New APIs in OpenSSL 3.0

Alexei Khlebnikov
Oslo C++ Users Group Meetup, 2022

> whoami

- Alexei Khlebnikov
- More than 20 years in IT
- Much experience with OpenSSL
- Wrote a book about OpenSSL
- Now working as a Senior Consutlant and the Leader of the Architects Group in bspoke AS

bspoke

Providers

- Collections of algorithm implementations
- Providers supplied with OpenSSL:
  - default, base, legacy, fips, null
- Providers are loaded:
  - Explicitly, using OSSL_PROVIDER_load()
  - Implicitly, the “default” provider

bspoke

OpenSSL Library Contexts

- Scopes for OpenSSL library configuration
- Scopes for loaded providers
- Contexts are loaded:
  - Explicitly using OSSL_LIB_CTX_load_config()
  - Implicitly, the default context

bspoke

Fetching Algorithm from Provider

- Explicit fetch using functions EVP_CIPHER_fetch(), EVP_MD_fetch(), etc
  - Select provider using the “properties” string, for example: “provider=default”
- Implicit fetch from the “default” provider using functions EVP_aes_128_cbc(), EVP_sha256(), etc

bspoke

New APIs

- EVP_MAC
- EVP_KDF
- EVP_KEM
- EVP_RAND
- HTTP(S) client
- CMP (Certificate Management Protocol)

bspoke

New Algorithms

- AES-GCM-SIV
- GMAC
- KMAC
- RSASVE

bspoke

Deprecated APIs

- AES_encrypt(), DES_encrypt3()
  - Use EVP_CIPHER API
- SHA256_Init(), MD5_Init()
  - Use EVP_MD API
- HMAC_Init(), CMAC_Init()
  - Use EVP_MAC API

Deprecated APIs

- PKCS5_PBKDF2_HMAC_SHA1(), PKCS5_PBKDF2_HMAC(), EVP_PBE_scrypt()
  - Use EVP_KDF API
- RSA_new(), DSA_sign(), ECDSA_verify()
  - Use EVP_PKEY API
- Engines and METHOD APIs
  - Use Providers

Code Example

```
// Define password, salt, and desired key length.
const char* password = "SuperPa$$w0rd";
const char* salt = "NaCl";
const size_t KEY_LENGTH = 256 / 8;
unsigned char key[KEY_LENGTH];
```

Code Example

```
// Fetch "SCRYPT" KDF algorithm.
OSSL_LIB_CTX* default_library_context = NULL;
const char* algorithm_name = OSSL_KDF_NAME_SCRYPT;
const char* default_algorithm_properties = NULL;
EVP_KDF* kdf = EVP_KDF_fetch(
    default_library_context,
    algorithm_name,
    default_algorithm_properties);
```

bspoke

Code Example

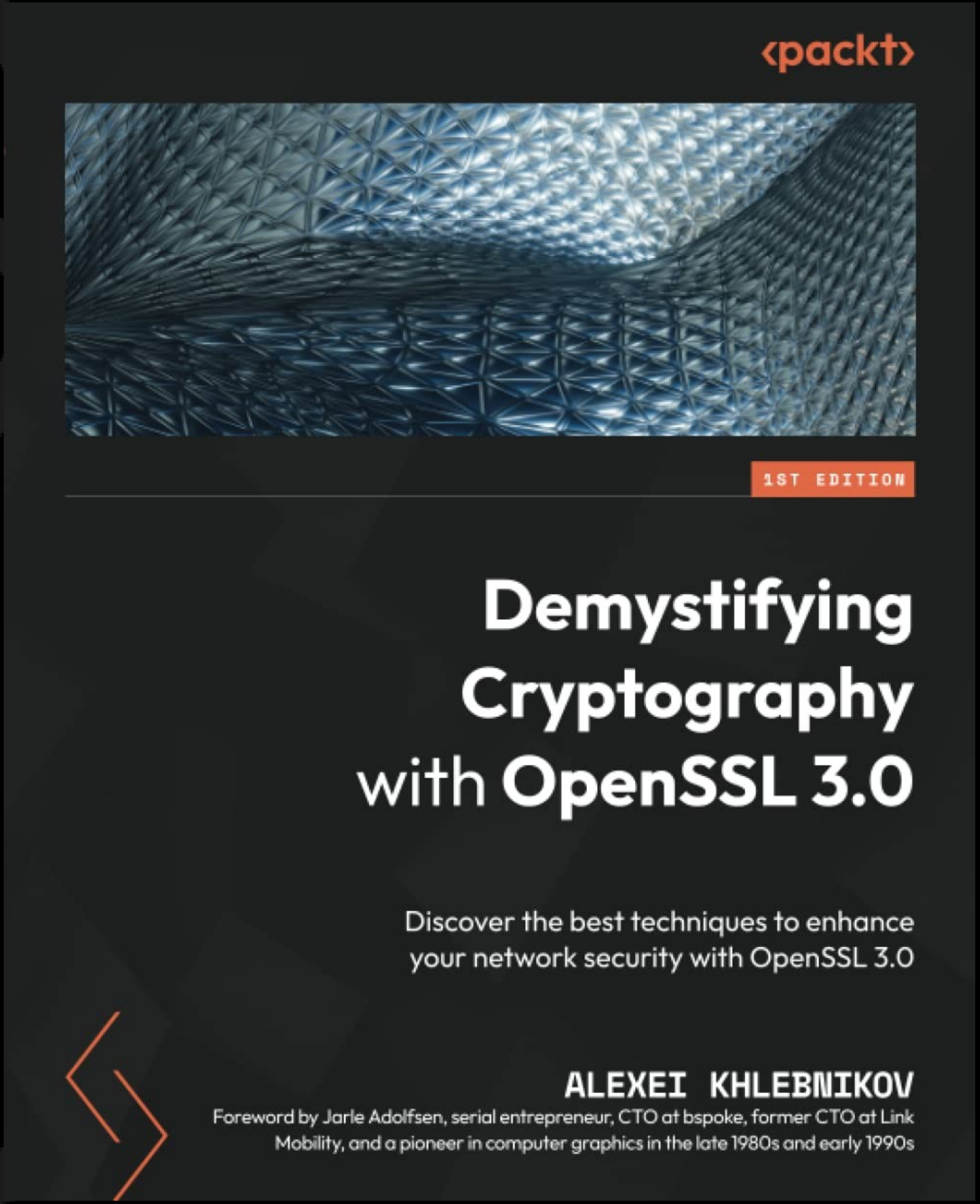
```c
// Define scrypt parameters. Use OWASP recommended settings.
uint64_t scrypt_n = 65536;
uint32_t scrypt_r = 8;
uint32_t scrypt_p = 1;

OSSL_PARAM params[] = {
    OSSL_PARAM_construct_octet_string(
        OSSL_KDF_PARAM_PASSWORD, (char*)password, strlen(password)),
    OSSL_PARAM_construct_octet_string(
        OSSL_KDF_PARAM_SALT, (char*)salt, strlen(salt)),
    OSSL_PARAM_construct_uint64(OSSL_KDF_PARAM_SCRYPT_N, &scrypt_n),
    OSSL_PARAM_construct_uint32(OSSL_KDF_PARAM_SCRYPT_R, &scrypt_r),
    OSSL_PARAM_construct_uint32(OSSL_KDF_PARAM_SCRYPT_P, &scrypt_p),
    OSSL_PARAM_construct_end()
};
```

bspoke

Code Example

```
// Generate encryption key.
EVP_KDF_CTX* ctx = EVP_KDF_CTX_new(kdf);
int ok = EVP_KDF_derive(ctx, key, KEY_LENGTH, params);
```



‹packt›

1ST EDITION

Demystifying Cryptography with OpenSSL 3.0

Discover the best techniques to enhance your network security with OpenSSL 3.0

ALEXEI KHLEBNIKOV

Foreword by Jarle Adolfsen, serial entrepreneur, CTO at bspoke, former CTO at Link Mobility, and a pioneer in computer graphics in the late 1980s and early 1990s

https://amzn.to/3FuJ5kt

bspoke