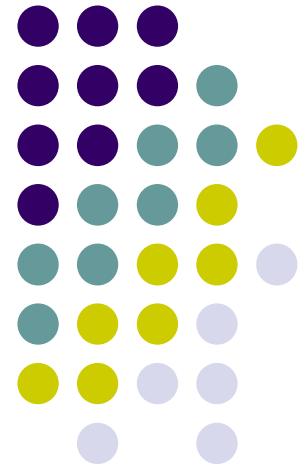


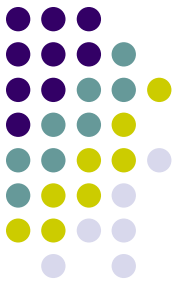
Crypto libraries introduction

Milan Brož
xbroz@fi.muni.cz



Crypto libraries...

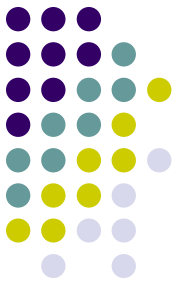
plan for next two PV181 labs



- Linux environment
 - Fedora in VirtualBox (image in IS) or
 - aisa.fi.muni.cz (OpenSSL only)
- Examples in C language
- 1st lab: Intro, crypto primitives examples
- 2nd lab: symmetric ciphers, asymmetric crypto, basic certificates
- Home assignments (max. 10 point each)

Lab environment

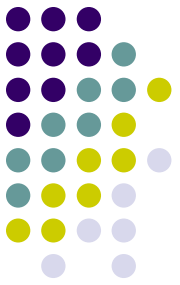
VirtualBox image



- Unpack zip archive from IS
- Open VirtualBox (click **blue** icon – config file)
D:\PV181 is shared between host and VM
- Login and password is “**pv181**”
(same for sudo and root password)
- Clone examples from FI gitlab:
`git clone https://gitlab.fi.muni.cz/xbroz/pv181.git`
(or use script in home dir in VM)
`make clean; make; ./example`
- OpenSSL examples works also on aisa

Cryptographic libraries

... An Introduction

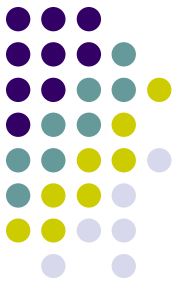


- opensource / proprietary
- static + embedded / dynamically linked
- low / high level abstractions
- multiplatform
- stable API/ABI
- specific implementations
 - side-channel resistance
 - HW acceleration support
 - ...

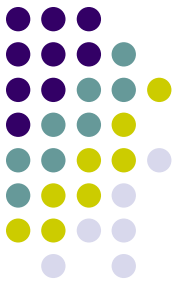
Open-source crypto libraries examples

- **Nettle**
- **gcrypt**
- **OpenSSL** (and derivatives)
- **NSS**
 - Network Security Services (Mozilla)
- ...
- **NaCl ("salt") -> libsodium**

low < ABSTRACTION > high

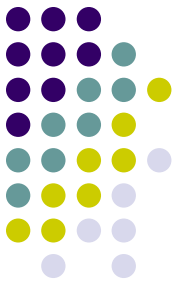


Crypto libraries



- Random Number Generator (RNG) access
- Hash, keyed-hash (HMAC, msg authentication)
- Symmetric ciphers and modes
- Asymmetric ciphers
- Certificate support, ASN.1, ...
- Key exchange, key derivation
- Helpers
 - secure memory
 - safe comparison
 - network / sockets
 - ...

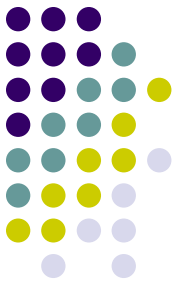
Today's exercise



- **Low-level crypto primitives**
 - RNG
 - Hash, HMAC
 - PBKDF
- Examples comparison in **OpenSSL, gcrypt, libsodium**
- Defensive approach:
It will fail, be prepared for it. 😊

Why implementation matters

Key from Linux RNG example



- Bad coding: How many bugs do you see?

```
#include <string.h>
#include <unistd.h>
#include <fcntl.h>

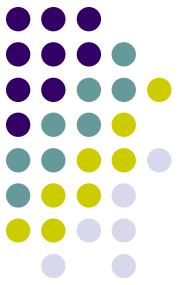
int main(int argc, char *argv[])
{
    int fd;
    char key[32];

    fd = open("/dev/random", O_RDONLY);
    read(fd, key, 32);
    close(fd);

    // Do something with the key[]

    memset(key, 0, sizeof(key));
    return 0;
}
```


Example 1: RNG in libraries



libgcrypt

see `1_rng_gcrypt` example

```
(void) gcry_randomize(buf, sizeof(buf), GCRY_STRONG_RANDOM);
```

OpenSSL

see `1_rng_openssl` example

```
(int) RAND_bytes(buf, sizeof(buf))
```

libsodium

see `1_rng_sodium` example

```
(void) randombytes(buf, sizeof(buf));
```

Simple? Not in real-world. RNG or pseudo RNG, optional parameters, initialization or another call for configuration, can/cannot fail, can/cannot block if not enough entropy, is it own implementation or wrapper to system RNG, can it be used in FIPS mode ...

Example 2: Hash functions



libgcrypt

See `2_hash_hmac_gcrypt` example

```
gcry_md_open(context, hash_id, flags)
gcry_md_write(context, data, data_len)
gcry_md_read(context, hash_id)
gcry_md_close(context)
```

OpenSSL (new 1.1.0 syntax)

EVP (envelope) interface, see `2_hash_hmac_openssl` example

```
EVP_MD_CTX_new();
EVP_DigestInit(context, hash_id)
EVP_DigestUpdate(context, data, data_len)
EVP_DigestFinal(context, out, &out_len)
EVP_MD_CTX_free(context);
```

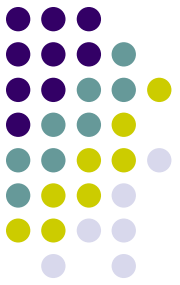
libsodium

See `2_hash_hmac_sodium` example

```
crypto_hash_sha256_init(context)
crypto_hash_sha256_update(context, data, data_len)
crypto_hash_sha256_final(context, out))
```

Example 2: HMAC

Keyed Hash Message Authentication Code



libgcrypt

See `2_hash_hmac_gcrypt` example

```
gcry_md_open(context, hash_id, GCRY_MD_FLAG_HMAC)
gcry_md_setkey(context, key, key_len)
gcry_md_write(context, data, data_len)
gcry_md_read(context, hash_id)
gcry_md_close(context)
```

OpenSSL (new 1.1.0 syntax)

EVP interface or direct calls, see `2_hash_hmac_openssl` example

```
HMAC_CTX_new();
HMAC_Init(context, key, key_len, hash_id)
HMAC_Update(context, data, data_len)
HMAC_Final(context, out, &out_len)
HMAC_CTX_free(context);
```

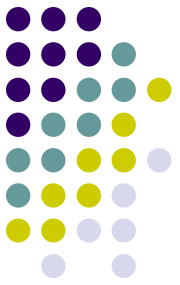
libsodium

NaCl compatible interface, see `2_hash_hmac_sodium` example

```
crypto_auth(out, data, data_len, key))
crypto_auth_verify(expected_out, data, data_len, key))
```

Example 3: PBKDF

Password-Based Key Derivation Functions



libgcrypt

See `3_pbkdf_gcrypt` example

```
gcry_kdf_derive(password, password_len,  
                GCRY_KDF_PBKDF2, GCRY_MD_SHA256,  
                salt, salt_len, iterations, key_len, key)
```

OpenSSL

See `3_pbkdf_openssl` example

```
PKCS5_PBKDF2_HMAC(password, password_len, salt, salt_len,  
                  iterations, EVP_sha256, key_len, key)
```

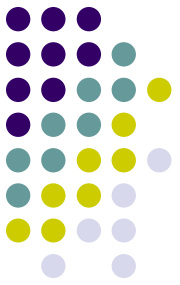
libsodium

(no example intentionally, default Argon2i is too recent :-)

```
crypto_pwhash(key, key_len, password, password_len,  
             salt, opslimit, memlimit, algorithm)
```

*Note: old API functions based on PBKDF2 (supports only time cost – iterations)
For recent algorithms (scrypt, Argon2i) API calls are often abused ...*

Assignment



- Goal is to
 - Work with standard (RFC) document
 - Use test vectors (self tests)
 - Use OpenSSL in Linux environment
- See Assignment.txt in IS
- You can start with the provided example
- Comment your code
 - but do not overuse comments
- NO plagiarism (even from previous years)
 - => 0 points for both sides (sender & receiver)
- Code quality matters!