Authentication Protocols for Mobile Network Environment Value-added Services

Günther Horn, Keith M. Martin and Chris J. Mitchell

Abstract— **The secure provision of mobile computing and telecommunication services is rapidly increasing in importance as both demand and applications for such services continue to grow. This paper is concerned with the design of public key based protocols suitable for application in upcoming third generation mobile systems such as UMTS. Candidate protocols are considered for the authentication of a mobile user to a value-added service provider with initialisation of a mechanism enabling payment for the value-added service. A set of goals for such a protocol are identified, as are a number of generic attacks; these goals and attacks are then used to evaluate the suitability of seven candidate third generation user-to-network authentication protocols. Many of these candidate protocols are shown to have highly undesirable features.**

I. Introduction

MOBILE computing and telecommunications are currently areas of rapid growth as the technology necessary to widely implement relevant services is becoming increasingly available. The advantages of wireless communications are likely to see these technologies featuring in upcoming third generation mobile systems such as UMTS, and offering many new services that will revolutionise the ways society handles information. In particular, these technologies are almost certain to find use in environments where a (roaming) mobile user purchases *value-added services* (VASs) from a *value-added service provider* (VASP). This paper evaluates a number of candidate third generation security protocols that mutually authenticate a mobile user and VASP and initialise an appropriate payment mechanism.

The above mobile scenario places a number of restrictions on the design of suitable security protocols. Most significantly, a mobile user will typically have limited computational capabilities compared to an entity in a fixed network. Thus the necessary computational effort at the user's end of any protocol should be minimised. It is worth noting that this constraint is likely to erode with time as mobile devices become more powerful. It is anticipated that in the future the more significant performance constraint may arise at the server (VASP) end, which could become a bottleneck during multiple service requests. For this reason we also pay attention to the computational effort required by the VASP, seeking to keep this as low as possible. Moreover, the communications bandwidth between user and VASP may be limited, and hence protocol messages should be kept as short as possible. The candidate protocols considered mostly use public key techniques, which are generally more computationally intensive than symmetric cryptography. However use of public key cryptography has significant advantages within large and complex communication networks, including scalability, easier key management and the lack of need for online authentication servers.

It is highly desirable that a user-to-VASP authentication and payment initialisation (API) process is closely based on the type of user-to-network authentication process that is likely to be used for system set-up procedures (for example user-to-network call set-up procedures in a mobile telecommunications network). This allows efficient integration of the two processes. For this reason we evaluate both user-to-network authentication protocols (which can be extended to API) and explicit API protocols. In the following sections we identify formal goals of an API protocol and consider a number of possible protocol attacks. We then consider seven candidate API protocols and evaluate them against the API goals, showing that several have fundamental weaknesses. We conclude with a general comparison of the seven candidates.

II. API protocol goals

An API protocol begins with a mobile user contacting a VASP for the first time over the vulnerable 'air interface' of a typical mobile telecommunications network. Each entity has a certified copy of its own public key but needs a certified copy of the other's public key. Both entities need to authenticate one another, establish a common cryptographic key, and initiate a payment scheme. The precise protocol goals are as follows (we refer the reader to Menezes et al. [24] for formal definitions of the italicised terms):

1. mutual *entity authentication* of user and VASP

G. Horn is with Siemens AG, Corporate Technology, D-81730, München, Germany (e-mail: guenther.horn@mchp.siemens.de)

K.M. Martin was with K.U.Leuven, ESAT/COSIC, Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium, and was supported by the European Commission under ACTS project AC095 (ASPeCT). He is now with the Information Security Group, Royal Holloway, University of London, Egham, Surrey TW20 0EX, U.K. (e-mail:keith.martin@rhul.ac.uk)

C.J. Mitchell is with the Information Security Group, Royal Holloway, University of London, Egham, Surrey TW20 0EX, U.K. (e-mail: C.Mitchell@rhul.ac.uk)

(to protect against masquerade of either entity over the air interface);
2. exchange of certified public keys between user and VASP (to support the authentication and key establishment processes);
3. mutual *key agreement* of a session key between user and VASP (used to protect data subsequently exchanged between user and VASP);
4. joint *key control* of the session key (to prevent either party accidentally or deliberately choosing a weakened key);
5. mutual *implicit key authentication* (to ensure that no other party can obtain the established session key);
6. mutual *key confirmation* (so that both user and VASP have assurance that they both possess the same shared session key);
7. mutual assurance of *key freshness* (to prevent replays of old messages being used to re-establish an 'old', possibly compromised, session key);
8. confidentiality of the user identity over the air interface (to prevent an interceptor of air interface communications learning the mobile user's identity, and/or being able to track particular mobile users, see Mitchell and Chen [25]);
9. initialisation of the payment mechanism (to support payment for VASs);
10. *non-repudiation* of the payment initialisation data (to support payment for VASs).

Goals 1 to 8 are suitable goals for a user-to-network authentication protocol. It is not the intention of this paper to discuss payment mechanisms for VASs in mobile computing networks; see Horn and Preneel [16] and Martin et al. [23] for a more detailed discussion. We assume that the API protocol initiates a suitable micropayment scheme such as those based on one-way hash chains proposed in Pedersen [29] and Rivest and Shamir [30]. Such schemes typically involve the VASP proposing some charging related data to the user, and the user committing to this data and some initialisation values using a digital signature. Throughout this paper we assume the use of digital signatures with appendix. We note that the use of digital signatures with message recovery (see [24] for an explanation of the differences) may further simplify some of the protocols.

III. Protocol attacks

In this section we describe a number of possible attacks against API protocols that feature in the proceeding discussion.

1. *Signer verification attack.* User anonymity can be compromised if an attacker can obtain a digital signature generated by the user. If the data was signed using a signature system giving (partial) message recovery, such as ISO/IEC 9796-2 signatures [17], then an attacker who has access to a large set of public verification keys applies them successively to the signature; if the verification process fails then the attacker can assume the owner of that key is not the signer. A similar attack applies to other signature systems if the attacker also has access to the (hashed) data that was signed.
2. *Content verification attack.* This attack is similar in principle to a signer verification attack and can be used to determine a piece of data in a signature that is unknown to the attacker (but which comes from a limited set of possible values). Assume that the attacker has a cleartext signature, and knows both the signer and most of the data that was signed. The attacker repeatedly guesses the missing data and attempts to verify the signature with the known user's verification key.
3. *Source substitution attack.* A source substitution attack as defined in Diffie et al. [11] involves an attacker taking another entity's public key and managing to obtain a certificate in the name of the attacker for that public key value. This then allows the attacker to masquerade as the other entity in a number of situations, such as when claiming to be the signer of data. Although such attacks can be prevented if a certification authority insists on proof of knowledge of the corresponding private key before issuing a certificate, it is wise to avoid exposure to such attacks.
4. *Time-memory tradeoff attack.* A time-memory tradeoff attack can be used to determine data for which a hashed version is available. Suppose that data K contains k bits, and $h(K)$ has been observed. The attacker pre-computes and stores 2^r values of $h(K)$. When the value $h(K)$ is observed during the protocol run, the attacker compares this value with the pre-computed values. The probability of success is 2^{k-r} and thus the attacker will need to intercept 2^{k-r} such values for every successful capture of K. Other versions of this type of attack have been used in different crptanalytic contexts, see for example Borst et al. [6] and Hellmann [15].
5. *Codebook attack.* In a codebook attack an attacker keeps a record of data that is encrypted using the same symmetric key. Even if the plaintext is not known, if the same data is ever encrypted a second time with the same key then the attacker can identify this from the stored record.
6. *Partial chosen key attack.* This type of attack was described in Mitchell et al. [26] and is relevant to protocols claimed to offer joint key control. Suppose two entities X and Y both contribute random inputs to the computation of a session key $f(x, y)$, but that X first sends x to Y. It is possible for Y to then compute 2^s variants of y and the corresponding $f(x, y)$ before sending y to X. In this way entity Y can "select" approximately s bits of the joint key. The value of s, and hence the degree of imbalance in the key control, is constrained by the computing resources and time available for Y to complete this process. Note that most public

key based key agreement protocols currently specified in ISO standards, for example ISO/IEC 11770-3 [18], are potentially vulnerable to partial chosen key attacks.

7. *Key separation attack.* Suppose entity A encrypts something during the authentication protocol (with entity B) using a symmetric key K. Now suppose that as part of a completely different protocol between entity E and entity A, entity E sends data to A that A encrypts with the same key K and returns to E. It is possible that entity E could exploit this lack of key separation to replace messages sent during the first protocol (between A and B) with messages manufactured during the second (by sending chosen messages to A for encryption using K). Protocols are particularly vulnerable to a key separation attack if their final message (from A to B say) is encrypted. In this case it may be possible for E to replace the message sent on to B while A believes that the protocol has executed correctly. Note that this kind of attack is avoided if, during any subsequent protocol that requires symmetric encryption with key K, key K is replaced by a key derived from K (by applying a pseudo-random function, for example) when used as a session key under natural assumptions on the encryption scheme. In this way the key used during authentication is separated from the key later used for data encryption. Other solutions to thwart the attack are possible.

8. *Known key attacks.* This attack applies if, in the event that an old session key is compromised, future session keys can also be compromised. Known key attacks are related to those discussed by Denning and Sacco [9] with respect to the use of timestamps to prevent message replay. Such attacks are of particular concern in any environment where the probability of compromise of session keys is significantly greater than that of long-term keys (see Menezes et al. [24] for further discussion).

IV. Seven user-to-network protocols

We now consider seven third generation candidate API protocols. We indicate any minor modifications to an established authentication protocol that we have added to make it suitable for API. Note that the seven selected protocols were chosen because they are either established authentication protocols worth evaluating for our API scenario (STS and Aziz-Diffie) or recently proposed protocols of which we are unaware of published weaknesses. Several other candidate authentication protocols such as those recently proposed by Park [28] and Yi et al. [32] have already been found to have significant weaknesses, see Boyd and Park [7] and Martin and Mitchell [22]. A number of the protocols involve the use of a *Trusted Third Party* (TTP). The following general notation is used throughout the descriptions (the reader is referred to Menezes et al. [24] for further explanation of the basic cryptographic primitives referred to below):

g:	a generator of a multiplicative group in which discrete logarithms are hard;
ID_U:	an identifier of the user;
ID_V:	an identifier of the VASP;
ID_T:	an identifier of a TTP;
u:	private key of the user;
v:	private key of the VASP;
r_U:	a random nonce generated by the user;
r_V:	a random nonce generated by the VASP;
TS:	a timestamp generated by the VASP;
K:	a session key established between the user and the VASP;
$\mathrm{E}_K(x)$:	the symmetric encryption of x using key K;
$\mathrm{E}_p(x)$:	the asymmetric encryption of x using public key p;
$h(x)$:	the result of applying a one-way hash function h to input x;
$\mathrm{Sig}_\mathrm{T}(x)$:	the value x signed by the TTP;
$\mathrm{Sig}_\mathrm{U}(x)$:	the value x signed by the user;
$\mathrm{Sig}_\mathrm{V}(x)$:	the value x signed by the VASP;
cd:	charging related data;
py:	payment initialisation data.

Our descriptions of the seven protocols are necessarily brief and we refer the interested reader to the original sources for fuller discussions of design and implementation issues.

A. The STS protocol

The Station-to-Station (STS) protocol proposed by Diffie et al. [11] is a three pass Diffie-Hellman variant that establishes a shared session key between two parties with mutual entity authentication and mutual explicit key authentication. Although not explicitly designed for a mobile scenario, we consider it here because it meets almost all protocol goals (with the main exception of those relating to initialisation of the payment scheme). The version we describe in Figure 1 is based on the STS protocol described in Menezes et al. [24], with the additional exchange of (encrypted) certified public signature keys. The following additional notation is needed:

CertU:	certified public signature key of the user;
CertV:	certified public signature key of the VASP.

$$\begin{array}{rll}
\mathbf{U}\rightarrow\mathbf{V}: & g^{r_\mathrm{U}} & (1)\\
\mathbf{V}: & K=(g^{r_\mathrm{V}})^{r_\mathrm{U}} & \\
\mathbf{U}\leftarrow\mathbf{V}: & g^{r_\mathrm{V}},\ \mathrm{E}_K\{\mathrm{Sig}_\mathrm{V}(g^{r_\mathrm{V}},g^{r_\mathrm{U}})\},\mathrm{CertV} & (2)\\
\mathbf{U}: & K=(g^{r_\mathrm{U}})^{r_\mathrm{V}} & \\
\mathbf{U}\rightarrow\mathbf{V}: & \mathrm{E}_K\{\mathrm{Sig}_\mathrm{U}(g^{r_\mathrm{U}},g^{r_\mathrm{V}}),\ \mathrm{CertU}\} & (3)
\end{array}$$

Fig. 1. The STS protocol

The user generates a temporary public key and transfers it to the VASP in (1). The VASP generates a temporary private key and can now compute the session key $K = g^{r_\mathrm{U} r_\mathrm{V}}$. The VASP replies in (2) with its temporary public key, its certified public signature key, and an encrypted signed copy of both temporary public keys. After verifying the VASP's signature, the user is now able to compute K and acknowledges this by sending the encrypted signed temporary public keys in (3). CertU is included in the encrypted part of (3) to preserve user anonymity over the air interface. The VASP verifies the user's signature to conclude the protocol.

The STS protocol appears to meet goals 1-8, and has the advantage that goals 9 and 10 can be met by simply including the payment scheme initialisation data in the signature of the third message. Blake-Wilson and Menezes [5] observe that certain other variants of the STS protocol may not satisfy goal 5, and provide some advice on implementation of the STS protocol as described in Figure 1. The main disadvantage with the STS protocol is however the level of computational effort at both the user and VASP ends of the protocol. Partial chosen key attacks and key separation attacks are technically possible (but see Section III).

B. The Aziz-Diffie protocol

The Aziz-Diffie protocol proposed by Aziz and Diffie [3] is illustrated in Figure 2. We need the following additional notation:

CertU:	certified public verification key of user;
CertV:	certified public verification key of VASP;
r'_U:	a random nonce generated by the user;
p_U:	public key of the user;
p_V:	public key of the VASP.

$$\begin{array}{rll}
\mathbf{U} \rightarrow \mathbf{V}: & \mathrm{CertU},\ r_\mathrm{U} & (1)\\
\mathbf{U} \leftarrow \mathbf{V}: & \left\{\begin{array}{l}\mathrm{E}_{p_\mathrm{U}}\{r_\mathrm{V}\},\ \mathrm{Cert\ V}\\ \mathrm{Sig}_\mathrm{V}(\mathrm{E}_{p_\mathrm{U}}\{r_\mathrm{V}\}, r_\mathrm{U})\end{array}\right. & (2)\\
\mathbf{U} \rightarrow \mathbf{V}: & \left\{\begin{array}{l}\mathrm{E}_{p_\mathrm{V}}\{r'_\mathrm{U}\}\\ \mathrm{Sig}_\mathrm{U}(\mathrm{E}_{p_\mathrm{V}}\{r'_\mathrm{U}\},\ \mathrm{E}_{p_\mathrm{U}}\{r_\mathrm{V}\})\end{array}\right. & (3)\\
\mathbf{U},\mathbf{V}: & K = r'_\mathrm{U} + r_\mathrm{V} &
\end{array}$$

Fig. 2. The Aziz-Diffie protocol

The Aziz-Diffie protocol is distinctive in that the common key K is not computed using a Diffie-Hellman variant. The protocol begins with the user generating a random nonce and sending it and the user's certificate to the VASP in (1). The VASP generates a nonce, encrypts it using the user's public key and then sends this to the user in (2), along with a signature on both nonces and the VASP's certificate. The user decrypts and verifies the contents of (2). The user then generates a second nonce r'_U and forms the key K by adding this to the VASP's nonce. Finally the user encrypts this second nonce with the VASP's public key and sends this to the VASP in (3), including a signed copy of both encrypted nonces. The VASP decrypts and verifies the contents of (3) and is now also able to compute K.

The protocol appears to meet goals 1–5 and goal 7. It can be easily modified to incorporate the last two goals since the user already signs data in the third message. The protocol does not offer key confirmation in either direction. More seriously it does not offer user confidentiality over the air interface because CertU is sent in clear in the first message. The obvious solution, i.e. to encrypt this message with p_U, does not guarantee confidentiality as the signature in the third message is vulnerable to a signer verification attack.

C. The revised BCY protocol

The BCY protocol was first proposed by Beller et al. [4] and was subsequently improved, first by Carlsen [8] and then by Mu and Varadharajan [27]. We refer to the protocol in [27] as the Revised BCY protocol and illustrate it in Figure 3. We need the following additional notation:

N_V:	the product of two large primes;
CertU*:	special user certificate;
CertV*:	special VASP certificate;
KK:	a key encrypting key established between the user and the VASP.

The value N_V is the public key of V in the Modular Square Root (MSR) public key system, see Williams [31], where the private key is represented by the corresponding factorisation. Encryption of x is performed by taking the square of x modulo the public key, and decryption is performed by taking square roots. In the Revised BCY protocol the MSR system is used for creating certificates, and the Diffie-Hellman technique is used to establish the common key. In particular CertU* is formed by a certificate authority signing the user's public Diffie-Hellman key g^u using the MSR system, and CertV* is formed by a certificate authority signing the VASP's public Diffie-Hellman key g^v together with the VASP's public MSR key N_V using the MSR system. Both these certificates must be bound to their owners by the inclusion of the appropriate identifier [27].

The Revised BCY protocol begins with the VASP generating a random nonce and sending this and the special VASP certificate to the user in (1). The user now generates a random nonce, encrypts it using MSR and sends the result to the VASP in (2). The user also sends a message in (2), which includes its public Diffie-Hellman key and special certificate, all encrypted using the user's random nonce. The

$$\begin{array}{rll}
\mathbf{U} \leftarrow \mathbf{V}: & r_{\mathrm{V}},\ \mathrm{CertV^*} & (1)\\
\mathbf{U}: & x = r_{\mathrm{U}}^2 \pmod{N_{\mathrm{V}}} & \\
\mathbf{U}: & KK = (g^v)^u & \\
\mathbf{U} \rightarrow \mathbf{V}: & x,\ \mathrm{E}_{r_{\mathrm{U}}}(r_{\mathrm{V}}, \mathrm{ID_U}, g^u,\ \mathrm{CertU^*}) & (2)\\
\mathbf{V}: & r_{\mathrm{U}} = \sqrt{x} \pmod{N_{\mathrm{V}}} & \\
\mathbf{V}: & KK = (g^u)^v & \\
\mathbf{U},\mathbf{V}: & K = \mathrm{E}_{KK}(r_{\mathrm{U}}) &
\end{array}$$

Fig. 3. The Revised BCY protocol

VASP now decrypts the first part of (2) using MSR to obtain the user's nonce. Both entities now extract the other's public Diffie-Hellman key from the corresponding special certificate and compute KK. They each then use the result to encrypt the user's nonce in order to compute K.

It would seem that an extension of the Revised BCY protocol to meet the last two goals of Section II would involve at least one extra signature at the user end and one extra verification at the VASP end. Unfortunately, the Revised BCY protocol meets very few of our protocol goals. There is no entity authentication in either direction, since anyone can play the roles of either user or VASP without revealing this to the other entity. Neither is there mutual implicit key authentication, since (for example) the VASP cannot be sure that only the user knows r_{U}, nor any level of key confirmation. In fact only goals 2, 3 and 8 from the list in Section II appear to be achieved.

The reason for the disappointing performance of the Revised BCY protocol against goals 1–8 is that, although designed for a similar application, it can be deduced from [4] that the envisaged threat scenario is quite different. The protocol emphasis is on key agreement and identity and location privacy rather than authentication. The protocol thwarts passive attackers on the air interface, but permanent insiders, who can tap information in real time, are not regarded as a threat. This would seem rather unrealistic since fraudulent base-stations for certain mobile systems have been available for some time.

D. The Aydos-Sunar-Koç protocol

Aydos, Sunar and Koç [2] describe a protocol (the ASK protocol) providing 'authentication and key agreement' for a wireless environment. Their protocol uses elliptic-curve techniques for key agreement. For uniformity of presentation we have translated the ASK scheme into the more normal Diffie-Hellman notation, where we represent the group multiplicatively.

The ASK Protocol is as described in Figure 4. The protocol specification uses the following additional notation:

p_{U}:	public key agreement key of user ($= g^u$, where u is user's private key);
p_{V}:	public key agreement key of VASP ($= g^v$, where v is VASP's private key);
CertU:	certified public key agreement key (p_{U}) of the user;
CertV:	certified public key agreement key (p_{V}) of the VASP.

A shared key K^* is established and used in the protocol; this key will always be the same every time a particular user/VASP pair communicate. During the protocol the user and VASP establish a session key K which does vary from one protocol instance to another.

$$\begin{array}{rll}
\mathbf{U} \leftarrow \mathbf{V}: & p_{\mathrm{V}} & (1)\\
\mathbf{U}: & K^* = (p_{\mathrm{V}})^u & \\
\mathbf{U} \rightarrow \mathbf{V}: & p_{\mathrm{U}} & (2)\\
\mathbf{V}: & K^* = (p_{\mathrm{U}})^v & \\
\mathbf{U} \leftarrow \mathbf{V}: & \mathrm{E}_{K^*}\{\ \mathrm{CertV},\ r_{\mathrm{V}}\} & (3)\\
\mathbf{U},\mathbf{V}: & K = (K^*)r_{\mathrm{V}} & \\
\mathbf{U} \rightarrow \mathbf{V}: & \mathrm{E}_{K^*}\{\ \mathrm{CertU},\ r_{\mathrm{V}}\} & (4)
\end{array}$$

Fig. 4. The ASK protocol

The ASK protocol opens with a Diffie-Hellman exchange of public key agreement keys in (1) and (2) which results in each entity being able to compute the shared key K^*. The VASP then generates a random nonce and multiplies it by K^* to establish the session key K. The VASP sends this nonce and its certificate to the user in (3), both encrypted using K^*. The user decrypts (3) to obtain the VASP's nonce and then computes the session key. Finally the user sends its certificate to the VASP in (4), also encrypted using K^*.

Unfortunately, despite claims in [2], very few of the goals are met because of the following problems.

- The VASP is not authenticated to the user because the first and third messages could be intercepted and replayed by a third party (neither the first or third messages contain any information which will enable the user to check their freshness).
- User identity confidentiality may be compromised, since the user's public key is sent in clear in the second message. Linking protocol instances involving the same user will thus be straightforward, and if the public key of a user is known then user confidentiality is completely lost.
- Because there is no freshness checking of the third message, if a session key K should ever be compromised then known key attacks become possible. More specifically, it means that the user has no way of checking the 'freshness' of the session key K.
- There are potential problems with the use of symmetric encryption in the third and fourth messages. The authors suggest in [2] that a block or stream cipher could be used. However, given that the third

and fourth messages involve encryption using the same shared key, if a stream cipher is used then knowledge of one of CertU and CertV will reveal the other, providing another threat to identity confidentiality.

- There are no signatures used in the protocol, and hence providing non-repudiation services will require protocol enhancements.

The only goals which are met by the protocol would appear to be goals 2, 3 and 4. Extending the ASK protocol to the last two goals of Section II will involve at least one extra signature at the user end and one extra verification at the VASP end.

E. The Zhou-Lam protocol

Zhou and Lam [33] describe a pair of protocols that respectively provide user-to-VASP authentication and initialisation of a payment mechanism within a slightly different context to the one so far described. As a result of this, and of several concerns with details of this protocol, we provide some extra discussion here. They consider the situation where a user roams into a network, and wishes to both register with this 'roamed network' (involving mutual authentication) and set up a payment mechanism with this network. This is readily recast into the environment considered in this paper by mapping the 'roamed network' into the VASP. We represent the 'home network' by a TTP.

The first of the Zhou-Lam protocols, the 'Registration Protocol', provides mutual authentication between mobile user and VASP (with the online assistance of a TTP, necessary because of the use of symmetric cryptography). In addition, apart from establishing a shared session key between user and VASP, it also provides the user with a temporary signature key, the VASP with the verification function for this signature key, and both user and VASP with a temporary user identifier. The signature key is designed for use in the second protocol, the 'Service Request Protocol', which initialises the payment scheme (and can, in principle at least, be used many times before the Registration Protocol needs to be used again). The protocol relies on the previous establishment of two 'long term' keys, shared by the user and TTP, and VASP and TTP, respectively.

The Registration Protocol is as described in Figure 5. The Zhou-Lam protocol specifications use the following additional notation:

K_{UT}:	a long-term shared symmetric key between user and TTP;
K_{VT}:	a long-term shared symmetric key between VASP and TTP;
p_{U}:	temporary public verification key for user;
Req:	details of the service request made by user to the VASP;
s_{U}:	temporary private signature key for user;
$\mathrm{TID}_{\mathrm{U}}$:	temporary identity for the user;
T_e:	expiry date/time for the user's temporary signature key.

$$\begin{array}{rll} \mathbf{U} \to \mathbf{V}: & r_{\mathrm{U}},\ \mathrm{ID}_{\mathrm{T}},\ \mathrm{E}_{p_{\mathrm{T}}}\{r_{\mathrm{U}}, K_{\mathrm{UT}}, \mathrm{ID}_{\mathrm{V}}\} & (1) \\ \mathbf{V} \to \mathbf{T}: & r_{\mathrm{V}},\ \mathrm{E}_{p_{\mathrm{T}}}\{r_{\mathrm{U}}, K_{\mathrm{UT}}, \mathrm{ID}_{\mathrm{V}}\} & (2) \\ \mathbf{V} \leftarrow \mathbf{T}: & \left\{ \begin{array}{l} \mathrm{E}_{K_{\mathrm{VT}}}\{r_{\mathrm{V}}, \mathrm{TID}_{\mathrm{U}}, K, p_{\mathrm{U}}, T_e\} \\ \mathrm{Sig}_{\mathrm{T}}(\mathrm{TID}_{\mathrm{U}}, \mathrm{ID}_{\mathrm{V}}, p_{\mathrm{U}}, T_e) \\ \mathrm{E}_{K_{\mathrm{UT}}}(r_{\mathrm{U}}, \mathrm{TID}_{\mathrm{U}}, K, s_{\mathrm{U}}, T_e) \end{array} \right. & (3) \\ \mathbf{U} \leftarrow \mathbf{V}: & \left\{ \begin{array}{l} \mathrm{E}_{K_{\mathrm{UT}}}(r_{\mathrm{U}}, \mathrm{TID}_{\mathrm{U}}, K, s_{\mathrm{U}}, T_e) \\ h(K, r_{\mathrm{U}}) \end{array} \right. & (4) \end{array}$$

Fig. 5. The Zhou-Lam Registration Protocol

The Registration protocol opens with the generation of a random nonce by the user. The user then sends this nonce to the VASP in (1), as well as a message encrypted using the public key of the TTP that includes this nonce. The VASP generates a random nonce and sends this to the TTP in (2). The VASP also forwards in (2) the encrypted message sent by the user in (1). The TTP decrypts this incoming message and generates a session key K, as well as a temporary verification/signature key pair and a temporary identity for the user. The TTP then prepares a message consisting of three components. The first component is a list of values that include the session key, the user's temporary verification key and the user's temporary identity, all encrypted using the long-term VASP-TTP shared key. The second component is a signature on, amongst other things, the user's temporary verification key. The third component is similar to the first component, except that it is encrypted using the long-term user-TTP shared key, and replaces the temporary verification key of the user with the temporary signature key of the user. The TTP sends all three components to the VASP in (3). The VASP decrypts the first component and extracts K, verifies the second component and forwards the third component to the user in (4). The VASP also sends in (4) a hash of the session key and the user's nonce. The user decrypts the component, extracts K and verifies the hash.

Note that, as pointed out in [33], for billing purposes the TTP needs to retain the map between the user's permanent identity and the temporary identity $\mathrm{TID}_{\mathrm{U}}$. The Service Request Protocol is then as specified in Figure 6.

$$\begin{array}{rll} \mathbf{U} \to \mathbf{V}: & \left\{ \begin{array}{l} \mathrm{TID}_{\mathrm{U}},\ \mathrm{Req},\ \mathrm{cd},\ \mathrm{py} \\ \mathrm{Sig}_{\mathrm{U}}(\mathrm{TID}_{\mathrm{U}},\ \mathrm{cd},\ \mathrm{py}) \end{array} \right. & (5) \\ \mathbf{U} \leftarrow \mathbf{V}: & h(K,\ \mathrm{Req}) & (6) \end{array}$$

Fig. 6. The Zhou-Lam Service Request protocol

The Service Request protocol consists simply of a message (5) from the user to the VASP that includes a request for service and a signature on the user's temporary identity and charging/payment initialisation data. The VASP verifies this signature using the user's temporary verification key and returns in (6) a hash of the session key and the service request. The user verifies this hash to conclude the protocol.

We start our analysis of these two protocols by noting some anomalies in the protocol specifications.

- The encrypted string sent from user to TTP via the VASP at the start of the Registration Protocol is asymmetrically encrypted rather than being encrypted using K_{UT} because the TTP will not know the user's identity in advance. The encrypted string is the only means by which the TTP can identify the user. However, the only user-specific information in this string is the shared key K_{UT}, and this may not be a convenient way for the TTP to identify the user. Moreover if K_{UT} is used in this way it is serving two different functions, i.e. to identify the user and to confirm the origin of data. For these reasons it would be better to explicitly include $\mathrm{ID_U}$ in the encrypted string sent from the user to the TTP via the VASP.
- There is an implicit assumption that the asymmetric encryption function provides inherent 'binding together' of data items. I.e. it is necessary for the security of the protocol that an interceptor cannot manipulate an encrypted string to produce another one with a predictable decryption, even if part of the plaintext is known. This assumption holds for some public key encryption schemes such as RSA, as long as all the data fits in one 'block', but it is not clear that it is guaranteed for all such schemes.
- It is also implicitly assumed that the symmetric encryption used to protect data sent from TTP to user via the VASP provides data integrity and origin authentication as well as confidentiality. If it does not, a malicious active interceptor might, for example, be able to persuade the user to accept a different key K. This combination of confidentiality and integrity protection is not easy to achieve in practice, and will typically require the computation of a MAC on the data using one variant of the key, followed by data encryption using a second key variant (see also recent work by Gligor and Donescu [14] and Jutla [20], who propose new modes of operation of block ciphers that claim to provide both confidentiality and message integrity).
- The value of the signed string sent from the TTP to the VASP in the Registration Protocol is unclear. Although it is intended to commit the TTP to the temporary identity $\mathrm{TID_U}$ and the temporary verification key p_{U}, problems arise because there are no guarantees regarding the 'freshness' of the signed string. The TTP could issue the same signed string (and the same temporary identity and signature/verification key pair) to a number of users of a single VASP. When the VASP presents signed commitments for payments by these users, the TTP could falsely refuse to honour them on the grounds that he only issued the signed string once. This problem could be avoided by requiring the TTP to include the nonce r_{V} inside the signed string it supplies to the VASP.

The Registration Protocol does not provide entity authentication of user to VASP, but this service is not strictly necessary within the registration process itself, since the subsequent receipt of messages signed using the user's temporary signature key (in the Service Request Protocol) will implicitly prove the user's presence during the registration process. There is no need for exchange of certified public keys in this protocol. Joint key control is not provided as the key K is selected by the TTP. There is the possibility of a limited content verification attack on the signed string sent from the TTP to the VASP, since an interceptor will know all the signed data except for the user's verification key and its expiry date. Time-memory tradeoff attacks are possible on the hashed value sent from VASP to user in the Service Request Protocol (since the parameter Req may only take a few different values), which implies that the key K should contain enough bits (e.g. 128 or more) to resist such attacks.

F. The Boyd-Park protocol

The Boyd-Park protocol proposed in Boyd and Park [7] is illustrated in Figure 7. The Boyd-Park protocol specification uses the following additional notation:

p_{V}:	public key of the VASP.

$$
\begin{array}{rll}
\mathbf{U} \rightarrow \mathbf{V}: & \mathrm{Enc}_{p_{\mathrm{V}}}\{\mathrm{ID_U}, r_{\mathrm{U}}\} & (1)\\
\mathbf{V}: & K = h(r_{\mathrm{U}}, r_{\mathrm{V}}) & \\
\mathbf{U} \leftarrow \mathbf{V}: & r_{\mathrm{V}},\ \mathrm{E}_K\{r_{\mathrm{U}}\} & (2)\\
\mathbf{U}: & K = h(r_{\mathrm{U}}, r_{\mathrm{V}}) & \\
\mathbf{U} \rightarrow \mathbf{V}: & \mathrm{Sig_U}\{\mathrm{ID_V}, h(r_{\mathrm{V}}, K)\} & (3)
\end{array}
$$

Fig. 7. The Boyd-Park protocol

The Boyd-Park protocol begins with the user generating a random nonce and sending it to the VASP in (1), encrypted using the public key of the VASP. The VASP decrypts this message, generates a random nonce of its own, and computes key K by hashing these two nonces together. The VASP then sends its nonce to the user in (2), including the user's nonce encrypted using K. The user can now also compute K, confirming it by decrypting the second part of (2). Finally the user signs a maessage that includes a hash of the VASP's nonce together with K and sends this in (3) to the VASP.

The VASP verifies this signature to conclude the protocol.

The Boyd-Park protocol is interesting because it does not use a Diffie-Hellman variant to establish the common key. Rather, the key is a hash of random nonces selected by the user and the VASP, with only the nonce of the user being a secret value. In this respect the key is very efficient for both entities to compute. The Boyd-Park protocol appears to satisfy all the goals except for goal 2, the exchange of certified public keys, and the last two goals. The last two goals can easily be met because the Boyd-Park protocol involves a signature in the third message from user to VASP. Meeting goal 2 is less straightforward because, although the user can easily send its encrypted certificate to the VASP in the first message, the user needs the public key of the VASP to encrypt this message. Thus to meet all the goals the Boyd-Park protocol must involve an additional initial pass in which the user obtains, and verifies, the public key of the VASP. The Boyd-Park protocol is vulnerable to a signer verification attack, but in [7] it is suggested that this can be prevented by the use of ElGamal signatures (or suitable variants).

G. The ASPeCT protocol

The ASPeCT protocol was developed by the European Commission ACTS project ASPeCT [1] and versions of the protocol have previously appeared in Horn and Preneel [16] and Martin et al. [23]. The protocol that is shown in Figure 8 is explicitly designed as an API protocol. The following additional notation is used:

$\mathrm{ID_T}$	the identifier of the user's CA;
CertU:	certified public signature key of user;
CertV:	certified public key agreement key (g^v) of the VASP;
$h1, h2, h3$:	one-way hash functions (see [16] for detailed requirements).

$$\begin{array}{rll}
\mathbf{U} \rightarrow \mathbf{V}: & g^{r_{\mathrm{U}}}, \mathrm{ID_T} & (1) \\
\mathbf{V}: & K = h1(r_{\mathrm{V}}, g^{v r_{\mathrm{U}}}) & \\
\mathbf{U} \leftarrow \mathbf{V}: & r_{\mathrm{V}}, h2(K, r_{\mathrm{V}}, \mathrm{ID_V}), \text{cd, TS, CertV} & (2) \\
\mathbf{U}: & K = h1(r_{\mathrm{V}}, g^{r_{\mathrm{U}} v}) & \\
\mathbf{U}: & H = h3(g^{r_{\mathrm{U}}}, g^{v}, r_{\mathrm{V}}, \mathrm{ID_V}, \text{cd, TS, py}) & \\
\mathbf{U} \rightarrow \mathbf{V}: & \mathrm{E}_K\{\mathrm{Sig_U}(H), \text{CertU, py}\} & (3)
\end{array}$$

Fig. 8. The ASPeCT protocol

The ASPeCT protocol also establishes the session key $K = h1(r_{\mathrm{V}}, g^{v r_{\mathrm{U}}})$ using a variant of Diffie-Hellman that is based on the private key of the VASP and a random nonce generated by the user. The user commences by generating a random nonce, uses it to compute a temporary public key, and sends this to the VASP in (1), along with an identifier of the TTP that can verify the user's certificate. The VASP generates a random nonce and computes key K. In (2) the VASP sends a certified copy of his public key, details of the charging data, a timestamp (for later use in the payment process), its random nonce and signifies knowledge of K by sending a copy of it hashed with its random nonce and identity. On receipt of (2) the user can now also compute K. The user replies in (3) by encrypting a signed hashed copy of a number of protocol variables and his certified public signature key. The signature provides non-repudiation on the billing data, satisfying the last goal in Section II. The VASP decrypts the message received in (3) and verifies the signature to conclude the protocol.

The ASPeCT protocol meets all the required API goals (see [16] for a detailed justification). We note that signer and content verification attacks are prevented by encryption of the signature in the third message. The inclusion of an identifier for the VASP and a random value in the hash of the second message prevents source substitution attacks and practical time-memory tradeoff attack to find K. As the key K is equal in two different protocol runs with negligible probability, the ASPeCT protocol is not vulnerable to codebook attacks to determine K. The protocol is potentially vulnerable to partial chosen key and key separation attacks (but see Section III).

V. Comparison of protocols

In this section we make a short comparison between the candidate protocols.

A. Cryptographic methods

The cryptographic technology required in most of the seven protocols is very similar. In all the protocols except Aziz-Diffie, Zhou-Lam and Boyd-Park, the shared session key established between user and VASP is computed using a variant of Diffie-Hellman. All the protocols except Aziz-Diffie require a symmetric encryption algorithm. To satisfy the last two goals all the protocols will at least require the user to perform digital signatures. Both the Zhou-Lam, Boyd-Park and ASPeCT protocols also use a one-way hash function. The Revised BCY protocol is unique in requiring the use of the MSR public key system. The most significant cryptographic difference between all of these protocols is probably the fact that the Zhou-Lam protocol is the only one that is almost entirely symmetric key based, and requires shared symmetric keys to have already been established and maintained between user and VASP, and VASP and the TTP.

B. Procedural and communication issues

Some slight differences exist in the protocol procedures. The first pass of the Revised BCY and ASK (and potentially Boyd-Park) protocols is from the VASP, while in the STS, Aziz-Diffie, Zhou-Lam and ASPeCT protocols it is from the user. The user-to-VASP scenario requires the user to initiate the session somehow, and so it makes some sense for the user to start the protocol, although this is not such an important point. More significant is the fact that the Revised BCY protocol only involves one pass in each direction between user and VASP, whereas the other protocols have three or four passes. In this respect the Revised BCY, although it fails to achieve several of the protocol goals, is the most efficient of the protocols.

It is worth commenting that in the ASPeCT protocol, and in all the envisaged extensions of the other protocols to include payment initialisation, the actual payment protocol is decoupled from this initialisation process. By contrast, the Zhou-Lam protocol links initialisation and payment together, both decoupled from the authentication mechanism. The relative merits of these two techniques very much depend on the likely time period over which a series of VAS requests can be made without a subsequent re-running of the initialisation process. The decoupling arrangement of the protocols discussed here is more favourable if the lifetime of temporary public keys is very short, or a series of VAS requests are made to one VASP within a short time period which can all be based on the same initialisation data. The alternative decoupling arrangement may be attractive in situations where temporary keys have slightly longer lifetimes, and only one authentication and key establishment based on them is needed in advance of a number of payment requests.

Finally note that, by use of public key techniques, all protocols but the Zhou-Lam protocol avoid the need for communications between the VASP and the TTP at the time the user establishes its initial relationship with the VASP (unless the appropriate certificates are not in place). The Zhou-Lam protocol's need for routine communications with the TTP involves a significantly increased load on the network.

C. Meeting the goals

Table I summarises how the seven protocols (listed from left to right in order of presentation in Section IV) meet the goals of Section II. For goals 9 and 10, an entry ($\star$) indicates that the protocol is easily modified to meet the goal, and $+\star$ indicates that the protocol can be modified to meet the goal, but involves at least one extra signature (one extra pass in the case of the Boyd-Park protocol). See Section IV-E for comments on how well the Zhou-Lam protocol meets goal 1.

TABLE I
GOALS ACHIEVED BY EACH OF THE PROTOCOLS

Goals	Protocol						
achvd	SS	AD	RB	AK	ZL	BP	AT
1	$\star$	$\star$			($\star$)	$\star$	$\star$
2	$\star$	$\star$	$\star$	$\star$		$\star$	$\star$
3	$\star$	$\star$	$\star$	$\star$	$\star$	$+\star$	$\star$
4	$\star$	$\star$	$\star$	$\star$		$\star$	$\star$
5	$\star$	$\star$			$\star$	$\star$	$\star$
6	$\star$				$\star$	$\star$	$\star$
7	$\star$	$\star$			$\star$	$\star$	$\star$
8	$\star$		$\star$		$\star$	$\star$	$\star$
9	($\star$)	($\star$)	$+\star$	$+\star$	$\star$	($\star$)	$\star$
10	($\star$)	($\star$)	$+\star$	$+\star$	$\star$	($\star$)	$\star$

From Table I we see that only the STS, Boyd-Park and ASPeCT protocols meet all the specified goals. Further, two of the protocols, the Revised BCY and ASK protocols, fail to meet at least three of the goals, including the goal of mutual entity authentication.

D. Computational loads

In Tables II and III we compare the computational load of the seven protocols. In the first row we measure the number of pre-computable exponentiations, while the number of online exponentiations are counted separately in the second row. The numbers of generic public key encryptions and decryptions are given in the third and fourth rows respectively; these may well involve exponentiations, but we count them separately when the public key algorithm has not been explicitly specified. Likewise the numbers of signatures and verifications are respectively indicated in the last two rows. Bracketed entries in Tables II and III indicate figures under the assumption that these protocols have been extended to offer initialisation of the payment mechanism and non-repudiation of the initialisation data, in some cases through the addition of one digital signature by the user, and one verification by the VASP.

TABLE II
PROTOCOL COMPUTATIONAL LOAD AT THE USER END

User	Protocol						
Comp.	SS	AD	RB	AK	ZL	BP	AT
PrEx	1	0	0	0	0	0	1
Ex	1	0	1	2	0	0	1
PKE	0	1	0	0	1	1	0
PKD	0	1	0	0	0	0	0
Sig	1	1	(1)	(1)	1	1	1
Ver	2	2	1	1	0	(1)	1

The results in Table II show that the most computationally efficient protocol is the symmetric key based Zhou-Lam protocol. The Revised BCY protocols involves one (pre-computable) exponentiation less than both the ASK and ASPeCT protocols, but with respect to ASPeCT achieves this at the cost of failing to meet several of the required goals. Of the three protocols that meet all the goals of Section II, the Boyd-Park protocol involves the least computational effort at the user end. However it achieves this at the expense of requiring an extra protocol pass. Of the other two, the ASPeCT protocol involves one verification less than the STS protocol. This gain is also accentuated by the fact that the shorter signature lengths of some ElGamal-type signatures (DSA [13] and elliptic curve variants, for example) make them preferable to RSA-based signatures for implementation on a smart-card, and that these come at an approximate computational cost of one pre-computable exponentiation for signing, and two exponentiations for verifying (see Menezes et al. [24]). Of course, the length advantage of certain discrete logarithm signatures over RSA signatures disappears if signatures with message recovery are used. However, such signature schemes are outside the scope of this review and their use has not been proposed with any of the cited protocols. Thus the approximate number of exponentiations in a run of the STS protocol is 7 (2 pre-computable) versus 5 (2 pre-computable) for ASPeCT, and 4 (one pre-computable) for Boyd-Park, assuming RSA is used for encryption. The Aziz-Diffie protocol has a similar user overhead to the STS protocol, but does not provide all protocol goals.

TABLE III
PROTOCOL COMPUTATIONAL LOAD AT THE VASP END

VASP	Protocol						
Comp.	SS	AD	RB	AK	ZL	BP	AT
PrEx	1	0	0	0	0	0	0
Ex	1	0	1	2	0	0	1
PKE	0	1	0	0	0	0	0
PKD	0	1	0	0	0	1	0
Sig	1	1	0	0	0	0	0
Ver	2	2	(3)	(2)	2	(2)	2

Although less practically significant at the time of writing (but see comments in Section I), the VASP computational efforts are given in Table III. The balance between user and VASP effort in each protocol is worth noting. The STS, Aziz-Diffie, ASK and Boyd-Park protocols have very similar user and VASP efforts. The Extended BCY and Zhou-Lam protocols both appear to have succeeded in reducing the user computation with respect to the VASP. The ASPeCT protocol appears at first glance to involve more computation at the user end; however if ElGamal-type signatures are used then the effort becomes marginally less at the user end.

E. Message length

The techniques for keeping the lengths of protocol messages as short as possible are often outside the scope of the protocol descriptions themselves. For the ASPeCT protocol, it is suggested in Horn and Preneel [16] that the user generates signatures using an elliptic curve based AMV signature as specified in ISO/IEC FDIS 14888-3 [19]. The ASPeCT protocol has also been implemented using a streamlined certificate format [16]. Similar procedures could easily be adopted for most of the other protocols.

VI. Concluding remarks

We have provided a clear set of goals for a third generation mobile system API protocol. We listed some generic attacks against such protocols; the possibility of these attacks has influenced the design of the ASPeCT protocol, and more generally provides a reference point for future protocol design.

Using our defined goals, the suitability of seven candidate API protocols was considered. Four of these are shown to have significant shortcomings. Three of these are largely through the failure to meet all of our protocol goals. A fourth protocol, the Zhou-Lam protocol, largely fails because, although it compares very favourably with the other protocols in terms of computational load, it imposes an increased load on the network because of the need for routine communications with a TTP. Of the three protocols that meet all the protocol goals, the STS protocol was not designed explicitly for such a mobile application and as a result has rather larger computational overheads than are necessary. The Boyd-Park protocol seems to be the most computationally efficient at the user's end, but requires the most communication passes. The ASPeCT protocol was purposely designed for API application and like the Boyd-Park protocol, appears to compare well very favourably with the prior art.

We acknowledge the cooperation and assistance of all members of the ASPeCT project and in particular express thanks to Peter Howard, Bart Preneel and Konstantinos Rantos, for informative discussions and comments.

References

[1] Advanced Security for Personal Communications Technologies. http://www.esat.kuleuven.ac.be/cosic/aspect/

[2] M. Aydos, B. Sunar and Ç.K. Koç. An elliptic curve cryptography based authentication and key agreement protocol for wireless communication. Presented at the *2nd International Workshop on Discrete Algorithms and Methods for Mobility (DIAL M 98)*, Dallas, Texas, October 1998.

[3] A. Aziz and W. Diffie. Privacy and authentication for wireless local area networks. *IEEE Personal Communications*, First Quarter:25–31, 1994.

[4] M.J. Beller, L.-F. Chang and Y. Yacobi. Privacy and authentication on a portable communications system.

IEEE Journal on Selected Areas in Communications, 11:821–829, 1993.

[5] S. Blake-Wilson and A. Menezes. Unknown key-share attacks on the Station-to-Station (STS) Protocol. *Proceedings of PKC '99*, Lecture Notes in Comput. Sci., 1560:154–170, 1999.

[6] J. Borst, B. Preneel and J. Vandewalle. On the time-memory tradeoff between exhaustive key search and precomputation. *Proceedings of the Nineteenth Symposium on Information Theory in the Benelux*, Veldhoven, 111–118, 1998.

[7] C. Boyd and D.-G. Park. Public key protocols for wireless communications. Presented at the 1998 International Conference on Information Security and Cryptology (ICISC '98), Seoul, Korea.

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

[9] D.E. Denning and G.M. Sacco. Timestamps in key distribution protocols. *Communications of the ACM*, 24:533–536, 1981.

[10] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Info. Theory*, 22:644–654, 1976.

[11] W. Diffie, P.C. van Oorschot and M.J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2:107–125, 1992.

[12] ETSI SMG DOC 73/95. A public key based protocol for UMTS providing mutual authentication and key agreement, September 1995.

[13] FIPS 186. Digital signature standard. *Federal Information Processing Standards Publication 186*, U.S. Department of Commerce/N.I.S.T., 1994.

[14] V.D. Gligor and P. Donescu. Fast encryption and authentication: XCBC Encryption and XECB Authentication modes. *Preproceedings of FSE 2001*, Yokohama, Japan, 97–111, 2001.

[15] M. Hellman. A cryptanalytic time-memory tradeoff. *IEEE Transactions on Information Theory*, 26:401–406, 1980.

[16] G. Horn and B. Preneel. Authentication and payment in future mobile systems. *Journal of Computer Security*, 8:183-207, 2000.

[17] ISO/IEC 9796-2: 1997. Information technology — Security techniques — Digital signature giving message recovery — Part 2: Mechanisms using a hash-function.

[18] ISO/IEC 11770-3: 1999. Information technology — Security techniques — Key management — Part 3: Mechanisms using asymmetric techniques.

[19] ISO/IEC 14888-3: 1998. Information technology — Security techniques — Digital signature with appendix — Part 3: Certificate-based mechanisms.

[20] C.S. Jutla. Encryption Modes with Almost Free Message Integrity. *Advances in Cryptology - EUROCRYPT 2001*, Lecture Notes in Comput. Sci., 2045:529–544, 2001.

[21] H.Y. Lin and L. Harn. Authentication protocols for personal communication systems. *Proceedings of ACM SIGCOMM'95*, 256–261, August 1995.

[22] K.M. Martin and C.J. Mitchell. Comments on an optimized protocol for mobile network authentication and security. *Mobile Computing and Communications Review*, 3 No.2, 37, 1999.

[23] 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. *5th International Conference in Services and Networks, IS&N '98*, Lecture Notes in Comput. Sci., 1430:535–548, 1998.

[24] A.J. Menezes, P.C. van Oorschot and S.A. Vanstone. *Handbook of Applied Cryptography*, CRC Press, 1997.

[25] C.J. Mitchell and L. Chen. Security in future mobile multimedia networks. Chapter 11 of *Insights into Mobile Multimedia Communications*, eds. D.R. Bull, C.N. Canagarajah and A.R. Nix, Academic Press, 1999, pp. 177-190.

[26] C.J. Mitchell, M. Ward and P. Wilson. Key control in key agreement protocols. *Electronics Letters*, 34:980–981, 1998.

[27] Y. Mu and V. Varadharajan. On the design of security protocols for mobile communications. *Information security and privacy*, Lecture Notes in Comput. Sci., 1172:134–145, 1996.

[28] C.-S. Park. On certificate-based security protocols for wireless mobile communication systems. *IEEE Network*, September/October, Vol.11, 5:50–55, 1997.

[29] T.P. Pedersen. Electronic payments of small amounts. *Proc. Int. Workshop Security Protocols*, Lecture Notes in Comput. Sci., 1189:56–68, 1996.

[30] R.L. Rivest and A. Shamir. PayWord and MicroMint: Two simple micropayment schemes. *Cryptobytes*, Vol.2, 1:7–11, May 1996.

[31] H.C. Williams. A modification of the RSA public-key encryption procedure. *IEEE Trans. Info. Theory*, 26:726–729, 1980.

[32] X. Yi, E. Okamoto and K.Y. Lam. An optimized protocol for mobile network authentication and security. *Mobile Computing and Communications Review*, Vol.2, 3:37–39, 1998.

[33] J. Zhou and K.-Y. Lam. Undeniable billing in mobile communications. In *Proceedings of the 4th ACM/IEEE International Conference on Mobile Computing and Networking, Dallas, Texas, October 1998*, ACM Press, 1998.

Günther Horn received a Ph.D. (Dr.rer.nat.) in Mathematics from the University of Tübingen in 1984. He was a Visiting Assistant Professor of Mathematics at the University of California at Irvine in 1985, and a visiting scientist at the University of Ottawa in winter 1986. He joined the Corporate Technology Labs of Siemens AG in Munich, Germany, in 1986. He has been engaged since in research on new telecommunications systems. For the past nine years, Mr. Horn has been mainly working on security in mobile systems, in particular on UMTS security, fraud control, electronic payments for mobile applications and security in train control systems. He participated in several collaborative research projects on mobile telecommunications security. He is currently active in the standardisation of UMTS security in 3GPP.

Keith M. Martin received his BSc in Mathematics from the University of Glasgow in 1988 and a PhD from Royal Holloway, University of London in 1991. Between 1992 and 1996 he held a Research Fellowship in the Department of Pure Mathematics at the University of Adelaide, investigating mathematical modeling of cryptographic key distribution problems. In 1996 he joined the Computer Security and Industrial Cryptography research group of the Katholieke Universiteit Leuven in Belgium as a Postdoctoral Fellow. In 2000 he returned to Royal Holloway as a Lecturer in the Information Security Group. He has also held visiting positions at the University of Wollongong and University of Adelaide. His current research interests include cryptography, mobile security and electronic payment systems.

Chris J. Mitchell Chris Mitchell obtained his BSc (1975) and PhD (1979) degrees in Mathematics from Westfield College, University of London. Prior to his appointment as a Professor of Computer Science at Royal Holloway in 1990, he was employed at Racal Comsec, Salisbury, UK (1979-85), and at Hewlett-Packard Laboratories, Bristol, UK (1985-90). He has worked in the field of Information Security for nearly 22 years, and whilst at Royal Holloway has continued his research in various aspects of the field, including participation in a series of collaborative projects on mobile telecommunications security, He has published over 100 research papers and has edited six international standards on cryptographic techniques.