PB173 - Tématický vývoj aplikací v C/C++ (Jaro 2016)
Domain specific development in C/C++

Skupina: Aplikovaná kryptografie a bezpečné programování

https://is.muni.cz/auth/predmety/uplny_vypis.pl?fakulta=1433;obdobi=6184;predmet=788705

Petr Švenda svenda@fi.muni.cz

Konzultace: A406, Pondělí 15-15:50

CROCS

Centre for Research on
Cryptography and Security

Block cipher modes for Authenticated Encryption

Modes for authenticated encryption

- Encryption preserves confidentiality but not integrity
- Common integrity functions (like CRC) protect against random faults
- Cryptographic message integrity protects intensional errors

Confidentiality, integrity, privacy

- Message confidentiality [encryption]
  - attacker is not able to obtain info about plaintext
- Message integrity [MAC]
  - attacker is not able to modify message without being detected (PTX, CTX)
- Message privacy [encryption]
  - attacker is not able to distinguish between encrypted message and random string
  - same message is encrypted each time differently

Encryption and MAC composition

- Modes for block ciphers (CBC, CTR, CBC-MAC)
- Compositions (encryption + MAC)
  - encrypt-and-mac [$E_{Ke,Km}(M) = E_{Ke}(M) \mid T_{Km}(M)$]
    - can fail with privacy and authenticity
  - mac-then-encrypt [$E_{Ke,Km}(M) = E_{Ke}(M \mid T_{Km}(M))$]
    - can fail with authenticity
  - encrypt-then-mac [$E_{Ke,Km}(M) = E_{Ke}(M) \,||\, T_{Km}(E_{Ke}(M)$]
    - always provides privacy and authenticity
- Paralelizability issue
- Authenticated-encryption modes (AE)
  - special block cipher modes for composed process

Usage scenarios

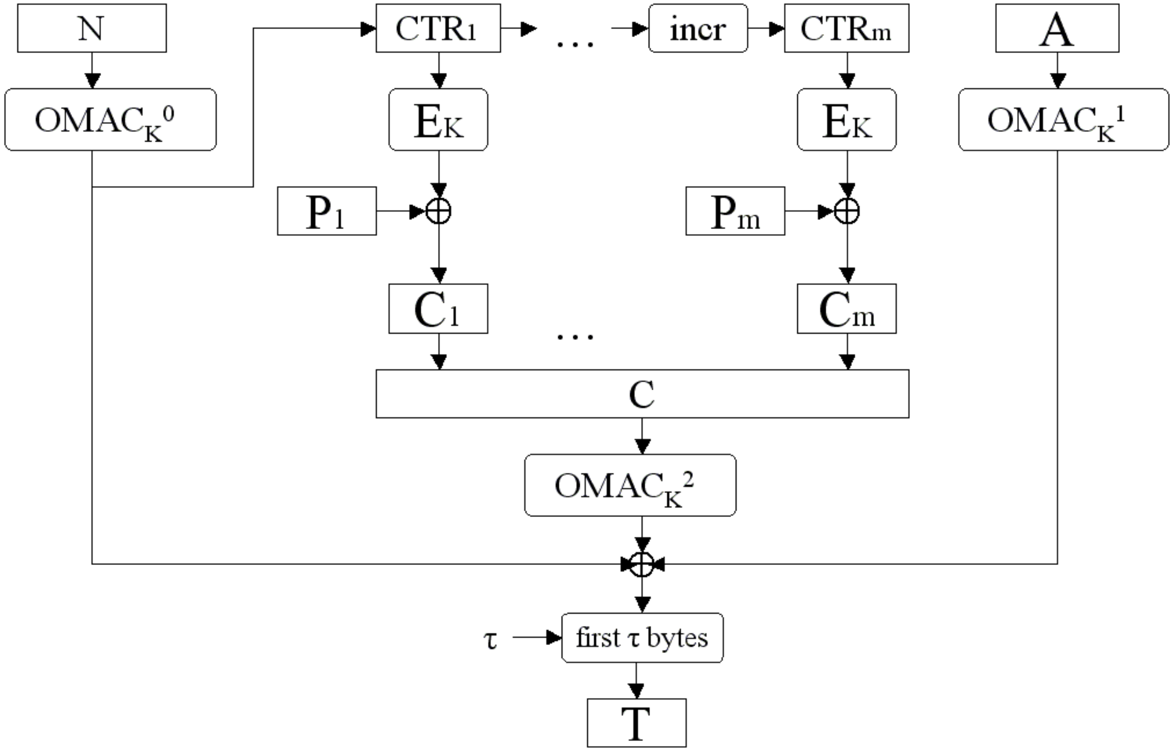
- Powerful, parallelizable environments
  - hardware accelerators
- Powerful, but almost serial environments
  - personal computer, PDA
- Restricted environments
  - smart card, cellular phone
- Different scenarios have different needs

Important features for AE modes

- Provable security
- Performance, paralelizability, memory req.
  - important for high-speed encryption, SC
- Patent
  - early AE modes were patented
- Associated data authentication
  - authentication of non-encrypted part
- Online, incremental MAC, number of keys, endian dependency …
- http://blog.cryptographyengineering.com/2012/05/how-to-choose-authenticated-encryption.html
- www.fi.muni.cz/~xsvenda/docs/AE_comparison_ipics04.pdf

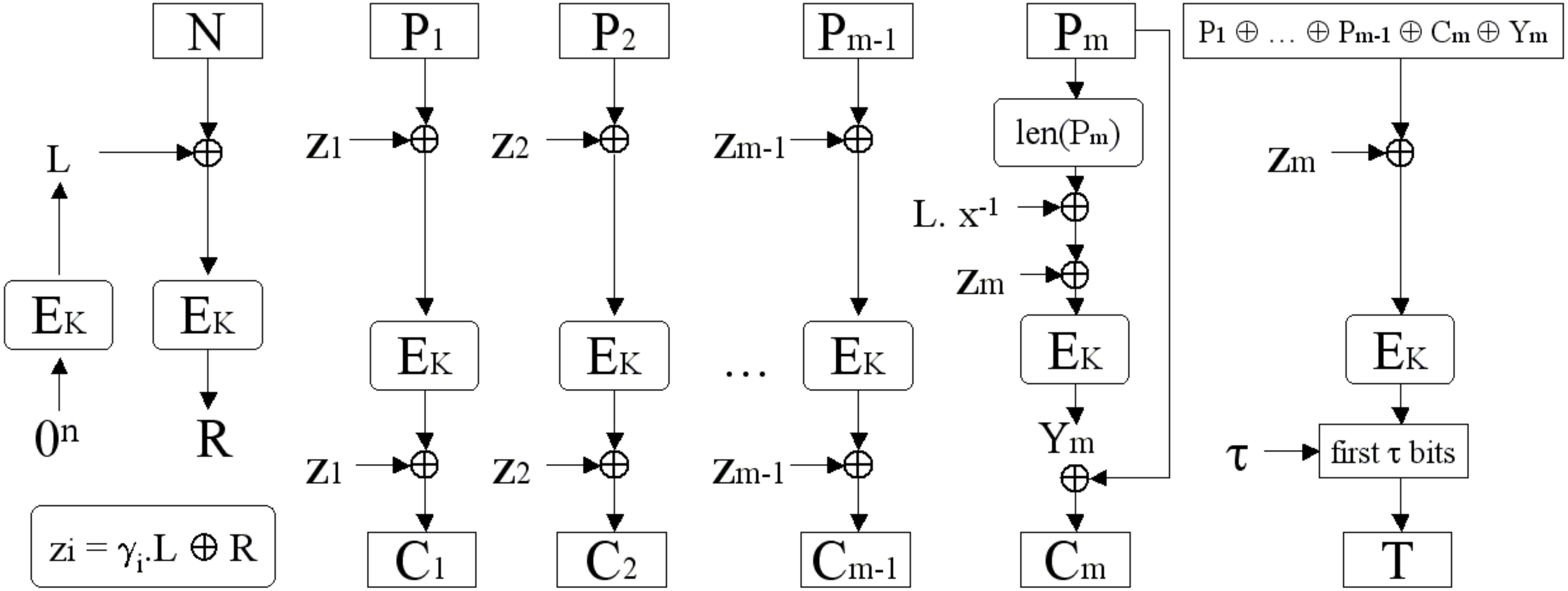
EAX mode

- Encrypt-than-mac composition
- Provable secure, unpatented



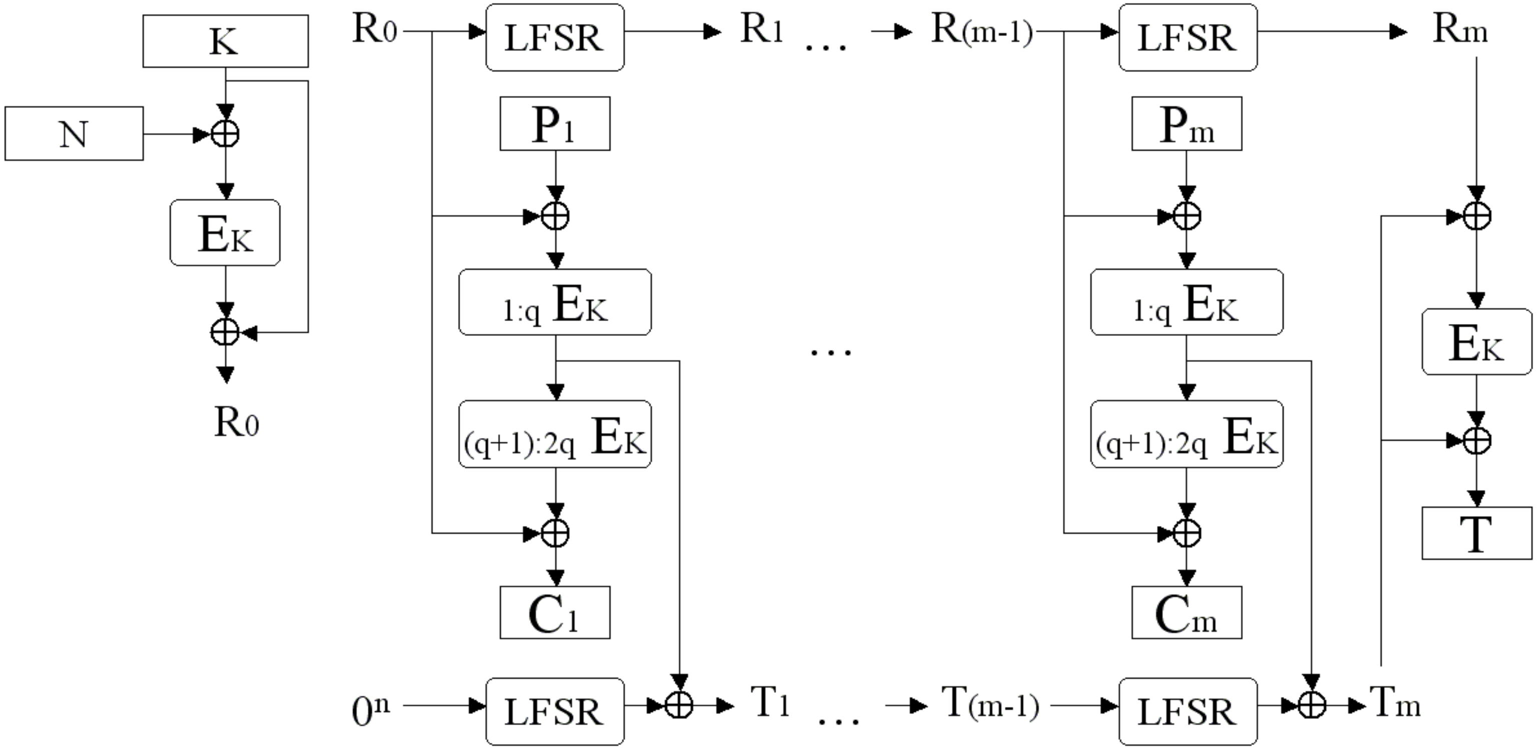
Offset CodeBook mode (OCB)

- Memory efficient, fast mode
- Provable secure, but patented



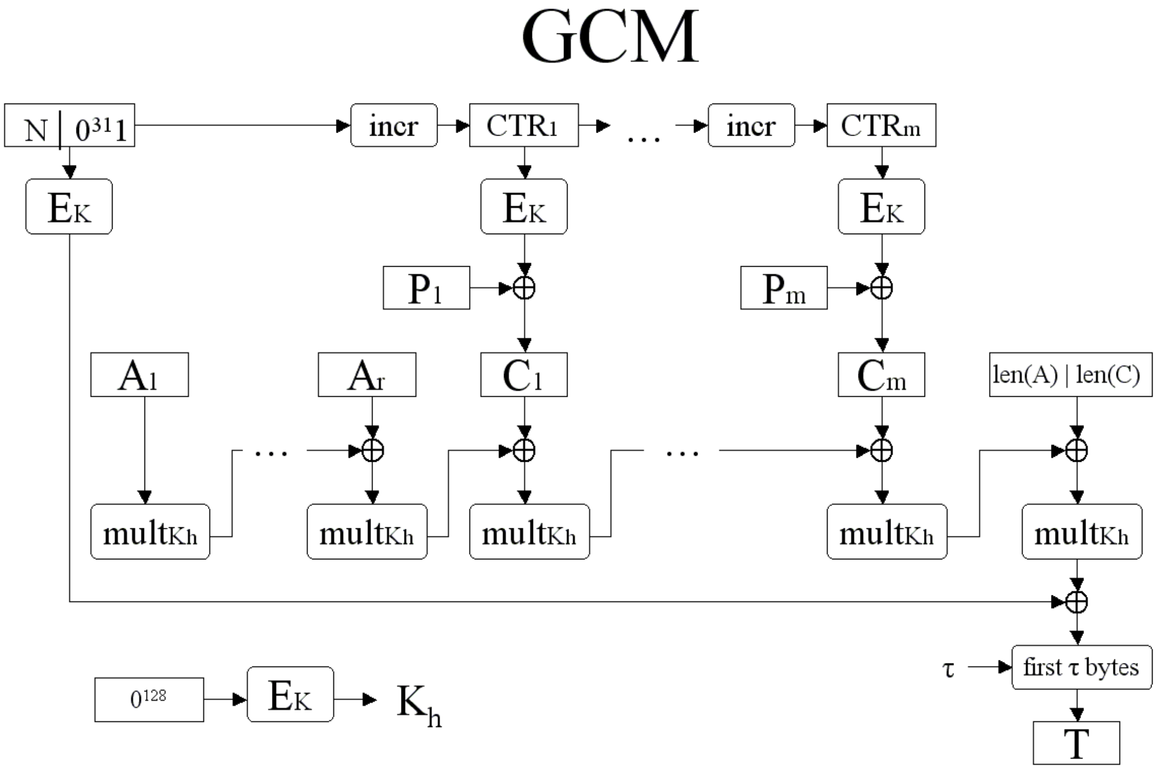
Cipher-State mode (CS)

- Memory efficient, fast mode, unpatented
- Not provable secure (inner state of cipher)



Galois/Counter Mode (GCM)

- Need pre-computed table (4kB-64kB)
- fast mode, provable secure, unpatented, NIST standard
- http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf



Implementation: AES-GCM from PolarSSL

- gcm.h, gcm.c

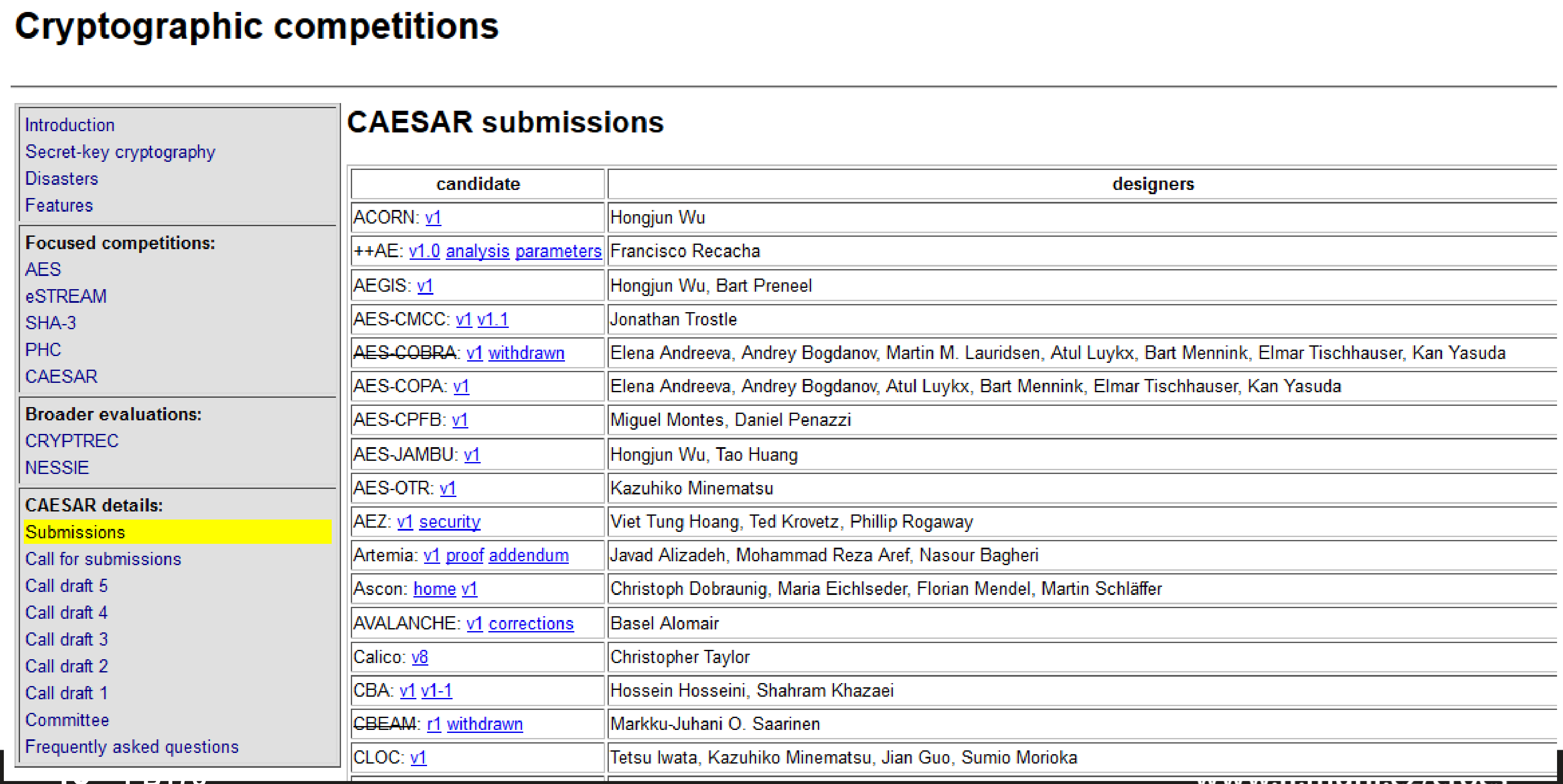
```
int gcm_init( gcm_context *ctx,
        const unsigned char *key,
        unsigned int keysize );

int gcm_crypt_and_tag( gcm_context *ctx,
        int mode, // GCM_ENCRYPT (alternatively GCM_DECRYPT)
        size_t length,
        const unsigned char *iv,
        size_t iv_len,
        const unsigned char *add, // authenticated, but not encrypted
        size_t add_len,
        const unsigned char *input, // authenticated and encrypted
        unsigned char *output,      // encrypted data
        size_t tag_len,
        unsigned char *tag );
```

```
int gcm_auth_decrypt( gcm_context *ctx,
        size_t length,              // length of input data
        const unsigned char *iv,
        size_t iv_len,
        const unsigned char *add,   // authenticated, but not encrypted
        size_t add_len,
        const unsigned char *tag,   // authenticator (MAC value)
        size_t tag_len,
        const unsigned char *input, // encrypted data
        unsigned char *output );    // decrypted data
```

CAESAR competition

- http://competitions.cr.yp.to/caesar-submissions.html



Cryptographic competitions

Introduction
Secret-key cryptography
Disasters
Features

Focused competitions:
AES
eSTREAM
SHA-3
PHC
CAESAR

Broader evaluations:
CRYPTREC
NESSIE

CAESAR details:
Submissions
Call for submissions
Call draft 5
Call draft 4
Call draft 3
Call draft 2
Call draft 1
Committee
Frequently asked questions

CAESAR submissions

candidate	designers
ACORN: v1	Hongjun Wu
++AE: v1.0 analysis parameters	Francisco Recacha
AEGIS: v1	Hongjun Wu, Bart Preneel
AES-CMCC: v1 v1.1	Jonathan Trostle
~~AES-COBRA~~: v1 withdrawn	Elena Andreeva, Andrey Bogdanov, Martin M. Lauridsen, Atul Luykx, Bart Mennink, Elmar Tischhauser, Kan Yasuda
AES-COPA: v1	Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Elmar Tischhauser, Kan Yasuda
AES-CPFB: v1	Miguel Montes, Daniel Penazzi
AES-JAMBU: v1	Hongjun Wu, Tao Huang
AES-OTR: v1	Kazuhiko Minematsu
AEZ: v1 security	Viet Tung Hoang, Ted Krovetz, Phillip Rogaway
Artemia: v1 proof addendum	Javad Alizadeh, Mohammad Reza Aref, Nasour Bagheri
Ascon: home v1	Christoph Dobraunig, Maria Eichlseder, Florian Mendel, Martin Schläffer
AVALANCHE: v1 corrections	Basel Alomair
Calico: v8	Christopher Taylor
CBA: v1 v1-1	Hossein Hosseini, Shahram Khazaei
~~CBEAM~~: r1 withdrawn	Markku-Juhani O. Saarinen
CLOC: v1	Tetsu Iwata, Kazuhiko Minematsu, Jian Guo, Sumio Morioka

Conclusions

- Composition of ENC and MAC can fail
  - encrypt-then-mac provable secure
  - specially designed composed modes
- One of the most promising mode is patented (OCB)
  - fast alternative GCM, CS
  - Searching for new modes (CAESAR competition)
- Suitable mode depends on usage
  - parallelizability, memory
  - specific needs (online, incremental MAC)

PRACTICAL ASSIGNMENT

Homework from last week – showtime ☺

- Update your design documents based on feedback
  - And add into GitHub repo (docs folder)
- Create implementation of server process
  - No network communication yet, just methods + tests!
- Functions to be implemented (and tested!)
  - new user registration (in: user name / password, out: status)
    - New user stored in local “database” (ini file, sqllite…)
  - user authentication to server (in:user/pass, out: status)
    - Check supplied info against info from local database
    - Use PBKDF2 or better function to generate hash to check
  - obtain list of other online users (out: formated list – JSON?)
    - Users that were successfully authenticated now assumed to be online
- Don’t forget to document functions in JavaDoc-style

Practical assignment

- Update your implementation of server functions based on feedback
  - And add into GitHub repo
- Create implementation of basic client process
  - No network communication yet, just methods + tests!
- Functions to be implemented (and tested!)
  - Login user (client side) (in: user name / password, out: status)
    - => prepared structure to be send to server
  - Prepare protected message for another user (in: user identification, session context (keys, counters…), out: protected message, status)
    - Encryption, MAC, Use suitable Authenticated Encryption mode
    - Update session context
  - Unprotect message from another user (in: protected message, session context (keys, counters…), out: unprotected message, status)
    - Think how to handle errors
- Don’t forget to document functions in JavaDoc-style

Submissions, deadlines

- Upload application source codes as single zip file into IS Homework vault (Crypto - 5. homework (AE))
  - Zip file from current version of repo
- DEADLINE 4.4. 12:00
  - Up to 10 points assigned