

# PB173 - Tématický vývoj aplikací v C/C++ (podzim 2011)

*Skupina: Aplikovaná kryptografie a bezpečné programování*

[https://minotaur.fi.muni.cz:8443/pb173\\_crypto](https://minotaur.fi.muni.cz:8443/pb173_crypto)

Petr Švenda, [svenda@fi.muni.cz](mailto:svenda@fi.muni.cz)  
Konzultace: B420, Pondělí 13-13:50

# Co **je** cílem předmětu

- Získat zkušenosti s implementací většího programu
- Používat vývojové nástroje
- Naučit se dobré programátorské postupy
  - programování obecně
  - ale speciálně v oblasti bezpečnostních aplikací
- Získat praktické postřehy z implementací kryptografických aplikací
  - co nakonec ve firmě vyžadují

# Co **není** cílem předmětu

- Detailní ovládnutí konkrétní technologie
  - zabrousíme do různých oblastí
- Pokročilé zvládnutí celého vývojového procesu
  - to jednoduše nestihneme
- Vysvětlovat základy kryptografie nebo srovnávat všechny možné varianty řešení problému
  - hlavně se budeme snažit prakticky programovat

# Organizační

- Formality výuky
  - každotýdenní dvojhodinovka
  - evidovaná účast, 2 neúčasti bez omluvení OK
- Způsob výuky
  - max. cca 30 min./týdně úvod do problematiky
  - zbytek programování přímo na hodině
  - z mé strany průběžná konzultace nad vznikajícími problémy
  - default Windows (ale můžete pracovat i na jiné platformě)
- Samostatná práce
  - v týmech, průběžná tvorba většího projektu
  - dodělávání práce z hodiny
  - pravidelné bodované předvádění stavu projektu (každé cvičení)

# Organizační (2)

- Používané nástroje
  - IDE, verzovací nástroje, Doxygen, debugger, analýza a kontrola kódu
  - konkrétní není striktně dané – použijte svoje oblíbené
  - default Visual Studio
- Hodnocení
  - účast
  - průběžná práce (10 bodů týdně)
  - prezentace celého projektu (30 bodů)
  - možné bonusy
  - max. 150 bodů, zisk alespoň 100 bodů na kolokvium

# Rozdělení do týmů

- 3 osoby
- Společná práce, ale každý prezentuje svůj přínos
  - prezentace na každém dalším cvičení
  - resp. za 14 dní při absenci
- Rozdělení teď!

# Celkový přehled

- Základní podklady na wiki
  - [https://minotaur.fi.muni.cz:8443/pb173\\_crypto](https://minotaur.fi.muni.cz:8443/pb173_crypto)
- Pro informaci přehled z minulého roku
  - [https://minotaur.fi.muni.cz:8443/~xsvenda/docuwiki/doku.php?id=public:pb173:pb173\\_2010\\_crypto](https://minotaur.fi.muni.cz:8443/~xsvenda/docuwiki/doku.php?id=public:pb173:pb173_2010_crypto)
- Může se ale částečně měnit
  - uvidíme dle reálné obtížnosti, rychlosti postupu a zájmu
- Můžete otevřít vlastní řešený problém

---

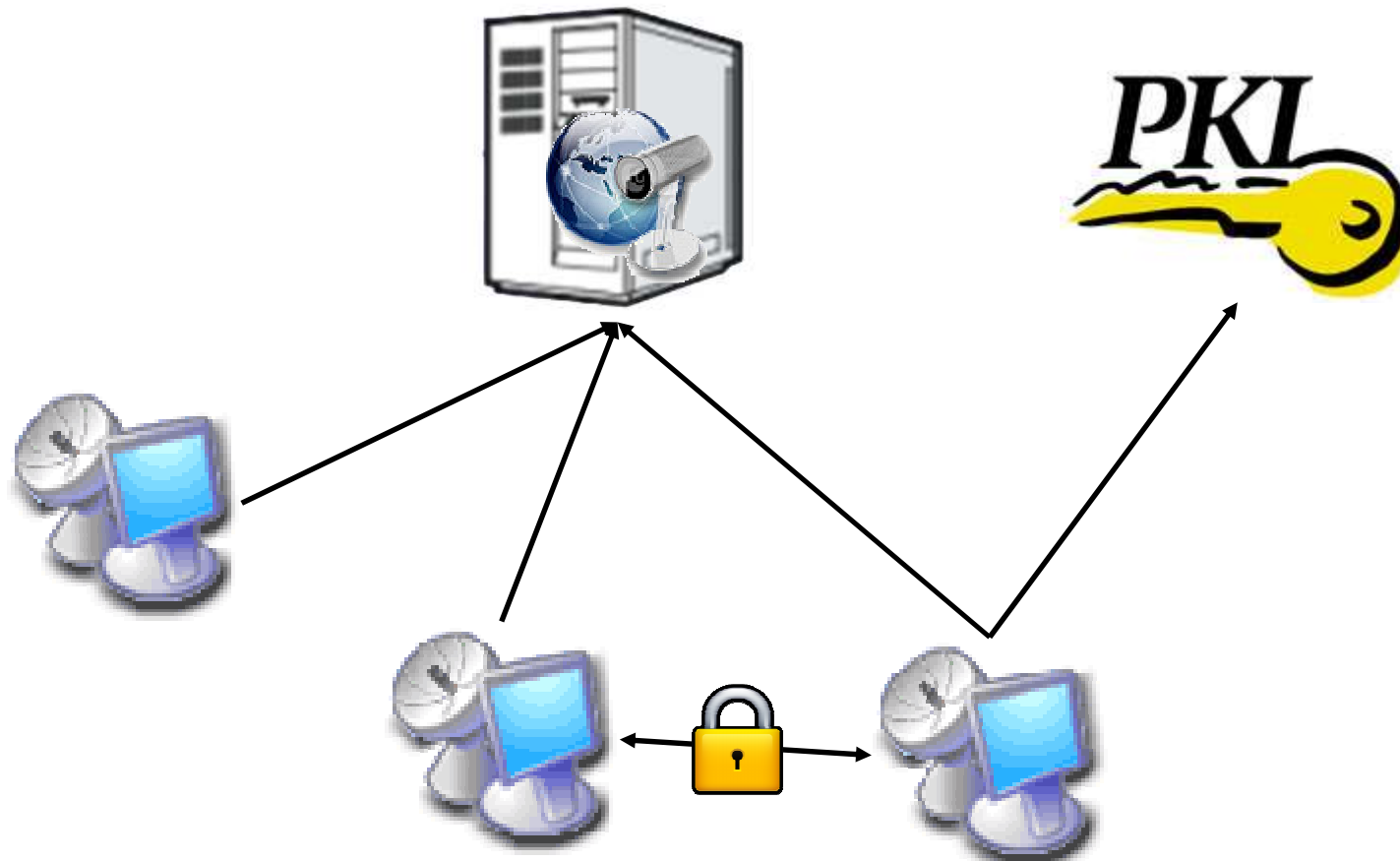
# How good YOU are in English?

*Apology all my mistakes,  
please.*



# "Theme" project

- Secure videoconferencing architecture



# "Theme" project

- Certification authority
  - validates and issue user certificates
- Videoconferencing server
  - register and faciliate connection between users
- Client
  - provides operations related to end user usage
- Main focus on solving parts of the architecture

# "Theme" project – some details

- Users obtains certificate of identity from Certification authority
- Users register with Videoconferencing server
- Videoconferencing server provides list of connected users, help to establish video connection and charge fee based on call length
- Client maintains user identity, related keys and provides high speed encryption of audio/video stream

# Short questionnaire

1. Do you know difference between symmetric and asymmetric cryptography? 3
2. Do you known difference between block and stream cipher? 7
3. Do you know DES and AES algorithm? 7
4. Do you know ECB and CBC encryption mode? 9
5. Do you know principle of hash functions? 6
6. Do you know MD5 and SHA-1 algorithm? 5
7. Do you known concept of digital signature? 2

---

# Cryptographic libraries

---

# Do not implement your own algorithms

- Time consuming (someone probably already did that before)
- Functional problems
- Low performance
- Security problems due to bugs
- Security problems due to missing defense against implementation attacks

# Use well-known implementations

- Use well-known libraries
  - OpenSSL, PolarSSL, GnuPG, BouncyCastle (Java)
- Or implementation of algorithms from well-established authors
  - Brian Gladman, Eric A. Young ...

# Complexity matters

- Complexity of library implementation should match your needs
  - usually, you need only one or two algorithms
- Multiprocessor or CPU-independent implementation can be overkill
  - and just increase risk of error
- Do you really need library with object-oriented design?



# Complexity matters (2)

- Large libraries are not always the most suitable ones
- OpenSSL is complex and interconnected
  - e.g., AES is extractable much easier from PolarSSL than from OpenSSL

# Code authenticity

- Source code signature
  - Do you really have original source codes?
  - MD5/SHA1 hash (where to get “correct” hash value?)
  - GPG/PGP
- Generate your own GPG/PGP signature keys
  - use them for inter-team communication
  - sign your code releases

# Resilience against bugs

- Do not design algorithms/protocols by yourself
- Try to find existing standards
  - NIST, RSA PKCS, RFC, ISO/ANSI
- Try not to deviate from standards
  - compatibility and compliance
  - no need for (time consuming) specification of detailed your scheme
  - small change can have big security impacts

# Libraries used often - OpenSSL

- Pros:

- Very rich library
  - lots of algorithms, protocols, paddings
  - not “just” SSL
- well tested functionally&security over time!
- significant amount of existing examples on web

- Cons:

- API is complex and sometimes harder to understand
- relatively low-level functions (can be pros!)
- code is significantly interconnected
  - not suitable for extraction of single algorithm
- poor official documentation

# Libraries used often - PolarSSL

- Pros:
  - API is simple and clear
  - easy to extract single algorithm
- Cons:
  - fewer supported algorithms and standards
  - dual licensing, but not BSD-like license

# How to use library

- Extract code and compile alone
  - some work with extraction
  - small, clean and self-containing result
- Compile against whole library
  - usually easy to do
  - but dependence on possibly unused code
- Link statically against dynamic library
  - dll must be always present to run program

## How to use library (2)

- Link dynamically against dynamic library
  - try to open dll file and obtain function handle
- Link against service provider functions
  - Cryptography Service Providers in particular
  - API for listing of available service providers (CryptEnumProviders)
  - standardized functions provided by providers  
[http://msdn.microsoft.com/en-us/library/aa380252%28v=VS.85%29.aspx#service\\_provider\\_functions](http://msdn.microsoft.com/en-us/library/aa380252%28v=VS.85%29.aspx#service_provider_functions)

# Security implications of dynamic libraries

- Library can be forged and exchanged
- Library-in-the-middle attack easy
  - data flow logging
  - input/output manipulation
- Library outputs can be less checked than user inputs
  - feeling that library is my “internal” stuff and should play by „my“ rules
- Library function call can be behind logical access controls



---

# Practical assignment

# Practical assignment

- Download OpenSSL and PolarSSL library
  - and check signature (gpg --verify)
- Write small project
  - read, encrypt and hash supplied file, write into out file
  - read, verify hash and decrypt file
  - use AES-128 in CBC mode and HMAC with SHA2-512
  - use PKCS#7 padding method for encryption (RFC 3852)

# Practical assignment (2)

- Requirements
  - user specifies input and output file
  - use functions from OpenSSL library as DLL calls
  - extract AES and SHA-2 code from PolarSSL library
  - user specifies if OpenSSL DLL or build-in PolarSSL methods will be used

# Practical assignment (3)

- Write following simple unit tests (CxxTest)
  - <http://morison.biz/technotes/articles/23>
  - file not exists or cannot be read/written into
  - encrypted blob was corrupted
  - wrong decryption key was used
  - test vectors for encryption and hashing
- Code will be used later in architecture
  - will be used again and extended, so write it well

---

# Questions?

