



## Crypto Basics

Recent block cipher: AES  
Public Key Cryptography  
Public key exchange: Diffie-Hellmann  
Homework suggestion

1

## What is a cryptosystem?

- $K = \{0,1\}^l$
- $P = \{0,1\}^m$
- $C' = \{0,1\}^n, C \subseteq C'$
  
- $E: P \times K \rightarrow C$
- $D: C \times K \rightarrow P$
  
- $\forall p \in P, k \in K: D(E(p,k),k) = p$ 
  - It is *infeasible* to find inversion  $F: P \times C \rightarrow K$

Lets start again!  
This time in English ... .

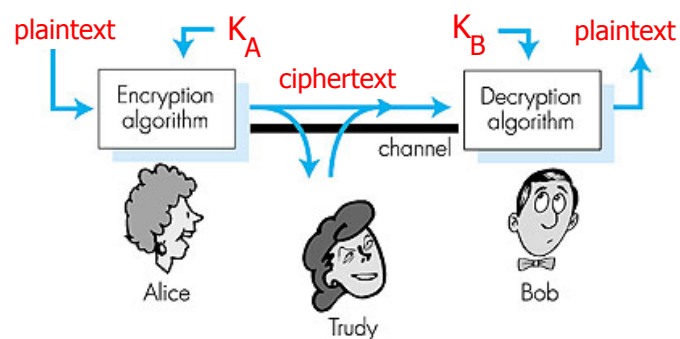
2

## What is a cryptosystem?

- A pair of algorithms that take a **key** and convert **plaintexts** to **ciphertexts** and backwards later
  - **Plaintext**: text to be protected
  - **Ciphertext**: should appear like random
- Requires sophisticated math!
  - Do not try to design your own algorithms!

3

## The language of cryptography



- **Symmetric or secret key crypto**: sender and receiver keys are identical and ***secret***
- **Asymmetric or Public-key crypto**: encrypt key public, decrypt key secret

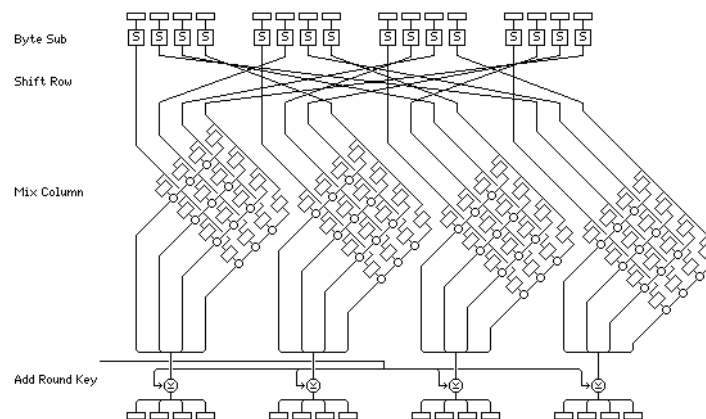
4

## Strength of DES???

- 56-bit keys have  $2^{56} = 7.2 \times 10^{16}$  values
- Brute force search **looked hard in the seventies/eighties**
- Recent advances have shown is possible
  - in 1997 on Internet in a few months
  - in 1998 on dedicated h/w (EFF) in a few days
  - in 1999 above combined in 22hrs!
- Now have several analytic attacks on DES
  - these utilise some deep structure of the cipher
    - by gathering information about encryptions
    - can eventually recover some/all of the sub-key bits
    - if necessary then exhaustively search for the rest
- Generally these are statistical attacks
- Include
  - differential cryptanalysis
  - linear cryptanalysis
  - related key attacks
- Thus, consider alternatives to DES – **AES**

5

## AES – Advanced Encryption Standard



6

## Origins

- ❑ A replacement for DES was needed
  - have theoretical attacks that can break it
  - have demonstrated exhaustive key search attacks
- ❑ Triple-DES possible – but slow, has small blocks
- ❑ US NIST issued a call for ciphers in 1997
- ❑ 15 candidates accepted in Jun 98
- ❑ 5 were shortlisted in Aug-99
- ❑ Rijndael was selected as the AES in Oct-2000
- ❑ Accepted as FIPS PUB 197 standard in Nov-2001

7

## AES Requirements

- ❑ Private key symmetric block cipher
- ❑ 128-bit data, 128/192/256-bit keys
- ❑ Stronger & faster than Triple-DES
- ❑ Active life of 20-30 years (+ archival use)
- ❑ Full open specification & design details
- ❑ Good C, Java, and hw implementations
- ❑ NIST releases all submissions & unclassified analyses

8

## AES Evaluation Criteria

- Initial criteria:
  - security – effort for practical cryptanalysis
  - cost – in terms of computational efficiency
  - algorithm & implementation characteristics
  
- Final criteria
  - general security
  - ease of software & hardware implementation
  - **implementation attacks**
  - flexibility (in en/decrypt, keying, other factors)

9

## AES Shortlist

- After testing and evaluation, shortlist in Aug-99:
  - MARS (IBM) - complex, fast, high security margin
  - RC6 (USA) - v. simple, v. fast, low security margin
  - Rijndael (Belgium) - clean, fast, good security margin
  - Serpent (Euro) - slow, clean, v. high security margin
  - Twofish (USA) - complex, v. fast, high security margin
  
- Then subject to further analysis & comment
  
- Saw contrast between algorithms with
  - few complex rounds verses many simple rounds
  - which refined existing ciphers vs. new proposals

10

## The AES Cipher - Rijndael

- ❑ Designed by Rijmen-Daemen in Belgium
- ❑ Has 128/192/256 bit keys, 128 bit data or block size
- ❑ An **iterative** rather than **feistel** cipher
  - processes data as block of 4 columns of 4 bytes
  - operates on entire data block in every round
- ❑ Designed to be:
  - resistant against known attacks
  - speed and code compactness on many CPUs
  - design simplicity

11

## Rijndael

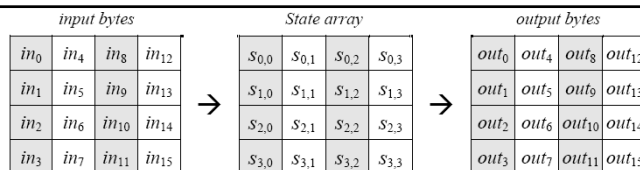
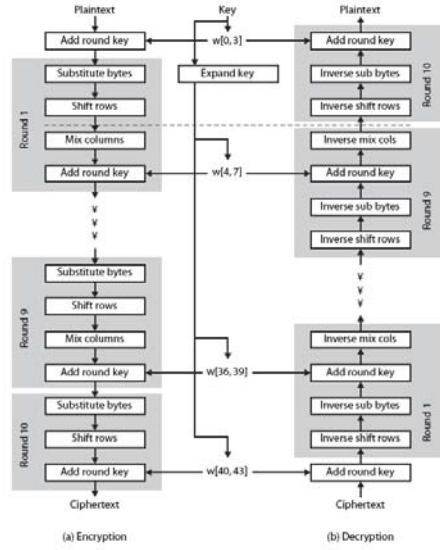


Figure 3. State array input and output.

- ❑ Data block of 4 columns of 4 bytes is state
- ❑ Key is expanded to array of words
- ❑ Has 9/11/13 rounds in which state undergoes:
  - byte substitution (1 S-box used on every byte)
  - shift rows (permute bytes between groups/columns)
  - mix columns (subs using matrix multiply of groups)
  - add round key (XOR state with key material)
  - view as alternating XOR key & scramble data bytes
- ❑ Initial XOR key material & incomplete last round
- ❑ With fast XOR & **table lookup implementation**

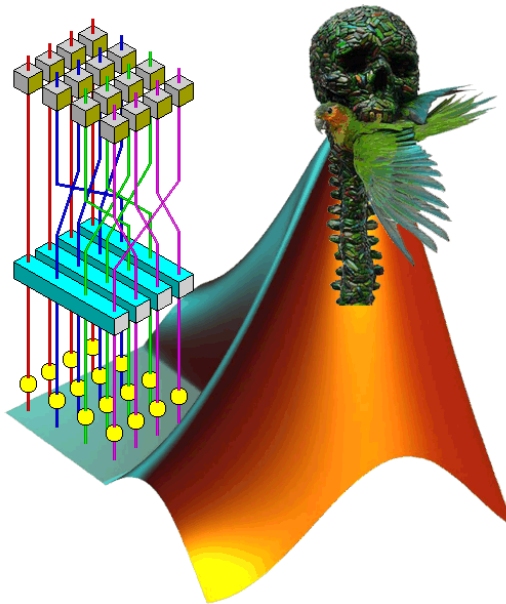
12

# Rijndael



13

# Rijndael



14

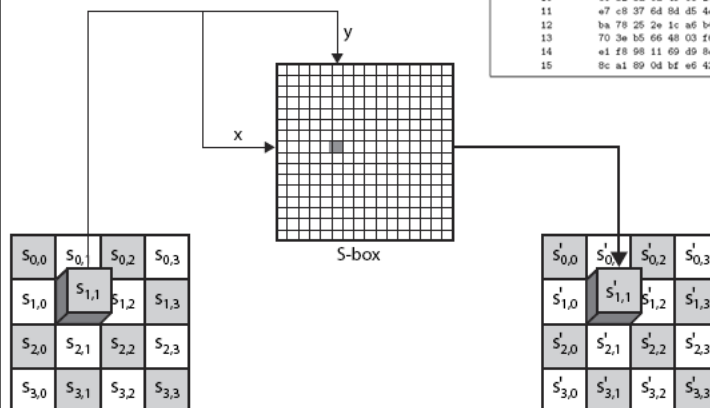
## Byte Substitution

- a simple substitution of each byte
- uses one table of 16x16 bytes containing a permutation of all 256 8-bit values
- each byte of state is replaced by byte indexed by row (left 4-bits) & column (right 4-bits)
  - eg. byte {95} is replaced by byte in row 9 column 5
  - which has value {2A}
- S-box constructed using defined transformation of values in  $GF(2^8)$
- designed to be resistant to all known attacks

15

## Byte Substitution

msb	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	lsb
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76	
1	ca	82	e9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0	
2	b7	f4	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d9	31	15	
3	04	c7	23	e3	18	06	05	9a	07	12	80	e2	ab	27	b2	75	
4	09	83	2c	1a	1b	6e	6a	a0	f2	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	ac	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73	
9	60	81	4f	4c	22	2a	90	88	46	ee	b5	14	de	5a	0b	db	
10	e0	32	3a	0a	49	06	24	5c	c2	43	ac	62	91	96	e4	79	
11	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08	
12	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a	
13	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e	
14	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df	
15	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16	



16

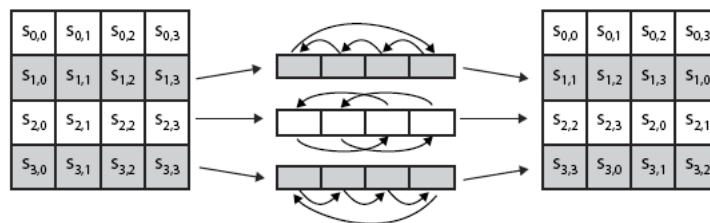


## Shift Rows

- a circular byte shift in each each
  - 1<sup>st</sup> row is unchanged
  - 2<sup>nd</sup> row does 1 byte circular shift to left
  - 3<sup>rd</sup> row does 2 byte circular shift to left
  - 4<sup>th</sup> row does 3 byte circular shift to left
  
- decrypt inverts using shifts to right
  
- since state is processed by columns, this step permutes bytes between the columns

17

## Shift Rows



18

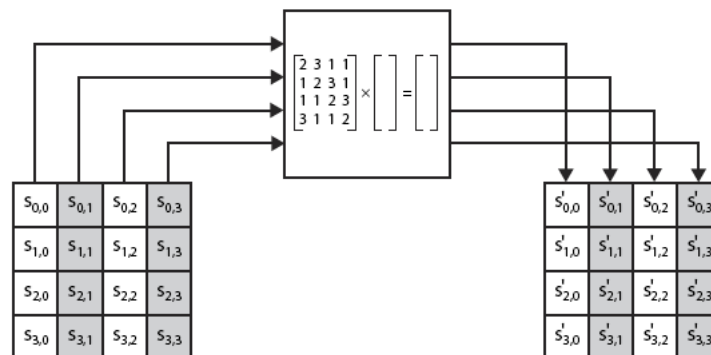
## Mix Columns

- each column is processed separately
- each byte is replaced by a value dependent on all 4 bytes in the column
- effectively a matrix multiplication in  $GF(2^8)$  using prime poly  $m(x) = x^8 + x^4 + x^3 + x + 1$

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

19

## Mix Columns



20

## Mix Columns

- ❑ can express each col as 4 equations
  - to derive each new byte in col
- ❑ decryption requires use of inverse matrix
  - with larger coefficients, hence a little harder
- ❑ have an alternate characterisation
  - each column a 4-term polynomial
  - with coefficients in  $GF(2^8)$
  - and polynomials multiplied modulo  $(x^4+1)$

21

## Add Round Key

- ❑ XOR state with 128-bits of the round key
- ❑ again processed by column (though effectively a series of byte operations)
- ❑ inverse for decryption identical
  - since XOR own inverse, with reversed keys
- ❑ designed to be as simple as possible
  - a form of Vernam cipher on expanded key
  - requires other stages for complexity / security

22

# Add Round Key

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

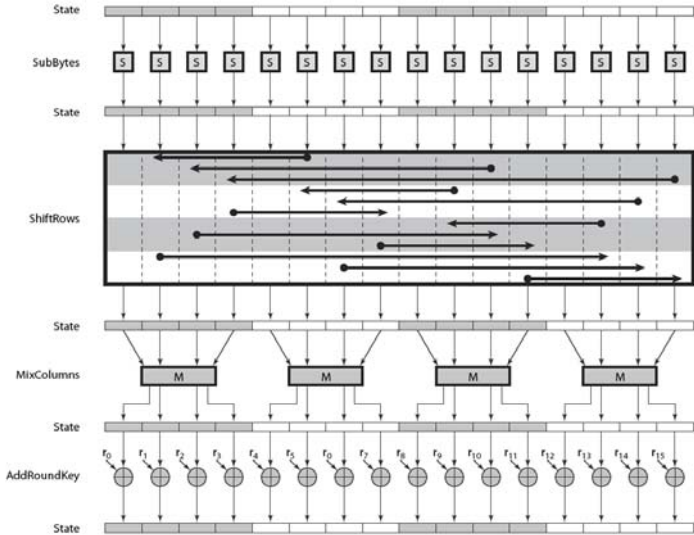
 $\oplus$ 

$w_i$	$w_{i+1}$	$w_{i+2}$	$w_{i+3}$
-------	-----------	-----------	-----------

 $=$ 

$s'_{0,0}$	$s'_{0,1}$	$s'_{0,2}$	$s'_{0,3}$
$s'_{1,0}$	$s'_{1,1}$	$s'_{1,2}$	$s'_{1,3}$
$s'_{2,0}$	$s'_{2,1}$	$s'_{2,2}$	$s'_{2,3}$
$s'_{3,0}$	$s'_{3,1}$	$s'_{3,2}$	$s'_{3,3}$

# AES Round

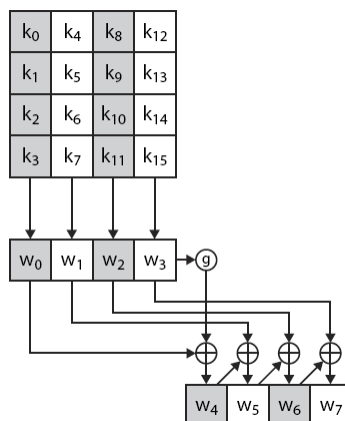


## AES Key Expansion

- takes 128-bit (16-byte) key and expands into array of 44/52/60 32-bit words
- start by copying key into first 4 words
- then loop creating words that depend on values in previous & 4 places back
  - in 3 of 4 cases just XOR these together
  - 1<sup>st</sup> word in 4 has rotate + S-box + XOR round constant on previous, before XOR 4<sup>th</sup> back

25

## AES Key Expansion



26

## Key Expansion Rationale

- ❑ designed to resist known attacks
  
- ❑ design criteria included
  - knowing part key insufficient to find many more
  - invertible transformation
  - fast on wide range of CPU's
  - use round constants to break symmetry
  - diffuse key bits into round keys
  - enough non-linearity to hinder analysis
  - simplicity of description

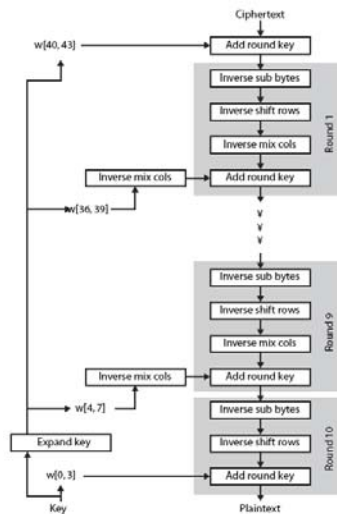
27

## AES Decryption

- ❑ AES decryption is not identical to encryption since steps done in reverse
  
- ❑ but can define an equivalent inverse cipher with steps as for encryption
  - but using inverses of each step
  - with a different key schedule
  
- ❑ works since result is unchanged when
  - swap byte substitution & shift rows
  - swap mix columns & add (tweaked) round key

28

## AES Decryption



29

## Implementation Aspects

- can efficiently implement on 8-bit CPU
  - byte substitution works on bytes using a table of 256 entries
  - shift rows is simple byte shift
  - add round key works on byte XOR's
  - mix columns requires matrix multiply in  $GF(2^8)$  which works on byte values, can be simplified to use table lookups & byte XOR's

30

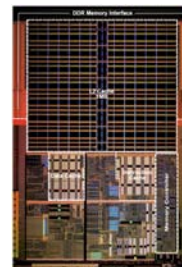
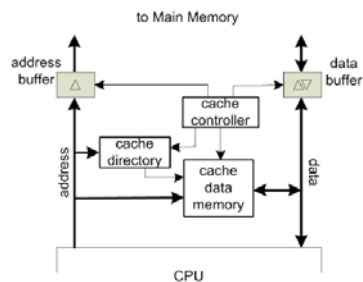
## Implementation Aspects

- ❑ can efficiently implement on 32-bit CPU
  - redefine steps to use 32-bit words
  - can precompute 4 tables of 256-words
  - then each column in each round can be computed using 4 table lookups + 4 XORs
  - at a cost of 4Kb to store tables
  
- ❑ designers believe this very efficient implementation was a key factor in its selection as the AES cipher

31

## Security of AES

- ❑ No crypt-analytical weaknesses for 10 round version
  - Shorter round versions of AES are provably less secure.
- ❑ Optimized implementation easiness produced several
  - Fundamental implementation security issues.
  - Difficulties for **fast and secure implementations** in hw and sw.



32



## Homework suggestion (for next lecture)

- ❑ Please read the following papers:
  - D. J. Bernstein, "Cache-timing attacks on AES," <http://cr.yp.to/antiforgery/cachetiming-20050414.pdf>
  - O. Aciğmez, J.-P. Seifert, and C. Koç, "Micro-Architectural Cryptanalysis," *IEEE Trans. Security and Privacy*, vol. 5, no. 4, Jul.-Aug. 2007, pp. 62-64.

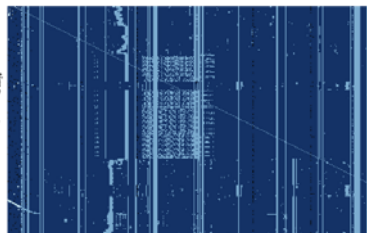


Figure 4: Evolution of the cache *versus* time. Each horizontal line represents the state of the cache lines (represented by a point) at a given time. Different AES encryption are clearly visible in the center. The brighter a point, the longer the time to access its corresponding cache line.

33

## Private-Key Cryptography

- ❑ traditional **private/secret/single key** cryptography uses **one** key
- ❑ shared by both sender and receiver
- ❑ if this key is disclosed communications are compromised
- ❑ also is **symmetric**, parties are equal
- ❑ hence does not protect sender from receiver forging a message & claiming is sent by sender

34

## Public-Key Cryptography

- probably most significant advance in the 3000 year history of cryptography
- uses **two** keys –
  - a public & a private key
- **asymmetric** since parties are **not** equal
- uses clever application of number theoretic concepts to function
- complements **rather than** replaces private key crypto

35

## Why Public-Key Cryptography?

- developed to address two key issues:
  - **key distribution** – how to have secure communications in general without having to trust a KDC with your key
  - **digital signatures** – how to verify a message comes intact from the claimed sender
- public invention due to Whitfield Diffie & Martin Hellman at Stanford Uni in 1976
  - known earlier in classified community

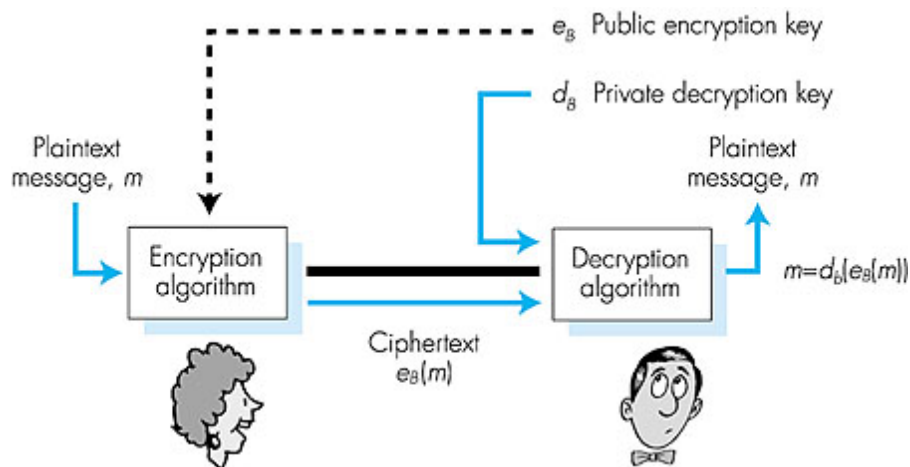
36

## Public-Key Cryptography

- **public-key/two-key/asymmetric** cryptography involves the use of **two** keys:
  - a **public-key**, which may be known by anybody, and can be used to **encrypt messages**, and **verify signatures**
  - a **private-key**, known only to the recipient, used to **decrypt messages**, and **sign** (create) **signatures**
  
- is **asymmetric** because
  - those who encrypt messages or verify signatures **cannot** decrypt messages or create signatures

37

## Public key cryptography



38

## Public-Key Characteristics

- Public-Key algorithms rely on two keys where:
  - it is computationally infeasible to find decryption key knowing only algorithm & encryption key
  - it is computationally easy to en/decrypt messages when the relevant (en/decrypt) key is known
  - either of the two related keys can be used for encryption, with the other used for decryption (for some algorithms)

39

## Public-Key Applications

- can classify uses into 3 categories:
  - **encryption/decryption** (provide secrecy)
  - **digital signatures** (provide authentication)
  - **key exchange** (of session keys)
  
- some algorithms are suitable for all uses, others are specific to one

40

## Security of Public Key Schemes

- ❑ like private key schemes brute force **exhaustive search** attack is always theoretically possible
- ❑ but keys used are too large (>512bits)
- ❑ security relies on a **large enough** difference in difficulty between **easy** (en/decrypt) and **hard** (crypt-analyse) problems
- ❑ more generally the **hard** problem is known, but is made hard enough to be impractical to break
- ❑ requires the use of **very large numbers**
- ❑ hence is **slow** compared to private key schemes

41

## Public Key Cryptography

### Symmetric key crypto

- ❑ Requires sender, receiver to know shared secret key
- ❑ Q: how to agree on key in first place (particularly if never "met")?
- ❑ Q: what if key is stolen?
- ❑ Q: what if you run out of keys?
- ❑ Q: what if A doesn't know she wants to talk to B?

### Public key cryptography

- ❑ Radically different approach [Diffie-Hellman76, RSA78]
- ❑ Sender, receiver do *not* share secret key
- ❑ Encryption key *public* (known to *all*)
- ❑ Decryption key private (known only to receiver)
- ❑ Allows parties to communicate without prearrangement

42

## Prime Numbers

- prime numbers only have divisors of 1 and self
  - they cannot be written as a product of other numbers
  - note: 1 is prime, but is generally not of interest
- eg. 2,3,5,7 are prime, 4,6,8,9,10 are not
- prime numbers are central to number theory
- list of prime number less than 200 is:

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61  
67 71 73 79 83 89 97 101 103 107 109 113 127 131  
137 139 149 151 157 163 167 173 179 181 191 193  
197 199

43

## Relatively Prime Numbers & GCD

- two numbers  $a, b$  are **relatively prime** if have **no common divisors** apart from 1
  - eg. 8 & 15 are relatively prime since factors of 8 are 1,2,4,8 and of 15 are 1,3,5,15 and 1 is the only common factor
- conversely can determine the greatest common divisor by comparing their prime factorizations and using least powers
  - eg.  $300=2^1 \times 3^1 \times 5^2$   $18=2^1 \times 3^2$  hence  $\text{GCD}(18, 300)=2^1 \times 3^1 \times 5^0=6$

44

## Fermat's Theorem

- $a^{p-1} = 1 \pmod{p}$ 
  - where  $p$  is prime and  $\gcd(a, p) = 1$
- also known as Fermat's Little Theorem
- also  $a^p = a \pmod{p}$
- useful in public key and primality testing

45

## Euler Totient Function $\phi(n)$

- when doing arithmetic modulo  $n$
- **complete set of residues** is:  $0 \dots n-1$
- **reduced set of residues** is those numbers (residues) which are relatively prime to  $n$ 
  - eg for  $n=10$ ,
  - complete set of residues is  $\{0,1,2,3,4,5,6,7,8,9\}$
  - reduced set of residues is  $\{1,3,7,9\}$
- number of elements in reduced set of residues is called the **Euler Totient Function  $\phi(n)$**

46

## Euler Totient Function $\phi(n)$

- to compute  $\phi(n)$  need to count number of residues to be excluded
- in general need prime factorization, but
  - for  $p$  ( $p$  prime)  $\phi(p) = p-1$
  - for  $p, q$  ( $p, q$  prime)  $\phi(pq) = (p-1) \times (q-1)$
- eg.
  - $\phi(37) = 36$
  - $\phi(21) = (3-1) \times (7-1) = 2 \times 6 = 12$

47

## Euler's Theorem

- a generalisation of Fermat's Theorem
- $a^{\phi(n)} = 1 \pmod{n}$ 
  - for any  $a, n$  where  $\gcd(a, n) = 1$
- eg.
  - $a=3; n=10; \phi(10)=4;$   
hence  $3^4 = 81 = 1 \pmod{10}$
  - $a=2; n=11; \phi(11)=10;$   
hence  $2^{10} = 1024 = 1 \pmod{11}$

48



## Primitive Roots

- from Euler's theorem have  $a^{\phi(n)} \pmod n = 1$
- consider  $a^m = 1 \pmod n$ ,  $\text{GCD}(a, n) = 1$ 
  - must exist for  $m = \phi(n)$  but may be smaller
  - once powers reach  $m$ , cycle will repeat
- if smallest is  $m = \phi(n)$  then  $a$  is called a **primitive root** or **generating element**
- if  $p$  is prime, then successive powers of  $a$  "generate" the group  $\pmod p$
- these are useful but relatively hard to find

49

## Discrete Logarithms

- the inverse problem to exponentiation is to find the **discrete logarithm** of a number modulo  $p$
- that is to find  $x$  such that  $y = g^x \pmod p$
- this is written as  $x = \log_g y \pmod p$
- if  $g$  is a primitive root then it always exists, otherwise it may not, eg.
  - $x = \log_3 4 \pmod{13}$  has no answer
  - $x = \log_2 3 \pmod{13} = 4$  by trying successive powers
- whilst exponentiation is relatively easy, finding discrete logarithms is generally a **hard** problem

50

## Public-Key distribution of Secret Keys

- ❑ use previous methods to obtain public-key
- ❑ can use for secrecy or authentication
- ❑ but public-key algorithms are slow
- ❑ so usually want to use private-key encryption to protect message contents
- ❑ hence need a session key
- ❑ have several alternatives for negotiating a suitable session

51

## Diffie-Hellman Key Exchange

- ❑ first public-key type scheme proposed
- ❑ by Diffie & Hellman in 1976 along with the exposition of public key concepts
  - note: now known that Williamson (UK CESG) secretly proposed the concept in 1970
- ❑ is a practical method for public exchange of a secret key
- ❑ used in a number of commercial products

52

## Diffie-Hellman Key Exchange

- ❑ a public-key distribution scheme
  - cannot be used to exchange an arbitrary message
  - rather it can establish a common key
  - known only to the two participants
- ❑ value of key depends on the participants (and their private and public key information)
- ❑ based on exponentiation in a finite (Galois) field (modulo a prime or a polynomial) – seems easy at first sight
- ❑ security relies on the difficulty of computing discrete logarithms (similar to factoring) – hard

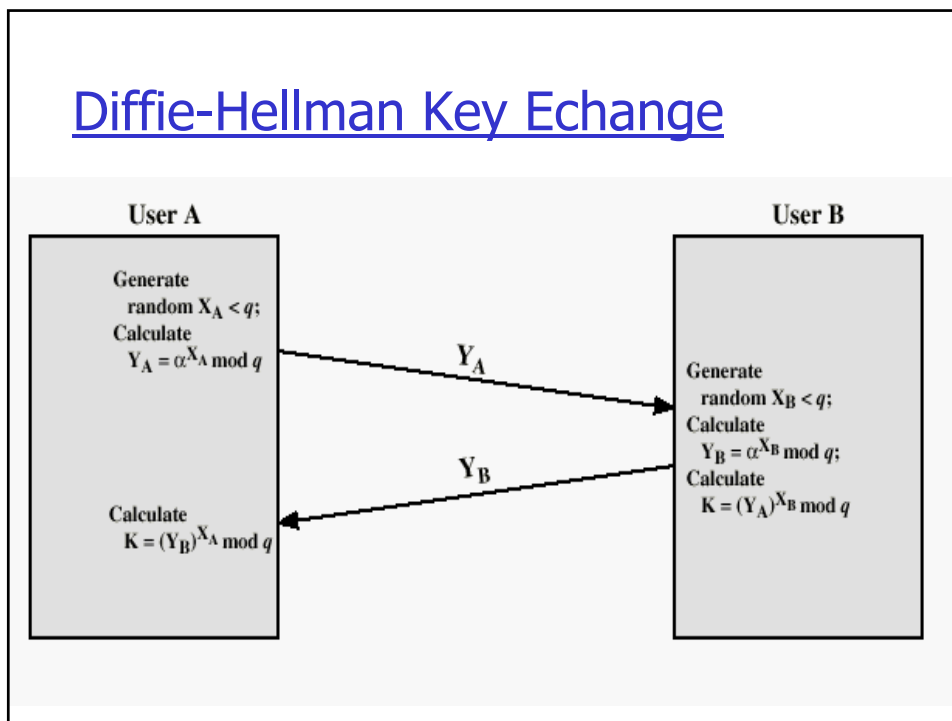
53

## Diffie-Hellman Setup

- ❑ all users agree on global parameters:
  - large prime integer or polynomial  $q$
  - $a$  being a primitive root mod  $q$
- ❑ each user (eg.  $A$ ) generates their key
  - chooses a secret key (number):  $x_A < q$
  - compute their **public key**:  
$$Y_A = a^{x_A} \text{ mod } q$$
- ❑ each user makes public that key  $Y_A$

54

## Diffie-Hellman Key Exchange



## Diffie-Hellman Key Exchange

- shared session key for users A & B is  $K_{AB}$ :

$$\begin{aligned} K_{AB} &= a^{x_A \cdot x_B} \text{ mod } q \\ &= Y_A^{x_B} \text{ mod } q \quad (\text{which } \mathbf{B} \text{ can compute}) \\ &= Y_B^{x_A} \text{ mod } q \quad (\text{which } \mathbf{A} \text{ can compute}) \end{aligned}$$

- $K_{AB}$  is used as session key in private-key encryption scheme between Alice and Bob
- if Alice and Bob subsequently communicate, they will have the **same** key as before, unless they choose new public-keys
- attacker needs an  $x$ , thus must solve discrete log, logarithm modulo  $q$ , i.e., compute  $x_A$  from  $Y_A = a^{x_A}$

56

## Diffie-Hellman Example

- ❑ users Alice & Bob who wish to swap keys:
- ❑ agree on prime  $q=353$  and  $a=3$
- ❑ select random secret keys:
  - A chooses  $x_A=97$ , B chooses  $x_B=233$
- ❑ compute respective public keys:
  - $Y_A=3^{97} \bmod 353 = 40$  (Alice)
  - $Y_B=3^{233} \bmod 353 = 248$  (Bob)
- ❑ compute shared session key as:
  - $K_{AB}=Y_B^{x_A} \bmod 353 = 248^{97} = 160$  (Alice)
  - $K_{AB}=Y_A^{x_B} \bmod 353 = 40^{233} = 160$  (Bob)

57

## Key Exchange Protocols

- ❑ users could create random private/public D-H keys each time they communicate
- ❑ users could create a known private/public D-H key and publish in a directory, then consulted and used to securely communicate with them
- ❑ both of these are vulnerable to a meet-in-the-Middle Attack
- ❑ authentication of the keys is needed
  - Next lectures more on this!

58

## Summary

- Have considered:
  - Details of Rijndael – the AES cipher
  - Principle of Public Key Cryptography
  - Number Theory basics
  - Diffie-Hellmann Key exchange