Avaliação do custo/benefício da adoção de autômatos celulares na geração de números pseudoaleatórios no Linux

Sílvia Regina Leite Magossi, Marco Aurélio Amaral Henriques

Faculdade de Engenharia Elétrica e de Computação – FEEC

Universidade Estadual de Campinas – Unicamp

[srlm,marco]@dca.fee.unicamp.br

Resumo. *Este trabalho propõe o uso de autômatos celulares na geração de números pseudoaleatórios no Linux. É proposta uma métrica de avaliação da qualidade das sequências binárias geradas e é calculado o custo de cada esquema. Os resultados mostram que algumas configurações de autômatos celulares podem substituir com vantagens o gerador padrão do Linux.*

Abstract*. This work proposes the use of cellular automata in the pseudorandom number generation in Linux. A new metric is proposed to evaluate the quality of generated bit sequences and the cost of each scheme is calculated. The results show that some configuration of cellular automata can replace the standard generator in Linux with advantages.*

1. Introdução

Um Autômato Celular (AC) consiste de um arranjo de células que podem ser organizadas em um espaço n-dimensional. Cada célula pode ter um número finito de estados, os quais variam com a aplicação de regras determinísticas em intervalos de tempo discretos. Apesar da simplicidade da sua construção, alguns ACs são capazes de produzir um comportamento bastante complexo, podendo ser usados como geradores de números pseudoaleatórios (PRNG), os quais têm grande importância em diversas aplicações, em particular nas de segurança.

Este trabalho avalia o uso de ACs como PRNG, por meio de uma nova métrica de consolidação de diversos testes estatísticos realizados segundo recomendações do NIST, [Bassham et al. 2010]. Além disso, ele faz uma comparação da razão custo/benefício entre o PRNG padrão do Linux, usando SHA-1 (Secure Hash Algorithm), e uma versão modificada usando autômatos celulares.

2. Trabalhos Relacionados

Stephen Wolfram foi o primeiro a propor o uso de ACs em aplicações de segurança e, em seu artigo [Wolfram (1986)], mostra que o AC com regra 30 apresenta aleatoriedade suficiente para ser usado como PRNG. Desde então outros trabalhos surgiram na mesma linha e apresentando várias alternativas e avaliando o potencial de emprego de ACs como PRNGs, os mais recentes como o de Girau [Girau and Vlassopoulos 2012] que propõe a evolução de um AC bidimensional utilizando regras aditivas elementares e também o trabalho de Shin [Shin, et.al. 2012], onde os autores apresentam um AC bidimensional baseado na evolução von Neumann com regras elementares (31, 59, 95, 127) e vizinhança não linear, mantendo a qualidade da aleatoriedade.

Apesar de existirem muitos artigos sobre o tema, e várias propostas apresentarem bons resultados estatísticos, todas elas utilizam ACs híbridos e não são úteis para criptografia como pode ser observado no trabalho de Guan e Tan em [Guan S.U. et. al. (2004)], que afirma que tais esquemas devem ser considerados apenas para aplicações com requisitos de segurança reduzidos, uma vez que a saída do AC pode ser vista como o estado interno do PRNG e, portanto, não deve ser exposta. Este trabalho focou em ACs uniformes e com uma coleta de bits tal que não se expõe o estado interno do autômato, o que os tornam opções seguras e viáveis para um PRNG.

3. Autômatos Celulares

3.1 Características básicas de um Autômato Celular Elementar

A estrutura unidimensional simples, representada na Fig. 1, mostra um AC binário com 16 células. Designaremos os parâmetros k como o número de estados por célula e r (raio) como distância da célula central até sua vizinha mais distante. Os ACs com configuração k = 2 e r = 1 são ditos ACs elementares e usam 2r + 1 = 3 células binárias em suas regras de evolução (vizinhança três). Desse modo eles têm $k^{2r+1}= 8$ possíveis padrões, apresentados na Fig. 2. Nesta figura, as células que aparecem isoladas na linha inferior representam o estado futuro das células centrais.

Fig.1 - Autômato celular binário com 16 células

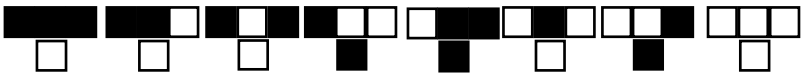

Fig. 2 – Uma regra de evolução para os 8 padrões possíveis em um AC binário de raio 1

A partir destes oito padrões possíveis, temos 2^8 possíveis regras de evolução para o futuro da célula central. Tomando os estados futuros da célula central de forma ordenada, podemos interpretar os bits do estado como um valor decimal, o qual é chamado de número da regra. Na Fig. 2, podemos ver que os bits do estado futuro representam o valor decimal 30. Esta configuração é conhecida como Regra 30 e é uma das mais conhecidas regras para geração de números aleatórios.

As transições em um AC_{elem} podem ser formalizadas por meio da equação: $a_i^{(t+1)} = f[a_{i-1}^{(t)}, a_i^{(t)}, a_{i+1}^{(t)}]$, onde f representa a função de transição local, i mostra a posição de uma célula no arranjo do AC, t indica o instante de tempo e $a_i^{(t)}$ denota o estado de saída da i-ésima célula no t-ésimo passo de tempo. As correspondentes expressões para as regras 30 e 150 são: Regra 30 : $a_i^{(t+1)} = a_{i-1}^{(t)} \oplus (a_i^{(t)} \wedge a_{i+1}^{(t)})$ e Regra 150: $a_i^{(t+1)} = a_{i-1}^{(t)} \oplus a_i^{(t)} \oplus a_{i+1}^{(t)}$. Os operadores $\oplus$ e $\wedge$, denotam as operações lógicas de ou-exclusivo e ou-inclusivo, respectivamente. A Fig. 3 mostra a evolução de um AC_{elem} pela regra 30 e 150, partindo de um estado inicial com apenas uma célula igual a um, e com condição de fronteira periódica (as células dos extremos são consideradas vizinhas).

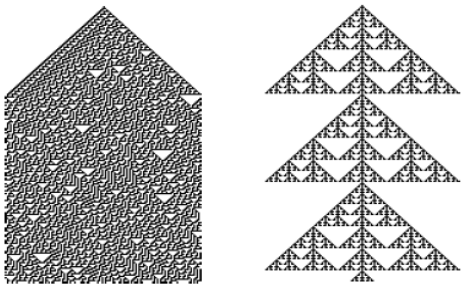

Fig. 3 - Evolução de ACs de 128 células pelas regras 30 e 150 (200 iterações)

3.2. Autômato Celular unidimensional de raio dois

Os ACs unidimensionais com configuração k = 2 e r = 2 usam 2r + 1 = 5 células binárias em suas regras de evolução e são chamados de ACs de vizinhança cinco. Desse modo existem k^{2r+1}= 32 possíveis padrões. A partir destes 32 padrões, tem-se um total de 2^{32} = 4.294.967.296 distintos mapeamentos referentes a todas as configurações de vizinhança para o próximo estado. A Regra 1436194405 unidimensional de raio 2 foi apresentada por Bouvry et al. como sendo caótica [Bouvry, P et al. (2003)] e, por este motivo, escolhida para avaliação neste trabalho.

3.3. Autômato Celular Bidimensional de Raio 1

Os ACs bidimensionais com configuração k = 2 e r = 2 usam 2r + 1 = 5 células binárias (vizinhança von Neumann) em suas regras de evolução e também existem k^{2r+1}= 32 possíveis padrões. Como ocorreu no AC unidimensional binário de raio 2, aqui também temos 2^{32} regras distintas. A regra 1453938345 apresentada em [Wolfram 2002] se mostrou suficientemente caótica e foi adotada em nossas avaliações. Entretanto, regras melhores podem existir e a busca de regras caóticas apropriadas para PRNG no universo de 2^{32} regras é ainda um desafio a ser vencido.

4. Utilização de autômatos celulares como PRNGs

Os valores das células de um AC em um determinado instante definem o estado deste AC. Logo, um AC de tamanho N pode assumir até 2^N estados distintos. Idealmente, todos estes estados deveriam formar um ciclo único, o que significa que uma repetição de estado só ocorreria após 2^N iterações. Na prática não é assim. O conjunto de estados pode se dividir em ciclos menores, o que é indesejável sob o ponto de vista de PRNG.

Este comportamento é flutuante, mas o tamanho dos ciclos menores vai crescendo com o aumento do tamanho do AC em alguns casos [Wolfram 1986]. Uma outra forma de obter os bits aleatórios é coletá-los de uma coluna de células apenas. Desta forma, fica mais difícil a determinação de estados anteriores e posteriores da célula (e do AC), tornando o PRNG mais atraente sob vários aspectos de segurança.

5. Gerador de números aleatórios do Linux (LRNG)

A geração de números aleatórios no Linux possui três procedimentos assíncronos, como pode ser observado na Fig. 4 [Gutterman et al 2006]. O primeiro repositório de 512 bytes coleta bits aleatórios de diversos eventos do sistema operacional. O segundo repositório é alimentado a partir do primeiro e entrega bits aleatórios por meio do dispositivo */dev/random.*

A capacidade de entrega de bits por unidade de tempo destes repositórios é bem limitada e por isso existe um terceiro repositório chamado *urandom* que, a partir dos bits aleatórios recebidos do primeiro repositório (usados como semente) e de uma função determinística (normalmente SHA-1 – Secure Hash Algorithm), definida em tempo de compilação do kernel, produz novos bits (pseudo)aleatórios em alta velocidade, os quais são entregues através do dispositivo */dev*/*urandom*. E no lugar desta função SHA-1 que propomos adotar um dos ACs avaliados neste trabalho, o que traria benefícios em termos de qualidade estatística das sequências binárias e de custo para produção de tais sequências.

6. Avaliação de geradores: testes de aleatoriedade do NIST

Para avaliar as sequências de bits aleatórios produzidas pelos diversos geradores analisados, utilizamos o conjunto de 15 testes desenvolvido pelo Instituto Nacional de Padrões e Tecnologia dos Estados Unidos (NIST) [Bassham at al 2010], atendendo e superando as recomendações fornecidas.

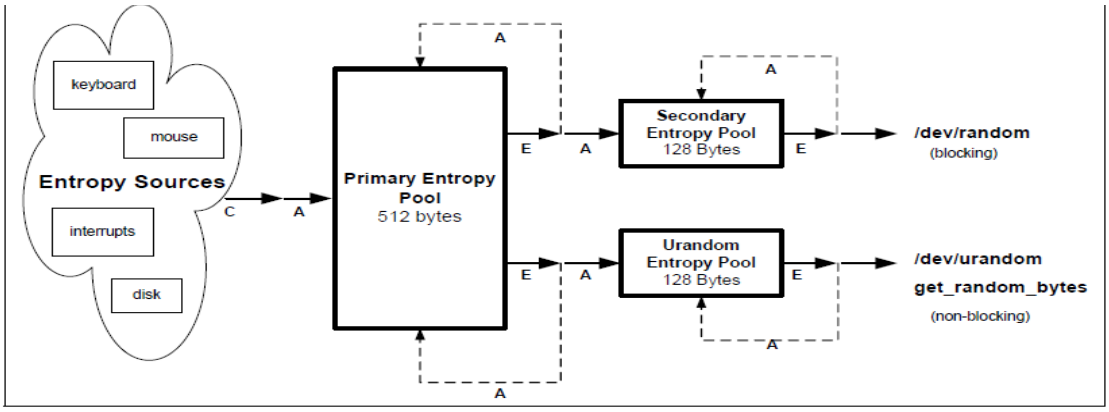

Fig. 4 – Estrutura do gerador de números aleatórios do Linux (fonte: Gutterman 2006)

6.1. Testes das sequências geradas pelos Autômatos Celulares

Foram geradas sequências binárias a partir das regras unidimensionais 30 (raio 1) e 1.436.194.405 (raio 2) e da regra bidimensional 1.453.938.345. Os ACs foram evoluídos a partir de estados iniciais aleatórios. Usamos os tamanhos 32, 64 e 128, e a coleta de bits foi feita apenas na coluna central. Cada conjunto de 100 sequências de um milhão de bits produzida por um AC passou pelos 15 testes do NIST, gerando vários resultados consolidados em um relatório padrão.

6.2. Proposta de uma métrica de consolidação de testes

A Tabela 1 apresenta um exemplo de resultados dos 15 testes sobre as 100 sequências de um milhão de bits cada, extraído do relatório padrão do NIST. Para cada teste o relatório indica um número mínimo de sequências, dentre as 100, que precisam ser aprovadas para o teste ser satisfatório. Este número é chamado de Nota Mínima e, apesar de ele ser 96 na maior parte dos casos neste exemplo, há alguns testes (Random Excursions e Random Excursions Variant) em que 54 aprovações são exigidas.

Como pode ser observado, há uma grande concentração de notas próximas ao valor máximo possível (100) e até mesmo a nota mínima fica próxima a este máximo. Este fato deixa uma faixa muito estreita para comparação entre os diversos ACs e, por este motivo, propomos uma escala relativa de comparação que transforma o intervalo entre nota mínima e máxima em um intervalo de 0 a 100.

A nota obtida no teste é então convertida para este intervalo e recebe o nome de Fator de Qualidade (FQ), definido como: *FQ = max(0, (Nota obtida - NMR) / (Nota máxima - NMR) x 100*, onde a Nota Mínima de Referência (NMR) = Nota Mínima – 1 para que FQ seja 0 somente quando a Nota Obtida estiver abaixo da Nota Mínima. Com o fator de qualidade FQ médio dos 15 testes, é possível agora fazer comparações entre

os diversos ACs, já que o FQ médio mostra o quão satisfatória foi a avaliação do conjunto de sequências binárias (Tabela 1).

Além do número de sequências aleatórias aprovadas nos testes, os relatórios fornecem também um valor estatístico p-value (teste de hipótese nula indicado nos relatórios como P_VALUE) para melhor qualificar cada resultado. Uma forma de interpretar o p-value de cada teste é vê-lo como a probabilidade de um gerador de números aleatórios perfeito produzir sequências com qualidade pior ou igual às sequências analisadas [Marton et al. (2015)].

Tabela 1 - Exemplo de resultados NIST para 100 sequências de 10^6 bits produzidas por AC regra 30 de 64 células usando FQ

NOME DO TESTE	NOTA MÍNIMA DE REFERÊNCIA (NMR)	NOTA OBTIDA	RESULTADO APROVADO - REPROVADO	FATOR DE QUALIDADE	P_VALUE	MÉDIA PARCIAL
Frequency	95,00	99,00	APROVADO	80,00	0,86769	69,41536
BlockFrequency	95,00	100,00	APROVADO	100,00	0,69931	69,93130
CumulativeSums	95,00	99,00	APROVADO	80,00	0,69865	55,89180
Runs	95,00	100,00	APROVADO	100,00	0,97170	97,16990
LongestRun	95,00	97,00	APROVADO	40,00	0,77919	31,16752
Rank	95,00	99,00	APROVADO	80,00	0,69931	55,94504
FFT	95,00	100,00	APROVADO	100,00	0,53415	53,41460
NonOverlappingTemplate	95,00	99,01	APROVADO	80,14	0,49987	40,05681
OverlappingTemplate	95,00	100,00	APROVADO	100,00	0,14533	14,53260
Universal	95,00	99,00	APROVADO	80,00	0,43727	34,98192
ApproximateEntropy	95,00	99,00	APROVADO	80,00	0,59555	47,64392
RandomExcursions	53,00	56,63	APROVADO	90,63	0,35888	32,52336
RandomExcursionsVariant	53,00	56,44	APROVADO	86,11	0,26477	22,79988
Serial	95,00	100,00	APROVADO	100,00	0,92204	92,20355
LinearComplexity	95,00	100,00	APROVADO	100,00	0,45594	45,59370
MÉDIA FINAL						50,88475

Logo, valores de p-value próximos de 1 indicam uma melhor qualidade das sequências e propomos utilizar o produto do Fator de Qualidade pelo p-value de cada teste como uma métrica que representa melhor o quão bem se saíram as sequências analisadas neste teste específico. Quanto maior forem o FQ e o p-value de cada teste, maior será o produto e mais bem avaliada será a sequência neste teste. A métrica final que consolida a qualidade da sequência aleatória em relação aos testes padronizados pelo NIST será dada então pela média aritmética simples dos produtos FQ_i x $p\text{-value}_i$.

A Tabela 2 mostra o resultado da métrica Média Final para as sequências aleatórias geradas por diversas configurações de ACs, sendo uma média de pelo menos 6 execuções dos testes estatísticos. Esta métrica proposta consolida vários resultados de testes estatísticos em um só valor que pode ser visto como a qualidade relativa da sequência aleatória analisada. Não temos conhecimento de outro trabalho que proponha métrica similar de consolidação de resultados de testes estatísticos de aleatoriedade.

Pode ser constatado que todos os ACs com tamanho 32 apresentaram Média Final muito baixa, indicando que não são grandes o suficiente para a produção de sequências de bits aleatórios satisfatórias. Já as demais configurações geram sequências que atingem Média Final acima de 35, com destaque para o AC unidimensional de raio 2 com tamanho 64.

Tabela 2. Avaliação por meio da métrica Média Final de sequências aleatórias produzidas por diferentes configurações de ACs

	Regra 30			Regra 1436194405			Regra 1453938345				
	Unidimensional Raio 1			Unidimensional Raio 2			Bidimensional Raio 1				
Tamanho	32	64	128	32	64	128	32 (4x8)	64 (4x16)	64 (8x8)	128 (8x16)	128 (4x32)
Média Alcançada	79,83	93,71	93,38	36,39	94,32	93,89	22,56	93,68	93,54	93,49	93,69
Número de Testes Aprovados	10	15	15	2	15	15	2	15	15	15	15
Média P_Value Total	0,12	0,47	0,50	0,02	0,50	0,47	0,02	0,48	0,48	0,50	0,48
Média Fator de Qualidade	45,17	80,60	75,31	0,00	83,01	79,28	0,00	73,53	77,91	79,28	77,21
Média Final	8,18	37,00	37,74	0,00	41,79	38,71	0,00	35,02	36,32	39,49	37,25

6.3. Testes das sequências geradas pelo PRNG do Linux

Foram coletadas 100 sequências de um milhão de bits cada a partir do dispositivo /dev/urandom e os resultados são mostrados na Tabela 3. Pode ser constatado pelo valor da Média Final de 37,7 que o Linux PRNG (LRNG) gera sequências com uma qualidade similar à das sequências produzidas por algumas configurações de ACs.

Tabela 3 - Resultados NIST para 100 sequências de 10^6 bits do LRNG

NOME DO TESTE	NOTA MÍNIMA DE REFERÊNCIA (NMR)	NOTA OBTIDA	RESULTADO APROVADO - REPROVADO	FATOR DE QUALIDADE	P_VALUE	MÉDIA PARCIAL
Frequency	95,00	100,00	APROVADO	100,00	0,30413	30,41260
BlockFrequency	95,00	97,00	APROVADO	40,00	0,17187	6,87468
CumulativeSums	95,00	99,50	APROVADO	90,00	0,32824	29,54120
Runs	95,00	98,00	APROVADO	60,00	0,96430	57,85770
LongestRun	95,00	100,00	APROVADO	100,00	0,77919	77,91880
Rank	95,00	98,00	APROVADO	60,00	0,33454	20,07228
FFT	95,00	100,00	APROVADO	100,00	0,43727	43,72740
NonOverlappingTemplate	95,00	98,95	APROVADO	79,05	0,45548	36,00718
OverlappingTemplate	95,00	99,00	APROVADO	80,00	0,81654	65,32296
Universal	95,00	97,00	APROVADO	40,00	0,30413	12,16504
ApproximateEntropy	95,00	96,00	APROVADO	20,00	0,61631	12,32610
RandomExcursions	58,00	61,63	APROVADO	90,63	0,47225	42,79755
RandomExcursionsVariant	58,00	61,67	APROVADO	91,67	0,55414	50,79623
Serial	95,00	100,00	APROVADO	100,00	0,20088	20,08770
LinearComplexity	95,00	100,00	APROVADO	100,00	0,59555	59,55490
MÉDIA FINAL						37,69749

7. Avaliação da relação custo/benefício dos geradores

Para avaliarmos a relação custo/benefício entre os geradores baseados em ACs e o LRNG com SHA-1, fizemos um estudo inicial levantando o número de operações necessárias para geração de bits. Avaliamos as operações que atualizam o estado e produzem novos bits no AC e as operações do algoritmo SHA-1 que produzem, a partir de uma semente aleatória, os bits disponibilizados para a interface /dev/urandom.

Como cada execução do SHA-1 no LRNG produz 80 bits pseudoaleatórios avaliamos o custo dos ACs produzirem os mesmos 80 bits aleatórios. Foram estudadas duas configurações. Na primeira foram coletadas apenas os bits da coluna central do AC. É a forma mais segura, pois não permite identificar o estado interno do AC e prever seu comportamento futuro.

Entretanto há uma grande ineficiência, pois a cada iteração são produzidos N bits, mas só 1 é aproveitado. Neste caso, o número de operações dos ACs para produzir 80 bits é muito superior ao do SHA-1, o que inviabilizaa sua adoção. Ja na segunda configuração, foi adotado o critério de se coletar os bits de ¼ das colunas do AC, o que aumenta a eficiência sem contudo expor demasiadamente o estado interno do mesmo. Desta forma , o AC se tornou mais competitivo como mostram os resultados descritos na sequência.

A Fig. 5 apresenta os resultados de cinco geradores: LRNG SHA-1, PRNG AC64 (Regra30), AC64 (Regra1436194405), AC64 (Regra1453938345) 8x8 e AC64 (Regra1453938345) 4x16. Observamos que a Regra 30 e a Regra 1436194405 possuem melhores resultados, se comparados ao LRNG SHA-1, ou seja, produzem o mesmo número de bits com menos operações que o SHA-1. Já o AC bidimensional Regra 1453938345 apresenta resultados muito próximos aos do LRNG SHA-1.

Enquanto o custo é determinado pelo número de operações, o benefício é estimado pela qualidade da sequência aleatória de acordo com a métrica Média Final descrita anteriormente. Sendo assim, a relação custo/benefício é R = número de operações / Média Final. A Tabela 4 mostra os resultados para os ACs avaliados e a Tabela 5 apresenta os resultados para o LRNG SHA-1.

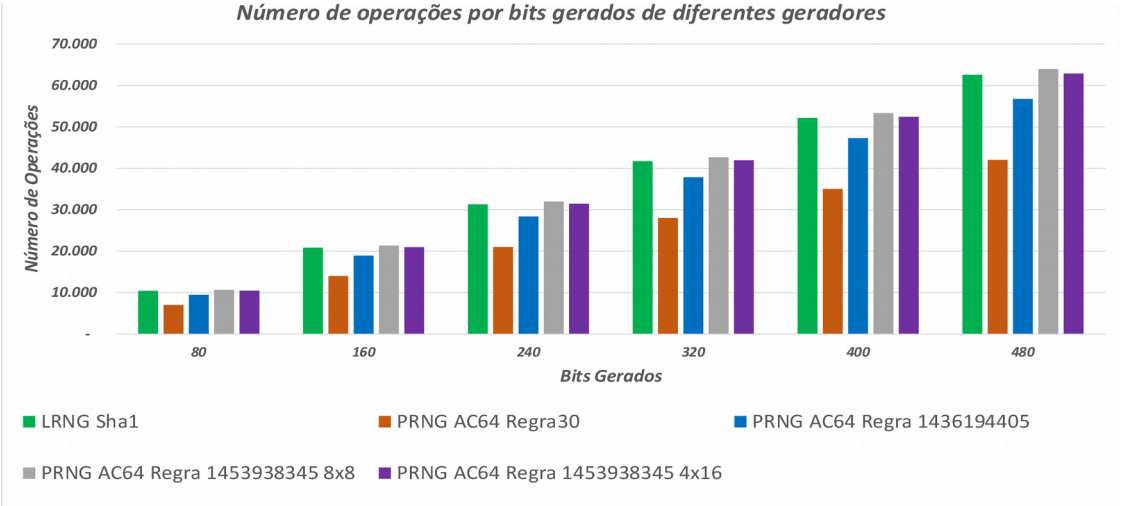

Figura 5 – Número de operações em função do número de bits gerados

Tabela 4 – Relação custo/benefício para gerar 80 bits com ACs

	Regra 30		Regra 1436194405		Regra 1453938345			
Tamanho	64	128	64	128	64 (4x16)	64 (8x8)	128 (8x16)	128 (4x32)
Média Final	37,00	37,74	41,79	38,71	35,02	36,32	39,49	37,25
operações executadas	6.780	8.100	9.455	11.241	10.480	10.660	12804	12.576
Relação Custo Benefício	183	215	226	290	299	293	324	338

Tabela 5 – Relação custo/benefício para LRNG SHA-1 gerar 80 bits

LRNG - SHA-1	
Média Final	37,70
operações executadas	10.428
Relação Custo/Benefício	277

Observamos que, dentre os ACs que são úteis na prática (tamanhos maiores que 32), a melhor razão R é apresentada pelos de raio 1, o que se justifica já que demandam menos cálculos e apresentam uma boa Média Final. Comparando com LRNG-SHA1, percebemos que os ACs de raio 1 apresentam a melhor razão R. Dentre os ACs mais complexos (raio 2 e bidimensionais) somente o de raio 2, tamanho 64, apresentou melhor razão R que o LRNG. Deve ser notada a dependência forte entre a razão R e a forma de coleta de bits nos ACs.

Formas de coleta mais eficientes, que buscam aproveitar mais bits produzidos sem, contudo, revelar o estado interno do AC devem ser buscadas em novos estudos. É recomendável também aprofundar as análises sobre os ACs, especialmente na busca de regras e configurações que produzam sequências binárias com boa aleatoriedade, segundo os testes do NIST, e com razões custo/benefício mais baixas que as obtidas até o momento.

8. Conclusões e trabalhos futuros

Este trabalho mostrou como podem ser geradas sequências de bits pseudoaleatórias a partir de Autômatos Celulares uni e bidimensionais. Mostrou ainda como o tradicional conjunto de testes estatísticos do NIST pode ter seus resultados consolidados em uma métrica que permita uma comparação mais direta entre sequências geradas por diferentes fontes.

Foi feita também uma análise do custo para geração de bits aleatórios com o gerador padrão do sistema operacional Linux e com uma versão do mesmo usando ACs, o que permitiu avaliar a relação custo/benefício das duas opções. Os resultados preliminares mostram que algumas configurações de autômatos podem substituir o algoritmo central do gerador do Linux (SHA-1) com maior desempenho e qualidade.

Como trabalhos futuros elencamos a busca por configurações de autômatos celulares mais eficientes na produção de bits aleatórios, levando em consideração as recomendações SP 800-90A e 800-90C [Barker E et. Al (2012)] como forma de construção desses algoritmos, bem como uma comparação com outras configurações propostas para o gerador de números aleatórios do Linux.

Referências

Bassham L. et al. “A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications”, Publication SP 800-22 Rev. 1a , NIST – National Institute of Standards and Technology.

Bouvry, Pascal; Franciszek Seredynski, and Albert Y. Zomaya (2003). "Application of cellular automata for cryptography." *PPAM*.

Marton, K., Suciu, A. (2015). On the interpretation of results from the NIST statistical test suite. Science and Technology, *18*(1), 18-32.

Wolfram S.(1986), Random Sequence Generation by Cellular Automata, Advances in Applied Mathematics 7, 123-169.

Wolfram,S.(2002).A new kind of science (Vol-5,p.130). Champaign, IL:Wolfram media,pp. 943.

Girau, B., Vlassopoulos, N. (2012, September). Evolution of 2-dimensional cellular automata as pseudo-random number generators. In International Conference on Cellular Automata(pp. 611-622). Springer, Berlin, Heidelberg.

Shin, S. H., Kim, D. S., Yoo, K. Y. (2012, April). A 2-dimensional cellular automata pseudorandom number generator with non-linear neighborhood relationship. In Int. Conf. on Networked Digital Technologies (pp. 355-368). Springer, Berlin.

Guan S.U., Tan S.K. (2004). Pseudorandom number generation with self-programmable cellular automata. IEEE Trans. on CAD of Integrated Circuits and Systems, 23(7):1095-1101.

Barker E., Kelsey J. NIST Special Publication 800-90A Recommendation for Random Number Generation using Deterministic Random. Bit Generators 2012.

Barker E., Kelsey J. NIST DRAFT Special Publication 800-90C Recommendation for Random Bit Generatior (RGB) Constructions. 2012.