CAESAR
A proposed method for evaluating security in component-based distributed information systems

Thesis project done at Information Theory, Linköping University

by

Mikael Peterson

LiTH-ISY-EX-3581-2004
Linköping, 2004
CAESAR
A proposed method for evaluating security in component-based distributed information systems

Thesis project done at Information Theory,
Linköping University

by

Mikael Peterson

LiTH-ISY-EX-3581-2004
Linköping, 2004

Examiner
Viiveke Fåk

Supervisors
Jonas Hallberg and Amund Hunstad
CAESAR - A proposed method for evaluating security in component-based distributed information systems

Mikael Peterson

Background: The network-centric defense requires a method for securing vast dynamic distributed information systems. Currently, there are no efficient methods for establishing the level of IT security in vast dynamic distributed information systems.

Purpose: The target of this thesis was to design a method, capable of determining the level of IT security of vast dynamic component-based distributed information systems.

Method: The work was carried out by first defining concepts of IT security and distributed information systems and by reviewing basic measurement and modeling theory. Thereafter, previous evaluation methods aimed at determining the level of IT security of distributed information systems were reviewed. Last, by using the theoretic foundation and the ideas from reviewed efforts, a new evaluation method, aimed at determining the level of IT security of vast dynamic component-based distributed information systems, was developed.

Results: This thesis outlines a new method, CAESAR, capable of predicting the security level in parts of, or an entire, component-based distributed information system. The CAESAR method consists of a modeling technique and an evaluation algorithm. In addition, a Microsoft Windows compliant software, ROME, which allows the user to easily model and evaluate distributed systems using the CAESAR method, is made available.

Keyword

computer security, distributed systems, modeling, security evaluation
Acknowledgements

In addition to steady support from my supervisors, Jonas Hallberg and Amund Hunstad, and from my examiner Viiveke Fåk, I have also received invaluable help from my colleague, Martin Karresand, throughout the work of this thesis.
Abstract

Background: The network-centric defense requires a method for securing vast dynamic distributed information systems. Currently, there are no efficient methods for establishing the level of IT security in vast dynamic distributed information systems.

Purpose: The target of this thesis was to design a method, capable of determining the level of IT security of vast dynamic component-based distributed information systems.

Method: The work was carried out by first defining concepts of IT security and distributed information systems and by reviewing basic measurement and modeling theory. Thereafter, previous evaluation methods aimed at determining the level of IT security of distributed information systems were reviewed. Last, by using the theoretic foundation and the ideas from reviewed efforts, a new evaluation method, aimed at determining the level of IT security of vast dynamic component-based distributed information systems, was developed.

Results: This thesis outlines a new method, CAESAR, capable of predicting the security level in parts of, or an entire, component-based distributed information system. The CAESAR method consists of a modeling technique and an evaluation algorithm. In addition, a Microsoft Windows compliant software, ROME, which allows the user to easily model and evaluate distributed systems using the CAESAR method, is made available.
# Table of contents

1 INTRODUCTION ........................................................................................................... 3
   1.1 MOTIVATION ........................................................................................................ 3
   1.2 PROBLEM FORMULATION .............................................................................. 4
   1.3 LIMITATIONS .................................................................................................... 5
   1.4 METHODOLOGY ............................................................................................... 5
   1.5 CONTRIBUTIONS ............................................................................................. 6
   1.6 LAYOUT ............................................................................................................. 6

2 BACKGROUND ............................................................................................................ 9
   2.1 DISTRIBUTED INFORMATION SYSTEMS ...................................................... 9
      2.1.1 Basic definitions ......................................................................................... 9
      2.1.2 Modeling ...................................................................................................... 10
      2.1.3 Dimensions ............................................................................................... 12
   2.2 IT SECURITY ...................................................................................................... 14
      2.2.1 Basic definitions ....................................................................................... 14
      2.2.2 Dimensions ............................................................................................... 16
      2.2.3 Measurability ............................................................................................ 17
      2.2.4 Challenges ................................................................................................. 20

3 RELATED WORK .................................................................................................... 21
   3.1 ANDERSSON’S SECURITY EVALUATION FRAMEWORK .............................. 21
      3.1.1 Component security evaluation ................................................................. 22
      3.1.2 System security evaluation ........................................................................ 23
      3.1.3 Discussion .................................................................................................. 26
   3.2 WANG AND WULF’S SECURITY MEASUREMENT FRAMEWORK ................. 27
      3.2.1 Selection of definition, units and scales ..................................................... 27
      3.2.2 Estimation methodology ......................................................................... 28
      3.2.3 Validation ................................................................................................... 31
      3.2.4 Discussion .................................................................................................. 31
   3.3 ANNA STJERNEBY’S SYSTEM COMPONENT CATEGORIZATION ..................... 34
      3.3.1 Discussion .................................................................................................. 35

4 THE CAESAR METHOD ............................................................................................ 37
   4.1 SCOPE ................................................................................................................ 38
   4.2 MODELING TECHNIQUE .................................................................................. 39
      4.2.1 Overview .................................................................................................... 39
      4.2.2 Component classes .................................................................................... 41
      4.2.3 Security level estimate .............................................................................. 42
      4.2.4 Traffic control estimate ............................................................................ 43
      4.2.5 Physical relations ....................................................................................... 44
      4.2.6 Logical relations ....................................................................................... 44
# Table of figures

| Figure 1: A model of a client-server architecture | 11 |
| Figure 2: A model of a peer process architecture | 11 |
| Figure 3: An example of a physical-relationship focused model of a distributed system | 12 |
| Figure 4: An example of layers of an IT system | 12 |
| Figure 5: The micro - macro dimension | 13 |
| Figure 6: Dimensions to consider when modeling a distributed system architecturally | 14 |
| Figure 7: Relating PDR to CIA | 16 |
| Figure 8: Gollmann’s human-machine dimension | 16 |
| Figure 9: Dimensions to consider when defining IT security | 17 |
| Figure 10: Measurable entities of an IT system in a security context | 18 |
| Figure 11: Modeled and real securability, security and risk | 19 |
| Figure 12: An example of Andersson’s model of distributed information systems | 23 |
| Figure 13: An example of Wang and Wulf’s system decomposition | 29 |
| Figure 14: General workflow of CAESAR | 37 |
| Figure 15: Building blocks of the CAESAR modeling technique | 40 |
| Figure 16: An example of a graphical representation of a CAESAR model | 45 |
| Figure 17: The CAESAR evaluation algorithm’s main concepts and their relations | 46 |
| Figure 18: Factors that influence the overall security level of the system | 47 |
| Figure 19: Factors that influence system-dependent security level | 48 |
| Figure 20: Examples showing when to use NSC and LSC | 49 |
| Figure 21: Factors that influence neighboring security contribution | 51 |
| Figure 22: An example of neighboring security contribution as an arbitrary function | 53 |
| Figure 23: Factors that influence logical security contribution | 54 |
| Figure 24: Factors that influence comparable security level | 55 |
| Figure 25: Screenshot of the ROME software | 64 |
| Figure 26: Adding components using the ROME software | 65 |
| Figure 27: Adjusting representation using the ROME software | 65 |
| Figure 28: An example of a modeled system using the ROME software | 66 |
"alea iacta est"

Gajus Julius Caesar  
Po Valley, Italy, January 10, 49 BC
1 Introduction

January 7, 49 BC, Gajus Julius Caesar, at the time governor of province Gaul of the Roman Empire, received a message in Ravenna. The message was from the Roman Senate that demanded him to hand over his ten legions to a new governor. Caesar had to choose; endure a humiliating prosecution or start a fierce rebellion.

January 10, Caesar crossed the southern border of his province into Italy and thereby started the second roman civil war. His chances did not look great; nine of his ten legions were left in Gaul. Two weeks later Caesar was master of Italy.

During the next few years, Caesar defeated the republican forces of Italy, Greece, North Africa, and Spain. After five years, at the beginning of 44 BC, Caesar made himself dictator of the Roman Empire for life.

His family name Caesar was taken by several of his ancestors of the Juliu-Claudian dynasty and was later made a formal title that was to be used for more than half a millennium. (NE, 2004)

1.1 Motivation

Much has changed since the year 50 BC. Currently, fully functional distributed information systems are as essential infrastructure as operational highways and working healthcare, in peaceful times, and in war. Even though the fundamental nature of warfare will never change, some principles definitely have changed with the development of information technology. In addition, for the first time in history, both technology and its applications are no longer pioneered by military organizations. (Alberts, Garstka & Stein, 1999)

Even though Caesar’s victorious platform-centric warfare almost certainly depended heavily on secure transmissions of messages between tactical units, today’s network-centric warfare requires enormously more complex communicational patterns.
Alberts, Garstka & Stein (1999) describes network-centric warfare as

an information superiority-enabled concept of operations that generates increased combat power by networking sensors, decision makers, and shooters to achieve shared awareness, increased speed of command, higher tempo of operations, greater lethality, increased survivability and a degree of self-information superiority into combat power by effectively linking knowledgeable entities in the battlespace.

Alberts, Garstka & Stein (1999) stress that the integration of civilian and military distributed information systems is necessary to enable the concept of network-centric warfare. To make such integration possible, it is utterly essential that there is a reliable method of determining the level of IT security of such distributed information systems. It is also necessary to be able to determine how the level of IT security is affected when new systems are connected or disconnected (Stjerneby, 2002).

- Currently, there are no efficient methods for establishing the level of IT security in vast and dynamic distributed information systems.

Needless to say, no matter how superior Caesar’s platform-centric warfare was at the time of 50 BC, it would stand no chance against the modern network-centric warfare currently being developed.

This thesis will update Caesar’s combat power to the 21 century.

### 1.2 Problem formulation

As described in the previous section; currently, there does not exist any efficient evaluation methods for establishing the level of IT security in vast and dynamic systems. Without such evaluation methods, it is impossible to integrate these systems, as necessary to enable the concept of Network Centric Warfare.

- The main target of this thesis is to design a method, capable of determining the level of IT security of vast dynamic component-based distributed information systems.

This problem formulation was translated into three sub targets:

- Establish a theoretical foundation
  Define IT security, distributed information systems, and related terms. Also, discuss basic measurement and modeling theory.
- **Review previous efforts**
  Using the theoretic foundation; review and assess previous evaluation methods aimed at determining the level of IT security of distributed information systems.

- **Develop a new method**
  Using the theoretic foundation and the ideas from reviewed efforts; develop a new evaluation method aimed at determining the level of IT security of vast dynamic component-based distributed information systems.

### 1.3 Limitations

Since the target of this thesis defines a broad research area that, due to its large size, would be impossible to fully explore, the following scope boundaries were introduced:

- **Evaluation is applied to no smaller part than an atom**
  Atoms of the evaluation method are computers and network components (further called components). No smaller part than a component may be considered by the evaluation method.

- **It is assumed that a reliable component evaluation method exists**
  This thesis will discuss how to aggregate already existing component evaluation results, but not how to generate such results.

- **Systems are considered from an architectural point of view**
  Systems are considered, and therefore modeled, architecturally, rather than from any other point of view (see section 2.1.2, page 10).

- **No non-technical aspects are considered**
  Human users and non-technical infrastructure are not considered by the evaluation method.

### 1.4 Methodology

In order to reach the main target, described in section 1.2, page 4, the work was carried out in seven phases.

- **Review general IT security literature and articles**
- **Review distributed system literature and articles**
- **Review measurement theory literature**
- Review existing published methods aimed at evaluating IT security
- Merge ideas and concepts from previous phases into a new method
- Refine the new method
- Develop demonstration software of the new method

1.5 Contributions

The main results produced by the effort described in this thesis are:

- **A survey of existing methods**
  Reviews and discussions of a selection of the existing methods aimed at determining the level of IT security of technical components or systems.

- **A modeling technique**
  A set of security relevant classes and properties used to model a distributed information system architecturally.

- **An evaluation algorithm**
  An algorithm used to aggregate security relevant properties of a modeled system into a measure of the level of IT security of the entire system.

- **Supporting computer software**
  Windows software supporting the modeling technique and demonstrating the basic functions of the evaluation algorithm.

1.6 Layout

Chapter 2 covers basic IT security and distributed systems theory. It provides the necessary conceptual framework, on which the following chapters depend.

Chapter 3 describes and analyzes earlier methods aimed at assessing security of distributed information systems.

Chapter 4 explains in detail a new, enhanced method aimed at determining the level of IT security of distributed information systems.

Chapter 5 describes the demonstration software designed to explain and aid refining the new method.
Chapter 6 summarizes the results and implications of the previous chapters and outlines the areas suitable for future work.
A proposed method for evaluating security of component-based distributed information systems
2 Background

This chapter defines and explains important concepts that are used in this thesis.

2.1 Distributed information systems

In this section, basic definitions regarding distributed information systems will be described. In addition, basic models describing distributed information systems will be described.

2.1.1 Basic definitions

“A distributed system is one in which components located at networked computers communicate and coordinate their actions only by passing messages” according to Coulouris, Dollimore, and Kindberg (2001). This definition implicates the following characteristics of distributed systems:

- Concurrency of components
- Lack of a global clock
- Independent failures of components

There is a notion that distributed systems, in contrast to computer networks in general, should be transparent to the user and appear as one local machine.

The term distributed information system is used to emphasize the distribution of information in the system and the fact that users and organizations are considered a part of the system. (Andersson, 2004)
2.1.2 Modeling

There are several different ways that a distributed system could be modeled. Two of the most common categories of modeling techniques are *architectural models*, which focus on the placement of the parts of a distributed system and the relationship between the parts, and *fundamental models*, which are concerned with a more formal description of the properties that are common in all of the architectural models. (Coulouris, Dollimore, & Kindberg, 2001)

Examples of fundamental models are *interaction models*, which deal with messages and synchronization and *failure models*, which define and classify faults as a basis for analysis of their effects. (Coulouris, Dollimore, & Kindberg, 2001) This thesis will focus primarily on architectural modeling techniques.

Modeling the structure of a distributed system requires abstraction of the functions of the individual components of the system. A common abstraction is the simplification of component functions into three types of *processes*: (Coulouris, Dollimore, & Kindberg, 2001)

- **Server processes**
  - Processes replying to other processes’ requests.

- **Client processes**
  - Processes requesting data from servers.

- **Peer processes**
  - Processes cooperating and communicating in a symmetrical manner.

One could of course think of other types of processes, both combinations of the above and others. After simplifying the component functions into processes, it is possible to investigate their relationships. This may be done from at least two different aspects: (Coulouris, Dollimore, & Kindberg, 2001)

- **Focusing on the physical relationship between components**
  - Consider the placement of components across a network of computers (leaning more towards a hardware layer perspective).

- **Focusing on the logical relationship between components**
  - Consider the functional roles of the components and the patterns of communication between them (leaning more towards an application layer perspective).

After deciding on process classification for each component and the relationships between the components, it is possible to model the system
architecturally. When focusing on the logical relationship between computers, there are two types of architectural styles often mentioned:

- **The client-server architecture**
- **The peer process architecture**

Client-server architecture is illustrated in Figure 1.

Peer process architecture is illustrated in Figure 2.

Modeling a distributed system with a physical relationship focus could instead appear as in Figure 3.
These different focuses or aspects are examples of different *dimensions* by which a modeling technique may be described.

### 2.1.3 Dimensions

These illustrate that there is no single way to model a distributed system. Deciding on an appropriate modeling technique is dependent of which aspects of a distributed system that are to be analyzed.

A common differentiation between such aspects is visible in the *classic layered model* of an IT system (see Figure 4). A distributed system may be perceived at any – or any combination – of these layers. (Gollmann, 1999)

Another differentiation between aspects is the *micro-macro dimension*. The micro-macro dimension makes it possible to clarify which the smallest parts – the atoms – of the modeling technique are. Figure 5 might help to illustrate this. On the lowest level, level C, computer components, such as network adapters and
operating systems, are located. These components make up computers and network components, which are located on level B. When computers and network components are connected, they build distributed information systems, which are located on level A.

Yet another, previously implicated, differentiation between different aspects is the physical-logical dimension. When modeling a distributed system using an architectural modeling technique, it is important to keep these dimensions in mind:

- **Hardware – software dimension**
- **Micro – macro dimension**
- **Physical – logical dimension**

One does not always have to choose a single point on one of these dimensions, a segment or several segments will sometimes work just as well. Every model covers a part of the layer-relationship-size space illustrated in Figure 6. A richer model obviously covers a greater part.
2.2 IT security

In this section, basic definitions and dimensions regarding IT security will be described. At the end of this section, basic measurement theory applied to IT security will be reviewed as well as the current challenges within this field of research.

2.2.1 Basic definitions

There are many different interpretations of IT security. These different definitions exist mainly because what are considered essential security issues vary between different applications. Therefore, different interpretations should not be considered redundant. It is, however, important to decide on a single definition of IT security regarding each application. (Wang & Wulf, 1997)

There are some common definitions of IT security, all within the general notion that the term is about the protection of information assets and the services delivered by information systems. An expression for such a disperse interpretation is the definition “prevention and detection of unauthorized actions by users of a computer system” (Gollmann, 1999).

Adding one required ability to the definition above, makes a common categorization of IT security into three protective measures often abbreviated PDR: (Gollmann, 1999)

- **Prevention**
  Measures to prevent assets from being manipulated.
• **Detection**  
  Measures to detect attempts to manipulate assets.

• **Reaction**  
  Measures to block or minimize the damage caused by such attempts.

Sometimes *survival* is considered a group of measures, orthogonal to all of the above. Survival denotes measures to recover from failures and security breaches.

PDR is a quite blurred and hardly measurable definition. One of the important decisions to make when deciding upon an adequate definition of IT security is which security abilities of a system to regard. (Wang & Wulf, 1997)

A common way to further categorize the concept of security is to make a list of security characteristics that should not be compromised. The categorization CIA, consists of such a list, which breaks the concept of security down into three characteristics, that a system should try to uphold: (Gollmann, 1999)

• **Confidentiality**  
  No unauthorized disclosure of information.

• **Integrity**  
  No unauthorized modification of information or system.

• **Availability**  
  No unauthorized withholding of information or resources.

Summarizing the above categorizations, CIA is about how information assets may be compromised and PDR is about abilities required to maintain system security (Andersson, 2004). The notion that CIA and PDR present different perspectives on security is shown in Figure 7.
A proposed method for evaluating security of component-based distributed information systems

<table>
<thead>
<tr>
<th></th>
<th>Confidentiality</th>
<th>Integrity</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Prevention</strong></td>
<td>Prevention of unauthorized disclosure of information</td>
<td>Prevention of unauthorized modification of information</td>
<td>Prevention of unauthorized withholding of information</td>
</tr>
<tr>
<td><strong>Detection</strong></td>
<td>Detection of unauthorized disclosure of information</td>
<td>Detection of unauthorized modification of information</td>
<td>Detection of unauthorized withholding of information</td>
</tr>
<tr>
<td><strong>Reaction</strong></td>
<td>Reaction when unauthorized disclosure of information</td>
<td>Reaction when unauthorized modification of information</td>
<td>Reaction when unauthorized withholding of information</td>
</tr>
</tbody>
</table>

Figure 7: Relating PDR to CIA

### 2.2.2 Dimensions

IT security, regardless of which of the above definitions preferred, may be perceived from different points of view. The human-machine dimension (Gollmann, 1999) indicates the possibility of such different points of view (see Figure 8). One could design, implement and evaluate IT security on any point on the line between total human orientation and total machine orientation.

The human-machine dimension is to some extent correlated to the classic layered model of an IT system, as illustrated in Figure 4, where the bottom layer is the hardware layer (machine oriented), and the top layer is the application layer (human oriented).
The hardware layer consists of the computer hardware supporting the other layers. In a security context, one could of course imagine a continuation of the layer model further down, describing buildings, power supply, et cetera.

When defining IT security, it is important to keep these dimensions in mind:

- **Human – machine dimension**
- **Hardware – software dimension**

One does not always have to choose a single point on one of these dimensions, a segment or several segments will sometimes work just as well. Every definition covers an area on the layer-complexity plane illustrated in Figure 9. A richer model obviously covers a greater area.

![Layer Complexity Diagram](image)

**Figure 9: Dimensions to consider when defining IT security**

Considering the human-machine dimension, and the imagined continuation of the layer model, it is important to keep in mind that technical solutions are merely a part of everything relating to IT security.

### 2.2.3 Measurability

IT security measurement is a complex issue. It is of course impossible to measure the security level of an IT system directly as when measuring the weight of a physical object by weighing it. Instead, one has to measure either factors that cause a certain security level, or consequences that are an effect of a certain security level, see Figure 10.
A proposed method for evaluating security of component-based distributed information systems

Factors that correlate to the security level of an IT system could be the number of users or if the network is connected to the Internet. Consequences that correlate to the security level of an IT system could be the number of unauthorized retrievals of certain information in the past month or the number of successful attempts to withhold certain information the past year.

Based on measured factors and/or consequences it is then possible to estimate the security level of an IT system. Since predictive models are the focus of this thesis, consequences are disregarded as an input for security estimation in the rest of this thesis. The verbs measure, estimate, evaluate and assess in implicit or explicit conjunction with the term IT security will be regarded somewhat synonymous throughout this thesis and denote the process of collecting measurable security factors and estimating the security level.

In order to estimate the security of an IT system, one first needs to answer the following questions:

- **What system properties are to be measured?**
  Decide on relevant security properties to parameterize. If measuring the weather, how warm it is might be a relevant property.

- **What measurable magnitude is to be measured for each parameter?**
  Decide on a suitable magnitude for each parameterized property. Continuing the previous example, how warm it is, might be measured as the temperature in shadow.

- **What representation is to be used for each magnitude?**
  Decide on a suitable unit and scale type for each magnitude. Continuing the example, the temperature might be represented as a ratio scale, as Kelvin, or as an interval scale, as Fahrenheit or centigrade. (Roberts, 1984)

When answering the first question, it is essential to keep the PDR-CIA-table (Figure 7) and the definition dimensions (Figure 9) in mind. These set the boundaries for a definition of a measurable security level. However, it is important to differentiate between securability, security level and risk level (Andersson 2003). Figure 11 shows the difference between these three concepts.
It is possible to estimate securability on an offline system without a context. The level of security in a system, however, is determined by how it performs and is operated online. If the system is put in context, the risk can be estimated. It is, again, important to remember that securability, security, and risk, due to their complex nature, always need to be aggregated magnitudes, depending on more detailed and measurable properties (Wang & Wulf, 1997).

![Figure 11: Modeled and real securability, security and risk](image)

To some degree, the term *representation* in the third question above could be regarded as equivalent to the term *metric*. The term metric will occur later in quotation of related work in this thesis.

When answering the third question it is important to differentiate between different scale types: *nominal scales, ordinal scales, interval scales, ratio scales* and *absolute scales*. Nominal scales are merely labels. Ordinal scales preserve ordering among classes (good for hardness, air quality, etc). Interval scales preserve ordering and differences between classes (good for intelligence test scores, calendar time, etc). Ratio scales preserve ordering, differences and ratios among classes (good for mass, loudness, etc). Ratio measurement mappings must start at an absolute zero and increase at equal intervals, called units. Absolute scales are completely mapped onto the described entity (good for counting).
2.2.4 Challenges

In their paper, Computer Security is Not a Science, Greenwald et al (2003) argue that despite the importance of current principles of IT security, these principles do not yield any way to determine the security level of a system. To make real progress in the field of IT security there is a need to focus on three key areas:

- **Develop better experimental techniques**
- **Develop better metrics of security**
- **Develop models with real predictive power**

This, according to Greenwald et al (2003), is entirely dependent on the establishment of a scientific foundation for future security research. Establishing such a scientific foundation is undoubtedly one of the major challenges in the field of IT security research.

The proceedings of Applied Computer Security Associates Workshop on Information Security System Scoring and Ranking (ACSA 2002) delivers some interesting observations on the current situation in the research area of developing better metrics of security. These observations are listed below (ACSA 2002):

- **There is a need to comprehend a complex reality**
  Processes, procedures, tools and people all interact to generate assurance. Security measures integrating these aspects remain critical.

- **There is a need for multifaceted measures**
  No single measure successfully quantifies the assurance in a system. Multiple measures are needed and need to be refreshed frequently.

- **Previous attempts have not been successful**
  Previous efforts to combine these insights, among them the Trusted Computer System Evaluation Criteria (TCSEC 1985) and the Common Criteria (CC 1999) are not successful.

To further complicate matters, Andersson, Hallberg and Hunstad (2003), concludes that IT security tends to be discussed either on a high level of abstraction or on the concrete system component level, with little or no abstraction. Closing the gap between general specifications and detailed implementations is yet another demanding challenge in the field of IT security research.

The new method presented later in this thesis aims at diminishing that gap.
3 Related Work

There has been little substantial work done previously in the field of evaluating security in distributed information systems. As mentioned in chapter two, the discussion seems to be on either a very high level of abstraction or on an utterly concrete level with little or no abstraction, probably due to the enormous complexity in the task of embracing the entire field.

This chapter reviews a few relevant, currently available methods aimed at evaluating security of distributed information systems, and analyzes them from the perspective of this thesis. The terms and definitions used in this chapter are obtained from respective work and may differ from the rest of this thesis.

3.1 Andersson’s Security Evaluation Framework

The first, and perhaps in this case, most relevant method, is one developed by Richard Andersson (2003) at the Swedish Defense Research Agency. This method is a part of his framework for evaluating security of distributed information systems. The framework aspires, as Anderson puts it, to handle all possible aspects that may affect the security of a distributed information system, and to divide the evaluation process into different parts, making it less complex.

The framework consists of a system modeling technique, and an evaluation method. The information system is modeled by dividing it into increasingly smaller parts, then evaluating the separate parts and finally building the modeled system by combining the smaller parts until the whole system is built. Anyone interested in reading more about Anderson’s security evaluation framework is referred to Anderson’s work. (Richard Andersson, 2003)
3.1.1 Component security evaluation

The foundation in Andersson’s work is the component evaluation technique that is based on a customization of the Security Functional Requirements of the Common Criteria. For those not familiar with Common Criteria, it represents the outcome of international efforts to align and improve the existing European Information Technology Security Evaluation Criteria (ITSEC 1991) and North American Trusted Computer Evaluation Criteria (TCSEC 1985) criteria towards a common standard for carrying out security evaluations. (CC 1999)

In a few words, Andersson’s customization of the Security Functional Requirements can be perceived as a catalogue of requirements that a technical component should apply to, in order to be recognized as secure.

Representation and interpretation

Each requirement has been mapped to both CIA and PDR (described in 2.2.1, page 14), making it possible to interpret the evaluation result of each and all requirements in terms of confidentiality, integrity and availability as well as prevention, detection and reaction. All requirements are also categorized into 11 classes (communication, resource utilization, trusted path/channel, security management, cryptographic support, identification and authentication, user data protection, privacy, security audit, protection), making it possible to interpret the result in terms of these classes as well.

Thus, the result of the evaluation can be presented in many different ways, but most usefully as either an 11-dimensional vector, where each element corresponds to one of the 11 security classes or a 33-dimensional vector, where each element corresponds to the confidentiality, integrity or availability factor for one of the 11 security classes.

Regardless of which form of results is preferred each of the 11 or 33 elements are assigned a value ranging from 0 to 1 as a result of the evaluation. Andersson suggests that this value should be considered somewhat of a probability estimate that a random attack would not succeed, in turn implying that probability may vaguely be regarded as the metric for the security value.

The value 0 would then indicate a significant possibility of vulnerability and the value 1 would in contrast indicate that the evaluated component or system is as secure as possible, regarding the specific security functionality evaluated. A higher value should always be considered as a sign of a more secure component or system, but to what degree it is more secure remains unclear.
Andersson discusses the possibility of other representations as well, and one could easily think of a 3-dimensional representation as CIA or PDR per component, or even a 1-dimensional representation in some cases.

### 3.1.2 System security evaluation

The system security evaluation method drafted in Andersson’s thesis is independent of the previous selection of component security representation; it is up to the user to decide in which aspect to evaluate the security of the system.

**Modeling the system**

To evaluate the security of a distributed information system, Andersson first draws a graph, architecturally representing the distributed information system that is the target of evaluation (see Figure 12). The graph consists of nodes that represent components or systems of components, and links between the nodes that binds the systems and components together. Figure 12 is somewhat misleading, since the security indicator is regularly a vector, not a scalar.

![Figure 12: An example of Andersson’s model of distributed information systems](image)

Each link in the graph is denoted with a value, ranging from 0 to 1, representing its importance. Each node is denoted with a *security indicator* (SI), representing its level of security in a particular aspect. For some reason a new terminology is chosen here, but security indicators are simply a new way of describing any of the vector representations mentioned previously.
Merging components into sub-systems

The method combines component security indicators in various manners, depending on the system characteristics, and then returns a new security indicator, of the same form as the ones combined, signifying the security of the evaluated sub-system. Because security indicators are used as both input and output, it is possible to apply the method iteratively.

This possibility to evaluate and encapsulate sub-systems in components in the graph is mentioned as one of the features of the model and referred to as merging of components. This would make it possible to get a better overview of the security in different parts of a large distributed information system.

If one would like to explore one of the merged sub-systems more closely, it would then be possible to split that system into several different components again. This Andersson refers to as the possibility to zoom into and out of different parts of the system.

Evaluating sub-systems

Evaluating a system is an iterative process of evaluating sub-systems by merging components with components and the result of that with another component, and so on, until, finally, there is just one component left representing the entire system to be evaluated.

Andersson mentions that the mathematical functions used when evaluating combinations of components must meet certain requirements. Since the proposed security indicators are of probabilistic nature, and ranges from 0 to 1, the functions cannot be allowed to return a value less than 0 or more than 1. Simple addition would therefore not suffice, while a maximum function would.

As previously mentioned, the mathematical functions used when evaluating combinations of components must also be able to handle not only security indicators consisting of merely a single scalar value, but also security indicators represented as vectors. It is also possible to imagine that different vector elements are evaluated with different mathematical functions.

There are four different mathematical functions described to evaluate combinations of components:

- **Cooperative**
  
  Used to combine and evaluate components that have a positive effect on security and are working together.
- **Coexisting**
  Used to combine and evaluate components that have a negative effect on security and have similar functions.

- **Counter effective**
  Used to combine and evaluate components working against each other.

- **Perplexing**
  Used to combine and evaluate components altering each other’s trustworthiness.

The cooperative functions are used when components are working together. If there is a large overlap in the components security functions, such as between a virus protections application and a software-based firewall, the resulting security indicator would be calculated as $SI = \text{Max}(SI_1, SI_2)$. If the components are completely independent, the mathematical function for a union is suggested; $SI = SI_1 + (1 - SI_1) \cdot SI_2$. If the level of dependability could be measured or estimated as $x$, a blend of the previous two functions is suggested; $SI = x \cdot \text{Max}(SI_1, SI_2) + (1 - x) \cdot (SI_1 + (1 - SI_1) \cdot SI_2)$.

The coexisting functions are used when a number of components are essential for the security of the system. For example, when combining multiple users ($SI_i$) of a single computer a minimum-function could be used; $SI = \text{Min}(SI_1, \ldots, SI_N)$. Such a function is appropriate when dealing with components with completely overlapping security functions.

The counter effective functions are used when security functions in one component somewhat negates the security functions in another. Anderson exemplifies this with a firewall ($SI_1$) filtering packets that are protected with cryptography ($SI_2$), and therefore unable to search packets for illicit data. The resulting security indicator would then be calculated $SI = SI_1 \cdot SI_2$.

The perplexing functions are used when the security indicator of one component would become less accurate if it were combined with components of less trustworthiness. This is exemplified with a workstation of a local user ($SI_1$) connected to a workstation of an unknown user ($SI_2$) with a trustworthiness-rating (TW). The resulting security indicator would then be calculated $SI = \text{Min}(SI_1, SI_2 \cdot TW)$.
3.1.3 Discussion

In this section, Andersson’s component evaluation, system modeling, and system evaluation method, are discussed and analyzed in the context of this thesis.

Component evaluation method

Andersson has developed a sophisticated approach for evaluating technical components. In general, the evaluation technique is well organized, and the mapping of customized Common Criteria’s Security Function Requirements onto CIA and PDR helps giving a clear and understandable view of the evaluation result.

However, the component evaluation method has a few negative aspects as well. The primary downside is the absence of a metric.

Since the resulting value or values of the component evaluation are barely semi-probabilistic, or “of probabilistic nature” as Andersson describes it, it is extremely difficult to examine, compare or even interpret the evaluation results – and probably even harder estimating the values to begin with, even though that is well beyond the scope of this thesis. For further theories on measurability, see 2.2.3, page 17. It should also be mentioned that it is a tremendously time-consuming task to evaluate components according to this method.

System evaluation method

The system modeling and evaluation method is certainly more vaguely constructed. However, the introduction of a graph used modeling the system is a daring initiative and both the modeling and evaluation method hold many novel ideas.

Then again, there are a few drawbacks of the system modeling and evaluation method. One of the most obvious ones is the absence of a proper description of how to estimate or interpret the probability values denoted on each link in the graph. It is also unclear exactly what these values correspond to in the actual information system modeled.

Another weakness is that when combining and evaluating components according to different mathematical functions, such as combining two cooperative components with one coexisting, the order in which the combination is performed affects the result. It would certainly increase the credibility of the method if the order was not affecting the result or if the order was strictly enforced by fixed rules.
Obviously many of the disadvantages of the model, has to do with the difficulty of measuring different quantities. In addition, such quantities are the factor describing the level of component dependability and the factor describing the level of functional overlapping between components. These values almost certainly have to be rough estimations, and therefore reduce the precision of the result.

Another shortcoming of the modeling method is that it is unclear what to regard as an atom; the graph shows computers as components of the system, but in the examples, anti-virus software is considered to be a component. This is perhaps not a weakness considering the method in itself, but in the context of this thesis, it is important to define a discrete, smallest element – an atom of the system.

As previously mentioned, the system evaluation method should probably be considered more of an outline, charting out future work, and the analysis here is merely underscoring that.

### 3.2 Wang and Wulf’s Security Measurement Framework

The second method, “Security Measurement Framework”, is invented by Chenxi Wang and William A. Wulf (Wang & Wulf, 1997) at University of Virginia. Their framework is aimed at quantifying security in complex IT systems. It consists of four distinct tasks that need to be executed in following order:

- **Definition of security**
- **Selection of units and scales**
- **Definition of an estimation methodology**
- **Validation of the measures**

#### 3.2.1 Selection of definition, units and scales

Wang & Wulf (1997) argues that the definition of “computer security” is context sensitive (in the sense that each case has unique needs) and must identify a set of security-related attributes that are important to the use of the system. It should also be decided whether the system security is to be represented as a
vector or a single value. If a single value is chosen, there has to be some algorithm, combining the measured attributes into a final value.

Since each attribute may be measured in different ways, it is important to select a unit and scale type for each attribute. Wang and Wulf mention four different scale types: nominal, ordinal, interval and ratio scales (see 2.2.3, page 17).

When deciding on units and scales, Wang and Wulf argue that two issues must be considered:

- **Plausibility**
  Do not use a scale type richer than appropriate considering the information that the measures represent.

- **Accuracy**
  Choose the unit and scale type that generates the least possible measurement errors.

### 3.2.2 Estimation methodology

According to Wang and Wulf, it is virtually impossible to measure end-to-end security properties of IT systems, due to the scopes and structures of those systems. It is, however, possible to develop estimation methodologies.

Wang and Wulf suggest such an estimation methodology, consisting of five steps:

- **Decomposition**
- **Functional relationships**
- **Weighting and priorities**
- **Basic measurements**
- **Component sensitivity analysis**

These stages will be described briefly here. Anyone interested in reading more about the estimation methodology is referred to “Towards a framework for security measurement” (Wang & Wulf, 1997).

**Decomposition**

Estimating the security of a large distributed system is difficult. Analyzing standalone components of the system is easier. Wang and Wulf therefore
suggest a decomposition of the system into smaller parts, using the following algorithm:

1. Identify security-related goals for the system.
2. Identify successive components that are necessary to reach the goals.
3. Repeat the second task for the new components.
4. Terminate the algorithm when it is impossible to identify any successive components.

This algorithm can be exemplified by decomposing a house. In order for the house to be secure, it is important that the door and the window are functioning; therefore, the door and the window are the successive components of the house. The door however, is dependent of the key storage and the lock; therefore, the key storage and the lock are the two successive components of the door. See Figure 13.

![Figure 13: An example of Wang and Wulf's system decomposition](image)

**Functional relationships**

When the decomposition algorithm has terminated, it is possible to analyze the relationship between different nodes in the resulting tree structure. Wang and Wulf suggest a categorization of relationships and identify three categories:

- **Weakest link**
- **Weighted weakest link**
- **Prioritized siblings**

Weakest link relationships occur when a parent is ultimately bounded by the child node with the least security strength. Mathematically this is described as $S(\text{parent}) = \min(S(\text{child}_1), S(\text{child}_2), \ldots, S(\text{child}_n))$, where $S$ represents the security...
estimation, assessment score, of the nodes and n is the number of children nodes. Using the previous example of the door, the key storage, and the lock would result in \( S(\text{door}) = \min(S(\text{keystorage}), S(\text{lock})) = \min(0.75, 0.83) = 0.75 \)

The weighted weakest link is similar to the weakest link, but differentiates between trivial and important factors. Each child node is then provided with a weight percentile. The exact algorithm to calculate the weighted weakest link is unfortunately too extensive to be further reviewed in this brief summary. It is, however, nor important to this abstract.

The last categorization is the prioritized siblings. It occurs when siblings that contributes to an independent aspect of the parents function. Each sibling is provided with a weight percentile and the assessment score is a weighted sum.

Wang and Wulf admit that these categories only cover a fraction of all relationships that may occur between nodes in the decomposed tree structure.

**Weighting and priorities**

As previously implicated, while decomposing, it is sometimes necessary to differentiate the relative importance or weights among components. Weights specify to what degree children nodes influence their parent.

Wang and Wulf suggest a weighting technique designed by Thomas Saaty (1980), called The Analytic Hierarchy Process. The technique is based on complicated pair-wise comparisons and will not be reviewed here. The result of the technique is a weight ranging from 0 to 1 for each node. The sum the weights of all nodes equals 1.

**Basic measurements**

As mentioned earlier it is virtually impossible to measure end-to-end security properties of IT systems. It is, however, possible to measure the security properties of its basic components.

It is common that security related component attributes are defined in terms of qualities. It is important to articulate a usable and clear definition for such attributes, so that they are unambiguous, Wang and Wulf reasons. They therefore suggest breaking attributes down into factors, which in turn are broken down into criteria, which are broken down into metrics, which may be measured.

Wang and Wulf emphasize that care must be taken in implementing the basic metrics. Regardless of whether they are mathematical equations, diagrams or
questionnaires, they must be defined in a clear and unambiguous form to minimize the possibility of misinterpretation.

Component sensitivity analysis

When observing the resulting tree after decomposition, Wang and Wulf note that increasing the security score of some components, would have a greater influence on the overall security score of the top node, than increasing the security in other parts of the tree. To what degree the security score of the top node is influenced by the security score of another component in the tree is determined by that component’s sensitivity index.

The sensitivity index of a particular component is defined as the derivate of the overall security score with respect to the security score of that component.

3.2.3 Validation

Wang and Wulf establish that it is important but difficult to certify that the measures are valid, that is that the mappings from an empirical domain to a numerical domain preserve the empirical relations. They suggest three areas suitable for further investigation:

- **Validation based on measurement theory**
  Ensure that the measurement definitions do not violate the basic axioms of measurement theory. For example, arithmetic operations should not be used with ordinal-scaled measures.

- **Validation using empirical relations**
  Use observed behaviors and relations to validate measures. Filter out false correlations and discover meaningful relationships.

- **Validation using formal experiments**
  Use scientific experiments to prove or disprove hypotheses, even though extremely time-consuming, difficult and costly to operate.

3.2.4 Discussion

In this section, Wang and Wulf’s framework for security measurement, is discussed and analyzed in the context of this thesis.
Summarizing the paper, it consists of a system modeling technique, an evaluation technique, and some examples of algorithms and mathematical operations relevant to the system modeling and evaluation techniques.

Scientific approach
The negative examiner of Wang and Wulf’s work would say that it does not solve the original problem; “How does one measure IT security?” Instead it has created numerous new, smaller problems; “What scale-type and unit are appropriate when measuring the cryptographic algorithm of a software component?” or “How can I be sure that my estimations, no matter how elaborate, are objective, in the sense that someone else would have made the exact same estimations?”

This is of course somewhat true, in the sense that there is still much work to do. However, Wang and Wulf have introduced a scientific language that is very uncommon to this field of research. They have also applied mathematic concepts and algorithms from other fields that are relevant to the task. This should be regarded as a huge step forward.

One could of course argue that as long as the fundamental measurements themselves are as inaccurate and subjective as they are, sophisticated mathematics will not make any difference. To some degree, however, the collection of system data before modeling the system is undoubtedly aided by these algorithms (for example the pair-wise comparison algorithm when determining relations).

System modeling technique
The decomposition of complex parts into smaller more measurable parts may sound ingenious at first, but when examining the decomposition algorithm closer, it presents three major problems:

- **The tree will grow enormously fast for each new decomposition level**
  In reality, at least a hundred smaller factors correlate to the security function of a component. Evaluating all components and determining their relationships would be enormously time consuming.

- **It will finally be a subjective decision when to terminate the algorithm**
  The possibility to divide a component into yet other components will never end. Since the decision to terminate the algorithm is subjective, there will be certain resulting trees better describing the system than other does.
There is no way of knowing if all subcomponents are found
It might very well exist more correlating components than discovered when finished decomposing a component.

The functional relationships remind of those described in Anderson’s Security Evaluation Framework. The Wang and Wulf versions are more intricate, which certainly gives the impression that they have a potential for increased accuracy. They are, however, apparently incomplete, in the sense that they cover only a fraction of all possible relationships between components. In addition, the framework does not allow different siblings to relate to a parent node in deviating ways.

Evaluation technique
Wang and Wulf present a way to define security and then an algorithm to break it down into measurable parts. Again, they have a very sophisticated and scientific approach when discussing the different decisions associated with such a procedure, which without doubt provides the theories certain poise.

Breaking down the definition of security into smaller measurable parts reminds a lot of Andersson mapping Common Criteria’s security functions onto the CIA, and again proves that evaluating system components this way may very well be possible, even though it implicates, as mentioned in the discussion following the portrayal of Andersson’s thesis, quite some work.

Introducing the sensitivity index as a way of ranking components that need to be improved is a fresh idea that, even if not in its original form directly applicable on the subject of this thesis, is definitely a notion worth recognizing.

The validation strategies described in Wang and Wulf’s work are well founded in measurement theory.

There is, however, one major problem with the evaluation technique; not every metric in every security model might be compatible with every component in every tree resulting of the decomposition algorithm. This issue is nicely ignored by Wang and Wulf throughout their entire article. What happens if a metric is incompatible with a component is therefore a well kept secret.

Andersson solved a similar problem by creating a null value applicable in such a situation. Others simply do not calculate these values. Wang and Wulf seem to have tried a third and not so good solution; ignoring the issue.
3.3 Anna Stjerneby’s system component categorization

In her work “Identification of security relevant characteristics in distributed information systems” Anna Stjerneby (2002) has created a categorization of components, which will be a source of inspiration for the necessary component classification in the new method presented later in this thesis.

Stjerneby has created a categorization, containing 12 basic component classes, to which all existing components can be referred. These components and Stjerneby’s descriptions of them are:

- **User terminal**
  Represents anything a human would use to communicate with the system. This includes computers, PDAs, mobile phones, and digital cameras.

- **Server**
  Represents a computer that handles requests from other computers.

- **Application engine**
  Represents a component that runs the application software. If an application engine is combined with a user terminal, the result is a PC, and if it is combined with a server, the result is a mainframe.

- **Firewall**
  Represents anything that functions as a firewall, both software and hardware.

- **Network link**
  Represents all physical connections between components.

- **Access point**
  Represents all nonphysical connections between components, including infrared ports.

- **Public network**
  Represents a network that does not lie in the domain of control of the modeling party and cannot be influenced.

- **Router**
  Represents an intersection connecting local networks. Either statistically programmed with paths to different destinations or provided with a routing table, built up according to paths to different components in the network.
- **Switch**
  Represents an intersection that connects and forwards messages to the right components by examining the messages’ attached address.

- **Proxy**
  Represents a component that mediates traffic between network segments.

- **Modem**
  Represents anything that functions as a modem.

- **Input/output devices**
  Represents all input/output devices that is not a user terminal, network link or storage media, is included in this component.

### 3.3.1 Discussion

Stjerneby’s categorization seems to be complete in the sense that all exiting system components of a distributed system can be referred to one of the classes presented in the categorization.

There is however, one reason for it not being directly applicable on the new method, presented in Chapter 4; both logical and physical components are mixed in the same categorization.
A proposed method for evaluating security of component-based distributed information systems
This chapter presents a new way of evaluating the security level of distributed information systems. The new method is titled CAESAR (a Component-based Approach to Estimating the level of IT Security of Architecturally Rendered distributed information systems). The purpose of CAESAR is to estimate the security level of an entire distributed information system, based on the security level of, and the relations between, its included components.

CAESAR consists of two main tools:

- **A modeling technique**
- **An evaluation algorithm**

Figure 14 illustrates how the modeling technique and the evaluation algorithm together produce the overall security level of the distributed information system. The user of CAESAR first creates a model of the distributed information system, using the modeling technique (see section 4.2, page 39), data of the real system, and previously made security evaluations of components. The model is then supplied to computer software that implements the evaluation algorithm (see section 4.3, page 45), and based on the modeled system, calculates the overall security level of the system.
4.1 Scope

In section 2.1.3, page 12 and section 2.2.2, page 16 several dimensions by which a distributed system or a security aspect may be analyzed were recognized and discussed. In this section, the new method, CAESAR, will be identified on these dimensions.

**Physical – logical dimension**

CAESAR takes into account both physical and logical, or communicational, aspects, of the system to be modeled. Regarding physical aspects, CAESAR respects components’ geo-physical locations and the implications of them. For example, the evaluation algorithm takes into account that an unsecured component physically connected to another component, may pose a threat to the connected component.

Regarding logical aspects, CAESAR respects components’ communicational premises and patterns and the implications of them. For example, the evaluation algorithm takes into account that a server component, essential for many other components, may be more important than a client component, which no other components depend upon.

**Hardware – software dimension**

CAESAR regards hardware aspects of the system to be modeled, in the sense that it supports hardware classifications of components. To what degree CAESAR considers software aspects depend on which component evaluation method that is used (see section 4.2.3, page 42).

**Human – machine dimension**

CAESAR is machine-based and does not regard humans or their influence on the system in any way.

**Micro – macro dimension**

The micro – macro dimension is illustrated in Figure 5, page 13. The atoms of the modeling technique exist on level B, and are therefore computers or network components. Items on level C are considered to influence the component evaluation that results in a security level estimate. Component evaluation is beyond the scope of this thesis. The concept of security level estimates is further elaborated in section 4.2.3, page 42.

Regarding the evaluation algorithm, all modeled properties exists on level B, which means that all input to the evaluation algorithm is from level B. All
partial results exist on level B, but the final result exists on level A, which means that the output from the evaluation algorithm is on level A. This is the utter purpose of CAESAR; to gather system related information including pre-evaluated estimates on network component level, and deliver a security evaluation of the system on a network level.

The security estimations delivered as an output from the evaluation algorithm, are on the same mathematical form as the security estimations used as input. Even though currently not fully implemented, this may enable CAESAR to be applied iteratively on larger and larger systems in the future.

It should be mentioned that some information used as input to the modeling technique, might be considered gathered on level C in Figure 5 – especially information belonging to the modeling of logical relations.

### 4.2 Modeling technique

The main purpose of the modeling technique is to capture characteristics of the distributed information system that are important to its overall security level.

#### 4.2.1 Overview

The modeling technique consists of several building blocks. A brief overview of these blocks and how they relate is presented in Figure 15. A modeled system consists of other modeled systems, system components, and component relations.

Component relations are physical relations, and logical relations. System components are traffic generators (computers and public networks) and traffic mediators (firewalls, routers, proxies and hubs). Properties in gray are virtual properties that are not allowed to be used in an actual model.
A proposed method for evaluating security of component-based distributed information systems

Creating a model using the modeling technique of CAESAR consists of the following steps:

- **Identify all components**
  Identify all computers, firewalls, routers, proxies and hubs, as described in section 4.2.2, page 41.

- **Determine each component’s class**
  Determine whether a component is a traffic generator or mediator, as described in section 4.2.2, page 41.

- **Determine each component’s security level estimate**
  Determine the security level of the component, as described in section 4.2.3, page 42.
• **Determine each traffic mediator’s traffic control estimate**
  Determine the traffic mediator’s ability to filter malicious traffic, as described in section 4.2.4, page 43.

• **Determine physical relations between components**
  Determine how all components relate physically, as described in section 4.2.5, page 44.

• **Determine logical relations between components**
  Determine how all components relate logically, as described in section 4.2.6, page 44.

These steps and their descriptions contain many new concepts. The rest of section 4.2 will describe these concepts in detail, and present a way to represent the model graphically.

Note that a modeled system may be constructed from other modeled systems. Even though currently not fully implemented, this may enable CAESAR to be applied iteratively on larger and larger systems in the future.

### 4.2.2 Component classes

In order to create a model of a distributed information system, it is essential to define the smallest parts or components of which the modeled system is built – the *atoms* of the modeling technique.

It is important to decide on a finite number of such atoms, here called *component classes*. The number of component classes should be large enough for the resulting model to give a sufficiently detailed image of the system, but small enough to be unambiguous and comprehensible for the human user of the modeling technique.

It is crucial to understand that the number of component classes will determine the complexity of the evaluation algorithm that is to be applied to the modeled system later. An increased number of component classes will result in an increased complexity of the evaluation algorithm and vice versa.

Anna Stjerneby’s (2001) component categorization, presented in section 3.3, page 34 was an inspiration and a foundation on which the component classes for CAESAR was built. The following physical component classes exist:

- **Computer**
- **Public network**
Firewall

Router

Proxy

Hub

Some would argue that a class called Switch has been left out. In this thesis, a switch is considered equal to a hub from a security standpoint. If the user of CAESAR would be of a different opinion, it would be straightforward to add a component class.

These physical component classes are grouped into the following super classes:

Traffic generators
Computers, public networks.

Traffic mediators
Firewalls, routers, proxies, and hubs.

It is apparent that traffic generators separate from traffic mediators by their ability to generate traffic. In a simplified view, traffic generators could also generally be considered as security-decreasing components, while traffic mediators generally are security-increasing components.

The functional difference between the traffic generators and traffic mediators as defined in this method will be explained in the following sections.

4.2.3 Security level estimate

Each component, corresponding to a component class, is also designated a security level estimate (SLE). Security level is theoretically defined in section 2.2.3, page 17. The security level estimate may be calculated in a number of ways (see section 3.1.1, page 22 and section 3.2.2, page 28). Since it is beyond the scope of this thesis, it will not be further argued that – or how – such estimation may be produced, but merely established that it is possible (as discussed in section 3.2.4 under “Evaluation technique”, page 33).

In this chapter, the security level estimate is assumed to be on the form of a scalar value, which simplifies the description of CAESAR greatly. It would however, be equally possible to let the security level estimate, and therefore almost all other modeled and calculated properties, to be on the form of a
vector. This vector might for example contain the three elements confidentiality, integrity, and availability.

The security level estimate is regarded as a ratio scale, regardless of what it is chosen to measure.

4.2.4 Traffic control estimate

Each traffic mediator is assigned a traffic control estimate (TCE) from 0 to 1, based on its ideal ability to conceal malicious network traffic. 0 indicates that the component has no flow control whatsoever, and 1 suggests that no malicious network traffic at all may pass the component.

One would of course like to think of the traffic control estimate as an ratio scale, and it is defined as such in this thesis, but to be truthful the measurement possibilities merely allows it to be an ordinal scale. The values are however treated as of ratio scale type when aggregated in the evaluation algorithm.

It is imperative to recognize that the traffic control estimate is not an estimate of the actual function of a traffic mediator but rather of its purpose or ideal function.

As a suggestion, the following values may be used as a guideline:

- Firewall: 0.9
- Router: 0.2
- Proxy: 0.1
- Hub: 0.0

These values have been chosen arbitrarily. As described previously, it would be nice to think of the scale as ratio, meaning the firewall is 4.5 times better at concealing malicious traffic than a router and is concealing 90% of all traffic. From an ordinal perspective, however, it would be enough to observe that the firewall is concealing malicious traffic to a greater extent than a router.

When deciding on a suitable traffic control estimate to a predefined or invented traffic mediator one should consider its ability to conceal malicious traffic relate to other traffic mediators.
4.2.5 Physical relations

Physical relations connect two components with each other. Physical relations contain no modeled properties, except references to the two connected components. Physical relations are considered symmetrical.

4.2.6 Logical relations

Logical relations connect two traffic generators with each other. The purpose of introducing logical relations into the modeling technique is to be able to regard components' communicational premises and patterns and the implications of them. For example, with logical relations the evaluation algorithm takes into account that a server component, essential for many other components, may be more important than a client component, which no other components depend upon.

Logical relations are non-symmetrical and contain a server-end and a client-end. Peer-to-peer relations are modeled as two logical relations; one in each direction.

Logical significance

Except the references to the two connected traffic generators and its direction, logical relations also contain a property that denotes the significance of the relation. Logical significance, ranging from 0 to $M$ is classified according to the following table:

<table>
<thead>
<tr>
<th>Logical significance</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insignificant</td>
<td>0</td>
</tr>
<tr>
<td>Little significance</td>
<td>$(1/3)M$</td>
</tr>
<tr>
<td>Average significance</td>
<td>$(2/3)M$</td>
</tr>
<tr>
<td>Great significance</td>
<td>$M$</td>
</tr>
</tbody>
</table>

The upper bound $M$ may be any value. If a logical significance of 5 is chosen for a component, its security level estimate will weigh five times that of a directly physically related component (six times, if the component is also both physically and logically related). When modeling large systems, it might therefore be appropriate to increase the upper bound of $M$. 
4.2.7 Graphical representation

When working with the modeling technique either manually or computer-aided, it may be helpful to be able to see a graphical representation of the model. Such a graphical representation is presented in Figure 16.

![Figure 16: An example of a graphical representation of a CAESAR model](image)

4.3 Evaluation algorithm

The purpose of the evaluation algorithm is to analyze the modeled system and determine an overall security level of the distributed information system.

4.3.1 Overview

The main goal of the algorithm is to calculate the overall security level (OSL) of the system, but many partial results are interesting in themselves and may be analyzed to understand how different factors affect the security level of different parts of the system.

To be able to calculate the overall security level of the system, it is necessary to perform an evaluation of each component of traffic generator class. Such an evaluation results in a system-dependent security level (SSL) for that component.

Figure 17 gives a general overview of which modeled and calculated properties that are necessary to perform an evaluation of a system.
Figure 17: The CAESAR evaluation algorithm’s main concepts and their relations

The system-dependent security level of each traffic generator, from here on referred to as the component of evaluation, is calculated by aggregating the neighboring security contributions of all physically related traffic generators and the logical security contributions of all logically related traffic generators acting as servers to the component of evaluation.

The neighboring security contribution of a physically related traffic generator is calculated by aggregating the security level estimate of that traffic generator, with all traffic mediators that are on the physical path or paths between the traffic generator and the component of evaluation taken into account by aggregating their comparable security level.

The logical security contribution of a logically related traffic generator acting as a server to the component of evaluation is calculated by aggregating the security level estimate of the traffic generator and the logical significance of the logical relation that connects the traffic generator with the component of evaluation.

The overall security level of the distributed system is then calculated by aggregating the system-dependent security level of all traffic generators.

The algorithm may also be described on a shorter form, as expressed below:
For each traffic generator, \( C_e \), in system

for each traffic generator, \( C_g \), physically related to \( C_e \)
calculate the aggregated CSL between \( C_g \) and \( C_e \).
calculate the NSC from \( C_g \) to \( C_e \) using CSL.

for each traffic generator \( C_g \), logically related to \( C_e \)
calculate the LSC from \( C_g \) to \( C_e \).
calculate the SSL of \( C_e \) using all CSL, all LSC, and the SLE.

Calculate the security level of the system.

Exactly how these modeled and calculated properties are aggregated are described in detail in the rest of section 4.3.

The aggregation formulas used in the CAESAR method, as presented in this thesis, may or may not be the optimal formulas for aggregating properties, considering the overall reliability of the method.

### 4.3.2 Overall security level

The overall security level (OSL) of the system is the final result of CAESAR. In order to calculate the overall security level of the system it is necessary to know the system-dependent security level of each traffic generator in the system. Figure 18 illustrates this graphically.

The overall security level is merely an average of all traffic generators’ system-dependent security level. The general formula for a system with traffic generators, \( C_1, C_2, \ldots, C_n \) is:
A proposed method for evaluating security of component-based distributed information systems

\[
OSL(\text{system}) = \frac{\text{SSL}(C_1) + \text{SSL}(C_2) + \ldots + \text{SSL}(C_n)}{n}
\]

References

How the system-dependent security level (SSL) of each component is calculated is explained in section 4.3.3, page 48.

4.3.3 System-dependent security level

In order to evaluate a component, further known as the component of evaluation, with respect to its environment – that is, calculating its system-dependent security level (SSL) – it is necessary to calculate the neighboring security contributions (NSC) of all other traffic-generating components that are physically related to the component of evaluation directly, or indirectly through a traffic-mediating component. It is also necessary to calculate the logical security contributions (LSC) of all traffic-generating components that are logically related to the component of evaluation. Finally, it is necessary to know the security level estimate (SLE) of the component of evaluation. Figure 19, which is a subset of Figure 17, illustrates this graphically.

![Figure 19: Factors that influence system-dependent security level](image)

When establishing which components that are allowed to give a neighboring security contribution to the component of evaluation, the following rule is used: All traffic generators that are directly or indirectly connected through one or more physical relation(s) and possibly one or more traffic mediator(s) should give a neighboring security contribution to the component of evaluation.

It is important to understand that a traffic generator connected in series with a traffic generator connected in series with the component of evaluation does not
give a neighboring security contribution to the component of evaluation. Traffic generators do not mediate traffic.

When determining which components that are allowed to give a logical security contribution to the component of evaluation, the rule is simple: All traffic generators that are logically related as servers should give a logical security contribution to the component of evaluation. Clients do not give any logical security contribution since the component of evaluation does not depend on them. However, they might still give a neighboring security contribution if directly physically related to the component of evaluation.

Figure 20 shows three examples of when neighboring security contribution (NSC) and logical security contribution (LSC) is used.

![Diagrams showing NSC and LSC](image)

Figure 20: Examples showing when to use NSC and LSC

The general formula for calculating the system-dependent security level of a component \( C_e \) with respect to neighboring security contributions from \( C_{f1}, C_{f2}, \ldots, C_{fn} \) as well as logical security contributions from \( C_{l1}, C_{l2}, \ldots, C_{lm} \), whose logical significance are \( S_1, S_2, \ldots, S_m \) respectively:

\[
\begin{align*}
a &= SLE(C_e) + \text{NSC}(C_{f1}) + \text{NSC}(C_{f2}) + \ldots + \text{NSC}(C_{fn}) + \\
&+ \text{LSC}(C_{l1}) + \text{LSC}(C_{l2}) + \ldots + \text{LSC}(C_{ln}) \\
b &= 1 + n + (S_1 + S_2 + \ldots + S_m) \\
\text{SSL}(C_e) &= \frac{a}{b}
\end{align*}
\]

The following examples will explain how to apply the general formula to two specific cases.

**Example 1:** Consider the simplest system; just two components, \( C_e \) and \( C_f \), both traffic generators (see section 4.2.2, page 41), connected with a physical relation.
In order to calculate the system-dependent security level of \( C_i \), we need to consider only the neighboring security contribution of \( C_i \):

\[
SSL(C_i) = \frac{SLE(C_i) + NSC(C_i)}{1 + 1}
\]

**Example 2:** Now instead, consider the scenario where the component of evaluation, \( C_i \), is connected with one other traffic generator, \( C_1 \), through a physical relation, and to two other traffic generators \( C_2 \) and \( C_3 \), whose logical significances are 3 and 4, through logical relations. In that case the system-dependent security level of \( C_i \) would be:

\[
SSL(C_i) = \frac{SLE(C_i) + NSC(C_1) + LSC(C_2) + LSC(C_3)}{1 + 1 + 3 + 4}
\]

In some cases, there may even more components, whose security contributions must be taken into account.

**References**

How neighboring security contributions (NSC) are calculated is explained in section 4.3.4, page 50. How logical security contributions (LSC) are calculated is described in section 4.3.5, page 53. The concept of logical significance is described in section 4.2.6, page 44. The concept of security level estimates (SLE) is detailed in section 4.2.3, page 42.

**4.3.4 Neighboring security contribution**

The term *neighboring security contribution* (NSC) is introduced to take into account the security level of physically related components, and especially to regard the possible threat a component with a low security level estimate may pose to the component of evaluation.

Figure 21, which is a subset of Figure 17, explains which system properties that are used when calculating the neighboring security contribution of a component.
As previously mentioned, when establishing which components that are allowed to give a neighboring security contribution to the component of evaluation, the following rule is used: All traffic generators that are directly or indirectly connected through one or more physical relation(s) and possibly one or more traffic mediator(s) should give a neighboring security contribution to the component of evaluation.

Consider the simplest system; just two components, \( C_e \) and \( C_g \), both traffic generators (see 4.2.2, page 41), connected with a physical relation. In order to calculate the neighboring security contribution (NSC) of traffic generator \( C_g \) to component of evaluation \( C_e \), the following expression is used:

\[
NSC(C_g) = SLE(C_g)
\]

In this simple case, the neighboring security contribution of \( C_g \) merely equals its security level estimate (SLE).

If there is a traffic mediator (see 4.2.2, page 41), \( C_m \), on the path between \( C_g \) and \( C_e \), this must be taken into account, since the traffic mediator might filter out malicious traffic and therefore increase the neighboring security contribution of \( C_g \).

If there is precisely one path, through which a message may be transmitted from \( C_g \) to \( C_e \) and that path contains precisely one traffic mediator, \( C_m \), the following expression gives the neighboring security contribution of \( C_g \) to \( C_e \):

\[
NSC(C_g) = \text{Max}(CSL(C_m), SLE(C_g))
\]
CSL stands for comparable security level, and is a measurement of how well a traffic mediator filters malicious traffic.

If there is more than one traffic mediator or several paths with traffic mediators between the traffic generator and the component of evaluation it is necessary to aggregate the traffic generators’ comparable security level into a single comparable security level, as described in section 4.3.6, page 55 and 56. This value may then be used as above.

The chosen expression for calculating neighboring security contributions may seem blunt at first, but delivers plausible results. Consider an example with a computer, A, connected to a firewall, F, connected to a computer B. Let us consider the neighboring security contribution of A to B in the following extremes:

- **A is 100% secure and F is 0% working**
  Since the security level estimate of computer A is 1, the comparable security level of firewall F is irrelevant and the resulting neighboring security contribution is 1.

- **A is 0% secure and F is 100% working**
  Since the comparable security level of firewall F is 1, the security level of computer A is irrelevant and the resulting neighboring security contribution is 1.

- **A is 50% secure and F is 0% working**
  Since the security level estimate of computer A is 0.5, and the comparable security level of firewall F is 0, the resulting neighboring security contribution is 0.5.

One could of course invent more intricate examples, as “A is 75% secure and F is 75% working”. Should the result of this be the exact same as when “A is 75% secure and F is 0% working”? No, common sense indicates that there should be some differences in result, between the case with an almost perfectly well working firewall and the one were it is equivalent to non-existing. The small risk A poses to B in the first case might be caught by the firewall. In the second case however, such risk will never be caught by the firewall.

There seems to be no simple algebraic expression to embrace this problem. However, if implementing the evaluation algorithm as computer software, there need not be. Figure 22 gives an example of how it would be possible to relate the security level estimate (SLE) and the comparable security level (CSL) to the resulting neighboring security contribution (NSC) using an arbitrary function.
This way it is possible for the user of the algorithm to take into account as many examples as necessary, estimate the result of the examples and then interpolate the possible cases in between. If that approach is considered unsuitable, the maximum-function described previously delivers reasonable results.

References

How to calculate the comparable security level (CSL) of a traffic mediator is explained in section 4.3.5, page 53. The concept of security level estimate (SLE) is described further in section 4.2.3, page 42.

4.3.5 Logical security contribution

The term logical security contribution (LSC) is introduced to increase the sensitivity of the overall security level to the security level of traffic generators that are acting as servers to other components in the system.

One could imagine that the security level of a computer that delivers essential data to the component of evaluation is more important than a terminal is. The logical security contribution is constructed to reflect such differences in importance between different components.

The logical security contribution of a component is determined by its security level estimate, its logical relation to the component of evaluation, and the logical significance of that relation. Figure 23, which is a subset of Figure 17, illustrates this graphically.
As previously mentioned, when determining which components that should be allowed to give a logical security contribution to the component of evaluation, the rule is simple: All traffic generators that are logically related as servers should give a logical security contribution to the component of evaluation.

The logical security contribution (LSC) of a traffic generator, \( C_g \), with a logical relation (LR) of a certain logical significance (LS), to the component of evaluation, \( C_e \), is expressed as follows:

\[
LSC(C_g) = LS(LR(C_g, C_e) \cdot SLE(C_g))
\]

References
The concept of logical significance (LS) and logical relations (LR) is described in section 4.2.6, page 44. The concept of security level estimates (SLE) is detailed in section 4.2.3, page 42.

4.3.6 Comparable security level

When calculating the neighboring security contribution (NSC) of a specific component (see 4.3.5, page 53), it is necessary to consider all physical paths through which a message may be sent across the network from \( C_n \) to \( C_e \). On each of these paths, all traffic mediators must be identified.

Figure 24, which is a subset of Figure 17, explains which system properties that are used when calculating the comparable security level of a traffic mediator.
In order to calculate the neighboring security contribution of \( C_n \) to \( C_e \), it is necessary to consider all traffic mediators comparable security level (CSL) which may affect a transmitted message. The comparable security level of a traffic mediator, \( C_m \), is calculated by multiplying the component’s traffic control estimate with its security level estimate:

\[
CSL(C_m) = TCE(C_m) \cdot SLE(C_m)
\]

### Aggregating serial traffic mediators

If there is one path, through which a message may be transmitted from \( C_n \) to \( C_e \), but two traffic mediators, \( C_1 \) and \( C_2 \), are in series along that path, the following expression gives the aggregated comparable security level \( CSL(C_a) \) for the traffic mediators, \( C_1 \) and \( C_2 \):

\[
CSL(C_a) = \text{Max}(CSL(C_1), CSL(C_2))
\]

This algebraic expression for aggregating traffic mediators in series assume that traffic mediators with a lower comparable security level are functional subsets of traffic mediators with a higher comparable security level.

Considering the classes Firewall, Router, Proxy and Hub, that might be the case if these are provided with a security level estimate near 1. If they have a security level estimate much lower than 1, it might, however, not be an appropriate approximation.

If the traffic mediators \( C_1 \) and \( C_2 \) mentioned above instead would be considered entirely complementary, the following expression would describe the comparable security level of those traffic mediators:
A proposed method for evaluating security of component-based distributed information systems

\[ CSL(C_a) = CSL(C_1) + (1 - CSL(C_1)) \cdot CSL(C_2) \]

These two cases are extreme ones that will virtually never appear in a realistically modeled distributed system. They might however still be useful as simplifications.

If the degree of overlapping functionality between \( C_1 \) and \( C_2 \) would be known as \( f \), ranging from 0 to 1, it would be possible to describe the comparable security level as below:

\[ CSL(C_a) = f \cdot \text{Min}(CSL(C_1), CSL(C_2)) + (1 - f) \cdot CSL(C_1) + (1 - CSL(C_1)) \cdot CSL(C_2) \]

Since the amount of overlapping functionality between components is rarely known (there need to be an \( f \) for each possible pair of traffic mediators in the network) and therefore unsuitable to require when modeling the system, it is suggested that serial traffic mediators are aggregated as first shown – with a simple maximum-function.

If there are more than two traffic mediators in series along a path, the maximum-function may easily be enlarged to contain more than two traffic mediators. If using one of the other expressions to aggregate more than two traffic mediators the formula may instead be applied iteratively (which would then require even more known \( f \)s, if that approach were to be used).

**Aggregating parallel traffic mediators**

If there are precisely two paths, through which a message may be transmitted from \( C_n \) to \( C_e \) and each path contains one traffic mediator only, totaling \( C_1 \) and \( C_2 \), the following expression would give the aggregated comparable security level \( CSL(C_a) \) for the traffic mediators, \( C_1, C_2 \):

\[ CSL(C_a) = \text{Min}(CSL(C_1), CSL(C_2)) \]

This approach assumes that a network is not stronger than its weakest link, and as long as the weaker component does not increase its security level estimate or is replaced with a component with a higher traffic control estimate, it does not matter how much the comparable security level of the stronger component is increased.

This is of course an extremely simplified argument, and an equally simplified argument would be that the resulting aggregated comparable security level for the two paths would be an average:
The CAESAR method

\[ CSL(C_a) = \frac{CSL(C_1) + CSL(C_2)}{2} \]

None of these extremes seem sufficient. Instead, a blend between the weakest link and average method would be appropriate. It may be achieved by introducing a network-wide parameter \( a \), ranging from 0 to 1 that determines the degree to which a weakest link perspective should be applied. The resulting expression would then be:

\[ CSL(C_a) = a \cdot \text{Max}(CSL(C_1), CSL(C_2)) + (1-a) \cdot \frac{CSL(C_1)+CSL(C_2)}{2} \]

The suggested way of aggregating parallel traffic mediators are therefore the above method, using a network-wide parameter \( a \), that determines the degree to which a weakest link perspective should be applied. An appropriate value of \( a \) would be somewhere around 0.5, but it is finally up to the user of the evaluation algorithm to determine a suitable value.

If there are more than two paths, the equations may easily be enlarged to contain more than two mediators at a time, or the equations may be applied iteratively on pairs of paths.

If some paths contain more than one traffic mediator in series, these must first be aggregated as described earlier, so that the path contains one aggregated traffic mediator only.

**References**

The security level estimate (SLE) is described further in section 4.2.3, page 42. The traffic control estimate (TCE) is detailed in section 4.2.4, page 43.

### 4.4 Discussion

Even though CAESAR is a fully functional and relevant method for evaluating security of component-based distributed information systems, it should be considered more of an idea of a way to evaluate systems rather than a fixed and finished method. In this section advantages and drawbacks of CAESAR are discussed, along with possible alternative design choices and their implications.

Since the beginning of the construction of CAESAR, it was clear that there was a delicate balance between a complicated but accurate method and comprehensible but less accurate method. There would certainly be no use of
CAESAR; no matter how sophisticated, if no one would be able to understand how to apply it.

Without doubt, the objective through the entire work of this thesis has been to make it as understandable and comprehensible as possible. Unfortunately, the method itself needs to attain a certain complexity in order to be able to predict something other than what would be obvious to any observer of the system.

Ultimately, it will be up to the user of this method to determine whether CAESAR incorporates the crucial balance between complexity and comprehensibility.

4.4.1 Advantages

One of the main advantages of CAESAR is that it is possible to apply to both designed, but not implemented systems, and fully implemented ones. This is because none of the estimation or measurement processes requires empirical data, such as statistical data derived from system logs. It could be an enormously important tool when laying out the overall structure of a new distributed information system, but also an asset in the everyday monitoring of system modifications and improvements.

The ability of CAESAR to regard traffic flow control should be considered one of its greatest assets. In the beginning of the construction of CAESAR, it was uncertain whether it should be possible to implement, and no other previous example of such a method could be found. It seems to be a unique quality of CAESAR.

Another advantage of CAESAR is that it is possible to evaluate one or several subsets of a system individually, which then become pre-evaluated components themselves. This feature implicates several rewards. One of them is the possibility to hide subnets or parts of a system, and receive a simplified and more graspable overview of an existing system. Another significant reward is that evaluated systems may be packaged as building blocks for use when trying out new designs or modifications.

Yet another advantage of CAESAR is that it withstands basic correctness tests. In all simple scenarios, the partial and final aggregations seem to correspond to the desirable outcome predicted by other security authorities. If the method holds for these simple examples, it is plausible that the partial and final aggregations are truthful in scenarios that are more complex and unpredictable.
Another way of assuring correctness of CAESAR is to aggregate properties in a different order, and ensure that the final result remains the same, independent of order. CAESAR withstands this correctness test as well.

The simplicity of CAESAR should also be regarded as one of its main strengths. It is both easy to use, and to implement if one would like to wrap the evaluation algorithm in computer software.

Finally, one of the major advantages of CAESAR is its modular structure, due to which it is very easy to enhance a part of the modeling technique or evaluation algorithm without affecting other parts.

4.4.2 Drawbacks

A very significant drawback of CAESAR is that it is impossible to compare the overall security level of systems with different amount of components. Consider the example where a computer, A is connected to computer B and C. A has a security level estimate of 0.5 and B and C a security level estimate of 0. One may easily recognize that the contributions of B and C will lower the system-dependent security level of A significantly.

If another computer, D, with a security level estimate of 1 is connected, the system-dependent security level of A would increase somewhat. This however, cannot be recognized as a sound quality of the evaluation algorithm. Before D was connected to A, there were no threats from D influencing A, and after D is connected, there are just as many, since it is completely secure. Therefore, the system-dependent security level should not be increased, but left unaffected.

This flaw of CAESAR may be circumvented by saying that only overall security levels of systems with the same amount of components may be compared. Otherwise, calculated properties may not be compared. Another more complicated way to dispose of this defect is to adopt a completely different way of approaching the entire problem of evaluating security in systems; by considering lack of security (risk) rather than level of security. It would then be possible to model the flows of risk through the system. This will be further elaborated in section 4.4.3, page 59.

4.4.3 Design choices

In this portrayal of CAESAR, the security level estimate has been referred to as a scalar. This may not necessarily be the only way to represent security levels in CAESAR, but has been chosen because of its pedagogic qualities. One could
easily imagine the security level estimate being a vector, for example consisting of the elements confidentiality, integrity and availability. Andersson (2003) describes many possible representations in his work (see section 3.1, page 21). In fact, most other modeled and calculated properties of CAESAR may very well be vectors. This has, however, not been further examined, since it lies somewhat outside the scope of this thesis, but it would be a natural step to examine this if one were to improve CAESAR.

During the early stages of the development of CAESAR, a simple symmetrical evaluation approach, similar to the one described in Andersson's work (see section 3.1, page 21), where small components are aggregated into larger ones, beginning with centrally located components and slowly stepping out to the periphery, was adopted. This however resulted in many undesired implications. It was hard to include any sorts of non-symmetrical directions (such as the logical relations) and the geo-physical location of components had a greater impact on overall system security than deserved.

Another approach that was temporarily examined was one were the network was converted into a tree structure with each node representing a component of the system. Then the algorithm worked its way down the tree aggregating security properties along the way, until it finally, when having traversed the entire tree, returned the overall system security. This approach soon became extremely complicated. It became necessary to introduce a number of exceptions such as how to handle loops of the network and it became difficult to regard logical relations.

Therefore, a third approach was adopted, in which the algorithm examined the network from each components perspective, and then aggregated all components perspectives into the overall security level of the system. This approach is used in CAESAR, as presented in this thesis.

As mentioned earlier it would be possible to model the flows of risk, rather than modeling the flows of security. The approach of modeling the flows of security brings many problems, such as the difficulties of measuring security and anomalies as the one described in section 4.4.2, page 59, where a computer that apparently makes no difference to the system security whatsoever, nevertheless influences the system-dependent security level of other components.

The design choice to model security rather than risk was made because this thesis was accepted on the premises of constructing a new method that aggregates existing security level estimates of components. If the method were to be further enhanced, introducing the possibility to model flows of risk would certainly be an interesting approach. This is further elaborated in section 7.2, page 71.
Choosing to model flows of risk rather than flows of security would result in less relativistic and more absolute results of the evaluation algorithm.

Another central design choice was whether traffic generators should be able to mediate traffic, and whether traffic mediators should be possible to generate traffic. In CAESAR, each of these component classes are considered a stereotype, and even though it is quite possible that a computer, correctly or incorrectly configured depending on perspective, with two network adapters installed, would be able to transmit traffic between two other network components. The way CAESAR is currently constructed, that would be impossible to model.

Changing this design choice is, however, very simple. Instead of regarding only physically connected traffic generators when calculating the system-dependent security level of a component, instead all directly and indirectly connected traffic generators should be able to contribute to the evaluated component’s system-dependent security level.

If the modeling technique was to be changed in this way, one could always emulate the current version of the modeling technique by providing each traffic generator with one or more traffic mediators with a traffic control estimate of 1 and a security level estimate of 1. This would block all traffic through that component and thus be equivalent to the current version of the modeling technique.
A proposed method for evaluating security of component-based distributed information systems
The software supporting the CAESAR method is called ROME, since the famous, Gajus Julius Caesar spent much of his time in Rome. This chapter gives an overview of the software and some examples of how it may be used.

5.1 Overview

The purpose of the ROME software is threefold:

- **To demonstrate the CAESAR method graphically**
  For those who may not have the time or interest to read about the CAESAR method, instead experimenting with ROME may give them a fundamental understanding of the method.

- **To evaluate existing or planned systems using the CAESAR method**
  Even tough in an early development, the CAESAR method is a working and usable method aimed at determining the level of IT security of vast dynamic component-based distributed information systems.

- **To evaluate and enhance the CAESAR method**
  ROME offers a quick, easy and intuitive way to evaluate the method itself. The algorithm may then be changed according to use cases and tried again. ROME therefore offers an excellent opportunity to evaluate and enhance the CAESAR method.

The ROME software, in its current version, offers the following functionality:

- **Modeling of systems using an intuitive point-and-click manner**
  ROME supports all classes and physical relations as described in this thesis. It is easy to create and save new systems as well as load, modify or examine existing ones.

- **Automatic real-time evaluation of systems**
  ROME supports all calculated properties as described in this thesis.
Connecting, deleting or modifying a component immediately affects other components and the graphical representation of the security levels.

- **Intuitive security level representation using shapes and colors**
  ROME displays evaluation results in highly configurable color or shape shifts, on either one of several component levels or on overall system level.

### 5.2 Layout and usage

Figure 25 shows a screenshot of what the main window of the ROME software looks like.

![Screenshot of the ROME software](image)

**Figure 25: Screenshot of the ROME software**

In the upper left corner, system information is displayed. Both modeled properties, such as the number of components, and calculated properties, such as the overall security level is displayed. In the lower left corner, the currently selected component is showed along with both modeled properties, such as its name, class and security level estimate, and calculated properties, such as its neighboring security contribution and system-dependent security level.
In the upper right corner, a pie chart along with a percentile is shown. The color of the pie chart changes depending on the current overall security level of the modeled system from blue when 100% secure to red when 0% secure.

The large window to the right is the workspace, where the modeled system is displayed. Each class has its own icon; computers are represented as boxes, and public networks as globes. Physical relations are shown as gray lines between components.

When first starting the program, it is possible to load an existing system, or create a new one. To create a new one, simply click with the right mouse button in the large space to the right, select Add > Standard Component > Computer as shown in Figure 26. It is possible to add supplementary components in the same manner. Physical relations are created by clicking on a component with the right mouse button, selecting Add > Relation > Physical relation, and then clicking with the left mouse button on another component.

![Figure 26: Adding components using the ROME software](image)

The texts and colors used to represent the system in the workspace are highly configurable. For example, to let the color of the text label of each component represent its system-dependent security level ranging from blue when 100% secure to red when 0% secure, click on the menu View > Node color > System-dependent Security Level (SSL) as shown in Figure 27.

![Figure 27: Adjusting representation using the ROME software](image)

Figure 28 shows what the workspace might look like using these settings to represent the system-dependent security level of each component.
5.3 Requirements and availability

The ROME software can be obtained from the following web site:

- http://www.mikaelpeterson.com/caesar

The ROME software runs on any computer with Windows 9x/200x/ME/XP and Microsoft .NET Framework 1.1 installed. Microsoft .NET Framework can be obtained from Microsoft’s web site.

5.4 Limitations

The ROME software is not intended to be considered an optimized, scalable, or fully functional product ready to use. It is merely a simple demonstration of how computer software may aid the use of the CAESAR method. It is built using C# and Microsoft’s .NET platform, which may, or may not, be the best platform on which to build a final version of the product.
6 Conclusions

The main target of this thesis was to design a method, capable of determining the level of IT security of vast dynamic component-based distributed information systems.

The main target was translated into three sub targets:

- Establish a theoretical foundation
- Review previous efforts
- Develop a new method

The sub targets were then further divided into seven work phases:

- Review general IT security literature and articles
- Review distributed system literature and articles
- Review measurement theory literature
- Review existing published methods aimed at evaluating IT security
- Merge ideas and concepts from previous phases into a new method
- Refine the new method
- Develop demonstration software of the new method

Each of these seven phases was carried out during the work of this thesis, and each of the three sub target were addressed.

During the initial phases of the work, it quickly became apparent that the research field of IT security lacked a common base of advanced theory. In addition, scientific approaches in general and measurement theory specifically was not widely used.
This insight was further proved during the fourth phase when the Internet and libraries were searched for relevant literature with little result. It was then obvious that most of the new method had to be invented from scratch.

During the fifth phase, several different approaches to both modeling techniques and evaluation algorithms were tried. The iterative refinement could obviously have continued forever, but was terminated after having achieved an adequate level of functionality.

After the seventh phase, when the demonstration software was finished and it was possible to test the evaluation algorithm on different scenarios more efficiently, it became evident that the current algorithm had several weaknesses; many of which could easily be eliminated in future work.

The main target has been accomplished; a new method, capable of determining the level of IT security of vast dynamic component-based distributed information systems, has been developed. In addition, computer software demonstrating the new method has been developed.
7 Future Work

This chapter outlines the areas suitable for future work if the CAESAR method or the ROME software were to be further enhanced.

7.1 Security measurement

An important area of the CAESAR method that may need improvement is the representation of measurements and aggregated measurements.

Measurement theory application

In this thesis, basic measurement theory applicable on IT security evaluation has been reviewed and related to the new CAESAR method briefly. If the CAESAR method were to be further developed, an important improvement would be to define all quantities, both measured and aggregated ones, with magnitudes and representations (see section 2.2.3, page 17).

This would of course presume an existing component evaluation method that is soundly grounded in measurement theory. Such a method, possible to integrate with the CAESAR method, does however currently not exist.

Without scientifically defined quantities the results of the CAESAR method, or any other similar method, will be extremely difficult to examine, compare or even interpret. This is of course affecting the reliability, which in turn affects the scientific credibility of the method.

To define all quantities with magnitudes and representations is possible, and would of course increase the scientific credibility of the method. In addition, it would probably disclose any inconsistencies within the method itself and clarify the method, making subsequent enhancements easier to implement.
Measurement representation

In this thesis all CAESAR-related quantities have been defined as scalars, which often hold a value ranging from 0 to 1 – a pedagogic simplification necessary for the description of the CAESAR method. If the CAESAR method were to be further developed, a natural step would be to enlarge these quantities to hold more than a single value.

Such vectors could hold three values, indicating perhaps the confidentiality, integrity or availability of a certain component or system. It could as easily represent the protection, detection and reaction capabilities of a component or system.

More importantly, such a vector could also hold more complex measurements, such as the 11-element vector described in Andersson’s Security Evaluation Framework (section 3.1.1, page 22) or as the complex metrics described in Wang and Wulf’s Security Measurement Framework (section 3.2.2, page 30).

If the advantage of the CAESAR method to predict the security level of modeled non-existent systems should be preserved, it is important not to measure statistical entities, such as the percentile of withstood intrusion attempts that may of course not be measured until the system is live and running. This is however a design choice.

7.2 The CAESAR modeling technique

The modeling technique determines which system properties that should be regarded and which that should be disregarded.

Modeling component publicity

A property that was left out in the CAESAR method as presented in this thesis is the publicity of each component. It is plausible that components physically accessible by the public require a higher security level to maintain a preserved impact on other components of the system, compared to a component that is physically inaccessible to anyone.

Exactly how such a property should be defined and aggregated with other properties is uncertain at this point, but its introduction into the CAESAR method would nevertheless improve the method significantly.
Modeling flows of risk

As extensively described in section 4.4.3, page 59, instead of modeling the security level of the system, it would in some cases perhaps be more efficient to model the security risk of the system, and in particularly how risk flows or spreads through the system.

Also, as previously discussed in section 4.4.3, page 59, such an approach would possibly result in less relativistic and more absolute properties of the evaluation algorithm, aiding the scientific target described in section 7.1, page 69.

It is of course possible that modeling only the risk level of components and the flows of risk, and disregarding the security level of components and systems would give more accurate results. However, it is also possible that modeling the risk level of components and the flows of risk, parallel to the current security level perspective would give the best results.

Modeling all components using a single class

As presented in this thesis, the modeling technique contains two super classes of components; traffic generators and traffic mediators. Traffic mediators generate traffic, but mediate no traffic and traffic mediators mediate traffic, but generate no traffic.

In a real system to be modeled, there would be components that fit none of these two super classes; components that both generate and mediate traffic, such as a computer with two network interfaces, connected through software. In fact, that might be the most common type of component in some networks.

It would therefore seem as a rational step when improving the CAESAR method, to collapse all component classes into a single class, which holds both a security level estimate (SLE) and a traffic control estimate (TCE) for each component. This would be an easy enhancement to implement.

A component previously classified as computer would then have a security level estimate as previously and a traffic control estimate of 1, since it admits no traffic to pass. All other components would be modeled as before.

All functionality in the current version of the CAESAR method would be preserved, but the new functionality of being able to model components that both generate and mediate traffic, would be added.

Modeling public networks

In the CAESAR method, as presented in this thesis, public networks are modeled as a single traffic generator. The effect of this is that the impact of a
public network, connected to a single computer, would be the exact same as that of another computer with a security level estimate of 0.

It would of course seem more reasonable that the weight of a public network, such as the Internet, was heavier than the weight of a single computer.

Modeling every computer of the public network is of course impossible, but perhaps it would be possible to assign a weight and a security level estimate of the public network, signifying the approximate number of contained traffic generators, and their average security level estimate.

**Modeling systems within systems**
Throughout the description of the CAESAR algorithm in this thesis, it has been suggested that a previously modeled system should be reusable when modeling a new system, meaning that is should not merely be expanded into components of the new system, but rather contained as a special component in the new system.

This would introduce the possibility to choose a survey view of the monitored network, were clusters of networks are collapsed into a single component, sacrificing details and accuracy to the benefit of a better overview of the system.

### 7.3 The CAESAR evaluation algorithm

The evaluation algorithm aggregates modeled properties to system specific properties.

**Enhance aggregation formulas**
The CAESAR method, as presented in this thesis, is provided with the necessary formulas to aggregate modeled properties to system specific properties that are finally aggregated to a single property describing the overall security level of the distributed information system.

These aggregation formulas, may be, but are probably not, the optimal way of aggregating properties within the framework of the CAESAR method. If the CAESAR method were to be further enhanced, this would probably be one of the most time consuming areas to work with, but also one of the most rewarding, considering the overall reliability of the method.
Sensitivity analysis

As presented in this thesis, the CAESAR algorithm generates one important value, the overall security level. It would however, often be relevant to investigate how the security level estimate or traffic control estimate of each component contribute to the overall security level, and in particular, how a change in a component property would reflect in the overall security level.

Since the evaluation algorithm adapted to computer usage, it would be easy to add the possibility to investigate the derivative of the overall security level with respect to a certain property.

If the sensitivity analysis would suggest that, a small, and therefore implicitly cheap, increase of security level or traffic control of a component would result in a large increase of the overall security level, that component would then be suitable for further examination by the system designers.

In addition, it would be possible to do the opposite; declaring a specific increase of the overall security level, and using the sensitivity analysis to distribute the necessary enhancements to properties that would result in the least or cheapest modifications.

7.4 The ROME software

The objective of the ROME software is to demonstrate, evaluate and enhance the CAESAR method as well as evaluate existing or planned systems.

Implement sensitivity analysis

The sensitivity analysis described in section 7.3, page 72, would be possible to perform in real time, while modeling the system, as the overall security level estimate is calculated when using the current version of the ROME software.

A graphical sensitivity indicator could then be attached as a color, shape or text to each component indicating the relative or absolute impact an increase in security level or traffic control of that component would have on the overall security level.

Implement dynamic algorithms

In the current version of the ROME software, the algorithms aggregating modeled properties into the overall security level are hard coded into the software. If the CAESAR method were to be further developed, it would be of
considerable value for the developer of the method to be able to try out
different ideas more easily using the ROME software.

Such a modification would probably have a significant impact on the time
budget for further enhancements of the method, and should therefore be a
priority if continuing the work inside the main framework of the CAESAR
method.

**Implement automatic modeling**

Currently, the ROME software lets the user model an existing or planned
distributed information system. It would however be of great help if the
software would be able to search the network and map out components itself.

The modeling could also be done in real time, immediately redrawing the
modeled network when mobile units are connected and disconnected,
visualizing how changes affect the overall security level.

The software would of course not be able to model the system entirely,
especially not if some components are shielded by either external or internal
firewalls, but it would certainly aid the user of the software, and provide
system administrators monitoring the system with a valuable tool.
8 Abbreviations

BC  Before Christ
CAESAR A component-based approach to estimating the level of IT security of architecturally rendered distributed information systems
CC  Common Criteria
CIA  Confidentiality, integrity, and availability
CSL  Comparable security level (CAESAR)
IT  Information technology
ITSEC Information Technology Security Evaluation Criteria
LS  Logical significance (CAESAR)
LSC  Logical security contribution (CAESAR)
LR  Logical relation (CAESAR)
NSC  Neighboring security contribution (CAESAR)
OSL  Overall security level (CAESAR)
PC  Personal Computer
PDA  Personal Digital Assistant
PDR  Prevention, detection, and reaction
PR  Physical relation (CAESAR)
SI  Security indicator (Anderson’s Security Evaluation Framework)
SLE  Security level estimate (CAESAR)
SSL  System-dependent security level (CAESAR)
TCE  Traffic control estimate (CAESAR)
TCSEC Trusted Computer System Evaluation Criteria
TW  Trustworthiness rating (Anderson’s Security Evaluation Framework)
9 References


Greenwald, Michael & Gunter, Carl (2003), Computer security is not a science, Large-Scale Network Security Workshop, Landsdowne, VA.


NE (2004), Nationalencyklopedin, Bra Böcker AB.

Roberts, Fred (1984), Measurement Theory with Applications to Decision-making, Utility, and the Social Sciences, Addison-Wesley


10 Index

absolute scale, 16
architectural model, 7
assess, 15
assessment score, 28
attributes, 28
CAESAR, 34
CIA, 12
classic layered model, 9, 13
comparable security level, 49, 52
component, 6
consequence, 14
criteria, 28
distributed information system, 6
distributed system, 6
estimate, 15
evaluate, 15
factor, 14, 28
failure model, 7
fundamental model, 7
human-machine dimension, 13
interaction model, 7
interval scale, 16
logical relation, 41
logical security contribution, 45, 50
logical significance, 41
measure, 15
metric, 16, 28
micro-macro dimension, 9
neighboring security contribution, 45, 47
network-centric warfare, 1
nominal scale, 16
ordinal scale, 16
overall security level, 42, 44
PDR, 11
physical relation, 41
physical-logical dimension, 10
process, 7
ratio scale, 16
representation, 16
risk level, 16
scale type, 15, 65
securability, 16
security indicator, 21
security level, 14, 16
security level estimate, 39
sensitivity index, 29
survival, 12
system-dependent security level, 42, 45
traffic control estimate, 40
traffic generator, 39
traffic mediator, 39
unit, 15, 65
På svenska

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsmann i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

In English

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© Mikael Peterson