Internet Engineering Task Force G. Montenegro
INTERNET DRAFT C. Castelluccia
July 2002

SUCV Identifiers and Addresses

draft-montenegro-sucv-03.txt

Status of This Memo

Abstract

This document addresses the identifier ownership problem. It does so by using characteristics of Statistic Uniqueness and Cryptographic Verifiability (SUCV) of certain entities which this document calls SUCV Identifiers (SUCV ID's). This note also proposes using these SUCV characteristics in related entities called SUCV Addresses in order to severely limit certain classes of denial of service attacks and hijacking attacks. SUCV addresses are particularly applicable to solve the 'address ownership' problem that hinders confidence in mechanisms like Binding Updates in Mobile IP for IPv6.

Table of Contents

1.0 Introduction

This document addresses the identifier ownership problem [ADDROWN] by using characteristics of Statistic Uniqueness and Cryptographic Verifiability (SUCV) of certain entities which this document calls SUCV Identifiers (SUCV ID's). This note also proposes using these SUCV characteristics in related entities called SUCV Addresses in order to severely limit certain classes of denial of service attacks and hijacking attacks. SUCV addresses can solve the 'address ownership' problem that hinders confidence in mechanisms like Binding Updates in Mobile IP for IPv6.

[ADDROWN] argues that there is a fundamental problem in handling operations like Binding Updates (BU's) in Mobile IP for IPv6 [MIPv6], source routing, etc) that allows hosts to modify how other hosts route packets to a certain destination. The problem is that these operations can be misused by rogue nodes to redirect traffic away from its legitimate destination. Authentication does not solve this problem. Even if a node unequivocally identifies itself, this has no bearing on its rights to modify how packets to any given address are routed. This is true even if its packets currently seem to emanate from the address in question. This last point is obvious if one considers DHCP leased addresses. It is imperative not to allow any node to redirect traffic for a DHCP address for which it held a valid lease previously. This would allow it to hijack traffic meant for the current valid user of the address in question. Hence, protection against hijacking of valid addresses requires cryptographic authorization for operations that modify routing (BU's, source routing, etc). One way to show authorization is by showing that the requesting node owns the address for which routing information is being altered. Quoting [ADDROWN]:

> Currently there exists no specified mechanism for proving address ownership in Internet-wide scale.

This document proposes a solution to the address ownership problem.

2.0 Terminology

This Section presents the notation used throughout this paper.

prf:
Pseudo-random function. SUCV mandates the use of the

HMAC-SHA-1 construct of the keyed hash function HMAC [HMAC] which produces 160 bits of output. Input key is assumed to also be 160 bits.

prfT:
Pseudo-random function whose output is truncated by taking the T leftmost bits of the output. In SUCV, HMAC-SHA1 is used, so prf96, for example, would be the keyed hash function HMAC-SHA1-96 [RFC2404].

hash:
Cryptographic hash function, SHA-1 [SHA1] for SUCV.

hashT:
Cryptographic hash function whose output is truncated by taking the T leftmost bits of the output.

SUCV:
Statistical uniqueness and cryptographic verifiability, the property exhibited by the identifiers and addresses which are the subject of this study. We also use SUCV to refer to the resultant mechanism as a whole.

sucvP:
The protocol developed here, whose objectives are proof of address ownership and session key generation.

sucvID:
128 bit identifier obtained as the keyed hash output of the hash of the public key, using an imprint value as the input key.

sucvHID:
64 bit SUCV identifier used instead of the interface identifier, and combined with the routing prefix to form an autoconfigured IPv6 address [IPV6ADDR]. Obtained as the keyed hash output of the hash of the public key, using an imprint value as the input key.

MIPv6:
Mobile IPv6 [MIPv6].

MN, HA, CN, BU, BA and CoA:
Abbreviations of mobile node, home agent, correspondent node, binding update, binding acknowledgement and care-of address, respectively, as defined by MIPv6 [MIPv6]

3.0 Overview of the Proposal

We assume that we have a network in which the nodes inherently distrust each other, and in which a global or centralized PKI (Public Key Infrastructure) or KDC (Key Distribution Center) is not available.

The goal is to arrive at some fundamental assumptions about trust on top of which one can build some useful peer-to-peer communication using opportunistic security.

But in such a network, is there a default rule we can follow safely? We posit this is it:

Default Trust Rule:

Redirect operations are allowed only with addresses which are securely bound to the requesting entity.

The above rule (to be refined later) constitutes the only rule that operates by default, allowing any other more dangerous operation only if authorized by strong cryptographic mechanisms.

4.0 Statistical Uniqueness and Cryptographic Verifiability (SUCV)

In the absence of a third party, how does a principal prove ownership of its identity to a peer?

Notice that usual owner verification relies on a third party to provide this function.

In this proposal, the principal self-generates a private/public key pair. It uses material obtained via a prf based on the public key as its identity (or as part of its address) and proves its ownership by signing it with its private key. The recipient verifies the signature, and, consequently, the ownership of the identity.

4.1 SUCV ID's

It is much more practical for protocols to use fixed length identifiers (representations of identities). Because of this, we do not use the public key itself as the identifier, but material obtained via a prf of the public key.

These identifiers have a strong cryptographic binding with their public components (of their private-public keys). This is exactly the purpose that certificates have. Let's call them Statistically Unique Cryptographically Verifiable ID's, or SUCV ID's.

Because of this, once a CN obtains information about one of these identifiers, it has a strong cryptographic assurance about which entity created it. Not only that, it knows that this identifier is owned and used exclusively by only one node in the universe: its peer in the current exchange. Thus, it is safe to allow that peer to effect changes (via BU's, for example) on how this address or identifier is routed to. Notice that with publically routable addresses, this assurance is much harder to arrive at, given that the address may be 'loaned' to (not owned by) the peer in question, perhaps thanks to the good service of a DHCP server.

Despite the fact that currently there is no way to prove address ownership in the Internet, these considerations lead to the following fundamental assumption:

Default Trust Rule

Redirect operations are only allowed to affect routing for entities which have the SUCV property.

The above constitutes perhaps the only rule that operates by default, allowing any other more dangerous operation only if authorized by strong cryptographic mechanisms

What should one use: pure identifiers with no routing significance or addresses? With pure identifiers, routing information must be included somewhere else in the packet. This takes up extra space in the packet via home address options, routing headers or tunneling headers.

A major advantage to using an address is that the data traffic need not carry extra information in the packet to guarantee proper delivery by routing. Because of this it is useful to create addresses that are routable and satisfy the SUCV property: SUCV addresses.

4.3 SUCV Addresses

In IPv6, addresses that satisfy the SUCV property may be obtained as follows (as it turns out, this is very similar to

and was predated by [CAM]):

- use the top 64 bits from your routing prefix (as in rfc3041)

- define the bottom 64 bits as an SUCV ID (called the sucvHID). Use these 64 bits instead of the 'interface identifier' in IPv6 [IPV6ADDR].

The resultant 128 bit field is an identifier that is also routable, avoiding the need to take extra space in the packet (for example, by sending routing options or tunneling headers). Notice that even after moving, it is possible to reuse the 'HID' portion of the address with the new network prefix at the new location. Thus it is possible to reuse the HID with different CoA's.

Nevertheless, by snooping on binding updates, it is possible for an attacker to learn the original network prefix used by the home address. This tells an eavesdropper where this home address began to be used, and to which network it belongs, potentially important information.

On the other hand, if you use a 'pure' SUCV ID (without any routing significance), then your packets will always need extra information somewhere else to assure they are routed properly. Eavesdroppers may still know where that identity is at any particular point in time. Nevertheless, from the point of view of privacy this is a tangible improvement over always including a valid 64 bit prefix, as this divulges information about an identity's topological connectivity or under what prefix a given identity began to be used.

4.4 SUCV Derivation and Formats

This section describes how to generate an SUCV ID (128 bits), an SUCV HID (64 bits) and an SUCV address (128 bits).

First of all, the node generates a public/private key pair. Then we define the required quantities as:

```
sucvID = hmac-sha-1-128(sha1(imprint), sha1(PK))
sucvHID = hmac-sha-1-64(sha1(imprint), sha1(PK))
```

Where, imprint is a 64 bit field:

It could be a quantity that depends on the MN's location or something created by the MN itself (a random value, for

example). The objective is to use the imprint to limit certain types of brute-force attacks by limiting their applicability, or by forcing interaction with the MN.

PK: The public key is the DSA public key.

Note that according to [IPV6ADDR], the leftmost 3 bits of the sucvID can be used to unequivocally distinguish them from IPv6 addresses. Accordingly, we assume only 125 bits may be used. Additionally, bit 6 of the sucvHID (the universal/local) has to be set to zero to indicate that the sucvHID is not guaranteed to be globally unique.

5.0 SUCV Protocol (sucvP) Overview

The following protocol, sucvP, is run between a MN and an arbitrary CN. It is used:

- by the MN to prove ownership of its home address and optionally of its CoA,

- to establish an IPsec ESP security association (Skey, Lifetime, SPI) between the MN and the CN that will be used to secure MIPv6 BU's, and,

- optionally, by the CN to prove ownership of its identifier or address (only if the CN itself uses an SUCV identifier or address).

Of course, the obtained security association (SA) could also be used for any other application of ESP. However, in the basic sucvP exchange, only the MN performs proof of ownership. The section on Risks outlines the dangers this implies. Accordingly, general ESP usage should be limited to the extended sucvP exchange in which, in addition to the MN, the CN also uses SUCV for proof of ownership.

As for the choice of using AH or ESP to protect the binding updates, we chose the latter. Given the benefits of integrity and data origin authentication inherent in the proof of ownership, we believe there is no added value in using AH to protect the IP headers of BU's once a security association has been established. This and the heated debate on the future of AH convinced us to use ESP.

sucvP is functionally independent of MIPv6, and is, in fact, a separate protocol. sucvP's proof of ownership provides the

authorization for the MIPv6 BU's, but the authentication is provided by IPsec ESP. These are two separate steps which could run serially. For example, the sucvP step could be carried out over UDP (as our initial experimental implementation does), after which the ESP-authenticated BU could be sent.

However for efficiency reasons, sucvP messages might contain MIPv6 BU's (along with sucvP3).

In order for sucvP to set up an IPsec security association (including an SPI) just in time to process an ESP header and its encapsulated BU, the sucvP payload is carried as an IP protocol number (currently unassigned). Furthermore, it must precede the ESP payload used to authenticate the binding update.

5.1 Goals and Constraints

This design allows sucvP to satisfy these two objectives:

- not affect existing IPsec implementations more than absolutely necessary

- support efficient BU processing by reducing as much as possible the number of round trips.

Furthermore, we assume there is no piggybacking with the BU, so no further payload follows.

sucvP has been designed based on the following considerations:

- the protocol should not rely on a third party (i.e. a global PKI, central key distribution center, etc), although it could use one if available

- not all nodes need to use SUCV addresses, only those that wish their binding updates to be heeded (mobile nodes)

- not all nodes need to verify the validity of SUCV addresses, only those CN's that accept and handle binding updates from MN's (these CN's must use SUCV as explained below to safely populate their binding caches)

sucvP packets are exchanged directly between the mobile node and its correspondent nodes. They are not routed through the Home agent because the mobile node might be homeless or the home agent might be out of order for a certain period of time. The implications for this decision are explored below.

5.2 Packet Exchanges

The proposed protocol that a mobile host uses to send a BU to its CN is the following:

sucvP1:
The MN sends a sucvP1 message (just to initiate the exchange) to its correspondent node. This message contains a Nonce, N1. This packet may contain a MIP HomeAddress Option containing the MN's home address. The CN might sometimes need the home address to decide whether it wants to pursue the protocol exchange or not. The source address of the packet is the MN's current CoA. Additionally, SUCV supports a very simple negotiation mechanism that works as follows: Optionally, the MN can express its desire to use certain Diffie-Hellman groups (for the ephemeral DH exchange), as well as algorithms for ESP authentication and for ESP encryption.

sucvP2:
The CN replies with a sucvP2 message that contains the following: N1, Client puzzle request, Diffie-Hellman value (g^y mod p), Session_Key_lifetime. The CN may respond to any optional parameter negotiation included by the MN in sucvP1, by choosing those algorithms it wishes to support.

In order to defend against sucvP1 storms, a host might use the same Diffie-Hellman (DH) value for a period of time. The sucvP2 contains a client puzzle to prevent DoS attacks PUZZLES. Along these line, the CN may wish to ignore the optional negotiation of parameters initiated by the MN in sucvP1. In this case, the default algorithms (see below) must be used by both parties.

When the MN receives sucvP2, it verifies that the nonce N1 is the same as what was sent in sucvP1. It then solves the puzzle. At this stage of the protocol, the MN:

1. generates a DH value (g^x mod p) and derives from it and the DH received from the CN the session keys.

2. computes skey_espauth (the ESP session key used to authenticate the MIPv6 binding update lifetime as the minimum of the lifetime value suggested by the CN and its lifetime value.

3. builds an IPsec SA. If ESP is used subsequently in the packet to secure a Binding Update, the MN must use a fixed

SPI assigned from the range 1 to 255 (currently unassigned).

4. sends a sucvP3 packet. Note that this message is sent directly from the MN's CoA to the CN.

sucvP3:

A sucvP3 message contains the following fields: Puzzle reply, Public key and imprint it has used to generate its HID, a Diffie-Hellman value, the skey_espauth lifetime and an SPI for the CN to use when sending BA's (secured via ESP) to the MN. This message must be signed by the MN with its private key (the public key is used to generate the HID).

Note that this sucvP3 might be followed by an ESP header authenticating an encapsulated BU. The authentication is performed using the SA available inline within this sucvP3 packet.

When the CN receives the sucvP3, it first checks for a valid Puzzle reply. It verifies the signature using the included Public key, and then verifies that this Public key and imprint produce the sucvHID used as part of the sender's address. The CN can then conclude that the MN owns its the Home and CoA addresses.

At this point, the CN makes a note of this Public key and HID.

The CN can then compute the session keys (using the ephemeral DH value). From the fixed SPI, the CN learns that the security association material is all inline in sucvP3. It proceeds to build an IPsec SA and processes this ESP header. In preparation for subsequent ESP processing of BU's, it computes an SPI and sends it in sucvP4. After this point, and thanks to this SPI, IPsec usage reverts to normal, i.e., future BU's can be secured via ESP, unaccompanied by any inline sucvP material.

sucvP4:

In sucvP4, the CN sends an SPI. The MN will use this SPI in association with ESP in order to authenticate subsequent BU's. The CN authenticates sucvP4 with HMAC-SHA1 using the Session key Skey_sucv derived previously. Additionally, a CN that uses an SUCV address could sign sucvP4 instead. This possibility is explored below.

A CN may include a BA (binding acknowledgement) along with

sucvP4, and if so, it must use ESP for authentication. The SPI used is that communicated by the MN in sucvP3. When the MN receives a sucvP4, it must make note of the SPI corresponding to the CN.

As long as the MN uses the same HID interface identifier for its CoA, it does not have to prove the CoA ownership and BU authentication is enough.

Proving the CoA ownership can be very useful to prevent a malicious host from bombing a victim with packets by using the victim's address as CoA. For example, with ``regular'' Mobile IPv6, a host can initiate a large stream transmission from a server, and then send a BU with the victim's address as CoA to the server. As a result, the stream will bombard the victim. If a host can prove that it owns its CoA, and that therefore it is not using another node's address as CoA, this attack can be avoided. However, this does not protect against a related attack in which the objective is not to bombard a particular host, but any given network prefix or link.

If for any reason the MN configures its CoA with a new interface identifier, it must restart the whole protocol sequence.

5.3 Deriving the Session Keys

We need to generate keying material and keys for the SUCV protocol itself and for use with ESP.

skeymat = prf(hash (g^{xy} mod p), N1 | imprint)

Where N1 is the nonce used in sucvP1 and sucvP2.

5.3.1 SUCV Session Key

skey_sucv = prf(skeymat, g^{xy} mod p | N1 | imprint | 0)

Used with sucvP4 for: authentication, and optionally with sucvP5 (see Section on Privacy Considerations) for both authentication and encryption.

5.3.2 Key for use with ESP

skeymat_espauth = prf(skeymat, skey_sucv | g^{xy} mod p | N1 |

imprint | 1)

Used to authenticate BU's unaccompanied by SUCV packets (once sucvP is completed) as well as other applications of ESP (subject to the warning at the beginning of Section 5).

Note that whereas skey_sucv is the actual key used by the SUCV protocol, skeymat_espauth is keying material used to derive the real key for use with ESP, i.e. skey_espauth in an algorithm-specific manner.

5.4 Default Algorithms

The following algorithms must be supported by any SUCV implementation:

DSA [DSA] for signing sucvP3.

Diffie-Hellman Oakley Group 1 [RFC2412] for the ephemeral Diffie-Hellman exchange.

HMAC-SHA-1-96 [RFC2404] for ESP authentication.

3DES-CBC [RFC2451] for sucvP5 and ESP encryption.

6.0 Extension for Constrained Devices

In our sucvP protocol, a MN must:

- generate a DSA public/private key pair.

- sign the sucvP3 message.

- perform a DH exponentiation to derive the Skey.

All these operations are computatively expensive especially if the MN is a constrained device (i.e. a PDA or a sensor with limited memory, battery or CPU). Elliptic curve cryptographic algorithms might be more efficient but still too expensive to execute for a constrained device.

In this section, we propose an extension to our scheme for this type of contrained devices. Our goal is to off-load most of the expensive cryptographic operations of a MN to its HA. We assume that the MN and HA share a secret key, and that the MN trusts its HA.

The proposed extension operates as follows:

1. The HA generates the DSA keys (public and private keys) and sends the public Key to the MN via the secured channel.

2. The SUCV id and HID is generated by the MN itself by choosing a k and computing sucvHID = prf64(hash(publicKey), k).

3. When a MN wants to initiate a sucvP exchange with CN, it sends a SUCV_request messages, that contains the CN address and the k value, to its HA (authenticated with the shared key). The HA then initiates a sucvP exchange with the CN. The HA then proves that it knows the private key corresponding to the public by signing the exchanged messages (sucvP has to be slightly modified here) and generates a session key, SKey using the DH algorithm.

4. The HA then sends the Skey to the MN via the secure channel.

5. The MN can then send authentication BU's to the CN using the SKey.

With this extension all the expensive cryptographic operations are offloaded to the home agent but the session key that is used to authenticated the MIPv6 BU (Skey) is only known to the MN, its HA and the CN. A malicious host that wants to redirect a MN's traffic needs either to discover the HA-MN secret key or to find a public key/private key pair and a k' such that

sucvHID = prf64(hash(public), k')

Both are very difficult to achieve.

7.0 Privacy Considerations

A normal sucvP exchange consists of sucvP1 through sucvP3, and a subsequent sucvP4 authenticated using the session key. This basic protocol does not allow any hijacking attacks, so it already fulfills the security requirements for protecting BU's in MIPv6 as defined by the Mobile IP working group [MIPv6SecReq].

7.1 Support for Random Addresses [RFC3041]

A first concern regarding privacy is how to use random

addresses as defined in RFC3041 in a mobile environment. The issue here is that, whereas these addresses hide a node's permanent identifier (perhaps derived from IEEE addresses), the node cannot prove address ownership of them so it cannot safely send binding updates. This means that an MN cannot use RFC3041 addresses with route optimization. SUCV addresses are indistinguishable from those defined in RFC3041, with the added benefit that an MN can use them in a route optimized fashion. The basic sucvP outlined above already handles this case. The only consideration is that nodes interested in being anonymous may want to use ephemeral SUCV identifiers (as opposed to more permanent or longer-lived SUCV ID's) for this purpose.

Furthermore, if nodes wish to have higher protection against attackers than what is afforded by 63 bits in the sucvAddr, they can use an sucvID. The protocol exchange is the same, but since an sucvID is a pure identifier, as shown below, routing information must be included somewhere else in the packet, via home address options and routing headers (alternatively, tunneling headers could be used as well). This poses no difficulty if the MN operates as a client, always initiating contact with the CN, but would otherwise require mechanisms beyond the scope of this paper.

7.2 Support for Confidentiality

7.2.1 Confidentiality

If confidentiality is a concern, there is the possibility of an intruder in the middle gaining knowledge of the session keys, as explained in the section on Security Analysis. In fact, sucvP prevents an intruder from impersonating an mobile node but not from impersonating a correspondent node. As a result, an MN might think that it is talking with its CN whereas it is actually talking with an intruder. The MN may wish to make sure it is indeed talking to a given CN whose address it has previously obtained (via, for example, a DNS search, or a preconfigured list). If in addition to the MN, the CN also uses an SUCV address this problem can be prevented. We suggest that a CN uses a SUCV address when confidentiality is an issue and that the CN sign sucvP4 to prove its address ownership. By doing so, both MN and CN have the assurance that they are talking to each other and not to an intruder.

7.2.2 Location Privacy

In Mobile IPv6:

Each packet (BU and data) sent by a MN contains a HomeAddress option that reveals the MN's home address.

Each packet sent to a MN contains a routing header with the MN's home address.

As a result it is very easy for any host in the network to track the location of a MN by snooping its packets. If location privacy is an issue, a MN can use an ephemeral home address sucvADDR_ephem instead of its real one (sucvADDR), and only reveal the latter to its CN. Packets (BU and data) sent over the network then use the ephemeral home address sucvADDR_ephem.

For this to work, the MN will need an ephemeral SUCV identity sucvID_ephem, and defer revealing its more permanent SUCV identity sucvID after the CN has proven ownership of its address. This is accomplished roughly via the following extended protocol sequence:

sucvP1: as usual

sucvP2: the CN adds a bit to advertise its SUCV capabilities

sucvP3: the MN proves ownership of its sucvADDR_ephem (derived from an ephemeral public-private key. At this point, the MN derives session keys but is not yet sure it is sharing them with the CN itself.

sucvP4: the CN proves ownership of its SUCV address by signing sucvP4 with its private key, at which point the MN knows the session keys have not been compromised by an intermediary.

sucvP5: the MN uses the session key obtained above to send an encrypted payload revealing its actual SUCV Home Address sucvADDR. sucvP5 must be signed with the key used to generate the sucvADDR in order to prove its ownership.

Notice that if the MN wishes to use the stronger mode, it can do so by using an sucvID_ephem and sucvID instead of sucvADDR_ephem and sucvAddr, respectively. As in the discussion above, this provides for more protection against attackers, with the proviso, in practice, the MN may be limited to being a client. That is, it must initiate communication with the CN,

because it is now using non-routable entities (SUCV ID's versus SUCV Addresses).

8.0 Security Analysis

This section includes a security analysis of the SUCV protocol.

8.1 Hash ID Size Considerations

In SUCV addresses, one of the lower 64 bits is reserved as the local/universal bit (the u bit), so only 63 bits are actually usable as a hash.

Suppose the hash function produces an n-bit long output. If we are trying to find some input which will produce some target output value y, then since each output is equally likely we expect to have to try 2^{(n-1) possible input values on average.

On the other hand, if we are worried about naturally ocurring SUCV address duplications, then by the birthday paradox we would expect that after trying 1.2*2^n/2 possible input values we would have a 50% probability of collision [HandBook].

So if n=63, you need a population of 1.2*2^{31.5} i.e. 3.64*10^9 nodes on average before any two produce duplicate addresses. This is acceptable especially if you consider that this collision is actually harmfull only if the 2 hosts (that collide) are in the same site (i.e. they have the same 64 bit prefix), and have the same correspondent nodes. This is very unlikely. Additionally, assuming the collision is not deliberate the duplicate address detection (DAD) will detect it.

If an attacker wishes to impersonate a given SUCV address, it must attempt 2^{62} (i.e. approximately 4.8*10^{18}) tries to find a public key that hashes to this SUCV address. If the attacker can do 1 million hashes per second it will need 142,235 years. If the attacker can hash 1 billion hashes per second it will still need 142 years.

If we use SUCV Addresses as suggested in RFC3041 (perhaps renewing them as often as once every 24 hours), an attacker would then have to to hash 5.3*10^{13} hashes/second in order to be able to find a public key that hashes to the sucvHID of a given host.

Note that the previous analysis only considers the cost of

computing the hash of the public key. Additionally, an attacker must also generate a valid (public, private) key pair. This is a significantly more expensive operation.

This would still leave open the possibility of brute-force attacks. In this scenario, a bad guy, BG, could generate a huge table of PK's and their corresponding HID's, assuming any fixed imprint. It could then look for matching real IP addresses. By doing so it would identify a victim for a hijacking attack. BG can send a BU to any CN without a binding entry for the victim's address (for example, by targetting non-mobile fixed hosts as victims).

In general, such attacks are possible with hash functions, but not with keyed hash functions because they require interacting with the legitimate user [HMAC]. Notice that normal usage of keyed hash functions requires an authenticated secret, which we do not have. Nevertheless, we can still limit exposure by creating the HID (or ID) using (in addition to the Public key) some key or known state that is established in advance of the sucvP interaction itself, and which will force interaction with the MN.

This is the role of the imprint, sent by the MN to the CN in sucvP. Since the imprint is not authenticated, the CN could verify it independently of sucvP, perhaps by checking directly with the MN by routing it via the HA. True, the imprint is not a secret as expected for HMAC use, but it serves to severely limit which entities can launch the attack to only those entities with this priviledged location, and within this time period.

As another possibility, the imprint may instead be a quantity which the CN knows about the MN, and which the CN can verify independently using a separate subsystem (DNS, routing fabric, etc). In this case, the attack is limited to only those nodes for which the imprint is also a valid quantity. However, tying the HID in this manner may have undesirable consequences with regards to privacy and location independence (for example homeless operation).

Alternatively, one could always use sucvID's (in which case the brute-force attacks would be nearly impossible).

Even for HID's, actually carrying out such brute-force attacks remain highly unlikely in practice, and we claim our scheme remains secure even without requiring any of the above counter-measures.

8.2 Key size considerations

There are three ways that an attacker could break the MIPv6 security protocol presented in the paper:

1. If an attacker find a DSA public/private key pair that hashes to the MN's sucvID, it can run a sucvP exchange with a CN and impersonate the MN. This can be achieved by a brute force attack. The attacker tries several public keys as input to the hash function used to generate the sucvID. The difficulty of this attack depends on the size of the sucvID and is at least as hard as breaking a symmetric key algorithm that uses the same key size as the sucvID size (actually this is more difficult because the attacker must also generate valid public/private key pairs before performing the hash function).

2. If an attacker can find the public/private key pair that is used to generate the sucvId and sign sucvP3, an attacker can impersonate a MN in sucvP. Breaking a DSA system depends on the DSA modulus and subgroup.

3. If an attacker can retrieve the generated session key it can send fake BU's on behalf of the MN and redirect its traffic. An attacker has two ways of retrieving the session key: (1) generate it from the DH values exchanged between the MN and the CN, or (2) perform a brute-force attack on the session key itself. The difficulty of these attacks depend respectively on the DH modulus size and the session Key size.

A security system is consistent if all the components of the security chain provide the same security levels and none of them is a weak link. Most of the security parameters used in our proposal (DH modulus size, Session key size, DSA subgroup) can be adjusted. The only fixed parameter is the SUCV identifier itself. It is either 63 bits long (i.e. we use an sucvHID) or 125 bits long (if using an sucvID itself).

If we use sucvHID's, the security of our proposal depends on these 63 bits. Accordingly, the symmetric key strength should not be less, not would we gain much by it being significantly stronger. In light of [DetStrengths], Oakley group 1 is about enough for this application (although there are other more conservative views [lenstra00selecting]).

However, if we use suvcID's, we will need a symmetric key strength of almost 128 bits (125 bits) of output from our prf. Notice that 96 bits symmetric keys are generally considered

safe for another 20 years or so. However, if we want to keep up with the strength afforded by the sucvID itself, we would need to use other MODP groups [MODPGrp]. For example, MODP group 5 with exponents of 1563 bits should be enough to derive 90 bit symmetric keys. MODP group 6 with 2048 bits should be used to produce 100 bit symmetric keys.

8.3 Intruder-in-the-middle attacks

As described above, a mobile node and its correspondent node derive a shared (symetric) key to authenticate the MIPv6 Binding updates sent by the MN.

The MN and its CN derive the shared key using the Diffie-Hellman algorithm.

The CN chooses a random secret y and sends g^y mod p to the MN (in the DH value field of sucvP2)

The MN chooses a random secret x and sends g^x mod p to its CN (in the DH value field sucvP3)

The session key shared by the MN and its CN is then a hash digest of g^{xy} mod p (g and p are known by the MN and CN).

8.3.1 Summary of the Attack

Diffie Hellman is know to be vulnerable to the intruder-in-the-middle attack on un-authenticated DH key agreement:

```
CN -->g^y-->Intruder-->g^y_i-->MN
CN<--g^x_i-->Intruder<--g^x<--MN
```

The intruder intercepts g^y sent by the CN and sends g^{y_i} to the MN. The intruder also intercepts g^x sent by the MN and sends g^x_i to the CN. As a result, MN shares the key g^{xy_i} with the intruder (it actually thinks that it is sharing this key with its CN). The CN shares the key g^{x_iy} with the intruder (it actually thinks that it is sharing this key with the MN). The Intruder can then impersonate the MN and the CN.

In our protocol, the MN signs sucvP3 (which contains g^x). As a result, the intruder can not modify nor replace this message. This only thing that the intruder could do is the

following attack:

sucvP1: CN<--HID'-->Intruder<--HID<--MN
sucvP2: CN-->g^y-->Intruder-->g^yi-->MN
sucvP3: CN<--g^xi-->Intruder<--g^x<--MN

In sucvP1, MN sends its HID by virtue of sending from its address (the HID is just the bottom 64 bits in the address). The intruder could replace this HID by another value, say HID_i, without affecting return routability, as long as the prefix remains the same. In sucvP2, the CN sends its DH value g^y, which is replaced by the intruder for g^{y_i}. In sucvP3, the MN sends its g^x. Notice that the intruder can replace it by another g^{x_i} as long as this {g^x_i} is used to create HID_i.

8.3.2 Risks

The keys created are derived from:

g^{xy_i} (between the MN and the intruder) and

g^{yx_i} (between the intruder and the CN).

So the intruder cannot pass itself off as MN (assuming it is computationally unfeasible to find another private-public pair that generates the same HID). It can, however, pass itself off as MN_i, where this is the address formed from HID_i. This means that it is not possible for an intruder to hijack an existing communication between MN and CN. But if the intruder is present at the very beginning of the communication, and if it sits on the path it could supplant MN. In so doing it could obtain knowledge of any session keys derived for this communication.

If the session supported encryption, the endpoints might be led to believe in the privacy of their conversation, oblivious to the fact that the intruder could snoop. For example, suppose an MN established an sucvP session with an CN. Subsequently, and using this optimized path, an application (for example telnet) started. If a security policy database required all such application traffic to be encrypted, a misconfigured system might leverage the existing sucvP session and use ESP for confidentiality. This would result in the intermediary being privy to all the application traffic.

Because of this, sucvP session keys must not be used for anything more than securing BU's. In other words, IPsec traffic selectors in the SPD must limit use of SA's obtained

via sucvP for the sole purpose of securing BU's. In order to avoid any potential misapplication of these SA's BU's must not be piggybacked.

Not heeding the above guidelines may result in the aforementioned snooping attack. Nevertheless, the attacker would have to remain on the path forever. This interception is possible because of the non-authenticated nature of the example. Of course, if the exchange is authenticated, this would not be possible. Even if this interception is possible, once the intruder ceases to be on the path between MN and CN there is nothing further it can do. In other words, the use of unauthenticated SUCV entities does not add any risk to those that currently exist. Even unauthenticated SUCV, eliminates the possibility of on the path redirection of traffic. Notice that with current MIPv6, ``off the path'' (as well as ``on the path'') redirection of traffic is possible.

In some case, a MN might request to its CN to acknowledge the reception of the BU. The intruder could actually fool the MN by sending an acknowledgement with the CN address as source address (note that the intruder could also authenticate this acknowlegement since it knows the key used by the MN, g^{xy}). This might confuse the MN that has received an acknowledgement but keeps receiving the packets from the CN via its home agent (note that the same problem exists also will current Mobile IPv6 specification)!

One solution to these problems is for the the CN to use an SUCV address and to sign sucvP2 (the message that contains the DH value). Then, the intruder will not be able to substitute g^y by g^{y_i}.

Of course, the intruder can hinder successful completion of the SUCV protocol, thus preventing the CN from heeding the MN's BU using route optimization to the MN. In effect, this is a denial-of-service attack against route optimization, and it leads to service degradation not disruption.

The previous security analysis shows that sucvP prevents any intruders from redirecting the traffic addressed to a mobile host's home address and consequently provides the minimal Mobile IP security requirement [MIPv6SecReq].

8.3.3 Why not Route sucvP2 Through the Home Agent?

What, if we assume sucvP1 was carried with a home address

option, and then sucvP2 travelled via the home agent. At this point, the home agent can check that the validity of this MN_i (corresponding to HID_i), its current care-of address, etc. In this case, none of the above snooping would be possible. In order to further mitigate the sucvP2 packet from being redirected, the MN must check upon its reception that it was sent tunneled by its home agent. Home address options can be misused to set up distributed denial of service attacks whereby these options are sent to numerous hosts prompting them all to respond to the same address. Even if CN's exercise caution when sending their sucvP2 packets as instructed via a home address option, the nature of DDoS attacks is such that any given CN may not send more than a few sucvP2's to the same home address region (same prefix), the collection of thousands of such responses may be sufficient to clog a target network.

The above analysis shows the pro's and cons of using the home address option. Notice that for our purpose of authenticating BU's we do not need to resort to the heavy requirement of routing sucvP2 via the HA. SUCV packets are exchanged directly between the MN and the CN.

8.4 Denial-of-Service Attacks

Denial-of-service (DOS) attacks that exhaust a host resource (memory and computional resources) is a major security threat on the Internet. In this section we study the behavior of the sucvP against DoS attacks.

sucvP1 storm:

Malicious hosts, could try to attack a host, by sending a storm of sucvP1 messages. We prevent this potential attack as follows:

1. When receiving a sucvP1, a host does not create any state and replies with a constant message (sucvP2) that contains a client puzzle [PUZZLES].

2. An host only creates state if it receives a valid puzzle reply to its puzzle request (in sucvP3).

sucvP2 storm:

Malicious host could try to attack a host by sending a storm of sucvP2 messages. We prevent this attack by inserting a nonce, N1, in the sucvP1. If a host receives a sucvP2

with a nonce N1 that is not equal to the nonce N1 that it has set in the initial sucvP1, this sucvP2 must be rejected.

Note that an intruder (between the MN and its CN) could intercept the sucvP1 and reply to the MN with a fake sucvP2 containing a valid N1 and an intentionally difficult puzzle request. The MN would then spend a lot of CPU and power computing the puzzle reply. This attack can be avoided if the MN had a mean to authenticate the address used by its CN. One solution is that the CN uses a SUCV address and signs sucvP2.

Instead of this heavy alternative, we suggest that a MN simply reject any sucvP2 messages that contain an overly complex client puzzle request Of course, the MN itself defines the complexity threshold of the puzzle request as a function of its processing power.

As a result, the attack that consists of sending complex puzzles (in sucvP2) to a MN, in order to exhaust its computing resources, will not be sucessful, because the MN will drop the sucvP2. The MN service will be degraded (because its incoming packets will then be routed through its home agent) but not disrupted.}

sucvP3 storm:

Malicious hosts could try to attack a host by sending a storm of sucvP3 messages. We prevent this attack by using a client puzzle. A host accepts a sucvP3 message only after verifying that the puzzle reply (contained in the sucvP3) is valid.}

9.0 Security Considerations

The technique introduced in this document is meant to increase the level of security in the Internet.

This document explains the concept of statistical uniqueness and cryptographic verifiability (SUCV), specially as it applies to IPv6 addresses in the form of SUCV addresses. The SUCV characteristics are used to prove address ownership, thus preventing a class of attacks which exploit this fault in many types of commands. In particular, commands which alter how an address is treated by peers or by the routing infrastructure can be used to launch denial of service attacks or hijacking attacks. Proving address ownership eliminates these attacks. However, given that this technique is meant to be used primarily

in the absence of global infrastructures, the possibility of man in the middle attacks does remain. Nevertheless, these attacks are limited by the use of cookies and client puzzles.

10.0 Conclusions

The present document focuses on the use of the SUCV property to enhance the security of exchanges between an arbitrary pair of peers in the absence of any third party. In particular, we propose that SUCV addresses be used to solve the issue of securing binding updates in Mobile IPv6.

11.0 Acknowledgements

The authors appreciate the helpful comments received from the following individuals: Erik Nordmark, Alberto Escudero, Lars Henrik Petander, Imad Aad, Pars Mutaf and the anonymous reviewers. Special thanks to Julien Laganier for his sundry comments and major contribution to the packet formats.

A. Appendix: sucvP Protocol Specification and Packet Formats

SUCV uses a very simple negotiation. First of all, hopefully no negotiation of cryptographic parameters is necessary, as the defaults should be applicable. Any departure from the default is proposed by the initiator using TLV encoding and either accepted (and used) or rejected by the responder.

Furthermore, SUCV deals with cryptographic suites, similar to TLS and JFK [JFK] usage.

The initial set of algorithms that MUST be supported is:

Suite	Identifier
SUCV_RSA_WITH_ESP_AES_128_CBC_HMAC_SHA1	{0x00,0x01}
SUCV_RSA_WITH_ESP_3DES_CBC_HMAC_SHA1	{0x00,0x02}
SUCV_RSA_WITH_ESP_NULL_HMAC_SHA1	{0x00,0x03}

Notice that SUCV only supports RSA certificates, ESP headers and SHA1. These are repeated in the names of the suites above for clarity, not to imply that departures are allowed (to DSS, AH or MD5, for example).

The default suite that requires no negotiation is:

SUCV_RSA_WITH_ESP_AES_128_CBC_HMAC_SHA1

This suite has been assigned an identifier, but the identifier will never be used in an SUCV payload because fixed payloads are enough to support the default case. As compared to IKEv2 [IKEv2], SUCV does not allow individualized negotiation for tranform types 1 (encryption algorithm), 2 (pseudo-random function), 3 (authentication type) and 4 (integrity algorithm). By using the above default suite, SUCV mandates the following: AES_128_CBC for encryption, HMAC_SHA1 for pseudo-random function, RSA for authentication type and SHA1 for integrity algorithm.

It does, however, allow for negotiation on Diffie-Hellman group (IKEv2's transform type 5). In so doing it reuses the numbering used in IKEv2. The mandatory group for Diffie-Hellman exchange is group 5 (1536 bit MODP) and RSA signature uses exponent 65537.

A.1. Packet Formats

sucvP is an IPv6 protocol, identified by protocol number (sucvP TBD by IANA) in the Next Header field of the immediately preceding header.

The Next Header field identifies the next protocol in the IPv6 daisy-chain header as per normal IPv6 usage.

The Version field for this version of sucvP MUST always contain the number 1.

The Length field indicate the length in bytes of the whole sucvP header, including any possible options.

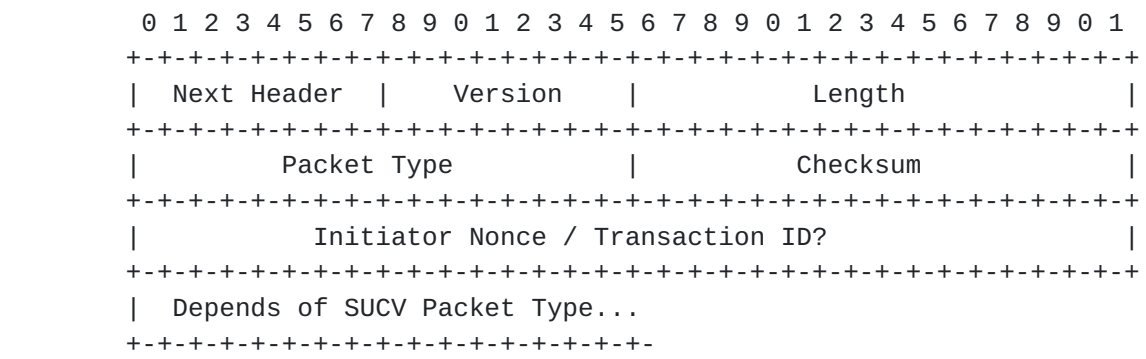
The Checksum is the 16-bit one's complement of the one's complement sum of the entire sucvP message starting with the Next Header field (see below), prepended with a "pseudo-header" of IPv6 header fields, as specified in [IPv6, section 8.1]. The Next Header value used in the pseudo-header is (sucvP TBD).

For computing the checksum, the checksum field is set to zero.

The Packet Type is defined to be {packet number|sequence type}; the packet number is related to the relative position within the exchange, while the sequence type is related to the sort of SUCV exchange we wish to perform. For example, an sucvP3 packet used to establish initiator's proof of ownership without privacy

considerations would be labeled with the packet type {0x03|0x01}. It would be labeled {0x03|0x02} if it is used to establish mutual proof of ownership or to avoid disclosure of identity. On the other hand, an sucvP1 packet would always be labeled {0x01|0x01}, as there is only one type of p1 in both exchange types {initiator|responder|privacy}.

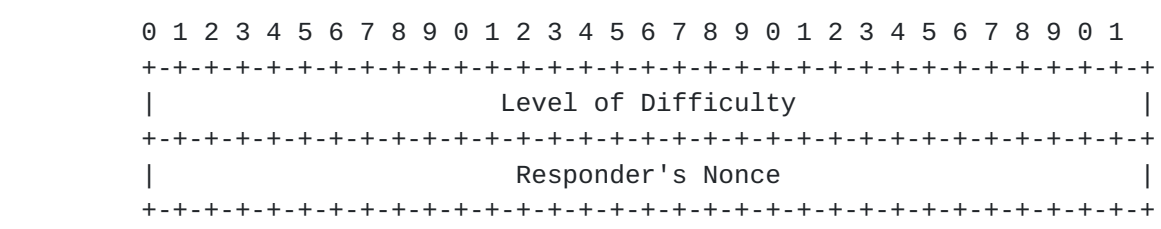
A.2. Common Header Format



A.3. Cookie Puzzle Request Format

Level of Difficulty is k

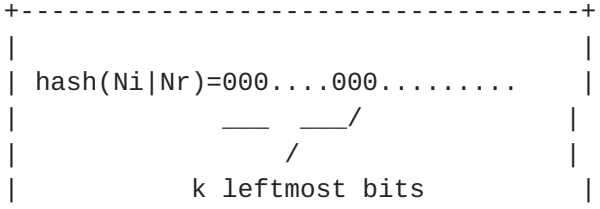
Responder's Nonce is Nr

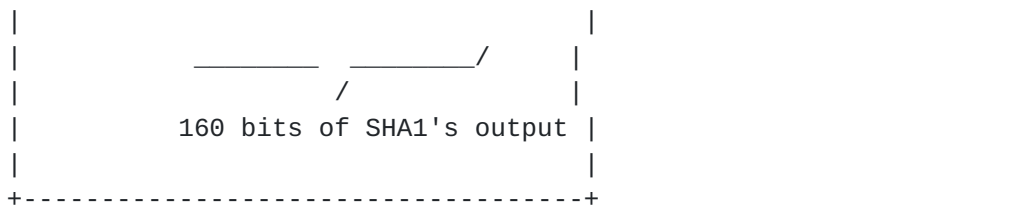


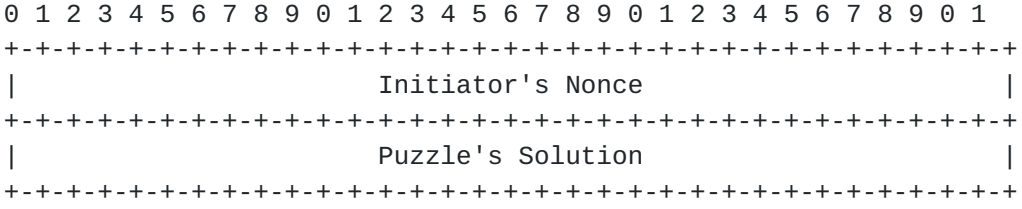
A.4. Cookie Puzzle Reply Format

Initiator's Nonce is Ni

Puzzle's Solution is an integer X which verifies the property :



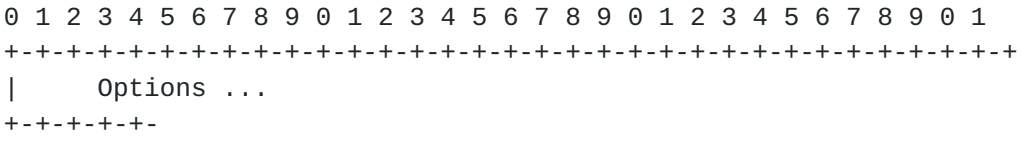




A.5. P1 Packet Format

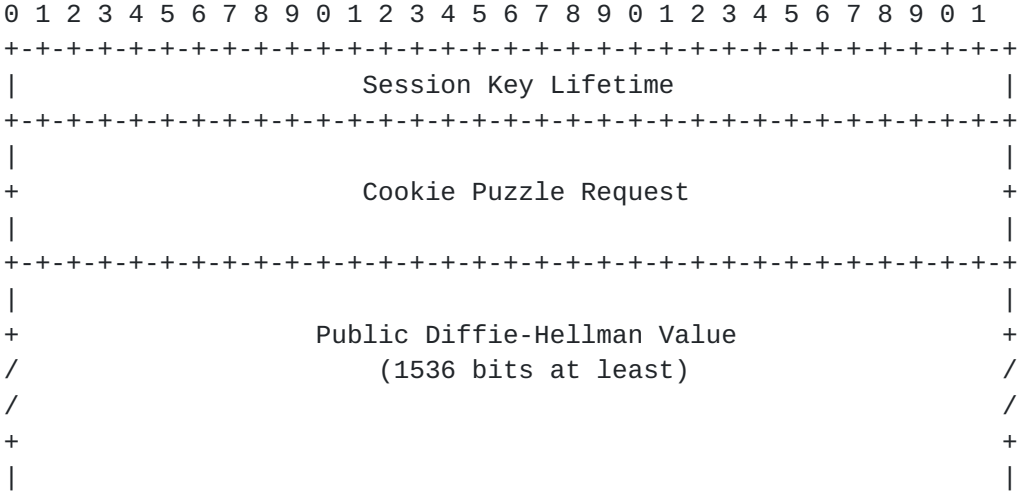
Type 0x0101

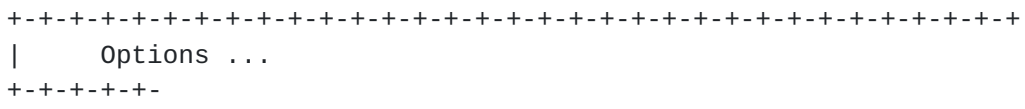
P1 is just the common packet format without additional fields, although a vendor-specific or application-specific options field is possible.



A.6. P2 Packet Format

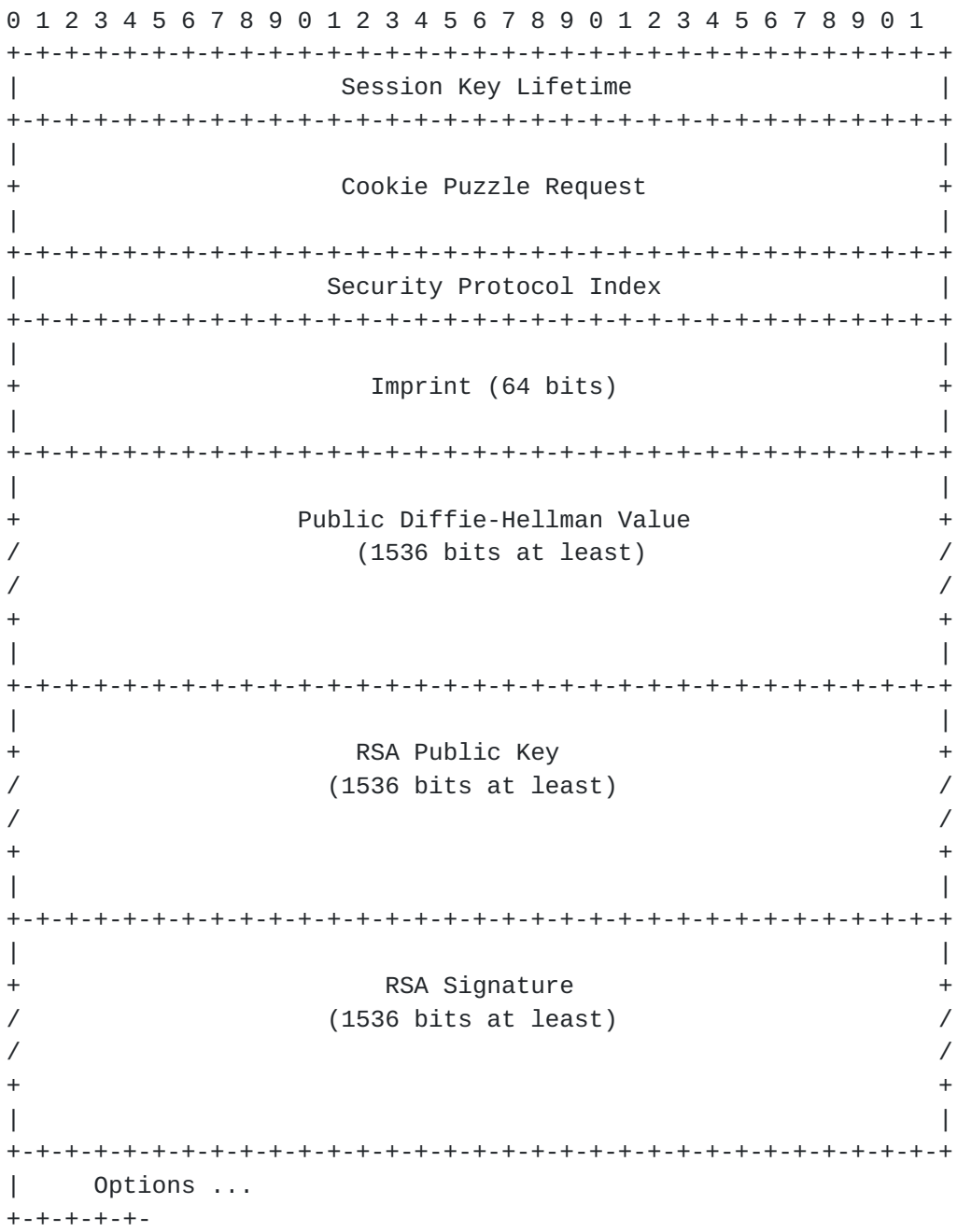
Type 0x0201





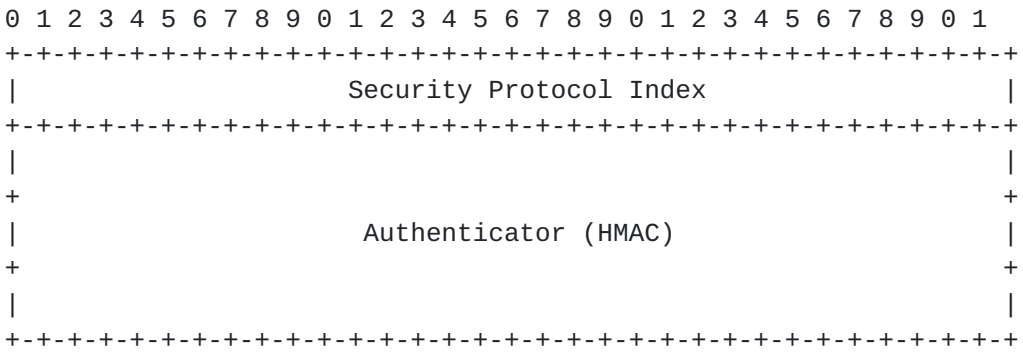
A.7. P3 Packet Format

Type 0x0301

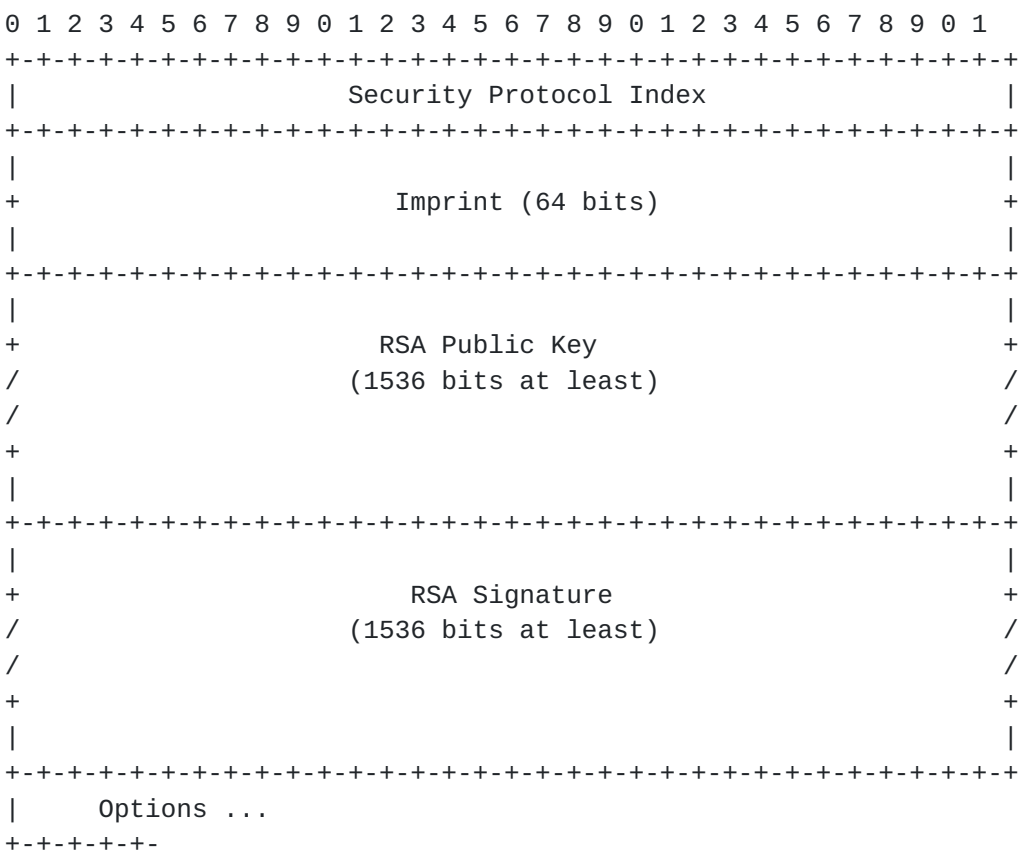


A.8. P4 Packet Format

Type 0x0401 (to be used to prove only initiator's ownership)



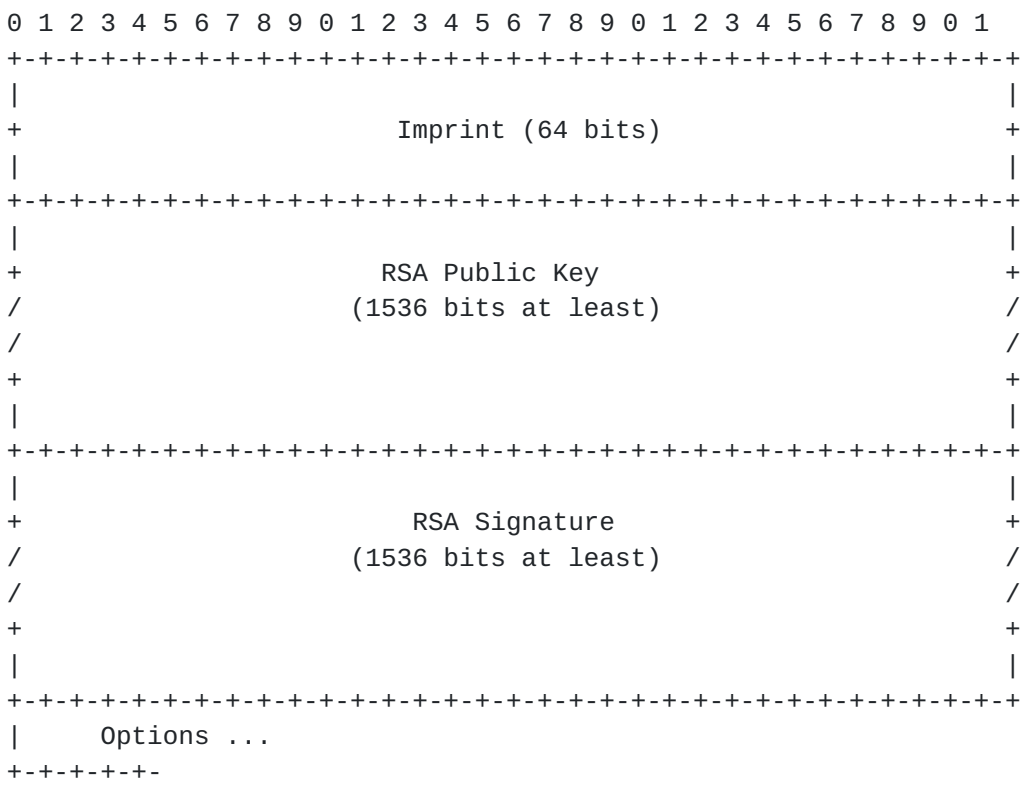
Type 0x0402 (to be used to prove responder's address ownership)



A.9. P5 Packet Format

Type 0x0502 (to be used to protect disclosure of initiator's identity)

Note that this packet must be protected within an ESP payload, to avoid disclosure of initiator indentity (Public Key).



References

[ADDROWN] Pekka Nikander, "An Address Ownership Problem in IPv6", draft-nikander-ipng-address-ownership-00.txt, February 2001.

[BIRTH] http://www.rsasecurity.com/rsalabs/faq/2-4-6.html

[CAM] Greg O'Shea and Michael Roe, "Child-proof Authentication for MIPv6 (CAM)", ACM Computer Communications Review, April 2001.

[DetStrengths] Hilarie Orman and Paul Hoffman, "Determining Strengths For Public Keys Used For Exchanging Symmetric Keys", draft-orman-public-key-lengths-02.txt

[HandBook] A.J. Menezes, P.C. van Oorschot and S.A. Vanstone. Handbook of applied cryptography. CRC Press, 1997.

[HMAC] Krawczyk, Bellare and Canetti, "HMAC: Keyed-Hashing for Message Authentication," RFC 2104, February 1997.

[IKEv2] D. Harkins, C. Kaufman, S. Kent, T. Kivinen, and R.

Perlman, "Proposal for the IKEv2 Protocol", draft-ietf-ipsec-ikev2-02.txt.

[IPV6ADDR] Hinden, Deering, "IPv6 Addressing Architecture," RFC 2373, July 1998.

[JFK] W. Aiello, S.M. Bellovin, M. Blaze, R. Canetti, J. Ioannidis, A.D. Keromytis and O. Reingold, "Just Fast Keying (JFK)", draft-ietf-ipsec-jfk-03.txt.

[lenstra00selecting] A.K. Lenstra and E.R. Verheul, "Selecting Cryptographic Key Sizes," manuscript, (Oct.1999). http://citeseer.nj.nec.com/lenstra99selecting.html

[MIPv6] D. Johnson, C. Perkins, J. Arkko, "Mobile IP for IPv6", draft-ietf-mobileip-ipv6-18.txt

[MIPv6SecReq] Mankin, Patil, Harkins, Nordmark, Nikander, Roberts, Narten, "Threat Models Introduced by Mobile IPv6 and Requirements for Security in Mobile IPv6", draft-ietf-mobileip-MIPv6-scrty_reqts-02.txt

[MODPGrp] T. Kivinen and M. Kojo, "More MODP Diffie-Hellman Groups for IKE", draft-ietf-ipsec-ike-modp-groups-03.txt

[PUZZLES] Tuomas Aura, Pekka Nikander and J. Leiwo. DOS-resistant Authentication with Client Puzzles.

[RFC2404] Madson and Glenn, "The Use of HMAC-SHA-1-96 within ESP and AH," RFC 2404, November 1998.

[RFC3041] T. Narten, R. Draves "Privacy Extensions for Stateless Address Autoconfiguration in IPv6," RFC 3041.

[SHA1] Eastlake, Jones, "US Secure Hash Algorithm 1 (SHA-1)," RFC 3174, September 2001.

[WeakMD5] H. Dobbertin, "Cryptanalysis of MD5 Compress", http://www.cs.ucsd.edu/users/bsy/dobbertin.ps

Authors' addresses

Questions about this document may be directed to:

Gabriel Montenegro
Sun Microsystems Laboratories, Europe
29, chemin du Vieux Chene
38240 Meylan, FRANCE

Voice: +33 476 18 80 45
E-Mail: gab@sun.com

Claude Castelluccia
INRIA Rhone-Alpes
655 avenue de l'Europe
38330 Montbonnot Saint-Martin
FRANCE
email: claude.castelluccia@inria.fr
phone: +33 4 76 61 52 15
fax: +33 4 76 61 52 52

Changes

Changes between version 02 and 03:

- Added packet formats and sucvP protocol specs (to be taken out into a separate draft later on).
- Deleted "dead" references to expired internet drafts and to HIP drafts.
- Sundry editorial changes and corrections.

Changes between version 01 and 02:

- Cosmetic editorial changes, reorganization, etc.
- Added more details on SKey derivation to the protocol overview section.
- Added a terminology section.
- Specified more fully the sucvP protocol, which means SUCV is now independent of HIP.
- Added a section on extensions for constrained devices.
- Added a section on Privacy considerations.