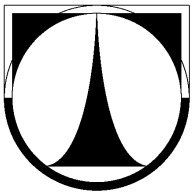
TECHNICAL UNIVERSITY OF LIBEREC

Faculty of Mechatronics and Interdisciplinary Engineering Studies



Self-organizing and Self-monitoring Security Model for Dynamic Distributed Environments

2008 ROMAN ŠPÁNEK

Self-organizing and Self-monitoring Security Model for Dynamic Distributed Environments

PhD. Thesis

Ing. Roman Špánek

Technical University of Liberec

Faculty of Mechatronics and Interdisciplinary Engineering Studies

February 2008

Name of the Thesis: Self-organizing and Self-monitoring Security Model for Dynamic Distributed Environments

Doctoral Candidate: Ing. Roman Špánek
Study Programme: 2612V Electrical Engineering and Computer Science
Study area: 2612V045 Technical Cybernetics
Department: Institute of Mechatronics and
Theoretical Informatics
Faculty of Mechatronics and Interdisciplinary
Engineering Studies
Technical University of Liberec

Supervisor: Ing. Július Štuller, CSc.(FM TUL, CS CAS CZ)

Author's Statement

I honestly state that I have worked out my PhD. thesis on my own using references listed in the thesis and consultations with my supervisor.

Date:

Signature:

Foreword

The thesis was proposed during my PhD. studies at the Technical University of Liberec, Faculty of Mechatronics and Interdisciplinary Engineering Studies. The PhD. studies started in 2003 after successful graduation at master studies at the Technical University of Liberec.

At the beginning the PhD. studies were oriented towards the mobile databases area in general. As my studies advanced I realized that my preferences oriented on security issues presented in the mobile databases. Majority of my enthusiasm into the security issues came from the fact that security was, and with no doubt still is, one of the most important aspect influencing the usability aspects. The amount of open questions together with severe limitations and constraints put on the security models in the field has been encouraging me in the most positive way.

From the very beginning of my PhD. studies I have been submitting papers to many international as well as local conferences. Presenting the ideas to experts who provided me with many valuable comments, remarks and suggestions has been pushing my work forward. The conferences also provided me a great deal of opportunities to establish many very nice relationships with experts in my research topic, as well as with other PhD. students from many countries. I hope that these experiences have positively influenced the thesis.

Acknowledgements

I would like to start by thanking my supervisor Július Štuller who has been helping me in my research activities for my PhD. studies and who provided me with uncountable contacts, funds and moral encouragement.

I would like to thank prof. Miroslav Tůma from Institute of Computer Science Academy of Sciences of the Czech Republic who has helped me a lot with the hypergraph model.

A special thanks belongs to my wife Jana who has never failed encouraging and supporting me, some times it must have been very hard for her though.

In Liberec, February 2008.

Anotace

Samostatně pracující bezpečnostní model pro dynamická distribuovaná prostředí

Ing. Roman Špánek

Disertační práce obsahuje návrh nového bezpečnostního modelu pro dynamické a distribuované systémy, ve kterých se klasické bezpečnostní mechanismy používané v centralizovaných systémech potýkají s celou řadou problémů. Jedním z úspěšných modelů pro zabezpečení distribuovaných systémů jsou tzv. reputační systémy, které umožňují budování důvěry mezi entitami. Následné řízení přístupu k datům či službám je pak založeno na míře důvěry mezi entitami. Pokud je důvěra dostatečně vysoká, je přístup k datům či službám možné povolit nebo doporučit.
Disertační práce je doplněna o návrh a experimentální ověření bezpečnostního modelu navrženého speciálně pro dynamické distribuované systémy s velkým počtem uživatelů. V dynamických systémech může docházet k častým změnám v úrovni důvěry mezi entitami, entity mohou do systému přicházet nebo naopak ze systému odcházet. Takovéto dynamické systémy představují problémy i pro tradiční reputační systémy. Představovaný bezpečnostní systém se proto liší od známých reputačních systémů v chápání důvěry mezi entitami. V našem modelu je důvěra společná skupině entit a ne pouze konkrétní dvojici entit. Takový model poskytuje lepší podporu pro budování důvěry především v dynamických systémech s velkým počtem entit, kde nelze využívat pouze osobní vztahy mezi entitami.

Vzhledem k odlišnému chápání důvěry používá námi navrhovaný bezpečnostní model pro popis důvěry mezi entitami matematického aparátu hypergrafů. Bezpečností model je tvořen námi navrženými a ověřenými algoritmy pro transformaci obecného grafového vstupu do modelu hypergrafů, algoritmu pro správu dynamických aspektů systému a také bezpečnostního podsystému.

Navrhovaný bezpečnostní model je pilotně implementován v experimentální implementaci SecGrid, která je také použita pro experimentální ověření všech částí modelu. Experimenty ukazují, že námi navrhovaný model překonává tradiční reputační systémy v dynamických systémech s velkým počtem entit.

Klíčová slova: reputační systémy, bezpečnost, důvěra, distribuované systémy

Annotation

Self-organizing and Self-monitoring Security Model for Dynamic Distributed Environments

Roman Špánek

The thesis deals with security hazards in distributed environments where traditional centralized approaches are only of limited serviceability. One of the very successful model for treating security and access management in distributed systems are so called reputation systems. The main goal of the reputation systems is to provide entities in the environment with mechanisms for inferring and building trust consequently used for access control. If the trust between two entities is high enough, transactions are likely to be allowed.

The thesis proposes a new security model with trust management system for dynamic and distributed environments with huge number of entities. In dynamic systems new entities or relationships are likely to emerge or existing entities or relationships may often disappear. Such dynamics pose severe problems even for traditional reputation systems. Therefore our approach differs from the traditional ones in the way adopted for establishment and management of trust between entities – in our point of view trust is not assigned to particular relationships but the trust is common for a group of entities. In this way, our proposal significantly enhances ability to infer trust between entities with no previous personal experiences with each other or in environments with huge number of entities.

For the proposal differs in understanding of trust, it uses a hypergraph model for representation of system of entities. The security model proposed in the thesis contains two algorithms for transformation of a general input graph structure into hypergraph model, an algorithm treating dynamics of the distributed environment and a security subsystem.

Our experimental implementation SecGrid utilizes proposed algorithms and it is used for experimental verification of the security models. The experiments investigate ability of the transformation algorithms; in details the dynamic part of our proposal together with the security subsystem proposed specially for the hypergraph model. Experiments show that our model overcame the traditional graph model in many ways especially in dynamic environments with huge amount of entities.

Key Words: trust management systems, security, trust, distributed systems

Contents

List of Symbols and Shorts Used in the Thesis

α	Condition triggering interactions
$\Delta\Phi$	Gain of percentage of good users in group
$\Delta\|N\|$	Amount of groups created by the security sub-system
ϵ	Parameter of the SD Algorithm influencing joining
λ	Parameter of the SD Algorithm influencing splitting
Ω	Starting amount of groups for the SD Algorithm
Φ	Percentage of good users in a group
Φ_1	Starting average of percentage of good users in groups created by the SD Algorithm
Φ_2	Final average of percentage of good users in groups
Ψ	Number of good users in system in percents
$\|N\|_1$	Amount of groups created by the SD Algorithm
$\|N\|_2$	Amount of groups created by the SD Algorithm plus groups created by the security sub-system
$1G$	First Generation of Cellular Networks
$2.5G$	Second and a Half Generation of Cellular Networks
$2G$	Second Generation of Cellular Networks
$3G$	Third Generation of Cellular Networks
$4G$	Forth Generation of Cellular Networks
$a_1, a_2, \ldots, a_n$	Parameters of the model
API	Application Programming Interface
BT	Bluetooth
C^0	Initial state of the model
C^k	State of the model after k interactions
$CLDC$	Connected Limited Device Configuration
DVO	Dynamic Virtual Organization
$f(I)$	Function transforming the input structure into the starting state of the model
$G2HAlgorithm$	Graph to hypergraph Algorithm
$GEAR$	Geographical Energy Aware Routing
$GPSR$	Greedy Perimeter Stateless Routing
GSM	Global System for Mobile Communications
GUI	Graphic User Interface
I	Input structure describing existing relationships between users
$IrDA$	Infrared Data Association
$J2ME$	Java Micro Edition
$JSR-82$	Java APIs for Bluetooth
$LDAP$	Lightweight Directory Access Protocol
$LEACH$	Low-Energy Adaptive Clustering Hierarchy

M	Proposed Security Model
$M_a(t)$	Model of behavior of entity a
MPI	Message Passing Interface
$P2P$	Peer-To-Peer
PDA	Personal Digital Assistant
$PEGASIS$	Power-Efficient GAthering in Sensor Information Systems
$PRIMA$	System for Privilege Management and Authorization
RMS	Record Management System
$SDAlgorithm$	Structure Dynamic Algorithm
SDK	Software Development Kit
$SPIN$	Sensor Protocols for Information via Negotiation
$T_{ab}(t)$	Level of trust of entity a to entity b
$T_{D_b}(t)$	Reputation of entity b given by the others
$T_{P_{ab}}(t)$	Level of trust derived from the history of interaction between entities a and b
$TEEN$	Threshold sensitive Energy Efficient sensor Network protocol
$TTDD$	Two-Tier Data Dissemination
UML	Unified Modeling Language
VO	Virtual Organization
$VOMS$	Virtual Organization Management Service
$X.509$	Internet X.509 Public Key Infrastructure
$XACML$	Extensible Access Control Markup Language
XML	Extensible Markup Language

Chapter 1

Introduction

It is thought that humans evolution was started at the precise moment when the first primogenitor used its hands to make a simple work. Even if it might not be the true, the truth is that our primogenitor used to live in groups. These groups were after some time enlarged and humans started to live in crowds. These crowds were nothing but simple societies, the predecessors of current complicated society spreading around the world with many members and very complicated relationships. Although the evolution from crowds to nowadays societies was a long and sometimes a painful process, at least one thing has reminded in the limelight – the communication. Were it not for the communication, it would have been very hard even unlikely to have achieved the progress in society. It is not surprising then that many great inventions have been in the field of the communication. The list might be started with the typography going through the Bell's telegraph and telephone reaching the contemporary hi-tech wireless communication devices and the Internet. The great success of the Internet, mobile phones or networking just naturally testifies our assertion on the importance of the communication.

Taking aside for a moment the older inventions, let take a closer look at recent progress in communication. At the beginning of the 20th century people would not even think of devices that appeared several decades later. The huge progress was started by research projects in the field of micro-technologies. At first simple and huge computers were introduced with just few capabilities compared to the computers of our days. As time moved on, the progress was accelerating and micro-computers became smaller and more capable, leading to a first personal computer appeared in the 80s.

But the microcomputers were not the only research area of great inventions and improvements. Even computers despite their computational power and abilities were limited in the same way as would be any isolated entity. Therefore the first computer networks came into the world. Although these first networks were very limited connecting just few computers in dozens meters radius, it was the beginning of a rapid development. This development

has brought the Internet as one of its most considerable results.

When the Internet became popular and personal computers were evolving following the Moor's law, communication was about to change medium and devices. It was a moment when micro-technologies together with wireless networking technologies enabled a new type of networks and devices – the mobile communication. At the beginning the mobile networks were simple, slow and prone to many errors. In addition to that mobile devices were closer to weight than to pocket devices for everyday use. Nevertheless the unceasing progress and interest of the users led to currently available devices such as tiny mobile phones, capable PDA (Personal Digital Assistance) or laptops. Moreover, the capabilities of the mobile networks have been under continual improvement.

However, all the progress and the great success of the Internet and mobile phones have also faced some severe problems. One of the most severe issues is the task of treating the security.

Even thought the security is a problem being common to many research and application areas, it becomes very important in the case of information exchange systems. This is due to the fact that current information exchange includes very often private information or information with a significant price. Albeit the security task is sometimes considered as the one coping with the encryption only, it should be treated differently, namely as consisting of several subtasks.

In the following basic scenario we would like to make clear the fact that security is not only a cryptography. Assume two distinct entities A and B. The entity B requires very private information from the entity A. Such information exchange requires at least the following:

- data formats for communication
- protocols for transmission of data
- encryption for securing transmission against the threat of tap.

Even thought this list might be seen as a complete one, in the following a missing element will be identified. Let the mentioned tools are available and secure enough[1]. At this point entities A and B are able to exchange the information. Nevertheless A would probably mind sending the private information to anybody. More probably the exchange will be restricted so that private information will be available to the reliable entities only and non-private information will be distributed with no or little restrictions.
At this moment we can step back to the list given above and address the missing element – the *trust*. Since trust is common for many research and development areas many trust definitions have been proposed till now.

[1]Note that we overlook for the moment the fact that the last assumption is clearly odd as no encryption can guarantee total invulnerability against unauthorized decryption.

From these definitions we have adopted the one by Deutsch [1] leading to:

Definition 1

a) An individual is confronted with an ambiguous path, a path that can lead to an event perceived to be beneficial or to an event perceived to be harmful;
b) He perceives that the occurrence of these events is contingent on the behavior of another person; and
(c) He perceives the strength of a harmful event to be greater than the strength of a beneficial event.

If he chooses to take an ambiguous path with such properties, he makes a trusting choice; else he makes a distrustful choice.

Therefore the security task can be viewed as consisting of the two main subtasks:

- **encryption** - strong cryptography algorithms for securing communication against the *threat of tap*, *man-in-the-middle* attack, etc.
- **trust** - trust between engaged entities simplifying the process of making the decision whether to accept or reject a request.

The important point is that the mentioned subtasks should not be treated separately but rather considered as the cryptography providing formats, methods and algorithms for the level of trust.

It is clear that research is very needed on both subtasks and should be driven by common aims.

We have discussed above that an interaction or a communication between entities with very limited knowledge about themselves can take place only if trust between parties is high enough. Nevertheless trust is not a static phenomena; humans are used to change their relationships on the fly as put in harm's way or security threats during their every day lives. Thence models coping with trust as static property fail to be appropriate in human driven communication.Thus, a dynamic model of trust is needed as new relationships may emerge, existing relationship may disappear or level of trust may change in time.

1.1 Motivation

The following extended basic scenario expresses the main motivation of this thesis:

Assume entity (let entity to be a user in the following) A being asked for personal information by user B. User A can reject the request, or accept it.

However, if user A cannot find out to whom the data will be sent (who user B is), it should rather reject the request. This, however, will lead to the situation when all similar requests are rejected, and consequently no communication is allowed. On the contrary, if user A is able to find out who user B is, he will make the decision whether share information or not much more easily. Simply, if user B is a reliable person, the access is granted, otherwise rejected.

The fact stressed by this scenario is that users should bound access to private information according to the level of trust between them [1]. Therefore the main aim of the thesis is to propose a stable system preserving overall security in a distributed environment by supporting shared knowledge and experience in groups of users. The important and novel point is the treatment of trust for the whole groups of users, which differ from the current systems where trust is treated for particular users. Furthermore, such system should be designed to be able to handle a dynamic evolution of groups of users. In addition to that the system should have distributed implementation with low time complexities.

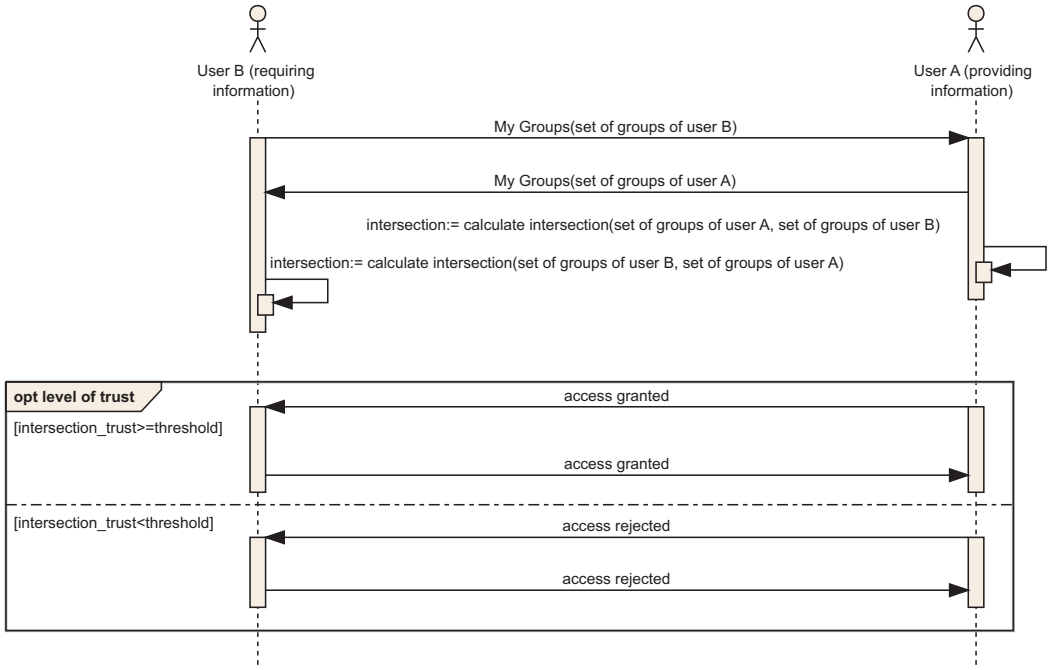


Figure 1.1.1: The Extended Motivation Scenario (describing private information exchange between two users)

In Figure 1.1.1 the extended scenario of private information exchange given above is shown in the UML interaction diagram format. This corresponds precisely to the main motivation and the aim of the thesis.

[1]Note that terms "level of trust" and "trust" will be used in the thesis interchangeably.

The figure shows two interacting users, user A and user B so that user B is the consumer of the information provided by user A. At first users exchange sets of groups they are members of. Then both retrieve their own group memberships and compute intersections – groups they are both members. If both users find out that they are both members of trusted group(s) the access is granted, rejected otherwise (shown in the figure in the *opt* block). The important point is that users don't use information valid for their particular relationship, but group membership information for trust derivation.

Chapter 2

Related Work

This chapter presents topics related to the thesis theme, the thesis is put into the context and the importance of the topic is discussed.

2.1 Mobile Databases

In [2] authors mention a new trend in the evolution of the Internet according to the recent progress in the mobile computing paradigm. The paper points to the fact that human users on the Internet are becoming a minority compared to the amount of artificial users or services communicating to themselves rather than human users using their computers (with browsers). As the human users are set a bit out, we can no longer assume only a pc-class device running a full browser. Instead of pc-class devices, much smaller (typically hand-held size) and less capable devices are to be expected to access the services. Examples of such devices are cell phones, palm devices, home appliances, and embedded systems (i.e. in cars). Some of these devices may stay stationary, but many of them turn to be mobile. The fact that many current protocols and even architectures (such as server/client) are not sufficient and appropriate under assumption of mobile and less capable devices is obvious.

The mobile databases [3],[4],[5] paradigm corresponds well to the ideas given above. Generally, the mobile database architecture allows users to access and exchange information *anywhere* and at *any time* through pocket-size mobile devices. Even thought the mobile databases have been a hot topic for quite a short time, the achieved progress and its rapidity is indisputable. The mobile networks have evolved from narrow band, error prone and highly unavailable networks to nowadays wide bandwidth networks available round the world.

Most people think of mobile databases as cellular networks, but the mobile databases paradigm may consist of various architectures:

- *cellular networks*
- *multihop wireless networks (ad-hoc networks)*
- *sensor networks*

Although **cellular networks** consist of a few components (such as *Mobile Switching Center*, *Home Location Register*, *Visitor Location Register*) we relax most of them for the sake of simplicity. Instead, for our purposes, cellular networks consist of some *specialized nodes* (*base stations*) and *mobile nodes*. Base stations are equipped with *sufficient capabilities* designed to *coordinate* and *control all transmissions* within their coverage area - a *cell*. They also grant access to wireless channels in response to service requests received from mobile nodes. Mobile nodes, on the contrary, are pocket-size mobile devices intended to be terminals into cellular networks. The recent noticeable trend in cellular networks is transformation from voice intended networks into wide bandwidth data exchange networks (the generations of the networks starting from 1G through 2G, 2.5G, 3G to 4G).

The primary characteristic of the **ad-hoc network** architecture [6],[7],[8],[9] is the *absence* of any *predefined (stationary) infrastructure*. Ad-hoc nodes can communicate directly with the nodes within their transmission range in the *peer-to-peer manner*. Communication to distant nodes is achieved through other nodes in the network in a *multihop fashion*. Thence, each ad-hoc node acts as client as well as a router, storing and forwarding packets on behalf of the other nodes. This results in a generalized wireless network that can be rapidly deployed and dynamically reconfigured to provide *on-demand* networking solutions.

Sensor networks consist of possibly *huge amount of sensors* being generally equipped with a *sensing unit*, a *radio transmitter*, a *processing unit* and a *battery*. The sensing unit provides measurement of the sensor vicinity and transformation of the measurements into electric signal. The measured data are processed by the processing unit and then sent via the radio transmitter to a command center (sink). The sink is responsible for forwarding measured data into common networks. Although sensor networks are in the essence very similar to ad-hoc networks, there exist some limitations tightly connected with sensor networks. Such limitations can be summarized as follows:

- *a centralized addressing scheme cannot be used,*
- *high redundancy* of measured data,
- *severe physical limitations* (e.g. transmission power, on-board energy, processing and storage capacity).

All mentioned limitations have led to the design of specialized routing algorithms:

- *data-centric* (flooding and gossiping [10], Sensor Protocols for Information via Negotiation - SPIN [11], Directed Diffusion [12],[13]),
- *hierarchical* (Low-Energy Adaptive Clustering Hierarchy - LEACH [14], Power-Efficient GAthering in Sensor Information Systems - PEGASIS [15], Threshold sensitive Energy Efficient sensor Network protocol - TEEN [16]),
- *location-based* (Greedy Perimeter Stateless Routing - GPSR [17], Geographical Energy Aware Routing - GEAR [18], Two-Tier Data Dissemination - TTDD [19]).

A deeper survey on routing protocols for sensor networks is provided in [20].

2.2 Peer-To-Peer Networks

Peer-To-Peer (P2P) networking [21],[22] has been given tremendous interest worldwide among either ordinary Internet users as well as networking professionals. The basic definition of P2P networks follows [23]:

Definition 2 *A distributed network architecture may be called a Peer-to-Peer (P2P) network, if the participants share a part of their own hardware resources (processing power, storage capacity, network link capacity, printers,...). These shared resources are necessary to provide the service and content offered by the network (e.g. file sharing or shared workspaces for collaboration). They are accessible by other peers directly, without passing intermediary entities. The participants of such a network are thus resource (service and content) providers as well as resource (service and content) requestors (servent-concept).*

P2P architecture was proposed to overcame several of client/server limitations [22]:

- *Scalability*
- *Single point of failure*
- *Central administration*
- *Unused resources.*

P2P networks, in contrast to server/client architecture, offer a great amount of benefits:

- *Efficient use* of resources
- *Scalability* and *reliability*
- *Consumers* of resources also *donate resources*

- Aggregate resources grow naturally with utilization
- Geographic distribution
- *No single point of failure* and *no central administration*
- Nodes have *self organizing capabilities*
- *Built-in fault tolerance*, *replication*, and *load balancing*.

As a simple example of a P2P network one can think of a home computer network where users configured their computers to enable simple sharing of files, printers and resources. The other examples of P2P architecture are file sharing networks dedicated for simple file sharing across the Internet. Examples of applications allowing such a simple file sharing were/are Napster [24], Kazaa and Kazaa Lite [25] and Gnutella [26].

In spite of the well known problems with *legacy* and problems concerning *privacy* and *security*, P2P networks offer a great amount of possibilities. As every node in the network is responsible for its own security policies and privacy, the *overall security is hard to achieve* by traditional approaches (e.g. passwords).

It is noticeable that P2P architecture is sometimes mentioned as a future architecture of the Internet.

2.3 Grids

The term *Grid* was introduced in the mid 1990s to denote a distributed computing infrastructure proposed for advanced scientific and engineering tasks. The Grids enable sharing resources in multi-institutional Virtual Organizations defined as [27]:

Definition 3 *Virtual Organization (VO) is a temporary or permanent coalition of geographically dispersed individuals, groups, organizational units or entire organizations that pool resources, services and information to achieve common objectives and that have precisely described mechanisms and rules when and what to share.*

Note that the sharing process is not a simple file exchange but rather a direct access to computers, software, data, and other resources, for the achievement of required collaboration for problem-solving in industry, science, and engineering.

Ian Foster at all gave in [27] a sketch of the Grid with precisely defined layer architecture (see Figure 2.3.1).

The first **Fabric layer** provides the resources offered to shared access. The resources might be *computational resources*, *storage systems*, *catalogs*, *network resources*, and *sensors*. The fabric layer implements resource-specific operations (details in [28]).

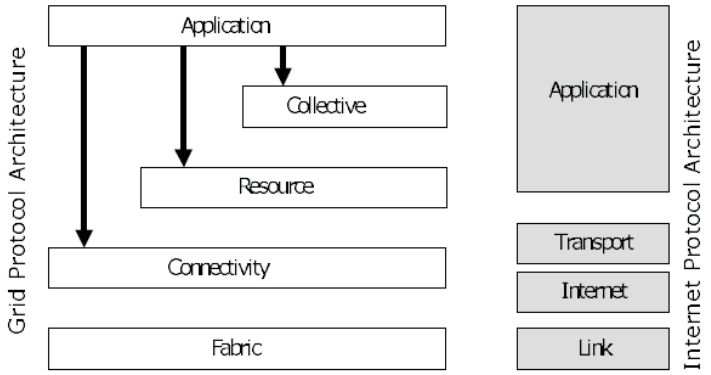


Figure 2.3.1: The Layered Grid Architecture and its Correspondence to the Internet Architecture (extracted from [28]).

The **Connectivity layer** defines the core *communication and authentication protocols* needed for network transactions between the Fabric layer resources. The Connectivity layer also provides protocols for authentication providing *security mechanisms* for verifying the identity of users and also resources (details in [28]).

The third **Resource layer** builds on Connectivity layer communication and authentication protocols (and APIs and SDKs) for the *secure negotiation, initiation, monitoring, control, accounting*, and *payment* of sharing operations on individual resources. Resource layer protocols are entirely concerned with individual resources ignoring the global state and atomic actions across distributed collections; such issues are the concern of the Collective layer discussed next.

The **Collective layer** takes care for the *global state* of the Grid and captures *interactions across collections* of resources.

Over the Collective layer the top **Application layer** is defined. The Application layer is the point where *calls of offered services* are made.

The layers are discussed in details in [28].

2.3.1 Security Systems for Grids

Treating the security and the authorization hazards in Grids is not an easy task. This is mainly due to the fact that a Grid is a *heterogeneous*, *distributed* system for *sharing expensive* and *valuable resources*.

The very common feature of security systems for Grids is the *replacement of identity based access* (e.g. password) with *authorization based on the attributes of users*. Such attributes can be mapped to users' accounts much easily achieving demanded properties of authorization (e.g. *delegation*).

- *Virtual Organization Management Service (VOMS)* [29] supports *attribute based access control*. The client retrieves a pseudo certificate consisting of client attributes (e.g. groups and roles) from VOMS servers and stores them in a non-critical extension of a common proxy certificate. These proxy certificates are then used to access the resource. The final decision is based on the attributes contained.

- *PERMIS* [30] project aimed at the creation of an X.509 role based *Privilege Management Infrastructure*. The main advantage brought by PERMIS is the ability to accommodate diverse access scenarios. PERMIS primary consists of two subsystems:

  1. the *privilege allocation subsystem* issuing a user X.509 certificate and storing it in LDAP (Lightweight Directory Access Protocol) [31] directories.
  2. the *privilege verification subsystem* which hauls the user certificates from a pre-configured list of LDAP.

- *AKENTI* [32] uses certificates signed by so called stakeholders from different domains in order to make a decision about access to a resource requested. AKENTI proposed three types of certificate stored in an XML format, particularly:

  1. *attribute certificates* binding an attribute-value pair,
  2. *use-condition certificates* indicating lists of relational expressions of required attributes to access rights and,
  3. *policy certificates* consisting of trusted *Certificate Authorities* CAs and stakeholders issuing use-condition certificates and lists of URLs where attribute certificates can be retrieved.

  Clients are then authenticated on their X.509 certificates.

- *System for Privilege Management and Authorization (PRIMA)* [33] also accommodates attribute X.509 certificates to enforce privilege and policy statements. Both certificates issued by a resource administrator and a stakeholder are used by a client to the resource Policy Enforcement Point (PEP). The PEP then validates the attributes and verifies with the resource Policy Decision Point (PDP) if the issuers are authoritative for user's presented privileges. All acknowledged privileges are gathered by the PEP and further presented to the PDP for verification against the access control policies. The PDP simply returns an authorization decision and a set of access recommendations (e.g. file accessible, user's quotas) for setting up a local account.

An important point worth mentioning is a way how to store policies representation in a standard language. EXtensible Access Control Markup Language (XACML) [34] is an example of standardized format (OASIS) [35] for specifying access control policies. However, current versions of XACML do not have expressiveness needed (e.g. *delegation of authority*).

2.4 Trust Management Systems

In the introduction we argued that the security cannot be treated as a single nor flat task. It would be rather a hierarchy where at least encryption algorithms and trust layers should be identified.

Trust management systems can be categorized into 3 categories:

- *credential* and *policy based trust management*;
- *reputation based trust management*, and;
- *social network based trust management*.

This categorization is based upon the way adopted for establishing and evaluating trust between entities.

Policy based approach has been proposed in the context of *open* and *distributed services architectures* [36],[37],[38],[39],[40] as well as in the context of Grids [41] as a solution to the *problem of authorization* and *access control* in open systems. Its focus is on trust management mechanisms employing different policy languages and engines for specifying and reasoning on rules for the establishment of the trust. Since the primary aim of such systems is to enable access control, trust management is limited to verification of credentials and restricting access to resources according to policies defined by a resources owner [42]. The resource owner provides an access to a restricted resource only if verification of credentials has been done successfully. Nevertheless, policy based systems need the requesting entity to establish trust with the resource owner, which unfortunately implies the fact that policy based systems do not provide a complete generic trust management solution for all decentralized applications.

On the contrary, **Reputation based** trust management systems provide a way in which *entities may evaluate* and *build a trust relationship* between resource provider and requester. Reputation approach emerged in the context of electronic commerce systems, e.g. eBay. In distributed settings, reputation-based approaches have been proposed for *managing trust in public key certificates*, P2P systems XREP [43], mobile ad-hoc networks, and recently, also in the Semantic Web [43], NICE [44], DCRC/CORC [45], EigenTrust [46],[47], [48],[49],[50],[51]. Typically, the reputation-based trust is used in distributed networks where any involved entity has only a limited knowledge about the whole network. In this approach, the reputation is based on recommendations

and experiences of other users/sites. Trust value assigned is a function of the combination of the peer's global reputation and the peer's perception of that entity.

Social network based trust management systems utilize, in addition, *social relationships between entities to infer trust*. In particular, the social network based system views the whole structure as a social network with relationships defined amongst entities. This social network can be further used as an input for deduction of trust among entities. Examples of such trust management systems include Regret [52], NodeRanking [53].

It is important to note, that there is clear distinction between *trust* and *reputation* [54]: a trust value T can be computed based on the reputation R, that is,

$$T = \phi(R, t) \tag{2.4.1}$$

where t is the time elapsed since the reputation was modified.

2.4.1 Dynamic Trust Management

Trust may be a highly dynamic phenomenon in many environments (e.g., P2P, mobile databases, the semantic web, the real human society). This fact has led researchers to investigation and proposals of new approaches for treating the trust dynamics.

In most approaches trust is defined as a vector comprising few factors contributing to the overall trust value (e.g. [49]):

- the *short term trust factor*,
- the *long term trust factor*,
- the *penalty factor*.

These factors are then combined into one value of *dynamic trust* of a particular connection between entities. The purpose of the factors can be generalized as an effort to accommodate sudden deviation in the normal behavior of an entity (so-called oscillation) together with long term behavior observation. The penalty factor is used to make a reaction of the system (decrease or increase of trust).

In the following equation we generalized the dynamic trust definition accordingly to our observation:

$$T_{ab}(t) = f(T_{P_{ab}}(t), T_{D_b}(t), M_a(t)) \tag{2.4.2}$$

where

- $T_{ab}(t)$ stands for the trust of entity a to entity b.
- $T_{P_{ab}}(t)$ is a personal observation of the entity b by entity a based on the history of interaction between the entities.
- $T_{D_b}(t)$ stands for the reputation of entity b given by the other entities.
- $M_a(t)$ is a model of behavior of an entity a, through which a personalization is achieved. We consider the personalization as a very important, since in the huge networks users will probably differ in the handling of trust.

All components consist of long as well as short time factors.

Chapter 3

Objectives

This section describes in details and more precisely the objectives of the thesis.

Let us begin with a short citation from [55] where demands for ideal security solution supporting trust are given as:
"The ideal solution would be a scalable distributed security approach where trust is easily discovered and realized and used to securely extend site autonomy to support collaborative work".
Our understanding of security corresponds narrowly to the given one.

The main objective of the thesis is to **propose**, **implement** and **verify** a **new security model for distributed environments** that *fulfills at least the following*:

- **General requirements**
  - the security model must capture *human intuition* about trust, thus being appropriate for securing human style communication,
  - *all entities involved in communication must be able to infer trust*,
  - the model should *support personalization*.
- **Implementation requirements**
  - the model must be able to work across *distributed environments*,
  - the model must be able to handle the *dynamic aspects of trust formation* and *trust evolution*.

3.1 Justification

The known approaches consider each entity as individual being *responsible for its own relationships*. Even the reputation systems where a non-direct trust can be inferred, consider entities as individuals. The same can be said for dynamic trust models.

In contrast, our proposal surpasses the known models for building trust in the *interpretation* of trust. In our point of view, the trust level is common for a group of users rather than a single user. In this way trust is not a single value for particular pair of entities but rather a shared value for a set (group) of entities. As the groups can differ in purpose[1], one entity can be member of more groups. Trust between two entities is then inferred based on their group memberships. Such model allows building trust between mutually unknown entities more easily and with less communication and computation load.

3.2 Dynamic Aspects

The human notion of trust is a long time driven process with possible severe oscillation phases followed by steady phases.

The proposed model M with several parameters $a_1, a_2, \ldots, a_n$ is able to describe dynamics of a systems of groups (dynamics corresponds to states in the following):

$$C^{k+1} \overset{\alpha}{\leftarrow} M(a_1, a_2, \ldots, a_n) \times C^k \tag{3.2.1}$$

where C^k is a state after k interactions and C^{k+1} is a state after $k+1$ interactions, α is the condition triggering the interactions.

The initial state C^0 is given as

$$C^0 \leftarrow f(I) \tag{3.2.2}$$

where I is an input structure describing existing relationships and trust among users transformed by a function f into starting state of the model.

The model M should fulfill the following requirements:

- **stability**: model *must not degenerate* into *any of limiting cases* (one *huge group of users*, or *many tiny groups*)
- model must preserve **overall** as well as **local security** of users
- **self-organization** and **self-monitoring** properties
- low time **complexities**.

[1] There might be groups of tennis players as well as lawyers, researchers, musicians, friends,...

Chapter 4

Mathematical Model

The chapter introduces the mathematical model and the terminology used in the rest of the thesis.

4.1 Basic Definitions

Definition 4 *An* **undirected graph** $G = (V, E)$ *is defined as a set of* **vertices** V *and a set of* **edges** E*. Every edge* $e_{ij} \in E$ *connects a pair of distinct vertices* v_i *and* v_j*.*
We call $Adj(v)$ *the* **adjacency set** *of vertex* v*, and we call the ordered pair* $(v, w) \in E$ *an* **edge***.*
Clearly

$$(v, w) \in E \text{ if and only if } w \in Adj(v).$$

If $(v, w) \in E(G)$ *then* v *and* w *are* **adjacent***, vertices of* G*, and the vertices* v *and* w *are incident with the edge* (v, w)*. We call a* **loop** *an edge which joins a vertex to itself.*

Definition 5 *The* **neighbor of vertex** v *(*$N(v)$*) is defined by*

$$N(v)=\{v\}+Adj(v).$$

Definition 6 *The* **degree** d_i **of a vertex** v_i *is defined as the size of the adjacent set of vertex* v_i *(*$|Adj(v_i)|$*).*

Definition 7 *A* **directed graph** *(***digraph***) G consists of a finite set V and an irreflexive binary relation on V.*
We call V the set of **vertices**.
The binary relation may be represented either as a collection E of **ordered** *pairs or as a function from V to its power set,*

$$Adj : V \to P(V).$$

Definition 8 *The* **weighted graph** *G is a quadruple (V, E, W_V, W_E), where V is a set of vertices, E is a set of edges, and W_V and W_E present weights of vertices and edges, respectively. (The weights are mappings from V and E to the set of reals.)*

Definition 9 *A graph G is* **complete** *if every pair of distinct vertices is adjacent. The complete graph on n vertices is usually denoted by K_n.*

Definition 10 *Given a subset $A \subseteq V$ of the vertices, we define the* **subgraph induced** *by A to be $G_A = (A, E_A)$, where*

$$E_A = \{xy \in E | x \in A \ \textbf{\textit{and}} \ y \in A\}.$$

Definition 11 Clique: *A subset $A \subseteq V$ of r vertices is an* **r-clique** *if it induces a complete subgraph, i.e. $G_A \cong K_r$.*

Definition 12 Stable set *is a subset X of vertices no two of which are adjacent.*

Definition 13 *A* **proper c-coloring** *is a partition of the vertices $V = X_1 + X_2 + \ldots + X_c$ such that each X_i is a stable set*
(the members of X_i may be "painted" with the same color i and adjacent vertices have to receive different colors).

Definition 14 *$\chi(G)$ is the smallest possible c for which there exists a proper* **c-coloring** *of G; it is called chromatic number of G.*

Definition 15 *A sequence of vertices* $[v_0, v_1, v_2, \ldots, v_m]$ *is a* **path** *from* v_0 *to* v_m of length m *in* G *provided* $v_{i-1}v_i \in E$ *for* $i = 1, 2, \ldots, m$.

Definition 16 Strongly connected graph *is a graph in which for any two vertices* x *and* y *there exists a path in* G *from* x *to* y.

Definition 17 *A* **subgraph** *of a graph* $G = (V, E)$ *is any graph* $H = (V', E')$ *satisfying* $V' \subseteq V$ *and* $E' \subseteq E$.

Definition 18 *Let* $G = (V, E)$ *be a simple directed graph without loops.*
A **triad** *is a subgraph induced by a given set of three vertices.*

Definition 19 *A* **hypergraph** $H = (U, N)$ *is defined as a set of vertices* U *and a set of hyperedges (nets)* N *among the vertices.*
Every **hyperedge** $n_j \in N$ *is a subset of the set of vertices* U.
The vertices in a hyperedge n_j *are called its* **pins** *and they are denoted as* $pins[n_j]$.
The set of hyperedges connected to a node u_j *is denoted as* $hyperedges(u_j)$.

Definition 20 *The* **size of a hyperedge** n_j *is given as:*

$$s_j = |pins[n_j]|.$$

Definition 21 *The* **degree of a vertex** v_i *is equal to the number of hyperedges it is connected to* ($d_i = |hyperedges[v_i]|$).

Definition 22 *A* **weighted hypergraph** H *is a quadruple* (U, N, W_U, W_N), *where* U *is a set of vertices,* $N \in 2^U$ *is a set of hyperedges, and* W_U *and* W_N *present vertex and edge weights, respectively.*
The weights are mappings from U *and* N *to the set of reals.*
(For simplicity, we will consider $U = \{1, ..., n\}$*).*

For hypergraphs a great deal of terminology like *paths, coloring,* etc. can be defined similarly as for graphs.

4.1.1 Partitioning problem

Linear system of equations are today widely solved by *iterative solvers* on *parallel computers*.
The *parallelization* is necessary as the size of the real matrix might be huge.
The goal of the **partitioning** is to enable parallelization of the following *sparse-matrix* vector product of the form of

$$\mathbf{y} = \mathbf{A}\mathbf{x} \tag{4.1.1}$$

where $\mathbf{A}$ is an $m \times m$ sparse square matrix, $\mathbf{y}$ and $\mathbf{x}$ are dense vectors.

In order to avoid the communication of vector components during the linear vector operations a **partition scheme** is adopted. It means all vectors used in the solver are decomposed conformally with the *row partitioning* to the *column partitioning* in the *rowwise* or *columnwise decomposition schemes*, respectively.

Graph Model for Decomposition

$\Pi = P_1, P_2, \cdots, P_K$ is a *K-way partition* of graph $G = (V, E)$ if the following conditions hold:

- each P_i, $1 \leq i \leq K$; $P_i \neq \oslash$ — *non-emptiness*
- $P_i \cap P_j = \oslash$ for all $1 \leq i < j \leq K$ — *disjoint*
- $\bigcup_{i=1}^{K} P_i = V$ — *completeness*

A K-way partition is called to be *balanced* if each part P_i satisfies the *balance criterion*:

$$W_i \leq W_{avg}(1 + \epsilon), \text{ for } i = 1, 2, \cdots, K. \tag{4.1.2}$$

where

- the weight W_i of a part P_i is $\sum_{v_i \in P_i} w_i$, $W_{avg} = (\sum_{v_i \in V} w_i)/K$ stay for the weight of each part under the *perfect load balance* condition,
- ϵ represents maximum acceptable *imbalance*.

A K-way partition is called *multiway* if $K > 2$ and a *bipartition* iff $K = 2$.
In a partition Π of G, an edge is said to be *cut* if its pair of vertices belong to two different parts, and *uncut* otherwise.
The cut and uncut edges are also referred as *external* and *internal edges*, respectively.
The set of external edges of a partition Π is denoted as ϵ_E.

The *cutsize definition* for representing the cost $\chi(\Pi)$ of a partition Π is:

$$\chi(\Pi) = \sum_{e_{ij} \in \epsilon_E} c_{ij}$$

where each cut edge e_{ij} contributes its cost c_{ij} to the cutsize. Hence, the graph partitioning problem can be defined as the task of dividing a graph into two or more parts such that the cutsize is minimized, while the balance criterion 4.1.2 on part weights is maintained.

The Graph model nevertheless faces some limitations[1] in this context therefore the hypergraph model was proposed instead.

Hypergraph Model for Decomposition

K-way partitioning of hypergraphs is defined similarly to that of graphs.
In a partition Π of H, a hyperedge having at least one pin (vertex) in a part is said to connect that part.
Connectivity set Λ_j of a hyperedge n_j is a set of parts being connected by n_j.
Connectivity $\lambda_j = |\Lambda_j|$ of a hyperedge n_j denotes the number of parts connected by n_j.
A hyperedge n_j is said to be *cut* (external) if it connects at least 2 parts ($\lambda_j > 1$), and *uncut* (internal) otherwise. The set of external hyperedges is denoted as N_E.
The *cutsize* representing the cost $\Pi(P)$ of a partition P is given as.

$$X(\Pi) = \sum_{n_j \in N_E} c_j(\lambda_j - 1) \tag{4.1.3}$$

In the *column* hyperedge model, a matrix $\mathbf{A}$ is represented as a hypergraph $H_R = (V_R, N_C)$, where vertex and hyperedge sets V_R and N_C corresponds to the rows and columns of matrix $\mathbf{A}$, respectively.
Each vertex $v_i \in \mathcal{V}_R$ corresponds to the atomic task of computing the inner product of row i and column vector $\mathbf{x}$.
The hyperedges of H_R represent the *dependency* relations of the atomic tasks on the $\mathbf{x}$-vector components in rowwise decomposition.

The most important advantage of the hypergraph model is the ability to fit the real communication need better.

[1] e.g. the graph model does not allow to express directly the real communication volume implied by the partitions.

4.2 Security Model for Dynamic Distributed Environments

This section describes relations between the given terminology and our security model together with the general requirements on the security model.

4.2.1 Security Model Requirements

The security model should fulfill at least the following requirements:

1. **ability** to *describe complicated relationships* between users
2. **ability** to *describe dynamic aspects* of groups of users
3. **efficient** implementation
4. **distributed** implementation

4.2.2 Security Model Proposal

Currently available solutions (mentioned in the Related Work Section) concern *concrete trust between members*[1] where a direct *relationship exists* or where a *transitive relationship* can be found. Such approaches can be very *naturally modeled* as *oriented weighted graphs* where *vertices* correspond to *members* and *edges* represent *relationships*. **Level of trust** is simply *described by weight of edges* and *can vary over time*.
Graph model is sufficient in the case of complicated relationships among members, but modeling groups of users efficiently could pose problems. A group of users can be modeled as a graph where group members are connected by edges with the same weights, but such approach suffers by a space load overhead. From practical point of view more accurate model is necessary. In hypergraphs a hyperedge connects arbitrary many vertices[2] and one vertex can be a pin of more hyperedges. Figure 4.2.1 shows a simple example of a hypergraph with 4 hyperedges and 10 vertices.

From the figure it follows that a hypergraph can describe a very complicated structure of groups of users straightforwardly. In addition, weights can be used in order to describe a structure of users in more details, not just who is connected to whom, but also reliability, security, error proneness or other additional properties.

Let us step back to the list of the model requirements given and check the items for the hypergraph model:

[1] In the rest of the thesis we will use terms members and users as equivalent.

[2] The upper bound is naturally given by the number of vertices $|U|$ of a hypergraph and the lower bound is 1 vertex.

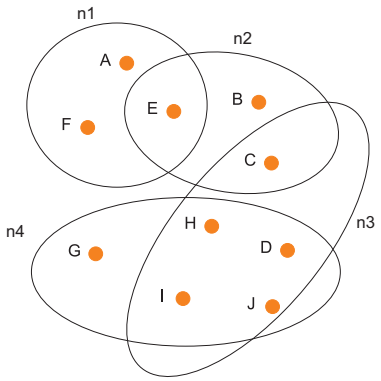


Figure 4.2.1: Example of a Hypergraph Representation (consisting of 10 vertices and 4 hyperedges corresponding to a system of 4 distinct groups with possibly different level of trust between 10 users)

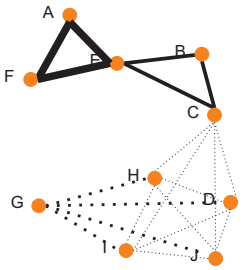


Figure 4.2.2: Example of a Graph Representation (the same situation as in Figure 4.2.1, but now described by a graph – edges in the same line type and width describe the same level of trust; orientation is omitted for better lucidity)

1. *ability to describe complicated relationships between users* - weighted hypergraph can describe arbitrarily complicated groups of users – a *hyperedge* corresponds to a *group of users*.

2. *ability to describe dynamic aspects of groups of users* - a weighted hypergraph can be easily reconfigured by adding/deleting vertices as well as hyperedges.

3. *efficient implementations* - there exist several efficient implementations of hypergraphs proposed for numerical mathematics problems (e.g. hypergraph model for decomposition).

4. *distributed implementation* - a hypergraph model can be easily distributed so that each vertex can store the set of hyperedges it is a pin.

4.2.3 Security Model Representation of a DVO

Our model uses hypergraphs for the description of groups of users. In the following we explain relations between a hypergraph $H = (U, N, W_U, W_N)$ and the structure of groups of users:

1. *vertices* U represent *users*
2. the *weight of a vertice* W_{u_i} represents *user* u_i *attributes*
3. *hyperedges* N represent *groups of users*
4. the *weight of a hyperedge* W_{n_i} represents *trust shared by the group*
5. *pins* of a hyperedge $pins(n_i)$ represent the *members of the group described* by the hyperedge n_i
6. $hyperedges(u)$ represents *set of groups* of a user u

In the rest of the thesis Virtual Organization term will be extended from its primary definition [27]. In our case the VO is not a temporal but rather long-live coalition of users with the same or very similar intentions. Furthermore, in a particular VO the trust between members is the same. The proposed model does not consider the only one VO but it is rather concern with a set of interconnected VOs. Therefore in the rest of the thesis we will denote a system of interconnected VOs with dynamic changes in the relationships between users as *Dynamic Virtual Organization* (DVO).

Chapter 5

Proposal of Algorithms

The chapter describes a *security model* SecGrid *built on the hypergraph model* presented in the previous chapter. The SecGrid model comprises the algorithm for transformation of an input structure into the hypergraph model, algorithms preserving consistency of the structure of DVOs and the security subsystem.

5.1 G2H Algorithm

It is known and accepted that graph structures can suitably describe relationships between users and they are used in this way. Since our security model uses hypergraphs for description of dynamic Virtual Organizations (DVO), a transformation of a general graph input structure into the hypergraph one is needed.

For generality, we will assume that the input structure representing trust between users is given in form of a directed graph[1] $G = (V, E, W_V, W_E)$ where the following holds [2]:

- a user is represented by a vertex $v_i \in V$
- relationship between users v_i and v_j is represented by a directed edge $(v_i, v_j) \in E$
- trust between users v_i and v_j connected by an edge is represented by the weight of the edge (v_i, v_j)
- a user v_i related information (name, address, abilities, interests, etc.) is represented by weight of the vertex W_{v_i}

In the rest of this sub-section we will describe how the Graph-to-Hypergraph (G2H) Algorithm is used for transformation of the input graph structure into the hypergraph model.

[1] Note the direction of edges is very important since trust may not be mutual.

[2] See Figures 4.2.1 for graphical representation of groups of users.

The transformation cannot be done arbitrarily. It should respect at least the following items important for getting a realistic model. The first two reflect the transformation, whereas the last two are important for implementation.

1. A hyperedge represents a group of users sharing the same level of trust – the pins of the hyperedge have to be represented by "closely related" vertices in the input graph where *closely related* means:
   - *type* of relationships,
   - *level of trust*.
2. The "orientation" of relationships has to be considered.
3. Transformation of *dense* as well as *sparse* input graphs.
4. Possible *distributed implementation*.

5.1.1 Transformation of an Input Graph

In our proposal, the Graph-to-Hypergraph (G2H) Algorithm realizes the needed transformation. The G2H Algorithm takes a weighted directed graph $G = (V, E, W_V, W_E)$ as the input. It creates possibly non-disjoint subsets V_i of the vertex set V:

$$V = \bigcup_{i=1}^{L} V_i \tag{5.1.1}$$

where L is the number of hyperedges and V_i are sets of pins of hyperedges for $i = 1, \ldots, L$.

Currently, we use two modifications of the G2H Algorithm;

- *based on search for* **strongly connected components**,
- *based on search for the so called* **triads**[56].

G2H Algorithm Based on Strongly Connected Components

The main idea of this version of the G2H Algorithm relies on the fact that any two vertices of the same strongly connected component are reachable through a path (see definition (16)).

From the social network point of view, such vertices (users) have direct knowledge of themselves or can infer mutual relationship via the other vertices (users). Therefore, vertices in the same strongly connected component are good candidates for the formation of a new group (hyperedge).

The implementation of this version of the G2H Algorithm is based on the very efficient Tarjan's [57],[58],[59] algorithm. The Tarjan's algorithm can be straightforwardly implemented in a distributed environment with the message passing (MPI [60]) therefore fulfilling all required properties (see section 5.1)

except for the ability to cope with dense input graphs as was shown by our experiments.

The time complexity is driven by the complexity of the Tarjan's algorithm given as:

$$O(N) \tag{5.1.2}$$

where N is the number of vertices in the input graph.

G2H Algorithm Based on Triads

The basic G2H Algorithm based on the search for strongly connected components is known to be unable to cope with dense input graphs. Therefore, we here propose an improved version of the G2H Algorithm based on search for triads[3].

Algorithm 1 The G2H Algorithm

```
1: procedure SearchSeed(G = (V, E))
2:     for all a ∈ V do
3:         if (∃b ∈ V)(a → b ∈ E ∧ b → a ∈ E) then
4:             add a, b into h_i ∈ H
5:             AppendSeed(G, h_i, max(|a → b|, |b → a|))
6:             i = i + 1
7:         end if
8:     end for
9: end procedure
10: procedure AppendSeed(G, h, max)
11:     while (∃x ∈ V : x ∉ h, a ∈ h, b ∈ h)(a → x ∈ E ∧ x → b ∈ E)
    ∨(b → x ∈ E ∧ x → a ∈ E) do
12:         if (|a → x → b| ≥ max) ∨ (|b → x → a| ≥ max) then
13:             add x into h
14:         end if
15:     end while
16: end procedure
```

In the Algorithm 1 the improved version of the G2H Algorithm is described in a pseudo code, where

- "$\rightarrow$" denotes a directed edge,
- "$|b \rightarrow a|$" denotes the weight of the directed edge between vertices a, b,
- "$|b \rightarrow x \rightarrow a|$" denotes the minimum weight of all edges in the directed path through vertices b, x, a,

[3]In the rest of the thesis by the G2H Algorithm will be meant the G2H Algorithm based on triads.

- "$|b \to x \to a|$" equals to 0 if no such directed path exists.

The algorithm is divided into two parts represented by

- *SearchSeed* procedure: The *SearchSeed* procedure analyzes all vertices in the input graph and creates *seeds* h_i. The seeds are basic graphs obtained from complete graphs of size 2 ($K2$) for which are created vertex subsets V_i (see lines 3 and 4).
- *AppendSeed* procedure: If any seed has been found by the *SearchSeed* procedure then the *AppendSeed* procedure tries to identify a vertex that is interconnected with the seed by a triad with required orientation (line 11). The triads accepted by the algorithm are shown in Figures 5.1.1 and 5.1.2 (the seed is shown in the dotted ellipse in both cases). If such vertex x exists then it is added to the seed (line 13).

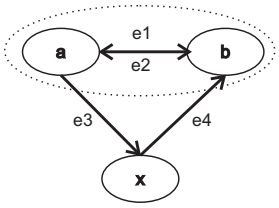


Figure 5.1.1: Accepted Triad "1"

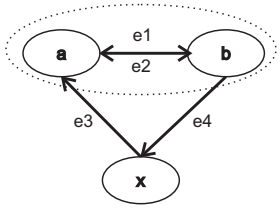


Figure 5.1.2: Accepted Triad "2"

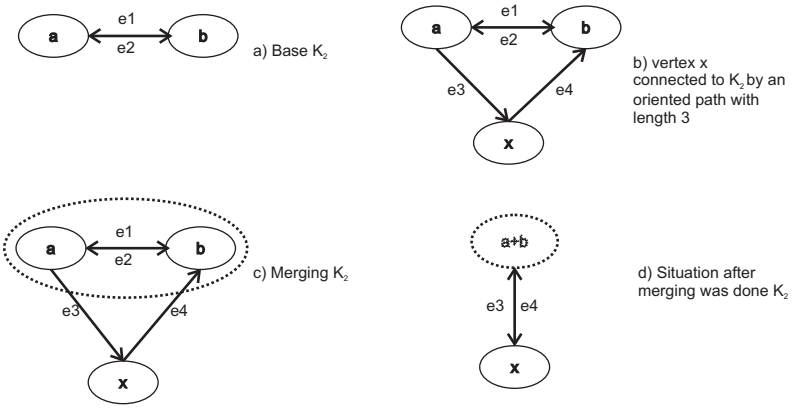


Figure 5.1.3: G2H Algorithm Transforming Edges into Hyperedges

In Figure 5.1.3 is depicted the G2H Algorithm graphically with comments provided.

a) The creation of a seed from vertices a and b.
b) The *AppendSeed* procedure searches for any vertex x connected to the seed by an acceptable triad.

c) The addition of vertices a and b into a new hyperedge – the vertices are now represented as one vertex.
d) The emerge of a new $K2$ between the seed and vertex x – vertex x is added into the hyperedge; return to a).

The G2H Algorithm works in this way until all vertices ($v_j \in V$) are examined by the *SearchSeed* procedure resulting in partition of vertices $\{V_i\}$ of the input graph.

The partitioned vertices of the input graph are then the input for the transformation operation itself. The output hypergraph will be in the form:

$$H = (U, N, W_U, W_{1N}, W_{2N}) \tag{5.1.3}$$

where the following relations between the input graph $G = (V, E, W_V, W_E)$ and the output hypergraph H hold:

- vertices U in H correspond to vertices V in G,
- weights of vertices W_N in H correspond to weights of vertices W_V in G,
- pins of a hyperedge $pins(n_i)$ in H correspond to a partition of vertices $\{V_i\}$ in G,
- the weight of hyperedge W_{1N} in H corresponds to the maximal weight of the edges in the $K2$ seed of V_i,
- the weight of hyperedge W_{2N} in H corresponds to the difference between the weight of edges in $K2$ seed of V_i,

whereas W_{1N} denotes *maximal level of trust* (remind that weight of an edge represents a level of trust between users) in the $K2$ seed, W_{2N} represents *difference in the level of trust* in the $K2$ seed.

The *AppendSeed* procedure of the G2H Algorithm uses the weights of the $K2$ seed for the formation of the group from triads (line 12). This step protects groups from penetration of users that do not satisfy condition on maximal level of trust and difference in weights during the group formation process. Generally, the more secure group (with higher level of trust) the higher W_{1N} (maximal weight) and lower W_{2N} (difference between the weight).

Apparently, the G2H Algorithm based on search for triads poses the finite termination property.

Time complexity of the G2H Algorithm based on triads is given by

$$O(\sum_{v \in V} |Adj(v)|) \tag{5.1.4}$$

The complexity is given by the fact that *SearchSeed* has to identify $K2$ seeds in the input graph. This can be done by searching the adjacency sets of vertices in $\sum_{v \in V} |Adj(v)|$ steps. If any seed has been detected, then the *AppendSeed* procedure traverses the vicinity of the seed. This traverse has complexity $O(|Adj(v)|)$. The overall time complexity is dominated by the complexity of the *SearchSeed* procedure.

5.2 Structure Dynamics

During the specification of the objectives we stressed the requirement on the dynamics of the security model. In other words, the security model should be able to cope with the fact that relationships among users subject to changes. Therefore, this section introduces our concept of the dynamic part of the security model – the Structure Dynamic (SD) Algorithm.

5.2.1 SD Algorithm

The main task of the SD Algorithm is the preservation of the *local security* of the dynamic groups of users (DVO). Since the security model should be in practice fully distributed, it is not an easy task.

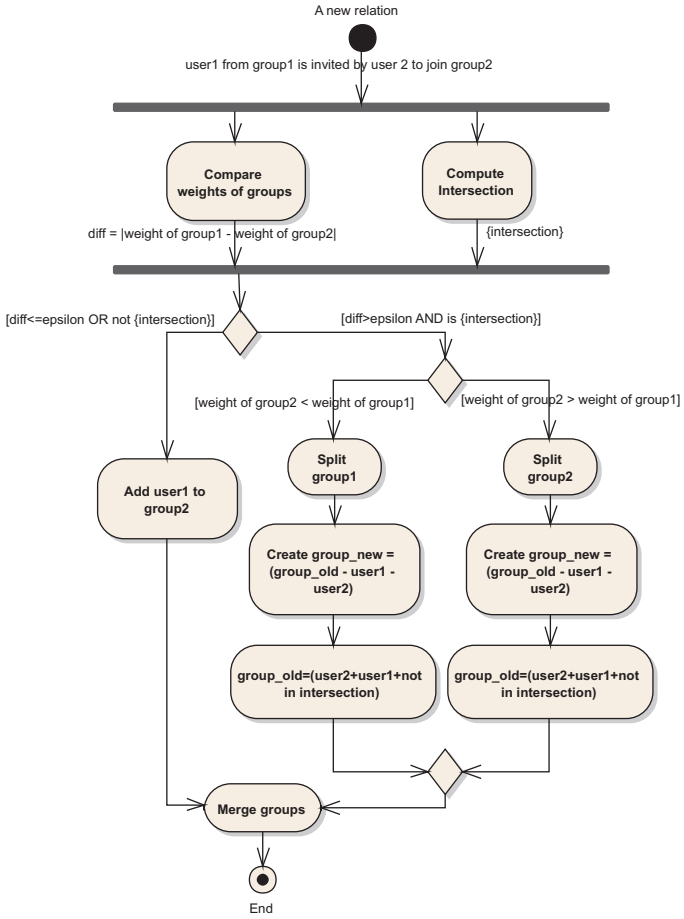


Figure 5.2.1: UML Activity Diagram (describing the procedures of the SD Algorithm).

The input of the SD Algorithm is the tuple $(user1, group1, user2, group2)$,

with the following interpretation: the $user1$ from the $group1$ is invited by the $user2$ from the $group2$ to join the $group2$. The SD Algorithm is described in the UML activity diagram format in Figure 5.2.1.

The algorithm begins with two procedures. The *Compare weights of groups* compares the weights (levels of trust) of $group1$ and $group2$ and returns the difference as *diff*. The second one *Compute Intersection* tries to identify the members of both groups by computing the *intersection*.

- If the difference *diff* is less than a predefined positive threshold ϵ or there are no members in the *intersection*, the $user1$ is simply *added* to the $group2$,
- otherwise the *split group* procedure splits the group with the higher level of trust.

At the end the *Merge groups* procedure is triggered merging groups with the *intersection* larger than a positive threshold λ and with shared levels of trust differing less the ϵ.

Let us describe the parts of SD Algorithm less formally. The *Compare weights of groups* procedure computes the difference in the levels of trust between the groups and the *Compute Intersection* identifies the *intersection* – members common for both groups.

- If this difference is lower or equal than the threshold ϵ, then groups share the same or almost the same trust and the new user is welcome to join the $group2$ (the *Add user1 to group2* procedure in Figure 5.2.1).
- In addition, if there are no members in the *intersection*, then none of current group members can identify potential dangerousness of incoming $user1$ as no member of $group2$ knows trust shared in the $group1$. Therefore the $user1$ is added to the $group2$.
- If the difference is higher than the threshold ϵ then the *Split* $group_x$ procedure preserves the trust of the group by isolating potentially untrustworthy users.

A non-empty *intersection* and the difference between levels of trust of $group1$ and $group2$ larger than the threshold ϵ suggests possible security violation.
In other words there are two groups ($group1$, $group2$) with different levels of trust. The difference is in addition to that larger that the allowed difference ϵ of $group2$. The important point here is that the difference in trust of the groups can be revealed only by user(s) being member of both groups. If no such user exists, that no one can reveal the threat and invited user is simply added.
Therefore, the group with the higher level of trust is split into *group_old* and *group_new*. The *group_old* contains users not in the intersection together with

$user1$ and $user2$, whereas the *group_new* comprises members of the former $group2$ apart from the $user1$ and the $user2$. This splitting preserve local security of the users, because both potentially malicious users ($user1$ and $user2$) are isolated in group with lower level of trust and the *group_new* with higher level of trust comprises only users not involved in the invitation. On the other hand, as some users may remain members of both groups (new and old ones), it is possible in the future to merge these groups, if $user1$ and $user2$ have been proved to be the "good" ones (remain the *Merge groups* procedure).

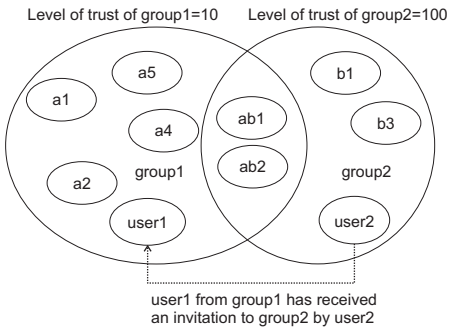


Figure 5.2.2: Initial Configuration

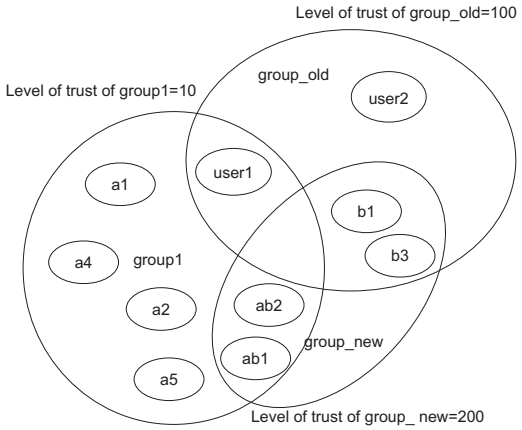


Figure 5.2.3: Situation After Splitting

Figures 5.2.2 and 5.2.3 graphically show the procedure described above. At the beginning (Figure 5.2.2) there are two groups with different levels of trust and two users in the intersection ($ab1, ab2$). The $user2$ issues the invitation for $user1$ to join the $group2$. The next Figure 5.2.3 shows the final state after splitting. Whereas $group1$ remains unchanged, $group2$ is divided into *group2_old* and *group2_new*.

Chapter 6

Security Sub-system

The algorithms mentioned previously take care for the transformation of a general input into the hypergraph model and for handling dynamics of groups of users. The security sub-system proposed in this chapter is responsible for treating threats and basic operations emerging during the model evolution.

The chapter first introduces the known threats to reputation systems and then a detailed proposal of the security sub-system followed by discussion on expected security improvements is presented.

6.1 Threats to Reputation Systems

The following list briefly describes general techniques available to adversaries to infiltrate or disturb a reputation system [61].

- **Traitors**. Malicious peers behave properly for a period of time in order to build up a strongly positive reputation. Once having the reputation build, they begin defecting. This technique is mostly effective when stronger positive reputation gives a peer additional privileges.
- **Collusion**. Multiple malicious peers cooperate together to cause more damage. This is especially true in peer-to-peer reputation systems, where covert affiliations are untraceable and the opinions of unknown peers impacts one's decisions [62].
- **Front peers** ("moles"). Malicious colluding peers always cooperate with others in order to receive strong reputation [63]. Then they provide misinformation to promote actively malicious peers. This form of attack is particularly difficult to prevent in an environment where there are no pre-existing trust relationships and peers guide their interactions only on the word and actions of others [64].
- **Whitewashers**. Peers that purposefully leave and rejoin the system with a new identity in an attempt to shed any bad reputation they have accumulated previously [65].

6.2 Security Sub-system Design Principles

Our security sub-system utilizes basic principles of reputation systems based on social networks, together with:

1. **Avoiding long-term secrets**. Even though long-term secrets are often used for security management, they may present a real security threat. Such long-term secrets require a storage to be hard or impossible to replicate, audits, repairs or regeneration. In long time scenario, such secrets are likely to leak, resulting in extraordinarily difficult corrections.
2. **Avoiding rapid changes**. In dynamic system changes are inevitable, but there is never a need for a rapid change. A system making changes to the system of groups (addition of new members, merging of groups,...) so quickly that users are overflowed by number of announcements of changes or users are unable to track such changes is not suitable.
3. **Reduced impact of third-party reputation**. Third-party reputation provides very useful information, however if used incautiously it may cause more damages than advantages (especially in long-term dynamic systems).

6.3 Security Sub-system Proposal

It is important to note the fact that in our point of view, the trust is not a single value valid for a particular pair of users, but it is a shared one for a group of users.

The list bellow contains the main building stones of the security sub-system:

1. **tKey**. The local security of a group is maintained by *tKeys*.
   tKeys are exchanged between users in order to verify their group(s) memberships (see section 6.3.1).
2. **Shadow groups**. In order to avoid the rapid changes in groups of users, each group maintains a *shadow group*. The *shadow group* contains users invited to join the group that have not been accepted by the enough group members (see 6.3.2 for further details).
3. **tKey Management Schema**. Dynamics of groups requires an efficient *tKeys* management schema preserving the local security of groups (see 6.3.3 for more details).

6.3.1 tKey

A user maintains one *tKey* for each group he/she is member.
A *tKey* (Figure 6.3.1) comprises 7 fields:

- The *Group* field contains identification of the group.
- The *GroupFrom* presents a group from which the member was invited.
- The *Base|user* field contains fingerprint(s)[1] of the member that issued the invitation plus the fingerprint of the user himself/herself, followed by the user identification itself (separated by |).
- The $Sign_{u1} \dots Sign_{un}$ field contains signatures received up to now from full privilege members of the group.
- The *Trust* field represents the level of trust of the group.
- The parameters λ and ϵ drive the *Split group* and the *Merge group* procedures of the SD Algorithm.
- The parameter α represents amount of signatures required for full privilege membership.
- *TTL* is a time validity of the *tKey*.

Figure 6.3.1: *tKey* Structure

6.3.2 Shadow Groups

Whenever a user is invited to join a new group, he/she receives a new *tKey* and he/she becomes a member of the corresponding *shadow group*. Shadow group members:

- contact the *full privilege group members* in order to *receive the required amount*[2] *of signatures* (a personal mark of a group member) approving their invitation,
- have limited privileges:
  - he/she cannot invite a new user,
  - he/she cannot trigger splitting or merging of groups,
  - shadow group members do not contribute to the size of intersections calculated for merging.

The full privilege members may apply restrictions on requests coming from shadow group members.

[1] Fingerprints can be realized for example by public/private keys,...

[2] Required amount of signatures may differ from group to group and it is set up by the group members.

6.3.3 tKey Management and Verification Schemata

The *SD Algorithm* presented previously relies on the fact that users can find out from which group and by whom was a new user invited, which may not be easy in the totally decentralized SecGrid model. In the following techniques for managing each operation of the *SD Algorithm* together with the verification of *tKeys* in the SecGrid model are described.

We use the following notations:

- $tKey(u, n) \ldots$ represents a value of the tKey of the user u for the group n
- $tKey(u, n) : base \ldots$ represents *base field* of the tKey of the user u for the group n
- $tKey(u, n) : signs \ldots$ represents *signatures* of the tKey of the user u for the group n
- $tKey(u, n) : TTL \ldots$ represents TTL *field* of the tKey of the user u for the group n
- $|tKey(u, n) : base| \ldots$ represents *number of fingerprints* in the base field of the tKey of the user u for the group n
- $|tKey(u, n) : signs| \ldots$ represents *number of signatures* of the tKey of the user u for the group n
- $fingerprint(u) \ldots$ represents the *fingerprint* of the user u

Creation of a New tKey

Let there be two users u_a and u_b of distinct groups n_a and n_b, respectively. Furthermore, assume that u_a has received an invitation from u_b. A new tKey for u_a for group n_b is created as follows:

1. u_b firstly sends to u_a $tKey(u_b, n_b) : base$ together with a new TTL
2. u_a appends his/her fingerprint to the *base* of the received tKey $tKey(u_a, n_b) : base = tKey(u_b, n_b) : base + fingerprint(u_a)$
3. u_a is a member of the shadow group of n_b until the following condition holds valid:

$$|tKey(u_a, n_b) : base| > \alpha + |tKey(u_a, n_b) : signs| \tag{6.3.1}$$

   where parameter α drives the required amount of signatures.

4. If condition (6.3.1) is invalid, the user u_a became a full privilege member of n_b. From this point TTL is not further important.

Members of a group are able to express their verdicts on the new members users through the corresponding *shadow group*. If a *shadow group* member has not received the needed amount of signatures in time (set by TTL), his/her

tKey became invalid. All users are aware of this, since every user can verify inequality (6.3.1) and TTL value in the tKey received with a request[3].

tKey Verification

The motivation scenario (sub-section 1.1.1) was based on the fact that users were able to deduce their common groups. In the SecGrid model this is put into effect by tKeys and the following verification schema:

1. Users exchange their *tKeys*.
2. Users examine *group* fields of each *tKey*: *in case of match* users verify the *tKey* by:
   (a) verify whether the *tKey* has been signed according to condition (6.3.1)
   (b) if succeeded, examine the *base* field which expresses by whom was the *tKey* holder invited,
   (c) otherwise check TTL against current time to verify shadow group members.

After this verification is done, both users know:

1. whether a common group(s) exists, if so then also:
   (a) *whether* users are *full privilege* or *shadow group* members
   (b) *who* has up to now signed their *tKeys*
   (c) *by whom* were users invited
   (d) *until* what time shadow group membership is *valid*

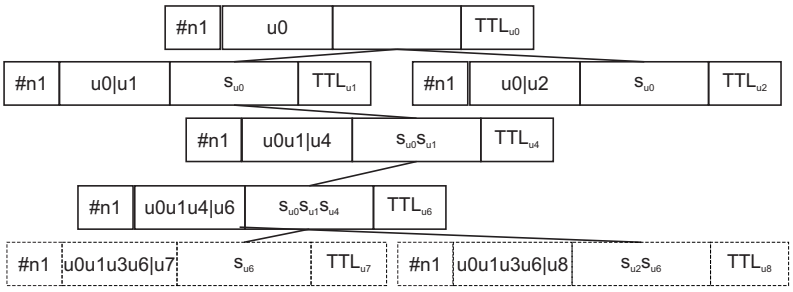


Figure 6.3.2: *tKeys* Tree Structure (representing sequence in which users were invited. Parameters *trust*, λ, ϵ and α are omitted for better lucidity.)

An example situation is given in Figure 6.3.2, where is shown a tree diagram of a model group *#n1* (assume parameter $\alpha = 1$).

[3]Note that users exchange their tKeys in order to infer their common groups used for trust derivation.

There are all together 7 members in the figure.
Members marked as $u7$ and $u8$ are in the shadow group as they are short in number of signatures for $\alpha = 1$, $|tKey(u8, \#n1) : base| = 4$ and $|tKey(u8, \#n1) : sign| = 2$ violates the condition (6.3.1). In addition, by verifying $tKey(u6, \#n1)$ one can found:

- member was invited by the member with base $u0u1u4$ (member $u4$),
- up to now the $tKey(u6, \#n1)$ has been signed by members $u0$,$u1$,$u4$,
- the user is a full privilege member, condition (6.3.1) does not hold.

6.4 Handling SD Algorithm Actions by tKeys

The management and verification schemata presented above provide users with information needed to handle each of the *SD Algorithm* actions, particularly:
Addition of a new user: when a new $user1$ from the $group1$ receives an invitation from $user2$ from $group2$, a new $tKey(user1, group2)$ is created. The $user1$ remains a member of the shadow group while condition (6.3.1) holds.
Splitting of groups: the splitting procedure is driven by the difference (given by the parameter ϵ) in weights of groups. The SecGrid model enables users to react on an invitation by checking members in the *shadow group*. For this purpose the *tKey* contains *GroupFrom* field and *base* field; any member can find out from which group the new user was invited and by whom – all information needed to trigger splitting.
Merging of groups: two groups are merged if number of common members is higher than a threshold λ. The number of common users (size of intersection) of groups contains the *base fields* of *tKeys*. For example assume three groups $\#n_1, \#n_2$ and $\#n_5$ and three representative members u_{11}, u_{15}, u_{23} with the following tKeys:

Group	GroupFrom	Base|user	$Sign_{u1}$ ... $Sign_{un}$	Trust	λ	ε	α	TTL
$\#n_1$	...	$u_1,u_3,u_6,u_8,u_9\|u_{11}$	$s_{u2},s_{u5},s_{u6},s_{u8}$	90	1	30	2	...
$\#n_3$	...	$u_1,u_2,u_4,u_5,u_8,u_{10}\|u_{15}$	$s_{u1},s_{u3},s_{u7},s_{u8},s_{u14}$	115	3	20	2	...
$\#n_5$	...	$u_1,u_3,u_4,u_5,u_9,u_{10}\|u_{23}$	$s_{u1},s_{u2},s_{u3},s_{u6},s_{u8},s_{u9}$	110	3	10	1	...

Figure 6.4.1: Merging of Groups Example: *tKeys* for Groups $\#n_1, \#n_3, \#n_5$ with the General tKey Structure (Fields not involved in the merging procedure are filled with "...").

By examining the base fields of the tKeys in the example (Figure 6.4.1) the following can be found:

$$intersection(n_1, n_3) = \quad (u_1, u_8) \tag{6.4.1}$$
$$intersection(n_1, n_5) = \quad (u_1, u_3, u_9) \tag{6.4.2}$$
$$intersection(n_3, n_5) = \quad (u_1, u_4, u_5, u_{10}) \tag{6.4.3}$$

The following conditions must hold in order to merge group $\#n_x$ with group $\#n_y$ (see the SD Algorithm in sub-section 5.2.1):

$$|intersection(n_x, n_y)| \quad > \lambda_{n_x} \tag{6.4.4}$$
$$|trust_{n_x} - trust_{n_y}| \quad < \epsilon_{n_x} \tag{6.4.5}$$

where

- $|intersection(n_x, n_y)|$ is the number of common members of groups n_x and n_y,
- $|trust_{n_x} - trust_{n_y}|$ is the difference in the levels of trust of groups n_x and n_y.

The following list gives an overview on merging of three groups in the example (Figure 6.4.1):

- group $\#n_1$ being merged with group $\#n_3$:
  - $|intersection(n_1, n_3)| = 2$ (6.4.1), $\lambda_{n_1} = 1$
  - $|trust_{n_1} - trust_{n_3}| = 25, \epsilon_{n_1} = 30$
  - implying that members of group $\#n_1$ **agree** on merging with $\#n_3$.
- group $\#n_1$ being merged with group $\#n_5$:
  - $|intersection(n_1, n_5)| = 3$ (6.4.2), $\lambda_{n_1} = 1$
  - $|trust_{n_1} - trust_{n_5}| = 20, \epsilon_{n1} = 30$
  - implying that members of group $\#n_1$ **agree** on merging with $\#n_5$.
- group $\#n_3$ being merged with group $\#n_1$:
  - $|intersection(n_3, n_1)| = 2$ (6.4.3), $\lambda_{n_3} = 3$
  - $|trust_{n_3} - trust_{n_1}| = 25, \epsilon_{n_3} = 20$
  - implying that members of group $\#n_3$ **disagree** on merging with $\#n_1$.
- group $\#n_5$ being merged with group $\#n_1$:
  - $|intersection(n_5, n_1)| = 3$ (6.4.2), $\lambda_{n_5} = 3$
  - $|trust_{n_5} - trust_{n_1}| = 20, \epsilon_{n_5} = 10$
  - implying that members of group $\#n_5$ **disagree** on merging with $\#n_1$.

- group $\#n_3$ being merged with group $\#n_5$:
  - $|intersection(n_3, n_5)| = 4$ (6.4.3), $\lambda_{n_3} = 3$
  - $|trust_{n_3} - trust_{n_5}| = 5, \epsilon_{n_3} = 20$
  - implying that members of group $\#n_3$ **agree** on merging with $\#n_5$.
- group $\#n_5$ being merged with group $\#n_3$:
  - $|intersection(n_5, n_3)| = 4$ (6.4.3), $\lambda_{n_5} = 3$
  - $|trust_{n_5} - trust_{n_3}| = 5, \epsilon_{n_5} = 10$
  - implying that members of group $\#n_5$ **agree** on merging with $\#n_3$.

From the list it follows that only groups $\#n_5$ and $\#n_3$ can be merged as fulfill the conditions (6.4.4) and (6.4.5).

The most important fact is that the information needed in order to trigger merging are accessible for every user without any support of a centralized unit, thus making users conscious about the state of the group by exchanging their tKeys in the SecGrid model.

In other words, whenever the $user1$ sends his/her *tKey* to $user2$, $user2$ receives a new *base*. Note that the freshest *base* field contains *tKeys* of novice members, therefore in order to make the updates to the base fields of group members more efficient, we suggest to use some kind of reconciliation protocol [66].

6.5 Back to the Security Sub-system Design Principles

1. **Avoiding long-term secrets**. The SecGrid model requires a private identification of a user in the system - *a fingerprint*.
   A user can issue a new *fingerprint* whenever needed (e.g. the old become compromised). The new *fingerprint* is valid if signed by the enough number of the other members (see section 6.6.4 below).
2. **Avoiding rapid changes**. The SecGrid model makes use of *shadow groups* to avoid rapid changes.
3. **Reduced impact of third-party reputation**. Note that in the SecGrid model there are only members of groups. No *transitive* (corresponds to third-party reputations in the graph model) group membership is allowed (members cannot access groups where they are not members).

6.6 Analysis of Attacks

Even thought the world with only honest users would be a wonderful place to live, the current world is not the case. Therefore, in the following we put the SecGrid model under investigation against the well known attacks on reputation systems.

6.6.1 Traitors

Traitors build strong *positive reputations*, which correspond to the membership in highly trusted groups of users in the SecGrid model, and then start defecting.

The SecGrid model treats potentially dangerous users by splitting groups. In this way, dangerous users are isolated in groups with a lower level of trust.

The splitting procedure splits group if any current full privilege member is conscious about difference in the trust of the groups included in the invitation – potential dangerousness of the newly invited member. The splitting makes the potentially malicious users isolated in groups with lower level of trust. The important point here is, that the loyal users that triggered splitting, remain in the group with the lower level of trust as well. In this way, the SecGrid model maintains knowledge (history) about potentially dangerous users. Moreover, the malicious users have no direct knowledge about the splitting. Whereas the splitting creates a new group with a new tKey from possibly only non-malicious members, the possibly malicious ones remain in the old group with no change to their tKeys. Thereby the malicious users have no direct way to reveal having been detected as possible threats.

The SecGrid model resistance against traitors is getting stronger with groups of users getting more *interconnected* – with non-empty intersections. The larger intersections mean more users being able to detect and reveal traitors to the others (see experiments section 7.3.1).

6.6.2 Collusion

In the SecGrid model, a collusion equals to the fact that some users try to infiltrate trusted groups and then cause some harm. The SecGrid model copes with this attack very naturally by the *shadow groups* and the parameter α. If a member would like to invite a (malicious) user into a group, he/she must issue a new *base*. Moreover, the invited user cannot do any harm until he/she has received requested number of signatures from the current full privilege members of the corresponding group. In addition, the members of the group can through the *base* field reveal that the member has systematically been inviting new users or that some members have signed *tKeys* by the same members – collusive members.

Note that, as in the case of traitors, malicious users are not rejected from the group, but the well-behaving members create a new group leaving the malicious ones behind keeping the knowledge about the malicious ones in the old group.

We consider this feature of the SecGrid model as important, as the collusion is a very severe and hard to solve attack for most reputation systems.

6.6.3 Front peers

The SecGrid model copes with front peers very naturally by the *shadow groups* and the parameter α in the same manner as in the case of collusion.

6.6.4 Whitewashers

This kind of attack is a real danger to the SecGrid model as some kind of identification (*fingerprint*) is required. Fortunately, the same system as for *tKeys* can be used for issuing identification. A user generates its own credential and until such credential has not been signed by required amount of full privilege members, it remains invalid (in some point view, such system is similar to PGP [67]). This technique also enables members to reveal a group of malicious users in the same manner as in the case of collusion.

It is worth noticing that such functionality can be implemented in the same way as *tKeys*, thus requiring no additional tools allowing full privilege members to generate new credentials if needed, thus fulfilling the requirement for no long-time secrets.

6.7 Personalization

Personalization is a very important feature of the SecGrid model allowing members to drive their own security policies. Among the parameters for personalization belong:

- *parameter* α - even though the parameter was consider to be common for the whole group, each member can handle this parameter differently. This corresponds to a different level of confidence of group members to novices.
- *level of trust* - even the level of trust field can be handled differently by different members.
- *TTL value* - personalization of this attribute allows current members to treat novices differently (e.g. to reduce time for collecting signatures)

Chapter 7

Experimental Results

The proposed security model SecGrid comprises several parts;

- the G2H and SD Algorithms, and
- the security sub-system.

The experimental results presented in this section follow the same structure.

7.1 G2H Algorithm Experiments

The G2H Algorithm transforms a general graph input describing relationships between set of users into the hypergraph model of SecGrid. The main purpose of the experiments for the both versions of the G2H Algorithm was to verify their usability.

The input graphs used for the experiments can be generally divided into two main categories:

1. *manually created* graphs,
2. *testing graphs describing a real societies.*

The main purpose of the manually created graphs is the verification of the behavior of the G2H Algorithm against cases where the correct output can be verified by a human user. The second category examines behavior of the algorithm on data describing concrete parts of reality.

7.1.1 Manually Created Input Graphs

The main aim of the first series of experiments was to verify the ability of both modifications of the G2H Algorithm to transform correctly basic input graphs.

The first example (Figure 7.1.1) contains 13 users with relationships denoted by oriented edges. In this case, reader can easily identify 3 groups of users, particularly:

- *group 1* with users *5,6,7,8,9*,
- *group 2* with users *10,11,13*,
- *group 3* containing users *1,2,3,4,12*.

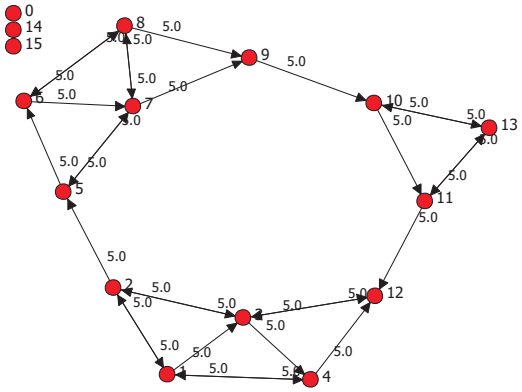


Figure 7.1.1: An Input Graph.

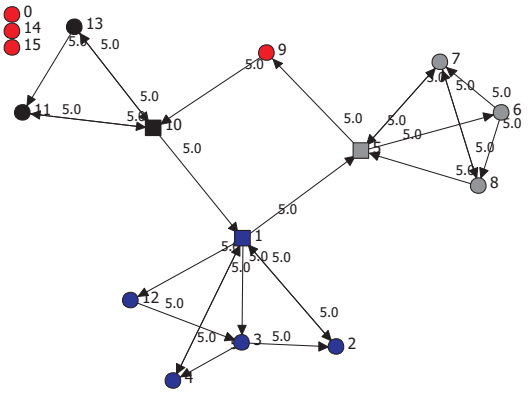


Figure 7.1.2: Groups Identified by the G2H Algorithm (based on strongly connected components).

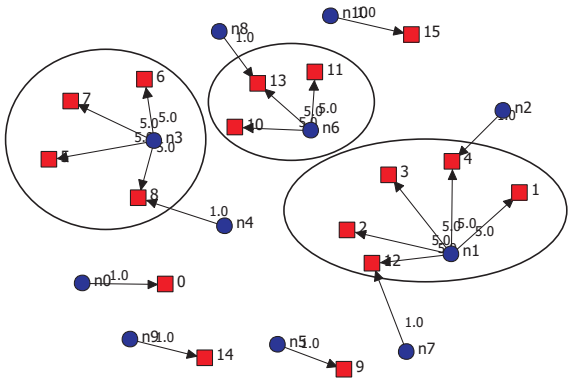


Figure 7.1.3: Groups Identified by the G2H Algorithm (based on triads).

In Figure 7.1.2 are shown groups created by the G2H Algorithm *based on strongly connected components*, and groups created by the G2H Algorithm *based on triads* in Figure 7.1.3. In Figure 7.1.2 users of the same group are shown in the same color. From the figure it is clear, that resulted groups correspond exactly to the expected results. In the case of the G2H Algorithm *based on triads* (Figure 7.1.3) members (denoted by red squares) of the same group are interconnected by the same hyperedge (denoted by a blue circle and bounded by an ellipse). Even in this case, groups are identified accordingly to the expectations.

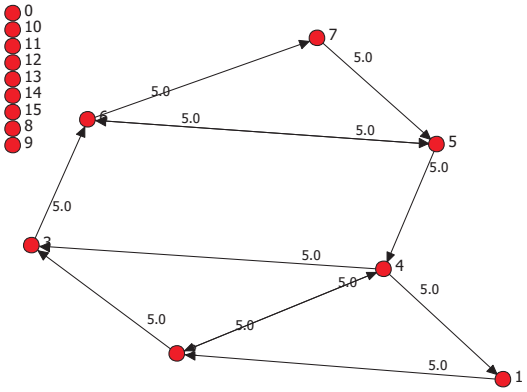


Figure 7.1.4: An Input Graph.

The second example of the input graph is given in Figure 7.1.4. This input graph poses a real problems to the G2H Algorithm *based on strongly connected components* (Figure 7.1.5). In this example input, there are two sets of users that have direct relationships; *5,6,7* and *1,2,4*. Note that user *3* does not have direct (bilateral) relationships with any user, thus he/she cannot be added into any group. In this case, only the G2H *based on triads* (Figure 7.1.6) provides

reasonable grouping. This is due to the fact, that the input graph contains only one strongly connected component including all users resulting in creation of one group from all users in the case of the G2H Algorithm *based on strongly connected components*.

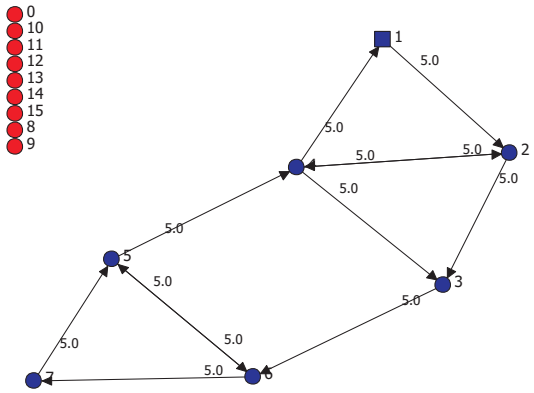


Figure 7.1.5: Groups Identified by the G2H Algorithm (based on strongly connected components).

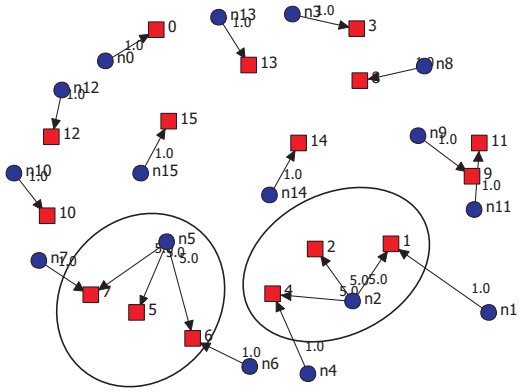


Figure 7.1.6: Groups Identified by the G2H Algorithm (based on triads).

The second example of input graph shows the main disadvantage of the G2H Algorithm *based on strongly connected components* – in most real input graphs, users are highly interconnected (graphs are often dense), thereby probability that a few strongly connected components containing majority of users is high.

7.1.2 Real Input Graphs

In the second series of experiments, input graphs describing a real societies were examined, particularly:

1. EIES social network structure [71]. EIES is broadly used for benchmarking algorithms for social network analysis (SNA),
2. records of calls realized in a mobile network in the Slovak republic.

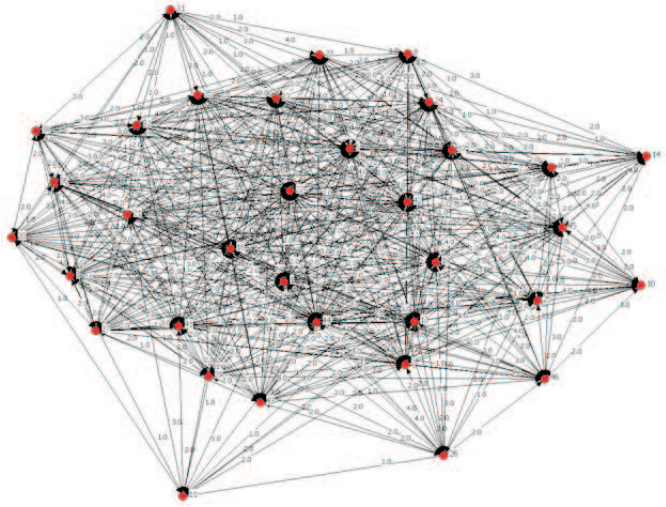


Figure 7.1.7: Input Graph for EIES Experimental Data.

In Figure 7.1.7 is shown the EIES input graph. It is clearly visible that the users are highly connected.

Figure 7.1.8 shows an output of the G2H Algorithm *based on strongly connected components*. In this case, the **main disadvantage** mentioned previously shows up and all users are transformed as members of the same group.

In the following Figure 7.1.9 result of the G2H Algorithm *based on triads* is shown. In this case, the input structure is divided into several groups (connected by the same hyperedges denoted as blue circles).

In the second experiment real data from a mobile operator from the Slovak republic were taken as an input. The input graph consists of 7898 vertices and 8609 edges[1] (note that many vertices are isolated). In this case, only the G2H Algorithm *based on triads* is used for experiments, for the G2H Algorithm *based on strongly connected components* identifies only **one group** with all users within (apart from isolated ones). The resulted grouping given by the G2H Algorithm *based on triads* consists of 112 groups (hyperedges). In fact the majority of vertices is isolated therefore not included in grouping.

[1]For the size of the input graph we do not provide graphical representation as it would be of no legibility.

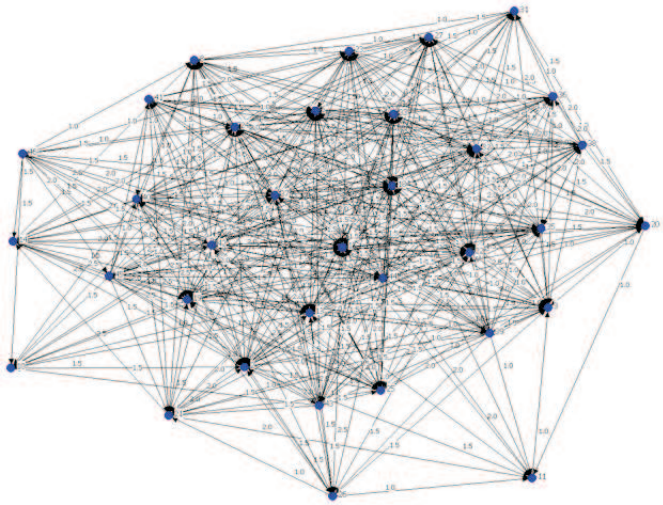


Figure 7.1.8: The Output of the G2H Algorithm (based on strongly connected components).

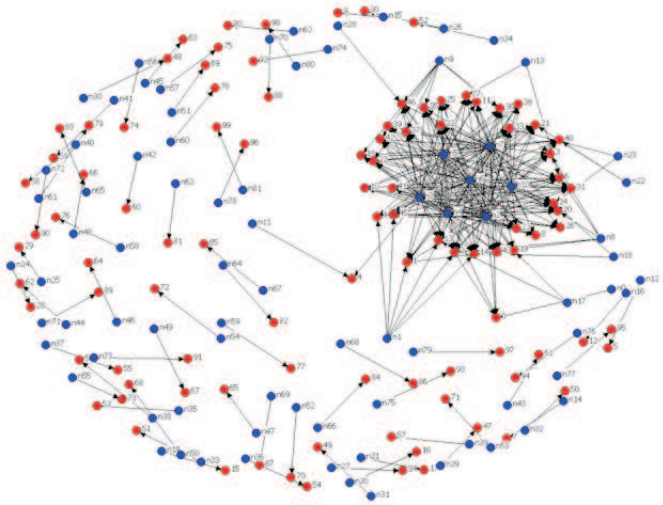


Figure 7.1.9: The Output of the G2H Algorithm (based on triads).

Distribution of the users into groups is shown in Table 7.1.1. The most groups are of size 2 followed by groups of size 3, 5 and 6. The G2H Algorithm *based on triads* therefore tends to create smaller groups with most of them created from complete graphs of size 2 (it corresponds to the seeds identified by this version of the G2H Algorithm). With respects to the results presented in the SD Algorithm experiments (see section 7.2), it can be asserted that starting configuration of groups with rather smaller groups provided by the G2H Algorithm *based on triads* is a good starting point for the SD Algorithm. In other words, it is safer to create smaller groups.

Table 7.1.1: G2H Algorithm Results - Distribution of Users Between Groups

Group size	**Amount of groups of this size**
2	104
3	6
5	1
6	1

Summarizing the experiments:

1. G2H Algorithm *based on strongly connected components* has lower time complexity, but it is unable to cope with a real input data.
2. The G2H Algorithm *based on triads*, on the contrary, can cope with dense as well as sparse inputs, but has rather higher complexity.

It is worth noticing that other techniques known in SNA [68] can be used for identification of groups. On the other hand, as the G2H Algorithm only prepares an input for the heart of the SecGrid model – the SD Algorithm realizing the dynamics. As the grouping provided by the G2H Algorithm *based on triads* is sufficient we did not pursue further experiments in the thesis.

7.2 SD Algorithm Experiments

The main aim of experiments for the SD Algorithm was to verify its **stability**.

The stability stands for the *ability of the SD Algorithm to cope with dynamic changes in the structure of groups of users without creating*:

1. **One group containing all users in the system**.
   As the SecGrid model derives level of trust between users based only on group memberships, if all users (malicious and "good" ones) are members of only one group, each and very pair of users in the system shares the same level of trust.
2. **Many very small groups** (*containing typically one or two users*).
   In such a case the SecGrid model has consequently very limited ability to infer trust between users, while most of them share no common group.

Behavior of the SD Algorithm can be generally driven by parameters λ and ϵ, where :

- λ *influences* **joining of groups** – the higher λ the more joining,
- ϵ *controls* **splitting of groups** – the lower ϵ the higher probability of splitting.

The stability was investigated by two sets of experiments:

1. input data of a cellular network operator from the Slovak republic,
2. input data for three different statistic distributions of random numbers representing invitation issued between users.

7.2.1 Cellar Network Input Data Experiments

The input to the SD Algorithm is extracted from a database containing records of calls made in a cellular network in the Slovak republic. Records are stored in the database as quartets (*recipient*, *sender*, *type of the request*, *duration*). For the experiment we extracted 161 404 phone calls between 121 672 users for the same type of request.

Results presented here show behavior of the SD Algorithm for three different combinations of the parameters λ and ϵ. For the sake of clarity, let us give a list of parameters and expected behavior of the SD Algorithm:

- $\lambda = 1$, $\epsilon = 1$ leading to *less joins and more splits*.
  This combination suggests that splitting of groups will override the joining of groups, thus leaving a system with rather smaller groups behind.
- $\lambda = 1$, $\epsilon = 3$ leading to *less joins and less splits*.
  In this case, at the beginning of the evolution users are simply added into groups, which suggest a system with rather bigger groups.
- $\lambda = 3$, $\epsilon = 1$ leading to *more joins and more splits*.
  This case starts from the very beginning by joining groups. Even thought splitting is also stimulated, the joining will provide more significant changes as splitting leaves users out of intersections between groups untouched. Therefore, in this combination a system with larger groups is expected.

At the beginning an initial system of 908 groups[1] (*starting amount of groups* will be in the rest of the thesis denoted as Ω) each containing 134 users was randomly created with an equal level of trust.

[1]We chose this number as it is common multiple close to 1000 giving reasonable large groups.

Description of the Experiment: During the experiments records are fetched from the database and put as the input to the SD Algorithm. Each record in the input expresses that the *recipient* invites the *sender* from randomly chosen sender's group to randomly chosen recipient's group. For the input data does not include information about grouping (the input data represents communication between users) we had to choose the groups randomly. Each invitation represents one cycle (iteration).

For better readability of the results, the evolution of the system of groups is shown as histograms representing relative frequency of groups (in percents) as a function of size of groups. We use the size of classes for histograms 20.

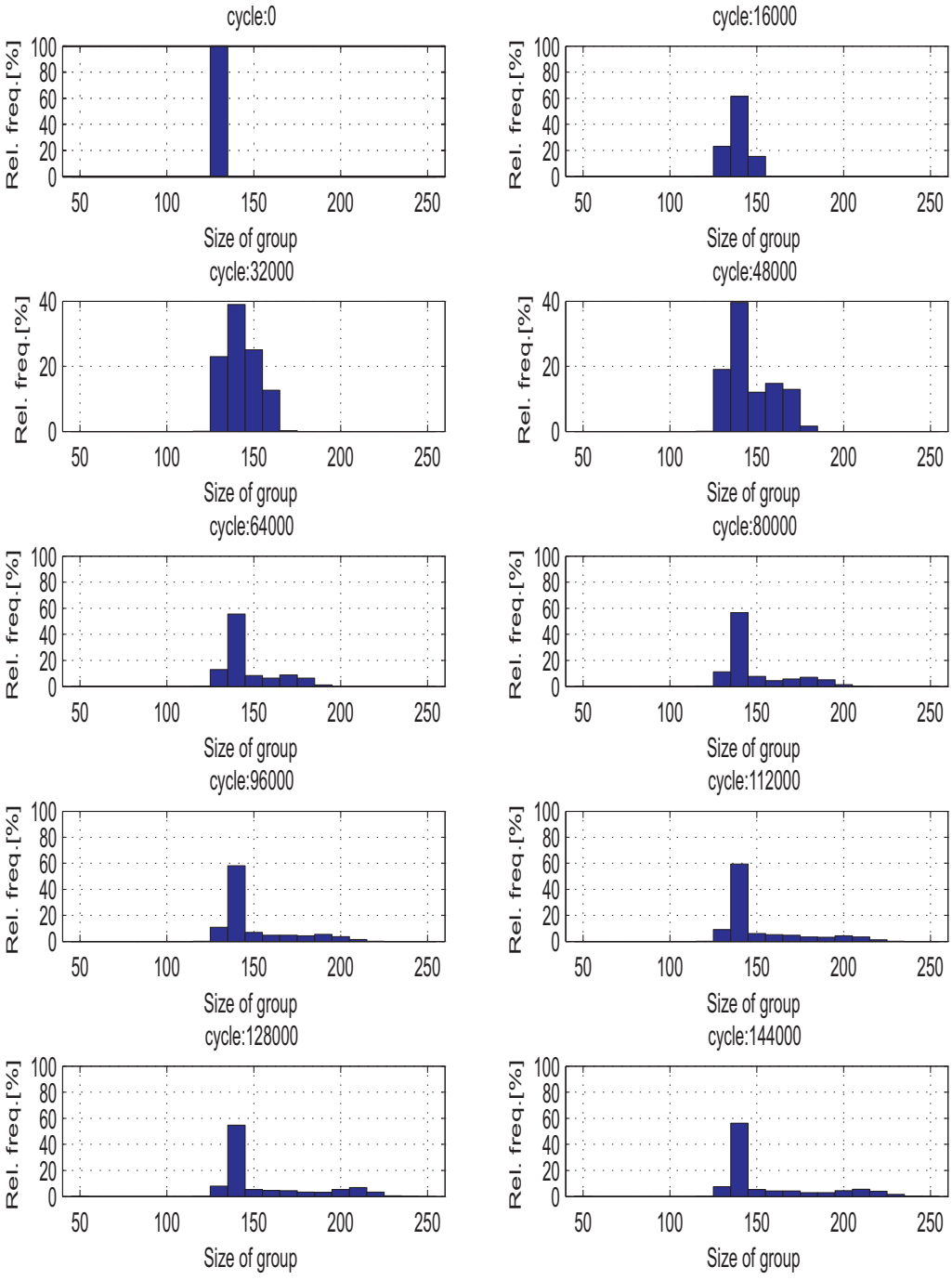


Figure 7.2.1: Histograms for $\lambda = 1$, $\epsilon = 1$, $\Omega = 908$

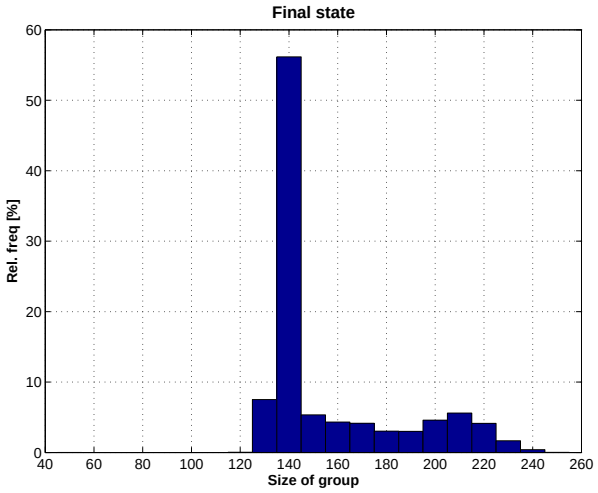


Figure 7.2.2: The Final Histogram ($\lambda = 1$, $\epsilon = 1$, $\Omega = 908$)

In the first combination of parameters (Figure 7.2.1), as well as in the other cases, the first histogram (*cycle* : 0) represents starting configuration – 100% of groups contain 134 users. With increasing number of processed cycles (histograms from from left to right and top to bottom) distribution changes. In the first three histograms (*cycles* : $16000, 32000, 48000$), changes are easily visible. With more cycles processed changes to the system are more subtle. The final state (Figure 7.2.2) represents the fact that more than 50 % of groups contain around 140 users. Most groups created are larger than starting configuration with a peak for 210 members per group and maximum at 240 members per group. A final resume is given at the end of this section.

Next series of histograms (Figure 7.2.3) represent evolution for parameters $\lambda = 1$, $\epsilon = 3$. As was mentioned previously, this configuration suggests less splits and less joins with many adding at the beginning. The result of initial adding is clearly visible in the histogram for *cycle* : 16000 where growth of groups of size 150 is mostly stimulated (compared to the previous case). With the increasing amount of processed cycles, the system of groups tends to achieve a stable configuration as probability of splitting and joining increases. The final histogram (Figure 7.2.4) presents very similar distribution of users into groups as in the previous case, apart from lower percentage of group of size 130 and more larger groups, two smaller peaks at 170 and 200 and maximum at 240 members per group.

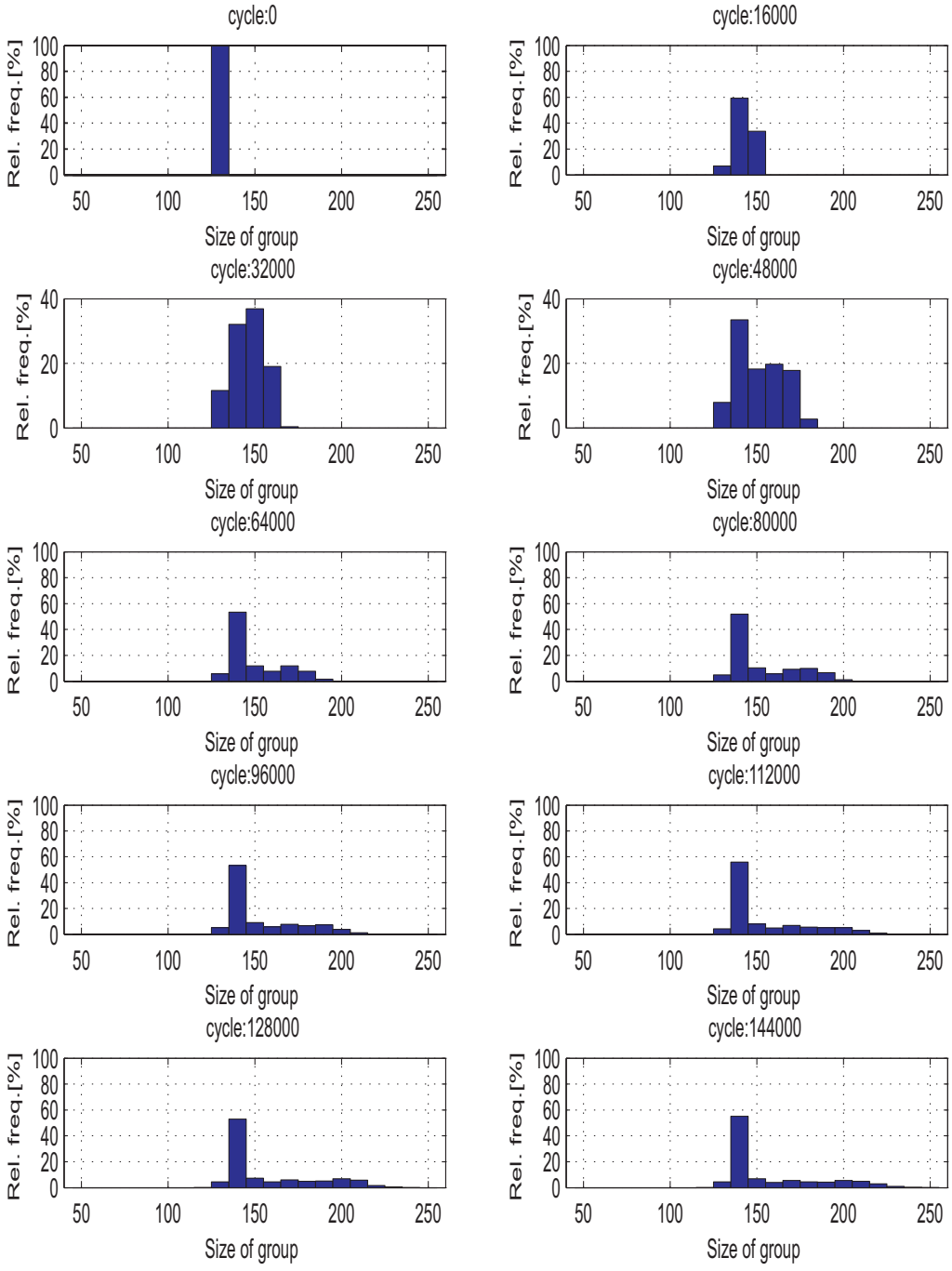


Figure 7.2.3: Histograms ($\lambda = 1$, $\epsilon = 3$, $\Omega = 908$)

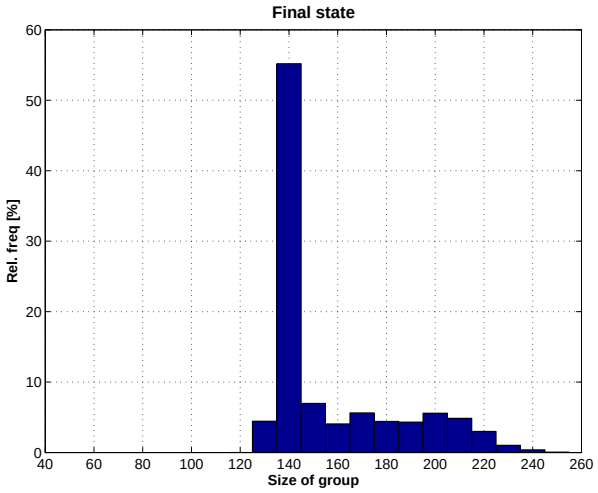


Figure 7.2.4: The Final Histogram (λ =1, $\epsilon = 3$, $\Omega = 908$)

The last combination of parameters presented in Figure 7.2.5 shows the simulation for the parameters $\lambda = 3$, $\epsilon = 1$. Under this configuration, the SD Algorithm should tend to process more joining of groups. It is clearly visible in histograms for *cycle* : $16000, 32000, 48000$ where percentage of groups of the lowest size (130) is the minimum of all tested combinations of the parameters. Nevertheless, with increased number of cycles the system tends to achieve a stable configuration as well as in the previous cases. In the final histogram shown in Figure 7.2.6 the joins cause creation of larger groups, which is clearly visible for sizes of groups 200 up to 220.

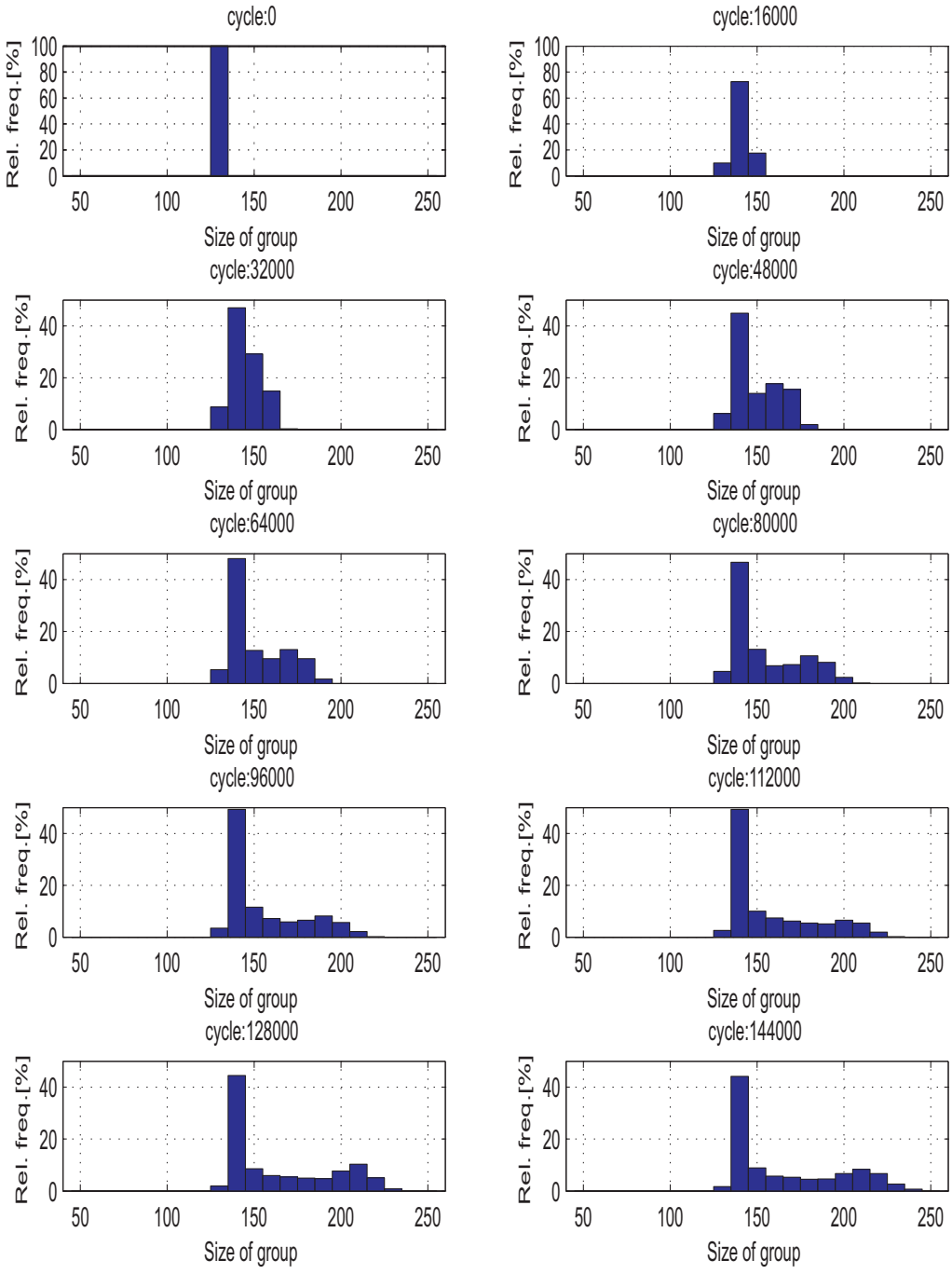


Figure 7.2.5: Histograms ($\lambda = 3$, $\epsilon = 1$, $\Omega = 908$)

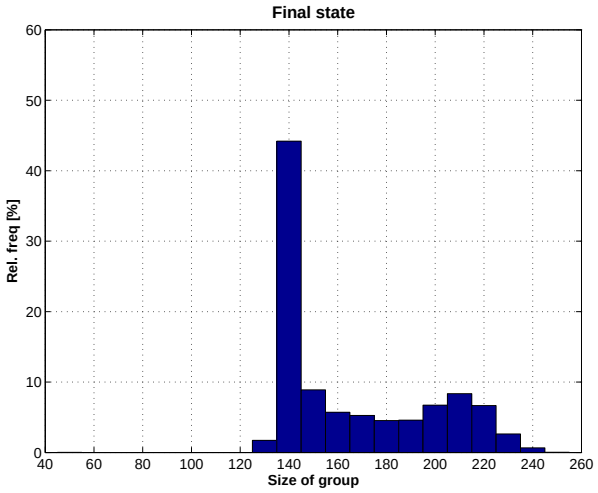


Figure 7.2.6: The Final Histogram ($\lambda = 3$, $\epsilon = 1$, $\Omega = 908$)

Discussion of the Experiments: The SD Algorithm has been shown to be stable against the data describing a real society. The stability seems to be only gently affected by the parameters at the end, but the parameters play important rule in the beginning of the evolution.

- In the case of low λ and low ϵ ($\lambda = 1$, $\epsilon = 1$ in Figure 7.2.2), the first several cycles create a system of rather smaller groups. This is due to the more splits. Even in the final histogram (Figure 7.2.1) it is visible the highest distribution of users toward smaller group for all combinations presented.
- The second combination $\lambda = 1$, $\epsilon = 3$ (Figure 7.2.4) creates at the beginning a system with larger groups, as expected.
- The last combination $\lambda = 3$, $\epsilon = 1$ tends to preserve the starting configuration least of all combinations. This is mainly due to the high probability of joining and splitting at the beginning.

In general, it can be said that the SD Algorithm is stable. Our experiments have shown that round 50% of users remain very close to the starting configuration. This feature fully supports the requirements from sub-section 6.2 on a conservative system that makes slow changes in the system of groups.

7.2.2 Input Data for Different Statistical Distributions

The experiments presented previously illustrated the ability of the SD Algorithm to preserve a stable system of groups. In the following set of experiments the SD Algorithm is investigated for existence of any dependency between stability and a type of input.

The experiments use randomly generated input data for the following statistical distributions of invitations between users:

1. uniform distribution,
2. normal distribution,
3. exponential distribution.

The parameters of the SD Algorithm were chosen ($\lambda = 2$, $\epsilon = 1$), for this configuration seems to be balanced (we used more different combinations of the parameters but without a noticeable influence on the results). Results are also shown in histograms with the same meaning of axes as previously.

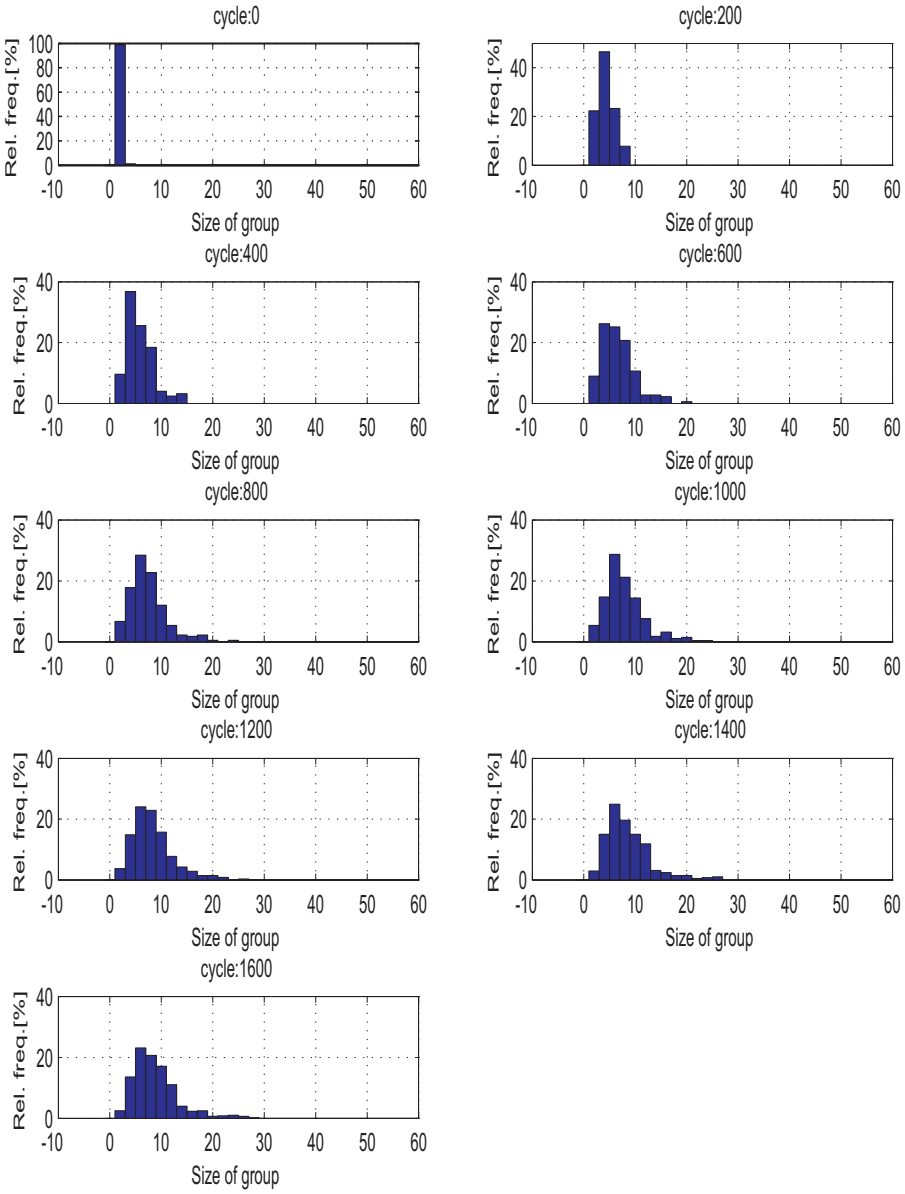


Figure 7.2.7: Histograms for Uniform Distribution ($\lambda = 2$, $\epsilon = 1$, $\Omega = 20$).

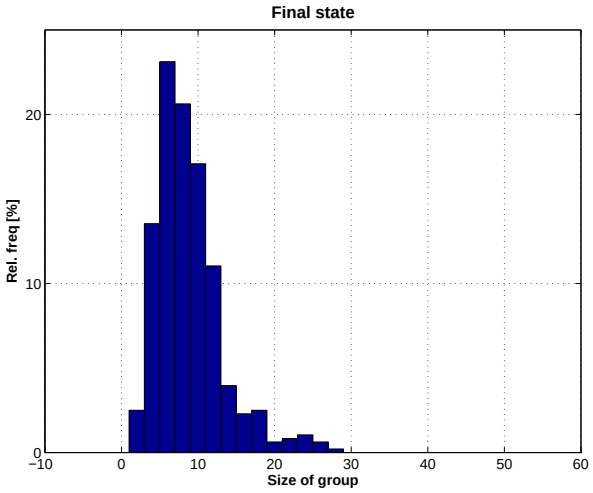


Figure 7.2.8: The Final Histogram for Uniform Distribution ($\lambda = 2$, $\epsilon = 1$, $\Omega = 20$).

Uniform Distribution

Uniform distribution (Figure 7.2.7) of random input data corresponds to randomly issued invitations between users. Due to the uniformity of this distribution, during the first cycles (the histogram for $cycle : 200$) there is a lot of adds – invitations are spread onto many users creating most groups of size 2. With increasing amount of cycles processed, splits and joins override adding resulting in stabilization of the system of groups. The final histogram (Figure 7.2.8) shows that invitations generated randomly with uniform distribution tends to preserve the starting configuration the least of all cases (see the next sub-sections).

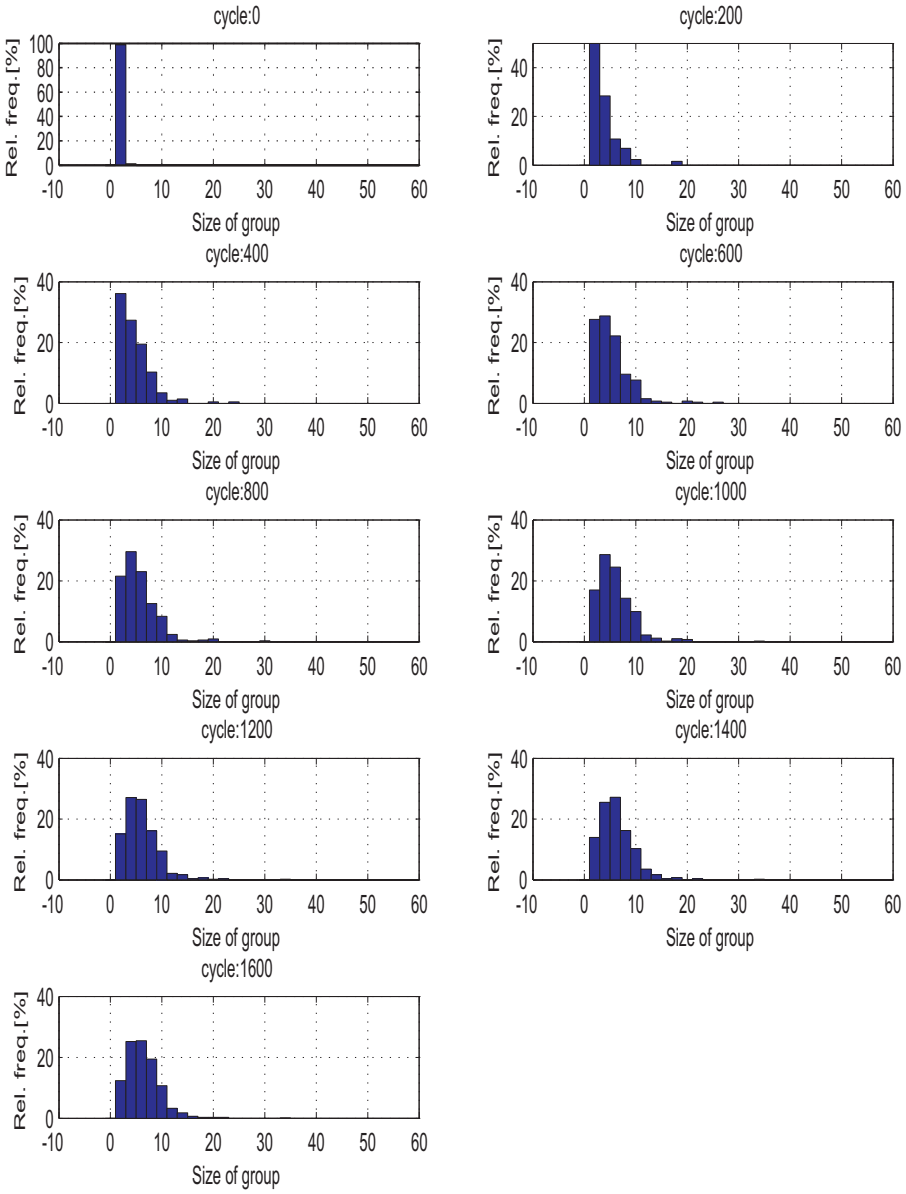


Figure 7.2.9: Histograms for Normally Distribution ($\lambda = 2$, $\epsilon = 1$, $\Omega = 100$).

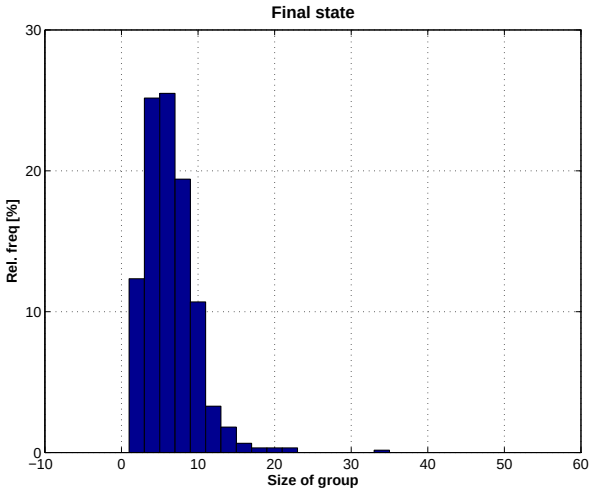


Figure 7.2.10: The Final Histogram for Normal Distribution ($\lambda = 2$, $\epsilon = 1$, $\Omega = 100$).

Normal Distribution

The normal distribution is used as a model of quantitative phenomena in natural and behavioral sciences due to the fact that many psychological measurements and physical phenomena can be approximated by the normal distribution. This is the main reason why we chose this distribution for the verification of the SD Algorithm.

Input data generated for normal distribution places most invitations onto a limited set of users, thus creating a system with groups having larger intersections (many common users). This fact is clearly visible from histograms in Figures 7.2.7 (the uniform distribution) and 7.2.9 (normal distribution), where histograms for *cycle* : 200 differs in much. Particularly, the uniform distribution creates many new groups with size 2, whereas normal distribution creates larger groups with descending tendency. The following histograms in Figure 7.2.9 shows that increasing amount of cycles causes stabilization of the groups.

Nevertheless in the final histogram (Figure 7.2.10) is clearly visible that in the case of the normal distribution of input data, the SD Algorithm preserves the starting configuration more than in the case of the uniform distribution. In addition to this the system contains less larger groups than in the case of the uniform distribution (see for instant interval 20-30) . This is due to the input data, that places many invitations onto a limited number of users. This results in system of groups where a few users are members of many groups, while the others remain in the starting configuration (they have not received any or few invitations).

Exponential Distribution

The last distribution chosen for the experiments is the exponential distribution. In this case, invitations are generated for even more limited number of users. Histograms in Figure 7.2.11 show very similar evolution of groups as in the case of the normal distribution presented above.

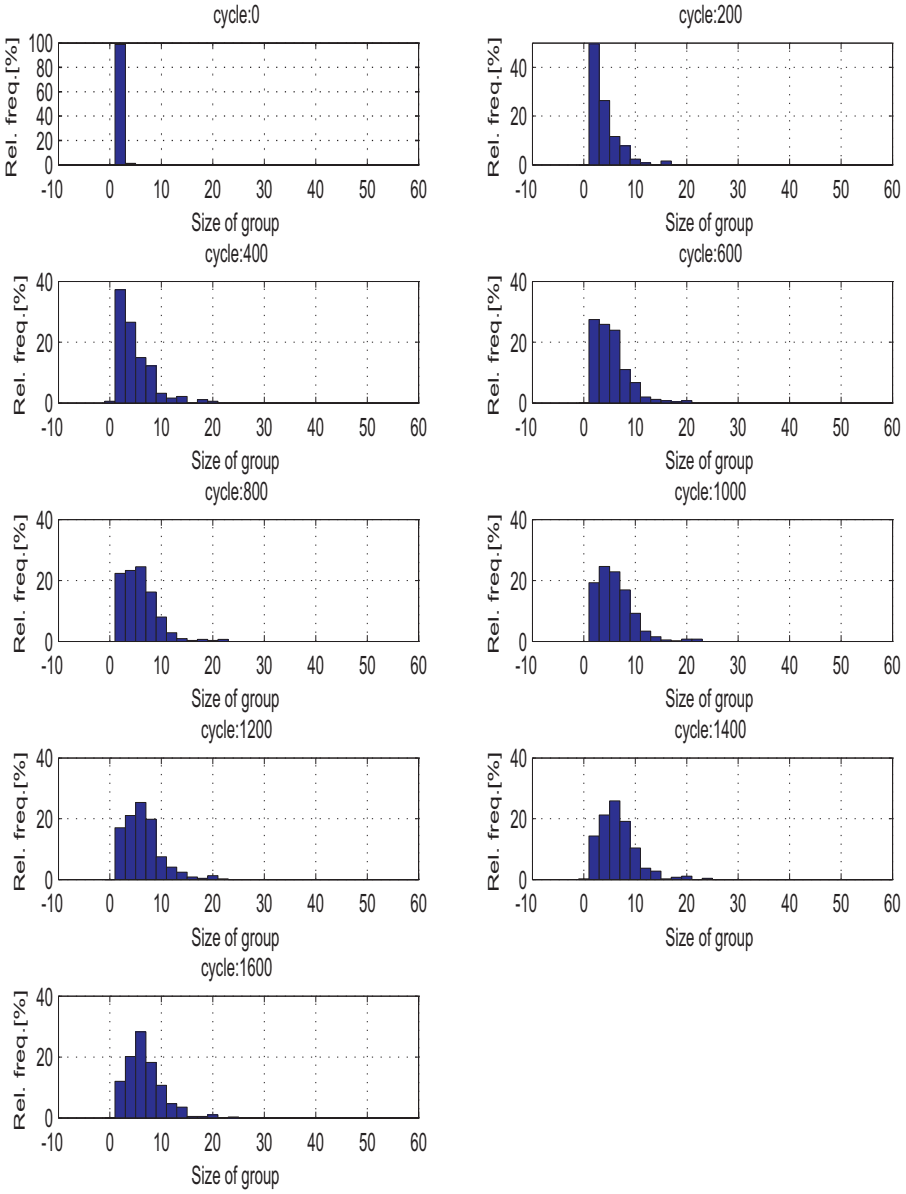


Figure 7.2.11: Histograms for Exponential Distribution ($\lambda = 2$, $\epsilon = 1$, $\Omega = 10$).

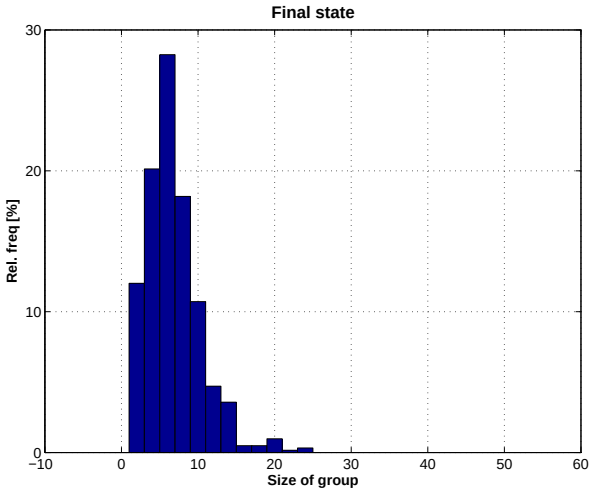


Figure 7.2.12: The Final Histogram for Exponential Distribution ($\lambda = 2$, $\epsilon = 1$, $\Omega = 10$).

The final histogram of distribution of users into groups in Figure 7.2.12 shows very similar distribution to the case of the normal distribution. The mostly visible difference is the percentage of frequency of groups of size 4 and 6. In the case of exponential distribution there is more groups of size 6. That exponential distribution generates invitation for even more limited number of users than in the former case, which leads to the more joins at the beginning of the evolution (see Figure 7.2.11).

The experiments presented in this section illustrate that the input has an impact on the distribution of users into groups. Apart from subtle different final distributions, these experiments illustrate the SD Algorithm to be stable even in the case of different input data. The mostly visible difference of the evolution and the final distribution of users into groups in the case of normally distributed invitations.

7.3 Experiments for Security Sub-system

The precedent experiments concentrated on the stability of the SecGrid model, particularly of the SD Algorithm. The SecGrid model, on the other hand, contains also a security component. Therefore, this section presents the test of the security aspects of the SecGrid model together with properties of the hypergraph model.

7.3.1 Overlapping of Groups

The SecGrid model uses the hypergraphs for the representation of a system of groups. We argued that the model provides better support for building direct trust between users (see section 4.2.2). In our hypergraph model users from the same group have a direct trust relationship with each other. In the graph model users have direct trust relationship only if they have been in direct contact. In the following experiments we investigate differences between direct trust relationships in graph and hypergraph model.

Amount of users having direct relationships equals in graph model to the size of the set of edges ($|E|$). In hypergraph model, on the other hand, amount of direct relationships is given by amount of edges of complete graphs[1] constructed for each hyperedge. Note that number of directed edges constructed for a complete graph with n vertices is given by:

$$|E| = n \cdot (n-1) \tag{7.3.1}$$

In addition, one vertex can be a pin of more hyperedges. Hereby, it is possible that one vertex contributes to more complete graphs representing hyperedges.

For better understanding of the experiment, let us firstly explain what happens during the splitting and joining phases of the SD Algorithm. In Figure 7.3.1 case a) an example of two groups being connected by two users E and B is shown. Assume that combination of group weights and parameter ϵ triggers the splitting of groups. The possible result of splitting procedure is depicted in Figure 7.3.1 case b), which now consists of three groups. It is clearly visible that intersections between groups are now larger including 7 users (E, B, I, J, H, D, G) instead of two users E, B in case a).

[1] This corresponds to *clique-net representation of hyperegraphs* used in numerical mathematics [69],[70].

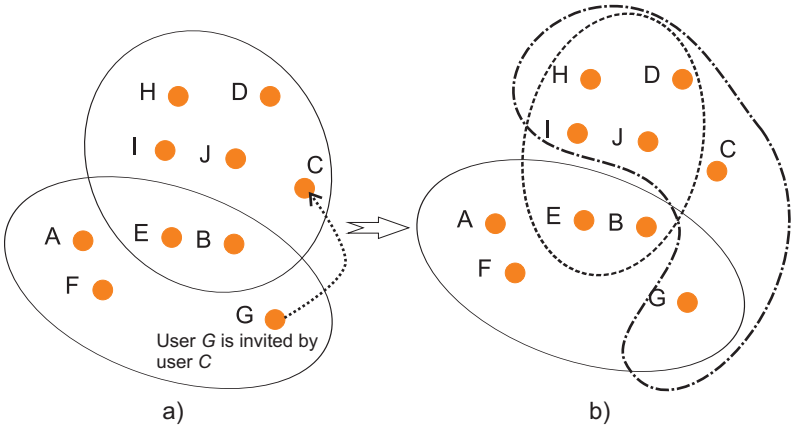


Figure 7.3.1: Contribution of the Splitting Procedure (to enlargement of the set of direct relationships between users in the hypergraph model).

case a)	case b)	
$5.4 = 20$	$6.5 = 30$	(7.3.2)
$7.6 = 42$	$6.5 = 30$	(7.3.3)
	$5.4 = 20$	(7.3.4)
$- - - - -$	$- - - - -$	
62	80	(7.3.5)

In expressions 7.3.2-7.3.5 are given calculations of direct relationships between users for examples given above in Figure 7.3.1. In the first column a calculation of number of edges for complete graphs created for hyperedges in case a) is given. As there are two hyperedges with 5 and 7 vertices, total number of edges is 62. In case b), on the contrary, there are 3 hyperedges; two with 6 and one with 5 vertices – three complete graphs with totally 80 edges. In this example, the splitting added 18 new direct relationships.

The same can be asserted for the case of joining (shown in Figure 7.3.2). In case a) there are two groups with 7 and 5 users, respectively. By joining of these groups one group containing all (10) users is created. Equations 7.3.6-7.3.8 show the calculations illustrating the enlargement of the set of direct relationships.

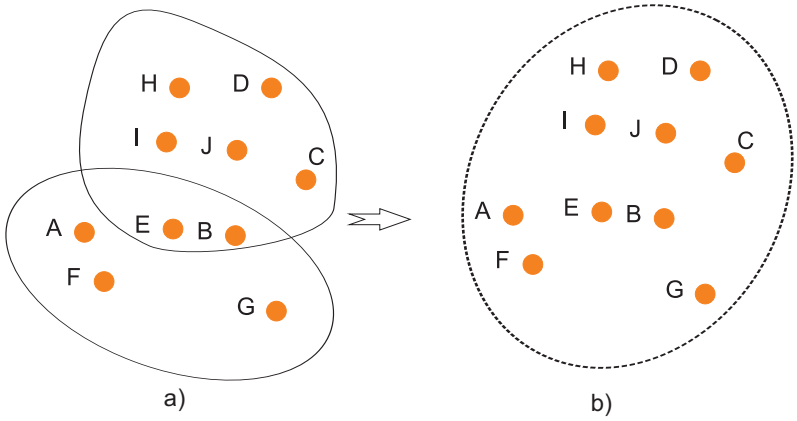


Figure 7.3.2: Contribution of the Joining Procedure (to the enlargement of the set of direct relationships between users in the hypergraph model).

case a)	case b)	
$5.4 = 20$	$10.9 = 90$	(7.3.6)
$7.6 = 42$		(7.3.7)
$- - - - -$	$- - - - -$	
62	90	(7.3.8)

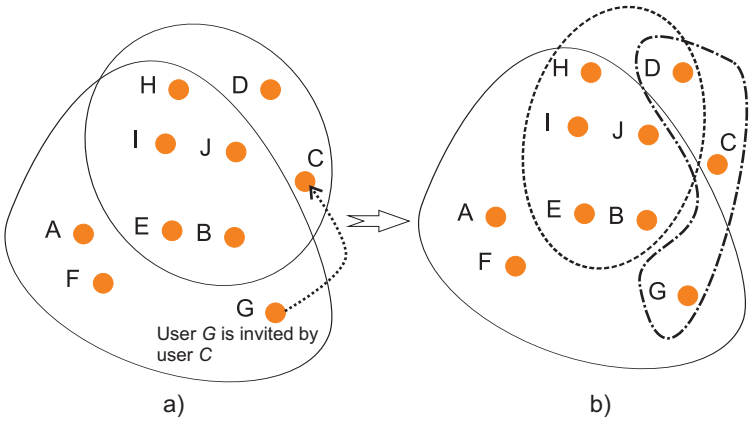


Figure 7.3.3: Contribution of the Splitting Procedure (to the decreasing of the set of direct relationships between users in the hypergraph model).

On the other hand, one might object that this is not valid for all cases. For example in Figure 7.3.3 a situation is shown where initial configuration of two groups (case a)) is split into three groups (case b)). In this case the splitting does not contribute but counteract to the number of direct relationships. Expressions 7.3.9-7.3.12 show that the total amount of direct relationships is reduced by 6.

$$\begin{array}{rll} \text{case a)} & \text{case b)} & \\ 8.7 = 56 & 8.7 = 56 & (7.3.9) \\ 7.6 = 42 & 6.5 = 30 & (7.3.10) \\ & 3.2 = 6 & (7.3.11) \\ - - - - - & - - - - - & \\ 98 & 92 & (7.3.12) \end{array}$$

That joining might also result is the decrease of direct relationships is straightforward. For example, assume configuration given in Figure 7.3.4 where given groups differ only by one user G. Joining of groups results in one group with 10 users. In this case, as shown by expressions 7.3.13-7.3.15, the decrease is by 72 direct relationships.

$$\begin{array}{rll} \text{case a)} & \text{case b)} & \\ 10.9 = 90 & 10.9 = 90 & (7.3.13) \\ 9.8 = 72 & & (7.3.14) \\ - - - - - & - - - - - & \\ 162 & 90 & (7.3.15) \end{array}$$

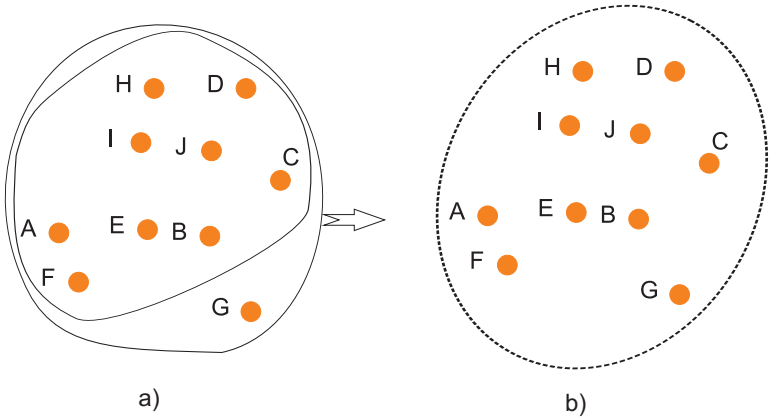


Figure 7.3.4: Contribution of the Joining Procedure (to the decrease of the set of direct relationships between users in the hypergraph model).

Based on the examples discussed above, it can be stated that:

1. **larger intersections** between groups – splitting as well as joining cause the **lessening** of the total amount of direct relationships,
2. **smaller intersections** between groups – splitting as well as joining cause the **enlargement** of the total amount of direct relationships.

Thereby, it would be advantageous for the clarity of the experiments to know relations between parameters of the SD Algorithm and the size of intersections:

1. $\downarrow \lambda, \uparrow \epsilon$ – **less** joins and **less** splits cause **larger** intersections,
2. $\uparrow \lambda \downarrow \epsilon$ – **more** joins and **more** splits cause **smaller** intersections,

where $\uparrow$ represents larger value, and $\downarrow$ represents smaller value of the parameters.

In the following sets of figures, results for experiments illustrating the precedent statements are presented. The SD Algorithm is tested for various input data and with different combinations of the parameters. Concretely, input data are generated by uniform, normal and exponential distributions and the combinations of parameters are:

1. $\lambda = 1, \epsilon = 1$;
2. $\lambda = 1, \epsilon = 3$;
3. $\lambda = 3, \epsilon = 1$.

The input for the SD Algorithm comprises 2000 pairs $(user1, user2)$ corresponding to an invitation issued by $user1$ for $user2$ from randomly chosen $user2$ group to randomly chosen $user1$ group. The starting configuration is created as each user has its own group and levels of trust are the same.

In the following figures graphs for all three combinations of parameters are given (the amount of direct relationships is represented on y-axis and cycles are on x-axis).

For uniform distribution of input data, the resulted graphs are shown in Figures 7.3.5, 7.3.6 and 7.3.7. Graphs show dependency of number of direct relationships of the graphs model (shown in red dash-dot-line), number of direct relationships in SecGrid with the hypergraph model (in magenta dash-and-dot line), and also SecGrid with the bare hypergraph model (in green doted line). The bare SecGrid hypergraph model represents amount of direct relationships between users, where each group (hyperedge) is represented as a complete graph and each pair of vertices is connected by at most one edge. In this model, a user can contribute to the overall number of direct relationships only in one of group (this model is presented for comparisons).

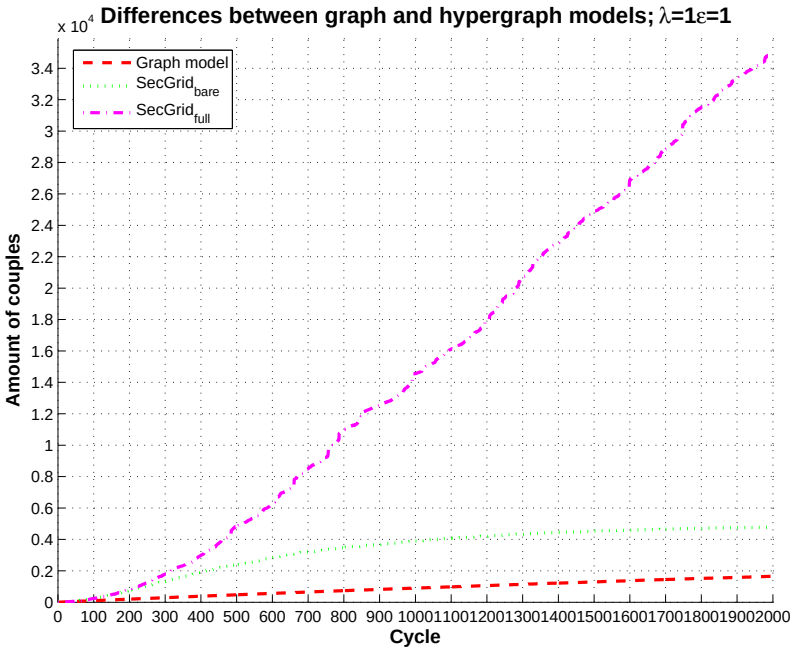


Figure 7.3.5: Uniform Distribution ($\lambda = 1$, $\epsilon = 1$).

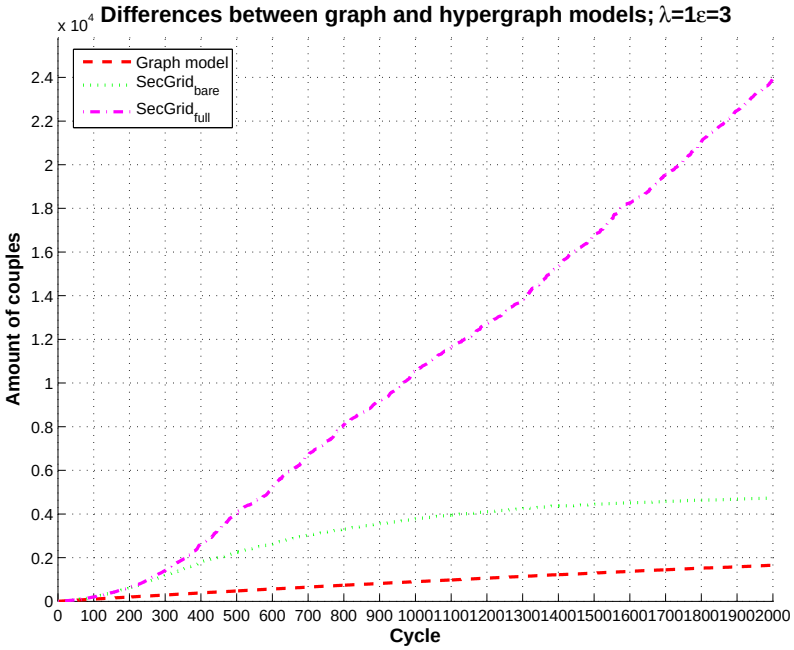


Figure 7.3.6: Uniform Distribution ($\lambda = 1$, $\epsilon = 3$).

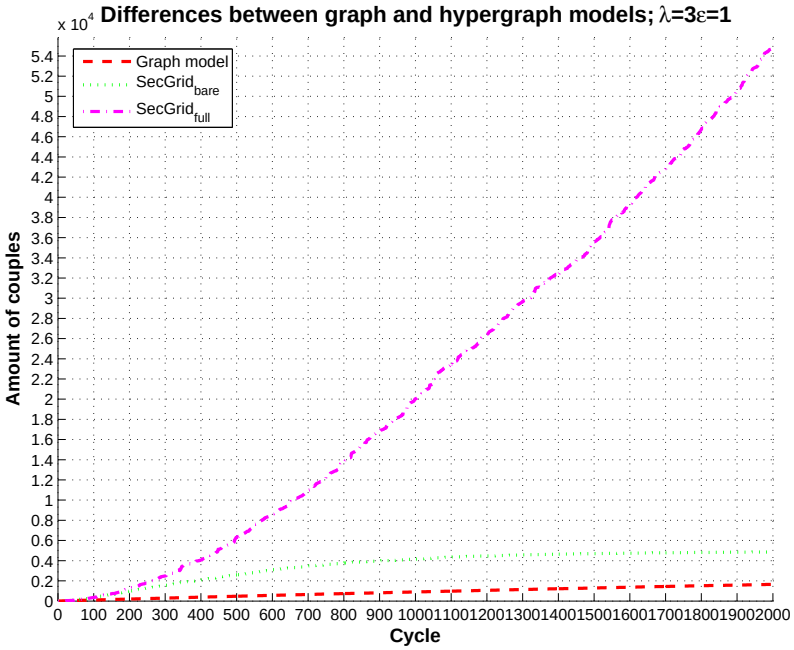


Figure 7.3.7: Uniform Distribution ($\lambda = 3$, $\epsilon = 1$).

In the first three graphs (7.3.5,7.3.6,7.3.7) it is clearly visible that the graph model offers the lowest number of direct relationships. The full SecGrid model,

on the other hand, surpasses the graph model dramatically. Even the bare model provides better (corresponding to approximately two times higher) number of direct relationships than the graph model.
The figures also confirm the preliminary assertion that combination $\lambda = 3$, $\epsilon = 1$ provides the highest number of direct connections as was expected. In the same way, combination $\lambda = 1$, $\epsilon = 3$ results in the lowest number of relationships.

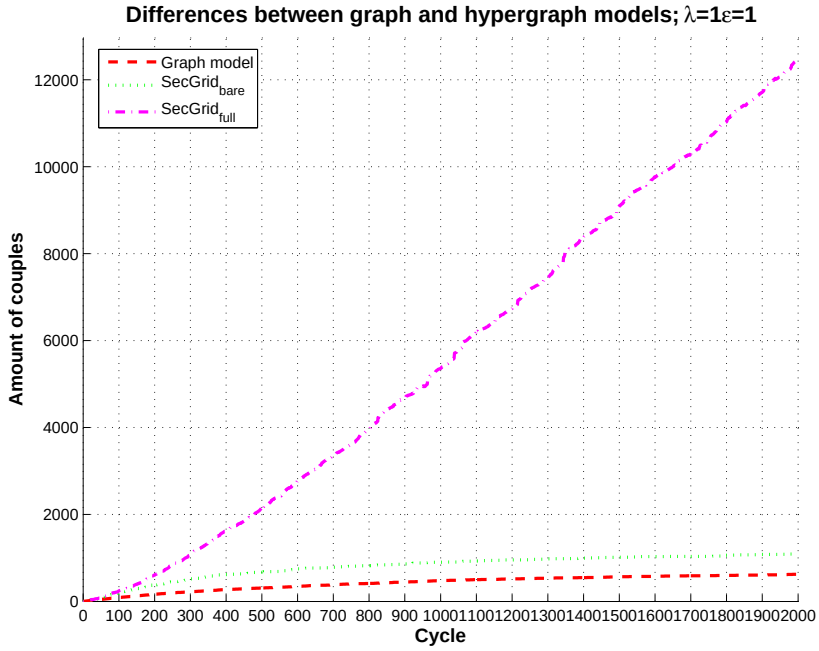


Figure 7.3.8: Normal distribution, $\lambda = 1$, $\epsilon = 1$

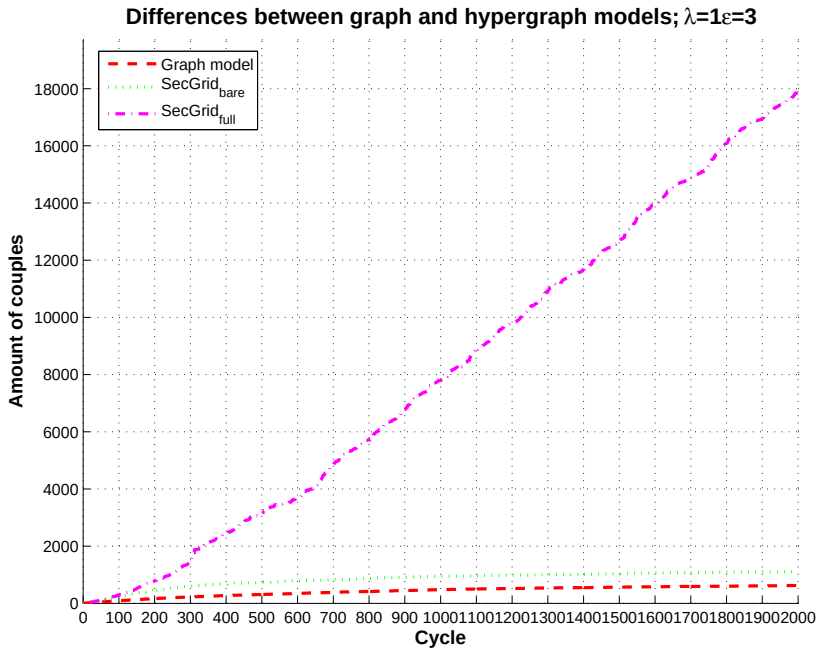


Figure 7.3.9: Normal distribution, $\lambda = 1$, $\epsilon = 3$

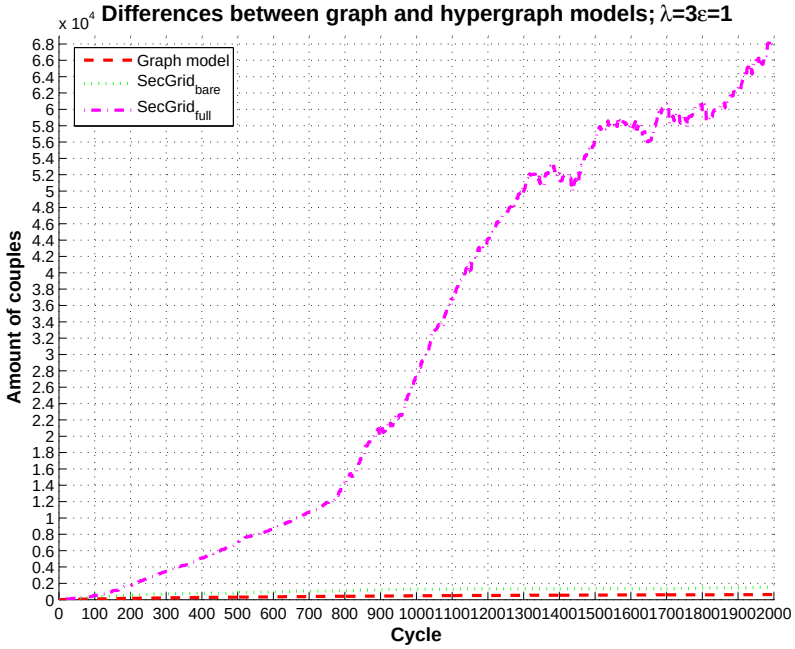


Figure 7.3.10: Normal distribution, $\lambda = 3$, $\epsilon = 1$

In the case of the normal distribution of random input data (Figures 7.3.8, 7.3.9, 7.3.10) the best results are also given by the combination $\lambda = 3$, $\epsilon = 1$.

The combination $\lambda = 1$, $\epsilon = 3$, on the other hand, overcame the combination $\lambda = 1$, $\epsilon = 1$, which is different from the case of uniform input data distribution. Here is different distribution of invitations issued by users. The uniform distribution spreads all 2000 invitations uniformly onto all users, unlike the normal distribution, which concentrates most invitations only onto limited number of users. In this way, normally distributed input data implies less splits with more joins resulting in larger groups with smaller intersections (the same can be seen in the stability experiments).

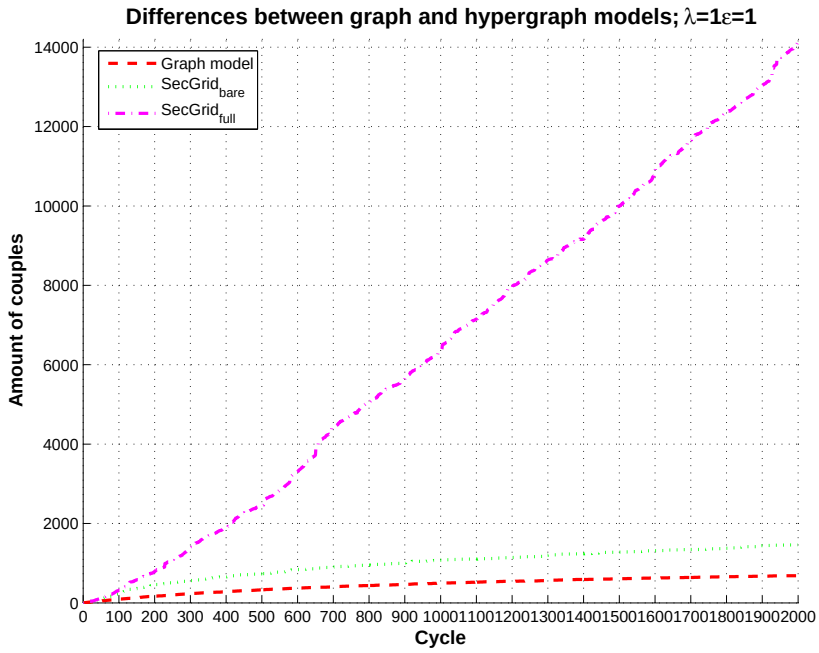


Figure 7.3.11: Exponential distribution, $\lambda = 1$, $\epsilon = 1$

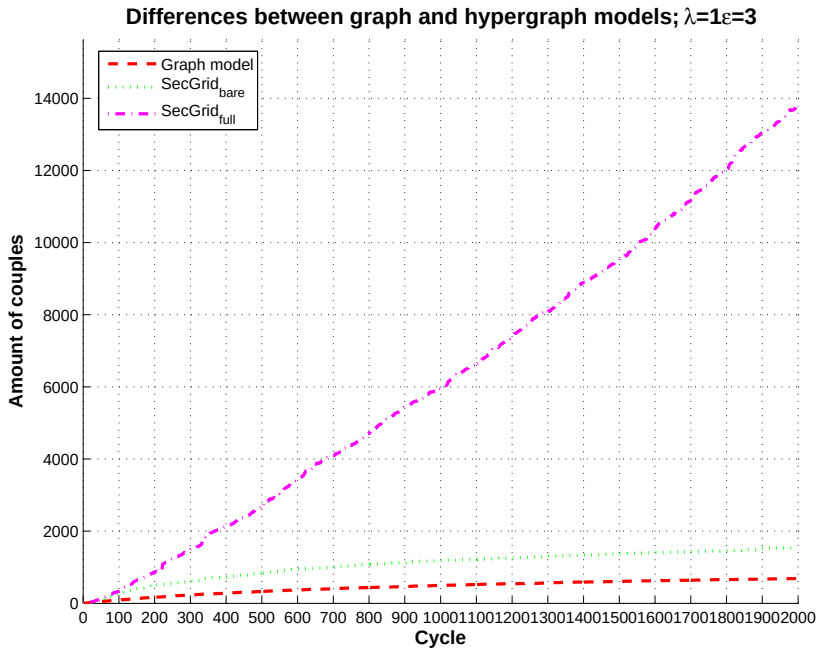


Figure 7.3.12: Exponential distribution, $\lambda = 1$, $\epsilon = 3$

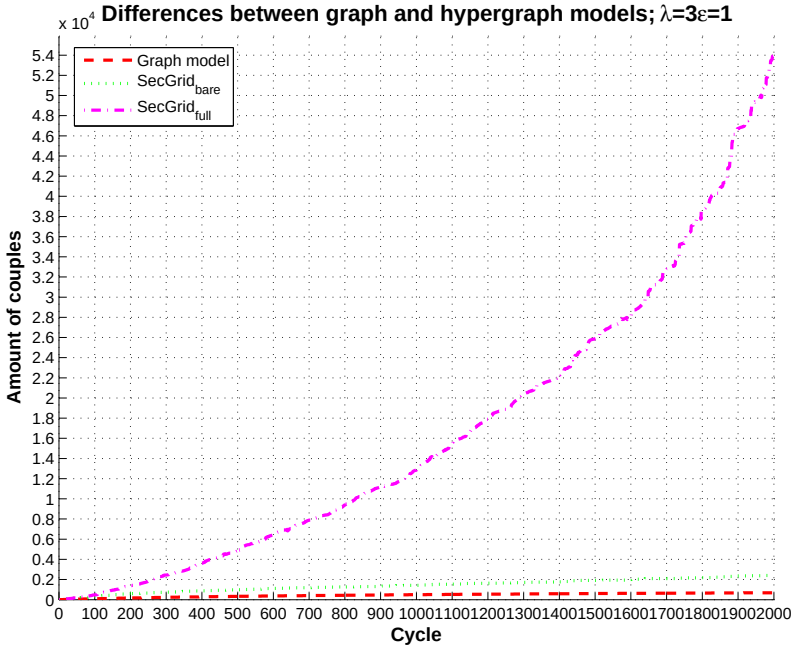


Figure 7.3.13: Exponential distribution, $\lambda = 3$, $\epsilon = 1$

The last set of figures (7.3.11, 7.3.12, 7.3.13) presents the results for exponential distribution of random invitations. In this case it is noticeable that final direct relationships in the graph model is 685, although one should expect 2000. The reason lies in the distribution of invitations. In the exponential distribution most invitations are concentrated between smaller number of users, which results in multiple invitations between the same users. The other interesting fact is the shape of curve for parameters $\lambda = 3$, $\epsilon = 1$ (Figure 7.3.13). This is due to the distribution of invitation as well, while bigger concentration of invitations between a few users creates a system of groups with larger intersections at the beginning. Nevertheless, with additional invitations groups are split into smaller groups with smaller intersections – resulting in the rapid increase in number of direct relationships.

The main aim of the presented experiments was to show advantages of the hypergraph model used by SecGrid over the graph model used in most contemporary trust building systems. The question whether the hypergraph model has any advantages in building direct trust between users has been positively answered by the experiment.

7.3.2 Trustworthiness of the SecGrid Model

The precedent experiments were oriented towards the stability of the SD Algorithm. In the following experiment we concentrated on the security aspects of the SecGrid model.

This sub-section is organized as follows:

- firstly we describe the experiment,
- followed by the list of experimental results given for eight different combinations of the parameters;
- at the end we provide summary of the experiment.

Description of the Experiment

By a *good user* will be called a user that causes no harm to the others.

A *malicious user* is a user that independently or in cooperation with other malicious users behaves in order to boost its own profit on the good users' coat-tails.

The scenario used for the experiment is described (in a pseudo code) in the following algorithm:

Algorithm 2 Trustworthiness of the SecGrid Model

```
1: Run the SD Algorithm for n cycles
2: for k steps do
3:     Get randomly a malicious user u_m                ▷ get user only once
4:     Get randomly a good user u_g
5:     for all n_i ∈ hyperedges(u_g) do
6:         for all n_j ∈ hyperedges(u_m) do
7:             if n_i ∩ n_j ≠ {0} then
8:                 Create a new group n_new, which contains all users,
9:                 except u_m and users invited by u_m
10:            end if
11:        end for
12:    end for
13: end for
```

The SD Algorithm is run for n cycles creating a system of groups (line 1). Subsequently the following is repeated for k steps:

1. (lines 3-4) a pair of a good user (u_g) and a malicious user (u_m) is randomly generated,
2. (line 5) to preserve its own as well as the overall security, the good user (u_g) spreads his knowledge about the malicious one (u_m) into all his/her groups,

3. (lines 8-9) each group that contains the good user (u_g) and also a malicious user (u_m) is split, so that a new group contains all users except:

   - the malicious one (u_m) and
   - all users invited by the malicious one.

   The newly created group has an increased level of trust (see bellow).

In this way the SecGrid model does not punish malicious users (e.g. by decreasing their reputation), but it creates a new group for users not invited by the malicious one with increased level of trust.

The SecGrid model further preserves knowledge about the malicious users by maintaining the old group (note that the old group remains unchanged). This is particularly important, as good users remain conscious about the users left behind. This allows them to block later attempts to infiltrate any of their trusted groups.

In the experiment the parameters were chosen as follows:

- $n = 300$ (amount of cycles for the SD Algorithm - chosen accordingly to the stability experiments in section 7.2),
- $k = 30$ (amount of steps for the security sub-system - this amount was shown by the experiments to be appropriate)
- 50 users (this amount seems to be upper bound enabling visual comparison of results)

Furthermore we used the following combinations of parameters (λ, ϵ) of the SD Algorithm:

1. $\lambda = 1$, $\epsilon = 1$,
2. $\lambda = 4$, $\epsilon = 1$,
3. $\lambda = 1$, $\epsilon = 4$.

These combinations lead the SD Algorithm to make changes more quick (see section 7.2), thus creating the system of groups used for the next part of the security experiments.

Each user in the system was marked as a good or a malicious one. We used eight different ratios between good and malicious users expressed as percentage (further marked as Ψ):

1. $\Psi = 95\%$ (system with most of good users),
2. $\Psi = 90\%$.
3. $\Psi = 80\%$,
4. $\Psi = 70\%$,
5. $\Psi = 50\%$ (system with half good and half malicious users),
6. $\Psi = 30\%$,
7. $\Psi = 20\%$,
8. $\Psi = 10\%$ (system with majority of malicious users).

For each ratio we randomly generated an input for the SD Algorithm containing three sets of invitations:

- good users invite good users (1/3 in the input),
- malicious users invite malicious ones (1/3),
- combination of invitations between the good users and the malicious ones (1/3).

This concrete input of three sets of invitations will be called *"well mixed"*.

The *trustworthiness* of the SecGrid model *is higher* if malicious users are *isolated in groups with lower level of trust* and groups with *higher level of trust contain mostly good users*. Therefore in the following we investigate ability of the SecGrid model to isolate malicious users in the groups with lower level of trust.

Interpretation of Results: The trustworthiness of the SecGrid model is shown in 3 graphs for each combination of parameters and various percentage of good users in groups (further denoted as Φ):

1. The first graph shows progress of **average** Φ as a function of cycles (case a)). The group splitting preserves the old group unchanged, therefore the average Φ computed for a new cycle shows the **trend** in the evolution.
2. The second graph shows **starting configuration** created by the SD Algorithm as histogram with the same axes as in the previously mentioned experiments (case b)).
3. The last graph shows **evolution of each group** in the system.
   On the x-axis are shown cycles and on the y-axis is shown percentage of good users in groups (Φ).

   - Each **small circle** in the graph corresponds to a **group**.

- In **blue small circles** are plotted **stable groups** (they have existed for a certain period of time).
- By a **blue solid line** is shown **life time of a stable group** until it is split into a new and an old one (according to Algorithm 2).
- The **new group** is shown in a **red asterisk**.
- A **red dash-dot line** connects the **stable** group and the **new group**. It goes upwards in case the new group contains more good users, downwards otherwise.

Results for Selected Parameters

The first set of experiments is for $\Psi = 95\%$. This suggests a trusted system with majority of good users. Let start with parameters $\lambda = 1$, $\epsilon = 1$. Such combination leads the SD Algorithm to a system of rather smaller groups with bigger intersections. Figure 7.3.14 a) shows that changes to the overall trustworthiness occurred only during the first 3 steps and the changes lead to the increase of the trustworthiness. In the rest of the 26 cycles no changes were triggered.

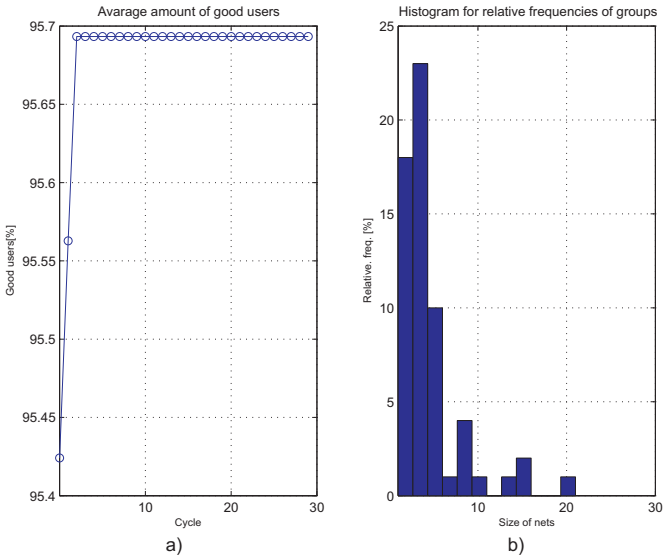


Figure 7.3.14: Evolution of Trustworthiness ($\lambda = 1$, $\epsilon = 1$, $\Psi = 95\%$).

Figure 7.3.15 shows that each new group created during splitting has 100% of good users. From the figure it is also visible that the splitting occurred only during the first three steps. This is mostly for low number of the malicious users in the system. Note that some groups with less then 100% of good users were not involved in splitting (no good user triggered splitting) – these groups corresponds to red circles on the y-axe connected with no blue line in Figure 7.3.15 (on the next page).

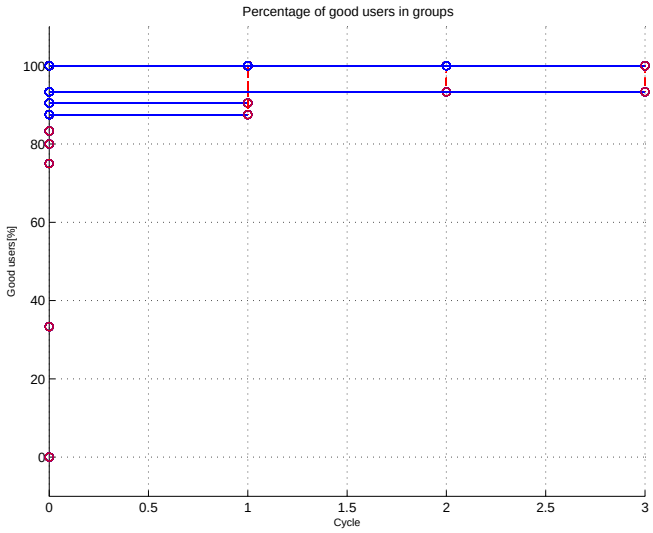


Figure 7.3.15: Evolution of Groups ($\lambda = 1$, $\epsilon = 1$, $\Psi = 95\%$)

The following list summarizes meaning of the Figure 7.3.15:

- more red asterisks on the top of the graph suggest that new groups contain more good users,
- longer blue lines represent latter splitting,
- longer upwards red lines represent larger increase of good users in new groups,
- longer downwards red lines represent larger decrease of good users in new groups.

(Ideally, the figure should contain only short blue lines, long upward red lines and most red asterisk at 100% of good users.)

The final summarization for all combinations of Ψ and parameters of the SD Algorithm is given at the end of this section.

The next combination of parameters ($\lambda = 4$, $\epsilon = 1$) leads the SD Algorithm to create a system of more interconnected ($\epsilon = 1$) rather larger groups ($\lambda = 4$). This is visible in Figure 7.3.16 b). Even in this case the splitting occurred only during the first three cycles.
The most important result can be seen in Figure 7.3.17, as each group (except for the one with no good users in) had 100% of good users in at the end of the first three cycles. It corresponds to the fact that after only three splitting each group in the system with higher level of trust contains only good users.

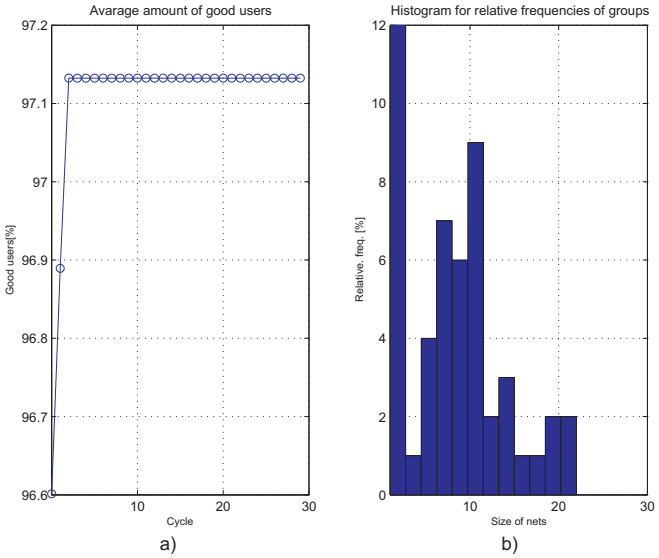


Figure 7.3.16: Evolution of Trustworthiness ($\lambda = 4$, $\epsilon = 1$, $\Psi = 95\%$).

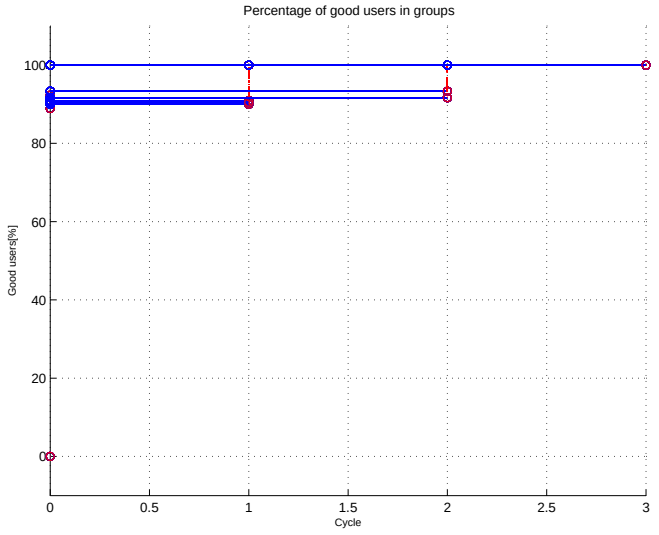


Figure 7.3.17: Evolution of Groups ($\lambda = 4$, $\epsilon = 1$, $\Psi = 95\%$)

The last combination of parameters ($\lambda = 1$, $\epsilon = 4$) provides very similar increase of trustworthiness (see Figure 7.3.18 a)) as the previous one. This is mainly due to the fact that for this combination of parameters the SD Algorithm creates a systems of rather smaller but highly interconnected groups (Figure 7.3.18 b)). Therefore every information about a malicious user should propagate into groups easily. Even under this configuration each group included in the splitting reached 100% of good users (Figure 7.3.19). Only two groups did not split and remained un-split.

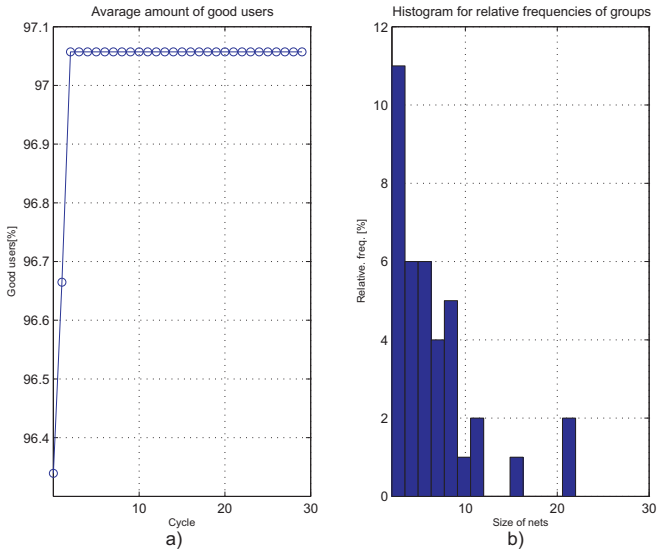


Figure 7.3.18: Evolution of Trustworthiness ($\lambda = 1$, $\epsilon = 4$, $\Psi = 95\%$).

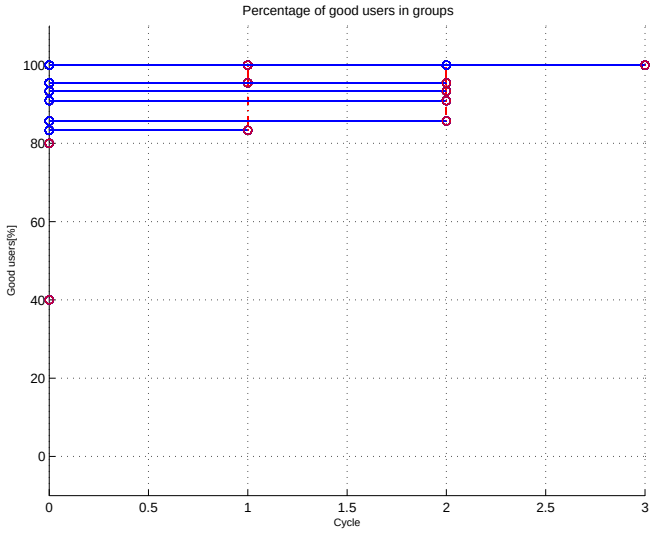


Figure 7.3.19: Evolution of Groups ($\lambda = 1$, $\epsilon = 4$, $\Psi = 95\%$)

The second set of experiments is for $\Psi = 80\%$. Figure 7.3.20 a) shows that the security subsystem improves overall trustworthiness in 10 cycles.

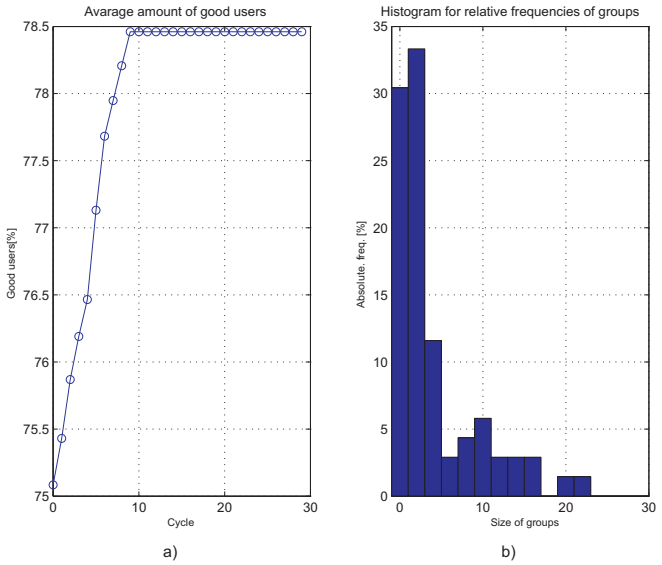


Figure 7.3.20: Evolution of Trustworthiness ($\lambda = 1$, $\epsilon = 1$, $\Psi = 80\%$).

The important result comes from Figure 7.3.21 – besides the group split at cycle 5 (percentage of good users had been around 18%) each group (with a higher level of trust) reached 100% of good users. Even the group split at cycle 5 improved to 50% of good users – this means by 32%. Compared to the experiments for $\Psi = 95\%$ the splitting occurred more often and to more groups. This is rather natural as number of malicious users was larger.

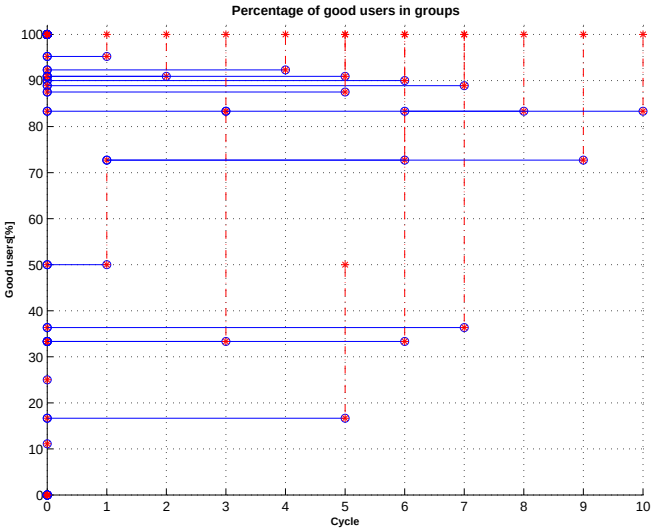


Figure 7.3.21: Evolution of Groups ($\lambda = 1$, $\epsilon = 1$, $\Psi = 80\%$)

The same can be asserted for the next combination of the parameters (Figures 7.3.22 and 7.3.23). In this case there are two groups that had not reached 100% of good users in. These groups are easily visible in Figure 7.3.23 at cycles 7 and 11. Nevertheless both groups were split the new groups contained majority of good users (an increase from 50% to around 68% and from 30% to around 75%, respectively). The other groups in the system had 100% of good users in at least at cycle 11. For the rest of cycles (from 12 up to 30) no change occurred as there was no new information in the input data.

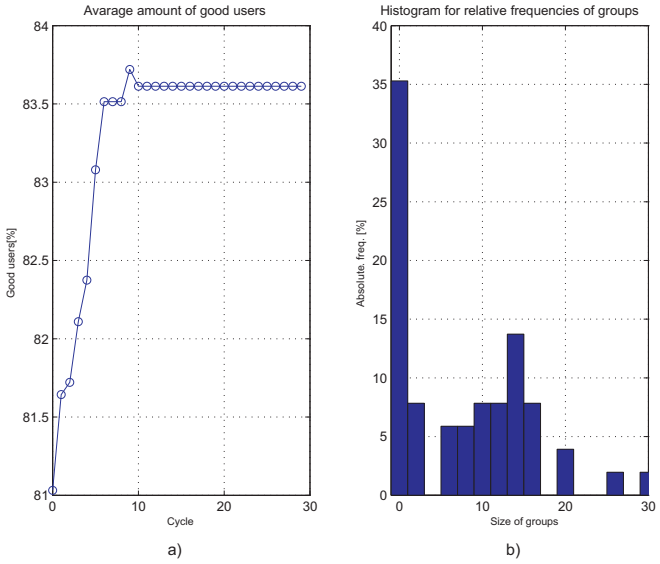


Figure 7.3.22: Evolution of Trustworthiness ($\lambda = 4$, $\epsilon = 1$, $\Psi = 80\%$).

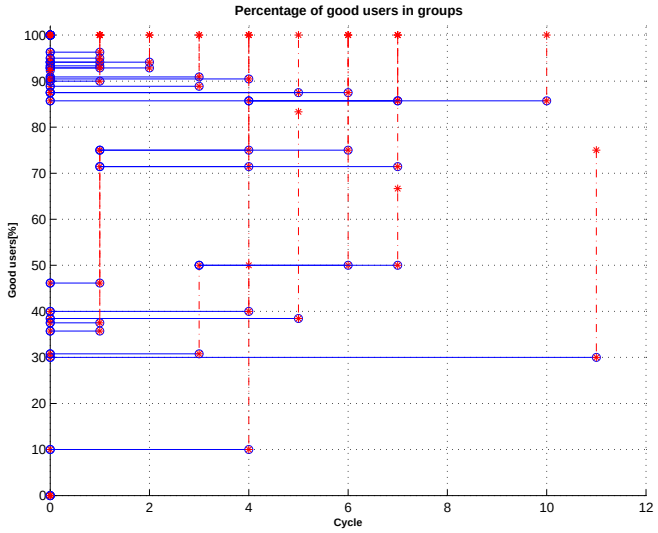


Figure 7.3.23: Evolution of Groups ($\lambda = 4$, $\epsilon = 1$, $\Psi = 80\%$)

For the last combination of parameters, Figure 7.3.24 ($\lambda = 1$, $\epsilon = 4$), invitations of users from groups that differ more in the level of trust are allowed.

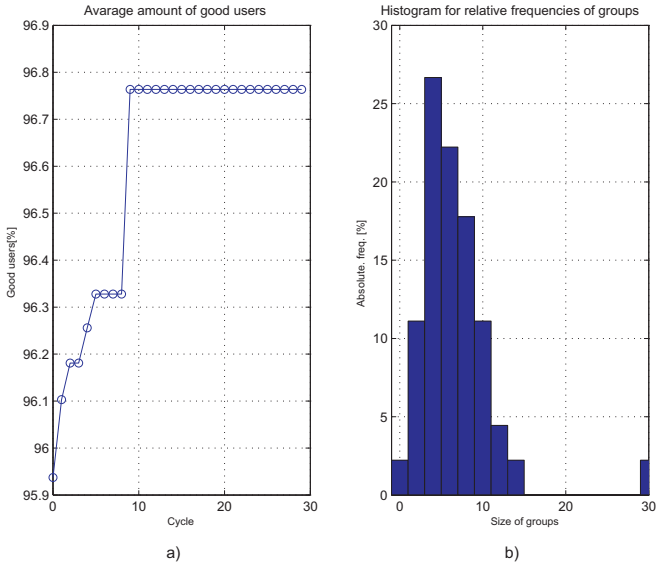


Figure 7.3.24: Evolution of Trustworthiness ($\lambda = 1$, $\epsilon = 4$, $\Psi = 80\%$).

Figure 7.3.25 shows that each group included in the splitting reached 100% of good users in at least at cycle 10. On the other hand, Figure 7.3.25 shows that most groups created by the SD Algorithm (shown on the y-axe) had had at least around 60% of good users in.

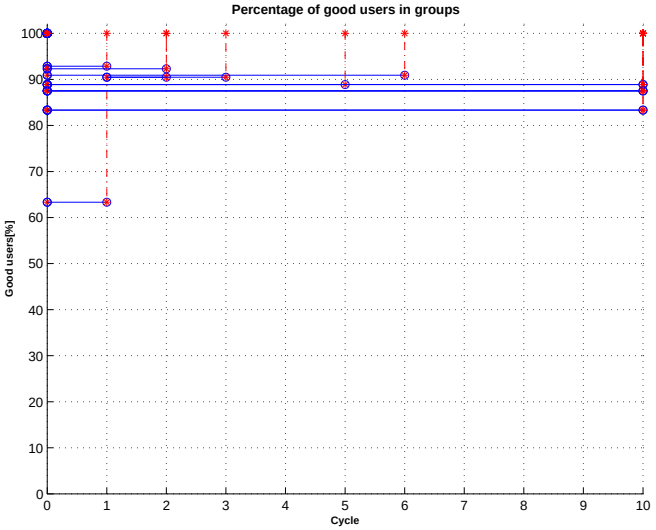


Figure 7.3.25: Evolution of Groups ($\lambda = 1$, $\epsilon = 4$, $\Psi = 80\%$)

In the third set of experiments each second user in the system is a malicious ($\Psi = 50\%$).
In the first case of parameters $\lambda = 1$, $\epsilon = 1$ (see Figure 7.3.26), the splitting causes improvements of the overall percentage of good users in groups. On the other hand, in this case of more malicious users in the system the splitting is triggered for 26 cycles. In this way it differs from the experiments for $\Psi = 80\%$ and $\Psi = 95\%$ where splitting was triggered at least at cycle 10.

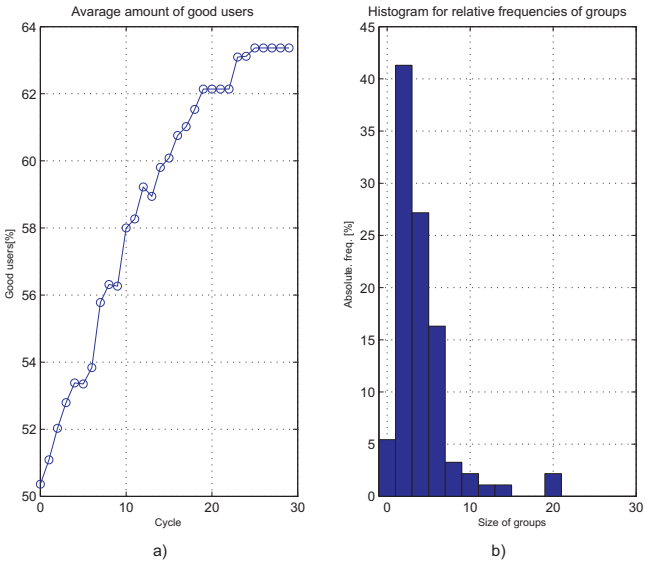


Figure 7.3.26: Evolution of Trustworthiness ($\lambda = 1$, $\epsilon = 1$, $\Psi = 50\%$).

Figure 7.3.27 shows that new groups were created with a remarkably high increase of number of good users (long red lines) – majority of newly created group (groups with higher level of trust) had 100% of good users in. Around half of groups had less or equal than 40% of good users in at the beginning (cycle 0). These groups were split and most (except for 2) reached 100% of good users. This is remarkable gain.

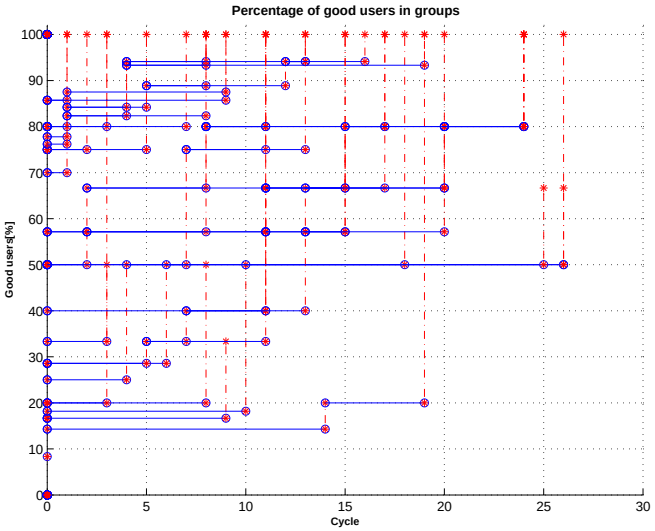


Figure 7.3.27: Evolution of Groups ($\lambda = 1$, $\epsilon = 1$, $\Psi = 50\%$)

The next combination of parameters ($\lambda = 4$, $\epsilon = 1$) presented in Figure 7.3.28 provides constant gain in the overall trustworthiness. Even in this case the splitting by security sub-system ended at cycle 26.
Figure 7.3.29 shows that there were quite a lot of splits (more than in any previous experiments). More than half of groups created by the SD Algorithm had less than 50% of good users in. In this case the security sub-system ended with one of the largest numbers of groups (around 10) that had less then 100% good users (this number is comparable to systems with only 20% of good users for which we provide experiment in the following). Nevertheless even in this case the number of groups with higher level of trust and with the percentage of good user equal or close to 100% is remarkably large.

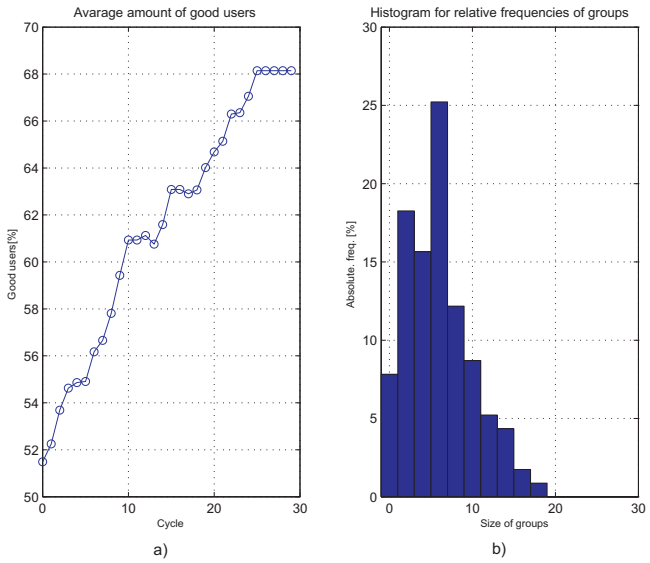


Figure 7.3.28: Evolution of Trustworthiness ($\lambda = 4$, $\epsilon = 1$, $\Psi = 50\%$).

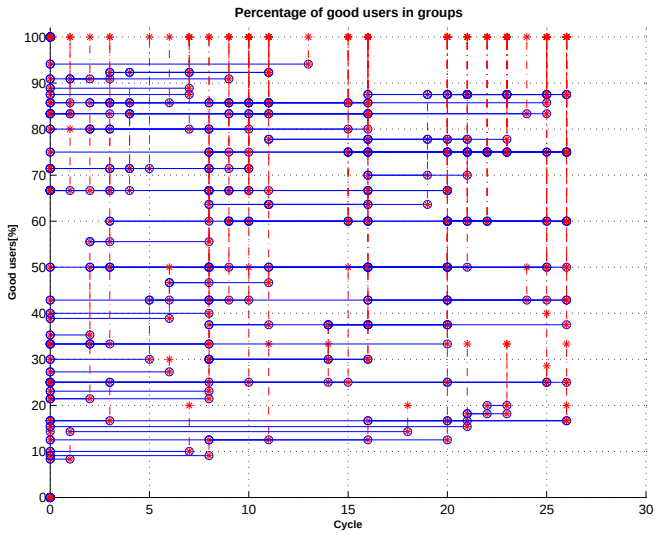


Figure 7.3.29: Evolution of Groups ($\lambda = 4$, $\epsilon = 1$, $\Psi = 50\%$)

Figure 7.3.30 shows results for the last combination of parameters ($\lambda = 1$, $\epsilon = 4$). In this case, the shape of the curve presented in Figure 7.3.30 is not so smooth as in the previous cases. It suggests that some splitting created a group(s) with less good users in.

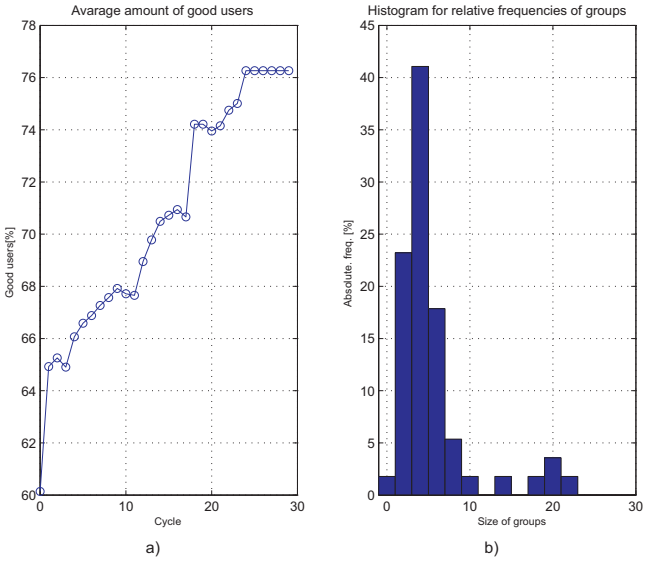


Figure 7.3.30: Evolution of Trustworthiness ($\lambda = 1$, $\epsilon = 4$, $\Psi = 50\%$).

Figure 7.3.31 provides details on splitting of groups by the security sub-system. It shows that the SD Algorithm with the parameters $\lambda = 1$ and $\epsilon = 4$ provides better grouping for the majority of groups with more than 50% of good users in. Moreover, the splitting by the security sub-system worked also better than in the previous case and the majority of groups had at the end of the experiment 100% of good users in.

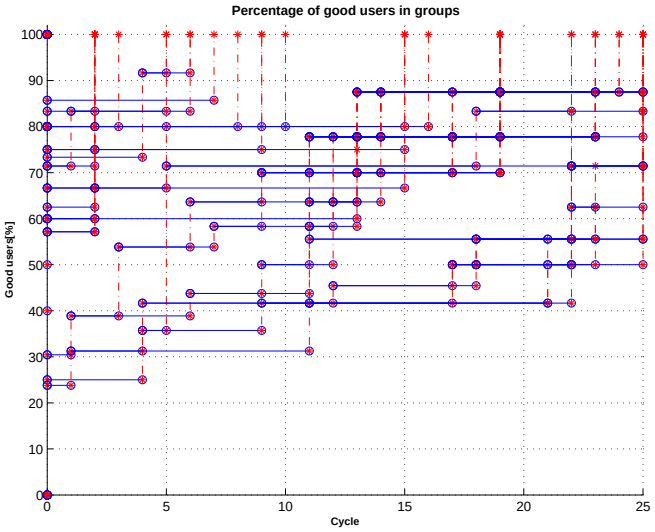


Figure 7.3.31: Evolution of Groups ($\lambda = 1$, $\epsilon = 4$, $\Psi = 50\%$)

Figure 7.3.32 a) shows the evolution of trustworthiness of the SecGrid model for the first combination of the parameters ($\lambda = 1$, $\epsilon = 1$) and only 20% of good users in the system. From the figure it is visible that the security sub-system triggered splitting up to 24 cycles.

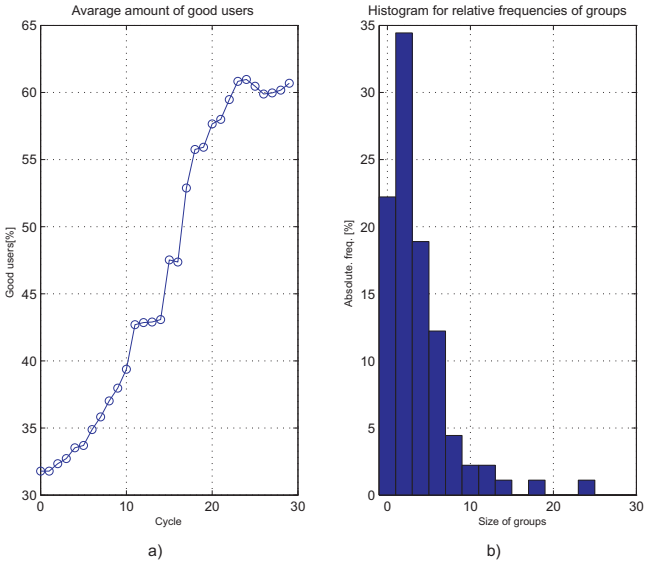


Figure 7.3.32: Evolution of Trustworthiness ($\lambda = 1$, $\epsilon = 1$, $\Psi = 20\%$).

The security sub-system further improves the trustworthiness by many splitting (see Figure 7.3.33) for 24 cycles. Despite the low number of good users in the system, the security sub-system together with the SD Algorithm were able to improve the percentage of good users for majority of newly created group up to 100%. This illustrates the ability of the SecGrid model to work even in the environment with low number of good users.

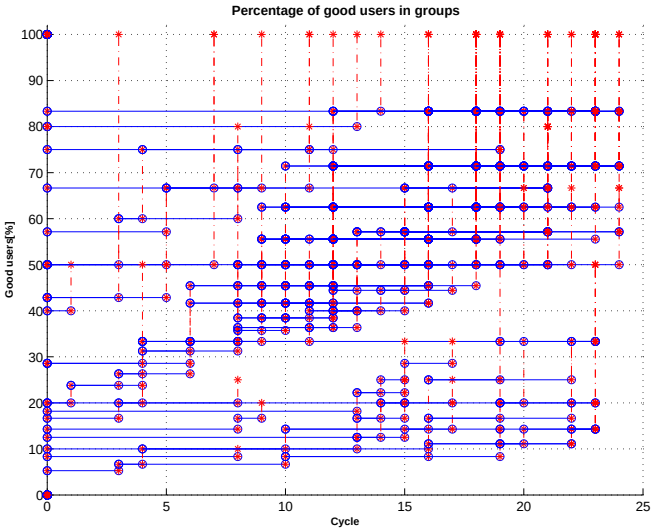


Figure 7.3.33: Evolution of Groups ($\lambda = 1$, $\epsilon = 1$, $\Psi = 20\%$).

Figures 7.3.34 and 7.3.35 shown a different combination of the parameters $\lambda = 4$ and $\epsilon = 1$. In this case the splitting is triggered only up to 18 cycles. Even the splitting by the security sub-system (see Figure 7.3.35) shows a rather lower number of events compared to the previous case.

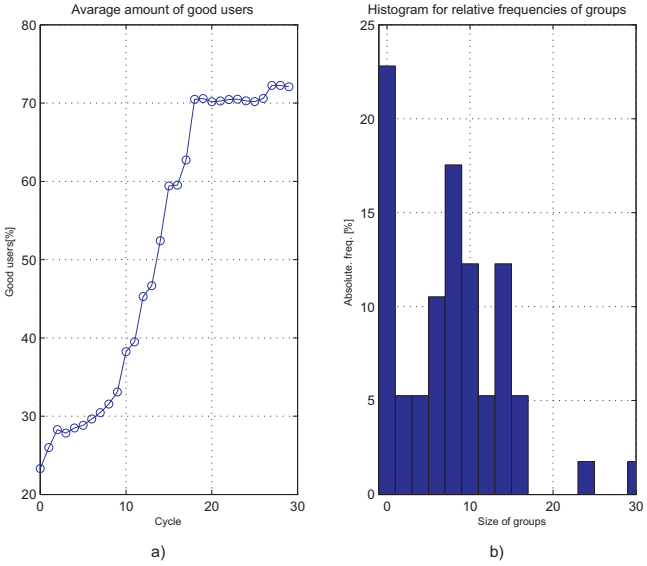


Figure 7.3.34: Evolution of Trustworthiness ($\lambda = 4$, $\epsilon = 1$, $\Psi = 20\%$).

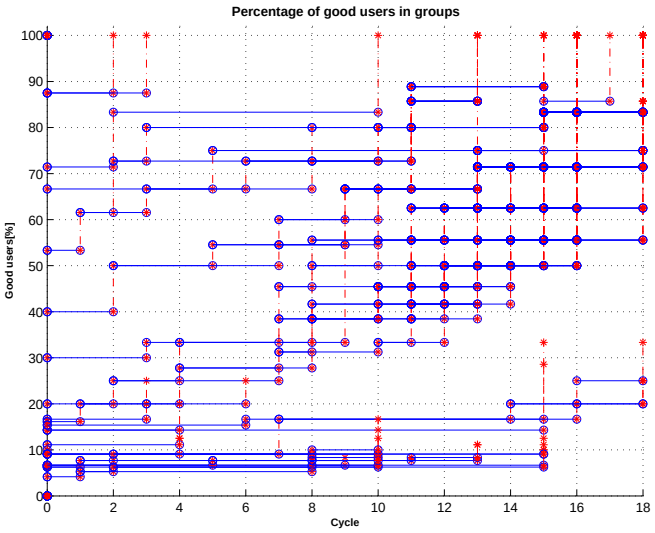


Figure 7.3.35: Evolution of Groups ($\lambda = 4$, $\epsilon = 1$, $\Psi = 20\%$).

Figure 7.3.36 and 7.3.37 show the last combination of parameters $\lambda = 1$, $\epsilon = 4$. This combination of parameters provided very similar results to parameters $\lambda = 4$, $\epsilon = 1$. The important point here shown in Figure 7.3.37 is that the number of groups with 100% of good users in at the end of the experiment is quite low compared to the previous cases. It suggests the combination $\lambda = 1$, $\epsilon = 4$ is not appropriate for system with low number of good users.

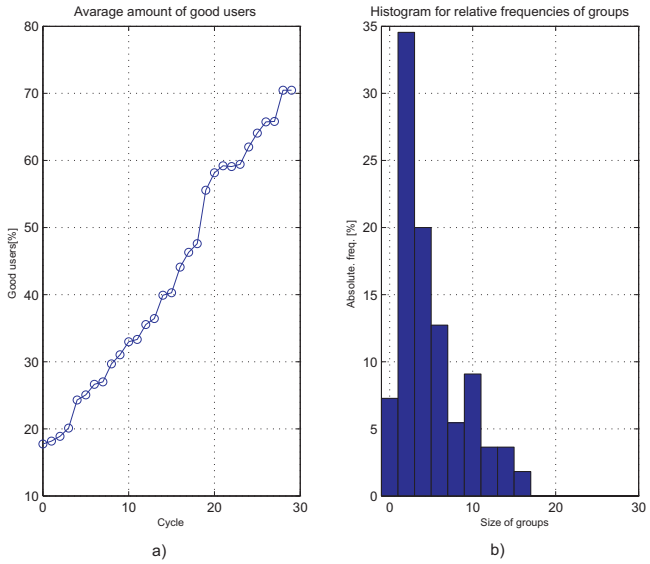


Figure 7.3.36: Evolution of Trustworthiness ($\lambda = 1$, $\epsilon = 4$, $\Psi = 20\%$).

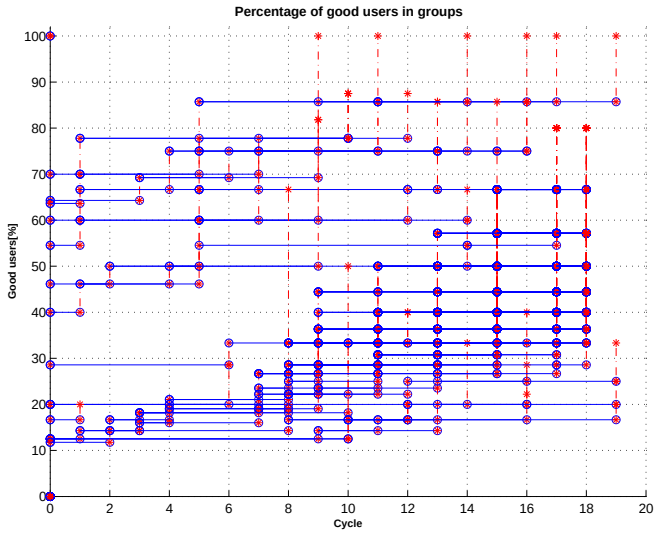


Figure 7.3.37: Evolution of Groups ($\lambda = 1$, $\epsilon = 4$, $\Psi = 20\%$).

Summary

We give here a full report on experiments for all combinations of parameters. The results are summarized in table 7.3.1 bellow.

Table 7.3.1: Overview on Experiments for Trustworthiness of SecGrid (*"well mixed"* input)

λ, ϵ	Φ_1	Φ_2	$\Delta\Phi$	$\|N\|_1$	$\|N\|_2$	$\Delta\|N\|$	Ψ
1,1	95	96	1	61	68	7	
4,1	96	98	2	50	64	14	95%
1,4	96	98	2	38	51	13	
1,1	92	98	6	65	82	17	
4,1	91	95	4	40	61	21	90%
1,4	93	96	3	48	68	20	
1,1	75	79	4	69	86	17	
4,1	81	84	3	51	81	30	80%
1,4	96	97	1	45	59	14	
1,1	64	82	18	85	198	113	
4,1	76	94	18	66	338	272	70%
1,4	84	92	8	60	139	79	
1,1	50	63	13	92	162	70	
4,1	52	68	17	115	334	219	50%
1,4	60	76	16	56	206	150	
1,1	45	70	25	92	292	200	
4,1	40	85	45	89	1145	1056	30%
1,4	44	85	41	54	1867	1813	
1,1	31	61	30	90	470	380	
4,1	23	72	49	57	419	362	20%
1,4	18	71	53	55	454	399	
1,1	19	50	31	74	147	73	
4,1	5	60	55	29	2541	2512	10%
1,4	9	59	50	40	194	154	

The meaning of symbols in the table:

- Φ_1 - starting average of percentage of good users in groups created by the SD Algorithm,
- Φ_2 - final average of percentage of good users in groups,
- $\Delta\Phi$ - gain of percentage of good users in groups,
- $|N|_1$ - amount of groups created by the SD Algorithm,
- $|N|_2$ - amount of groups created by the SD Algorithm plus groups created by the security sub-system,
- $\Delta|N|$ - amount of groups created by the security sub-system,

- Ψ - number of good users in system [%].

Interpretation of the Table 7.3.1. The security sub-system:

- always improves trustworthiness,
- works well even in the systems with low amount of good users,
- provides the best gain of the trustworthiness for system with low amount of good users (it demonstrates its ability to cope with such systems).

The Table 7.3.1 also gives basic information about the SD Algorithm:

- the maximum number of groups was created by the SD Algorithm with the parameters $\lambda = 1$, $\epsilon = 1$. This combination supports splitting thus creating more groups,
- the minimum number of groups was created for parameters $\lambda = 1$, $\epsilon = 4$. This combination supports adding thus creating smaller number of rather larger groups,
- the SD Algorithm provided best total trustworthiness for parameters $\lambda = 1$, $\epsilon = 4$ for system with at least half of good users. In more secure systems this combination suggests many adding which leads to higher trustworthiness.
- the best total trustworthiness for systems with low number of good users was on the other hand given by parameters $\lambda = 1$, $\epsilon = 1$. (This combinations makes a lot of splits thereby creating many smaller groups which are less prone to infiltration.)

One could object that the input invitations generated for the experiments create a system of groups where good users are concentrated in one set of groups and malicious users in the second one. Input data consisted of three sets of invitations (*"well mixed"* input) as mentioned at the beginning of this section. Therefore in the following Table 7.3.2 results for input data generated randomly for all users in the system (*"random"* input) are shown. The parameters and the experiment remained the same.

Table 7.3.2: Overview on Experiments for Trustworthiness of the SecGrid (*"random"* input)

λ, ϵ	Φ_1	Φ_2	$\Delta\Phi$	$\|N\|_1$	$\|N\|_2$	$\Delta\|N\|$	Ψ
1,1	95	96	1	83	101	18	
4,1	95	97	2	79	145	66	95%
1,4	95	97	2	59	78	19	
1,1	92	94	2	81	111	30	
4,1	92	95	3	66	136	66	90%
1,4	91	94	3	74	114	40	
1,1	83	85	2	89	121	32	
4,1	77	82	5	98	190	92	80%
1,4	80	84	4	69	108	39	
1,1	70	74	4	100	183	83	
4,1	73	77	4	135	448	313	70%
1,4	68	76	8	85	232	147	
1,1	72	69	-3	142	479	337	
4,1	62	54	-8	124	422	298	50%
1,4	69	66	-3	78	284	206	
1,1	28	44	16	106	302	196	
4,1	32	49	17	98	1213	1115	30%
1,4	26	49	23	85	343	258	
1,1	18	48	30	92	411	319	
4,1	21	61	40	123	424	301	20%
1,4	24	60	36	82	457	375	
1,1	14	47	33	100	484	384	
4,1	12	49	37	80	3877	3797	10%
1,4	12	47	35	66	716	650	

The SecGrid model provided very similar results to the *"well mixed"* input for systems with low or high number of good users. Even in this case uniformly generated invitations puts most invitation among only malicious users or among only good users. This corresponds to the creation of stable basic system of groups where malicious and good users are isolated in different groups, as in the case of *"well mixed"* input. Therefore the results provided for the *"well mixed"* and the *"random"* input data are comparable.

In systems with similar amount of good and malicious users (Φ equals to approximately 50%) the security sub-system might cause a small decrease in the overall trustworthiness.

According to the experiments done we can state that the SD Algorithm positively influence overall security, expressed by percentage of good users in groups. In addition to this improvements the security sub-system increases the overall security by isolating malicious users.

A remarkable feature of the SecGrid model is its ability to improve overall security in systems where ratio between good and malicious users is low. In this case the SecGrid model takes full advantages of the hypergraph model.

Chapter 8

Experimental Implementation of the SecGrid Model

In the previous chapters we proposed the SecGrid model – a model for building trust in decentralized environments by managing groups of users.

In such system the users infer their level of trust based on the groups they are members of. The SecGrid model is dynamic by supporting changes in the structure. One of the main peculiarities of the SecGrid model is its natural support of human intuition of trust. This was the reason why we wanted to verify the SecGrid model usability in the real world by ordinary users.

We prepared a simple experimental implementation enabling exchange of keys (e.g. tKeys) by devices and technologies accessible for mass users. We have chosen mobile phones as the target device. The main reasons can be summarized as:

1. mobile phones can be found everywhere around the world,
2. majority of today mobile phones offer enough computational and storage capabilities,
3. users have their mobile phone ready almost anytime and anywhere,
4. users have different applications installed in their mobile phones, thus using them for additional tasks (entertainment, scheduling, etc.).

The next step was the selection of communication technology used for transmission of tKeys. Today mobile phones usually support infrared beam (IrDA), GSM and Bluetooth (BT) [72] wireless communication standard. From our point of view, the most suitable was Bluetooth (for its parameters and hardware support).

Concerning the selection of mobile phones development environments, one can use various development environments for programming applications for mobile phones (e.g. Java Micro Edition [73], Symbian, etc.).

Java Micro Edition (J2ME) fulfilled all our requirements:

1. supports Bluetooth communication standard (JSR-82),
2. enables creation of rich graphic user interface,
3. enable storage and retrieval of byte data (this format is supported by J2ME),
4. it is supported by majority of modern mobile phone.

8.1 MyKeys Implementation

MyKeys (Figure 8.1.1) experimental implementation provides users with basic functionality needed for the storage and the exchange of tKeys stored in a mobile phone. MyKeys can be run on any mobile phones that fulfills the following requirements:

- J2ME support,
- Connected Limited Device Configuration (CLDC) 1.0,
- JSR-82 Bluetooth package,
- graphic display.

The main functionality of MyKeys can be structurally given as:

- storage, display and management of user keys,
- concurrent exchange of keys between up to 8 devices,
- service discovery and acceptance through Bluetooth.

In Figure 8.1.2 the UML class diagram of the basic classes of MyKeys is shown. The main idea behind the application is to program the Bluetooth server offering the *MyKeys service* to clients. Clients search for services provided by

Figure 8.1.1: The MyKeys Splash Screen

Bluetooth server in their communication range and if found, then they try to connect to the *MyKeys service*. For the purposes of running the server, MyKey includes the *server* class (shown on the left in the Figure 8.1.2) and also the *client* class for the purpose of service discovery (shown on the right in Figure 8.1.2).

Figure 8.1.2: MyKeys UML Diagram

The current version allows users to manually choose between running sever or client through GUI. We plan to automatize this functionality so that mobile phone will periodically[1] switch between the server and the client and if the *MyKey service* is discovered the user is notified.

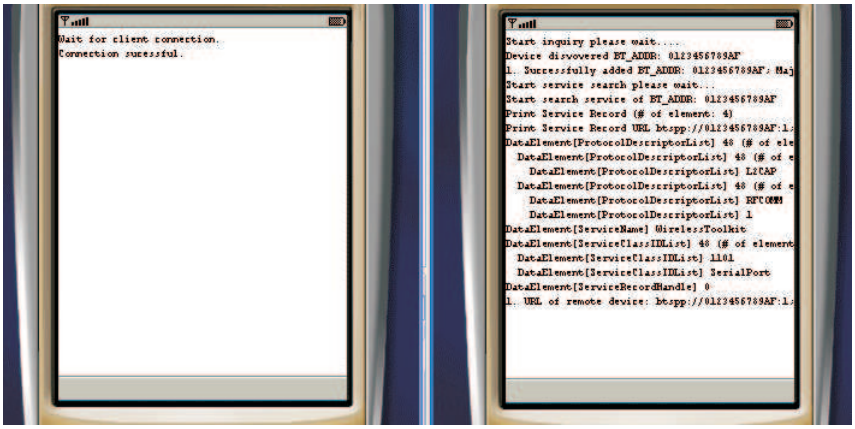


Figure 8.1.3: Debugging Screen of MyKeys (providing detailed information about the state of progress of the application used for experimental purposes.)

The *rmsManager* class (shown on the left-bottom corner in Figure 8.1.2) realizes storage and retrieval of user tKeys. The *RMS* storage provided by a mobile phone is used to store tKeys in stream of byte format. This storage type is useful for possible encryption of stored tKeys. The last class worth noticing is the *splash class*. This class realizes User Graphic Interface (GUI) by extending canvas class. In this way, MyKeys provides rich graphic interface for users.

We have implemented rich logs providing detailed information about current state of the MyKeys. This is important in the case of testing connectivity of mobile phones, as the Bluetooth communication range is naturally limited. Figure 8.1.3 shows an example of screens of two mobile phones in the case of running service discovery and the following connection establishment.

The *server* and *client classes* have been developed in cooperation with the BlueGame [89] project supported by ČVUT-Ericsson-Vodafone R&D Centre (RDC).

[1]The period must change for the de-synchronization reasons.

8.2 MyKeys Experiments

For experiments two types of Nokia mobile phones[1] with the MyKeys application installed were used, particulary

- Nokia N70
- Nokia 6600

Both mobile phones are *smart phones* with the Symbian operation system. Whereas the Nokia N70 is a representant of a relatively modern smart phone the second one (Nokia 6600) is quite old (it was one of the first smart mobile phones). We used these types of mobile phones to demonstrate the ability of the MyKeys implementation to work on new as well as older mobile phones.

We have tested the ability of MyKeys application to exchange and store tKeys in:

- an open area,
- an indoor environment (in the campus of the Technical university of Liberec).

We concentrated on the main parameter influencing usability of the MyKeys application – minimal distance needed for tKey exchange.

The results for minimal distance needed to establish connection have shown that in an open air area without obstacles the mobile phones were able to discover the other at distance of around 35 meters. On the other hand, the mobile phones were able to discover the *MyKey service* at distance of 30 meters. The results for indoor areas abilities are highly limited by the surroundings, type of construction of the building, shape of corridors, etc. Due to this we obtained really wide spectrum of distances varying from few meters (generally round 10 meters) down to several dozen of centimeters in the case of connection through a wall (Technical university of Liberec is generally built from bricks thick walls (50 cm)).

From our experimental operation of the MyKeys application it also followed that storage and retrieval of tKeys stored in the raw byte format posed no problems for both mobile phones.

We have also provided (with my colleagues Pavel Pirkl and Lukaš Závorka) some interesting results on energy consumption of Bluetooth modules in [89]. The main purpose of the experiments was to provide measurements summarizing impact of different type of transmission (type of packet, length of packet [72]) on energy consumption.

[1]The mobile phones were available thank to ČVUT-Ericsson-Vodafone R&D Centre (RDC)[74]

Conclusions

During my PhD. studies I have published several papers (see the list of author's references at the end of the thesis) at various international conferences (e.i. IEEE, ACM) as well as at local conferences including the mostly recognized local database conference DATAKON. One of my posters was awarded as one of the Best Poster at the international conference SOFSEM 2006.

In my PhD. studies I have concentrated on the security issues in distributed environments, but I have also published several papers on routing protocols for ad-hoc mobile networks and some papers in the field of the Semantic web. Nevertheless, my main interest has been the security in distributed and dynamic environments.

The thesis presents a new approach for treating the trust in a totally decentralized environment.
The most important contribution of the proposal is a different notion of trust. The majority of currently used approaches for trust management understand trust as a value connected to a particular pair of users. From our point of view, trust is a common phenomenon for a group of users. In this context, trust between users is inferred based on groups they are members of.
For the notion of trust as a single value for a particular pair of users, the graph model is used in a natural way. In the case of trust shared among group of users, the graph model faces severe drawbacks. For this reason, we proposed a new model representing a system of groups of users as a hypergraph, where one group is represented as a hyperedge.

As our proposal uses the hypergraph model for the representation of system of groups we proposed two versions of the algorithm for the transformation of a general (graph) input into the hypergraph model. The transformation follows specific requirements in order to provide a reasonable grouping.

Structure (which users share relationships) and weights (trust of the relationships) of relationships between users may be subject to changes, consequently we proposed an algorithm for managing the dynamics of the system of groups while preserving the overall security.

Our *Structure Dynamic Algorithm* manages basic security threats. We proposed a security sub-system designed especially for the hypergraph model.
All our algorithms together with the security sub-system were designed for totally distributed environments and experimentally implemented as *SecGrid* in ANSI C.

The stability of the dynamic part of the proposal was tested by a set of experiments in which our model proved its abilities against real social network data and artificial input data generated according to three different statistical distribution of random numbers.
The hypergraph model usage was verified by a couple of experiments where hypergraph model showed its advantages against the graph model. The main criterion of these experiments was the number of direct trust relationships between users: the hypergraph model outperformed the graph model by orders of magnitude.
The security sub-system of the SecGrid model was tested by several experiments in which SecGrid trustworthiness was proven even in the case of high ratio of malicious users.

At the end of the thesis an experimental implementation *MyKeys* was presented. It provides users with methods for tKey exchange and management through their mobile phones.

In the future I would like to continue in the work started and done during my PhD. Studies, more precisely the one presented in this thesis.
Firstly, I plan to implement the SecGrid model in a real world environment for data access and sharing (within my involvement in the Institute of Computer Sciences of the Academy of Sciences of the Czech Republic).
Furthermore, I would like to concentrate more on the algorithms for the transformation of the general input into hypergraph model as such algorithms can be used in Social Network analysis tasks.
Finally, I will enhance the *MyKey* implementation in order to provide a full SecGrid compliance.

Bibliography

[1] M. Deutsch. Cooperation and trust: Some theoretical notes. *Nebraska Symposium on Motivation. M. R. Jones, Nebraska University Press*, 1962.

[2] J. Waldo. Virtual organizations, pervasive computing, and an infrastructure for networking at the edge. *Information Systems Frontiers 4, Kluwer Academic Publishers*, 1:9–18, 2002.

[3] S. DasBit and S. Mitra. Challenges of computing in mobile cellular environment a survey. *Computer Communications*, 26:2090–2105(16), 2003.

[4] Y. Lu, B. Bhargava, W. Wang, Y. Zhong, and Wu X. Secure wireless network with movable base stations. *IEICE Trans. Community*, vol. E86-B, 2003.

[5] Y. Zong, B. Bhargava, and M. Mahoui. Trustworthiness based authorization on www. *IEEE Workshop on Security in Distributed Data Warehousing*, 2001.

[6] P. K. Behera and P. K.Meher. Prospects of group-based communication in mobile ad hoc networks. *Springer-Verlag Berlin Heidelberg*, 2002.

[7] A. Flaxman, A. Frieze, and E. Upfal. Efficient communication in an ad-hoc network. *Elsevier*, 2004.

[8] S. Basagni. Remarks on ad hoc networking. *Springer-Verlag, Berlin Heidelberg*, 2002.

[9] R. Molva and P. Michiardi. Security in ad hoc network. *IFIP International Federation for Information Processing*, 2003.

[10] S. Hedetniemi and A. Liestman. A survey of gossiping and broadcasting in communication networks. *Networks*, 18:319–349, 1988.

[11] W. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. *Proceedings of the* 5^{th} *Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99), Seattle, WA*, August, 1999.

[12] C. Intanagonwiwat, R. Govindan, and D. Estin. Adaptive protocols for information dissemination in wireless sensor networks. *Proceedings of the* 6^{th} *Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'00), Boston, MA*, August, 2000.

[13] D. Estrin and et al. Next century challenges: Scalable coordination in sensor networks. *Proceedings of the* 5^{th} *Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99), Seattle, WA*, August, 1999.

[14] W. Heinzelman, A. Chandrakasan, and H. Balakrishnam. Energy-efficient communication protocol for wireless sensor networks. *Proceedings of the Hawaii International Conference System Sciences, Hawaii*, January, 2000.

[15] S. Lindsey and C. S. Raghavendra. Pegasis: Power efficient gathering in sensor information systems. *Proceedings of the IEEE Aerospace Conference, Big Sky, Montana*, March, 2002.

[16] A. Manjeshwar and D. P. Agrawal. Teen: A protocol foe enhancement efficiency in wireless sensor networks. *Proceedings of the* 1^{st} *International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing, San Francisco, CA*, April, 2001.

[17] B. Karp and H. T. Kung. Gpsr: greedy perimeter stateless routing for wireless networks. *In Proceedings of International Conference on Mobile Computing and Networking (Mobicom), Boston, Massachusetts, USA*, August 2000.

[18] Y. Yu, R. Govindan, and D. Estrin. Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks. *Technical report, University of California, Los Angeles*, 2001.

[19] H. Luo, F. Ye, J. Cheng, S. Lu, and L. Zhang. Ttdd: a two-tier data dis-semination model for large-scale wireless sensor networks. *In Proceedings of International Con-ference on Mobile Computing and Networking (MobiCom), Atlanta, Georgia, USA*, September 2002.

[20] K. Akkaya and M. Younis. A survey on routing protocols for wireless sensor networks. *Elsevier Ad Hoc Network Journal*, 3/3:325–349, 2005.

[21] Ralf Steinmetz and Klaus Wehrle. *Peer-to-Peer Systems and Applications.* Springer, October 25, 2005.

[22] V. Muthusamy. An introduction to peer-to-peer networks. *Presentation for MIE456 - Information Systems Infrastructure II*, October 30, 2003.

[23] R. Schollmeier. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In *P2P '01: Proceedings of the First International Conference on Peer-to-Peer Computing (P2P'01)*, page 101, Washington, DC, USA, 2001. IEEE Computer Society.

[24] Napster. http://free.napster.com/.

[25] Kazza. http://www.kazaa.com/us/index.htm.

[26] Gnuttela. http://www.gnutella.com/.

[27] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International J. Supercomputer Applications*, 15(3), 2001.

[28] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration. *http://www.globus.org/research/papers/ogsa.pdf, citeseer.ist.psu.edu/foster02physiology.html*, 2002.

[29] R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, A. Frohner, A. Gianoli, K. L″orentey, and F. Spataro. Voms: An authorization system for virtual organizations. *In the 1st European Across Grids Conference, Santiago de Compostela*, Feb. 2003.

[30] D. Chadwick and O.Otenko. The permis x.509 role based privilege management infrastructure. *In 7th ACM Symposium on Access Control Models and Technologies*, 2002.

[31] Gordon S. Good and Mark C. Smith. *Understanding and Deploying LDAP Directory Services*. Addison-Wesley Professional, 2003.

[32] M. R. Thompson, A. Essiari, and S. Mudumbai. Tcertificate-based authorization policy in a pki environment. *ACM Trans. Inf. Syst. Secur.*, 6(4):566–588, 2003.

[33] M. Lorch, D. Adams, D. Kafura, M. Koneni, A. Rathi, and S. Shah. The prima system for privilege management, authorization and enforcement in grid environments. *In Proceedings of the 4th Int. Workshop on Grid Computing - Grid 2003, Phoenix, AZ, USA*, 2003.

[34] OASIS. extensible access control markup language, 2005.

[35] Internet Resource. Oasis standards. *http://www.oasis-open.org/specs/index.php*, 2007.

[36] P. Bonatti and P. Samarati. Regulating service access and information release on the web. *In CCS '00: Proceedings of the 7th ACM conference on computer and communications security, ACM Press*, page 134–143, 2000.

[37] N. Li and J. Mitchell. A role-based trust-management framework. *In DARPA Information Survivability Conference and Exposition (DISCEX), Washington, D.C.*, Apr. 2003.

[38] R. Gavriloaie, W. Nejdl, D. Olmedilla, K. E. Seamons, and M. Winslett. No registration needed: How to use declarative policies and negotiation to access sensitive resources on the semantic web. *In 1st European Semantic Web Symposium (ESWS 2004), Heraklion, Crete, Greece, Springer*, 3053 of Lecture Notes in Computer Science:342–356, may 2004.

[39] M. Y. Becker and P. Sewell. Cassandra: distributed access control policies with tunable expressiveness. *In 5th IEEE International Workshop on Policies for Distributed Systems and Networks, Yorktown Heights*, June 2004.

[40] P. A. Bonatti and D. Olmedilla. Driving and monitoring provisional trust negotiation with metapolicies. *In 6th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2005), Stockholm, Sweden, IEEE Computer Society*, pages 14–23, jun 2005.

[41] J. Basney, W. Nejdl, D. Olmedilla, V. Welch, and M. Winslett. Negotiating trust on the grid. *In 2nd WWW Workshop on Semantics in P2P and Grid Computing,New York, USA*, may 2004.

[42] T. Grandison and M. Sloman. Survey of trust in internet applications. *IEEE Communications Surveys*, 3(4), 2000.

[43] E. Damiani, S. D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. *In Proceedings of ACM Conference on Computer and Communications Security*, page 202–216, 2002.

[44] S. Lee, R. Sherwood, and et al. Cooperative peer groups in nice. *IEEE Infocom, San Francisco, USA*, 2003.

[45] M. Gupta, P. Judge, and et al. A reputation system for peer-to-peer networks. *Thirteenth ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video, Monterey, California.*, 2003.

[46] S. Kamvar, M. Schlosser, and et al. The eigentrust algorithm for reputation management in p2p networks. *WWW, Budapest, Hungary*, 2003.

[47] K. Aberer and Z. Despotovic. Managing trust in a peer-2-peer information system. *In Proceedings of 10th International Conference on Information and Knowledge Management*, page 310–317, 2001.

[48] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. Eigenrep: Reputation management in p2p networks. *In Proceedings of 12th International WWW Conference*, page 640–651, 2003.

[49] C. Duma, N. Shahmehri, and G. Caronni. Dynamic trust metrics for peer-topeer systems. *In Proceedings of 2nd IEEE Workshop on P2P Data Management, Security and Trust (in connection with DEXA'05)*, August 2005.

[50] L. Kagal, T. Finin, and A. Joshi. A policy based approach to security for the semantic web. *In Proceedings of the 2nd International Semantic Web Conference, Sanibel Island, Florida,USA*, Oct. 2003.

[51] G. Tonti, J. M. Bradshaw, R. Jeffers, R. Montanari, N. Suri, and A. Uszok. Semantic web languages for policy representation and reasoning: A comparison of kaos, rei and ponder. *In Proceedings of the 2nd International Semantic Web Conference, Sanibel Island, Florida,USA*, Oct. 2003.

[52] J. Sabater and C. Sierra. Regret: A reputation model for gregarious societies. *4th Workshop on Deception, Fraud and Trust in Agetn Societies, Montreal, Canada.*, 2001.

[53] J. Pujol, R. Sanguesa, and et al. Extracting reputation in multi agent systems by means of social network topology. *First International Joint Conference on Autonomous Agents and Multi-Agent Systems, Bologna, Italy.*, 2002.

[54] Milan Petkovic and Willem Jonker. *Security, Privacy and Trust in Modern Data Management (Data-Centric Systems and Applications).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[55] Oluwafemi Ajayi, Richard Sinnott, and Anthony Stell. Formalising dynamic trust negotiations in decentralised collaborative e-health systems. In *ARES '07: Proceedings of the The Second International Conference on Availability, Reliability and Security*, pages 3–10, Washington, DC, USA, 2007. IEEE Computer Society.

[56] Vladimir Batagelj and Andrej Mrvar. Pajek analysis and visualization of large networks. *Proceedings Graph Drawing*, 41:477–478, 2002.

[57] R. Tarjan. Depth first search and linear graph algorithms. *SIAM Jurnal of Computing*, 1(2):146–160, June 1972.

[58] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. McGraw-Hill Book Company, Boston, MA, 2001.

[59] Esko Nuutila and Eljas Soisalon-Soininen. On finding the strongly connected components in a directed graph. *Information Processing Letters*, 49(1), 1994.

[60] Message Passing Interface Forum. Mpi standards. *http://www.mpi-forum.org/docs/docs.html*, 2003.

[61] S. Marti and H. Garcia-Molina. Taxonomy of trust: Categorizing p2p reputation systems. *Computer Networks*, 50, Issue 4, Management in Peer-to-Peer Systems:472–484, 15 March 2006.

[62] P. Maniatis, M. Roussopoulos, T. Giuli, D. Rosenthal, M. Baker, and Y. Muliadi. Preserving peer replicas by rate-limited sampled voting. *Technical Report arXiv:cs.CR/0303026, Stanford University*, 2003.

[63] Michal Feldman, Kevin Lai, Ion Stoica, and John Chuang. *Robust incentive techniques for peer-to-peer networks*. ACM Press, New York, NY, USA, 2004.

[64] Sergio Marti and Hector Garcia-Molina. *Limited reputation sharing in P2P systems*. ACM Press, New York, NY, USA, 2004.

[65] K. Lai, M. Feldman, I. Stoica, and J. Chuang. Incentives for cooperation in peer-to-peer networks. *In Workshop on Economics of Peer-toPeer Systems*, 2003.

[66] C. Hoff and Ulf Wehling. econciliation of structured data in multihop manets. *4th International Conference on Advances in Mobile Computing and Multimedia, MoMM 2006, Yogyakarta, Indonesia*, December 2006.

[67] S. Garfinkel. Pgp: Pretty good privacy. *O'Reilly & Associates, Inc.*, 1995.

[68] W. de Nooy, A. Mrvar, and V. Batagelj. Exploratory social network analysis with pajek. *CUP, ESNA page*, January 2005.

[69] U. V. Catalyurek and C. Aykanat. Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication. *Parallel and Distributed Systems, IEEE Transactions on*, 10(7):673–693, 1999.

[70] A. George and J.W.H. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice Hall, 1981.

[71] S.C. Freeman and L.C. Freeman. The networkers network: A study of the impact of a new communications medium on sociometric structure. *Social Science Research Reports No.46. Irvine, CA: University of California.*

[72] Bluetooth.com. Bluetooth specification documents. *http://www.bluetooth.com/*, 2007.

[73] Sun-Developer-Network. Java me technology apis & docs. *http://java.sun.com/javame/reference/apis.jsp*, 2007.

[74] ČVUT-Ericsson-Vodafone R&D Centre (RDC). *http://www.feld.cvut.cz/vv/tymy/rdc.html.*

List of Author's References

[75] R. Špánek. Security in mobile environment. *Doktorandský den '04 - (Hakl, F.), MATFYZPRESS*, pages 149–155, 2004.

[76] R. Špánek. Sharing information in a large network of users. *Doktorandský den '05 - (Hakl, F.), MATFYZPRESS*, pages 134–140, 2005.

[77] R. Špánek. Data pozičně závislá a jejich dopad v mobilních databázích. *ITAT'2005, Information Technologies - Applications and Theory - (Vojtáš, P.)*, pages 273–278, 2005.

[78] R. Špánek. Web search engines and linear algebra. *ICS AS CR - (Technical Report, V-975)*, 2006.

[79] R. Špánek. Rollingball: Energy and qos aware protocol for wireless sensor networks. *SOFSEM 2006. Theory and Practice of Computer Science. Prague : Institute of Computer Science AS CR, 2006 - (Wiedermann, J.; Tel, G.; Pokorný, J.; Bieliková, M.; Štuller, J.)*, pages 166–173, 2006.

[80] R. Špánek and M. Tůma. Secure grid-based computing with social-network based trust management in the semantic web. *Neural Network World, International Journal on Neural and Mass-Parallel Computing and Information Systems*, 16(6):475–488, December 2006.

[81] R. Špánek. Security, privacy and trust in (semantic)web. *Inteligentní modely, algoritmy, metody a nástroje pro vytváření sémantického webu. Praha : Ústav informatiky AV ČR,(Štuller, J.; Linková, Z.)*, pages 114–122, 2006.

[82] R. Špánek. Security model based on virtual organizations for distributed environments. *Doktorandský den '06. Praha : Ústav informatiky AV ČR & MATFYZPRESS- (Hakl, F.)*, pages 164–171, 2006.

[83] R. Špánek and M. Tůma. Secure grid-based computing with social-network based trust management in the (semantic) web. *MoMM2006 & iiWAS2006, Frontiers in Mobile and Web Computing. Wien : Österreichische Computer Gesselschaft - (Barolli, L.; Abderazek, B.; Grill, T.; Nguyen, T.; Tjondronegoro, D.),Yogyakarta*, pages 663–667, 2006.

[84] R. Špánek and M. Tůma. Sdílení dat v prostředí s nehomogenními skupinami uživatelů. *Information Technologies - Applications and Theory. Košice : Prírodovedecká fakulta, Univerzita P. J. Šafárika, (Vojtáš, P.)*, pages 145–149, 2006.

[85] R. Špánek. Maintaining trust in large scale environments. *Doktorandské dny '07. Praha : Ústav informatiky AV ČR, v. v. i. & MATFYZ-PRESS,(Hakl, F.)*, pages 94–102, 2007.

[86] Martin Řimnáč, R. Špánek, and Zdeňka Linková. Sémantický web: vize globálního úložiště dat? *DATAKON 2007. Brno : Masaryk University, (Popelínský, L.; Výborný, O.)*, pages 176–186, 2007.

[87] Martin Řimnáč, R. Špánek, and Zdeňka Linková. Semantic web: Vision of distributed and trusted data environment? *ICDIM 2007, Lyon: INSA, 2007 - (Youakim Badr, Richard Chbeir, Pit Pichappan)*, pages 627–634, 2007.

[88] R. Špánek. Reputation system for large scale environment. *ICDIM 2007, Lyon: INSA, 2007 - (Youakim Badr, Richard Chbeir, Pit Pichappan)*, pages 627–634, 2007.

[89] R. Špánek and P. Kovář and P. Pirkl. The BlueGame Project: Ad-hoc Multilayer Mobile Game with Social Dimension. *CONEXT 2007, New York: Columbia University, 2007*

[90] R. Špánek, Martin Řimnáč, Zdeňka Linková. On Creating a Trusted and Distributed Data Source Environment. *SOFSEM 2008, Slovak Republic, Nový Smokovec, 2008.*