Statistically Unique and Cryptographically Verifiable (SUCV) Identifiers and Addresses

Gabriel Montenegro
SUN Labs, Europe
29, chemin du Vieux Chêne
38240 Meylan, France
gab@sun.com

Claude Castelluccia
INRIA Rhône-Alpes
655, avenue de l'Europe
38330 Montbonnot, France
claude.castelluccia@inrialpes.fr

Abstract

This paper 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 and Addresses. Their characteristics allow them 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 mechanisms like Binding Updates in Mobile IPv6.*

keywords: Security, Mobile IPv6, Address ownership.

1 Introduction

This paper addresses the identifier ownership problem [1] by using characteristics of Statistic Uniqueness and Cryptographic Verifiability (SUCV) of certain entities which this document calls SUCV Identifiers (SUCV ID's). This paper 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 mechanisms like Binding Updates in Mobile IPv6.

This paper is structured as follows: Section 2 defines the *address ownership* problem. Section 3 presents the notation used throughout this paper. Section 4 gives an overview of our proposal. Section 5 presents SUCV identifiers and addresses, and how to generate them. Section 6 describes *sucvP*, the protocol that is used by a mobile node to prove owernship of its addresses to its correspondent nodes and to generate the session keys. Section 7 presents an extension to sucvP for constrained devices (PDA's, sensors, phones, etc.). Section 8 explains how to provide data confidentiality and location privacy with sucvP. Section 9 presents a security analysis of our proposal. Section 10 compares our scheme with related work. Finally Section 11 concludes.

2 Problem Statement

[1] argues that there is a fundamental problem in handling operations like Binding Updates (BU's) in Mobile IP for IPv6 [16], 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 achieve authorization is by showing that the requesting node owns the address for which routing information is being altered. Quoting [1]: "Currently there exists no specified mechanism for proving address ownership in Internet-wide scale".

3 Notation

This Section presents the notation used throughout this paper.

- *prf*: Pseudo-random function. SUCV mandates the use of the keyed hash function HMAC [23] 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 [24].

- *hash*: Cryptographic hash function, SHA-1 [28] 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[13]. Obtained as the keyed hash output of the hash of the public key, using an imprint value as the input key.

- *MIPv6*: Mobile IPv6 [16].

- *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 [16]

4 Proposal Overview

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 securerly bound to the requesting entity.

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

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 our proposal, the principal self-generates a private/public key pair. However, 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. Instead, the principal uses material obtained via a *prf* of 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. These considerations lead to the following fundamental assumption with respect to the above Default Trust Rule:

- Default Trust Rule:

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

5 SUCV Identifiers and Addresses

In MIPv6, a node starts using its home address, and issues BU's as it moves. Handling these BU's securely is the issue. It is never evident to the CN that whoever was using an address actually owns it. At the very most, the MN can prove that at some point it was using a certain address, but it cannot prove ownership. Ignoring this subtle distinction leads to *denial-of-service* (DOS) and hijacking attacks.

Relying on ingress filtering may limit the risk, but essentially, the only way for a node to prove ownership of an identifier (in the absence of any other centralized or global mechanism), is for it to prove that it created this statistically unique series of bits.

5.1 SUCV Identifiers

The idea is to use identifiers that 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 one node: its peer in the current exchange.

Using identifiers that satisfy the SUCV conditions outlined above, it is possible to gain the tremendous advantage that other nodes can safely believe the node when it claims ownership of that identifier. Hence they can safely heed its redirects when it says that it is now available at some different CoA (and later at another). Furthermore, you do not rely on ingress filtering to limit exposure.

What should one use: pure identifiers with no routing significance or addresses? With pure identifiers, routing information must be included somewhere 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 both routable and satisfy the SUCV property: *SUCV addresses*.

5.2 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 [4]):

- use the top 64 bits from your routing prefix (as in [27])
- define the bottom 64 bits as an SUCV ID (called the sucvHID). Use these 64 bits instead of the *interface identifier* in IPv6 [13].

The resultant 128 bit field is an identifier that is also routable, avoiding the need for taking extra space in the packet by sending routing options. Notice that even after moving, it is possible to reuse the *sucvHID* 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 to assure they are routed properly. Eavesdroppers may still know where that identity is at any particular point in time. Nevertheless, 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 (see Section 8).

5.3 Generating SUCV Identifiers and Addresses

Identifiers and addresses for use with SUCV are generated as follows:

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

Where:

- *imprint*: The 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 (see Section 9.1 by limiting their applicability, or by forcing interaction with the MN.
- *PK*: The public key is the DSA public key.

Note that according to [13], 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.

6 SUCV Protocol (sucvP) Overview

The following protocol, sucvP, is run between a MN and an arbitrary CN to:

- prove the Mobile host home address and possibly CoA ownership
- to establish an IPSec ESP SA (Skey, Lifetime, SPI) between the MN and its CN that will be used to secure MIPv6 BUs.

As for the choice of using AH or ESP to protect the binding updates, we chose the latter, because we believe there is no added value in protecting 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 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 BUs (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.

6.1 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:

1. 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
2. not all nodes need to use SUCV addresses, only those that wish their binding updates to be heeded (mobile nodes)
3. 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)
4. 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.

6.2 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* ($g^y modp$)*, 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 [20]. 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 Section 6.4) 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 ($g^x modp$) and derives from it and the DH received from the CN the session keys (see Section 6.3).
  2. computes $skey_{espauth}$ (the ESP session key used to authenticate the MIPv6 binding update - see Section 6.3) 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 (as per Section 5.3. 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 as descibed in Section 6.3).

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 in Section 8.

  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 start a big file transfer from a server and then send a BU with the victim's address as CoA to the server. As a result, the file will be send to the victim. If an host can prove that it owns its CoA, and that therefore it is not using someone's else address as CoA, this attack can be avoided.

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

6.3 Deriving the Session Keys

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

$$skeymat = prf(hash(g^{xy} modp), N1|imprint)$$

where N1 is the nonce used in sucvP1 and sucvP2.

6.3.1 SUCV Session Key

$$skey\_sucv = prf(skeymat, g^{xy} modp|N1|imprint|0)$$

Used with sucvP4 for: authentication, and optionally with sucvP5 (see Section 8) for both authentication and encryption.

6.3.2 Keys for ESP-protected Binding Updates

$$skeymat\_espauth = \\ prf(skeymat, skey_{sucv}|g^{xy}|N1|imprint|1)$$

Used to authenticate BU's unaccompanied by SUCV packets (once sucvP is completed).

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.

6.4 Default Algorithms

The following algorithms must be supported by any SUCV implementation:

- DSA [5] for signing sucvP3.
- Diffie-Hellman Oakley Group 1 [25] for the ephemeral Diffie-Hellman exchange.
- HMAC-SHA-1-96 [24] for ESP authentication.
- 3DES-CBC [26] for sucvP5 and ESP encryption.

7 Extension for Constrained Devices

In our sucvP protocol, a MN must:

1. generate a DSA public/private key pair.
2. sign the sucvP3 message.
3. perform a DH exponentiation to derive the Skey.

All these operations are very computativally expensive especially if the MN is a constrained device (i.e. a PDA or a sensor with limited memory, battery or CPU) [3]. 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, possibly obtained via *imprinting* [7], 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 DH algorithm.

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

5. The MN can then send authentication BUs 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.

8 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 [17].

8.1 Support for Random Addresses [27]

A first concern regarding privacy is how to use random addresses as defined in RFC3041 [27] 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 in Section 6 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 without any routing information, the MN is restricted to being a client. Of course, as shown below, routing information must be included somewhere in the packet, via home address options and routing headers (alternatively, tunneling headers could be used as well).

8.2 Support for Confidentiality

8.2.1 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 Section 9. In fact, sucvP prevents an intruder from impersonating a Mobile node but not from impersonating a correspondent node. As a result, a 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 signs 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.

8.2.2 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 actual one and only reveal its actual home address sucvADDR to its CN (see [15] for more details). Packets (BU and data) sent over the network then use the ephemeral home address $sucvADDR_{ephem}$.

This privacy extension can actually be applied to our proposal. 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 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 that the MN is now 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).

9 Security Analysis

9.1 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 [8].

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 [19]. 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 [9]. 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. 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.

9.2 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 [6], Oakley group 1 is about enough for this application (although there are other more conservative views [14]).

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 [18]. 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.

9.3 Intruder-in-the-middle attacks

As described in Section 6, 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 Diffie-Hellman algorithm.

- The CN chooses a random secret y and sends $g^y modp$ to the MN (in the DH value field of sucvP2)
- The MN chooses a random secret x and sends $g^x modp$ 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 $g^{xy} modp$ (g and p are known by the MN and CN).

9.3.1 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 $g^{x_i y}$ 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 (with 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_i^x is used to create HID_i.

9.3.2 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, perhaps as contemplated by default by HIP [10, 11, 12], 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 the protocol described in Section 6 prevents any intruders from redirecting the traffic addressed to a mobile host's home address and consequently provides the minimal Mobile IP security requirement [17].

9.3.3 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.

9.4 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 the section we study the behaviors of the protocol described in Section 6 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 [20].
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.

10 Related Work

CAM [4] presents a solution to the Mobile IPv6 security problem that is very similar to our proposal. When we first submitted our work to the IETF in April 2001, we were unaware of this work. CAM also uses IPv6 addresses derived from cryptographic keys to solve the MIPv6 address ownership problem. The main differences between CAM and SUCV are:

- CAM relies on signatures to authenticate binding updates. In SUCV, a signature is only used by the sucvP protocol to prove address owernship. A session key is derived between the MN and its CN, after which the binding updates are authenticated using IPsec.
- CAM requires that the CN and MN have a synchronized clock to protect against replay attacks. We believe that this is a strong assumption that is not always practical. SUCV uses a puzzle mechanism to protect against such attacks.
- CAM only uses addresses derived from cryptographic keys. In addition, SUCV defines the concept of an SUCV identifier that is longer (128 bits) and therefore more secure. As explained previously, an SUCV identifier may be used as a non-routable Home Address when the mobile node is the client (i.e. when it initiates the communication).
- In addition to proposing a mechanism to solve the address ownership problem, SUCV also provides provide data and location privacy.
- Digital signatures are very expensive operations that can not be performed on small mobile devices such as PDA or sensors. SUCV proposes an extension for constrained devices that off-load all of the expensive computations (signature, DH exponentiation and session key generation) to the home agent, while still providing end-to-end security (see section 7).
- CAM uses a hash to derive what we call the sucvHID. We use a prf-based mechanism.

The BAKE proposal [2] presents a solution to the Mobile IPv6 security and key distribution problems. As compared to SUCV, BAKE is lighter in terms of computational overhead, but weaker security-wise. In fact, BAKE only requires a few hash operations but is subject to man-in-the middle attacks when the attacker resides along the routing path between the CN and the MN's home agent. The BAKE protocol uses three messages. The first one is a trigger sent by the MN. The CN replies with a second message that contains a cryptographic token. This message is sent to the MN via its HA. Upon reception of the message, the MN sends a third message to the CN that contains another cryptographic token. The generated key is derived from the two tokens. As a result, only an intruder that can hear messages 2 and 3 could reconstruct the session key. Also, since message 2 is sent via the home agent, the CN has reasonable assurance that the home address belongs to the MN. Although BAKE has its benefits we believe that it is not secure enough to be adopted. An intruder that is close to the CN (on the same wireless link, for example) can hear all three messages and be a potential attacker.

11 Conclusion

We propose a protocol for a mobile node to prove the ownership of its addresses and to authenticate the binding update that it sends to its CN. This protocol was made part of Mobile IPv6 for deployment reasons. However the address ownership problem is more general than Mobile IPv6 and other protocols and applications might need this functionality. The sucvP protocol, in fact, can be used by all protocols and applications above it. Communicating hosts can use it to prove to each other that they own their respective addresses. They can further use it to derive shared keys that can be used by the hosts' protocols and applications. This protocol provides mutual ownership proof (i.e. proves the address ownership of both hosts) and/or unilateral ownership proof (i.e. proves only the address ownership of one of the hosts).

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.

Appendix: Implementation Considerations

This section presents some of the design choices made in our sucvP prototype implementation.

sucvP is implementated as a user process that uses UDP to exchange sucvP messages. This process uses OpenSSL library (under the FreeBSD OS) for all the cryptographic functions (hash, signatures, DH exponentiation, etc).

This section describes the puzzle mechanism that we used, the digital signature implementation, the Session key generation and the packet formats.

Puzzle implementation

sucvP makes use of puzzle to protect it against DoS attacks. The puzzle that we implemented is the one described in [20] . In the proposed scheme, the server periodically generates a nonce, Ns, and sends it to the client with the target value, Y, and the number of bits to match.

To solve the puzzle, the client generates a random nonce, Nc and solves X from the following equation by brute force.

$$hash(Ns, Nc, X) = Y$$

The server can decide the difficulty level k of the puzzle by setting the k first bits of Y to 0. The difficulty level can be increased by decreasing k (i.e. the client will has to try more value to match the corresponding Y). If $k = 0$, no work is required, the puzzle is simply a cookie. If $k = 128$, the client must reverse the entire hash function, which is computionally very difficult.

This puzzle mechanism is well suited to sucvP because the server (i.e. the CN in our case) uses the same Ns for all the sucvP1 messages it receives within a given period of time and therefore does not have to keep a state per puzzle sent. The server changes the value of Ns periodically to limit the time clients have for precomputing solutions. We use $SHA1$ as the hash function in our implementation.

The format of the puzzle request is the following: a Ns field (4 bytes), a k field (4 bytes), a Y field (20 bytes).

The format of the puzzle reply (used in sucvP3) is: a Ns field (4 bytes), a Nc field (4 bytes) and a X field (20 bytes).

Session Key generation

sucvP generates a Session key between the MN and the CN, Skey, that is used to authenticated MIPv6 BU.

Skey is derived using the Diffie-Hellman algorithm:

- The CN chooses a random secret y and sends $g^y modp$ to the MN (in the DH value field of sucvP2).
- The MN chooses a random secret x and sends $g^x modp$ to its CN (in the DH value field sucvP3)

According to [21], the result of a DH exchange, denoted $g^{xy} modp$, as a direct key to cryptographic algorithms should be avoided. Whenever possible first hash this value and then use the hashed value as a key to a *prf* for deriving further keys. This has the effect of not relying on the security of each independent bit in g^{xy} but rather on the "overall cryptographic entropy" present in g^{xy}.

As a result, the Skey in sucvP is generated as follows: $SKey = prf(hash(g^{xy} modp), N1)$, where N1 is the nonce used in sucvP1 and sucvP2.

Our implementation uses HMAC-SHA1-96 as *prf* and SHA-1 as hash function.

Our implementation default group is Oakley group 1. P is fixed to $2^{768} - 2^{704} - 1 + 2^{64} * [2^{638}.pi] + 149686$, its size is 768 bits and g is set to 2 (since P and g are fixed they do not have to be sent in sucvP2 and sucvP3). As a result only $g^x modp$ and $g^y modp$ have to sent over the network. These values are 768 bits long.

Note that the generation of a session key is not sufficient to authenticate the MIPv6 BU. A lifetime and a SPI (i.e. a Security association) have to be associated with it. In our scheme, the Skey lifetime is the minimum value of the lifetime value suggested by the MN and the CN.

Signature implementation

Our sucvP implementation uses the DSA [5] Open SSL routines to sign the sucvP3 message.

In DSA, a message M's signature is defined by r and s, such that: $r = (g^k mod p) mod q$ and $s = k^{-1}(SHA1(M) + xr)) mod q$, where g, k, p, q and x are defined in FIPS 186 (note that the public key is y and the private key is x).

The signature (r and s) is then transmitted along with the message to the verifier. The public parameters p, q, g, and y must also be made available to the verifier.

Note that according to FIPS 186, r, s and q are 160 bit long and p, q, g, y are $64 + 8 * t$ bytes long. A signature format as used in sucvP3 contains the following field: a type field (8 bits), a lenght field (32 bits) that contains the lenght of the whole signature message in bytes, a r field (160 bits), a s field (160 bits), a q field (160 bits), a t field (8 bits), a p field ($64 + 8 * t$ bytes), a q field ($64 + 8 * t$ bytes), a g field ($64 + 8 * t$ bytes), and a y field ($64 + 8 * t$ bytes).

The message verification is then performed as described in FIPS 186.

References

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

[2] Pekka Nikander and Charles Perkins, "Binding Authentication Key Establishment Protocol for Mobile IPv6", draft-perkins-bake-00.txt

[3] N. Modadugu, D. Boneh and M. Kim, "Generating RSA Keys on a Handheld Using an Untrusted Server", RSA 2000.

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

[5] NIST FIPS 186 : Digital Signature Standard (DSS), May 1994. http://www.itl.nist.gov/fipspubs/fip186.htm

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

[7] Frank Stajano and Ross Anderson, "The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks", Security Protocols Workshop, 1999.

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

[9] M. Bellare, R. Canetti and H. Krawczyk. "Message Authentication using Hash Functions – The HMAC Construction," RSA CryptoBytes, Vol. 2, No. 1, Spring 1996.

[10] Bob Moskowitz, "HIP Architecture", draft-ietf-moskowitz-hip-arch-02.txt

[11] Bob Moskowitz, "HIP Implementation", draft-moskowitz-hip-impl-01.txt

[12] Bob Moskowitz, "Host Identity Payload and Protocol", draft-moskowitz-hip-03.txt

[13] R. Hinden and S. Deering, "IP Version 6 Addressing Architecture," RFC 2373, July 1998.

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

[15] Castelluccia, Dupont, "A Simple Privacy Extension for Mobile IPv6," draft-castelluccia-mobileip-privacy-00.txt, February 2001.

[16] D. Johnson, C. Perkins, "Mobile IP for IPv6", draft-ietf-mobileip-ipv6-14.txt, July 2001. Work in progress.

[17] 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-01.txt, October 2001. Work in progress.

[18] T. Kivinen and M. Kojo, "More MODP Diffie-Hellman Groups for IKE", draft-ietf-ipsec-ike-modp-groups-01.txt

[19] P. van Oorschot and M. Wiener, "Parallel Collision Search with Applications to Hash Functions and Discrete Logarithms, " Proceedings of the 2nd ACM Conference on Computer and Communications Security, Fairfax, VA, November 1994.

[20] Tuomas Aura, Pekka Nikander and J. Leiwo, DOS-resistant authentication with client puzzles. In Bruce Christianson, Bruno Crispo, and Mike Roe, editors, Proceedings of the 8th International Workshop on Security Protocols, Lecture Notes in Computer Science series, Cambridge, UK, April 2000. Springer.

[21] Hugo Krawczyk, "Rationale for the definitions of SKEYID", message to the IPsec mailing list, June 20, 2001.

[22] Vern Paxson, "An Analysis of Using Reflectors in Distributed Denial-of-Service Attacks," 2001.

[23] Krawczyk, H., Bellare, M. and R. Canetti, ”HMAC: Keyed- Hashing for Message Authentication”, RFC 2104, February 1997.

[24] C. Madson and R. Glenn ”The Use of HMAC-SHA-1-96 within ESP and AH”, RFC 2404, November 1998.

[25] H. Orman, ”The OAKLEY Key Determination Protocol”, RFC 2412, November 1998.

[26] R. Pereira, ”The ESP CBC-Mode Cipher Algorithms”, RFC 2451, November 1998.

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

[28] NIST, FIPS PUB 180-1: Secure Hash Standard, April 1995. http://www.itl.nist.gov/fipspubs/fip180-1.htm

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