**Hash Message Authentication Code (HMAC)**

Rreze Vrapçani

FSHMN – Shkencë Kompjuterike

Universiteti i Prshtinës

Prishtinë, Kosovë, 10000

3 rreze.vrapcani@student.uni-pr.edu

Rubina Berisha

FSHMN – Shkencë Kompjuterike

Universiteti i Prshtinës

Pejë, Kosovë, 30000

2 rubina.berisha@student.uni-pr.edu

Erë Dedinca

FSHMN – Shkencë Kompjuterike

Universiteti i Prshtinës

Prishtinë, Kosovë, 10000

1 ere.dedinca@student.uni-pr.edu

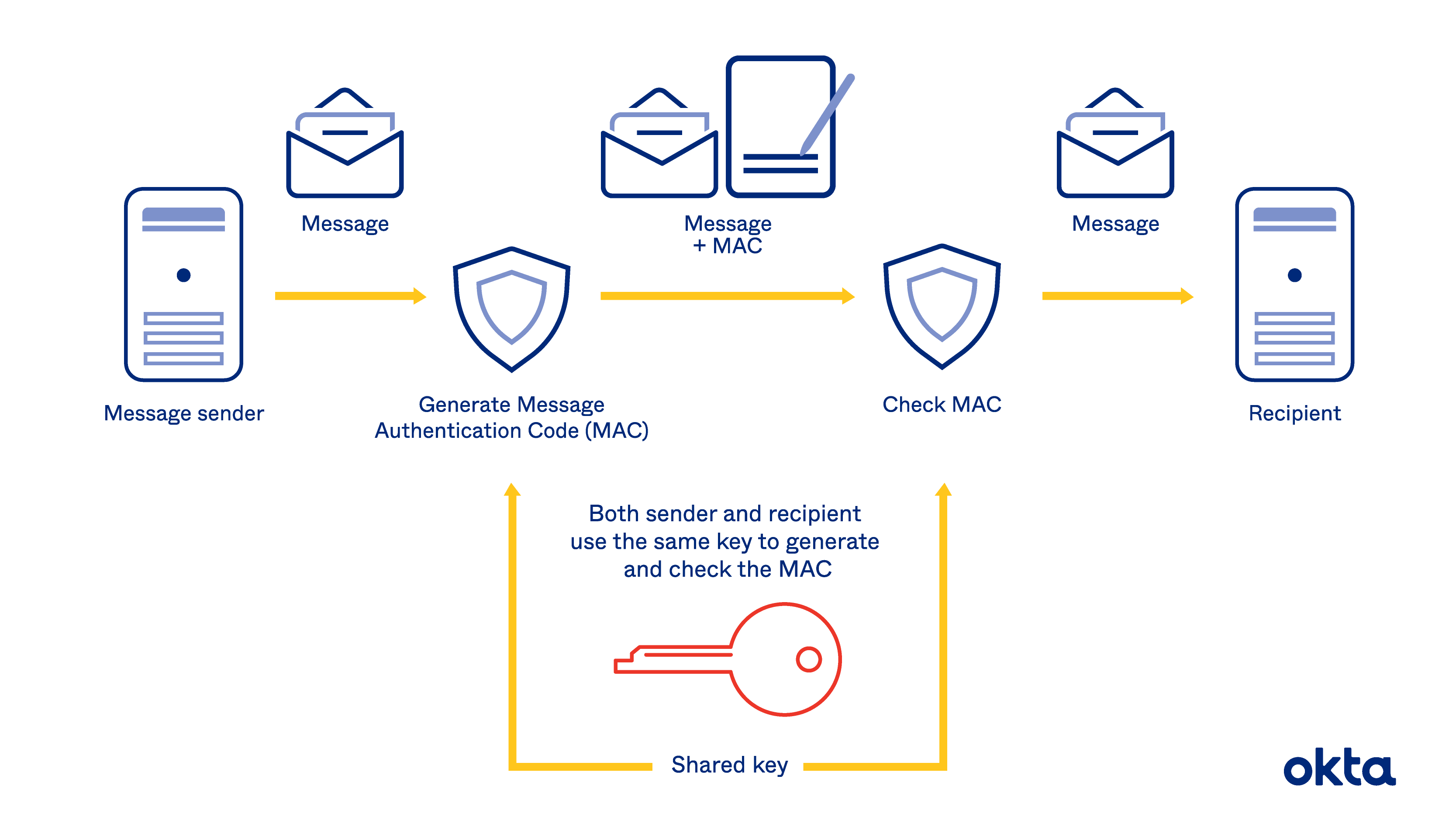


Figure 1. HMAC Algoritmi

**Abstrakt: Në këtë punim do të përfshihendisa detaje në lidhje me Hash Message Authentication Code ( HMAC) , do tw tregohet pwr dallimet nw mes tw HMAC\_Simplified\_SHA-1 dhe versioneve te tjera tw saj, si dhe do tw tregohet pwr kodin e implementuar me anw tw logjikws sw HMAC.**

**Fjalwt kyçe: HMAC, algoritëm, siguria, të dhënat, mesazhi, çelwsi.**

1. **ALGORITMI HMAC**

Hash Message Authentication Code (HMAC) është një algoritëm për kriptografi që përdoret për të siguruar autentifikimin dhe integritetin e të dhënave. Përveç algoritmit të hash-it, HMAC përfshin edhe një çelës sekret për të siguruar mbrojtjen e të dhënave.

HMAC ndjek një proces të thjeshtë për të llogaritur një kod autentifikimi për një mesazh të dhënë. Ky kod autentifikimi njihet si HMAC dhe gjenerohet duke përdorur një algoritem të hash-it dhe një çelës sekret. Çelësi sekret përdoret për të bërë të pamundur llogaritjen e kodit autentifikues pa të dhënat e duhura.

Procesi i llogaritjes së HMAC përfshin hapat e mëposhtëm:

* Përdorimi i një algoritmi të hash-it (si SHA-1, SHA-256, etj.).
* Përdorimi i një çelësi sekret për të siguruar HMAC-in.
* Përpunimi i mesazhit origjinal duke shtuar padding (mbushje) nëse është e nevojshme.
* Llogaritja e hash-it të paraprakët të mesazhit të zgjeruar duke përdorur algoritmin e hash-it të zgjedhur.
* Përpunimi i hash-it të paraprakët me çelësin sekret për të prodhuar rezultatin përfundimtar, që është HMAC-i.

HMAC është i përdorshëm në shumë skenarë të sigurisë kompjuterike, si për integritetin e të dhënave, autentifikimin dhe verifikimin e fjalëkalimeve, si dhe në protokollet e sigurisë si TLS (Transport Layer Security) dhe IPSec (Internet Protocol Security).

Është e rëndësishme të përdorni një algoritem të hash-it të sigurtë dhe një çelës sekret të fortë për të siguruar efektivisht autentifikimin dhe integritetin e të dhënave me HMAC.

1. **PERDORIMI I HMAC**

Hash Message Authentication Code (HMAC) përdoret kryesisht në skenarë ku siguria dhe autenticiteti i të dhënave janë të rëndësishme. Disa raste të përdorimit të HMAC janë:

1. ***Autentifikimi i përdoruesve***: HMAC mund të përdoret për të verifikuar identitetin e një përdoruesi në një sistem të sigurtë. Për shembull, në protokollet e autentifikimit në internet (si OAuth), HMAC përdoret për të verifikuar autenticitetin e kredencialeve të përdoruesit në mënyrë të sigurtë.
2. ***Siguria e transmetimit të të dhënave***: Në komunikimin e sigurtë midis dy palëve, HMAC përdoret për të verifikuar integritetin e të dhënave të transmetuara dhe për të identifikuar ndryshime të mundshme ose manipulime të të dhënave gjatë transmetimit
3. ***Autentifikimi i mesazheve***: HMAC mund të përdoret për të verifikuar autenticitetin e mesazheve të dërguara midis palëve të ndryshme. Kjo është e dobishme në sisteme të tilla si protokollet e dërgimit të mesazheve (si SMTP, IMAP) për të siguruar që mesazhet nuk janë ndryshuar dhe janë dërguar nga burimi i duhur.
4. ***Kontrolli i integritetit të skedarëve***: Në rastet kur është e nevojshme të verifikohet integriteti i skedarëve, HMAC mund të përdoret për të krijuar një kod autentifikimi për çdo skedar dhe për të verifikuar nëse skedari është ndryshuar ose manipuluar nga momenti i krijimit të kodit autentifikimi.
5. ***Verifikimi i integritetit të bazave të të dhënave***: Në sistemet e bazave të të dhënave, HMAC mund të përdoret për të verifikuar integritetin e të dhënave të ruajtura dhe për të zbuluar ndryshime ose manipulime të paligjshme të të dhënave në bazën e të dhënave.

Këto janë vetëm disa raste të përdorimit të HMAC, por algoritmi është shumë fleksibël dhe mund të aplikohet në shumë skenarë të tjera ku siguria dhe autenticiteti janë prioritet.

1. **Dallimi mes HMAC-SHA-1, HMAC-SHA-256 dhe HMAC-SHA-512**

Dallimi mes HMAC-SHA-1, HMAC-SHA-256 dhe HMAC-SHA-512 është në algoritmet e hashimit të përdorura për të llogaritur HMAC-in dhe gjatësia e rezultatit të HMAC-it.

* **HMAC-SHA-1**: Kjo përdor algoritmin SHA-1 për hashim dhe prodhon një rezultat me gjatësi 160 bitë (20 bajta). Megjithëse HMAC-SHA-1 ka qenë shumë herët i përdorur, tani konsiderohet i dobishëm për shkak të sigurisë së dobët të algoritmit SHA-1, i cili është prekur nga shumë lëndime në siguri.
* **HMAC-SHA-256**: Kjo përdor algoritmin SHA-256 për hashim dhe prodhon një rezultat me gjatësi 256 bitë (32 bajta). Algoritmi SHA-256 është më i sigurt se SHA-1 dhe përdoret gjerësisht në aplikacione të ndryshme për sigurinë e të dhënave.
* **HMAC-SHA-51**2: Kjo përdor algoritmin SHA-512 për hashim dhe prodhon një rezultat me gjatësi 512 bitë (64 bajta). SHA-512 është versioni më i zgjeruar dhe më i sigurt i familjes SHA, i cili siguron një nivel të lartë të sigurisë dhe rezistenca kundër sulmeve të njohura.

Për zgjedhjen midis HMAC-SHA-1, HMAC-SHA-256 dhe HMAC-SHA-512, rekomandohet të përdorni versionet më të fortë të hashimeve (në rastin më të mirë HMAC-SHA-256 ose HMAC-SHA-512) për shkak të sigurisë së tyre më të lartë. Përdorimi i algoritmeve të fortë të hashimit ndihmon në përmirësimin e rezistencës së aplikacioneve kundër sulmeve të mundshme të hashit.

A picture containing text, screenshot, diagram, design

Description automatically generated

1. **Dokumentimi**

Ky program Java gjeneron një kod të autentifikimit të mesazheve të bazuar në Hash (HMAC) për një mesazh të caktuar ("Përshëndetje, botë!") duke përdorur një çelës sekret ("secretKey"). HMAC më pas shtypet në format heksadecimal. Metoda kryesore: Fillimisht thërret llogaritjen HMAC për të llogaritur HMAC të mesazhit, më pas thërret bytesToHex për të kthyer HMAC në një varg karakteresh heksadecimal. Më në fund, printon HMAC-në që rezulton. Qëllimi i krijimit të një HMAC është të sigurohet integriteti dhe autenticiteti i të dhënave. Kur marrësi i një mesazhi ka të njëjtin çelës sekret të përbashkët, ata mund të llogarisin HMAC në mesazhin e marrë dhe ta krahasojnë atë me HMAC që kanë marrë. Nëse përputhen, ai konfirmon që mesazhi nuk është manipuluar dhe se ai me të vërtetë ka ardhur nga dërguesi i pritur.

|  |
| --- |
| import javax.crypto.Mac; import javax.crypto.spec.SecretKeySpec; import java.security.InvalidKeyException; import java.security.NoSuchAlgorithmException;  public class Punimi3 {  public static void main(String[] args) {         String message = "Hello, world!";         String key = "secretKey";   try{  byte[] hmac = calculateHMAC(message, key);  String hmacHex = bytesToHex(hmac);             System.out.println("HMAC: " + hmacHex);         } catch (NoSuchAlgorithmException | InvalidKeyException e) {             e.printStackTrace(); |
| *Ky kod Java gjeneron një Kod të Autentifikimit të Mesazhit të bazuar në Hash (HMAC) për një mesazh të caktuar duke përdorur një çelës sekret. Këtu është një shpjegim i shkurtër:*  *Përcaktohen një mesazh ("Përshëndetje, botë!") dhe një çelës sekret ("secretKey").*  *Metoda e llogaritjes HMAC është një funksion i cili gjeneron HMAC për mesazhin e dhënë duke përdorur çelësin sekret të specifikuar dhe algoritmin HMAC-SHA1. HMAC-SHA1 është një lloj specifik i HMAC që përdor funksionin hash SHA-1.*  *HMAC më pas konvertohet në një varg heksadecimal duke përdorur metodën bytesToHex – funksion që merr një grup bajtësh (daljen nga llogaritja HMAC) dhe e konverton atë në një varg heksadecimal. Çdo bajt në grup bëhet dy karaktere heksadecimal në varg.*  *Nëse ndodh ndonjë gabim gjatë këtyre proceseve (si p.sh. nëse algoritmi kriptografik nuk ekziston ose çelësi është i pavlefshëm), gjurmimi i raftes së gabimit do të printohet në tastierë.*  *Ky kod zakonisht përdoret për të siguruar integritetin dhe vërtetësinë e të dhënave.*  *Megjithatë, është e rëndësishme të theksohet se HMAC mund të jetë po aq i sigurt sa çelësi sekret i përbashkët.* |

|  |
| --- |
| private static byte[] calculateHMAC(String message, String key)             throws NoSuchAlgorithmException, InvalidKeyException {         Mac hmacSha1 = Mac.getInstance("HmacSHA1");         SecretKeySpec secretKey = new SecretKeySpec(key.getBytes(), "HmacSHA1");         hmacSha1.init(secretKey);         return hmacSha1.doFinal(message.getBytes());     } |
| Ky funksion, llogarit HMAC, llogarit HMAC (Kodi i Autentifikimit të Mesazhit të bazuar në Hash) të një mesazhi të dhënë duke përdorur çelësin e dhënë. Këtu është një shpjegim hap pas hapi:  Mac.getInstance("HmacSHA1"): Kjo linjë merr një shembull të klasës Mac (Message Authentication Code) që përdor algoritmin HMAC-SHA1. HMAC-SHA1 është një lloj kodi i vërtetimit të mesazhit që përfshin një funksion hash kriptografik (SHA-1) dhe një çelës sekret kriptografik.  SecretKeySpec secretKey = i ri SecretKeySpec(key.getBytes(), "HmacSHA1"): Kjo linjë gjeneron një specifikim të çelësit sekret për algoritmin HMAC-SHA1 nga çelësi i dhënë. Metoda key.getBytes() përdoret për të kthyer çelësin nga lloji String në llojin bajt[].  hmacSha1.init(secretKey): Kjo linjë inicializon instancën Mac me specifikimin e çelësit sekret. Pas kësaj, shembulli Mac është gati për të llogaritur HMAC.  Return - kthe hmacSha1.doFinal(message.getBytes()): Kjo linjë llogarit HMAC në mesazhin e dhënë. Metoda doFinal përpunon të dhënat e ofruara (d.m.th., mesazhin), llogarit HMAC dhe më pas rivendos instancën Mac. HMAC kthehet si një grup bajt.  Në përmbledhje, ky funksion llogarit dhe kthen HMAC-në e një mesazhi të caktuar duke përdorur algoritmin HMAC-SHA1 dhe një çelës sekret të dhënë. |

|  |
| --- |
| private static String bytesToHex(byte[] bytes) {         StringBuilder hexString = new StringBuilder();         for (byte b : bytes) {             String hex = Integer.toHexString(0xFF & b);             if (hex.length() == 1) {                 hexString.append('0');             }             hexString.append(hex);         }         return hexString.toString();     } |
| *Kjo metodë, bytesToHex, përdoret për të kthyer një grup bajtësh në vargun e tij heksadecimal ekuivalent. Ja se si funksionon:*  *Ai krijon një StringBuilder të quajtur hexString për të ruajtur vargun heksadecimal që rezulton.*  *Më pas kalon mbi çdo bajt në grupin e bajtit të hyrjes.*  *Për çdo bajt, b, ai përdor metodën Integer.toHexString(0xFF & b) për të kthyer bajtin në një numër të plotë dhe më pas në një varg heksadecimal. Pjesa 0xFF & b është një operacion AND në bit që siguron që bajt të trajtohet si një vlerë pozitive (pasi bajtet në Java janë të nënshkruara dhe ne duam të shmangim vlerat negative këtu).*  *Nëse gjatësia e vargut heksadecimal që rezulton është 1 (që do të thotë se bajt ishte më i vogël se 16, ose 0x10 në heksadecimal), ai i jep një '0' vargut. Kjo bëhet për të siguruar që çdo bajt të përfaqësohet saktësisht me dy shifra heksadecimal në vargun përfundimtar.*  *Vargu heksadecimal për bajtin më pas i shtohet hexString StringBuilder.*  *Pasi të jenë përpunuar të gjitha bajtet, ai konverton StringBuilder në një varg dhe e kthen atë.*  *Në përmbledhje, kjo metodë konverton një grup bajt në vargun e tij heksadecimal ekuivalent, ku çdo bajt përfaqësohet saktësisht nga dy shifra heksadecimal*. |

1. **Përfundimi**

Projekti HMAC (Hash-based Message Authentication Code) në Java që keni ndërmarrë shërben si një hyrje solide për kriptografinë dhe transmetimin e sigurt të të dhënave. Ai demonstron se si të gjenerohen HMAC për integritetin dhe autenticitetin e të dhënave, aspekte thelbësore të sigurisë së informacionit.

Këtu janë pikat kryesore të këtij projekti: ***Kuptimi i HMAC***: Projekti ofroi një kuptim praktik të HMAC-ve, të cilat përdoren gjerësisht në kriptografi. HMAC-të kombinojnë përdorimin e funksioneve hash kriptografike me një çelës sekret, duke ofruar një mënyrë për të verifikuar integritetin dhe vërtetësinë e një mesazhi. ***Arkitektura e Kriptografisë Java:*** Ky projekt ndihmoi për të mësuar rreth Arkitekturës së Kriptografisë Java (JCA) dhe aplikimit të saj. JCA është një kornizë për aksesin dhe zhvillimin e funksioneve kriptografike në Java. ***Përdorimi i algoritmit HMAC-SHA1:*** Algoritmi HMAC-SHA1 është përdorur në këtë projekt. Ai shërbeu si një hyrje në algoritmet e ndryshme të disponueshme HMAC. ***Përfaqësimi i të dhënave:*** Projekti demonstroi se si të konvertohen të dhënat binare (dalja HMAC) në një format heksadecimal të lexueshëm. ***Trajtimi i gabimeve:*** Projekti demonstroi përdorimin e trajtimit të përjashtimeve në aplikacionet kriptografike, gjë që është thelbësore për shkak të mundësisë që shumë gjëra të shkojnë keq, si një çelës i pasaktë ose një algoritëm i pambështetur.

Si përfundim, projekti HMAC është një hyrje e vlefshme në aplikimin e teknikave kriptografike në Java. Ai hap rrugën për një eksplorim më të thellë të kriptografisë, duke përfshirë kuptimin e algoritmeve të ndryshme kriptografike, nënshkrimeve dixhitale, teknikave të kriptimit/deshifrimit dhe protokolleve më komplekse të vërtetimit.