvadratikus szita módszert (quadratic sieve, QS), tovább fejlesztve Schroeppel munkáját, melynek műveletigénye szintén $exp(\sqrt{ln(n) \cdot ln(ln(n))}$[10].
1987-ben Hendrik W. Lenstra publikálta az elliptikus görbéket használó faktorizációs módszerét[11] (ECM), ez az akkor ismert leggyorsabb speciális célú faktorizáló algoritmus, mellyel $n = pq$ esetén megközelítőleg $exp(\sqrt{ln(p) \cdot ln(ln(p))}$ műveletigénnyel lehet faktorizálni, ahol $p < q$ prímek.
Később az 1988-ban kiadott X.509 szabványban ennek ellenére már elegendőnek tartották az 512 bit hosszú modulus megválasztását is, ennek a döntésnek a lehetséges okát szeretném szemléltetem:



2. ábra.

Az ábrán látható a becsült műveletigény logaritmikus skálán, az n bithosszának függvényében.

Jól látható, hogy a jelentős javulás ellenére még az ECM algoritmussal és n 512bit hosszúnak megválasztásával is megközelítőleg $exp(\sqrt{ln(\sqrt{2^{512}}) \cdot ln(ln(\sqrt{2^{512}}))} \approx 4.53 \cdot 10^{15}$ műveletre van szükség, ami az akkori leggyorsabb számítógépek számára is nehézséget okozott. Az 1988-ban leggyorsabbnak számító Cray Y-MP modell is csak $2.3 \cdot 10^9$ flop/s teljesítményre volt képes. Emellett a modulus bithosszának a meghatározásakor, figyelembe kellett venni azt is, hogy ekkora számokkal történő moduláris hatványozás nagyon számításigényes feladat.
1990-ben A.K. Lenstra, H.W. Lenstra, Jr., M.S. Manasse és J.M. Pollard közzétett egy publikációt[12], mely szerint $exp(c \cdot ln(n)^{1/3}(ln(ln(n))^{2/3}))$ műveletigénnyel faktorizálni lehet az n számot, c paraméterre $2(2/3)^{2/3} \approx 1.526$ értéket adtak. A módszerük segítségével 1996-ban sikeresen faktorizálták a 432 bit hosszú RSA-130 számot, mely jól mutatja, hogy az 1988-ban ajánlott 512 bit hosszú modulus valóban időtállónak bizonyult. (Az RSA számokra a későbbiekben még visszatérek). Ezt követően az RSA egyre szélesebb körben elterjedt. Ennek a hatására is számos faktorizációs módszerek gyorsításáról szóló tanulmány született, speciálisan arra az esetre, ha n pontosan 2 prím szorzata.
Ezzel párhuzamosan a jelentős technológiai fejlődés miatt a számítási kapacitás is folyamatosan nőtt:



3. ábra.

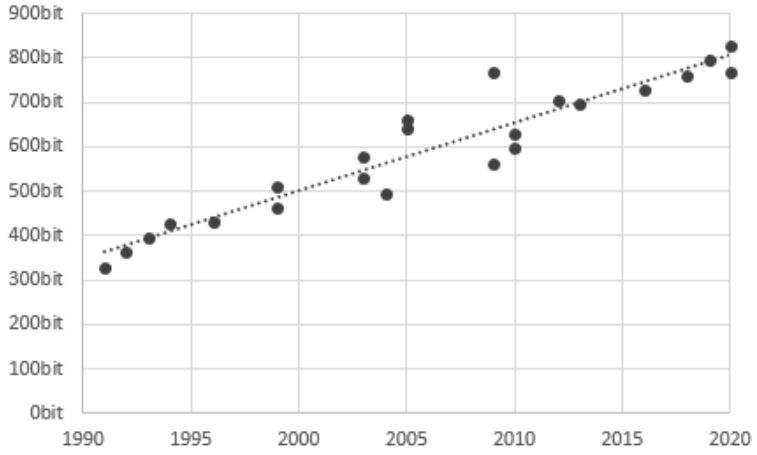
1993-tól ábrázoltam a világ leggyorsabb számítógépeinek a számítási teljesítményét Flop/s-ban mérve[13]. Látható, hogy a teljesítmény az évek során exponenciálisan nőtt.
A matematikai és technológiai fejlődést figyelembevéve, több szervezet is protokollokat dolgozott ki, melyekben a biztonságosnak tartott n bithosszúság is meg lett határozva. A szakdolgozatomban feldolgozott szervezetek is folyamatosan változtatták a paraméterek hosszával szembeni elvárásaikat, melyek közül néhányra korábban kitértem, a nagyságrendbeli változás

szemléltetéseként táblázatban összefoglaltam[ix]:

Elvárások n bithosszára						
ETSI					NIST	
kiadva	érvényes bithossz				kiadva	érvényes bithossz
	1évig	3évig	6évig	10évig		
					2000.01	$\geq$1024
2003.03.	1020					
2005.07.	-	768	1024	2048	2005.08.	2010-ig: $\geq$1024
2007.11.	1024	1536	2048	2048	2009.08.	1024, 2048
2011.07.	1536	2048	2048	3072		
2014.11.	1536	2048	3072	4096	2014.09.	2048, 3072
2017.05.	1900	1900	3000	-		
2018.09.	1900	1900	3000	-	2018.07.	2048$\leq$,$\leq$15360
2019.02.	1900	1900	3000	-	2019.03.	2048$\leq$,$\leq$15360

5.táblázat

Korábban 1991-ben az RSA Laboratories bejelentette az RSA Factoring Challenge kihívást, aminek a célja az volt, hogy ösztönözzék a kutatókat, hogy újabb faktorizáló módszereket dolgozzanak ki. Kiadtak több különböző bithosszú $n = pq$ modulust, melyek faktorizálása esetén pénzjutalom járt, ezeket RSA számoknak nevezik. Több feladatra megoldás is született, ennek nyomonkövetése is jól szemlélteti a probléma megoldásának a nehézségét.



4. ábra.

[ix]A táblázat összeállításához az irodalomjegyzékben található ETSI és NIST szabványokat használtam fel.

A 4. ábrán szemléltetem a faktorizált RSA számokat az idő függvényében, amikor sikeresen megtalálták p és q értéket. Jelenleg ismert legnagyobb RSA szám, melyet sikerült prímtényezőkre bontani 829bit-ből áll. Az ábrán az is látható, hogy lineáris tendenciát mutat a modulus hosszában mérve. Ez is jelzi az exponenciális fejlődést, viszont a 2048, 3072bit-es n-ek a paraméterek helyes megválasztása esetén valóban megfelelő védelmet nyújthatnak.
Azonban erre befolyással lehet a kvantumszámítógépek fejlődése. Erre részletesebben kitérek a következő fejezetben, ahol arról írok, milyen változások lehetségesek a jövőben, melyek kihathatnak az RSA algoritmus paraméter választásaira.

5.5. Lehetséges változások a jövőben

Előző fejezetben is rámutattam, hogy az RSA még megfelelő paraméterek választása mellett sem feltörhetetlen. A biztonságot az nyújtja, hogy az n modulus faktorizálásának akkora a számításigénye, hogy ez belátható időn belül kivitelezhetetlen hagyományos számítógépekkel, még a teljesítmény exponenciális ütemű fejlődése mellett is.
Azonban jelentős változást hozhatnak majd az informatikában a kvatumszámítógépek, ezeknek a titkosításokra gyakorolt hatásáról is számos kutatás indult. [x]Először Peter Shor publikált egy kvantum algoritmust, mellyel megfelelően nagy teljesítményű kvantumszámítógéppel elméletben hatékonyan (polinom időben) lehetne faktorizálni az n modulust. A gyakorlatban a fizikusoknak 2012-ben sikerült is prímtényezőkre bontaniuk a 143-at, mindössze 4 kvantumbitet használva. 2014-ben pedig hasonló géppel az 56153-at is. Nagyobb számok esetén a feladat összetettebbnek bizonyult, egy 2015-ös becslés szerint egy 2048bit-es modulus esetén a sikeres faktorizáláshoz egy milliárd kvantumbitre lenne szükség. Viszont 2019-ben kidolgoztak egy elméletet, mellyel 20 millió kvantumbittel 8 órán belül lehetne egy 2048bit-es RSA p és q paramétereit meghatározni. Ez is nagyságrendekkel több, mint 70 qubit, amivel a cikk publikálásakor a legnagyobb teljesítményű kvantumszámítógép rendelkezett. A kutatási eredményeket figyelembe véve azt becsülik, hogy 25 éven belül válik a modulus faktorizálásával feltörhetővé a 2048bit-es RSA ilyen gépeket használva. Ezért nem várható, hogy a közeljövőben a kvantumtechnológia fejlődése miatt változnak az RSA paramétereivel szembeni követelmények.

Hagyományos számítógépeken elvégezhető faktorizálással kapcsolatban is folyamatosan kutatások folynak, melyek a jövőben befolyással lehetnek az RSA paramétereinek a választására. Ezt jól mutatja az is, hogy a szakdolgozatom írása közben is született eredmény erre vonatkozóan, melyet Claus Peter Schnorr német kriptográfus tett közzé[18]. Az eprint-ben a következő erős állítást teszi: "This destroys the RSA cryptosystem.", ezzel szakmai vitát váltott ki, több kutató is állást foglalt az internet különböző felületein. Ezeknek utánajárva azt tapasztaltam, hogy a kutatási eredményt nem tartják elhanyagolhatónak, viszont az RSA biztonságát valószínűleg nem veszélyezteti megfelelő paraméterek választása esetén. Ennek bemutatására a gyakorlatban le is tesztelték[19], amivel arra jutottak, hogy valószínűleg a tanulmányban tett állításnál jóval nagyobb számítási kapacitást igényel Schnorr módszere.
A jövőben bármikor születhet olyan matematikai eredmény, melynek hatására a megfelelő védelem érdekében szükség lesz az RSA paraméter-követelményein változtatni.

A faktorizációs módszerek fejlődésén túl, más matematikai eredmények is születhetnek, melyek hatással lehetnek az RSA biztonságára, így a paraméterek választására is. Máig sem bizonyított, hogy az RSA feladat és a faktorizáció feladat megoldhatósága ekvivanens lenne.

[x]A kvantumszámítógépekről szóló bekezdést az MIT cikke alapján foglaltam össze.[17]

5.5.1. definíció[20] (faktorizáció feladat) Adott random választott N-hez találjunk (p, q)-t, melyre $N = pq$.

5.5.2. definíció[20] (RSA feladat) Adott random választott (N, e, c)-hez találjunk m-et, melyre $m^e \equiv c \ (mod\ N)$.

5.5.3 állítás[20] A faktorizáció feladat megoldható $\Rightarrow$ az RSA feladat megoldható.

Ennek az állításnak a $\Leftarrow$ iránya máig sem bizonyított, így előfordulhat, hogy az n faktorizálásánál könnyebb feladat az RSA-val rejtjelezett üzenet visszaállítása. Ezzel kapcsolatos kutatási eredmények is kihatással lehetnek a jövőben a paraméterek biztonságos megválasztására.

6. Jelölések és felhasznált matematikai alapismeretek

6.1. definíció[1] A b egész számot az a egész szám osztójának nevezzük, ha létezik olyan q egész szám, amelyre $a = bq$.
Jelölése: $b|a$.

6.2. definíció[1] Az a és a b számok legnagyobb közös osztója d, ha
(i) $d|a, d|b$; és
(ii) ha egy c-re $c|a$ és $c|b$ teljesül, akkor $|c| \leq |d|$.
Jelölése: $d = lnko(a, b)$.

6.3. definíció[1] Az a és b legkisebb közös többszöröse a k pozitív egész, ha
(i) $a|k, b|k$; és
(ii) ha egy $c > 0$-ra $a|c, b|c$ teljesül, akkor $c \geq k$.
Jelölése: $k = lkkt(a, b)$.

6.4. definíció[1] Az $a_1, a_2, ..., a_k$ számok relatív prímek, ha nincs egységtől különböző közös osztójuk.
Jelölése: $lnko(a_1, a_2, ..., a_k) = 1$.

6.5. definíció[1] Legyenek a és b egész számok és m pozitív egész. Azt mondjuk, hogy a kongruens b-vel modulo m, ha $m|a - b$.
Jelölése: $a \equiv b \ (mod \ m)$.

6.6. definíció[1] Legyenek a, b egészek és m pozitív egész. Ekkor az $ax \equiv b \ (mod \ m)$ kongruenciát lineáris kongruenciának nevezzük, és ennek egy megoldásán olyan s egész számot értünk, amelyet az x helyére beírva a kongruencia fennáll.

6.7. tétel[1] Az $ax \equiv b \ (mod \ m)$ lineáris kongruenciának akkor és csak akkor létezik megoldása, ha $lnko(a, m)|b$.

6.8. tétel[1] Ha az $ax \equiv b \ (mod \ m)$ lineáris kongruencia megoldható, akkor a megoldásszáma $lnko(a, m)$.

6.9. definíció[1] Tetszőleges n pozitív egész esetén $\varphi(n)$ az $1, 2, ..., n$ számok közül az n-hez relatív prímek számát jelenti.
Jelölése: $\varphi(n)$ (Euler-féle φ-függvény)

6.10. tétel[1] Legyen n kanonikus alakja $n = p_1^{\alpha_1} p_2^{\alpha_2} ... p_r^{\alpha_r} = \prod_{i=1}^{r} p_i^{\alpha_i}$, ahol $\alpha_i > 0$.
Ekkor $\varphi(n) = (p_1^{\alpha_1} - p_1^{\alpha_1 - 1}) ... (p_r^{\alpha_r} - p_r^{\alpha_r - 1}) = \prod_{i=1}^{r} (p_i^{\alpha_i} - p_i^{\alpha_i - 1})$.

6.11. megjegyzés A 6.7. tétel triviális következménye, hogy $n = pq$ esetén (,ahol $p > q$ prímek) $\varphi(n) = (p-1)(q-1)$.

6.12. tétel[1] (Euler-Fermat tétel): $lnko(a, m) = 1 \Rightarrow a^{\varphi(m)} \equiv 1 \ (mod \ m)$

6.13. definíció[1] (kvadratikus maradék) Legyen $p > 2$ prím és $lnko(a, p) = 1$. Az a számot aszerint nevezzük kvadratikus maradéknak, illetve kvadratikus nemmaradéknak modulo p, hogy az $x^2 \equiv a \ (mod \ p)$ kongruencia megoldható-e, vagy sem.

6.14. definíció[1] Az $\left(\frac{a}{p}\right)$ Legendre-szimbólumot a következőképp értelmezzük:

$$\left(\frac{a}{p}\right) = \begin{cases} 1, & \text{ha } a \text{ kvadratikus maradék } (mod \ p) \\ -1, & \text{ha } a \text{ kvadratikus nemmaradék } (mod \ p) \end{cases}.$$

6.15. definíció[1] A $2^{2^n} + 1$, $n = 0, 1...$ alakú számokat Fermat-számoknak nevezzük.
Jelölése: F_n.

6.16. definíció[1] Legyen $n > 1$ páratlan szám, $n - 1 = 2^k r$, ahol r páratlan. Az $a^r, a^{2r}, a^{4r}, ..., a^{2^{k-1}r}$ számokat jó sorozatnak nevezzük, ha ezek modulo n vett legkisebb abszolút értékű maradékai között előfordul -1 vagy pedig a^r maradéka 1.

6.17. tétel[1] Ha n prím, akkor a 6.16. definícióban szereplő sorozat minden $a \not\equiv 0 \ (mod \ n)$ esetén jó sorozat.
Ha n összetett, akkor 6.16. definícióban szereplő sorozat egy modulo n teljes maradékrendszer elemeinek kevesebb, mint a felére alkot jó sorozatot.

6.18. megjegyzés Boneh 1999-es p, q-t meghatározó algoritmusának[S17] és a Miller-Rabin prímtesztnek az alapját a 6.17. tétel jelenti, a pontos leírás megtalálható a [1] hivatkozásban.

6.19. tétel[1] (Prímszámtétel) $\lim_{x \to \infty} \frac{\pi(x)}{\frac{x}{ln(x)}} = 1$ (ahol $\pi(x)$ az x-nél nem nagyobb pozitív prímek száma)

6.20. tétel (Stirling-formula) $\lim_{n \to \infty} \frac{n!}{\sqrt{2\pi n}(\frac{n}{e})^n} = 1$

7. Zárszó

A szakdolgozatomban az RSA algoritmus paraméter-követelményeinek változásait mutattam be néhány korábbi szabvány alapján és igyekeztem rávilágítani arra, hogy az alap elgondoláson felül mennyi buktatót és támadási lehetőséget rejt magában pusztán a nem megfelelő paraméter választás is, amivel kezdetben a megalkotói nem is számoltak. Elsőre egyszerűnek tűnhet a moduláris hatványozást használó módszer és az alapötlet is, hogy a számok faktorizálása nagyságrendekkel bonyolultabb feladat, mint két szám szorzatának kiszámolása, de a gyakorlatban a téma szinte kimeríthetetlenül szerteágazó. Az 1977-es első publikálás óta számos tanulmány és kutatási eredmény született róla, melyek közül néhányat a szakdolgozatomban is bemutattam vagy említést tettem.

A biztonságos használatához a paramétereknek nem pusztán a nagyságrendbeli követelményeknek kell megfelelnük és a paraméterek közti elvárt matematikai összefüggéseknek, hanem ugyan olyan fontos a megfelelően történő véletlen megválasztásuk is, melyekre szintén útmutatást adnak különböző szervezetek szabványai, amiről írtam korábban is. Most viszont szeretném egy szemléletes példán bemutatni, annak a kiemelt fontosságát, hogy a paraméterek véletlen megválasztásának eljárása is a követelményeknek megfelelően történjen.

Írtam egy maple programot, ami generál egy (p, q) prímpárt. $nBits = 3072bit$ megválasztás mellett teljesíti a következő p, q-ra vonatkozóan elvárt kritériumokat:
$p, q \in [\sqrt{2} \cdot 2^{nBits/2-1}, 2^{nBits/2} - 1]$
$|p - q| > 2^{(nBits/2)-100}$

Az így kapott $n :=$
4172726490450564590581773113719866140979151739808210273638009888530014781309420048659625450881538853090350963490930132182667967904206270033772367995331654944218420376190761598344013384809616754052003228743951485985445437627392986072331599297417722961086073260410878517634498029985041629369900896758793903903981369368199117612655422318133563795179417523286872161028238324186976675197775266180930501266995765078145418613220378019408152048959892333373306339257112603782321368460044303522432273922619812951646932002191631610922068921743757666781927839699552396970963449557140567924540081563139673637748949670076947440613586510297424161910580961443697219201338232890798276384046409241123092120489062897258198704516874592595033244907175608763691995965070179151219013674294690439001542968710568406044827919601459053732374000016406720942314621734462573249729944589255495932822176908029238767091048596681425725611917053476829631174827

modulus láttán elsőre azt hinné az ember, hogy ha az e és d paramétereket is szintén a követelményeknek megfelelően választja, akkor esélytelen a támadás. Hiszen ekkora méretű n esetén a korábban leírt okok miatt Pollard és Williams speciális faktorizációs módszerével sem jutnánk eredményre, megfelelően megválasztott d esetén Wiener, Boneh, Durfee és még Weger támadási módszerei sem működnének. A p és a q megfelelő távolsága miatt Fermat faktorizációval sem érdemes próbálkozni, sem a Lehman által tovább fejlesztett változatával. Jelenlegi tudás szerint egyetlen általános vagy speciális faktorizációs technika sem létezik ekkora szám hatékony prímtényezőkre bontására, melyet az eddig faktorizált RSA számok nagyságrendjei is jeleznek. A kvantumtechnológia sem elég fejlett még ekkora számok faktorizálásához.

Mindezek ellenére egy maple-ben írt programmal sikerült megdöbbentő módon, megközelítőleg 12 másodperc alatt meghatároznom n-ből p és q értékeit. Az algoritmusok megvalósításában és szakdolgozatom megírásában a témavezetőm, Szabó István online is elérhető, RSA témájú érdekes előadásában elhangzottak is inspiráltak[22].

Ez elsőre hatalmas eredménynek tűnhet, viszont a módszer bemutatása során ki fog derülni, hogy mégsem az, csupán a kulcspár szándékosan helytelenül lett generálva:

Először létrehoztam egy 100 elemből álló listát L-et, mely véletlenül választott páros egész számokat tartalmaz a $[2^{1436}, 2^{1536}]$ intervallumból. Majd választottam egy véletlen p prímet a $[2^{1535.5}, 2^{1536} - 1]$-ból. A csapda viszont ezután következik, hiszen q-t már nem az előírtak szerint határoztam meg, hanem véletlenszerűen választottam az L listából egy $L[s]$ elemet, q-t pedig p-nek a függvényében határoztam meg úgy, hogy q egyenlő legyen $p + L[s]$ után következő legelső prímmel.
A paramétereket így megválasztva n két, első ránézésre véletlennek tűnő prímszám szorzata. Viszont, mivel a két prím egy egyszerű összefüggésben áll egymással, így a faktorizációs probléma leegyszerűsödik másodfokú egyenletek megoldási kísérleteire a következő módon:
$q = p + L[s] + \delta \Rightarrow n = pq = p(p + L[s] + \delta) \Rightarrow p^2 + (L[s] + \delta)p - n = 0.$
Innen a másodfokú egyenlet megoldóképletéből már triviálisan következik, hogy
$p_{1,2} = \dfrac{-(L[s] + \delta) \pm \sqrt{(L[s] + \delta)^2 + 4n}}{2}$, mivel $L[s], \delta \geq 0$ és tudjuk, hogy pozitív egész értéket várunk, ezért $p = \dfrac{-(L[s] + \delta) + \sqrt{(L[s] + \delta)^2 + 4n}}{2}$. Ezután L lista ismeretében faktorizálás helyett elegendő s és δ értékeinek visszakeresése, amit célszerű annak ellenőrzésével elvégezni, hogy a determináns négyzetszám-e. A forráskód a függelékben megtalálható.

A tárolt L lista mérete nyilvánvalóan számítási teljesítmény függvényében szabadon növelhető, illetve az egyszerű összeadás helyett más $f(p, L)$ függvény is választható és ebben az esetben

$n = p \cdot f(p, L)$ lesz, mely bonyolíthatja a számolásokat, viszont a megválasztás véletlenszerűbbnek tűnhet miatta.

Az egyszerű példám egyértelműen nem felel meg egyetlen szabvány elvárásainak sem és körültekintő paraméterválasztás során azonnal kiderülne, hogy csalásról van szó. Ezzel azt szeretném szemléltetni, hogy a paraméterek megválasztásakor milyen fontos szerepe van a generáló algoritmusnak és annak hitelességének. Így célszerű ellenőrzött algoritmusokat használni és előnyös, ha nyílt forráskódú, hogy bárki betekintést nyerhessen, az algoritmus pontosan milyen számolásokat végez.

A szakdolgozatommal kapcsolatban fontos megjegyeznem, hogy a témájából eredően kizárólag az RSA paraméter-követelményeivel és azok okaival foglalkoztam. Az RSA számos más módon is támadható, a megfelelő paraméterválasztás önmagában nem elég a támadási módszerek kivédésére.

8. Függelék

Maple forráskódok:

Moduláris hatványozás:

```
#a^b (mod n)-et állítja elő
#e biztonsági paraméter, e lépésen belül mindenképp leáll
expmod:=proc(a,b,n,e)
local u,v,pre,d,i,j,r;
r:=b;
u:=1;
v:=a;
pre:=0;
for i from 0 to e while r > 0 do
    if (r mod 2=0) then r := r/2; else
        r := (r-1)/2;
        d:=i-pre;
        pre:=i;
        for j from 1 to d do v:=v·v mod n od;
        u:=u·v mod n;
    end if;
end do;
u
end proc;
```

Pollard (p-1) módszere:

```
Pollard:=proc(b,n)
local p,pm,r,i;
p:=2;
pm:=1;
r:=2;
for i from 1 to b do
    pm:=p^round(log_p(n));
    r:=expmod(r,pm,n,1000000);
    p:=nextprime(p);
end do:
gcd(r-1,n)
end proc;
```

Szemléltető példa (p, q) csapdát rejtő generálására:

```
rp := rand(2^1535..2^1536 − 1):
rd := rand(2^1436..2^1536):
rs := rand(1..100):
L:=[seq(rd(), i=1 .. 100)]:
for i to 100 do if (L[i] mod 2 = 1) then L[i]:=L[i]+1 fi: od:

pg:=proc()
   local p, i;
   p:=rp():
   if (p mod 2 = 0) then p:=p+1 fi:
   for i from 1 to 50000 do
      if (isprime(p)) then break: else p:=p+2 fi:
   od:
   p
end proc:

qg:=proc(p)
   local i, r, q, s;
   s:=rs();
   q:=p+L[s];
   for i from 1 to 50000 do
      if (isprime(q)) then break: else q:=q+2 fi:
   od:
   q
end proc:

generation:=proc()
   local k, i, p, q;
   for i from 1 to 100 do
      p:=pg();
      q:=qg(p);
      if ((|p − q| > 2^(1356−100))and(p ≥ 2^1535.5)and(p ≤ 2^1536 − 1)and
            (q ≥ 2^1535.5)and(q ≤ 2^1536 − 1)) then break: fi:
   od:
   [p, q]
end proc:
```

```
n_factor:=proc(n, err)
    local i, j, sq, pp, qq;
    for i from 1 to 100 do
        for j from 1 to err do
            sq:=(L[i]+2· j)^2+4n;
            if (issqr(sq)) then pp:=(-(L[i] + 2 · j) + sqrt(sq))/2 break: fi;
        od:
    od:
    qq := n/qq;
    [pp, qq]
end proc:

#(p,q) generálása
gen:=generation():
p:=gen[1];
q:=gen[2];
n:=p·q;

#(p,q) meghatározása n-ből
st:=time():
fac:=n_factor(n,5000):
time()-st;
p2:=fac[1];
q2:=fac[2];
```

9. Irodalomjegyzék

[1] Freud Róbert, Gyarmati Edit - Számelmélet (2000)
Nemzeti Tankönyvkiadó Zrt., Budapest, 2000, 2006 ISBN 963 19 5888 4
[2] Wettl Ferenc – Antiszimmetrikus kulcsú titkosítás (2014)
http://math.bme.hu/~wettl/okt/kodkripto/2014/PKC.pdf (letöltve: 2020. szeptember)
[3] Szádeczky Tamás - Információbiztonsági szabványok (2014)
https://lipusz.hu/pedagogia_tanulas/nke_eiv/informaciobiztonsagi-szabvanyok.original.pdf (letöltve: 2020. szeptember)
[4] R.L. Rivest, A. Shamir, and L. Adleman - A Method for Obtaining Digital Signatures and Public-Key Cryptosystems (1977)
https://people.csail.mit.edu/rivest/Rsapaper.pdf (letöltve: 2020. szeptember)
[5] Chris Christensen - Mathematical attack on RSA (2006)
https://www.scribd.com/document/181164536/Mathematical-Attack-on-RSA (letöltve: 2020. október)
[6] Frank Moore, David Milan, Hoai Nam Tran - Implementing and Cracking the RSA Cryptosystem
http://www.math.unl.edu/~s-wmoore3/notes/RSAPaper.pdf (letöltve: 2020. október)
[7] Bc. Matúš Nemec - The properties of RSA key generation process in software libraries
https://is.muni.cz/th/oba30/thesis-nemec.pdf (letöltve: 2020. szeptember)
[8] Benne de Weger - Cryptanalysis of RSA with Small Prime Difference (2002)
http://www.enseignement.polytechnique.fr/profs/informatique/Francois.Morain/Master1/Crypto/projects/Weger02.pdf (letöltve: 2020. november)
[9] Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone - Handbook of Applied Cryptography (2001)
http://cacr.uwaterloo.ca/hac/about/chap5.pdf (letöltve: 2020. október)
[10] Márton Gyöngyvér - Kriptográfiai alapismeretek (2008)
https://docplayer.hu/3706407-Marton-gyongyver-kriptografiai-alapismeretek.html (letöltve 2020. szeptember)
[11] Hendrik W. Lenstra, Jr. - Factoring integers with elliptic curves
https://www.jstor.org/stable/1971363?origin=crossref&seq=1 (letöltve: 2021. április)
[12] A.K. Lenstra, H.W. Lenstra, Jr., M.S. Manasse, J.M. Pollard - The number field sieve
https://www.researchgate.net/publication/221590545_The_Number_Field_Sieve (letöltve: 2021. január)
[13] TOP500 project
https://top500.org/ (letöltve: 2021. április)
[14] Stefan Katzenbeisser - Recent Advances in RSA Cryptography
https://books.google.hu/books?id=8eLiBwAAQBAJ&pg=PR7&hl=hu&source=gbs_selecte

d_pages&cad=3#v=onepage&q&f=false(letöltve: 2021. április)
[15] HWSW Informatikai Hírmagazin - Szándékosan gyengíthetett az RSA
https://www.hwsw.hu/hirek/51525/rsa-nsa-dual-ec-drbg-veletlenszam-generator-biztonsag-snowden.html (letöltve: 2021. január)
[16] Dan Boneh - Twenty Years of Attacks on the RSA Cryptosystem
https://crypto.stanford.edu/~dabo/pubs/papers/RSA-survey.pdf (letöltve: 2021. március)
[17] STEVE JURVETSON - How a quantum computer could break 2048-bit RSA encryption in 8 hours
https://www.technologyreview.com/2019/05/30/65724/how-a-quantum-computer-could-break-2048-bit-rsa-encryption-in-8-hours/ (letöltve: 2021. április)
[18] Claus Peter Schnorr - Fast Factoring Integers by SVP Algorithms
https://eprint.iacr.org/2021/232.pdf (letöltve: 2021. április)
[19] Léo Ducas - Testing Schnorr's factoring Claim in SageMath [Sage]
https://github.com/lducas/SchnorrGate (letöltve: 2021. április)
[20] Ligeti Péter - Haladó kriptográfia (diasor)
https://web.cs.elte.hu/~turul/halkript_prez4.pdf (letöltve: 2021. április)
[21] Folláth János, Huszti Andrea, Pethő Attila - Informatikai biztonság és kriptográfia (2011)
https://regi.tankonyvtar.hu/hu/tartalom/tamop425/0046_informatikai_biztonsag_es_kriptografia/ch09s02.html (letöltve: 2020. október)
[22] Szabó István - RSA algoritmus: a jó paraméterek jellemzői
https://www.youtube.com/watch?v=Bk6bamAYeKI (letöltve: 2020. szeptember)

A szakdolgozathoz felhasznált szabványok:
CCITT X.509 szabvány (letöltve: 2020. szeptember)
[$S1$] https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.509-198811-S!!PDF-E&type=items

ETSI szabványok: (letöltve: 2020. szeptember)
[$S2$] https://www.etsi.org/deliver/etsi_sr/002100_002199/002176/01.01.01_60/sr_002176v010101p.pdf
[$S3$] https://www.etsi.org/deliver/etsi_ts/102100_102199/10217601/01.02.01_60/ts_10217601v010201p.pdf
[$S4$] https://www.etsi.org/deliver/etsi_ts/102100_102199/10217601/02.00.00_60/ts_10217601v020000p.pdf
[$S5$] https://www.etsi.org/deliver/etsi_ts/102100_102199/10217601/02.01.01_60/ts_10217601v020101p.pdf

[$S6$] https://www.etsi.org/deliver/etsi_ts/119300_119399/119312/01.01.01_60/ts_119312v010101p.pdf
[$S7$] https://www.etsi.org/deliver/etsi_ts/119300_119399/119312/01.02.01_60/ts_119312v010201p.pdf
[$S8$] https://www.etsi.org/deliver/etsi_ts/119300_119399/119312/01.02.02_60/ts_119312v010202p.pdf
[$S9$] https://www.etsi.org/deliver/etsi_ts/119300_119399/119312/01.03.01_60/ts_119312v010301p.pdf

SOG-IS Crypto Evaluation Scheme Agreed Cryptographic Mechanisms szabványok:
[$S10$] https://www.sogis.eu/documents/cc/crypto/obsolete/SOGIS-Agreed-Cryptographic-Mechanisms-1.0.pdf (letöltve: 2020. október)
[$S11$] https://www.sogis.eu/documents/cc/crypto/obsolete/SOGIS-Agreed-Cryptographic-Mechanisms-1.1.pdf (letöltve: 2020. október)
[$S12$] https://www.sogis.eu/documents/cc/crypto/SOGIS-Agreed-Cryptographic-Mechanisms-1.2.pdf (letöltve: 2020. október)

NIST szabványok (letöltve: 2021. február)
NIST 800-57 Part 1 szabványok:
[$S13$] https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-57p1.pdf
[$S14$] https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-57p1r2006.pdf
[$S15$] https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-57p1r2007.pdf
[$S16$] https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-57p1r3.pdf
[$S17$] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf
[$S18$] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf
NIST 800-56B szabványok:
[$S19$] https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-56b.pdf
[$S20$] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Br1.pdf
[$S21$] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Br2.pd

f

NIST 800-131A szabvány:

[$S22$] https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-131a.pdf

FIPS 186 szabványok:

[$S23$] https://csrc.nist.gov/CSRC/media/Publications/fips/186/2/archive/2001-10-05/documents/fips186-2-change1.pdf

[$S24$] https://csrc.nist.gov/CSRC/media/Publications/fips/186/3/archive/2009-06-25/documents/fips_186-3.pdf

[$S25$] https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

NIST SP 800-90A szabvány:

[$S26$] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf