USING NONCOOPERATIVE POTENTIAL GAMES

TO IMPROVE NETWORK SECURITY

By

PATRICK D. HARRINGTON

Bachelor of Arts, English
Oklahoma Baptist University
Shawnee, OK
1996

Bachelor of Science, Computer Science
Northeastern State University
Tahlequah, OK
1999

Masters of Science, Computer Science
University of Tulsa
Tulsa, OK
2002

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
July, 2010

USING NONCOOPERATIVE POTENTIAL GAMES TO IMPROVE NETWORK SECURITY

Dissertation Approved:

Dr. Johnson Thomas

Dissertation Adviser

Dr. George Hedrick

Dr. Nohpill Park

Dr. Marilyn Kletke

Dr. Mark E. Payton

Dean of the Graduate College

ACKNOWLEDGMENTS

I would like to first and foremost thank my wife, Rayna, for her unfailing support during my graduate career.

I would also like to thank my advisor, Dr. Johnson Thomas, for his guidance, wisdom, and perspective for my work, as well as Drs. Hedrick, Kletke, Park, and Chandler for their help.

I would also like to express thanks to the Department of Mathematics & Computer Science at Northeastern State University in Tahlequah, and in particular my department chair, Dr. Darryl Linde, for arranging my work schedule to facilitate making progress toward a PhD. This would also not have been possible without the support of the faculty, in particular Dr. Rick Matzen, Bill King, Gordon Shamblin, and Dr. Rad Alrifai. I would also like to especially acknowledge Bill, Gordon, and Rick for their lending of books and advice on mathematics. I would also like to thank my parents, Drs. Doug Harrington and Elisabeth Wentz-Harrington, for encouraging me to pursue a career in computer science and in teaching; and my late mother, Martha Harrington, who inspired me to live life fully.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

SYMBOLS

A	partition on graph G that divides it into coalitions
c	cost
$c\ (s_i\ )$	cost of security value s_i for vertex v_i
$c\ ((s_{ij}), \sigma_{ij}\ (s_{ij}\ ))$	cost of security value s_{ij} for edge e_{ij} given constraints σ_{ij}
$d\ (j)$	data of node j
E	set of edges
e_{ij}	edge from vertex i to vertex j
e_{-ij}	all other edges besides vertex i to vertex j
G	graph composed of the set of vertices and set of edges
Γ	game of G
i	node (player) i, represented in the graph by vertex v_i
j	node (player) j, represented in the graph by vertex v_j
$\prod$	potential function
$r_{ij}\ (d(j))$	read access from i to j
$\mathbb{R}^+$	set of positive real numbers
S	non-decreasing sequence of nonnegative security levels
s_i	security level of player i
s_{-i}	security levels of all other players besides i
s_{ij}	security level of e_{ij}
s_{-ij}	security level of e_{-ij}
$s*_i$	optimal action for player i
$s*_{-i}$	optimal action for all other players besides i

$s*$	action profile in equilibrium
$(s_{ij}, \sigma_{ij}(s_{ij}))$	security of s_{ij} given constraint σ_{ij} ; also known as move s_{ij} for player *i*
$(s_i', \sigma_i(s_i'))$	suboptimal move for player *i*
$(s_i^*, \sigma_i(s_i^*))$	optimal (equilibrium) move for player *i*
$c_{ij}(s_{jk}, \sigma_{jk}(s_{jk}))$	side payment move for player *i* to *j* to induce change in s_{jk}
σ_i	constraint of player *i*
σ_{-i}	constraints of all other players besides *i*
$\sigma_i(s_i)$	constraint of security value s_i for player *i*
$\sigma_{-i}(s_{-i})$	constraint of security value s_{-i} for all other players besides *i*
$\sigma_{ij}(s_{ij})$	constraint σ_{ij} of s_{ij}
u	utility function
V	set of vertices
v_i	vertex *i*, also referred to as node *i* or player *i*
v_j	vertex *j*, also referred to as node *j* or player *j*
$w_{ij}(d(j))$	write access from *i* to *j*
y_{ij}	current encryption strength of connection e_{ij}

DEFINITIONS

- Action: assigning a value to a variable.
- Action profile: a sequence of actions for each player in the game; there is one action corresponding to each player in the sequence.
- Coalition: a group of players who share a common security level and preferences.
- Constraint: a function which maps the domain of possible values a variable may take to the codomain or range of values that do not violate the constraints.
- Constraint satisfaction problem: abbreviated CSP, it is a problem defined by a set of variables and corresponding constraints on the variables.
- Cooperative game: a game in which players make binding commitments that require the consent of all parties to break the commitment.
- Cost: a measure of lost utility. Cost is denominated in units of security.
- Edge: see *Link*
- Equilibrium: the best possible move a player can make based on the assumption that all other players also make their best possible moves. Equilibrium maximizes utility, and is a partial order on the actions: they are reflexive, anti-symmetric, and transitive with respect to one another.

- Finite Improvement Property: all potential games possess this property whereby there is a best possible move for each state of the game. Neither the game nor the improvement can go on forever, and there can be no repeated states, as the game must have a finite end with unique moves and finite utility.
- Game: a game consists of the *Players*, *Information*, *Actions*, and *Utilities*.
- Game theory: the study of actions by decision-makers who are aware that their actions have an impact on others.
- Information: a player's knowledge about the set of variables describing the game, which includes constraints on what actions can and cannot be taken.
- Incomplete Preferences: see *Partial Preferences*
- Link: a one-way directed connection between nodes.
- Metric: see *Security metric*
- Mixed Strategy: a plan that maps a player's information to a probability distribution (ranging from 0 to 1) over the set of pure strategies.
- Nash equilibrium: see *Equilibrium*
- Node: nodes in graph *G* represent computers in the network. Nodes are the players in the game.
- Noncooperative game: a game in which any commitments made can be unilaterally broken.
- Objective function: describes a function to be maximized by all variables in the constraint satisfaction problem. Not all constraint satisfaction problems have an objective function.

- Pareto optimized: a state in which a player can take no action deviating from this state without decreasing utility for itself or some other player.
- Partial Preferences: equations in which a set of variables may have intermediate or unassigned values.
- Partition: to form coalitions, the network is divided into non-overlapping subsets according to a pairwise disjoint function known as a partition.
- Pay: compensation measured in utility to or from another node.
- Player: an individual making decisions and taking actions in the game. See also Node, as players are represented by nodes or computers in the network.
- Potential function: the potential function takes the same input as the utility function, and returns a calculation that is either the same sign (for an ordinal potential function) or equal to (for an exact potential function) the value calculated by the utility function of each player. The exact potential function is defined as applicable when players move according to mixed strategies, while the ordinal potential is defined to be applicable when players move according to pure strategies.
- Potential game: a game possessing a potential function.
- Preferences: the current best values for the variables in the game. Since the variables in the game represent security levels, the preferences are the current optimal security levels. They correspond to the environment and its current state.
- Problem state: some, but not necessarily all, variables have values assigned to them.
- Pure strategy: a plan that maps a player's information to a single action.
- Security: the opposite of vulnerability. In our game, security is optimized by minimizing vulnerability.

- Security metric: quantitative measurement of some aspect of security. Our metrics are cryptography, data sensitivity, and access control.
- Side payment: an action that consists of a payment made from node *i* to node *j* to induce *j* to change the security of its link s_{jk} to another node *k*.
- Solution: a complete assignment of values to variables in a constraint satisfaction problem that do not violate the respective constraints.
- Strategy: a plan to take a specific action in a game. Each player has a set of strategies that govern the player's actions in every conceivable situation.
- Utility: the quantity of positive feedback (which in our game is security) which a player receives for taking an action. Utility is denominated in units of security.
- Utility function: the utility function takes as input the set of possible actions available to a player and determines a player's preference over the given set of possible actions. These preferences enable a player to choose the best values that can be assigned to the game variables and maximize its utility at each turn in the game.
- Vulnerability: a weakness whereby an outsider may destroy, alter, or steal one's possessions, which is the data possessed by the nodes as well as the nodes themselves. Vulnerability is the opposite of security.
- Weighted potential game: a type of potential game that possesses a utility function which is directly related to other players' utility functions. A weighted potential game possesses a potential function for either mixed or pure strategies. We use a weighted potential game to optimize network security.

CHAPTER I

INTRODUCTION

Computer networks have changed significantly since their first invention. In the early days of the Internet, the network was composed of nearly identical computers connected to one another by wire, whereas today's wireless network consists of increasingly heterogeneous devices ranging from PCs to iPhones to servers. Not only are today's networks more complex to secure, but the attacks on the network are far more sophisticated. The advent of cloud computing and of cyber-physical systems composed of sensor networks, actuators, and control systems has only increased the heterogeneity and complexity of securing such a network. While a great deal of work has been reported on security for individual computers or devices, very little work has been done to optimize security for the overall network when all devices forming the network are not only heterogeneous but also autonomous. The security of the entire network itself is critical and cannot be done piecemeal.

The problem considered in our work is how to maximize an entire network's security when it is made up of computers or devices that may be very different from one another. The computers act independently and autonomously, with no broad or central coordinator. There has been previous research with focus on individual components forming the network, such as Agah [26,27] and Demirbas [29], but these approaches are limited as they fail to optimize security for the entire network. In these approaches, network security is only as good as the security of the weakest component. Our work proposes to allow individual computers to work together to achieve the optimum security for the entire network using game theory.

The problem considered in our work centers on a general representation of heterogeneous computers, each with different hardware and software. We consider hardware to include all physical, non-software related computer components such as CPU, RAM, and available power or battery life. Software thus includes both system software, which refers to the operating system, and application software, which includes all other software. The hardware and software of a computer will subsequently restrict its ability to sustain maximum security, which is only possible in an ideal world. The network under consideration is a wireless network, where connections between nodes are assumed to exist a priori. The nodes themselves are not physically mobile, but we assume connections change. These connections change during the network security optimization process.

Game theory is one solution to address the problem of maximizing security within a system’s limitations. Game theory, as defined by Rasmusen [8], is the study of actions by decision-makers who are aware that their actions have an impact on others. A game consists of the players, information, actions, and utilities. The players are defined as the individuals making decisions and taking actions in the game. We define the computers in the network as nodes, and these nodes are the players in our game theoretic security solution proposed in our thesis. In this situation, the players are the computers, or nodes, in the network. The information is defined as the players' knowledge about the set of variables describing the game, which includes constraints on what actions can and cannot be taken. These variables are input to a player's utility function, which determines a player's preference over a given set of possible actions. The utility function takes as input the set of possible actions available to a player and determines a player's preference over the given set of possible actions. These preferences enable a player to choose the best values that can be assigned to the game variables and maximize its utility at each turn in the game. This plan of action is its strategy. The player then takes the best action by assigning these best values to its variables, resulting in a new state of the game. At each turn in the game the player applies new information to its utility function, determines its preference over a given set of possible actions, chooses the best values that can be assigned to the game variables, and maximizes its security.

Game theory as an optimization technique offers several advantages over other rule-based, linear programming, tree-pruning or backtracking algorithms, as well as artificial intelligence (AI) techniques. Game theory allows the game designer to arrive at a solution by developing the model and evaluation criteria for players so they can determine the optimal next move to make in the game to maximize their utility; as the game plays out, the solution evolves, giving the optimal configuration for the network. The designer of the game need only model the moves of the game and their context, allow the players to act autonomously according to these rules, and an optimization results.

Furthermore, game theory has the advantage of finding solutions to problems that would otherwise be difficult to solve using other approaches. Game theory can be used to model complex interactions and environments, which in our problem is a heterogeneous network. The use of mixed strategies is unique to game theory and allows for modeling more complex decision-making processes than could be done by other techniques. Through the use of mixed strategies, the game designer is able to model the possibility of mistakes made by game players. With pure strategies, players always choose the optimal strategy. With mixed strategies, however, a player chooses a move according to a probability distribution over the set of pure strategies, thus taking the optimal action some percentage of the time and less than optimal otherwise.

In our thesis, we will examine the methodology behind how certain game subsets have been proven to have mathematical advantages over others and how such game subsets make it easier to prove a solution is optimal. Furthermore, our thesis will take advantage of the modeling techniques exclusive to a game theoretic problem-solving approach through mixed strategies. It will also allow coalitions, which are otherwise known as groups, to be formed among the game players as part of the game optimization process. Coalitions can be used to improve efficiency as well as give a broader and greater overall security than would be possible in the absence of coalitions. We will develop a heterogeneous game theoretic distributed security model in which game players choose the move that optimizes security; once a player acts on its choice of moves, the state of the game changes, and game play leads to security optimization of the network.

An important aspect of our work is that the security optimization found can be validated as opposed to existing or traditional solutions, whose effectiveness is subjective. Traditional security definitions such as “works as expected” [31] are no longer sufficient. An April 2009 report by the Homeland Security Newswire [34] revealed that because they are now connected to the Internet, a significant number of US power plants have, over the past few years, had their systems compromised by intruder programs, and that these potential sleeper agents are still in place in the systems across the entire US electrical grid. Furthermore, these programs have become embedded in other parts of the US infrastructure, which includes water treatment plants. To date, no one is certain how to remove them, or what these programs' function truly serves beyond gathering intelligence, but the sophistication with which they are embedded suggests a well-financed organization with skilled members, and is thus most likely related to a foreign government. If so, it is likely that these programs indeed will become active if the US goes to war with the foreign government that put the sleeper software there, and can potentially devastate the US infrastructure. Still, the systems are continuing to work "as expected," and "officials...do not see an immediate danger" [34].

We believe that statements such as these are representative of the traditional security paradigm and do not preclude the possibility that these programs will prevent our power plants to continue to work "as expected" in the future. Clearly, there is a significant need for paradigm shift in the definition of security from a qualitative definition to one that can be measured quantitatively. Security metrics can be used as means to accomplish this paradigm shift. Security metrics is a new field of computer security and its viability as a field of study was first recognized as serious work only a few years ago, but it is beginning to become more widely used. The goal of security metrics is to be able to better analyze security by evaluating a system and quantitatively representing its inner workings and relationships between entities that form the system itself. Security metrics can thus be used as a basis to give quantitative meaning to the qualitative definition.

To use metrics in our security solution, we must devise a method of introducing security metrics themselves to a game. To date, no one has done so. However, we believe defining the problem itself as a game possessing a system of variables and constraints on security and energy can allow for the introduction of security metrics. This type of problem, with constraints and variables, is referred to as a constraint satisfaction problem.

We define constraints as functions that restrict the domain of values a variable may take to a subset that forms a codomain or range. Constraint satisfaction problems are a traditional AI approach and can represent a multi-variable problem with varying domain and ranges. Constraint satisfaction problems are traditionally solved using backtracking, tree pruning, or linear programming. In our work, however, game theory will be used to solve the constraint satisfaction problem, allowing us to model optimal and suboptimal decisions along with problem constraints, and validate our work using security metrics. To further validate the security optimization by our algorithm, we will compare its results with that of the established Kerberos algorithm, which is similar to our own algorithm in that it is designed to provide security for an entire network and uses methods of encryption and access control. For a more complete description of Kerberos, we refer you to [35].

Our work can be generalized to optimize security for almost any network or system because it can be applied to a network made up of computers that possess any degree of heterogeneity. With regard to the sleeper software in the article mentioned earlier, since the parties responsible for securing the power plant have not divulged the manner in which the sleeper agent programs were detected, our only conclusion can be that their method is not one they wish to divulge to protect classified security methods or the programs were discovered by accident. We believe that since the past definition of security is based on the "works as expected" definition in [31], those individuals responsible for power plant security were simply following this definition. Since even today there have been no deviations in power plant expected performance, by this old definition the power plant is secure, and authorities even attest this when they say that there is "no immediate danger" [34]. We believe this event exemplifies the need to improve and redefine security; a first step in doing so would be to ensure that definitions of security are no longer quite as qualitative, as in the "works as expected" definition, but can be measured quantitatively.

Our work is the first of which we are aware that uses a weighted potential game for a network security optimization solution. We will use a noncooperative instead of a cooperative game because it better fits our problem definition: noncooperative games focus on strategy and utility maximization, whereas cooperative games do not. Noncooperative games model relationships that may be unilaterally changed by players; there are thus no binding contracts. At each turn of a game, a player chooses its best move to maximize its utility. The best possible move a player can make versus all best possible moves of all other players is defined as an equilibrium, which is also known as Nash equilibrium. This equilibrium is not necessarily the best outcome, because there could still be multiple equilibrium or even a solution that goes beyond what is achievable by equilibrium using traditional noncooperative game analysis. Such improved outcomes and “best” equilibrium are defined as Pareto optimal equilibrium. Potential games are a subset of the class of noncooperative games that indeed do find these Pareto optimal solutions to game theoretic problems.

Pareto optimization goes beyond the definition of equilibrium. It is, informally, the best of the equilibrium or more optimal solution than equilibrium. Pareto optimized is defined as a state in which a player can take no action deviating from this state without decreasing utility for itself or some other player. That said, the best way to illustrate the difference between Pareto optimal and equilibrium is by the example of the game known as the Prisoners’ dilemma. In the Prisoners’ dilemma, two players are accused of committing a crime. The players are separated and cannot communicate with one another. The following Table 1.1 illustrates the respective utilities for each action, with units measured in years of imprisonment:

Player 1	Player 2		
		Stay silent	Blame
	Stay silent	-1,-1	-10,0
	Blame	0,-10	-8,-8

Table 1.1: Prisoners’ dilemma

Clearly, the solution in equilibrium is for each player to *Blame* the other and accept 8 years imprisonment because of the real possibility of being blamed while staying silent, thus receiving 10 years in prison. However, the Pareto optimal solution is for both to *Stay silent* because it gives the most utility for all players. If prisoners could cooperate and had incentive to do so, they could get better than equilibrium. Shapley is recognized as being the first to develop a technique that can lead to a Pareto optimal solution for all players and games [2], which will be discussed in greater detail in Chapter II below. Chapter II contains the literature related to our security solution. In Chapters III - VI, we present the methodology of our solution and illustrate its improvements over previous work. Chapter VII contains our prototype simulation and its results, and Chapter VIII contains our conclusions.

CHAPTER II

REVIEW OF LITERATURE

Game theory is the analysis of games. Games can range from simple games, such as chess or checkers, to more complex games used in economic or strategic planning. While some roots of game theoretic strategy can be found in the philosophical writings of Plato and Kant [33], the first real work studying games themselves did not begin until the 19th century. This initial exploration into game theory was very simple and straightforward. Working separately, Cournot and Bertrand analyzed simple games where players chose the optimal move to maximize their utility [32]. While both made only initial contributions to game theory, Cournot is more widely recognized today due to his work on games that have an economic application. Regardless, nothing went beyond initial examination due to the lack of formal mathematical specification. True game theory began in the 20th century with the work by four men: Von Neumann and Morgenstern, Nash, and Shapley. In 1944, Von Neumann and Morgenstern wrote the groundbreaking "The Theory of Games and Economic Behavior" [17], which contained the analysis techniques, general formula for calculating utility to a player, and game terminology still in use today. VonNeumann and Morganstern's work created the tools necessary to successfully analyze games, which led to a number of future discoveries. In addition, their work ensured that future explorations of game theory to solve problems were restricted to situations and problems that could be fully described by mathematics. For example, using games to model computer problems has been largely successful for the reason that the factors affecting a problem with computers can be modeled with a great degree of certainty. As we know, although computers can malfunction, they are not capable of panic. Modeling a computer problem using game theory thus has greater likelihood of reaching an optimal result.

Building upon VonNeumann and Morganstern’s work, John Nash developed the noncooperative game and defined game equilibrium in the 1950s. At the same time, Shapley [21] - [22] took VonNeumann and Morganstern’s work and used it to define the field of cooperative games. Cooperative games tend to spend little effort focusing on the strategy of each player in the game; noncooperative games do the exact opposite. While noncooperative games are now more widely studied than cooperative games, with [8] devoting an entire textbook to them, Shapley is the only living member of the four founders of game theory, and he continues to publish to this day. His efforts have turned toward noncooperative games, but they contain new ideas of how to create noncooperative games that go beyond their traditional definition, giving these games cooperative characteristics. His most recent work [6] developed a game that allows coalitions to form even though variables possess intermediate values. Furthermore, Shapley has made significant improvements on the class of noncooperative games known as potential games [2]; this work is one of the foundations of our thesis.

A potential game is a game possessing a function, called a potential function, which is similar to a players' utility function. The potential function takes the same input as the utility function and outputs a value; in the case of an exact potential function, the output is equivalent to that of the result given by the utility function; in the case of an ordinal potential function, the output has only the same sign (positive or negative) as the result output by the utility function. When players move according to mixed strategies, games possessing a potential function can have only an exact potential function; when players move according to pure strategies, the game possesses an ordinal potential function.

Let us give the definition of a potential function in its relationship with a utility function as given by Shapley [2]. Given utility function *u* and potential function *v*, move *m* in the game is an element in the set of moves *M*, denoted

$$m \in M \tag{2.1}$$

where *M* is a member of the set of positive real numbers, denoted

$$M \in \mathbb{R}^{+} \tag{2.2}$$

When move *m* is input to the ordinal potential function and also to the utility function, the value calculated by the ordinal potential function is positive or zero if-and-only-if the value calculated by the utility function is positive or zero, denoted

$$u(m) \geq 0 \ \leftrightarrow v(m) \geq 0 \tag{2.3}$$

For the same utility function *u*, let us give the definition for the exact potential function according to [2]. For the exact potential function, which is also denoted *v*, the value calculated by the utility function *u* is equivalent to the value calculated by the exact potential function *v* for input *m*, denoted

$$u(m) = v(m) \tag{2.4}$$

Before delving further into Shapley's work with potential games, we must examine the history of the potential game and its definition. In 1973 Rosenthal [14] did the earliest work with a type of potential games called congestion games. These games built upon Cournot's initial work on analyzing oligopolies, whereby all players are selling the same item at different prices, and no new players are introduced once the game begins. The game players were all identical, and contained a finite set of actions. The utility was dependent upon the actions of other players. A linear inverse demand function, with respect to the item being sold in the game, was used to calculate utility. All moves in Rosenthals' work were considered to be "pure" and were not subject to a probability distribution. Prior to Shapley's proof in 1996 [2], showing that congestion games were an isomorphism of noncooperative games, a considerable amount of work on potential games was still done using Rosenthal's work as a foundation. Even today, much of the research still focuses on Rosenthal's work [11], [13] to the point that it contains the flaws that Shapley has pointed out in 1996. In this respect, it is somewhat myopic; it still follows the pattern of excluding mixed strategies and does not reference Shapley's recent work in 1996 [2], with the exception of the recent work by Komali [4].

Shapley's work with potential games [2] is significant. It improves on Rosenthal's work by adding proofs that give enhanced definitions of potential games and their requirements, making it easier to design a game that has equilibrium and is also Pareto optimal. Shapley's research deals with mixed and pure strategies and heterogeneous players, which was something that Rosenthal was unable to solve.

Shapley was also able to develop an improved type of potential game, the weighted potential game.

Defined by Shapley [2], a game that possesses a utility function which is directly related to each of the players' utility functions is a weighted potential game. A weighted potential game possesses a potential function for either mixed or pure strategies, and has efficiency and convergence improvements over other potential games or myopic, greedy strategy games. Weighted potential games possess at least one equilibrium, and are easier to prove Pareto optimal. Furthermore, the weighted potential game is valid for both mixed or pure strategies and players with any degree of heterogeneity. Shapley's work with potential games extends to graph theory as well, containing a proof that any game containing a network using weighted potential for utility evaluation is connected, at minimum, by a simple cycle.

Shapley explains why the weighted potential game gives a result that is more likely to be optimal than other types of potential games. Because the utility function that determines the weight of benefit between two players is piecewise continuously differentiable, results can be mathematically analyzed prior to any empirical simulations by using the equation of the potential. Non-weighted potential games do not possess this property, as was also proven by Shapley in [2]. Furthermore, the weighted potential game will lead to an optimal solution regardless of the weight used.

Besides these improvements, Shapley also recognized and established rules for what leads to optimal solutions for a potential game, which is something not studied by Rosenthal. Foremost is what Shapley defined as the Finite Improvement Property: all potential games possess this in that there is a best possible move for each state of the game, but this alone is not enough to guarantee an optimal solution. In addition, neither the game nor the improvement can go on forever, and there can be no repeated states; the game must have a finite end with unique moves and finite utility. This particular property contradicts Rosenthal and Shapley has a mathematical proof to back it up. In addition, Shapley defined the requirements for non-weighted mixed and pure strategy potential games.

As a result of his work Shapley discovered that a potential game admits a cooperative solution to noncooperative game; in theory this would allow the players in the Prisoners' dilemma to collaborate indirectly by maximizing the potential function. To explain this further, the incorporation of a potential function gives the noncooperative players, which have no binding contract to one another by the nature of the noncooperative game, a type of motivation and nonverbal collusion which creates a unique hybrid of noncooperative and cooperative games; if all players try to jointly maximize the potential function it effectively acts as a binding contract between the players. Hence in the case of the Prisoners' dilemma, instead of each accusing the other, both players play the move to remain silent by jointly maximizing the potential function between them. The Pareto optimal solution would thus be reached concurrently by both players because the potential function changed the conditions of the game, allowing for a more beneficial solution for both players to be reached.

Our understanding of Shapley's work in [2] is that Pareto-optimality supersedes equilibrium by its definition. While it is not spelled out in English words, but in equations, much of his most groundbreaking work takes this form. We have not found any other works that reference this point about Pareto optimality, although the work by [4] implies that he understands it. However, despite the lack of other sources found, based upon the fact that much of Shapley's work is purely in the form of an equation with little written English-language explanation, and that no authors found thus far are aware of Shapley's contributions until referenced second-hand by another author who does understand Shapley's equations, we believe that we are correctly reading his work.

Shapley goes on to say in [2] that the existence of a potential function guarantees the existence of equilibrium. However, there may be several local equilibrium in addition to the best equilibrium overall. Rosenthal had previously postulated that the way to find the best equilibrium, which yields the Pareto optimal solution, is to solve for the maximum of the function, known as the argmax. While it seems obvious, it is not game theoretic, and defeats the purpose of the game. Still, other authors [4], [11], follow Rosenthal's bad example despite this, and even go so far to use it as a method of proof as suggested by Rosenthal. Shapley's work in [2] sheds light on Rosenthal's shortcomings: experimental results showed that solving for the argmax could be used to determine which equilibrium is the optimum, but formal proofs demonstrate why solving for argmax is an invalid proof or predictor of the game itself. If one considers it logically, Shapley's reasoning becomes obvious: solving for the maximum value of an equation is insufficient to describe the complexities of a game. Instead, Shapley said, argmax should be used as a refinement tool to identify and eliminate any possible false equilibrium. As stated earlier, authors such as [4], [11] have ignored much of Shapley's work in this area, correctly using argmax as a refinement tool but incorrectly using it as a proof of equilibrium.

Komali's proof for Pareto optimality of the equilibrium is similar to [11]. The optimal point is examined; once reached, change in utility would result in a player disconnecting from the network and subsequently cause the player to no longer be optimal or in equilibrium, which is correct. Ironically, parts of the Pareto optimality proofs by [4] and [11] are both logically unsound.

Our understanding of logical equivalence leads us to believe that the proofs on Pareto-optimality in [4], [11] are overworked via proof by contradiction to the point that they do not prove their original goal; these Pareto optimal proofs instead prove a logical in-equivalence of their original statement. We refer the reader to [4] and [11] for the proofs themselves. We select the proof in [4]. To help clarify our own argument, we reproduce their statements; a game that meets these criteria is considered Pareto optimal:

> If the potential function will converge to equilibrium in the game, then no player can deviate from an action in equilibrium without violating its constraints and thus decrease utility and make the action not in Pareto optimal equilibrium.

Let us examine the logical statements used in their proof-by-contradiction of a Pareto optimal game to explain our reasoning. Let the statement *p* represent the first logical statement of the proof in [4]

p: the potential function will converge to equilibrium in the game.

Let the statement *q* represent the first part of the second logical statement of the proof in [4]

> *q*: no player can deviate from its equilibrium action without violating its constraints.

Let the statement *r* represent the second part of the second logical statement of the proof in [4]

> *r*: no player can deviate from Pareto optimal equilibrium action without decreasing its utility.

Let the statement *s* represent the third logical statement of the proof in [4]. We understand the action causing deviation from equilibrium to mean "deviates from equilibrium action," giving

> *s*: if a player deviates from its Pareto optimal equilibrium action, then the action is not in equilibrium.

Note that statements *q*, *r*, and *s* are negations of statements. The first part of statements *q*, *r* and *s* address deviating from an equilibrium action, which is the opposite of playing an equilibrium action. The second part of the statements addresses violating constraints and decreasing utility, which is the opposite of preserving constraints and maximizing utility, respectively. Let us now break up *q, r, s* even smaller using the following statements:

> *t:* player plays action in Pareto optimal equilibrium

> *u:* player preserves constraints of its action

> *v*: player maximizes utility

We can now write *q*, *r*, and *s* symbolically:

$$q = \bar{t} \rightarrow \bar{u}, \quad r = \bar{t} \rightarrow \bar{v} \; , s = \bar{t} \rightarrow \bar{t}$$

The latter statement

$$s = \bar{t} \rightarrow \bar{t}$$

is what [4], [11] focus their proof-by-contradiction upon. However, proof-by-contradiction works when the hypothesis and conclusion are different, which is not what we have here in statement *s*. Proof-by-contradiction works by assuming the hypothesis true and conclusion false, and using this to arrive at a contradiction of the negated conclusion. We understand statement *s* above to be vacuous as it literally reads, "if a player plays an action in equilibrium, then a player does not play an action in equilibrium." The authors in [4], [11] did not break down statement *s* into its sub-statements. Writing the statements symbolically has exposed the over-complexity of this proof-by-contradiction approach.

This work by Komali in [4] focuses optimizing communications energy consumption in a network with game theory. In combination with ordinal potential functions, Komali constructs a game with graphs and greedy strategies where all players are identical and use pure strategies only. This work builds upon a proof by Shapley stating that said graphs are connected, at minimum, by a simple cycle. Specifically, Shapley states [2] that a graph representing a game is connected if the function used to calculate utility and thereby determine the best strategies is a potential function. Komali's resulting game is noncooperative, and achieves an equilibrium that is Pareto optimal. However, its game only addresses homogeneous players and ordinal utility with pure strategies, and does not examine heterogeneous players or mixed strategies. Its equation to calculate utility using linear inverse demand is a variation on Shapley's [2] and Rosenthal's [14] work: each player receives diminishing utility for every other player to whom it is connected, but also receives additional constant utility for maintaining network connectivity. Our understanding of the utility equation in Komali [4] is that it differs from the work in [2] because Komali has tailored his solution to fit his greedy strategies that determine player moves in the game; the utility calculations are only part of the players' strategy. The proof that Komali's algorithm in [4] is in equilibrium is in his earlier work [30]: however his proof ignores Shapley's instructions [2] and instead uses argmax as a method of proof, which is invalid according to [2]. Thus Komali's work leaves open the possibility of a subset of solutions not in equilibrium.

In our own simulations of Komali's algorithm in [4], its utility improvement fluctuates back and forth in a sine-wave-like pattern around a threshold, never completely reaching the Pareto optimal equilibrium; see Figure 2.1 below. These results indicate Komali is indeed taking argmax of the final set of utilities fluctuating around a threshold to distinguish between optimal and sub-optimal equilibrium. We believe this should have been incorporated into his algorithm; as it stands, this inability to distinguish between Pareto and sub-optimal equilibrium is a flaw. We conclude the flaw is most likely caused by Komali's greedy strategy used by players, which Shapley [2] says can tend to a suboptimal solution for potential games. However, Komali's earlier work in [30] acknowledges that the optimal utility indeed does transition around a threshold and that this is sufficient for him for convergence. Regardless, we do not agree as mentioned earlier, and believe improvements could be made in his work. Our own simulation of the energy optimization algorithm in Komali [4] is shown in Figure 2.1 below.

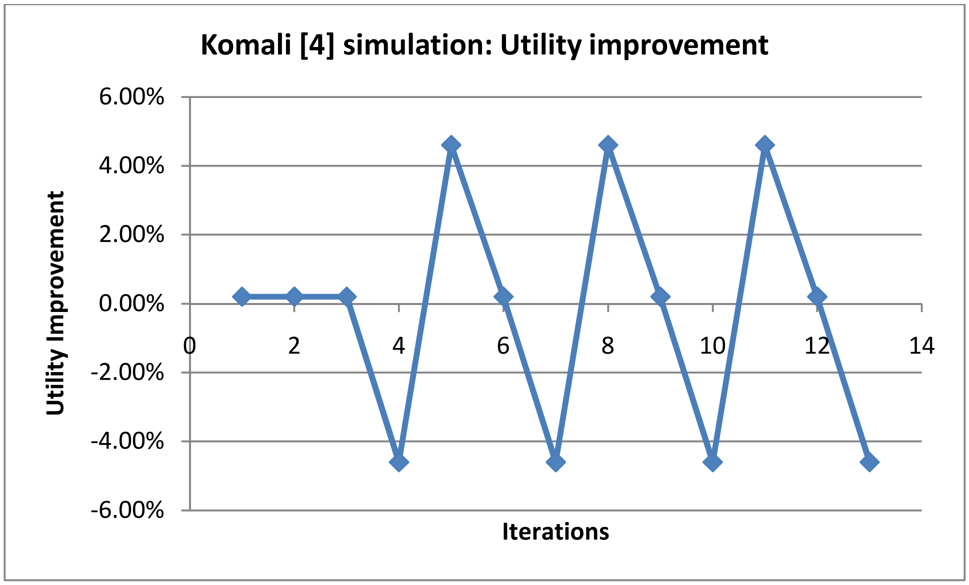


Figure 2.1: Our own simulation of the energy optimization algorithm in [4]

Besides the work by Komali, other work has been done in the area of tying noncooperative games to Pareto optimality. Heikkinen [11] examines a quality-of-service bandwidth allocation scheme in a distributed homogeneous network within the framework of a pure-strategy noncooperative game, and uses Pareto optimality to prove that the equilibrium reached is best overall. It also shares a common approach to [4] in that the algorithm used by the players to maximize utility is a greedy one, and was successful at achieving results for equilibrium that is also Pareto optimal for the examined cases. However, the work is applicable to homogeneous players and pure strategies only, as it does not build upon Shapley’s work in [2] despite its publication at a later date.

In addition to his work with potential games, Shapley also determined how to combine players in a game into coalitions by using a function that became known as the Shapley value [8], [21]. The Shapley value ensures that an individual player receives, as utility, an average of the utilities that other players receive for the player's actions; it acts as a sort of "golden rule" for players, and it was discovered by Shapley [2] that it is required for a game to have a potential function. In doing so, Shapley discovered additional requirements for all games using potential functions that were not listed before Shapley’s 1996 work. The game must also have a finite end with unique moves and finite utility. These requirements were not even hinted at in Rosenthal’s work. Shapley's 2008 work on multiplayer utility [6] continues this, examining forming coalitions without negotiation when only partial player preferences are known. This work assumes that that preference can be quantified. Preferences are the current best values for the variables in the game. Since the variables in the game represent security levels, the preferences are the current optimal security levels of the connections in the graph.

Partial preferences, which are sometimes called incomplete preferences, are equations in which a set of variables may have intermediate or unassigned values. In the case of a game, the variables correspond to the players or variables maximized by the players, and the partial preferences themselves refer to the intermediate states of the game prior to equilibrium. Coalition formation in [6] is done by quantitatively comparing partial and complete preferences of players and coalitions.

Individual preferences are used to facilitate coalition formation. These preferences are combined using the mathematical fact that equilibrium is a partial order, and any remaining unknowns are combined using a weighted sum. This technique is able to aggregate individual preferences to coalition preferences without needing to arbitrate between the players. While Shapley does not explain it in words, we understand his equations to indicate that the reason the coalition formation algorithm works so well with partial preferences is that it takes advantage of the mathematical foundations of game theory to form coalitions. Specifically, Shapley's coalition formation algorithm uses the property that an equilibrium is a partial order on the strategies as established in the mathematics of VonNeumann and Morganstern [17].

In addition, Shapley proves in [6] that because the design of a weighted potential game means the players are already receiving utility based upon the weight used, it leads more easily to coalition formation because of the collaboration already inherent in the weighted potential game. Coalition formation involves forming what are known as coalition preferences. We understand that these coalition preferences are quantified from a Cartesian product, which defines the actions of the game as described earlier, to a real number by using a weighted sum of the individual preferences that already exist prior to coalition formation. This is not the game theoretic weight as defined earlier in the context of a potential game, but refers to the algebraic definition, whereby a number is multiplied by each of the elements in the Cartesian product, and then the elements are added together.

Shapley points out that there is no formula given in past work indicating how to best determine these weights for coalition preferences, but if there is an already existing schema then it will facilitate easier coalition formation. Since this weight is between two players, and the coalition examined contains the same two players, it is in and of itself an optimized coalition preference weight. Coalitions themselves are already in wide use; for example, any network containing a Windows-NT system divides users into coalitions, which are usually referred to as groups in a Windows environment. It restricts certain users in a coalition from having access to resources on the network, while other users belonging to a different coalition are granted such access. The primary advantage of using coalitions to grant permissions is its speed: it is faster than other approaches. The first approach commonly used, a zero-knowledge based system, grants permissions to individuals if they can prove secret knowledge; but by doing so they do not reveal the secret itself. A graph isomorphism is a means of implementing a zero-knowledge based system, whereby a graph representation of knowledge is shown to be identical to another graph with different vertex names. The problem with this approach is its inefficiency: proving the two graphs are an isomorphism of one another involves examining all possible combinations.   The other approach commonly used for granting permissions involves secret-key encryption, but it is expensive to implement.

Other work besides [4], [11] studies Pareto optimal potential games: the work by Bauso [13] examines this in a network resource sharing game. Bauso builds primarily upon Rosenthal [14] for potential games. But, like [11], he does not build upon Shapley [2], [6] despite the later publication date; this in turn causes Bauso's results to be sub-optimal. Specifically, Bauso examines pure-strategy noncooperative non-repeated (one-shot) and repeated games. Bauso correctly uses argmax to refine his answer and select the best equilibrium, but again like [4], [11] also uses it incorrectly to do an empirical proof for Pareto optimality in one-shot games. This is not all: by overlooking Shapley [2], [6], Bauso's work fails to find a Pareto optimal solution for coalitions and repeated games. Furthermore, Bauso's coalition formation algorithm does not work. While Bauso's game is partially correct in that it uses the Finite Improvement Property, according to Shapley [2] the Finite Improvement Property does not guarantee that a game possess a potential function. Rather, as stated above, the game must also have a finite end with unique moves and finite utility. What Bauso did with a repeated game possessing a potential function is violate its requirements according to Shapley [2]: the game is repeated and so each time it is restarted and played again, there exists at least one state that is not unique as it was played identically the last time around.

Like Komali [4], Johari [1] develops interesting solutions using noncooperative games represented by a graph, but avoids potential functions altogether. Johari's work addresses energy conservation for message passing in networks, and uses both noncooperative and cooperative games. In these games, all strategies are pure, and all nodes homogeneous, but unlike Komali uses the idea of bids between nodes to establish connections between nodes. Bidding takes place using a type of currency and occurs in sequence from one player to the next. Johari found several types of simple topologies that result when game theory is applied to graphs for the purpose of reaching some type of equilibrium. If the cost is applied only to the initiator of a connection, as characterized by noncooperative games, the star topology has the least cost to all nodes and will form. Otherwise, if cost is applied to both parties, as in the case of a cooperative game, a simple cycle or wheel forms. The simple cycle was also found by Johari to be a degenerative topology: soon after the network formed by the noncooperative game degenerated from a star to a simple cycle, the network disconnected.

While the above work is significant with respect to the game theoretic aspect to our security optimization solution, it does not address security. All previous game theoretic work with security deals with individual elements or components in the network. Only a few authors have examined using game theory to help solve security problems, and none of these authors have examined optimizing overall security, let alone overall security in a network. In addition, we are not aware of any previous game theoretic security optimization that validates its work using security metrics.

For the little work that has been done in the area of game theory and security, the primary work has been done by Agah [26] - [27] to model a scenario where an attacker drops packets to cause communication disruption in a network. Here, game players represent nodes in the network, and players receive utility based on the number of successfully transferred packets. As a method of defense and attacker detection, each player individually examines the percentage of its own message packets not forwarded by other players. If another player fails to forward a percentage of messages beyond a threshold, then that other player is removed from its network. All other work in the area [28] - [29] follows a similar pattern.

The work by Demirbas [29] is similar in that it also examines network games that optimize security for each component. However, their work focuses more on reconfiguring individual nodes to stabilize the network so that it may address the problem of defending against a direct attack against each individual player. The third significant work with game theory and security is done by Sun [28], who uses game theory with security and economics at the component level. Each individual part of a network is examined, and cost/benefit analysis is done with respect to additional investment. Results showed that the cost inflicted by an attacker on each component is directly reduced by the security investment. While interesting in a broad sense, the results are generic and do not list any particular types of investment, but includes every possibility in one category.

Recent work in the area of security metrics has examined the shortfall caused by generic security definitions. Security metrics are based on the idea of going beyond traditional qualitative security definitions into the area of quantitative measurement and analysis. Past definitions of security include such phrases as “works as expected” [31]. Such qualitative definitions, while traditional in security, depend upon perceptions of environment; they do not preclude the possibility of hidden vulnerabilities or sleeper agents that fail to affect the expected workings of a system until they are activated and cause destruction, illustrating that the system was never secure even though, up until that point, it did indeed work “as expected” [31]. Preventing such types of attacks is one of the purposes of quantitative security metrics.

The earliest work on building a foundation for quantitative security analysis based on security metrics that received attention is by Almerhag and Woodward in 2005 [10]. The paper describes security levels related to the types of operations or rules of composition used to define them; these three levels from least important to most important are neighbor authentication, cryptography, and access control. Authentication is measured by whether or not a neighboring node computer’s identity is able to be verified by a trusted third party in a network. Cryptography is expensive to implement, but prevents a transmission from being altered or decoded by an unwanted third party. Access control is cheaper to implement than cryptography, and works to directly stop individuals from gaining access to unauthorized information, such as user IDs and passwords. Payne [23] of the SANS Institute has a somewhat different definition of security metrics, stating that its core foundation lies in analysis of security data. Here, the essence of effective security analysis is risk analysis, with three core areas: asset value, threat, and vulnerability.

Another technique that can be used to quantitatively analyze security is attack trees, which are sometimes called attack graphs. Attack trees are a data structure with nodes representing states of the environment; transitions exist between states. These transitions represent the conditions for moving from one state to another, much like a finite state automaton. Authors Sheyner [9] and Wang [25] have called for the need to quantitatively analyze security using attack trees, but their work lacks a means of quantitatively measuring security for the purpose of attack tree analysis; still, their work is one more step toward finding a means of tying this quantitative analysis to a practical security implementation. Existing approaches using attack trees, such as the work by [40], are representative of the implementation problems attack trees pose, as these works' analyses degenerate into an NP-complete problem of analyzing all connections between all nodes in a tree. In fact, much of the work in the area of attack trees, exemplified by [39], are on finding new ways to improve speedup for analyzing this NP-complete problem. This work by [39] uses a backtracking and sorting of the relationships between nodes in the attack tree prior to any actual analysis taking place.

As stated earlier in Chapter I, our thesis will use constraint satisfaction problems as a means to better describe our problem and tie metrics to our game theoretic solution. The idea of constraints in games themselves goes back to the coalitions of games possessing information, capacity, participation, and incentive compatibility constraints. Information constraints are limitations imposed by a game designer on the players, whereby they have limited information about some aspect of the game, be it observing or predicting other players' moves. As a result, information asymmetry between players results in these games and players must make decisions accordingly. A classic example of information constraints is the Prisoners' dilemma. The second constraint type, capacity, is generally tied to games involving money or currency, possibly in a generic sense, and the limited capacity players have to spend or produce to maximize their own utility while minimizing cost. The third constraint type, participation constraints, is imposed on players to take part in the game. Last, incentive and compatibility constraints are related to one another. They are used in repeated games describing participation incentives; one example is in a game involving players that can produce products of varying quality to sell, but because other players know that low-quality products are more likely to be defective but cheaper and high-quality products are less likely to be defective but more expensive, players know that they must produce high-quality products to have a chance to generate a profit. As a result, discounts become the focus of the game to maximize utility.

A constraint satisfaction problem is an AI technique that traditionally has nothing to do with game theory. Constraint satisfaction problems models variables' constraints and variables in a problem, and can be generalized and solved by several different methods. These methods are backtracking or other tree-pruning techniques. The work by Vickrey [16] was the first to examine the relationship between constraint satisfaction problems and games with complete information. Players in the game were represented by variables in the constraint satisfaction problem, and constraints were used as means of socially forcing players in the game to follow the strategy of other players in their local neighborhood. Constraints in this work were restricted to unary constraints, and it was thus limited by the inability to apply the constraints to any binary mathematical operations.

The work by Soni, Singh, and Wellman [7], however, does apply to binary constraints. It builds upon the work by Vickrey, examining repeated cooperative games with complete information for message-passing optimization, and reformulating the constraint satisfaction problem to allow binary and unary constraints. To do so, the problem in [7] was changed to take place within a graph so that constraints existed between each variable; constraints were related to equilibrium. In this, a constraint was satisfied if the player chose the best strategy to maximize its utility given the other players in the game also choose the best move. Each player, or variable, had its own unary constraint. While their work is significant because it is, as far as we are aware, the first to examine the relationship between constraint satisfaction problems and game theory, the solution is restricted to traditional constraint satisfaction problem solving techniques and can only apply to homogeneous players with pure strategies. The solution uses the authors’ own variation of a backtracking and tree-pruning algorithm from [16], the foundation of which is a classic AI technique. The work does incorporate game theoretic means of determining tree pruning, but this is secondary to the authors’ own backtracking algorithm; potential functions and optimization are not considered.

As mentioned in the introduction, we will compare our algorithm’s security optimization with a known algorithm; the one used will be Kerberos. Again, we refer the reader to [35] for a complete description. In the body of research with comparing new network security algorithms to Kerberos, there have been varying approaches to simulating Kerberos as exemplified by the more recent works of [36] – [38]. Older research in the field tends to focus on using off-the-shelf software packages to simulate Kerberos as well as the authors' own algorithms, while newer researchers develop their own model to study specific aspects of Kerberos compared to their own work. In the following more widely-referenced recent articles, each explores simulating and modifying a Kerberos system. The work by [36], which was published in 2007, apparently marks the beginning of the end of off-the-shelf software use for Kerberos simulations. Here, the authors used the popular GSS-API software package to simulate Kerberos and their own algorithm, only to discover that the accuracy and credibility of anyone’s results from the GSS-API software had come into question by the international community while the authors (and others) were conducting experiments [36]. The authors of [36] discontinued and abandoned their experiments with GSS-API, publishing only their formal model and algorithm with a note about the fate of GSS-API and its discontinued use by the international community. In [37], which was published in the following year, the authors compare computation time of their cryptography model with their own representation of standard Kerberos cryptography by using a different off-the-shelf software package called Crypto++. One of the most recent works is by [38], a 2009 journal article in which the authors abandon the off-the-shelf software approach and instead focus on building a mathematical model and prototype. These authors model the Kerberos network through an equation derived from fluid mechanics to study bandwidth usage in networks; this equation follows similar methodology as the equation used to represent their own network optimization technique.

CHAPTER III

SECURITY METRICS

In this chapter we begin to propose our game theoretic approach to optimize overall security for a heterogeneous network. As far as we are aware, our thesis is the first in the area of optimizing overall network security using game theory. Security, as mentioned earlier in Chapter II, suffers from what we believe to be an insufficiently descriptive definition. While many authors have had difficulty defining what security is, we believe it is relatively easy to define what security is not. Security is not vulnerability, whereby an outsider may destroy, alter, or steal one's possessions. Possessions are, in the case of our network, the data stored in the computers, the computers themselves, and the connections between computers in the network. We assume that maintaining control of possessions relates to access control, which is itself controlled by user IDs and passwords. Such data is sensitive and should be guarded carefully. Insensitive data is less important, but is still vulnerable to theft. There still exists the possibility in any network that an outsider may intercept or alter data transferred between computers in the network. These instances involving alteration, destruction, or theft of possessions characterize a vulnerable network. Since a vulnerable network is clearly not a secure one, we will define security in terms of what it is not, namely vulnerability. Vulnerability will thus be used as a foundation for our security definition, and network security will be optimized by minimizing vulnerability.

To solve the problem of optimizing security, we must first quantitatively measure security so that one can determine whether or not the security has improved. Measurement of security in the system, and not just an individual component, is essential if our proposed approach to security optimization is to be validated. To do so, we build upon existing work in the area of security metrics to come up with our own metrics for quantitative security measurement. Metrics will enable us to more fully describe the problem of security maximization and enable us to quantitatively measure security itself in a meaningful way. Using an approach that allows entities in the network to evaluate their environment without having to resort to qualitative, nebulous definitions could significantly improve evaluation of security and security itself. We propose the use of security metrics as a basis for quantifying security and validating our results. We will develop our metrics, which can enable us to take definitions of security that are typically qualitative and quantify them mathematically.

Attack trees are a means of analyzing security. Attack tree analysis is not game theoretic, but is a way of describing the means by which relationships and vulnerabilities in an environment are analyzed for decision-making. In our problem, attack trees enable analysis to take the action that maximizes security. Since actions involve changing the value of a variable, and the context of our problem is security, we will see that actions involve changing the values of the security levels between nodes. We develop a methodology for applying metrics for measuring security. Since the metrics quantitatively measure security, these can be used as parameters to the analysis. In an attack tree, the tree root represents the computer whose vulnerability is being assessed.

In the tree, if another node or relationship is closer to root it has more access, which can result in greater vulnerability.   It is clearly preferred to address a problem further away from the root rather than allowing it to reach levels closer to the root before addressing it.

In an ideal world, each computer in the network would be able to implement the maximum possible security. However, we are attempting to optimize security for a heterogeneous network in the world of the practical; there are limitations which prevent the ideal from taking place. The heterogeneity of the computers forming the network prevents them from reaching the same maximum security and places limitations, or constraints, with regard to maximum security of the overall network. Since all devices in the network under consideration are different from one another, there are constraints with respect to available hardware and software, as well as corresponding constraints of the hardware and software itself that limit maximum achievable security. We believe that optimizing security for this type of environment fits the definition of, and at the very least lends itself to, a constraint satisfaction problem. Constraint satisfaction problems have variables, corresponding constraints on the variables, and problem state. A state of the problem represents the current values of some or all variables. Some constraint satisfaction problems have an objective function maximized by all variables; doing so leads to its solution.

Thus we will describe the network as a constraint satisfaction problem so that we may better specify the problem of optimizing security within the security limitations. Since we propose the application of game theory to optimize security, we will solve our constraint satisfaction problem using a game. We propose players will use environment data, represented by variables quantified by security metrics, as input to players' attack tree analyses to determine vulnerability and thereby measure security. In doing so, players may choose the move that maximizes security. As connections change in the network during game play, metrics can be used to measure the changes in the network for attack tree analysis, choice of optimal move, and overall network security.

But not all constraint satisfaction problems possess an objective function and can be directly correlated with, and thus be solved by, games. Conversely, not all games can be directly correlated with, and thus solve, the corresponding subset of constraint satisfaction problems. We must identify which constraint satisfaction problem subset can solve a subset of games that fits our problem.   Hence we propose to identify the subset of constraint satisfaction problems that can be directly used with a subset of games that fit our security problem.

To implement more effective security we will use coalitions. Coalition formation involves separation of nodes according to preferences, which are represented by current assignment of values to security levels of all connections. To give an example, complete preferences for coalition formation takes into account a node *i* and its links to neighboring nodes. Partial preferences, which are also known as incomplete preferences, take into account the links not made by node *i* but have been made by other coalition nodes *j* and *k*. Partial preferences also include the state of links prior to final optimization. Coalitions are usually formed only once all variables, or preferences, have had their final values assigned to them. To integrate the coalition formation process directly into our security optimization algorithm, we propose using coalition formation with partial preferences. Approaches to coalition formation relying on complete preferences would not be able to be directly integrated into the optimization process itself, as they are instead formed after the optimization is complete, which can place a system in a suboptimal state. Through the above solutions, we propose our work presents a novel optimization technique that improves overall security for a heterogeneous network.

3.1 Problem definition

Before examining how to quantitatively measure and optimize security, let us first define the problem, beginning with the network model and its notation. Assume the network whose security is to be optimized is represented by an a priori connected directed graph

$$G = (V, E) \tag{3.1}$$

with set of vertices V and set of directed edges E. Vertex v_i, an element of the set vertices, is denoted

$$v_i \in V = \{v_1, v_2, \dots , v_i , v_j , v_k , \dots , v_n\} \tag{3.2}$$

G is minimally connected with at least one edge between each of the n vertices such that there are at minimum

$$n - 1 \tag{3.3}$$

edges, and thus the set of edges is at minimum denoted

$$E = \{e_1, e_2, \dots, e_{n-1} \} \tag{3.4}$$

Furthermore, let vertices represent computer nodes in the network. Let the vertex v_i be represented by its index i

$$i = v_i \in V \tag{3.5}$$

and let vertex v_j be represented by its index j, where

$$j \neq i \tag{3.6}$$

and likewise for all the other vertices in V.

A directed edge forming a connection from i to j is written

$$e_{ij} \in E \tag{3.7}$$

We define a directed edge to be synonymous with the terms edge, link, and connection.

Likewise, an edge representing a connection from j to i is written

$$e_{ji} \in E \tag{3.8}$$

Let the sequence of all k possible security levels of e_{ij} be represented by S, an n-tuple containing a non-decreasing sequence of nonnegative real numbers, denoted

$$S = (s_1,\ s_2, \dots, s_k),\ s_1 \geq 0 \tag{3.9}$$

Let all e_{ij} have numerical values associated with them. These are commonly called edge weights; this definition of a weight from graph theory is not to be confused with the type of game described in Chapter II called a weighted potential game. Therefore, we will define the edge weight as the security level of the connection between two nodes, and henceforth refer to it as such to avoid confusion. Thus, let

$$s_{ij} \in S \tag{3.10}$$

denote security level of connection e_{ij} between nodes i and j.

Changing a security level s_{ij} will be a valid move, or action, in our security optimization game. Hence, the moves in our game are changing the security levels of each s_{ij}.

If we denote all nodes other than i as

$$-i \tag{3.11}$$

And, likewise, all nodes other than j as

$$-j \tag{3.12}$$

where

$$-j \neq i \tag{3.13}$$

We define the security levels of all direct connections node i makes to its neighbors forming the local network of node i as the union of the security of its connection to node j and its connections to nodes other than j, $-j$, denoted

$$s_{ij} \cup \ s_{i-j} \tag{3.14}$$

And if we assume for all direct connections i makes to its neighbors, we can define s_i, the security level of node i, as the weakest of all the direct connections from node i. The security level of node *i* is defined as the minimum of the union of the security levels of all direct connections node *i* makes to its neighbors, written

$$s_i = min\left(s_{ij} \cup \; s_{i-j}\right) \tag{3.15}$$

3.2 Security metrics

We shall define our metrics to measure security. These metrics will consider cryptography, data sensitivity, and access control, but exclude authentication. Authentication is excluded because, in our model, all aspects of the network are visible to all nodes. We therefore assume that any node in the network has authenticated. However, we reserve exploration of the issue of authentication and false impersonation for future work.

The following metrics have binary representations. We define the binary representation of each metric as corresponding to security strength; thus if the vulnerability is high, the metric is assigned the binary number 0.

3.3 Cryptography

Cryptography allows computers to send data to one another along e_{ij} in a way that any outsider or third-party who intercepts the data cannot read it. Encryption enables computers to implement cryptography. With weak encryption, there is a greater chance for vulnerability since it is relatively easy for a third-party intercepting messages sent along e_{ij} to use the message to gain access to a node. An example of weak encryption, which we define as low encryption, would be a connection secured using a 32-bit key. An example of strong encryption, which we define as high encryption, would be a connection secured using a 256-bit key. Let all connections e_{ij} have encryption. Current encryption strength of connection e_{ij} is represented by

$$y_{ij} = \begin{cases} 0, & low\ encryption \\ 1, & high\ encryption \end{cases} \tag{3.16}$$

Since connection e_{ij} has been defined as a directed or one-way connection from node i to node j, according to the definition of e_{ij}, $y_{ij} \neq y_{ji}$.

3.4 Data sensitivity

The data stored at each node can be either sensitive or insensitive. With sensitive data there is a greater possibility for vulnerability, such as if an intruder gained access to passwords, than with insensitive data. Sensitive data includes, for example, passwords and usernames. Insensitive data excludes sensitive data by definition. Insensitive data includes, for example, time of day, schedules, or other data that does not increase vulnerability if it is stolen. Let all nodes have data; all data of node j, is written

$$d(j) = \begin{cases} 0, & sensitive \\ 1, & insensitive \end{cases} \tag{3.17}$$

3.5 Access control

Permissions to read and write a node's data also affect security. Access to read and write data increases the vulnerability as it increases the possibility of an attacker stealing or modifying the data. Let write access from *i* to *j* be represented by

$$w_{ij}(d(j)) \tag{3.18}$$

Again, the binary number relates to security strength. Consequently, if a node *i* has write access to data of node *j*, then it indicates that decreased vulnerability is false, or 0. Thus,

$$w_{ij}(d(j)) = \begin{cases} 0, & node\ i\ has\ write\ access\ to\ d(j) \\ 1, & node\ i\ does\ not\ have\ write\ access\ to\ d(j) \end{cases} \tag{3.19}$$

Let read access from *i* to *j* be represented by

$$r_{ij}(d(j)) \tag{3.20}$$

The binary number relates to security strength. Consequently, if a node *i* has read access to data of node *j*, then it indicates that decreased vulnerability is false, or 0.

$$r_{ij}(d(j)) = \begin{cases} 0, & node\ i\ has\ read\ access\ to\ d(j) \\ 1, & node\ i\ does\ not\ have\ read\ access\ to\ d(j) \end{cases} \tag{3.21}$$

With our metrics we can define s_{ij} using security metrics

$$s_{ij} = y_{ij} \times r_{ij}\big(d(j)\big) \times w_{ij}\big(d(j)\big) \in S \tag{3.22}$$

We will use algebra for translating a Cartesian product to a real number by applying an algebraic weighted sum to achieve a clearer security level conceptualization, translating a binary Cartesian product to a security value ranging from 0 to 10. In this context, the term "weight" will always be used with the word "sum" since it refers to the algebraic weighted sum and not the game theoretic definition of weight.

For node *i* having connection to *j* we translate the security definition of s_{ij}, which is a Cartesian product, to a real number according to the formula

$$s_{ij} = \omega_\alpha \cdot y_{ij} + \omega_\beta \cdot r_{ij}\big(d(j)\big) + \omega_\gamma \cdot w_{ij}\big(d(j)\big) \qquad (3.23)$$

3.6 Security metrics example

To give an example of translating encryption, read, and write metrics to a real number according to equation (3.23), let us assign nonnegative real numbers to the coefficients of each of the metrics in the weighted sum to give a range of integer security values from 0 through 10, such that

$$S = (0, 1, 2, \dots, 10) \qquad (3.24)$$

Also, to aid in clarifying our example, we will separate the metric of data sensitivity from the read and write metrics. Thus we redefine s_{ij} as

$$s_{ij} = \omega_\alpha \cdot y_i + \omega_\beta \cdot r_i + \omega_\gamma \cdot w_i + \omega_\delta \cdot d(j) \qquad (3.25)$$

We denote the set of all security levels of all *h* direct links from node *i* to all other nodes –*i*, forming the local neighborhood of *i*, as

$$\cup_{j=j}^{h} s_{ij} \tag{3.26}$$

Thus, for example, if node i has direct connections to nodes $j, k, a, b, c,$ then

$$\bigcup_{j=j}^{h} s_{ij} = \{s_{ij}, s_{ik}, s_{ia}, s_{ib}, s_{ic}\}$$

Next we determine the values of each weight so security ranges from 0…10, chosen for convenience, according to equation (3.24). We assume that data sensitivity is foremost in determining vulnerability. Since theft of data is of little consequence when it is insensitive and of great consequence when it is sensitive, we weigh data sensitivity the most heavily. Hence we shall assign the numerical value of 5 (out of 10) to ω_δ.

Next, we consider data tampering to be the second greatest vulnerability. Data tampering can cause lost passwords or system failure if data is sensitive, and is accomplished by writing. Access control can be used to prevent an outsider from tampering with data. Encryption can also be used to prevent an outsider from tampering with data. We shall thus assign the same numerical value to the encryption and write metrics, 2 to ω_α and ω_γ. Restricting read access can prevent an outsider from viewing data, which we consider to be less of a possible vulnerability than any of the other metrics, and we shall thus assign the numerical value of 1 to ω_β.

While encryption can also prevent an outsider from reading data, we consider its ability to prevent data tampering to be more important, and so it outweighs any security consideration for an outsider reading data. Thus, the weights add up to 10, the maximum security level that we assumed:

$$\omega_\alpha + \omega_\beta + \omega_\gamma + \omega_\delta = 2 + 1 + 2 + 5 = 10 \tag{3.27}$$

And in combination with the binary representation of our metrics, security levels will range from 0 to 10. We can now draw up a table, Table 3.1, which contains the Cartesian product of security metrics converted to real values using our algebraic weighted sum in equation (3.27), to represent the weighted sum and metrics to derive the security values of s_{ij}. Note in Table 3.1, if encryption is high and read and write access is restricted, the data sensitivity is irrelevant; hence the same security value for cases 14 and 15. The same can be said of the data for cases 6 and 7, as again, read and write access is denied. We denote these "don't care" statuses using an X in the respective table cell.

	Encryption **(high)**	**Read** **(restricted)**	**Write** **(restricted)**	**Data** **(insensitive)**	**Security** ***s***
Weights **Case**	**2**	**1**	**2**	**5**	
15.	1	1	1	X	**10**
14.	1	1	1	X	**10**
13.	1	1	0	1	**8**
12.	1	1	0	0	**3**
11.	1	0	1	1	**9**
10.	1	0	1	0	**4**
9.	1	0	0	1	**7**
8.	1	0	0	0	**2**
7.	0	1	1	X	**8**
6.	0	1	1	X	**8**
5.	0	1	0	1	**6**
4.	0	1	0	0	**1**
3.	0	0	1	1	**7**
2.	0	0	1	0	**2**
1.	0	0	0	1	**5**
0.	0	0	0	0	**0**

Table 3.1: Cartesian product of security metrics converted to real values

CHAPTER IV

CONSTRAINT SATISFACTION

To describe our model in further detail, we will specify its constraints and explain how we can use it to improve representation of the requirements of our optimization problem. We will show how our problem can be modeled as a constraint satisfaction problem, which will act as an intermediate step in designing our game.

4.1 Constraint satisfaction problems

Constraints represent the limitations of security for each computer in our network, and consequently, constraints are related to the overall network security limitations. A constraint is a function that maps the domain of possible security values S to the range of allowed security values for a computer. It thus restricts the domain of security values S to a subset of S that is achievable by the node given its hardware and software. There may be a great deal of difference in the processing power and other physical computational limits of the computers forming our heterogeneous network we are optimizing in our problem. These hardware and software limitations lead to actual security limits that each computer in the network can achieve. A constraint function is thus itself a function of the hardware and software of a computer. Each kind of software provides a range of security levels, and each kind of hardware provides another range of security levels. The actual range of security values achievable by a computer is somewhat complicated, as it is related to whether the hardware and software independently provide security, or are dependent upon one another. If we consider the domain of security values as the Cartesian product of the set of all computer hardware and software, trying to enumerate all the possibilities would be difficult. The security function depends upon the relationship between hardware and software. In some situations where hardware and software are dependent upon one another for security, if one element is defeated, the other may be worthless. For example, if a security feature of the hardware is defeated, the software could be worthless for providing security. For an example of a problem (outside the scope of this work) such as laptop theft, in this case if a laptop is stolen and hard drive erased, then any software encryption of data on the hard drive is worthless. In other cases, the hardware and software may make each other stronger, such as a physical lock on an office door combined with a password on the computer in the office. Here, an attacker would have to spend time getting past the lock before spending time getting past the password on the computer. Because of the difficulty in deriving a function for all possible combinations of hardware and software, we will characterize the properties the function will have to develop a heuristic that is a reasonable model for our problem.

For our problem, we can consider our definition of security as described above using metrics as first being software-related, since encryption and access control to data are related to software. However, we may safely reason that a hand-held device such as a cell phone is capable of a lower and smaller range of security than a server due to the hardware and the software differences. Hence, we believe that for most cases related to our problem of optimizing a heterogeneous network, the hardware and software are dependent upon one another. Granted, it is possible that in certain cases our model fails to make sense; we are assuming that computers in our network will have security constraints whereby hardware and software are dependent upon one another, and that all computers being optimized can be characterized by our definitions of security and constraints.

If we first characterize unary constraints, which form an essential part of a constraint satisfaction problem, we consider that this relates to the security of one computer's hardware and software. Unary constraints pertain to the limitations on maximum achievable security of one computer, or rather one node in the network. If we say that a function f_1 takes as input the hardware of a node i, $hardware_i$, and produces a range of security values that is a subset of S, and a second function f_2 takes the software of a node i, $software_i$, and produces another range of security values that is a subset of S, and we assume that these two subsets of f_1 and f_2 overlap with values in common between them; and if we have software that worsens overall security if the software is weaker, and hardware that worsens overall security if the hardware is weaker, then we can say that the unary constraint function $\sigma_{unary}(hardware_i, software_i)$ is at least as good as the weakest of the hardware and software for node i,

$$\sigma_{unary}(hardware_i, software_i) \geq min\ (f_1(hardware_i), f_2(software_i))$$

What the security is less than or equal to is another matter. If the quality of the software and hardware are good, giving high security, then we will say that this is a best-case scenario and consequently represents the maximum security. Here, we assume that we can characterize the maximum security all the possible nodes in our network with this equation, whereby the security is less than or equal to

$$\sigma_{unary}(hardware_i, software_i) \leq max\ (f_1(hardware_i), f_2(software_i))$$

Putting these together to characterize unary constraints, we have the inequality

$$\begin{aligned} min\big(f_1(hardware_i), f_2(software_i)\big) \leq \sigma_{unary}(hardware_i, software_i) \\ \leq max\ (f_1(hardware_i), f_2(software_i)) \end{aligned} \tag{4.1}$$

There are exceptions to this model, however, because it is possible that combined hardware and software can work together to give a maximum security value that is greater than the maximum of either the hardware or software. For example, if we have a computer that uses protected memory on a CPU to prevent processes from accessing unauthorized data in combination with encryption on the data, this computer would achieve a security level beyond the maximum of encryption and protected memory, since even if a process was able to view data it was not supposed to, it would then have to get past the encryption on the data. This type of system would represent a significantly reduced vulnerability. However, we are assuming that the computers optimized stay within our model of unary constraints on security in equation (4.1).

For ease of notation and to more clearly see the node to which the unary constraint function is being applied, let us denote an abbreviation for constraints in equation (4.1) for each node *i* as

$$\sigma_i(s_i) = \sigma_{unary}\ (f_1(hardware_i), f_2(software_i)) \qquad (4.2)$$

Binary constraints are the other type of constraints in a constraint satisfaction problem, and pertain to the security value of an edge between two nodes. Here, we are dealing with constraints on a network, albeit a small network that is between two nodes. Interactions are complicated, and thus the safest assumption is to base the range of values on the weakest node forming the link. Here, the binary constraints range from the minimum of the weakest node to the maximum of the weakest node. For a connection e_{ij} from node *i* to node *j* at security s_{ij},

$$min\begin{pmatrix} min\big(f_1(hardware_i), f_2(software_i)\big)\,, \\ min\left(f_1\big(hardware_j\big), f_2\big(software_j\big)\right) \end{pmatrix} \leq$$

$$\sigma_{binary}\big(hardware_i, software_i, hardware_j, software_j\big) \leq$$

$$max\begin{pmatrix} min\big(f_1(hardware_i), f_2(software_i)\big)\,, \\ min\left(f_1\big(hardware_j\big), f_2\big(software_j\big)\right) \end{pmatrix}$$

$$(4.3)$$

We shall denote an abbreviation for binary constraints of connection e_{ij} in equation (4.3) as

$$\sigma_{ij}\big(s_{ij}\big) = \ \sigma_{binary}\big(hardware_i, software_i, hardware_j, software_j\big) \qquad (4.4)$$

Note that if the set of constraints of node *i* and the set of constraints of node *j* have something in common between them, a link can be formed between the two nodes. Mathematically, this is written as the intersection of the set of constraints of node *i* and the set of constraints of node *j*, denoted

$$\sigma_i(s_i) \cap \ \sigma_j(s_j) \neq \emptyset \tag{4.5}$$

then a link e_{ij} at s_{ij} can be formed between nodes *i* and *j*.

Because security levels have to be determined within their constraints, we denote security given its constraints as an ordered pair

$$\left(s_i, \sigma_i(s_i)\right) \tag{4.6}$$

and

$$\left(s_{ij}, \sigma_{ij}(s_{ij})\right) \tag{4.7}$$

4.2 Costs

We define cost as a way of measuring lost security. We define pay, or payment, as compensation measured in utility to or from another node. Both payment and cost are denominated in units of security. To give an example, payment from one node *i* to another node *j* for creating link e_{ij} would come in the form of security gained by *j* through *i* allocating some of its CPU time to *j*. Because granting CPU time to node *j* restricts the available resources of node *i*, it decreases available resources for *i* to implement security, but increases the available resources of *j* to implement security. The cost to *i* for making the payment would be lost security. For the connection, node *i* receives utility for the link since it is connected to the rest of the network, but node *j* will incur cost in the future from retransmitting messages sent by node *i*. Therefore, future cost to *j* is offset by payment from *i* for establishing the link. In doing so, our game fits the definition of a noncooperative game. Cost *c* of payment from *i* to *j* for forming e_{ij} shall be denoted as cost of e_{ij}, written

$$c(e_{ij}) \tag{4.8}$$

The cost *c* of payment from node *i* to node *j* to form e_{ij} is determined by a varying price scheme, which is accomplished by bidding. The bidding process is tied to strength of need to establish a link measured by the effect it has on security. If a node benefits more by establishing the link, it would be willing to pay more than if little benefit was received. Bidding schemes include iterative or linear movement from the minimum to maximum amount a node is willing to pay. This scheme allows the player to move only some fixed amount each time it raises or lowers a bid; in other words, if a bid starts at 1, its next bid is at 2, then 3, etc… Another type of bidding scheme implements an exponential movement from minimum to maximum amount a node is willing to pay. This bidding scheme follows the pattern of an exponential function, whereby bidding increments are slow initially and increase exponentially as each bid is made. Finally, there exists the possibility of nodes implementing a scheme whereby bids follow a logarithmic movement from minimum to maximum amount willing to pay.

4.3 Side payments

The calculation to decide the move that gives the most utility may indicate that it is not beneficial for that node to raise its security beyond some level. In addition to payment from node *i* to node *j* to form a link e_{ij}, we define a second type of payment called a side payment. Side payments are used to induce a node to move beyond its selfish motivations to benefit another node. A side payment is defined as an action that consists of payment made from node *i* to node *j* to induce *j* to change the security, s_{jk}, of its link e_{jk} to node *k*. If analysis shows that a node *i* gains the most utility by another node *j* altering a link to node *k*, where *i* is not connected to *k*, node *i* can take action to maximize its utility by paying node *j* to alter its link to node *k*. Adding side payments to the set of possible moves has the possibility of raising the security level of *i* as well as other nodes connected to *j* beyond what would be possible without a system of side payments, swaying the utility calculation of an individual node in favor of distributed security over local security, therefore benefiting the network as a whole. We denote a side payment action from node *i* to another node *j* to induce action $\left(s_{jk}, \sigma_{jk}(s_{jk})\right)$ that changes security of *j*'s connection to node *k* as

$$c_{ij}\left(s_{jk}, \sigma_{jk}(s_{jk})\right) \tag{4.9}$$

Since a side payment is an action node *i* takes that changes a link, which in this case is altering link s_{jk}, it is considered to be a move for node *i*. As such, it must take place within the constraints of *i*, $\sigma_i(s_i)$. Since cost for the action is applied to *i* since *i* must pay *j* to take action altering link s_{jk}, cost *c* is additionally notated for clarity to indicate that cost is applied to *i* as it must pay *j* to take the action. We assume that *j* cannot refuse an action once it agrees to take the action.

By having nodes pay other nodes to make changes in security that do not affect the other nodes directly but can cause them to become more vulnerable, a type of cooperation takes place. Side payments act as a facilitator of a form of cooperation, regardless of coalition membership, to increase overall security. The end result of this payment is that overall security of the nodes along the attack tree from payee to payer is increased. Without side payments between nodes, there may be areas in the graph or particular nodes for which this solution is sub-optimal. This sub-optimality can be brought on by individual nodes' strategies or security limits due to the computers' constraints. However, we believe side payments help in avoiding these sub-optimal scenarios in a network.

Besides helping form links, payments between players denominated in the same unit of measure as utility aid in applying attack tree analysis to improve security via side payment. For the above scenario for nodes *i, j,* and *k,* if a node *i* wishes to implement a change between nodes *j* and *k,* it corresponds to making a change further down the attack tree. This is cheaper than implementing changes higher up in the tree; changes further down (node *k*) correspond to earlier changes versus later changes when the problem is imminent (node *j*) and vulnerability is greater. As a result, not only would it be cheaper to make earlier changes or fixes in the tree, more nodes (such as other nodes connected to node *j*) can benefit from these changes. Such a payment should only make sense in the context of utility: node *i* that received benefit should ensure the node *j* that made the change also benefit, especially when one considers that the node *j* making the change may not directly benefit from the change. Carrying out attack tree analyzed security changes through side payments enables the cooperation among the nodes. Algorithm 5.2 addresses how nodes perform attack tree analysis.

4.4 Objective function

The objective function is a function maximized in a constraint satisfaction problem. Since we have not yet specified all aspects of our approach, describing the objective function beyond any general notation would be premature at this point. We thus say that the objective function, *u*, is used by all nodes to maximize security given its constraints.

4.5 Constraint satisfaction problems and game subset solvability

The use of constraint satisfaction has aided in specifying our solution in more detail, but we must identify a subset of games that work with a constraint satisfaction problem subset for mathematical proof of optimization. This is achieved, explained in detail below, through a noncooperative potential game and constraint satisfaction problem possessing an objective function. Our reasoning is explained in Proof 4.1 below.

Theorem 4.1:

A potential game that maximizes security can be used to solve a constraint satisfaction problem to maximize security if-and-only if the constraint satisfaction problem has an objective function that maximizes security.

Proof 4.1 Proof of Theorem 4.1

Proving this Theorem 4.1 by contradiction is inappropriate as it creates a logical in-equivalence that is difficult to resolve due to the use of negation with *if-and-only-if*. Thus, we will use a direct proof.

We will represent the statement in Theorem 4.1 symbolically.

p: game with potential function

q: constraint satisfaction problem with an objective function

Since a constraint satisfaction problem with an objective function by definition maximizes the function per its input, let us define input as security value according to equation (3.9) as

$$s \in S$$

And also define the objective function of the constraint satisfaction problem as maximization of *s* given constraint $\sigma(s)$

$$max\big(s, \sigma(s)\big)$$

Statement *q* is thus written symbolically

$$q\colon max(s, \sigma(s))$$

Each player has its own variable that it maximizes, which we shall also call *s* because we are examining optimizing the same security domain as the constraint satisfaction problem. The domains of *p* and *q* are the same. Since a potential function by definition represents a function to be maximized by all parties, it is written

$$max(s)$$

But if we consider that there is a corresponding function that restricts *s* to an allowed set of values that map the domain to a range of allowed values

$$\sigma(s)$$

Then we are using the maximization function

$$max(s, \sigma(s))$$

which is the potential function.

Since

$$max\big(s, \sigma(s)\big) = \; max\big(s, \sigma(s)\big)$$

the objective and potential functions are equivalent and we can write statement *p* as

$$p\colon max(s, \sigma(s))$$

Next we need to prove

$$\exists p \;, \exists q \quad p \leftrightarrow q$$

Since we are not trying to prove that all potential games solve all constraint satisfaction problems with objective functions, but that a constraint satisfaction problem with an objective function is solvable by a game with a potential function addressing the same problem, we thus have

$$\exists p\ , \exists q \quad p \leftrightarrow q$$

Where

p: game with a potential function

q: constraint satisfaction problem with an objective function

With the above, we have

$$p\colon max(s, \sigma(s))$$

which, as before, is a potential function. We also have a constraint satisfaction problem with objective function, which as stated above is written

$$q\colon max(s, \sigma(s))$$

For the case of an exact potential function, the potential function is equivalent to the utility function u,

$$u(s, \sigma(s)) = max(s, \sigma(s))$$

Then we do not need to break p into sub-statements describing $u(s, \sigma(s))$ and $max(s, \sigma(s))$ since they are equivalent. We can now write symbolically what we are trying to prove as

$$\exists p\ ,\exists q \quad p \leftrightarrow q \equiv\ max(s,\sigma(s))\ \leftrightarrow max(s,\sigma(s))$$

which gives, using a truth table to examine logical equivalence for the exact potential function written above

	$\boldsymbol{p}$	$\boldsymbol{q}$	$\boldsymbol{p \leftrightarrow q}$
3.	T	T	T
2.	T	F	F
1.	F	T	F
0.	F	F	T

All but case 3 above are vacuous to some degree or are false.

We must now prove true for an ordinal potential function and objective function. We have

p: game with a potential function

q: constraint satisfaction problem with an objective function

where

$$p{:}\, max(s,\sigma(s))$$

and constraint satisfaction problem with objective function

$$q\colon max(s, \sigma(s))$$

Unlike exact potential, an ordinal potential function by definition does not have equality with utility. Instead, the potential function is greater than or equal to zero if-and-only-if the utility function is greater than or equal to zero, meaning *p* is actually written as

$$p\colon max(s, \sigma(s)) \geq 0 \leftrightarrow u(s, \sigma(s)) \geq 0$$

Unlike the proof for exact potential, where $u(s, \sigma(s)) = max(s, \sigma(s))$, we must break statement *p* into sub-statements. Since the sub-statements are not equal but are the same sign (positive or negative) if-and-only-if the other is the same sign, we must first construct a truth table to prove this before moving on to the larger proof for an ordinal potential function and objective function. Let us represent statement *p* as the sub-statements

$$r\colon u(s, \sigma(s)) \geq 0$$

$$t\colon max(s, \sigma(s)) \geq 0$$

Then *p* can be written as

$$\exists r\ , \exists s \quad r \leftrightarrow s$$

From the definition of *p*, *q*, *r*, and *s*, we can write what we are trying to prove as

$$\exists q\ , \exists r, \exists t \quad (r \leftrightarrow t) \leftrightarrow q$$

which gives, using a truth table to examine logical equivalence

	q	r	t	$(r \leftrightarrow t)$	$(r \leftrightarrow t) \leftrightarrow q$
7.	T	T	T	T	T
6.	T	T	F	F	F
5.	T	F	T	F	F
4.	T	F	F	T	T
3.	F	T	T	T	F
2.	F	T	F	F	T
1.	F	F	T	F	T
0.	F	F	F	T	F

We thus have four cases in the truth table where the result is true. Since cases 1, 2, and 4 are vacuous, but case 7 is valid when all exist, we drop the vacuous cases since they are not applicable. Case 7, however, is true and also valid for our problem definition. Thus because we proved true for the ordinal and exact potential functions, we can conclude that only potential games can be used to solve a constraint satisfaction problem if-and-only if the constraint satisfaction problem has an objective function.

Q.E.D.

4.6 Utility

Because utility is measured in security, choice of the action that gives maximum utility is the choice that maximizes security. To maximize security, node *i* must analyze the security of existing connections to itself as well as those to its neighbors to which it is directly connected, in order to determine whether connections should be modified. In addition, a node also estimates whether making a new connection or making a side payment to alter a connection would be the best decision to maximize security. We have already denoted the security levels of all existing connections. With their constraints this is denoted

$$\left(s_{ij}, \sigma_{ij}(s_{ij})\right) \cup \left(s_{i-j}, \sigma_{i-j}(s_{i-j})\right) \tag{4.10}$$

The security levels of non-existing connections from *i*, with their constraints, shall be denoted

$$\left(s_{-ij}, \sigma_{-ij}(s_{-ij})\ \right) \tag{4.11}$$

The utility function is a mathematical function used by all players to make an optimal decision at each point in a game. The utility function allows a player to choose the best values that can be assigned to the game variables and maximize its reward, or utility, at each turn in the game. Utility in our game is measured in security. Since the purpose of our work is optimizing security, and any game theoretic solution optimizes utility, utility is measured in terms of security. As game variables are represented by all s_{ij}, then the utility function allows a player to choose the action that gives maximum security. Since actions involve changing security levels of connections, for connection e_{ij} the utility function u is defined as

$$u\left(s_{ij}, \sigma_{ij}(s_{ij})\right) = \left(s_{ij}, \sigma_{ij}(s_{ij})\right) - c(e_{ij}) \tag{4.12}$$

4.7 Equilibrium in a game

The optimal action s_i^* for node i given its constraints $\sigma_i(s_i^*)$ is to choose the security level that maximizes equations (4.10) – (4.12), denoted

$$(s_i^*, \sigma_i(s_i^*)) = max\left(\left(s_{ij}, \sigma_{ij}(s_{ij})\right) \cup \left(s_{i-j}, \sigma_{i-j}(s_{i-j})\right) \cup \left(s_{-ij}, \sigma_{-ij}(s_{-ij})\right)\right) \tag{4.13}$$

The rationale for decision-making to maximize security is made according to Algorithm 5.2, which uses attack tree analysis.

The utility received for this optimal action is denoted

$$u(s_i^*, \sigma_i(s_i^*)) \tag{4.14}$$

An action profile is a sequence containing each player's move at that particular turn in the game. In some game-theoretic literature an action profile is referred to as a tuple or vector; this type of vector is not one with magnitude or direction, but instead refers to a row or column in a matrix. Hence we refer to this sequence of actions as simply an action profile in order to avoid confusion. This is not to be confused with a strategy, which is an action plan. An action profile is in equilibrium if it contains the best, or optimal, move for each player versus all other best actions of the other players. An action profile in equilibrium, denoted *s**, is written

$$s^* = \left((s_1^*, \sigma_1(s_1^*)), (s_2^*, \sigma_2(s_2^*)), \dots, (s_n^*, \sigma_n(s_n^*))\right) \tag{4.15}$$

In this equation, for example, $(s_1^*, \sigma_1(s_1^*))$ refers to the equilibrium action of player 1 given its constraints, and $(s_2^*, \sigma_2(s_2^*))$ refers to the equilibrium action of player 2 given that player's constraints.

Recall that actions involve changing security levels of links to other nodes. For example, if we assume best action for player *i* is forming a connection to player *j* at security level s_{ij}, then in this case $(s_i^*, \sigma_i(s_i^*))$ refers to

$$\left(s_{ij}, \sigma_{ij}(s_{ij})\right)$$

which involves bidding to establish or change the security s_{ij} of e_{ij}.

To give another example, it is also possible that since a side payment is an action which does not involve bidding but alters a connection, it can be an optimal action s_i* for node *i*. In this case, if node *i* is directly connected to node *j* but not directly connected to node *k*, if the side payment to *j* to have *j* change s_{jk} is calculated to be the optimal action $(s_i^*, \sigma_i(s_i^*))$, then in this case $(s_i^*, \sigma_i(s_i^*))$ refers to

$$c_{ij}\left(s_{jk}, \sigma_{jk}(s_{jk})\right)$$

The definition of equilibrium for player *i* means that its utility, or security, is maximized. No player has any incentive to change its action given no other player changes its action. An action that is not in equilibrium is a sub-optimal action. A sub-optimal action is defined, given its constraints, for player *i* as

$$(s_i', \sigma_i(s_i')) \tag{4.16}$$

where the sub-optimal action is never the equilibrium action, denoted

$$(s_i', \sigma_i(s_i')) \neq (s_i^*, \sigma_i(s_i^*)) \tag{4.17}$$

The following equation describes the best action, or equilibrium, of player *i*; security of *i* is maximized if it takes the best action when all other players take their best action, thereby maximizing the utility function *u*, denoted

$$u\big((s_i^*, \sigma_i(s_i^*)), (s_{-i}^*, \sigma_{-i}(s_{-i}^*))\big) \geq u\big((s_i', \sigma_i(s_i')), (s_{-i}, \sigma_{-i}(s_{-i}))\big) \tag{4.18}$$

CHAPTER V

GAME THEORETIC ANALYSIS

The best action a node can take is determined by attack tree analysis. However, before examining attack tree analysis to determine the best action, we need to address the reasoning behind the decision-making for granting other nodes access to data, which forms an essential component of attack tree analysis. Read and write access form two of the three security metrics. We have not specified whether or not access is granted on a case-by-case basis, and if it is, we need to address the reasons behind granting access. Addressing this raises the need for a refined security definition and analysis algorithm, which will be made possible through coalitions.

5.1 Coalition formation for a weighted potential game

Coalitions have the advantage of allowing aggregated or broad security levels and access control, which will be made possible in combination with security metrics and attack tree analysis. Without coalitions there is a less efficient method of analysis than is available with them. Without coalitions, the reasoning for nodes granting access to its own data is made on a case-by-case basis. To form coalitions, the network is divided into non-overlapping subsets according to a pairwise disjoint function known as a partition.

Coalition formation involves using the partition to combine preferences from individual nodes into a common coalition with one set of common constraints and preferences. First, we must define what we mean by preferences. Preferences are defined as the current best values for the variables in the game, which in other words is the current optimal security levels s_{ij} of the connections in the graph. They correspond to the environment and its current state. In our model these are the current values assigned to each variable within its constraints. Coalition formation can take place during the game, when preferences are being formed prior to stabilization. Otherwise, waiting for stabilization of preferences, which corresponds to final value assignment to all variables in the game once optimization is complete, could have the effect of placing the game into a non-equilibrium or non-Pareto optimal state.

Integrating the coalition formation process into our optimization algorithm without disturbing the optimization requires that coalitions be formed despite partial or incomplete preferences. Partial preferences in our algorithm are handled by the evaluation criteria for coalition formation, as well as the definition security s_i for each node *i*. Instead of treating a node as just a sum of its links to other nodes, security s_i of each node *i* is evaluated to determine coalition membership. Although nodes of different coalitions can communicate, each node can belong to only one coalition. Our definition of security s_i and its constraints includes criteria that indirectly takes into account security levels and constraints of other nodes connected to *i*, in which the nodes have shared preferences and privileges. Since s_i is defined as the minimum of all direct connections from node *i* to its neighbors, and each connection s_{ij} takes place within its constraints which include constraints of both *i* and *j*, our definition of s_i is related to the constraints of the other node *j*.

In combination with the definition of link formation, whereby both nodes in a link receive benefit, the definition of s_i will cause our game to be a weighted potential game. Because s_i takes into account the preferences of node *i* and all other nodes $-i$ to which it is connected, the utility of node *i* is directly related to the utility of these other nodes. In addition, the partition that evaluates s_i for coalition membership must consider the preferences, in our case partial preferences, of both *i* and the other nodes to which *i* is connected in the coalition. Despite the partitioning of nodes into non-overlapping subsets, the graph *G* of our network is connected because graphs of weighted potential games are, by definition, connected. Although each node can belong to only one coalition, in which the nodes have more access privileges to each other, nodes of different coalitions can still communicate.

Coalitions can be used with constraints and metrics to form a type of access control list, which makes coalitions an essential part of security metrics. Once coalitions are formed, there is a common set of constraints for members of the coalition. For example, members of the coalition might have read access to all other members, or write access. Coalition membership can be revoked if a node *i* changes s_i to violate coalition constraints σ_A which are a range of security levels for each member to belong to the coalition. The node is then removed from the coalition and all rights as a coalition member are revoked. The node can choose to change s_i or try to join another coalition.

Algorithm 5.1: Coalition formation

Algorithm 5.1 divides nodes into coalitions according to security. To form coalitions, the network is divided into non-overlapping subsets according to a pairwise disjoint function known as a partition. Each subset has a minimum security level which is a function of the partition. To do so, it applies the partition *A* to each node's security level s_i and s_{ij} to place each node into a subset, or coalition, and thus determine coalition membership. The partition *A* is a threshold function, which we assume has to be calculated, that gives the maximum security difference between nodes in *G* that can be members of the same coalition. The partition gives coalition constraints σ_A which are a range of security levels for each member to belong to the coalition. The partition ensures that any messages sent through the coalition are at that coalition's minimum level of security. Note that according to the definitions of security, link formation, utility, and potential, membership in a coalition according to the partition does not prevent a node from connecting to another node outside its coalition, as the determination as to whether to connect to another node is done according to these equations.

Define *A* as a pairwise disjoint function, or partition, dividing *G* into non-overlapping subsets; these subsets are coalitions on $A = \{\{a_1\}, \{a_2\}, \ldots, \{a_\kappa\}, \ldots, \{a_\eta\}\}$. Note that according to the definition of a partition that $\cup A = G$, and each coalition $\{a_\kappa\}$ may contain more than one node, e.g. $|\{a_\kappa\}| \geq 1$. The number of coalitions formed is dependent on the partition chosen for *A*.

Algorithm 5.1: Coalition formation:

Input*:* $G = (V, E)$,
partition A,
difference_threshold, security tolerance for coalition membership
Output: set of coalitions AQ for each node $i \in V$

```
Coalition_formation(G, A, difference_threshold)
for each node $i \in V$
{
  for each node $j = -i \in V$
  {
        if ( $e_{ij}$ )
        {
              $Q = A(s_{ij})$
              if ( $Q \leq$ difference_threshold )
              {
                    Add i to same coalition as j:
                        $AQ = \{\{i, j\}\}$
              }
              else
              {
                    Add i to different coalition from j:
                        $AQ = \{\{i\}, \{j\}\}$
               }
        }
        if ( ! $e_{ij}$ || $s_i < s_{ij}$ )
        {
              $Q = A(s_i)$
              if ( $Q \leq$ difference_threshold )
                    Add i to same coalition as j: $AQ = \{\{i, j\}\}$
              else
                    Add i to different coalition from j: $AQ = \{\{i\}, \{j\}\}$
         }
    }
}
return (AQ)
```

The advantage of our coalition formation algorithm is, by introducing coalitions, security of connections can be better described. We use a table, shown below in Table 5.1, as an example to show the effect of coalitions on security. Contrast this table with Tables 5.2 and 5.3, which reflects changes made to the security when coalitions are removed; note the less descriptive security characterized by the data in the Table 5.3.

	Level	**Security for connection**	**Read self data**	**Write self data**
Low	1	Minimum	All	All
	2	Maximum	All	All
	3	Minimum	All	None
	4	Maximum	All	None
	5	Minimum	Coalition	Coalition
	6	Maximum	Coalition	Coalition
	7	Minimum	Coalition	None
	8	Maximum	Coalition	None
	9	Minimum	None	None
High	10	Maximum	None	None

Table 5.1: Security levels with coalitions

In the absence of coalitions, security levels are less precise: coalitions are an essential part of fulfilling the requirements for a secure network per the security metrics. Changes to security when coalitions are removed are shown in Table 5.2.

	Level	**Security for connection**	**Read self data**	**Write self data**
Low	1	Minimum	All	All
	2	Maximum	All	All
	3	Minimum	All	None
	4	Maximum	All	None
	~~5~~	~~Minimum~~	~~Coalition~~	~~Coalition~~
	~~6~~	~~Maximum~~	~~Coalition~~	~~Coalition~~
	~~7~~	~~Minimum~~	~~Coalition~~	~~None~~
	~~8~~	~~Maximum~~	~~Coalition~~	~~None~~
	9	Minimum	None	None
High	10	Maximum	None	None

Table 5.2: Security levels and changes brought about by removing coalitions

Comparing Tables 5.1, 5.2, and 5.3, we see the changes that occur to the security definition if coalitions are removed. The example presented by Table 5.3 below reflects this, as quantization of security is less precise, and shows the final result of changes from Table 5.1 to Table 5.2.

	Level	**Security for connection**	**Read self data**	**Write self data**
Low	1	Minimum	All	All
	2	Maximum	All	All
	3	Minimum	All	None
	4	Maximum	All	None
	5 (*was 9*)	Minimum	None	None
High	6 (*was 10*)	Maximum	None	None

Table 5.3: Non-coalition security levels

5.2 Attack tree analysis to improve security

Attack trees model inter-relatedness of players' securities, which we will use for decision-making regarding connections. Attack trees can identify sensitive and insensitive data and can allow a node or a coalition of nodes to identify and fix security hazards, such as access to sensitive data by an intruder. In some cases it may be beneficial for a node performing attack tree analysis, node *i*, to pay another node *j* at further distance away in the tree to address a situation that has little or no bearing to the further node *j*, but could lead to the node *i's* compromise by an attacker. Quantization of payment in terms of utility has the means to overcome any selfish calculations by a node *j* closer to some other node *i*'s problem, and induce that node *j* to help the other node *i* by taking action to solve a security issue related to *i*. Algorithm 5.2 addresses how nodes perform attack tree analysis.

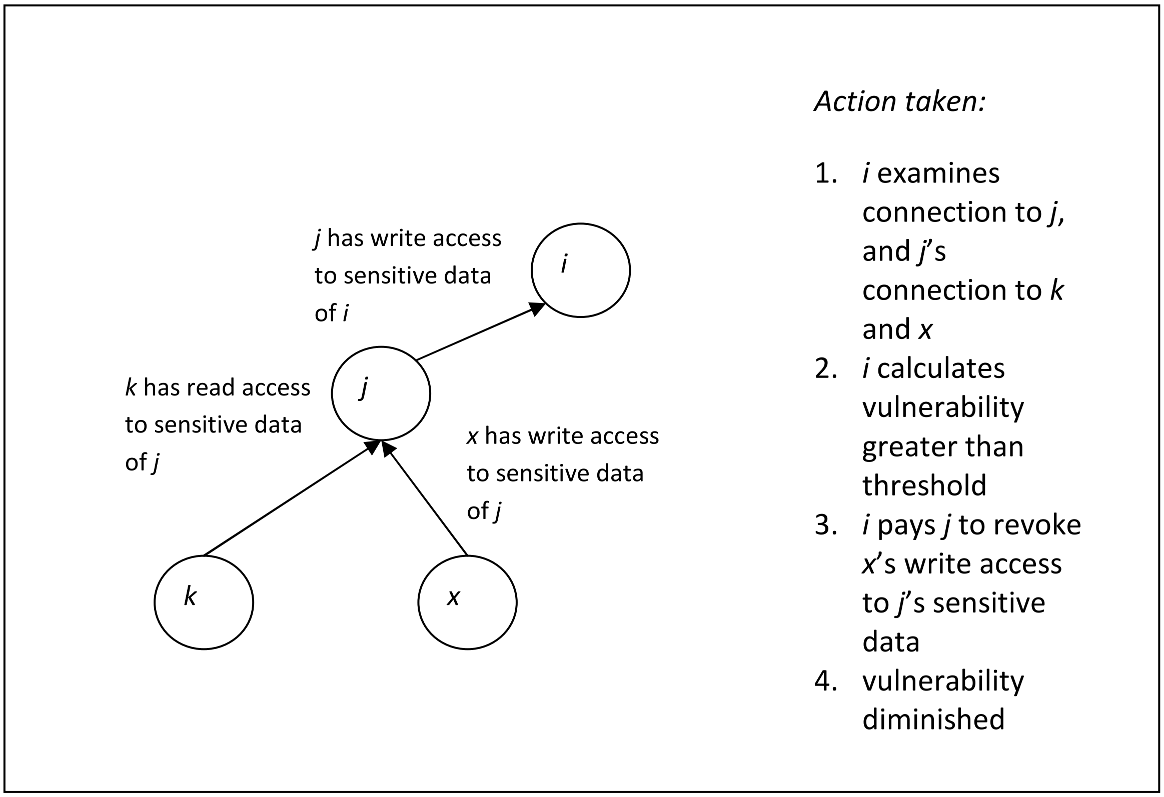
Algorithm 5.2: Attack tree analysis

The purpose of Algorithm 5.2 is to use security metrics with attack tree analysis to determine vulnerability to a node. This algorithm analyzes each path to node *i* from any node *k* to *j* and *j* to *i* for the purpose of establishing vulnerabilities. The level of vulnerability indicates the weakness in security, as security is the opposite of vulnerability. Algorithm 5.2 is called by each node at each turn in the game to analyze all the other connections to other nodes, within the parameters of the algorithm. In general, the vulnerability posed by another node decreases as distance from self decreases; it also decreases if another node belongs to the same coalition as the node performs the analysis by calling Algorithm 5.2.

In addition to examining coalition membership, each node examines each s_{ij} according to its metrics: encryption of the link, read or write permissions between nodes (both direct and indirect), and corresponding data sensitivity. This includes examining which nodes have direct or indirect access. The *danger_threshold* input to the algorithm is a threshold that describes vulnerability tolerance, and is not determined by any game theoretic means, but is established by the node doing the analysis. Consequently, if the threshold is too low, it will result in disconnecting from the network and having to subsequently raise the tolerance to vulnerability so that it will stay connected. The vulnerability is referred to as the *threat_level*; it represents the calculated vulnerability to node *i*. We assume node *i* represents the node performing the analysis.

Recall that connections to other nodes outside the coalition are a necessary part of preserving graph connectivity, but this does not mean that they are granted the same privileges as coalition members. Because of access control policies and coalition membership formation, other members of the coalition are assumed to be a decreased threat.

The advantage of our algorithm is each node examines the connections in the local neighborhood and local plus one (distance of two), which is realistic to analyze. Consequently, however, the disadvantage of our algorithm is the lack of comprehensiveness, as not all connections beyond this point are examined. Examining all connections has the disadvantage of decreased efficiency. We assume, however, that this attack tree analysis algorithm is sufficient to analyze and make the optimal decision, provided coalitions are used to provide information about the rest of the network. Our reasoning is explained in Proof 5.2, which follows after Algorithm 5.2 below. Figure 5.1, shown below, contains an example of a node *i* performing attack tree analysis according to Algorithm 5.2, followed by vulnerability reduction according to Algorithm 5.3.


j has write access to sensitive data of *i*
i
k has read access to sensitive data of *j*
j
x has write access to sensitive data of *j*
k
x

Action taken:

1. *i* examines connection to *j*, and *j*'s connection to *k* and *x*
2. *i* calculates vulnerability greater than threshold
3. *i* pays *j* to revoke *x*'s write access to *j*'s sensitive data
4. vulnerability diminished

Figure 5.1: Attack tree analysis and vulnerability reduction example diagram

Algorithm 5.2: Attack tree analysis:

Input: *G*= (*V*, *E*),
Partition A,
difference_threshold,
danger_threshold

Output: *X*, set containing all subsets of nodes connected to *i* with read/write permissions to sensitive data on a path to node *i* and their respective *threat_level* [0…1]

```
Attack_tree_analysis( G, A, difference_threshold, danger_threshold )
 $X = \emptyset$
for each node $j \in V$
{
    $Y_j = \emptyset$
    threat_level = 0.0
    if $(e_{ji}$ && $|j - i| == 1)$//if edge to self and distance is one
    {
        if $(\ w_{ji}(d(i)))$//if write access to self
         {
            if $((d(i)) == 0)$//if data is sensitive
            {
              Add $w_{ji}(d(i))$ as member of set X
              for each node $k \in V$
              {
                   if $(e_{kj}$ && $|k - i| == 2)$//if indirect edge to self, distance is two
                   {
                       if $(\ w_{kj}(d(j)))$//if write access to node with direct access
                       {
                            if $((d(j)) == 0)$//if data of that node is sensitive
                            {
                                  Add $w_{kj}(d(j))$ as member of set X
                                  if ( $y_{kj} = 0$ || $y_{ji} = 0$ )
                                         threat_level += danger_threshold / (|k-i|)
                                  else if ( $y_{kj} = 0$ && $A(s_{ij}) \leq$ difference_threshold )
                                         threat_level += danger_threshold / (|k-i|+2)
                                  else
```

```
                              $threat\_level$ += $danger\_threshold$ / (|$k$-$i$|+1)
                    }
                }
                if $\left(r_{kj}\left(d(j)\right)\right)$//if read access to node with direct access
                {
                    if $\left(\left(d(j)\right) == 0\right)$ //if data of that node is sensitive
                    {
                            Add $r_{kj}\left(d(j)\right)$ as member of set X
                            if ( $y_{kj}$ = 0 || $y_{ji}$ = 0 )
                                    $threat\_level$ += $danger\_threshold$ / (|$k$-$i$|+2)
                            else if ( $y_{kj}$== 0 &&
                                            $A(s_{ij}) \leq difference\_threshold$ )
                                    $threat\_level$ += $danger\_threshold$ / (|$k$-$i$|+4)
                            else
                                    $threat\_level$ += $danger\_threshold$ / (|$k$-$i$|+3)
                    }
                }
            }
          }
      }
}
if $\left(r_{ji}\left(d(i)\right)\right)$//if read access to self
{
    if $\left(\left(d(i)\right) == 0\right)$ // if data is sensitive
    {
        Add $r_{ji}(d(i))$ as member of set X
        for each node $k \in V$
        {
            if $\left(e_{kj}\ \&\&\ |k - i| == 2\right)$//if indirect edge to self, distance is two
            {
                if $\left(r_{kj}\left(d(j)\right)\right)$//if read access to node with direct access
                {
                    if $\left(\left(d(j)\right) == 0\right)$//if data is sensitive
                    {
                        Add $r_{kj}\left(d(j)\right)$ as member of set X
                        if $\left(y_{kj} == 0 \,||\, y_{ji} == 0\right)$
                                $threat\_level$ += $danger\_threshold$ / (|$k$-$i$|+2)
                        else if $(y_{kj} == 0\ \&\&\ A(s_{ij}) \leq$
                                        $difference\_threshold$ )
                                $threat\_level$ += $danger\_threshold$ / (|$k$-$i$|+4)
                        else
                                $threat\_level$ += $danger\_threshold$ / (|$k$-$i$|+3)
                        }
                    }
                    if $\left( w_{kj}(d(j)) \right)$//if write access to node with direct access
                    {
                        if $\left( (d(j)) == 0 \right)$// if data of that node is sensitive
                        {
                            Add $w_{kj}(d(j))$ as member of set X
                            if $(y_{kj} == 0 \,||\, y_{ji} == 0)$
                                threat_level += danger_threshold / (|k-i|)
                            else if $(y_{kj} == 0$ && $A(s_{ij}) \leq$difference_threshold)
                                threat_level += danger_threshold / (|k-i|+2)
                            else
                                threat_level += danger_threshold / (|k-i|+1)
                        }
                    }
                }
            }
        }
    }
    Add threat_level to $Y_j$
    Add $Y_j$ to X
    }
}
return (X )
```

Theorem 5.1

With coalitions the attack tree analysis of local, one-hop neighbors is sufficient for decision-making if information about the rest of the network *G* is provided by the local neighborhood, *i* = $\bigcup_{j=j}^{h} e_{ij}$, with at least one of the nodes belonging to the same coalition as node *i*.

Proof 5.1 Proof of Theorem 5.1

Basis Step:

According to Algorithm 5.2, node *i* receives information directly by examining its local neighbors to which it is directly connected,

$$\bigcup_{j=j}^{h} e_{ij}$$

having distance equal to one with one edge between *i* and each of its *h* neighbors forming its local graph,

$$|h - i| = 1$$

Because it is directly connected, node *i* receives information from each of its *h* neighbors.

at

$$i + 1$$

Inductive Step:

Assume the basis step is true for *n*, and show true for *n*+1.

Since we assumed true for *n*, node *i* is directly connected to its *h* neighbors with

$$|h - i| = 1$$

and receives information from these nodes at

$$i + 1$$

with neighbors of i+1 connected to each of their h neighbors at

$$i + 2$$

with

$$\bigcup_{j=j+1}^{h} e_{i+1j+1}$$

from which they receive information, with successive steps of information propagation along from nodes at

$$i + n$$

having distance or number of edges equal to

$$|n - i| = n$$

with

$$\bigcup_{j=j+n}^{h} e_{i+nj+n}$$

from which nodes at $i + n$ receive information, giving

$$S_n = \ (i + n) \cdot (i + n - 1) \cdot (i + n - 2) \cdot \ldots \ \cdot (i + 2) \cdot (i + 1) \cdot i$$

For successive steps of n+1 we have number of edges

$$|n + 1 - i| = n + 1$$

or, using the sequence notation for S_n above, is written

$$S_{n+1} = \ (i+n+1)\cdot(i+n)\cdot(i+n-1)\cdot(i+n-2)\cdot \ldots \cdot(i+2)\cdot(i+1)\cdot i$$

Because we assumed true for *n*, we have the well-known formula

$$S_n = \frac{n\cdot(n+1)}{2}$$

Then for n+1 we have successive steps of information propagation along the edges of the graph

$$e_{i+n+1j+n+1}$$

from n+1 to i equal to

$$|n+1-i| = n+1$$

which is

$$S_{n+1} = \frac{(n+1)\cdot(n+2)}{2}$$

And because we assumed true for *n*, we have

$$S_{n+1} = \ (i+n+1)\cdot(i+n)\cdot(i+n-1)\cdot(i+n-2)\cdot \ldots \cdot(i+2)\cdot(i+1)\cdot i$$

$$S_{n+1} = \ (i+n+1)\cdot\frac{n\cdot(n+1)}{2}$$

And since

$$(n + 1 - i)$$

is equivalent to the distance from node *i*

$$(n + 1 - i) = |n + 1 - i| = (n + 1)$$

Then

$$S_{n+1} = \ (n + 1) \cdot \frac{n \cdot (n + 1)}{2}$$

$$S_{n+1} = \frac{(n + 1) \cdot (n + 2)}{2}$$

And, because node *i* is also connected directly

$$\bigcup_{j=j}^{h} e_{ij}$$

and indirectly to its coalition members, denoted

$$\sigma_{Ai}$$

node *i* is able to get information

$$\lambda$$

on other coalition nodes along the sum of all neighbors of coalition members

$$\bigcup_{j=j}^{h} e_{ij} \cap \sigma_{Ai} \neq \emptyset$$

thus allowing information λ to propagate to *i* to analyze network *G* according to Algorithm 5.2.

Q.E.D.

Note that this proof does not guarantee that node *i* has comprehensively analyzed all connections in *G* through direct analysis and indirect information passed to it by coalition members. While coalition members pass along their own analysis of other nodes to which *i* is not directly connected or at distance greater than two, it is not necessarily comprehensive with respect to all nodes and connections in *G*. We believe the tradeoff of time versus comprehensive analysis by each node of each connection, which is essentially a variation of the traveling salesperson problem, is a more than fair one. It thus gives our approach the advantage with respect to time versus comprehensiveness.

5.3 Pareto optimization

Recall that Pareto optimized is defined as a state in which a player can take no action deviating from this state without decreasing utility for itself or some other player. Recall that Pareto optimization goes beyond the definition of equilibrium. It the best of the equilibrium or is more optimal than equilibrium. Pareto optimization is used to simultaneously minimize or maximize a finite set of real-valued functions, which in our optimization problem refers to each player choosing a move that maximizes the utility function. In the specific context of the problem our work is trying to solve, that of maximizing overall network security, the security of each node or player is Pareto optimal for that turn in the game if no player can increase its utility, or security, without decreasing the utility and consequently security of itself or another player. The Pareto optimal move for player *i* is denoted

$$(s_i^P, \sigma_i(s_i^P)) \tag{5.1}$$

The Pareto optimal move is

$$(s_i^P, \sigma_i(s_i^P)) \geq (s_i^*, \sigma_i(s_i^*)) > (s_i', \sigma_i(s_i')) \tag{5.2}$$

where all moves are members of the set of moves *S*. Recall that

$$(s_i', \sigma_i(s_i'))$$

represents any of the other sub-optimal moves. Thus, both the Pareto optimal and equilibrium move are better than a sub-optimal move.

The Pareto optimal move can be the equilibrium move, but it is always at least as good as or better than equilibrium, written

$$(s_i^P, \sigma_i(s_i^P)) \geq (s_i^*, \sigma_i(s_i^*))$$

Despite the possibility that the Pareto optimal and equilibrium moves may be the same move, the Pareto optimal move supersedes equilibrium because it is always at least as optimal or more optimal.

Recall that our game is a weighted potential game for either mixed or pure strategies. In a potential game, a player choosing the move that maximizes the utility function is also maximizing the potential function: for mixed strategies the two functions are equal, and for pure strategies the two functions have output that is positive if the other is positive. Hence, if a player chooses a move that maximizes the utility, hence playing the equilibrium move, it also maximizes the potential function. Since a Pareto optimized move means that if the player deviates from this move it decreases utility for itself or some other player, a player cannot play a move that decreases utility for some other player because it would not maximize the potential function. Hence, players who maximize utility maximize their security, and in doing so maximize the potential function and consequently are making a Pareto optimal move. It is impossible to not maximize the potential function without also making a Pareto optimal move. We will use this reasoning for our proof of Pareto-optimality below.

Theorem 5.2

If the game is a weighted potential game, then a player maximizing the utility function also maximizes the potential function and by doing so chooses the equilibrium action which is Pareto optimal.

Proof 5.2 Proof of Theorem 5.2

Let us break Theorem 5.2 into smaller statements. Let the statement *p* represent

> *p*: If the game is a weighted potential game, then a player maximizes the potential function

And let statements *q*, *r*, and *s* represent *q*: If a player maximizes the potential function, then a player maximizes the utility function.

r: If a player maximizes the utility function, the player chooses the equilibrium action.

s: If a player maximizes the utility function, it chooses the Pareto optimal action.

The definition of a potential game is that it possesses a potential function, which is maximized by all players. The definition of a weighted potential game is a potential game that possesses a utility function which is directly related to other players' utility functions. A weighted potential game possesses a potential function for either mixed or pure strategies. Thus we know statement *p* is true according to the definition of a weighted potential game. Furthermore, we also know by the definition of a potential function that players maximizing the utility function maximize the potential function. Thus, we also know statement *q* is true according to the definition of potential. And statement *r* is true because of the definition of equilibrium. However, proving statement *s* true is somewhat more involved.

The definition of a Pareto optimized action means that if the player deviates from this action, then it decreases utility for itself or some other player. But according to statements *p*, *q*, and *r*, a player cannot maximize the potential function without maximizing utility, and if it is not maximizing utility then it is not choosing the equilibrium action. If a player plays a move that decreases utility for some other player, it is not maximizing the potential function because the potential function is maximized by all players maximizing the utility function, and players maximizing utility choose the equilibrium action. If a player is not maximizing the potential function, the game is not a weighted potential game, which is a contradiction. Therefore, since a player maximizing utility in a weighted potential game chooses the equilibrium action, this action is Pareto optimal, and the weighted potential game is Pareto optimal.

Q.E.D.

Algorithm 5.3: Reduce vulnerability

This algorithm is used by each node to act upon the information generated by calling the above *Attack_tree_analysis* Algorithm 5.2. After calling the *Attack_tree_analysis* algorithm, nodes executing Algorithm 5.3 determine the best way to improve security by reducing vulnerability per the definition of security from equation (3.25). If the node performing the analysis determines that *threat_level* is greater than its tolerance for vulnerability, *danger_threshold*, the node can pay another node in the network to change the nature of its connection, or simply change the connection itself, provided it is already connected. Actions possible are related to the activities for the connection: security level could be increased, permissions could be revoked, or the connection could be broken in favor of another node. The disadvantages are the possibility that the graph splits into disconnected segments, or that coalition membership is inadvertently revoked due to actions taken by other nodes or the node itself via acting to change a connection on the prompting of another node. This algorithm allows a node to maximize its utility, or security by determining which action maximizes utility, thereby taking the equilibrium action.

Algorithm 5.3: Reduce vulnerability:

Equations 1 and 2 are either equations (3.25) and (4.18) if playing according to the pure strategies of Algorithm 5.4, or equations (3.25) and (5.6) if playing according to the mixed strategies of Algorithm 5.5

Input: X, set containing all subsets of nodes connected to i with read/write permissions to sensitive data on a path to node i and their respective *threat_level* [0…1],
danger_threshold,
Output: maximized utility u_i

```
Reduce_ vulnerability(X, danger_threshold)
max_ vulnerability = 0
X_vulnerable = ∅
for each subset $Y_j$ from $Y_{start} \dots Y_{end} \in X$
{
        get threat_level of $Y_j$
        if ( threat_level > max_vulnerability )
        {
                max_vulnerability = threat_level
                X_vulnerable = $Y_j$
        }
}
get threat_level of X_vulnerable
if ( threat_level > danger_threshold )
{
        pay j to revoke $w_{kj}(d(i))$ || pay j to revoke $r_{kj}(d(i))$
        || pay j to increase $s_{jk}$ || revoke $w_{ji}(d(i))$ || revoke $r_{ji}(d(i))$
}

best_new_action = 0
for each node $j$ == $-i$
{
      if ( $e_{ij} \notin X$ )
      {
            analyze theoretic $s_{ij}$ according to equations 1 and 2
            if( $s_{ij}$ > best_new_action )
                  best_new_action = $s_{ij}$
      }
}
take best_new_action to maximize $u_i$:
            preserve connection $e_{ij}$ at $s_{ij}$ || break connection $e_{ij}$ ||
            bid to form connection $e_{ij}$ at $s_{ij}$ || change security level of $s_{ij}$ ||
            receive payment from $j$
return $u_i$
```

5.4 Game to optimize network security

We denote a game using the symbol Γ. Game Γ is composed of the players, which are nodes in the graph *G*; the actions that the players can take, which involve choosing security levels from *S*; and utility function *u*, with the symbols ⟨ ⟩ are used instead of parentheses to indicate this is a game. Hence the game is denoted

$$\Gamma = \langle V, S, u \rangle \tag{5.3}$$

5.5 Pure strategy game

A game whereby players move according to their equilibrium action is a pure strategy game. Players always choose the equilibrium action in a pure strategy game. Ordinal potential games are used to optimize pure strategy games. We define a game Γ using pure strategies as a weighted ordinal potential game if there exists a potential function ∏ such that the result calculated by potential function ∏ is greater than or equal to zero if-and-only-if the result calculated by the utility function *u* is greater than or equal to zero, which is denoted for the equilibrium action of node *i* and the equilibrium actions of all other nodes –*i*,

$$\Pi\big((s_i^*, \sigma_i(s_i^*)), (s_{-i}^*, \sigma_{-i}(s_{-i}^*))\big) \geq 0 \quad \textit{if-and-only-if} \\ u\big((s_i^*, \sigma_i(s_i^*)), (s_{-i}^*, \sigma_{-i}(s_{-i}^*))\big) \geq 0 \qquad (5.4)$$

Algorithm 5.4: Pure strategy game

This algorithm is the actual game played to optimize security for the entire network. The players play according to pure strategies per equation (5.4) to maximize their security. Players form coalitions and perform attack tree analysis to determine the best move to maximize security, and in so doing choose the Pareto optimal equilibrium action. We assume that each node -*i* examined does not change state until after node *i* takes action. We denote the game using the symbol Γ as defined earlier.

Algorithm 5.4: Pure strategy game:

This algorithm references equations (3.25) and (4.18) for players acting according to pure strategies.

```
Input:  Γ = ⟨$V, S, u$⟩,
        $G$ = ($V$, $E$) of game Γ connected a priori,
        partition $A$,
        difference_threshold for coalition membership
Output:   $G$, the network optimized for overall security


Pure_strategy_game( Γ, $G$, $A$, difference_threshold)
```

```
while improvement to each player’s security $s_i$ can be made
{
    call Algorithm 5.1: Coalition formation(G, A, difference_threshold)
    for each player $i \in V$
    {
        call Algorithm 5.2: Attack_tree_analysis(G, A, difference_threshold,
                                                 danger_threshold)
        // Algorithm 5.3 is used with equation (4.18) whereby players move
        //     according to their pure strategy:
        call Algorithm 5.3: Reduce_vulnerability (X, danger_threshold)
        receive utility $u_i$
    }
}
return (G)
```

5.6 Mixed strategy game

A mixed strategy game is identical to a pure strategy game with the exception that players choose an action according to a probability distribution over the pure strategies. In other words, players choose the equilibrium action some percentage of the time, and a less than optimum (non-equilibrium) action the remaining percentage of the time. The percentage is chosen by the game designer, and has to do with the rate of player error the designer wants to model in a game. Exact potential games are used to optimize mixed strategy games. Our mixed strategy game algorithm, Algorithm 5.5, is identical to Algorithm 5.4 above with the exception that players move according to mixed strategies.

The equilibrium action by player *i* within its constraints, $(s_i^*, \sigma_i(s_i^*))$, given the probability of taking the equilibrium action, is denoted with the symbol | to indicate probability of that action as

$$\big((s_i^*, \sigma_i(s_i^*)) \,\big|\, s_i^*\big) \tag{5.5}$$

where the probability of taking the equilibrium action ranges from 0 to 1.

Utility of *i* is maximized if it takes the best action within its constraints, $\left(s_i^*, \sigma_i(s_i^*)\right)$, given the probability of taking that action s_i^*, when all other players take their best action $(s_{-i}^*, \sigma_{-i}(s_{-i}^*))$ given the probability of taking that action. Again, the symbol | indicates probability of that action, where

$$u\left(\left((s_i^*, \sigma_i(s_i^*)) \,\middle|\, s_i^*\right), \left((s_{-i}^*, \sigma_{-i}(s_{-i}^*)) \mid s_{-i}^*\right)\right) \geq u\left(\left((s_i', \sigma_i(s_i')) \mid s_i'\right), \left((s_{-i}, \sigma_{-i}(s_{-i})) \mid s_{-i}\right)\right) \tag{5.6}$$

We define a game Γ using mixed strategies as a weighted exact potential game if there exists a potential function ∏ such that the result calculated by potential function ∏ is equal to the result calculated by the utility function, which is for the equilibrium action of node *i* and the equilibrium actions of all other nodes –*i*, denoted

$$\Pi\left(\left((s_i^*, \sigma_i(s_i^*)) \,\middle|\, s_i^*\right), \left((s_{-i}^*, \sigma_{-i}(s_{-i}^*)) \mid s_{-i}^*\right)\right) = u\left(\left((s_i^*, \sigma_i(s_i^*)) \mid s_i^*\right), \left((s_{-i}^*, \sigma_{-i}(s_{-i}^*)) \mid s_{-i}^*\right)\right) \tag{5.7}$$

Algorithm 5.5: Mixed strategy game:

This algorithm references equations (3.25) and (5.6) for actions according to a probability distribution over the mixed strategies.

Input: Γ = $\langle V, S, u\rangle$,
G = (*V*, *E*) of game Γ connected a priori,
partition *A*,
difference_threshold for coalition membership
Output: *G*, the network optimized for overall security

```
Mixed_strategy_game ( $\Gamma$, G, A, difference_threshold)
while improvement to each player's security $s_i$ can be made
{
    call Algorithm 5.1: Coalition formation(G, A, difference_threshold)
    for each player $i \in V$
    {
        call Algorithm 5.2: Attack_tree_analysis(G, A, difference_threshold,
                                                 danger_threshold)
        // Algorithm 5.3 is used with equation (5.6) whereby players move
        //      according to a probability distribution over the pure strategies:
        call Algorithm 5.3: Reduce_vulnerability (X, danger_threshold)
        receive utility $u_i$
    }
}
return (G)
```

5.7: Game-based architecture

If we are to apply our game theoretic network security optimization algorithm to optimize an actual physical network, we need to develop an architecture to accomplish this task. While our simulations in prototypes one through four allowed us to use metrics to test and evaluate our work and its ability to optimize network security, these are simulations nonetheless.

If we were to implement our algorithm, and extend our prototypes, to an actual network composed of PCs and other devices that make up a heterogeneous wireless network, then we could better evaluate our algorithm's performance and possibly gain insight into additional ways our prototype might be improved. In addition, implementing our algorithm on an actual network would allow us to better test any future game theoretic work whereby we extend our definitions of security to include authentication as a metric, and implement this in the heterogeneous network. We did not consider authentication in our model due to the fact that our game uses only symmetric information; all the nodes in the network can observe all events in the game, including the identities of the other nodes. Computers in an actual network would likely have asymmetric information and therefore would often need to authenticate other nodes before sharing data or extending other privileges.

As a first step in accomplishing these tasks, we will specify an architecture that describes the relationship between hardware, software, the network, and our game theoretic algorithm. First, we must examine our assumptions and specifications in developing our architecture. We assume that the nodes forming the network do not change during the network lifetime. As such, we will not deal with new nodes entering the network at this time. Adding this capability to our architecture will be considered in future work. In addition, we will only address the top level layer of the TCP reference model; our work is considered to take place at a layer above the application layer of the TCP protocol stack. How our architecture interacts with the lower layers of the TCP reference model is considered to be beyond the scope of this work. This includes the interactions brought about by routing protocols and the topology extractor as detailed below, as these take place at the application layer of the TCP stack, which is the next layer beneath our work.

We must reiterate that our security architecture is for a network without a centralized coordinator. All nodes in the network are independent from one another and are heterogeneous. All of the nodes have the capability to implement our game based architecture as shown below in Figure 5.2, enabling the nodes to improve their own security in addition to improving the overall security of the network. Furthermore, the ability to independently construct an attack tree is available to each node in the network, and each node can consequently analyze the network using its implementation of an attack tree per Algorithm 5.2: Attack tree analysis.

In addition, we must detail our assumptions relating to the symmetry of information in our game, and how this symmetry is implemented in our architecture. As mentioned above, nodes are able to see the other nodes and the actions taking place in the game. Information in this network is symmetric. In our architecture, we assume routers are located in the network, and routing tables are used to show which neighboring nodes in the network are reachable. These nodes to which another node is connected form the local neighborhood, or local network, of a node and are the nodes with which any direct communication takes place. A topology extractor is used to determine which nodes are connected to other nodes. The topology extractor works by extracting information from the routing tables. This information describes any connections in the network, including node IP addresses and the characteristics or attributes of the links between nodes in the network. These attributes describe the encryption strength of the connection, the constraints on the link with regard to minimum and maximum security values of each node forming the link, and any hardware information used in creating the link itself. Furthermore, the topology extractor is able to provide information on the physical relationships and locations of the nodes in the network, including any path information describing available routes to a node.

The routing protocols used to disseminate information available in the routing tables include BGP and RIP over TCP or UDP. Border Gateway Protocol, or BGP, is used to exchange routing information regarding nodes, links, and destinations in the network. Routing Information Protocol, or RIP, is used to exchange information regarding the distance or number of nodes from a path to a destination. We assume that these routing tables are kept updated as the network changes during the optimization phase. The actual implementation of the routing tables and any packet information sent between nodes and routers is beyond the scope of this work, but will make for interesting future work. We also assume there are no errors in the routing tables and that no information has been falsified. Consideration of any errors in the routing table is reserved for future exploration, as these errors in addition to false node identities, are what would cause asymmetric information in our game theoretic architecture. We are currently considering only symmetric information. In using a topology extractor and routing tables in such a scenario, our assumptions allow us to fulfill the architectural implementation corresponding to the symmetric information in our game.

The diagram shown below in Figure 5.2 illustrates how our architecture will work and its relationship to the lower levels of the TCP stack. Our architecture forms a new layer, which we call the Game layer, and contains the actual implementations of our Algorithms 5.1 – 5.5 above. Each node is assumed to possess this layer. In its diagram below, we can see the relationship between the attack tree analysis of Algorithm 5.2, the coalition formation of Algorithm 5.1, and the subsequent actions taken by a node to increase its security by reducing its vulnerability through taking the optimal action. The topology extractor plays an essential role. It gathers, or extracts, all information about the topology of the network, including hardware and route information, distance between nodes, security levels, constraints of nodes and links, and encryption of links.

The order of execution of each part of our architecture is shown, listed by number, in Figure 5.2 below. The numbers on the arrows specify the order of the steps. The arrows themselves show the relationship between the algorithms, whereby the direction of the arrows show the input to that part of the architecture. When a node performs attack tree analysis of the network using Algorithm 5.2, the information previously gathered by the topology extractor is assumed to be current. As such, the topology extractor is run just prior to attack tree analysis in step 1. At step 2, attack tree analysis is performed. At steps 3 and 4, the information gathered by the topology extractor and the attack tree analysis is input to both the bidding and side payments aspects of Algorithm 5.3: Reduce vulnerability. Here in our architecture, these aspects of our Algorithm 5.3 are separated in order to better show their relationship. In addition, if an optimal action for a node is specified as including bidding to form a link and, if needed, a side payment, it would be separated as shown in our diagram. Such a separation is considered to be valid within the specifications of our game.

The two components of Algorithm 5.3: Reduce vulnerability are executed. First, the network is evaluated to determine the best action; actions will involve bids, and prices for actions must be determined. This work is done by the component identified as the bidding algorithm, and must take place prior to any side payments; hence its location and order in Figure 5.2. Whether the optimal action is determined to be a side payment or creating a link, bids must be made between nodes to agree upon the price. Then, this information is passed on to the latter component of Algorithm 5.3, which is represented in our architectural diagram as the side payments algorithm. Here, evaluation of the possibility of side payments takes place. The information gathered by the topology extractor regarding the current state of the network is essential, as this information will be used to route a message and payment regarding the action induced by the side payment.

Once the optimal action is determined through the bidding and side payments algorithms, the node moves on to step 6 in Figure 5.2 and re-evaluates its coalition membership and updates it if needed according to Algorithm 5.1: Coalition formation. Finally, the node executes its optimal action by moving through step 7 to steps 8 or 9, depending upon whether or not our node is executing a mixed strategy or pure strategy. If a node is executing a pure strategy, it takes the optimal action according to Algorithm 5.4. However, as detailed in Algorithm 5.5, nodes can use mixed strategies whereby they take a suboptimal action instead of an optimal action some percentage of the time. Mixed strategies can be deliberately used by nodes because there is the possibility that the exploration induced by the suboptimal action, in combination with the optimal action itself, can lead to a higher level of security than a node always choosing the optimal action. Determining the best percentage of the time that nodes correctly choose the optimal action is not addressed in our work; however, we assume it to be high. We did not consider the scenario in our prototype implementations where some nodes play mixed and some play pure strategies, but this can be explored in future work.

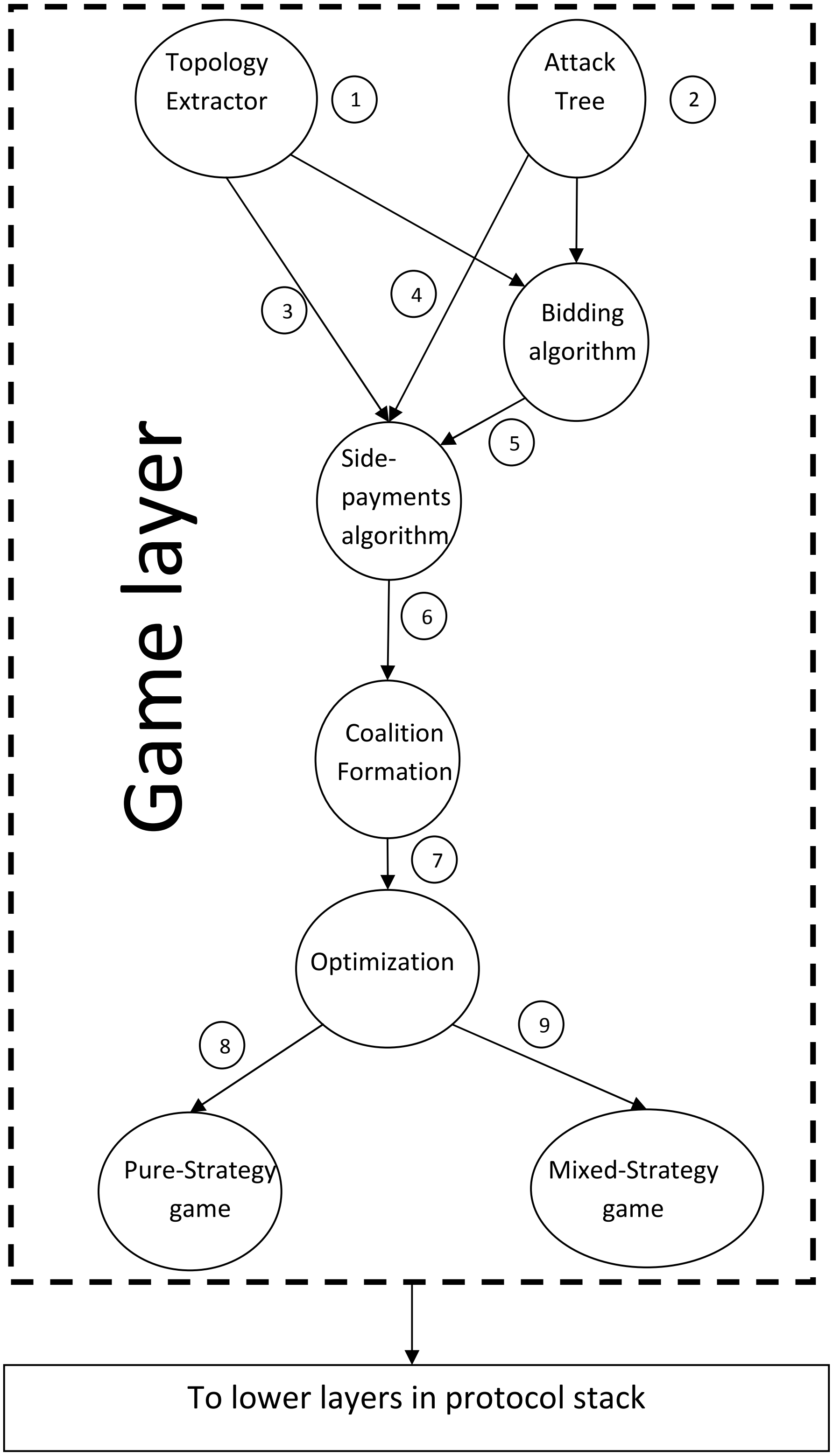


Figure 5.2: Game layer diagram

CHAPTER VI

VALIDATION

This chapter deals with describing an architecture which is very different from our own architecture; its purpose is enhanced evaluation and validation of our own architecture's performance by comparing the two using similar measurements. As detailed above, our work takes place at a very high layer of the TCP protocol stack. Kerberos, the architecture which is used to compare to our own, pervades all layers of the TCP reference model. Kerberos has just enough in common with our definition of security that it can be described using our own security metrics, and hence its performance can be quantitatively evaluated and measured using these same metrics that describe and quantitatively evaluate the security optimization of our algorithm. As a result, we can quantitatively compare our architecture with Kerberos, with the ultimate goal of showing how well our algorithm optimizes network security in our simulated networks.

6.1 Comparing our algorithm to an established algorithm

While this section involves describing the Kerberos system using our metrics of encryption and authentication, any comparison to our work beyond utilizing our metrics for evaluation ends there. If we compare our network security optimization algorithm to an established network security optimization algorithm, we may more effectively study and evaluate the performance of our own algorithm, if the same measurements are used to gauge performance. We chose Kerberos over other algorithms because it is well-known and has some similarities to our own algorithm, enough to be measured using metrics. Like our algorithm, Kerberos is designed to optimize security of an entire network, uses encryption, and has a means of access control. However, the similarities end there as Kerberos has an unrefined system of access control, whereby other nodes have all access to all data if they are connected. Furthermore, unless a node can achieve high encryption, it cannot join the network. And, Kerberos access control is implemented via a centralized controlling server. Our algorithm is not.

While we know its specifications, we must determine the methodology to represent the Kerberos system in a way that it can be fairly and appropriately compared to our game theoretic algorithm in both its mathematical model and prototype implementation simulations for actual tests and analysis. Kerberos will be represented using a mathematical model in the same vein as our own algorithm, but will stay faithful to the pure Kerberos algorithm. In no way will the Kerberos algorithm be massaged to look better than it actually is, as the purpose of this part of our work is to compare, as accurately as possible, our algorithm with another algorithm for the purposes of evaluating our own algorithm's security optimization. This will allow us to use metrics to measure its security and better compare Kerberos to our own algorithm. We will define and analyze Kerberos and implement a Kerberos prototype simulation to compare to the prototype simulations of our algorithm. We believe this is an effective way to measure the Kerberos system's optimization of network security and compare it to our own, and better measure our algorithm's own performance.

The Kerberos system consists of a ticket-granting server and the participating computers in the network. As we will see in the short proofs below, Kerberos cannot be characterized by a potential game and is not Pareto-optimal. In fact, developing a Kerberos game is not nearly as sophisticated or interesting as the game of our own algorithm. The game that characterizes Kerberos is barely a game at all; it has some striking similarities to a game with only one player since the valid moves are severely limited. The Kerberos system is entirely dependent on a node in the Kerberos network having high encryption: if it cannot achieve high encryption, it cannot participate in the network. Thus, without high encryption, there is no valid move for a node to join the network, and if no nodes can reach high encryption Kerberos is useless, and the network disintegrates.

The Kerberos system consists of the computers to be optimized in the network and the Kerberos server. The Kerberos server is not optimized; it is a centralized controller that grants tickets to computers which give access privileges to other computers in the Kerberos network. These access privileges are not at all refined, but according to the pure Kerberos specification, can be characterized by an almost sledgehammer-like approach to granting privileges: if a computer *i* has a ticket to get access to another computer *j*, then *i* has both read and write access to all data of *j*, both sensitive and insensitive. Unlike our own algorithm which has no central controller, allows nodes to have high or low encryption, and distinguishes between granting access to read or write of sensitive or insensitive data, Kerberos does not possess any of these characteristics; any distinction which goes beyond this level of permissions is beyond the definition of the Kerberos system and is instead defined by the operating system of a computer in the network, which is outside the scope of our work. We will thus use the level of permissions as specified by the pure Kerberos system.

Since the Kerberos server granting tickets is not optimized, we assume that the computers in the network that are attempting to participate in the Kerberos network are the same computers in our own network. Ticket-granted permissions for node *i* to have read and write access to all data of *j* we shall denote using a different notation from our own model. *K* represents Kerberos-granted per the ticket from the server, where

$$K(r,w)_{ij} = r_{ij}\big(d(j)\big) \wedge w_{ij}\,\big(d(j)\big) \tag{6.1}$$

$$K(r,w)_{ij} = \begin{cases} 0, & node\ i\ has\ read\ and\ write\ access\ to\ all\ d(j) \\ 1, & node\ i\ does\ not\ have\ read\ and\ write\ access\ to\ all\ d(j) \end{cases} \tag{6.2}$$

We will denote $d(j)$= 0 and 1 in equation (6.1) because any node that has access to another node's data has access to all of it, both sensitive and insensitive. Because the nodes in the Kerberos network implement encryption, but we already defined our encryption metric y_{ij} earlier, we can use the same encryption metric to describe the encryption of connections between nodes *i* and *j* in the Kerberos network. Thus, security provided by Kerberos can be defined using security metrics as

$$K\big(s_{ij}\big) = K(r,w)_{ij} \times y_{ij} \in S \tag{6.3}$$

Which denote the existence of a one-way link from node *i* to *j* at security level $K\left(s_{ij}\right)$ with possession of Kerberos ticket $K(r,w)_{ij}$ granting *i* access to all data of j. Kerberos security for a node *i* will be thus defined using a two-tuple Cartesian-product of access control and encryption. Access control is represented by the possession of a ticket granted by the central server to some node *i* to have access to node j, encryption pertains to the encryption for the connection from *i* to j. Since nodes in a Kerberos network must use high levels of encryption to contact the Kerberos server in order to get a Kerberos ticket $K(r,w)_{ij}$ granting *i* access to all data of *j*, and then use high encryption to connect to j, without high encryption the node cannot join the network.

Lemma 6.1

A node *i* can participate in the Kerberos network to get the ticket $K(r,w)_{ij}$ *if-and-only-if* $y_{ij} = 1$.

Proof of Lemma 6.1

We will first write the statement symbolically whereby

q: $K(r,w)_{ij}$

r: $y_{ij} = 1$

and is thus denoted

$(q \leftrightarrow r)$

If:

Since we are not trying to prove that all tickets grant access to all nodes because all nodes have high encryption, but rather that a node possessing high encryption can obtain a ticket, we have the existential quantifier whereby

$$\exists q \rightarrow \exists r$$

According to the requirements for a high level of encryption for connections between nodes in the Kerberos network, if a node *i* possesses a ticket to node *j*, written

$$\exists q: K(r, w)_{ij}$$

then the node *i* has high encryption,

$$\exists r: y_{ij} = 1$$

Since according to the definition of the Kerberos system requirements, if

$$r: y_{ij} = 0$$

node *i* cannot participate in the Kerberos network to create a link to node *j* since

$$\nexists q: K(r, w)_{ij}$$

because Kerberos excludes low encryption $y_{ij} = 0$ thus preventing the Kerberos server from granting a ticket to allow $\exists q: K(r, w)_{ij}$. Thus, if

$$y_{ij} = 1$$

then the server allows for a node *i* to obtain a ticket whereby

$$\exists q\colon K(r,w)_{ij}$$

yielding the existence of statements *q* and *r* for the Kerberos system making

$$\exists q \rightarrow \exists r$$

true.

Only-if:

According to the requirements for a high level of encryption for connections between nodes in the Kerberos network, if a node *i* has high encryption,

$$\exists r\colon y_{ij} = 1$$

then it can have a ticket, written

$$\exists q\colon\ K(r,w)_{ij}$$

According to the requirements for a Kerberos server to grant a ticket to some node *i* for connection to node *j*, whereby

$$\exists q\colon\ K(r,w)_{ij}$$

can exist only if

$$r\colon y_{ij} = 1$$

meaning there exists a high level of encryption for connections between nodes *i* and *j* in the Kerberos network, allowing for node *i* to participate in the Kerberos network for connection to j, since the Kerberos server excludes nodes who have links with low encryption, making

$$\exists r \rightarrow \exists q$$

true.

Q.E.D

Note that $y_{ij} = 0$ (low encryption) is not a valid action that a computer, or node, *i* can choose from the domain of actions to participate in the Kerberos network. This makes Kerberos significantly different from our own algorithm. Unless encryption is high, node *i* cannot possess a Kerberos ticket to connect to any other computer; if it cannot connect to any other computer, it only uses part of the definition of its security, whereby security is equal to its low encryption.

Connection is only possible with $y_{ij} = 1$. As such, any game definition of Kerberos has a primitive utility function, since a player either has high encryption and maximizes security, or it does not and gets the security equal to its low encryption. All metrics, including read and write, thus require that encryption be high. Only then can a node *i* be granted, by the Kerberos server, possession of ticket $K(r, w)_{ij}$. Since all metrics pertain to the need for node *i* to possess encryption level $y_{ij} = 1$ in order to subsequently possess Kerberos ticket $K(r, w)_{ij}$, this Kerberos requirement (constraint) for high encryption we denote

$$\sigma_K \tag{6.4}$$

which applies to all Kerberos-server granted security connections, $K(s_{ij})$. We shall thus denote this interdependence pertaining to the whole of Kerberos security as

$$\sigma_K\left(K(s_{ij})\right) \tag{6.5}$$

If we assume for all direct connections *i* makes to its neighbors, we can define *K*(s_i), the Kerberos security level of node *i*, as the weakest of all the direct connections from node *i*

$$K(s_i) = min\left(K(s_{ij}) \cup\ K(s_{i-j})\right) \tag{6.6}$$

Keep in mind that a node *i* defining security in equation 3.58 has an incomplete definition unless encryption is high. We next write the action profile in equilibrium for Kerberos. However, this is somewhat of a trivial definition since the truly best actions in a Kerberos network are all depending upon high encryption. Still, for completeness' sake, we define a Kerberos action profile in equilibrium as

$$K(s^*) = \left(\sigma_K\left(K(s_1^*)\right), \sigma_K\left(K(s_2^*)\right), \dots, \sigma_K\left(K(s_n^*)\right)\right) \tag{6.7}$$

We define Kerberos equilibrium for player *i* as player *i*'s best action (yielding maximum utility) given the actions of all the other players –*i*,

$$Ku_i\left(\sigma_K\left(K(s_i^*)\right), \sigma_K\left(K(s_{-i})\right)\right) \geq Ku_i\left(\sigma_K\left(K(s_i')\right), \sigma_K\left(K(s_{-i})\right)\right) \tag{6.8}$$

However, the Kerberos game is not a potential game. Our reasons for this are discussed below.

Theorem 6.1:

The Kerberos game is not a potential game.

Proof 6.1 Proof of Theorem 6.1

The Kerberos game is not a potential game due to its lack of potential function. By the definition of Kerberos constraints in Lemma 6.1 which requires

$$y_{ij} = 1$$

for a node *i* to participate in the Kerberos network and form a link to any other node *j*; and by the definition of *G* all nodes are heterogeneous, and thus

$$\exists - i: y_{-ij} \neq 1$$

making nodes –*i* unable to join the network. Since the definition of a potential game is one in which the potential function is maximized by all players, then to participate in the Kerberos network

$$\forall - i: y_{-ij} = 1$$

in order to maximize the potential function, which contradicts the definition of a heterogeneous network. Thus any players –*i* ≠ 1 fail to maximize the function, which would otherwise have been the potential function, if their encryption level $y_{-ij} = 0$. Thus, the Kerberos game is not a potential game as it lacks a potential function maximized by all players.

Q.E.D.

Theorem 6.2:

The Kerberos game is not Pareto-optimal.

Proof 6.2 Proof of Theorem 6.2

This proof is given using a direct proof.

While the Kerberos game is a game possessing an equilibrium function, Pareto optimization is more optimal than equilibrium. Since according to Lemma 6.1 players *–i* having weak encryption

$$y_{-ij} = 0$$

cannot form a connection to the network, they cannot maximize the utility; there is no potential function. And, since any player *i* with high encryption taking an equilibrium action to maximize its utility Ku_i causes all other players *-i* with low encryption to have utility Ku_{-i} decreased, the Kerberos game is not Pareto-optimal.

Q.E.D.

We consider implementing a mixed strategy game for Kerberos to be beyond the scope of our work. While doing so might help us better evaluate the performance of our prototype four, we are unsure whether implementing a mixed strategy version of Kerberos is faithful to the algorithm. Kerberos is not a potential or Pareto optimal game, as it has high inter-dependability of the variables and constraints which restrict the moves of the game. We believe that a mixed strategy game would be unfaithful to the pure Kerberos algorithm and its moves, and thus create a new algorithm altogether.

6.2 O(n) analysis

An O(n) analysis of our algorithm and Kerberos is below. We begin by examining the process of forming a connection from node i to node j for n nodes in the network. The actions taken for link formation in the Kerberos network are:

1. Node i sends request to central server node, presenting its credentials.
2. Node i receives answer from server.
3. Node i sends request to j presenting credentials (ticket and key) from central server node.
4. Node j sends request to central server node, presenting its credentials and those of i.
5. Node j receives response from server, allowing i to log in.
6. Node j sends message to i and grants permissions to i all that server allows, thus forming link e_{ij}.

Thus applying e_{ij} for n nodes, there are at most six actions, which can be written using O(n) notation

$$O(6n)$$

Furthermore, there are at least six actions, meaning the Kerberos network is

$$\Omega(6n)$$

and thus making it

$$\theta(6n)$$

or, more commonly

$$\theta(n)$$

However, this fails to account for the number of *m* nodes not allowed to participate in Kerberos at all due to their low encryption, thus making the more likely case

$$O(6n\text{-}m)$$

and

$$\Omega(6n\text{-}m)$$

giving thus

$$\theta(6n\text{-}m)$$

for Kerberos.

The actions taken in to form a link our game theoretic security algorithm are:

Best-case scenario:

1. Node *i* sends request and bid to *j* which includes its $s = y \times r \times w$
2. Node *j* accepts bid, and thus allows *i* to connect, granting access to its data per its constraints.

Thus extrapolating for *n* nodes, there are at least two actions, which can be written

$$\Omega(2n)$$

or, rather,

$$\Omega(n)$$

Worst-case scenario:

1. Node *i* sends request and bid to *j* which includes its $s = y \times r \times w$.
2. Node *j* rejects bid.
3. Bidding continues for some *b* number of iterations, whereby $b < n$ or $b \geq n$ depending on how soon *j* accepts the bid.
4. Node *j* accepts bid, and thus allows *i* to connect, granting access to its data per its constraints.

Thus for *n* nodes, there are at most $n \cdot b$ actions, which can be written

$$\mathrm{O}(n \cdot b)$$

Which, to give greater meaning to *b*, can be refined using the following:

Best-case for number of iterations in *b*, whereby

$$b < n$$

$$b \leq n - 1$$

Using O(*n*) notation, we write the best or "omega" case of O(*n*) as

$$\mathrm{O}(n \cdot b) \leq \mathrm{O}(n \cdot (n - 1)) = \mathrm{O}(n^2)$$

And similarly,

Worst-case for number of iterations in *b*, whereby

$$b > n$$

$$b \geq n + 1$$

Then using O(*n*) notation, we write the worst or "big-oh" case of O(*n*) as

$$\mathrm{O}(n \cdot b) \qquad = \mathrm{O}(n^2)$$

$$= \mathrm{O}(n \cdot 2n)$$

$$= \mathrm{O}\big(n \cdot (n + n)\big)$$

$$\geq \mathrm{O}\big(n \cdot (n + n - 1)\big) \geq \ldots \geq \mathrm{O}\big(n \cdot (n + 2)\big)$$

$$\geq \mathrm{O}\big(n \cdot (n + 1)\big) \geq \mathrm{O}(n \cdot (n)) \ \geq \mathrm{O}(n \cdot (n - 1))$$

And thus we can conclusively say, in a worst-case scenario our algorithm is O(n^2). Since it is also Ω(n), there is no way to write our algorithm using theta-notation θ(n).

For O(n) analysis of time, Kerberos has the advantage, whereas for Ω(n) analysis our algorithm has the advantage in some cases, but Kerberos has the advantage in other cases. Kerberos has the advantage when

$$n \leq m$$

Giving

$$\Omega(6n - m) < \Omega(n)$$

Since

$$\Omega(6n - m) \rightarrow \Omega(1) \text{ as } m \rightarrow n$$

However, our algorithm has the advantage, or performs not worse than Kerberos, when

$$n > m$$

Giving

$$\Omega(6n - m) > \Omega(n)$$

Since in this case

$$\Omega(6n - m) \rightarrow \Omega(n) \text{ as } m \rightarrow 1$$

This O(n) analysis demonstrates that under optimum conditions, our algorithm performs as quickly as the Kerberos algorithm, but in a worst-case scenario is slower than Kerberos. However, our algorithm's worsened performance is partially due to the fact that all nodes participate, whereas Kerberos will exclude those nodes that fail to have high encryption. Thus, the O(n) analysis shows the tradeoff between speed and security in Kerberos versus our own algorithm.

CHAPTER VII

FINDINGS

We used the C++ programming language to write the prototype simulations for our game theoretic network security optimization algorithm, as well as the Kerberos system. The network consisted of n = 43 nodes; this number was chosen because of its Gaussian distribution over a 10 by 10 matrix, but we will later examine varying node population to study the effects on network security at the end of this chapter. Constraints of our networks were related to hardware and software of each node, and were quantified numerically. For each prototype, the networks tested consisted of nodes that were general representations of heterogeneous computers, each with different hardware and software. The hardware and software differences between each node, and thus the differences in constraints, were represented numerically by positive real numbers. Any more specific hardware or software details beyond this general representation were not considered. In light of the fact that as discussed regarding constraints, we understand that there are a large number of hardware and software combinations that can result in different constraints, we felt that until we are able to confidently represent the actual numerical quantization of hardware and software, or even a subset of the domain of all computer hardware and software, the approach we would take is to follow from the definitions in equations (4.1) – (4.4). These functions take as input hardware and software of a node and map the domain of possible security values, which consist of the combination of all hardware and software, to a range of allowed security values for that particular node's hardware and software. This range of values is the node's constraints on security. We can confidently examine what is common and what is different among nodes according to equations (4.1) – (4.4). We refer to what is common among nodes as what is similar, and as such measures the homogeneity. We refer to what is not in common among nodes as what is different, and as such measures the heterogeneity. In our simulations, we can say for example node (1,7) is 36% similar to node (4,5). We can also say node (1,7) is 46% similar to the average of the constraints of the nodes of the network. The network consisted of 43 nodes with lowest possible minimum security of 1 and maximum of 10. Homogeneous minimum security means that all had minimum security of 1, and homogeneous maximum security means that all had maximum security of 10. Heterogeneous, or different, minimum security is defined as follows: thirty-three nodes started at minimum security of 1, seven nodes started at minimum security of 2, two nodes started at minimum security of 3, and one node started at minimum security of 5, for an average heterogeneous minimum security of 1.3. Results shown in the figures below are for the end of each iteration. Since all simulations began counting at zero, improvement at iteration zero is the improvement at the end of the "first" iteration versus previous improvement, which for that iteration was none.

The different prototypes one through four incorporated some or all of Algorithms 5.1 – 5.5 listed above; preliminary tests were made in prototype one before moving on to the intermediate prototype two and the complete prototypes three and four to model our game theoretic security optimization algorithm. Our initial tests of the first prototype were primarily confidence-building exercises to determine if our prototype was correct and the results made sense. In the second prototype, we implemented nearly all aspects of our game theoretic security optimization, with the exception of side payments. Prototypes three and four fully implemented our game theoretic network security optimization technique. The following table, Table 7.1, shows an overview of the similarities and differences between our prototypes. The details are below in the description of each prototype. See Table 7.4 for an overview on prototypes, networks, and results of security.

	Strategies	**Side Payments**	**Minimums**	**Maximums**	**Coalitions**	**Algorithms**
Proto 1	Pure	None	Hom./Het.	Hom.(10)/Het.(Min.+ 6 to 8)	N	5.2-5.4
Proto 2	Pure	None	Het.	Het.	Y	5.1-5.4
Proto 3.1	Pure	Effective	Het.	Het.	Y	5.1-5.4
Proto 3.2	Pure	More Effective	Het.	Het.	Y	5.1-5.4
Proto 4.1	Mixed	Effective	Het.	Het.	Y	5.1-5.3, 5.5
Proto 4.2	Mixed	More Effective	Het.	Het.	Y	5.1-5.3, 5.5
Kerb.	Pure	N/A	Het.	Het.	N/A	N/A

Table 7.1: Overview of all prototypes’ similarities and differences

Here, for example, we see that prototype three version two (labeled “proto 3.2”) had side payments that were able to decrease vulnerability to a greater extent than prototype three version one (labeled “proto 3.1”). In doing so, the utility for the side payment was increased. Likewise, we tested a similar scenario for prototype four, which contained a game where players moved according to mixed strategies. This is discussed in greater specification in each prototype below. Table 7.1 is meant to serve as an overview and as a guide for the reader to aid in sorting out the differences between the prototypes while examining the details below.

The networks considered in our tests varied according to heterogeneity. For our initial tests in prototype one, we used homogeneous networks whereby all nodes could reach the maximum security, followed by semi-heterogeneous networks with nodes having different minimum security from one another, but the same maximum security. The third type of network tested in prototype one contained nodes possessing both different minimum and maximum security, where the maximum security was a function of the minimum security. In the first network tested for prototype one, the maximum security of each node was equal to its minimum security plus six levels above it, followed by a second network with maximum security equal to the minimum plus seven, and a third network with maximum security equal to the minimum plus eight. Prototype one was primarily a confidence-building exercise, and as such we only tested it on these networks; these heterogeneous networks tested for prototype one were but a subset of the entirety of the heterogeneous networks tested on prototypes two through four. For the complete or nearly-complete implementations of our algorithm as done by prototypes two through four, as well as Kerberos, we tested only heterogeneous networks since the purpose of our algorithm is to optimize security of a heterogeneous network, and the purpose of Kerberos is to better evaluate how well our algorithm optimizes security. We will examine each prototype in detail and its results, and compare the results between them. After doing so, we compare all prototypes' performance with different network sizes, *n*.

7.1 Prototype One

The objective of our first prototype test is to examine utility improvement for individual nodes, and overall utility improvement. Prototype one was based on Algorithm 5.2: *Attack_tree_analysis*, Algorithm 5.3: *Reduce_ vulnerability*, and Algorithm 5.4: *Pure_strategy_game*. Algorithm 5.1 was not included as all nodes belonged to the same coalition from the beginning and no coalitions were formed during the game. An analysis of the output produced by our first prototype is shown below. The game is initialized at security level 1, all constraints and bids are assumed to have been resolved, and the utility function excluded side payments from calculations. As such it does not entirely implement our game theoretic security model, but will be used for preliminary tests before moving on to the intermediate second prototype and the final, complete model implemented by the third and fourth prototypes.

The simulation data from prototype one is found in Figures 7.1 through 7.13, as well as Tables 7.2 and 7.3. Some networks tested for prototype one were theoretically capable of reaching the maximum security level of ten, giving a homogeneous maximum. Other networks tested for prototype one had heterogeneous maximums, giving different maximum security levels for each node depending upon their minimum security level. These heterogeneous networks were a subset of the networks tested for prototypes two through four. All nodes were able to observe all $G = (V, E)$ and, consequently, moves in the game. Nodes had varying degrees of heterogeneity with respect to one another's security.

When we initially started testing, we chose two nodes to observe whether our test results were turning out in a way that made sense. After this confidence-building exercise, we switched to showing just our average network data, since the purpose of our work is to test whether our algorithm improves overall network security. Two candidate nodes are first examined, chosen at random, one with more heterogeneity compared to other nodes in the network (46% similar to the average of all security constraints of all nodes in the network, as described earlier according to equations (4.1) – (4.4)), and a second node with less heterogeneity (37% similar to the average of all security constraints of all nodes in the network, as described according to equations (4.1) – (4.4)). These two nodes differed from one another, with respect to each other's security constraints, by 36%, as according to equations (4.1) – (4.4). Figure 7.1, shown on the next page below, gives utility improvement for these two candidate nodes.

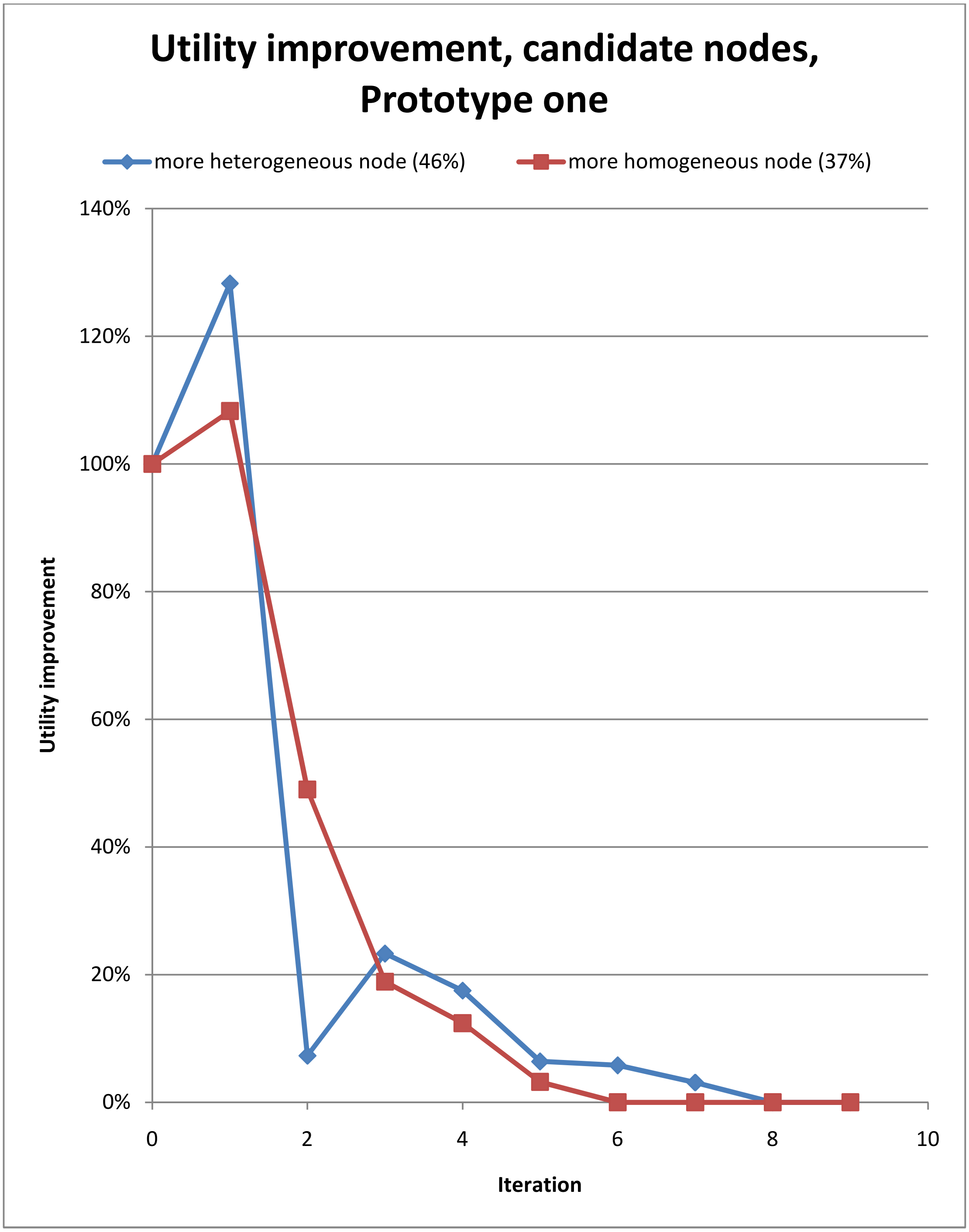


Figure 7.1: Utility improvement, candidate nodes, network consisting of nodes with a homogeneous minimum security level, prototype one, n = 43

Utility improvement is studied because it shows the improvements in security during game play, which aids in evaluating the performance of our game theoretic algorithm. These results show greater initial improvement by the heterogeneous node (130% vs. 110%), but both have comparable utility improvement after the first few iterations. This pattern indicates stabilization for the nodes and demonstrates equilibrium for varying degrees of heterogeneity.

Next we examine the result for average utility improvement for all nodes in the network, shown in Figure 7.2 below. We can compare these results with the above chart to see that the utility improvement of the two candidate nodes stabilized at the same number of game turns (iterations), but the initial utility improvement of the candidate nodes was greater than the average of the network utility improvement. The result shown in the graph confirms that the utility improvement demonstrated by the two candidate nodes is representative of the network average. Security levels are compared similarly in Figures 7.3 – 7.5.

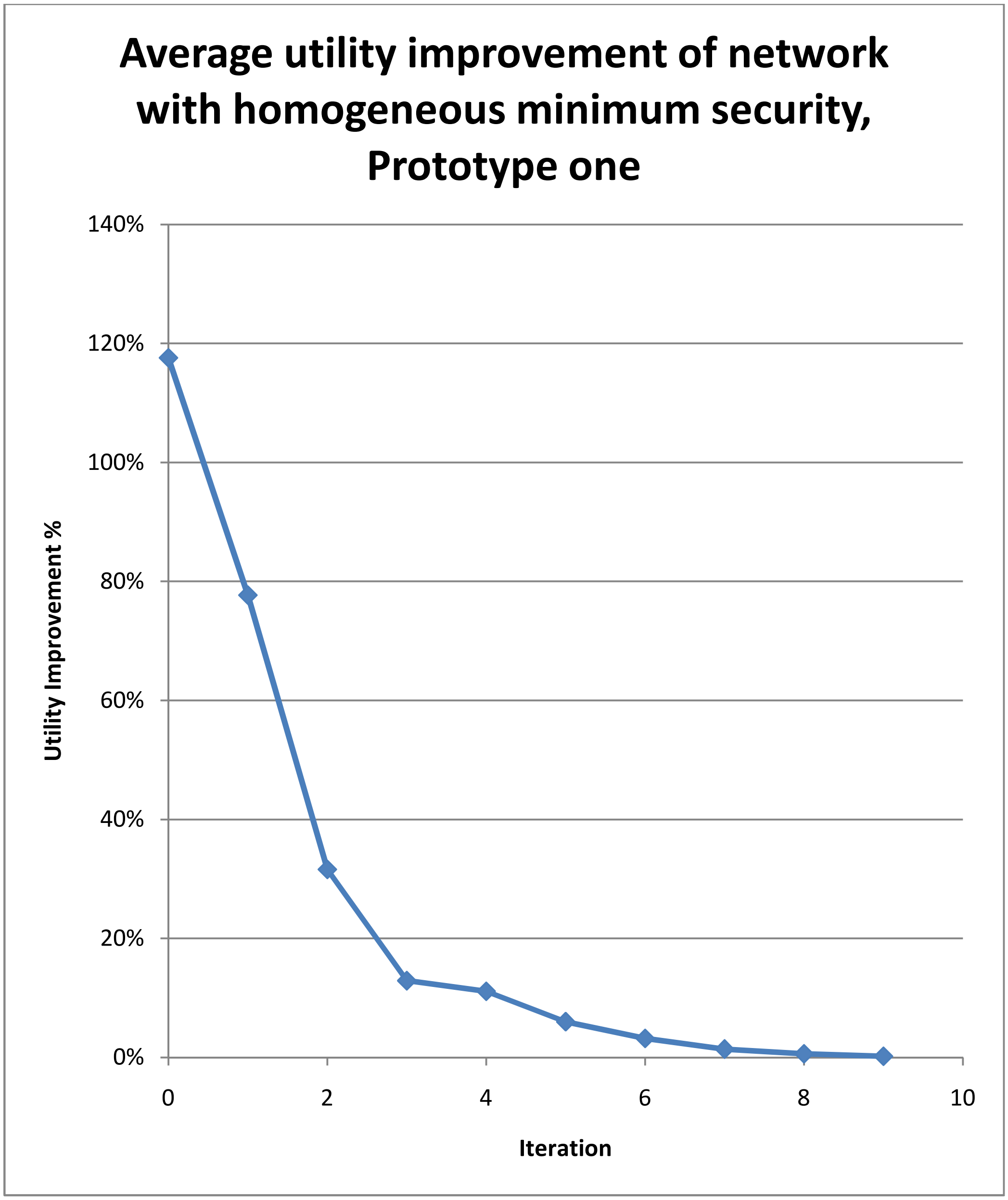


Figure 7.2: Network where all nodes had a homogeneous minimum security level, prototype one, *n* = 43. This is the average security of the entire network which contained the two candidate nodes in Figure 7.1

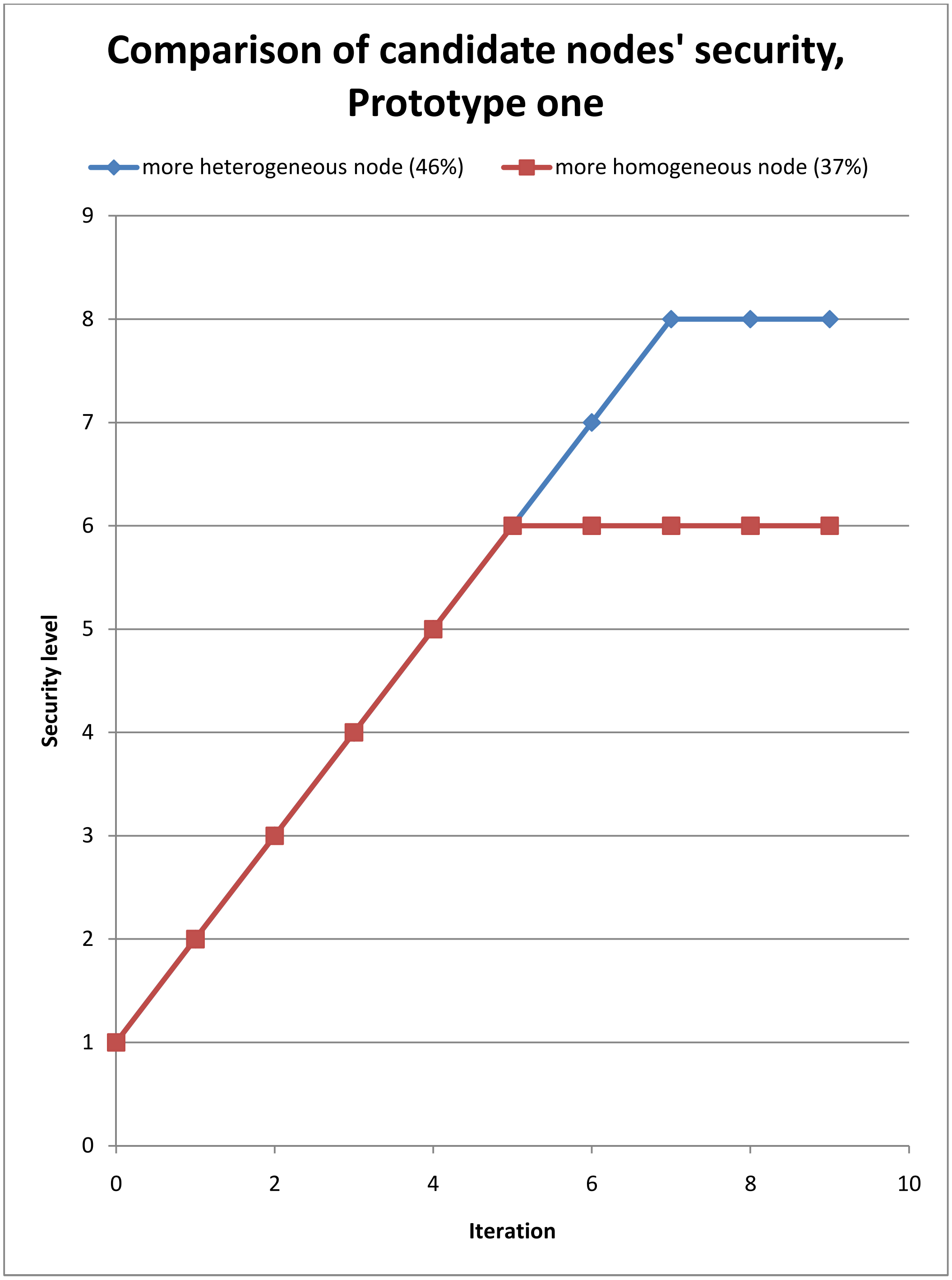


Figure 7.3: Candidate nodes security level (1 - 10), homogeneous minimum security level, prototype one, *n* = 43

The above chart compares the security of the two candidate nodes. We see that the more heterogeneous node achieved a greater maximum security at stabilization, but took longer to stabilize than the more homogeneous node which had less security. As we will see later in prototype four, there exists a correlation between extended convergence time and greater security. Our initial tests of prototype one only hinted at this possibility at this point.

The next chart, Figure 7.4, shows the average security for the entire network, which contained the candidate nodes. This is the same simulation, but different data, in the above figures. If we compare the chart below showing the average network security to the one above showing the individual candidate node security, we observe that the average network security is slightly higher than if we calculated the average security between the two candidate nodes. These charts illustrated to us that our network and algorithm's results made sense, and we began to move on to testing our prototype on networks with increasingly greater heterogeneity. By same minimum we mean homogeneous minimum of security of 1 for all nodes in the network, and by different minimum we mean heterogeneous minimum of security for the nodes forming the network, as defined at the start of this chapter.

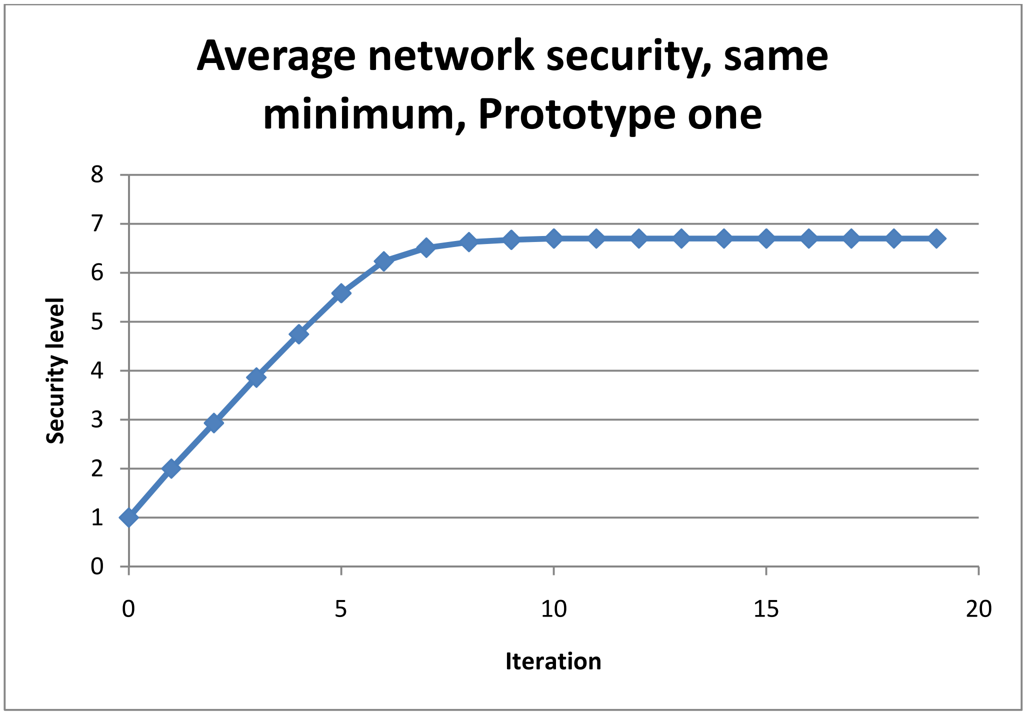


Figure 7.4: Average security, same (homogeneous) minimum security level, n = 43

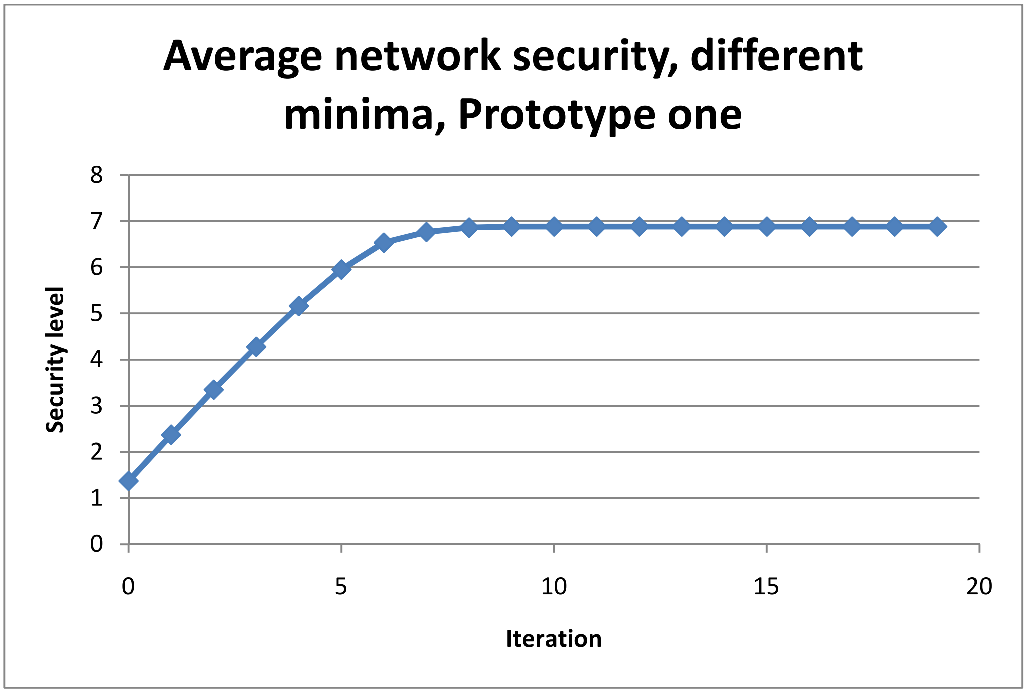


Figure 7.5: Average security, different (heterogeneous) minimum security levels, n = 43

The chart above, Figure 7.5, represents data for new simulations from in the previous figures. It shows the average network security for a new network with heterogeneity; the nodes had different, but not necessarily unique, minimum security. In the next chart we see this data compared to the security of the previously-tested homogeneous network. This chart combines data in Figure 7.4 and 7.5, the initial tests of prototype one.

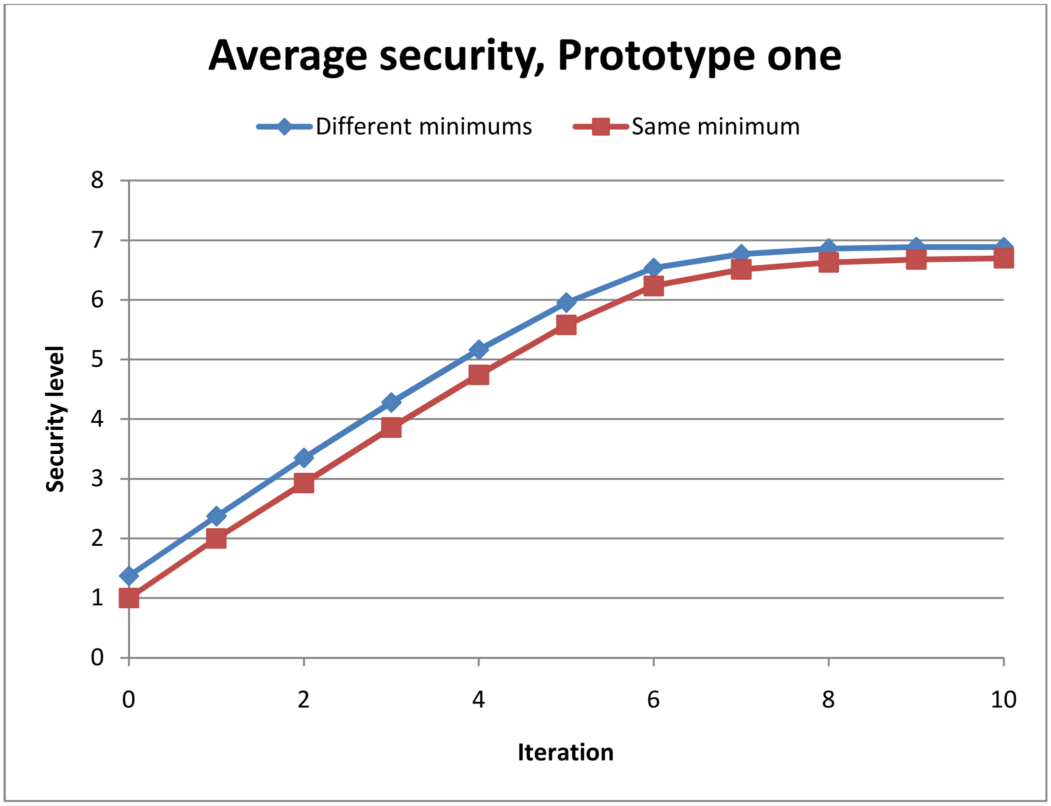


Figure 7.6: Comparison of average security for network with homogeneous minimum (same minimum) vs. heterogeneous minimum (different minimum) security, *n* = 43

Figure 7.6, shown above, illustrates that the more heterogeneous network starting at different minimum security reached a somewhat higher level of security (6.9) than the network that had the same minimum security (6.7). We hypothesize that greater differences might lead to even higher security, which will be examined in further tests. However, this graph illustrates that our work is applicable to any level of security and will give similar results regardless of a node's achievable security level.

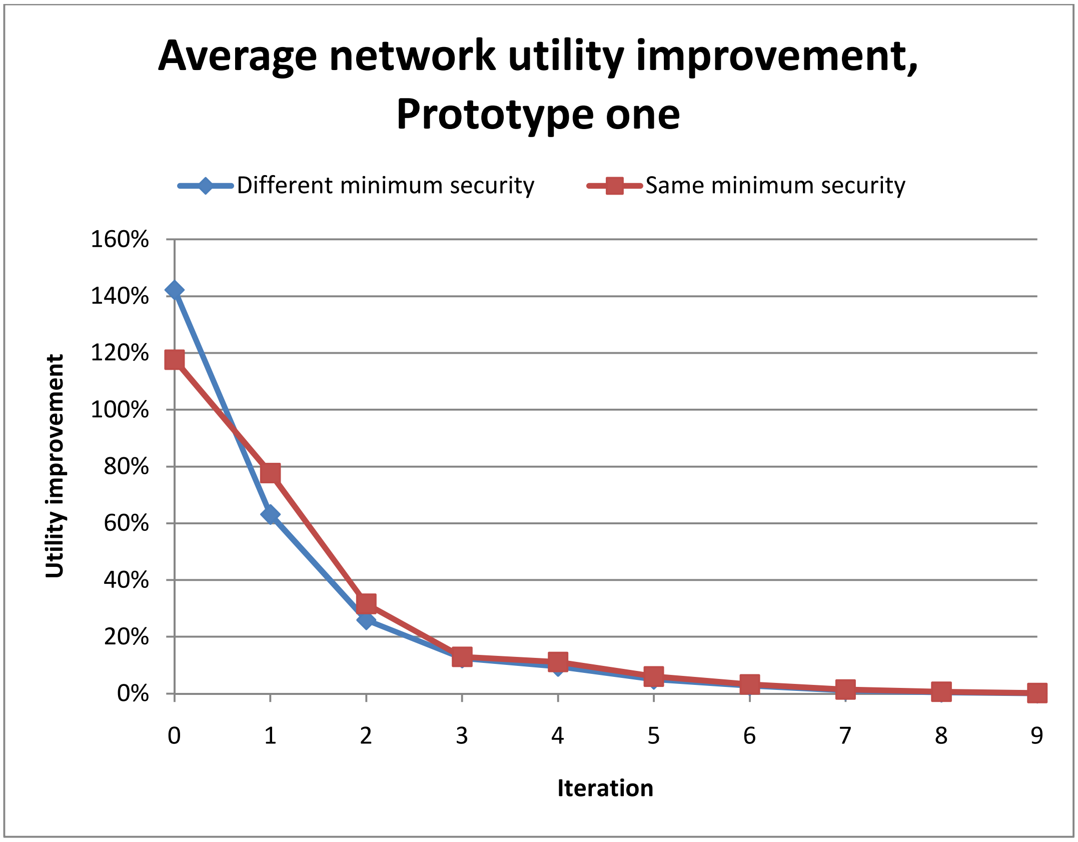


Figure 7.7: Average utility improvement for networks with different minimum security level, prototype one, n = 43

Figure 7.7, shown above, illustrates the difference in utility improvement when all nodes start at different minimum security versus the same minimum security, as in Figure 7.6. The graph of the network containing nodes starting at different minimum security converged somewhat similarly to the one starting at the same minimum security.

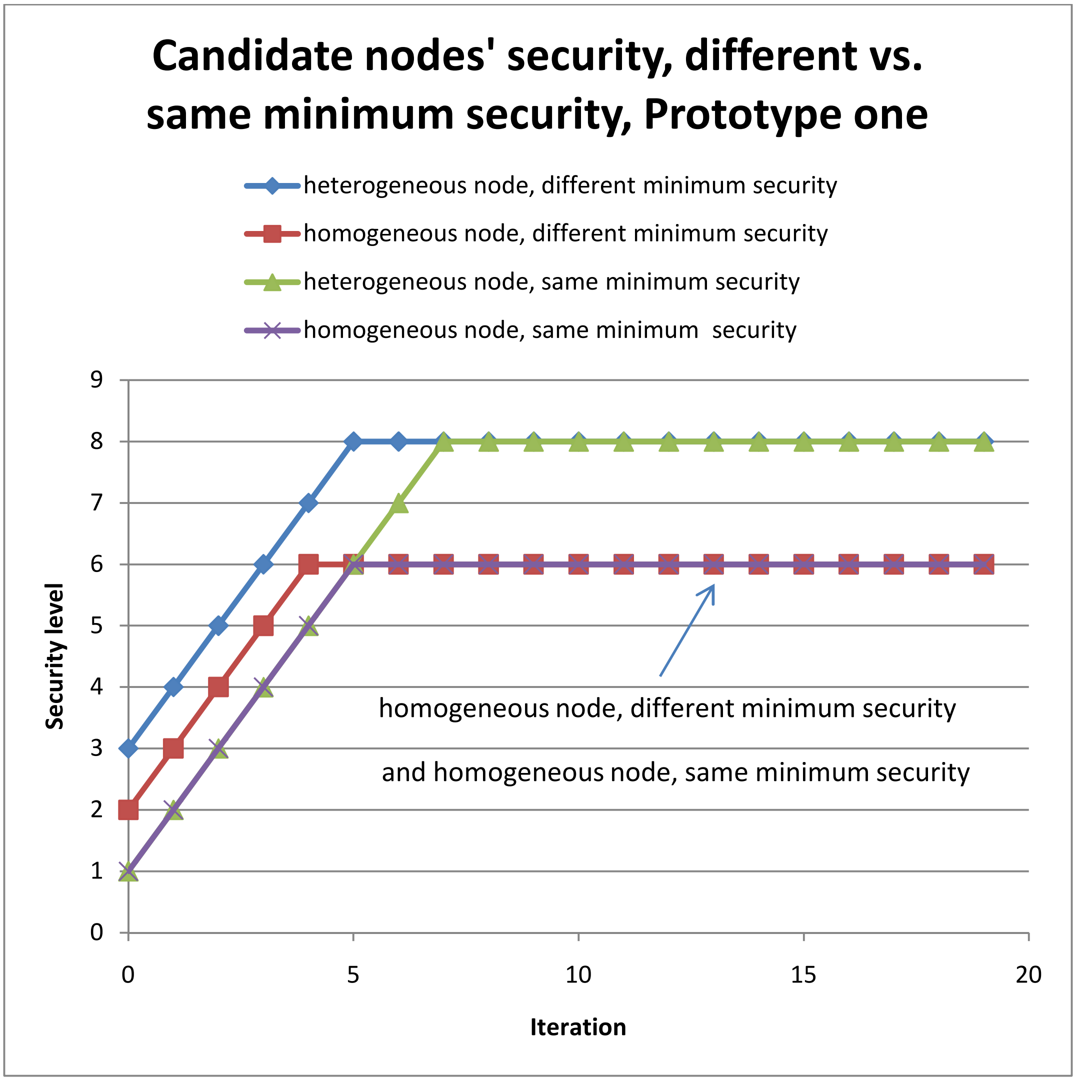


Figure 7.8: Security level comparison, candidate nodes, different versus same minimum security (prototype one, *n* = 43)

Figure 7.8, shown above, returns to showing the performance of the two candidate nodes and relates them to the network variations in which all nodes have the same or different minimums. In Figure 7.8, we examine our two candidate nodes' performance in networks where they had homogeneous (same) minimum security and networks with heterogeneous (different) minimum security. Both sets of nodes, the set with the different minimum security and the set with the same minimum security, reached the same overall maximum securities, respectively. The homogeneous node with a different minimum security converged to the same maximum security by one iteration faster than its counterpart with the same minimum security. It is interesting to note that the faster convergence corresponds to the higher security level from which the network started.

Since our tests of our candidate nodes showed similarity to the average network security and utility, we were confident that our algorithm was working as expected. We moved on to the focus of our work, that of evaluating the improvement in average network made by our algorithm. In the next chart, Figure 7.9, we see the utility improvement for networks with varying minimum and maximum security. This chart takes the networks shown in Figure 7.7 and adds tests of prototype one on a third network containing nodes with different minimum and maximum security. This new network was our first test of any truly heterogeneous network, as each node had a different minimum and maximum security compared to the other nodes.

For this initial test, we did not want the maximum to be overly different from the other networks, so we chose a heterogeneous maximum which gave some nodes the capability to reach the same maximum as the nodes in the other networks (a theoretical maximum of 10), and made other nodes so they reached a lower maximum. However, this being our first test of a network with this degree of heterogeneity, we did not want the nodes starting out at a minimum security of 1 to be capable of less than 70% of the theoretical maximum as the nodes in the other networks. Therefore, we established the theoretical maximum security of each node to be 6 levels of security above a node’s minimum security, enabling all nodes to have a theoretical maximum of at least 7. For an average minimum security of 1.3, the average theoretical maximum security of the network was calculated to be 7.3, or 6 levels of security above the minimum of 1.3. Regardless of a node’s minimum security, it could not exceed a maximum theoretical security of 10. The performance of the new network with greatest heterogeneity was almost identical to that of the previous networks in Figure 7.7.

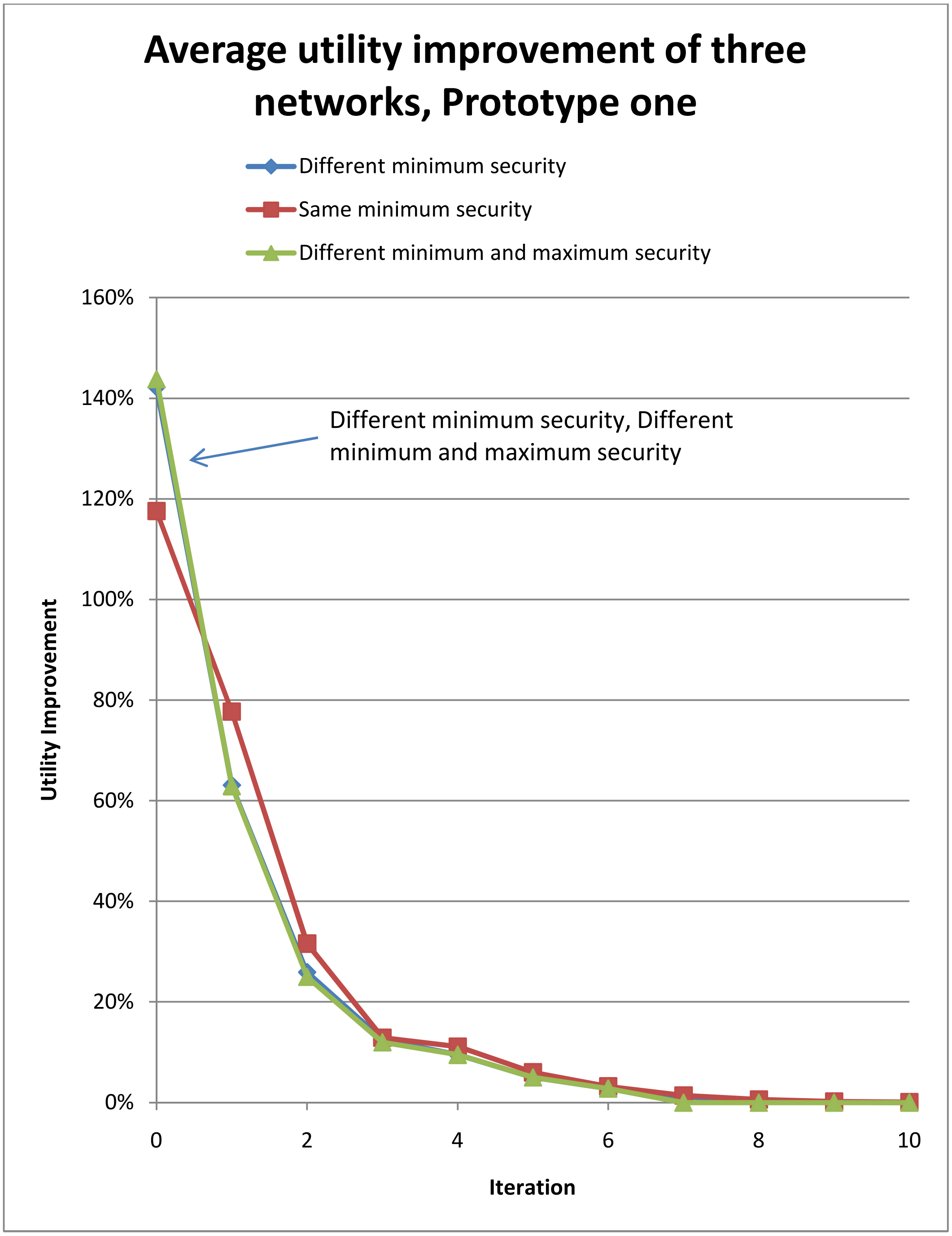


Figure 7.9: Average utility improvement comparison, different versus same minimum and different maximum security (prototype one, *n* = 43)

In Figure 7.10, shown below, we see the average security of the same three networks from Figure 7.9. Since the network containing nodes with different minimum and maximum security (most heterogeneity) had slightly lower security than the network where all nodes could reach the same maximum security, lowering maximum security for some nodes affects the maximum network security. This is consistent with what we would expect. However, our algorithm still maximizes overall security to a quantity that is comparable to the network where all nodes can reach the same maximum. More importantly, at six iterations the new network with the most heterogeneity converges at least 33% faster than the other two networks with less heterogeneity. These results indicate that the algorithm is robust and actually improves as networks become more heterogeneous, causing faster convergence as the heterogeneity increases.

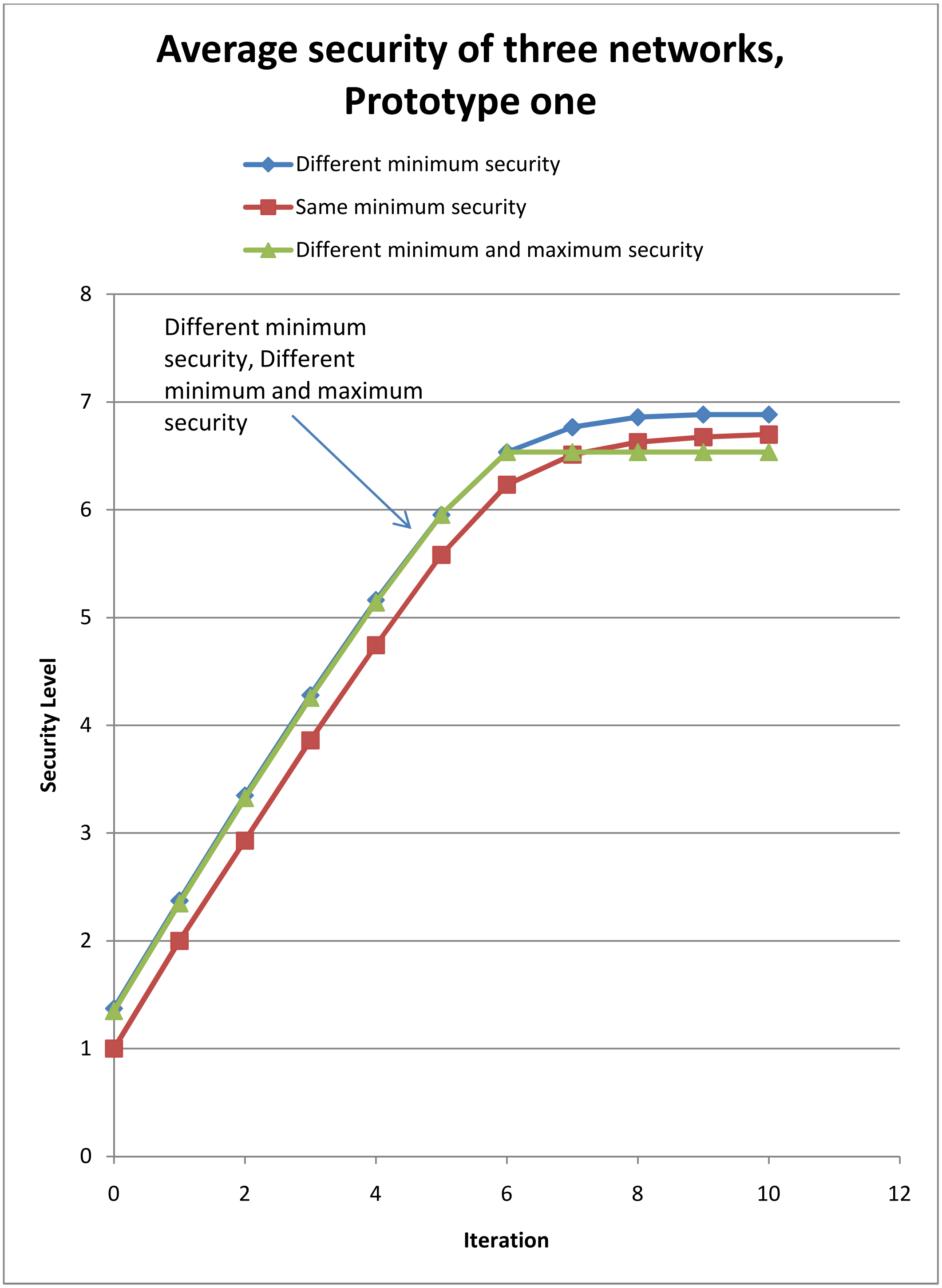


Figure 7.10: Comparison of average security for same minimum vs. different minimums vs. different minimums and maximums (prototype one, *n* = 43)

As in Figure 7.9, Figure 7.10 demonstrates that the network with heterogeneous minimum and maximum security matched almost perfectly that of the heterogeneous minimum and homogeneous maximum network, but only until six iterations are reached. At that point the heterogeneous minimum and maximum security network stabilizes and the other networks continue to improve. The most heterogeneous network, having different minimum and maximum security, reached the lowest maximum average security of 6.5, while the next most heterogeneous network, having the same minimum and different maximum security, reached the highest maximum security of 6.9. The network with the most homogeneity, having the same minimum and maximum security, was between the two other networks with maximum average security equal to 6.7.

We examined the actual versus theoretical performance of the networks in order to determine how well our algorithm is improving overall network security for a heterogeneous network with different minimum and maximum security. This test was another confidence-building exercise prior to moving on to testing our algorithm on exclusively heterogeneous networks. Since this heterogeneous network tested will have a lower achievable security than a network where all nodes can theoretically reach the upper limit of security (10), comparison of average security values does not reflect how well the heterogeneous network is performing. Figure 7.11 shows the actual performance relative to the theoretical performance of the three networks we have evaluated. Since the most heterogeneous network, having the different minimums and maximums, demonstrated an actual average security level of 6.5 out of a theoretical average of 7.3 (6 greater than the average minimum of 1.3), its performance far exceeded that of either of the previous two networks, which had a theoretical maximum of 10.

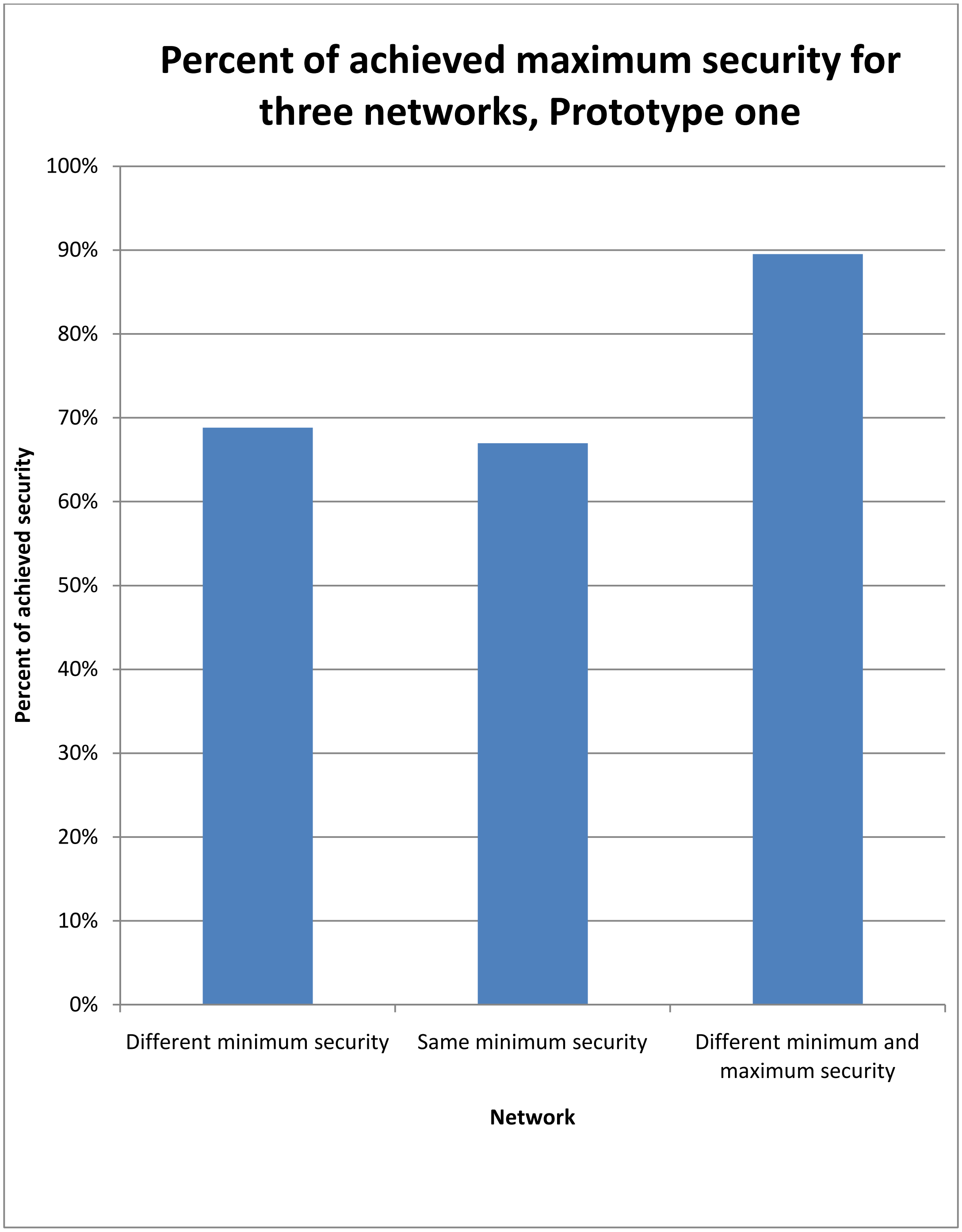


Figure 7.11: Comparison of maximum achievable vs. achieved security for networks with varying heterogeneity (prototype one, n = 43)

The next chart, Figure 7.12, shows results for the networks from in Figure 7.9 and adds a fourth network containing nodes with different minimum and a higher heterogeneous maximum security. We label the networks from above: “Different minimum security” is “Heterogeneous minimum,” “Same minimum security” is “Homogeneous minimum security,” and “Different minimum and maximum security” is “Max = hetero min +6.” The latter network name, “Max = hetero min +6,” identifies what the maximum security is for each node in that network in order to better distinguish that network from the fourth network we add to our examinations below. The “heterogeneous min” refers to the heterogeneous minimum security of the nodes forming that network; in other words, the nodes in the network have minimum security according to the statistics of minimum security as described at the start of this chapter. For example, for a node in the network of “Max = hetero min +6,” if the node has minimum security of 1, then its maximum security is 1 + 6 = 7. No node has security greater than 10 regardless of its minimum. The introduction of the fourth network is to test whether increasing the maximum heterogeneous security yields a better tradeoff between speed and security versus the third network. Like the third network, the fourth network contains nodes with varying minimum and maximum security. However, the fourth network enables a higher heterogeneous maximum than the nodes in the third network. With the third network, the heterogeneous maximum security was equal to the six levels above the heterogeneous minimum, or in other words, was equal to the heterogeneous minimum plus six. The fourth network has heterogeneous maximum security equal to the heterogeneous minimum plus seven. For example, nodes that had minimum security of 1 were able to reach a theoretical maximum of 8.

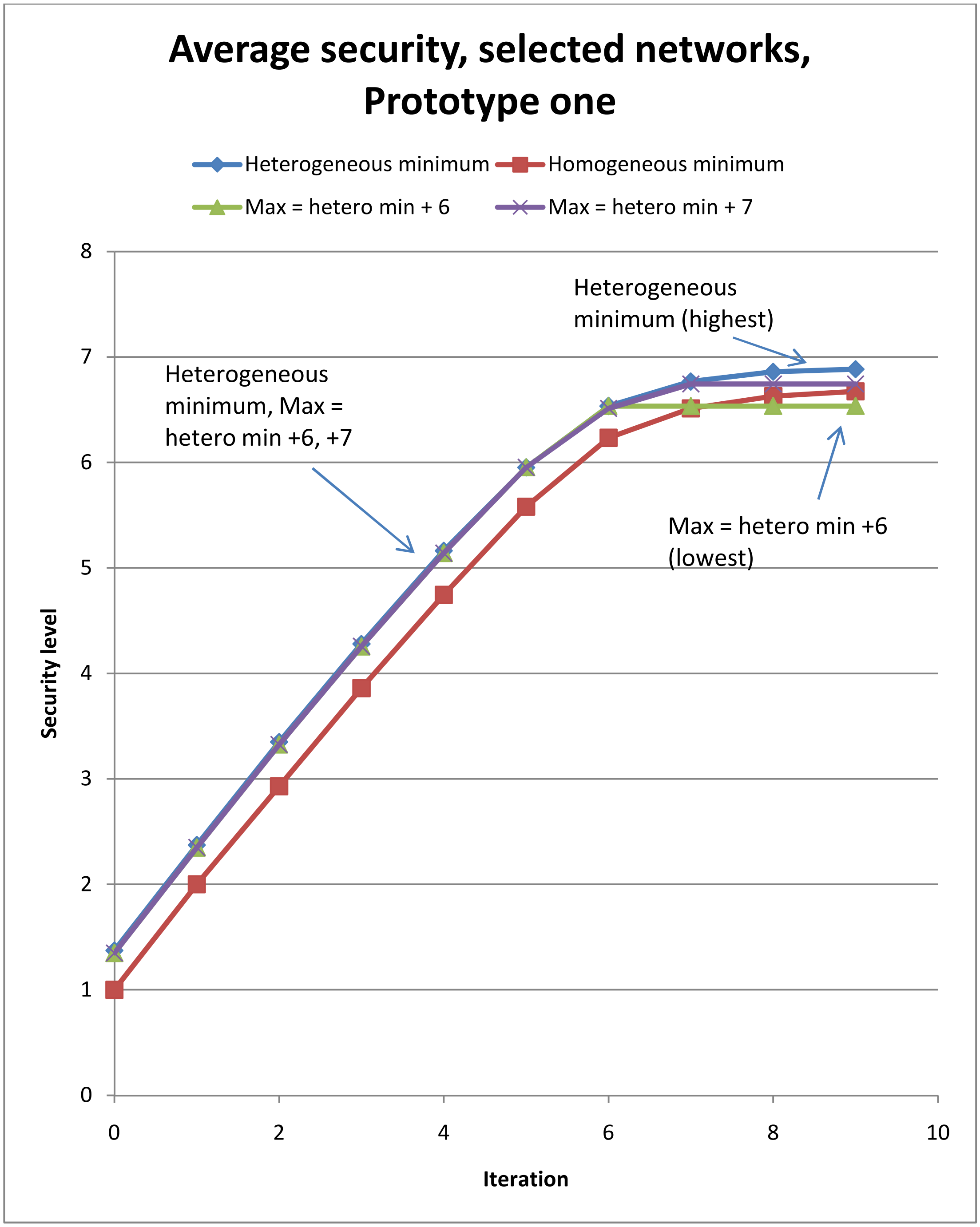


Figure 7.12: Comparison of maximum overall network security, selected networks with varying heterogeneity and security (prototype one, *n* = 43)

The results in Figure 7.11 and 7.12 show that the new network, which increased the maximum possible security to seven greater than the heterogeneous minimum, gave an increased average maximum from 6.5 to 6.7. However, the performance of the new network was decreased as the third network (maximum security equal to minimum plus six) converged faster.

For additional tests, we added a fifth network with increased heterogeneous maximum security, where the maximum security was equal to the heterogeneous minimum plus eight. As shown below in Table 7.2, the new network yielded nearly identical results to the heterogeneous minimum network. All nodes in the heterogeneous minimum network were capable of reaching the theoretical maximum security of 10; this network had been labeled “different minimums” in earlier charts, and is re-labeled “heterogeneous minimum” for comparison in subsequent tables and figures. These results of varying maximum security, as shown in Table 7.2, indicate that there exists an optimum network configuration for fastest convergence with the maximum average network security. However, the heterogeneity of the new network still paid off in terms of maximum security, as the network achieved a maximum average of approximately 6.9, the highest of the tests thus far.

Average security comparison, Prototype one					
Iteration	**Heterogeneous minimum**	**Homogeneous minimum**	**Max = hetero min + 6**	**Max = hetero min + 7**	**Max = hetero min + 8**
0	1.372	1.000	1.349	1.349	1.349
1	2.372	2.000	2.349	2.349	2.349
2	3.349	2.930	3.326	3.326	3.326
3	4.279	3.860	4.256	4.256	4.256
4	5.163	4.744	5.14	5.14	5.14
5	5.953	5.581	5.953	5.953	5.953
6	6.535	6.233	6.535	6.512	6.512
7	6.767	6.512	6.535	6.744	6.744
8	6.86	6.628	6.535	6.744	6.86
9	6.884	6.674	6.535	6.744	6.907
10	6.884	6.698	6.535	6.744	6.907

Table 7.2: Comparison of average security and iteration time for the varying degrees of maximum security (prototype one, *n* = 43)

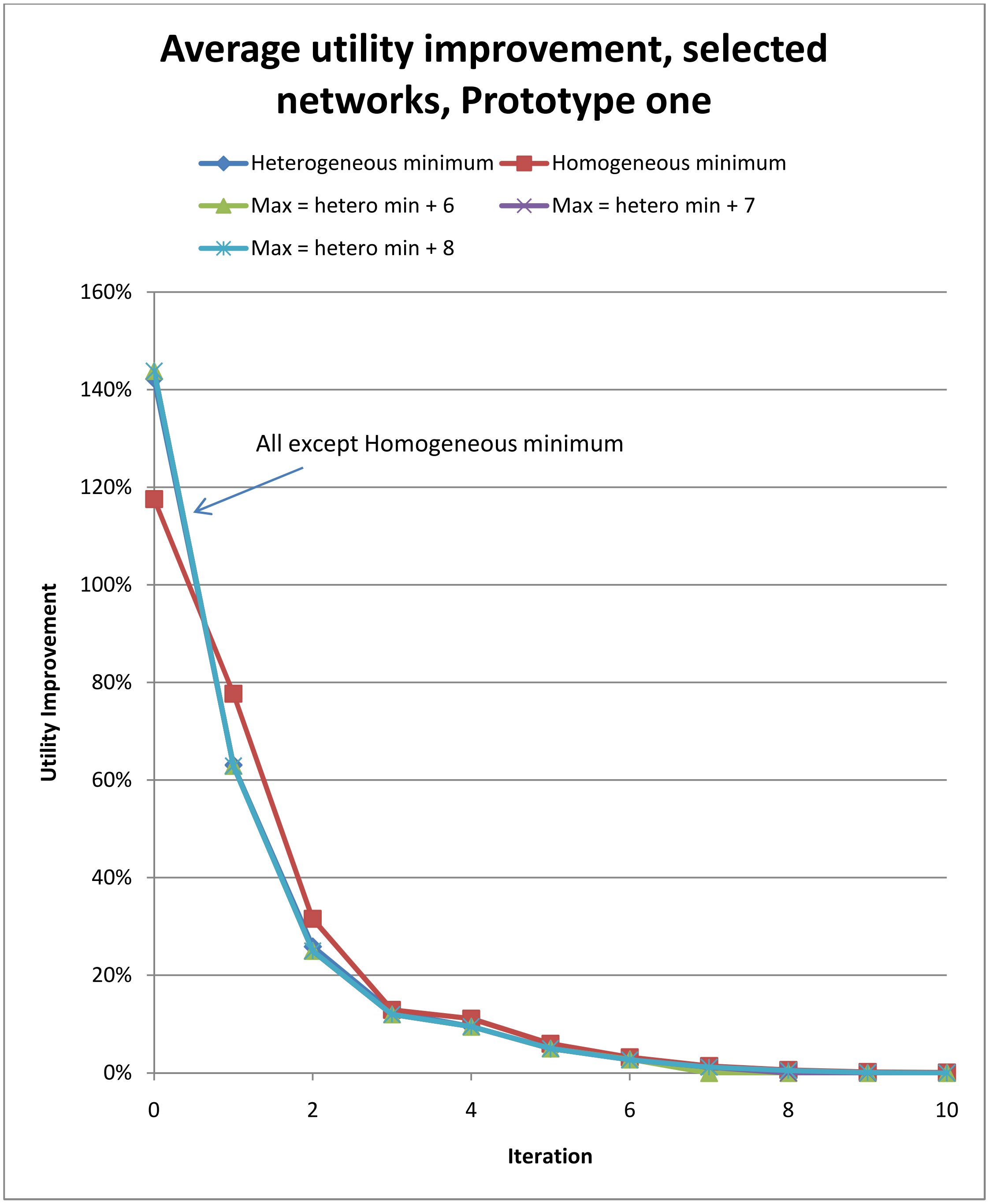


Figure 7.13: Average Utility Improvement comparison, varying security and heterogeneity (prototype one, *n* = 43)

Average utility improvement comparison, Prototype one					
Iteration	**Heterogeneous minimum**	**Homogeneous minimum**	**Max = hetero min + 6**	**Max = hetero min + 7**	**Max = hetero min + 8**
0	142%	118%	144%	144%	144%
1	63%	78%	63%	63%	63%
2	26%	32%	25%	25%	25%
3	13%	13%	12%	12%	12%
4	10%	11%	10%	10%	10%
5	5%	6%	5%	5%	5%
6	3%	3%	3%	3%	3%
7	1%	1%	0%	1%	1%
8	0%	1%	0%	0%	1%
9	0%	0%	0%	0%	0%
10	0%	0%	0%	0%	0%

Table 7.3: Average utility improvement comparison, varying degrees of heterogeneity. This is the same data as is presented in Figure 7.13.

In the above Table 7.3 and, in graphical form, Figure 7.13, we see nearly identical utility improvement performance for the varying degrees of heterogeneous maximum security, nearly matching that of the homogeneous maximum where all nodes could reach the same maximum of 10. While a security and convergence optimization occurred for networks with heterogeneous maximum plus seven or plus eight, this was not the case for utility improvement as shown in Figure 7.13 and Table 7.3. Here, networks performed similarly. These results ultimately show that the game theoretic algorithm presented in this work gives robust performance in terms of utility for varying degrees of maximum achievable security on the part of the nodes that make up the network, demonstrating that implementing our algorithm is feasible in terms of security improvement versus cost, as utility is largely unaffected by network heterogeneity.

7.2 Prototype Two

The objective of our second prototype is to test the effect on convergence time, utility improvement, and maximum overall security for a nearly complete implementation of our game theoretic security algorithm. Networks have greater and varying degrees of heterogeneity versus those tested for prototype one, which was a preliminary model. Nodes forming the networks have heterogeneous minimum and maximum security, which is representative of our heterogeneous network to be optimized. When performing our initial tests on our algorithm in prototype one, we used two networks with homogeneous and nearly-homogeneous nodes to observe whether our results were logical. After we were confident that our algorithm was working properly, we moved on to tests with truly heterogeneous networks. For our tests of prototype two, we used heterogeneous networks which included the networks from the latter tests of prototype one, and added networks with a maximum security equal to three, four, five, and six levels above the heterogeneous minimum security of each node. We will use these results for prototype two as a basis for comparison to the complete implementation of our algorithm in prototype three.

Like prototype one, prototype two does not entirely implement our game theoretic security model. However, with the exception of side payments, prototype two completely implements our game theoretic security model. Thus our second prototype implements Algorithm 5.3: *Reduce_ vulnerability*, with the exception lines 13-17, which relate to side payments. As it includes all else in Algorithm 5.3, as well as Algorithm 5.2: *Reduce_ vulnerability* and Algorithm 5.1: *Coalition_formation*, prototype two includes coalitions. Furthermore, in the prototype simulations all nodes have knowledge of all other nodes in $G = (V, E)$ and are thus able to observe events in the game in the same way that the nodes were able to for prototype one; all nodes in prototype two correspondingly have knowledge of nodes' maximum bidding price which in effect resolves the bidding scheme in our prototype from any iterative, logarithmic, or exponential process to immediate bid maximization. Also, as mentioned above each node has varying minimum and maximum security levels. This varying security and heterogeneity is representative of the heterogeneous network we seek to optimize; thus the networks examined in prototype one are a subset of the ones examined henceforth. The nodes are otherwise the same as the simulations for prototype one, with a network again consisting of $n = 43$ nodes.

As described in Algorithm 5.1: *Coalition_formation* and Algorithm 5.2: *Attack_tree_analysis*, at each turn of the game a node must re-examine its security and that of its neighbors through attack tree analysis and by applying the partition function. Those nodes that meet the security requirements for joining a coalition are added, those that no longer meet the requirements are removed, and the others are kept within the coalition. Those nodes that meet the security criteria and fall within *difference_threshold* from Algorithm 5.1: *Coalition_formation*, will join the coalition and be given access to insensitive data. Through the attack tree analysis of Algorithm 5.2, those nodes whose security is evaluated to be even higher are granted access to sensitive data. The simulation data for results from prototype two is displayed in Figures 7.14 through 7.18 below. In the first graph of prototype two, shown in Figure 7.14, we see average security for the network compared for variations on maximum achievable security; compare this to prototype one results from Figure 7.12.

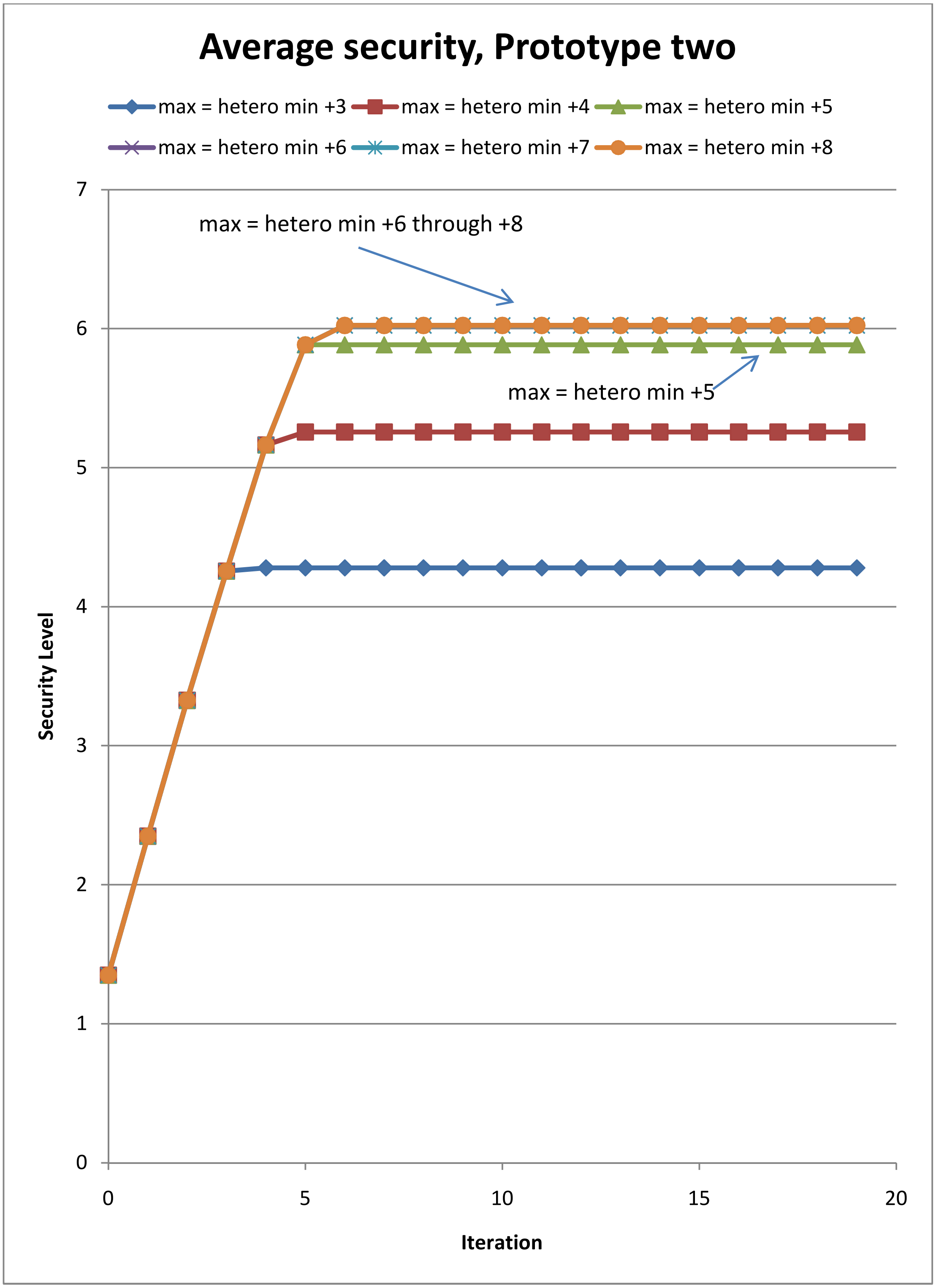


Figure 7.14: Average security comparison for varying maximum security, full prototype minus side payments (prototype two, *n* = 43)

Comparing the security data for prototype two shown in Figure 7.14 versus prototype one in Figure 7.12, we see a nearly consistent maximum for prototype one among all variations on heterogeneous maximum available security, reaching maximum of approximately 6.5 to 6.9. However, with prototype two there exists a divergent maximum among all variations on heterogeneous maximum achievable security for the network. In this graph, Figure 7.14, we see the difference achieved by the addition of coalitions, bidding, and analysis according to Algorithms 5.2 through 5.3. While the best two networks that had maximum security equal to minimum plus five through plus eight were slightly lower than that of prototype one, having a maximum of approximately 6.0, this is not a huge divergence from the average maximum of the same networks for prototype one at approximately 6.7. Because of the similarity, we believe it demonstrates the consistency of the foundation of our game theoretic algorithm for improving overall network security for the heterogeneous network. Furthermore, we believe that the marked difference between the performance on the network maximum plus four versus plus five shows that there is a point at which the optimum security can be reached, and that compressing the overall maximum security for the network has a detrimental effect on overall security, but only to a point, at which the algorithm is able to perform nearly as well as if all nodes had the same homogeneous maximum.

In Figures 7.15 through 7.18 shown below, we see the data generated for prototype two regarding relationships between nodes. Figure 7.15 shows average local neighborhood count and in the next chart, Figure 7.16, we see average utility improvement for the same simulation. Figure 7.17 shows corresponding information regarding the number of nodes forming a coalition, and Figure 7.18 the percentage of coalition members with sensitive data access. We examined the number of nodes to which a node is connected, number of nodes forming a coalition, and percentage of coalition members granted sensitive data access in order to better evaluate what is going on in the system, with respect to whether there exists any correlation between these statistics and network security. Also, by examining the number of connections, we are able to ensure the network is staying fully connected with the introduction of coalitions in prototype three. We are able to examine whether all nodes in a coalition are granted sensitive data access, or what percentage of nodes were granted sensitive data access. Recall that sensitive data access is granted to coalition members who meet more stringent security requirements than those to simply belong to a coalition. As it turns out, most, but not all coalition members are granted sensitive data access. Analysis and discussion of the data in these figures follows Figure 7.18 below.

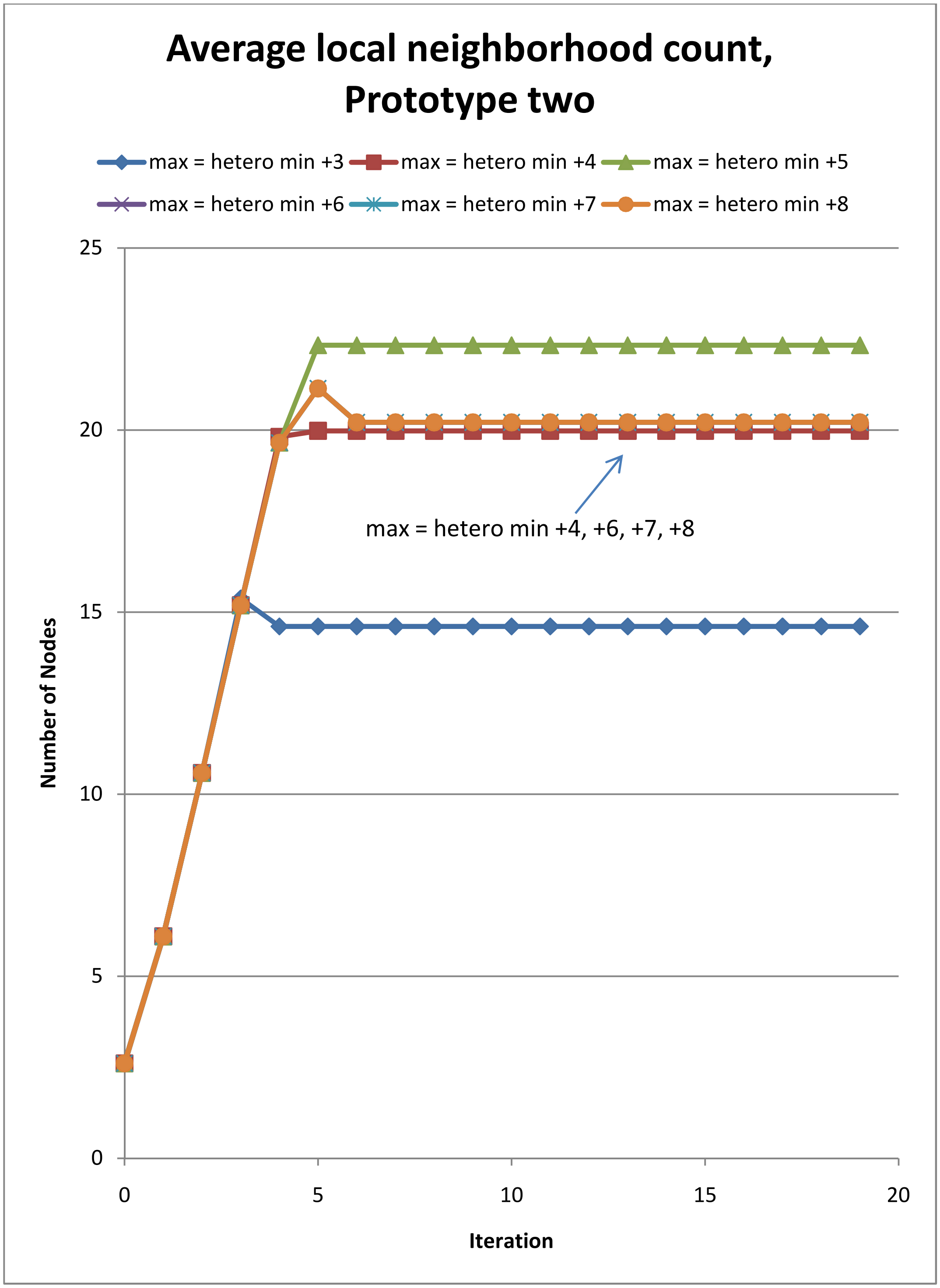


Figure 7.15: Comparison of average number of nodes making up local neighborhood for varying maximum security and heterogeneity, full prototype minus side payments (prototype two, *n* = 43)

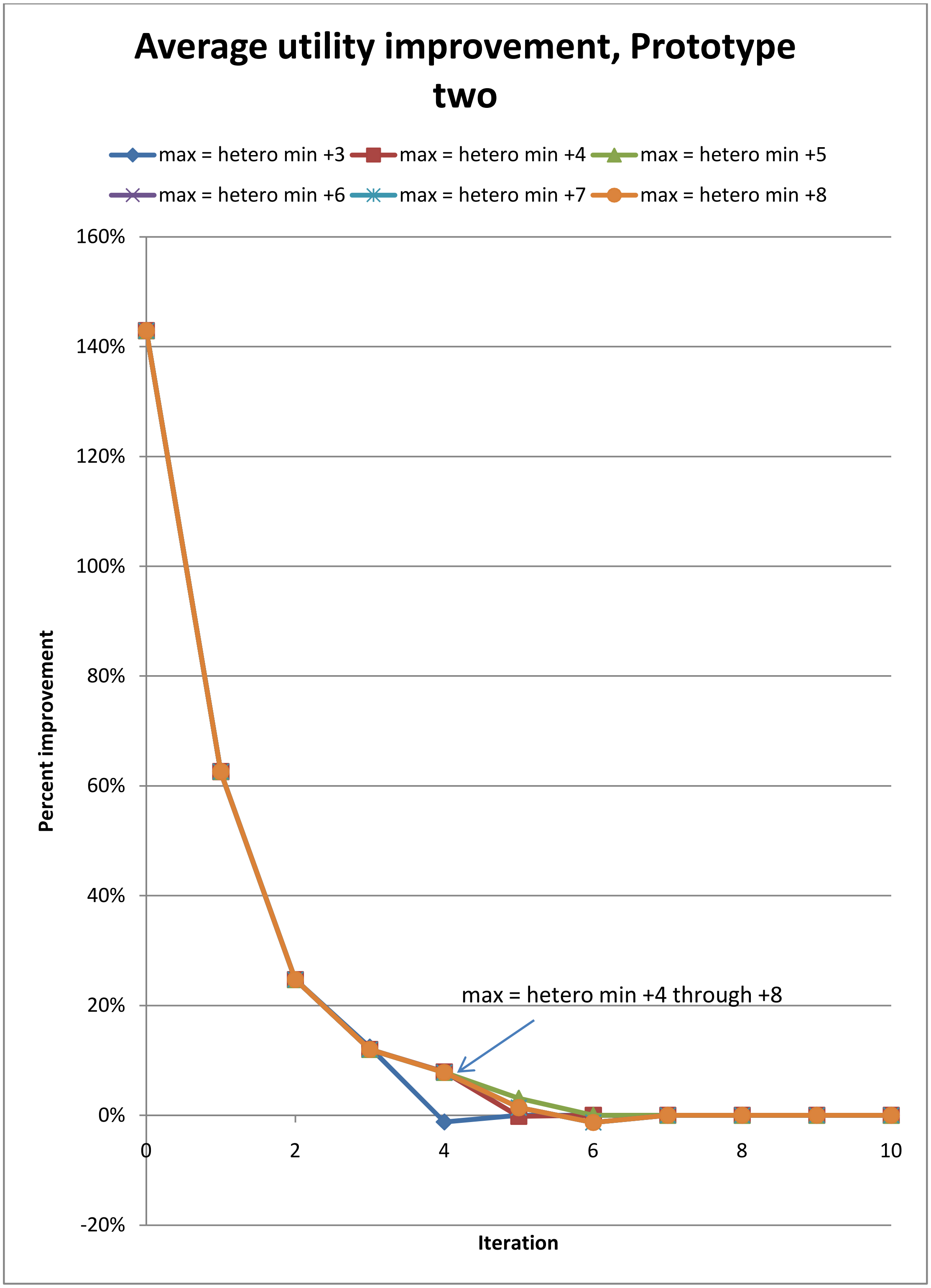


Figure 7.16: Average utility improvement comparison, varying maximum security and heterogeneity, full prototype minus side payments (prototype two, *n* = 43)

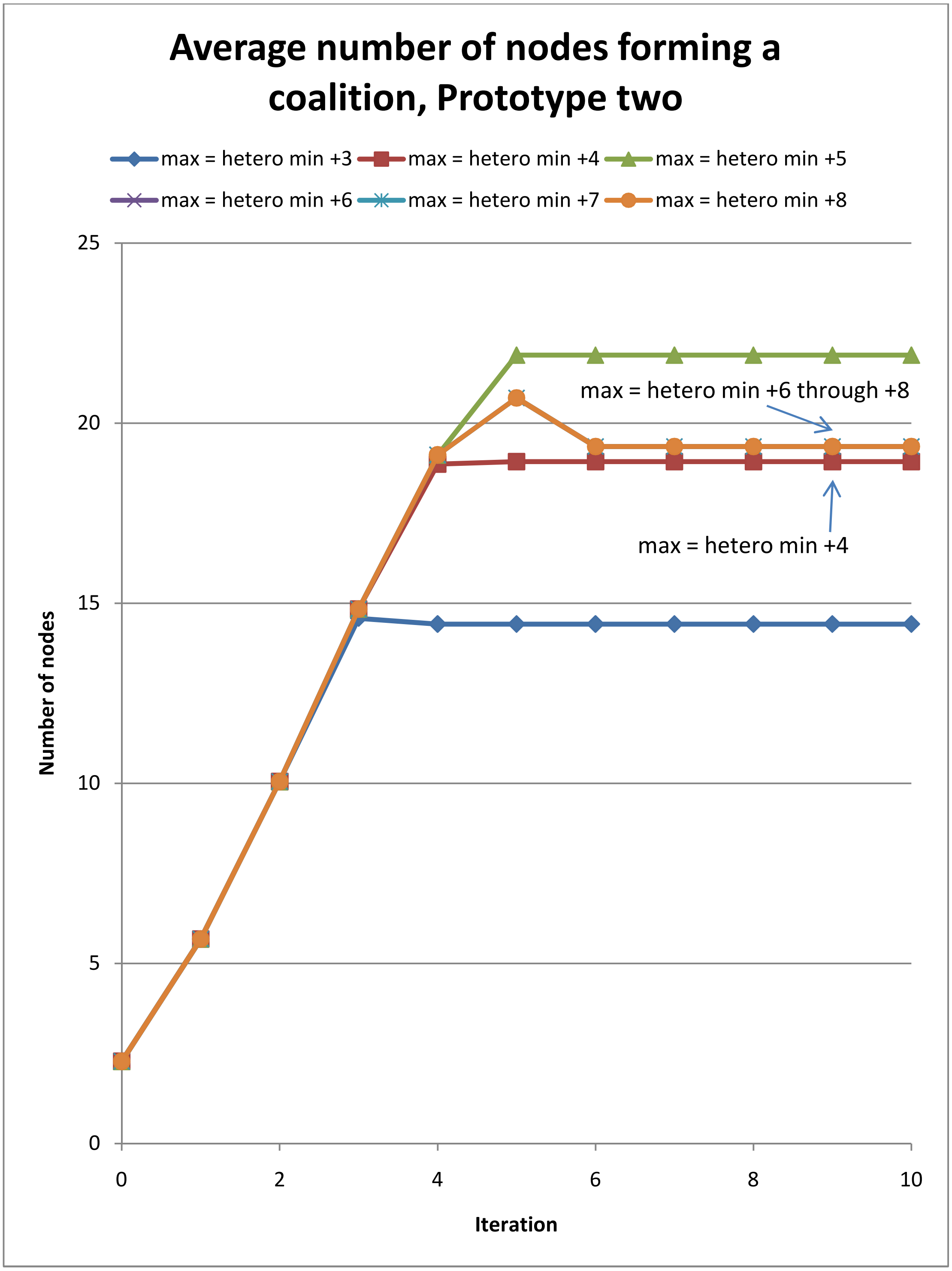


Figure 7.17: Comparison of average number of nodes forming a coalition, varying maximum security and heterogeneity, full prototype minus side payments (prototype two, *n* = 43)

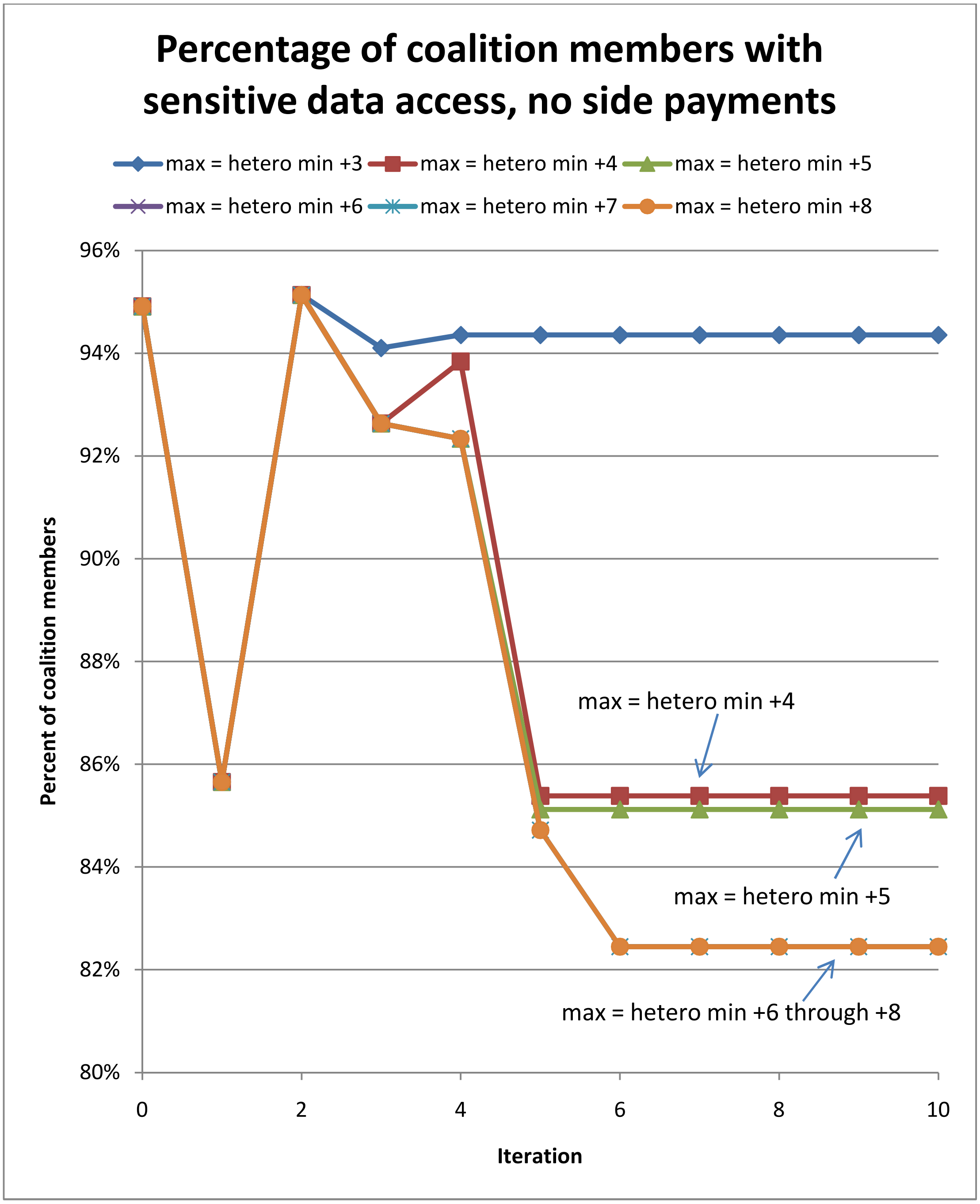


Figure 7.18: Comparison of coalition nodes granted sensitive data access, full prototype minus side payments (prototype two, *n* = 43)

In Figures 7.15 and 7.17, the average number of nodes forming a coalition for prototype two was the same as its average local neighborhood count. However, this was not the case for prototype three. In Figure 7.16, we see utility improvement, which shows decreasing improvement as the game progresses. In general, this pattern was observed for all prototypes. In Figures 7.14, 7.17, and 7.18, we see that there is an inverse relationship between network security and number of nodes granted sensitive data access, as well as between network security and the number of nodes forming a coalition. We hypothesize that as the difference in the nodes' maximum security increases, the minimum security required to join a coalition increases and consequently includes fewer nodes. Beyond these observations, there was no network-for-network correlation between number of neighbors and security.

In the above Figure 7.18, showing the percentage of nodes granted access to sensitive data, we see a great deal of divergence among the different networks with varying degrees of maximum heterogeneous security at approximately five to six iterations. If one examines the above graphs in Figures 7.14 through 7.18 for the same simulation, we see that approximately five iterations is the point at which stabilization of the network takes place for security, utility, coalition, or nodes forming a local area network. This finalization of decision-making by each node is most likely the cause of the divergence among the networks with varying degrees of maximum security as shown in Figure 7.15, which simply represents the near-finalization of partial preferences by each node in the network. This stabilization is thus most likely related to the utility equation itself, and not necessarily restricted to coalition formation with partial preferences, because in addition to the data presented in Figures 7.14 through 7.18, the previous figures for the prototype as represented in Figures 7.1 through 7.13 shows stabilization at approximately the same number of iterations.

7.3 Prototype Three

The objective of prototype three is to study the effect on overall network security and convergence by adding side payments to simulations, yielding a full implementation of our game theoretic security Algorithm 5.4: *Pure_strategy_game*. We examine convergence time, utility improvement, and maximum overall network security and compare our results with those of prototype two. Like prototype two, prototype three includes coalitions, and all nodes have knowledge of events in the game and network. This knowledge includes maximum bidding price. Network size was the same as for earlier prototypes, consisting of 43 nodes. Side payments, which established the difference between prototype three and prototype two, are denominated in terms of security; this is described earlier in Algorithm 5.3: *Reduce_ vulnerability*. The decision to make a side payment is based on attack tree analysis, whereby nodes prefer to solve possible vulnerabilities when they are close to nodes which currently have diminished vulnerability rather than allow a possible vulnerability to become an immediate problem to a node with sensitive data. We chose the cost of payments in our prototype simulation as equivalent to twice the immediate security benefit; while this is cynical it is more representative of the principal of delayed gratification which is at the core of the reasoning behind attack tree decision-making which leads nodes to address problems sooner rather than later, receiving less immediate benefit. Again, while addressing a problem late after its manifestation will likely yield more immediate gross benefit, it represents greater vulnerability and cost to the node itself. Besides being poor attack tree analysis, addressing a problem late also gives lower net benefit after costs are considered. The results from tests of prototype three are found in Figures 7.19 through 7.36 below.

The first chart of the results for prototype three, shown in Figure 7.19, compares average security among the networks with the same variations in heterogeneous minima and maxima as found in the tests of prototype two. Compared to maximum security for prototype two in Figure 7.14, we see results that show similar if not improved maximum security for the networks that could reach the highest security levels. Like prototype two, networks that had a maximum equal to heterogeneous minimum plus five and higher achieved a consistent maximum security. This similarity of performance, with marked improvement starting at plus five above minimum security, again demonstrates that our game theoretic security algorithm can perform equally well under less than optimum conditions.

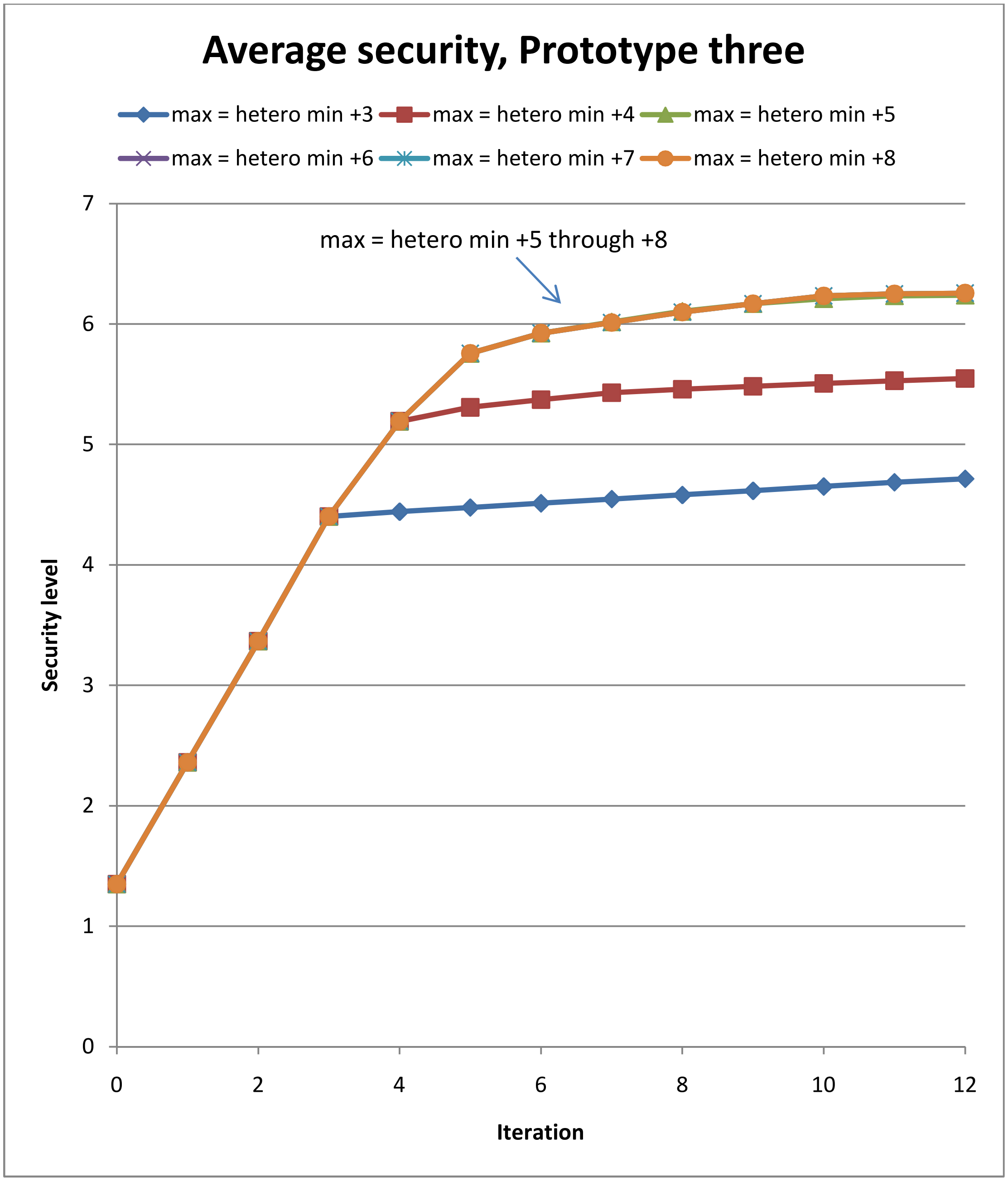


Figure 7.19: Comparison of average overall network security for networks with varying security and heterogeneity, full implementation of our algorithm (prototype three, *n* = 43)

Security for prototype three, represented in Figure 7.19, shows similar stabilization as the corresponding security graph for prototype two in Figure 7.14. Both stabilized after approximately five iterations. Prototype three had higher security, however. Since the only difference between prototypes two and three was the introduction of side payments, these test results show the effectiveness of side payments in improving security.

In general, utility for prototypes two and three showed similar convergence, as shown in Figures 7.16 and 7.21. However, we observed under repeated tests that utility for prototype three worsened at iterations 5 and 6 for two of the networks. These networks had the maximum security equal to the heterogeneous minimum plus five and plus eight. We do not know why this occurred for these specific networks.

Even though we see generally similar security and utility stabilization with (for prototype three) or without (for prototype two) side payments, this is not the case for the number of interactions taking place between nodes. In Figures 7.20 and 7.22 through 7.24 below, we see that prototype three takes nearly double the time to converge versus prototype two; see the corresponding graphs for prototype two in Figures 7.15 and 7.17. Since the only difference between prototypes two and three is the addition of side payments, we must conclude that side payments lengthen the stabilization time with respect to forming a local network and coalition. The increased time is likely a result of the added overhead to make and accept side payments.

For Figures 7.19 and 7.21 we see a pattern that after the same number of iterations, some of the networks diverged from other networks. This divergence was not based entirely on how similar the networks were to one another. In fact, security maximization was nearly identical for the networks with maximum security equal to the heterogeneous minimum plus five and plus eight. These are the same two networks that had the unusual decrease in utility at approximately iterations 5 and 6, which coincides with the point at which these networks began to outperform the others. For all networks, stabilization and convergence takes place after the same number of iterations, regardless of the graph, for the security, utility, and side payments made in different simulations of prototype three. Again, this stabilization is exemplar of the network being optimized according to our game theoretic algorithm. Despite the differences in prototypes one through three, the fundamental algorithm is the same, and we see this represented in the similarity of stabilization with respect to utility and security among the three prototypes. These results again show that our fundamental game-theoretic algorithm has the greatest effect on network performance with respect to security and utility, and the fundamental algorithm has greater significance than any other variations on the algorithm itself as in the early prototypes one and two.

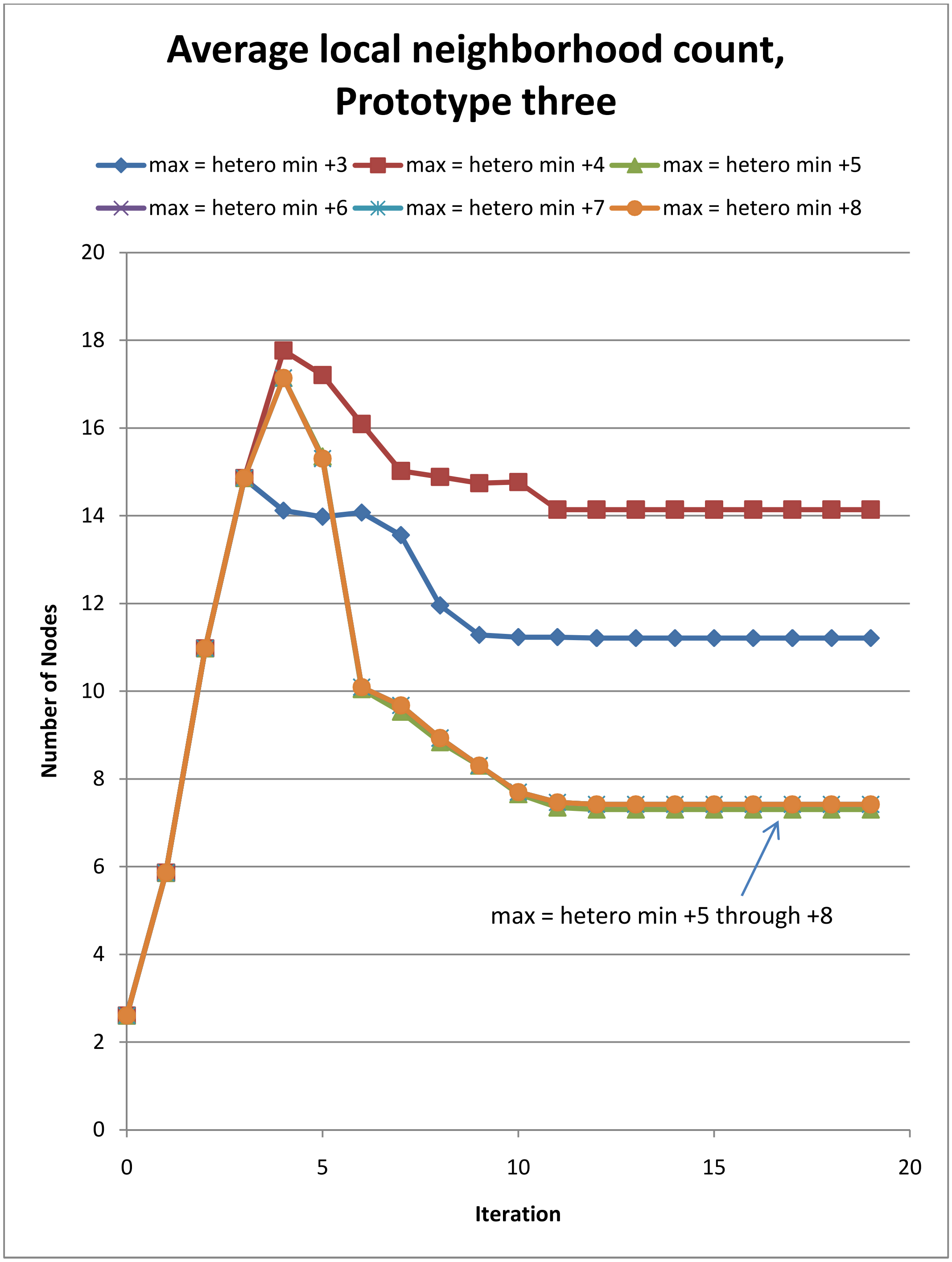


Figure 7.20: Comparison of average number of nodes forming a local neighborhood for networks with varying security and heterogeneity, full implementation of our algorithm (prototype three, *n* = 43)

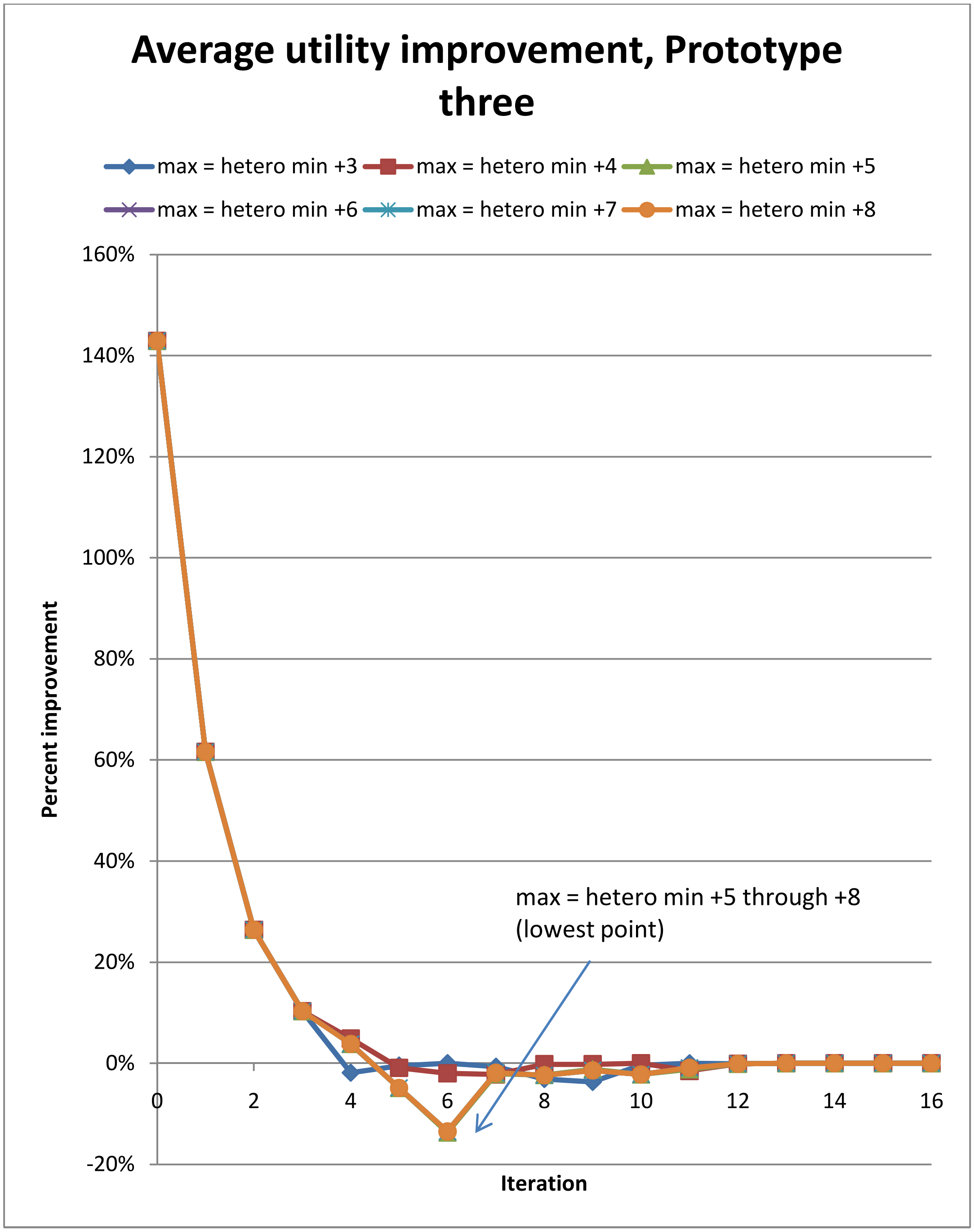


Figure 7.21: Comparison of average utility improvement for same networks as above, full implementation of our algorithm (prototype three, *n* = 43)

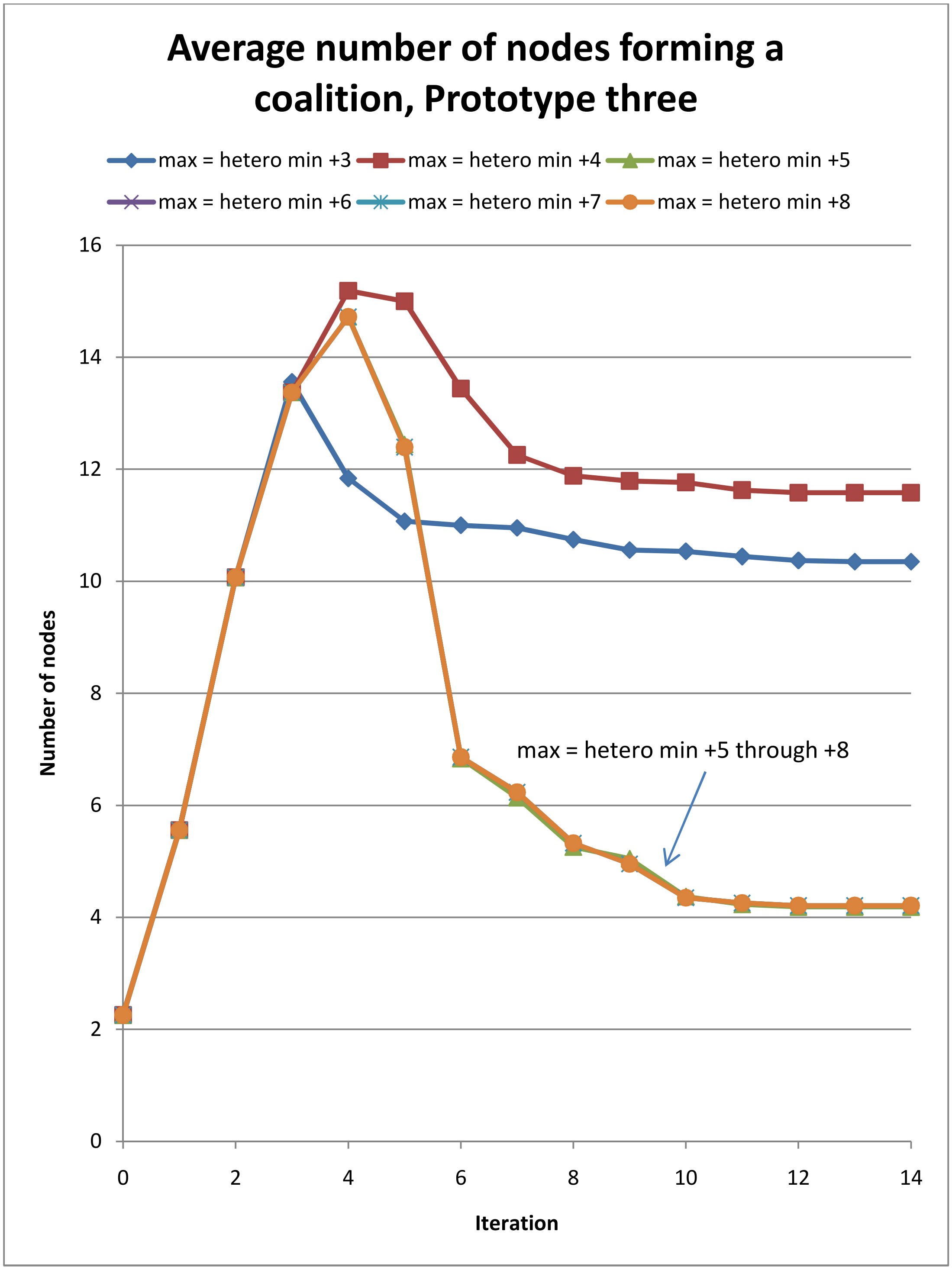


Figure 7.22: Comparison of average number of nodes forming a coalition, varying maximum security and heterogeneity, full implementation of our algorithm (prototype three, *n* = 43)

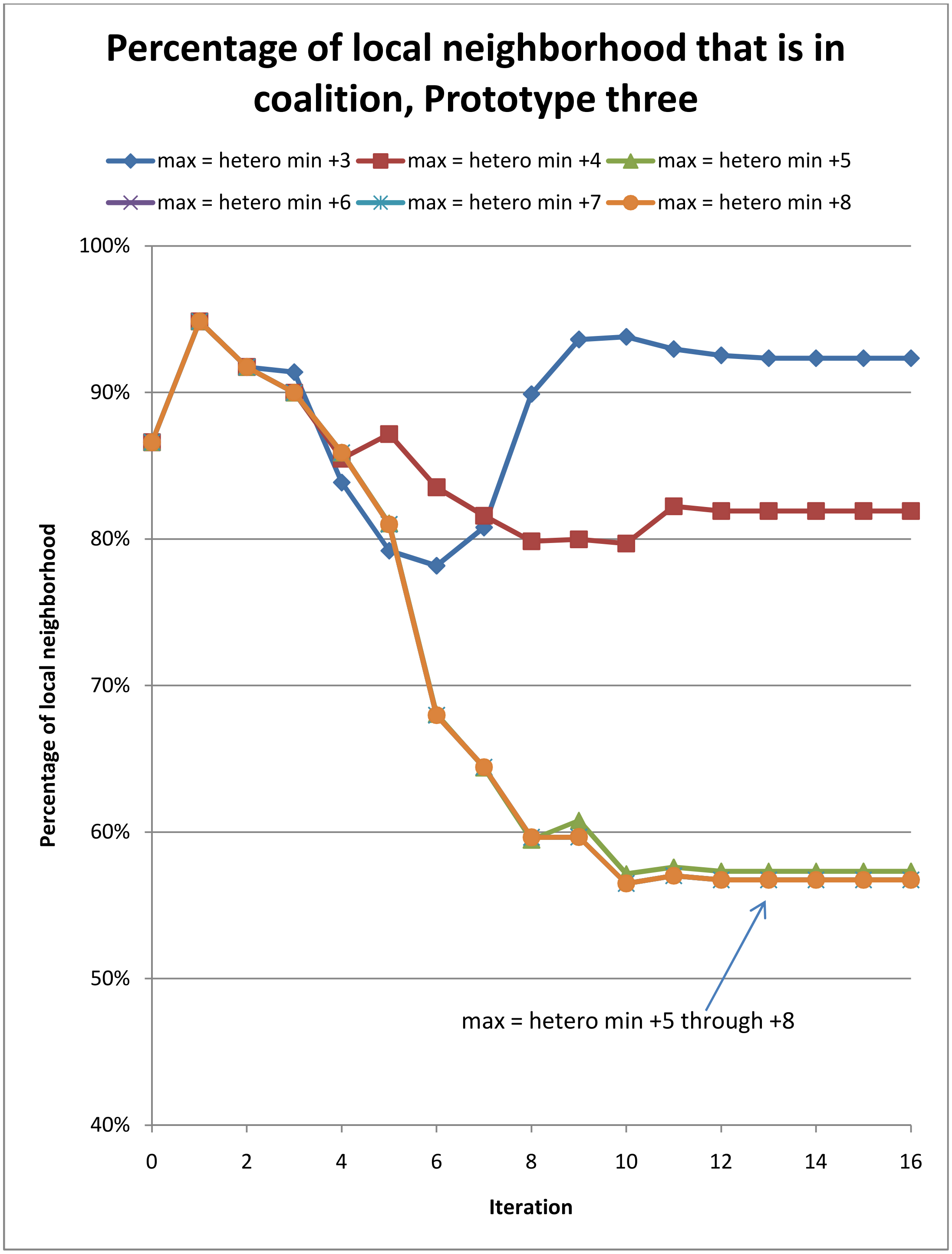


Figure 7.23: Comparison of percentage of local neighborhood (directly connected nodes) forming coalition, varying network heterogeneity, full implementation of our algorithm (prototype three, *n* = 43)

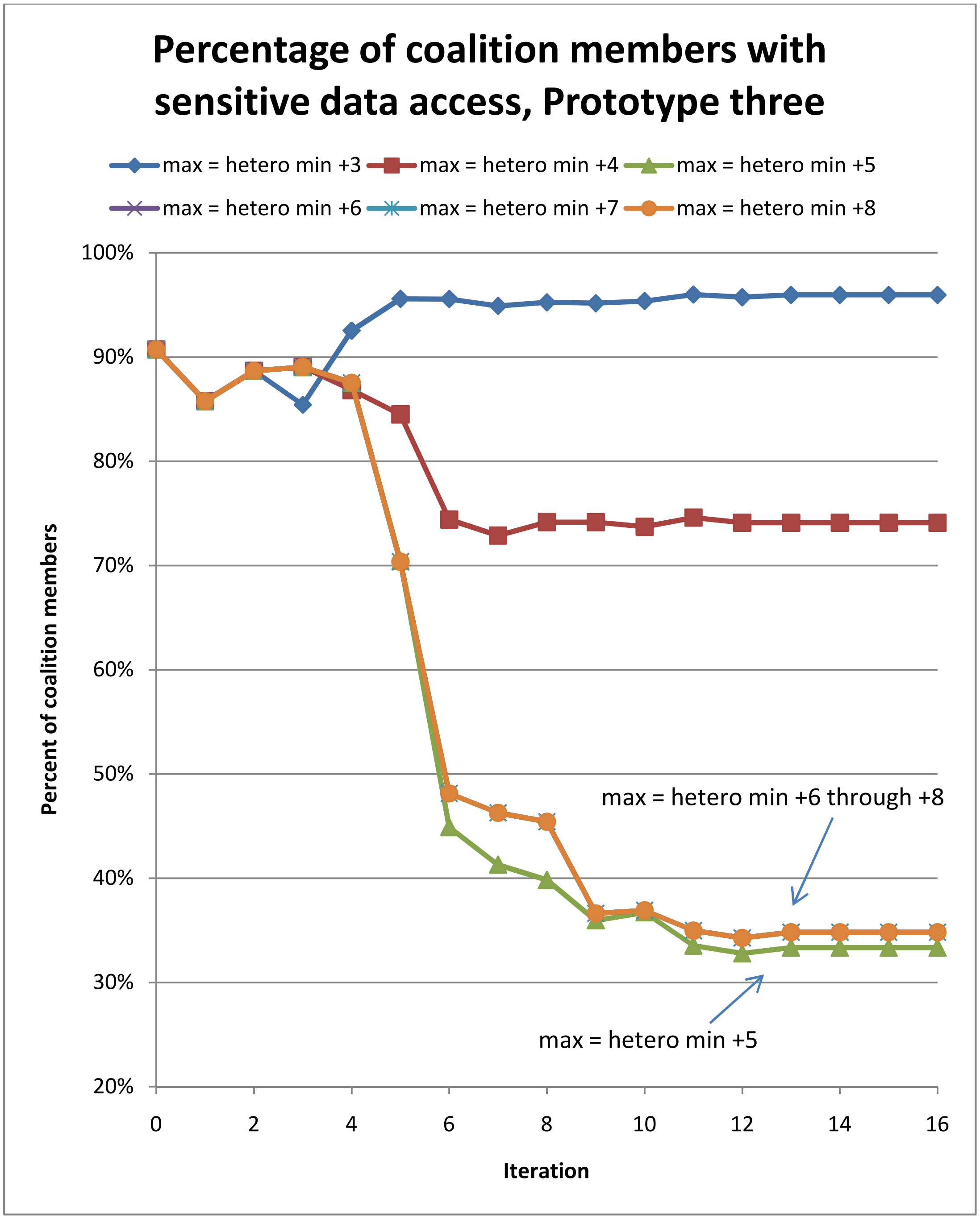


Figure 7.24: Percentage of coalition members granted access to a node’s sensitive data, $d(i) = 1$, for networks with different maximum security and heterogeneity, full implementation of our algorithm (prototype three), $n = 43$

In Figure 7.25, shown below, we see data unique to prototype three, that of the percentage of nodes making side payments. After twelve iterations, the network variations converge to stabilization and no more side payments are made. Since for the data shown in Figures 7.20 and 7.22 through 7.24 shows convergence after ten iterations and we hypothesized that this longer convergence time versus prototype two was directly related to the addition of side payments, we can see that there is some correlation between the percentage of nodes making side payments and stabilization of the related interactions between nodes in a graph.

In addition, we believe that there may be a point at which the percentage of nodes making side payments has no further effect on convergence time, as the side payment stabilization took two iterations longer than the security and utility stabilization. At iterations 11 and 12, the percentage of nodes making side payments dropped below 20% of the total number of nodes; this may be the point at which the percentage of nodes making side payments has no further effect on convergence time.

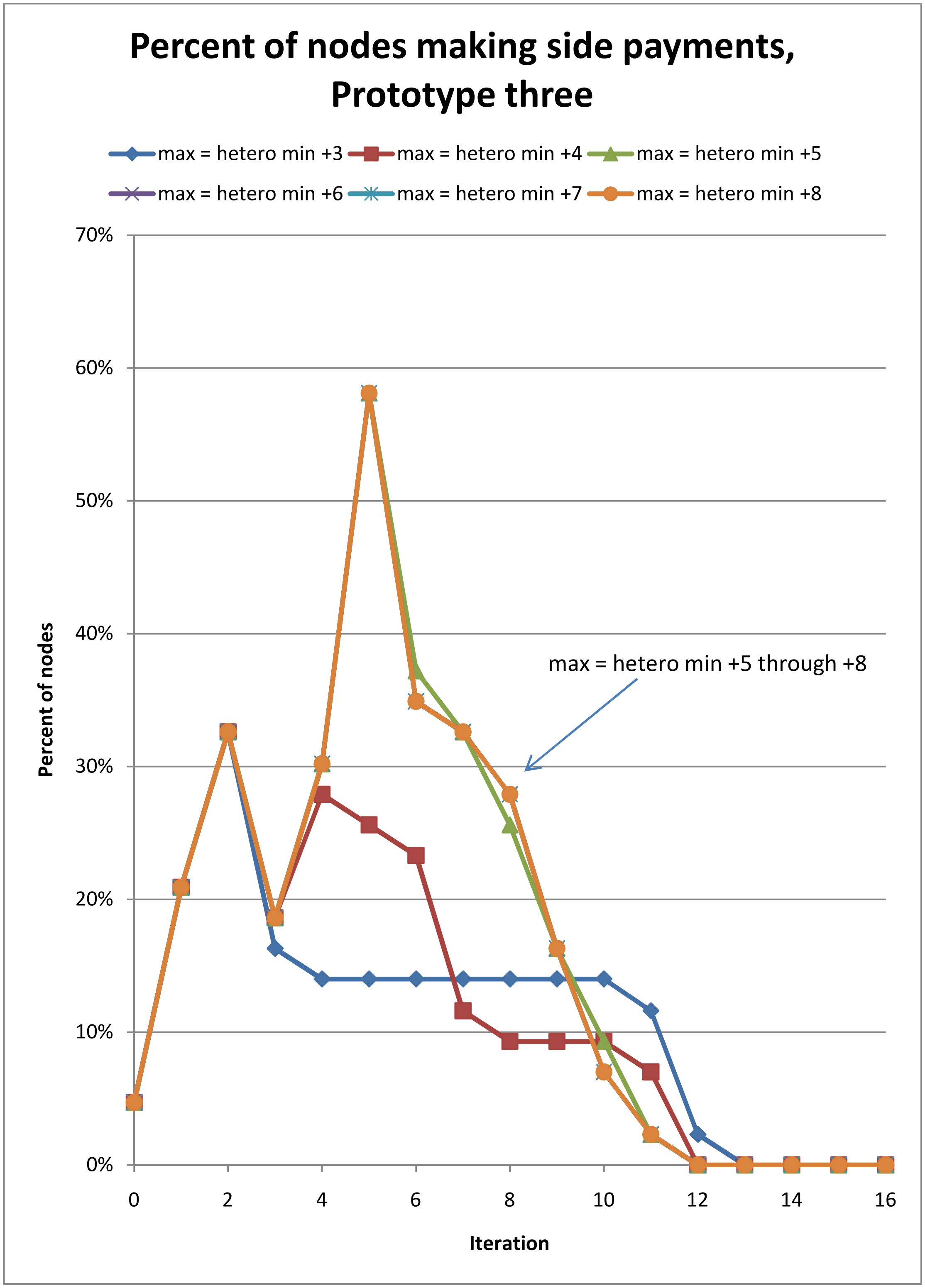


Figure 7.25: Comparison of percentage of network nodes making side payments, full implementation of our algorithm (prototype three), *n* = 43

It is interesting to note, in Figure 7.25 above, the correlation between side payments and the best-performing networks, in terms of security, as shown earlier in Figure 7.19: networks that maximized the use of side payments had the highest security. In fact, the networks with heterogeneous maximum of five or more above the minimum security had nearly identical graphs for side payments.   This shows that side payments indeed do improve security beyond what would be achievable in their absence, thus benefitting network security as a whole.

There is also an interesting correspondence between local network size (and coalition size) and side payments in prototype three: those networks making more side payments tend to have smaller local area networks. Nodes are thus granting fewer other nodes access to sensitive or insensitive data. Because of the decrease in the number of nodes forming a local area network in prototype two versus prototype three by a factor of two or three (see Figures 7.15 and 7.20, respectively), in addition to the similar decrease in the percentage of nodes allowed sensitive data access in the best-performing heterogeneous networks (heterogeneous maximum of plus five through eight, see Figures 7.18 and 7.24), we hypothesize that the nodes in the network, through the feedback mechanism of side payments to reduce vulnerability to other nodes, are in effect learning to reduce their own vulnerability by communication via the side payment mechanism. This hypothesis merits future study. As above in prototype two, there was no network-for-network correlation between number of neighbors and security in prototype three. However, there was a pattern for the networks having highest security, those of max = hetero min +5 through +8. With the exception of the two lowest-security networks,

networks with the highest security had fewer connections. The max = hetero min +5 through +8 networks had a smaller average local neighborhood count, smallest average number of nodes forming a coalition, smallest percentage of local neighbors in coalition, and smallest percentage of coalition members with sensitive data access.

7.3.1 Prototype Three Version Two

The purpose of prototype three, version two is to study the impact of more effective side payments on the network as a whole. It is otherwise identical to the earlier version of prototype three, which we shall henceforth denote as prototype three version one. Effectiveness is measured in terms of a two-fold increase in security versus side payments for prototype three version one. We chose to alter this side payment effectiveness quantization to determine whether our logic for choosing to denominate earlier side payments more closely to the principal of delayed gratification was correct, and also to see whether allowing nodes the capability of making changes to security that are more effective have an impact on overall network security. The data shown in Figure 7.26 gives the security of the network for this change in side payments.

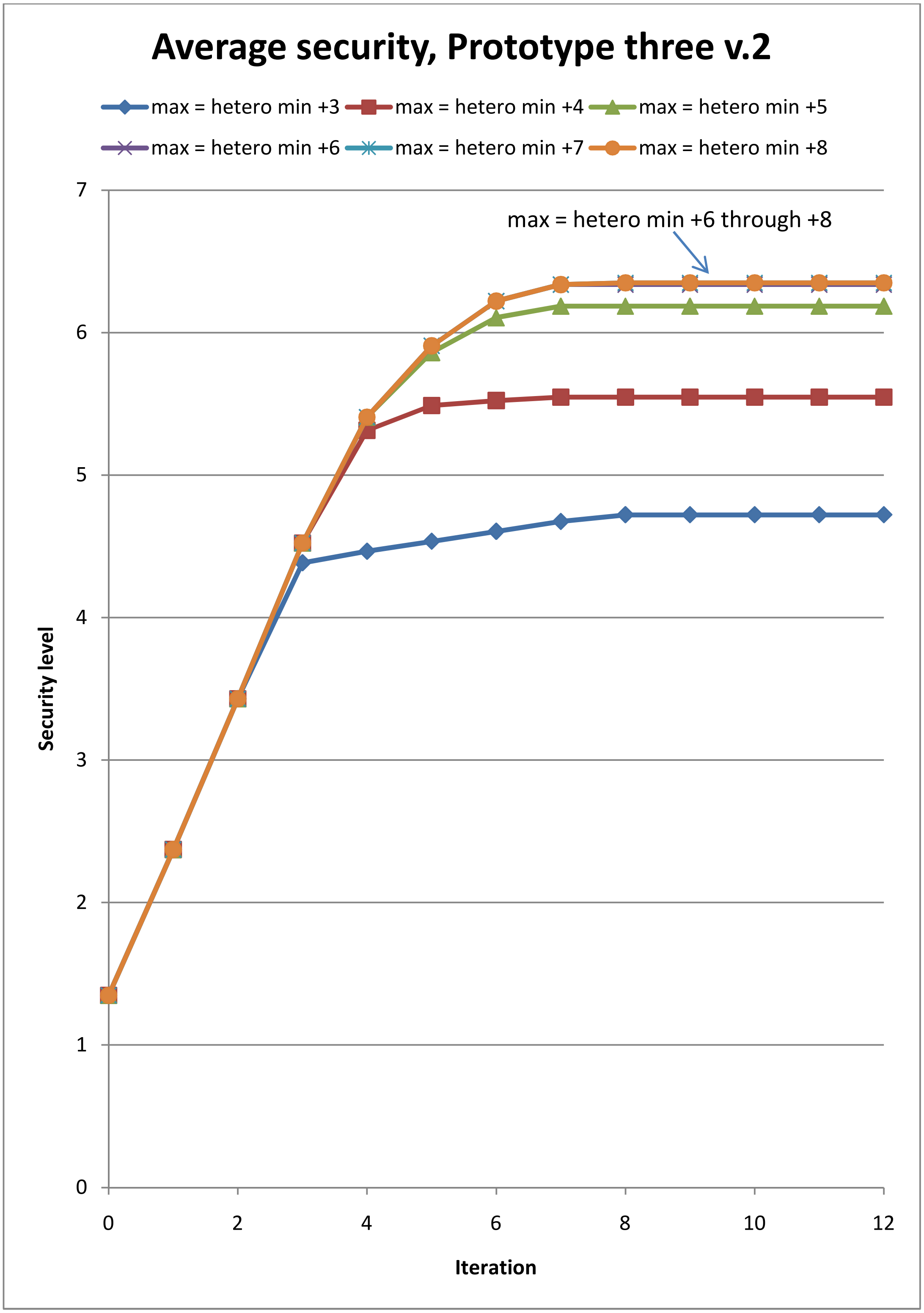


Figure 7.26: Average overall network security, more effective side payments, full implementation of our algorithm (prototype three version two), *n* = 43

Henceforth we label prototype three as prototype three version one, or, in short, prototype three v.1. If we compare the data in Figure 7.26 for prototype three version two to the corresponding data in Figure 7.19 for prototype three version one, we see little difference in improvement in overall network security for the best performing heterogeneous networks. However, we do see a marked decrease in convergence time for networks in prototype three version two, the network with more effective side payments, by a factor comparable to the increase in side payment effectiveness. Figure 7.27 below compares the performance of the fastest-converging network (heterogeneous maximum equal to the minimum plus five) in prototype three version one and prototype three version two. The data shown in Figure 7.27 illustrates the decrease in convergence time by almost half for prototype three version two, seven versus 11 iterations. We see nearly identical results in Figure 7.28 for the second-fastest converging network, with a maximum security equal to the heterogeneous minimum plus six. We also see similar results for the remainder of the heterogeneous maxima as shown in Figure 7.26 compared to the corresponding results for prototype three version one in Figure 7.19.

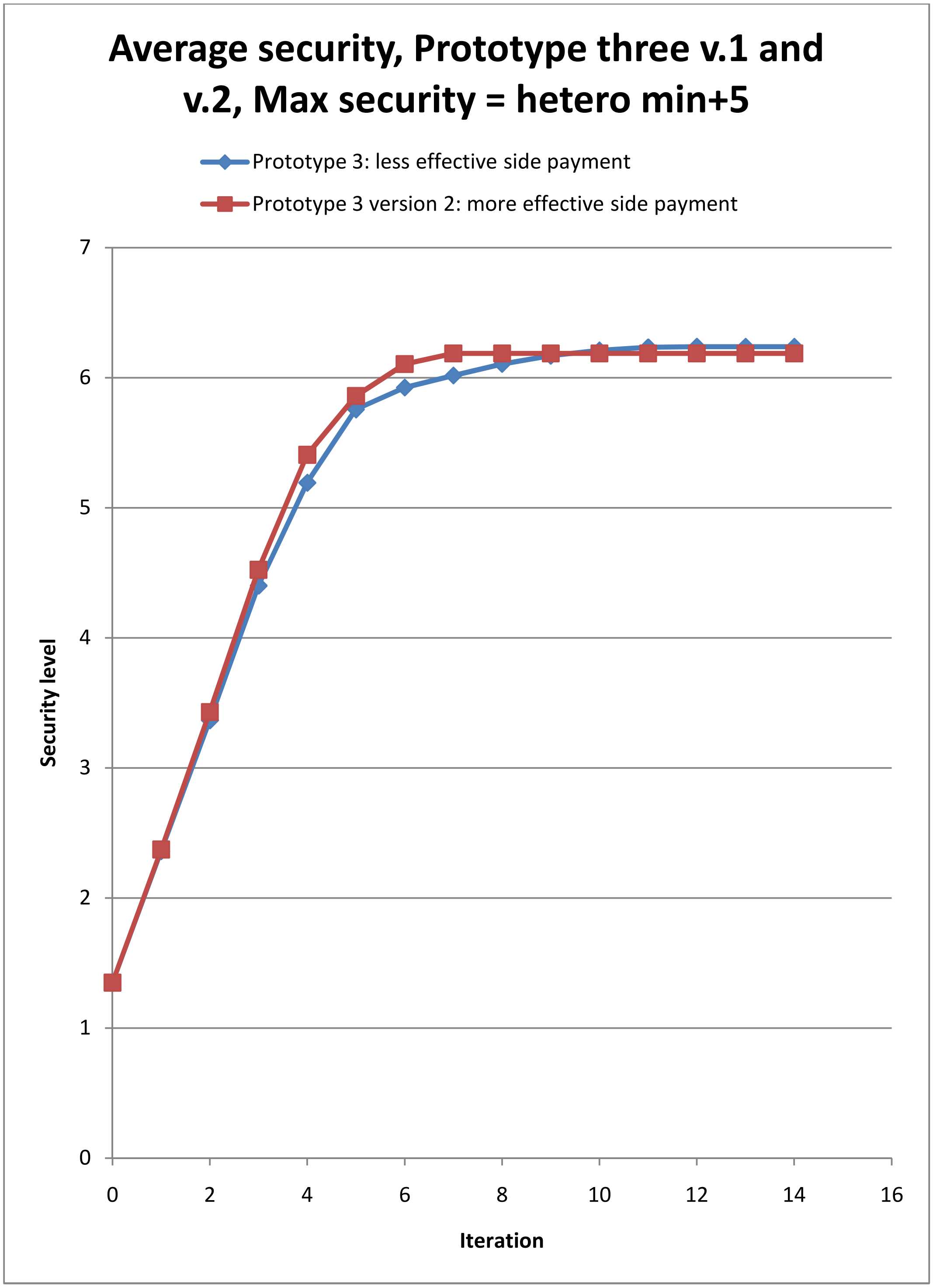


Figure 7.27: Average security comparison for prototype three versions one and two, (full implementations of our algorithm), network maximum security equal to minimum plus five, *n* = 43

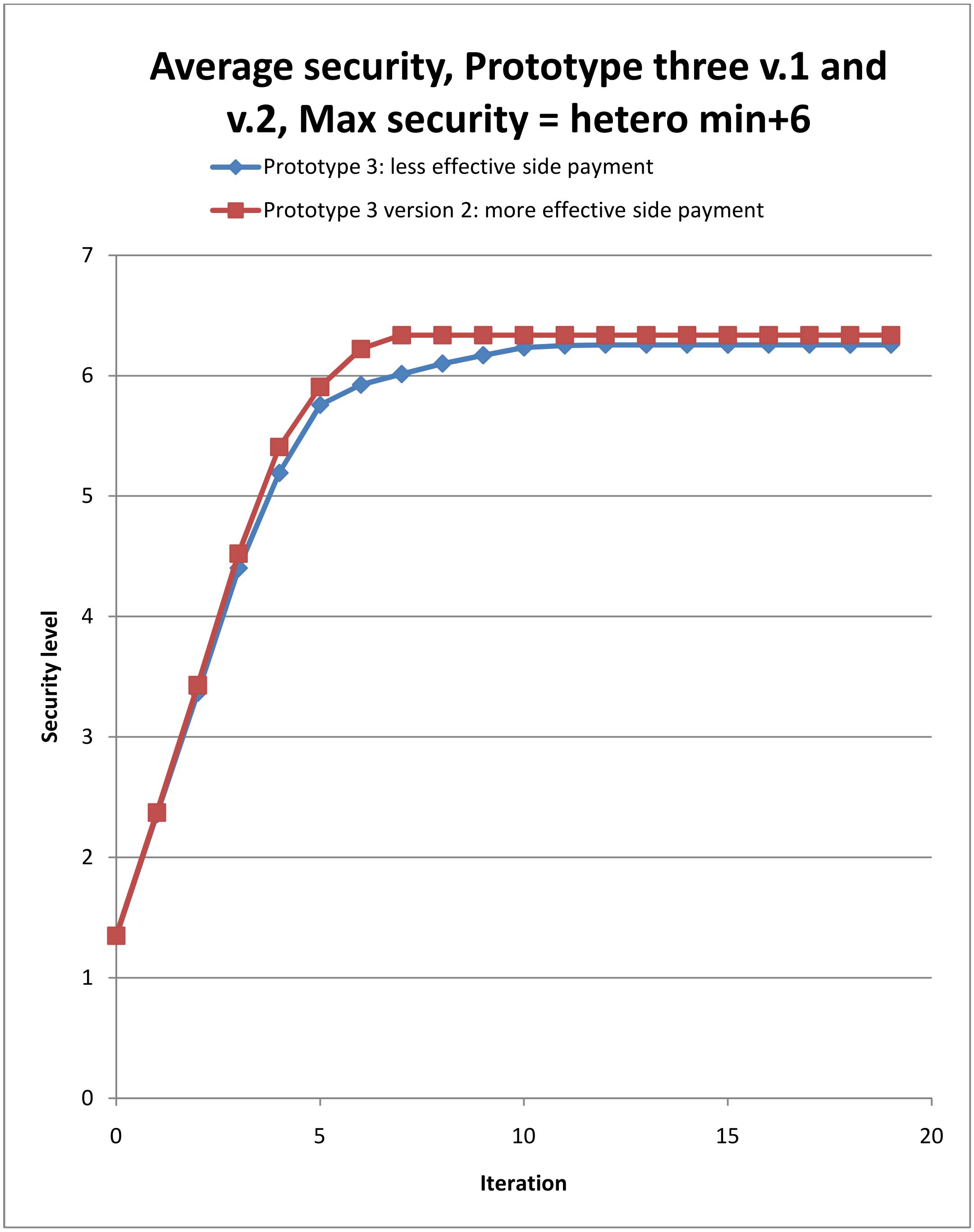


Figure 7.28: Average security comparison for prototypes three versions one and two, (full implementations of our algorithm), network with maximum security equal to minimum plus six, *n* = 43

Note in Figures 7.27 and 7.28 that security maximization is similar between the two prototypes. From these results we conclude that increased effectiveness of side payments decreases convergence time of the network by a comparable amount, almost half.

Figure 7.29, shown below, shows the percentage of nodes making side payments for prototype three, version two. If we compare this data with that of the corresponding data for prototype three version one as shown in Figure 7.25, we see a decrease in the number of side payments for prototype three version two. This decrease in side payments corresponds with the decreased convergence time of this variation on prototype three.

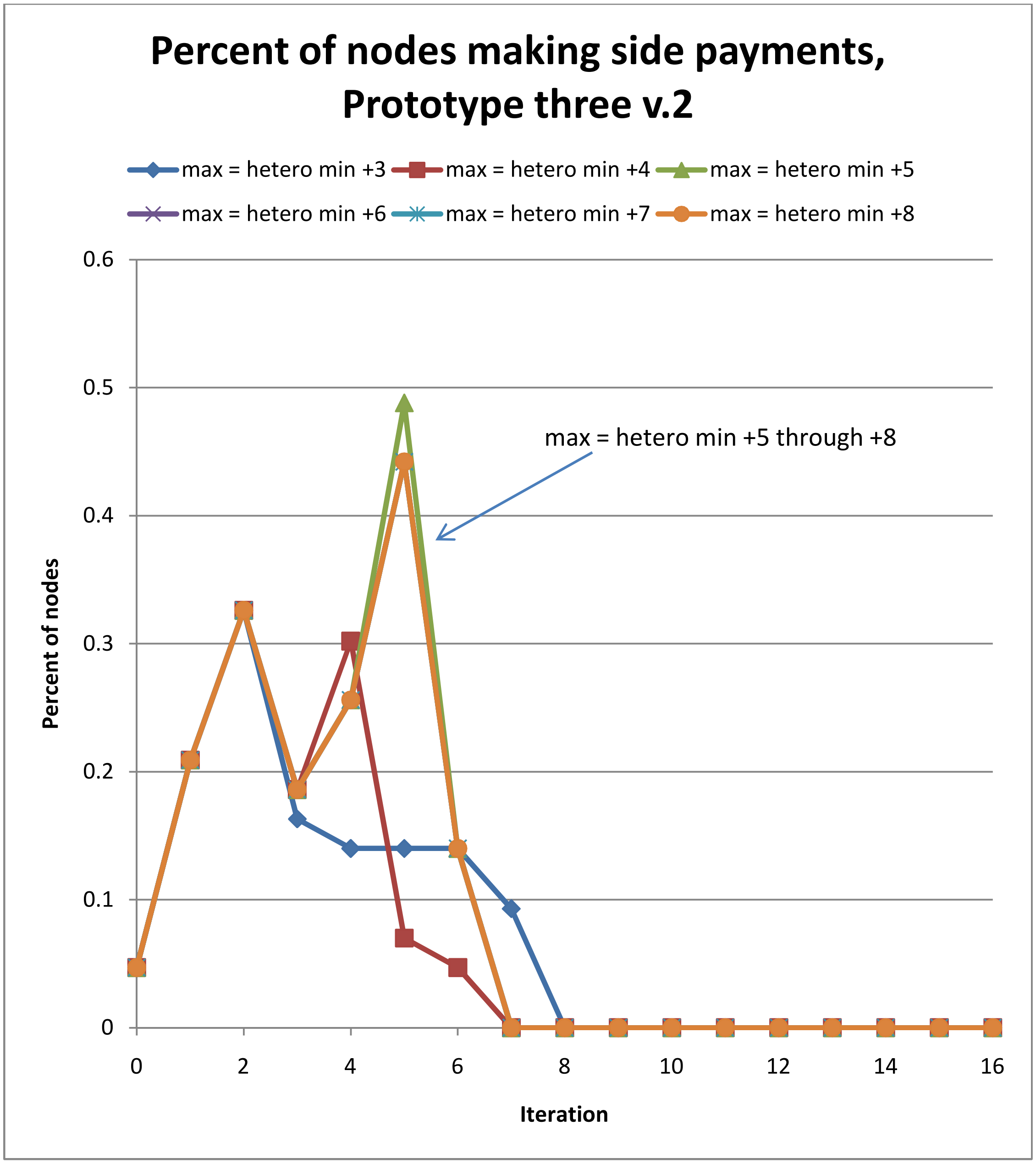


Figure 7.29: Percent of nodes making side payments, (full implementation of our algorithm), prototype three version two, *n* = 43

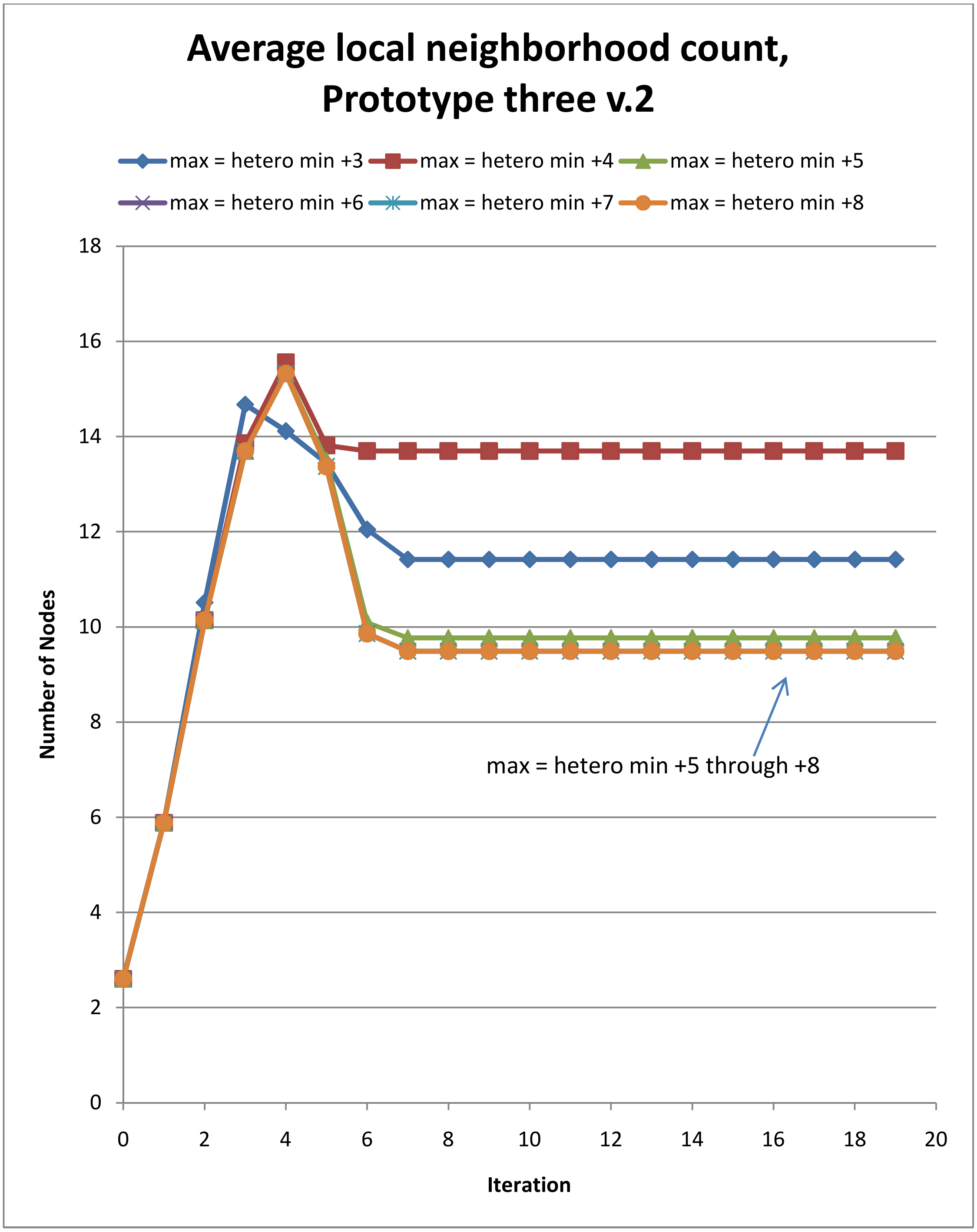


Figure 7.30: Average local neighborhood count, (full implementation of our algorithm), prototype three version two, *n* = 43

Comparing an average of the networks' results shown above in Figure 7.30 for number of nodes forming a local neighborhood with the average of the same data for the less effective side payments of prototype three version one in Figure 7.20, as shown in Figure 7.31 we see there is an increase in the number of nodes forming a local neighborhood when effectiveness of side payments is increased. There does exist, however, an initial increase by prototype three version one over that of version two. We hypothesize that this is caused by the effectiveness of side payments themselves in combination with nodes' initial perceived improvement, which gives way to the observed effect of side payments after a few iterations, whereby nodes learn that the increased side payments' effectiveness are not as predicted. It is also possible, and we believe more likely, that the nodes with less effective side payments are making more of them in order to overcome their deficiency in security improvement; whereas when side payments are more effective in improving security, nodes do not have to make as many payments to achieve the same effect.

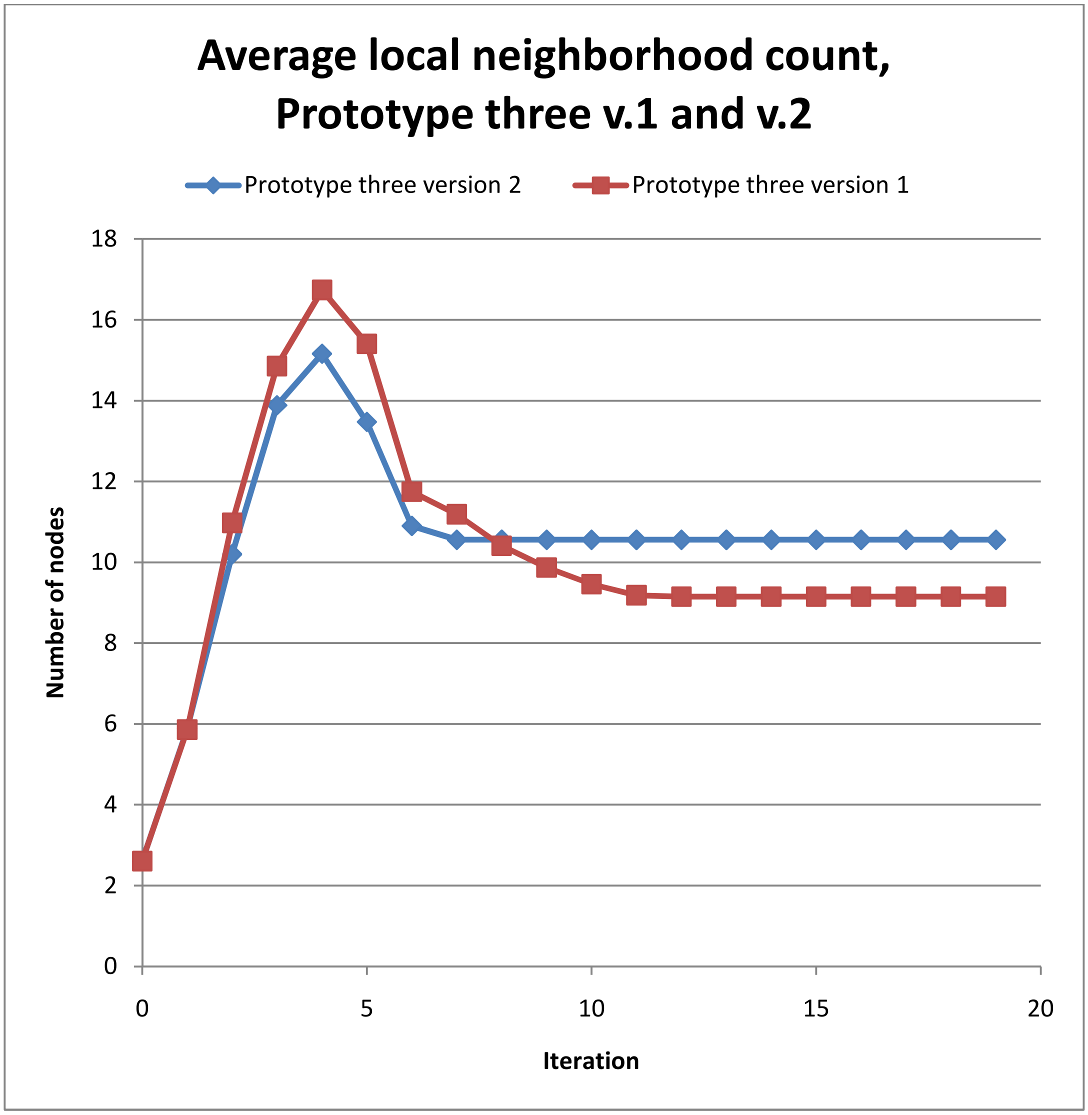


Figure 7.31: Number of nodes forming local neighborhood comparing side payment effectiveness in full implementation of our algorithm, (prototype three), *n* = 43

Compared to average security as shown in Figure 7.32 below, the difference is not as apparent as it is in Figure 7.31; instead the networks have similar results for maximum average security despite the decreased convergence time for networks with more effective side payments in prototype three version two.

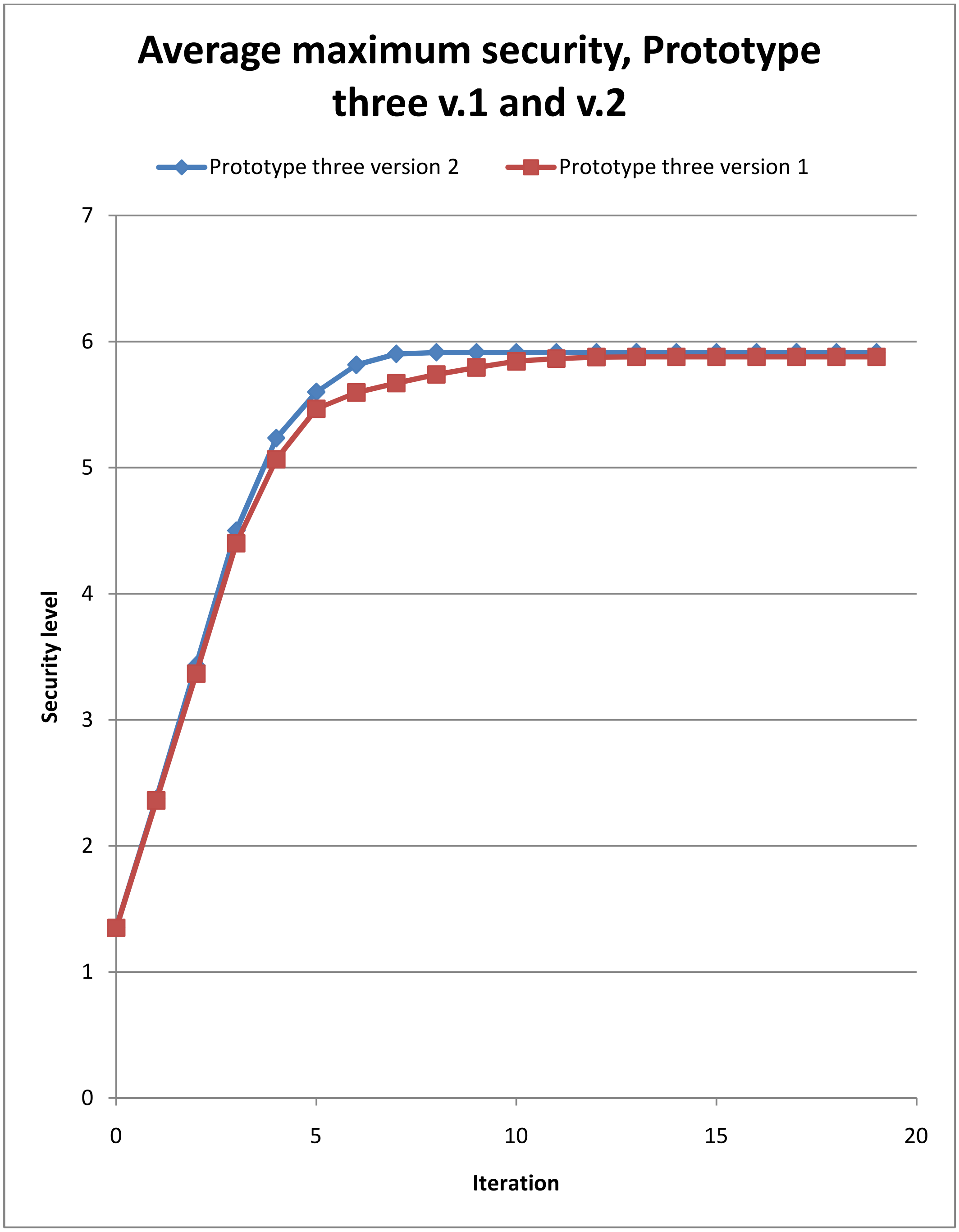


Figure 7.32: Average maximum security comparing side payment effectiveness in full implementation of our algorithm (prototype three), *n* = 43

In Figure 7.33 below we again see nearly identical utility improvement for the averages of prototypes three versions one and two, again illustrating that despite the increased convergence time the more effective side payments have almost no other effect. Note also that at six iterations, the utility improvement drops below zero, indicating a loss of utility, but recovers at the next iteration. Future work should include further study of the possible causes of this anomaly, as it was not predicted by any of our mathematical models.

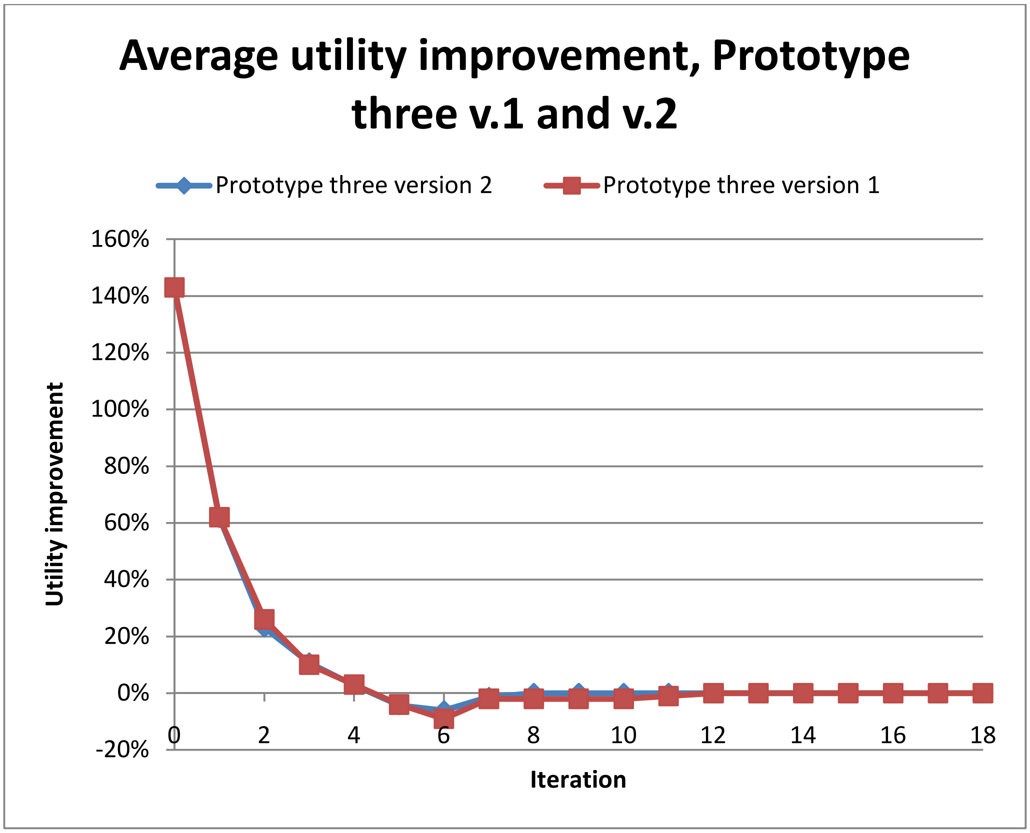


Figure 7.33: Utility improvement, comparing side payment effectiveness in full implementation of our algorithm (prototype three), *n* = 43

In Figure 7.34 below, we examine side payment effectiveness on coalition size. We see nearly identical results as in Figure 7.31 above, local neighborhood count. Again, prototype three version one, implementing less effective side payments, had more nodes in the coalition early on, but fewer nodes in the coalition at the stable maximum. This correlation indicates that increasing effectiveness in side payments has the most impact on improving network connectivity.

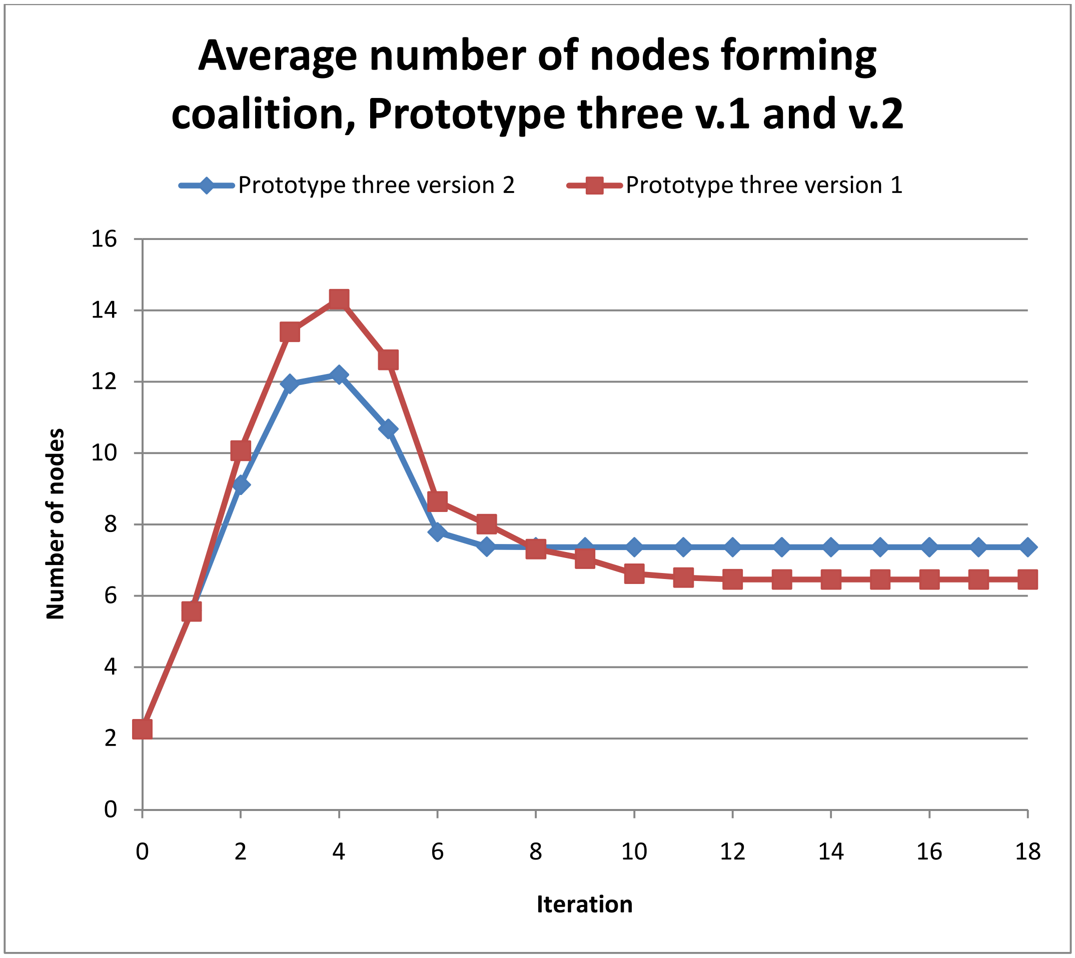


Figure 7.34: Average quantity of nodes forming coalition, comparing side payment effectiveness in full implementation of our algorithm (prototype three), *n* = 43

Figure 7.35 below shows the percentage of coalition members granted access to sensitive data for the continuing averages comparison. Here we observe that networks with more effective side payments have an increase in terms of number of coalition members granted sensitive data access. Based on these results and those described above for Figures 7.31 and 7.34, we conclude that the side payment effectiveness acts as a measure of increasing trust and collaboration between nodes.

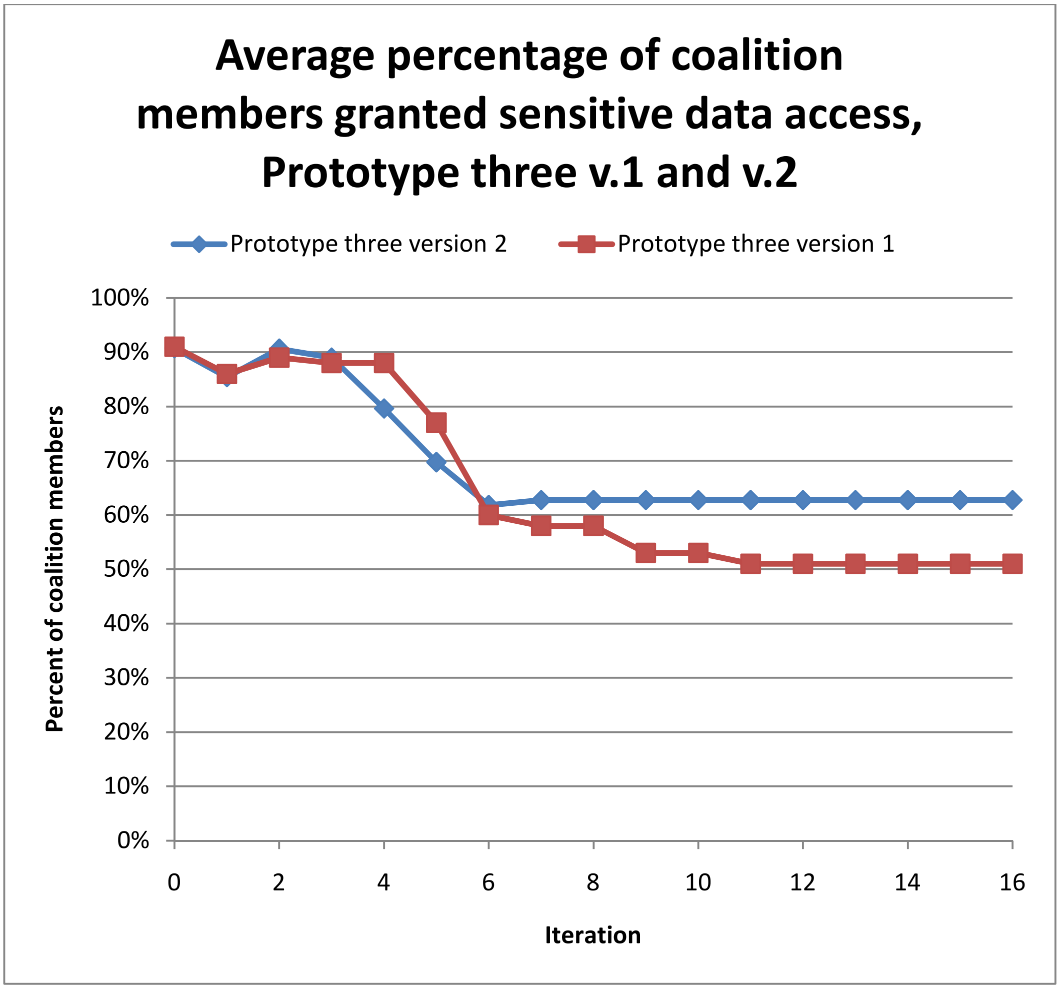


Figure 7.35: Percent of coalition granted sensitive data access, comparing side payment effectiveness in full implementation of our algorithm (prototype three), n = 43

If we examine the best performing networks for prototype two, prototype three version one, and prototype three version two, we may be able to see how well each prototype compares to the other's security improvement. Because prototype one made the assumption that all costs, constraints, and bids were resolved ahead of time and all nodes belonged to the same coalition, and no side payments took place, as such it did not implement several key points in our fully mature algorithm that are needed to effectively compare its results with those of prototypes two and three. Prototype one lacks the fully mature cost function and utility calculation of Algorithms 5.2 through 5.4. Thus prototype one is omitted from the graphs below as in Figure 7.36. As demonstrated in prototype three, version two, we see an improvement in security as shown in Figure 7.36 when side payments are used versus no side payments. We also see a slight increase in security when these side payments are more effective, but again not corresponding to the rate of improvement.   Again, improved side payments decreased the convergence time, giving speedup nearly equivalent to omitting side payments from consideration altogether. The prototype that had the fastest convergence, prototype three version 2, had largest average local neighborhood count, average number of nodes forming a coalition, and average percentage of coalition members granted sensitive data access.

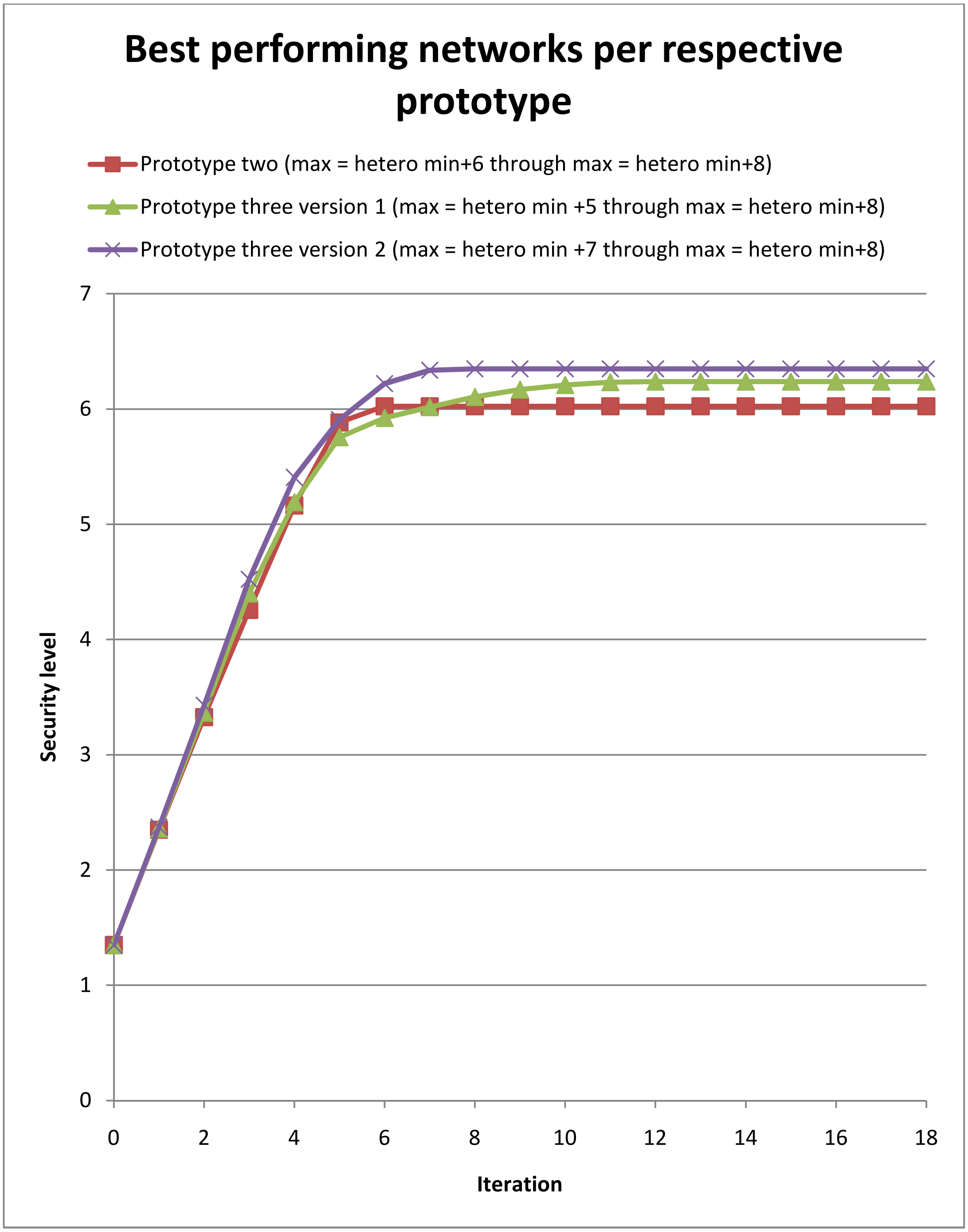


Figure 7.36: Maximum security comparison of best performing networks on prototypes that fully or nearly-fully implement our game theoretic algorithm for pure strategies, *n*=43

7.4 Prototype Four

The objective of prototype four is to implement Algorithm 5.5: *Mixed_strategy_game* in its entirety, and compare its results with prototype three. With the exception of players moving according to a mixed strategy, prototype four is otherwise identical to prototype three. Side payments are again used in the context of the same networks tested in prototype three. Coalitions are formed and nodes follow the same rules as prototype three. As with prototype three, nodes in prototype four are able to observe all events.

Since the players are moving according to a mixed strategy, they by definition move according to a probability distribution over the set of pure strategies. The probability distribution chosen for prototype four was for the players to choose the optimal strategy approximately 97% of the time. We implemented at least five runs of all variations on prototype four and visually inspected the results to ensure that the players did indeed choose the correct strategy, on average, 97% of the time. The actual numerical percentage we observed ranged from approximately 95% to 99% for an average of 97%; hence it is identified as the average percentage of the time that players in prototype four chose the optimal strategy. We chose this probability distribution because it gave the most consistent and repeatable results with random number generation. For more detail on the random number generator, we refer the reader to [41], [42], [47], and [48].

While the increased side payment effectiveness in prototype three version two directly decreased convergence time versus prototype three version one, we did not know whether this would hold true for the mixed strategies of prototype four. We tested the same variation in side payment effectiveness for prototype four version two. Prototype four version one kept all side payments at the same level as version one of prototype three, whereby the cost is equal to twice the security benefit. But in either prototype, if a node receiving a side payment chooses a suboptimal strategy, it will fail to receive or act upon the side payment. Its mistake also influences the node that made the side payment because the benefit to security is lost and never recovered.

Since the players in prototype four are moving according to a mixed strategy, we would expect there to be a decrease in overall security for prototype four versus prototype three. However, this was not the case. Instead, we found that convergence time increased significantly, but so did average security. Our results are shown in Figure 7.37, below, where we compare average security for prototypes two through four, this time averaged over all the networks' security for each prototype. Each chart below that lists average security by prototype means it represents the average of the security of all networks tested by prototype, ranging from maximum security equal to heterogeneous minimum plus three to maximum security equal to the heterogeneous minimum plus eight.

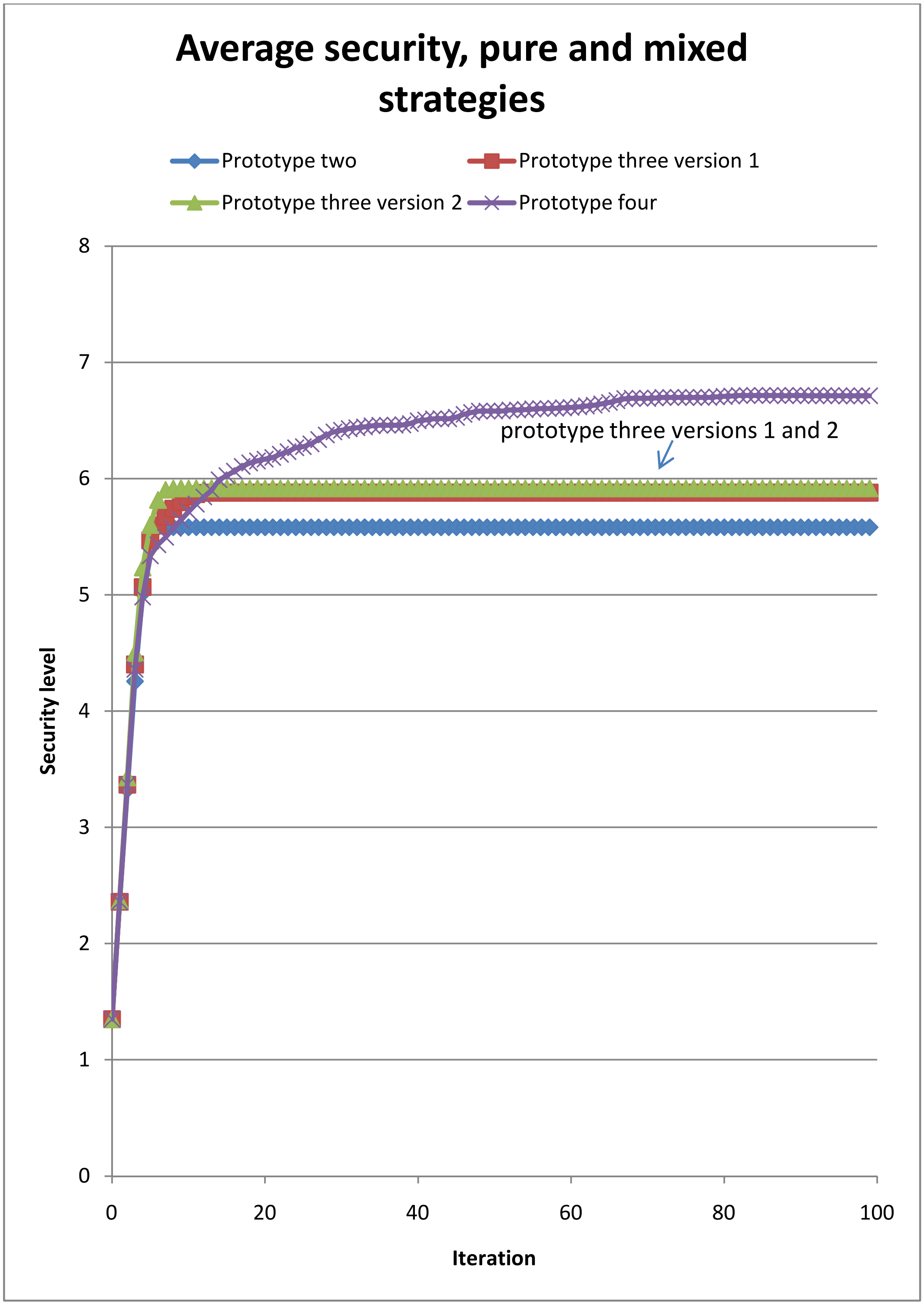


Figure 7.37: Average maximum security comparison, prototypes fully or nearly-fully implementing our game theoretic security algorithm, *n* = 43

As shown above in Figure 7.37, which gives the average security per prototype, we see that the introduction of side payments in prototype three improve average security by approximately 10%. Prototype two, which fully implemented our game theoretic algorithm with the exception of side payments, had an average security of approximately 5.5, while prototype three versions one and two had an average security of approximately 5.9. As mentioned earlier in comparing prototypes two and three, the introduction of side payments also caused an increased convergence time to stable network security. However, its convergence time is not nearly as significant as the convergence time for a prototype of a game using mixed strategies. While the average security for prototype four, using mixed strategies, took the longest time to stabilize at approximately 90 iterations, its security was greater than any other prototype.

In Figure 7.38 we see that each network's maximized security in prototype four is near or at its theoretical average maximum. Recall that networks made up of nodes with lower maximum theoretical security (e.g., maximum security equal to the heterogeneous minimum plus 3) had smaller maximum achievable average security. For example, since the average minimum security for each network was approximately 1.3, the average theoretical maximum for the network with maximum security equal to the heterogeneous minimum plus 3 would be equal to 1.3 (the heterogeneous minimum) plus 3, giving a theoretical maximum of 4.3. The theoretical maximum is reached for this network as shown in Figure 7.38. With the exception of the network that had a maximum security equal to the heterogeneous minimum plus seven, the networks shown in Figure 7.38 reached their theoretical maximum.

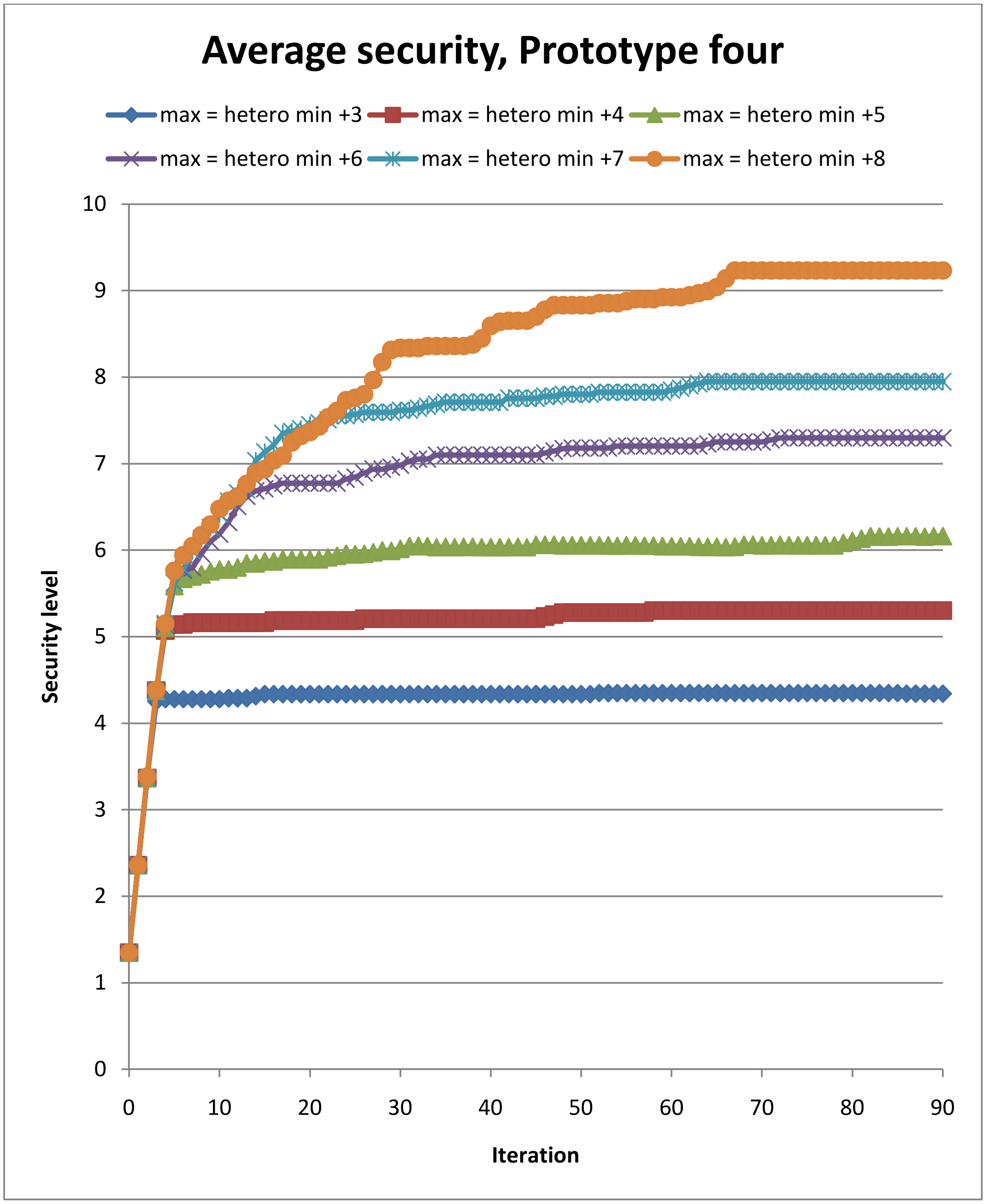


Figure 7.38: Average network security, full implementation of our algorithm for mixed strategies, prototype four, varying heterogeneous networks, *n* = 43

We hypothesize that the improvement in security for prototype four is either related to the extended iterations and thus interactions between nodes, or is related to players attempting to compensate for mistakes via additional side payments. It is also possible that the side payments may be acting as a sort of feedback mechanism for learning.

Recall that through side payments, utility is decreased while security is increased for a node paying a second node to make a security change; but if the player who received the payment (payee) makes an error according to the mixed strategy, the action to improve the payer's security is not made, causing the security improvement and utility to be lost and the payer to recalculate its security to correct for the other node's mistake. However, we observed no correlation with the decrease in utility and increase in side payments as shown in Figure 7.39.

When we compare security for prototype four shown in Figure 7.38 with the security for prototype three in Figure 7.26, we see some similarity in the network security distributions. In both prototypes three and four, the network with maximum security equal to the heterogeneous minimum plus eight is highest, the network with maximum security equal to the heterogeneous minimum plus four is in the midrange, and the network with maximum security equal to the heterogeneous minimum plus three is lowest, which is generally consistent with what we would expect. However, we did not expect the marked improvement in security made by prototype four. Since each network's security approaches its theoretical maximum in prototype four, we do not see the grouping of similar maximum securities for some of the networks as occurred in prototype three, shown in Figure 7.26.

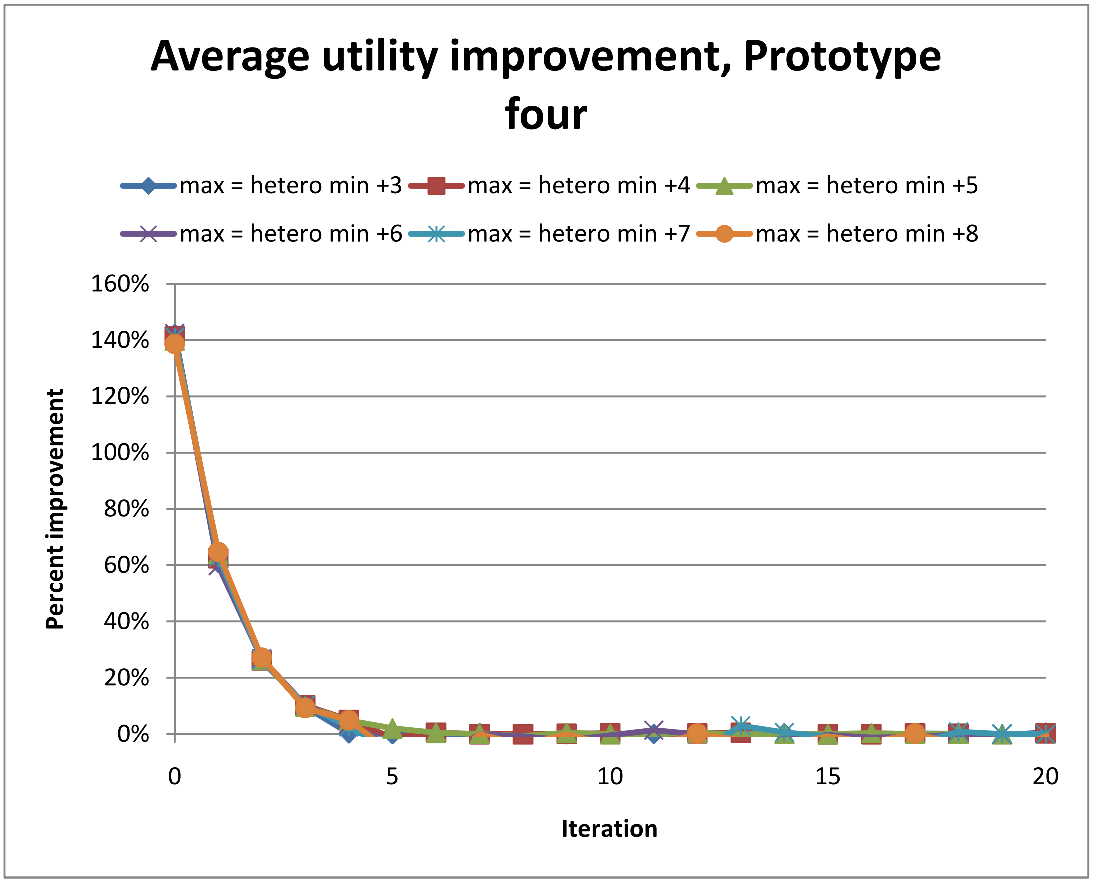


Figure 7.39: Mixed strategy utility improvement, full implementation of our algorithm for mixed strategies, (prototype four), *n* = 43

Figure 7.40 shown below shows the percentage of nodes making side payments. We examined the percentage of nodes making side payments in order to determine whether there was any correlation between it and security improvement. Note that side payments continue past security stabilization, and once past approximately 15 iterations do not follow the pattern for pure strategies as shown in Figure 7.25 above. Furthermore, the percentage of nodes making side payments for the network where the maximum was equivalent to four levels higher than a node’s minimum security had an unusual spike at approximately 45 iterations, even after repeated tests. We hypothesize that this occurred as a result of the network’s attempt to increase its own security through side payments in order to overcome its own security limitations. It is interesting to note that this phenomenon occurred exclusively to this network at approximately 45 iterations, which is approximately halfway through the optimization process (stabilization occurs at approximately 90 iterations). In this network, many nodes have a maximum security of 5 out of 10. Whether there is a correlation between convergence, maximum security, and side payment interaction is to be determined in future work.

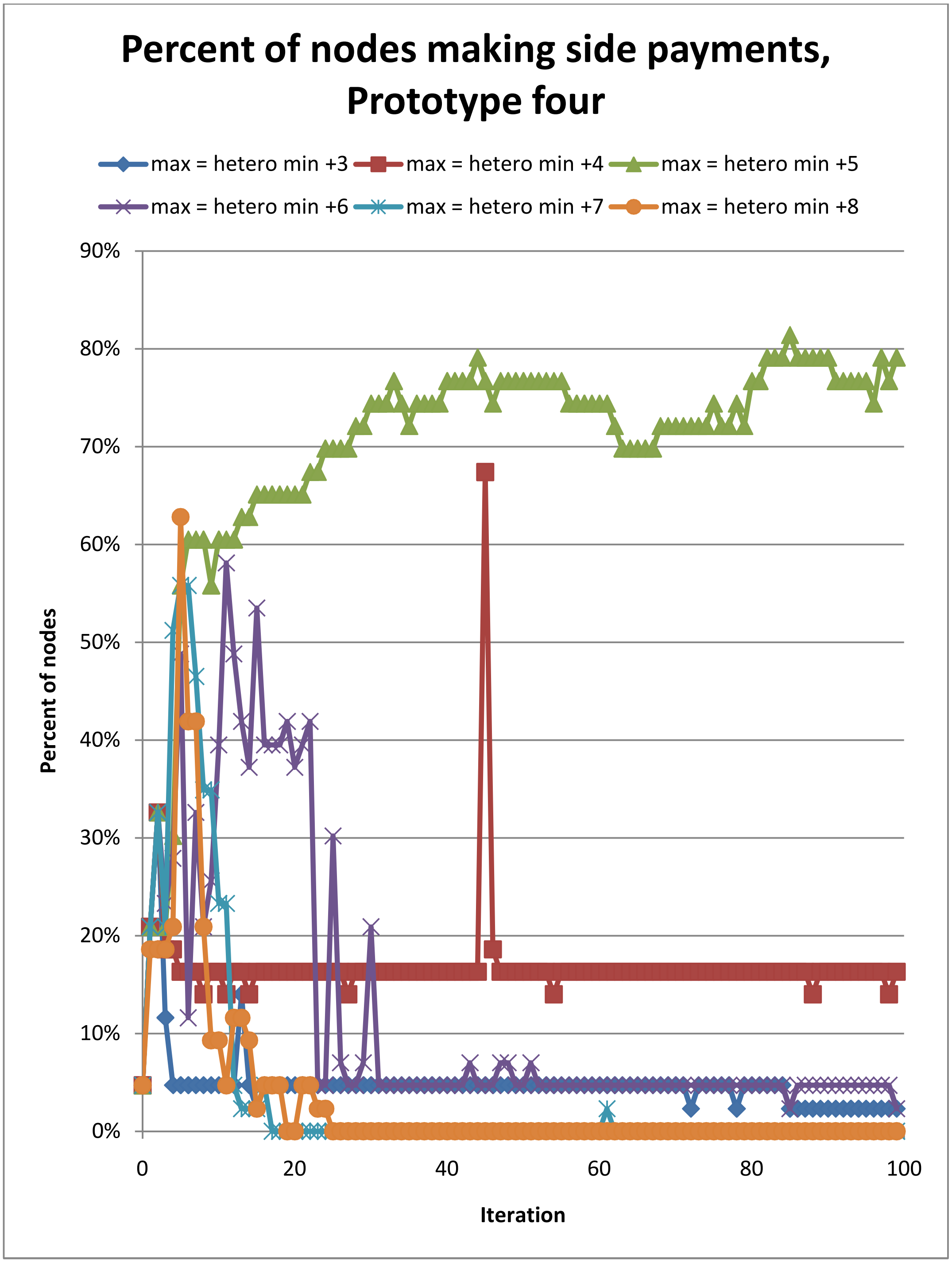


Figure 7.40: Percent of nodes making side payments, full implementation of our algorithm for mixed strategies, prototype four, n = 43

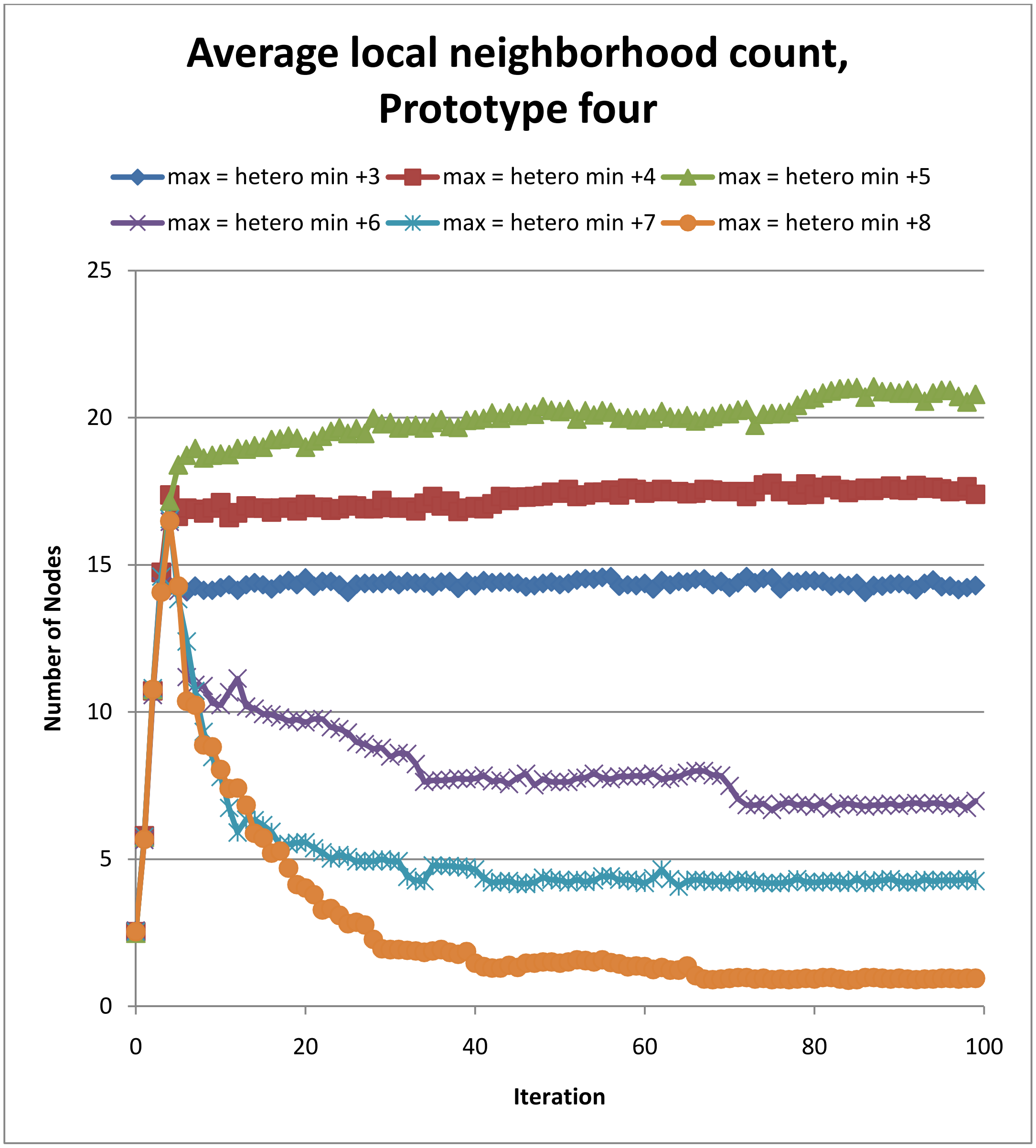


Figure 7.41: Average local neighborhood count, full implementation of our algorithm for mixed strategies, prototype four, *n* = 43

Figure 7.41, shown above, gives the average number of nodes to which another node is directly connected. We examined the local neighborhood count in order to study whether there was any correlation between the number of nodes to which another node is directly connected and the overall network security. In this case the performance of prototype four is similar to that of prototype three versions one and two, as shown earlier in Figures 7.20 and 7.30. However, the networks with the greatest maximum security, as shown in Figure 7.38, were the networks with the smallest average local neighborhood count.

7.4.1 Prototype Four Version Two

The objective of prototype four version two was to test whether the increased effectiveness in side payments had the same decrease in convergence time as with prototype three version two. Henceforth we shall identify the previous prototype four as prototype four version one. As shown in Figure 7.42 below, there was generally little or no effect. This is most likely caused by the mixed strategies extending the time to converge to maximum security. However, in some cases the more effective side payments actually increased convergence time versus the less effective side payments, which we did not expect. We hypothesize that this is the result of increased error in security calculations when a node makes a side payment to a second node, but the second node chooses a suboptimal decision and thus fails to act on the side payment. However, we do not know why the extended convergence time occurred for some networks but not others. Compare these results with the same diagram for prototype four version one as shown in Figure 7.38 above. This phenomenon bears future study.

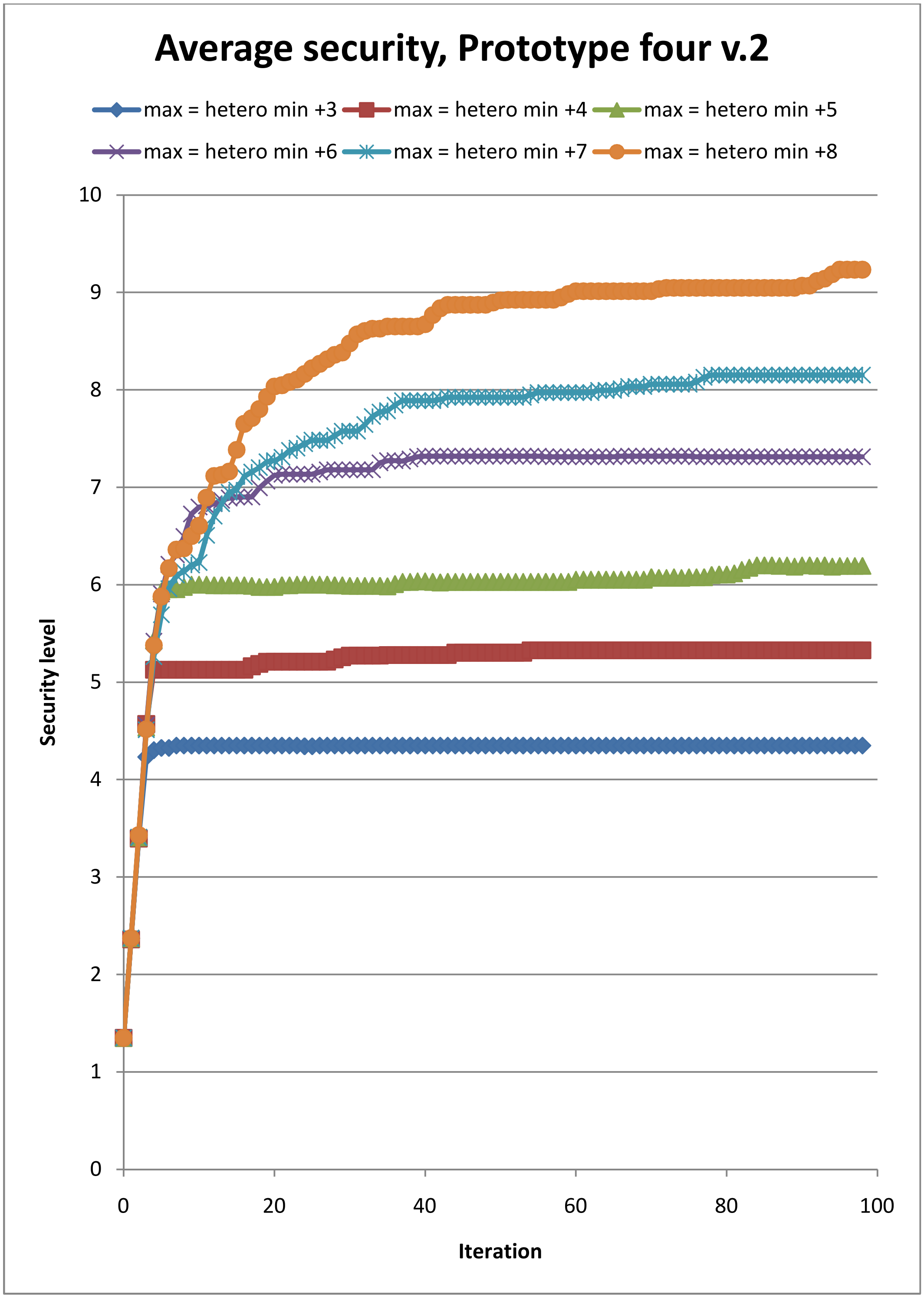


Figure 7.42: Average security, full implementation of our algorithm for mixed strategies, prototype four version two, *n* = 43

7.5 Kerberos comparison

The objective of developing a prototype of the well-known Kerberos algorithm is to compare our own algorithm's effectiveness versus an established algorithm in a manner that can be consistently and accurately measured. Using the Kerberos algorithm which implements the Kerberos system, we developed a prototype and compared its results with that of the prototypes of our own game theoretic algorithm. We normalized the definition of high security as defined by our metric onto Kerberos, thus correlating with the three highest security levels which correspond to eight through ten. Since until a ticket $K(r, w)$ is granted by the Kerberos server for the session, the definition of security for a node is incomplete as it consists only of encryption y; thus until the ticket is granted encryption strength y represents security.

The first result of testing the Kerberos prototype is shown in Figure 7.43 below for the nodes able to participate; here we see the advantages and disadvantages of Kerberos exemplified. Kerberos performs very well when nodes in a network can reach a sufficiently high level of security, but its exclusivity as shown in Figure 7.44 prevents other nodes from participating in the network at all, resulting in a loss of network connectivity. Note that in Figure 7.44, even when all nodes in a network can reach security level eight (as can be done with the max = hetero min plus 7 network), a sufficiently high number of nodes are still excluded from joining the Kerberos network. We did not expect this to happen. While it is possible we established constraints on security that were too high, we attempted to follow the pure Kerberos system as closely as possible, and believe that that the high security constraints of the system are what is indeed excluding those nodes from participating in the network.

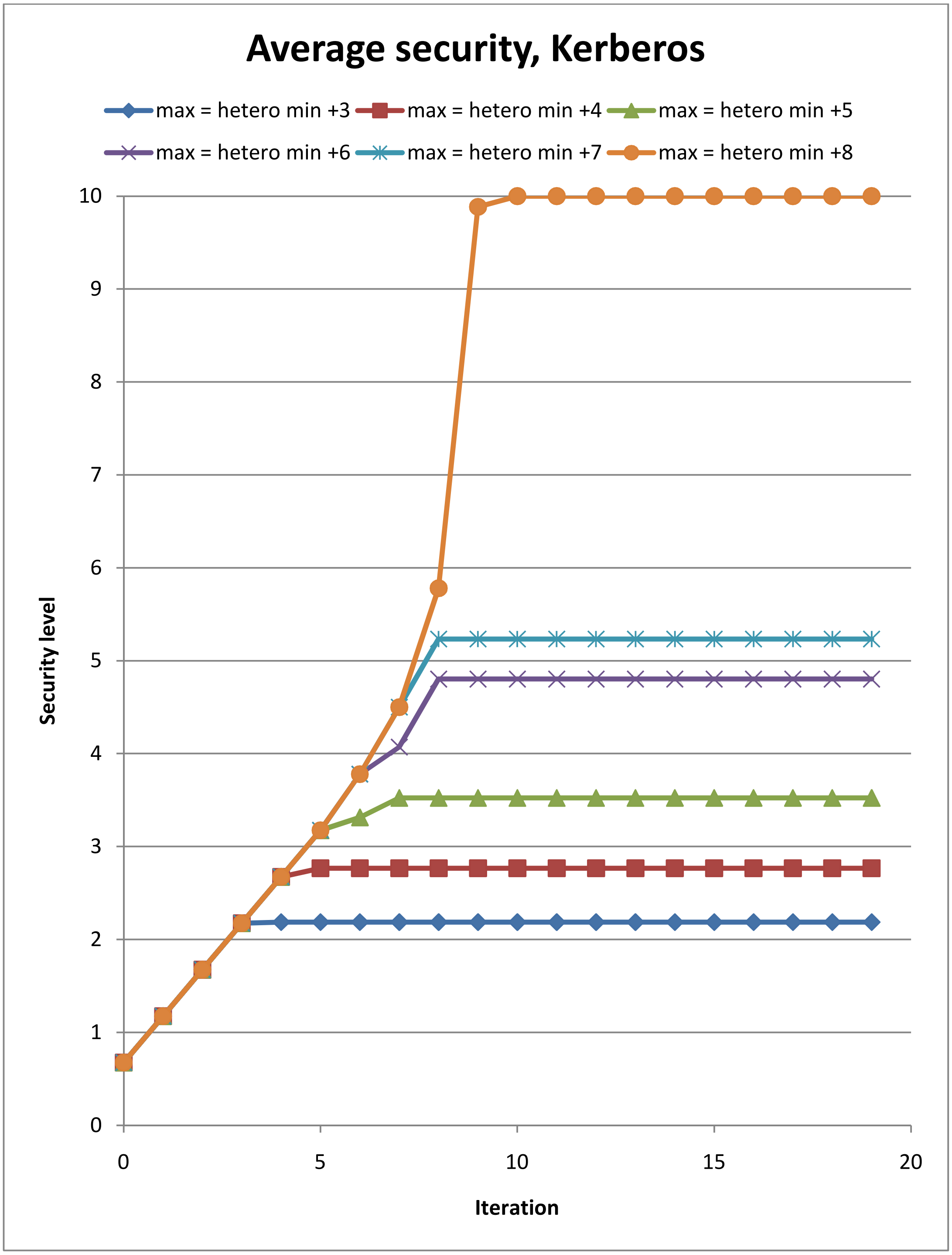


Figure 7.43: Kerberos network average security, participating nodes, n = 43

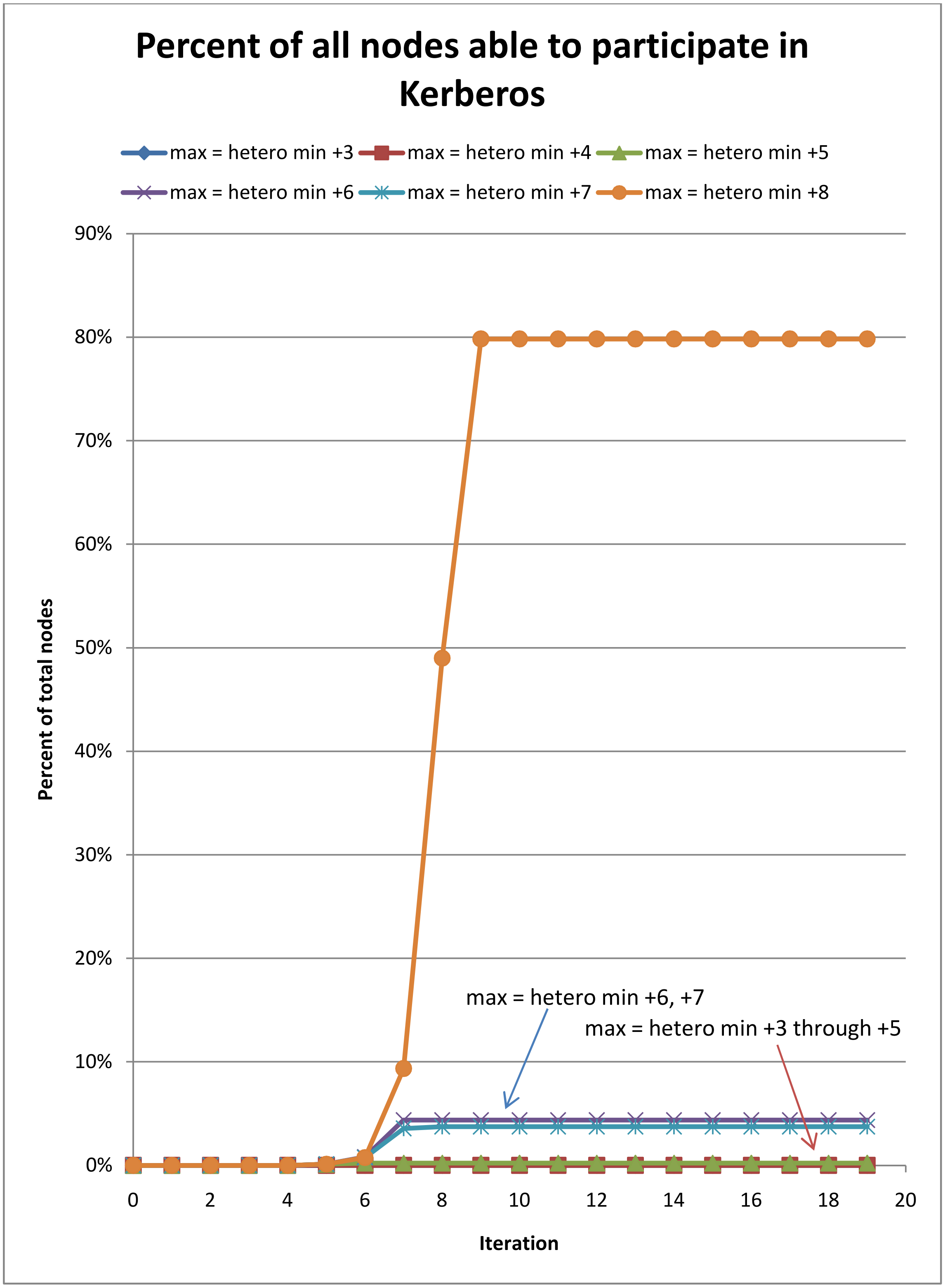


Figure 7.44: Kerberos network, variations on maximum security, percent of all nodes able to participate, *n* = 43

As demonstrated above in Figure 7.44, a significant number of nodes are unable to participate in the Kerberos network. In all but one of the networks tested, less than 10% of the nodes were able to join the network. Even for the network where all nodes were able to reach high encryption, 20% of the nodes were unable to join the Kerberos network due to security restrictions imposed by the Kerberos ticket-granting server. Contrast this with the definition of our own algorithm designed to optimize security for heterogeneous networks, in which no node is excluded. Our prototypes three and four have no graph corresponding to Figure 7.44 because no nodes are excluded by our algorithm.

In Figure 7.45, we extend the average security comparison of Figure 7.37 from earlier to include Kerberos. This chart will be used for comparison to Figures 7.59 – 7.61, which contain tests of all prototypes on different network sizes. Prototype one, while not fully implementing our security algorithm with respect to cost function, bids, coalitions, and side payments, is included in Figure 7.45 because of this comparison which is below. However, when one examines its numerical quantization, prototype one's performance is thus somewhat deceptive compared to the other prototypes. Again, this is due to the fact that prototype one was used for initial tests, and only considered a limited number of networks versus later prototypes. Recall that the networks tested in prototype one had high security. Networks had either homogeneous maximum security, where all nodes could reach the maximum theoretical security of ten, or high heterogeneous maximum security that was six or seven levels above the heterogeneous minimum. Prototype one was designed to perform initial tests of our basic game theoretic algorithm. Prototype two and later prototypes were designed to fully test our algorithm. Because of its differences with the more fully developed prototypes, prototype one's security quantization should be considered more ordinal than exact and can thus be considered similar to the performance of the prototypes two through four.

In Figure 7.45, Kerberos performs poorly compared to our algorithm's prototypes. In most cases fewer than 10% of the nodes met the security requirements to join the Kerberos network, leaving the remainder to have a decreased level of security. As shown in Figure 7.44, at least 20% of all nodes are unable to participate and join the Kerberos network.

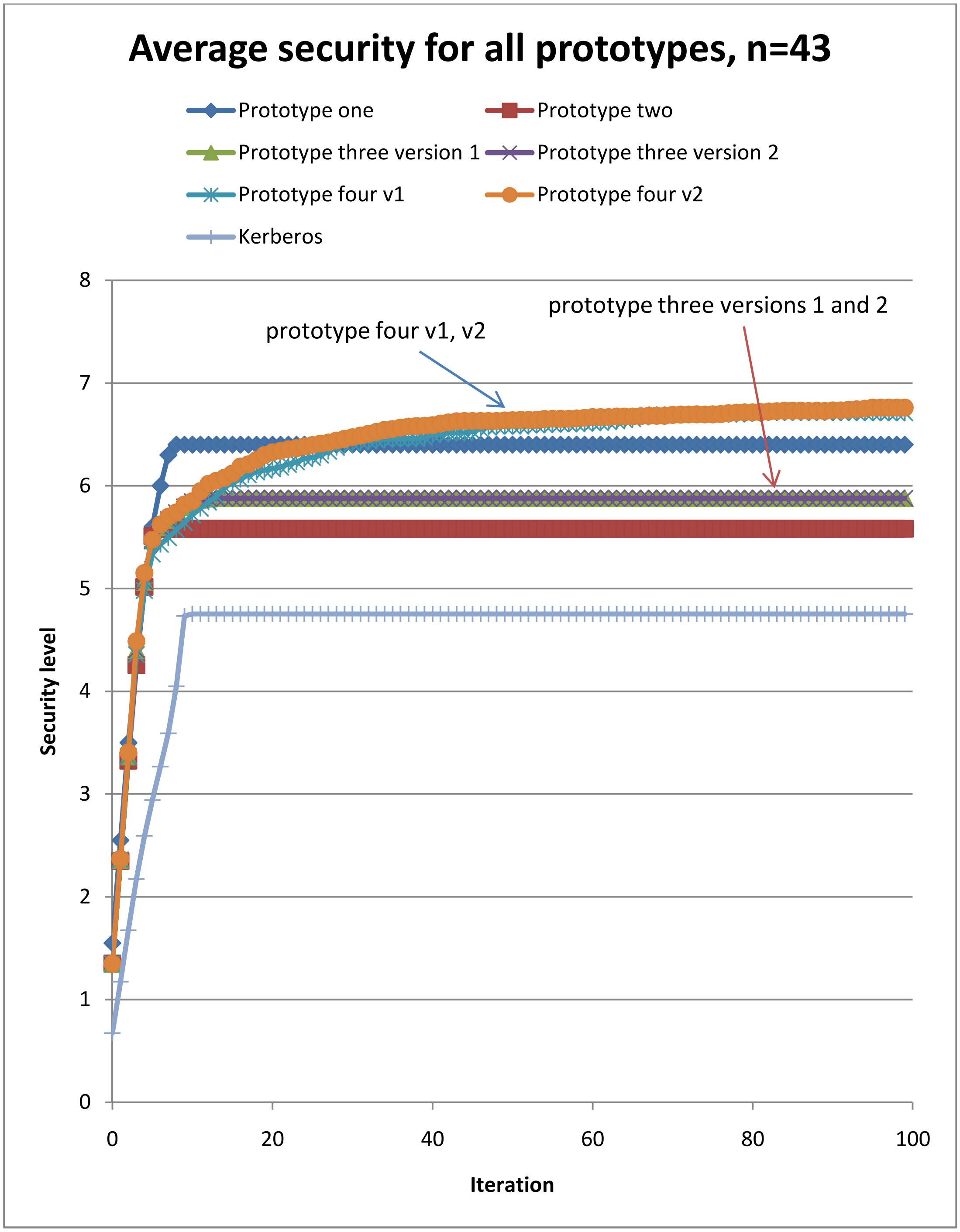


Figure 7.45: Average security comparison, all prototypes, *n* = 43. Compare with Figures 7.59 – 7.61 below for *n* = 20, 60, and 80, respectively.

As shown in Figures 7.45 through 7.50, Kerberos' weakness due to its inapplicability to heterogeneous networks is apparent. But comparing Kerberos to the performance of our own security algorithm in the same figures, the strength of our algorithm is apparent. No nodes are excluded in our algorithm's prototypes, and security is improved overall, for all possible members of the network. Still, for a more homogeneous network where most or all nodes are able to reach high levels of encryption, as in Figure 7.51, Kerberos can provide improved average security and much faster convergence versus our own security algorithm. Kerberos performs better when the nodes in the network can reach the highest security. Note that prototype one is excluded from the graphs in Figures 7.46 through 7.51 as it did not test all of these networks. These graphs are below.

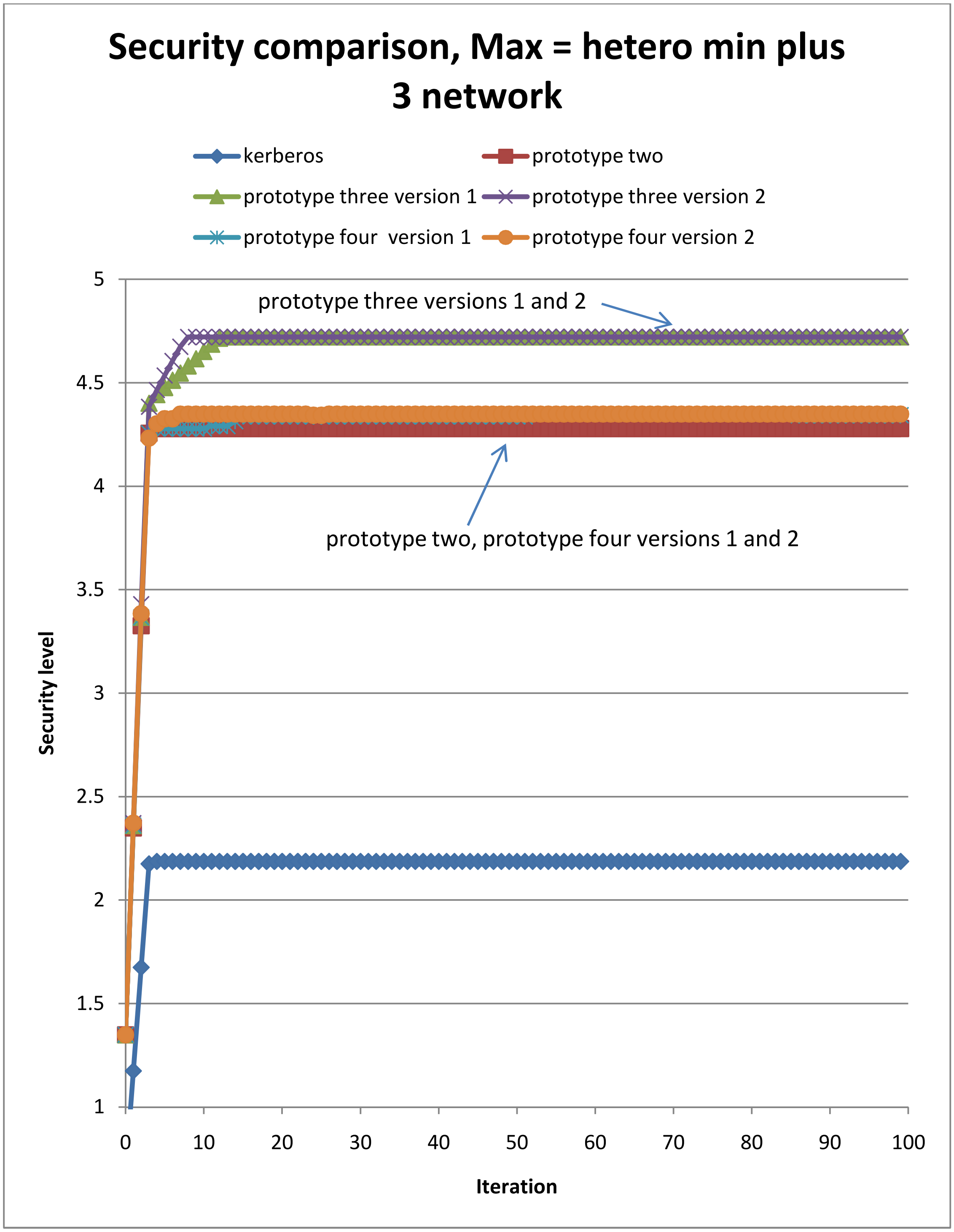


Figure 7.46, *n* = 43, security comparison for a network where the maximum security is equivalent to three levels above a node's minimum security.

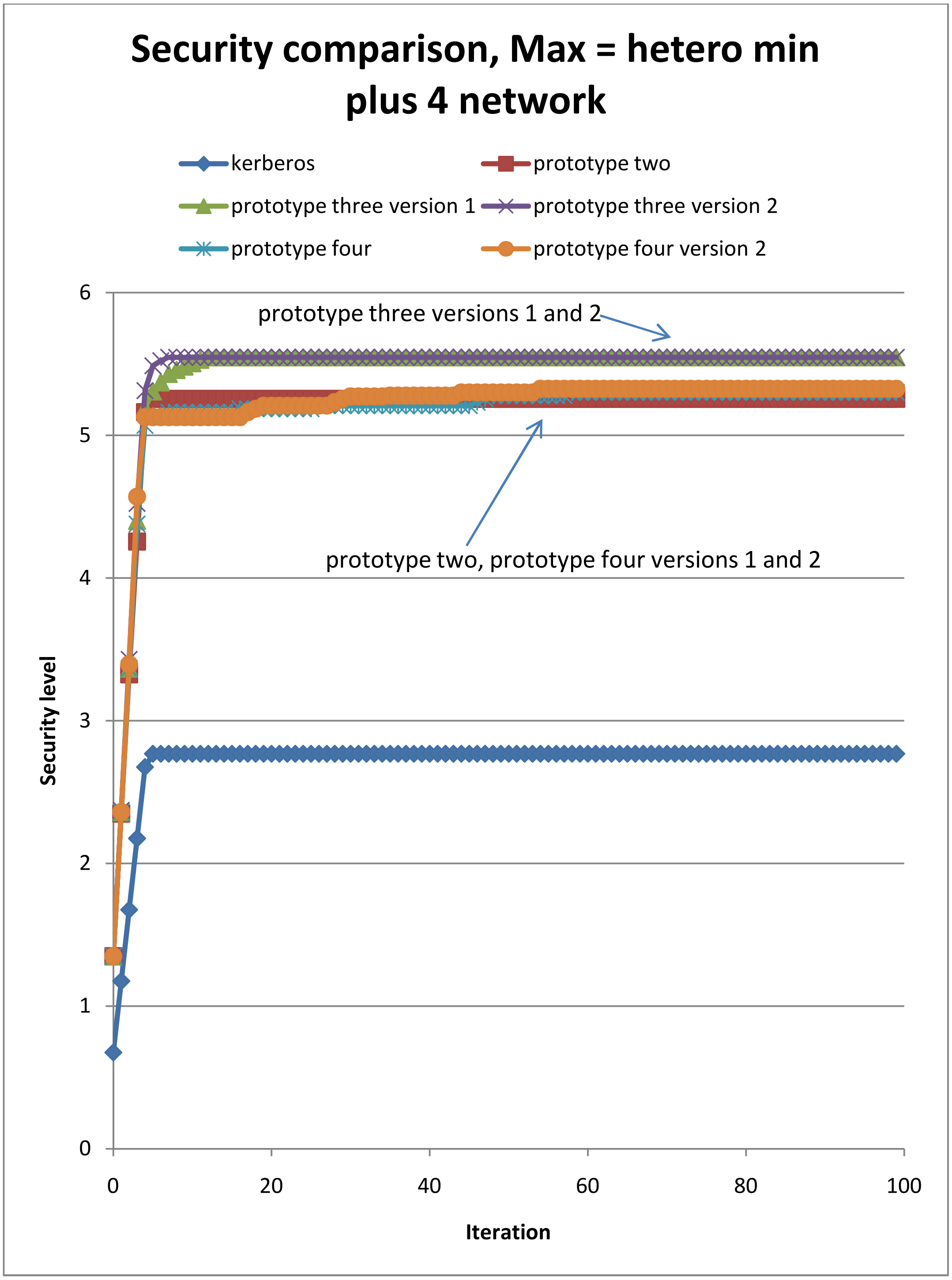


Figure 7.47, *n* = 43, security comparison for a network where the maximum security is equivalent to four levels above a node's minimum security

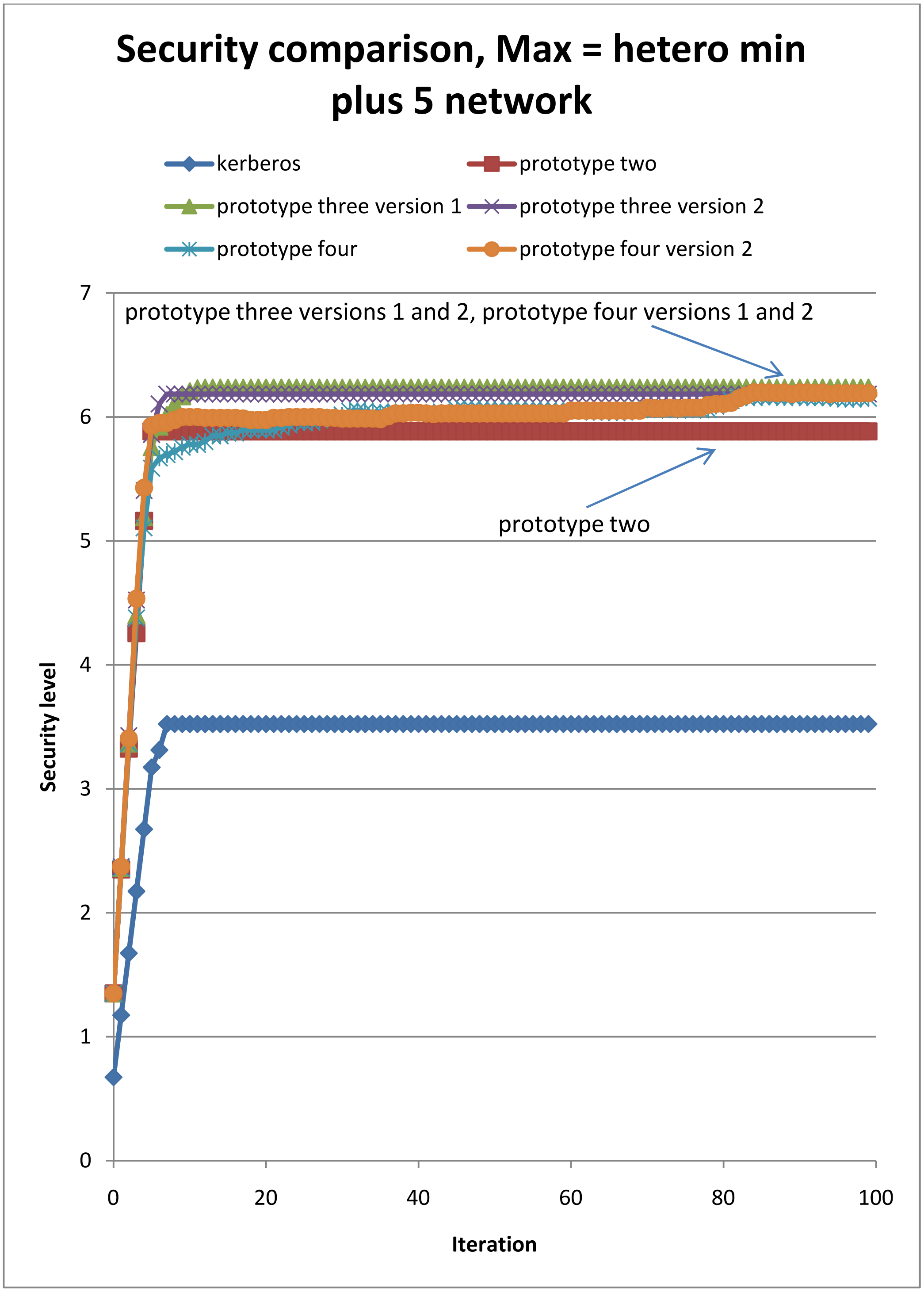


Figure 7.48, *n* = 43, security comparison for a network where the maximum security is equivalent to five levels above a node’s minimum security

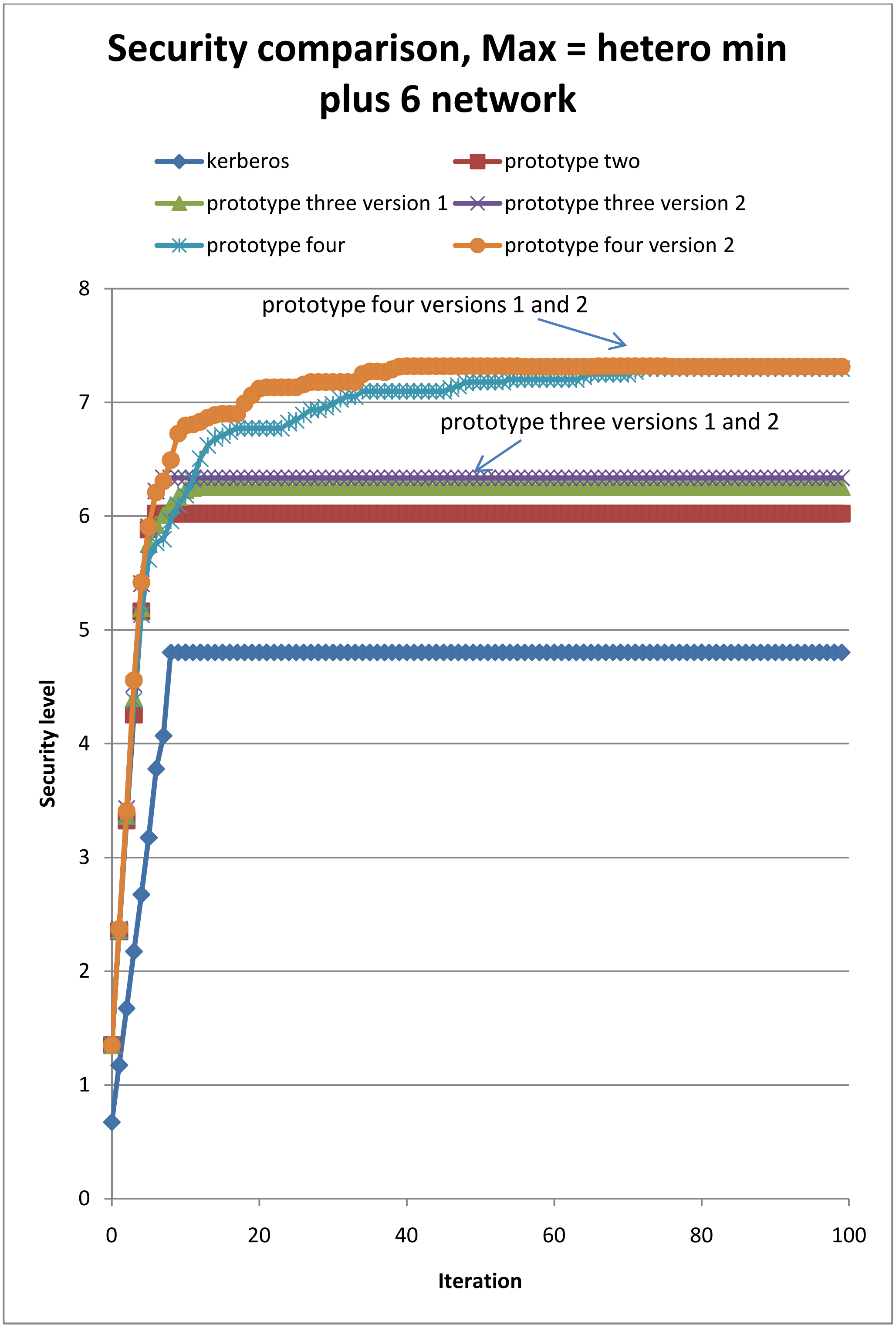


Figure 7.49, *n* = 43, security comparison for a network where the maximum security is equivalent to six levels above a node's minimum security

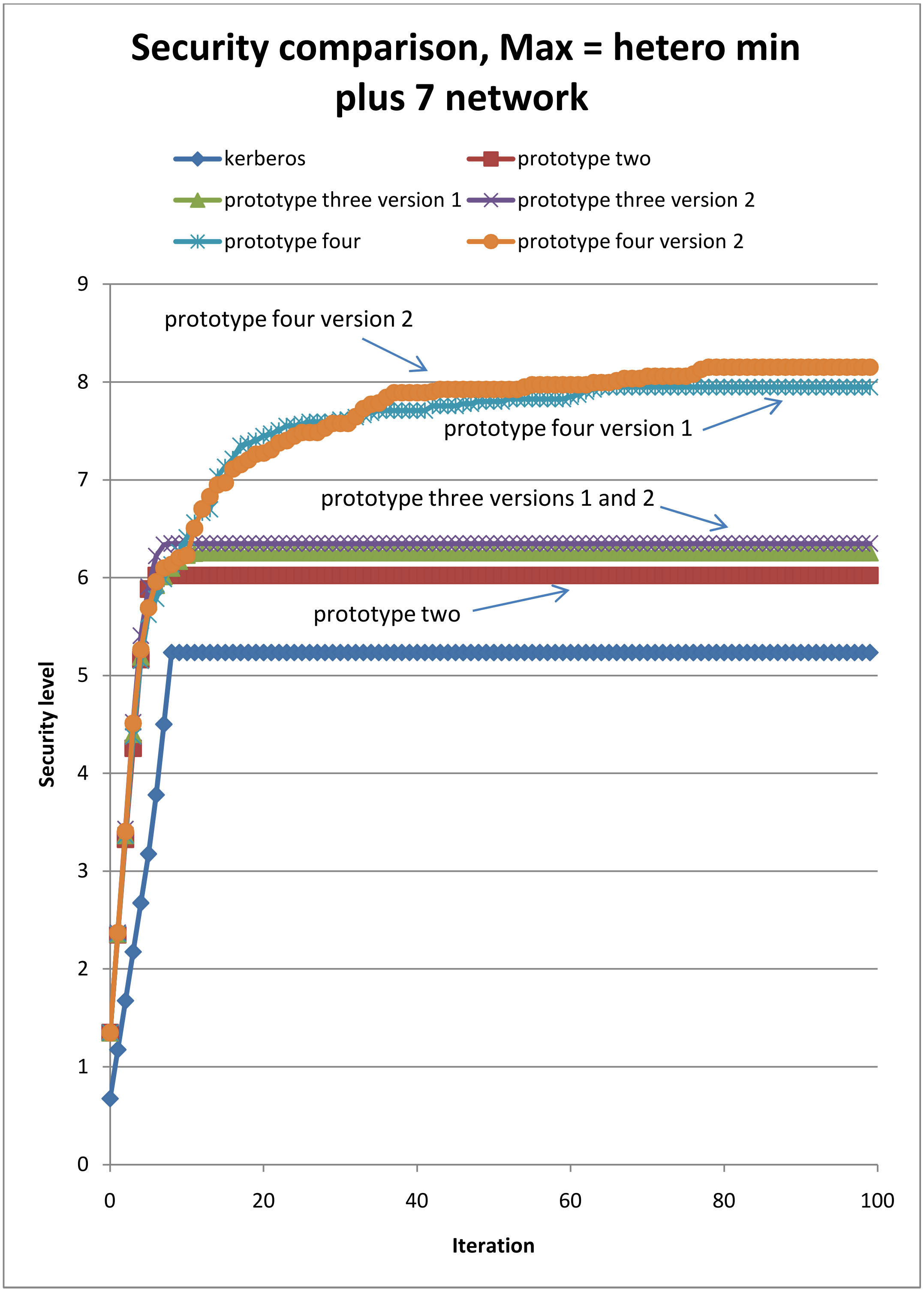


Figure 7.50, *n* = 43, security comparison for a network where the maximum security is equivalent to seven levels above a node’s minimum security

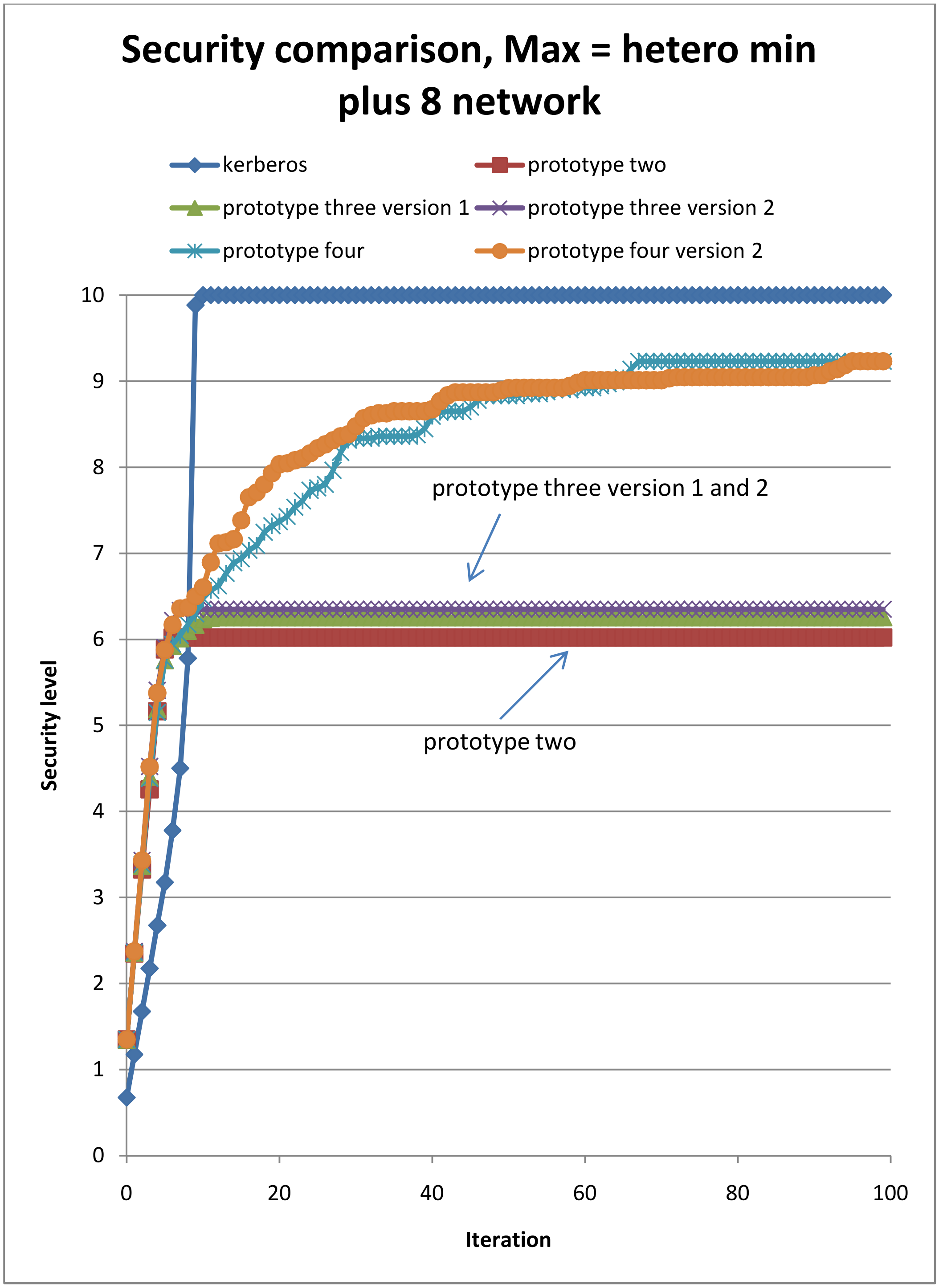


Figure 7.51, *n* = 43, security comparison for a network where the maximum security is equivalent to eight levels above a node's minimum security

In Figure 7.51, we see that Kerberos is ideal for a network composed primarily of nodes that can reach maximum security. However, prototype four performed nearly as well despite the longer convergence time. This may not be a fair comparison, however, since prototype four implements mixed strategies while Kerberos only implements pure strategies.

7.6 Varying network size: a comparison

The purpose of the following tests of security for prototypes one through four is to examine whether varying the size of the network, *n*, has an effect on the network security for our game theoretic algorithm. The above prototypes were run on our first network, which as mentioned earlier, had 43 nodes. To determine the effect increasing or decreasing the number of nodes has on the network security, we will examine networks with 20, 60, and 80 nodes in our 10 by 10 matrix averaged over the networks with varying heterogeneous maxima, as applicable. Similar tests for Kerberos are also shown below. All nodes were heterogeneous. For the network of 20 nodes, fourteen nodes started at minimum security of 1, three nodes started at minimum security of 2, two nodes started at minimum security of 3, and one node started at minimum security of 5. The average minimum security was 1.55. For the network of 60 nodes, fifty nodes started at minimum security of 1, seven nodes started at minimum security of 2, two nodes started at minimum security of 3, and one node started at minimum security of 5. The average minimum security was 1.25. For the network of 80 nodes, seventy nodes started at minimum security of 1, seven nodes started at minimum security of 2, two nodes started at minimum security of 3, and one node started at minimum security of 5. The average minimum security was 1.1875. These graphs follow below.

Our first graph, Figure 7.52, shows the effect varying the size of the network has on our first prototype, prototype one. Here we see a fairly consistent level of security among the varying network sizes; note that for 20 nodes the security was lowest, but beyond this size of network there was no significant difference between the average network security levels. This same phenomenon occurred for prototype two as well, which is shown in Figure 7.53. We believe that these results demonstrate the robustness of our core game theoretic security algorithm. Discussion of these results in Figure 7.52 – 7.55 follows Figure 7.55.

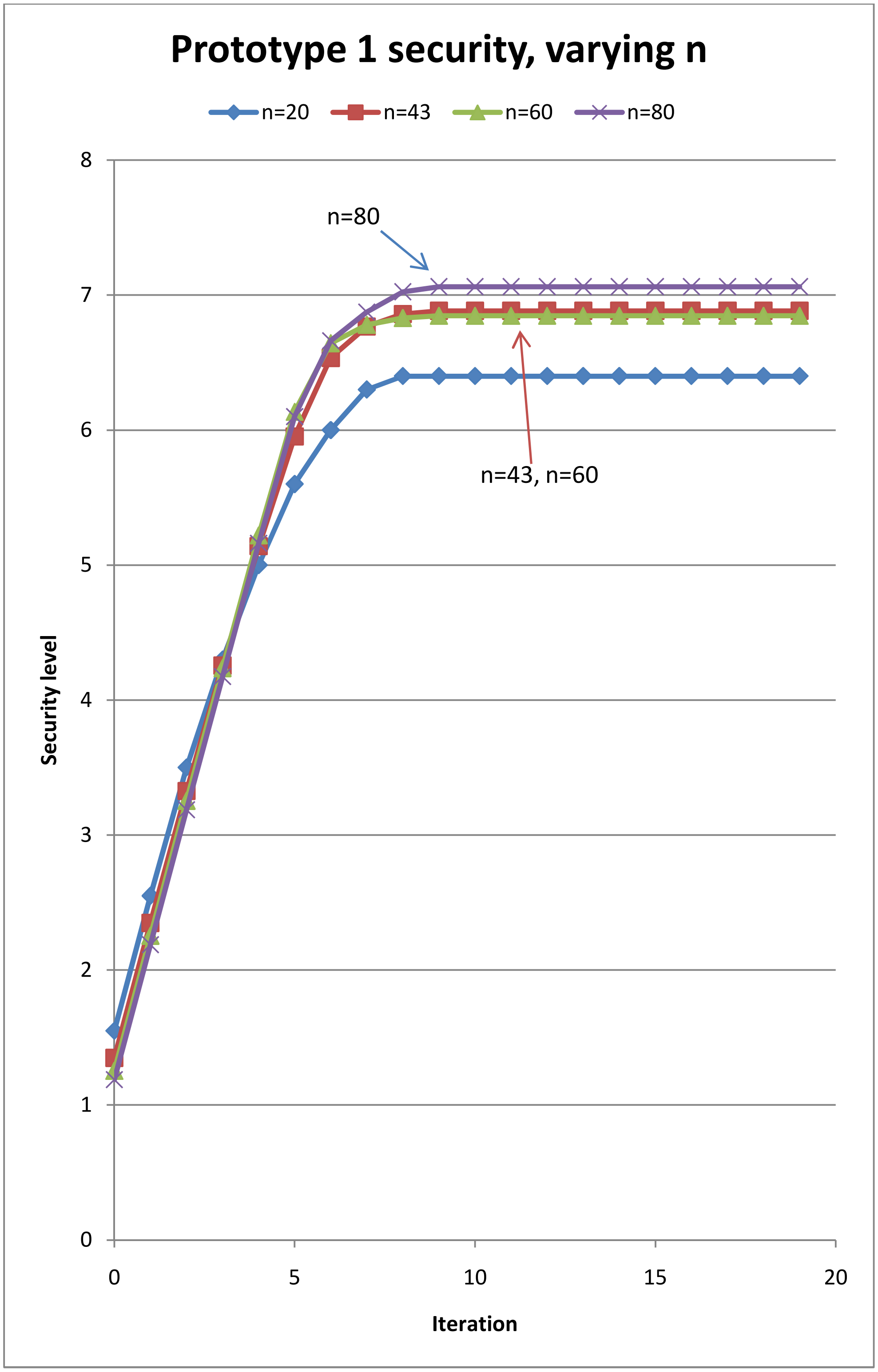


Figure 7.52: Security for prototype one, varying network size *n*

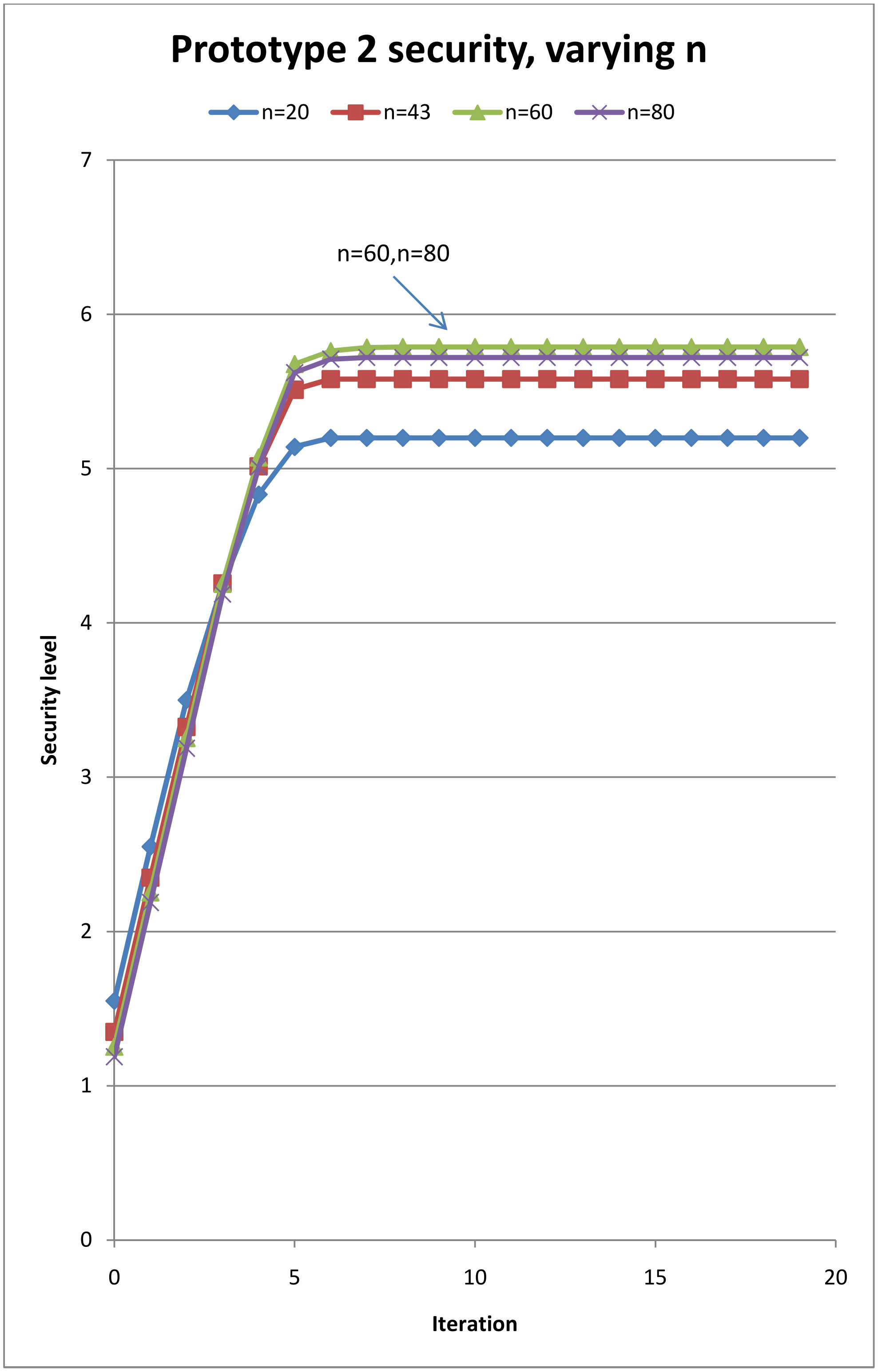


Figure 7.53: Security for prototype two, varying network size *n*

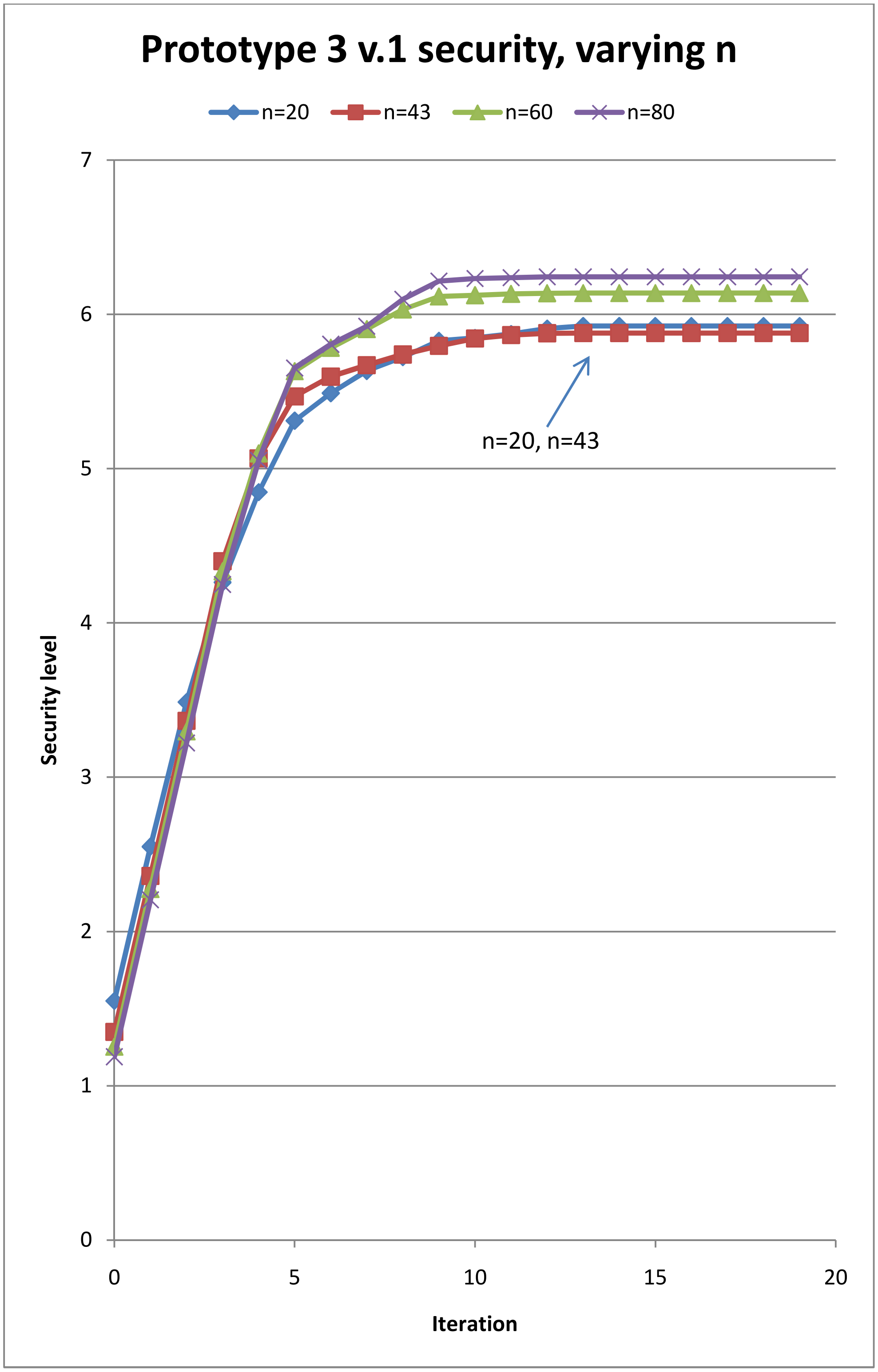


Figure 7.54: Security for prototype three version one, varying network size *n*

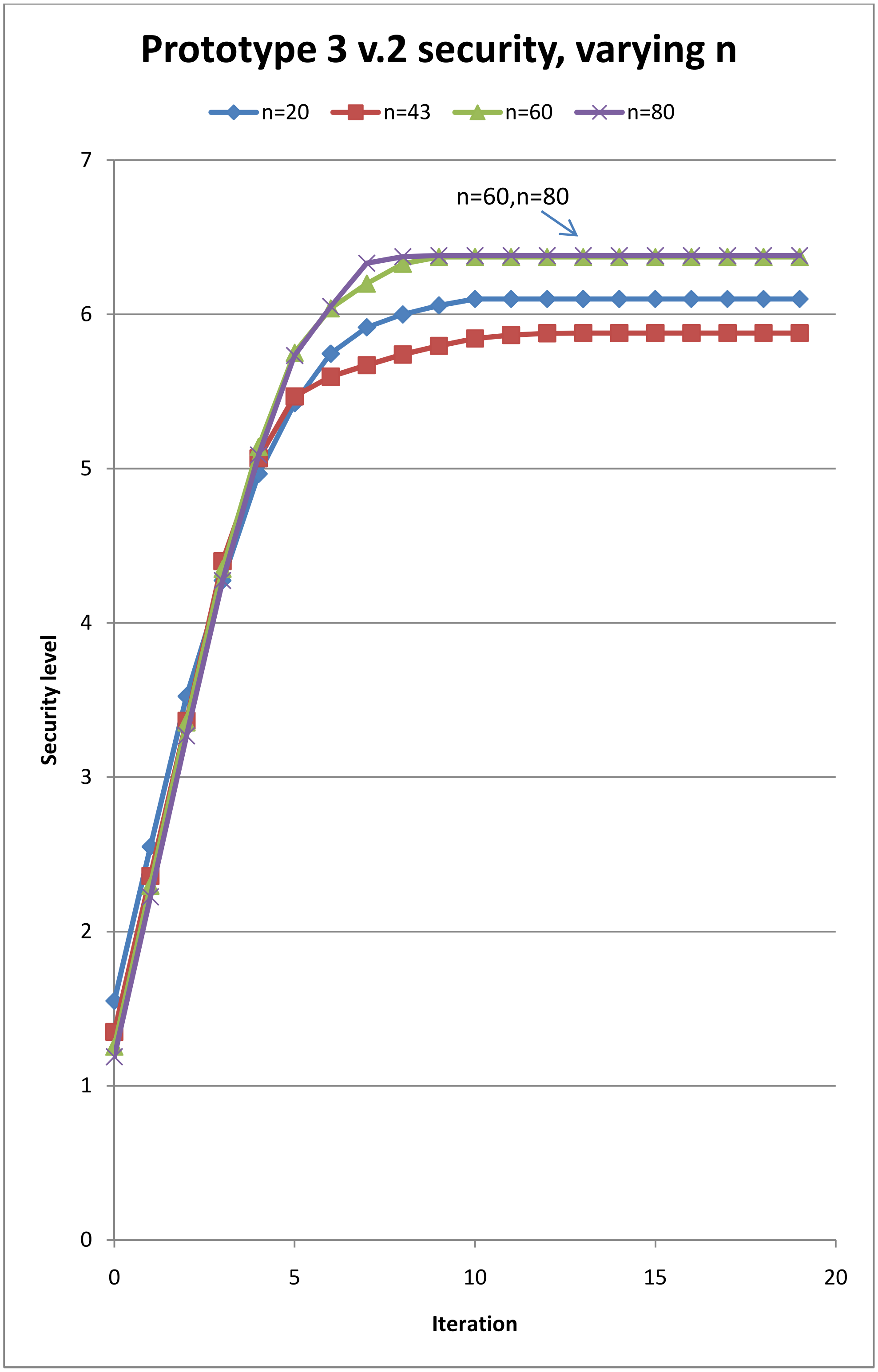


Figure 7.55: Security for prototype three version two, varying network size *n*

The results shown in Figures 7.52 and 7.53 indicate that varying the network size had little or no effect on prototypes one or two. The introduction of side payments brought on in prototype three versions one and two, shown in Figures 7.54 and 7.55, respectively, demonstrate improvements in security for all network sizes versus prototype two. The smallest network, *n*=20, showed the most improvement with the more effective side payments of prototype three version two. The larger networks, *n*=60 and *n*=80, had higher security than the smaller sized networks for either prototype. However, the middle-sized network, *n*=43, had the lowest or nearly the same security as the smallest network, *n*=20. For prototype four, however, we did not observe the phenomenon of larger networks having greater average security. See Figures 7.56 and 7.57 below. For prototype four version one, which had less effective side payments than prototype four version two, the smallest network of *n*=20 performed the best. However, for prototype four version two, all networks performed similarly. Note in Figure 7.57 (prototype four version two) there is a tighter bound in maximum security variations for network size versus prototype four version one in Figure 7.56. This tighter bound is clearly related to the more effective side payments, since that is the only difference between the two prototypes. It is evident in these graphs that the more effective side payments decreased security for larger networks, if only by a small amount.

Overall, however, the maximized average security for all networks stabilized between 6 and 7 for both prototypes. However, the network with *n*=43 nodes performed poorest. We do not know why this network had the lowest security. It may be possible that small networks optimized by mixed strategy games perform best, but if so there is no correlation beyond what we observed for a network of 20 nodes.

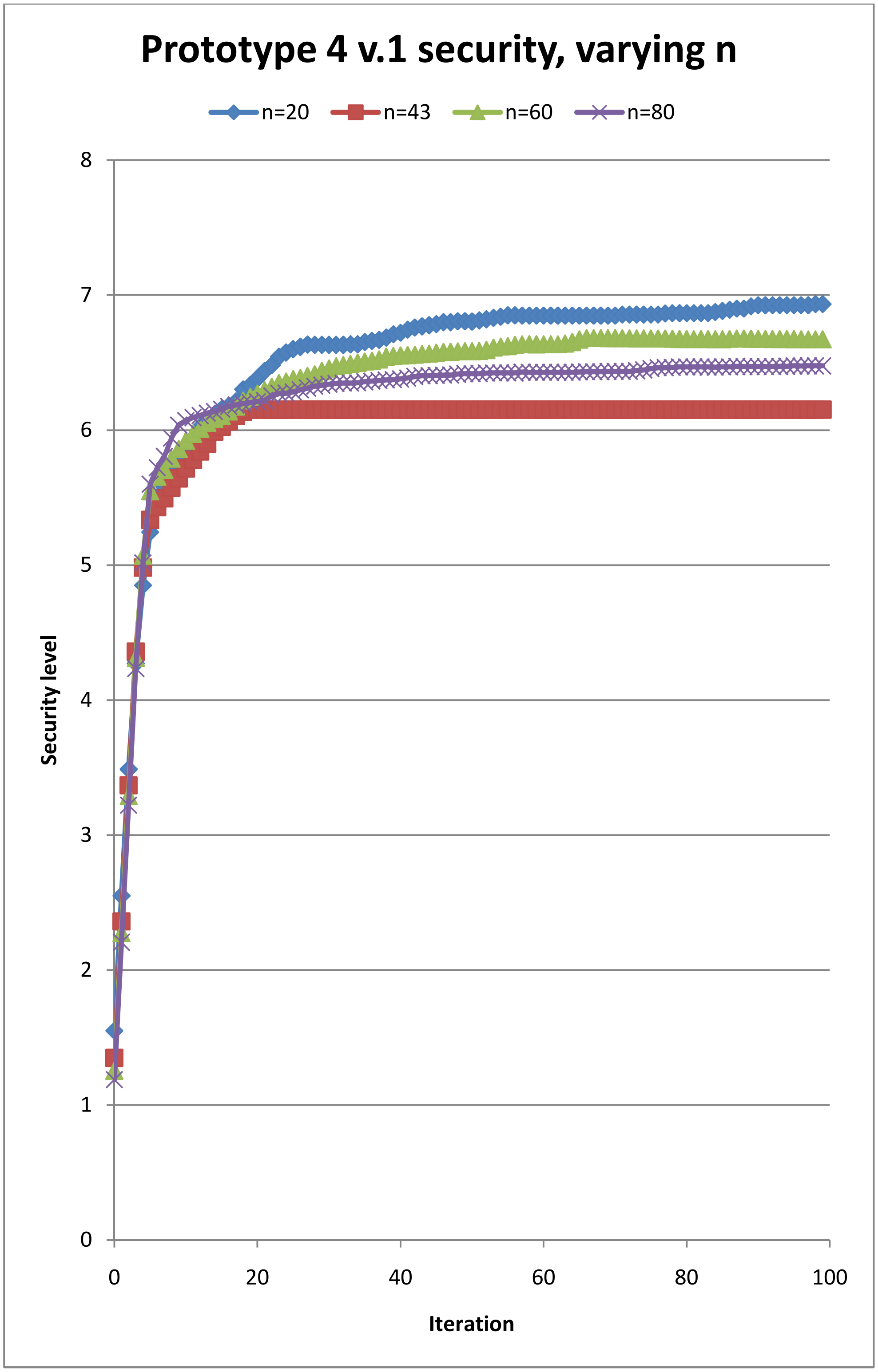


Figure 7.56: Security for prototype four version one, varying network size *n*

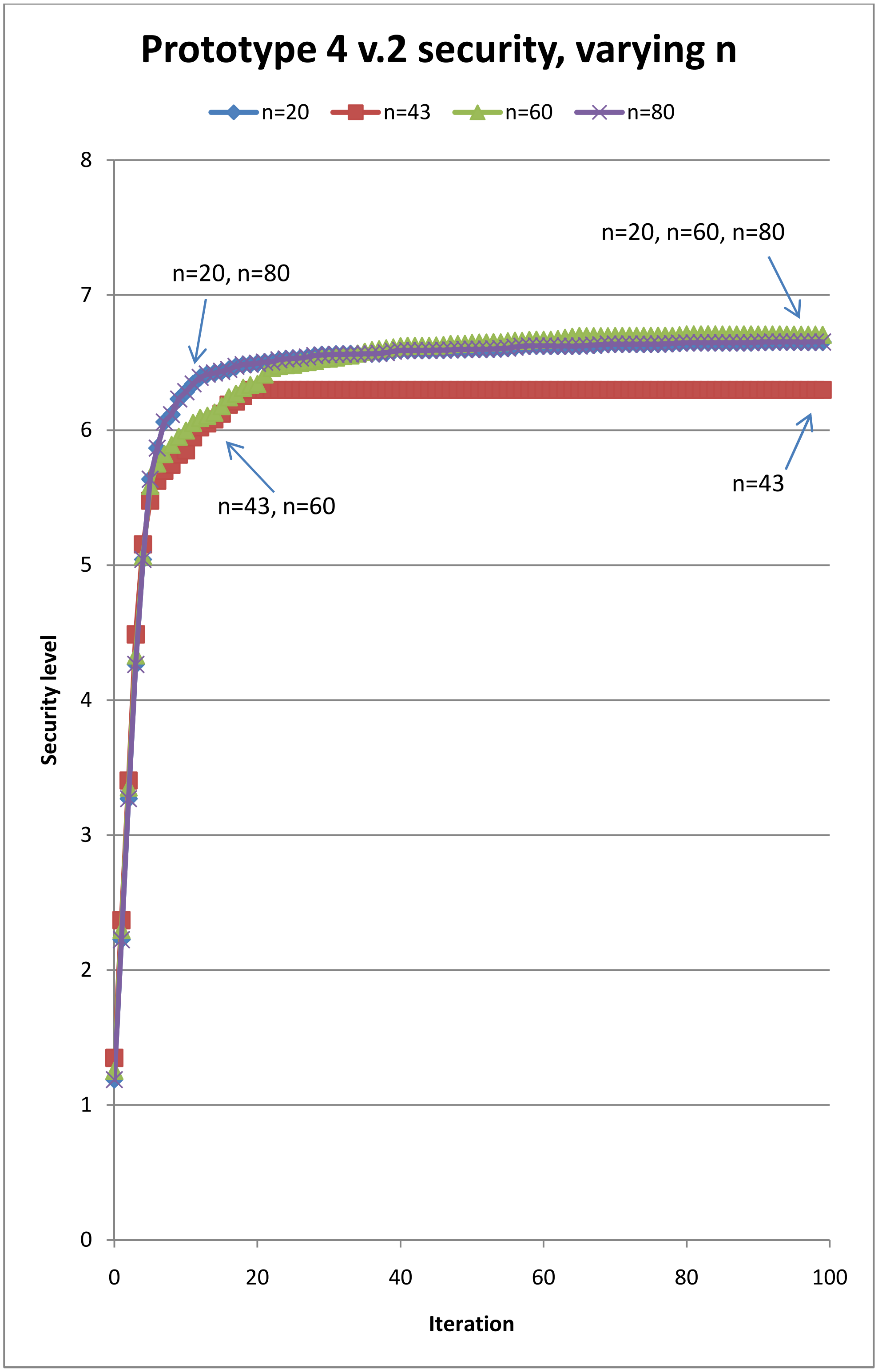


Figure 7.57: Security for prototype four version two, varying network size *n*

As shown below in Figure 7.58, Kerberos is also relatively consistent for varying network size, but shows an inverse relationship between network size and achieved security. This may be caused by the decreased overhead of a smaller Kerberos network resulting in improved efficiency and security.

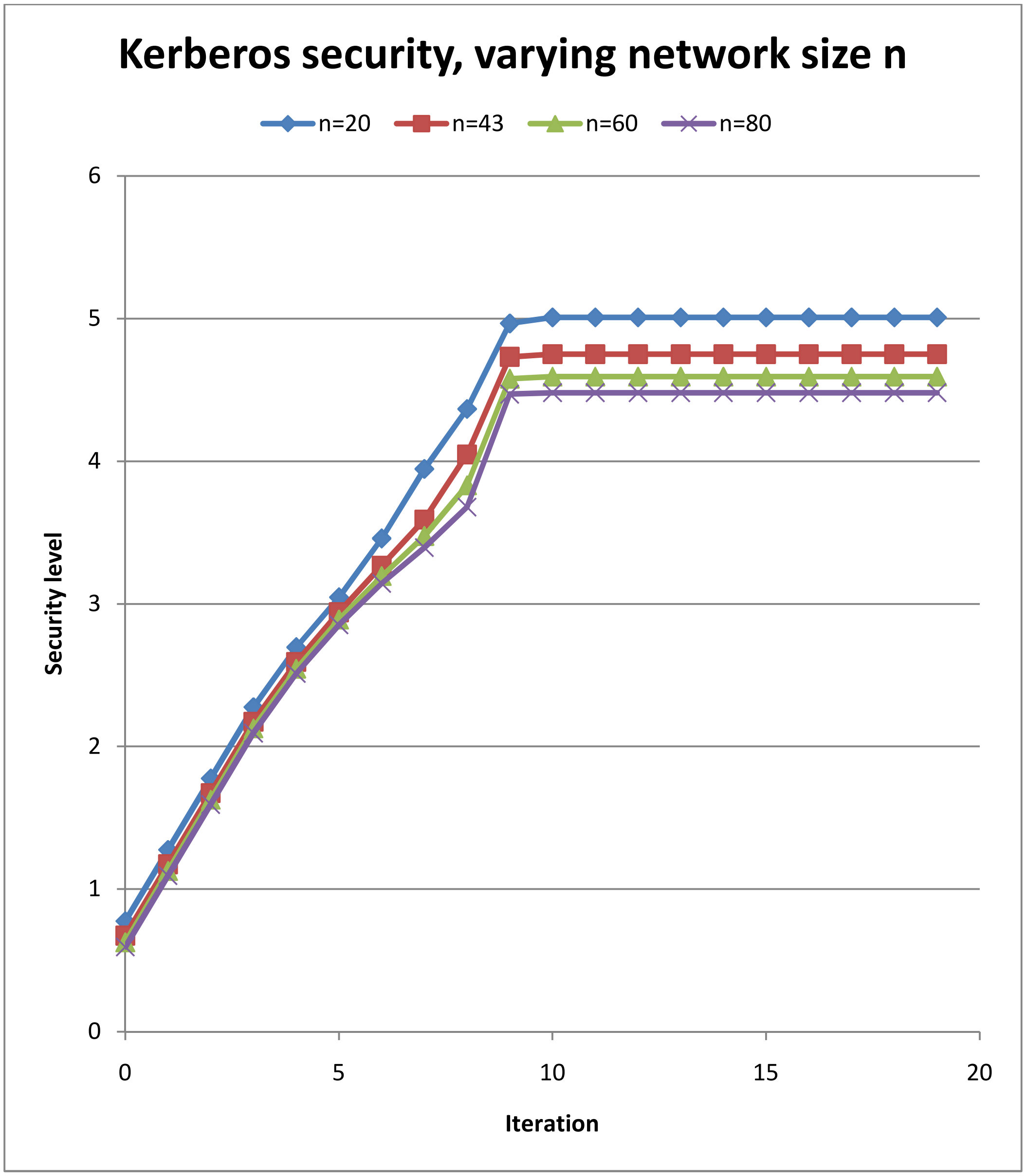


Figure 7.58: Security for Kerberos network, varying network size *n*

In Figures 7.59 through 7.61 we see the average security compared for all of our prototypes, including Kerberos. Network size varied from 20 to 60 to 80 nodes. The graph for the network of 43 nodes is in Figure 7.45 above. In Figure 7.45 and Figures 7.59 through 7.61, we observe Kerberos is consistently poorest when averaged over networks varying by heterogeneity, with the exception of the network consisting of 20 nodes. Instead, Kerberos performs approximately as well for security as prototype two, which implemented coalitions but no side payments; prototype two, however, converged more quickly to stable network security.

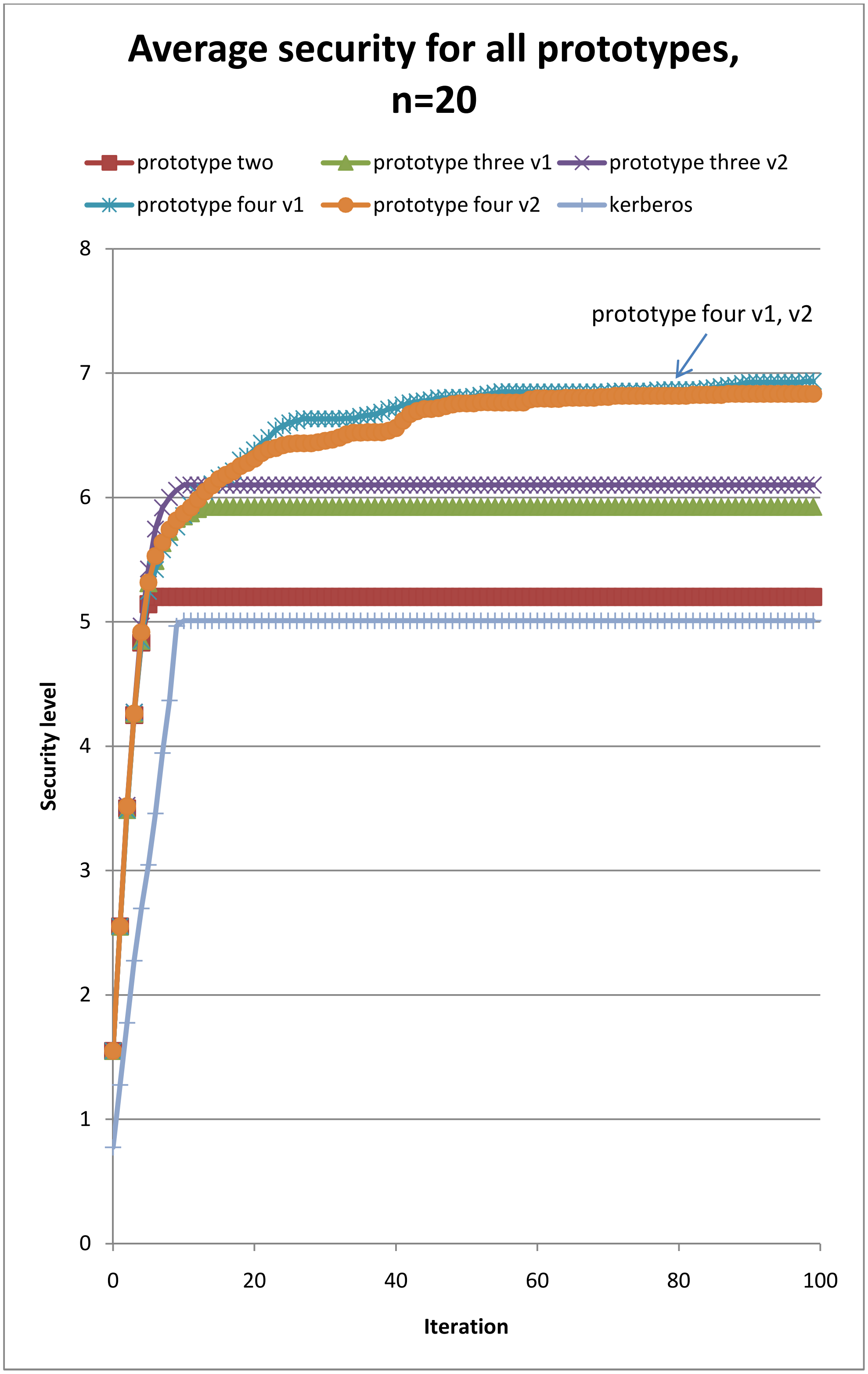


Figure 7.59: Average security comparison of all prototypes, *n* = 20

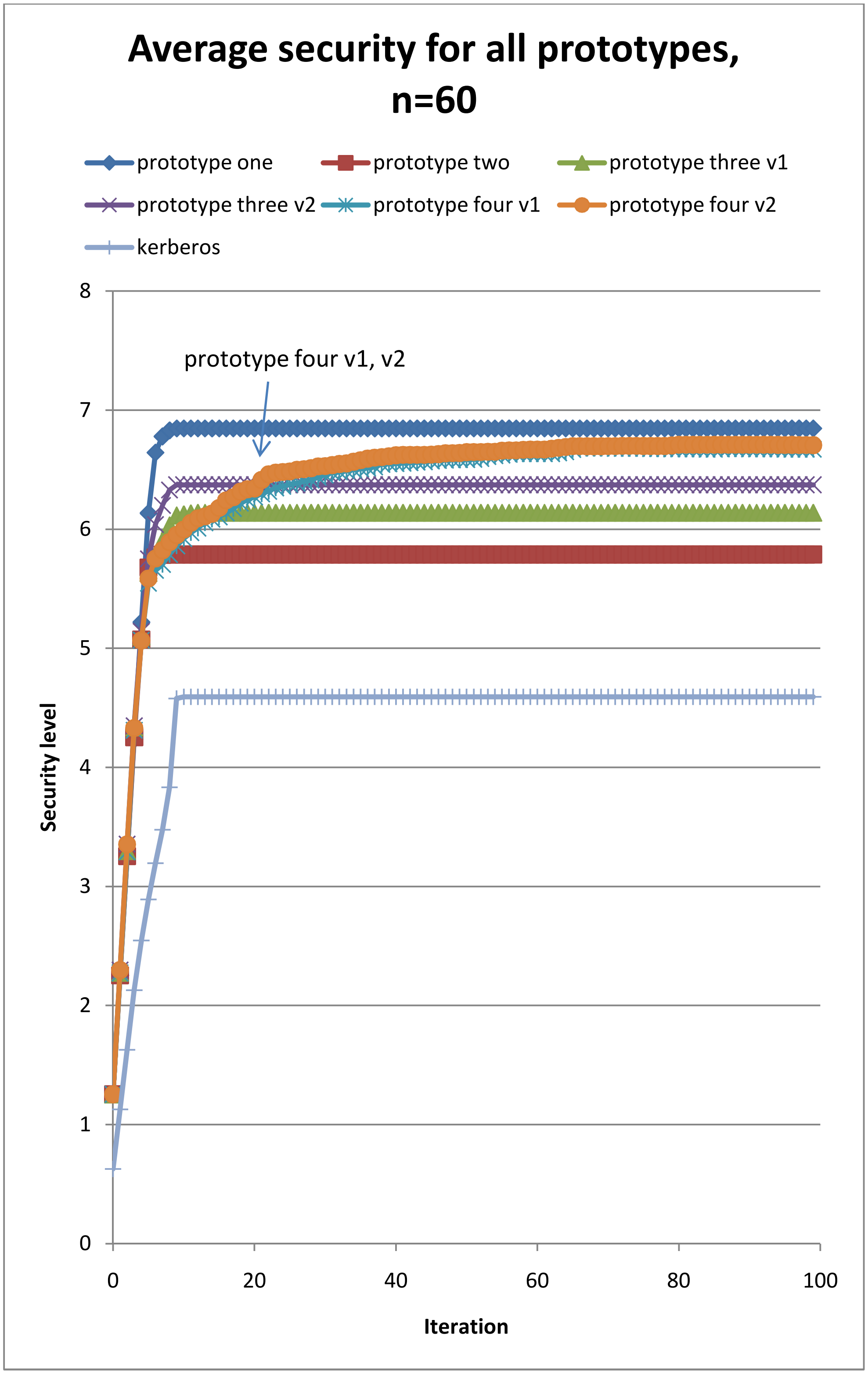


Figure 7.60: Average security comparison of all prototypes, *n* = 60

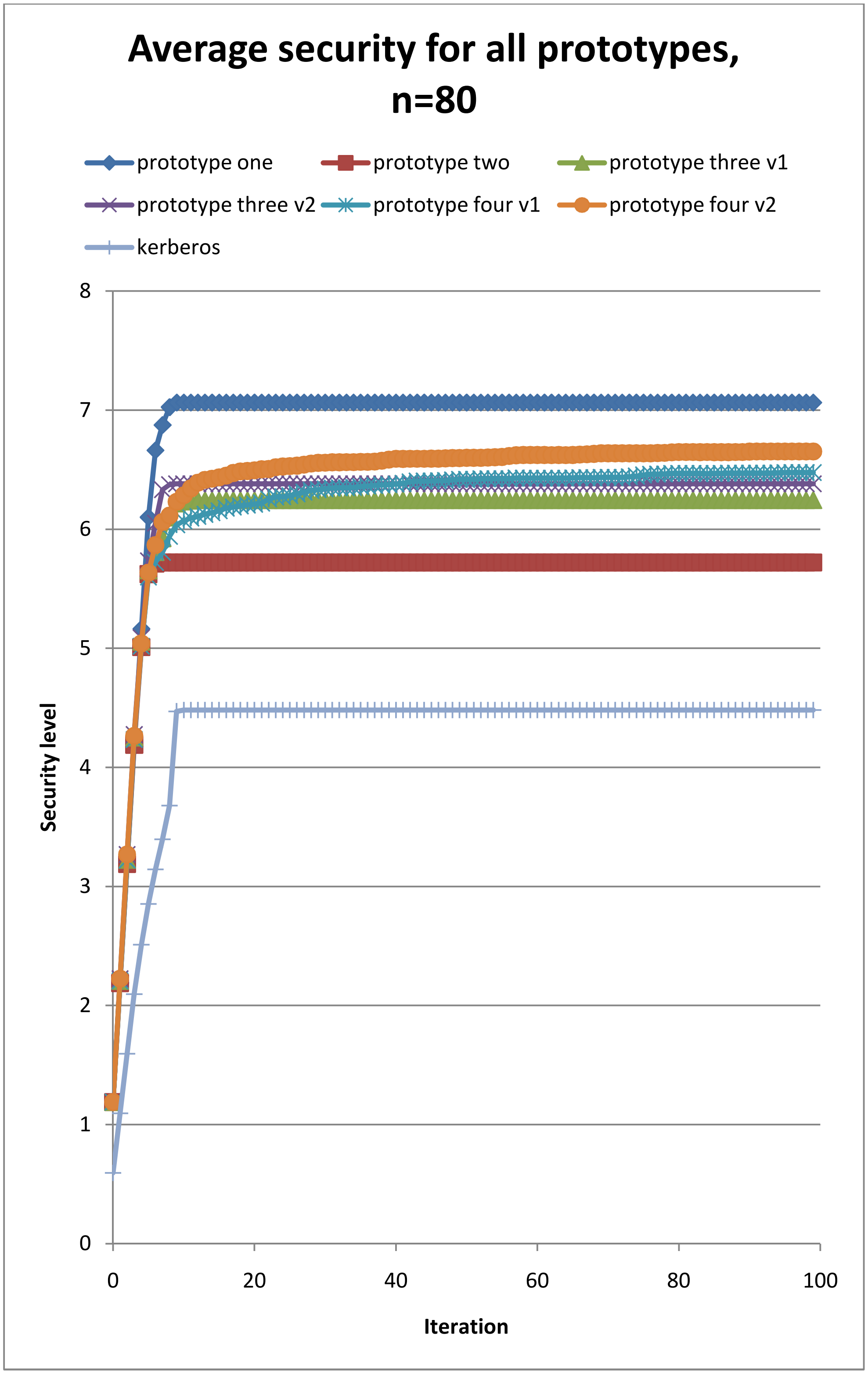


Figure 7.61: Average security comparison of all prototypes, *n* = 80

These graphs in Figure 7.45 and Figures 7.52 – 7.61 demonstrated the improvement in network security made by prototype implementations of our game theoretic algorithm regardless of network size. Again, our algorithm includes all nodes in any network, regardless of heterogeneity, which is something that Kerberos fails to do. As a result, our algorithm improves network security and is better suited to optimize security for a more heterogeneous network than Kerberos. Our detailed summaries are discussed below the overview of our results is given in Table 7.4.

	Theor. Max. Security Levels	**Avg. Security**	**Convergence**	**Nodes Included in Network**
Proto 1	Min_{avg} + 6 to 8	High	Fast	All
Proto 2	Min_{avg} + 3 to 8	Mid	Fast	All
Proto 3.1	Min_{avg} + 3 to 8	Mid	Fast	All
Proto 3.2	Min_{avg} + 3 to 8	Mid	Fast	All
Proto 4.1	Min_{avg} + 3 to 6	Mid	Fast	All
	Min_{avg} + 7 to 8	High	Slow	All
Proto 4.2	Min_{avg} + 3 to 5	Mid	Fast	All
	Min_{avg} + 6 to 8	High	Slow	All
Kerb.	Min_{avg} + 3 to 5	Low	Fast	<10%
	Min_{avg} + 6 to 7	Mid	Fast	<10%
	Min_{avg} + 8	High	Fast	80%

Table 7.4: Table of Prototypes and General Results

7.7 Detailed summary of results

When comparing different networks with varying heterogeneity with respect to minimum and maximum security, the results of our first prototype show that our game theoretic security algorithm converges faster for more heterogeneous networks. The achieved security for this network was, naturally, somewhat decreased when nodes could not reach the same maximum, but the security was not very different from the network where all nodes could reach the same maximum. Varying the size of the network had little effect on the overall network security.

The objectives of our second prototype were to develop a baseline before studying the effect of side payments and to study the results of a near-full implementation of our game theoretic security algorithm. As with prototype one, our second prototype was largely unaffected by variations in network size. With regard to the relationships between nodes forming the network, in particular coalitions, we saw a great deal of divergence among the different networks with varying degrees of maximum heterogeneous security. Nodes with a greater heterogeneous maximum granted fewer nodes access to sensitive data once security stabilization was reached.

The objective of prototype three was to study the effect on overall network security by adding side payments to implement a full version of our game theoretic security algorithm. As in prototype two, the third prototype achieved a consistent maximum security nearly equal to their theoretical maximum for the respective network. This similarity between prototypes demonstrates that the core of our security algorithm is sound, improving overall network security regardless of the degree of heterogeneity. We observed a correlation between those networks that maximized the use of side payments with those that performed best in terms of security. We conclude that there may be an optimum percentage of nodes making side payments which allow a network to maximize its overall security, since we observed that once side payments dropped below 20% they had little overall effect on security. Whether this can be used to reduce energy consumption or improve security will need to be explored in the future.

Comparing prototypes two and three with respect to convergence time, prototype three took longer to converge to stable network security. This extended convergence time is clearly brought on by the introduction of side payments, which are the only difference between prototypes two and three. However, when we increased the effectiveness of the side payment itself in terms of security, convergence time decreased substantially.   In some tests this prototype converged to maximum security nearly 50% faster than the prototype with less effective side payments. We also observed, for the same prototype, faster convergence with respect to maximum security, utility, and coalition formation.

The introduction of side payments in prototype three resulted in a significant decrease in the number of nodes to which another node is directly connected versus in prototype two. We also observed a corresponding decrease in the percentage of nodes that were granted sensitive data access for the same. We hypothesize that through the feedback mechanism of side payments, the nodes in the network are in effect learning more about the other nodes in the graph and in turn act to reduce their own vulnerability by revoking sensitive data access to a greater number of nodes, going beyond the requirements of coalition membership to do their own vulnerability analysis. In studying the effect of changing side payments' effectiveness, we would expect to see similar improvement as in both versions of prototype three. However, because of the observed decrease in coalition members and nodes forming a local network for prototype three version two, we hypothesize there may be a point at which the network would degenerate if side payments' effectiveness was sufficiently increased. Such a threshold would be detrimental to the network. As such, we believe there is a corresponding maximum security level that is detrimental because of unrealistic security constraints; this hypothesis is further supported by the evidence of the Kerberos prototypes. These hypotheses should be explored further in future work.

Prototype two was consistent with regard to security regardless of network size; prototype three, which implemented side payments, was affected by network size with larger networks generally reaching higher security. This difference would make sense in the context of quantitative effect of side payments on network security, except we observed that this was not always the case: the lowest security was observed for the middle-sized network made up of 43 nodes. Furthermore, increasing side payment effectiveness caused a tie for networks made up of 60 and 80 nodes, which may indicate that there is an ideal network size for our algorithm, or possibly the existence of an ideal size for any network. Still, the increased security for the network via side payments demonstrates that side payments improve network security overall for pure strategies.

In prototype four which used mixed strategies, however, we observed a different phenomenon with respect to side payments. Increase in side payment effectiveness had no observed effect on convergence time in the way that it did for prototype three, indicating that the effect is nullified by the increased number of interactions and possibly lost side payments brought on by mixed strategies. In addition, security for each network in prototype four approached its maximum theoretical average security, but only did so after a substantially larger number of iterations when compared to pure strategy prototypes. While it is possible such manifestation may be due to some sort of error in our prototype implementation, we believe the following three theories most likely. First, it is possible that the improved security is caused simply by the added interactions between the nodes. Or, the improved security may be resulting from an attempt to recover from mistakes made by playing mixed strategies. Furthermore, it is also possible that the combination of extended interactions with side payments may be creating a mechanism for nodes to learn from one another via the side payments, passing on information about other nodes in the network. While such occurrence was not observed for pure strategies, it is possible the nodes may need an extended number of interactions to learn from one another, much like a neural network. This hypothesis will need to be explored in future work.

We also observed prototype four had a spike in the percentage of nodes making side payments for networks at approximately 45 iterations, but only for a network with maximum security equal to four levels above the heterogeneous minimum security. We repeated our tests to verify this phenomenon. In this network, many nodes have a maximum security of 5 out of 10, and the spike in side payments occurred at slightly less than halfway through convergence to stable network security. Whether there is a correlation between convergence, maximum security, and side payment interaction has yet to be determined. Its repeatability combined with the fact that it is only observed for prototype four, implementing mixed strategies, leads us to conclude that it is related to the mixed strategies in combination with maximum average security for the network; this phenomenon merits further study in future endeavors.

We compared our work with the Kerberos security system to better measure the results generated by our network security algorithm. We measured the effectiveness of Kerberos and our own algorithm with the security metrics we developed. Metrics allowed us to verify and compare the security optimization generated by each very different approach in a manner that allowed us to substantiate the results. In our prototypes' experimental results, we found that while Kerberos did very well when the network was able to reach high levels of security, converging faster and reaching higher overall average network security levels than any of our own solutions, it excluded some nodes from the network because of the constraints imposed by the Kerberos server and system security requirements. In other cases where an increasing number of nodes forming the network were unable to reach high levels of security, Kerberos excluded 80-90% of the nodes from the network due to security constraints, and consequently had much lower average network security versus our own. Our approach always included nodes regardless of their maximum achievable security, allowing all nodes to join the network and receive improved security not to mention preventing any network from splitting apart. In this respect Kerberos was a failure, as the definition and purpose of a network is to connect computers so that they may share resources, communicate, and receive security benefit from one another. While it is possible the security constraints in our Kerberos prototype were higher than needed, our goal in designing the simulation was to stay as close as possible to the Kerberos system and avoid going beyond it to some sort of relaxed Kerberos that stretches the requirements to the point of creating a new security system. We felt our Kerberos model was fairly representative of the actual Kerberos system. Kerberos lacks the ability to restrict access to sensitive or insensitive data. Once the node in the Kerberos network accepts tickets it cannot reject access to specific items, such as allowing access to sensitive or insensitive data that were granted by the Kerberos server.

The experimental results of our own prototypes versus Kerberos matched our analysis. Prototypes implementing our algorithm were consistent with the mathematical models and proofs of our work, in particular regarding weighted potential games and Pareto-optimality. Our results exemplified a game possessing a Pareto optimal weighted potential game equilibrium, which eliminates any sub-optimal equilibrium. Contrast these results with those of Kerberos, wherein there were always some nodes unable to join the network; this is consistent with our mathematical proofs that Kerberos is neither a potential game nor is it Pareto optimal.

For prototype two and prototype three versions 1 and 2, among the networks with highest security was observed an inverse relationship between network security and the number of nodes forming a coalition and granted sensitive data access. Both prototype three versions 1 and 2 showed that these same networks with highest security had smallest local neighborhood count, but this was not consistent for prototype two. We can conclude that side payments affect the security and can reduce the number of neighbors, thereby reducing the number of connections in the network overall. Based upon the results of prototype three version 1 and 2, we can also conclude that side payments affect coalition size, as well as number of nodes granted sensitive data access.

CHAPTER VIII

CONCLUSION

We considered the problem of maximizing security for an entire network when it is made up of heterogeneous computers or devices that act autonomously, and attempted to solve this problem by our game theoretic solution with a potential function. We chose game theory because it can solve problems that would otherwise be difficult using traditional AI approaches, as game theory has the advantage that once the problem and corresponding game are properly specified, the players of the game act on their own according to the game specifications, resulting in a solution to the problem. Furthermore, we identified a subset of constraint satisfaction problems that can be used with a subset of games to solve our security problem, and laid out the game theoretic approaches to solve the problem. We determined that the subset of games known as noncooperative weighted potential games fit the subset of constraint satisfaction problems possessing an objective function. Our work differed from previous approaches to constraint satisfaction problems and games by taking into account the requirements of optimizing overall security for the entire network, and using a game to solve the problem itself. We also devised a new system for measuring network security using security metrics, allowing for a quantitative rather than qualitative measurement of security; we believe metrics are essential in improving the definition of security itself. We were able to introduce security metrics to our problem by defining the problem itself as a game possessing a system of variables and constraints. Using our metrics, we developed several equations defining network security, which enabled us to validate our work and create an algorithm that integrates attack tree analysis for the entire network at each step of network formation. Our work contrasts with previous works that focus on analyzing only network components, or perform late evaluation of security.

The results of our prototypes demonstrate the effectiveness of our game theoretic security algorithm with regard to optimizing overall network security in a heterogeneous environment. Our results also empirically demonstrate the validity of our theorems and proofs, showing the applicability of our game theoretic algorithm to improving the security of the heterogeneous computer network of today, with computers acting independently without a central coordinator. As opposed to Kerberos, our work can be generalized to optimize security for almost any network or system because it can be applied to a network made up of devices that possess any degree of heterogeneity. Determining whether there may be an optimum configuration or percentage of nodes making side payments that leads to faster convergence and greater utility bears further study.

Our work was able to show improved security and efficiency through the introduction of coalition formation with partial preferences for our game. By forming coalitions with partial preferences, we were able to integrate coalition formation into the optimization process during the game itself. Previous approaches to coalition formation depended heavily on knowing complete preferences prior to coalition formation and were not able to be directly integrated into the optimization process itself, leading to a system in a sub-optimal state. However, it may be possible to alter the definition of equilibrium itself so that coalition formation could be postponed until after this new equilibrium state had been reached, and subsequent formation of coalitions would not change the optimality of the solution for the system. In this, if we were to redefine equilibrium as imposing a total order instead of a partial order on the actions, so that every action in the set of actions is comparable instead of reflexive, anti-symmetric, and transitive, then it may be possible to postpone coalition formation until stabilization. This hypothesis should be explored in future work.

Granted that Pareto optimal equilibrium assumes that every element in an action profile is comparable, meaning there exists a total ordering on its elements, it does not necessarily mean that all elements between different action profiles are comparable. To address this issue, it may be possible to redefine the Pareto optimal solution using a definition whereby it is not just a total order within an action profile, but also a total order between all elements of all action profiles. We know that a potential function, by definition, reduces the number of all possible actions and action profiles to one action profile in equilibrium; within that action profile there must be a total order. Thus all nodes maximizing the potential should also have a total order on the actions. However, achieving this definition may be difficult, since we understand what we are defining is a solution that is already optimized, and we would be working backwards from an already optimal solution to a suboptimal problem. We are thus uncertain whether or not this is a paradox, as it is likely that the action profiles need to be comparable to one another or proven to be comparable, so that each element of every action profile is comparable, giving a total order between the action profiles. Exploring this possibility and its implication of equality or improvement will need to be done in the future; it could speed up the network optimization process as well as possibly strengthen the definition of an optimized network.

Overall, our solution improved overall security for any measured degree of heterogeneity in a network. Prototypes empirically demonstrated our mathematical models for Pareto-optimality. The introduction of coalitions and side payments to form connections showed approximately 10% security improvement, but somewhat slower convergence to stable network security. Our prototype also demonstrated that our approach was largely unaffected by variations in network size. Computers with greater maximum security granted fewer nodes access to sensitive data once security stabilization was reached. A correlation existed between networks that maximized the use of side payments with those that performed best in terms of security. Convergence time was observed to decrease as much as 50% when side payment effectiveness was improved in terms of security. With mixed strategies, increase in side payment effectiveness had no observed effect on convergence time. Security for mixed strategy prototypes approached the theoretical maximum after a substantially larger number of iterations when compared to pure strategy prototypes.

Through the solutions presented in our work, we were able to model and successfully test our network security optimization technique for a network formed of heterogeneous computers. Furthermore, by comparing its results to the known Kerberos algorithm, we were able to demonstrate the usefulness of our approach. We hope that our work can be used in the future to improve network security, and that our work in quantitative security measurement will be applied to other areas for a paradigm shift in the definition of security from the qualitative to the quantitative.

8.1 Future work

Because of the observed decrease in coalition members and nodes forming a local network for prototype three version two, we hypothesize there may be a point at which the network would degenerate if side payments' effectiveness was sufficiently increased. Such a threshold would be detrimental to the network. We also believe there is the possibility of hardware and software combinations that are harmful to a heterogeneous network because they can lead to unrealistic security constraints compared to the rest of the nodes in the network. This hypothesis is supported by the evidence of the Kerberos prototypes' performance on a network with low security. These hypotheses should be explored further in future work, as they may characterize limitations of our game theoretic algorithm. We also do not know why security for networks playing mixed strategies was so high versus corresponding networks playing pure strategies, or why it stabilized after a substantially large number of iterations. We believe future work should be done to determine its cause. In addition, we would like to consider the scenario where some nodes play mixed and some play pure strategies for the purpose of evaluating its effect on network security.

We would also like to evaluate the effect on convergence time and overall security using a hypothesis that may enable coalition formation to be postponed until after equilibrium has been reached. In this, it may be possible to alter the definition of equilibrium itself so that it imposes a total order instead of a partial order on the actions, so that every action in the set of actions is comparable instead of reflexive, anti-symmetric, and transitive. Future work should also include developing games with asymmetric information. Games with asymmetric information are more complicated than games with symmetric information, but as part of implementing such a game the issue of authentication and false impersonation can be explored as it fits the description of asymmetric information.

Furthermore, we would like to continue strengthening the relationship between our architecture and its application to a hardware and software security scheme in an actual network implementation. In particular, we would like to further develop our architecture to implement it on the TCP reference model. This work would involve specifying the relationship of our game theoretic algorithm with the lower layers of the TCP reference model, as well as specifying greater detail regarding packets and protocols. Furthermore, we would like to extend our work to optimize new nodes entering the network. All of these issues to be explored in future work can contribute to improving the means of optimizing network security through game theory and security metrics.

REFERENCES

[1] R. Johari, S. Mannor, and J. Tsitsiklis, “A contract-based model for directed network formation,” Games and Economic Behavior 56, pp. 201–224, 2006.

[2] D. Monderer and L.S. Shapley. “Potential games,” Games and Economic Behavior 14, pp.124-143, 1996.

[3] V. Bala and S. Goyal, “A noncooperative model of network formation,” Econometrica, vol. 68, no. 5, pp. 1181-1229, 2000.

[4] R.S. Komali, A.B. MacKenzie, and R.P. Gilles, “Effect of selfish node behavior on efficient topology design,” IEEE Trans. Mobile Computing, vol. 7, no. 9, pp.1057-1070, 2008.

[5] L.S. Shapley, “Cardinal utility from intensity comparisons,” RAND report R-1683-NSF, Santa Monica, CA, 1975.

[6] M. Baucells and L.S. Shapley, “Multiperson utility,” Games and Economic Behavior, vol. 62, pp. 329–347, 2008.

[7] V. Soni, S. Singh, and M. Wellman, “Constraint satisfaction algorithms for graphical games,” Sixth Int. Joint Conf. Autonomous Agents and Multi-Agent Systems, article no. 67, 2007.

[8] E. Rasmusen, Games and Information, Blackwell Publishing, 2007.

[9] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing, “Automated generation and analysis of attack graphs,” Proc. IEEE Symp. Security and Privacy, pp. 273- 284, 2002.

[10] I.A. Almerhag and M.E. Woodward, “Security as a quality-of-service routing problem,” Proc. ACM Conf. Emerging network experiment and technology, pp. 222-223, 2005.

[11] T. Heikkinen, “Distributed scheduling and dynamic pricing in a communication network,” Wireless Networks, vol. 10, pp. 233–244, 2004.

[12] D. Kincaid and W. Cheney, Numerical Analysis: Mathematics of Scientific Computing, 3rd ed., Brooks/Cole, 2001.

[13] D. Bauso, L. Giarre, and R. Pesenti, "Consensus in noncooperative dynamic games: a multiretailer inventory application," IEEE Trans. Automatic Control, vol. 53, no. 4, pp. 998-1003, 2008.

[14] R.W. Rosenthal, "A class of games possessing pure-strategy Nash equilibria," Int. J. Game Theory, vol. 2, pp. 65–67, 1973.

[15] T.H. Hai and E.-N Huh, "Detecting selective forwarding attacks in wireless sensor networks using two-hops neighbor knowledge," Seventh IEEE Int. Symp. Network Computing and Applications, pp. 325–331, 2008.

[16] D. Vickrey and D. Koller, "Multi-agent algorithms for solving graphical games," Eighteenth Natl. Conf. Artificial Intelligence, American Association for Artificial Intelligence, pp. 345–351, 2002.

[17] J. Von Neumann and O. Morgenstern, The Theory of Games and Economic Behavior, Wiley, 1944.

[18] J. Nash, "The bargaining problem," Econometrica, vol. 18, no. 2, pp.155-162, 1950.

[19] J. Nash, "Equilibrium points in n-person games," Proc. National Academy Sciences, vol. 36, no. 1, pp. 48-49, 1950.

[20] J. Nash, "Non-cooperative games," Annals of Mathematics, vol. 54, no. 2, pp. 286-295, 1951.

[21] L. Shapley, "Open questions," Report of an Informal Conference on the Theory of n-Person Games, Princeton Mathematics mimeo, p. 15, 1953.

[22] L. Shapley, "A value for n-person games," Kuhn & Tucker, pp. 307-317, 1953.

[23] S.C. Payne. "A Guide to Security Metrics," SANS Institute InfoSec Reading Room, 2006. [Online]. Available: http://www.sans.org/reading_room/whitepapers/auditing/a_guide_to_security_metrics_55?show=55.php&cat=auditing

[24] A. Tucker, "A two-person dilemma," Stanford University mimeo, May 1950.

[25] L. Wang, A. Singhal, and S. Jajodia, "Toward measuring security using attack graphs," Proc. ACM QoP, pp. 49-54, 2007.

[26] A. Agah, S. K. Das and K. Basu, "A game theory based approach for security in wireless sensor networks," IEEE Int. Conf. Performance, Computing, and Communications, pp. 259 - 263, 2004.

[27] A. Agah, K. Basu, and S. K. Das, "Security Enforcement in Wireless Sensor Networks using Non-Cooperative Game Theory Framework," Pervasive and Mobile Computing Journal, vol. 2, no. 2, pp.137-158, 2006.

[28] W. Sun, X. Kong, et. al., "Information security game analysis with penalty parameter," Proc. IEEE Int. Symp. Electronic Commerce and Security, pp. 453-456, 2008.

[29] M. Demirbas, A. Arora, and M. Gouda, "A pursuer-evader game for sensor networks," Proc. Sixth Symp. Self-Stabilizing Systems (SSS'03), Springer, pp. 1-16, 2003.

[30] R.S. Komali and A.B. MacKenzie, "Distributed Topology Control in Ad-Hoc Networks: A Game Theoretic Perspective," Proc. Third IEEE Consumer Comm. and Networking Conf. (CCNC '06), vol. 1, pp. 563-568, 2006.

[31] C. Kaufman, R. Perlman, M. Speciner, Network Security: Private Communication in a Public World, Prentice-Hall, 1995.

[32] D. Fudenberg and J. Tirole, Game Theory, MIT Press, 1991.

[33] S.E. Stumpf, Socrates to Sartre: A History of Philosophy, 5th ed, McGraw-Hill, 1993.

[34] "Hackers of U.S. electrical grid left behind 'sleeper' software programs," Homeland Security Newswire, 2009. [Online]. Available: http://homelandsecuritynewswire.com/hackers-us-electrical-grid-left-behind-sleeper-software-programs

[35] "Kerberos: The Network Authentication Protocol," MIT press, 2010. [Online]. Available: http://web.mit.edu/kerberos/

[36] S. Zrelli and Y. Shinoda, "Specifying Kerberos over EAP: Towards an integrated network access and Kerberos single sign-on process," 21st Int. Conf. Advanced Networking and Applications, pp. 490-497, 2007.

[37] C.-K. Han, H.-K. Choi, "An Adoption of Kerberos to 3G Network for Mutual Authentication: Challenges and Evaluations," Int. Symp. Performance Evaluation of Computer and Telecommunication Systems (SPECTS), pp. 448-455, 2008.

[38] A. Prasad S., K. J. Park, et. al., "Kerberos Based Authentication Protocol with Improved Identity Protection in 3G Network," Pacific-Asia Conference on Circuits, Communications and Systems, pp. 771-774, 2009.

[39] F. Chen, J.-S. Su, “A Flexible Approach to Measuring Network Security Using Attack Graphs,” IEEE Int. Symp. Electronic Commerce and Security, pp. 426-431, 2008.

[40] J. Homer, A. Varikuti, et. al., “Improving Attack Graph Visualization through Data Reduction and Attack Grouping,” 5th Int. Workshop Visualization for Cyber Security (VizSEC 2008), pp. 68-79, 2008.

[41] B. W. Kernighan and D. M. Ritchie, The C Programming Language, 2nd ed, Prentice Hall, 1988.

[42] H.M. Deitel and P.J. Deitel, C++ How to Program, Prentice Hall, 1994.

[43] A. Agah, S. K. Das and K. Basu, “Enforcing security for prevention of DoS attack in wireless sensor networks using economical modeling ,” 2nd IEEE Int. Conf. Mobile Ad-Hoc and Sensor Systems (MASS), pp. 137-158, 2005.

[44] S. Jha , O. Sheyner , J. Wing, “Two Formal Analyses of Attack Graphs,” Proc. 15th Computer Security Foundation Workshop, pp. 49-63, 2002.

[45] R. Johnsonbaugh, Discrete Mathematics, 7th ed, Prentice Hall, 2009.

[46] A. Feldmann, "Netdb: IP Network Configuration Debugger/Database," tech. rep., AT&T Research, July 1999.

[47] "Rand - C++ Reference," 2010. [Online]. Available: http://www.cplusplus.com/reference/clibrary/cstdlib/rand/

[48] “The C++ standard library,” 2010. [Online]. Available: http://gcc.gnu.org/onlinedocs/libstdc++/libstdc++-html-USERS-3.3/cstdlib-source.html

VITA

Patrick D. Harrington

Candidate for the Degree of

Doctor of Philosophy

Thesis: USING NONCOOPERATIVE POTENTIAL GAMES TO IMPROVE NETWORK SECURITY

Major Field: Computer Science

Biographical:

Education:

Completed the requirements for the Doctor of Philosophy in Computer Science at Oklahoma State University, Stillwater, Oklahoma in July, 2010.

Completed the requirements for the Master of Science in Computer Science at the University of Tulsa, Tulsa, OK, 2002.

Completed the requirements for the Bachelor of Science in Computer Science at Northeastern State University, Tahlequah, OK, 1999.

Completed the requirements for the Bachelor of Arts in English at Oklahoma Baptist University, Shawnee, OK, 1996.

Experience: Instructor, Northeastern State University, 2001- present

Professional Memberships:

Association for Computing Machinery
Kappa Mu Epsilon National Mathematics Honor Society
Rho Theta Sigma Academic Honorary Society
Sigma Tau Delta International English Honor Society
Phi Eta Sigma National Honor Society

Name: Patrick D. Harrington Date of Degree: July, 2010

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: USING NONCOOPERATIVE POTENTIAL GAMES TO IMPROVE NETWORK SECURITY

Pages in Study: 240 Candidate for the Degree of Doctor of Philosophy

Major Field: Computer Science

Scope and Method of Study:

Our work puts forth a game theoretic global security mechanism to optimize security in a large heterogeneous network consisting of autonomous devices. Our work is applicable to a network that includes various computing devices such as PCs, cell phones, sensors, and control systems. Constraint satisfaction is used to fulfill the requirements of the differing computers in the network. Security metrics are used to quantify network security in a meaningful way. Attack tree analysis of the quantified security measurements is performed for decision-making to maximize security by altering links that form the network. Coalitions of the computers forming the network are used to improve efficiency, as well as give a broader and greater overall security than would be possible in their absence. Side payments are used to induce a computer to move beyond its selfish motivations to benefit another computer. In keeping with noncooperative game rules, costs to form links are imposed only on the initiator of the link.

Findings and Conclusions:

Our solution is presented as a noncooperative weighted potential game using pure or mixed strategies. Findings showed that our game theory model of optimization improved overall security for heterogeneous networks, and demonstrated the viability of quantitative security analysis using metrics. Our game theory model of side payments and coalitions increased security by 10% in our experiments. Side payments more than doubled convergence time to optimal network security; however, side payments that were twice as effective in terms of security improvement reduced convergence time by 50%. Our game theory model of mixed strategies showed longer convergence time but overall improved security versus our pure strategy model. Findings also demonstrated that our game theory model of optimization improved security for networks of varying size. Through these solutions, our work presents a novel optimization technique that improves overall security for a heterogeneous network.

ADVISER’S APPROVAL: Dr. Johnson Thomas