Blockchain-Based Covert Communication: A Detection Attack and Efficient Improvement

Zhuo Chen, Liehuang Zhu, *Senior Member, IEEE*, Peng Jiang, *Member, IEEE*, Zijian Zhang, *Senior Member, IEEE*, and Chengxiang Si

***Abstract*— Covert channels in blockchain networks achieve undetectable and reliable communication, while transactions incorporating secret data are perpetually stored on the chain, thereby leaving the secret data continuously susceptible to extraction. MTMM (IEEE Transactions on Computers 2023) is a state-of-the-art blockchain-based covert channel. It utilizes Bitcoin network traffic that will not be recorded on the chain to embed data, thus mitigating the above issues. However, we identify a distinctive pattern in MTMM, based on which we propose a comparison attack to accurately detect MTMM traffic. To defend against the attack, we present an improvement named ORIM, which exploits the permutation of transaction hashes within inventory messages to transmit secret data. ORIM leverages a pseudo-random function to obscure the transaction hashes involved in the permutation to ensure unobservability. The obfuscated values, rather than the original transaction hashes, are utilized to encode the confidential data. Furthermore, we introduce a variable-length encoding scheme predicated on complete binary trees. This scheme considerably amplifies the bandwidth and facilitates efficient encoding and decoding of secret data. Experimental results indicate that ORIM maintains unobservability and that ORIM's bandwidth is approximately 3.7 × of MTMM.**

***Index Terms*— Blockchain, covert communication, covert channel, Bitcoin, inventory message.**

I. Introduction

COVERT communication facilitates the confidential exchange of data in the presence of a warden [1]. This has extensive use in secure data transmission of smart infrastructures, such as the Internet of Things (IoT) [2] and 5G networks [3]. Practical covert communication is typically accomplished via a covert channel [4], a communication medium purposely concealed within a public channel. A specific type of covert channel is called as the network covert channel [5], which is hidden within computer networks to transmit data. Traditional network covert channels [6] utilize network protocols to hide data. For example, they might alter the inter-packet delay (IPD) between network packets to encode data. However, due to the inherent vulnerability of network packets to tampering, traditional network covert channels face the risk of channel restriction [7] and elimination [8].

Contrarily, the covert channel within the blockchain network mitigates the above risk with hiding the secret data in immutable blockchain transactions [9]. Typically, in such a covert channel, the sender creates transactions whose fields contain secret data and broadcasts these transactions to the blockchain network. The receiver extracts the secret data by reading these transactions from the public blockchain ledger. Transactions carrying secret data remain invulnerable to tampering, rendering the covert channel inherently reliable. Furthermore, as the sender does not engage in direct network communication with the receiver, the channel becomes more concealed and less susceptible to detection. Nevertheless, these covert channels are subject to three fundamental weaknesses:

- *Substantial communication costs.* The sender needs fees to create transactions and transmit data. The average fee for a Bitcoin transaction is \$1.106[1]. State-of-the-art works can covertly transmit 32-Byte data within a Bitcoin transaction, making the cost of transmitting 1-KB data \$35.392–an exorbitant expense.
- *Data mining.* Transactions containing secret data are permanently stored on the chain. Given the public and transparent nature of the blockchain ledger, the warden can easily access the entire ledger at any time, allowing them to analyze transactions and extract secret data.
- *Identity tracing.* The permanent storage of transactions on the blockchain enables the warden to identify transactions containing secret data. Upon detecting these transactions, the warden can trace the identity of the transaction creator (the sender) using tracing techniques [10].

To address the above drawbacks, researchers construct covert channels in the network layer of the blockchain that utilize the network traffic to embed data. Such channels do not require the sender to create transactions, but only require the sender to forward transactions that already exists, thus circumventing high fees. MTMM [11] is a representative solution. In MTMM, the sender conceals the secret data in the inventory (inv) messages of the network layer, which comprise multiple transaction hashes. A previous transaction hash greater than its subsequent transaction hash indicates

Received 3 April 2024; revised 4 July 2024; accepted 4 October 2024. Date of publication 11 October 2024; date of current version 22 October 2024. This work was supported by the National Natural Science Foundation of China under Grant 62232002. The associate editor coordinating the review of this article and approving it for publication was Dr. Ning Zhang. *(Corresponding author: Peng Jiang.)*

Zhuo Chen, Liehuang Zhu, Peng Jiang, and Zijian Zhang are with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: chen.zhuo@bit.edu.cn; liehuangz@bit.edu.cn; pennyjiang0301@gmail.com; zhangzijian@bit.edu.cn).

Chengxiang Si is with the National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing 100029, China (e-mail: sichengxiang@cert.org.cn).

Digital Object Identifier 10.1109/TIFS.2024.3478834

[1]The average Bitcoin transaction fee for August 25, 2023 from https://ycharts.com/indicators/bitcoin_average_transaction_fee

bit 0, otherwise it indicates bit 1. Given that the inv message is ephemeral and not permanently uploaded to the chain, data mining and identity tracing are evaded. In summary, compared to other network-based covert channels, MTMM has stronger reliability and anonymity. In blockchain-based covert channels, MTMM is the only practical approach without economic costs and security risks.

However, MTMM displays a distinctive pattern whereby any transaction hash within the inv message is invariably larger or smaller than all subsequent hashes. The distinctive pattern is determined only by the principles of the MTMM implementation and is a inherent flaw of MTMM. It always exists even if the embedded data has excellent properties such as freshness and indistinguishability. It is difficult to attack MTMM without considering the correlation between different transaction hashes. This is because MTMM does not destroy the inherent randomness of the data carrier, and there is basically no change in the statistical features before and after the carrier is embedded in the data. Despite the pattern between transaction hashes, MTMM is still valuable. It demonstrates through experiments that MTMM's pattern cannot be identified by commonly used data analysis techniques. Moreover, MTMM proposes a framework that can avoid economic costs and data mining risks, and provides a future direction for subsequent research. Based on the pattern, we propose a comparison attack designed to detect MTMM traffic. A potential adversary can ascertain whether an inv message conforms to the above pattern by comparing the transaction hashes within the inv message, thereby exposing the insecurity of MTMM.

To defense against the comparison attack, it is imperative to ensure unobservability while encoding secret data into inv messages. This requirement, however, incurs a significant technical challenge. An ordinary inv message's transaction hash sequence can be viewed as a random string, whereas a transaction hash sequence carrying secret data may exhibit discernible statistical properties. The sender must ensure that the encoded inv message also maintains the property of randomness. Our proposed solution involves obfuscating the transaction hashes of the inv message using a pseudo-random function (PRF). Subsequently, instead of using the original transaction values, the obfuscated values are utilized for the encoding of the secret data. This method assures that the encoded transaction hash sequence retains its inherent randomness attribute.

To further enhance the bandwidth for efficient communication, we leverage the permutations of transaction hashes within an inv message to represent secret data. An inv message containing n transaction hashes offers $n!$ (the factorial of n) permutations. The considerable number of permutations enables the possibility of achieving efficient covert channels. However, the process of encoding and decoding each permutation is a non-trivial task. The number of transaction hashes in each inv message is variable, resulting in differing data capacities among inv messages. Consequently, formulating a uniform encoding and decoding scheme for inv messages with diverse numbers of transaction hashes and permutations is challenging. Moreover, as the number of transaction hashes in an inv message ascends, the potential permutations $n!$ can become overwhelmingly large. In this scenario, the sender must swiftly sort the transaction hashes based on the secret data to achieve the intended permutations. Failure to do so may slow down the broadcasting rate of the Bitcoin node's inv message, which could potentially expose the covert channel via slower broadcasting rates. To overcome the above challenges, we devise a variable-length encoding scheme based on the complete binary tree to encode all possible permutations. The sender builds a complete binary tree with $n!$ leaf nodes and maps them onto $n!$ permutations. By exploiting the arithmetic characteristics of the binary tree, the sender can swiftly encode data into a specific permutation.

In this paper, we draw inspiration from MTMM and design covert channels with higher security and efficiency. The challenge of improving security lies in resisting the comparison attack. More specifically, it involves embedding data into inventory messages while maintaining their inherent randomness, which we address through the utilization of a PRF. To enhance efficiency, the challenge is to design a more efficient encoding scheme, which is achieved by encoding data into permutations of transaction hashes. Main contributions of this paper are as follows:

- We propose a comparison attack to undermine MTMM. To achieve this, we initially model the security of a covert channel that uses network traffic to hide secret data in a formal manner. A covert channel is secure if an adversary is unable to differentiate between normal network traffic and network traffic carrying data with a non-negligible advantage. Based on this model, we design an adversary to break MTMM. For each transaction hash in an inv message, the adversary determines whether it is simultaneously greater or less than all its subsequent transaction hashes. If so, the adversary concludes that the inv message contains secret data. We provide the detailed attack algorithm to MTMM.
- We propose a covert channel named ORIM, which encodes secret data as permutations of transaction hashes within inv messages. In ORIM, the sender initially obfuscates the transaction hashes using a PRF. The sender then employs permutations of the obfuscated values, rather than permutations of the original transaction hashes, to store secret data. Subsequently, we propose a complete binary tree-based variable length coding scheme to design codewords for each permutation, leveraging the arithmetic features of binary trees for swift encoding and decoding. We formally prove the unobservability of ORIM in the security model.
- We implement ORIM on the Bitcoin mainnet. By modifying and recompiling the Bitcoin source code, we obtain a Bitcoin node equipped with ORIM. We capture the ORIM traffic broadcast by the node and conduct experiments to verify its unobservability and evaluate its bandwidth. We analyze ORIM traffic using Kullback–Leibler Divergence (KLD), Kolmogorov–Smirnov (K-S) tests, and deep learning models. Experimental results show that ORIM traffic possesses unobservability. We conduct experiments on ORIM's bandwidth, which is 3.7 $\times$ of MTMM. We also compare the economic cost and

TABLE I
COMPARISON OF MTMM AND OTHER NETWORK-BASED COVERT CHANNELS

Scheme	Network type	Reliability			Anonymity	Strong computational capability
		Data tampering	MITM attack	DDoS attack		
[12]	IoT	×	×	×	×	×
[13]		×	×	×	×	×
[14]	CRN	×	×	✓	✓	✓
[15]		×	×	✓	✓	✓
[16]	UAV	×	×	✓	×	✓
[17]		×	×	✓	×	✓
MTMM [11]	Blockchain	✓	✓	✓	✓	✓
ORIM		✓	✓	✓	✓	✓

unobservability of ORIM and baselines, and ORIM achieves the lowest cost while ensuring unobservability.

II. Related Work

A. Covert Communication in Non-Blockchain Networks

Wang et al. [12] and Feng et al. [13] implemented covert communication within IoT networks. Xu et al. [14] and Lu et al. [15] delved into covert communication in cognitive radio networks (CRNs) to safeguard data security. Yan et al. [16] and Wang et al. [17] designed covert channels in unmanned aerial vehicle (UAV) assisted networks. Nevertheless, covert communication within these networks demonstrated deficiencies in reliability [18]. By comparison, in a blockchain network, communication between the sender and receiver transpires through their normal participation in blockchain networks, with the sender forwarding blockchain transactions and the receiver synchronizing the blockchain ledger. These behaviors align with those of regular blockchain participants, thereby facilitating the hiding of sender and receiver activities. Additionally, the decentralization and immutability nature of blockchain networks enhances the reliability.

B. Covert Communication in Blockchain Networks

Covert communication in blockchain networks can be categorized into two types: ledger-level covert communication and traffic-level covert communication. The ledger-level covert communication is achieved via covert storage channels. Partala [19] embedded data into the least significant bit (LSB) of the transaction address. Subsequent research [20], [21], [22] expanded the number of embedded bits to enhance communication efficiency. Luo et al. [23] established a covert channel by utilizing ASCII-encoded data as the transaction amount. Zhang et al. [24] stored data using custom fields of Bitcoin and Ethereum. Cryptographic primitives during transaction generation are also frequently used to construct subliminal channels [25], [26]. For instance, random parameters of the transaction signing process and private keys can be utilized to store data. Chen et al. [27] construct a covert channel by embedding data into the private key and random factor fields of a transaction signature, and formally prove the channel's unobservability. Zhang et al. [28] combine Shamir threshold and STC mapping techniques to write data covertly into the amount field of a blockchain transaction. The introduction of Shamir threshold technique increases the channel security, and the STC mapping improves the detection resistance and increases the embedding capacity. However, all the above schemes require the sender to actively create the transaction. This not only increases the economic cost, but also increases the risk of mining the secret data. In contrast, our work simply forwards existing transactions to construct the covert channel, thus circumventing both the economic cost and the data mining risk. The traffic-level covert communication primarily employs blockchain network traffic to construct covert channels. Lv and Cao [29] used ADDR messages of Bitcoin to convey data. Zhu et al. [11] embedded data within inv messages of Bitcoin. However, these proposals may be discernible to adversaries and exhibit a considerably limited bandwidth. In contrast, our work introduces obfuscation and builds a high-bandwidth coding scheme based on complete binary trees to harmonize unobservability and bandwidth.

In this paper, we use MTMM [11] as the baseline due to its superiority, which we justify through the following comparison. Firstly, we added a comparison between blockchain-based covert channels and other notable network-based covert channels, as shown in Table I. This comparison considers the network type, reliability, anonymity, and computational capability of each scheme. Reliability refers to the ability to resist various network attacks, including data tampering, man-in-the-middle (MITM) attacks, and distributed denial-of-service (DDoS) attacks. Studies [12] and [13] construct covert channels in IoT, where communicating parties must facilitate covert communication using IoT devices. These IoT devices, often resource-constrained and lightweight, pose a difficulty for the implementation of sophisticated security measures in covert channels, making the channels challenging to defend against network attacks. Studies [14] and [15] utilize cognitive radio networks (CRN) to achieve covert communications. CRNs can be decentralized, but usually do not have the property of immutability. Therefore, CRN-based covert channels are not resistant to data tampering and the MITM attack. Meanwhile, CRN-based covert channels provide a degree of anonymity by dynamically perceiving and employing the spectrum. Studies [16] and [17] enable covert communication in UAV networks. While UAV networks can be designed with decentralization to defend against the DDoS attack, guaranteeing immutability is often difficult. Covert channels in UAV networks are challenging to resist data tampering and MITM attack, either. In contrast, MTMM and ORIM builds covert channels in blockchain networks. The inherent properties of blockchain network, such as decentralization and immutability, prevent the adversary from disrupting the covert communication through network attacks. In addition,

blockchain's compatibility with anonymization technologies such as the onion router (Tor) facilitates the anonymity of the communicating parties. Moreover, devices involved in blockchain typically have strong computational capability for verifying blockchain transactions. Therefore, blockchain-based covert channels have advantages such as reliability and anonymity that other traffic-based covert channels do not have. These advantages are attributed to blockchain, prompting our focus on covert channels in the blockchain network.

Further, we investigated the blockchain-based covert channels and compare them with ORIM, as depicted in Table II. Blockchain-based covert channels can be divided into two types: ledger-level covert channels and traffic-level covert channels. In the ledger-level covert channel, the sender encodes data in blockchain transaction fields, such as the transaction address and transaction amount. The data-carrying transaction is broadcast and recorded on the blockchain ledger. The receiver retrieves the transaction from the ledger and decodes it to get the data. Studies [24], [27], and [28] are representative of such schemes, where they encode data into the transaction signature, transaction amount, and bitcoin script, respectively. The ledger-level covert channel necessitates the sender to actively create the transaction and set the values of the corresponding transaction fields, with the transaction stored in the blockchain ledger. As a result, these schemes face additional economic costs, the risk of data mining, and the risk of identity traceability. In contrast, the traffic-layer covert channels utilize the communication traffic between blockchain nodes to encode data. The communicating parties are not required to actively create transactions, and the data-carrying traffic is not recorded in the ledger. Consequently, such channels are exempt from economic costs, the risk of data mining, and the risk of identity traceability. To the best of our knowledge, MTMM and [29] are two typical blockchain traffic-layer covert channels. Among them, [29] encodes data into Bitcoin's ADDR messages. However, the Bitcoin protocol restricts the frequency of sending ADDR messages, permitting only one ADDR message to be sent between the same two nodes within 24 hours, making it less practical. MTMM utilizes the inventory message to encode data. The inventory message is employed by nodes to synchronize transaction and block information with each other. This message occurs frequently, making MTMM more practical. Compared to ORIM, MTMM cannot resist the comparison attack. ORIM is thus possess stronger unobservability.

This paper focuses on blockchain-based covert channels due to their reliability and anonymity, better than other network covert channels. While most blockchain-based covert channels encode data into transactions at the ledger layer, these approaches can lead to issues such as high costs, data mining, and identity tracing. MTMM provides a solution that circumvents these problems and attains practicality. Thus, we consider MTMM in the comparison.

III. Attack to MTMM

A. MTMM Review

In MTMM [11], the sender encodes data within the inv message of Bitcoin, which is composed of multiple transaction hashes. Specifically, the sender rearranges the transaction hash sequence in the inv message to encode the secret data. Assume the sender is preparing to transmit an inv message containing n transaction hashes. If the sender is set to transmit bit 1, the smallest (in ASCII order) of the n transaction hashes is selected to be the first transaction hash in the inv message. Conversely, if the sender is set to transmit bit 0, the largest of the n transaction hashes is chosen as the first transaction hash. Subsequently, either the largest or smallest of the remaining $(n-1)$ transaction hashes is selected as the second transaction hash in the inv message, depending on next bit to be transmitted. This process continues until all n transaction hashes are sorted. For the decoding process, the receiver compares two adjacent transaction hashes within the inv message to decode either 0 or 1. If the earlier transaction hash is smaller than the following one, it decodes bit 1; if not, it decodes bit 0. Notably, an inv message containing n transaction hashes can store $(n-1)$ bits of data.

A distinctive pattern occurs regardless of the nature (like freshness and indistinguishability) of the transmitted encrypted data. Given some transaction hashes, if the next transmitted bit is 1, then the MTMM sender chooses the smallest transaction hash to append it to the inventory message, so that this selected transaction hash must be smaller than all subsequent hashes appended to the inventory message. If the next transmitted bit is 0, then the MTMM sender chooses the largest transaction hash to add to the inventory message, so that this selected transaction hash must be larger than all subsequent hashes appended to the inventory message. Thus, regardless of the properties possessed by the transmitted encrypted data, the MTMM traffic containing that data possesses the unique characteristics described above.

The carrier of the data in MTMM is the transaction hash, which can be viewed essentially as a randomized string. MTMM only reorders the transaction hashes without changing the value of the transaction hash or destroying its randomness. Therefore, it is actually difficult to distinguish the coding pattern of MTMM from its statistical characteristics. Experimental findings in MTMM [11] reveal that MTMM traffic resists detection by various data analytics methods, including K-S tests, KLD, and deep learning models. This suggests that commonly used data analytics methods have difficulty in detecting the pattern of MTMM traffic. Moreover, previous studies have failed to identify MTMM's encoding patterns, and we are the first to discover the potential exposure of MTMM traffic through the size relationship between multiple transaction hashes. Besides, a significant merit of MTMM is that it presents, for the first time, a framework for covert communications using Bitcoin inventory messages. This framework manages to circumvent economic costs, data mining risks, and identity tracing risks while ensuring the reliability and anonymity of covert channels. It also provides foundations and insights for subsequent research. Although the MTMM encoding pattern described in this paper may seem apparent, the idea of employing Bitcoin inventory messages to encode data offers a direction for researchers to explore and improve. Subsequent research can further optimize and devolve covert channels based on the MMTM framework.

TABLE II
COMPARISION OF MTMM AND OTHER BLOCKCHAIN-BASED COVERT CHANNELS

Scheme	Category	Free economic cost	No data mining risk	No identity tracing risk	Practicality	Resist comparison attack
[24]	Ledger	×	×	×	✓	/
[27]		×	×	×	✓	/
[28]		×	×	×	✓	/
[29]	Traffic	✓	✓	✓	×	✓
MTMM [11]		✓	✓	✓	✓	×
ORIM		✓	✓	✓	✓	✓

B. Threat Model

In the threat model, we consider an attacker that aims to detect covert channels in the blockchain. The attacker is able to actively collect the traffic in the blockchain network and analyze the traffic using statistical analysis methods such as Kolmogorov-Smirnov (K-S) tests. For example, an attacker is able to deploy blockchain nodes itself and connect to the blockchain network. The attackers can also establish network connections with other blockchain nodes to capture their sending traffic. Attackers generally do not compromise other nodes in the blockchain, and attackers prefer to control other nodes in the blockchain to gain access to their sending traffic and receiving traffic. In summary, the attacker collects the network traffic in the blockchain network and analyze the traffic to identify abnormal traffic carrying data.

We assume that the sender and the receiver share the IP addresses of their blockchain nodes in advance. They can also change the IP address of their blockchain nodes using techniques such as virtual private network (VPN) and inform the new IP address in the next covert communication. We also assume that the cryptographic primitives including PRF and the encryption/decryption algorithm used by the sender and receiver are secure, and an attacker cannot break these cryptographic algorithms.

C. Problem Formalization

A pivotal attribute of covert channels is their ability to evade detection by a warden, a property we classify as **unobservability**. Let us assume that the covert channel ingests normal network traffic and confidential data, and subsequently outputs the traffic imbued with this data. Further, we posit that the warden scrutinizes the network traffic to discern the traffic generated by the covert channel. We attribute **unobservability** to a covert channel if the warden is incapable of distinguishing the normal traffic from the traffic laden with data. The formal definition is shown below.

Definition 1 (Unobservability): Consider $\mathsf{CC} : \mathcal{NT} \times \mathcal{SD} \rightarrow \mathcal{DT}$ as a covert channel, wherein $\mathcal{NT}$ represents the normal traffic, $\mathcal{SD}$ symbolizes the confidential data, and $\mathcal{DT}$ indicates the traffic containing data. We propose the following probabilistic experiment involving a Probabilistic Polynomial-Time (PPT) adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

$\boldsymbol{IndExp}_{\mathsf{CC}}(1^{\lambda})$:

1. $\mathcal{C}$ formulates parameters for the covert channel denoted as CC.
2. $\mathcal{A}$ produces covert data, denoted as $sd \in \mathcal{SD}$.
3. $\mathcal{C}$ constructs normal traffic, denoted as $nt \in \mathcal{NT}$, and generates a uniform bit, $b \in \{0, 1\}$. If $b = 1$, $\mathcal{C}$ executes CC and yields $dt \in \mathcal{DT}$; otherwise, $\mathcal{C}$ produces nt. The output is then provided to $\mathcal{A}$.
4. Upon receiving nt or dt, $\mathcal{A}$ guesses the value of b and outputs a bit b' as his/her guess.

The output of the experiment is defined to be 1 if $b' = b$, and the adversary's advantage for breaking the experiment is defined as:

$$Adv_{\mathcal{A},\mathsf{CC}} = |Pr[\boldsymbol{IndExp}_{\mathsf{CC}}(1^{\lambda}) = 1] - \frac{1}{2}|. \tag{1}$$

We assert that CC possesses **unobservability** if, for all PPT adversaries $\mathcal{A}$, a negligible function $negl$ exists such that

$$|Pr[\boldsymbol{IndExp}_{\mathsf{CC}}(1^{\lambda}) = 1] - \frac{1}{2}| \leq negl(\lambda), \tag{2}$$

where λ signifies the security parameter.

D. Comparison Attack

We argue that an adversary could compromise the unobservability experiment of MTMM, relying on its unique pattern. Consider the following distinguisher, denoted as $\mathcal{D}$, that aims to break the **unobservability** of MTMM:

$\boldsymbol{Distinguisher}\ \mathcal{D}$:

Given an inv message, $\mathcal{D}$ determines whether it is an MTMM message containing secret data. To do this, $\mathcal{D}$ outputs the guess bit b' according to the following rule:

1. For each hash in the message, $\mathcal{D}$ observe whether its value is larger or smaller than all subsequent hashes.
2. If step 1 above is satisfied for all hashes, $\mathcal{D}$ outputs bit $b' = 1$; otherwise it outputs bit $b' = 0$.

Then $\mathcal{D}$'s advantage for breaking the **unobservability** of MTMM is expressed as:

$$\begin{aligned} Adv_{\mathcal{D},MTMM} &= |Pr[b' = b] - \frac{1}{2}| \\ &= |Pr[b' = 1|b = 1] \cdot Pr[b = 1] \\ &\quad + Pr[b' = 0|b = 0] \cdot Pr[b = 0] - \frac{1}{2}| \\ &= \frac{1}{2}|Pr[b'=1|b=1]+Pr[b'=0|b=0]-1|. \end{aligned} \tag{3}$$

According to the output rule of $\mathcal{D}$, $\mathcal{D}$ must output 1 when faced with an MTMM message, and we have

$$Pr[b' = 1|b = 1] = 1. \tag{4}$$

When confronted with a normal inv message, $\mathcal{D}$ outputs 1 if the message complies with the first step as mentioned above. It should be noted that an inv message may comprise

TABLE III
PROBABILITY OF THE NUMBER OF TRANSACTION HASHES IN 1000 INV MESSAGES

i	p_i	i	p_i	i	p_i	i	p_i	i	p_i
1	0.070	2	0.090	3	0.054	4	0.052	5	0.055
6	0.056	7	0.045	8	0.043	9	0.034	10	0.038
11	0.045	12	0.034	13	0.032	14	0.035	15	0.030
16	0.034	17	0.017	18	0.022	19	0.024	20	0.009
21	0.021	22	0.012	23	0.013	24	0.014	25	0.009
26	0.014	27	0.009	28	0.011	29	0.011	30	0.006
31	0.003	32	0.008	33	0.003	34	0.004	35	0.043

TABLE IV
PRACTICAL EFFECTIVENESS OF THE COMPARISON ATTACK

Actual type	Classification result			
	One flow as a group		Five flows as a group	
	Normal	MTMM	Normal	MTMM
Normal	4760	240	1000	0
MTMM	0	5000	0	1000

different numbers of hashes. Assuming the probability of an inv message consisting of i hashes is p_i, we have

$$Pr[b' = 0|b = 0] = \sum_i p_i(1 - \frac{2^{n-1}}{n!}). \quad (5)$$

Combining (4) and (5) gives the advantage:

$$Adv_{\mathcal{D},MTMM} = \frac{1}{2}\sum_i p_i(1 - \frac{2^{n-1}}{n!}). \quad (6)$$

We have collected 1,000 inv messages to compute p_i, and the result is shown in Table III. Based on the result, we calculate that $Adv_{\mathcal{D},MTMM} \approx 0.388$, a non-negligible value. This suggests that MTMM is not secure, indicating that $\mathcal{D}$ holds a significant advantage in distinguishing normal inv messages from MTMM messages. We then evaluate the effectiveness.

E. Effectiveness

We deploy MTMM on a bitcoin node to transmit real data. We use the node to capture 5000 MTMM traffic and 5000 normal traffic. Then the proposed comparison attack is employed to identify the MTMM traffic. We compute the correctly identified MTMM traffic, correctly identified normal traffic, misidentified MTMM traffic, and misidentified normal traffic. Table IV shows the results. It can be seen that the comparison attack is able to accurately identify all MTMM traffic while misclassifying normal traffic as MTMM traffic with a probability of only 240/5000 = 4.8%. This is because when the number of transaction hashes in the inventory message is small, normal traffic may also fulfill the characteristics of MTMM traffic. Practically, to avoid misidentifying the normal traffic as the MTMM traffic, several consecutive flows can be selected as a group for identification. An attacker only considers a group of traffic to be MTMM traffic if all the traffic within the group is identified as the MTMM traffic at the same time. For example, when 5 consecutive flows are considered as a group, the comparison attack is able to distinguish MTMM traffic from normal traffic with 100% accuracy.

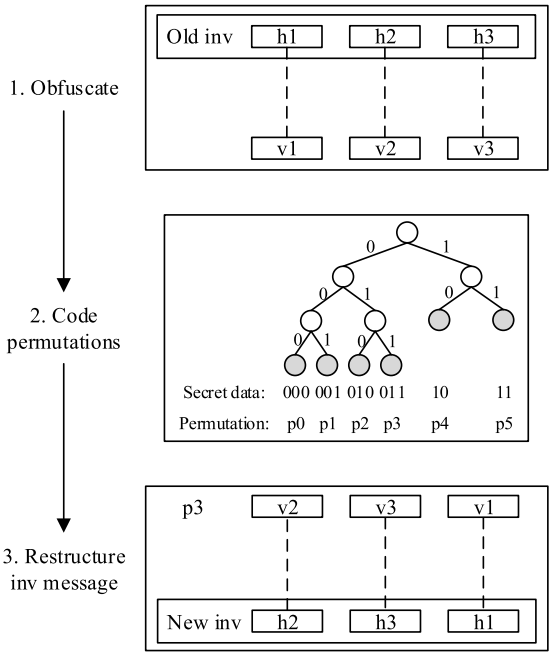


Fig. 1. The overview of encoding.

IV. ORIM: AN IMPROVED SCHEME

A. Overview

ORIM modifies the structure of the inv message to embody the secret data, as depicted in Fig. 1. To maintain the **unobservability** attribute, the sender refrains from directly permuting the hash values in the inv message. Instead, the sender initially obfuscates the hashes, which maps the original hashes to new values. These obfuscated values are subsequently permuted to represent the secret data. The next step involves encoding all permutations of obfuscated values. In particular, the sender encodes each permutation by constructing a complete binary tree, with the number of leaf nodes equivalent to that of all possible permutations. These permutations correspond on a one-to-one basis with the leaf nodes. Ultimately, the sender restructures the inv message based on the secret data to be transmitted and its corresponding permutation. It is important to note that Fig. 1 only illustrates the encoding process. The decoding process is the reverse of the encoding process and thus not shown in the figure.

B. Detailed Design

In this section, we delve into the intricate details of the encoding and decoding processes. It should be noted that the secret data is essentially segmented and transmitted via multiple inv messages, and this section focuses on the technicalities of embedding a segment of secret data within a singular inv message. We consider the encoding and decoding in an inv message containing n transaction hashes.

1) Data Encoding: The process of data encoding encompasses three stages: obfuscation, coding, and restructuring.

a) Obfuscation: During this phase, the sender deterministically maps the original hashes to obfuscated values using a PRF. The permutations of these obfuscated values are

subsequently utilized to represent secret data. This obfuscation process is elucidated in Algorithm 1. Every transaction hash within an inv message undergoes the PRF to yield its corresponding obfuscated value. The PRF key is shared with the receiver prior to enable the receiver to reproduce the obfuscation phase and decode the secret data. Utilizing the PRF ensures no correlation between the obfuscated values and the original hashes, or among the obfuscated values themselves, thereby assisting ORIM in maintaining the **unobservability** attribute.

Algorithm 1 Obfuscation

Input: k // PRF key.
$\mathbf{H}$ // Original transaction hashes $\mathbf{H} = (h_0, h_1, \cdots, h_{n-1})$.
Output: obfuscated value $\mathbf{V} = (v_0, v_1, \cdots, v_{n-1})$.
Initialize $\mathbf{V} = (v_0, v_1, \cdots, v_{n-1})$;
for $h_i \in \mathbf{H}$ **do**
  Compute $v_i = F_k(h_i)$;
end
return $\mathbf{V}$

Note that MTMM requires that the communicating parties pre-share a symmetric key for encryption and decryption, as well as start and end flags for the communication. It does not use PRF and therefore does not require the sharing of PRF keys, but still necessitates a sharing process. This sharing process is identical to that of ORIM. The only disparity is that ORIM additionally necessitates sharing a PRF key. This does not compromise the usability of the covert channel, as the PRF key is only 256 bits and occupies almost no additional resources. Therefore, in comparison to MTMM, the usability of ORIM remains virtually unaltered, yet it offers a superior degree of unobservability.

The use of PRFs is indeed an extra step, while it has a small impact on the covert channel. This is due to the fact that the calculation of PRF is efficient and can usually be done in milliseconds. In addition, experiments (see Section V) evaluate the effect of the extra PRF on the time-related properties (i.e., inter-packet delay, IPD) of the covert channel. The results show that there is no significant difference between the traffic with extra PRF and normal traffic, implying that the IPD for nodes to forward traffic is not affected by PRF. In practical deployment, the sender and receiver do not need to perform PRF operations manually. They implement it through automated procedures and codes. Thus, although PRF weakens the convenience of covert channels, they have minimal impact on practical use.

b) Coding: The coding phase involves the implementation of a complete binary tree-based variable-length coding scheme to encode the secret data, which is represented by permutations of transaction hashes. An inv message containing n transaction hashes corresponds to $n!$ (the factorial of n) structures, each corresponding to a full permutation of these hashes. As such, the sender constructs a complete binary tree with $n!$ leaf nodes to encode all permutations, as illustrated in Fig. 2. Notably, for $2^{m-1} \leq n! \leq 2^m$, the complete binary tree comprises m layers. The $(m-1)$-th layer contains $2^m - n!$ leaf nodes, and the m-th layer accommodates $2n! - 2^m$ leaf nodes, amounting to a total of $n!$ leaf nodes. For the assignment of codewords to these leaf nodes, corresponding to

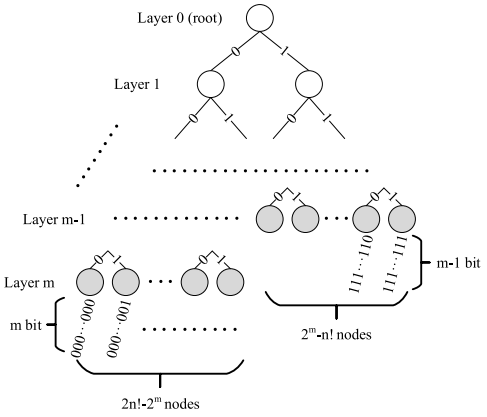


Fig. 2. The complete binary tree corresponding to $n!$ permutations, where $2^{m-1} \leq n! \leq 2^m$.

$n!$ permutations, left branches symbolize 0 and right branches represent 1. The secret data represented by each leaf node is then the concatenation of the branch symbols from the root node to that particular leaf node. For example, the first leaf node of the m-th layer represents an m-bit codeword "000...000", the second leaf node of the m-th layer signifies an m-bit codeword "000...001", and the last leaf node of the $(m-1)$-th layer denotes an $(m-1)$-bit codeword "111...111". Subsequently, we associate each leaf node with a permutation. Specifically, we determine the size order of two permutations and sort all permutations from smallest to largest according to the comparison rule below.

Considering two distinct permutations $\mathbf{P} = (p_1, p_2, \cdots, p_n)$ and $\mathbf{Q} = (q_1, q_2, \cdots, q_n)$, where p_i and q_i are strings, we proceed as follows:

1. Set $i = 1$;
2. If $p_1 > q_1$, then $\mathbf{P} > \mathbf{Q}$;
3. If $p_1 < q_1$, then $\mathbf{P} < \mathbf{Q}$;
4. If $p_1 = q_1$, then set $i = i + 1$ and return to step 2;

Next, we assign each leaf node (in an order from bottom to top and left to right) a one-to-one correspondence to each permutation (from smallest to largest). The first leaf node of layer m corresponds to the smallest permutation, and the last leaf node of layer $m-1$ corresponds to the largest permutation. Consequently, all permutations uniquely correspond to an m-bit or $(m-1)$-bit codeword. The sender can then structure the inv message according to the permutation that corresponds to the transmitted codeword.

c) Restructuring: The restructuring phase, given the transmitted secret data, ascertains the permutation of hashes necessary to structure the inv message. This is achieved in two steps. Initially, the size order of a specific permutation, based on the secret data, is located. Subsequently, the position of each transaction hash is determined, given the size order of a permutation. Let's assume that there are n transaction hashes in the inv message and that the data to be transmitted consists of m bits ($2^{m-1} \leq n! \leq 2^m$), for simplicity's sake. The locating process is delineated in Algorithm 2. The sender initially determines whether the leaf node corresponding to the secret data sd is located in the m-th or $(m-1)$-th layer of the complete binary tree. Notably, there are $2n! - 2^m$ leaf

nodes in layer m. The codeword represented by these leaf nodes commences from 0 and ascends from left to right. Consequently, the codeword range of the leaf nodes in layer m is from 0 to $2n! - 2^m - 1$ in m-bit binary. Similarly, the codeword range of the leaf nodes in layer $m-1$ is from $n! - 2^{m-1}$ to $2^{m-1} - 1$ in $(m-1)$-bit binary. Given this, the sender can determine the amount of data transmitted in this inv message (m-bit or $(m-1)$-bit) and the size order of the corresponding permutation according to the decimal values of the secret data (or that of the first $m-1$ bits of the secret data). If the decimal sd is within the range 0 to $2n! - 2^m - 1$, the inv message transmits m-bit data and the size order of the target permutation is decimal sd. If the decimal $sd[: m-1]$ (the first $m-1$ bits of sd) lies within the range $n! - 2^{m-1}$ to $2^{m-1} - 1$, the inv message transmits $(m-1)$-bit data and the size order of the target permutation is decimal $sd[: m-1] + n! - 2^{m-1}$.

Algorithm 2 Locate the Size Order of the Permutation Based on Transmitted Secret Data

Input: $sd \in \{0,1\}^m$ // Secret data.
n // The number of transaction hashes within an inv message.
Output: The size order of the permutation corresponding to sd.

// If the m-bit sd corresponds to one leaf node of the complete binary tree's m-th layer;
if *sd is less than or equal to* $(2n! - 2^m - 1)$ **then** this inv message transmits m bits of data
  return *demical* sd;
end
// Else the first $m-1$ bits of sd corresponds to one leaf node of the complete binary tree's $(m-1)$-th layer;
else this inv message transmits $m-1$ bits of data, i.e. the first $m-1$ bits of sd
  return *demical* $sd[: m-1] + n! - 2^{m-1}$;
end

After determining the size order of the permutation, the sender then ascertains the position of each transaction hash within that permutation. Let n denote the number of transaction hashes in the inv message and o represent the size order of the target permutation, and the process of position determination is portrayed in Algorithm 3. The sender individually confirms the obfuscated value at each position. Note that upon deciding the first element in a permutation, the size order of that permutation is confined to a specific range. For instance, if the first element is the smallest obfuscated value, then the permutation must have a size order between 0 and $(n-1)! - 1$. If the first element is the second smallest obfuscated value, then the permutation must have a size order between $(n-1)!$ and $2(n-1)! - 1$. Conversely, the size order of the target permutation can determine the first element, i.e., the element within the obfuscated values with a size order of $\lfloor o/(n-1)! \rfloor$. Upon identifying the first element, the sender can similarly discern the second element. At this juncture, the sender's objective becomes finding the first element of a permutation given a size order from the permutations of the remaining $n-1$ obfuscated values, where that given size order is specifically $o\%(n-1-i)!$. By analogy, the sender can ultimately determine the position of each obfuscated value in the permutation, based on which the sender positions the original hashes such that the original hash and the obfuscated hash at the same position correspond to each other.

2) Data Decoding: The data decoding process is outlined in Algorithm 4. Upon receipt of an inv message, the receiver

Algorithm 3 Determine the Position of Each Transaction Hash Given the Size Order of a Permutation

Input: o // The size order of the target permutation.
$\mathbf{V}$ // obfuscated values $\mathbf{V} = (v_0, v_1, \cdots, v_{n-1})$.
$\mathbf{H}$ // Original hashes $\mathbf{H} = (h_0, h_1, \cdots, h_{n-1})$.
n // The length of $\mathbf{V}$.
Output: Reordered hashes $\mathbf{H}'$.

Sort $\mathbf{V}$ from smallest to largest to get $\mathbf{SV} = (sv_0, sv_1, \cdots, sv_{n-1})$;
Initialize ordered obfuscated value list $\mathbf{V}' = []$;
// Determine the i-th element in $\mathbf{V}'$;
for *i ranges from* 0 *to* $n-1$ **do** determine the i-th element in $\mathbf{V}'$
  Compute $index = \lfloor o/(n-1-i)! \rfloor$;
  $\mathbf{V}'.append(sv_{index})$;
  $\mathbf{SV}.remove(sv_{index})$;
  Re-sort and reindex $\mathbf{SV}$;
  Compute $remainder = o\%(n-1-i)!$;
  Set $o = reminder$;
end
Initialize inv message list Inv;
Reorder $\mathbf{H}$ to $\mathbf{H}'$ such that for $j \in [0, n-1]$, $\mathbf{H}'$'s element h'_j corresponds to v'_j.;
return $\mathbf{H}'$;

initially computes the obfuscated values of the transaction hashes, followed by calculating the size order of the previously computed obfuscated values' permutation. As previously noted, the size order of the first element determines a broad range for that of the permutation. The receiver utilizes this to compute a preliminary range for the size order of the permutation. Subsequently, the size order of the second element within the remaining obfuscated values is used to refine the range. Once all elements have been considered in the aforementioned manner, the receiver obtains a precise size order of the computed permutation. Finally, the receiver finds the codeword of the leaf node corresponding to the obtained size order and recovers the secret data. Remember that the first $2n! - 2^m$ leaf nodes' codewords are the binary values from 0 to $2n! - 2^m - 1$ of m bits, respectively, and that the last $2^m - n!$ leaf nodes' codewords are the binary values from $n! - 2^{m-1}$ to $2^{m-1} - 1$ of $(m-1)$ bits, respectively. Thus, the receiver can extract the secret data based on the size order of the permutation. This process is the opposite of encoding and is therefore not described in further detail.

C. Security Proof

Theorem 1: If F is a secure PRF and the hash function used to computed the transaction hash is secure, ORIM possesses **unobservability**.

Proof: The key idea of our proof is to demonstrate that both a original $\mathbf{H}$ and a restructured $\mathbf{H}'$ can be considered as random strings. The original $\mathbf{H}$ is composed of hashes of mutually unrelated transaction data. As long as that hash function is secure, $\mathbf{H}$ can be considered a random string. Next prove that $\mathbf{H}'$ can also be considered random, and we prove this by arguing that any element h_i of $\mathbf{H}$ is at any position of $\mathbf{H}'$ with the same probability $\frac{1}{n}$. **We first consider the probability that a certain h_i is permuted to the first position in $\mathbf{H}'$, which is expressed as $\Pr[h_i \rightarrow h'_0]$.** Assume that $\mathbf{V} = \{v_0, v_1, \cdots v_{n-1}\}$ is the obfuscated values of $\mathbf{H}$, i.e., $v_i = F_k(h_i)$. Recall that the sender permutes $\mathbf{V}$ to get $\mathbf{V}'$ based on the secret data, and then $\mathbf{H}$ is permuted into $\mathbf{H}'$ according to $\mathbf{V}'$ such that $v'_j = F_k(h'_j)$. We use $\Pr[h_i \rightarrow h'_j]$ to denote

Algorithm 4 Data Decoding

Input: k // PRF key.
$\mathbf{H}'$ // Receiving hashes $\mathbf{H}' = (h'_0, h'_1, \cdots, h'_{n-1})$.
n // The length of $\mathbf{H}'$.
Output: Secret data sd.

// First compute the obfucated values;
Initialize obfuscated value list $\mathbf{V}' = (v'_0, v'_1, \cdots, v'_{n-1})$;
for $h'_i \in \mathbf{H}'$ **do**
  Compute $v'_i = F_k(h'_i)$;
end
// Then compute the size order of the obfuscated values' permutation $\mathbf{V}'$;
Initialize $o = 0$;
for i *ranges from* 0 *to* $n-1$ **do**
  Compute o_i which represents the size order of v'_0 among $\mathbf{V}'$;
  Compute $o \mathrel{+}= o_i * (n-1-i)!$;
  $\mathbf{V}'.remove(v'_0)$;
end
// Finally revover secret data;
Initialize sd;
Compute the integer m such that $2m - 1 \leq n! \leq 2^m$;
if $0 \leq o \leq 2n! - 2^m - 1$ **then** the leaf node corresponding to the permutation is at the layer m of the complete binary tree
  Assign sd to the binary form of o;
  Supplement 0 from the high side of sd until sd reaches m bits;
end
else the leaf node corresponding to the permutation is at the layer $m-1$ of the complete binary tree
  Assign sd to the binary form of $2^{m-1} - n! + o$;
  Supplement 0 from the high side of sd until sd reaches $m-1$ bits;
end
return sd;

the probability that h_i is permuted as h'_j in $\mathbf{H}'$. Then

$$\Pr[h_i \to h'_j] = \Pr[v_i \to v'_j]. \tag{7}$$

Now we discuss the size order of v_i in $\mathbf{V}$. We use $\Pr[v_i \to \mathbf{V} : j]$ to denote the probability that v_i is with size order j in $\mathbf{V}$. The PRF F can be regarded as a random function as it is a secure PRF. Therefore, $v_i = F_k(h_i)$ can be considered as a random value. Thus, for arbitrary i and j, we have

$$\Pr[v_i \to \mathbf{V} : j] = \frac{1}{n}. \tag{8}$$

We then discuss the probability that an element with size order j in $\mathbf{V}$ is permuted as the first element v'_0 in $\mathbf{V}'$, which is expressed as $\Pr[\mathbf{V} : j \to v'_0]$. For a certain v_i, let:

$$\begin{cases} \Pr[\mathbf{V} : 0 \to v'_0] = P_0, \\ \Pr[\mathbf{V} : 1 \to v'_0] = P_1, \\ \qquad \cdots \\ \Pr[\mathbf{V} : n-1 \to v'_0] = P_{n-1}. \end{cases} \tag{9}$$

Then $\sum_{j=0}^{n-1} \Pr[\mathbf{V} : j \to v'_0]$ represents the probability that the first element of $\mathbf{V}'$ is one of the elements of $\mathbf{V}$ whose size is ordered from 0 to $(n-1)$. Obviously, we have

$$\sum_{j=0}^{n-1} \Pr[\mathbf{V} : j \to v'_0] = 1. \tag{10}$$

Combining (7), (8), and (10) gives

$$\begin{aligned} \Pr[h_i \to h'_0] &= \Pr[v_i \to v'_0] \\ &= \sum_{j=0}^{n-1} \Pr[v_i \to \mathbf{V} : j] \Pr[\mathbf{V} : j \to v'_0] \\ &= \frac{1}{n} \sum_{j=0}^{n-1} \Pr[\mathbf{V} : j \to v'_0] \\ &= \frac{1}{n}. \end{aligned} \tag{11}$$

The same reasoning leads to that for arbitrary i and j, $\Pr[h_i \to h'_j] = \frac{1}{n}$ holds. This means that the transformation from $\mathbf{H}$ to $\mathbf{H}'$ can be considered completely random. $\mathbf{H}'$ can also be considered as random strings since $\mathbf{H}$ is considered as random strings. The adversary cannnot distinguish between $\mathbf{H}$ and $\mathbf{H}'$ by a negligible advantage. We complete the proof. □

V. Experiments

A. Setup

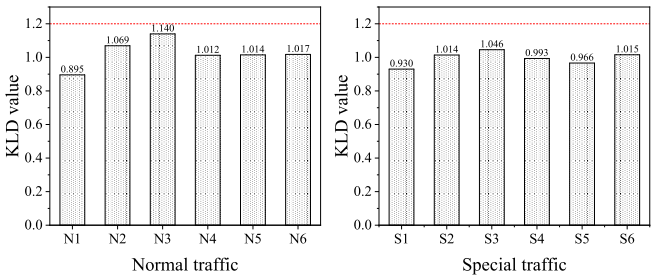
Unless otherwise specified, our experiments are conducted using two Alibaba cloud servers. One server acts as the sender, while the other serves as the receiver. The sender server, designated as S1, employs Ubuntu 20.04 with an Intel(R) Xeon(R) Platinum 8163 CPU @ 2.50GHz and 8GB of memory. In contrast, the receiver server operates on CentOS 7.9 with an Intel(R) Xeon(R) CPU E5-2682 v4 @ 2.50GHz and 8GB of memory. We deploy a prune-mode Bitcoin node on both servers and modify the source code of the sender node to implement ORIM. We utilize Tshark[2] to capture inv traffic on the Bitcoin mainnet.

B. Unobservability

ORIM modifies the transaction hash's ordering rules within the inv message, consequentially altering the construction time of the inv message and the transaction hash's arrangement within it. Consequently, we assess unobservability via the IPD and the hash content between inv messages and the transaction hash sequence within inv messages. Traditional statistical tests for measuring the dissimilarity between two probability distributions, such as the KLD, K-S test, Wasserstein distance, and cosine similarity, as well as AI classification models can be used to assess unobservability. We choose KLD and K-S tests as the criteria because they can show the difference between two distributions more intuitively. Moreover, state-of-the-art research related to covert communication [27], [30], [31] also evaluate unobservability using KLD and K-S tests. We thus adopt the same criteria to ensure fair evaluations. Meanwhile, we use deep learning models as a criterion because deep learning models tend to be able to extract intuitive features that can be used to classify the sample. Particularly, we choose the CTR model as the evaluation model because it is the more successful achievement in blockchain steganalysis related proposals, and state-of-the-art schemes like [11] also use it to evaluate unobservability.

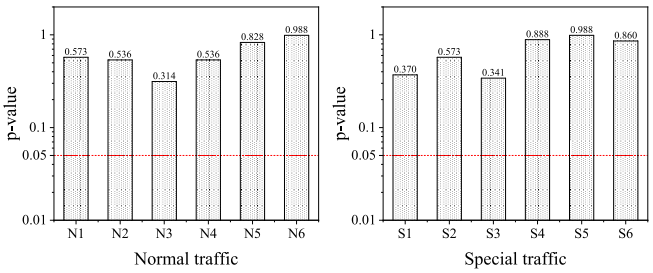
1) IPD: We demonstrate that the IPD of ORIM traffic is not significantly different from that of normal traffic through KLD, K-S tests, and deep learning models.

[2]https://www.wireshark.org/



(a) KLD between normal inv flows. (b) KLD between the normal inv flow and the special inv flow.

Fig. 3. The KLD result.



(a) The K-S test between normal inv flows. (b) The K-S test between the normal inv flow and the special inv flow.

Fig. 4. The K-S test result.

a) We first measure the difference between normal IPDs and special IPDs using KLD: KLD typically signifies the distance between two variables. A smaller KLD value suggests a higher degree of similarity between the two variables. We argue that normal and special IPDs are indistinguishable by demonstrating that the KLD between them mirrors that between normal IPDs. To do this, we capture 7 normal inv flows and compute their IPDs, each comprising 10,000 inv messages. One of these flows serves as a baseline, with the KLD calculated relative to the remaining 6 flows. The outcomes are presented in Fig. 3(a). It is observable that there exist slight differences within the IPDs of normal traffic itself, with their KLD values generally falling below 1.2. Consequently, we set 1.2 as the upper limit for the KLD threshold between normal flows. We proceed to capture 6 special inv flows, each also composed of 10,000 inv messages. Subsequently, the KLD values between the IPDs of these 6 special flows and that of the baseline normal flow are computed, with the results displayed in Fig. 3(b). As illustrated, the KLD values between the special traffic IPDs and the normal traffic IPDs fall below 1.2 as well, indicating no discernible difference in IPD between the special and normal traffic. Note that the 6 normal inv traffic flows and the special inv traffic flow must originate from the same time period. This eliminates the impact of the number of pending broadcast transactions in different time periods on the rate of inv traffic generated by a Bitcoin node. The number of blockchain transactions awaiting broadcast may fluctuate across different time periods, resulting in variations in the frequency at which Bitcoin nodes broadcast transactions.

b) We follow with experiments using the K-S test to measure the difference between the IPD distribution of normal traffic and the IPD distribution of special traffic: The K-S test is used to measure whether two distributions originate from the same distribution. Typically, when the p-value (the output of the K-S test) exceeds 0.05, the two distributions are considered to derive from the same distribution. We employ the K-S test to demonstrate that the IPD distribution of the special traffic aligns with the IPD distribution of the normal traffic. To accomplish this, we capture 7 normal flows (with one serving as a baseline) and 6 special flows, each encompassing 10,000 inv messages. We subsequently compute the IPD of these 13 flows separately to form a dataset. We then show that the IPD distribution between normal flows can be considered consistent. Fig. 4(a) depicts the K-S test results for the IPD distribution of the baseline traffic with respect to the IPD distribution of other normal traffic. All p-values exceed 0.05, suggesting that the normal flow IPDs adhere to the same distribution. Hence, to maintain unobservability, the special flow IPD distribution should also be consistent with the normal flow IPD distribution. Fig. 4(b) depicts the K-S test results for the 6 special flow IPD distributions relative to the baseline normal flow IPD distribution. It can be observed that all p-values are greater than 0.05, indicating that the IPD distributions of special flows align with those of normal flows.

c) We finally evaluate the unobservability using the deep learning model: CTR [32] is a Text-CNN-based blockchain steganography recognition model. We employ CTR to distinguish between the IPD of ORIM traffic and that of normal traffic, using the accuracy as a criterion for evaluating unobservability. The closer the accuracy rate is to 0.5, the greater the degree of unobservability. We capture 3 normal inv flows and 1 ORIM flow, each comprising 10,000 inv messages. Following this, we compute the IPDs of these 4 flows, and separately compose the IPD of the 3 normal inv flows and the IPD of the ORIM flow to generate 3 datasets (D1, D2, and D3), respectively. Each dataset consists of 10,000 IPDs from the ORIM flow and 10,000 IPDs from a normal inv flow, yielding a total of 20,000 IPDs. For each dataset, a portion of the data is used as input to CTR, with the volume of input data for each experiment being 5,000, 10,000, 15,000, and 20,000, respectively. Adhering to the model setup in [32], we conduct 10 experiments to obtain an average accuracy. The results are presented in Table V. The experimental outcomes consistently approach 0.5, suggesting that ORIM can effectively resist the CTR model in terms of IPD. Furthermore, as the volume of input data augments, the model's accuracy in identifying ORIM traffic approaches 0.5. This trend can be attributed to the fact that insufficient data introduces a certain degree of chance. As the sample size expands, the sampling becomes more adequate and comprehensive, thereby increasing the model's difficulty in identifying ORIM traffic. This phenomenon further corroborates that ORIM possesses the same characteristics as normal traffic in terms of IPD.

2) Txhash: In Section IV-C, we provide a theoretical proof that the transaction hash sequence (TxHash sequence) within an ORIM message is indistinguishable from that within a normal inv message. We further substantiate, using CTR, that an

TABLE V

RECOGNITION RESULTS OF CTR. THE POSITIVE AND NEGATIVE SAMPLES ARE WITH A RATIO OF 1:1, AND THE DATASET IS DIVIDED INTO A TRAINING SET AND A TEST SET WITH A RATIO OF 7:3

Input data size	D1	D2	D3
5,000	0.516	0.456	0.473
10,000	0.484	0.491	0.503
15,000	0.482	0.501	0.498
20,000	0.489	0.502	0.499

TABLE VI

THE RECOGNITION RESULTS FOR TXHASH

Input data size	O1	O2	O3
5,000	0.488	0.508	0.521
10,000	0.502	0.496	0.518
15,000	0.491	0.503	0.509
20,000	0.500	0.500	0.501

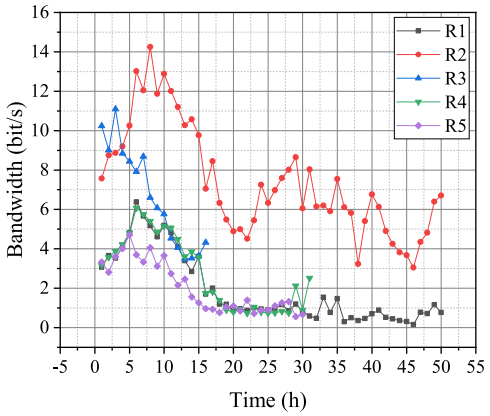


Fig. 5. Hourly bandwidth of the same sender node to different receiver nodes.

adversary cannot discern between ORIM messages and normal inv messages based on the TxHash sequence. Specifically, we capture 1 ORIM traffic and 3 standard traffic flows (each comprising 10,000 inv messages) transmitted by the sender node S1 during the same time period, and extract the TxHash sequences of these inv messages. Subsequently, we separately combine the TxHash sequences of the 3 normal traffic flows and the ORIM traffic flow to generate 3 ORIM datasets, namely O1, O2, and O3 (each consisting of 20,000 TxHash sequences). Each ORIM dataset comprises 10,000 ORIM TxHash sequences and 10,000 standard TxHash sequences. For each ORIM dataset, we randomly select 5,000, 10,000, 15,000, and 20,000 TxHash sequences and input them into the CTR model to calculate the accuracy, respectively. Table VI displays the recognition results of CTR for TxHash, with the experimental setting mirroring that of Table V. The accuracy consistently approaches 0.5, implying that ORIM's TxHash sequence is indistinguishable from the normal TxHash sequence. This is because that after the obfuscation of PRF, a single TxHash can be considered as a random string, with the original TxHash also exhibiting randomness. Additionally, the obfuscation process masks the correlation between obfuscated TxHashes, rendering each ORIM TxHash within a TxHash sequence as unrelated to each other. Consequently, the ORIM hash sequence can be treated as a long random string, aligning with the randomness of the original TxHash sequence. In conclusion, ORIM possesses unobservability in terms of TxHash.

C. Bandwidth

In this section, we investigate the bandwidth of the proposed scheme, employing the number of bits that can be transmitted per second as a performance metric. The metric is computed using the formula $Bandwidth = \frac{B}{T}$, where T represents the time interval from the moment the sender initiates the transmission of the first bit to the instant the receiver acquires the final bit of data, and B denotes the total amount of data transmitted within the time period T.

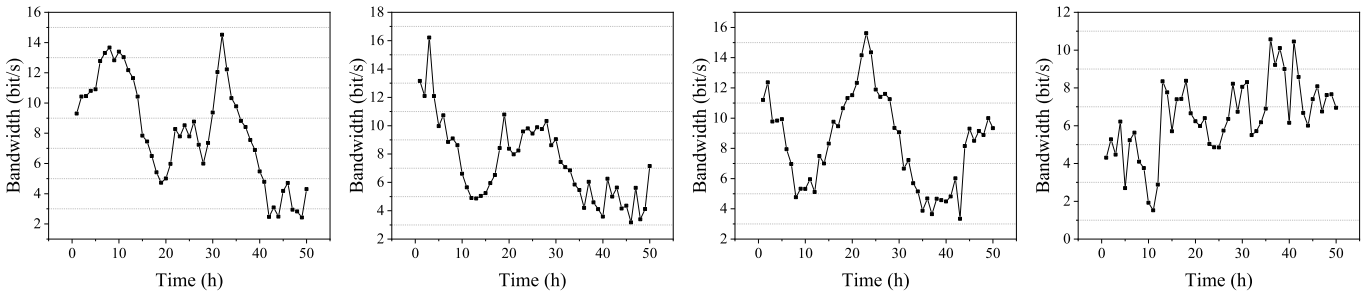
1) We Begin With an Initial Exploration of ORIM's Bandwidth: We implement ORIM on the sender's server, enabling it to broadcast ORIM inv messages to all neighboring nodes. Utilizing Tshark, we capture all outgoing inv messages from the sender's node over 50 consecutive hours and group these inv messages by the IP address of the neighboring node. We select five neighboring nodes (identified as R1, R2, R3, R4, and R5) and compute the bandwidth using the grouped inv messages. Fig. 5 illustrates the bandwidth variation of these five nodes on an hourly basis. R3 maintains bandwidth for the initial 16 hours, while R4 and R5 retain bandwidth for the first 31 hours. This is because these three nodes disconnecting from the sender node midway through the process. Additionally, the bandwidth of R2 significantly exceeds that of the other nodes. One plausible explanation for this is that R2 is connected to fewer neighboring nodes compared to the other receiver nodes. The bandwidth is directly related to the number of inv messages that the sender transmits to the receiver. The sender node forwards the received inv messages to its neighboring nodes, excluding the inv message source node. For instance, if node A is connected to nodes B, C, and D, and B broadcasts an inv message to A, A only broadcasts the received inv message to the other neighboring nodes, i.e., C and D. Therefore, if R2 is connected to fewer neighboring nodes than R1, R2's neighboring nodes may broadcast fewer inv messages to R2. Consequently, fewer inv messages are forwarded by R2 to the sender node, and more inv messages are forwarded by the sender node to R2, leading to higher bandwidth. Lastly, it can be seen that the trend of bandwidth variation each hour remains consistent. The reason is that the bandwidth trend is related to the rate of transaction generation within the Bitcoin network. The higher the number of Bitcoin transactions within a given time frame, the more transactions need to be broadcasted via inv messages. Hence, the trend in bandwidth should align with the trend in the rate of transaction generation.

2) We Further Conduct Experiments to Verify the Above Conjecture on the Number of Nodes Connected by the Receiver and the Rate of Transaction Generation: To achieve this, we alter the configuration of the receiver node to connect solely to the sender node, or in addition to other nodes. We then capture inv messages sent to the receiver node over 50 consecutive hours at varying time periods to compute the bandwidth. The time details and node configuration pertaining

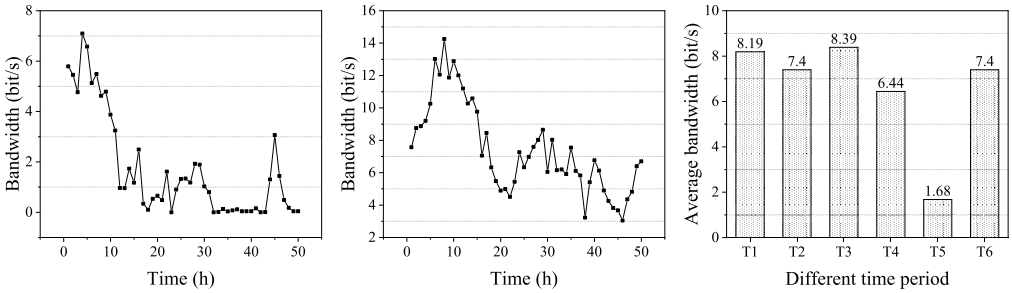
TABLE VII

THE TIME AND NODE CONFIGURATION INFORMATION. THE **CONNECT** CONFIGURATION CAUSES THE NODE TO CONNECT ONLY TO THE SPECIFIED NODE. THE **ADDNODE** CONFIGURATION CAUSES THE NODE TO CONNECT TO THE TARGET NODE WHILE CONSTANTLY ASKING THE TARGET NODE'S NEIGHBORING NODES AND INITIATING CONNECTIONS TO THEM. ALL TIMES ARE IN UTC-4 FORMAT

Number	Start time	End time	Node configuration	Average bandwidth (bit/s)
T1	Thursday, July 6, 2023 4:26	Saturday, July 8, 2023 6:26	Only **connect** to the sender node	8.19
T2	Friday, July 21, 2023 10:57	Sunday, July 23, 2023 12:57	Only **connect** to the sender node	7.40
T3	Thursday, July 27, 2023 13:21	Saturday, July 29, 2023 15:21	Only **connect** to the sender node	8.39
T4	Sunday, July 30, 2023 0:21	Tuesday, August 1, 2023 2:21	Only **connect** to the sender node	6.44
T5	Tuesday, August 1, 2023 9:51	Thursday, August 3, 2023 11:51	**Addnode** to several nodes	1.68
T6	Friday, September 22, 2023 3:07	Sunday, September 24, 2023 5:07	Only **connect** to the sender node	7.40



(a) Bandwidth for T1. (b) Bandwidth for T2. (c) Bandwidth for T3. (d) Bandwidth for T4.



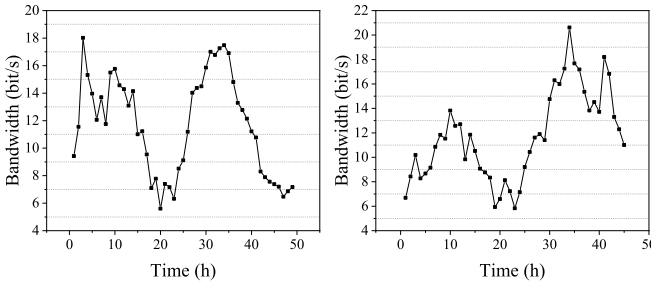
(e) Bandwidth for T5. (f) Bandwidth for T6. (g) Average bandwidth.

Fig. 6. Bandwidth of the same receiver node in different configurations and at different time periods.

to the captured inv messages are outlined in Table VII. We calculate the hourly bandwidth based on the captured inv messages and depict its variation curve in Fig. 6. Additionally, Fig. 6(g) displays the average bandwidth of T1-T6 over the duration of 50 hours. **On one side,** the bandwidth of T5 in Fig. 6(e) is considerably lower than the other instances, suggesting that a large number of nodes connected by the receiver can indeed diminish the bandwidth. T5's bandwidth curve peak barely exceeds 7 bit/s. The bandwidth trend for T5 also differs markedly from the other instances: there is some initial bandwidth, which eventually dwindens to almost zero. This is because it takes time for the receiver node to discover and connect to new neighboring nodes. Initially, the number of neighboring nodes is lesser for a period of time, but as time progresses, the number of receiver's neighboring nodes increases, causing the bandwidth to asymptotically approach zero. **On the other side,** the ascending and descending trends of the subfigures in Fig. 6, excluding T5, recur on a 24-hour cycle. This cyclical repetition is particularly evident in Fig. 6(a), Fig. 6(c), and Fig. 6(d). Fig. 6(a) depicts two cycles of rising and falling trends, with peaks at the 9th and 33rd hours, respectively. Combined with the start time of T1 in Table VII, the peaks consistently occur around 13:00 p.m. in UTC-4 time. Fig. 6(c) illustrates two cycles of decreasing and then increasing trends, with the peak occurring at the 23rd hour, which also corresponds to 13:00 p.m. in UTC-4 time. Each cycle in Fig. 6(d) contains a pattern of rising, falling, rising, and falling trends, with the peaks occurring at the second rise (13th and 36th hours, respectively), which aligns with 13:00-14:00 p.m. in UTC-4 time. Although the trends in Fig. 6(b) and Fig. 6(f) also repeat in two cycles, the bandwidth in the second cycle essentially equals the lowest bandwidth in the first cycle, resulting in a less pronounced cycle repetition. This can be attributed to the data capture occurring on both Friday and Saturday, with the transaction generation rate on Saturday being significantly lower than that on Friday, resulting in a less distinct second cycle peak. In light of these observations, it is rational to infer that bandwidth does indeed correlate with the rate of transaction generation. Furthermore, the volume of transactions generated between 13:00 and 14:00 p.m. is the highest during the day. During the week, the number of transactions is relatively lower on Saturday and Sunday. Fig. 6(g) demonstrates that T1 and T3 exhibit higher bandwidth compared to T2, T4, and T6. This is also because the traffic capturing time in T1 and T3 does not include Saturday and Sunday.

TABLE VIII
THE CAPTURED TIME AND BANDWIDTH OF S2. ALL TIMES ARE IN UTC-4 FORMAT

Number	Start time	End time	Average bandwidth (bit/s)
T7	Thursday, October 19, 2023 2:15	Saturday, October 21, 2023 3:15	11.68
T8	Sunday, October 22, 2023 2:23	Monday, October 23, 2023 23:23	11.79



(a) Bandwidth for T7. (b) Bandwidth for T8.

Fig. 7. Bandwidth of S2 at different time periods.

3) Proper Configuration of the Sender Node Can Also Further Increase the Bandwidth: The bandwidth is fundamentally dictated by the quantity of inv messages that the sender node broadcasts to the receiver node. Thus, it is plausible to increase the bandwidth not only by having the receiver connect solely to the sender node but also by enabling the sender to connect to more nodes that broadcast newly generated transactions first (e.g., nodes providing API functionality). This configuration allows the sender node to receive as many inv messages as possible within a short time frame. To validate this, we deploy another sender node, referred to as S2, on Ubuntu 20.04 with an Intel(R) Xeon(R) Platinum 8269CY CPU @ 2.50GHz and 8GB Memory. We modify its connection configuration and capture the inv messages it broadcasts to the receiver node over two different time periods. The messages for each time period, along with the average bandwidth, are presented in Table VIII. The average bandwidth of S2 exceeds that of S1, thereby substantiating that the appropriate configuration of the sender node can indeed enhance the bandwidth. Fig. 7 additionally illustrates the hourly bandwidth trend for T7 and T8. Mirroring the bandwidth trend for S1, the S2 bandwidth trend also exhibits two distinct cycles, with the peak occurring between 13:00 and 14:00 p.m. in UTC-4 time. Notably, the curve representing the Sunday data in Fig. 7(b) has a lower bandwidth, further reinforcing our prior discussion.

4) Bandwidth Comparison: Finally, we contrast the bandwidth of the proposed scheme with that of MTMM. To calculate the bandwidth of ORIM, we utilize the data from Fig. 6(c) and Fig. 7(b). To circumvent the impact of time and node configuration, we do not recapture the MTMM traffic, instead, we estimate the corresponding MTMM bandwidth based on the ORIM traffic. In MTMM, an inv message containing n transaction hashes can convey $(n-1)$-bit data. The comparison results presented in Fig. 8 reveal that the bandwidth of ORIM significantly surpasses that of MTMM. An inv message holding n transaction hashes can convey at least $\lfloor \log_2 n! \rfloor$-bit data in ORIM, in contrast to MTMM where it can only convey $(n-1)$ bits of data. Fig. 8(c) displays the specific bandwidth values. The bandwidth of the proposed scheme is nearly 3.7 $\times$ of MTMM under various configurations.

TABLE IX
COST AND UNOBSERVABILITY COMPARISON

Scheme	Fees (USD/KB)	Unobservability
STCM [28]	317.24	✓
RDSAC [27]	35.39	✓
DDSAC [27]	70.78	✓
EBDL [24]	14.16	✓
MTMM [11]	0	×
ORIM	0	✓

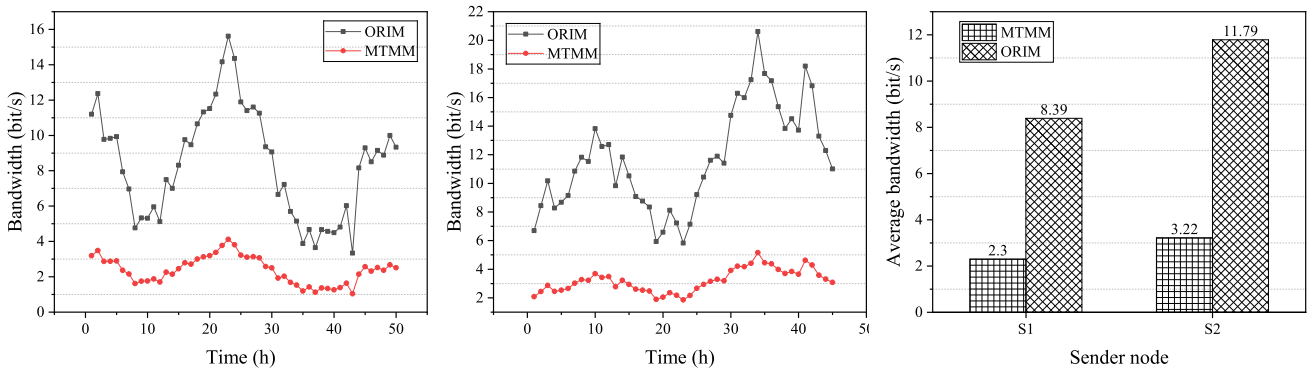
The bandwidth of ORIM depends entirely on the transaction generation rate of the blockchain network. The faster the blockchain network generates transactions, then the bandwidth of ORIM increases. For example, Ethereum generates an average of 14.8 transactions per second[3] and Polygan generates an average of 51.4 transactions per second,[4] compared to Bitcoin which generates an average of 3.57 transactions per second,[5] the implementation of ORIM on Ethereum and Polygan is able to increase the bandwidth up to 4.1$\times$ and 14.4$\times$, respectively. For scenarios with a high volume of covert communication, a combination of ORIM and other high-bandwidth transmission protocols (e.g., the inter planetary file system, IPFS) can be considered to accomplish covert communication together. ORIM can be used to transmit the key and other protocols can be used to transmit the ciphertext to meet scenarios with high-volume covert communication.

D. Economic Cost

We use the average fee of \$1.106 per Bitcoin transaction as of August 25, 2023 [33] to calculate the transaction fee required to transfer 1-KB data, and the results are shown in Table IX. Even the cheapest scheme costs close to \$14.16 to transmit 1KB of data. In contrast, both ORIM and MTMM enable data transmission for free. This is because both of them do not require the sender to create transactions, but rather only maintain blockchain nodes to forward already existing transactions. MTMM is detected by the proposed comparison attack and does not possess unobservability. Whereas ORIM achieves both zero cost and unobservability.

The proposed scheme requires the sender and receiver to maintain a prune mode-bitcoin node, respectively. The node occupies about 5 GB of storage space and run normal bitcoin node functions such as forwarding transactions (note that the mining function is not necessary as the communicating parties rely on propagation of transactions alone to complete the covert communication). The receiver needs to know the

[3]https://etherscan.io/
[4]https://polygonscan.com/
[5]https://www.blockchain.com/explorer



(a) Bandwidth comparison for S1. (b) Bandwidth comparison for S2. (c) Average bandwidth comparison.

Fig. 8. Bandwidth comparison.

network IP address of the sender node and connect to the sender node to become its neighbor node. When the sender starts transmitting data, the receiver receives the data-carrying inventory message in just a few milliseconds (depending on its network latency) as there is a direct network connection between the receiver and the sender's nodes. The receiver can then decode the received inventory messages and obtain the transmitted data at a rate of 11.79 bits per second. These data-carrying inventory messages exist only for a short period of time, and an attacker would have to incur significant costs to mine the secret data. Specifically, the attacker has to eavesdrop and record all traffic across the Bitcoin network in real time. Although the Bitcoin P2P protocol transmits plaintext data that provides traversal for an attacker to eavesdrop the traffic, the sheer size of the network makes it virtually impossible for an attacker to eavesdrop all traffic across the entire Bitcoin network, not to mention the fact that some nodes may be using techniques such as VPN and the onion router (Tor) to protect their communications.

VI. Discussion

A. Practicality

Implementing covert communication in a blockchain network does result in relatively low bandwidth, but the immutable nature of the blockchain network gives the covert channel a very high degree of reliability. In the real world, scenarios that require lower communication bandwidth and higher reliability prefer covert communications in blockchain networks. Here are some concrete examples.

- *Military confidential data transmission.* In military applications, identity-sensitive personnel such as spies and undercovers are usually prohibited from communicating outside their country. They can pass messages to their organization through blockchain-based covert communications, where they appear just normally to maintain a blockchain node. The communication content is usually just a few words like the enemy's next move. Organizations can also pass instructions in this way. In this scenario, the content transmitted is typically short, and the key is to get the message across reliably and covertly.
- *Building anti-censorship links.* When users use technologies such as VPN to evade censorship, there may be instances where the private link is compromised, rendering the user unable to contact the VPN proxy. At this point, they can utilize blockchain-based covert communications to transmit information such as new link parameters. The information is with small size, and can help build new anti-censorship links. In this case, it is crucial to ensure that the covert channel is not discovered and corrupted.
- *Implementing private key disclosure attacks to disrupt the blockchain community.* Attackers may utilize blockchain-based covert communications to implement covert channel attacks to compromise blockchain users' private keys. For example, an attacker releases the Bitcoin source code with backdoors through means such as social engineering and a fake official website. The source code with backdoors is actually the sender program of the blockchain-based covert communications. It reads the local user's Bitcoin private key and leaks the private key through covert communications. An attacker is able to obtain the IP address of the victim user and connect to the backdoor node to steal the Bitcoin private key and the digital currency. In this scenario, transmitting a very small amount of private key data can lead to great harm.

In addition, communicating parties can utilize covert channels to transmit a private link or uniform resource locator (URL) that points to a large amount of data, thereby increasing communication bandwidth and further increasing utility. Besides, the bandwidth of the proposed scheme is determined by the transaction rate. Implementing covert communications in more efficient blockchains such as Ethereum and Polygan can also increase the communication bandwidth.

B. Alternatives and Implications

In ORIM, we utilize a PRF to guarantee unobservability. In this section, we discuss other alternative methods for guaranteeing unobservability and their implications.

If networks outside the blockchain (like IoT) are considered, there are alternative ways to make traffic indistinguishable to attackers. For example, the sender can use deep learning

models to extract features of normal traffic and construct traffic based on these features. However, this method usually requires self-defined traffic data and thus is not applicable to blockchain traffic. Blockchain traffic data mainly consists of the output of cryptographic primitives such as transaction hashes and digital signatures, which are not self-definable and can only be derived by computation. Customized data cannot be validated by blockchain nodes. Another alternative way is to encrypt data at the network layer with technologies such as transport layer security (TLS) and secure sockets layer (SSL) protocols, which make encrypted data random and indistinguishable from each other. However, this is not applicable to blockchain networks, either, as communication between blockchain nodes takes place in plaintext by default.

In blockchain networks, to the best of our knowledge, there are few alternative ways to make traffic of network-layer covert channels indistinguishable to attackers. Comparatively, there are alternative ways to make transactions of ledger-layer covert channels indistinguishable to attackers. In these alternative ways, the sender creates blockchain transactions, encodes data into the transaction fields, and finally broadcasts the transactions to the blockchain ledger. Although these alternative ways guarantee unobservability, they require the sender to pay extra fees to create transactions, incurring additional economic costs. Furthermore, these data-carrying transactions are publicly stored in the blockchain ledger, and the encryption algorithms may be cracked by advanced techniques such as quantum computing, where an adversary may mine the transmitted data.

As for implementing blockchain-based covert communications through other media and encryption schemes, this is actually what existing ledger-layer blockchain-based covert channels do. In such channels, the sender uses encryption schemes to encrypt the data and writes the cipher text into the blockchain transaction field, such as the script and bytecode. However, data generated by encryption schemes is generally considered random, whereas Bitcoin (or Ethereum) scripts may not be random and have certain distributional properties. This allows an adversary to easily discover covert communications by using statistical tools to analyze whether a script satisfies randomness, which is contrary to the unobservable requirement of covert channels. Besides, such a kind of covert channel incurs economic costs and data mining risks.

Our approach which additionally requires a secret key and a PRF is indeed simple and effective. First, the method does not lead to an extra communication process, since the communicating parties would have already needed to negotiate several secret parameters including start and end flags. Second, the computation of PRF is very efficient and can be done within milliseconds, with little impact on the performance of the covert channel. Then, our method has sufficient theoretical and experimental support to guarantee unobservability. Finally, our approach is also applicable to other covert channels that encode data into blockchain traffic.

VII. Conclusion

In this paper, we propose a comparison attack to break MTMM. We give a formal definition of unobservability and detail the attack process under the definition. To defend the proposed attack, we present an obfuscation method to ensure the unobservable nature. Moreover, to enhance the bandwidth, we employ permutations of transaction hashes within inv messages to construct the covert channel, alongside a complete binary tree-based variable length coding scheme to efficiently encode secret data. Experimental results show that ORIM exhibits unobservability and provides much higher bandwidth.

References

[1] L. Lv, Z. Li, H. Ding, N. Al-Dhahir, and J. Chen, "Achieving covert wireless communication with a multi-antenna relay," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 760–773, 2022.

[2] P. Peng and E. Soljanin, "Covert, low-delay, coded message passing in mobile (IoT) networks," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 599–611, 2022.

[3] Y. Jiang, L. Wang, and H.-H. Chen, "Covert communications with randomly distributed adversaries in wireless energy harvesting enabled D2D underlaying cellular networks," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 5401–5415, 2023.

[4] X. Wang et al., "Detection of thermal covert channel attacks based on classification of components of the thermal signal features," *IEEE Trans. Comput.*, vol. 72, no. 4, pp. 971–983, Apr. 2023.

[5] J. Tian, G. Xiong, Z. Li, and G. Gou, "A survey of key technologies for constructing network covert channel," *Secur. Commun. Netw.*, vol. 2020, pp. 1–20, Aug. 2020.

[6] T. Zhang, B. Li, Y. Zhu, T. Han, and Q. Wu, "Covert channels in blockchain and blockchain based covert communication: Overview, state-of-the-art, and future directions," *Comput. Commun.*, vol. 205, pp. 136–146, May 2023.

[7] S. Ma et al., "Robust beamforming design for covert communications," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 3026–3038, 2021.

[8] W. Mazurczyk, S. Wendzel, M. Chourib, and J. Keller, "Countering adaptive network covert communication with dynamic wardens," *Future Gener. Comput. Syst.*, vol. 94, pp. 712–725, May 2019.

[9] Z. Chen et al., "Blockchain meets covert communication: A survey," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 4, pp. 2163–2192, 4th Quart., 2022.

[10] H. Yousaf, G. Kappos, and S. Meiklejohn, "Tracing transactions across cryptocurrency ledgers," in *Proc. USENIX Secur. Symp.*, 2019, pp. 837–850.

[11] L. Zhu, Q. Liu, Z. Chen, C. Zhang, F. Gao, and Z. Yang, "A novel covert timing channel based on Bitcoin messages," *IEEE Trans. Comput.*, vol. 72, no. 10, pp. 2913–2924, Oct. 2023.

[12] D. Wang, Q. Fu, J. Si, N. Zhang, and Z. Li, "Improper Gaussian signaling based covert wireless communication in IoT networks," in *Proc. IEEE Global Commun. Conf.*, May 2021, pp. 1–6.

[13] S. Feng, X. Lu, S. Sun, and D. Niyato, "Mean-field artificial noise assistance and uplink power control in covert IoT systems," *IEEE Trans. Wireless Commun.*, vol. 21, no. 9, pp. 7358–7373, Sep. 2022.

[14] R. Xu, D. Guo, B. Zhang, and G. Ding, "Finite blocklength covert communications in interweave cognitive radio networks," *IEEE Commun. Lett.*, vol. 26, no. 9, pp. 1989–1993, Sep. 2022.

[15] X. Lu, S. Yan, W. Yang, C. Liu, and D. W. K. Ng, "Short-packet covert communication in interweave cognitive radio networks," *IEEE Trans. Veh. Technol.*, vol. 72, no. 2, pp. 2649–2654, Feb. 2023.

[16] S. Yan, S. V. Hanly, and I. B. Collings, "Optimal transmit power and flying location for UAV covert wireless communications," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 11, pp. 3321–3333, Nov. 2021.

[17] C. Wang et al., "Covert communication assisted by UAV-IRS," *IEEE Trans. Commun.*, vol. 71, no. 1, pp. 357–369, Jan. 2023.

[18] X. Chen et al., "Covert communications: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 2, pp. 1173–1198, 2nd Quart., 2023.

[19] J. Partala, "Provably secure covert communication on blockchain," *Cryptography*, vol. 2, no. 3, p. 18, Aug. 2018.

[20] Z. Zhang, S. Wang, Z. Li, F. Gao, and H. Wang, "A multi-dimensional covert transaction recognition scheme for blockchain," *Mathematics*, vol. 11, no. 4, p. 1015, Feb. 2023.

[21] J. Liu et al., "DLCCB: A dynamic labeling based covert communication method on blockchain," in *Proc. Int. Wireless Commun. Mobile Comput. (IWCMC)*, Jun. 2023, pp. 168–173.

[22] B. Du, D. He, M. Luo, C. Peng, and Q. Feng, "The applications of blockchain in the covert communication," *Wireless Commun. Mobile Comput.*, vol. 2022, pp. 1–18, Jun. 2022.
[23] X. Luo, P. Zhang, M. Zhang, H. Li, and Q. Cheng, "A novel covert communication method based on Bitcoin transaction," *IEEE Trans. Ind. Informat.*, vol. 18, no. 4, pp. 2830–2839, Apr. 2022.
[24] C. Zhang, L. Zhu, C. Xu, Z. Zhang, and R. Lu, "EBDL: Effective blockchain-based covert storage channel with dynamic labels," *J. Netw. Comput. Appl.*, vol. 210, Jan. 2023, Art. no. 103541.
[25] A. Biryukov, D. Feher, and G. Vitto, "Privacy aspects and subliminal channels in zcash," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 1813–1830.
[26] H. Cao et al., "Chain-based covert data embedding schemes in blockchain," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 14699–14707, Aug. 2022.
[27] Z. Chen, L. Zhu, P. Jiang, C. Zhang, F. Gao, and F. Guo, "Exploring unobservable blockchain-based covert channel for censorship-resistant systems," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 3380–3394, 2024.
[28] P. Zhang, Q. Cheng, M. Zhang, and X. Luo, "A blockchain-based secure covert communication method via Shamir threshold and STC mapping," *IEEE Trans. Dependable Secure Comput.*, vol. 21, no. 5, pp. 4469–4480, Oct. 2024.
[29] J. Lv and X. Cao, "Covert communication technology based on Bitcoin," *J. Cyber Secur.*, vol. 6, no. 2, pp. 143–152, 2021.
[30] B. Hahn, R. Nithyanand, P. Gill, and R. Johnson, "Games without frontiers: Investigating video games as a covert channel," in *Proc. IEEE Eur. Symp. Secur. Privacy*, Mar. 2016, pp. 63–77.
[31] M. B. Rosen, J. Parker, and A. J. Malozemoff, "Balboa: Bobbing and weaving around network censorship," in *Proc. 30th USENIX Secur. Symp.*, 2021, pp. 3399–3413.
[32] M. Wang et al., "Practical blockchain-based steganographic communication via adversarial AI: A case study in Bitcoin," *Comput. J.*, vol. 65, no. 11, pp. 2926–2938, Nov. 2022.
[33] *Bitcoin Average Transaction Fee*. Accessed: Mar. 28, 2024. [Online]. Available: https://ycharts.com/indicators/bitcoin_average_transaction_fee/

Zhuo Chen received the B.E. degree in information security from North China Electric Power University, Beijing, China, in 2019. He is currently pursuing the Ph.D. degree with the School of Cyberspace Science and Technology, Beijing Institute of Technology. His current research interests include blockchain technology and covert communication.

Liehuang Zhu (Senior Member, IEEE) is currently a Full Professor with the School of Cyberspace Science and Technology, Beijing Institute of Technology. He is selected into the Program for New Century Excellent Talents in University from the Ministry of Education, China. He has published over 60 SCI-indexed research articles in these areas and a book published by Springer. His research interests include the Internet of Things, cloud computing security, internet, and mobile security. He won the Best Paper Award at IEEE/ACM IWQoS 2017 and IEEE TrustCom 2018. He serves on the editorial boards of three international journals, including IEEE INTERNET OF THINGS JOURNAL, *IEEE Network*, and IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY.

Peng Jiang (Member, IEEE) received the Ph.D. degree from Beijing University of Posts and Telecommunications in 2017. She is currently an Associate Professor with the School of Cyberspace Science and Technology, Beijing Institute of Technology. She has published more than 40 articles in the area of cybersecurity. Her research interests include cryptography, information security, and blockchain. She has served as a program committee member for many international conferences.

Zijian Zhang (Senior Member, IEEE) received the Ph.D. degree from Beijing Institute of Technology. He is currently a Professor with the School of Cyberspace Science and Technology, Beijing Institute of Technology. His research interests include authentication and key agreement, behavior recognition, and privacy-preserving.

Chengxiang Si is currently a Researcher with the National Computer Network Emergency Response Technical Team/Coordination Center of China. His research interests include computer and information security and insider/intrusion detection.