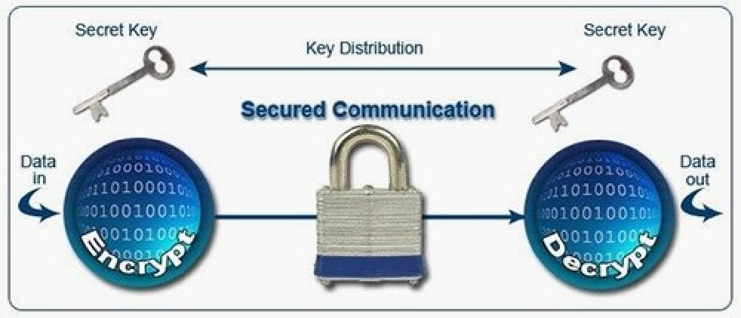
DOTA2024:4 Defense of the Ancients Fourth topic - Cryptography

Hugh Anderson

National University of Singapore
School of Computing

July, 2024

Keys...


Secret Key
Key Distribution
Secret Key
Secured Communication
Data in
Encrypt
Decrypt
Data out

Outline

Outline

Outline

1. **Encryption...**
   - Prehistory of crypto
   - Terms, definitions, goals
2. **Modern symmetric ciphers**
   - Symmetric cipher building blocks
   - DES
   - AES
3. **From symmetric to asymmetric**
   - Building blocks
   - Asymmetric systems: Diffie-Hellman, RSA and ECC
4. **The future...**
   - Not just confidentiality: SS, MPC, IBE...
   - The future of crypto

Outline

1. **Encryption...**
   - Prehistory of crypto
   - Terms, definitions, goals
2. **Modern symmetric ciphers**
   - Symmetric cipher building blocks
   - DES
   - AES
3. **From symmetric to asymmetric**
   - Building blocks
   - Asymmetric systems: Diffie-Hellman, RSA and ECC
4. **The future...**
   - Not just confidentiality: SS, MPC, IBE...
   - The future of crypto

Outline

1. **Encryption...**
   - Prehistory of crypto
   - Terms, definitions, goals
2. **Modern symmetric ciphers**
   - Symmetric cipher building blocks
   - DES
   - AES
3. **From symmetric to asymmetric**
   - Building blocks
   - Asymmetric systems: Diffie-Hellman, RSA and ECC
4. **The future...**
   - Not just confidentiality: SS, MPC, IBE...
   - The future of crypto

Julius Cæsar cipher...

The Cæsar cipher

Cæsar (rotation) cipher over Roman letters: Key is "+3".

I C L A V D I V S

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

M F O D Z G M Z X

Can define the transformation mathematically:

$$
\begin{aligned}
c &= \mathrm{Enc}_k(p) &= (p + k) \mod 23 \\
p &= \mathrm{Dec}_k(c) &= (c - k) \mod 23
\end{aligned}
$$

Cryptanalysis of rotation ciphers:

In the above example - we only have 22 possible useful ciphers! So an attacker can try each in turn: a brute force search

Union (North) and Confederate (South) ciphers

Used in the American Civil war, they can be used as simple rotation cipher.

The Confederate one has a keyspace of 26, and a useful keyspace of only 25.

Substitution

Substitution cipher systems encode the input stream using a substitution rule. The Cæsar cipher is an example of a simple substitution cipher system.

A (random) monoalphabetic substitution cipher

Code	Encoding
A	Q
B	V
C	X
D	W
...	...

If the mapping was more randomly chosen it is called a monoalphabetic substitution cipher, and the keyspace for encoding 26 letters would be $26! - 1 = 403,291,461,126,605,635,583,999,999$.

Substitution cipher...

Cryptanalysis of substitution cipher

How safe is this cipher? (Not at all!)

If we could decrypt $1,000,000$ messages in a second, then the average time to find a solution by trying decryptions would be about $6,394,144,170,576$ years!

We might be lulled into a sense of security by these big numbers, but of course this sort of cipher can be subject to frequency analysis...

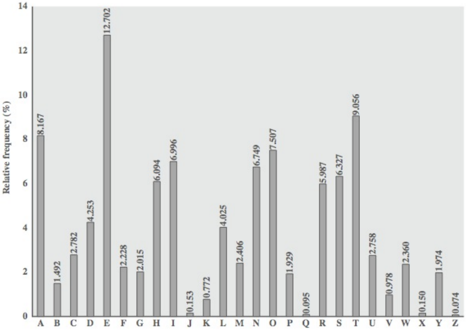
The problems are that:

1. letters are not equally common: ETAOINSHRDLU!
2. human languages have high levels of redundancy: (fr xmpl, hgh ndrsn s tchng DT ths smstr).

We have tables of single, double & triple letter frequencies for languages.

Cryptanalysis of mono-alphabetic ciphers

Frequencies of english text:



These ciphers do not change relative letter frequencies

The central concept of this was discovered and described by Arabian scientists in the 9th century. An attacker can calculate the frequencies for ciphertext. The most common ciphertext letter might translate to an E.

Cryptanalysis of mono-alphabetic ciphers

Example - first step of decoding:

```
EV YQS CVV MIWK FRPC FRQF FRV IQFV WM
FIQSCKPCCPWS ACPSN IVJVFPFPWS RQC FW
QJJIWQYR ZVIW FW QYRPVDV KWIV QSB KWIV
IVTPQXTV FIQSCKPCCPWS. RWEVDVI EV LSWE
FRQF FRV FRVWIVFPYQT IQFV CRWATB ...
```

V occurs most often, F next and so on, so replace V with E...

Example - first step of decoding:

```
EV YQS CVV MIWK FRPC FRQF FRV IQFV WM
-E -A- -EE F-O- THI- THAT THE -ATE OF

FIQSCKPCCPWS ACPSN IVJVFPFPWS RQC FW
T-A---I--IO- --I-- -E-ETITIO- HA- TO

QJJIWQYR ZVIW FW QYRPVDV KWIV QSB KWIV
A---OA-H -E-O TO A-HIE-E -O-E A-- -O-E
```

Polyalphabetic ciphers

Polyalphabetic substitution ciphers improve security:

There are more alphabets to guess and hence a flatter frequency distribution.
We use a key to select which cipher is used for each letter of message.

Vigenère (1520) uses a tableau, and a key:

	A	B	C	D	E	F	G	H	...
A	A	B	C	D	E	F	G	H	...
B	B	C	D	E	F	G	H	I	...
C	C	D	E	F	G	H	I	J	...
D	D	E	F	G	H	I	J	K	...
E	E	F	G	H	I	J	K	L	...
F	F	G	H	I	J	K	L	M	...
G	G	H	I	J	K	L	M	N	...
H	H	I	J	K	L	M	N	O	...
...	...	...	...	...	...	...	...	...	...

Vigenère

Keyword is BAD, so encoding HAD A FEED results in:

Key	B	A	D	B	A	D	B	A
Text	H	A	D	A	F	E	E	D
Cipher	I	A	G	B	F	H	F	D

If we can discover the length of the repeated key (in this case 3), and the text is long enough, we can just consider the cipher to be a group of interleaved substitution ciphers and solve accordingly.

Cryptanalysis of Vigenère cipher:

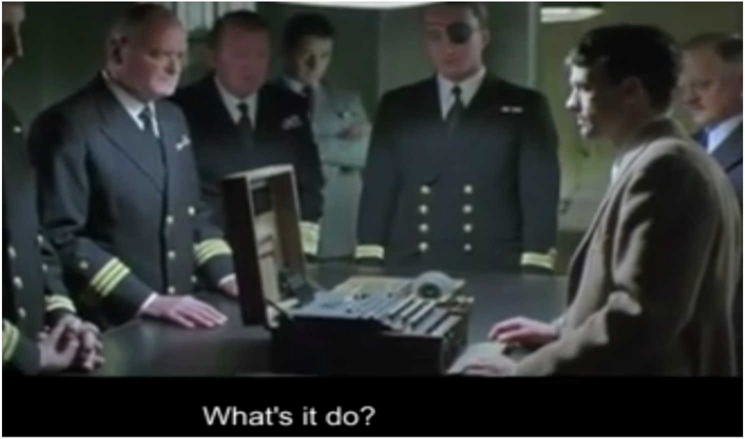
Multiple ciphertext letters for each plaintext letter, and so letter frequencies are obscured (but not totally lost)
Start with letter frequencies, see if monoalphabetic or not. If not, then need to determine number of alphabets.

Example of a polyalphabetic substitution cipher

The M-94 cipher

Used by the US army from 1922 to 1942. It had 25 disks, each containing a random sequence of the letters A-Z around the outside.

Polyalphabetic substitution cipher machine...

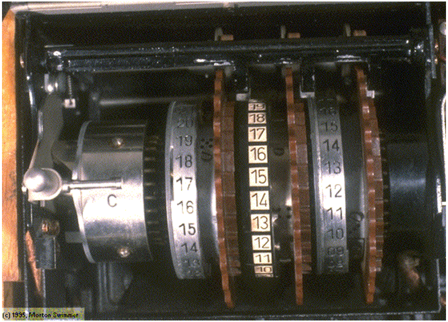


Polyalphabetic substitution cipher machine...

Dr James Grime on the Enigma...

Polyalphabetic machines: 70 years ago...

The Enigma machine, and closeup of its rotors...



US M209 Rotor machine

WWII Mechanical encryption machine

What is inside? A Beaufort cipher

Reversed tableau...

	A	B	C	D	E	F	G	H	...
0	Z	Y	X	W	V	U	T	S	...
1	A	Z	Y	X	W	V	U	T	...
2	B	A	Z	Y	X	W	V	U	...
3	C	B	A	Z	Y	X	W	V	...
4	D	C	B	A	Z	Y	X	W	...
5	E	D	C	B	A	Z	Y	X	...
6	F	E	D	C	B	A	Z	Y	...
7	G	F	E	D	C	B	A	X	...
...	...	...	...	...	...	...	...	...	...

H encoded with key 5 is X.
More material at `https://www.ciphermachinesandcryptology.com/en/m209sim.htm`

Cryptanalysis: Kasiski method

Method developed by Babbage (1854) and Kasiski (1863):

Repetitions in ciphertext give clues to the period; so find some plaintexts an exact period apart (of course, could also be a random fluke).
Then attack each monoalphabetic cipher individually using the same techniques as before.

Despite this - systems used into 20th century:

The Zimmermann Telegram (or Zimmermann Note; German: Zimmermann-Depesche; Spanish: Telegrama Zimmermann) was a 1917 diplomatic proposal from the German Empire to Mexico to make war against the United States. The proposal was declined by Mexico, but angered Americans and led in part to the declaration of war in April [Wikipedia].

Transposition ciphers

Transposition/permutation ciphers:

- Hide the message by rearranging letter order.
- Have the same frequency distribution as the original text

Rail-fence cipher:

Write message letters out diagonally over a number of rows then read off cipher row by row eg. write message out as:

```
m   e   m   a   t   r   h   t   g   p   r   y
  e   t   e   f   e   t   e   o   a   a   t
```

giving ciphertext

MEMATRHTGPRYETEFETEOAAT

Row transposition cipher

Row transposition - more complex:

Write letters of message out in rows over a specified number of columns then reorder the columns according to some key before reading off the rows:

```
Key:         4 3 1 2 5 6 7
Plaintext:   a t t a c k p
             o s t p o n e
             d u n t i l t
             w o a m x y z
Ciphertext:  TTNAAPTMTSUOAODWCOIXKNLYPETZ
```

Cryptanalysis of transposition cipher:

- Easily recognized because it has the same letter frequencies as the original plaintext.
- For the transposition just shown, cryptanalysis is easy and involves laying out the ciphertext in a matrix and playing around with column positions.
- Digram and trigram frequency tables can be useful.

Product ciphers

Substitution plus transposition:

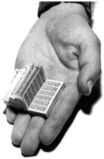
Is double encryption a thing? i.e. $c = \mathrm{Enc}_{k_1}(\mathrm{Enc}'_{k_2}(x))$ where k_1 is the key for the first encryption Enc and k_2 is the key for the second encryption Enc'.

- If both Enc and Enc' are transposition ciphers, then there is no advantage of doing so. The combined cipher is still a transposition.
- If Enc and Enc' are both substitution ciphers, then the final encryption is simply another substitution cipher.

But doing transposition and sustitution form an important class of ciphers: the Substitution-Permutation Networks (SPN). A bridge from classical to modern ciphers.

One time pad/Vernam's cipher

An "unconditionally secure" scheme:



- One time pad provides perfect secrecy.
- The key is a sequence of random key letters, each letter used once only, and available at only the sender and receiver.

Key points/jargon

Summary:

- Substitution ciphers
  - Cæsar/rotation,
  - Random (mono-alphabetic substitution)
  - Vigenère, Beaufort (poly-alphabetic substitution)
  - One time pad (Vernam's cipher)
  - Playfair
- Transposition/permutation ciphers
  - Rail fence cipher
- Frequency analysis for cryptanalysis

Outline

1. **Encryption...**
   - Prehistory of crypto
   - Terms, definitions, goals
2. **Modern symmetric ciphers**
   - Symmetric cipher building blocks
   - DES
   - AES
3. **From symmetric to asymmetric**
   - Building blocks
   - Asymmetric systems: Diffie-Hellman, RSA and ECC
4. **The future...**
   - Not just confidentiality: SS, MPC, IBE...
   - The future of crypto

Encryption goals: confidentiality and Integrity

E.g. Electronic commerce

Commerce relies on secure transfer of information, and the distance between you and an attacker is shrinking. Criminals have an access point into your living room
Often want things to be secret (C), and also want to be sure it is authentic, i.e. it has integrity (I).

The goals of cryptography are to ensure both C and I.

Encryption terminology

Terms commonly used:

- plaintext: original message
- ciphertext: coded message
- cipher: algorithm for transforming plaintext to ciphertext
- key: info used in cipher known only to sender/receiver
- encipher/encrypt: converting plaintext into ciphertext
- decipher/decrypt: recovering plaintext from ciphertext
- cryptography: study of encryption principles/methods
- cryptanalysis/codebreaking: study of principles/ methods of deciphering ciphertext without knowing key
- cryptology: field of both cryptography and cryptanalysis

Cryptographics and adversary's goals

Cryptographic systems are characterized by:

... the type of encryption operations used, the number of keys used, and the way in which plaintext is processed: block or stream.

Attacker's goals:

- If an attacker is able to find the key, we call this a total break.
- The attacker may be satisfied with a partial break. For instance, the adversary can determine some specific information about the plaintext (for example, the first bit).
- The attacker may be satisfied with distinguishability of ciphertext: the attacker is able to distinguish between encryption of two given plaintext, or between an encryption of a given plaintext and a random string.
- Total break is the "strongest" goal in the sense that, if an adversary is able to achieve total break, he is also able to achieve partial break and distinguishablility of ciphertext.

Cryptanalysis

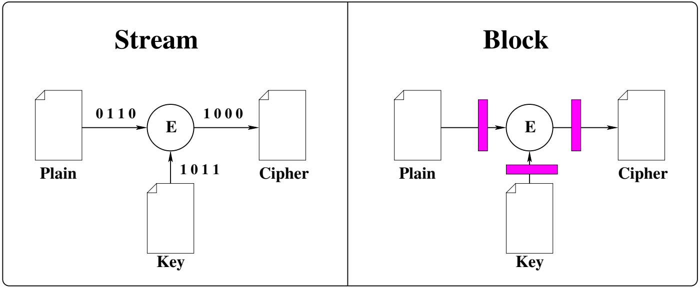
Attack models, based on information known to attacker:

- Ciphertext only: The adversary has a collection of ciphertext c.
- Known plaintext: The adversary has a collection of plaintext m and their corresponding ciphertext c.
- Chosen plaintext: The adversary has temporary access to a black box. He can choose a plaintext m and obtain the corresponding ciphertext c from the black box. He can access the black box for a reasonable large amount of time.
- Chosen ciphertext: same as chosen plaintext attack, but here, the adversary chooses the ciphertext and the blackbox gives the plaintext.
- Chosen text: select plaintext or ciphertext to en/decrypt

Outline

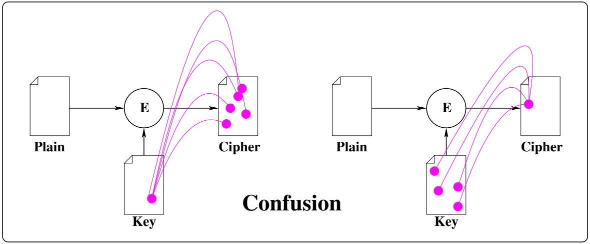
Block/stream cipher

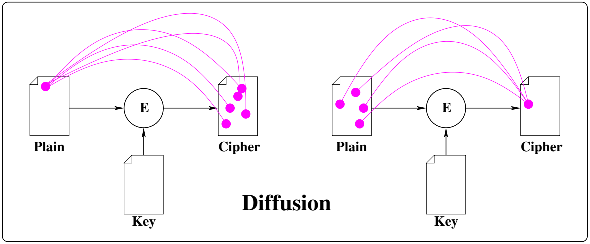
What are block and stream ciphers?



A stream cipher decrypts or encrypts the data stream one bit at a time.
A block cipher decrypts or encrypts the data stream a block (of bits) at a time.
In DES it is 64 bits. The keysize is also some fixed size.

Confusion and diffusion (Shannon)





Block cipher concepts

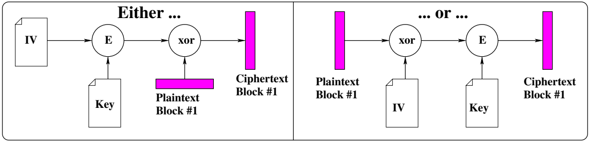
If data is not some multiple of the blocksize?

One technique is to add a single 1 bit, followed by enough 0 bits to fill out the block (if it ends on a block boundary, a whole padding block will be added).

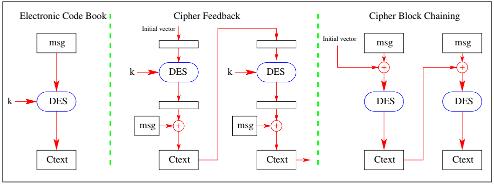
Data:	1110	100110101110101101011011	010111001000001101011011	011011101
Blocks:	1110	100110101110101101011011	010111001000001101011011	011011101100000000000000

Initalization vectors (IV) to randomize encryption...

The IV does not need to be secret, but should be unpredictable, and not reused with the same key. A (previously) common practice of re-using the last ciphertext block of a message as the IV for the next message is insecure.



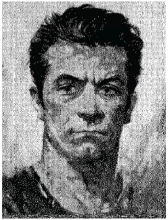
“Modes” of operation (for AES/DES/...)



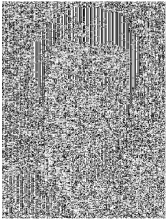
The US government recommends not using the Electronic Codebook mode.

From Bart Praneel

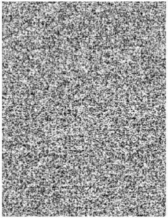
Original image



AES in ECB



AES in CBC



Middle image retains some easily retrievable information.

Avalanche - A "goodness" measure for crypto

Number of bits changed on each round

Assume we had a crypto system, and two plaintext messages p, and p', which had only one bit different.

If we were to encrypt the plaintext message p, ($c = E(k, p)$), and then encrypt the other plaintext message p' ($c' = E(k, p')$), a good cryptosystem should have two ciphertexts (c and c') with many bits different.

Why? How many bits (approximately) should be different?

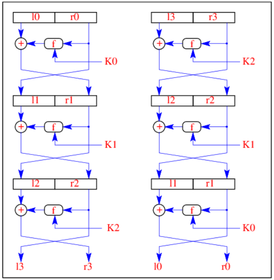
Outline

DES - Data Encryption Standard

DES - a (hardware oriented) BLOCK cipher

DES was first proposed by IBM (1974) using 128 bit keys. The Security was reduced by NSA (the National Security Agency) to a 56 bit key. The (shared) 56 bit key generates 16 48-bit subkeys, which each control a *round*.

At the core of DES is the Feistel network



Each of the 16 stages (rounds) of DES uses a Feistel structure which encrypts a 64 bit value into another 64 bit value using the 48 bit subkey. There is a substitution on the left data half, based on a round function. Then we have a permutation - swapping halves.

DES avalanche - 1 bit change in plaintext

Number of bits changed/64 on each round

Round		δ
	02468aceeca86420 12468aceeca86420	1
1	3cf03c0fbad22845 3cf03c0fbad32845	1
2	bad2284599e9b723 bad3284539a9b7a3	5
3	99e9b7230bae3b9e 39a9b7a3171cb8b3	18
4	0bae3b9e42415649 171cb8b3ccaca55e	34
5	4241564918b3fa41 ccaca55ed16c3653	37
6	18b3fa419616fe23 d16c3653cf402c68	33
7	9616fe2367117cf2 cf402c682b2cefbc	32
8	67117cf2c11bfc09 2b2cefbc99f91153	33
9	c11bfc09887fbc6c 99f911532eed7d94	32
10	887fbc6c600f7e8b 2eed7d94d0f23094	34
11	600f7e8bf596506e d0f23094455da9c4	37
12	f596506e738538b8 455da9c47f6e3cf3	31
13	738538b8c6a62c4e 7f6e3cf34bc1a8d9	29
14	c6a62c4e56b0bd75 4bc1a8d91e07d409	33
15	56b0bd7575e8fd8f 1e07d4091ce2e6dc	31
16	75e8fd8f25896490 1ce2e6dc365e5f59	32
IP^{-1}	da02ce3a89ecac3b 057cde97d7683f2a	32

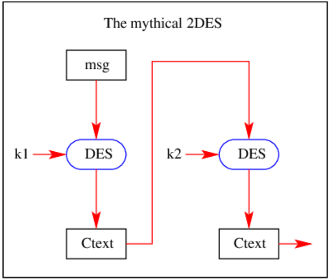
DES considered harmful

Or at least... weak ...

56-bit keys have $2^{56} = 7.2 \times 10^{16}$ values. A brute force search looks like it might be hard, but consider the speedup in computers from the 1970s onwards. In 1997 using a network of computers on the Internet, it took a few months. In 1998, on dedicated h/w (EFF), it took a few days

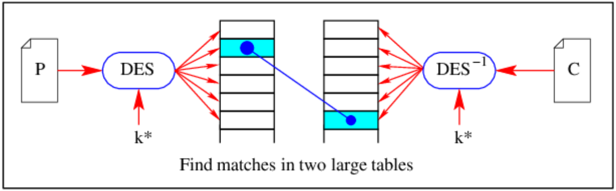
http://en.wikipedia.org/wiki/EFF_DES_cracker

In 1999, 22hrs (and so on). How long does it take now? We must consider alternatives to DES: 2DES?



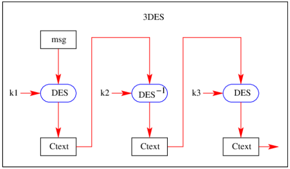
The problem with 2DES

2DES meet-in-the-middle attack



Starts with the attacker having a plaintext, ciphertext pair $\langle p, c\rangle$. The attacker computes two tables: $E(k,p)$ for each of the 2^{56} keys, and $D(k,c)$ for each of the 2^{56} keys. For each match in the two tables, you have found a possible key, with only 2^{57} DES operations (ie - not 2^{112}).

What we use: 3DES. Attack needs 2^{112} operations.



Outline

AES intro

Due to rising worries about DES

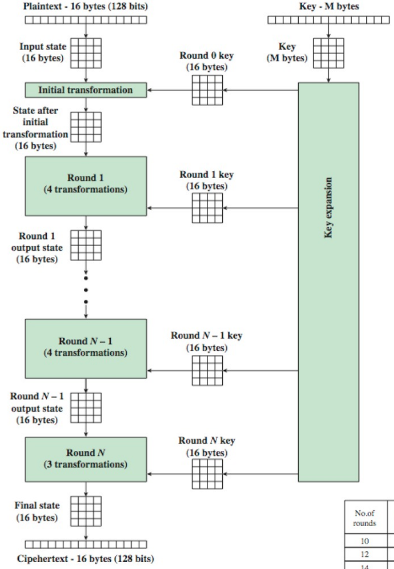
In 1997 NIST called for proposals for a royalty-free and public symmetric block cipher. In 1998 and 1999 the finalists were evaluated in public (and of course by NSA). The winner (Rijndael) became the FIPS 197 (Federal Information Processing Standard 197), commonly known as AES. The standard comprises three ciphers selected from Rijndael.
The authors were two Belgians, Dr Joan Daemen and Dr Vincent Rijmen (Rijndael is a play on their names).

The US Advanced Encryption Standard

The standard comprises three ciphers selected from Rijndael (AES-128, AES-192 and AES-256 with key sizes of 128, 192 and 256 bits). It is a symmetric block-based data encryption standard with a 128-bit block size. Designed to be resistant against known attacks, have speed and code compactness on many CPUs, along with design simplicity (so it can weather criticism).
It is not a feistel structure, instead it is state-based. The algorithm is normally specified in code form. Uses state, rounds, substitution, shifts, mixing, and roundkeys...

AES operations

10/12/14 rounds



No.of rounds	Key Length (bytes)
10	16
12	24
14	32

AES design

Transformations...

The main data structure is a data block of 4 columns of 4 bytes, and is called the state. The key is expanded to an array of (32-bit) words.
There is an initial XOR with the key (AddRoundKey), and then the number of rounds is dependent on the key size (10,12,14). Each round modifies the state:

- ***byte substitution:*** *one S-box used on every byte*
- ***shift rows:*** *permute bytes between groups/columns*
- ***mix columns:*** *substitutions using matrix multiply of groups*
- ***add round key:*** *XOR state with key material*

There is an incomplete last round (No MixColumns operation). All of the rounds can be done efficiently, using fast XOR operations, and table lookup.

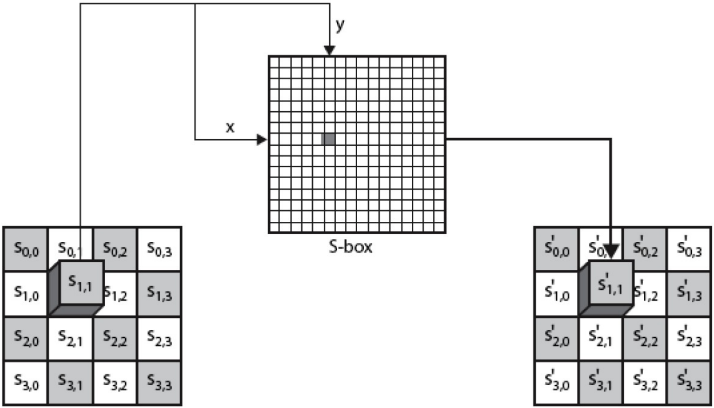
AES algorithm

AES code

```
Cipher(byte in[4*4],byte out[4*4],word w[4*(Nr+1)])
begin
byte state[4,4]
  state = in
  AddRoundKey(state, w[0, 3])
  for round = 1 step 1 to Nr-1
    SubBytes(state)
    ShiftRows(state)
    MixColumns(state)
    AddRoundKey(state, w[round*4, (round+1)*4-1])
  end for
  SubBytes(state)
  ShiftRows(state)
  AddRoundKey(state, w[Nr*4, (Nr+1)*4-1])
  out = state
end
```

AES operations...

Substitute bytes



GF(2^8) and AES

The byte substitution S-Box in AES:

The AES S-Box is built on polynomial computations done in GF(2^8), using the irreducible polynomial $x^8 + x^4 + x^3 + x + 1 = 100011011$.

The first step is to calculate the multiplicative inverse of the (non-zero) byte value in the field, giving an 8-bit value $\left[x^7 \ldots x^0\right]$.

The substitution is then

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x^0 \\ x^1 \\ x^2 \\ x^3 \\ x^4 \\ x^5 \\ x^6 \\ x^7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

The tables for all the byte values can be pre-computed.

AES avalanche - 1 bit change in plaintext

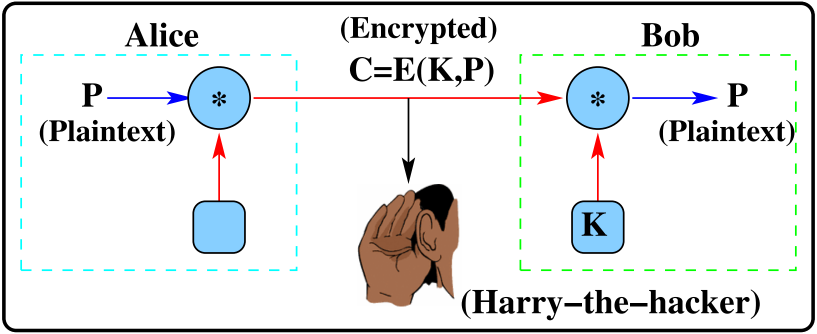
Number of bits changed/128 on each round

Round		Number of bits that differ
	0123456789abcdeffedcba9876543210 0023456789abcdeffedcba9876543210	1
0	0e3634aece7225b6f26b174ed92b5588 0f3634aece7225b6f26b174ed92b5588	1
1	657470750fc7ff3fc0e8e8ca4dd02a9c c4a9ad090fc7ff3fc0e8e8ca4dd02a9c	20
2	5c7bb49a6b72349b05a2317ff46d1294 fe2ae569f7ee8bb8c1f5a2bb37ef53d5	58
3	7115262448dc747e5cdac7227da9bd9c ec093dfb7c45343d689017507d485e62	59
4	f867aee8b437a5210c24c1974cffeabc 43efdb697244df808e8d9364ee0ae6f5	61
5	721eb200ba06206dcbd4bce704fa654e 7b28a5d5ed643287e006c099bb375302	68
6	0ad9d85689f9f77bc1c5f71185e5fb14 3bc2d8b6798d8ac4fe36a1d891ac181a	64
7	db18a8ffa16d30d5f88b08d777ba4eaa 9fb8b5452023c70280e5c4bb9e555a4b	67
8	f91b4fbfe934c9bf8f2f85812b084989 20264e1126b219aef7feb3f9b2d6de40	65
9	cca104a13e678500ff59025f3bafaa34 b56a0341b2290ba7dfdfbddcd8578205	61
10	ff0b844a0853bf7c6934ab4364148fb9 612b89398d0600cde116227ce72433f0	58

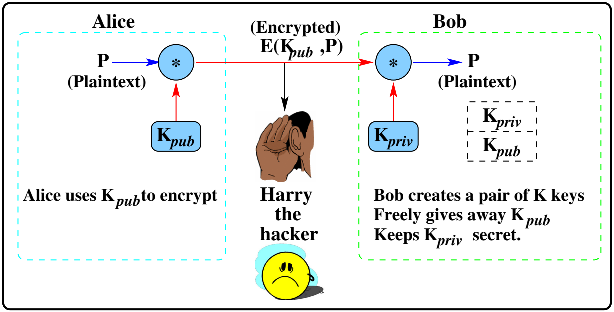
Outline

Symmetric key systems

Alice uses a key to send to Bob, who uses the same key...



A model for public/private keys



K_{pub} is public key for Bob, K_{priv} is his private key.
Only Bob can decrypt a message sent to him, but anyone can encrypt it.

Uses of asymmetric encryption

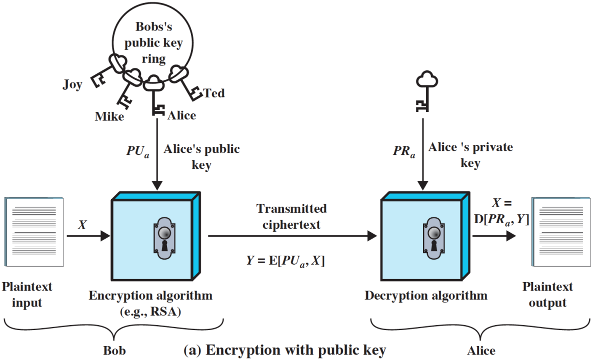
What use is asymmetric encryption?

1. Generating encrypted passwords with 1-way functions
2. Checking integrity by appending digital signature
3. Checking the authenticity of a message.
4. Encrypting timestamps with messages to prevent replay attacks.
5. Exchanging a key.

Note that...

Participants each have private and public keys, and that these two keys cannot be derived from each other

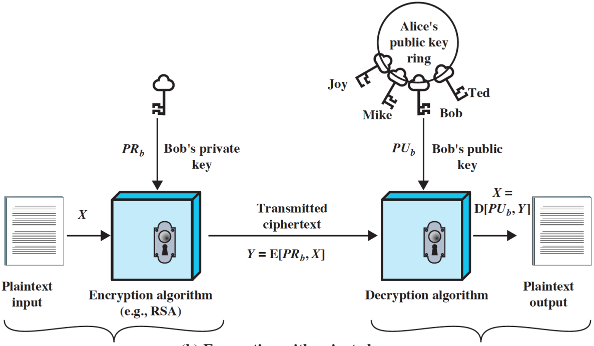
A model for public/private keys



(a) Encryption with public key

Asymmetric authentication

A model for asymmetric authentication



(b) Encryption with private key

Doing both...

Possible technique...

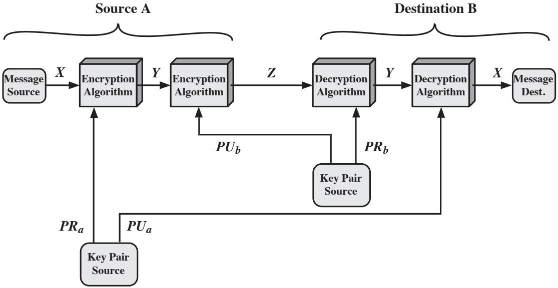
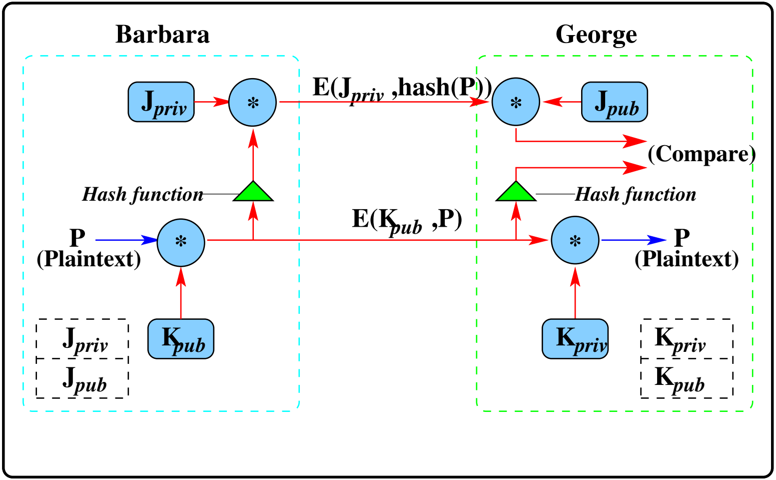


Figure 9.4 Public-Key Cryptosystem: Authentication and Secrecy

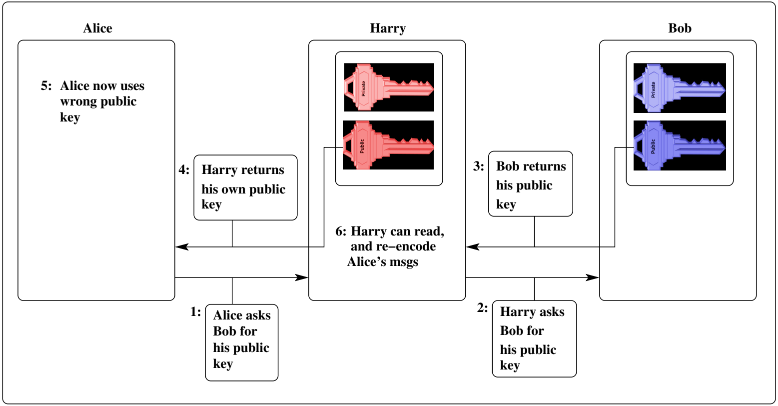
Doing both...

Actual technique...



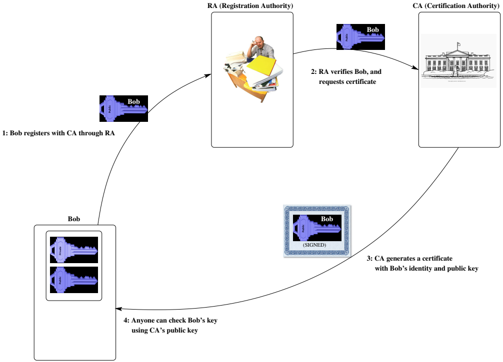
Man-in-the-middle for Public Keys

Motivation for PKI:



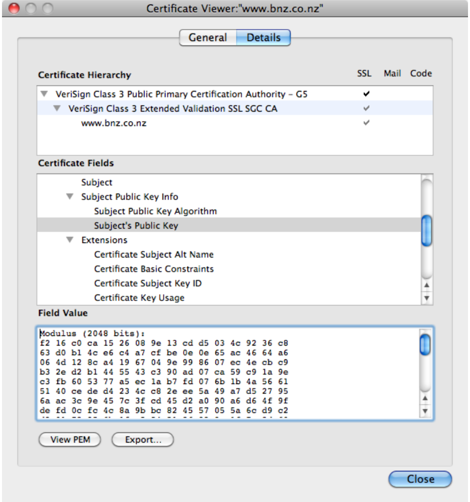
The certification mechanism

RA and CA:



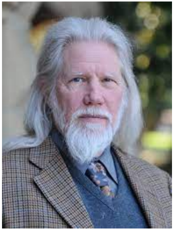
Certificates

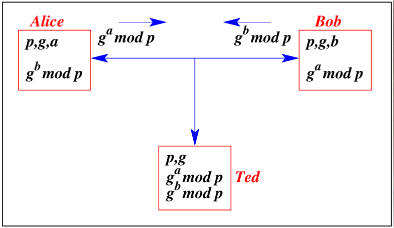
Viewing a signed certificate:

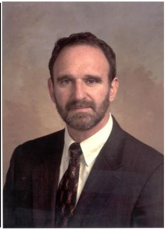


Outline

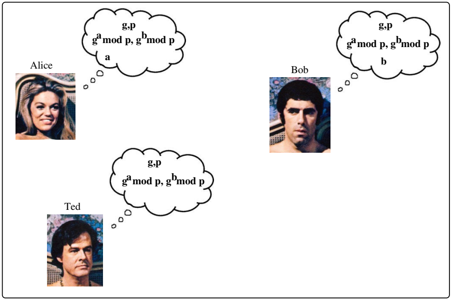
KA#1: Diffie (Whitfield)/(Martin) Hellman







After exchange, knowledge is different...



Construction #1: Diffie-Hellman (CDH) KA

Concrete: what do the parties do? (modulo math)

Init(): Alice and Bob pre-determine a choice of p, and a generator g.
DoProt():

1. Alice, Bob choose uniform values a, b from $\mathbb{Z}_p^*$ and exchange $h_A = g^a \bmod p$, $h_B = g^b \bmod p$ ($h_A \longrightarrow \text{Bob}$ and $h_B \longrightarrow \text{Alice}$).
2. Alice computes the secret $k_A = (g^b \bmod p)^a \bmod p = (g^b)^a \bmod p$.
3. Bob computes the secret $k_B = (g^a \bmod p)^b \bmod p = (g^a)^b \bmod p$.

Shared key is $k_A = k_B = (g^b)^a \bmod p = (g^a)^b \bmod p = g^{ab} \bmod p$.

Abstract: what do the parties do? ($\times$ cyclic group)

Init(): Alice and Bob pre-determine the choice of group $\mathcal{G}$ and a generator g.
DoProt():

1. Alice, Bob choose uniform values a, b from $\mathcal{G}$ and exchange $h_a = g^a$, $h_b = g^b$ ($h_a \longrightarrow \text{Bob}$ and $h_b \longrightarrow \text{Alice}$).
2. Alice computes the secret $k_A = h_b^a$.
3. Bob computes the secret $k_B = h_a^b$.

Note that $k_A = k_B = g^{ab}$. A common choice of group (as above) is $\mathcal{G} = \mathbb{Z}_p^*$.

Diffie-Hellman key agreement

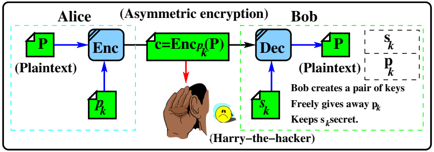
Forward function may be done in $\mathcal{O}(r)$:

Bit size	Forward	Reverse: Discrete logarithm solution
10	10	23
100	100	1,386,282
1,000	1,000	612,700,000,000,000,000,000,000

Relies on doing BIG number maths

- 1000 bit maths involves numbers with more than 300 decimal digits. The C "int" has 10 or so digits.
- To calculate $g^b \bmod p$ where g, b and p are small is easy, but we need some math tricks when they are large.
  - Why primes?
  - Fermat's little theorem

Asymmetric system #2: RSA



RSA is a well known public key encryption technique:

This public key system exploits the difficult problem of trying to find the complete factorization of a large composite integer whose prime factors are not known.

RSA can be used for encryption, digital signatures, and even key exchange.

The factorization problem

State of the art factorization

See `http://en.wikipedia.org/wiki/Integer_factorization`.

A 768 bits number (RSA-768) was factored in Dec 2009, using hundreds of machines over 2 years:

1230186684530117755130494958384962720772853569595334792197322452151726400507263657518745202299786469389956474942774063845925192557326303453731548268507917026122142913461670429214311602221240479274737794080665351419597459856902143413 = 334780716989568987860441698482126908177047949837137685689124313889828837938780022876147116525317430877378144679994 89 $\times$ 3674604366679959042824463379962795263227915816434308764267603228381573966651127923337341714339681027009279873630 8917

A Quantum computer can factor in polynomial time. In 2001, a 7-qubits quantum computer was built to factor 15.
In Nov 2007, D-Wave Systems announced a working 28-qubit computer: `http://www.nanowerk.com/news/newsid=3274.php`

4 processes for public key K_p and private key K_s

Step 1 - create public key

1. Select two large primes P and Q. Assign $x = (P - 1)(Q - 1)$.
2. Choose E relative prime to x. Assign $N = P * Q$.
3. K_p is N concatenated with E.

Step 2 - create private/secret key

1. Choose D: $D * E \bmod x = 1$ (i.e. multiplicative inverses)
2. K_s is N concatenated with D.

Step 3 - encoding

1. Pretend m is a number. Calculate $c = m^E \bmod N$.

Step 4 - decoding

1. Calculate $m = c^D \bmod N$.

Correctness and properties

Why does this work?

$$\begin{aligned} c^d \bmod N &= m^{ed} \bmod N \\ &= m^{k(p-1)(q-1)+1} \bmod pq \\ &= m \times m^{k(p-1)(q-1)} \bmod pq \\ &= m \end{aligned}$$

(See Euler's theorem if you cannot follow this).

Textbook RSA has some interesting properties

It is homomorphic w.r.t. multiplication - (this limits it in practice):

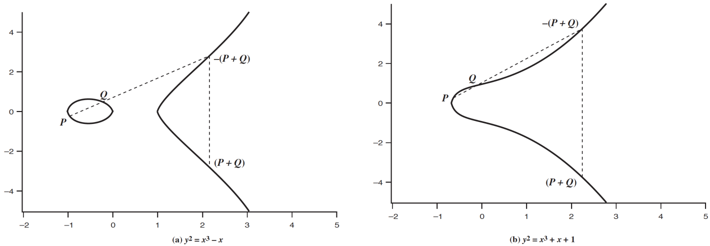
$$\mathrm{Enc}_{p_k}(m_1 \times m_2) = \mathrm{Enc}_{p_k}(m_1) \times \mathrm{Enc}_{p_k}(m_2) \bmod N$$

The above property is often called *Partially Homomorphic*, to distinguish it from *Fully Homomorphic*, that is, homomorphic w.r.t. both $+, \times$ in the ring.

Asymmetric system #3: ECC

Consider addition over cubic elliptic curves, such as

$y^2 = x^3 + ax + b$, with zero O, and the points $E(a,b) = \{(x_0,y_0),(x_1,y_1),\ldots\}$ on the curve. The elements are points on a plane not on a number line.



An addition operation $+_{E(a,b)}$ for points on this curve: the sum of $P +_{E(a,b)} Q$ is reflection of the intersection R. The group is $\langle +_{E(a,b)}, E(a,b) \rangle$.

ECC uses curves whose elements are finite: prime curves $E_p(a,b)$ defined over $\mathbb{Z}_p$, and binary curves $E_{2^m}(a,b)$ defined over $GF(2^m)$.

Algorithms for ECC cryptography

Step 1 - create Alices's private/secret key K_{s_A}

- Using an elliptic group $E_p(a, b)$, select a point G on the curve which has a large order n. The order of a point is the smallest value n such that $n \times G = 0$.
- Choose $n_A : n_A < n$. K_{s_A} is $\langle n_A, E_p(a, b), G\rangle$.

Step 2 - create Alice's public key K_{p_A}

Calculate $P_A = n_A \times G$. K_{p_A} is $\langle P_A, E_p(a, b), G\rangle$

Step 3 - encoding using Alice's public key K_{p_A}

Choose a random k, and calculate $C = \langle c_1, c_2\rangle = \langle k \times G, m + k \times P_A\rangle$

Step 4 - decoding using Alice's private key K_{s_A}

Calculate $m = c_2 - c_1 \times n_A$

Note that $m + kP_A - kGn_A = m + kn_AG - kGn_A = m$.

ECC adding: real and in $E_p(a, b)$

Adding points in $E(a, b)$ (Real curve)

If we had $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$, and $P \neq \pm Q$. We can find $R = P +_{E(a,b)} Q$ by finding gradient of line, and then intersection with curve:

$$\begin{aligned}
\text{Gradient:} \quad & \Delta = \frac{y_Q - y_P}{x_Q - x_P} \\
x \text{ coordinate for } R: \quad & x_R = \Delta^2 - x_P - x_Q \\
y \text{ coordinate for } R: \quad & y_R = \Delta(x_P - x_R) - y_P \\
\text{Finally:} \quad & R = (x_R, y_R)
\end{aligned}$$

$P +_{E(a,b)} P$ uses a different method

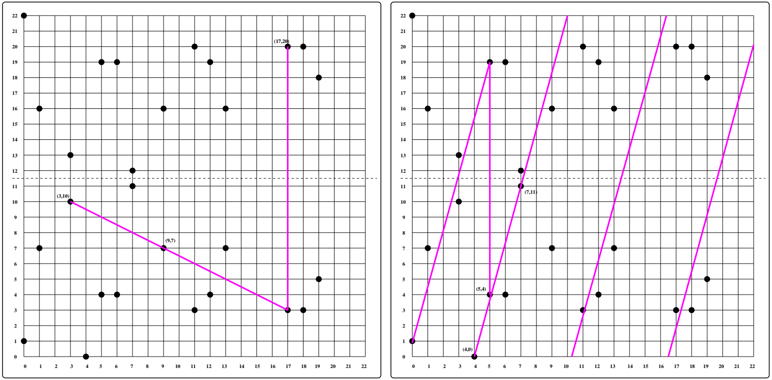
Adding points in $E_p(a, b)$ (Finite field)

We find R as before, modulo p:

$$\begin{aligned}
\text{Gradient:} \quad & \Delta = \frac{y_Q - y_P}{x_Q - x_P} \bmod p \\
x \text{ coordinate for } R: \quad & x_R = \Delta^2 - x_P - x_Q \bmod p \\
y \text{ coordinate for } R: \quad & y_R = \Delta(x_P - x_R) - y_P \bmod p \\
\text{Finally:} \quad & R = (x_R, y_R)
\end{aligned}$$

$P +_{E_p(a,b)} P$ again uses the different method.

Curve $y^2 = x^3 + x + 1 \bmod 23$ does not “look” nice



Example $P +_{E_{23}(1,1)} Q$ operations

P+Q	Gradient Δ	x coordinate x_R	y coordinate y_R	R
$(3,10)+(9,7)$	$= \frac{-3}{6} = 11$	$= 11^2 - 3 - 9 = 17$	$= 11(3-17) - 10 = 20$	$(17,20)$
$(4,0)+(7,11)$	$= \frac{11}{3} = 19$	$= 19^2 - 4 - 7 = 5$	$= 19(4-5) - 0 = 4$	$(5,4)$

Calculating $2P = P +_{E_P(a,b)} P$ for ECC

Doubling a point in $\langle +_{E_p(a,b)}, E_p(a,b) \rangle$

- If $y_P = 0$, return O, the zero point.
- $P = (x_P, y_P)$, and $y_P \neq 0$.
- Find $R = P +_{E_p(a,b)} P$ by finding gradient of the tangent, and then intersection with curve:

$$\begin{aligned}
\text{Gradient:} \quad & \Delta = \frac{3x_P^2 + a}{2y_P} \bmod p \\
x \text{ coordinate for } R: \quad & x_R = \Delta^2 - 2x_P \bmod p \\
y \text{ coordinate for } R: \quad & y_R = \Delta(x_P - x_R) - y_P \bmod p \\
\text{Finally:} \quad & R = (x_R, y_R)
\end{aligned}$$

All operations modulo 23...

P+P	Gradient Δ	x coordinate x_R	y coordinate y_R	R
$(7,11)+(7,11)$	$= \frac{10}{22} = 13$	$= 13^2 - 14 = 17$	$= 13(7-17) - 11 = 20$	$(17,20)$
$(9,7)+(9,7)$	$= \frac{14}{14} = 1$	$= 1^2 - 18 = 6$	$= 1(9-6) - 7 = 19$	$(6,19)$

On $\frac{a}{b} \bmod p$ calculation for ECC

Dividing by b is the same as multiplication by b^{-1}:

To calculate $\frac{a}{b} \bmod p$,

1. Use extended euclidean algorithm to calculate $b^{-1} \bmod p$.
2. Then multiply $a \times b^{-1} \bmod p$.

All operations modulo 23...

To calculate $\frac{7}{11} \bmod 23$

1. Calculate $11^{-1} \bmod 23 = 21$.
2. $7 \times 21 \bmod 23 = 9$

Why use ECC?

ECC: for signatures, encryption, key-exchange...

ECC addition is an analog of modulo multiply. ECC repeated addition is an analog of modulo exponentiation.

We have a "hard" problem, equivalent to the discrete log problem. Consider $Q = kP$, where Q, P belong to a prime curve... it is

- "easy" to compute Q given k, P but
- "hard" to find k given Q, P.

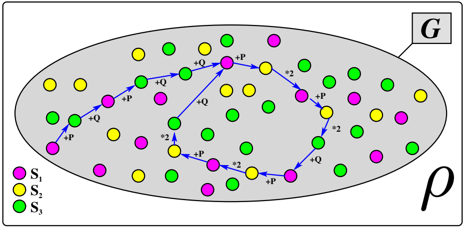
This is known as the elliptic curve logarithm problem.

Comparable key sizes (in bits)

Security	Symmetric	FFC: DSA	IFC: RSA	ECC,DH
128	AES-128	3072/256	3072	256
192	AES-192	7680/384	7680	384
256	AES-256	15360/512	15360	512

https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf

Best attack on ECC so far is Pollard-rho/Floyd!



Want to find $k : kP = Q$? Cast it as a collision!

The world is a chain of elements of the group, partitioned into three (S_1, S_2, S_3) types (coloured). We either double the element, or add P or Q:

$$P_{i+1} = f(P_i) \quad = \quad \begin{cases} P_i + P & \text{if} \quad P_i \in S_1 \\ 2P_i & \text{if} \quad P_i \in S_2 \\ P_i + Q & \text{if} \quad P_i \in S_3 \end{cases}$$

Each P_i is some sum $aP + bQ$. On collision, $a_1 P + b_1 Q = a_2 P + b_2 Q$, and

$$\frac{a_1 - a_2}{b_2 - b_1} P \quad = \quad Q$$

One way functions...

What we just saw were examples:

We use operations that are easy to do one way, say of $\mathcal{O}(k)$, and difficult to reverse and do the other way: perhaps $\mathcal{O}(e^k)$.

We want our mathematical systems to be

- of fixed size (i.e. modulo), and
- to operate over all values.

A suitable mathematical structure is to use finite cyclic groups, or fields.

Back to the high level view...

Attacks!

Can the keysize be reduced, perhaps by convincing systems to use a lower grade of encryption? This is the mechanism used by NSA to spy on HTTPS/SSL connections.

Can the key be brute-forced? Can some pre-computation scheme be used, storing the results on a disk?

Can the key be predicted? Keys are often generated as needed by generating numbers using pseudo random number generators.

Defences!

Do not downgrade encryption, or at least warn users if this is happening.

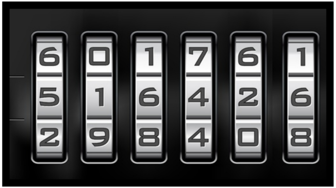
Use large sized keys, that are randomly generated, using high quality random number generators. If the key size is huge, neither brute-force, nor pre-computation would be feasible.

Perhaps use actual random number generator chips instead of pseudo random number generators.

Outline

1. Encryption...
   - Prehistory of crypto
   - Terms, definitions, goals
2. Modern symmetric ciphers
   - Symmetric cipher building blocks
   - DES
   - AES
3. From symmetric to asymmetric
   - Building blocks
   - Asymmetric systems: Diffie-Hellman, RSA and ECC
4. **The future...**
   - **Not just confidentiality: SS, MPC, IBE...**
   - The future of crypto

SS: Sharing a secret...



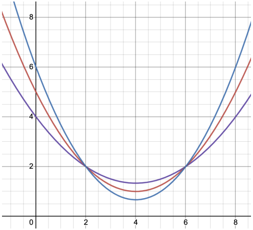
Are bank managers trustworthy? **Ch 13.3, pg 501**

Imagine that a bank has a vault that can be opened using a secret pin k. The secret k is to be kept by 3 managers. Only when three of them get together can the vault be opened. How should the secret k be shared among the three managers?

Here is a simple method. Suppose k is a 6 decimal digits sequence. The dealer splits it into 3 equal parts, i.e $k = s_1 \mathbin{+\!\!+} s_2 \mathbin{+\!\!+} s_3$. $\mathrm{Manager}_i$ will get the “share” s_i. Hence, each manager will keep 2 digits.

Question: Suppose $\mathrm{Manager}_1$ and $\mathrm{Manager}_2$ are malicious. They combine their shares and try to open the vault. How many combinations do they have to try in the worst case?

SS: Building blocks for a (t, w)-threshold scheme



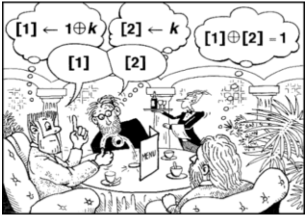
Intuition of Shamir's (t, w)-threshold scheme (1979)

The main idea is based on having sufficient points to fully define a polynomial curve. The dealer randomly picks a polynomial f of degree $t-1$ s.t. $f(0) = k$. The polynomial is over a finite field, say $\mathbb{Z}_p$, and each coefficient is uniformly chosen from $\mathbb{Z}_p$. Participant P_i gets the value $f(i)$.

When t participants get together, they have t samples of the polynomial f. With t samples, they can reconstruct the $t-1$ degree polynomial f and thus get the secret $f(0)$.

MPC: Computing values, anonymously

A single bit: The Dining Cryptographers...



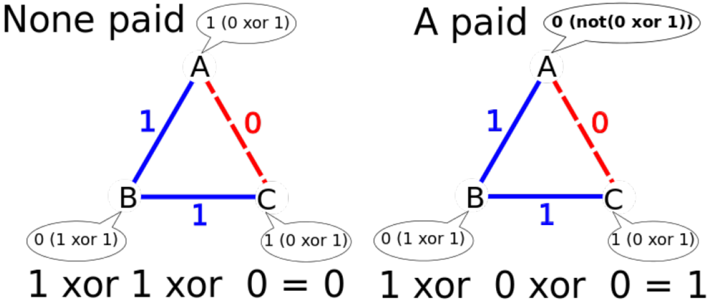
Consider a group of (3 or more) cryptographers, dining at a fancy restaurant. When they ask for the bill, the waiter tells them that the meal has been paid for, by someone who wishes to remain anonymous. The only other person who had been in the restaurant was from the NSA, so the cryptographers want to find out if the NSA guy paid, or if one of their own group paid.

Can they devise a protocol which lets them find out if one of their own group paid for the meal?

MPC: Anonymity protocol

The Dining Cryptographers...

Each pair of adjacent cryptographers decide on a randomly chosen bit (0/1) behind the menu (privately). If they did not pay for the meal, they announce the XOR of their two neighbours. Otherwise they announce the reverse.



The parity of the values is 1 if one of them paid for the meal.

Note that none of them know the other's "I paid" state.

IBE: Identity based encryption

Identity based encryption (IBE)?

It would be convenient if the name of an entity is its public key.

- Proposed by Shamir in 1984. However, construction of such a scheme remained an open problem for many years.
- Boneh/Franklin gave the first practical IBE based on *pairing* in 2002.

The scheme $\mathrm{IBE} = (\mathrm{Setup}, \mathrm{Enc}, \mathrm{Dec})$

We assume a central Trusted Authority (TA) with a master private key M_s, and public key M_p, securely distributed to each user. Each user has a name nm, perhaps their email address. Every user knows each other's names.

$\mathrm{Setup}(1^n)$: For each user U, TA generates the user private key k_U, and securely sends it to the user.

$\mathrm{Enc}_{\mathrm{nm},M_p}(m)$: Bob encrypts a message: $c \leftarrow \mathrm{Enc}_{\mathrm{nm},M_p}(m)$

$\mathrm{Dec}_{k_U,M_p}(c)$: Alice decrypts: $m \leftarrow \mathrm{Dec}_{k_U,M_p}(c)$

There is no PKI in this system. Bob doesn't need Alice's certificate. He simply takes her name (alice@comp.nus.edu.sg) to derive the public key.

IBE: remarks and a construction

Limitations of the scheme

1. It requires a trusted authority to generate all private keys. Since the TA knows the private keys, it can “forge” the signature of any user.
2. It is difficult to revoke a key.

Construction #5: Boneh/Franklin IBE using bilinear maps

$\mathrm{Setup}(1^n)$: Find a suitable bilinear map e with $\mathcal{G}, \mathcal{G}_T$, where $\mathcal{G}$ has order q. Let g be a generator of $\mathcal{G}$, and $h = e(g, g)$. Choose two suitable mappings $\mathcal{H}_1, \mathcal{H}_2$, where the range of $\mathcal{H}_1$ is $\mathcal{G}$, and the domain of $\mathcal{H}_2$ is $\mathcal{G}_T$. Uniformly pick a number $s \in \mathbb{Z}_q^*$ as the Master private key. Set the master public key $M_p = g^s$. Given a name $U = \mathrm{nm}$, the public key is $k = \mathcal{H}_1(\mathrm{nm})$ and the matching private key is $k_U = \mathcal{H}_1(\mathrm{nm})^s$.

$\mathrm{Enc}_{\mathrm{nm},M_p}(m)$: Randomly choose $r \in \mathbb{Z}_q$, compute $t := e(\mathcal{H}_1(\mathrm{nm}), M_p)$, and then output: $\langle c_1, c_2 \rangle \leftarrow \langle m \oplus \mathcal{H}_2(t^r), g^r \rangle$

$\mathrm{Dec}_{k_U,M_p}(\langle c_1, c_2 \rangle)$: Decrypt this way: $m \leftarrow c_1 \oplus \mathcal{H}_2(e(k_U, c_2))$

Outline

1. **Encryption...**
   - Prehistory of crypto
   - Terms, definitions, goals
2. **Modern symmetric ciphers**
   - Symmetric cipher building blocks
   - DES
   - AES
3. **From symmetric to asymmetric**
   - Building blocks
   - Asymmetric systems: Diffie-Hellman, RSA and ECC
4. **The future...**
   - Not just confidentiality: SS, MPC, IBE...
   - The future of crypto

Wrapping up

Applications: it is not just encryption!

Formal (mathematical) crypto has found its way into a range of application areas: **public-key encryption, signatures, IBE, MPC, MPKE,** non-interactive zero knowledge (NIZK) and non-interactive witness-indistinguishable (NIWI) proof systems, time-locked encryption, witness encryptions, 2-round MPC, attribute-based encryption, deniable encryption, functional encryption, quantum money...

Building blocks: It is not just factorizing and Dlog!

In a sense, the core of modern crypto centres on complexity ideas - hard vs easy, $\mathrm{P} \neq \mathrm{NP}$, and though the notions of the (hopefully hard) factorization and Dlog based constructions have worked so far, we already know of polynomial-time (quantum) solutions for most of the systems based on these. This drives the interest in other bases, perhaps ones with relevance in the post-quantum world.

A not-complete list of core hardness ideas that have been used, or could be applied: **factorization, Dlog, bilinear maps**, learning with errors (LWE), learning parity with noise (LPN), multilinear maps, iO ideas, latticed-based crypto...