Applicability of Host Identities to Implement Non-Repudiable Service Usage

Seppo Heikkinen
Tampere University of Technology
firstname.lastname@tut.fi

Abstract

In a typical roaming scenario the accounting information received from the roaming partner is expected to be trustworthy. Things like fear of losing one's reputation have been working as disincentives for fraudulent behaviour between the large operators. However, when smaller players enter the market and steps are taken towards more dynamic relationships as in the visions of ubiquitous computing environments, the need for reliable records becomes paramount. Thus, secure accounting mechanisms are needed for ensuring correct compensation amongst the interoperating partners. On top of that, the partners need to be authorised with sufficient granularity to be able to engage in the transaction in the first place. The mere authentication should not be enough.

In this article we present a solution concept for ensuring non-repudiation of the service usage, so that cryptographically secure accounting records can be generated, and the parties involved in the transaction make their commitments only to the resources actually consumed. The solution is based on the employment of Host Identity Protocol (HIP) and hash chains, so that we can provide a convenient binding between the identity and authorisation information. Also, in order to avoid service hijacking, mechanisms for binding this information to the actual traffic are discussed.

Keywords: hash chains, host identity, non-repudiation, service

1. Introduction

The communication environment is changing. As the ubiquitous computing paradigms gain more momentum and the technological development allows more dynamic usage patterns and relationships, more and more small players enter the market to get their piece of the service provisioning cake. Naturally, these players want to ensure that they receive authentic users that are able to pay for the service usage. On the other hand, the players vouching for the liability of the users want to make sure that the generated expenses are within certain limits, i.e., they want to control how much risk they are willing to take on behalf of their customers. This requires measures to ensure the correct authorisation for the users of the systems.

Thus, we have service providers, who want to receive compensation for the provision of their service resources. They are complemented by the third parties, such as home operators, who help in authenticating users and ensuring that the generated costs will be covered. Finally, we have the users, who want to make sure that they receive the service that is promised and that it is correctly charged. After all, the appearance of unauthorised charges on phone bills, i.e., cramming, is not unheard of amongst the consumers [1]. Sometimes, the user may not even have clear notion about the identity of the responsible service provider, as is often the case with visited access networks, even though the access network might be in the possession of authentication material generated by the home network.

As the interaction and the established relationships are more dynamic in nature and lasting perhaps only one transaction, typical assumption that the loss of reputation is incentive to ensure the correctness of accounting records is no longer valid. Hence, we need mechanisms that create secure accounting records so that the service transaction is undeniable and authentic for the both parties of the transaction. We propose such a simple non-repudiable mechanism that takes advantage of Host Identity Protocol (HIP) and hash chains. Our focus is on the interaction of the user and the service, not so much in the negotiation between the service and the third party nor the bootstrapping of trust between the user and the third party.

HIP already provides end point authentication and simple key exchange, but it does not currently address the problem of authorisation to the sufficient detail. In order to implement the suggested Non-Repudiable Service Usage (NoRSU), one point of this article is to

discuss how to include authorisation tokens into HIP and what are the consequences. Additionally, the hash chains are employed to introduce an incremental payment solution, i.e., a chain of tokens is created by repeatedly hashing a secret seed value. Thus, the service provider is able to generate undeniable charging records and the user can be sure that the charging is based on actual use. As HIP assigns cryptographic identities to the communication end points, the tokens can be tightly bound to the actual communication. HIP also introduces a handshake procedure for negotiating and establishing a security association between the end points. For the benefit of performance this procedure can be overloaded with the compensation related information. Thus, no additional roundtrips are introduced.

This article is organised as follows. The next section discusses the related work. The third section describes the details of the proposed system and the section after that gives examples of two use cases. The fifth section discusses the limitations the implementations have to take into account and suggests ways to efficiently encode the used information in order to overcome these limitations. The sixth section analyses the solution in terms of threats that can be faced. Finally, the seventh section concludes the article.

2. Related work

HIP is an experimental proposal for future network architectures that introduces a new identity layer between the network and transport layers [2]. This allows decoupling the dual role of the IP addresses. That is, currently they function as end point identities and locators. In the HIP model the end points are identified by their cryptographic identifiers, called Host Identity Tags (HIT), which are derived from their public keys. This accommodates for end host authentication and simple key exchange. Thus, the parties are able to setup a security association between themselves, which can be used to protect the control information exchange. Additionally, the protection of subsequent data transport is possible with IPsec ESP [3]. Other transports can be defined, too.

HIP uses four messages in the so called base exchange to establish the identity of the parties and to create the needed keying material with the help of Diffie-Hellman key exchange (see Figure 1). For the purposes of the paper the initiator and the responder can be considered as the client and the server, respectively. Besides securing the message exchange, the protocol mitigates denial of service (DoS) attacks by introducing a puzzle scheme.

An initial proposal for including authorisation to HIP has been introduced, but that work is still very much in the draft stage and basically provides a placeholder for the certificates [4]. Like the proposal, [5] and [6] also discuss the possibility of including Simple Public Key Infrastructure (SPKI) certificates in the protocol, but do not analyse the use case thoroughly, even though [5] provides a prototype implementation adapted to grid environments. There is also a general sketch of an attachment architecture, which includes both HIP and compensation related issues in [7]. A solution employing hash chains and KeyNote credentials to implement One Time Password (OTP) coins was depicted in [8], even though without clear binding to the actual communication. Similar ideas were used to sketch a high level solution presented in [9], but it used SPKI certificates instead of KeyNote and already took advantage of HIP to ensure the binding to the actual traffic. The text presented here extends that work with additional details.

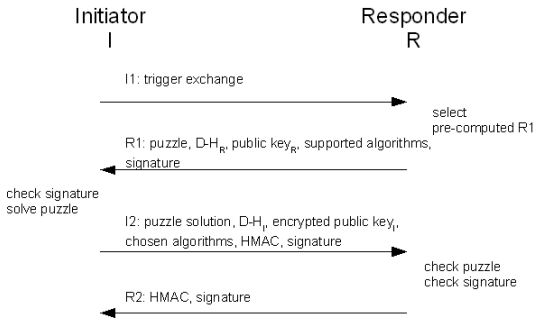

Figure 1. HIP base exchange

Hash chains have been used for password solutions and such one-time password authentication was suggested already in 1981 by Lamport [10]. The idea of hash chains is based on the irreversible nature of the hash functions. In other words, you are not able to calculate the source value once you have the result of the function. Hash chain is created by applying a secure hash function in successive fashion to a secret seed value and then using the values of the hash calculations in reverse order. So, it is very easy to check by applying one hash operation to the previously received value that the current value is part of the chain, but very hard to calculate additional values without the knowledge of the initial seed value of the chain. The idea behind hash chains is illustrated in Figure 2.

There exists also several other works, which have considered employing hash chains to introduce non-repudiable billing and micropayments in various

scenarios, so the concept is not new. [11] uses hash chains to implement a payment solution for ad hoc networks, but it requires the use of smart card technology to control the release of hash chain values. [12] also presents a protocol for undeniable billing with entity authentication and privacy support for mobile networks roaming access using hash chains, although it requires online interaction with the home network.

x = secret seed value

H = Hash function

n = length of chain

$H^n(x) = H(H^{n-1}(x))$

$H^n(x)$ = hash chain anchor

release chain values:

$H^n(x), H^{n-1}(x), H^{n-2}(x), ...$

Figure 2. Idea behind hash chains

3. HIP based non-repudiation

This section describes how the hash chains are integrated with the HIP base exchange.

3.1 General overview

Our proposal, which is based on the aforementioned HIP, works in the way depicted in Figure 3 (HIP specific parameters left out). The basic idea is to add extra information to the HIP messages in order to negotiate the usage of non-repudiative accounting within the communication. So, in a sense, we are negotiating a non-repudiation association in addition to the identity association.

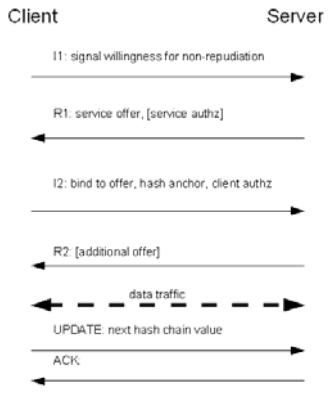

Figure 3. Base exchange with non-repudiation enhancements

A new HIP parameter is needed to signal the intent to access certain service with the capability of using NoRSU. Note that the server could also send this kind of indication to the client at some later point using the HIP UPDATE packets along with the corresponding offer, if the client tried to access a service, for which the server required extra accounting (provided they had an existing HIP association). Figure 4 gives an example of the said HIP parameter for indicating the use of non-repudiation for certain service and it also shows the general Type-Length-Value (TLV) format of HIP parameters (C bit denotes possible critical parameter).

Type (15 bits)		C	Length (16 bits)
Subtype of NoRSU	Encoding of name		Service name length (16 bits)
Service name with the indicated encoding + padding if needed (variable length)			

Figure 4. New HIP parameter for signalling the use of non-repudiation for a specific service

3.2. Modified base exchange messages

The tasks for individual HIP messages are as follows. The I1 message functions as a trigger message as in basic HIP base exchange [2], but with the addition of possibility to signal the capability of the client to engage in a NoRSU exchange as discussed above. The server's response, R1, contains an offer in the form of an SPKI certificate for the usage parameters, including the number of tokens needed for certain amount of time or byte count, e.g., you need one token per minute or you need one token per 100 kilobytes. That is really up to the charging scheme of the provider, but generally the value of one token should be kept small in order to avoid big losses in case of abuse. There could also be an additional advice of charge functionality telling the value of one hash chain token in monetary terms. However, this does not take into account the possibility that the tariffs are different between different parties, i.e., some client have a "better deal", because they are subscribers of some favoured organisation, for instance.

The offer is protected by a signature, which also binds the provider to the offer. The signature could also be made by a trusted third party (TTP) in order to guarantee that the server is a legitimate one, but this could also be done by using an additional authorisation certificate (see subsection 3.3). The latter case is more flexible and gives more control of the tariffs to the

service provider, naturally within the limits of the third party authorisation.

Offer should also contain validity date, so that the provider has better control of the expiration of the used offers. This allows, for example, using different tariffs at different times of the day. Naturally, if the session continues after the validity period the parties should renew their contract provided the new offer is satisfactory. It is the responsibility of the client to make sure that no additional hash chain values are sent with the assumption that the old offer is still valid. The server can in this case just stop serving the client, if there is no response to the new offer.

Note that instead of offering certain time or rate based traffic the offer could just be for the use of certain service, which could be described by a profile or a service name. This naturally requires that there is common consensus about the semantics of such profiles, but that can be agreed when establishing trust relationship, e.g., a roaming agreement, with the third party. There could also be several offers, e.g., choice between time and byte count, but this has restrictions as space is limited (see section 5).

If the offer meets the requirements of the client, it sends a response in the I2 message, which contains the signed acceptance of the offer. The acceptance is indicated by calculating a hash over the offer and signing it. Additionally, the response must contain the hash anchor value, which the server can use to validate the subsequent values, i.e., it acts as the starting point for the hash chain, which the client has created. It can also be used to identify the whole hash chain among several parallel chains.

The fourth message of the exchange, i.e., R2, can just acknowledge the validity of the offering process and, for instance, show as a summary what kind of agreement is in effect. Some advanced scenarios are possible, though. One could relate to special offers, i.e., if the customers of certain operator were allowed to get even cheaper service, R2 could contain a special offer with a reduced tariff. This could mean, for example, a longer interval between subsequent hash chain values. The client would need to send an additional control message to sign the offer with a new hash chain anchor value. Otherwise, it could still be charged the higher price. This could be found out, though, when (or if) the client disputes the costs and presents the alternative offer. So, in low value transactions it could be possible to just use R2 to signal that the hash chain value interval is shifted (for the benefit of the client). Another approach is that the value of token is lower between service provider and the third party, thus the generated bill is lower.

3.3. Authorisation issues

While the service provider might have some external knowledge about the client's liability for service usage based on its identity, generally the client also needs to attach an authorisation statement from TTP that states that the client is trustworthy to receive the specified service for the specified amount. The service might be specified based on service types or it could be specified on the provider level. In other words, the specified service and the service offer identities should then match. This is a slightly less flexible option, but provides more security, because overspending can be controlled more easily. In case the service granularity is just based on the service type, the client could use several different service providers for the maximum amount defined in the certificate during the validity period. Of course, at the time of the clearing the third party would notice this and could initiate appropriate procedures against the client. This is really no different from the way post-paid phone calls are charged. Thus, TTP has the liability, but it can still control what sort of certificates it issues and hence manage its own customer risk. Issuing only short lived authorisations is also one way of mitigating risk.

```
(
 (cert
  (subject (hash sha1 <hash value>))
  (issuer (hash sha1 <hash value>))
  (target-service-url (hash sha1 <hash value>)
  (amount-max (time (s 3600)))
  (propagate)
  (validity
   (not-before 2008-07-29_12:00:00)
   (not-after 2008-07-30_12:00:00))
)
 (signature (dsa-sha1 <sig>))
)
```

Figure 5. Example of TTP certificate

In Figure 5 we give an example of a third party authorisation in the form of an SPKI certificate, which allows certain subject to access the indicated service. The subject is identified by the hash of the public key, even though one could also use HIT to identify the party. However, as HIT includes IPv6 kind of interpretation (see [13]), there is slightly larger chance of collision than in the case of hashing a public key. The target service is also indicated with the hash value calculated from the service URL (or just with suitable URN) and the maximum service time is also indicated in order to limit the "credit" of the client. Note that

TTP certificates could authorise various other things as well and act as a policy distribution mechanism. This is really up to the agreement made by the service provider and the third party.

The example certificate also includes the propagate option, which allows the client to assign similar rights to some other entity, but ultimately it is still responsible for the incurred costs. Of course, there is no obligation for the third party to allow the delegation in the first place, but the privacy of the client is better served, if there is a possibility of delegating the authorisation to an ephemeral identifier, which is visible to the external observers of the base exchange. This kind of setting also enables the user to pay for service usage of others. Naturally, the client is also responsible for issuing a separate certificate signed with the original identity that authorises the ephemeral identifier to use the TTP certificate (see Figure 6). There should be an expiry time as well. This kind of scenario then requires that the certificates are encrypted, so that the correspondence of identifiers is not evident to the outsiders. Obviously, this does not provide anonymity towards the service provider.

```
(
 (cert
  (subject (hash sha1 <hash value>))
  (issuer (hash sha1 <hash value>))
  (validity
   (not-before 2008-07-30_08:00:00)
   (not-after 2008-07-30_09:00:00))
 )
(signature (rsa-sha1 <sig>))
)
```

Figure 6. Example of delegation certificate

The client should pay attention to the validity times and authorised amounts in the certificates, so that it has valid authorisation available, if the service requests new negotiation after the previous hash chain values have been used up to the specified maximum. At this point there is no need to do the whole base exchange again and the parties can take advantage of the HIP control packets to update the association.

3.4. Hash token handling

HIP UPDATE packets are used to transmit the next hash chain value, when it is due. This requires additions for HIP specifications. That is, a new HIP parameter needs to be defined, such as depicted in Figure 7, which identifies the used hash chain and the next value. The UPDATE must also be acknowledges with the corresponding ACK packet in order to make sure that the packet has not been lost. For added security, the parameter should be encrypted, so that someone else cannot capture the hash value and use it to pay for its own service. Naturally, the server should detect this kind of case and prevent the use as it knows which chain is related to which client, but to some less scrupulous servers just the acquisition of the token can be enough.

Type (15 bits)	C	Length (16 bits)
Hash chain id length (16 bit)		Hash value length (16 bit)
Hash chain id (variable length)		
Hash value (variable length with possible padding)		

Figure 7. HIP parameter to convey hash chains

Alternative approach would be to integrate the transmission of hash chain values into the transport protocols, e.g., IPv6 headers could be used in the use case described below. However, this would require making similar modification to every transport case for which the non-repudiation mechanism was applied. Clearly, it is easier to use more general approach with the available HIP update mechanism.

When the service wants to cash in the tokens, it contacts the third party in question and presents the given offer, the response and the relevant authorisation certificates. Also, the amount of tokens and the last received token value are submitted, so that the third party can verify the correct amount of used hash chain tokens. The third party compensates the service provider and at later point presents a bill to the client.

4. Use cases

Here we discuss potential use cases for the suggested NoRSU method. The cases presented deal with network access and streaming services.

4.1 Network attachment

The network attachment scenario is very much the basic use case for NoRSU. Thus, the idea is to "pay" one's net usage with the exchanged tokens and prove that one is authorised to receive service. This could mean, for instance, allocation of certain amount of transmitted bytes per time unit.

In the network setup we assume that we have an access point (or possibly several of them) and an access point controller, which also functions as a gateway to the external networks. The user makes the

initial attachment to the access points, but the actual base exchange is run with the access point controller. The setup resembles the architecture given in [14], which allows even the link level frames to be transmitted to the controller. Note, however, that the access points still can exhibit enough intelligence to check the validity of the puzzle solution, so that the invalid packets do not even reach the controller, hence further mitigating the denial of service concerns.

Even though we are working in the access domain and discuss mainly the interaction between the user and the access controller (corresponding the client and the server of the previous section), one could also device ways to include the home domain into the online transaction. For instance, a setup envisaged in [7] could be one alternative.

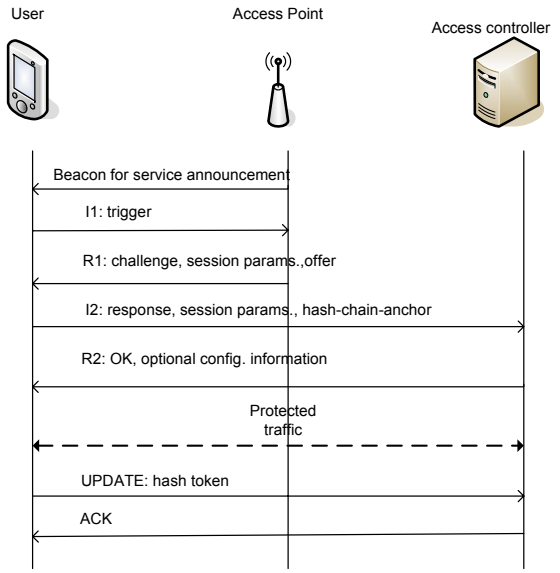

Figure 8. Attaching to network using non-repudiable service usage

The message exchange between the parties is depicted in Figure 8. We envisage the access points to be intelligent ones, so that they are able to respond to the initial I1 message with the precalculated R1 message, which also contains the offer for the network access use in the form of an SPKI certificate (see example in Figure 9). Naturally, there has been previous communication between the access point and the controller regarding the contents of R1 messages.

The user's response comes in the I2 message and it is forwarded to the controller in case the puzzle solution is correct. An example of the attached certificate is given in Figure 10. Once the controller has accepted the user as valid communication partner, R2 message is sent as an acknowledgement of the transaction as in typical base exchange. The accounting part is implemented with the help of HIP UPDATE packets as depicted earlier.

```
(
 (cert
  (issuer (hash sha1 <hash value>))
  (offer (time 1 (s 60)))
  (validity (not-after 2008-10-30_12:00:00))
  )
 (signature (rsa-sha1 <sig>))
)
```

Figure 9. Example of offer certificate

```
(
 (cert
  (hash-of-offer sha1 <hash value>)
  (hash-chain-anchor sha1 <anchor value>)
  (issuer (hash sha <hash value>)
  )
 (signature (rsa-sha1 <sig>))
)
```

Figure 10. Example of response certificate

While one might make the assumption that the network setup ensures that no traffic hijacking or redirecting can take place, it is not for certain in all environments. So, there is need to bind the actual traffic to the used identities and hash chains. The binding to the negotiated association could be done either on link or network layer, for which the base exchange provides keying material.

As discussed in [14] the link layer security can be extended all the way to the controller, which makes it transparent to the user from the network layer point of view. The similar kind of link layer setting is envisaged in the network attachment procedure described in [15] and it is based on the similar HIP alike protocol.

On the network layer the binding to the actual traffic can be done with the help of IPsec. In other words, the participants also establish IPsec association during the base exchange. As the same keying material is used to for the association setup, the binding to the tokens can be ensured. However, this basically requires that the user tunnels all the traffic to the controller that imposes extra overhead. This is similar as is envisaged to be done with Protocol for carrying Authentication for Network Access (PANA) based IPsec access control solution [16], even though key management solution is different. Using modified transport mode ESP might be one solution, but it violates the original end-to-end idea of it and requires changes to the packet processing at the controller side, i.e., it has to first process (and

remove) ESP part before forwarding the packet towards its final destination. Additionally, if the end user wishes to setup HIP associations with other hosts, one need to make sure that there is no Security Parameter Index (SPI) collisions with the existing association with the controller. Similar concerns touch Bound End-to-End Tunnel (BEET) mode [17]. Thus, tunnel mode and link layer approaches provide more feasible approach. Also, they have the possibility of protecting the privacy of the user in terms of with what other nodes it communicates. The actual binding is negotiated during the base exchange.

4.2 Accessing streaming service

Here we consider the case where a client wishes to access a streaming service provided by the server. This could be a multimedia service, such as downloading a song or a video, which is using Real-time Protocol (RTP) for executing the transport of streams [18]. Note that generally the stream is described beforehand, for example, with Session Description Protocol (SDP), but here we only concentrate on the transport part.

RTP itself provides little security, so in order to make the strong binding to the actual negotiation, we use the secure profile of RTP, i.e., Secure RTP (SRTP), which provides integrity and confidentiality services along with replay protection [19]. While using IPsec with real time traffic might be an option as well, the added latency and jitter can degrade the quality performance of such solution significantly [20]. However, many current tools might still favour IPsec due to more tried and interoperable key management.

RTP is a framework that is intended to be extensible enough to allow easy creation of profiles to meet the requirements of applications requiring transport of different kinds of real-time data, e.g., Voice over IP (VoIP). It consists of two different protocols: RTP for transporting the actual data and RTP control protocol (RTCP), which is used to report the characteristics of the connection, such as the quality of service, and convey information about the participants [18]. Hence, in very simplified terms one can consider RTP to be flowing from the server to the client and RTCP from the client to the server. Note, however, that RTP is intended to be applicable to multicast scenarios as well, although in our discussion we are concentrating on unicast transmission as HIP associations are mutual. HITs could be applied in multicast solutions, though.

There exists some work that has discussed the integration of SRTP with HIP [21], and that is the basis of this use case. Basically, the idea is to bind the RTP stream to the negotiation that has taken place within the base exchange. As SRTP leaves the question of key management open, we can use the HIP mechanisms to create the common master secret that can be used to establish the required session keys for the real-time session. [21] defines the additional parameters that have to be included in the HIP base exchange in order to achieve this, although the definitions are not yet complete. The modified protocol flow is depicted in Figure 11. Alternative is to run the offer-response interaction in the base exchange and then do the SRTP negotiation after that using the UPDATE packets. In any case, the UPDATE packets are used for re-keying.

The use of SRTP parameters provides the participants an agreement about the used encryption and authentication algorithms and their corresponding key lengths. Also, key derivation function is agreed, so that session keys can be derived from the master key and master salt. The salt is also exchanged, but the key is extracted from the keying material that is created during the base exchange (an index to the keying material can be provided). Other RTP specific parameters can be exchanged as well, such as those indicating the synchronisation source for identifying the participant and rollover and initial sequence numbers for packet indexing.

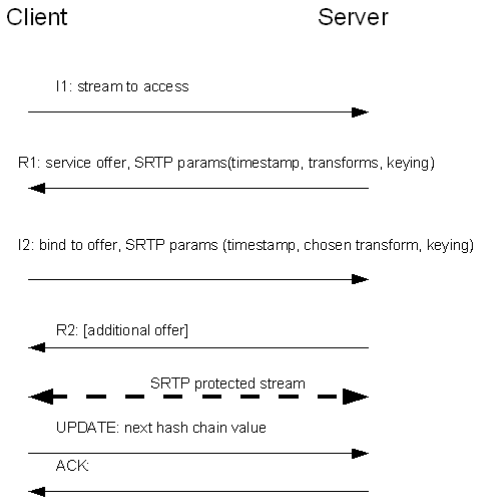

Figure 11. Using NoRSU with SRTP

However, we still need to add the non-repudiation property to this solution according to our suggested mechanisms. Thus, the I1 message already signals the client's intention to access a streaming service, so the R1 message can contain the corresponding response, which gives both the service offer and the proposed SRTP parameters. I2 contains the selected SRTP parameters along with the client's response to the offer. R2 does not contain any additional SRTP data. After

the exchange the both parties have an understanding about the frequency of the release of tokens.

During the service use the client needs to transmit the tokens to the server and this can be done using the same HIP UPDATE packets as described in the previous section. Another option that would integrate accounting more tightly with the stream itself would be to use the reporting functionality of RTCP, or more precisely, Secure RTCP (SRTCP), which provides the same services for RTCP as SRTP provides to RTP [19]. However, RTP philosophy does not take into account the acknowledgement of packets as loss of a single packet is not consider that important. Thus, detecting loss of packets transmitting the hash tokens would end up increasing the complexity. Also, as mentioned previously, we wish to prefer the general approach with the employment of HIP UPDATE.

So, in this use scenario we have described how a streaming service could take advantage of non-repudiation. The stream is strongly bound to the used identities, because the keying material used to protect the multimedia stream is derived from the negotiation done during the base exchange. That, in turn, translates to the used identities.

5. Implementation restrictions

In terms of packet size, overloading of HIP messages faces some challenges. This section talks about the relevant restrictions.

5.1 Available space in frames

While using the previously described certificates for authorisation is somewhat straightforward, one has to remember that the usage of HIP sets some restrictions. Mainly, due to the defined packet format, the length of HIP parameters is restricted to 2008 bytes, which has to accommodate the mandatory base exchange parameters, as well [2]. Also, the HIP headers (being logically IPv6 extension headers) are specified as unfragmentable, i.e., the IPv6 implementations are not allowed to fragment the header in order to meet the maximum transmission unit (MTU) of packets. This is mainly intended for avoiding DoS attacks caused by invalidly fragmented packets.

One additional limitation, i.e., around 1500 bytes, for MTU could be the use of Ethernet links on some section of the paths, but the situation can be actually worse than that. IPv6 specification states that the minimum IPv6 implementations are allowed to assume 1280-byte MTUs [22], and according to the survey done in [23] many of the current IPv6 paths seem to use it (well over 40% of the surveyed paths). Thus, length restriction due to small MTU cannot be ignored. Naturally, in environments that support higher MTUs or link layer fragmentation, like in many wireless technologies, the requirements are more relaxed, but one still needs to consider the whole path. Note that HIP is intended to be usable with IPv4 as well, where minimum MTU requirements are different, but focus of our discussion is on IPv6.

If one considers the typical HIP base exchange, it is quite obvious that the most of the information content is in R1 and I2 messages. Hence, they are the most restrictive ones for our purposes. Unfortunately, they are also the messages that will be carrying our extra payloads, i.e., offers and responses. As the amount of bytes changes from application to application, it is not easy to tell the exact amount needed for each messages, but looking at the HIP specifications and the available base exchange parameters, one can make an estimation of the used bytes. Table 1 presents mandatory HIP parameters for R1 and I2 messages along with a couple of most likely optional ones (R1_counter for indicating the current generation of the puzzle and Echo for echoing the transmitted value back), which are used to enhance the security properties of the protocol.

Table 1. Estimated sizes of R1 and I2 messages

Parameter	R1 bytes	I2 bytes
[R1_counter]	16	16
Puzzle	16	
Solution		24
Diffie-Hellman[1]	200	200
HIP_transforms	16	8
Host-id[2]	250	
Encrypted host-id[3]		280
[Echo_request/response]	20	20
HMAC		24
HIP_signature	140	140
Total	658	712

[1]Assumed just one 1536-bit D-H group (up to two groups could be proposed)
[2]1024-bit RSA key with 100 bytes domain part (which is optional)
[3]Encrypted using 128-bit AES-CBC (encryption is not mandatory)

As mentioned, table figures are just an estimation, which takes into account the "typical" parameters. The situation would be quite different, if one were to require, for instance, larger Diffie-Hellman groups and

longer keys in the name of better security (which is quite understandable, for instance, in the case of long term keys). Additionally, one might need extra parameters to signal additional associations, such as the case when negotiating the use of IPsec ESP for subsequent traffic.

As discussed in [9] the key types have significance as well. In the table above we assumed the use of RSA keys, but HIP also allows employing DSA keys. RSA and DSA keys provide roughly the same level of security for similar key lengths, but the RSA key generally has a shorter representation, because with DSA you basically also have to transmit domain parameters (this can be avoided in some special scenarios, though) [24][25]. However, the DSA signature takes less space than the corresponding RSA signature, which is dependant on the key size.

When considering these two things together, one can come to a conclusion that if one has to transmit both the key and signature, the RSA is more optimal solution length-wise. However, if it is required that only the signature is transmitted, DSA is a better alternative. Thus, this leads to a conclusion that within our context RSA is better suited for host identities, whereas the trusted third parties could use DSA keys to generate the signatures for the certificates. It is, after all, assumed that the parties have pre-established trust relationships with the trusted third party and know their relevant public keys.

The most optimal solution for this case would be elliptic curve cryptography (ECC), because it offers shorter key sizes and its performance is comparable to DSA [26]. Currently, however, HIP does not specify the possibility to use ECC keys for host identities. This should be a viable research direction for the future, especially considering the increase in key lengths over time.

5.2 Encoding

As discussed in the previous subsection, our working environment is somewhat restricted when it comes to the length of the messages. Thus, there is also need to consider the encoding of the embedded certificates. The previous examples were given using S-expressions, which, while human readable and good for examples, are unsuited for transmitting on the wire. For instance, the signatures and other binary data could be presented with base64 encoding, which clearly is wasteful when it comes to the used space.

SPKI drafts define the possibility to use canonical S-expressions, which aim at more efficient packing of the information [27]. It is also a form, which is expected to be used, when doing operations, such as hashing, on expressions. It basically presents the expressions as binary byte strings, i.e., octets, and precedes every token with the length value. This allows presenting binary data in a concise way, but still the textual tokens use the space inefficiently. In Figure 12 we present an example of canonical form of the delegation certificate given in Figure 6 (binary data omitted and line breaks added for readability). Using this encoding the length is reduced by around 60 bytes, but still the size of the certificate is around 350 bytes.

```
((4:cert(7:subject(4:hash4:sha120:<omitted>))
(6:issuer(4:hash4:sha120:<omitted>))
(8:validity(10:not-before19:2008-07-30_08:00:00)
(9:not-after19:2008-07-30_09:00:00)))
(9:signature(8:rsa-sha1128:<omitted>)))
```

Figure 12. Example of canonical encoding of S-expression (formatted for readability)

However, there is some work that considers binary encoding of SPKI certificates and that would suit our purposes as well. A SPKI authorisation certificate presented in [28] took a little over 250 bytes. It contained hashes of the issuer and subject public keys, validity dates, a simple attribute and a DSA signature, which actually could be made even more concise. In [28] it took 190 bytes, because it also contains the public key of the certifier (without domain parameters). Thus, if one makes the representation of the signature more concise, it is possible to save over 100 bytes. After all, HIP base exchange already conveys the public keys.

Table 2. Examples of records of the efficient encoding scheme

Expression type	Implied size in bytes
Subject 1024 bit rsa public key hashed using sha1	20
Subject expressed with HIT	16
Valid end date	4
Valid start and end date	8
Signature 1024 bit rsa using sha1	128
Signature 1024 bit dsa using sha1	40
Signature 2048 bit dsa using sha1	56

So, if we take into use similar kind of encoding that only has a 2-byte type field and a variable length value field per one record. The length is expected to be implicit based on the type. The type encodes much of the expression itself, e.g., we might have different

types for issuers expressed with HITs or just with SHA-1 hash values, i.e., multiple textual tokens are reduced. Also, some expressions, such as validity times can be reduced to more concise form by encoding multiple values into a record and using seconds to express dates. Thus, we are driving towards utmost efficiency at the expense of flexibility. Table 2 shows an example of some encoded expression types and the implied size of the following value field.

Table 3 shows how many bytes different certificates could take using this efficient encoding. When comparing, for instance, the efficient encoding of delegation certificate to the canonical encoding, the reduction is almost 50%. Note that encoded values could contain additional structure inside them, e.g., the encoding of the actual offer expression takes into account values such as the amount of hash tokens needed, the used unit, and the amount of units. So, one should be able to express things like 1 hash token per 60 seconds or per thousand kilobytes. A more complex offer would include the possibility to point to an external offer, which could be an XML document giving more details, but in the access scenario the client ought to be able to access the said document, i.e., have connectivity before connectivity service has been agreed on. We are, however, aiming for simplicity.

5.3 Summary of restrictions

When we consider the previous discussion regarding MTU and consumed bytes in HIP parameters, it is obvious to question, whether the suggested certificates can be included within the HIP messages. Taking into account the figures in Table 1 and amount of bytes needed for IPv6 and HIP fixed headers (both take 40 bytes), it can be concluded that with a safe margin one can use roughly 400 bytes for certificate information (R1 can contain bit more). One should also not forget the HIP parameter for signalling the non-repudiation and the target service. In a case of simple service naming (like a hash), the parameter would take 32 bytes, but with other encodings it naturally could be larger. There is room for optimisation, though. If we were to leave out the domain part of the host identity, which basically can contain Fully Qualified Domain Name (FQDN) or Network Access Identifier (NAI), we can save around 100 bytes compared to the given figures. Considering that we are planning on giving explicit authorisation for the used identities in the form of certificates, the dropping of domain part is not so crucial.

Now, if we also look at the data given in Table 3, we can come up with estimates for the amount of data added due to the certificates. If we first consider R1 message, one can expect that it contains an offer for the service, but also an authorisation issued to the server by a TTP. The table actually just gives figures for client authorisation, but the amount of needed bytes is similar. Thus, those two certificates fit within the constraints given. For I2 message one needs the response and also the authorisation ensuring the liability of the client and this should not be a problem, either. However, if we want to support the advanced scenario, where the right to use the service is delegated to another entity (or, in case of privacy protection, to another identifier of the same entity), we are hanging on the very edge of our constraints. Thus, implementations would need more care in such circumstances. The negotiation of key management procedures for additional protocols, such as those depicted in the use case of SRTP, might have to be postponed to the UPDATE messages after the base exchange has completed.

If such additional roundtrips are undesirable, there is still room for further optimisation in the used certificates. As the HIP packets already contain signatures of the client and the server, the end point generated certificates for offers and responses can do without signatures. This saves well over 100 bytes (in case of RSA signatures) and enables one to fit all the authorisation statements within the limits we have set. The downsides of this approach are increased storage requirements and added complexity, because the parties need to store the whole HIP packets instead of a set of certificates. The clearing party has to also be able to understand HIP structures.

It is worth noting that the previous discussion assumes 1024-bit keys, whereas longer keys make it even harder to fit the information within the HIP

Table 3. Byte count for different certificates

Offer	bytes	Response	bytes	TTP-client	bytes	Client deleg.	bytes
Issuer	22	Hash-of-offer	22	Subject	22	Subject	22
Offer	6	Chain-anchor	22	Issuer	22	Issuer	22
Validity-end	6	Issuer	22	Target serv.	22	Validity-range	10
Rsa-sign-1024	130	Rsa-sig-1024	130	Amount-max	6	Rsa-sig-1024	130
				Propagete	2		
				Validity-range	10		
				Dsa-sig-2048	56		
Total	164		196		140		184

header. It should be noticed, though, that the constant increase in computing power and the developments in the mathematical algorithms is bound to raise the bar for the required key lengths. The 1024-bit keys are considered to be adequate for the next couple of years, but scenarios needing longer term solutions, such as those related to the trusted third parties, should already use 2048-bit keys [29]. This further motivates the need to look into the possibility of using ECC keys.

One additional length consideration relates to the use of hashes. Even though SHA-1 is a very common hash function, it is showing some weaknesses [30]. Therefore, one should also consider the use of advanced forms, such as SHA-256, instead in places that require hashing of relatively free form messages. The increase in length is just 12 bytes, though, but can build up when used in multiple places Note, though, that when hashing public keys and the attacker wants to find another key that hashes to the same value, it is very unlikely to find such a value, because one is not able to modify the source value at will and still retain the required structure for a public key. This also applies to the case of HITs, even though they are shorter than SHA-1 hash values. The case where this matters most is the hashing of the offer of the server to indicate to which offer the client is binding itself.

6. Analysis

The following subsections analyse the potential threats and the corresponding countermeasures from the viewpoint of our proposal.

6.1 Threats within the context of the solution

In this section we discuss the potential threats that can emerge in an environment that plans on adopting the suggested token based solution. One should note that not all of them have a technical countermeasure, but those should then resort to other measures offered by the society in case of agreement dispute, such as litigation. While this subsection concentrates just on listing the threats, the way the countermeasures take place is discussed in the next one.

As has been described earlier the interaction is mainly between the user and the service, but from the threat analysis perspective one has to also remember the existence of a third party, such as the home operator, who acts as a trust and liability broker between the entities. There might also be an external entity, who could try to interfere with the service provisioning.

In the case of compensation of the service usage, the setup can pose several threats to different parties. The most obvious threats are that the service is not paid for or that the service is not received after paying. Especially when the user pays after the service usage (post-paid), there is a chance that the user repudiates it, i.e., claims that he has never used the service and the cost claims are unfounded.

Different collusion scenarios can be envisaged. In other words, two of the parties conspire against the remaining one. The home operator could assure the trustworthiness of the user without any intention of compensating the service afterwards at the time of the clearing. On the other hand, the user and the service could collude against the home operator in order to make the home operator compensate the service without the user having no intention of paying his bill later on. While being perhaps the most unlikely case of these, the service and the home operator could try to make the user pay more than the user originally thought (misleading advertising is another matter).

Double spending can occur when otherwise valid tokens are replicated to pay for several different transactions. In a similar sense, the service might try to charge the accessed service more than once. Very close is also the case of overspending, when the user consumes more resources than he can afford, i.e., overly large amount of tokens is created and used.

Hijacking of information by an unauthorised party could also take place. The payment tokens or the actual payment could be stolen by some other service or another user could try to use the tokens to pay for his service. Also, instead of tokens, another user could try to hijack the paid service from the legitimate user.

Integrity of the compensation agreement could be facing threats, as well. Either the user or the service provider might try to modify the agreed terms, so that the later claims would be more favourable to them. This also includes forging of additional tokens so that the service could claim more resource usage than really took place.

User privacy is always an existing threat in any communication system and it will become even more important as the transition towards ubiquitous communication takes place. This is especially evident in our proposed solution, which makes heavy use of different kind of identities. User privacy is at stake if the identity information is disclosed to unauthorised parties, who are then able to track the users, for instance. Also, users may also wish to prevent others from learning what sort of services they are using and what sort of usage patterns they follow. Thus, the users should be in control of the disclosure of information about themselves and their actions.

As mentioned above, within the limits of our proposed solution, some of these threats can only be addressed through litigation. For instance, if some party refuses to pay even when faced with technical evidence, the other parties have to initiate legal procedures in order to get the promised compensation. This is not, however, different from the case, where a user refuses to pay his post-paid subscription or credit card bill. This is a business risk, which should be embedded in the business models of the players.

Generally, when faced with collusion of other parties, the legal action with the technical evidence is the only solution. Naturally, fear of losing one's reputation can be enough disincentive for the home operator to not to cheat the user as the user trust is the very foundation of its business model. In the following section we analyse the properties of our proposal, which can provide solutions to the technical threats presented above.

6.2 Technical measures against the threats

The basic components in the proposed solution are the use of hash chains and the binding of the identities to them. The hash chains provide the means to pay for the service usage in a piecemeal fashion, i.e., as long as the service is received, additional hash chain values can be submitted. Analogously, as long as the server keeps receiving new hash values that are part of the chain, the service is provided. Thus, in case of malicious party, no further compensation or resource provisioning is provided, i.e., the granularity of the commitment is better controlled. Generally, the value of a single hash token should be kept small as that is the amount that can be lost in the case of misbehaviour.

The hash chain values have the added benefit of being easily verifiable, because the receiver has to only compute one hash function in order to make sure that the received value is part of the chain. Naturally, the used hash function has to be secure enough, so that the receiver is not able to calculate future values. This ensures that only the entity that has knows the secret seed of the hash chain knows the transmitted values beforehand. Hence, the service provider cannot easily create additional tokens, so that it could make cost claims for unused resources. Also, as the value of single token is kept small, the required effort of brute force attack clearly outweighs the benefit.

Non-repudiation property of the solution comes from the binding of the identities to the presented offers and responses. When the user presents the anchor value of the hash chain, he has signed the statement with his identity and also included in the statement the reference to the received offer. This, along with the assumption that the hash function is irreversible, dictates that only that user has been able to create the said hash values. Thus, if the user denies using the service, the service provider only needs to present the anchor value signed by the user and the last received chain value in order to prove that the user has used service with the offered terms. The service provider can naturally deny that it has provided any service, but from the point of view of our solution concept it does not matter as the user already has consumed the desired resources. The service, however, cannot deny that is has given a service offer on certain terms.

The trust to the client's ability to pay comes from the associated TTP certificate, which authorises the client to use a certain maximum amount of commodity. This can be seen as the credit the client has in the eyes of TTP and as an acknowledgement that TTP knows the client. This way the server has certainty that someone will provide compensation for the provided resources, because ultimately TTP has accepted the liability in the case of misuse when issuing the certificate to the client. It is then up to the agreement made between the third party and the client to settle the costs. This does not differ from the typical post-paid business model commonly used in the telecom or credit card industry.

It is also possible to grant an authorisation certificate for the service as well, to be presented as proof of its trustworthiness. Although, as stated above, in case the server is not providing the service it promised to deliver, the client just can stop sending any hash chain values. If the service in question is other than the typical access scenario, like buying a song, then the motivation to include such authorisation might be different. This is basically a risk management decision for the client. Of course, if the song, for example, is streamed, then the client has better control of what it is receiving and can pay it piecemeal, like in the second use case scenario described earlier.

The employment of HIP provides a natural way of taking advantage of the accompanying cryptographical identifiers for presenting the identities of the parties. As it also provides a key management solution, it can be used to create the necessary association so that the actual data traffic can be bound to the same identities, even though it requires an existing data traffic protection mechanism, such as IPsec. Thus, even though IP addresses could be spoofed, the mutually agreed keying material ensures that the traffic is useful only to the valid partners.

Fine tuning of the solution is done through the extra attributes given in the certificates and the procedures the parties conduct during the transaction. When the client clearly states the service provider identity in the response message, no other service provider can claim the costs, even though it somehow could manage to get hold of the hash tokens. It is possible for the home operator to give more granular authorisations to the client and only allow certain service providers or service types. One should remember, though, that if the same authorisation allows the use of several service providers for the specific service type, the maximum allowed resource consumption could not be controlled without online access to the home operator, which tends to complicate things and decrease performance. However, this can be found out during the clearing procedure and extra claims made towards the user. This is basically a risk management decision for the home operator, when deciding what sort of granularity to use in the issued authorisations.

In any case, the server has to remember to check the provided anchor values, so that it is not possible for the client to use the same anchor value within the validity period of the same offer. Otherwise the client might be able to use the same hash chain values again, but the server would only be able to bill them once. This could also be prevented by having an individual session identifier in every R1, but cannot be done without breaking the basic HIP properties as the signature in R1 is pre-computed for the sake of mitigating denial of service possibility.

The privacy of the client is preserved through the decoupling of authentication and authorisation. TTP issued certificate can state that the ephemeral identifier assigned to the client is trustworthy for certain actions. Naturally, TTP is able to connect this to a real identity. Also, this introduces additional overhead in terms of additional interaction with TTP, especially if the

Table 4. Threats and possible countermeasures

Technical threats	Preventive measures	Threats handled through litigation
User denies having used the service	Binding of identity to the hash chain	• User and home operator collude against service • Server and home operator collude against user • User and service collude against home operator • User is charged too much at the time of clearing • User refuses to pay at the time of clearing • Service does not get money from the home operator at the time of clearing • Privacy of the user (e.g., home operator releases information about the user without user consent)
Service provider does not provide agreed service	Stop transmitting additional hash tokens	
User gets no or other service that he paid for	Stop transmitting additional hash tokens	
Service hijacked by another user	Bind the payloads to the negotiated keying material	
Intercepted tokens used by another user to pay for his service	Service needs to ensure the strong binding between the identity and the hash chain, protection of tokens with the negotiated association	
User double spends the created compensation tokens	Authorise specific service, check anchor value uniqueness	
Service charges user multiple times	Non-repudiable accounting records are accepted only once	
Other service "cashes" the tokens	User authorises specific provider	
Service creates additional valid user tokens	Secure hash function prevents creating additional usage records and user signature protects the hash anchor value	
User modifies the offer to more favourable one	Offer is protected with signature	
Service modifies the offer to more favourable one	Offer is protected with signature	
User overspends his credit	User is authorised only to spend certain maximum amount	
Privacy of the user	Use of ephemeral identities and delegation	

identifier is changed often. This allows providing anonymity towards the service providers, though. The other option is that the TTP provides an authorisation for the long term identifier and the client constructs an ephemeral identifier for which it delegates the authorisation. Even though this is a more flexible option, it does not provide complete anonymity towards the service providers, because they need both the delegation and the original authorisation. However, it does, like the other alternative, provide privacy protection against external observers, because the real identity does not have to be visible, not even in the form of its HIT.

In Table 4 we have summarised the different kinds of threats that might emerge in this kind of service usage concept. Additionally, the table present how the suggested solution can answer to the technical threats.

7. Conclusion

In this paper we have proposed a simple accounting scheme to be used in conjunction with HIP. With his kind of solution the service provider is able to get undeniable evidence that it is entitled to compensation for the provision of its resources to a certain client. On the other hands, the client can control the charging procedure, so that it is only billed for the costs that are based on the actual usage.

We have showed that the combination of HIP and hash chains can provide a secure solution for non-repudiable service usage that also takes into account the binding to the actual data traffic. This is further enhanced with the employment of authorisation certificates to increase the level of trust the client and server have for each others. Thus, it also provides an authorisation mechanism for the participants.

However, the used environment poses some problems, namely in the form of length restrictions, that need to be taken into consideration. We have considered the most common IPv6 path MTU, 1280 bytes, and concluded that even though it is possible to introduce the solution to this environment, the advanced scenarios providing better privacy support and the delegation of service authorisation may face difficulties with certain implementations. There is a possibility for length optimisation, although at the expense of increased complexity. Also, the choice of used key types has considerable impact on that and RSA based host identities are better suited for the most length restricted cases. This still calls for efficient encoding mechanisms, which have the downside of limiting the flexibility.

It is worth remembering, however, that the wide adoption of HIP is still years away, so the restrictions set by the current MTUs can be quite different in the future networks. It shows, though, that this is additional incentive for pushing for higher MTUs. Also, the research done with technologies that provide shorter key lengths, such as ECC, provides measures to answer to these restrictions, even though the requirement for having larger key sizes goes hand in hand with the increase in computing power. In any case, the host identity enabled environment provides many interesting directions for the development of secure charging schemes.

Acknowledgment

The author wishes to thank Tuure Vartiainen and prof. Jarmo Harju for comments and suggestions.

References

[1] Federal Communication Commission, "Unauthorized, Misleading, or Deceptive Charges Placed on Your Telephone Bill - Cramming", FCC Consumer Facts, online article, available in http://www.fcc.gov/cgb/consumerfacts/cramming.html (accessed 01/2009), Jul 2008.

[2] Moskowitz R., Nikander P., Jokela P. (Ed.), Henderson T., "Host Identity Protocol", IETF RFC 5201, Apr 2008.

[3] Moskowitz R., Nikander P., Jokela P., "Using ESP transport format with HIP", IETF RFC 5202, Apr 2008.

[4] Heer T., Varjonen T., "HIP Certificates", IETF Internet-Draft draft-varjonen-hip-cert-01 (work in progress), Jul 2008.

[5] Laganier J., Vicat-Blanc Primet P., "HIPernet: A Decentralized Security Infrastructure for Large Scale Grid Environments", The 6th IEEE/ACM International Workshop on Grid Computing, Nov 2005.

[6] Tschofenig H., Nagarajan A., Ylitalo J., Shanmugam M., "Traversal of HIP aware NATs and Firewalls", IETF Internet-Draft draft-tschofenig-hiprg-hip-natfw-traversal-02, expired, Jul 2005.

[7] Heikkinen S., Priestley M., Arkko J., Eronen P.,Tschofenig H., "Securing Network Attachment and Compensation", Proceedings of the Wireless World Research Forum Meeting (WWRF#15), Nov 2005.

[8] Blaze M. et al., "TAPI: Transactions for Access Public Infrastructure", Proceedings of Personal Wireless Communications (PWC2003), Sep 2003.

[9] Heikkinen S., "Non-repudiable service usage with host identities", Proceedings of the Second International Conference on Internet Monitoring and Protection (ICIMP07), Jul 2007.

[10] Lamport L., "Password authentication with insecure communication", Communications of the ACM, vol. 24, no. 11, 1981.

[11] Tewari H., O'Mahon D., "Multiparty micropayments for Ad Hoc Networks", Proceedings of the IEEE Wireless Communications and Networking Conference, Mar 2003.

[12] Zhou J., Lam. K., "Undeniable Billing in Mobile Communication", Proceedings of 4th ACM/IEEE International Conference on Mobile Computing and Networking, Oct 1998.

[13] Nikander P., Laganier J., Dupont F., "An IPv6 Prefix for Overlay Routable Cryptographic Hash Identifiers (ORCHID)", IETF RFC 4843, Apr 2007.

[14] Calhoun P. et al., "Light Weight Access Point Protocol", IETF Internet-Draft draft-ohara-capwap-lwapp-04 (work in progress), Mar 2007.

[15] Rinta-aho T. et al., "Ambient Network Attachment", Proceedings of 16th IST Mobile and Wireless Communications Summit, Jul 2007.

[16] Parthasarathy M., "PANA Enabling IPsec based Access Control", IETF Internet-Draft draft-ietf-pana-ipsec-07 (work in progress), Jul 2005.

[17] Nikander P., Melen J., "A Bound End-to-End Tunnel (BEET) mode for ESP", IETF Internet-Draft draft-nikander-esp-beet-mode-09 (work in progress), Aug 2008.

[18] Schulzrinne H., Casner S., Frederick R., Jacobson V., "RTP: A Transport Protocol for Real-Time Applications", IETF RFC 3550, Jul 2003.

[19] Baugher M., McGrew D., Naslund M., Carrara E., Norrman K., "The Secure Real-time Transport Protocol (SRTP)", IETF RFC 3711, Mar 2004.

[20] Bou Diab W., Tohme S., Bassil C., "Critical vpn security analysis and new approach for securing voip communications over vpn networks", Proceedings of the 3rd ACM workshop on Wireless multimedia networking and performance modeling, Oct 2007.

[21] Tschofenig H., Shanmugam M., Muenz F., "Using SRTP transport format with HIP", IETF Internet-Draft draft-tschofenig-hiprg-hip-srtp-02 (expired), Oct 2006.

[22] Deering S., Hinder R., "Internet Protocol, Version 6 (IPv6) Specification", IETF RFC 2460, Dec 1998.

[23] Wang Y., Ye S., Li X., "Understanding Current IPv6 Performance: A Measurement Study", Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC'05), Jun 2005.

[24] Eastlake D., "RSA/SHA-1 SIGs and RSA KEYs in the Domain Name System (DNS)", IETF RFC 3110, May 2001.

[25] Eastlake D., "DSA KEYs and SIGs in the Domain Name System (DNS)", IETF RFC 2536, Mar 1999.

[26] Cronin E., Jamin S., Malkin T., McDaniel P., "On the Performance, Feasibility, and Use of Forward-Secure Signatures", Proceedings of the 10th ACM conference on Computer and communications security, Oct 2003.

[27] Ellison C. (Ed.), "Simple Public Key Certificate", IETF Internet-Draft draft-ietf-spki-cert-structure-06.txt, expired, Jul 1999.

[28] Arbaugh W., Keromytis A., Farber D., Smith J., "Automated Recovery in a Secure Bootstrap Process", Internet Society 1998 Symposium on Network and Distributed System Security, Mar 1998.

[29] National Institute of Standards and Technology, "Recommendation for Key Managent - Part 1: General (Revise)", NIST Special Publication 800-57, May 2006.

[30] Wang X., Yin Y., Yu H., "Finding Collisions in the Full SHA-1", Proceedings of Crypto'05, Aug 2005.