ივანე ჯავახიშვილის სახელობის თბილისის სახელმწიფო უნივერსიტეტი

გიორგი სეხნიაშვილი

ახალი tweakable ტიპის სიმეტრიული ალგორითმის აგება

ზუსტ და საბუნებისმეტყველო მეცნიერებათა ფაკულტეტი,
ინფორმაციული სისტემები

ნაშრომი შესრულებულია მაგისტრის ხარისხის მოსაპოვებლად

ზურაბ ქოჩლაძე, ასოცირებული პროფესორი

თბილისი

2016

სარჩევი

ანოტაცია

ნაშრომში განხილულია ამერიკელი მათემატიკოსის ლესტერ ჰილის მიერ შექმნილი ალგორითმის მოდიფიცირებული ვარიანტისთვის საწყისი გასაღებიდან რაუნდული გასაღებების გამომუშავების სქემა.

ჰილის ალგორითმში ღია ტექსტი გადადის რიცხვებში(ათობით სისტემაში) და მრავლდება კვადრატულ მატრიცაზე $n \times n$-ზე, რომელსაც აუცილებლად გააჩნია შებრუნებული მატრიცა. ეს მეთოდი თანამედროვე კრიპტოგრაფიული ალგორითმებისთვის გადaკეთდა შემდეგნაირად: ბიტური სტრიქონი(128 ბიტი), რომელიც მიიღება ტექსტის კომპიუტერში შეყვანის შემდეგ, დაიყობა ბაიტებად და მიიღება მატრიცა 4×4-ზე, რომლის ელემენტები(ბაიტები) კვლავ გადადის ათობით სისტემაში და მრავლდება თვითშებრუნებად მატრიცაზე ზომით 4×4. ეს წარმოადგენს ახალი tweakable ბლოკური შიფრის ერთ-ერთ ძირითად ოპერაციას.

ასეთ ალგორითმს სჭირდება გასაღების გაფართოების პროცედურა, რომელიც საშუალებას მოგვცემს საწყისი გასაღებიდან მივიღოთ რაუნდული გასაღებები(10 ცალი).

სამაგისტრო ნაშრომის მიზანია ასეთი გასაღების გაფართოების პროცედურის შექმნა.

Annotation

In this work we will discus generating round keys for modified version of symmetric block cipher algorithm proposed by American mathematician – Lester Hill.

In Hill's algorithm open text symbols transform into numbers(in decimal system) and are multiplied by square $n \times n$ matrix, which must have inverse matrix. This method for modern cryptographic algorithms was altered in following method: bit string(128 bit), which is generated after we input text in computer, divide into bytes and is generated as a matrix 4×4, the elements of which are converted to decimal again and are multiplied by inversable matrix 4×4. This is one of the major operation of new tweakable block cipher.

This type of algorithm needs procedure of extend key, which allows us to generate round keys(10 keys).

The main goal of this work is to create this kind of key extending procedure.

შესავალი

დღესდღეობით ინფორმაციის უსაფრთხოება ერთერთ უმნიშვნელოვანეს საკითხს წარმოადგენს თითქმის ყველა სფეროში. ეს განსაკუთრებით ეხება ელექტრონულად შენახულ ან მიმოცვლილ ინფორმაციას, რადგანაც ყველა სფერო აქტიურად იყენებს თანამედროვე ტექნიკურ საშუალებებს და მათი მეშვეობით აწარმოებს ინფორმაციის მიმოცვლას. არსებობს სხვადასხვა სახის შიფროსისტემები რომელთა საშუალებით ხდება ინფორმაციის დაშიფვრა და ამ სახით მიმოცვლა. ასეთია მაგალითად, ასიმეტრიული და სიმეტრიული შიფრო სისტემები, მაგრამ ვინაიდან ასიმეტრიული შიფრო სისტემების სიჩქარე საკმაოდ დაბალია ისინი არ გამოიყენება უშუალოდ მონაცემების უსაფრთხოებისათვის, მათ ძირითადად იყენებენ უსაფრთხო არხის მოსამზადებლად, რომლითაც შემდგომ გადაიცემა სიმეტრიული შიფრო სისტემებით დაშიფრული მონაცემები, ვინაიდან სიმეტრიული შიფრო სისტემები ბევრად სწრაფია. ასეთ სიმეტრიულ სისტემებს მიეკუთვნებიან: AES, DES, IDEA და ა.შ.

ამ ნაშრომშიც შევეხებით სიმეტრიულ შიფრო სისტემას და მისთვის რაუნდული გასაღებების გამომუშავების საკითხს.

Tweakable ბლოკური ალგორითმები

თანამედროვე კრიპტოგრაფიაში ინფორმაციის კონფიდენციალურობის დასაცავად, მათი სიჩქარიდან გამომდინარე, გამოიყენება სიმეტრიული ბლოკური ალგორითმები. მიუხედავად ბლოკური შიფრების არქიტექტურის და მათში გამოყენებული ოპერაციების განსხვავებულობისა, მათი მუშაობის შედეგი ყოველთვის ერთი და იგივეა. n სიგრძის

ბიტური სტრიქონი, რომლის სტრუქტურაც განსაზღვრულია ღია ტექსტით, k სიგრძის გასაღების გამოყენებით და მრავალჯერადი იტერაციის შემდეგ გადადის n სიგრძის ფსევდოშემთხვევით ბიტურ სტრიქონში. აქედან გამომდინარე, მათემატიკურად ნებისმიერი ბლოკური შიფრი შეგვიძლია წარმოვადგინოთ როგორც ორ ცვლადზე დამოკიდებული ფუნქცია:

$$E: \{0,1\}^n \times \{0,1\}^k \rightarrow \{0,1\}^n$$

სადაც n არის ბლოკის სიგრძე, ხოლო k გასაღების სიგრძე. მათი მნიშვნელობები დამოკიდებულია დაშიფვრის კონკრეტულ ალგორითმზე. შემავალ პარამეტრს წარმოადგენს n სიგრძის ბიტური სტრიქონი, რომლის სტრუქტურაც განსაზღვრულია ღია ტექსტით. ალგორითმში ხდება შემავალი პარამეტრის გასაღებთან დაკავშირების ოპერაცია და საბოლოოდ გამოსასვლელზე მიიღება ასევე n სიგრძის ბიტური სტრიქონი, რომელიც წარმოაგდენს შიფროტექსტს. ეს ოპერაციები პრაქტიკაში რამდენიმეჯერ მეორდება. მათ რაუნდები ეწოდებათ.

კლოდ შენონმა თავის ფუნდამენტურ ნაშრომში აჩვენა რომ არსებობს ერთადერთი თეორიულად გაუტეხავი სიმეტრიული შიფრი(ერთჯერადი ბლოკნოტი), რომლის წარმატებული ფუნქციონირებისათვის აუცილებლად უნდა სრულდებოდეს შემდეგი პირობები: გასაღების სიგრძე უნდა იყოს ღია ტექსტის სიგრძის ტოლი, გასაღები უნდა წარმოადგენდეს აბსოლუტურად შემთხვევით მიმდევრობას და ის უნდა გამოვიყენოთ მხოლოდ ერთხელ. ცხადია ასეთი შიფრის გამოყენება ყოველდღიურ პრაქტიკაში ძალიან მოუხერხებელია. ყველა დანარჩენი სიმეტრიული ალგორითმი კი შეიძლება იყოს მხოლოდ გამოთვლადად მედეგი კრიპტოანალიზური შეტევების მიმართ, რაც იმას ნიშნავს რომ თუ მოწინააღმდეგეს გააჩნია შემოუსაზღვრავი შესაძლებლობები, მას ყოველთვის შეუძლია გატეხოს ასეთი შიფრები, მაგრამ პრაქტიკაში არ გვხვდება შემოუსაზღვრავი შესაძლებლობების მქონე მოწინააღმდეგე, ამიტომ ალგორითმის უსაფრთხოების გასაზომად უნდა ვიპოვოთ თანაფარდობა კრიპტოანალიტიკოსის შესაძლებლობებს და ალგორითმის მედეგობას შორის.

თუ მოწინააღმდეგის მიზანი გასაღების გამოთვლაა და მისთვის ცნობილია შესასვლელი და გამოსასვლელი მნიშვნელობების რაიმე q რაოდენობის წყვილები

$(M_1, C_1), \ldots (M_q, C_q)$ მაშინ ბლოკური შიფრი იქნება უსაფრთხო, თუ საუკეთესო შეტევა, რომელიც შეუძლია განახორციელოს მოწინააღმდეგემ მოითხოვს ისეთი დიდი რაოდენობის q წყვილებს ან გამოთვლების ისეთ დიდ t დროს, რაც აღემატება კრიპტოანალიტიკოსის შესაძლებლობებს. თუმცა ბლოკური შიფრის გასაღების გამოთვლის შეტევის მიმართ უსაფრთხოება არ ნიშნავს მის უსაფრთხოებას მთლიანობაში, რადგანაც კ. შენონმა აჩვენა რომ ალგორითმი შეიძლება უშვებდეს ღია ტექსტის შესახებ რაიმე ინფორმაციის გაჟონვას. თუ ეს ფაქტი ხდება მაშინ კრიპტოანალიტიკოსს საკმარისი რაოდენობოს ინფორმაციის დაგროვების შემდეგ ეძლევა საშუალება მთლიანად გატეხოს ალგორითმი. ამიტომ თუ ალგორითმი გვინდა რომ იყოს უსაფრთხო, უნდა შეგვეძლოს დავამტკიცოთ, რომ მოწინააღმდეგის ხელთ არსებული გამოთვლითი საშუალებებით შეუძლებელია შიფროტექსტიდან რაიმე ინფორმაციის მიღება ღია ტექსტის შესახებ.

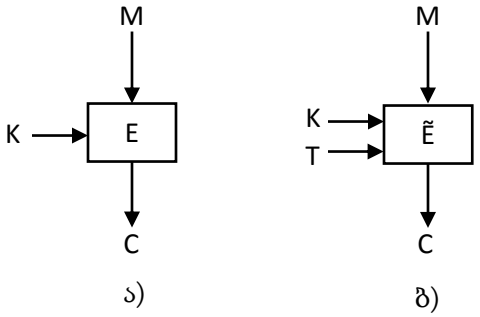
ღია ტექსტის სტრუქტურის დასამალად ყველაზე ეფექტურია ორი გარდაქმნის - მიმოფანტვის (confusion) და დიფუზიის (difusion) გამოყენება. მიმოფანტვა არის გარდაქმნა, რომლის მიზანია დამალოს კავშირი გასაღებსა და შიფროტექსტს შორის, ხოლო დიფუზიის მიზანია გახადოს შიფროტექსტის თოთეული სიმბოლო დამოკიდებული ღია ტექსტის ყველა სიმბოლოზე, რაც მოგვცემს საშუალებას დავმალოთ ღია ტექსტის სტრუქტურა. ამ მიზნების მისაღწევად თანამედროვე სიმეტრიულ კრიპტოგრაფიაში გამოიყენება ჩანაცვლების და გადანაცვლების ოპერაციები მრავალჯერადი იტერაციებით.

ბლოკური შიფრების ძირითად ნაკლს წარმოადგენს ის, რომ ისინი არიან დეტერმინირებული შიფრები, ანუ ერთი და იგივე გასაღებით ერთი და იგივე ღია ტექსტი ყოველთვის გადადის ერთსა და იმავე შიფროტექსტში. ეს განსაკუთრებით მნიშვნელოვანია როდესაც გვიხდება ერთი და იგივე შიფროტექსტის რამდენჯერმე გადაცემა (მცირეოდენი განსხვავებებით), რაც ძალიან ხშირად ხდება პრაქტიკაში. თუ ორივე შიფროგრამა ჩაუვარდება ხელში მოწინააღმდეგეს, მას პრაქტიკულად საშუალება აქვს შეუტიოს შიფრს ღია ტექსტის საფუძველზე. ასეთი შემთხვევები კრიპტოგრაფიის ისტორიაში ძალიან ბევრია (მაგალითად, ასე გატყდა პირველი მსოფლიო ომის დროს ფრიც ნებელის ADFGVX შიფრი).

წარმოვიდგინოთ ასეთი სიტუაცია, ღია არხით გადაიცემა ორ ბანკს შორის გადარიცხვები დაშიფრული სახით. ამ პროცედურაში ყველაფერი გაწერილია, სტანდარტულად, ამიტომ ორი დაშიფრული შეტყობინება იქნება ერთნაირი დასაწყისიდან,

სანამ არ შეგვხვდება კლიენტის ანგარიშის ნომერი და თანხის რაოდენობა, რომელიც გადაირიცხება. ცხადია ასეთი დაშიფრული ტექსტების გატეხვა არ გაუჭირდება მოწინააღმდეგეს. სწორედ ამიტომ თანამედროვე კომპიუტერული შიფრების შექმნასთან ერთად შეიქმნა ე.წ. კონფიდენციალურობის დაცვის რეჟიმები (სულ ხუთი რეჟიმი), რომლებიც განსაზღვრავენ, თუ როგორ უნდა იქნეს გამოყენებული კრიპტოგრაფიული ალგორითმი იმის და მიხედვით, თუ რომელ ინფორმაციულ ამოცანას ვწყვეტთ. კერძოდ, როდესაც საქმე ეხება დაშიფრული ინფორმაციის გადაცემას ღია არხში, გამოიყენება ე.წ. CBC რეჟიმი, რომლის დროსაც ღია ტექსტის ყველა ბლოკი დაწყებული მეორიდან, xor ოპერაციით იკრიბება წინა დაშიფრულ ბლოკთან და მხოლოდ შემდეგ იშიფრება. ამ შემთხვევაშიც კი ორი ღია ტექსტის საწყისი ბლოკები შეიძლება იყოს ერთნაირი და შესაბამისად დაშიფრული ტექსტის ბლოკებიც იქნება ერთნაირი. ამიტომ ამ რეჟიმში გათვალისწინებულია ინიციალიზაციის ვექტორი, რომელიც წარმოადგენს ბლოკის სიგრძის ბიტურ სტრიქონს, რომელიც ისევ XOR ოპერაციით იკრიბება ღია ტექსტის პირველ ბლოკთან, ამასთან, ეს ვექტორი ყოველი გადაცემის დროს აუცილებლად უნდა შეიცვალოს. ასეთ შემთხვევაში უკვე ერთი და იგივე ღია ტექსტი ერთი და იგივე გასაღების გამოყენების შემთხვევაშიც აღარ იქნება ერთნაირი. ანუ ალგორითმი ხდება ალბათური ალგორითმი.

2002 წელს მ. ლისკოვმა, რ. რაივესტმა და დ. ვაგნერმა შემოიტანეს ე.წ. tweakable ბლოკური შიფრების ცნება. ამ ტერმინით აღინიშნება ბლოკური ალგორითბი, რომელიც ღია ტექსტის დასაშიფრად გასაღების გარდა დამატებით იყენებს კიდევ ერთ, ე.წ. tweak სიდიდეს (იხ. სქემა 1), რომელიც შეიძლება იყოს როგორც საიდუმლო, ასევე ღია პარამეტრიც. ამასთან, განსხვავებით ინიციალიზაციის ვექტორისა, რომელიც ღია ტექსტთან იკრიბება მხოლოდ ერთხელ, დაშიფვრის დასაწყისში, tweak სიდიდე გამოყენება დაშიფვრის დროს რამდენჯერმე (სხვადასხვა), რაც კიდევ უფრო ზრდის იმის შესაძლებლობას, რომ შიფროტექსტის თითოეული ბიტი დამოკიდებული გახდება შესასვლელი ტექსტის მაქსიმალური რაოდენობის ბიტებზე, რაც კიდევ უფრო გაურთულებს მოწინააღმდეგეს გამოთვლებს.ყოველივე ეს საშუალებას გვაძლევს გავზარდოთ ერთ გასაღებზე გადაცემული შიფროტექსტების რაოდენობა.



სქემა 1. სტანდარტული(ა) და tweakable(ბ) შიფრების მარტივი სქემა

ამ ნაშრომში შევეხებით ერთ-ერთი ასეთი tweakable შიფრისთვის, რომელიც ამავე დროს იყენებს ჰილის ალგორითმის მოდიფიკაციას, რაუნდული გასაღებების დაგენერირების ალგორითმს და მის ეფექტურობა-ვარგისიანობას.

ჰილის ალგორითმი

როგორც ვიცით, კლასოკური კრიპტოგრაფიის ყველაზე სუსტ წერტილს წარმოადგენს ის ფაქტი, რომ როგორც მარტივი, ასევე პოლიალფაბეტური ჩანაცვლების ალგორითმებში შიფროტექსტის ერთ გამოსასვლელ სიმბოლოს განსაზღვრავს ღია ტექსტის ერთი სიმბოლო. ამის გამო ღია ტექსტის სტრუქტურა შიფროტექსტში ძალიან სუსტად იმალება, რაც მოწინააღმდეგეს საშუალებას აძლევს შეუტოს ამ შიფრებს მხოლოდ შიფროტექსტის საფუძველზე და თუ მას გააჩნია ერთსა და იმავე გასაღებზე დაშიფრული ტექსტების საკმარისი რაოდენობა, წარმატებით დააგვირგვინოს ეს შეტევა.

მეცხრამეტე საუკუნეში ინგლისელმა ფიზიკოსმა, სერ ჩარლზ უიტსონმა შექმნა პირველი ბიგრამული შიფრი, რომელშიც ღია ტექსტის ორი ასო ერთდროულად გადადიოდა შიფროტექსტის ორ ასოში, რაც რა თქმა უნდა ძალიან დიდი ნაბიჯი იყო ახალი ტიპის შიფრების განვითარებაში.

უკვე მეოცე საუკუნეში, 1929 წელს ამერიკელმა მათემატიკოსმა ლესტერ ჰილმა შეიმუშავა n-გრამული დაშიფვრის ალგორითმი, რომელიც წრფივი ალგებრის მარტივ

ოპერაციებს, კერძოდ მატრიცის ვექტორზე გადამრავლებას ეფუძნება. ალგორითმის საშუალებით შესაძლებელია შიფროტექსტის ერთი გამოსასვლელი სიმბოლო დამოკიდებული იყოს n ცალ შესასვლელ სიმბოლოზე. ამ მიზნით მან ღია ტექსტის ასოებს შეუსაბამა რიცხვები, ისე როგორც ამას აკეთებს კლასიკური კრიპტოგრაფიის მრავალი შიფრი. შემდეგ აიღო n ცალი რიცხვი (ღია ტექსტიდან) და გამოაცხადა ვექტორად. იმისათვის რომ დაეშიფრა ეს n ცალი რიცხვი , აიღო კვადრატული მატრიცა n×n-ზე და გაამრავლა ვექტორი მატრიცაზე მოდულით 26 (ასოების რაოდენობა ლათინურ ანბანში). მიიღო კვლავ n სიგრძის ვექტორი, რომელიც წარმოადგენს შიფროტექტს და მისი თითოეული სიმბოლო დამოკიდებულია შესასვლელი ვექტორის n სიმბოლოზე. ეს იყო ჰილის ალგორითმის ყველაზე მნიშვნელოვანი და არსებითი განსხვავება მანამდე არსებული ალგორითმებისგან. იმისათვის რომ შესაძლებელი იყოს გაშიფვრა, დამშიფრავ მატრიცას უნდა გააჩნდეს შებრუნებული მატრიცა მოდულით 26.

მაგალითად, თუ ჩვენ გვინდა რომ შიფროტექსტის თითოეული სიმბოლო დამოკიდებული იყოს სამ შესასვლელ სიმბოლოზე უნდა ავიღოთ A მატრიცა 3×3-ზე ისეთი, რომ $A * A^{-1} = E$, სადაც E არის ერთეულოვანი მატრიცა და გავამრავლოთ ღია ტექსტის სამ ასოიან(გადაყვანილ რიცხვებში) ტრიგრამზე.

$$M \times A = C$$

გაშიფვრა კი მოხდება ფორმულით:

$$C \times A^{-1} = M$$

რაც უფრო დიდი იქნება დამშიფრავი მატრიცის ზომა მით უფრო მეტი შესასვლელი სიმბოლო მიიღებს ერთი გამოსასვლელი სიმბოლოს გამოთვლაში მონაწილეობას და ღია ტექსტის სტრუქტურაც უკეთესად დაიმალება, მაგრამ ხელით დაშიფვრის დროს ამ ალგორითმის გამოყენება საკმაოდ რთულია და ამიტომ დამშიფრავი მატრიცის ზომა პატარაა რაც ართულებს დასახული მიზნის მიღწევას.

კომპიუტერული კრიპტოგრაფიის განვითარების პირველ ეტაპზე, ჰილის ალგორითმის გამოყენებაზე უარი ითქვა იმ მიზეზით, რომ ვექტორის მატრიცაზე გამრავლება წარმოადგენს წრფივ ოპერაციას და თუ ალგორითმში გამოვიყენებთ $n \times n$-ზე

მატრიცას, მის გასატეხად საჭირო იქნება მხოლოდ n^2 წრფივი განტოლების ამოხსნა, რაც კომპიუტერის შესაძლებლობებს თუ გავითვალისწინებთ, არც თუ ისე რთულია.

მაგრამ ბოლო წლებში გაჩნდა უამრავი შრომა, რომლებშიც ჰილის ალგორითმის სხვადასხვა მოდიფიკაცია გამოიყენება რომელიმე არაწრფივ ოპერაციასთან ერთად. ეს შეუძლებელს ხდის ალგორითმის მარტივად გატეხვას და ინარჩუნებს ჰილის ალგორითმის ყველა დადებით თვისებას და მათ შორის ძალიან მნიშვნელოვანს, ძალიან სწრაფად დიფუზიური გარდაქმნის შესრულებას.

ჰილის მოდიფიცირებული ალგორითმი

ერთერთი ასეთი ალგორითმი შეიქმნა ჩვენთანაც. ალგორითმის ბლოკის სიგრძეა 128 ბიტი, რომელიც წარმოიდგინება როგორც მდგომარეობის მატრიცა 4×4-ზე , რომლის ელემენტებიცაა ბაიტები.

$$M = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

აქ თითოეული a_{ij} წარმოადგენს რიცხვს ნულიდან 255-მდე $0 \le a_{ij} \le 255$. ეს მატრიცა მრავლდება A მატრიცაზე $M \times A \pmod{256}$, რომელიც წარმოადგენს თვითშებრუნებად მატრიცას. კერძოდ,

$$A = \begin{pmatrix} 2 & -1 & -2 & 2 \\ -1 & -2 & -2 & -2 \\ 1 & 1 & 1 & 2 \\ -1 & 1 & 2 & -1 \end{pmatrix} -$$

რადგანაც მატრიცის მატრიცაზე გამრავლება არაა ისეთი მარტივი ოპერაცია, როგორც ელემენტების გადასმა, ან xor-ით შეკრება, მატრიცის ელემენტები ისეა აღებული, რომ გამრავლება შესრულდეს რაც შეიძლება სწრაფად.

განვიხილოთ მარტივი მაგალითი. დავუშვათ,

$$M = \begin{pmatrix} 100 & 111 & 109 & 97 \\ 105 & 110 & 112 & 97 \\ 114 & 97 & 109 & 101 \\ 116 & 101 & 114 & 115 \end{pmatrix}$$

გავამრავლოთ ეს მატრიცა A მატრიცაზე მოდულით 256, მივიღებთ.

$$\begin{pmatrix} 100 & 111 & 109 & 97 \\ 105 & 110 & 112 & 97 \\ 114 & 97 & 109 & 101 \\ 116 & 101 & 114 & 115 \end{pmatrix} \times \begin{pmatrix} 2 & -1 & -2 & 2 \\ -1 & -2 & -2 & -2 \\ 1 & 1 & 1 & 2 \\ -1 & 1 & 2 & -1 \end{pmatrix} - \ mod\ 256 = \begin{pmatrix} 101 & 140 & 137 & 99 \\ 115 & 140 & 132 & 117 \\ 139 & 158 & 44 & 151 \\ 130 & 157 & 166 & 143 \end{pmatrix} mod\ 256,$$

შევადაროთ ერთმანეთს ბიტური სტრიქონები, რომლებიც შეესაბამებიან M მატრიცას და გამრავლების შემდეგ მიღებულ მატრიცასAnd bit string:

M= 01100100 01101111 01101101 01100001 01101001 01101110 01110000 01100001 01110010 01100001 01101101 01100101 01110100 01100101 01110010 01110011.

გამრავლების შედეგად მიღებულ მატრიცას კი შეესაბამება

01100101 10001100 10001001 01100011 01110011 10001100 10000100 01110101 10001011 10011110 00101100 10010111 10000010 10011101 10100110 10001111.

საწყისი ბიტური სტრიქონის 128 ბიტიდან გამრავლების შედეგად 67 ბიტმა განიცადა ცვლილება, რაც ძალიან კარგი შედეგია.

ამოცანის დასმა

სამაგისტრო ნაშრომის მიზანი იყო ასეთი ახალი ტიპის tweakable ბლოკური დაშიფვრის ალგორითმისთვის საწყისი გასაღებიდან რაუნდული გასაღებების გამომუშავების მექანიზმის შექმნა. ალგორითმში ხდება 128 ბიტიანი ბლოკის დაშიფვრა ასევე 128 ბიტიანი საიდუმლო გასაღებით. ალგორითმში შესვლის შემდეგ დასაშიფრი ბლოკი წარმოიდგინება არა ვექტორის, არამედ 4×4-ზე მატრიცის საშუალებით, რომელსაც ვუწოდებთ მდგომარეობის მატრიცას, სადაც თითოეული a_{ij} წარმოადგენს ოროობით ბაიტს.

დასაშიფრი ოროობითი სტრიქონი ჩაიწერება მატრიცაში მარცხნიდან მარჯვნივ ჰორიზონტალურად. ყველა ოპერაცია, რომელიც სრულდება ალგორითმში დასაშიფრ ტექსტზე სრულდება ამ მატრიცაზე. რაუნდების რაოდენობა არის 10. ე.ი. ჩვენ გვჭირდება 10 ერთმანეთისაგან განსხვავებული რაუნდული გასაღები რომელსაც გამოვიმუშავებთ საწყისი გასაღებიდან. განსხვაებული გასაღებები თითოეულ რაუნდში იმიტომ გვჭირდება რომ მოხდეს ღია ტექსტის სტრუქტურის დამალვა რაც შეიძლება უკეთესად. ღია ტექსტი დაგენერირებულ გასაღებთან შეიკრიბება XOR-ით რაუნდში შემოსვლისთანავე. მოთხოვნები რომლებსაც დაგენერირებული გასაღებები უნდა აკმაყოფილებდნენ:

1. ყველა რაუნდის გასაღები უნდა განსხვავდებოდეს ერთმანეთისაგან(მაგალითად ალგორითმ DES-ში არსებობს ოთხი საწყისი გასაღები რომელთაგანაც გამომუშავდება მხოლოდ ორი განსხვავებული რაუნდული გასაღები. შედეგად ერთი საწყისი გასაღებით დაშიფრული ტექსტი შეიძლება გაიშიფროს სხვა გასაღებით).
2. საწყისი გასაღების ყველა ბიტი დაახლოებით თანამბრად უნდა მონაწილეობდეს ათივე რაუნდული გასაღების გამომუშავებაში.
3. თუ საწყის გასაღებში შევცვალეთ ერთი ბიტი, რაუნდული გასაღებები მნიშვნელოვნად უნდა განსხვავდებოდნენ ერთმანეთისაგან.

პროგრამული რეალიზაცია

ნაშრომის ფარგლებში გაკეთდა მისი პროგრამული რეალიზაცია. კერძოდ, შეიქმნა მოდული დაპროგრამების ენა C#-ზე, რომელიც იქნება ძირითადი სისტემის ნაწილი და პასუხისმგებელი იქნება მიწოდებული საწყისი გასაღებიდან რაუნდული გასაღებების გამომუშავებაზე ჩვენს მიერ აღწერილი მეთოდით. ფიქსირებული მატრიცა და კონსტანტური ვექტორი შეინახება ამ მოდულის მხარეს. ძირითადმა სისტემამ მხოლოდ საწყისი გასაღების მოწოდება უნდა უზრუნველყოს.

რაუნდული გასაღებების გენერირება

გასაღებების გამოსამუშავებლად ვიყენებთ მატრიცული გამრავლების ოპერაციას. საწყისი გასაღების 128 ბიტი დავყოთ ბაიტებად. გვექნება 16 ბაიტი და ჩავწეროთ კვადრატული მატრიცის სახით. თითოეული ბაიტი გადავიყვანოთ ათობით სისტემაში და გავამრავლოთ რაიმე ფიქსირებულ კვადრატულ მატრიცაზე მოდულით 256. მიღებული მატრიცა კვლავ გადავიყვანოთ ბიტურ სტრიქონში და ეს ჩავთვალოდ პირველი რაუნდის გასაღებად. მეორე რაუნდის გასაღების მისაღებად პირველი რაუნდის გასაღები გავამრავლოთ იმავე ფიქსირებულ მატრიცაზე და მიღებული შედეგი ჩავთვალოთ მეორე რაუნდის გასაღებად. ჩავატაროთ იგივე მოქმედებები სანამ არ მივიღებთ ათივე რაუნდის გასაღებს.

კრიპტოანალიტიკოსმა ალგორითმის გატეხვის მიზნით შესაძლოა გამოიყენოს სტრუქტურულად შერჩეული გასაღები, მაგალითად, მთლიანად ერთიანებისგან შემდგარი საწყისი გასაღები. ამ მიზნით რაუნდული გასაღებების გამომუშავებამდე საწყის გასაღებს ყოველთვის XOR-ით ვკრებთ კონსტანტასთან(ბიტურ სტრიქონი) რომელიც წარმოადგენს ფსევდოშემთხვევით მიმდევრობას, რომელიც ასევე ინახება ფიქსირებული მატრიცის მსგავსად. ამ ოპერაციის შედეგი განხილულია ქვემოთ. ყოველივე ამის გათვალისწინებით ჩვენ ჩავატარეთ კვლევა.

ჩატარებული კვლევები

კვლევისას საწყის გასაღებად ხშირად აღებული იყო მთლიანად ერთიანებისგან შემდგარი გასაღები, რადგანაც შედეგები შეგვეფასებინა ყველაზე ცუდი ვარიანტისთვის. ამის მიზეზია ის, რომ რეალობაში ასეთი გასაღები არ უნდა გამოვიყენოთ, რადგან ასეთი გასაღები ვერ ჩაითვლება შემთხვევით მიმდევრობად და მოწინააღმდეგისთვისაც ადვილად გამოცნობადი იქნება, მაგრამ შესაძლოა მოწინააღმდეგემ ალგორითმის გატეხვის მიზნით სწორედ ასეთი სტრუქტურულად შერჩეული გასაღები გამოიყენოს.

რაუნდული გასაღებების უნიკალურობა

ავიღეთ სხვადასხვა საწყისი გასაღებები მათ შორის ნახსენები მთლიანად ერთიანებისგან შემდგარი გასაღებიც და ასევე სხვადასხვა ფიქსირებული მატრიცა და დავაგენერირეთ რაუნდული გასაღებები ზემოთ მოყვანილი მეთოდით. ყველაზე მნიშვნელოვანი შედეგი მივიღეთ ის რომ არ ხდება რაუნდული გასაღებების დამთხვევა, ანუ ყოველი რაუნდისთვის ათიდან ჩვენ გვაქვს შესაძლებლობა გამოვიყენოთ უნიკალური რაუნდული გასაღები, მიუხედავად იმისა თუ როგორ საწყის გასაღებს ან ფიქსირებულ მატრიცას ავიღებთ.

საწყის გასაღებში ბიტების შეცვლით მიღებული შედეგები

საწყის გასაღებში ერთი ბიტის ცლილება არ გვაძლევს რაუნდულ გასაღებებში მნიშვნელოვან ცვლილებას. განსხვავება დაახლოებით არის 2.1 ბიტი. რა თქმა უნდა უმჯობესი იქნებოდა თუ განსხვავება მნიშვნელოვანი იქნებოდა, მაგრამ რადგანაც ერთი ბიტის ცვლილება საწყისი გასაღების სტრუქტურას თითქმის არ ცვლის ამიტომ ეს შედეგი მოსალოდნელიც იყო.

მაგალითი: საწყისი გასაღების ყველა ბიტი უდრის ერთს. ფიქსირებული მატრიცა შემთხვევითადაა აღებული. მოხდა რაუნდული გასაღებების დაგენერირება. შედეგები:

საწყისი გასაღები:

11111111 11111111 11111111 11111111

11111111 11111111 11111111 11111111

11111111 11111111 11111111 11111111

11111111 11111111 11111111 11111111

ფიქსირებული მატრიცა:

11101110 01110111 10100101 10101101

00011000 00110011 10110110 01010100

00100100 10010001 11110010 01000011

00010000 11001101 10100101 11101011

რაუნდული გასაღებები:

1

11000110 11111000 00001110 11010001

11000110 11111000 00001110 11010001

11000110 11111000 00001110 11010001

11000110 11111000 00001110 11010001

2

01011100 10111101 11011111 10110011

01011100 10111101 11011111 10110011

01011100 10111101 11011111 10110011

01011100 10111101 11011111 10110011

3

11001100 00010001 11010111 11011110

11001100 00010001 11010111 11011110

11001100 00010001 11010111 11011110

11001100 00010001 11010111 11011110

4

01011100	11000100	11100110	01111111
01011100	11000100	11100110	01111111
01011100	11000100	11100110	01111111
01011100	11000100	11100110	01111111

5

00110000	11001001	11101011	01000011
00110000	11001001	11101011	01000011
00110000	11001001	11101011	01000011
00110000	11001001	11101011	01000011

6

10110100	00011101	00101011	01100110
10110100	00011101	00101011	01100110
10110100	00011101	00101011	01100110
10110100	00011101	00101011	01100110

7

01111100	01111100	00000110	00001011
01111100	01111100	00000110	00001011
01111100	01111100	00000110	00001011

01111100	01111100	00000110	00001011

8

01110000	10001101	11010111	00100111
01110000	10001101	11010111	00100111
01110000	10001101	11010111	00100111
01110000	10001101	11010111	00100111

9

00000100	00101001	11001111	00000110
00000100	00101001	11001111	00000110
00000100	00101001	11001111	00000110
00000100	00101001	11001111	00000110

10

00001100	00010100	01000110	11010111
00001100	00010100	01000110	11010111
00001100	00010100	01000110	11010111
00001100	00010100	01000110	11010111

საწყის გასაღებში შეიცვალა ერთი(პირველი) ბიტის მნიშვნელობა 0-ით, ფიქსირებული მატრიცა იგივე დარჩა და კვლავ დაგენერირდა რაუნდული გასაღებები. შედეგები:

საწყისი გასაღები:

01111111	11111111	11111111	11111111
11111111	11111111	11111111	11111111
11111111	11111111	11111111	11111111
11111111	11111111	11111111	11111111

რაუნდული გასაღებები:

1

11000110	01111000	10001110	01010001
11000110	11111000	00001110	11010001
11000110	11111000	00001110	11010001
11000110	11111000	00001110	11010001

2

01011100	00111101	01011111	10110011
01011100	10111101	11011111	10110011
01011100	10111101	11011111	10110011
01011100	10111101	11011111	10110011

3

11001100	00010001	11010111	01011110
11001100	00010001	11010111	11011110
11001100	00010001	11010111	11011110

11001100 00010001 11010111 11011110

4

01011100 01000100 01100110 11111111

01011100 11000100 11100110 01111111

01011100 11000100 11100110 01111111

01011100 11000100 11100110 01111111

5

00110000 01001001 01101011 01000011

00110000 11001001 11101011 01000011

00110000 11001001 11101011 01000011

00110000 11001001 11101011 01000011

6

10110100 00011101 00101011 11100110

10110100 00011101 00101011 01100110

10110100 00011101 00101011 01100110

10110100 00011101 00101011 01100110

7

01111100 11111100 10000110 10001011

01111100	01111100	00000110	00001011
01111100	01111100	00000110	00001011
01111100	01111100	00000110	00001011

8

01110000	00001101	01010111	00100111
01110000	10001101	11010111	00100111
01110000	10001101	11010111	00100111
01110000	10001101	11010111	00100111

9

00000100	00101001	11001111	10000110
00000100	00101001	11001111	00000110
00000100	00101001	11001111	00000110
00000100	00101001	11001111	00000110

10

00001100	10010100	11000110	01010111
00001100	00010100	01000110	11010111
00001100	00010100	01000110	11010111
00001100	00010100	01000110	11010111

ამ მაგალითში გამოვიყენეთ ორი საწყისი გასაღები ერთი(პირველი) ბიტით განსხვავებული. შედეგად მათგან დაგენერირებული რაუნდული გასაღებები საშუალოდ 2.1 ბიტით განსხვავდებიან

ორი ბიტის შეცვლა საწყის გასაღებში შედარებით უკეთეს შედეგს გვაძლევს მაგრამ ცდებში გამოჩნდა რომ რაც უფრო დაშორებულები არიან ეს ბიტები ერთმანეთს, ძირითადად განსხვავებასაც მით უფრო დიდს გვაძლევენ. მაგალითად ერთიანებისგან შემდგარი მატრიცის შემთხვევაში პირველი და ბოლო ბიტის შეცვლამ სხვადასხვა ფიქსირებული მატრიცისტვის საშუალოდ მოგვცა განსხვავება 18.05 ბიტი. იგივე ცდამ ყველა ერთნაირი ბაიტის მქონე საწყისი გასაღების შემთხვევაში მოგვცა შემდეგი შედეგები: პირველი ორი ბიტის შეცვლის შემთხვევაში საშუალოდ 3.6 ბიტის განსხვავება, პირველი და მეოთხე ბიტის შეცვლამ - 7.3 ბიტი, პირველი და მეშვიდეს შეცვლამ 15.3 ბიტი

მაგალითი: საწყისი გასაღების ყველა ბიტი უდრის ერთს. ფიქსირებული მატრიცა შემთხვევითაა აღებული. მოხდა რაუნდული გასაღებების დაგენერირება. შედეგები:

საწყისი გასაღები:

11111111 11111111 11111111 11111111

11111111 11111111 11111111 11111111

11111111 11111111 11111111 11111111

11111111 11111111 11111111 11111111

ფიქსირებული მატრიცა:

11101110 01110111 10100101 10101101

00011000 00110011 10110110 01010100

00100100 10010001 11110010 01000011

00010000 11001101 10100101 11101011

რაუნდული გასაღებები:

1

11000110	11111000	00001110	11010001
11000110	11111000	00001110	11010001
11000110	11111000	00001110	11010001
11000110	11111000	00001110	11010001

2

01011100	10111101	11011111	10110011
01011100	10111101	11011111	10110011
01011100	10111101	11011111	10110011
01011100	10111101	11011111	10110011

3

11001100	00010001	11010111	11011110
11001100	00010001	11010111	11011110
11001100	00010001	11010111	11011110
11001100	00010001	11010111	11011110

4

01011100	11000100	11100110	01111111
01011100	11000100	11100110	01111111
01011100	11000100	11100110	01111111

01011100 11000100 11100110 01111111

5

00110000 11001001 11101011 01000011

00110000 11001001 11101011 01000011

00110000 11001001 11101011 01000011

00110000 11001001 11101011 01000011

6

10110100 00011101 00101011 01100110

10110100 00011101 00101011 01100110

10110100 00011101 00101011 01100110

10110100 00011101 00101011 01100110

7

01111100 01111100 00000110 00001011

01111100 01111100 00000110 00001011

01111100 01111100 00000110 00001011

01111100 01111100 00000110 00001011

8

01110000 10001101 11010111 00100111

01110000 10001101 11010111 00100111

01110000 10001101 11010111 00100111

01110000 10001101 11010111 00100111

9

00000100 00101001 11001111 00000110

00000100 00101001 11001111 00000110

00000100 00101001 11001111 00000110

00000100 00101001 11001111 00000110

10

00001100 00010100 01000110 11010111

00001100 00010100 01000110 11010111

00001100 00010100 01000110 11010111

00001100 00010100 01000110 11010111

საწყის გასაღებში შეიცვალა ორი(პირველი და ბოლო) ბიტის მნიშვნელობა 0-ით, ფიქსირებული მატრიცა იგივეა. კვლავ დაგენერირდა რაუნდული გასაღებები. შედეგები:

საწყისი გასაღები:

01111111 11111111 11111111 11111111

11111111 11111111 11111111 11111111

11111111 11111111 11111111 11111111

11111111 11111111 11111111 11111110

რაუნდული გასაღებები:

1

11000110 01111000 10001110 01010001

11000110 11111000 00001110 11010001

11000110 11111000 00001110 11010001

10110110 00101011 01101001 11100110

2

01011100 00111101 01011111 10110011

01011100 10111101 11011111 10110011

01011100 10111101 11011111 10110011

01100000 11010010 01100000 10110111

3

11001100 00010001 11010111 01011110

11001100 00010001 11010111 11011110

11001100 00010001 11010111 11011110

11100000 01100001 11011111 11100101

4

01011100 01000100 01100110 11111111

01011100 11000100 11100110 01111111

01011100 11000100 11100110 01111111

00000100 00100011 10111101 11001000

5

00110000 01001001 01101011 01000011

00110000 11001001 11101011 01000011

00110000 11001001 11101011 01000011

00010100 00001010 00001000 00111111

6

10110100 00011101 00101011 11100110

10110100 00011101 00101011 01100110

10110100 00011101 00101011 01100110

10011000 01000101 00101011 10111001

7

01111100 11111100 10000110 10001011

01111100 01111100 00000110 00001011

01111100 01111100 00000110 00001011

01100100 11100111 11101001 01110000

8

01110000 00001101 01010111 00100111

01110000 10001101 11010111 00100111

01110000 10001101 11010111 00100111

01100100 00101010 00100000 00101011

9

00000100 00101001 11001111 10000110

00000100 00101001 11001111 00000110

00000100 00101001 11001111 00000110

00011000 01101001 01000111 00110101

10

00001100 10010100 11000110 01010111

00001100 00010100 01000110 11010111

00001100 00010100 01000110 11010111

01110100 10111011 01100101 11101000

ამ მაგალითში ორი საწყისი გასაღები გამოვიყენეთ ოღონდ ორი ბიტით(პირველი და ბოლო) განსხვავებული. შედეგად მათგან დაგენერირებული გასაღებები საშუალოდ 18.3 ბიტით განსხვავდებიან.

ფიქსირებული მატრიცის შერჩევა

იმის დასადგენად თუ როგორ შეგვერჩია ფიქსირებული მატრიცა ჩატარდა ცდები. ერთიდაიგივე საწყისი გასაღებისთვის, რომელიც შემთხვევითად იყო დაგენერირებული, შეირჩა რამდენიმე ფიქსირებული მატრიცა(ასევე შემთხვევითი) და დაგენერირდა რაუნდული გასაღებები. შემდეგ საწყის გასაღებში შეიცვალა მეხუთე და მეცხრე ბაიტების ბოლო ბიტები და კვლავ დაგენერირდა რაუნდული გასაღებები. ყველა ფიქსირებული მატრიცის შემთხვევაში განსხვავებები იყო თითქმის იგივე, საშუალოდ 31.1 ბიტი. სხვადასხვა ფიქსირებული მატრიცის შედეგებს შორის სხვაობა არ აღემატებოდა 2.2 ბიტს. რადგანაც ფიქსირებული მატრიცის შეცვლამ არ გამოიწვია რადიკალური და მნიშვნელოვანი განსხვავებები ე.ი. მისი შერჩევა შეიძლება თავისუფლად, შემთხვევით დაგენერირებული მატრიცებიდან.

მაგალითი: საწყისი გასაღები შემთხვევითაა აღებული. ფიქსირებული მატრიცაც შემთხვევითაა აღებული. დაგენერირდა რაუნდული გასაღებები. შედეგები:

საწყისი გასაღები:

00101100 01110110 00010001 00001110

00001111 00110010 00011000 11100010

11111101 00100000 00011100 10001010

11000111 01000000 10101111 10100001

ფიქსირებული მატრიცა:

11101110	01110111	10100101	10101101
00011000	00110011	10110110	01010100
00100100	10010001	11110010	01000011
00010000	11001101	10100101	11101011

რაუნდული გასაღებები:

1

00111100	11001101	01011000	11000001
00100010	10000001	10010001	01001001
11000110	01011001	00111011	01111011
10101110	01001101	11110110	00010011

2

01110000	00100000	11111111	00000011
10101000	00010111	10111111	01000100
01101000	10101111	11110001	01011100
11000100	11000110	10101111	10101101

3

00101100	01000110	11101101	10101110
01110100	01010000	00000100	01111101
10111100	01100010	10010000	00111011

00110100 00110110 00000111 00001000

4

10101100 11111001 01010000 01110101

10111000 00111001 11111101 01101111

11101000 10111001 11111111 00001101

11100100 01001101 10101110 00001001

5

11010000 10010000 11101011 01000111

11101100 00010011 11010011 00101000

10110100 10001011 01111101 00101000

00111000 00010110 11111011 00100101

6

01011100 01010110 01011001 01111110

01011100 00001000 11011100 10101001

01110100 00110010 11001000 10101111

10111100 00110110 11011011 10111000

7

11111100 00110101 11001000 01011001

11001000 01001101 11100001 10000011

10011000 01001101 00101011 11001001

00100100 10001001 00101110 11111101

8

11110000 01000000 10000111 10111011

11111100 10100111 11000111 10010000

00100100 01001111 11101001 11000000

10011000 10101110 00100111 10010001

9

11001100 10000110 01010101 00101110

11101100 01110000 00010100 01011101

10100100 00110010 01100000 01111011

00101100 10000110 11111111 00100000

10

00001100 10000001 11000000 01001101

10001000 11010001 10010101 11010111

01011000 00010001 01000111 01000101

01010100 00110101 01001110 11010001

შეიცვალა მეხუთე და მეცხრე ბაიტების ბოლო ბიტები, ფიქსირებული მატრიცა იგივეა. კვლავ დაგენერირდა რაუნდული გასაღებები. შედეგები:

საწყისი გასაღები:

00101100	01110110	00010001	00001110
00001110	00110010	00011000	11100010
11111100	00100000	00011100	10001010
11000111	01000000	10101111	10100001

რაუნდული გასაღებები:

1

00111100	11001101	01011000	11000001
00110100	00001010	11101100	10011100
11011000	11100010	10010110	11001110
10101110	01001101	11110110	00010011

2

01110000	00100000	11111111	00000011
00111000	11000010	01000100	01100100
11111000	01011010	01110110	01111100
11000100	11000110	10101111	10101101

3

00101100	01000110	11101101	10101110

00010000 01000110 11000000 00011000

01011000 01011000 01001100 11010110

00110100 00110110 00000111 00001000

4

10101100 11111001 01010000 01110101

11110000 01011010 00001100 00010000

00100000 11011010 00001110 10101110

11100100 01001101 10101110 00001001

5

11010000 10010000 11101011 01000111

01000000 00011010 01010100 10001100

00001000 10010010 11111110 10001100

00111000 00010110 11111011 00100101

6

01011100 01010110 01011001 01111110

10000000 10011110 01100000 01001000

10011000 11001000 01001100 01001110

10111100 00110110 11011011 10111000

7

11111100	00110101	11001000	01011001
11010000	00000010	11111100	10010000
10100000	00000010	01000110	11010110
00100100	10001001	00101110	11111101

8

11110000	01000000	10000111	10111011
00000000	00100010	10000100	01011100
00101000	11001010	10100110	10001100
10011000	10101110	00100111	10010001

9

11001100	10000110	01010101	00101110
10000000	00110110	01000000	00101000
00111000	11111000	10001100	01000110
00101100	10000110	11111111	00100000

10

00001100	10000001	11000000	01001101
10010000	10001010	00101100	10110000
01100000	11001010	11011110	00011110

01010100 00110101 01001110 11010001

ამ ორი მცირედ განსხვავებული საწყისი გასაღებით შვენ მივიღეთ საშუალოდ 31.9 ბიტით განსხვავებული რაუნდული გასაღებები. ჩვენი მიზანია დავადგინოთ სხვა ფიქსირებული მატრიცების შემთხვევაში რა შედეგი გვექნება. ამ მიზნით იგივე მაგალითს განვიხილავთ, იგივე მონაცემებით და ცვლილებებით, მხოლოდ შევცვლით ფიქსირებულ მატრიცას.

ფიქსირებული მატრიცა:

10001101 10111101 01101100 01110010

00011100 11100000 01111101 10011000

00001000 10100111 10101011 11101000

00011110 00101011 11001110 10100101

რაუნდული გასაღებები:

1

01010000 00101101 11001101 00010110

11110111 01110001 10100010 11001000

11100101 00111011 00011100 11111100

11110001 00011111 10100111 11111011

2

11111000 11011101 01011100 01001110

11100111 10000001 10000111 11001110

11111101 01001001 11100111 11001110

11000011 00100111 01010110 11011001

3

11001000	10010110	11000001	01001110
10110011	00010110	01100010	10010100
10110001	11110100	01110010	00100000
11001001	10100100	01011111	11001011

4

10111100	11101001	01001101	01001110
01100111	00110001	11010000	11111010
01111101	11101011	10110110	10100010
01100111	11110111	10101111	11010001

5

10010100	00000001	01001000	00011110
11100011	10011001	01111101	10011000
00111001	11011001	01101001	10001100
10110101	01101111	00100010	11010011

6

01100100	00100110	00101001	00010110
01111011	10001010	01001000	00101110

11010001 11111000 11001100 10011110

10011111 01100000 00001111 01010001

7

00011000 10000101 11010101 01101110

01111011 01000001 01100010 10011100

00100001 11101011 10101100 00001000

00001001 11000111 00100111 10011011

8

01010000 10000101 11011100 10010110

00110011 11010001 10011111 10111010

00110001 10001001 11111111 01000010

00011011 00111111 10111110 01101001

9

00010000 00100110 01011001 10100110

10110111 01111110 01110010 11001000

10101101 01111100 00000010 11001100

00000001 10100100 10001111 01001011

10

00110100 00000001 01010101 01010110

10010011 01010001 11010000 10000110

11010001 11001011 00000110 11110110

10111111 00011111 01011111 11000001

ბიტების ცვლილების შემდეგ:

რაუნდული გასაღებები:

1

01010000 00101101 11001101 00010110

01101010 10110100 00110110 01010110

01011000 01111110 10110000 10001010

11110001 00011111 10100111 11111011

2

11111000 11011101 01011100 01001110

11010110 01101110 11100010 01110010

11101100 00110110 01000010 01110010

11000011 00100111 01010110 11011001

3

11001000 10010110 11000001 01001110

01010010 11010010 10110000 11100110

01010000 10110000 11000000 01110010

11001001 10100100 01011111 11001011

4

10111100 11101001 01001101 01001110

10010110 10111100 11000110 11110010

10101100 01110110 10101100 10011010

01100111 11110111 10101111 11010001

5

10010100 00000001 01001000 00011110

10111010 00001110 00010010 11010110

00010000 01001110 11111110 11001010

10110101 01101111 00100010 11010011

6

01100100 00100110 00101001 00010110

10011110 01000010 10001000 01100010

11110100 10110000 00001100 11010010

10011111 01100000 00001111 01010001

7

00011000 10000101 11010101 01101110

11111010 10010100 10010110 11110110

10100000 00111110 11100000 01100010

00001001 11000111 00100111 10011011

8

01010000 10000101 11011100 10010110

01100110 00111110 11100010 10110010

01100100 11110110 01000010 00111010

00011011 00111111 10111110 01101001

9

00010000 00100110 01011001 10100110

11100010 11100010 10000000 11000110

11011000 11100000 00010000 11001010

00000001 10100100 10001111 01001011

10

00110100 00000001 01010101 01010110

01100110 01011100 10000110 01110010

10100100 11010110 10111100 11100010

10111111 00011111 01011111 11000001

ამჯერად მივიღეთ 31.7 ბიტით განსხვავებული რაუნდული გასაღებები. ამით ვაჩვენებთ რომ ფიქსირებული მატრიცის შესარჩევად რაიმე განსაკუთრებული კვლევის ჩატარება არაა საჭირო და შედარებით მარტივად, შემთხვევითად დაგენერირებულის გამოყენებაც შესაძლებელია.

კონსტანტის შემოღება შერჩეული გასაღებით მანიპულირების შესაზღუდად

მოწინააღმდეგემ ალგორითმის გატეხვის მიზნით შესაძლოა სცადოს შერჩეული გასაღებების გამოყენება, რომ დააკვირდეს გასაღების ცვლილება ერთი და იგივე ღია ტექსტის დაშიფვრის დროს რა ცვლილებებს იწვევს შიფრო ტექსტში. ამ მიზნით შესაძლოა მან გამოიყენოს ზემოთ ნახსენები, ერთიანებისგან შემდგარი გასაღებიც(ერთ ერთი ცუდი ვარიანტი). შესაძლოა დაშიფროს ღია ტექსტი ამ გასაღებით. შემდეგ მოახდინოს ერთი ბიტის ცვლილება გასაღებში და დააკვირდეს რა განსხვავებას მისცემს შიფროტექსტში ამ ერთი ბიტის ცვლილება. იმისათვის რომ მას არ ქონდეს გასაღების შერჩევით შეტევის განხორციელების საშუალება და ვერ შეძლოს უშუალოდ მის მიერ შერჩეული გასაღებით ზემოქმედება ღია ტექსტზე, შემოვიღეთ ასეთი მექანიზმი: გვაქვს 128 ბიტიანი შემთხვევითი მიმდევრობა, რომელსაც ვუწოდებთ კონსტანტას და ვინახავთ ფიქსირებული მატრიცის მსგავსად. რაუნდული გასაღებების გამომუშავებამდე საწყის გასაღებს XOR-ით ვკრებთ ამ კონსტანტასთან და შედეგად ვიღებთ საკმაოდ განსხვავებულ გასაღებს, რითიც მოწინააღმდეგეს ეზღუდება თავისი შერჩეული გასაღებით მანიპულირების საშუალება. ჩატარებულმა ცდებმა მოგვცა ასეთი შედეგი: ერთიანებისგან შემდგარი საწყისი გასაღების შემთხვევაში რამდენიმე ფიქსირებული მატრიცის და რამდენიმე კონსტანტის გამოყენებისას საყისი გასაღების კონსტანტასთან XOR-ით შეკრებით მიღებული გასაღები საშუალოდ 63 ბიტით განსხვავდება საწყისი გასაღებისგან, ხოლო რაუნდული გასაღებების საშუალო განსხვავებაა 56.9 ბიტი. ეს შედეგი დამაკმაყოფილებელია და მოწინააღმდეგეს მნიშვნელოვნად გაურთულებს გასაღებით მანიპულაციების ჩატარების საშუალებას.

მაგალითი: ამ მაგალითში ჩვენი მიზანია ვაჩვენოთ აქვს თუ არა კონსტანტის შემოღებას აზრი. საწყისი გასაღების ყველა ბიტი უდრის ერთს. ფიქსირებული მატრიცა შემთხვევითაა აღებული. მოხდა რაუნდული გასაღებების დაგენერირება. შედეგები:

საწყისი გასაღები:

11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111

ფიქსირებული მატრიცა:

11101110	01110111	10100101	10101101
00011000	00110011	10110110	01010100
00100100	10010001	11110010	01000011
00010000	11001101	10100101	11101011

რაუნდული გასღებები:

1

11000110	11111000	00001110	11010001
11000110	11111000	00001110	11010001
11000110	11111000	00001110	11010001
11000110	11111000	00001110	11010001

2

01011100	10111101	11011111	10110011
01011100	10111101	11011111	10110011
01011100	10111101	11011111	10110011
01011100	10111101	11011111	10110011

3

11001100	00010001	11010111	11011110
11001100	00010001	11010111	11011110
11001100	00010001	11010111	11011110
11001100	00010001	11010111	11011110

4

01011100	11000100	11100110	01111111
01011100	11000100	11100110	01111111
01011100	11000100	11100110	01111111
01011100	11000100	11100110	01111111

5

00110000	11001001	11101011	01000011
00110000	11001001	11101011	01000011
00110000	11001001	11101011	01000011
00110000	11001001	11101011	01000011

6

10110100	00011101	00101011	01100110
10110100	00011101	00101011	01100110

10110100 00011101 00101011 01100110

10110100 00011101 00101011 01100110

7

01111100 01111100 00000110 00001011

01111100 01111100 00000110 00001011

01111100 01111100 00000110 00001011

01111100 01111100 00000110 00001011

8

01110000 10001101 11010111 00100111

01110000 10001101 11010111 00100111

01110000 10001101 11010111 00100111

01110000 10001101 11010111 00100111

9

00000100 00101001 11001111 00000110

00000100 00101001 11001111 00000110

00000100 00101001 11001111 00000110

00000100 00101001 11001111 00000110

10

00001100 00010100 01000110 11010111

00001100 00010100 01000110 11010111

00001100 00010100 01000110 11010111

00001100 00010100 01000110 11010111

ამის შემდეგ ავიღეთ 128 ბიტიანი შემთხვევითი სტრიქონი და XOR-ით დავუმატეთ საწყის გასაღებს და კვლავ დავაგენერირეთ რაუნდული გასაღებები. შედეგები:

საწყისი გასაღები:

11111111111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111

კონსტანტა:

00010110001011101011000000001111111101100000100001101101001010101111000111110110010010000110101110100101000101101100110101011001

XOR-ით შეკრების შედეგი

11101001110100010100111111110000000010011111011110010010110101010000111000001001101101111001010001011010111010010011001010100110

რაუნდული გასაღებები:

1

01010010 11100001 00100001 00000110

01011110 10100111 10110100 11011110

11011000 01111000 11001110 00101011

11101100 10000001 11101010 10111110

2

01011000	01110000	11100000	01100011
00111100	10110001	10001110	00111000
10111000	01101101	11111011	10111011
01001000	00010111	01111100	01111000

3

00000000	01011111	11100111	10111001
11011000	01101101	11010110	00110010
01000100	00101001	11100011	01110110
00001000	01100001	01010010	11010000

4

11110100	11101001	00100101	01110100
01000000	01011111	00111100	10100100
01011100	11011000	10011110	00100011
00010000	00001101	10110010	10100010

5

00100100	10110000	10101000	10000011
00011000	11111101	00110110	10101100
00110000	01010001	11000111	10000111
01000000	10010011	00111100	01100000

6

11001000	11011011	10010011	01001101
01100000	11100001	00111110	01000010
10100100	01000101	10100111	00000110
10111000	11100101	01011010	01010000

7

11110100	10000101	00110001	00101100
00110000	01101011	11111100	10000100
11001100	01100000	01111110	10101111
00110000	00110001	00001010	01111010

8

11110100	11101000	10000000	10111111
01011000	00010001	01001110	10101100
01010000	01110101	10100011	11111011
01000000	01101111	11011100	00100000

9

10001000	00010111	01001111	11011001
00100000	00110101	01100110	01011010

11110100 11010001 10011011 10000110

11011000 00011001 11000010 10100000

10

01000100 01010001 10001101 01010100

10110000 01000111 10111100 01010100

10011100 00101000 10111110 00001011

01110000 01100101 10000010 11010010

შედეგად მივიღეთ რომ საწყისი გასაღებები ერთმანეთისგან განსხვავდება 63 ბიტით, ხოლო რაუნდული გასაღებები კი 56.9 ბიტით. ე.ი. კონსტანტის შემოღება ეფექტური გამოდგა.

საწყისი გასაღების ბიტების მონაწილეობა რაუნდული გასაღებების გამოთვლაში

როგორც უკვე ვთქვით, გასაღებების გამომუშავებას ვახდენთ მატრიცული გადამრავლების ოპერაციით. მატრიცის ელემენტები, როგორც საწყისი გასაღების, ისე ფიქსირებული მატრიცის, წარმოადგენენ ბაიტებს(სულ 16 ბაიტი). იმის დასადგენად თუ საწყისი გასაღების ერთი ბიტი რამდენი ბიტის გამოთვლაში იღებს მონაწილეობას რაუნდულ გასაღებებში, უნდა გავითვალისწინოთ მატრიცების გადამრავლების წესი(სტრიქონი სვეტზე). რაუნდული გასაღების პირველი ელემენტის(1 ბაიტი) მისაღებად საწყისი გასაღების პირველი სტრიქონი(4 ბაიტი) გადამრავლდება ფიქსირებული მატრიცის პირველ სვეტზე(4 ბაიტზე). მეორე ელემენტის მისაღებად საწყისი გასაღების ისევ პირველი სტრიქონი ფიქსირებული მატრიცის, უკვე მეორე სვეტზე. და ა.შ. ბოლო ელემენტის მიღებამდე პირველ სტრიქონში(ჩვენს შემთხვევაში მატრიცის განქომილებაა 4×4-ზე, ამიტომ საწყისი გასაღების პირველ სტრიქონს მოუწევს ფიქსირებული მატრიცის ოთხ სვეტზე გადამრავლება). პირველი

სტრიქონი მეტ მონაწილეობას სხვა ელემენტების გამოთვლაში აღარ ღებულობს. ე.ი. პირველი სტრიქონის 32-ივე ბიტმა მიიღო ასევე 32 ბიტის გამოთვლაში მონაწილეობა რაუნდულ გასაღებში. ზუსტად იგივე ხდება სხვა სტრიქონების შემთხვევაშიც - 1 ბიტი ამ სტრიქონიდან იღებს 32 ბიტის გამოთვლაში რაუნდულ გასაღებში. ე.ი. საწყისი გასაღების ყველა(და არა ერთი) ბიტი ჩართულია რაუნდული გასაღების ყველა ბიტის გამოთვლაში. ამასთან რაუნდებს შორის ნარჩუნდება დამოკიდებულება ბიტებს შორის, ანუ თუ საწყისი გასაღების 1-მა ბიტმა მიიღო მონაწილეობა პირველი რაუნდის გასაღების 32 ბიტის გამოთვლაში, თავის მხრივ პირველი რაუნდის გასაღების 32 ბიტი, რომელიც დამოკიდებულია იმ ერთ ბიტზე, მონაწილეობას მიიღებს მეორე რაუნდის გასაღების 32 ბიტის გამოთვლაში, მეორის 32 ბიტი - მესამის 32 ბიტის და ა.შ. შედეგად მივიღებთ რომ საწყის გასაღებში ამ ერთი ბიტის ცვლილება მოგვცემს ათივე რაუნდში ცვლილებას იმ 32 ბიტში რომლების გამოთვლაშიც მან მიიღო მონაწილეობა.

საწყის და რაუნდულ გასაღებებს შორის სხვაობა

ჩვენ უკვე ვახსენეთ, რომ საწყისი გასაღებიდან დაგენერირებული რაუნდული გასაღებები არ უნდა მეორდებოდნენ და უნდა იყვნენ უნიკალურები, მაგრამ ამასთან ერთად მნიშვნელოვანი საკითხია აგრეთვე რამდენად განსხვავდებიან ისინი ერთმანეთისგან. როგორც ვთქვით რაუნდული გასაღების დასაგენერირებლად პირველი რაუნდის შემთხვევაში ვიყენებთ საწყის გასაღებს ხოლო ყოველი შემდგომი რაუნდის გასაღებისთვის წინა რაუნდის გასაღებს, ასე რომ ჩვენთვის მნიშვნელოვანია სხვაობა გასაღებს და მისგან მიღებულ შემდეგი რაუნდის გასაღებს შორის. ჩატარდა ცდები რომლებშიც მონაწილეობდა როგორც შემთხვევითად დაგენერირებული გასაღებები, ისე უკვე მრავალჯერ ნახსენები შერჩევითი გზით აღებული გასაღებები და შედეგები არის ასეთი: გასაღებს რომელიც მონაწილეობს სხვა გასაღების დაგენერირებაში და მისგან დაგანერირებულ გასაღებს შორის სხვაობა საშუალოდ არის 54.36 ბიტი. როგორც შერჩევით აღებულ გასაღებს, ისე შემთხვევით გასაღებს შორის ეს მონაცემი თითქმის იგივე სიდიდისაა. ე.ი. საკმარისად განსხვავებული რაუნდული გასაღებების გამომუშავება ხდება ამ მეთოდით, რაც ასევე მისაღებია.

მაგალითი: გამოვთვალოთ რაიმე გასაღების და ფიქსირებული მატრიცის შემთხვევაში რაუნდული გასაღებები და ვნახოთ განსხვავება მათ შორის.

საწყისი გასაღები:

00101100011101100001000100001110000011100011001000011000111000101111110000100000000011100100010101100011101000000101010111101000001

ფიქსირებული მატრიცა:

10001101101111010110110001110010000111001110000001111101100110000000100010100111101010111110100000011110001010111100111010100101

რაუნდული გასაღებები:

1

განსხვავება: 58 ბიტი

01010000001011011100110100010110011010101011010000110110010101100101100000111111010110000100010101111000100011111101001111111011

2

განსხვავება: 58 ბიტი

11111000110111010101110001001110110101100110111011100010011100101110110000110110010000100111001011000011001001110101011011011001

3

განსხვავება: 43 ბიტი

1100100010010110110000010100111001010010110100101011000011100110010100001011000011000000011100101100100110100100010111111100101 1

4

განსხვავება: 63 ბიტი

1011110011101001010011010100111010010110101111001100011011110010101011000111011010101100100111010011001111111101111010111111010001

5

განსხვავება: 48 ბიტი

1001010000000001010010000001111010111010000011100001001011010110000100000100111011111111011001010101101010110111100100010110100 11

6

განსხვავება: 56 ბიტი

0110010000100110001010010001011010011111001000010100010000110001011110100101100000000110011010010100111111011000000000111101010001

7

განსხვავება: 64 ბიტი

0001100010000101110101010110111011111010100101001001011011110110101000000011111011100000011000100000100111000111001001111001101 1

8

განსხვავება: 51 ბიტი

0101000010000101110111001001011001100110001111101110001010110010011001001111011001000010000111010000110110011111111011111001101001

9

განსხვავება: 52 ბიტი

0001000000100110010110011010011011100010111000101000000011000110110110001110000000010000110010100000000110100100100011110100101 1

10

განსხვავება: 59 ბიტი

0011010000000000101010101010101100110011001011100100001100111001010100100110101101011110011100010101111110000111110101111111000001

ამ მაგალითში საშუალოდ 55.2 ბიტით განსხვავდებიან რაუნდული გასაღებები რაც დამაკმაყოფილებელი შედეგია.

კვლევის შედეგები და მათი განხილვა

ჩვენი მიზანია დავადგინოთ რამდენად ეფექტური და ვარგისია ზემოთ განხილული გასაღებების გამომუშავების მეთოდი.

ამ მეთოდით ხდება უნიკალური რაუნდული გასაღებების დაგენერირება რაც საკმაოდ მნიშვნელოვანი საკითხია.

მიუხედავად იმისა, რომ სასურველი იყო, მაგრამ საწყის გასაღებში ერთი ბიტის შეცვლით არ ხდება მნიშვნელოვანი ცვლილებები რაუნდულ გასაღებებში, ეს ვერ ჩაითვლება ამ მეთოდის სისუსტედ, რადგან ერთი ბიტის შეცვლა არ ცვლის მთელი გასაღების სტრუქტურას და დაახლოებით მოსალოდნელიც იყო ეს შედეგი.

ამ მეთოდის კიდევ ერთი დადებითი თვისება აღმოჩნდა ის, რომ ფიქსირებული მატრიცის შერჩევისთვის არაა საჭირო რაიმე დიდი კვლევის და სამუშაოების ჩატარება, საკმარისია რაიმე შემთხვევითად დაგენერირებული მატრიცის აღება, რადგანაც ცდების მიხედვით დაახლოებით ერთნაირ შედეგს გვაძლევს ყველა.

ასევე გაამართლა კონსტანტის შემოღებამ და საწყისი გასაღების შეკრებამ მასთან XOR-ით, რაც მოწინააღმდეგეს უზღუდავს გასაღებით მანიპულირებაში მოქნილობას.

დამაკმაყოფილებელია ის შედეგიც, რომ საწყისი გასაღების ყველა ბიტი იღებს მონაწილეობას რაუნდული გასაღებების ბიტების გამოთვლაში.

კიდევ ერთი მნიშვნელოვანი დადებითი თვისება ამ მეთოდის არის ის რომ გასაღებს და მისგან დაგენერირებულ შემდეგი რაუნდის გასაღებს შორის არსებობს საკმაო სხვაობა, რაც მეტად უზრუნველყოფს ღია ტექსტის სტრუქტურის დამალვას შიფროტექსტში.

ამ ყველაფრის გათვა;ოსწინებით დადგინდა, რომ გასაღებების გამომუშავების ეს მეთოდი ძირითადად აკმაყოფილებს ჩვენს მოთხოვნებს რის საფუძველზეც მიგვაჩნია, რომ მისი გამოყენება შესაძლებელია ჩვენი ამოცანისთვის.

დასკვნა

აღწერილი პროცედურა, რომელიც გამოიყენება ჰილის ალგორითმზე დაფუძნებული ახალი სიმეტრიული tweakable ალგორითმისთვის რაუნდული გასაღებების მისაღებად, შეიძლება ჩაითვალოს რომ არის დამაკმაყოფილებელი. მითუმეტეს რომ საწყისი გასაღები საიდუმლოა და არც მასთან და არც რაუნდულ გასაღებებთან მოწინააღმდეგეს წვდომა არ აქვს.

გამოყენებული ლიტერატურა

1. ჯულაყიძე ლ.ე., ქოჩლაძე ზ.ი., კაიშაური თ.ვ. ახალი tweakable ბლოკური შიფრის აგება ჰილის მოდიფიცირებული ალგორითმის გამოყენებით. საქართველოს საინჟინრო სიახლენი #1 (vol.73), 2015. გვ. 40-43.
2. ჯულაყიძე ლ.ე., ქოჩლაძე ზ.ი., კაიშაური თ.ვ. ახალი სიმეტრიული tweakable ბლოკური შიფრი. საქართველოს საინჟინრო სიახლენი #1 (vol.73), 2015. გვ.44-49.
3. Lester S. Hill Cryptography in an algebraic Alphabet. The American Mathematical Monthly Vol.56 #6 (1929) pp. 306-312.
4. Bibhudendra Acharya, SarojkumarPanigrahy, SaratkumarPatra, and Canapsti Panda Image Encryption Using Advanced Hill Cipher Algorithm. International Journal of Recent Trends in Engineering. Vol.1, No.1, May 2009. pp.663-667.
5. Zurab Kochladze Modified version of the Hill's algorithm. GESJ. 2014 09, #3 (43)
6. M. Blaze, W. Diffie, R.L. Rivest, B. Schneier, T. Shinonura, E. Thompson, M. Wiener Minimal key lengths for Symmetric Ciphers to Provide Adequate Commercial Security. A Report by an Ad Hoc Group of Cryptographers and Computer Scientists. January 1996.
7. M. Liskov, R.L.Rivest, D. Wagner Twaekable Block Ciphers Journal of Cryptology 24(3), July 2011.