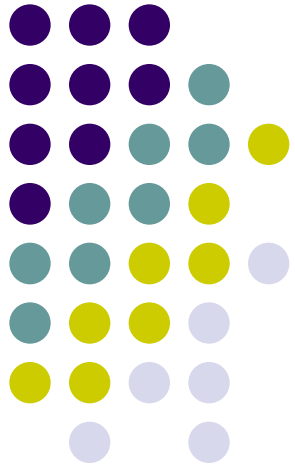
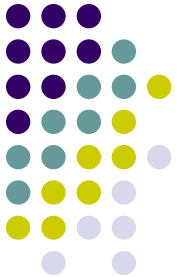
Crypto libraries introduction

Milan Brož
xbroz@fi.muni.cz

PV181, FI MUNI, Brno

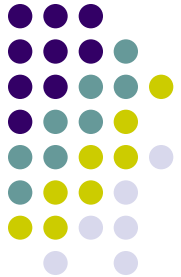


Cryptographic libraries plan for next three PV181 labs



- Linux environment
  - Fedora in VirtualBox (image in IS) or
  - aisa.fi.muni.cz (OpenSSL only) or
  - Your own distro
- Examples in C language
- Home assignments (10 points each)

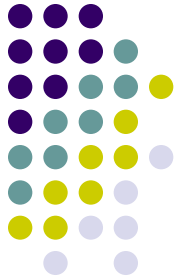
Lab environment
VirtualBox image



- Unpack zip archive from IS
- Open VirtualBox (click **blue** icon – config file)
- Login and password is **pv181**
  (same for sudo and root password)

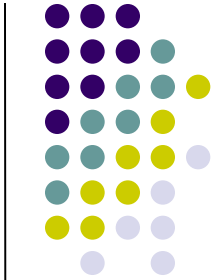
- Examples on gitlab
  git clone https://gitlab.fi.muni.cz/xbroz/pv181.git
  make clean; make; ./example

Cryptographic libraries Introduction



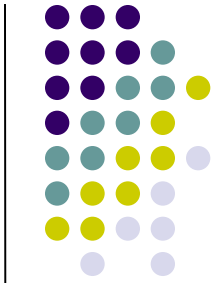
- Open-source / Proprietary
- Static + embedded / dynamically linked
- Low or high level abstractions
- Multiplatform
- Stable API and ABI
- Security or platform specific features
  - Safe memory use, side-channel resistance, …
  - HW acceleration support, “secure” HW support

Example libs (C and Linux) abstraction from low to high



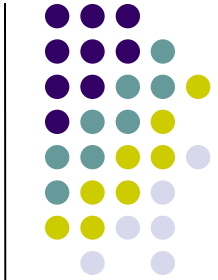
- **Nettle**
- **libgcrypt**
- **OpenSSL**
  - LibreSSL (clone), BoringSSL (Google)
- **NSS**
  - Network Security Services (Mozilla)
- **NaCl** ("salt")
  - more common as **libsodium**

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

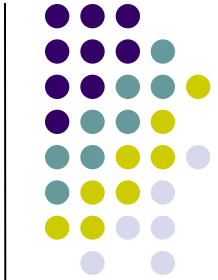


- **Low-level crypto primitives**
  - RNG
  - Hash, HMAC
  - PBKDF

- Examples comparison in **OpenSSL**, **gcrypt**, **libsodium**

- Learn to write code in a defensive approach **It will fail, be prepared for it.** ☺

Why implementation matters



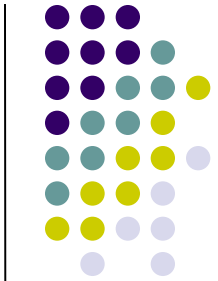
- It works, but …
- How many possible bugs do you see?

```c
/* Read a key from Linux RNG */
#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, 32);
    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))
```

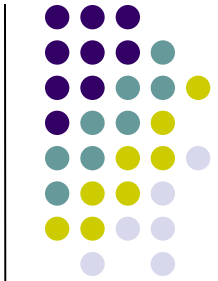
libsodium

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);
```

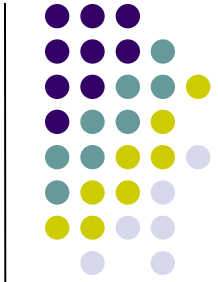
libsodium

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);
```

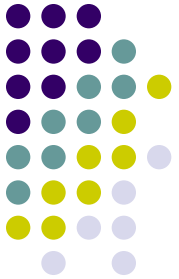
libsodium

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)
```

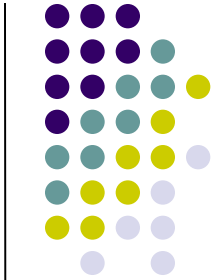
libsodium

(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-test)
  - Use OpenSSL in a Linux environment
- See Assignment.txt in IS
- You can use 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!