HAZARD ANALYSIS FOR SECURITY PROTOCOL REQUIREMENTS

Nathalie Foster*
Jeremy Jacob
Department of Computer Science
University of York
UK
{nathalie.foster, jeremy.jacob}@cs.york.ac.uk

Abstract This paper describes a process for the generation and analysis of security protocol requirements. It addresses some of the problems resulting from the inadequacies of present development methods. It is based on a hazard analysis technique which has been developed for safety critical systems engineering. This provides a structured method of analysis of the requirements whilst avoiding the problems of being too restrictive.

Keywords: security protocols, software engineering, requirements gathering and analysis, hazard analysis.

1. Introduction

1.1. Security Protocol Development

In comparison to the process of general software development, security protocol development is relatively unstructured. It is therefore hardly surprising that protocols are still being published that are later found to be vulnerable to attacks. Sound engineering practices need to be applied to protocol development and we need to consider the whole development lifecycle for protocols.

General models of software development have been proposed, the most famous of these are the Waterfall model by Royce[12] and the Spiral model by Boehm[5]. Many variations on these models have been suggested but common to all the models are a number of distinct activities: of these requirements gathering and analysis are the first [9, 13]. Al-

*Supported by an Engineering and Physical Sciences Research Council (EPSRC) studentship

though not perfect, these models go someway towards the development of higher quality software.

Research in protocol development has focused on the use of formal methods and logics in the design and verification of protocols at a late stage in the development. Little work has been carried out into the relevance to protocol development of other activities found in the software engineering process, such as requirements engineering.

We believe that many attacks and flaws could be guarded against through the proper gathering and analysis of the protocol requirements before the actual design. This would reduce the incidence of security failures due to the 'opportunistic exploitation of elementary design flaws' or 'implementation and management design errors' [1, 2].

In this paper, we address the issue of requirements gathering and analysis in protocol development. Our approach is based on a hazard analysis technique and is described in Section 1.2 and 2. It is used to analyse the goals of the protocol with reference to the security requirements of the protocol, in order to generate the low level protocol requirements.

Through the use of this technique, we are able to begin designing a protocol with a more thorough understanding about what it should do. We also have a higher level of confidence about the security of the protocol designed based on the requirements, before we carry out any verification of the protocol. This is necessary for the verification of the protocols: we have requirements which we can verify the protocol against, rather than having to guess what the requiremerits are before we can start verification.

1.2. Hazard Analysis

HAZOP. We propose the use of a hazard analysis technique which has its foundations in a method called Hazard and Operability Study (HAZOP) [6, 7]. HAZOP was developed by Imperial Chemical Industries (ICI) for the identification of hazards in process plant designs within the chemical industries, where the analysis is carried out on the pipework and instrumentation design of the plant. It has since been applied in the food-processing, pharmaceutical, nuclear, oil and gas industries and has also been adapted for use in the development of safety critical systems [11].

In a HAZOP study, a team identifies the entities and attributes of a design. A standard list of guide words is used to suggest deviations to these attributes. The deviations are analysed to determine their possible causes and effects and to consider what actions need to be taken to avoid or minimise the effects of the deviations. The results of the analysis are

GUIDE WORD	GENERIC MEANING
Omission	The service is never delivered, i.e. there is no communication. These are classified as either *total* or *partial.*
Commission	A service is delivered when not required, i.e. there is an unexpected communication. These are classified as either *spurious* or *repetition.*
Early	The service (communication) occurs earlier than intended. This may be *absolute* (i.e. early compared to a real-time deadline) or *relative* (early with respect to other events or communications in the system).
Late	The service (communication) occurs later than intended. As with early, this may be absolute or relative.
Value	The information (data) delivered has the wrong value.

Table 1. The SHARD guide words [10]

recorded in a table detailing the deviations, causes, effects, detection and protection, and the justification and recommendations. The analysis documentation is used to improve the safety of the system under study and may also be used in further investigation of the safety of the system.

SHARD. Software Hazard Analysis and Resolution in Design (SHARD) [10],is a 'projective computer system safety analysis technique based on HAZOP'. It is used to analyse designs and to obtain system safety related requirements for the detailed development of those designs. The guide words in SHARD are based on the communication of pieces of information, with specific values, at particular points in time (Table 1).

The analysis process in SHARD is even more structured than in HAZOP, with extra steps to be carried out in the analysis. The SHARD process is shown in the flow diagram in Figure 1.

The analysis is recorded in a table with at least the following column headings: *Guide word; Deviation; Possible Causes; Effects; Detection and Protection; Justification/Design Recommendations.*

The structured SHARD process and the more appropriate guide words for a system involving information flows lends itself to the analysis of security protocol requirements with some modifications. The application of SHARD to protocol requirements gathering and analysis is described

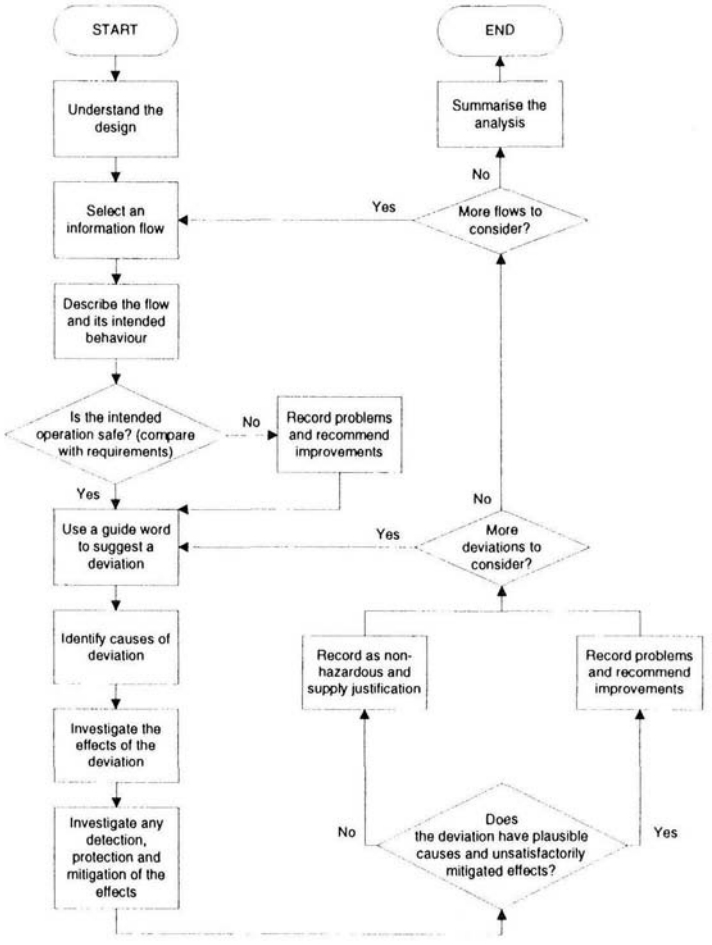

Figure 1. SHARD analysis [10]

in Section 2. An example use of the analysis of the requirements for an electronic commerce protocol is given in Section 3

2. Requirement Analysis for Security Protocol

2.1. Introduction

The aim of a requirements analysis process for security protocols is to analyse the high level requirements of the protocol to obtain the low level functional requirements of the protocol. These low level requirements can then be used in the design phase of the protocol development. This

ensures that the security requirements are carefully considered at the early stages of development and the features which are built into the protocol are justified.

The analysis is split into two levels, based on the distinction between high and low level protocol requirements:

- **High Level Requirements** These state what must be achieved by the end of the protocol run. The requirements are in terms of the differences in the *knowledge* state of the principals between the start and end of the protocol session, they indicate what the principals should and should not know. They are equivalent to the pre- and post-conditions of the protocol.

  The high level requirements can be subdivided into functional and non-functional requirements. The functional requirements indicate the functionality of the system under development; for example, "By the end of the protocol, Principal A should have received an order from Principal B". At this high level we are not interested in how this is achieved, nor what the order looks like. Non-functional requirements are more difficult to analyse, these requirements include safety, security and reliability requirements. In protocol requirements analysis, we are concerned with ensuring that protocols maintain a number of security properties, which are determined by the purpose of the protocol. We have designed this analysis method with reference to the following security properties: confidentiality; authenticity; integrity; non-repudiation; availability; timeliness; non-replicability. An example of a non-functional requirement is ''The order must be kept confidential between principals A and B.".

- **Low Level Requirements** These are the low level functional requirements of the protocol and are derived from the high level functional and non-functional protocol requirements. They state details such as what each protocol message will contain, how it will be constructed, any interactions between messages, such as if a particular message component is dependent on another message, and what checks will need to be carried out on the messages. An example of a low level requirement is: "The message should contain a component (such as a timestamp) to avoid replay attacks and to ensure timeliness of messages".

The analysis explores how an implementation may fail to meet its requirements, including how the external environment can affect the protocol. This may prompt further requirements of the protocol to de-

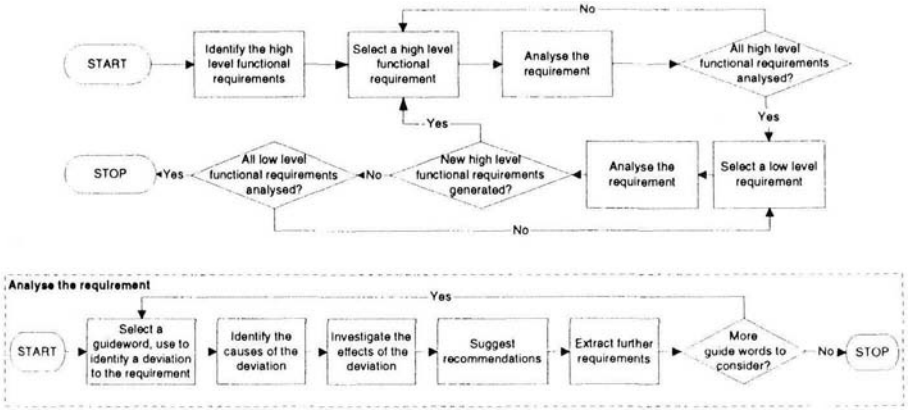

Figure 2. Hazard Analysis for Security Protocols Requirements

tect when such situations arise and also to protect or mitigate against the violations of the requirements.

Our method takes into account the differing views of the stakeholders in the protocol through the use of a team in the analysis. The team should consist of representatives of each of the stakeholders in the protocol and, ideally, someone who is familiar with attacks and flaws which are common in protocols, as well as the different verification techniques which can be used on the protocols.

2.2. Hazard Analysis for Security Protocols

The analysis process is based on the SHARD process using guide words to prompt deviations to the requirements and identifying the causes, effects, detection and mitigation mechanisms associated with these deviations.

Our analysis is carried out at the both the high and low levels of functional requirements. The analysis of the requirements will prompt further high level and low level functional requirements which will be subject to further analysis. Thus the analysis is an iterative process. It is outlined in Figure 2.

The guide words in the analysis process have been adapted to relate to message transfers, contents and checks on messages, to prompt deviations which make the requirements vulnerable to attacks which violate the required security properties. Once these vulnerabilities have been identified, measures can be taken, through the introduction of further

requirements to avoid the incorporation of the vulnerabilities into the design and implementation.

Table 2 contains the guide words and interpretations used to generate the deviations to the protocol requirements, these were influenced by the SHARD guide words in Table 1. In this table we also identify the security property violations which could result from the deviations suggested by the guide words. The guide words we have selected may not be the only guide words which could be used for security protocol analysis. Further guide words can be added to the process to reflect different properties required of different types of protocol.

Some of the steps in the analysis process of Figure 2 are described in more detail below:

Identification of high level functional requirements. The high level functional requirements are elicited from the informal scenario description which details the situation for which we wish to design a protocol. By identifying the *principals,* their *actions* and the *objects* on which they act, we can extract more structured requirements which describe the scenario which contain the following: initiating principal, responding principal, action, object.

Identification of Causes. This is based on the *primary - secondary - command* rule for identifying causes in SHARD. We interpret this as:

- **Primary (P)** causes are due to the failure of the principal who carries out the service. For instance, the principal may not have sent out the message, or may have sent out an incorrect message.
- **Secondary (S)** causes are due to the failure of the medium over which communication is made or an action or event is carried out, such as the network. Cases where an intermediary party, such as an intruder causes a deviation are also classed as secondary causes.
- **Command (C)** causes are due to the failure of the command which prompts the action to be carried out. Earlier messages in a protocol session act as a command, or prompt, to the principal to send out the next message. Therefore, if an incorrect message is received then a response dependent on that message may also be incorrect.

Identification of Effects. The immediate effects of the deviations are noted. Any possible *actions* **(A)** which can be carried out by the principals as a result of the deviation are identified and the *consequences* **(C)** of these actions are identified. The actions and their consequences

GUIDE WORD	GENERIC MEANING	SECURITY PROPERTIES VIOLATED
Omission	The event does not take place.	Availability
Commissior	The event which takes place is not as expected. The different types of commission: • *spurious:* a one off event. • *repetition:* a repeated event.	*Spurious:* Authenticity, Non-repudiat ion *Repetition:* Authenticity Non-repudiation Non-redication.
Value	The data obtained in the event has the wrong value arid this can be detected. This could be: • *total:* the data dclivered in the event is totally corrupted. • *extra:* an event occurs as expected but with some unexpectedextra data/behaviour. • *partial:* parts of the expected event are omitted.	Integrity, Authenticity, Non-repudiation
Disclosure	The data in this event has beendivulged to an unauthorised party.	Confidentiality.
Early	The event occurs earlier than intended. Early can be interpreted as: • *absolute:* early cornpared to a real-time deadline. • *relative:* early with respect to other events or communicationsin the svstem.	Timeliness.
Late	The event occurs later than intended. Late can be interpreted as: • *absolute:* late compared to a real-time deadline. • *relative:* late with respect to other events or communicationsin the svstem.	Authenticity, Timeliness, Availability.

Table 2. The protocol analysis guide words

must be considered because it is often a chain of events following from a deviation which results in an insecurity in the protocol. As noted in the SHARD analysis, effects may contribute to or be the causes of other deviations, therefore any dependencies should be recorded.

Recommendations. In order to protect the security of the protocol, recommendations to address the deviations are made. The recommendations are of three types:

- **Prevention (P)** Measures to prevent a potential violation of a security property, are incorporated into the design.
- **Detection (D)** Mechanisms to detect when, how and who violated the security of the protocol.
- **Reaction (R)** If we can detect when a violation has taken place then we can recover from the security violation by correction or mitigation mechanisms.

The recommendations depend upon the security properties which have been breached. In some circumstances we can react to the security violation and carry out an action to return the protocol to a secure state. However, in some cases it is impossible to recover from a security violation, such as when a confidentiality breach occurs. In such cases we must attempt to find protection mechanisms to prevent such security breaches. Similarly there are also situations in which detection of a security violation is very difficult. The choice of recommendations must be carefully considered to deal with such cases.

The recommendations can be implemented using a variety of methods. Software or hardware controls could be used to ensure that the protocol security is maintained; for example, encryption can be used to maintain integrity and confidentiality. Policies and physical controls can be used to govern the application of the protocols and information in a wider context.

Further high and low level requirements are elicited from the recommendations. These are then added to the list of requirements and are, in turn, analysed. If there are multiple recommendations to address the same problem, then design decisions about which recommendations to use will need to be taken and these should be documented.

Analysis Documentation. The analysis is documented in a table such as that in SHARD and HAZOP. The documentation table may also contain a column for recording comments arising in the course of the discussion, this is useful for recording other issues and cross references

relating to other parts of the analysis or design phase. Example headers in the documentation table are: *Guide word; Deviation; Causes; Effects; Recommendations; Comments.*

3. Example Application

In this section, we provide a partial example of the use of the requirements analysis process for security protocols.

3.1. Scenario

A vendor wishes to sell goods to its customers over the internet using an electronic commerce protocol. It is envisaged that the customer will send to the vendor an order for the goods and also payment details. The vendor will then be able to obtain payment through the customer's credit card company. In return the customer will obtain the goods ordered.

3.2. Example Analysis

Identification of High Level Functional Requirements. We extracted the following high level requirements from the scenario above by identifying the principals, actions and objects and their interactions:

1 Customer sends an order for goods to the vendor.

2 Customer sends payment details to the vendor.

3 Vendor submits the payment details to the payment authority.

4 Vendor obtains payment for the goods from the payment authority.

5 Vendor distributes the goods to customer.

High Level Analysis. Table 3 contains an analysis of the high level requirements "Customer sends and order for goods to the vendor" using the hazard analysis process. In a full analysis, a similar table is produced for each of the requirements.

Omission:	No order is made.
Cause:	(P) Customer doesn't send an order. (S) Order lost by network/intruder actions.
Effect:	(A) Customer waits indefinitely for response from vendor. (C) Vendor looses an order if not detected.

Table 3. Analysis of "Customer sends order for goods to vendor"

Recommendations:	(D) Timeout on waiting for response to order so customer does not wait indefinitely. (R) Recovery session to resend order. (D) A pre-protocol exchange enables vendor to detect if an order is missing. (P) Use a reliable network.
Comments:	Network reliability is out of the scope of this protocol since we have no control over the reliability of the internet. Prevention of intruder attacks is impossible, protection should make attacks infeasible.
Commission (Spurious):	An order takes place unexpectedly.
Cause:	(P) Customer accidentally sends the order. (S) An intruder fakes an order. (S) Network fault results in spurious order.
Effect:	(A) Vendor treats order as valid and waits indefinitely for a payment message which will not take place (if payment is before delivery).
Recommendations:	(D) Tinieouts on waiting for payment message so vendor doesn't wait indefinitely. (D) Order authentication. (D) Customer feedback to check ordeir is correct. (P) A pre-protocol exchange*so* valid orders received by vendor are riot unexpected.
Comment	As for Omission
Commission (Repetition):	An order is repeated.
Cause:	(P) Customer repeats an order intentionally. (P) Customer accidentally sends a repeat order. (S) An intruder replays the order maliciously. (S) Network fault causes message to be resent.
	(A) Vendor treats order as valid and customer receives unwanted goods. (A) Vendor rejects order and customer waits indefinitely for response from vendor.
Recommendations:	(D) Use of a fresh element (nonce or timestamp) to detect replay of an order. This allows valid repeat orders to take place. (D) Customer feedback to check order is correct.

Table 3. Analysis of "Customer sends order for goods to vendor"

Value (Total):	Order is totally corrupted
Cause:	(S) Corrupted on the network or by intruder.
Effect:	(A,C) Vendor rejects message as it is not identifiable as an order and customer waits indefinitely for response from vendor. (A) Message interpreted as an order, but not that intended by the customer.
Recommendations:	(D) Check integrity of order. (D) Customer feedback to check order is correct. (P) Avoid indefinite waiting by timing out waiting for a response to order. (R) Recovery session to resend the order (P) Use a reliable network.
Comment:	As for Omission.
Value (Extra):	Order is valid but there is some extra information with it.
Cause:	(P) Extra information added by customer. (S) Result of corruption on the network/by an intruder.
Effect:	(A) Order interpreted by vendor as an order with unwanted extra items included. (A) Order rejected by vendor and customer waits indefinitely for response from vendor.
Recommendations:	(D) Check integrity of order. (D) Customer feedback to check order is correct. (P) Avoid indefinite waiting by timing out waiting for a response to order. (R) Recovery session to resend the order. (P) Use a reliable network.
Comment:	As for Omission.
Value (Partial):	Only part of order message is received.
Cause:	(P) Customer missed off parts of order message. (S) Components of order message are lost on network/by an intruder.
Effect:	(A) Order accepted but parts of customer's order are missing. (A) Order rejected and customer waits indefinitely for response from vendor.
Recommendations:	(D) Check integrity of order. (D) Customer feedback to check order is correct.

Table 3. Analysis of ''Customer sends order for goods to vendor"

	(R) Recovery session to resend the order. (P) Use a reliable network.
Comment:	As for Omission.
Disclosure:	Order is disclosed.
Cause:	(C) Order is not protected and can be read by eavesdropper on network.
Effect:	(C) Customer's privacy is violated as order is public knowledge. (C) Vendor's order details are available to everyone, including their competitors.
Recommendations:	(P) Confidentiality protection of the order.
Early:	Order is received early.
	As for Commission (spurious).
Late:	Order is received late.
Cause:	(S) Delay on network or by an intruder.
Effect:	(A) Customer waits indefinitely for vendor's response to order.
Recommendations:	(D) Inclusion of a fresh component to enable the vendor to determine if a message is late. (D) Customer times-out waiting for messages for vendor's response to avoid indefinite waiting. (R) Recovery session to resend order message. (P) Use a reliable network.
Comment	As for Omission

Table 3. Analysis of "Customer sends order for goods to vendor"

Extraction of Further Requirements from High Level Analysis. The following requirements were extracted from the analysis of the requirement ''Customer sends an order for goods to vendor". In a full analysis, these are analysed in later iterations of the Hazard Analysis process.

- **High level requirements:**

  1 A recovery session should be available in case that order needs to be resent, if it is detected that order is incorrect or has not been received by the vendor.

  2 Pre-protocol exchange to ensure that vendor is alive and accepting orders and also so that vendor is able to anticipate receipt of orders.

3 Provide feedback (a confirmation of order) so customer can check that order is correct.

- **Low level requirements:**

1 Time-outs on waiting for orders and responses to orders to avoid principals waiting indefinitely.

2 Authentication of order messages.

3 A fresh element in order provides uniqueness of order, giving assurance that it has been created recently and allowing orders to be repeated.

4 Integrity checking and correction of order.

5 Confidentiality protection of order to protect customer's privacy.

6 Incorporation of a time component to detect if order is late.

Low Level Requirements. Table 4 contains an example of the low level analysis stage in the Hazard Analysis for Security Protocols process. This table shows the analysis of the low level requirement "A fresh element in order provides uniqueness of the order message and allows orders to be repeated." From this analysis we obtain further requirements for the protocol.

Omission:	No fresh element is included in the order message.
Cause:	(P) Not included by customer. (S) Unavailability of fresh element generator.
Effect:	(C) Vendor cannot check if order was created recently. (A) Intruder is able to replay order message.
Recommendations:	(D) Vendor checks for fresh element in order and reject order if it contains no fresh element. (P) Use of reliable fresh element generator.
Commission (spurious):	Fresh element is unexpectedly in order message.
Comment	Not applicable since message is expected to contain a fresh element.
Commission Repetition):	A fresh element is reused in order message.
Cause:	(P) Reused by principal. (S) Element replayed by intruder/network.

Table 4. Analysis of "A fresh element is included in order message"

Effect:	(A) Rejection of message by vendor. (C) Customer does not receive goods.
Recommendations:	(D) Check fresh element and reject if repeated. (R) Recovery session to deal with invalid fresh elements.
Value (Total):	A fresh element of unexpected format is in order message.
Value (Extra):	Expected fresh element plus extra information is in order message.
Value (Partial):	Partial fresh element is in order message.
Cause:	(P) Included by customer. (S) Element in format provided by generator.
Effect:	(A) Order message is rejected by vendor. (C) Customer does not receive goods.
Recommendations:	(D) Check fresh element and reject if invalid. (R) Recovery session for cases where fresh element is invalid.
Disclosure	Fresh element is disclosed.
Cause:	(P) Not protected by principal. (S) Disclosed on network/by intruder.
Effect:	None. Public knowledge should reveal nothing.
Early	Fresh element in order message is early.
Cause:	(S) Generator dispenses fresh items too early. (S) Other messages have not yet been received.
Effect:	(C) It is known that order message has been created recently and so is valid.
Recommendations:	(P)The fresh element generator for customer and vendor should be periodically synchronised.
Late	Fresh element in order message is late.
Cause:	(P) Principal sends order message late. (S) Fresh element generator generates late. (S) Message delayed by intruder/network.
Effect:	(A) Order is rejected because it is too late.
Recommendations:	(D) Check that messages are timely/fresh and reject if late. (P) Periodic synchronisation of customer and vendor fresh element generators. (R) Recovery session in case of late messages.

Table 4. Analysis of “A fresh element is included in order message"

Extraction of Further Requirements from Low Level Analysis. The analysis of the low level requirement "A fresh element is included in the order message" identified the following requirements of the protocol:

- High level requirements

  1 Recovery session to deal with messages with invalid or late fresh elements.

- **Low level requirements**

  1 Periodic synchronisation of fresh element generator.

  2 Use of reliable fresh element generator.

  3 Checks to ensure fresh elements are of valid format/timely and reject if not.

  4 Checks for the fresh element in order message and rejection if no fresh element.

3.3. What has been gained from this analysis?

From this fragment of an example of a Hazard Analysis for Security Protocol requirements, we can gain insight into the intuitive steps taken by the designer. We can identify items which need to be kept confidential, checked for authenticity, integrity and freshness, recovery sessions and feedback to the principals which is required. Using this analysis process, we can trace the generation of requirements and justify the features which are built into the protocol.

In a full analysis, each of the recommendations would be justified in more detail and labelled to make it easier to trace and refer to the protocol requirements during the later development phases.

4. Conclusions

In this paper, we have described a process for the gathering and analysis of the requirements of security protocols before the actual design of the protocol. This is the traditional starting point in the software engineering life cycle. It is preferable to spend time in the early stages of the protocol development than to risk a compromise of security, when the protocol is put into use. Our approach differs from previous research into the requirements of protocols which focused on the use of requirements in the verification of protocols [14]; for example, Syverson and Meadows [15] formalised the requirements of authentication protocols and used them to verify and find attacks on the Neuman-Stubblebine protocol.

The hazard analysis approach described in this paper provides a simpler, more structured and systematic approach to deviation identification than the heuristic methods in the literature. Work on inquiry-based requirements analysis [8] relies on the use of *what-if*? questions to prompt deviations. In goal-based requirements analysis [3, 4], trivial obstacles are assigned to each goal to investigate the possible ways in which a goal may fail to complete. These obstacles are identified through the use of an extensive set of heuristics. The obstacle analysis is further elaborated through scenario analysis which examines the concrete circumstances under which goals may fail. Lamsweerde and leitier [16] present formal and heuristic methods for obstacle identification and resolution based on temporal logic.

An advantage of the hazard analysis approach for protocol requirements over the temporal logic approach is the focus of the analysis on security features of the protocol. The temporal logic approach is very formal, requiring the gathering of the preconditions for the negation of the goal expressed in logic, these preconditions are obstacles to the goal. Some formal techniques have missed attacks due to their over abstraction of the protocols, since security attacks may be the result of the exploitation of properties which are not easily expressible in logic.

Our approach to the analysis of the requirements does not, of course, guarantee that all the attacks are avoided and secure protocols will be designed. The requirements analysis process is useful for highlighting weaknesses and flaws which have previously occurred in protocols.

Attack and threat avoidance techniques prompted by the guidelines may not be appropriate, for instance, if the recommendations would be too costly or time consuming. Consideration of the recommendations should be carefully evaluated with respect to the requirements of the protocol stakeholders. However, just being aware of potential problems which may be caused by a particular requirement is an important benefit of using the method. In such situations, if it is considered appropriate, higher level requirements may be weakened in the light of the analysis.

Our method is suitable for identifying and investigating common threats and attacks on protocols and prompting protection mechanisms against them. This method is a step forward in providing a more structured approach to the development of secure protocols and we believe that this approach to requirements analysis can be applied more widely in the field of computer security.

Acknowledgments

We would like to thank John Clark, Jonathan Moffett, colleagues in the High Integrity Systems Engineering research group and the anonymous reviewers for their helpful comments.

References

[1] Ross Anderson. How to Cheat at the Lottery (or, Massively Parallel Requirements Engineering). Invited Talk at the 15th Annual Computer Security Applications Conference, Phoenix, Arizona, December 1999.

[2] Ross J. Anderson. Why Cryptosystems Fail. *Communications of the ACM,* 37(11):32-40, 1994.

[3] Annie I. Antón. Goal-Based Requirements Analysis. In *2nd IEEE International Conference on Requirements Engineering,* pages 136–144, April 1996.

[4] Annie I. Antón. *Goal Identification and Refinement in the Specification of Software-Based Information Systems.* PhD thesis, Georgia Institute of Technology, Atlanta, June 1997.

[5] Barry W. Boehm. A Spiral Model of Software Development and Enhancement. *IEEE Computer,* pages 61–72, May 1988.

[6] CISHEC. A Guide to Hazard and Operability Studies. The Chemical Industry Safety and Health Council of the Chemical Industries Association Ltd, 1977.

[7] T. Kletz. *HAZOP and HAZAN: Identifying and Assessing Process Industry Hazards.* Institution of Chemical Engineers, third edition, 1992.

[8] Colin Potts, Kenji Takahashi, and Annie I. Antón. Inquiry-Based Requirements Analysis. *IEEE Software* ,11(2):21-32, March 1994.

[9] Roger S. Pressman. *Software Engineering: A Practitioner's Approach.* McGraw Hill, 5th edition, 2000. (European adaption by Daryl Ince).

[10] D.J. Pumfrey. The Principled Design of Computer System Safety Analyses. Dphil Thesis, University of York, 2000.

[11] Felix Redmill, Morris Chudleigh, and James Catmur. *System Safety: HAZOP and Software HAZOP.* Wiley, 1999.

[12] W. W. Royce. Managing the Development of Large Software Systems. In *Proceedings of IEEE WESCON,* pages 1–9, 1970. Reprinted in Thayer R.H.(ed.) (1988) *IEEE Tutorial on Software Engineering Project Management.*

[13] Ian Sommerville. *Software Engineering.* Addison Wesley, 6th edition, 2000.

[14] Paul Syverson and Catherine Meadows. A Logical Language for Specifying Cryptographic Protocol Requirements. In *IEEE Symposium on Research into Security and Privacy,* pages 165–177. IEEE Computer Society Press, 1993.

[15] Paul Syverson and Catherine Meadows. Formal Requirements for Key Distribution Protocols. In Alfredo De Santis, editor, *Advances in Cryptology - EUROCRYPT '94,* volume 950 of *Lecture Notes in Computer Science,* pages 320–331. Springer, May 1994.

[16] Axel van Lamsweerde and Emmanuel Letier. Handling Obstacles in Goal-Oriented Requirements Engineering. *IEEE Transactions on Software Engineering,* 26(10):978-1005, October 2000.