DRIP Implementation

Andrei Gurtov & students
Linköping University, Sweden

DRIP IETF 109th Online Meeting
November 18, 2020
Starting Points

• OpenHIP (v2 alpha branch) - 4 students
  • [https://bitbucket.org/openhip/](https://bitbucket.org/openhip/)
  • draft-moskowitz-hip-new-crypto-06

• OpenDroneID - 5 students
  • [https://www.opendroneid.org/code/](https://www.opendroneid.org/code/)

• TDDE21 Advanced Project: Secure Distributed and Embedded Systems (6 ECTS) Sep-Dec 2020
  • [https://www.ida.liu.se/~TDDE21/](https://www.ida.liu.se/~TDDE21/)
New Requirements for HIPv2

- New cryptographic algorithm (EdDSA)
- New ORCHIDs - include additional info in Host Identity Tag (HIT), needed for hierarchical HITs
- Hierarchical HITs - embed information about the issuing authority inside HIT
New ORCHID

- Overlay Routable Cryptographic Hash Identifiers
- Endpoint identifiers at applications
- Before => ORCHID := Prefix | OGA ID | Encode_96(Hash)
  - Prefix = 2001:20::/28 (IANA)
  - ODA ID = 4 bit hash identifier
  - Encode_96(Hash) = Middle 96 bits of the hash output
- Now => ORCHID := Prefix | OGA ID | Info (n) | Hash (m)
  - New prefix for HHITs
  - Hash(m) = Hash function which outputs m bits, use cSHAKE
  - Encode_96(Hash) split into Info(n) and Hash(m), Info(n) used in HHITs as a tag
New Crypto

- EdDSA (Edwards-curve Digital Signature Algorithm) is a digital signature scheme that is based on elliptic-curve cryptography.
  - Designed to be a fast algorithm without sacrificing security
  - Less dependent on a good random number generator, compared to ECDSA

- The Keyak cipher is used as a lightweight alternative to AES, and also supports authentication of the encrypted data
  - Move to Xoodyak, follow NIST

- The KKDF key derivation function (based on KMAC) is a more efficient alternative to the HMAC-based HKDF
Testing

- Common Open Research Emulator (CORE)
  - Emulate real computer networks
- Current tests
  - Python 3.8
  - Core 7.2
  - Use of standard libraries
  - Separation of concerns
HIPv2 New Base Exchange Works

with solution j: edfae99a 056ad684 95df2d05 6d2c9d52
521473c3 356c3c24 ba88bd18 35120bab
Using HTP transform of 6.

Drawing new HIP encryption/integrity keys:
Key 0 (6,32) keymat[ 0] 0xbdb775a4 6cc34d87 f1d4d794 775e1570
91a3e624 643c63b8 e9f0e0e9 e9aad150
Key 1 (5,32) keymat[ 32] 0xb9e31f27 1157e0e5 c64710e0 a0f1f377
acb1a8a 1532b19e a6e03e9f 0c3f6661
Key 2 (6,32) keymat[ 64] 0xeabc191 9483c1d f9deab4f e9bfc8f6
9bd1cb91 d992b21c b4db4ab 9651e963
Key 3 (5,32) keymat[ 96] 0xe35317bf 247f8f80 bef280e1 9e69e9ae2
10e9e5f0 35aa363f 31ffedc9 b1a3d068

Using DH public value of len 44: 0x

River Keyak encryption key: 0xbdb775a4 6cc34d87 f1d4d794 775e1570
91a3e624 643c63b8 e9f0e0e9 e9aad150

Encrypting 96 bytes using River Keyak.

HMAC computed over 336 bytes hred length=41

HMAC length=68

** HMAC_md_len=32, hmac_size=64
SHA1: c0e5e6ba 6778f7f1 50bfcf4c fdd399f0
6057a9a5 5a982e51 425f4261 a0b14fd6
Signature: Taa339ae cf1cd6ae c8a9866c c65c0ad2
faa28729 3df9f214 6426ff0e 6f210f69
52a7cbe9 64bb4159 993a04d1 ab4f1c18
e379ca1c 95a627d5 59d425d f6df2e80
Sun Nov 1 11:46:43 2020 (1) Sending HIP_I3 packet (488 bytes)...
Sun Nov 1 11:46:43 2020 (1) Sending HIP packet on UDP socket
Sun Nov 1 11:46:43 2020 (1) Sent 12 (492 bytes)
Sun Nov 1 11:46:43 2020 (1) Received NOTIFY packet from 10.0.0.21 on udp socket length 152
NOTIFY TLV type = 832 length = 4
** NOTIFY TLV type = 6169 length = 65
** Validating signature of type 5
SHA1: c74205dd efa2850a bbe17ad4 083c76d4
5886dede 6263e065 d2c2625e dfad3c4
EdDSA HIP signature is good.
** Received NOTIFY from 10.0.0.20: HIT does not validate HIT.
DRIP Implementation

Broadcast a Drone ID over Bluetooth or WiFi as a HIP Host identity tag

- Raspberry Pi + GPS receiver
- 20 Bytes
- Observer app in Android
- Specifications from DRIP IETF Working Group
  - UAS Remote ID: draft-ietf-drip-uas-rid-01
Android Application

Extending software

OpenDroneID
- Bluetooth only
- Includes parser for Bluetooth messages
- Functionality for Maps, points and information about the drone inside the GUI

Method to scan
- Connect to each individual drone (Since the raspberry Pi needs to be its own access point).
- Read and parse the messages the drone is sending.

WifiAnalyzer
- Analyzing Wifi-networks
- Bands, SSID, connectivity status, and more
Android Application for Observer

- New application to test the wifi in isolation
- Adding the wifi-module to the OpenDroneID Bluetooth receiver

Currently:

- Reading WiFi-SSIDs. Next: WiFi NAN mode?
- No low-level broadcast reading
- Researching open source WiFi scanning applications for methods already in use

Possible switch to Linux application to get low-level access to broadcast messages since methods in Android might require rooting the Android device.
This is WiFi. Maybe a dead-end.
Now switched to Bluetooth 4&5 connectivity, less issues.
Broadcasting over bluetooth

Multiple Beacon standards

Broadcasting over bluetooth

- Easy to switch between standards, they have very similar structure
- Easy to send out beacons using hcitool on the RPi, only requires a few lines of code
- Much easier to scan, there is already widespread support for scanning beacons on both Android and IOS devices
- Max range for BLE with Bluetooth 4 is around 50m
TODO

• Complete a working prototype by end of 2020
  • About 60% progress so far
  • Broadcast RID HIT with Bluetooth and WiFi
  • Observer Android App
  • With HIPv2 features, ORCHID2.5, Xoodyak, draft-ietf-drip-rid-04
• Test on a flying drone DJI Phantom 4 Pro+ V2.0
  • RPI+GPS+Battery+BT Dongle as a payload
• Future: draft-moskowitz-drip-secure-nrid-c2-01