Cryptographic Protocols for Third Generation Mobile Communication Systems

DongGook Park, BElecEng, MSc

Thesis submitted for the degree of

Doctor of Philosophy

21 July, 2001

Information Security Research Centre
School of Data Communications
Queensland University of Technology
Brisbane, Australia

Certificate Recommending Acceptance

Key Words

Authentication, key establishment, authentication model, formal analysis, forward secrecy, key recovery, key escrow, WAKE protocol, denial-of-service attack, electronic commerce, mobile security, mobile communication security, UMTS, IMT-2000, IS-41C, GSM.

Abstract

Design of public-key security protocols for wireless mobile communications requires considerable effort to satisfy two goals. The security protocols need to be (1) flexible and comprehensive to accommodate potentially complex public key infrastructure requirements due to roaming; and (2) computationally effective for mobile terminals but nonetheless fully functional for future mobile communications security requirements. A new suite of generic protocols meeting the first goal, and also a set of instance protocols which are compliant with the generic versions and satisfy the second goal have been designed.

To systematically design and analyze the required protocol, two new methodologies are devised: (1) classification of authentication protocols, and (2) a new model of authentication and key establishment.

Classification work was motivated by the design requirement to have a sufficiently general or generic protocol which can be made into several specific and concrete protocols to address a diverse range of security requirements. In this way, we can design a good protocol with regard to strong security, and equip the corresponding concrete protocol with several add-on cryptographic facilities like forward secrecy and denial-of-service attack proof mechanism. Also, we can focus our security analysis only upon the generic protocol, not upon many possible concrete instance protocols. It is demonstrated that this provides a very convenient and systematic way of protocol design.

A novel model of authentication and key establishment is developed to analyse the security of the cryptographic protocols because existing methods like BAN logic and SVO logic turned out to be somewhat unsatisfactory with regard to their model and protocol goal structures. Most of all, entirely new modelling of entity authentication and key authentication is achieved, which in turn lead to a new methodology developed for protocol analysis. This analysis method is applied to analyse the generic protocol designed proving it secure.

Table of Contents

List of Figures

List of Tables

Statement of Original Authorship

The work contained in this thesis has not been previously submitted for a degree or diploma at any other higher education institution. To the best of my knowledge and belief, this thesis contains no material previously published or written by another person except where due reference is made.

***Signed*:** DongGook Park

***Date*:** Thursday, July 18, 2001

Acknowledgments

I would like to thank my supervisor Ed Dawson for his guidance and advice, and most of all, his felicitous direction that I should research on authentication and key establishment protocols, before that I was obsessed with cryptographic algorithm study. It was truly correct decision for my research and has led me to achieve the result in this thesis.

I would also like to offer my heartfelt thanks to my associate supervisor Colin Boyd. He consistently and kindly guided me with a mysterious "Tao" which is very similar to that of Laotzu in that he did not draw my rein but instead brought me up to develop my ideas with passion and inspiration.

I would like to thank Juanma González Nieto, Kapali Viswanathan and Andrew Clark for their encouragements and friendship. Working together with them for three years will become a good old memory in my life. Thanks to my two roommates, Bill Millan and Jason Reid. Bill was my tutor in the early stage of my PhD course and has always given me energetic advice. Jason, with an ever bright smile on his face, encouraged and helped me when I was struggling with my stressful last year in a foreign country. Thanks to other people in ISRC for their kind help. They include Christine Orme, Betty Hansford and Gary Gaskell.

I would like to express my sincere gratitude to my bosses in Korea Telecom including Dr. JungJoon Kim, Dr. TaeGeun Kim and Mr. MyungSang Yoon for their steady support and encouragement throughout my PhD research. I would also like to thank Prof. Sang-Jae Moon in Kyungpook National University in Korea for his kind guidance and encouragement. Especially, discussion about instance protocols with him was invaluable to the instance protocol design work.

Final thanks go to my wife and my two daughters for their understanding and encouragement which helped me through many tough times.

This research is a part of the collaborative project for wireless security between ISRC QUT and my company Korea Telecom. Hence, almost all the objectives of my

research is in the same line of the collaborative project which aims to achieve a whole framework of future wireless mobile security including security architecture, security protocols, cryptographic algorithms and public key infrastructure (PKI).

Abbreviations

AC	Authentication Centre
APriKey	*A*'s Private Key
APubKey	*A*'s Public Key
APriKeyX	Short term uncertified *A*'s Private Key
APubKeyX	Short term uncertified *A*'s Public Key
ASPeCT	Advanced Security for Personal Communications Technologies
AUTHR	Authentication Response
CAVE	Cellular Authentication and Voice Encryption
DoS	Denial of Service
ESN	Electronic Serial Number
ETSI	European Telecommunications Standards Institute
FPLMTS	Future Public Land Mobile Telecommunications System
FS	Forward Secrecy
GSM	Global System for Mobile Communications
ITU	International Telecommunication Union
MIN	Mobile Identification Number
MS	Mobile Station
NFS	No Forward Secrecy
NO	Network Operator
RAND	Random Variable
RANDSSD	Random Number for SSD Update
SMEKEY	Signaling and Message Encryption Key
SP	Service Provider
SSD	Shared Secret Data
SSD-A	Authentication portion of SSD
SSD-B	Voice Privacy and Signaling Message Encryption portion of SSD
SSL	Secure Socket Layer
TLS	Transport Layer Security
TTP	Trusted Third Party
UMTS	Universal Mobile Telecommunications System
VASP	Value Added Service Provider

VLR	Visitor Location Register
VPMASK	Voice Privacy Mask
WAKE	Wireless Authentication and Key Establishment

Chapter 1.

Introduction

Voice communication through the wireless mobile service has become a worldwide trend and in the near future more than half of the voice traffic will be carried by mobile wireless channels in most developed and developing countries. Public mobile telecommunication services including radio paging, cordless, private mobile radio and cellular have been under the first evolution step from the first generation systems based on *analog* radio technologies to the second generation system which exploits full benefits of *digital* technologies which promise better service quality and more diverse services for service users [Calh88, GaHa93]. Furthermore, in the next step from the second to third generation systems, all these mobile systems and services will be integrated into universal mobile communication systems which will support multiple radio interface technologies such as TDMA (Time Division Multiple Access) and CDMA (Code Division Multiple Access) [Rapp96], and most of the current independent mobile services from paging to wireless mobile multimedia services [GrMa90].

The two keywords, "wireless" and "mobile" explain why the security problem has been critical for mobile communications. There is no fixed physical channel associated with each user and every user moves. These characteristics cause great difficulty for service providers to manage security services. In addition, future wireless systems are expected to provide more than just simple voice communications and very limited data services of the current generation wireless mobile services [ACTS97]. This means most of the security services used in computer communications will be required in the wireless mobile services as well.

Moreover, mobile communication requires special treatment when designing and implementing security technologies. Mobile terminals are on a ceaseless evolutionary path for light weight and small size, which causes a great limitation to their security technology design. Considerably low computing power and limited battery life is a constraining factor for the design of cryptographic algorithms and protocols. Nevertheless, roaming characteristics of mobile terminals also call for the more sophisticated cryptographic technology such as "asymmetric key cryptography (public key cryptography)". This means that users hold key pairs; the public key is available to all entities, while the private key is known only by the user. In contrast with shared key cryptography, used in the current generation wireless architectures, asymmetric solutions have two major advantages.

- There is no requirement for an on-line server to mediate every access to the network. This greatly simplifies the management of cryptographic keys.
- Digital signatures may be implemented. This provides non-repudiation services which will be required for wireless electronic commerce applications as well as providing enhanced services such as secure billing.

These advantages, however, always come at some cost: much more heavy cryptographic computation within both the user terminal and the network system. It may be too high to be justified in the mobile situation where the available resource for security, in terms of computation and bandwidth, is limited. This is why we need to tailor the cryptographic mechanisms such as security protocols for mobile applications.

Wireless authentication and key establishment (WAKE) protocols may be viewed as vehicles upon which most security tools such as electronic commerce mechanisms, end-to-end protocols and clone detection mechanisms can be mounted as add-on features. The WAKE protocols may in turn be divided into two sub-layers: *generic* and *instance* protocols. Generic protocols are such protocols whose description is not dependent on any underlying cryptographic primitives like discrete logarithm and RSA, while instance protocols are described using appropriate mathematical notations according to the relevant cryptographic primitives. Different instance protocols are required to address variant sets of security goals with regard to some optional mechanisms for forward secrecy, denial-of-service attack, etc.

1.1 Research Goals

With the above aspects in mind, the research on security protocols has been executed systematically under the following objectives and their conceptual relations are depicted in Figure 1-1.

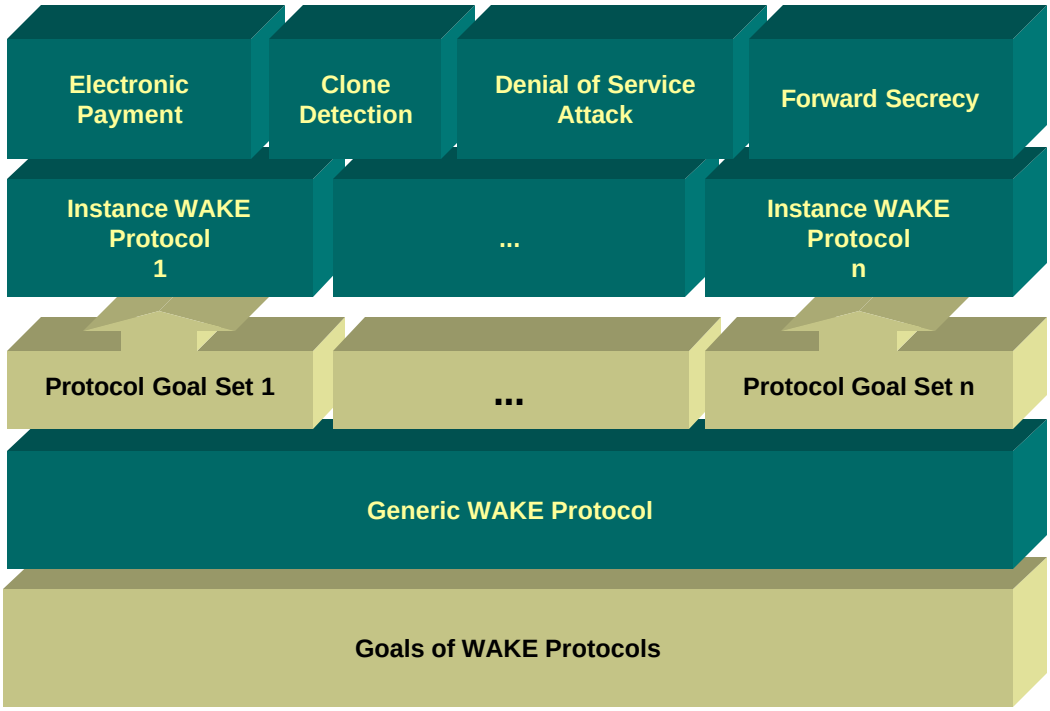


Figure 1-1: Framework of research goals

- **Investigation and identification of requirements or goals of WAKE protocols.** This work is a stepping stone to design of authentication and key establishment protocols. The structure and implementation cost of the WAKE protocols is determined depending upon which goals are aimed for the protocols.
- **Design and analysis of the *generic* WAKE protocol.** Once the goals are identified, we can determine the structure of the WAKE protocols in terms of public key/private key operations, encryption, message authentication codes (MACs) and the number of message exchanges, which result in a desirable generic protocol. Authentication and key establishment protocol design is notorious with all the difficulties of building a protocol, finding some flaw in it and fixing it. To avoid such a waste of effort, designing and analysing in the level of generic

protocols, and achieving a proper generic protocol are critical steps before we attempt to derive any specific concrete or instance protocols.

- **Design and analysis of the *instance* WAKE protocols.** The same generic protocol may be developed into different instance protocols to satisfy different sets of goals depending on different application environments such as user-network interface and user-value added service provider (VASP) interface. Computational efficiency is another key point when we design instance protocols. It is aimed to analyse the derived instance protocols with regard to complexity.
- **Investigation and design of add-on mechanisms for WAKE protocols.** Some cryptographic services such as electronic payments, clone detection, countermeasures against denial-of-service attacks, and forward secrecy are investigated and designed as add-on features for the WAKE protocols.

1.2 Goals of Third Generation WAKE Protocols

Current second-generation digital mobile systems, including Global Systems for Mobile Communications (GSM) [Redl98] and the North American Digital Cellular based on IS-41 [TIA95], provide security services pertinent to their system and service features [Garg96]. Future third generation wireless systems are required to have stronger and more diverse security features according to new service features (see Figure 1-2).

The main purpose of current generation wireless systems is to provide a wireless mobile equivalent service of the *plain old telephone services* (POTS), even though they support very limited data services. The corresponding security features of current systems belong to the two categories of authentication and confidentiality. User anonymity features and the encryption of user data, for instance, are two of the confidentiality requirements.

Future systems, on the other hand, shall have new security features in addition to those of current systems. Data integrity to guarantee that the received data has not been modified during transmission shall be provided because much more traffic of future systems will be due to data exchanges. More wide spread wireless terminals will make future wireless services one of the convenient methods for electronic commerce. Therefore, digital signature (authenticity) services and non-repudiation services will

also be important components of the future systems' security features. It can be seen that the increase in the number of application services of telecommunications will be more accelerated in future telecommunications services including wireless systems. The diversity in application services means diversity in user profiles to be managed by the network and modified by the user, which is directly related to the access to network resources and billing. This point makes access control an indispensable constituent of the security features of future wireless systems.

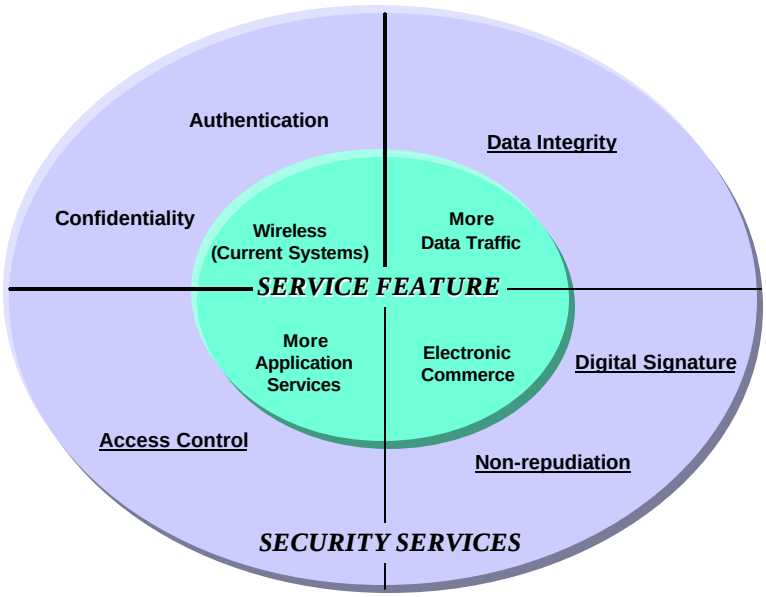


Figure 1-2: Security services expected in future mobile communication systems

All of these possible security services at least start from a sound secure execution of a *wireless authentication and key establishment* (*WAKE*) *protocol.* The third generation mobile communications systems such as IMT2000 of the ITU [Buha97, OjPr98] and Universal Mobile Telecommunications System (UMTS) of the ETSI [Blac99] have considered various security protocols for their WAKE purposes. Adopting asymmetric cryptography for WAKE protocols seems to be the most promising approach to solve challenging issues such as security with roaming over different network/service operators and the non-repudiation features for electronic commerce.

It seems that any future WAKE protocols based on conventional symmetric cryptography would either be enhancements of certain current generation mobile

communications systems such as GSM and the North American digital cellular systems, or quite a new protocol such as proposed for UMTS [Blac99]. Those based on asymmetric cryptography considered in UMTS and IMT2000 are using, by and large, variants of the Station-to-Station (STS) protocol [DvOW93] which seems to satisfy more security goals than are required for mobile security. Therefore, most effort has so far been focused on the design of more computationally efficient variants of the STS protocol bearing computational and bandwidth limitations in mind. The protocol description of the STS protocol appears in Section 3.2.2.

The WAKE protocols would be applied mainly to two different types of interfaces: the user to the network operator (NO), and the user to value added service provider (VASP). The former is an air interface, and the latter is a kind of end-to-end connection where the mobile user is one end and the VASP the other. Clearly, both may be rather different from each other in several aspects including security requirements. These different applications may justify a set of different protocols in the same network or at least multiple levels of security for sufficient flexibility to cater for different requirements of the two interfaces.

The authentication and key establishment protocol is the very starting point from which most security services such as data confidentiality and non-repudiation are able to be provided. Thus, clear and comprehensive identification of requirements for WAKE protocols, or in other words, the goals of WAKE protocols is an essential first step in WAKE protocol design and analysis.

The goals of a WAKE protocol for future mobile communications have been identified in the literature. Although different authors give slightly different descriptions the following items are widely considered to be essential [HMM99, ASPe96]:

- mutual authentication of user and network,
- secure and authentic establishment of a new session key,
- confidentiality of relevant data,
- non-repudiation of origin for relevant user data.

Some other goals not listed explicitly above, but described in the literature [HoPr98, ISO96, ASPe96], may include user anonymity, key freshness, key authentication and key confirmation. These additional goals, however, may be considered as either

outcomes of the complete successful achievement of the above stipulated goals, or arguably ambiguous and questionable goals. The latter point about protocol goals is examined in more detail in Section 2.2, where we build up an entirely new model of authentication and key establishment.

In addition to the above essential goals, we can consider several additional goals which enrich WAKE protocols to be flexibly applied to different requirements of different pairs of WAKE protocol principals such as user-network operator and user-value added service provider. The following features are identified and set as supplementary goals of WAKE protocols:

- forward secrecy,
- robustness against denial-of-service (DoS) attacks,
- support for electronic payments,
- clone detection.

A protocol is said to provide *forward secrecy* if the compromise of long-term keys does not compromise past session keys that have been established before the compromise of the long-term key [MvOV97]. Forward secrecy does not seem to have been considered in either IMT2000 or UMTS proposals. This is a surprise when considering that electronic commerce is widely discussed in the context of future mobile communication services. The reason is probably the additional computational cost associated with forward secrecy for both the user terminal (or smart card) and the network. Forward secrecy, so far as is known, can be achieved only through the use of asymmetric key cryptography, for example, key agreement using Diffie-Hellman key agreement [DiHe76] or any other public key schemes. However, public key based WAKE protocols have been extensively considered in both IMT2000 and UMTS, and hence, it is very reasonable to add forward secrecy to the desirable goals of WAKE protocol for future mobile systems.

Recently DoS attacks are of growing concern as the Internet has been widely introduced. For the research, a typical type of DoS attacks, *connection depletion attack* is focused on, in which an attacker tries to exhaust a server system's session resources by issuing a huge amount of bogus connection requests at the same time. Usually cryptographic security protocols are not able to defeat the DoS attack, but rather

increase the vulnerability to the attack because of the heavy computation associated with cryptographic operation. Nevertheless, many Internet security protocols including SSL/TLS protocol do not consider this aspect. This overlooked issue in authentication protocol design is addressed in the thesis, and robustness of WAKE protocols in this regard is established as an additional goal.

The electronic payment protocol is a separate issue on its own. Authentication and key establishment are crucial requirements for secure and authentic electronic payment. Considering the limitation of mobile terminals in terms of storage capacity, the WAKE protocols designed in this research may be required to be re-used for electronic payment applications. Combined with a suitable payment scheme, the WAKE protocols may provide a way of establishing entity authentication and secure establishment of the session key for electronic payment purposes. Thus, enabling the WAKE protocols to be designed in the research to be able to support electronic payments is also set as an additional goal of the WAKE protocols.

Authentication is not a panacea for identity cheating fraud. The secret key data as well as identity information of an authentication user terminal may somehow be copied into the attacker's facility enabling the attacker to impersonate an authentic victim user to the network. Detecting that cloning of a terminal has happened seems to be possible and this is set as a supplementary goal in the research.

It is never intended that some or all of these supplementary goals should be incorporated into a common WAKE protocol. Instead, it will be strategically desirable that we have a *set* of several instance protocols available, all of which comply with the same generic protocol, to provide a multiple level of security to mobile communications. The choice of a particular protocol out of the set will be dependent on the security requirements of the different interfaces or operational situations. This might be termed a *plug-in approach* towards protocol usage: a common WAKE protocol can be supplemented by plugging in several add-on features. In fact, several instance protocols of the same generic protocol are proposed to provide forward secrecy (see Section 3.2), and specific schemes are investigated and designed for other supplementary goals.

1.3 Summary of Major Accomplishments

The major accomplishments of research were in three related areas: new methods for design and analysis of authentication and key establishment protocols, design and analysis of a generic WAKE protocol, and design of add-on protocols.

1.3.1 New Methods for Design/Analysis of Authentication and Key Establishment Protocols

The first goal of this research, study of goals for WAKE protocols, led to the search for a new way of describing authentication and key establishment protocols, especially when it comes to several important definitions in relation to the protocol goals. Therefore, two novel methods for authentication and key establishment protocols have been built though they were not explicitly intended at first.

- **Classification of authentication protocols**
  Classification is one of the basic methodologies in most disciplines including cryptography. It is surprising that we still do not have any well-established classification scheme for authentication protocols. In this thesis, a new and possibly the first such scheme was proposed. This classification method allows for systematic reuse of previous analysis and design experience. Furthermore, the effort to establish a criterion to classify different protocols has led to a keen awareness of *cryptographic particles*[1] comprising authentication/key-establishment protocols, which in turn has developed into construction of a new model for authentication and key establishment.
- **Modelling and analysis of authentication and key establishment**
  A more concrete approach has been taken for analysis of authentication protocols based on cryptographic particles, which resulted in a new model of authentication and key establishment. A relevant syntax to describe authentication and key establishment was also devised. Sufficiently general and comprehensive, this model reduces the protocol analysis procedure down to mathematical proof-like

[1] This term is first introduced in the thesis for the classification work. Cryptographic particles in an authentication protocols are the most elementary data for authentication purposes. Typically, they are identities, random nonces and secret keys. Their definitions and descriptions are presented in Section 2.1.2.

steps. Achieving a model which represents things as they really are, we do not have to resort to sophisticated methodologies or concepts. The new model of authentication and key establishment has naturally led to a different interpretation of several goals of protocols, especially in relation to "key authentication". Consequently, a new hierarchy of authentication and key establishment goals has been proposed.

1.3.2 Design and Analysis of a Generic WAKE Protocol and its Instance Protocols

The classification work and its resultant method has identified a new generic protocol which seemed to have a proper structure to be implemented into more concrete instance protocols which are required to be efficient for mobile communication environments. The generic protocol belongs to a different category than the STS protocol according to the classification criteria in this thesis. It was enhanced and developed into several variant forms to satisfy the requirement of mobile communications, such as user anonymity and different conditions in relation to public-key availability at protocol principals. The new method in this thesis has been applied to analyse the generic protocol. Several instance protocols have also been designed from the same generic protocol, all of which are described in discrete logarithm based algorithm, and therefore can easily be translated to elliptic curve equivalents. A set of multiple instance protocols are intended to serve different security requirements especially with regard to forward secrecy which usually requires additional public-key related computations.

1.3.3 Add-on Mechanisms

Several add-on cryptographic features like forward secrecy, countermeasure against denial-of-service attack, electronic payment features and clone-detection mechanisms have been investigated and designed to be integrated into the WAKE protocol.

- **Forward secrecy**

  Two general prototypes for forward secrecy are surveyed and identified: the first one depending on a particular property of key agreement functions and the second one exploiting confidentiality by temporary asymmetric key pairs. The latter one

was previously identified but its significance not understood. Its true value is disclosed and several examples of implementation are presented. Both prototypes are applied to the WAKE protocol which resulted in three different instance protocols.

- **Countermeasure against denial-of-service (DoS) attacks**
  Cryptographic protocols are introduced into systems to secure them, but ironically this may add to target points of attackers. For bogus connection requests issued by attackers in a large amount at the same time may cause the system to be bogged down with heavy public-key related computation. A virtually zero cost solution is proposed in both level of generic and instance protocols. This basic idea comes from a new usage of random number other than challenge value, which may be called a sort of "cryptographic salt".
- **Electronic payments**
  Considering the use of the WAKE protocol for end-to-end transaction security on the application layer as well as the air interface between the user and the network, electronic payment was investigated in the research. A few micropayment schemes are surveyed and a new scheme is devised. Finally, the generic WAKE protocol is used to support end-to-end transaction security with hash chain scheme adopted as its micropayment engine.
- **Clone detection**
  However sophisticated the future wireless security mechanisms are, they cannot be panaceas for all the potential frauds, amongst which especially the mobile-terminal cloning fraud is out of the scope covered by user authentication mechanism. Almost all of the solutions to cope with this kind of fraud require the use of the mechanisms which are separate from the original built-in cryptographic system. It would surely be a burden in terms of investment cost and operation/maintenance. With this in mind, an alternative solution is proposed in this research purely based on a cryptographic technique, which can therefore be smoothly integrated into any built-in cryptographic mechanisms of mobile communication systems.

1.4 Organisation of Thesis

Centring around design and analysis of cryptographic protocols for third generation mobile communication systems, the material in the thesis is composed of four main parts.

- **Introduction to the research (Chapter 1)**
  Chapter one describes goals, methodologies and results of the research, and investigates the goals of the WAKE protocol.
- **Systematic methodologies for design/analysis of authentication and key establishment protocols (Chapters 2)**
  This chapter contains description about two new methods for design/analysis of authentication and key establishment protocols: 1) classification of authentication protocols, and 2) modelling of entity authentication and key authentication.
- **Cryptographic protocols for third generation mobile communication systems (Chapter 3)**
  With two new methods put forward in Chapter 2, a new generic WAKE protocol is designed and analysed in Chapter 3. A new concrete instance protocol is also derived from the generic protocol.
- **Add-on mechanisms (Chapter4)**
  This chapter describes several add-on mechanisms for the WAKE protocol, including forward secrecy, countermeasure against denial-of-service attacks, clone detection, and electronic payment.

1.5 Published Results

Many of the findings contained within the chapters that follow have appeared at conference proceedings in some form. Details of the publication and the chapter they pertain to are cited here.

Chapter 2

- D. Park, C. Boyd and E. Dawson, "Classification of authentication protocols: a practical approach", *ISW 2000, LNCS 1975*, Springer-Verlag, 2000.

Chapter 3

- C. Boyd and D. Park, "Public key protocols for wireless communications", *ICISC*

'98, Korea, December 1998, pp.47-57.

Chapter 4

- D. Park, C. Boyd and S.-J. Moon, "Forward secrecy and its application to future mobile communications security", *PKC 2000, LNCS 1751*, Springer-Verlag, 2000, pp.433-445.
- D. Park, C. Boyd and E. Dawson, "Micropayments for wireless communications", *ICISC 2000, LNCS 2015*, Springer-Verlag, 2000, pp.192-205.
- D. Park, M. Oh, and M. Looi, "A fraud detection method using is-41c protocols and its application to the third generation wireless systems", *GLOBECOM '98*, Australia, November, 1998, pp.1984-1989.
- D. Park, J. Kim, C. Boyd, and E. Dawson, "Cryptographic salt: a countermeasure against denial-of-service attacks", To appear in *ACISP 2001, LNCS 2119*, Springer-Verlag, 2001.

Chapter 2.

Methodologies of Design and Analysis of Authentication/Key Establishment Protocol

To systematically design and analyze the required WAKE protocol, two new methodologies are established:

- classification of authentication protocols, and
- formal model of authentication and key establishment.

Classification work was motivated by our design requirement to have a sufficiently general or generic protocol which can be made into several specific and concrete protocols to address a diverse range of security requirements. In this way, we can design a good protocol with regard to strong security, and equip the corresponding concrete protocol with several add-on cryptographic facilities like forward secrecy and denial-of-service attack proof mechanism. Also, we can focus our security analysis only upon the generic protocol, not upon many possible concrete instance protocols, which is a very convenient and systematic way of protocol design.

We devised a new model of authentication and key establishment to analyse the security of the cryptographic protocols because existing methods like BAN logic and SVO logic turned out to be somewhat unsatisfactory with regard to their model and protocol goal structures. We present the security analysis, based on our new method, of the generic protocol designed in the research in a later section. The new method is sufficiently general and comprehensive to be used for any cryptographic protocols for authentication and key establishment between two entities.

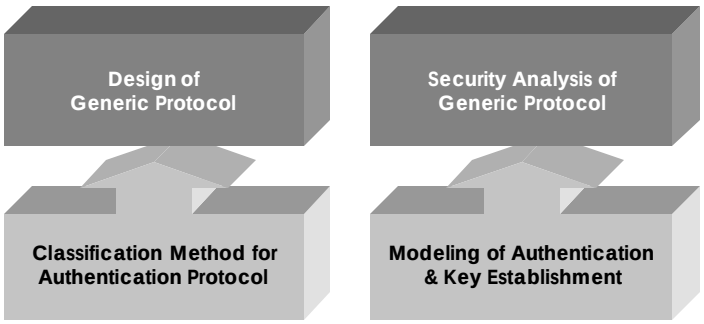


Figure 2-1: Two new methods for design and analysis of security protocols

2.1 Classification of Authentication Protocols: A Practical Approach

This chapter discusses a simple classification method for public-key based authentication protocols, which consists of identifying several basic properties leading to a large number of generic prototypes for authentication. Most published protocols can be identified as a concrete instance of one of the generic types. The classification method provides a means to clarify the similarities and differences between different concrete protocols, and thus facilitates avoidance of previous mistakes when designing a new protocol and allows re-use of analysis of a given abstract protocol when classifying any given concrete protocol.

2.1.1 Introduction

Authentication may be considered as one of the most critical elements of cryptographic security protocols; in some sense we can consider most cryptographic protocols as various extensions of the relevant authentication protocols. Most research in authentication protocols focuses either on a generic methodology of analysis (e.g., formal logic approaches [BAN90]) or design of novel protocols for a particular set of requirements from application environments. In this chapter we propose a simple, but nevertheless very useful, approach toward authentication protocol design and analysis: *classification of authentication protocols.*

With so many authentication protocols published and analysed (and often having turned out to be weak) it may be hard to believe that we still do not have a well-established classification methodology for them. This is probably because most

protocols are described in detailed mathematical notation. Secure authentication protocols rely upon sound integration of underlying cryptographic primitives and the series of message exchange between protocol principals. Consequently, the analysis of any given authentication protocol will require examining three points: underlying algorithms used in the protocol; the procedure of message exchange; and the secure combination of these two ingredients. We only focus on the second point, namely message exchanges in authentication protocols. This allows us to abstract away from the details of the cryptographic mechanisms and concentrate instead on the fundamental protocol structure.

2.1.2 Concepts, Definitions and Notations

The conventional ways to describe authentication protocols depends heavily on mathematical terms such as discrete exponentiations and hash functions. Even abstract level notations are sometimes not free from these mathematical languages. When trying to establish a method of classification, we have noticed that a new notation is required which is free from mathematical functions and, furthermore, too specific notions of security services such as digital signature. The declaration of freedom from a security service such as signature might sound rather radical. In practice many conventions for abstract description (such as $\{\bullet\}_{K_A^{-1}}$ or $Sig_A\{\bullet\}$) intend digital signatures using the private key as the concrete interpretation for the description. Our classification makes no such assumption. This may sound rather odd, but will be made clear when we deal with Protocol 12 in Section 2.1.6. There we will see that a form like $(g^{r_A})^b$ can be interpreted as a cryptographic operation over g^{r_A} using the entity B's private key b but cannot serve as signature because the same result can be achieved by the entity A computing $(g^b)^{r_A}$.

The inclusion of extraneous information when describing authentication protocols may sometimes turn out to be a stumbling block on the way to a clear perspective with regard to classification work. Classification implies a collection of general prototypes, each of which may cover many concrete instance protocols. In this regard, classification work seems to naturally require us to identify the essential elements in authentication protocols. These elements behave as cryptographic particles, and the way they are

combined and used seems to be a good criterion for classification of authentication protocols.

The ultimate goal of entity authentication in its cryptographic context is to check if the identity claimant or prover has the relevant private information, namely the private key of the principal whose identity has been claimed by the prover. In other words, the relevant private key should be used in the current session of an authentication protocol and the protocol must provide the verifier a clear indication of the application of the private key to particular data. This particular data, of course, must have a freshness property to prevent any replay attack. Another important property which is desirable in any entity authentication protocol is that the verifier needs to be sure about whom the prover is claiming an identity to.

From these simple observations, we can define cryptographic particles as follows.

Definition 2-1. *Cryptographic particles of an authentication protocol include the following two types, each of which is subdivided into two sub-types*:

- identities
  - *A, B*
- key values
  - *APriKey*: *the private key of the principal* A *whose identity is claimed to the verifier*
  - *APubKey*: *the public key of the principal* A *whose identity is claimed to the verifier*
- fresh data
  - *forced challenge*
  - *self challenge*

The two different types of fresh challenge data need to be more clearly described in the following definition.

Definition 2-2. *Forced Challenge* (**F**), *Self Challenge* (**S**) *and No Challenge* (Æ)
If the fresh data is a random nonce generated by the verifier and then delivered as a plaintext or a ciphertext to the prover, then we say that the protocol uses forced challenge to authenticate the prover. On the other hand, if the fresh data is generated

by the prover himself the protocol is said to use self challenge. When there is no challenge value exchanged in the protocol, we say that the protocol has no challenge.

Self challenge values may include sequence numbers and timestamps (both of these types will be denoted as TS_A or TS_B in this chapter) whose timeliness can be checked by the verifier.

Now, taking a look at how the key values may be used in authentication protocols, we can observe two different patterns as follows.

- Application of *APubKey* by the verifier and then *APriKey* by the prover
- Application of *APriKey* by the prover and then *APubKey* by the verifier

Each of these two patterns has to be woven together with fresh data. Hence the above two patterns can be developed into more detail as follows.

- Application of *APubKey* by the verifier and then *APriKey* by the prover:
  The following example uses a forced challenge, that is a random nonce chosen by the verifier B.

  1. A $\leftarrow$ B: $APubKey\{B, r_B\}$
  2. A $\rightarrow$ B: r_B

- Application of *APriKey* by the prover and then *APubKey* by the verifier:
  The first example example uses a forced challenge,

  1. A $\leftarrow$ B: r_B
  2. A $\rightarrow$ B: $APriKey\{B, r_B\}$

  whereas a self challenge is used in the second.

  1. A $\rightarrow$ B: $APriKey\{B, TS_A\}$

Here $APriKey\{B, r_B\}$ does not necessarily imply a signature transformation but simply an application of *APriKey* to the other cryptographic particles B and r_B. It can be implemented by $\{B, r_B\}_{K_A^{-1}}$, $\{h(B, r_B)\}_{K_A^{-1}}$ or some other way that the verifier B can validate the authenticity of the result. Likewise, $APriKey\{B, TS_A\}$ may correspond to a more concrete form such as "$TS_A, \{h(B, TS_A)\}_{K_A^{-1}}$". On the other hand, $APubKey\{B, r_B\}$ roughly corresponds to any encryption using *APubKey* from which only A can retrieve

r_B. In essence, we focus only on authentication, not any other property like signature or key establishment here.

We have so far obtained three different elementary types of authentication protocols, but clearly we are still far from complete. Sometimes, for instance, we see a protocol in which there is no explicit response message to its corresponding challenge. Before we enter into more comprehensive steps for classification, we introduce some terminology to avoid too verbose description of the classification steps.

Definition 2-3. *Origin Authentication* (***OA***) *and Destination Authentication* (***DA***)
If a protocol contains a message of the form APriKey$\{\cdot\}$ *then we say the protocol provides origin authentication of the entity A, whereas if it contains a message of the form APubKey*$\{\cdot\}$ *then it provides destination authentication of the entity A.*

Definition 2-4. *Implicit Authentication* (***IA***)

If a protocol contains no message of the form APriKey$\{\cdot\}$ *or APubKey*$\{\cdot\}$*, but still requires the entity A to compute a value of the form APriKey*$\{\cdot\}$*, then we say that this protocol provides implicit authentication of A.*

One thing to note regarding our new term *implicit authentication* is that there is a similar term *implicit key authentication* which is a feature of some key establishment protocols [MvOV97, p.492]. These two different usages of the same term are rather orthogonal to each other. Furthermore, although there seems to be no explicit definition of *implicit* (*entity*) *authentication* in the literature, some authors use the notion *implicit* in the context of entity authentication as well. The following excerpt shows this point [ISO96].

> *B is "explicitly" authenticated based on B's signature transformation. A is "implicitly" authenticated based on the use of its public encipherment transformation, since only A is able to decipher the key token.*

Hence the existing concept of implicit entity authentication corresponds roughly to our destination authentication, and explicit entity authentication to our origin authentication. On the other hand, with our definition of implicit authentication, there seems to be no corresponding terminology in the literature. The exact meaning of these terms in our

context will become clearer when we deal with the final classification table in a later section.

2.1.3 Classification Steps

We now have prepared all the tools we need to carry through the sequential steps for classification. Each of the following steps are related to the types of cryptographic particles and/or the way they are combined and used in authentication protocols.

Step I: Identify the type of authentication adopted in the given protocol; is it implicit authentication (**IA**), origin authentication (**OA**) or Destination Authentication (**DA**)?

Step II: Identify the type of challenge values used in the given protocol; is it forced challenge (**F**), self challenge (**S**) or no challenge (**Æ**)?

At the end of this step, the protocol is identified as one of the seven types[2]:

$$IA_{\varnothing}, IA_F, OA_{\varnothing}, OA_S, OA_F, DA_{\varnothing}, DA_F.$$

Step III: In the case of the type DA_F, namely DA type of authentication with forced challenge, is there any subsequent response from the prover to the challenge? According to the answer to this question, the protocol is divided into one of the two types: $DA_{F, NoAck}$ and $DA_{F, Ack}$. The former stands for the answer "No" and the latter "Yes".

Now, we have eight different types of authentication in total, which are summarized and described in detail in the following Table 2-1 and Table 2-2. In Table 2-2, the principal *B* is the verifier authenticating the claimed identity *A*, and the expression such as "A: $APriKey\{B, r_B\}$" means that the principal *A* computes $APriKey\{B, r_B\}$ whereas "A $\rightarrow$ B: $APriKey\{B, r_B\}$" means that *A* computes $APriKey\{B, r_B\}$ and then sends it to *B*. Also note that all the example protocols include only cryptographic particles: *identities, keys, forced challenges and/or self challenges.*

---

[2] Here not all of the combinations arise; for example, the types DA_S and IA_S cannot occur. DA_S message will be of the form A $\leftarrow$ B: $APubKey\{TS_B\}$. The use of self challenge such as timestamps is to guarantee to the verifier *B* that the private key of the prover, here *APriKey* is applied to the self challenge value. Hence DA_S message violates the very definition of self challenge. The *timeliness* property of self challenge also obviates the IA_S type. The self challenge based proving message is to be delivered to the verifier in timely manner, which leads us to OA_S.

The example protocols for the prototypes are not intended to be perfect in terms of security. In fact, some proper modifications will have to be made to these example protocols to make them flawless. Furthermore, all prototypes without any challenge at all such as $IA_{\varnothing}$, $OA_{\varnothing}$ and $DA_{\varnothing}$ can never be made secure against replay attacks because of their inherent lack of freshness. Also note that the classification table does not include the type IA_S, namely implicit authentication with self challenge, simply because it seems questionable to consider timeliness of any self challenge value or time variant parameter which is not delivered at all in the current session of a particular authentication protocol.

Although all the description of the classification have been made using asymmetric cryptography, the principle and result here is essentially the same for symmetric key protocols too. For example, an example symmetric key protocol corresponding to OA_S will be like "TS_A, $K_{AB}\{B, TS_A\}$", where K_{AB} is the secret shared key between A and B.

Table 2-1: Classification steps

IA	$IA_{\varnothing}$	
	IA_F	
OA	$OA_{\varnothing}$	
	OA_S	
	OA_F	
DA	$DA_{\varnothing}$	
	DA_F	$DA_{F, NoAck}$
		$DA_{F, Ack}$

Table 2-2: Eight different types of authentication and the corresponding examples

Authentication type		*Example*
IA *Implicit Authentication*	$IA_{\varnothing}$	A : $APriKey\{ B \}$
	IA_F	A $\leftarrow$ B : r_B A : $APriKey\ \{ B, r_B \}$
OA *Origin Authentication*	$OA_{\varnothing}$	A $\rightarrow$ B : $APriKey\{ B \}$
	OA_S	A $\rightarrow$ B : TS_A, $APriKey\{ B, TS_A \}$
	OA_F	A $\leftarrow$ B : r_B A $\rightarrow$ B : $APriKey\ \{ B, r_B \}$
DA *Destination Authentication*	$DA_{\varnothing}$	A $\leftarrow$ B : $APubKey\{ B \}$
	$DA_{F, NoAck}$	A $\leftarrow$ B : $APubKey\{ B, r_B \}$
	$DA_{F, Ack}$	A $\leftarrow$ B : $APubKey\{ B, r_B \}$ A $\rightarrow$ B : r_B

2.1.4 Mutual Authentication Protocols

In the previous section we derived all the possible prototypes; eight different ones in total. Now let us consider how many different prototypes of mutual authentication protocols there are altogether of each kind. We can easily count them exhaustively as follows. First, we can see that the answer will be not more than $8^2 = 64$ because we consider eight different types for both direction: authentication of *B* by *A* and vice versa. Any protocol has an initiator entity, which we call *A*, and a responder entity, *B*. In order to rule out protocols which are merely mirror images of some other protocols, with initiator and responder interchanged, we will regard prototypes where *B* acts as the initiator as *illegal*. Consider the following example protocol corresponding to the prototype $DA_{F,Ack}$-OA_S, which means $DA_{F,Ack}$ for *authentication of B by A*, and OA_S for *authentication of A by B*.

Protocol 1. *An example of the mutual authentication prototype $DA_{F,Ack}$-OA_S*

1. A $\rightarrow$ B: $BPubKey\{ A, r_A, TS_A, APriKey\{ B, TS_A \} \}$
2. A $\leftarrow$ B: r_A

The pattern *BPubKey*{*A*, r_A, …} from *A* to *B* and the subsequent response r_A from *B* to *A* contributes to destination authentication of *B* by *A*, whereas *APriKey*{*B*, TS_A, …} to origin authentication of *A* by *B*. Note that this example is not the only possible one; for instance, *APriKey*{*B*, TS_A, …, *BPubKey*{*A*, r_A, …}} can be considered as another example of the prototype as well. What about its symmetric counterpart OA_S-$DA_{F,Ack}$, that is OA_S for authentication of *B* by *A*, and $DA_{F,Ack}$ for authentication of *A* by *B*? According to this combination of prototypes, we can construct the following example.

Protocol 2. *An example of the mutual authentication prototype OA_S-$DA_{F,Ack}$, which is an illegal case*

1. A ← B: *APubKey*{ *B*, r_B, TS_B, *BPriKey*{*A*, TS_B } }
2. A → B: r_B

This protocol violates the condition that *A* is the initiator of the protocol. It is tempting to assume that every possible legal prototype has a symmetric illegal prototype unless both *A* and *B* use the same type of authentication. This would mean that the total number of mutual authentication prototypes should be the eight special cases with the same authentication type used for both *A* and *B*, plus half of the remaining 56, or 36 in total. But in fact this is not the case. For example, OA_F-$DA_{F,Ack}$ and its counterpart $DA_{F,Ack}$-OA_F are both legal combinations as shown below.

Protocol 3. *An example of the mutual authentication prototype OA_F-$DA_{F,Ack}$*

1. A → B: r_A
2. A ← B: *BPriKey*{ *A*, r_A, *APubKey*{ *B*, r_B } }
3. A → B: r_B

Protocol 4. *An example of the mutual authentication prototype $DA_{F,Ack}$-OA_F*

1. A → B: *BPubKey*{*A*, r_A }
2. A ← B: r_A, r_B
3. A → B: *APriKey*{*B*, r_B }

Although protocols 3 and 4 are symmetric in the sense of the use of prototypes, both are, nevertheless, different legal protocols with the principal *A* as the initiator. In fact, we constructed all the examples corresponding to the 64 (= 8^2) mutual authentication prototypes and then identified 17 prototypes among them to be illegal combinations, with 47 (= 64 – 17) legal prototypes finally at our hands.

All the possible prototypes including legal and illegal cases are tabulated in Appendix "Prototypes of Authentication Protocols". Many published protocols are sorted into their corresponding prototypes. The table in the appendix shows 12 prototypes developed into published protocols. This figure may not be exact because there can be other protocols we missed. When we examine unused prototypes from the table we see that many of them can be found as minor variations on used ones. For example, the prototype $DA_{F,Ack}$ - IA_F is found to be undeveloped, yet can be obtained from the developed prototype $DA_{F,Ack}$ - OA_F by simply deleting its final message.

Considering the lack of freshness property, it is not surprising that all but one (and hence 24) mutual authentication prototypes using no challenge (∅) for either or both directions are not developed into any published protocols. The only exception is the prototype, $IA_{\varnothing}$-$IA_{\varnothing}$, to which the "ISO/IEC Key Agreement Mechanism 1" known as non-interactive Diffie-Hellman key agreement belongs. The following Protocol 5 shows the protocol. Each enitity applies his own private key to the other entity's public key to obtain the shared secret key between them.

Protocol 5. *ISO/IEC Key Agreement Mechanism 1 (Prototype $IA_{Æ}$-$IA_{Æ}$)*

A: $APriKey\{BPubKey\}$
B: $BPriKey\{APubKey\}$

Through deriving all the possible prototypes, we also found that the prototype $DA_{F,Ack}$-OA_F had not been published as any particular instance form in spite of its attractive properties for mobile communication security[3].

Protocol 6. *Prototype $DA_{F,Ack}$-OA_F*

1. A → B: $BPubKey\{ A, r_A \}$
2. A ← B: r_A , r_B
3. A → B: $APriKey\{ B, r_B \}$

In Protocol 6, the cryptographic form of the first message is particularly felicitous for mobile communication environment as mobile terminals (*A* in Protocol 6) can access the public key of the network (*B* in Protocol 6) using the system broadcast channel,

[3] It was at much a later time that we noticed another protocol belonged to the same prototype. It is the famous SSL/TLS protocol widely used in the Web environment. This unintentional mistake is not surprising because the protocol was described in the conventional generic level by its designers, and its generic equivalent form was worked out by Paulson [Paul99] who had have to spend about two weeks to derive the generic form from the original SSL/TLS description [DiAl99].

which is, anyway, necessary for call setup aside from security purpose. The third protocol message is compliant with digital signature of relevant messages under the user's private key, *APriKey*. Hence, this prototype is particularly suitable for future mobile communication security, where nonrepudiation of user messages is important for electronic payment. This prototype was developed into a more specific and secure protocol, which is the new generic WAKE protocol designed in the thesis (see Section 3.1).

Of course we can apply more than one prototype to a unilateral authentication; hence we can achieve a very complex prototype like $OA_F DA_{F,NoAck}$ - $OA_F DA_{F,NoAck}$, which means each entity authenticates the other using two prototypes OA_F and $DA_{F,NoAck}$. For our 47 basic prototypes, however, we did not count this kind of multiple application of prototypes. This is because, first of all, that kind of excessive usage of expensive public-key computation cannot be justified without a sufficiently good reason, and secondly any trial to include all the possible multiple application of prototypes will end up with infinitely many prototypes! However, it is of interest to note that we can find that kind of multiple application of prototypes in the literature; we present one example from ISO/IEC key transport mechanisms [ISO96].

Protocol 7. *A simplified description of ISO/IEC key transport mechanism 5*

1. A $\rightarrow$ B: r_A
2. A $\leftarrow$ B: $S_B(r_B, r_A, A, E_A(B, K_B))$
3. A $\rightarrow$ B: $S_A(r_A, r_B, B, E_B(A, K_A))$

Session key, $K_{AB} = one\_way\_function(K_A, K_B)$

Here S_A and E_A denote *A*'s private signature transformation and *A*'s public encipherment transformation, respectively. After a simple check, we can identify the protocol as of the type $OA_F DA_{F,NoAck}$ - $OA_F DA_{F,NoAck}$. Here we counted each secret key of K_A and K_B as a random challenge or forced challenge because we can consider a secret key as a random value. We might have a view of this protocol that each entity authentication comes from a signature transformation and key confidentiality is guaranteed by the adoption of public key based encryption. In this way we may classify the protocol as an instance protocol of the prototype OA_F-OA_F. However, from another viewpoint, we may consider the encryption fields in the protocol contribute to entity

authentication as well and there is clearly multiple application of prototypes in either direction. When the signature operation is not critical, this protocol will be an overkill.

In the classification work, we have not considered the server-based authentication protocols since the main focus of this thesis is on the public key protocols where we are allowed to assume the availability of public keys in the protocol principals. Nevertheless, we can think about the application of this classification technique to the server-based authentication protocols by adopting the "divide and conquer" policy, namely dividing the protocols into two segments *A-S* and *B-S* where *A*, *B*, and *S* stand for principals *A*, *B* and the server *S*, and classifying each into one of the eight types described in Table 2-1.

2.1.5 Learning lessons from failure

Past experience has proven that many protocols have turned out to be weak or faulty in some way. We believe that these failures can be used to help us avoid repeating the same mistakes. Unfortunately, however, current practice regarding protocol design and analysis is not promising in this point. We believe that a classification-oriented perspective is critically important because it enables us to approach a particular authentication protocol with its corresponding prototype in our mind. In this way, we are able to reuse lessons learned from our past failure in a concrete protocol. This will be demonstrated as follows.

First let us take a look at the following example protocol belonging to the prototype $DA_{F,Ack}$-$DA_{F,Ack}$.

Protocol 8. *An example protocol of the prototype $DA_{F,Ack}$-$DA_{F,Ack}$*

1. A $\rightarrow$ B: $BPubKey\{A, r_A\}$
2. A $\leftarrow$ B: r_A, $APubKey\{B, r_B\}$
3. A $\rightarrow$ B: r_B

The cryptographic messages *BPubKey*{…} and *APubKey*{…} are meant to authenticate their respective destination entities (so this protocol is of the form DA-DA), and each of them contains a random nonce as the required (forced) challenge (DA_F-DA_F), to be followed by a corresponding response, thus resulting in the prototype $DA_{F,Ack}$-$DA_{F,Ack}$. This protocol is directly constructed from its corresponding prototype, not the other way around. In the construction process, of course, we applied a precious lesson learned

from failures of existing protocols: *include the proper identity field in encryption or signature messages* [AbNe94]. Hence A and B are included in the $BPubKey\{\ldots\}$ and $APubKey\{\ldots\}$ in the protocol; in fact we defined *identities* as one type of cryptographic particles earlier in this chapter. This protocol may be strengthened even further, thwarting any attack similar to the so called *Canadian Attack* [Goll94, MiTh93]. The original context in which this attack was introduced was signature based authentication protocol corresponding to the prototype OA_F-OA_F, where $APriKey\{\bullet\}$ and $BPriKey\{\bullet\}$ type messages are used instead of $APubKey\{\bullet\}$ and $BPubKey\{\bullet\}$. Despite this difference, Protocol 8 is vulnerable to the essentially same attack as shown below.

Canadian attack applied to Protocol 8

1. E\A $\rightarrow$ B: $BPubKey\{A, r_A\}$
2. E\A $\leftarrow$ B: r_A, $APubKey\{B, r_B\}$
   1'. A $\leftarrow$ B/E: $APubKey\{B, r_B\}$ — *E* initiates another session with *A*, masquerading *B* to *A*.
   2'. A $\rightarrow$ B/E: r_B, $BPubKey\{A, r_A'\}$
3. E\A $\rightarrow$ B: r_B

Here the notation E\A $\rightarrow$ B means that E is masquerading A to B[4]. The result of the attack is the same as when it is applied to protocols of the prototype OA_F-OA_F: the attacker E is accepted as a false identity A to B. Here, the victim entity A is used as an oracle by the attacker. This protocol can be fixed by including r_A in $APubKey\{B, r_B\}$ of the second message, resulting in the following protocol.

Protocol 9. *Fixed protocol of the prototype $DA_{F,Ack}$-$DA_{F,Ack}$*

1. A $\rightarrow$ B: $BPubKey\{A, r_A\}$
2. A $\leftarrow$ B: r_A, $APubKey\{B, r_A, r_B\}$
3. A $\rightarrow$ B: r_B

Inclusion of r_A in the encrypted message and the subsequent response r_B guarantees to B that A is aware of the r_A which he sent in the first message. Here we can see that the protection against the Canadian attack is not message authentication (of r_A here) but rather a sort of message awareness (of r_A by A).

[4] In fact, this notation is closely related to the work of authentication modelling in Section 2.2, and described in much greater detail in section 2.2.2.

Now we have developed a good generic protocol corresponding to the prototype $DA_{F,Ack}$-$DA_{F,Ack}$. Using this prototype and its generic protocol, we can analyse any other concrete protocols published so far. The first example is the Needham-Schroeder public key protocol [NeSc78], whose simplified description using our notation is as follows.

Protocol 10. *Needham-Schroeder public-key protocol*

1. A $\rightarrow$ B: $BPubKey\{r_A, A\}$
2. A $\leftarrow$ B: $APubKey\{r_A, r_B\}$
3. A $\rightarrow$ B: $BPubKey\{r_B\}$

It can be easily checked that this protocol belongs to the same prototype as Protocol 9. The third message is only for confidentiality of r_B for use in secret session key establishment. We can achieve more economically the same effect in the third message by replacing encryption of r_B with hashing of r_B. Now let us compare this protocol with Protocol 9. The Needham-Schroeder protocol lacks the identity B in $APubKey\{\bullet\}$ in the second message. This leads to the attack presented by Lowe [Lowe95].

Another instance of the same prototype is protocol 9, known as Helsinki Protocol [HoHs98], [MiYe98], which is specified as Key Transport Mechanism 6 in ISO/IEC DIS11770-3 [ISO96].

Protocol 11. *Helsinki protocol*

1. A $\rightarrow$ B: $BPubKey\{A, K_A, r_A\}$
2. A $\leftarrow$ B: $APubKey\{K_B, r_A, r_B\}$
3. A $\rightarrow$ B: r_B

Comparison with Protocol 9 shows that this protocol lacks the identity B in the second message, which is exactly the same pattern of weakness in the Needham-Schroeder protocol. Lowe's attack on the Needham-Schroeder appeared in 1995, which applies to the Helsinki protocol as well, which was proposed in the same year. Nevertheless, this weakness was found again in 1998 by Horng and Hsu [HoHs98]. Then, in the same year, Mitchell and Yeun amended it [MiYe98] by including the identity B in the encrypted part of the second message of the protocol, which is exactly the same way Lowe has done for the Needham-Schroeder protocol. Of course, Mitchell and Yeun were well aware of the same pattern of weakness and the corresponding remedy in both protocols.

In summary, we can see that every concrete protocol of the same prototype suffers from the same pattern of weakness, and can be fixed with the same pattern of remedy. This is why it is very important for us to identify the corresponding prototype when we are given a concrete protocol. We believe that our classification method is simple and very helpful for identification of the prototypes of authentication protocols.

2.1.6 Deriving a prototype for a given protocol

In the previous sections we have identified the type of a number of particular protocols. Sometimes, however, it is not so obvious. The first example protocol (Protocol 12) is an authentication and key establishment protocol proposed for the Universal Mobile Telecommunications System (UMTS) by the European ASPeCT project [ASPe96].

We first need to identify all the challenges exchanged. We can easily see that r_B plays the role of forced challenge from B to A. As for the reverse direction, it is not so clear, but a little thought shows that g^{r_A} is what we are looking for. The network B receives g^{r_A} and then raise it to the power of b which is its own private key. This corresponds to the generic form $BPriKey\{r_A\}$.

Protocol 12. *ASPeCT protocol for authentication and key establishment in UMTS*
A: UMTS user, B: UMTS network

1. A $\rightarrow$ B: g^{r_A}
2. A $\leftarrow$ B: r_B, $h_2(K_{AB}, r_B, B)$
3. A $\rightarrow$ B: $\{\{h_3(g^{r_A}, g^b, r_B, B, K_{AB})\}_{K_A^{-1}}, A\}_{K_{AB}}$

A, B: $K_{AB} = h_1(r_B, g^{br_A})$

Notation:

g: a generator of a finite group
r_A, r_B: random numbers chosen by A and B, respectively
b, g^b: a certified long-term private and public key of B for key agreement
K_{AB}: session key to be shared between A and B
$\{\bullet\}_{K_A^{-1}}$: a signature using A's private signature key K_A^{-1}
$\{\bullet\}_{K_{AB}}$: an encrypted message using the session key K_{AB}
h_1, h_2, h_3: one-way hash functions specified for ASPeCT protocol
$g^{\bullet}$: discrete exponentiation modulo a prime

Remember that here we need to ignore too specific mathematical forms like g^{r_A} and $h(\bullet)$; instead, we only focus on the essential elements, i.e., cryptographic particles as described in Definition 2-1. In this way, of course, we are inevitably led to overlook or omit other important properties of given protocols, such as key establishment and digital signature. However, our foremost goal of this classification work is to derive all possible generic forms of *entity authentication*. That is why our oversimplification may be justified in this work. In this particular protocol, for instance, we can think of g^{r_A} as a random challenge from A to B when entity authentication itself is a concern. In other words, there is no change with regard to the goal of entity authentication even if we replace g^{r_A} with simply r_A. The specific form like g^{r_A} is for secure key establishment, but it is not of any importance to our classification purpose.

The subsequent transformation of hashing $h_1(r_B, g^{br_A})$ does not affect this basic cryptographic property with regard to challenge-response. Focusing only on the cryptographic particles in this way, we can derive the corresponding authentication prototype of the protocol: OA_F-OA_F. That is, in this protocol, the user A and the network B authenticates each other with the basic prototype *origin authentication using forced challenge followed by response* (see Definition 2-2 and Definition 2-3).

The interesting point to be noted in this protocol is that we do not have to rely on signature transformation to obtain protocols of the prototype OA_F-OA_F. The message $g^{br_A} = (g^{r_A})^b$ which corresponds to $BPriKey\{ g^{r_A} \}$ does not provide any signature property at all because it can be computed as $(g^b)^{r_A}$ by the other party A as well. After all, as far as authentication is concerned, we do not consider any more compound property like digital signature. We can reason that this protocol may have a similar strength (and weakness if any) to its related protocols of the same prototype, of which the well-known STS protocol is an example [DvOW93].

Let us investigate the Yacobi-Shmuely protocol [YaSh89] as a second example.

Protocol 13. *Yacobi-Shmuely protocol*

1. $A \rightarrow B: a + r_A$
2. $A \leftarrow B: b + r_B$

A: $K_{AB} = ((g^{b+r_B})g^{-b})^{r_A} = g^{r_A r_B}$

B: $K_{AB} = ((g^{a+r_A})g^{-a})^{r_B} = g^{r_A r_B}$

Notation:

a, g^{-a} : *long-term private and public keys of A*

b, g^{-b} : *long-term private and public keys of B*

r_A , r_B : *short-term random nonces chosen by A and B, respectively*

K_{AB} : *new session key to be shared between A and B*

Here, *a* and *b* correspond to *APriKey* and *BPriKey*, respectively, and g^{-a} and g^{-b} to *APubKey* and *BPubKey*. The analysis regarding challenge-response is rather tricky in this protocol. Note that anyone can retrieve g^{r_A} and g^{r_B} from the protocol messages. Hence, we reason that, for instance, $a+r_A$ roughly corresponds to $APriKey\{\ g^{r_A}\ \}$ because anyone can retrieve g^{r_A} using *APubKey* which is a public data. We can see that g^{r_A} here does not play the role of a forced challenge as we defined it, because it is not combined with *B*'s private key but with *A*'s private key. This seems to be rather an anomaly; we can say that this protocol does not have any challenge values regardless of forced or self ones. Finally we can derive a generic protocol or prototype corresponding to this protocol as follows.

Protocol 14. *Generic form of the Yacobi-Shmuely protocol*

1. A $\rightarrow$ B : *APriKey*{ }
2. A $\leftarrow$ B : *BPriKey*{ }

We can see the protocol exactly corresponds to the $OA_{\varnothing}$-$OA_{\varnothing}$ (see Table 2-2). The concrete protocol now replaced by its generic prototype clearly shows that how strong or weak it is with regard to entity authentication. There is no guarantee of freshness in this protocol, which leads to its vulnerability to replay attacks. This weakness and attack was published several times, for example by Martin and Mitchell [MaMi98]. The translation from the concrete version to generic prototype described here is roughly the same procedure as that of BAN logic [BAN90]. The only difference is that our process is more concerned with prototype derivation. Even though the translation may appear trivial its significance is not trivial. We believe that if the Yacobi-Shmuely protocol had been known as an instance protocol which belongs to an inherently weak prototype $OA_{\varnothing}$-$OA_{\varnothing}$, then the following modification, [Park97], to the protocol would not have been tried at all.

Protocol 15. *A modified version of the Yacobi-Shmuely protocol*

1. $A \rightarrow B: g^{a+r_A}$
2. $A \leftarrow B: b + r_B$

A: $K_{AB} = ((g^{b+r_B})g^{-b})^{r_A} = g^{r_A r_B}$

B: $K_{AB} = ((g^{a+r_A})g^{-a})^{r_B} = g^{r_A r_B}$

The motive of modification is to alleviate the computational burden of discrete exponentiation within *B* (which may be a mobile terminal). However, changing $a + r_A$ to g^{a+r_A} $(= g^a g^{r_A})$ in the first message corresponds to changing *APriKey*{ } to a non-cryptographic message. Nobody except *A* can generate the message $a + r_A$ of the form *APriKey*{ } even though anyone can replay the message. However, the modified form allows anyone to generate a new value of g^{a+r_B} which is an entirely legal one.

2.2 Modelling of entity authentication and key authentication

Over the last twenty years a large number of authentication and key establishment protocols have been designed, and many different approaches were taken to cope with the difficulties in design and analysis of the protocols. Nevertheless, clear definitions and models of the key concepts and goals with regard to authentication and key establishment protocols do not seem to be finalised as yet. To partially complete this task we present a new model of authentication which enables us to look at key concepts in entity authentication and key authentication from a different angle. In particular, we have derived a quite different definition of key authentication. A few typical sets of protocol goals from the literature are critically reviewed. We also demonstrate the usefulness of the model as an analysis tool for authentication and key-establishment protocols.

2.2.1 Introduction

What is authentication? The answer to this question will inevitably be related to our interpretation of the world where the authentication process is performed. This world is one where a verifier authenticates somebody who claims an identity to the verifier. We first establish a new model of authentication by identifying every possible type of cast (entity or principal) living in the world of authentication. We argue that this is the most fundamental prerequisite before any formal logic is introduced to describe authentication. It is quite surprising that previous works in this field, however, do not seem to have considered this aspect of authentication thoroughly. Any formal logic for authentication may be viewed as a *description language* for *things* such as principals (the authenticator and the authenticatee) and their *interactions*. This is the reason why *any sophisticated logical method may be entirely useless unless it is based on a clear derivation of the objects which the logic is invented to describe*. Secondly, we present logical expressions describing most of the aspects of authentication and key establishment, and build up symbolic definitions corresponding to the existing relevant definitions such as entity authentication, key authentication and key confirmation. Thirdly, we set up several inference rules based on the proposed model. Thereafter, we

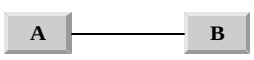
demonstrate how to apply our methodology to the analysis of two prominent protocols. Finally, a few typical sets of goals regarding entity authentication and key authentication are reviewed in comparison with ours, and a structured hierarchy of key concepts or goals in authentication protocols is presented.

2.2.2 World of Masks: Entity Authentication

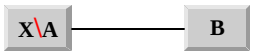
The world of authentication may be called a *world of masks*. There is no one without a mask on his face. Authentication can then be interpreted to mean the verification of the real entity under the mask. Saying the conclusion in advance, unfortunately we can never authenticate a principal who claims an identity, say A. This may sound ridiculous or, at least, too pessimistic, but it is true in some sense. Nevertheless, in a perfect authentication process, any true authentic entity will not fail in the process whereas any impersonating entity will *gain nothing* from cheating. These two seemingly incompatible conclusions show us two important aspects of authentication: its *ideal* (impossible) and *goal* (practical).

Now, we will cast actors for the world of authentication. What about the plain old characters such as Alice, Bob and Eve? What about P and Q who had once been casted by BAN logic? What about the A, B and an enemy (or intruder) E? We are not going to select the characters for ourselves. We must give the right of choice to the *authenticator* himself. We will use B to denote the authenticator or verifier, and A the claimed identity. Note that, in a strict sense, A is not a claiming *entity* but an *identity* which is claimed by some entity, which will be denoted by X. Of course, in everyday usage of language, we usually give up that kind of exactness in saying "I am David Copperfield" instead of "my name is David Copperfield", and in this chapter also, we often take that kind of practice. We cast X in addition to A and B because we want to see this world of masks not from an *omniscient viewpoint* but from the verifier's *point of* view. Most of the previous literature seems to have adopted the former, whereas we will choose the latter.

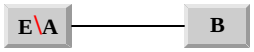
When we talk about authentication of principal A by B, we may like to depict the situation as follows.



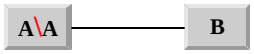
However, this might be considered as a kind of omniscient or naïve viewpoint. In the viewpoint of the authenticator *B*, the following depiction seems to be more exact.



Here, the notation $X \backslash A - B$ means that *B* sees a principal *X claiming* the identity *A*. In other words, some principal *X* is claiming the identity *A* to the principal *B*. Of course, the symbol "\" here is used to indicate a mask. That is, under the mask (or identity) *A* is some principal *X*. Everybody puts on a mask in the world of masks. As for *B*, he cannot see his own mask and we view the world from *B*'s viewpoint. The entity *X* may or may not equal *A*. If *X* is an entity *E* which is not *A*, the entity *E* is just impersonating *A*, which can be denoted as follows.



On the other hand, if the entity *X* is really equal to *A*, we may depict this as follows.



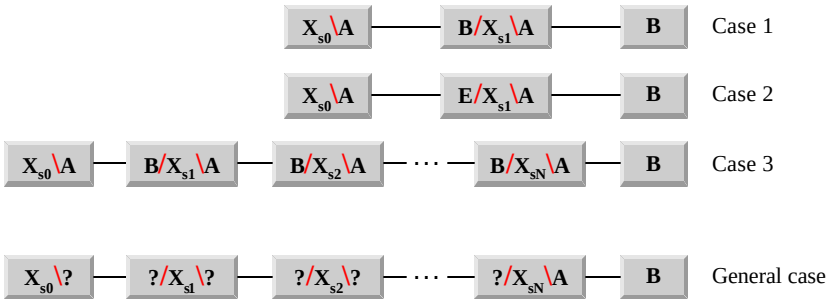
This is also another description from an omniscient viewpoint. In fact, B will never be able to know who was under the mask. We will later see why we must take such a view of the world of authentication. Now, we may specify our first definition of authentication as follows.

Definition 2-5. [EntityAuth0] Authentication *is the process for an entity* B *to verify that an entity* X *who says he is* A *is really* A.

The authentication satisfying this definition is, of course, an *ideal* which we have argued is *impossible*. This definition may have a meaning only when used as a reference point.

In the above description, we implicitly assumed that there are only two entities, one being the authenticator *B*, and the other *X* who is claiming the identity *A*. The real situation, however, may involve more entities than those two. As from the omniscient viewpoint, we can consider the possibility that a third entity plays the attack-in-the-middle. Similarly, when taking *B*'s point of view, we cannot be sure that there is only one entity *X* behind the mask *A*. In the worst case, there might be as many *X*s behind the

mask A as the number of all the entities except B himself in the world. The following diagrams show some possibilities.



In Case 1, $B/X_{s1}\backslash A$ means that X_{s1} is playing the man-in-the-middle attack between nominaly A and B by claiming the identity B to his left-side peer X_{s0} (niminally A) and at the same time by impersonating the identity A to his right-side peer B. Here, the number N of all the intermediate entities (attackers playing the man-in-the-middle attack) between the prover and the verifier may be equal to or greater than zero. In fact, only the last of the above pictorial depictions can be considered a complete picture from B's viewpoint. The verifier B is sure that the above chain must have an *end*, the original sender which we call the *originator* ($X_O = X_{S0}$). All the other entities may be called just *sending* entities or senders (X_{Si}, i = 1, 2, …, N). It should be noted that if there is no reasonable evidence provided, B cannot be confident about which mask (identity) has been put on by all the others except X_{SN} who is talking to B (hence just $X_{S0}\backslash ?$ and $?/X_{S\,k}\backslash ?$). This observation should be given its due importance but it was often ignored in previous work.

Cryptographic implementation of the claim of an identity A by a prover X is for X to send a relevant message to B to prove the possession of a secret data or key known only to the true entity A (possibly including B). The message contains at least one cryptographic message which is a result of *cryptographic operation* using A's secret key. The key may be A's private key (*APriKey*) in an asymmetric cryptosystem or a shared secret key between A and B (K_{AB}), either of which is to be used according to the protocol. Every resultant message of cryptographic operation using inappropriate keys would look like a garbage value to B and should lead to authentication failure.

According to this line of reasoning, we establish the following axiom regarding a principal and a key.

Axiom 2-1. *We say that a principal X "is" A when X shows that he "has" a secret key K which can be known only to A. In symbolic notation:* $X = A \Leftrightarrow X \ni K$.

The relation "*has*", in symbol "∋", must be interpreted only in the *current tense*. That is, when we say *"A ∋ m"* it means that *A* has *m now*. In the case of asymmetric key protocols, the key will be *A'*s private key (*APriKey*) corresponding to the certified public key (*APubKey*) of *A*. Thus, the above symbolic expression can be paraphrased as follows.

$$X = A \Leftrightarrow X \ni APriKey$$

Shared key based protocols cannot enjoy this kind of simplicity because there are two or more principals sharing the key. Assuming a shared key K_{AB} is always attributed to only two principals *A* and *B*, Axiom 2-1 leads to the following.

$$(X = A \vee X = B) \Leftrightarrow X \ni K_{AB}$$

This implies

$$X = A \Leftrightarrow (X \neq B \wedge X \ni K_{AB}).$$

Now, we need to introduce and define some terminology: *COM*, *NCOM*, an *originator* X_O, and a sender X_S. All the definitions here should be interpreted from the viewpoint of the verifier.

Definition 2-6. *Any protocol message falls into one of the two types: cryptographically operated messages* (*COMs*) *and non-COM messages* (*NCOMs*). *Any COM message has a form like* $Op(x)$ *for some input* x *and some operation* Op, *and* $Op(x)$ *can only be formed when* x *is known, where* x *may be a key or random number. In other words, if a message is a COM, it is a result of any cryptographic operation like encryption, decryption, signature, discrete exponentiation or hashing, using a relevant key, a random nonce, and/or a timestamp as a part of input data. All other messages without such a property belong to NCOMs; namely they can be generated without any cryptographic operation. We denote different COMs and NCOMs with* COM_i *and*

$NCOM_j$ *using indices i and j. We sometimes use* M_j *to denote the j-th protocol message which may include COMs and/or NCOMs.*

Definition 2-7. *Any COM in a protocol has a unique* originator X_O *who originated it in the current session. We use* X_O^i *to denote the originator of the message* COM_i *which is a COM.*

Definition 2-8. *A protocol message may have one or more senders, and the* set *of all the senders of the same message in a current session is denoted by* X_S *. We use* X_S^i *to denote the sender set of the message* $NCOM_i$ *or* COM_j *.*

Definition 2-7 does not necessarily mean that the originator should be at the point where the corresponding connection ends. An originator may be a man-in-the-middle between the honest prover *A* and the verifier *B*. If, for example, g^{r_A} from *A* to *B* is modified to g^{r_E} by an attacker *E*, then the originator of the *COM*, g^{r_E} is not *A* but the attacker *E*. Also note that, by definition, the originator X_O^k of the *k*-th message belongs to the set X_S^k of the corresponding senders. Therefore, it is trivially true that $X_O^k \in X_S^k$. The reason why the notion of X_O covers only *COM* is that the verifier is never able to verify who is the originator of a particular *NCOM*. The verification means a proper cryptographic check procedure and is not relevant for any *NCOM*.

Now on the basis of these observations, to verify the authenticity of the claimed identity *A* by *B* may be interpreted as *B*'s verifying the following question.

Is the originator X_O *really A?*

By Axiom 2-1, the above question may be paraphrased as follows.

Does the originator X_O *really have APriKey*? [asymmetric key protocol]
Does the originator X_O *really have* K_{AB} *and is he not B himself*? [symmetric key protocol]

For now we stipulate another definition of entity authentication as follows.

Definition 2-9. [EntityAuth1] Authentication *is the process for an entity B to verify an identity* (*A*) *claim by checking that the originator of the relevant cryptographic message has the relevant secret key. In symbolic notation*:

$X_O \ni APriKey$ [asymmetric key protocol]

$X_O \ni K_{AB} \ \wedge \ X_O \neq B$ [symmetric key protocol]

We can have arbitrarily many intermediate senders X_S in between the verifier *B* and the X_O as in the definition. This is why we cannot deny the impossibility of the ideal authentication as in the first definition of authentication [EntityAuth0].

We might have considered another suspicion of the authenticator B: *Is the originator X_O really the same identity as the entity who made the cryptographic message*? To address this question, we would require a new notion of some entity: the maker X_M of the cryptographic message. This new type of entity is, however, not essential in our model of authentication even though it may turn out to be a useful concept in some situations (see Section 2.2.6.3). This is because we have already incorporated essentially the same concept by adopting the "has ($\ni$)" operator in Axiom 2-1. Let's make this point clearer by looking at the following example.

Protocol 16.

1. A $\rightarrow$ B: r_A, *APriKey*{ r_A }

Here r_A denotes a random nonce chosen by *A*, and *APriKey*{ r_A } a cryptographic operation over r_A using *A*'s private key such as recoverable or non-recoverable signature. Can be *B* sure that the originator X_O of the cryptographic message 1 is really *A*, that is, has the relevant private key *APriKey*? There is no guarantee given to *B* here that $X_O = A$, i.e., $X_O \ni APriKey$ because X_O might have just replayed the cryptographic message 1 which was obtained from old session between *A* and *B*. If we adopt the notion of X_M we can say that *B* can be sure that $X_M \ni APriKey$ but cannot be confident about $X_O = X_M$ (or $X_O \ni APriKey$). However, if the nonce r_A is replaced by r_B chosen by *B* in the current session then *B* can be sure that $X_O \ni APriKey$ or $X_O = A$, because r_B is expected to be a *fresh* random value to prevent replay using a previous cryptographic message (see the following example).

Protocol 17.

1. A $\leftarrow$ B: r_B
2. A $\rightarrow$ B: *APriKey*{ r_B }

Thus we see that the "has (∋)" operator implying *current* tense obviates the need for a separate entity X_M to be involved in our model of authentication.

Now with **EntityAuth1**, we have captured the most important aspect of entity authentication, which can be directly used as a goal of any entity authentication protocol. However, is the definition comprehensive enough to accommodate most of the aspects of authentication? Maybe not, for there is no notion of a *peer entity* [5] in the verified details from the relevant message originated from the claimant *A*. In every day examples of authentication, we as claimants are well aware of the peer entity to whom we want to prove our identity, whereas the verifier has an implicit confidence that the corresponding claimant is aware of who his verifier is. This feature of authentication was not well understood in the early days of protocol design, which resulted in some weaknesses in many famous protocols such as Needham-Schroeder public-key protocol and Helsinki protocol. We may describe a stronger definition of authentication incorporating this feature as follows.

Definition 2-10. [EntityAuth2] *Authentication is the process for an entity* B *to verify* (*1*) *an identity* (A) *claim by checking that the originator of the relevant cryptographic message has the relevant secret key, and*
(*2*) *the originator is aware* ($\twoheadrightarrow$) *of* B *as his peer entity.*

In symbolic notation:

$$X_O \ni APriKey \wedge X_O \twoheadrightarrow \mathrm{B} \quad \text{[asymmetric key protocol]}$$

$$X_O \ni K_{AB} \wedge X_O \neq B \quad \text{[symmetric key protocol]}$$

Note that this stronger definition does not affect the symbolic expression for shared or symmetric key protocols. A shared key basically implies two peer entities, which is not the case for asymmetric key protocol. Consider the following example.

Protocol 18.

1. A $\leftarrow$ B: r_B
2. A $\rightarrow$ B: $K_{AB}\{ r_B \}$

[5] When two entities *A* and *B* are communicating with each other, we say *A* and *B* are the peers of the communication. *A* is the peer entity of *B* in the communication, and vice versa. Given the possibility of impersonation attack, we may have to use more verbose terminology such as *nominal* (or unauthenticated) *peer entity*. The entity *A* may have been fooled to believe that he is communicating with *B* but in fact with an attacker *E*. Nevertheless, to *A*'s eye, the (nominal) peer entity is *B*. Throughout the thesis, for simplicity, we use *peer entity* in place of *nominal peer entity*.

Here, $K_{AB}\{r_B\}$ denotes any cryptographic operation over r_B with the shared secret key K_{AB} between A and B including symmetric key based encryption and keyed-hashing. Can we consider this protocol as the shared key equivalent of the Protocol 17? Certainly not! This protocol may be considered to deliver more information than the asymmetric protocol. A more exact translation of this protocol to asymmetric language would be the following.

Protocol 19.

1. A ← B: r_B
2. A → B: $APriKey\{B, r_B\}$

Now in turn we can ask again: ' is this asymmetric key protocol equivalent to Protocol 18?' The answer could be yes or no. Note that the definitions of authentications, **EntityAuth1** and **EntityAuth2** stipulate that B has to check if $X_O \neq B$ in addition to $X_O \ni K_{AB}$. In a situation where the entity B is a server and A is a client, and only one way authentication is allowed, the X_O in Protocol 18 cannot ever be B which is not able to make an encrypted response in return to a random challenge. Then entity authentication of the claimant A is successful according to **EntityAuth1** or **EntityAuth2**. When Protocol 18 is used in a mutual authentication environment, however, B cannot be sure whether X_O = B or $X_O = A$. This aspect is closely related to the idea of reflection attack [BGH93], which is demonstrated for Protocol 18 as follows.

Reflection attack against Protocol 18

1. E\A ← B: r_B
   1'. E\A → B: r_B
   2'. E\A ← B: $K_{AB}\{r_B\}$
2. E\A → B: $K_{AB}\{r_B\}$

Here, from the message 2, B can be sure that the originator X_O^2 of the second protocol message has K_{AB} because the message includes the fresh random challenge which has been chosen by B itself. There is, however, no evidence that $X_O^2 \neq B$. To cope with this problem, Protocol 18 can be fixed by including either of the identities A and B as follows.

Protocol 20.

1. A $\leftarrow$ B: r_B
2. A $\rightarrow$ B: K_{AB} { B, r_B }

Of course, the identity B in the message 2 of the above protocol must be interpreted to mean the identity of the intended recipient of the message. If the identity A instead of B were used then it should mean the identity of the originator. Now we can say that Protocol 20 is exactly a symmetric key equivalent of Protocol 19.

The definitions **EntityAuth1** and **EntityAuth2** of entity authentication are based on the model of authentication where we have an entity claimant X_O, the claimed identity A, a verifier B and the secret and authentic key of A such as *APriKey* or K_{AB}. All the examples considered so far have only one protocol message which is a *COM* message (from the viewpoint of the verifier B). In many practical situations, however, the claimant may send more than one messages including even *NCOM* messages. Any *NCOM* message does not allow the verifier to check who its originator is. What should the verifier do to handle this problem? The reasonable check will be for B to verify whether all the originators (of course, they must be the same entity in successful authentication) belong to the set of the senders of the *NCOM* message. With this in mind, we propose an even stronger definition of authentication.

Definition 2-11. [EntityAuth3] *Authentication is the process for an entity* B *to verify*

(1) identity (A) claim by checking that for all COM_i the corresponding originator X_O^i has the relevant secret key of A,

(2) the claimant A is aware ($\twoheadrightarrow$) of the verifier B as his peer entity, and

(3) for all $NCOM_j$, the claimant A belongs to the set of the senders of the message.

In symbolic notation:

for asymmetric key protocols,

$\forall COM_i$ (X_O^i $\ni$ *APriKey*), ***Verification of Key Possession*** (**VKP**)

$A \twoheadrightarrow$ B, and ***Verification of Peer Entity Awareness*** (**VPA**)

$\forall NCOM_j$ ($A \in X_S^j$) ***Verification of Message Awareness*** (**VMA**)

for symmetric key protocols,

$\forall COM_i$ (X_O^i $\ni K_{AB} \wedge X_O^i \neq B$), and **VKP & VPA**

$\forall NCOM_j \ (A \in X_S^j)$ **VMA**

The additional feature captured in **EntityAuth3** is somewhat along the lines of the concept like *explicit context* or *correspondence* between peer entities in a protocol run. The following example protocol shows the significance of this new requirement of authentication.

Protocol 21.

1. A $\rightarrow$ B: r_A $(M_1 = NCOM_1)$
2. A $\leftarrow$ B: $K_{AB}\{B, r_A\}, r_B$
3. A $\rightarrow$ B: $K_{AB}\{A, r_B\}$ $(M_3 = COM_3)$

From the viewpoint of B, X_S^1 has sent r_A and X_O^3 has originated and sent the third message, i.e., COM_3. COM_3 gives a proof that X_O^3 has K_{AB} and is not B, and hence X_O^3 is the same entity as A. There is, however, no evidence that X_O^3 (=A) belongs to the sender set X_S^1. The following attack exploits this weakness.

Attack against Protocol 21

1. E\A $\rightarrow$ B: r_A
2. E\A $\leftarrow$ B: $K_{AB}\{B, r_A\}, r_B$
   1'. A $\leftarrow$ B/E: r_B
   2'. A $\rightarrow$ B/E: $K_{AB}\{A, r_B\}, r_{A'}$
3. E\A $\rightarrow$ B: $K_{AB}\{A, r_B\}$

At the end of the attack, the attacker E who is in fact not A has been authenticated as A to the verifier B. The cryptographic message 3 was generated by A as part of the second message in another session initiated by E. The victim entity A has never generated the random nonce r_A of the first message, that is A is the originator X_O^3 but does not belong to the sender set X_S^1, to which in fact E belongs.

One may argue that this attack does not violate the usual goal of authentication at all. However, when viewed from B as a verifier, the first message in the protocol may be thought of as a request to access B's time, effort, or any other resources, and the second message can be considered as a challenge to gather a proper proof of the identity claim stated in the first message. Consequently, the third message in the protocol may be regarded by B as the identity proof data to support the identity claim made just

before in the first message. In this regard, we can regard this attack as successful. Moreover, the cost to fix is just negligible, so why should we take a reluctant attitude toward making a suspicious protocol more robust? A simple remedy is to include r_A in the encrypted message 3 providing B evidence that $X_O^3 \in X_S^1$.

Our new approach in this section to define entity authentication may seem rather trivial, but if it is the case, then this might be just because our reasoning about authentication is so *natural*. Furthermore, the corresponding symbolic expression, when combined together with the *masking rules* to be introduced in a later section, will turn out to be an extremely simple and clear tool for protocol analysis and attack construction. This kind of symbolic definition should help us avoid most of the ambiguity which arises when we talk about the goals or features of authentication protocols as will also be shown in a later section.

2.2.3 Key Authentication and Entity Authentication

Key establishment is hard to separate from entity authentication. This aspect seems to account for confusion in the analysis of authentication and key establishment protocols. When it comes to define *key authentication*, most authors implicitly assume entity authentication as a prerequisite, as shown below [ISO96].

> ***Implicit key authentication to B***: *the assurance for one entity B that only another "identified" entity can (possibly) be in possession of the correct key*.

The following excerpt [ISO96] is another example of this tendency.

> ***Implicit key authentication*** *is the property whereby one party is assured that no other party aside from a specifically identified (authenticated) second party may gain access to a particular secret key.*

For reference, we translate the above concepts into a symbolic expression in the following definition.

Definition 2-12. [ImplicitKeyAuth1]

$$A!(\ni)\,K$$

Only ("!") the entity A **may** ("$(\cdot)$") **have** ("$\ni$") the secret key K.

As for another goal, *key confirmation* of key establishment protocols, there seem to be different trends. A definition in an international standard shows the notion of key confirmation implicitly combined with entity authentication [ISO96] whereas Menezes et al. rule out entity authentication from key confirmation [MvOV97] as shown below, respectively.

> ***Key confirmation to B***: *the assurance for one entity B that another identified entity is in possession of the correct key.* **[KeyConf1]**
>
> ***Key confirmation*** *is the property whereby one party is assured that a second party (possibly unidentified party) actually has possession of a particular secret key.* **[KeyConf2]**

We also translate the above definitions into symbolic expressions as follows.

Definition 2-13. [KeyConf1]

$$A \ni K$$

The entity A ***has*** *(“∋”) the secret key K.*

Definition 2-14. [KeyConf2]

$$\exists X (X \ni K)$$

Some entity X ***has*** *(“∋”) the secret key K.*

In Definition 2-14, X denotes some entity unidentified. When either one of these two different definitions is put together with *implicit key authentication* as shown above, we find ourselves having exactly the same conclusion of *explicit key authentication.*

Definition 2-15. [ExplicitKeyAuth1]

$$A! \ni K$$

Only *(“!”) the entity A* ***has*** *(“∋”) the secret key K.*

We now describe an important feature provided by the celebrated Diffie-Hellman key agreement protocol and show that all the above definitions fail to capture it. The following is a variant of the original protocol augmented with a third message for a sort of key confirmation from A to B (we may assume the reverse direction of key confirmation without loss of generality).

Protocol 22. *Diffie-Hellman key agreement protocol*

1. A $\rightarrow$ B: g^{r_A}
2. A $\leftarrow$ B: g^{r_B}
3. A $\rightarrow$ B: $h(K, A, B)$

where $K = g^{r_A r_B}$ and h is a common one-way hash function.

We summarize the conclusion of our analysis of this protocol in advance as a lemma, which will be used for analysis of the Station-to-Station protocol in a later section, and present the proof immediately.

Lemma 2-1. *After a successful run of the Diffie-Hellman key agreement protocol as described in Protocol 22, the entity* B *may conclude that the originators* X_O^1 *and* X_O^3 *of the first and third messages, respectively, are the same entity, and the originator and* B *are the only entities that have the new session key* K. *In symbolic notation*:

$$X_O^1 = X_O^3 \text{ and } (B, X_O^1)! \ni K$$

Proof. We take the viewpoint of B for the analysis. The first message g^{r_A} shows that if there is any entity who knows r_A he should be the originator X_O^1 of the first message. That is,

$$X_O^1 ! \ (\ni) \ r_A \tag{1}$$

If X_O^1 has sent just a garbage value which has nothing to do with discrete exponentiation, then there would be no entity in possession of the value r_A. Because B has chosen a *fresh* random nonce r_B, he knows that g^{r_B} is also fresh. Thus B can conclude that if some entity except himself has $K = g^{r_A r_B} = (g^{r_B})^{r_A}$ then the entity should have r_A too. In other words,

$$X \ni K \Rightarrow X \ni r_A . \tag{2}$$

From (1) and (2), B knows

$$X \ni K \Rightarrow X = X_O^1 . \tag{3}$$

Now, because g^{r_B} is fresh the resulting key K is also fresh. Hence the message 3 enable B to conclude

$$X_O^3 \ni K. \tag{4}$$

From (3) and (4), it follows that

$$X_O^3 = X_O^1. \quad (5)$$

Now let's denote this entity as $X_O^{1,3}$.

The basic property coming from Diffie-Hellman key agreement *B* is

$$\forall X (X \ni K \Leftrightarrow X \ni r_A \vee X \ni r_B). \quad (6)$$

B knows that r_A and r_B is known only to $X_O^{1,3}$ and *B* himself, respectively. Consequently, the session key *K* derived is only known to X_O^3 and *B*. ■

The conclusion of this lemma applies to the entity *A* as well if a relevant key confirmation message is sent from *B* to *A*. Even though we assumed an additional key confirmation message for the analysis of Diffie-Hellman protocol, it is a reasonable and practical attitude toward the analysis of protocols. Even without any key confirmation message assumed, we can describe the same property as in the lemma as follows, which will be a conditional or implicit truth for *B*.

$$\forall X (X \ni K \Rightarrow (X = X_O^1 \text{ and } (B, X)! \ni K)$$

Or,

$$X_O^1 ! (\ni) K$$

Now what should we call this property of Diffie-Hellman protocol? There is no entity authentication, and no implicit/explicit key authentication as defined in literature. We believe that this feature is better to be called *key authentication* or at least *unidentified key authentication* which has nothing to do with entity authentication unlike existing definition.

Definition 2-16. [ImplicitKeyAuth2] *Implicit key authentication is the process for an entity B to verify that*

(*1*) *all the originators of* COMs *should be the same entity*, *which is the* only *one to* be able to *get the new session key, and*

(*2*) *for all* $NCOM_j$, *the originator as in* (*1*) *belongs to the sender set* X_S^j *which has sent* $NCOM_j$.

In symbolic notation:

$$\exists X [(\forall COM_i (X = X_O^i \wedge X ! (\ni) K)) \wedge (\forall NCOM_j (X \in X_S^j))]$$

Definition 2-17. [ExplicitKeyAuth2] *Explicit key authentication is the process for an entity B to verify that*

(*1*) *all the originators of* COMs *should be the same entity, which is the* only *one* in possession of *the new session key, and*

(*2*) *for all* $NCOM_j$, *the originator as in* (*1*) *belongs to the sender set* X_S^j *which has sent* $NCOM_j$.

In symbolic notation:

$$\exists X\,[\,(\forall COM_i\,(X = X_O^i \wedge X\,!\ni K))\wedge(\forall NCOM_j\,(X\in X_S^j))\,]$$

Note that the above definitions tell that when implicit key authentication is provided to *B* and some entity turns out to have *K* then explicit key authentication is also satisfied. Namely,

$$\textbf{ImplicitKeyAuth2} \wedge \textbf{KeyConf2} \Rightarrow \textbf{ExplicitKeyAuth2}$$

Here **KeyConf2** as defined above is a *pure* key confirmation without any connotation of entity authentication. Also note the striking similarity between entity and key authentication (see **EntityAuth3** in Definition 2-11). Diffie-Hellman protocol gives us exactly the same service defined as in **ImplicitKeyAuth2**.

When we analyse a complex and complicated concept, it is important to be clear about elementary concepts and their combined meaning. To describe the Diffie-Hellman protocol as providing no key authentication available may lead us to undesirable deficiencies in design and analysis of security protocols. A description by Menezes et al. [MvOV97] about Shamir's no-key protocol may illustrate this kind of problem.

Protocol 23. *Shamir's no-key protocol*

1. A $\rightarrow$ B: $K^{r_A} \bmod p$
2. A $\leftarrow$ B: $(K^{r_A})^{r_B} \bmod p$
3. A $\rightarrow$ B: $(K^{r_A r_B})^{r_A^{-1}} \bmod p$

This protocol enables *A* to transport a new session key *K* to apparently *B*, which looks to achieve the same goal of Diffie-Hellman protocol. In fact, Menezes et al. state that the protocol achieves the same goals as an ElGamal variant for key agreement where the public key of one entity is replaced by an uncertified random public key to be used for key computation, which we understand is basically the same as the original Diffie-

Hellman protocol [DiHe76]. However, Shamir's no-key protocol does not provide any implicit or explicit key authentication property in the sense of Definition 2-16 and Definition 2-17 which the Diffie-Hellman protocol satisfies. The following attack procedures shows that *B* cannot conclude that the originators X_O^1 and X_O^3 of the message 1 and 3 should be the same entity. At the end of the first attack, *B cannot* conclude that $X_O^3 = X_O^1$, and moreover the confidentiality of the resultant key K (= $K^{r_A r_E} \bmod p$) to be shared with X_O^3 is very suspicious since it could be computed by X_O^1 as well. As for the second attack, X_O^3 is, in fact, not able to compute the key K and, of course, $X_O^3 \neq X_O^1$. These two attacks show that this protocol does not provide implicit key authentication as in Definition 2-16, which is satisfied by the Diffie-Hellman protocol.

An attack against Shamir's no-key protocol

1. A $\rightarrow$ B/E\A $\rightarrow$ B: $K^{r_A} \bmod p$ ($X_O^1 = A$)
2. E\A $\leftarrow$ B: $(K^{r_A})^{r_B} \bmod p$

   2'. A $\leftarrow$ B/E: $(K^{r_A})^{r_E} \bmod p$ (r_E : random nonce chosen by E)

   3'. A $\rightarrow$ B/E: $K^{r_E} \bmod p$
3. E\A $\rightarrow$ B: $(K^{r_A r_B})^{r_E} \bmod p$ ($X_O^3 = E \neq X_O^1$)

Session key for *A* and *B*/*E*: K (= $(K^{r_E})^{r_E^{-1}} \bmod p$)
Session key for *E**A* and *B*: $K^{r_A r_E} \bmod p$

Another attack against Shamir's no-key protocol

1. A $\rightarrow$ B/E\A $\rightarrow$ B: $K^{r_A} \bmod p$ ($X_O^1 = A$)
2. E\A $\leftarrow$ B: $(K^{r_A})^{r_B} \bmod p$

   2'. A $\leftarrow$ B/E: $(K^{r_A r_B})^{r_E} \bmod p$

   3'. A $\rightarrow$ B/E: $K^{r_B r_E} \bmod p$
3. E\A $\rightarrow$ B: $K^{r_B} \bmod p$ ($X_O^3 = E \neq X_O^1$)

Session key for *A* and *B*: *K*
Session key is not disclosed to *E*

It is interesting to reconsider the man-in-the-middle attack on Diffie-Hellman. From *B*'s viewpoint in this attack, X_O^1 (= *E*), i.e., the originator of the first message which is a *COM*, is the only entity who can compute the resultant key, and there is no other *COM* or *NCOM* to worry about. If a subsequent key-confirming message is delivered to *B* which shows someone (X_O^3) has got the key $K = g^{r_B r_E}$, then *B* can correctly conclude

that the entity X_O^3 is the only entity having K and the same entity as X_O^1. This kind of match is not provided in the Shamir's no-key protocol.

Man-in-the-middle attack against Diffie-Hellman protocol

1. A $\rightarrow$ B/E: g^{r_A}

   1'. E\A $\rightarrow$ B: g^{r_E} (X_O^1 = E)

   2'. E\A $\leftarrow$ B: g^{r_B}

2. A $\leftarrow$ B/E: g^{r_E}

Session key for A and B/E: $g^{r_A r_E}$

Session key for $E\backslash A$ and B: $g^{r_B r_E}$

Before we finish this section, we investigate a little more the relation between entity authentication and key authentication. First we define some terminology.

Definition 2-18. *We say that entity authentication and key authentication are* ***well combined*** *with each other in a protocol if the protocol satisfies*

- *entity authentication as defined in any one sense of the three definitions* ***EntityAuth1***, ***EntityAuth2*** *or* ***EntityAuth3****, and*
- *implicit or explicit key authentication in the sense of* ***ImplicitKeyAuth2*** *or* ***ExplicitKeyAuth2*** *respectively, and*
- *the identified entity is the same entity as the owner of the authenticated new session key.*

Looking at the **EntityAuth3** and **ImplicitKeyAuth2** or **ExplicitKeyAuth2**, we can easily derive the following conclusion.

Theorem 2-1. *If a protocol satisfies both* ***EntityAuth3*** *and* ***ImplicitKeyAuth2*** *or* ***ExplicitKeyAuth2*** *then both types of authentication are well combined in the protocol.*

Proof. We are going to prove that an identified entity A and the owner X of the new session key are the same entity. From the Definition 2-11 and Axiom 2-1,

$$\forall\ COM_i\ (X_O^i = A). \tag{1}$$

Also, by Definition 2-16 or Definition 2-17, we have

$$\exists X\ (\forall\ COM_i\ (X = X_O^i \wedge X\ !\ (\ni)\ K\ )), \text{ or} \tag{2}$$

$$\exists X\ (\forall\ COM_i\ (X = X_O^i \wedge X\ !\ \ni K\ )) \tag{3}$$

where A is the authentic identified entity and X the only owner of the new session key except B. The equation (1), and (2) or (3) lead to

$$\forall\ COM_i\ (A = X). \qquad (4)$$

If we suppose the negation of the conclusion, i.e., $A \neq X$, then this leads to a contradiction with the equation (4), and hence we can conclude that $A = X$. ■

2.2.4 Etiquette of Masquerade

Though it is very difficult for a principal B to find out what has actually happened over the communication channel from and to his peer principal, there are some rules of identity claim, which help B verify the authenticity of the other principal. More importantly, these rules are very useful for the construction of any possible attack against suspicious protocols with regard to the check items as identified in the definitions of entity authentication. The most general rule can be made as a form of an axiom: we call it *the axiom of loyalty*.

Axiom 2-2. [Axiom of loyalty] *Every entity involved in a particular authentication protocol should be loyal to his role including his mask.*

This axiom simply states that any authentic entity A should do his best to be authenticated by his peer entity, and any impersonator E would also do his best to be accepted as his disguised identity. Thus we do not have to consider any anomalous behaviour by the entities. The axiom, therefore, enables us to focus only on reasonable or meaningful attacks against a given protocol. This rather odd-looking axiom comes from a basic viewpoint from which we investigate a given protocol: *what kind of knowledge can be given to the verifier B after a "successful" run of the protocol*? This axiom is meant to capture our common sense view with regard to meaningful attacks, and thus may be difficult to formalize. Application of this axiom, therefore, will usually be done through careful derivation of some rules of inference described below.

Rules about masks (i.e., claimed identities) to be presented in this section are based on the above axiom. We present the first of them as follows.

Masking Rule 1. $$\frac{\ldots \not\sim ?/X \setminus A \not\sim B/?,\ X \neq A}{\forall X \in \boldsymbol{X}\ (\ldots — ?/X \setminus A — \ldots)}$$

Here $\boldsymbol{X}$ denotes the set of all the participating entities involved in a particular protocol run except the verifier B. This rule says that if a sender X ($\neq A$) in a protocol session is impersonating A to nominally B then all the entities but the verifier B involved in the session are claiming the identity A to the right. The leftmost entity may or may not be the entity A, and the rightmost entity is of course the verifier. The question mark as in ?/X\A simply means that we don't care about what identity is claimed by X to his leftThis kind of reasoning is based on an omniscient viewpoint. We take this viewpoint when we analyse the protocol to find any potential attack. Finding an attack against a given protocol is very much like finding a counterexample for a suspicious proposition to disprove it. Therefore, when constructing any attack, we are allowed to take any viewpoint and only the result will tell in the end.

Here is another rule about masks.

Masking Rule 2. $$\frac{?\setminus A \multimap B / E \setminus X \multimap B / ?}{X = A \ \vee \ X = E}$$

Here X is an identity claimed to the right by a man-in-the-middle attacker E. In this configuration E sees mask A and impersonates B to his left, and is putting on a mask to his right, i.e. to B. The only mask or identity which may be claimed by E to his right is A for impersonation or the true identity E. Any other identity, say C which is neither A nor E is not worth to be considered. This rule is justified because we are considering a particular session which E wants to compromise. We do not rule out that an attacker may produce multiple sessions to achieve his goal: the above inference rule is relevant to only a particular protocol run and not to multiple session attacks.

The last masking rule compliant with the axiom of loyalty is as follows.

Masking Rule 3. $$\frac{?\setminus A \multimap E / E \setminus X \multimap B / ?}{X = A}$$

When the attacker E sees the mask A to his left and puts its own mask E to apparently A, he has only one choice of the mask to his right, that is A. All these masking rules are frequently used to prove or disprove the goals of a given protocol, as will be demonstrated later.

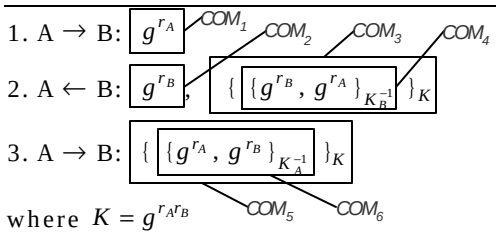
2.2.5 Protocol Analysis and Attack Construction

In this section, we demonstrate the usefulness of the new model as an analysis tool for authentication and key-establishment protocols, among which the STS and the Needham-Schroeder protocols are analysed here. The former protocol is proved to be secure while the latter disproved.

2.2.5.1 The Station-to-Station Protocol

We have already seen that the Diffie-Hellman protocol provides a nontrivial authentication feature which was not well captured so far. In this chapter, we derived a different definition of implicit key authentication by not allowing any extra feature like entity authentication to be involved (see Definition 2-16). This definition was in fact inspired by the Diffie-Hellman protocol and we proved in Lemma 2-1 that this protocol does satisfy implicit key authentication as defined in the definition. We show the usefulness of this slimmed down version of key authentication by using Lemma 2-1 to prove the following theorem about the STS protocol. In the following description of the protocol, K denotes the new session key between A and B. In this protocol there is only cryptographically operated messages, i.e., *COM*s.

Protocol 24. *Station-to-Station protocol*



Theorem 2-2. *The STS protocol provides the entity* B *the authentication of* A *as defined in Definition 2-11, i.e.,* ***EntityAuth3*** *and explicit key authentication as defined in Definition 2-17, i.e.,* ***ExplicitKeyAuth2****, which are well combined with each other. Namely,* B *can conclude that*

$$X_O^1 = X_O^5 = X_O^6 = A,$$

$$A \twoheadrightarrow B, \text{ and}$$

$$(A, B)! \ni K$$

Proof. The protocol has the same form as the variant Diffie-Hellman protocol described in Protocol 22, and so we can conclude directly the following using Lemma 2-1.

$$X_O^1 = X_O^5 \text{ and } (B,\ X_O^{1,5})! \ni K \qquad (1)$$

Here $X_O^{1,5}$ denotes X_O^1 who is also the same entity as X_O^5. We will show that

$$X_O^6 = X_O^5,$$

$$X_O^6 = A \text{ (i.e., } X_O^6 \ni K_A^{-1}) \text{ and}$$

$$A \twoheadrightarrow B$$

B knows that r_B is fresh and so is g^{r_B} too. This brings $X_O^6 \ni (K_A^{-1},\ g^{r_A},\ g^{r_B})$. Therefore $X_O^6 = A$ and $A \in X_S^1$ (that is, *A* has sent g^{r_A}). Now *B* knows that *A* has sent the following message.

$$\{\ \{g^{r_A},\ g^{r_B}\}_{K_A^{-1}}\ \}_{K'}$$

By the loyalty axiom, *A* has sent the g^{r_A}, received g^{r_B} and computed the key *K* which should be equal to $g^{r_A r_B}$. Thus *B* can conclude that K' is in fact *K*. This means $X_O^6 = X_O^5$. The fact that *A* has sent COM_6 means in turn that *A* has checked and verified COM_4 and COM_3 as described in the protocol according to the loyalty axiom again. If the message COM_4 were modified by an attacker *E* then it means that *E* has sent the following form.

$$\{\ \{g^{r_B},\ g^{r_A}\}_{K_E^{-1}}\ \}_K$$

But this is impossible because the attacker *E* is not $X_O^{1,5}$ (= *A*) who is the only entity in possession of the key *K* except the entity *B*. In other words, *E* does not know *K* and hence cannot construct the above message. Consequently, *A* has seen *B*'s signature in COM_4, which means that *A* is aware of *B* as his peer entity (i.e., $A \twoheadrightarrow B$). Thus we have proved that all the originators from *B*'s viewpoint are in fact the same entity, i.e., *A* and he is the only entity together with *B* who has the new session key *K*. Therefore, entity and key authentications are well combined with each other. ■

The STS protocol when accompanied by an additional authentic acknowledgement provides the reciprocal knowledge to *A* too; i.e. **EntityAuth3** and **ExplicitKeyAuth2**. Even without that kind of extra message, *A* can conclude the following after the protocol message 2.

$$X_O^2 = X_O^3 = X_O^4 = B,$$

$$(A, B)! \ni K, \text{ and}$$

if B receives the 3rd message then $B \twoheadrightarrow A$.

Only the awareness of peer entity is not explicitly guaranteed for A, and this can be completed when an authentic message exchange follows the protocol. However, it should be noted that if COM_4 is modified to include the identity of A in the signature then the awareness of peer entity is explicitly provided to A as well as B. Hence there is no point in talking about adding an additional protocol message to explicitly provide the feature.

2.2.5.2 Needham-Schroeder Public Key Protocol

The following is a simplified description of the Needham-Schroeder protocol [NeSc78].

Protocol 25. *Needham-Schroeder public-key protocol*

1. A $\rightarrow$ B: *BPubKey*$\{r_A, A\}$	(COM_1)
2. A $\leftarrow$ B: *APubKey*$\{r_A, r_B\}$	(COM_2)
3. A $\rightarrow$ B: *BPubKey*$\{r_B\}$	(COM_3)

Presenting our result of analysis of the protocol in advance, this protocol does not guarantee to B VPA (i.e., $A \twoheadrightarrow B$) although the first protocol message includes the identity A encrypted under the public encryption key of B, which guarantees A that B will be aware of A as his peer entity (i.e., $B \twoheadrightarrow A$). In this protocol, all messages are of the type *COM*.

Analysis from A's viewpoint

From COM_1 and COM_2, the entity A may conclude that B is alive somewhere, with his own mask put on to the left side, because the fresh nonce r_A returned in COM_2 could have been first retrieved only by B before re-encrypting using *APubKey*. And also, A can be sure that B must have seen the identity A after decryption of COM_1 with *BPriKey*. This means that B is aware of A as his peer entity. Also, again from the freshness of nonce r_A, A can be sure that the COM_2 is fresh too. Consequently, COM_2 guarantees to A the explicit possession of r_A and r_B by X_O^2. The use of *APubKey* by

X_O^2 for COM_2 trivially assures A as to X_O^2's awareness of A as the peer entity. Hence A has the following intermediate conclusions.

$$B \twoheadrightarrow A, \quad \text{B/B\?}, \quad X_O^2 \ni (r_A, r_B), \text{ and } \quad X_O^2 \twoheadrightarrow A. \tag{1}$$

Still there is no explicit evidence that X_O^2 is really B. It is interesting that in this protocol X_O^3 has made no claim about any identity through cryptographic operation. This lack of *cryptographic claim* of the proper identity (here B) leads to lack of VPA to the entity B as will be shown in the analysis from B's viewpoint. This lack of cryptographic claim of the relevant identity causes no guarantee of entity authentication not only to A but also to B.

The role of encryption used in the third message ($BPubKey\{r_B\}$) in the protocol is not so clear with regard to entity authentication. It only contributes to the confidentiality of r_B and hence it may be used as a key input for keeping the resulting session key secret. However, r_B need not be kept secret for secure key establishment because both entities will have the secret r_A in common for secure key computation provided that they have achieved successful entity authentication of each other. The value of r_B can contribute to the mutual key control even when it is not protected by any encryption.

Now, we investigate the possibility that $X_O^2 \neq B$. If someone who is not really B has been involved as X_O^2 in the successful run of this protocol, then this amounts to authentication failure according to our definition of entity authentication. We already know that X_O^2 has r_A which has been sent to B as an encrypted message COM_1. Thus X_O^2 must have received r_A as part of the first message or otherwise the second message as follows.

Case 1. 1. $\text{X\?} \rightarrow X_O^2 / X_O^2 : X_O^2 PubKey\{r_A, ?\}$, where $X_O^2 \neq B$ (2)

That is, X_O^2 has received the relevant message as the first protocol message with his own identity X_O^2 claimed to the left.

Case 2. 2. $X_O^2 \backslash X_O^2 \leftarrow$?/X: $X_O^2 PubKey\{r_A, r_B'\}$, (3)

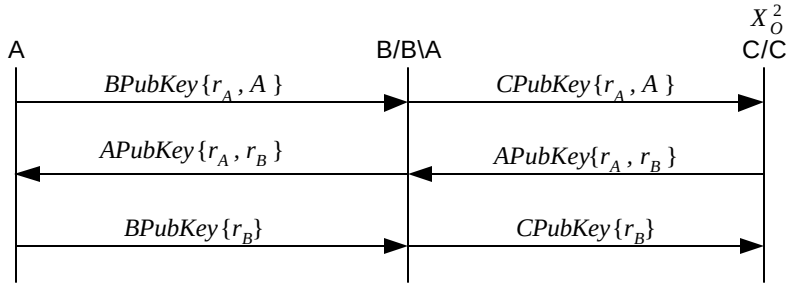
where $X_O^2 \neq B$ and $r_B' = r_B$ or not. *That is, X_O^2 has received the relevant message as the second protocol message with his own identity X_O^2 claimed to the right.*

We can use the above two cases to search for any possible attack which successfully violates VKP, one of the goals defined in **EntityAuth3**, namely, $X_O^2 \ni BPriKey$ or $X_O^2 = B$.

Attack construction for Case 1. In this scenario, we may reason that there must be B alive (who may be the entity X in (2)) somewhere to the left of X_O^2, decrypting r_A from COM_1 originated by A and then delivering to some entity, whom we may assume to be X_O^2. From the masking rule 3, B must have claimed A to X_O^2; hence we may construct the following attack configuration.

$$A \text{ — } B/B\backslash A \text{ — } X_O^2 / X_O^2 \qquad (4)$$

Based on this configuration, we may derive the following message flow, where we assumed $X_O^2 = C$ for some entity who is neither A nor B.



Note that in this attack construction the man-in-the-middle B can successfully imitate A to the third entity C because B can form any required message for the attack. He also gets the secret values r_A and r_B which have been generated by A and C respectively for the establishment of a secret session key. This attack was already described by Lowe [Lowe95] and discussed by Gollmann [Goll94]. It is quite interesting to note that inclusion of r_A in the third protocol message (i.e., 3. A $\rightarrow$ B: $BPubKey\{r_A, r_B\}$) does not improve the situation in this protocol although it is usually believed to be a prudent practice, and in fact complies with one of the four goals from Gollmann [Goll94] for authentication protocol, which will be discussed later in this chapter. The right fix for the protocol is to add the identity B to the second protocol message, providing VPA to B; hence a revised protocol will be as follows.

Protocol 26. *Revised Needham-Schroeder public-key protocol*

1. A $\rightarrow$ B: $BPubKey\{r_A, A\}$	(COM_1)
2. A $\leftarrow$ B: $APubKey\{r_A, r_B, B\}$	(COM_2)
3. A $\rightarrow$ B: r_B	(COM_3)

In this modified protocol, A can be sure that $X_O^2 = B$ and $B \twoheadrightarrow A$ while B can be sure that $X_O^1 = X_O^3 = A$ and $A \twoheadrightarrow B$. Therefore, this protocol satisfies all the goals specified in **EntityAuth3** perfectly. Note that the third message r_B, a plaintext, is counted not as a *NCOM* but as a *COM* message because it can only be returned after the required cryptographic operation, i.e., decryption of COM_2 using A's private key. As for key authentication, both A and B can believe that the resulting key computed using r_A and r_B is known only to both entities. If one of A and B deliberately discloses r_A to a third entity to play a man-in-the-middle attack, then the attack cannot succeed at all, which can be easily checked in a similar fashion to the above attack construction. Hence, in this revised protocol, both entity and key authentications are satisfied and well combined with each other.

The above generic form of the protocol can be implemented into a computationally effective protocol using hashing as follows.

Protocol 27. *More concrete form of the above protocol*

1. A $\rightarrow$ B: $\{r_A\}_{K_B}$, $h(r_A, A)$
2. A $\leftarrow$ B: $\{r_B\}_{K_A}$, $h(r_A, r_B, B)$
3. A $\rightarrow$ B: r_B

From the above analysis of the Needham-Schroeder protocol, we can make a nontrivial observation that an attacker can trick the entity B to accept a false identity, notwithstanding B appears to be in a better position than A in the sense that

- B has more cryptographic messages available than for A, and
- B is always assured about the completion of the protocol session whereas it is not the case for A.

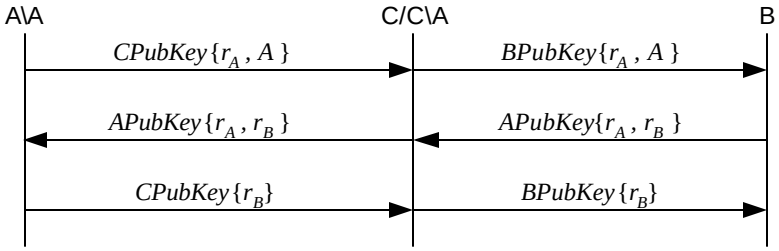
This result indicates that we may not be allowed to reason as follows: *the receiver of the last protocol message will be able to derive more information about entity/key authentication than the other entity*. We consider BAN analysis of this protocol below [BAN90].

$A \models B \models$ (r_B is a secret known only to A and B)

$B \models A \models$ (r_A is a secret known only to A and B)

$B \models A \models B \models$ (r_B is a secret known only to A and B)

Here, the BAN notation "A $\models$ something" means that "A believes something". BAN logic analysis clearly states that B is in a better position than A in the relevant evidences, which is in fact not true. We show below the attack derived from the above analysis with some modification regarding naming the relevant entities.



When comparing this attack diagram and the above belief of B derived by BAN logic, we can see that B's two beliefs are thoroughly violated by this attack. As for A's viewpoint, he has never heard of the name of B in this attack session. Here, r_B is known to three entities A, C and B. Hence the exclusivity of the possession of r_B described in the BAN analysis does not hold. In other words, the following belief of A is not correct.

$$A \models C \models (r_B \text{ is a secret known only to } A \text{ and } C)$$

The authentic entity C (from A's viewpoint) is not the maker of r_B; it is the third entity B. Furthermore, on the receipt by A of the second protocol message, C is still *not* in the possession of r_B, which was nevertheless claimed by the BAN analysis. This example gives us a lesson that any sophisticated analysis method based on formal logic might lead us to wrong results if it has not been built upon a sound interpretation and modelling of the object itself to be analysed, which is *authentication* in our case.

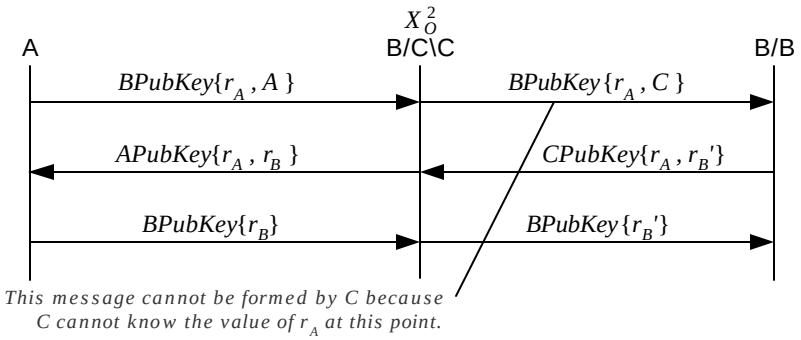
Attack construction for Case 2.

In this scenario, the relevant situation is that the entity B is alive (who may be the entity X in (2)) somewhere to the right of X_O^2, decrypting r_A from COM_1 originated by A and then delivering to some entity, whom we may assume to be X_O^2. Hence, by the masking rule 3, X_O^2 must have claimed the identity B to his left. Of course, B is putting on his

own mask because it is the only way for him to decrypt COM_1 and get the plaintext r_A. From this observation, we get to the following attack configuration.

$$\text{A} \text{—} \text{B}/X_O^2 \backslash X_O^2 \text{—} \text{B/B} \tag{5}$$

Based on this configuration, we may try the following message flow, where we assumed $X_O^2 = C$ for some entity who is neither A nor B, but an attacker.



The above diagram shows that this attack configuration cannot lead to a successful attack because the attacker C is not able to disclose the secret value of r_A when he intercepted the first message COM_1. Hence he cannot generate the required message $BPubKey\{r_A, C\}$.

Analysis from B' s viewpoint

COM_1 gives the entity B the following assurance.

$$X_O^1 \ (\exists)\ r_A, \ X_O^1 \twoheadrightarrow B, \text{ and } X_O^1 \backslash \text{A} \tag{6}$$

The first term means that the originator X_O^1 of COM_1 may or may not have r_A. If X_O^1 has just replayed COM_1 which was recorded from a previous session of the protocol between A and B, then X_O^1 will not be able to know the value of r_A. In the case that COM_1 has been delivered without modification from A through an attacker E to B, then X_O^1 is not E but A by our definition of the originator (see Definition 2-6 and Definition 2-7).

From COM_2 and COM_3, B knows that the entity A is alive somewhere now, having retrieved the fresh secret r_B. Therefore, B can conclude that A has received $APubKey\{r_A, r_B\}$ from somebody by claiming his own identity A to the right; that is,

$$?/\text{A}\backslash\text{A}. \tag{7}$$

The originator X_O^3 (= *A* or ≠ A) of the third message knows (and hence has) r_B and has encrypted it with the public key *BPubKey* of *B*. It means that he sees the mask or identity *B* as his peer entity. That is:

$$X_O^3 \ni r_B \quad \text{and} \quad X_O^3 \twoheadrightarrow B \tag{8}$$

Still, *B* cannot be sure about the following.

$$X_O^1 \ni APriKey \quad \text{and} \quad X_O^3 \ni APriKey, \tag{9}$$

or, in other words,

$$X_O^1 = X_O^3 = A. \tag{9'}$$

And worse, there is no explicit evidence that *A* is aware of *B* as the peer entity (VPA), although X_O^1 and X_O^3 are aware of *B*.

In fact, there seems to be no further information available to *B* in the protocol which may be exploited to derive the above fact. Now we have to investigate more deeply what may have happened behind the mask of *A*. Let us tackle a question: *how was the entity* X_O^3 *able to obtain the plaintext* r_B ? There might be two possibilities.

Case 1. 2. $X_O^3 \backslash X_O^3 \leftarrow$ B/X: $X_O^3 PubKey\{ r_A', r_B \}$, (10)

where $r_A' = r_A$ or not. *That is,* X_O^3 *has received the relevant message as the second protocol message with his own identity* X_O^3 *claimed to the right.*

Case 2. 3. X\? $\rightarrow X_O^3 / X_O^3$: $X_O^3 PubKey\{r_B\}$. (11)

That is, X_O^3 *has received the relevant message as the third protocol message with his own identity* X_O^3 *claimed to the left* (*and consequently* $X \twoheadrightarrow X_O^3$).

In a similar fashion as in the analysis for *A*'s viewpoint, we can search into any possible attack on the protocol. We omit the result here because Case 1 and 2 here lead respectively to the same results of Case 2 and Case 1 in the above analysis from the viewpoint of *A*.

What should be noted in this kind of attack construction methodology is that we do not have to resort to any brilliant intuition to find an attack on a given suspicious protocol. All the steps taken in the above procedures are quite straightforward. Moreover, this kind of methodology help us avoid any infeasible attack scenario, and hence focus only on a few possible attack scenarios.

2.2.6 Goals of Authentication and Key Establishment Protocols

In previous sections, we have defined two types of messages *COMs* and *NCOMs*, key confirmation, and entity/key authentication, and presented symbolic expression for them. In this section, we continue in this line and derive symbolic expressions for most of the goals arising when we talk about entity authentication or key establishment protocols.

A lot of work has been done to streamline the work of protocol analysis but not so much effort has been devoted to clearly define basic features or goals themselves regarding security protocols. Security features or goals as described in common English are easy to understand but may have a fatal weakness in the sense that it may lead to ambiguity, confusion and some redundancy. Consequently, with that approach, we cannot skillfully handle all the goals and their relations, which will limit us severely in distinguishing between elementary features and their tricky combinations. This kind of unclear and dubious approach toward security goals will often result in poorly designed protocols: *computationally expensive and nevertheless buggy*.

As we have already seen in previous sections, there is a remarkable similarity between entity authentication and key authentication. The former is concerned with verifying possession of an already established long-term secret authentic key of a particular identity-claiming principal, and the latter with generation of a new secret key between two peer principals. In any authentication and key establishment protocol, these two types of authentication must be well combined with each other (see Definition 2-18).

Many authors have addressed a range of goals relating to two types of authentication and key establishment protocols. There is a dichotomy to classify the goals: *intensional goals* and *extensional* goals [Boyd97]. The general principle is whether a goal specifies the details of how the protocol is executed (intensional) or what each principal gains from the protocol (extensional). Boyd [Boyd97] has stated:

> *While many authors have given extensional goals for key establishment only intensional goals seem to have been proposed for entity authentication. … When it comes to entity authentication authors seem to*

have had a harder time deciding what should be an extensional goal, … most resort to intensional specifications. One reason for this may be that it is difficult to be clear on the purpose of entity authentication in absence of key establishment.

We believe the *asymmetry* in the degrees of difficulty in deriving extensional goals for entity authentication and key establishment arises because most authors try to define entity authentication using too high level abstract concepts whereas building the definition of key authentication upon the specific concept of a particular session key. This is clearly an unfair attitude and it would be better to place the key itself at the center of definition of entity authentication as well as key authentication. Then deriving extensional goals for authentication protocol will be free from ambiguity and confusion, and as easy as for key establishment protocol. In fact, we regard some goals as extensional, which were categorized as intensional by Boyd [Boyd97].

Now we review several goals put forward in the previous literature regarding authentication and key establishment protocols, and will try to translate them into our own model and symbolic language. Thereafter, we summarize most of the discussed properties or goals in this paper into a structured hierarchy.

2.2.6.1 Gollmann

The first set of goals now to be presented are from Gollmann [Goll94].

Gollmann1: Authenticated Session Key *The protocol shall establish a fresh session key, known only to the participants in the session and some Trusted Third Parties.*

Translation $[\,(A, B)! \ni K\,] \wedge [\,(A, B) \models \text{fresh}(K)\,]$

Two protocol principals A and B only are in possession of the session key, and they know (believe) the key K is fresh.

This is an extensional goal for key establishment. The original description by Gollmann seems to assume entity authentication implicitly or, at least, is not so clear about what the *participants* means. The explicit stipulation of freshness property of the session key in the above description is rather redundant in the sense that the exclusive possession of the key should be based on the freshness of the key anyway. What is the use of

freshness other than to guarantee the *exclusive possession* of the session key? In other words, how can a principal be assured of the exclusive possession of the key only by his peer principal and himself? The concept of freshness in key establishment protocols should only be understood as a method to achieve the security and authenticity of the relevant session key. Namely, the freshness of the session key is not an ultimate goal but just an intermediate step toward secure and authentic establishment of the key. Someone may argue that "freshness" guarantees the key *K* is a new key, and therefore is worth being set as one of goals for key establishment protocols. But, then, what is the use of the property of "being new"? The gist here is that the principals *A* and *B* cannot be sure about their exclusive possession of a particular session key *K* as far as it is used repetitively or for too much a long time. Message freshness is only a cryptographic vehicle to carry the verifier *B* to the confidence that the proving entity is truly in possession of the relevant key (see Protocol 16 and the corresponding description) or that the newly generated session key is exclusively possessed by *B* himself and the other principal.

Gollmann2: Authenticated message *A cryptographic key associated with A was used in a message received by B during the protocol run* [***Gollmann2-1***]. *The protocol run is defined by B's challenge or a current timestamp* [***Gollmann2-2***].

Translation

Gollmann2-1:

$\exists COM_i$ (X_O^i ∋ APriKey) *for asymmetric key protocol*

$\exists COM_i$ (X_O^i ∋ K_{AB} ∧ $X_O^i \neq$ B) *for symmetric key protocol*

Several cryptographic methods for the verifier B to verify this proposition are as follows.

Gollmann2-2:

For asymmetric key protocols:

[A ← B: r_B] *and then* [A → B: *APriKey*{r_B, …}], *or*

[A → B: *APriKey*{TS_A, …}]

For symmetric key protocols:

[A ← B: r_B] *and then* [A → B: K_{AB} {r_B, *B*, …}], or

[A → B: K_{AB} {TS_A, *B*, …}]

Boyd classified this as an intensional goal[Boyd97], but we see that this goal stated by Gollmann, in fact, is composed of two different levels of intensional goals. The first part, **Gollmann2-1** seems to be more general than the latter **Gollmann2-2** which describes two different way to achieve the former goal. This goal is, however, not extensional because it basically rules out any protocol using encryption of the challenge, for example [A ← B: $APubKey\{r_B\}$]. That is, entity authentication as in **Gollmann2** is only concerned with particular protocols using the claimant's signature.

Gollmann3: Authenticated Protocol Run *A cryptographic key associated with A was used during the protocol run. It is not necessary for B to receive a message where this key had been used. The protocol run is defined by A's challenge or a current timestamp.*

Translation

$\exists X \in \boldsymbol{X}\ (X \ni APriKey)$ *for asymmetric key protocol*

$\exists X \in \boldsymbol{X}\ (X \ni K_{AB} \wedge X \neq B)$ *for symmetric key protocol*

where $\boldsymbol{X}$ *denotes the set of all participants except B involved in the current protocol run.*

The property described in this goal may also be called "aliveness" of the entity to be authenticated, as Gollmann himself does in conclusion [Goll94]. However, it is quite striking that Lowe uses this term in an entirely different sense as will be described later. This goal is more general than the previous one. The following example protocol and an attack will help make this subtle difference clear.

Protocol 28.

1. A ← B: $APubKey\{r_B\}$
2. A → B: r_B

Attack against Protocol 28

1. A ← B: $APubKey\{r_B\}$
   1'. C ← B/A: $CPubKey\{r_B\}$ *A is impersonating B to C.*
2. C → B/A\A → B: r_B

As the above attack shows, *B* cannot safely conclude that the second message has been "said" by *A* because it may have been said by somebody other than *A*. Nevertheless, *B* can be sure that *A* is "alive" on this protocol run because only *A* can decrypt the first

message to get the nonce r_B. In contrast, the following protocol assures B that A is not only "alive" but has also "said" the second message.

Protocol 29.

1. A ← B: r_B
2. A → B: $APriKey\{r_B\}$

The aliveness or **Gollmann3** is satisfied by both Protocol 28 and Protocol 29, but **Gollmann2** is only satisfied by Protocol 29. Thus, we are allowed to regard **Gollmann2** as a special case of **Gollmann3** because the former only covers signature-based cryptographic protocol while the latter covers another cryptographic primitive, i.e. encryption too.

Now, is **Gollmann3** an intensional goal or extensional one? It is evidently one step closer to being an extensional goal than **Gollmann2**. One may, however, argue that it is not extensional in the sense that it excludes some type of authentication protocols; basically the same reason has caused Boyd to conclude it to be intensional. Such an example would be the following protocol which is similar to the SKEME protocol by Krawczyk [Kraw96] but modified to be more robust.

Protocol 30.

1. A → B: $BPubKey\{A, r_A\}$
2. A ← B: $APubKey\{B, r_B\}$

where the session key K should be a function of r_A and r_B.

At the end of the protocol, neither of the principals can be sure about the aliveness of the other principal. And for this reason, one may argue that this protocol does not provide entity authentication at all and should only be thought of as a key authentication protocol. However, this is a fastidious attitude. Any security protocol is not for itself but a subsequent secure message exchange — immediate or some time later. After the second message, if one of A and B receives an encrypted message using the right session key then he can conclude that the other party is clearly alive. In this sense, the protocol seems to be said by some authors to provide *implicit* authentication. We suggest the term "pending authentication" to be used for this type of entity authentication because the authenticity of the claimed identity is not verified pending a relevant subsequent message. *If we consider some relevant subsequent messages for*

every pending authentication protocol when we analyse them, then the goal as in **Gollmann3** may be stated to be extensional. Likewise, our own definitions **EntityAuth 0~3** in Section 2.2.2 are, in fact, all extensional goals in this sense.

Gollmann has also has put forward a very strict goal as follows, the arguability of which was also described by Gollmann himself.

Gollmann4: Authenticated Request for Service *The origin of all messages in the protocol has to be authenticated.*

Translation $\forall COM_i\ (X_O^i \ni APriKey)\ and\ \forall NCOM_j\ (X_O^j \ni APriKey)$

We regard this goal as an extensional one because it does not explicitly require any particular message to be exchanged for the protocol. Here we can see there is a very close similarity between this goal and a previous one, **Gollmann2**. In fact, **Gollmann4** appears to be just an extension of **Gollmann2** over the set of all the received message to the verifier in the protocol run.

It is interesting to compare this goal with subgoals VKP and VMA of **EntityAuth3**, which are described below again.

$$\forall COM_i\ (X_O^i \ni APriKey) \text{ and } \forall NCOM_j\ (A \in X_S^j)$$

The only difference between these two is that **Gollmann4** requires every messages including *NCOMs* as well as *COMs* to be authenticated whereas **EntityAuth3** requires authenticity only for *COMs*, and a kind of awareness or acknowledgement by the claimant for *NCOMs*. We of course do not exclude the possibility that such a strong form of translation might have not been implied in **Gollmann4** and in fact **EntityAuth3** may form a close translation of **Gollmann4**. In that case, however, it would be just one more example inevitable ambiguity or confusion without an appropriate model and a relevant description language for authentication.

We have so far investigated the four goals put forward by Gollmann. Quite interestingly, Gollmann seems to believe that his first goal concerning authenticated session key is essential, or at least a critical "salt", for entity authentication; he states as follows.

> *It may not be coincidental that the only pure authentication protocol in* [*BAN90*] *is also the one which was broken.*

Thus, in Gollmann's architecture of protocol goals, key authentication seems to

constitute a set of several goals for entity authentication. This is, however, rather misleading because key authentication in this sense already implies entity authentication because **Gollmann1** states symbolically "$(A, B)! \ni K \wedge (A, B) \models \text{fresh}(K)$" as presented earlier. Hence, it is hard to find where we may cut the chain of cause and effect:

… ® *key authentication* ® *entity authentication* ® *key authentication* ® …

To be fair to the author, he states in conclusion that key exchange is not a prerequisite for authentication and ISO has standardized "pure" authentication protocols. Therefore, he appears to believe that key authentication is not a prerequisite but, at least, a robust tool making entity authentication protocol more secure. Anyway, **Gollmann1** seems to provide little tools for design and analysis of authentication protocols.

Gollmann2 is only a special case of **Gollmann3**. **Gollmann4** states that every message received by the verifier *B* should really *originate* from the claimant *A*, which is a rather strong requirement when compared to our requirement *Verification of Message Awareness* (VMA) in **EntityAuth3**. It is also interesting that Gollmann does not offer any similar goal to our *Verification of Peer Entity Awareness* (VPA). Consider the following protocol.

Protocol 31.

1. A $\rightarrow$ B: $APriKey\{TS_A, r_A\}$
2. A $\leftarrow$ B: $BPriKey\{r_B, r_A\}$
3. A $\rightarrow$ B: $APriKey\{r_A, r_B\}$

This example protocol clearly satisfies all of the goals **Gollmann2~4**. For example, from the viewpoint of *B*, the cryptographic key associated with *A*, i.e. *APriKey*, is applied to the first and the third messages, together with *B*'s challenge r_B or timestamp TS_A (**Gollmann2** and **Gollmann3**). Message authentication is also trivially provided (**Gollmann4**). Despite all these, this protocol is subtly weak to the following attack.

Attack on Protocol 31

1. A $\rightarrow$ E: $APriKey\{TS_A, r_A\}$
   1'. E\A $\rightarrow$ B: $APriKey\{TS_A, r_A\}$ — *E* impersonates *A* to *B*.
2. E\A $\leftarrow$ B: $BPriKey\{r_B, r_A\}$
   2'. A $\leftarrow$ E: $EPriKey\{r_B, r_A\}$
3. A $\rightarrow$ E: $APriKey\{r_A, r_B\}$
   3'. E\A $\rightarrow$ B: $APriKey\{r_A, r_B\}$

At the end of the attack, *B* believes that he has exchanged mutual authentication together with *A*, but actually it is *E* that is regarded as *A* to *B*. Furthermore, the other entity *A* believes that he has completed the authentication protocol with *E*; he has never heard of *B* in this protocol run. Clearly this attack violates the common sense of entity authentication no matter how significant it may be. One may argue that key establishment (**Gollmann1**) is not provided in this protocol, which would be the excuse for the weakness. This kind of diagnosis would hardly be considered as an appropriate one. The direct cause of the weakness comes from the lack of *Verification of Peer Entity Awareness* (VPA). The following modified protocol has no such a weakness and is *more effective* in terms of computation.

Protocol 32.

1. A $\rightarrow$ B: r_A
2. A $\leftarrow$ B: r_B , *BPriKey*{ *A*, r_A , r_B }
3. A $\rightarrow$ B: *APriKey*{ *B*, r_A , r_B }

Here note that the two cryptographically operated messages (*COMs*) using private keys can be implemented by signing after hashing the relevant messages. There are two *NCOMs* (r_A and r_B) and two *COMs* (*BPriKey*{ *A*, r_A , r_B } and *APriKey*{ *B*, r_A , r_B }). All the required properties such as VKP, VPA and VMA are provided in this protocol. For example, r_B in the signature of *B* guarantees to *A* that *B* is aware of the value r_B , which is the only *NCOM* from *B* to *A*, thus satisfying VMA. The inclusion of the other entity's identity in the signatures makes it clear of whom each entity is aware as his peer entity; thus VPA is satisfied for both entities. The private keys of each entity was applied to the relevant random challenge values, providing VKP to the other entity.

Design of secure authentication protocol has long been believed to be very difficult, and possibly expensive. Therefore, many authors seem to recommend a sort of balanced trade-off between the security requirements and protocol strength. We do not entirely agree with this trend. There is no reason for us to avoid providing a very strong authentication protocol even for an application in which the security requirement is not so strict. If we are able to identify most of the crucial elements or properties inherent in the notion of entity authentication, then it would not be so expensive to design a perfect protocol. In other words, it would be rather questionable to ask such a question as to

whether a given attack really is an attack before fixing any known weakness in the corresponding protocol when the required remedy entails no cost at all.

2.2.6.2 SVO

Another set of six "Generic Formal Goals" was introduced by Syverson and van Oorschot [vanO93], [SyvO94], which forms a ground work for their SVO logic. In the following, we list all of them in English and then translate them into our symbolic equivalents, where we swapped the original roles of *A* and *B* which appeared in their papers. It should be noticed in advance that the following translations of the SVO goal can not be exact since there is a significant difference between SVO model and ours with regard to the interpretation of the notion of "possession" (see the paragraphs describing **SVO5** and **SVO6** below).

SVO1: Far-end Operative *B believes A says something.*

Translation

$\exists\, COM_i\ (X_O^i \ni \text{APriKey})$ *for asymmetric key protocol*

$\exists\, COM_i\ (X_O^i \ni K_{AB} \wedge X_O^i \neq \text{B})$ *for symmetric key protocol*

SVO2: Targeted Entity Authentication *B believes A replied to a specific challenge chosen and sent to A by B.*

Translation

$[\ \text{A} \leftarrow \text{B}:\ r_B\ ]$ and then $[\ \text{A} \rightarrow \text{B}:\ APriKey\{r_B, \ldots\}\ ]$ *for asymmetric key protocol*

$[\ \text{A} \leftarrow \text{B}:\ r_B\ ]$ and then $[\ \text{A} \rightarrow \text{B}:\ K_{AB}\{r_B, B, \ldots\}\ ]$ *for symmetric key protocol*

SVO3: Secure Key Establishment *B believes a key K is shared with no party other than possibly party A.*

Translation $(B, A)!\ (\ni)\, K$

SVO4: Key Confirmation *B believes a key K is shared with party A alone, and that A has provided evidence of knowledge of the key to A.*

Translation $(B, A)!\ (\ni)\, K \ \wedge\ A \ni K$

SVO5: Key Freshness *B believes a key K is fresh.*

Translation $fresh(K)$

SVO6: Mutual Belief in Shared Secrets *B believes that party A believes K is an unconfirmed secret suitable for use with B.*

Translation $A \models (A, B)! (\exists) K$

The first goal **SVO1** corresponds to **Gollmann2** and means the aliveness of *A* in the current protocol run. **SVO2** seems to be a special case of **Gollmann2** where the protocol run is defined by *B*'s random challenge , not by a timestamp. It seems that both preclude the encryption-based authentication. SVO's description [vanO93] regarding these two goals is quite ambiguous, especially for **SVO1**.

> *Inherent* [*in* ***SVO1***] *is the fact that the identity of A has been corroborated, but it is unclear who A intended to convey the message to. … It* [***SVO2***] *provides authentication of A to B in the sense that the response is from a corroborated operational entity, and is targeted in response to a* (*preferably fresh*) *challenge from A. That A's formula is specifically in response to B's challenge ties A's reply to the protocol run B is executing.*

Here we might suppose that **SVO1** does not provide VPA as defined in Definition 2-11 whereas **SVO2** does; but it seems not the case because **SVO2** only says that *A*'s response should be tied to a *specific challenge* from *B*. Therefore it may still happen that *A* has replied to the specific fresh challenge chosen by *B* but nevertheless *A* intended to convey the response to some other entity, say *E*, than *A*. In other words, *A* is aware of not *B* but *E* as his peer entity. This means that, unlike SVO's intention, **SVO2** is still not free from "lack of the verification of the targeted identity of the response" which **SVO1** lacks as well. This is the first suspicious point. The second one is a little bit subtle but significant. Without a response being tied to *B*' s own fresh challenge, how can *B* be assured about *A*' s aliveness? One may argue that **SVO1** is intended to address not only the authentication protocols based on random challenge but also those based on time, which seems to be the only reason for **SVO1**.

SVO3 corresponds to implicit key authentication in the sense of **ImplicitKeyAuth1** while **SVO4** is different from the usual key confirmation as in **KeyConf1** but is essentially the same as explicit key authentication in **ExplicitKeyAuth1**. To be precise, it should be pointed out that the word or operator "has" in SVO logic is not the same as

"∋" in our setting. In SVO terminology, "$B \models A$ *has* K" can be satisfied even without the actual possession of K by A [vanO93]. In our notation, "$A \ni K$" is corresponds to not "$B \models A$ *has* K" but "$B \models$ confirm(K)" in SVO logic. In short, when we say "$A \ni K$" it means that B has *verified A's possession* (or the ability of possession) of K. It does not address, for example, claimed possession.

SVO5 is put forward as not a necessary but a desirable property or goal of key establishment protocols. This viewpoint is quite different from ours because we regard the freshness property as a necessary condition to derive any conclusion about possession of something secret by a principal. In other words, for the verifier B to be able to derive a conclusion saying some principal X has a secret data K (i.e., $X \ni K$), B always requires some proper freshness as a necessary condition. Let's have a look at the following example protocol.

Protocol 33.

A: $K = (g^b)^a = g^{ab}$

B: $K = (g^a)^b = g^{ab}$

1. A $\rightarrow$ B: $\{m\}_K$

where (a, g^a) and (b, g^b) are certified long-term asymmetric key pairs of A and B, respectively, and m is a recognizable message to B.

Here no freshness is provided in this no-pass key establishment protocol. In our model of authentication, after B's receipt of the first message, B cannot conclude that A is the only entity who can have the key K because there is no freshness in the key. On the other hand, according to SVO's analysis, B can believe that the resulting key is secret to only A and B, as insinuated in the proof of the Goss protocol in [vanO93].

SVO6 is the most difficult property for translation because we do not require the notion of mutual belief between two principals in our model. Furthermore, since our notion of possession requires a proper freshness as a prerequisite, our model does not provide any room for the notion. In other words, in our model, for B to infer that $A \models (A, B)!\ (\ni)\ K$, it is necessary for B to be assured that A believes in the freshness of A's own key input, say a random nonce r_A. This contradicts our model in which each principal can only believe the freshness of his own random nonce, and does not require any mutual belief about freshness to derive a proper conclusion. This is an example showing that there is a big difference between existing models and ours.

The first SVO analysis by van Oorschot [vanO93] of the STS protocol is flawed because the following weakened variation also satisfies all the intermediate steps to get to the six goals of SVO but in fact does not satisfy **SVO4** Key Confirmation and **SVO6** Mutual Belief in Shared Secrets.

Protocol 34. *Weakened STS protocol*

1. A → B: $\boxed{g^{r_A}}^{COM_1}$
2. A ← B: $\boxed{g^{r_B}}^{COM_2}$, $\boxed{\{g^{r_B}, g^{r_A}\}_{K_B^{-1}}}^{COM_3}$, $\boxed{\{g^{r_B}, g^{r_A}\}_{K}}^{COM_4}$
3. A → B: $\boxed{\{g^{r_B}, g^{r_A}\}_{K_A^{-1}}}^{COM_5}$, $\boxed{\{g^{r_A}, g^{r_B}\}_{K}}^{COM_6}$

where $K = g^{r_A r_B}$

Note that in the modified protocol an attacker E can play a man-in-the-middle resulting in the following wrong beliefs.

- Attack construction A—E/E\A—B leads to

  [**SVO4**] $A \models ((A,E)! \ni K \wedge E \ni K)$

  [**SVO6**] $A \models E \models (E,A)! (\ni) K$.

- Attack construction A—B/E\E— B leads to

  [**SVO4**] $B \models ((B,E)! \ni K \wedge E \ni K)$

  [**SVO6**] $B \models E \models (E,B)! (\ni) K$.

Here we do not detail our steps which, complying with the logical steps of [vanO93], finally lead to this erroneous belief. The basic reason of SVO's mistake is that their logic, at first, could not capture an important property which arises from the subtle difference between the original STS protocol and its weakened version. In their later work, they fixed this problem by using the concrete message form like $\{\{g^{r_B}, g^{r_A}\}_{K_B^{-1}}\}_K$ of the STS protocol directly in their logical steps [SyvO94]. This kind of problem arises since existing models of authentication including SVO are not well structured to capture two different aspects of authentications: *entity authentication* and *key authentication*. These two different types of authentication are just intermingled in the existing models, and hence the important or critical link between them cannot be well captured. In our model, we presented a pure *key authentication* which is not tainted by entity

authentication. The link between them is provided by the notion of "well-combinedness" (see Definition 2-18). Hence, our separation of these two aspects of authentication and our explanation of how they are linked ensures that the weakness identified cannot be missed. More concretely, our model basically requires *A* to make sure that COM_2 , COM_3 and COM_4 originated from the same principal, i.e., $X_O^2 = X_O^3 = X_O^4$, which results in immediate detection of the weakness in the modified protocol.

2.2.6.3 Lowe

We investigate only one goal, "aliveness" among the goals described by Lowe [Lowe97] because his usage of the term "aliveness" is quite different from that of others as mentioned earlier in this chapter.

Lowe1: Aliveness *Whenever B completes a run of the protocol, apparently with A, then A has previously been running the protocol.*

Translation

$\exists COM_i\ (X_M^i \ni APriKey)$ *for asymmetric key protocol*

$\exists COM_i\ (X_M^i \ni K_{AB} \wedge X_O^i \neq \mathrm{B})$ *for symmetric key protocol*

In the above translation, note that X_M^i is used in place of X_O^i because Lowe's aliveness, strangely enough, does not mean any currentness, which is very exceptional usage of the term aliveness in the light of both our common sense and the usage of the term in any other related work (for instance, see [Goll94, vanO93]). X_M^i denotes the maker of COM_i, which may or may not be the same entity as X_O^i. Therefore note that Lowe's aliveness is still satisfied even when X_O^i has replayed COM_i which was actually made and used by X_M^i (= *A*) in a previous run of the protocol. Lowe's aliveness should not be taken to mean any similar meaning of "far-end operativeness" or "aliveness" from other works.

This kind of usage of terminology is questionable for it exacerbates the difficulty of exact communication. Aliveness "at sometime but not now" can never be interpreted as true aliveness! It is no more alive now. It is trivially true that everything is always alive

when it occurs. Therefore, if the word "aliveness" is to have any reasonable meaning, then it should be interpreted as the aliveness "now".

2.2.6.4 Hierarchy of Authentication and Key Establishment Goals

Investigating several key concepts or goals for entity authentication and key authentication, we now summarize the goals as we defined in this paper into a hierarchical structure, as shown in the following Figure 2-2. In this diagram, *freshness* means freshness of the relevant cryptographic messages while *exclusiveness* means a principal's exclusive possession of the relevant keys.

We may think of entity authentication and key authentication as "long-term key" authentication and "session-key" authentication, respectively. This is a feasible reasoning because we can consider each authentication in terms of its relevant key: *long-term authentication key* or *short-term session key*. When we add the concept of the principal *identity* to the definition of key authentication, key authentication can no longer be free from entity authentication. This is because such a definition of key authentication in turn leads us finally to the concept of entity authentication. In other words, even though there is no *peer-entity awareness* required in the existing definition of key authentication, the expression "only the entity *A*" (see Definition 2-12 and Definition 2-15) requires some proper cryptographic guarantee, which corresponds to entity authentication at least in a weak sense.

The most important difference between entity authentication and key authentication is that exclusiveness of key possession is an "a priori condition" in the former whereas it is "a goal to be achieved" in the latter. Also the property "freshness" is positioned at the lowest level of the hierarchy, which means it is an essential condition to achieve other goals as opposed to existing viewpoints that key freshness is an important goal for key establishment protocols.

We believe that our new structure of goals and their relations is a clear and well-streamlined guideline for design and analysis of generic level protocols.

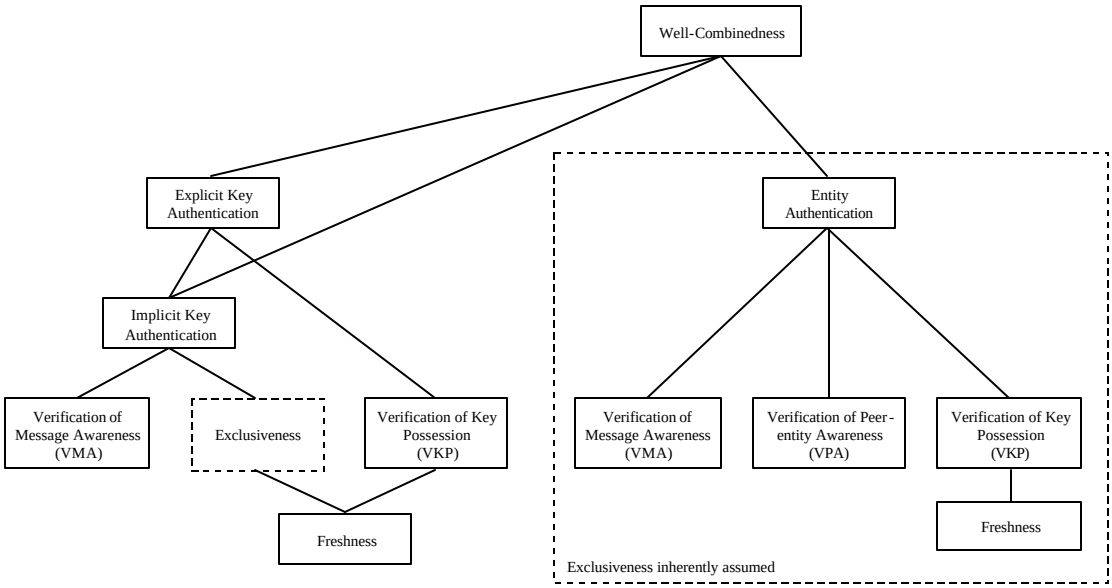


Figure 2-2: Hierarchy of Authentication and Key Establishment Goals

2.3 Summary

In this chapter, two methods for design and analysis of authentication and key establishment protocols were developed. First, a practical and simple classification scheme for authentication protocols was devised using abstract fundamental protocol elements which we call "cryptographic particles" in this chapter. We have demonstrated the practical use of the scheme with a number of examples. The classification can be useful in both design and analysis of protocols.

- Protocol analysis can be facilitated by identifying the class in which a particular protocol lies and then comparing its security with other protocols in the same class whose security is well understood.
- Protocol design can proceed by identifying the protocol requirements and using these to identify the desirable class of protocol to be used in the application.

Both these uses of the classification technique allow for systematic re-use of previous analysis and design experience, which aids in the move towards an engineering approach to protocol design and analysis.

Secondly, a new set of authentication model and protocol analysis methods were developed, thereby reviewing existing definitions and goals with regard to entity

authentication and key authentication. It seems that if we succeed in achieving a sufficiently clear and comprehensive model of entity and key authentications, then all that we need to prove/disprove security of a given authentication protocol will be just a few basic mathematical proof techniques such as the proof by contradiction. The new model in this chapter is such a model, which is demonstrated with a few protocols. This clear and modular way of modelling authentication leads to an entirely new definition of key authentication, which has been unduly tied to the concept of entity authentication in previous works. This new definition of key authentication provides us with more insight to key establishment protocols, which is illustrated with comparison of Shamir's no-key protocol and the Diffie-Hellman key agreement protocol. The new definition of key authentication also enables a very modular analysis of protocols, which was demonstrated with the analysis of the Diffie-Hellman protocol and the STS protocol.

Chapter 3.

WAKE Protocols

Design of public-key security protocols for wireless mobile communications requires considerable effort to satisfy two goals. The security protocols need to be (1) flexible and comprehensive to accommodate potentially complex public key infrastructure requirements due to roaming; and (2) computationally effective for mobile terminals but nonetheless fully functional for future mobile communications security requirements. We propose a suite of generic protocols meeting the first goal, and also a set of instance protocols which are compliant with the generic versions and satisfy the second goal.

The purposes of this chapter are to (1) propose a new generic protocol based on a different authentication scheme from that of the STS protocol which all the proposed protocols for UMTS and IMT2000 are based on; (2) analyse the designed generic WAKE protocol using the analysis methods developed in Section 2.2; and (3) propose some instance protocols based on the proposed generic protocols, having comparable performance with one of the UMTS candidate protocols.

3.1 Generic Protocol Design

Experience has shown that design and analysis of security protocols is a quite error prone activity. At least to some degree, this stems from the practice of describing protocols in detailed notations describing all the complexity of the algorithms and parameters. This practice does not allow for a simple understanding of the structure and purpose of each protocol message. When converted into a more abstract description, a protocol sometimes reveals a new feature, a redundancy, or a weakness which could not be seen in its detailed notations. The most important reason for starting the protocol

design in a general description, however, is that a generally described protocol can be implemented in many possible instance protocols. Thus, we do not need to go back to the starting point when an instance protocol designed using mathematical notations has been found to have a weakness. Furthermore, we can eventually obtain more than one instance protocol, enabling a flexible application of the security protocol to different security requirements.

In this section, we describe a generic WAKE protocol satisfying the goals as specified in Section 1.2. We include a number of protocol variants which satisfy different conditions in terms of the availability of certified copies of public keys.

3.1.1 Public Key Infrastructure for Future Mobile Communications

In any communication service, a well established public key infrastructure (PKI) would be the pivot on which WAKE protocols may run successfully. Any communicating principals needing to authenticate and derive a secure session key between them should be able to obtain an authentic copy of the relevant public key certificates of each other. The unique features of mobile communication systems such as roaming users, limited capabilities of mobile terminals and scarce radio bandwidths add up to the need to design flexible and scalable PKIs and hence the WAKE protocols as well.

The following Figure 3-1 shows main role players involved in the public key based security architecture of future mobile communications — the user, the network operator (NO), the service provider (SP), the value added service provider (VASP) and the trusted third parties (TTPs) — as well as the interfaces over which WAKE protocols will be run. The user and the SP uses and sells mobile communication service, respectively, using the network facilities of the NO. The VASP sells value added services to the user.

TTPs will serve as clearing houses with regard to all the required trust relations for WAKE protocol executions between the user and the NO/VASP, both on an on-line and off-line basis. The user-to-VASP interfaces is a logical interface to be established through NO networks, and has some interesting issues such as a rather strict requirement for non-repudiation services for electronic payment.

The user and the NO/VASP need to authenticate each other and establish a shared secret session key to encrypt the subsequent message exchange. This is achieved through execution of appropriate WAKE protocols.

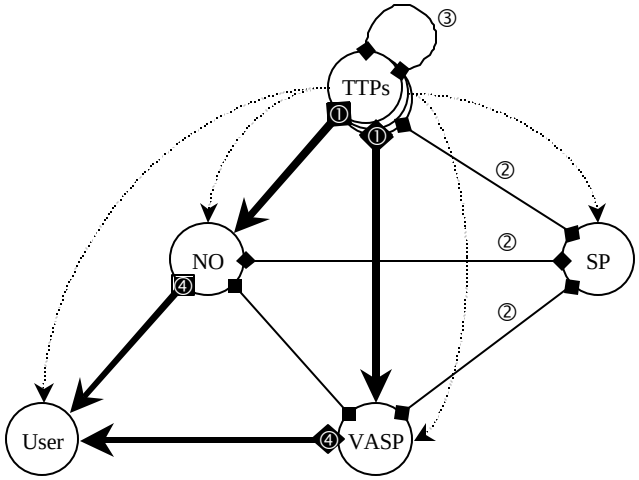


Figure 3-1: The flows of certificates over the interfaces between entities in future mobile communications

Notations and their meanings in Figure 3-1:

- **Thick line** means that the WAKE protocol is applied to the corresponding interface.
- **Thin line** means that the WAKE protocol is *not* applied to the corresponding interface.
- Any **arrow** means that the transmission of the relevant certificates is required for the execution of the WAKE protocol.
- Any **square** in black means that the transmission of the relevant certificates is not necessarily required for the execution of the WAKE protocols and depends on other factors.
- Any **dotted curve** means the initialisation or update of the certificate of the relevant entity through the certificate management service of the TTPs.

Comments for Figure 3-1:

①: When required by some operational reasons, the NO/VASP may submit their own public keys or certificates to be certified by a relevant TTP.

②: It may be preferable in some operational environments for the NO/VASP to access the user's SP for clearing certificates during the execution of the wireless authentication/key establishment protocol. In that case the SP may or may not require access to the TTP to clear the certificates.

③: According to operational environments, TTPs may have to exchange their signed certificates of the user, NO/SP or VASP to fill in the gap between the pieces of the required trust chain.

④: Under some operational conditions, the certificate of the user may be delivered to the NO/VASP not from the user but from the relevant TTP or the user's SP.

As described in the above comments, there would be a lot of different operational environments and thus PKIs and the WAKE protocols for future mobile systems should be sufficiently flexible in their architectures or mechanisms.

The following notations or *generic vocabularies* are used to describe the generic protocol design.

- A, B: the identities of the two entities who execute the WAKE protocol.
- TTP_A: the identity of the TTP of the entity A, that is, A's certificate is generated by TTP_A.
- $(APriKey, APubKey)$: a long-term certified authentic asymmetric key pair (private and public keys) of a principal A.
- $APubKey\{m\}_R$: a message m encrypted under the long-term public key of A for data confidentiality. Here the subscript R means that this generic form must provide *retrievability* of m.
- $APriKey\{m\}_{NR}$: a *non-recoverable* (NR) signature for a message m signed by A using the private key of A. In general, this need not be a true signature, but need only provide assurance that the sender has applied his own private key to form this field.
- $SessKey_{AB}$: a new session key generated by both A and B through the execution of the WAKE protocol.
- $SessKey_{AB}\{m\}_{NR}$: the result of a cryptographic operation which applies the session key $SessKey_{AB}$ to a message m, from which m must be non-recoverable. Most

effective implementation would be *hashing* of *m and* $SessKey_{AB}$, using a message authentication code (MAC).

- $SessKey_{AB}\{m\}_R$: recoverable symmetric cipher encryption of a message *m*, that is, ordinary symmetric cipher encryption which allows the corresponding data to be recovered by correct decryption.
- r_A: a random nonce chosen by *A*
- $f(r_A,r_B)$: a key agreement function f with its inputs r_A and r_B.
- *CertChain*(*A*,*B*): a certificate chain from which *A* can retrieve an authentic copy of *B*'s public key, which is usually required in the situation where TTP_A and TTP_B are different from each other.
- TS_{TTP_A}: a timestamp generated by the TTP of the user *A*.

3.1.2 Generic Protocol

The mobile communication networks have radio base stations to provide radio connection to mobile terminals. The base stations continue to broadcast all the required system information which mobile terminals have to keep to listen to access the network. The availability of broacast channels in mobile communication systems is exploited in the design of the generic WAKE protocol: *the public key BPubKey of the network B assumed to be broadcast to mobile users*. That is, *BPubKey* is regarded to be always available to the mobile user *A*. This assumption allows us to adopt challenge encryption such as $BPubKey\{r_A\}_R$ for the first protocol message from *A* to *B*. The benefit of this approach here is that the encryption can be prepared offline before *A* tries to access the network, which will help reduce the delay time in connection establishment. The decrypted response from *B* to the challenge results in the authentication of *B* by *A* with the one-way authentication type $DA_{F,Ack}$. On the other hand, the prototype of authentication of the user *A* by the network *B* should be OA_S or OA_F to satisfy "non-repudiation of origin for relevant user data". Although self challenge based on timestamps or sequence numbers offers the advantage of fewer messages, it incurrs overhead of managing and synchronization [MvOV97]. With roaming between different NOs basically assumed, that kind of overhead may be crucial. Moreover, it may be a false economy to reduce the number of required protocol messages too much. WAKE

protocols must help two principals (the user and the NO/VASP) with different capabilities and characteristics negotiate and agree on the conditions relevant to their session. Therefore, it will be a reasonable approach to try to achieve all the goals of WAKE protocols with three protocol messages. For this reason, we choose forced callenge, and hence the prototype OA_F for user to NO/VASP authentication. Combining two prototypes of one way authentication, we end up having $DA_{F,ACK}$-OA_F for mutual authentication between the user *A* and the NO/VASP *B*.

Note that not a concrete cryptographic primitive but a thoroughly abstract or generic level language developed in Section 2.1 was used in the above design procedure, which results in a particular prototype of mutual authentication between *A* and *B*. It is interesting that the prototype $DA_{F,ACK}$-OA_F chosen for our WAKE protocol is different from the prototype OA_F-OA_F which the ASPeCT and the STS protocols belong to. At first, we concluded that $DA_{F,ACK}$-OA_F had never been used in any existing protocol. At a later stage of the work for the thesis, however, it turned out that the famous SSL/TLS protocol (see Section 4.2.1.2) is also of the same prototype as our new WAKE protocol. There has been an international effort to incorporate SSL/TLS protocol for transaction layer security into the mobile communications, which resulted in the WTLS (Wireless Transport Layer Security) protocol of WAP (Wireless Application Protocol) [WTLS00]. This is a positive side of our design work because the same cryptographic primitives required for the WTLS protocol can be reused for our WAKE protocol implementation within mobile terminals, enabling economic realization of security for mobile terminals with strict storage space constraints.

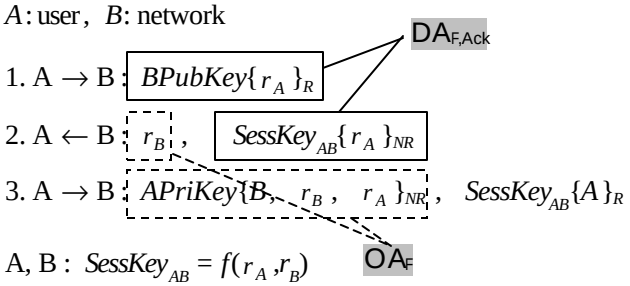
In the following sections, we describe the generic WAKE protocol of the prototype $DA_{F,ACK}$-OA_F which has three typical variants for different conditions with regard to the availability of certificates of the relevant entities. In the following description of all the variants of the generic protocol, the usage of the relevant fields for any certificate transmission may be regarded as neither absolute nor fixed. The actual usage may depend on the application environment, and moreover any WAKE protocols should do.

The first variant A is the most straightforward version where both the user and the network (or VASP) are in possession of all the required certified public keys to be used in the execution of the protocol. In fact, this version would be the most frequently used one among all three versions. The second variant B is for the more challenging situation

where each principal has no copy of the certified public key of the other but, fortunately, the network can handle this difficulty without online communication with the TTP. Lastly, variant C is devised to handle the worst case where the network needs to contact the TTP on-line. It should be noted again that the three variants do not exhaust all possible variants that may be desirable in different applications.

3.1.2.1 Variant A

Protocol 35. *Variant A*



This variant is the simplest form of the new genric WAKE protocol of the prototype $DA_{F,ACK}$-OA_F, and intended to be used for the situation where the user *A* and the network *B* have authentic copies of the public keys of the network and the user respectively. A typical example would be the current registration of the mobile to the network. Another example is the new registration where the two TTPs of *A* and *B* (TTP_A and TTP_B) are the same entity. In the generic protocol having three variants shown in this chapter, it is assumed that the network *B* broadcasts its certificate (and its TTP identity, TTP_B) over the system broadcast channel. If bandwidth becomes a critical problem, then only *B*'s public key without any certificate and TTP identity can be broadcast instead. In this variant, however, both principals are assumed to have the authentic copy of each other's public key.

The following briefly describes the above protocol execution.

- The user *A* chooses a random challenge value r_A, encrypts it using the public key (*BPubKey*) of the network, and sends the resulting ciphertext to the network *B* (message 1).
- The network *B* decrypts the received ciphertext and retrieves the challenge r_A from *A*. It also generates a random nonce value r_B and calculates a new session

key, $SessKey_{AB}$ using the key-agreement function f with r_A and r_B as its inputs. Using the new session key, it encrypts r_A and then sends it together with r_B to A (message 2).

- The user A calculates a new session key $SessKey_{AB}$ using the received random challenge r_B and his own challenge value r_A as two inputs for the key agreement function, and then computes and checks the value of $SessKey_{AB}\{r_A\}_{NR}$ against the received one, both of which should match.
- The user A makes and sends B a non-recoverable signature (for example, using pre-hashing) on the message including the network identity B and the challenge values, r_A and r_B. The user also encrypts his own identity A with the session key and sends it to the network (message 3).
- The network B verifies the received signature from A using A's public key and its own knowledge of the fields: B, r_B and r_A.

The successful run of the protocol guarantees the following assurances and hence achieves all the goals stipulated in Section 1.2:

- **Mutual authentication**. The exchange of $BPubKey\{r_A\}_R$ and r_A (messages 1 and 2) authenticates B to A, and that of r_B and $APriKey\{B, r_B, r_A\}_{NR}$ vice versa.
- **Secure and authentic session key agreement**. The new session key $SessKey_{AB}$ is calculated using fresh random nonces from both principals and one of these two key inputs, r_A, is known to only A and B, and is always exchanged in encrypted form, ensuring that the session key cannot be disclosed to any adversary. Furthermore, the encryption of r_A under the new session key, $SessKey_{AB}$, in message 2 and the inclusion of r_A and r_B in the nonrecoverable signature in the third message provide key confirmation to A and B, respectively. In fact, the encryption of the user identity using the session key $SessKey_{AB}$ in the third message provides much stronger key confirmation to B. The uniformity of all three generic protocols is the reason that variant A also includes r_A in the signature like the other variants B and C (see the subsequent sections). Moreover, the independence of pure authentication and key establishment related fields upon the

subsequent fields (at least in the logical sense) would be a prudent practice in security protocol design.

- **Confidentiality of relevant data**
  - User anonymity: *the user's identity A is transmitted encrypted under the network's public key BPubKey* (*message 1*). *Alternatively, the identity A may be encrypted under the new session key* $SessKey_{AB}$ *and transmitted in the third message.*
  - User data confidentiality: *the new session key* $SessKey_{AB}$ *enables encrypted transmission of any user data in the third message or the subsequent messages.*
- **Non-repudiation of origin for relevant user data**. The third message made by the user using his private key, *APriKey* is a placeholder for any data nonrepudiation which is needed, for example in a piggy-backed electronic commerce protocol. The inclusion of the random value r_B gives the network *B* a confidence that it has not been replayed.

The first two goals stipulated above will, in fact, be formally proved in Section 3.1.3 using the new analysis method presented in Section 2.2.

3.1.2.2 Variant B

Protocol 36. *Variant B*

A: user, *B*: network, TTP_A: TTP of *A*

1. A → B: $BPubKey\{r_A\}_R$, TTP_A
2. A ← B: r_B, $SessKey_{AB}\{r_A\}$, $CertChain(A, B)$
3. A → B: $APriKey\{B, r_B, r_A\}_{NR}$, $SessKey_{AB}\{ACert\}_R$

A, B: $SessKey_{AB} = f(r_A, r_B)$

This is the variant for the situation where the user and the network do not have authentic copies of the public keys of the network and the user respectively, but the network is able to handle this difficulty without online communication with the TTP. This variant includes all the fields used in the variant A with the identity *A* replaced by the certificate *ACert* in Step 3 (enclosed in the shaded box), and some additional fields enclosed in dotted boxes. Since the only additions concern certification data, it can be seen that all

the security goals identified to have been achieved in the variant A are still valid in this variant.

In this variant, the user *A* uses the public key of the network *B* for the generation of message 1 but cannot verify *BPubKey* until he receives message 2 containing the required certificate chain. That is why the user identity *A* is omitted in message 1. Including the identity of the user identity field would weaken *user anonymity* because the use of an *unverified* public key for encryption might cause unintended disclosure of the user identity to an adversary. Thus, along the same lines of ASPeCT protocol, the user identity is delayed until message 3, where an encrypted form of *A*'s certificate *ACert* is included.

Typical application of variant B would be the new registration of the mobile to the network and, additionally, when the TTP_B is different from TTP_A. The mobile can check this inequality of TTPs by retrieving the corresponding data field from the system broadcast channel of the network *B*. It should be noticed, however, that even without any pre-knowledge of TTP_B in *A*, this variant can be run successfully between *A* and *B*. Actual usage and implementation of the protocol prototype can be flexibly adapted as required.

3.1.2.3 Variant C

Now we describe the last variants for the situation where the user and the network do not have authentic copies of the public keys of the network and the user respectively, and, moreover, the network operator *B* has to contact the TTP_B or TTP_A online to get the required certificate chains. A typical example would be the new registration of the mobile to the network as in the case of the protocol *B*, but in this case the network *B* cannot provide any certificate chain to *A* from which *A* can verify *B'*s public key. This is usually because the network *B* and the TTP of A, TTP_A, has no pre-established trust chain available between them.

These difficulties make variant C have an appearance looking rather complex, but everything except the fields for the certificate clearing is the same as variant A or B. Two additional messages are included for *B* to access TTP_B or TTP_A , and get the required response.

We consider two sub-variants for this variant: the first one is where the network accesses its own TTP, and the second one the user's TTP. This approach of different variants for different TTPs is due to the significance of the difference between two TTPs with regard to the mobile situation. Usually, the TTP of the network, TTP_B is the local TTP for the relevant session, whereas the TTP of the user, TTP_A is the remote one. Hence we may want a variant (here C1) which simply fills in the gap between the required certificate chain between the user and the network, and another variant (here C2) where additional security goals such as checking certificate revocation status of the user and/or the network can be obtained.

3.1.2.3.1 Variant C1

Protocol 37. *Variant C1*

A: user, B: network,
TTP_A and TTP_B: TTPs of A and B

1. A $\rightarrow$ B : $BPubKey\{r_A\}_R$, TTP_A
2. B $\rightarrow$ TTP_B: B, TTP_A
3. B $\leftarrow$ TTP_B: $CertChain(A, TTP_B)$, $CertChain(B, TTP_A)$
4. A $\leftarrow$ B : r_B, $SessKey_{AB}\{r_A\}$, $CertChain(A, B)$
5. A $\rightarrow$ B : $APriKey\{B, r_B, r_A\}_{NR}$, $SessKey_{AB}\{ACert\}_R$

A, B : $SessKey_{AB} = f(r_A, r_B)$

This variant is a straightforward extension of the variant B. The network B has no certificate or certificate chain which can be verified by the user A. In most cases, it also means that the network cannot verify the user's certificate either. The network then accesses its own local TTP_B to obtain the required certificate chains. Here again, the dotted boxes indicate the added parts to the original fields in Variant A, and the shaded box in Step 3 encloses *ACert* which has replaced the identity A.

3.1.2.3.2 Variant C2

Protocol 38. *Variant C2*

A: user, B: network,
TTP_A: TTP of A

1. A $\rightarrow$ B: $BPubKey\{r_A\}_R$, TTP_A, $TTP_APubKey\{SessKey_{ABT}\}_R$, $SessKey_{ABT}\{A\}_R$
2. B $\rightarrow$ TTP$_A$: $BCert$, r_B, $TTP_APubKey\{SessKey_{ABT}\}_R$, $SessKey_{ABT}\{A\}_R$
3. B $\leftarrow$ TTP$_A$: $CertChain(B, TTP_A)$, $SessKey_{ABT}\{ACert\}_R$,
   $TTP_APriKey\{B, r_B, SessKey_{ABT}\{ACert\}\}_{NR}$, $SessKey_{ABT}\{BCert\}_R$
4. A $\leftarrow$ B: r_B, $SessKey_{AB}\{r_A\}$, $SessKey_{ABT}\{BCert\}_R$
5. A $\rightarrow$ B: $APriKey\{B, r_B, r_A\}_{NR}$, $SessKey_{AB}\{SessKey_{ABT}\}_R$

A, B: $SessKey_{AB} = f(r_A, r_B)$
$SessKey_{ABT}$: random value chosen by A as a secret key

This variant has an even more complex appearance which is an inevitable consequence of an additional goal of this variant protocol: confidence in each party that the other's certificate has not been revoked. This requires for both the user and the network to authenticate the TTP. Note that the signature for *A*'s certificate in the third message is just redundant unless it provides the current status of the certificate: revoked or not? Moreover, it would be too expensive for the network *B* to access the TTP_A (probably more distant than TTP_B) just for the retrieval of the required certificate chain. As for the user side, authenticating his own TTP means that he does not have to check every elementary step in the certificate chain from *A* to *B*. That is why the fourth message does not convey any certificate chain *CertChain*(*A*, *B*) but simply the encrypted certificate of the network *B*. Here, note that the encryption of *BCert* is not for the confidentiality but for data integrity and the authentication of TTP_A by *A* (as described below).

The random session key, $SessKey_{ABT}$ chosen by *A* protects his own identity from any entity except TTP_A and the network *B*. Even the network *B* is not able to know *A*'s identity until it receives $SessKey_{AB}\{SessKey_{ABT}\}_R$, in the message 5. This encrypted field using the new session key $SessKey_{AB}$ between *A* and *B* provides explicit key confirmation to *B* and also enables *B* to get another session key $SessKey_{ABT}$, from which it can decrypt $SessKey_{ABT}\{ACert\}_R$ and get *A*'s certificate, *ACert*.

$SessKey_{ABT}$ plays an additional role of challenging TTP_A and hence providing a replay-proof or message freshness of the data from TTP_A to A. This feature may help provide the assurance to the user A that the certificate *BCert* of the network (or VASP) has not been revoked if TTP_A checks the revocation status of *BCert* on behalf of A.

3.1.2.4 A scenario: New registration

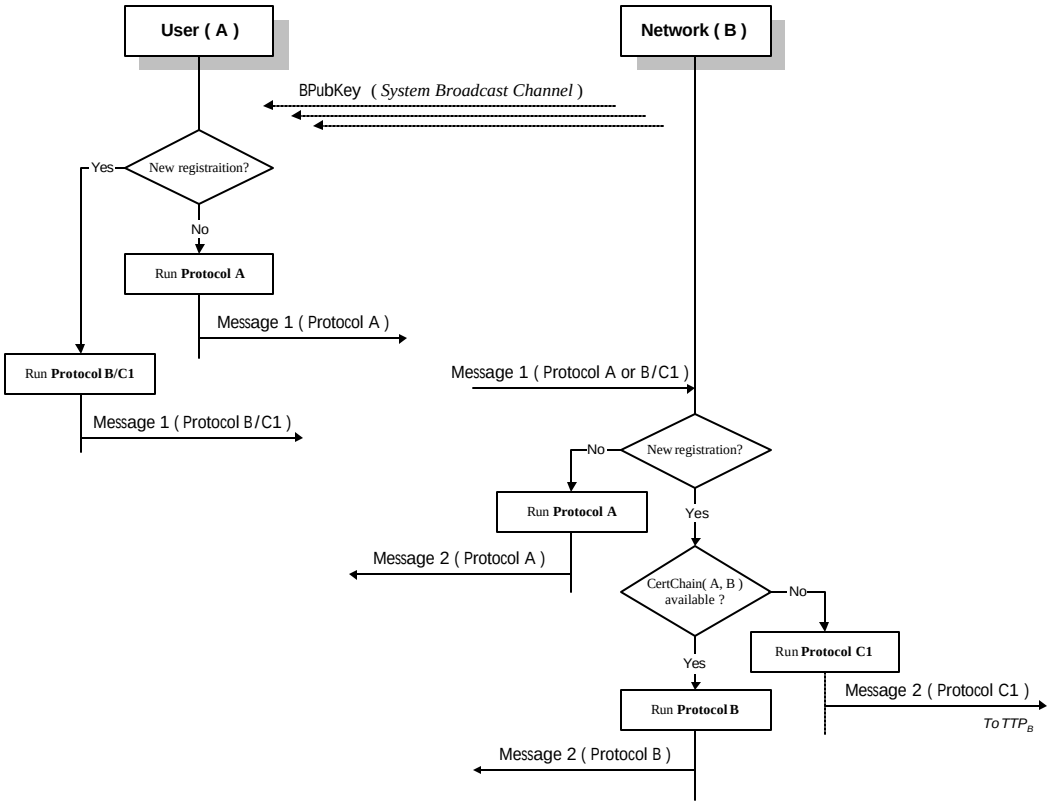


Figure 3-2: An application scenario of the generic protocol set A, B and C1 for new registration of the mobile to the visited network

Figure 3-2 shows a conceptual application scenario of the generic protocol set A, B and C for new registration of the mobile to the visited network. When a user roams into a new visited network operator's service area, it checks the system broadcast channel to get system information including the public key of the network (not necessarily the certificate of the network). For a new registration, the user terminal initiates protocol B/C1. It is desirable that the difference between the protocol versions B and C1 is

transparent to the mobile. The network checks the availability of the certificate chain of the public key of itself for the mobile. If it is available, it runs protocol B, and otherwise runs protocol C1 which leads to sending the message 2 to the TTP of the mobile.

In fact, many other scenarios for the usage of the protocol variants are possible, especially when we are allowed to modify some variants and their implied assumptions. For example, if we assume that the network broadcasts its certificate itself, not just its public key, and the variant A is modified to include the user's certificate instead of its identity in its third message, then the variant A itself can be applied to the new registration as well. This kind of flexibility, in fact, seems to be advantageous in most applications.

3.1.3 Security Analysis of the Protocol using the New Analysis Method

For both cryptographic algorithms and protocols, only limited confidence in security can be gained from an internal analysis. In order to gain acceptance and high confidence in security, algorithms and protocols must be openly published and scrutinised by the international research community. Our protocol had been published for only a short time, but we are aware that it has already been examined by a number of independent parties. In particular, it was one of seven protocols examined in a recent paper by three members of the ASPeCT team [HMM99]. In their analysis they find that of these seven protocols, only three are able to provide the goals that they regard as necessary for a wireless authentication protocol. These three protocols are our new WAKE protocol, the STS protocol [DvOW92], and their own ASPeCT protocol. In the conclusion to their paper they have stated the following.

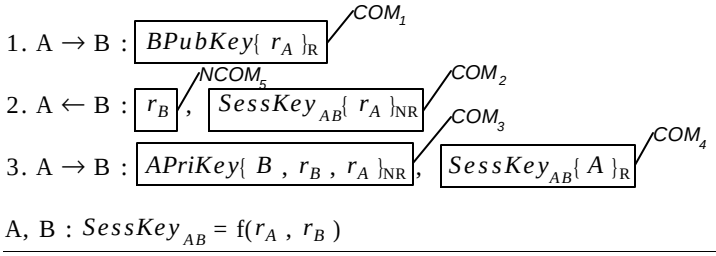
> *… The Boyd-Park protocol seems to be the most computationally efficient at the user's end, but requires more communication passes. The ASPeCT protocol was purposely designed for authentication and payment initialization application and like the Boyd-Park protocol, appears to compare well very favourably with the prior art. …*

Therefore we may conclude that initial assessment of our protocol by the international community is very positive. Nevertheless, we have continued to explore the

requirements and applicability of our protocol in different environments and compared it with the alternatives. This leads us to propose a set of variant protocols which are presented in this report.

In this section, we analyse the new WAKE protocol using our newly developed analysis method as described in Section 2.2. This analysis is a formal proof of the first two of the goals which are achieved by the generic protocol in Section 3.1.2.1. For convenience, we show again the generic variant A of the protocol below. The other variants B and C1/C2 are basically the same protocols as variant A; and hence we do not analyse those three variants.

Protocol 39. *Generic WAKE protocol*



We will show that, for both entities, this protocol satisfies entity authentication, explicit key authentication and their well-combinedness as defined in Definition 2-11, Definition 2-16, Definition 2-17, and Definition 2-18, respectively, after both entities finish the protocol run successfully.

B's viewpoint:

We start our analysis by assuming that B has just finished the protocol run with receiving the third protocol message and its successful verification. Now we prove the following statements for B.

$$X_O^1 = X_O^3 = X_O^4 = A \quad (1)$$

$$A \twoheadrightarrow B \quad (2)$$

$$(A, B)! \ni SessKey_{AB} \quad (3)$$

Statement (1) means that the originators of COM_1, COM_3, and COM_4 are in fact the entity A. Statement (2) denotes that the other party A is aware of B as his peer entity of the protocol run. The last statement (3) says that only A and B have the new session key

established in the protocol run. Note that these three statements satisfy two types of authentication and their well-combinedness.

First, looking at the message COM_3, we (and *B*) can derive that its originator, namely X_O^3 is in fact the entity *A* and he is aware of *B* as his peer entity, has r_A and r_B, and therefore has (or at least can compute) the session key $SessKey_{AB}$. That is,

$$X_O^3 = A \text{ (and trivially } X_O^3 \backslash A) \tag{4}$$

$$A \twoheadrightarrow B \tag{5}$$

$$A \ni (r_A, r_B, SessKey_{AB}). \tag{6}$$

Moreover, COM_3 includes the same random number r_A as delivered in COM_1, which means that *A* belongs to the set of all possible senders of the message COM_3. That is,

$$A \in X_S^1 \tag{7}$$

Meanwhile, the freshness of r_B also guarantees the freshness of $SessKey_{AB} = f(r_A, r_B)$ where f is a one-way function. This freshness of $SessKey_{AB}$ in turn gives us the following fact when we investigate COM_4.

$$X_O^4 \ni SessKey_{AB} \tag{8}$$

$$X_O^4 \backslash A \tag{9}$$

The statement (9) simply means that the originator of COM_4, i.e., X_O^4 is claiming the identity of *A* in this particular session.

Now, let's prove that X_O^1 is no other entity than *A*. We will use the simple proof method: *proof by contradiction*. At the moment, let's assume that

$$X_O^1 \neq A. \tag{10}$$

We know that $A \ni r_A$. How could *A* have been able to have or know r_A? Let's check all the possible ways *A* can have the value of r_A.

Case 1: The entity *A* himself may have generated r_A. It then means, however, that *A* is X_O^1 because the statement (5) $A \twoheadrightarrow B$ implies that *A* has sent r_A in the form $BPubKey\{r_A\}_{NR}$, which is in fact COM_1. Hence we have $X_O^1 = A$, which is a contradiction!

Case 2: The entity A has succeeded in disclosing the value of r_A from COM_1. This violates our basic assumption that $BPubKey\{r_A\}$ can only be decrypted by B. This case can also be disproved by the *rule of mask* as well (see Section 2.2.4). More specifically, A must have constructed the attack structure: X_O^1 — X/A\A — B. This means that A claims his own identity to B, and some identity X to X_O^1. The rule of mask requires that the claimed identity X should be either A or B. If $X = A$, then it means that X_O^1 has $APubKey\{r_A\}$, which is clearly a contradiction (in fact, this kind of mask construction has no meaning at all for any attack). Otherwise if $X = B$, then X_O^1 has sent $BPubKey\{r_A\}$ and hence A cannot access the secret value r_A.

Case 3: The entity A has retrieved r_A from the second protocol message, that is

$$A\backslash A \leftarrow B/\ X_S^2 : r_B, SessKey_{AB}\{r_A\}_{NR}. \qquad (11)$$

This case is impossible too because access to r_A requires the knowledge of $SessKey_{AB}$, which requires in turn the knowledge of r_A. By the way, the statement (11) comes from the *axiom of loyalty* (see Section 2.2.4), which says each entity (including attackers as well) involved in the protocol run must be *loyal* to his role including his mask (i.e., the claimed identity).

Case 4: The entity A has retrieved r_A from the third protocol message. That is

$$X'\backslash X \rightarrow B/A: XPriKey\{B, r_B, r_A\}_{NR}, SessKey_{XB}\{X\}.$$

However, this case is also impossible because the signature in the above message is in fact non-recoverable (NR).

From the analysis of all the possible cases above, we reach contradictions. Therefore, we can conclude

$$X_O^1 = A. \qquad (12)$$

Now we want to prove that only A and B can access the $SessKey_{AB}$. In other words, only A and B can have the knowledge of the value of r_A. Let's assume the opposite, i.e.,

$$\exists X\ (X \neq A \wedge X \neq B \wedge X \ni r_A). \qquad (13)$$

The fact that $X_O^1 = A$ obviates the possibility of the attack construction:

$$A\backslash A — B/X\backslash A — B.$$

The masks on both sides of X are determined by rules of mask. Here X as a man-in-the-middle cannot retrieve r_A from the first protocol message since it is encrypted under *BPubKey*. Hence, the only remaining possibility we need to consider is the following attack construction:

$$X \text{ — } B/A\backslash A \text{ — } B. \qquad (14)$$

The related masks on both sides of A as man-in-the-middle are determined by the fact (4) A\A and the mask rule. This configuration inevitably leads A's receiving of the first protocol message from his left side (i.e., X) of the form, $BPubKey\{r_A\}$. This is because r_A is chosen by X and was delivered to the right side of X, which is the only way A can access to r_A. However, this again turns out to be a contradiction. For it means that $X_O^1 = X \neq A$, which violates our discovery that $X_O^1 = A$. Therefore, we can be sure that there is no other entity X than A and B who is able to access the value of r_A, and hence the value of $SessKey_{AB}$. That is,

$$(A, B)! \ni SessKey_{AB}. \qquad (15)$$

This together with (8) and (9), in turn, leads to the fact that

$$X_O^4 = A. \qquad (16)$$

Now, from the statements (4), (5), (6), (12), (15) and (16), we can conclude that the statements (1), (2) and (3) are true. ■

A's viewpoint:

Before we enter detailed steps of proof, it should be noticed that the conclusions of A and B are not strictly symmetric because A cannot be sure whether B has actually received the final protocol message until any explicit proper confirmation is provided. In practice, however, the protocol run will be accompanied by the relevant subsequent data exchange containing some ciphertext using the new session key $SessKey_{AB}$. Hence this asymmetry has no practical significance and A's confidence level will be the same as that of B. To make the analysis clearer and easier, we derive A's confidence or conclusion assuming A has received a subsequent key confirmation message after the protocol run. Hence we use the following additional subsequent message from B to A.

---

4. A ← B: $SessKey_{AB}\{ m \}$ where m is a recognizable message.

We denote this additional cryptographic message by COM_6. Now we will show the following statements are all true from *A*'s viewpoint.

$$X_O^2 = X_O^6 = B \tag{17}$$

$$B \twoheadrightarrow A \tag{18}$$

$$B \in X_S^5 \tag{19}$$

$$(A, B)! \ni SessKey_{AB}. \tag{20}$$

Note that the above statements clearly satisfy the definitions of entity and key authentication and their well-combinedness.

Since r_A is fresh and recognizable to *A* (he generated the value of r_A), the part COM_2 of the second protocol message gives *A* the following assurance.

$$X_O^2 \ni (r_A, r_B, SessKey_{AB}) \tag{21}$$

Also, the fact that X_O^2 has used the value of r_B as received by *A* for computation of the $SessKey_{AB}$ gives *A* the assurance that X_O^2 belongs to the set of all possible senders of the message $NCOM_5$. In symbolic notation:

$$X_O^2 \in X_S^5. \tag{22}$$

The freshness of the $SessKey_{AB}$, and the assumed message COM_6 gives another assurance to *A* as follows.

$$X_O^6 \ni (r_A, r_B) \tag{23}$$

First we negate one of our final goal $X_O^6 = B$. That is, assume $X_O^6 \neq B$. Then, how could X_O^6 get the value of r_A? There is only one possible scenario where *B* played man-in-the-middle attack between *A* and X_O^6. For, in the attack construction for this scenario, X_O^6 cannot come between *A* and *B*. In other words, we cannot have the construction like

$$\text{A} \text{---} \text{B} / X_O^6 \backslash ? \text{---} \text{B/B} \tag{24}$$

since the only protocol message which allows X_O^6 to retrieve r_A is the first protocol message including COM_1. COM_1 is a ciphertext encoded with $BPubKey$, and so X_O^6 cannot recover the value of r_A with the above attack construction. Therefore, we need only to consider the following attack construction.

$$\text{A} \text{---} \text{B/B} \backslash \text{A} \text{---} X_O^6 / X_O^6 \tag{25}$$

The two masks on the face of *B* (that is *B* and *A*) are uniquely determined from the mask rule. Also we can say that X_O^6 could have retrieved r_A only by claiming his own identity to his left side. This attack configuration and the axiom of loyalty requires *B* to send the third protocol message to X_O^6 as shown below.

$$\text{B\A} \rightarrow X_O^6 / X_O^6 : APriKey\{\ X_O^6, r_B, r_A\}, SessKey_{AB}\{\ A\ \}$$

In fact, however, *B* cannot form the first part of the above message, since it requires *B* to have *A*'s private key. Hence our assumption that $X_O^6 \neq B$ has sent COM_6 to *A* is a contradiction. Therefore we have the following fact.

$$X_O^6 = B \tag{26}$$

Now *A* can be confident about the following fact since he can be confident about the truth of (23) and the fact that *B* has successfully received and verified the third protocol message.

$$B\ \ni (r_A, r_B, SessKey_{AB}) \tag{27}$$

$$B \twoheadrightarrow A \tag{28}$$

The fact that $B \ni r_B$ (27) makes sure that

$$B \in\ X_S^5 \tag{29}$$

Now we have to prove $X_O^2 = B$. Let's assume the negation, that is, $X_O^2 \neq B$. Again we have to consider all the possible scenario where X_O^2 could access r_A. In the argument about (24), we can say that this construction is impossible for $X_O^2 \neq B$ too. The following construction similar to (25) is the only way X_O^2 could access the value of r_A.

$$\text{A} \text{---} \text{B/B\A} \text{---}\ X_O^2 / X_O^2 \tag{30}$$

Hence we can construct the relevant message flow as follows.

1. A $\rightarrow$ B/B: $BPubKey\{\ r_A\ \}$
   1'. B\A $\rightarrow$ X_O^2 / X_O^2: $X_O^2 PubKey\{\ r_A\ \}_R$
   2'. B\A $\leftarrow$ X_O^2 / X_O^2: $r_B, SessKey_{AB}\{r_A\}_{NR}$
2. A $\leftarrow$ B/B: $r_B, SessKey_{AB}\{r_A\}_{NR}$
3. A $\rightarrow$ B/B: $APriKey\{\ B, r_B, r_A\}_{NR}, SessKey_{AB}\{\ A\ \}_R$
   3'. B\A $\rightarrow$ X_O^2 / X_O^2: $APriKey\{\ X_O^2, r_B, r_A\}_{NR}, SessKey_{AB}\{\ A\ \}_R$
   /* The first part of message cannot be constructed by B, and X_O^2 will abort the session */.
4. A $\leftarrow$ B/B: $SessKey_{AB}\{\ m\ \}$

In the above scenario, we may have $X_O^2 \neq B$. But it should be noticed that B, as a man-in-the-middle attacker between A and B, cannot complete the compromised session with X_O^2. Hence, after the completion of the session there is no other entity A and B who is alive and related to the newly established secure channel between A and B. Furthermore, B has the ability to construct COM_2 for given value of r_B in the above example session, and in fact he has computed the value of $SessKey_{AB}$ to send the fourth message COM_6. Therefore, although we cannot prove if $X_O^2 = B$, there seems to be no practical security problem at all. Furthermore, if it really matters then we can include the identity of B in the second protocol message as follows:

2. A $\leftarrow$ B : r_B , $SessKey_{AB}\{ B, r_A \}_{NR}$

Note that with this new form, we can prove that $X_O^2 = B$. This is a very interesting point, and which may be another example that shows the analysis method developed in Section 2.2 does not fail in capturing this kind of a subtle problem.

Now we have already seen that if there is some entity X such that $X \neq B$, $X \neq A$ and X ∋ r_A, this leads to a contradiction or failed man-in-the-middle attack attempted by B. Hence we can say from A's perspective that A and B are the only entities who have the value of r_A. Hence we can conclude

$$(A, B)! \ni SessKey_{AB} \qquad (31)$$

Now we have proved all the statements (17) to (20). ■

3.2 Instance Protocol Design

Due to limitations in mobile communications in terms of processing power, memory and bandwidth, any proposed security protocol should be carefully designed and analyzed with respect to its complexity. In fact, a good analysis has already been done on this issue over several protocols proposed for future mobile communications [HMM99] including the initial version of the generic protocol described above.

In this section, we propose a concrete instance protocol conforming to the abstract generic protocol. We also compare the instance protocol, ASPeCT protocol and STS protocol with respect to computational complexity, which, as pointed out by Horn et al. [HMM99], are the only protocols which seem to satisfy all the security goals for future wireless mobile communications.

We summarize below our notations used for instance protocol descriptions in this section.

- A, B: the identities of the two entities who execute the protocol.
- g: a common agreed generator of a finite group
- (b, g^b): an asymmetric key pair of B where b is the private and g^b is the public component
- r_A: a random nonce chosen by A
- $\{\ldots\}_{K_A^{-1}}$: Signature under A's private signature key, K_A^{-1} (We admit, of course, that this notational form is not really an *instance*, but rather quite *generic*. In fact, we consider the ElGamal type signature as a proper instance algorithm, which is assumed for complexity analysis in the later part of this chapter.)
- $\{\ldots\}_{K_{AB}}$: Encryption under the session key, K_{AB} between A and B
- $h(\ldots)$: a common agreed hash function applied to a message
- K_{AB}: a new session key established in the current session between A and B.

3.2.1 Instance WAKE Protocol

Protocol 40. *Instance WAKE protocol*

A: user, B: network

1. A $\rightarrow$ B: $(g^b)^{r_A}$
2. A $\leftarrow$ B: r_B, $h(g^{r_A}, K_{AB})$
3. A $\rightarrow$ B: $\{h(B, r_B, g^{r_A})\}_{K_A^{-1}}$, $\{A\}_{K_{AB}}$

A, B: $K_{AB} = h(g^{r_A}, r_B)$

This is an instance protocol designed to comply with the generic protocol described in Section 3.1.2.1. Therefore, it suffices to verify that all the mathematical expressions here satisfy the generic equivalents, which can be checked from Table 3-1. The description of the generic protocol in Section 3.1.2.1 exactly applies to this protocol as well. This protocol does not provide perfect forward secrecy because the compromise of B's long-term private key b leads to the disclosure of g^{r_A}, which in turn compromises the session key K_{AB}. (Note, however, that knowledge of *APriKey* does not disclose g^{r_A} so in this sense a partial forward secrecy provided.)

Table 3-1: Comparison of generic and instance forms

Generic Protocol	Instance Protocol	Description
r_A	g^{r_A}	Challenge from A to B
r_B	r_B	Challenge from B to A
$BPubKey\{r_A\}_R$	$(g^b)^{r_A}$	g^{r_A} can only be retrieved by the owner of B's private key, b, that is B. (r_A in the generic protocol corresponds to g^{r_A} in this instance protocol)
$SessKey_{AB}\{r_A\}_{NR}$	$h(g^{r_A}, K_{AB})$	The generic form is not for encryption but for key confirmation and authentication of B to A, and the instance equivalent does exactly the same thing.
$APriKey\{B, r_B, r_A\}_{NR}$	$\{h(B, r_B, g^{r_A})\}_{K_A^{-1}}$	Non-recoverable signature by A
$SessKey_{AB}\{A\}_R$	$\{A\}_{K_{AB}}$	Recoverable encryption of the identity A using the session key
$SessKey_{AB} = f(r_A, r_B)$	$K_{AB} = h(g^{r_A}, r_B)$	Hashing as a key agreement function

The mechanism used here to encrypt as $BPubKey\{r_A\}_R$ does not conform to any general public key encryption algorithm. Therefore it is worthwhile to consider whether this mechanism really does provide the desired properties for encryption. To obtain the value of g^{r_A} from the encrypted value, $(g^b)^{r_A}$, and the public key of B, g^b, is exactly the *Diffie-Hellman* problem. Solving this problem is believed to be as hard as finding discrete logarithms. To find any partial information on the value of g^{r_A} is the same as the *Decision Diffie-Hellman* problem, which is the problem of deciding whether or not a value g^x completes a Diffie-Hellman triple with respect to g^y and g^z. In fact, this exact same mechanism has recently been proposed by Schoenmakers [Scho99] in a completely different application; we refer the reader to his paper for formal security proofs.

3.2.2 Complexity Analysis of WAKE Protocols

In this section, we compare the computational complexity of the STS protocol and ASPeCT protocols with the instance WAKE protocols proposed in this chapter; the analysis of Horn et al. [HMM99] indicated that only these three protocols satisfy all the security goals desired in future mobile communication systems. The STS protocol was not particularly designed for mobile communications, but it may be used as a reference point for comparison.

Public key related computations are overwhelmingly the most dominant portion of cryptographic computational load. We assume that *discrete log* based cryptography such as ElGamal encryption and signatures [ElGa85], or a similar variant scheme, is used. This assumption is not only reasonable but also very useful for comparison between different protocols like the ASPeCT protocol and our new one. With this assumption, all the comparison work will come down to *counting the number of exponentiations needed for each protocol.*

Table 3-2: Exponentiations required for ElGamal schemes

Function	Encryption	Decryption	Signature Generation	Signature Verification
Exponentiations	2	1	1	2

Table 3-2 shows the number of exponentiations required for a basic implementation of ElGamal encryption and signatures. Depending on the application, some of these exponentiations may be performed off-line (before the message to be encrypted or signed is known). The possibility of off-line computations on the user side is very important due to the limited computational ability of mobile devices. If computations can be performed during idle time of the mobile device delays in call setup can be dramatically reduced. For example, the exponentiation for signature generation may always be done off-line since it is independent of the message. The same applies to both exponentiations for encryption as long as the public key of the recipient is known beforehand. (If not only one exponentiation may be performed off-line). The computations for signature verification and decryption are always assumed to be on-line. It should be noted that in practice there are many ways to optimise the calculation of the exponentiations involved, which cannot be easily included in the cost computation at this time since they depend on details of the algorithms used for implementation of modular exponentiation. For example, some algorithms give a trade-off between computation and storage requirements. Therefore the figures should be regarded only as a guide of probable comparative performance.

We only describe the computational cost for the proposed instance WAKE protocol in Figure 3-3. For an analysis of the STS and ASPeCT protocols, refer to [HMM99]. In Table 3-3, we summarize the required number of off-line and on-line exponentiations for all protocols. From this table, we can see that the proposed protocol is exactly the same as the ASPeCT protocol in terms of the required number of exponentiations.

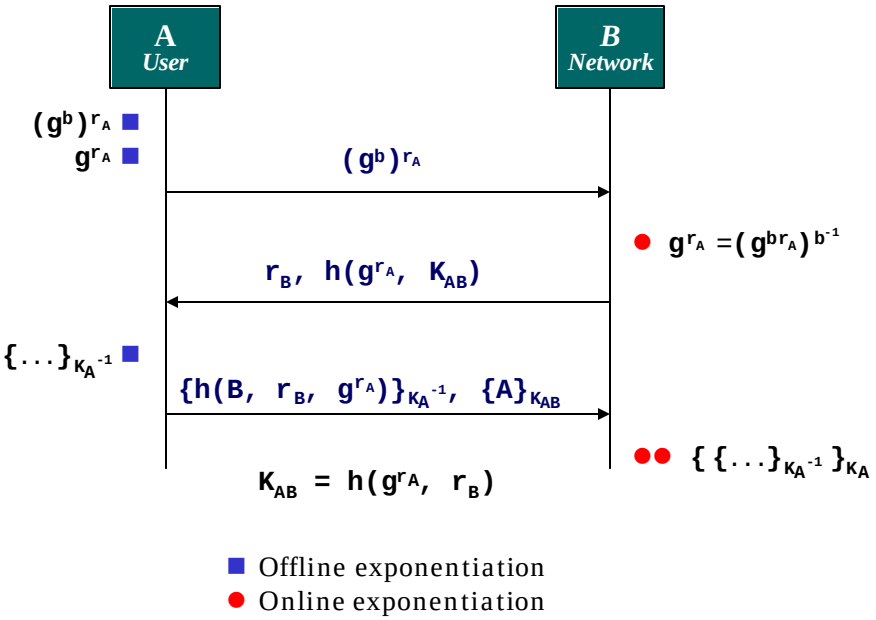


Figure 3-3: The number of exponentiations required to execute the instance WAKE protocol

Table 3-3: Computational cost comparison of the STS, ASPeCT, and the proposed instance WAKE protocol

Protocol	***Number of exponentiations***	
	***User* (*A*)**	***Network* (*B*)**
STS	■■●●●	■■●●●
ASPeCT protocol	■■■	●●●
Instance WAKE protocol	■■■	●●●

Notation: "■■■●", for example, means three *off-line* and one *on-line* exponentiations

3.3 Summary

Future generation mobile systems aim to have security features far beyond that of the current generation, for they will need to provide most of the computer communication services, including the Internet. This is a very challenging goal considering the limited computational and bandwidth resources and, nevertheless, more complex service management requirements in comparison with wireline based communications. Therefore, the cryptographic mechanisms for future mobile systems, especially the WAKE protocols, must be designed to be computationally effective and flexible.

To satisfy this goal, a new *generic* WAKE protocol is proposed, which is sufficient: (1) to fully exploit the future advances of implementation technologies in cryptography, (2) to flexibly accommodate a variety of protocol goals which depend upon more concrete instance protocols complying with the corresponding generic protocol. Several variant forms of the generic protocol have also been developed to cope with the different conditions in terms of *certificate availability* in the user and the network.

The new WAKE protocol is in fact a byproduct of the classification work described in Section 2.1. It was identified to belong to a different prototype ($DA_{F,Ack}$-OA_F) than ASPeCT protocols and the STS protocol of the prototype OA_P-OA_F.

The analysis methods developed in Section 2.2 has been applied to analyse the designed generic WAKE protocol, proving it to achieve the following two goals

- mutual authentication of user and network,
- secure and authentic establishment of a new session key,

These two goals correspond to **EntityAuth3** (Definition 2-11) and **ExplicitKeyAuth2** (Definition 2-17), respectively. These two goals are **well combined** (Definition 2-18) with each other in the WAKE protocol by Theorem 2-1.

Additionally, the following two remaining goals are achieved in the protocol (see Section 3.1.2.1)

- confidentiality of relevant data,
- non-repudiation of origin for relevant user data.

The instance WAKE protocol was compared with the ASPeCT protocol with regard to computational cost, and found to require the same number of modulo exponentiations when implemented with the cryptographic primitive of discrete logarithm.

Chapter 4.

Add-On Mechanisms for WAKE Protocol

As we discussed in Section 1.2, in addition to the commonly agreed basic goals of WAKE protocols for future mobile environments, we identified four more goals: *forward secrecy*, *robustness against denial of service attack*, *support for electronic payments*, and *clone detection*. The authentication and key establishment protocol is the starting point from which most security services such as data confidentiality and non-repudiation are provided. For efficiency reasons as well as overall system security, any other cryptographic facilities including the four features stipulated above should be built upon the authentication and key establishment protocols.

It is demonstrated how efficient protocols for these add-on mechanisms can be designed using the WAKE protocol as a basis. For forward secrecy, we identify two general prototypes, one of which was entirely overlooked in cryptographic protocol design by most authors. Both schemes are implemented into two instance WAKE protocols, i.e., *FS1* and *FS2* protocols (see Section 4.1.4).

4.1 Forward Secrecy

The use of session keys allows different sessions to be independently secure so that if one session key becomes compromised then this should not affect any other session key. Long term keys (symmetric or asymmetric) are used to establish session keys and so must be protected much more securely than session keys. Although it may be an unlikely event, the consequences of the compromise of a long-term key should be considered.

A protocol is said to provide *forward secrecy* if the compromise of long-term keys does not compromise past session keys that have been established before the compromise of the long-term key [MvOV97]. The idea of forward secrecy seems to have been coined by Günther in connection with an identity based protocol he proposed [Günt90]. In fact Günther used the term *perfect* forward secrecy; however since the word 'perfect' has connotations with unconditional security which are not relevant here, we will use the simpler term in common with a number of other authors. It should be noted that there seems to be a disagreement in the definition of forward secrecy because we can find literature where forward secrecy is intended to mean that a secret encryption key used in a session must be securely discarded after the session to prevent an adversary from obtaining the encryption key in any way and eavesdropping any future sessions protected by the same encryption key [Abel98]. In this thesis, however, we use only the former definition of forward secrecy, which appears to be the more generally agreed one.

We also note that there is a similar concept called forward *security* to address the significance of losing long-term private keys [BeMi99]. A signature scheme with forward security protects users from the threat of *signature forgery* in case their signature keys have been compromised. The basic idea to implement forward security is to update the signature key itself frequently to reduce the risk of key exposure. This may contribute also to the forward *secrecy* when it is the case that the signature key is used for authentication and key establishment as well, because the limited longevity of the signature key reduces the risk of relevant session key compromise down to the lifetime of the signature key. Still, however, forward security is not a sufficient condition for forward secrecy considering that the disclosure of the signature key would compromise any session keys computed using the signature key. In other words, if we confine our focus on a particular long-term private key (however long it lives), then it is only forward secrecy that protects the relevant session keys from the compromise of the long-term private key. With this in mind, we argue that the essential characteristic of forward secrecy is orthogonal to that of forward security. It should be noticed, however, that there seems to be a rather loose distinction in practice, which reflects, as we have already described, the fact that forward security may be regarded as a weak alternative

to forward secrecy in a practical sense [Aziz98]. In this report, however, we confine our discussion only to the forward secrecy in distinction to forward security.

It has long been known that protocols based on the Diffie-Hellman key agreement protocol [DiHe76] will usually provide forward secrecy. This is because the long-term keys are normally used only to authenticate messages and not to encrypt them. This property is widely regarded as a useful extra security feature of Diffie-Hellman based protocols, since most other protocols do not possess it. In the next section, however, we will see that forward secrecy does not require any particular type of cryptosystem such as Diffie-Hellman. Considering the significance of forward secrecy, it is rather surprising that this security feature has almost never been given a proper effort to understand it.

Unlike many other goals of security protocols, forward secrecy may have to be treated more practically. Its significance in real applications dramatically varies through both angles of communication types and user types. In communications between a private user and a public commercial entity, it is more the user than the commercial entity that is concerned about confidentiality for the past communications, and hence is more concerned about forward secrecy. On the other hand, forward secrecy usually requires some additional computations of asymmetric key cryptography, and hence might be a quite expensive cryptographic service in some types of communications, for example, voice communications or message broadcasting in some value added services.

In the next section two prototype constructions are presented for protocols providing forward secrecy. The first is based on the Diffie-Hellman protocol, while the second can be used with any chosen asymmetric encryption scheme. Section 4.1.2 then discusses the notion of "partial forward secrecy" and presents prototype constructions.

4.1.1 Two Prototypes for Forward Secrecy

Almost all authors discuss only Diffie-Hellman based key establishment protocols as examples providing forward secrecy and there seems to be an implicit assumption that these are the only possible examples. Consequently, there seems to be a tendency in protocol design that only Diffie-Hellman type key agreement functions are considered for forward secrecy implementation [Aziz98]. In this section, two prototype

constructions are presented, one based on the special algebraic properties of Diffie-Hellman key exchange, and the other which can work with any asymmetric encryption scheme. The second prototype addresses the question: *is there any other cryptographic algorithm than Diffie-Hellman satisfying forward secrecy?* We do not try to answer this question directly. Instead, we enlarge our scope of the search for forward secrecy to cover cryptographic *protocols*. Then, the answer to the above question is: "Yes. As many as the number of different asymmetric cryptographic algorithms."

We take a rather over-simplified approach to ease the capture of the very basic property which enables forward secrecy. Any other security goals such as entity authentication and key authenticity will be entirely omitted in the following descriptions. We believe that by taking this approach we can understand the mechanism of forward secrecy more clearly. Furthermore, these building blocks of forward secrecy mechanisms without any extra property would be more effectively integrated into authentication and key establishment protocols.

We first investigate what kind of property in Diffie-Hellman type protocols provides us forward secrecy. Two principals A and B select random secrets r_A and r_B, compute g^{r_A} and g^{r_B} respectively, and exchange them over an unsecured channel.

Protocol 41. *Basic Diffie-Hellman key agreement*

1. A $\rightarrow$ B: g^{r_A}
2. A $\leftarrow$ B: g^{r_B}

A: $K_{AB} = (g^{r_B})^{r_A}$
B: $K_{AB} = (g^{r_A})^{r_B}$

This basic protocol for key agreement is integrated into more sophisticated protocols which use long-term asymmetric keys of principals to provide entity authentication. The famous STS protocol is such an example.

In the STS protocol, the session key establishment is exactly in the same form as that of the basic Diffie-Hellman protocol, and does not depend on the long term asymmetric keys of A and B. Therefore a future possible disclosure of the long-term keys of A, B or both does not lead to the compromise of the session key. It should be noted that there are a number of other methods to incorporate authentication into basic

Diffie-Hellman and which also provide forward secrecy. Some examples may be found in the IEEE P1363 draft standard [IEEE99].

Protocol 42. *Station-to-Station (STS) protocol*

1. A $\rightarrow$ B: g^{r_A}
2. A $\leftarrow$ B: g^{r_B} , $\{\ \{g^{r_B}, g^{r_A}\}_{K_B^{-1}}\ \}_{K_{AB}}$
3. A $\rightarrow$ B: $\{\ \{g^{r_B}, g^{r_A}\}_{K_A^{-1}}\ \}_{K_{AB}}$

A, B: $K_{AB} = g^{r_A r_B}$

Now, we try to describe the very general property of Diffie-Hellman protocol in a quite abstract way. Using the abstract notation, we can re-describe the Diffie-Hellman protocol as follows, which we call the first prototype for forward secrecy.

Protocol 43. *An abstract level description of basic Diffie-Hellman key agreement*

1. A $\rightarrow$ B: *APubKeyX*
2. A $\leftarrow$ B: *BPubKeyX*

A: $K_{AB} = F(BPubKeyX, APriKeyX)$

B: $K_{AB} = F(APubKeyX, BPriKeyX)$

where F is a common key agreement function with the following property:
$F(BPubKeyX, APriKeyX) = F(APubKeyX, BPriKeyX)$

The public components, *APubKeyX* and *BPubKeyX* of the temporary uncertified asymmetric key pairs of both principals are transmitted in clear, and no long-term key is involved in the calculation of the session key. Only the principals *A* and *B* which are in possession of secret private components of the short-term asymmetric key pairs, i.e. *APriKeyX* and *BPriKeyX*, respectively, can derive the true authentic session key K_{AB}. Another rather trivial requirement for forward secrecy is that the short-term secrets of both parties should be securely discarded after the completion of the corresponding session.

Now, we consider another alternative protocol for forward secrecy, which is also described using the same abstract level notation.

Protocol 44. *An abstract level description of an alternative protocol for forward secrecy*

1. A $\rightarrow$ B: *APubKeyX*
2. A $\leftarrow$ B: $APubKeyX\{r_B\}$

A, B: $K_{AB} = h(APubKeyX, r_B)$ or alternatively $K_{AB} = r_B$ when key transport scheme is preferred.

This second prototype for forward secrecy is based on *confidentiality of a random nonce* r_B chosen by the principal *B*. Here, the temporary public key *APubKeyX* is used for temporary encryption of r_B. On receipt of this encrypted r_B, the principal *A* can decrypt and recover r_B using *APriKeyX* as the decryption key. This ephemeral characteristic of the encryption of a random secret is the source of forward secrecy.

The critical difference between the previous prototype derived from Diffie-Hellman scheme and this one is that the former depends on the special feature of a key agreement function *F* with an elegant symmetric property, whereas the latter relies upon confidentiality using a temporary public key. Therefore, the first prototype can be implemented only by a cryptosystem which satisfies the requirement of the key agreement function *F*. Presently, only Diffie-Hellman key exchange in various suitable groups is known to be such a system.

On the other hand, the second prototype can easily be applied to *any asymmetric cryptosystem*, which does not have to be Diffie-Hellman system. The random nonce r_A for temporary encryption from *A* to *B* may also be used as a challenge value for *A* to authenticate *B*. The response value then has to be an indication that *B* has used its own private key to generate it.

It is interesting to note that the second prototype can be deployed into both key *agreement* and *transport* schemes, unlike the first one which allows only key agreement.

We consider two instance protocols which are easily derived from the second prototype. The following protocols are an application of the prototype to discrete log based cryptosystem and RSA cryptosystem, respectively.

Protocol 45. *Discrete log cryptography based implementation of the second prototype*

1. A $\rightarrow$ B: g^{r_A}
2. A $\leftarrow$ B: $g^{r_A r_B}$

A: $K_{AB} = h(g^{r_A}, g^{r_B}) = h(g^{r_A}, (g^{r_A r_B})^{r_A^{-1}})$
B: $K_{AB} = h(g^{r_A}, g^{r_B})$

Protocol 46. *RSA cryptography based implementation of the second prototype*

1. A $\rightarrow$ B: e_A, n_A
2. A $\leftarrow$ B: $(r_B)^{e_A} \bmod n_A$

A, B: $K_{AB} = h(e_A, r_B)$

Even though the example protocol shown in Protocol 45 is very similar to the original Diffie-Hellman protocol, it should be noticed that this is just an example of the prototype, and any different kind of public key mechanisms can be adopted to provide forward secrecy. Of course, as with the prototype protocols, these examples must be used together with authentication in order to achieve a *secure* protocol. It is worth noting that the authentication required can be achieved through either symmetric or asymmetric cryptography, so it should not be assumed that forward secrecy requires asymmetric long-term keys. Indeed, several password based protocols, [Jabl96], [Wu98], in which the long-term key is a (short) shared secret, or a value derived from it, have been proposed and provide forward secrecy. Whilst it may be of questionable value to consider the efficiency of incomplete protocols, it is also interesting to note that the example based on RSA can be more efficient than that based on discrete log for entity *B* if the value e_A is chosen to be a small value such as is often recommended. However, we must note that a new temporary RSA key pair must be generated by *A* for each session.

In fact, we have noticed that basically the same concept of forward secrecy based upon RSA had already been described by Wiener [Wien98]. His description, however, still only concerns the implementation of forward secrecy in terms of a particular cryptographic *algorithm* like RSA rather than *protocols* using any asymmetric cryptography. This is likely to lead to a rather misleading conclusion that RSA (or even the second prototype itself) is always a more expensive way to achieve forward secrecy than Diffie-Hellman (or the first prototype). Furthermore, Wiener seems to ignore the possibility that the use of ephemeral encryption for forward secrecy (i.e., the second prototype above) can be just as efficient as the first prototype even in the case of discrete-log cryptosystems. As we can see from Protocol 41 and Protocol 45, the computational cost for forward secrecy is the same for both prototypes, that is two exponentiations in each principal. The choice of the suitable prototype for forward secrecy may be made according to the application scenario rather than the particular cryptographic primitives used in the protocols.

It is an obvious question to ask whether there are other protocols providing forward secrecy that do not fit into the two prototype constructions discussed in this section. It seems that forward secrecy can be provided only through use of a *one-way function*, so

that later the session key cannot be recovered because this function cannot be inverted. Because of its special algebraic properties, exponentiation modulo a prime is a suitable one-way function. Naturally other similar functions, such as point addition in an elliptic curve group can equally be used, and these provide natural generalisations of the original Diffie-Hellman key exchange protocol. Therefore any other one-way function with suitable algebraic properties may be candidates for alternative protocols providing forward secrecy, although there are no such functions currently available to our knowledge. The second prototype makes use of the trapdoor one-way function underlying any asymmetric cryptosystem. In this case the function cannot be inverted once the trapdoor is deleted. Therefore, although we offer no further evidence, we conjecture that forward secrecy can only be provided by protocols which either have similar algebraic properties as modular exponentiation (prototype one) or use trapdoor one-way functions (prototype two).

4.1.2 Two Prototypes for *Partial* Forward Secrecy

It seems that we do not need to insist only upon *complete* forward secrecy in some application environments. Instead, we may consider a sort of *imperfect* or *partial* forward secrecy if it is more computationally effective. There may be, for example, a particular communication type where only one of two peer entities is really concerned about confidentiality of the past data and/or the other entity's long-term private key is more likely to be compromised. In this situation, it may be reasonable to consider protection against the long-term key compromise of only one principal of two peers, which we may call *partial* forward secrecy as opposed to the usual *complete* forward secrecy.

The same reasoning for prototypes of forward secrecy can be directly applied to identify the prototypes for partial forward secrecy. The two prototypes shown below are partial forward secrecy analogues of the previous ones.

Protocol 47. *The first prototype for partial forward secrecy*

1. A ← B: *BPubKeyX*

A: K_{AB} = F(*BPubKeyX*, *APriKey*)
B: K_{AB} = F(*APubKey*, *BPriKeyX*)
where F is a common key agreement function with the following property:
F(*BPubKeyX*, *APriKey*) = F(*APubKey*, *BPriKeyX*)

Protocol 48. *The second prototype for partial forward secrecy*

1. A ← B: $APubKey\{r_B\}$

A, B: $K_{AB} = h(m, r_B)$ or alternatively $K_{AB} = r_B$ when key transport scheme is preferred.

Notation:
m: an optional key input data to the hash function
(m should be agreed previously between A and B in the more broad context of the relevant whole key establishment protocol which includes this forward secrecy scheme)

Here in both prototypes, we assume that *B* has an authentic copy of *A*'s long-term public key, *APubKey*. The only difference from *complete* versions is that *A*'s short-term private key is replaced with its long-term authentic public key. The computation procedures of the session key, K_{AB} take *A*'s long-term public key, but not that of the principal *B*. Therefore, the compromise of *B*'s long-term private key alone does not cause the compromise of the session key. It should be noticed, however, that these prototypes, including both partial and complete cases, do not guarantee any other security goals except forward secrecy. It is the whole integration of elements in a protocol that provides the ultimate security goals including entity authentication.

4.1.3 Modification of the ASPeCT Protocol for Forward Secrecy

In this section we apply both prototypes of forward secrecy to the ASPeCT protocol, as described already in Section 2.1.6, to derive two slightly different variant protocols. Significant effort was expended to make the ASPeCT protocol satisfy a demanding set of security goals and computation efficiency at the same time. Through a saving of the non-repudiation property of the network to the user, the resultant computational load is significantly lower than that of the STS protocol. One additional feature omitted from this protocol is forward secrecy which is supported in STS protocol. This difference comes from the session key generation of the two protocols. The ASPeCT protocol uses

a similar but subtly different method from the Diffie-Hellman key computation. The STS protocol complies with the original Diffie-Hellman form $g^{r_A r_B}$ where two factors of the exponent are the random nonces generated within two principals. Instead, ASPeCT protocol uses only one nonce r_A and the private key b of the network B. In this way, the protocol saves some public key based computations and succeeds in turning on-line exponentiation within the user terminal A into off-line because A may take the advantage of the pre-knowledge of B's public key g^b in most cases. For the sake of simplicity, we only refer to [HoPr98] for detailed security analysis of the protocol.

If the long term private key b of the VASP (or network) is compromised and all the protocol transcripts for a particular session are recorded (for the knowledge of g^{r_A} and r_B) by an attacker, the session key for the session is easily disclosed to the attacker, and so no forward secrecy is provided in this protocol. Of course, the disclosure of the private key of the mobile side alone does not lead to the disclosure of the session key (so partial forward secrecy is satisfied), but in that case, the real problem is more authentication rather than forward secrecy. Moreover, *forward secrecy concerns the user who cannot ensure that the private key of the VASP would not be compromised.*

It is surprising that the ASPeCT analysis of protocols has not considered the issue of forward secrecy at all. The likelihood of compromise of a long-term key is difficult to assess. With regard to a trusted network operator this probability may be regarded as sufficiently low. However, when it concerns any VASP, users may not have confidence that long term keys are sufficiently secure. It should be noted that if the long-term key of a VASP becomes compromised then all transactions made with that VASP during the lifetime of that key may be available to an eavesdropper, unless forward secrecy has been provided. Long term keys may have lifetimes of several months, so such an attack could be very attractive to the attacker.

We can apply the two prototypes of forward secrecy to the ASPeCT protocol. The resulting variants require more modulo exponentiations but guarantee forward secrecy. The first protocol is derived from the first prototype.

Protocol 49. *A modified ASPeCT protocol(FS1) to provide forward secrecy based on the first prototype*

A: user, B: network or VASP

1. A $\rightarrow$ B: g^{r_A}
2. A $\leftarrow$ B: g^{r_B}, $h_2(K_{AB}, g^{r_B}, B)$
3. A $\rightarrow$ B: $\{\{h_3(g^{r_A}, g^b, g^{r_B}, B, K_{AB})\}_{K_A^{-1}}, A\}_{K_{AB}}$

A, B: $K_{AB} = h_1(g^{r_A r_B}, g^{br_A})$

The only difference between this modified version and the original ASPeCT protocol is that g^{r_B} is delivered from B to A instead of r_B, and $g^{r_A r_B}$ is used as a key input to the hash function instead of r_B. The authentic key establishment in the original protocol comes from the facts that (1) from the viewpoint of B, a key input g^{r_A} from A is included in the signed message of A and the fresh nonce r_B of B's own is used for key computation; (2) from A's viewpoint, the authentic copy of B's private key b and A's own fresh nonce r_A are used together to compute the session key; and (3) from any attacker's viewpoint, he cannot compute a key input g^{br_A} with the knowledge of both g^b and g^{r_A}.

Now, the replacement of r_B with g^{r_B} does not affect the protocol with respect to the above three considerations, and helps both principals establish a secret session key satisfying forward secrecy. Note that the compromise of the long-term private key does not lead to the disclosure of the session key due to the inclusion of a Diffie-Hellman form, $g^{r_A r_B}$ in the key computation. The computational cost for forward secrecy in the user (A) side is one additional exponentiation to calculate $g^{r_A r_B}$ (still computationally more effective than the STS protocol), and in the network or VASP side, two additional exponentiations to compute $g^{r_A r_B}$ and g^{r_B} (roughly the same computational cost as the STS protocol). The term g^{br_A} is still required in the key computation because this term plays the essential role in authentication of B to A.

The second protocol is derived from the second prototype for forward secrecy using confidentiality.

Protocol 50. *Another modified ASPeCT protocol (FS2) to provide forward secrecy based on the second prototype*

A: user, B: network or VASP

1. A $\rightarrow$ B: g^{r_A}
2. A $\leftarrow$ B: $g^{r_A r_B}$, $h_2(K_{AB}, g^{r_A r_B}, B)$
3. A $\rightarrow$ B: $\{ \{ h_3(g^{r_A}, g^{b}, g^{r_A r_B}, B, K_{AB}) \}_{K_A^{-1}}, A \}_{K_{AB}}$

A, B: $K_{AB} = h_1(g^{r_B}, g^{b r_A})$

In this variant, r_B in the second message and the key computation of the original protocol is replaced by $g^{r_A r_B}$ and g^{r_B}, respectively. This modification does not compromise the protocol with respect to the three considerations as described for the first variant, and enables both principals to establish a secret key satisfying forward secrecy in a different way from the first variant. Retrieval of the term g^{r_B} in the A side requires the knowledge of r_A which can be viewed as a private component of the temporary asymmetric pair (r_A, g^{r_A}). The lifetime of the temporary asymmetric key pairs and r_B should be at most as long as the corresponding session, and hence g^{r_B} and the session key is free from the compromise of the long-term private keys of A and B. In other words, the secure deletion of r_A and r_B in both sides after the session thwarts any effort to retrieve the value of g^{r_B} because it was delivered to A after being encrypted using the temporary public key g^{r_A} of A.

Here again, compared to the original ASPeCT protocol, the computational cost for A is one more modulo exponentiation to retrieve g^{r_B} from $g^{r_A r_B}$, and for B, two more exponentiations to generate g^{r_B} and $g^{r_A r_B}$, which is exactly the same as the first variant for forward secrecy. This clearly shows that the choice of a particular prototype of forward secrecy does not necessarily lead to more expensive or cheaper implementation.

4.1.4 Instance WAKE Protocols with Forward Secrecy: FS1/2

Protocol 51. *Instance protocol*: *Forward secrecy variant 1* (*FS1*)

A: user, *B*: network

1. A $\rightarrow$ B: $(g^b)^{r_A}$
2. A $\leftarrow$ B: g^{r_B}, $h(g^{r_A}, K_{AB})$
3. A $\rightarrow$ B: $\{h(B, g^{r_B}, g^{r_A})\}_{K_A^{-1}}$, $\{A\}_{K_{AB}}$

A, B: $K_{AB} = g^{r_A r_B}$

Protocol 52. *Instance protocol*: *Forward secrecy variant 2* (*FS2*)

A: user, *B*: network

1. A $\rightarrow$ B: $(g^b)^{r_A}$
2. A $\leftarrow$ B: $g^{r_A r_B}$, $h(g^{r_A}, K_{AB})$
3. A $\rightarrow$ B: $\{h(B, g^{r_B}, g^{r_A})\}_{K_A^{-1}}$, $\{A\}_{K_{AB}}$

A, B: $K_{AB} = h(g^{r_A}, g^{r_B})$

In this section, we describe two instance protocols Protocol 51 (*FS1*) and Protocol 52 (*FS2*) which have been designed to provide forward secrecy. Their design is based on the result of investigation of all the possible archetypes of forward secrecy, which are described in detail in Section 4.1.1.

These two protocols also have been derived from the same generic protocol variant A. The only differences between these two are in the way that key agreement and forward secrecy are achieved. In the instance protocol *FS1*, we use Diffie-Hellman key agreement function to obtain forward secrecy (i.e., the first prototype for forward secrecy) , whereas in *FS2* we adopt the second prototype for forward secrecy : *use of temporary public key encryption of key input*. That is, the challenge[6] g^{r_B} from *B* to *A*, is delivered to *A* encrypted under g^{r_A} which can be thought of as the public part of the temporary asymmetric key pair (r_A, g^{r_A}) of *A*. This temporary asymmetric pair is deleted after the current session, and hence there is no way to retrieve the value of g^{r_B},

[6] Note that not r_B but g^{r_B} is delivered to *A* encrypted under *A*'s temporary public key g^{r_A}. Moreover, r_B is never disclosed to the principal *A*. Therefore we regard g^{r_B}, not r_B, as a challenge value from *B* to *A*.

which is a key input data, from the recording of the protocol messages. Note that the Diffie-Hellman key agreement function could be used instead of hashing in *FS2*, but will be just a waste of computation in this protocol.

4.1.5 Complexity of the WAKE Protocol with Forward Secrecy

Forward secrecy is an expensive cryptographic service, and hence should be addressed from a strategy of flexibility and multiple level security. This viewpoint might lead to a *protocol suite* which consists of alternative selective protocols. It may provide a great deal of flexibility to security service implementation for future mobile communication environment, where a wide range of telecommunication services will be available, and so different levels of security strength are highly desirable. Therefore, when analyzing and comparing complexity of the different security protocols, we should be careful that we do not apply only one absolute measure of complexity. Instead complexity must be measured against what security services the protocols provide.

In this section, we compare the new instance WAKE protocols *FS1* and *FS2* with the other instance WAKE protocol with no forward secrecy (henceforth *NFS*), the ASPeCT protocol and the two modified ASPeCT protocols *FS1* and *FS2* as described in Section 4.1.3, in terms of computational cost.

We only describe the computational cost for the proposed instance protocol *FS2* in Figure 4-1. The cost for *FS1* is the same as *FS2*. As shown in two boxes in the figure, forward secrecy requires additional computations, that is, one more on-line exponentiation in the user side, and one on-line and one off-line exponentiation in the network side. In Table 4-1, we summarize the required number of off-line and on-line exponentiations for all protocols. From this table, we can see that the our new WAKE protocols *FS1* and *FS2* are exactly the same as the modified ASPeCT protocols *FS1* and *FS2* in terms of the required number of exponentiations.

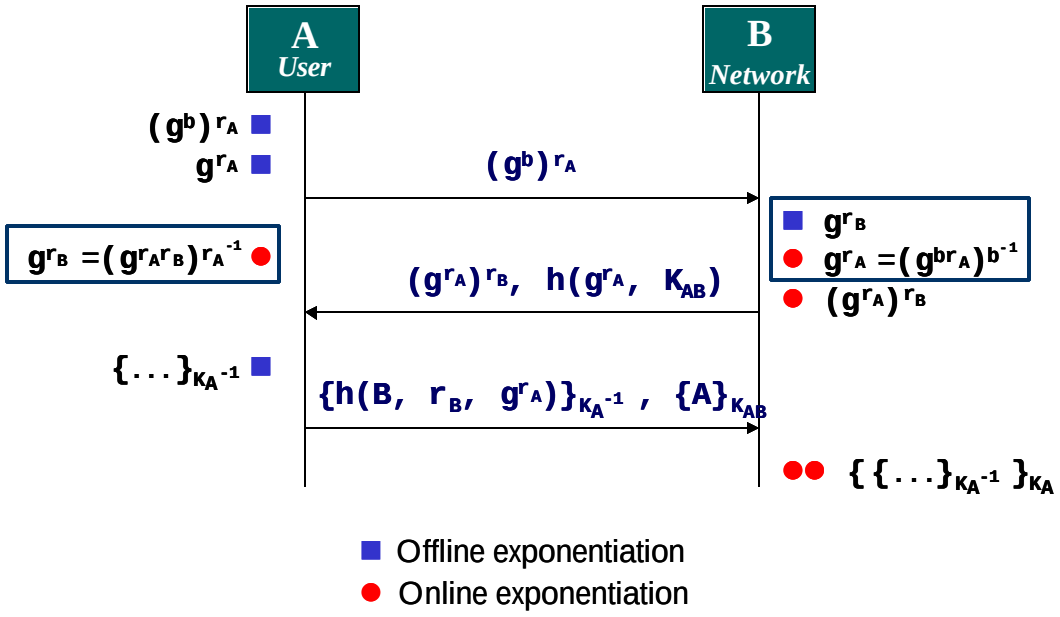


Figure 4-1: The number of exponentiations required to execute the protocol *FS2*

Table 4-1: Computational cost comparison of the ASPeCT protocols and the proposed instance protocols

Protocol	***Number of exponentiations***		***Forward secrecy***
	***User* (*A*)**	***Network* (*B*)**	
ASPeCT *original*	■ ■ ■	● ● ●	No
ASPeCT *modified FS1*	■ ■ ■ ●	■ ● ● ● ●	Yes
ASPeCT *modified FS2*	■ ■ ■ ●	■ ● ● ● ●	Yes
WAKE protocol *NFS*	■ ■ ■	● ● ●	No
WAKE protocol *FS1*	■ ■ ■ ●	■ ● ● ● ●	Yes
WAKE protocol *FS2*	■ ■ ■ ●	■ ● ● ● ●	Yes

Notation: "■ ■ ■ ●", for example, means three *off-line* and one *on-line* exponentiations

4.2 Denial-of-Service (DoS) Attacks

Recently, DoS attack is becoming a growing concern as the Internet services have been used in more aspects of human life. Many things in human life, turned out to have their counterpart in the Internet world: DoS attack would be one example of them. In this research, we focus on the most typical DoS attacks which may be called *connection depletion attacks* or *resource clogging attacks*: an attack in which an attacker seeks to initiate and leave unresolved a large number of connection requests to a Web server, exhausting its resources and rendering it incapable of servicing legitimate connection (or service) requests. SYN flooding attack in TCP/IP networks is the most well known example of this kind [Cert96, Fred99]. This attack exploits a weakness in the TCP connection establishment protocol. Attempting to establish a TCP connection, the client sends the server a SYN message. In response, the server sends a SYN-ACK message, and prepares the connection by allocating buffer space. The client then finishes establishing the connection by responding with an ACK message. After this sequence, both entities can exchange the service-specific data. The attacker, however, does not follow the above sequence of messages. He simply fails on purpose to send the third message, namely ACK to the server, leaving the session half-open. The attacker may initiate large amounts of SYN messages simultaneously, causing the server to be unable to handle the legitimate connection requests. A detailed analysis of this attack and possible remedies are described by Schuba et al. [Schu97].

Using an authentication protocol in Internet environment is orthogonal to prevention of DoS attacks. Authentication protocols themselves do not help prevent DoS attacks but instead may give rise to another environment for the attacks. Usually to run an authentication protocol, the involved entity has to assign to it a particular session and some memory to keep relevant data resulting from message exchanges and related computation during the execution of it. Thus, although the notorious SYN flooding attacks can be minimized through careful design and operation of the Internet communication systems, the introduction of authentication protocols just opens up another door to similar denial-of-service attacks.

This problem concerning authentication protocols and DoS attacks is well understood and a lot of previous work is invested to address it; a detailed survey of the

related work can be found in [ANL01, LNA00]. The most well studied and promising approach to date seems to be for the server to use *cookies* against a potential attacker. The concept of *cookies* for use in the context of client-server transactions started from "Netscape Cookie" in 1994 as part of the feature set of Netscape Version 1.1 [Laur98]. Since then, most Web browsers including Microsoft Explorer adopted cookies.

Cookies are pieces of information generated by a Web server and stored in the user's computer, ready for future access [Scho99b]. Basically the same concept of cookies started to be used to thwart DoS attacks on cryptographic protocols, the first example of which seems to be Photuris protocol by Karn and Simpson (most recent version 1999 [KaSi99] but originally published 1995). Several Internet security protocols followed this trend, including SKEME [Kraw96], OAKLEY [Orma98]. The basic idea of cookies in these protocols is as follows. When a client attempts to make a connection the server sends back a *cookie* which is a function of a secret known only to the server and other information unique to the particular connection. At this stage the server stores no state for this request. The client needs to return the cookie in the next message and its validity can be checked by the server from the information sent and its secret. The idea is to ensure, before investing significant resources, that the client is making a unique request for connection. This technique addresses DoS attacks in which the adversary sends random connection requests.

The benefits of *stateless* connections in the beginning of an authentication protocol were recognized by Janson et al. [JTY97] in the KryptoKnight protocol suite, and this concept was generalized by Aura and Nikander [AuNi97]. Their idea is to make the client store all the state information required by the server and return it to the server as necessary with each message sent. In this way the server need not store any state information. The cookie approach can be considered a special instance of the stateless connection approach in the sense that a cookie generated by the server can contain a session specific information, is stored in the client system, and later delivered back to the server to be verified.

A *cryptographic puzzle* for the client to solve to initiate a connection with the server is another approach to solving DoS attack problems. Dwork and Naor [DwNa98] first presented this concept in the context of electronic junk mailing, and later Juels and Brainard [JuBr99] presented a simpler client puzzle for the server to combat TCP SYN

flooding attacks. The same concept was further developed by Aura et al. [ANL01] to address DoS attacks against authentication protocols. In this scenario, the server in an authentication protocol can ask the client to solve a puzzle before the server creates a protocol state or computes expensive public-key related computations. In this way, the puzzle helps improve the DoS-resistance of an authentication protocol.

It can be seen that each of the countermeasures against DoS attacks that we have described carries some cost. If cookies are used as an initial stage in an protocol then additional message exchanges are usually required; this can be a significant overhead in some applications such as the limited signalling channels in mobile communications. Making a protocol stateless may require significant changes to the protocol structure and also increases storage and bandwidth requirements on the client side. Finally the use of cryptographic puzzles imposes a computational burden on both client and server as well as requiring additional message exchanges.

In this section, we propose a new countermeasure against DoS attacks for client-server security protocols in which *the client A authenticates the server B by sending a random nonce encrypted under the public encryption key of the server*. In other words, the countermeasure is only for the authentication protocols adopting $DA_{F,ACK}$ or $DA_{F,NoAck}$ for *B* to *A* authentication. Such protocols include the new WAKE protocol designed in the thesis, SSL/TLS [RFC99], SKEME [Kraw96], and the authentication and key agreement protocol of the PACS (Personal Access Communication System), one of the six PCS standards in North America [Bell94], [JTC94].

Our approach is on the same line of Aura and Nikander's stateless connection concept in that both approaches purport to make the cryptographic protocols themselves more robust against the attacks. Our approach has something in common with the client puzzle concept in that both use some cryptographic mechanisms to combat the DoS attacks, but differs in that our method can apply and be integrated directly into the authentication and key-establishment protocols themselves. This provides a mechanism for the design of more robust protocols. Our scheme requires only a minimal overhead on both the client and the server. The only limitation of the new method is its usage applies only to a specific type of authentication protocols as stated earlier. It should be noted, however, that the new method as well as other approaches described above can never stop DoS attacks but only reduce the burden caused by the attacks.

For further discussion, we classify the DoS attack into the following sub-categories.

- **DoS attack 1**: an attacker initiates a bogus connection request message which contain *garbage values*, and does not send any further messages. This attack can be addressed by a proper checking mechanism in some protocols as shown below for the case of the proposed protocols in this project. Equipped with the relevant checking, the garbage values can be detected and the corresponding session is cut off.
- **DoS attack 2**: an attacker initiates a bogus connection request message which contain *correct values* complying with the protocol, and does not send any further messages. The significance of this attack is that the checking mechanism for the DoS attack 1 does not help any more; more computation is wasted in the network than for attack 1 (the same goes for the attacker) and the corresponding session is left open until it is closed by the network itself.

To cope with the connection depletion attack, we may consider such a concept as the *enforcement of heavy computation*. We seek a way to enforce heavy computation upon the user to initiate the first message, but those required computations should not cause additional computational load in terms of modular exponentiation. Another method useful against the attack is to use *weak key confirmation*: a key confirmation where a principal (here, a network/VASP) only shows the other principal (here, a user) that it *is able to* compute the common new session key. The heavy computation technique is useful for both the DoS attack 1 and 2, whereas the weak key confirmation is only applicable to DoS attack 2.

4.2.1 Enforcement of Heavy Computation

The protection scheme against the DoS attack which is to be added as an add-on facility to the WAKE protocol should not cause computational overhead in terms of public key computation. It should be able to be applied, with virtually zero cost and modification, to the WAKE protocol. The heavy computation is not intended to be an additional cost but required in the WAKE protocol even without the add-on facility for the DoS attack protection. The basic idea is that the user is required to pay due cost which is determined by the protocol description.

4.2.1.1 Server Authentication and Random Numbers

To authenticate the server with any cryptographic challenge-response mechanism, the client chooses a random number and sends it to the server. According to the way this random challenge is handled, we may have two different methods of authentication. The first is that the client can send it in the clear and then the server signs over it with its own certified private key. The corresponding public verification key is available publicly and so the client can check whether the signature was generated by and came from the server. The unpredictability and randomness of the random challenge guarantees the required freshness of the signature: i.e., the server has generated the signature for the current session, not for another old session.

The second alternative is to encrypt the random number under the public encryption key of the server before delivery to the server. The authentic server is then the only entity to be able to retrieve the random number from the ciphertext. The server's response to the client with the decrypted random number provides the authenticity of the server's identity.

Each of the above two schemes has its own strengths and weaknesses. As far as DoS attack is concerned, however, the latter method is preferable. This is because in the latter method the random number from the client is not just a random number but an encrypted message thereof, which may be exploited to accommodate a countermeasure against the DoS attack. The basic idea of the countermeasure is to implant a *cryptographic salt*[7] or a random number chosen by the server in the public-key encryption operation by the client. That is, the client is required to encrypt a random nonce which he received from the server as well as his own fresh nonce. This is quite an unusual usage of random nonce encryption in public-key based authentication protocols. On receipt of the encryption message of random nonces, the server is able to check whether the message has been formed correctly since it leads to the successful retrieval of the server's random nonce after decryption *only when the message has been formed correctly*.

[7] A similar term "password salt" is used in the Unix environment. The salt in this environment is usedtogether with the user password to make dictionary attacks less effective. Details of this technique are described in [MvOV97].

Figure 4-2 depicts this concept in more detail. In the above figure, we assume that the client authenticates the server by sending the second message which is encrypted under the server's public encryption key, K_B. Furthermore, it should be noted that the first two messages just comprise a part of an authentication and key establishment protocol, which we want to make more robust against DoS attacks. The steps in this scheme can be outlined as follows.

1. The server *B* chooses a random number r_B and sends it to the client *A*.
2. On receipt of r_B, the client chooses its own random number r_A and encrypts it together with r_B using the server's public key K_B; the resulting ciphertext $\{r_A, r_B\}_{K_B}$ is sent back to the server.
3. On receiving the encryption message, the server decrypts and retrieves r_B and r_A from the received ciphertext. The value of the retrieved r_B and the value of r_B which has been sent to the client should match; otherwise the server concludes that the received message is simply a garbage value sent by a malicious attacker.

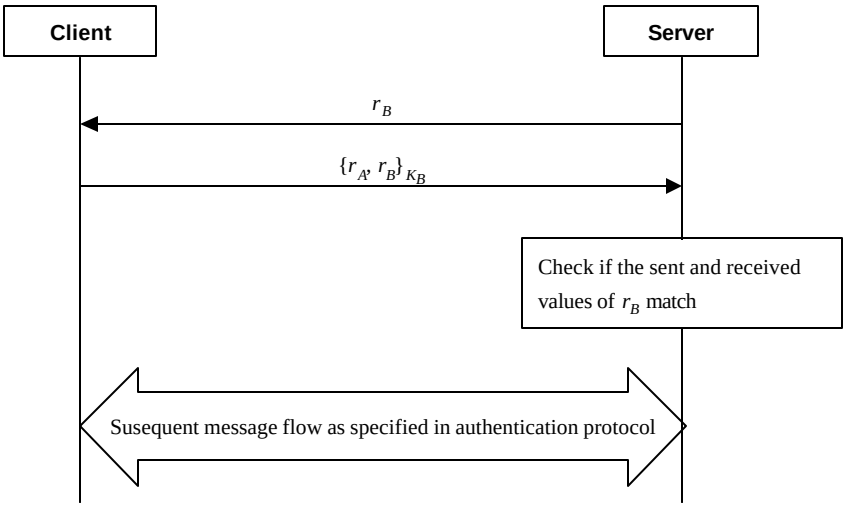


Figure 4-2: A random number can be used as a kind of cryptographic salt to combat the DoS attack.

Without using this kind of countermeasure, there is no way for the server to check whether the received ciphertext is really the result of a proper cryptographic computation and whether the computation has occurred for the current session. Otherwise even for a garbage or old message attack the server will execute a public key

computation for decryption, send the subsequent message to the attacker, and finally will result in a state of the session left open waiting the next message from the attacker, which is simply given up by the attacker. It can be seen that in a protocol in which the client already sends the challenge r_A encrypted using the server's public key K_B there is only a small change in the protocol messages and minimal additional computational effort required. This is in contrast to other DoS prevention mechanisms which may require additional messages, extra computation, and/or significant alterations to the protocol specification. In practice it is often possible to include the challenge from the server in an existing message, as we will see below.

In the next section, we demonstrate that the above technique can be easily applied to a typical Internet security protocol SSL/TLS where the *ServerHello* message and the *ClientKeyExchange* message correspond to the first and the second messages, respectively.

4.2.1.2 SSL/TLS Protocol

The SSL protocol has become a de facto standard for the Internet security, and its latest version 3.0 is used as the core protocol TLS by the IETF Transport Layer Security working group. The SSL/TLS protocol uses public key cryptography for authentication and key-establishment. Some analyses of cryptographic security of the protocol have been published, such as Paulson's formal inductive analysis [Paul99], and Wagner and Schneier's informal analysis [WaSc96]. Both analyses concluded that the protocol has no weakness with regard to its basic structure. We show below its simplified abstract description which is adopted from Paulson's abstract version of the protocol, where optional messages are boxed in dotted line.

Protocol 53. *A simplified description of the TLS handshake protocol*

A: Client, B: Server

1. A $\rightarrow$ B: A, r_A, Sid, Pa **client hello**
2. A $\leftarrow$ B: $r_B, Sid, BCert, Pb$ **server hello, server certificate**
3. A $\rightarrow$ B: $ACert$, $\{ r_A' \}_{K_B}$, $\{ hash(r_B, B, r_A') \}_{K_A^{-1}}$, $\{ finished \}_{K_{AB}^{A}}$
   client certificate, client key exchange, certificate verify, client finished
4. A $\leftarrow$ B: $\{ finished \}_{K_{AB}^{B}}$ **server finished**

A, B: $M = hash(r_A, r_B, r_A')$, $finished = hash(Sid, M, r_A, r_B, Pa, A, Pb, B)$

Here we use slightly different notation from Paulson's description of the TLS protocol; r_A and r_B replaces the original *Na* and *Nb* called client random and server random, respectively. Another random nonce $r_A^{'}$ denotes the pre-master-secret (*PMS*), which serves as a challenge data to the server *B*. The public key certificates of the client and the server are denoted as *ACert* and *BCert*, respectively. *Sid* means the session identifier. The notations $\{\bullet\}_{K_A^{-1}}$ and $\{\bullet\}_{K_B}$ stand for the message encryption under *B*'s public encryption key K_B and the signature with the *A*'s private signature key. Using r_A, r_B and *M*, the principals *A* and *B* compute the session keys K_{AB}^{A} and K_{AB}^{B} to be used for *A*-to-*B* and *B*-to-*A* encryptions, respectively. Whereas, *Pa* and *Pb* comply with the original notation, which mean the sets of *A* and *B*'s preferences for encryption and compression, respectively.

We can see that r_B in the message 2 of the SSL/TLS protocol, and $\{r_A^{'}\}_{K_B}$ can serve a good vehicle for the countermeasure described in the previous section. That is, $\{r_A^{'}\}_{K_B}$ can be modified to $\{r_A^{'}, r_B\}_{K_B}$. The countermeasure is very reasonable mechanism worthy to be considered for the SSL protocol because there is no additional public-key encryption/decryption required. Furthermore, it should be noted that the concatenation of $r_A^{'}$ and r_B is not the only way to implement the idea of the countermeasure. For instance, instead of $\{r_A^{'}, r_B\}_{K_B}$, we can adopt, for example, the following alternative:

$$\{r_A^{'} \oplus r_B\}_{K_B},\ hash(r_A^{'}).$$

In this way, we can keep the length of the encrypted message as the original one. The server decrypts the received ciphertext and subtracts the value of r_B from the decrypted value and takes hash value of it, comparing it with the received value of hash. The benefit of this countermeasure can be made clearer by comparing the significance of DoS attacks for both cases: the original protocol and the modified one, as shown in Table 4-2.

Table 4-2: Comparison of the original SSL/TLS and the modified protocols

	Original SSL/TLS	Modified SSL/TLS
After a DoS attack the server has spent	*one* decryption and *one* or *two* signature verifications	*one* decryption
and is left in a state of	*one* half-open session.	*no* half-open session.

Here both decryption and verification are public-key based operations, and original protocol requires two signature verifications when the client authentication is needed: one for the client certificate verification and another for the client signature verification. The countermeasure cannot prevent DoS attacks completely, but significantly mitigates the damage of the attacks with no additional public key operation or extra message exchange at all.

It is very important to note that the cryptographic salt explained so far is distinct from the idea of cookies. A cookie is a function of session specific information whereas a cryptographic salt is simply a nonce chosen arbitrarily by the server. Both ideas, however, may be combined as shown in the next section.

4.2.1.3 Cookies combined with the new countermeasure

The random number r_B in the countermeasure can be generated in a way similar to *cookies* in the Photuris protocol, thus enabling the server to achieve even more robustness against DoS attacks. Usually at the point of the delivery of r_B the server is expected to assign a unique session to the service requesting client. In this situation, a particular value of r_B is also uniquely related to the corresponding session. The value of r_B is stored in a memory within the server system to be compared with the received value of r_B from the client. The problem of this scheme is very similar to that of TCP/IP based client-server model which leads to the notorious SYN flooding attacks. In other words, the server must wait for the second message in the above figure after it sends r_B to the client. This problem can be avoided by the server delaying the assignment of a particular session resource to the client until the client proves that he has correctly carried out the encryption of the two random nonces. In other words, the server does not couple a specific value of r_B with a particular client before the client computes and sends the required cryptographic message.

To obviate the need to store the values of r_B, the server prepares a suitable hash function H, selects a random master key K_{master} and selects a sufficiently large value as the modulus M of the index for r_B. Here, the index runs from 0 to $M - 1$. When a new value of r_B is required, the server runs the hash function with the master key and the current index as the inputs, the hash result of which will be used as the value of r_B as shown in Figure 4-3.

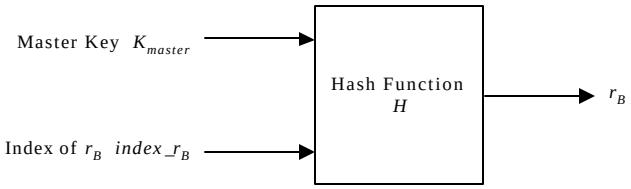


Figure 4-3: Generation of random number r_B

The following Figure 4-4 shows an example using this r_B generation method together with the countermeasure described before.

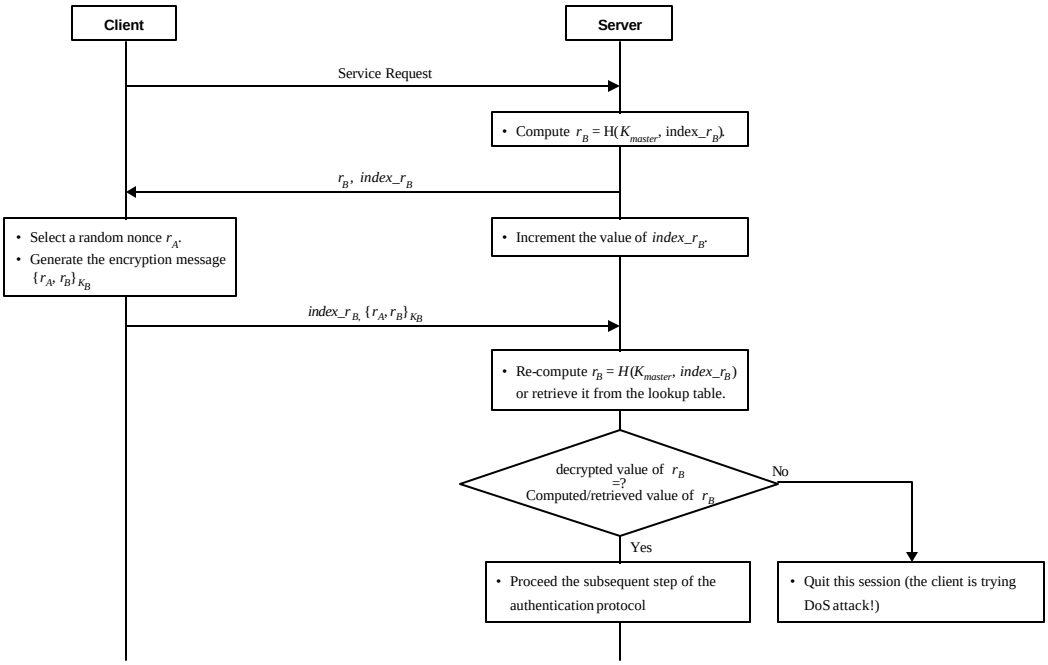


Figure 4-4: The cryptographic salt r_B as a cookie

The process is outlined in the following steps.

1. In response to a service request from the client, the server generates a new value of $r_B = H(K_{master}, index\_r_B)$, increments the index parameter $index\_r_B$, and sends the client the values of r_B and $index\_r_B$.
2. On receipt of r_B and $index\_r_B$, the client generates his own random nonce r_A, encrypts r_A and r_B under the public encryption key K_B, and sends the server the plaintext $index\_r_B$ and the ciphertext $\{r_A, r_B\}_{K_B}$.
3. When the server receives the response from the client, using the received value of the parameter $index\_r_B$, it retrieves from a look-up table or, alternatively, re-computes the corresponding value of r_B. The server also decrypts the received ciphertext $\{r_A, r_B\}_{K_B}$, and retrieves the value of r_B, which is compared with the value of r_B which was generated by itself using the given value of $index\_r_B$.
4. If both values match, the server is assured that the client has formed the ciphertext honestly and sent the ciphertext $\{r_A, r_B\}_{K_B}$. This leads the server to the next step specified in the authentication protocol to which the protection scheme is applied.
5. On the other hand, if the match fails, the server may conclude that the client is trying DoS attack by sending a bogus message which has nothing to do with the correct cryptographic operation to compute the cipher text $\{r_A, r_B\}_{K_B}$.

The usage scenario of r_B as a cookie is just an example, and there may be as many usages as different uses of cookies. It should be noted, however, that the basic idea of the new countermeasure presented in this research is rather independent from cookies. In other words, the cryptographic salt r_B may or may not be cookies. Rather, the new countermeasure can be more effective when combined with the cookie scheme.

4.2.1.4 Application to the WAKE protocol

We can easily modify the instance protocol NFS to enforce heavy computation on the attacker, and similarly for another variants, FS 1/2 for forward secrecy. The following shows a modified protocol of the original instance protocol NFS.

Protocol 54. *Instance protocol NFS modified for the enforcement of heavy computation*

A: user, B: network

1. A $\rightarrow$ B: $(g^b)^{r_A}$, $h(g^{r_A}, B)$
2. A $\leftarrow$ B: r_B, $h(g^{r_A}, K_{AB})$
3. A $\rightarrow$ B: $\{h(B, r_B, g^{r_A})\}_{K_A^{-1}}$, $\{A\}_{K_{AB}}$

A, B: $K_{AB} = h(g^{r_A}, r_B)$

The field enclosed in a dotted box shows the addition to the original protocol. Any user **A** is to compute two exponentiations, $(g^b)^{r_A}$ and g^{r_A} to initiate the connection. These two computations, however, are not extra computation cost for the user because those two are to be executed by the user for the original protocol NFS as well. The only difference here is the user should necessarily compute g^{r_A} in advance of sending the first message.

The added field now enables the network to check if the received initiation message is just garbage values or not. The network retrieves g^{r_A} from $(g^b)^{r_A}$ using its long-term private key b and then build its own value of $h(g^{r_A}, B)$ to check the match. The computation cost up to this point is one exponentiation for the network, which is the same as the original protocol.

Now, we consider the case in which an attacker tries the DoS attack 1.

Protocol 55. *DoS attack 1 for the protocol NFS with the enforcement of heavy computation*

A: attacker, B: network

1. A $\rightarrow$ B: x, y (x, y : garbage values)

B: checks if $h(x^{b^{-1}}, B) = y$

By the above checking, the network B detects that x and y are just garbage values, and stops executing the session (the second message does not happen and there is no open connection waiting any further message from the attacker). Here, the computational cost paid by the attacker is zero, whereas it is *one* exponentiation in the network to check x and y. It should be noticed, however, that there is no open session corresponding to an instance of DoS attack 1 after checking the first message. This desirable feature is not necessarily guaranteed by all protocols. For instance, the Station-to-Station (STS)

protocol (and similar protocols) does not provide that feature. The STS protocol is initiated by sending the system-common generator g raised to a random number r_A chosen by an initiator, and hence there is no vehicle on which checking mechanism can be mounted. The network simply computes its own values, sends the second message back to the attacker, and then waits the third message which is given up by the attacker. Therefore, as a result of consecutive, say 100, bogus initiations, the network ends up with 100 sessions left open in addition to the wasted computation.

We assumed above that an attacker sends simply garbage values, but it is in fact too naïve. What if the attacker computes correct values for the first message and then repeats them many times or copies any other protocol initialisation message? To overcome this problem and satisfy the requirements of no additional number of exponentiations, we modify the protocol NFS further to incorporate the cryptographic salt as described in Section 4.2.1.1 as follows.

Protocol 56. *Instance protocol NFS modified with the enforcement of heavy computation*

A: *user*, B: *network*

B: broadcasts a random number r in addition to its public key g^b

1. A $\rightarrow$ B: $(g^b)^{r_A+r}$, $h(g^{r_A}, B)$
2. A $\leftarrow$ B: r_B, $h(g^{r_A}, K_{AB})$
3. A $\rightarrow$ B: $\{h(B, r_B, g^{r_A})\}_{K_A^{-1}}$, $\{A\}_{K_{AB}}$

A, B: $K_{AB} = h(g^{r_A}, r_B)$

A new random number r is introduced to cope with the identified problem in the previous protocol to address the DoS attack. Note that it costs no additional exponentiation to either A or B. The checking procedure in the network B is as follows.

Protocol 57. *Checking procedure in the network for the enforcement of heavy computation*

A: *user*, B: *network*

A: computes $x = (g^b)^{r_A + r}$ and $y = h(g^{r_A}, B)$

1. A $\rightarrow$ B : x, y

B: checks if $h(x^{b^{-1}} g^{-r}, B) = y$

This new fix, however, comes at some operational cost to be paid by the network. The random data r need not to be unique to each user but distributed over the system broadcast channel, at least in the case of the user-network interface. (As for user-VASP interface, the VASP can use the same value of r for all the users over some time-frame.) The network has to update the values of r and g^{-r} periodically and the appropriate update frequency must be chosen to frustrate the DoS attack. A variant of this mechanism is to broadcast the random number r only when there is any symptom of DoS attack detected, which is along the same line of the mechanism proposed recently by Juels [JuBr99].

The most significant benefit of the above variant protocol using an additional random value r is that many potential DoS attacks in which wasted connections are left open, as well as wasting computations within the network, are turned into the lighter DoS attack 1.

4.2.2 Weak Key Confirmation

Key confirmation requires a principal to derive the final session key itself to present the other principal the required proof of the possession of the key. However, the corresponding computation would be simply a loss if the session was initiated by an attacker making the DoS attack 2 in which the attacker sends the correct initiation message but does not send the further message expected in the security protocol. Therefore, the computation of the session key is better put off until the final message from the user is returned to the network.

The following two variants are modified from the original instance protocols FS 1 and 2 to satisfy the weak key confirmation as well as the enforcement of heavy computation.

Protocol 58. *Instance protocol FS1 modified for heavy computation and weak key confirmation*

A: *user*, B: *network*

B: broadcasts a random number r in addition to its public key g^b

1. A $\rightarrow$ B: $(g^b)^{r_A+r}$, $h(g^{r_A}, B)$
2. A $\leftarrow$ B: g^{r_B}, $h(g^{r_A}, g^{r_B})$
3. A $\rightarrow$ B: $\{h(B, g^{r_B}, g^{r_A})\}_{K_A^{-1}}$, $\{A\}_{K_{AB}}$

A, B: $K_{AB} = g^{r_A r_B}$

Protocol 59. *Instance protocol FS2 modified for heavy computation and weak key confirmation*

A: *user*, B: *network*

B: broadcasts a random number r in addition to its public key g^b

1. A $\rightarrow$ B: $(g^b)^{r_A+r}$, $h(g^{r_A}, B)$
2. A $\leftarrow$ B: $g^{r_A r_B}$, $h(g^{r_A}, g^{r_A r_B})$
3. A $\rightarrow$ B: $\{h(B, g^{r_B}, g^{r_A})\}_{K_A^{-1}}$, $\{A\}_{K_{AB}}$

A, B: $K_{AB} = h(g^{r_A}, g^{r_B})$

The original field for key confirmation, K_{AB} was replaced by g^{r_B} and $g^{r_A r_B}$ in the variants FS1 and FS2, respectively. Hence, the network can delay one exponentiation for each of $g^{r_A r_B}$ and g^{r_B} in the modified FS1 and FS2, respectively. Weak key confirmation, therefore, reduces the computation cost for the network which will be wasted in the case of DoS attack 2 from *three* to *two* exponentiations in each of FS1 and FS2.

The bad effect of the DoS attack 2 is that the connection initiated by an attacker is left open waiting the third message in the security protocol.

4.2.3 Comparison of the ASPeCT and the WAKE Protocols Regarding DoS Attacks

This section briefly summarizes the computational cost of the ASPeCT protocol and the proposed WAKE protocols equipped with the two techniques: *enforcement of heavy computation* and *weak key confirmation*. Note that the mechanism for enforcement of heavy computation and the corresponding checking scheme described above is not

applicable to the ASPeCT protocol because of its similarity to the STS protocol. In fact it does not allow a computationally cost-free solution to the DoS attack 1. Instead, like the proposed WAKE protocols FS1/2, the concept of weak key confirmation can be easily applied to the modified ASPeCT protocols for forward secrecy which are described in Section 4.1.3.

Table 4-3: The result of the DoS attacks in terms of the number of exponentiations and the resultant state of the session

Protocol	***DoS Attack 1***		***DoS Attack 2***	
	Attacker (***A***)	***Network*** (***B***)	***Attacker*** (***A***)	***Network*** (***B***)
ASPeCT *original* *	0	1 *+ session left open*	1	1 *+ session left open*
ASPeCT *modified FS1/2 for forward secrecy* *	0	2 *+ session left open*	1	2 *+ session left open*
WAKE protocol *NFS*	0	1	2	1 *+ session left open*
WAKE protocol *FS1/2*	0	1	2	2 *+ session left open*

* In fact, the network using the ASPeCT protocol cannot detect a garbage value from an attacker and hence the damage to the network is exactly the same for both types of attacks.

Table 4-3 shows the computational cost in terms of the required number of exponentiations by both the attacker and the network, and the final state of the corresponding session. From this table, we can see that the WAKE protocols are more robust than the ASPeCT protocols, especially in the case of forward secrecy variants *FS1* and *FS2* of both protocols. The usual pattern of connection depletion attacks is of the DoS attack type 1 where the attacker's first message is garbage. Therefore, the most interesting part of the table is the third column shaded in grey. After an instance of the DoS attack, the ASPeCT system server results in having one session left open whereas no session is wasted in the WAKE case.

4.2.4 Complexity of the WAKE Protocol with DoS Attack Prevention

When the WAKE protocol is equipped with the countermeasure against the DoS attacks (see Protocol 56), the required number of exponentiations in the network is one for *one*

or more WAKE protocol sessions. This is because a particular value of cryptographic salt *r* in Protocol 56 may be used uniquely for each connection request or globally for every connection requests for a period of time. The required exponentiations can be executed off-line as a batch job in the network. Therefore, the exact complexity of the countermeasure is difficult to calculate but we can see that it is much less than one exponentiation for the network side, approaching to zero. On the other hand, the user side does not have to spend any additional exponentiations at all.

4.3 Clone Detection Mechanism

However sophisticated the future wireless security mechanisms are, they cannot be panaceas for all the potential frauds, amongst which especially the mobile-terminal cloning fraud is out of the scope covered by user authentication mechanism. According to an estimate by the Cellular Telecommunications Industry Association, nearly 90 % of cellular fraud in North America in 1995 was due to the cloning of cellular handsets and more than 75,000 subscribers' handsets were cloned each month in the year [Hill96].

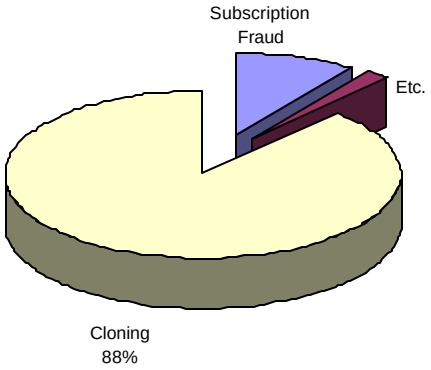


Figure 4-5 : Types of cellular fraud

Although made more difficult, copying or cloning of user/terminal data cannot be eradicated. Attackers will have to devise more sophisticated methods than just radio-scanning the mobile identity and the electronic serial number of a legitimate user's terminal because the secret information for authentication will never be transmitted over the air. One of the most common cloning methods is terminal-to-terminal copying which is believed to be one of the major sources of cloning fraud. Another method is getting secret authentication data from employees of wireless operators or certificate authorities. Authentication data could also be obtained from attacking the route for distributing secret authentication keys to subscribers, especially in the case of mail delivery and over-the-air activation. Finally, although rare, a cryptographic attack against authentication protocols and their corresponding algorithms may result in cloning being possible.

Almost all of the solutions to cope with this kind of fraud require the use of the mechanisms which are separate from the original built-in cryptographic system. It

would surely be a burden in terms of investment cost and operation/maintenance. With this in mind, an alternative solution is proposed in this section purely based on a cryptographic technique, which is therefore smoothly integrated into any built-in cryptographic mechanisms of mobile communication systems.

In the following, we review several existing methods for clone detection, describe a current generation mobile system and a newly devised clone detection mechanism for it, and finally adapt the method to the WAKE protocol.

4.3.1 Alternative solutions for fraud detection

To cope with cloning fraud, many kinds of detection technologies have been proposed. These include the use of PIN, profiling, RF fingerprinting, and call history counters. Most of these mechanisms were at first considered as add-on security solutions for the first generation analog cellular systems which have no inherent authentication protocols.

4.3.1.1 Analysis of Usage Pattern

The most prominent among existing fraud detection methods is based on the analysis of the usage patterns of mobile users. This type of detection mechanism can be implemented through "rule-based approach" or "neural network approach" [ASPe97]. User profile analysis and Call Detail Records (CDRs) analysis are other names for this type of detection mechanisms. At first, it was conceived as an add-on solution for the first generation cellular systems without inherent authentication functions and protocols. However, even the third generation wireless systems such as UMTS (Universal Mobile Telecommunications System) are expected to use this kind of fraud detection methodologies [ASPe97] because of the reasons discussed above.

This approach for clone detection uses a profile analyzer (profiler) which gathers and analyses all the relevant usage data such as calling time, geographic position of mobiles, call duration, and call frequency. If the profiler finds an extraordinary discrepancy between the current call pattern and well-established usage pattern of the user, it reports that the call is "likely" to have been made by a cloned terminal.

However, there are some significant shortfalls of this mechanism. This mechanism entails having to install a profiler system separate from an authentication system,

provide proper message transmission between the profiler and the other network elements, e.g. a mobile switch, and manage a considerable amount of data. Furthermore, the analysis of usage patterns of users - e.g. tracking of user's location, is an infringement on the privacy of users. Most of all, what this detection system gives out is not a fact but a probability of the occurrence of cloning fraud.

4.3.1.2 RF Fingerprinting

More recently, a new method called "RF fingerprinting" [Fred95] has been developed and tried in the AMPS services. This can be thought of as a "mobile" version of profile analysis method. This RF fingerprint is the unique signal pattern emitted by a mobile terminal. This scheme needs to accumulate signal patterns of new users over initialization periods, and then compare the stored pattern with the incoming pattern of a mobile terminal on subsequent calls. As such, it requires an extra control system which maintains and updates the central database for the fingerprint of each mobile terminal. Furthermore, it entails quite a large investment in equipment for every base stations.

4.3.1.3 PIN

The oldest method of fraud control is using a personal identification number (PIN). Each user is given a secret PIN hopefully known only to the user and the authentication center. It requires users to enter their PIN when making calls. However, this is often an annoyance and can be easily defeated because the PIN itself is transmitted over the air.

4.3.1.4 Call History Count (COUNT)

North American second generation digital cellular systems based on IS-41C use a call history parameter, COUNT which has been introduced to detect cloning [TIA95]. For every instance of mobile authentication, the network checks whether the received values of authentication response (AUTHR) and COUNT are equal to its own values of AUTHR' and COUNT' respectively (see Figure 4-6). If the values of both sides match, the mobile is given the access to the network. In the case of AUTHR match but COUNT mismatch, the subsequent action to handle the mismatch is not defined in the standard, and hence is up to the operators.

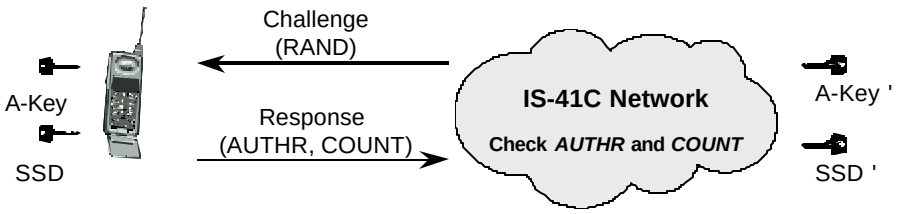


Figure 4-6: Authentication mechanism of IS41-C

The value of COUNT is a modulo 64 binary number and is incremented by one under the control of the network, at intervals defined by the network operator. Once the value of COUNT is updated for a terminal that has been cloned, the value of COUNT within the Authentication Center (AC) will match only one of either the original terminal or the cloned terminal.

In the case that the Shared Secret Data (SSD) value of a particular mobile has been cloned, as soon as a COUNT mismatch has been detected, the network can update the SSD of the terminal (refer to Figure 4-7 and Figure 4-9). Therefore the cloned terminal will operate no longer. However, if the cloned terminal includes a true copy of the authentication key (A-Key), the cloned terminal is able to respond correctly to the SSD update. Hence, using the call history count in the way described above for fraud detection and management is insufficient. Furthermore, the possibility of COUNT mismatch between a legitimate terminal and the AC cannot be ruled out, which may be a result of failure within the terminal or network or over the radio interface.

Our recommendation for better usage of COUNT is to simply increment it automatically whenever the user accesses the network without the need for additional message exchanges between the terminal and the network solely for COUNT update. All the instances of the received values of the parameter are reported by the NO to the SP, and analysed to check whether the expected number and actual number of the instances of the counting parameter reported match each other. This scheme is simple but very effective.

After all, the final step in fraud management will need some enquiry to the mobile customer whose terminal is believed to have been used by fraudsters to make another illegal copy. Such a process might annoy the customer, especially if the suspicion has

been found to be mistaken. Therefore, IS-41C network operators will need a more accurate clone detection method which enables them to be confident in their decision that cloning has been done for a particular mobile.

4.3.2 SSD: IS-41C's hidden fraud detection method

The Shared Secret Data (SSD) parameter mentioned above was originally introduced to share the load of authentication processing and signalling between the home Authentication Center (AC) and the visited system's Visiting Location Register (VLR). With SSD shared, the VLR can calculate the authentication response AUTHR and compare it with that received from the corresponding mobile terminal. The home AC does not need to authenticate the mobile. Of course, this secret data may or may not be shared, and it is up to the operators.

The SSD in this chapter, however, is used as another kind of security measure. We propose to use the SSD for clone detection as well.

4.3.2.1 Security Data in IS-41C

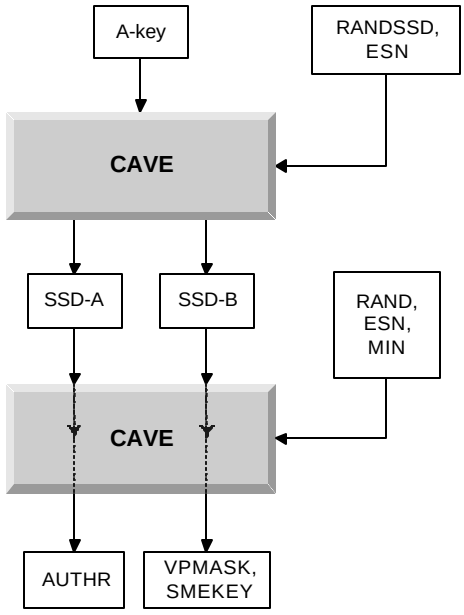


Figure 4-7 : Security related data and their relations in IS-41C

The security mechanism of IS-41C has more types of security related data than its European counterpart, GSM. Above all, it has one more key level which is composed of SSD-A and SSD-B. The former is for authentication purposes and the latter for ciphering services (Figure 4-7). This will add up to stronger security for the master key, A-Key because it is not the A-Key itself but the SSD that is used to derive the authentication response, AUTHR. SSD can be viewed as a temporary copy of the A-Key. The AC can decide the SSD update period and change the SSD value of a particular mobile anytime when it wants to do.

The CAVE algorithm as shown in Figure 4-7 is a hash function defined in [TR45]. It is used for generation of SSD, AUTHR, Voice Privacy Mask (VPMASK) and Signaling and Message Encryption Key (SMEKEY) from the input data comprising secret keys (A-Key, SSD), fresh random values (RANDSSD, RANDOM) and mobile station identity informations (MIN, ESN).

4.3.2.2 Authentication Response Handling and Local Administration Procedures

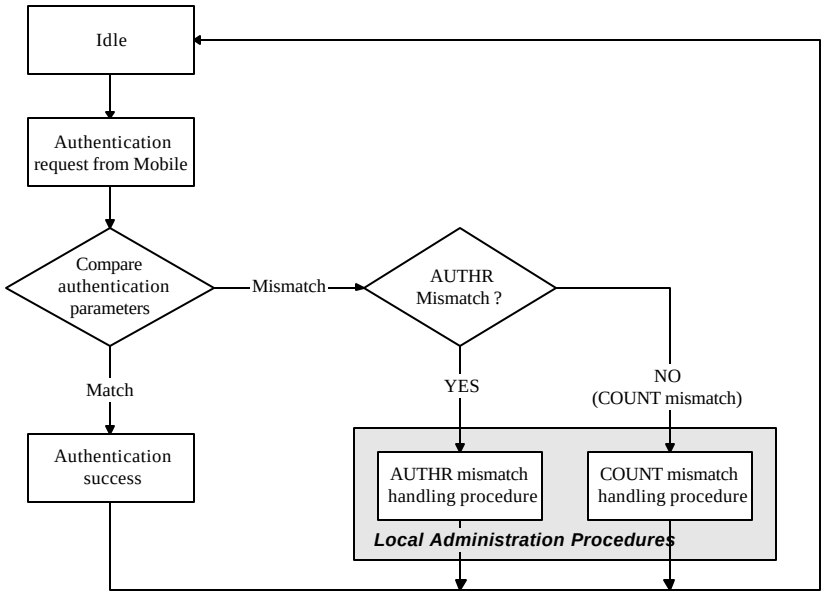


Figure 4-8 : Local administration procedures for handling "authentication failure"

Authentication success for a particular mobile station means that both of the values of AUTHR and COUNT sent from it equal the corresponding values stored within the Authentication Center. On the other hand, a mismatch in one or both of the authentication parameters is regarded as authentication failure. The standard does not specify how to handle the authentication failures. The required handling capability or logic is treated as a black box and just called "local administration procedures" within the standard documentation [TIA95].

IS-41C provides too many optional security measures compared to the simple features of European GSM system. These are call history count parameter (COUNT), temporary secret key (SSD) and two kinds of challenge-response protocols: global challenge and unique challenge. The local administration procedures may use some or all of the above security measures appropriately. In this chapter, we only focus on SSD, COUNT and their update protocols in the viewpoint of their use for cloning detection.

The COUNT parameter is described in a previous section, and we describe the SSD update procedure in the following section.

4.3.2.3 SSD Update Procedure

SSD Update Procedure is the most powerful response to any kind of security problem detected by the network. It provides a feature of restart or reset of security data to both the mobile station and the AC. It can be used to handle any kind of security violations such as AUTHR mismatch and COUNT mismatch. The SSD Update procedure, however, should be used sparingly because it is the most resource consuming method amongst several procedures provided by IS-41C security protocol.

Figure 4-9 provides a simplified view of this procedure. When the network determines to update the SSD value of a particular mobile station, it generates a random number called RANDSSD, enters it together with A-Key and ESN into the CAVE algorithm, achieving a new value of SSD. The network also sends the mobile station the value of RANDSSD using SSD Update Order message. The mobile station runs the CAVE algorithm using the received value of RANDSSD and thus achieves the same value of SSD as that of the network. The mobile station then authenticates the network; if the result is successful the mobile station updates the value of SSD, otherwise it does not change the SSD value. After SSD update, the mobile station sends the network SSD

Update Confirmation message. The network now authenticates the mobile station to make sure that the mobile station has the same new value of SSD as its own. After that the network may optionally execute the COUNT Update procedure to update the value of COUNT parameter shared between the mobile station and the network. Finally the network updates the value of SSD.

In the following section, we show that this SSD Update procedure is a built-in security bullet for cloning detection in IS-41C networks.

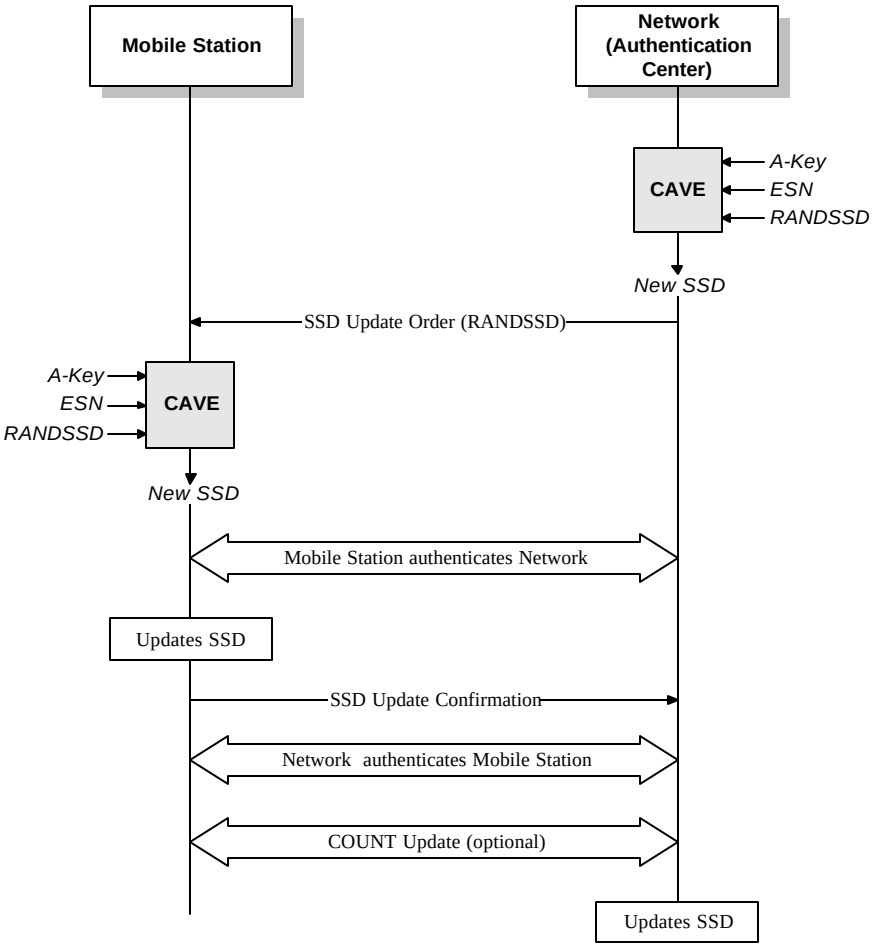


Figure 4-9: SSD Update procedure in IS-41C

4.3.3 Cloning Detection Using SSD Update Procedure

The basic idea of the detection mechanism to be introduced in this section is to use a temporary secret key like SSD in IS-41C systems which is shared between the

legitimate user and his home service provider. The temporary key is derived from a permanent secret key like A-Key in IS-41C systems assigned to the user, and updated periodically or when decided to be updated. This temporary key provides the service provider a sort of key-agreement state which may fall down to a disagreement state between the user and the service provider when the user's terminal has been cloned. This is because the temporary keys within the legitimate terminal, cloned terminal and the service provider's management system (e.g., authentication centre) may not match even though the corresponding permanent keys are all the same.

Assume that an attacker has succeeded in making an exact duplicate of a legitimate user's mobile station (MS). The cloned phone will have the same secret authentication key (A-Key) as the original phone. It is impossible for authentication center(AC) to distinguish between the authentic mobile station and its illegal copy. In fact, the detection of cloning in this chapter means that the AC has detected that cloning had happened for a particular MS. If we denote MS and MS' as two mobile stations having exactly the same authentication data, the authentication data status of MS, MS' and AC will be as shown in Table 4-4.

Table 4-4 : Initial state of authentication data in MS, MS' and AC

MS		AC		MS'
$A\text{-}Key_{MS}$	=	$A\text{-}Key_{AC}$	=	$A\text{-}Key_{MS'}$
SSD_{MS}	=	SSD_{AC}	=	$SSD_{MS'}$
$COUNT_{MS}$	=	$COUNT_{AC}$	=	$COUNT_{MS'}$

If one of the twin mobiles, e.g., MS, accesses to network and AC and MS execute the COUNT update procedure, the security data status will change as shown in Table 4-5.

Table 4-5 : The agreement/disagreement state of authentication data in three parties, after COUNT update between MS and AC

MS		AC		MS'
$A\text{-}Key_{MS}$	=	$A\text{-}Key_{AC}$	=	$A\text{-}Key_{MS'}$
SSD_{MS}	=	SSD_{AC}	=	$SSD_{MS'}$
$COUNT_{MS}$	=	$COUNT_{AC}$	?	$COUNT_{MS'}$

After that, if MS' tries to access the network, it will necessarily cause the COUNT mismatch event to happen within the AC system and MS' may be denied access. This access failure does not seem to dissuade the user of MS' (authentic or fraudulent) from second, third or more tries to make a call. Hence, if system operator has set the AC to issue SSD Update procedure for a preset threshold number of COUNT mismatches, MS' with the true A-Key will derive the same value of new SSD with that of AC, and succeed in the procedure. In addition to the SSD Update procedure, AC of IS-41C network should initiate some correction procedure for re-agreement of COUNT values between AC and MS'. Unfortunately, however large discrepancy of COUNT values between two entities is, there is no one-step method to get the agreement. AC should send COUNT update order (each for one increment) to MS repeatedly until they are in the agreement of COUNT values. Instead, the operators may prefer to change the COUNT to the same value of that stored in MS'. After that, MS' will be in the exact agreement state with AC and MS in a disagreement state (see the following table) in turn, which is illustrated in Table 4-6.

Table 4-6 : The agreement/disagreement state of authentication data in three parties, after SSD update between MS' and AC

MS		AC		MS'
$A\text{-}Key_{MS}$	=	$A\text{-}Key_{AC}$	=	$A\text{-}Key_{MS'}$
SSD_{MS}	?	SSD_{AC}	=	$SSD_{MS'}$
$COUNT_{MS}$	?	$COUNT_{AC}$	=	$COUNT_{MS'}$

Now, whenever the MS tries to access the network, it will suffer the AUTHR mismatch as well as COUNT mismatch because the MS' and AC has updated their values of SSD which is different from the old value of SSD which is still being used by the MS. This disagreement of SSD between the MS and the AC will cause subsequent AUTHR and COUNT mismatch events in the AC system quite often. This phenomenon will trigger SSD update procedure again, and the MS will succeed in the update procedure and authentication check. The agreement/disagreement state of the three entities is demonstrated in Table 4-7.

Table 4-7 : The agreement/disagreement state of authentication data in three parties, after SSD update between MS and AC

MS		AC		MS'
$A\text{-}Key_{MS}$	=	$A\text{-}Key_{AC}$	=	$A\text{-}Key_{MS'}$
SSD_{MS}	=	SSD_{AC}	?	$SSD_{MS'}$
$COUNT_{MS}$	=	$COUNT_{AC}$	?	$COUNT_{MS'}$

This scenario will necessarily repeat until the operator takes any kind of recovery action. Now we can find a pattern of the sequential events.

- *Repeated COUNT mismatches for a particular mobile station*
- *SSD update success for the mobile station*
- *Repeated AUTHR mismatches for the mobile station*
- *SSD update success for the mobile station*
- *Repeated AUTHR mismatches for the mobile station*

Here we can see that cloning will inevitably cause repeated events of AUTHR mismatch and the SSD update success to happen as illustrated in Figure 4-10.

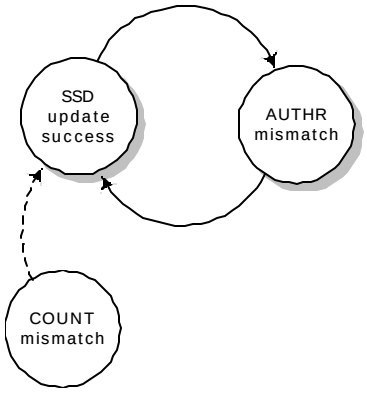


Figure 4-10: Symptom of cloning

Of course, this sequence of events might be interleaved by another irregular event such as repeated authentication successes. However, the repetition of the above pattern within a time period will be a clear indicator of the existence of a cloned phone. It should be noted that this pattern may happen even if the IS-41C network did not employ the COUNT parameter in the authentication protocol. This is because the SSD update

procedure is recommended to be executed periodically for security, which will bring a disagreement of SSD values between the AC and MS/MS'.

The observation of the stereotyped pattern may be implemented into the AC local administration procedure in many different ways. Using this, IS-41C network operators will be able to detect cloning with high confidence using the AC and its built-in local administration procedures without deploying any special add-on system at extra cost. Such an AC system is depicted in the following figure.

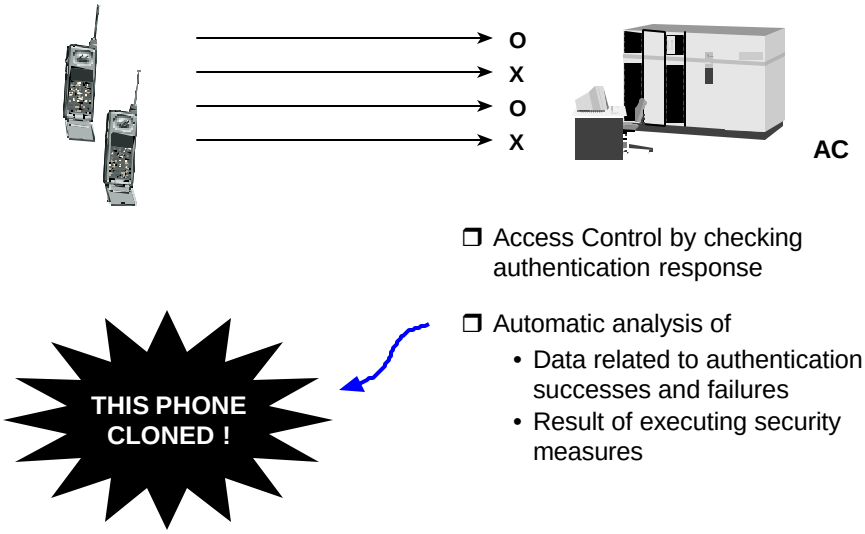


Figure 4-11 : Cloning detection within AC system without any add-on special purpose system

Fraudulent usage, especially by cloning fraud, in wireless mobile communication service has forced the operators to seek for effective clone detection methods. Most of the proposed schemes require the operators to invest considerable amount of money and modify their network infrastructure to deploy the schemes.

What if there is already built-in a more effective clone detection method in the authentication protocol itself? This section shows that the IS-41C system has a hidden mechanism for clone detection which enables the operators to be confident in their decision about cloning fraud. This method just makes use of the SSD, a temporary secret key in IS-41C which was originally introduced for load sharing between the home network and the visited network.

4.3.4 Application to third generation wireless systems

The concept of clone detection usage of a temporary secret key can be applied to future wireless systems. If the future systems' security infrastructure is compatible with that of IS-41C systems, the application is just straightforward. On the other hand, for systems with more sophisticated security infrastructure based on asymmetric techniques, the clone detection method can also be applied with some modifications as will be described in this section.

The network assigns a permanent secret key, K_p (analogous to A-Key in IS-41C) to a new user, from which a temporary secret key K_t (analogous to SSD in IS-41C) is to be calculated by computing $K_t = h(K_p, RAND_t)$ where h is a common agreed hash function between the user and the network, and $RAND_t$ is a random number chosen by the network. K_t is used as an input together with COUNT to the hash function to calculate the authentication response AUTHR. Figure 4-12 shows the computation procedure in both the mobile station and the network for the temporary key and the corresponding response value.

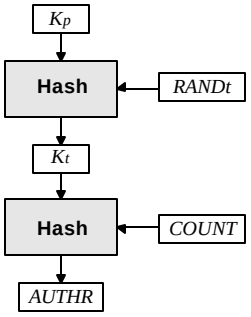


Figure 4-12: The procedure for the calculation of the temporary key and authentication response

RANDt, a random challenge for K_t update, is to be chosen only by the network like RANDSSD in IS-41C protocol. The K_t update procedure, which is similar to the SSD update procedure in IS-41C, will cause a discrepancy of AUTHR values between the network and the legitimate terminal or the illegal terminal once both terminals have been used subsequent to the cloning.

The COUNT parameter is also used in this scheme, not as a call history count, but as a "usage count" for the temporary secret key, K_t. The network does not need to explicitly check if COUNT values of both the mobile station and the network match because the mismatch will lead to the AUTHR mismatch (see Figure 4-12). In addition, the COUNT should be updated automatically without any additional message exchange between the mobile station and the network, and always reset to zero value after successful K_t update. This relieves the network of signalling load which was required for the COUNT Update procedure in IS-41C network. In the case of IS-41C, the COUNT parameter is a call history of the mobile station and can be updated only by the relevant update order from the AC. IS-41C's COUNT has no way to be reset and can just be incremented by one. In the event of COUNT mismatch, the overhead to restore COUNT value may be formidable.

Figure 4-13 shows the application scenario. In the diagram, it is assumed that there is more than one mobile station with the same identity and authentication data, except that the temporary secret key K_t may be different. The clone detection parameter, AUTHR is sent from the mobile station to the network during the execution of the public-key based authentication procedure. It may be transmitted in clear or encrypted The network updates the event counter for the mobile station whenever it passes the public-key based authentication but fails in the secret-key authentication, and then updates the temporary key successfully. If the event count is greater than or equal to the threshold value (the value of 2 will be enough), it indicates that cloning of the phone has occurred. It should be noted that this cloning detection procedure does not need to be executed entirely on-line in real-time. Public key based authentication procedure will be executed between the mobile station and the visited network, and the secret key based cloning detection check and calculations may be executed off-line within the mobile station's home network.

We can easily equip our WAKE protocol with the clone detection facilities of Figure 4-12. The only modification needed is to provide placeholders for the two parameters *AUTHR* and *COUNT*, which requires the third message to include *AUTHR* and *COUNT* as shown in Protocol 60. When a mobile station is suspicious with regard to cloning, the network can initiate the K_t update procedure (Protocol 61). All the

description with regard to cloning detection in Section 4.3.3 is applicable to this case with A-Key and SSD replaced with K_p and K_t, respectively. The corresponding symptom of cloning will be as shown in Figure 4-14, which is exactly the same pattern as illustrated already in Figure 4-10 for IS-41C system. This pattern can be exploited into security administration procedure in the network to detect cloning.

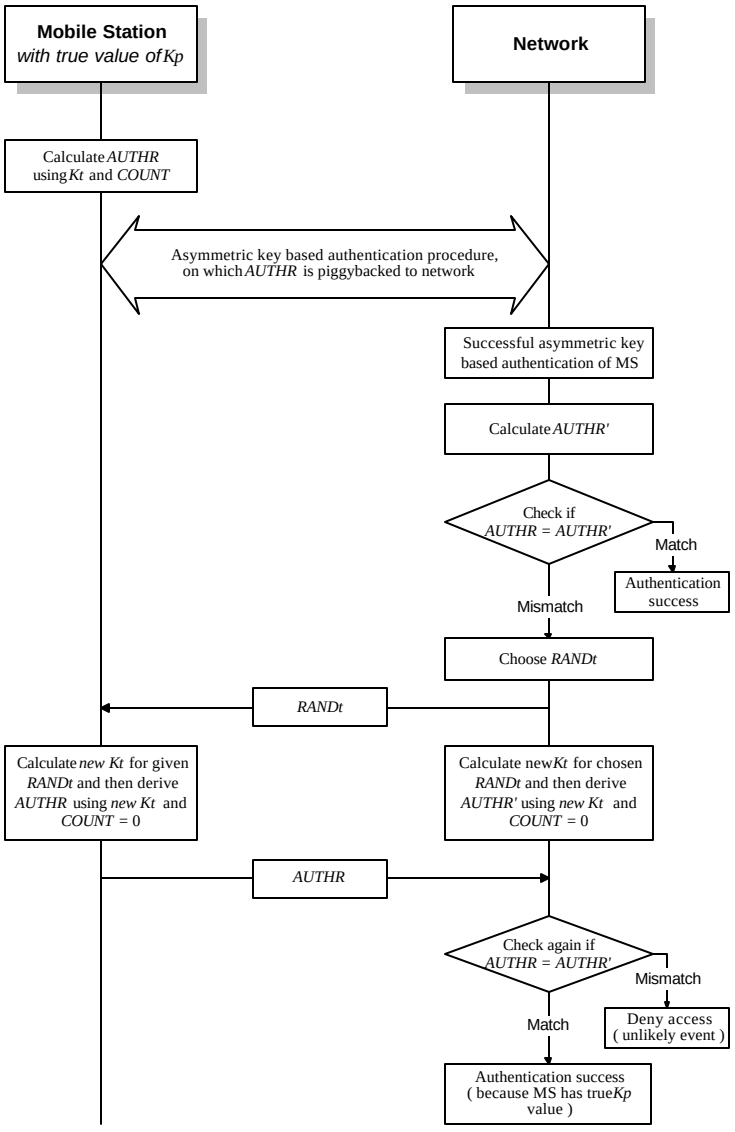


Figure 4-13 : Application scenario of the secret temporary key

Protocol 60. *Clone-detection enabled WAKE protocols*

A: user, *B*: network

1. A → B: $BPubKey\{r_A\}_R$
2. A ← B: r_B, $SessKey_{AB}\{r_A\}_{NR}$
3. A → B: $APriKey\{B, r_B, r_A\}_{NR}$, $SessKey_{AB}\{A\}_R$, $AUTHR$, $COUNT$
   where $AUTHR = h(K_t, COUNT)$

Protocol 61. K_t *update procedure*

A: user, *B*: network

1. A ← B: $RAND_t$ (K_t_**update_order**)

A: 1) updates $K_t = h(K_p, RAND_t)$
   2) reset $COUNT$ to zero
   3) computes $AUTHR = h(K_t, COUNT)$

2. A → B: $AUTHR$, $COUNT$

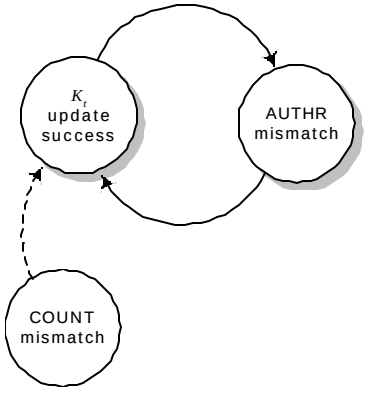


Figure 4-14: Symptom of cloning

4.3.5 Complexity of the WAKE Protocol with Clone Detection Support

The additional computation complexity for clone detection support is zero. The WAKE protocol simply provides placeholders for the relevant parameters in the third protocol message as shown in Protocol 60. The required analysis for clone detection is not a part of the WAKE protocol session, but a part of security administration processing in the network.

4.4 Electronic Payment Mechanism

Over recent years, there has been a significant increase in both the scale and the diversity of electronic transactions over the Internet. Electronic commerce (E-commerce) means electronic payment (E-payment) in a narrow sense but it may mean *electronic business* in a broader sense which also includes the exchange of information not directly related to the actual purchasing activities [GrFe99, p.2]. In this thesis, the term *E-payment* will be used to describe purchasing activity itself between buyers and sellers. Secure electronic payments will not only make purchasing activities more flexible and convenient but also create not yet imagined new markets. The latter aspect of E-payment is described in detail by Wayner [Wayn97].

E-commerce takes place over the telecommunication media, and hence a widespread mobile communication service will be a good vehicle for it. In this section, we present a general introduction to E-payment and micropayment as a light-weight version of the E-payment mechanism, and finally integration of micropayment into WAKE protocol.

4.4.1 E-Payment Environments

Figure 4-15 shows a general scenario in E-payment environments, which was adopted from [Ahuj96].

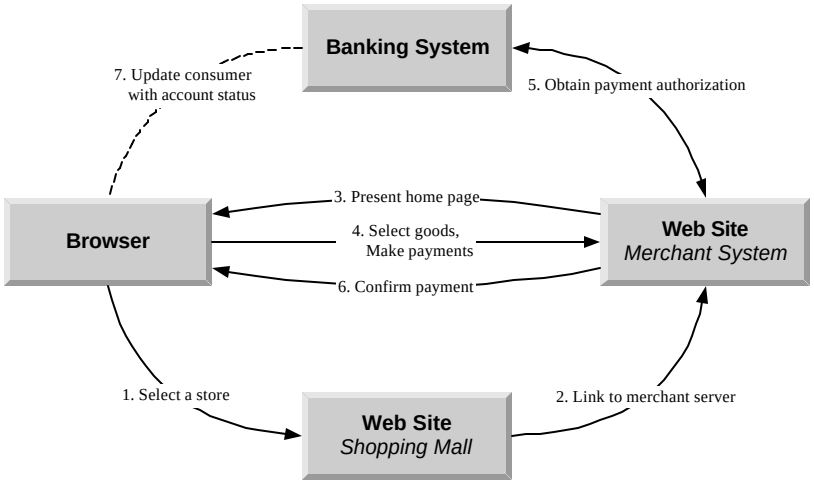


Figure 4-15: E-payment Environments

In this scenario, there are four elements or role players as follows.

- **A consumer** along with a Web browser uses the hyperlinks from the mall to access the merchant's home page.
- **A merchant** system residing on an online Web server with a connection to Web browsers over the Internet consists of the home page and related software to manage business.
- **An online shopping mall** may help direct consumers to the merchant server. It may pay to enlist with one or more well-known shopping malls.
- **A background banking network** supports electronic payments from consumers to the merchant. This network may include two types of banks:
  - ***a merchant's bank*** *maintains the account for the merchant, authorizes and processes the payments. It may use on-line real-time link to the merchant so as to allow online authorization of consumer payments, and the link with the consumer's bank for verifying the transactions.*
  - ***a consumer's bank*** *manages the account for the consumer, and has an offline link to the consumer, such as via postal mail or e-mail.*

These four role players take part in the following sequence of E-payment related activities.

1. The consumer accesses the shopping mall and selects a shop for purchasing certain items.
2. The shopping mall server accesses the merchant system for the selected shop.
3. The merchant system presents the store's home page to the consumer. It also includes information on the various goods available from this store.
4. The consumer selects the desired goods, interacts with the merchant's system, and makes the payments.
5. The merchant system accesses its bank for authorization of the consumer payment.
6. The merchant system informs the consumer that the payment is accepted and the transaction is completed. (At a later time, the merchant's bank obtains payment from the consumer's bank.)
7. The consumer's bank informs the consumer of the money transfer through mail such as a monthly report or online bank account.

4.4.2 E-Payment Mechanisms

Provision of secure and trustworthy E-payment mechanisms will be a critical factor for the success of E-commerce. Such a payment scheme must satisfy the following requirements [Ahuj96].

- Strong authentication of each party using certificate and digital signature
- Privacy of transaction using encryption
- Transaction integrity using message digest algorithms
- Nonrepudiation to handle disputes about the transaction

It should be noticed that the ASPeCT micropayment mechanism is able to satisfy all these security requirements, and thereby is applicable to macropayment environments as well.

There are many classification methods for E-payment schemes, many of them rather orthogonal to each other. The following list shows an example of many classification criteria, most of which are described in detail in [ASPe97b] .

- Electronic purse/cash/credit
- On-line/off-line
- Credit-based/debit-based
- Software-based/tamper-resistant hardware
- Macropayment/micropayment

In this report, we focus on the micropayment scheme because this category not only directly addresses the limited resources of mobile communications but also is the most reasonable option for applying to the light-weight E-payment by mobile users.

4.4.3 Micropayments

Many E-payment schemes have been proposed, and a lot of them assume the use of today's well-established credit card business environment. The best known E-payment protocol is the SET (Secure Electronic Transaction) protocol, which was produced by Visa and MasterCard to be their standard for processing credit card transactions that travel over networks like the Internet. Many other players, including a telecommunication operating company, are also involved in the protocol. This and other

similar schemes use extensive cryptographic technologies, a lot of which are based on public-key cryptography to satisfy high level security requirements such as nonrepudiation. These protocols are all appropriate for medium to large transactions (macropayments) of more than $5 or $10. These macropayment protocols will be too expensive and time-consuming solutions when applied to inexpensive transactions, 50 or 25 cents and less, because of the transaction charges of card companies and the computational cost of public-key signature/verification. Without appropriate cheap alternative schemes, the light-weight transaction market could not be developed in its full potential. This market typically includes selling *inexpensive* information software, and services (e.g. directory search or games), usually delivered online.

Several schemes have been proposed for micropayment. To reduce the computational and signalling burden down to a reasonable level which can be justified in micropayment environments, they try to avoid *public-key* cryptography partially or entirely. Their dependency on *on-line* access to banking/clearing systems is also small compared to macropayment schemes.

4.4.3.1 Hash Chain

The most widely studied and promising approach involves using a public-key signature together with a hash-chain. Four similar schemes have been proposed: PayWord [RiSh96], *i*KP's micropayment [HSW96], Netcard [AMS95], and Pedersen's scheme [Pede95]. The basic idea in these schemes is that a signature value generated using a public key operation is spread over many other cryptographic values derived by much more efficient one-way functions such as hash functions. In other words, the effect of a digital signature is reused many times over subsequent messages (containing preimages of a specific hash). This mechanism was, in fact, proposed for use in authentication scheme earlier by Lamport [Lamp81]. The following description of the hash chain scheme is based on *PayWord* proposal [RiSh96].

Issuing user certificate .

- The *user* U establishes an account with a *broker* B.
  - U → B: *credit card no., UPubKey, DeliveryAddr*
    user's aggregate charges will be charged to her credit-card number.

- The broker issues to U a PayWord Certificate containing:
  - *broker's name B*
  - *user* name *U*
  - *IP-address*
  - *user's public key UPubKey*
  - *expiration date ExpDate*
  - *other information OtherInfo*: *possibly user-specific information such as*:
    - a certificate serial number,
    - credit limits to be applied per vendor,
    - information on how to contact the broker,
    - broker/vendor terms and condition, etc.

  Summarizing, the user's certificate UCert has the form:

  - UCert = $BPriKey\{ B, U, DeliveryAddr, UPubKey, ExpDate, OtherInfo\}_R$
- The user's certificate has to be renewed by the broker (e.g. monthly), who will do so if the user's account is in good standing.
- The user's certificate authorizes the user to make Payword chains, and assures vendors that the user's paywords are redeemable by the broker.

Typical scenario.

- The user U clicks on a link to a vendor V's charged web page.
- The user's browser determines whether this is the first request to V that day.
  - *For a first request, U computes and signs a* commitment *to a new user-specific and vendor-specific chain of payments* $c_1, c_2, \ldots, c_N$.
    - The user creates the payword chain in reverse order by picking the last payword c_n at random, and then computing

      $c_i = h(c_{i+1})$ for $i = N-1, N-2, \ldots, 0$.

      Here c_0 is the root of the payword chain, and is not a payword itself. The *commitment* contains the root c_0, but not any payword c_i for $i > 0$.
    - commitment M = {V, UCert, c_0, Date, OtherInfo}
    - commitment includes both identities of the user and the vendor, and so user-specific and vendor-specific.
  - *The user provides this commitment and her certificate to the vendor V, who verifies their signatures.*
- The *i*-th payment (for $i = 1, 2, \ldots$) from U to V consists of the pair (c_i , *i*), which V can

verify using c_{i-1}.

- At the end of each day, V reports to the broker B the last (highest-indexed) payment (w_l, l) received from each user that day, together with each corresponding commitment.
- The broker charges subscription and/or transaction fees.

Figure 4-16 shows the generation of hash-chain and commitment in the above scheme.

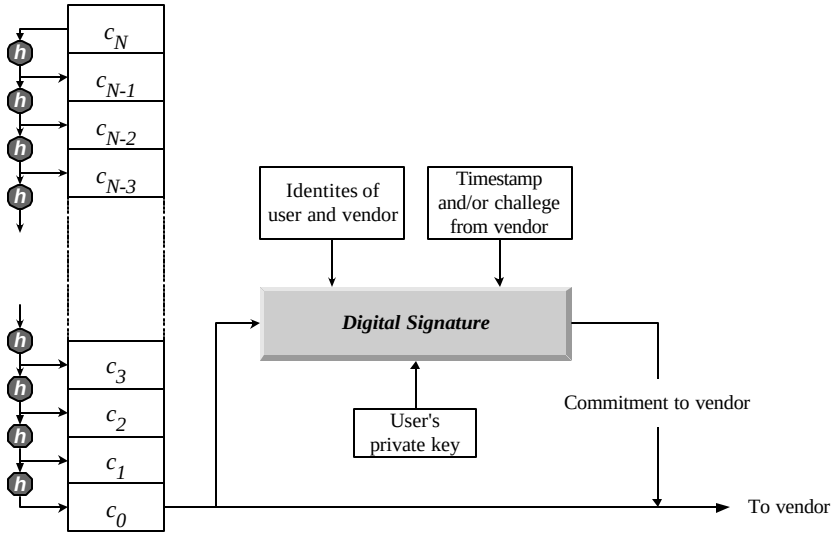


Figure 4-16: Hash Chain Protocol

4.4.3.2 Tamper-Resistant Device Using Shared Keys

An alternative to use of a digital signature for micropayments is to employ a tamper-resistant device together with symmetric key cryptography. One such scheme called Small Value Payment (SVP) was proposed by Stern and Vaudenay [StVa97]. It aims to provide even cheaper and effective micropayment scheme than the approach using hash chains, by avoiding the use of asymmetric key cryptography. However, it requires the use of tamper-resistant devices both at the consumer and the merchant sides. This system is outlined in Figure 4-17.

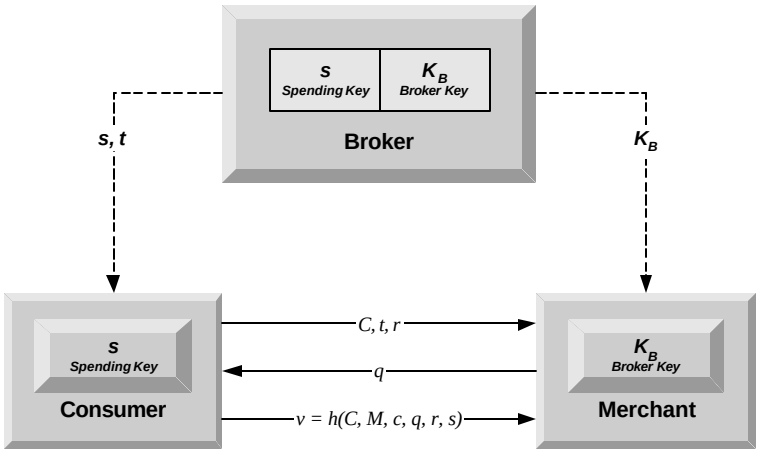


Figure 4-17: Small Value Payment Protocol

Initialization. The broker B fixes its own secret key k_B and communicates it in a secure way to the device of each merchant, where k_B is a common value to all merchants. It also generates and computes a *random token* t and a *spending key* $s = Mac_{k_B}(C,t)$ for a user, where s is unique to the user. The payment protocol can be described as follows.

Payment protocol.

C: consumer, M: merchant

C $\rightarrow$ M: C, t, r	(r: random number chosen by C)
C $\leftarrow$ M: q	(q: random challenge)
C $\rightarrow$ M: $v = h(C, M, c, q, r, s)$	(c: microamount)

M: checks if $v = h(C, M, c, q, r, Mac_{k_B}(C,t))$,
keeps an account balance for the user and increases C's account by c, and
(optionally) stores (t, q, r, v) if he is suspicious about this payment.

Payment clearing.

The merchant regularly sends the broker the amount of money spent by consumers, and the broker monitors if the accounts are consistent. If not, the broker requests a valid proof (C, c, t, q, r, v) of payment from M. If it cannot be provided, the broker just refuses the payment and records that there must be a problem with C or M. *If such a proof is released, the broker pays and checks if* (M, q, r) *has already credited to M.* If it has, the broker suspects the merchant to be dishonest. If not, the broker stores (M, q, r) in the (C, t)-records.

Problems with this scheme.

- There is no signature from the user, and thereby the scheme does not provide non-repudiation (the merchant and/or the broker can generate all the security parameters). This is why the scheme depends heavily on the use of tamper resistant devices. The compromise of only one tamper resistant device in the merchant side enables an attacker to impersonate other consumers.
- The shared secret key k_B between the broker and all the merchants must be the same, because the every merchant (more precisely, its tamper-proof device) in transaction with the customer should be able to compute the spending key, and the user's spending key is a function of the shared key k_B ($s = Mac_{k_B}(C,t)$).
- The user and the VASP must execute the three-way challenge-response protocol for every micropayment, which is inefficient compared with the hash chain approach exchanging only one message (preimage of a hash chain).
- *Weakness in the message freshness*: the mechanism of replay-detection against the merchant is vulnerable to the following attack scenario:
  - *Broker resets all the account records periodically, e.g., every month.*
  - *Merchant reuses the old parameters* (*used in the previous months*).
  - *Broker checks if* (M, q, r) *has been used before, but the check cannot be applied to all the transaction out of the manageable period.*

The last problem with regard to replay attack can be easily fixed by adopting an additional commitment which is generated by the consumer and checked by the broker, and including date information in the commitment computation procedure. This prevents a merchant from cheating the broker with old payment data received from the user previously.

4.4.3.3 An Enhanced Scheme Using Tamper-Resistant Devices

Exploiting the advantages of both the hash chain and the tamper-resistance scheme, we can devise a new scheme. That is, we can avoid both the expensive asymmetric cryptography even for the payment initialization, and challenge-response for each payment of microamount.

Figure 4-18 shows the required setting of this scheme assuming the use of tamper-resistant devices. The role players in this scheme are taken by those in mobile environments: the user, the VASP, and the user's SP. There are three distinct kinds of shared secret keys: K_{US} between the user and the SP, K_{VS} between the VASP and the SP, and K_{UV} between the user and the VASP. In fact, the shared key K_{UV} is derived from the K_{VS} which is common to every VASP. Also, for simplicity of key management, the user-SP shared key K_{US} is computed using the user identity and a master key K_S of the SP.

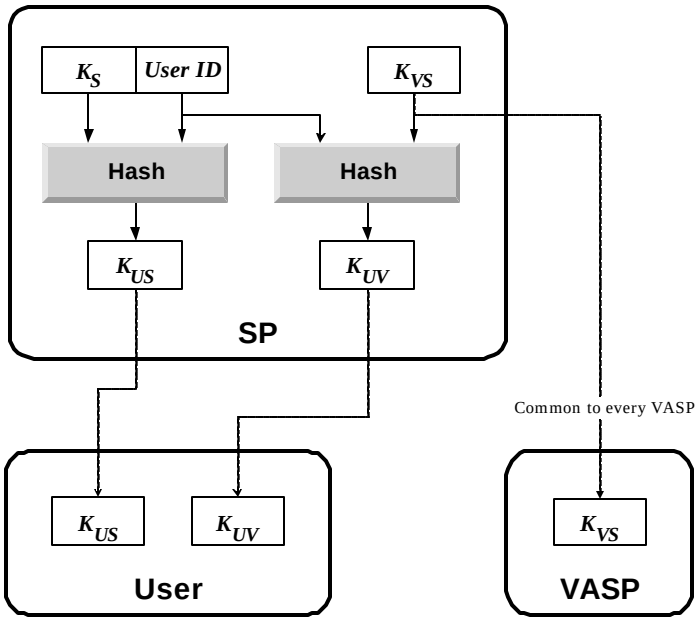


Figure 4-18: Enhanced Payment Scheme

Payment protocol assuming the use of hash chain is described in Figure 4-19. The user generates two separate commitments: one to the VASP, and another to the SP. The usage of pre-images of the hash chain is the same as in Section 4.4.3.1. The computation of commitment for the VASP uses the relevant shared keys K_{UV}, the identities of the user and the VASP, random challenge r_V and timestamp TS_V from the VASP, and the result of hashing c_0. This commitment value is, in turn, input, together with the shared key K_{UV}, to the commitment generation procedure for the SP. Note that by including the timestamp which may be simply the date (yymmdd), this scheme is secure against the replay attack by the VASP which was possible in the scheme described in the previous section. The burden of computing the hash chain, if any, may

be alleviated by reusing the previously generated hash chain in a sense that the remaining preimage with the largest index is used for the commitment generation. Summarizing, the setting of this scheme is basically from the SVP mechanism proposed by Stern and Vaudenay, and the actual payment protocol from the hash chain scheme.

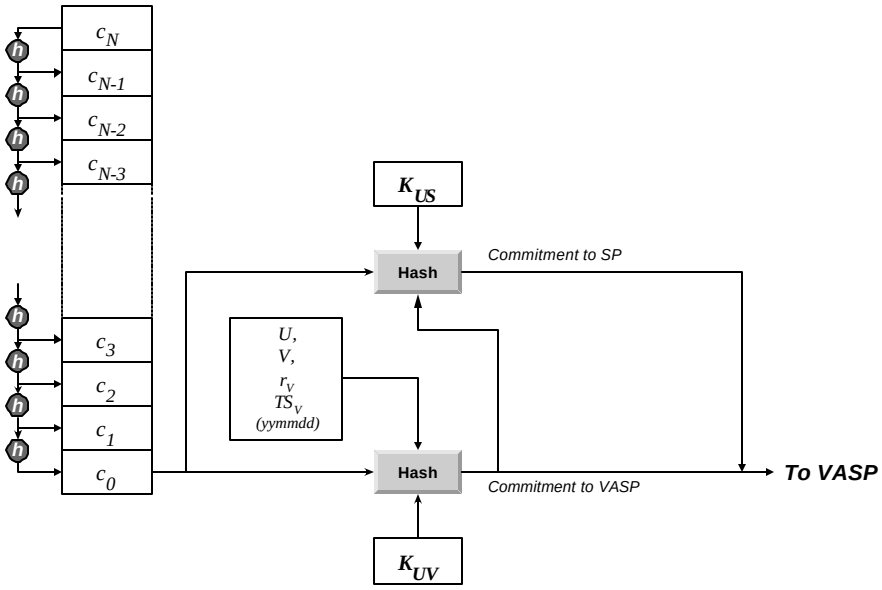


Figure 4-19: Payment Protocol

An example payment protocol set based on this enhanced scheme is shown in the following. We first summarise the goals of payment initialization protocol is as follows.

- Mutual authentication between the user and the VASP
- Authentic and secure key establishment
- Mutual session key control
- Weak non-repudiation of the user to both the VASP and the SP based on shared keys
- Payment parameter initialisation

Payment initialization protocol.

U: *User*, V: *VASP*, S: *SP*

1. U $\rightarrow$ V: U, r_U
2. U $\leftarrow$ V: r_V, $h(r_U, r_V, K_{UV})$, ch_data, TS_V
3. U $\rightarrow$ V: c_0, *commitment_V*, *commitment_S*

U: *commitment_V* = $h(U, V, K_{UV}, r_U, r_V, TS_V, ch\_data, c_0)$
commitment_S = h(*commitment_V*, K_{US})

Protocol description.

- *First message*: the user sends the VASP his identity U and a random challenge r_U.
- *Second message*: the VASP computes the common shared secret key K_{UV} using the user identity U and the share secret key K_{VS}, chooses a random challenge r_V and computes $h(r_U, r_V, K_{UV})$. It delivers to the user the random challenge r_V, $h(r_U, r_V, K_{UV})$, charging data ch_data and the timestamp TS_V.
- *Third message*: upon receiving the second message from the VASP, the user computes $h(r_U, r_V, K_{UV})$, the value of which is compared with the received hash value from the VASP. The match of two values guarantees that the VASP has an authentic secret key K_{VS}. After that, the user computes the two commitments to the VASP and the SP using the secret keys K_{UV} and K_{VS}, respectively. The VASP checks the first commitment value by computing the same calculation as the user and comparing the result with the received value. If both values match, then it may gain assurance that the user has the correct shared secret key K_{UV}.

Payment protocol.

U: *User*, V: *VASP*

1. U $\rightarrow$ V: c_j (j = 1,…,N)

Protocol description.

When the user and VASP need to exchange the actual payment data for a unit of charged service, the user sends the relevant preimage of the hash chain. Note that the three-way challenge-response messages are not used but a simple tick (a preimage of the hash chain) is delivered from the user to the VASP when required. Therefore this

scheme achieves a significant improvement in terms of signalling load from the tamper-resistant device scheme proposed by Stern and Vaudenay.

Payment clearing.

$V: VASP,\ S: SP,$

1. $V \rightarrow S: c_0,\ c_{j\_max}, j\_max, U, V, r_U, r_V, ch\_data, TS_V, commitment\_V, commitment\_S$

Protocol description.

After the transaction, the VASP stores the payment data for billing, which includes the user identity U, the user's commitments to the VASP and the SP, and the required data for the verification of the signature $r_U, r_V, ch\_data, TS_V, c_0$, the last received pre-image c_{j_max}, and the total number of ticks j_max paid by the user in the transaction. The SP checks the $commitment\_S$ field by computing the same calculation as the user and comparing the result with the received value. The confidence gained by this check is to ensure that the commitment could not have been formed by the VASP, even if the tamper resistance of the VASP's device has been compromised.

If we compare the disadvantages of the SVP scheme with our enhanced version we see that all disadvantages have been overcome except that there is still no signature to provide non-repudiation of user payments. However, even in this regard there is a significant improvement since only the broker itself is able to forge user commitments, and not merchants as in the original SVP scheme.

4.4.4 Support for Electronic Payment in the WAKE Protocol

The ASPeCT project investigated the electronic payment mechanisms in the future mobile communications [ASPe97b, ASPe98, MPM98] and chose the micropayment scheme, *hash chain together with digital signature*, as the solution, which is based on Pedersen's proposal [Pede95]. This is because the scheme supports non-repudiation of user action and achieves computational efficiency at the same time through the use of digital signature and hash chain, respectively. We follow the same line as the ASPeCT project in this research and our aim here is to make the WAKE protocol support the same micropayment scheme.

The WAKE protocol proposed in Chapter 3 was already designed with the nonrepudiation of user action in mind. Actual payment takes place in the charge ticks protocols which is very cheap in computation because of no public-key computation and off-line property.

The use of the same WAKE protocol for both user-NO and user-VASP interfaces will also be able to support more efficient integration of the incontestable charging for both value added services and basic telecommunication services. In mobile communication environments, we have already well established infrastructure for billing users. It means that we do not need to establish extra clearing/banking infrastructure for mobile E-payment. The roles with regard to billing is shown in Figure 4-20, which is adopted from the ASPeCT project [ASPe97b].

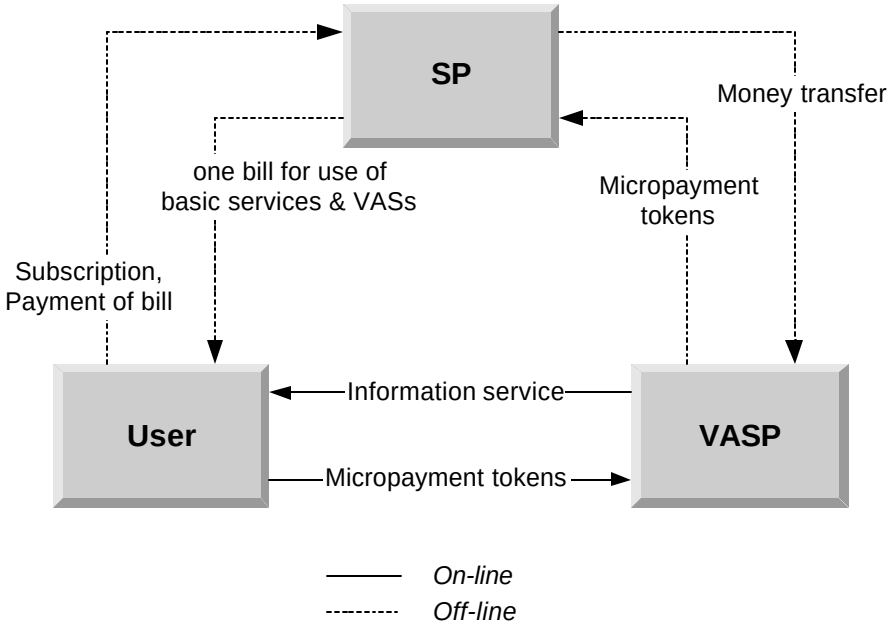


Figure 4-20: Billing of Micropayments

The role players in mobile micropayment comprise mobile users, SPs and VASPs. Here, a SP plays the role of the broker in general micropayment environments. It bills the user for both basic and value-added services, and then redeems the relevant payment to the VASP. Considering the light-weight nature of most transactions to be carried out through mobile communications, the VASP-SP interface will be usually off-line.

Satisfying this requirement is straightforward as shown in the next section, but the on-line interaction between the VASP and SP can be accommodated using the WAKE protocol variant C2 and the E-payment described in the next section.

The required modification of the WAKE protocol for authentication and payment initialization is the addition of the relevant data required for the micropayment protocol, including incorporation of the relevant fields into the signature calculation. Most of the E-payment related descriptions in this section are based on [ASPe97b], [ASPe98] and [MPM98]. The whole set of E-payment protocols consists of the following three protocols.

- Authentication and initialization of payment protocol
  - *Running the WAKE protocol, the user and the VASP authenticate each other and establish a new secret session key. The user also commits himself to payment related data such as initialization parameters and charging data containing the tariff.*
  - *Depending on the availability of public keys or certificate chains required, one of the three variants A, B and C is executed. In addition, variant C may be selected when the on-line checking of certificate status is required in line with security policy of one or both entities.*
- Re-initialization of payment protocol
  - *This protocol is simply a reduced version of protocol variant A to enable the user to give the VASP a new commitment to the payment related data when the user has run out of the charge ticks* (*preimages of hash*).
- Charge ticks protocol
  - *After establishing the user's commitment to the payment data, both entities execute the very light-weight charge ticks protocol, which delivers actual payment from the user to the VASP.*

Notations.

Finally, some additional notation is adopted to describe the E-payment mechanism, which roughly follows the convention used in ASPeCT protocol.

- *CertChain*(*A*,*B*): a certificate chain from which *A* can retrieve an authentic copy of *B*'s public key, which is usually required in the situation where TTP_A and TTP_B are

different from each other.

- *ch_data* : charging data field, containing the relevant tariff on which charges should be based
- TS_B: a timestamp generated by the VASP *B*.
- *IV*: A random initialization vector used for the hash function used in hash-chain
- c_T : the initialization parameter for E-payment (a commitment to VASP)
- *n*: the number of ticks whose payment is requested by the VASP
- *c*: a preimage of a hash value corresponding the number of ticks whose payment is requested by the VASP

4.4.4.1 Authentication And Initialisation Of Payment Protocol

Protocol 62. *Generic protocol variant A for E-payment*

A: user, *B*: VASP

1. A $\rightarrow$ B : $BPubKey\{r_A\}_R$
2. A $\leftarrow$ B : r_B, $SessKey_{AB}\{r_A\}_{NR}$, *ch_data*, TS_B
3. A $\rightarrow$ B : $APriKey\{B, r_B, r_A, ch\_data, TS_B, c_T, IV\}_{NR}$, $SessKey_{AB}\{A, c_T, IV\}_R$
4. A $\leftarrow$ B : *AuthAck*

A, B : $SessKey_{AB} = f(r_A, r_B)$

Protocol 63. *Instance protocol for E-payment: No forward secrecy (NFS) variant*

A: user, *B*: *VASP*

1. A $\rightarrow$ B : $(g^b)^{r_A}$
2. A $\leftarrow$ B : r_B, $h(g^{r_A}, K_{AB})$, *ch_data*, TS_B
3. A $\rightarrow$ B : $\{h(B, r_B, g^{r_A}, ch\_data, TS_B, c_T, IV)\}_{K_A^{-1}}$, $\{A, c_T, IV\}_{K_{AB}}$
4. A $\leftarrow$ B : *AuthAck*

A, B : $K_{AB} = h(g^{r_A}, r_B)$

The above protocols are extended versions of the proposed WAKE generic and instance protocols, respectively. Here, we show only the generic variant A and the instance protocol NFS, because all the E-payment data (indicated in italic font in the above protocol transcripts) just apply in the same way to the other generic/instance variants. The security of the protocols with regard to authentication and key establishment is not modified at all with the inclusion of E-payment data. The following description concerns only the E-payment related aspects.

Second message: the VASP includes a timestamp TS_B and the charging data field *ch_data*. By delivering the timestamp to be digitally signed by the user together with other relevant data, the VASP can have assurance about nonrepudiation of the user's payment data.

Third message: on receiving the third message, the user checks the acceptability of the timestamp TS_B and the charging data *ch_data* , which should be able to be confirmed by the corresponding human user via GUI according to the user's preference. In addition to TS_B and *ch_data*, the user's digital signature also includes an initialisation vector *IV* and the *T*-th image c_T of a hash chain that is calculated by applying *T*-iterations of a hash function with $c_0 \| IV$ as the start value, where c_0 is random number and "‖" denotes concatenation. For each i (i = 1, …, T), $c_i = h(c_{i-1} \| IV)$. From the signature of the user, the VASP can gain assurance that the user agrees with the charging tariff (because of *ch_data* included in the signature) and this signature can be used for incontestable charging (because of TS_B included in the signature).

4.4.4.2 Re-Initialisation Of Payment Protocol

Protocol 64. *Generic protocol*

A: user, *B*: VASP

1. A ← B : *ch_data*, TS_B
2. A → B : *APriKey*{*B*, r_A, *ch_data*, TS_B, c_T, *IV* }$_{NR}$, c_T, *IV*
3. A ← B : *AuthAck*

Protocol 65. *Instance protocol*

A: user, *B*: *VASP*

1. A ← B : *ch_data*, TS_B
2. A → B : $\{h(B, g^{r_A}, ch\_data, TS_B, c_T, IV)\}_{K_A^{-1}}$, c_T, *IV*
3. A ← B : *AuthAck*

When the previously computed tick tokens are exhausted or the user and the VASP need to agree upon a different service item and the related charging data, they re-initialize all the relevant transaction data using the re-initialization protocol as shown in Protocol 64 and Protocol 65. Authentication and key establishment have already completed in the first run of the WAKE protocol, and so all the relevant data can be

omitted in the protocol. Only the E-payment related data establishment is the goal of this protocol. The inclusion of r_A and g^{r_A} in the generic and instance protocols, respectively, which were used to challenge the VASP, is to give the user assurance that its own signature in the protocol is well bound to the current session. The timestamp TS_B from the VASP may look sufficient for that purpose, but it is not the case because the user is very limited with regard to the access to a reliable clock. The timestamp is adopted to address the incontestable charging which concerns the VASP.

4.4.4.3 Charge Ticks Protocol

Protocol 66. *Instance protocol*

A: user, B: VASP

1. A ← B: n
2. A → B: c

Protocol 66 describes a tick payment protocol for actual payment by the user, which is executed between the user and the VASP, following the authentication and initialization of payment protocol. The VASP requests the user to pay n ticks for the service or information requested by the user. The user, in turn, pays the n ticks by releasing the corresponding pre-image value c to the VASP. Here $c = F^{T-(t+n)}(c_0)$ where $F(x) = h(x \| IV)$ and t is the number of ticks paid previously.

Clearing of tick payments. After the transaction, the VASP stores the payment data for billing, which includes the user identity A, the user's commitment (digital signature) $APriKey\{B, r_B, r_A, ch\_data, TS_B, c_T, IV\}_{NR}$ and the data required for the verification of the signature $r_B, r_A, ch\_data, TS_B, c_T, IV$, the last received pre-image c, and the total number of ticks paid by the user in the transaction.

In the foreseeable future, mobile communication terminals will be a major method for electronic commerce at least in transactions of small amounts. The well-studied and efficiency-proven hash chain scheme has been integrated into the WAKE protocol in line with the ASPeCT project. Actual application environments may require some extra or fewer of the E-payment related data fields, but the essential framework of the E-payment protocol and mechanism proposed in this section need not be modified.

4.4.5 Complexity of the WAKE Protocol with Micropayment Support

The required exponentiations for the micropayment scheme adopted in this research is due to digital signature by the user and the corresponding signature verification by the network or VASP. These computations, however, are already built in the WAKE protocol, and hence no extra exponentiation is required to support the micropayment scheme as we can see from Protocol 62 or Protocol 63.

4.5 WAKE Protocol Equipped with All Add-On Mechanisms

In this section, for the sake of comprehensiveness, we present the instance WAKE protocol combined with all add-on mechanisms described in this chapter, as shown in Protocol 67. This protocol illustrates that any subset of all add-on mechanisms can be combined with the WAKE protocol.

Protocol 67. *Instance WAKE protocol equipped with all add-on mechanisms*

A: *user*, B: *network*

B: broadcasts a random number r in addition to its public key g^b

1. A $\rightarrow$ B: $(g^b)^{r_A+r}$, $h(g^{r_A}, B)$
2. A $\leftarrow$ B: $g^{r_A r_B}$, $h(g^{r_A}, g^{r_A r_B})$, *ch_data*, TS_B
3. A $\rightarrow$ B: $\{h(B, g^{r_B}, g^{r_A}, ch\_data, TS_B, c_T, IV)\}_{K_A^{-1}}$, $\{A, c_T, IV\}_{K_{AB}}$, $AUTHR$, $COUNT$
   where $AUTHR = h(K_t, COUNT)$

A, B: $K_{AB} = h(g^{r_A}, g^{r_B})$

This protocol adopts the second prototype for forward secrecy as described in Section 4.1.1 and therefore has very similar appearance to the instance WAKE protocol *FS2* as shown in the description, Protocol 52. It also supports DoS attack prevention, cloning detection and micropayment as described in the previous sections in this chapter.

The most interesting point in this protocol may be how much this all-contained version of the WAKE protocol costs. The answer may be, however, a bit different from our first expectation: "just the same as the instance WAKE protocol *FS1* or *FS2*". As described in the relevant sections, the support for clone detection and micropayment does not require additional number of exponentiations. The introduction of

cryptographic salt to mitigate the DoS attacks cost is entirely zero for the user side and virtually zero for the network side (see Section 4.2.4). Though equipped with all add-on facilities, this protocol is quite artificial, and hence is not necessarily a reasonable candidate for practical environment. The clone detection is not relevant for the user-VASP interface whereas forward secrecy mechanism may be somewhat costly for the user-NO interface.

4.6 Summary

The supplementary goals, identified in 1.2, are investigated and relevant mechanisms are devised when required. Forward secrecy will increasingly be a growing concern as today's network like Internet is moving to more and more open and distributed structure. To obtain computationally cheap solution to forward secrecy, all the possible ways or prototypes of forward secrecy are investigated, and it was found that we have two prototypes available, both requiring one discrete exponentiation in each principal. The first one depending on a particular property of key agreement functions F and the second one exploiting confidentiality by temporary asymmetric key pairs. The latter one was not known to most crypto people and has not been spotlighted with a second thought at all. Its true value is disclosed and several examples of implementation are presented. Both prototypes are applied to the WAKE protocol which resulted in two different instance protocols as presented in Section 4.1.4.

As for DoS attacks, a new concept of protection is devised, which can be applied to any protocols which use public-key encryption to authenticate the server or network to the user, i.e., the protocols with $DA_{F,ACK}$ or $DA_{F,NoAck}$ as the prototype for B to A authentication.. Among examples of such a server authentication scheme are the WAKE protocol in the thesis, the Internet security protocol SSL/TLS [RFC99] and the authentication and key agreement protocol of the PACS (Personal Access Communication System), one of the six PCS standards in North America [Bell94], [JTC94]. Several alternative schemes available today are reviewed, and argued to be more expensive than the scheme presented in this chapter. The cookie, an existing countermeasure, is useful against the DoS attack, but useless for a determined attacker because the cookie data can be eavesdropped by the attacker. The client puzzle approach solves the problem but requires additional computational overhead in both the

client and the server. The new concept proposed in the thesis solves all these problems without minimal overhead because it requires no additional public-key operation. In some concrete implementation of the concept, it may require one hash computation, which is practically insignificant. Furthermore, this protection method can be combined well with the existing cookie mechanism as well, providing more robustness against the DoS attack. The countermeasure proposed here or any other similar facility cannot be applied, without significant computation cost, to the STS or the ASPeCT protocol because they belong to a different generic protocol "OA_F-OA_F", the structure of which is not able to be equipped with such a faciltiy.

A clone detection method is proposed using *temporary* shared secret key like SSD of the North American cellular standard IS-41. Its applicability to third generation mobile communication is studied and described. The detection scheme, however, is found to be not directly related to the WAKE protocol but to operation and management of subscribers by mobile communication operators.

To support electronic payment, the WAKE protocol is enriched with regard to its field items. Micropayment schemes are reviewed and a new scheme proposed, which is in fact a selective combination of two existing approaches: hash chain concept without digital signature and shared key based challenge-response from the SVP scheme. In this way, we can take advantage of the benefits from both schemes, that is, cheap crypto by use of shared key scheme, and only one move for each microamount payment in a payment transaction by use of hash chain. The WAKE protocol, however, cannot use the new scheme because it is based on public-key cryptography. Instead, it provides non-repudiation of user initiation of the payment transaction by use of digital signature. Hash chain scheme is adopted for micropayment and a detailed description is given of how the WAKE protocol and hash chain can be combined.

Finally it will be worth making clear that all the add-on facilities, except for the coutermeasure againt DoS attacks, discussed in this section does not assume the use of a particular authentication prototype as defined in Section 2.1. For example, the ASPeCT protocol can benefit from the use of forward secrecy too.

Chapter 5.

Conclusions and Future Work

Design and analysis of authentication and key establishment protocols for mobile communication systems is an important issue in these days of highly mobilized society. The mobile terminal and the radio channel are two big hurdles to be overcome as we design secure and computationally efficient authentication and key establishment protocols for mobile communication. There have been a few relevant research activities for mobile security such as UK LINK project [LINK96] and ETSI initiated ASPeCT project [ASPe96]. In these projects, they designed several WAKE protocols based on symmetric key or asymmetric key based cryptography. The WAKE protocol designed in the LINK project uses symmetric key cryptography while the ASPeCT project investigated several proposals including the protocol from the LINK project. Two asymmetric key based WAKE protocols were analysed and refined, both of which belong to the same archetype or generic protocol OA_F-OA_F as the STS protocol (see Appendix: Prototypes of Authentication Protocols.).

In this thesis, a new generic WAKE protocol is designed based on a different prototype $DA_{F,Ack}$-OA_F, and several instance protocols are developed from the same generic protocol. The prototype later turned out to be the same one upon which the famous SSL/TLS protocol is built. Therefore several contributions from this thesis can be adopted to the SSL/TLS protocol as well. More importantly, however, in this research two novel schemes are proposed for systematic design and analysis of authentication and key establishment protocols as described in Chapter 2.

5.1 Summary of Contributions

The research for the thesis has started with the aim to design a new generic WAKE protocol for future mobile communication security. The effort to design in a systematic and error-free way has found that there is no commonly agreed classification scheme available of authentication protocols. This has led to establishment of the first classification scheme which is close to our intuition and simple but practical to be used for design and analysis of authentication and key establishment protocols.

The classification scheme is based on the essential elements such as identity and freshness data in cryptographic protocols, which are termed *cryptographic particles*. This work, in turn, was followed by establishment of a very simple but comprehensive modelling of authentication and key establishment, which may be built upon cryptographic particles and their interaction in protocols. With this new model, it turned out that we do not have to resort to sophisticated methods from advanced mathematical logic or computer science. Another important result in this work is a new perspective towards goals of authentication and key establishment protocols. Most goals frequently referred to in the literature were critically reviewed and, arguably, turned out to be quite ambiguous and sometimes redundant.

On the contrary, several new definitions of entity authentication and key authentication presented in the thesis are based on a fundamental key concept, i.e. cryptographic particles, and therefore achieved simplicity and clarity. Furthermore, this work enables us to see key authentication in a new light and showed that we can obtain a very different definition of key authentication which seems to be clearer and fundamental in the sense that we do not have to use another concept, *entity authentication*, to describe *key authentication*. These two different kinds of authentication may be achieved separately; that is, we consider pure entity authentication protocols and likewise pure key establishment protocols. The latter point is clear when we consider the Diffie-Hellman protocol. Previous work in the literature noted the important aspect of the Diffie-Hellman protocol that it enables two principals to agree on a new secret session key but with entity authentication not guaranteed. Unfortunately, previous work failed to go beyond this important observation to disclose a more complete definition of key authentication. Such a new definition of key

authentication is presented in the same fashion as that of entity authentication. The lack of such definition of key authentication may cause some misunderstanding about the features of a given protocol. Such a case is illustrated through discussion of Shamir's no-key protocol in Section 2.2.3. These findings were integrated to build a new hierarchy of goals of authentication and key establishment protocols, which might look quite radical but in fact accords with rational common sense.

With the above methodologies, a new generic WAKE protocol was designed and analysed. The WAKE protocol designed in the thesis is of a different prototype from the STS and the ASPeCT protocol according to the classification described earlier. Several variants of the same generic protocols are designed to cater for different conditions with regard to public-key availability in protocol principals. Protocol analysis based on new modelling of authentication and key establishment, as proposed in Section 2.2, is applied to *prove* that the intended goals of the protocol are achieved. Moreover, the characteristic structure of the generic protocol is found to be fertile so that it can be implanted with a protection mechanism against the DoS attack. This is not possible for the STS or the ASPeCT protocol because they belong to a different generic protocol "OA_F-OA_F", the structure of which is not able to support such a mechanism..

Several instance protocols compliant with the same generic protocol are also designed: NFS, FS1 and FS2. They differ in whether they do not or do provide forward secrecy using different prototypes of forward secrecy. Their complexities, in terms of the number of modular exponentiations, were compared to the STS and the ASPeCT protocol, and turned out to be equally effective as the ASPeCT protocol. Forward secrecy versions of the instance protocols, i.e. FS1 and FS2, do require more computations but the same goes for the ASPeCT protocol when it is modified to achieve forward secrecy.

The generic and instance protocols are enriched with several add-on facilities such as forward secrecy, robustness against denial-of-service (DoS) attacks, support for electronic payments, and clone detection. Especially, two generic prototypes of forward secrecy are identified and given a new light with regard to their computational cost and their usage for protocol design. A new scheme of micropayment is proposed to selectively take advantage of two different schemes: digital signature with hash chain, and the SVP scheme based on symmetric key cryptosystem. A detailed description is

given of how the generic WAKE protocol can be combined with hash chain micropayment. The notorious DoS attack was also addressed, and a virtually zero cost solution was proposed which is based on a new way of using of random numbers in public-key encryption. A detailed description of the scheme and its usage examples are given. Specifically, the countermeasure is plugged into the WAKE protocol designed in the thesis. Though being outside of the protection provided by authentication protocols, cloning fraud is also tackled. A novel scheme based on temporary shared key concept is proposed.

5.2 Future Work

In the future, it is hoped to further develop the authentication modelling presented in the thesis into a more thorough level and probe the way to achieve an automatic analysis tool. Application of the method to other security protocols, for instance group key distribution protocols, may also be investigated.

Authentication protocol classification scheme proposed in the thesis was originally intended to help systematic study of *mutual* authentication protocols between two principals. Its application to authentication protocols involving three or more entities may be searched.

Systematic design of authentication and key establishment protocols as demonstrated in the thesis may be further streamlined. The sequential step-wise design, from generic protocols for intended security goals to appropriated instance protocols with several plug-in mechanisms like forward secrecy and DoS attack proof, will be worth refining.

References

[Abel98] H. Abelson et al., "The risks of key recovery, key escrow, and trusted third-party encryption", A Report by an Ad Hoc Group of Cryptographers and Computer Scientists, 1998. Available: http://www.cdt.org/crypto/risks98/

[AbNe94] M. Abadi and R. Needham, "Prudent engineering practice for cryptographic protocols", *IEEE Symposium on Research in Security and Privacy*, IEEE Computer Society Press, 1994.

[ACTS97] Advanced Communications Technologies & Services. ACTS Overview. Advanced Communications Technologies & Services, European Commission, 1997.

[Ahuj96] V. Ahuja, *Secure Commerce on the Internet*, Academic Press, 1996.

[AMS95] R. Anderson, H. Manifavas, and C. Sutherland, "A practical electronic cash system", *Personal Communication*, December 1995.

[ANL01] T. Aura, P. Nikander, and J. Leiwo, "DOS-resistant authentication with client puzzles", *Proc. Security Protocols Workshop 2000*, Lecture Notes in Computer Science, Cambridge, UK, April 2000, Springer-Verlag 2001.

[ASPe96] ACTS AC095, project ASPeCT, *Initial report on security requirements*, AC095/ATEA/W21/DS/P/02/B, February 1996.

[ASPe97] ACTS AC095, project ASPeCT, *Definition of fraud detection concepts*, AC095/KUL/W22/DS/P/06/A, 1997.

[ASPe97b] ACTS AC095, project ASPeCT, *Secure billing: evaluation report*, AC095 / SAG/ W25 / DS / P / 16 / 1, May 1997.

[ASPe98] ACTS AC095, project ASPeCT, *Report on final trial and demonstration*, AC095 / PFN/ W12 / DS / P / 19 / 1, April 1998.

[AuNi97] T. Aura and P.Nikander, "Stateless connections", *Information and Communications Security (ICICS' 97)*, 1997, pp.87-97. Available: http://saturn.hut.fi/html/staff/tuomas.html

[Aziz98] A. Aziz, "SKIP extension for perfect forward secrecy". Available: http://www.skip-vpm.org/wetice98/HacknSlash.html

[BAN90] M. Burrows, M. Abadi, and R. Needham, *A logic of authentication*, DEC

Systems Research Center, Report 39, revised February 22, 1990.

[BeMi99] M. Bellare and S.K. Miner, "A forward-secure digital signature scheme", *Advances in Cryptology - CRYPTO '99, LNCS 1666*, Springer-Verlag, 1999.

[Bell94] TR-INS-001313, Generic Criteria for Version 0.1, *Wireless Access Communications Systems (WACS)*, Bellcore, Revision 1, June 1994.

[BeYa93] M.J. Beller and Y. Yacobi, "Fully-fledged two-way public key authentication and key agreement for low-cost terminals", *Electronics Letters*, 29, May 1993, pp.999-1001.

[BGH93] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, J. Bierbrauer, T. Johansson, R. Molva, and M. Yung, "Systematic design of a family of attack-resistant authentication protocols", *IEEE Journal on Selected Areas in Communications*, 11, 1993, pp.679-693.

[Blac99] U. Black, "Third generation mobile systems (TGMSs)", *Second generation mobile & wireless networks*, Parentice Hall, 1999.

[BoPr95] A. Bosselaers and B. Preneel, Eds., Integrity Primitives for Secure Information Systems: Final Report of RACE Integrity Primitives Evaluation RIPE-RACE 1040,LNCS 1007, Springer-Verlag, New York, 1995.

[Boyd97] C. Boyd, "Towards extensional goals in authentication protocols", *DIMACS Workshop on Cryptographic Protocol Design and Verification*, 1997.

[Buha97] K. Buhanal, et al., "IMT-2000: Service providers' perspective", *IEEE Personal Communications*, August 1997.

[Burm94] M. V. D. Burmester, "On the risk of opening distributed keys", *CRYPTO '94, LNCS 839*, Springer-Verlag, 1994, pp.308–317.

[Calh88] G. Calhoun, "Digital cellular — a view from America," *Communications Systems Worldwide*, Oct. 1988, pp.20-25.

[Cert96] CERT, *Advisory CA-96.21: TCP SYN flooding and IP spoofing attacks*. Available: http://www.cert.org/advisories/index.html

[ClJa95] J. Clark and J. Jacob, "On the security of recent protocols", *Information Processing Letters*, 56, 1995, pp.151-155.

[DeSa81] D. Denning and G. Sacco. "Timestamps in key distribution protocols", *Communications of the ACM*, 24(8), August 1981.

[DiAl99] T. Dierks and C. Allen, *The TLS protocol: Version 1.0. request for comments: 2246*. Available: http://www.faqs.org/rfcs/rfc2246.html

[DiHe76] W. Diffie and M.Hellman, "New direction in cryptography", *IEEE Transaction on Information Theory*, 22, 1976, pp.644-654.

[DvOW93] W. Diffie, P. van Oorschot and M. Wiener, "Authentication and authenticated key exchanges", *Designs, Codes and Cryptography*, 2,

1992, pp.107-125.

[DwNa98] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail", *Advances in Cryptology – CRYPTO '9 2, LNCS 1462*, Springer-Verlag, 1992, pp. 139–147.

[ElGa85] T. ElGamal, "A public key cryptosystem and signature scheme based on discrete logarithms", *IEEE Transactions on Information Theory*, 31, 1985, pp.469-472.

[Fred95] M. B. Frederick et al., "Cellular telephone fraud anti-fraud system", US Patent # 5,448,760, September 5, 1995.

[Fred99] S. Frede, "Attack scenarios", *Security Systems & Technologies*, September 1999, pp.4-11.

[GaHa93] G. Garrard and R. Harrison, "The introduction of GSM," *Pan-European Mobile Communications*, 12, Winter 1993.

[Garg96] V.K. Garg and J.E. Wilkes, *Wireless and personal communications systems*, Parentice Hall, 1996.

[Goll94] D. Gollmann, "What do we mean by entity authentication", *1994 IEEE Symposium on Research in Security and Privacy*, pp.46-54.

[Goss90] K.C. Goss, "Cryptographic method and apparatus for public key exchange with authentication", US Patent # 4956863, September 11, 1990.

[Gray97] J.W. Gray III, "On the Clark-Jacob version of SPLICE/AS", *Information Processing Letters*, 62(5), 1997, pp.251–254.

[GrFe99] M. Greenstein and T. M. Feinman, *Electronic commerce: Security, risk management and control*, McGraw-Hill, 1999.

[GrMa90] D. Grillo and G. MacNamee, "European perspectives on third generation personal communication systems," *40th IEEE Vehicular Tech. Conf.*, May 6-9, 1990, pp.135-140.

[Günt90] C. Günther, "An Identity-based Key-exchange Protocol", *Advances in Cryptology -EUROCRYPT '89*, Springer-Verlag, 1990, pp.29-37.

[Hill96] E. Hill, "Fraud trends in North Americ a - US and Canada," *Proceedings of the Conference on Fighting Mobile Fraud*, IBC UK Conferences, London, 1996.

[HMM99] G. Horn, K.M. Martin, and C. J. Mitchell, "Authentication protocols for mobile network environment value-added services", submitted, 1999 . Available: http://isg.rhbnc.ac.uk/cjm/Chris_Mitchell.htm.

[HoHs98] G. Horng and C.-K. Hsu, "Weakness in the Helsinki protocol", *Electronic Letters*, 34, 1998, pp.354-355.

[HoPr98] G. Horn and B. Preneel, "Authentication and payment in future mobile systems", *Computer Security – ESORICS 98, LNCS 1485*, 1998, pp.277-293.

[HSW96] R. Hauser, M. Steiner, and M. Waidner, "Micro-payments based on *i*KP", IBM Zurich Research Lab. Available: http://www.zurich.ibm.ch/Technology/Security/publications/1996/HSW96.ps.gz

[IAMi90] C. I' Anson and C. Mitchell, "Security defects in the CCITT Recommendation X.509 — The Directory Authentication Frame-work", *Computer Communication Review*, 20(2), April 1990, pp.30–34.

[IEEE99] IEEE *P1363 Standard Specifications For Public Key Cryptography*, February, 1999.

[ISO95] ISO/IEC 9798-3, "Information Technology – Security techniques – Entity Authentication Mechanisms – Part 3: Entity authentication using a public key algorithm", 1995.

[ISO96] ISO/IEC DIS 11770-3, "Information Technology – Security techniques – Key management – Part 3: Mechanisms using asymmetric techniques", 1996.

[ITU97] ITU-R Recommendation M.1223, "Evaluation of security mechanisms for IMT-2000", 1997.

[Jabl96] D. P. Jablon, "Strong password-only authenticated key exchange", *ACM Computer Communications Review*, October, 1996, pp.5-26.

[JTC94] JTC, Text Modification to JTC(AIR)/94.02.07-119R6, September 15, 1994.

[JTY97] P. Janson, G. Tsudik, and M. Yung, "Scalability and flexibility in authentication services: The KryptoKnight approach", *IEEE INFOCOM' 97*, Tokyo, April 1997.

[JuBr99] A. Juels and J. Brainard, "Client puzzles: A cryptographic counter-measure against connection depletion attacks," *Proceedings of the 1999 Network and Distributed System Security Symposium (NDSS' 99)*. Internet Society, March 1999. Available: http://www.isoc.org/ndss99/proceedings/

[KaSi99] P. Karn and W. A. Simpson, "Photuris: Session-key management protocol", RFC 2522, IETF Network Working Group, March 1999.

[Kraw96] H. Krawczyk, "SKEME: A versatile secure key exchange mechanism for Internet," *Proceedings of the Internet Society Symposium on Network and Distributed System Security*, February 1996. Available: http://bilbo.isu.edu/sndss/sndss96.html.

[Lamp81] L. Lamport, "Password authentication with insecure communication", *Communications of the ACM*, 24(11), November 1981, pp.770-771.

[Laur98] S. St.Laurent, *cookies*, McGraw-Hill, 1998.

[LINK96] LINK Personal Communications Programme [PCP], 3GS3, *Technical Report 2*: *Security Mechanisms for Third Generation Systems*, Final Version, 14th February 1996.

[LNA00] J. Leiwo, P. Nikander, and T. Aura, "Towards network denial of service resistant protocols", *Proc. Sixteenth Annual Working Conference on Information Security (SEC2000)*, IFIP Series, Vol. 175, Beijing, China, August 2000, Kluwer Academic Publishers.

[Lowe95] G. Lowe, "Breaking and fixing the Needham-Schroeder public-key protocol using FDR", Technical Report, PRG, Oxford University, 1995.

[Lowe97] G. Lowe, "A hierarchy of authentication specification", *10 th IEEE Computer Security Foundations Workshop*, IEEE Press, 1997, pp.31-43.

[MaMi98] K.M. Martin and C.J. Mitchell, "Evaluation of authentication protocols for mobile environment value-added services", Technical report, August 1998.

[Mead99] C. Meadows, "A formal framework and evaluation method for network denial of service," *Proceeding of the IEEE Computer Security Foundations Workshop*, IEEE Computer Society Press, June 1999. Available: http://www.itd.nrl.navy.mil/ITD/5540/publications/CHACS/CRYPTO0index.html

[MiTh93] C.J. Mitchell and A. Thomas, "Standardising authentication protocols based on public key techniques", *Journal of Computer Security 2*, 1993, pp.23-36.

[MiYe98] C.J. Mitchell and C. Y. Yeun, "Fixing a problem in the Helsinki protocol", *ACM Operating Review*, 32 no. 4, 1998, pp.21-24.

[MPM98] K.M. Martin, B. Preneel, C.J. Mitchell, H.J. Hitz, G. Horn, A. Poliakova, and P. Howard, "Secure billing for mobile information services in UMTS", in: S. Trigila, A. Mullery, M. Campolargo, H. Vanderstraeten, and M. Mampaey, editors, *Intelligence in Services and Networks: Technology for Ubiquitous Telecom Services*, (Proceedings of the Fifth International Conference, IS&N 98, Antwerp, Belgium, 25-28 May 1998), Springer-Verlag, LNCS 1430, Berlin, 1998, pp.535-548.

[MSSJ98] D. Maughan, M. Schertler, M. Schneider, and J. Turner: "Internet Security Association and Key Management Protocol (ISAKMP)", March 1998. Available: http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ISAKMP/draft-ietf-ipsec-isakmp-09.txt

[MTI86] T. Matsumoto, Y.Takashima, and H. Imai, "On seeking smart public-key-distribution systems", *The Transactions of the IECE of Japan*, E69, 1986, pp.99-106.

[MvOV97] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

[NeSc78] R. Needham and M. Schroeder, "Using encryption for authentication in large networks of computers", *Communications of the ACM*, 21, 1978, pp.993-999,.

[NyRu94] K. Nyberg, and R.A. Rueppel, "Message recovery for signature schemes based on the discrete logarithm problem", *Advances in Cryptology – EUROCRYPT '94, LNCS 950*, Springer-Verlag, 1994, pp.182-193.

[OjPr98] T. Ojanpera and R. Prasad, "IMT-2000 Applications", in *Wideband CDMA for Third Generation Mobile Communication*, T Ojanpera and R. Prasad, Eds., Artech House Publishers, 1998, pp.65-76.

[Orma98] H. Orman, "The Oakley key determination protocol", RFC 2412. The Internet Society, November 1998.

[Park97] C.-S. Park, "On certicate-based security protocols for wireless mobile communication systems", *IEEE Network*, September/October 1997, pp.50-55.

[Paul99] L.C. Paulson, "Inductive Analysis of the Internet Protocol TLS", *ACM Transactions on Computer and System Security* 2 3, 1999, pp.332-351.

[Pede95] T.P. Pedersen, "Electronic payments of small amounts", DAIMI PB-495, Computer Science Department, Aarhaus University, August 1995.

[Rapp96] T.S. Rappaport, *Wireless Communications: Principles & Practice*, Prentice-Hall, 1996, Ch. 8.

[Redl98] S.M. Redl, et al., *GSM and Personal Communications Handbook*, Artech House Publishers, 1998.

[RFC99] T. Dierks and C. Allen, The TLS Protocol, [RFC 22246], January 1999.

[RiSh96] R.L. Rivest and A Shamir, "PayWord and MicroMint: Two simple micropayment schemes", *Cryptobytes*, Vol. 2, No. 1, May 1996, pp.7-11. Available: http://theory.lcs.mit.edu/~rivest

[Scho99] B. Schoenmakers, "A simple publicly verifiable secret sharing scheme and its application to electronic voting", *Advances in Cryptology – CRYPTO '99, LNCS 1666*, Springer-Verlag, 1999, pp.148-164.

[Scho99b] V. M.-Schonberger, "The Internet and privacy legislation: Cookies for a treat?". Available: http://www.wvjot.wvu.edu/wvjolt/current/issue1/article

[Schu97] C.L. Schuba et al., "Analysis of a denial of service attack on TCP", *Proc. 1997 IEEE Symposium on Security and Privacy*, May 1997, IEEE Computer Society Press, pp.208–223.

[StVa97] J. Stern and S. Vaudenay, "SVP: a Flexible Micropayment Scheme", *Financial Crypto '97*.

[SyvO94] P. Syverson and P. van Oorschot, "On Unifying Some Cryptographic Protocol Logics", *1994 IEEE Symposium on Research in Security and Privacy*, IEEE Computer Society Press, 1994, pp.14-28.

[TIA95] TIA/EIA/IS-41-C (PN-2991.3), *Cellular Radiotelecommunications Intersystem Operations: Automatic Roaming Information Flows*, May 4, 1995.

[TR45] TR45.3, Appendix A to IS-54, Rev. B Feb 1992.

[vanO93] P. van Oorschot, "Extending cryptographic logics of belief to key agreement protocols", *The 1st ACM Conference on Communications and Computer Security*, November 3-5, 1993.

[WaSc96] D. Wagner and B. Schneier, "Analysis of the SSL 3.0 Protocol", *The Second USENIX Workshop on Electronic Commerce Proceedings*, USENIX Press, November 1996, pp.29-40.

[Wayn97] P. Wayner, *Digital Cash: Commerce on the Net*, Academic Press, 2nd Edition, 1997.

[Wien98] M. Wiener, "Performance Comparison of Public-Key Cryptosystems", *Cryptobytes*, Vol. 1, No. 2, RSA Laboratories, 1998.

[WTLS00] WAP Forum, *Wireless Transport Layer Security Specification*, February 18, 2000, WAP-199-WTLS, Available : HTTP ://www.wapforum.org

[Wu98] T. Wu, "The secure remote password protocol", *Proceedings of the 1998 Internet Society Network and Distributed Systems Symposium*, pp.97-111. Available : ftp://srp.stanford.edu/pub/srp/srp.ps

[YaSh89] Y. Yacobi and Z. Shmuely, "On key distributions", *Advances in Cryptology – CRYPTO '89, LNCS 435*, Springer-Verlag, 1989, pp.344-355.

[YOM90] S. Yamaguchi, K. Okayama, and H. Miyahara, "Design and implementation of an authentication system in WIDE Internet environment", *Proceedings of the 10th Regional Conference on Computers and Communication Systems*, 1990, pp.653-657.

Appendix:

Prototypes of Authentication Protocols

Notation:

- *A*: session initiator, *B*: responder
- $OA_{\varnothing}$- $DA_{\varnothing}$, for instance, means that A authenticates B with $OA_{\varnothing}$, and B authenticates A with $DA_{\varnothing}$.

Prototype-Prototype	Example (Generic Protocols)	Notes	
∅ - Elementary prototype	Æ -OA_S: no authentication of B by A ISO/IEC Public Key One-Pass Unilateral Authentication Protocol [ISO95] A → B: ACert, TS_A, B, Text2, APriKey{TS_A, B, Text1 } Æ -OA_F: no authentication of B by A ISO/IEC Public Key Two-Pass Unilateral Authentication Protocol [ISO95] A ← B: r_B, Text1 A → B: ACert, r_A, r_B, B, Text3, APriKey{ r_A, r_B, B, Text2}		
Elementary prototype - ∅	IA_F- Æ : no authentication of A by B **ISO/IEC key agreement mechanism 2** [ISO96], or **ElGamal** key agreement [MvOV97, p.517] Condition (using "public key transport" mechanism) A : BPubKey B : (BPriKey, BPubKey) Key Token Construction A : (APriKeyX , APubKeyX) A → B : APubKeyX Key Construction A : SessKeyAB = APriKeyX{ BPubKey} B : SessKeyAB = BPriKey{APubKeyX}	Attack construction: E\A → B E : EPriKeyX, EPubKeyX, $SessKey_{EB}$ = EPriKeyX{ BPubKey } (= BPriKey{ EPriKeyX }) E\A → B : EPubKeyX B : $SessKey_{EB}$ = BPriKey{ EPriKeyX } (= EPriKeyX{ BPubKey }) Result : B believes $SessKey_{EB}$ is a good key for A and B, which is known to E and not to A. Hence E can intercept and decipher all messages from B to A, and if A receives a message from B and tries to decipher the message using the old session key $SessKey_{AB}$ which was replaced by $SessKey_{EB}$ within B, the deciphered result will be just a garbage.	
	$DA_{\varnothing}$ - Æ : no authentication of A by B **ISO/IEC key transport mechanism 1**, or **RSA** A → B : BPubKey{ A, $SessKey_{AB}$, TS } Here, TS is not for the authentication of B to A, but for replay-proof of the message. This usage of TS is quite unusual because TS is usually included in OA_S -type authentication message to authenticate the message generator.		
$IA_{\varnothing}$ - $IA_{\varnothing}$	A: APriKey{ } B: BPriKey{ }	**ISO/IEC key agreement mech.1** [ISO96] or **Non-interactive Diffie-Hellman key agreement** A: APriKey{ BPubKey } B: BPriKey{ APubKey }	
$IA_{\varnothing}$ - IA_F (illegal type)	B BPriKey{ } A ← B r_B A APriKey{ r_B }		
$IA_{\varnothing}$ - $OA_{\varnothing}$	A → B: APriKey{ B } B: BPriKey{ }		

Prototype-Prototype	Example (Generic Protocols)	Notes	
$IA_{\varnothing}$ - OA_S	$A \rightarrow B$: TS_A , APriKey{ B, TS_A } B : BPriKey{ }		
$IA_{\varnothing}$ - OA_F (illegal type)	B BPriKey{ } $A \leftarrow B$ r_B $A \rightarrow B$ APriKey{ B, r_B }		
$IA_{\varnothing}$ - $DA_{\varnothing}$ (illegal type)	B BPriKey{ r_A } $A \leftarrow B$ APubKey{ A }		
$IA_{\varnothing}$ - $DA_{F, NoAck}$ (illegal type)	B BPriKey{ r_A } $A \leftarrow B$ APubKey{ A, r_B }		
$IA_{\varnothing}$ - $DA_{F, Ack}$ (illegal type)	B BPriKey{ r_A } $A \leftarrow B$ APubKey{ A, r_B } $A \rightarrow B$ r_B		
IA_F - $IA_{\varnothing}$	A : BPriKey{ } $A \rightarrow B$: r_A B : BPriKey{ r_A }		
IA_F - IA_F	$A \rightarrow B$: r_A B : BPriKey{ r_A } $A \leftarrow B$: r_B A : APriKey{ r_B }	**ISO/IEC KA mech. 4** or **Diffie-Hellman** key agreement (basic) protocol, where the basic assumption (*A, B*: each in possession of the public key of the other party) is not assumed $A \rightarrow B$: APubKeyX $A \leftarrow B$: BPubKeyX A : APriKeyX{ BPubKeyX } B : BPriKeyX{ APubKeyX }	$A \rightarrow B/E \setminus A - B$ (possible only for D-H basic protocol) $A \rightarrow B/E$: APubKeyX $A \leftarrow B/E$: EPubKeyX $E \setminus A \rightarrow B$: EPubKeyX $E \setminus A \leftarrow B$: BPubKeyX A : $SessKey_{AE}$ = APriKeyX{ EPubKeyX }
IA_F - $OA_{\varnothing}$	$A \rightarrow B$: r_A , APriKey{ B } B : BPriKey{ r_A }		
IA_F - OA_S	$A \rightarrow B$: r_A , TS_A , APriKey{ B, TS_A } B : BPriKey{ r_A }	**ISO/IEC KA mech.3** [ISO96] or **Nyberg-Rueppel** key agreement [NyRu94] : [IA_F - OA_S] + key confirmation to B $A \rightarrow B$: APubKeyX, TS_A , APriKey{ $SessKey_{AB}$ { A, TS_A } } B : BPriKey{ APubKeyX } (= F(BPriKey, APubKeyX) = F(APriKeyX, BPubKey*) = $SessKey_{AB}$) (* : This value has the same effect of the field B as in APriKey{ B, TS_A })	
IA_F - OA_F	$A \rightarrow B$: r_A B : BPriKey{ r_A } $A \leftarrow B$: r_B $A \rightarrow B$: APriKey{ B, r_B }		
IA_F - $DA_{\varnothing}$	$A \rightarrow B$: r_A B : BPriKey{ r_A } $A \leftarrow B$: APubKey{ A }		
IA_F - $DA_{F, NoAck}$	$A \rightarrow B$: r_A B : BPriKey{ r_A } $A \leftarrow B$: APubKey{ A, r_B }		
IA_F - $DA_{F, Ack}$	$A \rightarrow B$: r_A B : BPriKey{ r_A } $A \leftarrow B$: APubKey{ A, r_B } $A \rightarrow B$: r_B		
$OA_{\varnothing}$ - $IA_{\varnothing}$ (illegal type)	A APriKey{ } $A \leftarrow B$ BPriKey{ A }		

Prototype-Prototype	Example (Generic Protocols)	Notes	
$OA_{\varnothing}$ - IA_F (illegal type)	$A \leftarrow B$ r_B , BPriKey{ A } A APriKey{ r_B }		
$OA_{\varnothing}$ - $OA_{\varnothing}$	$A \rightarrow B$: APriKey{ B } $A \leftarrow B$: BPriKey{ A }		
$OA_{\varnothing}$ - OA_S	$A \rightarrow B$: TS_A , APriKey{ B, TS_A } $A \leftarrow B$: BPriKey{ A }		
$OA_{\varnothing}$ - OA_F (illegal type)	$A \leftarrow B$ r_B , BPriKey{ A } $A \rightarrow B$ APriKey{ B, r_B }		
$OA_{\varnothing}$ - $DA_{\varnothing}$ (illegal type)	$A \leftarrow B$ BPriKey{ A }, APubKey{ B }		
$OA_{\varnothing}$ - $DA_{F, NoAck}$ (illegal type)	$A \leftarrow B$ BPriKey{ A }, APubKey{ B, r_B }		
$OA_{\varnothing}$-$DA_{F, Ack}$ (illegal type)	$A \leftarrow B$ BPriKey{ A }, APubKey{ B, r_B } $A \rightarrow B$ r_B		
OA_S - $IA_{\varnothing}$ (illegal type)	$A \leftarrow B$ TS_B , BPriKey{ A, TS_B } A APriKey{ }		
OA_S - IA_F (illegal type)	$A \leftarrow B$ r_B , TS_B , BPriKey{ A, TS_B } A APriKey{ r_B }		
OA_S - $OA_{\varnothing}$	$A \rightarrow B$: APriKey{ A } $A \leftarrow B$: TS_B , BPriKey{ A, TS_B }		
OA_S - OA_S	$A \rightarrow B$: TS_A , APriKey{ B, TS_A } $A \leftarrow B$: TS_B , BPriKey{ A, TS_B }	**ISO/IEC Public Key Two-Pass Mutual Authentication Protocol** [ISO95] $A \rightarrow B$: ACert, TS_A, B, Text2, APriKey{ TS_A, B, Text1} $A \leftarrow B$: BCert, TS_B, A, Text4, BPriKey{ TS_B, A, Text3} **CCITT X.509 Protocol** (as described in [BAN90]) : This protocol has a very complex structure with multiple authentication prototypes for each direction: OA_S-OA_S, OA_F-OA_F, and $DA_{\varnothing}$-$DA_{\varnothing}$. Nevertheless, attacks have been found by L'Anson and Mitchell [IAMi90] and by the Burrows Abadi and Needham [BAN90]. $A \rightarrow B$: A, APriKey{ TS_A, r_A, B, X_A, BPubKey{Y_A} } $A \leftarrow B$: B, BPriKey{ TS_B, r_B, A, r_A, X_B, APubKey{Y_B} } $A \rightarrow B$: APriKey{ r_B }	
OA_S - OA_F (illegal type) OA_S - $DA_{\varnothing}$ (illegal type)	$A \leftarrow B$ r_B , TS_B , BPriKey{ A, TS_B } $A \rightarrow B$ APriKey{ B, r_B } $A \leftarrow B$ TS_B , BPriKey{ A, TS_B }, APubKey{ B }		
	$A \leftarrow B$ TS_B , BPriKey{ A, TS_B , APubKey{ B } }		
	$A \leftarrow B$ APubKey{ B, TS_B , BPriKey{ A, TS_B } }		
OA_S - $DA_{F. NoAck}$ (illegal type)	$A \leftarrow B$ TS_B , BPriKey{ A, TS_B }, APubKey{ B, r_B }		
	$A \leftarrow B$ TS_B , BPriKey{ A, TS_B , APubKey{ B, r_B } }		
	$A \leftarrow B$ APubKey{ B, r_B , TS_B , BPriKey{ A, TS_B } }		
OA_S - $DA_{F, A}$ (illegal type)	$A \leftarrow B$ TS_B , BPriKey{ A, TS_B }, APubKey{ B, r_B } $A \rightarrow B$: r_B		
	$A \leftarrow B$ TS_B , BPriKey{ A, TS_B , APubKey{ B, r_B } } $A \rightarrow B$ r_B		
	$A \leftarrow B$ APubKey{ B, r_B , TS_B , BPriKey{ A, TS_B } } $A \rightarrow B$ r_B		
OA_F - $IA_{\varnothing}$	$A \rightarrow B$: r_A $A \leftarrow B$: BPriKey{ A, r_A } A: APriKey{ }		

Prototype-Prototype	Example (Generic Protocols)	Notes	
OA_F - IA_F	$A \rightarrow B$: r_A $A \leftarrow B$: r_B , BPriKey{ A, r_A } A: APriKey{ r_B }		
OA_F - $OA_\varnothing$	$A \rightarrow B$: r_A , APriKey{ B } $A \leftarrow B$: BPriKey{ A, r_A }		
OA_F - OA_S	$A \rightarrow B$: r_A , TS_A , APriKey{ B, TS_A } $A \leftarrow B$: BPriKey{ A, r_A }	**Gray III's proposal** [Gray97] for SPLICE/AS protocol [YOM90] $A \rightarrow B$: A, APriKey{ B, TS_A, Lifetime, r_A } $A \leftarrow B$: B, BPriKey{ A, r_A }	
OA_F - OA_F	$A \rightarrow B$: r_A $A \leftarrow B$: BPriKey{ A, r_A } , r_B $A \rightarrow B$: APriKey{ B, r_B } (APriKey{ B, r_A , r_B } : a modified form to prevent the Canadian Attack)	**ISO/IEC KA mech. 7** [ISO96] or **STS protocol** [DvOW93] $A \rightarrow B$: APubKeyX $A \leftarrow B$: BPubKeyX, BPriKey{ BPubKeyX, APubKeyX, A }, $SessKey_{AB}${ BPubKeyX, APubKeyX, A }* $A \rightarrow B$: APriKey{ APubKeyX, BPubKeyX, B }, $SessKey_{AB}$ { APriKey{ APubKeyX, BPubKeyX, B } } * $SessKey_{AB}$ = F(APriKeyX, BPubKeyX) = F(BPriKeyX, APubKeyX) (* : Key confirmation) **ISO/IEC key transport mech. 5** [ISO96] This protocol is in fact a combination of 2 prototypes: **OA_F $DA_{F,\ NoAck}$ - OA_F $DA_{F,\ NoAck}$** $A \rightarrow B$: r_A $A \leftarrow B$: BPriKey{ r_B , r_A , A, APubKey{ B, K_B } } $A \rightarrow B$: APriKey{ r_A , r_B , B, BPubKey{ A, K_A } } $SessKey_{AB}$ = w(K_A , K_B) where w: common one-way function **ISO/IEC Three-Pass Mutual Authentication Protocol** [ISO95] $A \rightarrow B$: r_A, Text1 $A \leftarrow B$: BCert , r_B, r_A, A, Text3, BPriKey{ r_B, r_A, A, Text2 } $A \rightarrow B$: ACert , r_A, r_B, B, Text5, APriKey{ r_A, r_B, B, Text4 }	**ASPeCT Siemens protocol A** [ASPe96] $A \rightarrow B$: APubKeyX $A \leftarrow B$: r_B , BPriKey{ APubKeyX, r_B }, $SessKey_{AB}${ data1 } $A \rightarrow B$: $SessKey_{AB}$ { APriKey{ APubKeyX, r_B , BPubKey } }, $SessKey_{AB}$ { A }, $SessKey_{AB}$ { data2 } A : $SessKey_{AB}$ = APriKeyX{ BPubKey, r_B } B : $SessKey_{AB}$ = BPriKey{ APubKeyX, r_B } (Here A: UIM, B: NP) **ASPeCT KPN protocol** [ASPe96] $A \rightarrow B$: APubKeyX, CS $A \leftarrow B$: BPubKeyX, BCert, BPriKey{ APubKeyX, BPubKeyX } $A \rightarrow B$: $SessKey_{AB}$ { ACert, APriKey{ BPubKeyX, APubKeyX } } $SessKey_{AB}$ = APriKeyX{ BPubKeyX } = BPriKeyX{ APubKeyX } = $g^{APriKeyX \cdot BPriKeyX}$ (Here A: UIM, B: NO) This protocol misses the identity field A in the 2nd message, and may be easily attacked by impersonation attack in the middle ($A \rightarrow E/E \backslash A - B$).
OA_F - $DA_\varnothing$	$A \rightarrow B$: r_A $A \leftarrow B$: BPriKey{ A, r_A }, APubKey{ B }		
	$A \rightarrow B$: r_A $A \leftarrow B$: BPriKey{ A, r_A , APubKey{ B }}		
	$A \rightarrow B$: r_A $A \leftarrow B$: APubKey{ B, BPriKey{ A, r_A } }		
OA_F - $DA_{F,\ NoAck}$	$A \rightarrow B$: r_A $A \leftarrow B$: BPriKey{ A, r_A } , APubKey{ B, r_B } $A \rightarrow B$: r_A $A \leftarrow B$: BPriKey{ A, r_A , APubKey{ B, r_B } }	**ISO/IEC key transport mech. 4** [ISO96] $A \rightarrow B$: r_A $A \leftarrow B$: BPriKey{ A, r_A , r_B *, APubKey{ B, $SessKey_{AB}$ } } Here $SessKey_{AB}$'s role is a challenge from B to A when viewed in terms of authentication. * : Here, r_B is optional according to the protocol description of ISO/IEC doc., but the purpose of this field is very vague. It should be included in the APubKey{ ... } field if it plays any role in security. This mistake shows a kind of appalling immaturity of the area of security protocol design.)	
	$A \rightarrow B$: r_A $A \leftarrow B$: APubKey{ B, r_B , BPriKey{ A, r_A } }	**ISO/IEC KA mech. 6** [ISO96] or **Beller-Yacobi's** two-pass protocol [BeYa93] $A \rightarrow B$: r_A $A \leftarrow B$: APubKey{ B, BPriKey{ A, r_A , r_B } } $SessKey_{AB}$ = BPriKey{ A, r_A , r_B }	
OA_F -$DA_{F,\ Ack}$	$A \rightarrow B$: r_A $A \leftarrow B$: BPriKey{ A, r_A } , APubKey{ B, r_B } $A \rightarrow B$: r_B		
	$A \rightarrow B$: r_A $A \leftarrow B$: BPriKey{ A, r_A , APubKey{ B, r_B } } $A \rightarrow B$: r_B		

Prototype-Prototype	Example (Generic Protocols)	Notes	
OA_F-$DA_{F,\ ACK}$ (Cont'd)	$A \rightarrow B$: r_A $A \leftarrow B$: $APubKey\{ B, r_B, BPriKey\{ A, r_A \} \}$ $A \rightarrow B$: r_B		
$DA_{\varnothing}$- $IA_{\varnothing}$	$A \rightarrow B$: $BPubKey\{ A \}$ A: $APriKey\{ \}$		
$DA_{\varnothing}$- IA_F	$A \rightarrow B$: $BPubKey\{ A \}$ $A \leftarrow B$: r_B A: $APriKey\{ r_B \}$		
$DA_{\varnothing}$- $OA_{\varnothing}$	$A \rightarrow B$: $BPubKey\{ A \}, APriKey\{ B \}$		
$DA_{\varnothing}$- OA_S	$A \rightarrow B$: $BPubKey\{ A \}, TS_A, APriKey\{ B, TS_A \}$		
	$A \rightarrow B$: $TS_A, APriKey\{ B, TS_A, BPubKey\{ A \} \}$		
	$A \rightarrow B$: $BPubKey\{ A, TS_A, APriKey\{ B, TS_A \} \}$		
$DA_{\varnothing}$- OA_F	$A \rightarrow B$: $BPubKey\{ A \}$ $A \leftarrow B$: r_B $A \rightarrow B$: $APriKey\{ B, r_B \}$		
$DA_{\varnothing}$- $DA_{\varnothing}$	$A \rightarrow B$: $BPubKey\{ A \}$ $A \leftarrow B$: $APubKey\{ B \}$		
$DA_{\varnothing}$- $DA_{F,\ NoAck}$	$A \rightarrow B$: $BPubKey\{ A \}$ $A \leftarrow B$: $APubKey\{ B, r_B \}$		
$DA_{\varnothing}$-$DA_{F,\ Ack}$	$A \rightarrow B$: $BPubKey\{ A \}$ $A \leftarrow B$: $APubKey\{ B, r_B \}$ $A \rightarrow B$: r_B		
$DA_{F,\ NoAck}$ - $IA_{\varnothing}$	A: $APriKey\{ \}$ $A \rightarrow B$: $BPubKey\{ A, r_A \}$		
$DA_{F,\ NoAck}$ - IA_F	$A \rightarrow B$: $BPubKey\{ A, r_A \}$ $A \leftarrow B$: r_B A: $APriKey\{ r_B \}$		
$DA_{F,NoAck}$ - $OA_{\varnothing}$	$A \rightarrow B$: $BPubKey\{ A, r_A \}, APriKey\{ B \}$		
	$A \rightarrow B$: $APriKey\{ B, BPubKey\{ A, r_A \} \}$		
	$A \rightarrow B$: $BPubKey\{ A, r_A, APriKey\{ B \} \}$		
$DA_{F,\ NoAck}$ - OA_S	$A \rightarrow B$: $BPubKey\{ A, r_A \}, TS_A, APriKey\{ B, TS_A \}$		
	$A \rightarrow B$: TS_A, $APriKey\{ B, TS_A, BPubKey\{ A, r_A \} \}$	**ISO/IEC key transport mech. 2** [ISO96] $A \rightarrow B$: $APriKey\{B, TS, BPubKey\{ A, SessKey_{AB} \} \}$ application of this prototype: **$DA_{F,\ NoAck}$ OA_S - OA_S $DA_{F,\ NoAck}$**	
	$A \rightarrow B$: $BPubKey\{ A, r_A, TS_A, APriKey\{ B, TS_A \} \}$	**ISO/IEC key transport mech. 3** [ISO96] $A \rightarrow B$: $BPubKey\{ APriKey\{ B, SessKey_{AB}, TS \} \}$ **Denning Sacco public key protocol** [DeSa81] $A \rightarrow B$: $BPubKey\{ APriKey\{ K_{AB}, TS_A \} \}$	**WACS/PACS** public AKA protocol [Bell94], [JTC94] $A \rightarrow B$: $BPubKey\{ A, TS, APriKey\{ B, A, TS \}^* \}, SessKey_{AB}\{ ACert, APubKey \}$ (* : This value is used as $SessKey_{AB}$)
$DA_{F,\ NoAck}$ - OA_F	$A \rightarrow B$: $BPubKey\{ A, r_A \}$ $A \leftarrow B$: r_B $A \rightarrow B$: $APriKey\{ B, r_B \}$		

Prototype-Prototype	Example (Generic Protocols)	Notes	
$DA_{F, NoAck}$ - $DA_{\varnothing}$	$A \rightarrow B$: BPubKey{ A, r_A } $A \leftarrow B$: APubKey{ B }		
$DA_{F, NoAck}$ - $DA_{F, NoAck}$	$A \rightarrow B$: BPubKey{ A, r_A } $A \leftarrow B$: APubKey{ B, r_B }	**SKEME protocol** proposed by Krawczyk [Kraw96] with the following difference $A \rightarrow B$: BPubKey{ r_A } $A \leftarrow B$: APubKey{ r_B } (A, B missed in this protocol, and enables several partial success attack possible) **ISO/IEC key transport mech. 5** [ISO96] This protocol is in fact a combination of 2 prototypes: **$OA_F DA_{F, NoAck}$ - $OA_F DA_{F, NoAck}$** $A \rightarrow B$: r_A $A \leftarrow B$: BPriKey{ r_B , r_A , A, APubKey{ B, K_B } } $A \rightarrow B$: APriKey{ r_A , r_B , B, BPubKey{ A, K_A } } $SessKey_{AB}$ = w(K_A , K_B)	$A \rightarrow E/E \backslash A - B$ attack for SKEME protocol $A \rightarrow E/E$: EPubKey{ r_A } $E \backslash A \rightarrow B$: BPubKey{ r_A } $A \leftarrow E/E \backslash A \leftarrow B$: APubKey{ r_B } ▫ Result: A and B have in fact the same session key, but the key is understood as $SessKey_{AE}$ by A, and $SessKey_{AB}$ by B. As for E, he knows rA but not rB ▫ This kind of attack can be named as a "credit theft" attack because E can intercept a message from B to A and resend it to A by the name of E himself. The result is that A believes that the message was made by E with SessKeyAE . Of course, E cannot know the content of the message. ▫ $A \rightarrow B/E \backslash E - B$ attack also have the same symmetrical effect. The following attacks are also possible with partial success. $A \rightarrow B/E \backslash E - B$ (*A* initatiates the protocol) $A - E/E \backslash A \rightarrow B$ (*E* initatiates the protocol with *B* and uses *A* as an oracle) $A - B/E$
$DA_{F, NoAck}$ - $DA_{F, Ack}$	$A \rightarrow B$: BPubKey{ A, r_A } $A \leftarrow B$: APubKey{ B, r_B } $A \rightarrow B$: r_B		
$DA_{F, Ack}$ - $IA_{\varnothing}$	A: APriKey{ } $A \rightarrow B$: BPubKey{ A, r_A } $A \leftarrow B$: r_A		
$DA_{F, Ack}$ - IA_F	$A \rightarrow B$: BPubKey{ A, r_A } $A \leftarrow B$: r_A , r_B A: APriKey{ r_B }		
$DA_{F, Ack}$-$OA_{\varnothing}$	$A \rightarrow B$: BPubKey{ A, r_A } , APriKey{ B } $A \leftarrow B$: r_A		
	$A \rightarrow B$: APriKey{ B, BPubKey{ A, r_A } } $A \leftarrow B$: r_A		
	$A \rightarrow B$: BPubKey{ A, r_A , APriKey{ B } } $A \leftarrow B$: r_A		
$DA_{F, Ack}$-OA_S	$A \rightarrow B$: BPubKey{ A, r_A } , TS_A , APriKey{ B, TS_A } $A \leftarrow B$: r_A		
	$A \rightarrow B$: TS_A , APriKey{ B, TS_A , BPubKey{ A, r_A } } $A \leftarrow B$: r_A	**SPLICE/AS protocol** [YOM90] $A \rightarrow B$: A, B, APriKey{ A, TS_A , Lifetime, BPubKey{ r_A } } $A \leftarrow B$: B, A, APubKey{ B, r_A+1}	Weakness in SPLICE/AS protocol In the first message, the COM BPubKey{ r_A } lacks the identity A, hence not providing VPA to B. Clark and Jacob presented an attack exploiting the weakness and proposed a modification [ClJa95].
	$A \rightarrow B$: BPubKey{ A, r_A , TS_A, APriKey{ B, TS_A } } $A \leftarrow B$: r_A		

Prototype-Prototype	Example (Generic Protocols)	Notes	
$DA_{F, Ack}$-OA_{F}	$A \rightarrow B$: BPubKey{ A, r_A } $A \leftarrow B$: r_A , r_B $A \rightarrow B$: APriKey{ B, r_B }	the **WAKE protocol** in this thesis $A \rightarrow B$: BPubKey{ r_A } $A \leftarrow B$: r_B, $SessKey_{AB}${ r_A } $A \rightarrow B$: APriKey{ B, r_B , r_A }, $SessKey_{AB}${A}	**SSL/TLS protocol** [RFC99], [Paul99] $A \rightarrow B$: A, r_A $A \leftarrow B$: r_B, BCert $A \rightarrow B$: ACert, BPubKey{ r_A' }, APriKey{ B, r_B , r_A }, $SessKey_{AB}${finished} $A \leftarrow B$: $SessKey_{AB}${finished}
$DA_{F, Ack}$-$DA_{\varnothing}$	$A \rightarrow B$: BPubKey{ A, r_A } $A \leftarrow B$: r_A , APubKey{ B }		
$DA_{F, Ack}$- $DA_{F, NoAck}$	$A \rightarrow B$: BPubKey{ A, r_A } $A \leftarrow B$: r_A , APubKey{ B, r_B }		
$DA_{F, Ack}$- $DA_{F, Ack}$	$A \rightarrow B$: BPubKey{ A, r_A } $A \leftarrow B$: r_A , APubKey{ B, r_B } (APubKey{ B, r_B , r_A} : modified form to prevent the Canadian Attack. See the note on the right.) $A \rightarrow B$: r_B	**Needham-Schroeder public-key protocol** [NeSc78]: r_A , r_B used here as inputs for session key generation function $SessKey_{AB} = f(r_A, r_B)$, and hece the following variation: $A \leftarrow B$: APubKey{r_A,r_B} $A \rightarrow B$: BPubKey{r_B} Attacks: $A - E\backslash A - B$, $A - B/E - B$ all possible for both the prototype and N-S protocols Result: ▫ N-S: E: perfect impersonation and disclosed r_A , r_B **ISO/IEC key transport mech. 6** [ISO96] or **COMSET protocol** [BoPr95] $A \rightarrow B$: BPubKey{ A, r_A, K_A } $A \leftarrow B$: APubKey{ r_A , r_B , K_B } $A \rightarrow B$: r_B (Here, B is missing in the second msg. Designers are aware of this weakness.) **Helsinki** Protocol [HoHs98] $A \rightarrow B$: BPubKey{ A, K_A , r_A } $A \leftarrow B$: APubKey{ K_B, r_A , r_B } $A \rightarrow B$: r_B	◆ $A \rightarrow E/E\backslash A - B$ attack for N-S protocol $A \rightarrow E$: EPubKey {r_A, A } $E\backslash A \rightarrow B$: BPubKey{r_A , A} $A \leftarrow E/E\backslash A \leftarrow B$: r_A , APubKey{ r_B } $A \rightarrow E/E\backslash A \rightarrow B$: r_B Fix: $A \leftarrow B$: r_A , APubKey{**B** , r_B } ◆ $A \rightarrow B/E\backslash E - B$ attack for a modified protocol $A \rightarrow B/E\backslash E \rightarrow B$: BPubKey{ r_A} $E\backslash E \leftarrow B$: r_A , EPubKey{r_B} $A \leftarrow B/E$: r_A , APubKey{r_B } $A \rightarrow B/E\backslash E \rightarrow B$: r_B Fix: $A \rightarrow B$: B PubKey{**A** , r_A } ◆ $A \rightarrow E/E\backslash A - B$ attack for COMSET $A \rightarrow E/E$: EPubKey{ A, r_A, K_A } $E\backslash A \rightarrow B$: BPubKey{ A, r_A , K_A } $A \leftarrow E/E\backslash A \leftarrow B$: APubKey{ r_A , r_B , K_B } $A \rightarrow E/E\backslash A \rightarrow B$: r_B Fix : $A \leftarrow B$: APubKey{ **B**, r_A, r_B , K_B } ◆ Canadian Attack for prototype ($A \leftarrow B/E\backslash A \rightarrow B$) $B/E\backslash A \rightarrow B$: BPubKey{ A, r_A } $B/E\backslash A \leftarrow B$: r_A , APubKey{ B, r_B } $A \leftarrow B/E\backslash A$: APubKey{ B, r_B } $A \rightarrow B/E\backslash A$: r_B, BPubKey{ A, r_A'} $B/E\backslash A \rightarrow B$: r_B (Result: E know r_A and r_B. B belives tha he is communicating with A. A-$B/E\backslash A$ link fails)