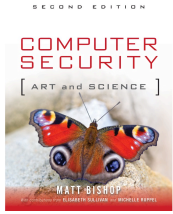
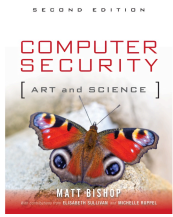


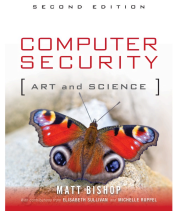
Basic Cryptography

Chapter 10



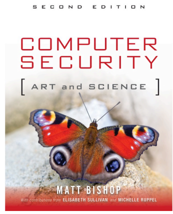
Overview

- Symmetric cryptography
  - Cæsar and Vigènere ciphers
  - DES, AES
- Public key (asymmetric) cryptography
  - El Gamal, RSA
  - Elliptic ciphers
- Cryptographic Checksums
  - HMAC
- Digital signatures



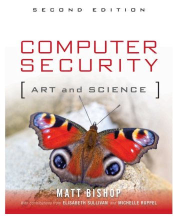
Cryptosystem

- Quintuple ($\mathcal{E}$, $\mathcal{D}$, $\mathcal{M}$, $\mathcal{K}$, $\mathcal{C}$)
  - $\mathcal{M}$ set of plaintexts
  - $\mathcal{K}$ set of keys
  - $\mathcal{C}$ set of ciphertexts
  - $\mathcal{E}$ set of encryption functions e: $\mathcal{M} \times \mathcal{K} \rightarrow \mathcal{C}$
  - $\mathcal{D}$ set of decryption functions d: $\mathcal{C} \times \mathcal{K} \rightarrow \mathcal{M}$



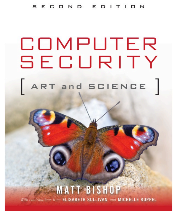
Example

- Example: Cæsar cipher
  - $\mathcal{M}$ = { sequences of letters }
  - $\mathcal{K} = \{ i \mid i$ is an integer and $0 \leq i \leq 25 \}$
  - $\mathcal{E} = \{ E_k \mid k \in \mathcal{K}$ and for all letters m, $E_k(m) = (m + k) \bmod 26 \}$
  - $\mathcal{D} = \{ D_k \mid k \in \mathcal{K}$ and for all letters c, $D_k(c) = (26 + c - k) \bmod 26 \}$
  - $\mathcal{C} = \mathcal{M}$



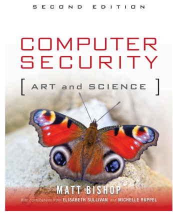
Attacks

- Opponent whose goal is to break cryptosystem is the *adversary*
  - Assume adversary knows algorithm used, but not key
- Three types of attacks:
  - *ciphertext only*: adversary has only ciphertext; goal is to find plaintext, possibly key
  - *known plaintext*: adversary has ciphertext, corresponding plaintext; goal is to find key
  - *chosen plaintext*: adversary may supply plaintexts and obtain corresponding ciphertext; goal is to find key



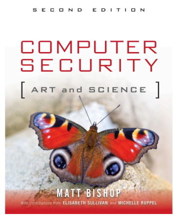
Basis for Attacks

- Mathematical attacks
  - Based on analysis of underlying mathematics
- Statistical attacks
  - Make assumptions about the distribution of letters, pairs of letters (digrams), triplets of letters (trigrams), *etc.*
    - Called *models of the language*
  - Examine ciphertext, correlate properties with the assumptions.



Symmetric Cryptography

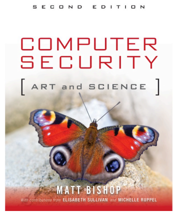
- Sender, receiver share common key
  - Keys may be the same, or trivial to derive from one another
  - Sometimes called *secret key cryptography*
- Two basic types
  - Transposition ciphers
  - Substitution ciphers
  - Combinations are called *product ciphers*



Transposition Cipher

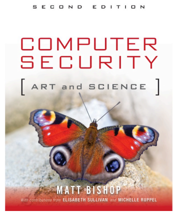
- Rearrange letters in plaintext to produce ciphertext
- Example (Rail-Fence Cipher)
  - Plaintext is `HELLO WORLD`
  - Rearrange as

    ```
    HLOOL
    ELWRD
    ```
  - Ciphertext is `HLOOL ELWRD`



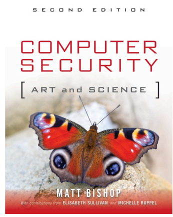
Attacking the Cipher

- Anagramming
  - If 1-gram frequencies match English frequencies, but other *n*-gram frequencies do not, probably transposition
  - Rearrange letters to form *n*-grams with highest frequencies



Example

- Ciphertext: HLOOLELWRD
- Frequencies of 2-grams beginning with H
  - HE 0.0305
  - HO 0.0043
  - HL, HW, HR, HD < 0.0010
- Frequencies of 2-grams ending in H
  - WH 0.0026
  - EH, LH, OH, RH, DH ≤ 0.0002
- Implies E follows H

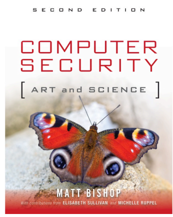


Example

- Arrange so the H and E are adjacent

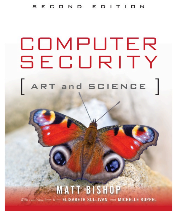
```
HE
LL
OW
OR
LD
```

- Read across, then down, to get original plaintext



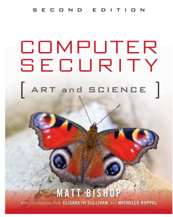
Substitution Ciphers

- Change characters in plaintext to produce ciphertext
- Example (Caesar cipher)
  - Plaintext is `HELLO WORLD`
  - Change each letter to the third letter following it (`X` goes to `A`, `Y` to `B`, `Z` to `C`)
    - Key is 3, usually written as letter ‘`D`’
  - Ciphertext is `KHOOR ZRUOG`



Attacking the Cipher

- Exhaustive search
  - If the key space is small enough, try all possible keys until you find the right one
  - Caesar cipher has 26 possible keys
- Statistical analysis
  - Compare to 1-gram model of English

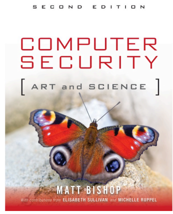


Statistical Attack

- Compute frequency of each letter in ciphertext:

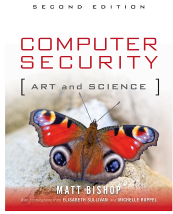
G	0.1	H	0.1	K	0.1	O	0.3
R	0.2	U	0.1	Z	0.1		

- Apply 1-gram model of English
  - Frequency of characters (1-grams) in English is on next slide



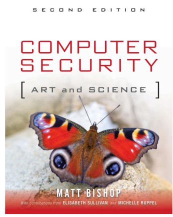
Character Frequencies

a	0.07984	h	0.06384	n	0.06876	t	0.09058
b	0.01511	i	0.07000	o	0.07691	u	0.02844
c	0.02504	j	0.00131	p	0.01741	v	0.01056
d	0.04260	k	0.00741	q	0.00107	w	0.02304
e	0.12452	l	0.03961	r	0.05912	x	0.00159
f	0.02262	m	0.02629	s	0.06333	y	0.02028
g	0.02013					z	0.00057



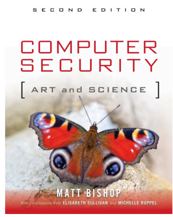
Statistical Analysis

- $f(c)$ frequency of character c in ciphertext
- $\varphi(i)$ correlation of frequency of letters in ciphertext with corresponding letters in English, assuming key is i
    - $\varphi(i) = \Sigma_{0 \le c \le 25} f(c)p(c - i)$ so here,
      $\varphi(i) = 0.1\ p(6 - i) + 0.1\ p(7 - i) + 0.1\ p(10 - i) + 0.3\ p(14 - i) + 0.2\ p(17 - i) + 0.1\ p(20 - i) + 0.1\ p(25 - i)$
        - $p(x)$ is frequency of character x in English



Correlation: $\varphi(i)$ for $0 \le i \le 25$

i	$\varphi(i)$	i	$\varphi(i)$	i	$\varphi(i)$	i	$\varphi(i)$
0	0.0469	7	0.0461	13	0.0505	19	0.0312
1	0.0393	8	0.0194	14	0.0561	20	0.0287
2	0.0396	9	0.0286	15	0.0215	21	0.0526
3	0.0586	10	0.0631	16	0.0306	22	0.0398
4	0.0259	11	0.0280	17	0.0386	23	0.0338
5	0.0165	12	0.0318	18	0.0317	24	0.0320
6	0.0676					25	0.0443

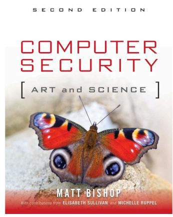


The Result

- Most probable keys, based on φ:
  - $i = 6$, $\varphi(i) = 0.0676$
    - plaintext `EBIIL TLOLA`
  - $i = 10$, $\varphi(i) = 0.0631$
    - plaintext `AXEEH PHKEW`
  - $i = 14$, $\varphi(i) = 0.0561$
    - plaintext `WTAAD LDGAS`
  - $i = 3$, $\varphi(i) = 0.0586$
    - plaintext `HELLO WORLD`
- Only English phrase is for $i = 3$
  - That's the key (3 or 'D')

Caesar's Problem

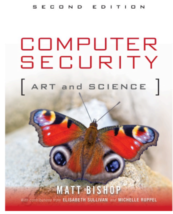
- Key is too short
  - Can be found by exhaustive search
  - Statistical frequencies not concealed well
    - They look too much like regular English letters
- So make it longer
  - Multiple letters in key
  - Idea is to smooth the statistical frequencies to make cryptanalysis harder



Vigènere Cipher

- Like Caesar cipher, but use a phrase
  - So it's effectively multiple Caesar ciphers
- Example
  - Message `A LIMERICK PACKS LAUGHS ANATOMICAL`
  - Key `BENCH`
  - Encipher using Caesar cipher for each letter:

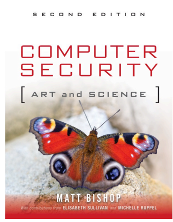
key	`BENCHBENCHBENCHBENCHBENCHBENCH`
plain	`ALIMERICKPACKSLAUGHSANATOMICAL`
cipher	`BPVOLSMPMWBGXUSBYTJZBRNVVNMPCS`



Relevant Parts of Tableau

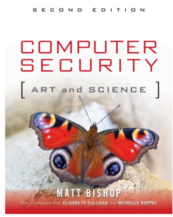
	B	*C*	*E*	*H*	*N*
A	B	C	E	H	N
C	D	E	G	J	P
E	F	G	I	L	R
G	H	I	K	N	T
H	I	J	L	O	U
I	J	K	M	P	V
K	L	M	O	R	X
L	M	N	P	S	Y
M	N	O	Q	T	Z
N	O	P	R	U	A
O	P	Q	S	V	B
P	Q	R	T	W	C
R	S	T	V	Y	E
S	T	U	W	Z	F
T	U	V	X	A	G
U	V	W	Y	B	H

- Tableau shown has relevant rows, columns only
  - Columns correspond to letters from the key
  - Rows correspond to letters from the message
- Example encipherments:
  - key `B`, letter `R`: follow `B` column down to `R` row (giving "`S`")
  - Key `H`, letter `L`: follow `H` column down to `L` row (giving "`S`")



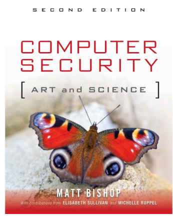
Useful Terms

- *period*: length of key
  - In earlier example, period is 3
- *tableau*: table used to encipher and decipher
  - Vigènere cipher has key letters on top, plaintext letters on the left
- *polyalphabetic*: the key has several different letters
  - Caesar cipher is monoalphabetic



Attacking the Cipher

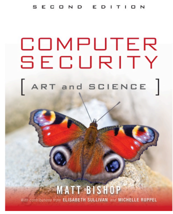
- Approach
  - Establish period; call it *n*
  - Break message into *n* parts, each part being enciphered using the same key letter
  - Solve each part; you can leverage one part from another
- We will show each step



The Target Cipher

- We want to break this cipher:

```
ADQYS MIUSB OXKKT MIBHK IZOOO EQOOG IFBAG KAUMF
VVTAA CIDTW MOCIO EQOOG BMBFV ZGGWP CIEKQ HSNEW
VECNE DLAAV RWKXS VNSVP HCEUT QOIOF MEGJS WTPCH
AJMOC HIUIX
```

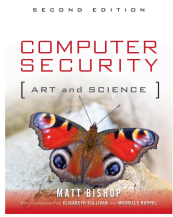


Establish Period

- Kaskski: *repetitions in the ciphertext occur when characters of the key appear over the same characters in the plaintext*
- Example:

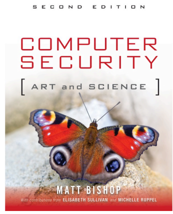
```
key    VIGVIGVIGVIGVIGV
plain  THEBOYHASTHEBALL
cipher OPKWWECIYOPKWIRG
```

Note the key and plaintext line up over the repetitions (underlined). As distance between repetitions is 9, the period is a factor of 9 (that is, 1, 3, or 9)



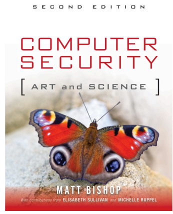
Repetitions in Example

Letters	Start	End	Gap Length	Gap Length Factors
OEQOOG	24	54	30	2, 3, 5
MOC	50	122	72	2, 2, 2, 3, 3



Estimate of Period

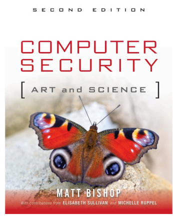
- OEQOOG is probably not a coincidence
  - It’s too long for that
  - Period may be 1, 2, 3, 5, 6, 10, 15, or 30
- MOC is also probably not a coincidence
  - Period may be 1, 2, 3, 4, 6, 8, 9, 12, 18, 24, 36, or 72
- Period of 2 or 3 is probably too short (but maybe not)
- Begin with period of 6



Check on Period

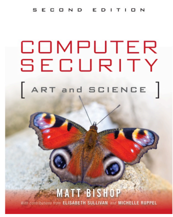
- Index of coincidence is probability that two randomly chosen letters from ciphertext will be the same
- Tabulated for different periods:

1	0.0660
2	0.0520
3	0.0473
6	0.0427



Compute IC for an Alphabet

- IC = $[n\ (n - 1)]^{-1} \sum_{0 \le i \le 25} [F_i\ (F_i - 1)]$
  - where n is length of ciphertext and F_i the number of times character i occurs in ciphertext
- For the given ciphertext, IC = 0.0433
  - Indicates a key of length 5 or 6
  - A statistical measure, so it can be in error, but it agrees with the previous estimate (which was 6)



Splitting Into Alphabets

alphabet 1: `AIKHOIATTOBGEEERNEOSAI`

alphabet 2: `DUKKEFUAWEMGKWDWSUFWJU`

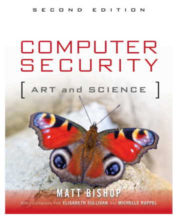
alphabet 3: `QSTIQBMAMQBWQVLKVTMTMI`

alphabet 4: `YBMZOAFCOOFPHEAXPQEPOX`

alphabet 5: `SOIOOGVICOVCSVASHOGCC`

alphabet 6: `MXBOGKVDIGZINNVVCIJHH`

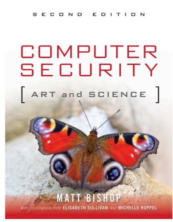
- ICs (#1, 0.0692; #2, 0.0779; #3, 0.0779; #4, 0.0562; #5, 0.1238; #6, 0.0429) indicate all alphabets have period 1, except #4 (between 1 and 2) and #6 (between 5 and 6); assume statistical variance



Frequency Examination

#	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	3	1	0	0	4	0	1	1	3	0	1	0	0	1	3	0	0	1	1	2	0	0	0	0	0	0
2	1	0	0	2	2	2	1	0	0	1	3	0	1	0	0	0	0	0	1	0	4	0	4	0	0	0
3	1	2	0	0	0	0	0	0	2	0	1	1	4	0	0	0	4	0	1	3	0	2	1	0	0	0
4	2	1	1	0	2	2	0	1	0	0	0	0	1	0	4	3	1	0	0	0	0	0	0	2	1	1
5	1	0	5	0	0	0	2	1	2	0	0	0	0	0	5	0	0	0	3	0	0	2	0	0	0	0
6	0	1	1	1	0	0	2	2	3	1	1	0	1	2	1	0	0	0	0	0	0	3	0	1	0	1
	H	M	M	M	H	M	M	H	H	M	M	M	M	H	H	M	L	H	H	H	M	L	L	L	L	L

The last row has general letter frequencies (H high, M medium, L low)



Begin Decryption

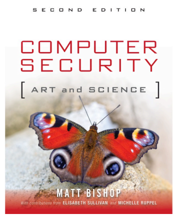
- First matches characteristics of unshifted alphabet
- Third matches if `I` shifted to `A`
- Sixth matches if `V` shifted to `A`
- Substitute into ciphertext (bold are substitutions)

AD**I**YS **RI**U**K**B O**CK**K**L** MI**GH**K **A**ZO**TO** E**I**OO**L** **I**F**T**AG

PAU**E**F V**AT**A**S** CI**IT**W **E**OC**NO** E**I**OO**L** **B**M**T**FV **EG**G**O**P

C**NE**K**I** HS**SE**W **N**EC**SE** D**D**AA**A** **R**W**C**XS **AN**S**N**P H**HE**U**L**

QO**NO**F **E**EG**OS** W**L**PC**M** **A**J**E**OC **MI**U**A**X



Look For Clues

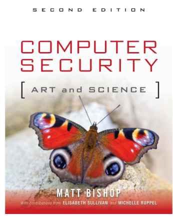
- **A**J**E** in last line suggests "are", meaning second alphabet maps `A` into `S`:

  ALIYS **RICK**B O**CKSL** MI**GHS** **A**ZO**TO** **MI**OO**L** **INT**AG

  PACEF V**ATIS** CI**ITE** **E**OC**NO** **MI**OO**L** **BUT**FV **EGOO**P

  C**NESI** HS**SEE** **N**EC**SE** **LD**AA**A** **REC**XS **ANAN**P H**HECL**

  QO**NON** **E**EG**OS** **EL**PC**M** **ARE**OC **MICA**X



Next Alphabet

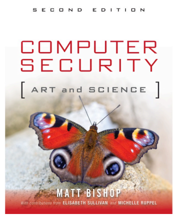
- **MICA**X in last line suggests "mical" (a common ending for an adjective), meaning fourth alphabet maps O into A:

ALIMS **RICKP** O**CKSL** **A**I**GHS** **AN**O**TO** **MIC**O**L** **INTO**G

PACET V**ATIS** **Q**I**ITE** **EC**C**NO** **MIC**O**L** **BUTT**V **EGOOD**

C**NESI** **V**S**SEE** **NS**C**SE** **LDO**A**A** **RECL**S **ANAND** H**HECL**

EO**NON** **ES**G**OS** **ELD**C**M** **AREC**C **MICAL**



Got It!

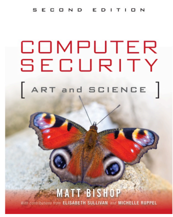
- **QI** means that U maps into I, as Q is always followed by U:

  ALIME RICKP ACKSL AUGHS ANATO MICAL INTOS

  PACET HATIS QUITE ECONO MICAL BUTTH EGOOD

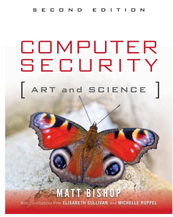
  ONESI VESEE NSOSE LDOMA RECLE ANAND THECL

  EANON ESSOS ELDOM ARECO MICAL



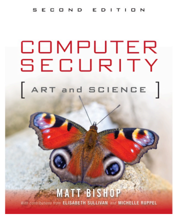
One-Time Pad

- A Vigenère cipher with a random key at least as long as the message
  - Provably unbreakable
  - Why? Look at ciphertext `DXQR`. Equally likely to correspond to plaintext `DOIT` (key `AJIY`) and to plaintext `DONT` (key `AJDY`) and any other 4 letters
  - Warning: keys *must* be random, or you can attack the cipher by trying to regenerate the key
    - Approximations, such as using pseudorandom number generators to generate keys, are *not* random



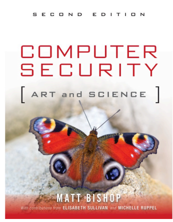
Overview of the DES

- A block cipher:
  - encrypts blocks of 64 bits using a 64 bit key
  - outputs 64 bits of ciphertext
- A product cipher
  - basic unit is the bit
  - performs both substitution and transposition (permutation) on the bits
- Cipher consists of 16 rounds (iterations) each with a 48 bit round key generated from the user-supplied key



Structure of the DES

- Input is first permuted, then split into left half (L) and right half (R), each 32 bits
- Round begins; R and round key run through function *f*, then xor'ed with L
  - *f* expands R to 48 bits, xors with round key, and then each 6 bits of this are run through S-boxes (substitution boxes), each of which gives 4 bits of output
  - Those 32 bits are permuted and this is the output of *f*
- R and L swapped, ending the round
  - Swapping does not occur in the last round
- After last round, L and R combined, permuted, forming DES output

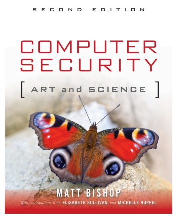


Controversy

- Considered too weak
  - Diffie, Hellman said in a few years technology would allow DES to be broken in days
    - Design using 1999 technology published
- Design decisions not public
  - S-boxes may have backdoors

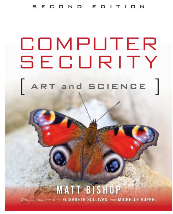
Undesirable Properties

- 4 weak keys
  - They are their own inverses
- 12 semi-weak keys
  - Each has another semi-weak key as inverse
- Complementation property
  - $\text{DES}_k(m) = c \Rightarrow \text{DES}_{k'}(m') = c'$
- S-boxes exhibit irregular properties
  - Distribution of odd, even numbers non-random
  - Outputs of fourth box depends on input to third box



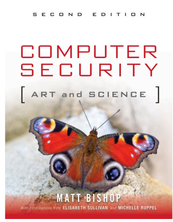
Differential Cryptanalysis

- A chosen ciphertext attack
  - Requires 2^{47} plaintext, ciphertext pairs
- Revealed several properties
  - Small changes in S-boxes reduced the number of pairs needed
  - Making every bit of the round keys independent did not impede attack
- Linear cryptanalysis improves result
  - Requires 2^{43} plaintext, ciphertext pairs



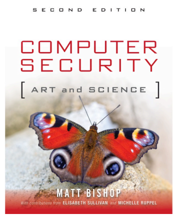
DES Modes

- Electronic Code Book Mode (ECB)
  - Encipher each block independently
- Cipher Block Chaining Mode (CBC)
  - Xor each block with previous ciphertext block
  - Requires an initialization vector for the first one
- Encrypt-Decrypt-Encrypt (2 keys: k, k')
  - $c = \mathrm{DES}_k(\mathrm{DES}_{k'}^{-1}(\mathrm{DES}_k(m)))$
- Triple DES(3 keys: k, k', k'')
  - $c = \mathrm{DES}_k(\mathrm{DES}_{k'}(\mathrm{DES}_{k''}(m)))$



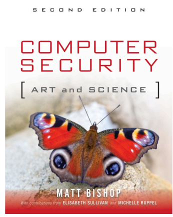
Current Status of DES

- Design for computer system, associated software that could break any DES-enciphered message in a few days published in 1998
- Several challenges to break DES messages solved using distributed computing
- NIST selected Rijndael as Advanced Encryption Standard, successor to DES
    - Designed to withstand attacks that were successful on DES
- DES officially withdrawn in 2005



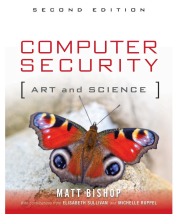
Advanced Encryption Standard

- Competition announces in 1997 to select successor to DES
  - Successor needed to be available for use without payment (no royalties, etc.)
  - Successor must encipher 128-bit blocks with keys of lengths 128, 192, and 256
- 3 workshops in which proposed successors were presented, analyzed
- Rijndael selected as successor to DES, called the Advanced Encryption Standard (AES
  - Other finalists were Twofish, Serpent, RC6, MARS



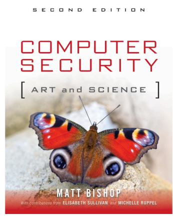
Overview of the AES

- A block cipher:
  - encrypts blocks of 128 bits using a 128, 192, or 256 bit key
  - outputs 128 bits of ciphertext
- A product cipher
  - basic unit is the bit
  - performs both substitution and transposition (permutation) on the bits
- Cipher consists of rounds (iterations) each with a round key generated from the user-supplied key
  - If 128 bit key, then 10 rounds
  - If 192 bit key, then 12 rounds
  - If 256 bit key, then 14 rounds



Structure of the AES: Encryption

- Input placed into a state array, which is then combined with zeroth round key
  - Treat state array as a 4x4 matrix, each entry being a byte
- Round begins; new values substituted for each byte of the state array
- Rows then cyclically shifted
- Each column independently altered
  - Not done in last round
- Result xor’ed with round key
- After last round, state array is the encrypted input

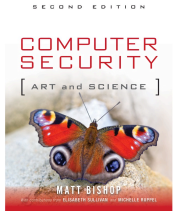


Structure of the AES: Decryption

- Round key schedule reversed
- Input placed into a state array, which is then combined with zeroth round key (of reversed schedule)
- Round begins; rows cyclically shifted, then new values substituted for each byte of the state array
  - Inverse rotation, substitution of encryption
- Result xor’ed with round key (of reversed schedule)
- Each column independently altered
  - Inverse of encryption; this is not done in last round
- After last round, state array is the decrypted input

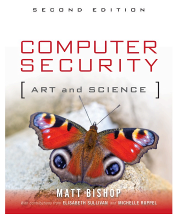
Analysis of AES

- Designed to withstand attacks that the DES is vulnerable to
- All details of design made public, unlike with the DES
  - In particular, those of the substitutions (S-boxes) were described
- After 2 successive rounds, every bit in the state array depends an every bit in the state array 2 rounds ago
- No weak, semi-weak keys



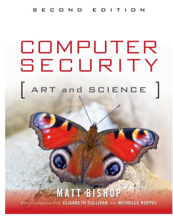
AES Modes

- DES modes also work with AES
- EDE and "Triple-AES" not used
    - Extended block size makes this unnecessary
- New counter mode CTR added



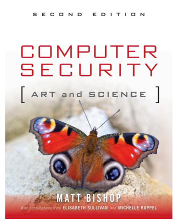
Public Key Cryptography

- Two keys
  - *Private key* known only to individual
  - *Public key* available to anyone
    - Public key, private key inverses
- Idea
  - Confidentiality: encipher using public key, decipher using private key
  - Integrity/authentication: encipher using private key, decipher using public one



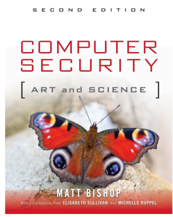
Requirements

1. It must be computationally easy to encipher or decipher a message given the appropriate key
2. It must be computationally infeasible to derive the private key from the public key
3. It must be computationally infeasible to determine the private key from a chosen plaintext attack



El Gamal Cryptosystem

- Based on discrete logarithm problem
  - Given integers *n*, *g*, and *b* with $0 \le a < n$ and $0 \le b < n$; then find an integer *k* such that $0 \le k < n$ and $a = g^k \bmod n$
  - Choose *n* to be a prime *p*
  - Solutions known for small *p*
  - Solutions computationally infeasible as *p* grows large

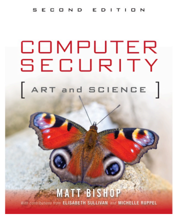


Algorithm

- Choose prime p with $p - 1$ having a large factor
- Choose generator g such that $1 < g < p$
- Choose k_{priv} such that $1 < k_{priv} < p - 1$
- Set $y = g^{k_{priv}} \bmod p$
- Then public key $k_{pub} = (p, g, y)$ and private key is k_{priv}

Example

- Alice: p = 262643; g = 9563, k_{priv} = 3632
  - 262643 = 2 x 131321, also prime
- Alice's public key k_{pub} = (262643, 9563, 27459)
  - As $y = g^{k_{priv}} \bmod p = 9563^{3632} \bmod 262643 = 27459$



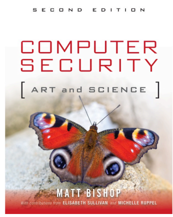
Enciphering and Deciphering

Encipher message m:

- Choose random integer k relatively prime to $p - 1$
- Compute $c_1 = g^k \bmod p$; $c_2 = my^k \bmod p$
- Ciphertext is $c = (c_1, c_2)$

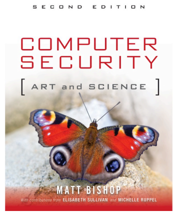
Decipher ciphertext (c_1, c_2)

- Compute $m = c_2 c_1^{-k_{priv}} \bmod p$
- Message is m



Example Encryption

- Bob wants to send Alice PUPPIESARESMALL
- Message to send: 152015 150804 180017 041812 001111
- First block: choose $k = 5$
  - $c_{1,1} = 9563^5 \bmod 262643 = 15653$
  - $c_{1,2} = (152015)27459^5 \bmod 262643 = 923$
- Next block: choose $k = 3230$
  - $c_{2,1} = 9563^{3230} \bmod 262643 = 46495$
  - $c_{2,2} = (150804)27459^{3230} \bmod 262643 = 109351$
- Continuing, enciphered message is (15653,923), (46495,109351), (176489,208811), (88247,144749), (152432,5198)



Example Decryption

Alice receives (15653,923), (46495,109351), (176489,208811), (88247,144749), (152432,5198)

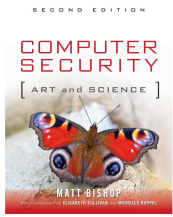
- First block: $(923)15653^{-3632} \bmod 262643 = 152015$
- Second block: $(109351)46495^{-3632} \bmod 262643 = 150804$
- Third block: $(208811)176489^{-3632} \bmod 262643 = 180017$
- Fourth block: $(144749)\ 88247^{-3632} \bmod 262643 = 41812$
- Fifth block: $(5198)\ 152432^{-3632} \bmod 262643 = 1111$

So the message is 152015 150804 180017 041812 001111

- Which translates to "PUP PIE SAR ESM ALL" or PUPPIESARESMALL

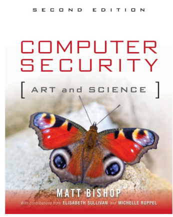
Notes

- Same letter enciphered twice produces two different ciphertexts
  - Defeats replay attacks
- If the integer *k* is used twice, and an attacker has plaintext for one of those messages, deciphering the other is easy
- c_2 linear function of *m*, so forgery possible
  - *m* message, (c_1, c_2) ciphertext; then (c_1, nc_2) is ciphertext corresponding to message *nm*



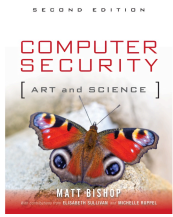
RSA

- First described publicly in 1978
  - Unknown at the time: Clifford Cocks developed a similar cryptosystem in 1973, but it was classified until recently
- Exponentiation cipher
- Relies on the difficulty of determining the number of numbers relatively prime to a large integer *n*



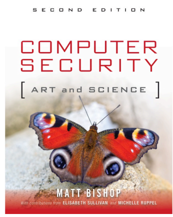
Background

- Totient function ϕ(n)
  - Number of positive integers less than *n* and relatively prime to *n*
    - *Relatively prime* means with no factors in common with *n*
- Example: ϕ(10) = 4
  - 1, 3, 7, 9 are relatively prime to 10
- Example: ϕ(21) = 12
  - 1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20 are relatively prime to 21



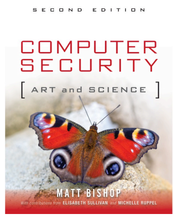
Algorithm

- Choose two large prime numbers *p, q*
  - Let $n = pq$; then $\phi(n) = (p-1)(q-1)$
  - Choose $e < n$ such that e is relatively prime to $\phi(n)$.
  - Compute d such that $ed \bmod \phi(n) = 1$
- Public key: (e, n); private key: d
- Encipher: $c = m^e \bmod n$
- Decipher: $m = c^d \bmod n$



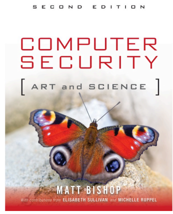
Example: Confidentiality

- Take p = 181, q = 1451, so n = 262631 and $\phi(n)$ = 261000
- Alice chooses e = 154993, making d = 95857
- Bob wants to send Alice secret message PUPPIESARESMALL (152015 150804 180017 041812 001111); encipher using public key
  - 152015^{154993} mod 262631 = 220160
  - 150804^{154993} mod 262631 = 135824
  - 180017^{154993} mod 262631 = 252355
  - 041812^{154993} mod 262631 = 245799
  - 001111_{154993} mod 262631 = 070707
- Bob sends 220160 135824 252355 245799 070707
- Alice uses her private key to decipher it



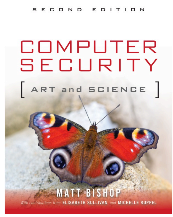
Example: Authentication/Integrity

- Alice wants to send Bob the message PUPPIESARESMALL in such a way that Bob knows it comes from her and nothing was changed during the transmission
  - Same public, private keys as before
- Encipher using private key:
  - $152015^{95857} \bmod 262631 = 072798$
  - $150804^{95857} \bmod 262631 = 259757$
  - $180017^{95857} \bmod 262631 = 256449$
  - $041812^{95857} \bmod 262631 = 089234$
  - $001111^{95857} \bmod 262631 = 037974$
- Alice sends 072798 259757 256449 089234 037974
- Bob receives, uses Alice’s public key to decipher it



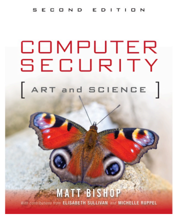
Example: Both (Sending)

- Same n as for Alice; Bob chooses e = 45593, making d = 235457
- Alice wants to send PUPPIESARESMALL (152015 150804 180017 041812 001111) confidentially and authenticated
- Encipher:
  - $(152015^{95857} \bmod 262631)^{45593} \bmod 262631 = 249123$
  - $(150804^{95857} \bmod 262631)^{45593} \bmod 262631 = 166008$
  - $(180017^{95857} \bmod 262631)^{45593} \bmod 262631 = 146608$
  - $(041812^{95857} \bmod 262631)^{45593} \bmod 262631 = 092311$
  - $(001111^{95857} \bmod 262631)^{45593} \bmod 262631 = 096768$
- So Alice sends 249123 166008 146608 092311 096768



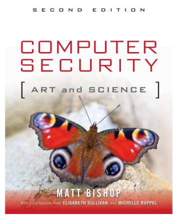
Example: Both (Receiving)

- Bob receives 249123 166008 146608 092311 096768
- Decipher:
  - $(249123^{235457} \bmod 262631)^{154993} \bmod 262631 = 152012$
  - $(166008^{235457} \bmod 262631)^{154993} \bmod 262631 = 150804$
  - $(146608^{235457} \bmod 262631)^{154993} \bmod 262631 = 180017$
  - $(092311^{235457} \bmod 262631)^{154993} \bmod 262631 = 041812$
  - $(096768^{235457} \bmod 262631)^{154993} \bmod 262631 = 001111$
- So Alice sent him 152015 150804 180017 041812 001111
  - Which translates to PUP PIE SAR ESM ALL or PUPPIESARESMALL



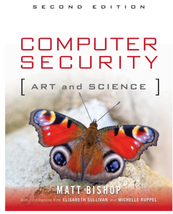
Security Services

- Confidentiality
  - Only the owner of the private key knows it, so text enciphered with public key cannot be read by anyone except the owner of the private key
- Authentication
  - Only the owner of the private key knows it, so text enciphered with private key must have been generated by the owner



More Security Services

- Integrity
  - Enciphered letters cannot be changed undetectably without knowing private key
- Non-Repudiation
  - Message enciphered with private key came from someone who knew it

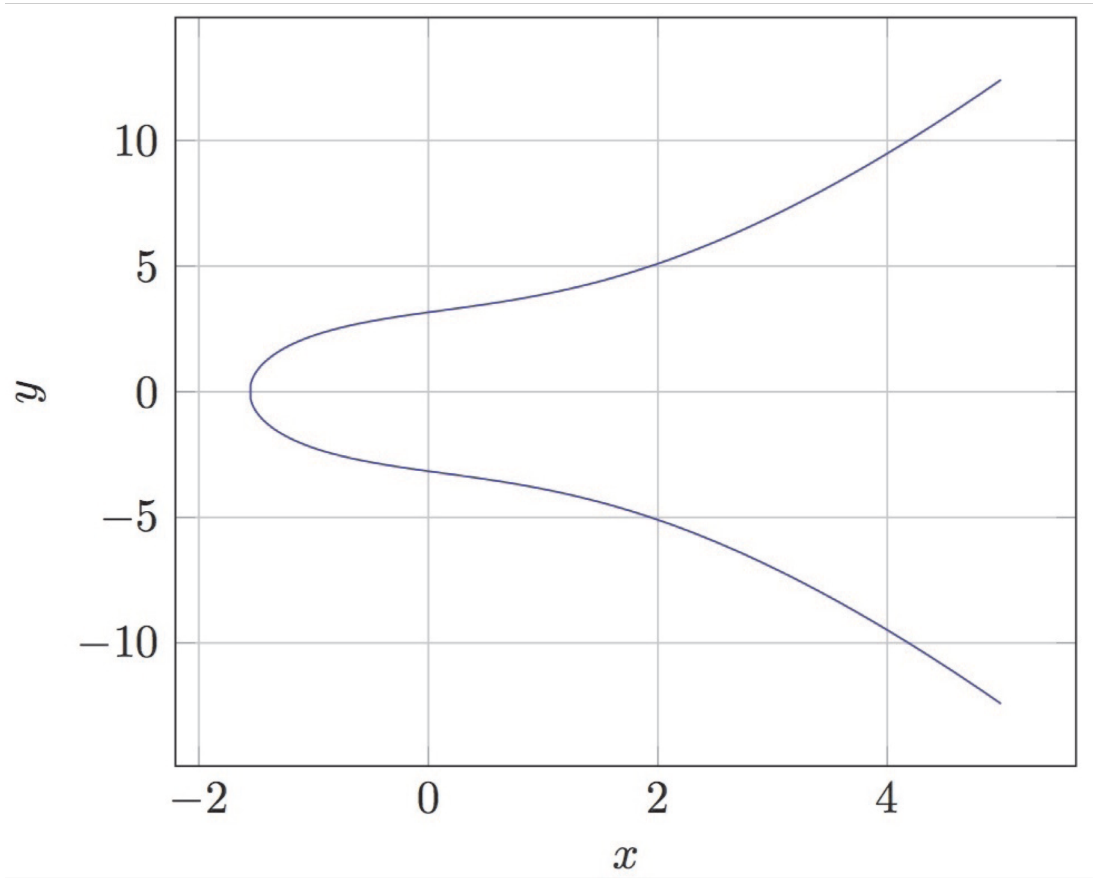


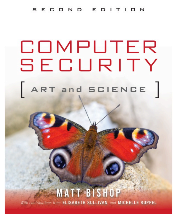
Warnings

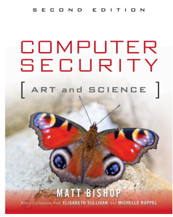
- Encipher message in blocks considerably larger than the examples here
  - If only characters per block, RSA can be broken using statistical attacks (just like symmetric cryptosystems)
- Attacker cannot alter letters, but can rearrange them and alter message meaning
  - Example: reverse enciphered message of text ON to get NO

Elliptic Curve Ciphers

- Miller and Koblitz proposed this
- *Elliptic curve* is a curve of the form $y^2 = x^3 + ax + b$
  - Curve $y^2 = x^3 + 4x + 10$ plotted at right
- Can be applied to any cryptosystem depending on discrete log problem
- Advantage: keys shorter than other forms of public key cryptosystems, so computation time shorter

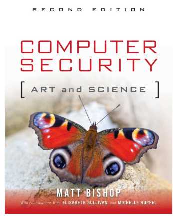






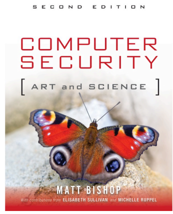
Basics

- Take 2 points on the elliptic curve P_1, P_2
  - If $P_1 \neq P_2$, draw line through them
  - If $P_1 = P_2$, draw a tangent to curve there
- If line intersects curve at $P_3 = (x_3, y_3)$
  - Take the sum of P_1, P_2 to be P4 = $(x_3, -y_3)$
- Otherwise, line is vertical, so take $P_1 = (x, y)$; treat ∞ as another point of intersection; third point of intersection is $P_2 = (x, -y)$
  - Given above definition of addition, $P_1 + \infty = (x, y) = P_1$
  - So ∞ is additive identity



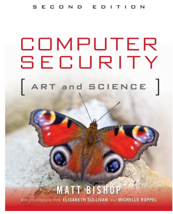
The Math

- $P_1 = (x_1, y_1)$; $P_2 = (x_2, y_2)$
- Then if $P_1 \neq P_2$, $m = (y_2 - y_1) / (x_2 - x_1)$
- Otherwise, $m = (3x_1^2 + a) / y_1$
- Next, $P_3 = P_1 + P_2 = (m^2 - x_1 - x_2, m(x_1 - x_3) - y_1) = (x_3, y_3)$
- And $P_4 = (x_{4,}, y_4)$, where $x_4 = x_3$, $y_4 = -y_3$
    - P_4 defined to be sum of P_1, P_2



Basis for the Cryptosystem

- Curve: $y^2 = x^3 + ax + b \bmod p$, where $4a^3 + 27b^2 \neq 0$ and p prime
- Pick a point P and add it to itself n times; call this Q, so $Q = nP$
  - If n is large, generally very hard to compute n from P and Q
- So, elliptic curve cryptosystem has 4 parameters (a, b, p, P)
- Private key k_{priv} chosen randomly such that $k_{priv} < p$
  - In practice, choose k_{priv} to be less than number of integer points on curve
- Public key $k_{pub} = k_{priv} P$
- In what follows, $(x, y) \bmod p = (x \bmod \mathrm{p}, y \bmod p)$



Elliptic Curve El Gamal Cryptosystem

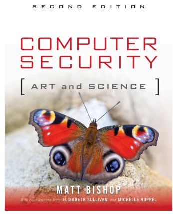
- Choose a point P on the curve, and a private key *kpriv*
- Compute $Q = k_{priv}P$
- Public key is (P, Q, a, p)

Encipher: express message as point m on curve; choose random number k

- $c_1 = kP$; $c_2 = m + kQ$
- Ciphertext is (c_1, c_2)

Decipher:

- $m = c_2 - k_{priv}c_1$
- Message is m



Example: Encryption

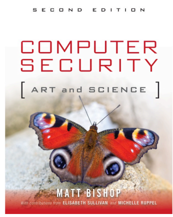
- Alice, Bob agree to use the curve $y^2 = x^3 + 4x + 14 \bmod 2503$ and the point $P = (1002, 493)$
- Bob chooses private key $k_{priv,Bob} = 1847$
  - Public key $k_{pub\ ,Bob} = k_{priv,Bob}P = 1847(1002, 493) \bmod 2503 = (460, 2083)$
- Alice wants to send Bob message $m = (18, 1394)$
  - She chooses random $k = 717$
  - $c_1 = kP = 717(1002, 493) \bmod 2503 = (2134, 419)$
  - $c_2 = m + k\,k_{pub\ ,Bob} = (18, 1394) + 717(460, 2083) \bmod 2503 = (221, 1253)$

so she sends Bob c_1 and c_2

Example: Decryption

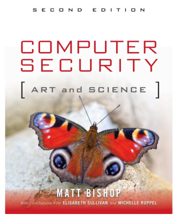
- From last slide, Alice, Bob agree to use the curve $y^2 = x^3 + 4x + 14 \bmod 2503$ and the point $P = (1002, 493)$
  - Bob's private key $k_{priv,Bob} = 1847$
  - Bob's public key $k_{pub,Bob}$ (460, 2083)
- To decrypt $c_1 = (2134, 419)$, $c_2 = (221, 1253)$, Bob computes:
  - $k_{priv,Bob}c_1 = 1847(2134, 419) \bmod 2503 = (652, 1943)$
  - $m = c_2 - c_1 = (221, 1253) - (652, 1943) \bmod 2503 = (18, 1394)$

obtaining the message Alice sent



Selection of Elliptic Curves

- For elliptic curves for cryptography, selection of parameters critical
  - Example: b = 0, p mod 4 = 3 makes the underlying discrete log problem significantly easier to solve
  - Example: so does a = 0, p mod 3 = 2
- Several such curves are recommended:
  - U.S. NIST: P-192, P-224, P-256, P-384, P-521 using a prime modulus and a binary field of degree 163, 233, 283 409, 571
  - Certicom: same, but degree 239 binary field instead of degree 233 binary field
  - Others: Curve1174, Curve25519

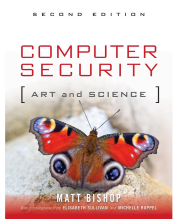


Cryptographic Checksums

- Mathematical function to generate a set of k bits from a set of n bits (where $k \le n$).
  - k is smaller then n except in unusual circumstances
- Example: ASCII parity bit
  - ASCII has 7 bits; 8th bit is "parity"
  - Even parity: even number of 1 bits
  - Odd parity: odd number of 1 bits

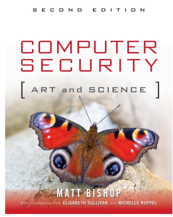
Example Use

- Bob receives “10111101” as bits.
  - Sender is using even parity; 6 1 bits, so character was received correctly
    - Note: could be garbled, but 2 bits would need to have been changed to preserve parity
  - Sender is using odd parity; even number of 1 bits, so character was not received correctly



Definition

- Cryptographic checksum h: $A \rightarrow B$:
  1. For any $x \in A$, $h(x)$ is easy to compute
  2. For any $y \in B$, it is computationally infeasible to find $x \in A$ such that $h(x) = y$
  3. It is computationally infeasible to find two inputs $x, x' \in A$ such that $x \neq x'$ and $h(x) = h(x')$
     – Alternate form (stronger): Given any $x \in A$, it is computationally infeasible to find a different $x' \in A$ such that $h(x) = h(x')$.

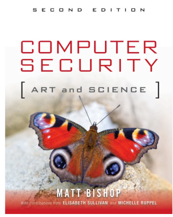


Collisions

- If $x \neq x'$ and $h(x) = h(x')$, x and x' are a *collision*
  - Pigeonhole principle: if there are n containers for n+1 objects, then at least one container will have at least 2 objects in it.
  - Application: if there are 32 files and 8 possible cryptographic checksum values, at least one value corresponds to at least 4 files

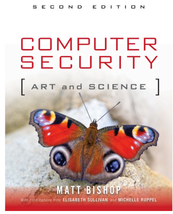
Keys

- Keyed cryptographic checksum: requires cryptographic key
  - AES in chaining mode: encipher message, use last *n* bits. Requires a key to encipher, so it is a keyed cryptographic checksum.
- Keyless cryptographic checksum: requires no cryptographic key
  - SHA-512, SHA-3 are examples; older ones include MD4, MD5, RIPEM, SHA-0, and SHA-1 (methods for constructing collisions are known for these)



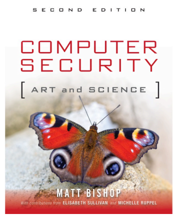
HMAC

- Make keyed cryptographic checksums from keyless cryptographic checksums
- *h* keyless cryptographic checksum function that takes data in blocks of *b* bytes and outputs blocks of *l* bytes. k' is cryptographic key of length *b* bytes
    - If short, pad with 0 bytes; if long, hash to length *b*
- *ipad* is 00110110 repeated *b* times
- *opad* is 01011100 repeated *b* times
- HMAC-$h(k, m) = h(k' \oplus opad \,||\, h(k' \oplus ipad \,||\, m))$
    - $\oplus$ exclusive or, $||$ concatenation



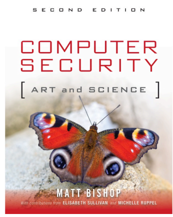
Strength of HMAC-*h*

- Depends on the strength of the hash function *h*
- Attacks on HMAC-MD4, HMAC-MD5, HMAC-SHA-0, and HMAC-SHA-1 recover partial or full keys
  - Note all of MD4, MD5, SHA-0, and SHA-1 have been broken



Digital Signature

- Construct that authenticates origin, contents of message in a manner provable to a disinterested third party (a “judge”)
- Sender cannot deny having sent message (service is “nonrepudiation”)
  - Limited to *technical* proofs
    - Inability to deny one’s cryptographic key was used to sign
  - One could claim the cryptographic key was stolen or compromised
    - Legal proofs, *etc.*, probably required; not dealt with here



Common Error

- Symmetric: Alice, Bob share key *k*
  - Alice sends *m* || { *m* } *k* to Bob
  - { *m* } *k* means *m* enciphered with key *k*, || means concatenation

  Claim: This is a digital signature

<u>WRONG</u>

This is not a digital signature

- Why? Third party cannot determine whether Alice or Bob generated message

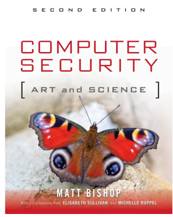
Classical Digital Signatures

- Require trusted third party
  - Alice, Bob each share keys with trusted party Cathy
- To resolve dispute, judge gets $\{ m \}\, k_{Alice}$, $\{ m \}\, k_{Bob}$, and has Cathy decipher them; if messages matched, contract was signed

$$\text{Alice} \xrightarrow{\{ m \} k_{Alice}} \text{Bob}$$

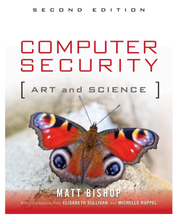
$$\text{Cathy} \xleftarrow{\{ m \} k_{Alice}} \text{Bob}$$

$$\text{Cathy} \xrightarrow{\{ m \} k_{Bob}} \text{Bob}$$



Public Key Digital Signatures

- Basically, Alice enciphers the message, or its cryptographic hash, with her private key
- In case of dispute or question of origin or whether changes have been made, a judge can use Alice's public key to verify the message came from Alice and has not been changed since being signed



RSA Digital Signatures

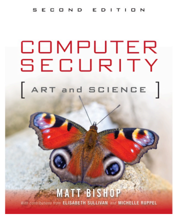
- Alice's keys are (e_{Alice},n_{Alice}) (public key), d_{Alice} (private key)
  - In what follows, we use e_{Alice} to represent the public key
- Alice sends Bob

$$m \,||\, \{ m \} e_{Alice}$$

- In case of dispute, judge computes

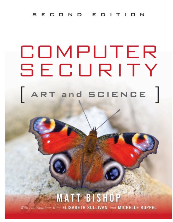
$$\{ \{ m \} e_{Alice} \} d_{Alice}$$

- and if it is *m*, Alice signed message
  - She's the only one who knows d_{Alice}!



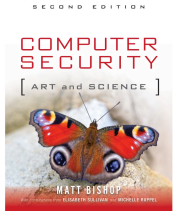
RSA Digital Signatures

- Use private key to encipher message
  - Protocol for use is *critical*
- Key points:
  - Never sign random documents, and when signing, always sign hash and never document
  - Don’t just encipher message and then sign, or vice versa
    - Changing public key and private key can cause problems
    - Messages can be forwarded, so third party cannot tell if original sender sent it to her



Attack #1

- Example: Alice, Bob communicating
  - $n_A = 262631$, $e_A = 154993$, $d_A = 95857$
  - $n_B = 288329$, $e_B = 22579$, $d_B = 138091$
- Alice asks Bob to sign 225536 so she can verify she has the right public key:
  - $c = m^{d_B} \bmod n_B = 225536^{138091} \bmod 288329 = 271316$
- Now she asks Bob to sign the statement AYE (002404):
  - $c = m^{d_B} \bmod n_B = 002404^{138091} \bmod 288329 = 182665$

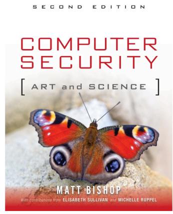


Attack #1

- Alice computes:
  - new message NAY (130024) by (002404)(225536) mod 288329 = 130024
  - corresponding signature (271316)(182665) mod 288329 = 218646
- Alice now claims Bob signed NAY (130024), and as proof supplies signature 218646
- Judge computes c^{e_B} mod n_B = 218646^{22579} mod 288329 = 130024
  - Signature validated; Bob is toast

Preventing Attack #1

- Do not sign random messages
  - This would prevent Alice from getting the first message
- When signing, always sign the cryptographic hash of a message, not the message itself

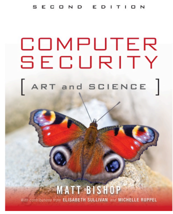


Attack #2: Bob's Revenge

- Bob, Alice agree to sign contract LUR (112017)
  - But Bob really wants her to sign contract EWM (042212), but knows she won't
- Alice enciphers, then signs:
  - $(m^{e_B} \bmod n_A)^{d_A} \bmod n_A = (112017^{22579} \bmod 288329)^{95857} \bmod 262631 = 42390$
- Bob now changes his public key
  - Computes r such that $042212^r \bmod 288329 = 112017$; one such $r = 9175$
  - Computes $re_B \bmod \phi(n_B) = (9175)(22579) \bmod 287184 = 102661$
  - Replace public key with (102661,288329), private key with 161245
- Bob claims contract was EWM
- Judge computes:
  - $(42390^{154993} \bmod 262631)^{161245} \bmod 288329 = 042212$, which is EWM
  - Verified; now Alice is toast

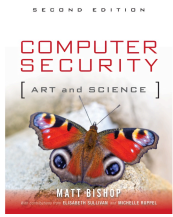
Preventing Attack #2

- Obvious thought: instead of encrypting message and then signing it, sign the message and then encrypt it
  - May not work due to surreptitious forwarding attack
  - Idea: Alice sends Cathy an encrypted signed message; Cathy deciphers it, re-enciphers it with Bob's public key, and then sends message and signature to Bob – now Bob thinks the message came from Alice (right) and was intended for him (wrong)
- Several ways to solve this:
  - Put sender and recipient in the message; changing recipient invalidates signature
  - Sign message, encrypt it, then sign the result



El Gamal Digital Signature

- Relies on discrete log problem
  - Choose p prime, g, $d < p$; compute $y = g^d \bmod p$
- Public key: (y, g, p); private key: d
- To sign contract m:
  - Choose k relatively prime to p–1, and not yet used
  - Compute $a = g^k \bmod p$
  - Find b such that $m = (da + kb) \bmod p–1$
  - Signature is (a, b)
- To validate, check that
  - $y^a a^b \bmod p = g^m \bmod p$



Example

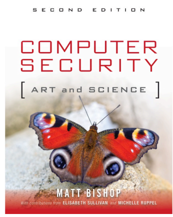
- Alice chooses p = 262643, g = 9563, d = 3632, giving y = 274598
- Alice wants to send Bob signed contract PUP (152015)
  - Chooses k = 601 (relatively prime to 262642)
  - This gives $a = g^k \bmod p = 9563^{601} \bmod 29 = 202897$
  - Then solving $152015 = (3632 \times 202897 + 601b) \bmod 262642$ gives b = 225835
  - Alice sends Bob message m = 152015 and signature (a,b) = (202897, 225835)
- Bob verifies signature: $g^m \bmod p = 9563^{152015} \bmod 262643 = 157499$ and $y^a a^b \bmod p = 27459^{202897} 202897^{225835} \bmod 262643 = 157499$
  - They match, so Alice signed

Attack

- Eve learns k, corresponding message m, and signature (a, b)
  - Extended Euclidean Algorithm gives d, the private key
- Example from above: Eve learned Alice signed last message with $k = 5$

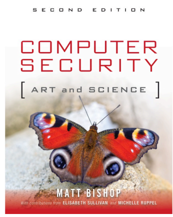
$$m = (da + kb) \bmod p{-}1 \Rightarrow 152015 = (202897d + 601 \times 225835) \bmod 262642$$

giving Alice's private key $d = 3632$



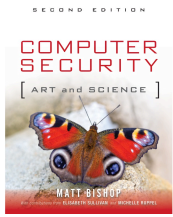
El Gamal Digital Signature Using Elliptic Curve Cryptography

- As before, curve is $y^2 = x^3 + ax + b \bmod p$ with n integer points on it
  - Choose a point P on the curve
  - Choose private key *kpriv;* compute $Q = k_{priv}P$, and the corresponding public key is (P, Q, a, b)
- To digitally sign, choose random integer k with $1 \le k < n$
  - Compute $R = kP$ and $s = k^{-1}(m - k_{priv}x) \bmod n$, where x is first component of R
  - Digital signature is (m, R, s)
- To validate, recipient computes:
  - $V_1 = xQ + sR$
  - $V_2 = mP$
  - If $V_1 = V_2$, signature valid



Example

- Alice, Bob use elliptic curve $y^2 = x^3 + 4x + 14 \bmod 2503$, point $P = (1002, 493)$
  - Curve has $n = 2477$ integer points on it
  - Bob chooses $k_{priv,Bob} = 1874$, so $Q = 1847(1002, 493) \bmod 2503 = (460, 2083)$
- Bob digitally signs message $m = 379$
  - Chooses $k = 877$
  - Computes $R = kP = 877(1002,493) = (1014, 788)$
  - Computes $s = k^{-1}(m - k_{priv,Bob}x) \bmod n = 877^{-1}(379 - 1847 \times 1014) \bmod 2477 = 2367$
  - Sends Alice (379, (1014, 788), 2367)



Example

- To validate signature, Alice computes:
  - $V_1 = xQ + sR = 1014(460,2083) + 2367(1014, 788) = (535, 1015)$
  - $V_2 = mP = 379(1002,493) = (535, 1015)$
- As $V_1 = V_2$, the signature is validated

Key Points

- Two main types of cryptosystems: classical and public key
- Classical cryptosystems encipher and decipher using the same key
  - Or one key is easily derived from the other
- Public key cryptosystems encipher and decipher using different keys
  - Computationally infeasible to derive one from the other
- Cryptographic checksums provide a check on integrity
- Digital signatures provide integrity of origin and content

  Much easier with public key cryptosystems than with classical cryptosystems