


PURDUE UNIVERSITY

# Computer Security CS 426

## Lecture 4

### Cryptography Basics - 2



Elisa Bertino  
Purdue University  
IN, USA  
bertino@cs.purdue.edu

Center for Education and Research  
in Information Assurance and Security

PURDUE UNIVERSITY

## Symmetric encryption standards

- **DES** – Data Encryption Standard
  - adopted in 1977 by NBS (now NIST) as FIPS PUB 46
  - controversy over its security
  - Strength of DES: with a 128 bits key, it takes 1018 years to break the cipher
- **TDES** - Triple DES –
  - based on TDEA algorithm (3 keys and 3 different executions of DES)
- **AES** – Advanced Encryption Standard
  - Approved in 2001 as FIPS 197


PURDUE UNIVERSITY

## Symmetric encryption standards

- **IDEA (128-bit keys)**
- **Blowfish (Schneier, up to 448-bit keys)**

PURDUE UNIVERSITY

## Block Ciphers



- Map  $n$ -bit plaintext blocks to  $n$ -bit ciphertext blocks ( $n$ : block length), each of which is then encrypted (ie like a substitution on very big characters - 64-bits or more)
- Main difference with stream ciphers:
  - In stream ciphers the way a character is encrypted depends only on the character itself (and on the position of the character – Vigenere)
  - In block ciphers the way a character is encrypted depends on the other characters in the same block
  - Advantages/disadvantages:
    - Stream ciphers are efficient and have low error propagation; however they have low diffusion and are susceptible to malicious insertions and modifications
    - Block cyphers have high diffusion and are more resistant against malicious insertions and modifications; however they are slower and prone to error propagation

## Block Ciphers Features

- **Block size:** in general larger block sizes mean greater security.
- **Key size:** larger key size means greater security (larger key space).
- **Number of rounds:** multiple rounds offer increasing security.
- **Encryption modes:** define how messages larger than the block size are encrypted, very important for the security of the encrypted message.

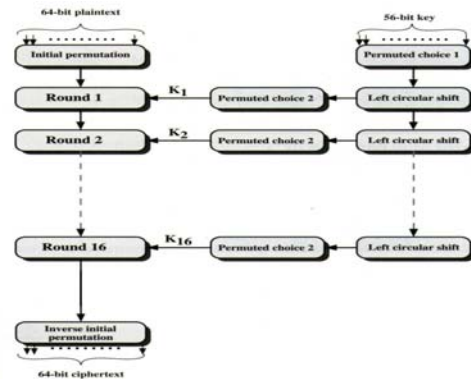
## History of Data Encryption Standard (DES)

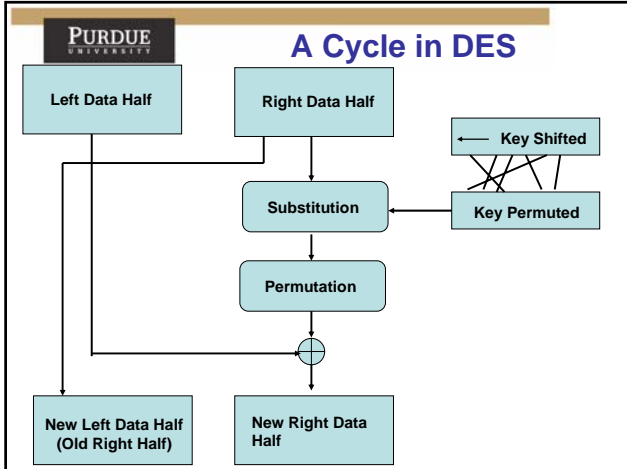
- 1967: Feistel at IBM
  - Lucifer: block size 128; key size 128 bit
- 1972: NBS asks for an encryption standard
- 1975: IBM developed DES (modification of Lucifer)
  - block size 64 bits; key size 56 bits
- 1975: NSA suggests modification
- 1977: NBS adopts DES as encryption standard in (FIPS 46-1, 46-2).
- 2001: NIST adopts Rijndael as replacement to DES.

## DES Features

- **Features:**
  - Block size = 64 bits
  - Key size = 56 bits
  - Number of rounds = 16
  - 16 intermediary keys, each 48 bits

## DES Rounds





### DES – Details Key Scheduling

- Although the input key for DES is 64 bits long, the actual key used by DES is only 56 bits in length.
- The least significant (right-most) bit in each byte is a parity bit, and should be set so that there are always an odd number of 1s in every byte.
- These parity bits are ignored, so only the seven most significant bits of each byte are used, resulting in a key length of 56 bits.
- The first step is to pass the 64-bit key through a permutation called Permuted Choice 1, or PC-1 for short. The table for this is given in next slide.
- Note that in all subsequent descriptions of bit numbers, 1 is the left-most bit in the number, and n is the rightmost bit.

### DES – Details PC-1: Permuted Choice 1

Bit	0	1	2	3	4	5	6
1	57	49	41	33	25	17	9
8	1	58	50	42	34	26	18
15	10	2	59	51	43	35	27
22	19	11	3	60	52	44	36
29	63	55	47	39	31	23	15
36	7	62	54	46	38	30	22
43	14	6	61	53	45	37	29
50	21	13	5	28	20	12	4

### DES – Details Key Scheduling

- Example: use the PC-1 table to determine how bit 30 of the original 64-bit key transforms to a bit in the new 56-bit key.
  - 1) Find the number 30 in the PC-1 table, and notice that it belongs to the column labeled 5 and the row labeled 36.
  - 2) Add up the value of the row and column to find the new position of the bit within the key. For bit 30,  $36 + 5 = 41$ , so bit 30 becomes bit 41 of the new 56-bit key.
- Note that bits 8, 16, 24, 32, 40, 48, 56 and 64 of the original key are not in the table. These are the unused parity bits that are discarded when the final 56-bit key is created.
- Now that we have the 56-bit key, the next step is to use this key to generate 16 48-bit subkeys, called  $K[1]-K[16]$ , which are used in the 16 rounds of DES for encryption and decryption.
- The procedure for generating the subkeys - known as *key scheduling* - has the following steps:
  1. Set the round number R to 1.
  2. Split the current 56-bit key, K, up into two 28-bit blocks, L (the left-hand half) and R (the right-hand half).
  3. Rotate L left by the number of bits specified in the subkey rotation table, and rotate R left by the same number of bits as well.
  4. Join L and R together to get the new K.
  5. Apply the **Permuted Choice 2 (PC-2)** table to K to get the final  $K[R]$ , where R is the round number we are on.
  6. Increment R by 1 and repeat the procedure until we have all 16 subkeys  $K[1]-K[16]$ .

## DES Subkey Rotation Table and PC-2

Subkey Rotation Table

<b>RN</b>	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
<b>NBR</b>	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Bit	0	1	2	3	4	5
1	14	17	11	24	1	5
7	3	28	15	6	21	10
13	23	19	12	4	26	8
19	16	7	27	20	13	2
25	41	52	31	37	47	55
31	30	40	51	45	33	48
37	44	49	39	56	34	53
43	46	42	50	36	29	32

**RN:** round number  
**NBR:** number of bits to rotate

PC-2

## DES Plaintext Preparation

- Once the key scheduling has been performed, the next step is to prepare the plaintext for the actual encryption.
- This is done by passing the plaintext through a permutation called the Initial Permutation, or IP for short.
- This table also has an inverse, called the Inverse Initial Permutation, or  $IP^{-1}$ . Sometimes  $IP^{-1}$  is also called the Final Permutation.

## DES – IP and $IP^{-1}$ Tables

IP

Bit	0	1	2	3	4	5	6	7
1	58	50	42	34	26	18	10	2
9	60	52	44	36	28	20	12	4
17	62	54	46	38	30	22	14	6
25	64	56	48	40	32	24	16	8
33	57	49	41	33	25	17	9	1
41	59	51	43	35	27	19	11	3
49	61	53	45	37	29	21	13	5
57	63	55	47	39	31	23	15	7

$IP^{-1}$

Bit	0	1	2	3	4	5	6	7
1	40	8	48	16	56	24	64	32
9	39	7	47	15	55	23	63	31
17	38	6	46	14	54	22	62	30
25	37	5	45	13	53	21	61	29
33	36	4	44	12	52	20	60	28
41	35	3	43	11	51	19	59	27
49	34	2	42	10	50	18	58	26
57	33	1	41	9	49	17	57	25

## DES – IP and $IP^{-1}$ Tables

- IP and  $IP^{-1}$  are one the inverse of the other
- For example, let's look how bit 32 is transformed under IP. In the table, bit 32 is located at the intersection of the column labeled 4 and the row labeled 25. So this bit becomes bit 29 of the 64-bit block after the permutation.
- Now let's apply  $IP^{-1}$ . In  $IP^{-1}$ , bit 29 is located at the intersection of the column labeled 7 and the row labeled 25. So this bit becomes bit 32 after the permutation. And this is the bit position that we started with before the first permutation.
- If we run a block of plaintext through IP and then pass the resulting block through  $IP^{-1}$ , we obtain with the original block.

## DES Core Function

The 64-bit block of input data, prepared as discussed before, is first split into two halves, L and R. L is the left-most 32 bits, and R is the right-most 32 bits. The following process is repeated 16 times, making up the 16 rounds of standard DES. We call the 16 sets of halves L[0]-L[15] and R[0]-R[15].

1. R[*i*-1] - where *i* is the round number, starting at 1 - is taken and fed into the **E-Bit Selection Table**, which is like a permutation, except that some of the bits are used more than once. This expands the number R[*i*-1] from 32 to 48 bits to prepare for the next step.
2. The 48-bit R[*i*-1] is XORed with K[*i*] and stored in a temporary buffer so that R[*i*-1] is not modified.
3. The result from the previous step is now split into 8 segments of 6 bits each. The left-most 6 bits are B[1], and the right-most 6 bits are B[8]. These blocks form the index into the S-boxes, which are used in the next step. The **Substitution boxes**, known as **S-boxes**, are a set of 8 two-dimensional arrays, each with 4 rows and 16 columns. The numbers in the boxes are always 4 bits in length, so their values range from 0-15. The S-boxes are numbered S[1]-S[8].
4. Starting with B[1], the first and last bits of the 6-bit block are taken and used as an index into the row number of S[1], which can range from 0 to 3, and the middle four bits are used as an index into the column number, which can range from 0 to 15. The number from this position in the S-box is retrieved and stored away. This is repeated with B[2] and S[2], B[3] and S[3], and the others up to B[8] and S[8]. At this point, you now have 8 4-bit numbers, which when strung together one after the other in the order of retrieval, give a 32-bit result.
5. The result from the previous stage is now passed into the P Permutation.
6. This number is now XORed with L[*i*-1], and moved into R[*i*]. R[*i*-1] is moved into L[*i*].
7. At this point we have a new L[*i*] and R[*i*]. Here, we increment *i* and repeat the core function until *i* = 17, which means that 16 rounds have been executed and keys K[1]-K[16] have all been used.

When L[16] and R[16] have been obtained, they are joined back together in the same fashion they were split apart (L[16] is the left-hand half, R[16] is the right-hand half), then the two halves are swapped, R[16] becomes the left-most 32 bits and L[16] becomes the right-most 32 bits of the pre-output block and the resultant 64-bit number is called the pre-output.

## DES S-Boxes - example

- Suppose we have the following 48-bit binary number:  
01110100010111010100011101000011100101101011101
- In order to pass this through steps 3 and 4 of the Core Function as outlined above, the number is split up into 8 6-bit blocks, labeled B[1] to B[8] from left to right:  
011101 000101 110101 000111 101000 011100 101101 011101
- Now, eight numbers are extracted from the S-boxes - one from each box:  
 $B[1] = S[1](01, 1110) = S[1][1][14] = 3 = 0011$   
 $B[2] = S[2](01, 0010) = S[2][1][2] = 4 = 0100$   
 $B[3] = S[3](11, 1010) = S[3][3][10] = 14 = 1110$   
 $B[4] = S[4](01, 0011) = S[4][1][3] = 5 = 0101$   
 $B[5] = S[5](10, 0100) = S[5][2][4] = 10 = 1010$   
 $B[6] = S[6](00, 1110) = S[6][0][14] = 5 = 0101$   
 $B[7] = S[7](11, 0110) = S[7][3][6] = 10 = 1010$   
 $B[8] = S[8](01, 1110) = S[8][1][14] = 9 = 1001$
- In each case of S[*n*][row][column], the first and last bits of the current B[*n*] are used as the row index, and the middle four bits as the column index.
- The results are now joined together to form a 32-bit number which serves as the input to stage 5 of the Core Function (the P Permutation):  
00110100111001011010010110101001

## DES E-Bit Selection Table

Bit	0	1	2	3	4	5
1	32	1	2	3	4	5
7	4	5	6	7	8	9
13	8	9	10	11	12	13
19	12	13	14	15	16	17
25	16	17	18	19	20	21
31	20	21	22	23	24	25
37	24	25	26	27	28	29
43	28	29	30	31	32	11

## DES P Permutation Table

Bit	0	1	2	3
1	16	7	20	21
5	29	12	28	17
9	1	15	23	26
13	5	18	31	10
17	2	8	24	14
21	32	27	3	9
25	19	13	30	6
29	22	11	4	25

### DES Substitution Boxes

Row/column	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S-Box 1

### DES Substitution Boxes

Row/column	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
3	13	8	10	1	4	15	4	2	11	6	7	12	0	5	14	9

S-Box 2

### DES Substitution Boxes

Row/column	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S-Box 3

### DES Substitution Boxes

Row/column	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S-Box 4

### DES Substitution Boxes

Row/column	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S-Box 5

### DES Substitution Boxes

Row/column	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S-Box 6

### DES Substitution Boxes

Row/column	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S-Box 7

### DES Substitution Boxes

Row/column	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

S-Box 8

### DES Ciphertext preparation

- The final step is to apply the permutation  $IP^{-1}$  to the pre-output. The result is the completely encrypted ciphertext.

### DES Decryption

- Decryption uses the same algorithm as encryption, except that the subkeys  $K_1, K_2, \dots, K_{16}$  are applied in reversed order



### Double DES

- DES uses a 56-bit key, this raised concerns (some keys have been derived in two days; the number of keys to try is  $2^{56}$ ) about brute force attacks.
- One proposed solution: double DES.
- Apply DES twice using two keys,  $K_1$  and  $K_2$ .



$$C = E_{K_2} [ E_{K_1} [ P ] ]$$

$$P = D_{K_2} [ D_{K_1} [ C ] ]$$

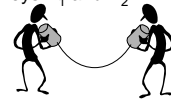
- This leads to a  $2 \times 56 = 112$  bit key, so it is more secure than DES. **Is it?**

### Meet-in-the-Middle Attack

- Goal: given the pair  $(P, C)$  find keys  $K_1$  and  $K_2$ .
- Based on the observation:

$$C = E_{K_2} [ E_{K_1} [ P ] ]$$

$$D_{K_2} [ C ] = E_{K_1} [ P ]$$



- The attack has higher chance of succeeding if another pair  $(P', C')$  is available to the cryptanalysis.

## Meet-in-the-Middle Attack (cont.)

$$C = E_{K_2} [ E_{K_1} [ P ] ]$$

$$E_{K_1} [ P ] = D_{K_2} [ C ]$$

Attack, assumes the attacker knows two pairs (P,C) and (P',C')

- Encrypt P with all  $2^{56}$  possible keys  $K_1$
- Store all pairs  $(K_1, E_{K_1}[P])$ , sorted by  $E_{K_1}[P]$ .
- Decrypt C using all  $2^{56}$  possible keys  $K_2$
- For each decrypted result, check to see if there is a match  $D_{K_2}(C) = E_{K_1}(P)$ .
- If yes, try another pair  $(P', C')$
- If a match is found on the new pair, accept the keys  $K_1$  and  $K_2$ .

## Why Two Pairs (P, C)?

- DES encrypts 64-bit blocks, so for a given plaintext P, there are  $2^{64}$  potential ciphertexts C.
- Key space: two 56-bit key, so there are  $2^{112}$  potential double keys that can map P to C.
- Given a pair (P, C), the number of double keys  $(K_1, K_2)$  that produce  $C = E_{K_2} [ E_{K_1} [ P ] ]$  is at most  $2^{112}/2^{64} = 2^{48}$
- **Therefore, for a pair (P, C),  $2^{48}$  false alarms are expected.**
- With one more pair  $(P', C')$ , extra 64-bit of known text, the alarm rate is  $2^{48}/2^{64} = 1/2^{16}$
- If meet-in-the-middle is performed on two pairs (P, C) and  $(P', C')$ , the correct keys  $K_1$  and  $K_2$  can be determined with probability  $1 - 1/2^{16}$ .
- **Known plaintext attack against double DES succeeds in  $2^{57}$  as opposed to  $2^{56}$  for DES (average).**
- **The 112-bit key provides a security level similar to the 56-bit key.**

## Triple DES

- Use three different keys

$$\text{Encrypt: } C = E_{K_3} [ D_{K_2} [ E_{K_1} [ P ] ] ]$$

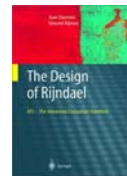
$$\text{Decrypt: } P = D_{K_3} [ E_{K_2} [ D_{K_1} [ C ] ] ]$$

- Key space is  $56 \times 3 = 168$  bits
- No known practical attack against it.



## Rijndael Features

- Designed to be efficient in both hardware and software across a variety of platforms.
- Uses a variable block size, **128, 192, 256-bits**, key size **of 128-, 192-, or 256-bits**.
- 128-bit round key used for each round (Can be pre-computed and cached for future encryptions).
- Note: AES uses a 128-bit block size.
- Variable number of rounds (10, 12, 14):
  - 10 if B = K = 128 bits
  - 12 if either B or K is 192 and the other is  $\leq 192$
  - 14 if either B or K is 256 bits



## Modes of Operation

- ECB (Electronic Code Book)
- CBC (Cipher Block Chaining)
- CFB (Cipher Feedback)
- OFB (Output Feedback)

## Modes of Operation - ECB

- This is the regular DES algorithm
- Data is divided into 64-bit blocks and each block is encrypted one at a time.
- Separate encryptions with different blocks are totally independent of each other. This means that if data is transmitted over a network or phone line, transmission errors will only affect the block containing the error.
- It also means, however, that the blocks can be rearranged, thus scrambling a file beyond recognition, and this action would go undetected.
- ECB is the weakest of the various modes because no additional security measures are implemented besides the basic DES algorithm.
- However, ECB is the fastest and easiest to implement, making it the most common mode of DES seen in commercial applications.

## Modes of Operation - CBC

- In this mode of operation, each block of ECB encrypted ciphertext is XORed with the next plaintext block to be encrypted, thus making all the blocks dependent on all the previous blocks.
  - This means that in order to find the plaintext of a particular block, one needs to know the ciphertext, the key, and the ciphertext for the previous block.
  - The first block to be encrypted has no previous ciphertext, so the plaintext is XORed with a 64-bit number called the *Initialization Vector*, or IV for short.
  - If data is transmitted over and there is a transmission error (adding or deleting bits), the error will be carried forward to all subsequent blocks since each block is dependent upon the last.
  - If the bits are just modified in transit (as is the more common case) the error will only affect all of the bits in the changed block, and the corresponding bits in the following block. The error doesn't propagate any further.
- This mode of operation is more secure than ECB because the extra XOR step adds one more layer to the encryption process.

## Modes of Operation - OFB

- In this mode, blocks of plaintext that are less than 64 bits long can be encrypted.
- Normally, special processing has to be used to handle files whose size is not a perfect multiple of 8 bytes, but this mode removes that necessity
- The plaintext itself is not actually passed through the DES algorithm, but merely XORed with an output block from it as follows:
  - A 64-bit block called the Shift Register is used as the input plaintext to DES. This is initially set to some arbitrary value, and encrypted with the DES algorithm.
  - The ciphertext is then passed through an extra component called the M-box, which simply selects the left-most M bits of the ciphertext, where M is the number of bits in the block we wish to encrypt.
  - This value is XORed with the real plaintext, and the output of that is the final ciphertext.
  - Finally, the ciphertext is fed back into the Shift Register, and used as the plaintext seed for the next block to be encrypted. As with CBC mode, an error in one block affects all subsequent blocks during data transmission. This mode of operation is similar to CBC and is very secure, but it is slower than ECB due to the added complexity.

## Modes of Operation - CFB

- This is similar to CFB mode, except that the ciphertext output of DES is fed back into the Shift Register, rather than the actual final ciphertext.
- The Shift Register is set to an arbitrary initial value, and passed through the DES algorithm.
- The output from DES is passed through the M-box and then fed back into the Shift Register to prepare for the next block.
- This value is then XORed with the real plaintext (which may be less than 64 bits in length, like CFB mode), and the result is the final ciphertext.
- Note that unlike CFB and CBC, a transmission error in one block will not affect subsequent blocks because once the recipient has the initial Shift Register value, it will continue to generate new Shift Register plaintext inputs without any further data input.
- However, this mode of operation is less secure than CFB mode because only the real ciphertext and DES ciphertext output is needed to find the plaintext of the most recent block.

## Asymmetric encryption

- **Asymmetric Encryption (also known as public-key/two-keys cryptography):**
  - uses two keys – a public & a private key
- **It uses clever application of number theory problems**
- **It complements rather than replacing symmetric encryption**

## Asymmetric Cryptography basics

- **Asymmetric/public-key/two-key cryptography uses two keys:**
  - a public-key  $e$ , available to anyone (used to encrypt messages and verify signatures)
  - a private-key  $d$ , known only to the individual (owner), used to decrypt messages, and sign (create) signatures
  - $d \neq e$  and it is “computationally infeasible in practice” to derive  $d$  from  $e$
- **It is called asymmetric because**
  - those who encrypt messages or verify signatures cannot decrypt messages or create signatures
- **Invented by Whitfield Diffie & Martin Hellman at Stanford Univ. in 1976**

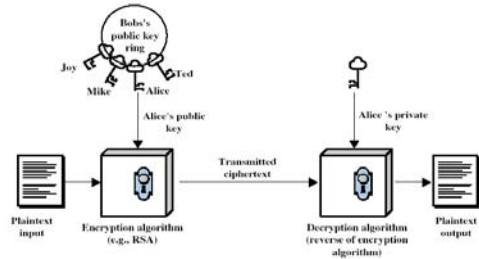
## Asymmetric Cryptography basics

- **Public-Key algorithms rely on two keys such that:**
  - 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

## Asymmetric Cryptography basics

- **One-way functions:**
  - Their computation is easy
  - *Computing the inverse function is computational infeasible*
- **One-way functions examples:**
  - It is easy to compute:  $2^3 = 8$ , as well as  $8 = 2^3$
  - **Given the integer  $x$ , it is easy to compute the function:**
    - $y = f(x) = 543 * x \pmod{21997}$
    - **but given  $y$  5787 it is computational infeasible to compute the value of  $x$  ( $x = f^{-1}(y)$ )**

## Asymmetric Cryptography basics



## Why Public-Key Cryptography?

- **Developed to address two key issues:**
  - **key distribution** – how to have secure communications in general without having to trust a Key Distribution Center with your key
  - **digital signatures** – how to verify a message comes intact from the claimed sender

## Asymmetric encryption – applications

- **encryption/decryption** (provides confidentiality)
- **digital signatures** (provides integrity/authentication)
- **key exchange** (of session keys)

## Security of Asymmetric encryption

- Like shared key schemes, brute force exhaustive search attack is always theoretically possible
  - but keys used are very large (>512bits)
- Security relies on a large enough difference in difficulty between easy (en/decrypt) and hard (cryptanalysis) problems
- More generally the hard problem is known, but it is too hard to solve in practice

## Public-key encryption algorithm: RSA

- RSA was defined by Rivest, Shamir & Adleman of MIT in 1977
- best known & widely used public-key scheme
- based on exponentiation in a finite (Galois) field over integers modulo a prime:
  - nb. exponentiation takes  $O((\log n)^3)$  operations (easy)
- uses large integers (eg. 1024 bits)
- security due to cost of factoring large numbers
  - nb. factorization takes  $O(e^{\log n \log \log n})$  operations (hard)

## Additional Resources

- Stallings, W. Cryptography and Network Security. Upper Saddle River, NJ: Prentice Hall 2006
- NIST Computer Security Resource Center: Cryptographic Toolkit  
<http://csrc.nist.gov/CryptoToolkit/tken/ncryption.html>