Haustmisseri 2005

Stærðfræðimynstur í Tölvunarfræði

HÁSKÓLI ÍSLANDS

RSA dulkóðun

Kennari: Hjálmtýr Hafsteinsson

Björn Patrick Swift

bps1@hi.is

1 Inngangur

Með dulkóðun er átt við brenglun gagna á fyrirfram skilgreindan hátt svo þau séu ólæsileg þeim sem ekki þekkja afbrenglunaraðferðina. Tilgangurinn er að vernda upplýsingar fyrir óviðkomandi aðilum. Ferlið er mikið notað í hvers konar samskiptum og þá sér í lagi á internetinu þar sem gögn eru send yfir óþekktar og hugsanlega óöruggar samskiptaleiðir. Segja má að dulkóðun sé forsenda fjölmargra þjónusta sem í boði eru á internetinu. Án dulkóðunar væri til að mynda útilokað að reka netverslanir, netbanka eða aðra þjónustu þar sem viðkvæmar upplýsingar eru sendar á milli notanda og netþjóns.

Algengasta ósamhverfa dulkóðunaraðferðin nefnist RSA og er nefnd eftir höfundum sínum, þeim Ron Rivest, Adi Shamir og Len Adleman. Sú aðferð var birt árið 1977, var þá sú fyrsta sem útfærði lyklaskipti (e. key-exchange) og er enn víða notuð. Megin markmið þessa verkefnis er að brjóta upp RSA dulkóðun. Gefinn var 257 bita RSA almenningslykill ásamt dulkóðuðum skilaboðum í tveimur hlutum, skilaboð að lengd 33 - 64 stafir. Leitast verður við að nota einungis frjálsan hugbúnað við framkvæmdina. Einnig verður dulkóðun kynnt lítillega sem og sögulegur aðdragandi RSA rakinn í grófum dráttum.

2 Hvað er dulkóðun?

Saga dulkóðunar er löng og nær lengra aftur í tímann en flesta grunar. Erfitt er að festa hendur á hvenær menn hófu að dulkóða gögn en Sesarskóðun er eitt fyrsta dæmið sem vitað er um. Júlíusar Sesars og fylgismenn hans notuðu aðferðina þegar viðkvæm gögn voru send með sendiboðum. Aðferðin var ekki ýkja flókin en gekk út á að hliðra öllum stöfum í stafrófinu um ákveðið mörg sæti. Varð orðið ABYZ þannig að orðinu BCZA, að því gefnu að hliðrunin væri um 1 staf.

Sesarskóðun fellur í flokk *samhverfra dulkóðunaraðferða* en þá er sama aðferð notuð til kóðunar og afkóðunar. Vafasamt getur verið að notast við slíkar dulkóðunaraðferðir því koma þarf kóðunaraðferðinni (lyklinum) til viðtakandans svo hann geti ráðið skilaboðin. Sé lykillinn sendur sömu leið og hin kóðuðu skilaboð verður leikurinn til lítils. Lengi vel tíðkaðist að senda lykilinn í venjulegum pósti sem deila má um hvort teljist örugg boðleið.

Eitt er þó víst. Sé dulkóðunaraðferðin sjálf góð felst öryggi upplýsinganna fremur í öryggi sendingu aðferðarinnar en henni sjálfri.

3 Ósamhverfar dulkóðunaraðferðir

Hér koma því *ósamhverfar dulkóðunaraðferðir* til sögunnar. Guðfaðir þessarar aðferðar er oft sagður vera Ralph Merkle þótt hún hafi ekki verið almennilega útfærð fyrr en í ritgerðinni New Directions in Cryptography eftir Whitfield Diffie and Martin Hellman árið 1976[1]. Ólíkt þeim samhverfu byggja ósamhverfar dulkóðunaraðferðir á því að tvær mismunandi aðferðir eru notaðar til að kóða og afkóða. Viðtakandi útbýr bæði dulkóðunar og afkóðunaraðferð og sendir kóðunaraðferðina til sendanda skilaboðana. Hann kóðar gögnin og sendir viðtakanda til baka. Engu máli skiptir hvort einhver óprúttinn aðili komist í

1 Wikipedia: Diffie-Hellman key exchange <http://en.wikipedia.org/wiki/Diffie_Hellman>

kóðunaraðferðina því hún segir ekkert til um afkóðunaraðferðina.

Skilgreinum aðferðir E , þekkta aðferð til dulkóðunar, og D , leynda aðferð til afkóðunar. Ef M eru skilaboðin sem skal dulkóða gildir

$$D(E(M))=M \tag{1}$$

Jafnvel þótt E sé opinberuð er ekki hægt að ákvarða D .

3.1 RSA

Ritgerð þeirra Diffle og Hellman ýtti við þremur mönnum við MIT háskólanum, þeim Ron Rivest, Adi Shamir og Len Adleman. Fundu þeir leið til að framkvæma þessa aðferð á raunhæfan hátt árið 1977 og hefur hún verið nefnd í höfuðið á þeim; RSA.

RSA dulkóðun byggist á tveimur stærðfræðivandamálum.

- Erfitt er að þátta stórar heiltölur
- Erfitt er að finna m svo $m^e = c \ \boldsymbol{mod}\ n$ þar sem n er margfeldi tveggja stórra prímtalna, c er dulkóðuð tala og e er þekkt stærð.

Með "erfitt" er átt við að ekki séu til algrímar sem leysa vandamálin á skikkanlegum tíma. Skikkanlegur tími er þá flækja af margliðustigi $\Theta(n^k)$, þar sem k er fasti, en ekki af vísisvexti, eins og þáttunin er almennt. Reyndar eru til algrímar sem geta þáttað hraðar en $\Theta(\mathrm{e}^k)$ og eru í raun á milli þess að hafa flækjustig af margliðustigi og vísisvexti. Sem dæmi má nefna GNFS algríminn sem talinn er sá afkastamesti í dag. Hann vinnur á tíma

$$\Theta\left(\exp\left(\left(\frac{64}{9}n\right)^{\frac{1}{3}}(\log n)^{\frac{2}{3}}\right)\right) \tag{2}$$

Stærsta tala sem opinberlega hefur tekist að þátta var 663 bitar að lengd [2] og var þáttuð 9 maí 2005. Ofutölvur háskólans í Bonn unnu við þáttunina frá lok árs 2003 og var fjöldi aðgerða slíkur að þáttunin hefði tekið 55 ár með 2,2 Ghz tölvu heimilistölvu[3].

3.1.1 ÚTFÆRSLA

Valdar eru tvær stórar prímtölur p og q þannig að margfeldi þeirra $n = p \cdot q$ sé mjög stór, yfirleitt 1024 - 2048 bitar. Setjum nú $\phi = (p-1)(q-1)$, veljum e ósamþátta ϕ og d sem andhverfu e mátað við ϕ , þ.e. $e \cdot d \equiv 1 \ (\boldsymbol{mod}\ \phi)$.

2 Wikipedia - RSA-200 <http://en.wikipedia.org/wiki/RSA-200>

3 RSA Security - RSA-200 is factored! <http://www.rsasecurity.com/rsalabs/node.asp?id=2879>

Talnaparið (n, e) er þá almenningslykillinn og (n, d) einkalykillinn. Einkalykillinn (n, d) helst leyndur jafnvel þótt (n, e) sé opinberaður vegna flækjunnar við að þátta n.

Skilgreinum nú M sem skilaboðin sem skal dulkóða, E sem dulkóðunaraðferðina, C sem dulkóðaða textann og D sem afkóðunaraðferðina. Þá getum við sett fram aðferðirnar á formlegan máta:

Dulkóðun:

$$C \equiv E(M) \equiv M^e (\mathbf{mod}\ n) \tag{3}$$

Afkóðun:

$$M \equiv D(C) \equiv M^d (\mathbf{mod}\ n) \tag{4}$$

3.1.2 NOTKUN

Aðferð er talin örugg en hefur þann ókost að um flóknar reikniaðgerðir er að ræða og hún er því nokkuð hægvirk. Þess vegna er algengt að nota RSA til að skiptast á samhverfum dulkóðunarlykli (t.d. AES) sem svo er notaður til að dulkóða samskiptin sjálf.

4 Afkóðun út frá almenningslykli

Þótt RSA sé talin nokkuð örygg dulkóðunaraðferð byggist hún, eins og áður var nefnt, á *erfiðleika* þess að þátta stórar heiltölur. Slíkt er ekki ógjörningur, heldur afar tímafrekt.

Öryggið felst því í stærð n, sem er yfirleitt 1024, 2048 eða jafnvel 4096 bita tala.

Hluti þessa verkefnis var að afkóða gefinn streng út frá gefnum almenningslykli. Almenningslykillinn (n, e) var 257 bita þannig að

$$n = 221873272483254945141922353186910788703120217007463592538956281140602541064561$$

og

$$e = 65537$$.

4.1 Þáttun n

Ýmsir möguleikar komu til greina hvað varðaði þáttun n í p og q. Til greina kom að skrifa forrit fyrir þáttunina en sú leið reyndist ekki ákjósanleg. Mikil fræði liggja að baki þáttun heiltalna og í ljósi þess að ekki lá fyllilega fyrir hversu langan tíma þáttunin myndi taka var sú hætta til staðar að ekki tækist að klára þáttunina á tilsettum tíma. Þess í stað var ákveðið að nota reikniforrit sem aðgengileg eru á netinu.

Nokkur forrit komu til greina, en ákveðið var að reyna að vinna verkið með frjálsum hugbúnaði og þá helst á linux. Octave, sem er frjálst reikniforrit í líkingu við Matlab, kom fyrst til greina. Slegnar voru inn nokkrar prufutölur til þáttunar, sem Octave réð við, en þegar þátta átti n gafst forritið upp og kvartaði undan minnisleysi.

```
maroon:/gogn/Documents/Nam/current/sit/10verkefni> octave
GNU Octave, version 2.1.69 (i686-pc-linux-gnu).
Copyright (C) 2005 John W. Eaton.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTIBILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type `warranty'.

Additional information about Octave is available at http://www.octave.org.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/help-wanted.html

Report bugs to <bug@octave.org> (but first, please read
http://www.octave.org/bugs.html to learn how to write a helpful report).

octave:1> factor(221873272483254945141922353186910788703120217007463 59\
   25389562811406025410645 61)
error: memory exhausted -- trying to return to prompt
octave:1>
```

Ýmsar leiðir voru reyndar til að gefa Octave meira minni en án árangurs. Virtist vera um takmörkun af hálfu forritsins að ræða en ekki tölvunnar þar sem Octave gafst strax upp, án þess að taka til sín minni. Engu breytti þótt sýndarminni væri bætt við vélina. Að vísu gæti verið að Octave hafi beðið um meira minni en 32 bita örgjörvinn ráði við, en það er talið ólíklegt. Ekki tókst að finna neinar upplýsingar með Google um lausn þessa vandamáls. Ætla má að Octave ráði við verkefni sem þetta, en engu að síður var ákveðið að leita annarra leiða.

Ýmis önnur forrit voru reynd til þáttunar; aðferðir skrifaðar í C++, Java og bc[4]. Allt kom þó fyrir ekki því þau ýmist gáfust upp, voru hönnuð fyrir ofurtölvuklasa eða virkuðu einungis á 64 bita örgjörvum. Lokst var reynt að nota PARI fallasafnið og gp reiknivélina sem nýtir sér það (http://pari.math.u-bordeaux.fr/). PARI notar samsafn algríma við þáttunina; SQUFOF, Pollard Rho, ECM og MPQS. Óljóst er hvernig PARI metur hvaða algrím skal nota í hvaða tilvikum, en út frá úttaki forritsins má ætla að MPQS algrímurinn hafi verið notaður. Merkilegt er að PARI skyldi ekki nýta sér GNFS að neinu leyti, en það er almennt talinn vera hraðvirkasti algrímurinn til þáttunar.

4 Bc er frjáls reiknivél sem fylgir með flestum UNIX og Linux dreifingum

```
maroon:/gogn/Documents/Nam/current/sit/10verkefni>   time   echo   "factor
(22187327248325494514192235318691078870312021700746359253895628114060254106
4561)" | gp
                  GP/PARI CALCULATOR Version 2.1.6 (released)
                i686 running linux (ix86 kernel) 32-bit version
                (readline v5.0 enabled, extended help available)

                       Copyright (C) 2002 The PARI Group

PARI/GP is free software, covered by the GNU General Public License, and
comes WITHOUT ANY WARRANTY WHATSOEVER.

Type ? for help, \q to quit.
Type ?12 for how to get moral (and possibly technical) support.

   realprecision = 28 significant digits
   seriesprecision = 16 significant terms
   format = g0.28
s
parisize = 4000000, primelimit = 500000
***    Warning: MPQS: the factorization of this number will take several
hours.
%1 =
[34830925363169860816412549967967105258 1 1]

[63700079791122410320223306087272681558 1 1]

Good bye!

real    46m12.749s
user    45m45.412s
sys     0m1.945s
```

Vinnslan tók merkilega skamman tíma - rétt rúm þrjú korter. Hinar aðferðirnar höfðu keyrt lengur áður en þær gáfust upp og voru töluvert minnisfrekari en PARI sem notaði aðeins um 3 megabæti. Tölvan sem notuð var í þáttunina býr yfir Intel Pentium M 1600MHz örgjörva með eins megabæta skyndiminni og hafði 512 MB í vinnsluminni. Hún keyrir linux kjarna af útgáfu 2.6.10-ck.

4.2 d fundið

Setjum nú $p = 34830925363169860816412549967967105258 1$ og $q = 63700079791122410320223306087272681558 1$. Nú höfum við viðkvæmustu upplýsingarnar í höndum og þurfum einungis að finna $d = e^{-1} \bmod \phi$.

Væri um lága tölu d að ræða mætti skrifa lítið forrit sem nýtti sér formúluna $d = \frac{(1 + i \cdot \phi)}{e}$ og ítra með hækkandi $i \in \mathbb{Z}$ þar til d yrði heil tala. Þetta er þó mjög seinleg aðferð og mun hraðvirkara er að nota reiknirit Evklíðs (Extended Euclidean algorithm).

Á ný var notuð vinna annarra og nú notuð tilbúin föll, í þessu tilviki í Java. Spyrja mætti hvort hér sé fallið frá hinu upprunalega markmiði um að nota einungis frjálsan hugbúnað, enda Java ekki frjáls hugbúnaður. Á móti kemur að Java er algengt, gjaldfrjálst forritunarmál og samþykkt til notkunar af GNU.

Eftirfarandi forrit var keyrt:

```
// Hleð inn BigInt skilgreiningum
import java.math.BigInteger;

public class rsa {
    public static void main(String[] args){
        // Skilgreini breytur
        BigInteger p, q, n, d, e, phi;

          // Set gildi P og Q skv. útkomu gp
        p = new BigInteger("34830925363169860816412549967967105258 1");
        q = new BigInteger("63700079791122410320223306087272681558 1");

        // Reikna n=p*q og phi=(p-1)*(q-1)
        n = p.multiply(q);
        phi = (p.subtract(BigInteger.ONE)).multiply(
                  q.subtract(BigInteger.ONE));

        // Set inn gildi e (gefið) og reikna d
        e = new BigInteger("65537");
        d = e.modInverse(phi);

        // Skrifa út gildi breytanna
        System.out.println("p   = "+ p);
        System.out.println("q   = "+ q);
        System.out.println("n   = "+ n);
        System.out.println("phi = "+ phi);
        System.out.println("e   = "+ e);
        System.out.println("d   = "+ d);
    }
}
```

Út kom:

```
p   = 34830925363169860816412549967967105258 1
q   = 63700079791122410320223306087272681558 1
n   = 22187327248325494514192235318691078870312021700746359253895628114\
      0602541064561
phi = 22187327248325494514192235318691078870213490695592066982758992258\
      0050143196400
e   = 65537
d   = 16886703436936015782960902966207043563883743776738823887270069332\
      274747307073
```

Til að sannreyna að um rétt gildi væri að ræða var prófað að dulkóða og afkóða töluna 1234 á eftirfarandi hátt, aftur með Java:

```
//Hleð inn BigInt skilgreiningum
import java.math.BigInteger;

public class encodedecode {
    public static void main(String[] args){
    // Skilgreini breytur
    BigInteger p, q, n, d, e, phi, m, md, c;

    // Set gildi P og Q skv. útkomu gp
    p = new BigInteger("34830925363169860816412549967967105258 1");
    q = new BigInteger("63700079791122410320223306087272681558 1");

    // Reikna n=p*q og phi=(p-1)*(q-1)
    n = p.multiply(q);
    phi = (p.subtract(BigInteger.ONE)).multiply(
                q.subtract(BigInteger.ONE));

    // Set inn gildi e (gefið) og reikna d
    e = new BigInteger("65537");
    d = e.modInverse(phi);

    // Dulkóða 1234
    c = (new BigInteger("1234")).modPow(e, n);

    // Afkóða c
    m = c.modPow(d, n);

    // Skrifa út niðurstöður
    System.out.println("m = "+ m);
    }
}
```

Keyrt og út kom:

```
m = 1234
```

svo talan d var rétt til fundin.

4.3 Afkóðun strengsins

Strengurinn var gefinn í tveimur hlutum og því ljóst að lengd upprunalegu skilaboðana væri einhversstaðar á milli 33 og 64 stafir. Það mátti sjá með því að skoða lengd n , en þar sem hún var rétt rúm 32 bæti mátti kóða 32 stafi. Þ.e. stafur fyrir hvert heilt bæti.

Afkóðunina var síðan framkvæmd með eftirfarandi Java kóða:

```
//Hleð inn BigInt skilgreiningum
import java.math.BigInteger;

public class decode {

    public static void main(String[] args){
        // Skilgreini breytur
        int i, iStafur;
        BigInteger n, d, m1, m2;
        String c1, c2, Mhex, Mstr, sStafurHex;

        // Set inn dulkóðuðu boðin
        c1 = "00E68D5EDF3416FF31D5FC22D52E2426DC5141873385D418C7AA7E136"+
             "83D23332F";
        c2 = "017ACD735513A85617B44C69EA00DD887054C408AF6E02CEDA268ED44"+
             "E7E3A91CB";

        // Skilgreini n og d
        n = new BigInteger("221873272483254945141922353186910788703"+
             "120217007463592538956281140602541064561");
        d = new BigInteger("168867034369360157829609029662070435638"+
             "837437767388238872700693322747473070 73");

        // Afkóða skilaboðin
        m1 = (new BigInteger(c1, 16)).modPow(d, n);
        m2 = (new BigInteger(c2, 16)).modPow(d, n);

        // Skeyti þeim saman í Mhex og gef Mstr upphafsgildi
        Mhex = m1.toString(16) + m2.toString(16);
        Mstr = "";
```

```
        // Breyti út HEX yfir í streng
        for (i=0; i<Mhex.length(); i+=2){
            // Tek eina tölu í einu
            sStafurHex = Mhex.substring(i, i+2);

            // Umbreyti henni úr 16base yfir í 10base
            iStafur = Integer.parseInt(sStafurHex, 16);

            // Og svo loks í staf, og bæti honum við Mstr
            Mstr += String.valueOf((char)iStafur);
        }

        // Skrifa út afkóðaða textann, M
        System.out.println("Lausn: '"+ Mstr +"'");
    }
}
```

Úr þessu má svo fá upphaf 4. málsgreinar 25. kafla Brennu-Njálssögu:

```
Lausn: 'Nú skal nefna sonu Njáls. Skarphéðinn hét hinn elsti.          '
```

Ef til vill má um það deila hvort hér sé um 55 eða 64 stafa streng að ræða, því bilin aftast eru óneitanlega hluti strengsins. Hér verður þetta skilgreint sem 64 stafa strengur þótt gera megi ráð fyrir að hugmyndir kennara séu nokkuð aðrar.

5 Niðurstöður

Þátta tókst lykilinn með frjálsum hugbúnaði á tilsettum tíma. Eftir að réttu tólin voru fundin tók framkvæmdin jafnframt mun skemmri tíma en í fyrstu var við búist, eða rúm þrjú korter á 1,6Ghz örgjörva. Þegar n hafði verið þáttað í frumþættina p og q mátti reikna út þær stærðir sem upp á vantaði og afkóða strenginn sem gefinn var á skotstundu.

Það kom að nokkru leyti á óvart hversu auðvelt reyndist að afkóða eftir að þáttun var lokið. Velta má því fyrir sér; hvað ef á morgun finnst aðferð til að þátta stórar heiltölur á skemmri tíma? Einstaka furðufugl skýtur reglulega upp kollinum og heldur því fram að hann hafi leyst RSA vandamálið, en flestir fræðimenn eru sammála um að ekki finnist betri aðferð án notkunar skammtatölva. Með slíkri tölvu mætti nota Shor's algríminn og væri þá vandamálið af margliðustigi $\Theta((\log N)^3)$. Skammtatölvur eru þó enn allt of skammt á veg komnar til alvöru reikninga sem þessa og umfjöllun um þær efni í ritgerð út af fyrir sig.

6 Heimildir

1. Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. 1978. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.* http://theory.lcs.mit.edu/~rivest/rsapaper.pdf
2. "Caesar cipher". 20. október 2005. Wikipedia, the free encyclopedia - http://en.wikipedia.org/wiki/Caesar_cipher
3. "Cryptography". 23. október 2005. Wikipedia, the free encyclopedia - http://en.wikipedia.org/wiki/Cryptography
4. "Diffie-Hellman key exchange". 18. október 2005. Wikipedia, the free encyclopedia - http://en.wikipedia.org/wiki/Diffie_Hellman
5. "Encryption". 20. október 2005. Wikipedia, the free encyclopedia - http://en.wikipedia.org/wiki/Encryption
6. "Integer factorization". 24. október 2005. Wikipedia, the free encyclopedia - http://en.wikipedia.org/wiki/Integer_factorization
7. "General number field sieve". 4. september 2005. Wikipedia, the free encyclopedia - http://en.wikipedia.org/wiki/General_number_field_sieve
8. "RSA problem". 15. apríl 2005. Wikipedia, the free encyclopedia - http://en.wikipedia.org/wiki/RSA_problem
9. "RSA". 28. október 2005. Wikipedia, the free encyclopedia - http://en.wikipedia.org/wiki/RSA
10. "RSA-200". 16. október 2005. Wikipedia, the free encyclopedia - http://en.wikipedia.org/wiki/RSA-200
11. "Symmetric key algorithm". 24. október 2005. Wikipedia, the free encyclopedia - http://en.wikipedia.org/wiki/Symmetric_algorithm
12. "Shor's algorithm". 28. október 2005. Wikipedia, the free encyclopedia - http://en.wikipedia.org/wiki/Shor%27s_algorithm
13. "Public-key cryptography". 28. október 2005. Wikipedia, the free encyclopedia - http://en.wikipedia.org/wiki/Public_key_cryptography
14. "The RSA Challenge Numbers". RSA Security - http://www.rsasecurity.com/rsalabs/node.asp?id=2093
15. "RSA-200 is factored!". RSA Security - http://www.rsasecurity.com/rsalabs/node.asp?id=2879
16. Paul Andrew Johnston, 19. júní 2004. "RSA Algorithm" - http://pajhome.org.uk/crypt/rsa/rsa.html
17. Rosen, Kenneth H. 2002. *Discrete Mathematics and Its Applications*, 5. útgáfa, bls 191 - 194. McGraw-Hill Education.