Auditable Medical Data Sharing through Recoverable Key Agreement

Van Assen, Jorrit; Kromes, Roland; Erkin, Zekeriya

DOI
10.1109/BRAINS63024.2024.10732363

Publication date
2024

Document Version
Final published version

Published in
2024 6th Conference on Blockchain Research and Applications for Innovative Networks and Services, BRAINS 2024

Citation (APA)
Van Assen, J., Kromes, R., & Erkin, Z. (2024). Auditable Medical Data Sharing through Recoverable Key Agreement. In N. Salhab (Ed.), *2024 6th Conference on Blockchain Research and Applications for Innovative Networks and Services, BRAINS 2024* (2024 6th Conference on Blockchain Research and Applications for Innovative Networks and Services, BRAINS 2024). IEEE. https://doi.org/10.1109/BRAINS63024.2024.10732363

2024 6th Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS) | 979-8-3503-6784-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/BRAINS63024.2024.10732363

Auditable Medical Data Sharing through Recoverable Key Agreement

Jorrit van Assen, Roland Kromes, and Zekeriya Erkin
Cyber Security Group, Department of Intelligent Systems
Delft University of Technology
Van Mourik Broekmanweg 5, 2628 XE, Delft, The Netherlands
{j.s.vanassen,r.g.kromes,z.erkin}@tudelft.nl

***Abstract*—Medical research benefits from large quantities of high-quality data. Internet-based data-sharing platforms bring the advantage of rapidly sharing data medical data. However, ensuring security and accountability in networked medical systems remains a challenge. In this paper, we propose a secure and auditable data-sharing platform for hospitals and research groups based on a distributed ledger. A two-party protocol for recoverable key agreement lies at the basis of securing the data sharing. This protocol enables two parties to agree on an encryption key and put the encryption key under the escrow of a board of semi-trusted auditors. A quorum of these auditors is required in order to recover the encryption key. The recoverable key agreement ensures that past communication can be audited, even if one of the two parties is malicious. We provide a realization of the protocol and analyze its complexity and performance. Based on these analyses, we demonstrate that the protocol is suitable for real-world use cases and resource-constrained devices.**

***Index Terms*—Medical Data-Sharing, Key-Escrow, Auditing, Distributed Key Generation**

I. Introduction

Thanks to the (r)evolution of Internet of Things (IoT) devices and, in particular, Wireless Sensor Networks (WSNs), medical records, also known as e-health records (EMRs/EHRs), enable early illness discovery and faster patient recovery [1], [2]. EHRs benefit patients since the practitioner can easily access relevant information and make decisions faster. In addition, machine learning and AI can also trait the records to help practitioners in decision-making [1]. The next step for digitalizing healthcare is finding a way to enable data sharing to further improve decision-making capabilities. According to Siyal et al. [3], exchanging medical data in future healthcare systems is essential because it can benefit hospitals, research institutions, and laboratories by accelerating research and enhancing current patients' medical treatment approaches. Scheibner et al. [4] noted that the biomedical research paradigm has been characterized by a shift from intrainstitutional research toward multiple collaborating institutions operating at an interinstitutional, national, or international level for multisite research projects.

While medical data sharing may significantly improve the quality of our healthcare systems, it further complicates security and privacy issues. According to [5], between 2019 and 2022, an average of 44 million medical records were breached per year in the U.S. alone. This scale of medical record leakage implies both the significant value of medical-related data and a lack of adequate protection of digital infrastructure used in healthcare. In 2017, the American Medical Association reported that an estimated 83% of U.S. physicians experienced some form of cyberattack that year [6]. A cybersecurity attack on Change Healthcare in early 2024 significantly disrupted a large group of medical institutions [7]. In a follow-up article [8], experts warn that, when it comes to cybersecurity and information security, the entire sector is currently severely under-resourced.

Both government and companies are taking steps to improve the digital resilience of healthcare. New systems must be created using security- and privacy-by-design principles to meet mandated standards. Requirements for medical data storage and sharing can be found in both the EU General Data Protection Regulation (GDPR) and the American Health Insurance Portability and Accountability Act (HIPAA). Additionally, to further incentivize the development of secure healthcare technologies, government institutions provide subsidies. For example, the European Commission explores a European Health Data Space that would allow medical data-sharing for short-term treatment of patients and long-term applications for researchers, innovators, policy-makers, and regulators [9]. Maintaining security, privacy, and trust is vital for the data-sharing solution. This is also reflected in the HIPAA, where the regulation requires medical data sharing protocols to support the recording and examining activities on the data [10].

In addition to external forces attacking medical systems, in [11], Geroge and Buyse discuss data fraud in clinical trials. They discuss that although the occurrence of such fraud is often assumed to be low, the true incidence is hard to establish. Nevertheless, there are high-profile cases of clinical data fraud. For example, Retraction Watch reports that 184 papers of Joachim Boldt were retracted on the grounds of scientific misconduct and fabrication of data [12]. Aside from preventive measures like improving awareness amongst scientific staff, the authors in [13] discuss improving national and international investigative structures. Considering both the legal requirements and the scientific need for accountability in research, medical data-sharing platforms provide an opportunity to enhance the transparency of medical research.

There are essential security and privacy requirements for

This paper is supported by the European Union's Horizon Europe research and innovation program under grant agreement No. 101094901, the SEPTON project (Security Protection Tools for Networked Medical Devices).

establishing medical data-sharing solutions for research purposes. For example, only authorized parties, such as medical institutions and researchers, should have access to medical data. The data-sharing method should be reliable, meaning its origin is known, and alterations to the data are either impossible or recorded. Since alterations to medical data in practice occur for various reasons, it is imperative to have auditability, particularly for resolving disputes. Additionally, during data audits, proper privacy protection should be offered to the patients through informed consent and technical safeguards.

Existing work around medical data sharing varies in approach. To create a reliable data-sharing platform, the authors of [3], [14] suggest using *distributed ledger technology* (DLT), namely blockchain [15], which enables authorized parties to share data. Using blockchain protects against alteration since the immutability property guarantees that the data is append-only and that any modification will be detected. However, note that using blockchain does not guarantee security or privacy implicitly, unlike some existing work assumes [16]: the data is transparent and available to the parties involved in the platform.

When it comes to privacy protection, we see approaches that rely on different tools such as secure multi-party computation [17], homomorphic encryption [18], and differential privacy [19]. In [20], the authors describe a cryptographic protocol for performing analytics on sensitive medical datasets distributed over multiple sites using MPC, a cryptographic tool that enables computation with secret inputs. In [4], the authors discuss how secure multi-party computation and homomorphic encryption, another cryptographic tool for manipulating data in the encrypted form, can be used to preserve patient privacy, enabling medical research while maintaining a high standard of data security. In [21], the authors describe securing trained machine-learning models against model extraction attacks using differential privacy and homomorphic encryption. These and many other works provide privacy protection in various scenarios; however, regarding the auditability of data-sharing, work is limited, and the exact definition differs from work to work. The authors in [22] propose a system where data requests, with motivations, are captured using smart contracts. Alternatively, the authors in [23] propose incorporating auditability by creating a verifiable key-sharing system.

In this work, we propose a medical data-sharing platform based on a custom-tailored cryptographic protocol with auditing. More precisely, we consider a network of hospitals and research groups exchanging medical data in encrypted form. To do so, the device of the hospital in question and the device of the corresponding research group uses a cryptographic protocol, namely *recoverable key agreement* (RKA), to establish a shared secret key for securing communication. The data queries and responses are sent over a distributed ledger, allowing the hospital and the research group to send and receive messages and creating an immutable trail of encrypted messages that can be audited later. We achieve auditing guarantees through our novel RKA protocol, which enables two parties to establish shared encryption keys, commits the keys under a specified public key to the DLT, and allows any third party to verify the correct execution of the protocol. The keys are committed under the public key of the board, whose members share the private key using distributed key generation. The board members are semi-trusted third parties that can decrypt the committed keys on the DLT using threshold secret sharing in case of a dispute. Given a quorum of DBMs with sufficient key shares, the quorum can audit any past data transactions without the help of the original sender or receiver. The decrypted keys can be used to audit the messages exchanged between hospitals and research groups. We, thus, provide strong auditability guarantees using our novel RKA protocol, which introduces limited computational overhead.

The contributions presented in this paper are as follows:

- We propose a medical sharing platform based on DLT for hospitals and research institutions to collaborate securely and in a privacy-preserving manner.
- We present a novel *recoverable key agreement* protocol that enables the two executing parties to establish a key amongst themselves and a designated third party, enabling auditing.
- By utilizing a DLT system, we achieve integrity guarantees, preventing alterations to the transcripts.
- We show that our data-sharing protocol and RKA protocol are secure. Additionally, our experimental results clearly show that the RKA protocol can be executed even on mid-range IoT devices: Creating a new session key takes only two messages, two broadcasts, and 27.26 ms of computations on a Raspberry Pi 4.

The rest of the paper is organized as follows: We present related work in Section II, we give the preliminaries necessary for our protocol in Section III. We describe our proposal, threat model, and assumptions in Section IV. We give our proposal's security and complexity analyses and provide experimental results in Section V. We discuss our work in Section VI. Finally, we present our conclusions in Section VII.

II. Related Work

This section discusses previous work that addressed auditable and secure data sharing. Han et al. [24] review works that propose distributed ledger technologies and artificial intelligence for record-keeping in accounting. They discuss the concept of continuous auditing, where blockchain can create a real-time data trail and improve auditing efficiency. They note that businesses will unlikely replace current IT infrastructure with DLT, instead seeing DLT as a technique that will become a component of the IT system. Work that explored the combination of DLT and data-sharing notes that publicly verifiable ledgers can be used to audit data-flows through transactions [25]. Kokoris-Kogias et al. [26] propose a data-management system that achieves accountability, user control, and permissionless functionality using an on-chain secrets abstraction. A board of semi-trusted members is created to verify data transactions. Oh et al. [27] present privacy-preserving auditing in broker-based health information exchanges. Health information exchange systems are used to share patient records between institutions. A broker system was introduced to index and share the data to constrain the amount of sensitive health records that institutions have to hold. However, brokers identify and prevent illegitimate access to medical records. Privacy-preserving audits enable brokers to verify the legitimacy of data access while protecting the privacy of the records. Their solution is based on encrypting audit logs through hierarchical identity-based encryption, where the logs are in a standardized form and verified automatically. The authors note that although the audit subsystem creates trust by enabling automated audits without requiring full access to unencrypted log data, their solution cannot provide trust when human judgment is required.

Few works provide unencrypted auditing of data exchanges. Reniers et al. [23] propose a protocol for incorporating auditability and traceability of actions in file-sharing between organizations. The protocol is tailored towards multi-disciplinary optimization processes, which requires the collaboration of institutions with different expertise. Their solution removes the need for a third party to verify all steps in the data-sharing process and act as a witness in case of disputes. The system achieves the so-called auditability of data operations. This means that any data access operation, read or write, can be checked by anyone, including external auditors. In [28], Cha et al. use key escrow encryption and blockchain to prevent fraudulent behavior in the supply chain. They propose dividing the escrow key system over a group of agents, preventing denial-of-service attacks, and relying on a single party for security. To secure communication between two parties, the sending party chooses a session key that is uniformly random over the key space. The session key is encrypted twice, using the public key of the receiver and using the public key of the key escrow agent group. The ciphertexts are uploaded with a zero-knowledge proof to the ledger, proving that both ciphertexts encrypt the same session key. However, the details of the zero-knowledge proof are not discussed, which indirectly creates doubts about the verifiability of the session key, meaning that it is not ensured that the escrow group and the recipient obtain the same session key after the public key is decrypted.

III. Preliminaries

In this section, we discuss the preliminaries of our protocol. Our protocol combines basic data-sharing concepts in distributed systems with more advanced cryptographic protocols for establishing and sharing keys. Therefore, we give primary attention to building blocks related to key management. We first discuss the concept of key encapsulation mechanisms and our seeded variant. Later, we discuss secret sharing amongst a group of participants, distributed key generation, and threshold decryption.

A. Key Encapsulation Mechanisms

A *key encapsulation mechanism* (KEM) is a method for encapsulating a symmetrical key in a ciphertext using a public key encryption system. Since symmetric encryption is significantly more efficient than public key encryption, KEMs enable efficient communication when keys have not been pre-shared. A KEM consists of three separate steps: 1. Generation, 2. Encapsulation, and 3. Decapsulation. In the first step, the receiver generates a private key and computes a public key. In the encapsulation step, a sending party uses a public key to create a ciphertext and a symmetrical key. In the decapsulation phase, the private key of the receiver is used to retrieve the symmetrical key from the ciphertext. Note that it is not possible to specify the desired symmetric key beforehand. Instead, during encapsulation, a symmetrical key is generated stochastically. For ECIES [29], PSEC [30], FrodoKEM [31], and Kyber [32], this non-determinism is created by using a random value sampled during encapsulation. We introduce a variant on the KEM, the seeded KEM (SKEM), where the KEM also takes a seed as input in addition to a public key. The additional input transforms the stochastic KEM into a deterministic SKEM by seeding all random samples using a predefined seed τ. The advantage of the SKEM is that the two parties can verify that a ciphertext will decrypt to an agreed value, even if the corresponding secret key is unknown. In Algorithm 1, we give an algorithm for the encapsulation based on ECIES encapsulation. Unlike ECIES, we add parameter τ and use this as the ephemeral secret key for the DH key agreement instead of sampling a random integer. The ECIES decapsulation can be used as a decapsulation method without any adjustments.

Algorithm 1 SKEM encapsulation of a symmetric key based on the ECIES KEM scheme.

```
Input: $(\mathsf{pk}_B, \tau)$
Output: $(\mathsf{k}, c_\mathsf{k})$
$(P_x, P_y) \leftarrow \tau \cdot pk_B$
$\mathsf{k} \leftarrow P_x$
$c_\mathsf{k} \leftarrow \tau \cdot G$
return $(\mathsf{k}, c_\mathsf{k})$
```

B. Authenticated Key Exchange

Authenticated key exchange (AKE) is a key establishment technique in which both parties influence the resulting shared secret, and the identities of both parties are confirmed. The fact that both parties can influence the outcome of the shared secret guarantees the freshness of the established secret. This prevents replay attacks, in which an adversary reruns a previous transcript to break the security of the key exchange. On the other hand, the fact that the participants' identities are confirmed prevents adversaries from executing man-in-the-middle attacks.

AKE protocols are widespread on the internet. For example, the *transport layer security* protocol encrypts HTTPS communication between clients and websites. Many classic authenticated key exchange protocols are based on the *Diffie-Hellman* (DH) key exchange. For this paper, we explicitly look at HMQV [33], a DH-based key exchange proposed by Krawczyk. This protocol modifies the MQV key exchange by Menezes et al. [34] which allows the protocol to be formally proven. The two-pass HMQV variant is proven to be CK-secure under the *computational Diffie-Hellman* (CDH) assumption, with implicit authentication and weak forward secrecy. The concept of weak forward secrecy was introduced by Bellare et al. [35], as losing long-lived secret keys does not compromise previously established session keys as long as the adversary did not take an active part in that session. Although for specific parameters of the two-pass HMQV variant, there exists a *key compromise impersonation* (KCI) attack, it can be mitigated using a group membership test [36].

C. Distributed Key Generation

Distributed key generation (DKG) allows a group of participants to collaboratively compute a public key, whose corresponding private key is divided amongst the participants in secret shares. A threshold of participants is required to recover the private keys. DKG does not rely on a trusted third party but instead on verifying commitments to remain secure in the presence of an adversary. The first DKG protocol was proposed by Pedersen in [37]. In later work by Gennaro et al., the Pedersen DKG was found to be insecure. An adversary can manipulate the distribution of the private key so that it is no longer distributed uniformly. The same authors proposed a new DKG scheme that is proven secure against this attack and can replace Pedersen DKG for discrete-logarithm-based cryptosystems [38]. In later work by Aniket Kate and Ian Goldberg [39], a modification of the DKG protocol was proposed, which can run in an asynchronous communication model, making DKG suitable for use over the internet. Work by Schindler et al. [40] builds on this by proposing a DKG that can be executed on Ethereum.

IV. Our Proposal

Now, we describe our medical data-sharing platform that enables auditability. We introduce our setting, briefly discuss our security assumptions and privacy requirements, and explain our protocol.

A. Setting

We define four stakeholders in our platform:

- **Hospitals**, denoted by $\mathcal{H}$, are the key stakeholder that uploads the necessary data to the data-sharing platform. Each $\mathcal{H}$ has a unique identifier and engages with the research groups to process incoming queries and securely transmit the results. Hospitals have permission to read and append to the distributed ledger.
- **Research Groups**, denoted by $\mathcal{R}$, are the stakeholders interested in researching medical data provided by the hospitals. We assume that the authorities authenticate these research groups; we omit this authentication step for the sake of simplicity in this work. Each research group also has a unique identifier. Research groups have permission to read the distributed ledger's data and append the query they requested to the ledger.
- **Board Members**, denoted by $\mathcal{B}m$, are independent stakeholders responsible for resolving disputes, as we introduced before. To protect the shared data against corrupted board members, a predefined threshold of board members, t, is required in order to access data on the distributed ledger.
- **Data-sharing platform** is essentially a distributed ledger to store the data used by the research groups. The realization of this ledger can vary: a cloud system or a private ledger run by hospitals or board members are all possible.

In our setting, the data stored on the ledger is encrypted for privacy reasons. The corresponding hospital and the research group initiate an RKA protocol that outputs a symmetric key. The hospital then encrypts the medical data using this symmetric key and uploads the ciphertext to the ledger. Then, the research group can download and decrypt the medical data. The data format is unspecified in this protocol since any can be used.

The novelty of our key agreement protocol is that it enables the board members to have key shares, which can be used later to decrypt data in case of a dispute. Hence, our protocol is a *recoverable key agreement* protocol.

B. Security and Privacy Requirements

We assume that there is PKI available. This implies that authorities authenticate the hospitals and researchers. We believe that only authorized hospitals and authorized research groups can upload data. We require that in case the secret key of a stakeholder is leaked, previously encrypted data on the ledger cannot be accessed, implying that we have forward secrecy. As the data is stored as ciphertexts on the DLT, external parties that might obtain the data stored on the ledger cannot access the medical data. However, we enable only a representative quorum of the board members to access the relevant medical data in case of a dispute, which a juridical process should initiate. We assume the hospitals and the research groups do not expose their symmetric keys. Furthermore, internal attackers, such as malicious hospitals or research groups, cannot alter the data on the ledger.

C. Protocol Description

Our protocol consists of three parts: (1) Setup, (2) Medical Data Research, and (3) Dispute Resolution. For the sake of simplicity, we describe our protocol for a single hospital ($\mathcal{H}$) and a research group ($\mathcal{R}$).

Setup: Recall that we assume a PKI is available. This means that $\mathcal{H}$ has a signature key-pair $(\mathsf{spk}_H, \mathsf{ssk}_H)$, encryption key-pair $(\mathsf{epk}_H, \mathsf{esk}_H)$ and a unique identifier ID_H. Likewise, $\mathcal{R}$ has $(\mathsf{spk}_R, \mathsf{ssk}_R)$ for signing, $(\mathsf{epk}_R, \mathsf{esk}_R)$ for encrypting and the unique identifier ID_R. Any stakeholder can look up public keys and identifiers using the PKI. The board members perform distributed key sharing, as we explain in Section III. After the distributed key agreement is finalized, each board member possesses the board's public key pk_B and a secret share ss_{B_i} of the secret key sk_B. The board public key, pk_B, is made available in the PKI.

Medical Data Research: In this part, $\mathcal{H}$ and $\mathcal{R}$ collaborate in a medical data sharing session. At first, the session is started by $\mathcal{R}$ that initiates the RKA protocol for creating the symmetric keys. Afterward, the symmetric keys are used to securely encrypt data transfers between $\mathcal{R}$ and $\mathcal{H}$ over a distributed ledger through queries and responses.

Algorithm 2 shows the RKA protocol. $\mathcal{H}$ uses as the input (pk_B, epk_H, esk_H, ID_R, epk_R) and $\mathcal{R}$ uses (pk_B, epk_H, ID_H, epk_R, esk_R). The outputs of our protocol are two symmetric keys: qk and rk. The symmetric key qk is used for encrypting the queries created by $\mathcal{R}$, and rk is used for encrypting responses by $\mathcal{H}$. The purpose of using separate keys is that the board can investigate the queries and responses independently in case of disputes. These keys are used for the duration of the data-sharing session.

Our protocol consists of the following steps:

1) Authenticated key establishment between $\mathcal{H}$ and $\mathcal{R}$;
2) Seeded key encapsulation;
3) Signing of the encapsulated key;
4) Publication of the encapsulated key and signatures;
5) Verification of the signatures.

Step 1: Authenticated key exchange. The first step of the protocol is to securely establish a shared secret, T, between $\mathcal{H}$ and $\mathcal{R}$. $\mathcal{R}$ starts the RKA protocol by initiating a key exchange with $\mathcal{H}$. A suitable authenticated key exchange protocol, in short AKE, provides at the least key freshness and implicit authentication. Note that since the shared keys $(\mathsf{qk}, \mathsf{rk})$ between $\mathcal{H}$ and $\mathcal{R}$ are *confirmed* in Step 5, we can achieve explicit key authentication even if we use a key exchange protocol which only provides implicit authentication. For our realization, we use the 2-pass HMQV key exchange variant. This HMQV variant ensures key freshness, is implicitly authenticated, and provides weak forward secrecy as described in Section III. The key exchange protocol is invoked with the long-term encryption key pairs of $\mathcal{H}$ and $\mathcal{R}$, $(\mathsf{esk}_H, \mathsf{epk}_H)$ and $(\mathsf{esk}_R, \mathsf{epk}_R)$, as well as their unique identifiers, ID_H and ID_R. After invoking the key exchange protocol, we convert the shared secret, T, into two separate bit-strings of length 256, (τ_1, τ_2), using $\texttt{KDF}$, a key derivation function.

Step 2: Seeded key encapsulation. In this step, we obtain the symmetric keys, qk and rk, and the encryptions of the symmetric keys, c_{qk} and c_{rk}, using boards public key, pk_B, and (τ_1,τ_2) generated in the previous step. To ensure that both $\mathcal{H}$ and $\mathcal{R}$ end up with the same symmetric key, we use a SKEM instead of a normal KEM. A KEM is a non-deterministic function that outputs a symmetrical key and corresponding ciphertext, which encrypts the symmetrical key using a public key as input. The seeded KEM returns the symmetrical key and corresponding ciphertext deterministically using a seed as input. In Section III, we discuss how KEMs can be turned into SKEMs and give an example construction based on ECIES. Additionally, we limit the choice of KEM schemes to KEM, whose private/public key-pair can be generated using a distributed key generation scheme. $\mathcal{H}$ and $\mathcal{R}$ perform the SKEM twice, using both bit-string τ_1 and τ_2, to create two distinct pairs of a symmetric key and encapsulation, $(c_{\mathsf{qk}}, \mathsf{qk})$ and $(c_{\mathsf{rk}}, \mathsf{rk})$ respectively.

Step 3: Signing of the encapsulated key. This step authenticates $\mathcal{H}$ and $\mathcal{R}$ to the board members by tying the encapsulated key to their identities using their respective private key. $\mathcal{H}$ signs encapsulated key, c_{qk}, with its long-term signature private key, ssk_H, and creates the signature σ_{qk}. Similarly, $\mathcal{R}$ signs encapsulated key, c_{rk}, with its long-term signature private key, ssk_R, and creates the signature σ_{rk}. Note that the encapsulated key is signed instead of the symmetric key, allowing an external party to verify the correctness of the signature. In our realization, we generate the signatures using the ECDSA signature scheme.

Step 4: Publication of the encapsulated key and signatures. The encapsulated keys, c_{qk} and c_{rk}, and signatures, σ_{qk} and σ_{rk}, are appended to the public ledger by $\mathcal{H}$ and $\mathcal{R}$. The public ledger means that the board members are guaranteed access to the original encapsulated keys and signatures. Additionally, this step provides key confirmation and thus ensures that the key establishment is explicitly authenticated. To reduce the message size, $\mathcal{H}$ uploads the encapsulated query key, c_{qk}, and $\mathcal{R}$ uploads the encapsulated response key, c_{rk}.

Step 5: Verification of the signatures. The final step is to validate the validity of the generated symmetric keys, qk and rk. $\mathcal{H}$ and $\mathcal{R}$ do so by verifying the other's generated signature and encapsulated keys. Additionally, note that any stakeholder accessing the distributed ledger can verify the correctness of the RKA. Any discrepancies with the signature or encapsulated key indicate that the agreement was compromised. In the case that either $\mathcal{H}$ or $\mathcal{R}$ detect a discrepancy, a special *block* is appended to the distributed ledger, indicating that the key is invalid and further communication is aborted.

After the symmetric keys have been successfully established, $\mathcal{H}$ and $\mathcal{R}$ can engage in data sharing. $\mathcal{R}$ creates a query, encrypts this using the symmetrical key, and appends the query to the distributed ledger. $\mathcal{H}$ watches the distributed ledger for a data query and executes the query on its medical data. After the results have been computed, the corresponding results are appended to the distributed ledger.

Dispute Resolution: In this part, we describe how a representative quorum of $\mathcal{B}m$'s can recover the symmetric key used to secure the communication between $\mathcal{H}$ and $\mathcal{R}$. The ciphertext containing the symmetric key used for communication can be decrypted using partial decryption, such that the Board secret key, sk_B, remains unknown to any board member.

The first step of the dispute resolution is for a quorum to elect a leader. This leader receives the decrypted query or response key at the end of the decryption. Each board member uses its own private key ss_{BM_i} to create a partial decryption of the ciphertext c. The partial decryptions are securely transmitted

Algorithm 2 Recoverable key agreement between $\mathcal{H}$ and $\mathcal{R}$.

```
$\mathcal{H}$: Research Group                                   $\mathcal{R}$: Hospital
Input                                                           Input
$(\mathsf{pk}_B, \mathsf{epk}_H, \mathsf{spk}_H, \mathsf{esk}_R, \mathsf{ssk}_R, ID_R, ID_H)$      $(\mathsf{pk}_B, \mathsf{epk}_R, \mathsf{spk}_R, \mathsf{esk}_H, \mathsf{ssk}_H, ID_H, ID_R)$
Output                                                          Output
$(\mathsf{qk}, \mathsf{rk})$                                    $(\mathsf{qk}, \mathsf{rk})$

1 : $Z \leftarrow \mathrm{AKE}(ID_R, \mathsf{esk}_R, ID_H, \mathsf{epk}_H)$    $\xleftrightarrow{\boxed{\mathrm{AKE}}}$    $Z' \leftarrow \mathrm{AKE}(ID_H, \mathsf{esk}_H, ID_R, \mathsf{epk}_R)$
    $(\tau_1, \tau_2) \leftarrow \mathrm{KDF}(Z)$                                    $(\tau_1', \tau_2') \leftarrow \mathrm{KDF}(Z')$
.....................................................................................
2 : $(c_{\mathsf{qk}}, \mathsf{qk}) \leftarrow \mathrm{SKEM.Encaps}(\mathsf{pk}_B, \tau_1)$      $(c_{\mathsf{qk}'}, \mathsf{qk}') \leftarrow \mathrm{SKEM.Encaps}(\mathsf{pk}_B, \tau_1')$
    $(c_{\mathsf{rk}}, \mathsf{rk}) \leftarrow \mathrm{SKEM.Encaps}(\mathsf{pk}_B, \tau_2)$      $(c_{\mathsf{rk}'}, \mathsf{rk}') \leftarrow \mathrm{SKEM.Encaps}(\mathsf{pk}_B, \tau_2')$
.....................................................................................
3 : $\sigma_{\mathsf{qk}} \leftarrow \mathsf{Sign}(\mathsf{ssk}_R, c_{\mathsf{qk}})$                          $\sigma_{\mathsf{rk}'} \leftarrow \mathsf{Sign}(\mathsf{ssk}_H, c_{\mathsf{rk}'})$
.....................................................................................
4 : Publicize $((c_{\mathsf{qk}}, c_{\mathsf{rk}}, \sigma_{\mathsf{rk}}))$    $\xrightarrow{(c_{\mathsf{qk}}, c_{\mathsf{rk}}, \sigma_{\mathsf{rk}})}$
                                       $\xleftarrow{(c_{\mathsf{qk}'}, c_{\mathsf{rk}'}, \sigma_{\mathsf{qk}'})}$    Publicize $((c_{\mathsf{qk}'}, c_{\mathsf{rk}'}, \sigma_{\mathsf{qk}'}))$
.....................................................................................
5 : if $(c_{\mathsf{qk}} = c_{\mathsf{qk}'}) \wedge \mathsf{Verify}(\mathsf{spk}_H, \sigma_{\mathsf{qk}'}, c_{\mathsf{qk}})$ then      if $(c_{\mathsf{rk}} = c_{\mathsf{rk}'}) \wedge \mathsf{Verify}(\mathsf{spk}_R, \sigma_{\mathsf{rk}}, c_{\mathsf{rk}'})$ then
        return $(\mathsf{qk}, \mathsf{rk})$                                             return $(\mathsf{qk}', \mathsf{rk}')$
    else                                                                else
        Publicize(REJECT)                                               Publicize(REJECT)
        return $\varnothing$                                            return $\varnothing$
```

to the leader of the quorum. The key k can be recovered by combining the partial decryptions.

After the key k has been recovered, the quorum leader can decrypt and audit all query or response blocks broadcasted on the public bulletin board and encrypted with this k. The Board secret key, sk_B, corresponding with the board's public key, pk_B, is never calculated during auditing. This prevents an adversary from abusing the audit process to decrypt communication trails outside the audited trail. Note also that it is possible to audit data sharing in real-time. The moment that $\mathcal{H}$ and $\mathcal{R}$ publicize the RKA results, a quorum could execute the dispute resolution to obtain the keys used for securing data and eavesdropping on the communication between $\mathcal{H}$ and $\mathcal{R}$. Depending on the extent of control the board members have over the network, messages may be intercepted and blocked.

V. Analyses

In this section, we analyze the complexity, security, and performance of our proposed RKA protocol since this is our main contribution. The analysis of the data-sharing platform is out of scope as suitable DLT constructions can be deployed to upload data.

A. Complexity Analysis

We analyze the computation and communication complexity of our RKA protocol. We base the complexity of the protocol on our proposed realization, as described in Section IV. For this realization, we use HMQV as the key agreement scheme, ECDSA as the signature scheme, and the SKEM variant of ECIES Encapsulation, described in Section III, as the SKEM Encapsulation. Since all these schemes are elliptic curve-based, we describe the complexity in terms of elliptic curve multiplications. We express the message complexity using the number of direct, D, and broadcast messages, B. The worst-case complexity of this realization of the RKA for a single party is 12 multiplications, as shown in Table I. For publicizing, we assume ECDSA is used to generate a signature of the message sent. Therefore, an additional two curve multiplications are added. Additionally, note that apart from a single multiplication in the verification of the symmetric key signature, all elliptic curve multiplications are between a scalar and G or pk_B. The computational cost of the multiplications could be reduced by constructing pre-computation tables for G and pk_B.

B. Security Analysis

As we use encryption to secure data on the medical data-sharing platform, an adversary can only access raw queries and

TABLE I
WORST CASE COMPUTATIONAL AND MESSAGE COMPLEXITY OF THE RKA SCHEME FOR A SINGLE PARTY, EXPRESSED IN ELLIPTIC CURVE MULTIPLICATIONS AND MESSAGES AND BROADCASTS.

Steps	1st	2nd	3rd	4th	5th	Total
Ops.	AKE	SKEM	Sign	Sign	Verify + Sign	
EC mult.	3	4	1	1	3	12
Msg.	$1 \cdot D$			$1 \cdot B$	$1 \cdot B$	$D + 2B$

responses recorded on the DLT if they know the encryption key. Therefore, the security of the overall platform depends on the security of the recoverable key establishment protocol. We provide an informal sketch for the security analysis of the RKA protocol. Assuming that the underlying cryptographic primitives, namely the hash function, signature algorithm, and AKE, are secure, our RKA protocol is also secure. We note that a hospital and a research group engaging in our RKA protocol are assumed not to collude.

An adversary $\mathcal{A}$, will receive public inputs $\vec{pk} = [\mathsf{epk}_{\mathcal{H}}$, $\mathsf{epk}\mathcal{R}$, $\mathsf{pk}_B]$, and public outputs the signatures, $\vec{\sigma} = [\sigma_{\mathsf{qk}}, \sigma_{\mathsf{rk}}]$, the ciphertexts, $\vec{c} = [c_{\mathsf{qk}}, c_{\mathsf{rk}}]$, and the AKE transcript, $\mathcal{T}$. We argue the RKA is secure through contradiction.

Assume $\mathcal{A}$ can recover query key qk from the public in- and outputs in polynomial time. Since qk is derived using the SKEM, the adversary either derived qk from the ciphertext, c_qk, or the AKE transcript, T. If the query key qk was derived from c_qk, the adversary would have an efficient way of computing $\tau_1 \leftarrow c_qk \cdot G^{-1}$ or $\mathsf{sk}_B \leftarrow \mathsf{pk}_B \cdot G^{-1}$ to find $qk \leftarrow \mathsf{sk}_B \cdot c_{\mathsf{qk}} = \tau_1 \cdot \mathsf{pk}_B$. This would mean that $\mathcal{A}$ could efficiently compute the discrete logarithm in EC, which contradicts the ECDLP hardness assumption. The second option, deriving qk from the AKE transcript T, the $\mathcal{A}$ would require breaking the security of the AKE protocol. Both cases contradict widely established hardness assumptions and, therefore, we deem our RKA secure.

We argue that the ciphertext, c_{qk}, does not provide any additional information that could be used to compute the shared secret Z. This is because Z is not directly used as a seed for the SKEM but instead only as input for the KDF. Because KDF can securely expand the shared secret Z, there is no correlation between the query key, qk, and the response key, rk. Additionally, because of the forward secrecy property, the query and result keys remain secure even if the long-term secret key of either party is compromised. This means that the $\mathcal{A}$ would be unable to recover km from the transcript and secret key.

C. Performance Analysis

The main objective of our experiments is to highlight the execution time of our proposed RKA in a traditional computer execution environment and an IoT environment. We implement our RKA protocol in Golang. Our protocol contains multiple cryptographic building blocks like HMQV, SKEM, and ECDSA. Since the Golang standard libraries do not include HMQV, we propose an open-source implementation of this protocol in Golang.

The cryptographic building block implementation is based on the "crypto/elliptic" standard Golang library with the NIST-recommended P-256 elliptic curve. Applying a NIST-recommended curve when using the "crypto/elliptic" library is essential since not all custom curves provide the same security properties. In the RKA, each party publicizes two messages per invocation. Although providing the exact workings of publicizing the DLT depends on the specific technology, DLT messages are signed in most cases. Given that signing has a non-negligible computation cost for IoT devices, the implementation emulates this cost by including a signature in steps 4 and 5 of the RKA. The HMQV building block includes the standard SHA-256 cryptographic hash function from the Golang standard library, and the standard ECDSA library is used to perform the digital signatures. For the experiment, the RKA protocol implementation is subdivided into blocks, which are run using the built-in Golang benchmark testing tool. The full implementation of the SKEM, HMQV, and RKA protocols, along with the benchmark, can be found online[1].

We use a general-purpose computer equipped with a 13th Gen Intel(R) Core(TM) i7-1365U @ 5.2GHz with 16GB RAM running GNU/Linux to measure the execution time of our proposed protocol for the traditional computer execution environment. Additionally, we use a Raspberry Pi 4 equipped with a Broadcom BCM2711, Quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8GHz and 4GB of RAM, and running Gnu/Linux to emulate a mid-range IoT device.

Table II shows the average execution time of the RKA on the different machines for a single participant. Each step of the protocol, the setup, and the complete protocol is repeated 10.000 times using the Go testing library. The benchmark is repeated five times and then averaged. The 95% confidence interval is calculated over these five runs. Note that the `Setup` generates the static key, also called long-term key pair generation. The generation of a long-term key pair is not executed frequently, which implies that this step is not considered while calculating the total execution time. Since the 3rd and 4th steps both only consist of signing, they are combined in the results.

The execution time results show that the two most expensive steps are HMQV and EC-KEM. Our protocol shows a relatively fast overall execution time in the PC execution environment (284.5μs), making it suitable for time-sensitive applications. For the IoT environment, the results demonstrate an overall execution time of 27.62 ms. The results also imply that our protocol is suitable for mid-range IoT devices such as Raspberry Pi boards, gateways, and smartphones.

VI. DISCUSSION

In this section, we discuss the merits of our proposal along with its limitations. Recall that our goal is to create an auditable medical data-sharing platform. To achieve this goal, we propose an RKA system that outputs an encryption key shared between a hospital and a research group. In case of a dispute, the

[1]https://github.com/septoneu/ardka

TABLE II
THE AVERAGE TIME OF OUR PROPOSED SCHEME ON A PC AND MID-RANGE IoT DEVICE OVER 10,000 SAMPLES ON FIVE DISTINCT RUNS. THE PERCENTAGE SHOWS A 95% CONFIDENCE INTERVAL.

Steps	Setup	1st	2nd	3rd/4th	5th	Total
Ops.	KeyGen	AKE	SKEM	Sign	Verify	
PC	$19.88\mu s \pm 11\%$	$97.73\mu s \pm 9\%$	$92.38\mu s \pm 5\%$	$16.81\mu s \pm 2\%$	$66.09\mu s \pm 1\%$	$284.5\mu s \pm 4\%$
IoT	$2.191ms \pm 0\%$	$10.41ms \pm 0\%$	$10.09ms \pm 0\%$	$1.429ms \pm 0\%$	$6.334ms \pm 0\%$	$27.26ms \pm 0\%$

number of board members should exceed the threshold before the encryption keys can be recovered using the RKA protocol. As long as the RKA protocol is executed correctly between a hospital and the research group, we argue that no other hospital or research group can access the data.

Furthermore, a malicious hospital or a research group cannot alter data that was publicized on the DLT due to the underlying immutability property of the DLT. Moreover, we also rely on signature schemes in our RKA construction, meaning that non-repudiation is attained. As we also provide a security argument for the RKA we proposed in Section V, we achieve our goals of a secure medical data-sharing platform among hospitals and research groups with auditability.

Although we have described the participants of the data-sharing protocol as being hospitals and research groups, these groups could use any kind of middle-range and high-end IoT device according to the given use case. Allowing auditable data exchanges between networked medical devices and designated third parties would provide new opportunities for directly collecting research information from patients' homes. Although implantable devices are likely too computationally constrained, less expensive gateways based on middle-range or high-end IoT devices could function as data uploaders at the patients' homes.

Notice that our RKA protocol can also be used in other domains, meaning that it is a generic construction that is usable beyond the medical domain. Finance, smart grid data, and cyber threat intelligence are some examples of where disputes might occur. It is possible to deploy our proposal for verifiable research in general and even for scientific publication systems where anonymous reviewers are vital.

Without a doubt, when we deploy our proposal in practice, some parameters need to be well-thought-out, such as the periodicity of invoking RKA protocol between a particular hospital and a research group. We advise keeping the interval short since only a limited amount of data encrypted with a specific key will be available to the board members. The short intervals are advised since our protocol is lightweight. However, the cost of publicizing on the DLT can be a limiting factor.

An important point is the construction of the board: We assume that the board is constructed out of one independent entity. Clearly, in cases where there is only one board member or where members are under the influence of another entity, such as a corrupt government or commercial parties, our proposal cannot guarantee that the encrypted medical research data available on the DLT cannot be accessed beyond our intention. Hence, the number of board members required for reconstructing the key should be chosen carefully.

VII. CONCLUSIONS

In this paper, we proposed a secure and auditable medical data-sharing platform. Auditing is achieved using a board of semi-trusted members that share a key and a distributed ledger to track data transactions. A hospital and a research group pair that wish to share data first engage in a key agreement protocol to establish encryption keys. A board of semi-trusted members can audit the data transactions if a quorum is established. We provided a security sketch, demonstrating that our protocol is secure. Additionally, we have provided a realization of the RKA protocol and verified its performance experimentally in both a PC and mid-range IoT execution environment. Lastly, we described how parameters should be chosen to deploy the system in a natural hospital setting.

REFERENCES

[1] W. Li, Y. Chai, F. Khan, S. R. U. Jan, S. Verma, V. G. Menon, Kavita, and X. Li, "A comprehensive survey on machine learning-based big data analytics for iot-enabled smart healthcare system," *Mob. Networks Appl.*, vol. 26, no. 1, pp. 234–252, 2021. [Online]. Available: https://doi.org/10.1007/s11036-020-01700-6

[2] H. H. Nguyen, F. Mirza, M. A. Naeem, and M. Nguyen, "A review on iot healthcare monitoring applications and a vision for transforming sensor data into real-time clinical feedback," in *Proc. IEEE 12st Int. Conf. Comput. Supported Cooperat. Work Design (CSCWD)*, W. Shen, P. Antunes, N. H. Thuan, J. A. Barthès, J. Luo, and J. Yong, Eds. IEEE, 2017, pp. 257–262. [Online]. Available: https://doi.org/10.1109/CSCWD.2017.8066704

[3] A. A. Siyal, A. Z. Junejo, M. Zawish, K. Ahmed, A. Khalil, and G. Soursou, "Applications of blockchain technology in medicine and healthcare: Challenges and future perspectives," *Cryptogr.*, vol. 3, no. 1, p. 3, 2019. [Online]. Available: https://doi.org/10.3390/cryptography3010003

[4] J. Scheibner, J. L. Raisaro, J. R. Troncoso-Pastoriza, M. Ienca, J. Fellay, E. Vayena, and J. Hubaux, "Revolutionizing medical data sharing using advanced privacy enhancing technologies: Technical, legal and ethical synthesis," *J. Med. Internet Res.*, vol. 23, no. 2, p. e25120, 2021.

[5] A. Petrosyan, "Number of healthcare data records breached in the united states from 2009 to 2022 (in millions)," Nov 2023. [Online]. Available: https://www.statista.com/statistics/798564/number-of-us-residents-affected-by-data-breaches/

[6] American Medical Association, "Medical cybersecurity: A patient safety issue," https://www.ama-assn.org/delivering-care/patient-support-advocacy/medical-cybersecurity-patient-safety-issue, 2017, [Accessed 04-04-2024].

[7] R. Abelson and J. Creswell, "Cyberattack paralyzes the largest u.s. health care payment system," https://www.nytimes.com/2024/03/05/health/cyberattack-healthcare-cash.html, March 5, 2024, [Accessed 04-04-2024].

[8] R. Abelson and M. Sanger-Katz, "4 things you need to know about health care cyberattacks," https://www.nytimes.com/2024/03/29/health/cyber-attack-unitedhealth-hospital-patients.html, March 29, 2024, [Accessed 04-04-2024].

[9] Directorate-General for Health and Food Safety, *Impact Assessment on the European Health Data Space*. Brussel, Belgium: The European Commission, May 2022. [Online]. Available: https://health.ec.europa.eu/publications/impact-assessment-european-health-data-space_en

[10] H. Jin, Y. Luo, P. Li, and J. Mathew, "A review of secure and privacy-preserving medical data sharing," *IEEE Access*, vol. 7, pp. 61 656–61 669, 2019. [Online]. Available: https://doi.org/10.1109/ACCESS.2019.2916503

[11] S. L. George and M. Buyse, "Data fraud in clinical trials," *Clin Investig (Lond)*, vol. 5, no. 2, pp. 161–173, 2015.

[12] A. Marcus, "The new retraction record holder is a German anesthesiologist, with 184," https://retractionwatch.com/2023/07/12/the-new-retraction-record-holder-is-a-german-anesthesiologist-with-184/, July 12, 2023, [Accessed 03-04-2024].

[13] E. Cogan, "Preventing fraud in biomedical research," *Frontiers in cardiovascular medicine*, vol. 9, p. 932138, 2022.

[14] I. Yaqoob, K. Salah, R. Jayaraman, and Y. Al-Hammadi, "Blockchain for healthcare data management: opportunities, challenges, and future recommendations," *Neural Comput. Appl.*, vol. 34, no. 14, pp. 11 475–11 490, 2022. [Online]. Available: https://doi.org/10.1007/s00521-020-05519-w

[15] K. Wüst and A. Gervais, "Do you Need a Blockchain?" in *Proc. Crypto Valley Conf. Blockchain Tech. (CVCBT)*, Zug, Switzerland, 2018, pp. 45–54. [Online]. Available: https://doi.org/10.1109/CVCBT.2018.00011

[16] T.-T. Kuo, H.-E. Kim, and L. Ohno-Machado, "Blockchain distributed ledger technologies for biomedical and health care applications," *J. Amer. Med. Informat. Assoc.*, vol. 24, no. 6, pp. 1211–1220, 2017.

[17] Y. Lindell, "Secure multiparty computation," *Commun. ACM*, vol. 64, no. 1, p. 86–96, dec 2020. [Online]. Available: https://doi.org/10.1145/3387108

[18] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Comput. Surv.*, vol. 51, no. 4, pp. 79:1–79:35, 2018. [Online]. Available: https://doi.org/10.1145/3214303

[19] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, no. 3-4, pp. 211–407, 2014. [Online]. Available: https://doi.org/10.1561/0400000042

[20] M. Blanton, A. R. Kang, S. Karan, and J. Zola, "Privacy preserving analytics on distributed medical data," 2018, *arXiv:1806.06477*.

[21] B. Jia, X. Zhang, J. Liu, Y. Zhang, K. Huang, and Y. Liang, "Blockchain-enabled federated learning data protection aggregation scheme with differential privacy and homomorphic encryption in iiot," *IEEE Trans. Ind. Informatics*, vol. 18, no. 6, pp. 4049–4058, 2022. [Online]. Available: https://doi.org/10.1109/TII.2021.3085960

[22] Q. Xia, E. B. Sifah, K. O. Asamoah, J. Gao, X. Du, and M. Guizani, "Medshare: Trust-less medical data sharing among cloud service providers via blockchain," *IEEE Access*, vol. 5, pp. 14 757–14 767, 2017. [Online]. Available: https://doi.org/10.1109/ACCESS.2017.2730843

[23] V. Reniers, Y. Gao, R. Zhang, P. Viviani, A. Madhusudan, B. Lagaisse, S. Nikova, D. V. Landuyt, R. Lombardi, B. Preneel, and W. Joosen, "Authenticated and Auditable Data Sharing via Smart Contract," in *Proc. 35th Annu. ACM Symp. Appl. Comp.*, C. Hung, T. Cerný, D. Shin, and A. Bechini, Eds. Brno, Czech Republic: ACM, 2020, pp. 324–331. [Online]. Available: https://doi.org/10.1145/3341105.3373957

[24] H. Han, R. K. Shiwakoti, R. Jarvis, C. Mordi, and D. Botchie, "Accounting and auditing with blockchain technology and artificial intelligence: A literature review," *Int. J. Account. Inf. Syst.*, vol. 48, p. 100598, 2023. [Online]. Available: https://doi.org/10.1016/j.accinf.2022.100598

[25] H. Shafagh, L. Burkhalter, A. Hithnawi, and S. Duquennoy, "Towards blockchain-based auditable storage and sharing of iot data," in *Proc. Cloud Comput. Secur. Workshop*, B. Thuraisingham, G. Karame, and A. Stavrou, Eds. Dallas, TX, USA: ACM, Nov. 2017, pp. 45–50. [Online]. Available: https://doi.org/10.1145/3140649.3140656

[26] E. Kokoris-Kogias, E. C. Alp, L. Gasser, P. Jovanovic, E. Syta, and B. Ford, "CALYPSO: private data management for decentralized ledgers," *Proc. VLDB Endow.*, vol. 14, no. 4, pp. 586–599, 2020. [Online]. Available: https://doi.org/10.14778/3436905.3436917

[27] S. E. Oh, J. Y. Chun, L. Jia, D. Garg, C. A. Gunter, and A. Datta, "Privacy-preserving audit for broker-based health information exchange," in *Proc. 4th ACM Conf. Data Appl. Secur. Privacy*, E. Bertino, R. S. Sandhu, and J. Park, Eds. San Antonio, TX, USA: ACM, Mar. 2014, pp. 313–320. [Online]. Available: https://doi.org/10.1145/2557547.2557576

[28] S. Cha, S. Baek, and S. Kim, "Blockchain based sensitive data management by using key escrow encryption system from the perspective of supply chain," *IEEE Access*, vol. 8, pp. 154 269–154 280, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.3017871

[29] N. P. Smart, "The exact security of ECIES in the generic group model," in *Proc. IMA Int. Conf. Cryptogr. Coding*, B. Honary, Ed. Berlin, Germany: Springer, 2001, pp. 73–84. [Online]. Available: https://doi.org/10.1007/3-540-45325-3_8

[30] NTT Information Sharing Platform Laboratories, "PSEC-KEM specification," NTT Coorporation, Tech. Rep., 2008.

[31] J. W. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan, and D. Stebila, "Frodo: Take off the ring! practical, quantum-secure key exchange from LWE," in *Proc. ACM Comput. Commun. Secur.*, E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds. Vienna, Austria: ACM, Oct. 2016, pp. 1006–1018. [Online]. Available: https://doi.org/10.1145/2976749.2978425

[32] J. W. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS - kyber: A cca-secure module-lattice-based KEM," in *Proc. IEEE Eur. Symp. Secur. Privacy*. London, U.K.: IEEE, Apr. 2018, pp. 353–367. [Online]. Available: https://doi.org/10.1109/EuroSP.2018.00032

[33] H. Krawczyk, "HMQV: A high-performance secure diffie-hellman protocol," in *Proc. Annu. Int. Cryptol. Conf.*, V. Shoup, Ed. Santa Barbara, CA, USA: Springer, Aug. 2005, pp. 546–566. [Online]. Available: https://doi.org/10.1007/11535218_33

[34] L. Law, A. Menezes, M. Qu, J. A. Solinas, and S. A. Vanstone, "An efficient protocol for authenticated key agreement," *Des. Codes Cryptogr.*, vol. 28, no. 2, pp. 119–134, 2003.

[35] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," in *Proc. 39th Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, B. Preneel, Ed. Springer, 2000, pp. 139–155. [Online]. Available: https://doi.org/10.1007/3-540-45539-6_11

[36] A. P. Sarr and P. Elbaz-Vincent, "On the security of the (F)HMQV protocol," in *Proc. Int. Conf. Cryptol. Africa*, D. Pointcheval, A. Nitaj, and T. Rachidi, Eds. Fes, Morocco: Springer, 2016, pp. 207–224. [Online]. Available: https://doi.org/10.1007/978-3-319-31517-1_11

[37] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Proc. Annu. Int. Cryptol. Conf.*, J. Feigenbaum, Ed. Santa Barbara, CA, USA: Springer, Aug. 1991, pp. 129–140. [Online]. Available: https://doi.org/10.1007/3-540-46766-1_9

[38] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Secure distributed key generation for discrete-log based cryptosystems," *J. Cryptol.*, vol. 20, no. 1, pp. 51–83, 2007. [Online]. Available: https://doi.org/10.1007/s00145-006-0347-3

[39] A. Kate and I. Goldberg, "Distributed key generation for the internet," in *Proc. 29th IEEE Int. Conf. Distrib. Comput. Syst.* Montreal, QC, Canada: IEEE Computer Society, Jun. 2009, pp. 119–128. [Online]. Available: https://doi.org/10.1109/ICDCS.2009.21

[40] P. Schindler, A. Judmayer, N. Stifter, and E. Weippl, "Ethdkg: Distributed key generation with ethereum smart contracts," Cryptology ePrint Archive, Paper 2019/985, 2019. [Online]. Available: https://eprint.iacr.org/2019/985