Proof of Spacetime as a Defensive Technique Against Model Extraction Attacks

Tatsuki Fukuda
Organization for Liberal Arts and Education, Shimonoseki City University, Shimonosek, Japan

***Abstract*—When providing a service that utilizes a machine learning model, the countermeasures against cyber-attacks are required. The model extraction attack is one of the attacks, in which an attacker attempts to replicate the model by obtaining a large number of input-output pairs. While a defense using Proof of Work has already been proposed, an attacker can still conduct model extraction attacks by increasing their computational power. Moreover, this approach leads to unnecessary energy consumption and might not be environmentally friendly. In this paper, the defense method using Proof of Spacetime instead of Proof of Work is proposed to reduce the energy consumption. The Proof of Spacetime is a method to impose spatial and temporal costs on the users of the service. While the Proof of Work makes a user to calculate until permission is granted, the Proof of Spacetime makes a user to keep a result of calculation, so the energy consumption is reduced. Through computer simulations, it was found that systems with Proof of Spacetime, compared to those with Proof of Work, impose 0.79 times the power consumption and 1.07 times the temporal cost on the attackers, while 0.73 times and 0.64 times on the non-attackers. Therefore, the system with Proof of Spacetime can prevent model extraction attacks with lower energy consumption.**

***Keywords*—Proof of spacetime; model extraction attacks; machine learning; security**

I. Introduction

In recent years, services known as Machine Learning as a Service (MLaaS), which utilize platforms such as Microsoft Azure and Amazon Machine Learning, have become increasingly popular. MLaaS enables users to pass input data to models stored on servers and receive corresponding output. In order for businesses to profit from MLaaS, they must allocate considerable resources towards training models on servers. However, there is a risk of model theft through attacks that target server-based models. These types of attacks, such as model extraction attacks that may not require training data [1], are increasing in frequency and severity.

Model extraction attacks involve repeatedly inputting data to a pre-trained machine learning model, obtaining many input-output pairs, and then training a local model based on that information in an attempt to replicate the original model. If a model is stolen, the MLaaS that was targeted could suffer a decline in revenue, and depending on the type of service, it may even be used as a foothold for other attacks.

There are various types of model extraction attacks, including Copycat CNN [2] and Knockoff Nets [3]. Copycat CNN involves creating a mimicked dataset by linking input data and their corresponding output from the targeted model, and then using it to train a local model to steal the model. On the other hand, Knockoff Nets use reinforcement learning to efficiently select input images. There are also methods that steal the model by aligning the gradients of the targeted model and the local model [4].

As defense mechanisms against model extraction attacks, there are active defense, passive defense, reactive defense, and proactive defense.

Active defense is a method of hindering the training of the attacker's local model by altering the output of the targeted model. For example, it is possible to distort the probability of the output without changing the most probable class in the last activation layer of the targeted model [5] or intentionally poisoning the output to prevent the attacker from obtaining accurate output and obstructing the training of the local model [6]. However, these methods also affect the output accuracy of the model, which is a problem that also affects the output obtained by non-attackers.

Passive defense is a defense mechanism that protects the targeted model by truncating its output or detecting attacks. For example, there are methods that analyze the distribution of data that users have previously queried to detect attackers. However, existing methods are limited to research results based on assumptions such as the small distance between natural data and synthetic data [7] or the distribution of the attacker's dataset showing significant deviations as anomalies [8].

Reactive defense methods are techniques that aim to prove that the attacker's local model has stolen the victim model, rather than detecting the attack itself. There are methods that use digital watermarks [9] or verify whether the model suspected of theft has a certain level of common knowledge with the victim model [10]. However, these methods cannot prevent the theft of the model, and there is no guarantee that the stolen model will not be used for other attacks.

Proactive defense is a technique that increases the attacker's burden by imposing some form of cost on the users who query the model. For example, there are techniques that use Proof of Work [11,12], which requires computation, to demand costs in terms of electricity or time from attackers. This technique makes it difficult for attackers to acquire many input-output relationships at a low cost and is a method that does not affect output accuracy for non-attackers. Furthermore, since this defense technique does not require any changes to the learning model itself, there is no need to train the model with specialized data or retrain a pre-trained model, making the cost of introducing the defense technique relatively low.

However, the calculation required in the Proof of Work consumes a significant amount of electricity, which is not sustainable from the perspective of the United Nations' Sustainable Development Goals (SDGs). As a solution, applying the Proof of Spacetime into the defense method against model extraction attacks is proposed in this paper. The Proof of Spacetime aims to limit access by attackers without repeating high load calculation and can reduce the unnecessary consumption of electricity.

II. Literature Review

Firstly, we describe the method of the Proof of Work and the Proof of Spacetime.

A. Proof of Work

Proof of Work is mainly used in cryptocurrencies such as Bitcoin [13], where a reward can be received in exchange for solving a given problem through computation. Generally, a hashcash [14] is used in Proof of Work, which imposes a calculation to find a string of characters that has a certain number of zeros in the upper bits when the hash is converted. The difficulty of the calculation can be determined based on the number of zeros required.

By using Proof of Work's hashcash, it is possible to inhibit an attacker from obtaining the input-output relationship of a model by making MLaaS users perform calculations to obtain the model's output. However, since non-attackers also use MLaaS, it is necessary to increase the difficulty of hashcash, i.e., the number of consecutive zeros, as the suspicion of the attacker increases.

As a result, the attacker needs to perform many computations to obtain the output of the model, which imposes time and power consumption constraints. Additionally, because the computation occupies the CPU or GPU, it prevents the attacker from evading time constraints by using multiple accounts. However, Proof of Work calculations force unnecessary computations and power consumption, which leads to the consumption of fossil fuels, making it environmentally unfavorable.

B. Differential Privacy

In a defense method using Proof of Work, it is necessary to distinguish between attackers and non-attackers, which can be achieved using differential privacy [15] as an indicator. Differential privacy is a concept that originally aims to protect personal information on a database and make statistical analysis possible. Differential privacy considers the privacy is secured if it is impossible to distinguish between the results obtained using a dataset that contains personal data and the results obtained using a dataset that excludes the personal data.

PATE [16] is a method for measuring differential privacy. PATE trains multiple models including personal data and builds a model that adopts the majority vote of their outputs. Then, the output of this model with noise added is used to train another model without using personal data. Through this process, the final model is trained without directly using personal data and with noise added, leading to the protection of differential privacy.

In our method, the differential privacy is used to judge the user whether or not an attacker and to determine how much cost to impose the user.

C. Merkle Tree

Merkle tree [17] is a technique that uses hash functions to summarize and verify data. Multiple data are hashed and then the hash values are added together in pairs, which are then hashed again. This process is repeated until a tree is created with hash values on each node. Fig. 1 shows an example of a Merkle tree generated from four data sets. The root node, h_{ABCD}, is called the Merkle root. The hash values h_A and h_{CD} that are needed to calculate the Merkle root from data *B* are called the Merkle path of data *B*.

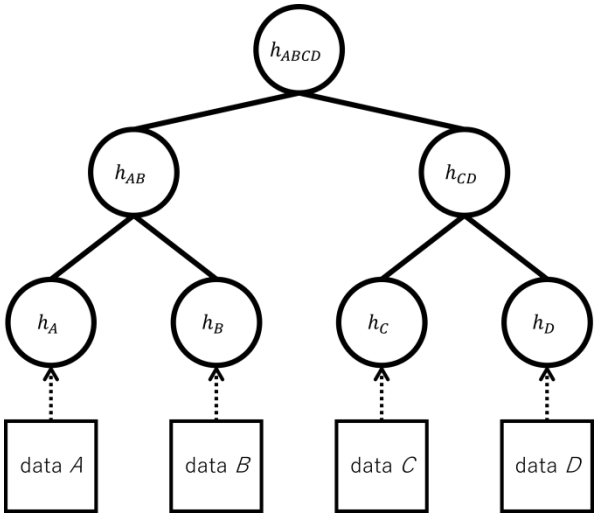

Fig. 1. Example of Merkle tree.

D. Proof of Spacetime

Proof of Spacetime [18] is a consensus algorithm used in virtual currencies such as Spacemesh. It is a method that occupies a certain amount of memory space for a certain amount of time. Specifically, it generates initial data in advance using a hash cache and considers the memory space and time occupied by that initial data as the cost.

Proof of Spacetime is a consensus algorithm used in virtual currencies such as Spacemesh. It is a method that occupies a certain amount of memory space for a certain period of time. It generates initial data using a hash cache and considers the memory space and time occupied by the initial data as costs. The Proof of Spacetime flow can be divided into initialization, proof, and verification stages. In the initialization stage, users save all hash-transformed data using a hash cache. In the proof stage, users create proof data using a Merkle tree. In the verification stage, the server verifies whether the proof data created in the proof stage is correct. The power consumed to build a Merkle tree is smaller than the power consumed to generate data in the initialization stage, because the hash transformation used to construct the Merkle tree is a repeated hash function that is faster than the hash function used in the initialization stage. The user-side cost in Proof of Spacetime can be represented as a monetary cost using (1) and (2),

$$C_{init} = p \times n \times C_p, \quad (1)$$

$$C_{proof} = s \times t \times C_{st}, \quad (2)$$

where C_{init} and C_{proof} are the cost in the initial and the proof stage, respectively. The cost in initial stage can be said also the cost of hash cash. Note that p is the amount of power consumed for one hash transformation, n is the number of times the hash is performed, C_p is the cost per power consumption, s is the size of occupied storage, t is the occupied time, and C_{st} is the cost per second for occupying 1 MB. A general flow of Proof of Spacetime is shown in Algorithm I.

The Proof of Work requires the user to perform a high-load calculation and continues until it is solved, whereas the Proof of Spacetime requires a relatively simple calculation and keeping the "evidence" of the calculation. In other words, the Proof of Spacetime does not require the user to keep moving at high power all the time, and power consumption can be suppressed.

Algorithm I:

```
function init_stage(id):
    sigma = hashcash(id)
    for i = 0…t:
        G[i] = hash_init(i, sigma)
        if G[i] has log2t or more leading zeros:
            return True
    return False
function proof_stage( ):
    tree = Merkle(G)
    path_list = all of the Merkle paths from all of leaves
    transmit G and path_list to the server
function verification_stage( ):
    for i = 0…t:
        if G[i] has log2t or more leading zeros:
            reconstruct tree from path_list
            return if tree is equivalent to Merkle(G)
    return False
```

III. METHODS

When using Proof of Work as a defense mechanism against model extraction attacks, excessive energy consumption and its negative impact on the environment have already been noted. Therefore, this study proposes a method using Proof of Spacetime. Specifically, the following steps STEP 1 to STEP 5 are taken:

STEP 1: Measure the differential privacy of the user that is

considered a non-attacker, denoted as D_n.

STEP 2: Measure the differential privacy of the current user,

denoted as D_c.

STEP 3: Calculate the difference D between D_n and D_c.

STEP 4: Calculate the temporal and spatial cost based on D.

STEP 5: Impose the temporal cost and spatial cost on the

current user using Proof of Spacetime.

The temporal cost reduces the efficiency of attackers obtaining input/output data, and the spatial cost prevents attackers from obtaining input/output data by parallel processing, i.e., using multiple accounts. However, there is a possibility that even non-attacker users may use the service multiple times, so it is desirable to minimize the initial hashcash as much as possible, which will also reduce unnecessary energy consumption. Therefore, the spatial cost for storage is fixed.

Compared to using the defense method with Proof of Work, using Proof of Spacetime provides the following four benefits:

- Power consumption can be reduced.
- Time costs imposed on users can be adjusted with little
- Change in computational complexity.
- Fine-tuning of costs is also possible.
- Costs can be demanded regardless of differences in machine resources.

The first benefit comes from the fact that Proof of Spacetime can reduce power consumption by using proof stages that require less power consumption than repeating hash calculations. This is because, as mentioned earlier, the hash conversion used for constructing a Merkle tree in the proof stage is a repetition of a high-speed hash function. Similarly, the second benefit is due to the small computational complexity of the proof stage, making it possible to adjust time costs without worrying too much about computational complexity.

The third benefit is due to the difficulty of adjusting difficulty levels when increasing the number of zeros in a hash cache from n to $n+1$, as increasing the number of zeros results in an average computational complexity increase of 2^n. With Proof of Spacetime, it is easy to simply change the occupied time.

The fourth benefit is that while costs that depend on computational complexity such as hash caches can be relatively reduced by increasing the performance of the attacker's machine, the time cost of Proof of Spacetime is not affected by the performance of the machine, and the space cost can be easily increased accordingly.

IV. EXPERIMENTS

Comparing the power consumption between Proof of Work and Proof of Spacetime:

A. Experimental Procedure

We prepared a target model for the attack: a ResNet34 for classifying cifar10 data (95.60% accuracy). Both of attackers and non-attackers randomly selected data from dataset not used for training the model to query. The number of queries is 5000 for each experiment. For the classification server, we implemented a defense method using Proof of Work and a defense method using Proof of Spacetime. We conducted the following measurements for a total of four servers:

- Power consumption of non-attackers.
- Power consumption of attackers using Knockoff Nets.
- Power consumption of attackers using Copycat CNN.

Power consumption was measured by measuring the overall power consumption of a computer shown in Table I with the minimum required processes running for program execution. In addition, the average power consumption during normal times when the program was not running was also measured.

TABLE I. COMPUTER USED FOR EXPERIMENTS

CPU	Core i5-9400F BOX
Memory	64GB
OS	Ubuntu Desktop 22.04

B. Results

The experimental results are shown in Table II. Fig. 2 to Fig. 4 illustrate the power consumption during program execution, where the vertical line shows the elapsed time from the query threw and the horizontal one shows the cumulative power consumption. Note that the power consumption while the computer is in idled for 30 minutes was 3.0×10^{-2} kWh and an average wattage is 60W.

TABLE II. RESULT FOR PROOF OF WORK

Client	Defence Method	Erapsed Time [Sec]	Power consumption [kWh]	Average wattage [W]
Non Attacker	PoW	48047	2.40	180
Knockoff	PoW	299563	9.36	112
Copycat	PoW	374146	11.52	110
Non Attacker	PoST	30685	1.75	205
Knockoff	PoST	360930	8.44	84
Copycat	PoST	358576	8.13	82

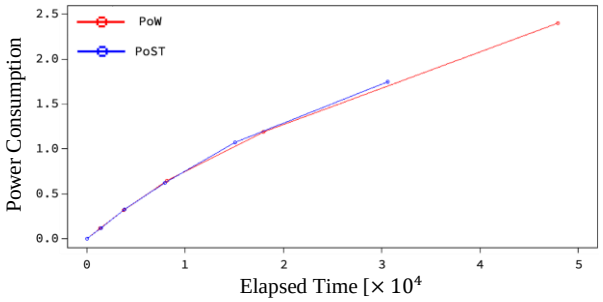

Fig. 2. Power consumption for a non-attacker.

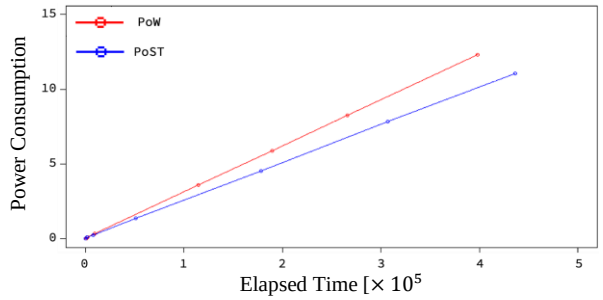

Fig. 3. Power consumption for an attacker with knockoff nets.

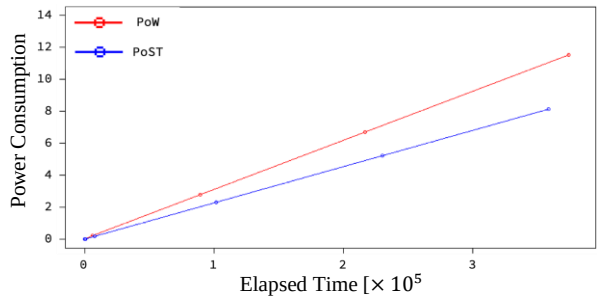

Fig. 4. Power consumption for an attacker with copycat.

Fig. 5 to Fig. 7 show the monetary cost for each attack. Note that the unit prices in (1) and (2) were set to $C_{st} = 2.50 \times 10^{-10}$ and $C_p = 31$ [19-21], respectively. The vertical line shows the elapsed time from the query threw and the horizontal one shows the monetary cost.

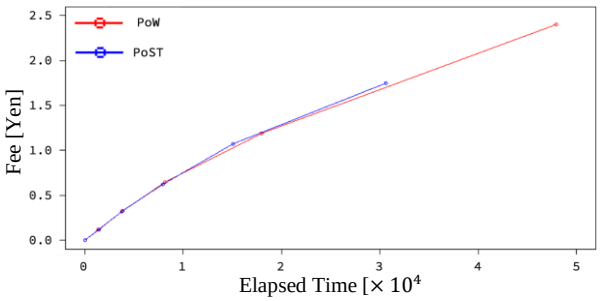

Fig. 5. Monetary cost based on the online storage services for a non-attacker.

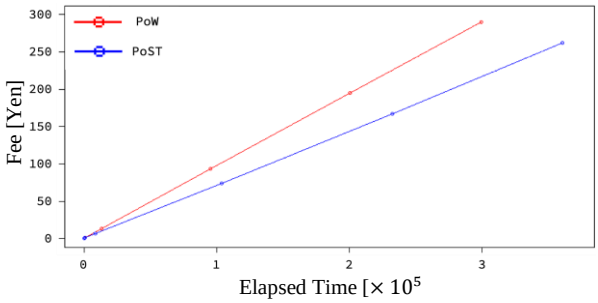

Fig. 6. Monetary cost based on the online storage services for an attacker with knockoff nets.

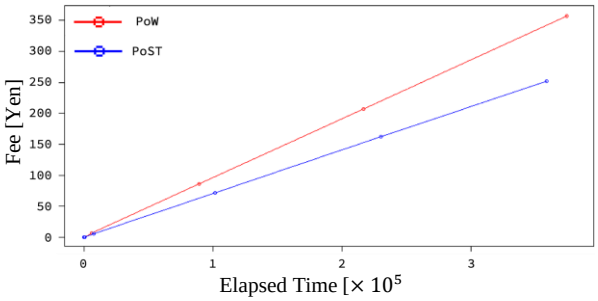

Fig. 7. Monetary cost based on the online storage services for an attacker with copycat.

V. Consideration of Results

Now, we consider the results obtained from the experiment from the viewpoint of temporal cost and power consumption.

A. Temporal Cost

According to Table I, the average time required per query for the defense mechanism using Proof of Work was 9.61 seconds for non-attackers and 67.4 seconds for attackers. On the other hand, when using Proof of Spacetime, the times were 6.14 seconds for non-attackers and 72.0 seconds for attackers. Therefore, the defense mechanism using Proof of Spacetime requires 0.64 times the temporal cost for non-attackers while 1.07 times for attackers, indicating its high performance as a defense mechanism.

B. Power Consumption

For non-attackers, it can be seen that Proof of Spacetime reduces the total power consumption by 0.73 times compared to Proof of Work, but the average power required is 1.14 times higher. This is because Proof of Spacetime requires more computation in the initialization phase, which occupies a larger portion of the non-attacker's time cost.

For attackers, it can be seen that Proof of Spacetime reduces the total power consumption by 0.79 times compared to Proof of Work, and the average wattage required is also reduced by 0.75 times.

C. Comprehensive Perspective

Comparing the defense mechanisms using Proof of Work and Proof of Spacetime, it was found that the latter has the following characteristics:

- Higher average power consumption for non-attackers
- Lower total power consumption for non-attackers
- Lower time cost for non-attackers
- Lower average power consumption for attackers
- Lower total power consumption for attackers
- Higher time cost for attackers

Since the goal of this study was to reduce unnecessary power consumption while preventing attacks, these results indicate that the goal was successfully achieved.

In addition, since the graphs of the monetary cost in Fig. 5 to Fig. 7 have a similar shape to the power consumption graphs in Fig. 2 to Fig.4, respectively. That means that it can be seen that storage costs are negligible compared to power costs. In other words, in terms of monetary cost, there is a trade-off between time cost and space cost in Proof of Spacetime. While storage was assumed to be the target of space cost in this study, cost-effectiveness can be improved by storing data in main memory.

When choosing between Proof of Spacetime and Proof of Work as defense mechanisms, the following points should be considered. Specifically, Proof of Work should be used to impose costs on attackers when the suspicion of an attack is low due to differential privacy. Proof of Spacetime should be used when the suspicion of an attack is high, to reduce power consumption while imposing time and space costs. By doing so, the costs imposed on non-attackers can be kept small, while the costs imposed on attackers can be increased.

VI. Conclusion

This paper proposed the use of Proof of Spacetime as a defense mechanism against model extraction attacks. The existing defense method, Proof of Work, is effective in preventing attackers from obtaining the input-output relationship of the model for model extraction attacks. However, it imposes unnecessary power consumption, which is not environmentally preferable. With our method, the user has

to calculate a relatively simple calculation and keeps the "evidence" of the calculation, so the total power consumption decrease compared to Proof of Work. The cost to impose to the user is determined according to the differential privacy.

The only drawback of Proof of Spacetime is the large average wattage for the non-attackers. As the future work, it is necessary to consider efficient ways to use Proof of Work and Proof of Spacetime for attackers and non-attackers separately.

REFERENCES

[1] Jean-Baptiste Truong, Pratyush Maini, Robert J. Walls, and Nicolas Papernot, "Data-free model extraction," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp.4771-4780, 2021.

[2] J. R. Correia-Silva, R. F. Berriel, C. Badue, A. F. de Souza, and T. Oliveira-Santos, "Copycat CNN: Stealing Knowledge by Persuading Confession with Random Non-Labeled Data," Proceedings of the 2018 International Joint Conference on Neural Networks, pp.1-8, 2018.

[3] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz, "Knockoff Nets: Stealing Functionality of Black-Box Models," Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp.4949-4958, 2019.

[4] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, Ananthram Swami, "Practical Black-Box Attacks against Machine Learning," Proceedings of the 2017 ACM on Asia conference on computer and communications security, pp.506-519, 2017.

[5] T. Lee, B. Edwards, I. Molloy and D. Su, "Defending Against Neural Network Model Stealing Attacks Using Deceptive Perturbations," Proceeding of the 2019 IEEE Security and Privacy Workshops, pp.43-49, 2019.

[6] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz, "Prediction Poisoning: Towards Defenses Against DNN Model Stealing Attacks," arXiv preprint arXiv:1906.10908, 2019.

[7] Mika Juuti, Sebastian Szyller, Samuel Marchal, N. Asokan, "PRADA: Protecting Against DNN Model Stealing Attacks," 2019 IEEE European Symposium on Security and Privacy, pp.512-527, 2019.

[8] Soham Pal, Yash Gupta, Aditya Kanade, Shirish Shevade, "Stateful Detection of Model Extraction Attacks," arXiv preprint arXiv:2107.05166, 2021.

[9] Hengrui Jia, Christopher A. Choquette-Choo, Varun Chandrasekaran, Nicolas Papernot, "Entangled Watermarks as a Defense against Model Extraction," arXiv preprint arXiv:2002.12200, 2020.

[10] Pratyush Maini, Mohammad Yaghini, Nicolas Papernot, "Dataset Inference: Ownership Resolution in Machine Learning," arXiv preprint arXiv:2104.10706, 2021.

[11] Adam Dziedzic, Muhammad Ahmad Kaleem, Yu Shen Lu, Nicolas Papernot, "Increasing the Cost of Model Extraction with Calibrated Proof of Work," arXiv preprint arXiv:2201.09243, 2022.

[12] Markus Jakobsson and Ari Juels, "Proofs of Work and Bread Pudding Protocols(Extended Abstract)," Secure Information Networks, pp.258-272, 1999.

[13] Nakamoto Satoshi. "Bitcoin: A Peer-to-Peer Electronic Cash System," Decentralized Business Review, 2008.

[14] Back Adam, "Hashcash-A Denial of Service Counter-Measure," 2002.

[15] Cynthia Dwork and Aaron Roth, "The Algorithmic Foundations of Differential Privacy," Foundations and Trends® in Theoretical Computer Science, vol.9, no.3-4, pp.211-407, 2014.

[16] Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian Goodfellow, Kunal Talwar, "Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data," arXiv preprint arXiv:1610.05755, 2016.

[17] Ralph C. Merkle, "Protocols for Public Key Cryptosystems," Secure communications and asymmetric cryptosystems, Routledge, pp.73-104, 2019.

[18] Tal Moran, Ilan Orlov, "Simple Proofs of Space-Time and Rational Proofs of Storage," Proceedings of the 39th Annual International Cryptology Conference, pp.18-22, 2019.

[19] "Plans and Pricing," Google LLC, https://one.google.com/about/plans, accessed May 14, 2023.

[20] "iCloud+ plans and pricing," Apple Inc., https://support.apple.com/en-asia/HT201238, accessed May 14, 2023.

[21] "FAQ," HOME ELECTRIC APPLIANCES FAIR TRADE CONFERENCE, https://www.eftc.or.jp/qa/, accessed May 14, 2023.