# Implementation of Secure Multipath Resilience in HIPLS using Mininet and SDN Integration


Fabian Aronsson
*IDA*
*Linköping university*
Linköping, Sweden
fabar453@student.liu.se

Viktor Bergström
*IDA*
*Linköping university*
Linköping, Sweden
vikbe588@student.liu.se

Sandeep Chiru Kandoth
*IDA*
*Linköping university*
Linköping, Sweden
sanch066@student.liu.se

Axel Eklöf
*IDA*
*Linköping university*
Linköping, Sweden
axeek324@student.liu.se

André Koch
*IDA*
*Linköping university*
Linköping, Sweden
andko485@student.liu.se



***Abstract*—This project explores the implementation of multi-party key exchange with the Host Identity Protocol (HIP) using Mininet and Software defined networking (SDN). This is done by implementing Burmester-Desmedt (BD) key exchange with an elliptic curve in an open source HIPLS implementation. The result is a more efficient implementation that still can establish trust in a zero trust environment. A hub-spoke topology was adopted to reduce full-mesh tunnels and decrease attack surface. The results demonstrate enhanced security and scalability compared to previous implementations, while addressing limitations like computational overhead. This work contributes to advancing secure communication in decentralized, zero trust networks.**



***Index Terms*—Multi-party key exchange, Host identity protocol, HIP, Mininet, Burmester-Desmedt, Diffie-Hellman, Elliptic curves, zero trust.**


## I. Introduction

This report presents the project for implementation of Secure Multipath Resilience in HIPLS using Mininet and SDN Integration. It does this by describing the background, methodology, result, discussion and conclusion. The goal of the project was to implement secure multi-party key exchange in the open-source Python implementation of HIPLS [1]. The result is meant to improve the already existing implementation, so it can be further developed in the future. The proposed implementation aims to make the current one more efficient by enabling better scalability for use with more nodes. In this first chapter the motivation and purpose behind the project, the research questions as well as the context of the project is presented.

### A. Motivation

In an increasingly interconnected digital landscape, the need for secure and efficient communication is growing increasingly relevant. The Host Identity Protocol (HIP) was designed to address certain design flaws in the basic TCP/IP architecture. These flaws are primarily related to the IP address being used for both identity and location [2].

HIPLS is an open-source Python implementation of HIP. However the current implementation lacks support for multi-party key exchange, which is critical for scaling communications in a zero trust environment.



This project aims to address this problem by implementing secure multi-party key exchange in the current implementation. Multi-party key exchange would allow multiple parties to establish a shared secret key, reducing the need for pairwise key generation. This would greatly improve efficiency and scalability. The key exchange also provides integrity, while the HIP provides authentication, which both are requirements to establish trust in a zero trust network.

### B. Purpose

The purpose of the project is to make the HIPLS implementation more efficient and scalable to larger networks. As well as determine if it is possible to implement multi-party key exchange in the current implementation in an effective way. The result of this project also serves to give the participants experience working with network-protocols and advanced subjects in computer security such as cryptology and networking. The purpose of the report is to document the project.

### C. Research Questions

This project aims to implement and evaluate secure HIP multi-party key exchange, ensuring a more efficient but still secure communications. To do this it answers the following research questions:

1) How to implement the Burmester-Desmedt key exchange protocol into the current Python HIP implementation?
2) How do the changes to the Python implementation affect the performance of the Host Identity Protocol?
3) What differences can be identified between the proposed and current implementation with regards to trust, complexity, attack surface and IPsec mode?

### D. Limitations

The project and the report will only cover multiparty key exchange in HIPLS and not how to secure multi-path resilience. In the implementation of the HIPLS the group will have to use IPsec as a base, but it will not be thoroughly explained in this report as it is outside the scope of the project. For further reading on IPsec see [3].

### E. Context

This project was done by five students. It is part of the advanced project course in information security: TDDE63 (6hp) for the secure systems master at Linköping University. All participants are in their fifth year of studies. The purpose of the course is to do a project in the student's master's area before they do their master's thesis. In the context of this report, all mentions of HIP refers to the second version of the protocol HIPv2 [4].

## II. Background

This section describes the necessary background knowledge to understand the project, as well as the background of the group members and the project itself.

### A. Abbreviations and Acronyms

The abbreviations and acronyms in table I are used in the report.

TABLE I
List of Abbreviations and Acronyms

| Acronym | Definition |
|---|---|
| LAN | Local Area Network |
| HIP | Host Identity Protocol Version 2 |
| HI | Host Identity |
| HIT | Host Identity Tag |
| LSI | Local-scope Identifier |
| HIPLS | Host Identity Protocol-based Virtual Private LAN Service |
| BD | Burmester-Desmedt |
| DH | Diffie-Hellman |
| SDN | Software Defined Networking |
| BEX | Base exchange |
| IPsec | Internet Protocol Security |
| CE | Customer Edge |
| MTU | Maximum transmission unit |
| ESP | Encapsulating Security Payload |
| AH | Authentication Header |

### B. Zero trust environment

To give a better understanding of the reasoning behind the motivation of the project, zero trust must be explained.

A zero trust environment is, as the name implies, an environment where no device or resource is to be trusted. It operates under the principle "never trust, always verify" [5]. This puts an emphasis on developing tools and systems to be able to authenticate and authorize resources and devices to ensure security. In a zero trust environment the assumptions are that the network constantly operates in a hostile environment, facing threats both internal and external throughout its life cycle. Therefore, the credibility of the network cannot be determined by location alone, and all devices, users, and traffic must be continuously authenticated and authorized [6].

### C. IPsec

The current implementation uses IPsec in Encapsulating Security Payload (ESP) mode to secure communications. While ESP provides confidentiality, integrity, and authentication, it introduces additional overhead by encrypting the payload data. In the setup, where information is already encrypted by other mechanisms, ESP results in double encryption, which is computationally redundant and adversely impacts efficiency [3]. To address this, the group attempted to change to Authentication Header (AH) mode. AH ensures data integrity and authentication without encrypting the payload, thereby eliminating the unnecessary computational burden associated with double encryption.

*1) ESP vs AH:* The ESP and AH are two IPsec modes that do different things. ESP is designed to encrypt the contents of IP packets, so the data is confidential. It can also do optional authentication and integrity checks to ensure that the data hasn't been tampered with during transit. AH on the other hand is focused on integrity and authenticity of the data by securing the entire IP packet including certain fields in the header. But AH doesn't encrypt the packet contents, so the data is visible. The main difference is that ESP provides encryption for privacy and AH provides authentication and integrity without confidentiality [3].

*2) ESP NULL Encryption:* An alternative to AH is ESP NULL Encryption. ESP NULL Encryption works the same way as normal ESP but it uses an encryption algorithm called NULL encryption [7]. This algorithm doesn't encrypt the packet. In practice and for the purposes of this project this achieves about the same effect as AH. This is because the reason to use AH in this project is to not double encrypt the packages.

### D. Multi-party key exchange

Regardless of which mode is used, IPsec requires a key exchange mechanism in order to establish symmetric keys between its participants. The current HIPLS implementation establishes a unique key for each pair of connected nodes in the network by running a Diffie-Hellman key exchange. This approach, while conventional and secure, leads to a large number of keys needing to be generated in order to provide the benefits of IPsec for the network.

An alternative method is to implement a multi-party key exchange, for example Burmester-Desmedt. In such a scheme all nodes are able to agree on a single shared key, thus dramatically reducing the performance cost of securing the network as it scales.

*1) Diffie-Hellman:* Diffie-Hellman key agreement is a cryptographic protocol in which two participants agree on a secret shared key. The protocol can be performed over a public or unsecured channel without revealing the shared key to a potential eavesdropper [8].

The method works as follows. Both participants must first agree on a public generator ($g$) and a public prime number ($p$). The two participants ($A$,$B$) then each generate a private secret ($x_a$ and $x_b$). Each participant then computes and sends the following public values to each others using equations 1 and 2.

$$Y_A = g^{x_a} \bmod p \tag{1}$$

$$Y_B = g^{x_b} \bmod p \tag{2}$$

Both participants can now compute the same shared key, using the equations 3 and 4.

$$K = Y_B^{x_a} \bmod p \tag{3}$$

$$K = Y_A^{x_b} \bmod p \tag{4}$$

*2) Burmester-Desmedt:* The Burmester-Desmedt Key Agreement protocol is a conference keying scheme which allows an arbitrary number of participants to agree on a shared secret key over a potentially insecure channel. Burmester-Desmedt can be viewed as a generalization of Diffie-Hellman for three or more participants.

The protocol consists of two rounds and works as following [9].

First, all participants must agree on a public generator ($g$) and a public prime number ($p$). Then each participant $i \in [0, n]$ generates a private secret ($r_i$). After this each participant computes and broadcasts a public value given by equation 5.

$$z_i = g^{r_i} \bmod p \tag{5}$$

Upon receiving all public $z_i$ values each participant now starts the second round by computing and broadcasting another public value given by equation 6

$$X_i = \left(\frac{z_{i+1}}{z_{i-1}}\right)^{r_i} \bmod p \tag{6}$$

Upon receiving all public $X_i$ values each participant now computes the shared key using equation 7

$$K = z_{i-1}^{r_i n} \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \cdot \ldots \cdot X_{i-2} = g^{r_1 r_2 + r_2 r_3 + \ldots + r_1 r_n} \bmod p \tag{7}$$

### *E. Elliptic curves*

Diffie-Hellman and Burmester-Desmedt can be extended to make use of elliptic curve cryptography (ECC). In the context of this project an elliptic curve is a plane curve over a finite field which consists of the points that satisfy equation 8.

$$y^2 = x^3 + ax + b \tag{8}$$

With elliptic curves the generator is represented as a point on the curve $G$ instead of an integer modulo $n$. Elliptic curves have grown in popularity [10] due to its smaller key size needed to maintain proper security, for example a 256 bit key created with an elliptic curve gives the same security as a 3072 bit RSA key. This is because the security of the elliptic curve discrete logarithm problem has a solution with worse time complexity than the "normal" discrete logarithm problem [11]

There are many different elliptic curves used in cryptographic contexts that come with different drawbacks and benefits. Two common ones being secp256k1 and curve25519. For this project the group chose secp256k1 V-B2.

*1) The Host Identity:* It achieves this separation of identity and location by the introduction of a new namespace, the Host Identity (HI). The host identity is derived from a self-generated public-private key-pair, so a host using the private key can prove it owns the public key and thereby prove its identity [12]. Since the public-key is too large for use in a packet and to keep compatibility with existing applications, two other identifications are used. First the Host Identity Tag (HIT) which is a hash of the public-key with the same length as an IPv6 address, and secondly The Local-Scope Identifier (LSI). The LSI can be constructed by taking the last bytes of the HIT. IPsec is used to carry the HIP packets, using Encapsulating Security Payload (ESP) in transport mode [2].

*2) Base exchange:* For a visual overview of the exchange, see fig. 1. The base exchange is initiated when the initiator first sends a packet (I1) containing its HIT and the responders HIT, which it gets from the DNS server.

The responder then responds by sending back the HITs together with the Diffie-Hellman key (DH), the public-key of the responder and a puzzle used to combat Distributed Denial-of-Service (DDoS) attacks. This packet (R1) is also signed.

The initiator then sends back the HITs together with the solution to the puzzle and DH, this packet (I2) is also signed.

The final packet (R2) from the responder is only a signature which the initiator verifies [2].

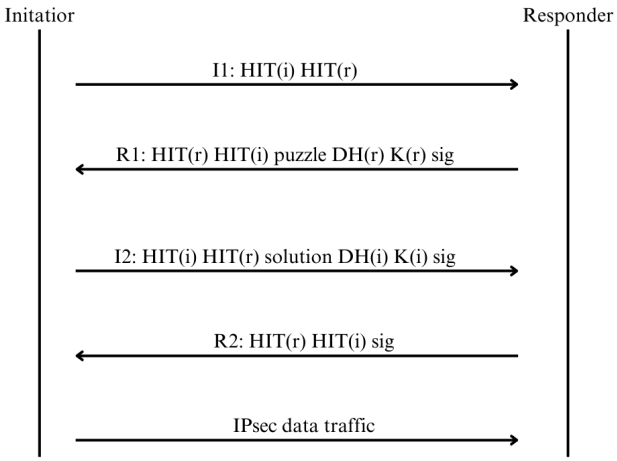


Fig. 1. Diagram of the HIPv2 base exchange

What these steps accomplish is a four-way handshake where the initiator and responder authenticate each other and share DH public values which results in the generation of a shared session key [12].

### F. Topology

A network topology defines the arrangement of nodes and connections in a network [13]. To integrate BD into the existing HIP implementation, the previous mesh topology required modification. A hub-and-spoke topology was selected for this purpose, as illustrated in fig. 2. Every node in the figure except the switch is a router.

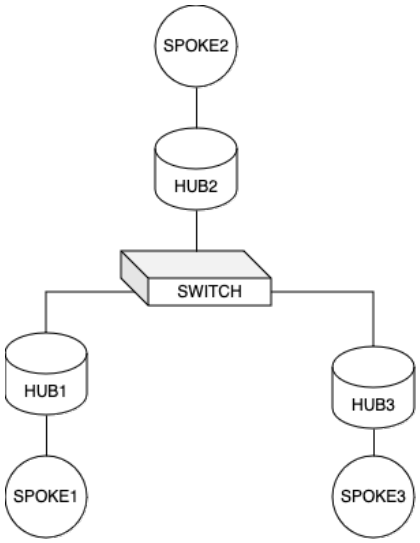


Fig. 2. Hub-spoke topology

The hub-spoke topology is beneficial because adding more spokes will not affect the other nodes, which provides flexibility. It is also more economical in terms of use of resources since there are less edges in total compared to a topology where everything is connected [14].

For the purpose of multi-party key exchange there is also a demand for a separation between different types of nodes. One of them (spokes) has to initiate the key exchange and one has to wait until the initiation is complete (hubs).

One advantage with the hub and spoke topology is that it avoids the full mesh problem, were all nodes are connected to each other, see fig. 3. The problem with full mesh is that it makes the amount of edges grow quadratically for each node added. With the hub-spoke topology only the hubs grow quadratically when a hub node is added while the spokes have a linear growth for each spoke added to a hub. This allows for a less computationally complex topology and a ”cheaper” setup [15].

### G. Host Identity Protocol

This project will implement multi-party key exchange into HIP, which is a protocol that aims to address certain flaws in the basic TCP/IP protocol. These flaws mostly stem from the fact that the IP address is used for both identity and location in a conventional network. The Host Identity Protocol (HIP) was therefore designed to separate the identity from it.

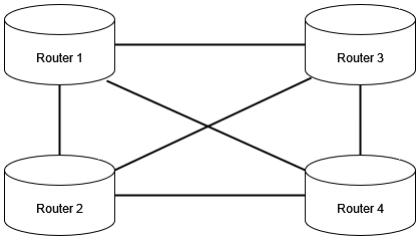


Fig. 3. Full mesh topology

### H. Tools

During the project several tools will be used.

- Python - This programming language will be used to write the implementation [16].
- Mininet - Used to create virtual networks that can be used to experiment with the HIPLS implementation, as well as for testing future code-changes [17].
- Github - Used for collaboration between team-members, to manage tasks and to version control the code [18].
- Virtual Machine - To run the environment for implementing the proposed HIPv2 the groups used different virtualization programs. The Windows users used VirtualBox [19] and the Mac users used UTM [20].

### I. Project background

A basic version of HIPLS, with Diffie-Hellman key exchange is already implemented in Python and the repository on Github is regularly updated [1]. The group will also have regular meetings with a supervisor who is experienced in HIPLS and Mininet.

### J. The groups earlier experience

The group consists of final-year Computer Science students, all specializing in cybersecurity. Collectively, the group have solid understandings in computer networks, gained through courses taken at Linköping University and other professional experiences. This shared background provides the group with practical insights into network protocols and secure communications. Three members of the group have also completed a course in cryptology, giving them understanding cryptographic protocols. No group member has previously worked with HIPLS or Mininet. The combination of expertise in networks and cryptography ensures a well-rounded skill set for this project allowing the group to efficiently design a solution.

## III. Methodology

This section will explain how each research-question was answered.

### A. Literature review

To find the information about which key exchange protocols are currently available and how they work the group conducted a literature review wherein the members searched for information by themselves. They also received some introductory papers and books from the course supervisors. The course

supervisor also gave a presentation on some of the concepts and their opinion on the different key exchange protocols. Besides choosing a protocol the group also had to decide on a topology, this was done during meetings with the supervisor. The group also studied documentation for different python libraries and for Mininet.

To find more papers and books Google search, Google scholar and Linköping University's own search tool was used. The criteria for the source were that it was either a published book or that the paper was linked to a scientific institution. If someone in the group found a good source to use and read they shared it with the rest of the group. The group then discussed whether or not source was relevant for the project.

### B. Experimentation

To implement multi-party key exchange the group first had to conduct experiments to get an understanding of how the current implementation worked.

The experiments were conducted on a virtual machine running Ubuntu 22.04.1 [21]. They mostly consisted of examining log-files while running Mininet, with the HIPLS configuration. The group also modified the source code of the HIPLS and the setup-script to investigate what changes occurred in the log output. Finally the group also sent different commands in the Mininet command line interface, for example they pinged different routers or looked at the interfaces of the routers and hosts.

A small mock version of the BD protocol was also implemented. This small mock version served to demonstrate the functionality of BD to the supervisor and other students taking the course, as well as a proof of concept for the group. It also gave the group a better understanding of how the final version of the protocol should look and function.

To calculate the time for the BEX in the current implementation two more routers were added so that there were an equal number of nodes in the proposed and current implementation. To get the correct timings for the BEX the time span that was counted was from the first I1 packet sent from the spoke/router to when they received the R2. For the proposed implementation only, the spoke was timed as they initialize and are the end of the BEX. The timings were timed on Windows laptop running VirtualBox with Ubuntu 24.04 [22]. To make sure the performance was the same for all runs they were done while the laptop was not charging and in a normal performance mode.

### C. Implementation

As stated earlier, the existing code was only functional for key exchange between two parties and it used a topology similar to the one in fig. 3. This code therefore needed to be changed and improved to allow for more parties. The first step was to review the existing implementation. From the review the group got a better understanding of how the code worked and where to implement the changes to allow for the multi-party key exchange.

The group broke the issue down into different parts. The first part was to rewrite the code for the cryptology and key exchange. The second part was to rewrite the topology setup to conform to the hub-spoke topology. The third part was to rewrite the BEX to implement the newly written cryptology into the new topology. The first and second tasks were done in parallel.

## IV. Results

This section describes the result of the project, which includes the literature review, the implementation of the multi-party key exchange and the technical documentation.

### A. Literature review

From the information gathered during the literature review the group decided to use BD with an elliptic curve, together with a hub-spoke topology. The literature review also served as the foundation for the background and theory used in the report.

### B. Implementation

The results of the implementation is a working version of HIPv2. It uses the Burmester-Desmedt algorithm for the key establishment, and a reworked topology with hubs and spokes. It also uses ESP with NULL-encryption to avoid double encryption of packets.

*1) Topology:* To conform to a hub-spoke topology with three hubs and three spokes, the Mininet setup-script was altered. This involved updating the IP addresses and names of existing nodes, as well as adding new nodes and reconfiguring the switches to establish new connections.

Since the base HIPLS implementation uses one directory per node the folder structure was also changed to follow the altered setup script. This means the router directories were replaced with three spoke- and three hub-directories. Each directory consisting of the same content as the routers from before. The exception being that the files in the config-directory were changed to conform to the new topology. This included the following files:

**Mesh:** Which describes the mesh by mapping the HITs of the different nodes to each other. This was changed so every hub was connected to one of the spokes and all the other hubs, while the spokes were only connected to one hub.

**Hosts:** Which maps the IP addresses of each node to their HIT. This was changed so the hubs file included the other hubs and their own spoke and so every spokes file only included only their own hub.

**config.py:** This file includes configurations like interfaces and IPs for the specific node. Here the interface name and IP was changed to correspond to the one specified in the Mininet setup. Later a list of HITS for the connecting spokes were added to the config of the hubs.

**rules:** This file acts like a form of firewall, which allows or disallows different nodes from connecting to

each other. It was modified so the correct nodes can communicate with each other.

**public.pem & private.pem:** Are both keys used to generate HITs and keys. New ones had to be generated for spoke two and three. The other nodes could reuse the HITs and keys from router 1 to 4.

The new keys and HITs were generated using utility files included in the current implementation, these were located in the Util directory of each router. There were also changes made specifically in the hubs BEX-related files and specifically in the spokes BEX-related files, this was to ensure that the BEX was working correctly.

*2) HIP Base exchange:* The final implementation uses the BD with an elliptic curve for key establishment.

To implement BD for HIP, the structure of the HIP packets had to be changed, this will be the same for I1, R1, I2 and R2. Each packet was replaced by a collection of segments, where the original HIP packet, see fig. 4 was the first segment. This first segment was also modified to not include any DH parameters, and instead make use of Burmester-Desmedt for key establishment. There are also added segmentation parameters. These consist of the id for the packet (shared by each of its segments) as well as the id for the individual segment and a flag signifying whether or not this was the last segment (both are zero for the first segment).

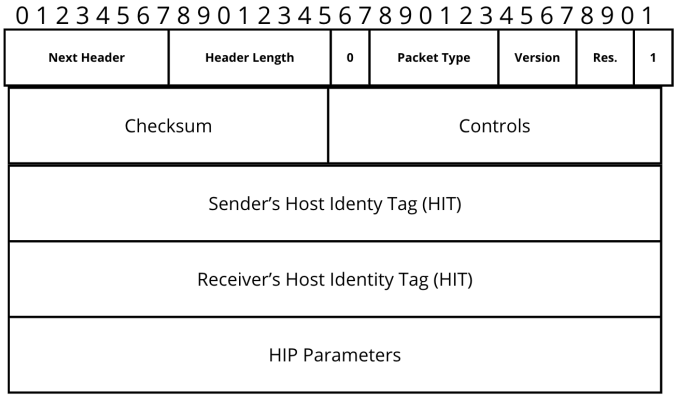


Fig. 4. Current HIP packet

After the first packet follows several identical HIP packets (segments), each containing a HIP header, trailer and two parameters. The first parameter is a segment parameter (same as in the first segment), the second is a BD parameter containing one point on the elliptic curve. The number of packets scales linearly with the number of participants in the BEX. The HIP segments are reassembled by the receiver by gradually adding them into a list. For a visual representation of the parameters in the segments, see fig. 5.

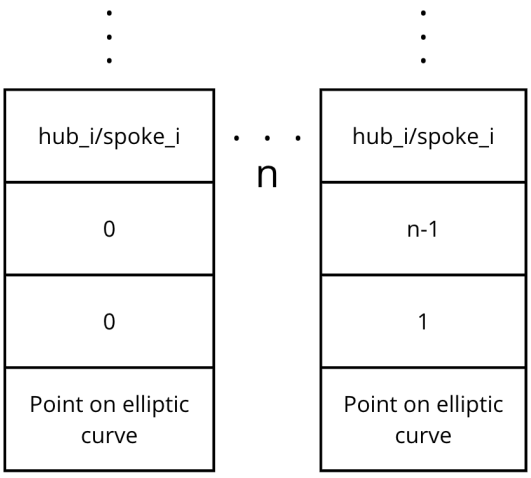


Fig. 5. segments 0 to n

Following is a detailed step by step for the proposed BEX:

1) Spokes send I1 to their respective hub (containing $z_i$ for that spoke)
2) Once one I1 per spoke has been received and processed the hub attempts to initiate the BEX with the other hubs by sending an I1 (containing $z_i$ for spokes belonging to the hub together with hubs $z_i$)
3) The hubs process each others I1 and respond with R1 (containing $z_i$ for spokes belonging to the hub together with the hubs $z_i$)
4) The hubs now have completed lists containing all $z_i$.
5) The hubs respond to each spoke with another R1 (containing all $z_i$)
6) Spokes send I2 to their respective hub (containing their $x_i$)
7) Once each spokes I2 has been received and processed, the hub sends I2 to the other hubs (containing $x_i$ for spokes belonging to hub together with hubs $x_i$)
8) The hubs process each others I2 and respond with R2 (containing $x_i$ for spokes belonging to the hub together with the hubs $x_i$)
9) Now the hubs have a complete lists containing all $x_i$
10) The hubs respond to each spoke with another R1 (containing all $x_i$)
11) Every participant now has the complete x list, the key can be computed and the BEX is complete.

Note that for steps 2-4 and 7-9 for any pair of hubs only one will send I1/I2 and one R1/R2.

For the DH version $n^2 \cdot 2$ packets in total are needed, where $n$ is the number of participants. Comparatively the BD version only requires $s \cdot 4 + h^2 \cdot 2$ where $s$ is the number of spokes and $h$ is the number of hubs. This comparison can be seen in fig. 6, where the blue dots represent the number of packets required for the proposed solution and the red represents the number of packets for the current implementation.

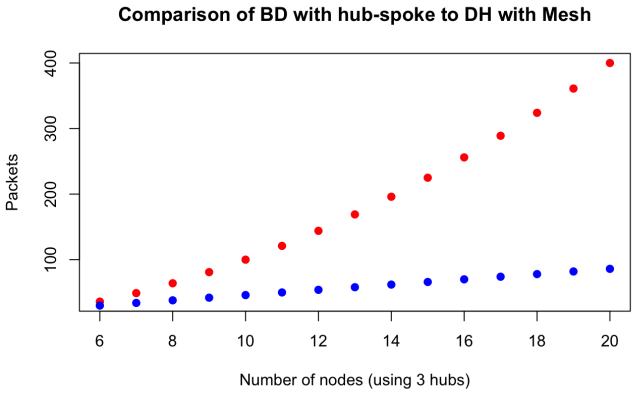


Fig. 6. Comparison of number of packets. BD is blue and DH is red.

From this it is clear that a small number of hubs relative to spokes results in good performance for bd, especially when the number of participants approach to infinity. For a visual overview of the steps mentioned, see fig. 7. This shows the steps between spoke and hub as well as the steps between hub and hub.

*3) Benefits of proposed protocol in the HIP:* The benefits from using HIP with the proposed key exchange protocol can be seen in table II. The key exchange still provides authentication through the use of self-owned public and private keys. It still also provides DDoS protection through the use of a puzzle, as well as resilience to IP spoofing through the use of HITs. Finally multi-party key exchange makes the implementation more efficient since only one shared key has to be generated.

TABLE II
BENEFITS OF THE HIP WITH THE PROPOSED IMPLEMENATION

| Benefit | Explanation |
|---|---|
| Authentication | By providing self-owned public and private keys |
| DDoS protection | By using a puzzle |
| Resilience to IP spoofing | By using HITs |
| Multi-party key exchange | Single shared key is more efficient than one key for each pair |

By comparing the proposed implementation of the HIP BEX to the current one, its clear that the proposed one offers improvements over the current one. This can be seen in table III, where the Attributes of the packets are compared, as well as if the solutions provide trust or not. In the proposed one trust is achieved on the spokes. The number of messages need to complete the BEX stays the same (four). However, it is overall decreased since less nodes are connected and there are therefore less BEX's being completed.

TABLE III
COMPARISON BETWEEN PROPOSED AND CURRENT IMPLEMENTATION

| Parameters | Proposed | Current |
|---|---|---|
| Trust | Yes, on CE (spoke) | No, on provider |
| Number of messages (BEX) | 4 | 4 |
| Number of messages (growth) per node (n), spoke (s) and hubs (h). | $s \cdot 4 + h^2 \cdot 2$ | $n^2$ |
| Data plane | IPsec ESP with null-encryption | IPsec with regular ESP |
| I1 Packet size (s), with number of participants (n) | $s$ | $s \cdot n$ |

The proposed implementation of the HIP BEX also helps with resource efficiency of the overall HIP-based VPLS network by using ESP with null-encryption instead of regular ESP. This achieves about the same result as using AH, since it does not double encrypt the packages which results in a quicker BEX.

Even without AH the proposed implementation is more efficient for a large number of nodes. However, when only six routers are used the current implementation is mostly faster. This can be seen by comparing fig. 8, which is the timing of the current BEX, and fig. 9, which is the timing of the proposed one. The timings of the BEX were measured using Pythons Time library and the times in the graphs show time taken from when the I1 packet was sent to when the R2 packet was received. For the current implementation six routers were used in a mesh network and all of them were timed. For the proposed network three spokes and three hubs was used instead and the time was only measured for each spoke. It was considered unnecessary to measure the timing of the Hubs, since when the key-exchange is complete for the spokes it is also complete for the Hubs.

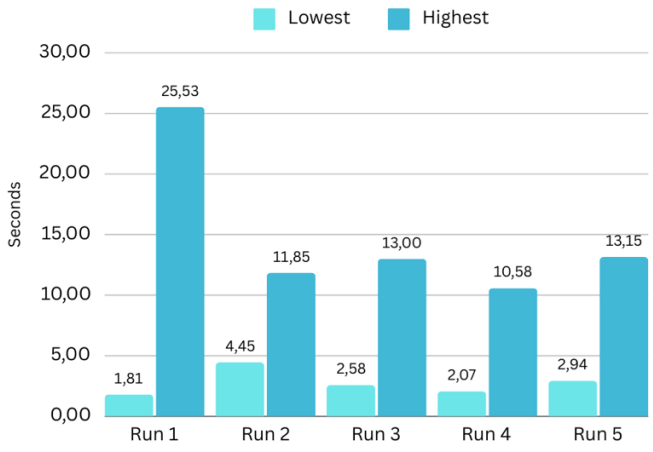


Fig. 8. Timing of current BEX

For the current BEX it's also possible to see major inconsistencies with the timing for different nodes, where the slowest one is more than 20 seconds slower than the fastest in some

runs. Dissimilarly, in the proposed BEX the timings of every node (spoke) is very consistent, and the slowest and fastest times never deviate more than a second. This results in a more dependable and measurable network. The consistency of results also shows that there exist some outlier times in the original implementation that are not present in the proposed implementation.

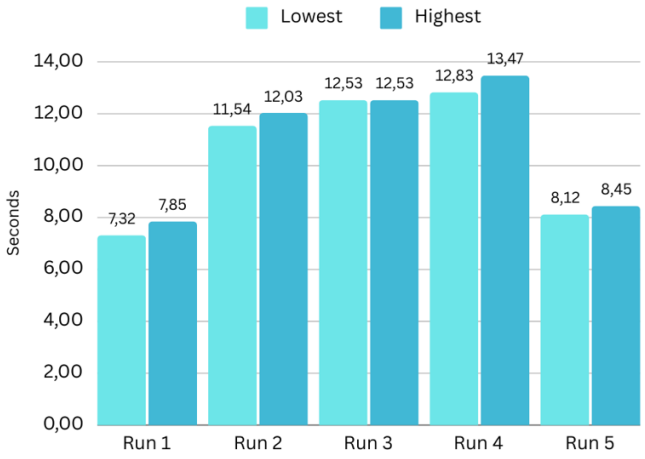


Fig. 9. Timing of proposed BEX

Fig. 10 shows the average time to complete the BEX for both implementations. The time for BEX completion is calculated by measuring R2 received times, from this it's clear that on average the original full mesh implementation is approximately 1.39272 faster compared to the proposed Hub-Spoke implementation.

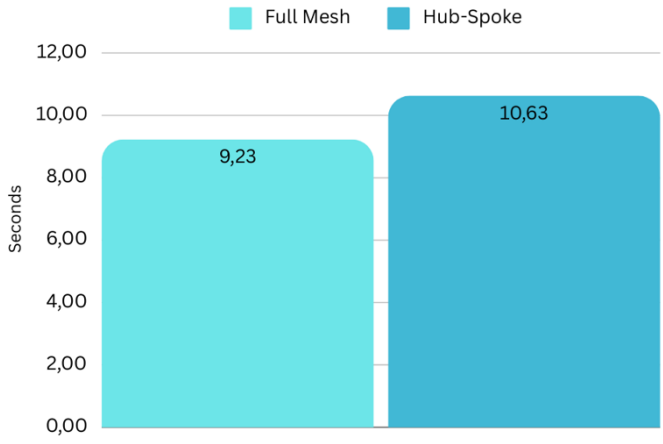


Fig. 10. Average timing of proposed vs. current BEX

### C. Technical documentation

Technical documentation was written to compile information and to provide a clear description of the implemented changes. This documentation was written in the README file of the groups Github repository. The documentation includes how to setup the environment as well as how to run the Mininet network.

## V. Discussion

This chapter discusses the chosen methodologies and the result of the project.

### A. Methodology

This section presents a discussion about the chosen methodologies and how they could have been improved.

*1) Literature review:* Overall the literature review phase of the project went well and gave members a good understanding of the underlying theory for the code implementation. The discussion the group had among itself and with the supervisor

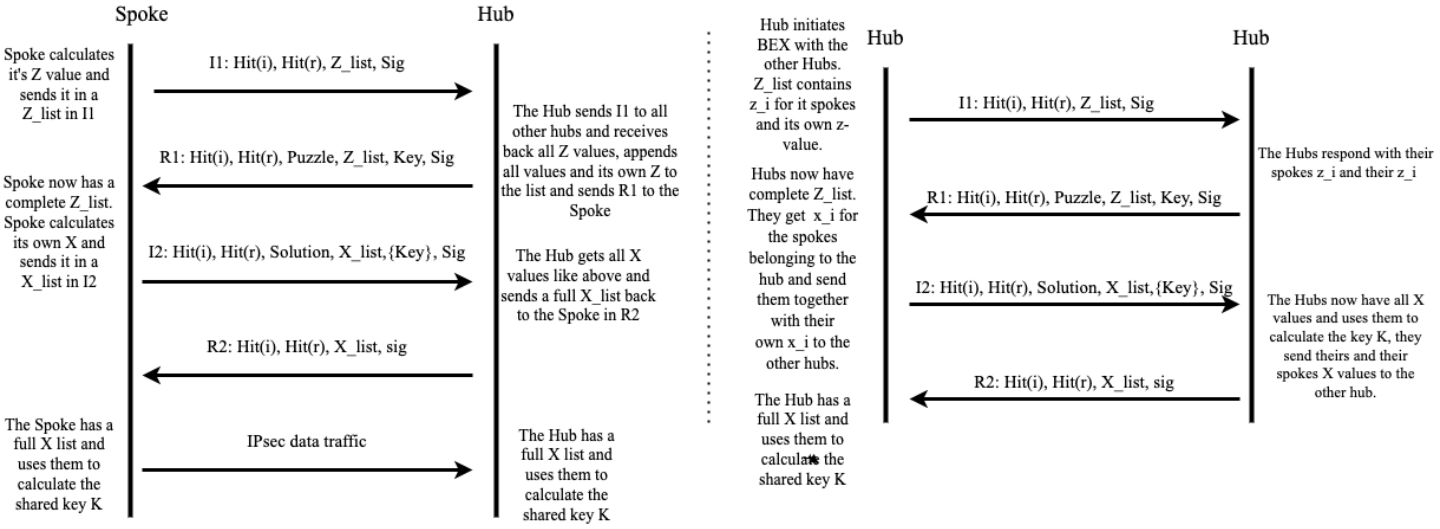


Fig. 7. Spoke to hub and hub to hub BEX with BD

were an important part in information sharing, giving all participants a similar level of knowledge. Most of the sources found with the different search functions, as well as the ones from the supervisor were of high quality and gave relevant information. The exception was the source used to implement elliptic curve, which had incorrect information. The criteria written in the methodology for the sources used ensured a high quality of the sources used.

The literature review was a smaller part of the project which meant that the methodology was not as rigid and fleshed out as it should be if more time could be spent on it. Things to improve would be to have a more systematic criteria for how to search for information and to save specific search words.

*2) Experimentation:* Experimentation played a big role in the groups understanding of the HIP implementation. During the implementation, functionality was regularly tested with simpler setups in order to effectively develop without getting stuck at bigger parts.

When reading the code it is very hard to keep track of the BEX timeline because there are several things going on. When the topology was then increased to more hubs and spokes knowing how the BEX worked was very important in order to troubleshoot issues. Logging the I1, R1, I2 and R2 packets was a good and commonly used method to see progress and troubleshoot issues. At times the group found the previous HIP implementation quite confusing, especially because hlib.py was such a large file. However, understanding its functionality was made easier by running the network and logging information about the state of the BEX.

The experimentation could have begun at an earlier stage to allow all group members to get a better understanding of the current implementation. To allow for this, the literature study could have been done concurrently to the experimentation, which might have resulted in a better understanding of the code. This could however have slowed down the experimentation, since the group would have had less of an understanding of the problems they were solving.

*3) Implementation:* While implementing the code the group could have worked on one task at a time instead of splitting up the problem into smaller tasks. This could have improved the understanding of the code, but at the same time it would have slowed down the process, since less code could be written at any given moment.

Sources for using BD with ECC in Python code were hard to find and often did not have a correct implementation even if they claimed to have it. This made the implementation take more time and it was difficult to find problems with the code, since the sources were trusted to be true.

The group could also have tried to make contact with the author of the original implementation, this could have sped up the process of understanding the code.

### B. Result

This section presents a discussion about the results that were found during the project. This includes discussing the efficiency and scalability, as well as the security of the proposed implementation.

*1) Efficiency and Scalability:* A difference between the original Diffie-Hellman implementation and the proposed Burmester-Desmedt implementation is the use of elliptic curve cryptography (ECC). Therefore the proposed implementation makes use of ECC's inherently smaller key size to increase efficiency of the implementation. The smaller key size itself makes the implementation more efficient compared to the current implementation.

The results in fig. 10 show that the average time until the BEX completes is better than expected with the proposed implementation of BD compared to the current DH implementation. This was surprising as the expectation was that the proposed implementation would be much slower because of the proposed implementations larger overhead. The believed reason for the proposed implementations above expectation performance, is because of the pseudo-threading of the packet handling, removing the wait time that can be seen in the current implementation. This is what we believe causes the long outlier times seen in the current implementation. With this we can also note that the proposed implementation doesn't suffer from outliers or inconsistent timings and is instead consistent and predictable for all spokes. However, the proposed implementation is still slower than the current one for six nodes, but since it's such a small difference for such a small network it should be much better with a larger network.

The reasoning behind the change from Diffie-Hellman to Burmester-Desmedt is the scalability problem that comes with Diffie-Hellman in large networks. The Burmester-Desmedt key exchange effectively reduces the complexity of key sharing from $O(n^2)$ to $O(n)$ reducing the potential network load when $n$ is big. However, to achieve this there is some loss of efficiency of the overall key exchange protocol which can lead to slight performance losses at lower network loads. The final implementation manually sends each $Z$ and $X$ value in separate packets to avoid MTU issues. This can lead to inefficiencies as this might not always be necessary if the entire Z and/or X list can fit within the MTU. It is also slightly counterproductive in that it does not decrease the workload of the client as it needs to manually reassemble these packets correctly when received.

*2) Secp256k1:* The final implementation makes use of the secp256k1 elliptic curve which outside this work is prominently used in Bitcoin's cryptographic framework [23]. The reason for choosing this elliptic curve is the existence of previous Python implementations for conducting point mathematics on it. The use of these tools streamlined the development of the implementation and greatly reduced the time needed. The choice was also motivated by its established security, efficiency and wide use within Bitcoin. However, using secp256k1 comes with limitations. This elliptic curve was chosen for Bitcoin for many reasons that are irrelevant to this projects use case. Because of this it might not offer the best performance or security characteristics compared to other available elliptic curves that can be used for this purpose.

Exploring these characteristics is outside the scope of this project, however exploring alternative elliptic curves could be an area of interest for future study in order to further increase the efficiency of this solution. Curves developed by the National Institute for Standards & Technology (NIST) and other popular curves such as Curve25519 [24] comes to mind. NIST curves, though occasionally criticized for potential weaknesses, remain industry standards and are supported by extensive cryptographic infrastructure.

*3) General Security:* Compared to pair-wise key exchange, with multi-party key exchange the risk of key compromise is the same. However, if the shared key and a private key is compromised it affects all of the nodes instead of only affecting one pair. BD is also not resistant to malicious participant attacks, which involves a malicious participant disrupting the key agreement [25].

Complexity is also minimized by the use of a hub-spoke topology, because since the spokes only have to be connected to one of the hubs there are fewer connections between the nodes. This also results in a decrease in the number of attack surfaces, and it makes a compromised spoke less difficult to isolate. A more efficient use of connections also means it is easier to add more nodes later.

The group choose to use three hubs instead of only using one centralized one, this was to provide fault tolerance since if one hub fails or is compromised the whole network will not go down.

*4) Zero trust:* The implemented key exchange protocol does still enable zero trust. Firstly it does this by establishing a shared secret among all participants while not exposing each participants secret key. Secondly it minimizes complexity by minimizing key management and the number of keys.

HIP itself is also beneficial for zero trust, it is based on decentralized identity where each node has a self-owned credentials (used for its identity) [2]. So these credentials are used to provide secure communication channels, providing confidentiality, removing a requirement for a centralized authentication service. The key exchange will also provide integrity, while the HIP provides authentication, both of these are requirements to establish trust in a zero trust network.

*5) Tables and graphs:* The timings measured in the graphs in section IV-B2 are dependent on the hardware of the machine that was running them. For a more definite result the tests should have been run on multiple machines. There is also a need to test networks with an increased amount of nodes to confirm the hypothesis that the proposed implementation would be faster.

## VI. Conclusion

This section will answer each research question and provide a conclusion to the report.

### *A. How to implement the Burmester-Desmedt key exchange protocol into the current Python HIP implementation.*

To implement the BD key exchange protocol the topology was changed. In this paper a hub-spoke topology was chosen, but other topologies that have two different types of nodes could work as well.

The BEX itself was also modified to allow for the use of BD with an elliptic curve. This required modifying the initiation process to enable the spokes to initiate the BEX. Additionally support for sending a list of points was added. This meant changing the structure of the packets them self and adding manual packet segmentation.

Finally the cryptographic functions had to be rewritten to account for the switch from DH to BD using elliptic curves.

### *B. How does the changes to the Python implementation affect the performance of the Host Identity Protocol.*

As can be seen in table III and fig. 6 the proposed implementation is better for scalability and requires less packets than the current implementation, which is very relevant when the amount of nodes increases. It is especially optimized when the topology contains a lot of spokes, because the spokes are much more efficient than the routers in the current implementation, while the Hubs are a slightly less efficient.

There is more overhead compared to the current implementation, so for a small amount of nodes the BEX is slower. This can be seen in fig. 10. However, even though the proposed BEX is slower for a small amount of nodes, it is surprisingly close to the performance of the current one. The believed reason for the proposed implementations performance, is because of the pseudo-threading of the packet handling, removing the wait time that can be seen in the current implementation. With this change the proposed implementation does not suffer from outliers or inconsistent timings either.

### *C. What differences can be identified between the proposed and current implementation with regards to trust, complexity, attack surface and IPsec mode?*

In regards to trust, this implementation provides trust and authentication through BD and the IPsec null-encryption mode. The complexity of the code was increased with the introduction of elliptic curves and a more complex topology but the practical complexity is reduced. However, it now only requires one shared key between all members instead of unique ones between them. Removing the double encryption by the IPsec ESP mode also further reduces the complexity of the system as it can cause confusion in what keys and encryption is provided where in the network. It is now clearer to the hosts where and what is provided by network and what they need to provide themselves.

The attack surface is decreased in a zero trust environment as the hosts no longer need to trust their spoke/hub with encrypting its data as it is now encrypted through the entire transportation. Using separate hubs for starting the exchange the system is not subject to a single point of failure and can handle system failures better than the previous implementation.

## Acknowledgment

The group thanks Mohammad Borhani for his help and supervision during the project.

## REFERENCES


[1] Strangebit-IO, "hip-vpls," https://github.com/strangebit-io/hip-vpls, [Online; read 16 oct, 2024].

[2] A. Gurtov, *Host Identity Protocol (HIP): Towards the Secure Mobile Internet*. Wiley, 2008.

[3] S. Frankel and S. Krishnan, "IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap," RFC 6071, Feb. 2011. [Online]. Available: https://www.rfc-editor.org/info/rfc6071

[4] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson, "Host Identity Protocol Version 2 (HIPv2)," RFC 7401, April 2015. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7401.html

[5] NIST, "Zero trust cybersecurity: Never trust, always verify," 2020, accessed: 2024-12-04. [Online]. Available: https://www.nist.gov/blogs/taking-measure/zero-trust-cybersecurity-never-trust-always-verify

[6] Y. He, D. Huang, L. Chen, Y. Ni, and X. Ma, "A survey on zero trust architecture: Challenges and future trends," *Wireless Communications and Mobile Computing*, vol. 2022, p. 1–13, Jun. 2022. [Online]. Available: http://dx.doi.org/10.1155/2022/6476274

[7] K. R. Glenn and S. Kent, "The NULL Encryption Algorithm and Its Use With IPsec," RFC 2410, Nov. 1998. [Online]. Available: https://www.rfc-editor.org/info/rfc2410

[8] N. Li, "Research on diffie-hellman key exchange protocol," in *2010 2nd International Conference on Computer Engineering and Technology*, vol. 4, 2010, pp. V4–634–V4–637.

[9] M. Burmester and Y. Desmedt, "A secure and efficient conference key distribution system," in *Advances in Cryptology — EUROCRYPT'94*, A. De Santis, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 275–286.

[10] VMware, Inc. (2024) Elliptic curve cryptography (ecc). Accessed: November 27, 2024. [Online]. Available: https://www.vmware.com/topics/elliptic-curve-cryptography

[11] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, 1987, accessed: November 27, 2024. [Online]. Available: https://link.springer.com/article/10.1023/A:1008354106356

[12] N. B. Taqi and A. B. Zubair, "Design, implement, and evaluate the performance of an ipsec- inspired security framework for hip-vpls environment," Master's thesis, Linköping University, 2024.

[13] IBM, "What is network topology?" https://www.ibm.com/topics/network-topology, [Online; read 26 Nov 2024].

[14] V. S. Solomi, D. Stela, S. D., and Tanu, "Implementation of hub and spoke topology in vpn using eigrp," in *2021 Sixth International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, 2021, pp. 135–142.

[15] N. Matsubayashi, M. Umezawa, Y. Masuda, and H. Nishino, "A cost allocation problem arising in hub–spoke network systems," *European Journal of Operational Research*, vol. 160, no. 3, pp. 821–838, 2005, decision Analysis and Artificial Intelligence. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221703005083

[16] Python, "Python 3.13.0 documentation," https://docs.python.org/3/, [Online; read 14 oct, 2024].

[17] Mininet, "Mininet overview," https://mininet.org/overview/, [Online; read 14 oct, 2024].

[18] Github, "Home," https://github.com/, [Online; read 14 oct, 2024].

[19] "virtualbox.org," https://www.virtualbox.org/, [Accessed 12-12-2024].

[20] "mac.getutm.app," https://mac.getutm.app/, [Accessed 12-12-2024].

[21] "ubuntu.com," https://ubuntu.com/download/desktop, [Accessed 04-12-2024].

[22] "osboxes.org," https://www.osboxes.org/ubuntu/, [Accessed 5-11-2024].

[23] Standards for Efficient Cryptography Group (SECG), "SEC 2: Recommended Elliptic Curve Domain Parameters, Version 2.0," Standards for Efficient Cryptography Group, Tech. Rep., Jun. 2010, accessed: November 27, 2024. [Online]. Available: https://www.secg.org/sec2-v2.pdf

[24] R. M. Naik, S. Sathyanarayana, and T. Sowmya, "Key management using elliptic curve diffie hellman curve 25519," in *2020 Third International Conference on Multimedia Processing, Communication & Information Technology (MPCIT)*, 2020, pp. 33–39.

[25] Y.-M. Tseng, "A robust multi-party key agreement protocol resistant to malicious participants," *The Computer Journal*, vol. 48, no. 4, pp. 480–487, 2005.