# Anonymous Social-Tie Routing in MANET

Chih Chin Chang, Manna Lin, Jiayu Li
Computer Science Department
University of California, Los Angeles, CA 90095
{changcc5, linmanna, lijiayu1027}@ucla.edu

***Abstract*— A mobile ad hoc network(MANET) is an infrastructure-less ad hoc wireless network formed by mobile users. The devices in such an network communicate with each other directly within transmission range or through intermediate nodes. In a severe situation, when a government shuts off its country's internet access, MANETs will be a great power to defy autocracy and fight for freedom. Data shared in such situation is no longer a image or video but a important encrypted message of plan or leakage of documents. Thus the network should be source anonymous to protect the sender.**

**Our protocol, Anonymous Social-Tie Routing in MANET, is a delay tolerant network that deployed in hostile environments. Trust nodes and malicious nodes randomly distributed. Malicious nodes can hear everything within their range, follow someone, send fake values and spam to destroy such anonymous network. We address two problems: (1) Achieve anonymity through broadcast and send encrypted message. The key to the encrypted message broadcast at another location and time. (2) Build Social-Tie Routing. This protocol routes encrypted message and its key to at least one trust node and avoid routing to malicious nodes.**

## I. INTRODUCTION

### A. Motivation

Recently, there have been many cases of those in power using the infrastructure of a network to monitor and silence a population. Increasingly, some governments have been crippling access to networks and certain information while others have began mass surveiling of the Internet traffic of their citizens. And as networks become more socially significant, privacy and security concerns have become more important.

Fortunately, a mobile ad hoc network allows for data communication amongst peers without an infrastructure. And without an infrastrucutre, certain malicious behaviors on the network becomes impossible. A node cannot affect the broadcast of others that are outside of its own one hop broadcast range. And a node cannot monitor or receive data outside of its one hop neighbors.

Specifically, in a scenario where someone has some valuable but sensitive data that needs to be distributed to others that can be trusted in a network. This individual may wish to stay anonymous as the source, and the data should only be readable by some non-malicious others. Therefore, we are introducing a protocol to ensure data encryption and source anonymity. Ultimately, the goal of this paper is to demonstrate a method of allowing the distribution of some encrypted data anonymously according to an implicit assumption of trust in the network.

Intuitively, the nodes will establish and quanitfy trust with neighboring nodes by counting the frequency and recentness of encounters. Using this trust value, a receiving node will only forward to nodes whose trust values exceeds a threshold. Initially, a source node will broadcast indescriminantly to neighboring nodes, thus ensuring anonymity. The source will broadcast an encrypted message, then travel to another location, and broadcast the key for that message. This delay is in hopes of the source acquiring new neighbors so that overlaps are minimized. We investigates the effectiveness and correctness of this method using the NS-3 and will also discuss limitations.

### B. Related Work

Anonymity in a network has been an active field in networking. Particularly, ANODR has offered a protocol that borrows ideas from the onion routing protocol. Specifically, in ANODR, routes are established on demand but each forwarding nodes will encrypt the messages with its own private key. A node will know to forward a reply when it can decrypt its own private key. A trapdoor is used in a way such that only the receiver can decrypt the message.

One of the issues with ANODR is that it assumes the network contains only altruistic nodes. In case of malicious nodes that may perform attacks such as replay, the network can be compromised. SDAR is a similar routing protocol to ANODR except it solves the malicious problem by using community manager nodes. Because nodes can overhear each other, managers can recognize malicious behaviors and avoid broadcasting to those areas. However, a highly mobile network may not be able to accommodate the idea of a community.

Social tie routing is an idea that was introduced for non-anonymous encrypted routing. The idea is to route packets towards most popular nodes. Popularity is measured by the number of frequent and recent encounters and this value is stored in a table that is traded amongst neighbors. In this way nodes can always broadcast towards the most popular nodes. We borrow the idea of the trust calculation but nodes cannot share popularity scores because malicious nodes may claim to be more popular than it is.d

### C. Anonymity

A node's anonymity is measured by the size of its neighbors' anonymity set. An anonymity set is a set of candidate nodes that could have potentially sent the broadcast. Therefore, when a node receives a broadcast, the candidate nodes are its one hope neighbors. When every neighbor has a large anonymity set, then a source node is highly anonymous because the likeliness of a neighbor randomly guessing the source is low.

## II. Adversary and attack model

An adversary is a malicious node that can perform any function that non-malicious nodes have. Every node can broadcast, forward, or drop packets. An adversary may use these functions in creative ways to compromise anonymity and also to overcome the broadcast delay between encrypted message and key to decrypt the original message.

To compromise anonymity, a malicious node may increase the frequency of the encounter messages, which can affect its trust values in neighboring nodes. And to compromise the encryption and delay, a malicious node may ignore packet time out and continue to try to decrypt old messages in hopes that a key may eventually be forwarded to it.

## III. Social-Tie Routing Protocal Design

In this section, we describe our proposed social-tie-based anonymous routing protocol. The Protocol consists of two phases: reliable neighbor discovery and anonymous data transmission. During reliable neighbor discovery phase, every node records encounters with its neighbors and finds out most reliable neighbors to send the message.

### A. Notation

TABLE I
Notation Table

| | |
|---|---|
| $N_i$ | node with global id i |
| $N_s$ | source node |
| $S_i^j$ | Reliance score node $N_i$ hold for node $N_j$ |
| $t_i^{jk}$ | Global encounter time when node $N_i$ received message from node $N_j$ the $k$th time |
| $F(x)$ | Reliance score computation function |
| $MSG_i^m$ | $m$th message encrypted by node $N_i$ |
| $KEY_i^m$ | $m$th message of node $N_i$ with key |
| $M_type$ | Message type = HELLO, DATA or KEY |
| $T_r$ | Threshold for recognizing reliable neighbors from all neighbors |

### B. Anonymous Routing

In our routing protocol, we aim to broadcast a message anonymously and to avoid malicious eavesdropping. Anonymity can be achieved by broadcasting of source node. Without concluding the node identifier in the message, neighbor nodes will not recognize the sender unless there is only one node in their transmission range. Malicious eavesdropping can be avoid by our proposed message-key pair sending schema based on reliance table. The source node message-key pair sending schema is first broadcasting a message with encrypted data to neighbors. Then, after a certain delay, it sends a message with the key coordinated to the encrypted data. In this way, based on the reliance table, only people who trusted by most people can receive both encrypted data and the key to decrypt it.

### C. Reliance Table Creation

In order to avoid malicious node, every node stores the messages and forwards them to its most reliable neighbors once it receives a message. To find those reliable neighbors, each node maintains a Encounter History Table and Reliance Table. Discovery of the most reliable neighbor includes three steps:

*1) Advertise Hello Message:* Each node periodically advertises Hello messages to discover neighbors in its transmission range and to compute their reliance score. Hello Message is organized as

($i$,HELLO,content)

*2) Record Encounter Event:* When receiving a hello message, the node records the sender-id in header of the message and time it arrived in the Encounter List. In order to achieve this without acknowledge the neighbours who we are. We use global ID to store neighbours social score in Reliance table which contains global id of sender and receiving time.

*3) Compute Reliable Relationship:*

- **Encounter History Table** holds every encounter events with each neighbor node.

TABLE II
Encounter History Table Attributes

| Node ID | Encounter time |
|---|---|
| ... | ... |

- **Reliance Table** maintains every neighbor node with its reliance score connected to current node.

TABLE III
Table Attributes

| Node ID | Reliance Score |
|---|---|
| ... | ... |

According to social relationship[14], in order to send message to a certain receiver in human networks, people intend to find others that are more popular than themselves to forward the message. Similarly, when people want to send out information, they always share it with ones they trust most. First, I consider a person reliable if he has connection with me. Such as, we worked together for a class project or we were in the same department. Then, frequency and freshness are taking into consideration. If two people meet frequently, they are more likely to have a strong relationship. Moreover, if a person recently got in touch with me, I can be more familiar with him than others. Thus, reliable relationship is derived from encounter history of a node with its neighbors. global id $i$ of Node $N_i$ and encountering time $t_i$ is used to illustrate encounter event. $R_i^j$ is score used to evaluate reliable relationship of Node $N_i$ for Node $N_j$.

Assume the global encounter history starting with global time 0, the reliance score of node $N_i$ relationship with node $N_j$ at time $t_i^{jk}$ denoted by $R_i^j$ is defined as

$$R_i^j = \sum_{k=1}^{K_i^j} F(t_i^{jk})$$

where $F(x)$ is a weighing function to calculate freshness score. As what we illustrated before, $F(x)$ should be monotone decreasing function. $K_i^j$ is the total number of hello messages node $N_i$ received from node $N_j$. $t_i^{jk}$ is the global time the kth encounter event between node $N_i$ with node $N_j$. We take $F(x) = (\alpha)^{\lambda x}$, where $\alpha$ and $\lambda$ are control parameters and $0 \leqslant \lambda \leqslant 1$ as the weighing function.

*4) Adversary Issue:* There can be malicious cheating in this phase. If malicious nodes send hello messages more frequently than ordinary nodes, they are more likely to gain trust from their neighbors due to our reliance computation function. In consideration of this case. we require source nodes to only record one encounter for each neighbor node after they send neighbor request message, Hello message, in step 2.

### *D. Message Delivery*

When a source node $N_s$ schedules to broadcast information to the whole network anonymously and confidentially, both broadcasting and encryption mechanism are required due to our analysis. We need to create a uniform format for both messages sent from source nodes and messages sent from intermediate nodes.

*1) Encrypted Message Creation:* SHA-2 encryption mechanism is utilized in our design to avoid key confliction. By using unique ip address of node to create key, we can create a unqiue key_hash_id for every key sending from different nodes.

(dest_id, $M_{type}$, key_hash_id, encrypted data)

*2) Key Message Creation:* Source node $N_s$ wraps key into key key message in the following format at the same time when it creates encrytped message.

(dest_id, $M_{type}$, key_hash_id, key)

*3) Message Broadcasting:* To decrease the possibility that a malicious receive both key and encrypted message, when sending Key Message, source node $N_s$ requires moving around to meet a group of neighbors that is different from the one that it broadcasted Encrypted Message to. That is because when there are same nodes in both group, it is more likely that those two group have same reliable nodes which eliminates our assumption and thus makes our encryption useless. In consider of this movement requirement, a delay tolerance should be introduced into our network. Sending the key and message separately guarantees that no malicious node will get both the key and the encrypted message to decode the plain text during the broadcasting step and the more popular guys among popular guys can get both the key and encrypted message and thus the plain message. Thus, each node $N_i$ has to maintain two more tables

- **Encrypted Message Table Attributes**
- **Key Message Table**

Upon receiving a message, node $N_i$ proceeds with following actions

(a) $N_i$ analyzes the message, get message type, key hash id and data.
(b) If message type is Hello message, $N_i$ inserts this encounter event into Encounter History Table and renews Reliance Table.
(c) If message type is Encrypted message or Key message, $N_i$ searches its key hash id in its existing Message-key Matching Table.
(d) If key hash id does not exist in the table, which reflects that Ni has never received this packet or its corresponding key/message before, $N_i$ first stores the id and corresponding packet in the table. Then, it finds most reliable neighbors whose reliance score is above a certain threshold in Reliance Table and sends message to those nodes. To implement unicast-like message forwarding, $N_i$ replaces reliableID of the message with its reliable neighbors IDs and then broadcast.
(e) If key hash id exists in the table: if the already stored message is the same as current receiving message, Ni simply drops the packet. This means that $N_i$ receives the same packet as before. If they are different, $N_i$ uses data in key message to decrypt encrypted message and thus can get the plain data. In both cases, $N_i$ doesnt broadcast the message it receives to anyone else.

*4) Adversary Issue:* Malicious node can ignore the protocol and broadcast every message to its neighbors in order to let more malicious nodes receive the message and thus increase possibility to get both key and corresponding encrypted message.

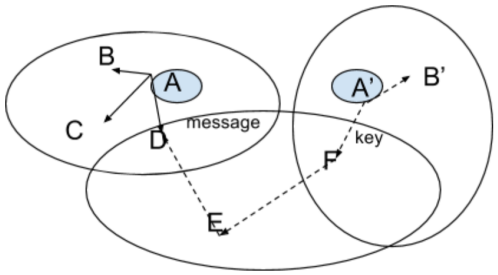


Fig. 1. Message Delivery Diagram

TABLE IV
ENCRYPTED MESSAGE TABLE

| key_hash_id | Encrypted data |
|---|---|
| ... | ... |

TABLE V
KEY MESSAGE TABLE ATTRIBUTES

| key_hash_id | key |
|---|---|
| ... | ... |

## IV. Anonymity Analysis and Evaluation

### A. Simulation Model

We deploy our protocol in NS3 using 2D random walk model and wifi ad hoc mode. This model can monitor MANETs since every node can randomly move to any direction with setting speed and travel distance. The following are parameter definitions that used in our protocol.

- **Broadcast Range**: This parameter constrain the broadcast range for every node in the model. We set the default value to 10.0 units.
- Hello Message Broadcast Interval: All the nodes broadcast hello message in every 2 seconds.
- Simulation Run Time: The whole process of setup social tie routing table, the source node send encrypted message and the key to that message is 55 seconds.
- Node Size: The number of nodes in this simulation. Default 50.
- MaliRatio: The percentage of malicious node. Default 100.
- Node Sparseness: Density of the network. Default value is 30 unit.
- Threshold: The threshold for every node to choose their trust neighbor according to the Social Tie Routing trust score. Default 1.
- Node Travel: How far a node will travel. This parameter combine with speed can monitor a more randomly changing network. Default 300 unit.
- Node Speed: The speed of each node. Default 100 unit.
- Sending Delay: The time period between sending encrypted message and key. Default 3 ms.
- Receiving Delay: The time period between message A received and key A received.(Key A can decode message A).
- Source Node: The node chosen to be the source. Default is 2. The location of the source node is vital important. If the source node move around at the corner of the network, the simulation would not be too good.

*1) Simulation Stage:* We implement our simulation in four main stages. (a) Social Tie Trust Score. The Social Tie score setup through broadcast hello message by each node. For each node, they will broadcast hello message in every 2 seconds. This is imitating the encounter process. The nodes that more frequently meet will have higher score. A time out mechanism is used for this trust score table.
(b) Send encrypted message. The source node will broadcast encrypted message in every 0.321 seconds.
(c) The key to the encrypted message will be scheduled in every 0.321 + delay seconds.
(d) Every node in this anonymous set will check the package they received. If they receive the encrypted message they will search their social score table and forward the message to the most trust node set. If any node receive the message and its key, we count the number of message received by trust node and malicious node. The message will not forward again.

*2) Evaluation:* We evaluate our proposed routing schemes in three aspects. (a) First we calculate number of message decode including the message decode by trust and malicious node. This result can on the one hand indicate if our model can direct the message and key to any trust node. In the other hand, it can test how many malicious node decode the message. Our goal is to decrease the percentage of decode by malicious node. In the meanwhile, increase the message decode by trust node.
(b) The second aspect is the average receiving delay. This means the time between an encrypted message received at node A and the message's key received at node A. This can see approximately how long we can get the message decoded. The longer the delay, the less possible to get decoded. Since some of the message were lost due to no neighbors when the node broadcast the message and the key can also be lost as time going.
(c) The most important part is node anonymity. We hope that the source can be anonymous during the process of sending message and key. However, if the source sending message in a set that is sparse or full of malicious node. The malicious node can guess in this range, who can be the source to send the node. This greatly increase the risk for the source. We calculate the anonymity by this formula:

$$neverguess = (currentReceiver->GetNeighborNum() - 1)/currentReceiver->GetNeighborNum()) \quad (1)$$

$$guess = 1 - neverguess \quad (2)$$

For each node, if they want to guess who is the source, they need to find all its neighbors. For every neighbors, we apply the above equations. $Neverguess$ means the rate that the node cannot guess who is the source. In equation (1), the minus one means the source node. The guess rate will be $1 - neverguess$. Then we sum up all the guess rate by each node. The average anonymity rate can be calculated.

### B. Simulation Results

*1) Message Decode Analysis:* In the simulation, a global vector marked the node that are trusted and malicious. When the message decode by a node, we record its information for message decode analysis. We recorded the message decode by trust node and malicious node. We hope to find out the correlation between message decode versus those parameters we have introduced in last section. We change one parameter at a time to see how those parameters effect the results.

TABLE VI

| Sparseness | 10 | 20 | 30 | 40 | 50 | 60 |
|---|---|---|---|---|---|---|
| Trust Node Decode | 2 | 3 | 4 | 5 | 6 | 7 |
| Malicious Node Decode | 4 | 2 | 6 | 5 | 4 | 5 |

As we can see in the Table VI, with all the default value describing before, we only change the sparseness of the node. The higher the value of sparseness, the node set more sparse. The total number of message sent is 16. From this table we can conclude that the sparseness can decrease the malicious node decode rate in some degree.

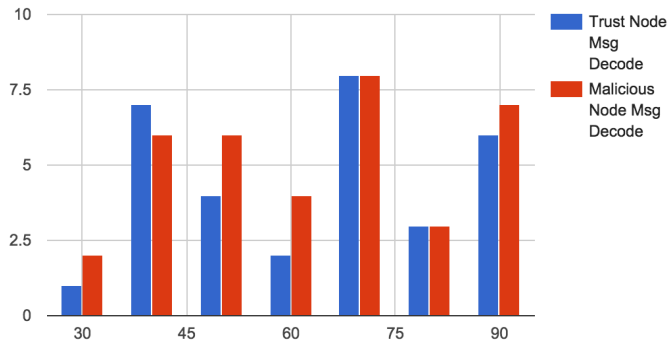


Fig. 2. Node Size VS Msg Decode

In figure 2. it indicates that if the malicious ratio and all other parameters is set, the node size does little effect on $trustnodedecoderate/maliciousnodedecoderate$

We also test on other parameters, like Figure 3. and Figure 4. When the delay sets more higher, it is less likely to match the message and key so both lines go downwards. When malicious ratio increase, the message decode by malicious node increased largely during 0.4 to 0.5. after 0.6, the message decode by malicious node does not change much.

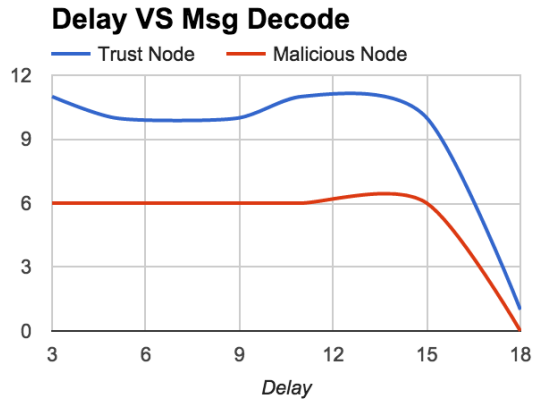


Fig. 3. Delay vs Msg Decode

*2) Receiving Delay Performance:* Lost of test proves that the changes of node size, sparseness, malicious ratio did little affect to the receiving delay. Since the sending message delay is 3 ms, the delay, the average value of receiving delay range from 3.1 to 3.7 as shown from Table VII. This is the data gathered from the change of node size and the corresponding average message delay when the message and key reached. On average, the message delay is not very apparent.

TABLE VII

| Node Size | 30 | 40 | 50 | 60 | 70 | 80 |
|---|---|---|---|---|---|---|
| Delay(ms) | 3.075 | 3.291 | 3.167 | 3.14 | 3.472 | 3.084 |

*3) Anonymity Analysis:* Finally, we evaluate the anonymity. There are three groups of data in Table VIII. The first group test the effect of node size on anonymity.

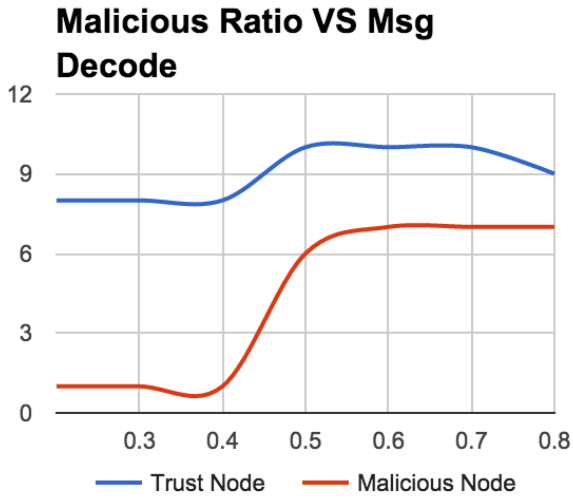


Fig. 4. Malicious Ratio VS Msg Decode

Increasing the node size can make it harder for malicious node to guess the source. Although the probability seems a little bit high, this values should be applied square root. We calculate the node anonymity in a way that have duplicate the count of guess. The second group of data in Table VIII is testing sparseness versus probability guess. The probability gets a rise after the sparseness become bigger. It is likely that the increase of sparse make every set of broadcasting with fewer neighbors. This supposed to be another cause of this rise of probability guess.

TABLE VIII

| Node Size | 30 | 40 | 50 | 60 | 70 | 80 |
|---|---|---|---|---|---|---|
| Prob Guess | 0.517 | 0.6 | 0.764 | 0.212 | 0.662 | 0.326 |
| Sparseness | 10 | 20 | 30 | 40 | 50 | 60 |
| Prob Guess | 0.62 | 0.63 | 0.764 | 0.758 | 0.818 | 0.75 |
| Malicious Ratio | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 |
| Prob Guess | 0.76 | 0.75 | 0.761 | 0.76 | 0.764 | 0.764 |

## V. Discussion

In this paper, we have introduced a method of taking advantage of social ties to establish trust amongst nodes implicitly in a network as a foundation to strategize a delayed encrypted message and key distribution from a source node to broadcast distribute to potentially trusted nodes. This protocol is one possible proposal to demonstrate a solution to the problem of distributing sensitive data in a maliciously infested network.

And while the correctness of the protocol is clearly demonstrated, there are some critical concerns yet to be addressed. Firstly, adversaries may cooperate with one another to increase their chances of decoding encrypted messages and reducing the anonymity set due to overlap. If two malicious nodes hears an initial broadcast, they can assume that the source lies somewhere in their neighborhood. And if there

is an overlap, then they do not need to double count the overlapped nodes; therefore the anonymous set is reduced. This problem remains inherent in this protocol and may not be trivially resolved.

Another issue arise when adversaries may lie anywhere along a forwarding path neighborhood. Therefore, if a path is long, there is more chance that more adversaries can overhear and retain messages and keys, waiting for a matching decryption. On the other hand, if the path is short, it means that the network is dense and the neighborhoods have large overlaps, which means that it is very likely that a malicious node will overhear both encrypted message and key.

This leads to a final important observation from the experiments. We have noticed that the performance and success of this protocol is prone to the network properties. And we have found that modifying properties have beneficial and detrimental effects in different metrics. For example, while increasing delay between broadcasting the encrypted message and the key will reduce the likeliness of a malicious node decoding the message, this has significant performance impact. The delay between sending the message and its decode is increased significantly.

Still, while some issues remain to be resolved, we have demonstrated a way to establish trust values without relying on information provided by other nodes. We have also considered possible adversarial behavior and the complexity of this non-trivial problem. And finally, we have introduced a method that raises possible improvements in the future.

## REFERENCES


[1] Lu, You, et al. "Social-tie based content retrieval for delay-tolerant mobile ad-hoc networks." Proceedings of the 8th ACM MobiCom workshop on Challenged networks. ACM, 2013.

[2] Kong, Jiejun, and Xiaoyan Hong. "ANODR: anonymous on demand routing with untraceable routes for mobile ad-hoc networks." Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing. ACM, 2003.

[3] Zhu, Bo, et al. "Anonymous secure routing in mobile ad-hoc networks." Local Computer Networks, 2004. 29th Annual IEEE International Conference on. IEEE, 2004.

[4] Lu, Y., Li, X., Yu, Y. T., & Gerla, M. (2014, April). Information-centric delay-tolerant mobile ad-hoc networks. In Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on (pp. 428-433). IEEE.

[5] Chen, Jiefeng, Roksana Boreli, and Vijay Sivaraman. "Taro: Trusted anonymous routing for manets." Embedded and Ubiquitous Computing (EUC), 2010 IEEE/IFIP 8th International Conference on. IEEE, 2010.

[6] El Defrawy, Karim, and Gene Tsudik. "ALARM: anonymous location-aided routing in suspicious MANETs." Mobile Computing, IEEE Transactions on 10.9 (2011): 1345-1358.

[7] Reed, Michael G., Paul F. Syverson, and David M. Goldschlag. "Anonymous connections and onion routing." Selected Areas in Communications, IEEE Journal on 16.4 (1998): 482-494.