Софийски университет "Св. Климент Охридски"

Факултет по математика и информатика

Въведение в криптографията и сигурността на данни

Курсов проект на тема

Някои приложения на клетъчните автомати в криптографията

Изготвил:

Венета Кирева, ФН: 6MI3400573

2025 г.

Съдържание

1 Въведение **2**

2 Клетъчни автомати - същност **3**

3 Приложение на клетъчните автомати в криптографията **5**

3.1 Генериране на псевдослучайни числа 5

3.2 Поточни шифри 9

3.3 Блокови шифри 11

3.4 Хеширащи функции 16

3.5 Шифриране на изображения 19

4 Изводи **21**

5 Литература **22**

1 Въведение

Клетъчните автомати представляват математически модел, който има широк набор от приложения за моделиране на процеси в природните науки. Техните простота, бързина и способност за генериране на сложни последователности, ги правят мощен инструмент за решаване на различни задачи. В криптографията, те могат да се използват за генериране на псевдослучайни числа, като основа на поточни и блокови шифри и хеширащи функции.

Настоящият проект ще разгледа основните принципи на клетъчните автомати, ще въведе някои техни приложения в криптографията и ще представи някои практически примери.

2 Клетъчни автомати - същност

Клетъчните автомати (КА) представляват абстрактна изчислителна система. Те са съставени от n-мерна решетка от изброим брой клетки, като всяка клетка се намира в точно едно от краен набор (азбука) от състояния [1]. Графично КА често се представят чрез двумерна решетка.

Веднъж на единица време, състоянията на клетките се променят едновременно, следвайки набор от предварително дефинирани правила за преход. Обновяването на състоянието на дадена клетка се получава като се вземат предвид състоянията на съседите ѝ. За целта се използва предварително определен *радиус* на съседство - цяло число, което показва колко клетки от всяка страна определяме като съседни. При едномерна решетка и радиус 1, за изчисляване на състоянието използваме само клетките, долепени до избраната.

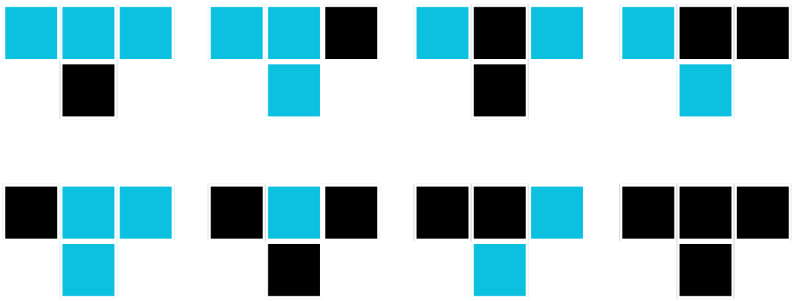
Правило R за КА с радиус r можем да представим формално като

$$s_i^{t+1} = R(s_{i-r}^t, ..., s_{i-1}^t, s_i^t, s_{i+1}^t, ..., s_{i+r}^t)$$

където s_i^t е състоянието, в което се намира i-та клетка в момента t.

Правилата при радиус $r = 1$ могат да се представят графично като мрежа от по 4 полета (фиг. 1). За дадена клетка, всяка тройка отгоре представлява възможна конфигурация на съседство в момента t, като разглежданата клетка е тази в средата: за всяка конфигурация квадратчето отдолу определя състоянието на клетката в следващия момент $t + 1$.

За простота на представянето, Волфрам създава механизъм за именуване на всяко правило [2]: като се зададе стойност 0 или 1 на всяко от двете състояния от азбуката, долният ред може да се прочете като двоично число (01011010).



Фигура 1: Графично представяне на правило 90 - синият цвят е представен като 1, а черният - като 0.

Името на правило се получава при преобразуване на числото от двоична в десетична бройна система (фиг. 2).

Number	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Neighborhood Rule no.	**111**	**110**	**101**	**100**	**011**	**010**	**001**	**000**
30	0	0	0	1	1	1	1	0
51	0	0	1	1	0	0	1	1
15	0	0	0	0	1	1	1	1
90	0	1	0	1	1	0	1	0
60	0	0	1	1	1	1	0	0
85	0	1	0	1	0	1	0	1
102	0	1	1	0	0	1	1	0
150	1	0	0	1	0	1	1	0
153	1	0	0	1	1	0	0	1
195	1	1	0	0	0	0	1	1
170	1	0	1	0	1	0	1	0
204	1	1	0	0	1	1	0	0
240	1	1	1	1	0	0	0	0

Фигура 2: Някои правила и техните имена. [12]

3 Приложение на клетъчните автомати в криптографията

3.1 Генериране на псевдослучайни числа

Генераторите на псевдослучайни числа (Pseudorandom Number Generator, PRNG) имат важна роля в криптографията - много голяма част от криптографските алгоритми разчитат на случайността, за да осигурят сигурност.

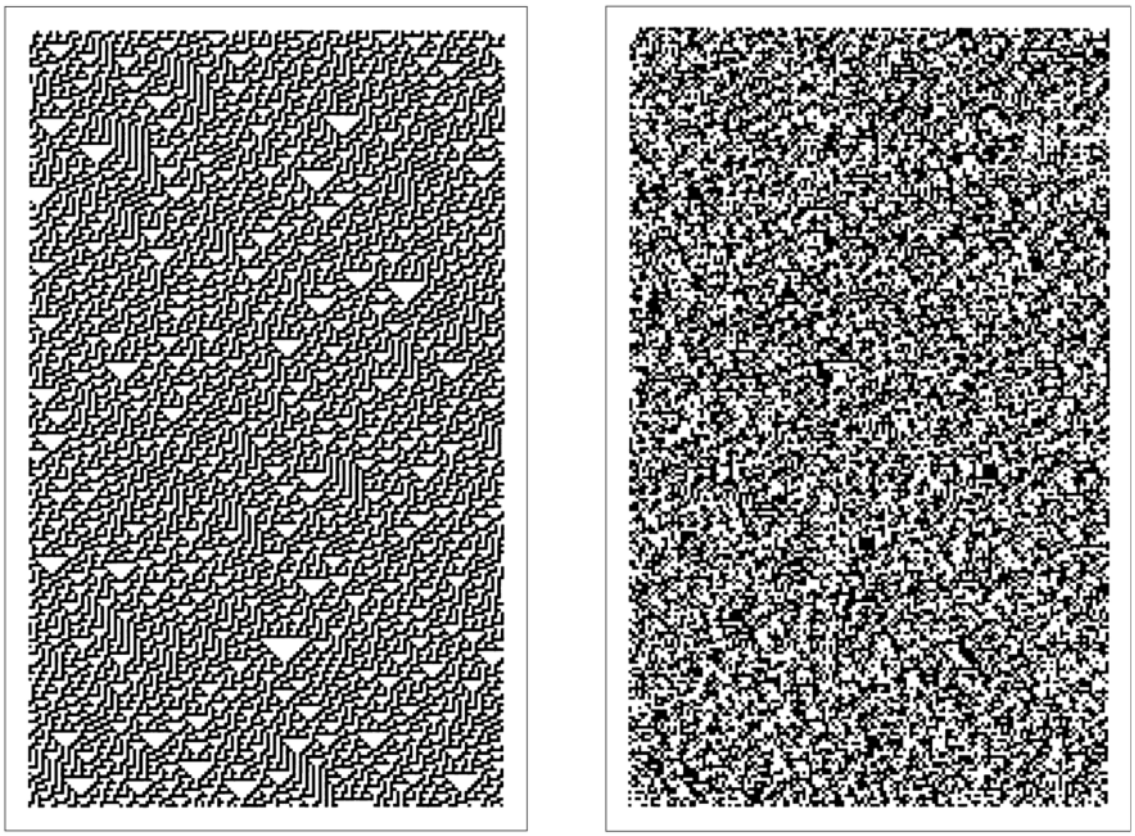
Приложение на КА като PRNG е предложено първо от Волфрам през 1983 г. [2]. В разработката е разгледан едномерен КА с азбука от 2 състояния. Генерирането се основава на правило 30, приложено локално върху едномерна решетка с дължина поне 127 клетки, при радиус r = 1. Само няколко години по-късно - в [3], е доказано, че предложеният от Волфрам PRNG има лоши криптографски свойства, поради алгебричната структура на правило 30 - то зависи линейно от s_{i+1}^t:

$$s_i^{t+1} = s_{i-1}^t \oplus s_i^t \oplus s_{i-1}^t s_i^t \oplus s_{i+1}^t$$

Макар алгоритъмът на Волфрам да е доказано уязвим на различни криптографски атаки, по-късно КА намират приложение в различни разработки за генериране на псевдослучайни числа.

Един начин за получаване на допълнително разсейване на получените резултати е да се използват *разреждане по време* и *разреждане по място* [4] (фиг. 3). Времево разреждане означава, че не всички генерирани битове се считат за част от случайната последователност - напр. ако се запази само един бит от все-

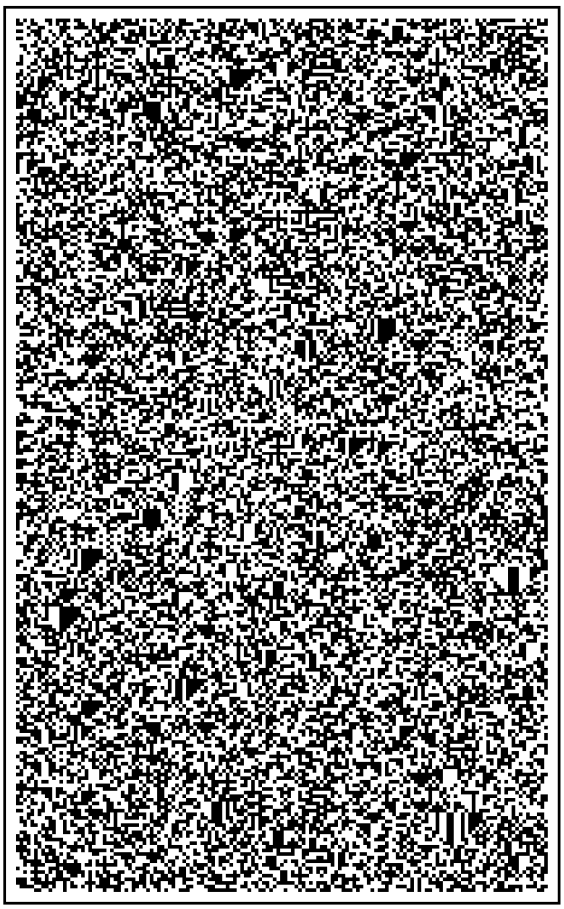
ки два, имаме времево разреждане 1. Пространствено разреждане се изразява в това, да се разглеждат само определени позиции в ред, като се указва колко позиции трябва да бъдат пропуснати между два последователни елемента.



Фигура 3: Визуализация на PRNG, използващ на КА с решетка от 150 клетки с правило 30: (a) - без времево разреждане (b) времево разреждане с 2. [4]

Комбинирането на две или повече правила е друг подход, използван в множество разработки (фиг. 4).

Проведените в [5] експерименти с правила 90 и 150 показват, че произволното прилагане на едно от двете правила върху дадена клетка дава по-добри резултати от редуването им в точно определена последователност. Използване-



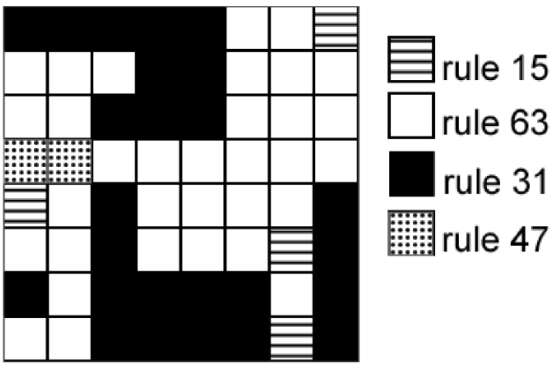
Фигура 4: PRNG, използващ произволна комбинация от правила 90, 105, 150, и 165. [6]

то на пространствено разреждане също подобрява резултата от тестовете. Експерименти са направени и с комбинация между правила 30 и 45. Получените резултати показват по-голяма корелация и поведение, значително по-различно от предходната комбинация. Големите разлики в представянето демонстрират нуждата от внимателен подбор на параметрите при употребата на КА като PRNG.

Друга методология, основана на двумерни КА, е предложена в [7]. Изборът на правила е направен чрез генетични алгоритми, като за функция на приспо-

собимост е избрана ентропията на последователността от битове. Подходът е тестван чрез опити, проведени върху двумерни решетки с размерност 8×8. Всяка популация се развива за 200 епохи. Използвани са адитивни правила (включващи само XOR и XNOR логика) с радиус $r = 5$.

Резултатите от 30-те проведени експеримента показват, че решетките с най-добри свойства съдържат между 4 и 9 различни правила. В много от тях се съдържа подмножеството от правила 31, 47 и 63 (фиг. 5).



Фигура 5: Примерен КА от финална популация, съдържащ само 4 правила - 15, 31, 47 и 63. [7]

За генериране на числа в двумерния случай, автоматът се изпълнява за 4 времеви стъпки. Всеки 4 бита от клетка се преобразуват в шестнадесетична цифра, а решетката се обхожда от горе надолу и от ляво надясно, за да се генерира последователност от случайни числа. Така, на всеки 4 стъпки се генерират 64 шестнадесетични числа.

3.2 Поточни шифри

КА се използват успешно за шифриране с поточни шифри.

В [8] се разглежда тяхната употреба при едномерна и двумерна решетка. Ако c_i е шифриран бит, k_i е бит от ключа, а p_i е бит от откритото съобщение, всички на позиция i, то шифрирането се състои в прилагане на следвана формула върху всяко p_i:

$$c_i = k_i \oplus p_i$$

Аналогично, за да дешифрираме прилагаме

$$p_i = c_i \oplus k_i$$

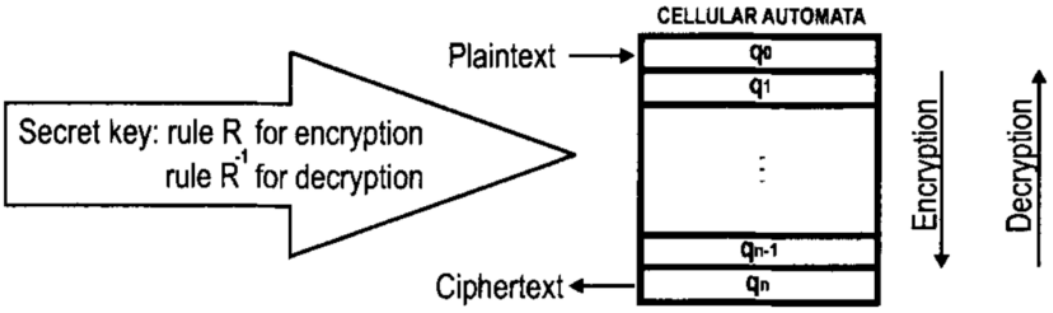
В едномерния случай, на всяка клетка се определя (на случаен принцип и с еднаква вероятност) кое от правилата 90, 105, 150 и 165 действа. Ключът $K = (R_i, C(0))$ се формира, като текущият набор от правила R_i се приложи върху произволно инициализираната решетка $C(0)$. Начините за инициализиране на едномерна решетка от N клетки, използвайки азбука от 2 състояния, са 2^N, а начините за определяне кое на това правило действа върху коя клетка - 4^N.

В резултат, ако недобронамерен актьор направи опит да получи чрез пълно изчерпване на възможностите ключа, то той ще се наложи да обходи пространство от $4^N \times 2^N = 2^{3N}$ състояния. Добавянето на отместване на време и избор на голямо N може допълнително да уголеми пространството за търсене и да намали комбинациите, които генерират ключовата последователност от битове. Направеният криптоанализ показва, че, при подходящо достатъчно голямо N, се постига добро ниво на сигурност.

В двумерния случай, пространството от състояния е значително по-голямо поради 7-те налични адитивни правила - $7^N \times 2^N$. Получената последователност от битове е по-трудна за криптоанализ от едномерния случай, поради по-големия брой съседи и нуждата от значителни изчислителни ресурси.

3.3 Блокови шифри

Един подход за блоково шифриране при КА е чрез т.нар. *обратими* правила. Правило R наричаме *обратимо*, ако съществува такова правило R^{-1}, че, при прилагането му върху резултата от R, независимо от подадените входни данни, ги възстановява. При използването на обратими правила, дешифрирането се състои в изпълнение на обратното на шифриращото правило върху шифрирания текст (фиг. 6).



Фигура 6: Схема на шифрирането чрез обратими КА. [9]

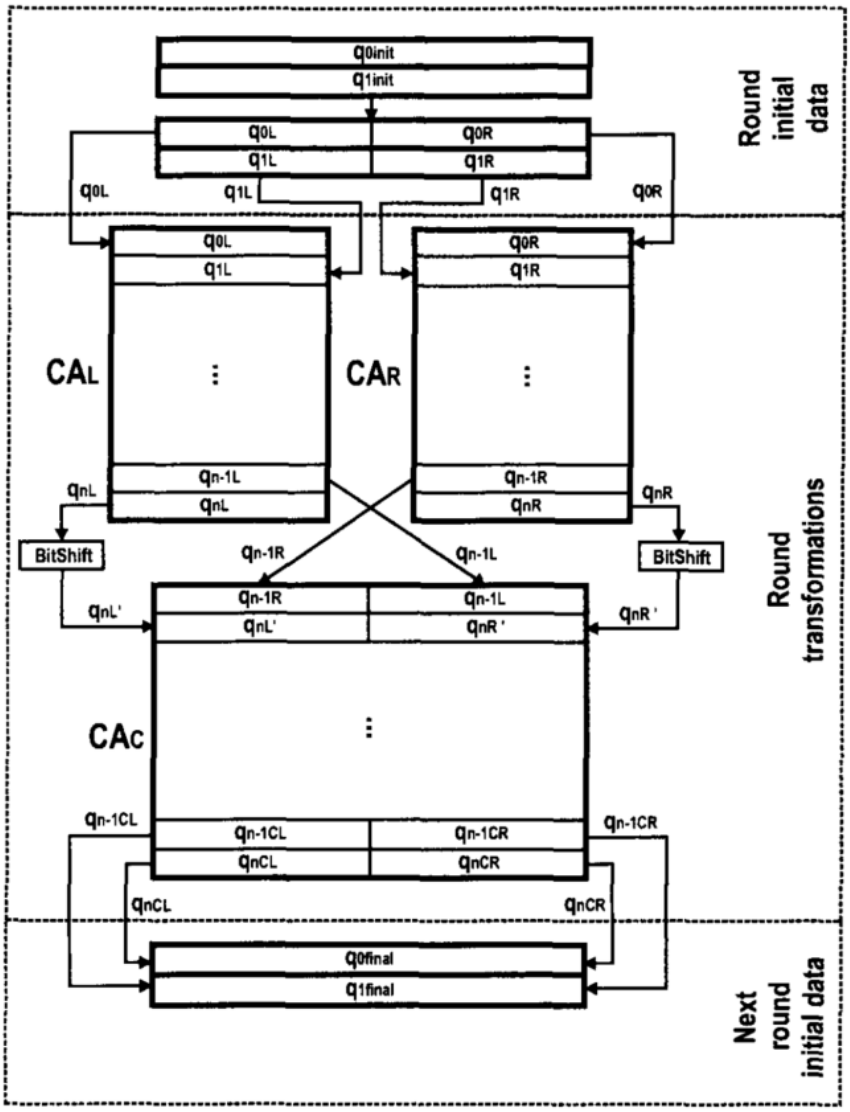
Само малка част от правилата са обратими - обратното правило на 15 е правило 85, а правила 51 и 204 са обратими на себе си. [9]

В алгоритъма, разгледан в [9], са използвани 4 КА - CA_L, CA_R, състоящи се от 32 клетки, CA_C - от 64 клетки, и CA_S - 16 клетки. CA_C използва правила с радиус 3, а останалите - с радиус 2.

Освен откритото съобщение, на входа на алгоритъма се подава и втори набор от битове, който е генериран случайно чрез PRNG. След това откритото съобщение и случайно генерираните битове се разделят на блокове с дължина 64.

Всяка 64-битова двойка се разделя на по 2 - лява и дясна, от по 32 бита.

Тези стойности се използват за инициализирането на началните състояния на CA_L и CA_R.



Фигура 7: Една стъпка от алгоритъма. [9]

Автоматите CA_L и CA_R изпълняват n_1 стъпки, където $n_1 > 18$, след което върху резултата се прилага BitShift операция, като битовете от левия автомат са изместени надясно, а тези от десния - наляво. Получените стойности се из-

ползват за начална конфигурация на CA_C, който се изпълнява за n_2 стъпки при $n_2 > 16$.

Ключът представлява 224-битовата конкатенация на правилата на автоматите. След шифриране на последния блок, XOR операция с определени битове от ключа се прилага върху крайните данни и последните конфигурации на клетъчните автомати.

Дешифрирането се състои в прилагането на операциите в обратен ред, като при операцията BitShift се прилагат противоположни изместванията върху съответните конфигурации.

Алгоритъмът осигурява висока сигурност чрез големия брой възможни ключове - 2^{224}. Дори по-голяма сигурност може да се постигне чрез увеличаване на размера на блоковете на 128 бита, което значително увеличава пространството от възможни стойности. Това прави търсенето с пълно изчерпване практически невъзможно.

В [10] се разглежда метод за блоково шифриране, използващ генетични алгоритми. Предложеният модел шифрира 128-битови блокове в с помощта на 128-битов ключ.

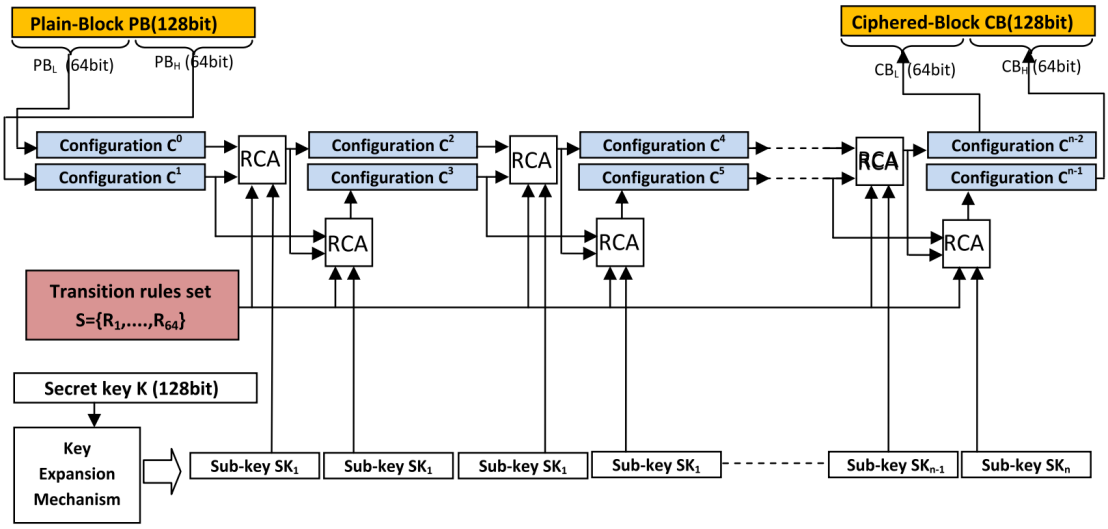
Всеки 128-битов блок се разделя на два по-малки. Дефинира се набор от 64 преходни правила, всяко отговарящо за клетката на определена позиция. На всяка стъпка, новите състояния на конфигурацията се изчисляват, като на всяка позиция се прилага или съответното правило, или неговото допълнение, в зависимост от ключа.

От първоначалния ключ се генерира набор от 64-битови под-ключове, които се използват на всяка стъпка от процеса на шифриране. Разширяването на ключа представлява ляво изместване с 8 бита, след което се избират за под-

ключ най-десните 64 бита.

Генетичен алгоритъм се използва за определяне на набора от правила, които се прилагат на всяка стъпка. Всяко от 64-те правила може да бъде представено чрез цяло число от множеството $\{0, ..., 2^{2^{2r+1}}\}$, където r е радиуса на съседството. Така всяка комбинация от правила може да се представи като наредена 64-ка.

Функцията на приспособимост е избрана така, че да отразява степента на нелинейност на шифъра.



Фигура 8: Диаграма на алгоритъма. [10]

Експериментите са проведени при следните параметри:

- размер на популацията: 200
- скорост на кръстосване: 70%
- скорост на мутация: 10%
- брой поколения: 500

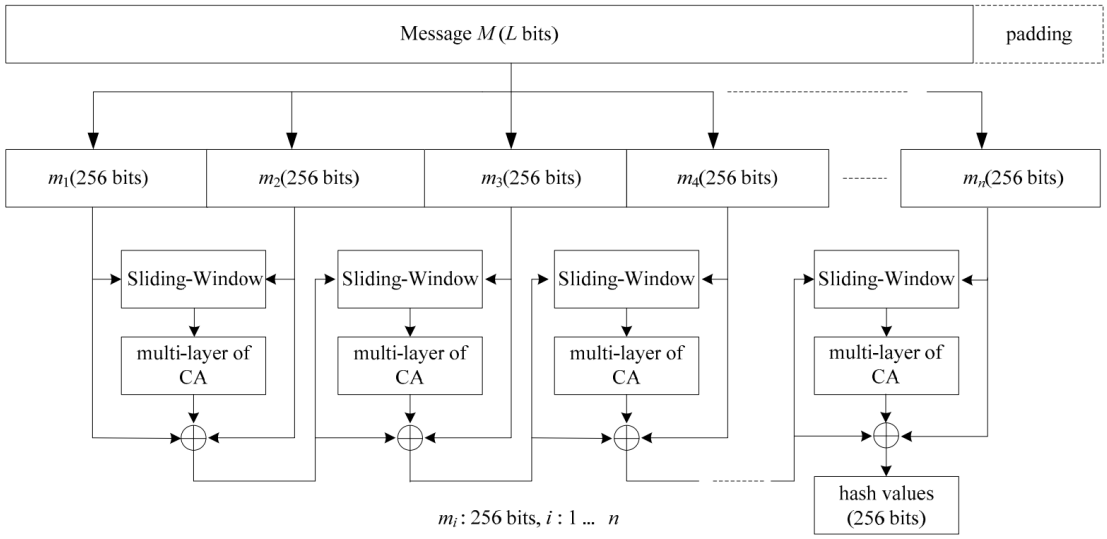
- радиус: 2
- брой стъпки за всеки КА: 32

Направените статистически тестове показват отлични свойства при най-добрата получена комбинация - ентропия 7.993 (оптимална стойност 8.0) и средна стойност на получените байтове 127.15 (оптимална стойност 127.5). Издържани са и всички тестове от DIEHARD [14].

3.4 Хеширащи функции

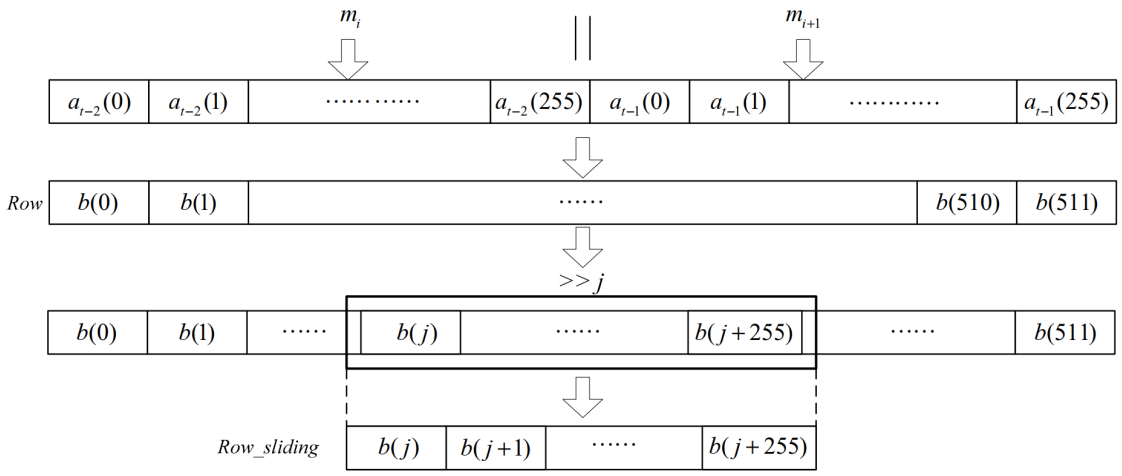
Поради простотата и скоростта на алгоритмите, базирани на КА, те намират приложение и при създаването на хеширащи функции.

Една такава имплементация е представена в [11]. Първо входните данни се разделят на n блока от по 256 бита, като, ако дължината на входа не е кратна на 256, се допълва с 1, следвано от необходимия брой 0. (фиг. 9)



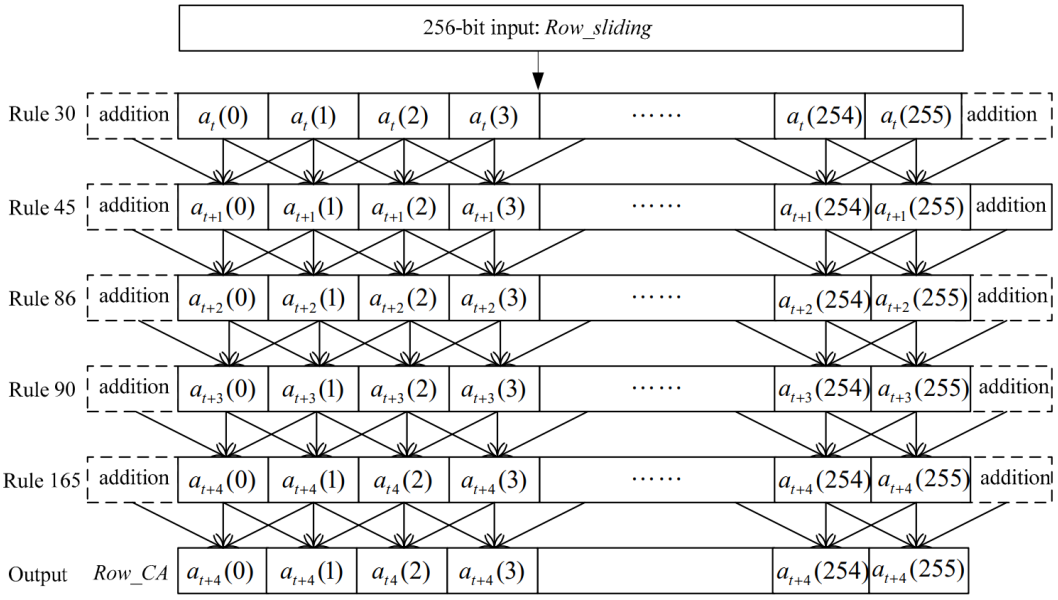
Фигура 9: Диаграма на алгоритъма. [11]

След разделянето, блоковете от 256 бита - m_i и m_{i+1}, се конкатенират. Въз основа на първите 8 бита от m_i се определя изместването от началото на плъзгащ се прозорец с големина 256 (фиг. 10). Данните във всеки прозорец биват подадени на КА, състоящ се от 5 слоя, всеки от които работи с различно правило (фиг. 11). За експеримента са избрани правила 30, 45, 86, 90 и 165, тъй като корелацията помежду им е 0.5. Тъй като правилата използват радиус $r = 1$, допълнително се добавят първият и последният бит от прозореца съответно в края и в началото на входа на всеки слой.



Фигура 10: Определяне на плъзгащ се прозорец. [11]

Между резултата от изчислението (Row_CA) и входните блокове - m_i и m_{i+1}, се прилага XOR операция: $result = m_i \oplus m_{i+1} \oplus Row\_CA$



Фигура 11: Отработка на данните в плъзгащия се прозорец с многослоен КА. [11]

На следващата стъпка, полученият резултат $result$ се използва като m_i, а m_{i+1} приема стойността на следващия блок от началното съобщение.

Резултатът от прилагането на алгоритъма върху акумулираната стойност и m_n е и финалната хеш стойност.

При анализа на предложения метод, изложен в [11], подходът има много добра устойчивост на колизии, почти съпоставима с тази на SHA-256. Изследваните статистически характеристики също са отлични и близки до тези на SHA-256.

3.5 Шифриране на изображения

Всяко изображение може да се разглежда като матрица от цели числа - стойностите на пикселите. Това представяне е подходящо за шифриране.

[12] разглежда алгоритъм за шифриране на черно-бели изображения, основан на КА.

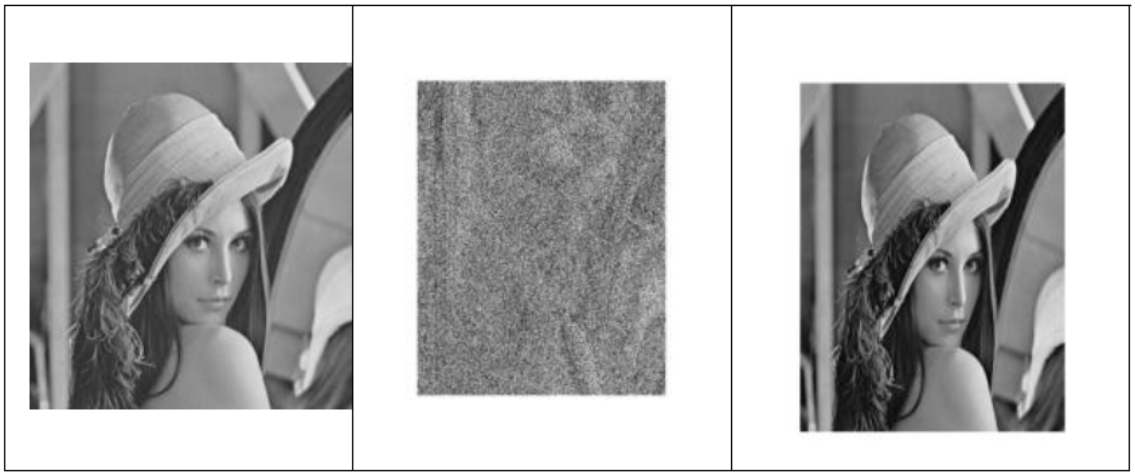
Ключът се състои от наредената тройка (k_1, k_2, R). В нея, k_1 е цяло число между 1 и 255, а k_2 е комбинация от правила (разгледано в [13]) - в експериментите е използвана стойността 11110000, като правило 90 съответства на 1, а правило 102 - на 0. R е матрица, елементите на която са произволно генерирани цели числа от 1 до 12, която има същата размерност като матрицата от пикселите на изображението.

За шифриране, стойността p на всеки пиксел се обновява по формулата $p = (p+k_1)\ mod\ 256$ - използвана е идеята на шифъра на Цезар. Новополученото число се преобразува в своето 8-битово представяне. За всеки бит, на който в k_2 съответства 1, се прилага r пъти правило 90, а при 0 - правило 102. r е елемент от R, който стои на позицията на пиксела в изображението.

Процедурата се повтаря за всеки пиксел, за да се получи шифрираното изображение. За дешифриране се прилага обратната процедура (фиг. 12).

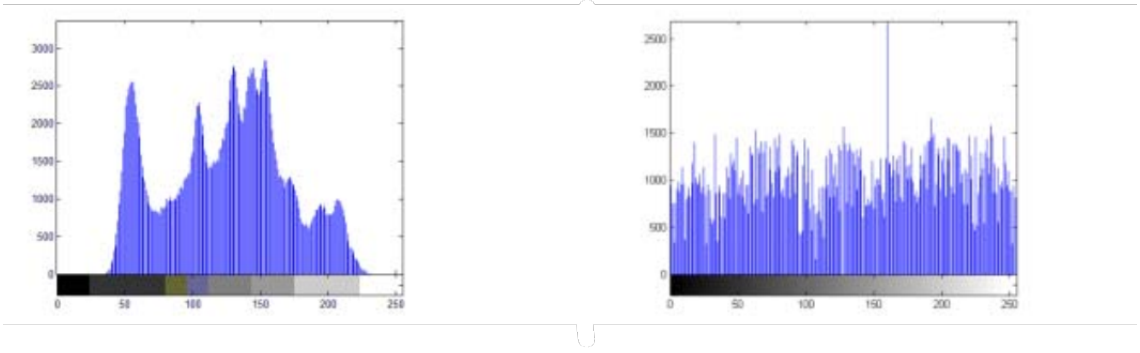
Алгоритъмът е достатъчно устойчив на атака, използваща пълно изчерпване на възможностите за избор - за шифриране на изображение с големина $m \times n$ пиксела чрез 2 правила, броят на възможните комбинации за ключа е $m \times n \times C_2^{256} \times 2^8 \times 255$. Комбинациите на 2 правила са $C_2^{256} \times 2^8$, а ключа от шифъра на Цезар може да се избере по 255 начина.

Направен е анализ на пикселите на откритото и на шифрованото изображе-



Фигура 12: Резултати от прилагането на шифриращия и дешифриращия алгоритми. [13]

ние (фиг. 13) - разпределенията са значително различни. Анализирани са също корелацията между двойки съседни пиксели в откритото и шифровано съобщение - резултатите потвърждават, че в генерираното изображение пикселите имат произволни стойности.



Фигура 13: Хистограма на стойностите на пикселите в откритото (ляво) и шифрованото (дясно) изображения. [13]

4 Изводи

Клетъчните автомати намират приложение в различни области на криптографията.

Техните леснота на паралелизацията, бързина на изчисленията и способността им да генерират сложни случайни последователности, им дават значително предимство пред други подходи и ги правят удобен инструмент за решаване на различни криптографски задачи.

Разнообразието от предложените в различни разработки модификации - различни правила, азбуки и комбинации между тях, ги прави приложими при голям набор от проблеми. Въпреки това, за постигане на оптимални резултати, е важно внимателно да се подберат параметрите на модела и архитектурата, които се използват.

5 Литература

[1] Berto, Francesco, and Jacopo Tagliabue. "Cellular automata."(2012).

[2] Wolfram, Stephen. "Statistical mechanics of cellular automata."Reviews of modern physics 55.3 (1983): 601.

[3] Meier, Willi, and Othmar Staffelbach. "Analysis of pseudo random sequences generated by cellular automata."In Workshop on the Theory and Application of of Cryptographic Techniques, pp. 186-199. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991.

[4] Tomassini, Marco, Moshe Sipper, Mosé Zolla, and Mathieu Perrenoud. "Generating high-quality random numbers in parallel by cellular automata."Future Generation Computer Systems 16, no. 2-3 (1999): 291-305.

[5] Hortensius, Peter D., Robert D. Mcleod, Werner Pries, D. Michael Miller, and Howard C. Card. "Cellular automata-based pseudorandom number generators for built-in self-test."IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 8, no. 8 (1989): 842-859.

[6] Tomassini, Marco, and Mathieu Perrenoud. "Nonuniform cellular automata for cryptography."Complex systems 12, no. 1 (2000): 71-82.

[7] Tomassini, Marco, Moshe Sipper, and Mathieu Perrenoud. "On the generation of high-quality random numbers by two-dimensional cellular automata."IEEE Transactions on computers 49, no. 10 (2000): 1146-1151.

[8] Tomassini, Marco, and Mathieu Perrenoud. "Stream cyphers with one-and two-dimensional cellular automata."In International Conference on Parallel Problem Solving from Nature, pp. 722-731. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000.

[9] Seredynski, Marcin, and Pascal Bouvry. "Block cipher based on reversible cellular automata."New Generation Computing 23 (2005): 245-258.

[10] Faraoun, Kamel Mohamed. "A genetic strategy to design cellular automata based block ciphers."Expert Systems with Applications 41, no. 17 (2014): 7958-7967.

[11] Wu, Shyi-Tsong, and Jieh-Ren Chang. "Secure One-Way Hash Function Using Cellular Automata for IoT."Sustainability 15, no. 4 (2023): 3552.

[12] Mantri, Jibendu Kumar, Rajalaxmi Mishra, and P. Gahan. "A Novel Encryption Scheme Using Hybrid Cellular Automata."In 2019 International Conference on Information Technology (ICIT), pp. 382-387. IEEE, 2019.

[13] Roy, Satyabrata, Subrata Nandi, Jayanti Dansana, and Prasant Kumar Pattnaik. "Application of cellular automata in symmetric key cryptography."In 2014 International Conference on Communication and Signal Processing, pp. 572-576. IEEE, 2014.

[14] Marsaglia, George. "DIEHARD: a battery of tests of randomness."http://stat. fsu. edu/geo (1996).