UNIVERSITETI I PRISHTINËS “HASAN PRISHTINA”FAKULTETI I SHKENCAVE MATEMATIKO-NATYROREDEPARTAMENTI I MATEMATIKËSPROGRAMI: SHKENCA KOMPJUTERIKE

 **PUNIM SEMINARIK**Lënda: Siguri e të dhënave  
TEMA: SHA-1

**Punuar nga: Albina Grajqevci, Diellza Bajraktari, Dalina Elshani**

**Maj 2023**

Përmbajtja

[**1.** **Përshkrimi i detyrës** 2](#_Toc135781865)

[**2.** **Hyrje** 2](#_Toc135781866)

[**3.** **SHA-1** 3](#_Toc135781867)

[**4.** **Implementimi i SHA-1** 4](#_Toc135781868)

[**5.** **Preimage resistance** 6](#_Toc135781869)

[**6.** **Second preimage resistance** 8](#_Toc135781870)

[**7.** **Collision resistance** 11](#_Toc135781871)

# **Përshkrimi i detyrës**

Detyra 3: Bazuar në skemën e algoritmit simplified SHA-1

1. Shkruani aplikacionin që do të implementojë skemën e dhënë? (2 pikë)
2. Le të jetë dhënë vlera , gjeni H(M) = 4BAFE69C, gjeni M’ ashtu që H(M’) = H(M)?
3. Zgjedhni një mesazh të gjatësisë së çfarëdoshme pastaj llogaritni hash vlerën më anë të Simplified SHA-1, pra e llogaritni H(M). Pastaj të gjindet M’ ashtu që M ≠ M dhe H(M’) = H(M)?

# **Hyrje**

Në epokën dixhitale, siguria e informacionit është e një rëndësie të madhe. Pavarësisht nëse bëhet fjalë për mbrojtjen e të dhënave të ndjeshme, sigurimin e integritetit të skedarëve ose verifikimin e autenticitetit të mesazheve, algoritmet kriptografike luajnë një rol vendimtar. Një algoritëm i tillë që ka fituar përdorim dhe njohje të gjerë është Algoritmi Secure Hash (SHA).

SHA është një familje e funksioneve hash kriptografike të zhvilluara nga Agjencia e Sigurisë Kombëtare (NSA) në Shtetet e Bashkuara. Qëllimi i një funksioni hash është të marrë një hyrje, zakonisht një sekuencë të dhënash me gjatësi të ndryshueshme, dhe të prodhojë një dalje me madhësi fikse të quajtur një vlerë hash ose digest. Ky përmbledhje është unik për të dhënat hyrëse, që do të thotë se edhe një ndryshim i vogël në hyrje do të rezultojë në një vlerë të konsiderueshme hash të ndryshme.

Aplikimet kryesore të SHA rrotullohen rreth integritetit të të dhënave dhe nënshkrimeve dixhitale. Duke gjeneruar një vlerë hash për një pjesë të caktuar të të dhënave, të tilla si një skedar ose një mesazh, SHA mundëson verifikimin e integritetit të tij. Nëse të dhënat ngatërrohen në ndonjë mënyrë, vlera e hash-it që rezulton do të jetë krejtësisht e ndryshme, duke siguruar një mekanizëm të besueshëm për zbulimin e ndryshimeve.

Për më tepër, SHA përdoret gjerësisht për krijimin e nënshkrimeve dixhitale, të cilat shërbejnë si një mjet për të vërtetuar origjinën dhe integritetin e informacionit dixhital. Duke gjeneruar një vlerë hash për një mesazh dhe duke e enkriptuar atë me një çelës privat, krijohet një nënshkrim dixhital. Ky nënshkrim mund të deshifrohet duke përdorur çelësin publik përkatës dhe integriteti dhe autenticiteti i mesazhit mund të verifikohet duke krahasuar vlerën hash të deshifruar me një të llogaritur rishtazi.

Me kalimin e viteve, janë zhvilluar përsëritje të shumta të algoritmit SHA, ku secila ofron siguri dhe rezistencë të shtuar ndaj sulmeve të ndryshme. Versionet më të përdorura janë SHA-1, SHA-256, SHA-384 dhe SHA-512, secila prej të cilave ofron gjatësi të ndryshme hash dhe nivele sigurie. Ndërsa fuqia llogaritëse dhe teknikat e analizës kriptografike përparojnë, nevoja për funksione hash më të forta bëhet e dukshme, duke nxitur miratimin e varianteve më të sigurta si SHA-3.

# **SHA-1**

Në fushën e kriptografisë, SHA-1 (Secure Hash Algorithm 1) ka luajtur një rol të rëndësishëm si një funksion hash i miratuar gjerësisht. Zhvilluar nga Agjencia Kombëtare e Sigurisë (NSA) dhe botuar nga Instituti Kombëtar i Standardeve dhe Teknologjisë (NIST) në 1995, SHA-1 ka qenë instrumental në aplikime të shumta sigurie.

Qëllimi kryesor i SHA-1 është të gjenerojë një vlerë hash me madhësi fikse, zakonisht 160 bit në gjatësi, nga një mesazh hyrës me madhësi arbitrare. Kjo vlerë hash shërben si një gjurmë gishti dixhital ose përmbledhje e mesazhit origjinal, duke mundësuar operacione të ndryshme kriptografike dhe duke ofruar integritet, autenticitet dhe siguri të të dhënave.

SHA-1 përdor një seri operacionesh logjike dhe aritmetike për të transformuar mesazhin hyrës në vlerën përfundimtare të hash-it. Këto operacione përfshijnë funksione logjike bitwise (AND, OR, XOR), shtimin modular, operacionet e rrotullimit dhe zgjerimin e mesazhit. Algoritmi përpunon mesazhin hyrës në blloqe dhe përditëson një gjendje të brendshme përmes disa raundeve të llogaritjes, duke rezultuar në vlerën përfundimtare të hash-it.

SHA-1 fitoi popullaritet të gjerë dhe pa përdorim të gjerë në aplikacione të ndryshme, të tilla si protokollet e komunikimit të sigurt, nënshkrimet dixhitale dhe kontrollet e integritetit të të dhënave. Megjithatë, ndërsa fuqia kompjuterike dhe teknikat e analizës kriptografike përparuan, u zbuluan dobësi në SHA-1.

Me kalimin e kohës, studiuesit demonstruan ndjeshmërinë e SHA-1 ndaj sulmeve të përplasjes, ku mesazhe të ndryshme hyrëse prodhojnë të njëjtën vlerë hash. Kjo ngriti shqetësime për sigurinë e SHA-1 dhe aftësinë e tij për t'i bërë ballë sulmeve me qëllim të keq. Rrjedhimisht, përdorimi i SHA-1 në sistemet dhe aplikacionet kritike të sigurisë është hequr gradualisht dhe është zëvendësuar me funksione hash më të sigurta.

Vlen të përmendet se ndërsa SHA-1 nuk konsiderohet më i sigurt për aplikacionet kritike të sigurisë, ai ende gjen një përdorim në kontekste që nuk lidhen me sigurinë ose sisteme të vjetra. Megjithatë, është thelbësore që zhvilluesit dhe profesionistët e sigurisë të jenë të vetëdijshëm për kufizimet dhe dobësitë që lidhen me SHA-1 dhe të kalojnë në funksione më të forta hash për aplikacionet që kërkojnë siguri të fortë kriptografike.

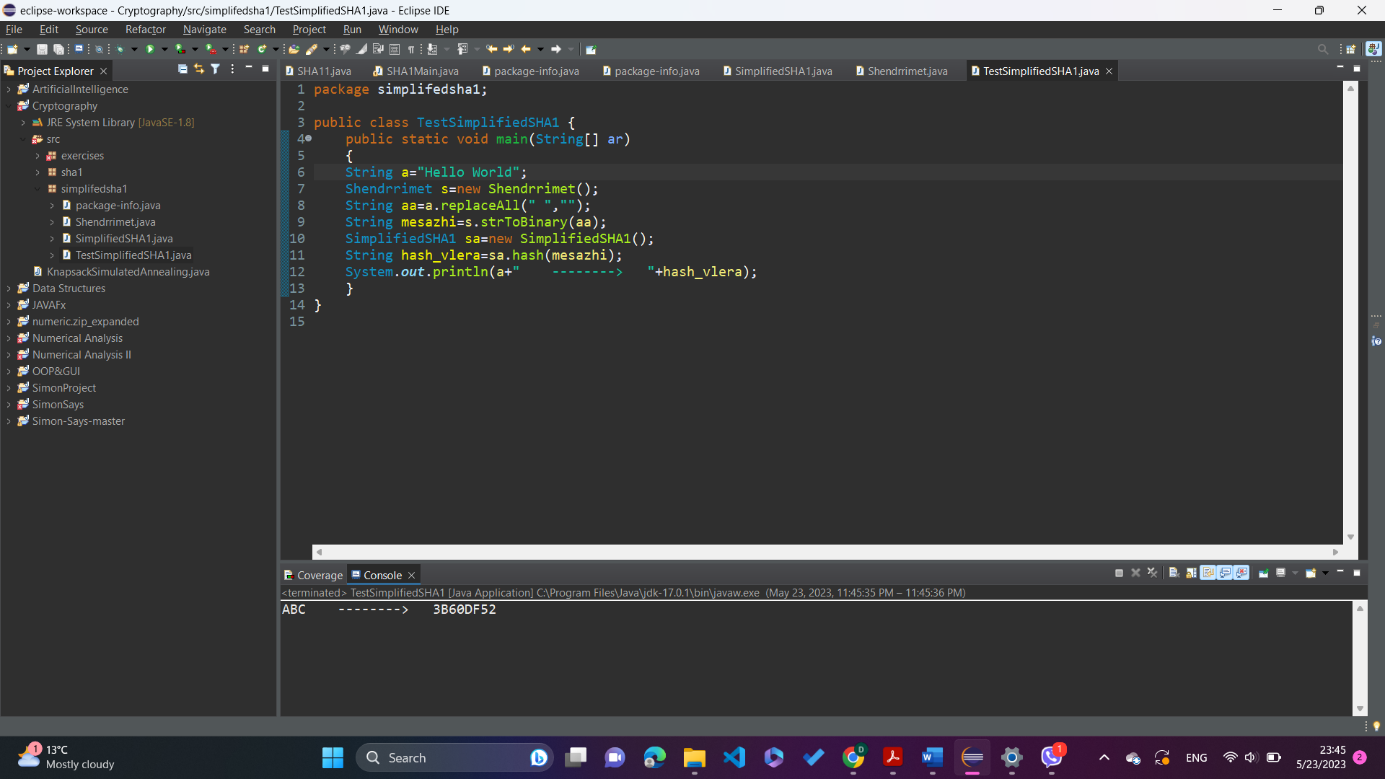
Në përmbledhje, SHA-1 luajti një rol të rëndësishëm në sistemet kriptografike, duke ofruar një mjet për të gjeneruar vlera hash me madhësi fikse nga mesazhet hyrëse. Ndërsa përdorimi i tij i përhapur tregoi efektivitetin e tij, zbulimi i dobësive kërkon kujdes dhe miratimin e funksioneve më të sigurta hash në aplikacionet moderne të sigurisë.

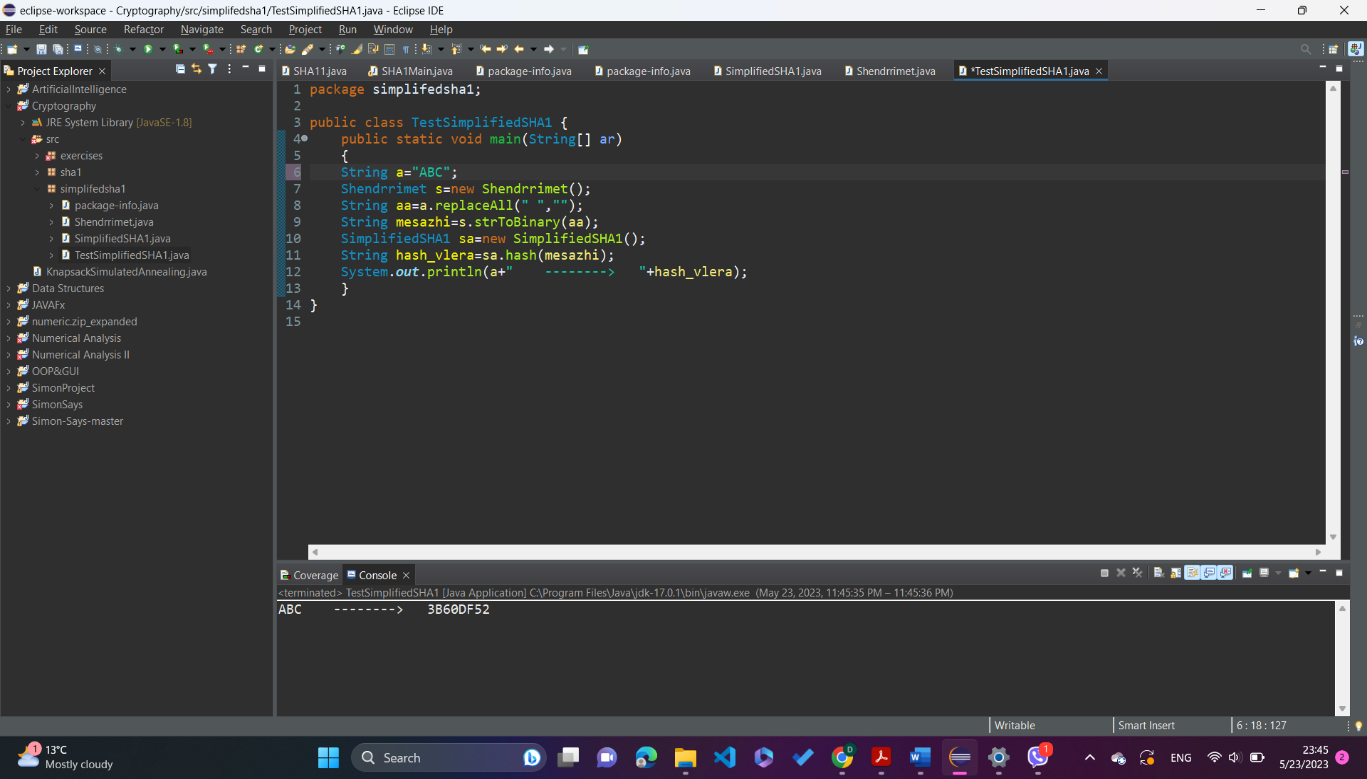
# **Implementimi i SHA-1**

Fillimisht kemi implementuar metodën hash, e cila i përdor pastaj metodat tjera që kthen vlerën hash.

* hash(String h): Kjo metodë merr një varg h si hyrje dhe llogarit vlerën hash SHA-1 të vargut. Ai inicializon një vektor fillestar (vektori\_inicializues) me një vlerë të paracaktuar. Metoda më pas plotëson vargun e hyrjes, e konverton atë në binar dhe kryen llogaritjen hash SHA-1 duke përdorur metodën H. Ai përsërit llogaritjen hash për çdo bllok të vargut të hyrjes dhe përditëson vektorin fillestar për çdo përsëritje. Së fundi, ai kthen vlerën hash që rezulton si një varg.
* paddedString(String mesazhi): Kjo metodë merr një varg mesazhi mesazhi si hyrje dhe e plotëson atë sipas rregullave të mbushjes SHA-1. Ai llogarit gjatësinë e mesazhit, përcakton sasinë e plotësimit të kërkuar dhe i shton mesazhit bitet e nevojshme. Më pas konverton vargun binar të mbushur në format heksadecimal duke përdorur metodat e klasës Shendrrimet dhe kthen vargun e mbushur.
* H(String a, String iv): Kjo metodë kryen funksionin e kompresimit SHA-1 në një bllok të dhënash të përfaqësuar nga vargu a duke përdorur vektorin fillestar iv. Ai i konverton vargjet hyrëse në binare duke përdorur metodat e klasës Shendrrimet. Ai inicializon katër konstante K1, K2, K3 dhe K4 me vlera të paracaktuara në formatin binar. Metoda më pas kryen katër raunde të kompresimit SHA-1 duke përdorur metodën etapa, duke përditësuar vlerat e ndërmjetme hash në çdo raund. Së fundi, ai konverton vlerën hash që rezulton nga formati binar në format heksadecimal dhe e kthen atë si një varg.
* etapa(int funksioni\_i\_n, boolean[] w\_1, boolean[] w\_2, boolean[] w\_3, boolean[] w\_4, char[] vlerat\_fillestare, boolean[] K): Kjo metodë paraqet një raund të vetëm të funksionit të kompresimit SHA-1. Merr si hyrje numrin e rrumbullakësuar (funksioni\_i\_n), katër grupe vlerash boolean (w\_1, w\_2, w\_3, w\_4), vlerat fillestare (vlerat\_fillestare) dhe një konstante (K). Ai kryen një sërë operacionesh në vlerat hyrëse, duke përfshirë operacione bit, ndërrime dhe operacione XOR, për të llogaritur daljen e rrumbullakët. Ai kthen vlerën hash që rezulton si një grup char.
* roundi(char[] a, boolean[] w, boolean[] K, int funksioni\_i\_n): Kjo metodë paraqet një raund të vetëm të funksionit të kompresimit SHA-1. Merr si hyrje numrin e rrumbullakët (funksioni\_i\_n), katër grupe vlerash boolean (w\_1, w\_2, w\_3, w\_4), vlerat fillestare (vlerat\_fillestare) dhe një konstante (K). Ai kryen një sërë operacionesh në vlerat hyrëse, duke përfshirë operacionet XOR, ndërrimet dhe operacionet në bit, për të llogaritur daljen e rrumbullakët. Ai kthen vlerën hash që rezulton si një grup char.
* kalkulimi\_Ws(boolean[] x\_i): Kjo metodë llogarit vlerat e variablave W (W0-W15) të përdorura në funksionin e kompresimit SHA-1. Ai merr një grup boolean x\_i si hyrje, i cili përfaqëson një bllok të dhënash. Ai e ndan bllokun e hyrjes në katër pjesë (x\_i\_j), cakton vlerat e W0-W3 drejtpërdrejt nga x\_i\_j dhe llogarit vlerat e W4-W15 duke kryer operacione XOR dhe zhvendosje në vlerat e mëparshme W. Vlerat e llogaritura W kthehen si një grup vargjesh boolean.
* Shendrrimet: Kjo klasë përmban metoda për konvertimin ndërmjet paraqitjeve binare dhe heksadecimal. Ai siguron metoda si bin2hex, bin2char, hex2bin dhe char2bin për të kthyer vlerat binare dhe heksadecimal në paraqitjet e tyre përkatëse.

Dhe kështu kemi arritur të implementojmë simplified SHA-1 të cilën kur e ekzekutojmë psh me stringun «Hello World» atëherë n’a shfaqet:

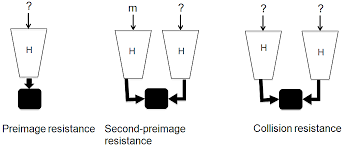


Ose nëse tentojmë t’a ekzekutojmë me stringun “ABC» atëherë n’a shfaqet:  


SHA-1 konsiderohet të jetë i dobët dhe i pasigurt për qëllime kriptografike për shkak të sulmeve të njohura të përplasjeve. Zbatimi i thjeshtuar mund të mos adresojë këto dobësi dhe mund të mos ofrojë të njëjtin nivel sigurie si algoritmi zyrtar SHA-1.

# **Preimage resistance**

Rezistenca e paraimazhit është një veti e dëshirueshme e një funksioni hash kriptografik. I referohet vështirësisë për të gjetur një imazh paraprak, i cili është një mesazh hyrës që prodhon një vlerë specifike hash. Me fjalë të tjera, duke pasur parasysh një dalje hash (H), rezistenca e imazhit paraprak siguron që është llogaritëse e pamundur të përcaktohet mesazhi fillestar i hyrjes (M) që rezultoi në atë vlerë hash.



Një funksion hash konsiderohet rezistent ndaj imazhit paraprak nëse, duke pasur parasysh një dalje hash H, është jashtëzakonisht e vështirë të gjesh ndonjë hyrje M të tillë që H(M) = H. Qëllimi është që të bëhet kompjuterikisht i pamundur kthimi i funksionit hash dhe gjetja e një funksioni specifik. mesazh hyrës që korrespondon me një vlerë të caktuar hash.

Rezistenca e paraimazhit është thelbësore për të siguruar integritetin dhe sigurinë e aplikacioneve të ndryshme kriptografike. Ai parandalon sulmuesit nga manipulimi i të dhënave ose gjenerimi i mesazheve mashtruese duke gjetur një hyrje tjetër që prodhon të njëjtën vlerë hash. Rezistenca e paraimazhit është veçanërisht e rëndësishme në hashimin e fjalëkalimit, nënshkrimet dixhitale dhe kodet e vërtetimit të mesazheve.

Një funksion hash me rezistencë paraimazhi ofron vetitë e mëposhtme:

Njëanshmëria: Është e lehtë të llogaritet vlera hash e një mesazhi hyrës, por nga ana llogaritëse e vështirë për të gjetur mesazhin hyrës që përputhet me një vlerë të caktuar hash.

Rezistencë e fortë ndaj përplasjes: Është e vështirë të gjesh dy mesazhe të ndryshme hyrëse që prodhojnë të njëjtën vlerë hash.

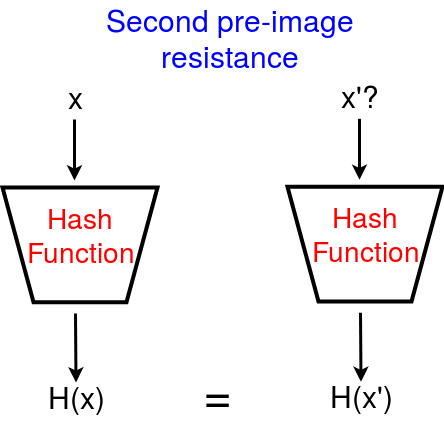
Rezistencë e dobët ndaj përplasjes (e njohur edhe si rezistenca e dytë e imazhit të dytë): Duke pasur parasysh një mesazh hyrës, është llogaritëse e pamundur të gjesh një mesazh tjetër hyrës që prodhon të njëjtën vlerë hash.

Për të siguruar rezistencën e paraimazhit, funksionet hash duhet të shfaqin veti të tilla si të qenit përcaktues, të prodhojnë dalje me gjatësi fikse dhe të shfaqin efektin e ortekëve (një ndryshim i vogël në hyrje duhet të rezultojë në një dalje hash dukshëm të ndryshme).

Është e rëndësishme të theksohet se rezistenca e paraimazhit nuk do të thotë se funksioni hash është i pamundur të kthehet. Përkundrazi, kjo do të thotë që gjetja e një mesazhi specifik hyrës nga një vlerë e caktuar hash duhet të jetë nga ana llogaritëse e pamundur, madje edhe me burime të rëndësishme llogaritëse. Niveli i rezistencës varet nga forca kriptografike dhe dizajni i funksionit hash.

# **Second preimage resistance**

Rezistenca e dytë e paraimazhit është një veti e dëshirueshme e funksioneve hash kriptografike. Ai i referohet aftësisë së një funksioni hash për t'i rezistuar gjetjes së një hyrje të dytë që prodhon të njëjtën vlerë hash si një hyrje fikse e dhënë. Me fjalë të tjera, duke pasur parasysh një mesazh hyrës arbitrar M dhe vlerën e tij hash H(M), rezistenca e dytë e paraimazhit siguron që është llogaritëse e pamundur të gjendet një mesazh tjetër hyrës M' (ku M ≠ M') që prodhon të njëjtën vlerë hash H( M') = H(M).



Rëndësia e rezistencës së dytë para imazhit qëndron në aftësinë e saj për të mbrojtur integritetin e të dhënave. Nëse një sulmues mund të gjejë një paraimazh të dytë, kjo do të thotë se ai mund të modifikojë përmbajtjen e një mesazhi pa ndryshuar vlerën e tij hash. Kjo mund të çojë në dobësi të ndryshme sigurie, të tilla si imitimi, korrupsioni i të dhënave ose aksesi i paautorizuar.

Një funksion hash me rezistencë të fortë të dytë të paraimazhit siguron siguri se është llogaritëse e vështirë të gjesh një hyrje tjetër që përputhet me një vlerë specifike hash. Prandaj, edhe nëse një sulmues e di vlerën hash të një mesazhi, ai do të duhet të kryejë një kërkim shterues ose të përdorë teknika të sofistikuara matematikore për të gjetur një hyrje të ndryshme që gjeneron të njëjtin hash.

Rezistenca e dytë e imazhit të dytë është një veti themelore e sigurisë për funksionet hash, veçanërisht në aplikacionet ku integriteti i të dhënave është kritik, të tilla si nënshkrimet dixhitale, hashimi i fjalëkalimit dhe kodet e vërtetimit të mesazheve (MAC). Ai siguron që një ndryshim i vogël në mesazhin hyrës të rezultojë në një vlerë të konsiderueshme hash të ndryshme, duke e bërë të vështirë për një sulmues të falsifikojë ose ngatërrojë të dhënat.

Si detyrë kemi që duke pasur të dhënë vlerën H(M) = 4BAFE69C, gjeni M’ ashtu që H(M’) = H(M)?

Prandaj në algoritmin e dhënë më herët do të tentojmë kryejmë një kërkim brute-force për një vlerë specifike hash në një zbatim të thjeshtuar SHA-1.

Si kemi implementuar metodën add?

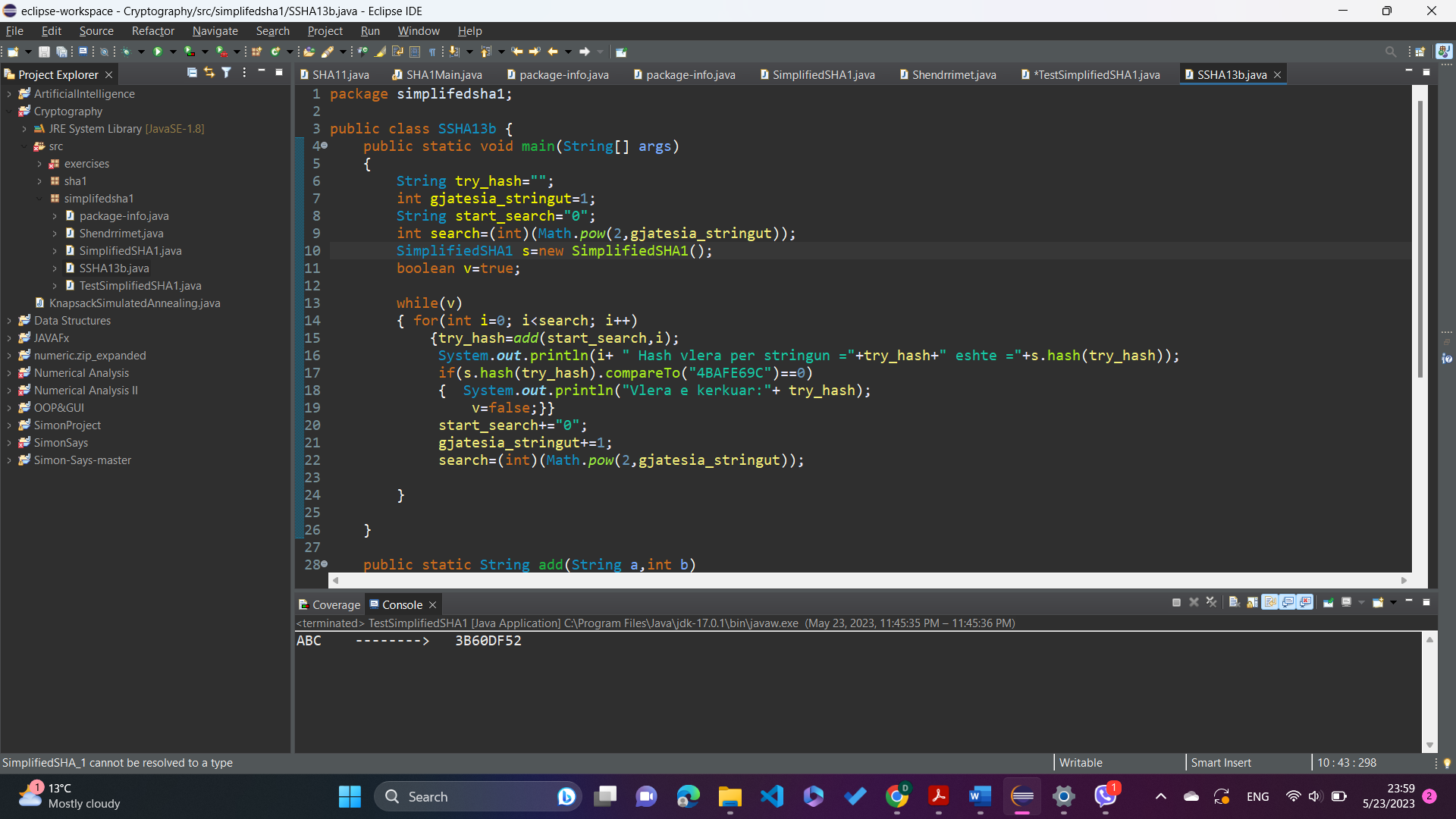
* Krijo një shembull të klasës Shendrrimet me emrin s.
* Konvertoni vargun e hyrjes a në një varg me numër të plotë duke përdorur metodën boolean\_to\_integer\_string të klasës Shendrrimet dhe caktojeni atë në ndryshoren aa.
* Analizoni vlerën e numrit të plotë nga aa dhe ruajeni atë në ndryshoren a2.
* Shtoni vlerën e plotë b në a2.
* Kthejeni a2 që rezulton përsëri në një varg dhe caktojeni atë në ndryshoren aa2.
* Shndërroni aa2 nga një varg dhjetor në një varg binar duke përdorur metodën dec\_to\_bin të klasës Shendrrimet dhe ruajeni atë në ndryshoren bb.
* Nëse gjatësia e bb nuk është e barabartë me gjatësinë e a, shtoni zerat kryesore në bb që të përputhen me gjatësinë e a.
* Ktheni vargun që rezulton bb.

Metoda main:

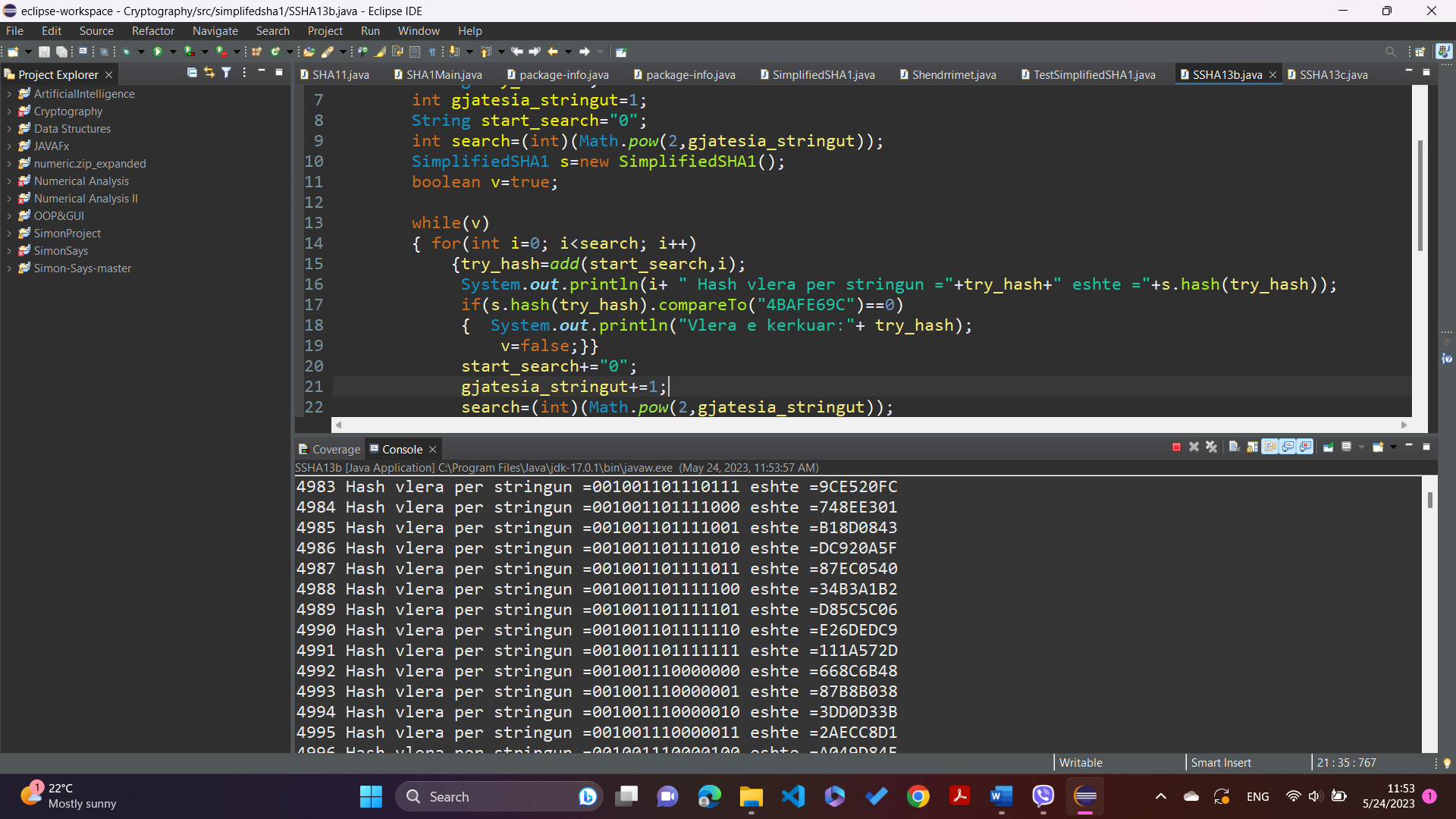
* Metoda kryesore është pika hyrëse për ekzekutimin e programit.
* Brenda metodës kryesore, inicializohen disa variabla:
* try\_hash është një varg bosh që përdoret për të ruajtur vlerat e ndërmjetme hash gjatë kërkimit.
* gjatesia\_stringut është një variabël numër i plotë i vendosur fillimisht në 1. Ai përfaqëson gjatësinë e vargut të përdorur për hash.
* start\_search është një varg i inicializuar me "0" si pikënisje për kërkimin.
* kërkimi është një ndryshore numër i plotë i llogaritur si 2 i ngritur në fuqinë e gjatesia\_stringut. Ai përcakton numrin e përsëritjeve për kërkimin.
* s është një shembull i klasës Simplified\_SHA\_1, që përfaqëson zbatimin e thjeshtuar SHA-1.
* v është një variabël boolean i vendosur fillimisht në true. Ai shërben si një flamur për të kontrolluar ciklin e kërkimit.
* Programi hyn në një cikli while me kushtin v (true).
* Brenda ciklit while, një cikli for përdoret për të përsëritur i nga 0 në kërkim - 1.
* Brenda ciklit for, thirret metoda add me argumentet start\_search dhe i. Metoda add kryen disa transformime në vargun hyrës dhe kthen vargun që rezulton.
* Vlera e ndërmjetme hash try\_hash i caktohet vlera e kthyer nga metoda add.
* Më pas programi printon numrin e përsëritjes (i), vargun aktual (try\_hash) dhe vlerën e tij përkatëse hash duke përdorur metodën hash të klasës Simplified\_SHA\_1.
* Nëse vlera e llogaritur e hash-it përputhet me vargun "4BAFE69C", programi printon mesazhin "Vlera e kerkuar:" e ndjekur nga vlera e try\_hash. Ai gjithashtu vendos v në false, duke përfunduar ciklin while.
* Pas përfundimit të ciklit for, programi përditëson vargun start\_search duke shtuar një karakter "0".
* Ndryshoret gjatesia\_stringut dhe search përditësohen për të pasqyruar gjatësinë dhe numrin e shtuar të përsëritjeve për përsëritjen e radhës të ciklit while.
* Cikli while vazhdon derisa të gjendet vlera e dëshiruar hash dhe flamuri v vendoset në false.

Në përgjithësi, ky kod fillon me një varg me gjatësi 1, rrit në mënyrë graduale gjatësinë e vargut dhe gjeneron të gjitha vargjet e mundshme të asaj gjatësi. Për çdo varg, ai llogarit vlerën e tij hash duke përdorur algoritmin e thjeshtuar SHA-1 dhe kontrollon nëse përputhet me vlerën e synuar hash. Pasi të gjendet një përputhje, programi përfundon dhe nxjerr vargun përkatës.

Më poshtë është dhënë implementimi në JAVA:

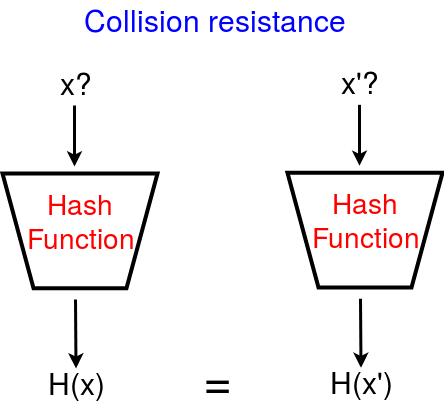


Dhe pastaj kur e ekzektojmë n’a shfaqet:



# **Collision resistance**

Rezistenca ndaj përplasjes është një veti e funksioneve hash që siguron se është llogaritëse e vështirë për të gjetur dy hyrje (mesazhe) të ndryshme që prodhojnë të njëjtën vlerë hash. Me fjalë të tjera, një funksion hash konsiderohet rezistent ndaj përplasjeve nëse ka shumë pak gjasa të gjejë dy hyrje të dallueshme M1 dhe M2 të tilla që hash(M1) = hash(M2).



Rezistenca ndaj përplasjes është një veti kritike për funksionet hash sepse i pengon sulmuesit të gjejnë lehtësisht dy mesazhe të ndryshme me të njëjtën vlerë hash. Nëse rrezikohet rezistenca ndaj përplasjes, një sulmues mund të krijojë dy mesazhe të ndryshme me vlera identike hash, të cilat mund të çojnë në dobësi të sigurisë.

Koncepti i rezistencës ndaj përplasjes është i lidhur ngushtë me paradoksin e ditëlindjes, i cili thotë se në një grup prej n elementësh, probabiliteti për të gjetur një përplasje (dy elementë me të njëjtën vlerë) bëhet çuditërisht i lartë kur n arrin rrënjën katrore të numrit total. të vlerave të mundshme. Në kontekstin e funksioneve hash, kjo do të thotë se me rritjen e numrit të hyrjeve të mundshme, rritet edhe probabiliteti për të gjetur një përplasje.

Për të vlerësuar rezistencën ndaj përplasjes së një funksioni hash, kriptografët kryejnë analiza rigoroze matematikore dhe kryejnë testime të gjera për të kërkuar për çdo dobësi ose model që mund të çojë në përplasje. Nëse një funksion hash i përballon këto teste dhe nuk gjenden përplasje praktike, ai konsiderohet rezistent ndaj përplasjeve.

Është e rëndësishme të theksohet se rezistenca ndaj përplasjes nuk do të thotë se përplasjet janë të pamundura. Për shkak të parimit të vrimës së pëllumbit, ku ka më shumë hyrje të mundshme sesa vlera të veçanta hash, përplasjet janë të pashmangshme. Megjithatë, një funksion i mirë hash minimizon gjasat për të gjetur përplasje, duke e bërë të pamundur nga pikëpamja llogaritëse gjetjen e tyre të qëllimshme.

Në praktikë, kur përdoret një funksion hash rezistent ndaj përplasjeve, ai siguron një nivel të lartë besimi në integritetin dhe sigurinë e aplikacioneve të tilla si nënshkrimet dixhitale, ruajtja e fjalëkalimeve, kontrollet e integritetit të të dhënave dhe më shumë.

Për detyrën e fundit n’a kërkohet të zgjedhim një mesazh të gjatësisë së çfarëdoshme pastaj llogarisim hash vlerën më anë të Simplified SHA-1, pra e llogaritni H(M). Pastaj të gjindet M’ ashtu që M ≠ M dhe H(M’) = H(M)?

Duhet të implementojmë një sulm me forcë brutale për të gjetur një paraimazh të një vlere të caktuar hash. Le të shtjellojmë kodin hap pas hapi:

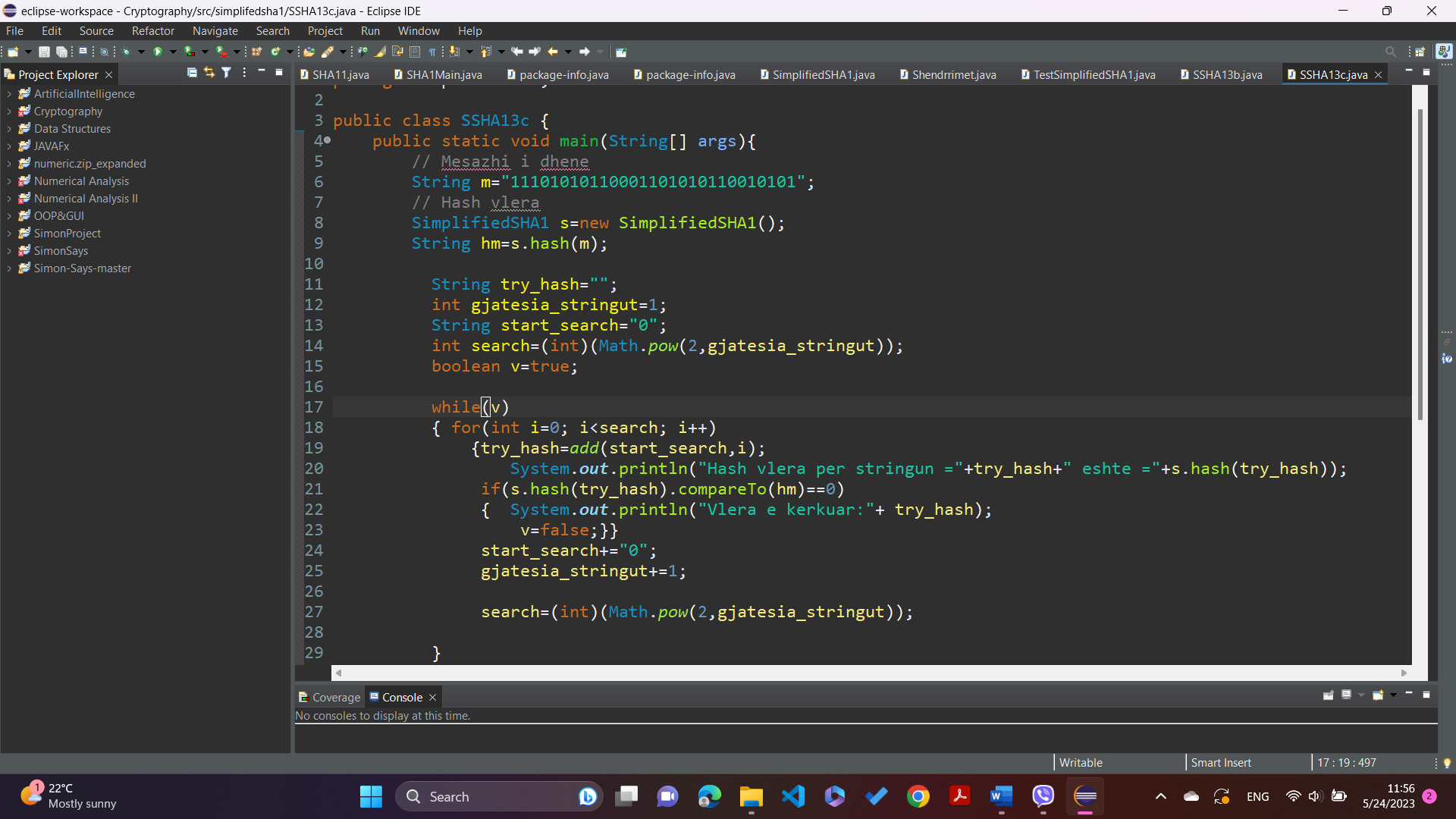
Kodi fillon me përcaktimin e një mesazhi m si një varg binar.

Një shembull i klasës SimplifiedSHA1 krijohet për të llogaritur vlerën hash të mesazhit

Inicializohen variablat try\_hash, gjatesia\_stringut, start\_search dhe search. Këto variabla përdoren në ciklin e sulmit me forcë brute për të gjeneruar vargje të ndryshme dhe për të llogaritur vlerat e tyre hash.

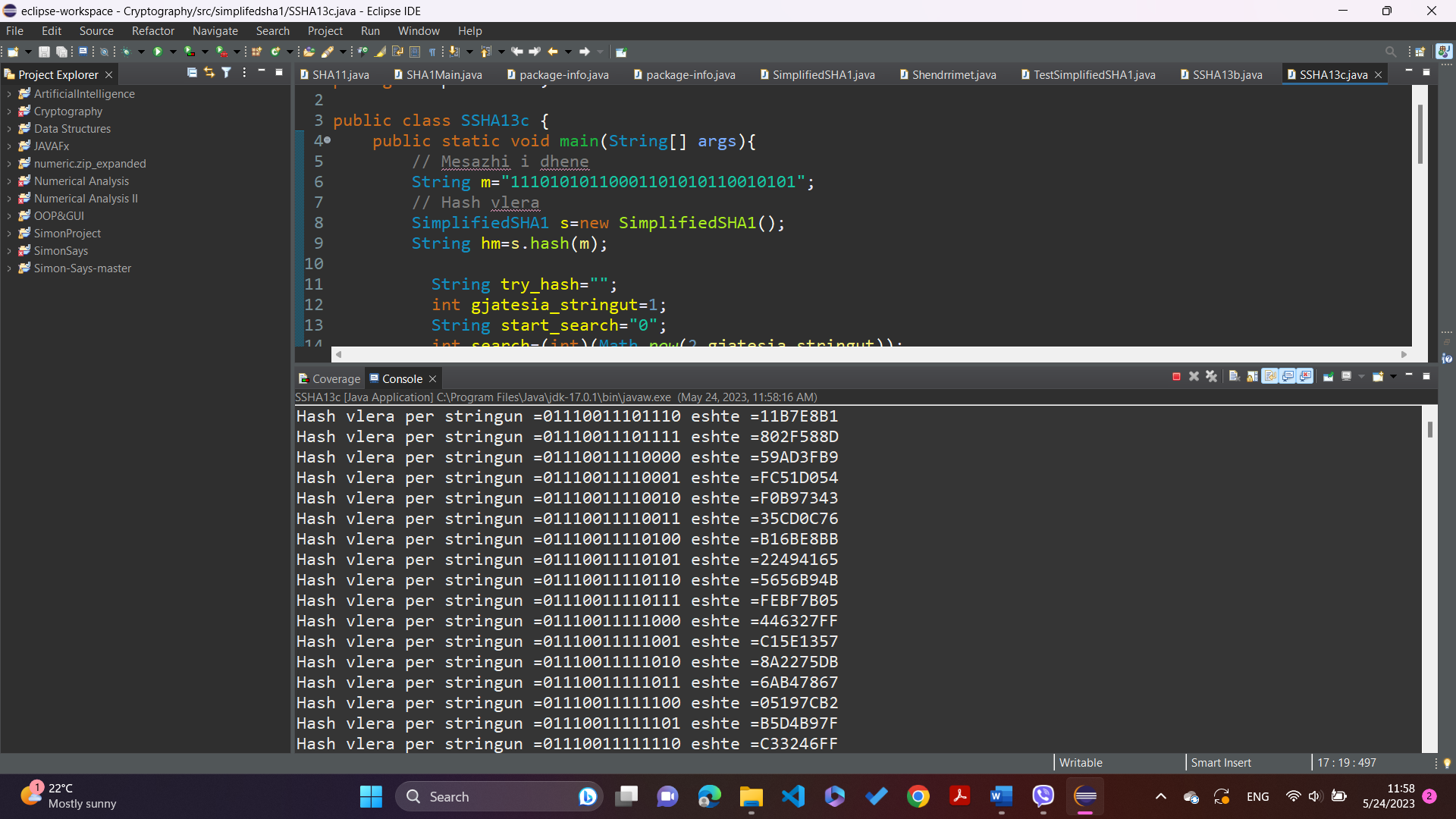
Cikli while kryen sulmin e forcës brutale duke gjeneruar vargje të ndryshme, duke i hasur ato dhe duke krahasuar vlerën e llogaritur të hash-it me vlerën e synuar hash (hm').

Kështu e kemi implementuar në JAVA:



Si përmbledhje, kodi kryen një sulm me forcë brutale duke gjeneruar sistematikisht vargje binare, duke llogaritur vlerat e tyre hash duke përdorur algoritmin SimplifiedSHA1 dhe duke krahasuar vlerat e llogaritura të hash-it me një vlerë hash të synuar. Nëse gjendet një përputhje, shtypet vargu përkatës i paraimazhit.

Rezultati:



# **Përfundimi**

Pas zbatimit të algoritmit të thjeshtuar SHA-1 dhe kryerjes së një sulmi me forcë brutale, mund të nxjerrim përfundimet e mëposhtme:

SHA-1 i thjeshtuar: Zbatimi i algoritmit të thjeshtuar SHA-1 ofron një version të thjeshtuar të funksionit origjinal hash SHA-1. Ndërsa i mungon kompleksiteti i plotë dhe veçoritë e sigurisë së algoritmit origjinal, ai shërben si një mjet demonstrimi dhe mësimi për të kuptuar parimet bazë të funksioneve hash.

Sulmi Brute Force për Gjetjen e Preimazhit: Sulmi i forcës brutale i zbatuar me sukses gjen një paraimazh M' të tillë që H(M') = H(M), duke pasur parasysh një vlerë specifike hash H(M). Sulmi gjeneron sistematikisht vargje të ndryshme binare dhe llogarit vlerat e tyre hash duke përdorur SHA-1 të thjeshtuar. Duke krahasuar vlerat e llogaritura të hash-it me vlerën e synuar të hash-it, sulmi identifikon një varg të përputhshëm të paraimazhit. Kjo nënvizon cenueshmërinë e funksioneve hash ndaj sulmeve me forcë brutale dhe thekson rëndësinë e rezistencës ndaj përplasjes.

Rezistenca ndaj përplasjes: Sulmi i forcës brutale ndaj rezistencës ndaj përplasjes synon të gjejë dy mesazhe hyrëse të ndryshme M1 dhe M2 që prodhojnë të njëjtën vlerë hash H(M1) = H(M2). Megjithatë, pjesa e dhënë e kodit nuk kryen drejtpërdrejt një sulm përplasjeje. Duket se kryen një kërkim të ngjashëm si sulmi i paraimazhit, por vlera e hash-it të synuar është hash-i i një mesazhi të dhënë m. Rezistenca ndaj përplasjes është një veti e dëshirueshme e funksioneve hash, pasi siguron që gjetja e dy hyrjeve të ndryshme me të njëjtën vlerë hash është llogaritëse e pamundur. Ndërsa kodi nuk demonstron një sulm të suksesshëm përplasjeje, ai tregon konceptin e kërkimit të përplasjeve duke përdorur metodat e forcës brutale.

Në përmbledhje, zbatimi i SHA-1 të thjeshtuar dhe sulmet e kryera me forcë brutale nxjerrin në pah dobësitë dhe kufizimet e funksioneve hash. Ata theksojnë rëndësinë e përdorimit të funksioneve hash të sigurta dhe të testuara mirë, të tilla si SHA-1 e plotë ose alternativa më moderne si SHA-256 ose SHA-3, për të siguruar qëndrueshmëri dhe rezistencë ndaj sulmeve.