JOURNAL OF
Optical Communications
and Networking

Extending quantum key distribution through proxy re-encryption

Nathan Lemons,[1] Boris Gelfand,[2] Nigel Lawrence,[2] Austin Thresher,[2] Justin L. Tripp,[3] William Pierre Gammel,[4,5] Aniruddha Nadiga,[1] Kristina Meier,[6,*] and Raymond Newell[7]

[1]Theoretical Division, Los Alamos National Laboratory, Los Alamos, New Mexico 87545, USA
[2]Analytics, Intelligence and Technology Division, Los Alamos National Laboratory, Los Alamos, New Mexico 87545, USA
[3]Computer, Computational and Stats Sciences Division, Los Alamos National Laboratory, Los Alamos, New Mexico 87545, USA
[4]Program in Applied Mathematics, The University of Arizona, Tucson, Arizona 85721, USA
[5]X-Computational Physics Division, Los Alamos National Laboratory, Los Alamos, New Mexico 87545, USA
[6]Intelligence and Space Research Division, Los Alamos National Laboratory, P.O. Box 1663, MS B244, Los Alamos, New Mexico 87545, USA
[7]Materials Physics and Applications Division, Los Alamos National Laboratory, Los Alamos, New Mexico 87545, USA
*kameier@lanl.gov

Received 8 September 2022; revised 8 May 2023; accepted 25 May 2023; published 29 June 2023

Modern quantum key distribution (QKD) network designs are based on sending photons from one node to another and require free-space or dedicated fiber optic cables between nodes. The purpose of this is to co-generate secret key material on both sides of the quantum channel. In addition to this quantum link, there are several insecure classical channels that allow QKD algorithms to exchange book-keeping information and send symmetrically encrypted data. The attenuation of photons transmitted through fiber becomes too high to practically generate key material over fiber at distances of more than 100 km. Free-space transmission through the atmosphere or the vacuum of space can reduce attenuation, but at the cost of system complexity and sensitivity to other impairments, such as weather. To extend the effective range of QKD networks, we present a method that combines QKD algorithms with post-quantum, homomorphic key-switching to allow multiple parties to effectively share secret key material over longer distances through semi-trusted relay nodes. We define how such a system should work for arbitrary network topologies and provide proofs that our scheme is both correct and secure. We assess the feasibility of this solution by building and evaluating two implementations based on lattice-based cryptography: learning with errors. © 2023 Optica Publishing Group under the terms of the Optica Open Access Publishing Agreement

https://doi.org/10.1364/JOCN.474487

1. INTRODUCTION

Key distribution schemes allow multiple parties to agree upon a secret key or keys that can be used to encrypt and decrypt messages sent between parties. Most modern cryptographic systems today rely upon public key infrastructures in which users generate public/private key pairs and deposit their public keys on a centralized server. When Alice wants to send Bob an encrypted message, Alice queries a trusted server for Bob's public key and uses that to encrypt the message. While public key encryption is convenient, it is generally easier to attack and less efficient in implementation. Furthermore, a centralized and trusted server provides a single major focal point for adversaries to attack or disrupt the whole system. Because of the computational overhead of public/private key encryption, it is used primarily to establish temporary symmetric private keys, called session keys, between users. Any data exchanged between users is encrypted with symmetric key encryption using these temporary session keys [1]. For symmetric key encryption, two parties have to agree on a shared secret key. As in the above mechanism, they can do so by relying on public/private key infrastructure. Another obvious, but less practical, way is to meet privately and agree on a shared secret key, or use some other secure out-of-band communication to share the key.

A relatively new and novel way to generate secret shared keys for use in symmetric key encryption is quantum key distribution (QKD) [2], which uses fundamental properties of quantum mechanics to allow symmetric keys to be co-generated between physically distant parties. That is, Alice and Bob do not need to meet privately or use public/private infrastructure to agree upon a strong (random) shared key. Using QKD algorithms, such as BB84, the quantum mechanical properties of photons ensure that the secret key is distributed to Alice and Bob without the possibility of interception or compromise by eavesdroppers [3]. In particular, potential

eavesdroppers will either be detected or will only observe an impractically small portion of the key(s) being co-generated. The knowledge an eavesdropper (Eve) obtains about a key can be made arbitrarily small through a technique called privacy amplification [4]. This results in an information theoretical guarantee that Eve cannot learn anything about the shared secret key; attacks through interception are guaranteed to fail. Other attacks such as side-channel attacks are possible and may be successful [5–7]. This security does, however, come with a cost: to agree on a key, Alice and Bob may send only single photons at a time—a tall engineering order! If multiple copies of identical photons are sent, it is possible that an eavesdropper could intercept and measure some of the copies without detection. This limitation has several practical consequences. Most notably for our purposes, when photons are transferred over optical fiber, the photon attenuation of this channel limits practical QKD to distances of no more than approximately 80 km [8].

To distribute keys over longer distances, new enabling technologies must be found. "Quantum relays" present an attractive option; however, such relays present extraordinary technical challenges and are not expected to be suitable for wide-scale deployment within 10 years at least. In particular, current embodiments of quantum relays transmit information about quantum states over classical channels and therefore offer no security advantages over conventional cryptography schemes [9,10].

A second option, which is more practical in the near and mid terms, is a classical relay node. This type of relay node acts as a middleman between long-distance parties. Consider a classical relay node, named Ray, between Alice and Bob: two parties who want to exchange messages with each other via Ray while putting as little trust in Ray as possible. Ray can separately generate two sets of secure keys with Alice and Bob, assuming Ray is trusted by both Alice and Bob. This trusted relay approach is the current industry standard and has been utilized in many quantum networks [11–18]. However, it is done by physical security and is less practical and more expensive than using a cryptographic solution. To alleviate the need to fully trust the relay nodes, we describe a key proxy re-encryption scheme to enable practical relay nodes operating under significantly fewer security assumptions. Furthermore, our protocols and algorithms can generalize to quantum distribution networks with multiple relay nodes of various topologies. In Section 3, we describe a generic method of incorporating proxy re-encryption within a QKD network to provide less trusted relay nodes. In Section 4, we give two concrete examples of such a scheme using encryption methods based on the learning with errors (LWE) problem. We give proofs of security and discuss their relative merits.

2. BACKGROUND

A. Quantum Key Distribution

QKD came about in 1984 with Charles Bennett and Gilles Brassard's seminal work titled "Quantum cryptography: public key distribution and coin tossing" [4]. In this paper, they describe the BB84 protocol where a transmitter (Alice) sends single photons to a receiver (Bob) to create a secret key that

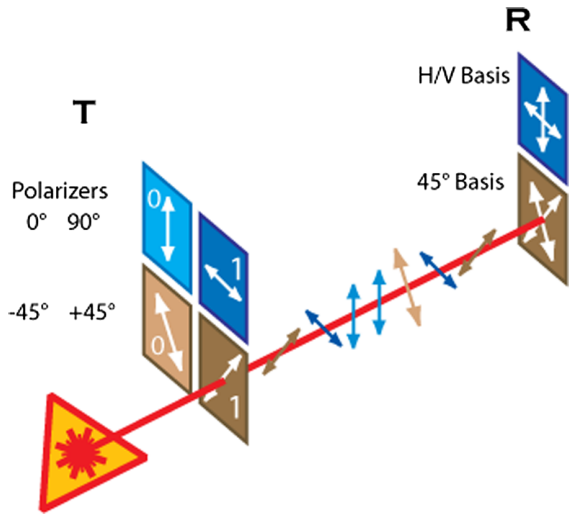

Fig. 1. BB84 protocol for generating shared secret random numbers. Alice (transmitter, T) generates single photons, randomly selects a basis value (horizontal/vertical in blue, or diagonal/anti-diagonal in brown), and then randomly selects a bit value, 0 or 1. These choices determine the polarization state of the photon that is then sent to Bob (receiver, R). For each photon, Bob randomly chooses the basis in which to measure the polarization. In the subset of cases where Alice and Bob use the same basis, the resulting polarization measurement can yield a shared secret random bit.

can later be used to encrypt a message for transmission over a classical channel, as shown in Fig. 1. The security of QKD is guaranteed by quantum mechanics and does not depend on computational complexity. Over the past four decades, other versions of QKD have been developed to improve the security of the BB84 protocol by protecting against various attacks that could be made by an eavesdropper (Eve). Notably, entanglement-based QKD, such as the E91 protocol by Artur Ekert in 1991, leverages properties of quantum entanglement to detect Eve [19]. The presence of an eavesdropper would break strong quantum correlations, thereby alerting Alice and Bob. QKD has proven itself to be a practical method for quantum-secured information transfer, and fiber-based [20], free-space [21], and satellite-mounted [22] versions have been accomplished. The current limiting factor is the inability to send single photons over long distances without insurmountable channel loss, which this work aims to help overcome through the use of trusted relay nodes, as described in Fig. 2.

B. Learning with Errors

The LWE problem, introduced by Oded Regev [23,24], is a computational problem on lattices involving the learning of noisy inner products. It is conjectured to be a hard problem for both classical and quantum computers given certain sets of parameters and configurations. Thus encryption based on the LWE problem offers a degree of security that current cryptographic methods, e.g., factoring or solving the discrete logarithm, do not. Such lattice-based problems are well studied, and the best known quantum and classical algorithms to solve these problems run in time $O(2^n)$ [25]. Because of this, LWE-based encryption is assumed to be *post-quantum secure*; even with functional quantum computers, the encryption is

Fig. 2. Quantum Hardware Security Module (Q-HSM) designed and built at Los Alamos. Each Q-HSM device acts as a quantum key transceiver, with transmit and receive capability included. The transmitter comprises an attenuated laser that produces weak coherent pulses of light, with polarization per pulse set in one of four states. The receiver comprises an avalanche photodiode single-photon detector, optics for polarization analysis, and custom microelectronics for synchronization and control.

still computationally impractical to break. The LWE problem is extremely versatile; a multitude of cryptographic schemes rely on it, including two of the four current finalists in NIST's post-quantum cryptography [26] standardization process (a third, NTRU, is closely related [27]).

Perhaps surprisingly, the LWE problem is also simple to state and understand. The problem asks to recover a secret $\mathbf{s} \in \mathbb{Z}_q^n$ given $\mathbf{a}_i$ and b_i, where b_i is given by the approximate linear equation over the finite field $\mathbb{Z}_q$:

$$b_i := \mathbf{a}_i \cdot \mathbf{s} + \epsilon_i. \tag{1}$$

Note that an adversary with access to $\mathbf{a}_i$ and b_i can use Gaussian elimination to determine $\mathbf{s}$ if there is no noise term ϵ_i. However, with the addition of a small amount of noise chosen from an appropriate distribution, the problem of finding $\mathbf{s}$ becomes essentially intractable; it is as hard as certain worst-case lattice problems and takes exponential time to find $\mathbf{s}$. To put this in perspective, the best known algorithms run in time $O(2^n)$. While this construction is very simple, it has proven to be extremely fruitful. There are a host of important cryptographic primitives that have been realized through LWE, including the well-known fully homomorphic encryption. There are also many variants of LWE studied in the literature; perhaps the most popular is ring-LWE, which achieves much better efficiency than plain LWE through the sacrifice of some simplicity. Here we focus on LWE due to its simplicity and note that ring-LWE could be used in place of LWE.

Theorem 2.1 (Regev) *LWE is chosen-plaintext attack (CPA)-secure* [24]: Regev proved that if LWE is not CPA-secure, then there exists an efficient quantum algorithm to solve certain lattice problems including the "shortest vector problem" (SVP). This and related problems have been widely studied for decades and are believed to be hard for both classical and quantum computers. Later, Brakerski *et al.* proved that if LWE is not semantically secure, then there exists an efficient classical algorithm to solve SVP [28].

C. Homomorphic Encryption

Fully homomorphic encryption gives a third party the ability to perform arbitrary computations on encrypted data without access to the encryption keys in such a way that the resulting ciphertexts can be decrypted by the original data owner. The ability to perform fully homomorphic encryption under reasonable assumptions was only recently discovered. Note that, under LWE, ciphertexts encrypted under the same key can be added together, and the result is an encryption of the sum of the original messages. However, to go from addition to arbitrary computation required more thought. Performing multiplication on ciphertexts was possible, but required the sizes of the keys to grow. To control this, key-switching was invented to allow for a reduction in the sizes of keys. Key-switching transforms a ciphertext as an encryption under a key $\mathbf{s}_1$ to a ciphertext encrypted under a different key $\mathbf{s}_2$ without revealing either key. In addition, $\mathbf{s}_2$ can be significantly smaller than $\mathbf{s}_1$. In this work, we re-purpose this technology to allow for proxy re-encryption along QKD links, using a simplified version of key-switching where the sizes of the keys remain fixed.

D. Proxy Re-encryption

Suppose Alice and Bob have two different secret keys, sk_1 and sk_2, respectively. Alice would like to send an encryption of a message m using sk_1 to Bob via an untrusted relay. The relay would need to transform the ciphertext into an encryption of the same message under sk_2 without having access to sk_1, sk_2, or m. In addition, the information available to the relay (allowing it to perform the transformation) should not help the relay obtain any information on sk_1, sk_2, or m. In other words, it should be just as hard for the relay to break the encryption as it is for any other untrusted party. Proxy re-encryption schemes allow exactly these types of protocols to be implemented. Blaze, Bleumer, and Strauss were the first to provide an implementation of such a scheme, and since then, a multitude of competing schemes have been introduced [29–33]. While most of the literature concentrates solely on public key schemes, we are interested in proxy re-encryption based around symmetric key encryption. For completeness, we define such a scheme below.

Definition 2.2. A *symmetric-key encryption scheme* is a set of three algorithms:

- $\mathsf{SecretKeyGen}$: generates a secret key, sk, used for encryption (Algorithm 1).
- Enc_{sk}: takes as input a message m, outputs an encryption of m (Algorithm 2).
- Dec_{sk}: takes as input some ciphertext that encrypts m, outputs m (Algorithm 3).

Definition 2.3. A *symmetric-key proxy re-encryption scheme* consists of a symmetric key encryption scheme and two additional algorithms:

- $\mathsf{TransformGen}$: takes as inputs two keys $\mathbf{sk}_1$ and $\mathbf{sk}_2$ and returns the transforms $T_{\mathbf{sk}_2 \leftarrow \mathbf{sk}_1}$ and $T_{\mathbf{sk}_1 \leftarrow \mathbf{sk}_2}$ (Algorithm 4).
- $\mathsf{Trans}_{\mathbf{sk}_2 \leftarrow \mathbf{sk}_1}$: takes as input a ciphertext c_1 encrypted under $\mathbf{sk}_1$ and returns a ciphertext c_2 satisfying $\mathsf{Dec}_{\mathbf{sk}_2}(c_2) = \mathsf{Dec}_{\mathbf{sk}_1}(c_2)$ (Algorithm 5).

Encryption and proxy re-encryption schemes are generally useful only if one has a sense of their security, that is, a measure of how hard it is for an attacker to break the schemes. A scheme

Algorithm 1. `SecretKeyGen`

Input: $n, q \in \mathbb{Z}$
$\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$
Output: $\mathbf{s} \in \mathbb{Z}_q^n$

Algorithm 2. `Encryption`

Input: $\mathbf{s} \in \mathbb{Z}_q^n$, $m \in \{0, 1\}$
$\mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^n$
$\epsilon \xleftarrow{\$} \Gamma_{\alpha,n,q}$
$b \leftarrow 2\epsilon + m - \mathbf{a} \cdot \mathbf{s}$
Output: $(b, \mathbf{a}) \in \mathbb{Z}_q^{n+1}$

Algorithm 3. `Decryption`

Input: $\mathbf{s} \in \mathbb{Z}_q^n$, $m \in \{0, 1\}$, $(b, \mathbf{a}) \in \mathbb{Z}_q^{n+1}$
$m \leftarrow \mathbf{b} + \mathbf{a} \cdot \mathbf{s}$ (modulo 2)
Output: m

Algorithm 4. `TransformGen`

Input: $\{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k\} \in \mathbb{Z}_q^{nk}$
for $i = 2, \ldots, k$ **do**
  $\mathbf{T}_{1\leftarrow i} \leftarrow \mathbf{s}_i - \mathbf{s}_1$
Securely delete $\{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k\}$
$\mathbf{T}_{1\leftarrow 1} \leftarrow \mathbf{0}$
for $i = 1, 2, \ldots, k$ **do**
  for $j = 1, 2, \ldots, k$ **do**
    $\mathbf{T}_{i\leftarrow j} \leftarrow \mathbf{T}_{1\leftarrow j} - \mathbf{T}_{1\leftarrow i}$
Output: $\{\mathbf{T}_{i\leftarrow j}\}_{i,j=1}^{n}$

Algorithm 5. `KeySwitch`

Input: $\mathbf{c}_1 \in \mathbb{Z}_q^{n+1}$, $\tau_{\mathbf{s}_1\to\mathbf{s}_2} \in \mathbb{Z}_q^{n+1} \cdot \lceil \log q \rceil \times (n \times 1)$
$\mathbf{c}_2 \leftarrow \mathtt{BitDecomp}(\mathbf{c}_1)^T \cdot \tau_{\mathbf{s}_1\to\mathbf{s}_2}$
Output: $\mathbf{c}_2 \in \mathbb{Z}_q^{n+1}$

is said to be information-theoretically secure if it cannot be broken by an adversary with unlimited computational power and unlimited access to sample encrypted ciphertext. An example of such a scheme is QKD where the generated quantum keys are used as one-time pads. However, one time pads are often impractical and alternate schemes relying on computational complexity arguments are used. If any information that can be efficiently computed from a ciphertext could also have been efficiently computed knowing only the length of the ciphertext, then that scheme is said to be semantically secure. Note that an information-theoretically secure scheme can still be vulnerable to attacks that do not solely rely on examining the encrypted ciphertexts, such as man-in-middle or side-channel attacks.

3. APPROACH

A. QKD with Proxy Re-encryption

Now we define a generic QKD with a proxy re-encryption scheme that allows a relay node to be untrusted. The scheme consists of two parts. During the first part, the *Trusted Setup*, the relay is as trusted as the endpoint nodes, Alice and Bob. Once the *Trusted Setup* phase is completed, the *Normal Operations* phase begins, and the relay is fully untrusted.

These security assumptions are consistent with a use case in which a system is installed and configured by a trusted service technician for the setup phase, but is then left unattended during normal operations. Any unauthorized personnel who access the relay node during normal operations will not learn any of the secret keys, nor have any access to the cleartext messages. This is in contrast to the existing method of daisy-chaining trusted nodes where each node has locally stored copies of the keys, and all messages must exist in cleartext as they pass through the node.

Definition 3.1. A *QKD with a trusted relay node* scheme consists of two parts: a *Trusted Setup* and *Normal Operations*. During the *Trusted Setup*, the following algorithms are performed:

1. Along each quantum link in the network, secret keys are generated via SecretKeyGen (Algorithm 1).
2. Each relay node in the network now computes the necessary transformations using the secret keys generated in step 1 as inputs to TransformGen (Algorithm 4).
3. Finally, all relay nodes securely delete the secret keys generated over their quantum links.

During *Normal Operations*, Alice can securely communicate with Bob via a relay node, Ray, as follows:

1. Alice encrypts message m under her secret key sk_1, producing ciphertext, c_1 (Algorithm 2).
2. Alice sends Ray c_1.
3. Ray transforms c_1 into an encryption of m under sk_2 using the transform $\mathsf{Trans}_{\mathbf{sk}_2\leftarrow\mathbf{sk}_1}$, producing c_2 (Algorithm 5).
4. Ray sends c_2 to Bob.
5. Bob decrypts c_2 using his key sk_2, recovering the message m (Algorithm 3).

Security during the normal operation phase means that the proxy re-encryption scheme is just as secure as the underlying encryption scheme itself. In other words, an attacker with access to the encrypted ciphertexts, as well as the key transforms $\mathsf{Trans}_{\mathbf{sk}_2\leftarrow\mathbf{sk}_1}$ will have no more luck obtaining information from the ciphertexts than would an attacker with access to just the ciphertexts. By extension, if the underlying encryption scheme is secure, then the proxy-encryption scheme is also secure.

The basic scheme for exchanging encrypted messages between Alice and Bob through one relay node is shown in Fig. 3. Note that this scheme reduces the amount of trust Alice and Bob need to place in Ray; Ray needs to be trusted only during the key generation and setup steps. Afterwards, during the transmission of encrypted messages, Ray can be compromised. In particular, we prove that an adversary can observe transforms A and B, as well as the encrypted messages passed

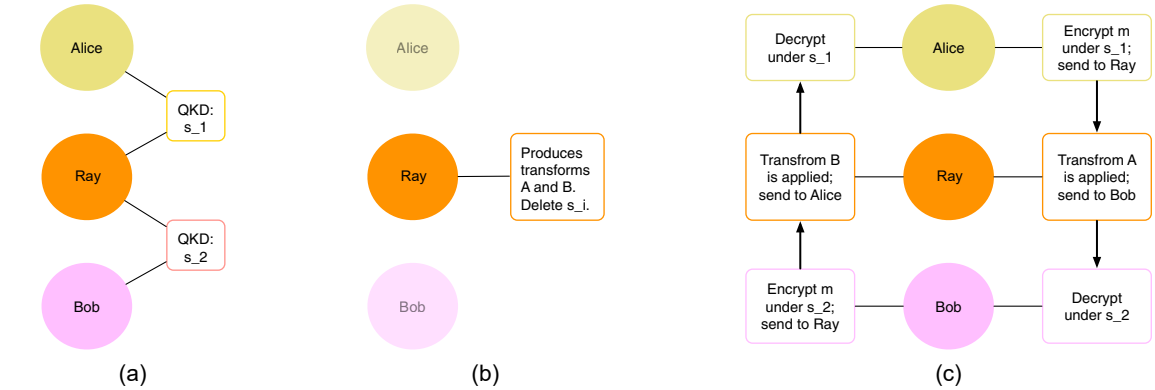

Fig. 3. (a) Intial key generation and (b) setup. During these two steps, Ray is implicitly as trusted as Alice and Bob. (c) After setup, Alice and Bob can securely exchange encrypted messages. Ray need no longer be trusted during this step. (a) Alice and Bob establish secret keys s_1 and s_2, respectively, with Ray. (b) Ray produces proxy re-encryption transforms, A and B, then deletes both secret keys from memory. (c) Alice and Bob can encrypt and decrypt messages to each other provided Ray applies the appropriate transform.

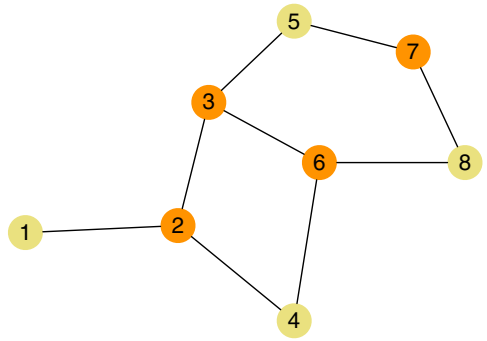

Fig. 4. Example of a more complicated topology. Yellow nodes are trusted nodes; orange nodes are relays. When node 1 sends a message to node 5 the message undergoes two transformations—one each by nodes 2 and 3—before reaching 5.

along by Ray; knowledge of transforms A and B does not help in decrypting encrypted messages.

While Fig. 3 depicts communication between Alice and Bob through only one relay node, our scheme can be generalized to any topology. See Fig. 4 for a more complicated topology. To generalize the scheme to such topologies, we assume that each node knows the network topology (and can thus correctly route its messages). Furthermore, each relay node will need to produce $\binom{d}{2}$ transforms, where d denotes the number of neighbors it has.

This basic scheme allows users of a quantum distribution network with multiple possible relays to communicate securely. In particular, the relay nodes do not need to be trusted during normal operations; trust is placed in them only during the system setup steps. When the system is in the setup mode, additional security measures such as physical security are used to guarantee the trustworthiness of the relay nodes. Note that this can be a problem if the keys need to be refreshed frequently, if new nodes are often added or removed, and if the system often breaks down necessitating a fresh setup. In one of the two implementation methods described below, key refreshes are handled during normal operations without requiring any additional trust of the relay nodes.

B. Network Security Model

Secure quantum communication and computational-based classical cryptography are typically evaluated with different attacker models. In the conventional case of classical cryptography, attackers are usually modeled as computationally bounded adversaries that are presumed unable to violate a computational hardness assumption. However, in contexts leveraging secure quantum communication, the attacker is assumed to be computationally unbounded and generally makes no assumptions about the computational hardness of a problem. Our proposed network architecture is a hybrid between secure quantum approaches and post-quantum, computationally secure classical schemes, requiring a mixture of the two types of adversaries.

4. IMPLEMENTATIONS

Here we describe two different key-switching schemes. Each algorithm is described in detail with its respective security proof. One difference between the two lies in their relative efficiencies. The Brakerski–Gentry–Vaikuntanathan (BGV) implementation requires $O(n^2)$ arithmetic operations during a key-switching operation, while the XOR implementation uses only $O(n)$ such operations.

A. BGV Implementation

In this first implementation of proxy re-encryption-enabled QKD, we use a simple symmetric LWE encryption scheme with key-switching algorithms taken from the BGV implementation of fully homomorphic computing [34,35]. The LWE encryption algorithm requires small error terms to be generated randomly from an appropriate discrete Gaussian distribution [24] denoted $\Gamma_{\alpha,n,q}$. The parameter values depend on the desired level of security; getting everything including the generation algorithm correct is highly nontrivial [36].

Algorithm 6. `BitDecomp`

1: **Input:** $\mathbf{x} \in \mathbb{Z}_q^m$
2: $\mathbf{x} = \sum_{i=0}^{\lfloor \log q \rfloor} 2^i \mathbf{u}_i$, where $\mathbf{u}_i \in \mathbb{Z}_2^m$.
3: **Output:** $(\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{\lfloor \log q \rfloor}) \in \mathbb{Z}_2^{m \cdot \lceil \log q \rceil}$

Algorithm 7. `Powersof2`

1: **Input:** $\mathbf{x} \in \mathbb{Z}_q^m$
2: **Output:** $\mathbf{x}, \mathbf{2x}, \ldots, 2^{\lfloor \log q \rfloor}\mathbf{x} \in \mathbb{Z}_q^{m \cdot \lceil \log q \rceil}$

Algorithm 8. `SwitchGen`

1: **Input:** $\mathbf{s}_1, \mathbf{s}_2 \in \mathbb{Z}_q$
2: $\boldsymbol{A}_1 \xleftarrow{\$} \mathbb{Z}^{(n+1)\cdot\lceil \log q \rceil \times n}$
3: $\boldsymbol{A}_2 \xleftarrow{\$} \mathbb{Z}^{(n+1)\cdot\lceil \log q \rceil \times n}$
4: $\mathbf{e}_1 \xleftarrow{\$} \Gamma_{\alpha,(n+1)\cdot\lceil \log q \rceil,q}$
5: $\boldsymbol{e}_2 \xleftarrow{\$} \Gamma_{\alpha,(n+1)\cdot\lceil \log q \rceil,q}$
6: $\mathbf{b}_1 \leftarrow 2\mathbf{e}_1 - \mathbf{A}_1 \cdot \mathbf{s}_2$
7: $\mathbf{b}_2 \leftarrow 2\mathbf{e}_2 - \mathbf{A}_2 \cdot \mathbf{s}_1$
8: $\tau_{\mathbf{s}_1 \to \mathbf{s}_2} \leftarrow (\mathbf{b}_1 + \texttt{Powersof2}((1, \mathbf{s}_1)), \mathbf{A}_1)$
9: $\tau_{\mathbf{s}_2 \to \mathbf{s}_1} \leftarrow (\mathbf{b}_2 + \texttt{Powersof2}((1, \mathbf{s}_2)), \mathbf{A}_2)$
10: **Output:** $\tau_{s_1 \to s_2}, \tau_{s_2 \to s_1} \in \mathbb{Z}_q^{(n+1)\cdot\lceil \log q \rceil \times (n+1)}$

To generate the key-switching matrix from [35], we first define two auxiliary methods: `BitDecomp` (Algorithm 6) and `Powersof2` (Algorithm 7). `BitDecomp` writes $\mathbf{x}$ as a binary vector in the space $\mathbf{Z}_2^{n\lceil \log q \rceil}$. Also note that the output of `Powersof2` is a vector with all of the entries concatenated vertically.

To better understand the `SwitchGen` algorithm (Algorithm 8), one can view the matrices $\mathbf{A}_i$ as many encryptions of zero under key $\mathbf{s}_i$, and $\tau_{\mathbf{s}_1 \to \mathbf{s}_2}$ can be thought of as encryptions in some format of $\mathbf{s}_1$ under key $\mathbf{s}_2$. $\tau_{\mathbf{s}_1 \to \mathbf{s}_2}$ contains the information needed to "decrypt" the message under $\mathbf{s}_1$ while remaining encrypted under $\mathbf{s}_2$. Note that the secret does not come from the message space, so it is not a true encryption; its decryption using Algorithm 3 would be meaningless.

B. XOR Implementation

System setup comprises two algorithms: Secret Key Generation, `SecretKeyGen` (Algorithm 1) and Proxy Transform Generation, `TransformGen` (Algorithm 4). During the execution of these routines, we assume the relay node, Ray, is *as trusted* as Alice and Bob; i.e., Ray is assumed to be completely trustworthy during the setup. After Ray securely deletes the keys $\{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k\}$, the initial trusted setup is complete. From here on, Ray is no longer as trusted as Alice and Bob. In our proofs of security below, we show that a computationally bounded attacker, Eve, observing the ciphertexts sent between Alice and Bob, learns nothing extra from also observing the transforms $\mathbf{T}_{i \leftarrow j}$.

Once the secure setup is complete, the mode of normal operations consists of three algorithms: `XOR Encryption` (Algorithm 9), `ProxyReencrypt` (Algorithm 10), and `XOR Decryption` (Algorithm 11). These are performed as follows. Suppose Alice would like to send a message to Bob. Alice encrypts the message m with her secret key $\mathbf{s}_1$. She sends the resulting ciphertext $\mathbf{c}_1$ to Ray, who then applies the appropriate transform $\mathbf{T}_{2\leftarrow 1}$. Ray then sends the transformed ciphertext $\mathbf{c}_2$ on to Bob who decrypts with his secret key $\mathbf{s}_2$, recovering m. As the encryption scheme in LWE is probabilistic, there is a small probability of decryption error, e.g., Bob recieves m' instead of m. This can effectively be mitigated through error correction. Fortunately, Ray cannot decrypt either $\mathbf{c}_1$ or $\mathbf{c}_2$, as Alice and Bob would like to place as little trust in Ray as possible. However, they do assume Ray will apply the appropriate transform. If he does not do so, Ray is essentially instigating a denial-of-service attack on Alice and Bob.

Algorithm 9. `XOR Encryption`

1: **Input:** $\mathbf{s} \in \mathbb{Z}_q^n$, $m \in \{0, 1\}$
2: $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^n$
3: $\epsilon \xleftarrow{\$} \Gamma_{\alpha,n,q}$
4: **Output:** $(\mathbf{a}, \mathbf{a} \cdot \mathbf{s} + 2\epsilon + m)$

Algorithm 10. `ProxyReencrypt`

1: **Input:** $\mathbf{T}_{j\leftarrow i} \in \mathbb{Z}_q^n$, $(\mathbf{a}, b) \in \mathbb{Z}_q^{n+1}$
2: $b' \leftarrow b + \mathbf{T}_{j\leftarrow i} \cdot \mathbf{a}$
3: **Output:** $(\mathbf{a}, b')$

Algorithm 11. `XOR Decryption`

1: **Input:** $\mathbf{s} \in \mathbb{Z}_q^n$, $(\mathbf{a}, b) \in \mathbb{Z}_q^{n+1}$
2: $m' \leftarrow \mathbf{b} - \mathbf{a} \cdot \mathbf{s}$ (modulo 2)
3: **Output:** m'

C. Key Refresh

Secret keys need to be periodically refreshed, i.e., replaced with new random material to maintain operational security. This can be done using a key encapsulation mechanism where a new key $\mathbf{s}'$ is generated by either Alice or Bob, and then encapsulated into a ciphertext using the old key $\mathbf{s}$. The ciphertext is sent to Bob who can retrieve the new key $\mathbf{s}'$ via decapsulation. Note that neither key-switching nor QKD algorithms are used in this method of refreshing keys.

There is, however, a method whereby Alice and Bob can refresh their keys and Ray's XOR transform using an untrusted Ray and the QKD systems. First Alice and Ray use their QKD links to generate random material $\mathbf{t}_1 \in \mathbb{Z}_q^n$. Bob and Ray also use their QKD links to generate independent random material $\mathbf{t}_2$. Then Alice and Bob perform `XORKeyRefresh` (Algorithm 12) on inputs $(\mathbf{s}_1, \mathbf{t}_1)$ and $(\mathbf{s}_2, \mathbf{t}_2)$, respectively. Concurrently, Ray performs `XORKeyTransfromRefresh` (Algorithm 13) on inputs $(\mathbf{T}_{1\leftarrow 2}, \mathbf{t}_1, \mathbf{t}_2)$.

Algorithm 12. `XORKeyRefresh`

1: **Input:** Secret key $\mathbf{s}$, and random material $\mathbf{t}$
2: $\mathbf{s}' \leftarrow \mathbf{s} - \mathbf{t}$
3: **Output:** $\mathbf{s}'$

Algorithm 13. `XORKeyTransformRefresh`

1: **Input:** $(\mathbf{T}_{1\leftarrow 2}, \mathbf{t}_1, \mathbf{t}_2)$
2: $\mathbf{T}_{1\leftarrow 2}{}' \leftarrow \mathbf{T}_{1\leftarrow 2} - \mathbf{t}_1 + \mathbf{t}_2$
3: **Output:** $\mathbf{T}_{1\leftarrow 2}{}'$

5. SECURITY PROOFS

A. Proof of Security for BGV Key-Switching

The security of the individual secret-key schemes of each edge is well established. Our method connects these schemes through key-switching matrices that contain some relation between the secret keys. We show that publishing these matrices publicly will not leak any information about the keys. Note that the shared node for two edges, l and m, will have key-switching matrices $\tau_{\mathbf{s}_l\to\mathbf{s}_m}$ and $\tau_{\mathbf{s}_l\to\mathbf{s}_m}$. Thus our proof shows that "cycles" of key-switching matrices will not betray any secret information.

1. Outline of Proof

We set up the following game. Let $\mathbf{s}_1, \ldots, \mathbf{s}_u$ be all of the secret keys in the protocol. A challenger has access to $\tau_{\mathbf{s}_l\to\mathbf{s}_m}$ for any ordered pair of secret keys $(\mathbf{s}_l, \mathbf{s}_m)$, and randomly chooses a bit $B \in \{0, 1\}$. In a query, any polynomial-time adversary $\mathcal{A}$ can ask the challenger for any $\tau_{\mathbf{s}_l\to\mathbf{s}_m}$. If $B = 0$, the challenger returns $\tau_{\mathbf{s}_l\to\mathbf{s}_m}$, and if $B = 1$, it returns a uniformly random matrix. $\mathcal{A}$ is allowed any number of queries. If $\mathcal{A}$ guesses the value B with probability greater than 1/2, then the protocol is insecure.

We will define a process $\mathscr{P}$ that has access either to an oracle that can sample an LWE distribution or an oracle that can sample a uniform distribution. The output of $\mathscr{P}$ will be some matrices. We will then define two modified versions of the game. In the first modified version, we suppose that the oracle is for an LWE distribution, and we replace the τ matrices with the matrices output by $\mathscr{P}$. We will show that $\mathcal{A}$ will not be able to distinguish between the original game and the modified game. The second modified game is the same as the previous one, but this time $\mathscr{P}$ is given access to an oracle for uniform distribution. Since decision LWE is hard, the two modified games are indistinguishable from the point of view of $\mathcal{A}$. However, we will show that all of the matrices output by $\mathscr{P}$ in this version will be uniformly random. Thus, all of the matrices output by the original game are indistinguishable from uniformly random matrices.

2. Proof Details

Let the LWE distribution $A_{\mathbf{s},\chi}$ be the distribution with samples of the form $(b, \mathbf{a})$, where $\mathbf{a}$ is uniformly random in $\mathbb{Z}_q^n$, and $b = 2e - \mathbf{a}\cdot\mathbf{s}\rangle$, where $e \leftarrow \chi$. Note that since q is odd, 2 is a unit in the ring $\mathbb{Z}_q$. This means that regardless of whether $2e$ or e is added to calculate b, the hardness of distinguishing the sample from a uniformly random one is unchanged (i.e., the LWE problem is the same hardness for error $2e$ and error e).

First we describe a certain transformation that will take distribution $A_{\mathbf{s}_0,\chi}$ to distribution $A_{\mathbf{x},\chi}$, where $\mathbf{s}_0, \mathbf{x} \in \mathbb{Z}_q^n$ are unknown uniformly random secrets. This same transformation will take uniform distribution $U(\mathbb{Z}_q^n \times \mathbb{Z}_q)$ to itself. Furthermore, this transformation will produce some vector $\mathbf{k}$ such that $\mathbf{s} + \mathbf{k} = \mathbf{x}$:

- Pick a uniformly random $\mathbf{k} \in \mathbb{Z}_q^n$.
- To transform a sample $(\mathbf{a}, b = \langle\mathbf{a}, \mathbf{s}_0\rangle + 2e) \leftarrow A_{\mathbf{s}_0,\chi}$, then output $(\mathbf{a}' = \mathbf{a}, b' = b + \langle\mathbf{a}', \mathbf{k}\rangle)$.

$\mathbf{a}'$ is uniformly random, and $b' = \langle\mathbf{a}', \mathbf{s} + \mathbf{k}\rangle + e$. So the transformation takes $A_{\mathbf{s}_0,\chi}$ to $A_{\mathbf{x},\chi}$, where $\mathbf{x} = \mathbf{s}_0 + \mathbf{k}$. On the other hand, if the same transformation is applied to samples from $U(\mathbb{Z}_q^n \times \mathbb{Z}_q)$, the output will clearly be a valid sample from $U(\mathbb{Z}_q^n \times \mathbb{Z}_q)$.

We can use this transformation to create the modified games. For the first modified game, $\mathscr{P}$ has access to an oracle to sample $A_{\mathbf{s}_0,\chi}$. It applies the transformation many times to define the secret keys $\mathbf{s}_1, \ldots, \mathbf{s}_u$ and gain the ability to sample $A_{\mathbf{s}_1,\chi}, \ldots, A_{\mathbf{s}_u,\chi}$. $\mathscr{P}$ knows that $\mathbf{s}_l - \mathbf{k}_l = \mathbf{s}_m - \mathbf{k}_m$ for any m, l. Note that $\mathscr{P}$ does not know the actual secrets, but only the relationship between any pairs.

Now we show how $\mathscr{P}$ generates $\hat{\tau}_{\mathbf{s}_l\to\mathbf{s}_m}$, the matrix that will replace $\tau_{\mathbf{s}_l\to\mathbf{s}_m}$ in the first modified game. We shorten $\hat{\tau}_{\mathbf{s}_l\to\mathbf{s}_m}$ to $\hat{\tau}_{lm}$. We index the rows of $\hat{\tau}_{lm}$ with pairs ij. Recall that $\hat{\tau}_{lm}$ has $n\log q$ rows. We can think of $\hat{\tau}_{lm}$ as having n consecutive blocks of rows, and each has $\log q$ rows. i will represent the block, and j will represent the row within the block, e.g., the row identified by ij is row $i\cdot\log q + j$. $\mathscr{P}$ generates row $i\cdot\log q + j$ of $\hat{\tau}_{lm}$ by first sampling some $(\mathbf{a}', b')$ from $A_{\mathbf{s}_m,\chi}$. It adds $2^i(\mathbf{k}_m + \mathbf{k}_l)[j]$ [the ith component of $(-\mathbf{k}_m + \mathbf{k}_l)$] to the first component and subtracts $\mathbf{d}_{ij}$ from the second, where $\mathbf{d}_{ij}$ is the vector that is zero everywhere except in the jth position where it is 2^i. It then negates the second component and switches the ordering of the components. $\mathscr{P}$ generates $\hat{\tau}_{lm}$ by going over all possible values of i, j.

3. Indistinguishability of $\hat{\tau}_{lm}$

We want to show that $\hat{\tau}_{lm}$ is indistinguishable from τ_{lm} from the original game. Let $\mathbf{a} = \mathbf{a}' - \mathbf{d}_{ij}$. We substitute this into a row $i\cdot\log q + j$ of $\hat{\tau}_{lm}$:

$$
\begin{aligned}
&(\langle\mathbf{a}', \mathbf{s}_m\rangle + 2e + 2^i(-\mathbf{k}_m + \mathbf{k}_l)[j], -\mathbf{a}' + \mathbf{d}_{ij}\\
&= (\langle\mathbf{a}, \mathbf{s}_m\rangle + 2e + 2^i(\mathbf{s}_m - \mathbf{k}_m + \mathbf{k}_l)[j], -\mathbf{a})\\
&= (\langle\mathbf{a}, \mathbf{s}_m\rangle + 2e + 2^i\mathbf{s}_l[j], -\mathbf{a})\\
&= (\langle\mathbf{a}, \mathbf{s}_m\rangle + 2e + \texttt{Powersof2}(\mathbf{s}_1[i\cdot\log q + j], -\mathbf{a}).
\end{aligned}
$$

Since $(\mathbf{a}, \langle\mathbf{a}, \mathbf{s}_m\rangle + 2e)$ is a sample from $A_{\mathbf{s}_m,\chi}$, this is of the same form as the corresponding row of τ_{lm}. Therefore $\hat{\tau}_{lm}$ is indistinguishable from τ_{lm}. Thus, the set of matrices generated by $\mathscr{P}$ is indistinguishable from the original key-switching matrices, and the first modified game is indistinguishable from the original game.

The second modified game proceeds exactly as the previous one except $\mathscr{P}$ is given access to an oracle that samples uniform random distribution. Since LWE is hard, it is impossible for $\mathcal{A}$ to distinguish this second modified game from the previous one. We note that all of the matrices $\hat{\tau}_{lm}$ produced this time will be uniformly random. This means that all of the matrices returned by the challenger in the second modified game will be uniform (regardless of whether the challenger decides to return $\hat{\tau}_{lm}$ or a random matrix). Since this game is indistinguishable from the original, we conclude that any polynomial-time adversary cannot distinguish between the real key-switching matrices and uniformly random matrices.

B. Security of XOR Implementation

In the XOR implementation, the relay nodes store the pairwise differences of the secret keys of their trusted neighbors. Without loss of generality, we can consider the case when d trusted endpoints are all connected through one relay node. Thus each of the d trusted endpoints has a secret key, $\mathbf{s}_1, \ldots, \mathbf{s}_d$. After the trusted setup is over, the relay node maintains a list of the pairwise differences of these keys $\mathbf{s}_i - \mathbf{s}_j$ and can thus transform an encryption of the form $(\mathbf{a}, \mathbf{a}\cdot\mathbf{s}_i + \epsilon + m)$ (an encryption of m under key $\mathbf{s}_i$) into an encryption of the form $(\mathbf{a}, \mathbf{a}\cdot\mathbf{s}_j + \epsilon + m)$, an encryption of m under key $\mathbf{s}_j$. We will show that an adversary with bounded computational power and access to transforms $s_i - s_j$ and ciphertexts $\{\mathbf{c}_k\}$ can learn only as much as a polynomially bounded adversary with access to ciphertexts $\{\mathbf{c}_k\}$. In other words, an adversary with bounded computational powers gains no information from the transforms. However, an adversary with unbounded computational powers would be able to break ciphertexts $\{\mathbf{c}_k\}$ with or without the transforms. Therefore, giving access of the transforms and ciphertexts to the relay node during normal operations is no less secure than giving access to just the ciphertexts.

Proof. We show that an adversary $\mathcal{A}_1$ with access to ciphertexts $\{\mathbf{c}_k\}$ and transforms $\{\mathbf{s}_i - \mathbf{s}_j\}_{i,j=1}^{d}$ can gain no more information than an adversary $\mathcal{A}_2$ with access to only ciphertexts $\{\mathbf{c}_k\}$.

First note that we can restrict our attention to the case where the ciphertexts are encryptions using a fixed key, s_1. (Recall that with the transforms, the ciphertexts can easily and quickly be changed into encryptions under any of the other keys.) Second, we may also assume that the transforms are only of the form $\mathbf{s}_i - \mathbf{s}_1$ for $i = 2, \ldots, d$. This follows from the fact that all the other transforms can be generated by these $d-1$ transforms: $\mathbf{s}_i - \mathbf{s}_j = (\mathbf{s}_i - \mathbf{s}_1) - (\mathbf{s}_j - \mathbf{s}_1)$.

Let $\mathcal{A}_2$ generate $d-1$ vectors uniformly at random from the space of secret keys. We denote these $\{\mathbf{t}_i\}_{i=2}^{d}$. It is clear that

$$\mathbf{t}_i = (\mathbf{t}_i + \mathbf{s}_1) - \mathbf{s}_1.$$

Thus the vectors $\{\mathbf{t}_i\}_{i=2}^{d}$ generate all transforms between the secret keys:

$$\{\mathbf{s}_1\} \cup \{\mathbf{t}_1 + \mathbf{s}_1\}_{i=2}^{n}.$$

Now suppose there is some program $\mathscr{P}$ that can be applied with inputs $\{\mathbf{c}_k\}$ and $\{\mathbf{s}_i - \mathbf{s}_1\}_{i=2}^{d}$. Then $\mathcal{A}_2$ can apply this same program $\mathscr{P}$ to the inputs $\{\mathbf{c}_k\}$ and $\{\mathbf{t}_i\}_{i=2}^{d}$. Now $\{\mathbf{t}_i\}_{i=2}^{d}$ is a generating set of transforms between $\mathbf{s}_1$ and $d-1$ uniformly chosen secret keys, while $\{\mathbf{s}_i - \mathbf{s}_1\}_{i=2}^{d}$ is also a generating set of transforms between $\mathbf{s}_1$ and $d-1$ other secret keys; the two sets of inputs are distributionally equivalent. Thus $\mathcal{A}_2$ will learn no more and no less than $\mathcal{A}_1$. □

C. Security of XOR Key Refresh

The key-refresh algorithms (Algorithms 12 and 13) simply mask the original keys and transforms of the XOR method (Algorithms 9–11) with random material. Thus the key refresh is *no less secure* than simply reusing the original keys of the XOR method. We consider two possibilities depending on whether Ray is compromised by an attacker during the refresh or not. If Ray is compromised, then the refresh will not improve the security of Alice and Bob's communication. An attacker, Eve, with control over Ray could apply appropriate transforms to future ciphertexts to get modified ciphertexts that are encrypted under the old set of keys. On the other hand, if Ray is uncompromised, refreshing the keys will make breaking the encryption scheme harder, as an attacker Eve, with access to the ciphertexts, will not know the (random) relation between the respective secret keys.

6. DISCUSSION

QKD is usually thought of as offering perfect, information-theoretic security. It uses properties of the physical world to thwart even computationally unbounded attackers. In this paper, we weaken the attacker model—by relying on (post-quantum) classical cryptography as the glue between QKD links, our methods are safe only from computationally bounded adversaries. What have we gained? We have essentially replaced public key encryption with QKD links and symmetric key encryption. This provokes the question: do our methods allow perfect forward secrecy? This is usually used in the context of public key encryption and refers to the ability of certain systems to use a public keys infrastructure to create symmetric session keys in such a way that even if the public keys are later broken by an adversary, any session keys created prior remain hidden from the adversary. The implementations described in this paper differ in that there are no public keys; nor are there session keys *per se*. If a QKD link is attacked via a side channel attack, or if one of the nodes is successfully attacked, any ciphertexts encrypted with the revealed keys could be easily decrypted by the attacker. However, any ciphertexts encrypted before the latest key-refresh would remain incomprehensible to a computationally bounded attacker. We note that refresh rates rely on the application or use case and are out of scope for this work. We describe an implementation using an academic version of LWE for ease of description. In a real world development, a more efficient and tunable version of LWE would be recommended such as Crystals–Kyber [37], the latest winner in the NIST post-quantum computation for public key encryption [38].

Note that we do not consider problems concerning authentication in this paper. Thus an adversary who took control of Ray could apply inappropriate transforms to the encrypted

messages passed through Ray. In effect, this would be similar to an adversary cutting a wire; it would block communication between Alice and Bob, but would also be detectable. Designing a signature scheme to allow for the correct attribution of such an adversarial transform when there are several relays in series is left for future work. We anticipate a follow-on paper describing our empirical implementation of the concepts in this paper.

Funding. Office of Electricity (M620000160); U.S. Department of Energy (89233218CNA000001).

Acknowledgment. This material is based upon work supported by the U.S. Department of Energy, Office of Electricity Delivery and Energy Reliability, Cybersecurity for Energy Delivery Systems Program. Los Alamos National Laboratory is operated by Triad National Security, LLC, for the National Nuclear Security Administration of U.S. Department of Energy.

REFERENCES

1. B. Schneier, *Applied Cryptography Protocols, Algorithms, and Source Code in C*, 2nd ed. (Wiley, 1996).
2. R. J. Hughes, J. E. Nordholt, K. P. McCabe, R. T. Newell, C. G. Peterson, and R. D. Somma, "Network-centric quantum communications with application to critical infrastructure protection," arXiv, arXiv:1305.0305 (2013).
3. R. Alléaume, C. Branciard, J. Bouda, *et al.*, "Using quantum key distribution for cryptographic purposes: a survey," Theor. Comput. Sci. **560**, 62–81 (2014).
4. C. H. Bennett and G. Brassard, "Quantum cryptography: public key distribution and coin tossing," in *Proceedings of the International Conference on Computers, Systems and Signal Processing* (1984).
5. A. Lamas-Linares and C. Kurtsiefer, "Breaking a quantum key distribution system through a timing side channel," Opt. Express **15**, 9388–9393 (2007).
6. H. Ko, B.-S. Choi, J.-S. Choe, K.-J. Kim, J.-H. Kim, and C. J. Youn, "Critical side channel effects in random bit generation with multiple semiconductor lasers in a polarization-based quantum key distribution system," Opt. Express **25**, 20045–20055 (2017).
7. A. Biswas, A. Banerji, P. Chandravashi, R. Kumar, and R. P. Singh, "Experimental side channel analysis of BB84 QKD source," arXiv, arXiv:2106.10500 (2021).
8. M. Takeoka, S. Guha, and M. M. Wilde, "Fundamental rate-loss tradeoff for optical quantum key distribution," Nat. Commun. **5**, 5235 (2014).
9. R. Qiao and Y. Meng, "Research progress of quantum repeaters," J. Phys. Conf. Ser. **1237**, 052032 (2019).
10. S. Das, M. S. Rahman, and M. Majumdar, "Design of a quantum repeater using quantum circuits and benchmarking its performance on an IBM quantum computer," Quantum Inf. Process. **20**, 245 (2021).
11. C. Elliott, "Building the quantum network," New J. Phys. **4**, 46 (2002).
12. M. Peev, C. Pacher, R. Alléaume, C. Barreiro, J. Bouda, W. Boxleitner, T. Debuisschert, E. Diamanti, M. Dianati, J. Dynes, and A. Zeilinger, "The SECOQC quantum key distribution network in Vienna," New J. Phys. **11**, 075001 (2009).
13. D. Stucki, M. Legre, F. Buntschu, B. Clausen, N. Felber, N. Gisin, L. Henzen, P. Junod, G. Litzistorf, P. Monbaron, and H. Zbinden, "Long-term performance of the SwissQuantum quantum key distribution network in a field environment," New J. Phys. **13**, 123001 (2011).
14. M. Sasaki, M. Fujiwara, H. Ishizuka, W. Klaus, K. Wakui, M. Takeoka, S. Miki, T. Yamashita, Z. Wang, A. Tanaka, and A. Zeilinger, "Field test of quantum key distribution in the Tokyo QKD network," Opt. Express **19**, 10387–10409 (2011).
15. T.-Y. Chen, J. Wang, H. Liang, W.-Y. Liu, Y. Liu, X. Jiang, Y. Wang, X. Wan, W.-Q. Cai, L. Ju, and J. W. Pan, "Metropolitan all-pass and inter-city quantum communication network," Opt. Express **18**, 27217–27225 (2010).
16. S. Wang, W. Chen, Z.-Q. Yin, H.-W. Li, D.-Y. He, Y.-H. Li, Z. Zhou, X.-T. Song, F.-Y. Li, D. Wang, and Z. F. Han, "Field and long-term demonstration of a wide area quantum key distribution network," Opt. Express **22**, 21739–21756 (2014).
17. Y. Mao, B.-X. Wang, C. Zhao, G. Wang, R. Wang, H. Wang, F. Zhou, J. Nie, Q. Chen, Y. Zhao, and J. W. Pan, "Integrating quantum key distribution with classical communications in backbone fiber network," Opt. Express **26**, 6010–6020 (2018).
18. P. G. Evans, M. Alshowkan, D. Earl, D. Mulkey, R. Newell, G. Peterson, C. Safi, J. Tripp, and N. A. Peters, "Trusted node QKD at an electrical utility," arXiv, arXiv:2103.09877 (2021).
19. A. K. Ekert, "Quantum cryptography based on Bell's theorem," Phys. Rev. Lett. **67**, 661–663 (1991).
20. R. J. Hughes, G. L. Morgan, and C. G. Peterson, "Quantum key distribution over a 48 km optical fibre network," J. Mod. Opt. **47**, 533–547 (2000).
21. R. J. Hughes, J. E. Nordholt, G. L. Morgan, and C. G. Peterson, "Free space quantum key distribution over 10 km in daylight and at night," in *Nonlinear Optics: Materials, Fundamentals and Applications* (Optica Publishing Group, 2002), paper FA2.
22. S.-K. Liao, W.-Q. Cai, W.-Y. Liu, *et al.*, "Satellite-to-ground quantum key distribution," Nature **549**, 43–47 (2017).
23. O. Regev, "Lattice-based cryptography," in *Annual International Cryptology Conference* (Springer, 2006), pp. 131–141.
24. O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," J. ACM **56**, 34 (2009).
25. O. Regev, "The learning with errors problem (Invited Survey)," in *IEEE 25th Annual Conference on Computational Complexity* (2010), pp. 191–204.
26. I. T. L. Computer Security Division, NIST Post-Quantum Cryptography (2021).
27. A. Pellet-Mary and D. Stehlé, "On the hardness of the NTRU problem," in *Advances in Cryptology–ASIACRYPT 2021*, M. Tibouchi and H. Wang, eds. (Springer, 2021), pp. 3–35.
28. Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé, "Classical hardness of learning with errors," in *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing* (2013), pp. 575–584.
29. M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *International Conference on the Theory and Applications of Cryptographic Techniques* (Springer, 1998), pp. 127–144.
30. S. S. Chow, J. Weng, Y. Yang, and R. H. Deng, "Efficient unidirectional proxy re-encryption," in *International Conference on Cryptology in Africa* (Springer, 2010), pp. 316–332.
31. M. Green and G. Ateniese, "Identity-based proxy re-encryption," in *International Conference on Applied Cryptography and Network Security* (Springer, 2007), pp. 288–306.
32. R. Canetti and S. Hohenberger, "Chosen-ciphertext secure proxy re-encryption," in *Proceedings of the 14th ACM Conference on Computer and Communications Security* (2007), pp. 185–194.
33. Z. Qin, H. Xiong, S. Wu, and J. Batamuliza, "A survey of proxy re-encryption for secure data sharing in cloud computing," IEEE Trans. Serv. Comput. (to be published).
34. Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," in *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS '11)* (IEEE Computer Society, 2011), pp. 97–106.
35. Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," ACM Trans. Comput. Theory **6**, 13 (2014).
36. D. Micciancio and C. Peikert, "Hardness of SIS and LWE with small parameters," in *Annual Cryptology Conference* (Springer, 2013), pp. 21–39.
37. W. J. Buchanan" Crystals-Kyber (lattice + LWE) - key exchange mechanism (KEX)," 2023, https://asecuritysite.com/encryption/kyber2.
38. Computer Security Division, Information Technology Laboratory, NIST" Selected algorithms 2022 - post-quantum cryptography: CSRC," 2022, https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

150.135.165.71 on 2024-08-28 17:33:10