Cryptography[1]

Hans-Wolfgang Loidl
http://www.macs.hw.ac.uk/~hwloidl

School of Mathematical and Computer Sciences
Heriot-Watt University, Edinburgh

[1]Based on Goodrich's textbook, HAC, and Aspinall's slides

Stream Ciphers

Question

Recall Caesar's cipher: Why is it so easy to crack?

We can improve the strength of the Caesar cipher by:

- Performing a more general substitution of characters, rather than simple rotation. Keyspace grows from 26 to $26! = 403291461126605635584000000 =\tilde{} 4 \cdot 10^{27}$

Question

Does such increased keyspace make Caesar's cipher more secure?

- Using **different keys** on different characters.
- Transforming groups of characters, rather than individual characters.

Stream Ciphers

- A **stream cipher** encrypts a message **character by character**.
- The transformation that is applied typically varies over time.
- Stream ciphers are usually faster than block ciphers.
- They can be used even if the full message is not available, i.e. good for internet-style streaming.
- In some cases, hardware accelerators have been developed for stream ciphers, to speed up en-/de-cryption further.
- Because the handle character-by-character, they have limited error propagation, and transmission errors are less disruptive.

Vigenere Cipher

Like simple Caeser cipher, but use a repeated key to specify “k” for each character in plaintext:

```
key: abc = (k values 1, 2 3)

plaintext:  attack
key         abcabc..
k           123123

ciphertext: bvwben
```

Harder to decrypt. Can’t use character frequencies, and with long key, will take a long time to try all possible keys.

One-time pads

A one-time pad, or **Vernam cipher**, is an unbreakable stream cipher:

- Invented by Joseph Mauborgne and Gilbert Vernam in 1917.
- It uses rotations, similar to the Caesar cipher, but a **different** key for each element in the plain text
- This cipher is **provable secure** if keys are never reused.
- Reportedly, it has been used during the Cold War for secure communication between Washington and Moscow.
- However, it is **impractical** because it needs a sequence of keys as long as the message length
- Reportedly, key re-use happened quite often in communication between Soviet spies, allowing the US to attack their communication.
- To get around the problem of key re-use, sometimes **pseudo-random sequences** are used, which generate sequences of numbers that appear to be random, from a much smaller, secret key.
- For efficient implementation, this cipher is often used over binary numbers, performing an XOR as operation, i.e. $\mathcal{M} = \mathcal{C} = \mathcal{A} = \{0, 1\}$

Block ciphers

- A **block cipher** encrypts groups of characters in a message.
- The transformation is the same for each group.
- Pure block ciphers are memoryless, i.e. earlier data in a message does not influence the encryption of a later part of the message.

Basic design concepts for good ciphers:

- **Confusion:** A good cipher should add confusion, obscuring the relationship between the key and the ciphertext.
- **Diffusion:** A good cipher should add diffusion, spreading out redundancy in the plaintext across the ciphertext.
- Typically, modern block ciphers
  - use **substitution** to add confusion;
  - use **transpositions** to add diffusion;
  - apply rounds consisting of substitution and transposition steps to improve both.
- **Note:** a large keyspace does not guarantee a strong cipher, as we have seen with the generalised Caesar’s cipher.

Simple substitution ciphers

A simple substitution cipher is a block cipher for arbitrary block length *t*. It swaps each letter for another letter.
Caesar cipher is a simple substitution cipher with a fixed rotation.
In general, any **permutation** of the alphabet can be used, instead of a shift.

Simple transposition ciphers

The simple transposition cipher is a block cipher with block-length t. It simply permutes the symbols in the block.
To encrypt: Write the plaintext in columns of depth k (= key), padding (with *) the end as necessary, read off in rows, e.g. if key = 4.
Plaintext: `Transposition_ciphers_are_easy!`

```
T  s  i  n  p  s  e  s
r  p  t  _  h  _  _  y
a  o  i  c  e  a  e  !
n  s  o  i  r  r  a  *
```

Ciphertext: `Tsinpsesrpt_h__yaoiceae!nsoirra*`
To decrypt: write ciphertext as rows of (message length/k), and read off columns.

DES

Data Encryption Standard (DES) is a symmetric block cipher:

- DES is a block cipher based on Feistel's principle in order to provide effective confusion and diffusion. Block-size is 64 bits, key-size 56 bits (+8 parity bits). Invented by IBM in 1970s, tweaked by NSA. Still widely used, esp. in financial sector. Much analysed.
- The **Feistel principle** gives a way of constructing a cipher so that the same circuit is used for both encryption and decryption.
- Main threat isn't cryptanalytic, but (slightly optimised) exhaustive search in small key-space. Remedied by 3DES (triple DES), 3 keys:

  $$C = E_{k_3}(D_{k_2}(E_{k_1}(P))) \quad P = D_{k_1}(E_{k_2}(D_{k_3}(C))).$$

  Security of 3DES is not obvious: repeated encryption may not gain security (one-step DES is not closed, so it in fact does), and new attacks may be possible (meet-in-the-middle attack). With 3 independently chosen keys, security is roughly the same as expected with 2 keys.
- Several other DES variants, including DESX, using whitenening keys k_1, k_2 as $C = E_k(P \oplus k_1) \oplus k2$. (Used in Win2K encrypting FS).

The AES (or Rijndael) symmetric cipher

- Rijndael is a block cipher with a fixed block length of 128 bits.
- Since 2001 it is the AES standard for symmetric encryption set by the U.S. National Institute for Standards and Technology (NIST)
- It replaces the older DES cipher, which has a proven weakness.
- Rijndael can be used with key lengths of 128, 192 or 256-bits.
- Rijndael satisfied a number of requisite criteria for the AES:
  - **Security:** mathematical, cryptanalytic resistance; randomness;
  - **Efficiency:** time/space, hardware and software;
  - **Flexibility:** block sizes 128 bits, key sizes 128/192/256 bits.
  - **Intellectual property:** unclassified, published, royalty-free.
- Rijndael is built as a network of linear transformations and substitutions, with 10, 12 or 14 rounds, depending on key size.

Structure of the Rijndael (or AES) algorithm

- Initialise the state, by performing an XOR of plain-text and key
- Perform **10 rounds** of modifying the state
- The result is the state after these 10 rounds.

Each round consists of the following steps:

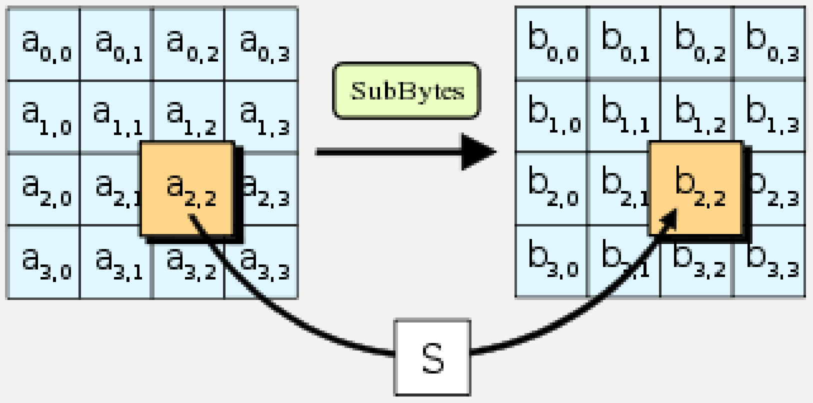
- **SubByte step:** perform an S-box substitution
- **ShiftRows step:** perform a permutation
- **MixColumns step:** a matrix multiplication step
- **AddRoundKey step:** an XOR operation with the round key, derived from the key

Note: Each step is invertible, so de-crypting amounts to running the algorithm in the reverse order.
Note: All steps operate on a **state**, which is a 4×4 matrix of the (8-bit) bytes of a block

SubByte Step

- **Input:** 4×4 matrix A
- **Output:** 4×4 matrix B, by performing a **component-wise operation** S-Box transformation S one each element of A.
- S computes for one byte of the input matrix the **multiplicative inverse** over the finite field $GF(2^8)$ combined with an invertible affine transformation.
- Importantly, this operation assures non-linearity of the transformation.
- This operation S can be performed efficiently, by performing a lookup in a 16×16 table.

SubByte Step

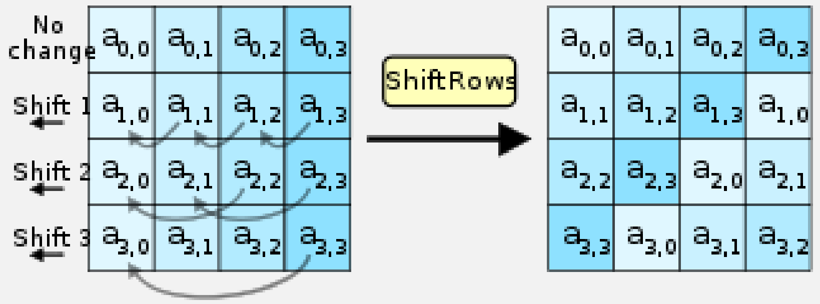

$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
SubBytes
S
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$

[1] From Wikipedia: Advanced Encryption Standard

ShiftRows Step

- **Input:** 4×4 matrix A
- **Output:** 4×4 matrix B, by performing a **permutation** on the elements of A.
- The permutation is done row-wise, **shifting** the 0-th row by 0, 1-st row by one etc.

ShiftRows Step

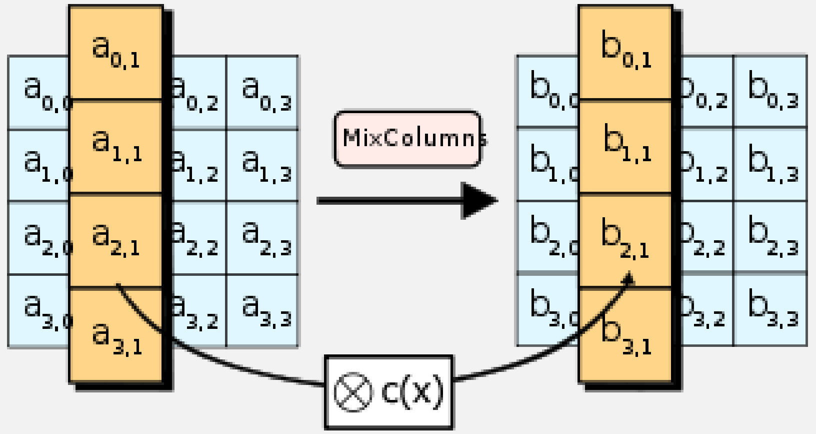

No change
Shift 1
Shift 2
Shift 3
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
ShiftRows
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,1}$ $a_{1,2}$ $a_{1,3}$ $a_{1,0}$
$a_{2,2}$ $a_{2,3}$ $a_{2,0}$ $a_{2,1}$
$a_{3,3}$ $a_{3,0}$ $a_{3,1}$ $a_{3,2}$

[1] From Wikipedia: Advanced Encryption Standard

MixColumns Step

- **Input:** 4×4 matrix A
- **Output:** 4×4 matrix B, by performing a Hill-cipher **matrix multiplication** of a fixed matrix E with A.
- The matrix multiplication uses an XOR operation to add two bytes, and uses a polynomial product, modulo a fixed base, for multiplication.
- Although this seems complicated, it can be implemented very efficiently.

MixColumns Step

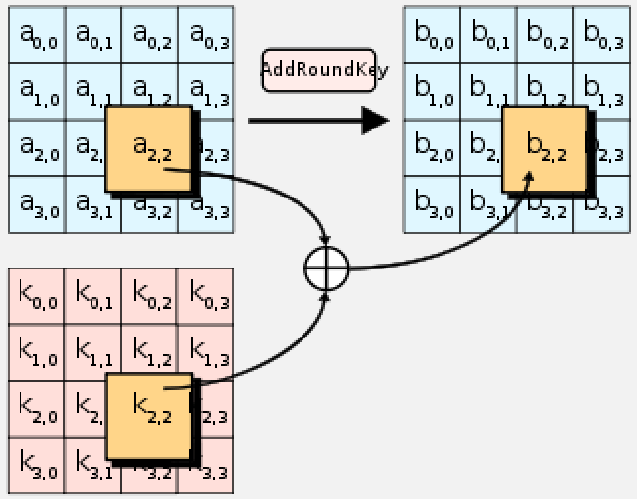


[1]From Wikipedia: Advanced Encryption Standard

AddRoundKey Step

- **Input:** 4×4 matrix A
- **Output:** 4×4 matrix B, by performing a **component-wise XOR** with a key set S.
- The key set S is a 4×4 matrix of keys derived from the main key.
- This derivation starts with the main key, and generates a **pseudo-random-number** sequence with it as seed value

AddRoundKey Step



[1]From Wikipedia: Advanced Encryption Standard

Properties of AES

To summarise, the main notable properties of AES are:

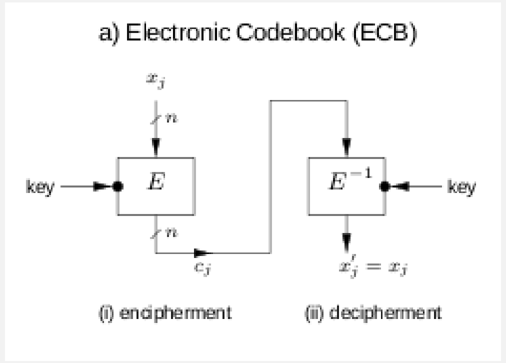
- It is a **symmetric block cipher**, with possible key lengths of 128, 192 or 256-bits.
- It is an iterative cipher, which applies the same sequence of operations in 10, 12 or 14 rounds.
- It performs permutations in order to prevent statistics-based attacks.
- All operations are invertible, so that for decryption the algorithm is run with the same key in reverse order.
- All operations are designed to be efficiently implementable.
- It is probably the “best” symmetric cipher today.

Attacks on AES

- As of 2010, the only known practical attacks on AES are side channel attacks, in particular timing attacks.
- A **side channel attack** observes the behaviour of the machine that performs en-/de-cryption.
- A **timing attack** observes the **time** the algorithm takes.
- Because the main operations in AES are table lookups, and because lookups have largely varying time when performed in memory or in cache, by timing the runtime of the algorithm on known plaintext, ciphertext pairs gives a possibility to learn the key.
- To defend against this weakness, the implementation of AES should provide constant time memory lookup.
- This can be done by disabling the cache, but this slows down en-/de-cryption a lot.

ECB: electronic codebook mode

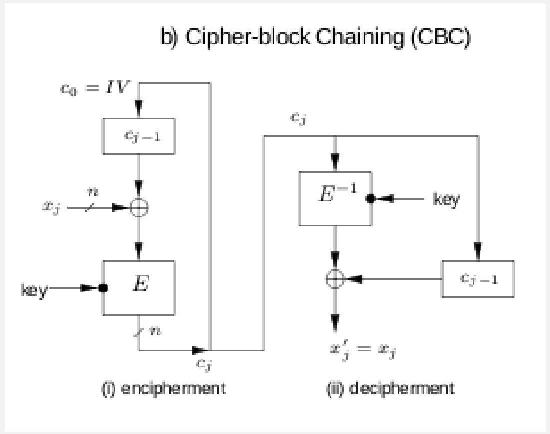
ECB: electronic codebook mode. Each block of plaintext x_j is **enciphered independently**: $c_j = E_k(x_j)$



[1]From: HAC, Chapter 8, p 229

CBC: cipherblock chaining mode

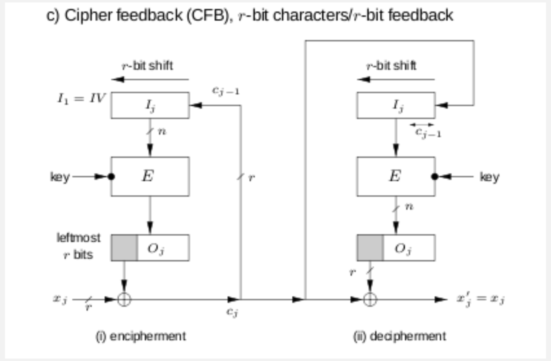
CBC: cipherblock chaining mode. Each plaintext block x_j is XORed with the previous ciphertext c_{j-1} block before encryption. An initialization vector (IV) (optionally secret, fresh for each message) is used for c_0: $c_j = E_k(x_j \oplus c_{j-1})$



[1]From: HAC, Chapter 8, p 229

CFB cipher-feedback mode

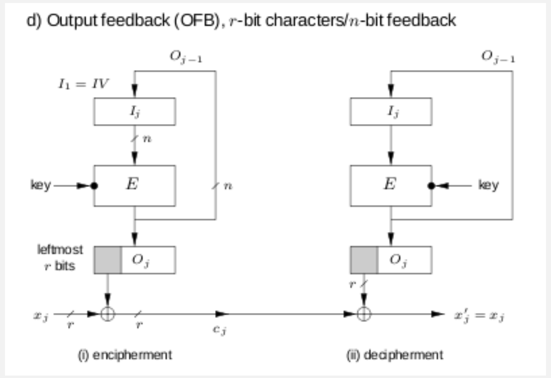
CFB cipher-feedback mode. Encryption function of block cipher used as self-synchronizing stream cipher for symbols of size up to block size, i.e. $c_j = E_k(c_{j-1}) \oplus x_j$



[1]From: HAC, Chapter 8, p 229

OFB: output-feedback mode

OFB: output-feedback mode. Conceptually, this mode works like a one-time pad, but with the sequence of blocks that are generated with the block cipher. Note, that the encryption function is only directly applied to the vectors, not directly to the plain-text:
$c_j = O_j \oplus x_j$ where $O_j = E_k(O_{j-1})$



[1]From: HAC, Chapter 8, p 229

CTR counter mode

CTR counter mode is similar to CFB mode in structure. Each element in the one-time-pad is computed directly from a seed value. Therefore, the entire sequence can be computed in parallel:
$c_j = O_j \oplus x_j$ where $O_j = E_k(s + j - 1)$

Summary of modes of symmetric key encryption

Different **modes** apply block-level encryption in different ways.

- **ECB: electronic codebook mode.** Each block of plaintext x_j is **enciphered independently**.

Question

Can you see a problem with this mode?

- **CBC: cipherblock chaining mode.** Each plaintext block x_j is XORed with the previous ciphertext c_{j-1} block before encryption. An initialization vector (IV) (optionally secret, fresh for each message) is used for c_0 .
- **CFB cipher-feedback mode.** Encryption function of block cipher used as self-synchronizing stream cipher for symbols of size up to block size.
- **OFB: output-feedback mode.** Block cipher encryption function used as synchronous stream cipher (internal feedback).

Exercises (optional)

Exercise

Symmetric encryption

- *Read Section 7.2.2. of HAC, for details on the different modes of block ciphers.*
- *Check Section 2.1 of Goodrich for details on symmetric encryption.*
- *Familiarise yourself with the `openssl` toolset installed on the Linux lab machines. These will be used in the next coursework.*

Limitations of symmetric encryption

- Symmetric encryption is fine to assure privacy of your own data, i.e. only you should be able to read the data
- In this case it's natural to have just one secret key
- But what if you want to securely transmit data and you want to assure that only the intended recipient should be able to read the message?
- You could use a secret key, known only to both of you.
- But what if you want to securely communicate with many recipients?

Question

If n participants want to communicate securely, making sure that only the 2 participants can read a message, how many secret keys are needed in total?

The idea of public-key encryption

Public-key encryption takes a radically different approach:

- Every participant in a communication has two keys
  - a **private key**, which is kept secret
  - a **public key**, which can be published e.g. on the web
- For safe communication, use the persons **public key** to **encrypt** the data.
- The recipient uses his **private key** to **decrypt**.
- Since the private key is kept secret, only he can read the message.
- This technology relies on the fact, that the private key cannot be efficiently computed from just knowing the public key (and the crypto algorithm).
- This idea was a **major research breakthrough** in the area of cryptography.

Features of public-key cryptosystems

Features of public key (PK) encryption:

- PK encryption provides privacy, ie. prevents unauthorised persons from reading the cipher-data
- PK encryption **does not** provide data origin authentication or data integrity
- A public key infrastructure (PKI) is used to distribute keys
- PKIs need separate techniques to ensure data origin authentication

Features of public-key cryptosystems (cont'd)

Advantages of public key encryption:

- No secret shared key is needed
- Needs only *n* keys (the public keys) for secure communication between *n* recipients.

Disadvantages of public key encryption:

- Encryption is significantly slower than symmetric key encryption
- The key length for public key cryptosystems is typically one order of magnitude larger than for symmetric cryptosystems (typically 2048-bit as opposed up to 256-bit for AES)

Browsers, like Firefox, use PK encryption only on **session keys**:

- Alice randomly generates a strong, symmetric key (a *session key*).
- Alice uses Bob's public key to encrypt and send the session key.
- Alice encrypts all plaintext for one session using the session key, and sends it.
- Bob uses his secret key to decrypt the session key.
- Bob now can decrypt all messages from Alice that use the same session key.

RSA encryption: a public-key cryptosystem

RSA encryption, a milestone in cryptography:

- RSA is the most commonly used public-key encryption systems
- It is named after the inventors: R. Rivest, A. Shamir, L. Adleman (**Turing Award 2002**).
- Its security is based on the intractability of **integer factorisation**.
- It is conceptually very simple, and founded on strong, mathematical properties.
- It can be used for encryption and for digital signatures.

RSA encryption in a nutshell

Alice wants to securely communicate with Bob, using RSA. First Bob generates a key-pair as follows:

- Bob picks two large, random **prime numbers** p, q. Let $n = pq$.
- Bob picks e that is **relatively prime** to $\Phi(n) = (p-1)(q-1)$.
- Bob computes the **inverse of e**, modulo $\Phi(n)$, i.e. $d = e^{-1} \mod \Phi(n)$
- Bob's public key is (e, n). He publishes this key, e.g. on the web.
- Bob's private key is d.

Now, Alice can use Bob's public key as follows:

- Alice encrypts the plaintext $\mathbf{M}$, using Bob's public key, by computing $\mathbf{C} = \mathbf{E}_{(\mathbf{e},\mathbf{n})}(\mathbf{M}) = \mathbf{M}^{\mathbf{e}} \mod \mathbf{n}$.
- **Note:** Encryption amounts to a single, modular exponentiation.
- Alice sends the ciphertext **C** to Bob.
- Bob decrypts the ciphertext **C**, using his private key, by computing $\mathbf{D}_{\mathbf{d}}(\mathbf{C}) = \mathbf{C}^{\mathbf{d}} \mod \mathbf{n}$.
- **Note:** Decryption amounts to a single, modular exponentiation.

Example of RSA

An example of RSA:

- Key generation:
  - Pick: $p = 19, q = 23$ and $n = p * q = 437$, ok because both are prime numbers
  - Pick: $e = 5$, ok because relatively prime to $\Phi(n)$ with $\Phi(n) = (p-1) * (q-1) = 396 = 2 * 3 * 11$
  - Compute: $d = e^{-1} \mod \Phi(n) = 5^{-1} \mod 396 = 317$
- **Encryption** of $m = 99$: $c = m^e \mod n = 99^5 \mod 437 = 17$
- **Decryption** of $c = 17$: $c^d \mod n = 17^{317} \mod 437 = 99 = m$

RSA Remarks

- RSA is an example of a reversible public-key encryption scheme. This is because e and d are symmetric in the definition. RSA digital signatures make use of this.
- RSA is a **deterministic** algorithm, i.e. if messages $M_1 = M_2$ then the cipher texts $C_1 = C_2$. Thus, RSA is often used with randomisation (e.g., salting with random appendix) to prevent chosen-plaintext and other attacks.
- RSA is the most popular and cryptanalysed public-key algorithm. Largest modulus factored in the (now defunct) RSA challenge is 768 bits (232 digits), factored using the Number Field Sieve (NFS) on 12 December 2009.
  - It took the equivalent of 2000 years of computing on a single core 2.2GHz AMD Opteron. On the order of 267 instructions were carried out.
  - Factoring a 1024 bit modulus would take about 1000 times more work (and would be achievable in less than 5 years from now).
  - Thus, key lengths of **at least 2048 bits** are recommended.

Security of RSA

- The security of RSA is based on the difficulty of finding the private key d, from the public key (e, n).
- If an attacker knew $\Phi(n)$ it would be easy to compute d, because $d = e^{-1} \mod \Phi(n)$ which can be computed using the extended Euclidean algorithm.
- **Note:** p, q, with $n = pq$, and therefore $\Phi(pq)$ are only needed in the key generation. They can, and should, be destroyed thereafter.
- But, if the attacker can factor $\Phi(pq)$, he can compute $p - 1$ and $q - 1$, and from that d.
- Thus, conceptually the security of RSA is tied to the complexity of factoring a large integer number.

Exercises

Exercise

RSA encryption:

1. *Perform RSA encryption of the message* 4*, using the public key* $(5, 91)$.
2. *In order to, crack the public key* $(5, 91)$*, which concrete computation do you have to perform.*
3. *Use the cracked private key to decrypt the message produced in (1).*

Suggested Exercises (optional)

Exercise

Download and install modules as described in `http://www.macs.hw.ac.uk/~hwloidl/Courses/F21CN/Labs/exercises.html`*. Launch* `caesar.hs` *as described in its header. Once launched, run the* `test` *command. Then, edit* `test`*, to pick the* 5*-th prime number for p and* 6*-th prime number for q and rerun test. Observe how the keys are picked, the message is encrypted and decrypted. Which of the values is the public, which of the values is the secret key? Which computation is performed for encryption, which for decryption?*

Social and Ethical Issues

Encryption world is secretive: often organisations don't reveal if a code can be broken.
Governments

- monitor communications, e.g. GCHQ and phone satellites scan phone calls and emails for keywords like 'bomb' 'Columbia', 'CIA'
- keep transmission traces
- seek to break codes

Individuals have a right to **privacy**, and commonly available software, e.g. PGP freeware, secure communication, encrypted disks: disliked by governments.
In the context of **terrorism**, crime, drug trafficking and money laundering should secure communication be widely available?

Summary

- Simple **symmetric** cipher: Caesar cipher, columnar transposition, Vernam cipher
- Simple codes and ciphers can be broken using frequencies, cribs, brute force.
- Pseudo-keys: long keys generated from shorter ones; harder to break.
- Standards for strong symmetric ciphers: **DES, AES**.
- Different modes for symmetric encryption have different strengths in hiding patterns.
- **Asymmetric** ciphers, e.g. RSA public key encryption, don't need to transfer secret key.
- With **RSA**, en/de-cryption amounts to **modular exponentiation**.
- WWW: public key encryption of session keys.
- Social and Ethical Issues.