HENKOS Encryption System (HKS)

Lect. Dominic BUCERZAN, Universitatea „Aurel Vlaicu" Arad
Marius GHEORGHIȚĂ, Institutul Național de Meteorologie

The purpose of this paper is to recommend to cryptographic community and information security specialists, for analysis and testing, a new cryptosystem based on a synchronous stream cipher and a performant keystream generator. This pseudorandom number generator is the main part of the proposed stream cipher and use two keys to produce more diffusion and confusion în the generated keystrem.
The paper describes the main parts of the cryptosystem, its implementation and analysis of the statistical tests results for the keystream generator. Through its design and conception, HENKOS algorithm could be a new approach în the symmetrical encryption system evolution.
Keywords*: synchronous stream cipher, pseudorandom number generator (PRNG), master key (MK), data key (DK), HENKOS cryptosystem (HKS), statistical tests.*

Introducere

Scopul acestei lucrări este de a prezenta un nou algoritm criptografic denumit HENKOS (HKS) şi în mod deosebit generatorul de numere pseudo aleatoare pe care se bazează. Algoritmul a fost conceput pentru a răspunde la următoarele obiective: să nu aibă cicluri scurte; să fie sigur din punct de vedere criptografic; să fie uşor de implementat; codul C/C++ să fie optimizat pentru viteză; codul ASM să fie optimizat pentru viteză; să creeze cât mai multă confuzie şi difuziune.
Noul sistem criptografic este bazat pe un algoritm simetric de tip şir şi foloseşte două chei: o cheie master (MK) şi o cheie de date (DK); cheia master este o cheie unică secretă iar cheia de date este o cheie autogenerabilă pentru fiecare sesiune; iniţial expeditorul mesajului cifrat şi destinatarul îşi fac cunoscute cele două chei într-o formă care să excludă deconspirarea cheilor. În fiecare nouă sesiune generatorul HENKOS va folosi cheia master şi ultima cheie de date generată, pentru a produce o nouă cheie de date împreună cu şirul de chei cifrat din care printr-un XOR cu textul în clar va rezulta mesajul cifrat care va fi expediat către destinatar.
Într-un sistem simetric clasic cei doi pot comunica folosind o cheie secretă dar apare problema retransmiterii cheilor care se schimbă de la o sesiune la alta. Rezultate obţinute:
- un algoritm uşor de implementat;
- un algoritm "sigur" din punct de vedere criptografic (testele statistice demonstrează aceasta propoziţie);
- un algoritm extrem de "rapid": un fişier de 6 MB este criptat în mai puţin de o secundă;
- un generator de numere pseudoaleatoare foarte performant: generarea unei chei de 12.500.000 de octeţi durează doar 0,64 secunde (C,C++) respectiv 0,33 secunde (ASM).

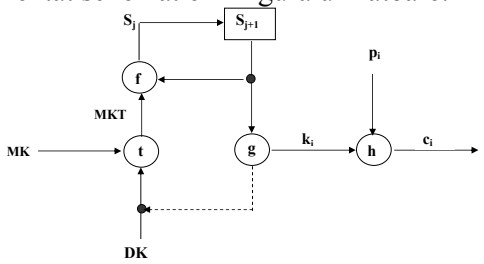
Descrierea sistemului

Acest sistem criptografic foloseşte un algoritm binar aditiv de tip şir şi două tipuri de chei:

- O cheie pe termen scurt denumita **cheie de date (DK)** cu o lungime fixă de 1024 octeţi care este una din intrările din generatorul de chei de criptare. Aceasta cheie poate fi generată cu un algoritm pseudoaleator sau poate fi ales chiar un fişier oarecare din sistemul de operare; DK este folosită doar o dată în prima sesiune.
- O cheie pe termen lung denumita **cheie master (MK)** care conţine un şir de 1024 de numere (cheia secreta) folosită pentru a schimba cheia de date şi starea internă a generatorului. Această cheie trebuie aleasă cu grijă, generată eventual cu un generator de numere aleatoare şi transmisă între părţi doar în condiţii de siguranţă totală; numărul posibilităţilor de combinaţii pentru această cheie poate fi parcurs de un sistem cu o putere de

calcul de 10^{24} operaţii /an în 10^{4072} ani. Dacă în cursul transmisiei un atacator interceptează mesajul, el nu poate fi descifrat fără cheia master. Fiecare încercare de a simula cheia master dă rezultate diferite; chiar folosirea aceloraşi numere din cheie nu dă rezultate încurajatoare pentru că inclusiv ordinea lor este decisivă (existând 1024! posibilităţi de aranjare a lor în şir). Chiar şi în cazul în care acelaşi mesaj este trimis de mai multe ori rezultatul criptat este diferit pentru că de fiecare dată se foloseşte altă cheie de date; în consecinţă atacurile bazate pe analize statistice au şanse minime de succes.

Algoritmul generat de funcţiile t,f,g,h este prezentat schematic în figura următoare:



p_i – textul clar
$S_0 = \mathbf{t}$ (MK, DK)
k_i – cheile de cifrare
$S_{j+1} = \mathbf{f}$ (S_j, MKT)
c_i – mesajul cifrat
$k_i = \mathbf{g}$ (S_i, S_{i-1})
S_j – starea internă *j*
$c_i = \mathbf{h}$ (p_i, k_i)
MKT – cheia MK transformată

Algoritmul poate fi împărţit în patru părţi:
- generarea cheii master
- generarea cheii de date
- generarea şirului de chei de criptare
- procesul de criptare/decriptare

Generarea cheii master (MK)

În această secţiune a algoritmului are loc o transformare a cheii MK într-o cheie MKT în doi paşi: pentru aceasta am definit două funcţii SUM şi INV, prima fiind o funcţie aditivă iar cea de-a doua definind o transformare de simetrie a numerelor:

Notaţie: MK_i este al i-lea element în cheia MK .

$$MKS_i = SUM(MK_i) \qquad i = 0,1023 \qquad \text{unde} \qquad (1)$$

$$SUM(MK_i) = \sum_{j=0}^{i} MK_j \qquad \text{modulo } 1024 \quad i = 0,1023$$

În final MKT se obţine astfel:

$$MKT_i = INV(SUM(MK_i)), \quad i = 0,1023 \qquad (2)$$

Transformările duc la următoarele consecinţe:

- Cheia originală MK nu este folosită direct în proces
- Se creează confuzie şi difuzie pentru cheia master

Generarea cheii de date (DK)

În această secţiune a algoritmului au loc transformări succesive a cheii DK pentru a obţine în final cheia reală (K) de cifrare; pentru aceste transformări se folosesc două funcţii: o funcţie de comutare (SW) şi o funcţie aditivă (AD) descrise în continuare:

$$(SW): \quad DK_j \leftrightarrow DK_k \text{ unde j,k depind de MK şi MKT} \qquad i = 0,1023 \qquad (3)$$

$$(AD): \quad DK_i = DK_i + DK_{i+1} \text{ modulo } 256 \qquad i = 0,1023 \qquad (4)$$

Aceste funcţii vor crea o imagine total schimbată a cheii DK. După cele două transformări se obţine DK1; acest ciclu se va repeta de 64 de ori:

$$DK \rightarrow DK1 \rightarrow DK2 \rightarrow \ldots. \rightarrow DK63 \rightarrow DK64 \qquad (5)$$

Generarea şirului de chei de cifrare

Pentru a obţine octeţii (K_i) cheii finale de cifrare (K), ultima operaţiune va fi următoarea:

$$K_i = (DK64_i + DK64_{i+1}) \oplus DK64_i \qquad i = 0,1023;\ DK64_{1024} = DK64_0 \qquad (6)$$

$\oplus$ = "SAU exclusiv"

Procesul de criptare/decriptare

Funcția de criptare este cea obișnuită pentru această categorie de algoritmi (SAU exclusiv):

$c_i = h(K_i, p_i)$, c_i = textul cifrat, p_i = textul clar, K_i = cheia de cifrare.

Algoritmul fiind prin definiție simetric, procesul de decriptare va folosi aceleaşi funcții şi aceeaşi parametrii.

Procesul de criptare/decriptare va transforma un şir de 1024 octeți de text clar; pentru următorul şir de 1024 octeți se va folosi o altă cheie DK generată din DK64 definit în (5); această secvență va continua până la epuizarea textului de cifrat pentru o sesiune de lucru.

Observații

1. Confuzia şi difuzia biților va fi datorată în special funcțiilor definite la (3) şi (4).

2. Cheia master am generat-o cu acelaşi algoritm folosind o secvență dintr-un fişier oarecare; testele folosite au dat rezultate foarte bune pentru cheia obținută.

3. Cheia de date din sesiuni succesive a fost generată cu acelaşi algoritm cu rezultate excelente (a se vedea testele)

4. Un punct vulnerabil ar fi generat de folosirea unei chei master care conține un şir lung de "0"; pornind de la o cheie fără această slăbiciune am generat un şir lung de chei pentru care testele au dat rezultate foarte bune.

5. În această lucrare se tratează în special algoritmul de generare a numerelor pseudoaleatoare şi mai puțin probleme conexe cum ar fi sincronizarea cheilor între sesiuni; acest subiect şi altele vor fi tratate ulterior

Implementare

Sistemul criptografic HENKOS a fost implementat atât în C/C++ cât şi în ASM pentru Windows/32 biți. Toate aplicațiile şi testele asupra fişierelor generate au fost făcute pe un PC Pentium IV 2,4 GHz/ 256 MB RAM. Pentru comparare am ales două generatoare de numere pseudoaleatoare 'clasice': SHA-1, unul dintre cele mai renumite şi CCG (Cubic Congruential Generator), un exemplu de PRNG 'slab' complet predictibil. Pe parcursul testelor a fost monitorizată o secvență de 12,5 MB.

Testele de viteză au relevat rezultate excelente în comparație cu alte generatoare :

- o medie de 0,64 secunde pentru varianta C
- o medie de 0,33 secunde pentru varianta ASM

Aceste rezultate au fost obținute pentru o lungime a cheilor de 1024 octeți; pentru chei mai lungi acest timp scade.

Analiza testelor statistice a demonstrat că HENKOS este un generator veritabil de numere pseudoaleatoare; avantajul față de alte generatoare, viteza sa net superioară îl recomandă pentru folosirea în sisteme criptografice.

Teste de analiză statistică

Generatorul de chei este un algoritm nou şi rapid; el acceptă chei de lungime mare la inițializare şi trece cu succes majoritatea testelor statistice importante. Aceste teste (FIPS 1401, FIPS 1402, testele ENT, bateria de teste DIEHARD, suita de teste statistice NIST – o suită de teste statistice utilizată pentru testarea generatoarelor de numere pseudoaleatoare folosite în aplicațiile criptografice) au fost efectuate pentru eşantioane de text cifrat de mari dimensiuni - 12.5 MB. Rezultatele testelor statistice sunt prezentate în continuare:

1. Testele FIPS 1401/FIPS1402

Testele statistice FIPS conțin testul monobit, testul poker, testul serial şi testul serial lung. Următoarele teste se bazează pe testul statistic de succes/insucces executat pe 5000 de şiruri de 2500 de octeți fiecare produse de către generatorul de numere pseudoaleatoare.

Generatoarele SHA-1, CCG şi HKS trec cu succes testul FIPS 140-1 în proporție de 100%.

SHA-1 trece FIPS 140-2 în proporție de 99.6%, CCG în proporție de 99.4% iar HENKOS în proporție de 99.64%.

Pentru aceste teste statistice, chiar dacă generatoarele prezintă rezultate statistice bune, nu există garanția că sunt recomandate pentru a fi folosite în scopuri criptografice (vezi rezultele de la testele DIEHARD).

2. Testele ENT

Testele ENT se aplică unor şiruri de octeți organizate sub formă de fişiere şi raportează

rezultatele obținute, fiind folosite pentru evaluarea generatoarelor de numere pseudoaleatoare utilizate în criptare. Se calculează entropia, compresia optimă, distribuția chi-pătrat, media aritmetică, valoarea Monte Carlo pentru constanta pi şi coeficientul de corelație serială.

	SHA-1	**CCG**	**HENKOS**
Entropia	7.999988 biți per octet	7.997488 biți per octet	7.999987 biți per octet
Compresia optimă reduce dimensiunea fişierului de 12500000 octeți cu un procent de	0	0	0
Distribuția hi-pătrat pentru un eşantion de 12500000 octeți este	201.61	46584.63	222.04
aleator se va depăşi această valoare în următoarea proporție (% numărului de cazuri)	99.00	0.01	90.00
Media aritmetică este	127.5123 (127.5 = aleator)	127.4921 (127.5 = aleator)	127.5179 (127.5 = aleator).
Valoarea Monte Carlo pentru constanta pi este	3.141492983 (procent de eroare 0.00)	3.141199828 (procent de eroare 0.01)	3.141108983 (procent de eroare 0.02)
Coeficientul de corelație serială este	0.000406 (total necorelat = 0.0)	0.000266 (total necorelat = 0.0)	0.000124 (total necorelat = 0.0)

3. Testele statistice DIEHARD

Următorul set de teste a fost proiectat pentru a identifica slăbiciunile existente în majoritatea algoritmilor non-criptografici folosiți în generatoarele de numere pseudoaleatoare. Aceste teste analizează un unic fişier de mari dimensiuni (peste 11 MB) care conține date generate.

Bateria de teste include o sumă de teste dintre care menționăm: testul de şir de biți, testul de permutări cu suprapuneri, testele opso, oqso şi DNS, testul valorilor 1 din şirul de octeți etc.

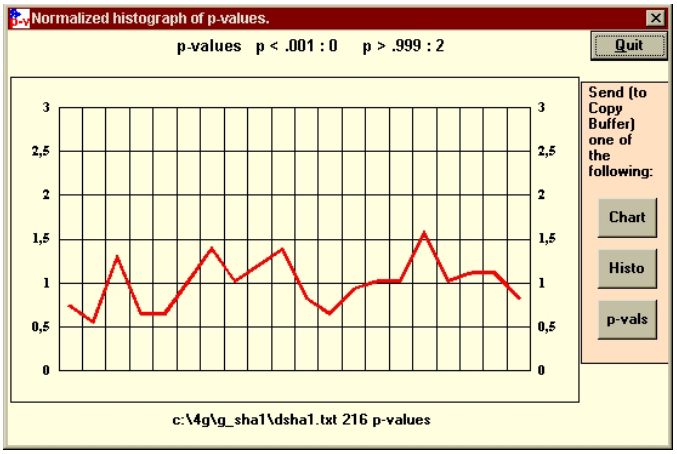


Fig. 1. Interpretarea valorilor *p* pentru generatorul SHA-1

Majoritatea testelor din DIEHARD generează o valoare *p*, care se distribuie uniform în intervalul [0,1), dacă fişierul de intrare conține biți generați aleator. Aceste valori *p* sunt obținute prin *p*=F(X), unde F este funcția de distribuție atribuită variabilei aleatoare de eşantion X. Un şir de biți se consideră că nu trece testul dacă produce mai mult de 6 valori *p* de 0 sau 1.

Pentru generatorul SHA-1 avem 2 valori *p* apropiate de 1, iar pentru generatorul CC avem 28 de astfel de valori *p*. Pentru genera-

torul HENKOS nu avem valori *p* apropiate de 0 sau 1. Reprezentarea grafică a valorilor *p* ideale este o curbă cât mai plată. Observând graficele corespunzătoare, se observă ca CC este un generator prost, iar HENKOS prezintă o curbă bună, care poate fi comparată cu cea generată de SHA-1, un generator cunoscut ca având o calitate criptografică foarte bună. Putem măsura calitatea generatoarelor folosind schema descrisă în [7].

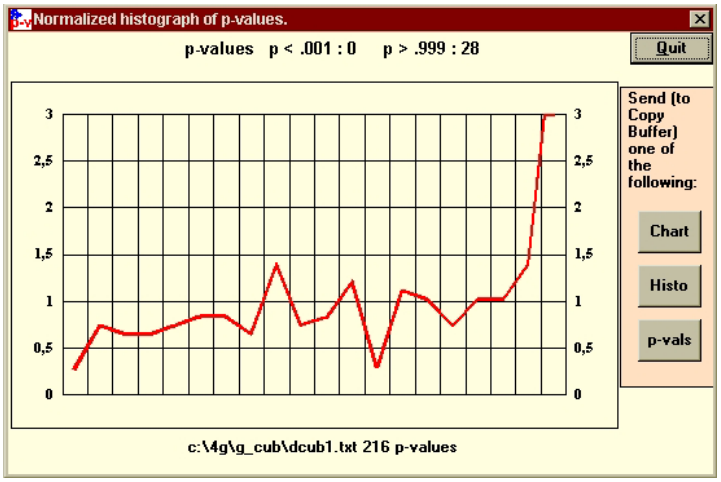


Fig. 2. Interpretarea valorilor *p* pentru generatorul CC

Fiecare test DIEHARD produce una sau mai multe valori *p* (216 în total), care pot fi considerate bune, proaste sau suspecte:

- valoarea $p > 0.998$ este considerată ca proastă şi notată cu valoarea 4
- valoarea p între 0.95 şi 0.998 este considerată ca suspectă şi notată cu valoarea 2
- valoarea $p < 0.995$ este considerata ca bună şi notată cu valoarea 0

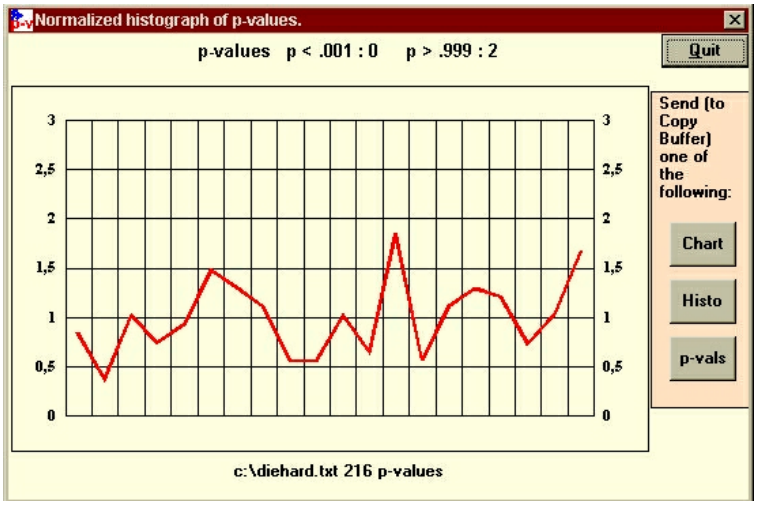


Fig. 3. Interpretarea valorilor *p* pentru generatorul HENKOS

Pentru fiecare generator se calculează valoarea totală prin însumare. Valorile mari indică un generator de numere pseudoaleator slab, iar valorile mici caracterizează un generator de calitate. Procentul se obţine raportând valoarea obţinută la numărul total de valori *p* – 216. Rezultatele pentru SHA-1, CCG şi HENKOS (HKS) sunt prezentate în tabelul următor. Pentru HENKOS se prezintă 6 eşantioane de date.

În concluzie valoarea medie pentru HKS este mai bună decât toate celelalte valori (chiar şi cea mai slabă valoare pentru HKS este mai bună decât valorile calculate pentru celelalte).

Eşantioane	Valoare totală	% faţă de cele 216 valori *p*
SHA-1	**31**	**0.1759**
CCG	**154**	**0.7129**
HKS-1	16	0.0740
HKS-2	10	0.0462
HKS-3	16	0.0740
HKS-4	24	0.1111
HKS-5	22	0.1018
HKS-6	30	0.1388
HKS mediu	**19.66**	**0.0910**

4. Suita de teste statistice NIST

Acest pachet de teste include teste statistice pentru frecvenţă, frecvenţa blocurilor, sume cumulative, serii, serii lungi, entropia aproximativă, complexitatea Lempel-Ziv etc. În următorul tabel sunt prezentate rezultatele obţinute de generatoare.

Ca multe alte teste, şi NIST se bazează pe testarea ipotezelor:

1. Se formulează ipoteza nula. Se presupune că şirul binar generat este aleator.
2. Se obţine o secvenţă de test. Testarea se execută la nivel de bit.
3. Se calculează valoarea p din intervalul [0, 1].
4. Se compară valoare p cu pragul α, unde $\alpha =. 0.01$. Testul este trecut cu succes dacă valoarea p este mai mare decât pragul α.

TEST	SHA-1	CCG	3DES	BBS	MICALI	HKS
Frecvenţă	Pass	Failure	Pass	Pass	Pass	Pass
Frecvenţă de bloc	Pass	Pass	Pass	Pass	Pass	Pass
Custom	Pass	Failure	Pass	Pass	Pass	Pass
Serii	Pass	Failure	Pass	Pass	Pass	Pass
Serii lungi	Pass	Pass	Pass	Pass	Pass	Pass
Rang	Pass	Pass	Pass	Pass	Pass	Pass
FFT	Pass	Failure	Pass	Pass	Pass	Pass
Neperiodic	Pass	Failure 11	Failure 3	Pass	Failure 2	Pass
Periodic	Pass	Pass	Pass	Pass	Pass	Pass
Universal	Pass	Pass	Pass	Pass	Pass	Pass
Apen	Pass	Failure	Pass	Pass	Pass	Pass
Parcurs aleator	Pass	Pass	Pass	Pass	Pass	Pass
Parcurs aleator –V	Failure 1	Pass	Failure 4	Pass	Pass	Pass
Serial	Pass	Failure1	Pass	Pass	Pass	Pass
	Pass	Pass	Pass	Pass	Pass	Pass
Lempel-Ziv	Pass	Failure	Pass	Pass	Pass	Pass
Complexitate lineară	Pass	Pass	Pass	Pass	Pass	Pass
Rata minimă de succes (parcurs aleator–V)	.95239	.94598	.95353	.95146	.95406	.95178
Rata minimă de succes	.96015	.96015	.96015	.96015	.96015	.96015

Din acest tabel rezultă că doar HENKOS şi BBS au trecut cu succes toate testele.

Concluzii

În această lucrare am prezentat un nou sistem criptografic denumit HENKOS (HKS) bazat

pe un algoritm generator de numere pseudoaleatoare care s-a dovedit a fi foarte rapid în implementare şi care a trecut cu rezultate excelente testele statistice folosite. Considerăm acest nou generator util pentru simulări şi pentru uz criptografic şi aşteptăm de la specialiştii în domeniu să-i găsească puncte slabe, ajutându-ne astfel în dezvoltările ulterioare.

Bibliografie

[1] A. Menezes, P. van Oorschot, and S. Vanstone, Handbook of Applied Cryptography CRC Press, 1996. capitolul 5,7,12,13 www.cacr.math.uwaterloo.ca/hac

[2] B. Schneier Applied Cryptography, Second Edition, J. Wiley & Sons Inc. 1996

[3] G. Marsaglia, Diehard Statistical Tests. http://stat.fsu.edu/~geo/

[4] J.Kelsey, B.Schneier, D.Wagner, C. Hall,"Cryptanalytic Attacks on Pseudorandom Number Generators" *Fast Software Encryption, Fifth International Workshop Proceedings (March 1998)*, Springer-Verlag, 1998, http://www.counterpane.com/pseudorandom_number.html

[5] Johnson B.C. Radix-b extensions to some common empirical tests for pseudorandom number generators. ACM Transactions on Modelling and Computer Simulations, S(4):261-273, 1996

[6] Mark M. Meysenburg,James A. Foster, "Random generators quality and GP performance"
Proceedings of the genetic and evolutionary computation conference, vol. 2, pages 1121-1126, Orlando, Florida 13-17 July 1999

[7] Peter Gutmann, "Software Generation of Practically Strong Random Numbers" (updated June 2000), Original version presented at the 1998 Usenix Security Symposium. http://www.cryptoengines.com/~peter/06_random.pdf

[8] P. Martin, "An Analysis of Random Number Generator for a Hardware Implementation of Genetic Programming using FPGAs and Handel-C 2002 www.celoxica.com/techlib/files/CEL-W0307171J2F-23.pdf

[9] A Statistical Test Suite for Random and PseudoRandom Number Generators for Cryptographic Application, NIST Special Publication 800-22 (with revision May 15, 2001) http://csrc.nist.gov/rng/

[10] ENT- A Pseudorandom Number Sequence Test Program www.fourmilab.ch/

[11] FIPS PUB 140-1, Security Requirements for Cryptographic Modules http://csrc.nist.gov/cryptval/140-1.htm

[12] FIPS PUB 140-2, Security Requirements for Cryptographic Modules http://csrc.nist.gov/cryptval/140-2.htm