Introduction à la Sécurité

Cryptologie symétrique 1/2

Damien Vergnaud

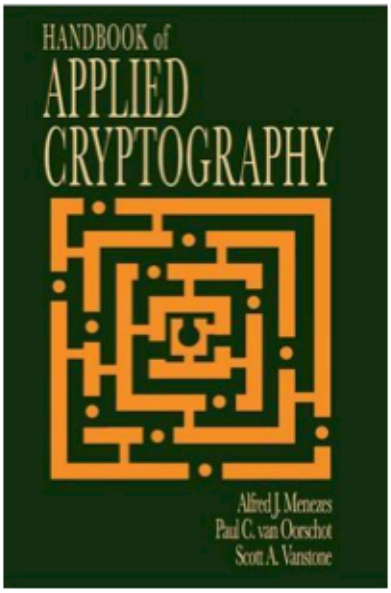
Sorbonne Université
Institut Universitaire de France

ISEC

References

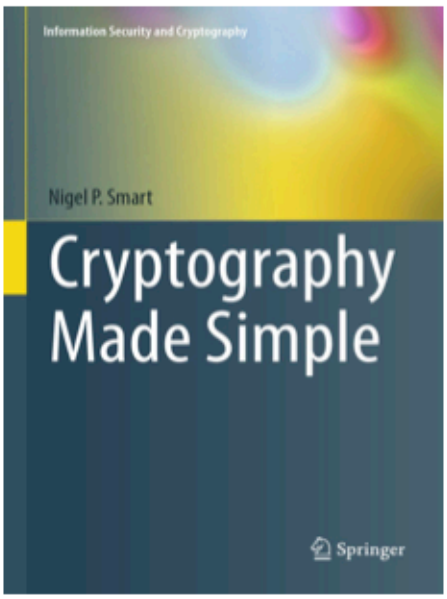
Cryptographie : Théorie et pratique
Douglas Stinson

References



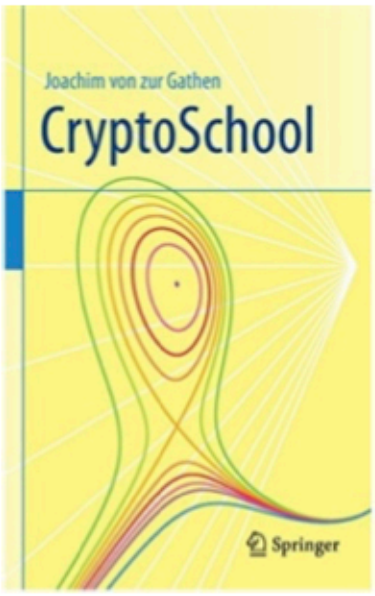
Handbook of Applied Cryptography
Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone

References



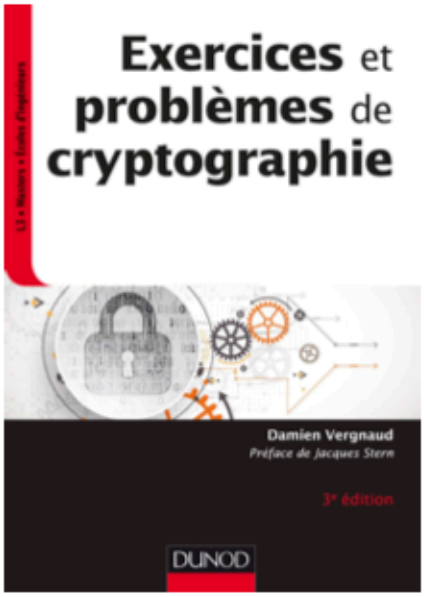
Cryptography Made Simple
Nigel P. Smart

References



CryptoSchool
Joachim von zur Gathen

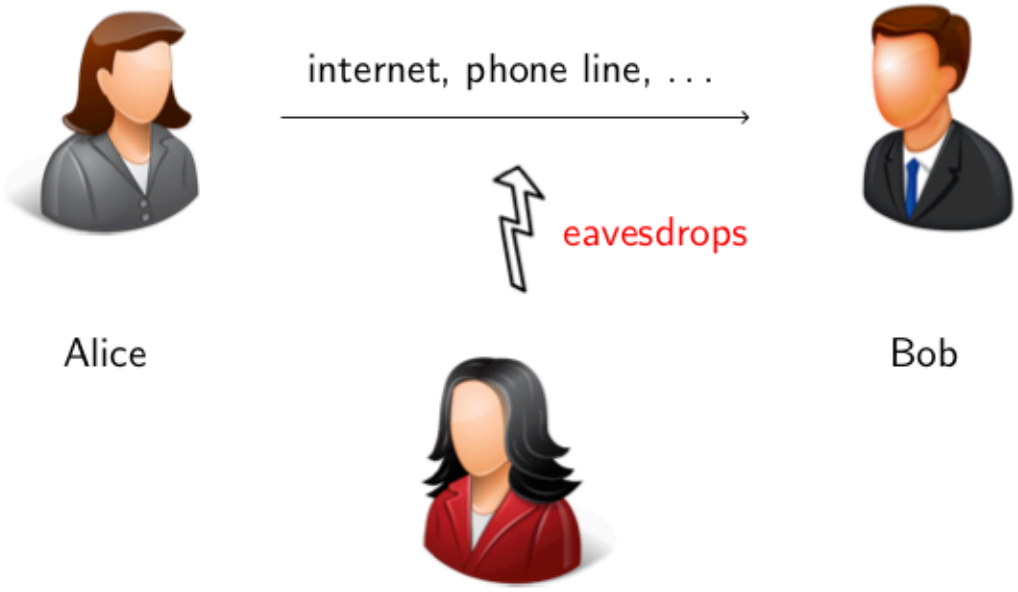
References



Exercices et problèmes de cryptographie
Damien Vergnaud

Contents

The basic goal: **secure** communication



Information security objectives

Cryptology = practice and study of hiding information

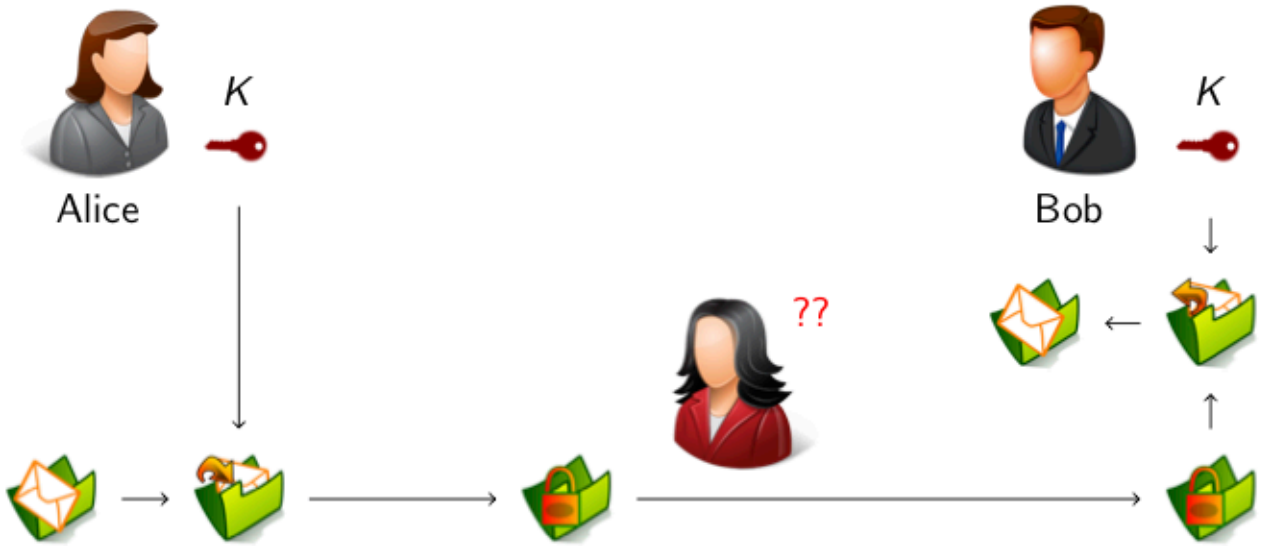
confidentiality	keeping information secret from all but those who are authorized to see it
integrity	ensuring information has not been altered by unauthorized or unknown means
message authentication	corroborating the source of information
signature	a means to bind information to an entity
receipt	acknowledgement that information has been received
anonymity	concealing the identity of an entity involved in some process
non-repudiation	preventing the denial of previous commitments or actions
etc	*etc*

Terminology

- Cryptography
- Crytanalysis (Cryptanalyst)
- Cryptology
- Cipher/Encryption Algorithm
- Encryption/Encipherment
- Decryption/Decipherment
- Plaintext
- Ciphertext

Secret Key Cryptosystems

Symmetric encryption: Alice and Bob share a "key" K



- Bob can use the same method to send messages to Alice.
  $\rightsquigarrow$ **symmetric setting**
- How did Alice and Bob establish K?

Kerckhoffs's principle

In 1883 Auguste Kerckhoffs wrote two journal articles on *La Cryptographie Militaire*:

- Le système doit être matériellement, sinon mathématiquement, indéchiffrable
  - **The system must be practically, if not mathematically, indecipherable**
- Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi
  - **It must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience**
- La clef doit pouvoir en être communiquée et retenue sans le secours de notes écrites, et être changée ou modifiée au gré des correspondants
  - **Its key must be communicable and retainable without the help of written notes, and changeable or modifiable at the will of the correspondents**

Kerckhoffs's principle

- Il faut qu'il soit applicable à la correspondance télégraphique
  - **It must be applicable to telegraphic correspondence**
- Il faut qu'il soit portatif, et que son maniement ou son fonctionnement n'exige pas le concours de plusieurs personnes
  - **It must be portable, and its usage and function must not require the concourse of several people**
- Enfin, il est nécessaire, vu les circonstances qui en commandent l'application, que le système soit d'un usage facile, ne demandant ni tension d'esprit, ni la connaissance d'une longue série de règles à observer
  - **Finally, it is necessary, given the circumstances that command its application, that the system be easy to use, requiring neither mental strain nor the knowledge of a long series of rules to observe**

Shift Cipher (Caesar Cipher)

Each letter we identify with a number

- A = 0
- B = 1
- C = 2
- ...
- Z = 25

The key k is a number in the range $\{0, \ldots, 25\} = \mathbb{Z}_{26}$

- Encryption is add k onto each letter modulo 26.

Julius Caesar used the key $k = 3$.

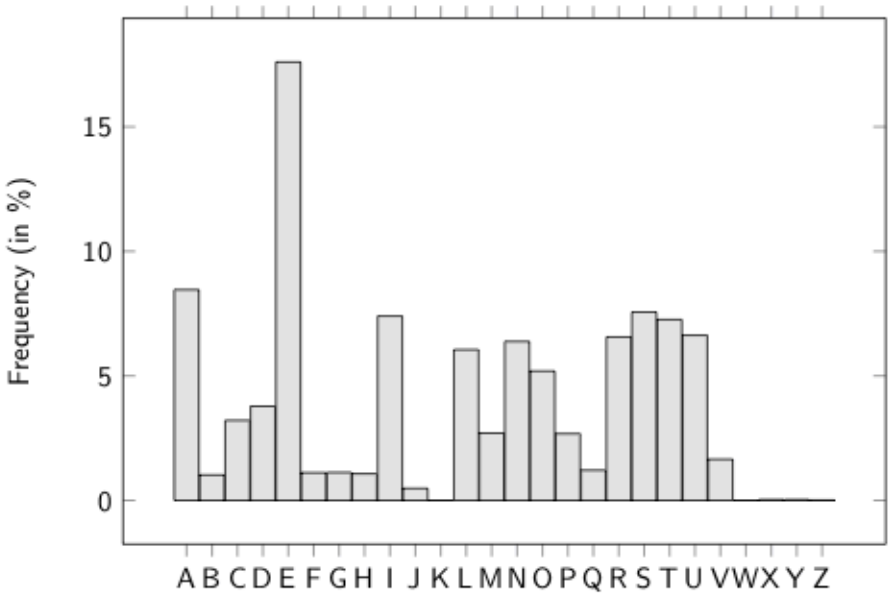
- HELLO becomes KHOOR

We break a shift cipher by using the **statistics** of the underlying language

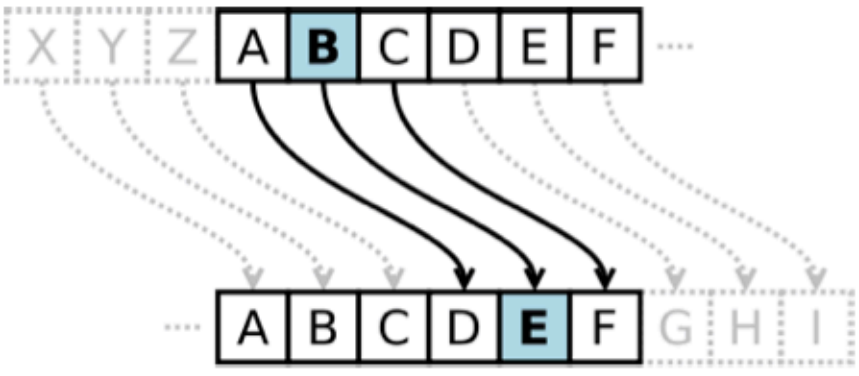
Frequency Distribution of Letters (in French)

(*Notre-Dame de Paris*)

a	b	c	d	e	f	g	h	i	j	k	l	m
8,46	1,02	3,21	3,78	17,60	1,11	1,12	1,07	7,40	0,48	0	6,05	2,70
n	o	p	q	r	s	t	u	v	w	x	y	z
6,38	5,19	2,68	1,21	6,56	7,56	7,26	6,63	1,65	0	0,03	0,03	0,01



Shift Cipher (Caesar Cipher)



Problems:

- the number of keys is too small (only have 26 possible keys)
- easy cryptanalysis: match up the frequency distribution of the letters

Substitution cipher

- increase the number of keys $\rightsquigarrow$ **substitution cipher**
- Encryption replaces each letter by its permuted version
- Decryption uses the inverse permutation

ABCDEFGHIJKLMNOPQRSTUVWXYZ

TMKGOYDSIPELUAVCRJWXZNHBQF

$\rightsquigarrow$ HELLO encrypts to SOLLV

Substitution cipher

- Number of keys is $26! \simeq 4.03 \cdot 10^{26} \simeq 2^{88}$
- Feasible to only run a computer on a problem which takes under 2^{80} steps. This is far too large a number to brute force search using modern computers.
- Still we can break these ciphers using **statistics** of the underlying plaintext language.

$\rightsquigarrow$ Un exemple en français ...

Substitution mono-alphabétique

v ubcfb osu ymoqsuu n cxqfj dqmfnu ub vjcfqu juz amqjmruz zmsscfusb bquflu auoquz
hfszbms zwfba ju wusbms qusbqu ncsz ju vmo z uddmqvcfb n uxfbuq ju xusb wcoxcfz fj
eczzc qcefnuwusb jc emqbu xfbquu no ijmv nuz wcfzmsz nu jc xfvbmfqu ecz czzul qcefnuwusb
vueusncsb emoq uweuvauq kou z usrmoddqu us wuwu buwez kou jof os bmoqifjjms nu emozzfuqu
ub nu zciju
ju acjj zusbcfb ju vamo vofb ub ju xfuog bcefz c j osu nu zuz ugbquwfbuz osu cddfvau nu
vmojuoq bqme xczbu emoq vu nuejmfuwusb fsbuqfuoq ubcfb vjmouu co woq ujju quequzusbcfb
zfwejuwusb os usmqwu xfzcru jcqru nu ejoz n os wubqu ju xfzcru n os amwwu n usxfqms
kocqcsbu vfsk csz c j uecfzzu wmozbcvau smfqu cog bqcfbz cvvusbouz ub iucog

hfszbms zu nfqfruc xuqz j uzvcjfuq fj ubcfb fsobfju n uzzcpuq nu equsnqu j czvuszuoq
wuwu cox wufjjuoquz uemkouz fj dmsvbfmsscfb qcquwusb cvboujjuwusb n cfjjuoqz ju vmoqcsb
ujuvbqfkou ubcfb vmoeu ncsz jc ymoqsuu v ubcfb osu nuz wuzoquz n uvmsmwfu eqfzuz us xou nu
jc zuwcfsu nu jc acfsu
zms ceecqbuwusb ubcfb co zuebfuwu hfszbms kof cxcfb bqusbu suod csz ub zmoddqcfb n os
ojvuqu xcqfkouog co nuzzoz nu jc vauxfjju nqmfbu wmsbcfb jusbuwusb fj z cqqubc ejozfuoqz
dmfz us vauwfs emoq zu quemzuq c vackou ecjfuq zoq osu cddfvau vmjjuu co woq dcvu c jc
vcru nu j czvuszuoq j usmqwu xfzcru xmoz dfgcfb no qurcqn v ubcfb os nu vuz emqbqcfbz
cqqcsruz nu bujju zmqbu kou juz puog zuwijusb zofxqu vujof kof eczzu osu jurusnu zmoz ju
emqbqcfb nfzcfb ifr iqmbauq xmoz qurcqnu

c j fsbuqfuoq nu j ceecqbuwusb nu hfszbms osu xmfg zovquu dcfzcfb usbusnqu osu zuqfu nu
smwiquz kof cxcfusb bqcfb c jc eqmnovbfms nu jc dmsbu jc xmfg eqmxuscfb n osu ejckou nu
wubcj mijmsrou wfqmfq buqsu usvczbqu ncsz ju woq nu nqmfbu hfszbms bmoqsc os imobms ub
jc xmfg nfwfsoc nu xmjowu wcfz juz wmbz ubcfusb usvmqu nfzbfsvbz ju zms nu j ceecqufj no
bujuvqcs vmwwu ms nfzcfb emoxcfb ubqu czzmoqnf wcfz fj s p cxcfb covos wmpus nu j ubufsnqu
vmwejubuwusb hfszbms zu nfqfruc xuqz jc dusubqu fj ubcfb nu zbcboqu dquju ejobmb eubfbu
ub zc wcfrquoq ubcfb zmojfrsuu ecq jc vmwifscfzms ijuou osfdmqwu no ecqbf fj cxcfb juz
vauxuog bquz ijmsnz ju xfzcru scboqujjuwusb zcsrofs jc euco noqvfu ecq ju zcxms rqmzzfuq
juz jcwuz nu qczmfq uwmozzuuz ub ju dqmfn nu j afxuq kof xuscfb nu equsnqu dfs

Substitution mono-alphabétique

Nombre d'occurrences de chaque caractère du texte chiffré :

a	b	c	d	e	f	g	h	i	j	k	l	m
17	141	150	23	48	146	11	6	13	106	14	2	95
n	o	p	q	r	s	t	u	v	w	x	y	z
71	113	4	129	20	129	0	329	51	57	35	2	128

Il est vraisemblable que le caractère "u" représente le caractère "e" dans le texte clair

Substitution mono-alphabétique

Les bigrammes les plus fréquents de la langue française sont

es, *le*, *re*, *de*, *en*, *et*, *ai*, *te*, *ou*, *nt*, *it*, *er* et *la*

Les bigrammes commençant par "u" dans le texte chiffré ne sont pas assez fréquents pour décider quel caractère correspond à la lettre "s" dans le texte clair.

Les bigrammes les plus fréquents du texte chiffré finissant par "u" sont

- "ju" (18 occurrences)
- "nu" (15 occurrences)

ce qui suggère que "j" représente "l" et "n" représente "d".

v ebcfb ose ymoqsee d cxqfl dqmfde eb vlcfqe lez amqlmrez zmsscfesb bqefle aeoqez
hfszbms zwfba le wesbms qesbqe dcsz le vmo z eddmqvcfb d exfbeq le xesb wcoxcfz fl
eczzc qcefdewesb lc emqbe xfbqee do ilmv dez wcfzmsz de lc xfvbmfqe ecz czzel qcefdewesb
veeesdcsb emoq eweevaeq koe z esrmoddqe es wewe bewez koe lof os bmoqifllms de emozzfeqe
eb de zcile
le acll zesbcfb le vamo vofb eb le xfeog bcefz c l ose de zez egbqewfbez ose cddfvae de
vmoleoq bqme xczbe emoq ve deelmfewesb fsbeqfeoq ebcfb vlmoee co woq elle qeeqezesbcfb
zfwelewesb os esmqwe xfzcre lcqre de eloz d os webqe le xfzcre d os amwwe d esxfqms
kocqcsbe vfsk csz c l eecfzze wmozbcvae smfqe cog bqcfbz cvvesboez eb iecog

hfszbms ze dfqfrec xeqz l ezvclfeq fl ebcfb fsobfle d ezzcpeq de eqesdqe l czveszeoq
wewe cog weflleoqez eemkoez fl dmsvbfmsscfb qcqewesb cvboellewesb d cflleoqz le vmoqcsb
elevbqfkoe ebcfb vmoee dcsz lc ymoqsee v ebcfb ose dez wezoqez d evmsmwfe eqfzez es xoe de
lc zewcfse de lc acfse
zms ceecqbewesb ebcfb co zeebfewe hfszbms kof cxcfb bqesbe seod csz eb zmoddqcfb d os
olveqe xcqfkoeog co dezzoz de lc vaexflle dqmfbe wmsbcfb lesbewesb fl z cqqebc elozfeoqz
dmfz es vaewfs emoq ze qeemzeq c vackoe eclfeq zoq ose cddfvae vmllee co woq dcve c lc
vcre de l czveszeoq l esmqwe xfzcre xmoz dfgcfb do qercqd v ebcfb os de vez emqbqcfbz
cqqcsrez de belle zmqbe koe lez peog zewilesb zofxqe velof kof eczze ose leresde zmoz le
emqbqcfb dfzcfb ifr iqmbaeq xmoz qercqde

c l fsbeqfeoq de l ceecqbewesb de hfszbms ose xmfg zovqee dcfzcfb esbesdqe ose zeqfe de
smwiqez kof cxcfesb bqcfb c lc eqmdovbfms de lc dmsbe lc xmfg eqmxescfb d ose elckoe de
webcl milmsroe wfqmfq beqse esvczbqe dcsz le woq de dqmfbe hfszbms bmoqsc os imobms eb
lc xmfg dfwfsoc de xmlowe wcfz lez wmbz ebcfesb esvmqe dfzbfsvbz le zms de l ceecqefl do
belevqcs vmwwe ms dfzcfb emoxcfb ebqe czzmoqdf wcfz fl s p cxcfb covos wmpes de l ebefsdqe
vmwelebewesb hfszbms ze dfqfrec xeqz lc desebqe fl ebcfb de zbcboqe dqele elobmb eebfbe
eb zc wcfrqeoq ebcfb zmolfrsee ecq lc vmwifscfzms ileoe osfdmqwe do ecqbf fl cxcfb lez
vaexeog bqez ilmsdz le xfzcre scboqellewesb zcsrofs lc eeco doqvfe ecq le zcxms rqmzzfeq
lez lcwez de qczmfq ewmozzeez eb le dqmfd de l afxeq kof xescfb de eqesdqe dfs

v ebcfb ose ymoqsee d cxqfl dqmfde eb vlcfqe lez amqlmrez zmsscfesb bqefle aeoqez hfszbms zmfba le mesbms qesbqe dcsz le vmo z eddmqvcfb d exfbeq le xesb mcoxcfz fl eczzc qcefdemesb lc emqbe xfbqee do ilmv dez mcfzmsz de lc xfvbmfqe ecz czzel qcefdemesb veeesdcsb emoq emeevaeq koe z esrmoddqe es meme bemez koe lof os bmoqifllms de emozzfeqe eb de zcile

le acll zesbcfb le vamo vofb eb le xfeog bcefz c l ose de zez egbqemfbez ose cddfvae de vmoleoq bqme xczbe emoq ve deelmfemesb fsbeqfeoq ebcfb vlmoee co moq elle qeeqezesbcfb zfmelemesb os esmqme xfzcre lcqre de eloz d os mebqe le xfzcre d os ammme d esxfqms kocqcsbe vfsk csz c l eecfzze mmozbcvae smfqe cog bqcfbz cvvesboez eb iecog

hfszbms ze dfqfrec xeqz l ezvclfeq fl ebcfb fsobfle d ezzcpeq de eqesdqe l czveszeoq meme cog meflleoqez eemkoez fl dmsvbfmsscfb qcqemesb cvboellemesb d cflleoqz le vmoqcsb elevbqfkoe ebcfb vmoee dcsz lc ymoqsee v ebcfb ose dez mezoqez d evmsmmfe eqfzez es xoe de lc zemcfse de lc acfse

zms ceecqbemesb ebcfb co zeebfeme hfszbms kof cxcfb bqesbe seod csz eb zmoddqcfb d os olveqe xcqfkoeog co dezzoz de lc vaexflle dqmfbe mmsbcfb lesbemesb fl z cqqebc elozfeoqz dmfz es vaemfs emoq ze qeemzeq c vackoe eclfeq zoq ose cddfvae vmllee co moq dcve c lc vcre de l czveszeoq l esmqme xfzcre xmoz dfgcfb do qercqd v ebcfb os de vez emqbqcfbz cqqcsrez de belle zmqbe koe lez peog zemilesb zofxqe velof kof eczze ose leresde zmoz le emqbqcfb dfzcfb ifr iqmbaeq xmoz qercqde

c l fsbeqfeoq de l ceecqbemesb de hfszbms ose xmfg zovqee dcfzcfb esbesdqe ose zeqfe de smmiqez kof cxcfesb bqcfb c lc eqmdovbfms de lc dmsbe lc xmfg eqmxescfb d ose elckoe de mebcl milmsroe mfqmfq beqse esvczbqe dcsz le moq de dqmfbe hfszbms bmoqsc os imobms eb lc xmfg dfmfsoc de xmlome mcfz lez mmbz ebcfesb esvmqe dfzbfsvbz le zms de l ceecqefl do belevqcs vmmme ms dfzcfb emoxcfb ebqe czzmoqdf mcfz fl s p cxcfb covos mmpes de l ebefsdqe vmmelebemesb hfszbms ze dfqfrec xeqz lc desebqe fl ebcfb de zbcboqe dqele elobmb eebfbe eb zc mcfrqeoq ebcfb zmolfrsee ecq lc vmmifscfzms ileoe osfdmqme do ecqbf fl cxcfb lez vaexeog bqez ilmsdz le xfzcre scboqellemesb zcsrofs lc eeco doqvfe ecq le zcxms rqmzzfeq lez lcmez de qczmfq emmozzeez eb le dqmfd de l afxeq kof xescfb de eqesdqe dfs

v etcft one ymoqnee d cxqfl dqmfde et vlcfqe lez amqlmrez zmnncfent tqefle aeoqez hfnztmn zmfta le mentmn qentqe dcnz le vmo z eddmqvcft d exfteq le xent mcoxcfz fl eczzc qcefdement lc emqte xftqee do ilmv dez mcfzmnz de lc xfvtmfqe ecz czzel qcefdement veeendcnt emoq emeevaeq koe z enrmoddqe en meme temez koe lof on tmoqifllmn de emozzfeqe et de zcile

le acll zentcft le vamo voft et le xfeog tcefz c l one de zez emtqemftez one cddfvae de vmoleoq tqme xczte emoq ve deelmfement fnteqfeoq etcft vlmoee co moq elle qeeqezentcft zfmelement on enmqme xfzcre lcqre de eloz d on metqe le xfzcre d on ammme d enxfqmn kocqcnte vfnk cnz c l eecfzze mmoztcvae nmfqe cog tqcftz cvventoez et iecog

hfnztmn ze dfqfrec xeqz l ezvclfeq fl etcft fnotfle d ezzcpeq de eqendqe l czvenzeoq meme cog meflleoqez eemkoez fl dmnvtfmnncft qcqement cvtoellement d cflleoqz le vmoqcnt elevtqfkoe etcft vmoee dcnz lc ymoqnee v etcft one dez mezoqez d evmnmmfe eqfzez en xoe de lc zemcfne de lc acfne

zmn ceecqtement etcft co zeetfeme hfnztmn kof cxcft tqente neod cnz et zmoddqcft d on olveqe xcqfkoeog co dezzoz de lc vaexflle dqmfte mmntcft lentement fl z cqqetc elozfeoqz dmfz en vaemfn emoq ze qeemzeq c vackoe eclfeq zoq one cddfvae vmllee co moq dcve c lc vcre de l czvenzeoq l enmqme xfzcre xmoz dfmcft do qercqd v etcft on de vez emqtqcftz cqqcnrez de telle zmqte koe lez peog zemilent zofxqe velof kof eczze one lerende zmoz le emqtqcft dfzcft ifr iqmtaeq xmoz qercqde

c l fnteqfeoq de l ceecqtement de hfnztmn one xmfg zovqee dcfzcft entendqe one zeqfe de nmmiqez kof cxcfent tqcft c lc eqmdovtfmn de lc dmnte lc xmfg eqmxencft d one elckoe de metcl milmnroe mfqmfq teqne envcztqe dcnz le moq de dqmfte hfnztmn tmoqnc on imotmn et lc xmfg dfmfnoc de xmlome mcfz lez mmtz etcfent envmqe dfztfnvtz le zmn de l ceecqefl do televqcn vmmme mn dfzcft emoxcft etqe czzmoqdf mcfz fl n p cxcft covon mmpen de l etefndqe vmmeletement hfnztmn ze dfqfrec xeqz lc denetqe fl etcft de ztctoqe dqele elotmt eetfte et zc mcfrqeoq etcft zmolfrnee ecq lc vmmifncfzmn ileoe onfdmqme do ecqtf fl cxcft lez vaexeog tqez ilmndz le xfzcre nctoqellement zcnrofn lc eeco doqvfe ecq le zcxmn rqmzzfeq lez lcmez de qczmfq emmozzeez et le dqmfd de l afxeq kof xencft de eqendqe dfn

v etcft one yoornee d cxrfl drofde et vlcfre lez aorlorez zonncfent trefle aeorez hfnzton zmfta le menton rentre dcnz le voo z eddorvcft d exfter le xent mcoxcfz fl eczzc rcefdement lc eorte xftree do ilov dez mcfzonz de lc xfvtofre ecz czzel rcefdement veeendcnt eoor emeevaer koe z enrooddre en meme temez koe lof on toorifllon de eoozzfere et de zcile

le acll zentcft le vaoo voft et le xfeog tcefz c l one de zez egtremftez one cddfvae de vooleor troe xczte eoor ve deelofement fnterfeor etcft vlooee co mor elle reerezentcft zfmelement on enorme xfzcre lcrre de eloz d on metre le xfzcre d on aomme d enxfron kocrcnte vfnk cnz c l eecfzze mooztcvae nofre cog trcftz cvventoez et iecog

hfnzton ze dfrfrec xerz l ezvclfer fl etcft fnotfle d ezzcper de erendre l czvenzeor meme cog meflleorez eeokoez fl donvtfonncft rcrement cvtoellement d cflleorz le voorcnt elevtrfkoe etcft vooee dcnz lc yoornee v etcft one dez mezorez d evonomfe erfzez en xoe de lc zemcfne de lc acfne

zon ceecrtement etcft co zeetfeme hfnzton kof cxcft trente neod cnz et zooddrcft d on olvere xcrfkoeog co dezzoz de lc vaexflle drofte montcft lentement fl z crretc elozfeorz dofz en vaemfn eoor ze reeozer c vackoe eclfer zor one cddfvae vollee co mor dcve c lc vcre de l czvenzeor l enorme xfzcre xooz dfocft do rercrd v etcft on de vez eortrcftz crrcnrez de telle zorte koe lez peog zemilent zofxre velof kof eczze one lerende zooz le eortrcft dfzcft ifr irotaer xooz rercrde

c l fnterfeor de l ceecrtement de hfnzton one xofg zovree dcfzcft entendre one zerfe de nomirez kof cxcfent trcft c lc erodovtfon de lc donte lc xofg eroxencft d one elckoe de metcl oilonroe mfrofr terne envcztre dcnz le mor de drofte hfnzton toornc on iooton et lc xofg dfmfnoc de xolome mcfz lez motz etcfent envore dfztfnvtz le zon de l ceecrefl do televrcn vomme on dfzcft eooxcft etre czzoordf mcfz fl n p cxcft covon mopen de l etefndre vomeletement hfnzton ze dfrfrec xerz lc denetre fl etcft de ztctore drele elotot eetfte et zc mcfrreor etcft zoolfrnee ecr lc vomifncfzon ileoe onfdorme do ecrtf fl cxcft lez vaexeog trez ilondz le xfzcre nctorellement zcnrofn lc eeco dorvfe ecr le zcxon rrozzfer lez lcmez de rczofr emoozzeez et le drofd de l afxer kof xencft de erendre dfn

v etait une yournee d axril droide et vlaire les aorlores sonnaient treile aeures hinston smita le menton rentre dans le vou s eddorvait d exiter le xent mauxais il passa rapidement la porte xitree du blov des maisons de la xivtoire pas assel rapidement vependant pour empevaer kue s enrouddre en meme temps kue lui un tourbillon de poussiere et de sable

le aall sentait le vaou vuit et le xieug tapis a l une de ses egtremites une addivae de vouleur trop xaste pour ve deploiement interieur etait vlouee au mur elle representait simplement un enorme xisare larre de plus d un metre le xisare d un aomme d enxiron kuarante vink ans a l epaisse moustavae noire aug traits avventues et beaug

hinston se dirirea xers l esvalier il etait inutile d essaper de prendre l asvenseur meme aug meilleures epokues il donvtionnait rarement avtuellement d ailleurs le vourant elevtrikue etait voupe dans la yournee v etait une des mesures d evonomie prises en xue de la semaine de la aaine

son appartement etait au septieme hinston kui axait trente neud ans et souddrait d un ulvere xarikueug au dessus de la vaexille droite montait lentement il s arreta plusieurs dois en vaemin pour se reposer a vaakue palier sur une addivae vollee au mur dave a la vare de l asvenseur l enorme xisare xous digait du rerard v etait un de ves portraits arranres de telle sorte kue les peug semblent suixre velui kui passe une lerende sous le portrait disait bir brotaer xous rerarde

a l interieur de l appartement de hinston une xoig suvree daisait entendre une serie de nombres kui axaient trait a la produvtion de la donte la xoig proxenait d une plakue de metal oblonrue miroir terne envastre dans le mur de droite hinston tourna un bouton et la xoig diminua de xolume mais les mots etaient envore distinvts le son de l appareil du televran vomme on disait pouxait etre assourdi mais il n p axait auvun mopen de l eteindre vompletement hinston se dirirea xers la denetre il etait de stature drele plutot petite et sa mairreur etait soulirnee par la vombinaison bleue unidorme du parti il axait les vaexeug tres blonds le xisare naturellement sanruin la peau durvie par le saxon rrossier les lames de rasoir emoussees et le droid de l aixer kui xenait de prendre din

a	b	c	d	e	f	g	h	i	j	k	l	m
h	t	a	f	p	i	x	w	b	l	q	z	o
n	o	p	q	r	s	t	u	v	w	x	y	z
d	u	y	r	g	n	k	e	c	m	v	j	s

Vigenère Cipher

- **Problem with Caesar and Substitution cipher:** each plaintext letter always encrypted to the same ciphertext letter.
- $\rightsquigarrow$ **statistics** of the language could be used to break the cipher.

- (circa 1800) cipher designers tried to break this link between the plain and cipher texts.
- The most famous cipher during the 1800's is the Vigenère Cipher
  - invented in 1553 by Giovan Batista Belaso.
  - believed to be unbreakable for a number of years.

Vigenère Cipher

- The Vigenère cipher again identifies letters with the numbers $0, \ldots, 25$
- The secret key is a short sequence of letters (e.g. a word).
- Encryption involves adding the plaintext letter to a key letter.
  - With the key letters used in rotation.
- Thus if the key is SESAME, encryption works as follows,

T	H	I	S	I	S	A	T	E	S	T	M	E	S	S	A	G	E
S	E	S	A	M	E	S	E	S	A	M	E	S	E	S	A	M	E
L	L	A	S	U	W	S	X	W	S	F	Q	W	W	K	A	S	I

- This is a polyalphabetic substitution cipher

Vigenère Cipher

. . . but Vigenère is easy to break.

1. Find the length ℓ of the keyword
2. Breaking the message is the same as breaking the Shift Cipher ℓ times.

How to find the length ℓ of the keyword ?

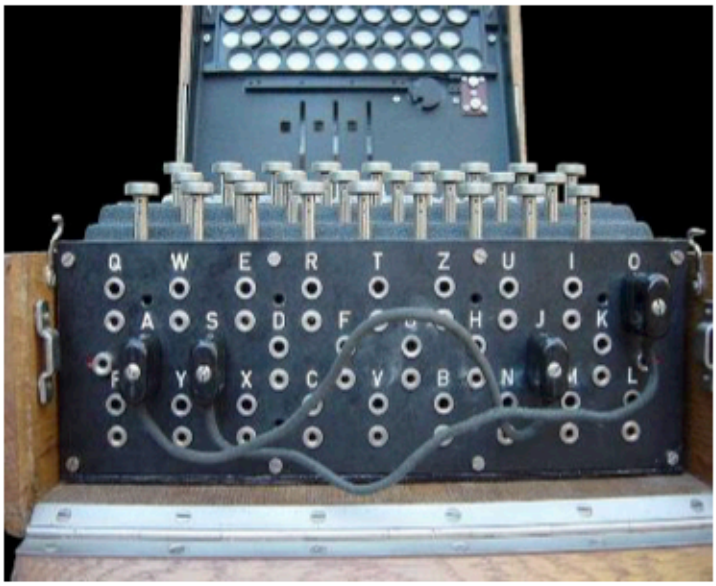
$\rightsquigarrow$ see **TME 1**

Transposition cipher

transposition cipher: method of encryption

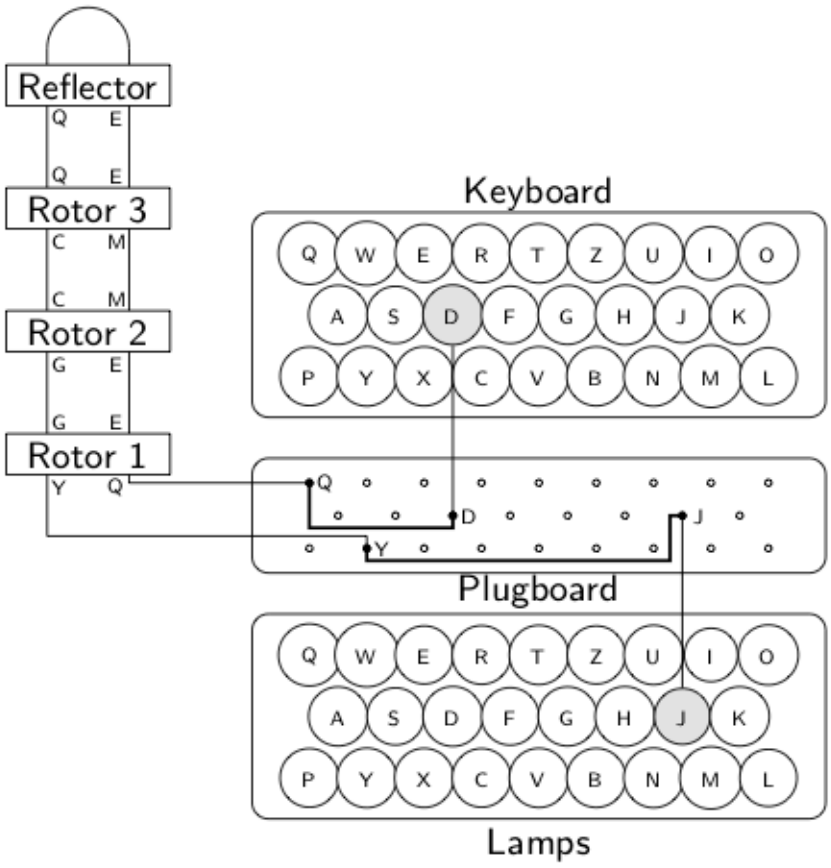
- acts on positions held by units of plaintext (which are commonly characters or groups of characters)
- they are shifted according to a **regular system**
- $\rightsquigarrow$ ciphertext constitutes a permutation of the plaintext.
- e.g. Scytale

Enigma



http://www.youtube.com/watch?v=elYw4Ve4F-I

Enigma



Encryption: Security Notions

Encryption is supposed to provide confidentiality of the data.

But what exactly does this mean?

Security goal	But ...
Recovery of secret key is infeasible	True if data is sent in the clear
Obtaining plaintext from ciphertext is infeasible	Might be able to obtain half the plaintext
etc	*etc*

So what is a **secure** encryption scheme ?

Not an easy question to answer ...

Attackers should not be able to compute any information about m.

How to formalize it ?

Attackers should not be able to compute any information about m.

Probabilistic approach

- M some random variable that takes values from $\mathcal{M}$
- K random variable distributed uniformly over $\mathcal{K}$
- $C = \mathcal{E}_K(M)$

Definition

An encryption scheme is perfectly secret if for every random variable M, every $m \in \mathcal{M}$ and every $c \in \mathcal{C}$ with $\Pr(C = c) > 0$:

$$\Pr(M = m) = \Pr(M = m | C = c)$$

$\rightsquigarrow$ C and M are independent

A perfectly secure scheme: **one-time pad**

Description

- $\ell \in \mathbb{N}$ a parameter. $\mathcal{M} = \mathcal{K} = \{0,1\}^\ell$.
- Let $\oplus$ denote component-wise XOR.
- **Vernam's cipher:** $\mathrm{Enc}(K, m) = m \oplus K$ and $\mathrm{Dec}(K, c) = c \oplus K$.

- One-time pad is perfectly secret!

$$\begin{aligned} \Pr(C = c | M = m) &= \Pr(K \oplus M = c | M = m) \\ &= \Pr(K = m \oplus c | M = m) = 2^{-\ell} \end{aligned}$$

- Each key cannot be used more than once!

$$\mathrm{Enc}(K, m_0) \oplus \mathrm{Enc}(K, m_1) = (m_0 \oplus K) \oplus (m_1 \oplus K) = m_0 \oplus m_1$$

- One time-pad is optimal in the class of perfectly secret schemes

Does encryption guarantee message integrity?

- **Idea:**
  - Alice encrypts m and sends $c = \mathrm{Enc}(K, m)$ to Bob.
  - Bob computes $\mathrm{Dec}(K, m)$, and if it "makes sense" accepts it.
- **Intuition:** only Alice knows K, so nobody else can produce a valid ciphertext.

It does not work!

Example

one-time pad.

Need a way to ensure that data arrives at destination in its original form (as sent by the sender and it is coming from an authenticated source)

Block ciphers

- Block cipher
  - deterministic algorithm
  - operates on fixed-length groups of bits, called **blocks**,
  - an unvarying transformation specified by a **symmetric key**.
- Design of block ciphers
  - based on the concept of an **iterated product cipher**
  - suggested and analyzed by Claude Shannon
  - by combining simple operations such as substitutions and permutations
- Two main techniques
  - **Feistel network** or Feistel scheme
  - **substitution-permutation networks** (SPN network)

Block ciphers - Definition

Problem: the plaintexts may be extremally long

$\rightsquigarrow$ hard to analyse security of the cipher.

Idea: Design ciphers that work on small blocks ...

two paired algorithms

- one for encryption $\mathcal{E}$
- one for decryption $\mathcal{D}$

that accept two inputs:

- an input block of size n bits
- a key of size k bits

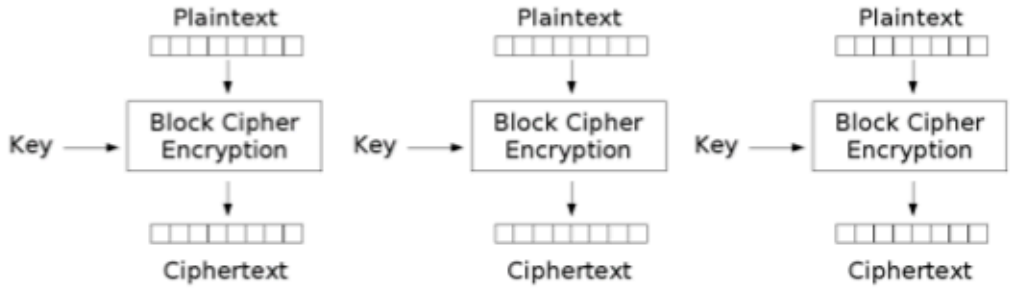
$$\begin{aligned} \mathcal{E}_K(P) &:= \mathcal{E}(K,P) : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n \\ \mathcal{E}_K^{-1}(C) &:= \mathcal{D}_K(C) = \mathcal{D}(K,C) : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n \end{aligned}$$

$$\boxed{\forall K, \forall P : \mathcal{D}_K(\mathcal{E}_K(P)) = P}$$

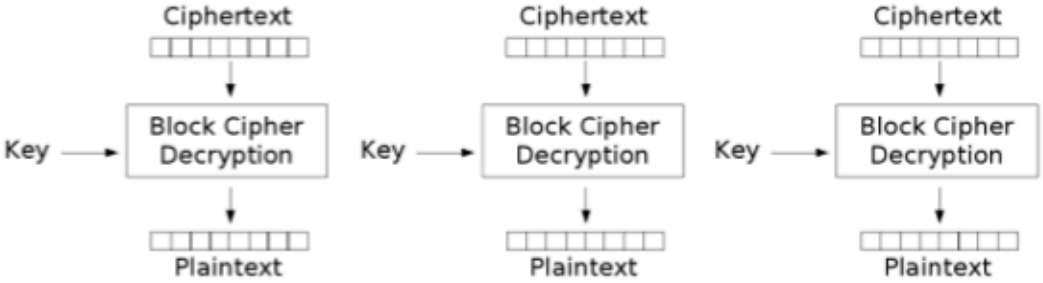
Mode of operations

- Block ciphers cannot be used **directly** for encryption.
- They are always used in some "modes of operation"
  - Electronic Codebook (ECB) mode
  - Cipher-Block Chaining (CBC) mode,
  - Output Feedback (OFB) mode,
  - Counter (CTR) mode,
  - ...

Electronic codebook (ECB)

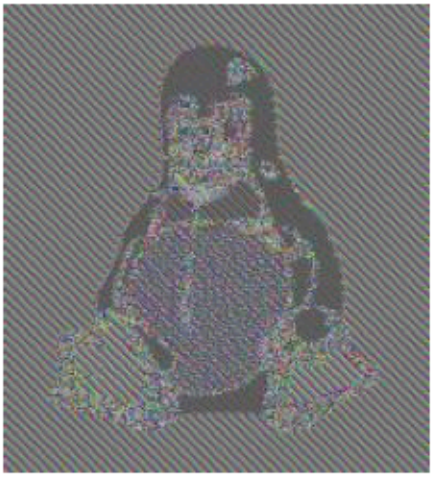


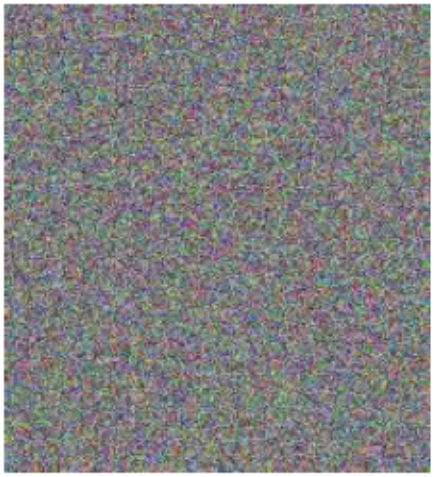
Electronic Codebook (ECB) mode encryption



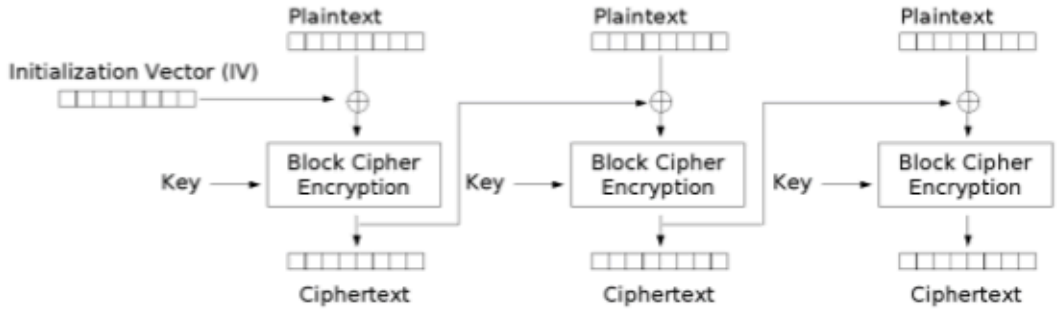
Electronic Codebook (ECB) mode decryption

Electronic codebook (ECB) should not be used !

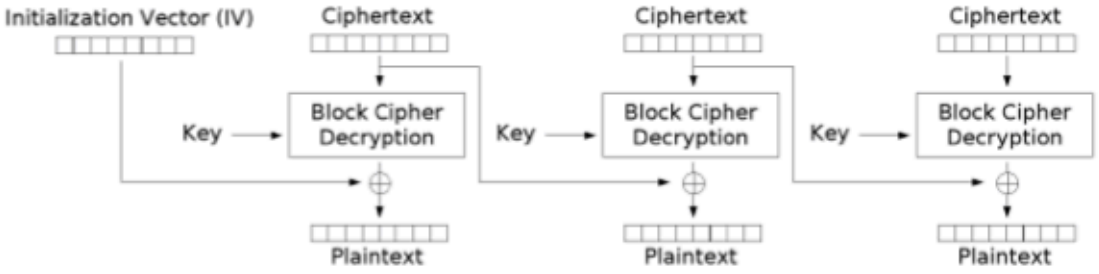




Cipher-block chaining (CBC)



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

Cipher-block chaining (CBC)

- **Error propagation?**

Error in block c_i affects only m_i and m_{i+1}
$\rightsquigarrow$ errors do not propagate

- **Can encryption be parallelized?**

No !

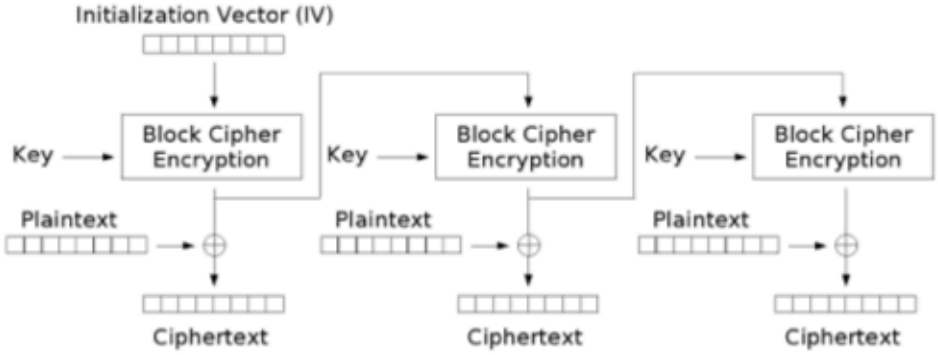
- **Can decryption be parallelized?**

Yes !

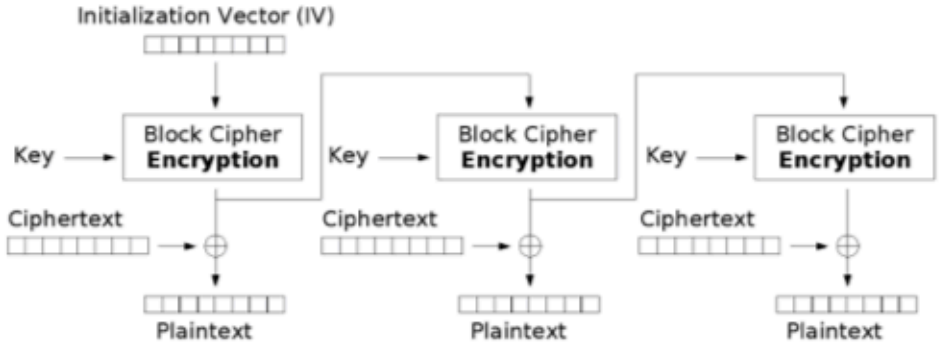
- **What if one bit of plaintext is changed (somewhere at the beginning)?**

Everything needs to be recomputed
(not so good e.g. for disc encryption)

Output feedback (OFB)



Output Feedback (OFB) mode encryption



Output Feedback (OFB) mode decryption

Output feedback (OFB)

- **Error propagation?**

Error in block c_i affects only m_i and m_{i+1}
$\rightsquigarrow$ errors do not propagate

- **Can encryption/decryption be parallelized?**

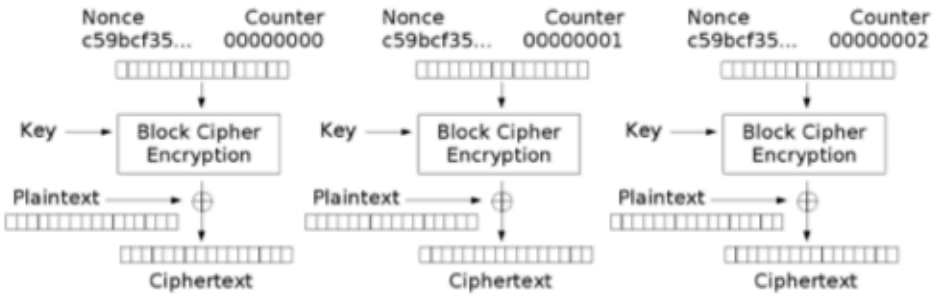
No ! . . .

. . . but we can we can use precomputation

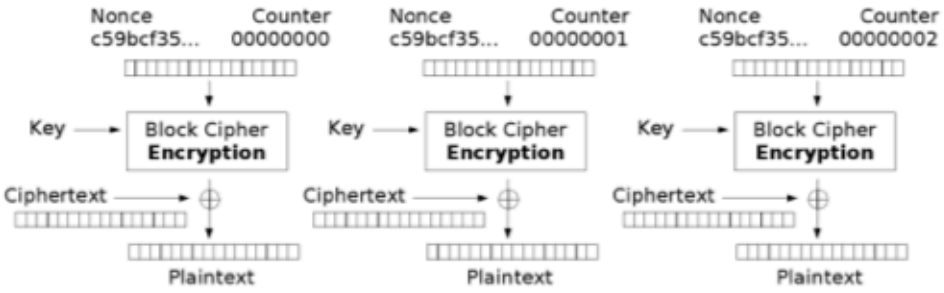
- **What if one bit of plaintext is changed (somewhere at the beginning)?**

Only one block needs to be recomputed

Counter (CTR)



Counter (CTR) mode encryption



Counter (CTR) mode decryption

Counter (CTR)

- **Error propagation?**

Error in block c_i affects only c_i and c_{i+1}
$\rightsquigarrow$ errors do not propagate

- **Can encryption/decryption be parallelized?**

Yes ! ...

... and we can we can use precomputation

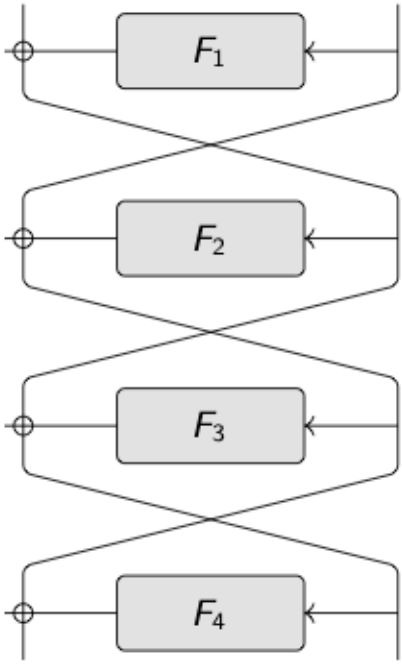
possible to decrypt one block without decrypting anything else

- **What if one bit of plaintext is changed (somewhere at the beginning)?**

Only one block needs to be recomputed

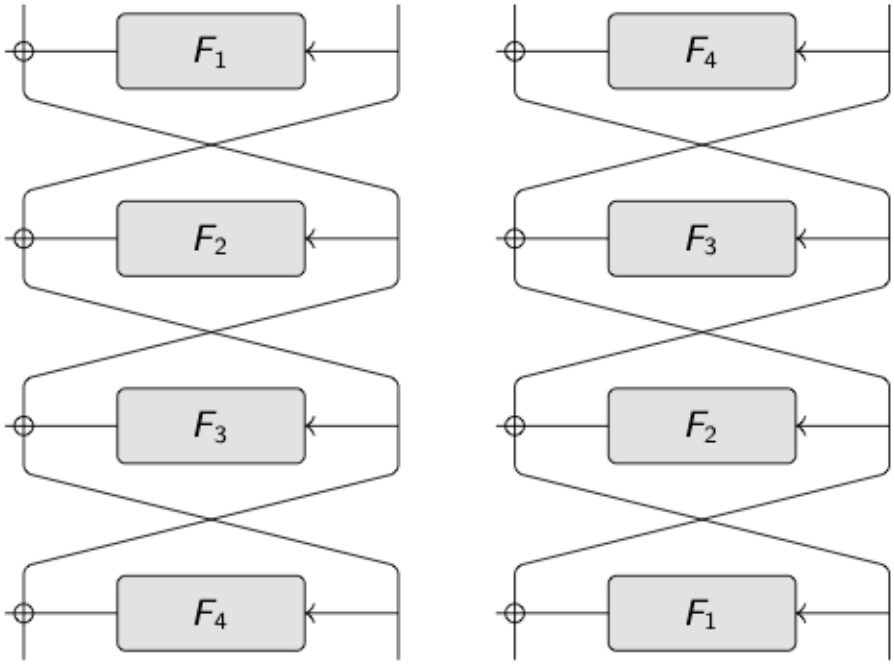
Feistel Scheme

- symmetric structure used in the construction of block ciphers
- invented by cryptographer Horst Feistel



Feistel Scheme

- invertible !

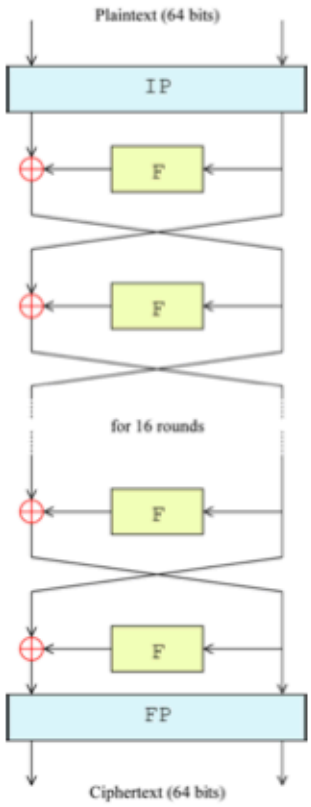


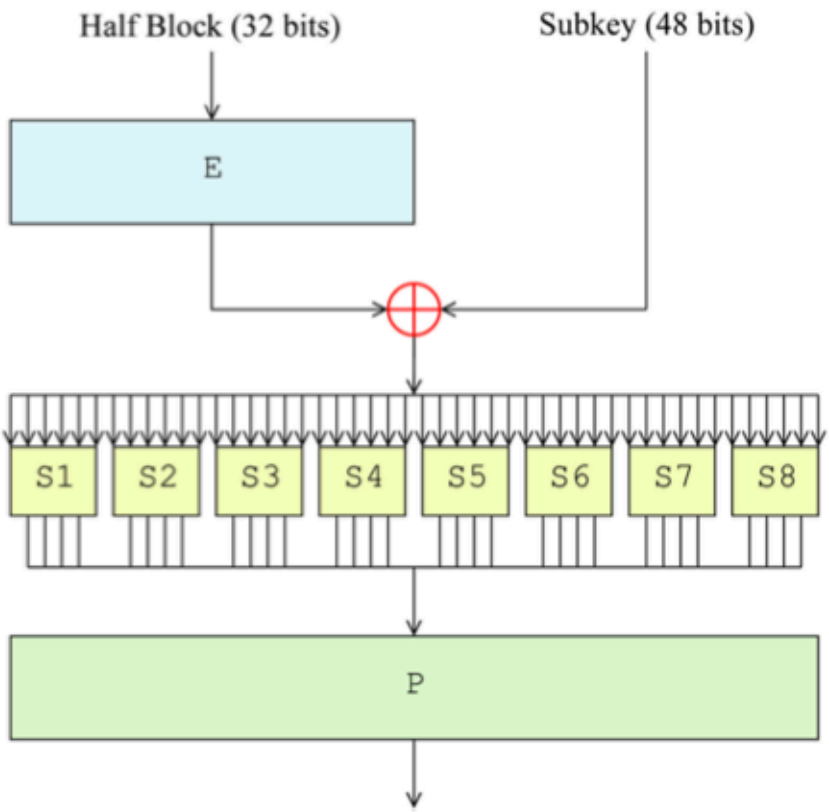
Data Encryption Standard (DES)

- First version designed by IBM in 1973-74, based on a **Lucifer cipher** (by Horst Feistel).
- **National Security Agency** (NSA) played some (unclear ...) role in the design of DES.
- made public in 1975.
- approved as a US federal standard in November 1976.

- Key length:
  - effective: 56 bits
  - **formally:** 64 bits (8 bits for checking parity).
- Block length: 64 bits

Data Encryption Standard (DES)



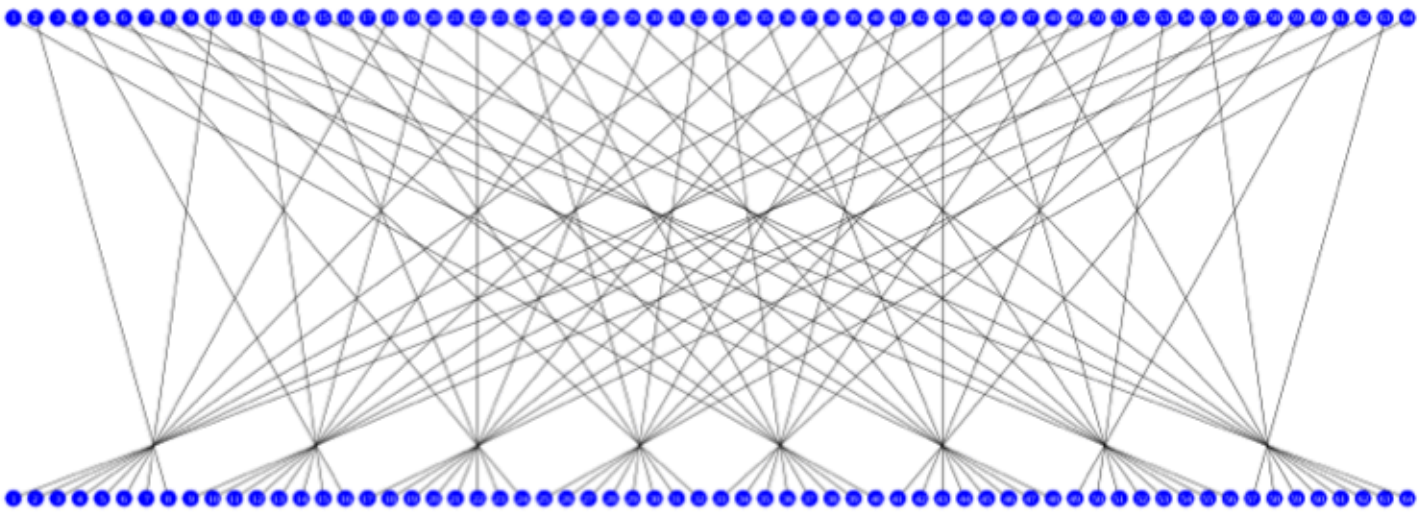


DES - Initial Permutation

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

- specifies the input permutation on a 64-bit block.
- the first bit of the output is taken from the 58th bit of the input;
- the second bit from the 50th bit,
- ...
- the last bit is taken from the 7th bit of the input.

DES - Initial Permutation

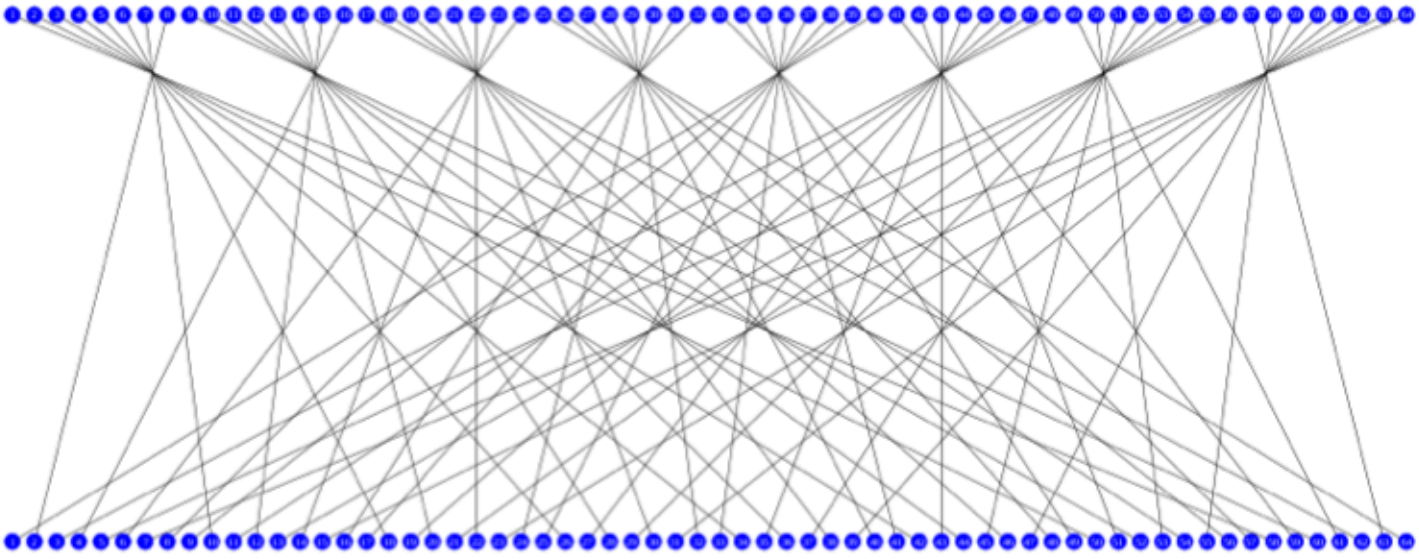


DES - Final Permutation

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

- specifies the input permutation on a 64-bit block.
- the first bit of the output is taken from the 37th bit of the input;
- the second bit from the 5th bit,
- ...
- the last bit is taken from the 25th bit of the input.

DES - Final Permutation

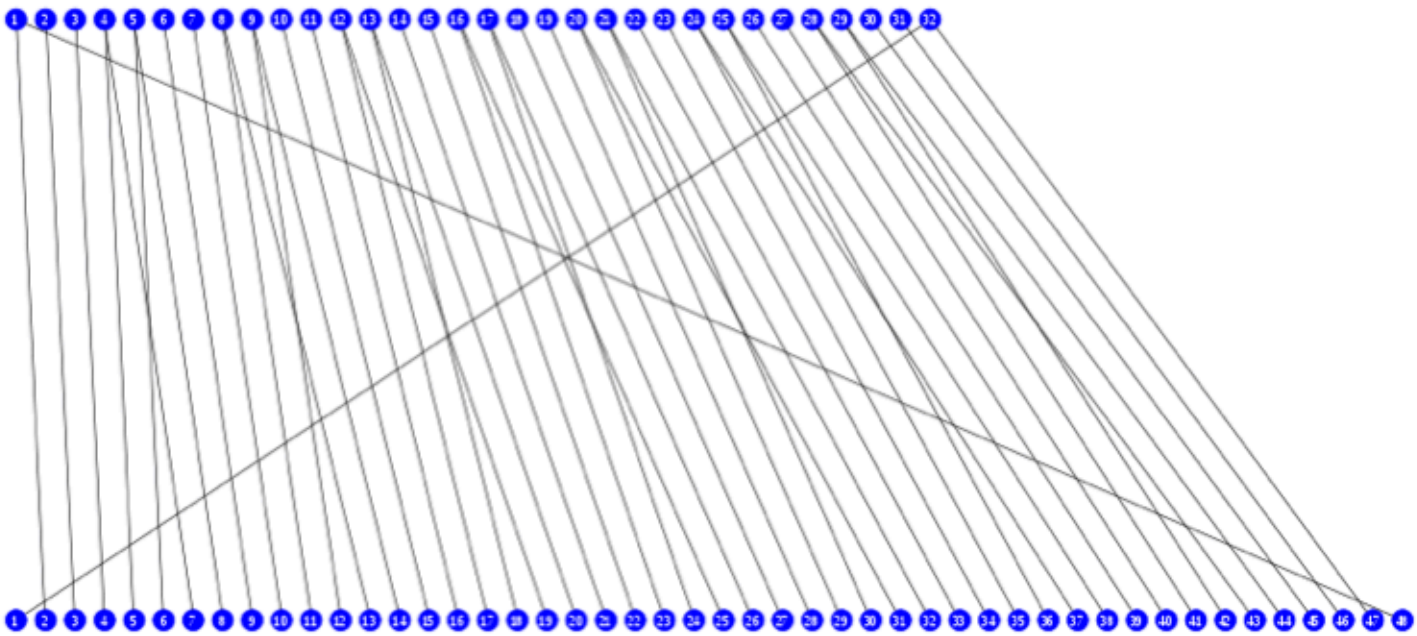


DES - Expansion Function

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

- some bits from the input are duplicated at the output
- the 32-bit half-block is expanded to 48 bits.

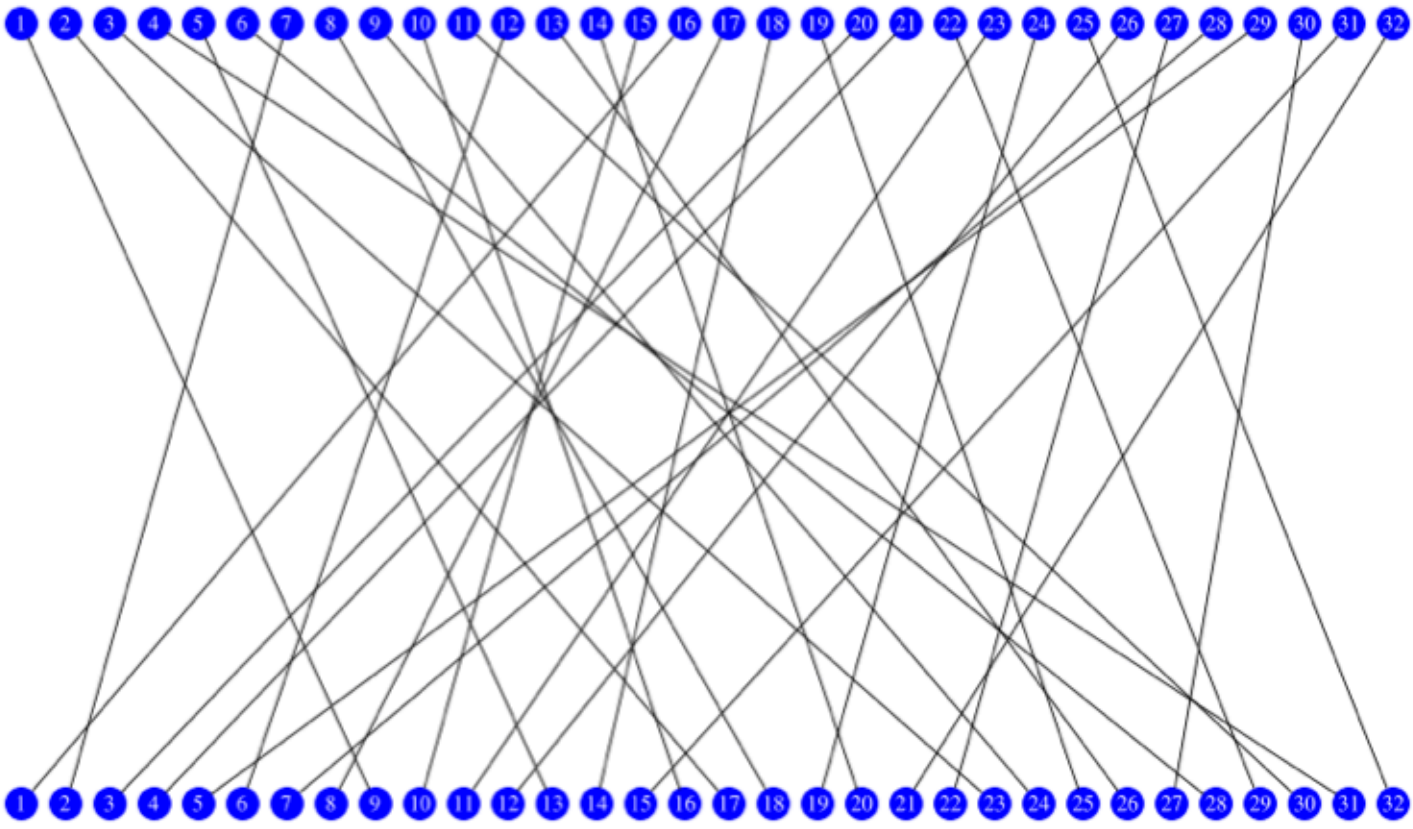
DES - Expansion Function



DES - Permutation P

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

DES - Permutation P



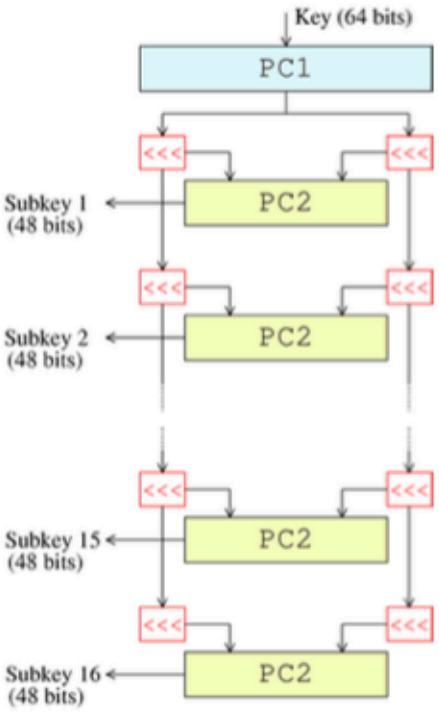
DES - S-box

- S-box: $\{0,1\}^6 \longrightarrow \{0,1\}^4$
- Example of S_1:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7
1	0	F	7	4	E	2	D	1	A	6	C	B	9	5	3	8
2	4	1	E	8	D	6	2	B	F	C	9	7	3	A	5	0
3	F	C	8	2	4	9	1	7	5	B	3	E	A	0	6	D

- $\alpha = \alpha_0\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5 \in \{0,1\}^6$
  $S_1(\alpha) \rightsquigarrow$ entry indexed by row $\alpha_r = \alpha_0\alpha_5$ and column $\alpha_c = \alpha_1\alpha_2\alpha_3\alpha_4$.
  Example : $S_1(011011) = 0101$ since $5 = 0101$ is at the intersection of row $1 = 01$ and column $\mathtt{D} = 1101$.

DES Key Schedule



DES - Permuted Choice

- 56 bits of the key are selected by Permuted Choice 1 (PC-1)

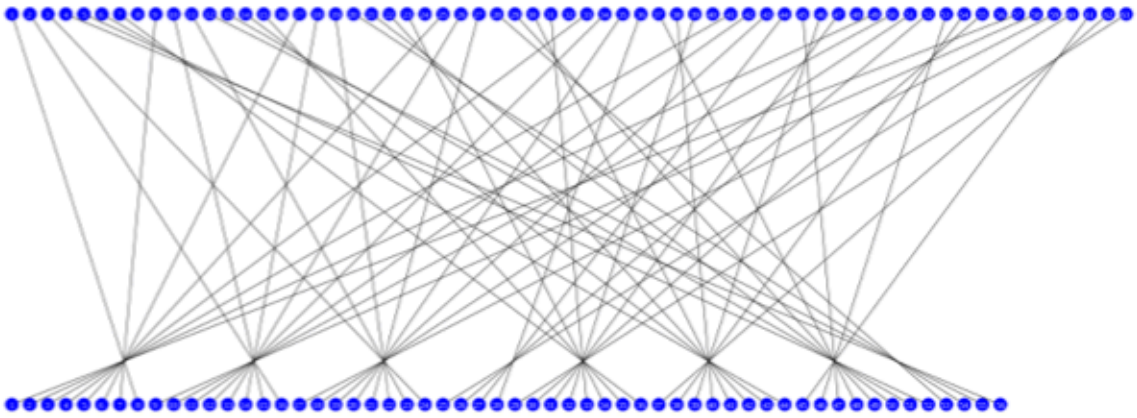
57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36

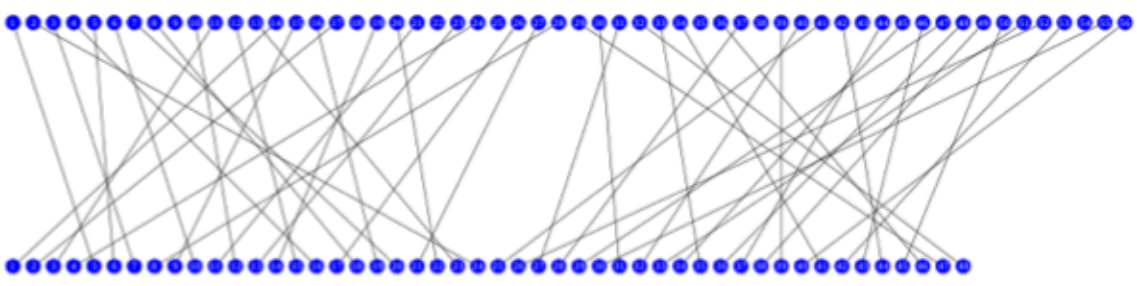
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

- 48 subkey bits are selected by Permuted Choice 2 (PC-2)
(24 bits from the left half, and 24 from the right)

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

DES - Permuted Choice





Introduction à la Sécurité

Cryptologie symétrique 2/2

Damien Vergnaud

Sorbonne Université
Institut Universitaire de France

17 janvier 2019

Contents

AES Origins

- a **replacement** for DES was needed
  - theoretical attacks that can break it
  - exhaustive key search attacks
- can use Triple-DES – but slow, has small blocks

- US NIST issued call for ciphers in 1997
  - **Block size:** 128 bits (possibly 64, 256, ...)
  - **Key size:** 128, 192, 256 bits

- 15 candidates accepted in June 98
- 5 were shortlisted in August 99
- Rijndael was selected as the AES in October 2000
- issued as FIPS PUB 197 standard in November 2001

AES – Rijndael

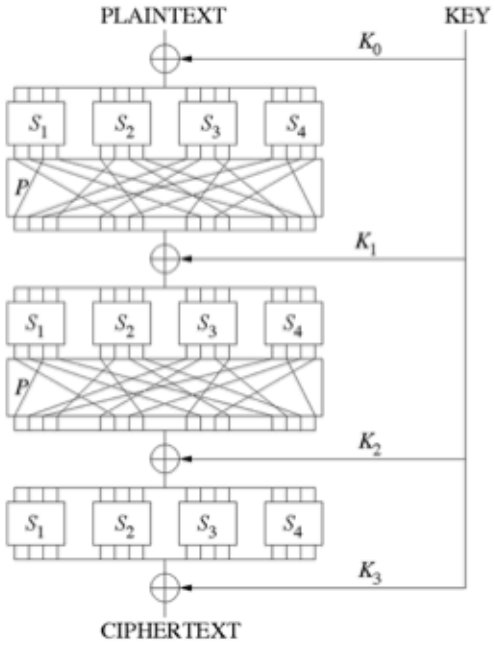
- designed by Rijmen-Daemen in Belgium
- has 128/192/256 bit keys, 128 bit data (for AES)

- a **substitution-permutation network** (SPN) (rather than Feistel cipher)
  - processes data as block of 4 columns of 4 bytes
  - operates on entire data block in every round

- designed to have:
  - resistance against known attacks
  - speed and code compactness on many CPUs
  - design simplicity

Substitution-Permutation Network

- to provide Confusion and Diffusion (Shannon)
- **Substitution:** S-boxes substitute a small block of input bits into output bits
  - invertible, non-linear
  - changing one input bit $\rightsquigarrow$ change about half of the output bits
- **Permutation:** P-boxes permute bits for the next-round S-box inputs
  - output bits of an S-box distributed to as many S-box inputs as possible.
- **Key:** in each round using group operation ($\oplus$)
- one S-box/P-box produces a *limited* amount of confusion/diffusion
- enough **rounds** $\rightsquigarrow$ every input bit is diffused across every output bit



AES Description

- AES with 128-bit key (see refs. for 192-bit and 256-bit keys)
- **Data block:** 128 bits $\rightsquigarrow$ 16 bytes in a 4×4 matrix

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

- A byte $b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0$ is represented by a polynomial

$$b_7 x^7 + b_6 x^6 + b_5 x^5 + b_4 x^4 + b_3 x^3 + b_2 x^2 + b_1 x^1 + b_0$$

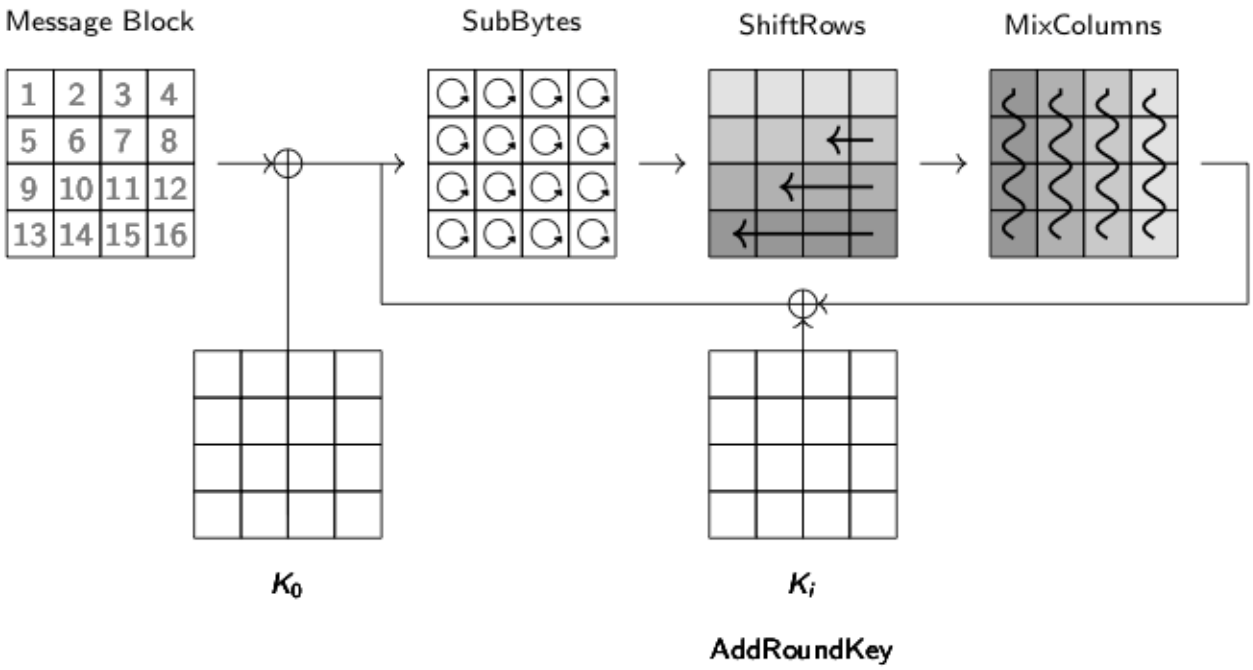
  with $b_i \in \{0, 1\} = \mathbb{F}_2$.
- **Example:** 5A = 01011010

$$\rightsquigarrow x^6 + x^4 + x^3 + x^1$$

- Bytes are identified with elements of the finite field $\mathbb{F}_{256} = \mathbb{F}_2[x]/(m(x))$ with

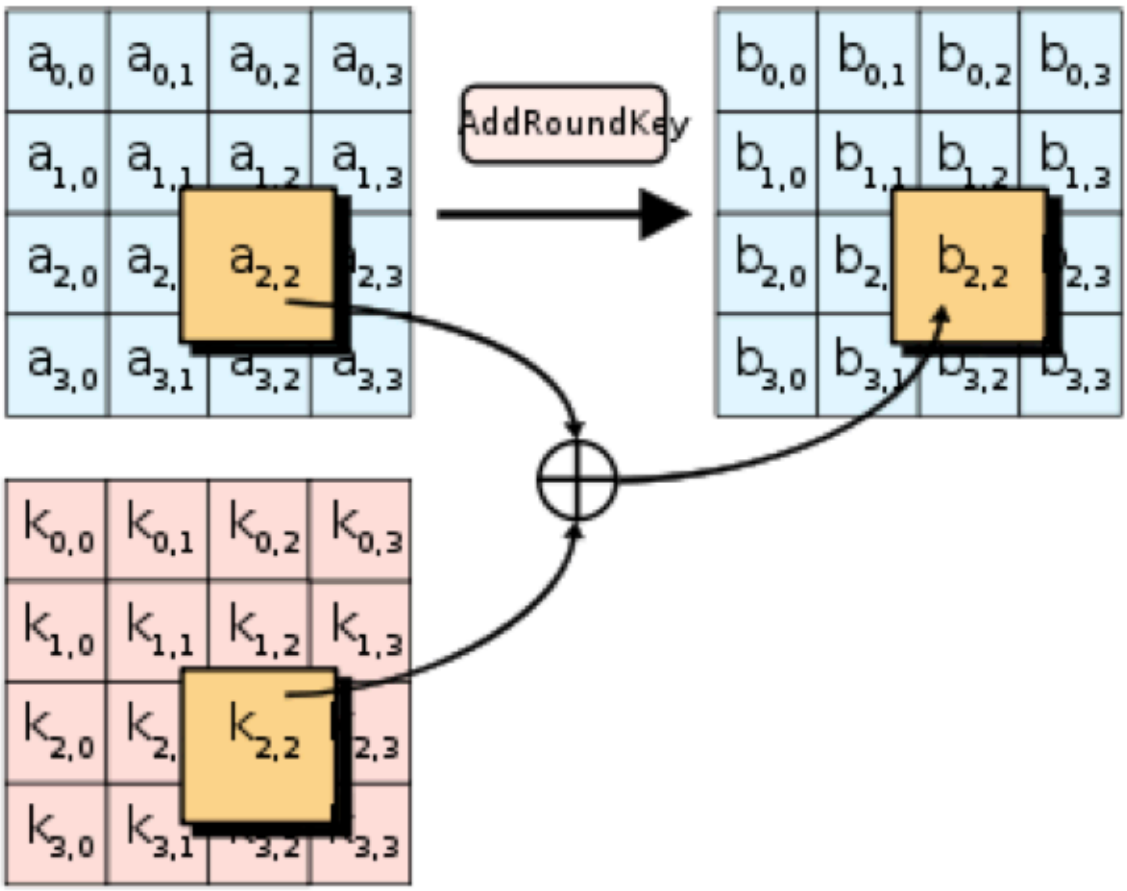
$$m(x) = x^8 + x^4 + x^3 + x + 1$$

AES Structure

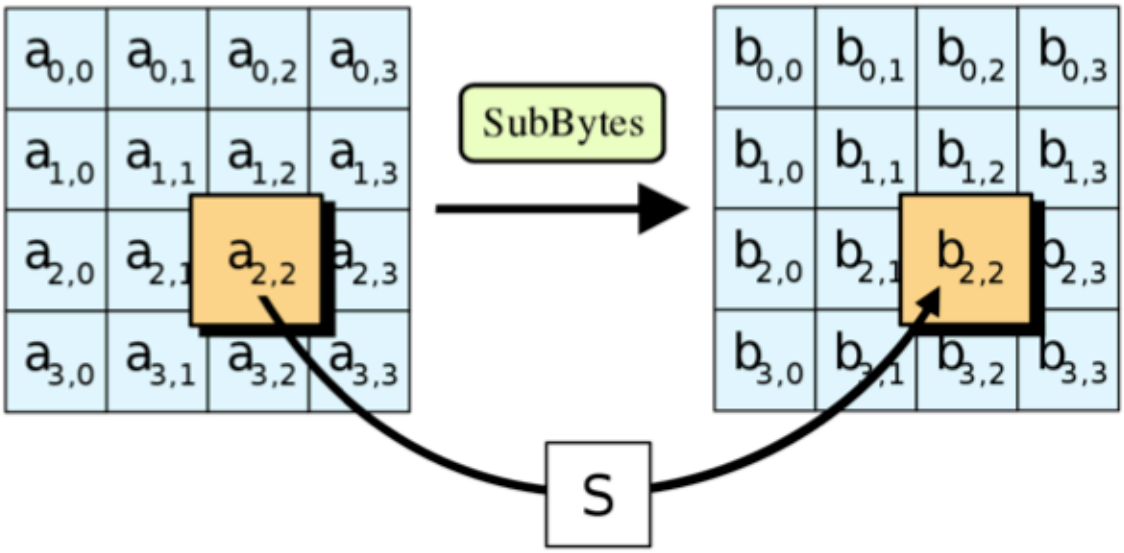


- no MixColumns in the last round

AddRoundKey



SubBytes



SubBytes

- S-box defined algebraically over $\mathbb{F}_{256}$
- First invert the byte (interpreted as an alement of $\mathbb{F}_{256}$:

$$a \longmapsto \begin{cases} a^{-1} & \text{if } a \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

- Then apply affine transformation:

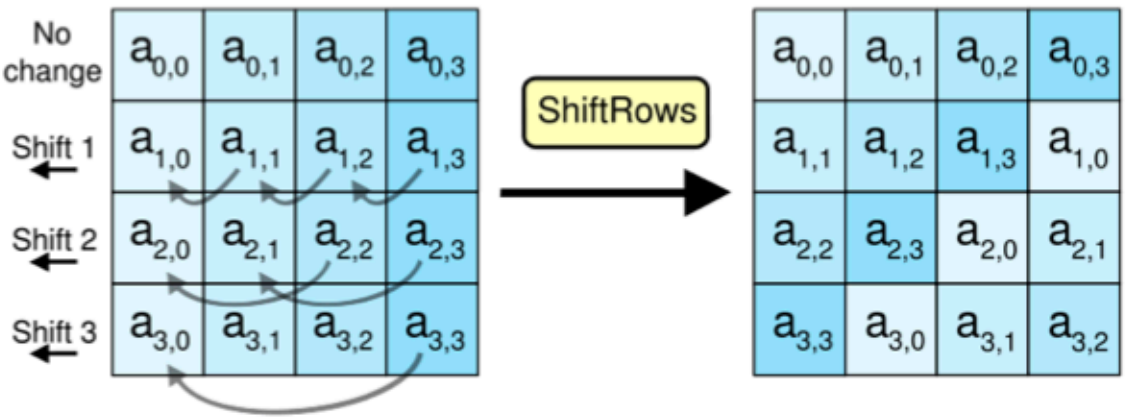
$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

SubBytes

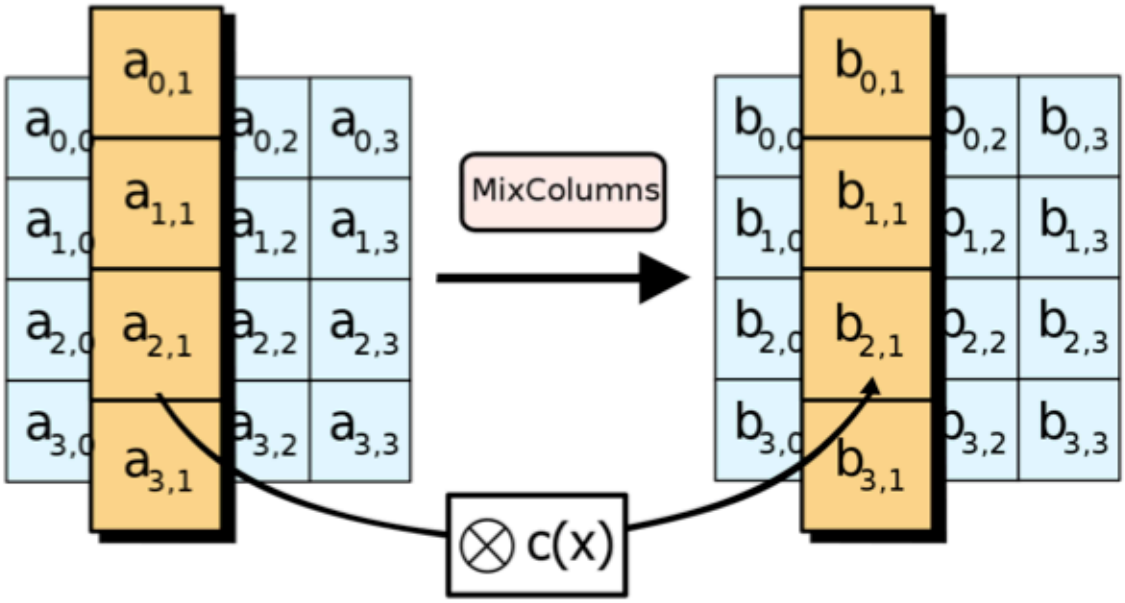
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

- the column is determined by the least significant nibble,
- the row is determined by the most significant nibble.
- **Example:** $S(\texttt{9A}) = \texttt{B8}$

ShiftRows



MixColumns



MixColumns

- in the ring $A = \mathbb{F}_{256}[X]/(X^4 + 1)$.
- using $a(X)$ in A:

$$\begin{aligned} a(X) &= \{03\}X^3 + \{01\}X^2 + \{01\}X + \{02\} \\ &= (x+1)X^3 + X^2 + X + x \end{aligned}$$

- given a column of four bytes (b_0, b_1, b_2, b_3), consider the polynomial

$$b_0 + b_1 X + b_2 X^2 + b_3 X^3 \in A$$

- the new column is

$$a(X) \cdot b(X) \in A$$

MixColumns

$$\begin{aligned} a(X) &= \{03\}X^3 + \{01\}X^2 + \{01\}X + \{02\} \\ &= (x+1)X^3 + X^2 + X + x \end{aligned}$$

$$b(X) = b_0 + b_1 X + b_2 X^2 + b_3 X^3$$

Step 1: Polynomial multiplication

$$\begin{aligned} a(x) \cdot b(x) = c(x) &= (a_3X^3 + a_2X^2 + a_1X + a_0) \cdot (b_3X^3 + b_2X^2 + b_1X + b_0) \\ &= c_6X^6 + c_5X^5 + c_4X^4 + c_3X^3 + c_2X^2 + c_1X + c_0 \end{aligned}$$

where:

$$\begin{aligned} c_0 &= a_0 \cdot b_0 \\ c_1 &= a_1 \cdot b_0 \oplus a_0 \cdot b_1 \\ c_2 &= a_2 \cdot b_0 \oplus a_1 \cdot b_1 \oplus a_0 \cdot b_2 \\ c_3 &= a_3 \cdot b_0 \oplus a_2 \cdot b_1 \oplus a_1 \cdot b_2 \oplus a_0 \cdot b_3 \end{aligned}$$

$$\begin{aligned} c_4 &= a_3 \cdot b_1 \oplus a_2 \cdot b_2 \oplus a_1 \cdot b_3 \\ c_5 &= a_3 \cdot b_2 \oplus a_2 \cdot b_3 \\ c_6 &= a_3 \cdot b_3 \end{aligned}$$

MixColumns

Step 2: Modular reduction

$$\begin{cases} X^6 \bmod (X^4+1) = -X^2 = X^2 \text{ over } \mathrm{GF}(2^8) \\ X^5 \bmod (X^4+1) = -X = X \text{ over } \mathrm{GF}(2^8) \\ X^4 \bmod (X^4+1) = -1 = 1 \text{ over } \mathrm{GF}(2^8) \end{cases}$$

$$\begin{aligned} & a(X) \cdot b(X) = c(X) \bmod (X^4+1) \\ &= (c_6X^6 + c_5X^5 + c_4X^4 + c_3X^3 + c_2X^2 + c_1X + c_0) \bmod (X^4+1) \\ &= c_6X^2 + c_5x + c_4 + c_3X^3 + c_2X^2 + c_1X + c_0 \\ &= c_3X^3 + (c_2 \oplus c_6)X^2 + (c_1 \oplus c_5)X + c_0 \oplus c_4 \\ &= d_3X^3 + d_2X^2 + d_1X + d_0 \end{aligned}$$

where

$$d_0 = c_0 \oplus c_4, \quad d_1 = c_1 \oplus c_5, \quad d_2 = c_2 \oplus c_6, \quad d_3 = c_3$$

MixColumns

Step 3: Matrix representation

$$
\begin{aligned}
d_0 &= a_0 \cdot b_0 \oplus a_3 \cdot b_1 \oplus a_2 \cdot b_2 \oplus a_1 \cdot b_3 \\
d_1 &= a_1 \cdot b_0 \oplus a_0 \cdot b_1 \oplus a_3 \cdot b_2 \oplus a_2 \cdot b_3 \\
d_2 &= a_2 \cdot b_0 \oplus a_1 \cdot b_1 \oplus a_0 \cdot b_2 \oplus a_3 \cdot b_3 \\
d_3 &= a_3 \cdot b_0 \oplus a_2 \cdot b_1 \oplus a_1 \cdot b_2 \oplus a_0 \cdot b_3
\end{aligned}
$$

$$
\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} \{02\} & \{03\} & \{01\} & \{01\} \\ \{01\} & \{02\} & \{03\} & \{01\} \\ \{01\} & \{01\} & \{02\} & \{03\} \\ \{03\} & \{01\} & \{01\} & \{02\} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}
$$

- **addition** is an XOR operation
- **multiplication** is a (complicated) multiplication in $\mathbb{F}_{256}$

MixColumns

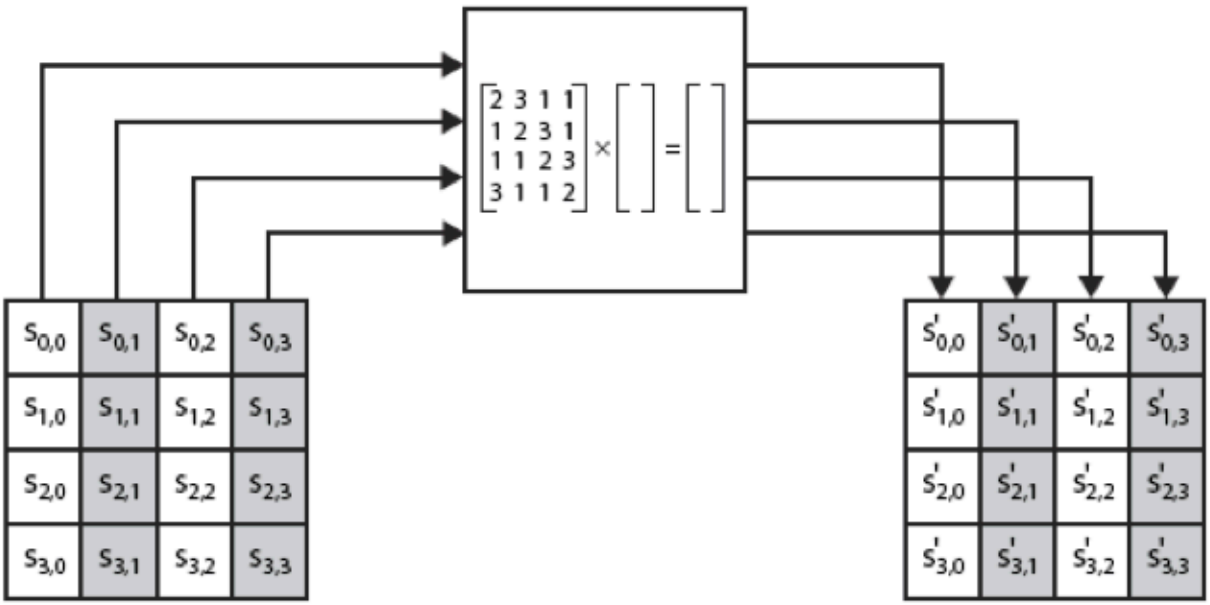
Step 3: Matrix representation

$$
\begin{aligned}
d_0 &= a_0 \cdot b_0 \oplus a_3 \cdot b_1 \oplus a_2 \cdot b_2 \oplus a_1 \cdot b_3 \\
d_1 &= a_1 \cdot b_0 \oplus a_0 \cdot b_1 \oplus a_3 \cdot b_2 \oplus a_2 \cdot b_3 \\
d_2 &= a_2 \cdot b_0 \oplus a_1 \cdot b_1 \oplus a_0 \cdot b_2 \oplus a_3 \cdot b_3 \\
d_3 &= a_3 \cdot b_0 \oplus a_2 \cdot b_1 \oplus a_1 \cdot b_2 \oplus a_0 \cdot b_3
\end{aligned}
$$

$$
\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} x & (x+1) & 1 & 1 \\ 1 & x & (x+1) & 1 \\ 1 & 1 & x & (x+1) \\ (x+1) & 1 & 1 & x \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}
$$

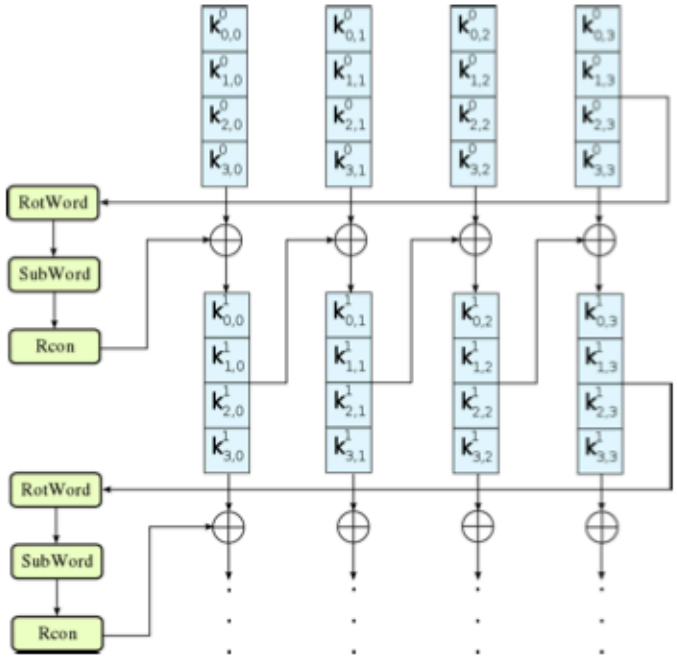
- **addition** is an XOR operation
- **multiplication** is a (complicated) multiplication in $\mathbb{F}_{256}$

MixColumns



AES Key Schedule

- takes 128-bit (16-byte) key
- expands into array of 44/52/60 32-bit words



Does encryption guarantee message integrity?

- **Idea:**
  - Alice encrypts m and sends $c = \mathrm{Enc}(K, m)$ to Bob.
  - Bob computes $\mathrm{Dec}(K, m)$, and if it "makes sense" accepts it.
- **Intuition:** only Alice knows K, so nobody else can produce a valid ciphertext.

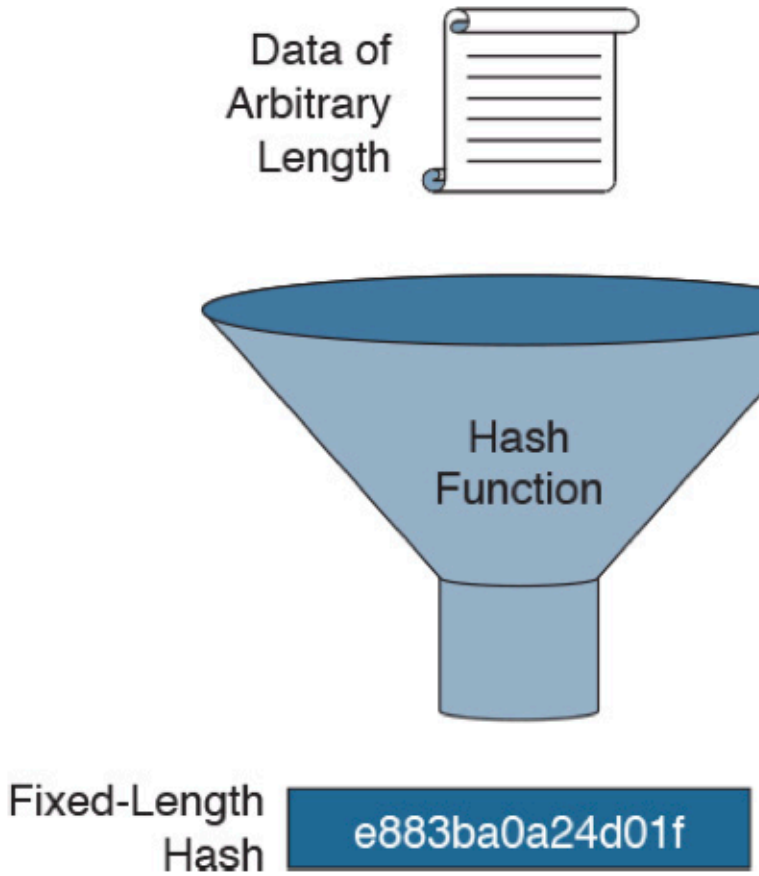
It does not work!

Example

one-time pad.

Need a way to ensure that data arrives at destination in its original form (as sent by the sender and it is coming from an authenticated source)

Hash Functions



Hash Functions

- map a message of an **arbitrary** length to a **fixed** length output
- **output:** fingerprint or message digest
- What is an example of hash functions?
  - **Question:** Give a hash function that maps Strings to integers in $[0, 2^{32} - 1]$
- additional security requirements $\rightsquigarrow$ cryptographic hash functions

Security Requirements for Cryptographic Hash Functions

Given a function $\mathcal{H} : X \longrightarrow Y$, then we say that h is:

- pre-image resistant (one-way):
  if given $y \in Y$ it is computationally infeasible to find a value $x \in X$ s.t. $\mathcal{H}(x) = y$
- second pre-image resistant (weak collision resistant):
  if given $x \in X$ it is computationally infeasible to find a value $x' \in X$, s.t. $x' \neq x$ and $\mathcal{H}(x') = \mathcal{H}(x)$
- collision resistant (strong collision resistant):
  if it is computationally infeasible to find two distinct values $x', x \in X$, s.t. $x' \neq x$ and $\mathcal{H}(x') = \mathcal{H}(x)$

Generic Attack against Pre-image resistance

Input: $y \in \{0,1\}^n$, $m \in \mathbb{N}$ with $m > n$
Output: $x \in \{0,1\}^m$ s.t. $y = \mathcal{H}(x)$

```
while TRUE do
    x ←R {0,1}^m
    if H(x) = y then
        return x
    end if
end while
```

- Time Complexity: $O(2^n)$ (random $\mathcal{H}$)
- Space Complexity: $O(1)$

Generic Attack against Second Pre-image resistance

Input: $x \in \{0,1\}^m$
Output: $x' \in \{0,1\}^m$ s.t. $\mathcal{H}(x') = \mathcal{H}(x)$

```
y ← H(x)
while TRUE do
    x' ←R {0,1}^m
    if H(x') = y then
        return x'
    end if
end while
```

- Time Complexity: $O(2^n)$ (random $\mathcal{H}$)
- Space Complexity: $O(1)$

Generic Attack against Second Pre-image resistance

Input: $m \in \mathbb{N}$ with $m > n$
Output: $x, x' \in \{0,1\}^m$ s.t. $\mathcal{H}(x) = \mathcal{H}(x')$ and $x \neq x'$

```
Υ ← ∅                                              ▷ hash table
while TRUE do
    x_i ←R {0,1}^m
    y_i ← H(x_i)
    j ← LookUp(y_i, Υ)
    if j ≠ −1 then
        return (x_i, x_j)                          ▷ H(x_i) = H(x_j)
    end if
    AddElement(Υ, (x_i, y_i))                      ▷ sorted using the second coordinate
end while
```

Birthday Paradox: (see **TD 1**)

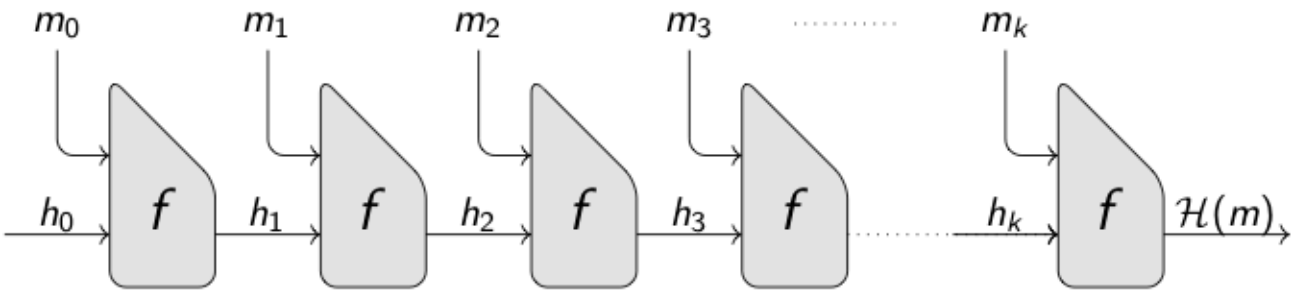
- Time Complexity: $O(2^{n/2})$ (random $\mathcal{H}$)
- Space Complexity: $O(2^{n/2})$

Hash functions in Security

- Digital signatures
- Random number generation
- Key updates and derivations
- One way functions
- MAC
- Detect malware in code
- User authentication (storing passwords)
- ...

Merkle-Damgaard

- compression function $f : \{0,1\}^{n+\ell} \longrightarrow \{0,1\}^n$
- **How to hash** $m = (m_0, \ldots, m_k) \in (\{0,1\}^\ell)^{(k+1)}$**?**



- h_0 initial value (intialization vector)
- **Theorem:** f collision-resistant $\Rightarrow$ $\mathcal{H}$ collision resistant (with appropriate padding) (see **TD 2**)

MD5

- 128-bit hashes
- designed by Ronald Rivest in 1991
- "MD" stands for "Message Digest"
  - `MD5`("The quick brown fox jumps over the lazy dog") = `9e107d9d372bb6826bd81d3542a419d6`
  - `MD5`("The quick brown fox jumps over the lazy dog.") = `e4d909c290d0fb1ca068ffaddf22cbd0`
- cryptographically broken (since 2004!)

- input message broken up into chunks of 512-bit blocks
- (message padded $\rightsquigarrow$ length is a multiple of 512)

MD5 (for reference only)

Input: $m \in \{0,1\}^*$, $|m| < 2^{64} - 1$
Output: $h \in \{0,1\}^{128}$, $h = \text{MD5}(m)$

$r[0..15] \leftarrow \{7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22\}$ ▷ initialisation
$r[16..31] \leftarrow \{5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20\}$
$r[32..47] \leftarrow \{4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23\}$
$r[48..63] \leftarrow \{6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21\}$
for i de 0 à 63 **do**
  $k[i] \leftarrow \lfloor (|sin(i+1)| \cdot 2^{32}) \rfloor$
end for
$h^0 \leftarrow 67452301$; $h^1 \leftarrow$ EFCDAB89; $h^2 \leftarrow$ 98BADCFE; $h^3 \leftarrow 10325476$
$i = |m| \bmod \ell$
$(m_0, \ldots, m_k) \leftarrow \mathcal{R}(m) = m \| 10^{\ell - i - 65} \| \tau_m$ ▷ with $|m_i| = 512$
...

MD5 (for reference only)

... ▷ main loop
for j from 1 to k **do**
  $(w_0, \ldots, w_{15}) \leftarrow m_k$ ▷ with $|w_0| = 32, \ldots, |w_{15}| = 32$
  $a \leftarrow h^0$; $b \leftarrow h^1$; $c \leftarrow h^2$; $d \leftarrow h^3$
  for i from 0 to 63 **do**
    if $0 \leq i \leq 15$ **then**
      $f \leftarrow (b \wedge c) \vee ((\neg b) \wedge d)$; $g \leftarrow i$
    else if $16 \leq i \leq 31$ **then**
      $f \leftarrow (d \wedge b) \vee ((\neg d) \wedge c)$; $g \leftarrow (5i + 1) \bmod 16$
    else if $32 \leq i \leq 47$ **then**
      $f \leftarrow b \oplus c \oplus d$; $g \leftarrow (3i + 5) \bmod 16$
    else if $48 \leq i \leq 63$ **then**
      $f \leftarrow c \oplus (b \vee (\neg d)$; $g \leftarrow (7i) \bmod 16$
    end if
    $(a, b, c, d) \leftarrow (d, ((a + f + k[i] + w[g]) \lll r[i]) + b, b, c)$
  end for
  $h^0 \leftarrow h^0 + a$; $h^1 \leftarrow h^1 + b$; $h^2 \leftarrow h^2 + c$; $h^3 \leftarrow h^3 + d$
end for
return $(h^0 \| h^1 \| h^2 \| h^3)$

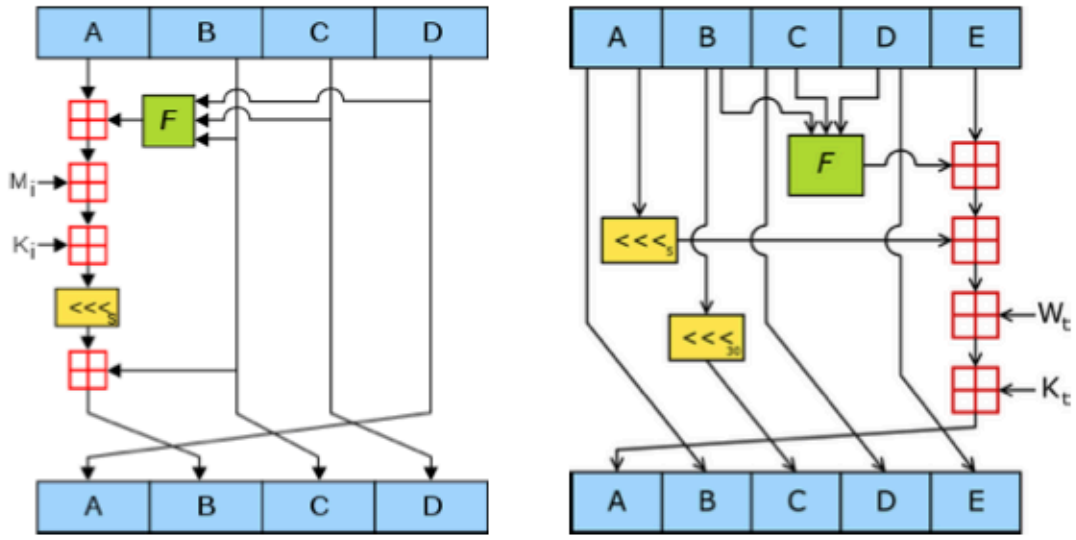
Collisions in MD5

- **Birthday attack complexity:** 2^{64}
  - small enough to brute force collision search
- **1996,** collisions on the compression function
- **2004,** collisions
- **2007,** chosen-prefix collisions
- **2008,** rogue SSL certificates generated
- **2012,** MD5 collisions used in cyberwarfare
  - Flame malware uses an MD5 prefix collision to fake a Microsoft digital code signature

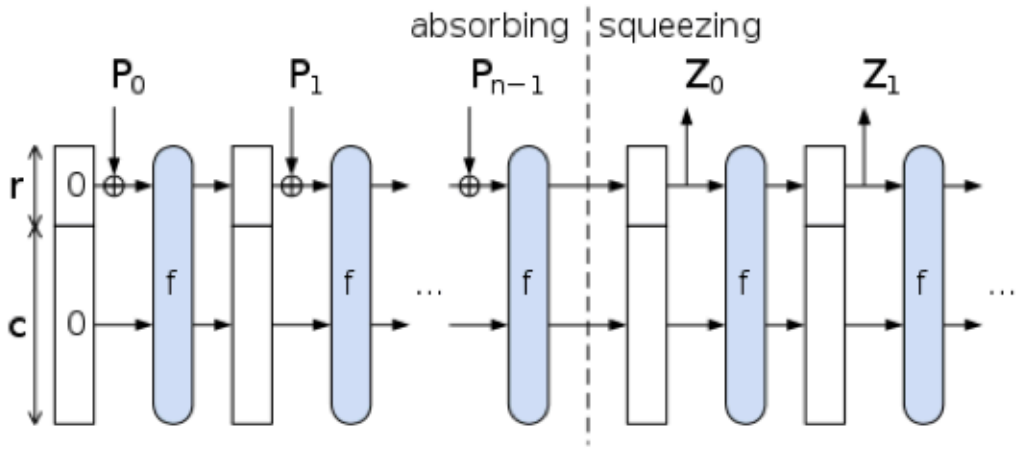
SHA Family - Secure Hash Algorithm

- **SHA-0**: (1993). 160 bit digest
  - unpublished weaknesses in this algorithm
  - **1998,** collision attack with complexity 2^{61}
- **SHA-1**: (1995). 160 bit digest
  - **2005,** collision attack with claimed complexity of 2^{69}
  - **2010,** SHA1 was no longer supported
  - **2017,** first collisions found
- **SHA-2**: (2001). digest of length 224, 256, 384, 512 (+2 truncated versions)
  - No collision attacks on SHA-2 as yet
- **SHA-3**: (2015). Also known as Keccak
  - (Bertoni, Daemen, Peeters and Van Assche)

MD5 vs SHA-1

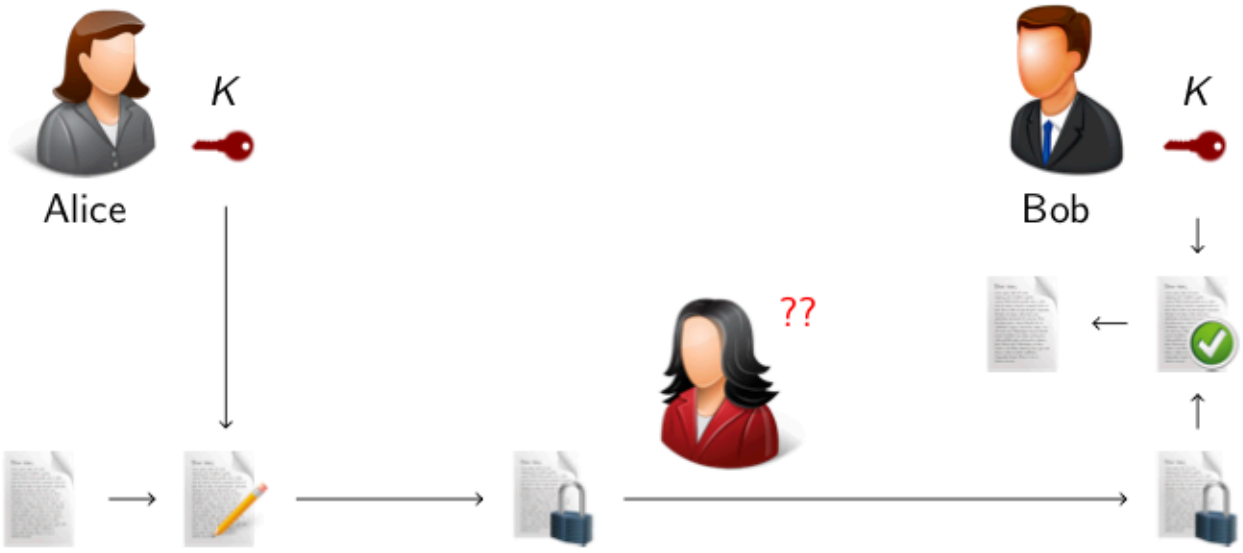


SHA-3



Message Authentication Codes

Symmetric authentication: Alice and Bob share a "key" K



- Bob can use the same method to send messages to Alice.
  $\rightsquigarrow$ **symmetric setting**
- How did Alice and Bob establish K?

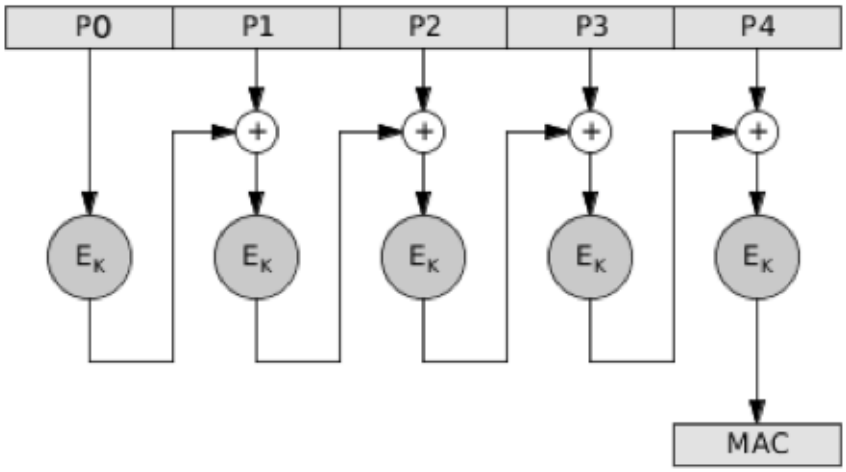
Security Requirement for MAC

- resist the **Existential Forgery under Chosen Plaintext Attack**
  - challenger chooses a random key K
  - adversary chooses a number of messages $m_1, m_2, \ldots, m_\ell$ and obtains $\tau_i = \mathrm{MAC}(K, m_i)$ for $1 \leq i \leq \ell$
  - adversary outputs $m^\star$ and $\tau^\star$
  - adversary wins if $\forall i m^\star \neq m_i$ and $\tau^\star = \mathrm{MAC}(K, m^\star)$

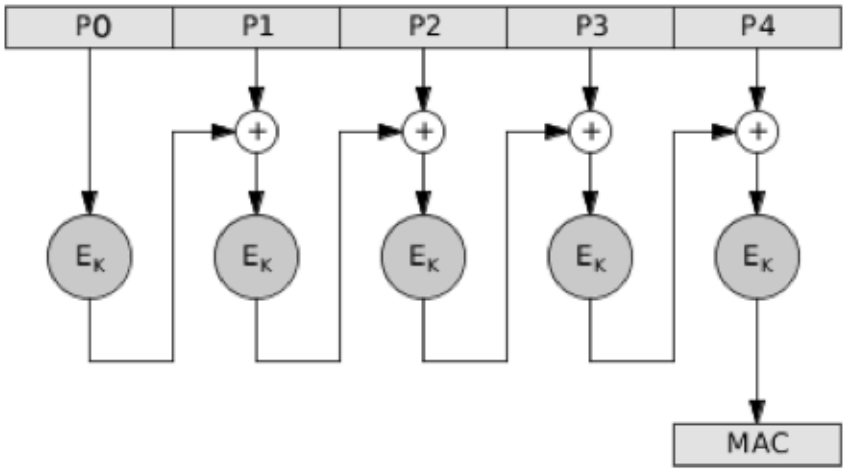
- Adversary cannot create the MAC for a message for which it has not seen a MAC

CBC-MAC

- E a block cipher (DES, AES, ...) on n-bit blocks
- produces a n-bit MAC



Forgery on CBC-MAC



- Message $m = (m_1, \ldots, m_\ell)$ with MAC τ
- Message $m' = (m'_1, \ldots, m'_k)$ with MAC τ'
- Message

$$m'' = (m_1, \ldots, m_\ell, m'_1 \oplus \tau, \ldots, m'_k)$$

has MAC τ'!

Fixing CBC-MAC

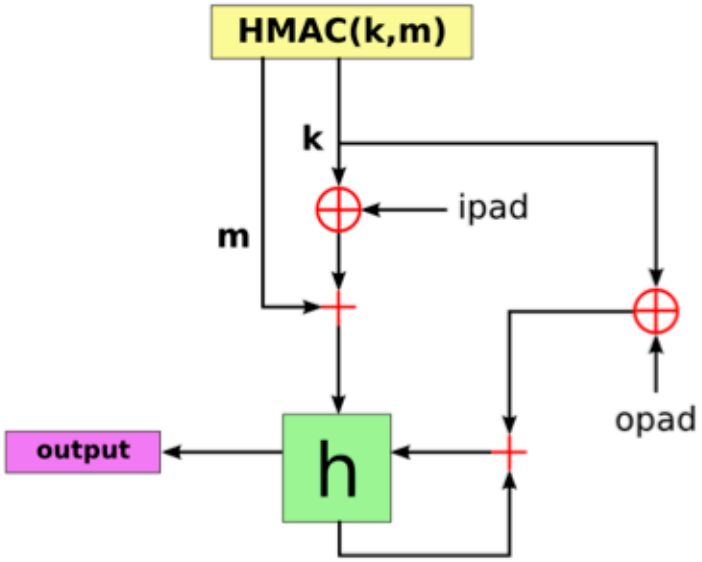
- **Length prepending**
- **Encrypt-last-block**
  - Encrypt-last-block CBC-MAC (ECBC-MAC)
  - $E(k_2, CBC - MAC(k_1, m))$

Other flaws:

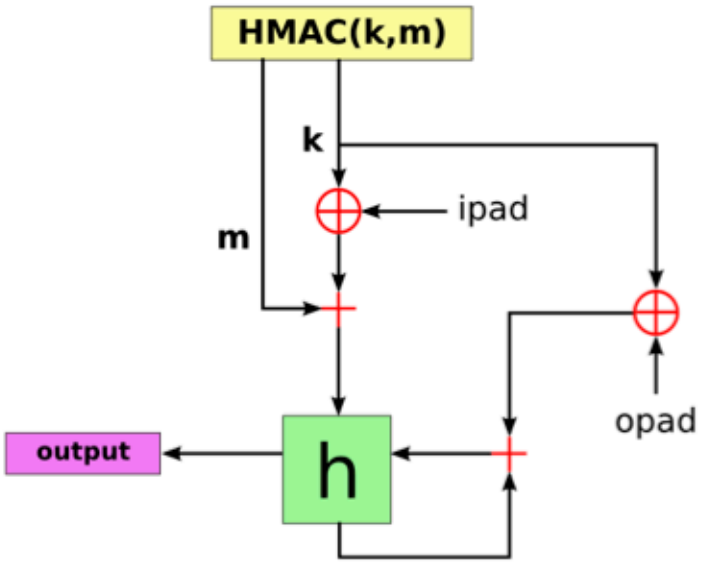
- Using the same key for encryption and authentication
- Allowing the initialization vector to vary in value
- Using predictable initialization vector

HMAC

- $\mathcal{H}$ a hash function (SHA-2, SHA-3, ...) with n-bit digests
- produces a n-bit MAC (Krawczyk, Bellare and Cannetti – 1996)



HMAC



$$\mathrm{HMAC}(K, m) = \mathcal{H}\Big((K' \oplus opad) \| \mathcal{H}\big((K' \oplus ipad) \| m\big)\Big)$$

- $K' = K$ padded with zeroes (to the right)
- $opad = 0\mathrm{x}5\mathrm{c}5\mathrm{c}5\mathrm{c}\ldots5\mathrm{c}5\mathrm{c}$ (one-block-long hexadecimal constant)
- $ipad = 0\mathrm{x}363636\ldots3636$ (one-block-long hexadecimal constant)

Introduction à la Sécurité

Cryptologie asymétrique 1/2

Damien Vergnaud

Sorbonne Université
Institut Universitaire de France

Contents

Limitations of Secret Key (Symmetric) Cryptography

- Secret key cryptography
  - symmetric encryption $\rightsquigarrow$ confidentiality
  - MAC $\rightsquigarrow$ authentication (integrity)
- Sender and receiver must share the same key
  - **needs** secure channel for key distribution
- Other limitation of authentication scheme
  - cannot authenticate to **multiple** receivers
  - does not have non-repudiation

How to distribute the cryptographic keys?

- If the users can meet in person beforehand - it's simple.
- But what to do if they cannot meet? (e.g. on-line shopping)

A Naive solution

- give to every user P_i a separate key K_{ij} to communicate with every P_j
- $\rightsquigarrow$ quadratic number of keys is needed
- $\rightsquigarrow$ someone needs to "give the keys"
- $\rightsquigarrow$ the users need to store large numbers of keys in a secure way

The solution: Public-Key Cryptography

- first proposed by Diffie and Hellman:

W.Diffie and M.E.Hellman,
New directions in cryptography
IEEE Trans. Inform. Theory, IT-22, 6, 1976, pp. 644-654.

- similar idea by Merkle:
  - **1974**: a project proposal for a Computer Security course at UC Berkeley (it was rejected)
  - **1975**: submitted to the CACM journal (it was rejected)
    (see `http://www.merkle.com/1974/`)

- 2015 Turing Award

- It 1997 the GCHQ revealed that they new it already in 1970 (James Ellis).

The idea

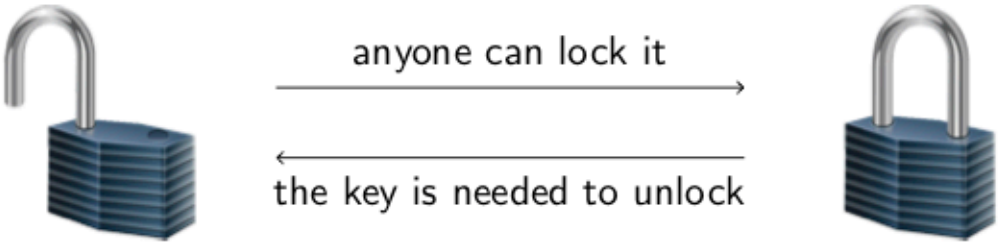
- instead of using one key K: use 2 keys (e, d)
  - $e \rightsquigarrow$ encryption,
  - $d \rightsquigarrow$ decryption,
- e can be public and only d has to be kept secret!

- **Public Key Encryption**
  - Message + Bob's Public Key = Ciphertext
  - Ciphertext + Bob's Private Key = Message
- anyone with Bob's public key can send Bob a secret message.
- only Bob can decrypt the message, since only Bob has the private key.

- **Digital signatures**
  - Message + Alice's Private Key = Signature
  - Message + Signature + Alice's Public Key = 0 or 1
- anyone with Alice's public key can verify that the message comes from Alice.
- only Alice can produce the signature, since only Alice has the private key.

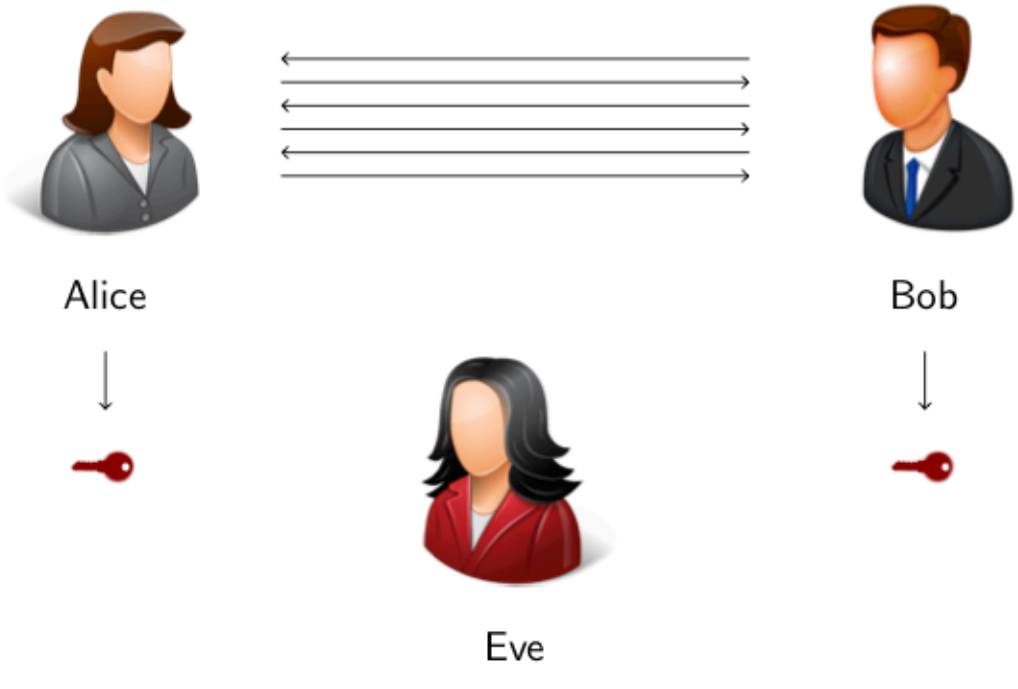
But is it possible?

- In "physical world" : yes!
  ⇝ Example: padlock



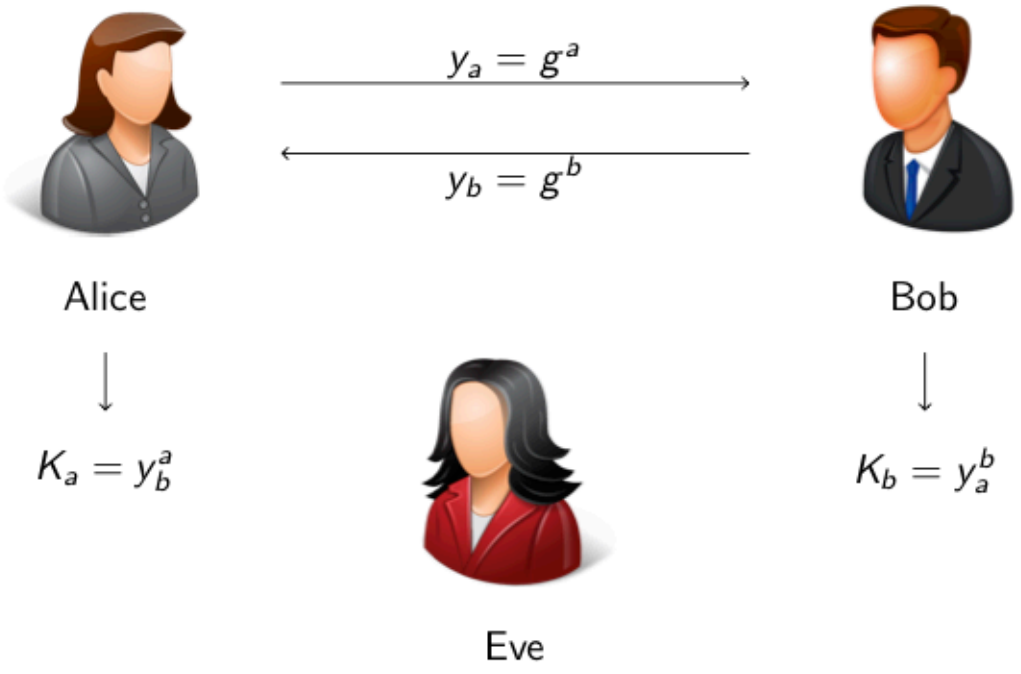
- Diffie and Hellman proposed the public key cryptography in 1976.
  - They just proposed the concept, not the implementation.
  - But they have shown a protocol for key-exchange

Key Exchange



Diffie-Hellman Key Exchange

$(\mathbb{G}, \cdot)$ a finite cyclic group; $\langle g \rangle = \mathbb{G}$



$$K_a = y_b^a = (g^b)^a = g^{ab} = (g^a)^b = y_a^b = K_b$$

Diffie-Hellman Key Exchange: Security

Eve knows:

- $(\mathbb{G}, g)$
- $y_a = g^a$
- $y_b = g^b$

and should have no information on $K = g^{ab}$.

- If finding a from y_a is easy then the DH key exchange is not secure.
- Even if it is hard, then

  ...the scheme may also not be completely secure

- How to choose $\mathbb{G}$?

  ...First choice (**bad**): $\mathbb{G} = (\mathbb{Z}/n\mathbb{Z}, +)$ for some integer n.

  ...Second choice (**good**): $\mathbb{G} = (\mathbb{Z}/n\mathbb{Z}^*, \cdot)$ for some integer n.

The Fundamental Equation

$$Z = Y^X \mod N$$

When Z is unknown, it can be efficiently computed

Exponentiation by squaring – *square-and-multiply*

$$y^x = \begin{cases} 1 & \text{if } x = 0 \\ y \cdot y^{x-1} & \text{if } x \text{odd} \\ (y^2)^{x/2} & \text{if } x \text{even} \end{cases}$$

```
long pow(long y, long x)
{
    long result = 1;
    while ( x ) {
        if ( x & 1 ) {
            result *= y;
        }
        y *= y;
        x /= 2;
    }
    return result;
}
```

Efficiency of computation modulo n

Suppose that n is a k-bit number, and $0 \leq x, y \leq n$

- $(x \pm y) \mod n \rightsquigarrow O(k)$
- $(xy) \mod n \rightsquigarrow O(k^2)$ (ou $\tilde{O}(k)$)
- $(x)^c \mod n \rightsquigarrow O((logc)k^2)$ ou $\tilde{O}((\log c)k)$
- $(x^{-1}) \mod n \rightsquigarrow O(k^3)$ (ou $\tilde{O}(k^2)$) ou $O(k^2)$ (ou $\tilde{O}(k)$)

The Fundamental Equation

$$Z = Y^X \mod N$$

When X is unknown, the problem is known as the **discrete logarithm** *and is generally believed to be hard to solve*

We will see that later...

The Fundamental Equation

$$Z = Y^X \mod N$$

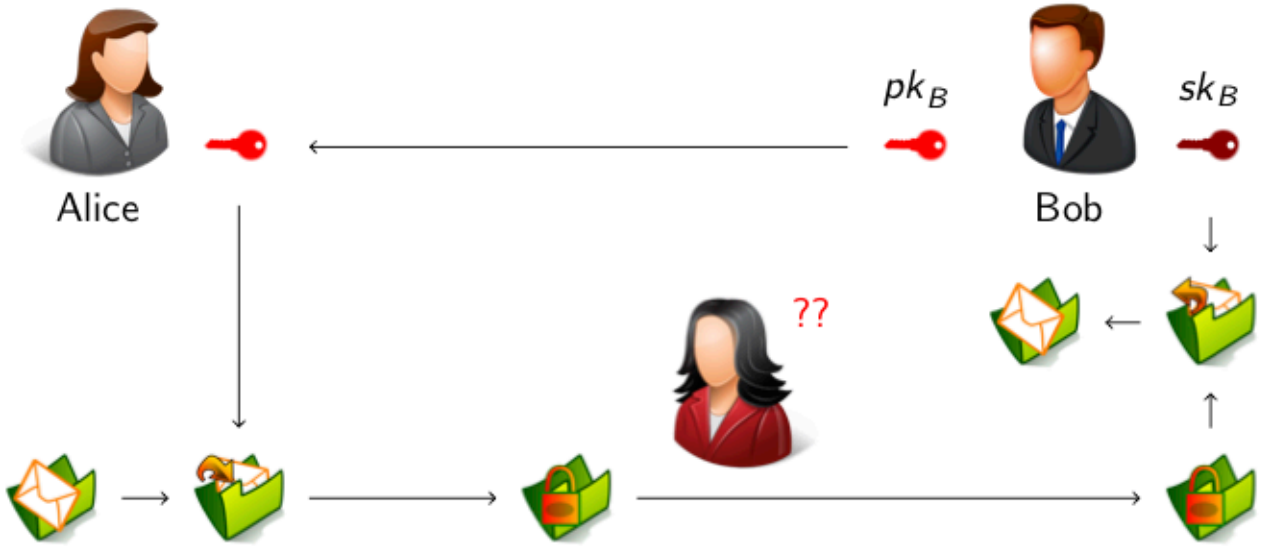
When Y is unknown, the problem is known as the **discrete root extraction** *and is generally believed to be hard to solve unless the factorisation of N is known.*

We will see that later...

Public Key Cryptosystems

Asymmetric encryption: Bob owns two "keys"

- a public key — known by everybody (including Alice)
- a secret key — known by Bob only



Public-Key Encryption

An asymmetric encryption scheme is a triple of algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ where

- $\mathcal{K}$ is a probabilistic **key generation algorithm** which returns random pairs of secret and public keys (sk, pk) depending on the security parameter κ,
- $\mathcal{E}$ is a probabilistic **encryption algorithm** which takes on input a public key pk and a plaintext $m \in \mathcal{M}$, runs on a random tape $u \in \mathcal{U}$ and returns a ciphertext c,
- $\mathcal{D}$ is a deterministic **decryption algorithm** which takes on input a secret key sk, a ciphertext c and returns the corresponding plaintext m or the symbol $\perp$. We require that if $(sk, pk) \leftarrow \mathcal{K}$, then $\mathcal{D}_{sk}(\mathcal{E}_{pk}(m, u)) = m$ for all $(m, u) \in \mathcal{M} \times \mathcal{U}$.

Public-Key Encryption: Security Notions

Encryption is supposed to provide confidentiality of the data.

But what exactly does this mean?

Security goal	But ...
Recovery of secret key is infeasible	True if data is sent in the clear
Obtaining plaintext from ciphertext is infeasible	Might be able to obtain half the plaintext
etc	*etc*

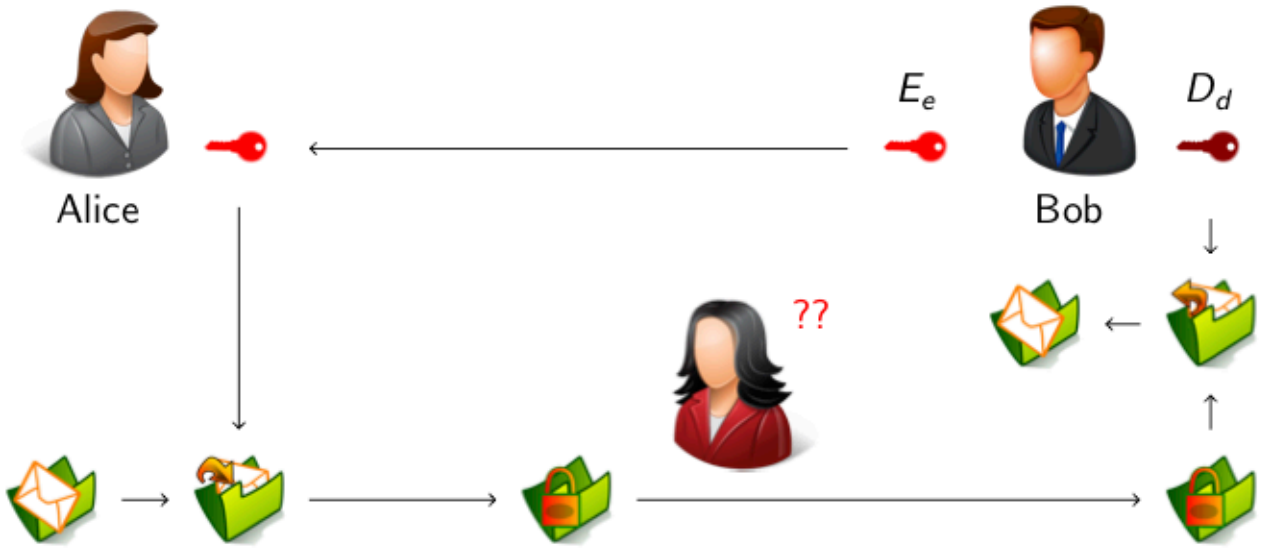
So what is a **secure** encryption scheme ?

Not an easy question to answer ...

Trapdoor permutations

- A **trapdoor function** is a function that
  - is easy to compute in one direction,
  - yet believed to be difficult to compute in the opposite direction (finding its inverse) without special information, called the "trapdoor".
- A trapdoor permutation family $\{E : X \longrightarrow X\}_{(e,d)}$
  - easy to compute $y = E_e(x)$ for any $x \in X$,
  - (believed to be) difficult to compute $E_e^{-1}(y)$ for any $y \in X$,
  - except if one knows d : $E_e^{-1}(y) = D_d(y) = x$.
- Do such functions exist?

How to encrypt a message m



Warning: in general it's not that simple. We will explain it later.

RSA - Key Generation

Rivest, Shamir, Adleman (1978)
A method for obtaining digital signatures and public key cryptosystems.
Communications of the ACM 21 (2): pp.120-126.

2002 Turing Award

- **Key generation:**
  - Generate two large primes p and q ($p \neq q$). How ?
  - Compute $N = p \cdot q$ and $\varphi(N) = (p-1)(q-1)$.
  - Select a random integer e, $1 < e < \varphi(N)$, such that $\gcd(e, (p-1)(q-1)) = 1$.
  - Compute the unique integer d, $1 < d < \varphi(N)$ with $e \cdot d \equiv 1 \mod \varphi(N)$.

  Public key $= (N, e)$ which can be published.
  Private key $= (d, p, q)$ which needs to be kept secret

RSA - Encryption / Decryption

- **Encryption:** if Alice wants to encrypt a message for Bob, she does the following:
  - Obtain Bob's authentic public key (N, e).
  - Represent the message as a number $0 < m < N$.
  - Compute $c = m^e \mod N$.
  - Send the ciphertext c to Bob.

- **Decryption:** to recover m from c, Bob does the following:
  - Use the private key d to recover $m = c^d \mod N$.

RSA - Proof That Decryption Works

Recall that $e \cdot d \equiv 1 \mod \varphi(N)$, so there exists an integer k such that

$$e \cdot d = 1 + k \cdot \varphi(N).$$

- If $\gcd(m, p) = 1$:
  - By **Fermat's Little Theorem** we have $m^{p-1} \equiv 1 \mod p$.
  - Taking $k(q-1)$-th power and multiplying with m yields

$$m^{1+k(p-1)(q-1)} \equiv m \mod p$$

- If $\gcd(m, p) = p$, then $m \equiv 0 \mod p$ and the previous equality is valid again.

Hence, in all cases $m^{e \cdot d} \equiv m \mod p$ and by a similar argument we have $m^{e \cdot d} \equiv m \mod q$.
Since p and q are distinct primes, the **CRT** leads to

$$c^d = (m^e)^d = m^{ed} = m^{k(p-1)(q-1)+1} = m \mod N.$$

Prime Numbers

- prime numbers are needed for RSA

Theorem (Prime number theorem)

The number of primes less than x is about $x/\log x$.

- $\rightsquigarrow$ primes are quite common ($\simeq 2^{503}$ primes $\leq 2^{512}$).
- testing primes can be done very fast!
- generating primes can be done very fast!
  (on average, one need to test 177 numbers before one find a 512-bit prime)

Fermat's test

Theorem (Fermat's little theorem)

For $a \in (\mathbb{Z}/n\mathbb{Z})^*$, $a^{\varphi(n)} \equiv 1 \bmod n$.

- if n is prime we have $a^{n-1} \equiv 1 \bmod n$ always
- if n is not prime we have $a^{n-1} \equiv 1 \bmod n$ is unlikely

Fermat's test

For $i = 1$ to k do

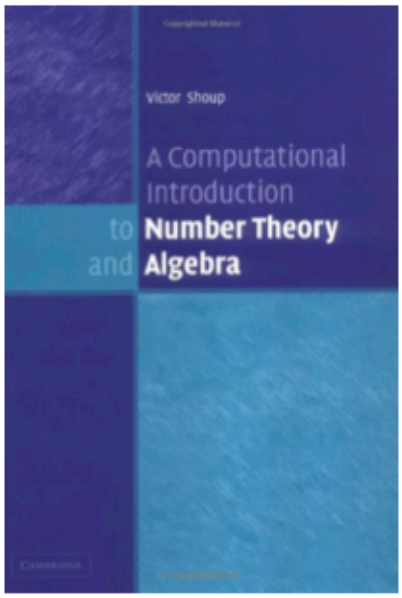
- Pick a randomly from $(\mathbb{Z}/n\mathbb{Z})^*$
- Compute $b = a^{n-1} \bmod n$
- If $b \not\equiv 1$ output (Composite,a)

output Possibly Prime

Carmichael numbers

- Carmichael numbers are composite numbers n which fail the Fermat Test for every a not dividing n.
- There are **infinitely many** Carmichael Numbers
  - the first three are 561, 1105, 1729
- **Exercise:** Carmichael Numbers N have the following properties
  1. always odd
  2. are square free
  3. if p divides N then $p - 1$ divides $N - 1$.
  4. have at least three prime factors
- Need for other tests

References



A Computational Introduction to Number Theory and Algebra
Victor Shoup

References

Prime Numbers: A Computational Perspective
Crandall, Richard, Pomerance, Carl B.

Security of RSA

- Security of RSA relies on difficulty of finding d given N and e.
- If we can **factor** N then we can find p and q
  - Hence we can calculate d.
- i.e. **If factoring is easy we can break RSA.**
  - Currently 768 bit numbers are the largest that have been (2010) factored
  - Hence best to choose (at least) **2048 bit** numbers
- Is RSA as strong as factorization? Will next show that knowing d we can factor N.
  - Still does not rule out possibility that breaking RSA is easier than factoring

Integer Factoring

- Exponential methods:
  - **trial division**
  - **Pollard's $p-1$ method**
  - **Pollard's ρ method**
- Three most effective algorithms are:
  - **quadratic sieve**
  - **elliptic curve factoring algorithm** (ECM)
  - **number field sieve** (NFS)
- One idea many factoring algorithms use:
  - Suppose one fine $x^2 \equiv y^2 \mod N$ s.t. $x \neq \pm y \mod N$.
  - Then $N \mid (x-y)(x+y)$.
  - Neither $(x-y)$ nor $(x+y)$ is divisible by N; thus,

$\gcd(x-y, N)$ *has a non-trivial factor of* N.

Time complexity of Integer Factoring

- **quadratic sieve:**

$$O(\exp((1+o(1))\sqrt{\ln N \ln\ln N}))$$

[For $N \simeq 2^{1024}$, "$O(e^{68})$"]

- **elliptic curve factoring algorithm:**

$$O(\exp((1+o(1))\sqrt{2\ln p \ln\ln p})),$$

where p is N's smallest prime factor
[For $N = pq$ and $p, q \simeq 2^{512}$, "$O(e^{65})$"]

- **number field sieve:**

$$O(\exp((1.92+o(1))(\ln N)^{1/3}(\ln\ln N)^{2/3}))$$

[For $N \simeq 2^{1024}$, "$O(e^{60})$"]

- Multiple 512-bit moduli have been factored
- Extrapolating trends of factoring suggests that:
  - 1024-bit moduli will be factored by **2018** ...

Knowledge of $\varphi(N)$

- We will show knowledge of $\varphi(N)$ allows us to factor N as well.
- We have

$$\varphi(N) = (p-1)(q-1) = N - (p+q) + 1.$$

- Hence

$$\begin{aligned} S &= p+q = N+1-\varphi(N) \\ P &= pq = N \end{aligned}$$

- p and q are the **roots** of $X^2 - SX + P = 0$.

Security of RSA

- Suppose you can find d for a given N and e.
- Then for some integer s

$$ed - 1 = s(p-1)(q-1).$$

- Hence for any $x \neq 0$

$$x^{ed-1} = 1 \mod N.$$

- We want to put

$$y_1 = \sqrt{x^{ed-1}} = x^{(ed-1)/2}$$

and then use

$$y_1^2 - 1 \equiv 0 \mod N$$

to recover a factor of N from $\gcd(y_1 - 1, N)$.

- This will only work when $y_1 \neq \pm 1 \mod N$.

Security of RSA

- Now suppose $y_1 = 1 \mod N$, then we take a square root of y_1

$$y_2 = \sqrt{y_1} = x^{(ed-1)/4}$$

- We know $y_2^2 = y_1 = 1 \mod N$. Hence we compute $\gcd(y_2 - 1, N)$ and see if this gives a factor of N.
- We repeat until
  - either we have factored N
  - or $(ed-1)/2^s$ is no longer divisible by 2.
- We will factor N with probability $1/2$.

Shared Modulus

- Assume for efficiency that each user has
  - The **same modulus** N
  - Different public/private exponents (e_i, d_i)
- Suppose I am user number one, and I want to find user number two's d_2.
  - User one computes p and q since they know d_1.
  - User one computes $\varphi(N) = (p-1)(q-1)$
  - User one computes $d_2 = e_2^{-1} \mod \varphi(N)$
- So each user can then find every other users key.

What about an eavesdropper ?

Shared Modulus

- **Now suppose the attacker is not one of the people who share a modulus**
- Suppose Alice sends the message m to two people with public keys
  - $(N, e_1), (N, e_2)$, i.e. $N_1 = N_2 = N$.
- Eve can see the messages c_1 and c_2 where
  - $c_1 = m^{e_1} \mod N$
  - $c_2 = m^{e_2} \mod N$

Shared Modulus

- Eve can now compute
  - $t_1 = e_1^{-1} \mod e_2$
  - $t_2 = (t_1 e_1 - 1)/e_2$

- Eve can then **retrieve the message** from

$$\begin{aligned} c_1^{t_1} c_2^{t_2} &\equiv m^{e_1 t_1} m^{-e_2 t_2} \mod N \\ &\equiv m^{1+e_2 t_2} m^{-e_2 t_2} \mod N \\ &\equiv m^{1+e_2 t_2 - e_2 t_2} \mod N \\ &\equiv m \mod N \end{aligned}$$

Small Public Exponent

Hastad (1988)
Solving Simultaneous Modular Equations of Low Degree.
SIAM J. Comput. 17(2): 336-341

- Suppose we have three users
  - With public moduli N_1, N_2 and N_3
  - All with public exponent $e = 3$

- Suppose Alice sends them the **same** message m

- Eve sees the messages
  - $c_1 = m^3 \mod N_1$
  - $c_2 = m^3 \mod N_2$
  - $c_3 = m^3 \mod N_3$

- Now Eve, using the **CRT**, computes the solution to

$$X = c_i \mod N_i$$

to obtain

$$X \mod N_1 N_2 N_3.$$

Small Public Exponent

- So the attacker has
$$X \mod N_1 N_2 N_3.$$
- But since $m^3 < N_1 N_2 N_3$ we must have
$$X = m^3$$
over the integers. Hence
$$m = X^{1/3}.$$
- This attack is interesting since we find the message without factoring the modulus.
- This is evidence that breaking RSA is **easier than factoring** .

Small Private Exponent

Wiener (1990)
Cryptanalysis of short RSA secret exponents.
IEEE Transactions on Information Theory 36(3): 553-558

- To **reduce** the work load of the exponentiation, one may wish to use a small value of d rather than a random value
- Since modular exponentiation takes time linear in $\log(d)$, a small private key can improve performance
- If the card has limited computing power, a relatively small value of d would be handy.
- We present an attack, due to Wiener, that succeeds in computing the secret decryption exponent under certain conditions.

Small Private Exponent

- $N = pq$ with $q < p < 2q$, $d < (1/3) \cdot N^{0.25}$.
  $\rightsquigarrow$ Given (N, e) with $ed \equiv 1 \mod N$, attacker can efficiently recover d.
- The proof is using **continued fractions technique**
- There is an integer k such that $ed - k\varphi(N) = 1$.
- We have
$$\left| \frac{e}{\varphi(N)} - \frac{k}{d} \right| = \frac{1}{d\varphi(N)}.$$
- Since $N = pq > q^2$, we have $q < \sqrt{N}$ hence
  $N - \varphi(N) = p + q - 1 < 2q + q - 1 < 3\sqrt{N}$.
- Now we see that
$$\left| \frac{e}{N} - \frac{k}{d} \right| = \left| \frac{ed - kN}{dN} \right| = \left| \frac{1 + k(\varphi(N) - N)}{dN} \right| < \frac{3k\sqrt{N}}{dN} = \frac{3k}{\sqrt{N}}.$$

Small Private Exponent

- Since $k < d$, we have that $3k < 3d < N^{0.25}$, and hence
$$\left| \frac{e}{N} - \frac{k}{d} \right| < \frac{1}{dN^{0.25}}.$$
- Finally, since $3d < N^{0.25}$, we have that
$$\left| \frac{e}{N} - \frac{k}{d} \right| < \frac{1}{3d^2}.$$
- **Legendre theorem on continued fractions:** there are at most $\log N$ fractions k/d with $d < N$ approximately e/N so tightly, and they can be obtained by computing the $\log N$ convergents of the continued fraction expansion of e/N (*i.e.* Euclidean algorithm).

Introduction à la Sécurité

Cryptologie asymétrique 2/2

Damien Vergnaud

Sorbonne Université
Institut Universitaire de France

Contents

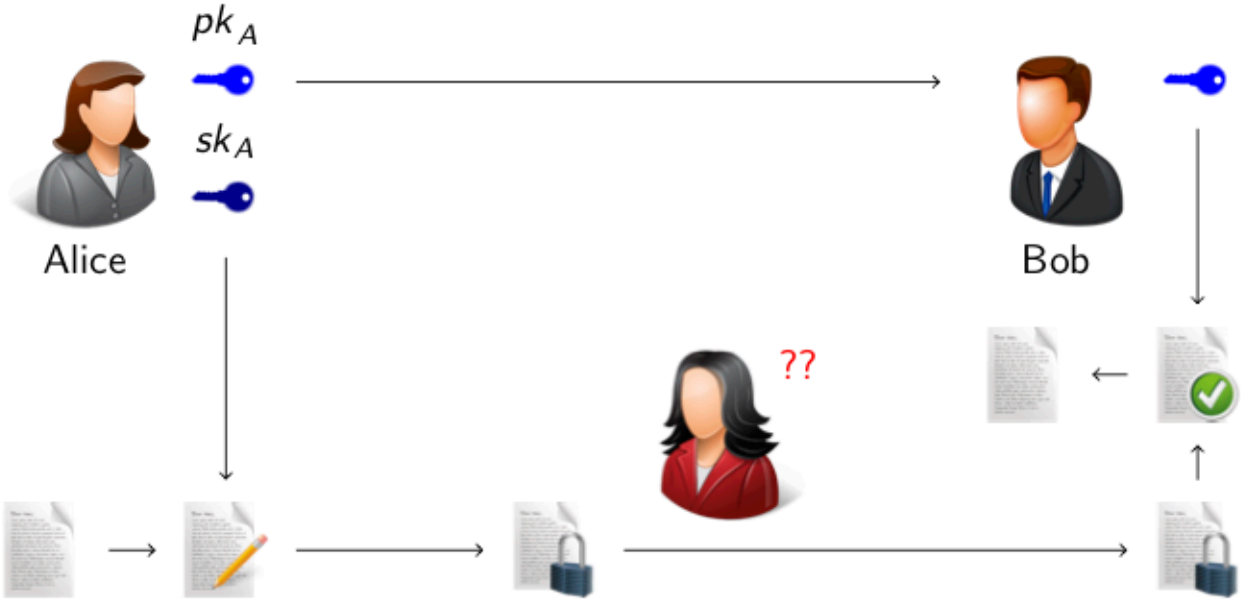
Digital Signatures

- A very important public key primitive is the **digital signature**.
- The idea is
  - Message + Alice's Private Key = Signature
  - Message + Signature + Alice's Public Key = YES/NO
- Alice can sign a message using her private key.
- Anyone can verify Alice's signature, since everyone can obtain her public key.
- the verifier is convinced that only Alice could have produced the signature
  - **only Alice knows her private key!**

Digital signature schemes

Digital signatures: Alice owns two "keys"

- a public key — known by everybody (including Bob)
- a secret key — known by Alice only



Digital Signatures : Services

- The verification algorithm is used to determine whether or not the signature is properly constructed.
- the verifier has guarantee of
  - message integrity and
  - message origin.
- also provide non-repudiation - not provided by MACs.

Most important cryptographic primitive!

Security Notions

Depending on the context in which a given cryptosystem is used, one may formally define a security notion for this system,

- by telling what **goal** an adversary would attempt to reach,
- and what means or information are made available to her (the **attack model**).

A security notion (or level) is entirely defined by pairing an adversarial goal with an adversarial model.

Examples: UB-KMA, UUF-KOA, EUF-SOCMA, EUF-CMA.

Signature Schemes

- Signer Alice generates a public/private key pair (pk, sk) by running a probabilistic **key generation algorithm** $G(k)$, k being the security parameter. Alice publishes pk.
- Whenever Alice wishes to sign a digital document $m \in \{0,1\}^*$, she computes the signature $s = S(sk, m)$ where S is the (possibly probabilistic) **signing algorithm**. She outputs s and maybe also m.
- Knowing m and s (and Alice's public key pk), Bob can verify that s is a signature of m output by Alice by running the **verification algorithm** $V(pk, m, s)$ returning 1 if $s = S(sk, m)$ or 0 otherwise.

The signature scheme is the triple (G, S, V) and their domains.

Security Goals

[Unbreakability] the attacker recovers the secret key sk from the public key pk (or an equivalent key if any). This goal is denoted **UB**. Implicitly appeared with public-key cryptography.

[Universal Unforgeability] the attacker, without necessarily having recovered sk, can produce a valid signature of any message in the message space. Noted **UUF**.

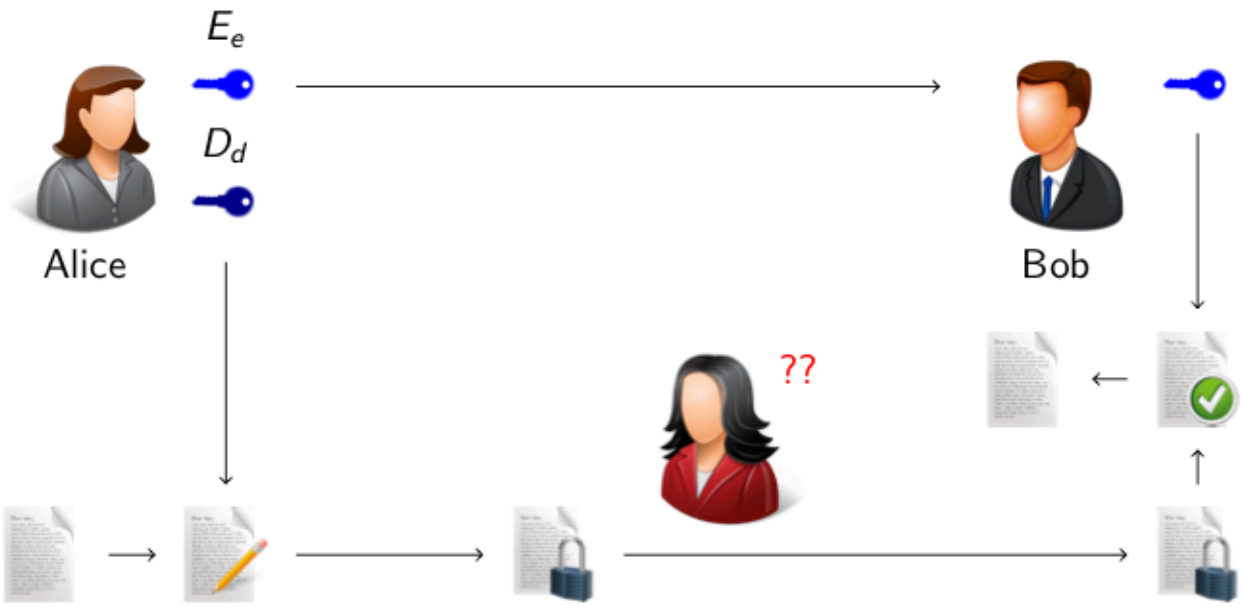
[Existential Unforgeability] the attacker creates a message and a valid signature of it (likely not of his choosing). Denoted **EUF**.

Adversarial Models

- **Key-Only Attacks** (KOA), unavoidable scenario.
- **Known Message Attacks** (KMA) where an adversary has access to signatures for a set of known messages.
- **Chosen-Message Attacks** (CMA) the adversary is allowed to use the signer as an oracle (full access), and may request the signature of any message of his choice

Digital signature from a Trapdoor Permutation ...

Trapdoor permutation $\{E : X \longrightarrow X\}_{(e,d)}$



Warning: in general it's not that simple. We will explain it later.

... Isn't Fully Secure On Its Own

Remark. Assume Eve picks some random σ and computes $m = E_e(\sigma)$. Then (m, σ) is a valid pair since $\sigma = D_d(m)$ is a valid signature of m.

> *Eve can generate signatures for messages he doesn't control. This capability is known as existential forgery.*

- weak form of forgery
- **What if stronger attacks exist ?**

RSA - Key Generation

Rivest, Shamir, Adleman (1978)
A method for obtaining digital signatures and public key cryptosystems.
Communications of the ACM 21 (2): pp.120-126.

- **Key generation:**
  - Generate two large primes p and q ($p \neq q$).
  - Compute $N = p \cdot q$ and $\varphi(N) = (p-1)(q-1)$.
  - Select a random integer e, $1 < e < \varphi(N)$, such that $\gcd(e, (p-1)(q-1)) = 1$.
  - Compute the unique integer d, $1 < d < \varphi(N)$ with $e \cdot d \equiv 1 \pmod{\varphi(N)}$.

Public key $= (N, e)$ which can be published.
Private key $= (d, p, q)$ which needs to be kept secret

RSA - Signature / Verification

- **Signature:** if Alice wants to sign a message, she does the following:
  - Represent the message as a number $0 < m < N$.
  - Use her private key d to compute $s = m^d \mod N$.
  - Send the signature s to Bob.

- **Verification:** to check the validity of s on m, Bob does the following:
  - Obtain Alice's authentic public key (N, e).
  - Check whether $m = s^e \mod N$.

Universal Forgery of RSA Signatures

RSA is a morphism: for $m_1, m_2 \in \mathbb{Z}_N^*$,

$$(m_1 m_2)^d = m_1^d m_2^d \pmod N ,$$

meaning that $S(m_1 m_2) = S(m_1)S(m_2)$.

Attack: now suppose Eve wants Alice's signature for some specific message $m =$"I owe Eve 10,000 euros".

1. she picks a random $m_1 \in \mathbb{Z}_N^*$ and computes $m_2 = m/m_1 \mod N$,
2. assume Eve asks Alice to sign m_1 and m_2 and receives $S(m_1)$ and $S(m_2)$,
3. Eve computes $S(m) = S(m_1)S(m_2) \mod N$ on her own.

Universal Forgery of RSA Signatures

This is a universal forgery under a chosen-message attack.

- much worse than existential forgery
- **but**, this attack assumes that Eve has access to Alice's signing operation

The Need for Hashing

Instead of signing the message m directly, let's apply a hash function H to it:

- Alice generates and publishes some trapdoor permutation E_e,
- she keeps D_d private,
- to sign m, Alice computes $s = D_d(H(m))$ and sends the pair (m, s) to Bob,
- to verify the signature, Bob checks whether $H(m) = E_e(s)$.

$\rightsquigarrow$ Hash-then-Invert paradigm. H is now a part of the scheme.

It must map messages to elements of E's domain, say X.

The Need for Hashing (Cont'd)

What have we done? Well, if H maps $\{0,1\}^*$ to X, then arbitrarily long messages can now be signed. Better.

What about existential forgery? Assume Eve picks some random σ and computes $\mu = E(\sigma)$, she faces the problem of finding an m such that $H(m) = \mu$.

The hope is that with a "good choice" for H, Eve cannot do that (in particular H must be one-way). Hopefully better.

What about universal forgery? Getting back to the multiplicative attack for E =RSA, the attacker has to find m_1, m_2 such that

$$H(m_1)H(m_2) = H(m) \pmod{N}.$$

Again, with a "good choice" for H, these should be difficult to find (H must somehow destroy the algebraic structure of $\mathbb{Z}_n^*$). Hopefully better.

The Need for Hashing (Cont'd)

Is there a drawback? Well, yes but moderate.

The use of H introduces a new type of attacks based on finding collisions. If the attacker finds m_1, m_2 such that $H(m_1) = H(m_2)$ then a signature of m_1 is also a signature of m_2.

$\rightsquigarrow$ existential forgery under a chosen-message attack: Eve queries Alice on m_1 to get s and then outputs (m_2, s) as a valid signature.

Here too, we hope that H is chosen in a way that makes collisions hard to find

So, How Good is DH's Approach?

- We pinpointed features of H that are necessary for the Hash-then-Invert scheme to thwart certain attacks.
- **But** the true question should be: what features of H are **sufficient** to prevent **all** attacks?
- $\rightsquigarrow$ provable security, that is, the set of techniques by which one assesses the security level of a cryptosystem given assumptions on its ingredients.

Lamport signatures

L. Lamport
Constructing digital signatures from a one-way function
Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, Oct. 1979.

- a Lamport signature or **Lamport one-time signature scheme** is a method for constructing efficient digital signatures.
- Lamport signatures can be built from any cryptographically secure **one-way** function; usually a **cryptographic hash function** is used.
- Unfortunately each Lamport key can only be used to sign a single message.
- However, we will see how a single key could be used for **many** messages, making this a fairly efficient digital signature scheme.

How to sign **one** bit **just once** ?

$\mathcal{M} = \{0,1\}$

- **Key generation:**
  - Generate $f : X \longrightarrow Y$ a **one-way function**.
  - Select two random elements $x_0, x_1 \in X$.
  - Compute their images $y_i = f(x_i)$ for $i \in \{0,1\}$.

  Public key $= (y_0, y_1)$ which can be published.
  Private key $= (x_0, x_1)$ which needs to be kept secret

- **Signature:** if Alice wants to sign a bit b, she does the following:
  - Use her private key (x_0, x_1) to send the signature $s = x_b$ to Bob.
- **Verification:** to check the validity of s on b, Bob does the following:
  - Obtain Alice's authentic public key (y_0, y_1).
  - Check whether $f(s) = y_b$.

How to sign **k** bits **just once** ?

$\mathcal{M} = \{0,1\}^k$

- **Key generation:**
  - Generate $f : X \longrightarrow$ a **one-way function**.
  - Select $2k$ random elements $x_{0,1}, x_{1,1}, \ldots, x_{0,k}, x_{1,k} \in X$.
  - Compute their images $y_{i,j} = f(x_{i,j})$ for $i \in \{0,1\}$ and $j \in [\![1,k]\!]$.

  Public key $= (y_{0,1}, y_{1,1}, \ldots, y_{0,k}, y_{1,k})$ which can be published.
  Private key $= (x_{0,1}, x_{1,1}, \ldots, x_{0,k}, x_{1,k})$ which needs to be kept secret

- **Signature:** if Alice wants to sign $m = m_1 \ldots m_k$, she does the following:
  - Use her private key $(x_{0,1}, x_{1,1}, \ldots, x_{0,k}, x_{1,k})$ to send the signature $s = (x_{m_1,1}, x_{m_1,2}, \ldots, x_{m_k,k})$ to Bob.
- **Verification:** to check the validity of $s = (s_1, \ldots, s_k)$ on m, Bob does the following:
  - Obtain Alice's authentic public key $(y_{0,1}, y_{1,1}, \ldots, y_{0,k}, y_{1,k})$.
  - Check whether $f(s_i) = y_{m_b,i}$ for all $i \in [\![1,k]\!]$.

How to sign **k** bits **just once** ?

- Lamport's scheme is EUF-CMA secure assuming **only** the one-wayness of f.
- The signature generation is very efficient.

- For a hash function that generates a n-bit message digest, the ideal preimage resistance on a single hash function invocation implies on the order of 2^n operations and 2^n bits of memory effort to find a preimage under a classical computing model.
- For a 128-bit security level, $Y = \{0,1\}^{128}$ and the public-key is made of $256 \cdot k$ bits and its generation requires 256 evaluations of the function f.
- The signature is made of k elements from X. If $X = \{0,1\}^{128}$ the signature length is $128 \cdot k$ bits.
- Can sign only one message

Lamport's signatures: variants

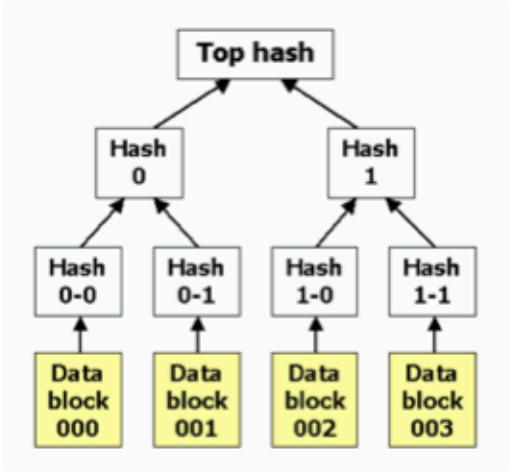
- **Short private key.** Instead of creating and storing all the random numbers of the private key a single key of sufficient size can be stored.

  The single key can then be used as the seed for a **cryptographically secure pseudorandom number generator** to create all the random numbers in the private key when needed.

- **Short public key** A Lamport signature can be combined with a hash list, making it possible to only publish a single hash instead of all the hashes in the public key.

- **Hashing the message.**
  - Unlike some other signature schemes the Lamport signature scheme **does not** require that the message m is hashed before it is signed.
  - A system for signing long messages can use a collision resistant hash function h and sign $h(m)$ instead of m.

Lamport's signatures: variants



- **Public key for multiple messages.**
  - many keys have to be published if many messages are to be signed.
  - a hash tree can be used on those public keys, publishing the top hash of the hash tree instead.
  - this increases the size of the resulting signature (parts of the hash tree have to be included in the signature)
  - it makes it possible to publish a single hash that then can be used to verify any given number of future signatures.

Textbook ElGamal signatures

ElGamal (1985)
A Public-Key Cryptosystem and a Signature Scheme based on Discrete Logarithms.
IEEE Transactions Information Theory, 31 pp. 469-472.

Key generation. $G(1^k)$ randomly selects a k-bit prime p and a generator g of $\mathbb{Z}_p^*$. The secret key is $x \leftarrow \mathbb{Z}_{p-1}$ and setting $y = g^x \mod p$, the public key is (p, g, y).

Signature. To sign a message $m \in \mathbb{Z}_{p-1}$, one generates (r, s) such that $g^m = y^r r^s \mod p$ as follows. Randomly select $k \leftarrow \mathbb{Z}_{p-1}^*$, set $r = g^k \mod p$ and $s = (m - xr)/k \mod p - 1$. Output (r, s).

Verification. Verify that $1 < r < p$ and $g^m = y^r r^s \mod p$.

Insecure! *cf* **TD 5**. *Hash the message first!*

The Fiat-Shamir heuristic

Fiat, Shamir (1986)
How to Prove Yourself: Practical Solutions to Identification and Signature Problems.
Advances in Cryptology - Crypto'86, Lect. Notes Comput. Science 263, pp. 186-194.

- Fiat and Shamir presented a (zero-knowledge) identification scheme consisting of 3 messages between a **prover** P and a **verifier** V.
- In a 3-pass (public-coin) identification scheme, these messages are called **commitment**, **challenge** and **response**. The challenge is randomly chosen by V.

Fiat-Shamir Transform: replace the challenge by a hash value taken on scheme parameters and t, thereby removing V. This transforms the protocol by making it *non-interactive*.

The intuition is that any "sufficiently random" hash function should preserve the security of the protocol.

Fiat-Shamir Identification Protocol

- **One-time setup:**
  - Trusted party T publishes an RSA modulus $N = pq$ but keeps primes p and q secret;
  - Alice select a secret $s \in \mathbb{Z}_n^*$ and publishes $v = s^2 \mod n$ as her public key.
- **Protocol:** (repeated ℓ times)
  - Alice picks $k \in \mathbb{Z}_n^*$ at random and sends $t = k^2 \mod n$ (commitment);
  - Bob picks at random a bit $c \in \{0, 1\}$ and sends it to Alice (challenge);
  - Alice computes $r = k \cdot s^c \mod n$ and sends it to Bob (response);
  - Bob accepts the proof if $r \neq 0$ and $r^2 = t \cdot v^c$.

Fiat Shamir signature

$\ell =$ **security parameter** (e.g. $\ell = 100$); $H : \{0,1\}^* \longrightarrow \{0,1\}^\ell$ hash function.

- **Key generation:**
  - Alice publishes an RSA modulus $N = pq$ but keeps primes p and q secret;
  - Alice select a secret $s \in \mathbb{Z}_n^*$, computes $v = s^2 \mod n$ and publishes (n, v) as her public key.

- **Signature generation:** message $m \in \{0,1\}$
  - Alice picks $k_1, \ldots, k_\ell \in \mathbb{Z}_n^*$ at random and computes $t_i = k_i^2 \mod n$ for $i \in [\![1, \ell]\!]$;
  - She computes $\vec{c} = (c_1, \ldots, c_\ell) = H(m, t_1, \ldots, t_\ell)$
  - Alice computes $r_i = k_i \cdot s^{c_i} \mod n$ for $i \in [\![1, \ell]\!]$ and the signature is $\sigma = (r_1, \ldots, r_n, \vec{c})$

- **Verification:**
  - Bob computes $\tilde{t}_i = r_i^2 v_i^{-c_i}$ for $i \in [\![1, \ell]\!]$ and accepts σ iff $\vec{c} = H(m, \tilde{t}_1, \ldots, \tilde{t}_n)$.

The Fiat-Shamir Transform

The same heuristic transformation can be applied to any 3-pass identification protocol to construct a signature scheme: the message m becomes a parameter of the hash value $e = H(t, m)$.

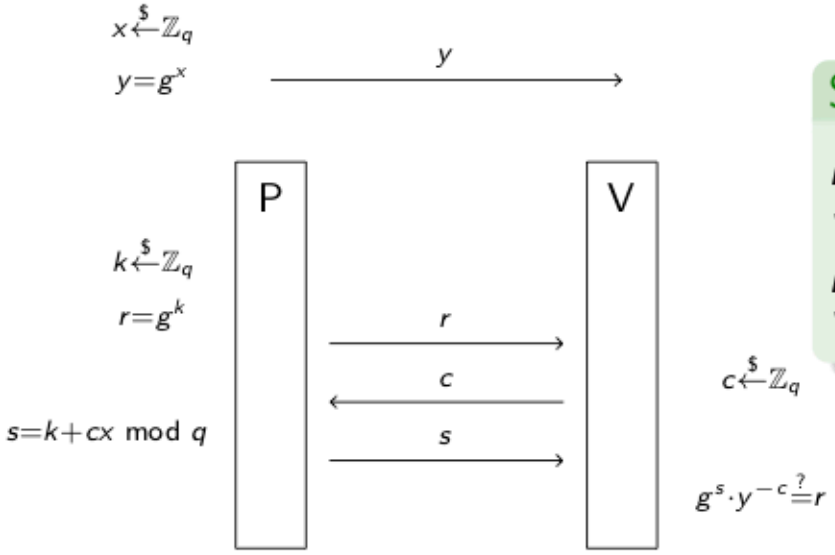
Intuitively, it should be hard for a forger to find m and a protocol transcript (t, e, s) for which it is true both that $e = H(t, m)$ and (t, e, s) is a valid transcript w.r.t. a given public key.

So-obtained signature schemes are **more efficient** (while hopefully achieving strong security) than other constructions.

Schnorr's ID Protocol (91)

Let $\mathbb{G} = \langle g \rangle$ be a group of prime order q

Prover P proves to verifier V that he knows the discrete log x of a public group element $y = g^x$. It is a 3-move protocol.



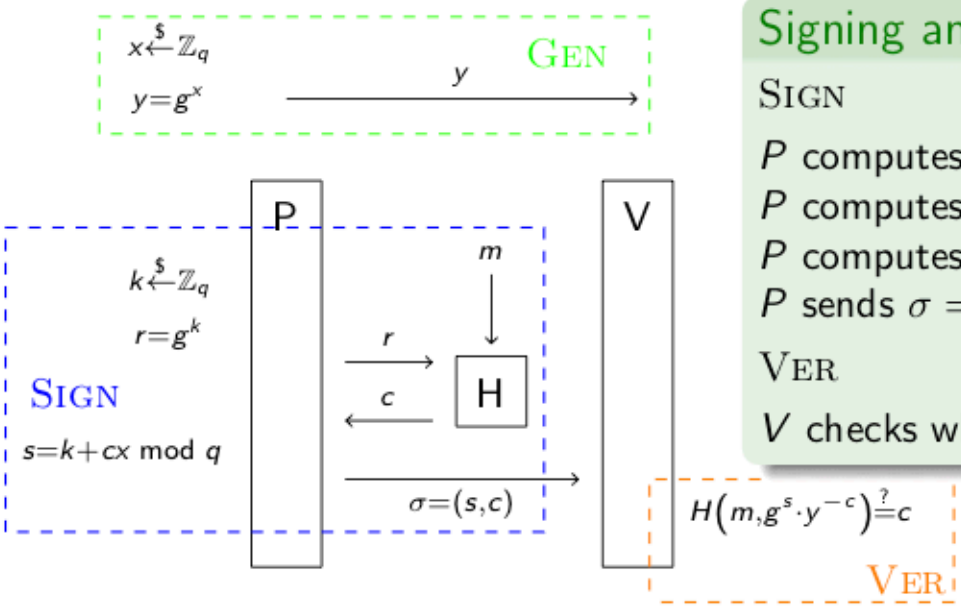
Scenario

P sends $r = g^k$ where $k \xleftarrow{\$} \mathbb{Z}_q$
V sends $c \xleftarrow{\$} \mathbb{Z}_q$
P sends $s = k + cx \bmod q$
V checks whether $g^s \cdot y^{-c} = r$

Schnorr Signatures (via the Fiat-Shamir Transform)

Introduce a hash function $H : \{0,1\}^\star \mapsto \mathbb{Z}_q$

Schnorr's signature scheme Σ_H is a tuple of probabilistic algorithms $\Sigma_H = (\textsc{Gen}, \textsc{Sign}, \textsc{Ver})$ defined as follows.



Signing and Verifying

SIGN

P computes $r = g^k$ where $k \xleftarrow{\$} \mathbb{Z}_q$
P computes $c = H(m, r)$
P computes $s = k + cx \bmod q$
P sends $\sigma = (s, c)$

VER

V checks whether $H(m, g^s \cdot y^{-c}) = c$

Digital Signature Algorithm (DSA)

- The Digital Signature Algorithm (DSA) is a United States Federal Government standard or FIPS for digital signatures.
- It was proposed by the National Institute of Standards and Technology (NIST) in **August 1991** for use in their Digital Signature Standard (DSS), specified in FIPS 186, adopted in **1993**.
- DSA makes use of a cryptographic hash function $\mathcal{H}$.
  In the original DSS, $\mathcal{H}$ was always SHA, but stronger hash functions from the SHA family are also in use.
- The original DSS constrained the key length to be a multiple of 64 between 512 and 1024 (inclusive).

Digital Signature Algorithm (DSA)

Textbook ElGamal signature scheme

Key generation. $G(1^k)$ randomly selects a k-bit prime p and a generator g of $\mathbb{Z}_p^*$.

The secret key is $x \leftarrow \mathbb{Z}_{p-1}$
The public key is $(p, g, y = g^x \mod p)$.

Signature. To sign a message $m \in \mathbb{Z}_{p-1}$, one generates (r, s) such that

$$g^m = y^r r^s \mod p$$

as follows. Randomly select $k \leftarrow \mathbb{Z}_{p-1}^*$, set $r = g^k \mod p$ and

$$s = (m - xr)/k \mod p - 1.$$

Output (r, s).

Verification. Verify that $1 < r < p$ and

$$g^m = y^r r^s \mod p$$

.

Digital Signature Algorithm (DSA)

Hashed ElGamal signature scheme

Key generation. $G(1^k)$ randomly selects a k-bit prime p and

a generator g of $\mathbb{Z}_p^*$.

The secret key is $x \leftarrow \mathbb{Z}_{p-1}$
The public key is $(p, g, y = g^x \mod p)$ and a hash function $\mathcal{H}$.

Signature. To sign a message $m \in \mathbb{Z}_{p-1}$, one generates (r, s) such that

$$g^{\mathcal{H}(m)} = y^r r^s \mod p$$

as follows. Randomly select $k \leftarrow \mathbb{Z}_{p-1}^*$, set $r = g^k \mod p$ and

$$s = (\mathcal{H}(m) - xr)/k \mod p - 1.$$

Output (r, s).

Verification. Verify that $1 < r < p$ and

$$g^{\mathcal{H}(m)} = y^r r^s \mod p$$

.

Digital Signature Algorithm (DSA)

Hashed ElGamal signature scheme with Schnorr's trick

Key generation. $G(1^k)$ randomly selects a k-bit prime p and

a generator g of $\mathbb{G} \subset \mathbb{Z}_p^*$ of prime order q.

The secret key is $x \leftarrow \mathbb{Z}_q$
The public key is $(p, q, g, y = g^x \mod p)$ and a hash function $\mathcal{H}$.

Signature. To sign a message $m \in \mathbb{Z}_{p-1}$, one generates (r, s) such that

$$g^{\mathcal{H}(m)} = y^r r^s \mod p$$

as follows. Randomly select $k \leftarrow \mathbb{Z}_q^*$, set $r = g^k \mod p$ and

$$s = (\mathcal{H}(m) - xr)/k \mod q.$$

Output (r, s).

Verification. Verify that $1 < r < q$ and

$$g^{\mathcal{H}(m)} = y^r r^s \mod p$$

.

Digital Signature Algorithm (DSA)

DSA

Key generation. $G(1^k)$ randomly selects a k-bit prime p and a generator g of $\mathbb{G} \subset \mathbb{Z}_p^*$ of prime order q.

The secret key is $x \leftarrow \mathbb{Z}_q$

The public key is $(p, q, g, y = g^x \mod p)$ and a hash function $\mathcal{H}$.

Signature. To sign a message $m \in \mathbb{Z}_{p-1}$, one generates (r, s) such that

$$g^{\mathcal{H}(m)} = y^r r^s \mod p$$

as follows. Randomly select $k \leftarrow \mathbb{Z}_q^*$, set $r = g^k \mod p$ and

$$s = (\mathcal{H}(m) + xr)/k \mod q.$$

Output (r, s).

Verification. Verify that $1 < r < q$, calculate $w = s^{-1} \mod q$, $u_1 = \mathcal{H}(m) \cdot w \mod q$, $u_2 = r \cdot w \mod q$ and check whether

$$g^{u_1} y^{u_2} \mod p \mod q = r.$$