A Formal Verification Centred Development Process for Security Protocols

Professor Tom Coffey,
Data Communications Security Laboratory
University of Limerick,
Ireland

Phone: +353-61-202268
Fax: +353-61-338176
Email: tom.coffey@ul.ie

Dr Reiner Dojen
Data Communications Security Laboratory
University of Limerick,
Ireland

Phone: +353-61-202268
Fax: +353-61-338176
Email: reiner.dojen@ul.ie

A Formal Verification Centred Development Process for Security Protocols

Tom Coffey, Reiner Dojen
Data Communications Security Laboratory
University of Limerick, Ireland

ABSTRACT
This paper concerns the correct and reliable design of modern security protocols. It discusses the importance of formal verification of security protocols prior to their release by publication or implementation. A discussion on logic-based verification of security protocols and its automation provides the reader with an overview of the current state-of-the-art of formal verification of security protocols. The authors propose a formal verification centred development process for security protocols. This process provides strong confidence in the correctness and reliability of the designed protocols. Thus, the usage of weak security protocols in communication systems is prevented. A case-study on the development of a security protocol demonstrates the advantages of the proposed approach. The case-study concludes with remarks on the performance of automated logic-based verification and presents an overview of formal verification results of a range of modern security protocols.

INTRODUCTION

The security of electronic networks and information systems is a critical issue for the use of new technologies in all fields of life. Mobile and fixed networks are trusted with highly sensitive information. Thus, security protocols (also called cryptographic protocols) are required to ensure the security of both the infrastructure itself and the information that runs through it. These security protocols can be thought of as the keystones of a secure architecture. Basic cryptographic protocols allow agents to authenticate each other, to establish fresh session keys for confidential communication and to ensure the authenticity of data and services. Building on such basic cryptographic protocols, more advanced services like non-repudiation, fairness, electronic payment and electronic contract signing are achieved.

The massive growth in communications technologies – in particular in the wireless sector – cause an ever changing environment for today's communication services. The pervasive nature of emerging Information and Communications Technologies has added new areas of concern to information security. For example, the increased virtual and physical mobility of users enhances their possibilities for interaction but leads to an increasing demand for reliable trust relationships. To address new security risks and threats arising from such changes in the communication environment a constant supply of novel security protocols is required.

However, security protocols are vulnerable to a host of subtle attacks and designing protocols to be impervious to such attacks has been proven to be extremely challenging and error prone. This is evident from the surprisingly large number of published protocols, which have later been found to contain various flaws, in many cases several years after the original publication (Huima, 1999; Brackin, 2000; Zhang & Varadharajan, 2001; Gürgens & Rudolph, 2002; Newe & Coffey, 2002; Coffey, Dojen & Flangan, 2003a-d; Ventuneac, Coffey & Newe, 2004; Zhang & Fang, 2005; Ventuneac, Dojen & Coffey, 2006).

Motivation

The importance of formal verification of cryptographic protocols during the development process cannot be overstated (Coffey, Dojen & Flanagan, 2003a), as the absence of formal verification of these protocols can lead to flaws and security errors remaining undetected. However, an ad-hoc

survey carried out at the Data Communication Security Laboratory, UL in 2005 on over 200 recent papers on cryptographic protocols revealed that only 15% of these publications contained formal verifications.

The BCY protocol (Beller, Chang & Yacobi, 1993) provides a very interesting case study to highlight the problems with the correct design of security protocols. Published in 1993, the BCY protocol demonstrated the feasibility of public-key cryptography in mobile communications. Subsequently, Carlsen (1994) discovered weaknesses in the protocol and published an amended protocol. Weaknesses in Carlsen's amended protocol were discovered by Mu and Varadharajan (1996) and another modified protocol was published. Horn, Martin and Mitchell (2002) identified a weakness in the Mu and Varadharajan version of the BCY protocol, but did not publish a corrected version. Coffey, Dojen and Flanagan (2003a) published formal verifications of the BCY protocol and its derivates. In addition to all previously identified weaknesses a hitherto unknown flaw in the BCY protocol and its derivatives were detected. A corrected protocol was proposed and also formally verified. This case study highlights that the design of security protocols is a highly complex and error-prone process. It also shows that formal verification is an imperative step in the design of security protocols.

This chapter advocates the use of formal verification during the development of modern security protocols. It discusses the importance of formal verification of security protocols prior to their release by publication or implementation. A discussion on logic-based verification of security protocols and its automation provides the reader with an insight into the current state-of-the-art of formal verification of security protocols. The authors propose a formal verification centred development process for security protocols. This approach results in a strong confidence in the correctness and reliability of the designed protocols and prevents the usage of weak security protocols in communication systems. Further, a case-study demonstrates the advantages of the proposed approach to the design of security protocols. This case study provides a discussion on the difficulties involved in designing correct and reliable security protocols. As an example, the authentication and key-agreement protocol by Beller, Chang and Yacobi (1993) is taken. The BCY protocol contains a number of well-known weaknesses and, thus, the ability of the proposed process to detect these weaknesses is demonstrated. The case-study concludes with remarks on the performance of automated logic-based verification and presents an overview of formal verification results of a range of modern security protocols.

FORMAL VERIFICATION OF SECURITY PROTOCOLS

Needham and Schroeder (1978) are generally credited with the introduction of formal methods as a possible tool for cryptographic protocol analysis. However, the first experimental work in this area was done by Dolev, Even and Karp (1982) and by Dolev and Yao (1983), who developed a set of polynomial-time algorithms for deciding the security of a restricted class of protocols. It was soon found that relaxing the restrictions on the protocols made the security problem undecidable (Even & Goldreich, 1983). However, Dolev and Yao's work was significant in that it was the first to develop a formal model that provided:

1. The possibility of multiple executions of the protocol running concurrently.
2. Cryptographic algorithms behaving like black boxes, obeying a set of algebraic properties.
3. A model of an intruder who can read, alter and destroy traffic, and who can also control some legitimate members of the system.

Based on the Dolev-Yao model, new verification techniques for the analysis of security protocols have been developed. These can be divided by their underlying principles into:

1. State-Space-based techniques like The Interrogator (Millen, Clark & Freedman, 1987), Murφ (Dill, 1996), FDR (Lowe, 1996), NRL Protocol Analyser (Meadows, 1996), Huima's

model checker (Huima, 1999), Brutus (Clarke, Jha & Marrero, 2000) and Athena (Song, Berezin & Perrig 2001).
2. Logic-based techniques using logics of belief and/or knowledge like the BAN logic (Burrows, Abadi & Needham, 1990), the GNY-logic (Gong, Needham & Yaholom, 1990), AUTLOG (Kessler & Wendel, 1994), SVO (Syverson & van Oorschot, 1994), CS logic (Coffey & Saidha, 1997), KM logic (Kudo & Mathuria, 1999), ZV logic (Zhang & Varadharajan, 2001), SW Logic (Stubblebine & Wright, 2002) and CPL (Kramer, 2004).
3. Theorem Proving Techniques based on generic theorem provers like Isabelle (Paulson, 98), or on special purpose theorem proving techniques developed for formal verification of cryptographic protocols like CSP (Schneider, 1998) and TAPS (Cohen, 2000).

Logic-Based Techniques

Logic-based formal verification of cryptographic security protocols proves the correctness of the protocol against its goals. The technique of logic-based formal verification is accredited largely to Burrow, Abadi and Needham (1990), developers of the BAN logic. This work initiated intense research in the area of logic-based formal verification and several logics, such as the logics of Gong, Needham and Yaholom (1990), Syverson and van Oorschot (1994), Coffey and Saidha (1997) and Kudo and Mathuria (1999) have been developed on the basis of BAN.

Logic-based verification of cryptographic protocols involves a process of deductive reasoning, where the desired protocol goals are deduced from the initial assumptions and message exchanges of the protocols as illustrated in Figure 1.

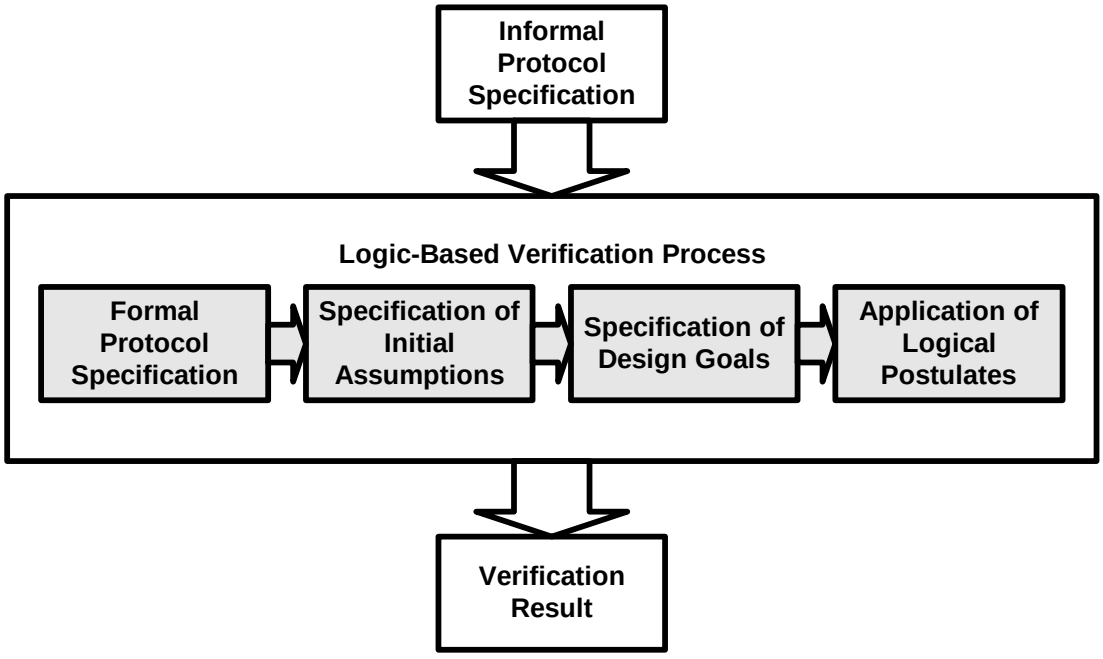

Fig. 1: Process of Logic-Based Verification

The first steps in logic-based verification involve specifying the protocol steps, the initial assumptions and the protocol goals in the language of the logic. The final verification step concerns the application of logical postulates to establish the beliefs and possessions of protocol principals. The objective of the logical analysis is to verify whether the desired goals of the protocol can be derived from the initial assumptions and protocol steps. If such a derivation exists, the protocol is successfully verified; otherwise, the verification fails. A successfully verified protocol can be considered secure within the scope of the logic. On the other hand, even the results of a failed verification are useful, as these may point to missing assumptions or weaknesses in the protocol. If a weakness is discovered, the protocol should be redesigned and re-verified.

Logic-based techniques have an advantage in that they are usually decidable and often efficiently computable, and thus can be completely automated (Kessler & Wedel, 1994; Brackin, 2000; Dojen & Coffey, 2005). Numerous protocol flaws in security protocols have been detected by researchers employing modal logics (Burrows, Abadi & Needham, 1990; Gong, Needham & Yaholom 1990; Kessler & Wedel, 1994; Syverson & van Oorschot, 1994; Coffey & Saidha, 1997; Brackin, 2000; Zhang & Varadharajan, 2001; Newe & Coffey, 2002; Coffey & Dojen, 2003; Coffey, Dojen & Flanagan, 2003abc; Coffey, Saidha & Burrows, 2003; Newe & Coffey, 2003abc; Ventuneac, Coffey & Newe, 2004).

Automation of Logic-based Verification

The manual application of formal techniques remains difficult and error-prone. As a result, the use of formal verification techniques for cryptographic protocols has not found widespread use outside the research community. It is important to get the advancements in formal verification techniques out to the people who can make best use of them, namely the designers and evaluators of the cryptographic systems that are being deployed in today's communication networks. Probably the highest obstacle currently prohibiting widespread use of formal techniques at the early development stages of cryptographic protocols is the complexity of their application. While some techniques have been already automated, often the use of these automated tools remains similarly complex to manual application.

Logic-based techniques are often developed with manual application in mind. Most current attempts to automate verification logics use generic theorem provers, such as HOL, Coq, Isabelle or PVS. However, logics often need to be modified to be automated by such generic theorems provers. Further, theorem provers usually only return results of the performed proofs, but will not provide any details of how these are achieved. However, in the case of security protocol verification, such details are very useful for the protocol designer. This is particularly true in the case of a failed verification, as the details of the proof are helpful in identifying the exact cause of the verification failure. Specialised automation techniques can be used to overcome such limitations. For example, Dojen and Coffey (2005) developed a novel automation technique for logic-based verification of security protocols. The main advantages of the novel technique include low resource requirement, traceability of verification results and the ability to accurately model manual verification logics.

A FORMAL VERIFICATION-CENTERED DEVELOPMENT PROCESS FOR SECURITY PROTOCOLS

The design of reliable and correct security protocols is an intricate and error-prone process. This is evident from the large number weaknesses found in protocols, in many cases years after their publication. While it can be shown that complete protocol security is undecidable (Cervesato et. al., 1999), formal verification provides a level of confidence in the security of protocols unrivalled by informal or intuitive techniques.

Traditionally, such informal techniques are used to verify security protocols during the development process. However, the absence of formal verification of these protocols prior to their publication can lead to flaws and security errors remaining undetected. Frequently, such flaws are discovered by subsequent formal verification of published protocols and amendments are proposed (Huima, 1999; Gürgens & Rudolph, 2002; Coffey, Dojen & Flanagan, 2003a; Newe & Coffey, 2003a; Ventuneac, Coffey & Newe, 2004; Zhang & Fang, 2005). Figure 2 illustrates this approach to security protocol design.

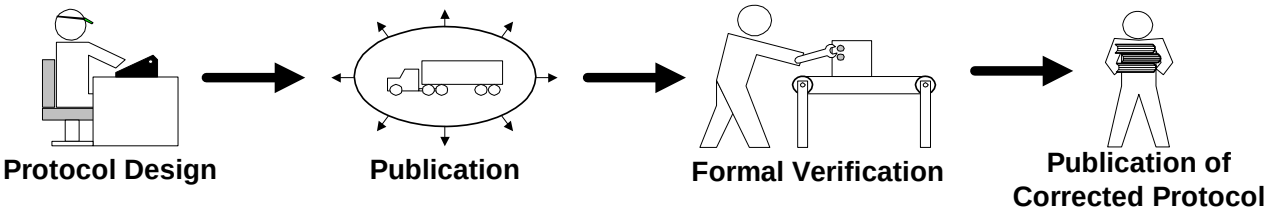

Fig. 2: Traditional Development process of Security Protocols

The large number of published protocols, which have later been found to contain various flaws, shows that this traditional development process is inadequate and can result in weak protocols being employed in communication systems. The main difficulty in the development of effective security protocols is to address the vast possibilities of adversaries to gain information (Dojen & Coffey, 2005). In contrast to communication protocols, where the main issues are reachability of all legal states and avoidance of infinite loops, security protocol verification deals with the gain of information by adversaries. The adversaries can be either passive or active. Further, adversaries might use simultaneous protocol runs to gain information. One has to keep in mind that adversaries will not play fair by any rules. Therefore, formal verification should be considered an imperative step in the design of security protocols (Coffey, Dojen & Flanagan, 2003a). As a consequence, the authors propose a formal verification centred development process as illustrated in Figure 3.

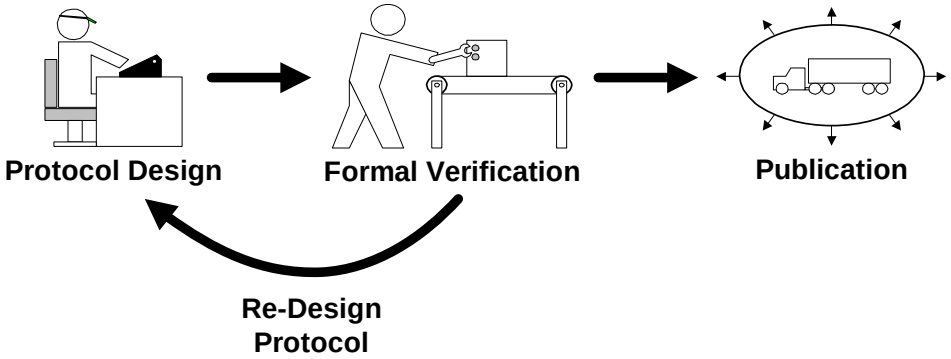

Fig.3: Proposed Development process for Security Protocols

In this development process, a draft of the protocol is subject to formal verification prior to its publication. If any weaknesses of the protocol are discovered during the formal verification, the draft is passed back to the design stage and corrected. This corrected version of the protocol will be again subject of the formal verification process. Design and Verification of a security protocol are considered two inseparable steps: protocol design and formal verification are iterated until formal verification does not reveal any weaknesses of the protocol. Only then is the protocol published or implemented. The incorporation of formal verification as an integrated part of the development process significantly increases the confidence in the correctness of the security protocol and, thus, risk of using faulty protocols in electronic networks or information systems drastically minimized.

CASE-STUDY: DESIGNING A PROVABLY SECURE CRYPTOGRAPHIC PROTOCOL

This case-study demonstrates the advantages of the proposed development process by applying it to the BCY protocol (Beller, Chang & Yacobi, 1993). The BCY protocol is chosen as it contains a number of well-known problems. The original protocol is treated as an initial draft of a newly designed protocol and subject to formal verification. Subsequently, the protocol is re-designed to address the detected weaknesses. Thus, the ability of the proposed development process to prevent the usage of weak protocols is demonstrated.

Formal verification during this case study is performed by an automated verification system that is based on Layered Proving Trees (Dojen & Coffey, 2005) and the GNY Logic (Gong, Needham & Yaholom, 1990). Figure 4 outlines the structure of the verification system. It features a library incorporating definitions of the verification logics, including the logical postulates and the grammar for the logic. The user inputs the formal protocol specification which consists of the initial assumptions, the protocol steps and the protocol goals and selects the logic to be used from the library. The system will automatically apply the logical postulates and provides the verification results.

A successfully verified protocol can be considered secure within the scope of the logic. The results of a failed verification help to identify any weaknesses in the protocol or any missing initial assumptions. Thus, the reasons for protocol failure are detected and can be corrected. The corrected version of the protocol should be again subject to formal verification.

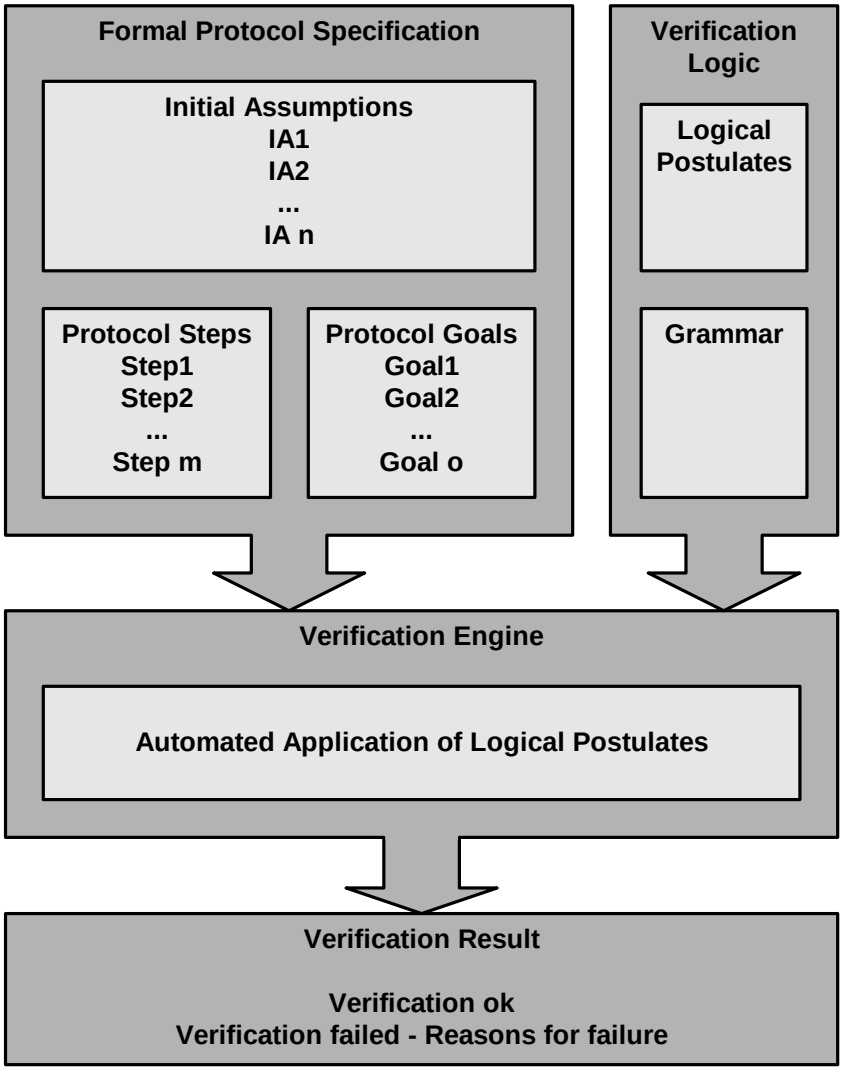

Fig. 4: Verification System Overview

The BCY Protocol

The protocol by Beller, Chang and Yacobi (1993), referred to as the BCY protocol is aimed at providing authentication and key agreement in low-power portable devices such as mobile phones. It demonstrates the computational feasibility of using public-key cryptography in mobile communications. The protocol combines a Diffie-Hellman key exchange (Diffie & Hellman, 1976)

with the modular square root (MSR) encryption technique (Rabin, 1979) to reduce the computational burden on the low-power portable device. Some weaknesses of the BCY protocol have been identified previously by Carlsen (1994), Mu and Varadharajan (1996), Horn, Martin and Mitchell (2002) and Coffey, Dojen and Flanagan (2003a). The following notation is used to describe the BCY protocol.

U: An identifier of the user.
V: An identifier of the service provider.
S: An identifier of the certification authority.
Nx: A random nonce generated by X.
TSx: An expiration time for the certificate for X.
Kx+: public key of X.
Kx-: private key of X.
KK: a key-encrypting key.
SK: a session key.
dataX: identification data for X

As the service provider uses two public keys, one for Diffie-Hellman key agreement and one for MSR encryption, these will be termed Kvd+ and Kvm+ respectively. Certificates, denoted by (X1,…,Xn)Ks-, contain the components X1 to Xn signed by a certification authority. The steps of the protocol are as follows:

BCY1: $V \rightarrow U$: {V, Kvd+,Kvm+}Ks-
U computes Y = {Nu}Kvm+, KK={Kvd+}Ku-, SK = {Nu}KK
BCY2: $U \rightarrow V$: Y, {{U,Ku+}Ks-}Nu
V computes R_U ={Y}Kvm-, KK={Ku+}Kvd-, SK = {Nu}KK
BCY3: $V \rightarrow U$: {dataV}SK
BCY4: $U \rightarrow V$: {dataU}SK

Formal Logic-Based Verification of the BCY Protocol

Once the initial protocol is created it is subject to formal verification. In order to verify a security protocol it must be translated into the language of the used verification technique. The following input script represents the formal protocol specification and has been used to verify the BCY protocol with the automated verification system:

```
// BCY Formal Protocol Specification
// Assumptions
A1 : U possess KuPriv;
A2 : U possess {U,KuPub}KsPriv;
A3 : U possess dataU;
A4 : U possess Nu;
A5 : U possess KsPub;
A6 : U believe fresh(Nu);
A7 : U believe recognised(dataV);
A8 : U believe recognised(V);
A9 : U believe hasKey(S,KsPub);
A10: U believe S hasJurisdiction S believe allStatement;
A11: U believe S hasJurisdiction hasKey(V,KvmPub),hasKey(V,KvdPub);
A12: V possess KvdPriv;
A13: V possess KvmPriv;
A14: V possess {V,KvdPub,KvmPub}KsPriv;
A15: V possess dataV;
```

```
A16: V possess KsPub;
A17: V believe recognised(dataU);
A18: V believe recognised(U);
A19: V believe hasKey(S,KsPub);
A20: V believe S hasJurisdiction S believe allStatement;
A21: V believe S hasJurisdiction hasKey(U,KuPub);

// Steps
S1 : U told notHere({(V,KvdPub,KvmPub),[S believe hasKey(V,KvmPub),
                      hasKey(V,KvdPub)]}KsPriv);
S2 : V told notHere({notHere(Nu)}KvmPub),notHere({notHere({(U,KuPub),
                      [S believe hasKey(U,KuPub)]}KsPriv)}Nu);
S3 : U told notHere({dataV}{Nu}{KvdPub}KuPriv);
S4 : V told notHere({dataU}{Nu}{KuPub}KvdPriv);

// Goals
G1 : U believe hasKey(V,KvmPub),hasKey(V,KvdPub);
G2 : U possess {Nu}{KvdPub}KuPriv;
G3 : U believe fresh({Nu}{KvdPub}KuPriv);
G4 : U believe shareKey(U,{Nu}{KvdPub}KuPriv,V);
G5 : U believe V conveyed {dataV}{Nu}{KvdPub}KuPriv;
G6 : U believe fresh({dataV}{Nu}{KvdPub}KuPriv);
G7 : V believe hasKey(U,KuPub);
G8 : V possess {Nu}{KuPub}KvdPriv;
G9 : V believe fresh({Nu}{KuPub}KvdPriv);
G10: V believe shareKey(V,{Nu}{KuPub}KvdPriv,U);
G11: V believe U conveyed {dataU}{Nu}{KuPub}KvdPriv;
G12: V believe fresh({dataU}{Nu}{KuPub}KvdPriv);
```

The assumptions section of the script identifies the possessions and believes of protocol participants at the beginning of a protocol run. Here, the assumptions A1 to A5 model U's possession of its own private key, certificate, identification data, random nonce and the certification authority's public key, respectively. The next assumptions model U's believe in the freshness of its nonce (A6), its own ability to recognise both V's identification data (A7) and name (A8). Further, U believes that KsPub truly is the certification authority's proper public key (A9) and that the certification authority is honest (A10) and can be trusted to provide public keys for V (A11). The assumptions A12 through to A21 model the matching possessions and believes for V (note that V has two public/private keys and V has no information about Nu).

The steps section models the message exchange between principals. Note that GNY's message extensions are indicated with square brackets. These message extensions are used to convey some principal's beliefs to other principals.

The goals section states the expected believes and possessions required for a successful verification of the protocol. Here the goal G1 asserts that U believes the received public keys truly belong to V. Goals G2, G3 and G4 model key agreement for U, i.e. that U possesses the session key, believes it to be fresh and that it is a suitable key for use with V. Authentication of V by U is modelled through goals G5 and G6 by stating that U believes that message BCY3 has been send by V and that it has been send during the current protocol run. The goals G7 through to G12 assert the matching conditions for V.

After completion of the verification process the following results are obtained for the BCY protocol:

```
(1) : U believe hasKey(V,KvmPub),hasKey(V,KvdPub)is False
(2) : U possess {Nu}{KvdPub}KuPriv is True
(3) : U believe fresh({Nu}{KvdPub}KuPriv) is True
(4) : U believe shareKey(U,{Nu}{KvdPub}KuPriv,V) is False
(5) : U believe V conveyed {dataV}{Nu}{KvdPub}KuPriv False
(6) : U believe fresh({dataV}{Nu}{KvdPub}KuPriv) is True
(7) : V believe hasKey(U,KuPub) is False
(8) : V possess {Nu}{KuPub}KvdPriv is True
(9) : V believe fresh({Nu}{KuPub}KvdPriv) is False
(10): V believe shareKey(V,{Nu}{KuPub}KvdPriv,U) is False
(11): V believe U conveyed {dataU}{Nu}{KuPub}KvdPriv is False
(12): V believe fresh({dataU}{Nu}{KuPub}KvdPriv) is False
```

These results identify the following failed goals:

1. U cannot establish validity of V's public keys/certificate (Goal 1)
2. U cannot establish key agreement with V (Goal 4)
3. U cannot authenticate V through message BCY 3 (Goal 5)
4. V cannot establish validity of U's public key/certificate (Goal 7)
5. V cannot establish key agreement (Goals 9, 10)
6. V cannot authenticate U through message BCY 4 (Goals 11,12)

These failures stem essentially from two weaknesses of the protocol:

- The principals cannot validate the certificates.
- V cannot establish that the session key is fresh.

Due to these weaknesses the original BCY protocol provides neither authentication nor key agreement.

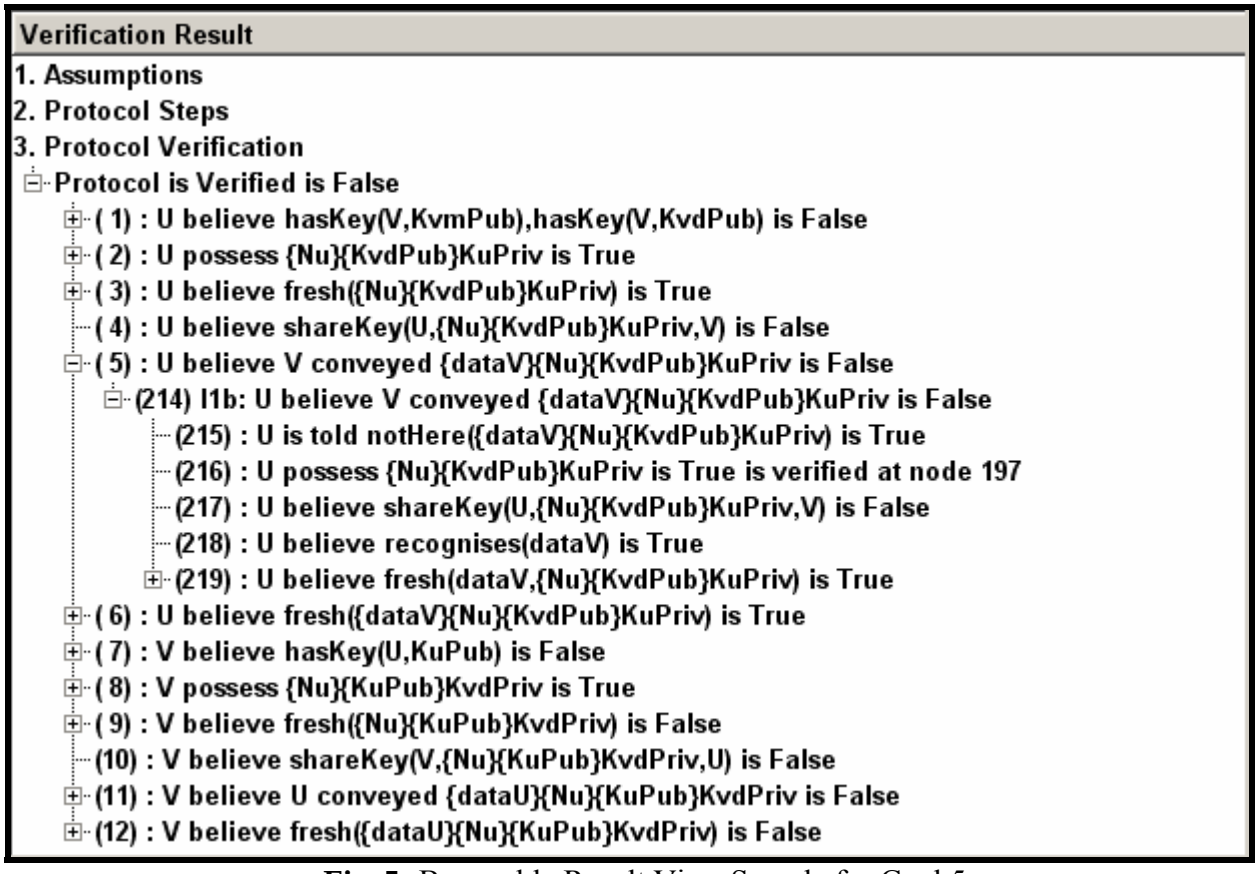

Fig. 5: Browsable Result View Sample for Goal 5

The used automated verification tool presents the verification results in a browsable tree, which provides detailed information about the performed reasoning. This allows the use to investigate the reasons for any identified protocol failures. As an example, Figure 5 outlines the reasoning used to establish Goal 5. Here, the result shows that Goal 5 fails since U does not believe that SK is a suitable shared key with V (node 217).

Re-Design of the BCY Protocol

In light of the weaknesses discovered in formal verification and analysis, the redesign of the BCY protocol can be implemented with focus on two aspects:

- Validate freshness of the certificates exchanged in the key agreement stage.
- Enhance the session key so that V can establish its freshness.

The following amendments are proposed:

Firstly, an expiration time TS is included in U's and V's certificate. Thus, the certificates are changed to {U,KuPub,TSu}KsPriv and {V,Kv1Pub,Kv2Pub,TSv}KsPriv respectively. These amendments to the certificates avoid attacks where an intruder replays a compromised certificate.

Secondly, a nonce Nv, created by V, is added and included in the construction of the session key, i.e. SK becomes {Nu,Nv}{Kv1Pub}KuPriv and {Nu,Nv}{KuPub}Kv1Priv for U and V respectively. As both parties now contribute towards the session key, each can confirm its freshness. In particular, V is now able to confirm freshness of SK and, consequently, is able to derive that SK is a suitable secret shared with user U. This results in the following amended protocol:

BCY'1: V → U: {V, Kvd+,Kvm+, TSv}Ks-
U computes Y = {Ru}Kvm+, KK={Kvd+}Ku-, SK = {Ru,Rv}KK
BCY'2: U → V: Y, {{U,Ku+, TSu}Ks-}Ru
V computes Ru = {Y}Kvm-, KK={Ku+}Kvd-, SK = {Ru,Rv}KK
BCY'3: V → U: {dataV}SK
BCY'4: U → V: {dataU}SK

The following input script to the verification system details the re-designed BCY protocol.

```
// Amended BCY Formal Protocol Specification
// Assumptions
A1 : U possess KuPriv;
A2 : U possess {U,KuPub}KsPriv;
A3 : U possess dataU;
A4 : U possess KsPub;
A5 : U possess Nu;
A6 : U believe fresh(Nu);
A7 : U believe recognised(dataV);
A8 : U believe recognised(V);
A9 : U believe hasKey(S,KsPub);
A10: U believe S hasJurisdiction S believe allStatement;
A11: U believe S hasJurisdiction hasKey(V,KvmPub),hasKey(V,KvdPub);
A12: U believe fresh(TSv);
A13: U believe shareKey(U,{Nu,Nv}{KvdPub}KuPriv,V);
A14: V possess KvdPriv;
A15: V possess KvmPriv;
A16: V possess {V,KvdPub,KvmPub}KsPriv;
```

```
A17: V possess dataV;
A18: V possess KsPub;
A19: V possess Nv;
A20: V believe fresh(Nv);
A21: V believe recognised(dataU);
A22: V believe recognised(U);
A23: V believe hasKey(S,KsPub);
A24: V believe S hasJurisdiction S believe allStatement;
A25: V believe S hasJurisdiction hasKey(U,KuPub);
A26: V believe fresh(TSu);
A27: V believe shareKey(V,{Nu,Nv}{KuPub}KvdPriv,U);

// Steps
S1 : U told notHere(Nv, notHere({(V,KvdPub,KvmPub,TSv),[S believe
                     hasKey(V,KvmPub),hasKey(V,KvdPub)]}KsPriv));
S2 : V told notHere(notHere({Nu}KvmPub), {Nv,{(U,KuPub,TSu),[S believe
                     hasKey(U,KuPub)]}KsPriv}Nu);
S3 : U told notHere({dataV}{Nu,Nv}{KvdPub}KuPriv);
S4 : V told notHere({dataU}{Nu,Nv}{KuPub}KvdPriv);

// Goals
G1 : U believe hasKey(V,KvmPub),hasKey(V,KvdPub);
G2 : U possess {Nu,Nv}{KvdPub}KuPriv;
G3 : U believe fresh({Nu,Nv}{KvdPub}KuPriv);
G4 : U believe shareKey(U,{Nu,Nv}{KvdPub}KuPriv,V);
G5 : U believe V conveyed {dataV}{Nu,Nv}{KvdPub}KuPriv;
G6 : U believe fresh({dataV}{Nu,Nv}{KvdPub}KuPriv);
G7 : V believe hasKey(U,KuPub);
G8 : V possess {Nu,Nv}{KuPub}KvdPriv;
G9 : V believe fresh({Nu,Nv}{KuPub}KvdPriv);
G10: V believe shareKey(V,{Nu,Nv}{KuPub}KvdPriv,U);
G11: V believe U conveyed {dataU}{Nu,Nv}{KuPub}KvdPriv;
G12: V believe fresh({dataU}{Nu,Nv}{KuPub}KvdPriv);
```

The following highlights the differences between the input script for the original BCY and the amended BCY protocols: U has the added initial believes in the freshness of TSv (A12) and that a Diffie-Hellman key exchange results in a suitable shared key (A13). For V the additional assumptions A19 and A20 state the possession and freshness of V's nonce, and assumptions A26 and A27 add V's initial believes in the freshness of TSu and that a Diffie-Hellman key exchange results in a suitable shared key. The Steps section incorporates the amendments as discussed above. The goals are the same for the amended BCY protocol than for the original BCY protocol. After completion of the verification process the following results are obtained for the BCY protocol:

```
(1) : U believe hasKey(V,KvmPub),hasKey(V,KvdPub)is True
(2) : U possess {Nu}{KvdPub}KuPriv is True
(3) : U believe fresh({Nu}{KvdPub}KuPriv) is True
(4) : U believe shareKey(U,{Nu}{KvdPub}KuPriv,V) is True
(5) : U believe V conveyed {dataV}{Nu}{KvdPub}KuPriv True
(6) : U believe fresh({dataV}{Nu}{KvdPub}KuPriv) is True
(7) : V believe hasKey(U,KuPub) is True
(8) : V possess {Nu}{KuPub}KvdPriv is True
(9) : V believe fresh({Nu}{KuPub}KvdPriv) is True
(10): V believe shareKey(V,{Nu}{KuPub}KvdPriv,U) is True
(11): V believe U conveyed {dataU}{Nu}{KuPub}KvdPriv is True
(12): V believe fresh({dataU}{Nu}{KuPub}KvdPriv) is True
```

This successful verification of the re-designed protocol provides strong evidence in the correctness of the protocol. Thus, the ability of the proposed development process to prevent the use of weak protocols by detecting protocol flaws is demonstrated.

Performance of automated logic-based verification
The automated verification system has been applied to a range of modern protocols, using different logics. A Pentium® 4 CPU 3GHz with 1GB of RAM running Microsoft Windows XP has been used for the empirical tests. The following gives a summary of some results:

- ASK Protocol (Aydos, Sunar & Koc, 1998): Verification using GNY logic reveals known weaknesses. Memory 34MB, time 0.8s
- CDFASK Protocol (Coffey, Dojen & Flanagan, 2003c): Verification using GNY logic succeeds. Memory 20MB, time 0.3s
- ASPeCT Protocol (Horn & Preneel, 2000): Verification using GNY logic reveals known weakness. Memory 26MB, time 1.4s
- Boyd-Park Protocol (Boyd & Park, 1998): Verification using CS logic reveals known weakness. Memory 77MB, time 1.3s
- Amended Boyd-Park (Newe & Coffey, 2003a): Verification using CS logic succeeds. Memory 41MB, time 0.8s
- HYS Protocol (Hwang, Yang & Shiu, 2003): Verification using CS logic reveals known weakness. Memory 33MB, time 0.3s
- 3GPP AKA Protocol (Zhang & Fang, 2005): Verification using CS logic reveals known weakness. Memory 20MB, time 0.2s
- AP AKA Protocol (Zhang & Fang, 2005): Verification using CS logic succeeds. Memory 46MB, time 0.6s.

These results show that verification logics can be efficiently automated. Further, the process of applying the logical postulates is performed in a manner of seconds. This demonstrates the verification system's ability to efficiently verify security protocols.

FUTURE TRENDS
Historically, the formal verification of security protocols has been approached in two fundamentally different ways, one based on the work of Dolev and Yao (1983) and another based on Turing machines.

The approach based on Dolev and Yao assumes perfect cryptography. This assumption states that the content of encrypted messages can only be extracted with the same key (symmetric encryption) or with the inverse key (public key encryption). However, perfect cryptography cannot always be guaranteed: weaknesses are often found in employed cryptographic systems and the continuous increase in computational power enables brute-force attacks that were infeasible only a few years before. Recent work tries to weaken or remove this assumption by adding equational theories (Comon-Lundh & Shmatikov, 2003; Abadi & Cortier, 2004) and modifying the Dolev-Yao model (Lowe, 2004; Baudet, 2005). In general, techniques based on the Dolev-Yao model are often efficiently automated.

In the Turing machine-based approach, no idealization of cryptographic systems is required: cryptographic functions only operate on strings and attackers are modelled as Turing machines. In this framework, correctness is defined in terms of a high complexity and negligible probability of success (Goldwasser & Micali, 1984; Bellare, Kilian & Rogaway, 1994). While this approach appears to be more realistic than the formal approach based on the Dolev-Yao model, its complexity makes it difficult to develop automated or semi-automated verification tools.

In recent years research has been undertaken to integrate these two approaches. The optimal goal is to prove the security of a protocol in the formal model and then to establish properties of the used cryptographic systems. Both results can then be combined to deduce the security of the protocol in a realistic framework without the perfect cryptography assumption. So far, very restrictive assumptions are required to keep the complexity of the realistic framework computational feasible An example of this approach is (Micciancio & Warinschi, 2004), where the authors prove that under certain conditions security in the formal model also implies security in a well defined more realistic model.

CONCLUSION

The protection of electronic networks and information systems against attack is a critical issue for the use of new technologies in all fields of life. Security protocols are designed to provide such protection. However, the design of correct and reliable security protocols is highly complex and error prone. Therefore, the importance of formal verification of cryptographic protocols during the development process cannot be overstated.

This chapter proposes a formal verification centred development process for security protocols, where design and verification of a security protocol are considered two inseparable steps: protocol design and formal verification are iterated until formal verification does not reveal any weaknesses of the protocol. Only then is the protocol published or implemented. This development process will significantly minimize the risk of using faulty protocols in electronic networks or information systems.

A case study demonstrated this formal verification centred development process using the BCY security protocol. The original BCY protocol was formally verified revealing a number of weaknesses. Subsequently, the protocol was redesigned to address these weaknesses and the amended protocol again subjected to formal verification. As the verification of the modified protocol succeeded, the amended protocol can be considered secure within the limits of its stated goals.

In conclusion, the discovery of various flaws in a surprisingly large number of published protocols, in many cases several years after the original publication, highlight the importance of formal verification and the strength of the proposed development process.

REFERENCES:

Abadi, M., & Cortier, V. (2004). Deciding knowledge in security protocols under equational theories. In: 31th International Colloquium on Automata, Languages and Programming ICALP'04 (pp. 46-58), Turku, Finnland.

Aydos, M., Sunar, B., & Koc, C.K. (1998). An elliptic curve cryptography based authentication and key agreement protocol for wireless communication. In: Proceedings of 2nd International Workshop on Discrete Algorithms and Methods for Mobility DIAL M 98 (pp.1-12), Dallas, USA

Baudet, M. (2005). Deciding security of protocols against off-line guessing attacks. In: Proceedings of the 12th ACM Conference on Computer and Communications Security CCS05 (pp.16-25), Alexandria, VA, USA.

Bellare, M., Kilian, J., & Rogaway, P. (1994). The security of cipher block chaining. In Desmedt, Y.G. (ed.), Proceedings of CRYPTO 94 (pp. 341–358), Santa Barbara, California, USA.

Boyd, C., & Park, D.G. (1998). Public Key Protocols for Wireless Communications. In: Proceedings of International Conference on Information Security and Cryptology ICISC '98 (pp.47-57), Seoul, Korea.

Brackin, S. (2000). Automatically Detecting Most Vulnerabilities in Cryptographic Protocols. In: Proceedings of DARPA Information Survivability Conference & Exposition, Volume I (pp. 222-236), IEEE CS Press.

Burrows, M., Abadi, M., & Needham, R. (1990). A logic of authentication. *ACM Transactions on Computer Systems*, 8(1), 18-36.

Carlsen, U. (1994). Optimal privacy and authentication on a portable communications system. *ACM Operating Systems Review*, 28(3), 16-23.

Cervesato, I., Durgin, N., Lincoln, P., Mitchell, J., & Scedrov, A. (1999). A meta-notation for protocol analysis. In: Proceedings of IEEE Computer Security Foundations Workshop (pp. 55-69), Mordano, Italy.

Chang, Y.F., & Chang, C.C. (2005). An efficient authentication protocol for mobile satellite communication systems. *ACM SIGOPS Operating Systems Review*, 39(1), 70 – 84.

Clarke E.M., Jha S., & Marrero W. (2000). Verifying security protocols with Brutus. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 9(4), 443-487.

Coffey T., & Dojen R. (2003) Analysis of a mobile communication security protocol. In: Proceedings of International Symposium on Information and Communication Technologies ISICT03 (pp.329-335), Dublin, Ireland

Coffey, T., Dojen, R., & Flanagan, T. (2003a). Formal Verification: An Imperative Step in the Design of Security Protocols. *Computer Networks Journal, Elsevier Science*, 43 (5), 601-618.

Coffey, T., Dojen, R., & Flanagan, T. (2003b). On the Automated Implementation of Modal Logics used to Verify Security Protocols. In: Proceedings of the International Symposium on Information and Communication Technologies (pp.324-347), Dublin, Ireland.

Coffey, T., Dojen, R., & Flanagan, T. (2003c) On Different Approaches to Establish the Security of Cryptographic Protocols. In: Proceedings of Conference on Security and Management SAM'03 (pp.637-643), Vol. II, Las Vegas, USA.

Coffey, T., & Saidha, P. (1997). A Logic for Verifying Public Key Cryptographic Protocols. *IEE Journal Proceedings on Computers and Digital Techniques*, 144(1), 1997, 28-32

Coffey T., Saidha, P., & Burrows P. (2003). Analysing the security of a non-repudiation communication protocol with mandatory proof of receipt. In: Proceedings of International Symposium on Information and Communication Technologies ISICT03 (pp. 351-356), Dublin, Ireland.

Cohen, E. (2000). TAPS: a first-order verifier for cryptographic protocols. In: Proceedings of IEEE Computer Security Foundations Workshop (pp. 144-158), Cambridge, UK.

Comon-Lundh, H., & Shmatikov, V. (2003). Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In: Proc. of the 18th Annual IEEE Symposium on Logic in Computer Science LICS'03 (pp. 271-280). IEEE Computer Society, Ottawa, Canada).

Diffie, W., & Hellman, M.E. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6), 644-654

Dill D.L. (1996). The Murφ verification system. In: Proceedings of International Conference of Computer Aided Verification (pp. 390-393), New Brunswick, NJ, USA.

Dojen, R ., & Coffey, T. (2005). The Concept of Layered Proving Trees and Its Application to the Automation of Security Protocol Verification. *ACM Transactions on Information and System Security (TISSEC)*, 8(3), 287 - 31.

Dolev, D., Even, S., & Karp, R. (1982). On the Security of Ping-Pong Protocols. *Information and Control*, 55(1), 57 - 68.

Dolev, D., & Yao, A. (1983). On the Security of Public Key Protocols. *IEEE Transactions on Information Theory*, 29(2), 198 - 208.

Even, S., & Goldreich, O. (1983). On the security of multi-party ping-pong protocols. In: Proceedings of IEEE Symposium on the Foundations of Computer Science (34-39), Tucson, AZ, USA

Goldwasser, S., & Micali, S. (1984). Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2), 270–299.

Gong, L., Needham, R., & Yahalom, R. (1990). Reasoning about belief in Cryptographic Protocols. In: Proceedings of IEEE Computer Society Synopsis on Research in Security and Privacy (pp. 234-248), Oakland, USA.

Gürgens, S., & Rudolph, C. (2002). Security Analysis of (Un-) Fair Non-repudiation Protocols. In: A. E. Abdallah, P. Ryan, S. Schneider (Eds.): Formal Aspects of Security, First International Conference, FASec 2002 (pp.07-114), London, UK .

Horn, G., Martin, K., & Mitchell, C. (2002). Authentication Protocols for Mobile Network Environment Value-added Services. *IEEE Transactions on Vehicular Technology*, 51(2), 383-392.

Horn, G., & Preneel, H. (2000). Authentication and Payment in Future Mobile Systems. *Journal of Computer Security* 8(2/3), 183-207.

Huima, A. (1999). Efficient infinite-state analysis of security protocols. In: Proceedings of Workshop on Formal Methods and Security Protocols FLOC'99, Trento, Italy.

Hwang, M.S., Yang, C.C., & Shiu, C.Y. (2003). An authentication scheme for mobile satellite communication systems. *ACM SIGOPS Operating Systems Review*, 37(4), 42 - 47

Kessler, V., & Wendel, G. (1994). AUTLOG – An advanced logic of authentication. In: Proceedings of IEEE Computer Security Foundations (pp. 90-99), Menlo Park, California, USA.

Kramer, S. (2004). CPL: An Evidence-Based 5-Dimensional Logic for the Compositional Specification and Verification of Cryptographic Protocols - Part I: Language, Process Model, Satisfaction. In: Proceedings of the LICS/ICALPA affiliated Workshop on Foundations of Computer Security FCS04 (pp. 77-96), Turku, Finland.

Kudo, M., & Mathuria, A. (1999). An Extended Logic for Analyzing Timed-Release Public-Key Protocols, In: Varadharajan, V., & Mu, Y. (Eds.), *Lecture Notes In Computer Science Vol. 1726, Proceedings of the Second International Conference on Information and Communication Security* (pp. 183 – 198), Springer Verlag

Lowe, G. (1995). An attack on the Needham-Schroeder Public-Key authentication protocol. *Information Processing Letters*, 56(3), 131-133.

Lowe, G. (2004). Analysing Protocol Subject to Guessing Attacks. *Journal of Computer Security*, 12(1), 83-98

Meadows C. (1996). The NRL Protocol Analyser: an overview. *Journal of Logic Programming*, 26(8), 113-131.

Micciancio, D., & Warinschi, B. (2004). Soundness of formal encryption in the presence of active adversaries. In: Proceedings of the Theory of Cryptography Conference (pp. 133–151), Cambridge, MA, USA.

Millen, J.K., Clark, S.C., & Freedman, S.B. (1987). The Interrogator: protocol security analysis. *IEEE Transactions on Software Engineering*, 13(22), 274-288.

Mu, Y., & Varadharajan, V. (1996). On the design of security protocols for mobile communications. In: Proceedings of Australasian Conference on Information Security and Privacy (134-145), Wollongong, Australia.

Needham, R.M., & Schroeder, M.D. (1978). Using Encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12), 993-999.

Newe, T., & Coffey, T. (2002). Hybrid Mobile Security Protocol: Formal Verification using a New Modal Logic. In: Proceedings of Int. Conf. on Automation and information ICAI-2002, Puerto De La Cruz, Tenerife, Spain.

Newe T., & Coffey, T. (2003a). Formal Verification logic for hybrid security protocols. *International Journal of Computer Systems Science & Engineering*, 18(1), 17-25.

Newe, T., & Coffey, T. (2003b). Minimum-Knowledge Schemes for low-power, low-memory Devices. *WSEAS Transactions on Circuits and Systems*, 2(2), 460-465.

Newe, T., & Coffey, T. (2003c). On the Logical Verification of Key Exchange Protocols for Mobile Communications. In: Mastorakis N.E., Antoniou, G.E., Manikopoulos, C., Bojkovic, Z, Gonos, I.F. (Eds.) *Recent Advances in Communications and Computer Science* (pp. 76-81), WSEAS Press.

Paulson, L.C. (1998). The Inductive Approach to Verifying Cryptographic Protocols, *Journal of Computer Security*, 6(1-2), 85-128.

Rabin, M.O. (1979). *Digitalized signatures and public-key functions as intractable as factorisation*. (Tech. Rep. MIT/LCS/TR-212), Massachusetts Institute of Technology MIT, Cambridge, MA, USA

Schneider, S.A. (1998). Verifying Authentication Protocols in CSP. *IEEE Transactions on Software Engineering (TSE)*, 24(9), 741-758.

Song, D., Berezin, S., & Perrig, A. (2001). Athena: a novel approach to efficient automatic security protocol analysis. *Journal of Computer Security*, 9(1), 47–74.

Stubblebine, S., & Wright, R. (2002). An Authentication Logic with Formal Semantics Supporting Synchronization, Revocation, and Recency. *IEEE Transactions on Software Engineering*, 28(3), 256-285.

Syverson, P.F., & van Oorschot, P.C. (1994). On unifying some cryptographic protocols logics. In: Proceedings of IEEE Symposium on Security and Privacy (pp. 14-28), Oakland, CA, USA.

Ventuneac, M., Coffey, T., & Newe, T. (2004). Reasoning on Properties of Non-Repudiation Security Protocols. *WSEAS Transactions on Information Science and Applications* 1(5), 1262-1267.

Ventuneac, M., Dojen, R., & Coffey, T. (2006). Automated Verification of Wireless Security Protocols using Layered Proving Trees. *WSEAS Transactions on Communications*, 5(2), 252-258.

Zhang, M., & Fang, Y. (2005). Security analysis and enhancements of 3GPP authentication and key agreement protocol. *IEEE Transactions on Wireless Communications*, 4(2), 734 – 742

Zhang, Y., & Varadharajan, V. (2001). A logic for modelling the dynamics of beliefs in cryptographic protocols. In: Proceedings of Australasian Computer Science Conference (pp. 215-222), Gold Coast, Queensland, Australia.

Key Terms and Their Definition

- **Authentication:** The verification of the identity of the source of information
- **Confidentiality**: The protection of information so that someone not authorized to access the information cannot read the information even though the unauthorized person might see the information's container
- **Cryptography**: Cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication and data origin authentication
- **Data Integrity**: Data Integrity is the provision of the property that data and data sequences have not been altered or destroyed in an unauthorized manner
- **Formal Verification**: Application of formal methods to establish the correctness of a system.
- **Formalisation of a Security Protocol**: Specifying the protocol in the language of the logic by expressing each protocol message as a logical formula is known as protocol formalisation (or idealisation). A formal description of the protocol, obtained by formalisation, does not simply list the components of each message but attempts to show the purpose of these components so as to avoid ambiguity.
- **Nonce**: A large random number that is intended to be employed only once and is used for binding messages to a specific protocol run.
- **Non-Repudiation**: Non-repudiation is the concept of ensuring that a contract, especially one agreed to via the Internet, cannot later be denied by one of the parties involved. With regards to digital security, non-repudiation means that it can be verified that the sender and the recipient were, in fact, the parties who claimed to send or receive the message, respectively.
- **Postulate**: A proposition that is accepted as true in order to provide a basis for logical reasoning.
- **Security Protocol:** (also called cryptographic protocol) constitutes transferring specially constructed encrypted messages between legitimate protocol participants to fulfil objectives such as mutual authentication or key-exchange in a predefined procedure