PRIVACY PRESERVING AND INCENTIVE COMPATIBLE PROTOCOLS FOR COOPERATION IN DISTRIBUTED COMPUTATIONS

A Dissertation Presented

by

TINGTING CHEN

Submitted to the Graduate School of the
University at Buffalo, State University of New York in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2011

Department of Computer Science and Engineering

PRIVACY PRESERVING AND INCENTIVE COMPATIBLE PROTOCOLS FOR COOPERATION IN DISTRIBUTED COMPUTATIONS

A Dissertation Presented

by

TINGTING CHEN

Approved as to style and content by:

Sheng Zhong, Chair

Chunming Qiao, Member

Shambhu Upadhyaya, Member

Aidong Zhang, Department Chair
Department of Computer Science and Engineering

ACKNOWLEDGMENTS

I would like to first thank my dissertation advisor, Prof. Sheng Zhong very much for teaching me how to be a good researcher during my 5 years of Ph.D. study. It is his help that makes all my completed work and future career possible.

I am deeply grateful to my dissertation committee members, Prof. Chunming Qiao and Prof. Shambhu Upadhyaya. Their valuable inputs and constructive guidance always lead me towards the right direction in research.

I thank my coauthors, Prof. Yang Richard Yang, Dr. Li Erran Li, Prof. Chang Wen Chen, Prof. Lifang Wu, Dr. Fan Wu and Ankur Bansal, very much for their contributions in our collaborations. Without their hard work, I could not complete this dissertation.

Thank you to my colleagues in our research group, Haifan Yao, Zhuo Hao, Yuan Zhang and Yu Li. Their encouragement and support always cheer me up.

Last but not least, I give my deepest heart-felt thanks to my parents and my husband, Yi. They are the reason why I am here and will keep going.

ABSTRACT

PRIVACY PRESERVING AND INCENTIVE COMPATIBLE PROTOCOLS FOR COOPERATION IN DISTRIBUTED COMPUTATIONS

MAY 2011

TINGTING CHEN

Ph.D., State University of New York at Buffalo

Directed by: Professor Sheng Zhong

In today's society, advanced cyber-infrastructure are connecting a huge number of mutually unfamiliar people, and forming a computing environment with untrusted parties. The human factors of untrusted parties, especially the considerations of privacy, security and incentives, bring new challenges in designing our systems, services and software with distributed computations. This thesis addresses the incentive and privacy issues in distributed computing with untrusted parties.

Wireless networks with recent technical advances have become one of the most popular platforms for computing with untrusted parties. However, in civilian wireless networks, nodes often belong to different users and those users may be selfish. Existing protocols that were designed without considering user incentives often require nodes to spend their own resources for others. Hence selfish users may deviate from these protocols, in order to maximize their benefits. It causes serious problems to performance and even functioning of wireless networks. So, it is of great importance to provide incentives to those selfish users in wireless networks. First part of this thesis

focuses on designing incentive-compatible packet forwarding protocols for three different types of wireless networks, i.e., ad hoc wireless networks, wireless networks using network coding and vehicular ad hoc networks. First, for traditional ad hoc wireless networks, the first reputation system that has rigorous analysis and guaranteed incentive compatibility in a practical model is proposed, to enhance the cooperation among nodes in forwarding packets for others. Second, for newly emerging wireless networks using network coding, the first-ever enforceable incentive scheme to stimulate packet forwarding is proposed that uses a combination of game theoretic and cryptographic techniques. Third, the problem on how to stimulate message forwarding in vehicular ad hoc networks that have no end-to-end connections is solved, and an incentive scheme with rigorous analysis based on coalitional game theory is proposed.

In the distributed computing environments with untrusted parties, besides incentive issues, privacy concerns from the participants also impede the process of obtaining better computation results. To address the privacy concerns, the second part of this thesis focuses on designing privacy preserving distributed data mining protocols, when the input data comes from different parties. In particular, the first privacy preserving back-propagation learning algorithms for multi-layer neural networks with vertically partitioned data, using cryptographic tools, with provable security is proposed. Furthermore, a privacy preserving algorithm for growing neural gas with input data from two parties is also presented. This algorithm allows two parties to jointly conduct grow neural gas algorithm without revealing any party's data.

TABLE OF CONTENTS

INTRODUCTION

In today's society, advanced networking technologies connect a huge number of mutually unfamiliar people. The pervasive cyber-infrastructure forms a distributed computing environment with untrusted parties. The human factors of untrusted parties, especially the considerations of privacy, security and incentives, bring new challenges in designing our systems, services and software with distributed computations. On one hand, in many computing scenarios, incentive problems exist and become increasingly prevail. For instance, the selfish users in user-contributed networks only want to take advantage of peer users' services, but are not willing to contribute their own resources. It creates the famous free-rider problem [2]. On the other hand, privacy concerns have also become very important in many applications of data usage. For example, FCC [31] restricted the use of individually identifiable information in users' phone call records. It means that telecommunication companies must take consider their customers' privacy issues when designing data mining protocols for market research. This thesis addresses the incentive and privacy issues in distributed computing with untrusted parties.

Wireless networks with recent technical advances have become one of the most popular platforms for computing with untrusted parties. However, existing protocols that were designed without considering user incentives often require nodes to spend their own resources for others. Hence selfish users may deviate from these protocols, in order to maximize their benefits. It causes serious problems to performance and even functioning of wireless networks. Therefore, it is of great importance to provide incentives to those selfish users in wireless networks.

The studies on incentives in this thesis have focused on designing incentive-compatible packet forwarding protocols for three different types of wireless networks, i.e., ad hoc wireless networks, wireless networks using network coding and vehicular ad hoc networks. The first part of thesis focuses on designing incentive-compatible packet forwarding protocols because packet forwarding is essential in providing reliable end-to-end packet transmission services in wireless networks. On the other hand, different types of wireless networks require different packet forwarding protocols according to their network conditions and quality of service needs. As a result, the work is divided into three parts, with each part particularly addressing the packet-forwarding incentive problems in one type of wireless networks. In this thesis, in order to rigorously show the system-wide performance with presence of selfish users, models and concepts in game theory are applied [48, 49] in the analysis.

To address the privacy concerns, in this thesis, the second part focuses on designing privacy preserving distributed data mining protocols, when the input data comes from different parties. In the recent years, privacy preserving data mining has attracted a lot of research efforts. The existing works can be classified into three categories, i.e., summarization-based approaches (e.g., [3, 118]), perturbation-based approaches [4, 43, 23] and cryptography-based approaches [69, 70, 101, 102, 65]. As discussed in Section 1.3.2, there is always a three way tradeoff among the metrics of result accuracy, efficiency and security. Any approach, if it aims to optimize one metric, must sacrifice at least one of the other two metrics. The works in this thesis fall into the category of cryptography-based approaches. The reason why to choose cryptography-based approaches is that in many applications especially in the health-care related applications, patients' privacy and result accuracy are the top considerations. Cryptographic-based approaches can minimize the accuracy loss and provide provable security guarantee. Different from the general secure multi-party computation [115], specific privacy preserving protocols for different mining applications are

designed and thus these protocols achieve better efficiency than the existing general multi-party computation methods.

The rest of this thesis is organized as follows.

Part I: Preliminaries

In Chapter 1, first some concepts in game theory and the cryptographic techniques that will frequently be use in this thesis are briefly reviewed. Then the related work for incentive-compatible protocols in wireless networks and for privacy preserving data mining are reviewed respectively.

Part II: Incentive-Compatible Packet Forwarding in Wireless Networks

This part of thesis mainly focuses on solving the incentive issues for packet forwarding in wireless networks.

In Chapter 2, for traditional ad hoc wireless networks, the first reputation system that has rigorous analysis and guaranteed incentive compatibility in a practical model is proposed, to enhance the cooperation among nodes in forwarding packets for others.

In Chapter 3, for newly emerging wireless networks using network coding, the first-ever enforceable incentive scheme to stimulate packet forwarding, is proposed which uses a combination of game theoretic and cryptographic techniques.

Chapter 4 studies how to stimulate message forwarding in vehicular ad hoc networks that have no end-to-end connections, and proposes an incentive scheme with rigorous analysis based on coalitional game theory.

Part III: Privacy Preserving Distributed Data Mining

Part III focuses on designing privacy preserving distributed data mining protocols to address the privacy concerns in computations with untrusted parties.

In Chapter 5, the first privacy preserving back-propagation learning algorithms for multi-layer neural networks with vertically partitioned data is proposed, using cryptographic tools, with provable security.

In Chapter 6, a privacy preserving algorithm for growing neural gas with input data from two parties is proposed. This algorithm allows two parties to jointly conduct grow neural gas algorithm without revealing any party's data.

Part I

Preliminaries

CHAPTER 1

TECHNICAL PRELIMINARIES

1.1 A Review of Some Concepts in Game Theory

1.1.1 Non-cooperative Strategic Game

In a non-cooperative strategic game, any player i in player set N tries to maximize its own utility (payoff), denoted by u_i. We denote one strategy of player i by s_i and the strategy space of i by S_i. The set of chosen strategies of all players constitutes a strategy profile, written as $s = \{s_1, s_2, \cdots, s_{|N|}\}$. Note that a strategy profile includes one and only one strategy for *each* player. s_{-i} is the strategies set chosen by all the other players except player i. Formally, $s_{-i} = \{s_1, \cdots, s_{i-1}, s_{i+1}, \cdots, s_{|N|}\}$. Every player prefers strategy s_i to s'_i, if s_i brings higher utility, $u_i(s_i, s_{-i}) > u_i(s'_i, s_{-i})$.

In game theory, Nash equilibrium is an important solution concept. The formal definition as in [83] is described below.

Definition 1. *Nash Equilibrium: The strategy profile s^* constitutes a Nash equilibrium, if for each player i,*

$$u_i(s_i^*, s_{-i}^*) \geq u_i(s_i, s_{-i}^*), \forall s_i \in S_i. \tag{1.1}$$

A Nash equilibrium solution guarantees that no players can benefit by deviating from it if other players do not change their strategies.

1.1.2 Repeated Games

A repeated game is is divided into T stages. When $T > 0$ is a finite number, we call the game a finite repeated game; otherwise, we call it an infinite repeated game.

In each stage of the game, there is an action set available to player v_i. In stage t, player v_i chooses an action $a_{i,t}$ from A_i. The utility of player v_i in stage t is decided by the actions of all players in the game. When taking an action, a player knows the actions previously chosen by the other players.

One of the most popular forms of preference relations that can be used to model in an infinitely repeated game is discounting form, where a *discounting* total utility in the entire game is considered. Let $\delta < 1$ be a constant—we call it the discount factor. The total utility of v_i in the entire game is

$$u_{i,\mathbf{total}} = \sum_{\ell=1}^{\infty} \delta^{\ell-1} u_{i,\ell}.$$

Intuitively, this means the player v_i has more interest in the current stage and the near future than in the far future.

In a repeated game, a history refers to a number of continuous stages starting from the beginning of the entire game, such that the actions of all players in all these stages have been chosen. The length of a history is defined as the number of stages in it. Given a history, the players can play the rest of the game, which constitutes a subgame. In each subgame, we can consider the utility of each player just as in the entire repeated game. For example, for a history H of length L, the utility of v_i in the subgame immediately following H is

$$u_i|_H = \sum_{\ell>L} \delta^{\ell-1} u_{i,\ell}.$$

For the entire game, a strategy s_i for each player v_i specifies what action v_i should choose after each possible history. Clearly, once every player has determined its strategy, the utilities of all players are also fixed. Hence, for each player v_i, the total utility, the utility in any stage, or the utility in any subgame, is always a function of the profile of all players' strategies. Denote by s ($s = (s_1, \ldots, s_n)$) the profile of all

players' strategies. We use $u_i|_H(s)$, to represent the utility of node v_i in the subgame immediately following history H, when the strategy profile s is used by the players.

We say a strategy profile $s = (s_1, \ldots, s_n)$ induces a *Nash equilibrium* in the subgame immediately following history H if for all v_i, for all $s_i' \neq s_i$,

$$u_i|_H(s_1, \ldots, s_{i-1}, s_i', s_{i+1}, \ldots, s_n) \leq u_i|_H(s).$$

We say s is a *subgame perfect equilibrium* if s induces a Nash equilibrium in every subgame.

1.1.3 Coalitional Games

In coalitional game theory, the central concept is the formation of coalitions. Each coalition is a subset of game players who cooperatively join their forces. Each selfish player always tries to join the coalition that can maximize its own payoff share. Denote by $\mathbf{R}$ the set of real numbers. Formally, a coalitional game can be defined as follows.

Definition 2. *A coalitional game is an ordered pair ($\boldsymbol{N}$, v), where N is the set of players and v is a characteristic function from 2^N to $\boldsymbol{R}$ such that $v(\phi) = 0$. Each subset of $\boldsymbol{N}$ is called a coalition. Hence, the characteristic function v actually assigns a real number to each coalition, called the payoff of that coalition. The coalition $\boldsymbol{N}$, which consists of all players, is called a grand coalition.*

Intuitively, for a coalition S, $v(S)$ is the amount of overall benefit that can be obtained by the players in S from cooperation agreements among them.

Ideally, we want all players to join the grand coalition, so that any two players cooperate with each other. Since each player has the freedom to choose the coalition to join based on its own interests, we must ensure that joining the grand coalition is to the best interest of every player. In coalitional game theory, there is a classic solution concept, *core*, which gives us such a guarantee.

Definition 3. *In a coalitional game* $(\boldsymbol{N}, v)$*, the core* $C(v)$ *is the set of payoff allocations* $x \in \boldsymbol{R}^N$*, s.t.*

$$C(v) = \{x \in \boldsymbol{R}^N | \sum_{i \in N} x_i = v(N); \sum_{i \in S} x_i \geq v(S), \forall S \subseteq N\}$$

From the above definition we can see that an allocation lies in the core is *efficient* [83] in that $\sum_{i \in N} x_i = v(N)$, which means no payoff is wasted. Moreover, because $x_i \geq v(\{i\})$, an allocation in the core is *individually rational* [83], which means each player can obtain a payoff share no less than acting alone. (In fact, each player's payoff share is no less than joining any other coalition.)

Note that the core of a coalitional game may be of any size; it may even be empty. If the core is empty, then we cannot guarantee it is to the best interest of every player to join the grand coalition.

1.2 Frequently Used Cryptographic Techniques

1.2.1 ElGamal Encryption Scheme

ElGamal is a public-key encryption scheme which can be defined on any cyclic group. Let G be a cyclic group of prime order q with generator g. We assume that decisional Diffie-Hellman assumption (DDH) [14] holds in G such that ElGamal is semantically secure.

Components. ElGamal scheme consists of three components, i.e., key generation, encryption and decryption.

- Key Generation.

  A value $x \in Z_p$ is randomly chosen as the private key. The corresponding public key is (G, q, g, h), where $h = g^x$.

- Encryption.

A message $m \in G$ is encrypted as follows. A value $r \in Z_p$ is chosen as random. Then the ciphertext is constructed as $(C_1, C_2) = (g^r, m \cdot h^r)$.

- Decryption.

  The plain text is computed as

$$\frac{C_2}{C_1^x} = \frac{m \cdot h^r}{g^{x \cdot r}} = \frac{m \cdot h^r}{h^r} = m.$$

Homomorphic Property. ElGamal scheme is homomorphic in that for two messages, m_1 and m_2, an encryption of $m_1 + m_2$ can be obtained by an operation on $E(m_1, r)$ and $E(m_2, r)$ without decrypting any of the two encrypted messages.

Probabilistic Property. ElGamal scheme is also probabilistic, which means that besides clear texts, the encryption operation also needs a random number as input. Let an encryption of message m using public key (G, q, g, h) and a random number r be denoted as $E_{(G,q,g,h)}(m, r)$. For simplicity we use notation $E(m, r)$ instead of $E_{(G,q,g,h)}(m, r)$.

In probabilistic encryption schemes, there are many encryptions for each message. ElGamal allows an operation that takes one encrypted message as input and outputs another encrypted message of the same clear message. This is called rerandomization operation. For instance, taking the encrypted message $(C_1, C_2) = (g^r, m \cdot h^r)$ as input, one can do the rerandomization and obtain another cyphertext of m as,

$$(C_1', C_2') = (C_1 \cdot g^s, C_2 \cdot h^s) = (g^{r+s}, m \cdot h^{r+s}).$$

1.3 State of Art

1.3.1 Incentive Compatible Protocols in Wireless Networks

There has been extensive study of the incentive problems in routing and packet forwarding of wireless ad hoc networks. Generally, the solutions proposed follow one

(or both) of the two approaches: the approach of reputation systems [15, 16, 52, 78, 9, 74, 79, 88] and the approach of credit-based systems [19, 96, 97, 119, 12, 7, 59, 105, 120, 106, 107, 121].

1.3.1.1 Reputation-based systems

In a reputation system, the behavior of a node is observed by other nodes in the network; it gets a bad reputation if it appears to have dropped packets. Reputation systems can be divided into two categories: end-to-end reputation systems and the more recently proposed hop-by-hop reputation systems.

The first solution to the incentive-compatible packet forwarding problem is an end-to-end reputation system proposed by Marti, et al. [76]. They design a watch dog and a path rater, which monitor the reputation of nodes. Another early solution by Buchegger and Le Boudec, is called CONFIDANT [16, 15]. It maintains a state machine at each node for the reputation of other nodes. These two systems are good examples of end-to-end reputation systems in that each node monitors nodes that are not its neighbors and calculates their reputations.

In recent years, hop-by-hop reputation systems have been proposed to reduce the overheads of reputation systems. Good example are SORI, proposed by He, et al.[52] and Catch, proposed by Mahajan, et al. [74]. In SORI, each node calculates the reputation of its neighbors only based on the observations made by itself and its other neighbors. Another example of hop-by-hop reputation systems is Catch, proposed by Mahajan, et al. [74].

The above reputation systems, along with many other reputation systems (e.g., [78, 9, 79, 88]), have a common limitation: they do not have rigorous analysis of their incentive compatibility. Hence, it is not clear what guarantee they can provide in terms of incentive compatibility.

As far as we know, Milan, et al.'s GTFT [80](which was first studied in a different content by Wu and Axelrod [110]) and Jaramillo and Srikant's DARWIN [60] are two reputation systems in wireless networks that have rigorous analysis. Both of them are shown to provide SPNE solutions.

1.3.1.2 Credit-based Systems

Buttyan and Hubaux [19, 12] are the first to propose to use credit-based systems for the packet forwarding problem in wireless ad hoc networks. Zhong, et al. propose Sprite [119] that does not need tamper-proof hardware. Another credit-based system is due to Ben Salem, et al. [93], which uses symmetric key cryptography as the main tool. Later, Wan, et al. [105] design a credit-based system for multicast.

Other important credit-based systems include [59, 41, 42, 106, 107, 121], among others.

There are also systems like OCEAN[9], which use a combination of reputation system and credit-based system.

Some ideas in combinatorial auctions [87, 85, 103, 32, 38] may be helpful in providing incentive compatible solutions for the packet forwarding problem in wireless networks. For example, Rassenti et. al. [87] presented a sealed-bid combinatorial auction for the allocation of airport takeoff and landing slots to competing airlines. Another example is the most famous combinatorial auction, the Vickrey-Clarke-Groves (VCG) mechanism. In a VCG auction, bidders report their valuations for all goods and items are allocated efficiently to maximize total value. If we model the packet forwarding process among nodes as a combinatorial auction, then such ideas could possibly be applied to solve the packet forwarding issue in network coding. For example, we may consider maximizing total utility by choosing different combinations and consideration of incentive compatibility for the auctioneer. However, deriving a solution in this way is not trivial.

1.3.2 Privacy Preserving Distributed Data Mining

The notion of privacy preserving data mining was proposed by two different papers ([4] and [69]) in the year 2000. Both of the two papers addressed the problem of performing data analysis on distributed data sources with privacy constraints. In [4], Agrawal et al. presented a solution by adding noise to the source data, while in [69] Lindell and Pinkas used cryptographic tools to efficiently and securely build a decision tree classifier.After these two papers, a good number of data mining tasks have been studied with the consideration of privacy protection, for example classification [118], clustering [56, 101], association rule mining [43], etc.

In particular, privacy preserving solutions have been proposed for the following classification algorithms (to name a few): decision trees [69, 70, 39], Naive Bayes classifier [102, 108], and SVM [23, 116, 65]. Generally speaking, the existing works have taken either randomization based approaches (e.g., [4, 23]) or cryptography based approaches (e.g., [70, 101, 102, 65]). Randomization based approaches, by perturbing data, only guarantee a limited degree of privacy. In contrast, cryptography based approaches provide better guarantee on privacy than randomized based approaches, but most of the cryptography based approaches are difficult to be applied with very large databases, because they are resource demanding. For example, although Laur et al. proposed an elegant solution for privacy preserving SVM in [65], their protocols are based on circuit evaluation which is considered very costly in practice.

In cryptography, there is also a general-purpose technique called secure multi-party computation. The works of secure multi-party computation originate from the solution to the millionaire problem proposed by Yao [115], in which two millionaires can find out who is richer without revealing the amount of their wealth. In [114], a protocol is presented which can privately compute any probabilistic polynomial function. Although secure multi-party computation can theoretically solve all problems of privacy-preserving computation, it is too expensive to be applied to practical prob-

lems [50]. This general solution is especially infeasible for cases in which parties hold huge amounts of data.

Now we review the related works on privacy preserving neural networks learning as it is the particular data mining problem that we have studied. There are few works on the problem of privacy preserving neural networks learning (limited to [22, 10, 104]). The most recent one is [104]. As discussed above, the difference between our work and their is that we focus on privacy preserving neural networks and provide a light-weight algorithm applicable to more complex neural networks configurations, while their protocol is for gradient descent methods in general and thus loses some power for neural networks in particular.

Protocols in [22, 10] are also designed for privacy-preserving neural-network-based computations. In particular, Chang and Lu [22] proposed a cryptographic protocol for non-linear classification with feed-forward neural networks. Barni et al. in [10] presented three algorithms with different levels of privacy protections, i.e., protecting private weight vectors, protecting private activation functions, and preventing data providers from injecting fake cells to the system.

The fundamental difference between our work and the protocols in [22] and [10] is that they work in different learning scenarios. In [22] and [10], it is assumed that there is a neural network owner; this neural network owner owns a neural network, but does not have any data to train it. In addition, there are some data providers who have data that can be used to train the neural network. The goal of [22] and [10] is to ensure that the neural network owner does not get any knowledge about the data, and at the same time, the data providers do not get the knowledge embedded in the neural network (e.g., the node activation functions). In constrast, we consider a totally different scenario in which there are at least two neural network owners, each having his own set of data. The two parties want to jointly build one neural network based on all the data, but each party does not want to reveal his own data.

As a result, protocols in [22] and [10] cannot be applied in the scenario that we are studying.

Moreover, [22] and [10] are of theoretical interests only; they have not implemented their protocols, neither have they tested their protocols in any experiments. In contrast, we have implemented our algorithm and carried out extensive experiments. The results of our experiments show that our algorithm is practical.

Part II

Incentive-Compatible Packet Forwarding in Wireless Networks

CHAPTER 2

PACKET FORWARDING COOPERATION IN WIRELESS AD-HOC NETWORKS[26]

2.1 Background and Motivation

A wireless ad hoc network does not have an infrastructure. In such a network, the cooperation of nodes is needed for forwarding other nodes' packets. If nodes do not forward each other's packets, the entire wireless ad hoc network cannot function properly. Nevertheless, in civilian ad hoc networks, nodes belong to different users and thus have their own interests. Consequently, we need to give nodes incentives to make them cooperative.

There are mainly two approaches to give nodes incentives: reputation-based systems and credit-based systems. In this chapter, we focus on the reputation-based approach, which is highly efficient and has been effectively applied to wireless ad hoc networks.

The existing works on reputation systems suffer from one of two problems. First, most of them (e.g., [76, 16, 15, 52, 74]) do *not* have rigorous analysis of incentive-compatibility. Hence, it is not clear what guarantee for cooperation these reputation systems can provide. Second, although some other reputation systems [80, 60] do have rigorous analysis, their analyses are in *unrealistic models*. Therefore, in practice, their work cannot guarantee cooperation as well.

Specifically, the existing reputation systems which have rigorous analyses study the interaction between each pair of neighbor nodes, as an *infinite* repeated reputation game. Hence, their analyses of incentive compatibility can be valid *only if* the

reputation game between each pair of nodes continues infinitely. However, in reality, the games are finite. Therefore, even though their analyses are mathematically correct for an infinite repeated game, they do not really guarantee cooperation in a finite repeated game, i.e., in reality.

To see more clearly the problem caused by a finite repeated game, let us consider a simple example of two selfish neighbor nodes, v_0 and v_1. Suppose v_0 knows that his game will end, say, after two hours (e.g., he is going to move out of the transmission range of his neighbors at that time). Recall that a finite repeated game consists of a finite number of stages. As a selfish node, his best strategy is to figure out the total number of stages in the repeated game and refuse to cooperate in the last stage, because no punishment can be made after that. Unfortunately, his neighbor v_1 is as smart as him. She is also aware of his best strategy. Consequently, given the strategy of v_0 that drops all packets of v_1 in the last stage, the best strategy for v_1 is not to cooperate in the last *two* stages, because no matter v_1 cooperates or not, her own packets are not forwarded by v_0 in the last stage, and if she chooses to not cooperate in the last two stages, no more punishment she will receive. This sounds pretty bad, but is still not the end of story. Considering the strategy of v_1, v_0 decides that he had better change his strategy to no cooperation in the last *three* stages ... Therefore, we will have a *cascade* of no cooperation, until finally both v_0 and v_1 decide not to cooperate in the entire game.

In fact, we can prove that, not only the existing reputation systems, but also any reputation system designed in the traditional way, will fail to provide incentive compatibility in the realistic model (see Theorem 4).

To overcome the difficulty in the realistic model, in this chapter we propose FITS (FInite-Time reputation System). It is the first reputation system that has a proof of incentive compatibility in a practical model. FITS uses a novel technique called *Threat To Interfere* (TTI). The idea of TTI is very simple. We allow a node to

threaten to interfere with his neighbor's communications after a finite reputation game if his neighbor does not cooperate in the last stage of the game. In this way, the cascade of no cooperation is prevented from its very beginning. Hence, all nodes will cooperate in the entire reputation games, and thus *no real interference is needed*. (For more details, see Section 2.4.)

Our contributions are briefly summarized below:

- First, we show that traditional reputation systems *cannot* provide any Subgame Perfect Nash Equilibrium (SPNE) solution in a finite reputation game. Therefore, it is necessary to enhance the reputation system approach.

- Second, we enhance the reputation system approach by introducing the novel TTI technique. Using this technique, we design the FITS-D scheme and show that, when the FITS-D scheme is used, under an assumption called Perceived Probability Assumption (PPA), there is a SPNE in which the forwarding probabilities of all nodes are close to 1.

- Third, we also propose the FITS-I scheme, which does not need the PPA. Without the PPA, we show that it also has a SPNE in which the forwarding probabilities of all nodes are close to 1.

- Finally, experiments verify that our FITS schemes provide strong incentives for nodes to cooperate.

The rest of this chapter is organized as follows. Section 2.2 presents technical preliminaries. In Section 2.3, we show that traditional reputation systems cannot provide SPNE solutions when the reputation game has a finite number of stages. In Sections 2.4 and 2.5, we present the FITS-D and FITS-I schemes, respectively. Evaluation results are presented in Section 2.6. We summarize in Section 2.7.

2.2 Technical Preliminaries

In this chapter, we study a wireless ad hoc network in which the nodes are in promiscuous mode. That is, each node can hear the transmissions of its neighbors. Furthermore, each link in this network is bidirectional. That is, if a node can receive packets from another node, then the latter node can also receive packets from the former node. Like in [60], we assume that nodes are selfish *not malicious*. That is, each node is interested only in maximizing its own utility.

Same as in a lot of previous works [52, 74, 80, 60], we consider a *hop-by-hop* reputation system and isolate each pair of neighbors (v_0, v_1). We study the *reputation game* between them. (What we describe in the rest of this section is the *basic model* of the reputation game, which is used to model *traditional reputation systems* in this chapter. In Section 2.4, the model is extended so that it can model reputation systems using a newly introduced technique.) The reputation game is divided into T stages, where $T > 0$ is a finite number. In each stage, a sufficiently large number of packets are transmitted between v_0 and v_1.

Formally, our reputation game is a finite repeated game. The players of this game are the pair of neighbor nodes, v_0 and v_1. In each stage of the game, there is an action set available to player v_i ($i \in \{0, 1\}$): $A_i = \{a_i | 0 \leq a_i \leq 1\}$. In stage t, player v_i chooses an action $a_{i,t}$ from A_i. Intuitively, $a_{i,t}$ is the probability that v_i forwards v_{1-i}'s packets in stage t. The utility of player v_i in stage t is decided by the actions of both players:

$$u_{i,t} = a_{1-i,t}u - a_{i,t}c,$$

where u is the amount of benefit player v_i can receive if all its packets are forwarded by v_{1-i} in the entire stage, and c is the amount of cost needed by v_i for forwarding all v_{1-i}'s packets in the entire stage. In this chapter, we assume that all nodes have the same (u, c) in all stages. Clearly, $u > c > 0$.

The utility of player v_i in the entire reputation game is the sum of its utilities in all stages:

$$u_i = \sum_{t=1}^{T} u_{i,t}.$$

Note that u_i is a function of the two players' actions in all stages, sometimes written as $u_i((a_{0,1}, a_{1,1}), \ldots, (a_{0,T}, a_{1,T}))$. However, unlike in a standard repeated game, in our reputation game, the actions of each player are *invisible* to the other player. That is, player v_i cannot directly see player v_{1-i}'s actions $a_{1-i,t}$ (for $t = 1, \ldots, T$), because there are collisions. Suppose p_c is the probability that a packet forwarded by v_i cannot be overheard by v_{1-i} due to collision. (As in previous work, we assume all nodes have the same p_c in all stages.) Then, what player v_i can see in stage t is a *perceived probability* that player v_{1-i} forwards its packets:

$$\hat{a}_{1-i,t} = (1 - p_c)a_{1-i,t}.$$

In the basic model, we also call $\hat{a}_{1-i,t}$ the perceived action of player v_{1-i}. Previous work [60] has introduced an assumption that we call the *Perceived Probability Assumption* (PPA): both players (v_0 and v_1) can see both perceived actions ($\hat{a}_{0,t}$ and $\hat{a}_{1,t}$) in every stage. Note that here the *actions* refer to the actions (i.e., forwarding probabilities) defined in the finite repeated game. In this chapter, our FITS system has two schemes, FITS-D which uses the PPA, and FITS-I which does not.

Consequently, our definition of strategy is also different from that in a standard repeated game. Based on whether the PPA assumption is used or not, we distinguish two types of strategies: PPA-dependent strategies and PPA-independent strategies. Here a PPA-dependent strategy s_i for player v_i is a function defined on all possible histories of both players' perceived actions in stages 1 through t, where $t \leq T - 1$, together with other information v_i obtains in these stages. For each such history, s_i specifies an action for player v_i to take in the next stage. Formally, if v_i uses strategy

s_i, the action v_i should take in stage $t+1$ is $s_i((\hat{a}_{i,1}, \hat{a}_{1-i,1}, I_{i,1}), \ldots, (\hat{a}_{i,t}, \hat{a}_{1-i,t}, I_{i,t}))$, where $(\hat{a}_{i,1}, \hat{a}_{1-i,1}), \ldots, (\hat{a}_{i,t},\ \hat{a}_{1-i,t})$ are the perceived actions of the two players in the first t stages, and $(I_{i,1}, \ldots, I_{i,t})$ is other information v_i obtains from v_{i-1} about its forwarding probabilities in stage 1 through t. (If there is no other information, $(I_{i,1}, \ldots, I_{i,t}) = (\perp, \ldots, \perp)$, where $\perp$ is a special symbol denoting no information.) In particular, in the first stage, the history is an empty history (also denoted by $\perp$). Thus, $s_i(\perp)$ is the action v_i should take in the first stage if v_i is using PPA-dependent strategy s_i.

Besides PPA-dependent strategies, we also study PPA-independent strategies. Here a PPA-independent strategy s_i for player v_i is a function defined on all possible histories of v_i's own actions and v_{1-i}'s perceived actions in stages 1 through t, where $t \leq T-1$, together with other information v_i obtain in these stages. For each such history, s_i specifies an action for player v_i to take in the next stage. Formally, $s_i((a_{i,1}, \hat{a}_{1-i,1}, I_{i,1}), \ldots, (a_{i,t}, \hat{a}_{1-i,t}, I_{i,t}))$ is the action v_i should take in stage $t+1$ if v_i is using strategy s_i, where $(a_{i,1}, \ldots, a_{i,t})$ are the actions taken by v_i in the first t stages, and $(\hat{a}_{1-i,1}, \ldots, \hat{a}_{1-i,t})$ are the perceived actions of v_{1-i} in the first t stages. Note that, as we have mentioned, $(I_{i,1}, \ldots, I_{i,t})$ is other information v_i obtains in stage 1 through t; it does not include the perceived probabilities. In particular, $s_i(\perp)$ is the action v_i should take in the first stage if v_i is using PPA-independent strategy s_i.

Let $s = (s_0, s_1)$ be a profile of PPA-dependent or PPA-independent strategies. If the two players are using this strategy profile, then the actions they take in all strategies can be easily determined. These actions are called the *outcome* of s, denoted by $O(s)$. The utility of player v_i when s is used by the two players is defined as the utility of v_i when $O(s)$ are taken by the two players. Denote this utility by $u_i(s)$. Consequently, $u_i(s) = u_i(O(s))$.

2.2.1 SPNE and One Deviation Theorem

The main objective of this chapter is to design a system that converges to a SPNE in which both players forward packets with probability 1. To define SPNE, we first briefly review the definition of subgame.

A subgame of our reputation game starts in stage t_0 ($1 \leq t_0 \leq T$) of the reputation game and ends in stage T of the reputation game. Hence a subgame can be viewed as a finite repeated game of $T - (t_0 - 1)$ stages. We can identify a subgame with its *initial history*, *i.e.*, what happened in stages 1 through $t_0 - 1$. Suppose the initial history is $h = ((a_{0,1}, a_{1,1}), \ldots, (a_{0,t_0-1}, a_{1,t_0-1}))$. Then the corresponding subgame is denoted by $\Gamma(h)$.

The players of $\Gamma(h)$ are the two players of the reputation game, v_0 and v_1; the action set available to each player in the subgame is also the same as that in the reputation game. The utility of player v_i in the subgame $\Gamma(h)$ is the sum of v_i's utilities in all stages of $\Gamma(h)$:

$$u_{i,\Gamma(h)} = \sum_{t=t_0}^{T} u_{i,t}.$$

Since $u_{i,\Gamma(h)}$ can be determined by the actions taken in stages t_0 through T, or by the strategy profile the two players use, we can write it as $u_{i,\Gamma(h)}((a_{0,t_0}, a_{1,t_0}), \ldots, (a_{0,T}, a_{1,T}))$ or $u_{i,\Gamma(h)}(s_0, s_1)$.

A strategy (resp., strategy profile) in the reputation game induces a strategy (resp., strategy profile) in any subgame. For simplicity of notation, we often use the same symbol to represent a strategy (resp., strategy profile) and its induced strategy (resp., strategy profile).

In the reputation game, a PPA-dependent (resp., PPA-independent) strategy profile $s^* = (s_0^*, s_1^*)$ is a Nash equilibrium (NE) if for all $i \in \{0, 1\}$, for all PPA-dependent (resp., PPA-independent) strategy s_i,

$$u_i(s^*) \geq u_i(s_i, s_{1-i}^*).$$

Similarly, in subgame $\Gamma(h)$, a PPA-dependent (resp., PPA-independent) strategy profile s^* induces a NE if for all $i \in \{0,1\}$, for all PPA-dependent (resp., PPA-independent) strategy s_i,

$$u_{i,\Gamma(h)}(s^*) \geq u_{i,\Gamma(h)}(s_i, s^*_{1-i}).$$

We say a PPA-dependent (resp., PPA-independent) strategy profile s^* is a SPNE of the reputation game if for every subgame $\Gamma(h)$, s^* induces a NE in $\Gamma(h)$.

Since a finite repeated game is also a finite-horizon extensive game, we can apply the famous One Deviation Theorem [83] to our reputation game. In the context of our reputation game, the theorem states that s^* is a SPNE if and only if for each player, a deviation from s^* in any single stage cannot bring more utility to the player in the subgame starting from the stage of deviation. Formally, s^* is a SPNE if and only if for $i \in \{0,1\}$, for every subgame $\Gamma(h)$, for all $a_i \in A_i$,

$$u_{i,\Gamma(h)}(s^*) \geq u_{i,\Gamma(h)}(s^*_i|_{h \to a_i}, s^*_{1-i}),$$

where $s^*_i|_{h \to a_i}$ denotes a strategy in which action a_i is taken right after history h, and the action specified by s^*_i is used after any other history.

2.3 Impossibility Of SPNE Solution for Traditional Reputation Systems

Given the basic model of finite reputation game, now we show that it is impossible to design a traditional reputation system (in which punishment can only be dropping packets) that provides a SPNE solution in this model. Intuitively, this impossibility result means that we cannot use *any* traditional reputation systems to achieve strong incentive compatibility in finite reputation games. More precisely, if a traditional

reputation system is used in a finite reputation game, then in all SPNE, both players of the game drop all packets of each other. A formal theorem and proof are given below.

Theorem 4. *In the basic model of reputation game (as defined in Section 2.2), for all SPNE s^*, and all history h such that its length $|h| < T$, the outcome of s^* in the subgame $\Gamma(h)$ is*

$$O_{\Gamma(h)}(s^*) = (0,0)^{T-|h|} = ((0,0),\ldots,(0,0)).$$

In particular, this implies that, $\forall i \in \{0,1\}$, $u_i(s^*) = 0$.

Proof. Here we prove this theorem for the case s^* is PPA-dependent. If s^* is PPA-independent, we can easily obtain a similar proof.

Note that it suffices to show that for all SPNE s^*, all history h such that its length $|h| < T$, and all $i \in \{0,1\}$, we have $s_i^*(h) = 0$.

We prove this theorem by induction on $|h|$. If $|h| = T-1$, by the definition of SPNE, $s^*(h)$ is a NE in stage T. Hence,

$$s_i^*(h) = \arg\max_{a_{i,T}}(a_{1-i,T}u - a_{i,T}c) = 0.$$

Now, suppose that the above proposition is true for $|h| \geq h_0$, where $h_0 \in N^+$ and $1 \leq h_0 \leq T-1$. We show that it is true for $|h| = h_0 - 1$ as well. Let $h' = (h, (\hat{a}_i, \hat{s}_{1-i}^*(h), \perp))$, where $\hat{a}_i$ and $\hat{s}_{1-i}^*(h)$ are the perceived probabilities of a_i and $s_{1-i}^*(h)$, respectively. For all $i \in \{0,1\}$, all a_i, we have

$$\begin{aligned} u_{i,h_0}(a_i, s_{1-i}^*(h)) &= u_{i,\Gamma(h)}(s_i^*|_{h\to a_i}, s_{1-i}^*) \\ &\quad -u_{i,\Gamma(h')}(s^*) \\ &\leq u_{i,\Gamma(h)}(s^*) - u_{i,\Gamma(h')}(s^*) \\ &= u_{i,\Gamma(h)}(s^*) - \sum_{t=h_0+1}^{T}(a_{1-i,t}^{h'}u \\ &\quad -a_{i,t}^{h'}c), \end{aligned}$$

where $((a^{h'}_{i,h_0+1}, a^{h'}_{1-i,h_0+1}), \ldots, (a^{h'}_{i,T}, a^{h'}_{1-i,T}))$ is the outcome of s^* in $\Gamma(h')$.

By the induction assumption, we get that, for all $i \in \{0,1\}$, for $t = h_0+1, \ldots, T$, $a^{h'}_{i,t} = 0$. Therefore,

$$u_{i,h_0}(a_i, s^*_{1-i}(h)) \leq u_{i,\Gamma(h)}(s^*). \tag{2.1}$$

Let $h'' = (h, (\hat{s}^*(h), \perp))$, where $\hat{s}^*(h)$ is the perceived probabilities of $s^*(h)$. Similar to the above, we can obtain that

$$u_{i,h_0}(s^*(h)) = u_{i,\Gamma(h)}(s^*) - \sum\nolimits_{t=h_0+1}^{T}(a^{h''}_{1-i,t}u - a^{h''}_{i,t}c),$$

where $((a^{h''}_{i,h_0+1}, a^{h''}_{1-i,h_0+1}), \ldots, (a^{h''}_{i,T}, a^{h''}_{1-i,T}))$ is the outcome of s^* in $\Gamma(h'')$. By the induction assumption, we also have that, for all $i \in \{0,1\}$, for $t = h_0+1, \ldots, T$, $a^{h''}_{i,t} = 0$. Therefore,

$$u_{i,h_0}(s^*(h)) = u_{i,\Gamma(h)}(s^*). \tag{2.2}$$

Combining (2.1) and (2.2), we get that

$$u_{i,h_0}(a_i, s^*_{1-i}(h)) \leq u_{i,h_0}(s^*(h)). \tag{2.3}$$

Since (2.3) is true for all a_i, it must be the case that

$$s^*_i(h) = \arg\max_{a_i} u_{i,h_0}(a_i, s^*_{1-i}(h)) = 0,$$

which means the proposition is true for $|h| = h_0 - 1$. □

2.4 Extended Model and FITS-D Scheme

Since traditional reputation systems cannot provide SPNE solutions, in this section, we introduce a new technique and use it to significantly enhance the incentive compatibility of reputation systems. This technique is called Threat To Interfere (TTI).

The main idea of TTI is that a node can *threaten* to drop a neighbor's packets *and* to interfere with the neighbor's communications. In contrast, in traditional reputation systems, a node can only threaten to drop its neighbor's packets, which is the only way to force its neighbor to forward packets. Because the combined threat (of packet dropping and communication interference) is stronger than a single threat of packet dropping, it can be a more effective method to force the neighbor to forward packets. Note that TTI does not require real interference with communications. In fact, our analysis will show that there is no real interference at all when the system converges to a stable state.

2.4.1 Extended Model

To allow formal analysis of TTI, we extend the basic model by introducing an additional action INTERFERENCE to each node v_i, which means v_i drops all packets of v_{1-i} and interferes with the communications of v_{1-i} (using dumb packets containing its own identity). Formally, we replace the action set A_i with $A'_i = A_i \cup \{\text{INTERFERENCE}\}$. Correspondingly, the perceived action of player v_i in stage t is redefined as

$$\hat{a'}_{i,t} = \begin{cases} \text{INTERFERENCE} & \text{if } a'_{i,t} = \text{INTERFERENCE} \\ (1-p_c)a'_{i,t} & \text{otherwise,} \end{cases}$$

(We note that, in the extended model, *perceived actions* are different from *perceived probabilities*.) Furthermore, the utility of player v_i in stage t is redefined as

$$u'_{i,t} = \begin{cases} a_{1-i,t}u - a_{i,t}c & \text{if } a_{i,t} \neq \text{INTERFERENCE} \\ & \text{and } a_{1-i,t} \neq \text{INTERFERENCE} \\ a_{1-i,t}u & \text{if } a_{i,t} = \text{INTERFERENCE} \\ & \text{and } a_{1-i,t} \neq \text{INTERFERENCE} \\ u_{\text{INT}} & \text{if } a_{1-i,t} = \text{INTERFERENCE.} \end{cases}$$

where $u_{\mathsf{INT}} < 0$. Here we assume that $u_{\mathsf{INT}} < -u$. Intuitively, this means the loss caused by communication interference is greater than the benefit of forwarding packets. The utility in the entire game is still the sum of utilities in all stages:$u'_i = \sum_{t=1}^{T} u'_{i,t}$.

2.4.2 FITS-D Scheme

Now we design the FITS-D scheme using the TTI technique.

The first idea of our design is that a node following the scheme can simply choose the "worst" action appeared in the history as its new action. That is, assuming no node has ever taken the action INTERFERENCE, then a cooperative node should take the lowest forwarding probability ever appeared in the history of the game. (Note that this lowest forwarding probability in the history can be easily computed by a node: it knows all its own forwarding probabilities in the history; by the PPA, it also knows all the perceived probabilities in the history, which allows it to compute the forwarding probabilities of the other node in the history.) The advantage of such a scheme is clear: if a misbehaving node drops packets of its neighbor with a certain probability in a stage, then its cooperative neighbor drops its packets with at least the same probability in all future stages. This threat of punishment is strong enough to prevent a misbehaving node from dropping packets.

Recall the problem caused by the finite repeated game model, which we have discussed in Section 2.1. To solve this problem, we introduce an additional stage in which there is *no* data transmission, and use our TTI technique in this stage. Intuitively, we can view this stage as a brief extension period of the reputation game. In this additional stage, a cooperative node examines whether the other node was cooperative in the last stage of data transmission. If so, the cooperative node does nothing; otherwise, the cooperative node interferes with the other node's communications.

In this way, we can effectively prevent packet dropping in the last stage of data transmission (i.e., in the second last stage of the entire game), because communication interference is a strong threat to any misbehaving node. Furthermore, we do not need to worry about misbehavior in the additional stage (i.e., in the last stage of the entire game), because a misbehaving node cannot benefit from cheating in a stage that has no data transmission at all. (For detailed analysis, see Theorem 5 and its proof.)

A detailed description of the FITS-D scheme is given in Figure. 2.1.

$\triangleright$ (v_i, v_{1-i}) is a pair of neighbors.
$\triangleright$ t is the index of stage in the reputation game.

if $t = 1$,
    then $a_{i,t} \leftarrow 1$;
else if $\exists t' \in \{1, \ldots, t-1\}$ s.t. $\hat{a}_{i,t'} = \mathsf{INTERFERENCE}$
or $\hat{a}_{1-i,t'} = \mathsf{INTERFERENCE}$,
    then $a_{i,t} \leftarrow \mathsf{INTERFERENCE}$;
else if $t < T$,
    then $\rho_{i,t} \leftarrow \frac{\min_{1 \le t' \le t-1}\{\hat{a}_{i,t'}, \hat{a}_{1-i,t'}\}}{\hat{a}_{i,t-1}}$;
        $a_{i,t} \leftarrow \rho_{i,t} a_{i,t-1}$;
else if $\hat{a}_{1-i,t-1} < \min_{1 \le t' \le t-2}\{\hat{a}_{i,t'}, \hat{a}_{1-i,t'}\}$
    then $a_{i,t} \leftarrow \mathsf{INTERFERENCE}$;
else $a_{i,t} \leftarrow 0$.

Figure 2.1. FITS-D scheme

2.4.3 Analysis of FITS-D Scheme

Now we present a formal analysis of the FITS-D scheme. We show it is a SPNE solution to the packet forwarding problem. More precisely, we show that, in a SPNE, nodes forward packets with probability 1 in all stages except the last stage that does not have any data to forward.

Theorem 5. *In the extended model of reputation game (as defined in Section 2.4.1), if the FITS-D scheme is used, assuming $u > 2c$, then there is a SPNE s^* such that*

$$O(s^*) = (1,1)^{T-1}(0,0) = ((1,1),\ldots,(1,1),(0,0)),$$

which implies that, for all $i \in \{0,1\}$, $u_i(s^*) = (T-1)(u-c)$.

Proof. It is easy to see that, if all participants follow the FITS-D scheme, then the outcome is $(1,1)^{T-1}(0,0)$. Hence, denote by s^* the strategy profile defined by our FITS-D scheme; it suffices to show that s^* is a SPNE.

Let $h = ((\hat{a}_{0,1}, \hat{a}_{1,1}, \perp),\ldots,(\hat{a}_{0,|h|}, \hat{a}_{1,|h|}, \perp))$. Let $t = |h|$. Let $s_i = s_i^*|_{h \to a_i}$, where $a_i \neq s_i^*(h)$. We distinguish a number of cases to analyze the relationship between $u_{i,\Gamma(h)}(s^*)$ and $u_{i,\Gamma(h)}(s_i, s_{1-i}^*)$ (We will use notation $u_{\Gamma(h)}(s^*)$ (resp. $u_{\Gamma(h)}(s_i, s_{1-i}^*)$) instead of $u_{i,\Gamma(h)}(s^*)$ (resp. $u_{i,\Gamma(h)}(s_i, s_{1-i}^*)$) in this proof for convenience):

Case A: There exists $t' \in \{1,\ldots,t\}$ s.t. $\hat{a}_{i,t'} = \mathsf{INTERFERENCE}$ or $\hat{a}_{1-i,t'} = \mathsf{INTERFERENCE}$. In this case,

$$O_{\Gamma(h)}(s^*) = (\mathsf{INTERFERENCE}, \mathsf{INTERFERENCE})^{\mathsf{T-t}}$$
$$\Rightarrow \quad u_{\Gamma(h)}(s^*) = (T-t)u_{\mathsf{INT}},$$

and

$$O_{\Gamma(h)}(s_i, s_{1-i}^*) = (a_i, \mathsf{INTERFERENCE})(\mathsf{INTERFERENCE}, \mathsf{INTERFERENCE})^{\mathsf{T-t-1}}$$
$$\Rightarrow \quad u_{\Gamma(h)}(s_i, s_{1-i}^*) = (T-t)u_{\mathsf{INT}}.$$

Hence, $u_{\Gamma(h)}(s^*) = u_{\Gamma(h)}(s_i, s_{1-i}^*)$.

Case B: $t = 0$. In this case,

$$O_{\Gamma(h)}(s^*) = (1,1)^{T-t-1}(0,0)$$
$$\Rightarrow \quad u_{\Gamma(h)}(s^*) = (T-t-1)(u-c),$$

We have two subcases.

Case B.1: $a_i = \mathsf{INTERFERENCE}$.

$$O_{\Gamma(h)}(s_i, s^*_{1-i}) = (\mathsf{INTERFERENCE}, 1)(\mathsf{INTERFERENCE}, \mathsf{INTERFERENCE})^{\mathsf{T-t-1}}$$

$$\Rightarrow \qquad u_{\Gamma(h)}(s_i, s^*_{1-i}) = (T-t-1)u_{\mathsf{INT}} + u.$$

Since $u_{\mathsf{INT}} < -u < -c$, we can easily obtain that

$$u_{\Gamma(h)}(s^*) \geq u_{\Gamma(h)}(s_i, s^*_{1-i}).$$

Case B.2: $a_i \neq \mathsf{INTERFERENCE}$.

$$O_{\Gamma(h)}(s_i, s^*_{1-i}) = (a_i, 1)(a_i, a_i)^{T-t-2}(0, 0)$$

$$\Rightarrow \quad u_{\Gamma(h)}(s_i, s^*_{1-i}) = (T-t-2)a_i(u-c) + u - a_i c.$$

Since $u > 2c$, we can easily obtain that

$$u_{\Gamma(h)}(s^*) \geq u_{\Gamma(h)}(s_i, s^*_{1-i}).$$

Case C: $\forall t' \in \{1, \ldots, t\}$, $\hat{a}_{i,t'} \neq \mathsf{INTERFERENCE}$ and $\hat{a}_{1-i,t'} \neq \mathsf{INTERFERENCE}$, and $0 < t < T-1$. In this case,

$$\begin{aligned} & O_{\Gamma(h)}(s^*) \\ = \quad & (\tfrac{a_{i,t-1}}{\hat{a}_{i,t-1}} \cdot \min_{1 \leq t' \leq t}\{\hat{a}_{i,t'}, \hat{a}_{1-i,t'}\}, \\ & \tfrac{a_{i,t-1}}{\hat{a}_{i,t-1}} \cdot \min_{1 \leq t' \leq t}\{\hat{a}_{i,t'}, \hat{a}_{1-i,t'}\})^{T-t-1}(0, 0) \\ \Rightarrow \quad & u_{\Gamma(h)}(s^*) \\ = \quad & (T-t-1)(u-c) \cdot a_{i,t-1} \\ & \cdot \min_{1 \leq t' \leq t}\{\hat{a}_{i,t'}, \hat{a}_{1-i,t'}\}/(\hat{a}_{i,t-1}). \end{aligned}$$

We have four subcases.

Case C.1: $a_i = \mathsf{INTERFERENCE}$. Thus we have

$$\begin{aligned} & O_{\Gamma(h)}(s_i, s^*_{1-i}) \\ = \; & (\mathsf{INTERFERENCE}, \tfrac{\mathsf{a_{i,t-1}}}{\mathsf{\hat{a}_{i,t-1}}} \cdot \min_{1 \le \mathsf{t'} \le \mathsf{t}} \{\mathsf{\hat{a}_{i,t'}}, \mathsf{\hat{a}_{1-i,t'}}\}) \\ & (\mathsf{INTERFERENCE}, \mathsf{INTERFERENCE})^{\mathsf{T-t-1}} \\ \Rightarrow \quad & u_{\Gamma(h)}(s_i, s^*_{1-i}) \\ = \quad & \tfrac{a_{i,t-1}}{\hat{a}_{i,t-1}} \cdot \min_{1 \le t' \le t} \{\hat{a}_{i,t'}, \hat{a}_{1-i,t'}\} u \\ & + (T - t - 1) u_{\mathsf{INT}}. \end{aligned}$$

Clearly, $u_{\Gamma(h)}(s_i, s^*_{1-i}) < 0 < u_{\Gamma(h)}(s^*)$.

Case C.2: $a_i \neq \mathsf{INTERFERENCE}$ and $a_i > s^*_i(h)$.

$$\begin{aligned} & O_{\Gamma(h)}(s_i, s^*_{1-i}) \\ = \quad & (a_i, \tfrac{a_{i,t-1}}{\hat{a}_{i,t-1}} \cdot \min_{1 \le t' \le t} \{\hat{a}_{i,t'}, \hat{a}_{1-i,t'}\}) \\ & (\tfrac{a_{i,t-1}}{\hat{a}_{i,t-1}} \cdot \min_{1 \le t' \le t} \{\hat{a}_{i,t'}, \hat{a}_{1-i,t'}\}, \\ & \tfrac{a_{i,t-1}}{\hat{a}_{i,t-1}} \cdot \min_{1 \le t' \le t} \{\hat{a}_{i,t'}, \hat{a}_{1-i,t'}\})^{T-t-2} \\ & (0, 0) \\ \Rightarrow \quad & u_{\Gamma(h)}(s_i, s^*_{1-i}) = (T - t - 2)(u - c) \cdot a_{i,t-1} \\ & \cdot \min_{1 \le t' \le t} \{\hat{a}_{i,t'}, \hat{a}_{1-i,t'}\} / (\hat{a}_{i,t-1}) - a_i c \\ & + u \cdot a_{i,t-1} \cdot \min_{1 \le t' \le t} \{\hat{a}_{i,t'}, \hat{a}_{1-i,t'}\} / (\hat{a}_{i,t-1}). \end{aligned}$$

Hence, $u_{\Gamma(h)}(s^*) > u_{\Gamma(h)}(s_i, s^*_{1-i})$.

Case C.3: $a_i \neq \mathsf{INTERFERENCE}$ and $a_i < s^*_i(h)$, and $t < T - 2$.

$$
\begin{aligned}
& O_{\Gamma(h)}(s_i, s^*_{1-i}) \\
= \quad & (a_i, \tfrac{a_{i,t-1}}{\hat{a}_{i,t}} \cdot \min_{1\le t'\le t}\{\hat{a}_{i,t'}, \hat{a}_{1-i,t'}\}) \\
& (a_i, a_i)^{T-t-2}(0,0) \\
\Rightarrow \quad & u_{\Gamma(h)}(s_i, s^*_{1-i}) = (T-t-2)(u-c)\cdot a_i - a_i c \\
& + u \cdot a_{i,t-1} \cdot \min_{1\le t'\le t}\{\hat{a}_{i,t'}, \hat{a}_{1-i,t'}\}/(\hat{a}_{i,t-1}).
\end{aligned}
$$

Since $u > 2c$ and $t < T-2$, we can easily obtain that $u_{\Gamma(h)}(s^*) > u_{\Gamma(h)}(s_i, s^*_{1-i})$.

Case C.4: $a_i \neq \mathsf{INTERFERENCE}$ and $a_i < s^*_i(h)$, and $t = T-2$.

$$
\begin{aligned}
& O_{\Gamma(h)}(s_i, s^*_{1-i}) \\
= \quad & (a_i, \tfrac{a_{i,t-1}}{\hat{a}_{i,t-1}} \cdot \min_{1\le t'\le t}\{\hat{a}_{i,t'}, \hat{a}_{1-i,t'}\})(\mathsf{INTERFERENCE}, \mathsf{INTERFERENCE}) \\
\Rightarrow \quad & u_{\Gamma(h)}(s_i, s^*_{1-i}) = u_{\mathsf{INT}} - a_i c \\
& + u \cdot a_{i,t-1} \cdot \min_{1\le t'\le t}\{\hat{a}_{i,t'}, \hat{a}_{1-i,t'}\}/(\hat{a}_{i,t-1}).
\end{aligned}
$$

Since $u_{\mathsf{INT}} < -u < -c$, we can easily obtain that $u_{\Gamma(h)}(s^*) > u_{\Gamma(h)}(s_i, s^*_{1-i})$.

Case D: $\forall t' \in \{1, \ldots, t-1\}$, $\hat{a}_{i,t'} \neq \mathsf{INTERFERENCE}$ and $\hat{a}_{1-i,t'} \neq \mathsf{INTERFERENCE}$, and $t = T-1$. We have two subcases.

Case D.1: $s^*_{1-i}(h) = \mathsf{INTERFERENCE}$. Hence, we have

$$
\begin{aligned}
u_{\Gamma(h)}(s^*) &= u_{\mathsf{INT}} \\
&= u_{\Gamma(h)}(s_i, s^*_{1-i}).
\end{aligned}
$$

Case D.2: $s^*_{1-i}(h) = 0$. If $s^*_i(h) = \mathsf{INTERFERENCE}$, then

$$
\begin{aligned}
u_{\Gamma(h)}(s^*) &= 0 \\
&\geq u_{\Gamma(h)}(s_i, s^*_{1-i}).
\end{aligned}
$$

If $s_i^*(h) \neq \mathsf{INTERFERENCE}$, then, since $t = T - 1$, from the scheme we know that $s_i^*(h) = 0$. Hence,

$$\begin{aligned} u_{\Gamma(h)}(s^*) &= -s_i^*(h) \cdot c \\ &= 0 \\ &\geq u_{\Gamma(h)}(s_i, s_{1-i}^*). \end{aligned}$$

Hence, we always have $u_{\Gamma(h)}(s^*) \geq u_{\Gamma(h)}(s_i, s_{1-i}^*)$. So, by the One Deviation Theorem of finite repeated game, we know that s^* is a SPNE. □

2.5 FITS-I Scheme

The FITS-D scheme provides strong incentive for packet forwarding as long as the PPA is valid. However, in reality there exist scenarios in which the perceived actions of the two nodes can not be seen by both of them, i.e. the PPA is not valid.[1] In these cases, we need to use a scheme that is independent from the PPA. In this section, we develop such a scheme.

2.5.1 Scheme for PPA-Independence

The main idea to achieve PPA-independence is to use (claimed) real forwarding probabilities instead of perceived forwarding probabilities. We assume that, at the beginning of each stage t, each node v_i claims its real forwarding probability in this stage to the other node v_{1-i}. Denote this claim by $\overline{a}_{i,t}$. If v_i is cooperative, we must have $\overline{a}_{i,t} = a_{i,t}$. (Of course, if v_i is a misbehaving node, then we may have $\overline{a}_{i,t} \neq a_{i,t}$.) Using these claimed probabilities, we can establish a scheme that is similar to the

[1] In such scenarios, a node v_i still knows $\hat{a}_{1-i,t}$, but does not know $\hat{a}_{i,t}$. Note that v_i always knows its own action $a_{i,t}$, but it may have no idea about p_c and thus may not know $\hat{a}_{i,t}$.

FITS-D scheme but does not need any perceived probability, as long as we can make sure there is no false claim of forwarding probability.

To prevent players from making false claims, we divide each stage into m small time intervals. Each player is responsible for keeping a transcript of the packets it has forwarded in the *current* time interval. At the end of each time interval, with probability p_v node v_i chooses to verify the forwarding probability of v_{1-i} in this time interval. (To reduce the overall overheads of computation and communication, v_i does not verify the forwarding probability in all time intervals. Instead, v_i randomly picks some time intervals and verifies the forwarding probability in these time intervals only.) If v_i chooses to verify the forwarding probability in a time interval, it requests the transcript from v_{1-i}. Then it uses this transcript to decide whether v_{1-i} has really forwarded packets with probability $\overline{a}_{1-i,t}$. (There are various ways to design a verification algorithm for this purpose. We give one example algorithm in Section 2.5.3.) If cheating is detected, v_i punishes v_{1-i} by dropping its packets and interfering with its communications in all future stages.[2] If no cheating is detected in this time interval, or if this time interval is not chosen for verification of the forwarding probability, then the transcript can be discarded at the beginning of next time interval to save space.

The details of FITS-I scheme are shown in Fig. 2.2 and 2.3. In this scheme, we use an algorithm VerProb() to verify the claimed forwarding probabilities. This algorithm works by comparing two transcripts $\overline{X}_{1-i}$ and $X_{1-i,i}$. Here both of these two transcripts are supposed to be the packets forwarded in the current time interval by v_{1-i}. However, they are from different sources: $\overline{X}_{1-i}$ is provided by v_{1-i} (and thus should be consistent with the claimed forwarding probability of v_{1-i}), while $X_{1-i,i}$ is

[2]Note that this strong punishment is to make sure that nodes do not cheat in reporting $\overline{a}_{1-i,t}$. As shown later, when the system converges to the stable state (SPNE) under FITS-I, there is actually *no real intereference* in the system.

overheard by v_i (and thus includes collisions it hears in addition to forwarded packets). If v_{1-i} is honest, these two transcripts should be consistent (except for collisions in $X_{1-i,i}$). Even though v_{1-i} can provide a false $\overline{X}_{1-i}$ by including the packets that it has not forwarded, v_i can detect the false claim by comparing $\overline{X}_{1-i}$ and its own record $X_{1-i,i}$. We discuss how to design this algorithm in Section 2.5.3.

```
▷ (v_i, v_{1-i}) is a pair of neighbors.
▷ t is the index of stage in the reputation game.
▷ r_v = TRUE at the beginning of stage 1.

if r_v = TRUE
   then
     if t = 1,
        then a_{i,t} ← 1;
     else if ∃t' ∈ {1, ..., t-1} s.t.
        a_{i,t'} = INTERFERENCE   or â_{1-i,t'} = INTERFERENCE,
        then a_{i,t} ← INTERFERENCE;
     else if t < T,
        then a_{i,t} ← min_{1≤t'≤t-1}{a_{i,t'}, ā_{1-i,t'}};
     else if ā_{1-i,t-1} < min_{1≤t'≤t-2}{a_{i,t'}, ā_{1-i,t'}}
        then a_{i,t} ← INTERFERENCE;
     else
        a_{i,t} ← 0;
else
   a_{i,t} ← INTERFERENCE.
```

Figure 2.2. FITS-I scheme: deciding $a_{i,t}$ at beginning of stage t.

One may notice that the FITS-I scheme also uses the perceived forwarding probability $\hat{a}_{1-i,t'}$. However, this perceived probability of node v_{1-i} is always known to node v_i, regardless of whether the PPA is valid or not. So, the PPA-independence of the scheme is not affected.

2.5.2 Analysis of FITS-I Scheme

The incentive compatibility of FITS-I scheme is formally stated in the following theorem:

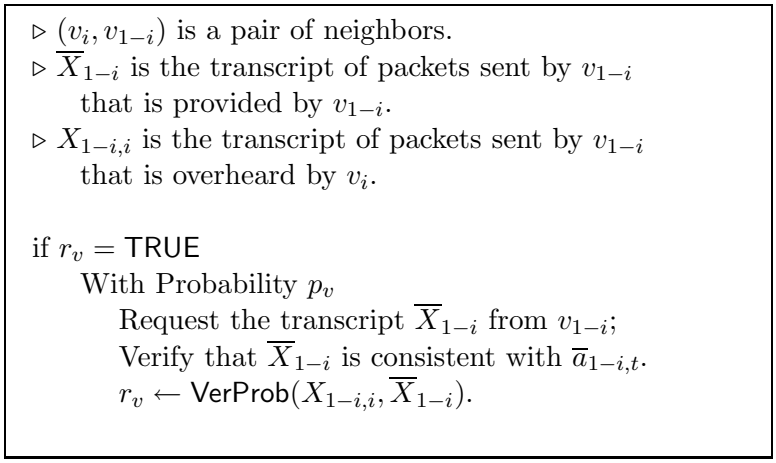


Figure 2.3. FITS-I scheme: end of a time interval in stage t.

Theorem 6. *In the extended model of reputation game, if the FITS-I scheme is used, and if* $\forall i \in \{0,1\}$, $\forall t \in \{1,\ldots,T\}$, $\overline{a}_{i,t} = a_{i,t}$, *assuming* $u > 2c$, *then there is a SPNE* s^* *such that*

$$O(s^*) = (1,1)^{T-1}(0,0) = ((1,1),\ldots,(1,1),(0,0)),$$

which implies that, for all $i \in \{0,1\}$, $u_i(s^*) = (T-1)(u-c)$.

The proof is analogous to that of Theorem 5. We omit the proof here.

One may ask whether we can have a similar formal analysis in the case when $\overline{a}_{i,t} = a_{i,t}$ does not always hold. To achieve this goal, we need to redefine the finite repeated game by adding the reports of $\overline{a}_{i,t}$ into the action space. The result would be a much more complex game. We conjecture that, in such a complex game, we will only be able to prove a much weaker variant of SPNE, rather than the standard SPNE we consider in this chapter. Hence, it would be very difficult to formally analyze the case when $\overline{a}_{i,t} = a_{i,t}$ does not always hold. In this chapter, to guarantee $\overline{a}_{i,t} = a_{i,t}$, we take a lightweight approach for preventing players from making false claims of forwarding probabilities. When this approach is used, with a certain probability making false claims will be detected and punished (as shown in Theorem 7). Consequently, a selfish node has incentives to report $\overline{a}_{i,t}$ that is equal to $a_{i,t}$.

2.5.3 VerProb: Example Algorithm for Verifying Forwarding Probabilities

Now we design the algorithm VerProb that compares $X_{1-i,i}$ with $\overline{X}_{1-i}$.

Recall $\overline{X}_{1-i}$ is the transcript provided by v_{1-i}. Therefore, it is a sequence of packets. In contrast, because $X_{1-i,i}$ is overheard by v_i, it is a sequence of packets *and collisions*. (Note that, when the packets forwarded by v_{1-i} collide with other transmissions, v_i can hear the collisions, although it cannot determine what are the collided packets.) If v_i is honest, $\overline{X}_{1-i}$ should be identical to $X_{1-i,i}$, except that some segments of $\overline{X}_{1-i}$ correspond to collisions in $X_{1-i,i}$. The restriction on these segments is that each collision of time length τ can only match with a sequence of $\frac{\tau}{\tau_{\mathsf{col}}^{\mathsf{max}}}$ to $\frac{\tau}{\tau_{\mathsf{col}}^{\mathsf{min}}}$ packets, where $\tau_{\mathsf{col}}^{\mathsf{max}}$ (resp., $\tau_{\mathsf{col}}^{\mathsf{min}}$) is the maximum (resp., minimum) length of transmission time for a packet.

The above problem can be easily reduced to the problem of *variable-length gap matching*. Hence, the main idea of the VerProb algorithm is to reduce it to variable-length gap matching and then apply the Rahman-Iliopoulos-Lee-Mohamed-Smyth (RILMS) algorithm [86]. Details of the VerProb algorithm are given in Fig. 2.4.

Theorem 7. *Let N be the total number of packets that need to be forwarded by v_{1-i} in stage t. If the algorithm* VerProb *is used, and if $\overline{X}_{1-i}$ includes $N\overline{a}_{i,t}$ packets forwarded for v_i while only $Na_{i,t}$ packets are actually forwarded for v_i (where $\overline{a}_{i,t} > a_{i,t}$), then with probability*

$$p_d \geq 1 - (1 - p_v(1 - p_c))^{\lceil \frac{N(\overline{a}_{i,t} - a_{i,t})}{n_{\mathsf{int}}} \rceil},$$

the algorithm VerProb *outputs* FALSE*, where n_{int} is an upper bound for the number of packets transmitted in a time interval.*

Proof. Let n_j be the number of packets that v_{1-i} falsely claims to be forwarded for v_i in time interval j. Clearly,

$$\sum_{j=1}^{t} n_j = N(\overline{a}_{i,t} - a_{i,t}). \tag{2.4}$$

```
▷ (v_i, v_{1−i}) is a pair of neighbors.
▷ X̄_{1−i} is the transcript of v_{1−i}'s messages
    provided by v_{1−i}.
▷ X_{1−i,i} is the transcript of v_{1−i}'s messages
    overheard by v_i.
▷ COL is the symbol for an overheard collision.
▷ VLGSearch is the RILMS algorithm.

j ← 1; l ← 1; l' ← 1;
J ← NumberOfPackets(X_{1−i,i});
while j ≤ J do
    X'[l] ← longest prefix of X_{1−i,i}[j ··· J] that
            does not contain COL;
    if X'[l] is not empty
        then l ← l + 1;
             j ← j + NumberOfPackets(X'[l]);
    if j ≤ J
        then X''[l'] ← longest prefix of
             X_{1−i,i}[j ··· J] that contains only COL;
             l' ← l' + 1;
             j ← j + NumberofCOL(X''[l']);
    else
        return VLGSearch(X̄_{1−i}, X'[1 ··· l − 1],
                         {COLPack(X''[j])}_{j=1}^{l'−1});
X'[l] ← empty string;
return VLGSearch(X̄_{1−i}, X'[1 ··· l],
                 {COLPack(X''[j])}_{j=1}^{l'−1});

COLPack(x)
    L_1 ← 0; L_2 ← 0;
    for each COL in x
        τ ← Time of COL;
        L_1 ← L_1 + ⌈ τ / τ_col^max ⌉;
        L_2 ← L_2 + ⌊ τ / τ_col^min ⌋;
    return(L_1, L_2);
```

Figure 2.4. VerProb: verification of forwarding probability.

Node v_i detects cheating in this time interval if the algorithm VerProb is executed in this time interval, and at least one of these n_j packets is claimed to be sent at a time when v_i does not hear a collision. So the probability of detecting cheating in

time interval j is

$$p_{d,j} = p_v(1 - p_c^{n_j}).$$

Recall m is the number of time intervals in stage t. Hence, the probability of detecting cheating in the entire stage is

$$\begin{aligned} p_d &= 1 - \prod_{j=1}^{m}(1 - p_{d,j}) \\ &= 1 - \prod_{j=1}^{m}(1 - p_v(1 - p_c^{n_j})) \\ &\geq 1 - (1 - p_v(1 - p_c))^{\lceil \frac{N(\overline{\sigma}_{i,t} - a_{i,t})}{n_{\mathsf{int}}} \rceil}. \end{aligned}$$

The last inequality is due to (2.4) and the pigeonhole principle. □

2.6 Evaluations

In the previous sections, we have presented the two schemes of FITS. To evaluate the performance of FITS, we implement these two schemes in the network layer using wireless network simulator GloMoSim [46] and perform three sets of experiments:

- The first set of our experiments study the punishment of FITS schemes on misbehaving nodes. The objective is to verify that FITS schemes effectively punish misbehaving nodes and thus give nodes incentives to cooperate in packet forwarding.

- The second set of our experiments compare FITS with an existing reputation system, in terms of their convergence to stable states. The objective is to verify that FITS converges to a stable state with significantly higher forwarding probabilities.

- The third set of experiments examine the effect of interference, in case it is really used for punishment.

- The fourth set of our experiments measure the efficiency of FITS schemes in terms of computation and communication overheads.

In the remainder of this section, we describe the settings and results of our experiments.

2.6.1 Settings

In our experiments, the MAC layer is based on IEEE 802.11; the Dynamic Source Routing protocol (DSR) is used for routing. We use the two-ray propagation path-loss model. The radio transmission power level is at 12 dBm and the radio to noise ratio threshold is set to 8.0 dB. The network has a bandwidth of 2 Mbps.

Within an area of 2000 by 2000 meters, 50 nodes are randomly distributed. The topology of the network is shown in Fig. 2.5.[3] We generate the traffic of 50 sessions. Every node is a session source and the destination of each session is randomly picked. Throughout the simulation time each source transmits packets at a constant bit rate of 2 packets/s with the packet size being 512 bytes. For FITS-I scheme, we set $m = 20$ (which means there are 20 time intervals in each stage) and $p_v = 0.2$.

2.6.2 Punishment on Misbehaving Nodes

In the first set of experiments, we study the punishment that misbehaving nodes receive in FITS schemes.

We randomly pick 5 nodes to be the misbehaving nodes, which means these nodes do not implement FITS and drop packets that are not destined to them with a fixed probability. All the other nodes are cooperative, i.e., they use FITS schemes to forward packets for other nodes. There are 8 stages in the reputation game and each stage lasts for 100 seconds. The entire simulation time is 800 seconds. The results described below are the average of 200 runs.

[3]Here we assume that during each of our data sessions, network topology remains the same.

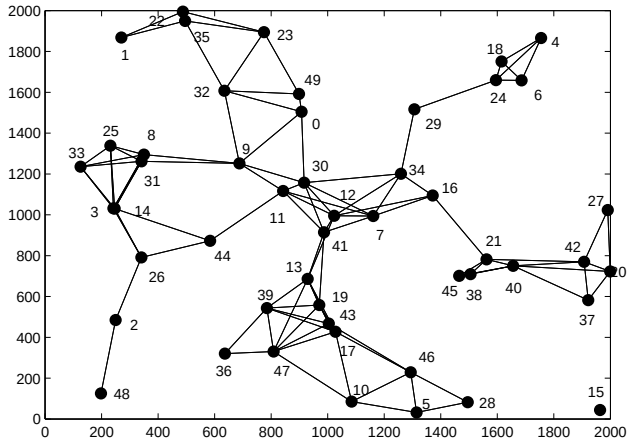


Figure 2.5. Topology of the randomly generated network. Nodes are labeled with their IDs. A link between two nodes represents that these two nodes are within each other's transmission range.

Fig. 2.6 shows the utilities of misbehaving nodes when they drop packets in the FITS-D scheme (resp., FITS-I scheme) in subfigure (1) (resp., (2)). Here, we use $u = 3.0$ and $c = 1.0$ when calculating the utility. From Fig. 2.6 we observe that, when the packet dropping probability of misbehaving nodes grows, their utilities obtained in the reputation games decrease quickly. This reflects that the FITS schemes effectively punishes the misbehaving nodes in terms of utilities.

Another way to study the punishment on misbehaving nodes is to measure the message success rates (the percentage of packets from the source node that successfully arrive at the destinations) of the misbehaving nodes. Fig. 2.7 shows the results of our measurements in the two FITS schemes. For comparison, we also include the average message rates of cooperative nodes in these figures. We can see that the message success rates of all misbehaving nodes are fast decreasing (as the packet dropping probability grows), while the message success rates of cooperative nodes only decrease slightly.

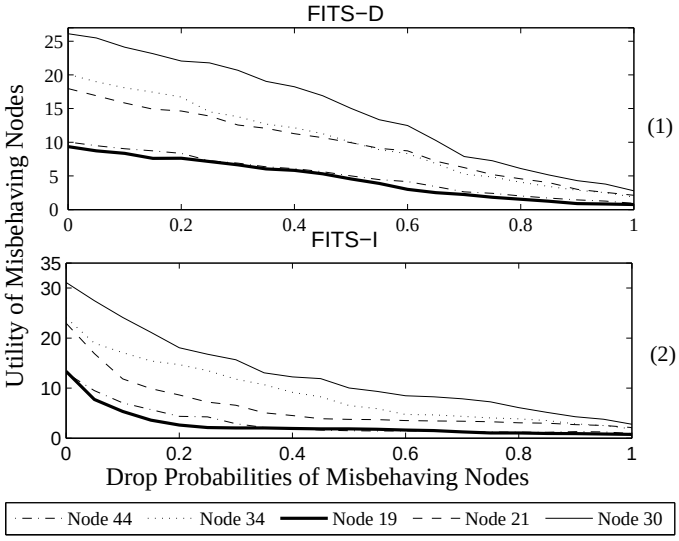


Figure 2.6. Utilities of misbehaving nodes in FITS schemes.

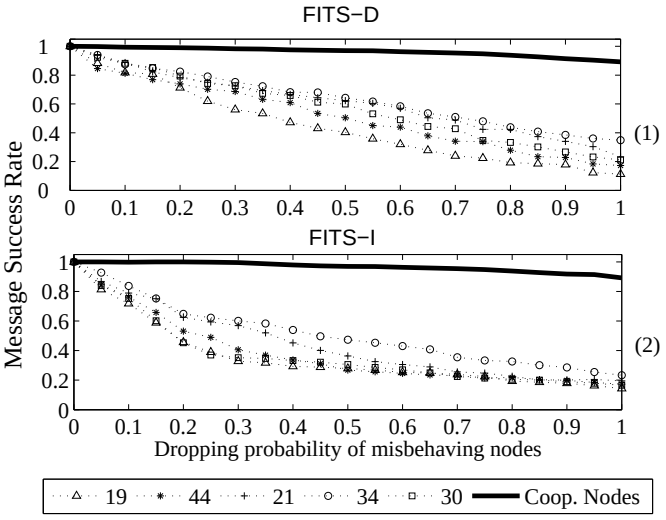


Figure 2.7. Message success rate of misbehaving nodes and cooperative nodes in FITS schemes.

2.6.3 Comparison of Stable States

Our second set of experiments is to compare FITS with an existing reputation system, DARWIN [60], and to observe their convergence to stable states. In this set of experiments, nodes are allowed to choose any stage of the game to start dropping packets. A stable state is a state such that for each node, changing its strategy in any way cannot increase its utility. To find the stable state, in this set of experiments we let nodes go through a sequence of reputation games and allow the nodes to change their strategies between games. A node stops changing its strategy when it finds that there is no way to improve its utility by changing the current strategy. When the involved nodes stop changing their strategies, a reputation system is in its stable state.

We randomly pick a pair of neighbors, Node 24 and Node 29, and observe how their average forwarding probabilities for each other evolve in a sequence of reputation games. These two nodes are selfish so try to maximize their utilities by dropping packets. At the beginning of each experiment, each node forwards packets with probability 1. In the sequence of reputation games, a node can make an attempt to change its strategy as follows: it randomly picks a stage and drops all the packets in and after that stage. Then it compares the utility it gets in this game with the utility in the previous game. If there is a loss in utility, it goes back to the old forwarding strategy, and stays with that strategy for 3 games before it makes another attempt. If there is a gain in utility, it is happy for that and thus stays with the new forwarding strategy for 20 more games before it makes the next attempt to drop more packets.

We test FITS and DARWIN, respectively, in the above experiments. Each stage is 30 seconds long and each game has 5 stages. Each experiment lasts for 500 minutes, so that both reputation systems are guaranteed to converge to stable states.

Fig. 2.8 shows the results for DARWIN, FITS-D and FITS-I in subfigures (1), (2) and (3) respectively. The forwarding probability of each node is the average

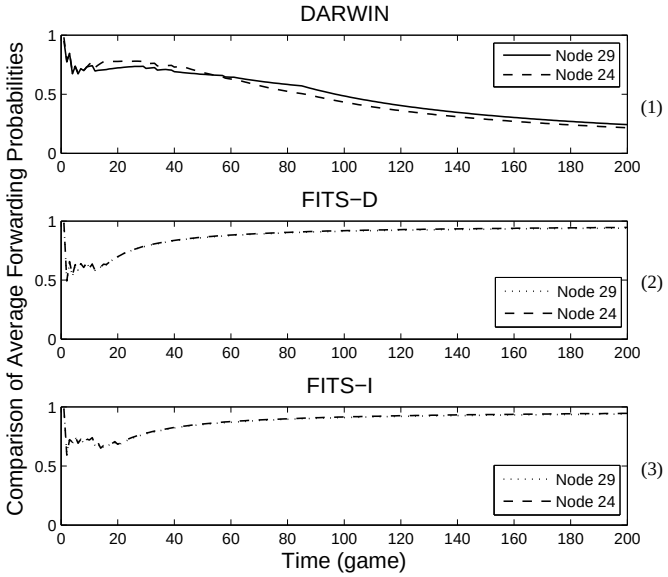


Figure 2.8. Comparison of average forwarding probabilities of DARWIN and FITS as system converges.

forwarding probability in the sequence of games that the node has been through from the beginning. In subfigure (1) we can see that the average forwarding probabilities of both Node 29 and Node 24 decrease as the system evolves. It implies that when the system evolves, the nodes can find better strategies of dropping packets rather than forwarding them with probability 1.

In contrast, in the two FITS schemes, as shown in subfigures (2) and (3), average forwarding probabilities increase to a constant close to 1, after the oscillations in the first few games. In fact, at the beginning the nodes make all possible attempts, and find no way to increase their utilities by dropping packets (because they are punished by FITS). Consequently, in the rest of the experiment they stay with the strategy to forward all packets.

2.6.4 Effect of Interference

This set of experiments are to examine the effect of interference on cooperative nodes, when interference is used to punish misbehaving nodes. We first measure the

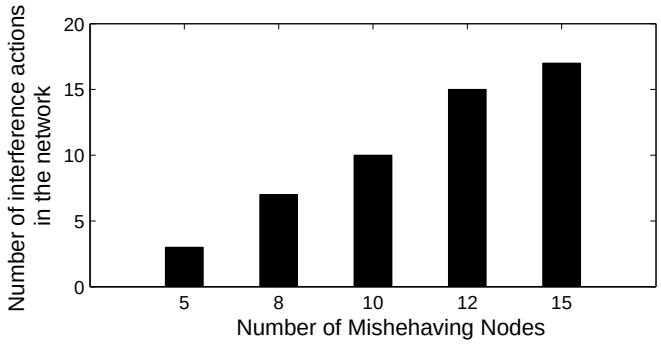


Figure 2.9. Number of Interference Actions introduced by FITS vs. Number of Misbehaving Nodes.

number of interference actions taken by cooperative nodes, for different number of misbehaving nodes in the network. Then we measure the average message success rates of the cooperative nodes in their own sessions in the following two settings: In the first setting, our FITS schemes are used; in the second setting, a modification of FITS is used in which there is definitely no interference. We compare the cooperative nodes' average message success rates in these two settings to show the effect of interference on message success rates. In this set of experiments the misbehaving nodes drop packets with probability 0.5.

In Fig. 2.9, we show the number of interference actions taken by cooperative nodes in the network, when there are 5 to 15 misbehaving nodes. From our schemes, it is easy to see that the number of interference actions taken by the cooperative nodes in FITS-D scheme is the same as in FITS-I scheme. So we only use one figure to illustrate the results. We can see that the number of interferences actions is below 20 when there are no more than 15 misbehaving nodes in the network.

Fig. 2.10 gives a comparison of cooperative nodes' average message success rates in the two settings, i.e., with and without interference introduced by FITS. We vary the number of misbehaving nodes in the network from 5 to 20. From the figure, it is clear that, when there are more misbehaving nodes, the message success rate becomes

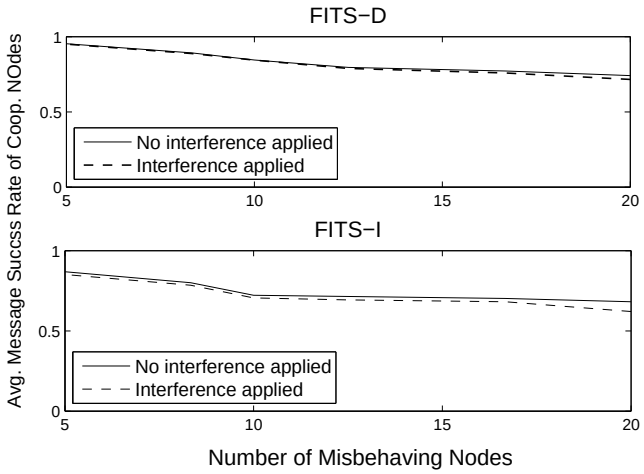


Figure 2.10. Message Success Rates of Cooperative Nodes in Their Own Sessions: FITS vs. No Interference.

lower. However, the loss in average message rate caused by interference introduced by FITS is very small.

2.6.5 Efficiency

In the third set of experiments, we measure the efficiency of FITS.

Fig. 2.11 shows the computation overheads of the two FITS schemes. Here by computation overhead we mean the average extra computation time needed by FITS for each node in each session, when the system is in the stable state. In this set of experiments, we have 1600 packets in each session. Fig. 2.11 shows that the computation overhead of FITS-D scheme always remains below 3 milliseconds. The computation overhead of FITS-I scheme is slightly higher, but it is still less than 3.5 milliseconds.

Clearly, in addition to computation overheads, FITS also has communication overheads—the time to send and receive control packets for the FITS schemes. However, for the FITS-D scheme, the communication overheads are only several microseconds per session. For the FITS-I scheme, the communication overheads are several hundred microseconds per session. Consequently, compared with computation overheads, communication overheads can be ignored.

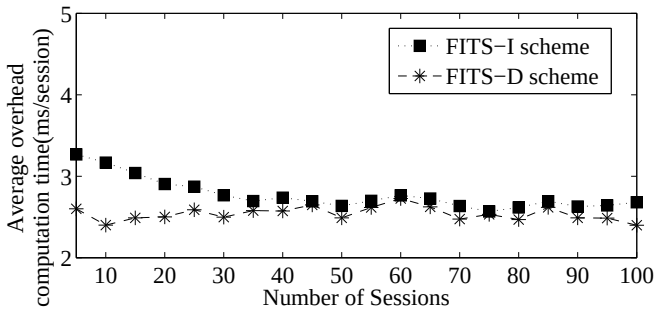


Figure 2.11. Computation overheads of FITS schemes.

2.7 Summary

Reputation system is an important approach to solve the incentive compatibility problem in packet forwarding of wireless ad hoc networks. In this chapter, we present the *first* formal study of reputation system in the model of *finite* repeated game. We believe this is a more realistic model for reputation system and thus our results have significant practical implications.

The first result we obtain is the impossibility of building a SPNE solution using traditional reputation systems. This result implies that we must introduce a new technique if we want to establish SPNE solutions for our problem.

Then we introduce the TTI technique and use it to build FITS, our new reputation system. FITS provide strong incentive compatibility for nodes to cooperate in packet forwarding. More precisely, there is a SPNE in which nodes forwards all packets. This is proved theoretically and verified by experiments.

We note that our work has only addressed one aspect of reputation systems, and left out many other aspects. For example, we have assumed a network of fixed topology. In contrast, there are also networks with dynamically changing topologies in reality. In these networks, building a reputation system is much more challegining. It would be non-trivial to establish finite-time reputation systems with provable in-

centive compatibility in these networks. Hence, this is an interesting topic for future study.

We also note that, besides threat of interference, there are other techniques that can possibly be used in building finite-time reputation systems. One such technique is that a node shuts down its connection with a neighbor whenever it detects that the neighbor is dropping its packets. Again, it is non-trivial to use such techniques to establish a complete reputation system and prove the incentive compatibility. Hence, we also leave this to future explorations.

CHAPTER 3

PACKET FORWARDING COOPERATION IN NETWORK-CODING-BASED WIRELESS NETWORK[28]

3.1 Background and Motivation

Recently, wireless mesh networks [6, 24, 77, 81] have been widely deployed to provide broadband network access to schools, communities, and participants of various events. It is very challenging and highly important to improve the performance of wireless mesh networks so that the throughput scalability of such networks can meet the needs of different users. One way to achieve significantly better performance for wireless mesh networks is to apply a technique called *network coding* [57, 64, 21, 62, 63]. Unlike in conventional networks, in wireless networks using network coding, intermediate nodes do not store and forward the same packets as sent by the source node. In stead, intermediate nodes forward new coded packets computed by themselves from the packets they have received. Hence, the data is actually "mixed" at each intermediate node before it is forwarded. This idea of mixing data at intermediate nodes takes advantage of the broadcast nature of wireless transmissions, and achieves great performance gains.

Many wireless mesh networks have user-contributed wireless devices as their nodes. Since users normally have their own interests, *economic incentives* become a crucial problem. A selfish or economically rational user may let her wireless device deviate from the communication protocol, as long as the deviation is beneficial to herself. However, this deviation may harm the network's performance, or even lead the network to stop functioning in the worst case. Therefore, we need to make the com-

munication protocol *incentive compatible*, so that nodes have incentives to faithfully follow the protocol.

In this chapter, we consider the incentive compatibility in wireless mesh networks using network coding. To be concrete, we assume that the network coding system used is MORE [21]. (In fact, our results can be adapted to some other network coding systems like MIXIT [63], through moderate modifications.) In a wireless mesh network using MORE, incentive compatibility is needed in at least two fundamental aspects: *routing* and *packet forwarding*. Here routing refers to the procedure of computing the number of transmissions each involved node should make for a data packet; packet forwarding refers to the procedure after routing that actually transmits packets from the source to the destination. These are two closely related, but completely different procedures. The incentive compatible routing problem in wireless networks using MORE has been studied by Wu et al. [109]. They propose a protocol that gives nodes incentives to honestly measure and report link loss probabilities in the routing procedure, and prove that following the protocol in the routing procedure is to the best interest of user nodes. Nevertheless, incentive compatible packet forwarding in the same kind of wireless networks has not received sufficient attention.

The main objective of this chapter is to solve the incentive compatible packet forwarding problem in wireless mesh networks using a network coding system like MORE. We note that the incentive compatible packet forwarding problem has been studied extensively in the context of conventional wireless networks, i.e., wireless networks *not* using network coding. A lot of solutions have been proposed, e.g., [76, 19, 20, 119, 59, 93]. However, we emphasize that these existing solutions for conventional wireless networks cannot be used for wireless networks using MORE. For example, consider one such solution, Sprite [119]. In Sprite, in order to stimulate cooperation in packet forwarding, the source node makes payments to intermediate nodes along *the path to the destination* (which is usually the shortest path). For each packet orig-

inally sent by the source, the amount paid to each node is decided by whether *this particular packet* has been received by the destination, and whether the *next hop node* along the path reports having received this particular packet. In contrast, in a wireless mesh network using MORE, for at least three reasons we cannot use Sprite: (a) Packets are not forwarded along a predetermined *path* from a source to a destination. (b) Given an intermediate node, there is no well defined *next-hop node*. (c) Because intermediate nodes only transmit newly computed coded packets, it is nontrivial to decide the *correspondence relationship* between a packet sent by the source node and a packet transmitted by an intermediate node. Consequently, we have to look for a new solution for the incentive compatible packet forwarding problem in wireless mesh networks using MORE.

In wireless mesh networks using MORE, the problem of incentive compatible packet forwarding can be described as follows: Suppose that the routing procedure has already computed the number of transmissions each node should make in order to forward a packet from a source node to a destination. We need to design an incentive scheme that stimulates nodes to faithfully follow the protocol and make exactly the number of transmissions computed by the routing procedure.

This problem is technically challenging both in the economic aspect and in the security aspect. To address the two-fold challenges, we use novel techniques from game theory and cryptography as we briefly describe below.

The first technical challenge is in the economic aspect: It is nontrivial to find an economic method that gives nodes incentives to make the right number of transmissions—as far as we know, there is no existing method in game theory that we can directly apply. To meet this challenge, we use game theoretic techniques to design a novel formula for paying packet forwarders. As long as this payment formula is enforced, we can guarantee that it is to the best interest of each forwarder to make the right number of transmissions.

The second technical challenge is in the security aspect: The enforcement of the payment formula mentioned above requires monitoring the transmissions each node has made to forward a packet, and this monitoring task must be carried out by some other node(s). However, if the latter node(s) do not report their results of monitoring correctly, we will not be able to calculate the right amount of payment that should be paid to the former node, and thus the former node may lose its incentives to follow the protocol. To address this challenge, we apply a combination of game theoretic and cryptographic techniques to allow nodes to punish each other for misbehavior like making incorrect reports. In this way, we can guarantee that our payment formula is properly enforced.

In summary, we have the following major contributions in this chapter.

- We are the *first* to study the incentive compatible packet forwarding problem in the context of wireless mesh networks with network coding.

- To solve this problem, we use *novel* techniques to address the technical challenges and design an incentive scheme, *INPAC*. We formally prove that, if INPAC is used, then following the protocol faithfully is a subgame perfect equilibrium.

- To make INPAC more practical, we also provide an extension of INPAC in which two improvements are achieved: (a) an online authority is no longer needed; (b) the computation and communication overheads are reduced.

- To guarantee that INPAC can be effectively used in practice, we consider possible security attacks on INPAC and discuss defenses against them.

- We have completely implemented INPAC on the Orbit Lab testbed [92]. Our experimental evaluation results verify the incentive compatibility of INPAC and demonstrate that it is efficient.

The rest of this chapter is organized as follows. In Section 3.2, we describe the network model; since our work assumes an existing network coding system, MORE, we also briefly review MORE. In Section 3.2, we design the INPAC basic scheme. In Section 3.3, we present a formal incentive analysis of the INPAC basic scheme. We present the INPAC extended scheme in Section 3.4. We discuss two possible security attacks and the defenses against them in Section 3.5. Our evaluation results are described in Section 3.6. We conclude in Section 3.7.

3.2 Technical Preliminaries

Consider a wireless mesh network that has a set V of nodes. For $v_i, v_j \in V$, we denote by (v_i, v_j) the wireless link from node v_i to node v_j. Let $\epsilon_{i,j}$ be the loss probability of this link (v_i, v_j). So, if a packet is sent by node v_i, then node v_j can receive it with probability $1 - \epsilon_{i,j}$.

We assume that this wireless mesh network uses the network coding system MORE [21], and we will design our incentive scheme based on MORE. For completeness, we briefly review the packet forwarding procedure of MORE and some related terminologies.

Brief Review of MORE When a source node S sends packets to a destination node D, MORE works as follows:

Source Node: The source node S sends the packets in batches, where each batch has K *native* (i.e., uncoded) packets $\mathsf{NP}_1, \mathsf{NP}_2, \ldots, \mathsf{NP}_K$. S does not directly send these native packets; in stead, it sends *coded* packets, where each coded packet CP_j is a random linear combination of native packets: $\mathsf{CP}_j = \sum_{i=1}^{K} \mathsf{CV}_{ji}\mathsf{NP}_i$. The vector $\vec{\mathsf{CV}}_j = (\mathsf{CV}_{j1}, \mathsf{CV}_{j2}, \ldots, \mathsf{CV}_{jK})$ is called the *coding vector* of the coded packet CP_j. A MORE header is attached to each coded data packet, which contains the batch number, the coding vector, the source and destination addresses, and a list of (potential) forwarder nodes.

The list of forwarders is decided by S using the ETX metrics [37]. For $v_i, v_j \in V$, the ETX distance from v_i to v_j is denoted by $\mathsf{dist}(v_i, v_j)$. Intuitively, this means the expected number of transmissions to deliver a packet from v_i to v_j is $\mathsf{dist}(v_i, v_j)$. Given the destination node D, we say v_i is a *downstream node* of v_j if $\mathsf{dist}(v_i, D) < \mathsf{dist}(v_j, D)$. The source S chooses all its downstream nodes as forwarders, and orders them in the forwarder list according to their ETX distances to D.

Forwarder: When a node v_i hears a data packet, it checks whether it is in the packet's list of forwarders. Then, it checks whether the packet is *innovative* (i.e., linearly independent from all previous packets in the same batch that v_i has heard). If so, v_i makes a number of transmissions to forward this packet, where each packet transmitted by v_i is a random linear combination of all packets it has heard from this batch.

The number of transmissions v_i needs to make is precomputed in the routing procedure of MORE. We denote this number by $t_i^\star$. We assume that nodes follow the protocol faithfully in the routing procedure. Our main objective of this chapter is to guarantee that each forwarder v_i will have incentives to make $t_i^\star$ transmissions for forwarding each data packet.

Destination Node: The destination D counts the number of innovative packets it has received. When it has received K innovative packets in the same batch, it sends an acknowledgement (which stops all nodes from transmitting packets in this batch) and decodes the received packets.

For further details of MORE, please refer to [21].

System Architecture The overall architecture of INPAC consists of a number of wireless nodes, on which MORE is implemented, and a central authority, called *Credit Clearance Center* (CCC). Note that having a central authority like the CCC is a standard assumption for incentive mechanisms in wireless networks (e.g., [7, 106, 105, 68]). We assume that the CCC issues a certificate to each node in the wireless mesh network and maintains an account of *credit* (i.e., *virtual currency*) for it, just like a

central bank; so all the financial transactions between nodes will be cleared at the CCC. This authority can be either online or offline. For conceptual simplicity, in the basic scheme of INPAC as described in Section 3.2, we assume an online authority. In Section 3.4, we present the INPAC extended scheme in which only an offline authority is needed.

When a node forward packets, it will receive payments from the source nodes of these packets as rewards. In other words, forwarders get credit for their forwarding services and source nodes loses credit for receiving these services. In order to prevent nodes from making false claims about their forwarding services, we require that their downstream nodes submit reports to the CCC as evidence of such forwarding services. Details about how to guarantee correct reports and how the CCC processes them are presented in Sections 3.2 and 3.4.

sectionDesign of INPAC Basic Scheme

Given the network model and the system architecture, we now design an incentive scheme—the INPAC basic scheme, which stimulates cooperation in packet forwarding. Just like many existing incentive schemes in wireless networks (e.g., [7, 105, 120, 106, 121, 68]), INPAC uses *credit* to simulate cooperation. However, as we describe below, INPAC uses novel techniques that have never been used in existing schemes.

3.2.1 Main Ideas of the Design

To develop the main ideas of our INPAC basic scheme, let us consider a node v_i, which receives a packet that it is supposed to forward. Node v_i needs to decide the number of transmissions to make in order to forward this packet. Each of the transmissions v_i makes induces a cost c_i. However, the source node S will also make a payment to compensate v_i for the transmissions. For node v_i, the utility of making these transmissions equals the received payment minus the induced costs.

Recall that our objective is to guarantee that v_i has incentives to make exactly $t_i^\star$ transmissions, where $t_i^\star$ is computed by the routing procedure of MORE. To achieve this objective, we need to carefully design a payment formula, such that v_i maximizes the utility when it makes exactly $t_i^\star$ transmissions.

Payment Formula The first difficulty in designing the payment formula is that, in practice, no one except v_i itself can count precisely how many transmissions v_i actually makes; so, if the payment to v_i is based on the number of transmissions made by v_i, then there is no way to enforce the payment in reality. To sidestep this difficulty, we propose that node v_i should be paid in a constant amount p_i for *each packet received by at least one of its downstream nodes*. In this way, the incentive scheme can be enforced as long as the following two conditions are satisfied: (a) Every downstream node correctly reports the transmissions it has received from v_i. (b) There is a formula for calculating p_i, which does not need the number of transmissions actually made by v_i or any other node.

Now we develop a formula for calculating p_i under the assumption that every node correctly reports to the CCC the transmissions that it has heard as a downstream node. (After developing the formula for p_i, we will study the case in which this assumption does not hold.)

When v_i makes t_i transmissions, the expected amount of payment v_i receives is

$$\bar{p}_i(t_i) = (1 - \epsilon_i^{t_i})p_i,$$

where

$$\epsilon_i = \prod_{\mathsf{dist}(v_h,D)<\mathsf{dist}(v_i,D)} \epsilon_{i,h} \tag{3.1}$$

is the probability that a packet sent by v_i is not received by any downstream node[1]. Hence, v_i's utility of making t_i transmissions is

$$\begin{aligned}\bar{u}_i(t_i) =& \bar{p}_i(t_i) - t_i c_i \\ =& (1-\epsilon_i^{t_i})p_i - t_i c_i, \end{aligned} \tag{3.2}$$

where c_i is v_i's cost of making one transmission. From the first order derivative of $\bar{u}_i$, considering t_i as the single variable, we can easily find that $\bar{u}_i(t_i)$ is maximized when

$$t_i = \frac{\ln -\frac{c_i}{p_i \ln \epsilon_i}}{\ln \epsilon_i}.$$

Since our objective is that $\bar{u}_i(t_i)$ is maximized when $t_i = t_i^\star$, we need that

$$t_i^\star = \frac{\ln -\frac{c_i}{p_i \ln \epsilon_i}}{\ln \epsilon_i}.$$

Solving this equation, we get the formula for p_i:

$$p_i = \frac{c_i}{\epsilon_i^{t_i^\star} \ln \frac{1}{\epsilon_i}}, \tag{3.3}$$

which will encourage each node to make the required number of transmissions in order to maximize its utility.

Preventing Incorrect Reports The above derivations are under the assumption that downstream nodes correctly report the transmissions they have received. Thus we need additional measures to prevent downstream modes from cheating in their reports about transmissions.

[1]Recall that the values of these $\epsilon_{i,h}$ are measured in MORE. We assume the measured values are correct

There are two types of possible cheating in downstream nodes' reports: *over-reporting* (i.e., reporting transmissions that they have not actually received) and *underreporting* (i.e., not reporting transmissions they have received).[2]

To prevent overreporting, we propose that, before any node v_i sends any data packet, v_i should attach its own signature on the batch number and the coding vector to the packet. When the downstream nodes report v_i's transmissions that they have received, they must present v_i's signatures to the CCC as evidence.

To prevent underreporting, we propose that v_i punishes any downstream node that underreports the transmissions from itself. Specifically, for any downstream node v_j, using the link loss probability $\epsilon_{i,j}$ and the number of transmissions v_i has made, node v_i can easily calculate the number of transmissions v_j should hear during a time interval.[3] Hence, by comparing this calculated number, with the number of transmissions v_j has reported to the CCC, node v_i can find out whether v_j has underreported transmissions from itself. If v_j has, then v_i punishes v_j by disallowing v_j to forward any future packets sent by v_i.

To implement this punishment, we propose that v_i encrypts its future data packets using a key unknown to v_j, but known by all other downstream nodes. Note that v_i's signatures on batch number and coding vector are parts of the encrypted cleartexts. In this way, even if v_j forwards these packets, it will not be able to replace v_i's signature with its own, and thus will not get paid for forwarding these packets.[4] Other downstream nodes are not affected and still can forward these packets and get payments. (In Section 3.2.3, we describe a key setup that satisfies the above

[2]In fact, there is also a possibility that overreporting is mixed with underreporting, which can be easily prevented using our approaches for preventing overreporting and underreporting.

[3]We assume that the number of transmissions v_i makes is sufficiently large during the time interval, so that the number of transmissions v_j hears converges to its mathematical expectation.

[4]Node v_j may be able to figure out the data in some of these packets by looking at packets forwarded by other downstream nodes. However, in this case, forwarding the former packets is no more than forwarding the latter packets. Overall v_j still loses profits in forwarding some packets.

requirement.) We stress that, with the encryptions and decryptions introduced by this punishment, our total overheads remain small (see Section 3.6 for the experimental evaluation results), because we use a *symmetric key* cryptosystem.

Preventing Punishment Abuse Given the punishment power as described above, node v_i may punish a downstream node that does not make incorrect reports. To prevent such abuse, we propose that each node monitors its upstream nodes. If v_j finds that its upstream node v_i punishes itself while v_j itself has not made any incorrect report, v_j stops reporting any transmissions it has received from v_i. Consequently, v_i is "deterred" from punishing v_j unless v_j has underreported its transmissions.

So far we have intuitively explained our main ideas in the design of INPAC basic scheme. For precise and formal analysis of why these ideas can work, please see Section 3.3.

3.2.2 INPAC Basic Scheme

Putting together the ideas we have discussed in Section 3.2.1, we obtain the INPAC basic scheme which stimulates nodes' cooperation in packet forwarding, as described below.

We assume that the communications between the CCC and any other node use a reliable protocol, such that lost packets are always retransmitted. We also assume that the source node S submits a signed copy of the forwarder list to the CCC, so that the CCC knows the upstream/downstream relationships among nodes.

The INPAC basic scheme consists of two parts: nodes' operations and the CCC's algorithm.

Nodes' Operations In the INPAC basic scheme, nodes have two types of operations: *regular operations* on data packets and *periodic operations*.

Fig. 3.1 lists the regular operations for processing a data packet.

INPAC Basic Scheme – Regular Operations

▷ *batch_no*: the batch number of a packet.
▷ *code_vec*: the coding vector of a packet.
▷ ID_{v_i}: the identity of a node v_i.
▷ SIG_{v_i}: a node v_i's signature on $(batch\_no, code\_vec)$.

Source Node: Same as the source node's operations in MORE. In addition, the source node S attaches an INPAC header to each outgoing data packet. The INPAC header contains SIG_S.

Forwarders: When node v_i receives a data packet from an upstream node v_j for which v_i is in the forwarder list, v_i does the follows:

1. If the data packet is encrypted using a key known by v_i, v_i decrypts it; if the data packet is encrypted using a key unknown to v_i, v_i discards it and remembers ID_{v_j}.
2. Node v_i verifies SIG_{v_j}.
3. Node v_i checks whether the coding vector is linearly independent from the previous packets in the same batch sent by v_j. If so, v_i generates a new record $(ID_{v_j}, batch\_no, code\_vec, SIG_{v_j})$ and keeps it.
4. Node v_i checks whether the coding vector is linearly independent from *all* previous packets in the same batch received by v_i (i.e., whether it is *innovative*). If it is, then v_i makes transmissions as specified in MORE. But before making these transmissions, v_i replaces SIG_{v_j} with its own signature and encrypts the packet if in step 1 the packet was decrypted.

Destination Node: Same as the destination's operations in MORE.

Figure 3.1. INPAC Basic scheme - Regular Operations on Packets

Fig. 3.2 lists the three types of periodic operations of each node v_i. Note that each type of periodic operations may have a different cycle according to the system requirements.

INPAC Basic Scheme – Periodic Operations

1. ***Submitting Report:*** Node v_i submits a report to the CCC, which contains all the records v_i created in the most recent cycle.

2. ***Monitoring Downstream Nodes:***
For each downstream node v_j, v_i compares the number of its own transmissions that v_j has reported to the CCC in the most recent cycle with the estimated number of transmissions that v_j should report. If v_i finds that v_j has underreported its own transmissions, v_i does the follows:

- Before forwarding each future packet, v_i encrypts $[SIG_{v_i}, payload]$, using key k_{-j} (see Section 3.2.3 for how to compute k_{-j});
- Node v_i replaces its locally stored value of $\epsilon_{i,j}$ with 1, and recalculates $t_i^\star$ using the algorithm in MORE.[4]

3. ***Monitoring Upstream Nodes:***
Node v_i checks, for each upstream node v_j, whether v_j has ever encrypted packets using a key unknown to itself. If so, v_i stops making records for v_j's transmissions in the future.

Figure 3.2. INPAC Basic Scheme - Periodic Operations

CCC's Algorithm After a node v_j submits a report to the CCC, the CCC processes the report as follows, in order to clear transactions:

1. Verify all signatures in the report.
2. Verify that all coding vectors for the same batch and the same sender are linearly independent, and that they are linearly independent from all coding vectors for the same batch and the same sender in previous reports submitted by v_j.

3. For each sender identity ID_{v_i} in the report, verify that v_i is an upstream node of v_j.

4. Mark the verified records as submitted by v_j and keep them.

5. For each record in the report, check whether its coding vector is linearly independent from all previous coding vectors in the same batch for which its sender has been paid. If so, pay its sender (say v_i) the amount of $p_i = \frac{c_i}{\epsilon_i^{t_i^\star} \ln \frac{1}{\epsilon_i}}$ from the account of the source node S and mark the record with "paid v_i".

3.2.3 Key Setup for Punishing Downstream Nodes

Suppose that node v_i would like to punish its downstream node v_j. We use the Akl-Taylor technique [5] to establish a key setup. This key setup allows v_i to compute a key k_{-j}, such that k_{-j} can be derived by any node except v_j. In this way, v_i can punish v_j by encrypting future packets using the symmetric key k_{-j}.

Let $N = Q_1 Q_2$ be an RSA modulus, where Q_1 and Q_2 are two large primes. Suppose that $k_0 \in Z_N^*$ is confidential to *all* nodes (i.e., no node knows k_0). We assume that each node v_i is preloaded with a large prime P_i and $k_i = k_0^{P_i}$. Node v_i keeps k_i secret and makes P_i public.

The key k_{-j} for punishing v_j can be computed as

$$k_{-j} = k_i^{\prod_{h \neq i,j} P_h}.$$

It is easy to see that, for any $i' \neq j$,

$$k_{-j} = k_i^{\prod_{h \neq i,j} P_h} = k_0^{\prod_{h \neq j} P_h} = k_{i'}^{\prod_{h \neq i',j} P_h}. \tag{3.4}$$

Hence, any other node $v_{i'}$ ($i' \neq j$) can compute k_{-j} using Equation (3.4). However, it is infeasible for v_j to compute k_{-j}.

3.3 Game Theoretic Analysis of INPAC Basic Scheme

In this section, we present a game theoretic model and formally analyze our INPAC basic scheme in this model.

3.3.1 Game Theoretic Model

We model the packet forwarding procedure of a particular session as a repeated game. The players of this game are the nodes that are required by the MORE protocol to forward the packets in this session. We assume that there are n players in total.

The game is divided into stages. In each stage ℓ, each node v_i chooses an action $a_{i,\ell}$, which is a tuple: $a_{i,\ell} = (t_{i,\ell}, \mathsf{PU}_{i,\ell}, \mathsf{PD}_{i,\ell})$. Here, $t_{i,\ell}$ is the number of transmissions v_i chooses to make for forwarding each packet in stage ℓ; $\mathsf{PU}_{i,\ell}$ ($\mathsf{PD}_{i,\ell}$, resp.) is the set of upstream (downstream, resp.) nodes v_i chooses to punish in stage ℓ. The utility of v_i in stage ℓ is

$$u_{i,\ell} = f_{i,\ell} \cdot (p_i(1 - \prod_{\mathsf{dist}(v_h,D)<\mathsf{dist}(v_i,D)}^{v_i \notin \mathsf{PU}_{h,\ell}} \epsilon_{i,h}^{t_{i,\ell}}) - c_i t_{i,\ell}),$$

where $f_{i,\ell}$ is the number of new packets that are received and need to be forwarded by v_i in stage ℓ.

[4]Setting $\epsilon_{i,j}$ to 1 reflects the fact that v_j does not report hearing packets from itself. The recalculation of $t_i^\star$ allows v_i to increase its number of transmissions when v_i finds one or more downstream nodes do not report hearing its packets. Technically, it is crucial to have this step in our protocol so that we can establish a subgame perfect equilibrium. Of course, we note that v_i might be making more transmissions than necessary to deliver packets in this case. However, this additional cost is not high and we get it only if some node deviates from the protocol. When the system converges to the subgame perfect equilibrium, this cost is not incurred. Note that, the per-packet payment p_i is *never* recalculated.

As in the standard theory of repeated games [83], we consider a *discounting* total utility in the entire game. Let $\delta < 1$ be a constant—we call it the discount factor. The total utility of v_i in the entire game is

$$u_{i,\mathbf{total}} = \sum_{\ell=1}^{\infty} \delta^{\ell-1} u_{i,\ell}.$$

Intuitively, this means the player v_i has more interest in the current stage and the near future than in the far future.

3.3.2 Incentive Analysis

In the game theoretic model we have presented, we can obtain the following theorem regarding the incentive compatibility of our INPAC basic scheme.

Theorem 8. *The strategy profile in which all nodes follow the protocol faithfully is a subgame perfect equilibrium in the game.*

Proof. Denote by $s^\star$ the strategy profile in which all nodes follow the protocol faithfully. Consider an arbitrary history H of length L. Suppose that $H = H_1 H_2 \dots H_L$. When the strategy profile $s^\star$ is used, after history H, each node v_i makes $t_i^\star$ transmissions and punishes upstream nodes in the set PU_i and downstream nodes in the set PD_i, i.e., $s_i^\star(H) = (t_i^\star, \mathsf{PU}_i^\star, \mathsf{PD}_i^\star)$. Given our INPAC basic scheme, $\mathsf{PU}_i^\star$ and $\mathsf{PD}_i^\star$ are decided as follows:

$$\mathsf{PU}_i^\star = \{v_j | \exists \ell, 1 \le \ell \le L \wedge H_\ell = (a_1, a_2, \dots, a_j, \dots, a_n) \\ \wedge a_j = (t_j, \mathsf{PU}_j, \mathsf{PD}_j) \wedge v_i \in \mathsf{PD}_j\};$$

$$\mathsf{PD}_i^\star = \{v_j | \exists \ell, 1 \le \ell \le L \wedge H_\ell = (a_1, a_2, \dots, a_j, \dots, a_n) \\ \wedge a_j = (t_j, \mathsf{PU}_j, \mathsf{PD}_j) \wedge v_i \in \mathsf{PU}_j\}.$$

Now consider an arbitrary v_i. Let $s^{\triangle}$ be an arbitrary strategy profile that differs from $s^{\star}$ only in v_i's action immediately following history H. Suppose that $s_i^{\triangle}(H) = (t_i^{\triangle}, \mathsf{PU}_i^{\triangle}, \mathsf{PD}_i^{\triangle})$. If we can show that $u_i|_H(s^*) \geq u_i|_H(s^{\triangle})$ always holds, by One-Deviation Theorem [83], we get that $s^{\star}$ is a subgame perfect equilibrium.

of steps.

First, we can calculate the utilities as follows:

$$u_i|_H(s^{\star}) = \sum_{\ell \geq L+1} \delta^{\ell-1} f_{i,\ell}^{\star}(p_i(1 - \prod_{\mathsf{dist}(v_h,D)<\mathsf{dist}(v_i,D)}^{v_i \notin \mathsf{PU}_{h,\ell}^{\star}} \epsilon_{i,h}^{t_{i,\ell}^{\star}}) - c_i t_{i,\ell}^{\star}), \tag{3.5}$$

$$u_i|_H(s^{\triangle}) = \sum_{\ell \geq L+1} \delta^{\ell-1} f_{i,\ell}^{\triangle}(p_i(1 - \prod_{\mathsf{dist}(v_h,D)<\mathsf{dist}(v_i,D)}^{v_i \notin \mathsf{PU}_{h,\ell}^{\triangle}} \epsilon_{i,h}^{t_{i,\ell}^{\triangle}}) - c_i t_{i,\ell}^{\triangle}), \tag{3.6}$$

where $f_{i,\ell}^{\star}$, $t_{i,\ell}^{\star}$, and $\mathsf{PU}_{h,\ell}^{\star}$ are the number of packets needed to be forwarded by node v_i, number of transmissions made by v_i for each packet needed to be forwarded, and the set of upstream nodes punished by node v_h, respectively, in stage ℓ when the strategy profile $s^{\star}$ is used; correspondingly, $f_{i,\ell}^{\triangle}$, $t_{i,\ell}^{\triangle}$, and $\mathsf{PU}_{h,\ell}^{\triangle}$ are the counterparts when the strategy profile $s^{\triangle}$ is used.

Second, for all $\ell > L$, all upstream node v_h of v_i, clearly we have that

$$\mathsf{PD}_{h,\ell}^{\star} = \mathsf{PD}_{h,L}^{\star} \cup \bigcup_{\mathsf{dist}(v_{h'},D)<\mathsf{dist}(v_h,D)} \mathsf{PU}_{h',L}^{\star},$$

and that

$$\mathsf{PD}_{h,\ell}^{\triangle} \supseteq \mathsf{PD}_{h,L}^{\triangle} \cup \bigcup_{\mathsf{dist}(v_{h'},D)<\mathsf{dist}(v_h,D)} \mathsf{PU}_{h',L}^{\triangle}.$$

Since $\mathsf{PD}^{\star}_{h,L} = \mathsf{PD}^{\triangle}_{h,L}$ and $\mathsf{PU}^{\star}_{h',L} = \mathsf{PU}^{\triangle}_{h',L}$, the above two equations imply that

$$\mathsf{PD}^{\star}_{h,\ell} \subseteq \mathsf{PD}^{\triangle}_{h,\ell}.$$

Since v_i's number of received packets is determined by its upstream nodes' sets of punished downstream nodes, we have that, for all $\ell > L$,

$$f^{\star}_{i,\ell} \geq f^{\triangle}_{i,\ell}. \tag{3.7}$$

Third, we observe that, for all $h \neq i$,

$$\begin{aligned} & \mathsf{PU}^{\star}_{h,L+1} \\ = \; & \{v_j | \exists \ell, 1 \leq \ell \leq L \wedge H_\ell = (a_1, a_2, \ldots, a_j, \ldots, a_n) \\ & \wedge a_j = (t_j, \mathsf{PU}_j, \mathsf{PD}_j) \wedge v_h \in \mathsf{PD}_j\} \\ = \; & \mathsf{PU}^{\triangle}_{h,L+1}. \end{aligned}$$

Hence,

$$\begin{aligned} & p_i(1 - \prod_{\mathsf{dist}(v_h,D)<\mathsf{dist}(v_i,D)}^{v_i \notin \mathsf{PU}^{\triangle}_{h,L+1}} \epsilon_{i,h}^{t^{\triangle}_{i,L+1}}) - c_i t^{\triangle}_{i,L+1} \\ = \; & p_i(1 - \prod_{\mathsf{dist}(v_h,D)<\mathsf{dist}(v_i,D)}^{v_i \notin \mathsf{PU}^{\star}_{h,L+1}} \epsilon_{i,h}^{t^{\triangle}_{i,L+1}}) - c_i t^{\triangle}_{i,L+1}. \end{aligned} \tag{3.8}$$

For all $\ell > L + 1$, since

$$\mathsf{PU}^{\triangle}_{h,\ell} \supseteq \mathsf{PU}^{\triangle}_{h,L+1} = \mathsf{PU}^{\star}_{h,L+1} = \mathsf{PU}^{\star}_{h,\ell},$$

we have that,

$$\begin{aligned} & p_i(1 - \prod_{\mathsf{dist}(v_h,D)<\mathsf{dist}(v_i,D)}^{v_i \notin \mathsf{PU}^{\triangle}_{h,\ell}} \epsilon_{i,h}^{t^{\triangle}_{i,\ell}}) - c_i t^{\triangle}_{i,\ell} \\ \leq\ & p_i(1 - \prod_{\mathsf{dist}(v_h,D)<\mathsf{dist}(v_i,D)}^{v_i \notin \mathsf{PU}^{\star}_{h,\ell}} \epsilon_{i,h}^{t^{\triangle}_{i,\ell}}) - c_i t^{\triangle}_{i,\ell}. \end{aligned} \tag{3.9}$$

Now, we define a function of a single variable $t^{\triangle}_{i,\ell}$ (for an arbitrary $\ell > L$):

$$g(t^{\triangle}_{i,\ell}) = p_i(1 - \prod_{\mathsf{dist}(v_h,D)<\mathsf{dist}(v_i,D)}^{v_i \notin \mathsf{PU}^{\star}_{h,\ell}} \epsilon_{i,h}^{t^{\triangle}_{i,\ell}}) - c_i t^{\triangle}_{i,\ell}. \tag{3.10}$$

From (3.10), we can easily obtain that

$$\begin{aligned} & \frac{\mathsf{dg}(\mathsf{t}^{\triangle}_{\mathsf{i},\ell})}{\mathsf{dt}^{\triangle}_{\mathsf{i},\ell}} \\ =& p_i \prod_{\mathsf{dist}(v_h,D)<\mathsf{dist}(v_i,D)}^{v_i \notin \mathsf{PU}^{\star}_{h,\ell}} \epsilon_{i,h}^{t^{\triangle}_{i,\ell}} \ln \frac{1}{\prod_{\mathsf{dist}(v_h,D)<\mathsf{dist}(v_i,D)}^{v_i \notin \mathsf{PU}^{\star}_{h,\ell}} \epsilon_{i,h}} - c_i. \end{aligned}$$

Hence, $\frac{\mathsf{dg}(\mathsf{t}^{\triangle}_{\mathsf{i},\ell})}{\mathsf{dt}^{\triangle}_{\mathsf{i},\ell}} = 0$ if

$$t^{\triangle}_{i,\ell} = \log_{\prod_{\mathsf{dist}(v_h,D)<\mathsf{dist}(v_i,D)}^{v_i \notin \mathsf{PU}^{\star}_{h,\ell}} \epsilon_{i,h}} \frac{c_i}{p_i \ln \frac{1}{\prod_{\mathsf{dist}(v_h,D)<\mathsf{dist}(v_i,D)}^{v_i \notin \mathsf{PU}^{\star}_{h,\ell}} \epsilon_{i,h}}}.$$

Plugging the payment formula into the above, we get that $\frac{\mathsf{dg}(\mathsf{t}^{\triangle}_{\mathsf{i},\ell})}{\mathsf{dt}^{\triangle}_{\mathsf{i},\ell}} = 0$ if $t^{\triangle}_{i,\ell} = t^{\star}_{i,\ell}$.

Furthermore, from (3.10), we see that $\frac{\mathsf{dg}(\mathsf{t}^{\triangle}_{\mathsf{i},\ell})}{\mathsf{dt}^{\triangle}_{\mathsf{i},\ell}}$ always decreases. So, we have that $\frac{\mathsf{dg}(\mathsf{t}^{\triangle}_{\mathsf{i},\ell})}{\mathsf{dt}^{\triangle}_{\mathsf{i},\ell}} > 0$ if $t^{\triangle}_{i,\ell} < t^{\star}_{i,\ell}$, and that $\frac{\mathsf{dg}(\mathsf{t}^{\triangle}_{\mathsf{i},\ell})}{\mathsf{dt}^{\triangle}_{\mathsf{i},\ell}} < 0$ if $t^{\triangle}_{i,\ell} > t^{\star}_{i,\ell}$. Therefore, for all $t^{\triangle}_{i,\ell}$,

$$g(t^{\triangle}_{i,\ell}) \leq g(t^{\star}_{i,\ell}). \tag{3.11}$$

Combining (3.8)(3.9)(3.10)(3.11), we get that, for all $\ell > L$,

$$p_i(1 - \prod_{\mathsf{dist}(v_h,D)<\mathsf{dist}(v_i,D)}^{v_i \notin \mathsf{PU}^{\triangle}_{h,\ell}} \epsilon_{i,h}^{t^{\triangle}_{i,\ell}}) - c_i t^{\triangle}_{i,\ell}$$
$$\leq p_i(1 - \prod_{\mathsf{dist}(v_h,D)<\mathsf{dist}(v_i,D)}^{v_i \notin \mathsf{PU}^{\star}_{h,\ell}} \epsilon_{i,h}^{t^{\star}_{i,\ell}}) - c_i t^{\star}_{i,\ell}. \quad (3.12)$$

From (3.5)(3.6)(3.7)(3.12), we get that

$$u_i|_H(s^\star) \geq u_i|_H(s^\triangle). \quad (3.13)$$

By equation (3.13), we know that $s^\star$ is a subgame perfect equilibrium. □

Theorem 8 tells us that there is a subgame perfect equilibrium in which all nodes follow the protocol. Clearly, in this subgame perfect equilibrium, each node makes exactly the number of transmissions required by MORE.

3.4 INPAC Extended Scheme

The INPAC basic scheme, which we have presented and analyzed in the previous sections, requires the CCC to always stay online. In this section, we present the INPAC extended scheme, which does not require the CCC to stay online. This extended scheme also has reduced computation and communication overheads compared with the basic scheme.

3.4.1 Main Ideas of Extended Scheme

Before we present our INPAC extended scheme in details, we intuitively explain the main ideas we use in our design of this extended scheme.

Using Offline CCC We no longer require nodes to clear transactions periodically when they are using the wireless mesh network; in stead, we allow nodes to use the

wireless mesh network first, and clear the transactions only when they have high speed connections to the CCC. In this way, the mesh network operator only needs to set up an Internet server as the CCC in order to satisfy our new requirement—this is a much easier task for the operator than maintaining an online authority.

Specifically, we let each node v_i periodically sign its records about each upstream node v_j's transmissions and submit the signed records to v_j itself, as receipts for transmissions from v_j. These receipts *may* allow v_j to get the corresponding payments when the transactions are cleared. As we have mentioned above, node v_j clears transactions only when it has a high speed connection to the Internet (i.e., to the CCC). At that time, the CCC checks each pair of $(batch\_no, code\_vec)$ in each receipt to see whether the coding vector is linearly independent from all coding vectors with the same batch number for which v_j has received payments. Node v_j receives a payment for this pair of $(batch\_no, code\_vec)$ only if the above condition is satisfied.

As in the basic scheme, each node v_i needs to monitor its downstream nodes for possible underreporting of its transmissions. Node v_i monitors a downstream node v_j by periodically checking the number of receipts it has received from v_j and comparing it with the number expected by itself.

Improving Efficiency It is easy to see that a large portion of the computation and communication overheads of our INPAC scheme comes from the generation, transmission, and processing of receipts. Consequently, to improve the efficiency of INPAC, we use a random sampling approach to significantly reduce the number of receipts that need to generated, transmitted, and processed.

Suppose that we would like to reduce the number of receipts to $\frac{1}{2^m}$ of the original, where m is a positive integer. We use a cryptographic hash function $Hash()$ to help us do the sampling: For each node v_i, let x_i be a secret known by v_i and the CCC only. Whenever v_i receives a packet from its upstream node v_j, v_i needs to generate and submit a receipt for this packet only if the first m bits of $Hash(x_i,$

$ID_{v_j}, batch\_no, code\_vec, SIG_{v_j}$) are all zeros. Since $Hash()$ can be viewed as a *random oracle* [11], each packet satisfies this condition with probability $\frac{1}{2^m}$.

This approach is secure and incentive compatible for the following reasons: (a) Upstream node v_j cannot cheat in this procedure. In particular, v_j cannot selectively transmit the sampled packets because it does not know x_i. (b) Node v_i cannot cheat to increase the number of generated receipts, because v_i cannot forge v_j's signature, which is part of the input to the hash function. (c) Node v_i cannot cheat to decrease the number of generated receipts, because then the cheating will be detected and punished by v_j, in a manner similar to the basic scheme.

3.4.2 INPAC Extended Scheme

Using the ideas we have just discussed, we obtain our INPAC extended scheme as follows.

Nodes' Regular Operations on Packets

Source Node: Same as the basic scheme.

Forwarders: When node v_i hears a packet from an upstream node v_j for which v_i is in the forwarder list, v_i does the follows:

1. A forwarder's regular operations in the basic scheme.

2. v_i checks: (a) whether the coding vector is linearly independent from the previous packets in the same batch sent by v_j and heard by v_i; (b) whether the first m bits of $Hash(x_i, ID_{v_j}, batch\_no, code\_vec, SIG_{v_j})$ are all 0. If so, v_i makes a record $(ID_{v_j}, batch\_no, code\_vec, SIG_{v_j})$.

Destination Node: Same as the basic scheme.

Nodes' Periodic Operations

1. *Receipt Submission*: Each node v_i periodically signs its records about each upstream node v_j's transmissions and submits the signed records to v_j itself.When

receiving the receipts, v_j verifies v_i's signatures and also verifies that all pairs of $(batch\_no, code\_vec)$ have indeed been transmitted by itself with v_i being a downstream node. Then, v_j keeps the receipts.

2. *Monitoring Downstream Nodes:* Each node v_i periodically counts, for each downstream node v_j, the number of packets sent by itself for which v_j have submitted receipts. Node v_i compares this number with $\frac{1}{2^m}$ of the total number of packets that v_j should have received from v_i. If v_j reports fewer packets than expected, then v_i punishes v_j using the same method as in the basic scheme.

3. *Monitoring Upstream Nodes:* Same as the basic incentive scheme.

Transaction Clearance

When a node v_i has a high speed connection to the CCC, v_i submits all the receipts it keeps. The CCC clears the transactions in a way similar to the basic scheme.

3.5 Possible Attacks and Defenses

When our INPAC (basic or extended) scheme is used, security attacks may be launched by selfish or malicious nodes. Although the focus of this chapter is incentives rather than security, for practical purposes, we still briefly consider two possible attacks and discuss the defenses against them.

3.5.1 Extra Signature Attack

A selfish forwarder node v_i may launch an attack on our protocol by putting some extra signed pairs of $(batch\_no, code\_vec)$ in the payload of a packet. For example, suppose v_i is going to send a packet that has batch number 001 and coding vector $(1, 1, 1)$. So, the pair $(001, (1, 1, 1))$ is in the MORE header of this packet, and the signature on this pair is in the INPAC header. Node v_i launches an attack by putting signed pairs $(001, (1, 2, 3))$ and $(002, (2, 1, 4))$ in the payload of this packet, in hope

that the latter two signed pairs will also bring some payments to itself. Note that this attack will *not* work if all nodes hearing this packet follow the protocol, because nodes following the protocol should not take these signed pairs of $(batch\_no, code\_vec)$ (i.e, signed $(001, (1, 2, 3))$ and $(002, (2, 1, 4))$) from the payload and make records for them—they should make only one record for $(001, (1, 1, 1))$ from the MORE header and the corresponding signature from the INPAC header. Nevertheless, if a downstream node v_j hearing this packet does not follow the protocol, v_j may make records for these extra signed pairs and then submit the records to the CCC. In this case, v_i may get undeserved payments with the help of v_j.

We argue that the above attack is actually a colluding attack, because in this case v_i and v_j must have a prior agreement on the format of packet payload. One possible defense is that every node randomly samples a portion of packets it hears, to detect signed pairs of $(batch\_no, code\_vec)$ in the payload. If the attack is detected, it is reported to the network operator, who excludes the attacker nodes from the system and pursues liability against the owners of these nodes.

A better solution to this problem can be provided by a collusion-resistant incentive scheme. However, since collusion resistance is technically highly challenging, we leave this topic to future study.

3.5.2 Corrupted Data Attack

A forwarder node v_i may also launch an attack on our protocol by tampering with the payloads of data packets. When v_i receives a packet that it should forward, v_i can modify or remove part or all of the bits in the payload of this packet. This attack allows v_i to gets payments for forwarding packets while the destination does not receive the correct data in these packets.

We propose a simple defense against this attack: On one hand, when a node forwards a packet, it signs $(batch\_no, code\_vec, payload)$ rather than just $(batch\_no, code\_vec)$.

Since this signature will be part of the receipt, the packet receipt can serve as an evidence of cheating if this node corrupts the payload. On the other hand, the source node S and the destination D should establish a secret key known to only S and D and protect the entire batch of data using a message authentication code (MAC). If a corrupted data attack is launched, the destination will detect the attack using the MAC. Then, the destination requests all forwarders to submit all their receipts to the source so that the source can determine who has corrupted the data. (This defense works for the basic scheme. If it is used for the extended scheme, then it finds the attacker node with a probability, which may deter the attacker.)

It is worth noting that this corrupted data attack is actually an *independent* security problem for network coding that exists *even if no incentive scheme is used*. It has been studied in, e.g., [57, 117]. Hence, we can also adapt existing solutions to our settings in order to defend against this attack.

3.6 Evaluations

We completely implement INPAC and evaluate it on the Orbit wireless testbed [92]. Specifically, we carry out three sets of experiments:

- The first set of experiments are on the utilities of cheating nodes. The results show that a node cannot increase its own utility if it cheats in making transmissions, in reporting heard transmissions, or in punishing downstream nodes.

- The second set of experiments show that, starting from a network system where selfish wireless nodes have random (not necessarily cooperative) behaviors, INPAC makes the system quickly converge to a stable state. In this stable state, every node maximizes its own utility by faithfully following the protocol.

- The third set of experiments are on the computation and communication overheads. Our results show that INPAC is quite efficient.

Below we first describe the setups of all our experiments, and then present the detailed results for each set of experiments, respectively. Note that, since the basic scheme and the extended scheme are equivalent in terms of nodes' utilities and the system's convergence (except that the speeds of utility changes and system convergence depend on different parameters), *for the first two sets of experiments, we only present our results on the basic scheme*; the results on the extended scheme are similar. For the third set of experiments, we present our results for both the basic scheme and the extended scheme.

3.6.1 Setups of Experiments

From the Oribit radio grid testbed, we randomly select 30 nodes in the 20×20 grid. Fig. 3.3 shows the locations and IDs of these nodes. Each node has a 1-GHz VIA C3 processor with 512 MB of RAM and a 20 GB local hard disk, and is equipped with Atheros AR5002X Mini PCI 802.11a/b/g wireless card attached to an omni-directional antenna. Nodes are set to transmit at a power level of 20dbm and operate in the 802.11b mode with the bit rate 11Mbps. Softwares on each node include Linux Debian kernel v2.6.22, Mad-Wifi v0.9.3.3 [73], Click v1.6.0 [98], the MORE package [21], the Cryptopp Library 5.5.2 [35].

Before running the experiments, we use a module in MORE to measure the loss rate of each link, and find that the link loss rates vary between 17.07% and 100%. In MORE, we set the batch size to 32 packets, and the size of each packet is 1500 bytes. The minimum load threshold is set to 0.2 for MORE pruning module.

In the experiments on the basic scheme, we place the CCC on node 2 at location $(2, 2)$. Unless stated differently, nodes submit their reports to the CCC every 30 seconds. Each node checks, every 1 minute, whether any downstream node underreports its transmissions. In payment calculations, the cost of making each transmission is 1.

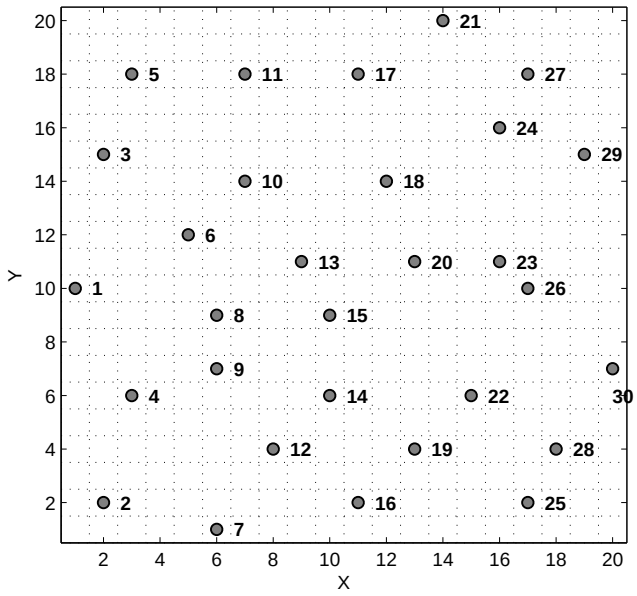


Figure 3.3. Testbed Topology.

3.6.2 Nodes' Utilities and Cheating Behaviors

When a node deviates from the protocol in packet forwarding, it can have three types of basic cheating behaviors: changing the number of transmissions, underreporting upstream nodes' transmissions, and improperly punishing downstream nodes.[5] In this set of experiments, we study the nodes' utilities for each type of basic cheating behavior, respectively, and for a mixture of basic cheating behaviors.

We have 100 runs of each experiment described below. In each run we randomly choose two nodes as the source and the destination, to have a session of 120 seconds, in which the source node is always backlogged. Unless stated differently, in each run, we randomly select an involved node and compare the utilities it receives when it cheats and when it follows the protocol.

[5]Actually a node may also cheat by overreporting upstream nodes' transmissions, but we ignore this possibility in our experiments because it is easily detected by the CCC through signature verifications.

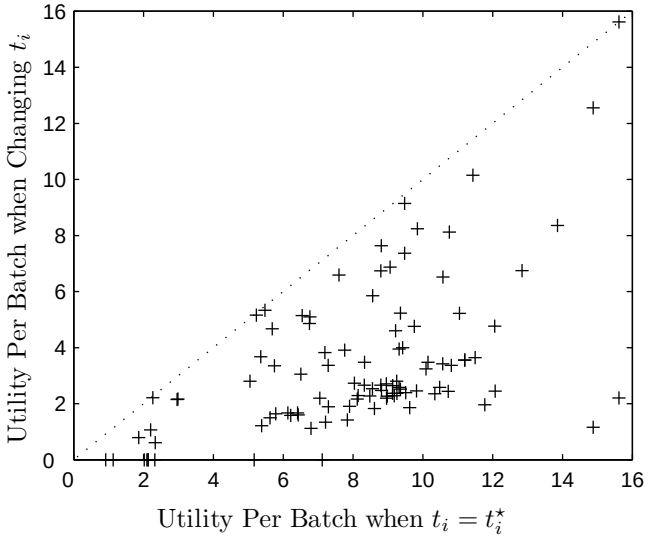


Figure 3.4. Scatter plot of nodes' utilities, changing t_i vs. following the protocol. Each point represents a node. Points below the 45 degree line $y = x$ indicate that changing t_i yields lower utilities than following the protocol.

Changing Number of Transmissions We measure the per-batch utilities of nodes when they cheat by changing their t_i, the number of transmissions for forwarding each packet. When a node cheats, its t_i is randomly chosen between 0 and $2.0t_i^\star$, where $t_i^\star$ is the number of transmissions it should make when it follows the protocol.

Fig. 3.4 shows the scatter plot for utility comparison in the 100 runs. Each point represents the utilities of a randomly selected node in one run: The y-coordinate of the point is the node's utility when it changes its t_i and the x-coordinate is the same node's utility when it follows the protocol. We can see that all points are below the 45 degree line $y = x$, which means cheating always reduces a node's utility.

We further investigate the relationship between values of t_i and the received utility. In particular we observe the utilities of 4 selected nodes in one flow from node 4 to node 27. Each time we let one selected node change its t_i by $\pm 0.5t_i^\star$ or $\pm 0.8t_i^\star$, respectively, and other nodes remain cooperative. Fig. 3.5 shows the utilities per

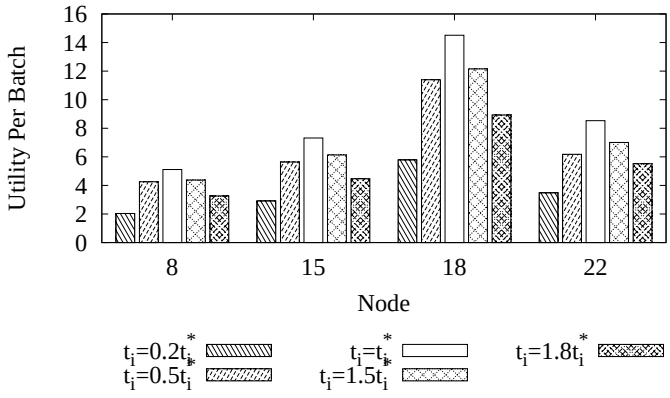


Figure 3.5. Per batch utilities of four nodes when each of them uses different values of t_i. For each node, the utility is maximized when $t_i = t_i^\star$.

batch for different values of t_i. Clearly, for each node, the maximum utility per batch is achieved when $t_i = t_i^\star$, i.e., when the node follows the protocol.

Underreporting Transmissions of Upstream Nodes Now we consider the cheating behavior of underreporting transmissions of upstream nodes.

Fig. 3.4 is the scatter plot for utility comparison in 100 runs. The y-coordinate of each point is the node's utility when it underreports the transmissions of its upstream nodes and the x-coordinate is the same node's utility when it reports correctly. Again all points are below the line $y = x$, meaning that underreporting always reduces a node's utility. To be more precise, in the 100 runs, the behavior of underreporting reduces the utility of cheating node by 4.65% to 44.03% (with median case 31.49%) compared with the utility when reporting correctly.

Similarly, in all the 100 runs, improperly punishing downstream nodes always leads to utility losses, ranging from 10.87% to 129.65%. Hence, neither type of cheating behavior can benefit the cheating node in any case.

Under-punishing or Over-punishing Downstream Nodes

Now we focus on the utility of the cheating node when it improperly punishes the downstream nodes. In particular, we consider the two cases of unde-punishing and

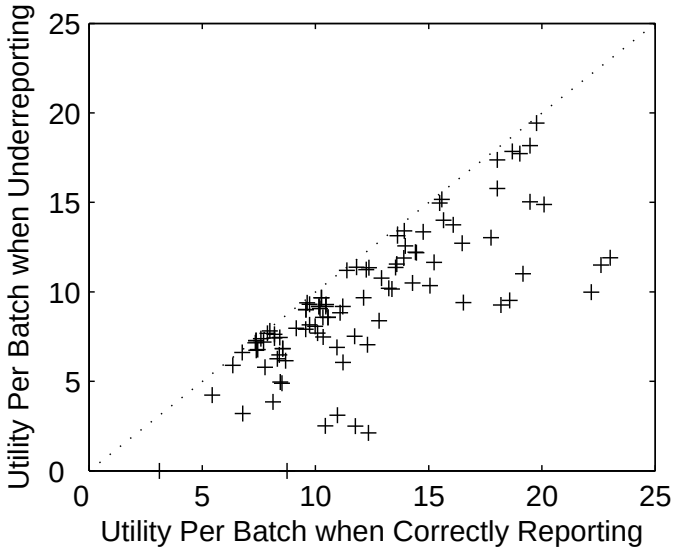


Figure 3.6. Scatter plot of nodes' utilities, under-reporting the transmissions of upstream nodes vs. correctly reporting. Each point represents a node. Points below the 45 degree line $y = x$ indicate that underreporting yields lower utilities than correctly reporting.

over-punishing respectively. In Fig. 3.7, we measure the utility gain for the cheating behavior of under-punishing downstream nodes, where the *utility gain* is defined as $\frac{\text{utility when cheating}}{\text{utility when following the protocol}} - 1$. The results show that under-punishing the downstream nodes can never increase the utility for the cheating node. In fact, the cheating node's utility decreases by 3.34% to 69.15% compared with following the protocol.

Fig. 3.8 shows the utility gain for over-punishing downstream nodes. The definition of utility gain is the same as described above. As we can see, over-punishing downstream nodes always leads to utility losses, ranging from 10.87% to 129.65%.

Mixed Cheating Behaviors It is definitely possible that nodes use more than one cheating behaviors mentioned above at the same time. In this experiment, we allow the selected node to have more than one basic cheating behaviors simultaneously. These cheating behaviors are chosen at random in each run of our experiment. In Fig. 3.9, we illustrate the utility gains of the selected node when it has a mixture of basic cheating behaviors. Clearly, mixing up the cheating behaviors cannot provide

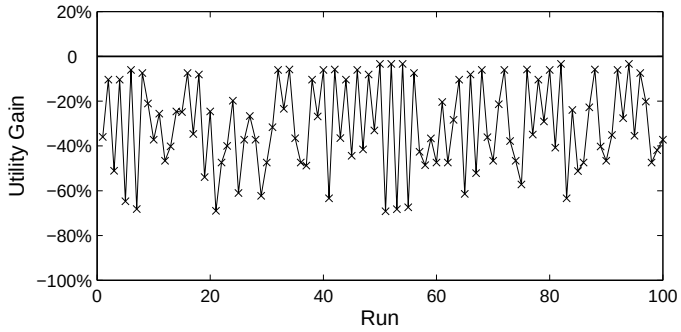


Figure 3.7. Utility gains for the cheating behavior of under-punishing downstream nodes. All utility gains are negative.

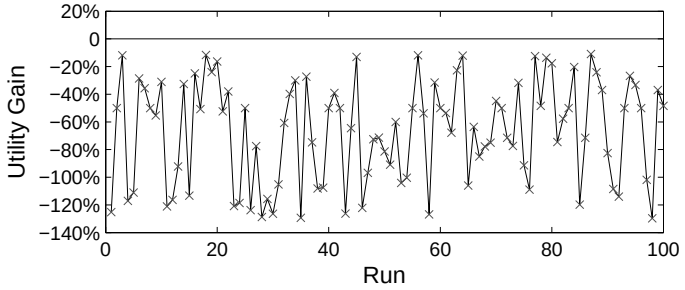


Figure 3.8. Utility gains for the cheating behavior of over-punishing upstream nodes. All utility gains are negative.

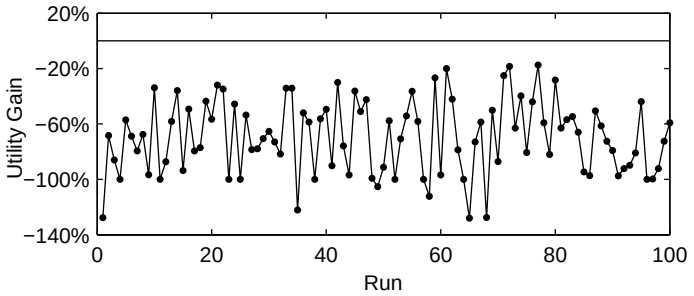


Figure 3.9. Utility gains for a mixture of basic cheating behaviors. All utility gains are negative.

any benefit to the cheating node. In fact, the utility losses range from 17.46% to 135.93%.

3.6.3 System Convergence

When INPAC is used, the wireless network has a stable state, namely the equilibrium state, in which all nodes faithfully follow the protocol. In this set of experiments, we study the procedure that the network system converges to the stable state.

At the beginning of each experiment, we let each node randomly select one of the following three behaviors: following the protocol, cheating by making only $0.5 * t_i^\star$ transmissions for forwarding each packet, and cheating by underreporting transmissions from an upstream node. After the experiment begins, each node repeatedly changes its behavior randomly. If the new behavior increases its own utility, the node moves to the new behavior; otherwise, it returns to its old behavior. The node terminates this procedure when it finds no way to further increase its own utility. When all nodes stop changing their behaviors, the entire network system is in a stable state.

We randomly pick 4 nodes in one experiment and observe their utilities in Fig. 3.10. (Due to space limitation, we cannot show the results of other experiments, which are similar.) As Fig. 3.10 shows, it takes about 10 minutes for the system to converge to a stable state, in which all nodes faithfully follow the protocol. Given the

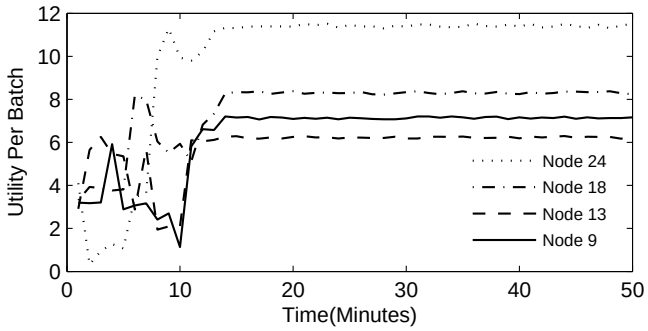


Figure 3.10. Nodes' utilities when the network system converges to its stable state.

setup of our experiment, the convergence is fairly fast. We can make it even faster if the transactions are cleared more frequently.

3.6.4 Overheads

Now we measure the overheads of INPAC in two different situations: when the system is in the stable state, and before the system converges to the stable state. Our experiments cover both the basic scheme and the extended scheme.

Overheads in Stable State We measure the overheads of both the INPAC basic scheme and the INPAC extended scheme, each in a session of transmitting 4800 packets, when the network system is in a stable state. We use RSA digital signature schemes with a modulus of 1024 bits. In the extended scheme, we set $m = 6$. The results of our measurements are shown in Table 3.1. We can see that the overheads of both the basic scheme and the extended scheme are reasonably low. If we compare the basic scheme with the extended scheme for the total overheads in the entire session, then the extended scheme is about 38.74% more efficient than the basic scheme, because the extended scheme has fewer operations of making reports.

Overheads in Convergence Before the system converges to a stable state, there may be additional overheads for punishing downstream nodes, which include the time for packet encryptions and decryptions. We use the 128-bit AES in ECB mode. On

Table 3.1. Overheads of INPAC.

Average time for processing a packet in either scheme	1.45 ms
Average time for making a report in either scheme	0.78 ms
Basic scheme's total overheads in entire session	4.75 s
Extended scheme's Total overheads in entire session	2.91 s

average, the overhead for punishing a downstream node is 0.143 ms per packet. We note that the keys for punishments need to be set up in advance. In a network of size 30, the key setup time is 10.09 ms per node.

3.7 Summary

In this Chapter, INPAC is proposed, the first incentive scheme for packet forwarding in wireless mesh networks using network coding. It is complementary to the existing work on incentive compatible routing in the same type of wireless networks. Since packet forwarding is a fundamental procedure for computer networks, INPAC is of great importance to the application of network coding technology in environments with selfish users. Extensive evaluations for INPAC on the Orbit Lab testbed have been performed, and the results demonstrate that INPAC is both efficient and incentive compatible.

Since this scheme is designed to provide incentives to each individual node, one interesting open problem is to design a collusion resistant incentive scheme. It is expected that this open problem to be very challenging and I plan to study it in the future work.

CHAPTER 4

MESSAGE FORWARDING COOPERATION IN VEHICULAR AD HOC NETWORKS[29]

4.1 Background and Motivation

Vehicular ad hoc networks support communications among smart vehicles, and between vehicles and nearby roadside equipment. There can be numerous useful and interesting services on the road provided by VANETs [89, 112, 111, 72, 84, 17] in the near future. In VANETs, effective and efficient message delivery among vehicles must be guaranteed. Under some circumstances, (e.g., night-time with low vehicular density, or disseminating commercial ads through VANETs), to overcome the difficulty of intermittent connectivity, store-carry-and-forward message switching becomes an important idea of routing in VANETs. A node stores and carries messages; it considers forwarding a message to another node whenever these two nodes come into the communication range of each other. In this way, each message is forwarded from one node to another. A number of routing protocols (e.g. [61, 58, 99, 18, 95]) have been proposed to increase the likelihood of successfully delivering a message, which can be applied to VANETs.

However, even if we have a good routing protocol for a VANET, it is still a crucial question whether nodes will *follow the protocol or not*. The necessity of solving this problem can been observed in the perspectives of two types of nodes. On the one hand, an ordinary node of the VANET may belong to an individual user and thus be *selfish*. It may be unwilling to forward messages of others for nothing, and moreover carrying message takes its own storage space. On the other hand, in many routing

protocols, nodes with special abilities, (e.g. those with more active mobility on the road, like Taxi cars), are more likely to be picked as forwarders. For these nodes, the situation is worse: even though they are willing to forward messages initially, the overwhelming load of services for others will soon consume so much of their communication resource (e.g., wireless bandwidth and storage space) that they have to deviate from the protocol to save their own resource. Therefore, it is highly important to give nodes incentives, stimulating them to cooperate in forwarding messages.

Indeed, automotive industry controls the vehicle manufacture. However, we can also foresee some problems of cooperation even if the manufacturers do not leave it as an option for the users to choose being cooperative or not. Actually, after the vehicles are sold, they are under the full control of the users. Thus, although the manufacturer does not leave an option for the users to choose being cooperative or selfish, the users can still get help from some expert hackers in changing the VANETs protocols running in the vehicles, so that they can be 'free riders' in the network without contributing anything. Hence, we believe that mandatory cooperation in VANETs is difficult to achieve and designing incentive-compatible packet forwarding protocols can help providing a feasible way to enforce the mandatory cooperation in VANETs.

There are two types of existing incentive mechanisms for stimulating cooperation in wireless networks: reputation-based approaches (e.g., [76, 78, 88]) and credit-based approaches (e.g., [119, 120]). Reputation-based approaches rely on observing the behavior of neighbor nodes and punishing the detected uncooperative nodes to stimulate cooperation. In VANETs, however, for a distributed reputation system, the deviating behaviors of a selfish node are more difficult to be observed and determined by other nodes, because the connections with the same nodes are occasional. A recent work on incentive aware routing in delay tolerant networks[94] cannot be applied to VANETs for a similar reason: The authors use tit-for-tat mechanism in which nodes

reward or punish their neighbors based on the history they have observed; however, in VANETs, the connection opportunity between any two nodes may only be once and thus the neighborhood relationship cannot easily be established. Of course, if a centralized reputation controller is established in the VANET, it could collect and broadcast the reputation of any node to help stimulate the cooperation in the network. In this chapter, we aim at another approach, credit-based mechanisms, to encourage cooperation by rewarding credits to the cooperative nodes. This idea is especially appropriate for many applications in VANETs, such as disseminating advertisement using vehicles. In existing works for traditional multi-hop networks, the credit-based mechanisms depend on end-to-end connections to determine how many credits each node should receive. In VANETs, since end-to-end paths are not guaranteed at all, existing credit-based mechanisms cannot be used either.

We use an approach based on *coalitional game theory* to solve the *forwarding cooperation problem in VANETs*. In particular, we say a node is cooperative in forwarding in VANETs, if it follows the routing protocol. In a coalitional game, there are a number of players. These players correspond to the nodes in a VANET. When the players in a subset decide to cooperate within the subset, the subset is called a *coalition*. In particular, the coalition of all players is called the *grand coalition*. Hence, our goal is to ensure that, whenever a message needs to be forwarded in a VANET, all involved nodes have incentives to form a grand coalition. In coalitional game theory, there is a strong solution concept, namely *core*, that can provide such guarantees.

We propose an incentive scheme for VANETs and rigorously analyze it in the framework of coalitional games, showing that, when it is used, following the protocol is in the core of the coalitional game. In addition, we extend our scheme to take the limited storage space of each node into consideration. When a node does not have sufficient space for storage, it has to discard some of the messages. To decide

which messages to discard, a lot of routing protocols (e.g., [18, 17, 8]) require some auxiliary information to be transmitted in control messages, such as the probabilities of meeting the destinations. Although in principle we can stimulate the forwarding of these control messages using the same method that we use for data messages, it would require lots of overheads to do so. To make our scheme more efficient, we propose a light-weight approach which makes full use of the selfishness of the autonomous nodes, giving them the freedom to choose which messages to discard. Our extended scheme guarantees that it is to the best interest of each node to discard the messages that the system prefers to drop.

There are a few existing works [66, 67] on the incentive problems of packet forwarding in VANETs. However, they either target a specific routing goal (e.g., [66]), or does not have a rigorous proof for nodes' cooperation (e.g., [67]). In contrast, our work considers the incentives for all nodes including the sources and guarantees the cooperation of them under rigorous theoretical analysis.

The rest of the chapter is organized as follows. In Section 4.2 we introduce basic concepts in coalitional game theory, and present a model of the forwarding cooperation problem in VANETs. Section 4.3 describes the incentive scheme. In Section 4.4, an extension to limited storage space is considered. Experimental evaluation results are presented in Section 4.5. We summarize this chapter in Section 4.6.

4.2 System Model

4.2.1 Coalitional Game Formation in VANETs Message Forwarding

In this subsection, we introduce the VANET system model used in this chapter and present a coalitional game model for message forwarding in VANETs.

We consider a VANET with a set of mobile nodes. Two nodes can exchange messages when they are within the transmission range of each other. Here we consider a general routing protocol, denoted by $\Re$. Note that $\Re$ could be one of the many

existing routing protocols. In this chapter, we assume that there is only one routing protocol in the system. It is a very interesting problem when there are different routing protocols coexisting in the network. If each node knows which routing protocol should be chosen, our incentive scheme can be extended to cope with this situation by adding one more piece of information into the message receipt, indicating the routing protocol used for the message transmission. Otherwise, we will have a new challenging problem; We leave it to future study.

In the VANET, messages can be delivered directly to the destination or forwarded by some intermediate nodes before reaching destination. The intermediate node may or may not replicate a copy of the message and keep it during the transmission, according to different routing protocols. Note that in this chapter we use the term, *forwarding*, in a very general sense; by forwarding a message we mean either the transfer of the message itself or the transfer of its copies to the next hop.

A directed graph $G = (V, E)$ is used to describe the forwarding of each message. V is the set of nodes that are required to participate in routing this message by $\Re$. Each directed edge in E represents that the message is forwarded from the tail node to the head node. In other words, the graph G records the traces of a message and its copies. In some application scenarios, nodes are all equipped with GPS. Then it is possible to modify the V to create geo-referenced coalitional games. In particular, using geo-location information, we can only consider the nodes that meaningful with respect to source and destination, so that the signaling and communication overhead can be reduced. Here in our game, to keep it general, the nodes are those who are required to forward packets by the routing protocol.

We now model the transfer of a message from its source (*src.*) to its destination (*dest.*) as a forwarding coalitional game. The forwarding coalitional game $(\mathbf{N}, v)$ starts when the message is generated by *src.*, and ends after it and its copies disappear in the network, either successfully received by the *dest.* or discarded by all intermediate

nodes in halfway. The players are the nodes in V (i.e. $\mathbf{N} = V$), including *src.* and *dest.*. In the process of this message being transferred from one node to another, two nodes are in a coalition, if the message (or its copy) is transmitted between them in the way defined in $\Re$. We call the forwarding behaviors specified by $\Re$, legal forwarding, for convenience in the rest of this chapter. The coalitional relationship is *transitive*, i.e. if node p and q are in a coalition, and meanwhile q and r are in a coalition, then p and r are in the same coalition.

Recall that to form the forwarding coalitional game $(\mathbf{N}, v)$, $\mathbf{N}$ and v must be specified. Since in VANETs the end-to-end connections are not guaranteed, the first challenge to form the forwarding coalitional game, is to determine the nodes that should be involved the message forwarding according to $\Re$, i.e. players set $\mathbf{N}$. The difficulty lies in the fact that in VANETs routing protocols, the next forwarder can only be determined when the carrier and the potential forwarder actually meet based on some routing information (In this chapter, by *meet* we mean two nodes come in the communication range with each other.). To clarify the $\mathbf{N}$ for each game, we use a stimulating approach to encourage the nodes to report their meetings. Every time two nodes meet, each node keeps a brief record of their meeting and the routing information. For reporting each record, nodes (except *src.*) can obtain an amount of reward, u, for assisting to determine the player set $\mathbf{N}$. We note that the reason of keeping records of neighbors is for enforcing the incentive scheme, not for routing messages in the VANETs. Our incentive scheme can work with routing protocols that do not need the information about neighbors. We will present the specific system design, e.g., where and how the records will be reported and content of the record in detail in Section 4.3.

An important component in a coalitional game is the definition of the payoff (worth) of a coalition S, $v(S)$. Naturally, the total payoff of a coalition should reflect their success in forwarding the message to *dest.*. Let $d(S)$ denote the number of

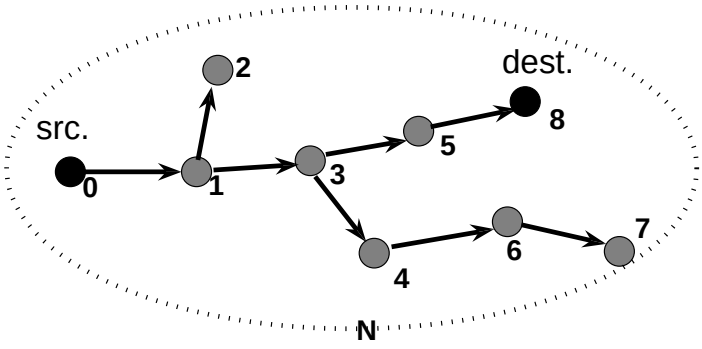
message copies that are successfully delivered to *dest.* within coalition S. We can formulate the worth of coalition S as follows.

$$v(S) = \delta \cdot d(S) + u \cdot n_{rec}(S), \tag{4.1}$$

where δ is a system parameter representing the reward for successfully delivering one message copy,[1] and u is the unit amount of reward for reporting a record. $n_{rec}(S)$ is the number of meeting records submitted by the members in S. In words, the worth of a coalition consists of two parts, rewards for successfully transferring data to *dest.*, and rewards for helping determine the player set $\mathbf{N}$. Clearly, if *dest.* is not in S, then $d(S) = 0$. Moreover, by the transitivity of the coalition, $d(S) > 0$ if and only if both *src.* and *dest.* are in S.

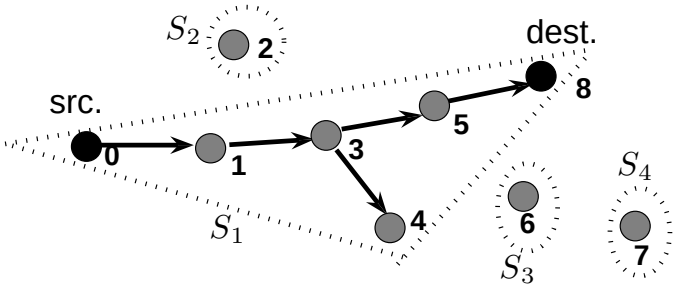
In Figure 4.1, we illustrate the forwarding coalitional game model with two examples. Their description graphs G are in subfigures (a) and (b) respectively. The locations of the nodes in the graph have no physical meaning. The number labeling each node is the node ID. Recall that each edge represents a legal forwarding between the two nodes. In subfigure (a), all players form a grand coalition $\mathbf{N}$, that is, all players involved in the transmission follow the routing protocol $\Re$. In (b), the legal forwarding between nodes 1 and 2 does not happen when the two nodes meet; neither do those between 4 and 6, 6 and 7. As a result, in the forwarding coalitional game, 4 coalitions are formed. In the two games, all nodes report their meeting records, for the rewards. Let $\delta = 10$ and $u = 0.5$. For the grand coalition, $d(N) = 1$, and $n_{rec}(N) = 15$ (because each meeting is reported twice by the nodes excluding the source), then according to Eq. (4.1) $v(N) = 17.5$. In Figure 4.1(b), $d(S_1) = 1$, but

[1]From the source and the destination's point of view, it suffices to have a single copy transferred to the destination, and so it seems unnecessary to reward the transfer of each copy of the message. Nevertheless, since each copy of message is typically transferred by different nodes, if we don't reward the transfer of every copy, the result could be that no copy of the message is transferred.



$v(N) = 17.5$

(a) A grand coalition



$v(S_1) = 15.5$, $v(S_3) = 1$
$v(S_2) = v(S_4) = 0.5$

(b) Four coalitions are formed

Figure 4.1. Illustration of forwarding coalitional game model.

$d(S_2) = d(S_3) = d(S_4) = 0$ because the destination is not included in S_2, S_3, S_4. Similarly, according to Eq. (4.1) we can calculate the worth of each coalition in each game as shown in the figure.

4.3 Incentive Scheme for VANETs Message Forwarding

After establishing the forwarding coalitional game model, in this section we design an incentive scheme for VANETs message forwarding based on this model. First, we present the system architecture and we introduce a payoff allocation method that we will use in the incentive scheme. Then, we rigorously show that it can result in a strongly stable state which is in the core. After that, we present a complete design of our incentive scheme based on our payoff allocation method. Finally, we describe how our scheme deals with cheating.

4.3.1 System Architecture

The overall architecture of the system consists of a number of smart vehicles that have VANET communication devices installed and a central authority, called the virtual credit center (VCC). As in many other incentive schemes for wireless networks and especially VANET (e.g., [119, 120] and [66]), the VCC is used. We assume that the VCC issues a certificate to each node and each node has an account (of virtual currency) in the VCC. Nodes do not need to connect to the VCC all the time. Instead, nodes save and store the information that they need to communicate with the VCC temporarily and when they are close to some infrastructures, they connect to the VCC and communicate with it (including receiving credits). For example, they can connect to the VCC in the gas station.

Initially, each node in the VANET system has an equal amount of virtual currency in its account stored by the VCC. If a node has helped in the forwarding of a message, whenever the node have chance to connect to the VCC, it submits the evidences (i.e.,

records of meetings and message receipts, which will be described in details later) to the VCC and receives the credits from the VCC. The VCC gives credits to a node in the form of virtual currency, i.e., it increases the amount of virtual currency that the node keeps in its account kept by the VCC. Correspondingly, the source node will be charged (the VCC decreases the amount of virtual currency in the source node's account). We note that in some cases, the source of a message may send it as a reply for a request for the benefit of the destination. We consider this problem as the incentive issues in the application layer, e.g., providing data service for others. There are some works on the incentive issues in the application layer (e.g., stimulating cooperation file sharing in peer-to-peer networks). We think that the incentive for the source to send packets and the incentive for the intermediate nodes to forward packets are two separate issues. The source is motivated by the incentive scheme in the application layer to send messages to the destination, while the intermediate nodes should also be incentivized in the network layer. Once the source is motivated by the application layer mechanism, it makes sense to let the source pay the forwarders since only when the data messages are delivered the source can receive rewards from the destination by the incentive scheme in the application layer. In this chapter, we only focus on the incentive issues in the network layer. When a node needs more virtual money, it can buy some using real money. All transactions are cleared within the VCC. The details about how the VCC will process the evidences will be presented in Section 4.3.4.

4.3.2 Allocation of Payoff

Our goal is to design a payoff allocation method ($X \in \mathbf{R^N}$) in the forwarding coalitional game such that for the transmissions of each message in VANETs, the grand coalition is guaranteed. To achieve the grand coalition, the challenge is to make sure the non-emptiness of the core in the game, and to assign a payoff allocation to

each player in the coalition which satisfies the core requirement. Naturally, the source node and intermediate nodes should be treated differently in the payoff allocation, due to their different roles in the game. Therefore, we consider them separately.

4.3.2.1 Payoff Allocation to Intermediate Nodes

For each intermediate node, its share of payoff should reflect its contribution in the game. Hence the payoff allocation function for intermediate nodes, is designed as follows, based on two types of behaviors in the coalition, receiving and forwarding.

$$x_i = \alpha \cdot m_r(i) + \beta \cdot m_f(i) + u \cdot n_{rec}(i), \forall i \neq src. \tag{4.2}$$

In Eq. (4.2) $m_r(i)$ is the number of times that intermediate node i receives one copy of the message from some other node. $m_f(i)$ is the number of times that i successfully forwards one copy of the message to another node following the routing protocol $\Re$. α and β are the rewards for the receiving and forwarding behaviors respectively. $u \cdot n_{rec}(i)$ is the amount of reward to node i for reporting the meeting records. Note that *dest.* can be viewed as an intermediate node, which only receives copies without further forwarding.

4.3.2.2 Payoff Allocation to The Source Node

The payoff allocation to the source node contains two parts: the gains by successfully delivering the message copies to *dest.*, subtracted by rewards used to pay the intermediate nodes. The payoff allocation function for *src.* is defined in Eq. (4.3).

$$x_{src} = \delta \cdot d(\mathbf{N}) - (\alpha \sum_{i \in N - \{src\}} m_r(i) + \beta \sum_{i \in N - \{src\}} m_f(i)). \tag{4.3}$$

4.3.3 Sufficient Conditions to Achieve Core

With the payoff allocation functions described above, will the forwarding coalitional game automatically achieve a stable grand coalition? Actually it depends on

the parameters δ, α and β. If the values of δ, α and β are chosen inappropriately, the core of the game may become empty. So in the sequel, we study how to choose the parameters and ensure that the payoff allocation of our incentive scheme is in the core, i.e., the payoff allocation satisfies individual rationality, coalitional rationality and efficiency respectively. At the end of this section, we summarize the results and give the sufficient conditions on δ, α and β for achieving the core.

4.3.3.1 Individual Rationality

First we examine the individual rationality of the players, i.e. no player receives less than what it could get on its own. For the source node, if it does not send the message to any intermediate node, then $v(\{src.\}) = 0$. Therefore, it is necessary to make sure that $x_{src.} \geq 0$ in grand coalition $\mathbf{N}$ whenever $d(\mathbf{N}) > 0$ to guarantee the individual rationality for the source node. [2].

Before introducing the parameter conditions for *src.*'s individual rationality, we define two terms m_r and m_f. Denote m_r, i.e. $m_r = \sum_{i \in \mathbf{N} - \{src.\}} m_r(i)$, the total number of receiving behaviors of intermediate nodes in grand coalition. Similarly, we let $m_f = \sum_{i \in \mathbf{N} - \{src.\}} m_f(i)$.

The following lemma specifies the condition to achieve individual rationality.

Lemma 9. *If the equation Eq. (4.4) holds for the payoff allocation defined in (4.2) and (4.3), then the individual rationality is guaranteed.*

$$max(\alpha, \beta) \leq \frac{\delta \cdot d(\boldsymbol{N})}{m_r + m_f}, \quad whenever\ d(\boldsymbol{N}) > 0 \tag{4.4}$$

[2] If $d(\mathbf{N}) = 0$, it means that although all involved nodes follow the routing protocol, *dest.* still does not receive any copy of the message. In this case, *src.* will get negative payoff allocation according to (4.3). But we argue that it is reasonable for *src.*, if it wants to transmit the message. Moreover, it is necessary to have this negative payoff allocation in order to prevent the cheating of *src.* and *dest.* in a collusion (see Section 4.3.5)

Proof. For each intermediate node i, if it does not join the coalition, which means it does not record or forward any copy of the message, then $m_r(i) = 0$ and $m_f(i) = 0$. Hence $v(\{i\}) = 0$. Since in the definition of x_i (Eq. (2)) all components are non-negative, we have that $x_i \geq v(\{i\})$.

For the source node, it is easy to see that, if Eq. (4.4) holds, then

$$\begin{aligned} x_{src.} &= \delta \cdot d(\mathbf{N}) - (\alpha \cdot \sum_{i \in \mathbf{N} - \{src.\}} m_r(i) \\ &\quad + \beta \cdot \sum_{i \in \mathbf{N} - \{src.\}} m_f(i)) \\ &\geq \delta \cdot d(\mathbf{N}) - max(\alpha, \beta)(\sum_{i \in \mathbf{N} - \{src.\}} m_r(i) \\ &\quad + \sum_{i \in \mathbf{N} - \{src.\}} m_f(i)) \\ &= \delta \cdot d(\mathbf{N}) - max(\alpha, \beta) \cdot (m_r + m_f) \\ &\geq 0 \end{aligned}$$

Since $v(\{src\}) = 0$, the individual rationality for $src.$ is also guaranteed if $max(\alpha, \beta) \leq \frac{\delta \cdot d(\mathbf{N})}{m_r + m_f}$, whenever $d(\mathbf{N}) > 0$. □

Given the result of Lemma 9, in designing our incentive scheme, we make $max(\alpha, \beta) = \frac{\delta \cdot d(\mathbf{N})}{m_r + m_f} - \varsigma$, where ς is a constant small number. Since we also need to guarantee that $\alpha > 0$ and $\beta > 0$, we choose ς such that $\varsigma < 1/(m_r + m_f)$.

4.3.3.2 Coalitional Rationality

Even with the individual rationality of each node, it still cannot guarantee that no coalition of nodes can benefit from deviating the grand coalition. To see this, we revisit the example in Figure 4.1 part (a). In this game, $m_r = 8$, $m_f = 7$. Let $\alpha = 0.55$, $\beta = 0.6$ to satisfy condition (4.4). Then in grand coalition the total payoff allocations that nodes in S_1 can get is $\sum_{i \in S_1} x_i = 13.25$, while the worth of the coalition S_1 is $v(S_1) = 15.5$. Intuitively, coalition S_1 can collectively get better payoff

allocations by excluding nodes $2, 6, 7$ from their coalition and saving the payments to them, which worth 2.25 in total. Consequently, nodes in S_1 have the incentive to deviate from the grand coalition, which leads some nodes in $\mathbf{N}$ not to follow the routing protocol.

To overcome the difficulty in ensuring coalitional rationality, we modify the δ in Eq. (4.1), from a constant parameter to a function of the coalition S. $\delta(S)$ is the reward for successfully delivering one message copy in coalition S. In particular, define $\delta(S)$ as the ratio of the cooperative behaviors (receiving or forwarding) in S to the total number in $\mathbf{N}$.

$$\delta(S) = \frac{\sum_{i \in S - \{src.\}} (m_r(i) + m_f(i))}{m_r + m_f}.$$

In the grand coalition $\mathbf{N}$, all nodes are cooperative, so $\delta(\mathbf{N}) = 1$. Therefore the condition (4.4) can be rewritten as

$$max(\alpha, \beta) \leq \frac{d(\mathbf{N})}{m_r + m_f} \tag{4.5}$$

Now we are going to prove that given the condition (5), no node can benefit by deviating from the grand coalition and forming a coalition consisting of a subset of nodes.

Lemma 10. *In the forwarding coalitional game* $(\boldsymbol{N}, v)$*, where* $v(S) = \frac{\sum_{i \in S - \{src.\}} (m_r(i) + m_f(i))}{m_r + m_f}$. $d(S) + u \cdot n_{rec}(S)$*, the payoff allocations defined in (4.2) and (4.3) with condition that*

$$max(\alpha, \beta) \leq \frac{d(\boldsymbol{N})}{m_r + m_f}$$

guarantee that no coalition has incentives to deviate from the grand coalition.

Proof. For coalition S that does not include *src.* and *dest.*, it has only one member in S. Consequently the coalitional rationality is immediately guaranteed because it is equivalent to the individual rationality in this case.

Now consider an arbitrary coalition S. We compare the total value of the coalition S, $v(S)$ and the value sum of those nodes in S if they are in the grand coalition, $\sum_{i\in S} x_i$. If $\sum_{i\in S} x_i \geq v(S)$, it means that no subset of nodes can form a coalition obtaining higher total value than they are in the grand coalition.

$\forall S \subseteq N$, s.t. S contains *src.* and *dest.*, we have

$$\begin{aligned}
& \sum_{i\in S} x_i - v(S) \\
= \; & \frac{\sum_{i\in \mathbf{N}}(m_r(i)+m_f(i))}{m_r+m_f} d(\mathbf{N}) + n_{rec}(S) \\
& - \sum_{i\notin S}(\alpha m_r(i) + \beta m_f(i)) \\
& - \frac{\sum_{i\in S}(m_r(i)+m_f(i))}{m_r+m_f} d(S) - n_{rec}(S) \\
\geq \; & d(\mathbf{N}) - max(\alpha,\beta)\sum_{i\notin S}(m_r(i)+m_f(i)) \\
& -(1-\frac{\sum_{i\notin S}(m_r(i)+m_f(i))}{m_r+m_f})d(S) \\
\geq \; & \sum_{i\notin S}(m_r(i)+m_f(i))(\frac{d(\mathbf{N})}{m_r+m_f} - max(\alpha,\beta)).
\end{aligned}$$

The last step of the above derivation is due to the fact that $d(\mathbf{N}) - d(S) \geq \frac{\sum_{i\notin S}(m_r(i)+m_f(i))}{m_r+m_f}(d(\mathbf{N}) - d(S))$.

Since $\sum_{i\notin S}(m_r(i)+m_f(i)) \geq 0$ and we have condition (4.5), we can obtain that $\sum_{i\in S} x_i - v(S) \geq 0$. Therefore, the coalitional rationality is guaranteed. □

4.3.3.3 Efficiency

Finally it is easy to verify the efficiency of the payoff allocation. Actually

$$\begin{aligned}\sum_{i\in\mathbf{N}} x_i &= x_{src.} + \sum_{i\in\mathbf{N}-\{src.\}} x_i \\ &= \delta(\mathbf{N})d(\mathbf{N}) + u \cdot n_{rec}(\mathbf{N}) = v(\mathbf{N}).\end{aligned}$$

We now summarize our analysis results in the following theorem.

Theorem 11. *In the forwarding coalitional game* $(\boldsymbol{N}, v)$*, where*

$$v(S) = \frac{\sum_{i\in S-\{src.\}}(m_r(i) + m_f(i))}{m_r + m_f} \cdot d(S) + u \cdot n_{rec}(S),$$

the payoff allocation X *s.t.,* $\forall i \in N$

$$x_i = \begin{cases} \alpha \cdot m_r(i) + \beta \cdot m_f(i) + u \cdot n_{rec}(i) & \text{if } i \neq src. \\ d(\boldsymbol{N}) - (\alpha m_r + \beta m_f) & \text{otherwise,} \end{cases} \tag{4.6}$$

with the condition that

$$max(\alpha, \beta) = \frac{d(\boldsymbol{N})}{m_r + m_f} - \varsigma, \quad \text{if } d(\boldsymbol{N}) > 0 \tag{4.7}$$

is sufficient to be guaranteed in the core.

Proof. Due to Lemma 9, Lemma 10 and the efficiency analysis in Section 4.3.3.3. □

Theorem 11 guarantees that by using the payoff allocation functions, no coalition of the selfish nodes have the interest to break with the grand coalition. The system will converge to a strongly stable state that nodes are willing to follow the routing protocol and cooperate in forwarding messages.

4.3.4 Complete Design of Incentive Scheme

Based on the payoff allocation functions designed above, in this subsection we specify our complete incentive scheme.

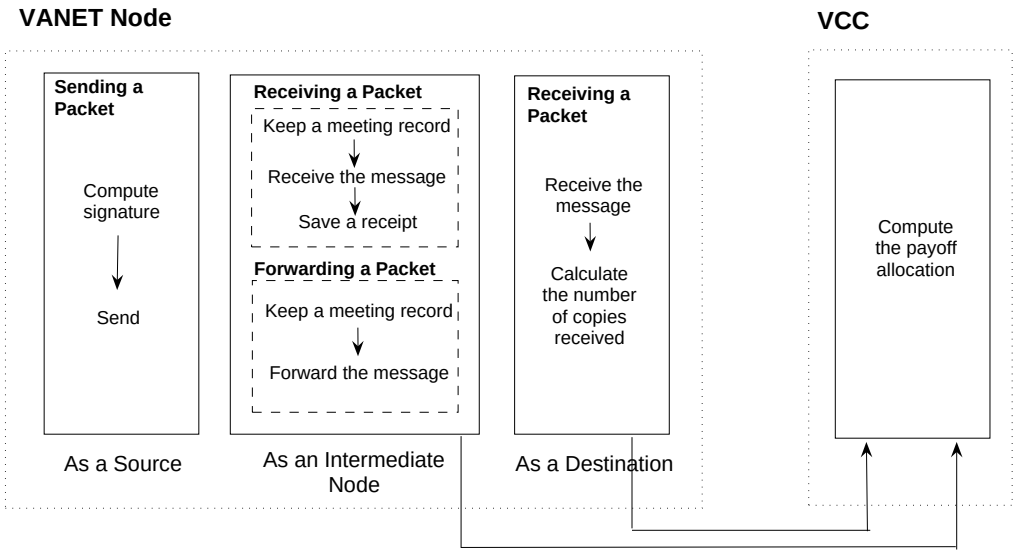


Figure 4.2. Incentive scheme implementation architecture.

We assume that there is a public key infrastructure in the VANETs. Each node i has a public/private key pair Kp_i, Ks_i and a certificate that is digitally signed by a trusted Certificate Authority. Denote $(sign_{Kp}(), verify_{Ks}())$ the digital signature scheme used in VANETs.

The complete incentive scheme consists of the programs installed at each node in the VANETs and the algorithm running at the VCC. The programs at each node can be further divided into three groups of functions, for the source node, the intermediate node and the destination, respectively. The detailed architecture of this incentive scheme is shown in Fig. 4.2.

• **Source Node.** Suppose that *src.* wants to send a message M to *dest.*. *src.* computes a digital signature $sign_{Ks_{src.}}(md(M))$ based on the message it is about to send. *src.* will send the message (or copies) together with the message-specific digital signature $sign_{Ks_{src.}}(md(M))$ to the adjacent intermediate nodes, where $md()$ is a message digest function.

• **Intermediate Nodes.** When a node carrying M meets a subsequent node, the two

nodes first verify each other's identity using the authentication certificates. Each node keeps a brief record of their meeting $(ts, id, Rinf)$, where ts denotes the time when they meet. id is the identity of the other node, and $Rinf$ is the routing information available (that may needs to be exchanged with the other node in many routing protocols such as [18, 8]). Different content of $Rinf$ is defined according to the routing protocols used in VANETs. For example, if $\Re$ makes routing decisions based on historic meeting information [36, 18, 17], $Rinf$ can be the expected probabilities of meeting other nodes in the system.

If the node carrying M decides to forward according to routing protocol $\Re$, it sends the message M together with $sign_{Ks_{src}}(md(M))$ to the subsequent node. After receiving M, the subsequent node saves $sign_{Ks_{src}}(md(M))$ as a receipt. Nodes submits their meeting reports and message receipt to VCC whenever they can connect to it.

• **Destination Node.** If *dest.* receives M or its copies, it waits for a certain amount of time and calculates the total number of copies it receives $d(\mathbf{N})$. When *dest.* can connect to VCC, it submits its receipt, one copy of M together with $d(\mathbf{N})$ to the system.

• **Computing payoff allocations.** The VCC computes the payoff allocations once in a certain time interval, long enough to collect receipts and meeting reports. Whenever nodes can connect to VCC, they can receive their payoff allocations in form of credits. Before VCC starts to compute the payoff allocations, it first matches all meeting records into pairs by the same timestamp and corresponding node *id*s, and produce pairs of meeting record vectors in from of $(ts, id_1, id_2, Rinf_1, Rinf_2)$, $(ts, id_2, id_1, Rinf_2, Rinf_1)$. VCC discards the single meeting records which fail to match with any other ones. VCC counts the number of meeting records submitted by each node, and obtains each $n_{rec}(i)$ in the payoff allocation functions (4.6). Figure 4.3 specifies the protocol to compute the payoff allocations to the nodes who were involved in the transmission of

→ Meeting record vectors
→ Receipts.
→ Routing protocol $\Re$ in the VANET.
→ A FIFO queue, Q, composed of IDs of nodes.

IF not received $d(\mathbf{N})$ from *dest.*
    $d(\mathbf{N}) \leftarrow 0$.
For each node i, $m_r(i) \leftarrow 0$; $m_f(i) \leftarrow 0$.
Add *src.* to Q.
WHILE (Q is not Empty){
    Take an id $id_{current}$ out of Q.
    IF not found any $(ts, id_1, id_2, Rinf_1, Rinf_2)$
      s.t. $id_1 = id_{current}$
        BREAK.
    ELSE FOR each $(ts, id_{current}, id_2, Rinf_1, Rinf_2)$
        IF found receipt from id_2
            Based on $Rinf_1, Rinf_2$ check whether the
              forwarding between $id_{current}$ and id_2 follows $\Re$.
            IF not legal, **BREAK**.
            ELSE
                $m_f(id_{current})$++; $m_r(id_2)$++; Add id_2 to Q.
}
$m_r \leftarrow \sum_i^N m_r(i)$; $m_f \leftarrow \sum_i^N m_f(i)$.
FOR each $i \neq src.$
    $x_i \leftarrow \alpha m_r(i) + \beta m_f(i) + u \cdot n_{rec}(i)$.
$x_{src.} \leftarrow d(\mathbf{N}) - \alpha m_r - \beta m_f$.

Figure 4.3. Protocol to compute the payoff allocations in one game

message M. In order to compute the number of receiving and forwarding behaviors for each node, the protocol adopts a breath-first-search starting from *src.* to trace all cooperative behaviors, using the meeting records information and receipts.

From Fig. 4.3, we can see that the algorithm to allocate payoff is essentially a breadth-first search of the message forwarding tree. Therefore, the time complexity of the algorithm is $O(n)$, where n is the number of receipts that nodes in the system have submitted. Usually the number of n is depending on several factors, e.g., the total number of nodes in the network, the number of messages being transmitted, the basic routing protocol in the system, etc. [3]

[3]Note that the algorithm of the VCC is running on backend machines, so the computing ability of the VCC is not a major concern here.

4.3.5 Preventing Cheating Behaviors

In Section 4.3.3 the analysis shows that the payoff allocation functions in our incentive scheme stimulate the nodes to cooperate. However nodes may still cheat by submitting false information that is used in computing the payoff allocations. In this subsection, we analyze the possible false information that nodes may submit and discuss solutions to prevent these cheating behaviors.

• **False receipts.** Since the payoff allocation of each node in the system essentially depends on the number of receipts that they submit, nodes may save and submit the receipts without forwarding the message. If the nodes behave like this, it will cause the number of copies delivered to *dest.* less than what it should be. In this case, according to the payoff allocation condition (4.7), the amount of payoff allocation that each intermediate node gets decreases as $d(\mathbf{N})$ drops. So by carefully choosing parameters α and β, it can be guaranteed that nodes get punished by losing their payoff shares.

• **False d(N).** Now we consider the case that *dest.* reports false $d(\mathbf{N})$ in collusion with *src.*. Since if $d(\mathbf{N})$ is higher, *src.* can get more payoff shares, *dest.* may declare to receive more than $d(\mathbf{N})$ copies. Actually, our payoff allocation computing protocol can prevent this cheating behavior. Because the protocol traces all effective routes and verifies all forwarders' receipts, any false $d(\mathbf{N})$ will be detected.

• **False meeting records.** According to our analysis in Section 4.3.3 hiding meetings records and not following the routing protocol will not result in higher payoff shares for the nodes. Therefore, the remaining problem is to prevent them from forging false meeting records which have not really happened.

There are two types of forged meeting records: 1) meeting records with false time and meeting nodes ids, i.e. totally forged meeting records; 2) meeting records only with false routing information. One node cannot generate a totally forged meeting record by itself, because our protocol discards all non-paired records as mentioned

above. If two nodes collude in generating false routing information, they can transfer more messages that are not allowed by $\Re$, and hence obtain more payoff allocations than they should. To prevent this kind of cheating behavior, different solutions for different routing protocols are needed. If $\Re$ is based on historic transfer information, (e.g., some $\Re$ bounds the number of replicates of one message), our protocol can detect the forged information since it can verify and record all legal forwardings in the breath-first search. In some other $\Re$s, nodes exchange control information, for instance the expected transfer probabilities. To enforce the nodes to honestly measure and report routing information, similar approaches to those in [109] can be adopted.

4.4 Extended Scheme

In this section, we extend our incentive scheme to address the challenge brought by the limited storage space of nodes. Indeed, storage space is more available for VANETs nodes than traditional multi-hop wireless network nodes. However, it could still be limited since there may be a lot of applications running inside the vehicles which could also consume storage space. It is not likely that the vehicle owner would buy a lot of extra storage space for carrying data messages for other nodes, especially when it can decide the space capacity. Therefore, we believe that under some circumstances, storage space could still be limited for message forwarding in VANETs. Most existing routing protocols (such as [8, 54]) have taken limited storage into consideration; they disseminate some control information to make the decision on how to better utilize the storage space. Consequently, a theoretical solution would be extending our incentive scheme to guarantee the cooperation in truthfully reporting and transferring control information. Nevertheless, such a theoretical solution suffers from a very large overhead. So, in this section, we provide an alternative light-weight incentive approach to solve this problem. Specifically, we extend the payoff allocation functions in the incentive scheme, so that the system can intentionally choose a per-

formance metric to optimize and distribute the payoff to each node according to how its forwarding behavior satisfies the routing goal. As the nodes are selfish and aim to maximize the total payoff shares of their own, we show that it is their dominant strategies [83] to always drop the messages that the system prefers to drop.

It is assumed that in a VANET, nodes only have limited space to store at most P messages. Hence, although forwarding more messages will bring them higher payoff shares, nodes can only carry some of those that they receive. We classify the time to discard a message into two categories: before meeting the subsequent node and after forwarding to the subsequent node. Clearly, in the first case, the forwarding behavior does not occur while in the second case it occurs. Recall that transmission of each message from source to destination is modeled as a forwarding coalitional game. We assume there are Q messages, with different sources or destinations, transferred in the VANET. Therefore there are Q games that a node could possibly participate. Denote G the game set, and $|G| = Q$. Each game g in G can be labeled by the source-destination pair.

We now extend the payoff allocation functions in our incentive scheme. The payoff allocation of node i in game g is defined as

$$x_i(g) = \alpha_i(g) \cdot m_r(i, g) + \beta_i(g) \cdot m_f(i, g) + u \cdot n_{rec}(i, g), \tag{4.8}$$

where $\alpha_i(g)$ (resp. $\beta_i(g)$) is the amount of reward that i can obtain for receiving (resp. forwarding) a message copy in game g. In words, we change the constant unit reward to a reward function on the player and the game. When the game g ends, VCC computes $\alpha_i(g)$ and $\beta_i(g)$ first, before allocating the payoffs.

The design of $\alpha_i(g)$ and $\beta_i(g)$ depends on which performance metric that the system wants to optimize and the corresponding routing protocol. Here we present an example of $\alpha_i(g)$ and $\beta_i(g)$ for systems aiming to maximize the delivery ratio. Define

$$\alpha_i(g) = \beta_i(g) = (d_g(\mathbf{N}) - d_g(\mathbf{N} - \{i\})) \cdot m_f(i, g) \cdot \gamma,$$

where $d_g(\mathbf{N})$ denotes the number of message copies delivered to the destination in game g, and $d_g(\mathbf{N} - \{i\})$ is the number of delivered copies if the node i is excluded from the game. γ is a constant parameter used to scale the total payoff. Intuitively, if $d_g(\mathbf{N}) - d_g(\mathbf{N} - \{i\}) = 0$, it means that the node contributes nothing to the delivery of message. $d_g(\mathbf{N} - \{i\})$ can be computed in the VCC using the meeting records submitted by the nodes. Greater $\alpha_i(g)$ and $\beta_i(g)$ imply that the receiving and forwarding of node i result in higher delivery ratio.

With the above extension, the total payoff shares that a player can obtain in the Q games is $X_i = \sum_{g \in G} x_i(g)$. In the following theorem, the dominant strategy of each node is to contribute more in the games which can bring higher payoff shares to it.

Theorem 12. *Assume that for each game g, the payoff share for node i is defined as Eq. (4.8), and $\alpha_i(g) \propto \beta_i(g) \propto m_f(i, g)$. Then it is a dominant strategy for each node to accept the messages with highest $\alpha_i(g)$ during a transfer opportunity and to remove the messages with lowest $\beta_i(g)$ to make room for the incoming messages.*

Proof. Denote s^* the strategy such that nodes accept the messages with highest $\alpha_i(g)$ during a transfer opportunity and remove the messages with lowest $\beta_i(g)$ to make room for the incoming messages. There are two cases of strategy s' s.t. $s^* \neq s'$.
Case 1. Let s' denote the strategy that in some transfer opportunity, the node decides to accept a message in g' instead of g s.t. $\alpha_i(g') < \alpha_i(g)$. Other actions are the same as in s. Then for player i the total payoff allocation difference of taking strategy s' and s is

$$\begin{aligned} & X_i(s_i') - X_i(s^*) \\ = \; & \alpha_i(g') - \alpha_i(g) + \beta_i(g') \cdot m_f(i, g') - \beta_i(g) \cdot m_f(i, g) \\ \leq \; & 0. \end{aligned}$$

Case 2. Let s'' denote the strategy that in some transmission, in order to make room for the incoming messages, the node remove a message g' instead of g, s.t. $\beta_i(g') > \beta_i(g)$. We can obtain that

$$X_i(s_i'') - X_i(s^*) = \beta_i(g) \cdot m_f(i,g) - \beta_i(g') \cdot m_f(i,g') \leq 0.$$

Therefore, strategy s^* is a dominant strategy for each node. □

We note that $\alpha_i(g)$ and $\beta_i(g)$ are computed by the VCC after the game g ends, and the knowledge of $\alpha_i(g)$ and $\beta_i(g)$ is not forwarded in the VANETs to reach node i. Then how can each node know $\alpha_i(g)$ and $\beta_i(g)$ in order to maximize its own total payoff? Actually each node can approximate the parameters using the local information and what it receives from the VCC. There are a lot of algorithms that nodes can apply to estimate $\alpha_i(g)$ and $\beta_i(g)$ for each game. The key idea is that if in history a node got high unit payoff from forwarding for a source-destination pair, it is likely that this trend will last for some time as long as its mobility pattern does not change dramatically. Here by mobility pattern we mean the path followed by a vehicle during an extensive time frame. Based on its historic behaviors and the corresponding payoff shares, nodes can estimate $\alpha_i(g)$ and $\beta_i(g)$ in the current game. After the nodes learn for long enough time, the system will converge to the equilibrium in which nodes take their dominant strategies and meanwhile the system objective can be met. For example, one VANET node passes by a department store every morning and afternoon on the way between home and office, and this department store regularly disseminates the announcement of sale information. Hence this node can learn from its previous experience that helping forwarding the messages for the department store gains more payoffs than for other unknown sources. Therefore it can decide which message to discard if space is limited, to the best of its own interest. We will verify this by the experiments in Section 4.5.4.

4.5 Evaluation

In this section, we extensively evaluate our incentive scheme using GloMoSim [46]. Our objectives are two folds: a) to verify that our scheme effectively stimulates cooperation in VANETs, b) to evaluate the impact of our scheme in improving the system performance in terms of delivery ratio and delay time, when selfish behaviors appear in VANETs. The experiments are conducted on the traces from a real vehicular network, DieselNet [17]. We test our incentive scheme based on two different routing protocols, MV [18] and binary Spray-and-Wait [95]. In Section 4.5.4, we also evaluate the performance of our incentive scheme with limited storage space of nodes.

4.5.1 Settings

• **Traces from DieselNet.** We evaluate our incentive scheme on testbed traces from DieselNet [17]. It is a vehicular network testbed consisting of 40 buses, of which only a subset is on the road each day. Each bus in DieselNet carries a computer of 40G storage space and a GPS device. They are set to transmit random data to other nodes whenever they are within the range. The traces from Feb 6, 2007 until May 14, 2007 [8] (58 files) are used. These traces are from the buses running routes serviced by UmassTransit. The mobility of these buses are determined by UmassTransit and the bus routes can be found at $http://www.umass.edu/campus\_services/transit/$. The average number of meetings between buses per day is 147.5. Each tracefile consists of the connection events occurring during a day. For each meeting event, the following information is recorded as a tuple: the MAC address of the bus sending data, the MAC address of the bus receiving data, the time of meeting, transmission size and meeting location. The traces are generated using a default rate of 4 messages per hour of each bus for every other bus on the road and the size of each message is 1KB. We import the traces of 11 buses each day into GloMoSim, and vary the message

generating rate in our experiments. The experiment results are averaged over 58 traces.

• **Routing protocols.** In each of the VANET network, we test our incentive scheme with two different routing protocols, MV and binary Spray-and-Wait. The objective of using two routing protocols in our evaluation is not to compare them, but to show that our incentive scheme can guarantee packet forwarding cooperation for the systems with different routing protocols. Although MV and Spray-and-Wait are initially designed for delay-tolerant routing, they are also very useful in the multi-hop delay-tolerant scenarios for vehicular networks, such as delivering commercial advertisements regarding sale information at a store, with low vehicle density. We choose these two protocols because each of them is representative in the two categories of routing protocols. The key idea of MV is to estimate the delivery probability of each node to the message destination using historic contact information, while Spray-and-Wait is based on message replication but restricts the number of copies for each message to L. In the experiments on MV, the time unit in calculating the delivery probabilities is set to 1 minute and node do not keep copies after forwarding them. In binary Spray-and-Wait set $L = 12$.

• **Metrics.** To show that our incentive scheme indeed provides effective stimulation for forwarding cooperation, we measure the *accumulative credit* of the forwarding nodes when they have different forwarding behaviors. Note that nodes have to spend credits to send their own messages, so they have the incentive to earn more credits for future use. Our incentive scheme computes payoff allocations every 30 minutes. Set $u = 0.01$, $\beta = \frac{d(\mathbf{N})}{m_r+m_f} - 0.02$ and $\alpha = \beta - 0.05$. We set u, β and α as above because we need to guarantee that $\alpha > 0$, $\beta > 0$ and $\max(\alpha, beta) \leq \frac{d(\mathbf{N})}{m_r+m_f}$. In addition, we let $\alpha < \beta$ due to the reason that the behavior of forwarding a message requires not only sending it but also storing the message until meeting the subsequent node. Hence it makes sense to reward a little more to the behavior of forwarding. We set

the value of u relatively small because making meeting records requires less energy consumption than receiving and forwarding data messages. To evaluate the impact of our incentive scheme on the system performance, we measure both *delivery ratio* and *delay time*.

Nodes use IEEE 802.11 (at 11Mbps) as the MAC layer protocol. The radios' transmission range is set to 250 meters. The radio propagation model is two-way.

4.5.2 Accumulative Credit

The first set of experiments is to verify that with our incentive scheme, nodes always lose credits if they do not faithfully follow the routing protocols. Specifically, we define the selfish behavior as only forwarding the messages destined to the nodes in its own coalition. In other words, selfish nodes do not follow routing protocol if the incoming message is not destined within its coalition. We vary the size of the coalition which consists of selfish nodes, and all other nodes remain cooperative. We set up two different coalition scenarios for the selfish users. The first scenario is that there are two coalitions in total, with one of size 6 and the other of size 5. The second scenario is that the 11 active nodes form 3 coalitions, consisting of 4 nodes, 4 nodes, and 3 nodes respectively. We record the average accumulative credits of selfish nodes in coalitions of different sizes and compare them with the average accumulative credits of cooperative nodes.

Figure 4.4 and Figure 4.5 respectively show the results of MV and Spray-and-Wait in DieselNet. We can observe that at any time, nodes get the most credits if they cooperatively follow the routing protocol for all the messages. The smaller the coalition is, the less credits can the selfish nodes obtain. Note that because the buses only operate in the daytime, the credits of the nodes remain the same when there are no messages transmissions taking place. From the figures it is clear that with either MV or Spray-and-Wait applied in the network, selfish nodes can never receive

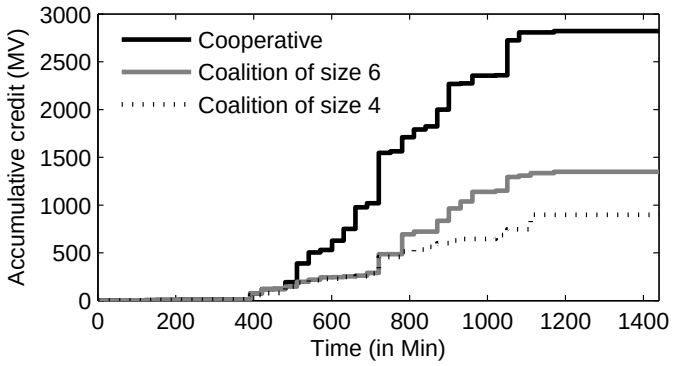


Figure 4.4. (MV) Accumulative credit of node in coalition of different sizes v.s. cooperative nodes.

more credits than cooperatively forwarding all the messages. Therefore, our incentive scheme provides an effective stimulating mechanism for nodes to cooperate.

4.5.3 Impacts on System Performance

Our second set of experiments is to show that when the network has selfish nodes, our incentive scheme can improve the system performance. In particular, we demonstrate how the incentive scheme can impact the delivery ratio and delay time when 30% and 10% of the nodes in the VANET are selfish. The selfish nodes are randomly picked and the selfish behavior is defined the same as above.

We vary the message generating rate and measure the delivery ratio and the average max-delay time per message. Figure 4.6 and Figure 4.8 shows the results of the experiments on MV, and Figure 4.7 and Figure 4.9 demonstrates the results on Spray-and-Wait. As shown in Figure 4.6, our scheme increases the delivery ratio of MV routing protocol by up to 23.9%, when there are 30% nodes form a coalition in the network. We also find that when there are 10% nodes in the system, the delivery ratio is higher than the case of 30% selfish nodes. Similar conclusion can be drawn from Figure 4.7 that our incentive scheme can increase the delivery ratio of Spary-and-Wait by up to 9.44% when there are 30% selfish nodes. Furthermore, it can be

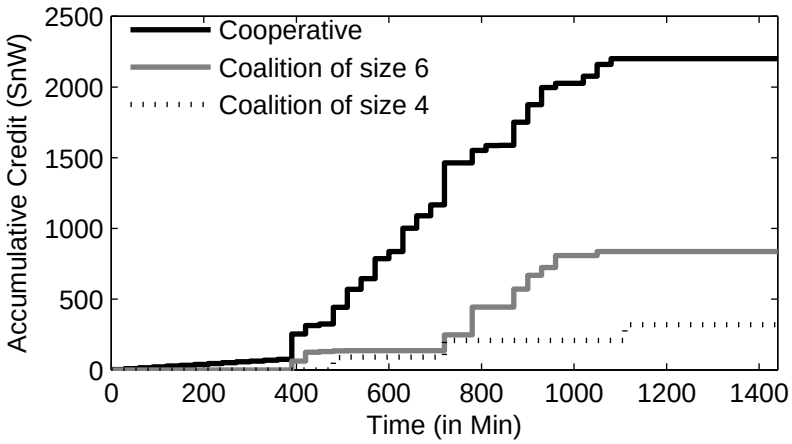


Figure 4.5. (Spray-and-Wait) Accumulative credit of node in coalition of different sizes v.s. cooperative nodes.

seen from Figure 4.8 and Figure 4.9 that our incentive scheme can always shorten the average max-delay time of messages (up to 9.5% for MV, and up to 14.5% for Spray-and-Wait). Again more selfish nodes in the system result in longer delay time.

4.5.4 Experiments on Extended Scheme

In this subsection, we evaluate our extended scheme when the nodes only have limited storage space. We assume that all nodes are cooperative in that they always receive and forward the packets for others, and compare the results of two sets of experiments. In one set, we let the nodes randomly drop messages when the storage space is full, while in the other set, we let nodes learn from the credits received in history, and keep the message destined to the most profitable destinations to them. The cooperative nodes learn from the credits received in history, and keep the message destined to the most profitable destinations to them. The noncooperative nodes just randomly choose some of the messages in the storage space to drop.

Figure 4.10 represents the system delivery ratios when the system converges to the stable state. We vary the storage capacity from 50 messages to 250 messages

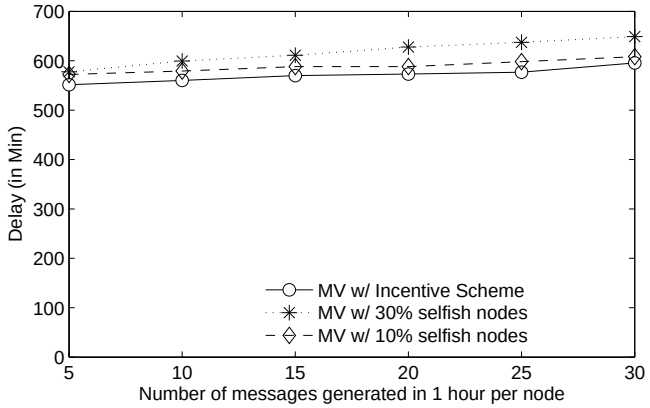


Figure 4.6. Delivery ratios achieved with and without our incentive scheme when MV is used.

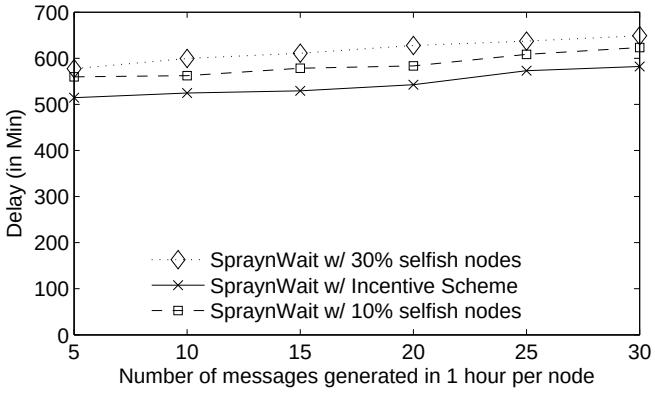


Figure 4.7. Delivery ratios achieved with and without our incentive scheme when Spray-and-Wait is used.

and compare the results from the two dropping behaviors. It is can be observed that cooperative behavior always results in higher delivery ratio than random dropping. The difference is more significant when the storage space is smaller. Hence, we can conclude that the cooperative dropping behavior can increase the system delivery ratio compared with randomly dropping.

Figure 4.11 shows the accumulative credits of the cooperative and random behaviors. It is clear that at any time, the cooperative behavior brings the nodes more

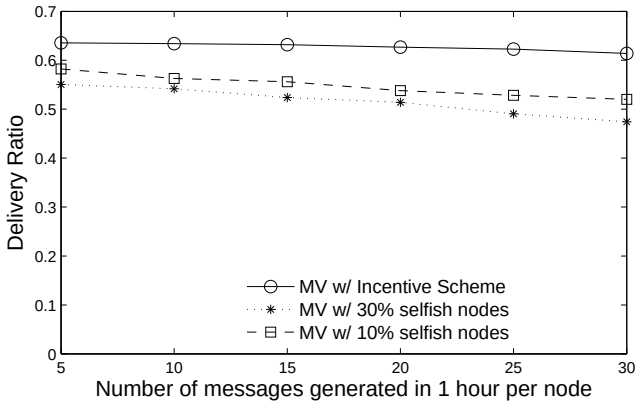


Figure 4.8. Average delay of messages in the system with and without our incentive scheme when MV is used.

credits than randomly dropping. Therefore, the extended scheme indeed encourages the nodes to cooperatively drop messages.

4.5.5 Overhead

In this subsection, we examine the overhead introduced by our scheme. For mobile nodes, we focus on the storage space occupied by our scheme and the overhead for making meeting records. For the VCC, we examine the time to calculate the credit for each node. We assume that mobile nodes can connect with the VCC once a day. We use crypto++ 5.5.2 [35] for the cryptographic scheme implementation. The tests are performed on a laptop Intel Core 2.67 GHz processor under Windows Vista in 32-bit mode. **Communication Overhead** We use Elliptic Curve Cryptography (ECC) for the PKI implementation. We set the key length of ECDSA to 192 bits, and the digital signature of each message digest is 48 bytes. Assume the length of the message is x bytes, and the total length of the data message is $(48 + x)$ bytes. In our experiments, $x = 1000$, so the communication overhead for data transmission is about 4.6%.

For the authentication process when two nodes meet, we also use 48 bytes ECDSA certificate. On average, it requires 6.38 mseconds for the verification per node.

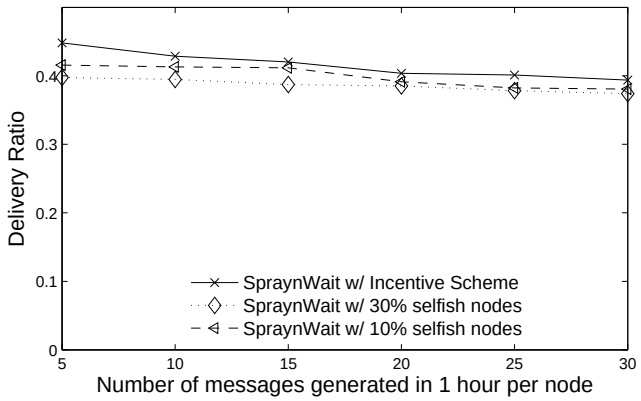


Figure 4.9. Average delay of messages in the system with and without our incentive scheme when Spray-and-Wait is used.

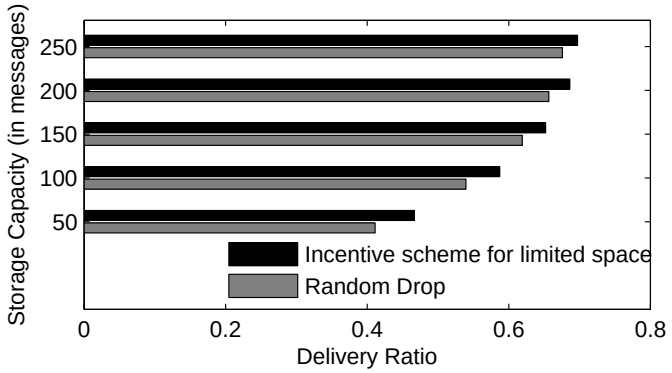


Figure 4.10. Delivery Ratio in experiments on extended scheme as function of different space limits.

Storage Requirement In our scheme, the storage requirement comes from two parts: meeting records and message receipts that nodes need to keep before connecting to the VCC.

The average storage usage for meeting records on each node is 118.4 bytes.

We evaluate the storage requirement for message receipts with different message rates per node, and show the results in Figure 4.12.

As we can see that the space requirements for storing message receipts are very small per node. For MV protocol the storage overhead is within the range of (5,25)

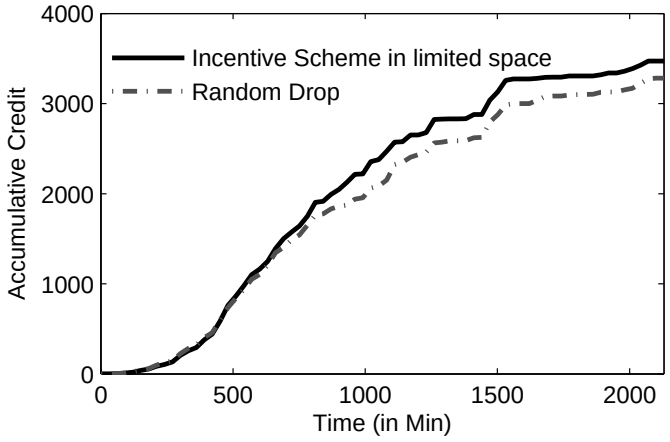


Figure 4.11. Accumulative credit of the nodes. Following our extended scheme vs. random strategy to drop messages.

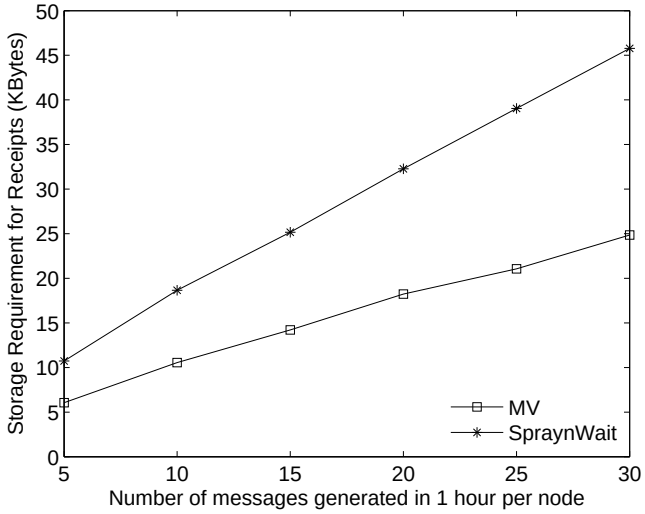


Figure 4.12. Storage requirement for saving receipts on each node.

KBytes when the number of per-hour messages changes from 5 to 30. SpraynWait protocol requires more storage, ranging from 10 to about 45 KBytes.

Computation Overhead on VCC We measure the time to compute the allocation of credits for all nodes in the VCC as shown in Figure 4.13. When there are more messages generated in the system, the VCC uses more time to verify each message forwarding behavior and correspondingly allocate the credits to each cooperative nodes. Overall, it is very fast for VCC to conduct such computation within about one minute.

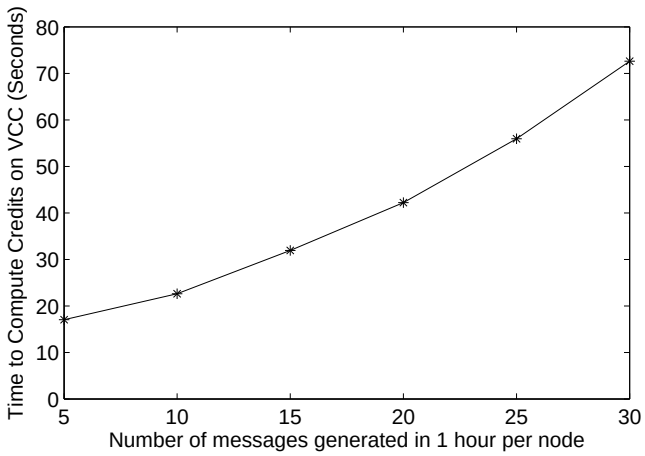


Figure 4.13. Time to compute credits on VCC.

4.6 Summary

In this chapter, a simple and effective incentive scheme in VANETs is proposed to stimulate the forwarding cooperation of nodes. We are the first to present an incentive scheme for VANETs with theoretical guarantee. We formally prove, in a coalitional game model, that with our scheme every relevant node cooperates in forwarding messages as required by the routing protocol. An extension is made to scenarios with constrained storage space, and a light-weight approach to stimulate cooperation is proposed. We integrate our incentive scheme with MV and Spray-

and-Wait respectively and evaluate the system performance on testbed traces. The experimental results show that our incentive scheme provides effective stimulation for nodes to cooperate and prevents the degradation of system performance in VANETs with selfish nodes. Although our work provides theoretical guarantee on the cooperation, we only test it using testbed traces. In the future, more testbed experiments in the real world are needed to further verify our schemes and improve the design based on real implementation problems.

In designing our schemes, we assume that there are no communication failures for the control messages at physical level of each link. However, in reality, the communication capacity is affected by conditions related to the environment, e.g., shadow fading. [51, 34, 33] It means that in the inter-vehicle communications of our scheme (e.g., identity verification and making meeting records), errors may occur due to failures of physical lever and thus consequently the link drops. The authors of [1] proved that the error probability is log-concave for a wide class of multidimensional modulation formats. Based on this finding, they derived nice results on upper and lower bounds, and local bounds that are tight in a given region of interest for the error probability. All the works above show that the performance of our incentive scheme could be affected by the physical level communication failures. In particular, if the communication failure occurs when the nodes have made meeting records but the data transmission has not finished, the VCC will allocate inaccurate amount of credits to the intermediate nodes, since the destination cannot receive the correct data in this case.

In our future work, we hope to reduce the impact of communication failures on our incentive schemes. We can work towards the following two directions: a) We will try to further reduce the length of communication overhead introduced by our schemes. In this way, the probability of link failures occurring in transmitting control information can be reduced; b) We can leverage existing physical layer techniques

in wireless networks to estimate the link residual time based on the surrounding conditions. Consequently, more accurate calculations of credits can be conducted on the VCC.

Part III

Privacy Preserving Distributed Data Mining

With the development of distributed computing environment, many learning problems now have distributed input data. In such distributed scenarios, privacy concerns often become a big issue. For example, if medical researchers want to apply machine learning to study health care problems, they need to collect the raw data from hospitals and the follow-up information from patients. Then the privacy of the patients must be protected, according to the privacy rules in Health Insurance Portability and Accountability Act (HIPAA), which establishes the regulations for the use and disclosure of Protected Health Information [53].

A natural question is why the researchers would want to build a learning model (e.g, neural networks) without first collecting all the training data on one computer. If there is a learner trusted by all the data holders, then the trusted learner can collect data first and build a learning model. However, in many real-world cases it is rather difficult to find such a trusted learner, since some data holders will always have concerns like "What will you do to my data?" and "Will you discover private information beyond the scope of research?". On the other hand, given the distributed and networked computing environments nowadays, collaborations will greatly benefit the scientific advances. The researchers have the interest to obtain the result of cooperative learning even before they see the data from other parties. As a concrete example, [30] stated that the progress in neuroscience could be boosted by making links between data from labs around the world, but some researchers are reluctant to release their data to be exploited by others because of privacy and security concerns. More specifically, the neuroscientist in Dartmouth College found it difficult to encourage the sharing of brain-imaging data because there was possibility that the raw data could be misused or misinterpreted [40]. Therefore, there is a strong motivation for learners to develop cooperative learning procedures with privacy preservation.

CHAPTER 5

PRIVACY PRESERVING BACK-PROPAGATION NEURAL NETWORKS[27]

5.1 Background and Motivation

In this section, we focus on one of the most popular techniques in machine learning, multi-layer neural networks [91, 71], in which the privacy preservation problem is far from being practically solved. In [104] a preliminary approach is proposed to enable privacy preservation for gradient descent methods in general. However, in terms of multi-layer neural networks, their protocol is limited as it is only for one simple neural network configuration with one node in the output layer and no hidden layers. Although their protocol is elegant in its generality, it may be very restricted in practice for privacy preserving multi-layer neural networks.

We propose a light-weight two-party distributed algorithm for privacy preserving back-propagation training with vertically partitioned data[1] (i.e., when each party has a subset of features). Our contributions can be summarized as follows. (1) Our work is the first to investigate the problem of training multi-layered neural networks over vertically partitioned databases with privacy constraints. (2) Our algorithms are provably private in the semi-honest model [48] and light-weight in terms of computational efficiency. (3) Our algorithms include a solution to a challenging technical

[1]For horizontally partitioned scenario (i.e., when each party holds a subset of data objects with the same feature set), there is a much simpler solution that one party trains the neural network first and passes the training result to another party so that she can further train it with her data, and so on. So in this chapter we only focus on the vertical partition case, which is much more technically challenging.

problem, namely privacy preserving computation of activation function. This problem is highly challenging because most of the frequently used activation functions are infinite and continuous while cryptographic tools are defined in finite fields. To overcome this difficulty, we propose the first cryptographic method to securely compute sigmoid function, in which an existing piecewise linear approximation of the sigmoid function [82] is used. In order to make our algorithms practical, we do not adopt the costly circuit evaluation based approaches [114]. Instead, we use a homomorphic encryption based approach and the cryptographic scheme we choose is ElGamal [45]. (4) Both analytical and experimental results show that our algorithms are light-weight in terms of computation and communication overheads.

The rest of the chapter is organized as follows. In Section 5.2, we introduce the technical preliminaries including notations, definitions and problem statement. In Section 5.3, we present the novel privacy preserving back-propagation learning algorithm and two key component secure algorithms. Then we provide security analysis of the algorithm as well as the computation and communication overhead. In Section 5.6, with comprehensive experiments on various datasets, we verify the effectiveness and efficiency of our algorithm. After that, we summarize this chapter.

5.2 Technical Preliminaries

In this section we give a brief review of the version of the Back-Propagation Network (BPN) algorithm we consider [90] and introduce the piecewise linear approximation we use for the activation function. We also give a formal statement of problem with a rigorous definition of security. Then we briefly explain the main cryptographic tool we use, ElGamal [45].

5.2.1 Notations for back-propagation learning

For ease of presentation, in this chapter we consider a neural network of three layers, where the hidden layer activation function is sigmoid and the output layer is linear. Note that it is trivial to extend our work to more layers.

Given a neural network with a-b-c configuration, one input vector is denoted as $(x_1, x_2, \cdots, x_a)$. The values of hidden layer nodes are denoted as $\{h_1, h_2, \cdots, h_b\}$, and the values of output layer nodes are $\{o_1, o_2, \cdots, o_c\}$. w^h_{jk} denotes the weight connecting the input layer node k and the hidden layer node j. w^o_{ij} denotes the weight connecting j and the output layer node i, where $1 \leq k \leq a, 1 \leq j \leq b, 1 \leq i \leq c$.

We use Mean Square Error (MSE) as the error function in the back-propagation algorithm, $e = \frac{1}{2}\sum_i (t_i - o_i)^2$. For the neural networks described above, the partial derivatives are listed as (5.1) and (5.2), for future reference.

$$\frac{\partial e}{\partial w^o_{ij}} = -(t_i - o_i)h_j \tag{5.1}$$

$$\frac{\partial e}{\partial w^h_{jk}} = -h_j(1 - h_j)x_k \sum_{i=1}^{c}[(t_i - o_i)w^o_{ij}] \tag{5.2}$$

5.2.2 The piecewise linear approximation of activation function

In this subsection, we introduce the piecewise linear approximation of activation function. The major reason of introducing the approximation is that cryptographic tools work in finite fields and thus can not be directly applied to the secure computation of functions like sigmoid. Approximating the activation function in a piecewise way offers us an opportunity to apply cryptographic tools to make the computation of sigmoid function privacy-preserving.

Equation (5.3) is a piecewise linear approximation [82] of the sigmoid function $\frac{1}{1+e^{-x}}$. Our privacy preserving algorithm for back-propagation network learning is based on this approximation.[2]

$$y(x) = \begin{cases} 1 & x > 8 \\ 0.015625x + 0.875 & 4 < x \leq 8 \\ 0.03125x + 0.8125 & 2 < x \leq 4 \\ 0.125x + 0.625 & 1 < x \leq 2 \\ 0.25x + 0.5 & -1 < x \leq 1 \\ 0.125x + 0.375 & -2 < x \leq -1 \\ 0.03125x + 0.1875 & -4 < x \leq -2 \\ 0.015625x + 0.125 & -8 < x \leq -4 \\ 0 & x \leq -8 \end{cases} \tag{5.3}$$

5.2.3 Security definition and problem statement

• **Semi-honest model.** As many existing privacy preserving data mining algorithms (e.g., [70, 113]), we adopt semi-honest model in this chapter. Semi-honest model is a standard adversary model in cryptography [48]. In this chapter the security of our algorithm is guaranteed in this model. When computing function f in a distributed fashion, semi-honest model requires that each party that participates in the computation follow the algorithm, but a party may try to learn additional information by analyzing the messages that she receives during the execution. In order to guarantee the security of distributed algorithm of computing f, it must be

[2]It is easy to extend our work to other piecewise linear approximations of activation function. Here we choose this specific approximation as an example to demonstrate in detail how our algorithms work.

ensured that each party can learn nothing beyond what can be implied by her own input and output.

Semi-honest model is a right fit for our setting, because normally participants want to learn the neural network learning results and thus they are willing to follow the algorithm to guarantee the results correctness. The security guaranteed in semi-honest model can relieve the concerns about their data privacy. Of course, in reality there may be scenarios in which there are malicious adversaries. It has been shown (see [48]) that a distributed algorithm that is secure in the semi-honest model can be converted to one that is secure in the malicious model, with some additional costs in computation and communications for zero knowledge proofs. (It is a highly challenging task to reduce these additional costs to achieve security in the malicious model. We leave it as one of our future research topics.)

Based on semi-honest model, the problem of privacy preserving back-propagation neural networks learning in this chapter, is stated below.

• **Privacy preserving two-party distributed neural network training.** Suppose that a set of training samples are vertically partitioned between two parties A and B. A holds a dataset D_1 with m_A attributes for each data entry. B holds a dataset D_2, with m_B attributes for each data entry. We denote one data entry in D_1 as $x_A = (x_1, x_2, \cdots, x_{m_A})$ and in D_2 $x_B = (x_{m_A+1}, \cdots, x_{m_A+m_B})$.

Privacy preserving two-party distributed neural network training is that in each round of neural network learning, two parties jointly compute the additive values of connection weights without compromising their privacy of input data.[3] Formally, with training samples x_A and x_B from party A and B respectively and a target value

[3]In this chapter, we provide an algorithm in which two parties learn all the weights after each round of training. Note that this algorithm can be extended to a more secure fashion by making each party hold only a random additive share of each weight at the end of each round and continue to the next round with the partitioned weights [48]. But in this case much more computational overhead will be added. So for efficiency reasons, we keep the algorithm as it is.

$t(x)$, our goal is to let each party get her own share of the additive value of each weight Δw, without revealing the original training data, x_A or x_B, to each other.

Note that, in this chapter, we restrict our attention to the privacy concerns brought by insiders (i.e., participants of the distributed neural network training) only. The security and privacy issues associated with outsider attacks (i.e., attacks by non-participating parties and attacks in the communication channels) are orthogonal issues beyond the scope of this chapter. In practice, if our algorithms are to be used, appropriate access control and security communication techniques must also be used, to guarantee that all sensitive information is transmitted over secure channels and unauthorized access to the system is prevented.

5.3 Privacy preserving neural network learning

In this section, we present a privacy-preserving distributed algorithm for training the neural networks with back-propagation algorithm. A privacy preserving testing algorithm can be easily derived from the feed-forward part of the privacy-preserving training algorithm.

Our algorithm is composed of many smaller private computations. We will look into them in detail after first giving an overview.

5.3.1 Privacy preserving neural network training algorithm

Here we build the privacy preserving distributed algorithm for the neural network training process under the assumption that we already have the algorithm to securely compute the piecewise linear function (Algorithm 2) and the algorithm to securely compute the product of two numbers held by two parties (Algorithm 3). We will explain the two component algorithms in detail later.

For each training iteration, the input of the privacy-preserving back-propagation training algorithm is $\langle \{x_A, x_B\}, t(x) \rangle$, where x_A is held by party A, while x_B is held

by party B. $t(x)$ is the target vector of the current training data and it is known to both parties. The output of algorithm is the connection weights of output layer and hidden layer, i.e., $\{w_{ij}^o, w_{jk}^h | \forall k \in \{1, 2, \cdots, a\}, \forall j \in \{1, 2, \cdots, b\}, \forall i \in \{1, 2, \cdots, c\}\}$.

The main idea of the algorithm is to secure each step in the non-privacy-preserving back-propagation algorithm, with two stages, feeding forward and back-propagation. In each step, neither the input data from the other party nor the intermediate results can be revealed. In particular, we apply Algorithm 2 to securely compute the sigmoid function, and Algorithm 3 is used to guarantee privacy preserving product computation.

To hide the intermediate results such as the values of hidden layer nodes, the two parties randomly share each result so that neither of the two parities can imply the original data information from the intermediate results. Here by "randomly share", we mean that each party holds a random number and the sum of the two random numbers equals to the intermediate result. Note that with intermediate results randomly shared among two parties, the learning process can still securely carry on to produce the correct learning result (see correctness analysis in Section 5.3.1.1).

After the entire process of private training, without revealing any raw data to each other, the two parties jointly establish a neural network representing the properties of the union dataset.

Our training algorithm for back-propagation neural networks can be summarized as in Algorithm 1.

For clarity of presentation, in Algorithm 1, we separate the procedure to compute $\sum_i [-(t_i - o_i) w_{ij}^o] h_j (1 - h_j)$ and explain it in Algorithm 1.1.

Algorithm 1 Privacy preserving distributed algorithm for back-propagation training

Initialize all weights to small random numbers, and make them known to both parties.

Repeat

for all training sample $\langle\{x_A, x_B\}, t(x)\rangle$ **do**

Step1: feed forward stage

(1.1) **For each** hidden layer node h_j, party A computes $\sum_{k\leq m_A} w^h_{jk}x_k$, and party B computes $\sum_{m_A<k\leq m_A+m_B} w^h_{jk}x_k$.

(1.2) Using Algorithm 2, party A and B jointly compute the sigmoid function for each hidden layer node h_j, obtaining the random shares h_{j1} and h_{j2} respectively s.t. $h_{j1} + h_{j2} = f(\sum_k w^h_{jk}x_k)$.

(1.3) **For each** output layer node o_i, Party A computes $o_{i1} = \sum_i w^o_{ij}h_{j1}$ and party B computes $o_{i2} = \sum_i w^o_{ij}h_{j2}$, s.t. $o_i = o_{i1} + o_{i2} = \sum_i w^o_{ij}h_{j1} + \sum_i w^o_{ij}h_{j2}$.

Step2: back-propagation stage

(2.1) **For each** output layer weight w^o_{ij}

Party A and B apply Algorithm 3 to securely compute the product $h_{j1}o_{i2}$, obtaining random shares r_{11} and r_{12} respectively, s.t. $r_{11} + r_{12} = h_{j1}o_{i2}$. Similarly they compute the random partitions of $h_{j2}o_{i1}$, r_{21} and r_{22}, s.t. $r_{21} + r22 = h_{j2}o_{i1}$.

Party A computes $\Delta_1 w^o_{ij} = (o_{i1} - t_i)h_{j1} + r_{11} + r_{21}$ and B computes $\Delta_2 w^o_{ij} = (o_{i2} - t_i)h_{j2} + r_{12} + r_{22}$.

(2.2) **For each** hidden layer weight w^h_{jk}

Using Algorithm 1.1, party A and B jointly compute $\sum_i[-(t_i - o_i)w^o_{ij}]h_j(1 - h_j)$, obtaining random shares q_1 and q_2 respectively, s.t. $q_1 + q_2 = \sum_i[-(t_i - o_i)w^o_{ij}]h_j(1 - h_j)$.

If $k \leq m_A$, that is A holds the input attribute x_k, applying Algorithm 3 to securely compute $x_k q_2$, A and B respectively get r_{61} and r_{62} s.t. $x_k q_2 = r_{61} + r_{62}$. Then $\Delta_1 w^h_{jk} = q_1 x_k + r_{61}$ and $\Delta_2 w^h_{jk} = r_{62}$.

If $m_A < x_k \leq m_A + m_B$, A and B apply Algorithm 3 to get r_{61} and r_{62} s.t $x_k q_1 = r_{61} + r_{62}$. In this case, $\Delta_1 w^h_{jk} = r_{61}$, $\Delta_2 w^h_{jk} = q_2 x_k + r_{62}$.

A and B respectively compute $\Delta_1 w^h_{jk}$, $\Delta_2 w^h_{jk}$.

Step3:

A (B, resp.) sends $\Delta_1 w^o_{ij}$, $\Delta_1 w^h_{jk}$($\Delta_2 w^o_{ij}$, $\Delta_2 w^h_{jk}$, resp.) to B(A, resp.). A and B compute $w^o_{ij} \leftarrow w^o_{ij} - \eta(\Delta_1 w^o_{ij} + \Delta_2 w^o_{ij})$ for each hidden layer weight, and $w^h_{jk} \leftarrow w^h_{jk} - \eta(\Delta_1 w^h_{jk} + \Delta_2 w^h_{jk})$ for each output layer weight.

end for

Until (termination condition)

Algorithm 1.1 Securely computing

$$\sum_i [-(t_i - o_i) w_{ij}^o] h_j (1 - h_j).$$

Input: h_{j1}, o_{i1} (h_{j2}, o_{i2} resp.) for party A (party B resp.) abtained inside Algorithm 1.
Output: random shares q_1, q_2 for A and B resp.

1. Using Algorithm 3, party A and B get random shares of $h_{j1}h_{j2}$, r_{31} and r_{32} respectively, s.t. $h_{j1}h_{j2} = r_{31} + r_{32}$.
2. For clarity, we name intermediate results as $h_{j1} - h_{j1}^2 - 2r_{31} = p_1$, $\sum_i (-t_i + o_{i1}) w_{ij}^o = s_1$, $h_{j2} - h_{j2}^2 - 2r_{32} = p_2$ and $\sum_i o_{i2} w_{ij}^o = s_2$.
3. Party A computes p_1 and s_1; B computes p_2 and s_2.
4. Applying Algorithm 3, party A gets r_{41} and r_{51}, party B gets r_{42} and r_{52}, s.t. $r_{41} + r_{42} = s_1 p_2$, $r_{51} + r_{52} = s_2 p_1$.
5. Name $q_1 = s_1 p_1 + r_{41} + r_{51}$ and $q_2 = s_2 p_2 + r_{42} + r_{52}$. Party A computes q_1 and party B computes q_2 locally.

5.3.1.1 Correctness

We show that if party A and party B follow Algorithm 1, they will jointly derive the correct weight update result in each learning round, with each of them holding a random share.

For any output layer weight w_{ij}^o, in step (2.1) of Algorithm 1 we have

$$\begin{aligned}
& \Delta_1 w_{ij}^o + \Delta_2 w_{ij}^o \\
= \; & (o_{i1} - t_i) h_{j1} + r_{11} + r_{21} + (o_{i2} - t_i) h_{j2} + r_{12} + r_{22} \\
= \; & (o_{i1} - t_i) h_{j1} + (o_{i2} - t_i) h_{j2} + h_{j1} o_{i2} + h_{j2} o_{i1} \\
= \; & -(t_i - o_{i1} - o_{i2})(h_{j1} + h_{j2}) \\
= \; & -(t_i - o_i) h_j \\
= \; & \frac{\partial e}{\partial w_{ij}^o} \\
= \; & \Delta w_{ij}^o.
\end{aligned}$$

Therefore the additive random shares of the two parties can add up to the correct value for hidden layer connection weights.

Now we show that the correctness is also guaranteed for hidden layer weights.

In step (2.2) of Algorithm 1, it is easy to get that no matter which party holds the attribute x_k, we always have

$$\begin{aligned}
& \Delta_1 w_{jk}^h + \Delta_2 w_{jk}^h \\
= \; & x_k q_1 + x_k q_2 \\
= \; & (s_1 p_1 + r_{41} + r_{51} + s_2 p_2 + r_{42} + r_{52}) x_k \\
= \; & (s_1 + s_2)(p_1 + p_2) x_k \\
= \; & (\sum_i (-t_i + o_{i1}) w_{ij}^o + \sum_i o_{i2} w_{ij}^o) \\
& (h_{j1} - h_{j1}^2 - 2r_{31} + h_{j2} - h_{j2}^2 - 2r_{32}) x_k \\
= \; & \sum_k [-(t_i - o_i) w_{ij}^o](h_{j1} + h_{j2})(1 - h_{j1} - h_{j2}) x_k \\
= \; & \sum_i [-(t_i - o_i) w_{ij}^o] h_j (1 - h_j) x_k \\
= \; & \frac{\partial e}{\partial w_{jk}^h} \\
= \; & \Delta w_{jk}^h
\end{aligned}$$

Hence using Algorithm 1, the additive update of hidden layer weights can be correctly computed by the two parties without compromising their data privacy.

5.3.2 Securely computing the piecewise linear sigmoid function

In this subsection we present a secure distributed algorithm for computing piecewise linear approximated sigmoid function $y(x)$ (as shown in Algorithm 2).

Although each party only holds one part of the input to the sigmoid function, this algorithm enables them to compute the approximate value of the function without knowing the part of input from the other party. Actually, in this algorithm there is

no way for each party to explore the input of the other, but the function value can still be computed.

Formally, the input of the algorithm is x_1 held by party A, x_2 held by party B. The output of function y, $y(x_1 + x_2)$, is also randomly shared by the two parties. Note that the parties can always exchange their random shares of result at the end of the algorithm, so that they can learn the complete value of sigmoid function, but since in our chapter the result of sigmoid function is only an intermediate result for the whole learning process, here we keep it randomly shared.

For ease of presentation, we write the algorithm under the assumption that x_1 and x_2 are integers. Note that we can easily rewrite the algorithm to allow real numbers with precision of a few digits after the dot. Actually the algorithm for factorial numbers is the same in essence, given that we can shift the float point to the right to get integer numbers, or in other words, integers and factorials are easy to exchange to each other in binary representation.

After the piecewise linear approximation, as the input is splitted between two parties, both parties do not even know which linear function to use without the knowledge of which range the input falls in. Our main idea is to let one party compute all possible values according to her own input and among those values, the other party picks the result they are looking for.

As shown in Algorithm 2, party A first computes the sigmoid function values for different possible inputs of B. After subtracting a same random number R from each of the function values, A encrypts each of them (step 1). Actually the random number generated by A is the final output for A, the random share of the sigmoid function value. The random share for B is one of the clear texts hidden behind ElGamal scheme. So the remaining task of this algorithm is to let B obtain that clear text, $y(x_1 + x_2) - R$, which corresponds to her input without revealing the input of any party to each other. As A does not know the input of B, she sends all the encrypted

results to B and lets B pick the one based on her own input. Here since all the results are encrypted, B cannot get any information about A's original data either. Because A can get to know the value of x_2 by comparing the encrypted message chosen by B with all those generated by herself, a rerandomization is conducted by B, before B sends back her choice to A to protect B's data, x_2. After A and B sequentially decrypt, B gets her share of the final sigmoid function value (step 3 and step 4).

Algorithm 2 Securely computing the piecewise linear sigmoid function

Step 1: Party A generates a random number R and computes $y(x_1 + i) - R$ for each i, s.t. $-n < i \leq n$. Define $m_i = y(x_1 + i) - R$. Party A encrypts each m_i using ElGamal scheme and gets $E(m_i, r_i)$, where each r_i is a new random number. Party A sends each $E(m_i, r_i)$ in the increasing order of i.
Step 2: Party B picks $E(m_{x_2}, r_{x_2})$. She rerandomizes it and sends $E(m_{x_2}, r')$ back to A, where $r' = r_{x_2} + s$, and s is only known to party B.
Step 3: Party A partially decrypts $E(m_{x_2}, r')$ and sends the partially decrypted message to B.
Step 4: Party B finally decrypts the message (by doing partial decryption on the already partially decrypted message) to get $m_{x_2} = y(x_1 + x_2) - R$. Note R is only known to A and m_{x_2} is only known to B. Furthermore, $m_{x_2} + R = y(x_1 + x_2) = f(x)$.

Note that in Step 1 of Algorithm 2, party A must generate a *new* random number r_i for encrypting each message. Although party B can get $h^{r_{x_2}}$ at the end of the algorithm from $E(m_{x_2}, r_{x_2})$ and the clear message m_{x_2}, he has no way to get other numbers of h^{r_i}. On the other hand, party A can always conduct the partial decryption without knowing which encrypted message B has chosen, because the decryption of ElGamal scheme is independent from random number r_i).

5.3.3 Privacy-preserving distributed algorithm for computing product

To securely compute product, some existing secure multi-party computation protocols for dot product can be utilized (e.g. [100, 47]) by taking integer numbers as a special form of vector input. Here we provide another option for privacy preserving product computation, stated as Algorithm 3. The advantage of Algorithm 3 is that it can be very efficient for some applications, when the finite field of input is small.

Before applying Algorithm 3, a pre-processing step is needed to convert the input of each party into a small integer.

Assume party A holds integer M and party B holds integer N, both M and N are in the same range $-n < M \leq n$, $-n < N \leq n$ (Recall that n is a small integer). Basically it follows a similar idea as of Algorithm 2. After running Algorithm 3, A and B get random numbers respectively which are summed to $M \cdot N$.

Algorithm 3 Securely computing the product of two integers

Step 1: Party A generates a random number R and computes $M \cdot i - R$ for each i, s.t. $-n < i \leq n$. Note $m_i = M \cdot i - R$. A encrypts each m_i using ElGamal scheme and gets $E(m_i, r_i)$, where each r_i is a new random number. Then party A sends each $E(m_i, r_i)$ to party B in the increasing order of i.

Step 2: Party B picks $E(m_N, r_N)$. She rerandomizes it and sends $E(m_N, r')$ back to A, where $r' = r_N + s$, and s is only known to party B.

Step 3: Party A partially decrypts $E(m_N, r')$ and sends the partially decrypted message to B.

Step 4: Party B finally decrypts the message (by doing partial decryption on the already partially decrypted message) to get $m_N = M \cdot N - R$. Note R is only known to A and m_N is only known to B. Furthermore, $m_N + R = M \cdot N$.

5.4 Security analysis

In this section, we explain why our algorithms are secure in the semi-honest model. Recall that in semi-honest model, the parties follow the protocol and may try to analyze what she can see during the protocol execution. To guarantee security in the semi-honest model, we must show that parties can learn nothing beyond their outputs from the information they get throughout the protocol process[4]. A standard

[4]The definition of security we use here is actually standard for a *distributed algorithm* in the semi-honest model (see [49], in which a distributed algorithm is called a "protocol"). Note that we do not use the definition of semantic security, because it is for an *encryption scheme* (see [49], in which an encryption scheme is called a "cryptosystem"), not for a distributed algorithm. In this chapter, we build a distributed learning algorithm and study its security. Hence, we have to use this definition, not the definition of semantic security.

way to show this is to construct a simulator which can simulate what the party can see in the protocol given only the input and output of the protocol for this party.

Since the privacy preserving back-propagation training (Algorithm 1) uses Algorithm 2 and Algorithm 3 as building blocks, we first analyze the security of Algorithm 2 and Algorithm 3, and then we conduct a security analysis of Algorithm 1 in the viewpoint of overall security level.

5.4.1 Security of Algorithm 2 and Algorithm 3

Before we discuss the security of computing piecewise linear sigmoid function (Algorithm 2), recall that the encryption scheme, ElGamal, which we are using here is semantically secure [14] and satisfies the properties discussed in the earlier section. Since ElGamal is semantically secure, each ciphertext can be simulated by a random ciphertext. Hence, we can construct two simulators, for A and B in Algorithm 2, respectively.

In step 1, the simulator for A does what A should do in the protocol. In step 2, the simulator for A generates a random ciphertext to simulate the ciphertext A should receive. In step 3, again the simulator for A does what A should do in the protocol. Since A does not observe anything in step 4, the simulator for A does not need to simulate this step.

In step 1, the simulator for B generates random ciphertexts to simulate the ciphertexts B should receive. In step 2, the simulator for B does what B should do in the protocol. In step 3, the simulator for B generates a random encryption of the output of B in this algorithm to simulate the ciphertext B should receive. In step 4, again the simulator for B does what B should do in the protocol.

The simulators for Algorithm 3 can be constructed in the same way as simulators for Algorithm 2.

5.4.2 Security of Algorithm 1

In Algorithm 1, the termination condition of training is known to both parties, so the simulators can set up the training loop for both parties. In each training round, most of the message transmissions are taking place inside the calls of Algorithm 2 and 3, except in step 3, where B and A respectively receive $\Delta_1 w_{ij}^o, \Delta_2 w_{ij}^o$ and $\Delta_1 w_{jk}^h, \Delta_2 w_{jk}^h$. We will first show that $\Delta_1 w_{ij}^o$ and $\Delta_1 w_{jk}^h$ can be simulated by the simulator for party B, and then likewise $\Delta_2 w_{ij}^o$ and $\Delta_2 w_{jk}^h$ can be simulated for party A.

In Section 2.1, $\Delta_2 w_{ij}^o$ is defined as $(o_{i2} - t_i)h_{j2} + r_{12} + r_{22}$. The variable h_{j2} can be simulated because of the fact that Algorithm 2 is secure (as shown above). The variables r_{12} and r_{22} can also be simulated based on the security of Algorithm 3. Since o_{i2} and t_i are part of the input of party B, $\Delta_2 w_{ij}^o$ can be simulated for B by only looking at its own input. Meanwhile, since weights are output of the training algorithm and η is known to both parties as input, by the weight update rule, $w_{ij}^o \leftarrow w_{ij}^o - \eta(\Delta_1 w_{ij}^o + \Delta_2 w_{ij}^o)$, $\Delta_1 w_{ij}^o$ can be simulated for party B. The simulation of $\Delta_1 w_{jk}^h$ for party B is likewise. Similarly, we can construct the simulator for party A to simulate $\Delta_2 w_{ij}^o$ and $\Delta_2 w_{jk}^h$.

Note that, in the above, the simulators for Algorithm 1 integrates the simulators for Algorithm 2 and 3 when the two component algorithms are called together with the simulations of $\alpha_1, \alpha_2, \beta_1, \beta_2$ in step 3. This completes the construction of simulators for Algorithm 1.

5.5 Analysis of algorithm complexity and accuracy loss

5.5.1 Complexity analysis

In this subsection, we analyze the computation and communication complexity of our privacy preserving back-propagation algorithms. First we present the computation and communication cost in the two component algorithms(i.e., for computing

piecewise linear sigmoid function and product computing) and then use the result to further analyze the running time of Algorithm 1.

5.5.1.1 Securely computing the piecewise linear sigmoid function

In step 1 of Algorithm 2, there are $2 \times n$ encryptions by party A, where n is the parameter in piecewise linear sigmoid function definition. In step 2, 1 rerandomization is conducted. In step 3 and step 4, party A and party B perform one partial decryption respectively. So the total computation cost in Algorithm 2 is $T = (2n + 1)C + 2D$, where C is the cost of encryption and D is the cost of partial description.

Similarly, the total computation cost in Algorithm 3 is also $(2n + 1)C + 2D$.

5.5.1.2 Execution time of one round training

The running time of one round of back-propagation training consists of two parts, time for feedforward stage and for back-propagation stage.

We first consider the execution time of non-privacy preserving back-propagation algorithm. When executing a non-privacy-preserving fully-connected neural network with $a-b-c$ configuration defined earlier in this chapter, one multiplication operation and one addition operation are needed for each connecting weight. Besides, we also need to call activation function one time for each hidden layer node. Therefore the running time for feeding forward is $(ab + bc)S + b \times G$, where S is the cost for one multiplication and one addition, G is the cost for computing activation function, and a, b, c represent the number of units in each layer, respectively.

In step 1 of Algorithm 1, the major difference from the non-privacy-preserving version is the execution of Algorithm2 in (1.2) instead of calling the sigmoid function. In (1.3), one more addition and multiplication is needed for each connection since the value of each hidden layer unit of activation is splitted by the two parties. But since the party A and B can run the algorithm in parallel, this does not increase the

execution time of the whole algorithm. With the time cost for Algorithm 2 is T, we get the running time in feedforward step is $(ab + bc)S + b \times T$.

Now we consider the back-propagation stage. In step (2.1) there are two calls of the Algorithm 3 for each hidden-output connection. Time for one multiplication and three additions is also needed. Because multiplication time is much more significant than addition time, for simplicity, we also use S to denote the time for one multiplication and three additions. In step (2.2), Algorithm 3 are called 4 times, and thus the time for multiplications and additions is $(c+4)S$. The total execution time for back-propagation is $bc(2\times T+S)+ab(4\times T+4\times S+c\times S) = (2bc+4ab)T+(bc+4ab+abc)S$.

Combining the time for the two stages, we obtain the running time of one round privacy preserving back-propagation learning, $(5ab + 2bc + abc)S + (2bc + 4ab + b)T$.

5.5.1.3 Communication overhead

In Algorithm 2 and 3, there are $2n + 2$ messages have been passed between party A and B. With each message being s bits long, the communication overhead is $(2n + 2) \times s$. The total communication overhead of one round learning in Algorithm 1 is the overhead caused by calling algorithm 2 and 3, plus $2 \times s$ in step 3, which is $(b + 2bc + 4ab)(2n + 2)s + 2s$.

5.5.2 Analysis of accuracy loss

There are two places in our algorithms where we introduce approximation for the goal of privacy. One is that the sigmoid function used in the neuron computing is replaced by a piece-wise linear function. The other approximation is introduced by mapping the real numbers to fixed-point representations to enable the cryptographic operations in Algorithm 2 and Algorithm 3. This is necessary in that intermediate results, for example the values of neurons, are represented as real numbers in normal neural network learning, but cryptographic operations are on discrete finite fields. We will empirically evaluate the impact of these two sources of approximation on

the accuracy loss of our neural network learning algorithm in Section 5.6. Below we give a brief theoretical analysis of the accuracy loss caused by the fixed-point representations.

5.5.2.1 Error in truncation

Suppose that the system, in which neural network is implemented, uses μ bits for representation of real numbers. Recall that before applying Algorithm 2 and Algorithm 3, we preprocess the input numbers into finite field that is suitable for crytographic operations. Assume that we truncate the μ-bit numbers by chopping off the lowest ν bits and leaves the new lowest order bit unchanged. The precision error ratio can be bounded by $\epsilon = 2^{\mu-\nu}$.

5.5.2.2 Error in Feeding-forward stage

In the feed forward stage of Algorithm 1, since only Algorithm2 is applied once, the error ratio bound introduced by number conversion for cryptographic operations is ϵ.

5.5.2.3 Error in Output-layer Delta

In Step (2.1) of Algorithm 1, $\Delta_1 w_{ij}^o = (o_{i1} - t_i)h_{j1} + r_{11} + r_{21}$, in which o_{i1} and h_{j1} are already approximated in preceding operations. Therefore, the error ratio bound for $\Delta_1 w_{ij}^o$ is $(1+\epsilon)^2 - 1$. We can obtain the same result for $\Delta_2 w_{ij}^o$.

5.5.2.4 Error in Hidden-layer Delta

In Step (2.2) of Algorithm 1, Algorithm 3 is applied 4 times sequentially. The error ratio bound for $\Delta_1 w_{jk}^h$ and $\Delta_2 w_{jk}^h$ is $(1+\epsilon)^4 - 1$.

5.5.2.5 Error in Weight update

In Step (3) of Algorithm 1, the update of weights introduces no successive error.

5.6 Evaluation

In this section, we perform experiments to measure the accuracy and overheads of our algorithms. We have two sets of experiments on the accuracy. In the first set, we compare the testing error rates in privacy-preserving and non-privacy-preserving cases. In the second set, we distinguish two types of approximation introduced by our algorithms: piece-wise linear approximation of sigmoid function and conversion of real numbers to fixed-point numbers when applying cryptographic algorithms, and analyze how they affect the accuracy of the back-propagation neural networks model. Our experiments on overheads cover the computation and communication costs as well as comparisons with an alternative solution using general purpose secure computation.

5.6.1 Setup

The algorithms are implemented in C++ and compiled with g++ version 3.2.3. The experiments were executed on a Linux (Red Hat 7.1) workstation with dual 1.6 GHz Intel processors and 1 Gb of memory. The programs used GNU Multiple Precision Arithmetic Library in implementation of ElGamal scheme. The results shown below are the average of 100 runs.

The testing datasets are from UCI data set repository [13]. We choose a variety of datasets, kr-vs-kp, Iris, Pima-indian-diabetes (diabetes), Sonar and Landsat with different characteristics, in the number of features, the number of labeled classes, the size of datasets and data distributions. Different neural network models are chosen for varying datasets. Table 5.1 shows the architecture and training parameters used in our neural network model. We choose the number of hidden nodes based on the number of input and output nodes. This choice is based on the criteria of having at least one hidden unit per output, at least one hidden unit for every ten inputs, and five hidden units being a minimum. Weights are initialized as uniformly random values in the range [-0.1, 0.1]. Feature values in each dataset are normalized between

Table 5.1. Datasets and Parameters

Dataset	Sample	Class	Architecture	Epochs	Learning Rate
kr-vs-kp	3196	2	$36-15-1$	20	0.1
Iris	150	3	$4-5-3$	80	0.1
diabetes	768	2	$8-12-1$	40	0.2
Sonar	104	2	$60-6-2$	150	0.1
Landsat	6435	6	$36-3-6$	12	0.1

Table 5.2. Test Error Rates Comparison

Dataset	Non-privacy-preserving Version	Privacy-preserving Algorithm 1
kr-vs-kp	12.5%	15.5%
Iris	14.17%	19.34%
Pima-indian-diabetes	34.71%	38.43%
Sonar	18.26%	21.42%
Landsat	4.12%	5.48%

0 and 1. The privacy preserving back-propagation neural networks have the same parameters as the non-privacy-preserving version. For privacy preserving version, we use the key length of 512 bits.

5.6.2 Accuracy loss

First we measure the accuracy loss of our privacy preserving algorithms when the neural network is trained with a fixed number of epochs (shown in Table 5.1). The number of epochs set is based both on the number of examples and on the parameters (i.e., topology) of the network. Specifically, we use 80 epochs for small problems involving fewer than 250 examples; 40 epochs for the mid-sized problems containing between 250 to 500 examples; and 20 epochs for larger problems.

Table 5.2 shows testing set error rates for both non-privacy-preserving back-propagation neural network and privacy preserving back-propagation neural networks.

Since the numbers of training epochs are fixed, the global error minimum may not be achieved when the training ends. That explains the relatively high error rates for

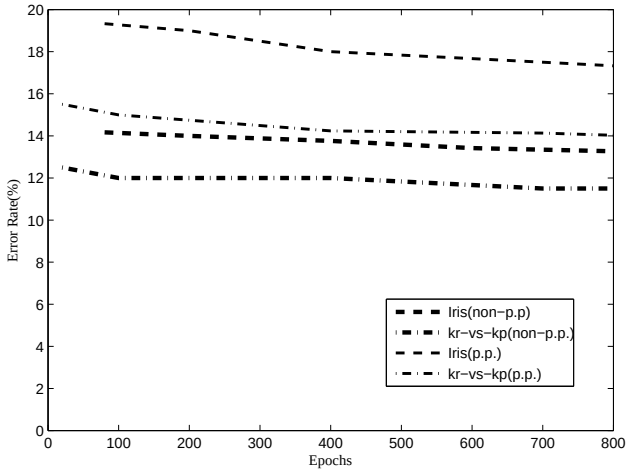


Figure 5.1. Error Rates on Training Epochs

both privacy-preserving training and the non-privacy-preserving case. From Table 5.2, we can see that for experiments on different datasets, the increase of test error rates by privacy-preserving algorithm remain in small range, 1.26% for Landsat to 5.17% for Iris.

We extend the experiments by varying the number of epochs and evaluate the accuracy of privacy preserving back-propagation neural networks on different training epochs. For clarity of presentation, we only show the results of dataset Iris and kr-vs-kp in Figure 1. The result of other datasets have the similar characteristics but with different epochs scales. From Figure 1, we can see clear that the error rates are decreasing when the number of epochs increases. Furthermore, the error rates of privacy preserving back-propagation network decrease faster than the standard algorithm, which means increasing the number of epochs can help to reduce the accuracy loss.

5.6.3 Effects of two types of approximation on accuracy

In this set of experiments, we aim to analyze the causes of accuracy loss in our privacy preserving neural network learning algorithms.

Table 5.3. Error Rates by Different Approximations

Dataset	Non-P.P.	Sigmoid Apx.	Algorithm 1
kr-vs-kp	11.3%	12.2%	13.98%
Iris	14.10%	14.80%	17.04%
Pima-indian-diabetes	33.11%	34.98%	37.28%
Sonar	18.01%	18.85%	21.02%
Landsat	4.10%	4.56%	5.48%

Recall that we have two types of accuracy loss, introduced by sigmoid function approximation and mapping real numbers to fixed point representation respectively. We distinguish and evaluate the effects of these two approximation types by performing a back-propagation learning on approximated piece-wise linear sigmoid function, without cryptographic operations (we call it sigmoid approximation test). This will eliminate the security of the learning, but note that the purpose of this modification is to measure the two kinds of accuracy loss, and thus we have to separate them.

Table 5.3 displays the training error rates and testing error rates comparison of backpropatation learning without privacy concern, versus sigmoid approximation test and privacy preserving learning with two types of approximations. In this set of experiments, we make the training process stop when the error is less than the error tolerance threshold 0.1 and if the the number of epochs reaches the maximum number of 1000 before converging, the training also stops. We call the latter case a failure case of training.

From Table 5.3, we can observe that both of the approximation types cause a certain amount of accuracy loss. The approximation brought by piecewise linear sigmoid function is less significant than by real numbers conversion to fixed-point numbers. For example, in testing of dataset Iris, sigmoid function approximation only contributes 0.70% out of 2.96% in the total accuracy loss while, while conversion of real numbers to fixed-point numbers causes the remaining accuracy loss, which is 2.26%.

Table 5.4. Computation overhead and Communication overhead

Dataset	Our Algorithm		Modified Algorithm	
	Comp.	Comm.	Comp.	Comm.
kr-vs-kp	63.49	39.20	1778.29	443.43
Iris	10.94	6.23	310.62	75.19
diabetes	24.19	14.24	628.32	169.33
Sonar	317.43	197.37	8868.04	2193.13
Landsat	8.14	4.92	211.92	56.98

5.6.4 Computation and Communication overhead

We now examine the computation and communication overhead of our algorithm to verify that it is light-weight and practical. In particular, we measure and record the overall time for privacy computation and the time for communication between party A and B in the entire training process.

Since we are the first to study privacy preserving neural network learning with vertically partitioned data, no other complete algorithm is now available for this problem. Consequently, in order to demonstrate that our algorithm is efficient among possible cryptographic privacy preserving solutions, we consider a modification of Algorithm 1, in which we replace the calls to Algorithms 2 and 3 with executions of Yao's general-purpose two party secure computation protocol [114]. (We utilize the Fairplay secure function evaluation system [75] in our implementation of Yao's protocol.) In the setting described in Section 5.6.1, we compare our original Algorithm 1 with this modified algorithm in terms of computation and communication overheads.

Table 5.4 shows the computation and communication overhead measurements (in minutes) of our algorithm and the modified algorithm. It is clear that, in experiments on the five different datasets, our algorithm is significantly more efficient than the algorithm integrated with Yao's protocol. More precisely, our algorithm is $25.97 - 28.39$ times more efficient in terms of computation overhead, and $11.11 - 12.06$ times more efficient in terms of communication overhead.

Table 5.5. Computation overhead and Communication overhead Comparison with [22] and [10]

Overhead	Our Algoirthm	Protocol in [22]	Protocol in [10]
Computation	63.49	> 1057.10	> 111.65
Communication	39.20	> 219.24	> 71.38

Now we compare the computation and communication overhead of our work with [22] and [10]. As we have mentioned, the objectives of [22] and [10] are different from ours; the authors of [22] and [10] have not implemented their protocols; neither have they performed any experiment. Consequently, in order to compare their overhead with ours, we have to implement these protocols by ourselves and measure their overheads.

Nevertheless, for practical purposes, we do not implement and measure the complete protocols of [22] and [10]. The reason is that these protocols are so slow in practice that even *parts of them* have significantly more overheads than ours. So measuring the overheads of such parts is sufficient to demonstrate the better efficiency of our algorithm. Furthermore, if we measure the overheads of the complete protocols, [22] and [10] (especially [22]) will take more time than we can afford.

Hence, we have implemented a key compnent of both [22] and [10], namely secure evaluation of the activation function. When we measure the overheads of [22] and [10], we count the accumulative overhead of secure activation function evaluations only, and ignore the overhead of all other operations. We compare the measured partial overhead of [22] and [10] with the total overhead of our own algorithm, which includes the computation and communication overheads of *all* components. Table 5.5 shows the comparison results when we train the neural network using dataset kr-vs-kp. We put the symbol ">" before the measured partial overheads of [22] and [10], to emphasize that these are not the total overheads. As we can see, for both [22] and [10], the measured partial overheads are already significantly more than the total overhead of our protocol, either in terms of computation or in terms of

communication. Therefore, we can safely claim that our algorithm is more efficient than the protocols in [22] and [10].

5.7 Summary

In this chapter, we present a privacy preserving algorithm for back-propagation neural network learning. The algorithm guarantees privacy in the semi-honest model. Although approximations are introduced in the algorithm, the experiments on real world data show that the amount of accuracy loss is reasonable.

Using our techniques, it should not be difficult to develop the privacy preserving algorithms for back-propagation network learning with three or more participants. In this chapter we have considered only the back-propagation neural network. A future research topic is to extend our work to other types of neural networks training.

CHAPTER 6

PRIVACY PRESERVING GROWING NEURAL GAS LEARNING[25]

6.1 Background and Motivation

Growing neural gas [44] is a well known algorithm in evolutionary computing. This algorithm applies natural selections to neural network training and has been shown to be very effective. It has found broad applications in fields like data clustering and visualization.

In many practical scenarios, the training datasets used to grow neural gas are distributed between two or more participants. For example, consider multiple medical researchers, who collaborate on a research project that aims to build a neural network to predict the risk of a certain disease. Suppose that they are using the growing neural gas algorithm, and that the training data used in this project is the medical profiles of patients collected by these researchers. Each of these researchers has her own set of medical profiles that can be used in the research. Clearly, if the neural network can be trained using the data from all the researchers, the research project will achieve the greatest success. Nevertheless, none of the researchers is willing to reveal her own data to others.

Similarly, consider two local supermarkets that try to predict the patterns of the purchases of their customers. Suppose that they apply the growing neural gas algorithm to transaction records, in order to build a neural network for this purpose. Again, these supermarkets will benefit from collaborating with each other and training the neural network using the union of their transaction records, but neither of them is willing to reveal its own transaction records to the other supermarket.

In general, when the training dataset is distributed between two or more parties, the growing neural gas algorithm usually benefits from using the data from all involved parties. However, the involved parties may not be willing to reveal their own data to other parties, due to their own privacy concerns, or due to legislations such as HIPAA [53].

The target of this chapter is to design algorithms that protect privacy of participants in the above scenarios. That is, we present an algorithm that allow two parties to jointly grow neural gas using all the available training data. Our algorithms are *privacy preserving* in the sense that no participant needs to reveal her own data to the other party. Consequently, the participants can enjoy the benefits of collaboration without worrying about privacy of their data.

In particular, we distinguish two cases: *horizontally partitioned dataset* and *vertically partitioned dataset*. In the former case, each participant has a training dataset and our objective is to grow neural gas using the union of these datasets. Formally, all the participating parties in the growing neural gas algorithm, switches back and forth the network (between parties having training data) after each sample, without revealing anything to other parties about their sample. The supermarket example(above) constitutes horizontal partitioned dataset, where each grocery store has customer data and wants to use this data for growing neural gas algorithm.

In the latter case, each participant has a set of the attributes that can be used in the training, while the attributes of different participants belong to the same dataset. Our objective is to grow neural gas using all these attributes. Formally, in this type of partitioning for each sample, all the participating parties (for growing neural gas algorithm) use our proposed algorithm (described below) to train the growing neural gas network. Predicting credit card history of people using data from different banks is an example of training growing neural gas network with vertically partitioned data.

For the scenario of horizontally partitioning, since each party holds a subset of data samples with the same attribute set, there is a very simple solution for protecting privacy in growing neural gas algorithm. Basically one party trains the network using its data and passes on the network structure to the other party so that it can further train it with its own data, without revealing the data of any party. Therefore, in this chapter, we focus on the scenario of vertically partitioning. We propose a privacy preserving algorithm that provides privacy guarantee in growing neural gas. The main tools we use are novel cryptographic techniques that can add strong privacy protection to distributed algorithms. To evaluate our algorithms, we have presented a detailed analysis of correctness and privacy. Furthermore, we conduct extensive experiments using real world datasets. The results of our evaluations show that our algorithm is correct, secure and efficient.

Our contributions can be summarized as follows.

- We are the *first* to study privacy protection in growing neural gas and propose a privacy preserving algorithm.
- To the best of our knowledge, we are also the first to study privacy protection in evolutionary computing.
- Our algorithm for protecting privacy in growing neural gas is quite efficient in terms of computational and communication time.
- In terms of privacy, our algorithm leaks no knowledge about each party's data except the network obtained through growing neural gas.

The rest of the chapter is organized as follows. In section 6.2, we describe the technical preliminaries including the definitions, notations, security model that we use in this chapter. In section 6.3, we propose our privacy preserving algorithm for growing gas neural network when the data is vertically partitioned between two

parties. In section 6.5 we perform the experiments on the real world data. In the end in section 6.6, we summarize this chapter.

6.2 Technical Preliminaries

In this section we present the notations, problem definition and the security model used in this chapter.

6.2.1 Notations for Growing Neural Gas Algorithm

The growing neural gas algorithm creates a network of nodes incrementally, based on the input vectors of dimension m from the training dataset. Each node k in the network has a position. We use a m-dimensional vector $\overrightarrow{w}_k$ to denote both node k's position and the reference vector of node k. We take the squared distance between a node k and an input data as the so called error. In this chapter we use Euclidean distance. Each node k has a variable, E_k to store the local accumulated error. The set C is the set of edges that define neighbor nodes in the network. The set N contains all the nodes present in the network. Since in the growing neural gas algorithm nodes are added incrementally, set N is growing until certain user-defined criteria is met.

6.2.2 Definitions and Problem Statement

Vertically Partitioned Data We consider vertically partitioning of data between two parties in the chapter. In the scenario of vertically partitioned data, each Party has certain number of attributes for each sample in the dataset.

In particular, suppose that a training dataset D consists of n samples $\{DB_1, DB_2, \cdots, DB_n\}$, and each sample DB_i $(1 \leq i \leq n)$ contains m attributes. For each sample DB_i, Party A holds first m_A $(0 < m_A < m)$ attributes and Party B holds the rest m_B attributes such that $m_A + m_B = m$. So suppose $\{x_1, x_2, \cdots, x_m\}$ are the m attributes of any sample $\overrightarrow{x}$ in the dataset. Party A holds $\{x_1, x_2, \cdots, x_{m_A}\}$ and Party B holds $\{x_{m_A+1}, x_{m_A+2}, \cdots, x_{m_A+m_B}\}$.

Semi-honest model As many privacy preserving data mining protocols and also in Chapter 5, we use the semi-honest security model [49].

Privacy Preserving Growing Neural Gas over Vertically Partitioned Data As described above, we consider the scenario that the input data is vertically partitioned among two parties. Privacy preserving growing neural gas is that in the process of incrementally creating a network of nodes, each party does not reveal its own data to the other party. The topological information excluding the exact positions of each node is known to both parties, such as the total number of nodes in the network and the set of edges C. In the privacy preserving growing neural gas algorithm, each party only holds its own share of the reference vector for each node k, $\overrightarrow{w}_{kA}$ and $\overrightarrow{w}_{kB}$ for party A and party B respectively. Similarly, party A (resp. party B) has its share of the local accumulative error for k, E_{kA} (resp. E_{kB}). In every iteration of the algorithm, each party updates the nodes set N, the edges set C, and their own shares of nodes' positions and accumulative errors, without compromising their privacy of the input data. Moreover, the combination of network information that the two parties holds produces the same results as the network is directly generated from the join of data sets without privacy concerns.

6.2.3 Yao's protocol

In our privacy preserving growing neural gas algorithm, a general secure two-party computation protocol, Yao's protocol is utilized as an important component. Now we briefly introduce the Yao's protocol. For further details, please refer to [114]. There are two parties, each of whom holds part of the input to a computation function. For the computation function that the two parties want to conduct, a Boolean circuit C made of wires and gates is constructed. Then the two parties interact in order to evaluate C securely. The basic idea of Yaos protocol is to compute a circuit so that values obtained on all wires except circuit-output wires will not be revealed. In

order to do this, one party construct the circuit, converts it into a garbled circuit and transfers to the other party. Then the two parties execute an oblivious transfer per each input wire, so that the receiver gets the input-wire keys. Using these keys, the two parties compute the circuit and obtain the output. In this chapter, we use Yao's protocol to solve the secure comparison problem. In the rest of the chapter, we use Algorithm SecureCompare() to represent Yao's secure two-party computation solution for the secure comparison problem.

6.3 Privacy Preserving Growing Neural Gas

In this section we present our privacy preserving growing neural gas algorithm when the input data is vertically partitioned between two parties. We first give an overview of our privacy preserving algorithm for growing neural gas. Then we focus on one key component of our algorithm, i.e., privacy preserving search for the minimum(maximum) sum and describe our solution for this component.

6.3.1 Privacy Preserving Growing Neural Gas Algorithm

Algorithm 1 presents our privacy preserving growing neural gas algorithm. For clarity, we separate the routine for inserting a new node into the network, from the main algorithm and describe it as in Algorithm 2.

In the network initialization stage, the two parties jointly generate two randomly-positioned nodes in the network. In particular, party A generates the first m_A components of the reference vector for each of the two nodes, while party B generates the last m_B components. They keep the node vectors that they generate private. For each node, each party holds a local error variable. They set the local error variables for newly-generated two nodes to 0.

For each iteration, the input to the growing neural gas algorithm with vertically-partitioned data is $(\overrightarrow{x}_A, \overrightarrow{x}_B)$, where $\overrightarrow{x}_A$ is held by party A and $\overrightarrow{x}_B$ is held by party

Algorithm 4 Privacy preserving Growing Neural Gas algorithm with vertically distributed datasets

Initialize : Create two randomly-positioned nodes in the network $N = \{c, d\}$. Party A holds $\overrightarrow{w}_{cA}$ (resp. $\overrightarrow{w}_{dA}$), where $\overrightarrow{w}_{cA}$ (resp. $\overrightarrow{w}_{dA}$) is the first m_A components of $\overrightarrow{w}_c$. Party B holds $\overrightarrow{w}_{cB}$ (resp. $\overrightarrow{w}_{dB}$), where $\overrightarrow{w}_{cB}$ (resp. $\overrightarrow{w}_{dB}$) is the last m_B components of $\overrightarrow{w}_d$.
Set $C = \phi$. N and C are known to both parties.
Party A sets $E_{cA} = 0$; $E_{dA} = 0$. Party B sets $E_{cB} = 0$; $E_{dB} = 0$
for each input data $< \overrightarrow{x}_A,\ \overrightarrow{x}_B >$ **do**

1. For each node k in the network, party A computes $D_{kA} = \|\overrightarrow{w}_{kA} - \overrightarrow{x}_A\|^2$ and party B computes $D_{kB} = \|\overrightarrow{w}_{kB} - \overrightarrow{x}_B\|^2$. Party A obtains $\{D_{1A}, D_{2A}, \cdots, D_{|N|A}\}$ and party B gets $\{D_{1B}, D_{2B}, \cdots, D_{|N|B}\}$.

2. By applying twice Algorithm 3, party A and party B find the first and the second nearest nodes for $\overrightarrow{x}$ are s and t, out of all the nodes in the network, i.e., $s = \mathsf{argmin}_{\mathsf{k}\in\mathsf{N}}(\|\overrightarrow{w}_{kA} - \overrightarrow{x}_A\|^2 + \|\overrightarrow{w}_{kB} - \overrightarrow{x}_B\|^2)$. $t = \mathsf{argmin}_{\mathsf{k}\in\mathsf{N}-\mathsf{s}}(\|\overrightarrow{w}_{kA} - \overrightarrow{x}_A\|^2 + \|\overrightarrow{w}_{kB} - \overrightarrow{x}_B\|^2)$.

3. At both parties, an edge between s and t, $(s,\ t)$ is created if not existed. $C = C \cup \{(s, t)\}$. Both parties set the age of this edge to 0: $age_{s,t} = 0$.

4. Party A updates the local error variable for node s: $E_{sA} = E_{sA} + \|\overrightarrow{w}_{kA} - \overrightarrow{x}_A\|^2$.

5. Party B updates the local error variable for node s: $E_{sB} = E_{sB} + \|\overrightarrow{w}_{kB} - \overrightarrow{x}_B\|^2$.

6. Party A updates $\overrightarrow{w}_{sA}$: $\overrightarrow{w}_{sA} = \overrightarrow{w}_{sA} + e_s(\overrightarrow{x}_A - \overrightarrow{w}_{sA})$. Party A also updates $\overrightarrow{w}_{iA}$ for each neighbor node i of s: $\overrightarrow{w}_{iA} = \overrightarrow{w}_{iA} + e_i(\overrightarrow{x}_A - \overrightarrow{w}_{iA})$, where $e_s, e_i \in [0, 1]$.

7. Party B updates $\overrightarrow{w}_{sB}$: $\overrightarrow{w}_{sB} = \overrightarrow{w}_{sB} + e_s(\overrightarrow{x}_B - \overrightarrow{w}_{sB})$. Party B also updates $\overrightarrow{w}_{iB}$ for each neighbor node i of s: $\overrightarrow{w}_{iB} = \overrightarrow{w}_{iB} + e_i(\overrightarrow{x}_B - \overrightarrow{w}_{iB})$.

8. Increase ages of all the edges connecting s: $age_{(s,i)} = age_{(s,i)} + 1$. where node i is the neighbor of s.

9. Remove edges with an age larger than a_{max}. Also remove the nodes if this results in nodes having no edges.

10. If the current iteration is an integer multiple of λ, use Algorithm 2 to insert a new node into the network.

11. For each node k, both parties decrease the local error values $E_{kA} = E_{kA} - \beta E_{kA}$; $E_{kB} = E_{kB} - \beta E_{kB}$.

end for
Stop when stopping criteria is met.

B. In each iteration, as in the non-privacy-preserving growing neural gas algorithm, certain changes are made to the network topology, such as changing nodes' reference vectors (i.e., moving their positions), generating new nodes and creating/deleting edges. The major difference here is that each variable in the non-privacy preserving algorithm is split into two parts, with each party holding one part. The main idea of our algorithm lies in that each party uses its own data attributes to update the partial network information that it maintains, as much as it could except some operations relying on the information from both parties. For such operations, we provide a secure solution which involves some data communications between the two parties but still keeps each party's own data private. In particular, one important key component in our algorithm is to find the winning node with the minimum distance to the input vector in the network. Since the winning node is computed based on the distances defined using the attributes, it requires the information from both parties. In Section 6.3.2, we will present the solution for this problem. In Algorithm 1 and Algorithm 2, we just use this solution as a building block.

In Algorithm 1, in all steps of each iteration except step 3), both parties compute locally to update the network information without communicating with each other. In step 3), the two parties are applying Algorithm 3 to securely find the first and second winning nodes in the network without revealing its own data.

In Algorithm 2, in step 1) and step 2), party A and party B use Algorithm 3 to find the node with the maximum accumulated local error and to find its neighboring node with maximum local error. In other steps, each party locally computes the partial reference vector and local error that it holds for the newly-inserted node, and locally updates the local error for the two nodes found in step 1) and step 2).

Algorithm 5 Privacy preserving Growing Neural Gas algorithm - Adding a new node in the network

INPUT: Set N and C are known by both parties. For each node in the network $k \in N$, party A holds $\overrightarrow{w}_{kA}$ and E_{kA}; party B holds $\overrightarrow{w}_{kB}$ and E_{kB}, s.t. $E_k = E_{kA} + E_{kB}$.

1. Party A and party B use Algorithm 3 to find the node p with the maximum accumulated error, i.e., $p = \mathsf{argmax}_{\mathsf{k}\in\mathsf{N}} E_k$.

2. Using Algorithm 3 again, parties A and B find out the neighbor node of p with the maximum accumulated error. We call this neighbor node q.

3. Insert a new node r in the network at mid way between p and q. Party A computes $\overrightarrow{w}_{rA} = (\overrightarrow{w}_{pA} + \overrightarrow{w}_{qA})/2$. Party B computes $\overrightarrow{w}_{rB} = (\overrightarrow{w}_{pB} + \overrightarrow{w}_{qB})/2$. Add this new node into set N s.t. $N = N \cup \{r\}$.

4. Insert edges between p and r and between r and q. Remove the original edge between s_3 and s_4. Update set C = C $\cup\{(p, r), (r, q)\}$ and $C = C - \{p, q\}$.

5. Party A (resp. party B) decreases the shares of local error values for p and q: $E_{pA} = E_{pA} - \alpha E_{pA}$ (resp. $E_{pB} = E_{pB} - \alpha E_{pB}$), and $E_{qA} = E_{qA} - \alpha E_{qA}$ (resp. $E_{qB} = E_{qB} - \alpha E_{qB}$).

6. Party A (resp. party B) sets the local error value for r: $E_{rA} = (E_{pA} + E_{qA})/2$. (resp. $E_{rB} = (E_{pB} + E_{qB})/2$).

6.3.2 Privacy Preserving Search for the Minimum(Maximum) Sum

In this subsection, we introduce our solution for privacy preserving search for the minimum or maximum sum. We assume that each party holds a sequence of numbers. We are interested in the sum of each pair of same-positioned numbers in the two sequences. The goal is to find which sum is the maximum or minimum without revealing any party's data to each other. Our solution is described in Algorithm 3. In Algorithm 3, we use the idea of the first round of bubble sort to find the maximum or minimum sum. In the comparison between any two sums, e.g., $d_{(i+1)A} + d_{(i+1)B}$ and $d_{jA} + d_{jB}$, party A holds $d_{(i+1)A}$, d_{jA} and party B holds $d_{(i+1)B}$, d_{jB}. In order to hide the data from each other, we let each party add a random number to the sum. In this way, the comparison result is the same and privacy for each party is preserved.

Algorithm 6 Privacy Preserving Search for the Minimum(Maximum) Sum

1: **INPUT**: Party A holds $\{d_{1A}, d_{2A}, \cdots, d_{nA}\}$. Party B holds $\{d_{1B}, d_{2B}, \cdots, d_{nB}\}$.
2: **OUTPUT**: $\mathsf{arg_i min}(d_{iA} + d_{iB})$
($\mathsf{arg_i max}(d_{iA} + d_{iB})$ if searching for maximum sum.)
3: Both parties set $j = 1$.
4: **for** $(i = 1; i < n; i++)$ **do**
5: Party A generates a new random number R_A, and sends $d_{jA} + R_A$ to party B.
6: Party B generates a new random number R_B, and sends $d_{(i+1)B} + R_B$ to party A.
7: Party A calculates $d_{(i+1)A} + R_A + d_{(i+1)B} + R_B$.
8: Party B calculates $d_{jA} + R_A + d_{jB} + R_B$.
9: Party A and Party B securely compare $d_{(i+1)A} + R_A + d_{(i+1)B} + R_B$ and $d_{jA} + R_A + d_{jB} + R_B$ using $\mathsf{SecureCompare}()$ algorithm.
10: Both parties Store the index of the smaller (larger if searching for maximum sum) number in j.
11: **end for**
12: Return j.

6.4 Correctness and Security Analysis

So far, we have completely described our privacy preserving growing neural gas algorithm. In this section, we conduct detailed analysis on the correctness and security of our algorithm.

6.4.1 Correctness Analysis

After applying our privacy preserving growing neural gas algorithm, party A and party B can combine their partial network information to get the final complete network. Now we show that this combined result is correct, i.e., equal to the network obtained when training the network with data stored at one site.

To conduct the analysis, we focus on the possible update to the network information when each sample is input into the network. More precisely, we show that in each iteration of training, the changes of the network information are the same, when using our algorithm and when applying growing neural gas algorithm at one site without privacy concerns. By showing correctness of each iteration, we can easily arrive at the conclusion that the result that our entire privacy preserving growing neural gas algorithm obtains is correct.

We notice that the possible changes of the network information in each iteration of training include the change of network topologies, ΔN, ΔC, the nodes' vectors $\Delta\overrightarrow{w}_k$, nodes' local error variables ΔE_k and nodes' ages. The following analytical result shows that in each iteration of training, our privacy preserving algorithm yields the correct ΔN, ΔC, $\Delta\overrightarrow{w}_k$, ΔE_k and nodes' ages.

Theorem 13. *In an iteration of growing neural gas, our Algorithm 1 (with the input $\overrightarrow{x}_A$ for party A and $\overrightarrow{x}_B$ for party B) produces the changes to the network which is same as the results of running growing neural gas algorithm with input $\overrightarrow{x} = (\overrightarrow{x}_A, \overrightarrow{x}_B)$.*

Proof. We first show the correctness of Algorithm 3. Then we show that in each iteration, ΔN and ΔC are the same as in the algorithm without privacy concerns. Finally we will show that $\Delta\overrightarrow{w}_k$, ΔE_k and nodes' ages are also correct.

In Algorithm 3, in each iteration of the loop (Line 4 to 11), party A correctly compare two numbers $d_{(i+1)A} + R_A + d_{(i+1)B} + R_B$ and $d_{jA} + R_A + d_{jB} + R_B$, since the correctness of SecureCompare() algorithm is guaranteed. It is easy to see that the comparison result is equal to the comparison result between $d_{(i+1)A} + d_{(i+1)B}$ and $d_{jA} + d_{jB}$. Therefore, based on the correctness of comparison between any two sums, the correctness of Algorithm 3 is proved.

Now we show that in each iteration, our algorithm produces correct ΔN and ΔC. We note that the steps that updates the N and C are steps 2) 3) and 9) in Algorithm 1 and steps 1)-4) of Algorithm 2. Based on correctness of Algorithm 3 that we have proved, it is not difficult to see that the above steps are correct.

For $\Delta\overrightarrow{w}_k$, we can see that $\forall k \in N$, $\Delta\overrightarrow{w}_k = (\Delta\overrightarrow{w}_{kA}, \Delta\overrightarrow{w}_{kB})$. Since we have proved that in each iteration, the correct winner and second winner will be chosen, the updates of $\overrightarrow{w}_k$ in steps 6) 7) of Algorithm 1 and step 3) of Algorithm 3 are correct.

For ΔE_k, since we are using Euclidean distance, we have

$$\|\overrightarrow{w}_k - x\|^2 = \|\overrightarrow{w}_{kA} - \overrightarrow{x}_A\|^2 + \|\overrightarrow{w}_{kB} - \overrightarrow{x}_B\|^2 .$$

Hence $\Delta E_k = \Delta E_{kA} + \Delta E_{kB}$. The correctness of nodes' ages is also clear.

This completes the proof of Theorem 1.

□

6.4.2 Security Analysis

In this subsection, we explain why our algorithm is secure in the semi-honest model. Recall that in the semi-honest model, we say an algorithm is secure if each party can learn nothing beyond its output from the information obtained throughput

the algorithm. A standard way to conduct security analysis is to construct a simulator for each party that can simulate what it can see in the algorithm given only the input and output for this party in the algorithm.

We notice that in Algorithm 1 and Algorithm 2, except step 2) in Algorithm 1 and steps 1) and 2) in Algorithm 2, both parties do not receive any information from each other. In other words, except when running Algorithm 3, there is no way for both parties to learn any new information from what they obtain. Therefore, we mainly focus on constructing simulators for Algorithm 3 in our security analysis. After we show the security of Algorithm 3, we can conclude that our privacy preserving growing neural gas algorithm is secure in the semi-honest model.

For Algorithm 3, we construct two simulators, one for each party. The simulators work as follows. For line 3-5, the simulators do what each party should do in the algorithm. In line 5, the simulator for party B generates a random number to simulate $d_{jA} + R_A$. In line 6, the simulator for party A generates a random number to simulate $d_{(i+1)B} + R_B$. In line 7 (resp. line 8), the simulator for party A (party B) does what it should do in the algorithm. Since we know that algorithm SecureCompare is secure in the semi-honest model, we can construct two simulators for party A and party B to simulate SecureCompare. In line 9, the simulators for our algorithm just follow what the simulators for SecureCompare do. In line 10, the simulators do what each party should do in the algorithm.

6.5 Experiments

In this section we perform experiments to evaluate the efficiency of our privacy preserving growing neural gas algorithm. The experiments are carried out using the datasets from the UCI dataset repository [13].

Table 6.1. Datasets and Parameters

Dataset	Samples	Attributes	λ	max.network.size
Iris	150	4	100	10
Lung cancer	32	56	30	10
Zoo	101	17	20	20
Vowel Recognition	528	10	55	30
Ecoli	336	8	50	25

6.5.1 Set Up

We have used C++ programming language to perform our experiments with g++ version 2.8, experiments being carried out on Redhat Linux with memory of 256MB. We utilize the Fairplay secure function evaluation system [75] in our implementation of SecureCompare.

Table 1 shows the parameters used in the experiment. Specifically we have used Iris, Lung cancer, Zoo, Vowel Recognition and Ecoli datasets for our experiments from the UCI dataset repository. These datasets are vertically partitioned between two parties. The value of λ, used when to add a new node in the network, is varied between 20 and 100, depending upon the number of samples in the dataset. The maximum network size is set based on the total clusters in the dataset. Except for the parameters shown in Table 1, the other parameters used in the experiment are $\alpha = 0.5$, $\beta = 0.005$, $a_{max} = 10$, $\epsilon_b = 0.2$ and $\epsilon_i = 0.05$.

6.5.2 Experimental Results

We perform experiments to evaluate the efficiency of our algorithm which protects the privacy of the individuals, using a cryptographic component.

Table 2 shows the total running time taken by our algorithm on the five datasets respectively. We can see that the total running time for the privacy preserving growing neural gas algorithm varied from about 100 seconds to more than 10 minutes. We note that the total running time depends on the number of input samples as well as the stopping criteria. Overall, our algorithm is very efficient.

Table 6.2. Total Running Time

Dataset	Total Running Time (sec.)
Iris	519.41
Lung cancer	103.95
Zoo	452.82
Vowel Recognition	1737.48
Ecoli	1081.30

Table 6.3. Communication Overhead & Computation Overhead

Dataset	Communication Time	Computation Time
Iris	105.89	413.52
Lung cancer	22.59	81.36
Zoo	92.35	360.47
Vowel Recognition	337.79	1399.69
Ecoli	247.76	833.54

Now we further measure the time used for computation and the time used for communication between the two parties, in the total running time. Specifically, communication occurs in Algorithm 3 when the parties sending random number to each other, and also occurs when running SecureCompare.

Table 3 shows the results of communication time and computation time when running our privacy preserving growing neural gas algorithm on the five datasets. From the results, we can see that the algorithm takes more time in computing than communication between the two parties.

6.6 Summary

In this chapter we proposed a privacy preserving growing neural gas algorithm when the input data is vertically partitioned between two parties. We show that our algorithm is correct and provides strong privacy guarantees. We also demonstrate that our algorithm is very efficient using extensive experiments.

CONCLUSION

In this thesis, studies have been done on the incentive issues and the privacy issues in the distributed computing environments with untrusted parties. For incentive problems, work has been focused on one important and popular application field, wireless networks, while for privacy issues, this thesis focuses on designing privacy preserving distributed data mining protocols due to their wide impact in some critical applications such as in medical systems.

Part II is devoted to the works on incentive issues for packet forwarding in wireless networks. In particular, Chapter 2 studies the reputation systems in *finite* repeated game for traditional wireless ad hoc networks in order to provide nodes incentives to forward packets. It is shown that it is impossible of building a SPNE solution using traditional reputation systems. Then the TTI technique is introduced and used to build FITS, the new reputation system. FITS provide strong incentive compatibility for nodes to cooperate in packet forwarding. More precisely, there is a SPNE in which nodes forwards all packets. This is proved theoretically and verified by experiments.

In Chapter 3, INPAC is proposed, the first incentive scheme for packet forwarding in wireless mesh networks using network coding. It is complementary to the existing work on incentive compatible routing in the same type of wireless networks. Since packet forwarding is a fundamental procedure for computer networks, INPAC is of great importance to the application of network coding technology in environments with selfish users. INPAC has extensively evaluated on the Orbit Lab testbed, and the results demonstrate that INPAC is both efficient and incentive compatible.

In Chapter 4, the first simple and effective incentive scheme in VANETs with theoretical guarantee is proposed. It is formally proved that, in a coalitional game

model, that with our scheme every relevant node cooperates in forwarding messages as required by the routing protocol. The incentive scheme is integrated with MV and Spray-and-Wait respectively and the system performance is evaluated on testbed traces. The experimental results show that this incentive scheme provides effective stimulation for nodes to cooperate and prevents the degradation of system performance in VANETs with selfish nodes.

In Part III, two privacy preserving distributed data mining protocols are presented. In Chapter 5, a privacy preserving algorithm for back-propagation neural network learning is provided. The algorithm guarantees privacy in the semi-honest model. Both analytical and experimental results show that this algorithms are light-weight in terms of computation and communication overheads. In Chapter 6, a privacy preserving growing neural gas algorithm when the input data is vertically partitioned between two parties is proposed. It is shown that this algorithm is correct and provides strong privacy guarantees.

This thesis opens up my research on privacy and incentive issues in the advanced distributed computing environments. In the future, I will continue to identify and solve new privacy, incentive and security problems in the emerging and critical applications.

BIBLIOGRAPHY

[1] S. S. A. Conti, D. Panchenko and V. Tralli. Log-concavity property of the error probability with application to local bounds for wireless communications. *IEEE Transactions on Information Theory*, 55(6):2766–2775, 2009.

[2] E. Adar and B. A. Huberman. Free riding on gnutella. *First Monday*, 5(10), 2000.

[3] D. Agrawal, R. Srikant, and D. Thomas. Privacy preserving olap. In *In Proceedings of ACM SIGMOD International Conference on Management of Data*, 2005.

[4] R. Agrawal and R. Srikand. Privacy preserving data mining. In *Proceedings of ACM SIGMOD Conference on Management of Data*, May 2000.

[5] S. Akl and P. Taylor. Cryptographic solution to a problem of access control in a hierarchy. *ACM Trans. Computer Systems*, 1(3):239–248, 1983.

[6] I. F. Akyildiz and X. Wang. A survey on wireless mesh networks. *IEEE Communications Magazine*, 43(9), 2005.

[7] L. Anderegg and S. Eidenbenz. Ad hoc-VCG: a truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents. In *Proceedings of ACM MOBICOM*, 2003.

[8] A. Balasubramanianm, B. N. Levine, and A. Venkataramani. Dtn routing as a resource allocation problem. In *Proceedings of ACM SIGCOMM '07*, Kyoto, Japan, Aug. 2007.

[9] S. Bansal and M. Baker. Observation-based cooperation enforcement in ad hoc networks. In *Technical report, Stanford University, Stanford, CA*, 2003.

[10] M. Barni, C. Orlandi, and A. Piva. A privacy-preserving protocol for neural-network-based computation. In *PProceeding of the 8th workshop on Multimedia and security*, pages 146–151, 2006.

[11] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of ACM CCS 1993*, pages 62–73, 1993.

[12] C. Bettstetter and C. Hartmann. Connectivity of wireless multihop networks in a shadow fading environment. In *Proceedings of the 6th International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems*, pages 28–32, San Diego, CA, 2003. ACM Press.

[13] C. Blake and C. Merz. Uci repository of machine learning databases. Department of Information and Computer Science, 1998.

[14] D. Boneh. The decision diffie-hellman problem. *ANTS 1998*, pages 48–63, 1998.

[15] S. Buchegger and J.-Y. Le Boudec. Nodes bearing grudges: Towards routing security, fairness, and robustness in mobile ad hoc networks. In *10th Euromicro Workshop on Parallel, Distributed and Network-based Processing*, 2002.

[16] S. Buchegger and J.-Y. Le Boudec. Performance analysis of the CONFIDANT protocol: Cooperation of nodes - fairness in dynamic ad-hoc networks. In *Proceedings of the Third ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Lausanne, Switzerland, June 2002.

[17] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine. Maxprop: Routing for vehicle-based disruption- tolerant networks. In *Proceedings of IEEE INFOCOM '06*, Apr. 2006.

[18] B. Burns, O. Brock, and B. N. Levine. Mv routing and capacity building in disruption tolerant networks. In *Proceedings of IEEE INFOCOM '05*, Miami, FL, Apr. 2005.

[19] L. Buttyan and J. P. Hubaux. Enforcing service availability in mobile ad-hoc WANs. In *Proceedings of ACM MOBIHOC*, 2000.

[20] L. Buttyan and J. P. Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *ACM Journal for Mobile Networks (MONET), special issue on Mobile Ad Hoc Networks*, summer 2002.

[21] S. Chachulski, M. Jennings, S. Katti, and D. Katabi. Trading structure for randomness in wireless opportunistic routing. In *Proceedings of ACM SIGCOMM*, 2007.

[22] Y. C. Chang and C. J. Lu. Oblivious polynomial evaluation and oblivious neural learning. In *Proceedings of Asiacrypt*, pages 369–384, 2001.

[23] K. Chen and L. Liu. Privacy preserving data classification with rotation perturbation. In *Proceeding of ICDM'05*, pages 589–592, 2005.

[24] S. Chen, P. Lin, D. Huang, and S. R. Yang. A study on distributed/centralized scheduling for wireless mesh network. In *Proceedings of the 2006 International Conference on Wireless Communications and Mobile Computing*, pages 599–604, 2006.

[25] T. Chen, A. Bansal, S. Zhong, and X. Chen. Protecting data privacy in growing neural gas. *Neural Computing & Applications*, 2011.

[26] T. Chen, F. Wu, and S. Zhong. Fits: A finite-time reputation system for cooperation in wireless ad-hoc networks. *IEEE Transactions on Computers*, 2010.

[27] T. Chen and S. Zhong. Privacy preserving backpropagation neural network learning. *IEEE Transactions on Neural Networks*, 20(10):1554–1564, 2009.

[28] T. Chen and S. Zhong. Inpac: An enforceable incentive scheme for wireless networks using network coding. In *Proceedings of IEEE INFOCOM'10*, 2010.

[29] T. Chen, L. Zhu, F. Wu, and S. Zhong. Stimulating cooperation in vehicular ad hoc networks: A coalitional game theoretic approach. *IEEE Transactions on Vehicular Technology*, Feb. 2011.

[30] M. Chicurel. Databasing the brain. *Nature*, pages 822–825, 2000.

[31] F. C. Commissions. Fcc 222(c)(1).

[32] V. Conitzer and T. Sandholm. Self-interested automated mechanism design and implications for optimal combinatorial auctions. In *Proceedings Fifth ACM Conference on Electronic commerce (EC-04)*, pages 132–141, 2004.

[33] A. Conti, W. M. Gifford, M. Win, and M. Chiani. Optimized simple bounds for diversity systems. *IEEE Transactions on Communications*, 57(9):2674–2685, 2009.

[34] A. Conti, M. Z. Win, and M. Chiani. Slow adaptivm-qam with diversity in fast fading and shadowing. *IEEE Transactions on Communications*, 55(5):895–905, 2007.

[35] W. Dai. Crypto++5.5.2. Available at http://www.eskimo.com/ weidai/cryptlib.html.

[36] J. Davis, A. Fagg, and B. N. Levine. Wearable computers and packet transport mechanisms in highly partitioned ad hoc networks. In *Proceedings of IEEE ISWC*, Oct. 2001.

[37] D. De Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proceedings of ACM MOBICOM*, 2003.

[38] S. de Vries and R. Vohra. Combinatorial auctions: A survey. *INFORMS Journal on Computing*, 15(3):284–309, 2003.

[39] W. Du and Z. Zhan. Using randomized response techniques for privacy preserving data mining. In *Proceeding of ACM SIGKDD'03*, pages 505–510, 2003.

[40] Editorial. Whose scans are they, anyway? *Nature*, page 443, 2000.

[41] S. Eidenbenz, V. S. A. Kumar, and S. Zust. Equilibria in topology control games for ad hoc networks. In *Proceedings of the 2003 Joint Workshop on Foundations of Mobile Computing*, 2003.

[42] S. Eidenbenz, G. Resta, and P. Santi. Commit: A sender-centric truthful and energy-efficient routing protocol for ad hoc networks with selfish nodes. In *Proceedings of IPDPS'05*, 2005.

[43] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *Proceedings of ACM SIGKDD*, pages 217–228, 2002.

[44] B. Fritzke. A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems 7*, pages 625–632. MIT Press, 1995.

[45] T. E. Gamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *ACM Transactions on Information Theory*, IT-31(4):469–472, 1985.

[46] GloMoSim. http://pcl.cs.ucla.edu/projects/glomosim/.

[47] B. Goethals, S. Laur, H. Lipmaa, and T. Mielik. On private scalar product computation for privacy-preserving data mining. In *Proc. the 7th Annual International Conference on Information Security and Cryptology*, pages 104–120, 2004.

[48] O. Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge University Press, Aug. 2001.

[49] O. Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, May 2004.

[50] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of Annual ACM Conference on Theory of Computing*, pages 218–229, 1987.

[51] A. Goldsmith and S.-G. Chua. Variable-rate variable-power mqam for fading channel. *IEEE Transactions on Communications*, 45(10):1218–1230, 1997.

[52] Q. He, D. Wu, and P. Khosla. Sori: A secure and objective reputation-based incentive scheme for ad-hoc networks. In *Proceedings of WCNC'04*, Atlanta, GA, Mar. 2004.

[53] HIPAA. National standards to protect the privacy of personal health information. In *http://www.hhs.gov/ocr/hipaa/finalreg.html*.

[54] P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: Social-based forwarding in delay tolerant networks. In *Proceedings of ACM MOBIHOC'08*, May 2008.

[55] *Proceedings of IEEE INFOCOM '03*, San Francisco, CA, Apr. 2003.

[56] G. Jagannathan and R. N. Wright. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *Proceedings of ACM SIGKDD*, pages 593–599, 2005.

[57] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Medard. Resilient network coding in the presence of byzantine adversaries. In *Proceedings of IEEE INFOCOM*, 2007.

[58] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. In *Proceedings of ACM SIGCOMM '04*, Portland, OR, Aug. 2004.

[59] M. Jakobsson, J. P. Hubaux, and L. Buttyan. A micropayment scheme encouraging collaboration in multi-hop cellular networks. In *Proceedings of Financial Crypto 2003*, La Guadeloupe, Jan. 2003.

[60] J. J. Jaramillo and R. Srikant. Darwin: Distributed and adaptive reputation mechanism for wireless ad-hoc networks. In *Proceedings of ACM MOBICOM*, 2007.

[61] E. Jones, L. Li, and P. Ward. Practical routing in delay-tolerant networks. In *ACM Chants Workshop*, Aug. 2005.

[62] S. Katti, S. Gollakota, and D. Katabi. Embracing wireless interference: Analog network coding. In *Proceedings of ACM SIGCOMM*, 2007.

[63] S. Katti, D. Katabi, H. Balakrishnan, and M. Medard. Symbol-level network coding for wireless mesh networks. In *Proceedings of ACM SIGCOMM*, 2008.

[64] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft. XORs in the air: Practical wireless network coding. In *Proceedings of ACM SIGCOMM '06*, Pisa, Italy, Sept. 2006.

[65] S. Laur, H. Lipmaa, and T. Mielikaihen. Cryptographically private support vector machines. In *Proceeding of ACM SIGKDD*, pages 618–624, 2006.

[66] S. Lee, G. Pan, J. Park, M. Gerla, and S. Lu. Secure incentives for commercial ad dissemination in vehicular networks.

[67] F. Li and J. Wu. Frame: An innovative incentive scheme in vehicular networks. In *Proceedings of IEEE International Conference on Communications (ICC)*, 2009.

[68] X.-Y. Li, Y. Wu, P. Xu, G. Chen, and M. Li. Hidden information and actions in multi-hop wireless ad hoc networks. In *Proceedings of ACM MOBIHOC*, 2008.

[69] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO 00)*, Aug. 2000.

[70] Y. Lindell and B. Pinkas. Privacy preserving data mining. *Journal of Cryptology*, 15(3):177–206, 2002.

[71] R. P. Lippmann. An introduction to computing with neural networks. *IEEE Acoustics Speech and Signal Processing Magazine*, pages 4–22, 1987.

[72] J. Luo and J.-P. Hubaux. A survey of research in inter-vehicle communications. *Securing Current and Future Automotive IT Applications*, pages 111–122, 2005.

[73] http://madwifi.org.

[74] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Sustaining cooperation in multi-hop wireless networks. In *Proceedings of NSDI'05*, Boston, MA, May 2005.

[75] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay a secure two-party computation system. In *Proceedings of the 13th conference on USENIX Security Symposium*, 2004.

[76] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of ACM MOBICOM*, 2000.

[77] Meraki Networks. http://meraki.com.

[78] P. Michiardi and R. Molva. Core: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc network. In *Communications and Multimedia Security Conference (CMS) 2002*, Portoroz, Sept. 2002.

[79] P. Michiardi and R. Molva. Analysis of coalition formation and cooperation strategies in mobile ad hoc networks. *Ad Hoc Networks*, 3(2):193–219, 2005.

[80] F. Milan, J. J. Jaramillo, and R. Srikant. Achieving cooperation in multihop wireless networks of selfish nodes. In *Proceedings of GameNets'06*, 2006.

[81] MuniWireless LLC. http://www.muniwireless.com.

[82] D. Myers and R. Hutchinson. Efficient implementation of piecewise linear activation function for digital vlsi neural networks. *Electron. Letter*, pages 1662–1663, 1989.

[83] M. J. Osborne and A. Rubenstein. *A Course in Game Theory*. The MIT Press, 1994.

[84] J. Ott and D. Kutscher. A disconnection-tolerant transport for drive-thru internet environments. 2005.

[85] D. C. Parkes, R. Cavallo, N. Elprin, A. Juda, S. Lahaie, B. Lubin, L. Michael, J. Shneidman, and H. Sultan. ICE: An iterative combinatorial exchange. In *Proc. 6th ACM Conf. on Electronic Commerce (EC'05)*, pages 249–258, 2005.

[86] M. S. Rahman, C. S. Iliopoulos, I. Lee, M. Mohamed, and W. F. Smyth. Finding patterns with variable length gaps or don't cares. In *Proceedings of COCOON'06*, Aug. 2006.

[87] S. J. Rassenti, V. L. Smith, and R. L. Bulfin. A combinatorial auction mechanism for airport time slot allocation. *Bell Journal of Economics*, 13:402–417, 1982.

[88] M. T. Refaei, V. Srivastava, L. DaSilva, and M. Eltoweissy. A reputation-based mechanism for isolating selfish nodes in ad hoc networks. In *Proceedings of MobiQuitous'05*, 2005.

[89] D. Reichardt, M. Miglietta, L. Moretti, P. Morsink, and W. Schulz. Safe and comfortable driving based upon inter-vehicle-communication. In *Proceedings of IEEE Intelligent Vehicle Symposium*, 2002.

[90] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. *Parallel Distributed Processing*, pages 318–362, 1986.

[91] D. E. Rumelhart, B. Widrow, and M. A. Lehr. The basic ideas in neural networks. *Communications of the ACM*, pages 87–92, 1994.

[92] Rutgers ORBIT project team. http://www.orbit-lab.org.

[93] N. B. Salem, L. Buttyan, J. P. Hubaux, and M. Jakobsson. A charging and rewarding scheme for packet forwarding in multi-hop cellular networks. In *Proceedings of ACM MOBIHOC*, 2003.

[94] U. Shevade, H. H. Song, L. Qiu, and Y. Zhang. Incentive-aware routing in dtns. In *Proceedings of ICNP'08*, 2008.

[95] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Spray and wait: An efficient routing scheme for intermittently connected mobile networks. In *Proceedings of WDTN'05*, Aug. 2005.

[96] V. Srinivasan, P. Nuggehalli, C.-F. Chiasserini, and R. Rao. Energy efficiency of ad hoc wireless networks with selfish users. In *Proceedings of European Wireless Conference*, 2002.

[97] V. Srinivasan, P. Nuggehalli, C.-F. Chiasserini, and R. Rao. Cooperation in wireless ad hoc networks. In Infocom 2003 [55].

[98] http://www.read.cs.ucla.edu/click/.

[99] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. In *Technical Report CS-200006, Duke University*, 2000.

[100] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of SIGKDD'02*, pages 639–644, 2002.

[101] J. Vaidya and C. Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *Proceedings of ACM SIGKDD*, pages 206–215, 2003.

[102] J. Vaidya and C. Clifton. Privacy preserving naive bayes classifier for vertically partitioned data. In *Proceedings of SIAM International Conference on Data Mining*, 2004.

[103] W. E. Walsh, D. C. Parkes, T. Sandholm, and C. Boutilier. Computing reserve prices and identifying the value distribution in real-world auctions with market dynamics. In *Proc. 23rd AAAI Conference on Artificial Intelligence (AAAI'08)*, pages 1499–1502, 2008. Short paper.

[104] L. Wan, W. K. Ng, S. Han, and V. C. S. Lee. Privacy-preservation for gradient descent methods. In *Proc. of ACM SIGKDD*, pages 775–783, 2007.

[105] W. Wan, X.-Y. Li, and Y. Wang. Truthful multicast in selfish wireless networks. In *Proceedings of ACM MOBICOM*, 2004.

[106] W. Wang, S. Eidenbez, Y. Wang, and X.-Y. Li. OURS-optimal unicast routing systems in non-cooperative wireless networks multihop routing in sensor networks. In *Proceedings of ACM MOBICOM*.

[107] W. Wang and X.-Y. Li. Low-cost routing in selfish and rational wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(5), 2006.

[108] R. Wright and Z. Yang. Privacy-preserving bayesian network structure computation on distributed heterogeneous data. In *Proceeding of ACM SIGKDD'04*, pages 713–718, 2004.

[109] F. Wu, T. Chen, S. Zhong, L. E. Li, and Y. R. Yang. Incentive compatible opportunistic routing for wireless networks. In *Proceedings of ACM MOBICOM*, 2008.

[110] J. Wu and R. Axelrod. How to cope with noise in the iterated prisoner's dilemma. *The Journal of Conflict Resolution*, 39(1), 2005.

[111] Q. Xu, T. Mark, J. Ko, and R. Sengupta. Vehicle-to-vehicle safety messaging in dsrc. In *Proceedings of ACM Workshop VANET*, 2004.

[112] X. Yang, J. Liu, F. Zhao, and N. Vaidya. A vehicle-to-vehicle communication protocol for cooperative collision warning. In *Proceedings of MobiQuitous 2004*, 2004.

[113] Z. Yang, S. Zhong, and R. Wright. Privacy-preserving classification of customer data without loss of accuracy. In *Proceedings of 5th SIAM International Conference on Data Mining (SDM)*, 2005.

[114] A. Yao. How to generate and exchange secretes. In *Proc. of the 27th IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.

[115] A. C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, 1982.

[116] H. Yu, X. Jiang, and J. Vaidya. Privacy-preserving svm using nonlinear kernels on horizontally partitioned data. In *Proceeding of SAC'06*, 2006.

[117] Z. Yu, Y. Wei, B. Ramkumar, and Y. Guan. An efficient signature-based scheme for securing network coding against pollution attacks. In *Proceedings of IEEE INFOCOM*, 2008.

[118] N. Zhang, S. Wang, and W. Zhao. A new scheme on privacy-preserving data classification. In *Proceedings of ACM SIGKDD*, pages 374–383, 2005.

[119] S. Zhong, J. Chen, and Y. R. Yang. Sprite, a simple, cheat-proof, credit-based system for mobile ad-hoc networks. In Infocom 2003 [55].

[120] S. Zhong, L. Li, Y. Liu, and Y. R. Yang. On designing incentive-compatible routing and forwarding protocols in wireless ad-hoc networks — an integrated approach using game theoretical and cryptographic techniques. In *Proceedings of ACM MOBICOM*, 2005.

[121] S. Zhong and F. Wu. On designing collusion-resistant routing schemes for non-cooperative wireless ad hoc networks. In *Proceedings of ACM MOBICOM*, 2007.