Demonstration of Flexible and Scalable Quantum-resistant Encryption with Threshold Key Management in Optical Networks

Joo Yeon Cho[(1)], Andrew Sergeev[(2)], Sai Patri[(3)]

[(1)] ADVA Optical Networking SE, jcho@adva.com
[(2)] ADVA Optical Networking Israel Ltd., asergeev@adva.com
[(3)] ADVA Optical Networking SE, spatri@adva.com

Abstract *We demonstrate a flexible and scalable quantum-resistant encryption framework applicable to existing optical networks. Computationally intensive key exchange processes are offloaded on a centralized crypto module while encryption keys are converted into n shares and distributed to key servers in such a way that those keys can be securely reconstructed by a threshold key management protocol.*

Introduction

Quantum threat is well recognized in optical networks since strong encryption over fiber is a key requirement of many applications e.g. the critical infrastructure. While data encryption using a symmetric-key cryptography (e.g. AES) can survive with enlarged key size, key exchange or authentication schemes using public-key cryptography (e.g. RSA) are in danger.

To defeat this threat, two approaches are usually taken in the industry. One is to establish a Quantum Key Distribution Network (QKDN) and operate a key management system, by which secret keys are supplied from QKDN to data encryptors. The other is to implement post-quantum (PQ) cipher suites directly on data encryptors and run a PQ key exchange protocol as well as data encryption on the same machine.

Although both approaches have been investigated for a while, industry is unpleasant to deploy these methods widely on existing optical networks because the details of the methodology are still under development and, hence, there is no standard to follow. National Institute of Standards and Technology (NIST) is driving the standardization of new PQ crypto algorithms that could withstand quantum attacks[1]. The European Telecommunications Standards Institute (ETSI) and the International Telecommunication Union (ITU) put efforts on the standardization of QKD and its application on telecommunication networks at scale[2],[3]. Hence, from an industrial perspective, it is necessary to adapt both approaches in parallel until the standard processes are finalized for the sake of satisfying various user requirements.

However, deploying quantum-secure solutions in optical networks is not a simple task in practice. QKD suffers from a limit of distance and the requirement of expensive hardware equipment. PQC solutions are flexible and easy to apply, nevertheless they often require a powerful CPU and relatively large memory resources that most of embedded platforms in optical networks may not have. In particular, it is likely that NIST will announce multiple finalists in the end of the PQC project[4]. Hence, it is necessary to accommodate multiple PQC solutions in the platform, which may cause a significant change in the hardware architecture e.g. CPU, memory, etc. A commercial PQ crypto chipset or a FPGA-based accelerator might be another option but it will take a while until such packages are available in the market and obviously this approach is even more challenging in the industry.

Innovation

We demonstrate a new framework of quantum-secure solutions deployable on existing optical networks in a flexible and scalable manner. In our framework, both QKD and PQC are handled as standalone key suppliers. Secret keys are accessed from a key server of either QKD or PQC, depending on the user requirement, to data encryptors in a unified key interface.

In particular, a PQ key exchange protocol is executed on a dedicated crypto module (which we call CryptoM). CryptoM is capable of hosting multiple key exchange processes and supplying session keys to data encryptors on their requests. Each session key is identified by a keyID and tied to a specific encryptor.

Furthermore, in order to minimize the latency of key delivery, key exchange protocols are periodically executed and secret keys are generated

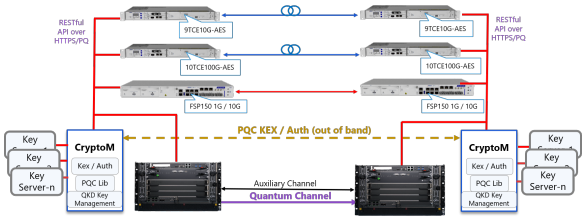

Fig. 1: A block diagram of the demo system: quantum-resistant encryption with $(1, 3)$-threshold key management

before requested. However, this scenario needs to store secret keys in a certain place, which has a potential risk of a single point of failure. To eliminate such risk, we implement a (t, n)-threshold key management protocol. A set of servers can jointly act as a key server in a way that no individual server knows any of the secret keys, and so that services remain available and a secret key is correctly reconstructed as long as a certain threshold number of servers have not been hacked or taken offline.

System setup

Our demo system consists of three building blocks; user data encryption (DE), authenticated key supply (AKS), and threshold key management (TKM), each of which targets 128-bit quantum security. A block diagram of the demo system configuration is shown in Fig. **??**.

DE: User data are encrypted using a symmetric-key crypto algorithm such as AES-256-GCM. Due to the quadratic speedup of a key exhaustive search by Grover's algorithm on quantum computers[5] encryption using a 256-bit key would suffice a target security.

AKS: A symmetric encryption key is derived by executing either a QKD protocol such as BB84 or a PQ key exchange protocol such as IKE/PQ. The derived key is delivered to an encryptor via RESTful APIs over HTTPS/PQ. Note that a PQ authentication should be completed before a key delivery process is initiated[6].

TKM: Secret keys are actually not stored in CryptoM, rather they are converted into n shares and distributed to multiple key servers. When requested, CryptoM acts as a dealer and collects n shares from the key serves and reconstructs a key by a threshold key management protocol. A secret key is correctly reconstructed as long as at least any $t + 1$ out of n shares are correct.

Post-quantum crypto primitives

Although the standardization process is on-going, the 3rd round finalists of NIST PQC project would be the best candidates for PQ key exchange and signature primitives. They are listed in Table **??**. In addition, hash-based signatures should be counted since they have been already standardized in IETF[7],[8] and supported by NIST[9]. Note that KEM stands for Key Encapsulation Mechanism by which a data encryption key is derived. Signature schemes are typically used for the entity authentication.

Tab. 1: PQC primitives: the 3rd round finalists from NIST[1] and hash-based signatures from IETF[9]

SDO	***PQ KEM***	***PQ Signature***
NIST	Classic McEliece CRYSTALS-KYBER NTRU SABER	DILITHIUM FALCON Rainbow
IETF	-	XMSS LMS

Each primitive provides multiple parameter sets for different security levels. The target security proposed in this demo falls on Category 5 which is the strongest security level equivalent to that of AES-256. The downside is a large size of a public/secret key. For example, Classic McEliece defines more than 1M bytes of a public key for Category 5 security. We demonstrate our framework can accommodate all the primitives listed in Tab. **??**. Among various parameter sets, we chose those of the category 5; mceliece6960119 (Classic McEliece), ntruhps4096821 (NTRU), kyber1024 (Crystal-Kyber) and FireSaber (Saber) for KEM, Crystal-Dilithium (Dilithium IV) and Falcon1024 (Falcon) for digital signature.

Integrating QKD system

We integrated our testbed with ID Quantique QKD system (Cerberis3) as well as Toshiba QKD system successfully, both of which are based on the BB84 QKD protocol. Unfortunately, none of these QKD vendors are involved in our demo proposal. Hence, the QKD system itself is not a part of our demo.

PQ Secure key delivery interface

HTTPS is a widely used secure communication protocol based on Transport Layer Security (TLS) and it is used for the QKD key delivery in the ETSI standard[2]. HTTPS/PQ is an extended version of the HTTPS protocol using post-quantum signature schemes in Tab. **??**. In our framework, a secret key in QKD and CryptoM is accessible with a set of RESTful APIs which are listed in Table **??**.

An example of the key delivery flow, as shown in Fig. **??**, is briefly explained as follows.

1. NCU_A sends a `Get Key` to $CryptoM_A$.

Tab. 2: RESTful API for key delivery

API Name	***Description***
GetKey	Request a key
GetKey with KeyID	Request a key that keyID matches
Get Status	Request the status of key storage

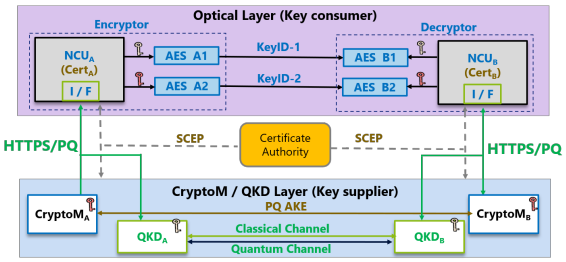

Fig. 2: An example of the key delivery flow for quantum-resistant optical communication

2. $CryptoM_A$ and NCU_A mutually authenticate using PQ certificates.
3. $CryptoM_A$ sends an encrypted secret key with its keyID to NCU_A.
4. NCU_A transfers the keyID to NCU_B.
5. NCU_B sends a `GetKey` with the KeyID to $CryptoM_B$.
6. $CryptoM_B$ and NCU_B mutually authenticate using PQ certificates.
7. $CryptoM_B$ sends to NCU_B an encrypted secret key that the keyID is matched.

PQ key exchange and management

Suppose I and J denote a set of encryptors maintained by $CryptoM_A$ and $CryptoM_B$, respectively. Then, an authenticated key exchange (AKE) protocol for $NCU_A \in I$ and $NCU_B \in J$ is performed as drawn in Fig. **??**.

Once an AKE protocol is completed, a secret key is shared in both CryptoM modules. Then, this secret key is converted into n shares using Shamir's secret sharing scheme[12] and distributed to n key servers. When a secret key is requested, CryptoM aggregates at least $t+1$ shares from the key servers, reconstructs the secret key and delivers it to the user. After delivered, the secret key is completely deleted from the key servers and is never used again.

To instantiate our scheme, a Shamir secret sharing scheme with $(t, n) = (2, 5)$ is implemented in our demo. A degree-2 random polynomial $q(x) = a_2x^2 + a_1x + a_0$ is generated and $a_0 = K$ is set. To share the secrets, five points on the curve are randomly chosen and distributed to five key servers, respectively. Hence, it is possible to reconstruct the $q(x)$ if any three points are correctly collected.

$CryptoM_A$		$CryptoM_B$
Required: pk_i, sk_i Initiate AKE for i		Required: pk_j, sk_j Initiate AKE for j
	$\overset{Start}{\longleftrightarrow}$	
Generate r_i, k_i $X_i = \text{SIGN}_{sk_i}(i, pk_i, r_i)$		Choose r_j, k_j $X_j = \text{SIGN}_{sk_j}(j, pk_j, r_j)$
	$\overset{X_i, X_j}{\longleftrightarrow}$	
Verify X_j and pk_j		Verify X_i and pk_i
$Y_i = \text{ENC}_{pk_j}(k_i)$		$Y_j = \text{ENC}_{pk_i}(k_j)$
	$\overset{Y_i, Y_j}{\longleftrightarrow}$	
$k_j = \text{DEC}_{sk_i}(Y_j)$		$k_i = \text{DEC}_{sk_j}(Y_i)$
$K = k_i \Vert k_j \Vert r_i \Vert r_j$ $M_i = H(K)$		$K = k_i \Vert k_j \Vert r_i \Vert r_j$ $M_j = H(K)$
	$\overset{M_i, M_j}{\longleftrightarrow}$	
Verify M_i		Verify M_j

Fig. 3: Authenticated key exchange protocol

Our implementation can be extended for $n > 5$. However, the complexity of reconstruction process is exponentially increased accordingly since every $\binom{n}{t+1}$ combinations of shares should be tested before a correct key is derived.

Conclusions

Even though a large scale of quantum computers are not arrived yet, deployment of new solutions against quantum threats is already on the way in industry. However, it is quite challenging to implement those solutions on existing devices, in particular, on optical networking systems which are based on resource-limited embedded platform. In this paper, we presented a framework of flexible and scalable quantum-resistant encryption combined with threshold key management. We minimized the requirement of change on optical networking systems; only an external key interface needs to be implemented. Instead, a centralized crypto module takes a role of key exchange and a key service to the devices in combination with threshold key management. Our experiments show that a quantum-resistant solution can be deployed on existing optical networking system with minimal changes.

Acknowledgements

This research is co-funded by OpenQKD project under the Horizon 2020 Framework Program of the European Union (Grant agreement No 857156) and by the QuaSiModO project under the Federal Ministry of Education and Research of Germany (Grant agreement No 16KIS1051).

References

[1] G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, Y. Liu, C. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, and D. Smith-Tone, "Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process", Jul. 2020.

[2] ETSI, *Quantum key distribution (qkd)*, https://www.etsi.org/technologies/quantum-key-distribution.

[3] ITU, *New itu standard for networks to support quantum-safe encryption and authentication*, https://news.itu.int/new-itu-standard-networks-support-quantum-safe-encryption-authentication/.

[4] W. Barker, W. Polk, and M. Souppaya, *Getting ready for post-quantum cryptography*, NIST cybersecurity white paper, https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.05262020-draft.pdf, May 2020.

[5] K. Grover, "A fast quantum mechanical algorithm for database search", in *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, ser. STOC '96, ACM, 1996, pp. 212–219.

[6] D. Sikeridis, P. Kampanakis, and M. Devetsikiotis, *Post-quantum authentication in tls 1.3: A performance study*, Cryptology ePrint Archive, Report 2020/071, https://eprint.iacr.org/2020/071, 2020.

[7] A. Huelsing, D. Butin, S. Gazdag, J. Rijneveld, and A. Mohaisen, "XMSS: Extended Hash-Based Signatures", Internet Engineering Task Force, Internet-Draft draft-irtf-cfrg-xmss-hash-based-signatures-12, Jan. 2018, Work in Progress, 72 pp.

[8] D. McGrew, M. Curcio, and S. Fluhrer, *Leighton-Micali Hash-Based Signatures*, RFC 8554, Apr. 2019. [Online]. Available: https://rfc-editor.org/rfc/rfc8554.txt.

[9] D. Cooper, D. Apon, Q. Dang, M. Davidson, M. Dworkin, and C. Miller, *Recommendation for stateful hash-based signature schemes*, Draft NIST Special Publication 800-208, NIST.SP.800-208-draft.pdf, Dec. 2019.

[10] E. Rescorla, "The transport layer security tls protocol version 1.3, ietf rfc 8446", Mar. 2016.

[11] F. Günther, M. Thomson, and C. Wood, "Usage limits on aead algorithms", Aug. 2020, https://www.ietf.org/id/draft-irtf-cfrg-aead-limits-00.txt.

[12] A. Shamir, "How to share a secret", *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.