VIPR   J. Rosenberg
Internet-Draft   jdrosen.net
Intended status: Standards Track   C. Jennings
Expires: December 14, 2011   Cisco
                                                        M. Petit-Huguenin
                                                               Stonyfish
                                                           June 12, 2011

The Public Switched Telephone Network (PSTN) Validation Protocol (PVP)
                   draft-petithuguenin-vipr-pvp-01

Abstract

   One of the main challenges in inter-domain federation of Session
   Initiation Protocol (SIP) calls is that many domains continue to
   utilize phone numbers, and not email-style SIP URI. Consequently, a
   mechanism is needed that enables secure mappings from phone numbers
   to domains. The main technical challenge in doing this securely is
   to verify that the domain in question truly is the "owner" of the
   phone number. This specification defines the PSTN Validation
   Protocol (PVP), which can be used by a domain to verify this
   ownership by means of a forward routability check in the PSTN.

material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 14, 2011.

Copyright Notice

Table of Contents

1. Introduction

   The validation protocol is the key security mechanism in ViPR. It is
   used to couple together PSTN calls with IP destinations based on
   shared knowledge of a PSTN call. This document relies heavily on the
   concepts and terminology defined in [VIPR-OVERVIEW] and will not make
   sense if you have not read that document first.

   The protocol assumes that two enterprises, the originating one
   (enterprise O) initiates a call on the PSTN to an E.164 number
   ECALLED that terminates on the terminating enterprise (enterprise T).
   Each enterprise has a ViPR server, acting as a P2P node. The node in
   enterprise O is PO, and the node in enterprise T is PT. This PSTN
   call completes successfully, and knowledge of this call is known to
   PO and PT. Later on, PO will query the P2P network with number
   ECALLED. It comes back with a Node-ID PCAND for a node. At this
   time, PO can't know for sure that PCAND is in fact PT. All it knows
   is that some node, PCAND, wrote an entry into the DHT claiming that
   it was the owner of number ECALLED. The objective of the protocol is
   for PO to determine that node PCAND can legitimately claim ownership
   of number ECALLED, by demonstrating knowledge of the previous PSTN
   call. It demonstrates that knowledge by demonstrating it knows the
   start time, stop timer, and possibly caller ID for the PSTN call made
   previously.

```
                        /----------\
                      ///            \\\
                     ||                ||
                     |       ViPR        \
                     ||      DHT         ||\
                      X\\              ///   \
                      /    \----------/        \
                --------/-                  ----\------
              ///          \\\            ///          \\\
             //              \\          //              \\
             |              |///---\\\ |                  |
             |  Enterprise O |  PSTN   |   Enterprise T   |
             |              |\\\---/// |                  |
              \\            //          \\              //
               \\\        ///            \\\          ///
                -----+-----                  ------+----
                +---+----+                  +---+----+
                | Phone O|                  |Phone T |
                +--------+                  +--------+
```
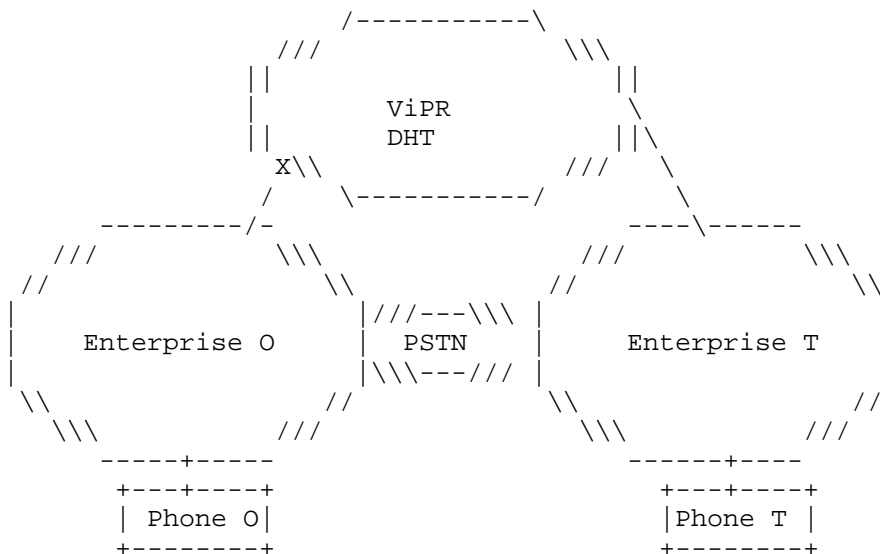
                     Figure 102: Validation Model

If node PCAND can demonstrate such knowledge, then enterprise O can
assume that node PCAND had in fact received the call, which could
only have happened if it had knowledge of the call to number ECALLED,
which could only have happened if PCAND is in enterprise T, and thus
it is PT. This is because PSTN routing is assumed to be "secure", in
that, if someone calls some number through the PSTN, it will in fact
reach a terminating line (whether it be analog, PRI, or other) which
is the rightful "owner" of that number. If enterprise T was not the
owner of the number, if would not have received the call, would not
know its start/stop/caller ID, not be able to provide that
information to PT, and not be able to satisfy the knowledge proof.
This basic approach is shown in Figure 102.

A first question commonly asked is, why not just do regular
authentication? What if we give each node a certificate, and then
have the nodes authenticate each other? The answer is that a
certificate certifies that a particular node belongs to a domain -
for example, that node PT is part of example.com. A certificate does
not assert that, not only is PT example.com, but example.com owns the
following phone numbers. Therefore simple certificate authentication
does not provide any guarantee over ownership of phone numbers.

In principle, it might be possible to ask certificate authorities,
such as Verisign, to assert just that. However, traditionally,
certificate authorities have been extremely hesitant to certify much
at all. The reason is, the certifier needs to be able to assure that
the information is correct. How can a certifier like Verisign verify
that, in fact, a particular enterprise owns phone numbers? It could
make a few test calls, perhaps, to check if they look right.
However, these test calls are disruptive to users that own the
numbers (since their phones will ring!). If the test calls are done
for a subset of the numbers, it is not secure. If the certifier
simply required, as part of the business agreement, that the
enterprises provided correct information, the certifier might avoid
legal liability, but the legitimacy of the service will be
compromised and customers will stop using it. Furthermore, it has
proven incredibly hard to do this kind of certification worldwide
with a single certificate authority.

ViPR has, as a goal, to work anywhere in the world and do guarantee
correct call routing with five nines of reliability. Consequently,
traditional certificates and authentication do not work. It turns
out to be quite hard to design a secure version of this validation
protocol. To demonstrate this, we will walk through some initial
attempts at it, and show how they fail.

2. The Wrong Way

   The first attempt one might make is the following. PO takes the
   caller ID for the call, ECALLING and called number ECALLED for the
   call, and sends them to candidate node PCAND. These two identifiers
   - the called number E and the caller ID, form a unique handle that
   can be used to identify the call in question. Node PCAND looks at
   all of the ViPR Call Records (VCRs) of the calls over the last 48
   hours, and takes those with the given called party number and calling
   party number. If there is more than one match, the most recent one
   is used. We now have a unique call.

   Now, node PCAND demonstrates knowledge of this call by handing back
   the start and stop times for this call in a message back to PO. This
   approach is shown in Figure 103.

```
              Po                Pt
               |                 |
               |                 |
               |                 |
               |Tell me start+stop
               |------------->|
               |                 |
               |                 |
               |                 |Retrieve records
               |                 |
               |                 |
               |                 |
               |start and stop|
               |<-------------|
               |                 |
               |                 |
               |                 |
               |                 |
```
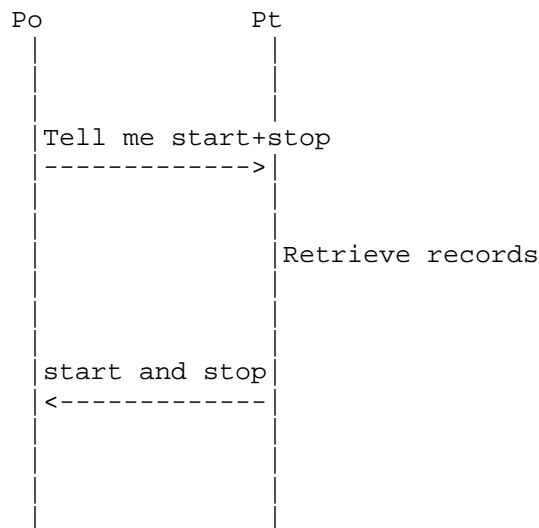
          Figure 103: Incorrect Validation Protocol: Take 1

   Unfortunately, this method has a major problem, shown in Figure 104.

```
            Po              Pbad            Pt             DHT
            |               |               |               |
            |               |               |               |
            |               |I own Ecalled  |               |
            |               |------------------------------>|
            |               |               |               |
            |               |               |I own Ecalled  |
            |               |               |------------>|  |
            |               |               |               |
            |Who owns Ecalled?              |               |
            |---------------------------------------------->|
            |               |               |               |
            |Pbad and Pt    |               |               |
            |<----------------------------------------------|
            |               |               |               |
            |Tell me start+stop             |               |
            |------------>|  |               |               |
            |               |               |               |
            |               |Tell me start+stop             |
            |               |------------>|  |               |
            |               |               |               |
            |               |               |Retrieve records
            |               |               |               |
            |               |               |               |
            |               |start+stop     |               |
            |               |<------------|  |               |
            |               |               |               |
            |start+stop     |               |               |
            |<------------|  |               |               |
            |               |               |               |
            |               |               |               |
            |               |               |               |
            |               |               |               |
```
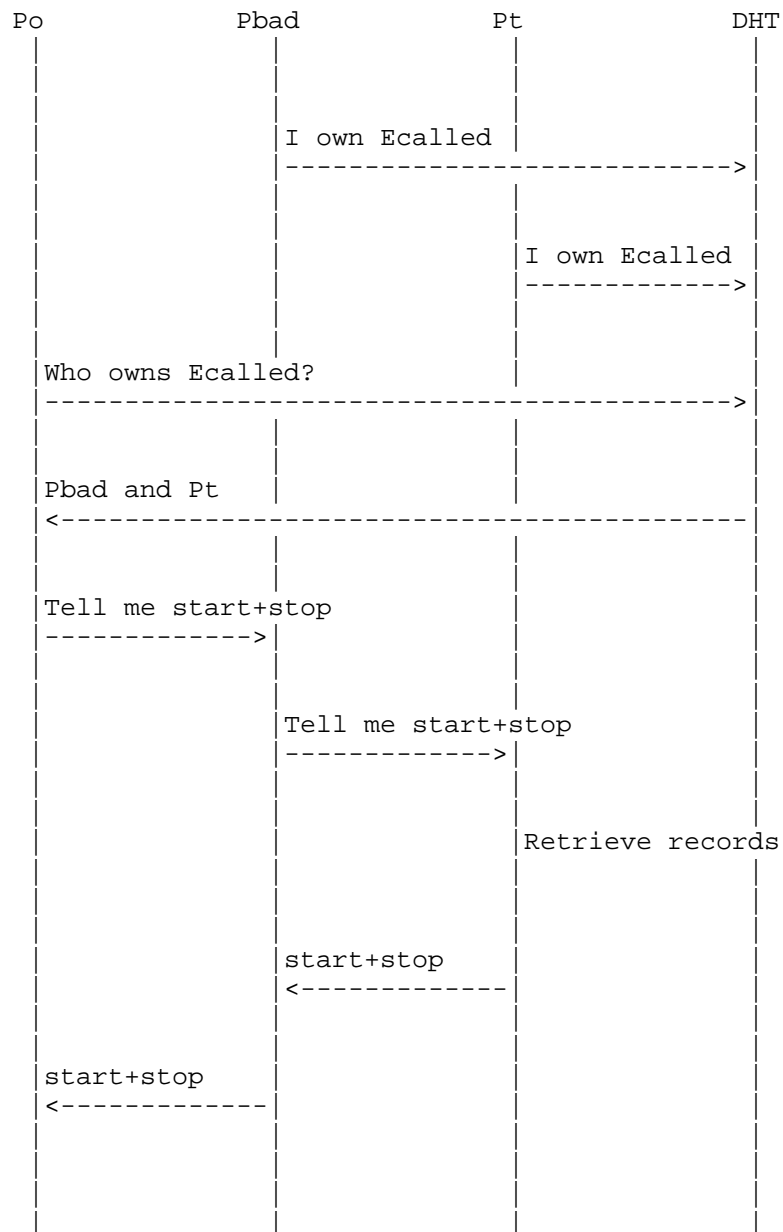
                Figure 104: Attack for Incorrect Validation Protocol

   Consider an attacker BadGuy PBAD. PBAD joins the P2P network, and
   advertises a number prefix they do NOT own, but which is owned by

enterprise T and node PT. Now, when PO queries the DHT with number
ECALLED, it comes back with two results - the one from PBAD and the
one from node PT. Details of querying the DHT are provided in
[VIPR-RELOAD-USAGE]. It begins validation procedures with both.
PBAD will now be asked to show the start and stop times for the call,
given ECALLED and ECALLING. It doesn't know that information.
However, node PT does. So now, PBAD, acting as if it where the
originating party, begins the validation protocol with node PT. It
passes the calling and called numbers sent by PO. PT finds a match
and returns the call start and stop times to PBAD. PBAD, in turn,
relays them back to PO. They are correct, and as a consequence, PO
has just validated PBAD!

Typically, the first response to this is, "Well the problem is, you
let two separate people write the same number into the DHT. Why
don't you make sure on the right one is allowed to write it in?".
That is not possible, since there is no mechanism by which an
arbitrary node in the DHT can determine who is the rightful owner of
this number. "OK", the reader responds, "So instead, why don't you
define a rule that says, if there are two entries in the DHT for a
particular number, consider this an attack and don't try to validate
the number". That would prevent the attack above. However, it
introduces a Denial of service attack. An attacker can pick a target
number, write it into the DHT, and prevent successful validation from
happening towards that number. They can't misroute calls, but they
can stop ViPR from working for targeted numbers. That is not
acceptable. ViPR has to be immune from attacks like this; it should
not be possible, through simple means such as configuration, for an
attacker to cause a targeted number to never be validated.

One might be tempted to add a signature over the call start and stop
times, but it does not help. BadGuy can just resign them and relay
them on.

In essence, this simple approach is like a login protocol where the
client sends the password in the clear. Such mechanisms have serious
security problems.

Realizing the similarities between the validation protocol and a
login protocol, a next attempt would be to use a much more secure
login mechanism - digest authentication. To do this, domain O takes
the called number E and the caller ID, and send them to node P. Node
P treats these as a "username" of sorts - an index to find a single
matching call. The start time and stop times of the call become the
"password". Enterprise O also sends a big random number - a nonce -
to node P. Node P then takes the random number, takes the password,
hashes them together, and sends back the hash. All of this is done
over a TLS connection between enterprise O and node P. Digest over

TLS is very secure, so surely this must be secure too, right? Wrong!

It is not. Indeed it is susceptible to EXACTLY the same attack
described previously. This is shown in Figure 105.

```
          Po              Pbad              Pt              DHT
          |                |                |                |
          |                |                |                |
          |                | I own Ecalled  |                |
          |                |----------------------------->|
          |                |                |                |
          |                |                | I own Ecalled  |
          |                |                |------------>|
          |                |                |                |
          | Who owns Ecalled?               |                |
          |----------------------------------------------->|
          |                |                |                |
          | Pbad and Pt    |                |                |
          |<-----------------------------------------------|
          |                |                |                |
          | TLS            |                |                |
          |------------>|                   |                |
          |                |                |                |
          | Login user=Ecaller+Ecalled     |                |
          |------------>|                   |                |
          |                |                |                |
          |                | Login user=Ecaller+Ecalled     |
          |                |------------>|                   |
          |                |                |                |
          |                |                | Retrieve records
          |                |                |                |
          |                |                |                |
          |                | Digest response |               |
          |                |<-------------|                  |
          |                |                |                |
          | Digest response|                |                |
          |<-------------|                  |                |
          |                |                |                |
          |                |                |                |
          |                |                |                |
          |                |                |                |
```
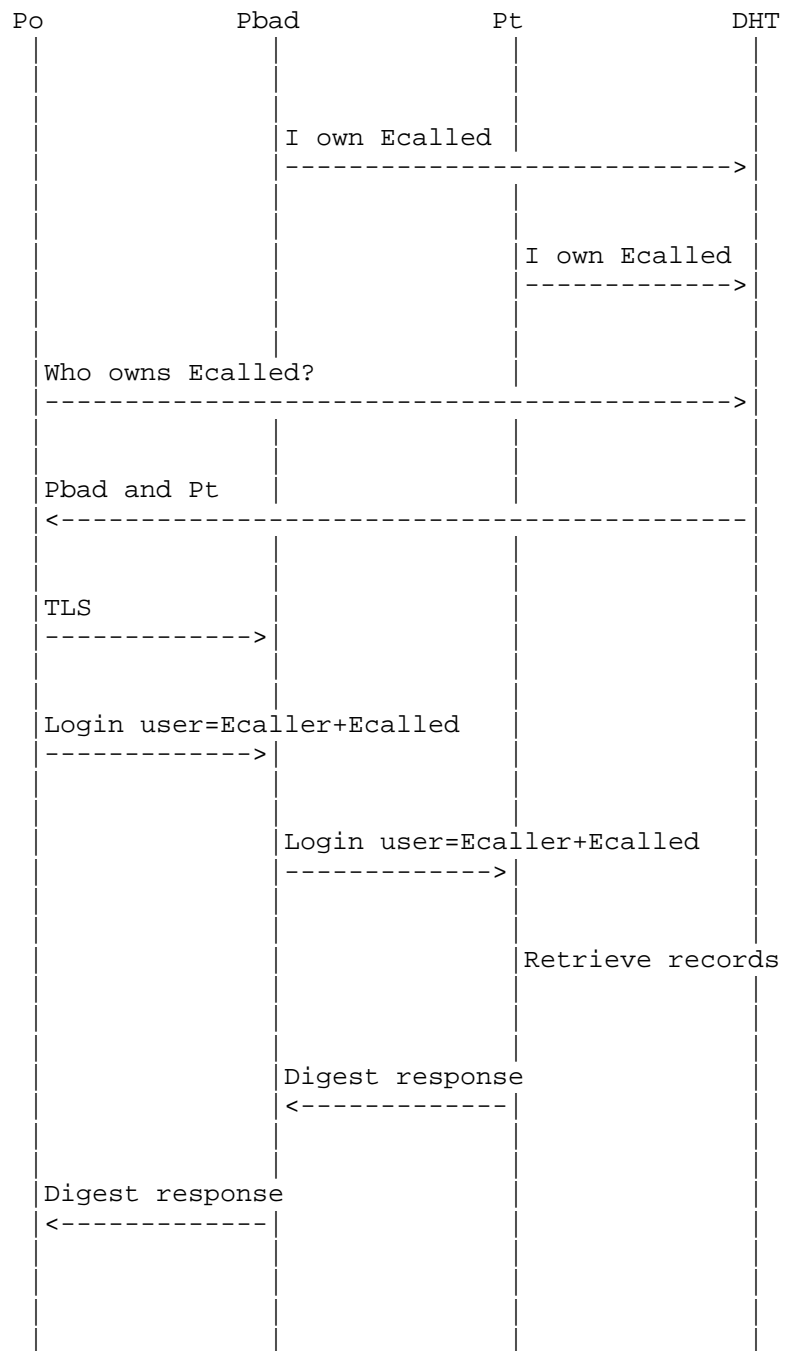
Figure 105: Trying Digest for Validation

In a similar attack, PBAD could pick a random called number it is
interested in, query the P2P network for it, find node PT. Then,
provide node PT the number ECALLED to attack, and ECALLING, assuming
it can guess a likely caller ID. It then takes the received digest
response, and goes through every possible start/stop time over the
last 24 hours, running them through the hash function. When the hash
produces a match, the PBAD has just found a full VCR for node PT. It
can then write into the DHT using number E as a key, pointing to
itself, and satisfy validation requests against it, without even
needing to ask node P again. Our first attempt is susceptible to
this attack too.

The problem here is that the call start and stop times have "low
entropy" - they are not very random and are easily guessable, just
like a poorly chosen password.

What we really want to do here is have a "login" protocol that
creates a secure connection between a client and a server, where we
use the called number and caller ID as a "username" to identify a
PSTN call, and then use the start and stop times as a "password".
But our login protocol has to have some key features:
1. Someone posing as a server, but which does not have the username
    and password, cannot determine the username and password easily
    as a consequence of an authentication operation started by a
    valid client, aside from successfully guessing in the one attempt
    it is given on each connection attempt.
2. Someone posing as a client, but which does not have the username
    and password, cannot determine the username and password as a
    consequence of an authentication operation started against a
    valid server, aside from guessing in the one attempt it is given
    on each TLS connection attempt.
3. An active MITM, who is explicitly on the path of the exchanges
    and has visibility and the ability to modify messages, cannot
    obtain the shared secret, nor can it observe or modify
    information passed between the client and real server.
4. It is impossible for a passive observer to view the exchange and
    obtain the shared secret or any of the material that is
    exchanged.
5. It is impossible for a rogue client or rogue server to
    participate in a login with a legitimate peer, and then take the
    messages exchanged, and run an offline dictionary attack to work
    through every possible combination of start and stop times.
    Fortunately, these properties are provided by a class of password
    authentication protocols called Encrypted Key Exchange or EKE
    protocols.

3. EKE Protocols

   EKE protocols were proposed in 1992 by Steve Bellovin. Since their
   proposal, numerous variations have been defined. One of them, the
   Secure Remote Password protocol, was standardized by the IETF in RFC
   2945 [RFC2945]. A TLS mode of SRP was later defined in RFC 5054
   [RFC5054]. It is the latter protocol which is actually used by ViPR.
   A high level overview of EKE protocols is shown in Figure 106. Alice
   and Bob share a shared secret P. Alice generates a public/private
   keypair. She then takes her public key, and encrypts it using her
   password as a symmetric encryption key. She sends this encrypted key
   to Bob. Bob, who shares the password, uses it as a symmetric key and
   decrypts the message, obtaining Alice's new public key. Bob then
   constructs a big random number R, which is to be used as a session
   key. Bob then encrypts R with the public key he just got from Alice,
   and sends that to Alice. Now, Alice, using her public key, decrypts
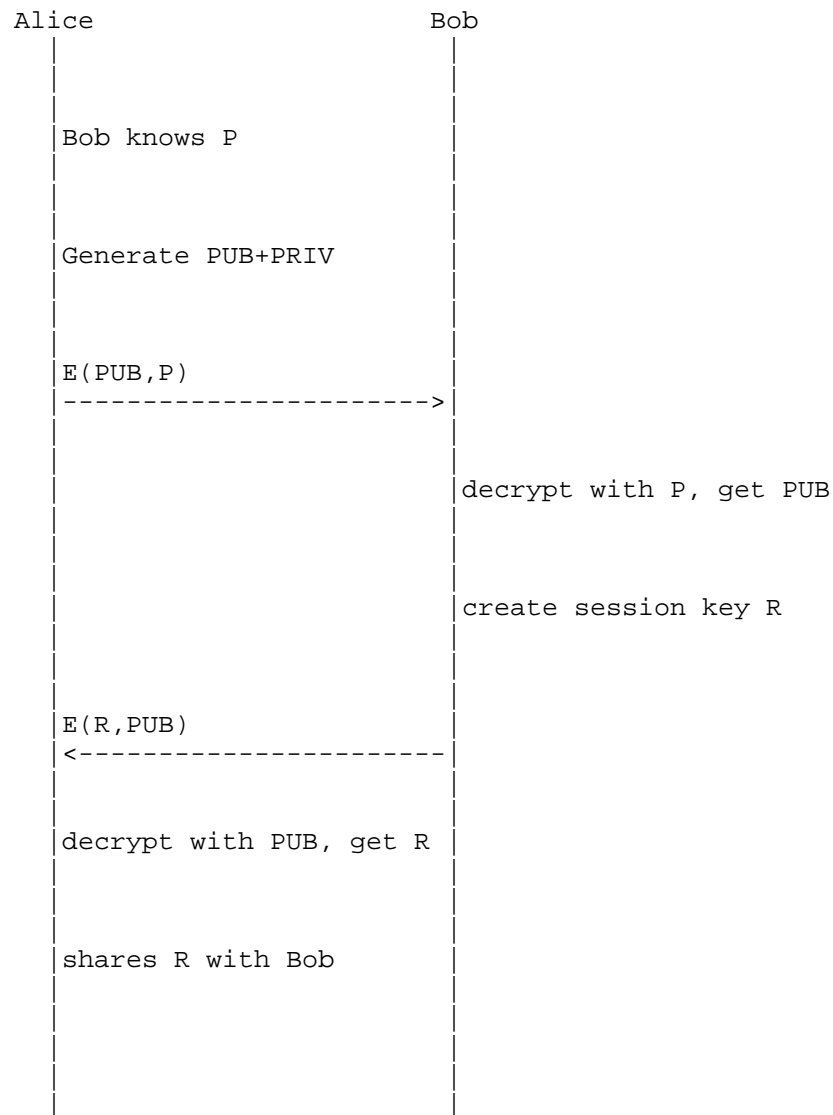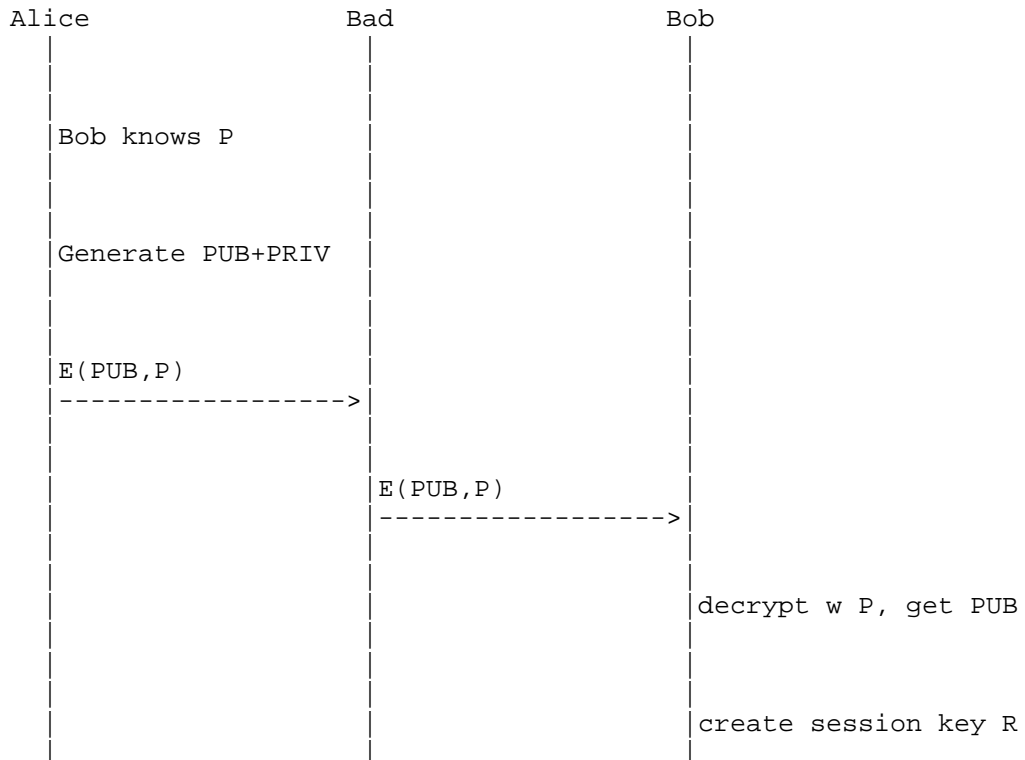   the message and obtains the session key R.

```
            Alice                    Bob
              |                       |
              |                       |
              |                       |
              |Bob knows P            |
              |                       |
              |                       |
              |                       |
              |Generate PUB+PRIV      |
              |                       |
              |                       |
              |                       |
              |E(PUB,P)               |
              |---------------------->|
              |                       |
              |                       |
              |                       |decrypt with P, get PUB
              |                       |
              |                       |
              |                       |
              |                       |create session key R
              |                       |
              |                       |
              |                       |
              |E(R,PUB)               |
              |<----------------------|
              |                       |
              |                       |
              |decrypt with PUB, get R|
              |                       |
              |                       |
              |                       |
              |shares R with Bob      |
              |                       |
              |                       |
              |                       |
              |                       |
              |                       |
```

Figure 106: High Level EKE Model

At this point Alice and Bob share a session key R which can be used
for authentication (by having Alice and Bob prove to each other that
they have the same value for R) or for encrypting data back and
forth. How does this help? Consider our man-in-the-middle attack
again, in Figure 107. Once again, Alice shares a password with
legitimate user Bob. However, she begins the "login" process with

BadGuy. She passes E(PUB,P) to BadGuy. BadGuy doesn't know P, so he
can't decrypt the message. More importantly, he can't run through
each possible password P and decrypt the message. If he did, he
wouldn't be able to tell if he got it right, since PUB appears
random; the decryption process would produce a random string of bits
whether it was successful or not. So for now, BadGuy can only pass
it on. BadGuy now intercepts E(R,PUB). Now, BadGuy can try the
following. He can run through each P, decrypt E(PUB,R), obtain PUB.
However, since we are using asymmetric encryption (i.e., public key
encryption), even with PUB he cannot DECRYPT E(R,PUB)! BadGuy does
not have the private key, which he needs to decrypt. Given a public
key, he cannot guess the private key either. That is how public/
private keying systems work. That is the secret here to making this
work. So, once again, BadGuy has no choice but to pass the message
on. Now, Alice and Bob share R but it is unknown to BadGuy. Bob now
takes his Node-ID, encrypts it with R, and sends to Alice. Once
again, BadGuy doesn't have R and can't get it, so he has no choice
but to pass it on. Alice decrypts this Node-ID with R, and now knows
that she is actually talking to Bob - since she has Bob's Node-ID.
Other data can be substituted for the Node-ID, and indeed this is
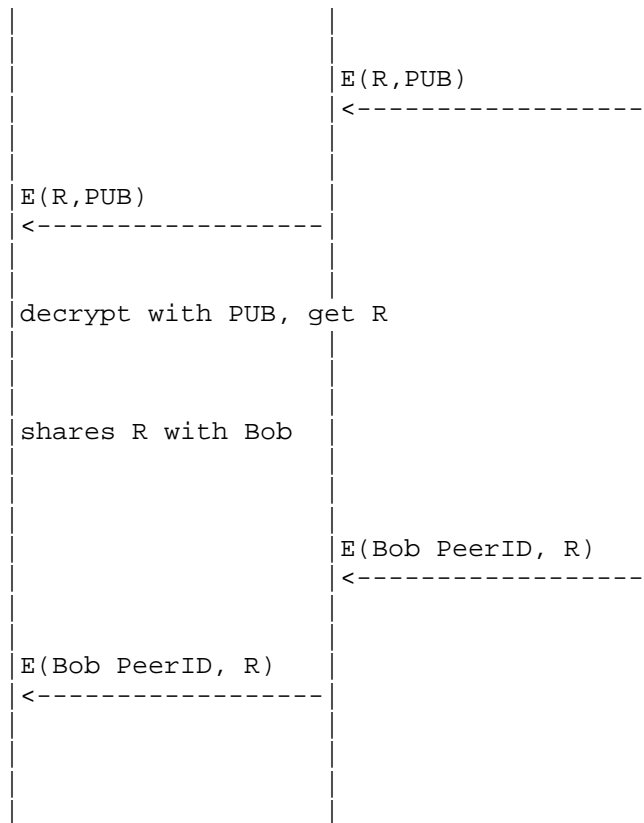what happens in the actual validation protocol.

```
         Alice                  Bad                  Bob
           |                     |                    |
           |                     |                    |
           |                     |                    |
           |Bob knows P          |                    |
           |                     |                    |
           |                     |                    |
           |                     |                    |
           |Generate PUB+PRIV    |                    |
           |                     |                    |
           |                     |                    |
           |                     |                    |
           |E(PUB,P)             |                    |
           |----------------->|                       |
           |                     |                    |
           |                     |                    |
           |                     |E(PUB,P)            |
           |                     |----------------->|  |
           |                     |                    |
           |                     |                    |
           |                     |                    |decrypt w P, get PUB
           |                     |                    |
           |                     |                    |
           |                     |                    |
           |                     |                    |create session key R
           |                     |                    |
```

```
|                   |                   |
|                   |                   |
|                   |E(R,PUB)           |
|                   |<------------------|
|                   |                   |
|                   |                   |
|E(R,PUB)           |                   |
|<------------------|                   |
|                   |                   |
|                   |                   |
|decrypt with PUB, get R                |
|                   |                   |
|                   |                   |
|                   |                   |
|shares R with Bob  |                   |
|                   |                   |
|                   |                   |
|                   |                   |
|                   |E(Bob PeerID, R)   |
|                   |<------------------|
|                   |                   |
|                   |                   |
|E(Bob PeerID, R)   |                   |
|<------------------|                   |
|                   |                   |
|                   |                   |
|                   |                   |
|                   |                   |
```

                    Figure 107: Attacking EKE Protocols

   However, the main point of this exercise is to demonstrate that EKE
   protocols have the desired properties.

4. Terminology

   The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
   "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
   document are to be interpreted as described in [RFC2119].

5. Protocol Overview

   The validation protocol begins with the following assumptions:

1. Node PO wishes to validate with node PCAND, and has its Node-ID
    (which it obtained via the DHT) and VServiceID (which it also
    obtained via the DHT Fetch).
2. Node PO and PCAND have a series of call records over the last 48
    hours, uploaded by their call agents. Each call record contains
    an E.164 calling and called party number, and a start and stop
    time in NTP time. On the terminating side, each call record is
    also associated with a VServiceID.
3. Node PO is seeking to validate a call to called number ECALLED
    with caller ID ECALLING.

The validation protocol operates by having the originating node make
a series of attempts to connect to, and "login" to the terminating
node. Each "login" attempt consists of establishment of a TCP
connection, and then execution of TLS-SRP procedures over that
connection. TLS-SRP[RFC5054] relies on a shared secret - in the form
of a username and password - in order to secure the connection. In
ViPR, the username and password are constructed by using information
from a target VCR along with the VServiceID learned from the DHT.
The "username", instead of identifying a user, identifies a
(hopefully) unique VCR shared between the originating and terminating
nodes. The "password" is constructed from the VCR such that it
knowledge of the information is unique to knowledge of the VCR
itself.

Unfortunately, it is difficult to construct usernames and passwords
that always uniquely identify a VCR. To deal with this, the
validation protocol requires the originator to construct a series of
usernames and passwords against a series of different nodes and their
corresponding IP addresses and ports, and then run through them until
a connection is securely established.

6. Username and Password Algorithms

ViPR provides two different algorithms for mapping from a particular
VCR to a username and password:

1. Method A: This method makes use of the called party and calling
    party number to form the username, and the start and stop times
    of the call to form the password.
2. Method B: This method makes use of the called party number,
    along with a point in time in the middle of the call, as the
    username, and then the start and stop times to form the password.

The originating node will first try validations with method A, and if
those all fail, then try with method B. The method itself, along with
necessary information on how to use the method, is encoded into the

username itself. The format of the username is (using ABNF
[RFC4234]):

```
username = method-a / method-b / future-method
future-method = method ":" method-data
method = 1ALPHA
method-data = 1*(alphanum / method-unreserved)

method-a = "a:" vserviceid originating-number terminating-number
           rounding-time
method-b = "b:" vserviceid terminating-number timekey rounding-time

vserviceid = "vs=" 1*32HEXDIG ";"
originating-number = "op=+" 1*15DIGIT ";"
terminating-number = "tp=+" 1*15DIGIT ";"
timekey = "tk=" 1*10DIGIT "." 1*10DIGIT ";"
rounding-time = "r=" 1*6DIGIT ";" ; Cannot be equal to 0
```

This format starts with the method, followed by a colon, followed by
a sequence of characters that are specific to the method. Both
methods a and b rely on conveyance of information attributes that
make up the username. Each attribute follows a specific format.

Examples include:

```
a:vs=7f5a8630b6365bf2;op=+17325552496;tp=+14085553084;r=1000;
b:vs=7f5a8630b6365bf2;tp=+14085553084;tk=172636364.133622;r=1000;
```

Both methods use a rounding factor R. This is used to round the start
and stop times in the password to a specific nearest multiple of R
(which is in milliseconds). This rounding is done because the
passwords need to be bit exact and we need to compensate for
different measured values.

If we will fallback to method B (which works more often), why have
both? There are two answers:

1. The caller ID mechanism (method A) will work, and the non-caller
    ID (method B) won't work, for numbers like 8xx.
2. Method A has much higher entropy (see analysis in Section 10.1).
    Validating with it provides greater confidence in the validity of
    the number. In this phase, nothing is done with this
    "confidence". However, in later phases, it is anticipated that
    low-confidence numbers will require multiple validations for
    different calls to occur before they are trusted. To allow for
    this feature to be added later, validation with both methods must
    be present in the initial release.

The sections below detail precisely how these are constructed.

7. Originating Node Procedures

   Most of the work for validation is on the side of the originator. It
   establishes connections and performs a series of validation checks.

7.1. Establishing a Connection

   The first step in the process is to establish a TCP connection to
   PCAND. To do that, PO sends a RELOAD AppAttach message targeted
   towards PCAND, using the Application-ID defined in
   [VIPR-RELOAD-USAGE]. Once connected, TLS-SRP is run over the
   connection.

7.2. Constructing a Username and Password

   When a terminating node receives a username in a format it doesn't
   understand, it fails the validation. This allows for graceful
   upgrade to new mechanisms in the future.

7.2.1. Method A

   The PO examines the VCR it is using for validation. It extracts the
   calling and called party numbers, both of which are E.164 based.
   This VCR will have been uploaded at a previous point in time. PO
   then examines the VCRs posted in the time since this one was
   uploaded, and looks for any more recent VCRs with the same calling
   and called party numbers regardless of VService. If it finds one or
   more, it takes the most recent one (as measured by its end time). If
   it finds no more recent, it uses the VCR which triggered the
   validation in the first place.

   Why do this? This deals with the following case. User A calls user
   B, causing a VCR to be uploaded. The originating node sets a timer,
   which fires 12 hours later. However, within that 12 hour period, A
   called B again. If node A provides the caller ID and called party
   numbers as the "key" to select a VCR, it will match multiple records
   over the past day. We need to pick one, so the most recent is always
   used. This requires the originating node to know and use the most
   recent VCR. Furthermore, we must choose the most recent VCR
   regardless of its VService, because the originating Upload VCRs are
   sent using an arbitrary VService. Thus, the more recent call may
   have been done using a different VService than the one which
   triggered the validation. Since the actual Vservices are not common
   between originating and terminating sides, we must choose the VCR on
   the originating side regardless of VService. The username is

constructed by using the syntax for method A described above. The
calling party number is set to "op", and the called party number is
set to "tp", and "r" is the value of Tr as an integral number of
milliseconds. The VServiceID learned from the dictionary entry is
used as the value of "vs".

This username will select the identical VCR at the terminating node,
under the following conditions:

1. PT is aware of all calls made to the called party number. This
    property is true so long as each incoming number is handled by a
    single call agent within a domain, and furthermore, the VCR for
    calls to that number is always posted to a ViPR server which
    advertises that number into the DHT. These properties are
    readily met by ViPR for typical single user numbers. For 8xx
    numbers, which are translated within the PSTN and may route to a
    multiplicity of non-8xx numbers, it is more difficult. ViPR will
    only work with 8xx numbers if all calls to those numbers get sent
    to agents which share the same ViPR server.
2. PO is aware of all calls made to the called party number with the
    given caller ID. This property can be hard to meet. If the
    caller ID for a call is set to the number of the calling phone,
    and all VCRs made from that phone are posted to the same ViPR
    server, that server will know about all calls made by the domain
    with the given DID in the caller ID. However, in domains that
    set the PSTN caller ID to the attendant line number, it is
    possible that there would be two separate agents, each utilizing
    different ViPR servers. A user in each agent calls the same
    number, and the same PSTN caller ID is used. However, one ViPR
    server knows about one of the calls, and a different ViPR server
    knows about the other call. However, PT knows about both. In
    that case, validation from one of the ViPR servers will fail, and
    from the other, succeed.
3. There were no calls on the PSTN to the called party which spoofed
    the caller ID to match the caller ID used by the valid
    enterprise. In that case, PT will have a VCR for a call with a
    matching calling/called party number, but this VCR is unknown to
    PO since the call was not actually made by the originating
    enterprise. This attack is described in more detail in XXXX.

Next, the password is selected. The password is basically the start
and stop times for the call. However, the SRP protocol requires a
bit exact agreement on the password. Unfortunately, the calling and
called parties will not have the same values for the start and stop
times, for several reasons:

1. The call start time at the originating and terminating ends will
    differ by the propagation delay of the call acceptance message
    through the PSTN. This can be several hundreds of milliseconds.
2. The clocks at the originating and terminating ends may not be
    synchronized, which can also introduce different values for the
    start and stop times.
3. The call termination time at the originating and terminating ends
    will also differ by the propagation time; this propagation time
    may in fact be different for the call acceptance and termination.

It is also important to note that agreement on a call acceptance and
termination time assumes an explicit signaling message is sent for
these two events. In the case of analog FXO ports, there is no
signaling at all, and consequently, these points in time cannot be
measured. It is possible to agree upon other call characteristics
when analog lines are in use, but they have much worse accuracy and
consequently much, much lower entropy. For this reason, this
specification of ViPR only works in telephony systems with explicit
messaging for call acceptance and termination, which includes PRI,
SS7, BRI, analog trunks with answer and disconnect supervision, and
CAS trunks.

To deal with these inaccuracies in timing, the start and stop times
need to be rounded. Let Tr be the rounding interval, so that each
time is rounded to the value of N*Tr for integral N, such that N*Tr
is less than the start or stop time, and (N+1)*Tr is greater than it.
In other words, "round down". If Tr=1 second, this would round down
to the nearest second.

Unfortunately, rounding doesn't fully help. Lets say that the
difference between the start times on the originating and terminating
nodes is delta. We can still have different values for the start
time if one side rounds to one value, and the other side to a
different value. If delta=100ms and Tr=1s, consider a start time of
10.08 seconds on one side, and 9.98 on the other side. One side will
round to 10 seconds and the other to nine seconds. The probability
of this happening is approximately delta/Tr. We could just make Tr
really large to compensate, but this reduces the entropy of the
system (see below).

To deal with this, the originating node will actually compute FOUR
different passwords. For the start time and stop time both, the
originating node will round down as follows. Let T be the time in
question. Let N be the value such that N*Tr <= T < (N+1)*Tr. In
other words, N*Tr is the nearest round-down value, and (N+1)*Tr is
the nearest round up. Let T1 and T2 be the two rounded values of T.
We have:

```
if (T >= ((2N+1)/2) * Tr) then:
                        T1 = N*Tr
                        T2 = (N+1)*Tr
if (T <  ((2N+1)/2) * Tr) then:
                        T1 = N*Tr
                        T2 = (N-1)*Tr
```

In other words, if T is in the top half of the rounding interval, we try the rounded values above and below. If T is in the bottom half, we try the rounded values below, and below again. Pictorially:

[[TBD]]

         Figure 108: Rounding mechanism for validation protocol

These are tried in the following sequence:

1. Try Tstart-1 and Tend-1.
2. Try Tstart-2 and Tend-1.
3. Try Tstart-1 and Tend-2.
4. Try Tstart-2 and Tend-2.

For example, if the originating side has a start time of 10.08 and a stop time of 30.87, the four start and stop times with Tr=1s are:

```
              +-------+------+
              | Start | Stop |
              +-------+------+
              | 10    | 30   |
              | 9     | 30   |
              | 10    | 31   |
              | 9     | 31   |
              +-------+------+
```

Each of these times is represented in 64 bit NTP time (Tr can be configured to less than 1s in which case there will be non-zero values in the least significant 32 bits). Each password is then computed by taking the 64 bit start time, followed by the 64 bit end time, resulting in a 128 bit word. This word is base64 encoded to produce an ASCII string representation of 21 characters. To perform the caller ID based validation, the SRP-TLS procedure is done four times, once with each of the four username/password combinations (of course the username is identical in all four cases). As long as delta is less than Tr/2, one of this is guaranteed to work.

7.2.2. Method B

   Unfortunately, in many cases caller ID cannot be used as an
   identifier for the VCR. This is because:

   1. CallerID is frequently suppressed in the PSTN, and not delivered.
       This is especially true in international cases.
   2. CallerID is sometimes munged by the PSTN, and delivered, but with
       a different value than was sent by the originator. This happens
       in certain arbitrage interexchange carriers.

   Consequently, if no caller ID was delivered at all, the terminating
   side will not have a matching record. In that case, it informs the
   calling side that it should abort and revert to method B. If munged,
   it will also abort for the same reason.

   If the caller ID attempt aborts, PO now tries a different approach.
   In this approach, the "username" is the combination of the called
   party number and a point during the call, selected at random. The
   password is equal to the start and stop times of the call. This
   method uses the method-tag "B" in the username.

   Unlike method A, with method B, the VCR which triggered the
   validation is used, regardless of whether there were other, more
   recent, calls to the same calledparty number! This is because, in
   method B (unlike method A), the time itself is used as a key to
   select a VCR. Furthermore, using a more recent VCR does not interact
   properly with multi-tenancy. The called number and point during that
   call will select an identical VCR on the terminating side if the
   following conditions are met:

   1. For the called party number, there was not more than one call in
       progress made to that number at the same time. This is generally
       true for numbers for a single user; typically there is only one
       active call at a time. Of course, it is possible a user receives
       a call, and then gets another. It then puts the first on hold
       while the second call is taken. In these cases, it is possible
       that the "username" will select a different VCR on PT, in which
       case the validation fails. More troubling are numbers
       representing call centers, conference bridges, 8xx numbers, and
       attendant numbers, all of which frequently have multiple calls in
       progress to them at the same time. As a consequence, for these
       types of called numbers, validation is typically only going to
       work if caller ID is delivered. Fortunately, 8xx numbers are
       only national in the first place, so it is likely that this will
       work.

   2. PO is aware of all calls made from within its enterprise to
       ECALLED. This can fail if there are multiple ViPR servers
       serving different agents, and a call is made from one agent, sent
       to one ViPR server, and a call to that same number is made on a
       different agent, send to a different ViPR server. As in the
       caller ID case, this will still be OK in many cases - the
       validation from one ViPR server succeeds, the other fails.
   3. PT is aware of all calls made to ECALLED. The same caveats as
       described above for the caller ID mechanism apply. PO takes the
       VCR, and chooses a time Tkey which is uniformly distributed
       between Tstart+Tr and Tstop-Tr. The usage of the Tr here is to
       make sure that Tkey is squarely inside of the call start and stop
       for PT as well. Note that, because Tkey is not a password, it is
       sent in the clear and does not need to be rounded.

   The username encodes the called party number, Tkey, the DHT, and the
   VServiceID learned from the DHT query. The password is computed
   identically to method A.

7.3. Requesting Validation

   Once the SRP-TLS connection is up, data is exchanged. This is done
   through a single VAP transaction initiated by PO. This transaction
   is only VAP in the sense that it utilizes the basic syntax (the
   header and TLV attribute structure), and its request/response model.
   Other than that, it is effectively a different protocol - the
   validation protocol.

   PO sends a VAP request with method ValExchange (0x00d). It contains
   one attribute, Domain. The originating ViPR server obtains this
   domain by looking at the VService of the VCR that was eventually used
   for the validation. Note that, in cases where the VCR which
   triggered the validation, is different than the one actually used for
   validation (because a more recent VCR to the same number was found),
   it is important to use the VService associated with the VCR which was
   actually used for validation, and NOT the VService associated with
   the VCR which triggered the attempt. Multi-tenancy does not work
   properly without this. The domain from the VService is placed into
   the message. This is basically the domain name of the originating
   enterprise. It is included since it is needed by PT to compute the
   ticket.

   PO will then receive a response. If it never receives a response
   within a timeout, it considers the validation to have failed, and
   continues to the next choice. If it receives any kind of error
   response, including a rejection due to a blacklisted domain, it
   considers validation to have failed, and continues to the next
   choice. If it is a success response, it will contain one attribute -

ServiceContent, which contains a ValInfo XML object. ValInfo is an
XML object which contains the SIP URIs and the ticket. The ViPR
server must parse the ValInfo XML object and perform verification on
it to avoid attacks. The following checks are done:

1. Extract the <number> element. This will contain a single number.
    That value is compared with the E.164 called party number which
    was just validated. If they do not match, this is a potential
    attack, and the XML is discarded and the ViPR server acts as if
    validation failed. However it does not generate an alarm.
2. Remove any extensions to the XML which are not supported by the
    ViPR server (no extensions defined, so in this version, any
    elements except for the <ticket>, <number>, <route>s and their
    embedded <SIPURI> are removed.
3. Verify that the <route> element contains a single element,
    <SIPURI>.
4. Verify that the SIP URI is not larger than 614 characters,
    contains a domain name that is a valid set of domain name
    characters, contains a user part that is a valid set of
    characters, if it contains maddr, that the maddr is a valid
    domain or IP and less than 255 characters, and if there is a
    port, it is within 0-65535. This is for security purposes; to
    make sure a malicious ViPR server on the terminating side cannot
    send invalid URI and attack the call agent.
5. Verify that each SIP URI contains the same domain name. Once the
    checks and fixes are done, the patched XML is passed to
    subscribers in a Notify as described in [VIPR-VAP].

8. Terminating Node Procedures

8.1. Waiting for SRP-TLS

   PT will wait for an AppAttach request on the Application-ID defined
   in [VIPR-RELOAD-USAGE] and the connection is established, it begins
   waiting for SRP-TLS. The TLS messaging will provide PT with a
   username.

   It parses the username and determines the method. If the value of
   the method is not "a" or "b", this is a new method not supported by
   the node. The SRP-TLS procedures should be failed. If the method is
   "a", it is the caller ID mechanism. The called number, calling
   number, VService, and rounding time are extracted. PT then searches
   through its VCRs over the last 48 hours for one with a matching
   called number and caller ID and VService whose VServiceID matches the
   one from the username:

1. If none are found, PT proceeds with the SRP-TLS exchange, but
    using a fake username and password. This will cause the
    validation to eventually fail.
2. If one is found, it is used.
3. If more than one is found, the one with the most recent end time
    is used.

The start and stop times from the selected VCR are taken. Using the
value of Tr from the username, both times are rounded down to the
nearest multiple of Tr. Note that, this rounding is different than
the one used on the originating side. The values are ALWAYS rounded
down. So if the stop time is 10.99 and Tr is one second, the rounded
down value of 10 is used. The start and stop times are then
represented as 64 bit NTP times (after rounding), concatenated, and
base64ed to produce a 21 character password. This is the password
used with SRP-TLS.

Note that, the originating node will try up to four different
password combinations. One of these should work, the others will
cause SRP-TLS to fail due to differing shared secrets. However, it
is the job of the originator to perform these four; to the
terminating node, they are four separate attempts. Processing of
SRP-TLS login attempts is stateless on the terminating side. This
means that each attempt is treated independently by PT. It performs
identical processing on each SRP-TLS attempt - examine the username,
find a matching VCR, extract password, and fail the attempt or
continue to success. The originating side has the main burden of
sequencing through the various mechanisms.

If the method is "b", PT uses the extracted called party ID and a
time in the middle of the call. It searches through all VCR records
whose called number matches and whose VServiceID matches, and of
those, takes the ones where Tkey is between Tstart and Tstop. Of
those, if more than one match, the one with the most recent Tstop is
used. Tstart and Tstop for that VCR are extracted, and converted to
a password just as is done for the PO. The resulting SRP-TLS
procedure will then either succeed or fail. Note that, if a domain
has multiple Vservices that contain the same number, there will be
multiple VCRs for calls to that number, and there will be multiple
validation attempts, one for each of the Vservices.

Note that there could be multiple successful validations coming from
different domains for one specific VCR, so VCRs should not be removed
before the end of the 48 hours period. This can happen when a
calling domain uses a PSTN provider that is itself VIPR enabled.

8.2. Receiving Validation Requests

   PT listens for incoming VAP/validation requests once the TLS
   connection is up. It rejects anything but a ValExchange method with
   a 400 response. This allows for future extensibility of the
   validation protocol. If the request was ValExchange, it extracts the
   domain name. This will be something like "example.com". PT knows
   the VCR against which validation succeeded. That VCR is associated
   with a VService. The ViPR server checks the domain in the
   ValExchange request against the black/white list associated with that
   VService. If no VService is currently active, the ValExchange is
   rejected with a 403. If there is one active, and if the domain
   appears on the black list, or does not appear in the white list, the
   ViPR server rejects the ValExchange request with a 403 error
   response, indicating that this domain is not allowed to call.

   If the domain was in the whitelist or not in the blacklist, or there
   was no whitelist/blacklist, PT constructs a successful response to
   the ValExchange request. It contains one attribute: ServiceContent.
   It has a ValInfo XML object, which contains a number, a ticket, and a
   series of routes.

   The number is always the E.164 number which was just validated,
   including the plus sign. Note that this will also appear in the
   ticket. The route element is the sequence of route elements for each
   instance associated with the vservice.

   Details of the ticker are provided in [VIPR-SIP-ANTISPAM] but the
   ticket attribute is constructed as follows:
   1. A ticket unique ID TLV is created, containing a randomly chosen
       128 bit value as the ticket ID. That is the first TLV in the
       ticket.
   2. A salt TLV is created, containing a random 32 bit value. This is
       the second TLV in the ticket.
   3. The validity has the start time set using the current time as the
       start time, and the current time + the ticket lifetime as the end
       time. The ticket lifetime is a per-DHT configurable parameter.
       The terminating ViPR server will have performed the validation
       using a particular VService; the DHT for that VService is used to
       find the right value for this parameter.
   4. Number: This is the terminating number, in E.164 format, which
       was just validated.
   5. Granting node: this is set to one of the Node-IDs associated
       with this ViPR server. Any will do.
   6. Granting domain: This value is taken from the domain part of the
       SIP URI associated with the VService in which the validated VCR
       was found.

   7. Granted-To domain: This is formed using the Domain sent in the
       ValExchange request.
   8. Epoch: This is the current epoch associated with the password.
   9. Integrity: Using the current password, this is computed from the
       rest of the Ticket.

   The resulting sequence of TLVs is base64 encoded and that is placed
   into the ticket element in the ServiceContent attribute in the
   ValExchange response.

9. Syntax Details

   This section enumerates the methods and attributes used by PVP.

   The methods and their corresponding method values, are:

   Method   Value
   ------   ------
   ValExchange   0x00d

                        Figure 1: PVP Methods

   The attribute names and corresponding types are:

   Attribute Name   Type
   --------------   ----
   Domain   0x3001

                        Figure 2: PVP Attributes

10. Security Considerations

   [[This section is mostly missing and needs to be done.]]

10.1. Entropy

   [[The entropy obtained in the information from the PSTN calls
   significantly impacts the security of this protocol. This section
   needs to provide an analysis of how much entropy actually exists in
   this information.]]

   [[Defines the worst case of conference calls and resulting entropy]]

   [[Describe the idea of doing multiple validations to aggregate
   entropy]]

10.2. Forward Routing Assumptions

   [[Discuss forward routing security in PSTN and explain how this
   protocol is reliant on that.]]

11. IANA Considerations

   [[TBD Define ports used.]]

12. Acknowledgements

   Thanks to Patrice Bruno for his comments, suggestions and questions
   that helped to improve this document.

13. References

13.1. Normative References

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
              Requirement Levels", BCP 14, RFC 2119, March 1997.

   [RFC4234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax
              Specifications: ABNF", RFC 4234, October 2005.

   [RFC5054] Taylor, D., Wu, T., Mavrogiannopoulos, N., and T. Perrin,
              "Using the Secure Remote Password (SRP) Protocol for TLS
              Authentication", RFC 5054, November 2007.

   [VIPR-OVERVIEW]
              Jennings, C., Rosenberg, J., and M. Petit-Huguenin,
              "Verification Involving PSTN Reachability: Requirements
              and Architecture Overview",
              draft-jennings-vipr-overview-00 (work in progress),
              April 2011.

   [VIPR-VAP]
              Jennings, C., Rosenberg, J., and M. Petit-Huguenin,
              "Verification Involving PSTN Reachability: The ViPR Access
              Protocol (VAP)", draft-jennings-vipr-vap-00 (work in
              progress), April 2011.

   [VIPR-SIP-ANTISPAM]
              Rosenberg, J., Jennings, C., and M. Petit-Huguenin,
              "Session Initiation Protocol (SIP) Extensions for Blocking
              VoIP Spam Using PSTN Validation",

                    draft-petithuguenin-vipr-sip-antispam-01 (work in
                    progress), June 2011.

    [VIPR-RELOAD-USAGE]
                    Rosenberg, J., Jennings, C., and M. Petit-Huguenin, "A
                    Usage of Resource Location and Discovery (RELOAD) for
                    Public Switched Telephone Network (PSTN) Verification",
                    draft-petithuguenin-vipr-reload-usage-01 (work in
                    progress), June 2011.

13.2. Informative References

    [RFC2945]   Wu, T., "The SRP Authentication and Key Exchange System",
                RFC 2945, September 2000.

Appendix A. Release notes

    This section must be removed before publication as an RFC.

A.1. Modifications between vipr-01 and vipr-00

    o Added text explaining that VCRs should not be removed before the
       end of the 48 hours delay.
    o Inserted Terminology section.
    o Fixed the timekey ABNF.
    o Specified that rounding-time cannot be equal to 0.

A.2. Modifications between vipr-00 and dispatch-03

    o Moved to new Working Group.

A.3. Modifications between dispatch-03 and dispatch-02

    o Nits.
    o Shorter I-Ds references.
    o Removed sentence saying that Tkey is converted to base64.
    o Added ValExchange method and Domain attribute definitions.
    o Fixed the last sentence of 7.2 - the ticket goes into the ticket
       element in the ServiceContent attribute.
    o Expanded first usage of VCR initialism.
    o Replaced any insteance of peerID by Node-ID.
    o Rewrote the ABNF.

Authors' Addresses

    Jonathan Rosenberg
    jdrosen.net
    Monmouth, NJ
    US

    Email: jdrosen@jdrosen.net
    URI: http://www.jdrosen.net

    Cullen Jennings
    Cisco
    170 West Tasman Drive
    San Jose, CA 95134
    USA

    Phone: +1 408 421-9990
    Email: fluffy@cisco.com

    Marc Petit-Huguenin
    Stonyfish

    Email: marc@stonyfish.com