*dit-upm*

# Cybersecurity Cryptography exercises

José A. Mañas < http://www.dit.upm.es/~pepe/>
Information Technology Department
Universidad Politécnica de Madrid

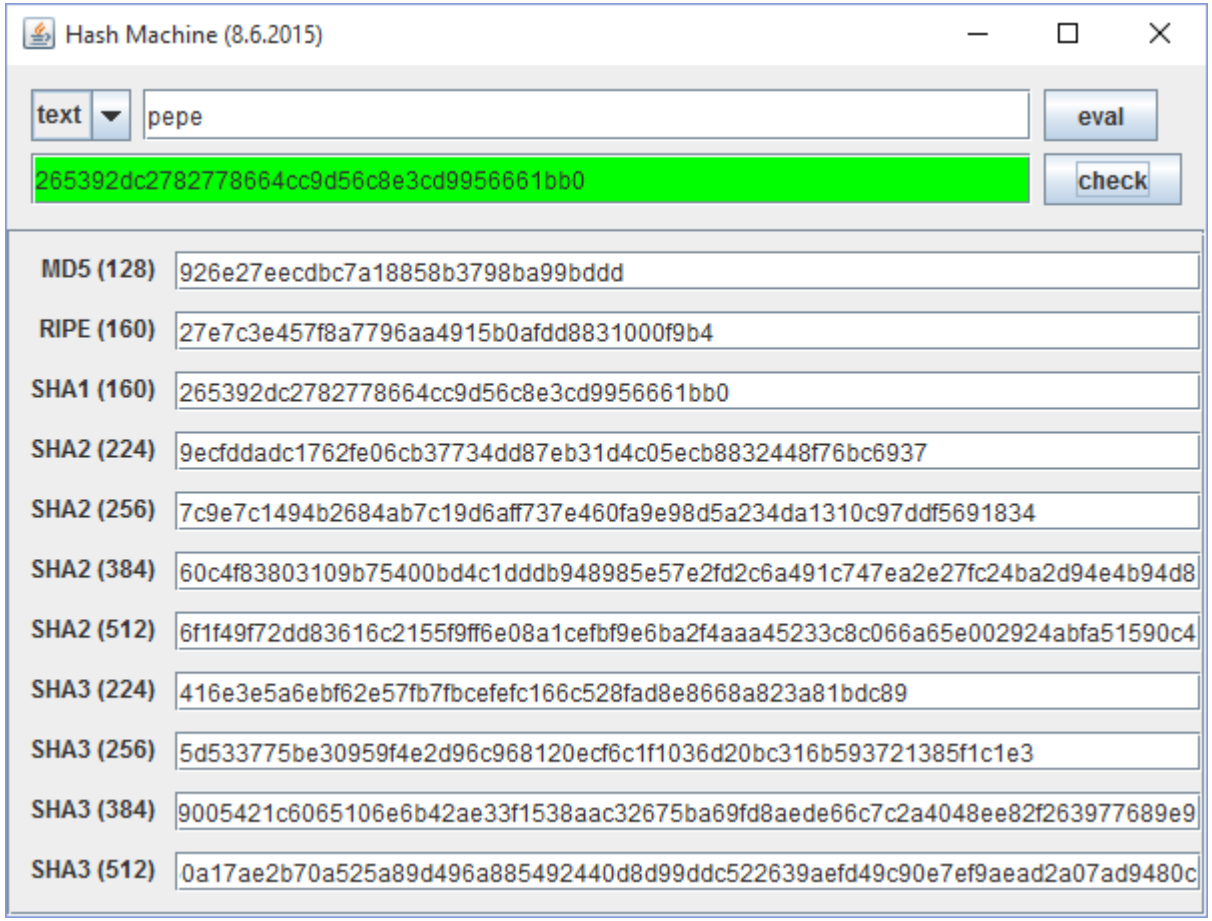
**10 October 2018**

# *dit* black ⊕ key → red

1. generate a key for a fake message
2. cryptanalyze a key used with several messages
3. cryptanalyze double lock exchange

# HashMachine

- http://www.dit.upm.es/~pepe/crypto/demos.html



# password storage

- comment pros & cons of each option
  1. save pw
  2. save H(pw)
  3. save <salt, H(salt || pw)>

- references
  - https://crackstation.net/
  - ask google “most common passwords”
- algorithms
  - PBKDF2 → bcrypt → scrypt

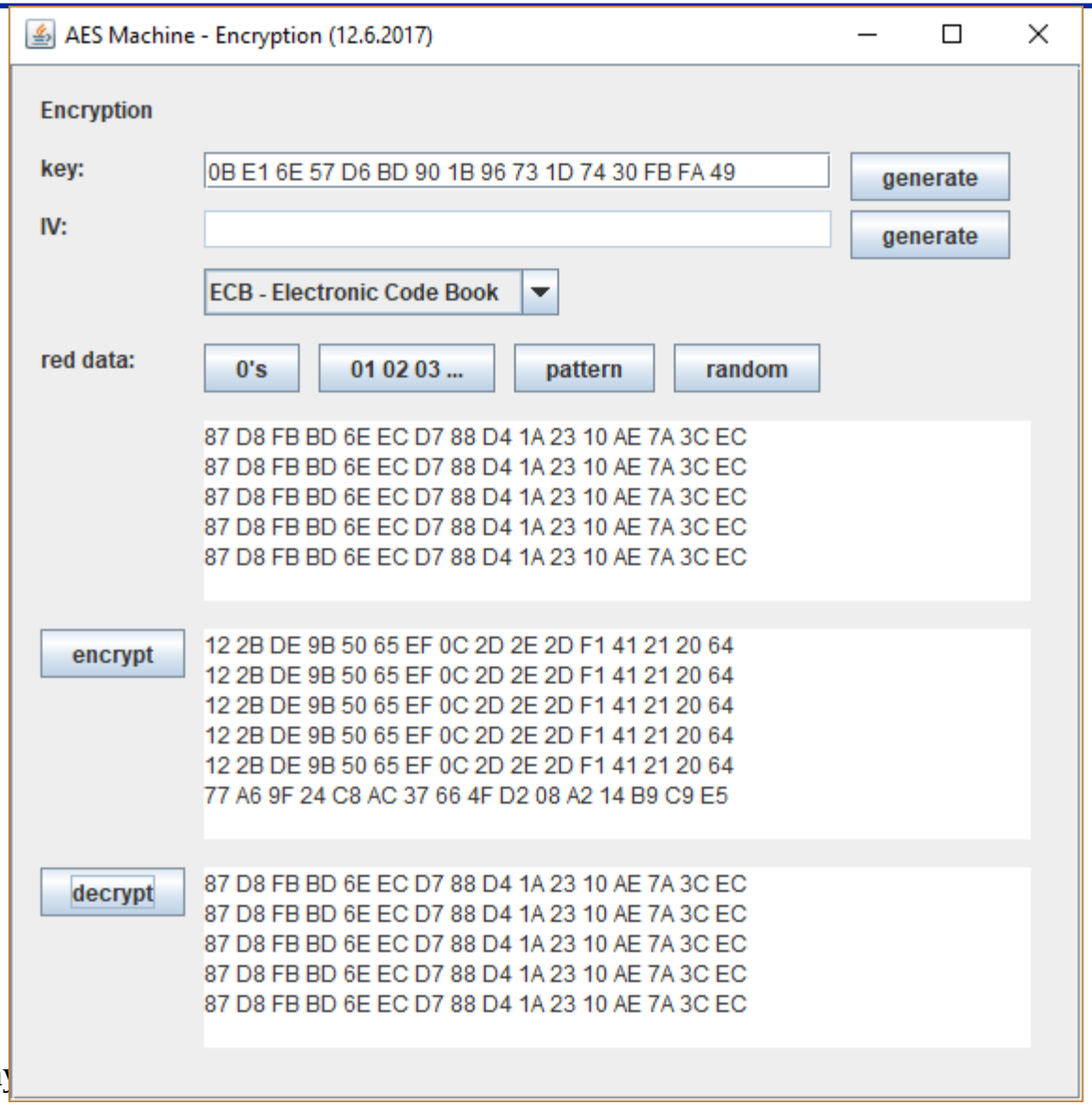
# use of hash functions

1. One-time password
   - user U chooses a secret S, sends $H^{1000}(S)$ to verifier V
   - for each login 1, 2, 3, …
     U sends < i, $H^{1000-i}(S)$ >
   - question 1: how would V validate U authenticity?
   - question 2: is it a useful I&A mechanism?

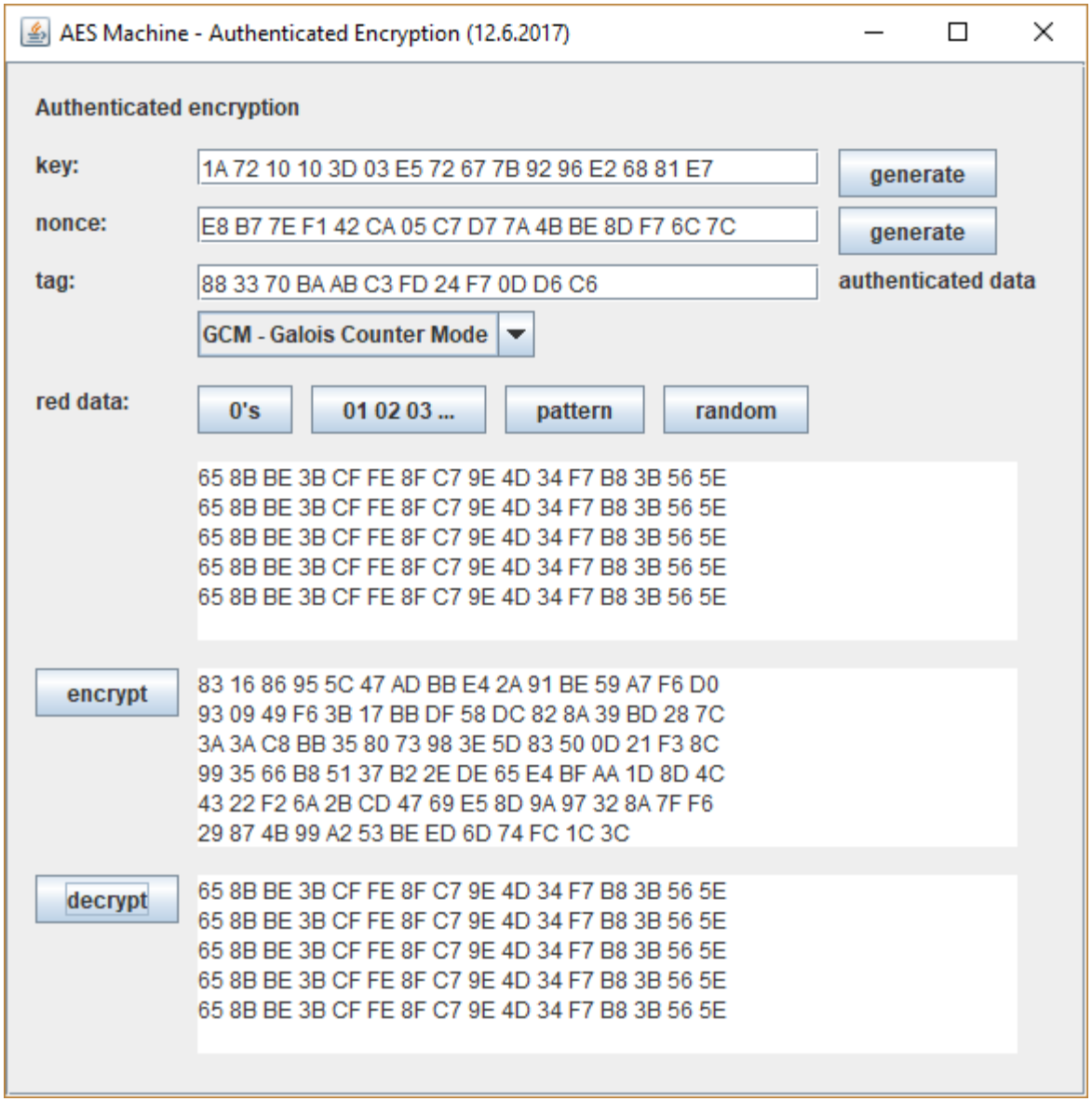
# use of hash functions

1. Hash-based encryption
   - let's have a secret key K
   - let's generate a bit-sequence $H(K) \| H^2(K) \| H^3(K) \| \ldots$ and use it as one-time pad xor'ed with the message
   - question 1: pros and cons, would you use it?
   - question 2: improvements?

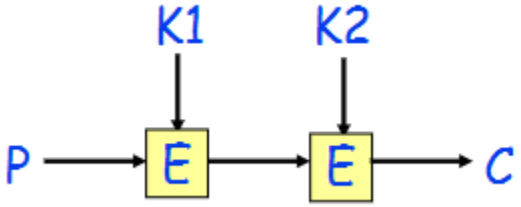
# AES demo


AES Machine - Encryption (12.6.2017)
Encryption
key:
0B E1 6E 57 D6 BD 90 1B 96 73 1D 74 30 FB FA 49
generate
IV:
generate
ECB - Electronic Code Book
red data:
0's
01 02 03 ...
pattern
random
87 D8 FB BD 6E EC D7 88 D4 1A 23 10 AE 7A 3C EC
87 D8 FB BD 6E EC D7 88 D4 1A 23 10 AE 7A 3C EC
87 D8 FB BD 6E EC D7 88 D4 1A 23 10 AE 7A 3C EC
87 D8 FB BD 6E EC D7 88 D4 1A 23 10 AE 7A 3C EC
87 D8 FB BD 6E EC D7 88 D4 1A 23 10 AE 7A 3C EC
encrypt
12 2B DE 9B 50 65 EF 0C 2D 2E 2D F1 41 21 20 64
12 2B DE 9B 50 65 EF 0C 2D 2E 2D F1 41 21 20 64
12 2B DE 9B 50 65 EF 0C 2D 2E 2D F1 41 21 20 64
12 2B DE 9B 50 65 EF 0C 2D 2E 2D F1 41 21 20 64
12 2B DE 9B 50 65 EF 0C 2D 2E 2D F1 41 21 20 64
77 A6 9F 24 C8 AC 37 66 4F D2 08 A2 14 B9 C9 E5
decrypt
87 D8 FB BD 6E EC D7 88 D4 1A 23 10 AE 7A 3C EC
87 D8 FB BD 6E EC D7 88 D4 1A 23 10 AE 7A 3C EC
87 D8 FB BD 6E EC D7 88 D4 1A 23 10 AE 7A 3C EC
87 D8 FB BD 6E EC D7 88 D4 1A 23 10 AE 7A 3C EC
87 D8 FB BD 6E EC D7 88 D4 1A 23 10 AE 7A 3C EC

# AES /GCM demo


AES Machine - Authenticated Encryption (12.6.2017)
Authenticated encryption
key:
1A 72 10 10 3D 03 E5 72 67 7B 92 96 E2 68 81 E7
generate
nonce:
E8 B7 7E F1 42 CA 05 C7 D7 7A 4B BE 8D F7 6C 7C
generate
tag:
88 33 70 BA AB C3 FD 24 F7 0D D6 C6
authenticated data
GCM - Galois Counter Mode
red data:
0's
01 02 03 ...
pattern
random
65 8B BE 3B CF FE 8F C7 9E 4D 34 F7 B8 3B 56 5E
65 8B BE 3B CF FE 8F C7 9E 4D 34 F7 B8 3B 56 5E
65 8B BE 3B CF FE 8F C7 9E 4D 34 F7 B8 3B 56 5E
65 8B BE 3B CF FE 8F C7 9E 4D 34 F7 B8 3B 56 5E
65 8B BE 3B CF FE 8F C7 9E 4D 34 F7 B8 3B 56 5E
encrypt
83 16 86 95 5C 47 AD BB E4 2A 91 BE 59 A7 F6 D0
93 09 49 F6 3B 17 BB DF 58 DC 82 8A 39 BD 28 7C
3A 3A C8 BB 35 80 73 98 3E 5D 83 50 0D 21 F3 8C
99 35 66 B8 51 37 B2 2E DE 65 E4 BF AA 1D 8D 4C
43 22 F2 6A 2B CD 47 69 E5 8D 9A 97 32 8A 7F F6
29 87 4B 99 A2 53 BE ED 6D 74 FC 1C 3C
decrypt
65 8B BE 3B CF FE 8F C7 9E 4D 34 F7 B8 3B 56 5E
65 8B BE 3B CF FE 8F C7 9E 4D 34 F7 B8 3B 56 5E
65 8B BE 3B CF FE 8F C7 9E 4D 34 F7 B8 3B 56 5E
65 8B BE 3B CF FE 8F C7 9E 4D 34 F7 B8 3B 56 5E
65 8B BE 3B CF FE 8F C7 9E 4D 34 F7 B8 3B 56 5E

# double encryption




which is the effective key length if …?

1. exercise 1: caesar cipher: c = (m + k) % 26
2. exercise 2: vernan cipher: c = k ⊕ m
3. exercise 3: c = H(k) ⊕ m
   - for k: 40 bits, H: 256 bits (SHA-256)
4. exercise 4: c = E(k, c);
   - e.g. DES (56 bits), AES (128, 192, 256 bits)

# demo: meet in the middle

- https://www.dit.upm.es/~pepe/crypto/MeetInTheMiddle.java
- https://www.dit.upm.es/~pepe/crypto/MeetInTheMiddle2.java

```
block size: 8
key size:   7

k1= 102; k2= 60
  157 -> 152

found: k1= 0; k2= 0
  157 -> [8] -> 152
found: k1= 46; k2= 0
  157 -> [8] -> 152
found: k1= 73; k2= 0
  157 -> [8] -> 152
found: k1= 96; k2= 0
  157 -> [8] -> 152
found: k1= 19; k2= 3
  157 -> [66] -> 152
found: k1= 72; k2= 3
  157 -> [66] -> 152
found: k1= 104; k2= 5
  157 -> [153] -> 152
…
```

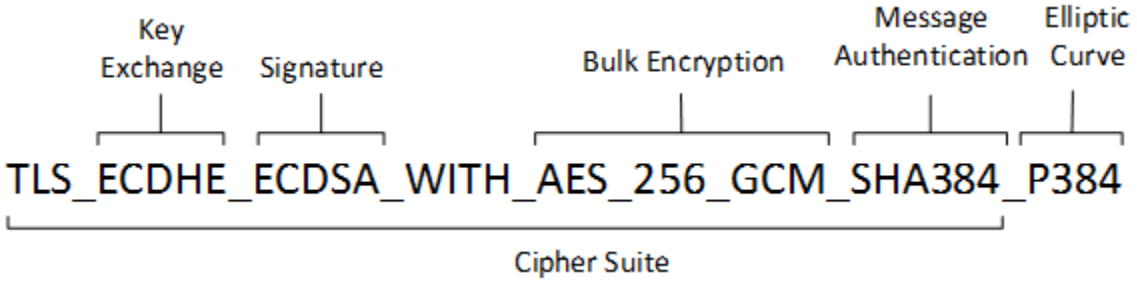
```
block size: 8
key size:   7

k1= 85; k2= 98
  157 -> 111
  105 -> 24

1 known pair: 53 matches
2 known pairs: 1 matches
```

# TLS

- cipher suites
- RFC 5246



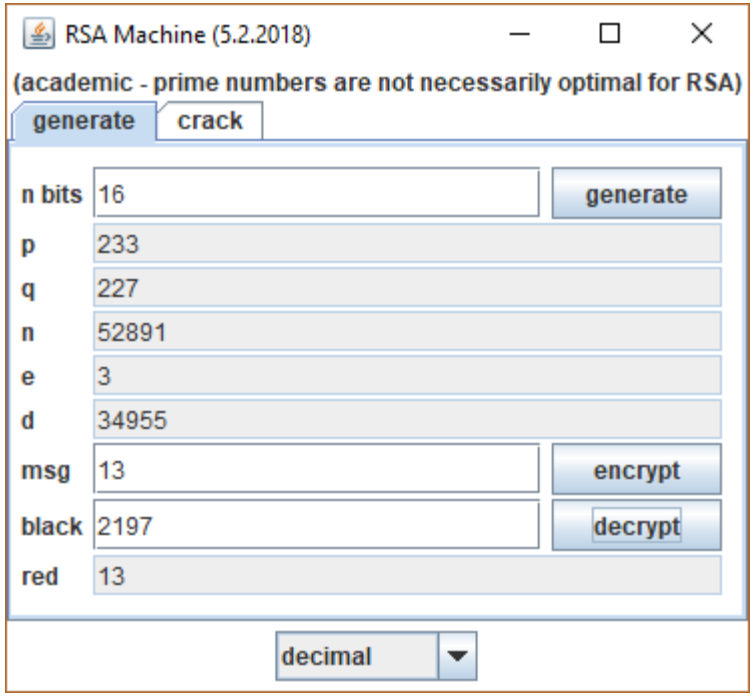

- server analysis
  - https://www.ssllabs.com/ssltest/
  - https://sslanalyzer.comodoca.com/
  - https://www.wormly.com/test_ssl
- client
  - https://www.howsmyssl.com/

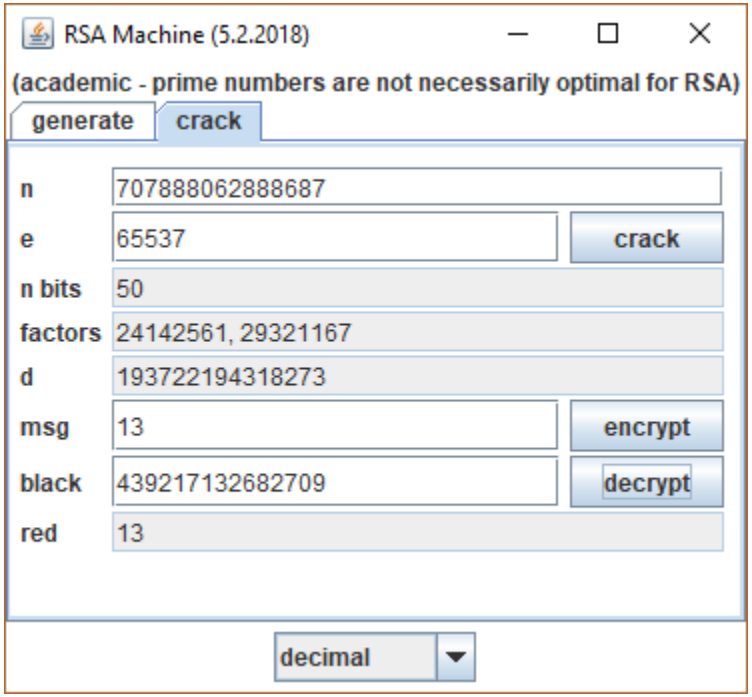
openssl
- openssl s_client
- openssl ciphers

# MyPGP

- https://sourceforge.net/projects/my-pgp/
- https://www.dit.upm.es/~pepe/mypgp/
- preinstalación
  - https://www.dit.upm.es/~pepe/mypgp/MyKeys.zip
- usage
  - https://www.dit.upm.es/~pepe/mypgp/videos/index_en.html
  - generate a key
  - encrypt / decrypt
  - sign / verify

# RSA Machine


RSA Machine (5.2.2018)
(academic - prime numbers are not necessarily optimal for RSA)
generate
crack
n bits
16
generate
p
233
q
227
n
52891
e
3
d
34955
msg
13
encrypt
black
2197
decrypt
red
13
decimal


RSA Machine (5.2.2018)
(academic - prime numbers are not necessarily optimal for RSA)
generate
crack
n
707888062888687
e
65537
crack
n bits
50
factors
24142561, 29321167
d
193722194318273
msg
13
encrypt
black
439217132682709
decrypt
red
13
decimal

# RSA

1. check validity of pub = (133, 40501), sec = (38797, 40501)
2. check validity of pub = (133, 40501), sec = (397, 40501)
3. check validity of pub = (133, 40501), sec = (16413, 40501)
4. design a key pair where p = 53, q = 113
   choose minimal e
5. design a key pair where p = 193, q = 163
   choose minimal e
   try e= 3, 5, 7, 9, 11, 13, …, 17, …, 25, ...
6. idem p= 193, q= 181

# RSA (exam)

- Home automation. We need reduced RSA keys. Specifically, we have a thermostat with this public reception key: {e = 7, n = 33}.
- The owner of the house sends it an order to set the temperature of the house to "31", where 31 is the desired temperature, encrypted with the previous RSA key.
- It is requested:
  1. Is the RSA key correct? What is its strength in bits?
  2. What temperature range can we mark?
  3. Break the key; that is, find out the private part to decipher.
  4. Find out at what temperature you want to set the thermostat. Decipher and verify.

# RSA

- Alice uses the RSA Crypto System to receive messages from Bob
- Alice publishes n = 299 and e = 35
  - Check that e = 35 is a valid exponent for the RSA algorithm
  - Compute d , the private exponent of Alice
- Bob wants to send to Alice the (encrypted) plaintext P=15
  - What does he send to Alice?
  - Verify she can decrypt this message

# RSA

- Alice publishes the following data
  - $n = pq = 221$ and $e = 13$
- Bob receives the message $M = 65$
  and the corresponding digital signature $S = 182$.
- Verify the signature

# diffie hellman

1. on shared parameters p= 1319, g= 69
    1. if A has secret 45, which is its public key?
    2. if B has secret 321, which is its public key?
    3. calculate agreed secret between A and B
2. check validity of
   public 1223, secret 123, for p= 1319, g= 69
3. find public key for p = 1319, g = 69
   secret = 100
4. find private key for p = 1319, g = 69
   public = 1232

# ECC

- let's use curve
  - p= 29
  - $y^2 = x^3 + 4x + 9$ ($F_{29}$)

1. points
   - is (4, 4) in the curve?
   - is (2, 5) in the curve?
2. calculate
   - (5, 3) + (12, 4)
   - 13 * (23, 28)

- using G = (0, 3)

1. A's secret = 3
   - calculate A's public key
2. B's secret = 23
   - calculate B's public key
3. calculate shared AB key
4. if X public key is (28, 2), find secret key

# dit

# ECC

- let's use curve
  - p= 263
  - $y^2 = x^3 + 2x + 3$ ($F_{263}$)

1. points
   - is (4, 4) in the curve?
   - is (61, 20) in the curve?
2. calculate
   - (61, 20) + (128, 182)
   - 3 * (61, 20)
   - 8 * (61, 20)

- using G = (200, 39)

1. A's secret = 3
   - calculate A's public key
2. B's secret = 23
   - calculate B's public key
3. calculate shared AB key
4. if X public key is (228, 106), find secret key

https://www.johannes-bauer.com/compsci/ecc/

# ECC

- let's use curve
  - p= 263
  - $y^2 = x^3 + 2x + 3$ ($F_{263}$)

1. points
   - is (4, 4) in the curve?
   - is (61, 20) in the curve?
2. calculate
   - (61, 20) + (128, 182)
   - 3 * (61, 20)
   - 8 * (61, 20)

```
>>> from ecc_pp import *
>>> C = ECC(263, 2, 3)
>>> P= Point(4, 4); C.valid(P)
False
>>> P= Point(61, 20); C.valid(P)
True
>>> C.add(Point(61, 20), Point(128, 182))
Point(x=217, y=66)
>>> C.mul(3, Point(61, 20))
Point(x=175, y=83)
>>> C.mul(7, Point(61, 20))
Point(x=175, y=180)
>>> for k in range(2, 30):
        print(k, C.mul(k, P))


2 Point(x=175, y=180)
3 Point(x=175, y=83)
4 Point(x=61, y=243)
5 Origin
6 Point(x=61, y=20)
7 Point(x=175, y=180)
8 Point(x=175, y=83)
9 Point(x=61, y=243)
10 Origin
11 Point(x=61, y=20)
12 Point(x=175, y=180)
```

ecc_pp.py

# *dit*

- inspect digital signature of this pdf document
  - certification chain
  - timestamp
  - LTV
- inspect certificate of
  - https://www.google.com
- generate a personal certificate (free)
  - https://www.cacert.org/