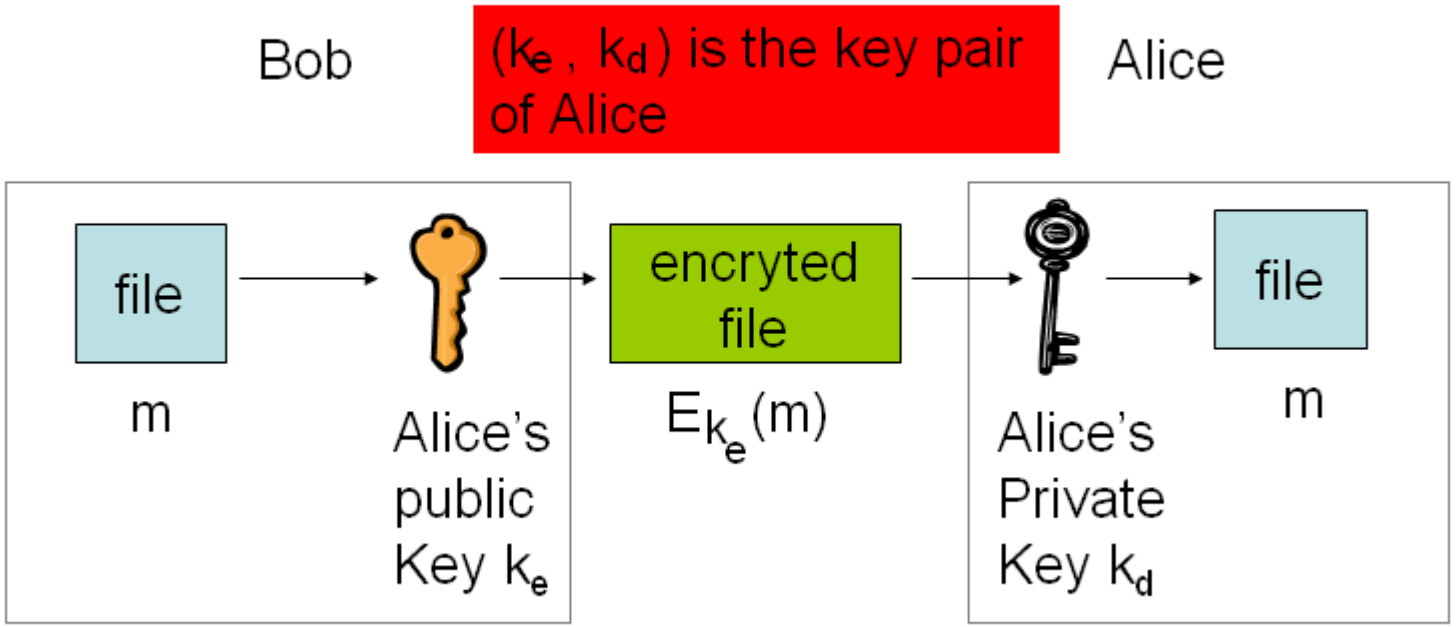
Информациска безбедност

Предавање 3:
Криптографија – втор дел

Проф. д-р Весна Димитрова

Криптографија со јавен клуч

Основни поими



Криптографија со јавен клуч

- Се користат два клуча
  - еден за шифрирање (јавен) и
  - сосема друг, различен, за дешифрирање (таен).

- Се темели на т.н. *trap door one-way functions*:
  - функции кои лесно се пресметуваат во една насока,
  - неверојатно тешко во обратната насока,
  - Се користи „*trap door*“ за наоѓање на клучот
  - Пример:
    За дадени *P* и *Q*, лесно е да се пресмета *N=P*Q*, но за дадено *N*, тешко е да се најдат *P* и *Q.*

Криптографија со јавен клуч

- Шифрирање
  - За да изврши шифрирање со јавен клуч *Bob* мора да има пар клучеви кој се состои од јавен и приватен клуч.
  - Било кој може да го користи јавниот клуч на *Bob* за да шифира порака до него.
  - Само *Bob* може да ја дешифрира, бидејќи само тој го има приватниот клуч.

Криптографија со јавен клуч

- Дигитални потписи
  - *Bob* може дигитално да потпише порака со тоа што ќе ја „шифрира“ со неговиот приватен клуч.
  - Секој ќе може да ја „дешифрира“ користејќи го јавниот клуч на *Bob.*
  - Сите може да видат дека пораката е испратена т.е. потпишана токму од *Bob*.
  - На овој начин *Bob* својот физички потпис го заменува со дигитален.

RSA

- Пронајден од *Rivest, Shamir* и *Adleman* (*MIT*)
  - Идеја на *Cliff Cocks* (*GCHQ*) неколку год. порано
- Нека p и q се два големи прости броја
- Пресметуваме $N = p*q$ - **модул**
- Бираме број e взаемно прост со $(p–1)(q–1)$
- Пресметуваме d т.ш. $e*d = 1 \bmod (p–1)(q–1)$
- **Јавен клуч:** (N,e)
- **Таен клуч:** d

RSA

- За шифрирање на M пресметуваме
  - $C = M^e \bmod N$
- За дешифрирање на C пресметуваме
  - $M = C^d \bmod N$
- e и N се познати
- Ако напаѓачот го факторизира N, користејќи го e лесно ќе го најде d, бидејќи $e*d = 1 \bmod (p–1)(q–1)$
- Со факторизација на N се разбива RSA.

RSA – пример за клуч

- Нека $p = 11$, $q = 3$
- Тогаш $N = p*q = 33$ и $(p–1)(q–1) = 20$
- Бираме $e = 3$ (взаемно прост со 20)
- Пресметуваме d така што $e*d = 1 \bmod 20$, и добиваме дека $d = 7$
- **Јавен клуч:** $(N, e) = (33, 3)$
- **Таен клуч:** $d = 7$

RSA – пример за шифрирање/дешифрирање

- **Јавен клуч:** (N, e) = (33, 3)
- **Таен клуч:** d = 7
- Нека оригиналната порака M = 8
- Шифрираната порака C се пресметува како:

  $C = M^e \bmod N = 8^3 = 512 = 17 \bmod 33$
- За да ја дешифрираме C и да ја добиеме M пресметуваме

  $M = C^d \bmod N = 17^7 = 8 \bmod 33.$

RSA – генерирање на клуч

- *P* и *Q* не смеат да бидат премногу блиски еден до друг бидејќи, методот за факторизација на Fermat може да биде успешен.
- Доколку *P*-1 или *Q*-1 имаат само мали прости фактори, *N* би можел брзо да се факторизира со помош на *P*-1 алгоритмот на *Polard*, па и ваквите вредности на *P* и *Q* не треба да се земаат во предвид.
- …

Diffie-Hellman

- Пронајден од *Diffie* и *Hellman* (*Stanford*)
  - Идејата на *Williamson* (*GCHQ*)
- Алгоритам за размена на клуч (“*key exchange*” *algorithm*)
  - се користи за воспоставување на заеднички симетричен клуч
  - не е за шифрирање или за потпишување
  - сигурноста е во тешкотијата да се реши дискретниот логаритамски проблем:
    - за дадени g, p, и $g^k \bmod p$ да се најде k

Diffie-Hellman

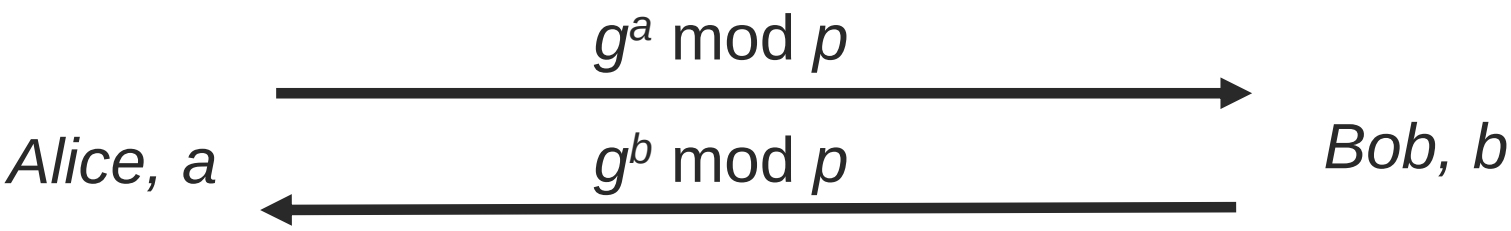
- Нека p е прост број, и g е **генератор**
  - За секој $x \in \{1,2,\ldots,p\text{-}1\}$ постои n т.ш. $x = g^n \bmod p$
- *Alice* бира тајна вредност a
- *Bob* бира тајна вредност b
- *Alice* праќа $g^a \bmod p$ на *Bob*
- *Bob* праќа $g^b \bmod p$ на *Alice*
- Двајцата пресметуваат $g^{ab} \bmod p$ кој ќе го користат како таен клуч
- Пример: p=7, g=5, a=2, b=3

Diffie-Hellman

- Да претпоставиме дека *Bob* и *Alice* користат $g^{ab} \bmod p$ како таен клуч
- *Trudy* може да ги види
  - $g^a \bmod p$ и $g^b \bmod p$
- *Trudy* не го знае тајниот клуч
  - $g^a g^b \bmod p = g^{a+b} \bmod p \neq g^{ab} \bmod p$
- Ако *Trudy* го најде *a* или *b*, системот е скршен
- Ако *Trudy* може да го реши дискретниот логаритамски проблем, тогаш таа може да го најде *a* или *b.*

Diffie-Hellman

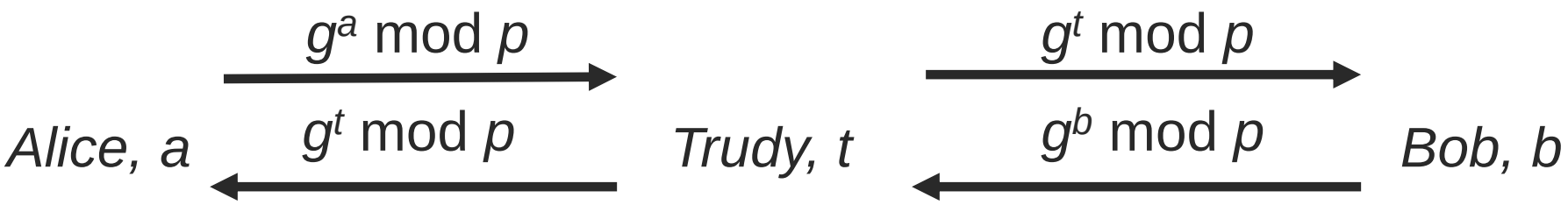
- **Јавно:** g и p
- **Тајно:** Експонентот a на *Alice*, експонентот b на *Bob*



- *Alice* пресметува $(g^b)^a = g^{ba} = g^{ab} \bmod p$
- *Bob* пресметува $(g^a)^b = g^{ab} \bmod p$
- Тие го користат $K = g^{ab} \bmod p$ како таен клуч

Diffie-Hellman

- Напад *man-in-the-middle*



- Активен напад во кој напаѓачот ги пресретнува пораките.
- Напаѓачот воспоставува заедничка тајна $g^{at} \bmod p$ со *Alice* и уште една заедничка тајна $g^{bt} \bmod p$ со *Bob*.
- *Alice* и *Bob* не знаат дека*Trudy* постои!
- Напаѓачот е во можност да ја измени која било порака.

Примена на криптографијата со јавен клуч

- Доверливост (*Confidentiality*)
- Автентикација (*Authentication*)
- Дигиталните потписи обезбедуваат интегритет (*integrity*) и неоспорливост (*non-repudiation*)
  - Нема неоспорливост кај симетричната криптографија

Нема неоспорливост

- *Alice* нарачува 100 акции за *Bob*
- *Alice* пресметува *MAC* користејќи симетричен клуч
- Вредноста на акциите паѓа и *Alice* тврди дека таа не нарачала
- Дали *Bob* може да докаже дека *Alice* направила нарачка?
- Не! *Bob* го знае симетричниот клуч, па и тој можел да ја фалсификува пораката
- Проблем: *Bob* знае дека *Alice* ја направила нарачката, но не може да докаже.

Неоспорливост

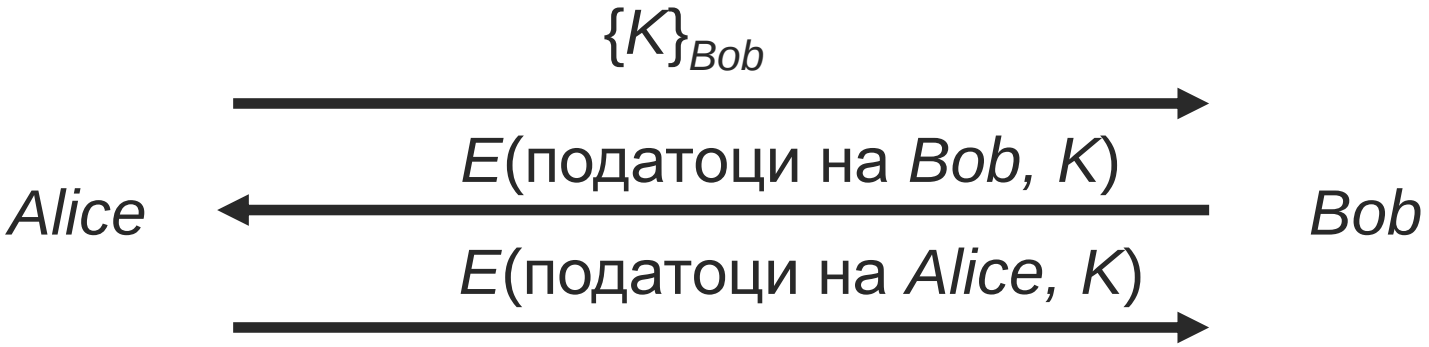
- *Alice* нарачува 100 акции за *Bob*
- *Alice* ја потпишува нарачката со нејзиниот таен клуч
- Вредноста на акциите паѓа и *Alice* тврди дека таа не нарачала
- Дали *Bob* може да докаже дека *Alice* направила нарачка?
- Да! Само оној со тајниот клуч на *Alice* можел да ја потпише пораката
- Претпоставуваме дека клучот на *Alice* не е украден.

Предости на криптографијата со јавен клуч

- Примарната предност на криптографијата со јавен клуч е тоа што не е потребен споделен клуч.
- Криптографијата со јавен клуч обезбедува доверливост.
- Криптографијата со јавен клуч може да се користи за потврда на интегритетот.
  - Кај криптографија со таен клуч, се користи MAC (Message Authentication Code – код за автентикација на пораки) за да се потврди интегритетот.
- Потписите со јавен клуч покрај тоа што обезбедуваат интегритет на податоците обезбедуваат и неоспорливост, нешто што симетричните клучеви не се во состојба да обезбедат.

Доверливост во реалниот свет

- Хибридни криптосистеми
  - Криптографија со јавен клуч за размена на клучот
  - Симетрична криптографија за шифрирање на пораките



- Дали *Bob* е сигурен дека зборува со *Alice*?

Хаш функции

Што е хаш функција?

- Хаш функцијата е функција која како влез има произволно долга низа од битови (или бајти), а излез е резултат со фиксна големина
- Се користи за шифрирање, автентификација, дигитален потпис...

Безбедносни својства

- Постојат неколку барања за хаш функциите
  - Мора да имаат својство на еднонасочност: за дадената порака m лесно е да се пресмета $h(m)$, но за дадена вредност x е невозможно да се најде m така што $h(m)=x$.
  - Мора да бидат отпорни на колизии: да биде невозможно да се конструираат две пораки m_1 и m_2 за кои хаш функцијата има иста вредност, т.е важи $h(m_1)=h(m_2)$.

Безбедносни својства

- Секоја хаш функција има бесконечен број на колизии.
  - Постојат бесконечен број на можни влезни вредности и само конечен број на возможни излезни вредности.
- Хаш функцијата никогаш не е ослободена од колизии.
- Отпорноста на колизии само го покажува тоа дека, иако колизиите постојат, тие не може да се најдат.

Отпорност на колизии

- Отпорноста на колизии е својство кое што ги прави хаш функциите погодни за користење во шемите за потпишување.
  - Отпорност на слаба колизија (*Weak collision resistance*): За дадено x и $h(x)$, тешко е да се најде $y \neq x$, така што $h(y) = h(x)$.
  - Отпорност на јака колизија (*Strong collision resistance*): Тешко е да се најдат x и y, $x \neq y$, така што $h(x) = h(y)$.

Роденденски проблем и хаш функции

- Каква е поврзаноста на роденденскиот проблем со хаш функциите?
- Да претпоставиме дека *h(x)* произведува излез со *N* бита.Тогаш постојат 2^N различни хаш вредности.
- Според роденденскиот проблем имаме дека ако направиме хаш функција на $2^{N/2}$ различни влезови, може да очекуваме дека ќе најдеме колизија, т.е. два влеза кои со хаш функција се пресликуваат во една иста вредност.

Роденденскиот напад

- Еден од општите напади на хаш функцијата е роденденскиот напад, кој што генерира колизии.
- За хаш функцијата која што има *N*-битен излез, потребни се $2^{N/2}$ чекори.

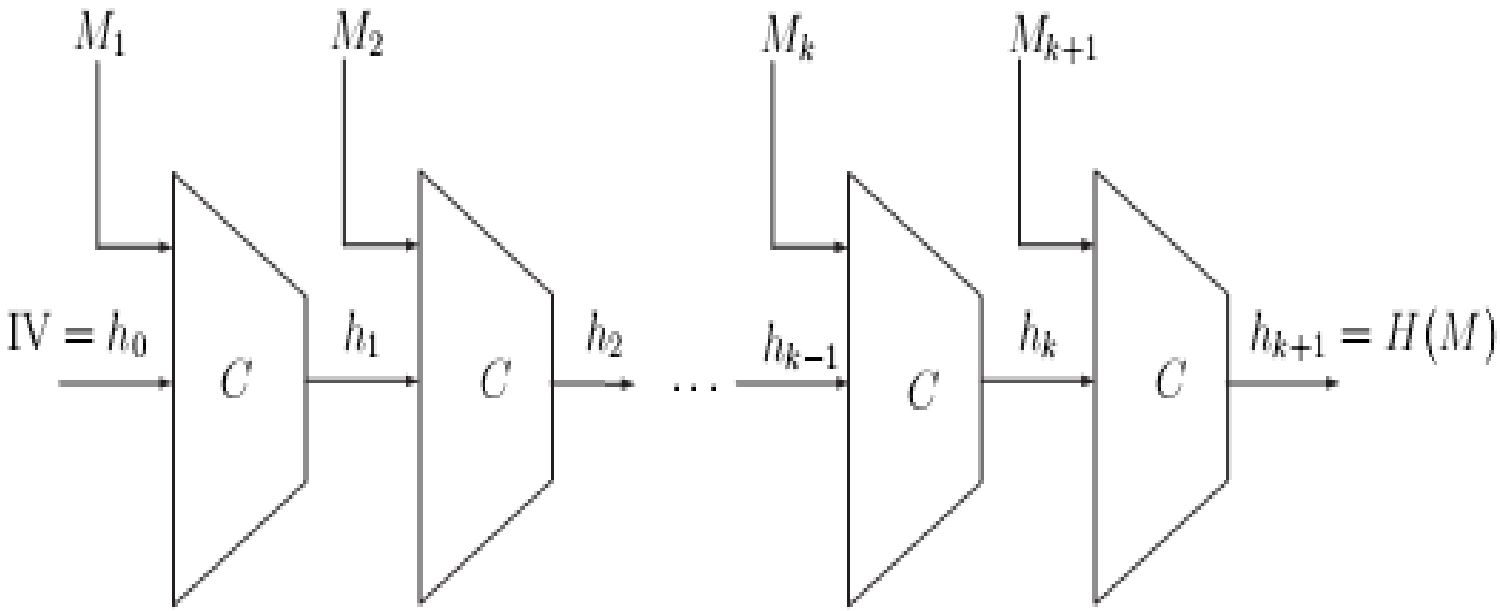
Реални хаш функции

- Речиси сите реални хаш функции се итеративни хаш функции.
- Итеративните хаш функции го делат влезот во низа од блокови со фиксна големина $m_1,\ldots,m_k$, користејќи правила за пополнување за да се пополни последниот блок.
  - Типична големина на блокот е 512 бита.

Реални хаш функции

- Блоковите од пораки редоследно се процесираат, користејќи функција за компресија и меѓусостојба со фиксна големина.
- Процесот започнува со фиксна вредност H_0 и дефинира $H_i=h'(H_i\text{-}1,m_i)$.
- Крајната вредност H_k е резултат од хаш функцијата.

Реални хаш функции



Реални хаш функции

- Попознати хаш функции.
  - SHA - стандардизирани од NIST

    “SHA” - е кратенка од “secure hash algorithm”
  - MD5 - “MD” е кратенка од “message digest”

MD5

- MD5 е 128-битна хаш функција
- Развиена од Рон Ривест (Ron Rivest)
- Таа претставува надградба на MD4 со додатна отпорност против напади.
- Оваа хаш функција работи со 32-битни зборови и е многу ефикасна на 32-битни процесори.

MD5

- За повеќето примени 128-битната хаш големина на MD5 е недоволна.
- Користејќи го роденденскиот парадокс, може да најдеме реална MD5 колизија при приближно 2^{64} извршувања на хаш функцијата.
- Ова е недоволно за модерен систем.

SHA

- SHA (Secure Hash Algorithm), е направен од страна на NSA и стандардизиран од страна на NIST.
- Првата верзија е наречена едноставно SHA (често се нарекува SHA-0), но таа содржела слабости.
- NSA ги нашла овие недостатоци, и направила исправки кои што NIST ги публикувал во подобрена верзија наречена SHA-1.

SHA

- SHA-1 е 160-битна хаш функција базирана на MD4.
- Бидејќи има заеднички родител со MD5 имаат и слични заеднички особини
- Таа е два до три пати поспора од MD5.
- SHA-1 е широко користена.

SHA

- Слично како MD5 таа работи со 32-битни зборови.
- Главниот проблем со SHA-1 е нејзината 160-битна големина на резултатот.
- Колизиите може да бидат генерирани во само 2^{80} чекори.
- Ова е недоволно за модерен систем.

SHA-256, SHA-384 и SHA-512

- Тие имаат 256-, 384- и 512-битен излез соодветно.
- Тие се дизајнирани да бидат користени со 128-, 196- и 256-битни клучеви од AES.
- Нивната стуктура е многу слична на SHA-1.

SHA функции - споредба

Comparison of SHA functions

Algorithm and variant		Output size (bits)	Internal state size (bits)	Block size (bits)	Max message size (bits)	Rounds	Operations	Security (bits)	Example Performance[29] (MiB/s)
MD5 (as reference)		128	128 (4 × 32)	512	$2^{64} - 1$	64	And, Xor, Rot, Add (mod 2^{32}), Or	<64 (collisions found)	335
SHA-0		160	160 (5 × 32)	512	$2^{64} - 1$	80	And, Xor, Rot, Add (mod 2^{32}), Or	<80 (collisions found)	-
SHA-1		160	160 (5 × 32)	512	$2^{64} - 1$	80		<80 (theoretical attack[30] in 2^{61})	192
SHA-2	*SHA-224*	224	256 (8 × 32)	512	$2^{64} - 1$	64	And, Xor, Rot, Add (mod 2^{32}), Or, Shr	112	139
	SHA-256	256						128	
	SHA-384	384	512 (8 × 64)	1024	$2^{128} - 1$	80	And, Xor, Rot, Add (mod 2^{64}), Or, Shr	192	154
	SHA-512	512						256	
	SHA-512/224	224						112	
	SHA-512/256	256						128	
SHA-3	*SHA3-224*	224	1600 (5 × 5 × 64)	1152	Unlimited	24	And, Xor, Rot, Not	112	-
	SHA3-256	256		1088				128	
	SHA3-384	384		832				192	
	SHA3-512	512		576				256	
	SHAKE128	*d* (arbitrary)		1344				min (*d*/2, 128)	-
	SHAKE256	*d* (arbitrary)		1088				min (*d*/2, 256)	

Кодови за автентификација на пораки

- *Кодот за автентификација на пораки*, MAC (message authentication code) е конструкција која спречува поигрување со пораките.
- Шифрирањето ја спречува Ева-натрапникот да ја прочита пораката, но не ја спречува да манипулира со неа.
- MAC користи таен клуч, *K*, познат и на Алис и на Боб, но не и на Ева.

Кодови за автентификација на пораки

- Алис не само што ја испраќа пораката *m*, туку испраќа и MAC вредност пресметана со помош на MAC функција.
- Боб проверува дали MAC вредноста на добиената пораката одговара со применатa MAC вредност.
- Ева не може да манипулира со пораката затоа што без *K* таа не може да најде коректена MAC вредност.

MAC

- MAC е функција која што како влез има два аргумента MAC(*K*,*m*):
  - клуч *K* со фиксна големина и
  - порака *m* со произволна големина,

  а како излез дава MAC вредност со фиксна големина.
- За да ја автентификува пораката, Алис праќа не само порака *m* туку и MAC кодот MAC(*K*,*m*).

HMAC

- Идејата е да се користи хаш функција за да се изгради MAC.
- HMAC пресметува $h(K \oplus a || h(K \oplus b || m))$, каде a и b се определени константи.
- Пораката е хаширана само еднаш и излезот е повторно хаширан со клучот.
- HMAC работи со било која итеративна хаш функција и широко е користена со MD5 и SHA-1.

Генератори на случајни броеви

Генератори на псевдо случајни броеви

- “Псевдо” е употребен да означи дека постапката на генерирање на случајни броеви со познат метод ја отстранува потенцијалната чиста случајност
- Целта на секоја генераторна шема е да произведе низа од броеви помеѓу нула и еден која симулира идеални особини на рамномерната распределба и независноста што е можно поблиску.

Генератори на псевдо случајни броеви

- Кои грешки или отклонувања може да настанат од идеалната случајност:
  - Генерираните броеви може да не бидат рамномерно распределени.
  - Средната вредност на генерираните броеви може да биде премногу голема или премногу мала.

Генератори на псевдо случајни броеви

- Постојат многу зависности:
  - Автокорелација помеѓу броевите
  - Последователните броеви се поголеми или помали од претходните броеви
  - Неколку броеви под средната вредност се следени со неколку броеви над средната вредност

Генератори на псевдо случајни броеви

- Кои работи треба да се разгледуваат при избор на метод:
  - Методот треба да биде брз. Индивидуалните пресметувања се доста ефтини, но понекогаш се потребни неколку илјади случајни броеви. Вкупната цена на пресметување на методот може да се добие ако се пресмета пресметувачката ефикасност на методот за генерирање на случајни броеви.
  - Методот треба да биде преносliва за различни компјутери и идеална за различни програмски јазици.

Генератори на псевдо случајни броеви

- Низата треба да има доволно долг период. Должината на периодата претставува должината на низата на случајните броеви и таа треба да биде голема за да не се случи броевите повторно да се повторуваат во ист редослед.
- И најважно и како што е спомнато претходно, генерирањето на случајни броеви треба да биде апроксимирано со идеалните статистички особини на рамномерноста и на независноста.

Генератори на псевдо случајни броеви

- Тестови за Случајни броеви
  - Тест на фрекфренција
  - Runs тестови
  - Автокорелациони тестови
  - Тестови за откривање на празнини
  - Покер тест