PB173 - Tématický vývoj aplikací v C/C++ (podzim 2014)

Skupina: Aplikovaná kryptografie a bezpečné programování

https://is.muni.cz/auth/el/1433/podzim2013/PB173/index.qwarp?fakulta=1433;obdobi=5983;predmet=734514;prejit=2957738;

Petr Švenda svenda@fi.muni.cz

Konzultace: A406, Monday 15-15:50

CROCS

Centre for Research on Cryptography and Security

Optimizing crypto

Optimizing crypto

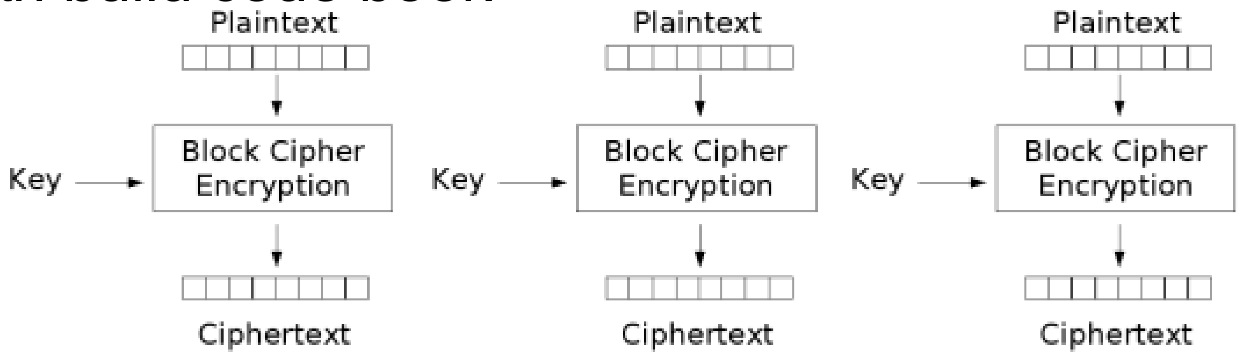
- Clever tricks both on design and implementation
  - optimization of both algorithm and mode used
  - see aestab.h and aesopt.h for example (Gladman)
- Possibility for pre-computation
  - code itself: macros, templates, static arrays
  - pre-computed tables
    - AES optimized with large tables
    - table lookup only implementation (AES/DES)
    - see http://cr.yp.to/aes-speed.html
  - pre-computed key stream (if mode supports)
    - key stream in advance, then simple xor

Parallelization of operations

- Speedup by parallel execution
- Purpose build hardware
  - cryptographic coprocessors
  - e.g., fast modulo exponentiation
- Using multiple CPU cores
  - multiple threads running
  - http://msdn.microsoft.com/en-us/library/69644x60%28v=VS.80%29.aspx
  - use so-called worker threads **`AfxBeginThread(mywork)`**

Mode of cipher usage - ECB

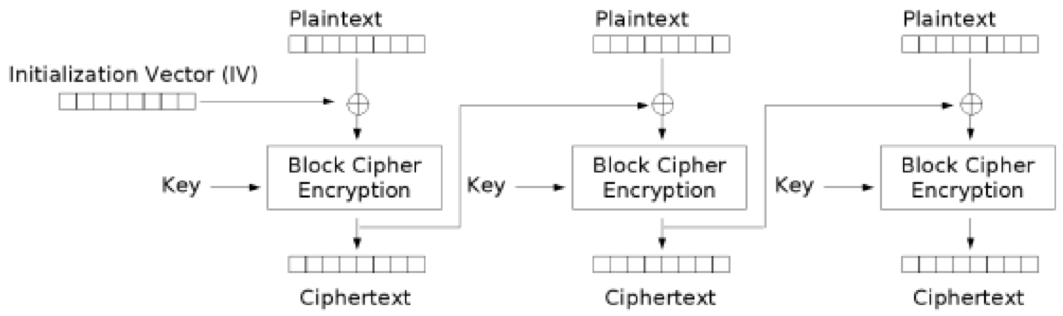
- ECB (Electronic Code Book mode)
  - used for block ciphers
  - processing of one block does not influence others
- Main problem
  - same data with same key result in same ciphertext
  - attacker can build code book



Electronic Codebook (ECB) mode encryption

Mode of cipher usage - CBC

- CBC (Cipher Back Chaining mode)
  - used for block ciphers
  - previous cipher block is xored with next plaintext
  - prefer before ECB
- Problem
  - Initialization vector must be somehow shared
  - (random first block)



Cipher Block Chaining (CBC) mode encryption

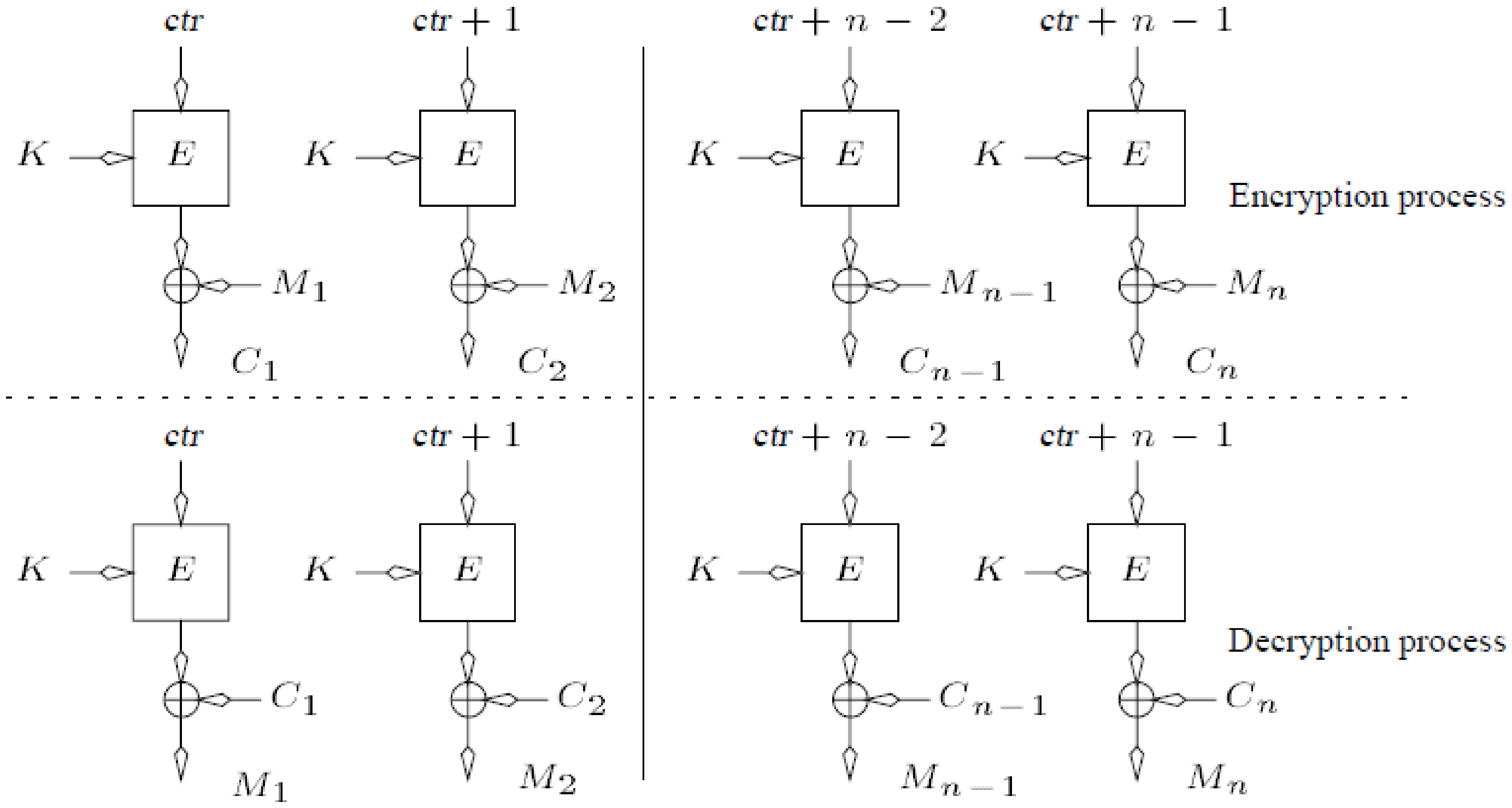
Parallelization of modes

- Assume that algorithm itself is sufficiently optimized
- Algorithm is used in some mode
  - e.g., block encryption modes (ECB, CBC…)
- We need parallelizable modes!
  - CBC encryption is not parallelizable
    - (decryption is – why?)
- Counter (CTR) mode

Counter (CTR) mode for encryption

- Mode approved by NIST (US standardization)
  - http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf
- Designed for confidentiality with parallelization and pre-computation in mind
- Key stream is produced by iterated encryption of the incremental counter
  - counter is incremented for each new block
  - key stream is then xored to message
  - **key stream(=counter) must never repeat with a same key**

Counter (CTR) mode for encryption



http://www.mindspring.com/~dmcgrew/ctr-security.pdf

Practical assignment

- Create robust packet protection between clients
  – packet encryption and authentication
  – reasonable key distribution (session keys, no fixed master key only)
- Speed-up encryption of data packets between two clients with CTR mode
  – pre-compute key stream (e.g., 100MB in RAM array)
  – use parallel threads to prepare key stream
    - number of available cores is parameter for function
    - at least one thread required ;)

Practical assignment

- Document performance gains
  - speed before and after the optimization
  - account correctly for key stream pre-computation
- Write tests for continuous transmission of large data between clients
  - (Used also or performance documentation)