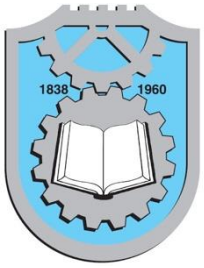
Универзитет у Крагујевцу
ФАКУЛТЕТ ИНЖЕЊЕРСКИХ НАУКА



Семинарски рад из предмета:

Програмирање система који раде у реалном времену

Хеш алгоритми
Типови и поређење

Професор:
Проф. Владимир Миловановић

Студент:
Матеја Вујсић 617/2017

Крагујевац**, јун 2020. године**

Садржај

1. Увод…… 3

2. Принцип рада хеш функција……..4

2.1 SHA-1 принцип рада.....5

2.2 SHA-2, SHA-3.....7

2.3 MD5.....8

3. Слабости алгоритама.....9

3.1 РОЂЕНДАСКИ НАПАД -БФ алгоритам.....10

3.2 СУДАРИ И ОТПОРНОСТ НА СУДАРЕ.....11

4.Примена у дигиталном свету.....12

4.ЛИТЕРАТУРА.....13

1. УВОД

Хеширање је процес превођења улазног податка било које величине(дужине) у низ битова фиксне дужине коришћењем специјално дизајниране математичке функције. Порука која се треба хеширати назива се улаз, математичка функција кроз коју се улаз пропушта назива се хеш функција(хеш алгоритам) а производ улаза и хеш функције је хеш вредност. Фајл било које дужине може бити приказан као низ бројева и слова фиксне дужине што се показало као веома корисно.

За овај процес дизајнирано је на десетине алгоритама и неки од најпознатијих који се данас користе су SHA породица алгоритама хеширања, MD5, CRC32,MD4, TIGER, Adler32,RipeMD128…

Међутим све алгоритми хеширања морају да задовољавају одређене критеријуме да би добили одређену функцију у савременом дигиталном свету. Свака хеш функција мора да тежи томе да на свом излазу да уникатну вредност. Ово значи да функцију треба пројектовати тако да је немогуће да за два различита улаза добијемо исту хеш вредност. Једна те иста порука мора увек да да исту хеш вредност. Такође, битно је да дати алгоритам генерише хеш вредност веома брзо, и што мање зависи од величине улаза.

Веома битна ставка у оцењивању хеш алгоритма је и његова сигурност. На основу хеш вредности не сме се ништа знати о подацима који су хеширан. Исто алгоритам се формира тако да мала измена улазних података драстично измени хеш вредност (ефекат “лавине”). Промена једног бита у улазним подацима мења изглед хеш вредности.

Данас се овакви алгоритми користе да би осигурали интегритет поруке која се преноси преко не тако сигурног канала комуникације ,где се често дешава да трећа страна мења трансмитоване поруке. У даљем току рада биће објашњени принципи рада, захтеви, примена и доказане мане хеш функција које се данас користе.

2. Принципи рада хеш функција

Свака функција за хеширање или **message digest (MD)** мора да задовољава четири главна критеријума, а то су:

1. **За задато П, лако се израчунава MD(П).**

2. **За задато MD(П), практично је немогуће наћи П.**

3. **За задато П, нико не може да израчуна такво П' да важи MD(П)=MD(П').**

4. **Измена улазних података чак и за један бит даје веома различит резултат.**

За **испуњење трећег услова,** резултат хеширања мора бити дужине барем 128 битова тј. 16 алфанумеричких карактера.

Да би се **испунио четврти услов**, функција за хеширање мора добро да промеша битове, прилично слично као код алгоритама за шифровање симетричним кључем.

Свака функција која испуњава ове услове такође мора да се има задовољиву временску комплексност. Не треба бити превише једноставна у

мешању битова а не треба бити ни превише сложена, пошто неки поруке које је потребно сублимирати имају и до неколико гигабајта.

2.1 SHA-1-принцип рада

AAF4C61DDCC5E8A2DABEDE0F3B482CD9AEA9434D

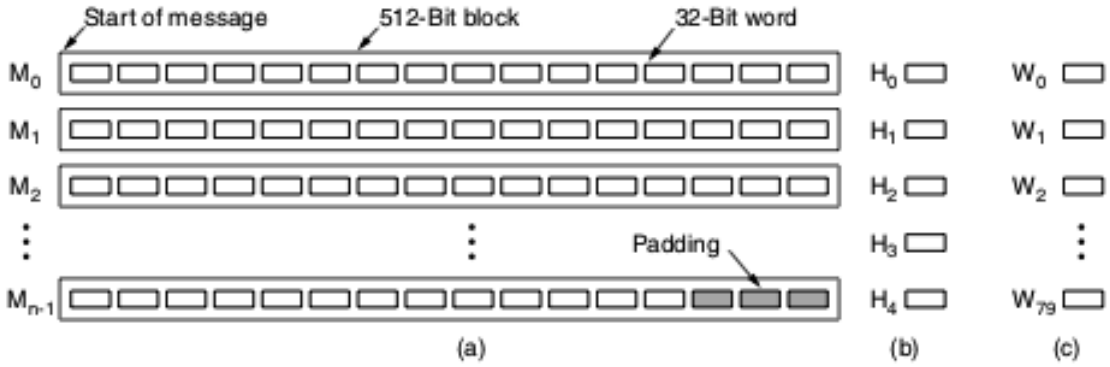
Сл. 1 Генерисана SHA1 хеш за стринг “hello”

SHA-1(Secure Hash Algorithm 1- безбедности алгоритам за хеширање 1) је један од најпопуларнијих алгоритама за хеширање. Слично свим осталим алгоритмима за сажимање порука и SHA-1 меша битове на начин довољно компликован да на сваки излазни бит утиче сваки улазни бит. Алгоритам SHA-1 развила је организација NSA, а агенција НИСТ објавила га је 1995. године. Данас је овај метод хеширања података застарео и пун мана и превазишле су га новије генерације SHA. [1]

Алгоритам обрађује улазне податке у блоковима од 512 битова и генерише сажетак поруке дужине 160 битова (20 карактера,сл.1).

Размотримо сада укратко како ради SHA-1 .Почиње тако што на крај поруке додаје бит 1 и онолико нула колико је потребно да њена дужина

постане **умножак од 512 битова**.



сл.2. Порука допуњена до умношка од 512 битова(а),(б)Промењива за прихватање резултата. (ц) Низ рачунарских речи

Затим се израчунава дисјункција(операција OP) између 64-битног броја који представља дужину поруке пре допуњавања и најмање значајна 64 бита поруке који се затим замењује тим резултатом. SHA-1 води рачуна о архитектури па увек допуњава крај поруке независно да ли је то "big-endian" или "little-endian".

Током израчунавања, SHA-1 одржава пет 32-битних промењивих (од H0-H4) у којима се акумулира резултат хеширања (сл2.б). Оне се иницијализују константама које прописује стандард и увек су на почетку исте.

Онда се редом обрађује сваки блок ,од M_0 до M_{n-1}. За први блок се прво копира 16 речи помоћног низа W (дужине 80 речи) , као на сл2(ц) .Затим се низ попуњава са следеће 64 речи према формули:

$$W_i = S^1(W_{i-3} \text{ XOR } W_{i-8} \text{ XOR } W_{i-14} \text{ XOR } W_{i-16}) \quad (16 \le i \le 79)$$

где је s^b(W) представља циркуларно(повратно) ротирање улево 32-битне речи W за b-битова. Онда се пет помоћних променљиве (од A-E) иницијализују вредностима променљивих од H_0 -H_4.

Само израчунавање може се приказати у облику псеудокода на језику C:

```
for (i = 0; i < 80; i++) {
    temp = S^5(A) + f_i (B, C, D) + E + W_i + K_i;
    E = D;  D = C;  C = S^30(B);  B = A;  A = temp;
}
```

где су константе Ki дефинисане стандардом. Функције Fi дефинисане су следећим изразима:

$$f_i(B,C,D) = (B \text{ AND } C) \text{ OR } (\text{NOT } B \text{ AND } D) \qquad (0 \le i \le 19)$$
$$f_i(B,C,D) = B \text{ XOR } C \text{ XOR } D \qquad (20 \le i \le 39)$$
$$f_i(B,C,D) = (B \text{ AND } C) \text{ OR } (B \text{ AND } D) \text{ OR } (C \text{ AND } D) \qquad (40 \le i \le 59)$$
$$f_i(B,C,D) = B \text{ XOR } C \text{ XOR } D \qquad (60 \le i \le 79)$$

Када се заврши свих 80 итерација петље, вредности променљивих од A-E додају се (XOR) вредностима одговарајућих променљивих од H0-H4.

Пошто је тако обрађен први 512-битни блок, почиње обрада следећег. При томе, W низ се поново иницијализује новим блоком, али H низ се узима из претходног. По завршетку обраде тог блока прелази се на следећи, и тако до краја поруке. По завршетку обраде последњег блока, од пет 32-битних речи из низа H формира се 160-битни криптографски хеширан сажетак. Потпуна имплементација SHA-1 у C језику може се наћи у RFC документу 3174.

2.2 SHA-2, SHA-3

Развијене су и нове верзије алгоритама SHA које производе хешеве дужине 224,256,384,512 бита. Те верзије носе заједничко име SHA-2. Не само да су ти хешеви дужи од SHA-1(160 битова) ,него је измењена и функција за сажимање да би се уклониле неке потенцијалне слабости алгоритама SHA-1. 2015. године

у званичну употребу ушла и трећа генерација ових алгоритама која превазилази све потенцијалне недостатке претходних генерација.

```
2CF24DBA5FB0A30E26E83B2AC5B9E29E1B161E5C1FA7425E73043362938B9824
```

сл.3 SHA256- хеш вредност за стринг "hello"

2.3 MD-5

MD-5 је још један популаран алгоритам за сажимање порука. Осмишљен је давне 1992, и сада већ припада историји информатичке мисли. Ово је први алгоритам за сажимање порука који је нашао широку употребу, историјски дошао је пре SHA алгоритма и нека решења такође су врло слична са овом групом алгоритама. Дужина хеш вредности је 128 битова, дакле 16 карактера. Једна карактеристика овог алгоритма је што се у функцијама мешања јавља и функција синуса. Наиме, творац овог алгоритма Роналд Ривест је на овај начин хтео да прикаже да није направио "мала врата" на која само он може да уђе.

```
5D41402ABC4B2A76B9719D911017C592
```

сл.4 MD-5- хеш вредност за стринг "hello"

3. ХЕШ ФУНКЦИЈЕ – СЛАБОСТИ

Постоје неколико типова напада на хеш вредности. Овде ћемо их напоменути. Неки су прилично интуитивни, док се неки заснивају на теорији вероватноћа. То су:

- ПРВИ ТИП – Нападач зна хеш вредност улаза и покушава да нађе исти улаз који даје исту хеш вредност. Овај тип напада се обично одиграва када имамо украдене хеш вредности шифре неког налога и покушавамо да преко доброг WordList-а да нађемо шифру.
- ДРУГИ ТИП – Нападач је прибавио фајл и хеш вредност истог и покушава да нађе сличан фајл са истом хеш вредности. Рецимо да хоћемо да неки фајл заменимо сличним у који је уграђен spyware.
- **COLLISIONS -Судари** - Нападач налази два улаза који дају исту хеш вредност. Ово је једна од битних ставки која ће бити обрађена додатно.
- Продужавање дужине поруке – Нападач зна дужину поруке M и вредност потписа и покушава да допуни поруку новом поруком H тако да ће попунити празне битове који алгоритам на почетку пуни и онда ће да дода нову поруку. Овај тип напада обично се примењује кад се зна формат поруке.SHA-1,SHA2,MD5 су веома повољни за овај тип напада.

Рецимо, поред вишедеценијске употребе неке хеш функције показале су своје недостатке. Слабости алгоритма MD5 (1992) довеле су до откривања судара, тј. различитих порука са једнаким хешом те се са тренутном технологијом судари откривају за око 17 секунди (2018). Другим речима, сматра се да је MD5 разбијен и где год је то могуће потребно га је заменити. Упркос томе, MD5 се и даље користи у постојећим системима.

Function	Output Size	Security Strengths in Bits		
		Collision	Preimage	2nd Preimage
SHA-1	160	< 80	160	$160-L(M)$
SHA-224	224	112	224	$\min(224, 256-L(M))$
SHA-512/224	224	112	224	224
SHA-256	256	128	256	$256-L(M)$
SHA-512/256	256	128	256	256
SHA-384	384	192	384	384
SHA-512	512	256	512	$512-L(M)$
SHA3-224	224	112	224	224
SHA3-256	256	128	256	256
SHA3-384	384	192	384	384
SHA3-512	512	256	512	512
SHAKE128	d	$\min(d/2, 128)$	$\geq \min(d, 128)$	$\min(d, 128)$
SHAKE256	d	$\min(d/2, 256)$	$\geq \min(d, 256)$	$\min(d, 256)$

сл5. Различите функције и сигурности од напада [2]

3.1 РОЂЕНДАСКИ НАПАД -БФ алгоритам*

У свету шифровања ништа није онако како изгледа. Прва помисао да је довољно отприлике 2^mоперација да би се фалсификовао сажетак поруке од m-битова. Међутим, често је довољно $2^{m/2}$ операција ако се примени ткз. **РОЂЕНДАНСКИ НАПАД .**

Основна идеја за овај напад је питање које се најчешће поставља:Колико људи би требало да се налазе у неком затвореном простору тако да са сигурношћу можемо да тврдимо да се налазе бар две особе које су рођене истог дана? Први интуитивно нагађање би било преко 100? Узмимо за пример 50. Логиком, 365 дана у години/50 људи, дакле негде око 17%.Међутим грешимо, вероватноћа да ће се наћи је 97%!

Ако упоредимо сваки дан рођења са сваким другим даном рођења у соби добијемо:

$$C_k(n) = \binom{n}{k} = \frac{n!}{k!(n-k)!}$$

$$n = 50$$
$$k = 2$$

$$C_2(50) = \binom{50}{2} = \frac{50!}{2!(50-2)!} = \frac{50 \cdot 49}{2 \cdot 1} = 1225$$

добијамо 1,225 комбинација. Велика је вероватноћа да ће једна комбинација бити права тј. иста. Проблем можемо и да уопштимо. Ако постоји одређено пресликавање између улазних и излазних података, при чему је n улазних а k излазних података, постоји n(n-1)/2 улазних парова . Ако је n(n-1)/2>k ,вероватноћа да постоји барем једно поклапање је прилично велика.

3.2 СУДАРИ И ОТПОРНОСТ НА СУДАРЕ

Више поменути судари дешавају се када два различита улаза(фајл, отворен текст) дају исту хеш вредност.

Свака хеш функција која има више комбинација улаза него излаза (могућности за улаз су неограничене, а ипак морају се пресликати на велики али ограничени скуп) гарантовано има неке различите улазе који генеришу исту хеш вредност (PigeonHole principle).

Као што је већ речено Парадокс Рођендана формира горњу границу за БФ-напад на хеш тј. број случаја насумичних улаза које генерално морамо испитати да би се десио сигурно десио судар. Дакле, као нападач морамо испитати $2^{n/2}$ случаја на насумичан улаз да би се највероватније десио судар. Сматра се да је хеш функција лоша ако постоји бољи поступак за налажење судара од БФ-алгоритма. Мада неке функције као што је SHA-3 генерација доказује да је тешко наћи судар као исто као што је тешко наћи решење за рецимо факторизацију целих бројева. Трошкови су превелики, па се за овакве функције кажу да су доказано сигурне.

Застареле генерације хеш функција као што су MD5 и SHA-1 су разбијене. Нађен је поступак за налажење судара који јефтиније кошта од трошкова БФ алгоритма .[3][4]

4.ПРИМЕНА У ДИГИТАЛНОМ СВЕТУ

Хеш функције су екстремно корисне и појављују се у скоро свим безбедносно-сигурним апликацијама. Пошто је израчунавање сажетка поруке из делића отвореног текста много брже од шифровања тог истог текста јавним кључем, сажеци порука се могу искористити за убрзавање дигиталног потписивања.

Многе друштвене мреже користе различите хеш функције да складиште шифре корисника. Ово је веома ефективан начин да се шифре складиште, пошто сама фирма која одржава нема базу података шифара у отвореном тексту већ у хешу. Међутим, већ показане слабости хеша као што су судари доказали су да није само довољно да се шифре складиште у хешу, технологија је напредовала да рецимо можемо да применимо рођендански напад. Зато многе фирме користе технике под називом **SOLTING-мењају plain-text,додају неке симболе и затим примењују хеш.**

4.ЛИТЕРАТУРА

[1] Tanenbaum A. "Racunarske mreze", Mikroknjiga, Beograd, Srbija 2013., str. 783-788.

[2] IMG:Tutorialspoint.com/cryptohraphy/cryptography_hash_funcitons.

[3] Xiaoyun Wang; Honbho Yu. „How to break MD5 and other Hash function“ 2009-12-21

[4] Xiaoyun Wang, Yinqun Lisa Yin.”Finding Collisions in Full SHA-1”