Implementing Host Identity Protocol using Python

Abstract—**Host Identity Protocol, or HIP, is layer 3.5 solution, which was initially designed to split the dual role of the IP address - locator and identifier. Using HIP protocol one can solve not only mobility problems, but also establish authenticated secure channel. In this short report we will introduce description of the implementation of HIP protocol using Python. We will also present the microbanchmarking results for various cryptographic primitives.**

I. Introduction

II. Background

In this section we will describe basic background. First, we will discuss the problem of mobile Internet and introduce the Host Identity Protocol. We then move to a discussion of other layer 3 security protocols. We will conclude the section with the discussion of Elliptic Curves and a variant of Diffie-Hellman algorithm, which uses EC cryptographgy (ECC).

A. Dual role of IP

Internet was designed initially so that the Internet Protocol (IP) address was playing dual role: it was the locator, so that the routers could find the recipient of a message, and identifier, so that the upper layer protocols (such as TCP and UDP) can make bindings (for example, transport layer sockets use IP addresses and ports to make a connections). This becomes a problem when a networked device roams from one network to another, and so the IP address changes, leading to failures in upper layer connections.

B. Layer 3 security protocols

There are a lot of solutions today which allow communicating parties to authenticate each other and establish secure channel. But only few provide a separation of identifier and locator.

Locator Identifier Separation Protocol (LISP)

Identifier/Locator Network Protocol (ILNP)

Secure Shell protocol (SSH) is one security solution [1]. SSH is the application layer protocol which provides an encrypted channel for insecure networks. SSH was originally designed to provide secure remote command-line, login, and command execution. But in fact, any network service can be secured with SSH. Moreover, SSH provides means for creating VPN tunnels between the spatially separated networks.

IPSec

Internet Key Exchange protocol (IKE)

Mobile TCP (mTCP) There are a lot of solutoins for mobility support. For sampling see Mobile IP, ROAMIP and Cellular IP.

C. Diffie-Hellman (DH) and Elliptic Curve DH

Because `pycryptodome` library does not support Diffie-Hellman (DH) and Elliptic Curve Diffie-Hellman (ECDH) algorithms, we have sat down and derived our own implementation of these protocols. Here we will mention some background on Elliptic Curve Cryptography (ECC) and discuss the implementation details of ECDH.

Elliptic curves have the following form $y^2 \equiv x^3 + ax + b \mod p$. For the curve to have at least one root the discriminant should be non zero. In other words, $\Delta = -16(4a^3+27b^2) \not\equiv 0 \mod p$, where p is a large enough prime number.

By defining a binary operation, which is an addition operation, we can make elliptic curve form an abelian group. Remember, abelian group has the following properties: (i) closure, meaning that if $A, B \in E$, then $A + B \in E$, (ii) associativity: $\forall A, B, C \in E$ follows that $(A + B) + C = A + (B + C)$. (iii) existence of identity element I, such that $A + I = I + A = A$, (iv) existence of inverse: $\forall A \in E$ $A + A^{-1} = A^{-1} + A = I$; (v) commutativity: $A + B = B + A$ $\forall A, B \in E$. Finally, we should mention that there should exist an element G, such that multiple additions of such element with itself, kG, generates all other elements of the group. Such groups are called `cyclic` abelian groups.

Lets define O, a point at infinity, to be identity element, such that $P + O = O + P = P, \forall P \in E$. Also, we define $P + (-P) = O$, where $-P = (x, -y)$. Next lets suppose that $P, Q \in E$ (reads P and Q belong to elliptic curve), where $P = (x_1, y_1)$ and $Q = (x_2, y_2)$. We can then distinguish the following three cases: (i) $x_1 \neq x_2$ (in this case the line, which passes through the two given points, must intersect the curve somewhere at a third point $R = (x_3, y_3)$), (ii) $x_1 = x_2$ and $y_1 = -y_2$ (in this case the line is vertical, and it does not pass through a third point on the curve); and finally (iii) $x_1 = x_2$ and $y_1 = y_2$ (in this case the line is tangent to a curve, but still crosses the curve at a third point $R(x_3, y_3)$). Given case (ii), we can define a negative point as $-R = (x, -y)$.

In the first case, line L passes through points P and Q. Using simple geometry, we can derive an equation of the line as follows: $y = \beta x + \upsilon$, such that

$$\beta = (y_2 - y_1)(x_2 - x_1)^{-1}$$

Also, we can find υ as

$$\upsilon = y_1 - \beta x_1 = y_2 - \beta x_2$$

In order to find the points that intersect the curve, we can substitute $y = \beta x + \upsilon$ into equation of an eliptic curve:

$$(\beta x + \upsilon)^2 = x^3 + ax + b$$

By rearranging the terms of the equation, we obtain:

$$x^3 + ax + b - \beta^2 x^2 - 2\beta v x - v^2 =$$
$$x^3 + (a - 2\beta v)x - \beta^2 x^2 + b - v^2 = 0$$

But since the obtained equation has three roots we have:

$$(x-x_1)(x-x_2)(x-x_3) = (x^2-xx_2-xx_1+x_1x_2)(x-x_3) =$$
$$x^3-x^2x_3-x^2x_2+xx_2x_3-x^2x_1+xx_1x_3+xx_1x_2-x_1x_2x_3 =$$
$$x^3 - (x_3+x_2+x_1)x^2 + (x_2x_3+x_1x_3+x_1x_2)x - x_1x_2x_3$$

But noticing that the $\beta^2 = x_1 + x_2 + x_3$, we have:

$$x_3 = \beta^2 - x_1 - x_2$$

Moreover, since $P + Q = -R$, we have:

$$-y_3 = \beta(x_3 - x_1) + y_1$$

or

$$y_3 = \beta(x_1 - x_3) - y_1$$

The second case is simple, by definition we have $P - Q = O$. And finally, we should mention that the third case is much like the first case, but with one difference - the line that passes through P and Q is tangent to the curve, because $P = Q$. By applying an implicit differentiation to an original function of an elliptic curve, we have:

$$2y\frac{\partial y}{\partial x} = 3x^2 + a$$

From this we can derive β as follows:

$$\beta = \frac{\partial y}{\partial x} = (3x_1^2 + a)(2y_1)^{-1}$$

.

This expression allows us to derive the x_3 as follows:

$$x_3 = \beta^2 - 2x_1$$

Finally, just as in the first case, we have $y_3 = \beta(x_1 - x_3) - y_1$. Of course, all operations are done modulo prime p.

We now turn to discussion of ECDH protocol and some of the implementation details. We have used the parameters for the elliptic curve which are defined in RFC5903 [2]. ECDH proceeds in the following manner: Party A generates random number i, where the size of this random number is equal to the number of bytes that make up the prime number p (this is specified in the parameters set). Party B generates in a similar fashion number j. Both parties, using point on a curve G, which is also a generator (again specified in RFC5903), compute public keys: $K_A = iG$ and $K_B = jG$. We have used well-known **double and add algorithm** [3] to efficiently compute the multiplication. Next parties exchange the public keys and derive a shared secret as follows $S = iK_B = jK_A = ijG$.

To test the implementation we have used test vectors provided in previously mentioned RFC.

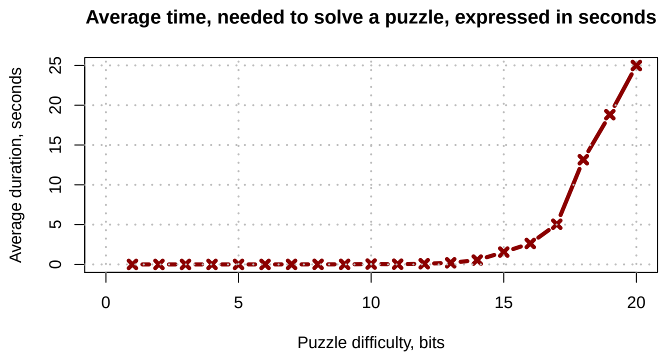

Fig. 1: Average duration of puzzle solving

Symmetric key sizes, bits	DH keys, bits	ECDH keys, bits
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

TABLE I: Security strength of keys

III. Hardware and software

IV. Experimental evaluation

To demonstrate the performance of ECHD and DH algorithms we have executed the key exchange algorithms 100 times for various groups. Thus, in Figure 2 we show the performance of DH and in Figure 3 we show the performance of ECDH for various curve parameters. To understand how two are related in Table I we show the sizes of various keys and how they are related to symmetric keys. Obviously, ECDH shows far better performance than regular DH algorithm. This performance improvement is largely due to reduced key sizes.

[4]

V. Conclusions

In this short document we have discussed the implementation details of Host Identity Protocol (HIP), which as a layer 3.5 security solution aiming at separation of dual role of an IP address. HIP protocol not only solves the problem of separation of roles of the IP addresses, but can also be a secure mobility solution. There are many applications of HIP in modern networks - from establishing a secure channel between stationary network entities, to secure mobility solutions.

We have created a Python based implementation of HIP and experimented with it: (i) we have made several microbanchmarkings, and (ii) we completed several rounds of stress tests of the solution (basically, we have made several concurrent connections to HIP server and measured duration of HIP base exchange).

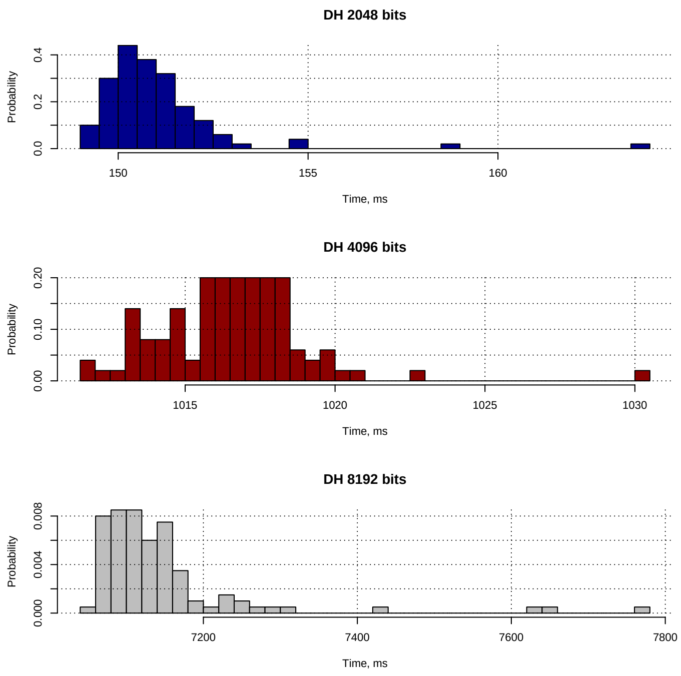

Fig. 2: Diffie-Hellman key exchange duration (total)

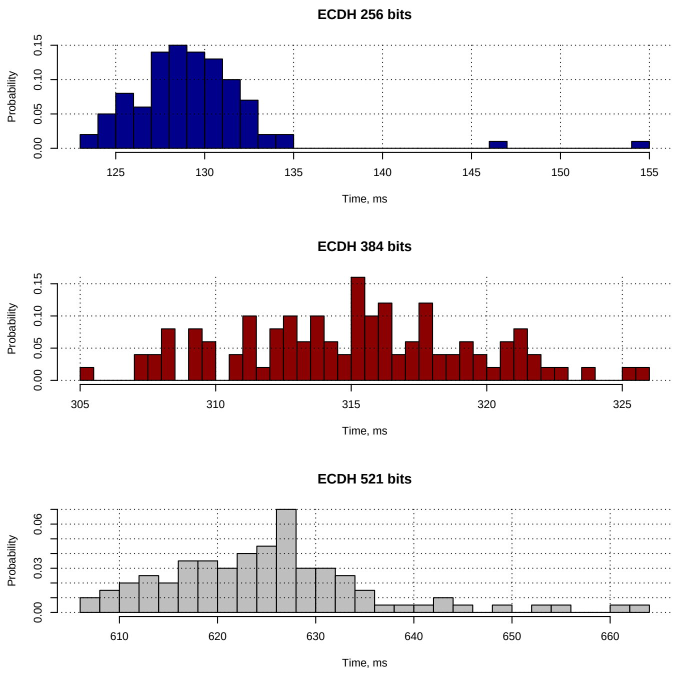

Fig. 3: Elliptic Curve Diffie-Hellman key exchange duration (total)

References

[1] Secure shell. https://en.wikipedia.org/wiki/Secure_Shell.

[2] D. Fu and J. Solinas. Elliptic Curve Groups modulo a Prime (ECP Groups) for IKE and IKEv2, 2010.

[3] D. Stinson. *Cryptography: Theory and Practice, Second Edition*. CRC/C&H, 2nd edition, 2002.

[4] M. who. Some article. pages x–y, 2019.