Improving the Open Source HIPv2 implementation

TDDE21 - Secure Distributed and Embedded Systems

Gustaf Bodemar, Nils Lenti, Tony Lindbom,
Hannes Linde, Max Skanvik, Frans Öhrström

Examiner: Andrei Gurtov
Supervisor: Mohammad Borhani

December 18, 2023

Contents

1 Introduction and Background

This report is the result of a project as part of the course *TDDE21: Advanced Project: Secure Distributed and Embedded Systems* given the year 2023 at Linköping University (LiU) by the Department of Computer and Information Science [7]. The course has been taught for a few years at this point (seemingly since 2017). This year, students were divided into two groups working on one project each, whereby this project revolved around improving the open source implementation of the Host Identity Protocol (OpenHIP).

1.1 HIP

Host Identity Protocol (HIP) is a secure network protocol that uses asymmetric encryption keys for identification of hosts, and additionally provides a symmetrically encrypted layer for communication between hosts. It allows for the separation of a host's identity (HI) from the location identity (how to reach the host currently). In traditional networks, both of these identities are bound to an IP address, but distinguishing between the two provides a major benefit: the actual host identity becomes more stable and viable long term (similarly to the name of a person). This allows computers to identify one another and even to stay connected while the location of a host (its IP address) changes, enabling both multi-homing and increased mobility of devices. The encrypted tunnel for secure communication is established through a 4-way base exchange (the HIP BEX), based on a Diffie Hellman key exchange, as depicted in figure 1. The latest version of HIP, HIPv2, is described in *RFC 5201* [4].

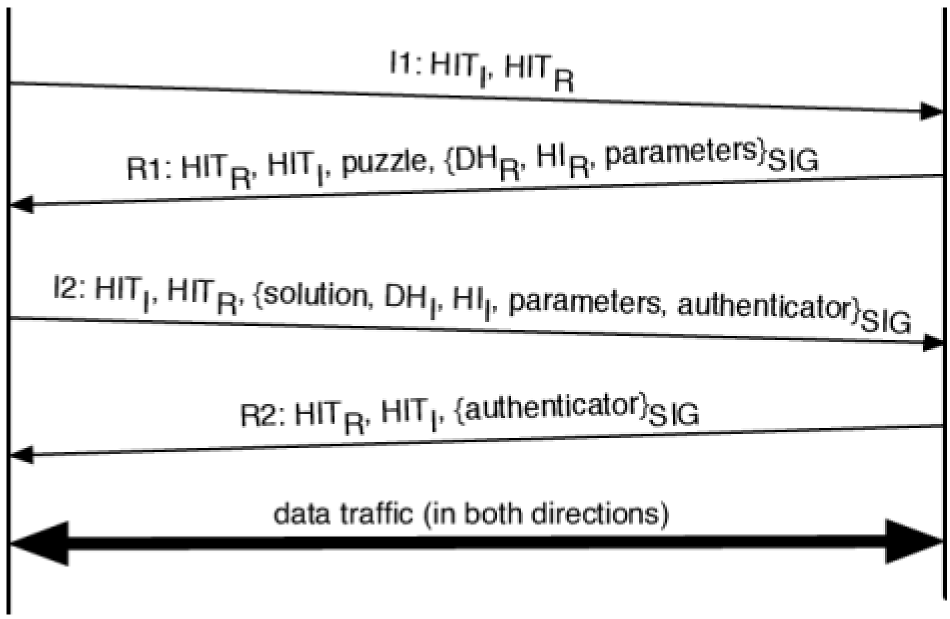

Figure 1: BEX (from 2022 report)

1.2 OpenSSL

OpenSSL is a popular and important open source library that primarily provides a number of different cryptographic algorithms. The OpenSSL Project's official website (OpenSSL Project) states

"The OpenSSL Project develops and maintains the OpenSSL software - a robust, commercial-grade, full-featured toolkit for general-purpose cryptography and secure communication".

1.3 CORE

As formulated by the official documentation [1],

> *CORE (Common Open Research Emulator) is a tool for building virtual networks. As an emulator, CORE builds a representation of a real computer network that runs in real time, as opposed to simulation, where abstract models are used. The live-running emulation can be connected to physical networks and routers. It provides an environment for running real applications and protocols, taking advantage of tools provided by the Linux operating system.*

In this project, CORE (see CORE GitHub) is used for testing and debugging OpenHIP by creating an emulated network with two hosts connected via a switch. The tool can alternatively be used through a graphical user interface (GUI), but with great help from efforts of the 2022 group, a Python script (test/debug.py) was created for debugging OpenHIP that sets up the virtual network automatically.

1.4 Docker

Docker is a program for creating containers which operate applications. These containers can be run on most systems which helps to simplify the development on different machines[1]. As described by Docker:

A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings [2].

1.5 Wireshark

Wireshark [2] is an open source network protocol analysis tool. It was used for capturing and evaluating network packets, both within the encrypted HIP layer and from the outside point of view. The tool is natively able to interpret and present the HIP base exchange packets. The debugging script created this year (test/debug.py) automatically does the packet capturing using the command line equivalent tool (called tshark). A shell script can then be run that gathers the captured files (that are in the .pcap format) outside the temporary folders of the CORE virtual network (see 1.3).

1.6 Virtual Machines

An Ubuntu 22.04 server with remote GUI capability was installed as a Linux KVM for debugging OpenSSL (using Virtual Network Computing forwarded over a Secure Shell tunnel on an open port). This server was used throughout the project, and enabled each student working on the OpenSSL milestone to connect and control the server over the internet, allowing for a flexible and collaborative environment for debugging the codebase.

The virtual machine that was used for the development of HHIT was installed using VirtualBox by Oracle[3]. The benefit from using virtual machines was that we could install and remove things on the system without it affecting our host machines (such as different SSL versions). Also, we were able to make copies (as backups) of the machine when we had a working solution of a configuration and/or code. These copies could also be sent between the team members.

[1] https://www.docker.com/
[2] https://www.wireshark.org/
[3] https://www.virtualbox.org/

We have uploaded two OVA files (of our VM's), which can be used by the next year's project group, if they want to have a working configuration that can run the code. We have two OVA files of VM's, for PyHIP and HHIT with the following specifications:

- Ubuntu 22.04 x64, with the virtual machine configured with 50GB and 4GB RAM for HHIT.
- Lubuntu 22.04.3 x64, with virtual machine configured with 20GB memory and 4GB RAM for PyHIP.

These were given to the examiner of the course and is meant to enable the next years students to quickly get started on a working system to speed up the development.

As stated above, for developing HHIT, we used a virtual machine of Ubuntu 22.04, where we needed downgrade the currently installed version of openssl, to version 1.1.1f which was downloaded from `https://www.openssl.org/source/old/1.1.1/`. We also needed to downgrade libssl to version 1.1_1.1.1f which was downloaded from `http://nz2.archive.ubuntu.com/ubuntu/pool/main/o/openssl/`. These files can be found in the Downloads directory in the Ubuntu VM for HHIT.

2 Milestones

This section lists the milestones for this year's group and explains the ones that the group of this year worked on. The HIPv2 project in the course had 5 milestones proposed for the year 2023 (listed in order of priority):

- Complete migration of OpenSSL from version 1.1.1 to version 3.0.10
- Implement Hierarchical Host Identity Tags as in RFC 9374
- Interoperability with PyHIP
- Latest crypto support
- Diet-ESP

2.1 OpenSSL

The OpenSSL library is used for all parts of hashing and encryption (both symmetric and asymmetric) in OpenHIP. A major part of this project was dedicated to continue to debug and update all parts of the codebase to use version 3.0.X of the library, the lastest long term support (LTS) version as of 2023, instead of version 1.1.1 which has now reached end of life (EOL).

2.2 HHIT

The milestone for HHIT was to develop it in accordance with RFC 9374. This included primarily updating the function hi_to_hhit in hip_util.c as specified in RFC 9374 as well as in section 4.3. It also requires changing some other functions that are related to the generation of the HHIT, also mentioned in section 4.3. In hi_to_hhit, the structure of the currently implemented code had the structure specified in section 3.2.

2.3 PyHIP

The goal for PyHIP is to make it faster and less resource intense.

2.4 Latest crypto support and Diet-ESP

These were not started due to lack of time.

3 Previous Work

This section summarizes the work done by previous groups in the course relevant to the milestones of this year.

3.1 OpenSSL

The OpenSSL migration has been a milestone since 2021 with two previous groups attempting to finish it. The group from 2021 tried an initial overhaul of the functions with the migration guide as a guideline. Their implementation led to a version that could compile but did not work correctly. The 2022 group continued the work with a changed strategy where the focus was on the initial handshake between hip nodes. Their work mostly focused on the input, output and esp files. Their stated result was a handshake that "almost worked".

3.2 HHIT

The previously implemented code for HHIT was not in accordance with RFC 9374. They had the following structure of the generated HHIT in the hi_to_hhit function:

- 28 bits IPv6 prefix
- 4 bits HHSI
- 32 bits HID (16 bits RAA and 16 bits HDA)
- 64 bits ORCHID hash

3.3 Debugging and testing

In the OpenHIP project there are two Python unit-test scripts. One is a basic and broad script that tests that two nodes can communicate within a CORE simulation using HIP. The other is a slightly more advanced one that, as we understand it, aims to capture the packet traffic in an active CORE HIP network.

3.4 Docker file

The group from the previous year created a Docker build file to build a docker container installed with CORE and the OpenHIPv2 branch[4] from the OpenHIP repository.

[4]https://bitbucket.org/openhip/openhip/src/docker-xoodyak-hhit-openssl3/

4 Contributions

The group of this year has worked on three of the proposed milestones: OpenSSL migration, HHIT and PyHIP interoperability. Below are summaries of the work contributed for each of these three milestones. In summary, *OpenSSL migration* and *PyHIP interoperability* are both still work in progress while the milestone *HHIT to DRIP integration* was completed.

4.1 OpenSSL migration

The focus this year was to verify last years implementation and continue their work.

We narrowed down the tests to only run a ping command and debugging that exchange to find the faults and needed changes. To do this we initially worked on easing debugging and testing of the code (see section 4.2) and then use this system to verify all the steps of the initial handshake.

The main problem we encountered was that a different group from last year made changes to the xml file where the cipher priority was overridden. Our main contribution was finding this and commenting out those functions.

The initial handshake now seems to be working:

- The nodes choose the correct cipher accourding to the hardcoded cipher priority.
- The nodes agree on the right cipher, meaning that both use the same one.
- The nodes create a common shared DH secret meaning initial DH key exchange works.
- A common DH secret suggests working DH_list for both nodes.

It should be noted that there may be a problem with the padding for the ENCRYPTED field in the I2 packet of the handshake as it looks different before being encrypted and sent (10101010...) compared to after being received and decrypted by the receiver node (00000000...). This is also handled differently with the data length variables, as it does not include the padding in the initiator but does in the responder, possibly causing problems for further functions.

4.2 Simplifying debugging and maintainability

4.2.1 Docker

We upgraded the prior year's Docker build file to build from the new official CORE docker image. This new CORE docker image itself builds from the Ubuntu 22.04, which comes with OpenSSL version 3 installed by default. We also changed the docker build file to build from the OpenHIPv2 experimental branch instead of the OpenHIPv2 stable branch. Older commits include the prior year's docker image, which might be helpful to extract and merge to the stable branch in order to have a Dockerfile for OpenSSL version 1.

4.2.2 Debugging Framework

This year we added a new way to debug and test OpenHIP. This addition is in the form of a *debug.py* script. This script does not use the Python unit-test framework, but instead does much of the configuration manually. The goal of *debug.py* was to be more granular and enable deeper debugging. The *debug.py* script offers the following:

- Automatic Wireshark interface packet capture for all OpenHIP nodes in the CORE network
- Exposes a SSH Node entry point, for manual node inspection
- Simplified OpenHIP CORE Node log extraction

We also fast-forwarded the current Python3 CORE node wrapper for all OpenHIP debugging scripts to COREv9, these changes was mainly due to architecture changes in COREv9 compared to COREv7.

4.2.3 Commenting

We made an attempt to update the comments, and some code format, in OpenHIP. The changes made were mainly to update the comments to use modern Doxygen format, indented some code, and relocated other comments. We also added some more comments on parts of the code we read through and did not understand clearly. This effort was mainly done to enable modern IDEs to better parse the project.

4.3 HHIT to DRIP integration

We were given the task to continue developing Hierarchical Host Identity Tags (HHIT). We updated it to follow RFC 9374 [5]. RFC 9374 specifies that the HHIT (just like the HIT) should be 128 bits long. It specifies that the input to the hash function (before generating the HHIT) should contain the following:

- 28 bits IPv6 prefix
- 28 bits HID (14 bits RAA and 14 bits HDA)
- 8 bits HHSI
- 64 bits HI (public key)

This input is hashed, and the output is used in the HHIT, which RFC 9374 specifies should contain the following:

- 28 bits IPv6 prefix
- 28 bits HID (14 bits RAA and 14 bits HDA)
- 8 bits HHSI
- 64 bits ORCHID hash output

This generation of the HHIT is done in the function hi_to_hhit in the file hip_util.c.

The following files and functions have been modified to update the current code according to RFC 9374.

4.3.1 hi_to_hhit in hip_util.c

The pipe symbol (|) below means concatenation.

- data is the variable that the entire HHIT is stored in (IPv6 prefix | HID | HHSI | public key).
- The IPv6 prefix was changed from being 32 bits to 28 in data.

- HID should start on bit 28, so we insert it in the 3.5th byte in data (bit 28).
- On bit 28 + 28 = 56 in data, meaning byte 7, the HHSI is inserted.
- On byte 8 and the following last 8 bytes in data, the public key is inserted.
- Everything stored in data is hashed and the output is stored along with the first part from data, in the end of the hit variable (IPv6 prefix | HID | HHSI | hash output). Everything before the hash output is stored in hit on the same location as it was in data, and the hash output is on the same location as the public key was stored in data.

4.3.2 main in hitgen.c

We added a new else if statement that handles when the argument -drip is passed to the program when calling it. This takes the argument with the value of the hda from the command line, creates a raa, and stores it in opts.info for it to be used by by hi_to_hhit hip_util.c

4.3.3 validate_hit in hip_util.c

We implemented a check that handles whether the tag has a HIT/HHIT prefix. We also fixed so that if there is an HHIT prefix then it will load data and info/OGA ID from a different memory location than if it were HIT.

4.3.4 The class TestEdDSA25519_Drip in test_switch.py

This is a class that specifies what one specific test should supply in terms of parameters in the command line, when calling the program. This class specifically is for testing generating a HHIT. This is done by supplying the argument "-drip" along with a number, e.g. "10" or "20".

It also has a couple of other arguments that specify the algorithm ID, etc.

4.4 Interop with PyHIP

Interop with PyHIP is something we started but were not able to finish due to lack of time. We started to change the use of the PyCryptoDome module to Cryptography, as this will likely increase the performance. However, we were not able to get it to work.

5 Future work

As stated earlier, two of the milestones (OpenSSL and PyHIP interoperability) are still works in progress, while two of them were untouched by the group of this year (*Latest crypto support* and *Diet-ESP*) primarily due to prioritization of the other milestones with the limited time available.

5.1 OpenSSL

This year we primarily focused on the initial handshake. This covers the input and output files (protocol/hip_input.c and protocol/hip_output.c) where we focused on I1, R1, I2 and R2.

For next year the first recommendation is to get the environment working and the ping test running. There is little point in trying to figure out all the functions and previous implementation. It is more beneficial to work through the packets being sent to find exactly where the problem lies and there look at previous years commits and the migration guide.

The OpenSSL group had a major issue, that we decided not to include in the git repository, but instead mention here. This was that the cipher priority should not be decided by the xml file (from 2022 group) but should be hardcoded in main for now (that is a problem for later). After the debugging/testing environment is working you have to check util/hip_xml and comment out the following lines to override the cipher preference for R1 being read from xml instead of the hardcoded values in main:

- `memset(HCNF.hip_ciphers, 0, sizeof(__u16) * HIP_CIPHER_MAX);`
- `HCNF.hip_ciphers[t] = (__u16)tmp;`
- `memset(HCNF.esp_transforms, 0, sizeof(__u16) * ESP_MAX);`
- `HCNF.esp_transforms[t] = (__u16)tmp;`

We have also identified three locations where more implementation might be needed:

5.1.1 Encrypted and decrypted data in the esp file

When looking at the encrypted and decrypted data after the initial handshake in the esp file we note that the decrypted data can not match the encrypted. This means that there is a problem in either the sender encrypting wrong or the receiver decrypting wrong, or both. Here we can also see a lot of changes done last year suggesting faulty implementation.

5.1.2 Padding for the initial handshake

As noted in contribution section 4.1, there may be a problem with the padding for the initial handshake where the length of the buffer is different for the initiator and responder.

5.1.3 Increasing packet size

When looking at packets being sent after the initial handshake with Wireshark the packets are increasing its size with zeroes. Wireshark has a hard time understanding and states "hop-by-hop" packets with "destination unreachable". This may be a problem concerning memory allocation for the packets. When looking at last year they changed the memory allocation for the packets in the initial handshake. Similar changes could be required for following packages.

5.2 HHIT

We recommend fixing the problem with large numbers in the argument, as explained in section 6.4.

Something that is good to know is that the code hasn't been tested with any other parameters and any other values than the ones currently in the class TestEdDSA25519_Drip in test_switch.py. This means that we only tested generation of HHIT with hi->algorithm_id being HI_ALG_EDDSA, and type being HIT_SUITE_4BIT_EDDSA_CSHAKE128 (both in hi_to_hhit in hip_util.c)

Just like in the previously implemented code, we don't have a way of automatically assigning an ”real” RAA based on a HDA, so this is done by calling the function register_to_hda.

5.3 PyHIP

The goal for PyHIP is to make it faster and less resource intense. The supervisor has a PDF file with some tips on how this can be achieved.

5.4 XML

As stated in contribution section 4.1 last years xml implementation overrides cipher priority. Here further work might be needed for cipher priority selection.

6 Discussion and advice

6.1 Technical Debt

We want to introduce a, what we believe is a relevant topic, called *Technical Debt*.

"Technical debt describes the consequences of software development actions that intentionally or unintentionally prioritize client value and/or project constraints such as delivery deadlines, over more technical implementation and design considerations. These include matters such as achieving and sustaining test coverage or code extensibility. Conceptually, technical debt is an analog of financial debt, with associated concepts such as levels of debt, debt accrual over time and its likely consequences, and the pressure to pay back the debt at some point in time"[3]

Technical Debt is relevant to this project since multiple years with different student have worked on this project. No single student have either had sufficient time or focus for project sustainability. For some students this is likely an entirely new kind of course, leading to technical debt accumulating in the OpenHIP project. This situation can be summed up to:

- Old project – Requires continuous documentation updating.
- Diverse workers – Yet no code and project standards have been chosen.
- Lack of time – Workers focus more on implementing functionality than documenting what they add.
- New experience – Workers on OpenHIP are not used to this kind of project, leading to different types of coding.

Having technical debt in mind, we advice further years' project groups to put more focus on maintainability and documentation and not to try implementing as much as possible. This project will take many years to finish at the current pace, and will only successfully move forward with this mindset.

6.2 Hard to get started for OpenSSL group

Starting out from where the two earlier groups left us posed a major challenge. We did not know what approach to use in the beginning to start tackling such a major task. The groups of 2022 and 2021 both had tried to migrate the OpenSSL version from scratch, since the 2022 group decided not to continue with the 2021 code. This was a tempting strategy for us too as it was a huge undertaking trying to figure out if the problem(s) with the partly migrated code was/were due to the earlier implementation or a further need for migration. We are happy that we chose to continue with what we were given though, despite the group from last year not really having a concrete description of how to proceed further, causing a lot of time spent looking through and verifying code that was already working. We would recommend that next year's group does the same, but using the improved debugging and testing tools we created.

6.3 Help for Starting up in the course

Here is our collection of useful documents for starting out with the course, gitlab.liu.se/gusbo010/tdde21. Feel free to fork it.

6.4 Problems with arguments when calling HHIT

Not all numbers work when calling HHIT, i.e. the argument supplied along with "-drip". Too large numbers don't work. We believe this is because Python converts the number (argument) to hexadecimal, and then it's converted to ASCII. The problem is that large hexadecimal numbers don't have a conversion to ASCII.

6.5 Potential problem with installing OpenHIP

The following changes need to be done before the first time you compile OpenHIP.

According to the openhip/README.md, you should run the following to install OpenHIP.

```
./bootstrap.sh
./configure
```

After these two commands have been run, open the following files: openhip/src/util/Makefile, openhip/src/Makefile, openhip/docs/Makefile and openhip/Makefile. In all of these files, remove "-Werror" (can be found by searching in the file).

Then, you can continue with the installation according to openhip/README.md, i.e. run the following commands:

```
make
sudo make install
```

If you use the VM OVA's supplied (see 1.6), this has already been done.

6.6 How to easily find parts of the code that were changed for HHIT by our project group

While this can be seen in git commits, we also added comments containing the string "HHIT 2023" which you can search for in the files, to easily find the most important changes that we made for HHIT.

6.7 How to run and test the HHIT generation

A virtual machine (OVA file) has been saved from our project group, as mentioned in 1.6. To test HHIT in this VM, the following lines can be run:

```
cd ~/Desktop/openhip && make && sudo make install && cd /opt/core/venv/bin/

sudo ./core-cleanup && sudo ./python3.10 /home/hhit/Desktop/openhip/test/test_switch_9.py
```

6.8 A potential fix for a problem when running test for HHIT

Sometimes, when running the commands specified in 6.7, the tests fail. This can sometimes be fixed by removing the files /tmp/pycore.1 and /tmp/pycore.2.

6.9 Debugging and testing

Debugging and testing are both done easiest by running the script "test/scripts/compilerundump.sh" that in turn compiles OpenHIP, runs debug.py in the CORE Python environment and dumps its output to a text file located in ”test/logdumps/HHMM.txt” (where HH is the current hour and MM is the current minute). An example of such a log file can be found in appendix C. While the test is running in the background, in another shell one can then run ”test/scripts/gather_ws_files_2023.sh” to put Wireshark files that were captured from the beginning of the test in a desired directory (that can be specified within the script). One may also execute the generated ssh scripts (requires system root privileges unfortunately, so it should only be used in a VM!) in the test folder in order to connect to one of the nodes emulated in CORE by debug.py. See also the README.md files in the OpenHIP repository for further explanation on how to use the debugging/testing environment.

We recommend using a VM for continuing development of HHIT, if it is necessary. This is for the reasons mentioned in section 1.6, where you will also find instructions on where you can find OVA files of our VM's.

References

[1] CORE. *CORE Documentation.* 2023. URL: https://coreemu.github.io/core/index.html (visited on 12/04/2023).

[2] Docker. *About docker.* 2023. URL: https://www.docker.com/resources/what-container/ (visited on 12/15/2023).

[3] Johannes Holvitie et al. "Technical debt and agile software development practices and processes: An industry practitioner survey". In: *Information and Software Technology* 96 (2018), pp. 141–160. ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2017.11.015. URL: https://www.sciencedirect.com/science/article/pii/S0950584917305098.

[4] Robert Moskowitz, Pekka Nikander, and T Henderson. *Rfc 5201: Host identity protocol.* 2008.

[5] Robert Moskowitz et al. *DRIP Entity Tag (DET) for Unmanned Aircraft System Remote ID (UAS RID).* RFC 9374. Mar. 2023. DOI: 10.17487/RFC9374. URL: https://www.rfc-editor.org/info/rfc9374.

[6] *RFC definition.* https://www.rfc-editor.org/. Accessed: 2023-11-30.

[7] *TDDE21 is a course given at Linköpings University in Sweden for computer science division.* https://www.ida.liu.se/~TDDE21/info/courseinfo.en.shtml. Accessed: 2023-11-30.

A Wordlist

Word	Description
IETF	Internet Engineering Task Force
IRTF	Internet Research Task Force
IAB	Internet Architecture Board
RFC	*Request For Comment* contains technical and organizational documents about the Internet, including the specifications and policy documents produced by five streams: IETF, IRTF, IAB, Independent Submissions, and Editorial [6].
HIT	Host Identity Tag
DRIP	Drone Remote ID Protocol
HHIT	Hierarchical HIT
ORCHID	Overlay Routable Cryptographic Hash Identifiers

B OpenHIP handshake and IPSec overview

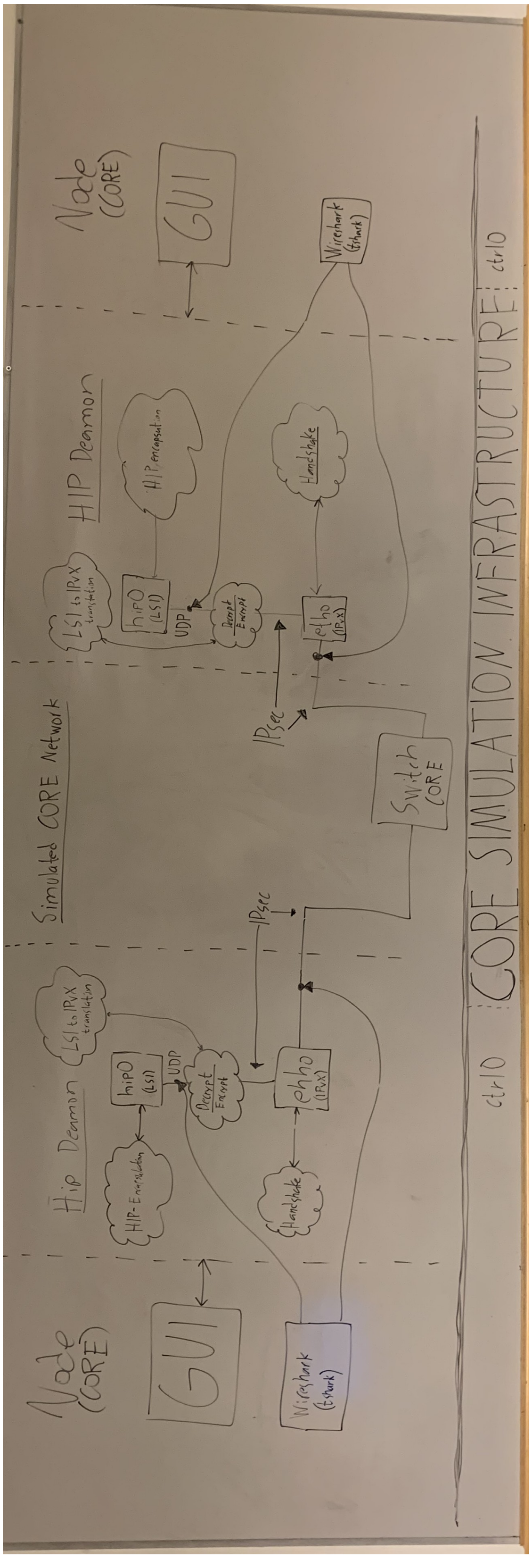

Figure 2: This is how we believe that the HIP handshake and IPsec tunnel work. May not be 100% accurate.

C Last logfile from OpenSSL group

```
killing vnoded processes ... done
killing emane processes ... none found
cleaning up devices
removed b.1.1
removed veth1.0.1
failed to remove beth1.0.1
removed b.3.1
removed veth1.1.1
failed to remove beth1.1.1
removed b.5.1
removed ctrl0.1
removing session directories ... done
public: /tmp/pycore.1/known_host_identities.xml
Node List:
HIPNode2: IPv4(10.0.0.2       ), LSI(1.146.95.148   ), PWD(/tmp/pycore.1/HIPNode2.conf), SSH_cmd(./s
HIPNode4: IPv4(10.0.0.4       ), LSI(1.17.52.127    ), PWD(/tmp/pycore.1/HIPNode4.conf), SSH_cmd(./s
INFO: SSH_cmd are scripts created to simplify ssh login to each node, can only be used when this scri
INFO: IPv6 address is only assigned address to underlying interface and not part of CORE!

   1.146.95.148 attempting: ping -c 20 -W 1 -4 1.17.52.127    Fail: []
HIPNode2 hip dump ###############################
OpenHIP v0.9svn1 HIP daemon
init_tap()
Using TAP device hip0.
Initialized TAP device.
Thu Nov 30 16:49:06 2023 (4) hipd v0.9svn1 (40) started.
Setting options: daemon = no  debug level = 1  permissive = no
     no_retransmit = no  opportunistic = no any = no rvs = no mr = no
Hit suite list
1
0
0
0
0
0
0
Using configuration file: ./hip.conf
Using my host IDs file: ./my_host_identities.xml
My host identities:
 HI: HIPNode2-1024 HIT: 2001:21:d21f:38e8:cdba:a60f:8292:5f94
LSI: 1.146.95.148
Using known host IDs file: ./known_host_identities.xml
Known peer host identities:
 HIT: HIPNode2-1024  2001:21:d21f:38e8:cdba:a60f:8292:5f94
LSI: 1.146.95.148 [10.0.0.2 ]
 HIT: HIPNode4-1024  2001:21:d1b1:acc8:c339:e992:ed11:347f
LSI: 1.17.52.127 [10.0.0.4 ]
Local addresses: (1)127.0.0.1 (100)10.0.0.2 (111)172.16.0.2 (1)::1 (100)fe80::216:3eff:fe2d:aece (11
10.0.0.2 selected as the preferred address (first in list).
Initializing R1 cache entries for identity HIPNode2-1024 (8 slots).
tlv_set_hostid_len() hi_len==144
tlv_set_hostid_len() EVP_PKEY_get_size(hi->evp_pkey)==128

-------------------------------- 2023 --------------------------------
```

```
hip_output.c : generate_R1() : HCNF.hip_ciphers==4
hip_output.c : generate_R1() : HCNF.hip_ciphers==2
hip_output.c : generate_R1() : HCNF.hip_ciphers==6
hip_output.c : generate_R1() : HCNF.hip_ciphers==7
hip_output.c : generate_R1() : HCNF.hip_ciphers==9
------------------------------ END 2023 ------------------------------

...

-------------------------------- 2023 --------------------------------
hip_output.c build_tlv_transform : type=PARAM_HIP_CIPHER
hip_output.c build_tlv_transform : multiple==0
hip_output.c build_tlv_transform : transform_list==4
hip_output.c build_tlv_transform : transform_list==2
hip_output.c build_tlv_transform : transform_list==6
hip_output.c build_tlv_transform : transform_list==7
hip_output.c build_tlv_transform : transform_list==9
hip_output.c build_tlv_transform : transform_id==4
hip_output.c build_tlv_transform : transform_id==2
hip_output.c build_tlv_transform : transform_id==6
hip_output.c build_tlv_transform : transform_id==7
hip_output.c build_tlv_transform : transform_id==9
------------------------------ END 2023 ------------------------------


-------------------------------- 2023 --------------------------------
hip_output.c build_tlv_transform : type!=PARAM_HIP_CIPHER
hip_output.c build_tlv_transform : multiple==0
hip_output.c build_tlv_transform : transform_list==1
hip_output.c build_tlv_transform : transform_list==2
hip_output.c build_tlv_transform : transform_list==3
hip_output.c build_tlv_transform : transform_list==4
hip_output.c build_tlv_transform : transform_list==5
hip_output.c build_tlv_transform : transform_list==6
hip_output.c build_tlv_transform : transform_list==7
hip_output.c build_tlv_transform : transform_list==8
hip_output.c build_tlv_transform : transform_id==1
hip_output.c build_tlv_transform : transform_id==2
hip_output.c build_tlv_transform : transform_id==3
hip_output.c build_tlv_transform : transform_id==4
hip_output.c build_tlv_transform : transform_id==5
hip_output.c build_tlv_transform : transform_id==6
hip_output.c build_tlv_transform : transform_id==7
hip_output.c build_tlv_transform : transform_id==8
------------------------------ END 2023 ------------------------------

tlv_set_hostid_len() hi_len==144
tlv_set_hostid_len() EVP_PKEY_get_size(hi->evp_pkey)==128
tunreader() thread started (5)...


-------------------------------- 2023 --------------------------------
hip_output.c : generate_R1() : HCNF.hip_ciphers==4
hip_output.c : generate_R1() : HCNF.hip_ciphers==2
hip_output.c : generate_R1() : HCNF.hip_ciphers==6
hip_output.c : generate_R1() : HCNF.hip_ciphers==7
hip_output.c : generate_R1() : HCNF.hip_ciphers==9
```

```
------------------------------ END 2023 ------------------------------


Adding address 1.146.95.148 to interface 2.
Adding address 2001:21:d21f:38e8:cdba:a60f:8292:5f94 to interface 2.
Thu Nov 30 16:49:06 2023 (1) Listening for HIP control packets...
Thu Nov 30 16:49:08 2023  HIP threads initialization completed.
main.c UDP HANDLE
 01000000 00000080 02000000 0111347f
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000
Thu Nov 30 16:49:11 2023 (1) Received ACQUIRE for LSI 1.17.52.127 creating new association.
Adding address 10.0.0.2 to association.
Adding address 172.16.0.2 to association.
Thu Nov 30 16:49:11 2023 (4) Base exchange initiated from
2001:21:d21f:38e8:cdba:a60f:8292:5f94 / 10.0.0.2 / 1.146.95.148 to
2001:21:d1b1:acc8:c339:e992:ed11:347f / 10.0.0.4 / 1.17.52.127
 20010021 d1b1acc8 c339e992 ed11347fSending HIT corresponding to 10.0.0.4.
Thu Nov 30 16:49:11 2023 (1) Sending HIP_I1 packet (48 bytes)...

2023 - ABOUT TO HIP_SEND PACKET WITH CONTENTS :
 00000000 00000000 00000000 3b050111
00000000 20010021 d21f38e8 cdbaa60f
82925f94 20010021 d1b1acc8 c339e992
Thu Nov 30 16:49:11 2023 (1) Sending HIP packet on UDP socket
ESP: get encrypted in input thread
 45000228 b9cc4000 40116af3 0a000004
0a000002 29042904 0214a848 00000000
3b400211 00000000 20010021 d1b1acc8
c339e992 ed11347f 20010021 d21f38e8
cdbaa60f 82925f94 0081000c 00000000
00000000 0000000b 01010024 0a270000
51abc25e 6742c688 edeafdbf e10eb33f
2d943641 f5148691 e65626f4 5d0ed91f
02010043 0a004030 3e301006 072a8648
ce3d0201 06052b81 04000803 2a0004f3
09ac9353 e6fb7163 70db58f8 fbb6b5b4
9333988e 6e054b8e 1525ab10 4c11a09c
0e033a34 f0ab3500 01ff0004 0a030d0e
0243000a 00040002 00060007 00090000
02c10099 0088100d 0202ff05 03010001
dd6cd34b fd37db75 6bbf1303 c4803059
20e8f4ff 4824f8f5 e972df28 ba31784d
4afb99e7 e10052c0 532cb467 4fdffe07
b8f3702b 97f9298e e9ff5a7b cf173bd1
ff3e9c1f 22661103 17acf039 428586d6
5fc517d0 9752e818 55e1b6c1 8e204003
0848b6c1 7eb70850 e2f1db89 554d150a
3236516f 61470187 8f410038 a576ce55
4849504e 6f646534 2d313032 34000000
02cb0001 10000000 0fff0012 00000001
```

```
00020003 00040005 00060007 00080000
f0c10081 05bc19fd f49e0526 0fb08bc2
975811af beb1fe2d 062f1fc7 6994060e
4c5917d0 a28230de 6fde2f93 a2c34e30
4e74f209 9eedf185 5a9dbc8f b245b403
e288457d 51e92224 224e2f6c a08e5d66
fa946b62 4c618be6 7c29acc8 35ed8795
62acbbf0 ede085d9 ec1d3f3b ecf93203
a18f95eb ac60e274 4e0e5e3a 293a62f6
de2744e4 1e000000
ESP_RECEIVE_UDP_HIP_PACKET
 45000228 b9cc4000 40116af3 0a000004
0a000002 29042904 0214a848 00000000
3b400211 00000000 20010021 d1b1acc8
c339e992 ed11347f 20010021 d21f38e8
cdbaa60f 82925f94 0081000c 00000000
00000000 0000000b 01010024 0a270000
51abc25e 6742c688 edeafdbf e10eb33f
2d943641 f5148691 e65626f4 5d0ed91f
02010043 0a004030 3e301006 072a8648
ce3d0201 06052b81 04000803 2a0004f3
09ac9353 e6fb7163 70db58f8 fbb6b5b4
9333988e 6e054b8e 1525ab10 4c11a09c
0e033a34 f0ab3500 01ff0004 0a030d0e
0243000a 00040002 00060007 00090000
02c10099 0088100d 0202ff05 03010001
dd6cd34b fd37db75 6bbf1303 c4803059
20e8f4ff 4824f8f5 e972df28 ba31784d
4afb99e7 e10052c0 532cb467 4fdffe07
b8f3702b 97f9298e e9ff5a7b cf173bd1
ff3e9c1f 22661103 17acf039 428586d6
5fc517d0 9752e818 55e1b6c1 8e204003
0848b6c1 7eb70850 e2f1db89 554d150a
3236516f 61470187 8f410038 a576ce55
4849504e 6f646534 2d313032 34000000
02cb0001 10000000 0fff0012 00000001
00020003 00040005 00060007 00080000
f0c10081 05bc19fd f49e0526 0fb08bc2
975811af beb1fe2d 062f1fc7 6994060e
4c5917d0 a28230de 6fde2f93 a2c34e30
4e74f209 9eedf185 5a9dbc8f b245b403
e288457d 51e92224 224e2f6c a08e5d66
fa946b62 4c618be6 7c29acc8 35ed8795
62acbbf0 ede085d9 ec1d3f3b ecf93203
a18f95eb ac60e274 4e0e5e3a 293a62f6
de2744e4 1e000000
main.c UDP HANDLE
 03000000 00000228 45000228 b9cc4000
40116af3 0a000004 0a000002 29042904
0214a848 00000000 3b400211 00000000
20010021 d1b1acc8 c339e992 ed11347f
20010021 d21f38e8 cdbaa60f 82925f94
0081000c 00000000 00000000 0000000b
01010024 0a270000 51abc25e 6742c688
edeafdbf e10eb33f 2d943641 f5148691
e65626f4 5d0ed91f 02010043 0a004030
3e301006 072a8648 ce3d0201 06052b81
```

```
04000803 2a0004f3 09ac9353 e6fb7163
70db58f8 fbb6b5b4 9333988e 6e054b8e
1525ab10 4c11a09c 0e033a34 f0ab3500
01ff0004 0a030d0e 0243000a 00040002
00060007 00090000 02c10099 0088100d
0202ff05 03010001 dd6cd34b fd37db75
6bbf1303 c4803059 20e8f4ff 4824f8f5
e972df28 ba31784d 4afb99e7 e10052c0
532cb467 4fdffe07 b8f3702b 97f9298e
e9ff5a7b cf173bd1 ff3e9c1f 22661103
17acf039 428586d6 5fc517d0 9752e818
55e1b6c1 8e204003 0848b6c1 7eb70850
e2f1db89 554d150a 3236516f 61470187
8f410038 a576ce55 4849504e 6f646534
2d313032 34000000 02cb0001 10000000
0fff0012 00000001 00020003 00040005
00060007 00080000 f0c10081 05bc19fd
f49e0526 0fb08bc2 975811af beb1fe2d
062f1fc7 6994060e 4c5917d0 a28230de
6fde2f93 a2c34e30 4e74f209 9eedf185
5a9dbc8f b245b403 e288457d 51e92224
224e2f6c a08e5d66 fa946b62 4c618be6
7c29acc8 35ed8795 62acbbf0 ede085d9
ec1d3f3b ecf93203 a18f95eb ac60e274
4e0e5e3a 293a62f6 de2744e4 1e000000
Thu Nov 30 16:49:11 2023 (1) Received HIP_R1 packet from 10.0.0.4 on udp socket length 552

-------------------------------- 2023 --------------------------------
hip_input.c parse_R1() : type==129
------------------------------- END 2023 ------------------------------

 R1 TLV type = 129 length = 12

-------------------------------- 2023 --------------------------------
hip_input.c parse_R1() : type==257
------------------------------- END 2023 ------------------------------

 R1 TLV type = 257 length = 36

-------------------------------- 2023 --------------------------------
hip_input.c parse_R1() : type==513
------------------------------- END 2023 ------------------------------

 R1 TLV type = 513 length = 67

-------------------------------- 2023 --------------------------------
hip_input.c parse_R1() : type==511
------------------------------- END 2023 ------------------------------

 R1 TLV type = 511 length = 4

-------------------------------- 2023 --------------------------------
hip_input.c parse_R1() : type==579
------------------------------- END 2023 ------------------------------

 R1 TLV type = 579 length = 10
```

```
-------------------------------- 2023 --------------------------------
hip_input.c parse_R1() : type==705
------------------------------ END 2023 ------------------------------

 R1 TLV type = 705 length = 153

-------------------------------- 2023 --------------------------------
hip_input.c parse_R1() : type==715
------------------------------ END 2023 ------------------------------

 R1 TLV type = 715 length = 1

-------------------------------- 2023 --------------------------------
hip_input.c parse_R1() : type==129
------------------------------ END 2023 ------------------------------

 R1 TLV type = 129 length = 12

-------------------------------- 2023 --------------------------------
hip_input.c parse_R1() : type==257
------------------------------ END 2023 ------------------------------

 R1 TLV type = 257 length = 36

-------------------------------- 2023 --------------------------------
hip_input.c parse_R1() : type==513
------------------------------ END 2023 ------------------------------

 R1 TLV type = 513 length = 67

-------------------------------- 2023 --------------------------------
hip_input.c parse_R1() : type==511
------------------------------ END 2023 ------------------------------

 R1 TLV type = 511 length = 4

-------------------------------- 2023 --------------------------------
hip_input.c parse_R1() : type==579
------------------------------ END 2023 ------------------------------

 R1 TLV type = 579 length = 10

-------------------------------- 2023 --------------------------------
hip_input.c parse_R1() : type==705
------------------------------ END 2023 ------------------------------

 R1 TLV type = 705 length = 153
Found RSA HI with public modulus: 0x dd6cd34b fd37db75 6bbf1303 c4803059
20e8f4ff 4824f8f5 e972df28 ba31784d
4afb99e7 e10052c0 532cb467 4fdffe07
b8f3702b 97f9298e e9ff5a7b cf173bd1
ff3e9c1f 22661103 17acf039 428586d6
5fc517d0 9752e818 55e1b6c1 8e204003
0848b6c1 7eb70850 e2f1db89 554d150a
3236516f 61470187 8f410038 a576ce55
HI has name: HIPNode4-1024 length: 13
HI in R1 validates the sender's HIT.
```

```
-------------------------------- 2023 --------------------------------
hip_input.c parse_R1() : type==715
------------------------------ END 2023 ------------------------------

 R1 TLV type = 715 length = 1

-------------------------------- 2023 --------------------------------
hip_input.c parse_R1() : type==4095
------------------------------ END 2023 ------------------------------

 R1 TLV type = 4095 length = 18

-------------------------------- 2023 --------------------------------
hip_input.c parse_R1() : type==61633
------------------------------ END 2023 ------------------------------

 R1 TLV type = 61633 length = 129
Validating signature of type 1
RSA HIP signature is good.

-------------------------------- 2023 --------------------------------
hip_input.c parse_R1() : type==129
------------------------------ END 2023 ------------------------------

 R1 TLV type = 129 length = 12

-------------------------------- 2023 --------------------------------
hip_input.c parse_R1() : type==257
------------------------------ END 2023 ------------------------------

 R1 TLV type = 257 length = 36
Got the R1 cookie: (k=10 lifetime=39 (128 seconds) opaque=0
I=0x 51abc25e 6742c688 edeafdbf e10eb33f
2d943641 f5148691 e65626f4 5d0ed91f)

-------------------------------- 2023 --------------------------------
hip_input.c parse_R1() : type==513
------------------------------ END 2023 ------------------------------

 R1 TLV type = 513 length = 67
*** Warning: public key len = 64, expected -1 for this group id (10)
Got DH public value of len 64: 0x 303e3010 06072a86 48ce3d02 0106052b
81040008 032a0004 f309ac93 53e6fb71
6370db58 f8fbb6b5 b4933398 8e6e054b
8e1525ab 104c11a0 9c0e033a 34f0ab35
EVP_PKEY type: 408-----BEGIN PUBLIC KEY-----
MD4wEAYHKoZIzj0CAQYFK4EEAAgDKgAEVmBBciD3ieYHxOtj8zPCijYMh5ey+c0I
sfPtokTLjPaxUXLe4zB+Ag==
-----END PUBLIC KEY-----
************
DH secret key set to:
0x 5c7b40bd f488f610 5fcb1c3d 63a12a83 34c4d4a8
***********

-------------------------------- 2023 --------------------------------
hip_input.c parse_R1() : type==511
```

```
------------------------------ END 2023 ------------------------------

 R1 TLV type = 511 length = 4

------------------------------- 2023 -------------------------------
hip_input.c parse_R1() : type==579
------------------------------ END 2023 ------------------------------

 R1 TLV type = 579 length = 10

------------------------------- 2023 -------------------------------
hip_input.c parse_R1() : tlv->cipher_id==4
hip_input.c parse_R1() : tlv->cipher_id==6
hip_input.c parse_R1() : tlv->cipher_id==9
hip_input.c parse_R1() : tlv->cipher_id==2c1
hip_input.c parse_R1() : tlv->cipher_id==88
hip_input.c parse_R1() : tlv->cipher_id==202
length==10
------------------------------ END 2023 ------------------------------


------------------------------- 2023 -------------------------------
hip_input.c parse_R1() : type==705
------------------------------ END 2023 ------------------------------

 R1 TLV type = 705 length = 153

------------------------------- 2023 -------------------------------
hip_input.c parse_R1() : type==715
------------------------------ END 2023 ------------------------------

 R1 TLV type = 715 length = 1

------------------------------- 2023 -------------------------------
hip_input.c parse_R1() : type==4095
------------------------------ END 2023 ------------------------------

 R1 TLV type = 4095 length = 18

------------------------------- 2023 -------------------------------
hip_input.c parse_R1() : type==61633
------------------------------ END 2023 ------------------------------

 R1 TLV type = 61633 length = 129
Using cookie from R1: (k=10 lifetime=39 (128 seconds) opaque=0
I=0x 51abc25e 6742c688 edeafdbf e10eb33f
2d943641 f5148691 e65626f4 5d0ed91f)
Calculating Ltrunc(SHA256(I|Rand),K)...found match in 450 tries (~0 seconds).
MD= e2fca4ce d9a3bafb 015be04a 55b52ec6
2744260a f2365d7e 00fc4681 216b6a21
IJ= 51abc25e 6742c688 edeafdbf e10eb33f
2d943641 f5148691 e65626f4 5d0ed91f
20010021 d21f38e8 cdbaa60f 82925f94
20010021 d1b1acc8 c339e992 ed11347f
16128f35 a1e85942 d1dbd130 cd8878d6
026a8c4a 58e2bcf2 15809878 6a7b149a
Sending the I2 cookie: (k=10 lifetime=39 (128 seconds) opaque=0
```

```
I=0x 51abc25e 6742c688 edeafdbf e10eb33f
2d943641 f5148691 e65626f4 5d0ed91f)
with solution j:  16128f35 a1e85942 d1dbd130 cd8878d6
026a8c4a 58e2bcf2 15809878 6a7b149a
Using HIP transform of 4.
Drawing new HIP encryption/integrity keys:
Key 0 (4,32) keymat[  0] 0x 34148939 a0ebd976 194996a8 83267126
01a00861 b1bcec79 7d0719da 5fa9866a
Key 1 (1,32) keymat[ 32] 0x de6d2119 862cead6 e31da99a 4b7d2a33
6829ae1b cdccf24b 787742f6 9ba0a0c8
Key 2 (4,32) keymat[ 64] 0x f13a4fcf 3ce4cbfe 72264e72 0b67d21f
12ddd3f4 ab100544 cfd63572 7cb76855
Key 3 (1,32) keymat[ 96] 0x 3f2032bb e60748c5 d7079ff2 16992529
310f728a 1bb3139a 9f3d4549 cc487519
Using DH public value of len 64: 0x

-------------------------------- 2023 --------------------------------
hip_output.c build_tlv_transform : type=PARAM_HIP_CIPHER
hip_output.c build_tlv_transform : single==4
hip_output.c build_tlv_transform : transform_id==4
------------------------------ END 2023 ------------------------------

tlv_set_hostid_len() hi_len==144
tlv_set_hostid_len() EVP_PKEY_get_size(hi->evp_pkey)==128
---Test - (id=4)
---Test - suite(id=1)
---Test - 176 data_len.
---Test - 16 iv_len.
2023: NOT YET ENCRYPTED DATA IS:
 02c10099 0088100d 0202ff05 03010001
b6730c65 59b5350f e56f8784 fee5d619
92f25206 a37a7cfd 5e640451 ab7a041b
60abb6d9 e3dca41b ed9ca3c6 df39e091
94bc3401 a77038b7 ab34fbd3 bdba15c2
403ca41a 4230bc90 bf504f4d 7dbbec25
e894d52a 753a5c9e 6f1e7d8f 4f596cc4
a465286f cbfe6296 a3abbdb0 d0167b3f
822d431b f65332c9 45dba1d4 9cc8984f
4849504e 6f646532 2d313032 34000000
10101010 10101010 10101010 10101010
---TEST - 0 enc_data AES encryption key: 0x 34148939 a0ebd976 194996a8 83267126
01a00861 b1bcec79 7d0719da 5fa9866a
Encrypting 176 bytes using AES.
2023: ENCRYPTED DATA IS:
 e8bb58b7 ee0217ac e0f045a9 54725811
5776d166 142231e8 3b413aea ac56913a
4743a577 faee2e08 8b651d6e 2ab358ab
cc96f58d 69c9231e 0731bc33 85e465db
8210f2ba cbda85a1 04ff752d 83ccf784
1f8a563e 334cd1b5 3de8f434 511cad3b
f7ee8b9c a86858e1 b1452237 defaf6b5
512c088f 53457f5b 8abc5988 4f381460
58092c33 9c7b140c 962fb987 af23e63d
8dde8a39 5b2a5fde 721da8d8 c1893b40
390c5635 55b14719 ee1b1448 2db139a5
-------------------------------- 2023 --------------------------------
hip_output.c build_tlv_transform : type!=PARAM_HIP_CIPHER
```

```
hip_output.c build_tlv_transform : single==1
hip_output.c build_tlv_transform : transform_id==1
------------------------------ END 2023 ------------------------------

HMAC computed over 432 bytes hdr length=53
 HMAC length=68
*** HMAC_md_len=32, hmacsize=64
SHA1:  b8e8f829 d46507cf 707d2ec0 4e1fe064
7492f13a 06bcfa12 6be8168c e1a93d31
Signature:  16dbe84a 1f93bcc4 408cdcd6 0aa8e209
4f7523c6 f21d2ecb 286a3ee0 bb6bc806
19b41bf4 a8879825 08ca1f01 685fbac2
04546163 47edbde0 b6dc73d1 0ded756b
632304cd 37bd0a66 0d7f4675 98cf0d2f
8dce040b 8daad529 2ad263b2 a5e49495
9c4a6f85 e1944c7a 0dbe9192 34fac6fa
d610eea9 5d2a89a5 daf9be7b d509a54e
Thu Nov 30 16:49:11 2023 (1) Sending HIP_I2 packet (640 bytes)...
---Test - buff 3b 4f 3 11 0 0 0 0 20 1 0 21 d2 1f 38 e8 cd ba a6 f 82 92 5f 94 20 1 0 21 d1 b1 ac c8
2023 - ABOUT TO HIP_SEND PACKET WITH CONTENTS :
 00000000 00000000 00000000 3b4f0311
00000000 20010021 d21f38e8 cdbaa60f
82925f94 20010021 d1b1acc8 c339e992
ed11347f 0041000c 00000080 00000000
908252e4 0081000c 00000000 00000000
0000000b 01410044 0a270000 51abc25e
6742c688 edeafdbf e10eb33f 2d943641
f5148691 e65626f4 5d0ed91f 16128f35
a1e85942 d1dbd130 cd8878d6 026a8c4a
58e2bcf2 15809878 6a7b149a 02010043
0a004030 3e301006 072a8648 ce3d0201
06052b81 04000803 2a000456 60417220
f789e607 c4eb63f3 33c28a36 0c8797b2
f9cd08b1 f3eda244 cb8cf6b1 5172dee3
307e0200 02430002 00040000 028100c4
00000000 ec415691 9eb6aaa6 4144baad
2f47dce3 e8bb58b7 ee0217ac e0f045a9
54725811 5776d166 142231e8 3b413aea
ac56913a 4743a577 faee2e08 8b651d6e
2ab358ab cc96f58d 69c9231e 0731bc33
85e465db 8210f2ba cbda85a1 04ff752d
83ccf784 1f8a563e 334cd1b5 3de8f434
511cad3b f7ee8b9c a86858e1 b1452237
defaf6b5 512c088f 53457f5b 8abc5988
4f381460 58092c33 9c7b140c 962fb987
af23e63d 8dde8a39 5b2a5fde 721da8d8
c1893b40 390c5635 55b14719 ee1b1448
2db139a5 0fff0004 00000001 f0410040
00000000 20000000 0053114b 1b7f0000
b054114b 1b7f0000 00e10746 166e7d42
85865509 d42bf835 0289d9a6 797b33b9
372b2f87 7d9e05e0 69144c87 334002dc
00000000 f1010081 0516dbe8 4a1f93bc
c4408cdc d60aa8e2 094f7523 c6f21d2e
cb286a3e e0bb6bc8 0619b41b f4a88798
2508ca1f 01685fba c2045461 6347edbd
e0b6dc73 d10ded75 6b632304 cd37bd0a
```

```
660d7f46 7598cf0d 2f8dce04 0b8daad5
292ad263 b2a5e494 959c4a6f 85e1944c
7a0dbe91 9234fac6 fad610ee a95d2a89
Thu Nov 30 16:49:11 2023 (1) Sending HIP packet on UDP socket
ESP: get encrypted in input thread
 45000128 2a6d4000 4011fb52 0a000004
0a000002 29042904 01144cf9 00000000
3b200411 00000000 20010021 d1b1acc8
c339e992 ed11347f 20010021 d21f38e8
cdbaa60f 82925f94 0041000c 00000080
00000000 e3892018 f0810040 10cb07f8
20000000 40c87c07 a97f0000 18c97c07
a97f0000 008a290a a97f0000 4a9d9b62
c7fd4d45 c36636b5 49d73562 03dcd13e
0ca2715f a4d355ac e9fa69d8 97f9298e
f1010081 052a9da6 280b91c5 462b30c8
0bf0b28b d50a9708 95c53ff0 03519a17
d1239568 cb5b9441 c1440844 434ee4da
93cce24a 29aafbd9 3e1138fd 52d39ca7
bd2873a8 66481578 c5deba48 86dda5af
103bfbea 6f4bdd5a f338801f f4153bfe
85f4cad8 c58d9e3e 54487e99 de9cf67e
45baad24 97cdf916 c7c83b44 73b0cd02
3ed387f1 0c6636b5
ESP_RECEIVE_UDP_HIP_PACKET
 45000128 2a6d4000 4011fb52 0a000004
0a000002 29042904 01144cf9 00000000
3b200411 00000000 20010021 d1b1acc8
c339e992 ed11347f 20010021 d21f38e8
cdbaa60f 82925f94 0041000c 00000080
00000000 e3892018 f0810040 10cb07f8
20000000 40c87c07 a97f0000 18c97c07
a97f0000 008a290a a97f0000 4a9d9b62
c7fd4d45 c36636b5 49d73562 03dcd13e
0ca2715f a4d355ac e9fa69d8 97f9298e
f1010081 052a9da6 280b91c5 462b30c8
0bf0b28b d50a9708 95c53ff0 03519a17
d1239568 cb5b9441 c1440844 434ee4da
93cce24a 29aafbd9 3e1138fd 52d39ca7
bd2873a8 66481578 c5deba48 86dda5af
103bfbea 6f4bdd5a f338801f f4153bfe
85f4cad8 c58d9e3e 54487e99 de9cf67e
45baad24 97cdf916 c7c83b44 73b0cd02
3ed387f1 0c6636b5
Thu Nov 30 16:49:11 2023 (1) Sent I2 (652 bytes)
main.c UDP HANDLE
 03000000 00000128 45000128 2a6d4000
4011fb52 0a000004 0a000002 29042904
01144cf9 00000000 3b200411 00000000
20010021 d1b1acc8 c339e992 ed11347f
20010021 d21f38e8 cdbaa60f 82925f94
0041000c 00000080 00000000 e3892018
f0810040 10cb07f8 20000000 40c87c07
a97f0000 18c97c07 a97f0000 008a290a
a97f0000 4a9d9b62 c7fd4d45 c36636b5
49d73562 03dcd13e 0ca2715f a4d355ac
e9fa69d8 97f9298e f1010081 052a9da6
```

```
280b91c5 462b30c8 0bf0b28b d50a9708
95c53ff0 03519a17 d1239568 cb5b9441
c1440844 434ee4da 93cce24a 29aafbd9
3e1138fd 52d39ca7 bd2873a8 66481578
c5deba48 86dda5af 103bfbea 6f4bdd5a
f338801f f4153bfe 85f4cad8 c58d9e3e
54487e99 de9cf67e 45baad24 97cdf916
c7c83b44 73b0cd02 3ed387f1 0c6636b5
Thu Nov 30 16:49:11 2023 (1) Received HIP_R2 packet from 10.0.0.4 on udp socket length 296
 R2 TLV type = 65 length = 12
 R2 TLV type = 61569 length = 64
 R2 TLV type = 61697 length = 129
tlv_set_hostid_len() hi_len==144
tlv_set_hostid_len() EVP_PKEY_get_size(hi->evp_pkey)==128
HMAC_2 verify over 216 bytes. hdr length=26
validate hmac: 1
computed hmac: (64) HMAC_2 verified OK.
Validating signature of type 1
RSA HIP signature is good.
Using HIP transform of 4 ESP transform of 1.
Drawing new ESP keys from keymat index 128:
Key 4 (1,16) keymat[128] 0x 55f8c4c1 c2263afb 06761c83 69ce3c55
Key 5 (1,20) keymat[144] 0x 8003ca5c eaa16a7f 19a529cb f56574e2 80db6bc2
Key 6 (1,16) keymat[164] 0x 79a2ee57 f3d341c3 9746ae54 299f5ddb
Key 7 (1,20) keymat[180] 0x bb83dba7 ab80b9a1 f365e8be 0ab64b36 c52e1339
---------- HIP exchange complete. ----------
Thu Nov 30 16:49:11 2023 (1) Adding security association:
src ip = 10.0.0.2 dst ip = 10.0.0.4
SPIs in = 0x908252e4 out = 0xe3892018
Thu Nov 30 16:49:11 2023 (4) Base exchange completed from
2001:21:d21f:38e8:cdba:a60f:8292:5f94 / 10.0.0.2 / 1.146.95.148 to
2001:21:d1b1:acc8:c339:e992:ed11:347f / 10.0.0.4 / 1.17.52.127
Retransmitting 1 user data packets for 1.17.52.127.
ESP: send raw (unencrypted) in output thread
 9e648b96 a0b62e8c ce0b0ed3 08004500
005401a3 40004001 a2500192 5f940111
347f0800 ac31766c 000107bd 68650000
00009d6b 09000000 00001011 12131415
16171819 1a1b1c1d 1e1f2021 22232425
26272829 2a2b2c2d 2e2f3031 32333435
3637
ESP: send encrypted in output thread
 29042904 007c9af1 e3892018 00000001
9a940033 23f9ff00 850ab7cb e3ae4a93
4441bd1f ebe3d415 79b9ac8a 69e3949a
a8fb6605 4e3c1c33 be36d66d 6d1b7d03
e1255d34 dacb0f1f 99e8f475 13bc1ec0
80bd25c4 8d08fe52 e7d8c22b 1229f46b
4467c7cb 5bf6e6d0 06a975be a13a7931
69d1b470 dc68e5f3 ed18740c
ESP: get encrypted in input thread
 450000b0 e5dd4000 4011405a 0a000004
0a000002 29042904 009ce773 908252e4
00000001 dbba28fe 23e5bb72 933c05c8
600d2c03 ddb4c8c4 7c84c170 f37682da
cfd104de 42b2e028 300a6b70 f95c0e52
ef815fe3 ec07e5fa 9b4b6389 b2304afc
```

```
f86031f7 d8d4247b 02f99b5a 919d0e0c
0690c018 ad9f17c5 5c6ad467 43579b0f
cc85daa7 34b5617d f2f22f02 a827f2ab
9fadfced da775e69 6f76e0d1 b8ed6c33
04a8df5d 2dce6770 9233b657 612748d5
ESP: get raw in input thread
 00000000 00000000 00000000 00000000
00000000 2e8cce0b 0ed39e64 8b96a0b6
08004500 0082e5dd 40004000 bde80111
347f0192 5f940000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
ESP: send raw (unencrypted) in output thread
 9e648b96 a0b62e8c ce0b0ed3 080045c0
009e0292 00004001 e0570192 5f940111
347f0302 fcfd0000 00004500 0082e5dd
40004000 bde80111 347f0192 5f940000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000
ESP: send encrypted in output thread
 29042904 00bcae0e e3892018 00000002
1309b185 a5ee39ad 708c95b6 42e672ae
b0749939 519136b9 2450957e ea15b212
1178afff 30d7807f fb7d66ca 7e553239
68df462a 9e21df41 255b413a 3b85a9df
329cd95f 12b91787 f6efaa54 bfb47a38
3d0956d9 e4a9f672 86fc9dae 3ed83af0
6090d5e8 130d35ba fbb2354b 07593ed7
c4aebd4f c20ab6e9 0cc39df3 aa4d90e7
eac9c88f 882a7abb 6c2af59a f68e1e71
4f4db305 61686ec3 b41ec93a 1c3c7b33
867ab04d 6a5bb8c7 64459809
ESP: get encrypted in input thread
 450000f0 e5de4000 40114019 0a000004
0a000002 29042904 00dc8869 908252e4
00000002 017471b0 7fd11b67 a75cca19
39f5248b f6ed0b5f ee34e79b de447c73
018d90d0 1fb53224 7dc0856a ac586fa8
8e799669 6ace6e88 bbb28a93 f8570556
691c94e6 db4c24a1 1b382b3f 33f3fd92
b86951f9 87ebe4da 47ee4ee7 067719d3
e80731af 1fefdf4f 7251ed58 524c54e8
b3cad198 56d9f7c1 548c3153 b4131294
aaf95431 86f4ff7f e499f8fc a920871e
c2a9bd52 2378eb14 3ec7db23 6b7c76fd
5387dcef 9d8ed64a 080eff4e e3f74fda
2a59016f 62a22775 e22167f0 38ffafcc
cbf3dd59 94c5ee5c d016e106 6a20c2b7
ESP: get raw in input thread
```

```
 00000000 00000000 00000000 00000000
00000000 2e8cce0b 0ed39e64 8b96a0b6
08004500 00c2e5de 40004000 bda70111
347f0192 5f940000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
ESP: send raw (unencrypted) in output thread
 9e648b96 a0b62e8c ce0b0ed3 080045c0
00de0293 00004001 e0160192 5f940111
347f0302 fcfd0000 00004500 00c2e5de
40004000 bda70111 347f0192 5f940000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000
ESP: send encrypted in output thread
 29042904 00fc5ac0 e3892018 00000003
dd57c88e f2439afa c393eb9e 50e4798d
6d41f9ac 5f42d696 3f91298d 92626bdf
25dae323 366f1d19 77bf5c51 fb180548
00bff884 f0cca7db a51778e1 7bcd0934
2d4a760b b7545904 410a8ebe 17d552db
d235b2c9 0e69cff7 bb19d440 d31c5f28
ee989d56 a2d29bb1 349a39d2 7b86a95a
73020a8f 166ed363 9daced52 d7fc1b6a
b950f87c 0f064022 aa380e67 ba4eb3b4
c5328284 36a3edb1 58547c86 c42071cf
41a16f92 676c3152 91b1f765 ef534778
d40dc1e7 986f63ca b1a492f7 edd31a46
922ced7d 2495c5d1 d406f5e7 1a019d7f
8b8a8984 df71800f 1fe9e3f5 ac041cfd
b097de27 230222a5 0e173562
ESP: get encrypted in input thread
 45000130 e5df4000 40113fd8 0a000004
0a000002 29042904 011cb79d 908252e4
00000003 dee4df58 0ded71d6 79cf8cff
e11c9ab2 41de7e2f 194d997e 3f4c5d5e
71c4cc5d c5c712b7 6a475b6d 713317e9
e7cc55ea b7ed8468 24ac45c2 314739c5
d6f00e7e 00fba60a 0f1696dd 3938c14c
6d250a5a bb5eac6d 2e87f13d 4aefb96c
d1ce7339 32caf150 02b19700 8878a7ab
70be243b 28fc6afc 203bfb0c aa56d31c
```

```
05d39b5e a452081f 394c2616 ea75e96d
e46ba1ad 21e8bb9e d8457ea5 07bc249f
7d8cd409 22f4bdfd d0f08ed5 c116e726
e0a8b4e9 438d5ca7 0088cd3d 8d2bbdef
17739128 4212a854 bf4b1c18 9d1a3fa4
bae4516f 23d94a3e 20487e62 448444cc
9c77fd36 bab881c6 62808b0b d22db965
baf098fa b0e10877 ff0258ad 36103f9d
b164a0ca 9f56cd0d 4feb363e 04c9007b
ESP: get raw in input thread
 00000000 00000000 00000000 00000000
00000000 2e8cce0b 0ed39e64 8b96a0b6
08004500 0102e5df 40004000 bd660111
347f0192 5f940000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
ESP: send raw (unencrypted) in output thread
 9e648b96 a0b62e8c ce0b0ed3 080045c0
011e0294 00004001 dfd50192 5f940111
347f0302 fcfd0000 00004500 0102e5df
40004000 bd660111 347f0192 5f940000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000
ESP: send encrypted in output thread
 29042904 013cae92 e3892018 00000004
8d60bed6 dbac0e86 69c562ab 36ada175
22987a77 66828b6a a9aa6547 74ee8dbe
ccf482fe aa238aea b460aaad 6aa63798
4c2e3e09 091106f6 36e0261a e8465f7c
fe335482 a64bbb5c 493f434f 339246a6
331f493d b02155e7 5d567f7b 1509bef7
c8535266 fc25c2d4 e9c911c8 620c2899
55a7ebaf f58eaef1 23293482 cd132456
```

```
23fce2fc ca4b7cc7 52e58928 5c8fe2dc
6eb22f5f 61b93cf3 b196501e 239867e6
6a4b86b4 f971c796 1586494a e6212c82
a93e1fcc af6607db a76e444b 26257fb0
84aa9d49 5e14c320 df028bfa 22255b90
dd196586 591e66fe a72a4cc5 a3e7b859
e3182679 675d5d89 dfa9ee81 15cdbd12
0d79770d 15dfb663 07f645ef 6bfe8a3f
b431d952 c39df5cd b3ac2ca7 9e9b03ad
533a1819 7c206f1b 97842661 fcc349db
aa38e5df fa9f172a 8a76ea0c
ESP: get encrypted in input thread
 45000170 e5e04000 40113f97 0a000004
0a000002 29042904 015ce0c0 908252e4
00000004 3b655c8d f4e367ff ea4d022d
1c05e7bd b406c292 47eaa4d6 14fe1a5a
262c02f8 1ac05715 a4da1293 6917b45c
df364b32 02324075 ee0ee8c9 61db9ff8
f6fd2b56 c480711f 830bcd79 df9c1809
fe55f0e2 e9604fdb ab66e645 427ba325
fa41be82 3fdbd783 63f8000b 2658a8c1
ceb09b44 404fec70 cc0f02c8 f01d327e
662f20b0 715f3a64 da84d585 83cbe71e
869fd26c 5c43f1b7 24248185 7a4c0662
eeec6b48 294e28bf cc2adb89 864ba5b0
fe4db3a1 7751ae7d fe9f33bd da567e73
b301725c 378ad3e1 5ea8184c 747571cd
a7f0b578 428ecb87 3b67fb26 344a7a02
47276613 19c2b4a4 ed6db140 23838a72
a3394e46 ed180691 f223ae2f 918665e8
cde4fb4e 144efacb e53c4758 3ca30241
f21b4ccd fd146c59 8d0d6f25 54768274
730a3926 2e8fd375 a473a857 fd8a2d74
ba8091be 09db6326 08a510ce f29301e8
15e28e2d 8837dcfe 141d7068 8092fe25
ESP: get raw in input thread
 00000000 00000000 00000000 00000000
00000000 2e8cce0b 0ed39e64 8b96a0b6
08004500 0142e5e0 40004000 bd250111
347f0192 5f940000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
```

```
ESP: send raw (unencrypted) in output thread
 9e648b96 a0b62e8c ce0b0ed3 080045c0
015e0295 00004001 df940192 5f940111
347f0302 fcfd0000 00004500 0142e5e0
40004000 bd250111 347f0192 5f940000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000
ESP: send encrypted in output thread
 29042904 017c6980 e3892018 00000005
16dd452b 342fec05 296859de 64c0a51c
2a203ad5 4f626c39 351b7426 c5c5bf0d
9db3f54e 589ddcfc 80bf5b31 1614a466
7b19b092 72b28988 75eb3e23 05e8998c
2f8cc8f5 b788d3d5 689c2260 f9706404
56b49aad 0a7a6cda 5b9e42a5 a19031df
17c03fbb 691a7c4c 4165fd81 50d9c115
1bb5b15a 68de545e 5f50b876 f99b518d
b9b6e22c ade2b9d6 111189ed 8980c1cd
d2ce4e9f 300085eb d14329ea 9b3cb949
678944a9 7586e527 917912e8 a9a0425d
599e896d 101001ce d12581f1 7142b54b
cfd98b1f 950d814c 29cc35a9 cd2f7cb0
9cfffcf5 28907008 4afa3445 0ed56626
f4d16fcd 955a1567 765b8025 54abf656
faf2d725 2ac6f21c 220c6683 9253cddb
16dc4ab7 9b1812ad 9f092fa6 17fbbd3f
bf046d8a c80795d7 1d42f63c b084dd61
064a4844 934eae88 e0472e78 d8e05f34
7af935de cacda258 c8d3f248 b61d287c
8b495864 f90cbfd8 b95b0fef 06256ad9
88b1b694 ea2c3e50 3a2ed56b c2ca9e20
56a1c9eb f801fcb5 b361cee7
ESP: get encrypted in input thread
 450001b0 e5e14000 40113f56 0a000004
0a000002 29042904 019c6553 908252e4
00000005 29f38b0a f635b81e a27cd9a0
7dffeeda b5fcd592 00b7802a accc8bfd
6fa0ec10 bcbf4fe7 b01b4110 f37d9a0c
65f05813 79b0e2a7 5a5449f6 8b18eff1
85504293 974b867f 07ddf069 4ec3c547
```

```
18b3bacf 83b18f12 9fef3f50 9eb76be4
e4577e32 bbbeda78 705959a9 95172881
5595173a 54d6b632 7e8a9506 eb9be094
98065433 86972ada 06efc0ae eed276a5
551c52d8 fe4a6431 ff9de957 a7635d82
726d606e 24b688f1 f8034601 c5bdf838
80cd1933 7c967c38 2f86525f 383bd559
97720c3e acca4029 d5229ed5 259522df
2527f035 def82a38 b213eaa3 1154d1dc
98ed215c a098b94a 32544321 922d371a
72bf6c31 8bc99ed0 72286697 7c2d99a8
7e274189 32735e85 df999fad 568c7bba
c2c562ef ffbe9b3b 4a1afd31 ea528960
28e44edd 330f83e1 89aa28b7 549d3e7e
063f9cbf 7bffae6f 09794db0 85caa42d
c764b5c8 ea45e757 6155ec6d f1faa656
5c82567f bbd0d2e1 bdd10e01 d853eced
892eaa8f ae67b243 c00f7044 b3f59ce5
8058e7db 31e58f78 8ae8ed1d f89718d2
441b73a1 af181f75 9b9573d0 ded543ab
ESP: get raw in input thread
 00000000 00000000 00000000 00000000
00000000 2e8cce0b 0ed39e64 8b96a0b6
08004500 0182e5e1 40004000 bce40111
347f0192 5f940000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000

...

None
HIPNode2 known_hosts ########################################
HIPNode4 known_hosts ########################################
Hit enter to poweroff the system
```