Funded by the European Union under Grant Agreement No. 101087529. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Research Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

Co-funded by the European Union

Applied PQC in Software Security

CHESS

Cyber-security Excellence Hub in Estonia and South Moravia

Post Quantum Transition Workshop (PQT)

4.12.2024, Prague

Petr Muzikant

Information Security Research Institute @ Cybernetica AS, Estonia

[This presentation includes clickable links]

CYBERNETICA

For Public

Introduction

- Use case:
  - **Threat actor:** someone with sufficient quantum computer
  - **Operational cost:** enormous
  - **Main target:** probably important/valuable systems
  - **Cryptography implementation:** more advanced, probably hardware based
- Is PQC required everywhere, especially in every software application?

Introduction

- **SW cryptography is everywhere**
  - TLS, VPN, cloud services, password managers, certificates, digital signatures, 5G
- **Do/Should we bother with non-critical, low-level target systems?**
  - cheap
  - cryptography evolution
  - gateway to critical systems
  - makes sense

Our Experience

- **What:** PQ proof-of-concept implementations of existing applications (SW)
- **Why:** PQC research → Research&Development → exciting
- **Since:** almost 2 years ago
- **Applications:**
    - Web-eID (authentication)
    - CDOC2 (encryption)
    - IVXV (e-voting)
    - eID (certificates, OCSP, TSA)
    - supporting projects (library wrappers, lattice-helper, custom protocols)

Our Intentions

- More then just *"PQ-ize e-government applications"*
    - **explore** and support current PQC open-source
    - **focus on engineering aspects** of PQ implementations
    - **gather** experience, problems, and remarks
    - **learn**
    - **disseminate**

Applied PQC

Cryptography Libraries, Applications

PQClean and libOQS

- Well known, developed, and maintained
- PQClean (C)
  - cleaned aggregation of NIST-submitted algorithms (and FIPS standards)
  - source of source-code (i.e. not a library)
- libOQS (C)
  - + wrappers for C++, Python, Java, Go, .NET, Rust, and PHP
  - + applications built with libOQS (OpenSSL, OpenSSH, OpenVPN forks)

Bouncy Castle (Java)

- Quite hidden, not well documented, very early
- Java
  - needs to be full-featured → a lot of shortcuts
  - workflow different from others
- The implementation seems to go smoothly after some time
- Useful "Java Keytool" benefits too

CIRCL (Go)

- "Cloudflare Interoperable Reusable Cryptographic Library"
- Pure implementations in Go language by Cloudflare
- libOQS
  - similar workflow and API
  - objects are compatible with libOQS
- Works well with "crypto" standard library

Other Libraries

- rustpq/pqcrypto (Rust)
- smuellerDD/leancrypto (C with minimal dependencies)
- air-crypto/botan-pq (C++, PQ version of popular Botan library)
- terra-quantum-public/tq42-pqc-oss (C++, looks promising)
- many others
  - FIPS 203-205 contain detailed implementation instructions

OpenQuantumSafe organization

- Many applications built with libOQS
- **OpenSSL with PQ provider (extension)**
  - → TLS
  - → SSH
  - → certificates, OCSP, timestamping servers
  - → C API
  - → basically everything
- Many external users of libOQS
  - Microsoft, Thales, Cisco, IBM, Entrust, etc.

Transport Layer Security (TLS)

- **Biggest target of PQC implementation**
  - attempts to introduce KEM-TLS
- Cloudflare
  - documents PQ TLS
  - monitors clients connecting via PQ TLS (16 %)
- OpenQuantumSafe
  - provides PQ-TLS implementations for many server and client solutions
- Google
  - first hybrid TLS in Google Chrome

Virtual Private Networks

- Experiments together with Brno University of Technology
    - includes also pre-quantum (and quantum) key exchange
- Microsoft PQ VPN (but archived)
- Rosenpass
    - PQ key-exchange extension for WireGuard VPN
    - seems well maintained
- ExpressVPN, QAL VPN (paid solutions)

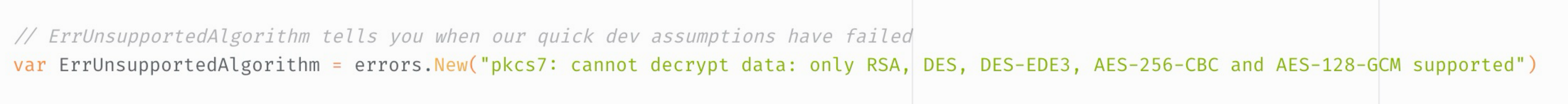
Engineering Obstacles

- Inconsistent PQ object encodings
- Inconsistent key handling
- Inconsistent naming
- Hash-then-Sign dilemma
- Standard vs external libraries
- Deep dependency chains
- Interoperability issues
- PQC in all layers of SW - it is NOT just an algorithm switch

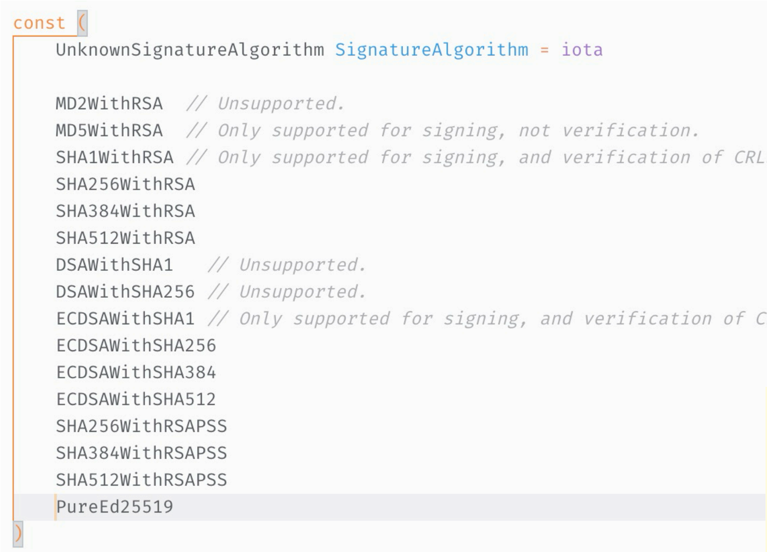
Crypto Agility

- **Hardcoded cryptography implementation**
  - *"ERROR: only X, Y, Z are supported, nothing else!"*
- **Introducing new algorithm** often requires adapting whole application codebase (demanding)
- **Crypto agile** application
  - zero knowledge about cryptography algorithms
- Thankfully, we still have time to figure this out

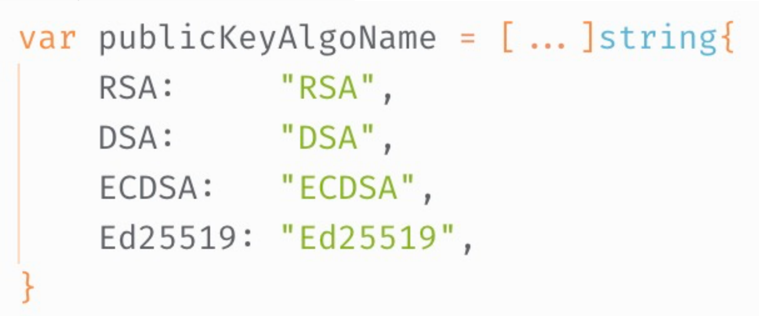
Crypto Agility



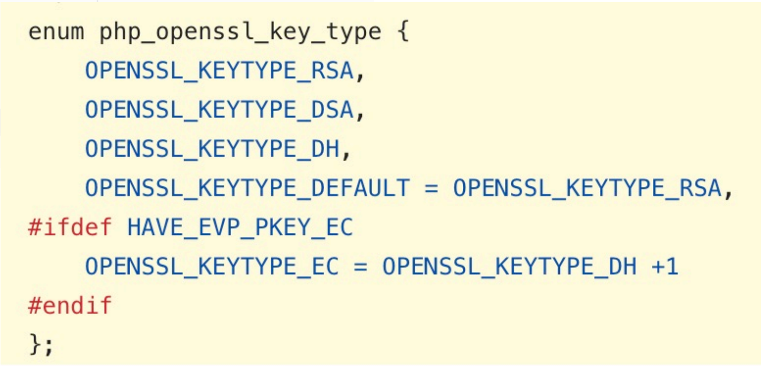
```
// ErrUnsupportedAlgorithm tells you when our quick dev assumptions have failed
var ErrUnsupportedAlgorithm = errors.New("pkcs7: cannot decrypt data: only RSA, DES, DES-EDE3, AES-256-CBC and AES-128-GCM supported")
```



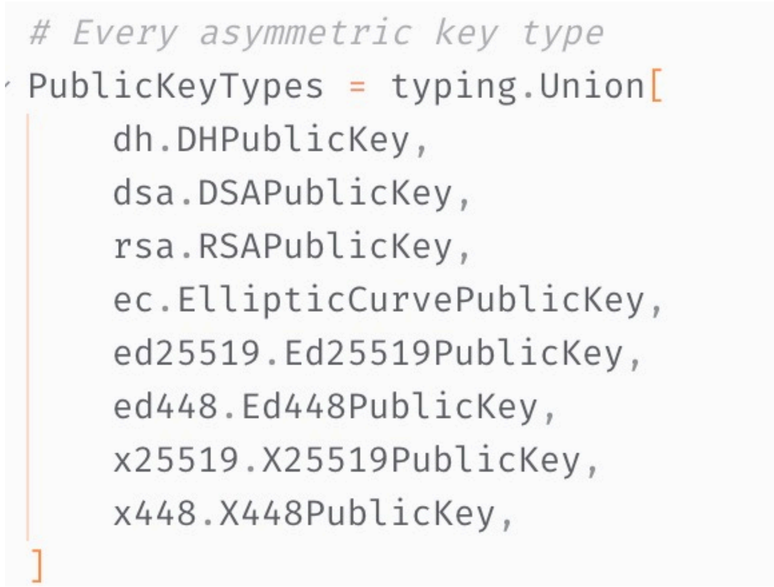
```
const (
	UnknownSignatureAlgorithm SignatureAlgorithm = iota

	MD2WithRSA  // Unsupported.
	MD5WithRSA  // Only supported for signing, not verification.
	SHA1WithRSA // Only supported for signing, and verification of CRLs,
	SHA256WithRSA
	SHA384WithRSA
	SHA512WithRSA
	DSAWithSHA1   // Unsupported.
	DSAWithSHA256 // Unsupported.
	ECDSAWithSHA1 // Only supported for signing, and verification of CRL
	ECDSAWithSHA256
	ECDSAWithSHA384
	ECDSAWithSHA512
	SHA256WithRSAPSS
	SHA384WithRSAPSS
	SHA512WithRSAPSS
	PureEd25519
)
```



```
var publicKeyAlgoName = [...]string{
	RSA:     "RSA",
	DSA:     "DSA",
	ECDSA:   "ECDSA",
	Ed25519: "Ed25519",
}
```



```
enum php_openssl_key_type {
	OPENSSL_KEYTYPE_RSA,
	OPENSSL_KEYTYPE_DSA,
	OPENSSL_KEYTYPE_DH,
	OPENSSL_KEYTYPE_DEFAULT = OPENSSL_KEYTYPE_RSA,
#ifdef HAVE_EVP_PKEY_EC
	OPENSSL_KEYTYPE_EC = OPENSSL_KEYTYPE_DH +1
#endif
};
```



```
# Every asymmetric key type
PublicKeyTypes = typing.Union[
    dh.DHPublicKey,
    dsa.DSAPublicKey,
    rsa.RSAPublicKey,
    ec.EllipticCurvePublicKey,
    ed25519.Ed25519PublicKey,
    ed448.Ed448PublicKey,
    x25519.X25519PublicKey,
    x448.X448PublicKey,
]
```

~~PQ~~ → Cryptosystems "Transition"

- Managing Cryptography: Cryptographic Discovery & PQC Migration Panel (NIST 5th PQC Standardization Conference, session 7):
  - *"Don't migrate to PQC, **migrate to better management** of cryptography in your systems"*
  - *"**Cryptography management system was needed years ago**..."*
- Chance to make cryptography in SW better
  - **agile**, **modular**, **adaptive**
  - new **best practices**
  - **IT blindspots**
    - *"What do you mean we are still using SHA1?!"*

Key Takeaways

1. **PQC in software is very much a real thing**, regardless of CRQC
    - not perfect, but developing quite fast
2. **PQ solutions exist**, no need to build from scratch
3. **Very strong community** (forums, orgs, IETF, alliances)
4. Chance to **overhaul cryptography management** in our applications

Resources

- Lot of resources to help make a decision (e.g. PQC Migration Handbook)
- **Where to look as an engineer?**
  - IETF RFC Drafts
    - most are not complete, already expired, contradict each other
    - basics: PQC for engineers RFC
  - pqc-forum Google Group
  - state-of-protocols-and-pqc repository
  - NIST Special Publication series 1800-38 ??
  - NCCoE, PQCA (migration groups/movements)

Thank you for listening!

References:

- links in presentation
- previous slide
- write me an email!

Petr Muzikant, petr.muzikant@cyber.ee

https://cyber.ee/

info@cyber.ee

cybernetica

CyberneticaAS

cybernetica_ee

Cybernetica