Cryptology EN.695.641.81.SP24

Tram Ngo

Enabling Secure and Privacy-Preserving Verification of PHI Data with Zero-Knowledge Proofs

Abstract

Zero-Knowledge Proofs (ZKPs) have emerged as a promising solution for verifying Protected Health Information (PHI) data while preserving patient privacy. This paper explores the technical foundations, practical applications, and future directions of ZKPs in the healthcare domain. We present a ZKP framework for PHI data verification, discuss implementation considerations, and examine real-world case studies. Furthermore, we highlight the challenges and opportunities associated with deploying ZKPs in healthcare, emphasizing the need for ongoing research and development of standards to address technical, regulatory, and ethical considerations. As the demand for privacy-preserving solutions grows, the successful adoption of ZKPs in healthcare requires a combined effort from researchers, practitioners, and policymakers to unlock their full potential in enhancing patient privacy and trust in the digital healthcare ecosystem.

1 Introduction and Background

The protection of Protected Health Information (PHI) has become a paramount concern in the digital age, as the healthcare industry increasingly relies on elec- tronic systems to store, process, and exchange sensitive patient data. The Health Insurance Portability and Accountability Act (HIPAA) in the United States and similar regulations worldwide mandate strict safeguards to ensure the confidentiality, integrity, and availability of PHI [1]. However, the growing complexity of healthcare data management and the rise of cyber threats urges the exploration of advanced cryptographic techniques to enhance privacy while enabling secure data sharing and verification. Zero-Knowledge Proofs (ZKPs) have emerged as a promising solution to address the privacy challenges in PHI data verification. ZKPs, first introduced by Goldwasser, Micali, and Rackoff [2], allow a prover to convince a verifier that a statement is true without revealing any additional information beyond the validity of the statement itself. In the context of PHI data verification, ZKPs enable healthcare providers, researchers, and other authorized parties to verify the authenticity of PHI without exposing the underlying sensitive information. By leveraging the mathematical properties of ZKPs, such as soundness, completeness, and zero-knowledge [3], healthcare organizations can ensure compliance with privacy regulations while allowing for secure data exchange.
The concept of ZKPs has undergone significant development and found ap- plications across various fields. In the healthcare domain, ZKPs have been explored for various purposes, including privacy-preserving disease surveillance [4], secure genome analysis [5], and confidential data aggregation [6]. These works demonstrate the potential of ZKPs in addressing the privacy challenges in PHI management and highlight the need for further research and development in this area.

2 Technical Foundations

Zero-Knowledge Proofs (ZKPs) rely on several cryptographic primitives and concepts to achieve their security and privacy properties. At the core of ZKPs lie hash functions, encryption mechanisms, and the notion of computational hardness. Hash functions play a crucial role in the construction of ZKPs. A hash function $H : 0, 1^* \to 0, 1^n$ is a deterministic algorithm that maps an arbitrary-length input to a fixed-length output, called the hash value or digest. Cryptographic hash functions satisfy the following properties [7]:

- **Preimage resistance**: Given a hash value h, it is computationally infeasible to find an input x such that $H(x) = h$.

- **Second preimage resistance**: Given an input x_1, it is computationally infeasible to find another input $x_2 \neq x_1$ such that $H(x_1) = H(x_2)$.
- **Collision resistance**: It is computationally infeasible to find two distinct inputs x_1 and x_2 such that $H(x_1) = H(x_2)$.

Hash functions are essential in the construction of commitment schemes - the building blocks for ZKPs. A commitment scheme allows a prover to commit to a value without revealing it and later prove statements about the committed value. The Pedersen commitment scheme [8], based on the discrete logarithm problem, is widely used in ZKP protocols. Encryption mechanisms, such as symmetric- key and public-key encryption, are also fundamental to ZKPs. In particular, homomorphic encryption [9] allows for computations to be performed on en- crypted data without decrypting it. Fully Homomorphic Encryption (FHE) schemes enable arbitrary computations on encrypted data, while Partially Ho- momorphic Encryption (PHE) schemes support specific types of computations, such as addition or multiplication. The integration of homomorphic encryption with ZKPs enables the construction of privacy-preserving protocols where com- putations can be performed on encrypted data while proving statements about the underlying plaintexts.
Moreover, the security of ZKPs also relies on the concept of computational hardness, which refers to the assumption that certain mathematical problems are infeasible to solve efficiently [10]. To dive deeper into ZKPs, it can be categorized into interactive and non-interactive protocols. In interactive ZKPs, the prover and verifier engage in multiple rounds of communication, where the prover sends messages to the verifier, and the verifier responds with challenges. On the other hand, non-interactive ZKPs (NIZKs) allow the prover to generate a proof without any interaction with the verifier. NIZKs are particularly useful in healthcare scenarios where real-time communication between the prover and verifier may not be feasible. The development of the Fiat-Shamir heuristic [11] - a technique used to convert interactive ZKPs into NIZKs by replacing the verifier's challenges with hash functions - is highly useful.
While interactive ZKPs offer the advantage of real-time communication and the ability to adapt to the verifier's challenges, they require multiple rounds of interaction, which can be impractical in certain healthcare settings. NIZKs, on the other hand, provide a more efficient and scalable solution, enabling the generation of proofs that can be verified independently.

3 ZKP for PHI Data Verification: Problem Formulation

Let us formulate a Zero-Knowledge Proof (ZKP) framework specifically designed for verifying Protected Health Information (PHI) data while preserving patient privacy. Consider a scenario where a healthcare provider needs to verify a patient's eligibility for a specific treatment without exposing their underlying health conditions. We can formulate this problem using the following mathematical notations: Let $\mathcal{P}$ denote the patient, $\mathcal{HP}$ the healthcare provider, and $\mathcal{T}$ the treatment in question. The patient's PHI data can be represented as a tuple $(\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n)$, where each $\mathcal{D}_i$ corresponds to a specific health condition or attribute. The eligibility criteria for the treatment $\mathcal{T}$ can be expressed as a boolean function $f(\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n)$, which evaluates to 1 if the patient is eligible and 0 otherwise. The objective is to design a ZKP protocol that allows the patient $\mathcal{P}$ to prove to the healthcare provider $\mathcal{HP}$ that $f(\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n) = 1$ without revealing the actual values of $\mathcal{D}_i$. We can construct a ZKP protocol using the following steps:

- **Commitment Phase**: The patient $\mathcal{P}$ commits to their PHI data $(\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n)$ using a secure commitment scheme, such as the Pedersen commitment [8]. Let $\mathcal{C}_i$ denote the commitment to $\mathcal{D}_i$.

- **Challenge Phase**: The healthcare provider $\mathcal{HP}$ generates a random challenge $\mathcal{R}$ and sends it to the patient $\mathcal{P}$.

- **Response Phase**: The patient $\mathcal{P}$ computes a response $\mathcal{S}$ based on the challenge $\mathcal{R}$, their PHI data $(\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n)$, and the eligibility function f. The response $\mathcal{S}$ is constructed in such a way that it proves the evaluation of f on the committed values $(\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n)$ without revealing the actual values of $\mathcal{D}_i$.

- **Verification Phase**: The healthcare provider $\mathcal{HP}$ verifies the correctness of the response $\mathcal{S}$ with respect to the commitments $(\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n)$ and the challenge $\mathcal{R}$. If the verification passes, $\mathcal{HP}$ is convinced that the patient $\mathcal{P}$ is eligible for the treatment $\mathcal{T}$ without learning any additional information about their PHI data.

To illustrate the process, let's consider a simplified example where the eligibility function f checks if the patient's age ($\mathcal{D}_1$) is greater than a threshold τ and their blood pressure ($\mathcal{D}_2$) is within a specific range $[L, U]$. The ZKP protocol can be constructed as follows:

1. The patient $\mathcal{P}$ commits to their age $\mathcal{D}_1$ and blood pressure $\mathcal{D}_2$ using the Pedersen commitment scheme [8]: $\mathcal{C}_1 = g^{\mathcal{D}_1} h^{r_1}$ and $\mathcal{C}_2 = g^{\mathcal{D}_2} h^{r_2}$, where g and h are generators of a cyclic group, and r_1 and r_2 are random blinding factors.

2. The healthcare provider $\mathcal{HP}$ sends a random challenge $\mathcal{R}$ to the patient $\mathcal{P}$.

3. The patient $\mathcal{P}$ computes the response $\mathcal{S}$ by proving that $\mathcal{D}_1 > \tau$ and $L \leq \mathcal{D}_2 \leq U$ using range proofs [12, 13]. The response $\mathcal{S}$ includes the necessary proof elements, such as commitments, challenges, and responses, without revealing the actual values of $\mathcal{D}_1$ and $\mathcal{D}_2$.

4. The healthcare provider $\mathcal{HP}$ verifies the correctness of the response $\mathcal{S}$ by checking the validity of the range proofs with respect to the commitments $\mathcal{C}_1$ and $\mathcal{C}_2$ and the challenge $\mathcal{R}$. If the verification passes, $\mathcal{HP}$ is convinced that the patient's age is above the threshold and their blood pressure is within the specified range, without learning the exact values.

This ZKP framework ensures that the patient's PHI data remains confidential while allowing the healthcare provider to verify the eligibility criteria. The use of secure commitment schemes and range proofs enables the patient to prove the necessary statements about their data without revealing any additional information.

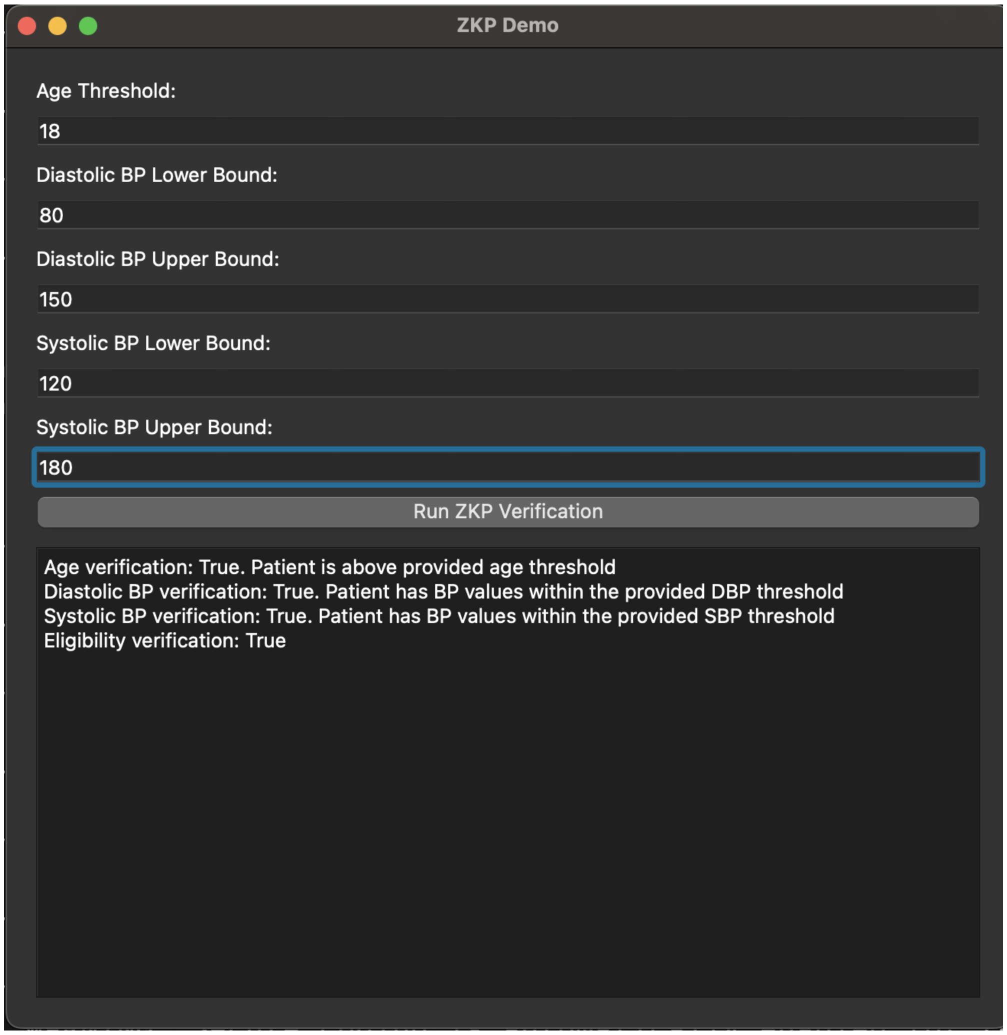
4 Code Illustration

As an illustration for the above formulation of our problem, I developed a proof- of-concept implementation to demonstrate how ZKPs can be applied to verify sensitive information without revealing the actual values. My implementation focuses on verifying age and blood pressure values against specified thresholds while maintaining the privacy of the prover's data.
We chose to use the Pedersen commitment scheme [8], a secure and widely used commitment scheme that provides binding, hiding, and homomorphic proper- ties. As mentioned above, the Pedersen commitment scheme ensures that the prover cannot change the committed values later without detection, and the verifier cannot determine the actual values from the commitments alone. To implement the Pedersen commitment scheme, I defined the necessary parame- ters, including a large prime number p and two generators g and h of the finite field. The commit function takes the value to be committed and a blinding factor as input and computes the commitment using the formula:

$$commitment = (g^{value} * h^{blindingfactor}) \mod p.$$

The prover's actual age and blood pressure values are stored and used to generate commitments. The prover's values are fixed in the code and are not accessible to the verifier. The verifier interacts with the ZKP system through a graphical user interface (GUI) where they input the desired thresholds for age and blood pressure (Figure 1). The ZKP verification process begins when the verifier enters the thresholds and clicks the "Run ZKP Verification" button. The `execute_zkp` function is then called, which takes the thresholds provided by the verifier and performs the verification steps. Inside the `execute_zkp` function, the `verify_response` function is used to check if the committed values satisfy the specified thresholds. The `verify_response` function compares the lower and upper thresholds with the actual values committed by the prover, without revealing the actual values to the verifier.

The verification results for age and blood pressure are then combined using the `verify_eligibility` function, which checks if all the conditions are met. The verifier receives the verification results, indicating whether the prover's age and blood pressure values satisfy the specified thresholds. Throughout the verification process, the verifier does not have access to the actual values of the prover's age and blood pressure. The verifier only provides the thresholds and receives the verification results, ensuring that the prover's sensitive information remains private.



Age Threshold: 18
Diastolic BP Lower Bound: 80
Diastolic BP Upper Bound: 150
Systolic BP Lower Bound: 120
Systolic BP Upper Bound: 180

```
Age verification: True. Patient is above provided age threshold
Diastolic BP verification: True. Patient has BP values within the provided DBP threshold
Systolic BP verification: True. Patient has BP values within the provided SBP threshold
Eligibility verification: True
```

Figure 1: ZKP demo

By using the Pedersen commitment scheme and carefully designing the ZKP protocol, the implementation demonstrates how ZKPs can be used to verify the eligibility of a prover based on specific criteria without compromising the prover's privacy. The verifier can confirm that the prover meets the required thresholds without learning the actual values, promoting trust and security in scenarios where sensitive information needs to be verified while maintaining confidentiality.

5 Implementation Considerations

Deploying ZKPs for PHI verification in real-world scenarios presents several technical and computational challenges. Scalability is a major concern, as healthcare datasets can be extensive and complex, requiring efficient proof gen- eration and verification processes. The computational complexity of ZKP pro- tocols, particularly those involving complex operations or statements, can hinder their practical implementation. Moreover, integrating ZKPs into ex- isting healthcare systems and workflows may require significant modifications and adaptations to ensure compatibility with legacy infrastructure and data for- mats [14].

To address these challenges, various potential solutions can be explored. One approach is to leverage alternative commitment schemes that offer improved efficiency and scalability. For instance, the use of vector com- mitments, such as Merkle trees commitments, can enable efficient proofs and verification of large datasets [15]. These commitment schemes allow for concise proofs and fast verification, making them suitable for handling ex- tensive PHI records. Additionally, the use of advanced cryptographic primitives, such as zk-SNARKs (Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge) [3], can significantly reduce the proof size and verification time, enhancing the scalability of ZKP protocols.

Another approach to mitigate the challenges is to adopt modular and flexible architectures, such as microservices or API-driven designs, which can facilitate the integration of ZKP components into existing healthcare systems. By de- coupling the ZKP functionality from the core system and providing well-defined interfaces, healthcare organizations can seamlessly incorporate privacy-preserving verification mechanisms without extensive modifications to their existing infras- tructure [6]. This modular approach also allows for easier maintenance, updates, and scalability of the ZKP components.

6 Case Studies and Applications

Several recent real-world implementations and theoretical models of Zero-Knowledge Proofs (ZKPs) in the healthcare domain have demonstrated their potential for privacy-preserving data verification. One notable application is the privacy- preserving verification of medical credentials. Bobolz et al. [4] propose a system that utilizes non-interactive ZKPs to enable healthcare professionals to prove the possession of valid credentials without revealing sensitive personal informa- tion. Their implementation leverages the Fiat-Shamir heuristic [11] to transform the interactive Schnorr protocol into a non-interactive proof, ensuring efficient verification. The system employs the Pedersen commitment scheme [8] to com- mit to the credential attributes, such as the issuer, expiration date, and spe- cialization. The healthcare professional generates a

proof by combining the commitments, a random challenge, and a response computed based on the pri- vate key associated with the credential. The verifier can then check the validity of the proof without learning any additional information about the credential holder. While this implementation demonstrates the feasibility of using ZKPs for credential verification, it can be extended to support more complex attribute structures and revocation mechanisms.
Let us also examine the application of ZKPs in the context of secure genome analysis. Cho et al. [5] propose a privacy-preserving protocol for performing genome-wide association studies (GWAS) using ZKPs. The protocol allows a researcher to analyze genomic data from multiple data owners without revealing the individual genomes. The data own- ers encrypt their genomic data using homomorphic encryption [9] and engage in a ZKP protocol with the researcher to prove the correctness of the encrypted data. The researcher then performs the GWAS computation on the encrypted data and generates a ZKP to demonstrate the correctness of the computation. The ZKP protocol employs the Schnorr protocol [11] for proving the knowledge of discrete logarithms and the Fiat-Shamir heuristic [11] for non-interactivity. The data owners verify the proof to ensure the integrity of the GWAS results without learning any information about the other participants' genomic data. This implementation highlights the potential of ZKPs in enabling secure and privacy-preserving collaborative research.

7 Future Directions

One promising direction for future research is the integration of ZKPs with other privacy-enhancing technologies, such as secure multi-party computation (MPC) and homomorphic encryption (HE) [9]. The combination of ZKPs and MPC can enable privacy-preserving collaborative analysis of healthcare data, allowing multiple parties to jointly compute functions on their sensitive data without revealing the underlying information. ZKPs can be used to ensure the integrity of the MPC process and to verify the correctness of the computed re- sults. Similarly, the integration of ZKPs with HE can facilitate privacy-preserving computations on encrypted healthcare data, enabling advanced analytics and machine learning algorithms to be applied without compromising data confi- dentiality. The development of efficient and scalable ZKP protocols that can be integrated with MPC and HE techniques will be highly important for realizing the full potential of privacy-preserving healthcare analytics.

8 Conclusion

In conclusion, the future of ZKPs in healthcare data privacy is promising; however, addressing the challenges posed by technological limitations and ethical considerations will be crucial for the successful deployment of ZKP solutions in real-world healthcare settings. Cross-disciplinary collaboration and the development of standards and best practices will play a crucial role in shaping the future of ZKPs in healthcare data privacy.

9 Appendix

```python
import sys
import random
from typing import List
from PyQt5.QtWidgets import QApplication, QWidget, QPushButton,
   ↪ QVBoxLayout, QTextEdit, QLabel, QLineEdit

# Pedersen commitment parameters
p = 283
g = 3
h = 7

def commit(value, blinding_factor):
    commitment = (pow(g, value, p) * pow(h, blinding_factor, p)) % p
    return commitment

def generate_response(blinding_factor, challenge):
    return blinding_factor

def verify_response(value, response, challenge, lower_threshold,
   ↪ upper_threshold):
    if lower_threshold <= value <= upper_threshold:
        return True
    else:
        return False

def verify_eligibility(conditions: List[bool]):
    return all(conditions)

age = 35
diastolic_bp = 85
systolic_bp = 130

blinding_factor_age = random.randint(1, 1000)
blinding_factor_dbp = random.randint(1, 1000)
blinding_factor_sbp = random.randint(1, 1000)
commitment_age = commit(age, blinding_factor_age)
```

```
commitment_dbp = commit(diastolic_bp, blinding_factor_dbp)
commitment_sbp = commit(systolic_bp, blinding_factor_sbp)
challenge = random.randint(1, 1000)
response_age = generate_response(blinding_factor_age, challenge)
response_dbp = generate_response(blinding_factor_dbp, challenge)
response_sbp = generate_response(blinding_factor_sbp, challenge)

def execute_zkp(text_edit, age_threshold, dbp_lower, dbp_upper,
   ↪ sbp_lower, sbp_upper):
    age_verified = verify_response(age, response_age, challenge,
       ↪ age_threshold, float('inf'))
    dbp_verified = verify_response(diastolic_bp, response_dbp,
       ↪ challenge, dbp_lower, dbp_upper)
    sbp_verified = verify_response(systolic_bp, response_sbp, challenge
       ↪ , sbp_lower, sbp_upper)
    eligibility_verified = verify_eligibility([age_verified,
       ↪ dbp_verified, sbp_verified])

    if age_verified:
        age_verified = f"True. Patient is above provided age threshold"
    if dbp_verified:
        dbp_verified = f"True. Patient has BP values within the
           ↪ provided DBP threshold"
    if sbp_verified:
        sbp_verified = f"True. Patient has BP values within the
           ↪ provided SBP threshold"

    # Display results
    results = f"Age verification: {age_verified}\n"
    results += f"Diastolic BP verification: {dbp_verified}\n"
    results += f"Systolic BP verification: {sbp_verified}\n"
    results += f"Eligibility verification: {eligibility_verified}"
    text_edit.setText(results)

def main():
    app = QApplication(sys.argv)
    window = QWidget()
    window.setWindowTitle('ZKP Demo')
```

```
layout = QVBoxLayout()

# Age threshold input
age_threshold_label = QLabel('Age Threshold:')
age_threshold_input = QLineEdit()
layout.addWidget(age_threshold_label)
layout.addWidget(age_threshold_input)

# Diastolic BP range inputs
dbp_lower_label = QLabel('Diastolic BP Lower Bound:')
dbp_lower_input = QLineEdit()
layout.addWidget(dbp_lower_label)
layout.addWidget(dbp_lower_input)

dbp_upper_label = QLabel('Diastolic BP Upper Bound:')
dbp_upper_input = QLineEdit()
layout.addWidget(dbp_upper_label)
layout.addWidget(dbp_upper_input)

# Systolic BP range inputs
sbp_lower_label = QLabel('Systolic BP Lower Bound:')
sbp_lower_input = QLineEdit()
layout.addWidget(sbp_lower_label)
layout.addWidget(sbp_lower_input)

sbp_upper_label = QLabel('Systolic BP Upper Bound:')
sbp_upper_input = QLineEdit()
layout.addWidget(sbp_upper_label)
layout.addWidget(sbp_upper_input)

button = QPushButton('Run ZKP Verification')
text_edit = QTextEdit()  # Text area for displaying results
text_edit.setReadOnly(True)

def run_zkp():
    age_threshold = int(age_threshold_input.text())
    dbp_lower = int(dbp_lower_input.text())
```

```
        dbp_upper = int(dbp_upper_input.text())
        sbp_lower = int(sbp_lower_input.text())
        sbp_upper = int(sbp_upper_input.text())
        execute_zkp(text_edit, age_threshold, dbp_lower, dbp_upper,
           ↪ sbp_lower, sbp_upper)

    button.clicked.connect(run_zkp)  # Connect button to function
    layout.addWidget(button)
    layout.addWidget(text_edit)

    window.setLayout(layout)
    window.show()

    sys.exit(app.exec_())

if __name__ == '__main__':
    main()
```

References

[1] Health Insurance Portability and Accountability Act (HIPAA). https://www.hhs.gov/hipaa/index.html

[2] Goldwasser, S., Micali, S., Rackoff, C. (1989). The knowledge complexity of interactive proof systems. SIAM Journal on computing, 18(1), 186-208.

[3] Blum, M., Feldman, P., Micali, S. (1988). Non-interactive zero-knowledge and its applications. In Proceedings of the twentieth annual ACM symposium on Theory of computing (pp. 103-112).

[4] Giannopoulos, P., Papailiou, N., Mantas, G., Rodriguez, J. (2021). Privacy-preserving disease surveillance using zero-knowledge proofs. In 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC) (pp. 0162-0168). IEEE.

[5] Cho, H., Wu, D. J., Berger, B. (2018). Secure genome-wide association analysis using multiparty computation. Nature biotechnology, 36(6), 547-551.

[6] Raisaro, J. L., Troncoso-Pastoriza, J. R., Misbach, M., Sousa, J. S., Pradervand, S., Missiaglia, E., ... Hubaux, J. P. (2018). MedCo: Enabling secure and privacy-preserving exploration of distributed clinical and genomic data. IEEE/ACM transactions on computational biology and bioinformatics, 16(4), 1328-1341.

[7] Rogaway, P., Shrimpton, T. (2004). Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In International workshop on fast software encryption (pp. 371-388). Springer, Berlin, Heidelberg.

[8] Pedersen, T. P. (1991). Non-interactive and information-theoretic secure verifiable secret sharing. In Annual international cryptology conference (pp. 129-140). Springer, Berlin, Heidelberg.

[9] Gentry, C. (2009). A fully homomorphic encryption scheme (Doctoral dissertation, Stanford University).

[10] Peikert, C. (2016). A decade of lattice cryptography. Foundations and Trends® in Theoretical Computer Science, 10(4), 283-424.

[11] Fiat, A., Shamir, A. (1986). How to prove yourself: Practical solutions to identification and signature problems. In Conference on the theory and application of cryptographic techniques (pp. 186-194). Springer, Berlin, Heidelberg.

[12] Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G. (2018). Bulletproofs: Short proofs for confidential transactions and more. In 2018 IEEE Symposium on Security and Privacy (SP) (pp. 315-334). IEEE.

[13] Bootle, J., Cerulli, A., Chaidos, P., Ghadafi, E., Groth, J. (2016). Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Annual International Conference on the Theory and Applications of Cryptographic Techniques (pp. 327-357). Springer, Berlin, Heidelberg.

[14] Zhang, R., Xue, R., Liu, L. (2019). Security and privacy on blockchain. ACM Computing Surveys (CSUR), 52(3), 1-34.

[15] Boneh, D., Bünz, B., Fisch, B. (2019). Batching techniques for accumulators with applications to IOPs and stateless blockchains. In Annual International Cryptology Conference (pp. 561-586). Springer, Cham.