

**Cologne University of Applied Sciences
&
Pi-lar GmbH**

Master Thesis

**Symbolic Modeling and Analysis of Security Properties of Neuropil
Protocol**

Author

Shabnaz Khanam
Matr./No :11143655

Supervisor

Prof. Dr.-Ing. Harald Elders-Boll (TH Köln)
Dip.Ing/MBA Schwichtenberg,Stephan(Pi-lar GmbH)

**A thesis submitted in fulfillment of the requirements for the degree
of Master of Science
at the**

**Communication System and Networks Program
Faculty of Information, Media and Electrical Engineering
TH Köln**

Date of Submission
10.10.2023

Master's Thesis

Fakultät für
Informations-, Medien-
und Elektrotechnik

Technology
Arts Sciences
TH Köln

Symbolic Modeling and Analysis of Security Properties of Neuropil Protocol

Name : Shabnaz Khanam

Mat.Num : 11143655

Student ID : shabnaz.khanam@smail.th-koeln.de

Program : Master Communication Systems and Networks

Name 1st examiner : Prof.Dr.-Ing. Harald Elders-Boll (TH Köln)

Name 2nd examiner : Dipl. Ing(FH), MBA. Stephan Schwichtenberg (Pi-lar GmbH)

Deadline: 10.10.2023

Declaration:

I hereby confirm in my honour that this thesis is the result of my independent work. All sources and auxiliary material used in this thesis are cited completely. Place, date, signature:

Place, date, signature

CONFIDENTIALITY CLAUSE

This master thesis contains confidential information of Pi-lar GmbH.

Therefore, this project could only be accessed by the first and second examiner and authorized members of the board of examiner. Any publication, duplication and making copies or disclosure is strictly prohibited.

Access could only be granted to the third parties with written approval by the author and Pi-lar GmbH.

ABSTRACT

Symbolic modeling and verification of the communication protocol are now important to identify the activity of intruders. Here the goal is to security analyze the Neuropil protocol and find its vulnerabilities, which could be advantageous for adversaries to inject into the communication by modifying messages. The challenging part could be creating a simulation environment of real-life communication, verifying the Neuropil Protocol, and discovering the invisible holes during data transmission.

Neuropil is a light-weight protocol that is designed to generate secure communication between applications and systems. It is suitable for IoT devices by using attribute-based encryption. To grow the network, n2n connections focus on TLS. After completing authentication based on the subject, identities establish connections, maintaining application layer security. To prevent unauthorized access, cipher text is provided with all possible security throughout the communication. It is now required to verify whether those security mechanisms are enough for all kinds of protections or not .

We have implemented formal automated verification of the Neuropil protocol using Verifpal. Verifpal is a newly developed automated cryptography verification tool. It verifies the protocol based on message secrecy and authentication. Furthermore, we showed the scheme's performance compared with another related protocol, secret-share redistribution, and session keys.

ACKNOWLEDGEMENT

I would like to thank Almighty God for his mercy on me throughout my life. I am grateful to my parents and my sisters, who are pillars of my life. I would like to remind my father, whom I lost during my Masters program.

I would like to express my gratitude to Mr. Scheichtenberg Stephen for his support, guidance, and contribution throughout this master thesis. I could have undertaken this journey without his trains. He is an ideal example of the captain of the ship. I have learned a lot from him. I am deeply indebted to Pi-lar GmbH for allowing me to be a small part of it.

I am also grateful to Prof. Elders Boll. I have found a motivated professor during my thesis. He answered every queries in detail.

I am especially thankful to my two supervisors. They are always ready to respond whenever I have any confusion.

I would like to thank all my MCSN Professors and supportive colleagues at Pi-lar GmbH .

My appreciation wouldn't be complete without mentioning the following people in my life: my coursemates, all Bangladeshi students in my course of study, and project mates.

LIST OF ABBREVIATION

FACC - Fischer Advanced Composite Components (Austrian Company)

USD - United States Dollar

TLS - Transport Layer Security

HTTPS - HyperText Transfer Protocol Secure

E2E - End to End

E2EE- End-to-End Encryption

MAC- Message Authentication Code

DH - Diffie-Hellman

DHKE- Diffie-Hellman key Exchange

MITM- Man-in-the-middle attack

MQTT- Message Queuing Telemetry Transport

CIA- Confidentiality, Integrity, and Availability

MLSP- Messaging Layer Security Protocol

JWT- JSON Web Token

JSON- JavaScript object notation

UUID- Universally Unique Identifier

TCP - Transmission Control Protocol

TLS- Transport Layer Security

ALS- Application layer security

AEAD- Authenticated Encryption with Associated Data

MIT - Message Intent TOn

DHT- Distributed hash table

BF- Bloom Filter

EK- Ephemeral Key

LIST OF FIGURES

- Fig.1 : Syntan of Equation[33]
- Fig. 2 : The syntax for Queries[33]
- Fig. 3 : Defining queries syntax about freshness[33]
- Fig. 4 : Defining queries syntax about unlinkability[33]
- Fig. 5 : The sample of output
- Fig. 6 : First Step Explanation's sample of tool analyzation
- Fig. 7 : Second Step Explanation's sample of tool analyzation
- Fig. 8 : Analyzation output sample after attack
- Fig. 9 : Analyzation output sample without attack
- Fig. 10: A simple Example of Verifpal
- Fig. 11: Analyzation output sample of the example
- Fig. 12: MQTT general Architecture
- Fig. 13: MLS Architecture
- Fig. 14: The Combination of Concept of Neuropil
- Fig. 15 : Token Structure[45]
- Fig. 16 : The connection between node and identity
- Fig. 17 : General format of Neuropil message
- Fig. 18 : The elements of cipher text
- Fig. 19 : Handshake Message[45]
- Fig. 20 : Identity token with nfp[45]
- Fig. 21 : Node token with identity fingerprint[45]
- Fig 22 : Message intent token[45]
- Fig. 23 : The connection between nodes according to DHT[38]
- Fig. 24 : Through the subject discover the match of peer[38]
- Fig. 25 : The structure of Bloom filter bits
- Fig. 26 : Detect the shortest through timestamp mechanism

Fig. 27 : Different kinds of session ID used

Fig. 28 : Installed software version name

Fig. 30 : Sequence diagram of Handshake messaging

Fig. 31 : Sequence diagram of Join messaging

Fig. 32 : Establish the connection with new nodes

Fig.33 : DHT message transmission

Fig. 34 : Pheromone message transmission sequence flow

Fig. 35 : Sequence diagram of Message intent token transmission

Fig. 36 : Sequence diagram of userspace message

Fig. 37: Sequence diagram of E2EE with multiple recipient

Fig 38 : The model of basic MQTT protocol

Fig. 39: Queries of Handshake message

Fig. 40: The result of Handshake Message

Fig. 41: The queries of Join message

Fig. 42: The Result of join message

Fig. 43: Queries of DHT message

Fig. 44: The result of DHT message 1

Fig. 45: The result of DHT message 2

Fig. 46: The queries of pheromone message

Fig. 47: The result is about confidentiality of mdc_a in Pheromone message

Fig. 48: The result is about confidentiality of h11 in Pheromone message

Fig. 49: The queries of MIT

Fig. 50: The result of MIT part 1

Fig. 51 : The result of MIT part 2

Fig. 52: The queries of User space message

Fig. 53: The result of User space message

Fig. 54: The queries of E2EE message

Fig. 55: The result of E2EE message

Fig. 56: The queries of MQTT

Fig. 57: The result of MQTT protocol

LIST OF TABLE

Table 1: Syntax of Encryption Primitives with explanation

Table2 : Syntax of Signature Primitives with explanation

Table3 : Syntax of Secret Sharing Primitives with explanation

Table 4 : Content of Neuropil Token

Table 5: Defining target field for every messages

Table 6 : Define the session ID

Table 7: The Pheromone message queries with explanation

Table 8: Security property verification by Verifpal

TABLE OF CONTENTS

Content	Page
Master's Thesis	i
Confidentiality Clause.....	ii
Abstract	iii
Acknowledgement.....	iv
List of Abbreviation.....	v
List of Figure.....	vii
List of Table.....	ix
Table of Content.....	x
 Chapter 1.Overview	 01
1.1 Introduction.....	01
1.2.Objectives.....	02
1.3.Processing planing.....	02
1.4.Thesis structure.....	03
 Chapter 2 Cryptographic Tool (Verifpal)	 04
2.1 Preliminaries.....	04
2.2 Different Kinds of Tools.....	04
2.2.1 Proverif.....	04
2.2.2 AVISPA.....	05
2.2.3 Scyther.....	05
2.2.4 Tamarin Prover.....	05
2.3 Design of Verifpal.....	05
2.3.1 The Reason To Choose The Verifpal tool.....	06
2.4 Verifpal Principal.....	06
2.5 Related Work.....	06
2.6 Declaring Syntax.....	07
2.6.1 Basic.....	07

2.6.2 Primitives.....	08
2.6.2.1 Core Primitives.....	08
2.6.2.2 Hashing Primitive	08
2.6.2.3 Encryption Primitives.....	09
2.6.2.4 Signature Primitives.....	10
2.6.2.5 Secret Sharing Primitives.....	10
2.6.2.6 Checked Primitives	11
2.6.3 Equations.....	11
2.6.4 Messages.....	11
2.6.5 Comments.....	12
2.6.6 Queries.....	12
2.6.6.1 Confidentiality Queries	12
2.6.6.2 Authentication Queries	12
2.6.6.3 Freshness Queries.....	12
2.6.6.4 Unlinkability Queries.....	13
2.6.6.5 Equivalence Queries.....	13
2.7 Output.....	14
2.7.1 Outline.....	15
2.8 A Simple and Complete Example.....	17
2.8.1 Result.....	19
2.8.1.1 Understanding Verification Result.....	19
2.9 Summary.....	20
Chapter 3: New Protocol: Neuropil Protocol.....	21
3.1 Inspiration.....	21
3.2 Theoretical Concept.....	22
3.3 Token.....	23
3.3.1 Node Token.....	24
3.3.2 Identity Token.....	25
3.4 Message.....	25

3.4.1 Handshake Message.....	27
3.4.2 Join Message.....	28
3.4.3 Message Intent Token	28
3.5 Connection.....	29
3.5.1 Node-to-Node Connection.....	29
3.5.1.1 DHT.....	29
3.5.1.2 Pheromone Message	29
3.5.1.2.1 messaging concept.....	32
3.5.1.2.2 Bloom filter.....	32
3.5.1.2.3 Timestamp.....	33
3.6 End to End Connection.....	34
3.7 Target field.....	34
3.7.1 Session ID.....	34
3.8 Encryption Method.....	36
3.9 Conclusion.....	36
Chapter 4. Implementation Phase.....	37
4.1 Creating Environment	37
4.1.1 Downloading Verifpal.....	37
4.1.2 Installing.....	37
4.1.3 Updating.....	37
4.1.4 Execution.....	37
4.2 Neuropil Protocol with Verifpal Tool.....	38
4.2.1 Handshake Message Scenario.....	38
4.2.2 Join Message Scenario.....	39
4.2.3 DHT Message Scenario.....	41
4.2.4 Pheromone Message Scenario.....	42
4.2.5 MIT Scenario.....	45
4.2.6 Userspace Message Scenario.....	46
4.2.7 End-to-End Encrypted Message with Multiple Receivers Scenario	47

4.3 Architecture Of MQTT.....	48
Chapter 5. Result and Analysis.....	51
5.1 Analysis The output.....	51
5.1.1 Handshake Message Result.....	51
5.1.2 Join Message Result.....	52
5.1.3 DHT Message Result.....	53
5.1.4 Pheromone Message Result.....	54
5.1.5 MIT Result.....	57
5.1.6 User Space Message Result.....	59
5.1.7 E2EE with Multiple Receivers Result.....	60
5.2 The Result of MQTT.....	61
5.3 Comparison With MQTT.....	63
5.4 Information Security of Neuropil.....	63
5.4.1 Confidentiality.....	63
5.4.2 Integrity	64
5.4.3 Authentication	64
5.4.4 Non Repudiation.....	64
5.4.5 Authorization.....	64
5.4.6 Accounting.....	65
5.4.7 Freshnes	65
5.4.8 Key Mechanism.....	65
5.4.9 Perfect Forward Secrecy.....	65
5.5 Security properties Verified by Verifpal.....	65
Chapter 6. Conclusion.....	67
6.1 Limitation and Future Work.....	67
REFERENCE.....	68
APPENDIX.....	75

1. Overview

1.1 Introduction

Whenever security properties are discussed, cryptography[1] comes automatically. Both are inseparable. Security is ensured in communication when cryptography is properly implemented in data transmission.

The term “cryptography” originated from the Greek words "Kriptos," meaning secret, and "Graphia," meaning writing. [2] So in the full form, the information is written in a secret way so that only a specific person or host can understand it. According to the modern definition, cryptography is the mechanism of securing information and communications through the use of codes so that only those persons for whom the message is purposefully transmitted are able to understand it and process it. [3]

In Cryptography There are so many techniques available that are derived from mathematical concepts. As a result, the theory is used in the calculation of a set of rules implemented in message encoding, but the same rules aren't applicable for decoding. Many algorithms are built for better performance in communication.[3]

Several protocols exist to improve system performance, i.e., the Advanced Encryption Standard (AES) [4,5], Rivest Shamir Adleman (RSA)[6], Diffie-Hellman key (DHK)[61] and the Data Encryption Standard (DES)[7]. Moreover, data is encrypted using a symmetric method or asymmetric method or sometimes considered plaintext transmission. All encryption processing depends on keys and key generation. According to the situation, users prefer the algorithms.

These algorithms have only one purpose to structure a secure connection with high-level security. Through their implementation, messages are supposed to be secured, and they will be accessed by dedicated users. However, cybercrime continues to occur as third parties are capable of tempering the session and controlling access by revealing credentials.

Now, cyber attacks have become very popular and are increasing day by day. Users lose their identity or face manipulation; all are included in cyberattacks. Sometimes the output becomes more costly than others. On May 22, 2021, CNA news website published "One of the biggest US insurance companies reportedly paid hackers a \$40 million ransom after a cyberattack". [8] On May 21 May 2021, another attack was published, titled "VIENNA (Reuters): The head of Austrian aerospace parts maker FACC has been fired after the company was hit by a cyber fraud that cost it 42 million euros (\$47 million)". [9] By one estimate, cybercrime will cost the world economy an increasing trillion USD per year. [10] [11]

This kind of attack can be prevented if a communication achieves all kinds of security properties. To establish a secure communication session, protocols have to perform as a framework with algorithms. Many kinds of protocols are available that are implemented in IoT devices. Data exchanging between devices in an IoT system has become very sensitive and critical, and the security requirements for any IoT-based system are high. Some of these security requirements include authentication and secure communication between the

devices.[12] In this thesis we will introduce our new protocol called Neuropil that is already utilized in IoT by generating stable communication based on attributed access control. Our focus is on analysis of its security properties.

1.2 Objectives

During data transmission, the security properties of the implemented protocol analysis are the center of attention in our thesis. Because whether communication is secured or not depends on its implementation. In the absence of protection mechanisms, data is controlled by attackers, and attack variation happens depending on security properties. For example, eavesdropping attacks can happen if a message is transmitted without encryption, or man-in-the-middle attack is possible if authentication is missing or replay attack is possible in the absence of a timestamp or validation mechanism.

Our main goal is to define the mechanisms of protocol that help achieve security properties or not. We focus on the topics given below:

Confidentiality: According to protocol structure, encoded cipher text is sent out between two users. Our investigation is whether the message maintains its secrecy during transmission or not.

Integrity: During transmission, cipher text can be modified by the attacker or not. If attacker able to, the question will be arised “Is there any algorithm available that can detect the modification ? ”

Authentication: To identify the real user, the authentication mechanism is needed. Our examining part will be an Authentication scheme and verification also.

Encryption : To protect data or resource the mechanisms will be in analyzation part

Key Management: One of important objective part is key management, its design and implementation for message encryption

1.3 Processing Planing

To define the security properties of our protocol, we create a simulation environment of real-life message transmission by using the cryptographic tool Verifpal, where ciphertext design, key management, and authentication code are shown. The message will be transmitted in the presence of active or passive attacks, and the result will be evaluated in accordance with its security mechanisms.

1.4 Thesis Structure

The Thesis is structured as follows:

- ❖ The introduction has started with Chapter 1.
- ❖ Chapter 2 provides an explanation of different kinds of cryptographic tools, and then verifpal tool is chosen by comparing other tools. So its features and the input language's explanation are also part of this chapter. To support this understanding, an example is also added.
- ❖ Chapter 3 is about Neuropil protocol and its basic architecture explanation
- ❖ Chapter 4 inspects the Implementation of Cryptography tool in Neuropil protocol
- ❖ Chapter 5 evaluates The result and Security Analysation
- ❖ Last not the least The completion of Thesis will accomplished by the conclusion of Chapter 6 and recommendation of improvement and future development are also part this chapter

2.Cryptographic Tool (Verifpal)

2.1 Preliminaries

Security protocols aim at securing communications. They are used in various applications: the establishment of secure channels over the Internet, secure messaging, electronic voting, mobile communications, etc. Their design is known to be error-prone, and flaws are difficult to fix once a protocol is largely deployed. Hence, a common practice is to analyze the security of a protocol using formal techniques and in particular, automatic tools. [13] It is important to define system correctness. While a new protocol has been designed, the research group used formal verification to find vulnerabilities, for example: TLS1.3.[14]

2.2 Different Kinds of Tools

There are several kinds of tools that are structured with mathematical or logical models of a system plus primitive security requirements. Through utilizing this method, they identify the necessity of a security protection mechanism in the data log during transmission, which is the main goal of cryptographic tools. [15] Some of the verification tools are ProVerif [16], AVISPA [17], Scyther [18], TAMARIN [19], Athena [20], NRL Protocol Analyser [21] and many more.

2.2.1 Proverif [13]

ProVerif is a tool for automatically analyzing the security of cryptographic protocols. This was developed by Bruno Blanchet. It utilizes a particular language on an augmentation of Pi-calculus. ProVerif can demonstrate security properties, correspondence, and observational proportionality. This capacity helps the data security domain investigate the mystery and confirmation of safety conventions that are allowed from the supplier. ProVerif is an instrument that upholds cryptographic primitives, including encryption and decryption (symmetric and asymmetric), digital signatures, and hash functions. [13]

In ProVerif, the protocol is changed over into a bunch of Horn clauses and the security properties are deciphered through queries on these clauses. In ProVerif, an attacker attempts to arrive at the data from the channels between processes. [13]

Whenever ProVerif confirms a security property for a cryptographic protocol, the outcomes ensure the following, as shown. When the confirmation is valid, the verification isn't connected to the number of sessions of the protocol or the pre-owned message size. It likewise tests every one of the opportunities for an attacker to break the ideal property. [13]

Whenever the evidence fizzles, ProVerif gives an assault follow, which is reproduced from a determination of realities acquired from the clause. However, ProVerif doesn't give a solution that will help overcome the failure. [16]

2.2.2 AVISPA

AVISPA (Automated Validation of Internet Security Protocols and Applications) is a push-button tool for the automated validation of Internet security-sensitive protocols and applications. [22] The High Level Protocol Specification Language (HLPSL) is developed in the framework of the AVISPA project. It is taken as input language, then the model is translated into IF(intermediate format) and the result is analyzed by invoking state-of-the-art back-ends that return attacks (if any) to the user. [23] The whole performance is quite understandable to security experts.

2.2.3 Scyther

Scyther [24] is known as a push-button tool that is used for the verification, falsification, and analysis of security protocols. It is similar to Proverif, which verifies the protocol for an unbound number of sessions and uses role scripts. [25] The tool provides a graphical user interface that complements the command-line and Python scripting interfaces. The command-line and scripting interfaces facilitate the use of Scyther for large-scale protocol verification tests. [25]

2.2.4 Tamarin Prover

Tamarin is another strong cryptographic tool to analyze security protocols. For experts, it is one of the best choices for automated verification. It is written in the Haskell programming language. Its interactive mode is implemented as a web server, serving HTML pages with embedded Javascript. The source code for Tamarin is publicly available on its webpage. In Tamarin's interactive mode, it integrates automated analysis and interactive proof guidance and provides detailed information about the current constraints or counterexample traces. [26]

These all tools are structured with different languages as well as their input syntax and formalism are equally variant. In spite of being suitable for experts, it is harder for non experts to understand.

2.3 Design of Verifpal

During Dr. Nadim Kobeissi's Ph.D. program, he realized that he was missing a method that is used to produce my security assurances, which requires highly specialized knowledge and is accessible to a more general audience of students and engineers. He has designed a Verifpal cryptography tool. That time, 'Verifpal', a software framework for the automated examination of cryptographic protocols, would be structured for fun and its layout remained simple so that anyone could use it. [27]

The purpose of starting with the Verifpal design is about its performance. From what the author thought, it has to function swiftly compared to existing tools so that the engineer or practitioner will require less effort to draw the model of their desired protocol and will be able to obtain a meaningful result immediately. He also maintains simplicity and is fast to assess, so it won't cover all aspects of security features that will be considered as proof of the protocol's capabilities. [28]

Now it is noticeable that the tool is becoming popular and compatible with any kind of cryptographic protocol. Then it provides the explanation through the result. It doesn't mean that the tool is considered a provider of protocol proof or that its contribution makes the protocol being stronger. Its functional process is used as the jurisdiction for tools such as Tamarin. [28]

2.3.1 The Reason To Choose The Verifpal tool

Basically, Verifpal is preferred because of its simplicity. The author's thought matches our motive. Our intention is to reach a large number of people. Even if they aren't professionals, they can easily understand the neuropil protocol feature without any interruption. It is very new and has never been utilized with the Neuropil protocol before. Its analysis portion also makes us more interested because direct queries are identified, whether the message is sent by an authenticated user or an attacker. From the output, viewers can avoid all confusion and select a suitable secure protocol.

2.4 Verifpal Principal

For symbolic verification, we chose the Verifpal cryptographic tool, which verifies the model in the presence of an active attacker. It performs with an unbound session and fresh values. Through queries, it analyzes advanced security properties that define whether the protocol is able to achieve perfect forward secrecy or not. Verifpal is inspired from proverif [29]. The basic principle has been formalized using the Coq theorem prover. [29]

2.5 Related Work

Since the Verifpal tool is a recent invention and at this moment people are doing research on this topic, previous work records are limited. The available relevant works will be reviewed in this section.

In [30], the author uses Verifpal and Proverif cryptographic tools to analyze the security properties of the QUIC Handshake protocol, which secures the transport layer protocol by improving the transport performance of HTTPS traffic. It is capable of faster implementation and evolution of transport mechanisms. [30]

In [31], the author analyzes the E2EE protocol Stick, which is a signal-based protocol suited for social network platforms. This protocol is the first that is able to re-establish encryption sessions on multi-devices that are connected to asynchronous communication. It retains forward secrecy and backward secrecy. It adds a new session concept with several unique features, including multiple pairwise sessions and a refreshing identity key. The verification is accomplished by using Verifpal cryptographic tools. After exploration, the result shows that the protocol achieves post-compromise security in multiple communications, and during re-establishment encryption sessions, it ensures authentication and confidentiality. [31]

In [32], the author describes a lightweight group authentication protocol with a session key agreement. It retains authentication by transmitting an authentication code with a message. In every session, a new session key between an Internet of Things node and the authenticating server is established, eliminating the need for redistributing new shares. Its security properties are verified through the cryptographic tool Verifpal. After analysis, it shows the protocol's better performance in secret-share redistribution and session key derivations. [32]

2.6 Declaring Syntax^[33]

User-friendly The Verifpal tool is divided into three parts of operation. They are attackers, principles of users, and queries.

2.6.1 Basic

Attacker

First The model starts by defining an attacker. That means in the presence of which attacker the message is transmitted. It could be active or passive. An active attacker is able to modify a message and a passive attacker can obtain the message. The attacker's behavior will be explained in the next section. The syntax of the attacker declaration is simple: **attacker [passive]** or **attacker [active]**

Principal

The next operation is about the action of the user. The message is illustrated using the algorithm and sent across the network. In this operation, equation theory is also shown. The principal operation is divided into three parts, which are constants, primitives, and equations.

Constants

As a constant user can use any alphabet (for example: c0, c1, m1, m2, b, gb) for once. It can't be re-assigned similarly users can't declare the same constant. For example, if Alice declares a constant "c", Bob can't declare a constant c again, not even it privately. The syntax with constants is given below with an explanation.

Knows

The principal user has prior knowledge of constants. This prior knowledge considered private and public qualifiers is acceptable to everyone. The syntax is declared.

**principal Alice[
knows private a**

Another qualifier is password, which is used to declare private constants. then the syntax is written" **knows password"** instead of private or public. Attackers can gain the password from ciphertext if it is used directly in encryption primitives. Therefore, it should be transmitted through password hashing primitives.

Generates

Through syntax, it indicates that a principal contains a fresh value, such as a private ephemeral key. It defines values that will be regenerated while the protocol is executed. All other values are derived from these values. The syntax is

**principal Alice[
generates a1**

Leaks

It is used to define that the principal has leaked an existence constant that is known to the attacker. Through this leakage, attackers may gain value.

2.6.2 Primitives

Cryptographic primitives are an essential part of cryptographic protocol. Similarly, Verifpal is designed with different kinds of primitives, such as one-way hash functions, digital signatures, encryption, and so on. The primitives, which are parts of the Verifpal tool, are explained below.

2.6.2.1 Core Primitives

ASSERT(MAC(key, message), MAC(key, message)): unused

Using for checking the equality between two values. It is useful to define MAC equality

CONCAT(a, b...): c

Using for Concatenation, the single value is the combination multiple value or string

SPLIT(CONCAT(a, b)): a, b

It is the opposite of Concatenation. The multiple values are split from a single value. In transmission communication, the sender uses the concat function for multiple values, after receiving, the user gets those multiple values using split function.

2.6.2.2 Hashing Primitives

HASH(a, b...):

Hash function which is secured is similar to BLAKE2s [34]. Attacker can gain the information from hash value.

MAC(key, message): hash

This function uses key and message as input, hash value comes out as output which is used as message authentication code.

HKDF(salt, ikm, info): a, b.....

This function is inspired by the Krawczyk HKDF scheme [35]. It is used to divide multiple keys from a single

secret value. It generates a maximum 5 number of keys.

PW_HASH(a...): x

After declaring password as private qualifier then a security purpose hash function is used which is similar to Scrypt [36] or Argon2 [37]. The output can be used as a private or secret key.

2.6.2.3 Encryption Primitives

Verifpal is structured with encryption primitives which is used to define unauthenticated encryption, and authenticated encryption with associated data.

Encryption Primitives	Description
ENC (key, plaintext): ciphertext	Defining Symmetric Encryption which similar to AES-CBC[62] or to ChaCha20[63]
DEC (key, ENC(key, plaintext)): plaintext	Defining Symmetric Decryption
AEAD_ENC (key, plaintext, ad): ciphertext	Authenticated Encryption with Associated Data(example: AES-CBC or to ChaCha20)
AEAD_DEC (key, AEAD_ENC(key, plaintext, ad), ad): plaintext	Authenticated decryption with associated data
PKE_ENC (G ^{key} , plaintext): ciphertext	Public-key Encryption.
PKE_DEC (key, PKE_ENC(G ^{key} , plaintext)): plaintext	Public-key Decryption

Table 1: Syntax of Encryption Primitives with explanation

2.6.2.4 Signature Primitives

Declare Syntax	Explanation
SIGN (key, message): signature	Used for creating Signature(Classic signature primitive)
SIGNVERIF (G^{key} , message, SIGN (key, message)): message	For signature verification
RINGSIGN (key_a, $G^{\text{key_b}}$, $G^{\text{key_c}}$, message): signature	Creating a Ring signature that is signed by one party or group and defines all other members.
RINGSIGNVERIF (G^{a} , G^{b} , G^{c} , m, RINGSIGN (a, G^{b} , G^{c} , m)): m	For verification by using members of group's public key
BLIND (k, m): m	Define Blinding Message primitive where sender use blinding factor k to blind message And sent to the signer to generate a signature using the blinding message.
UNBLIND (k, m, SIGN (a, BLIND (k, m))): SIGN (a, m)	After getting signature from signer, sender convert it into regular signature by using unblinding message

Table 2 : Syntax of Signature Primitives with explanation

2.6.2.5 Secret Sharing Primitives

Declare Syntax	Explanation
SHAMIR_SPLIT (k): s1, s2, s3	Used for splitting a key into three shares
SHAMIR_JOIN (sa, sb): k	Two element of the set(s1,s2,s3) is required to obtain K

Table 3 : Syntax of Secret Sharing Primitives with explanation

2.6.2.6 Checked Primitives

For checking primitives `ASSERT`, `AEAD_DEC`, `SIGNVERIF` and `RINGSIGNVERIF` are used to define logically the input's accuracy. Noted part is added with a question mark(?) after all checked primitives.

`SIGNVERIF(k, m, sgn)?`

Through the `SIGNVERIF`, Verifpal verifies the signature by providing a message and public key. `AEAD_DEC` is for authentication decryption. `ASSERT` is used for checking whether two inputs are equal or not. The syntax is written :

`AEAD_DEC(ss_b, e1, gb)?`
`ASSERT(ha, HASH(a)) ?`

During model execution under a passive attacker, the verification procedure will stop if a mismatch happens in checking Primitives. However, under active attackers, the procedure will continuously execute with all inputs. The processing keeps going multiple times and other input might be affected. The result is not reliable because of attackers.

2.6.3 Equations

The mathematical expression is also used in Verifpal to generate a public key or secret key. For example: public key G^a is obtained from the ephemeral value "a".

Example Equations

```
principal Server[
  generates x
  generates y
  gx = G^x
  gy = G^y
  gxy = gx^y
  gyx = gy^x
]
```

Fig 1: Syntan of Equation[33]

2.6.4 Messages

Generally, transmitted messages are verified by key. If the key is sent in plaintext, the attacker will replace it and modify the message. So Verifpal uses the guard constant, which ensures message integrity. Therefore, the attacker can read it, not tempt it. It is used against modification by active attackers. The following chapter will go into greater detail.

2.6.5 Comments

The comment is important for a description for indexing or making a note. For a comment, Verifpal uses a double backslash. “//”

2.6.6 Queries

The last **query defines** that the user asks Verifpal to show the security properties for a specific message. Then the tool starts to analyze the model and show that it is lacking.

2.6.6.1 Confidentiality Queries

The first security property is confidentiality. Verifpal uses this query to define message secrecy. It requires a specific message to analyze. The output shows that an attacker can obtain the data if it loses confidentiality.

Example: Queries

```
queries[
  confidentiality? e1
  confidentiality? m1
  authentication? Bob → Alice: e1
  equivalence? ss_a, ss_b
]
```

Fig 2 : The syntax for Queries[33]

2.6.6.2 Authentication Queries

Another basic quality is authentication. When authentication queries are used, the direction must be shown (in Fig. 2). If message encryption is unbreakable, then Bob can decrypt the message. If the message lacks security properties, the attacker will insert the communication and transmit a new message to Bob, pretending to be Alice.

2.6.6.3 Freshness Queries

Freshness is useful against replay attacks. So Verifpal checks the input values, whether they are fresh or not. Attackers can gain access to communication if messages contain non-fresh value.

Example Freshness Query

```

attacker[active]
principal Alice[
  knows private a
  generates b
  ha = HASH(a)
  hb = HASH(b)
]
Alice → Bob: ha, hb
principal Bob[
  knows private a
  _ = ASSERT(ha, HASH(a))
]
queries[
  freshness? ha
  freshness? hb
]

```

Fig 3 : Defining queries syntax about freshness[33]

2.6.6.4 Unlinkability Queries

In Verifpal, use this query based on the definition that “for two observed values, the adversary cannot distinguish between a protocol execution in which they belong to the same principal and a protocol execution in which they belong to two different principals.”

For example, a and b are the outputs of the same input using the HKDF operation. Verifpal verifies that an attacker is able to reform the same primitive and detect the output. If he can't, the output is treated as unlinkable.

```

h1, h2, h3 = HKDF(a, b, nil)
h4, h5, h6 = HKDF(c, c, nil)
h7, h8, h9 = HKDF(a, c, d)
]
queries[
  unlinkability? h1, h2, h3
  unlinkability? h4, h5, h6
  unlinkability? h7, h8, h9
]

```

Fig 4: Defining queries syntax about unlinkability[33]

2.6.6.5 Equivalence Queries

It defines whether the two values are equal or not, as shown in Fig. 2. During communication, the shared secret key is derived from the DH value and applies for encryption and decryption. Through this, it is identified whether different users in the same session are equivalent or not.

2.7 Output

In order to interpret the result, we have to gain knowledge about how the tool analyzes the model. The symbolic model is used for protocol verification by creating a simulation system of data transmission, but it faces difficulties for complex models, which have a large number of user states and value combinations. As a consequence, it stops analyzing while the processing time crosses the reasonable period. To avoid these problems, Verifpal discovers a unique mechanism to define user states. In every stage, the tool analyzes user space and gradually modifies the more and more elements of the principal state's. This operation is processed in multiple stages. In the next stages, only internal primitive values are mutated. Similarly to Proverif, Verifpal operates an unbound stage for analysis because, because of a unique mechanism, it is able to accomplish its performance in a reasonable time, even for complex protocols.

```
shabnaz-dev% verifpal verify rough.vp
Verifpal 0.26.1 - https://verifpal.com
Warning • Verifpal is Beta software.
Verifpal • Parsing model 'rough.vp'...
Verifpal • Verification initiated for 'rough.vp' at 10:22:50 AM.
Info • Attacker is configured as active.
Info • Running at phase 0.
Analysis • Constructed skeleton SIGNVERIF(G^nil, nil, SIGN(nil, nil)) based on SIGNVERIF(G^s, nonce, SIGN(s, nonce)).
Analysis • Initializing Stage 1 mutation map for Server...
Deduction • Output of SIGN(s, nil) obtained by equivalizing with the current resolution of proof. (Analysis 2)
Analysis • Initializing Stage 1 mutation map for Client...
Analysis • Initializing Stage 3 mutation map for Server...
Deduction • Output of SIGN(nil, nil) obtained by reconstructing with nil, nil. (Analysis 10)
Analysis • Initializing Stage 2 mutation map for Server...
Deduction • Output of SIGN(attestation, attestation) obtained by reconstructing with attestation, attestation. (Analysis 13)
Deduction • Output of SIGN(attestation, nil) obtained by reconstructing with attestation, nil. (Analysis 14)
Deduction • Output of SIGN(nil, attestation) obtained by reconstructing with nil, attestation. (Analysis 13)
Deduction • Output of SIGN(nonce, attestation) obtained by reconstructing with nonce, attestation. (Analysis 15)
Deduction • Output of SIGN(nonce, nonce) obtained by reconstructing with nonce, nonce. (Analysis 15)
Deduction • Output of SIGN(attestation, nonce) obtained by reconstructing with attestation, nonce. (Analysis 23)
Deduction • Output of SIGN(nonce, nil) obtained by reconstructing with nonce, nil. (Analysis 23)
Analysis • Initializing Stage 2 mutation map for Client...
Deduction • Output of SIGN(nil, nonce) obtained by reconstructing with nil, nonce. (Analysis 12)
Analysis • Initializing Stage 3 mutation map for Client...
Analysis • Initializing Stage 5 mutation map for Server...
Analysis • Initializing Stage 4 mutation map for Server...
Deduction • Output of SIGN(s, attestation) obtained by equivalizing with the current resolution of proof. (Analysis 282)
Analysis • Initializing Stage 5 mutation map for Client...
Analysis • Initializing Stage 4 mutation map for Client...
Analysis • Initializing Stage 6 mutation map for Server...
Analysis • Initializing Stage 6 mutation map for Client...
Stage 6, Analysis 730...

Verifpal • Verification completed for 'rough.vp' at 10:22:50 AM.
Verifpal • All queries pass.

Verifpal • Thank you for using Verifpal.
shabnaz-dev% █
```

Fig 5: The sample of output

Due to multiple stages, Verifpal becomes faster for analysis because every stage has a fixed goal that is possible to achieve by element mutation. [33]

Stage 1: Stage 1 is for passive attackers. That means the attacker is able to obtain the data or not. From this stage, all elements are gradually mutated and converted into nil for example constant or equation exponents (g^a convert into g^{nil}).

```

shabnaz-dev% verifpal verify rough.vp
Verifpal 0.26.1 - https://verifpal.com
Warning • Verifpal is Beta software.
Verifpal • Parsing model 'rough.vp'...
Verifpal • Verification initiated for 'rough.vp' at 10:22:50 AM.
Info • Attacker is configured as active.
Info • Running at phase 0.
Analysis • Constructed skeleton SIGNVERIF( $G^{\text{nil}}$ , nil, SIGN(nil, nil)) based on SIGNVERIF( $G^s$ , nonce, SIGN(s, nonce)).
Analysis • Initializing Stage 1 mutation map for Server...
Deduction • Output of SIGN(s, nil) obtained by equivalizing with the current resolution of proof. (Analysis 2)
Analysis • Initializing Stage 1 mutation map for Client...

```

Fig 6 : First Step Explanation's sample of tool analysis

Stage 2: In stage 1, the elements of principal are continuously mutated, but primitives aren't, they are pre-defined for every operation. For example when a user uses the hash function in transmitting a message $\text{HASH}(\text{HASH}(x))$. In analysis it won't change into $\text{HASH}(\text{HASH}(\text{HASH}(y)))$ or user transmits encrypted message $\text{ENC}(\text{HASH}(k), G^y)$, it will not mutate to $\text{ENC}(\text{PW_HASH}(k), k)$.

```

Analysis • Initializing Stage 2 mutation map for Server...
Deduction • Output of SIGN(attestation, attestation) obtained by reconstructing with attestation, attestation. (Analysis 13)
Deduction • Output of SIGN(attestation, nil) obtained by reconstructing with attestation, nil. (Analysis 14)
Deduction • Output of SIGN(nil, attestation) obtained by reconstructing with nil, attestation. (Analysis 13)
Deduction • Output of SIGN(nonce, attestation) obtained by reconstructing with nonce, attestation. (Analysis 15)
Deduction • Output of SIGN(nonce, nonce) obtained by reconstructing with nonce, nonce. (Analysis 15)
Deduction • Output of SIGN(attestation, nonce) obtained by reconstructing with attestation, nonce. (Analysis 23)
Deduction • Output of SIGN(nonce, nil) obtained by reconstructing with nonce, nil. (Analysis 23)
Analysis • Initializing Stage 2 mutation map for Client...

```

Fig 7 : Second Step Explanation's sample of tool analysis

Stage 3: In this stage the mutation is happening and all elements are part of primitives which is convert into nil(which is unknown).

Stage 4: Here all elements are already mutated if additional constant and equation exponents exist, they will be replaced with another not just nil.

Stage 4 and

beyond: Addition elements of primitive mutation is continuously performing following (n-3) formula where n defines the current stage, so the ultimate result of mutation is equal to stage 2.

2.7.1 Outline

After analyzing all elements of the principal, the tool provides the answer, which is asked through queries. If the attacker is able to modify the message, it is pointed out through the question shown in Fig. 8.

```

Result • authentication? Server -> Client: proof - When:
  nonce → nil - mutated by Attacker (originally nonce)
  proof → SIGN(s, nil)
  valid → SIGNVERIF(G^s, nonce, SIGN(s, nil))
  attestation → nil - mutated by Attacker (originally attestation)
  signed → SIGN(nil, nonce) - mutated by Attacker (originally SIGN(c, attestation))
  storage → SIGNVERIF(G^c, nil, SIGN(nil, nonce))?

nil is obtained:
  gs → G^nil - mutated by Attacker (originally G^s)
  proof → SIGN(c, attestation) - mutated by Attacker (originally SIGN(s, nonce))
  valid → SIGNVERIF(G^nil, nonce, SIGN(c, attestation))
  storage → nil

nil is obtained:
  gs → G^nil - mutated by Attacker (originally G^s)
  proof → SIGN(nil, nonce) - mutated by Attacker (originally SIGN(s, nonce))
  valid → nil - obtained by Attacker
  storage → nil - obtained by Attacker
proof (SIGN(nil, nonce)), sent by Attacker and not by Server, is successfully used in SIGNVERIF(G^nil, nonce, SIGN(c, attestation))

(Analysis 47)
Analysis • Initializing Stage 5 mutation map for Server...
Analysis • Initializing Stage 4 mutation map for Server...
Deduction • Output of SIGN(s, attestation) obtained by equivalizing with the current resolution of proof. (Analysis 312)
Analysis • Initializing Stage 5 mutation map for Client...
Analysis • Initializing Stage 4 mutation map for Client...
Analysis • Initializing Stage 6 mutation map for Server...
Analysis • Initializing Stage 6 mutation map for Client...
Stage 6, Analysis 860...

Verifpal • Verification completed for 'rough.vp' at 11:26:12 AM.
Verifpal • Summary of failed queries will follow.

Result • authentication? Server -> Client: proof - When:
  nonce → nil - mutated by Attacker (originally nonce)
  proof → SIGN(s, nil)
  valid → SIGNVERIF(G^s, nonce, SIGN(s, nil))
  attestation → nil - mutated by Attacker (originally attestation)
  signed → SIGN(nil, nonce) - mutated by Attacker (originally SIGN(c, attestation))
  storage → SIGNVERIF(G^c, nil, SIGN(nil, nonce))?

nil is obtained:
  gs → G^nil - mutated by Attacker (originally G^s)
  proof → SIGN(c, attestation) - mutated by Attacker (originally SIGN(s, nonce))
  valid → SIGNVERIF(G^nil, nonce, SIGN(c, attestation))
  storage → nil

nil is obtained:
  gs → G^nil - mutated by Attacker (originally G^s)
  proof → SIGN(nil, nonce) - mutated by Attacker (originally SIGN(s, nonce))
  valid → nil - obtained by Attacker
  storage → nil - obtained by Attacker

```

Fig 8: Analysation output sample after attack

But if the attacker doesn't provide an effect on the message or even can't obtain the message, it shows "all queries pass," as shown in Fig.

```

shabnaz-dev% verifpal verify rough.vp
Verifpal 0.26.1 - https://verifpal.com
Warning • Verifpal is Beta software.
Verifpal • Parsing model 'rough.vp'...
Verifpal • Verification initiated for 'rough.vp' at 11:37:15 AM.
Info • Attacker is configured as active.
Info • Running at phase 0.
Analysis • Constructed skeleton SIGNVERIF(G^nil, nil, SIGN(nil, nil)) based on SIGNVERIF(G^s, nonce, SIGN(s, nonce)).
Analysis • Initializing Stage 1 mutation map for Server...
Deduction • Output of SIGN(s, nil) obtained by equivalizing with the current resolution of proof. (Analysis 2)
Analysis • Initializing Stage 1 mutation map for Client...
Analysis • Initializing Stage 3 mutation map for Server...
Analysis • Initializing Stage 2 mutation map for Server...
Deduction • Output of SIGN(attestation, attestation) obtained by reconstructing with attestation, attestation. (Analysis 16)
Deduction • Output of SIGN(nonce, nonce) obtained by reconstructing with nonce, nonce. (Analysis 16)
Deduction • Output of SIGN(nil, nonce) obtained by reconstructing with nil, nonce. (Analysis 17)
Deduction • Output of SIGN(attestation, nil) obtained by reconstructing with attestation, nil. (Analysis 18)
Deduction • Output of SIGN(nil, nil) obtained by reconstructing with nil, nil. (Analysis 21)
Deduction • Output of SIGN(attestation, nonce) obtained by reconstructing with attestation, nonce. (Analysis 22)
Deduction • Output of SIGN(nonce, nil) obtained by reconstructing with nonce, nil. (Analysis 25)
Deduction • Output of SIGN(nil, attestation) obtained by reconstructing with nil, attestation. (Analysis 29)
Deduction • Output of SIGN(nonce, attestation) obtained by reconstructing with nonce, attestation. (Analysis 18)
Analysis • Initializing Stage 3 mutation map for Client...
Analysis • Initializing Stage 2 mutation map for Client...
Analysis • Initializing Stage 5 mutation map for Server...
Analysis • Initializing Stage 4 mutation map for Server...
Deduction • Output of SIGN(s, attestation) obtained by equivalizing with the current resolution of proof. (Analysis 305)
Analysis • Initializing Stage 5 mutation map for Client...
Analysis • Initializing Stage 4 mutation map for Client...
Analysis • Initializing Stage 6 mutation map for Server...
Analysis • Initializing Stage 6 mutation map for Client...
Stage 6, Analysis 730...

Verifpal • Verification completed for 'rough.vp' at 11:37:15 AM.
Verifpal • All queries pass.

Verifpal • Thank you for using Verifpal.
shabnaz-dev%

```

Fig 9 : Analysation output sample without attack

2.8 A Simple and Complete Example

Here we will show a complete example to define how Verifpal verifies the model that uses encryption primitives. It provides the protection of messages against attackers. In the scenario, we use two principals, Alice and Bob. They transmit data using a secret key. The secret key is derived from key pairs, but users have to send out their public key first. It is transmitted as plaintext.

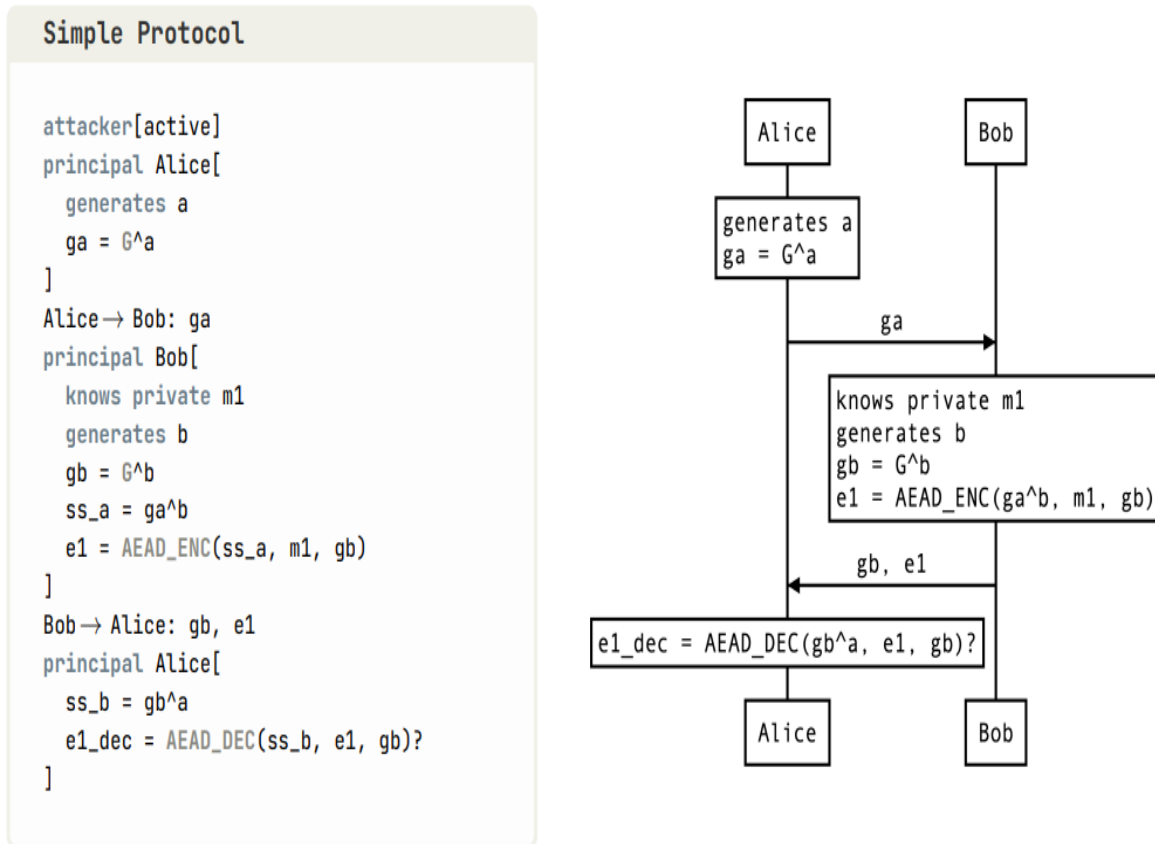


Fig 10: A simple Example of Verifpal

First, Alice sends her public key. Bob generates a secret key from her public key and her own private key, encrypts the message with this key, and then transmits it to Alice with his public key. After having encrypted cipher text with the sender's public key, she derives the secret key and decrypts the message. The user asks through the queries (shown in Fig. 2):

confidentiality? e1
confidentiality? m1
authentication? Bob → Alice: e1
equivalence? ss_a, ss_b

2.8.1 Result:

```

Result • confidentiality? e1 - When:
  ss a ~ G^a^b
  e1 ~ AEAD_ENC(G^a^b, m1, G^b) ~ obtained by Attacker
  ss b ~ G^b^a
  e1_dec ~ AEAD_DEC(G^b^a, AEAD_ENC(G^a^b, m1, G^b), G^b)?
  e1 (AEAD_ENC(G^a^b, m1, G^b)) is obtained by Attacker.

Result • confidentiality? m1 - When:
  gb ~ G^nil ~ mutated by Attacker (originally G^b)
  ss a ~ G^a^b
  e1 ~ AEAD_ENC(G^a^b, m1, G^b)
  ss b ~ G^nil^a
  e1_dec ~ AEAD_DEC(G^nil^a, AEAD_ENC(G^a^b, m1, G^b), G^nil)?

In another session:
  ga ~ G^nil ~ mutated by Attacker (originally G^a)
  ss a ~ G^nil^b
  e1 ~ AEAD_ENC(G^nil^b, m1, G^b)
  ss b ~ G^b^a
  e1_dec ~ AEAD_DEC(G^b^a, AEAD_ENC(G^nil^b, m1, G^b), G^b)?
  m1 (m1) is obtained by Attacker.

Result • authentication? Bob -> Alice: e1 - When:
  gb ~ G^nil ~ mutated by Attacker (originally G^b)
  ss a ~ G^a^b
  e1 ~ AEAD_ENC(G^a^b, m1, G^b)
  ss b ~ G^nil^a
  e1_dec ~ AEAD_DEC(G^nil^a, AEAD_ENC(G^a^b, m1, G^b), G^nil)?

m1 is obtained:
  gb ~ G^nil ~ mutated by Attacker (originally G^b)
  ss a ~ G^a^b
  e1 ~ AEAD_ENC(G^nil^a, m1, G^nil) ~ mutated by Attacker (originally AEAD_ENC(ss_a, m1, gb))
  ss b ~ G^nil^a
  e1_dec ~ m1 ~ obtained by Attacker
  e1 (AEAD_ENC(G^nil^a, m1, G^nil)), sent by Attacker and not by Bob, is successfully used in AEAD_DEC(G^nil^a, AEAD_ENC(G^nil^a, m1, G^nil), G^nil)? within Alice's state.

Result • equivalence? ss_a, ss_b - When:
  gb ~ G^nil ~ mutated by Attacker (originally G^b)
  ss a ~ G^a^b
  e1 ~ AEAD_ENC(G^a^b, m1, G^b)
  ss b ~ G^nil^a
  e1_dec ~ AEAD_DEC(G^nil^a, AEAD_ENC(G^a^b, m1, G^b), G^nil)?
  G^a^b, G^nil^a are not equivalent.

Verifpal • Thank you for using Verifpal.

```

Fig 11 : Analyzation output sample of the example

2.8.1.1 Understanding Verification Result

In Fig. 11, it shows the output of the result where a simple model is presented with encryption and a message is verified by using checking primitives (AEAD_DEC). Now we will explain the result:

Confidentiality ?(e1) It means attacker able to decrypt the message or not .Here attacker only obtained the message

Confidentiality ?(m1) This question for attacker,he obtain the ciphertext but does he able to gain the internal private information? The result shows that he already achieve the element which message contains

Authentication? Bob→ Alice: e1	The result shows the attacker modified the message, like MITM Attack and he forward the message. So the message is sent by the attacker instead of Bob.
Equivalence? ss_a, ss_b	The generated secret keys of both sides (Alice and Bob) are equal or not. It indicates that they aren't. Because the MITM attacker changed message without having knowledge Alice is communicating with attacker. So the secret keys of Alice and the attacker will be equal.

2.9 Summary

This chapter has been intended to narrate Verifpal's structure, processes, and algorithms. It starts with installation and concludes with an explanation of an existing example. In between them, declaring syntax is placed along with the description and purpose. The uniqueness of Verifpal is its adaptability to any protocol. Its implementation is simple. So students studying cryptography can practice the algorithms with this tool.

3. New Protocol: Neuropil Protocol

Through the Verifpal tool, we will analyze the neuropil protocol, which was newly invented at Pi-Lar GmbH. It is a lightweight open-source solution that is implemented in IoT devices to establish stable communication with two-layer encryption. It communicates based on attribute access control and builds symmetric end-to-end communication between authenticated identities. Its uniqueness is about sustainability of data quality, data transparency, and data sovereignty during transmission. [38]

3.1 Inspiration

From the beginning of the creation of Neuropil, the main motive has been to design a protocol that would be user-friendly and cost-effective while maximizing availability and increasing reliability. The author observed the protocol's weight, which is inspired by the Message Queuing Telemetry Transport protocol.[39]

The specialty of **MQTT** [39] is the protocol's lightness and low cost. It creates sessions depending on the public topics or subjects similar to the Neuropil protocol. The publisher creates the topics and sends them to the broker. Clients who are interested subscribe to the topic and send it to the broker again. The credential information of the client including username and password is transmitted as plaintext which is considered as an authentication scheme. That is the vulnerability of this protocol. Due to transmitting a long-term password as plaintext, the message doesn't maintain the standard security (CIA) during communication.

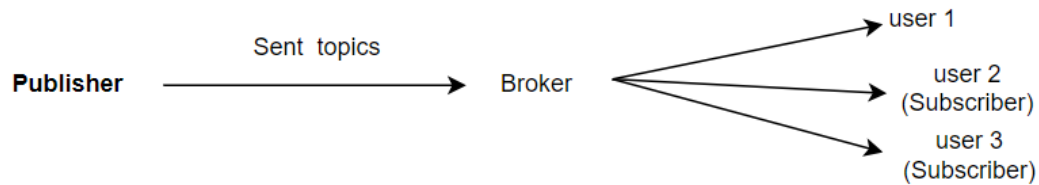


Fig 12: MQTT general Architecture

Though the neuropil idea comes from the MQTT protocol, the authentication steps are not similar. Generating a secure communication process is also different. It doesn't have a central broker like MQTT and uses a key pair instead of a password. Except for the initial message, the rest of them are protected due to the use of transport layer security. It provides all security parameters added to messages during transmission.

For multi-communication, Neuropil is also inspired by **the Messaging Layer Security Protocol (MLSP)** [40], which focuses on different services. For example, the Authentication Service (AS) is for identity authentication by providing credential information and the Delivery Service (DS) is for incoming message redistribution. Generally, identities create the account with the service provider and obtain credentials from the AS. Then they authenticate DS and store key material on it. When a group member needs key material, it asks DS and

generates a new set of keys for message encryption. Other members respond to the message using the updated key material. [41]

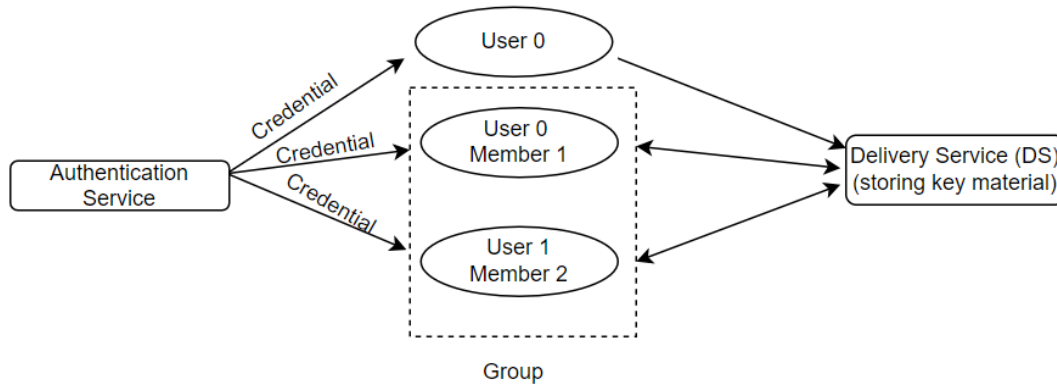


Fig 13: MLSP Architecture

Similarly, Neuropil can generate group communications by creating publishers and subscribers. The publisher generates subjects, other clients will be capable of communicating with him depending on the subject subscription. It maintains end-to-end encryption like MLS, but it doesn't depend on DS for key material storage. It also contains a symmetric key whose validation depends on the subject. But in MLS, group members use the same ephemeral key whenever they communicate.[41]

3.2 Theoretical Concept

Neuropil is a framework that combines other vital concepts to generate a trustworthy interaction between machine to machine. They are data networks, self-sovereign identities, zero trust architectures, and attribute-based access control.

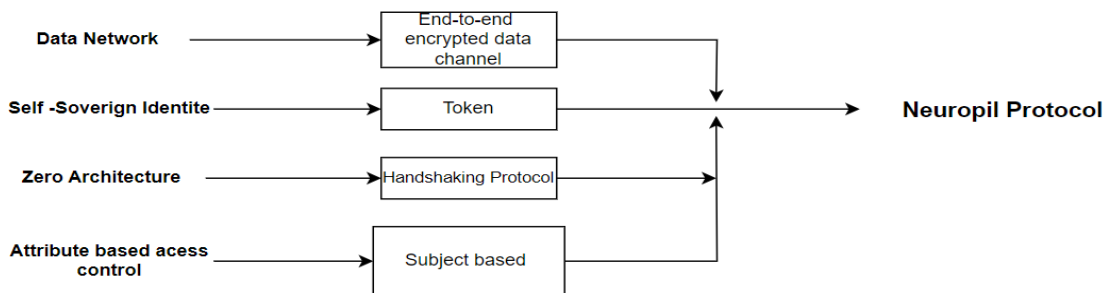


Fig 14 : The Combination of Concept of Neuropil

The concept of the Data network defines the connection between two systems which provides assurance of data accessibility. That means data can be transmitted easily without anyone's interruption. For maintaining reliability, neuropil utilizes end-to-end or point-to-point communication.

The second theory is about Self-Sovereign Identity[42,43] which is another essential section of communication. Its principle is about individuals having the capability to authenticate others. That means one can generate their own digital signature, and another can verify it. They don't need to rely on other authorities. Following this concept, the neuropil maintains its self sign signature.

The main motive of zero-trust architecture[65] is "never trust, always verify". According to this strategy, communication should be verified at every step with its security parameters. Continuous verification helps to minimize the security of all kinds of threats and unauthorized access. Before establishing an encrypted data channel, Neuropil ensures identity authentication by verifying digital signatures.

Attributed-Based Access Control[65] is a method where communication will be established based on the assigned attributes of the subject. When a subject is sent as a request, a policy decision point (PDP) is used to check user attributes. Then an access control policy decides whether the requested access is granted. Neuropil also generates a secret session-based identity-assigned topic, which ensures that only subscribers are able to access the session.

To generate stable communication three questions should be mentioned: what will be sent, where it will be sent, and how it will be sent. Following this question, an answer will be provided in this way: For user identification, Neuropil provides a token through the message, and in the targeted field, it will be indicated that the message will be sent to single or multiple recipients. To define the process, Neuropil maintains multi-layer encryption to establish a secure connection. In the upcoming sections, we will explain every feature in detail.

3.3 Token

The starting of communication in neuropil messages is transmitted containing tokens considered as a record that represents the nodes of a directed acyclic graph. It is structured with predefined fields and extended user-defined fields could be placed according to the demand.[38] In those fields, the information is contained that helps to trace the identity. The structure of Neuropil token is inspired by JSON web tokens (JWT)[44].

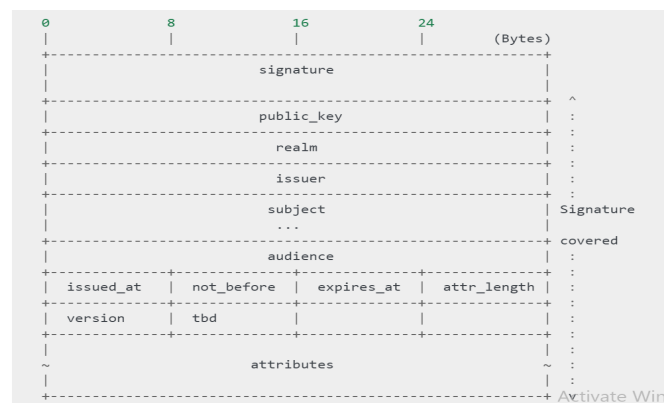


Fig 15 : Token Structure[45]

Generally, the user's identification is verified by username and password, likely MQTT, but the neuropil token contains two signatures for authentication. One is generated with attributes, and the other is without attributes. To ensure entity authenticity as an extended field, UUID and fingerprints are included on the token. The hash values of fingerprints is used as a physical address with 256 bits. It helps to create the routing table so that other nodes can easily generate the links. Other information about what is in the neuropil token is given below:

Token field Name	Definition
Universal unique identifiers (UUID)	Containing unique ID which differs one each other
Char Realm [64 bytes]	Defining the owner
Char Issuer [64 bytes]	The entity who issued this token
Char Subject[255 bytes]	Define what is this token about
Char Audience	Defining the intended audience
Unsigned char public key	Public key of this token
Unsigned char private key	Defining the private key
Issue at/ not before/ expires at/ attr_length	Token validation
Attribute	A variable length, MessagePack encoded map
Unsigned char signature	Define authentication code

Table 4 : Content of Neuropil Token[45]

Neuropil uses three different types of tokens to achieve authentication, authorization, and accounting (AAA) capabilities on top of its overlay network.[38] They are node token,identity token,message intent token. However Node and identity are unique and the rest of the token is a combination of these two tokens. So Message intent Token will be explain in message transmission section 3.4

3.3.1 Node Token

A node token is used for establishing connections over the network. For being activated in the network, tokens exchange their initial information. Every entity has ephemeral key peers: one will appear via token in communication, the other is kept hidden. Through the node fingerprint(NFP), a node is objectified by others. The fingerprint contains the hash value of node token and signature [38]

$$\text{NFP} = \text{Hash}(\text{node token, signature})$$

3.3.2 Identity Token

Node refers to the points that generate the connection in the network to transmit data. Identity defines the user who is connected to the internet through a node shown in Fig. 16. Generally, every identity sits on top of its own separate node. The identity token has long-lived key pairs, which are used for establishing end-to-end communication. Similar to the node token, the identity token has its own fingerprint called the identity fingerprint (IFP).[38]

$$\text{IFP} = \text{Hash}(\text{idtoken}, \text{signature})$$

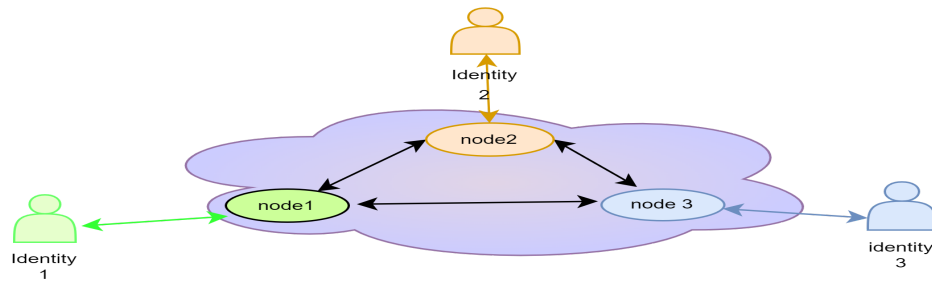


Fig 16: The connection between node and identity

3.4 Message

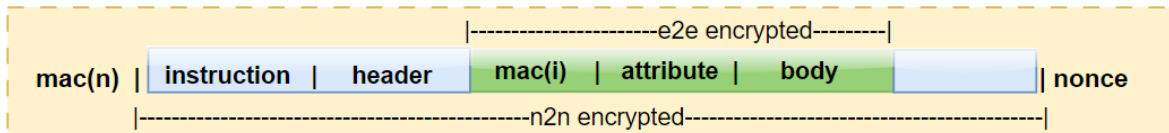


Fig 17 : General format of Neuropil message

The second element of Neuropil is "message" which contains mostly the token to identify the entities. After establishing a dedicated data channel, the message is incorporated with confidential information. To secure the ciphertext Neuropil is designed to maintain stringent criteria and follows traditional encryption protocols by adding complex security parameters. The element is used in the message structure shown in fig 19 with explanation.

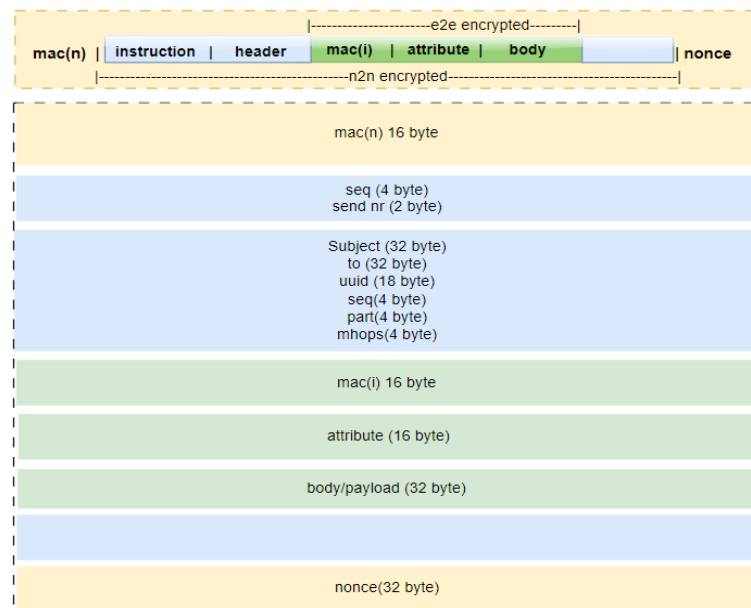


Fig 18: The elements of cipher text

MAC(n) : It is used as authentication code of node

Instruction : It contain the extra information of communication between two nodes such as sequence and sender number

Seq : It denotes delivered message sequence number. For a encrypted message sequence number would be increased during transmission .All mid nodes adds it own sequence number.

Header : Contains the information of sender and recipient

Subj: Defines the hash value of subject

To : It consist with nfp of recipient or session ID

UUID: Defines universally unique identifier number that help to identify the entity with unique ID .Every message carries a fresh uuid that can't be repeated.

Part: The encrypted message is splitted into chunks which define with value.For example message is splitted in four portion.Every portion defines 1/ 4, 2/ 4,3/4, 4/ 4

TTL : Time to live define message validation

Timestamp: elaborately discuss in connection session

mac(i) : similar to mac of node

Attribute: contain extra information of message

Body : define payload

Nonce : contain random value

In Neuropil, seven types of messages are transmitted for secure communication where handshake, join and MIT are for authentication. Before establishing an E2EE connection, messages contain tokens in the payloads. The all-transmitted token is composed of a node and an identity token.

3.4.1 Handshake Message

For communication the first essential element is message or data that carries credential information. It can be used for various purposes such as transmitting confidential information from one person to another.

In digital communication The first message is transmitted called Handshake message. Naturally handshaking is accomplished by the number of steps (TCP-three way) to verify the identity. Initially They exchange cryptographic parameters, protocol and algorithms for stable communication.

Neuropil follows a single step to exchange handshake messages known as handshake tokens that contains basic information about the node. The main purpose is exchanging node's public keys by using node tokens. It transmits as a plaintext containing 1024 bytes. [45]

```

realm      := <empty> | <fingerprint(realm)>
issuer     := <empty> | <fingerprint(issuer)>
subject    := 'urn:np:node:<protocol>:<hostname>:<port>'
audience  := <empty> | <fingerprint(realm)> | <fingerprint(issuer)>
attributes := { <?identity: ifp>, <?user supplied data> }
public_key := <pk(node)>
signature  := <signature of above fields excluding attributes>
signature_ext := <signature of all above fields>

```

Fig.19 : Handshake Message [45]

By the network scanning the hostname and port number are found from clear text. They don't have the certificate authority for authentication and the entities follow self-signed certificates that are signed by the private key. The token contains also a node fingerprint that is formed by node token using a hash algorithm. The hash values aren't only confined in a handshake message, for join message it is equally important. [37]

Certificate = signature algorithm (node token, private key)

3.4.2 Join Message

The join message contains an identity and a node token. So it depicts the node that belongs to a particular identity. The identity token specifies the node, containing its fingerprint in the attribute field. However, a pure identity token doesn't carry a node fingerprint. To verify the handshake message, the join message contains the same node fingerprint, which is part of the issuer field in the handshake token.[45]

```

realm      := <empty> | <fingerprint(realm)>
issuer     := <empty> | <fingerprint(issuer)>
subject    := 'urn:np:id:'<hash(userdata)>
audience  := <empty> | <fingerprint(realm)> | <fingerprint(issuer)>
attributes := { _np.partner_fp: nfp, <?user supplied data> }
public_key := <pk(identity)>
signature  := <signature of above fields excluding attributes>
signature_ext := <signature of all above fields>

```

Fig 20: Identity token with nfp[45]

In the join message, the identity token has nfp to define that identity is connected to the internet through this node. However, nodes can have connections with multiple identities. To ensure connection between node and identity, Neuropil adds the node token one more time in the join message, including the identity fingerprint. Through these fingerprints, it ensures that the message comes from a genuine source.[45]

```

realm      := <empty> | <fingerprint(realm)>
issuer     := <empty> | <fingerprint(issuer)>
subject    := 'urn:np:node:<protocol>:<hostname>:<port>'
audience  := <empty> | <fingerprint(realm)> | <fingerprint(issuer)>
attributes := { <?identity: ifp>, <?user supplied data> }
public_key := <pk(node)>
signature  := <signature of above fields excluding attributes>
signature_ext := <signature of all above fields>

```

Fig 21 : Node token with identity fingerprint[45]

3.4.3 Message Intent Token

MIT is sent after a pheromone message detects a match of peers. As a result, it includes pheromone message containing subject in subject field. It is structured by identity token that references to the node token. That means that the token's authenticity is verified by the identity's public key and signature. Then receivers get assurance about the identity's genuineness from the IFP in the issuer field. To define a virtual address and

routing path, it contains a node fingerprint. The main motive of MIT transmission is to authenticate identity with a pheromone message subject. Before establishing E2E communication, it creates a connection between node and identity that serves as confirmation of interested identities authenticity. Then E2EE data channel is generated automatically with user-space message.

```

realm      := <empty> | <fingerprint(realm)>
issuer     := <ifp>
subject    := 'urn:np:sub:'<hash(subject)>
audience  := <empty> | <fingerprint(realm)> | <fingerprint(issuer)>
attributes := { _np.partner_fp: nfp, <mx properties>, <?user supplied data> }
public_key := <pk(identity)>
signature  := <signature of above fields excluding attributes>
signature_ext := <signature of all above fields>

```

Fig 22: Message intent token[45]

3.5 Connection

From the initial step, the payload of the message contains a token for the purpose of authentication verification. For spreading the connection in the network, the message doesn't carry the tokens anymore. First participants ensure node-to-node(N2N) connection next end-to-end(E2E) communication.

3.5.1 Node-to-Node Connection

For n2n communication, Neuropil uses the Handshake Message, Join Message that we explained earlier. But for growing the network with new nodes or defining the shortest path to reach the destination, DHL messages and pheromone messages are used.

3.5.1.1 DHT

A distributed hash table defines a distributed system that is used for a lookup service by retrieving and storing data. The concept is similar to a distributed hash table, but it doesn't depend on a table for data storage. It transmits data among multiple nodes, and every node stores a portion of the data. When the client needs data, it sends the request to the network, and the request is forwarded to the accurate node, where it responds from data storage. One of the good parts of DHT is maintaining the network by adding and removing nodes. [46]

For communication, Neuropil uses a different routing mechanism on the overlay network, which is implemented according to a Distributed Hash Table (DHT)[51]. When activated in the network, a node contains a virtual address, which is randomly chosen from a 256-bit address space. For having a large address space, a few addresses are used, and the rest are untouched. When the new node is present in the network, the message announces to all nodes that the node is available for communication. According to the DHT table, nodes spread their connections.

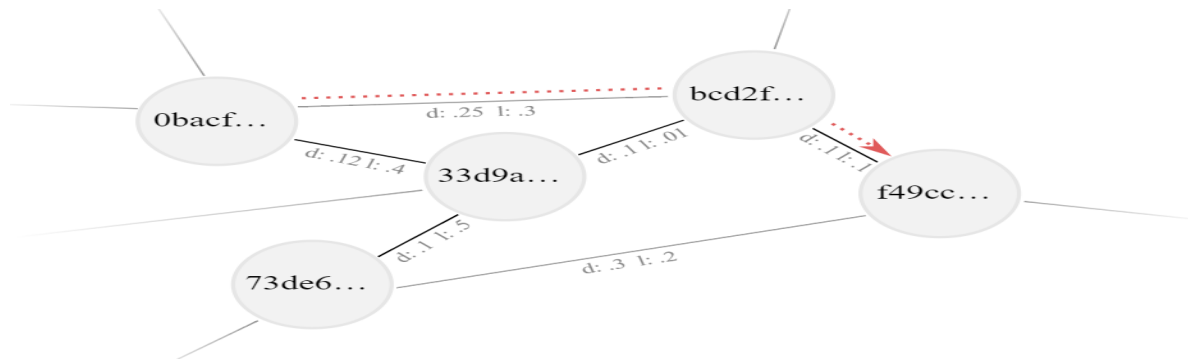


Fig 23: The connection between nodes according to DHT[38]

To support the system The DHT message in the neuropil protocol contains the node's fingerprint and is transmitted through the network for a variety of purposes. Different kinds of DHT messages are used.

Ping Message is for measuring latency between two nodes. It defines the speed of data transmission during communication. It provides information on the performance of communication channels. The message contains the timestamp, which depicts the difference between the message's sending and receiving times. The round-trip time (RTT) is obtained from ping messages that describe the network's responsiveness. The shorter RTT defines faster communication with low latency.

Update The message is about a node entering a network. When a new node enters the network, other nodes get informed through this message. It contains a unique identifier for the hash value of the new node. Through broadcasting, the new node announces its availability and intention to participate.

The **piggy message** is similar to the update message. Its goal is to grow the network by adding new nodes. Through the update message, other nodes are aware of the new node. The network adopts the new identifier in their data-distributed hash table and increases its capacity. then new nodes are able to cooperate in network operation.

Leave messages are the opposite of update messages. If any node wants to disconnect or stop the communication, the leave message is broadcast on the network. It means that the node is removed from the network.

An **acknowledgement message** is used to provide confirmation of a specific message. It contains a universally unique identifier (UUID) of the corresponding message that is being acknowledged.

When a message requires confirmation, the recipient sends an acknowledgement message back to the sender with a specific subject. This acknowledgement message serves as an affirmative answer that the initial message was successfully received.

3.5.1.2 Pheromone Message

Pheromone message is a type of routing mechanism that detects the destination in the shortest way. To explain the pheromone message, two topics will be highlighted. One is the messaging concept, and another is the bloom filter.

3.5.1.2.1 Messaging concept

Pheromone messaging concept is inspired by the Ant Colony Routing algorithm[47], which is based on metaheuristic ant behavior.

By nature, ants come out of the nest to search for food. Initially, they are randomly looking for the source of the food. When they have found multiple paths to reach the food, they provide a route mark during their return. New ants can get the mark if they use the same path and leave their mark while they come back with food. If the path is populated with the mark It indicates many ants use this path to collect food. As a large number of ants left their mark, they may have reached the nest earlier than other groups that collect food from other sources. So it is considered the nearest path from food to a nest. [47]

Following this algorithm, a pheromone message is transmitted in a probabilistic way to reach the destination. User-to-user communication will be established after forwarding pheromone messages. A pheromone message containing user-generated subjects is forwarded from node to node to discover the identities of those who are interested in the same subject. As it is traveling among the cyber security mesh according to the routing table, nodes are able to see the message. Due to the bloom filters' friction with the subject, the neighbor node can't understand and gain specific information. Basically, a pheromone message contains scent, which consists of a bloom filter of the hash value of the subject. If a bloom filter can detect the match with the peer, then an end-to-end encrypted data channel will be established.

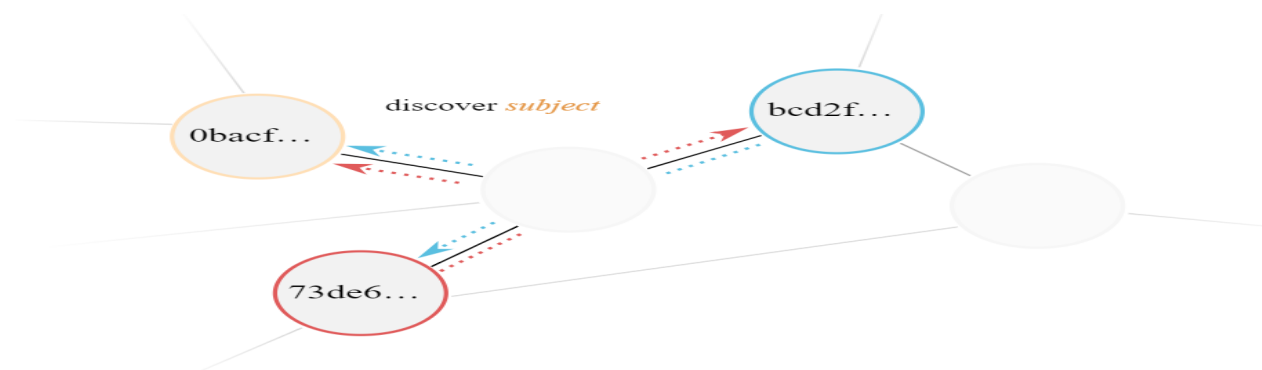


Fig 24: Through the subject discover the match of peer[38]

3.5.1.2.2 Bloom filter

BF[59,60] is constructed using the hash value of the subject. This hash value contains 256 bits. These bits are split into eight units, so every unit contains 32 bits. Every two units generates a block of 64 bits. The total of four blocks is structured as a bloom filter. The three-dimensional calculation is used to define the unit's position in the block. During transmission, intermediate hops achieve a counting/aging bloom filter to define the number of receivers. Sender and recipient both transmit pheromone messages, which are transmitted among authenticated nodes to discover the peer. As both pheromone messages carry the same subject, it is difficult to define whether the message belongs to the sender or recipient. So messages carry extra bits, which makes a difference between two entities' messages.

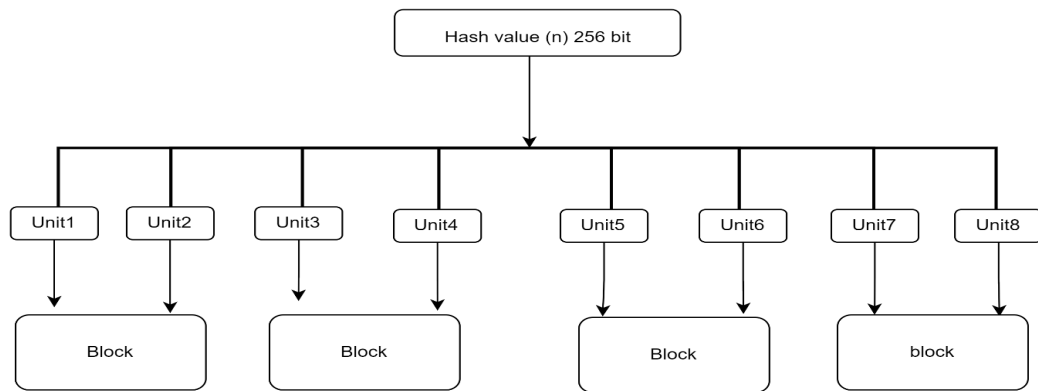


Fig 25:the structure of Bloom filter bits

3.5.1.2.3 Timestamp

During transmission of the bloom filter in a cyber security mesh, the timestamp of the message is decayed sequentially. The timestamp depends on round-trip times. The measurement is observed at two traffic connections. [48]

The message flow, both incoming and outgoing, is used to measure timestamps for defining the shortest path. The neuropil source and recipient send out the same subject. When a message can reach the destination node, its incoming and outgoing BF's time summation defines the shortest path. The recipient accepts BF in all possible ways and calculates their time stamp. The calculation defines the latency of transmission. When the message reaches one node to another with less difference considering low latency, So the path that has low latency addresses the shortest path.

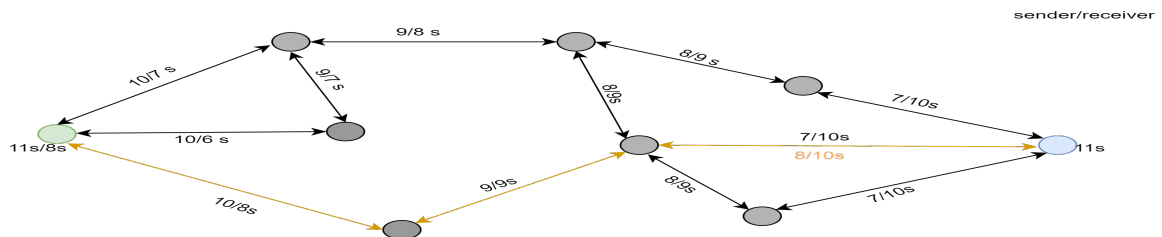


Fig.26 : Detect the shortest through timestamp mechanism

3.6 End To End Connection

After revealing the peers via pheromone message end-to-end communication will be established. Before E2E connection, interested identities have to authenticate each other by exchanging MIT messages. Hereafter the symmetric cipher text is transmitted in the data channel. The dedicated data channel depends on the hash value of the subject and transmitting messages on the data channel maintains transport and application layer security. So Only authorized identity can decrypt the confidential payload which is confined with ALS.

3.7 Target Field

The targeted field is part of the header field of the message, where it provides information on the subject, UUID destination and so on. While a general message typically contains both source and destination information within its header field, neuropil messages carry only the relevant details of the targeted node. To provide a clear overview of the target field, a comprehensive table displaying the relevant information is presented.

Message	Subject	Target
Handshake	Hash("handshake")	-
Join	Hash("Join")	nfp
Update/ping/piggy	Hash("update/ping/piggy")	nfp
Pheromone	Hash("Pheromone")	Hash(subject)
MIT	Hash("MIT")	Hash("subject")
Userspace	Hash("subject")	Session id
Ack("user")	Hash ("ack")	nfp
Ack("node")	Hash("ack")	nfp

Table 5: Defining target field for every messages

3.7.1 Session ID

Once identity is identified by a handshake and join message, the node fingerprint isn't included in the message anymore. After successful authentication through the message intent token, an ephemeral key is automatically transmitted within the userspace message. Following this, the messages are structured to include the session ID as the target in the header field.

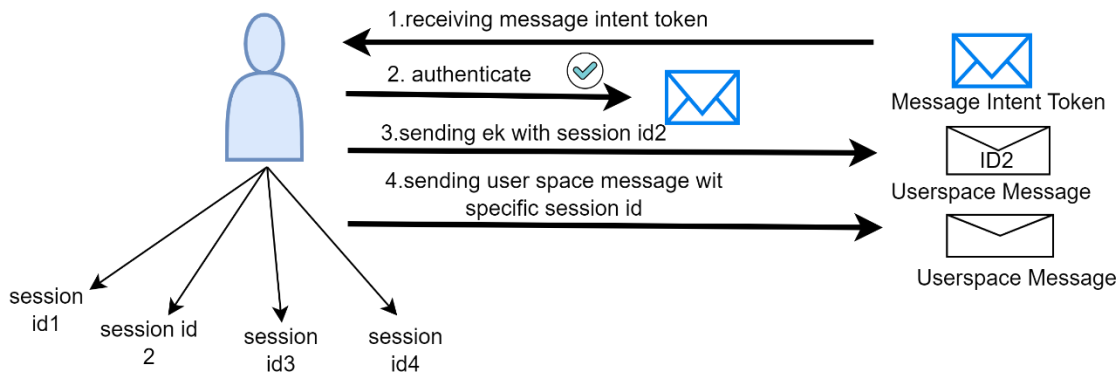


Fig 27 : Different kinds of session ID used

Incorporating the session Id in the header field provides the effectiveness in session management .Each message which is part of an ongoing session carries the corresponding session ID.It defines the number of recipients involved in the session.Neuropil allows four kinds of session ID.

Session ID1:After transmitting EK user choose this session ID1 when there are multiple recipients. This session ID carries the MIT fingerprint of the initiator which is associated with the user generated subject. It provides assurance to the recipients that the initiator is active in the network. After identity authorization the message containing ID1 is designated as public to those recipients who subscribe on the same subject.

Session ID2 :The second session ID2 is level as initial. For maintaining two layer encryption ephemeral key is transmitted automatically with this session ID.To generate this ID which is a unique fingerprint derived from fingerprint of MIT. This calculation is processed with XOR operation.

$$\text{Session ID2} = \text{Initiator MIT(IFP)} \text{ XOR } \text{Recipient MIT(IFP)}$$

Session ID3 : This session ID calculation processing is quite similar with session ID2 , but it is used AND operation. After transmitting the EK, the user sent another user space to identify the recipient. When there is a single recipient , a message is transmitted with this ID. That is a specific reason to consider it as a shared secret session .

$$\text{Session ID3} = \text{Initiator MIT(IFP)} \text{ AND } \text{Recipient MIT(IFP)}$$

Session ID4: The message with session ID4 is labeled as "protected" because its purpose is for a specific recipient among multiple receivers. Only those designated recipients have the ability to access and view the contents of this message.

Session ID	Reason
ID1 (public)	For multiple recipient
ID2 (initial)	Transmitting EK with sender MIT (IFP)
ID3(share secret session)	For single recipient
ID4 (protected)	For specific recipients from multiple receivers

Table 6 : Define the session ID

3.8 Encryption Method

From the initial step to the dedicated data channel, Neuropil focuses on encryption mechanisms. It utilizes asymmetric and symmetric encryption methods. The asymmetric encryption method involves key pairs where data encryption and decryption are accomplished by two different keys. On the other hand, symmetric encryption methods depend on a single key.

In neuropil, user-generated pairs of keys are used in asymmetric signatures. That means the authenticated scheme is encrypted using the user's private key and decrypted by a public key that other nodes achieve from a handshake message.

After the handshake message, the rest of the messages maintain transport layer security and generate symmetric ciphertext using the AEAD ChaCha20-Poly1305 algorithm[63]. Here, the secret key is generated using the Diffie-Helman key algorithm. After authentication, the connection is established between user and user, and then messages are encrypted for one more time using an ephemeral key.

3.9 Conclusion

In this chapter the highlight topic is about the architecture of neuropil protocol communication where the basic two types of token (node token & identity token) are described. Next essential elements are divided according to their purpose. for example : for authentication message contains different kinds of token that are formed from to prime two tokens .

For spreading the network the message doesn't carry the token anymore. To define latency and acknowledgement DHT message has UUID of corresponding message . For detecting the match with a peer, pheromone is transmitted with the subject. Moreover the mechanisms used for transmission are also described.

Next Chapter we will structure the symbolic model of corresponding messages.

4. Implementation Phase

4.1 Creating Environment

Our main goal is to define the security parameters of the neuropil protocol through analysis. To analyze the protocol, we use the Verifpal cryptographic tool for the first time. We already discussed the reason for choosing this tool in Chapter 2. In this chapter, we will explain step-by-step the implementation.

4.1.1 Downloading Verifpal

Verifpal is common and suitable for any operating system. For download, there are two options. One is directly software downloading, which is available on the official website [49], and the latest version v0.27.0, is currently published. Another option is Verifpal Source Repository [50].

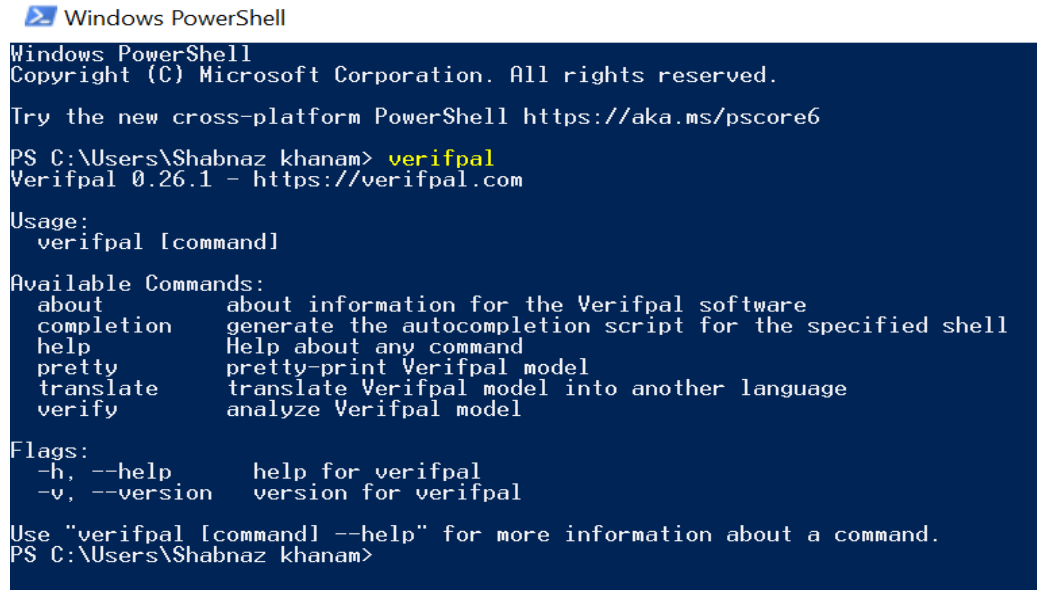
4.1.2 Installing

Verifpal is deployed through the command line; there is no fixed installation process. But its use is easier if it is added to the system command path. [33] First, we use the **Scoop package** manager on the Windows operating system, and then we can install Verifpal software. The installation steps are given below:

1. For installation Scoop, we open Windows PowerShell and provide this command “**iwr -useb get.scoop.sh | iex**”. Sometimes it shows an error such as “The operation has timed out” then “**iex(new-object new.webclient).downloadstring('https://get.scoop.sh')**” will work.
2. Next command is “**scoop status**” for define status
3. Then gradually These commands “**scoop bucket add extras**”,
“**Set-ExecutionPolicy RemoteSigned -scope CurrentUser**”,
and “**scoop install curl**” are given to install verifpal.
4. To connect scoop bucket with verifpal Repository we give “**scoop bucket add verifpal https://source.symbolic.software/verifpal/verifpal.git**”
5. Finally Install process will be completed by giving this command “**scoop install verifpal**”

4.1.3 Updating

The author continuously updated the Verifpal software, releasing new versions that provide better performance with new features and fixing bugs. To know the installed software versions, we run Verifpal only.



```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Shabnaz khanam> verifpal
Verifpal 0.26.1 - https://verifpal.com

Usage:
  verifpal [command]

Available Commands:
  about      about information for the Verifpal software
  completion generate the autocompletion script for the specified shell
  help       Help about any command
  pretty     pretty-print Verifpal model
  translate  translate Verifpal model into another language
  verify     analyze Verifpal model

Flags:
  -h, --help      help for verifpal
  -v, --version   version for verifpal

Use "verifpal [command] --help" for more information about a command.
PS C:\Users\Shabnaz khanam>

```

Fig 28 : Installed software version name

If it is needed to update, the new version has to be downloaded and installed following the same steps as in the installation section.

4.1.4 Execution

After the installation of Verifpal through the dependency software, one script should be executed. To define the model, the file is saved as ".vp". For example, if the file name is "tls", we will save it as "tls.vp". Then we call this file using this command: "verifpal.exe verify .\tls.vp " on Windows PowerShell .

4.2 Neuropil Protocol with Verifpal Tool

The argument is about secured communication with the neuropil protocol. It has been said that the protocol provides authentication at every step. Its final goal is to generate an encrypted, secure connection between identities. To verify the claim, we will implement a verification tool to define data in every step of communication, addressing as scenario and its full symbolic form are available in appendix. In these scenarios, we consider identities as Alice, Bob, Carl, Della, and so on. They are connected through their own nodes called NA, NB, NC, ND etc.

4.2.1 Handshake Message Scenario

The first message is transmitted for handshaking and addressed as "Handshake_Message.vp" in the appendix . The nodes in the network exchange their initial information through a node token. Hence, we introduce two nodes, NA and NB . They exchange their handshake messages to objectify their individuality. Initially, they knew their private keys, na and nb. From these private keys, the public keys gna and gnb are generated. Generally, a handshake token contains a public key, hostname and port number in the subject field, and the rest of the fields are empty. We use those values to define the token and put them together in "data" and "messg"

consecutively for NA and NB using CONCAT algorithms. Applying the SIGN algorithm with data and a private key, nodes generate their signature. They exchange their "data" and "messg" with their signature. When NB receives the digital signature of NA (sign_NA) according to the fig 30, it verifies the value with the public key. For verification it processes sign_NA with NA's public key (gna) and data by implementing SIGNVERIF algorithm. Same way NB's signature sign_NB is verified by using messg and gnb. Through the queries we ask Verifpal to analyze the model.

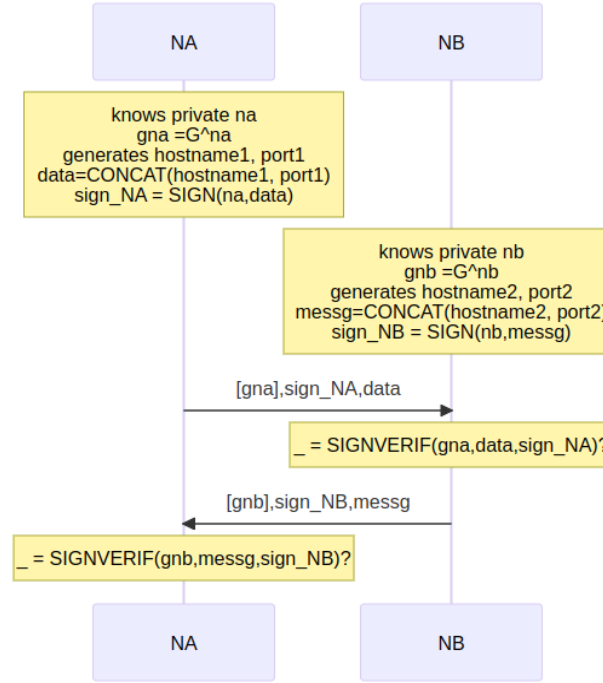


Fig 30 : Sequence diagram of Handshake messaging

4.2.2 Join Message Scenario

Every identity belongs to the individual node. To identify the identity introduced by the node, a join message is transmitted containing both the node and the identity token. The full codes with result are in the appendix named as Join_Message1.vp and Join_Message2.vp. Both files are intended for verification of join message of NA and NB consecutively. They follow the same procedures. Only we will discuss Join_Message1.vp for node NA to prevent repetition. In this scenario of fig 31, the join message carries the tokens of Alice and NA. From the handshake message, nodes NA and NB exchanged each other's public keys. From the initial state, node and identity both know each other's token with a fingerprint. According to Fig. 31, the node token (nt) and identity token (it) are common information for Alice and NA. Due to the compatibility of the Verifpal tool, they generate each other's fingerprints. In the real scenario, an individual entity is allowed to generate its own fingerprint. In real communication, they are connected to a secure link that we define here with a symmetric key sa (it doesn't exist). Both know the key, so encryption and decryption can be performed easily. In the first token of the join message, Alice generates symmetric ciphertext "j0" with the concatenation message "j" that is combined with pk "ga", signature "js1" and node fingerprint "nfp". Node NA decrypts "j0" and encrypts i

again with a secret key "s" that is computed from NB's pk "gnb". The message is transmitted to NB with nonce c1. NB decrypts the message and splits the decrypted message "j1_de". Then he found pk "ga", signature "js1", node fingerprint "nfp". For the signature verification, he considered pk "ga" and "nfp" as input.

In the second portion of the join message, NA sent his signature js2 which is derived from private key na and identity fingerprint ifp. Using the CONCAT algorithm, signatures and identity fingerprints are assembled in js3. By using the same key "s", node NA encrypts the concatenated message with nonce c1. Same way, NB decrypts again the second part, "j2". Since both tokens are part of the join message, so they carry the same nonce "c1". Then he receives js2, ifp_ after splitting the decrypted message "j2_de" and verifies the signature js2 with ifp_, gna.

A special note is needed to mention that all messages will be encrypted through the AEAD_ENC formula in Verifpal. Nodes have a copy of each other's authentic public key from a handshake token, so they use it to generate a secret key with their own private key by applying the exponential function. For example: (node NB's public key)^{NA's private key} = gnb^{na} .

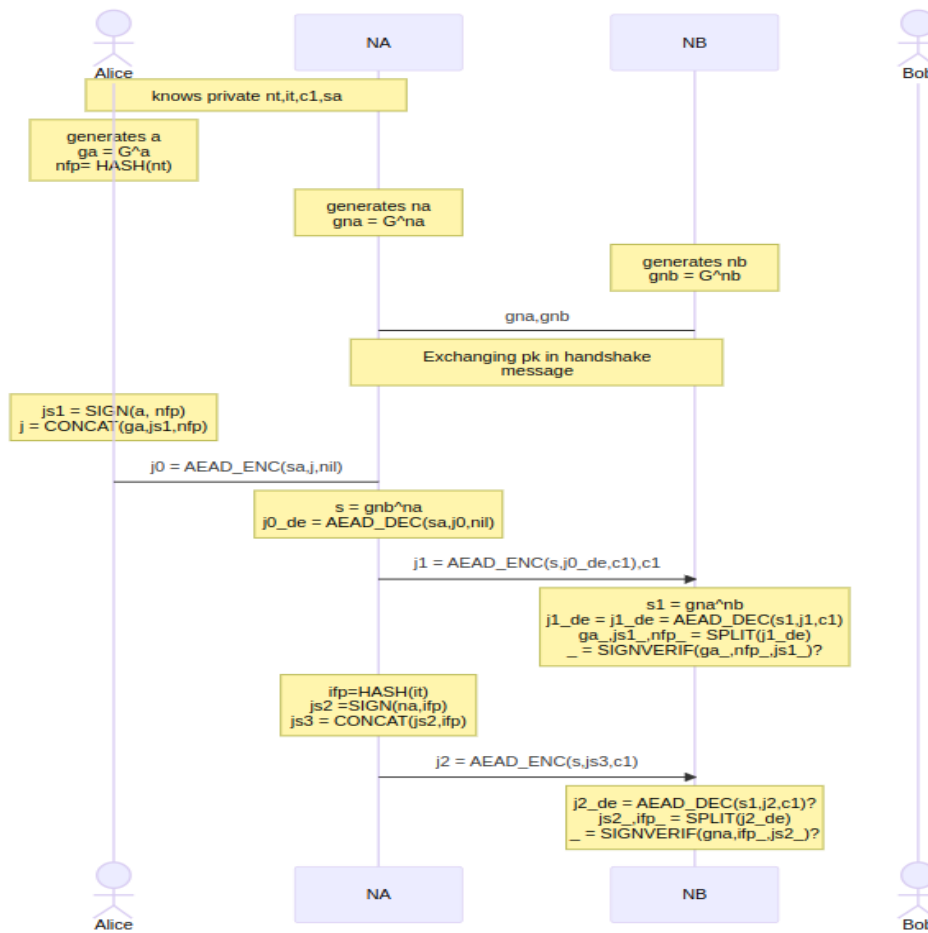


Fig 31 : Sequence diagram of Join messaging

4.2.3 DHT Message Scenario

The third message is for growing the network through the DHT message (in appendix:DHT_Message.vp). To create the environment for DHT messages, two new nodes are added here. Fig. 32 and 33 show connection establishment. Through the public key and fingerprint ((gna,fpA),(gnb,fpB),(gnc,fpC),(gnd,fpD)) we define the handshake message so that nodes can generate the secret key (shAB, shBA, shBC, shCB, shCD, shDC). Through the secret key, it is clear that node NA establishes a connection with NB, and node NB also has links with node NC. Node ND has only a connection with Node NC.

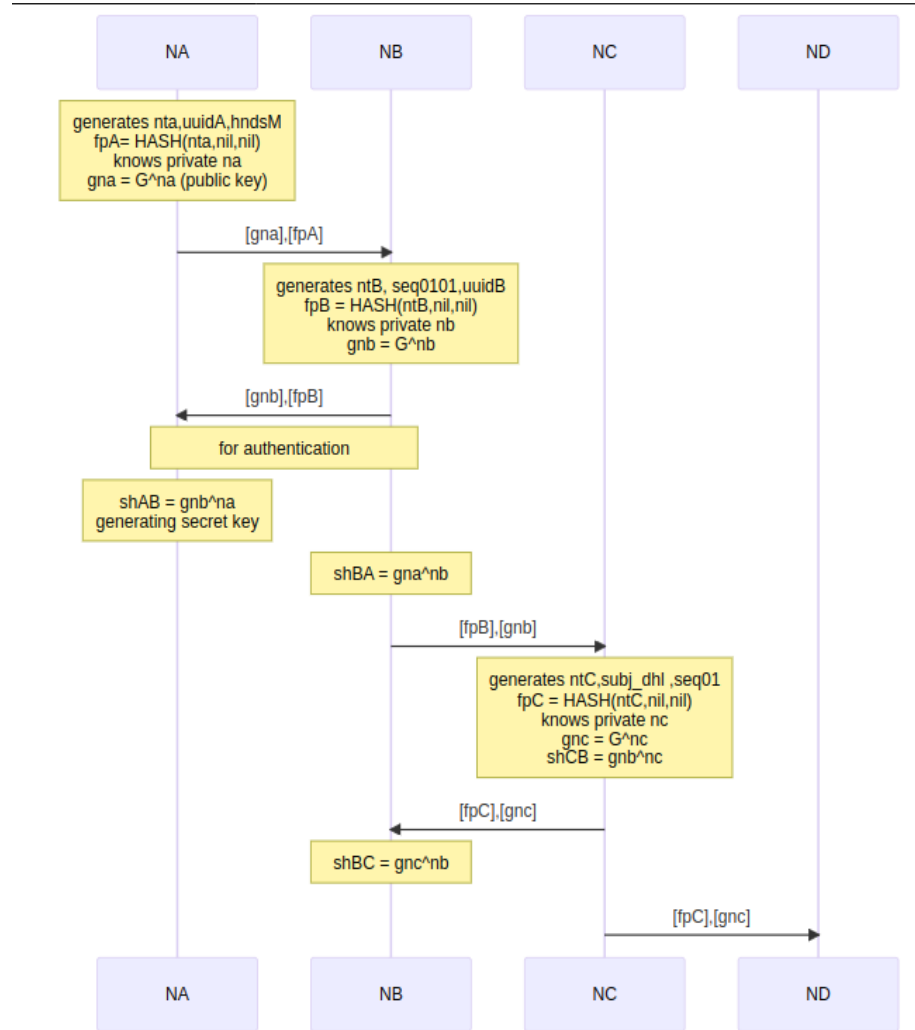


Fig 32: Establish the connection with new nodes

Since node NC has a connection with node NB and ND, it transmits ND's fingerprint to node NB through an update message "dht". Generally, "[" is used to ensure that the attacker can't modify the message. NB decrypts the message and forwards the encapsulated message "dht01" to node NA, adding NC's fingerprint. NA decrypts message dht01 and transmits another message "ping" to NB. NB sends back the acknowledgement through the ack message, which contains only the uuid of the ping message named uuidA. To define the

equality of uuidA and decrypted ack messages, the ASSERT algorithm is used. From the decrypted message dht01_de, NA receives fingerprints with sequence number. According to the distributed hash table, node NA chooses the nearest node and establishes the connection with node NC as like handshake messaging.

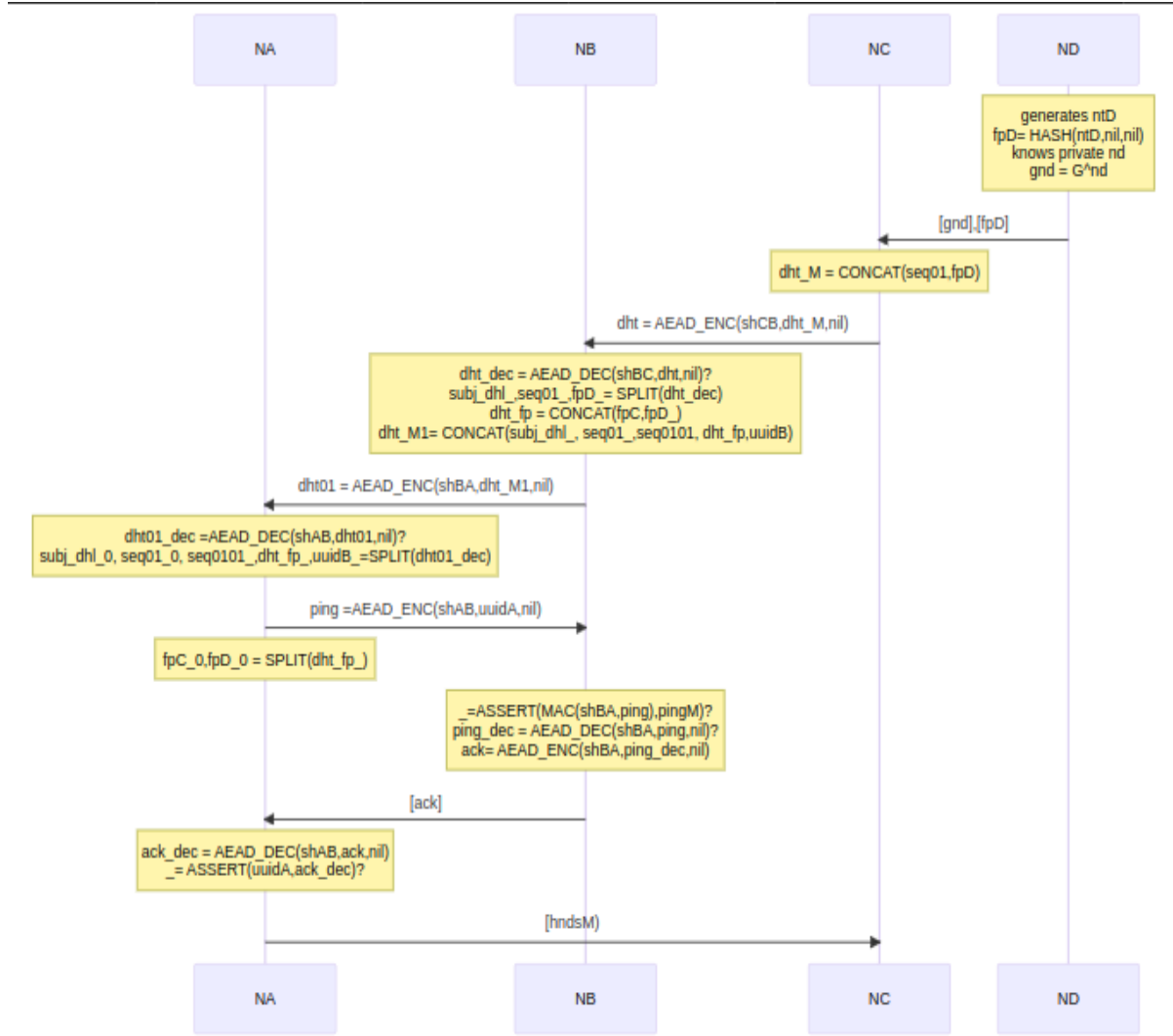
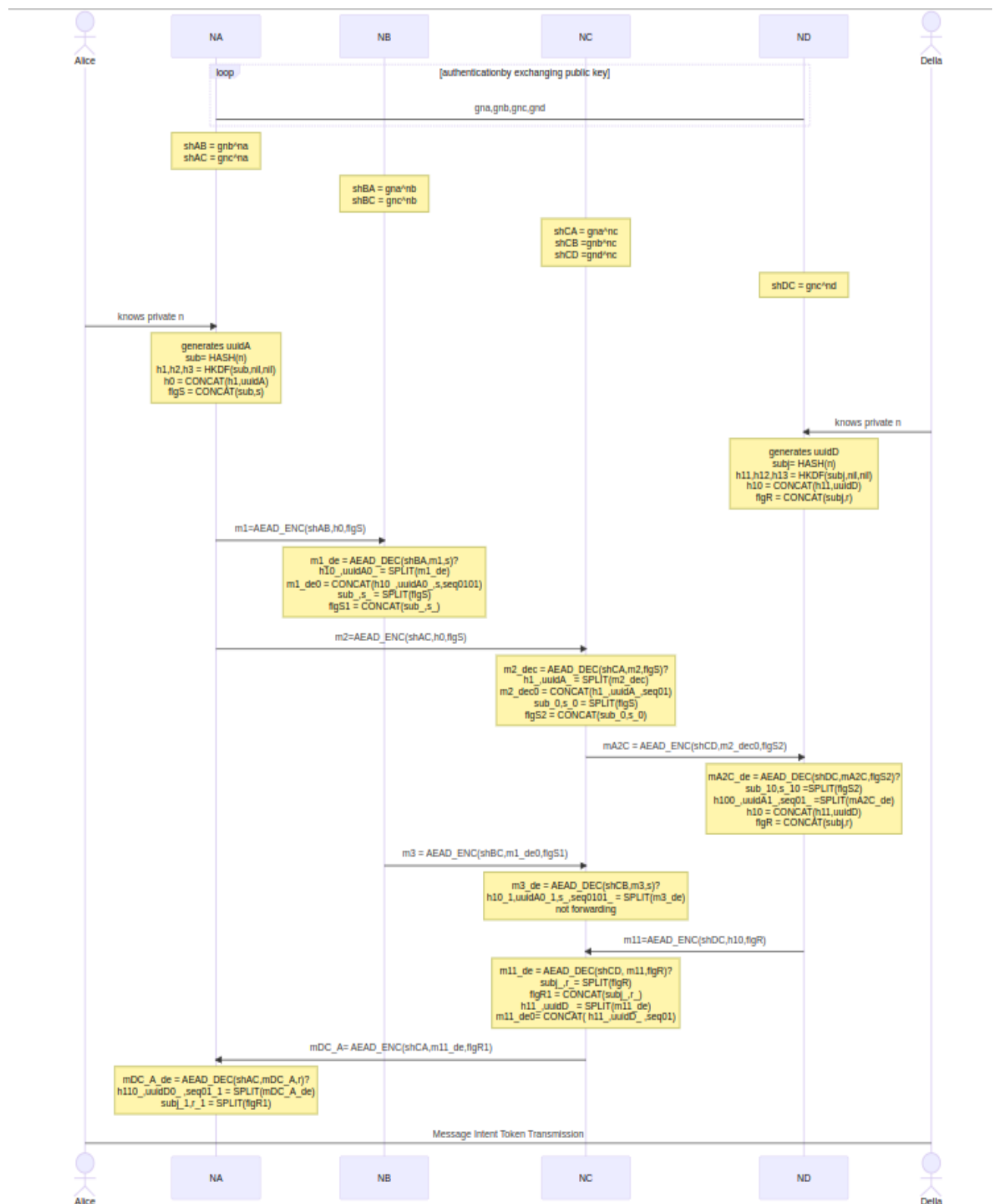


Fig 33 : DHT message transmission

4.2.4 Pheromone Message Scenario

This scenario is structured around discovering the peers through the pheromone message (appendix: Pheromone_Message.vp). For every message, a new specific file is created, so every time we need to authenticate the node and generate a secret key. After DHT message transmission, a connection is established between NA and NC with a secret key (called $shAC$ and $shCA$ and both are equal). There are two identities named Alice and Della.



We have already introduced the fact that Alice is connected to the network through node NA. Similarly, in maintaining a join message, Della is involved with node ND. As identity Alice generates subject n, node NA creates its hash value, called subj. According to theory, a bloom filter will be generated that is formed with the hash values. Due to the lack of bloom filter formulas in Verifpal, we use the HKDF (HMAC-based Key Derivation Function) [52] algorithm to produce BL (h1, h2, h3, or h11, h12, h13). Generally, every message contains a uuid, here we generate the scent message h0 with a bloom filter including uuidA. The sender and receiver both have an interest in the same subject, and both broadcast bloom filters in all probabilistic ways. To distinguish between the initiator's and the recipient's pheromone message, it contains a flag. Generally, it is part of the "To field" of the header and sits at the end hash value of the subject. Here we use the private value s/r and add it to the concat message (flgS) showing fig. 34. Then NA sent an encrypted scent message (m1) to nodes NB and NC with the flag address flgS. Node NB decrypts the message, adds the sequence number (seq0101), encrypts again, and forwards m2 to NC. When NC got two messages with the same uuidA, based on sequence number, it chose NA's message for sending forward. Nodes always prefer low-sequence numbers for communication. The sequence value defines the number of nodes used to forward the message. It also helps to mark the shortest path. NC forwards NA's pheromone message at the same time, it got a pheromone message (m11) from ND. Containing flags, nodes can differ between initiator and recipient, then forward the message to NA in the shortest path. The message intent token will be transmitted from an entity that receives the pheromone message first. According to fig. 34, the pheromone message of NA discovers Della first so The message intent token is transmitted from Node ND

4.2.5 MIT Scenario

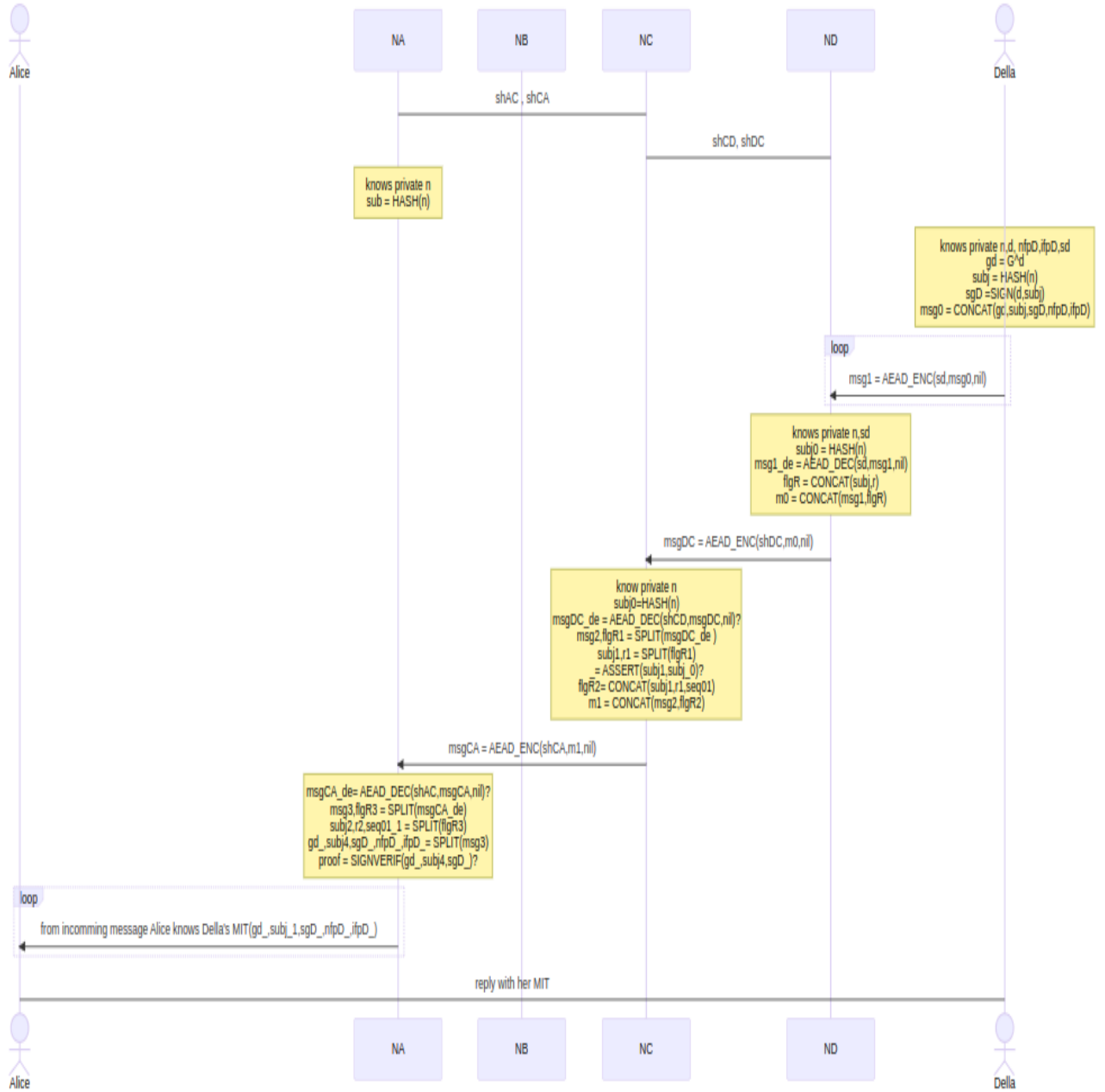


Fig 35 :Sequence diagram of Message intent token transmission

Through the pheromone message the connection is established via the shortest path (NA->NC->ND). Next all messages of the same subject will reach the destination using the same route. After defining the subject, The Message intent token (appendix:Message_Intent_Token.vp) is transmitted to authenticate the identity (Della) who are interested in the same topic. For this specific reason an identity token is sent out containing the hash value of the subject. Here the signature of Della is generated based on MIT. According to the function of

neuropil node ND receives automatically what identity generates. But that is not possible in Verifpal. So we create symmetric cipher text between node and identity to indicate a secure connection. Node ND encrypts MIT and sends node NC. From decrypted message, NC achieves header field address as flgR in our scenario. Then it splits and finds the hash value of subject(subj) that node know from pheromone message and r what is considered a flag of recipient. Adding sequence number seq01, the Message(msgCA) is encrypted by NC and goes forward to node NA. The initiator node of the subject decrypts the message. It receives the hash value of the subject from (flgR) and signature after splitting the message(msg3). To define the identity and node, NA gets the fingerprint of Della's (ifpD) and node's(nfpD). The final part of this message is about verifying the authentication. So signature(sgD_) is verified by Della's public key using SIGNVERIF algorithm. Since Alice and node NA are interconnected, by default Alice achieves the Della's public key.

Similar way Alice transmits MIT through NA. It follows the same processing of Della. we didn't show Alice message transmission process because the output of exchanging MIT takes a long time to provide the final result. To avoid repetition and unnecessary extra files we present only Della's message transmission.

4.2.6 Userspace Message Scenario

After authentication by MIT, the communication maintains end-to-end encryption and only identities have access to the message. From the previous step, nodes have generated a long-term secret key "shAC,shCD" (in the previous message, it is proved that shAC = shCA, so here one key is taken) to communicate node to node, and they already knew the subject named here as sub. In this scenario (Fig. 36), Alice generates ephemeral key addresses as epk. She shares this key as an encryption cipher with a secret key (sk). The secret key is formed with Della's public key which is contained in MIT. The message is encrypted with a nonce "c". During transmission, NA adds a header (hd1) with the message "en" using the CONCAT algorithm. The header field carries the pheromone message's hash values for the subject "sub" and the flag of the initiator named "s". In a pheromone message, the process of the hash function is already shown, so we choose a fixed value in this step.

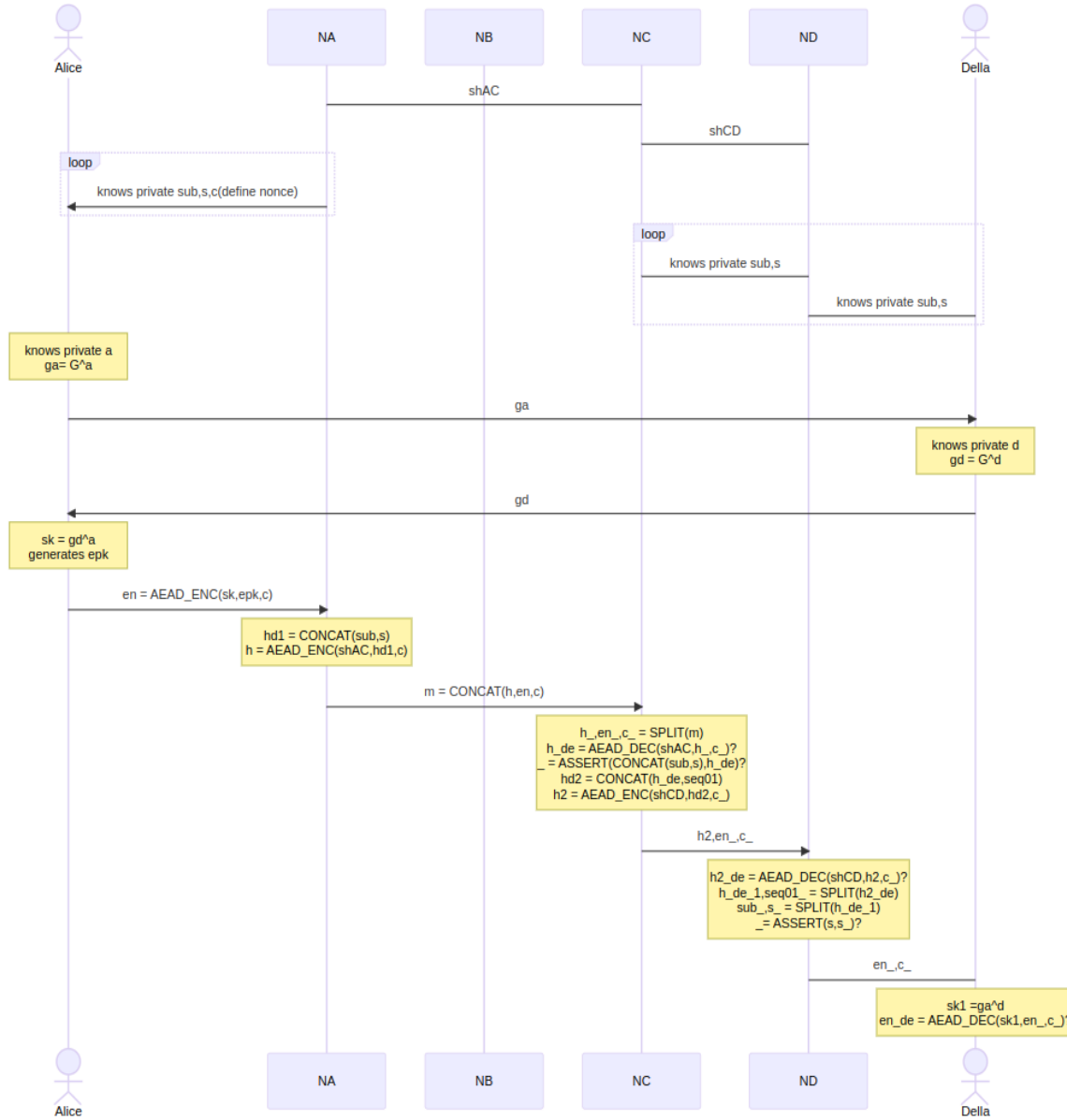


Fig 36: Sequence diagram of userspace message

Then the header is encrypted by "shAC" for node-to-node communication. Node "NA" assembles all encrypted ciphers (h, en) with nonce " c " using the CONCAT algorithm and sends them to NC. Node "NC" splits the message and finds the header " $h_$ " that is equal to h . Then it decrypts the message and discovers the pheromone message subject " sub " and the initiator's flag " s ". For confirmation of the subject of the pheromone

message, the ASSERT algorithm is used. Next, the new sequence number is added as a new member in the header field (hd2) and encrypted (h2) with shCD.

Node "ND" decrypts the header "h2," then splits the message two times using the SPLIT algorithm and receives the same subject. To ensure an authenticated sender, it applies the ASSERT algorithm to the flag of the initiator. Since identity and node share a secure communication, So Della receives the encrypted cipher of the ephemeral key, and she is able to decrypt the message or not, which will be asked in queries in Chapter 5. Moreover full code is attached in the appendix: User_space_Message.vp

4.2.7 End-To-End Encrypted Message With Multiple Receivers Scenario

In every scenario, the first public key transmission and secret key generation are shown because every file of a message is individual. One file of Verifpal can't use another file, like the "import" of Python programming.

In earlier steps, the shared secret key and EK creation and transmission methods are depicted, so to avoid repentance, the keys are directly sent out in this step. Fig. 37 draws direct transmission of the ephemeral key "epk" and the secret keys (shAC, shCD). Two recipients are Della and Dorry for multiple communication. Alice authenticated Dorry, and the same process is followed for Della. Afterward she receives the same ek from Alice, since they are interested in the same subject. Here the message "Up" is used that the sender transmits to receivers. Since Neuropil is suitable for IoT devices, So to change the movement of the robot, "Up" and "Down" are common messages sent by the user. Following the concept, Alice generates a message "up" and encrypts the message (en) with the ephemeral key. Then the rest of the steps are similar to the previous message. For NA, encrypt the header hd1 with shAC, and for NC, check the decrypted header using the ASSERT algorithm and forward the encrypted messages (h2, en_) with the nonce (c_). Then ND decrypts the part of the message (h2) that he is able to. Della and Dory both belong to node ND. For secure communication, both identities are connected with a symmetric key to node ND. They are able to decrypt the message "en_" and achieve the real message "up". Through the queries, the question will be asked to verify whether the message was modified by an active attacker or not. The queries and explanations will be given in Chapter 5 and appendix (E2EE_with_Multiple_Recivers.vp)

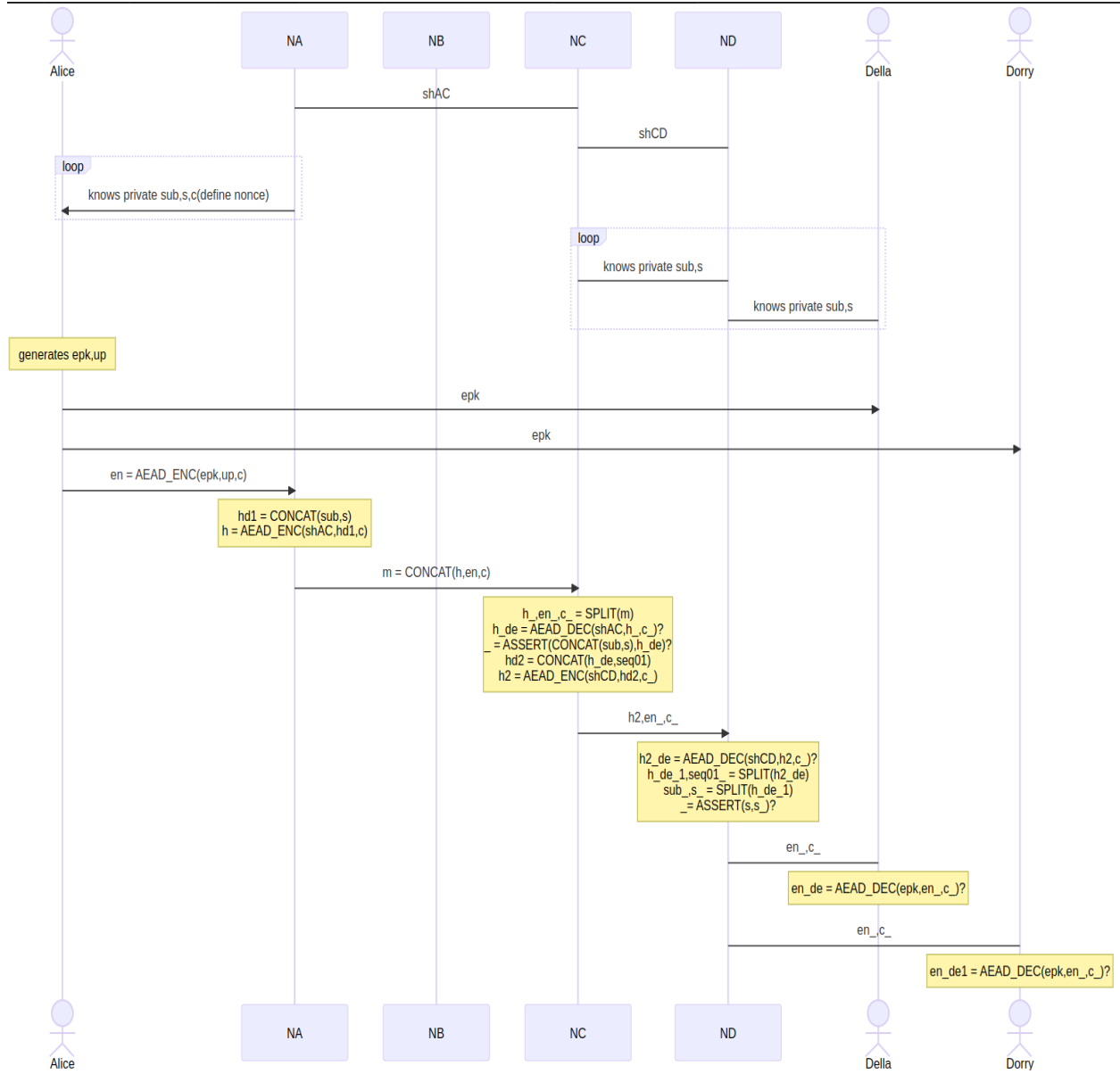


Fig 37: Sequence diagram of E2EE with multiple recipient

4.3 Architecture Of MQTT

To understand the security parameters of neuropil protocol, a comparison is also added here. The basic MQTT model is designed for Verifpal analysis. traditional MQTT doesn't consider any encryption methods. So the protocol is unsafe. All credentials of identity (username and password) are transmitted in plaintext. During communication, anyone on the network can see the message.[53] It might use the pre-shared key for encryption. The model is drawn in this paper based on the default concept of MQTT.

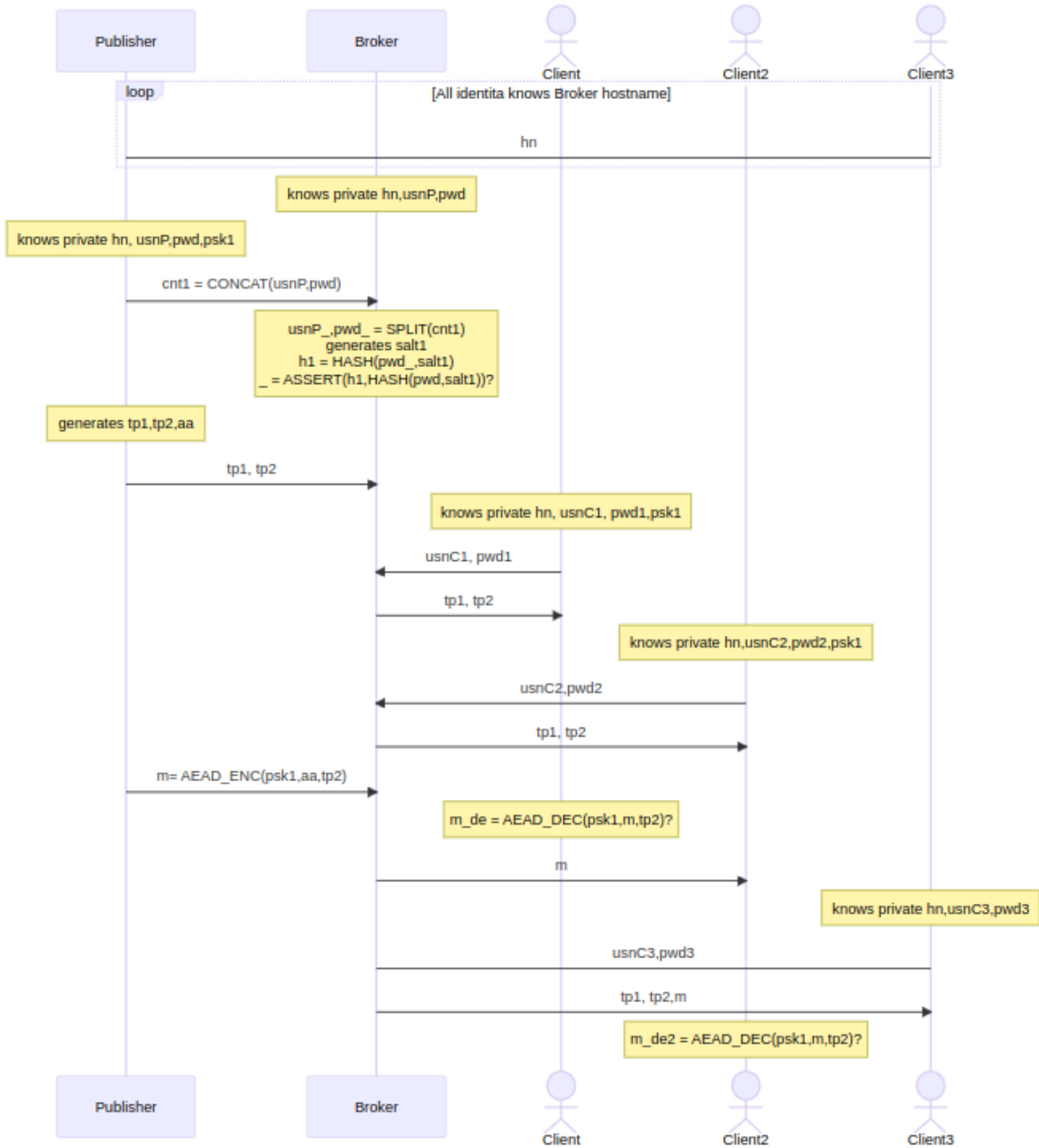


Fig 38: The model of basic MQTT protocol [appendix:MQTT_Model.vp]

In Fig. 38, we introduce three clients, publishers and brokers. Every identity in the network knows the broker's hostname (hn), and other identities are authenticated by username (usnP, usnC1, usnC2, usnC3) and password (pwd, pwd1, pwd2, pwd3). The first publisher sends the connect message "cnt" to the broker. It is a concatenated message of username and password using the CONCAT algorithm. The broker receives the message and splits

it. It discovers password `pwd`. For authentication, the broker adds the salt value `salt1` to the password and produces the hash value. The authentication of identity will be successful if hash values are matched with broker-stored hash values. Parallely, other clients send their username and password.

After authentication, the publisher publishes topics that we explain by sending topics (`tp1`, `tp2`) to the broker. Then the topics are forwarded. All identities are able to subscribe to their interests. Fig. 55 shows that Client and Client2 subscribe to the topic **tp2**. The publisher sends the broker a message "m" with "tp2," and it is encrypted using a pre-shared key. As it is confidential, the broker can't decrypt the message but duplicates the message and transmits it to all clients. The subscribers are only able to decrypt. Only Client and Client2 decrypt the message in Fig. 38. Now the question is if the attacker can break the message or not that will show in the next chapter.

In this Chapter, we have designed the model for every message. Their procedure is independent in Verifpal. The symbolic form of implementation is attached at the end of the report in the appendix. The result is also included. In the next chapter the queries will be added that need to ask to Verifpal tool for verification.

5.Result and Discussion

5.1 Analysis The output

The chapter is designed with queries and outcomes explanation. After the overview of the scenarios,a few questions arise that are asked of Verifpal. It displays the vulnerabilities in the result.All code and result are included in the appendix. This chapter will also include a comparison with the current well-known protocol MQTT to help readers understand the security of neuropil.

5.1.1 Handshake Message Result

The model of Handshake message is already discussed in Chapter 4. There, NA and NB exchange their digital signatures with messages called sign_NA,sign_NB,data,messg including the public key. We wanted to know whether transmitting messages is authenticated and confidential or not.

```

30
31 queries[
32   authentication? NA->NB: sign_NA
33   authentication? NB->NA: sign_NB
34   confidentiality? data
35   confidentiality? messg
36   confidentiality? hostname1
37 ]

```

Fig 39 : Queries of Handshake message

In queries, we ask about the node's authenticity through the authentication scheme. That means asking about the authentication of NA when it transmits a digital signature (sign_NA) to NB, and similarly asking about the signature (sign_NB) of node NB. In queries, we also mention the confidentiality of certificate-verified messages, which are used to create signatures such as data, messg, and hostname1.

```

Verifpal • Verification completed for 'hm.vp' at 11:26:21 AM.
Verifpal • Summary of failed queries will follow.

Result • confidentiality? data –
        data (CONCAT(hostname1, port1)) is obtained by Attacker.

Result • confidentiality? messg –
        messg (CONCAT(hostname2, port2)) is obtained by Attacker.

Result • confidentiality? hostname1 –
        hostname1 (hostname1) is obtained by Attacker.

Verifpal • Thank you for using Verifpal.
shabnaz-dev%

```

Fig 40:The result of Handshake Message

The output of Verifpal is shown by pointing out the red mark if the answer to queries is "no." In Fig. 40, we ask about the authentication and confidentiality of the message, but the answer is no. So only the red mark answer is visible in Fig. 40. For example, the first two questions are about the authentication of sign_NA and sign_NB. **The recipient** verifies it with a public key. After analysis, it gains proof of the authenticity of digital identities as a result the answers aren't shown in the result. The rest of the message, the attacker can obtain, so the answer to those messages is in red.

In neuropil handshakes, messages are transmitted in plaintext. Attackers are able to see the message if it is available on the network. He can gain all the information, including the hostname and port number of the handshake tokens which are considered as data and messg in our model. Due to ensuring authentication through the signature, he can't modify the message otherwise the token will be broken. Noted : The result is added in appendix.

5.1.2 Join Message Result

Next steps, nodes exchange the signature of their identity and node in the join message. For proper examination, the message is divided into two parts in Verifpal (j1,j2). Then the first two queries are about the authentication of j1 and j2. Since they are symmetric ciphers, we ask about the equivalence of secret keys (s,s1). J1 contains the signature of Alice (sign_A). So the internal element, such as nfp or ifp, is also designated as a question to define its confidentiality.

```
queries[
  authentication? NA->NB:j1
  authentication? NA->NB:j2
  equivalence? s,s1
  confidentiality? ifp
  confidentiality? nt
  confidentiality? nfp
]
```

Fig 41: The queries of Join message

Verifpal assesses the questions while under attack. So, the outcome only displays information to which the attacker has access. It means that the attacker is able to obtain or modify the message. The image of the result is added but full output is provided in the appendix Join_Message1.vp and Join_Message2.vp.

```

Deduction • Output of AEAD_ENC(nil, CONCAT(G^nil, SIGN(c1, HASH(nil))), HASH(
Deduction • Output of AEAD_ENC(nil, CONCAT(G^nil, SIGN(c1, HASH(nil))), HASH(
Deduction • Output of AEAD_ENC(c1, CONCAT(G^na, SIGN(nil, HASH(nil))), HASH(d
Deduction • Output of AEAD_ENC(c1, CONCAT(G^na, SIGN(nil, HASH(nil))), HASH(d
Deduction • Output of AEAD_ENC(c1, CONCAT(G^na, SIGN(c1, HASH(nil))), HASH(n
Deduction • Output of AEAD_ENC(c1, CONCAT(G^na, SIGN(c1, HASH(nil))), HASH(n
Deduction • Output of AEAD_ENC(c1, CONCAT(G^na, SIGN(c1, HASH(nil))), HASH(c
Deduction • Output of AEAD_ENC(c1, CONCAT(G^na, SIGN(c1, HASH(nil))), HASH(c
Deduction • Output of AEAD_ENC(c1, CONCAT(G^nb, SIGN(nil, HASH(nil))), HASH(d
Deduction • Output of AEAD_ENC(c1, CONCAT(G^nb, SIGN(nil, HASH(nil))), HASH(d
Deduction • Output of AEAD_ENC(c1, CONCAT(G^nb, SIGN(c1, HASH(nil))), HASH(n
Deduction • Output of AEAD_ENC(c1, CONCAT(G^nb, SIGN(c1, HASH(nil))), HASH(n
Deduction • Output of AEAD_ENC(c1, CONCAT(G^nb, SIGN(c1, HASH(nil))), HASH(c
Deduction • Output of AEAD_ENC(c1, CONCAT(G^nb, SIGN(c1, HASH(nil))), HASH(c
Deduction • Output of AEAD_ENC(c1, CONCAT(G^nil, SIGN(nil, HASH(nil))), HASH(
Deduction • Output of AEAD_ENC(c1, CONCAT(G^nil, SIGN(nil, HASH(nil))), HASH(
Deduction • Output of AEAD_ENC(c1, CONCAT(G^nil, SIGN(c1, HASH(nil))), HASH(r
Deduction • Output of AEAD_ENC(c1, CONCAT(G^nil, SIGN(c1, HASH(nil))), HASH(r
Deduction • Output of AEAD_ENC(c1, CONCAT(G^nil, SIGN(c1, HASH(nil))), HASH(d
Deduction • Output of AEAD_ENC(c1, CONCAT(G^nil, SIGN(c1, HASH(nil))), HASH(d
Stage 6, Analysis 210...

Verifpal • Verification completed for 'jml.vp' at 02:16:39 PM.
Verifpal • All queries pass.

Verifpal • Thank you for using Verifpal.
shabnaz-dev% █

```

Fig 42 : The Result of join message

In Fig. 42, the evaluation is shown. All information has been successfully passed. Because of secure mechanisms, the attacker can't obtain the message.

5.1.3 DHT Message Result

In the DHT, the update and piggy message contain only the fingerprint of the node. That is structured with the hash values. Logically, the attacker can't gain information from **nfp**. It was shown in the previous scenario. So the ping and acknowledgement message are highlighted here. First, we inquired about the equivalence of the secret keys, then the confidentiality and authentication of **ack** and **ping messages**. The question about confidentiality and freshness **uuidA** of the ack or ping message is an essential element of these messages.

```

queries[
  equivalence?shAB,shBA
  confidentiality?ack
  confidentiality?ping
  authentication?NB ->NA : ack
  confidentiality? uuidA
  freshness? uuidA
  equivalence?uuidA,ack_dec
]

```

Fig 43: Queries of DHT message

```

Result • confidentiality? ack — When:
shab → G^nb^na
shba → G^na^nb
shcb → G^nb^nc
shbc → G^nc^nb
dht_m → CONCAT(seq01, HASH(ntd, nil, nil))
dht → AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil)
dhtm → MAC(G^nb^nc, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil))),
unnamed_0 → ASSERT(MAC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(
dht_dec → AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd,
subj_dhl → SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, H
seq01 → SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(n
dht_fp → CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC
dht_m1 → CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq0
(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil))), nil), u
dht01 → AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb
), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil))),
dht01m → MAC(G^na^nb, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb
(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(nt
unnamed_1 → ASSERT(MAC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_D
), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil))),
dht01_dec → AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC
CAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01,
subj_dhl_0 → SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD
01, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT
seq01_0 → SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD
CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(se
seq0101 → SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD
, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(s
dht_fp → SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD
CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(se
uuidb → SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD
CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq
ping → AEAD_ENC(G^nb^na, uuida, nil) — obtained by Attacker
pingm → MAC(G^nb^na, AEAD_ENC(G^nb^na, uuida, nil))
fpc_0 → SPLIT(SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD
101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCA
fpc_0 → SPLIT(SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD
101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCA
unnamed_2 → ASSERT(MAC(G^na^nb, AEAD_ENC(G^nb^na, uuida, nil)), MAC(G^nb^na
ping_dec → AEAD_DEC(G^na^nb, AEAD_ENC(G^nb^na, uuida, nil), nil)?
ack → AEAD_ENC(G^na^nb, AEAD_DEC(G^na^nb, AEAD_ENC(G^nb^na, uuida, nil), nil)
ack_dec → AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, AEAD_DEC(G^na^nb, AEAD_ENC(G
unnamed_3 → ASSERT(uuida, AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, AEAD_DE
ack (AEAD_ENC(G^nb^na, uuida, nil)) is obtained by Attacker.

```

Fig 44 : the result of DHT message 1

```

Result • confidentiality? ping — When:
shab → G^nb^na
shba → G^na^nb
shcb → G^nb^nc
shbc → G^nc^nb
dht_m → CONCAT(seq01, HASH(ntd, nil, nil))
dht → AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil)
dhtm → MAC(G^nb^nc, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil))),
unnamed_0 → ASSERT(MAC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd,
dht_dec → AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil,
subj_dhl → SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(n
seq01 → SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, n
fpc_0 → SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, n
dht_fp → CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb
dht_m1 → CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HA
(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil))), nil), u
dht01 → AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc,
), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil))),
dht01m → MAC(G^na^nb, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD
(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, n
unnamed_1 → ASSERT(MAC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^
CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HA
), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil))),
dht01_dec → AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc
CAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH
subj_dhl_0 → SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_D
01, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq0
seq01_0 → SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC
CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01,
seq0101 → SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC
, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01,
dht_fp → SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC
CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01,
uuidb → SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G
CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, H
ping → AEAD_ENC(G^nb^na, uuida, nil) — obtained by Attacker
pingm → MAC(G^nb^na, AEAD_ENC(G^nb^na, uuida, nil))
fpc_0 → SPLIT(SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD
101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCA
fpc_0 → SPLIT(SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD
101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCA
unnamed_2 → ASSERT(MAC(G^na^nb, AEAD_ENC(G^nb^na, uuida, nil)), MAC(G^nb^na
ping_dec → AEAD_DEC(G^na^nb, AEAD_ENC(G^nb^na, uuida, nil), nil)?
ack → AEAD_ENC(G^na^nb, AEAD_DEC(G^na^nb, AEAD_ENC(G^nb^na, uuida, nil), n
ack_dec → AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, AEAD_DEC(G^na^nb, AEAD_ENC(G
unnamed_3 → ASSERT(uuida, AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, AEAD_DEC(G^n
ping (AEAD_ENC(G^nb^na, uuida, nil)) is obtained by Attacker.

```

Fig 45 : The result of DHT message 2

According to the result, the attacker is able to obtain the **ack** and **ping** messages, but he can't achieve uuidA. That means he can see encrypted messages but can't decrypt them. So uuidA isn't exposed.

5.1.4 Pheromone Message Result

The fourth scenario is about detecting matches among peers based on a user-generated subject. Here, the subject "n" is a sensitive element. The hash value subj or sub is generated from "n." The BF h1, h2, and h3 are produced from **sub** (similar to h11, h12, and h13 from **subj**). The message mDC_A contains BF with uuidA.

```
queries[]
    confidentiality? n
    equivalence? sub,subj
    confidentiality? h11
    confidentiality?mDC_A
    equivalence?h1,h110_
    confidentiality? uuidD_
    authentication?NC->ND:mA2C
    equivalence?h100_,h11
    equivalence? uuidA0_,uuidA0_1
    confidentiality? uuidA0_
    equivalence?h10_1,h1_
    equivalence? subj,sub_10
```

Fig 46 : The queries of pheromone message

The question explanation is given on table -

Queries	Illustration
confidentiality? n	Identity generated subject are hidden or not
equivalence?sub,subj	The generated subject's hash value of NA and ND are same or not
Confidentiality?h11	The BF of ND is confidential or not
Confidentiality?mDC_A	The transmitted message from NC to NA is confidential or not.The message is originated at node ND
Equivalence?h1,h110_	NA's BF and receiving BF are equal or not
Confidentiality? uuidD_	The uuid of message mDC_A is confidential or not
Authentication?NC->ND:mA2C	The transmitted message from NC to ND is authenticated or not.
Equivalence?h100,h11	The BF that ND receives from mA2C is equal to h11
Equivalence? uuidA0_,uuidA0_1	The uuid of node NB(receive from NA) and

	NC(receive from NB) is same or not
Confidentiality?uuidA0_	The uuid of node NB(receive from NA) is confidential or not
Equivalence?h10_1,h1_	NC receives two messages(NA->NB->NC,NA->NC).they are equivalent or not?
Equivalence?subj,sub_10	subj(generated in ND and sub_10 receive from NC) are same or not

Table 7:The pheromone message queries with explanation

```

Result • confidentiality? mdc_a - When:
h1 → HKDF(HASH(n), nil, nil)
h2 → HKDF(HASH(n), nil, nil)
h3 → HKDF(HASH(n), nil, nil)
h11 → HKDF(HASH(n), nil, nil)
h12 → HKDF(HASH(n), nil, nil)
h13 → HKDF(HASH(n), nil, nil)
shab → G^nb^na
shac → G^nc^na
h0 → CONCAT(HKDF(HASH(n), nil, nil), uuida)
flgs → CONCAT(HASH(n), s)
m1 → AEAD_ENC(G^nb^na, CONCAT(HKDF(HASH(n), nil, nil), uuida), CONCAT(HASH(n), s))
m2 → AEAD_ENC(G^nc^na, CONCAT(HKDF(HASH(n), nil, nil), uuida), CONCAT(HASH(n), s))
shba → G^na^nb
shbc → G^nc^nb
m1_de → CONCAT(HKDF(HASH(n), nil, nil), uuida)
h10 → HKDF(HASH(n), nil, nil)
uuida0 → uuida
m1_de0 → CONCAT(HKDF(HASH(n), nil, nil), uuida, seq0101)
sub_ → HASH(n)
s_ → s
flgs1 → CONCAT(HASH(n), s)
m3 → AEAD_ENC(G^nc^nb, CONCAT(HKDF(HASH(n), nil, nil), uuida, seq0101), CONCAT(HASH(n), s))
shca → G^na^nc
shcb → G^nb^nc
shcd → G^nd^nc
m2_dec → CONCAT(HKDF(HASH(n), nil, nil), uuida)
h1 → HKDF(HASH(n), nil, nil)
uuida → uuida
m2_dec0 → CONCAT(HKDF(HASH(n), nil, nil), uuida, seq01)
sub_0 → HASH(n)
s_0 → s
flgs2 → CONCAT(HASH(n), s)
ma2c → AEAD_ENC(G^nd^nc, CONCAT(HKDF(HASH(n), nil, nil), uuida, seq01), CONCAT(HASH(n), s))
m3_de → CONCAT(HKDF(HASH(n), nil, nil), uuida, seq0101)
h10_1 → HKDF(HASH(n), nil, nil)
uuida0_1 → uuida
seq0101 → seq0101
sub_1 → HASH(n)
s_1 → s
unnamed_0 → ASSERT(HASH(n), HASH(n))?
shdc → G^nc^nd
ma2c_de → CONCAT(HKDF(HASH(n), nil, nil), uuida, seq01)
sub_10 → HASH(n)
s_10 → s
h100 → HKDF(HASH(n), nil, nil)
uuida1 → uuida
seq01 → seq01
h10 → CONCAT(HKDF(HASH(n), nil, nil), uuidd)
flgr → CONCAT(HASH(n), r)
m11 → AEAD_ENC(G^nc^nd, CONCAT(HKDF(HASH(n), nil, nil), uuidd), CONCAT(HASH(n), r))
m11_de → CONCAT(HKDF(HASH(n), nil, nil), uuidd)
subj_ → HASH(n)
r_ → r
flgr1 → CONCAT(HASH(n), r)
h11 → HKDF(HASH(n), nil, nil)
uuidd → uuidd
m11_de0 → CONCAT(HKDF(HASH(n), nil, nil), uuidd, seq01)
mdc_a → AEAD_ENC(G^na^nc, CONCAT(HKDF(HASH(n), nil, nil), uuidd), CONCAT(HASH(n), r)) ← obtained by Attacker
mdc_a_de → CONCAT(HKDF(HASH(n), nil, nil), uuidd)
h110 → HKDF(HASH(n), nil, nil)
uuidd0 → uuidd
seq01_1 → nil
sub_1_1 → HASH(n)
r_1 → r
mdc_a (AEAD_ENC(G^na^nc, CONCAT(HKDF(HASH(n), nil, nil), uuidd), CONCAT(HASH(n), r))) is obtained by Attacker.

```

Fig 47: The result is about confidentiality of mdc_a in Pheromone message


```

Result • confidentiality? h11 - when:
h1 - HKDF(HASH(n), nil, nil) - obtained by Attacker
h2 - HKDF(HASH(n), nil, nil) - obtained by Attacker
h3 - HKDF(HASH(n), nil, nil) - obtained by Attacker
h11 - HKDF(HASH(n), nil, nil) - obtained by Attacker
h12 - HKDF(HASH(n), nil, nil) - obtained by Attacker
h13 - HKDF(HASH(n), nil, nil) - obtained by Attacker
shab - G^nb^na
shac - G^nc^na
h0 - CONCAT(HKDF(HASH(n), nil, nil), uuida)
flgs - CONCAT(HASH(n), s)
m1 - AEAD_ENC(G^nb^na, CONCAT(HKDF(HASH(n), nil, nil), uuida), CONCAT(HASH(n), s))
m2 - AEAD_ENC(G^nc^na, CONCAT(HKDF(HASH(n), nil, nil), uuida), CONCAT(HASH(n), s))
shba - G^na^nb
shbc - G^nc^nb
m1_de - CONCAT(HKDF(HASH(n), nil, nil), uuida)
h10 - HKDF(HASH(n), nil, nil) - obtained by Attacker
uuida0 - uuida
m1_de0 - CONCAT(HKDF(HASH(n), nil, nil), uuida, seq0101)
sub_ - HASH(n)
s_ - s
flgs1 - CONCAT(HASH(n), s)
m3 - AEAD_ENC(G^nc^nb, CONCAT(HKDF(HASH(n), nil, nil), uuida, seq0101), CONCAT(HASH(n), s))
shca - G^na^nc
shcb - G^nb^nc
shcd - G^nd^nc
m2_dec - CONCAT(HKDF(HASH(n), nil, nil), uuida)
h1 - HKDF(HASH(n), nil, nil) - obtained by Attacker
uuida - uuida
m2_dec0 - CONCAT(HKDF(HASH(n), nil, nil), uuida, seq01)
sub_0 - HASH(n)
s_0 - s
flgs2 - CONCAT(HASH(n), s)
ma2c - AEAD_ENC(G^nd^nc, CONCAT(HKDF(HASH(n), nil, nil), uuida, seq01), CONCAT(HASH(n), s))
m3_de - CONCAT(HKDF(HASH(n), nil, nil), uuida, seq0101)
h10_1 - HKDF(HASH(n), nil, nil) - obtained by Attacker
uuida0_1 - uuida
seq0101 - seq0101
sub_1 - HASH(n)
s_1 - s
unnamed_0 - ASSERT(HASH(n), HASH(n))?
shdc - G^nc^nd
ma2c_de - CONCAT(HKDF(HASH(n), nil, nil), uuida, seq01)
sub_10 - HASH(n)
s_10 - s
h100 - HKDF(HASH(n), nil, nil) - obtained by Attacker
uuida1 - uuida
seq01 - seq01
h10 - CONCAT(HKDF(HASH(n), nil, nil), uidd)
flgr - CONCAT(HASH(n), r)
m11 - AEAD_ENC(G^nc^nd, CONCAT(HKDF(HASH(n), nil, nil), uidd), CONCAT(HASH(n), r))
m11_de - CONCAT(HKDF(HASH(n), nil, nil), uidd)
subj_ - HASH(n)
r_ - r
flgr1 - CONCAT(HASH(n), r)
h11 - HKDF(HASH(n), nil, nil) - obtained by Attacker
uidd - uidd
m11_de0 - CONCAT(HKDF(HASH(n), nil, nil), uidd, seq01)
mdc_a - AEAD_ENC(G^na^nc, CONCAT(HKDF(HASH(n), nil, nil), uidd), CONCAT(HASH(n), r))
mdc_a_de - CONCAT(HKDF(HASH(n), nil, nil), uidd)
h110 - HKDF(HASH(n), nil, nil) - obtained by Attacker
uidd0 - uidd
seq01_1 - nil
subj_1 - HASH(n)
r_1 - r
h11 (HKDF(HASH(n), nil, nil)) is obtained by Attacker.

Verifpal • Thank you for using Verifpal.
shabnaz-dev%

```

Fig 48 : The result is about confidentiality of h11 in Pheromone message

The hash value of subject "n" is contained in mDC_A, and h11 is generated from this hash value. Now it is important to define whether the subject is still hidden or not. The confidentiality of the "n" inquiry yields yes answers. As a result, it states that an attacker cannot obtain "n." If the subject isn't disclosed, the contents of the message won't be revealed. As a result, the attacker might view but not comprehend the data. The other question's answer is yes, so Verifpal doesn't provide their result in output.

5.1.5 MIT Result

After pheromone messages, interested identities authenticate each other using MIT. The authentication is accomplished by signature verification. From the previous messages, it is clear that Alice and Della are both interested in the same subject. Alice is the initiator, and Della is the recipient. So through ND, Della transmits the message to node NC, and node NC transmits it to Alice through NA.

In queries, the first question is about authentication of the msgDC (from ND to NC) and its confidentiality. The third question is the confidentiality of the "r" that node ND added in msgDC. Most of the questions are about the confidentiality of the messages (msgCA) and their content (n,nfpD,gd). Through an equivalence query, it is verified that elements (for example, comparing between two hash values of subject subj4, sub, and between two public keys) are modified by the attacker or not.

```
queries[
  authentication? ND->NC:msgDC
  confidentiality? msgDC
  confidentiality? r1
  confidentiality?msgCA
  confidentiality? n
  confidentiality? nfpD
  equivalence?subj4, sub
  confidentiality? nfpD_
  confidentiality? gd
  equivalence? gd,gd_
]
```

Fig 49:The queries of MIT

```
Result • confidentiality? msgdc - When:
shac → G^nc^na
shca → G^na^nc
shdc → G^nc^nd
shcd → G^nd^nc
sgd → SIGN(d, HASH(n))
msg0 → CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd)
msg1 → AEAD_ENC(sd, CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), nil)
msg1_de → CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd)
flgr → CONCAT(HASH(n), r)
m0 → CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), CONCAT(HASH(n), r))
msgdc → AEAD_ENC(G^nc^nd, CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), CONCAT(HASH(n), r)), nil) ← obtained by Attacker
msgdc_de → CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), CONCAT(HASH(n), r))
msg2 → CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd)
flgr1 → CONCAT(HASH(n), r)
subj1 → HASH(n)
r1 → r
unnamed_0 → ASSERT(HASH(n), HASH(n))?
flgr2 → CONCAT(HASH(n), r, seq01)
m1 → CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), CONCAT(HASH(n), r, seq01))
msgca → AEAD_ENC(G^na^nc, CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), CONCAT(HASH(n), r, seq01)), nil)
msgca_de → CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), CONCAT(HASH(n), r, seq01))
msg3 → CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd)
flgr3 → CONCAT(HASH(n), r, seq01)
subj2 → HASH(n)
r2 → r
seq01_1 → seq01
gd → G^d
subj4 → HASH(n)
sgd → SIGN(d, HASH(n))
nfpd → nfpd
ifpd → ifpd
proof → nil
msgdc (AEAD_ENC(G^nc^nd, CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), CONCAT(HASH(n), r)), nil)) is obtained by Attacker.
```

Fig 50: The result of MIT part 1

According to the result, the attacker obtains the message msgdc. So it is not confidential any more, so all elements (r1) of the message are supposed to be disclosed. But there is no result for r1. Only obtaining the message means the attacker only sees the message and can't decrypt it.

```

Result • confidentiality? msgca - When:
shac → G^nc^na
shca → G^na^nc
shdc → G^nc^nd
shcd → G^nd^nc
sgd → SIGN(d, HASH(n))
msg0 → CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd)
msg1 → AEAD_ENC(sd, CONCAT(G^d, HASH(n), SIGN(d, HASH(n))), nfpd, ifpd, nil)
msg1_de → CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd)
flgr → CONCAT(HASH(n), r)
m0 → CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n))), nfpd, ifpd, CONCAT(HASH(n), r))
msgdc → AEAD_ENC(G^nc^nd, CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n))), nfpd, ifpd), CONCAT(HASH(n), r), nil)
msgdc_de → CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n))), nfpd, ifpd, CONCAT(HASH(n), r))
msg2 → CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd)
flgr1 → CONCAT(HASH(n), r)
subj1 → HASH(n)
r1 → r
unnamed_0 → ASSERT(HASH(n), HASH(n))?
flgr2 → CONCAT(HASH(n), r, seq01)
m1 → CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n))), nfpd, ifpd, CONCAT(HASH(n), r, seq01))
msgca → AEAD_ENC(G^na^nc, CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n))), nfpd, ifpd), CONCAT(HASH(n), r, seq01), nil) - obtained by Attacker
msgca_de → CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n))), nfpd, ifpd, CONCAT(HASH(n), r, seq01))
msg3 → CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd)
flgr3 → CONCAT(HASH(n), r, seq01)
subj2 → HASH(n)
r2 → r
seq01_1 → seq01
gd → G^d
subj4 → HASH(n)
sgd → SIGN(d, HASH(n))
nfpd → nfpd
ifpd → ifpd
proof → nil
msgca (AEAD_ENC(G^na^nc, CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n))), nfpd, ifpd), CONCAT(HASH(n), r, seq01), nil)) is obtained by Attacker.

Verifpal • Thank you for using Verifpal.
shabnaz-dev%

```

Fig 51: The result of MIT part 2

The message msgca is similar to msgdc. The contents of the message (gd_, nfpD_, gd) are still confidential.

5.1.6 User Space Message Result

The goal of this message is to share the ephemeral key "epk". The message "m" is concatenated with the EK containing the message "en" and headers "h" and nonce "c". To define the authentication of message "m", the query is asked to Verifpal for verification in Fig. 52. After receiving it from NA, NC forward it to ND, adding header h2. The addition of part "h2" adds only the sequence number of the previous header. During transmission, it is important to inquire about its authentication. The next question is about equivalence. The sending and receiving subjects (sub,sub_) are the same or not, and the question is asked through equivalence. The question about the confidentiality of Ek "epk" and "c" means whether an attacker obtains it or not. Through equivalence queries, it is asked whether the nonce "c" and after decryption "c_" both are the same or not. The same explanation is applicable to the query of equivalence between EK "epk" and en_de.

```
queries[
  authentication? NA->NC:m
  authentication? NC->ND:h2
  equivalence? sub_,sub
  equivalence? sk,sk1
  confidentiality? epk
  confidentiality? c
  equivalence? c,c_
  equivalence? epk, en_de
]
```

Fig 52: The queries of User space message

After analyzing the model, Verifpal provides the result that is shown in Fig. 53. Attackers are able to know the confidentiality of nonce "c". As it is transmitted as plaintext with the message. Attackers can obtain But it contains random values in real scenarios. So the attacker can't gain confidential information from nonce.

```
Result • confidentiality? c – When:
  sk1 → G^a^d
  hn → CONCAT(CONCAT(sub, s), c)
  sk → G^d^a
  en → AEAD_ENC(G^d^a, epk, c)
  h → AEAD_ENC(shac, CONCAT(sub, s), c)
  m → CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(G
  h → AEAD_ENC(shac, CONCAT(sub, s), c)
  en → AEAD_ENC(G^d^a, epk, c)
  c → c ← obtained by Attacker
  h_de → CONCAT(sub, s)
  unnamed_0 → ASSERT(CONCAT(sub, s), CONCAT(sub, s))?
  hd2 → CONCAT(CONCAT(sub, s), seq01)
  h2 → AEAD_ENC(shcd, CONCAT(CONCAT(sub, s), seq01), c)
  m2 → CONCAT(AEAD_ENC(shcd, CONCAT(CONCAT(sub, s), seq01),
  h2_de → CONCAT(CONCAT(sub, s), seq01)
  h_de_1 → CONCAT(sub, s)
  seq01_ → seq01
  sub → sub
  s → s
  unnamed_1 → ASSERT(s, s)?
  en_de → epk
  c (c) is obtained by Attacker.

Verifpal • Thank you for using Verifpal.
shabnaz-dev% □
```

Fig 53: The result of User space message

5.1.7 E2EE with Multiple Receivers Result

The message is end-to-end encryption with EK. Most questions in queries resemble the previous message. The exception query is about the authentication of the cipher text "en_" while it is transmitted from ND to Dory. To define advanced security, we ask about the confidentiality of message **en_** and containing information "**up**".

```
queries[
  authentication? NA->NC:m
  authentication? NC->ND:h2
  equivalence? sub_,sub
  authentication?ND->Dory:en_
  confidentiality?en_
  confidentiality? up
  confidentiality? c
  equivalence? c,c_
]
```

Fig 54: The queries of E2EE message

```
Verifpal • Verification completed for 'e2e.vp' at 06:22:57 PM.
Verifpal • Summary of failed queries will follow.

Result • confidentiality? en _ —
en_ (AEAD_ENC(epk, up, c)) is obtained by Attacker.

Result • confidentiality? c — When:
h → AEAD_ENC(shac, CONCAT(sub, s), c)
m → CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c), c)
h_ → AEAD_ENC(shac, CONCAT(sub, s), c)
en_ → AEAD_ENC(epk, up, c)
c_ → c ← obtained by Attacker
h_de → CONCAT(sub, s)
unnamed_0 → ASSERT(CONCAT(sub, s), CONCAT(sub, s))?
hd2 → CONCAT(CONCAT(sub, s), seq01)
h2 → AEAD_ENC(shcd, CONCAT(CONCAT(sub, s), seq01), c)
h2_de → CONCAT(CONCAT(sub, s), seq01)
h_de_1 → CONCAT(sub, s)
seq01_ → seq01
sub_ → sub
s_ → s
unnamed_1 → ASSERT(s, s)?
en_de → up
en_de1 → up
c (c) is obtained by Attacker.

Verifpal • Thank you for using Verifpal.
shabnaz-dev% □
```

Fig 55: The result of E2EE message

According to the result, the attacker is able to obtain the messages "en_" and nonce "c". The rest of the primitives are confidential, including "up". So the attacker knows the "en_" but can't obtain the "up". So it is clear that the attacker can record the cipher text but isn't able to decrypt it and nonce "c" explanation is the same as User-space-message.

5.2 The Result of MQTT

In section 4.3 of the previous chapter, we have explained the default architecture of MQTT. In the model, The publisher connects with the broker providing cnt1 with username(usnP) and password (pwd). There is an explanation about the process of storing the broker password. Additionally, the process for verifying password accuracy is given which involves comparing the incoming password with stored.

Publisher transmitted another message "m" to Client through the Broker. Its integrity and authenticity is asked to Verifpal by queries.

```
queries[]
  authentication? Publisher->Broker:cnt1
  confidentiality? cnt1
  equivalence? pwd,pwd_
  confidentiality? pwd
  authentication? Broker->Client:m
  confidentiality?m
  confidentiality?aa
```

Fig 56: The queries of MQTT

In queries, the transmitted message "cnt1" from publisher to broker is asked to verify its authentication and confidentiality. Through equivalence, it is asked to check whether the stored password (Pwd) and the receiving password (pwd_) are the same or not. "Confidentiality?" means that the password of the publisher is hidden or not. Another authentication query of "m" which resembles "cnt1" is transmitted from the broker to the client. But it originated with the publisher. The last two queries are about the confidentiality of **m** and **aa** that are contained in **m**.

```
Verifpal • Verification completed for 'mqq.vp' at 12:15:03 PM.
Verifpal • Summary of failed queries will follow.

Result • authentication? Publisher -> Broker: cnt1 - When:
cnt1 -> CONCAT(nil, nil) - mutated by Attacker (originally CONCAT(usnp, pwd))
usnp -> nil - obtained by Attacker
cnt1 (CONCAT(nil, nil)), sent by Attacker and not by Publisher,
is successfully used in SPLIT(CONCAT(nil, nil)) within Broker's state.

Result • confidentiality? cnt1 -
cnt1 (CONCAT(usnp, pwd)) is obtained by Attacker.

Result • equivalence? pwd, pwd_ - When:
cnt1 -> tp1 - mutated by Attacker (originally CONCAT(usnp, pwd))
usnp -> SPLIT(tp1)
pwd, pwd_ are not equivalent.

Result • confidentiality? pwd - When:
usnp -> usnp
pwd -> pwd - obtained by Attacker
h1 -> HASH(pwd, salt1)
unnamed_0 -> ASSERT(HASH(pwd, salt1), HASH(pwd, salt1))?
m_de -> aa
m_de2 -> aa
pwd (pwd) is obtained by Attacker.

Result • confidentiality? m -
m (AEAD_ENC(PSK1, aa, tp2)) is obtained by Attacker.

Verifpal • Thank you for using Verifpal.
shabnaz-dev% □
```

Fig 57: The result of MQTT protocol

According to the result, the attacker obtains cnt1 and is able to modify it since it is transmitted as plaintext. Equationally, if the attacker modifies cnt1, then pwd and pwd_ will not be the same. So pwd will not be confidential anymore. Using pre-shared keys, the message m is still confidential, though the pre-shared key transmission isn't clear yet.

5.3 Comparison With MQTT

The basic concept of MQTT doesn't have certificates for authentication, but it can support mutual authentication, which is also possible to implement in MQTT. It is unsafe to transmit authentication credentials in plaintext. With the deployment of the TLS protocol, MQTT becomes more secure, and the information will be transmitted without compromise. The X.509 certificate [54] is a standard authentication method for TLS. Public key infrastructure is confined to the certificate authorities who issue the certificate. A certificate is connected in a hierarchy chain that links one or more entities called root CAs with their self-signed certificates. [54] So the traditional X.509 certificates are too heavy for battery-powered or energy-harvesting Internet of Things (IoT) devices. [55] MQTT will not be a lightweight protocol anymore if TLS is applied with the digital certificate. The broker might consider a certificate but not other identities. Otherwise, performance will be slower.

Neuropil supports the JSON web token [56], which is very popular as an authentication scheme in web applications. The token is a compact and light-weight mechanism to secure the resource. It is suitable and easy for implementation and maintenance. [57] The token of Neuropil is designed with two different signatures, which ensures the identity's authenticity. In Chapter 3, an elaborate explanation is given.

Traditional MQTT follows a small payload without any encryption. So the message isn't secure. However, the payload encryption can be executed at the application level. Then publishers and subscribers are able to access the message that we showed in our model. Consequently, the question appears about the pre-shared key. It isn't clear how the peers can gain the key without telling the broker.

Neuropil maintains application and transport layer security. Using the Diffie-Hellman key algorithm, end-to-end communication is established. After MIT, the user assures secured communication by transmitting ek. The identity is able to obtain the ek if he only knows the opposite user's private key. Other security targets that Neuropil achieves are provided in the next section.

5.4 Information Security of Neuropil

5.4.1 Confidentiality

In Neuropil, messages are transmitted using transport encryption following the concept of the TLS1.3 protocol [58]. Only authenticated nodes in the network are able to decrypt the message. So the payload is secured and can be avoided by eavesdropping attacks. So adversaries can see the ciphertext but can't obtain the information of the message (which is already shown through Verifpal). Afterward, Identity assures the confidentiality of messages through end-to-end communication. The message is encrypted with random keys. So during communication from user to user, the messages are secured and hidden to maintain transport and application layer security together.

5.4.2 Integrity

Message integrity means purity, which means the message is protected from unauthorized access. Messages may be modified during transmission by active attackers. A man-in-the-middle attack could be possible if the payload isn't protected with proper security parameters. The neuropil protocol uses Authentication Encryption with Associated Data (AEAD) algorithms. The algorithm is executed by xchacha20 for encrypted ciphertext and Poly1305 for message authentication code (MAC). After encryption, the packet adds the MAC as an authentication tag to the message. Neuropil supports symmetric ciphers. A single key is computed for accessing the message. So it is difficult for an MITM attacker to modify the cipher text. MAC will be broken if an attacker is able to insert new information. The message with an encryption mechanism is presented in our model with the Verifpal tool.

5.4.3 Authentication

During communication, Neuropil raises concerns about authentication. It ensures the authenticity of identities through their signatures that are part of the token. The signature is verified with the publishing public key, but it is generated based on the private key. Adversary needs the private key if he would like to modify the token or re-create the signature. He can't obtain it until he has access to the internal storage of the system. Otherwise, the recipient decrypts the signature with a copy of the authenticated public and compares the decrypted information with the token. If it matches, he declares that the sender is genuine. It also depends on the fingerprint that will be discussed in Non-repudiation.

In unicast communication, authentication is processed until an encrypted data channel is established for a specific subject. End-to-end communication is formed only between genuine users. Since Neuropil supports self-sign signatures, Verifpal has already analyzed the signature with its built-in signature algorithm.

5.4.4 Non Repudiation

Repudiation refers to the capability of the user to deny his originated transaction. In neuropil cybermesh, the message provides non-repudiation through the fingerprint of the header field. The fingerprint is structured by the token that carries the signatures. After verification of the signature, the obtained information is applied to the hash function. The hash value is compared with a fingerprint. The recipient accepts the message if they are matched. Otherwise, it is dropped, and the processing also provides message integrity. Moreover, in E2E communication, identity gains non-repudiation via MIT with interesting subjects. Because the purpose of MIT is to provide proof of no-reputation through an identity's fingerprint and an interested subject.

In Verifpal, we define non-repudiation by determining whether the initiator's received and generated subjects are equivalent.

5.4.5 Authorization

Accessing network permission depends on the authentication. Before sending the encrypted message, participant entities have to obtain authorization. Otherwise, a leave message is sent by DHT to terminate communication. The first nodes must have authorization to actively connect to the network.

The authorization of a node can be acquired if it can yield the long-term secret key. It is able to generate the secret key when it has a copy of another party's public key. The public key authentication will be verified through signature checking.

For end-to-end communication, identities are authorized through the symmetric ephemeral key. To obtain EK, identities have to authenticate each other via MIT signature with a subscribed subject.

5.4.6 Accounting

Accounting is for tracking user activities while accessing the network. It marks transferred data and the duration of network access. [64] In Neuropil, the activities of users can be tracked throughout time, so the authenticated user's actions are recorded in the network. The mechanism works against unauthorized access to any specific user action.

5.4.7 Freshness

A replay attack is possible if the protocol doesn't provide freshness in message transmission. Neuropil preserves the freshness through uuid, timestamp, and nonce. Initially, identity could check the nonce before decryption. The nonce is formed with a random value that isn't reused. The recipient drops the message if he receives cipher text with the same random value.

5.4.8 Key Mechanism

Neuropil utilizes symmetric and asymmetric keys. A signature is considered an asymmetric cipher, and a message for end-to-end communication (E2E) is a symmetric cipher. The public key is visible to everyone in the network. Anyone can verify the signature with the authenticated public key, but it is generated from a private key, which is a hidden ephemeral key. For TLS, the secret key isn't pre-shared, it is computed from an ephemeral private key. For ALS, it is considered a shared ephemeral key.

5.4.9 Perfect Forward Secrecy

Symmetric ciphering using TLS1.3 ensures PFS. The long-term secret key is computed from a stored ephemeral private key using the DHKE algorithm. The public key is known to the attacker, but he can't discover the secret key. He needs the private keys. Application layer security supports ephemeral keys that consist of random values. It is generated according to the subject. For every subject, a new ephemeral key is produced, and its validity depends on the session of the subject. So it won't reveal past session keys if it is compromised. EK is independent, and it isn't computed from long-term secret keys. The message will still be secured even if the long-secret key is compromised.

5.5 Security Properties Verified by Verifpal

In the previous section, we discussed the security properties of the Neuropil protocol. Here we mention the security of what Verifpal is able to verify. There are a limited number of queries available. To define some of

the properties we prefer assuming, such as latency, It could be determined by the tool that captures live message transmission(for example, Wireshark), but in reality, it is not possible. Some properties are asked indirectly, for example availability, which depends on the same subject in the PH message. This query has been solved by the question of equivalence. The table shows the possible verifications by Verifpal.

Message	Security Properties	Possible the verification with tool
Handshake	Authentication	True
Join Message	Confidentiality & Authentication	True
DHT	Latency	False
Pheromone message	Availability	Partially True (using equivalence)
Message Intent Token	Non-repudiation & Authentication	Partially True (using equivalence)
Key-Agreement (Sk & EK)	Authorization & confidentiality	True
Userspace Message & end-to-end encryption	Integrity	True (queries with confidentiality)
	Perfect forward secrecy	True (by asking confidentiality of ephemeral key)
	Freshness(UUid and Timestamp)	True (only for uuid)

Table 8: Security property verification by Verifpal

6. Conclusion

We have formalized the symbolic model of the Neuropil protocol and shown the result of our investigation about its security properties. For analysis, we have utilized the formal verification tool Verifpal. We chose the tool because it is very new and has never been used with the neuropil protocol. The uniqueness of choosing the tool is its compatibility and relatability to a wide number of learners. Additionally, it specifies security properties through the queries that are more forwarded and not common in other tools. There are a limited number of built-up algorithms available in Verifpal. With these limitations, it analyzes the message that is transmitted through a network where an active attacker exists. It is like a simulation of real-life communication.

Neuropil ensures authentication in three steps. In primary stage node exchanges handshake message then transmit join message to introduce the identity and lastly shares MIT with the subject of pheromone message. To define the current state of node DHT message is sent. Every messaging scenario, we create individual model and analyze with Verifpal tool.

After an investigation of security properties, Verifpal proves that the message has maintained authentication from the initial steps due to carrying the signature. It shows that neuropil is able to avoid eavesdropping and other passive attacks for providing TLS. The messages are confined to high-level safeguards. Though E2E communication is established based on a public subject, the two-layer encryption method assures message secrecy. So the message will be confidential even if the past session key is compromised. Hence It achieves the perfect forward secrecy that Verifpal has ensured by answering the question of the confidentiality of ek (shown in the last scenario). The final result shows that the symmetric cipher is also secured from MITM attacks until it gains an ephemeral key. Otherwise, by using DHK for secret key generation, the AEAD method with Chacha20 and Poly1305 for message confidentiality and integrity, and Black2b for authentication identifies, Neuropil offers secure and stable communication with high-level security.

6.1 Limitation and Future Work

The main purpose of our thesis is to verify the security of the neuropil protocol by veripal. We are successful at 80% for analysis through specific queries in parallel, but some issues create obstacles during implementation, for example, the timing of execution. Every file is an individual and can't be imported into other files. It takes time to provide the output of the result for the model of each message. In the future, it could be developed. However, built-in algorithms are limited, which creates problems in forming the model. The number of security properties that are asked in queries is fixed. It requires improvement by adding other security queries such as availability and so on .

REFERENCES

1. Junaid Chaudhry, Uvais A. Qidwai, Robert G. Rittenhouse, Malrey Lee, "Vulnerabilities and verification of cryptographic protocols and their future in Wireless Body Area Networks", 2012 International Conference on Emerging Technologies, 08-09 October 2012, Islamabad, Pakistan, IEEE 06 December 2012.
<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6375433>
2. M. McLoone and J. V. McCanny, "System-on-chip architectures and implementations for private-key data encryption". New York, New York, USA: Kluwer Academic Pub, 2003
3. JASHKOTHARI1. "Cryptography and its Types."Geeksforgeeks.org, 22 Apr,2023
<https://www.geeksforgeeks.org/cryptography-and-its-types/>.
4. Cihangir Tezcan "Optimization of Advanced Encryption Standard on Graphics Processing Units"IEEE Access, Volume: 9, Page(s): 67315 - 67326, 04 May 2021, IEEE.
<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9422754>
5. P. Hamalainen, T. Alho, M. Hannikainen, and T. D. Hamalainen, "Design and implementation of low-area and low-power aes encryption hardware core," in 9th EUROMICRO Conference on Digital System Design (DSD'06), Aug 2006, pp. 577–583.
<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1690090>
6. Wahyu Wijaya Widiyanto, Dwi Iskandar, Sri Wulandari, Edy Susena, Edy Susanto, "Implementation Security Digital Signature Using Rivest Shamir Adleman (RSA) Algorithm As A Letter Validation And Distribution Validation System"2022-IEEE International Interdisciplinary Humanitarian Conference for Sustainability (IIHC-2022), November 18th & 19th 2022, pp.599-601, 17 March 2023, IEEE.
<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10060839>
7. M.E. Smid; D.K. Branstad, "Data Encryption Standard: past and future" Volume: 76, pp.550 - 559, May 1988, IEEE.
<https://ieeexplore.ieee.org/document/4441>
8. Brittany Chang "One of the biggest US insurance companies reportedly paid hackers \$40 million ransom after a cyberattack", May 22, 2021, 5:47 PM GMT+2.
<https://www.businessinsider.com/cna-financial-hackers-40-million-ransom-cyberattack-2021-5>
9. Shadia Nasralla, "Austria's FACC, hit by cyber fraud, fires CEO", Reuters, MAY 25, 2016 11:15 AM
<https://www.REUTERS.com/article/us-facc-ceo-idUSKCN0YG0ZF>

10. Vala Afshar, "Cybercrime threatens business growth. Take these steps to mitigate your risk", ZDNET, April 21, 2022.
<https://www.zdnet.com/article/cybercrime-can-be-the-biggest-threat-to-business-growth/>
11. IBM, "What is a cyberattack?", <https://www.ibm.com/topics/cyber-attack>
12. Michelle S. Henriques; Nagaraj K. Vernekar, "Using symmetric and asymmetric cryptography to secure communication between devices in IoT", 2017 International Conference on IoT and Application (ICIOT), 19 October 2017, IEEE.
13. Bruno Blanchet; Vincent Cheval; Véronique Cortier, "ProVerif with Lemmas, Induction, Fast Subsumption, and Much More", 2022 IEEE Symposium on Security and Privacy (SP), 27 July 2022, IEEE.
<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9833653>
14. K. Bhargavan, B. Blanchet, and N. Kobeissi. "Verified models and reference implementations for the TLS 1.3 standard candidate", In IEEE Symposium on Security and Privacy (S&P'17), pages 483–503, May 2017.
<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7958594>
15. C. Meadows, "Formal methods for cryptographic protocol analysis: emerging issues and trends," IEEE Journal on Selected Areas in Communications, vol. 21, no. 1, pp. 44–54, 2003.
<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1159654>
16. B. Blanchet, "Automatic verification of correspondences for security protocols", Journal of Computer Security, vol. 17, no. 4, pp. 363–434, July 31, 2008.
<https://bblanche.gitlabpages.inria.fr/proverif/publications/BlanchetJCS08.pdf>
17. L. Viganò "Automated security protocol analysis with the AVISPA tool," Electronic Notes Theoretical Computer Science, vol. 155, pp. 61–86, 12 May 2006.
<https://www.sciencedirect.com/science/article/pii/S1571066106001897>
18. C. J. F. Cremers, "The scyther tool: Verification, falsification, and analysis of security protocols," in Computer Aided Verification, 20th International Conference, CAV 2008, Princeton, NJ, USA, July 7-14, 2008, Proceedings, 2008, pp. 414–418.
19. S. Meier, B. Schmidt, C. Cremers, and D. A. Basin, "The TAMARIN prover for the symbolic analysis of security protocols," in Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings, 2013, pp. 696–701.

20. D. X. Song, S. Berezin, and A. Perrig, "Athena: A novel approach to efficient automatic security protocol analysis," *Journal of Computer Security*, vol. 9, no. 1/2, pp. 47–74, 2001.
21. C. Meadows, "The NRL protocol analyzer: An overview," *J. Log. Program.*, vol. 26, no. 2, pp. 113–131, 1996.
22. AVISPA Team, "AVISPA v1.0 User Manual", IST-2001-39252 Automated Validation of Internet Security Protocols and Applications, March 31, 2006, pp. 06-07.
http://people.irisa.fr/Thomas.Genet/Crypt/AVISPA_manual.pdf
23. Alexey Gotsman, Fabio Massacci, Marco Pistore, "Towards an Independent Semantics and Verification Technology for the HLPSL Specification Language", *Electronic Notes in Theoretical Computer Science*, Volume 135, Issue 1, 5 July 2005, pp.59-77
24. Cas Cremers, Sjouke Mauw, "Operational Semantics and Verification of Security Protocols", *Information Security and Cryptography*, Springer Heidelberg New York Dordrecht London, ISBN 978-3-540-78636-8 (eBook), Springer, 2012
25. C.J.F. Cremers, "Scyther: Unbounded Verification of Security Protocols", *Information Security*, ETH Zurich, "IFW C Haldeneggsteig 4, CH-8092 Zurich Switzerland. pp.01-02.
https://people.cispa.io/cas.cremers/downloads/papers/Cr2008-Scyther_tool.pdf
26. Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin, "The TAMARIN Prover for the Symbolic Analysis of Security Protocols ", *Institute of Information Security, ETH Zurich, Switzerland, IMDEA Software Institute, Madrid, Spain*, pp.4.
<https://beschmi.net/cav13.pdf>
27. Dr Nadim Kobeissi, Professor at NYU Paris and Director of Symbolic Software, "Verifpal: a new effort to make it easy to verify the security of Internet protocols", WHO'S NGI: NADIM KOBEISSI TELLS US MORE ABOUT VERIFPAL, NGI.
https://www.ngi.eu/blog/2019/10/04/whos_ngi_nadim_kobeissi_tells_more_about_verifpal/
28. Symbolic Software Blog, Insight into Symbolic Software's applied cryptography work and research, "One Year of Verifpal: Understanding Verifpal's Relationship With Cryptographic Protocol Security", 2.9.2020.
<https://symbolicsoft.wordpress.com/2020/09/02/one-year-of-verifpal-understanding-verifpals-relationship-with-cryptographic-protocol-security/>
29. Nadim Kobeissi¹, Georgio Nicolas¹ and Mukesh Tiwari², "Verifpal: Cryptographic Protocol Analysis for the Real World", *Symbolic Software University of Melbourne*.
<https://eprint.iacr.org/2019/971.pdf>

30. Jingjing Zhang, Lin Yang, Xianming Gao, Gaigai Tang, Jiyong Zhang, (Member, IEEE), and Qiang Wang, "Formal Analysis of QUIC Handshake Protocol Using Symbolic Model Checking", IEEE Access, Volume: 9, 18 January 2021, pp. 14836 - 14848, 18 January 2021.
<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9328313>
31. Omar Basem, Abrar Ullah, and Hani Ragab Hassen, "Stick: An End-to-End Encryption Protocol Tailored for Social Network Platforms", IEEE Transactions on Dependable and Secure Computing, Volume: 20, Issue: 2, 01 March-April 2023, pp. 1258 - 1269, 18 February 2022, IEEE.
<https://ieeexplore.ieee.org/document/9716793>
32. Mouna Nakkar, Member, IEEE, Riham AlTawy, Senior Member, IEEE, and Amr Youssef, Senior Member of IEEE, "GASE: A Lightweight Group Authentication Scheme With Key Agreement for Edge Computing Applications", IEEE Internet of Things Journal, Volume: 10, Issue: 1, 01 January 2023, pp. 840 - 854, 06 September 2022, IEEE.
<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9878136>
33. Nadim Kobeissi, "Verifpal User Manual first edition", Symbolic Software, May 31, 2022, pp. -7
<https://verifpal.com/>
34. Jean-Philippe Aumasson, Samuel Neves, Zooko Wilcox-O'Hearn, and Christian Winnerlein. BLAKE2: simpler, smaller, fast as MD5. In International Conference on Applied Cryptography and Network Security, pages 119–135. Springer, 2013.
35. Hugo Krawczyk, "Cryptographic extraction and key derivation: The HKDF scheme. In Advances in Cryptology (CRYPTO)", * IBM T.J. Watson Research Center, Hawthorne, New York, pages 631–648. IACR, 2010.
<https://eprint.iacr.org/2010/264.pdf>
36. Colin Percival and Simon Josefsson, "The scrypt Password-Based Key Derivation Function draft-josefsson-scrypt-kdf-00", Network Working Group, Internet-Draft, Intended status: Informational Expires: March 21, 2013.
<https://datatracker.ietf.org/doc/draft-josefsson-scrypt-kdf/00/>
37. Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich. "Argon2: new generation of memory hard functions for password hashing and other applications.", In 2016 IEEE European Symposium on Security and Privacy (EuroS&P), pages 292–302. IEEE, 2016.
38. Pi-lar GmbH "Neuro:pil Secure Interaction for things" Core Concepts.
https://neuropil.org/tutorial/core_concepts.html
39. EDUARDO BUETAS SANJUAN, ISMAEL ABAD CARDIEL, JOSE A. CERRADA, AND CARLOS CERRADA, "Message Queuing Telemetry Transport (MQTT) Security: A Cryptographic Smart Card Approach", IEEE Access, Volume: 8, pp. 115051 - 115062, 22 June 2020, IEEE.

- <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9121966>
40. Chris Brzuska; Eric Cornelissen; Konrad Kohbrok,"Security Analysis of the MLS Key Derivation",2022 IEEE Symposium on Security and Privacy (SP),pp.2535-2553,27 July 2022,IEEE.
<https://ieeexplore.ieee.org/document/9833678>
41. B. Beurdouche Inria & Mozilla, E. Rescorla Mozilla, E. Omara Google,S. Inguva, A. Duric Wire, "The Messaging Layer Security (MLS) Architecture" , Network Working Group, draft-ietf-mls-architecture-latest, 11 September 2023.
<https://messaginglayersecurity.rocks/mls-architecture/draft-ietf-mls-architecture.html#name-step-1-account-creation>
42. Md Sadek Ferdous, Farida Chowdhury, Madini O.Alassafi, "In Search of Self Sovereign Identity Leveraging Blockchain Technology", IEEE Access,Volume7,pp: 103063-103070,2019.
<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8776589>
43. Špela Čučko,Šeila Bećirović, Aida Kamišalić,Saša Mrdović, Muhamed Turkanović, "Towards the Classification of Self-Sovereign Identity Properties ", IEEE Access, Volume:10, 17 August 2022.
<https://ieeexplore.ieee.org/document/9858139>
44. Muhamad Haekal,Eliyani ,"Token-Based Authentication Using JSON Web Token on SIKASIR RESTful Web Service", 2016 International Conference on Informatics and Computing (ICIC),24 April 2017.
<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7905711>
45. Pi-lar GmbH "Neuro:pil Secure Interaction for things", Protocol token.
46. Tutorialspoint,"Distributed Hash Tables (DHTs)",10-Jan-2023.
<https://www.tutorialspoint.com/distributed-hash-tables-dhts>
47. Bibhash Roy,Suman Banik,Parthi Dey,Sugata Sanyal,Nabendu Chaki,"Ant Colony based Routing for Mobile Ad-Hoc Networks towards Improved Quality of Services ",Journal of Emerging Trends in Computing and Information Sciences, VOL. 3, NO. 1, January 2012.
https://www.tifr.res.in/~sanyal/papers/Ant_Colony_NC.pdf
48. Haijin Yan, Kang Li, Scott Watterson and David Lowenthal,"Improving Passive Estimation of TCP Round-Trip Times Using TCP Timestamps",2004 IEEE International Workshop on IP Operations and Management,pp.182,05 December 2005,IEEE.
<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1547614&tag=1>
49. Official website Verifpal: <https://verifpal.com/>
50. Verifpal Source Repository:<https://github.com/symbolicsoft/verifpal>.

51. Fabius Klemm, Sarunas Girdzijauskas, Jean-Yves Le Boudec, Karl Aberer, "On Routing in Distributed Hash Tables", School of Computer and Communication Sciences Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland.
<https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=5fe885364d55a1a108e96748b7c004c4d2de4d59>
52. Marc Fischlin; Christian Janson; Sogol Mazaheri, "Backdoored Hash Functions: Immunizing HMAC and HKDF", 2018 IEEE 31st Computer Security Foundations Symposium (CSF), 09-12 July 2018, pp. 110-111, 09 August 2018, IEEE .
<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8429299&tag=1>.
53. Muhammad Zulhamizan Ahmad, Alisa Rafiqah Adenan, Mohd Saufy Rohmad , *Yusnani Mohd Yussoff, "Performance Analysis of Secure MQTT Communication Protocol", 2023 19th IEEE International Colloquium on Signal Processing & Its Applications (CSPA), 03-04 March 2023, Kedah, Malaysia, pp. 225-229, 05 April 2023, IEEE.
<https://ieeexplore.ieee.org/document/10087603>
54. D. Cooper, S. Farrell, S. Boeyen, R. Housley, W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile" May 2008, Network Working Group, RFC 5280, May 2008.
<https://datatracker.ietf.org/doc/html/rfc5280> .
55. Filip Forsby , Martin Furuheid , Panos Papadimitratos , and Shahid Raza1, "Lightweight X.509 Digital Certificates for the Internet of Things", Third International Conference, InterIoT 2017, and Fourth International Conference, SaSeIoT 2017, Valencia, Spain, November 6-7, 2017, pp. 01, Proceedings July 2018.
https://www.researchgate.net/publication/326524588_Lightweight_X509_Digital_Certificates_for_the_Internet_of_Things_Third_International_Conference_InterIoT_2017_and_Fourth_International_Conference_SaSeIoT_2017_Valencia_Spain_November_6-7_2017_Proceedi.
56. P. Varalakshmi, Guhan B, Vignesh Siva P, Dhanush T4 and Saktheeswaran K, "IMPROVISING JSON WEB TOKEN AUTHENTICATION IN SDN", 2022 International Conference on Communication, Computing and Internet of Things (IC3IoT), pp. 01-08, 12 May 2022, IEEE.
<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9767873&tag=1>
57. Sovan Misra, "JWT Token: Lightweight, Token-Based Authentication", Securing your website is the goal of every developer, DZone, 29 May 2019.
<https://dzone.com/articles/jwt-token-lightweight-token-based-authentication#:~:text=This%20is%20a%20very%20compact,JWT%20token%2Dbased%20web%20authentication>
58. E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3", Internet Engineering Task Force (IETF), RFC 8446, August 2018.

<https://datatracker.ietf.org/doc/html/rfc8446>

59. Ripon Patgiri1, Sabuzima Nayak, and Samir Kumar Borgohain,"Preventing DDoS using Bloom Filter:A Survey", EAI Endorsed Transactions on scalable information system(Research Article),pp: 1-3,2018.

https://www.researchgate.net/publication/328332457_Preventing_DDoS_using_Bloom_Filter_A_Survey.

60. Ripon Patgiri;Sabuzima Nayak;Naresh Babu Muppalaneni "Is Bloom Filter a Bad Choice for Security and Privacy?", 2021 International Conference on Information Networking(ICOIN), pp:648-650,IEEE-2021.

61. Ankit Taparia, Saroj Kumar Panigrahy and Sanjay Kumar Jena,"Secure Key Exchange Using Enhanced Diffie-Hellman Protocol Based on String Comparison", 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), DOI: 10.1109/WiSPNET.2017.8299856,,pp.722-726, 22 February 2018,IEEE.

62. Hayder T. Assafli,Ivan A. Hashim,"Security Enhancement of AES-CBC and Its Performance Evaluation Using The Avalanche Effect ", 2020 3rd International Conference on Engineering Technology and its Applications (IICETA),DOI: 10.1109/IICETA50496.2020.9318803,pp.07-11,21 January 2021,IEEE.

<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9318803>

63. Ronaldo Serrano, Ckristian Duran, Trong-Thuc Hoang, Marco Sarmiento,"ChaCha20-Poly1305 Crypto Core Compatible with Transport Layer Security 1.3", 2021 18th International SoC Design Conference (ISOC),DOI: 10.1109/ISOC53507.2021.9614016,pp.17-18,25 November 2021,IEEE.
<https://ieeexplore.ieee.org/document/9614016>

64. Book Author(s): Crystal Panek, "Security Fundamentals", Wiley(OnlineLibrary), Lesson 2: Understanding authentication , authorization, and accounting, pp.33 - 109, 12 November 2019 (First edition)

<https://doi.org/10.1002/9781119650737.ch2>

65. Chenxin Zhang,Baixiang Fan,Shuo Li,Yongfeng Lin,Jin He,Yaqiang Gong,Bo Yin,"Tag-Based Trust Evaluation In Zero Trust Architecture", 2022 4th International Academic Exchange Conference on Science and Technology Innovation (IAECST),DOI:10.1109/IAECST57965.2022.10062213, pp.772-776, 17 March 2023,IEEE

<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1006221>

APPENDIX

File Name:Handshake_Message.vp

attacker[active]

```
principal NA[
  knows private na
  gna = G^na
  generates hostname1, port1
  data=CONCAT(hostname1, port1)
  sign_NA = SIGN(na,data)
]
```

```
principal NB[
  knows private nb
  gnb = G^nb
  generates hostname2, port2
  messg=CONCAT(hostname2, port2)
  sign_NB = SIGN(nb,messg)
]
```

NA->NB:[gna],sign_NA,data

```
principal NB[
  _ = SIGNVERIF(gna,data,sign_NA)?
]
NB->NA:[gnb],sign_NB,messg
```

```
principal NA[
  _ = SIGNVERIF(gnb,messg,sign_NB)?
]
queries[
  authentication? NA->NB: sign_NA
  authentication? NB->NA: sign_NB
  confidentiality? data
  confidentiality? messg
  confidentiality? hostname1
]
```

The Result

Run In terminal # verifpal verify Handshake_Message.vp

Verifpal 0.26.1 - <https://verifpal.com>

Warning • Verifpal is Beta software.

Verifpal • Parsing model 'Handshake_Message.vp'...

Verifpal • Verification initiated for 'Handshake_Message.vp' at 11:07:48 AM.

Info • Attacker is configured as active.

Info • Running at phase 0.

Analysis • Constructed skeleton SIGNVERIF($G^{\wedge}nil$, CONCAT(nil , nil), SIGN(nil , CONCAT(nil , nil))) based on SIGNVERIF($G^{\wedge}na$, CONCAT($hostname1$, $port1$), SIGN(na , CONCAT($hostname1$, $port1$)))?.

Result • confidentiality? data —

data (CONCAT($hostname1$, $port1$)) is obtained by Attacker.

Result • confidentiality? messg —

messg (CONCAT($hostname2$, $port2$)) is obtained by Attacker.

Deduction • $hostname1$ obtained as a concatenated fragment of CONCAT($hostname1$, $port1$).

Deduction • $port1$ obtained as a concatenated fragment of CONCAT($hostname1$, $port1$).

Result • confidentiality? $hostname1$ —

$hostname1$ ($hostname1$) is obtained by Attacker.

Deduction • $hostname2$ obtained as a concatenated fragment of CONCAT($hostname2$, $port2$).

Deduction • $port2$ obtained as a concatenated fragment of CONCAT($hostname2$, $port2$).

Analysis • Initializing Stage 1 mutation map for Na...

Analysis • Initializing Stage 1 mutation map for Nb...

Analysis • Initializing Stage 3 mutation map for Na...

Analysis • Initializing Stage 2 mutation map for Na...

Deduction • Output of SIGN(nil , CONCAT($hostname2$, $port2$)) obtained by reconstructing with nil , CONCAT($hostname2$, $port2$). (Analysis 73)

Deduction • Output of SIGN(nil , CONCAT($hostname1$, $port1$)) obtained by reconstructing with nil , CONCAT($hostname1$, $port1$). (Analysis 73)

Deduction • Output of SIGN($hostname1$, CONCAT($hostname1$, $port1$)) obtained by reconstructing with $hostname1$, CONCAT($hostname1$, $port1$). (Analysis 77)

Deduction • Output of SIGN($hostname1$, CONCAT($hostname2$, $port2$)) obtained by reconstructing with $hostname1$, CONCAT($hostname2$, $port2$). (Analysis 77)

Deduction • Output of SIGN($hostname2$, CONCAT($hostname1$, $port1$)) obtained by reconstructing with $hostname2$, CONCAT($hostname1$, $port1$). (Analysis 78)

Deduction • Output of SIGN($port1$, CONCAT($hostname2$, $port2$)) obtained by reconstructing with $port1$, CONCAT($hostname2$, $port2$). (Analysis 78)

Deduction • Output of SIGN($hostname2$, CONCAT($hostname2$, $port2$)) obtained by reconstructing with $hostname2$, CONCAT($hostname2$, $port2$). (Analysis 78)

Deduction • Output of SIGN($port1$, CONCAT($hostname1$, $port1$)) obtained by reconstructing with $port1$, CONCAT($hostname1$, $port1$). (Analysis 78)

Deduction • Output of SIGN($port2$, CONCAT($hostname1$, $port1$)) obtained by reconstructing with $port2$, CONCAT($hostname1$, $port1$). (Analysis 78)

Deduction • Output of SIGN($port2$, CONCAT($hostname2$, $port2$)) obtained by reconstructing with $port2$, CONCAT($hostname2$, $port2$). (Analysis 83)

Analysis • Initializing Stage 2 mutation map for Nb...

Deduction • Output of CONCAT(nil , nil) obtained by reconstructing with nil , nil . (Analysis 212)

Deduction • Output of CONCAT(nil , $hostname1$) obtained by reconstructing with nil , $hostname1$. (Analysis 227)

Deduction • Output of CONCAT(nil , $port1$) obtained by reconstructing with nil , $port1$. (Analysis 242)

Deduction • Output of CONCAT(nil , $hostname2$) obtained by reconstructing with nil , $hostname2$. (Analysis 257)

Deduction • Output of CONCAT(nil , $port2$) obtained by reconstructing with nil , $port2$. (Analysis 272)

Deduction • Output of CONCAT($hostname1$, nil) obtained by reconstructing with $hostname1$, nil . (Analysis 287)

Deduction • Output of CONCAT($hostname1$, $hostname1$) obtained by reconstructing with $hostname1$, $hostname1$. (Analysis 300)

Deduction • Output of CONCAT($hostname1$, $hostname2$) obtained by reconstructing with $hostname1$, $hostname2$. (Analysis 330)

Deduction • Output of CONCAT($hostname1$, $port2$) obtained by reconstructing with $hostname1$, $port2$. (Analysis 346)

Deduction • Output of CONCAT($port1$, nil) obtained by reconstructing with $port1$, nil . (Analysis 359)

Deduction • Output of CONCAT($port1$, $hostname1$) obtained by reconstructing with $port1$, $hostname1$. (Analysis 366)

Deduction • Output of CONCAT(port1, port1) obtained by reconstructing with port1, port1. (Analysis 392)

Deduction • Output of CONCAT(port1, hostname2) obtained by reconstructing with port1, hostname2. (Analysis 407)

Deduction • Output of CONCAT(port1, port2) obtained by reconstructing with port1, port2. (Analysis 421)

Deduction • Output of CONCAT(hostname2, nil) obtained by reconstructing with hostname2, nil. (Analysis 436)

Deduction • Output of CONCAT(hostname2, hostname1) obtained by reconstructing with hostname2, hostname1. (Analysis 450)

Deduction • Output of CONCAT(hostname2, port1) obtained by reconstructing with hostname2, port1. (Analysis 466)

Deduction • Output of CONCAT(hostname2, hostname2) obtained by reconstructing with hostname2, hostname2. (Analysis 473)

Deduction • Output of CONCAT(port2, nil) obtained by reconstructing with port2, nil. (Analysis 499)

Deduction • Output of CONCAT(port2, hostname1) obtained by reconstructing with port2, hostname1. (Analysis 524)

Deduction • Output of CONCAT(port2, port1) obtained by reconstructing with port2, port1. (Analysis 540)

Deduction • Output of CONCAT(port2, hostname2) obtained by reconstructing with port2, hostname2. (Analysis 557)

Deduction • Output of CONCAT(port2, port2) obtained by reconstructing with port2, port2. (Analysis 571)

Analysis • Initializing Stage 3 mutation map for Nb...

Analysis • Initializing Stage 5 mutation map for Na...

Analysis • Initializing Stage 4 mutation map for Na...

Deduction • Output of SIGN(nil, CONCAT(nil, nil)) obtained by reconstructing with nil, CONCAT(nil, nil). (Analysis 1070)

Deduction • Output of SIGN(nil, CONCAT(nil, hostname1)) obtained by reconstructing with nil, CONCAT(nil, hostname1). (Analysis 1077)

Deduction • Output of SIGN(nil, CONCAT(nil, port1)) obtained by reconstructing with nil, CONCAT(nil, port1). (Analysis 1079)

Deduction • Output of SIGN(nil, CONCAT(nil, hostname2)) obtained by reconstructing with nil, CONCAT(nil, hostname2). (Analysis 1081)

Deduction • Output of SIGN(nil, CONCAT(port1, nil)) obtained by reconstructing with nil, CONCAT(port1, nil). (Analysis 1079)

Deduction • Output of SIGN(nil, CONCAT(port1, hostname1)) obtained by reconstructing with nil, CONCAT(port1, hostname1). (Analysis 1080)

Deduction • Output of SIGN(nil, CONCAT(hostname1, nil)) obtained by reconstructing with nil, CONCAT(hostname1, nil). (Analysis 1081)

Deduction • Output of SIGN(nil, CONCAT(port1, port1)) obtained by reconstructing with nil, CONCAT(port1, port1). (Analysis 1079)

Deduction • Output of SIGN(nil, CONCAT(nil, port2)) obtained by reconstructing with nil, CONCAT(nil, port2). (Analysis 1087)

Deduction • Output of SIGN(nil, CONCAT(port2, nil)) obtained by reconstructing with nil, CONCAT(port2, nil). (Analysis 1093)

Deduction • Output of SIGN(nil, CONCAT(port1, hostname2)) obtained by reconstructing with nil, CONCAT(port1, hostname2). (Analysis 1094)

Deduction • Output of SIGN(nil, CONCAT(port2, hostname2)) obtained by reconstructing with nil, CONCAT(port2, hostname2). (Analysis 1094)

Deduction • Output of SIGN(nil, CONCAT(port1, port2)) obtained by reconstructing with nil, CONCAT(port1, port2). (Analysis 1094)

Deduction • Output of SIGN(nil, CONCAT(port2, port1)) obtained by reconstructing with nil, CONCAT(port2, port1). (Analysis 1095)

Deduction • Output of SIGN(nil, CONCAT(port2, port2)) obtained by reconstructing with nil, CONCAT(port2, port2). (Analysis 1094)

Deduction • Output of SIGN(hostname1, CONCAT(nil, port1)) obtained by reconstructing with hostname1, CONCAT(nil, port1). (Analysis 1096)

Deduction • Output of SIGN(hostname1, CONCAT(hostname1, hostname1)) obtained by reconstructing with hostname1, CONCAT(hostname1, hostname1). (Analysis 1097)

Deduction • Output of SIGN(nil, CONCAT(hostname2, hostname1)) obtained by reconstructing with nil, CONCAT(hostname2, hostname1). (Analysis 1097)

Deduction • Output of SIGN(nil, CONCAT(port2, hostname1)) obtained by reconstructing with nil, CONCAT(port2, hostname1). (Analysis 1097)

Deduction • Output of SIGN(nil, CONCAT(hostname1, hostname2)) obtained by reconstructing with nil, CONCAT(hostname1, hostname2). (Analysis 1097)

Deduction • Output of SIGN(nil, CONCAT(hostname2, port1)) obtained by reconstructing with nil, CONCAT(hostname2, port1). (Analysis 1097)

Deduction • Output of SIGN(hostname1, CONCAT(port1, nil)) obtained by reconstructing with hostname1, CONCAT(port1, nil). (Analysis 1097)

Deduction • Output of SIGN(nil, CONCAT(hostname1, hostname1)) obtained by reconstructing with nil, CONCAT(hostname1, hostname1). (Analysis 1098)

Deduction • Output of SIGN(nil, CONCAT(hostname2, nil)) obtained by reconstructing with nil, CONCAT(hostname2, nil). (Analysis 1111)

Deduction • Output of SIGN(nil, CONCAT(hostname2, hostname2)) obtained by reconstructing with nil, CONCAT(hostname2, hostname2). (Analysis 1115)

Deduction • Output of SIGN(hostname1, CONCAT(nil, port2)) obtained by reconstructing with hostname1, CONCAT(nil, port2). (Analysis 1118)

Deduction • Output of SIGN(hostname1, CONCAT(port1, hostname1)) obtained by reconstructing with hostname1, CONCAT(port1, hostname1). (Analysis 1119)

Deduction • Output of SIGN(nil, CONCAT(hostname1, port2)) obtained by reconstructing with nil, CONCAT(hostname1, port2). (Analysis 1119)

Deduction • Output of SIGN(hostname1, CONCAT(nil, hostname1)) obtained by reconstructing with hostname1, CONCAT(nil, hostname1). (Analysis 1119)

Deduction • Output of SIGN(hostname1, CONCAT(nil, hostname2)) obtained by reconstructing with hostname1, CONCAT(nil, hostname2). (Analysis 1126)

Deduction • Output of SIGN(hostname1, CONCAT(nil, nil)) obtained by reconstructing with hostname1, CONCAT(nil, nil). (Analysis 1128)

Deduction • Output of SIGN(hostname1, CONCAT(port1, port1)) obtained by reconstructing with hostname1, CONCAT(port1, port1). (Analysis 1130)

Appendix

Deduction • Output of SIGN(hostname2, CONCAT(nil, hostname2)) obtained by reconstructing with hostname2, CONCAT(nil, hostname2). (Analysis 1328)

Deduction • Output of SIGN(hostname2, CONCAT(hostname2, port1)) obtained by reconstructing with hostname2, CONCAT(hostname2, port1). (Analysis 1339)

Analysis • Initializing Stage 4 mutation map for Nb...

Analysis • Initializing Stage 5 mutation map for Nb...

Analysis • Initializing Stage 6 mutation map for Na...

Analysis • Initializing Stage 6 mutation map for Nb...

Stage 6, Analysis 58000...

Verifpal • Verification completed for 'Handshake_Message.vp' at 11:07:52 AM.

Verifpal • Summary of failed queries will follow.

Result • confidentiality? data —

data (CONCAT(hostname1, port1)) is obtained by Attacker.

Result • confidentiality? messg —

messg (CONCAT(hostname2, port2)) is obtained by Attacker.

Result • confidentiality? hostname1 —

hostname1 (hostname1) is obtained by Attacker.

Verifpal • Thank you for using Verifpal.

File Name: Join_Message1.vp

attacker[active]

principal Alice[

 knows private nt,it,sa

 generates a

$ga = G^a$

$nfp = \text{HASH}(nt)$

]

principal NA[

 generates na

$gna = G^{na}$

]

principal NB[

 generates nb

$gnb = G^{nb}$

]

NA->NB :[gna]

NB->NA :[gnb]

```
principal NA[
  knows private nt,it,c1,sa
  ifp=HASH(it)
]
```

```
principal Alice [
  js1 = SIGN(a, nfp)
  j = CONCAT(ga,js1,nfp)
  j0 = AEAD_ENC(sa,j,nil)
]
Alice-> NA:[j0]
```

```
principal NA[
  s = gnb^na
  j0_de = AEAD_DEC(sa,j0,nil)
  j1 = AEAD_ENC(s,j0_de,c1)
]
NA-> NB :[j1],c1
```

```
principal NB[
  s1 = gna^nb
  j1_de = AEAD_DEC(s1,j1,c1)
  ga_js1_nfp_ = SPLIT(j1_de)
  _ = SIGNVERIF(ga_nfp_js1_)?
]
```

```
principal NA[
  js2 =SIGN(na,ifp)
  js3 = CONCAT(js2,ifp)
  j2 = AEAD_ENC(s,js3,c1)
]
NA-> NB :[j2]
```

```
principal NB[
  j2_de = AEAD_DEC(s1,j2,c1)?
  js2_ifp_ = SPLIT(j2_de)
  _ = SIGNVERIF(gna,ifp_js2_)?
]
```

```
queries[
  authentication? NA->NB:j1
  authentication? NA->NB:j2
  equivalence? s,s1
  confidentiality? ifp
]
```


confidentiality? nt
confidentiality? nfp

]

The Result

Run In terminal # verifpal verify Join_Message1.vp

shabnaz-dev% verifpal verify rough/Join_Message1.vp

Verifpal 0.26.1 - <https://verifpal.com>

Warning • Verifpal is Beta software.

Verifpal • Parsing model 'Join_Message1.vp'...

Verifpal • Verification initiated for 'Join_Message1.vp' at 11:15:52 AM.

Info • Attacker is configured as active.

Info • Running at phase 0.

Analysis • Constructed skeleton HASH(nil) based on HASH(nt).

Analysis • Constructed skeleton SIGN(nil, HASH(nil)) based on SIGN(a, HASH(nt)).

Analysis • Constructed skeleton CONCAT(G^nil, SIGN(nil, HASH(nil)), HASH(nil)) based on CONCAT(G^a, SIGN(a, HASH(nt)), HASH(nt)).

Analysis • Constructed skeleton AEAD_DEC(nil, AEAD_ENC(nil, CONCAT(G^nil, SIGN(nil, HASH(nil)), HASH(nil)), nil, nil) based on AEAD_DEC(sa, AEAD_ENC(sa, CONCAT(G^a, SIGN(a, HASH(nt)), HASH(nt)), nil, nil).

Analysis • Constructed skeleton AEAD_DEC(G^nil, AEAD_ENC(G^nil, AEAD_DEC(nil, AEAD_ENC(nil, CONCAT(G^nil, SIGN(nil, HASH(nil)), nil, nil), nil, nil), nil) based on AEAD_DEC(G^na^nb, AEAD_ENC(G^nb^na, AEAD_DEC(sa, AEAD_ENC(sa, CONCAT(G^a, SIGN(a, HASH(nt)), HASH(nt)), nil, nil), c1), c1).

Analysis • Constructed skeleton SPLIT(AEAD_DEC(G^nil, AEAD_ENC(G^nil, AEAD_DEC(nil, AEAD_ENC(nil, CONCAT(G^nil, SIGN(nil, HASH(nil)), HASH(nil)), nil, nil), nil, nil) based on SPLIT(AEAD_DEC(G^na^nb, AEAD_ENC(G^nb^na, AEAD_DEC(sa, AEAD_ENC(sa, CONCAT(G^a, SIGN(a, HASH(nt)), HASH(nt)), nil, nil), c1), c1)).

Analysis • Constructed skeleton SPLIT(AEAD_DEC(G^nil, AEAD_ENC(G^nil, AEAD_DEC(nil, AEAD_ENC(nil, CONCAT(G^nil, SIGN(nil, HASH(nil)), HASH(nil)), nil, nil), nil, nil) based on SPLIT(AEAD_DEC(G^na^nb, AEAD_ENC(G^nb^na, AEAD_DEC(sa, AEAD_ENC(sa, CONCAT(G^a, SIGN(a, HASH(nt)), HASH(nt)), nil, nil), c1), c1)).

Analysis • Constructed skeleton SPLIT(AEAD_DEC(G^nil, AEAD_ENC(G^nil, AEAD_DEC(nil, AEAD_ENC(nil, CONCAT(G^nil, SIGN(nil, HASH(nil)), HASH(nil)), nil, nil), nil, nil) based on SPLIT(AEAD_DEC(G^na^nb, AEAD_ENC(G^nb^na, AEAD_DEC(sa, AEAD_ENC(sa, CONCAT(G^a, SIGN(a, HASH(nt)), HASH(nt)), nil, nil), c1), c1)).

Analysis • Constructed skeleton SIGNVERIF(SPLIT(AEAD_DEC(G^nil, AEAD_ENC(G^nil, AEAD_DEC(nil, AEAD_ENC(nil, CONCAT(G^nil, SIGN(nil, HASH(nil)), HASH(nil)), nil, nil), nil, nil), SPLIT(AEAD_DEC(G^nil, AEAD_ENC(G^nil, AEAD_DEC(nil, AEAD_ENC(nil, CONCAT(G^nil, SIGN(nil, HASH(nil)), HASH(nil)), nil, nil), nil, nil), SPLIT(AEAD_DEC(G^nil, AEAD_ENC(G^nil, AEAD_DEC(nil, AEAD_ENC(nil, CONCAT(G^nil, SIGN(nil, HASH(nil)), HASH(nil)), nil, nil), nil, nil), nil))) based on SIGNVERIF(SPLIT(AEAD_DEC(G^na^nb, AEAD_ENC(G^nb^na, AEAD_DEC(sa, AEAD_ENC(sa, CONCAT(G^a, SIGN(a, HASH(nt)), HASH(nt)), nil, nil), c1), c1)), SPLIT(AEAD_DEC(G^na^nb, AEAD_ENC(G^nb^na, AEAD_DEC(sa, AEAD_ENC(sa, CONCAT(G^a, SIGN(a, HASH(nt)), HASH(nt)), nil, nil), c1), c1)), SPLIT(AEAD_DEC(G^na^nb, AEAD_ENC(G^nb^na, AEAD_DEC(sa, AEAD_ENC(sa, CONCAT(G^a, SIGN(a, HASH(nt)), HASH(nt)), nil, nil), c1), c1)))?.

Analysis • Constructed skeleton CONCAT(SIGN(nil, HASH(nil)), HASH(nil)) based on CONCAT(SIGN(na, HASH(it)), HASH(it)).

Analysis • Constructed skeleton AEAD_DEC(G^nil, AEAD_ENC(G^nil, CONCAT(SIGN(nil, HASH(nil)), HASH(nil)), nil, nil) based on AEAD_DEC(G^na^nb, AEAD_ENC(G^nb^na, CONCAT(SIGN(na, HASH(it)), HASH(it)), c1), c1)?.

Analysis • Constructed skeleton SPLIT(AEAD_DEC(G^nil, AEAD_ENC(G^nil, CONCAT(SIGN(nil, HASH(nil)), HASH(nil)), nil, nil) based on SPLIT(AEAD_DEC(G^na^nb, AEAD_ENC(G^nb^na, CONCAT(SIGN(na, HASH(it)), HASH(it)), c1), c1)?).

Analysis • Constructed skeleton SPLIT(AEAD_DEC(G^nil, AEAD_ENC(G^nil, CONCAT(SIGN(nil, HASH(nil)), HASH(nil)), nil, nil) based on SPLIT(AEAD_DEC(G^na^nb, AEAD_ENC(G^nb^na, CONCAT(SIGN(na, HASH(it)), HASH(it)), c1), c1)?).

Analysis • Constructed skeleton SIGNVERIF(G^nil, SPLIT(AEAD_DEC(G^nil, AEAD_ENC(G^nil, CONCAT(SIGN(nil, HASH(nil)), HASH(nil)), nil, nil), SPLIT(AEAD_DEC(G^nil, AEAD_ENC(G^nil, CONCAT(SIGN(nil, HASH(nil)), HASH(nil)), nil, nil))) based on SIGNVERIF(G^na, SPLIT(AEAD_DEC(G^na^nb, AEAD_ENC(G^nb^na, CONCAT(SIGN(na, HASH(it)), HASH(it)), c1), c1)?), SPLIT(AEAD_DEC(G^na^nb, AEAD_ENC(G^nb^na, CONCAT(SIGN(na, HASH(it)), HASH(it)), c1), c1)?)?.

84

Deduction • Output of $\text{AEAD_ENC}(c1, \text{CONCAT}(G^{\text{na}}, \text{SIGN}(\text{nil}, \text{HASH}(\text{nil})), \text{HASH}(c1)), \text{nil})$ obtained by reconstructing with $c1, \text{CONCAT}(G^{\text{na}}, \text{SIGN}(\text{nil}, \text{HASH}(\text{nil})), \text{HASH}(c1)), \text{nil}$. (Analysis 130)

Deduction • Output of $\text{AEAD_ENC}(c1, \text{CONCAT}(G^{\text{na}}, \text{SIGN}(\text{nil}, \text{HASH}(\text{nil})), \text{HASH}(c1)), c1)$ obtained by reconstructing with $c1, \text{CONCAT}(G^{\text{na}}, \text{SIGN}(\text{nil}, \text{HASH}(\text{nil})), \text{HASH}(c1)), c1$. (Analysis 130)

Deduction • Output of $\text{AEAD_ENC}(c1, \text{CONCAT}(G^{\text{na}}, \text{SIGN}(c1, \text{HASH}(\text{nil})), \text{HASH}(\text{nil})), \text{nil})$ obtained by reconstructing with $c1, \text{CONCAT}(G^{\text{na}}, \text{SIGN}(c1, \text{HASH}(\text{nil})), \text{HASH}(\text{nil})), \text{nil}$. (Analysis 131)

Deduction • Output of $\text{AEAD_ENC}(c1, \text{CONCAT}(G^{\text{na}}, \text{SIGN}(c1, \text{HASH}(\text{nil})), \text{HASH}(\text{nil})), c1)$ obtained by reconstructing with $c1, \text{CONCAT}(G^{\text{na}}, \text{SIGN}(c1, \text{HASH}(\text{nil})), \text{HASH}(\text{nil})), c1$. (Analysis 133)

Deduction • Output of $\text{AEAD_ENC}(c1, \text{CONCAT}(G^{\text{na}}, \text{SIGN}(c1, \text{HASH}(\text{nil})), \text{HASH}(c1)), \text{nil})$ obtained by reconstructing with $c1, \text{CONCAT}(G^{\text{na}}, \text{SIGN}(c1, \text{HASH}(\text{nil})), \text{HASH}(c1)), \text{nil}$. (Analysis 133)

Deduction • Output of $\text{AEAD_ENC}(c1, \text{CONCAT}(G^{\text{na}}, \text{SIGN}(c1, \text{HASH}(\text{nil})), \text{HASH}(c1)), c1)$ obtained by reconstructing with $c1, \text{CONCAT}(G^{\text{na}}, \text{SIGN}(c1, \text{HASH}(\text{nil})), \text{HASH}(c1)), c1$. (Analysis 135)

Deduction • Output of $\text{AEAD_ENC}(c1, \text{CONCAT}(G^{\text{nb}}, \text{SIGN}(\text{nil}, \text{HASH}(\text{nil})), \text{HASH}(c1)), \text{nil})$ obtained by reconstructing with $c1, \text{CONCAT}(G^{\text{nb}}, \text{SIGN}(\text{nil}, \text{HASH}(\text{nil})), \text{HASH}(c1)), \text{nil}$. (Analysis 138)

Deduction • Output of $\text{AEAD_ENC}(c1, \text{CONCAT}(G^{\text{nb}}, \text{SIGN}(\text{nil}, \text{HASH}(\text{nil})), \text{HASH}(c1)), c1)$ obtained by reconstructing with $c1, \text{CONCAT}(G^{\text{nb}}, \text{SIGN}(\text{nil}, \text{HASH}(\text{nil})), \text{HASH}(c1)), c1$. (Analysis 138)

Deduction • Output of $\text{AEAD_ENC}(c1, \text{CONCAT}(G^{\text{nb}}, \text{SIGN}(c1, \text{HASH}(\text{nil})), \text{HASH}(\text{nil})), \text{nil})$ obtained by reconstructing with $c1, \text{CONCAT}(G^{\text{nb}}, \text{SIGN}(c1, \text{HASH}(\text{nil})), \text{HASH}(\text{nil})), \text{nil}$. (Analysis 139)

Deduction • Output of $\text{AEAD_ENC}(c1, \text{CONCAT}(G^{\text{nb}}, \text{SIGN}(c1, \text{HASH}(\text{nil})), \text{HASH}(\text{nil})), c1)$ obtained by reconstructing with $c1, \text{CONCAT}(G^{\text{nb}}, \text{SIGN}(c1, \text{HASH}(\text{nil})), \text{HASH}(\text{nil})), c1$. (Analysis 140)

Deduction • Output of $\text{AEAD_ENC}(c1, \text{CONCAT}(G^{\text{nb}}, \text{SIGN}(c1, \text{HASH}(\text{nil})), \text{HASH}(c1)), c1)$ obtained by reconstructing with $c1, \text{CONCAT}(G^{\text{nb}}, \text{SIGN}(c1, \text{HASH}(\text{nil})), \text{HASH}(c1)), c1$. (Analysis 141)

Deduction • Output of $\text{AEAD_ENC}(c1, \text{CONCAT}(G^{\text{nb}}, \text{SIGN}(c1, \text{HASH}(\text{nil})), \text{HASH}(c1)), \text{nil})$ obtained by reconstructing with $c1, \text{CONCAT}(G^{\text{nb}}, \text{SIGN}(c1, \text{HASH}(\text{nil})), \text{HASH}(c1)), \text{nil}$. (Analysis 141)

Deduction • Output of $\text{AEAD_ENC}(c1, \text{CONCAT}(G^{\text{nil}}, \text{SIGN}(\text{nil}, \text{HASH}(\text{nil})), \text{HASH}(c1)), \text{nil})$ obtained by reconstructing with $c1, \text{CONCAT}(G^{\text{nil}}, \text{SIGN}(\text{nil}, \text{HASH}(\text{nil})), \text{HASH}(c1)), \text{nil}$. (Analysis 142)

Deduction • Output of $\text{AEAD_ENC}(c1, \text{CONCAT}(G^{\text{nil}}, \text{SIGN}(\text{nil}, \text{HASH}(\text{nil})), \text{HASH}(c1)), c1)$ obtained by reconstructing with $c1, \text{CONCAT}(G^{\text{nil}}, \text{SIGN}(\text{nil}, \text{HASH}(\text{nil})), \text{HASH}(c1)), c1$. (Analysis 142)

Deduction • Output of $\text{AEAD_ENC}(c1, \text{CONCAT}(G^{\text{nil}}, \text{SIGN}(c1, \text{HASH}(\text{nil})), \text{HASH}(\text{nil})), \text{nil})$ obtained by reconstructing with $c1, \text{CONCAT}(G^{\text{nil}}, \text{SIGN}(c1, \text{HASH}(\text{nil})), \text{HASH}(\text{nil})), \text{nil}$. (Analysis 144)

Deduction • Output of $\text{AEAD_ENC}(c1, \text{CONCAT}(G^{\text{nil}}, \text{SIGN}(c1, \text{HASH}(\text{nil})), \text{HASH}(c1)), \text{nil})$ obtained by reconstructing with $c1, \text{CONCAT}(G^{\text{nil}}, \text{SIGN}(c1, \text{HASH}(\text{nil})), \text{HASH}(c1)), \text{nil}$. (Analysis 146)

Deduction • Output of $\text{AEAD_ENC}(c1, \text{CONCAT}(G^{\text{nil}}, \text{SIGN}(c1, \text{HASH}(\text{nil})), \text{HASH}(\text{nil})), c1)$ obtained by reconstructing with $c1, \text{CONCAT}(G^{\text{nil}}, \text{SIGN}(c1, \text{HASH}(\text{nil})), \text{HASH}(\text{nil})), c1$. (Analysis 147)

Deduction • Output of $\text{AEAD_ENC}(c1, \text{CONCAT}(G^{\text{nil}}, \text{SIGN}(c1, \text{HASH}(\text{nil})), \text{HASH}(c1)), c1)$ obtained by reconstructing with $c1, \text{CONCAT}(G^{\text{nil}}, \text{SIGN}(c1, \text{HASH}(\text{nil})), \text{HASH}(c1)), c1$. (Analysis 147)

Stage 6, Analysis 210...

Verifpal • Verification completed for 'Join_Message1.vp' at 11:15:52 AM.

Verifpal • All queries pass.

Verifpal • Thank you for using Verifpal.

File Name: Join_Message2.vp

attacker[active]

principal Bob[

 knows private ntB,itb,sb

 generates b

 gb = G^b

```

    nfpB= HASH(ntB)
]
principal NB[
    generates nb
    gnb = G^nb

]

principal NA[
    generates na
    gna = G^na

]

NB->NA :[gnb]
NA->NB :[gna]

principal NB[
    knows private ntB,itb,c2,sb
    ifpb=HASH(itb)
]

principal Bob [
    js1 = SIGN(b, nfpB)
    j = CONCAT(gb,js1,nfpB)
    j0 = AEAD_ENC(sb,j,nil)
]
Bob-> NB:[j0]

principal NB[
    s1 = gna^nb
    j0_de = AEAD_DEC(sb,j0,nil)
    j1 = AEAD_ENC(s1,j0_de,c2)
]
NB-> NA :[j1],c2

principal NA[
    s = gnb^na
    j1_de = AEAD_DEC(s,j1,c2)
    gb_,js1_,nfpB_ = SPLIT(j1_de)
    _ = SIGNVERIF(gb_,nfpB_,js1_)?
]
principal NB[

```

```
js2 =SIGN(nb,ifpb)
js3 = CONCAT(js2,ifpb)
j2 = AEAD_ENC(s1,js3,c2)
]
NB-> NA :[j2]

principal NA[
  j2_de = AEAD_DEC(s,j2,c2)?
  js2_,ifpb_ = SPLIT(j2_de)
  _ = SIGNVERIF(gnb,ifpb_,js2_)?
]
```

```
queries[
  authentication? NB->NA:j1
  authentication? NB->NA:j2
  equivalence? s,s1
  confidentiality? ifpb
  confidentiality? ntB
  confidentiality? nfpB
]
```

The Result

Run In terminal # verifpal verify Join_Message2.vp

Verifpal 0.26.1 - <https://verifpal.com>

Warning • Verifpal is Beta software.

Verifpal • Parsing model 'Join_Message2.vp'...

Verifpal • Verification initiated for 'Join_Message2.vp' at 11:28:25 AM.

Info • Attacker is configured as active.

Info • Running at phase 0.

Analysis • Constructed skeleton HASH(nil) based on HASH(ntb).

Analysis • Constructed skeleton SIGN(nil, HASH(nil)) based on SIGN(b, HASH(ntb)).

Analysis • Constructed skeleton CONCAT(G^nil, SIGN(nil, HASH(nil)), HASH(nil)) based on CONCAT(G^b, SIGN(b, HASH(ntb)), HASH(ntb)).

Analysis • Constructed skeleton AEAD_DEC(nil, AEAD_ENC(nil, CONCAT(G^nil, SIGN(nil, HASH(nil)), HASH(nil)), nil, nil) based on AEAD_DEC(sb, AEAD_ENC(sb, CONCAT(G^b, SIGN(b, HASH(ntb)), HASH(ntb)), nil, nil).

Analysis • Constructed skeleton AEAD_DEC(G^nil, AEAD_ENC(G^nil, AEAD_DEC(nil, AEAD_ENC(nil, CONCAT(G^nil, SIGN(nil, HASH(nil)), HASH(nil)), nil, nil), nil) based on AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, AEAD_DEC(sb, AEAD_ENC(sb, CONCAT(G^b, SIGN(b, HASH(ntb)), HASH(ntb)), nil, nil), c2), c2).

Analysis • Constructed skeleton SPLIT(AEAD_DEC(G^nil, AEAD_ENC(G^nil, AEAD_DEC(nil, AEAD_ENC(nil, CONCAT(G^nil, SIGN(nil, HASH(nil)), HASH(nil)), nil, nil), nil)) based on SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, AEAD_DEC(sb, AEAD_ENC(sb, CONCAT(G^b, SIGN(b, HASH(ntb)), HASH(ntb)), nil, nil), c2), c2)).

Analysis • Initializing Stage 5 mutation map for Na...

89

Deduction • Output of AEAD_ENC(c2, CONCAT(G^nil, SIGN(c2, HASH(nil)), HASH(nil)), c2) obtained by reconstructing with c2, CONCAT(G^nil, SIGN(c2, HASH(nil)), HASH(nil)), c2. (Analysis 146)

Deduction • Output of AEAD_ENC(c2, CONCAT(G^nil, SIGN(c2, HASH(nil)), HASH(c2)), c2) obtained by reconstructing with c2, CONCAT(G^nil, SIGN(c2, HASH(nil)), HASH(c2)), c2. (Analysis 148)

Deduction • Output of AEAD_ENC(c2, CONCAT(G^nil, SIGN(c2, HASH(nil)), HASH(c2)), nil) obtained by reconstructing with c2, CONCAT(G^nil, SIGN(c2, HASH(nil)), HASH(c2)), nil. (Analysis 148)

Stage 6, Analysis 210...

Verifpal • Verification completed for 'Join_Message2.vp' at 11:28:25 AM.

Verifpal • All queries pass.

Verifpal • Thank you for using Verifpal.

File Name: DHT_Message.vp

attacker[active]

principal NA[

generates nta,uuidA,hndsM

fpA= HASH(nta,nil,nil)

knows private na

gna = G^na

]

NA ->NB:[gna],[fpA]

principal NB[

generates ntB, seq0101,uuidB

fpB = HASH(ntB,nil,nil)

knows private nb

gnb = G^nb

]

NB ->NA:[gnb],[fpB]

principal NA[

shAB = gnb^na

]

principal NB[

shBA = gna^nb

]

NB ->NC:[fpB],[gnb]

principal NC[

generates ntC,subj_dhl ,seq01

fpC = HASH(ntC,nil,nil)

knows private nc

gnc = G^nc

shCB = gnb^nc

]

NC ->NB :[fpC],[gnc]

```
principal NB [
    shBC = gnc^nb
]
```

NC ->ND:[fpC],[gnc]

```
principal ND[
    generates ntD // fp = fingerprint of Della
    fpD= HASH(ntD,nil,nil)
    knows private nd
    gnd = G^nd
]
```

ND ->NC:[gnd],[fpD]

```
principal NC[
    dht_M = CONCAT(seq01,fpD) //dhk_M is Message of dhk
    dht = AEAD_ENC(shCB,dht_M,nil)
```

```
]
```

NC ->NB:[dht]

```
principal NB[
    dht_dec = AEAD_DEC(shBC,dht,nil)?
    subj_dhl_,seq01_,fpD_ = SPLIT(dht_dec)
    dht_fp = CONCAT(fpC,fpD_)
    dht_M1= CONCAT(subj_dhl_, seq01_,seq0101, dht_fp,uuidB) // adding Bob's sequence number,
add bob's fingerprint instead of carl
    dht01 = AEAD_ENC(shBA,dht_M1,nil)
    //dht01_m = MAC(shBA,dht01)
```

```
]
```

NB ->NA :[dht01] //,dht01_m

```
principal NA[
    //_ =ASSERT(MAC(shAB,dht01),dht01_m)?
    dht01_dec =AEAD_DEC(shAB,dht01,nil)?
    subj_dhl_0, seq01_0, seq0101_,dht_fp_,uuidB_=SPLIT(dht01_dec)
    ping =AEAD_ENC(shAB,uuidA,nil)
```

```
]
```

NA ->NB : [ping]

```
principal NA[
    fpC_0,fpD_0 = SPLIT(dht_fp_)
]
principal NB[
    ping_dec = AEAD_DEC(shBA,ping,nil)?
    ack= AEAD_ENC(shBA,ping_dec,nil)
    //ackM = MAC(shBA,ack)
]
NB ->NA : [ack] //,ackM
```

```
principal NA[
    //_=ASSERT(MAC(shAB,ack),ackM)?
    ack_dec = AEAD_DEC(shAB,ack,nil)?
    _= ASSERT(uuidA,ack_dec)?
]
```

NA ->ND:[hndsM]

```
queries[
    equivalence?shAB,shBA
    confidentiality?ack
    confidentiality?ping
    authentication?NB ->NA : ack
    confidentiality? uuidA
    freshness? uuidA
    equivalence?uuidA,ack_dec
]
```

The Result

Run In terminal # verifpal verify DHT_Message.vp

```
verifpal 0.26.1 - https://verifpal.com
Warning • Verifpal is Beta software.
Verifpal • Parsing model 'DHT.vp'...
2023/09/22 11:36:40 open rough/DHT.vp: no such file or directory
shabnaz-dev% verifpal verify rough/DHT_Message.vp
Verifpal 0.26.1 - https://verifpal.com
Warning • Verifpal is Beta software.
Verifpal • Parsing model 'DHT_Message.vp'...
Verifpal • Verification initiated for 'DHT_Message.vp' at 11:36:53 AM.
Info • Attacker is configured as active.
Info • Running at phase 0.
```

Analysis • Constructed skeleton AEAD_DEC(G^{nil} , AEAD_ENC(G^{nil} , CONCAT(nil , HASH(nil , nil , nil)), nil), nil) based on AEAD_DEC($G^{\text{nc}^{\text{nb}}}$, AEAD_ENC($G^{\text{nb}^{\text{nc}}}$, CONCAT(seq01 , HASH(ntd , nil , nil)), nil), nil)?.

Analysis • Constructed skeleton `SPLIT(AEAD_DEC(G^nil, AEAD_ENC(G^nil, CONCAT(nil, HASH(nil, nil, nil)), nil, nil))` based on `SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil, nil)?).`

Analysis • Constructed skeleton $\text{CONCAT}(\text{HASH}(\text{nil}, \text{nil}, \text{nil}), \text{SPLIT}(\text{AEAD_DEC}(G^{\text{nil}}, \text{AEAD_ENC}(G^{\text{nil}}, \text{CONCAT}(\text{nil}, \text{HASH}(\text{nil}, \text{nil}, \text{nil})), \text{nil}), \text{nil})))$ based on $\text{CONCAT}(\text{HASH}(\text{ntc}, \text{nil}, \text{nil}), \text{SPLIT}(\text{AEAD_DEC}(G^{\text{nc}^{\text{nb}}}, \text{AEAD_ENC}(G^{\text{nb}^{\text{nc}}}, \text{CONCAT}(\text{seq01}, \text{HASH}(\text{ntd}, \text{nil}, \text{nil})), \text{nil}), \text{nil})))$).

Analysis • Constructed skeleton AEAD_DEC($G^{\wedge}nil$, AEAD_ENC($G^{\wedge}nil$, CONCAT(SPLIT(AEAD_DEC($G^{\wedge}nil$, AEAD_ENC($G^{\wedge}nil$, CONCAT(nil , HASH(nil , nil , nil)), nil), nil)), SPLIT(AEAD_DEC($G^{\wedge}nil$, AEAD_ENC($G^{\wedge}nil$, CONCAT(nil , HASH(nil , nil , nil)), nil), nil)), nil , CONCAT(HASH(nil , nil , nil), SPLIT(AEAD_DEC($G^{\wedge}nil$, AEAD_ENC($G^{\wedge}nil$, CONCAT(nil , HASH(nil , nil , nil)), nil), nil))), nil), nil) based on AEAD_DEC($G^{\wedge}nb^{\wedge}na$, AEAD_ENC($G^{\wedge}na^{\wedge}nb$, CONCAT(SPLIT(AEAD_DEC($G^{\wedge}nc^{\wedge}nb$, AEAD_ENC($G^{\wedge}nb^{\wedge}nc$, CONCAT(seq01, HASH(ntd , nil , nil)), nil), nil ?), SPLIT(AEAD_DEC($G^{\wedge}nc^{\wedge}nb$, AEAD_ENC($G^{\wedge}nb^{\wedge}nc$, CONCAT(seq01, HASH(ntd , nil , nil)), nil), nil ?), seq0101, CONCAT(HASH(ntc , nil , nil), SPLIT(AEAD_DEC($G^{\wedge}nc^{\wedge}nb$, AEAD_ENC($G^{\wedge}nb^{\wedge}nc$, CONCAT(seq01, HASH(ntd , nil , nil)), nil), nil ?)), $uuidb$, nil), nil ?).

Analysis • Constructed skeleton `SPLIT(AEAD_DEC(G^nil, AEAD_ENC(G^nil, CONCAT(SPLIT(AEAD_DEC(G^nil, AEAD_ENC(G^nil, CONCAT(nil, HASH(nil, nil, nil)), nil), nil)), SPLIT(AEAD_DEC(G^nil, AEAD_ENC(G^nil, CONCAT(nil, HASH(nil, nil, nil)), nil), nil)), nil, CONCAT(HASH(nil, nil, nil), SPLIT(AEAD_DEC(G^nil, AEAD_ENC(G^nil, CONCAT(nil, HASH(nil, nil, nil)), nil), nil))), nil), nil), nil))` based on `SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), seq0101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?)), uuidb, nil, nil)?)`.

Analysis • Constructed skeleton `SPLIT(AEAD_DEC(G^nil, AEAD_ENC(G^nil, CONCAT(SPLIT(AEAD_DEC(G^nil, AEAD_ENC(G^nil, CONCAT(nil, HASH(nil, nil, nil)), nil), nil)), SPLIT(AEAD_DEC(G^nil, AEAD_ENC(G^nil, CONCAT(nil, HASH(nil, nil, nil)), nil), nil)), nil, CONCAT(HASH(nil, nil, nil), SPLIT(AEAD_DEC(G^nil, AEAD_ENC(G^nil, CONCAT(nil, HASH(nil, nil, nil)), nil), nil))), nil), nil), nil))` based on `SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), seq0101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?)), uuidb, nil, nil)?)`.

94

Analysis • Constructed skeleton SPLIT(SPLIT(AEAD_DEC(Gⁿnil, AEAD_ENC(Gⁿnil, CONCAT(SPLIT(AEAD_DEC(Gⁿnil, AEAD_ENC(Gⁿnil, CONCAT(nil, HASH(nil, nil, nil))), nil), nil)), SPLIT(AEAD_DEC(Gⁿnil, AEAD_ENC(Gⁿnil, CONCAT(nil, HASH(nil, nil, nil))), nil), nil)), nil, CONCAT(HASH(nil, nil, nil), SPLIT(AEAD_DEC(Gⁿnil, AEAD_ENC(Gⁿnil, CONCAT(nil, HASH(nil, nil, nil))), nil), nil))), nil), nil), nil))) based on SPLIT(SPLIT(AEAD_DEC(Gⁿnb^{na}, AEAD_ENC(G^{na}nb, CONCAT(SPLIT(AEAD_DEC(G^{nc}nb, AEAD_ENC(G^{nb}nc, CONCAT(seq01, HASH(ntd, nil, nil))), nil), nil)?), SPLIT(AEAD_DEC(G^{nc}nb, AEAD_ENC(G^{nb}nc, CONCAT(seq01, HASH(ntd, nil, nil))), nil), nil)?), seq0101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^{nc}nb, AEAD_ENC(G^{nb}nc, CONCAT(seq01, HASH(ntd, nil, nil))), nil), nil)?)), uidb), nil, nil)?)).

Analysis • Constructed skeleton AEAD_DEC($G^{\wedge}nil$, AEAD_ENC($G^{\wedge}nil$, nil, nil), nil) based on AEAD_DEC($G^{\wedge}na^{\wedge}nb$, AEAD_ENC($G^{\wedge}nb^{\wedge}na$, uuida, nil), nil)?.

Analysis • Constructed skeleton ASSERT(nil, AEAD_DEC(G^nil, AEAD_ENC(G^nil, AEAD_DEC(G^nil, AEAD_ENC(G^nil, nil, nil), nil), nil), nil), nil) based on ASSERT(uuida, AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, AEAD_DEC(G^na^nb, AEAD_ENC(G^nb^na, uuida, nil), nil)?, nil), nil)?).

```

shab → Gnbna
shba → Gnanb
shcb → Gnbnc
shbc → Gncnb
dht_m → CONCAT(seq01, HASH(ntd, nil, nil))
dht → AEAD_ENC(Gnbnc, CONCAT(seq01, HASH(ntd, nil, nil)), nil)
dht_dec → AEAD_DEC(Gncnb, AEAD_ENC(Gnbnc, CONCAT(seq01, HASH(ntd, nil, nil)), nil, nil)?
subj_dhl → SPLIT(AEAD_DEC(Gncnb, AEAD_ENC(Gnbnc, CONCAT(seq01, HASH(ntd, nil, nil)), nil, nil)?
seq01 → SPLIT(AEAD_DEC(Gncnb, AEAD_ENC(Gnbnc, CONCAT(seq01, HASH(ntd, nil, nil)), nil, nil)?
fpd → SPLIT(AEAD_DEC(Gncnb, AEAD_ENC(Gnbnc, CONCAT(seq01, HASH(ntd, nil, nil)), nil, nil)?
dht_fp → CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(Gncnb, AEAD_ENC(Gnbnc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),

```

```

dht01 → AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)),
nil, nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil, nil)?), seq0101, CONCAT(HASH(ntc,
nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil, nil)?)), uuidb), nil)

```

$$\text{subj_dhl_0} \rightarrow \text{SPLIT}(\text{AEAD_DEC}(\text{G}^{\text{nb}}\text{na}, \text{AEAD_ENC}(\text{G}^{\text{na}}\text{nb}, \text{CONCAT}(\text{SPLIT}(\text{AEAD_DEC}(\text{G}^{\text{nc}}\text{nb}, \text{AEAD_ENC}(\text{G}^{\text{nb}}\text{nc}, \text{CONCAT}(\text{seq01}, \text{HASH}(\text{ntd}, \text{nil}, \text{nil})), \text{nil}), \text{nil})), \text{SPLIT}(\text{AEAD_DEC}(\text{G}^{\text{nc}}\text{nb}, \text{AEAD_ENC}(\text{G}^{\text{nb}}\text{nc}, \text{CONCAT}(\text{seq01}, \text{HASH}(\text{ntd}, \text{nil}, \text{nil})), \text{nil}), \text{nil})), \text{seq0101}, \text{CONCAT}(\text{HASH}(\text{ntc}, \text{nil}, \text{nil}), \text{SPLIT}(\text{AEAD_DEC}(\text{G}^{\text{nc}}\text{nb}, \text{AEAD_ENC}(\text{G}^{\text{nb}}\text{nc}, \text{CONCAT}(\text{seq01}, \text{HASH}(\text{ntd}, \text{nil}, \text{nil})), \text{nil}), \text{nil}))), \text{uuidb}, \text{nil}), \text{nil}))$$

```
seq0101_ → SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc,
CONCAT(seq01, HASH(ntd, nil, nil)), nil, nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
```

```

nil?), seq0101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil))), uuidb, nil, nil)?
    dht_fp → SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc,
CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil)?), seq0101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil))), uuidb, nil, nil)?
    uuidb → SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc,
CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil)?), seq0101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil))), uuidb, nil, nil)?
    ping → AEAD_ENC(G^nb^na, uuida, nil) ← obtained by Attacker
    fpc_0 → SPLIT(SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc,
CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil)?), seq0101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil))), uuidb, nil, nil)?
    fpd_0 → SPLIT(SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc,
CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil)?), seq0101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil))), uuidb, nil, nil)?
    ping_dec → AEAD_DEC(G^na^nb, AEAD_ENC(G^nb^na, uuida, nil), nil)?
    ack → AEAD_ENC(G^na^nb, AEAD_DEC(G^na^nb, AEAD_ENC(G^nb^na, uuida, nil), nil)?, nil)
    ack_dec → AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, AEAD_DEC(G^na^nb, AEAD_ENC(G^nb^na, uuida, nil), nil)?, nil), nil)?
    unnamed_0 → ASSERT(uuida, AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, AEAD_DEC(G^na^nb, AEAD_ENC(G^nb^na, uuida, nil),
nil)?, nil), nil)?)?
    ack (AEAD_ENC(G^nb^na, uuida, nil)) is obtained by Attacker.

```

Result • confidentiality? ping — When:

```

shab → G^nb^na
shba → G^na^nb
shcb → G^nb^nc
shbc → G^nc^nb
dht_m → CONCAT(seq01, HASH(ntd, nil, nil))
dht → AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil)
dht_dec → AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?
subj_dhl → SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?
seq01 → SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?
fpd → SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?
dht_fp → CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil)))
    dht_m1 → CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?,
SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), seq0101, CONCAT(HASH(ntc, nil, nil),
SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), uuidb)
    dht01 → AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)),
nil), nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), seq0101, CONCAT(HASH(ntc,
nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), uuidb, nil)
    dht01_dec → AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc,
CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil)?), seq0101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil))), uuidb, nil, nil)?
    subj_dhl_0 → SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc,
CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil)?), seq0101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil))), uuidb, nil, nil)?
    seq01_0 → SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc,
CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),

```

Deduction • Output of CONCAT(HASH(ntc, nil, nil), nil) obtained by reconstructing with HASH(ntc, nil, nil), nil.

Deduction • Output of AEAD_ENC(G^{nil} , CONCAT(nil, HASH(nil, nil, nil)), nil) obtained by decomposing AEAD_DEC(G^{nil} , AEAD_ENC(G^{nil} , CONCAT(nil, HASH(nil, nil, nil)), nil), nil) with G^{nil} .

Deduction • Output of AEAD_ENC(G^{nil} , CONCAT(SPLIT(AEAD_DEC(G^{nil} , AEAD_ENC(G^{nil} , CONCAT(nil, HASH(nil, nil, nil)), nil, nil)), SPLIT(AEAD_DEC(G^{nil} , AEAD_ENC(G^{nil} , CONCAT(nil, HASH(nil, nil, nil)), nil, nil)), nil, nil), nil, nil), nil, nil) obtained by decomposing AEAD_DEC(G^{nil} , AEAD_ENC(G^{nil} , CONCAT(SPLIT(AEAD_DEC(G^{nil} , AEAD_ENC(G^{nil} , CONCAT(nil, HASH(nil, nil, nil)), nil, nil)), SPLIT(AEAD_DEC(G^{nil} , AEAD_ENC(G^{nil} , CONCAT(nil, HASH(nil, nil, nil)), nil, nil)), nil, nil), nil, nil), nil, nil), nil, nil) with G^{nil} .

Deduction • Output of AEAD_ENC(G^{nil} , nil, nil) obtained by decomposing AEAD_DEC(G^{nil} , AEAD_ENC(G^{nil} , nil, nil), nil) with G^{nil} .

Deduction • Output of AEAD_ENC(G^{nil} , AEAD_DEC(G^{nil} , AEAD_ENC(G^{nil} , nil, nil), nil), nil) obtained by decomposing AEAD_DEC(G^{nil} , AEAD_ENC(G^{nil} , AEAD_DEC(G^{nil} , AEAD_ENC(G^{nil} , nil, nil), nil), nil), nil) with G^{nil} .

Deduction • Output of AEAD_ENC($G^{na^{nb}}$, CONCAT(seq01, HASH(ntd, nil, nil), seq0101, CONCAT(HASH(ntc, nil, nil), nil), uuidb), nil) obtained by equalizing with the current resolution of dht01.

Deduction • Output of HASH(nil, nil, nil) obtained as a concatenated fragment of CONCAT(nil, HASH(nil, nil, nil)).

Analysis • Initializing Stage 1 mutation map for Na...

Deduction • G^{nil} obtained by reconstructing with nil. (Analysis 4)

Analysis • Initializing Stage 1 mutation map for Nb...

Analysis • Initializing Stage 1 mutation map for Nc...

Analysis • Initializing Stage 1 mutation map for Nd...

Analysis • Initializing Stage 3 mutation map for Na...

Analysis • Initializing Stage 2 mutation map for Na...

Deduction • Output of HASH(nil, nil, hndsm) obtained by reconstructing with nil, nil, hndsm. (Analysis 313)

Deduction • Output of HASH(nil, hndsm, nil) obtained by reconstructing with nil, hndsm, nil. (Analysis 323)

Deduction • Output of HASH(nil, hndsm, hndsm) obtained by reconstructing with nil, hndsm, hndsm. (Analysis 336)

Deduction • Output of HASH(hndsm, nil, nil) obtained by reconstructing with hndsm, nil, nil. (Analysis 348)

Deduction • Output of HASH(hndsm, nil, hndsm) obtained by reconstructing with hndsm, nil, hndsm. (Analysis 360)
Deduction • Output of HASH(hndsm, hndsm, nil) obtained by reconstructing with hndsm, hndsm, nil. (Analysis 372)
Deduction • Output of HASH(hndsm, hndsm, hndsm) obtained by reconstructing with hndsm, hndsm, hndsm. (Analysis 382)
Analysis • Initializing Stage 2 mutation map for Nb...
Analysis • Initializing Stage 2 mutation map for Nc...
Analysis • Initializing Stage 2 mutation map for Nd...
Analysis • Initializing Stage 3 mutation map for Nb...
Analysis • Initializing Stage 3 mutation map for Nc...
Analysis • Initializing Stage 3 mutation map for Nd...
Analysis • Initializing Stage 5 mutation map for Na...
Analysis • Initializing Stage 4 mutation map for Na...
Analysis • Initializing Stage 4 mutation map for Nb...
Analysis • Initializing Stage 4 mutation map for Nc...
Analysis • Initializing Stage 4 mutation map for Nd...
Analysis • Initializing Stage 5 mutation map for Nb...
Analysis • Initializing Stage 5 mutation map for Nc...
Analysis • Initializing Stage 5 mutation map for Nd...
Analysis • Initializing Stage 6 mutation map for Na...
Analysis • Initializing Stage 6 mutation map for Nb...
Analysis • Initializing Stage 6 mutation map for Nc...
Analysis • Initializing Stage 6 mutation map for Nd...

Verifpal • Verification completed for 'DHT_Message.vp' at 11:37:24 AM.

Verifpal • Summary of failed queries will follow.

Result • confidentiality? ack — When:

```

shab → G^nb^na
shba → G^na^nb
shcb → G^nb^nc
shbc → G^nc^nb
dht_m → CONCAT(seq01, HASH(ntd, nil, nil))
dht → AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil)
dht_dec → AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil, nil)?
subj_dhl_ → SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil, nil)?
seq01_ → SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil, nil)?
fpd_ → SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil, nil)?
dht_fp → CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil,
nil)?))
dht_m1 → CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil, nil)?),
SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil, nil)?), seq0101, CONCAT(HASH(ntc, nil, nil),
SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil, nil)?)), uuidb)
dht01 → AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)),
nil, nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil, nil)?), seq0101, CONCAT(HASH(ntc,
nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil, nil)?)), uuidb), nil)
dht01_dec → AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc,
CONCAT(seq01, HASH(ntd, nil, nil)), nil, nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil,
nil)?), seq0101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil,
nil)?)), uuidb), nil, nil)?
subj_dhl_0 → SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc,
CONCAT(seq01, HASH(ntd, nil, nil)), nil, nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil,
nil)?), seq0101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil,
nil)?)), uuidb), nil, nil)?
seq01_0 → SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc,
CONCAT(seq01, HASH(ntd, nil, nil)), nil, nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil,

```

```

nil)?, seq0101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil))), uuidb), nil, nil)?
    seq0101_ → SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc,
CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil)?), seq0101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil))), uuidb), nil, nil)?
    dht_fp_ → SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc,
CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil)?), seq0101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil))), uuidb), nil, nil)?
    uuidb_ → SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc,
CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil)?), seq0101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil))), uuidb), nil, nil)?
    ping → AEAD_ENC(G^nb^na, uuida, nil) ← obtained by Attacker
    fpc_0 → SPLIT(SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc,
CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil)?), seq0101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil))), uuidb), nil, nil)?
    fpd_0 → SPLIT(SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc,
CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil)?), seq0101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil))), uuidb), nil, nil)?
    ping_dec → AEAD_DEC(G^na^nb, AEAD_ENC(G^nb^na, uuida, nil), nil)?
    ack → AEAD_ENC(G^na^nb, AEAD_DEC(G^na^nb, AEAD_ENC(G^nb^na, uuida, nil), nil)?, nil)
    ack_dec → AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, AEAD_DEC(G^na^nb, AEAD_ENC(G^nb^na, uuida, nil), nil)?, nil), nil)?
    unnamed_0 → ASSERT(uuida, AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, AEAD_DEC(G^na^nb, AEAD_ENC(G^nb^na, uuida, nil),
nil)?, nil), nil)?)?
    ack (AEAD_ENC(G^nb^na, uuida, nil)) is obtained by Attacker.

```

Result • confidentiality? ping — When:

```

shab → G^nb^na
shba → G^na^nb
shcb → G^nb^nc
shbc → G^nc^nb
dht_m → CONCAT(seq01, HASH(ntd, nil, nil))
dht → AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil)
dht_dec → AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?
subj_dhl_ → SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?
seq01_ → SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?
fpd_ → SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?
dht_fp → CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil)))
    dht_m1 → CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?),
SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), seq0101, CONCAT(HASH(ntc, nil, nil),
SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), uuidb)
    dht01 → AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)),
nil), nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), seq0101, CONCAT(HASH(ntc,
nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), uuidb), nil)
    dht01_dec → AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc,
CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil)?), seq0101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil))), uuidb), nil, nil)?
    subj_dhl_0 → SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc,
CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),

```

```

nil?), seq0101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil))), uuidb, nil, nil)?
    seq01_0 → SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc,
CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil)?), seq0101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil))), uuidb, nil, nil)?
    seq0101_ → SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc,
CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil)?), seq0101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil))), uuidb, nil, nil)?
    dht_fp_ → SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc,
CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil)?), seq0101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil))), uuidb, nil, nil)?
    uuidb_ → SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc,
CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil)?), seq0101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil))), uuidb, nil, nil)?
    ping → AEAD_ENC(G^nb^na, uuida, nil) ← obtained by Attacker
    fpc_0 → SPLIT(SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc,
CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil)?), seq0101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil))), uuidb, nil, nil)?
    fpd_0 → SPLIT(SPLIT(AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, CONCAT(SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc,
CONCAT(seq01, HASH(ntd, nil, nil)), nil), nil)?), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil)?), seq0101, CONCAT(HASH(ntc, nil, nil), SPLIT(AEAD_DEC(G^nc^nb, AEAD_ENC(G^nb^nc, CONCAT(seq01, HASH(ntd, nil, nil)), nil),
nil))), uuidb, nil, nil)?
    ping_dec → AEAD_DEC(G^na^nb, AEAD_ENC(G^nb^na, uuida, nil), nil)?
    ack → AEAD_ENC(G^na^nb, AEAD_DEC(G^na^nb, AEAD_ENC(G^nb^na, uuida, nil), nil)?, nil)
    ack_dec → AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, AEAD_DEC(G^na^nb, AEAD_ENC(G^nb^na, uuida, nil), nil)?, nil), nil)?
    unnamed_0 → ASSERT(uuida, AEAD_DEC(G^nb^na, AEAD_ENC(G^na^nb, AEAD_DEC(G^na^nb, AEAD_ENC(G^nb^na, uuida, nil),
nil)?, nil), nil)?
    ping (AEAD_ENC(G^nb^na, uuida, nil)) is obtained by Attacker.

```

Verifpal • Thank you for using Verifpal.

File Name: Pheromone_Message.vp

```

attacker[active]
principal NA[
    generates na,s
    gna = G^na
]
principal NB[
    generates nb,seq0101
    gnb = G^nb
]

```

NA-> NB:[gna]

NB-> NA:[gnb]

```
principal NC[
    generates nc,seq01
    gnc =  $G^{nc}$ 
]
```

```
NB-> NC:[gnb]
NC->NB:[gnc]
```

```
principal ND[
    generates nd,r
    gnd =  $G^{nd}$ 
]
```

```
NC-> ND:[gnc]
ND->NC:[gnd]
```

```
principal Alice[
    knows private n
]
```

```
principal NA[
    knows private n,uuidA
    sub = HASH(n)
    h1,h2,h3 = HKDF(sub,nil,nil)
    h0 = CONCAT(h1,uuidA)
    flgS = CONCAT(sub,s)
]
```

```
principal Della[
    knows private n
]
```

```
principal ND[
    knows private n,uuidD
    subj= HASH(n)
    h11,h12,h13 = HKDF(subj,nil,nil)
]
```

```
NA-> NC:[gna]
NC-> NA:[gnc]
```

```
principal NA[
    shAB =  $gnb^{na}$ 
]
```

```

    shAC = gnc^na
    m1=AEAD_ENC(shAB,h0,flgS)
    m2=AEAD_ENC(shAC,h0,flgS)
]
NA-> NB:[m1],[flgS]
principal NB[
    shBA = gna^nb
    shBC = gnc^nb
    m1_de = AEAD_DEC(shBA,m1,flgS)?
    h10_,uuidA0_ = SPLIT(m1_de)
    m1_de0 = CONCAT(h10_,uuidA0_,seq0101)
    sub_,s_ = SPLIT(flgs)
    flgS1 = CONCAT(sub_,s_)
    m3 = AEAD_ENC(shBC,m1_de0,flgS1)
]
NA-> NC:[m2],[flgS]
principal NC[
    shCA = gna^nc
    shCB =gnb^nc
    shCD =gnd^nc
    m2_dec = AEAD_DEC(shCA,m2,flgS)?
    h1_,uuidA_ = SPLIT(m2_dec)
    m2_dec0 = CONCAT(h1_,uuidA_,seq01)
    sub_0,s_0 = SPLIT(flgs)
    flgS2 = CONCAT(sub_0,s_0)
    mA2C = AEAD_ENC(shCD,m2_dec0,flgS2)
]

NB->NC :[m3],[flgS1]

principal NC[
    m3_de = AEAD_DEC(shCB,m3,flgS1)?
    h10_1,uuidA0_1,seq0101_ = SPLIT(m3_de)
    sub_1,s_1 = SPLIT(flgs1)
    _ = ASSERT(sub_0,sub_1)?
]

NC->ND:[mA2C],[flgS2]

principal ND[
    shDC = gnc^nd

```

```

mA2C_de = AEAD_DEC(shDC,mA2C,flgS2)?
sub_10,s_10 =SPLIT(flgs2)
h100_,uuidA1_,seq01_ =SPLIT(mA2C_de)
h10 = CONCAT(h11,uuidD)
flgR = CONCAT(subj,r)
m11=AEAD_ENC(shDC,h10,flgR)

```

]

ND->NC:[m11],[flgR]

principal NC[

```

m11_de = AEAD_DEC(shCD, m11,flgR)?
subj_,r_ = SPLIT(flgsR)
flgR1 = CONCAT(subj_,r_)
h11_,uuidD_ = SPLIT(m11_de)
m11_de0= CONCAT( h11_,uuidD_,seq01)
mDC_A= AEAD_ENC(shCA,m11_de,flgR1)

```

]

NC->NA:[mDC_A],[flgR1]

principal NA[

```

mDC_A_de = AEAD_DEC(shAC,mDC_A,flgR1)?
h110_,uuidD0_,seq01_1 = SPLIT(mDC_A_de)
subj_1,r_1 = SPLIT(flgsR1)

```

]

queries[

```

confidentiality? n
equivalence? sub,subj
confidentiality? h11
confidentiality?mDC_A
equivalence?h1,h110_
confidentiality? uuidD_
authentication?NC->ND:mA2C
equivalence?h100_,h11
equivalence? uuidA0_,uuidA0_1
confidentiality? uuidA0_

```


Deduction • Output of $\text{AEAD_ENC}(G^{\text{nil}}, \text{AEAD_DEC}(G^{\text{nil}}, \text{AEAD_ENC}(G^{\text{nil}}, \text{CONCAT}(\text{HKDF}(\text{HASH}(\text{nil}), \text{nil}, \text{nil}), \text{nil}), \text{CONCAT}(\text{HASH}(\text{nil}), \text{nil})), \text{CONCAT}(\text{HASH}(\text{nil}), \text{nil})), \text{CONCAT}(\text{SPLIT}(\text{CONCAT}(\text{HASH}(\text{nil}), \text{nil})), \text{SPLIT}(\text{CONCAT}(\text{HASH}(\text{nil}), \text{nil}))))$ obtained by decomposing $\text{AEAD_DEC}(G^{\text{nil}}, \text{AEAD_ENC}(G^{\text{nil}}, \text{AEAD_DEC}(G^{\text{nil}}, \text{AEAD_ENC}(G^{\text{nil}}, \text{CONCAT}(\text{HKDF}(\text{HASH}(\text{nil}), \text{nil}, \text{nil}), \text{nil}), \text{CONCAT}(\text{HASH}(\text{nil}), \text{nil})), \text{CONCAT}(\text{HASH}(\text{nil}), \text{nil})), \text{CONCAT}(\text{SPLIT}(\text{CONCAT}(\text{HASH}(\text{nil}), \text{nil})), \text{SPLIT}(\text{CONCAT}(\text{HASH}(\text{nil}), \text{nil}))))$ with G^{nil} .

Deduction • First output of $\text{HKDF}(\text{HASH}(n), \text{nil}, \text{nil})$ obtained by reconstructing with $\text{HASH}(n), \text{nil}, \text{nil}$.

Result • confidentiality? h11 — When:

```

h1 → HKDF(HASH(n), nil, nil) ← obtained by Attacker
h2 → HKDF(HASH(n), nil, nil) ← obtained by Attacker
h3 → HKDF(HASH(n), nil, nil) ← obtained by Attacker
h0 → CONCAT(HKDF(HASH(n), nil, nil), uuida)
flgs → CONCAT(HASH(n), s)
h11 → HKDF(HASH(n), nil, nil) ← obtained by Attacker
h12 → HKDF(HASH(n), nil, nil) ← obtained by Attacker
h13 → HKDF(HASH(n), nil, nil) ← obtained by Attacker
shab → Gnbna
shac → Gncna
m1 → AEAD_ENC(Gnbna, CONCAT(HKDF(HASH(n), nil, nil), uuida), CONCAT(HASH(n), s))
m2 → AEAD_ENC(Gncna, CONCAT(HKDF(HASH(n), nil, nil), uuida), CONCAT(HASH(n), s))
shba → Gnanb
shbc → Gncnb
m1_de → CONCAT(HKDF(HASH(n), nil, nil), uuida)
h10_ → HKDF(HASH(n), nil, nil) ← obtained by Attacker
uuida0_ → uuida
m1_de0 → CONCAT(HKDF(HASH(n), nil, nil), uuida, seq0101)
sub_ → HASH(n)
s_ → s
flgs1 → CONCAT(HASH(n), s)
m3 → AEAD_ENC(Gncnb, CONCAT(HKDF(HASH(n), nil, nil), uuida, seq0101), CONCAT(HASH(n), s))
shca → Gnanc
shcb → Gnbnc
shcd → Gndnc
m2_dec → CONCAT(HKDF(HASH(n), nil, nil), uuida)
h1_ → HKDF(HASH(n), nil, nil) ← obtained by Attacker
uuida_ → uuida
m2_dec0 → CONCAT(HKDF(HASH(n), nil, nil), uuida, seq01)
sub_0 → HASH(n)
s_0 → s
flgs2 → CONCAT(HASH(n), s)
ma2c → AEAD_ENC(Gndnc, CONCAT(HKDF(HASH(n), nil, nil), uuida, seq01), CONCAT(HASH(n), s))
m3_de → CONCAT(HKDF(HASH(n), nil, nil), uuida, seq0101)
h10_1 → HKDF(HASH(n), nil, nil) ← obtained by Attacker
uuida0_1 → uuida
seq0101_ → seq0101
sub_1 → HASH(n)
s_1 → s
unnamed_0 → ASSERT(HASH(n), HASH(n))?
shdc → Gncnd
ma2c_de → CONCAT(HKDF(HASH(n), nil, nil), uuida, seq01)
sub_10 → HASH(n)
s_10 → s
h100_ → HKDF(HASH(n), nil, nil) ← obtained by Attacker
uuida1_ → uuida
seq01_ → seq01
h10 → CONCAT(HKDF(HASH(n), nil, nil), uuidd)
flgr → CONCAT(HASH(n), r)

```

```

m11 → AEAD_ENC(G^nc^nd, CONCAT(HKDF(HASH(n), nil, nil), uuid), CONCAT(HASH(n), r))
m11_de → CONCAT(HKDF(HASH(n), nil, nil), uuid)
subj_ → HASH(n)
r_ → r
flgr1 → CONCAT(HASH(n), r)
h11_ → HKDF(HASH(n), nil, nil) ← obtained by Attacker
uuid_ → uuid
m11_de0 → CONCAT(HKDF(HASH(n), nil, nil), uuid, seq01)
mdc_a → AEAD_ENC(G^na^nc, CONCAT(HKDF(HASH(n), nil, nil), uuid), CONCAT(HASH(n), r))
mdc_a_de → CONCAT(HKDF(HASH(n), nil, nil), uuid)
h110_ → HKDF(HASH(n), nil, nil) ← obtained by Attacker
uuid0_ → uuid
seq01_1 → nil
subj_1 → HASH(n)
r_1 → r
h11 (HKDF(HASH(n), nil, nil)) is obtained by Attacker.

```

Deduction • Second output of HKDF(HASH(n), nil, nil) obtained by reconstructing with HASH(n), nil, nil.

Deduction • Third output of HKDF(HASH(n), nil, nil) obtained by reconstructing with HASH(n), nil, nil.

Deduction • Output of ASSERT(HASH(n), HASH(n))? obtained by reconstructing with HASH(n), HASH(n).

Deduction • First output of SPLIT(CONCAT(HASH(n), s)) obtained as a concatenated fragment of CONCAT(SPLIT(CONCAT(HASH(n), s)), SPLIT(CONCAT(HASH(n), s))).

Deduction • Second output of SPLIT(CONCAT(HASH(n), s)) obtained as a concatenated fragment of CONCAT(SPLIT(CONCAT(HASH(n), s)), SPLIT(CONCAT(HASH(n), s))).

Deduction • Output of AEAD_ENC(G^nc^nb, CONCAT(HKDF(HASH(n), nil, nil), uuid, seq0101), CONCAT(HASH(n), s)) obtained by equivalizing with the current resolution of m3.

Deduction • Output of AEAD_ENC(G^nd^nc, CONCAT(HKDF(HASH(n), nil, nil), uuid, seq01), CONCAT(HASH(n), s)) obtained by equivalizing with the current resolution of ma2c.

Deduction • r obtained as a concatenated fragment of CONCAT(HASH(n), r).

Deduction • First output of SPLIT(CONCAT(HASH(n), r)) obtained as a concatenated fragment of CONCAT(SPLIT(CONCAT(HASH(n), r)), SPLIT(CONCAT(HASH(n), r))).

Deduction • Second output of SPLIT(CONCAT(HASH(n), r)) obtained as a concatenated fragment of CONCAT(SPLIT(CONCAT(HASH(n), r)), SPLIT(CONCAT(HASH(n), r))).

Deduction • Output of AEAD_ENC(G^na^nc, CONCAT(HKDF(HASH(n), nil, nil), uuid), CONCAT(HASH(n), r)) obtained by equivalizing with the current resolution of mdc_a.

Result • confidentiality? mdc_a — When:

```

h1 → HKDF(HASH(n), nil, nil)
h2 → HKDF(HASH(n), nil, nil)
h3 → HKDF(HASH(n), nil, nil)
h0 → CONCAT(HKDF(HASH(n), nil, nil), uuid)
flgs → CONCAT(HASH(n), s)
h11 → HKDF(HASH(n), nil, nil)
h12 → HKDF(HASH(n), nil, nil)
h13 → HKDF(HASH(n), nil, nil)
shab → G^nb^na
shac → G^nc^na
m1 → AEAD_ENC(G^nb^na, CONCAT(HKDF(HASH(n), nil, nil), uuid), CONCAT(HASH(n), s))
m2 → AEAD_ENC(G^nc^na, CONCAT(HKDF(HASH(n), nil, nil), uuid), CONCAT(HASH(n), s))
shba → G^na^nb
shbc → G^nc^nb
m1_de → CONCAT(HKDF(HASH(n), nil, nil), uuid)
h10_ → HKDF(HASH(n), nil, nil)
uuida0_ → uuida
m1_de0 → CONCAT(HKDF(HASH(n), nil, nil), uuida, seq0101)
sub_ → HASH(n)
s_ → s

```

```

flgs1 → CONCAT(HASH(n), s)
m3 → AEAD_ENC(G^nc^nb, CONCAT(HKDF(HASH(n), nil, nil), uuida, seq0101), CONCAT(HASH(n), s))
shca → G^na^nc
shcb → G^nb^nc
shcd → G^nd^nc
m2_dec → CONCAT(HKDF(HASH(n), nil, nil), uuida)
h1_ → HKDF(HASH(n), nil, nil)
uuida_ → uuida
m2_dec0 → CONCAT(HKDF(HASH(n), nil, nil), uuida, seq01)
sub_0 → HASH(n)
s_0 → s
flgs2 → CONCAT(HASH(n), s)
ma2c → AEAD_ENC(G^nd^nc, CONCAT(HKDF(HASH(n), nil, nil), uuida, seq01), CONCAT(HASH(n), s))
m3_de → CONCAT(HKDF(HASH(n), nil, nil), uuida, seq0101)
h10_1 → HKDF(HASH(n), nil, nil)
uuida0_1 → uuida
seq0101_ → seq0101
sub_1 → HASH(n)
s_1 → s
unnamed_0 → ASSERT(HASH(n), HASH(n))?
shdc → G^nc^nd
ma2c_de → CONCAT(HKDF(HASH(n), nil, nil), uuida, seq01)
sub_10 → HASH(n)
s_10 → s
h100_ → HKDF(HASH(n), nil, nil)
uuida1_ → uuida
seq01_ → seq01
h10 → CONCAT(HKDF(HASH(n), nil, nil), uidd)
flgr → CONCAT(HASH(n), r)
m11 → AEAD_ENC(G^nc^nd, CONCAT(HKDF(HASH(n), nil, nil), uidd), CONCAT(HASH(n), r))
m11_de → CONCAT(HKDF(HASH(n), nil, nil), uidd)
subj_ → HASH(n)
r_ → r
flgr1 → CONCAT(HASH(n), r)
h11_ → HKDF(HASH(n), nil, nil)
uidd_ → uidd
m11_de0 → CONCAT(HKDF(HASH(n), nil, nil), uidd, seq01)
mdc_a → AEAD_ENC(G^na^nc, CONCAT(HKDF(HASH(n), nil, nil), uidd), CONCAT(HASH(n), r)) ← obtained by Attacker
mdc_a_de → CONCAT(HKDF(HASH(n), nil, nil), uidd)
h110_ → HKDF(HASH(n), nil, nil)
uidd0_ → uidd
seq01_1 → nil
subj_1 → HASH(n)
r_1 → r
mdc_a (AEAD_ENC(G^na^nc, CONCAT(HKDF(HASH(n), nil, nil), uidd), CONCAT(HASH(n), r))) is obtained by Attacker.

```

Analysis • Initializing Stage 1 mutation map for Na...

Deduction • G^{nil} obtained by reconstructing with nil. (Analysis 6)

Deduction • Output of $\text{CONCAT}(\text{HASH}(\text{nil}), \text{nil})$ obtained by reconstructing with $\text{HASH}(\text{nil}), \text{nil}$. (Analysis 9)

Analysis • Initializing Stage 1 mutation map for Nb...

Analysis • Initializing Stage 1 mutation map for Nc...

Analysis • Initializing Stage 1 mutation map for Nd...

Analysis • Initializing Stage 1 mutation map for Alice...

Analysis • Initializing Stage 1 mutation map for Della...

Analysis • Initializing Stage 3 mutation map for Na...

Analysis • Initializing Stage 2 mutation map for Na...

Analysis • Initializing Stage 2 mutation map for Nb...
Analysis • Initializing Stage 2 mutation map for Nc...
Deduction • Output of CONCAT(HASH(nil), s) obtained by reconstructing with HASH(nil), s. (Analysis 173)
Analysis • Initializing Stage 2 mutation map for Nd...
Deduction • Output of CONCAT(HASH(nil), r) obtained by reconstructing with HASH(nil), r. (Analysis 186)
Deduction • Output of CONCAT(HASH(n), nil) obtained by reconstructing with HASH(n), nil. (Analysis 202)
Analysis • Initializing Stage 3 mutation map for Nb...
Analysis • Initializing Stage 3 mutation map for Nc...
Analysis • Initializing Stage 2 mutation map for Alice...
Analysis • Initializing Stage 2 mutation map for Della...
Analysis • Initializing Stage 3 mutation map for Nd...
Analysis • Initializing Stage 3 mutation map for Alice...
Analysis • Initializing Stage 3 mutation map for Della...
Analysis • Initializing Stage 5 mutation map for Na...
Analysis • Initializing Stage 4 mutation map for Na...
Deduction • Output of HASH(s) obtained by reconstructing with s. (Analysis 526)
Deduction • Output of CONCAT(HASH(s), nil) obtained by reconstructing with HASH(s), nil. (Analysis 526)
Deduction • Output of CONCAT(HASH(s), s) obtained by reconstructing with HASH(s), s. (Analysis 542)
Deduction • Output of CONCAT(HASH(s), r) obtained by reconstructing with HASH(s), r. (Analysis 557)
Deduction • Output of HASH(r) obtained by reconstructing with r. (Analysis 574)
Deduction • Output of CONCAT(HASH(r), nil) obtained by reconstructing with HASH(r), nil. (Analysis 574)
Deduction • Output of CONCAT(HASH(r), s) obtained by reconstructing with HASH(r), s. (Analysis 589)
Deduction • Output of CONCAT(HASH(r), r) obtained by reconstructing with HASH(r), r. (Analysis 605)
Analysis • Initializing Stage 4 mutation map for Nb...
Analysis • Initializing Stage 4 mutation map for Nc...
Analysis • Initializing Stage 4 mutation map for Nd...
Analysis • Initializing Stage 5 mutation map for Nb...
Analysis • Initializing Stage 5 mutation map for Nc...
Analysis • Initializing Stage 5 mutation map for Nd...
Analysis • Initializing Stage 4 mutation map for Alice...
Analysis • Initializing Stage 4 mutation map for Della...
Analysis • Initializing Stage 5 mutation map for Alice...
Analysis • Initializing Stage 5 mutation map for Della...
Analysis • Initializing Stage 6 mutation map for Na...
Analysis • Initializing Stage 6 mutation map for Nb...
Analysis • Initializing Stage 6 mutation map for Nc...
Analysis • Initializing Stage 6 mutation map for Nd...
Analysis • Initializing Stage 6 mutation map for Alice...
Analysis • Initializing Stage 6 mutation map for Della...

Verifpal • Verification completed for 'Pheromone_Message.vp' at 11:51:01 AM.

Verifpal • Summary of failed queries will follow.

Result • confidentiality? h11 — When:

h1 → HKDF(HASH(n), nil, nil) ← obtained by Attacker
h2 → HKDF(HASH(n), nil, nil) ← obtained by Attacker
h3 → HKDF(HASH(n), nil, nil) ← obtained by Attacker
h0 → CONCAT(HKDF(HASH(n), nil, nil), uuida)
flgs → CONCAT(HASH(n), s)
h11 → HKDF(HASH(n), nil, nil) ← obtained by Attacker
h12 → HKDF(HASH(n), nil, nil) ← obtained by Attacker
h13 → HKDF(HASH(n), nil, nil) ← obtained by Attacker
shab → G^{nb}^{na}
shac → G^{nc}^{na}
m1 → AEAD_ENC(G^{nb}^{na}, CONCAT(HKDF(HASH(n), nil, nil), uuida), CONCAT(HASH(n), s))

```

m2 → AEAD_ENC(G^nc^na, CONCAT(HKDF(HASH(n), nil, nil), uuida), CONCAT(HASH(n), s))
shba → G^na^nb
shbc → G^nc^nb
m1_de → CONCAT(HKDF(HASH(n), nil, nil), uuida)
h10_ → HKDF(HASH(n), nil, nil) ← obtained by Attacker
uuida0_ → uuida
m1_de0 → CONCAT(HKDF(HASH(n), nil, nil), uuida, seq0101)
sub_ → HASH(n)
s_ → s
flgs1 → CONCAT(HASH(n), s)
m3 → AEAD_ENC(G^nc^nb, CONCAT(HKDF(HASH(n), nil, nil), uuida, seq0101), CONCAT(HASH(n), s))
shca → G^na^nc
shcb → G^nb^nc
shcd → G^nd^nc
m2_dec → CONCAT(HKDF(HASH(n), nil, nil), uuida)
h1_ → HKDF(HASH(n), nil, nil) ← obtained by Attacker
uuida_ → uuida
m2_dec0 → CONCAT(HKDF(HASH(n), nil, nil), uuida, seq01)
sub_0 → HASH(n)
s_0 → s
flgs2 → CONCAT(HASH(n), s)
ma2c → AEAD_ENC(G^nd^nc, CONCAT(HKDF(HASH(n), nil, nil), uuida, seq01), CONCAT(HASH(n), s))
m3_de → CONCAT(HKDF(HASH(n), nil, nil), uuida, seq0101)
h10_1 → HKDF(HASH(n), nil, nil) ← obtained by Attacker
uuida0_1 → uuida
seq0101_ → seq0101
sub_1 → HASH(n)
s_1 → s
unnamed_0 → ASSERT(HASH(n), HASH(n))?
shdc → G^nc^nd
ma2c_de → CONCAT(HKDF(HASH(n), nil, nil), uuida, seq01)
sub_10 → HASH(n)
s_10 → s
h100_ → HKDF(HASH(n), nil, nil) ← obtained by Attacker
uuida1_ → uuida
seq01_ → seq01
h10 → CONCAT(HKDF(HASH(n), nil, nil), uidd)
flgr → CONCAT(HASH(n), r)
m11 → AEAD_ENC(G^nc^nd, CONCAT(HKDF(HASH(n), nil, nil), uidd), CONCAT(HASH(n), r))
m11_de → CONCAT(HKDF(HASH(n), nil, nil), uidd)
subj_ → HASH(n)
r_ → r
flgr1 → CONCAT(HASH(n), r)
h11_ → HKDF(HASH(n), nil, nil) ← obtained by Attacker
uidd_ → uidd
m11_de0 → CONCAT(HKDF(HASH(n), nil, nil), uidd, seq01)
mdc_a → AEAD_ENC(G^na^nc, CONCAT(HKDF(HASH(n), nil, nil), uidd), CONCAT(HASH(n), r))
mdc_a_de → CONCAT(HKDF(HASH(n), nil, nil), uidd)
h110_ → HKDF(HASH(n), nil, nil) ← obtained by Attacker
uidd0_ → uidd
seq01_1 → nil
subj_1 → HASH(n)
r_1 → r
h11 (HKDF(HASH(n), nil, nil)) is obtained by Attacker.

```

Result • confidentiality? mdc_a — When:

```

h1 → HKDF(HASH(n), nil, nil)
h2 → HKDF(HASH(n), nil, nil)
h3 → HKDF(HASH(n), nil, nil)
h0 → CONCAT(HKDF(HASH(n), nil, nil), uuida)
flgs → CONCAT(HASH(n), s)
h11 → HKDF(HASH(n), nil, nil)
h12 → HKDF(HASH(n), nil, nil)
h13 → HKDF(HASH(n), nil, nil)
shab → G^nb^na
shac → G^nc^na
m1 → AEAD_ENC(G^nb^na, CONCAT(HKDF(HASH(n), nil, nil), uuida), CONCAT(HASH(n), s))
m2 → AEAD_ENC(G^nc^na, CONCAT(HKDF(HASH(n), nil, nil), uuida), CONCAT(HASH(n), s))
shba → G^na^nb
shbc → G^nc^nb
m1_de → CONCAT(HKDF(HASH(n), nil, nil), uuida)
h10_ → HKDF(HASH(n), nil, nil)
uuida0_ → uuida
m1_de0 → CONCAT(HKDF(HASH(n), nil, nil), uuida, seq0101)
sub_ → HASH(n)
s_ → s
flgs1 → CONCAT(HASH(n), s)
m3 → AEAD_ENC(G^nc^nb, CONCAT(HKDF(HASH(n), nil, nil), uuida, seq0101), CONCAT(HASH(n), s))
shca → G^na^nc
shcb → G^nb^nc
shcd → G^nd^nc
m2_dec → CONCAT(HKDF(HASH(n), nil, nil), uuida)
h1_ → HKDF(HASH(n), nil, nil)
uuida_ → uuida
m2_dec0 → CONCAT(HKDF(HASH(n), nil, nil), uuida, seq01)
sub_0 → HASH(n)
s_0 → s
flgs2 → CONCAT(HASH(n), s)
ma2c → AEAD_ENC(G^nd^nc, CONCAT(HKDF(HASH(n), nil, nil), uuida, seq01), CONCAT(HASH(n), s))
m3_de → CONCAT(HKDF(HASH(n), nil, nil), uuida, seq0101)
h10_1 → HKDF(HASH(n), nil, nil)
uuida0_1 → uuida
seq0101_ → seq0101
sub_1 → HASH(n)
s_1 → s
unnamed_0 → ASSERT(HASH(n), HASH(n))?
shdc → G^nc^nd
ma2c_de → CONCAT(HKDF(HASH(n), nil, nil), uuida, seq01)
sub_10 → HASH(n)
s_10 → s
h100_ → HKDF(HASH(n), nil, nil)
uuida1_ → uuida
seq01_ → seq01
h10 → CONCAT(HKDF(HASH(n), nil, nil), uidd)
flgr → CONCAT(HASH(n), r)
m11 → AEAD_ENC(G^nc^nd, CONCAT(HKDF(HASH(n), nil, nil), uidd), CONCAT(HASH(n), r))
m11_de → CONCAT(HKDF(HASH(n), nil, nil), uidd)
subj_ → HASH(n)
r_ → r
flgr1 → CONCAT(HASH(n), r)
h11_ → HKDF(HASH(n), nil, nil)
uidd_ → uidd

```

```

m11_de0 → CONCAT(HKDF(HASH(n), nil, nil), uuid, seq01)
mdc_a → AEAD_ENC(G^na^nc, CONCAT(HKDF(HASH(n), nil, nil), uuid), CONCAT(HASH(n), r)) ← obtained by Attacker
mdc_a_de → CONCAT(HKDF(HASH(n), nil, nil), uuid)
h110_ → HKDF(HASH(n), nil, nil)
uidd0_ → uuid
seq01_1 → nil
subj_1 → HASH(n)
r_1 → r
mdc_a (AEAD_ENC(G^na^nc, CONCAT(HKDF(HASH(n), nil, nil), uuid), CONCAT(HASH(n), r))) is obtained by Attacker.

```

Verifpal • Thank you for using Verifpal.

File Name: Message_Intent_Token.vp

attacker[active]

```

principal NA[
  generates na
  gna = G^na
]

```

```

principal NC[
  generates nc,seq01
  gnc = G^nc
]

```

```

NC->NA:[gnc]
principal NA[
  shAC = gnc^na
]

```

```

NA->NC:[gna]
principal NC[
  shCA = gna^nc
]
NC->ND:[gnc]

```

```

principal ND[
  generates nd,r
  gnd = G^nd
  shDC = gnc^nd
]

```

```

ND->NC:[gnd]

```

```

principal NC[
    shCD = gnd^nc
]
principal NA[
    knows private n
    sub = HASH(n)
]

principal Della[
    knows private n,d, nfpD,ifpD,sd
    gd = G^d
    subj = HASH(n)
    sgD =SIGN(d,subj)
    msg0 = CONCAT(gd,subj,sgD,nfpD,ifpD)
    msg1 = AEAD_ENC(sd,msg0,nil)
]
Della-> ND :[msg1]

principal ND[
    knows private n,sd
    subj0 = HASH(n)
    msg1_de = AEAD_DEC(sd,msg1,nil)
    flgR = CONCAT(subj0,r)
    m0 = CONCAT(msg1_de,flgR)
    msgDC = AEAD_ENC(shDC,m0,nil)
]
ND->NC:[msgDC]

principal NC[
    knows private n
    subj_0 = Hash(n)
    msgDC_de = AEAD_DEC(shCD,msgDC,nil)?
    msg2,flgR1 = SPLIT(msgDC_de )
    subj1,r1 = SPLIT(flgR1)
    _ = ASSERT(subj1,subj_0)?
    flgR2= CONCAT(subj1,r1,seq01)
    m1 = CONCAT(msg2,flgR2)
    msgCA = AEAD_ENC(shCA,m1,nil)
]
NC->NA:[msgCA]

principal NA[

```



```

msgCA_de= AEAD_DEC(shAC,msgCA,nil)?
msg3,flgR3 = SPLIT(msgCA_de)
subj2,r2,seq01_1 = SPLIT(flgR3)
gd_,subj4,sgD_,nfpD_,ifpD_ = SPLIT(msg3)
proof = SIGNVERIF(gd_,subj4,sgD_)?
]

```

```

queries[
  authentication? ND->NC:msgDC
  confidentiality? msgDC
  confidentiality? r1
  confidentiality?msgCA
  confidentiality? n
  confidentiality? nfpD
  equivalence?subj4, sub
  confidentiality? nfpD_
  confidentiality? gd
  equivalence? gd,gd_
]

```

The Result

Run In terminal # verifpal verify Message_Intent_Token.vp

Verifpal 0.26.1 - <https://verifpal.com>

Warning • Verifpal is Beta software.

Verifpal • Parsing model 'MIT.vp'...

Verifpal • Verification initiated for 'MIT.vp' at 11:56:16 AM.

Info • Attacker is configured as active.

Info • Running at phase 0.

Analysis • Constructed skeleton HASH(nil) based on HASH(n).

Analysis • Constructed skeleton SIGN(nil, HASH(nil)) based on SIGN(d, HASH(n)).

Analysis • Constructed skeleton CONCAT(G^nil, HASH(nil), SIGN(nil, HASH(nil)), nil, nil) based on CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd).

Analysis • Constructed skeleton AEAD_DEC(nil, AEAD_ENC(nil, CONCAT(G^nil, HASH(nil), SIGN(nil, HASH(nil))), nil, nil, nil) based on AEAD_DEC(sd, AEAD_ENC(sd, CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), nil), nil).

Analysis • Constructed skeleton CONCAT(HASH(nil), nil) based on CONCAT(HASH(n), r).

Analysis • Constructed skeleton CONCAT(AEAD_DEC(nil, AEAD_ENC(nil, CONCAT(G^nil, HASH(nil), SIGN(nil, HASH(nil))), nil, nil, nil), nil), CONCAT(HASH(nil), nil)) based on CONCAT(AEAD_DEC(sd, AEAD_ENC(sd, CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), nil), nil), CONCAT(HASH(n), r)).

Analysis • Constructed skeleton AEAD_DEC(G^nil, AEAD_ENC(G^nil, CONCAT(AEAD_DEC(nil, AEAD_ENC(nil, CONCAT(G^nil, HASH(nil), SIGN(nil, HASH(nil))), nil, nil, nil), nil), nil) based on AEAD_DEC(G^nd^nc, AEAD_ENC(G^nc^nd, CONCAT(AEAD_DEC(sd, AEAD_ENC(sd, CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), nil), nil), CONCAT(HASH(n), r)), nil, nil)?.

Analysis • Constructed skeleton SPLIT(AEAD_DEC(G^nil, AEAD_ENC(G^nil, CONCAT(AEAD_DEC(nil, AEAD_ENC(nil, CONCAT(G^nil, HASH(nil), SIGN(nil, HASH(nil))), nil, nil, nil), nil), nil)) based on SPLIT(AEAD_DEC(G^nd^nc, AEAD_ENC(G^nc^nd, CONCAT(AEAD_DEC(sd, AEAD_ENC(sd, CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), nil), nil), CONCAT(HASH(n), r)), nil, nil)?).

[illegible][illegible]

116

[illegible]

$\text{msgdc}(\text{AEAD_ENC}(G^{\wedge}\text{nc}^{\wedge}\text{nd}, \text{CONCAT}(\text{CONCAT}(G^{\wedge}\text{d}, \text{HASH}(n), \text{SIGN}(d, \text{HASH}(n))), \text{nfpd}, \text{ifpd}), \text{CONCAT}(\text{HASH}(n), r), \text{nil}))$ is obtained by Attacker.

Deduction • Output of $\text{AEAD_ENC}(\text{nil}, \text{CONCAT}(G^{\wedge}\text{nil}, \text{HASH}(\text{nil}), \text{SIGN}(\text{nil}, \text{HASH}(\text{nil})), \text{nil}, \text{nil}), \text{nil})$ obtained by decomposing $\text{AEAD_DEC}(\text{nil}, \text{AEAD_ENC}(\text{nil}, \text{CONCAT}(G^{\wedge}\text{nil}, \text{HASH}(\text{nil}), \text{SIGN}(\text{nil}, \text{HASH}(\text{nil})), \text{nil}, \text{nil}), \text{nil}), \text{nil})$ with nil .

Deduction • Output of $\text{AEAD_ENC}(G^{\wedge}\text{nil}, \text{CONCAT}(\text{AEAD_DEC}(\text{nil}, \text{AEAD_ENC}(\text{nil}, \text{CONCAT}(G^{\wedge}\text{nil}, \text{HASH}(\text{nil}), \text{SIGN}(\text{nil}, \text{HASH}(\text{nil})), \text{nil}, \text{nil}), \text{nil}), \text{nil}), \text{CONCAT}(\text{HASH}(\text{nil}), \text{nil})), \text{nil})$ obtained by decomposing $\text{AEAD_DEC}(G^{\wedge}\text{nil}, \text{AEAD_ENC}(G^{\wedge}\text{nil}, \text{CONCAT}(\text{AEAD_DEC}(\text{nil}, \text{AEAD_ENC}(\text{nil}, \text{CONCAT}(G^{\wedge}\text{nil}, \text{HASH}(\text{nil}), \text{SIGN}(\text{nil}, \text{HASH}(\text{nil})), \text{nil}, \text{nil}), \text{nil}), \text{nil}), \text{CONCAT}(\text{HASH}(\text{nil}), \text{nil})), \text{nil}), \text{nil})$ with $G^{\wedge}\text{nil}$.

Deduction • Output of $\text{AEAD_ENC}(G^{\wedge}\text{nil}, \text{CONCAT}(\text{SPLIT}(\text{AEAD_DEC}(G^{\wedge}\text{nil}, \text{AEAD_ENC}(G^{\wedge}\text{nil}, \text{CONCAT}(\text{AEAD_DEC}(\text{nil}, \text{AEAD_ENC}(\text{nil}, \text{CONCAT}(G^{\wedge}\text{nil}, \text{HASH}(\text{nil}), \text{SIGN}(\text{nil}, \text{HASH}(\text{nil})), \text{nil}, \text{nil}), \text{nil}), \text{nil}), \text{CONCAT}(\text{HASH}(\text{nil}), \text{nil})), \text{nil}), \text{nil})), \text{CONCAT}(\text{SPLIT}(\text{SPLIT}(\text{AEAD_DEC}(G^{\wedge}\text{nil}, \text{AEAD_ENC}(G^{\wedge}\text{nil}, \text{CONCAT}(\text{AEAD_DEC}(\text{nil}, \text{AEAD_ENC}(\text{nil}, \text{CONCAT}(G^{\wedge}\text{nil}, \text{HASH}(\text{nil}), \text{SIGN}(\text{nil}, \text{HASH}(\text{nil})), \text{nil}, \text{nil}), \text{nil}), \text{nil}), \text{CONCAT}(\text{HASH}(\text{nil}), \text{nil})), \text{nil}), \text{nil})), \text{SPLIT}(\text{SPLIT}(\text{AEAD_DEC}(G^{\wedge}\text{nil}, \text{AEAD_ENC}(G^{\wedge}\text{nil}, \text{CONCAT}(\text{AEAD_DEC}(\text{nil}, \text{AEAD_ENC}(\text{nil}, \text{CONCAT}(G^{\wedge}\text{nil}, \text{HASH}(\text{nil}), \text{SIGN}(\text{nil}, \text{HASH}(\text{nil})), \text{nil}, \text{nil}), \text{nil}), \text{nil}), \text{CONCAT}(\text{HASH}(\text{nil}), \text{nil})), \text{nil}), \text{nil}), \text{CONCAT}(\text{HASH}(\text{nil}), \text{nil})), \text{nil}), \text{nil}), \text{CONCAT}(\text{HASH}(\text{nil}), \text{nil})), \text{nil}), \text{nil}), \text{CONCAT}(\text{SPLIT}(\text{SPLIT}(\text{AEAD_DEC}(G^{\wedge}\text{nil}, \text{AEAD_ENC}(G^{\wedge}\text{nil}, \text{CONCAT}(\text{AEAD_DEC}(\text{nil}, \text{AEAD_ENC}(\text{nil}, \text{CONCAT}(G^{\wedge}\text{nil}, \text{HASH}(\text{nil}), \text{SIGN}(\text{nil}, \text{HASH}(\text{nil})), \text{nil}, \text{nil}), \text{nil}), \text{nil}), \text{CONCAT}(\text{HASH}(\text{nil}), \text{nil})), \text{nil}), \text{nil})), \text{SPLIT}(\text{SPLIT}(\text{AEAD_DEC}(G^{\wedge}\text{nil}, \text{AEAD_ENC}(G^{\wedge}\text{nil}, \text{CONCAT}(\text{AEAD_DEC}(\text{nil}, \text{AEAD_ENC}(\text{nil}, \text{CONCAT}(G^{\wedge}\text{nil}, \text{HASH}(\text{nil}), \text{SIGN}(\text{nil}, \text{HASH}(\text{nil})), \text{nil}, \text{nil}), \text{nil}), \text{nil}), \text{CONCAT}(\text{HASH}(\text{nil}), \text{nil})), \text{nil}), \text{nil}), \text{CONCAT}(\text{HASH}(\text{nil}), \text{nil})), \text{nil}), \text{nil}), \text{nil})$ with $G^{\wedge}\text{nil}$.

Deduction • Output of $\text{AEAD_ENC}(G^{\wedge}\text{na}^{\wedge}\text{nc}, \text{CONCAT}(\text{CONCAT}(G^{\wedge}\text{d}, \text{HASH}(n), \text{SIGN}(d, \text{HASH}(n))), \text{nfpd}, \text{ifpd}), \text{CONCAT}(\text{HASH}(n), r, \text{seq01}), \text{nil})$ obtained by equalizing with the current resolution of msgca .

Result • confidentiality? msgca — When:

```

shac → G^nc^na
shca → G^na^nc
shdc → G^nc^nd
shcd → G^nd^nc
sgd → SIGN(d, HASH(n))
msg0 → CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd)
msg1 → AEAD_ENC(sd, CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), nil)
msg1_de → CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd)
flgr → CONCAT(HASH(n), r)
m0 → CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), CONCAT(HASH(n), r))
msgdc → AEAD_ENC(G^nc^nd, CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), CONCAT(HASH(n), r), nil)
msgdc_de → CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), CONCAT(HASH(n), r))
msg2 → CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd)
flgr1 → CONCAT(HASH(n), r)
subj1 → HASH(n)
r1 → r
unnamed_0 → ASSERT(HASH(n), HASH(n))?
flgr2 → CONCAT(HASH(n), r, seq01)
m1 → CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), CONCAT(HASH(n), r, seq01))
msgca → AEAD_ENC(G^na^nc, CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), CONCAT(HASH(n), r, seq01)), nil)

```

← obtained by Attacker

```

msgca_de → CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), CONCAT(HASH(n), r, seq01))
msg3 → CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd)
flgr3 → CONCAT(HASH(n), r, seq01)
subj2 → HASH(n)
r2 → r
seq01_1 → seq01
gd_ → G^d
subj4 → HASH(n)
sgd_ → SIGN(d, HASH(n))
nfpd_ → nfpd
ifpd_ → ifpd
proof → nil

```

Appendix

msgca (AEAD_ENC($G^{na^{nc}}$, CONCAT(CONCAT(G^d , HASH(n), SIGN(d , HASH(n)), nfpd, ifpd), CONCAT(HASH(n), r , seq01)), nil)) is obtained by Attacker.

Deduction • G^{nil} obtained as a concatenated fragment of CONCAT(G^{nil} , HASH(nil), SIGN(nil , HASH(nil)), nil , nil).

Analysis • Initializing Stage 1 mutation map for Na...

Deduction • Output of CONCAT(CONCAT(G^{nil} , HASH(nil), SIGN(nil , HASH(nil)), nil , nil), CONCAT(HASH(nil), nil)) obtained by reconstructing with CONCAT(G^{nil} , HASH(nil), SIGN(nil , HASH(nil)), nil , nil), CONCAT(HASH(nil), nil). (Analysis 7)

Deduction • Output of AEAD_ENC(G^{nil} , CONCAT(CONCAT(G^{nil} , HASH(nil), SIGN(nil , HASH(nil)), nil , nil), CONCAT(HASH(nil), nil)), nil) obtained by reconstructing with G^{nil} , CONCAT(CONCAT(G^{nil} , HASH(nil), SIGN(nil , HASH(nil)), nil , nil), CONCAT(HASH(nil), nil)), nil . (Analysis 7)

Analysis • Initializing Stage 1 mutation map for Nc...

Analysis • Initializing Stage 1 mutation map for Nd...

Analysis • Initializing Stage 1 mutation map for Della...

Analysis • Initializing Stage 3 mutation map for Na...

Analysis • Initializing Stage 2 mutation map for Na...

Analysis • Initializing Stage 2 mutation map for Nc...

Analysis • Initializing Stage 2 mutation map for Nd...

Analysis • Initializing Stage 2 mutation map for Della...

Analysis • Initializing Stage 3 mutation map for Nc...

Analysis • Initializing Stage 3 mutation map for Nd...

Analysis • Initializing Stage 3 mutation map for Della...

Analysis • Initializing Stage 5 mutation map for Na...

Analysis • Initializing Stage 4 mutation map for Na...

Deduction • Output of CONCAT(G^{na} , HASH(nil), SIGN(nil , HASH(nil)), nil , nil) obtained by reconstructing with G^{na} , HASH(nil), SIGN(nil , HASH(nil)), nil , nil . (Analysis 84)

Deduction • Output of AEAD_ENC(nil , CONCAT(G^{na} , HASH(nil), SIGN(nil , HASH(nil)), nil , nil), nil) obtained by reconstructing with nil , CONCAT(G^{na} , HASH(nil), SIGN(nil , HASH(nil)), nil , nil), nil . (Analysis 84)

Deduction • Output of CONCAT(G^{nc} , HASH(nil), SIGN(nil , HASH(nil)), nil , nil) obtained by reconstructing with G^{nc} , HASH(nil), SIGN(nil , HASH(nil)), nil , nil . (Analysis 95)

Deduction • Output of AEAD_ENC(nil , CONCAT(G^{nc} , HASH(nil), SIGN(nil , HASH(nil)), nil , nil), nil) obtained by reconstructing with nil , CONCAT(G^{nc} , HASH(nil), SIGN(nil , HASH(nil)), nil , nil), nil . (Analysis 95)

Deduction • Output of CONCAT(G^{nd} , HASH(nil), SIGN(nil , HASH(nil)), nil , nil) obtained by reconstructing with G^{nd} , HASH(nil), SIGN(nil , HASH(nil)), nil , nil . (Analysis 109)

Deduction • Output of AEAD_ENC(nil , CONCAT(G^{nd} , HASH(nil), SIGN(nil , HASH(nil)), nil , nil), nil) obtained by reconstructing with nil , CONCAT(G^{nd} , HASH(nil), SIGN(nil , HASH(nil)), nil , nil), nil . (Analysis 109)

Analysis • Initializing Stage 4 mutation map for Nc...

Analysis • Initializing Stage 4 mutation map for Nd...

Analysis • Initializing Stage 4 mutation map for Della...

Analysis • Initializing Stage 5 mutation map for Nc...

Analysis • Initializing Stage 5 mutation map for Nd...

Analysis • Initializing Stage 5 mutation map for Della...

Analysis • Initializing Stage 6 mutation map for Na...

Analysis • Initializing Stage 6 mutation map for Nc...

Analysis • Initializing Stage 6 mutation map for Nd...

Analysis • Initializing Stage 6 mutation map for Della...

Verifpal • Verification completed for 'MIT.vp' at 11:56:17 AM.

Verifpal • Summary of failed queries will follow.

Result • confidentiality? msgdc — When:

shac $\rightarrow G^{nc^{na}}$

shca $\rightarrow G^{na^{nc}}$

shdc $\rightarrow G^{nc^{nd}}$

shcd $\rightarrow G^{nd^{nc}}$

sgd $\rightarrow \text{SIGN}(d, \text{HASH}(n))$

```

msg0 → CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd)
msg1 → AEAD_ENC(sd, CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), nil)
msg1_de → CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd)
flgr → CONCAT(HASH(n), r)
m0 → CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), CONCAT(HASH(n), r))
msgdc → AEAD_ENC(G^nc^nd, CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), CONCAT(HASH(n), r)), nil) ←

```

obtained by Attacker

```

msgdc_de → CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), CONCAT(HASH(n), r))
msg2 → CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd)
flgr1 → CONCAT(HASH(n), r)
subj1 → HASH(n)
r1 → r
unnamed_0 → ASSERT(HASH(n), HASH(n))?
flgr2 → CONCAT(HASH(n), r, seq01)
m1 → CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), CONCAT(HASH(n), r, seq01))
msgca → AEAD_ENC(G^na^nc, CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), CONCAT(HASH(n), r, seq01)), nil)
msgca_de → CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), CONCAT(HASH(n), r, seq01))
msg3 → CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd)
flgr3 → CONCAT(HASH(n), r, seq01)
subj2 → HASH(n)
r2 → r
seq01_1 → seq01
gd_ → G^d
subj4 → HASH(n)
sgd_ → SIGN(d, HASH(n))
nfpd_ → nfpd
ifpd_ → ifpd
proof → nil
msgdc (AEAD_ENC(G^nc^nd, CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), CONCAT(HASH(n), r)), nil)) is

```

obtained by Attacker.

Result • confidentiality? msgca — When:

```

shac → G^nc^na
shca → G^na^nc
shdc → G^nc^nd
shcd → G^nd^nc
sgd → SIGN(d, HASH(n))
msg0 → CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd)
msg1 → AEAD_ENC(sd, CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), nil)
msg1_de → CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd)
flgr → CONCAT(HASH(n), r)
m0 → CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), CONCAT(HASH(n), r))
msgdc → AEAD_ENC(G^nc^nd, CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), CONCAT(HASH(n), r)), nil)
msgdc_de → CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), CONCAT(HASH(n), r))
msg2 → CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd)
flgr1 → CONCAT(HASH(n), r)
subj1 → HASH(n)
r1 → r
unnamed_0 → ASSERT(HASH(n), HASH(n))?
flgr2 → CONCAT(HASH(n), r, seq01)
m1 → CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), CONCAT(HASH(n), r, seq01))
msgca → AEAD_ENC(G^na^nc, CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), CONCAT(HASH(n), r, seq01)), nil)

```

← obtained by Attacker

```

msgca_de → CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), CONCAT(HASH(n), r, seq01))
msg3 → CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd)
flgr3 → CONCAT(HASH(n), r, seq01)

```



```
subj2 → HASH(n)
r2 → r
seq01_1 → seq01
gd_ → G^d
subj4 → HASH(n)
sgd_ → SIGN(d, HASH(n))
nfpd_ → nfpd
ifpd_ → ifpd
proof → nil
msgca (AEAD_ENC(G^na^nc, CONCAT(CONCAT(G^d, HASH(n), SIGN(d, HASH(n)), nfpd, ifpd), CONCAT(HASH(n), r, seq01)), nil)) is
obtained by Attacker.
```

Verifpal • Thank you for using Verifpal.

File Name: User_space_Message.vp

```
attacker[active]
```

```
principal NA[
  knows private shAC,sub,s,c
]
```

```
principal NC[
  knows private shAC,shCD,sub,s
  generates seq01
]
```

```
principal ND[
  knows private shCD,sub,s
]
```

```
principal Alice[
  knows private a,sub,s,c
  ga = G^a
]
Alice->Della:[ga]
```

```
principal Della[
  knows private d,sub,s
  gd = g^d
  sk1 = ga^d
]
Della->Alice:[gd]
```

```
principal Alice[
  generates epk
  sk = gd^a
]
```

```

        en = AEAD_ENC(sk,epk,c)
    ]
    Alice->NA:[en]

principal NA[
    hd1 = CONCAT(sub,s)
    h = AEAD_ENC(shAC,hd1,c)
    m = CONCAT(h,en,c)
]

NA->NC:[m]

principal NC[
    h_en_c_ = SPLIT(m)
    h_de = AEAD_DEC(shAC,h_c_)?
    _ = ASSERT(CONCAT(sub,s),h_de)?
    hd2 = CONCAT(h_de,seq01)
    h2 = AEAD_ENC(shCD,hd2,c_)
]
NC->ND:[h2],[en_],[c_]

principal ND[
    h2_de = AEAD_DEC(shCD,h2,c_)?
    h_de_1,seq01_ = SPLIT(h2_de)
    sub_s_ = SPLIT(h_de_1)
    _ = ASSERT(s,s_)?
]

ND->Della:[en_],[c_]

principal Della[
    en_de = AEAD_DEC(sk1,en_,c_)?
]

queries[
    authentication? NA->NC:m
    authentication? NC->ND:h2
    equivalence? sub_sub
    equivalence? sk,sk1

```

confidentiality? epk
confidentiality? c
equivalence? c,c_
equivalence? epk, en_de
]

The Result

Run In terminal # verifpal verify User_space_Message.vp

Verifpal 0.26.1 - <https://verifpal.com>

Warning • Verifpal is Beta software.

Verifpal • Parsing model 'User_space_Message.vp'...

Verifpal • Verification initiated for 'User_space_Message.vp' at 12:23:47 PM.

Info • Attacker is configured as active.

Info • Running at phase 0.

Analysis • Constructed skeleton CONCAT(nil, nil) based on CONCAT(sub, s).

Analysis • Constructed skeleton AEAD_ENC(nil, CONCAT(nil, nil), nil) based on AEAD_ENC(shac, CONCAT(sub, s), c).

Analysis • Constructed skeleton SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(G^nil, nil, nil), nil)) based on SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(G^d^a, epk, c), c)).

Analysis • Constructed skeleton AEAD_DEC(nil, SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(G^nil, nil, nil), nil)), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(G^nil, nil, nil), nil))) based on AEAD_DEC(shac, SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(G^d^a, epk, c), c)), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(G^d^a, epk, c), c)))?.

Analysis • Constructed skeleton ASSERT(CONCAT(nil, nil), AEAD_DEC(nil, SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(G^nil, nil, nil), nil)), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(G^nil, nil, nil), nil))) based on ASSERT(CONCAT(sub, s), AEAD_DEC(shac, SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(G^d^a, epk, c), c)), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(G^d^a, epk, c), c)))?.

Analysis • Constructed skeleton CONCAT(AEAD_DEC(nil, SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(G^nil, nil, nil), nil)), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(G^nil, nil, nil), nil))), nil) based on CONCAT(AEAD_DEC(shac, SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(G^d^a, epk, c), c)), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(G^d^a, epk, c), c)))?, seq01).

Analysis • Constructed skeleton AEAD_DEC(nil, AEAD_ENC(nil, CONCAT(AEAD_DEC(nil, SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(G^nil, nil, nil), nil)), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(G^nil, nil, nil), nil))), nil), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(G^nil, nil, nil), nil))), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(G^nil, nil, nil), nil))) based on AEAD_DEC(shed, AEAD_ENC(shed, CONCAT(AEAD_DEC(shac, SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(G^d^a, epk, c), c)), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(G^d^a, epk, c), c)))?, seq01), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(G^d^a, epk, c), c))), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(G^d^a, epk, c), c)))?.

Analysis • Constructed skeleton SPLIT(AEAD_DEC(nil, AEAD_ENC(nil, CONCAT(AEAD_DEC(nil, SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(G^nil, nil, nil), nil)), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(G^nil, nil, nil), nil))), nil), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(G^nil, nil, nil), nil))), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(G^nil, nil, nil), nil))) based on SPLIT(AEAD_DEC(shed, AEAD_ENC(shed, CONCAT(AEAD_DEC(shac, SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(G^d^a, epk, c), c)), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(G^d^a, epk, c), c)))?, seq01), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(G^d^a, epk, c), c))), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(G^d^a, epk, c), c)))?.

Analysis • Constructed skeleton SPLIT(AEAD_DEC(nil, AEAD_ENC(nil, CONCAT(AEAD_DEC(nil, SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(G^nil, nil, nil), nil)), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(G^nil, nil, nil), nil))), nil), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(G^nil, nil, nil), nil))), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(G^nil, nil, nil), nil))) based on SPLIT(AEAD_DEC(shed, AEAD_ENC(shed, CONCAT(AEAD_DEC(shac, SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(G^d^a, epk, c), c)), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(G^d^a, epk, c), c)))?, seq01), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(G^d^a, epk, c), c))), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(G^d^a, epk, c), c)))?.

nil), AEAD_ENC(G^{nil} , nil, nil, nil))), nil), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(G^{nil} , nil, nil, nil))), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(G^{nil} , nil, nil, nil)))) with nil.

Deduction • Second output of SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(G^{nil} , nil, nil, nil))) obtained by decomposing AEAD_DEC(G^{nil} , SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(G^{nil} , nil, nil, nil))), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(G^{nil} , nil, nil, nil)))) with G^{nil} .

Deduction • Output of AEAD_ENC(shcd, CONCAT(CONCAT(sub, s), seq01), c) obtained by equalizing with the current resolution of h2.

Analysis • Initializing Stage 1 mutation map for Na...

Deduction • G^{nil} obtained by reconstructing with nil. (Analysis 5)

Analysis • Initializing Stage 1 mutation map for Nc...

Analysis • Initializing Stage 1 mutation map for Nd...

Analysis • Initializing Stage 1 mutation map for Alice...

Analysis • Initializing Stage 1 mutation map for Della...

Analysis • Initializing Stage 3 mutation map for Na...

Analysis • Initializing Stage 2 mutation map for Na...

Analysis • Initializing Stage 3 mutation map for Nc...

Analysis • Initializing Stage 2 mutation map for Nc...

Analysis • Initializing Stage 2 mutation map for Nd...

Analysis • Initializing Stage 3 mutation map for Nd...

Analysis • Initializing Stage 2 mutation map for Alice...

Analysis • Initializing Stage 2 mutation map for Della...

Deduction • Output of CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(G^{d^a} , epk, c), nil) obtained by reconstructing with AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(G^{d^a} , epk, c), nil. (Analysis 25)

Deduction • Output of CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(G^{d^a} , epk, c), c) obtained by reconstructing with AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(G^{d^a} , epk, c), c. (Analysis 27)

Deduction • Output of CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(G^{d^a} , epk, c), nil) obtained by reconstructing with AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(G^{d^a} , epk, c), nil. (Analysis 29)

Analysis • Initializing Stage 3 mutation map for Alice...

Analysis • Initializing Stage 3 mutation map for Della...

Analysis • Initializing Stage 5 mutation map for Na...

Analysis • Initializing Stage 4 mutation map for Na...

Analysis • Initializing Stage 5 mutation map for Nc...

Analysis • Initializing Stage 4 mutation map for Nc...

Analysis • Initializing Stage 4 mutation map for Nd...

Analysis • Initializing Stage 5 mutation map for Nd...

Deduction • Output of AEAD_ENC(nil, CONCAT(nil, nil), c) obtained by reconstructing with nil, CONCAT(nil, nil), c. (Analysis 74)

Deduction • Output of CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), c), AEAD_ENC(G^{d^a} , epk, c), nil) obtained by reconstructing with AEAD_ENC(nil, CONCAT(nil, nil), c), AEAD_ENC(G^{d^a} , epk, c), nil. (Analysis 74)

Deduction • Output of CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), c), AEAD_ENC(G^{d^a} , epk, c), c) obtained by reconstructing with AEAD_ENC(nil, CONCAT(nil, nil), c), AEAD_ENC(G^{d^a} , epk, c), c. (Analysis 74)

Deduction • Output of AEAD_ENC(c, CONCAT(nil, nil), nil) obtained by reconstructing with c, CONCAT(nil, nil), nil. (Analysis 76)

Deduction • Output of CONCAT(AEAD_ENC(c, CONCAT(nil, nil), nil), AEAD_ENC(G^{d^a} , epk, c), nil) obtained by reconstructing with AEAD_ENC(c, CONCAT(nil, nil), nil), AEAD_ENC(G^{d^a} , epk, c), nil. (Analysis 77)

Deduction • Output of CONCAT(AEAD_ENC(c, CONCAT(nil, nil), nil), AEAD_ENC(G^{d^a} , epk, c), c) obtained by reconstructing with AEAD_ENC(c, CONCAT(nil, nil), nil), AEAD_ENC(G^{d^a} , epk, c), c. (Analysis 79)

Deduction • Output of AEAD_ENC(c, CONCAT(nil, nil), c) obtained by reconstructing with c, CONCAT(nil, nil), c. (Analysis 82)

Deduction • Output of CONCAT(AEAD_ENC(c, CONCAT(nil, nil), c), AEAD_ENC(G^{d^a} , epk, c), nil) obtained by reconstructing with AEAD_ENC(c, CONCAT(nil, nil), c), AEAD_ENC(G^{d^a} , epk, c), nil. (Analysis 82)

Deduction • Output of CONCAT(AEAD_ENC(c, CONCAT(nil, nil), c), AEAD_ENC(G^{d^a} , epk, c), c) obtained by reconstructing with AEAD_ENC(c, CONCAT(nil, nil), c), AEAD_ENC(G^{d^a} , epk, c), c. (Analysis 84)

Analysis • Initializing Stage 4 mutation map for Alice...

Analysis • Initializing Stage 4 mutation map for Della...

Analysis • Initializing Stage 5 mutation map for Alice...

Analysis • Initializing Stage 5 mutation map for Della...

Analysis • Initializing Stage 6 mutation map for Na...

Analysis • Initializing Stage 6 mutation map for Nc...

Analysis • Initializing Stage 6 mutation map for Nd...

127

Deduction • Output of $\text{CONCAT}(\text{AEAD_ENC}(c, \text{CONCAT}(c, c), \text{nil}), \text{AEAD_ENC}(G^{d^a}, \text{epk}, c), c)$ obtained by reconstructing with $\text{AEAD_ENC}(c, \text{CONCAT}(c, c), \text{nil}), \text{AEAD_ENC}(G^{d^a}, \text{epk}, c), c$. (Analysis 287)

Deduction • Output of $\text{AEAD_ENC}(c, \text{CONCAT}(c, c), c)$ obtained by reconstructing with $c, \text{CONCAT}(c, c), c$. (Analysis 287)

Deduction • Output of $\text{CONCAT}(\text{AEAD_ENC}(c, \text{CONCAT}(c, c), c), \text{AEAD_ENC}(G^{d^a}, \text{epk}, c), \text{nil})$ obtained by reconstructing with $\text{AEAD_ENC}(c, \text{CONCAT}(c, c), c), \text{AEAD_ENC}(G^{d^a}, \text{epk}, c), \text{nil}$. (Analysis 287)

Deduction • Output of $\text{CONCAT}(\text{AEAD_ENC}(c, \text{CONCAT}(c, c), c), \text{AEAD_ENC}(G^{d^a}, \text{epk}, c), c)$ obtained by reconstructing with $\text{AEAD_ENC}(c, \text{CONCAT}(c, c), c), \text{AEAD_ENC}(G^{d^a}, \text{epk}, c), c$. (Analysis 287)

Analysis • Initializing Stage 6 mutation map for Alice...

Analysis • Initializing Stage 6 mutation map for Della...

Stage 6, Analysis 600...

Verifpal • Verification completed for 'User_space_Message.vp' at 12:23:47 PM.

Verifpal • Summary of failed queries will follow.

Result • confidentiality? c — When:

```
sk1 → G^a^d
sk → G^d^a
en → AEAD_ENC(G^d^a, epk, c)
h → AEAD_ENC(shac, CONCAT(sub, s), c)
m → CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(G^d^a, epk, c), c)
h_ → AEAD_ENC(shac, CONCAT(sub, s), c)
en_ → AEAD_ENC(G^d^a, epk, c)
c_ → c ← obtained by Attacker
h_de → CONCAT(sub, s)
unnamed_0 → ASSERT(CONCAT(sub, s), CONCAT(sub, s))?
hd2 → CONCAT(CONCAT(sub, s), seq01)
h2 → AEAD_ENC(shcd, CONCAT(CONCAT(sub, s), seq01), c)
h2_de → CONCAT(CONCAT(sub, s), seq01)
h_de_1 → CONCAT(sub, s)
seq01_ → seq01
sub_ → sub
s_ → s
unnamed_1 → ASSERT(s, s)?
en_de → epk
c (c) is obtained by Attacker.
```

Verifpal • Thank you for using Verifpal.

File Name: E2EE_with_Multiple_Recivers.vp

attacker[active]

```
principal Alice[
  knows private epk,sub,s,c
  generates up
  en = AEAD_ENC(epk,up,c)
]
Alice->NA:[en]
```

principal NA[

```

    knows private shAC,sub,s,c
    hd1 = CONCAT(sub,s)
    h = AEAD_ENC(shAC,hd1,c)
    m = CONCAT(h,en,c)
]

NA->NC:[m]

principal NC[
    knows private shAC,shCD,sub,s
    generates seq01
    h_en_c_ = SPLIT(m)
    h_de = AEAD_DEC(shAC,h_c_)?
    _ = ASSERT(CONCAT(sub,s),h_de)?
    hd2 = CONCAT(h_de,seq01)
    h2 = AEAD_ENC(shCD,hd2,c_)
]

NC->ND:[h2],[en_],[c_]

principal ND[
    knows private shCD,sub,s
    h2_de = AEAD_DEC(shCD,h2,c_)?
    h_de_1,seq01_ = SPLIT(h2_de)
    sub_s_ = SPLIT(h_de_1)
    _ = ASSERT(s,s_)?
]

ND->Della:[en_],[c_]

principal Della[
    knows private epk,sub,s
    en_de = AEAD_DEC(epk,en_,c_)?
]

ND->Dory:[en_],[c_]

principal Dory[
    knows private epk,sub,s
    en_de1 = AEAD_DEC(epk,en_,c_)?
]

```



```
queries[
  authentication? NA->NC:m
  authentication? NC->ND:h2
  equivalence? sub_,sub
  authentication? ND->Dory:en_
  confidentiality?en_
  confidentiality? up
  confidentiality? c
  equivalence? c,c_
]
```

The Result

Run In terminal # verifpal verify E2EE_with_Multiple_Recivers.vp

Verifpal 0.26.1 - <https://verifpal.com>

Warning • Verifpal is Beta software.

Verifpal • Parsing model 'E2EE_with_Multiple_Recivers.vp'...

Verifpal • Verification initiated for 'E2EE_with_Multiple_Recivers.vp' at 12:37:17 PM.

Info • Attacker is configured as active.

Info • Running at phase 0.

Analysis • Constructed skeleton CONCAT(nil, nil) based on CONCAT(sub, s).

Analysis • Constructed skeleton AEAD_ENC(nil, CONCAT(nil, nil), nil) based on AEAD_ENC(shac, CONCAT(sub, s), c).

Analysis • Constructed skeleton SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil)) based on SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c), c)).

Analysis • Constructed skeleton AEAD_DEC(nil, SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil)), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil))) based on AEAD_DEC(shac, SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c), c)), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c), c)))?.

Analysis • Constructed skeleton ASSERT(CONCAT(nil, nil), AEAD_DEC(nil, SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil)), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil))) based on ASSERT(CONCAT(sub, s), AEAD_DEC(shac, SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c), c)), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c), c)))?).

Analysis • Constructed skeleton CONCAT(AEAD_DEC(nil, SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil)), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil))) based on CONCAT(AEAD_DEC(shac, SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c), c)), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c), c)))?, seq01).

Analysis • Constructed skeleton AEAD_DEC(nil, AEAD_ENC(nil, CONCAT(AEAD_DEC(nil, SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil)), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil))), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil))) based on AEAD_DEC(shac, AEAD_ENC(shac, CONCAT(AEAD_DEC(shac, SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c), c)), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c), c)))?, seq01), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c), c)), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c), c)))?.

Analysis • Constructed skeleton SPLIT(AEAD_DEC(nil, AEAD_ENC(nil, CONCAT(AEAD_DEC(nil, SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil)), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil))), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil))) based on SPLIT(AEAD_DEC(shac, AEAD_ENC(shac, CONCAT(AEAD_DEC(shac, SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c), c)), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c), c)))?, seq01), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c), c)), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c), c)))?.

AEAD_ENC(epk, up, c, c)))?, seq01), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c, c))), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c, c)))?).

Analysis • Constructed skeleton SPLIT(AEAD_DEC(nil, AEAD_ENC(nil, CONCAT(AEAD_DEC(nil, SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil)), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil))), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil))), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil))) based on SPLIT(AEAD_DEC(shcd, AEAD_ENC(shcd, CONCAT(AEAD_DEC(shac, SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c, c))), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c, c)))?, seq01), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c, c))), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c, c)))?).

Analysis • Constructed skeleton SPLIT(SPLIT(AEAD_DEC(nil, AEAD_ENC(nil, CONCAT(AEAD_DEC(nil, SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil)), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil))), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil))), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil))) based on SPLIT(SPLIT(AEAD_DEC(shcd, AEAD_ENC(shcd, CONCAT(AEAD_DEC(shac, SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c, c))), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c, c)))?, seq01), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c, c))), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c, c)))?).

Analysis • Constructed skeleton SPLIT(SPLIT(AEAD_DEC(nil, AEAD_ENC(nil, CONCAT(AEAD_DEC(nil, SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil)), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil))), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil))) based on SPLIT(SPLIT(AEAD_DEC(shcd, AEAD_ENC(shcd, CONCAT(AEAD_DEC(shac, SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c, c))), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c, c)))?, seq01), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c, c))), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c, c)))?).

Analysis • Constructed skeleton ASSERT(nil, SPLIT(SPLIT(AEAD_DEC(nil, AEAD_ENC(nil, CONCAT(AEAD_DEC(nil, SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil)), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil))), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil))) based on ASSERT(s, SPLIT(SPLIT(AEAD_DEC(shcd, AEAD_ENC(shcd, CONCAT(AEAD_DEC(shac, SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c, c))), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c, c)))?, seq01), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c, c))), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c, c)))?).

Analysis • Constructed skeleton AEAD_DEC(nil, SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil)), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil))) based on AEAD_DEC(epk, SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c, c))), SPLIT(CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c, c)))?).

Result • confidentiality? en_ —
en_ (AEAD_ENC(epk, up, c)) is obtained by Attacker.

Deduction • Output of AEAD_ENC(shac, CONCAT(sub, s), c) obtained as a concatenated fragment of CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c), c).

Deduction • c obtained as a concatenated fragment of CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c), c).

Result • confidentiality? c — When:

h → AEAD_ENC(shac, CONCAT(sub, s), c)
m → CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c), c)
h_ → AEAD_ENC(shac, CONCAT(sub, s), c)
en_ → AEAD_ENC(epk, up, c)
c_ → c ← obtained by Attacker
h_de → CONCAT(sub, s)
unnamed_0 → ASSERT(CONCAT(sub, s), CONCAT(sub, s))?
hd2 → CONCAT(CONCAT(sub, s), seq01)
h2 → AEAD_ENC(shcd, CONCAT(CONCAT(sub, s), seq01), c)
h2_de → CONCAT(CONCAT(sub, s), seq01)
h_de_1 → CONCAT(sub, s)
seq01_ → seq01
sub_ → sub
s_ → s
unnamed_1 → ASSERT(s, s)?

en_de → up
en_del → up
c (c) is obtained by Attacker.

Deduction • Output of AEAD_ENC(nil, CONCAT(AEAD_DEC(nil, SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil))), nil), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil))), nil), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil))), nil), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil))) obtained by decomposing AEAD_DEC(nil, AEAD_ENC(nil, CONCAT(AEAD_DEC(nil, SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil))), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil))), nil), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil))) with nil.

Deduction • Second output of SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil)) obtained by decomposing AEAD_DEC(nil, SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil)), SPLIT(CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(nil, nil, nil), nil))) with nil.

Deduction • Output of AEAD_ENC(shcd, CONCAT(CONCAT(sub, s), seq01), c) obtained by equivalizing with the current resolution of h2.

Analysis • Initializing Stage 1 mutation map for Alice...

Analysis • Initializing Stage 1 mutation map for Na...

Analysis • Initializing Stage 1 mutation map for Nc...

Analysis • Initializing Stage 1 mutation map for Nd...

Analysis • Initializing Stage 1 mutation map for Della...

Analysis • Initializing Stage 1 mutation map for Dory...

Analysis • Initializing Stage 3 mutation map for Alice...

Analysis • Initializing Stage 3 mutation map for Na...

Analysis • Initializing Stage 3 mutation map for Nc...

Analysis • Initializing Stage 2 mutation map for Alice...

Analysis • Initializing Stage 2 mutation map for Na...

Analysis • Initializing Stage 2 mutation map for Nc...

Deduction • Output of AEAD_ENC(nil, nil, nil) obtained by reconstructing with nil, nil, nil. (Analysis 20)

Deduction • Output of AEAD_ENC(nil, nil, c) obtained by reconstructing with nil, nil, c. (Analysis 21)

Deduction • Output of AEAD_ENC(nil, c, c) obtained by reconstructing with nil, c, c. (Analysis 22)

Deduction • Output of AEAD_ENC(c, c, nil) obtained by reconstructing with c, c, nil. (Analysis 26)

Deduction • Output of AEAD_ENC(c, nil, nil) obtained by reconstructing with c, nil, nil. (Analysis 22)

Deduction • Output of AEAD_ENC(nil, c, nil) obtained by reconstructing with nil, c, nil. (Analysis 21)

Stage 2-3, Analysis 34...Deduction • Output of AEAD_ENC(c, c, c) obtained by reconstructing with c, c, c. (Analysis 26)

Deduction • Output of AEAD_ENC(c, nil, c) obtained by reconstructing with c, nil, c. (Analysis 26)

Analysis • Initializing Stage 2 mutation map for Nd...

Analysis • Initializing Stage 3 mutation map for Nd...

Deduction • Output of CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(epk, up, c), nil) obtained by reconstructing with AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(epk, up, c), nil. (Analysis 40)

Deduction • Output of CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(epk, up, c), c) obtained by reconstructing with AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(epk, up, c), c. (Analysis 47)

Deduction • Output of CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(shcd, CONCAT(CONCAT(sub, s), seq01), c), nil) obtained by reconstructing with AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(shcd, CONCAT(CONCAT(sub, s), seq01), c), nil. (Analysis 50)

Deduction • Output of CONCAT(AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(shcd, CONCAT(CONCAT(sub, s), seq01), c), c) obtained by reconstructing with AEAD_ENC(nil, CONCAT(nil, nil), nil), AEAD_ENC(shcd, CONCAT(CONCAT(sub, s), seq01), c), c. (Analysis 51)

Deduction • Output of CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c), nil) obtained by reconstructing with AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c), nil. (Analysis 54)

Deduction • Output of CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(shcd, CONCAT(CONCAT(sub, s), seq01), c), nil) obtained by reconstructing with AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(shcd, CONCAT(CONCAT(sub, s), seq01), c), nil. (Analysis 57)

Deduction • Output of CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(shcd, CONCAT(CONCAT(sub, s), seq01), c), c) obtained by reconstructing with AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(shcd, CONCAT(CONCAT(sub, s), seq01), c), c. (Analysis 59)

Analysis • Initializing Stage 2 mutation map for Della...

Analysis • Initializing Stage 2 mutation map for Dory...

Analysis • Initializing Stage 3 mutation map for Della...

Analysis • Initializing Stage 3 mutation map for Dory...

Analysis • Initializing Stage 5 mutation map for Alice...

Analysis • Initializing Stage 4 mutation map for Alice...

133

134

135

136

137

138

139

140

141

142

143

Deduction • Output of `CONCAT(AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(nil, nil, nil), nil)` obtained by reconstructing with `AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(nil, nil, nil), nil`. (Analysis 8802)

Deduction • Output of `CONCAT(AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(nil, nil, c), c)` obtained by reconstructing with `AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(nil, nil, c), c`. (Analysis 8804)

Deduction • Output of `CONCAT(AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(nil, c, nil), nil)` obtained by reconstructing with `AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(nil, c, nil), nil`. (Analysis 8805)

Deduction • Output of `CONCAT(AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(nil, c, nil), c)` obtained by reconstructing with `AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(nil, c, nil), c`. (Analysis 8807)

Deduction • Output of `CONCAT(AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(nil, c, c), c)` obtained by reconstructing with `AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(nil, c, c), c`. (Analysis 8807)

Deduction • Output of `CONCAT(AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(c, nil, nil), nil)` obtained by reconstructing with `AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(c, nil, nil), nil`. (Analysis 8807)

Deduction • Output of `CONCAT(AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(nil, c, c), nil)` obtained by reconstructing with `AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(nil, c, c), nil`. (Analysis 8807)

Deduction • Output of `CONCAT(AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(c, nil, nil), c)` obtained by reconstructing with `AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(c, nil, nil), c`. (Analysis 8810)

Deduction • Output of `CONCAT(AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(c, c, nil), nil)` obtained by reconstructing with `AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(c, c, nil), nil`. (Analysis 8811)

Deduction • Output of `CONCAT(AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(c, c, nil), c)` obtained by reconstructing with `AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(c, c, nil), c`. (Analysis 8812)

Deduction • Output of `CONCAT(AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(c, c, c), nil)` obtained by reconstructing with `AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(c, c, c), nil`. (Analysis 8812)

Deduction • Output of `CONCAT(AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(c, nil, c), c)` obtained by reconstructing with `AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(c, nil, c), c`. (Analysis 8812)

Deduction • Output of `CONCAT(AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(shcd, CONCAT(CONCAT(sub, s), seq01), c), c)` obtained by reconstructing with `AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(shcd, CONCAT(CONCAT(sub, s), seq01), c), c`. (Analysis 8812)

Deduction • Output of `CONCAT(AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(shcd, CONCAT(CONCAT(sub, s), seq01), c), nil)` obtained by reconstructing with `AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(shcd, CONCAT(CONCAT(sub, s), seq01), c), nil`. (Analysis 8813)

Deduction • Output of `CONCAT(AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(c, nil, c), nil)` obtained by reconstructing with `AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(c, nil, c), nil`. (Analysis 8814)

Deduction • Output of `CONCAT(AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(c, c, c), c)` obtained by reconstructing with `AEAD_ENC(c, CONCAT(c, c), c), AEAD_ENC(c, c, c), c`. (Analysis 8815)

Analysis • Initializing Stage 6 mutation map for Della...

Analysis • Initializing Stage 6 mutation map for Dory...

Stage 6, Analysis 31000...

Verifpal • Verification completed for 'E2EE_with_Multiple_Recivers.vp' at 12:37:32 PM.

Verifpal • Summary of failed queries will follow.

Result • confidentiality? `en_` —
`en_ (AEAD_ENC(epk, up, c))` is obtained by Attacker.

Result • confidentiality? `c` — When:

- `h` → `AEAD_ENC(shac, CONCAT(sub, s), c)`
- `m` → `CONCAT(AEAD_ENC(shac, CONCAT(sub, s), c), AEAD_ENC(epk, up, c), c)`
- `h_` → `AEAD_ENC(shac, CONCAT(sub, s), c)`
- `en_` → `AEAD_ENC(epk, up, c)`
- `c_` → `c` ← obtained by Attacker
- `h_de` → `CONCAT(sub, s)`
- `unnamed_0` → `ASSERT(CONCAT(sub, s), CONCAT(sub, s))?`
- `hd2` → `CONCAT(CONCAT(sub, s), seq01)`
- `h2` → `AEAD_ENC(shcd, CONCAT(CONCAT(sub, s), seq01), c)`
- `h2_de` → `CONCAT(CONCAT(sub, s), seq01)`
- `h_de_1` → `CONCAT(sub, s)`
- `seq01_` → `seq01`
- `sub_` → `sub`


```
s_ → s
unnamed_1 → ASSERT(s, s)?
en_de → up
en_de1 → up
c (c) is obtained by Attacker.
```

Verifpal • Thank you for using Verifpal.

File Name: MQTT_Model.vp

```
attacker[active]
principal Broker[
  knows private hn,usnP,pwd //hn=hostname of
]
principal Publisher[
  knows private hn, usnP,pwd,psk1
  cnt1 = CONCAT(usnP,pwd)
]
Publisher->Broker:cnt1
principal broker[
  usnP_,pwd_ = SPLIT(cnt1)
  generates salt1
  h1 = HASH(pwd_,salt1)
  _ = ASSERT(h1,HASH(pwd,salt1))?
]
principal Publisher[
  generates tp1,tp2,aa
]
Publisher->Broker: tp1, tp2 // it transmitted like tp1/tp2
principal Client[
  knows private hn, usnC1, pwd1,psk1
]
Client-> Broker:usnC1, pwd1
Broker-> CLient :tp1, tp2
principal Client2[
  knows private hn,usnC2,pwd2,psk1
]
Client2->Broker:usnC2,pwd2
Broker-> CLient2 :tp1, tp2
principal Publisher[
  m = AEAD_ENC(psk1,aa,tp2)
]
Publisher->Broker:m
Broker->Client: m
principal Client[
```

```

        m_de = AEAD_DEC(psk1,m,tp2)?
    ]
    Broker->Client2:m
    principal Client3[
        knows private hn,usnC3,pwd3
    ]
    Client3->Broker:usnC3,pwd3
    Broker->Client3:tp1, tp2,m
    principal Client2[
        m_de2 = AEAD_DEC(psk1,m,tp2)?
    ]
    queries[

        authentication? Publisher->Broker:cnt1
        confidentiality? cnt1
        equivalence? pwd,pwd_
        confidentiality? pwd
        authentication? Broker->Client:m
        confidentiality?m
        confidentiality?aa

    ]

```

The Result

Run In terminal # verifpal verify MQTT_Model.vp

Deduction • Output of AEAD_ENC(tp2, pwd3, pwd3) obtained by reconstructing with tp2, pwd3, pwd3. (Analysis 796)
 Deduction • Output of AEAD_ENC(tp2, pwd, usnc2) obtained by reconstructing with tp2, pwd, usnc2. (Analysis 809)
 Deduction • Output of AEAD_ENC(tp2, pwd2, pwd1) obtained by reconstructing with tp2, pwd2, pwd1. (Analysis 810)
 Deduction • Output of AEAD_ENC(usnc1, usnc1, pwd3) obtained by reconstructing with usnc1, usnc1, pwd3. (Analysis 810)
 Deduction • Output of AEAD_ENC(tp2, pwd, usnc3) obtained by reconstructing with tp2, pwd, usnc3. (Analysis 813)
 Deduction • Output of AEAD_ENC(usnc1, nil, nil) obtained by reconstructing with usnc1, nil, nil. (Analysis 816)
 Deduction • Output of AEAD_ENC(usnc1, usnc2, usnc2) obtained by reconstructing with usnc1, usnc2, usnc2. (Analysis 816)
 Deduction • Output of AEAD_ENC(tp2, pwd, pwd3) obtained by reconstructing with tp2, pwd, pwd3. (Analysis 816)
 Deduction • Output of AEAD_ENC(usnc1, usnc2, pwd3) obtained by reconstructing with usnc1, usnc2, pwd3. (Analysis 817)
 Deduction • Output of AEAD_ENC(usnc1, usnc1, usnp) obtained by reconstructing with usnc1, usnc1, usnp. (Analysis 818)
 Deduction • Output of AEAD_ENC(usnc1, tp1, usnc3) obtained by reconstructing with usnc1, tp1, usnc3. (Analysis 818)
 Deduction • Output of AEAD_ENC(usnc1, tp2, tp1) obtained by reconstructing with usnc1, tp2, tp1. (Analysis 822)
 Deduction • Output of AEAD_ENC(usnc1, usnc3, tp2) obtained by reconstructing with usnc1, usnc3, tp2. (Analysis 823)
 Deduction • Output of AEAD_ENC(usnc1, usnc1, usnc3) obtained by reconstructing with usnc1, usnc1, usnc3. (Analysis 823)
 Deduction • Output of AEAD_ENC(usnc1, tp2, tp2) obtained by reconstructing with usnc1, tp2, tp2. (Analysis 824)
 Deduction • Output of AEAD_ENC(usnc1, usnc3, pwd) obtained by reconstructing with usnc1, usnc3, pwd. (Analysis 824)
 Deduction • Output of AEAD_ENC(tp2, pwd, usnp) obtained by reconstructing with tp2, pwd, usnp. (Analysis 826)
 Deduction • Output of AEAD_ENC(usnc1, tp2, usnc3) obtained by reconstructing with usnc1, tp2, usnc3. (Analysis 826)
 Deduction • Output of AEAD_ENC(usnc1, tp1, nil) obtained by reconstructing with usnc1, tp1, nil. (Analysis 826)
 Deduction • Output of AEAD_ENC(usnc1, usnp, usnp) obtained by reconstructing with usnc1, usnp, usnp. (Analysis 827)
 Deduction • Output of AEAD_ENC(usnc1, usnc2, tp2) obtained by reconstructing with usnc1, usnc2, tp2. (Analysis 829)

Deduction • Output of AEAD_ENC(usnc1, usnc2, tp1) obtained by reconstructing with usnc1, usnc2, tp1. (Analysis 931)

Deduction • Output of AEAD_ENC(pwd1, tp1, pwd2) obtained by reconstructing with pwd1, tp1, pwd2. (Analysis 933)

Deduction • Output of AEAD_ENC(usnc1, usnc3, pwd2) obtained by reconstructing with usnc1, usnc3, pwd2. (Analysis 935)

Deduction • Output of AEAD_ENC(pwd1, nil, tp2) obtained by reconstructing with pwd1, nil, tp2. (Analysis 935)

Deduction • Output of AEAD_ENC(usnc1, pwd2, tp1) obtained by reconstructing with usnc1, pwd2, tp1. (Analysis 936)

Deduction • Output of AEAD_ENC(usnc1, pwd, pwd2) obtained by reconstructing with usnc1, pwd, pwd2. (Analysis 939)

Deduction • Output of AEAD_ENC(pwd1, nil, usnp) obtained by reconstructing with pwd1, nil, usnp. (Analysis 922)

Deduction • Output of AEAD_ENC(usnc1, tp2, pwd1) obtained by reconstructing with usnc1, tp2, pwd1. (Analysis 928)

Deduction • Output of AEAD_ENC(pwd1, tp1, usnc3) obtained by reconstructing with pwd1, tp1, usnc3. (Analysis 940)

Deduction • Output of AEAD_ENC(usnc1, pwd3, usnc2) obtained by reconstructing with usnc1, pwd3, usnc2. (Analysis 941)

Deduction • Output of AEAD_ENC(usnc1, usnc1, usnc2) obtained by reconstructing with usnc1, usnc1, usnc2. (Analysis 943)

Deduction • Output of AEAD_ENC(pwd1, tp1, pwd3) obtained by reconstructing with pwd1, tp1, pwd3. (Analysis 952)

Deduction • Output of AEAD_ENC(usnc1, usnp, pwd2) obtained by reconstructing with usnc1, usnp, pwd2. (Analysis 955)

Deduction • Output of AEAD_ENC(pwd1, nil, usnc2) obtained by reconstructing with pwd1, nil, usnc2. (Analysis 955)

Deduction • Output of AEAD_ENC(pwd1, tp1, nil) obtained by reconstructing with pwd1, tp1, nil. (Analysis 922)

Deduction • Output of AEAD_ENC(usnc1, pwd, nil) obtained by reconstructing with usnc1, pwd, nil. (Analysis 956)

Deduction • Output of AEAD_ENC(usnc1, usnc3, tp1) obtained by reconstructing with usnc1, usnc3, tp1. (Analysis 957)

Deduction • Output of AEAD_ENC(pwd1, nil, tp1) obtained by reconstructing with pwd1, nil, tp1. (Analysis 922)

Deduction • Output of AEAD_ENC(usnc1, usnp, usnc3) obtained by reconstructing with usnc1, usnp, usnc3. (Analysis 958)

Deduction • Output of AEAD_ENC(usnc1, usnc1, tp2) obtained by reconstructing with usnc1, usnc1, tp2. (Analysis 928)

Deduction • Output of AEAD_ENC(usnc1, tp2, usnp) obtained by reconstructing with usnc1, tp2, usnp. (Analysis 963)

Deduction • Output of AEAD_ENC(usnc1, pwd2, pwd) obtained by reconstructing with usnc1, pwd2, pwd. (Analysis 963)

Deduction • Output of AEAD_ENC(usnc1, tp2, usnc2) obtained by reconstructing with usnc1, tp2, usnc2. (Analysis 963)

Deduction • Output of AEAD_ENC(usnc1, usnc2, usnc3) obtained by reconstructing with usnc1, usnc2, usnc3. (Analysis 963)

Deduction • Output of AEAD_ENC(usnc1, usnc3, pwd3) obtained by reconstructing with usnc1, usnc3, pwd3. (Analysis 963)

Deduction • Output of AEAD_ENC(usnc1, nil, usnp) obtained by reconstructing with usnc1, nil, usnp. (Analysis 964)

Deduction • Output of AEAD_ENC(usnc1, usnc2, nil) obtained by reconstructing with usnc1, usnc2, nil. (Analysis 965)

Deduction • Output of AEAD_ENC(usnc1, pwd2, pwd3) obtained by reconstructing with usnc1, pwd2, pwd3. (Analysis 966)

Deduction • Output of AEAD_ENC(usnc1, tp1, tp1) obtained by reconstructing with usnc1, tp1, tp1. (Analysis 923)

Deduction • Output of AEAD_ENC(pwd1, nil, nil) obtained by reconstructing with pwd1, nil, nil. (Analysis 928)

Deduction • Output of AEAD_ENC(usnc1, usnc1, usnc1) obtained by reconstructing with usnc1, usnc1, usnc1. (Analysis 928)

Deduction • Output of AEAD_ENC(usnc1, usnp, nil) obtained by reconstructing with usnc1, usnp, nil. (Analysis 967)

Deduction • Output of AEAD_ENC(usnc1, usnc1, pwd1) obtained by reconstructing with usnc1, usnc1, pwd1. (Analysis 970)

Deduction • Output of AEAD_ENC(usnc1, usnp, tp2) obtained by reconstructing with usnc1, usnp, tp2. (Analysis 974)

Deduction • Output of AEAD_ENC(usnc1, pwd2, pwd2) obtained by reconstructing with usnc1, pwd2, pwd2. (Analysis 978)

Deduction • Output of AEAD_ENC(usnc1, usnc3, pwd1) obtained by reconstructing with usnc1, usnc3, pwd1. (Analysis 981)

Deduction • Output of AEAD_ENC(pwd1, nil, usnc1) obtained by reconstructing with pwd1, nil, usnc1. (Analysis 982)

Deduction • Output of AEAD_ENC(usnc1, pwd1, pwd2) obtained by reconstructing with usnc1, pwd1, pwd2. (Analysis 986)

Deduction • Output of AEAD_ENC(usnc1, pwd, pwd1) obtained by reconstructing with usnc1, pwd, pwd1. (Analysis 924)

Deduction • Output of AEAD_ENC(usnc1, pwd2, usnc1) obtained by reconstructing with usnc1, pwd2, usnc1. (Analysis 987)

Deduction • Output of AEAD_ENC(usnc1, pwd3, usnc3) obtained by reconstructing with usnc1, pwd3, usnc3. (Analysis 991)

Deduction • Output of AEAD_ENC(pwd1, tp2, pwd1) obtained by reconstructing with pwd1, tp2, pwd1. (Analysis 992)

Deduction • Output of AEAD_ENC(pwd1, usnc3, nil) obtained by reconstructing with pwd1, usnc3, nil. (Analysis 995)

Deduction • Output of AEAD_ENC(usnc1, pwd3, pwd3) obtained by reconstructing with usnc1, pwd3, pwd3. (Analysis 926)

Deduction • Output of AEAD_ENC(pwd1, usnc3, pwd) obtained by reconstructing with pwd1, usnc3, pwd. (Analysis 997)

Deduction • Output of AEAD_ENC(usnc1, usnp, pwd1) obtained by reconstructing with usnc1, usnp, pwd1. (Analysis 998)

Deduction • Output of AEAD_ENC(usnc1, pwd1, usnc2) obtained by reconstructing with usnc1, pwd1, usnc2. (Analysis 924)

Deduction • Output of AEAD_ENC(pwd1, tp2, pwd) obtained by reconstructing with pwd1, tp2, pwd. (Analysis 999)

Deduction • Output of AEAD_ENC(pwd1, pwd1, pwd2) obtained by reconstructing with pwd1, pwd1, pwd2. (Analysis 999)

Deduction • Output of AEAD_ENC(pwd1, pwd3, tp1) obtained by reconstructing with pwd1, pwd3, tp1. (Analysis 999)

Deduction • Output of AEAD_ENC(usnc1, nil, tp2) obtained by reconstructing with usnc1, nil, tp2. (Analysis 967)

Deduction • Output of AEAD_ENC(usnc1, pwd1, nil) obtained by reconstructing with usnc1, pwd1, nil. (Analysis 939)

Deduction • Output of AEAD_ENC(pwd1, nil, pwd2) obtained by reconstructing with pwd1, nil, pwd2. (Analysis 922)

Deduction • Output of AEAD_ENC(pwd1, tp1, usnc2) obtained by reconstructing with pwd1, tp1, usnc2. (Analysis 1000)

Deduction • Output of AEAD_ENC(pwd1, pwd3, tp2) obtained by reconstructing with pwd1, pwd3, tp2. (Analysis 1003)

Deduction • Output of AEAD_ENC(pwd1, usnp, pwd1) obtained by reconstructing with pwd1, usnp, pwd1. (Analysis 1006)

Deduction • Output of AEAD_ENC(pwd1, usnc1, usnc2) obtained by reconstructing with pwd1, usnc1, usnc2. (Analysis 1006)

Deduction • Output of AEAD_ENC(pwd1, usnc2, pwd) obtained by reconstructing with pwd1, usnc2, pwd. (Analysis 1008)

Deduction • Output of AEAD_ENC(pwd1, usnp, pwd2) obtained by reconstructing with pwd1, usnp, pwd2. (Analysis 1009)

Deduction • Output of AEAD_ENC(pwd1, usnp, usnc2) obtained by reconstructing with pwd1, usnp, usnc2. (Analysis 1011)

Deduction • Output of AEAD_ENC(usnc1, pwd, usnc3) obtained by reconstructing with usnc1, pwd, usnc3. (Analysis 1012)

Deduction • Output of AEAD_ENC(usnc1, pwd3, pwd) obtained by reconstructing with usnc1, pwd3, pwd. (Analysis 1013)

Deduction • Output of AEAD_ENC(usnc1, pwd1, usnc1) obtained by reconstructing with usnc1, pwd1, usnc1. (Analysis 1017)

Deduction • Output of AEAD_ENC(usnc1, nil, usnc3) obtained by reconstructing with usnc1, nil, usnc3. (Analysis 1018)

Deduction • Output of AEAD_ENC(usnc1, pwd, tp1) obtained by reconstructing with usnc1, pwd, tp1. (Analysis 1019)

Deduction • Output of AEAD_ENC(usnc1, nil, pwd3) obtained by reconstructing with usnc1, nil, pwd3. (Analysis 1020)

Deduction • Output of AEAD_ENC(usnc1, nil, pwd) obtained by reconstructing with usnc1, nil, pwd. (Analysis 1021)

Deduction • Output of AEAD_ENC(usnc1, usnp, usnc2) obtained by reconstructing with usnc1, usnp, usnc2. (Analysis 1024)

Deduction • Output of AEAD_ENC(pwd1, pwd1, pwd) obtained by reconstructing with pwd1, pwd1, pwd. (Analysis 1027)

Deduction • Output of AEAD_ENC(pwd1, usnc3, usnp) obtained by reconstructing with pwd1, usnc3, usnp. (Analysis 1034)

Deduction • Output of AEAD_ENC(pwd1, pwd3, nil) obtained by reconstructing with pwd1, pwd3, nil. (Analysis 1030)

Deduction • Output of AEAD_ENC(pwd1, pwd2, pwd1) obtained by reconstructing with pwd1, pwd2, pwd1. (Analysis 1030)

Deduction • Output of AEAD_ENC(pwd1, pwd2, usnp) obtained by reconstructing with pwd1, pwd2, usnp. (Analysis 1031)

Deduction • Output of AEAD_ENC(pwd1, usnp, usnp) obtained by reconstructing with pwd1, usnp, usnp. (Analysis 1035)

Deduction • Output of AEAD_ENC(usnc1, tp1, pwd1) obtained by reconstructing with usnc1, tp1, pwd1. (Analysis 1035)

Deduction • Output of AEAD_ENC(pwd1, pwd, nil) obtained by reconstructing with pwd1, pwd, nil. (Analysis 1027)

Deduction • Output of AEAD_ENC(usnc1, pwd1, usnp) obtained by reconstructing with usnc1, pwd1, usnp. (Analysis 1039)

Deduction • Output of AEAD_ENC(pwd1, pwd1, usnc2) obtained by reconstructing with pwd1, pwd1, usnc2. (Analysis 1033)

Deduction • Output of AEAD_ENC(pwd1, tp2, usnc1) obtained by reconstructing with pwd1, tp2, usnc1. (Analysis 1043)

Deduction • Output of AEAD_ENC(pwd1, tp1, usnp) obtained by reconstructing with pwd1, tp1, usnp. (Analysis 1043)

Deduction • Output of AEAD_ENC(pwd1, tp2, pwd3) obtained by reconstructing with pwd1, tp2, pwd3. (Analysis 1044)

Deduction • Output of AEAD_ENC(pwd1, pwd1, tp1) obtained by reconstructing with pwd1, pwd1, tp1. (Analysis 1049)

Deduction • Output of AEAD_ENC(usnc1, usnc1, tp1) obtained by reconstructing with usnc1, usnc1, tp1. (Analysis 1051)

Deduction • Output of AEAD_ENC(pwd1, tp2, tp2) obtained by reconstructing with pwd1, tp2, tp2. (Analysis 1054)

Deduction • Output of AEAD_ENC(pwd1, usnc2, nil) obtained by reconstructing with pwd1, usnc2, nil. (Analysis 1060)

Deduction • Output of AEAD_ENC(pwd1, tp2, usnc3) obtained by reconstructing with pwd1, tp2, usnc3. (Analysis 1068)

Deduction • Output of AEAD_ENC(pwd1, usnc1, usnc3) obtained by reconstructing with pwd1, usnc1, usnc3. (Analysis 1068)

Deduction • Output of AEAD_ENC(pwd1, nil, pwd3) obtained by reconstructing with pwd1, nil, pwd3. (Analysis 1070)

Deduction • Output of AEAD_ENC(pwd1, usnc2, usnc1) obtained by reconstructing with pwd1, usnc2, usnc1. (Analysis 1072)

Deduction • Output of AEAD_ENC(pwd1, tp1, usnc1) obtained by reconstructing with pwd1, tp1, usnc1. (Analysis 1073)

Deduction • Output of AEAD_ENC(pwd1, usnc1, tp2) obtained by reconstructing with pwd1, usnc1, tp2. (Analysis 1074)

Deduction • Output of AEAD_ENC(pwd1, pwd3, pwd1) obtained by reconstructing with pwd1, pwd3, pwd1. (Analysis 1081)

Deduction • Output of AEAD_ENC(usnc2, nil, pwd1) obtained by reconstructing with usnc2, nil, pwd1. (Analysis 1081)

Deduction • Output of AEAD_ENC(pwd1, usnc2, pwd2) obtained by reconstructing with pwd1, usnc2, pwd2. (Analysis 1084)

Deduction • Output of AEAD_ENC(pwd1, pwd3, usnc2) obtained by reconstructing with pwd1, pwd3, usnc2. (Analysis 1093)

Deduction • Output of AEAD_ENC(pwd1, tp2, nil) obtained by reconstructing with pwd1, tp2, nil. (Analysis 1095)

Deduction • Output of AEAD_ENC(usnc2, nil, tp1) obtained by reconstructing with usnc2, nil, tp1. (Analysis 1095)

Deduction • Output of AEAD_ENC(pwd1, pwd, usnc2) obtained by reconstructing with pwd1, pwd, usnc2. (Analysis 1096)

Deduction • Output of AEAD_ENC(pwd1, pwd, usnc3) obtained by reconstructing with pwd1, pwd, usnc3. (Analysis 1100)

Deduction • Output of AEAD_ENC(pwd1, usnp, pwd3) obtained by reconstructing with pwd1, usnp, pwd3. (Analysis 1100)

Deduction • Output of AEAD_ENC(usnc2, nil, usnc2) obtained by reconstructing with usnc2, nil, usnc2. (Analysis 1107)

Deduction • Output of AEAD_ENC(pwd1, tp1, tp1) obtained by reconstructing with pwd1, tp1, tp1. (Analysis 1108)

Deduction • Output of AEAD_ENC(pwd1, tp1, tp2) obtained by reconstructing with pwd1, tp1, tp2. (Analysis 1093)

Deduction • Output of AEAD_ENC(pwd1, usnc3, usnc1) obtained by reconstructing with pwd1, usnc3, usnc1. (Analysis 1112)

Deduction • Output of AEAD_ENC(usnc1, pwd3, pwd2) obtained by reconstructing with usnc1, pwd3, pwd2. (Analysis 1112)

Deduction • Output of AEAD_ENC(pwd1, pwd2, tp2) obtained by reconstructing with pwd1, pwd2, tp2. (Analysis 1113)

Deduction • Output of AEAD_ENC(pwd1, usnp, pwd) obtained by reconstructing with pwd1, usnp, pwd. (Analysis 1113)

Deduction • Output of AEAD_ENC(pwd1, pwd2, pwd) obtained by reconstructing with pwd1, pwd2, pwd. (Analysis 1117)

Deduction • Output of AEAD_ENC(usnc1, usnp, tp1) obtained by reconstructing with usnc1, usnp, tp1. (Analysis 1118)

Deduction • Output of AEAD_ENC(pwd1, nil, usnc3) obtained by reconstructing with pwd1, nil, usnc3. (Analysis 1121)

Deduction • Output of AEAD_ENC(pwd1, nil, pwd) obtained by reconstructing with pwd1, nil, pwd. (Analysis 1122)

Deduction • Output of AEAD_ENC(usnc2, nil, usnc1) obtained by reconstructing with usnc2, nil, usnc1. (Analysis 1131)

Deduction • Output of AEAD_ENC(pwd1, pwd, usnp) obtained by reconstructing with pwd1, pwd, usnp. (Analysis 1131)

Deduction • Output of AEAD_ENC(usnc2, nil, nil) obtained by reconstructing with usnc2, nil, nil. (Analysis 1132)

Deduction • Output of AEAD_ENC(usnc2, pwd2, usnp) obtained by reconstructing with usnc2, pwd2, usnp. (Analysis 1132)

Deduction • Output of AEAD_ENC(usnc2, pwd2, pwd3) obtained by reconstructing with usnc2, pwd2, pwd3. (Analysis 1132)

Deduction • Output of AEAD_ENC(usnc2, usnc3, tp2) obtained by reconstructing with usnc2, usnc3, tp2. (Analysis 1138)

Deduction • Output of AEAD_ENC(pwd1, usnc1, pwd3) obtained by reconstructing with pwd1, usnc1, pwd3. (Analysis 1138)

Deduction • Output of AEAD_ENC(pwd1, pwd1, pwd3) obtained by reconstructing with pwd1, pwd1, pwd3. (Analysis 1138)

Deduction • Output of AEAD_ENC(usnc2, pwd3, pwd) obtained by reconstructing with usnc2, pwd3, pwd. (Analysis 1140)

Deduction • Output of AEAD_ENC(usnc2, pwd3, usnc2) obtained by reconstructing with usnc2, pwd3, usnc2. (Analysis 1140)

Deduction • Output of AEAD_ENC(usnc2, usnp, tp1) obtained by reconstructing with usnc2, usnp, tp1. (Analysis 1140)

Deduction • Output of AEAD_ENC(usnc2, pwd3, usnc1) obtained by reconstructing with usnc2, pwd3, usnc1. (Analysis 1143)

Deduction • Output of AEAD_ENC(pwd1, pwd3, usnp) obtained by reconstructing with pwd1, pwd3, usnp. (Analysis 1143)

Deduction • Output of AEAD_ENC(usnc2, pwd3, usnc3) obtained by reconstructing with usnc2, pwd3, usnc3. (Analysis 1143)

Deduction • Output of AEAD_ENC(usnc1, pwd, usnp) obtained by reconstructing with usnc1, pwd, usnp. (Analysis 1143)

Deduction • Output of AEAD_ENC(pwd1, usnc2, usnc3) obtained by reconstructing with pwd1, usnc2, usnc3. (Analysis 1143)

Deduction • Output of AEAD_ENC(pwd1, usnp, tp2) obtained by reconstructing with pwd1, usnp, tp2. (Analysis 1146)

Deduction • Output of AEAD_ENC(pwd1, usnc2, usnc2) obtained by reconstructing with pwd1, usnc2, usnc2. (Analysis 1146)

Deduction • Output of AEAD_ENC(usnc2, pwd3, pwd3) obtained by reconstructing with usnc2, pwd3, pwd3. (Analysis 1146)

Deduction • Output of AEAD_ENC(usnc2, usnp, pwd) obtained by reconstructing with usnc2, usnp, pwd. (Analysis 1148)

Deduction • Output of AEAD_ENC(usnc2, pwd, nil) obtained by reconstructing with usnc2, pwd, nil. (Analysis 1149)

Deduction • Output of AEAD_ENC(usnc2, pwd, tp1) obtained by reconstructing with usnc2, pwd, tp1. (Analysis 1149)

Deduction • Output of AEAD_ENC(usnc2, pwd, usnc1) obtained by reconstructing with usnc2, pwd, usnc1. (Analysis 1153)

Deduction • Output of AEAD_ENC(pwd1, pwd, pwd) obtained by reconstructing with pwd1, pwd, pwd. (Analysis 1155)

Deduction • Output of AEAD_ENC(pwd1, usnc2, pwd1) obtained by reconstructing with pwd1, usnc2, pwd1. (Analysis 1155)

Deduction • Output of AEAD_ENC(usnc2, nil, pwd3) obtained by reconstructing with usnc2, nil, pwd3. (Analysis 1159)

Deduction • Output of AEAD_ENC(pwd1, pwd3, pwd2) obtained by reconstructing with pwd1, pwd3, pwd2. (Analysis 1166)

Deduction • Output of AEAD_ENC(pwd1, usnc2, tp2) obtained by reconstructing with pwd1, usnc2, tp2. (Analysis 1169)

Deduction • Output of AEAD_ENC(pwd1, nil, pwd1) obtained by reconstructing with pwd1, nil, pwd1. (Analysis 1171)

Deduction • Output of AEAD_ENC(pwd1, usnc1, tp1) obtained by reconstructing with pwd1, usnc1, tp1. (Analysis 1175)

Deduction • Output of AEAD_ENC(pwd1, pwd2, usnc3) obtained by reconstructing with pwd1, pwd2, usnc3. (Analysis 1180)

Deduction • Output of AEAD_ENC(pwd2, tp2, tp2) obtained by reconstructing with pwd2, tp2, tp2. (Analysis 1180)

Deduction • Output of AEAD_ENC(usnc2, usnc1, usnc3) obtained by reconstructing with usnc2, usnc1, usnc3. (Analysis 1180)

Deduction • Output of AEAD_ENC(usnc2, usnc1, pwd3) obtained by reconstructing with usnc2, usnc1, pwd3. (Analysis 1186)

Deduction • Output of AEAD_ENC(pwd2, tp2, usnc1) obtained by reconstructing with pwd2, tp2, usnc1. (Analysis 1187)

Deduction • Output of AEAD_ENC(usnc2, usnc1, nil) obtained by reconstructing with usnc2, usnc1, nil. (Analysis 1187)

Deduction • Output of AEAD_ENC(pwd1, pwd2, usnc1) obtained by reconstructing with pwd1, pwd2, usnc1. (Analysis 1190)

Deduction • Output of AEAD_ENC(pwd1, usnc1, nil) obtained by reconstructing with pwd1, usnc1, nil. (Analysis 1192)

Deduction • Output of AEAD_ENC(pwd2, tp2, pwd1) obtained by reconstructing with pwd2, tp2, pwd1. (Analysis 1194)

Deduction • Output of AEAD_ENC(pwd1, tp1, pwd1) obtained by reconstructing with pwd1, tp1, pwd1. (Analysis 1194)

Deduction • Output of AEAD_ENC(pwd1, pwd, pwd2) obtained by reconstructing with pwd1, pwd, pwd2. (Analysis 1194)

Deduction • Output of AEAD_ENC(usnc2, usnc1, pwd1) obtained by reconstructing with usnc2, usnc1, pwd1. (Analysis 1187)

Deduction • Output of AEAD_ENC(pwd1, usnc1, pwd) obtained by reconstructing with pwd1, usnc1, pwd. (Analysis 1200)

Deduction • Output of AEAD_ENC(pwd2, tp2, usnc2) obtained by reconstructing with pwd2, tp2, usnc2. (Analysis 1201)

Deduction • Output of AEAD_ENC(pwd1, tp2, usnc2) obtained by reconstructing with pwd1, tp2, usnc2. (Analysis 1202)

Deduction • Output of AEAD_ENC(pwd1, usnc1, pwd2) obtained by reconstructing with pwd1, usnc1, pwd2. (Analysis 1203)

Deduction • Output of AEAD_ENC(pwd1, usnc1, usnc1) obtained by reconstructing with pwd1, usnc1, usnc1. (Analysis 1208)

Deduction • Output of AEAD_ENC(pwd2, tp2, pwd2) obtained by reconstructing with pwd2, tp2, pwd2. (Analysis 1209)

Deduction • Output of AEAD_ENC(pwd1, usnc3, usnc3) obtained by reconstructing with pwd1, usnc3, usnc3. (Analysis 1211)

Deduction • Output of AEAD_ENC(pwd2, tp2, usnc3) obtained by reconstructing with pwd2, tp2, usnc3. (Analysis 1217)

Deduction • Output of AEAD_ENC(pwd1, usnp, tp1) obtained by reconstructing with pwd1, usnp, tp1. (Analysis 1227)

Deduction • Output of AEAD_ENC(pwd1, tp2, tp1) obtained by reconstructing with pwd1, tp2, tp1. (Analysis 1227)

Deduction • Output of AEAD_ENC(usnc2, usnc2, nil) obtained by reconstructing with usnc2, usnc2, nil. (Analysis 1228)

Deduction • Output of AEAD_ENC(pwd1, pwd1, usnc3) obtained by reconstructing with pwd1, pwd1, usnc3. (Analysis 1229)

Deduction • Output of AEAD_ENC(pwd1, usnc1, pwd1) obtained by reconstructing with pwd1, usnc1, pwd1. (Analysis 1229)

Deduction • Output of AEAD_ENC(pwd1, pwd1, usnc1) obtained by reconstructing with pwd1, pwd1, usnc1. (Analysis 1232)

Deduction • Output of AEAD_ENC(usnc2, usnc1, tp2) obtained by reconstructing with usnc2, usnc1, tp2. (Analysis 1233)

Deduction • Output of AEAD_ENC(pwd1, pwd1, nil) obtained by reconstructing with pwd1, pwd1, nil. (Analysis 1203)

Deduction • Output of AEAD_ENC(usnc2, tp2, pwd) obtained by reconstructing with usnc2, tp2, pwd. (Analysis 1234)

Deduction • Output of AEAD_ENC(usnc2, tp2, usnc3) obtained by reconstructing with usnc2, tp2, usnc3. (Analysis 1235)

Deduction • Output of AEAD_ENC(usnc2, usnp, usnc1) obtained by reconstructing with usnc2, usnp, usnc1. (Analysis 1236)

Deduction • Output of AEAD_ENC(usnc2, pwd1, pwd3) obtained by reconstructing with usnc2, pwd1, pwd3. (Analysis 1237)

Deduction • Output of AEAD_ENC(pwd1, pwd2, pwd3) obtained by reconstructing with pwd1, pwd2, pwd3. (Analysis 1239)

Deduction • Output of AEAD_ENC(pwd1, tp2, pwd2) obtained by reconstructing with pwd1, tp2, pwd2. (Analysis 1239)

Deduction • Output of AEAD_ENC(pwd2, usnc1, tp1) obtained by reconstructing with pwd2, usnc1, tp1. (Analysis 1240)

Deduction • Output of AEAD_ENC(pwd1, usnc3, pwd2) obtained by reconstructing with pwd1, usnc3, pwd2. (Analysis 1246)

Deduction • Output of AEAD_ENC(pwd1, tp1, pwd) obtained by reconstructing with pwd1, tp1, pwd. (Analysis 1254)

Deduction • Output of AEAD_ENC(pwd1, tp2, usnp) obtained by reconstructing with pwd1, tp2, usnp. (Analysis 1254)

Deduction • Output of AEAD_ENC(usnc2, tp1, pwd1) obtained by reconstructing with usnc2, tp1, pwd1. (Analysis 1256)

Deduction • Output of AEAD_ENC(pwd2, pwd1, usnc2) obtained by reconstructing with pwd2, pwd1, usnc2. (Analysis 1258)

Deduction • Output of AEAD_ENC(pwd2, usnc1, pwd3) obtained by reconstructing with pwd2, usnc1, pwd3. (Analysis 1258)

Deduction • Output of AEAD_ENC(pwd1, pwd2, pwd2) obtained by reconstructing with pwd1, pwd2, pwd2. (Analysis 1261)

Deduction • Output of AEAD_ENC(usnc2, usnc3, pwd) obtained by reconstructing with usnc2, usnc3, pwd. (Analysis 1263)

Deduction • Output of AEAD_ENC(pwd2, tp2, pwd) obtained by reconstructing with pwd2, tp2, pwd. (Analysis 1264)

Deduction • Output of AEAD_ENC(usnc2, usnc3, usnc1) obtained by reconstructing with usnc2, usnc3, usnc1. (Analysis 1266)

Deduction • Output of AEAD_ENC(usnc2, pwd2, pwd) obtained by reconstructing with usnc2, pwd2, pwd. (Analysis 1267)

Deduction • Output of AEAD_ENC(pwd1, pwd1, pwd1) obtained by reconstructing with pwd1, pwd1, pwd1. (Analysis 1266)

Deduction • Output of AEAD_ENC(pwd1, pwd2, usnc2) obtained by reconstructing with pwd1, pwd2, usnc2. (Analysis 1268)

Deduction • Output of AEAD_ENC(usnc2, usnc3, usnc2) obtained by reconstructing with usnc2, usnc3, usnc2. (Analysis 1269)

Deduction • Output of AEAD_ENC(usnc2, pwd2, usnc1) obtained by reconstructing with usnc2, pwd2, usnc1. (Analysis 1269)

Deduction • Output of AEAD_ENC(usnc2, usnc1, usnp) obtained by reconstructing with usnc2, usnc1, usnp. (Analysis 1270)

Deduction • Output of AEAD_ENC(usnc2, pwd2, usnc2) obtained by reconstructing with usnc2, pwd2, usnc2. (Analysis 1273)

Deduction • Output of AEAD_ENC(usnc2, pwd2, tp2) obtained by reconstructing with usnc2, pwd2, tp2. (Analysis 1275)

Deduction • Output of AEAD_ENC(usnc2, nil, pwd2) obtained by reconstructing with usnc2, nil, pwd2. (Analysis 1278)

Deduction • Output of AEAD_ENC(usnc2, usnc3, tp1) obtained by reconstructing with usnc2, usnc3, tp1. (Analysis 1279)

Deduction • Output of AEAD_ENC(usnc2, usnc3, pwd3) obtained by reconstructing with usnc2, usnc3, pwd3. (Analysis 1289)

Deduction • Output of AEAD_ENC(pwd2, usnc3, tp2) obtained by reconstructing with pwd2, usnc3, tp2. (Analysis 1292)

Deduction • Output of AEAD_ENC(pwd2, usnc3, pwd1) obtained by reconstructing with pwd2, usnc3, pwd1. (Analysis 1292)

Deduction • Output of AEAD_ENC(pwd2, usnc3, pwd2) obtained by reconstructing with pwd2, usnc3, pwd2. (Analysis 1292)

Deduction • Output of AEAD_ENC(pwd2, tp1, pwd3) obtained by reconstructing with pwd2, tp1, pwd3. (Analysis 1296)

Deduction • Output of AEAD_ENC(pwd2, usnc3, tp1) obtained by reconstructing with pwd2, usnc3, tp1. (Analysis 1297)

Deduction • Output of AEAD_ENC(pwd1, usnc2, usnp) obtained by reconstructing with pwd1, usnc2, usnp. (Analysis 1297)

Deduction • Output of AEAD_ENC(pwd2, pwd3, pwd1) obtained by reconstructing with pwd2, pwd3, pwd1. (Analysis 1297)

Deduction • Output of AEAD_ENC(pwd2, usnp, pwd) obtained by reconstructing with pwd2, usnp, pwd. (Analysis 1297)

Deduction • Output of AEAD_ENC(pwd2, usnp, usnc2) obtained by reconstructing with pwd2, usnp, usnc2. (Analysis 1297)

Deduction • Output of AEAD_ENC(pwd2, usnp, pwd2) obtained by reconstructing with pwd2, usnp, pwd2. (Analysis 1300)

Deduction • Output of AEAD_ENC(pwd2, usnp, usnc1) obtained by reconstructing with pwd2, usnp, usnc1. (Analysis 1303)

Deduction • Output of AEAD_ENC(pwd2, tp1, pwd2) obtained by reconstructing with pwd2, tp1, pwd2. (Analysis 1303)

Deduction • Output of AEAD_ENC(pwd2, pwd3, pwd3) obtained by reconstructing with pwd2, pwd3, pwd3. (Analysis 1304)

Deduction • Output of AEAD_ENC(pwd2, pwd, tp2) obtained by reconstructing with pwd2, pwd, tp2. (Analysis 1304)

Deduction • Output of AEAD_ENC(pwd2, pwd3, usnp) obtained by reconstructing with pwd2, pwd3, usnp. (Analysis 1304)

Deduction • Output of AEAD_ENC(pwd2, usnp, pwd3) obtained by reconstructing with pwd2, usnp, pwd3. (Analysis 1304)

Deduction • Output of AEAD_ENC(pwd2, usnc1, usnc1) obtained by reconstructing with pwd2, usnc1, usnc1. (Analysis 1304)

Deduction • Output of AEAD_ENC(usnc2, pwd2, pwd1) obtained by reconstructing with usnc2, pwd2, pwd1. (Analysis 1307)

Deduction • Output of AEAD_ENC(usnc2, pwd3, tp2) obtained by reconstructing with usnc2, pwd3, tp2. (Analysis 1312)

Deduction • Output of AEAD_ENC(usnc3, nil, nil) obtained by reconstructing with usnc3, nil, nil. (Analysis 1312)

Deduction • Output of AEAD_ENC(usnc2, usnp, pwd1) obtained by reconstructing with usnc2, usnp, pwd1. (Analysis 1312)

Deduction • Output of AEAD_ENC(pwd2, usnc1, pwd1) obtained by reconstructing with pwd2, usnc1, pwd1. (Analysis 1313)

Deduction • Output of AEAD_ENC(usnc3, nil, usnc3) obtained by reconstructing with usnc3, nil, usnc3. (Analysis 1315)

Deduction • Output of AEAD_ENC(pwd2, pwd1, usnp) obtained by reconstructing with pwd2, pwd1, usnp. (Analysis 1315)

Deduction • Output of AEAD_ENC(usnc3, usnc1, nil) obtained by reconstructing with usnc3, usnc1, nil. (Analysis 1316)

Deduction • Output of AEAD_ENC(usnc3, usnc1, pwd2) obtained by reconstructing with usnc3, usnc1, pwd2. (Analysis 1318)

Deduction • Output of AEAD_ENC(pwd2, usnc2, usnc1) obtained by reconstructing with pwd2, usnc2, usnc1. (Analysis 1318)

Deduction • Output of AEAD_ENC(usnc3, usnc1, pwd3) obtained by reconstructing with usnc3, usnc1, pwd3. (Analysis 1319)

Deduction • Output of AEAD_ENC(pwd2, usnc2, nil) obtained by reconstructing with pwd2, usnc2, nil. (Analysis 1321)

Deduction • Output of AEAD_ENC(usnc3, pwd1, pwd2) obtained by reconstructing with usnc3, pwd1, pwd2. (Analysis 1322)

Deduction • Output of AEAD_ENC(usnc3, pwd1, pwd) obtained by reconstructing with usnc3, pwd1, pwd. (Analysis 1322)

Deduction • Output of AEAD_ENC(usnc3, usnc2, pwd1) obtained by reconstructing with usnc3, usnc2, pwd1. (Analysis 1323)

Deduction • Output of AEAD_ENC(usnc3, usnc2, tp1) obtained by reconstructing with usnc3, usnc2, tp1. (Analysis 1323)

Deduction • Output of AEAD_ENC(pwd2, usnc2, pwd1) obtained by reconstructing with pwd2, usnc2, pwd1. (Analysis 1323)

Deduction • Output of AEAD_ENC(pwd2, pwd2, usnc1) obtained by reconstructing with pwd2, pwd2, usnc1. (Analysis 1324)

Deduction • Output of AEAD_ENC(pwd2, pwd2, usnp) obtained by reconstructing with pwd2, pwd2, usnp. (Analysis 1324)

Deduction • Output of AEAD_ENC(usnc3, usnc2, pwd) obtained by reconstructing with usnc3, usnc2, pwd. (Analysis 1325)

Deduction • Output of AEAD_ENC(usnc3, pwd1, pwd3) obtained by reconstructing with usnc3, pwd1, pwd3. (Analysis 1325)

Deduction • Output of AEAD_ENC(pwd2, usnc3, usnc2) obtained by reconstructing with pwd2, usnc3, usnc2. (Analysis 1329)

Deduction • Output of AEAD_ENC(usnc3, pwd2, nil) obtained by reconstructing with usnc3, pwd2, nil. (Analysis 1330)

Deduction • Output of AEAD_ENC(pwd2, usnc2, tp2) obtained by reconstructing with pwd2, usnc2, tp2. (Analysis 1330)

Deduction • Output of AEAD_ENC(usnc3, pwd2, tp2) obtained by reconstructing with usnc3, pwd2, tp2. (Analysis 1330)

Deduction • Output of AEAD_ENC(pwd2, pwd, usnc1) obtained by reconstructing with pwd2, pwd, usnc1. (Analysis 1331)

Deduction • Output of AEAD_ENC(usnc2, pwd3, pwd1) obtained by reconstructing with usnc2, pwd3, pwd1. (Analysis 1332)

Deduction • Output of AEAD_ENC(usnc3, usnc2, nil) obtained by reconstructing with usnc3, usnc2, nil. (Analysis 1332)

Deduction • Output of AEAD_ENC(usnc3, usnc1, pwd) obtained by reconstructing with usnc3, usnc1, pwd. (Analysis 1332)

Deduction • Output of AEAD_ENC(usnc3, pwd2, pwd1) obtained by reconstructing with usnc3, pwd2, pwd1. (Analysis 1332)

Deduction • Output of AEAD_ENC(pwd2, pwd3, nil) obtained by reconstructing with pwd2, pwd3, nil. (Analysis 1334)

Deduction • Output of AEAD_ENC(pwd2, pwd3, tp1) obtained by reconstructing with pwd2, pwd3, tp1. (Analysis 1334)

Deduction • Output of AEAD_ENC(usnc3, pwd2, usnp) obtained by reconstructing with usnc3, pwd2, usnp. (Analysis 1334)

Deduction • Output of AEAD_ENC(pwd1, usnp, nil) obtained by reconstructing with pwd1, usnp, nil. (Analysis 1334)

Deduction • Output of AEAD_ENC(usnc2, pwd3, pwd2) obtained by reconstructing with usnc2, pwd3, pwd2. (Analysis 1334)

Deduction • Output of AEAD_ENC(pwd1, usnp, usnc1) obtained by reconstructing with pwd1, usnp, usnc1. (Analysis 1334)

Deduction • Output of AEAD_ENC(pwd2, pwd3, pwd2) obtained by reconstructing with pwd2, pwd3, pwd2. (Analysis 1336)

Deduction • Output of AEAD_ENC(usnc3, usnc3, usnc2) obtained by reconstructing with usnc3, usnc3, usnc2. (Analysis 1340)

Deduction • Output of AEAD_ENC(pwd2, nil, usnc3) obtained by reconstructing with pwd2, nil, usnc3. (Analysis 1344)

Deduction • Output of AEAD_ENC(usnc2, usnp, usnc2) obtained by reconstructing with usnc2, usnp, usnc2. (Analysis 1346)

Deduction • Output of AEAD_ENC(pwd1, pwd3, usnc3) obtained by reconstructing with pwd1, pwd3, usnc3. (Analysis 1353)

Deduction • Output of AEAD_ENC(pwd2, pwd, tp1) obtained by reconstructing with pwd2, pwd, tp1. (Analysis 1353)

Deduction • Output of AEAD_ENC(pwd2, pwd3, usnc2) obtained by reconstructing with pwd2, pwd3, usnc2. (Analysis 1346)

Deduction • Output of AEAD_ENC(usnc3, usnc3, usnc1) obtained by reconstructing with usnc3, usnc3, usnc1. (Analysis 1355)

Deduction • Output of AEAD_ENC(pwd2, nil, pwd3) obtained by reconstructing with pwd2, nil, pwd3. (Analysis 1355)

Deduction • Output of AEAD_ENC(pwd2, nil, usnp) obtained by reconstructing with pwd2, nil, usnp. (Analysis 1356)

Deduction • Output of AEAD_ENC(pwd1, pwd, tp2) obtained by reconstructing with pwd1, pwd, tp2. (Analysis 1357)

Deduction • Output of AEAD_ENC(usnc2, tp1, pwd3) obtained by reconstructing with usnc2, tp1, pwd3. (Analysis 1357)

Deduction • Output of AEAD_ENC(usnc2, pwd, usnp) obtained by reconstructing with usnc2, pwd, usnp. (Analysis 1359)

Deduction • Output of AEAD_ENC(usnc2, pwd3, usnp) obtained by reconstructing with usnc2, pwd3, usnp. (Analysis 1360)

Deduction • Output of AEAD_ENC(usnc2, nil, usnc3) obtained by reconstructing with usnc2, nil, usnc3. (Analysis 1361)

Deduction • Output of AEAD_ENC(pwd2, pwd, nil) obtained by reconstructing with pwd2, pwd, nil. (Analysis 1363)

Deduction • Output of AEAD_ENC(usnc1, pwd, pwd) obtained by reconstructing with usnc1, pwd, pwd. (Analysis 1364)

Deduction • Output of AEAD_ENC(usnc2, tp1, usnc3) obtained by reconstructing with usnc2, tp1, usnc3. (Analysis 1364)

Deduction • Output of AEAD_ENC(pwd2, nil, pwd2) obtained by reconstructing with pwd2, nil, pwd2. (Analysis 1366)

Deduction • Output of AEAD_ENC(usnc2, usnp, nil) obtained by reconstructing with usnc2, usnp, nil. (Analysis 1366)

Deduction • Output of AEAD_ENC(usnc2, tp2, nil) obtained by reconstructing with usnc2, tp2, nil. (Analysis 1371)

Deduction • Output of AEAD_ENC(usnc2, tp2, usnc1) obtained by reconstructing with usnc2, tp2, usnc1. (Analysis 1374)

Deduction • Output of AEAD_ENC(usnc2, tp2, pwd1) obtained by reconstructing with usnc2, tp2, pwd1. (Analysis 1375)

Deduction • Output of AEAD_ENC(pwd1, pwd3, pwd) obtained by reconstructing with pwd1, pwd3, pwd. (Analysis 1376)

Deduction • Output of AEAD_ENC(usnc2, usnp, usnp) obtained by reconstructing with usnc2, usnp, usnp. (Analysis 1379)

Deduction • Output of AEAD_ENC(usnc3, nil, usnp) obtained by reconstructing with usnc3, nil, usnp. (Analysis 1382)

Deduction • Output of AEAD_ENC(pwd1, usnp, usnc3) obtained by reconstructing with pwd1, usnp, usnc3. (Analysis 1386)

Deduction • Output of AEAD_ENC(usnc3, usnp, pwd1) obtained by reconstructing with usnc3, usnp, pwd1. (Analysis 1386)

Deduction • Output of AEAD_ENC(usnc3, tp1, usnc3) obtained by reconstructing with usnc3, tp1, usnc3. (Analysis 1387)

Deduction • Output of AEAD_ENC(usnc2, tp2, tp2) obtained by reconstructing with usnc2, tp2, tp2. (Analysis 1387)

Deduction • Output of AEAD_ENC(usnc2, tp2, tp1) obtained by reconstructing with usnc2, tp2, tp1. (Analysis 1387)

Deduction • Output of AEAD_ENC(pwd2, nil, pwd1) obtained by reconstructing with pwd2, nil, pwd1. (Analysis 1390)

Deduction • Output of AEAD_ENC(pwd2, nil, nil) obtained by reconstructing with pwd2, nil, nil. (Analysis 1390)

Deduction • Output of AEAD_ENC(usnc3, tp1, pwd2) obtained by reconstructing with usnc3, tp1, pwd2. (Analysis 1390)

Deduction • Output of AEAD_ENC(usnc2, pwd, usnc2) obtained by reconstructing with usnc2, pwd, usnc2. (Analysis 1392)

Deduction • Output of AEAD_ENC(usnc3, pwd, pwd1) obtained by reconstructing with usnc3, pwd, pwd1. (Analysis 1392)

Deduction • Output of AEAD_ENC(usnc3, usnp, usnc2) obtained by reconstructing with usnc3, usnp, usnc2. (Analysis 1394)

Deduction • Output of AEAD_ENC(pwd2, nil, usnc2) obtained by reconstructing with pwd2, nil, usnc2. (Analysis 1395)

Deduction • Output of AEAD_ENC(usnc2, tp2, pwd2) obtained by reconstructing with usnc2, tp2, pwd2. (Analysis 1398)

Deduction • Output of AEAD_ENC(usnc3, tp2, usnc1) obtained by reconstructing with usnc3, tp2, usnc1. (Analysis 1399)

Deduction • Output of AEAD_ENC(usnc3, pwd, usnp) obtained by reconstructing with usnc3, pwd, usnp. (Analysis 1399)

Deduction • Output of AEAD_ENC(usnc3, tp2, pwd) obtained by reconstructing with usnc3, tp2, pwd. (Analysis 1401)

Deduction • Output of AEAD_ENC(usnc3, tp2, usnc3) obtained by reconstructing with usnc3, tp2, usnc3. (Analysis 1403)

Deduction • Output of AEAD_ENC(usnc2, tp1, pwd2) obtained by reconstructing with usnc2, tp1, pwd2. (Analysis 1403)

Deduction • Output of AEAD_ENC(usnc3, usnc1, usnc2) obtained by reconstructing with usnc3, usnc1, usnc2. (Analysis 1403)

Deduction • Output of AEAD_ENC(usnc3, pwd, pwd2) obtained by reconstructing with usnc3, pwd, pwd2. (Analysis 1406)

Deduction • Output of AEAD_ENC(pwd2, tp1, usnc1) obtained by reconstructing with pwd2, tp1, usnc1. (Analysis 1408)

Deduction • Output of AEAD_ENC(usnc3, pwd1, nil) obtained by reconstructing with usnc3, pwd1, nil. (Analysis 1408)

Deduction • Output of AEAD_ENC(usnc2, tp2, usnc2) obtained by reconstructing with usnc2, tp2, usnc2. (Analysis 1408)

Deduction • Output of AEAD_ENC(usnc3, pwd1, usnp) obtained by reconstructing with usnc3, pwd1, usnp. (Analysis 1411)

Deduction • Output of AEAD_ENC(pwd3, tp1, nil) obtained by reconstructing with pwd3, tp1, nil. (Analysis 1412)

Deduction • Output of AEAD_ENC(pwd3, tp2, usnc3) obtained by reconstructing with pwd3, tp2, usnc3. (Analysis 1413)

Deduction • Output of AEAD_ENC(pwd3, tp2, usnp) obtained by reconstructing with pwd3, tp2, usnp. (Analysis 1414)

Deduction • Output of AEAD_ENC(pwd2, tp1, usnc3) obtained by reconstructing with pwd2, tp1, usnc3. (Analysis 1414)

Deduction • Output of AEAD_ENC(pwd1, usnc2, tp1) obtained by reconstructing with pwd1, usnc2, tp1. (Analysis 1415)

Deduction • Output of AEAD_ENC(pwd2, tp1, pwd) obtained by reconstructing with pwd2, tp1, pwd. (Analysis 1419)

Deduction • Output of AEAD_ENC(pwd2, tp1, usnp) obtained by reconstructing with pwd2, tp1, usnp. (Analysis 1419)

Deduction • Output of AEAD_ENC(usnc3, usnc3, tp2) obtained by reconstructing with usnc3, usnc3, tp2. (Analysis 1423)

Deduction • Output of AEAD_ENC(usnc3, pwd2, usnc1) obtained by reconstructing with usnc3, pwd2, usnc1. (Analysis 1425)

Deduction • Output of AEAD_ENC(usnc3, pwd2, pwd2) obtained by reconstructing with usnc3, pwd2, pwd2. (Analysis 1426)

Deduction • Output of AEAD_ENC(usnc3, usnc3, usnc3) obtained by reconstructing with usnc3, usnc3, usnc3. (Analysis 1426)

Deduction • Output of AEAD_ENC(pwd1, pwd2, tp1) obtained by reconstructing with pwd1, pwd2, tp1. (Analysis 1428)

Deduction • Output of AEAD_ENC(usnc3, pwd2, usnc2) obtained by reconstructing with usnc3, pwd2, usnc2. (Analysis 1429)

Deduction • Output of AEAD_ENC(pwd3, usnc1, usnc2) obtained by reconstructing with pwd3, usnc1, usnc2. (Analysis 1429)

Deduction • Output of AEAD_ENC(pwd2, tp2, nil) obtained by reconstructing with pwd2, tp2, nil. (Analysis 1429)

Deduction • Output of AEAD_ENC(usnc3, pwd3, tp1) obtained by reconstructing with usnc3, pwd3, tp1. (Analysis 1429)

Deduction • Output of AEAD_ENC(pwd3, usnc1, pwd2) obtained by reconstructing with pwd3, usnc1, pwd2. (Analysis 1429)

Deduction • Output of AEAD_ENC(usnc2, tp2, usnp) obtained by reconstructing with usnc2, tp2, usnp. (Analysis 1436)

Deduction • Output of AEAD_ENC(usnc2, pwd1, pwd2) obtained by reconstructing with usnc2, pwd1, pwd2. (Analysis 1438)

Deduction • Output of AEAD_ENC(usnc2, pwd1, tp1) obtained by reconstructing with usnc2, pwd1, tp1. (Analysis 1439)

Deduction • Output of AEAD_ENC(usnc2, pwd1, usnc3) obtained by reconstructing with usnc2, pwd1, usnc3. (Analysis 1442)

Deduction • Output of AEAD_ENC(pwd3, usnc1, pwd1) obtained by reconstructing with pwd3, usnc1, pwd1. (Analysis 1443)

Deduction • Output of AEAD_ENC(usnc2, pwd1, usnc2) obtained by reconstructing with usnc2, pwd1, usnc2. (Analysis 1443)

Deduction • Output of AEAD_ENC(pwd2, pwd1, nil) obtained by reconstructing with pwd2, pwd1, nil. (Analysis 1454)

Deduction • Output of AEAD_ENC(usnc3, pwd3, nil) obtained by reconstructing with usnc3, pwd3, nil. (Analysis 1457)

Deduction • Output of AEAD_ENC(usnc2, pwd2, nil) obtained by reconstructing with usnc2, pwd2, nil. (Analysis 1462)

Deduction • Output of AEAD_ENC(pwd3, tp1, usnc3) obtained by reconstructing with pwd3, tp1, usnc3. (Analysis 1462)

Deduction • Output of AEAD_ENC(usnc2, pwd2, tp1) obtained by reconstructing with usnc2, pwd2, tp1. (Analysis 1477)

Deduction • Output of AEAD_ENC(pwd3, tp1, pwd3) obtained by reconstructing with pwd3, tp1, pwd3. (Analysis 1479)

Deduction • Output of AEAD_ENC(pwd2, tp1, pwd1) obtained by reconstructing with pwd2, tp1, pwd1. (Analysis 1484)

Deduction • Output of AEAD_ENC(usnc2, pwd1, pwd1) obtained by reconstructing with usnc2, pwd1, pwd1. (Analysis 1486)

Deduction • Output of AEAD_ENC(usnc2, usnc1, usnc2) obtained by reconstructing with usnc2, usnc1, usnc2. (Analysis 1486)

Deduction • Output of AEAD_ENC(usnc2, usnc3, pwd2) obtained by reconstructing with usnc2, usnc3, pwd2. (Analysis 1486)

Deduction • Output of AEAD_ENC(usnc2, usnc1, usnc1) obtained by reconstructing with usnc2, usnc1, usnc1. (Analysis 1488)

Deduction • Output of AEAD_ENC(usnc2, usnc3, usnp) obtained by reconstructing with usnc2, usnc3, usnp. (Analysis 1490)

Deduction • Output of AEAD_ENC(usnc2, usnc1, pwd2) obtained by reconstructing with usnc2, usnc1, pwd2. (Analysis 1491)

Deduction • Output of AEAD_ENC(pwd1, pwd, pwd1) obtained by reconstructing with pwd1, pwd, pwd1. (Analysis 1492)

Deduction • Output of AEAD_ENC(pwd3, tp1, usnp) obtained by reconstructing with pwd3, tp1, usnp. (Analysis 1492)

Deduction • Output of AEAD_ENC(pwd1, usnc3, pwd3) obtained by reconstructing with pwd1, usnc3, pwd3. (Analysis 1494)

Deduction • Output of AEAD_ENC(usnc2, tp2, pwd3) obtained by reconstructing with usnc2, tp2, pwd3. (Analysis 1499)

Deduction • Output of AEAD_ENC(pwd1, usnc3, usnc2) obtained by reconstructing with pwd1, usnc3, usnc2. (Analysis 1502)

Deduction • Output of AEAD_ENC(pwd1, pwd1, tp2) obtained by reconstructing with pwd1, pwd1, tp2. (Analysis 1505)

Deduction • Output of AEAD_ENC(pwd2, pwd1, usnc1) obtained by reconstructing with pwd2, pwd1, usnc1. (Analysis 1507)

Deduction • Output of AEAD_ENC(pwd1, usnc3, tp2) obtained by reconstructing with pwd1, usnc3, tp2. (Analysis 1508)

Deduction • Output of AEAD_ENC(pwd1, usnc3, pwd1) obtained by reconstructing with pwd1, usnc3, pwd1. (Analysis 1514)

Deduction • Output of AEAD_ENC(pwd1, usnc2, pwd3) obtained by reconstructing with pwd1, usnc2, pwd3. (Analysis 1514)

Deduction • Output of AEAD_ENC(pwd1, usnc3, tp1) obtained by reconstructing with pwd1, usnc3, tp1. (Analysis 1514)

Deduction • Output of AEAD_ENC(usnc2, tp1, usnp) obtained by reconstructing with usnc2, tp1, usnp. (Analysis 1515)

Deduction • Output of AEAD_ENC(usnc2, pwd2, usnc3) obtained by reconstructing with usnc2, pwd2, usnc3. (Analysis 1515)

Deduction • Output of AEAD_ENC(usnc2, tp1, tp1) obtained by reconstructing with usnc2, tp1, tp1. (Analysis 1515)

Deduction • Output of AEAD_ENC(pwd2, nil, tp2) obtained by reconstructing with pwd2, nil, tp2. (Analysis 1515)

Deduction • Output of AEAD_ENC(pwd1, pwd3, usnc1) obtained by reconstructing with pwd1, pwd3, usnc1. (Analysis 1515)

Deduction • Output of AEAD_ENC(usnc2, pwd2, pwd2) obtained by reconstructing with usnc2, pwd2, pwd2. (Analysis 1515)

Deduction • Output of AEAD_ENC(pwd1, pwd1, usnp) obtained by reconstructing with pwd1, pwd1, usnp. (Analysis 1517)

Deduction • Output of AEAD_ENC(usnc2, usnc2, pwd1) obtained by reconstructing with usnc2, usnc2, pwd1. (Analysis 1518)

Deduction • Output of AEAD_ENC(pwd2, tp1, nil) obtained by reconstructing with pwd2, tp1, nil. (Analysis 1519)

Deduction • Output of AEAD_ENC(usnc2, pwd1, pwd) obtained by reconstructing with usnc2, pwd1, pwd. (Analysis 1517)

Deduction • Output of AEAD_ENC(usnc2, usnc2, pwd2) obtained by reconstructing with usnc2, usnc2, pwd2. (Analysis 1521)

Deduction • Output of AEAD_ENC(pwd2, usnc1, nil) obtained by reconstructing with pwd2, usnc1, nil. (Analysis 1521)

Deduction • Output of AEAD_ENC(pwd2, nil, usnc1) obtained by reconstructing with pwd2, nil, usnc1. (Analysis 1507)

Deduction • Output of AEAD_ENC(pwd1, usnc1, usnp) obtained by reconstructing with pwd1, usnc1, usnp. (Analysis 1515)

Deduction • Output of AEAD_ENC(usnc2, usnc1, pwd) obtained by reconstructing with usnc2, usnc1, pwd. (Analysis 1522)

Deduction • Output of AEAD_ENC(pwd3, pwd2, pwd1) obtained by reconstructing with pwd3, pwd2, pwd1. (Analysis 1523)

Deduction • Output of AEAD_ENC(usnc2, pwd, pwd3) obtained by reconstructing with usnc2, pwd, pwd3. (Analysis 1527)

Deduction • Output of AEAD_ENC(usnc2, usnc2, pwd) obtained by reconstructing with usnc2, usnc2, pwd. (Analysis 1527)

Deduction • Output of AEAD_ENC(pwd3, pwd2, usnc1) obtained by reconstructing with pwd3, pwd2, usnc1. (Analysis 1531)

Deduction • Output of AEAD_ENC(pwd3, usnc1, pwd3) obtained by reconstructing with pwd3, usnc1, pwd3. (Analysis 1531)

Deduction • Output of AEAD_ENC(usnc2, pwd1, usnc1) obtained by reconstructing with usnc2, pwd1, usnc1. (Analysis 1531)

Deduction • Output of AEAD_ENC(pwd3, usnc1, usnc3) obtained by reconstructing with pwd3, usnc1, usnc3. (Analysis 1532)

Deduction • Output of AEAD_ENC(pwd3, pwd2, pwd3) obtained by reconstructing with pwd3, pwd2, pwd3. (Analysis 1535)

Deduction • Output of AEAD_ENC(pwd3, pwd3, tp1) obtained by reconstructing with pwd3, pwd3, tp1. (Analysis 1536)

Deduction • Output of AEAD_ENC(usnc2, usnc3, nil) obtained by reconstructing with usnc2, usnc3, nil. (Analysis 1538)

Deduction • Output of AEAD_ENC(pwd3, pwd3, pwd1) obtained by reconstructing with pwd3, pwd3, pwd1. (Analysis 1538)

Deduction • Output of AEAD_ENC(pwd3, pwd3, pwd3) obtained by reconstructing with pwd3, pwd3, pwd3. (Analysis 1539)

Deduction • Output of AEAD_ENC(usnc2, usnc2, tp2) obtained by reconstructing with usnc2, usnc2, tp2. (Analysis 1539)

Deduction • Output of AEAD_ENC(pwd3, pwd3, pwd2) obtained by reconstructing with pwd3, pwd3, pwd2. (Analysis 1540)

Deduction • Output of AEAD_ENC(pwd3, usnc2, usnp) obtained by reconstructing with pwd3, usnc2, usnp. (Analysis 1541)

Deduction • Output of AEAD_ENC(pwd3, pwd, pwd2) obtained by reconstructing with pwd3, pwd, pwd2. (Analysis 1551)

Deduction • Output of AEAD_ENC(usnp, nil, usnc1) obtained by reconstructing with usnp, nil, usnc1. (Analysis 1554)

Deduction • Output of AEAD_ENC(usnp, nil, usnc2) obtained by reconstructing with usnp, nil, usnc2. (Analysis 1554)

Deduction • Output of AEAD_ENC(usnp, tp1, usnc2) obtained by reconstructing with usnp, tp1, usnc2. (Analysis 1556)

Deduction • Output of AEAD_ENC(pwd3, pwd, usnp) obtained by reconstructing with pwd3, pwd, usnp. (Analysis 1556)

Deduction • Output of AEAD_ENC(usnp, nil, tp2) obtained by reconstructing with usnp, nil, tp2. (Analysis 1556)

Deduction • Output of AEAD_ENC(pwd2, pwd2, usnc3) obtained by reconstructing with pwd2, pwd2, usnc3. (Analysis 1556)

Deduction • Output of AEAD_ENC(pwd2, usnc3, nil) obtained by reconstructing with pwd2, usnc3, nil. (Analysis 1556)

Deduction • Output of AEAD_ENC(pwd3, pwd2, tp2) obtained by reconstructing with pwd3, pwd2, tp2. (Analysis 1556)

Deduction • Output of AEAD_ENC(pwd2, pwd2, pwd2) obtained by reconstructing with pwd2, pwd2, pwd2. (Analysis 1558)

Deduction • Output of AEAD_ENC(pwd3, pwd, usnc3) obtained by reconstructing with pwd3, pwd, usnc3. (Analysis 1558)

Deduction • Output of AEAD_ENC(pwd2, pwd3, usnc3) obtained by reconstructing with pwd2, pwd3, usnc3. (Analysis 1558)

Deduction • Output of AEAD_ENC(pwd3, pwd2, usnp) obtained by reconstructing with pwd3, pwd2, usnp. (Analysis 1558)

Deduction • Output of AEAD_ENC(pwd3, usnc3, tp1) obtained by reconstructing with pwd3, usnc3, tp1. (Analysis 1559)

Deduction • Output of AEAD_ENC(usnp, tp2, tp1) obtained by reconstructing with usnp, tp2, tp1. (Analysis 1568)

Deduction • Output of AEAD_ENC(pwd, tp1, usnp) obtained by reconstructing with pwd, tp1, usnp. (Analysis 1610)

Deduction • Output of AEAD_ENC(usnp, pwd1, pwd3) obtained by reconstructing with usnp, pwd1, pwd3. (Analysis 1611)

Deduction • Output of AEAD_ENC(usnp, usnc2, tp1) obtained by reconstructing with usnp, usnc2, tp1. (Analysis 1611)

Deduction • Output of AEAD_ENC(usnp, pwd2, tp1) obtained by reconstructing with usnp, pwd2, tp1. (Analysis 1612)

Deduction • Output of AEAD_ENC(usnp, usnc2, pwd) obtained by reconstructing with usnp, usnc2, pwd. (Analysis 1612)

Deduction • Output of AEAD_ENC(pwd, usnc1, nil) obtained by reconstructing with pwd, usnc1, nil. (Analysis 1613)

Deduction • Output of AEAD_ENC(usnp, pwd2, tp2) obtained by reconstructing with usnp, pwd2, tp2. (Analysis 1616)

Deduction • Output of AEAD_ENC(pwd, pwd1, usnc2) obtained by reconstructing with pwd, pwd1, usnc2. (Analysis 1616)

Deduction • Output of AEAD_ENC(pwd, pwd1, pwd1) obtained by reconstructing with pwd, pwd1, pwd1. (Analysis 1618)

Deduction • Output of AEAD_ENC(usnp, pwd2, pwd) obtained by reconstructing with usnp, pwd2, pwd. (Analysis 1620)

Deduction • Output of AEAD_ENC(pwd, pwd1, nil) obtained by reconstructing with pwd, pwd1, nil. (Analysis 1620)

Deduction • Output of AEAD_ENC(usnc3, nil, tp2) obtained by reconstructing with usnc3, nil, tp2. (Analysis 1621)

Deduction • Output of AEAD_ENC(pwd, pwd1, pwd) obtained by reconstructing with pwd, pwd1, pwd. (Analysis 1621)

Deduction • Output of AEAD_ENC(pwd, pwd1, tp1) obtained by reconstructing with pwd, pwd1, tp1. (Analysis 1630)

Deduction • Output of AEAD_ENC(usnc3, tp1, tp1) obtained by reconstructing with usnc3, tp1, tp1. (Analysis 1630)

Deduction • Output of AEAD_ENC(pwd1, pwd, usnc1) obtained by reconstructing with pwd1, pwd, usnc1. (Analysis 1630)

Deduction • Output of AEAD_ENC(usnc3, nil, pwd1) obtained by reconstructing with usnc3, nil, pwd1. (Analysis 1633)

Deduction • Output of AEAD_ENC(pwd2, pwd1, usnc3) obtained by reconstructing with pwd2, pwd1, usnc3. (Analysis 1633)

Deduction • Output of AEAD_ENC(usnp, usnp, pwd) obtained by reconstructing with usnp, usnp, pwd. (Analysis 1633)

Deduction • Output of AEAD_ENC(usnp, usnc3, pwd1) obtained by reconstructing with usnp, usnc3, pwd1. (Analysis 1633)

Deduction • Output of AEAD_ENC(pwd2, pwd2, pwd3) obtained by reconstructing with pwd2, pwd2, pwd3. (Analysis 1635)

Deduction • Output of AEAD_ENC(usnc3, usnc3, pwd1) obtained by reconstructing with usnc3, usnc3, pwd1. (Analysis 1635)

Deduction • Output of AEAD_ENC(pwd2, pwd1, pwd) obtained by reconstructing with pwd2, pwd1, pwd. (Analysis 1637)

Deduction • Output of AEAD_ENC(pwd3, tp1, tp1) obtained by reconstructing with pwd3, tp1, tp1. (Analysis 1637)

Deduction • Output of AEAD_ENC(pwd1, pwd2, nil) obtained by reconstructing with pwd1, pwd2, nil. (Analysis 1640)

Deduction • Output of AEAD_ENC(usnp, pwd3, usnc3) obtained by reconstructing with usnp, pwd3, usnc3. (Analysis 1640)

Deduction • Output of AEAD_ENC(pwd3, tp2, pwd) obtained by reconstructing with pwd3, tp2, pwd. (Analysis 1642)

Deduction • Output of AEAD_ENC(pwd1, pwd3, pwd3) obtained by reconstructing with pwd1, pwd3, pwd3. (Analysis 1642)

Deduction • Output of AEAD_ENC(pwd2, pwd3, usnc1) obtained by reconstructing with pwd2, pwd3, usnc1. (Analysis 1642)

Deduction • Output of AEAD_ENC(usnc2, pwd, tp2) obtained by reconstructing with usnc2, pwd, tp2. (Analysis 1644)

Deduction • Output of AEAD_ENC(usnc3, usnc3, usnp) obtained by reconstructing with usnc3, usnc3, usnp. (Analysis 1644)

Deduction • Output of AEAD_ENC(pwd2, pwd3, pwd) obtained by reconstructing with pwd2, pwd3, pwd. (Analysis 1644)

Deduction • Output of AEAD_ENC(usnc3, nil, pwd) obtained by reconstructing with usnc3, nil, pwd. (Analysis 1644)

Deduction • Output of AEAD_ENC(usnc2, pwd1, nil) obtained by reconstructing with usnc2, pwd1, nil. (Analysis 1644)

Deduction • Output of AEAD_ENC(usnc2, nil, tp2) obtained by reconstructing with usnc2, nil, tp2. (Analysis 1656)

Deduction • Output of AEAD_ENC(usnc2, usnp, pwd2) obtained by reconstructing with usnc2, usnp, pwd2. (Analysis 1658)

Deduction • Output of AEAD_ENC(pwd, tp2, usnc3) obtained by reconstructing with pwd, tp2, usnc3. (Analysis 1658)

Deduction • Output of AEAD_ENC(pwd, tp2, pwd3) obtained by reconstructing with pwd, tp2, pwd3. (Analysis 1658)

Deduction • Output of AEAD_ENC(usnc2, pwd1, tp2) obtained by reconstructing with usnc2, pwd1, tp2. (Analysis 1659)

Deduction • Output of AEAD_ENC(pwd, tp2, pwd) obtained by reconstructing with pwd, tp2, pwd. (Analysis 1659)

Deduction • Output of AEAD_ENC(pwd, pwd, pwd3) obtained by reconstructing with pwd, pwd, pwd3. (Analysis 1659)

Deduction • Output of AEAD_ENC(pwd, usnc1, pwd3) obtained by reconstructing with pwd, usnc1, pwd3. (Analysis 1661)

Deduction • Output of AEAD_ENC(pwd, usnc1, pwd2) obtained by reconstructing with pwd, usnc1, pwd2. (Analysis 1663)

Deduction • Output of AEAD_ENC(pwd, usnc1, usnp) obtained by reconstructing with pwd, usnc1, usnp. (Analysis 1664)

Deduction • Output of AEAD_ENC(pwd, usnc1, tp1) obtained by reconstructing with pwd, usnc1, tp1. (Analysis 1665)

Deduction • Output of AEAD_ENC(usnc3, pwd3, usnc2) obtained by reconstructing with usnc3, pwd3, usnc2. (Analysis 1665)

Deduction • Output of AEAD_ENC(pwd, pwd1, usnc3) obtained by reconstructing with pwd, pwd1, usnc3. (Analysis 1668)

Deduction • Output of AEAD_ENC(pwd, usnc1, pwd) obtained by reconstructing with pwd, usnc1, pwd. (Analysis 1669)

Deduction • Output of AEAD_ENC(pwd, pwd1, pwd3) obtained by reconstructing with pwd, pwd1, pwd3. (Analysis 1672)

Deduction • Output of AEAD_ENC(pwd, usnc2, tp1) obtained by reconstructing with pwd, usnc2, tp1. (Analysis 1674)

Deduction • Output of AEAD_ENC(pwd, usnc2, usnc1) obtained by reconstructing with pwd, usnc2, usnc1. (Analysis 1675)

Deduction • Output of AEAD_ENC(pwd, usnc2, pwd2) obtained by reconstructing with pwd, usnc2, pwd2. (Analysis 1676)

Deduction • Output of AEAD_ENC(pwd2, pwd1, pwd1) obtained by reconstructing with pwd2, pwd1, pwd1. (Analysis 1678)

Deduction • Output of AEAD_ENC(usnc2, pwd, usnc3) obtained by reconstructing with usnc2, pwd, usnc3. (Analysis 1678)

Deduction • Output of AEAD_ENC(pwd, usnc2, pwd3) obtained by reconstructing with pwd, usnc2, pwd3. (Analysis 1679)

Deduction • Output of AEAD_ENC(usnc3, usnp, nil) obtained by reconstructing with usnc3, usnp, nil. (Analysis 1679)

Deduction • Output of AEAD_ENC(usnc2, usnc3, pwd1) obtained by reconstructing with usnc2, usnc3, pwd1. (Analysis 1682)

Deduction • Output of AEAD_ENC(usnc2, pwd, pwd2) obtained by reconstructing with usnc2, pwd, pwd2. (Analysis 1684)

Deduction • Output of AEAD_ENC(usnc2, usnc2, usnc1) obtained by reconstructing with usnc2, usnc2, usnc1. (Analysis 1686)

Deduction • Output of AEAD_ENC(usnc3, pwd, tp2) obtained by reconstructing with usnc3, pwd, tp2. (Analysis 1690)

Deduction • Output of AEAD_ENC(usnc2, usnc3, usnc3) obtained by reconstructing with usnc2, usnc3, usnc3. (Analysis 1693)

Deduction • Output of AEAD_ENC(pwd3, usnc1, tp2) obtained by reconstructing with pwd3, usnc1, tp2. (Analysis 1693)

Deduction • Output of AEAD_ENC(usnc2, usnc1, tp1) obtained by reconstructing with usnc2, usnc1, tp1. (Analysis 1695)

Deduction • Output of AEAD_ENC(usnc3, pwd, nil) obtained by reconstructing with usnc3, pwd, nil. (Analysis 1698)

Deduction • Output of AEAD_ENC(usnc3, usnp, usnc3) obtained by reconstructing with usnc3, usnp, usnc3. (Analysis 1706)

Deduction • Output of AEAD_ENC(pwd, usnc3, tp1) obtained by reconstructing with pwd, usnc3, tp1. (Analysis 1708)

Deduction • Output of AEAD_ENC(usnc2, pwd, pwd1) obtained by reconstructing with usnc2, pwd, pwd1. (Analysis 1708)

Deduction • Output of AEAD_ENC(pwd, usnc3, tp2) obtained by reconstructing with pwd, usnc3, tp2. (Analysis 1708)

Deduction • Output of AEAD_ENC(usnc3, usnp, pwd3) obtained by reconstructing with usnc3, usnp, pwd3. (Analysis 1711)

Deduction • Output of AEAD_ENC(pwd, pwd3, usnc3) obtained by reconstructing with pwd, pwd3, usnc3. (Analysis 1712)

Deduction • Output of AEAD_ENC(pwd, pwd, pwd) obtained by reconstructing with pwd, pwd, pwd. (Analysis 1712)

Deduction • Output of AEAD_ENC(pwd, pwd2, pwd3) obtained by reconstructing with pwd, pwd2, pwd3. (Analysis 1718)

Deduction • Output of AEAD_ENC(usnc3, usnp, pwd) obtained by reconstructing with usnc3, usnp, pwd. (Analysis 1719)

Deduction • Output of AEAD_ENC(pwd, pwd3, usnp) obtained by reconstructing with pwd, pwd3, usnp. (Analysis 1720)

Deduction • Output of AEAD_ENC(pwd, usnc3, nil) obtained by reconstructing with pwd, usnc3, nil. (Analysis 1722)

Deduction • Output of AEAD_ENC(pwd, pwd3, pwd3) obtained by reconstructing with pwd, pwd3, pwd3. (Analysis 1722)

Deduction • Output of AEAD_ENC(pwd, pwd2, pwd2) obtained by reconstructing with pwd, pwd2, pwd2. (Analysis 1724)

Deduction • Output of AEAD_ENC(usnc2, usnp, pwd3) obtained by reconstructing with usnc2, usnp, pwd3. (Analysis 1724)

Deduction • Output of AEAD_ENC(pwd, usnp, pwd3) obtained by reconstructing with pwd, usnp, pwd3. (Analysis 1726)

Deduction • Output of AEAD_ENC(usnc3, pwd, usnc2) obtained by reconstructing with usnc3, pwd, usnc2. (Analysis 1731)

Deduction • Output of AEAD_ENC(pwd3, pwd1, usnc2) obtained by reconstructing with pwd3, pwd1, usnc2. (Analysis 1733)

Deduction • Output of AEAD_ENC(usnc3, pwd, usnc1) obtained by reconstructing with usnc3, pwd, usnc1. (Analysis 1735)

Deduction • Output of AEAD_ENC(pwd2, usnc1, usnc2) obtained by reconstructing with pwd2, usnc1, usnc2. (Analysis 1736)

Deduction • Output of AEAD_ENC(usnc3, pwd, usnc3) obtained by reconstructing with usnc3, pwd, usnc3. (Analysis 1739)

Deduction • Output of AEAD_ENC(pwd3, usnc2, nil) obtained by reconstructing with pwd3, usnc2, nil. (Analysis 1741)

Deduction • Output of AEAD_ENC(usnc2, usnc2, tp1) obtained by reconstructing with usnc2, usnc2, tp1. (Analysis 1741)

Deduction • Output of AEAD_ENC(pwd3, usnc2, tp2) obtained by reconstructing with pwd3, usnc2, tp2. (Analysis 1741)

Deduction • Output of AEAD_ENC(pwd2, usnc1, pwd) obtained by reconstructing with pwd2, usnc1, pwd. (Analysis 1746)

Deduction • Output of AEAD_ENC(pwd3, usnc2, pwd3) obtained by reconstructing with pwd3, usnc2, pwd3. (Analysis 1749)

Deduction • Output of AEAD_ENC(pwd2, pwd1, tp1) obtained by reconstructing with pwd2, pwd1, tp1. (Analysis 1751)

Deduction • Output of AEAD_ENC(pwd3, nil, usnp) obtained by reconstructing with pwd3, nil, usnp. (Analysis 1752)

Deduction • Output of AEAD_ENC(usnc2, pwd, pwd) obtained by reconstructing with usnc2, pwd, pwd. (Analysis 1752)

Deduction • Output of AEAD_ENC(usnc2, usnc2, pwd3) obtained by reconstructing with usnc2, usnc2, pwd3. (Analysis 1758)

Deduction • Output of AEAD_ENC(pwd3, usnc1, usnc1) obtained by reconstructing with pwd3, usnc1, usnc1. (Analysis 1760)

Deduction • Output of AEAD_ENC(pwd3, tp1, tp2) obtained by reconstructing with pwd3, tp1, tp2. (Analysis 1761)

Deduction • Output of AEAD_ENC(usnc2, usnc2, usnc3) obtained by reconstructing with usnc2, usnc2, usnc3. (Analysis 1763)

Deduction • Output of AEAD_ENC(usnc2, usnc2, usnc2) obtained by reconstructing with usnc2, usnc2, usnc2. (Analysis 1764)

Deduction • Output of AEAD_ENC(pwd1, pwd, tp1) obtained by reconstructing with pwd1, pwd, tp1. (Analysis 1765)

Deduction • Output of AEAD_ENC(pwd2, tp1, tp2) obtained by reconstructing with pwd2, tp1, tp2. (Analysis 1765)

Deduction • Output of AEAD_ENC(usnc2, tp1, nil) obtained by reconstructing with usnc2, tp1, nil. (Analysis 1765)

Deduction • Output of AEAD_ENC(pwd3, pwd2, nil) obtained by reconstructing with pwd3, pwd2, nil. (Analysis 1771)

Deduction • Output of AEAD_ENC(usnc2, usnc2, usnp) obtained by reconstructing with usnc2, usnc2, usnp. (Analysis 1765)

Deduction • Output of AEAD_ENC(pwd2, usnc2, tp1) obtained by reconstructing with pwd2, usnc2, tp1. (Analysis 1781)

Deduction • Output of AEAD_ENC(usnc2, nil, pwd) obtained by reconstructing with usnc2, nil, pwd. (Analysis 1781)

Deduction • Output of AEAD_ENC(pwd2, usnc1, usnp) obtained by reconstructing with pwd2, usnc1, usnp. (Analysis 1784)

Deduction • Output of AEAD_ENC(pwd3, tp1, usnc1) obtained by reconstructing with pwd3, tp1, usnc1. (Analysis 1787)

Deduction • Output of AEAD_ENC(usnc2, tp1, usnc2) obtained by reconstructing with usnc2, tp1, usnc2. (Analysis 1788)

Deduction • Output of AEAD_ENC(usnc2, pwd1, usnp) obtained by reconstructing with usnc2, pwd1, usnp. (Analysis 1788)

Deduction • Output of AEAD_ENC(pwd2, nil, pwd) obtained by reconstructing with pwd2, nil, pwd. (Analysis 1789)

Deduction • Output of AEAD_ENC(pwd2, pwd1, pwd2) obtained by reconstructing with pwd2, pwd1, pwd2. (Analysis 1801)

Deduction • Output of AEAD_ENC(pwd3, usnp, usnc3) obtained by reconstructing with pwd3, usnp, usnc3. (Analysis 1801)

Deduction • Output of AEAD_ENC(usnc2, tp1, usnc1) obtained by reconstructing with usnc2, tp1, usnc1. (Analysis 1801)

Deduction • Output of AEAD_ENC(pwd3, usnc1, nil) obtained by reconstructing with pwd3, usnc1, nil. (Analysis 1801)

Deduction • Output of AEAD_ENC(usnc2, pwd3, tp1) obtained by reconstructing with usnc2, pwd3, tp1. (Analysis 1801)

Deduction • Output of AEAD_ENC(pwd2, usnc2, pwd2) obtained by reconstructing with pwd2, usnc2, pwd2. (Analysis 1801)

Deduction • Output of AEAD_ENC(pwd3, pwd1, tp1) obtained by reconstructing with pwd3, pwd1, tp1. (Analysis 1802)

Deduction • Output of AEAD_ENC(pwd3, pwd1, usnp) obtained by reconstructing with pwd3, pwd1, usnp. (Analysis 1808)

Deduction • Output of AEAD_ENC(pwd3, usnc3, pwd3) obtained by reconstructing with pwd3, usnc3, pwd3. (Analysis 1808)

Deduction • Output of AEAD_ENC(pwd3, pwd1, pwd1) obtained by reconstructing with pwd3, pwd1, pwd1. (Analysis 1809)

Deduction • Output of AEAD_ENC(pwd3, pwd1, tp2) obtained by reconstructing with pwd3, pwd1, tp2. (Analysis 1809)

Deduction • Output of AEAD_ENC(pwd3, pwd3, usnc2) obtained by reconstructing with pwd3, pwd3, usnc2. (Analysis 1810)

Deduction • Output of AEAD_ENC(usnc2, tp1, tp2) obtained by reconstructing with usnc2, tp1, tp2. (Analysis 1810)

Deduction • Output of AEAD_ENC(usnc2, tp1, pwd) obtained by reconstructing with usnc2, tp1, pwd. (Analysis 1812)

Deduction • Output of AEAD_ENC(pwd2, usnc2, usnc3) obtained by reconstructing with pwd2, usnc2, usnc3. (Analysis 1815)

Deduction • Output of AEAD_ENC(pwd3, usnc2, tp1) obtained by reconstructing with pwd3, usnc2, tp1. (Analysis 1817)

Deduction • Output of AEAD_ENC(pwd3, pwd1, usnc1) obtained by reconstructing with pwd3, pwd1, usnc1. (Analysis 1820)

Deduction • Output of AEAD_ENC(pwd2, pwd1, pwd3) obtained by reconstructing with pwd2, pwd1, pwd3. (Analysis 1821)

Deduction • Output of AEAD_ENC(pwd3, pwd3, tp2) obtained by reconstructing with pwd3, pwd3, tp2. (Analysis 1822)

Deduction • Output of AEAD_ENC(pwd3, usnp, usnc1) obtained by reconstructing with pwd3, usnp, usnc1. (Analysis 1822)

Deduction • Output of AEAD_ENC(pwd3, usnc3, nil) obtained by reconstructing with pwd3, usnc3, nil. (Analysis 1822)

Deduction • Output of AEAD_ENC(pwd3, pwd2, usnc2) obtained by reconstructing with pwd3, pwd2, usnc2. (Analysis 1824)

Deduction • Output of AEAD_ENC(pwd3, pwd3, pwd) obtained by reconstructing with pwd3, pwd3, pwd. (Analysis 1824)

Deduction • Output of AEAD_ENC(pwd3, usnp, pwd2) obtained by reconstructing with pwd3, usnp, pwd2. (Analysis 1825)

Deduction • Output of AEAD_ENC(pwd2, pwd2, pwd1) obtained by reconstructing with pwd2, pwd2, pwd1. (Analysis 1825)

Deduction • Output of AEAD_ENC(usnp, nil, pwd1) obtained by reconstructing with usnp, nil, pwd1. (Analysis 1828)

Deduction • Output of AEAD_ENC(pwd2, pwd2, tp1) obtained by reconstructing with pwd2, pwd2, tp1. (Analysis 1830)

Deduction • Output of AEAD_ENC(pwd3, usnc3, pwd) obtained by reconstructing with pwd3, usnc3, pwd. (Analysis 1831)

Deduction • Output of AEAD_ENC(pwd3, pwd, pwd1) obtained by reconstructing with pwd3, pwd, pwd1. (Analysis 1832)

Deduction • Output of AEAD_ENC(usnp, nil, nil) obtained by reconstructing with usnp, nil, nil. (Analysis 1832)

Deduction • Output of AEAD_ENC(pwd3, usnc2, pwd) obtained by reconstructing with pwd3, usnc2, pwd. (Analysis 1836)

Deduction • Output of AEAD_ENC(pwd2, usnc3, usnc3) obtained by reconstructing with pwd2, usnc3, usnc3. (Analysis 1838)

Deduction • Output of AEAD_ENC(pwd3, pwd2, tp1) obtained by reconstructing with pwd3, pwd2, tp1. (Analysis 1838)

Deduction • Output of AEAD_ENC(pwd3, pwd2, usnc3) obtained by reconstructing with pwd3, pwd2, usnc3. (Analysis 1838)

Deduction • Output of AEAD_ENC(pwd3, pwd2, pwd) obtained by reconstructing with pwd3, pwd2, pwd. (Analysis 1841)

Deduction • Output of AEAD_ENC(usnp, tp2, tp2) obtained by reconstructing with usnp, tp2, tp2. (Analysis 1841)

Deduction • Output of AEAD_ENC(usnp, tp2, pwd1) obtained by reconstructing with usnp, tp2, pwd1. (Analysis 1841)

Deduction • Output of AEAD_ENC(usnp, tp2, usnc2) obtained by reconstructing with usnp, tp2, usnc2. (Analysis 1841)

Deduction • Output of AEAD_ENC(usnp, usnc1, nil) obtained by reconstructing with usnp, usnc1, nil. (Analysis 1846)

Deduction • Output of AEAD_ENC(usnp, usnc1, tp2) obtained by reconstructing with usnp, usnc1, tp2. (Analysis 1846)

Deduction • Output of AEAD_ENC(pwd3, usnc3, usnp) obtained by reconstructing with pwd3, usnc3, usnp. (Analysis 1855)

Deduction • Output of AEAD_ENC(pwd2, usnp, tp1) obtained by reconstructing with pwd2, usnp, tp1. (Analysis 1857)

Deduction • Output of AEAD_ENC(pwd3, pwd3, nil) obtained by reconstructing with pwd3, pwd3, nil. (Analysis 1864)

Deduction • Output of AEAD_ENC(pwd3, pwd3, usnc1) obtained by reconstructing with pwd3, pwd3, usnc1. (Analysis 1865)

Deduction • Output of AEAD_ENC(usnp, pwd1, pwd1) obtained by reconstructing with usnp, pwd1, pwd1. (Analysis 1865)

Deduction • Output of AEAD_ENC(pwd2, pwd, usnp) obtained by reconstructing with pwd2, pwd, usnp. (Analysis 1867)

Deduction • Output of AEAD_ENC(pwd3, usnc3, pwd2) obtained by reconstructing with pwd3, usnc3, pwd2. (Analysis 1867)

Deduction • Output of AEAD_ENC(usnp, usnc1, usnp) obtained by reconstructing with usnp, usnc1, usnp. (Analysis 1868)

Deduction • Output of AEAD_ENC(pwd3, usnp, pwd) obtained by reconstructing with pwd3, usnp, pwd. (Analysis 1872)

Deduction • Output of AEAD_ENC(usnp, usnc3, pwd2) obtained by reconstructing with usnp, usnc3, pwd2. (Analysis 1872)

Deduction • Output of AEAD_ENC(pwd3, usnp, pwd1) obtained by reconstructing with pwd3, usnp, pwd1. (Analysis 1872)

Deduction • Output of AEAD_ENC(usnp, pwd1, usnc2) obtained by reconstructing with usnp, pwd1, usnc2. (Analysis 1874)

Deduction • Output of AEAD_ENC(usnp, usnc2, tp2) obtained by reconstructing with usnp, usnc2, tp2. (Analysis 1878)

Deduction • Output of AEAD_ENC(usnp, usnc3, pwd3) obtained by reconstructing with usnp, usnc3, pwd3. (Analysis 1878)

Deduction • Output of AEAD_ENC(usnp, usnc3, usnp) obtained by reconstructing with usnp, usnc3, usnp. (Analysis 1880)

Deduction • Output of AEAD_ENC(usnp, pwd2, usnc1) obtained by reconstructing with usnp, pwd2, usnc1. (Analysis 1881)

Deduction • Output of AEAD_ENC(usnp, pwd2, pwd3) obtained by reconstructing with usnp, pwd2, pwd3. (Analysis 1884)

Deduction • Output of AEAD_ENC(usnp, usnc3, tp2) obtained by reconstructing with usnp, usnc3, tp2. (Analysis 1884)

Deduction • Output of AEAD_ENC(usnp, usnc3, usnc1) obtained by reconstructing with usnp, usnc3, usnc1. (Analysis 1884)

Deduction • Output of AEAD_ENC(usnp, usnc2, usnc2) obtained by reconstructing with usnp, usnc2, usnc2. (Analysis 1884)

Deduction • Output of AEAD_ENC(usnp, pwd3, tp2) obtained by reconstructing with usnp, pwd3, tp2. (Analysis 1888)

Deduction • Output of AEAD_ENC(usnp, nil, pwd2) obtained by reconstructing with usnp, nil, pwd2. (Analysis 1891)

Deduction • Output of AEAD_ENC(usnp, pwd3, tp1) obtained by reconstructing with usnp, pwd3, tp1. (Analysis 1892)

Deduction • Output of AEAD_ENC(usnp, pwd3, usnc2) obtained by reconstructing with usnp, pwd3, usnc2. (Analysis 1892)

Deduction • Output of AEAD_ENC(usnp, nil, usnc3) obtained by reconstructing with usnp, nil, usnc3. (Analysis 1893)

Deduction • Output of AEAD_ENC(usnp, pwd3, usnc1) obtained by reconstructing with usnp, pwd3, usnc1. (Analysis 1893)

Deduction • Output of AEAD_ENC(usnp, pwd3, pwd1) obtained by reconstructing with usnp, pwd3, pwd1. (Analysis 1893)

Deduction • Output of AEAD_ENC(usnp, nil, pwd3) obtained by reconstructing with usnp, nil, pwd3. (Analysis 1894)

Deduction • Output of AEAD_ENC(usnp, tp1, nil) obtained by reconstructing with usnp, tp1, nil. (Analysis 1900)

Deduction • Output of AEAD_ENC(usnp, tp1, tp1) obtained by reconstructing with usnp, tp1, tp1. (Analysis 1900)

Deduction • Output of AEAD_ENC(pwd, nil, tp2) obtained by reconstructing with pwd, nil, tp2. (Analysis 1901)

Deduction • Output of AEAD_ENC(usnp, usnp, tp1) obtained by reconstructing with usnp, usnp, tp1. (Analysis 1901)

Deduction • Output of AEAD_ENC(pwd, nil, pwd) obtained by reconstructing with pwd, nil, pwd. (Analysis 1901)

Deduction • Output of AEAD_ENC(usnp, tp1, pwd3) obtained by reconstructing with usnp, tp1, pwd3. (Analysis 1903)

Deduction • Output of AEAD_ENC(pwd, nil, usnc1) obtained by reconstructing with pwd, nil, usnc1. (Analysis 1903)

Deduction • Output of AEAD_ENC(usnp, usnc1, tp1) obtained by reconstructing with usnp, usnc1, tp1. (Analysis 1904)

Deduction • Output of AEAD_ENC(usnp, usnc2, pwd1) obtained by reconstructing with usnp, usnc2, pwd1. (Analysis 1904)

Deduction • Output of AEAD_ENC(pwd, tp1, usnc3) obtained by reconstructing with pwd, tp1, usnc3. (Analysis 1905)

Deduction • Output of AEAD_ENC(usnp, pwd, pwd3) obtained by reconstructing with usnp, pwd, pwd3. (Analysis 1906)

Deduction • Output of AEAD_ENC(usnp, tp2, usnc3) obtained by reconstructing with usnp, tp2, usnc3. (Analysis 1911)

Deduction • Output of AEAD_ENC(usnp, usnc1, usnc3) obtained by reconstructing with usnp, usnc1, usnc3. (Analysis 1913)

Deduction • Output of AEAD_ENC(usnp, tp1, pwd) obtained by reconstructing with usnp, tp1, pwd. (Analysis 1915)

Deduction • Output of AEAD_ENC(pwd, tp1, pwd2) obtained by reconstructing with pwd, tp1, pwd2. (Analysis 1915)

Deduction • Output of AEAD_ENC(pwd, tp2, pwd2) obtained by reconstructing with pwd, tp2, pwd2. (Analysis 1915)

Deduction • Output of AEAD_ENC(usnp, pwd1, pwd) obtained by reconstructing with usnp, pwd1, pwd. (Analysis 1919)

Deduction • Output of AEAD_ENC(usnp, usnc2, usnc1) obtained by reconstructing with usnp, usnc2, usnc1. (Analysis 1920)

Deduction • Output of AEAD_ENC(usnp, usnc2, nil) obtained by reconstructing with usnp, usnc2, nil. (Analysis 1921)

Deduction • Output of AEAD_ENC(pwd, usnc1, tp2) obtained by reconstructing with pwd, usnc1, tp2. (Analysis 1926)

Deduction • Output of AEAD_ENC(pwd, usnc1, usnc1) obtained by reconstructing with pwd, usnc1, usnc1. (Analysis 1926)

Deduction • Output of AEAD_ENC(pwd, usnc1, pwd1) obtained by reconstructing with pwd, usnc1, pwd1. (Analysis 1929)

Deduction • Output of AEAD_ENC(pwd, usnc1, usnc2) obtained by reconstructing with pwd, usnc1, usnc2. (Analysis 1930)

Deduction • Output of AEAD_ENC(pwd, pwd1, usnc1) obtained by reconstructing with pwd, pwd1, usnc1. (Analysis 1930)

Deduction • Output of AEAD_ENC(pwd, pwd1, pwd2) obtained by reconstructing with pwd, pwd1, pwd2. (Analysis 1931)

Deduction • Output of AEAD_ENC(pwd2, pwd1, tp2) obtained by reconstructing with pwd2, pwd1, tp2. (Analysis 1944)

Deduction • Output of AEAD_ENC(usnc3, usnc1, tp2) obtained by reconstructing with usnc3, usnc1, tp2. (Analysis 1946)

Deduction • Output of AEAD_ENC(usnc2, usnp, tp2) obtained by reconstructing with usnc2, usnp, tp2. (Analysis 1946)

Deduction • Output of AEAD_ENC(usnc3, nil, usnc1) obtained by reconstructing with usnc3, nil, usnc1. (Analysis 1949)

Deduction • Output of AEAD_ENC(usnc3, usnc1, usnc3) obtained by reconstructing with usnc3, usnc1, usnc3. (Analysis 1949)

Deduction • Output of AEAD_ENC(usnc3, tp2, nil) obtained by reconstructing with usnc3, tp2, nil. (Analysis 1950)

Deduction • Output of AEAD_ENC(usnc3, nil, pwd3) obtained by reconstructing with usnc3, nil, pwd3. (Analysis 1950)

Deduction • Output of AEAD_ENC(usnp, pwd, nil) obtained by reconstructing with usnp, pwd, nil. (Analysis 1951)

Deduction • Output of AEAD_ENC(usnp, usnp, usnc2) obtained by reconstructing with usnp, usnp, usnc2. (Analysis 1951)

Deduction • Output of AEAD_ENC(pwd, usnp, tp1) obtained by reconstructing with pwd, usnp, tp1. (Analysis 1954)

Deduction • Output of AEAD_ENC(pwd, usnc3, usnc3) obtained by reconstructing with pwd, usnc3, usnc3. (Analysis 1958)

Deduction • Output of AEAD_ENC(usnc3, tp1, nil) obtained by reconstructing with usnc3, tp1, nil. (Analysis 1958)

Deduction • Output of AEAD_ENC(pwd, pwd2, tp1) obtained by reconstructing with pwd, pwd2, tp1. (Analysis 1958)

Deduction • Output of AEAD_ENC(usnp, pwd, usnc2) obtained by reconstructing with usnp, pwd, usnc2. (Analysis 1959)

Deduction • Output of AEAD_ENC(pwd, pwd2, pwd) obtained by reconstructing with pwd, pwd2, pwd. (Analysis 1962)

Deduction • Output of AEAD_ENC(pwd2, usnc2, pwd) obtained by reconstructing with pwd2, usnc2, pwd. (Analysis 1963)

Deduction • Output of AEAD_ENC(usnp, usnp, pwd2) obtained by reconstructing with usnp, usnp, pwd2. (Analysis 1967)

Deduction • Output of AEAD_ENC(pwd, pwd2, usnc3) obtained by reconstructing with pwd, pwd2, usnc3. (Analysis 1982)

Deduction • Output of AEAD_ENC(usnc3, tp2, tp1) obtained by reconstructing with usnc3, tp2, tp1. (Analysis 1986)

Deduction • Output of AEAD_ENC(usnc2, nil, usnp) obtained by reconstructing with usnc2, nil, usnp. (Analysis 1989)

Deduction • Output of AEAD_ENC(usnc3, pwd1, tp2) obtained by reconstructing with usnc3, pwd1, tp2. (Analysis 1991)

Deduction • Output of AEAD_ENC(usnc3, usnc1, pwd1) obtained by reconstructing with usnc3, usnc1, pwd1. (Analysis 1991)

Deduction • Output of AEAD_ENC(usnc3, usnc3, pwd3) obtained by reconstructing with usnc3, usnc3, pwd3. (Analysis 1991)

Deduction • Output of AEAD_ENC(usnc3, tp1, usnc2) obtained by reconstructing with usnc3, tp1, usnc2. (Analysis 2004)

Deduction • Output of AEAD_ENC(pwd, pwd2, nil) obtained by reconstructing with pwd, pwd2, nil. (Analysis 2013)

Deduction • Output of AEAD_ENC(pwd3, nil, usnc2) obtained by reconstructing with pwd3, nil, usnc2. (Analysis 2013)

Deduction • Output of AEAD_ENC(usnc3, pwd2, tp1) obtained by reconstructing with usnc3, pwd2, tp1. (Analysis 2015)

Deduction • Output of AEAD_ENC(usnc3, tp2, pwd3) obtained by reconstructing with usnc3, tp2, pwd3. (Analysis 2015)

Deduction • Output of AEAD_ENC(usnc3, usnc2, usnc1) obtained by reconstructing with usnc3, usnc2, usnc1. (Analysis 1991)

Deduction • Output of AEAD_ENC(usnc3, usnc3, pwd2) obtained by reconstructing with usnc3, usnc3, pwd2. (Analysis 2030)

Deduction • Output of AEAD_ENC(pwd3, tp2, pwd3) obtained by reconstructing with pwd3, tp2, pwd3. (Analysis 2034)

Deduction • Output of AEAD_ENC(usnc3, nil, pwd2) obtained by reconstructing with usnc3, nil, pwd2. (Analysis 2040)

Deduction • Output of AEAD_ENC(usnc3, pwd3, pwd3) obtained by reconstructing with usnc3, pwd3, pwd3. (Analysis 2040)

Deduction • Output of AEAD_ENC(usnc3, usnc3, nil) obtained by reconstructing with usnc3, usnc3, nil. (Analysis 2040)

Deduction • Output of AEAD_ENC(pwd2, usnc1, pwd2) obtained by reconstructing with pwd2, usnc1, pwd2. (Analysis 2041)

Deduction • Output of AEAD_ENC(usnc3, pwd2, pwd) obtained by reconstructing with usnc3, pwd2, pwd. (Analysis 2041)

Deduction • Output of AEAD_ENC(pwd2, usnc2, usnp) obtained by reconstructing with pwd2, usnc2, usnp. (Analysis 2042)

Deduction • Output of AEAD_ENC(usnc3, pwd2, usnc3) obtained by reconstructing with usnc3, pwd2, usnc3. (Analysis 2042)

Deduction • Output of AEAD_ENC(pwd3, pwd1, nil) obtained by reconstructing with pwd3, pwd1, nil. (Analysis 2043)

Deduction • Output of AEAD_ENC(usnc3, pwd3, tp2) obtained by reconstructing with usnc3, pwd3, tp2. (Analysis 2044)

Deduction • Output of AEAD_ENC(pwd, usnp, usnc2) obtained by reconstructing with pwd, usnp, usnc2. (Analysis 2047)

Deduction • Output of AEAD_ENC(pwd2, pwd2, usnc2) obtained by reconstructing with pwd2, pwd2, usnc2. (Analysis 2052)

Deduction • Output of AEAD_ENC(usnc3, pwd3, pwd) obtained by reconstructing with usnc3, pwd3, pwd. (Analysis 2054)

Deduction • Output of AEAD_ENC(pwd2, usnc1, tp2) obtained by reconstructing with pwd2, usnc1, tp2. (Analysis 2060)

Deduction • Output of AEAD_ENC(pwd, pwd, usnp) obtained by reconstructing with pwd, pwd, usnp. (Analysis 2065)

Deduction • Output of AEAD_ENC(pwd, pwd, pwd2) obtained by reconstructing with pwd, pwd, pwd2. (Analysis 2066)

Deduction • Output of AEAD_ENC(pwd3, nil, usnc3) obtained by reconstructing with pwd3, nil, usnc3. (Analysis 2067)

Deduction • Output of AEAD_ENC(pwd, tp2, usnp) obtained by reconstructing with pwd, tp2, usnp. (Analysis 2068)

Deduction • Output of AEAD_ENC(usnc3, usnc1, usnc1) obtained by reconstructing with usnc3, usnc1, usnc1. (Analysis 2069)

Deduction • Output of AEAD_ENC(usnc3, tp2, usnp) obtained by reconstructing with usnc3, tp2, usnp. (Analysis 2070)

Deduction • Output of AEAD_ENC(pwd, pwd1, usnp) obtained by reconstructing with pwd, pwd1, usnp. (Analysis 2078)

Deduction • Output of AEAD_ENC(usnc2, pwd3, nil) obtained by reconstructing with usnc2, pwd3, nil. (Analysis 2080)

Deduction • Output of AEAD_ENC(pwd, usnc2, pwd1) obtained by reconstructing with pwd, usnc2, pwd1. (Analysis 2081)

Deduction • Output of AEAD_ENC(pwd, usnc2, tp2) obtained by reconstructing with pwd, usnc2, tp2. (Analysis 2082)

Deduction • Output of AEAD_ENC(pwd, usnc2, usnc2) obtained by reconstructing with pwd, usnc2, usnc2. (Analysis 2082)

Deduction • Output of AEAD_ENC(pwd1, pwd, pwd3) obtained by reconstructing with pwd1, pwd, pwd3. (Analysis 2083)

Deduction • Output of AEAD_ENC(usnc3, usnp, usnp) obtained by reconstructing with usnc3, usnp, usnp. (Analysis 2086)

Deduction • Output of AEAD_ENC(usnc2, usnp, usnc3) obtained by reconstructing with usnc2, usnp, usnc3. (Analysis 2100)

Deduction • Output of AEAD_ENC(usnc3, usnp, tp1) obtained by reconstructing with usnc3, usnp, tp1. (Analysis 2102)

Deduction • Output of AEAD_ENC(pwd, usnp, usnc3) obtained by reconstructing with pwd, usnp, usnc3. (Analysis 2105)

Deduction • Output of AEAD_ENC(pwd, usnp, pwd2) obtained by reconstructing with pwd, usnp, pwd2. (Analysis 2106)

Deduction • Output of AEAD_ENC(pwd2, nil, tp1) obtained by reconstructing with pwd2, nil, tp1. (Analysis 2108)

Deduction • Output of AEAD_ENC(pwd2, usnc2, pwd3) obtained by reconstructing with pwd2, usnc2, pwd3. (Analysis 2112)

Deduction • Output of AEAD_ENC(usnc3, pwd, pwd3) obtained by reconstructing with usnc3, pwd, pwd3. (Analysis 2113)

Deduction • Output of AEAD_ENC(pwd3, tp1, pwd1) obtained by reconstructing with pwd3, tp1, pwd1. (Analysis 2113)

Deduction • Output of AEAD_ENC(pwd3, pwd1, usnc3) obtained by reconstructing with pwd3, pwd1, usnc3. (Analysis 2115)

Deduction • Output of AEAD_ENC(pwd3, pwd1, pwd) obtained by reconstructing with pwd3, pwd1, pwd. (Analysis 2108)

Deduction • Output of AEAD_ENC(pwd3, usnp, nil) obtained by reconstructing with pwd3, usnp, nil. (Analysis 2120)

Deduction • Output of AEAD_ENC(pwd2, tp1, tp1) obtained by reconstructing with pwd2, tp1, tp1. (Analysis 2122)

Deduction • Output of AEAD_ENC(pwd3, pwd1, pwd2) obtained by reconstructing with pwd3, pwd1, pwd2. (Analysis 2124)

Deduction • Output of AEAD_ENC(pwd3, nil, pwd3) obtained by reconstructing with pwd3, nil, pwd3. (Analysis 2125)

Deduction • Output of AEAD_ENC(pwd3, pwd1, pwd3) obtained by reconstructing with pwd3, pwd1, pwd3. (Analysis 2108)

Deduction • Output of AEAD_ENC(pwd2, usnc1, usnc3) obtained by reconstructing with pwd2, usnc1, usnc3. (Analysis 2113)

Deduction • Output of AEAD_ENC(pwd3, tp2, usnc2) obtained by reconstructing with pwd3, tp2, usnc2. (Analysis 2126)

Deduction • Output of AEAD_ENC(pwd2, tp1, usnc2) obtained by reconstructing with pwd2, tp1, usnc2. (Analysis 2131)

Deduction • Output of AEAD_ENC(pwd3, pwd3, usnp) obtained by reconstructing with pwd3, pwd3, usnp. (Analysis 2131)

Deduction • Output of AEAD_ENC(pwd3, usnp, usnc2) obtained by reconstructing with pwd3, usnp, usnc2. (Analysis 2135)

Deduction • Output of AEAD_ENC(pwd, usnp, nil) obtained by reconstructing with pwd, usnp, nil. (Analysis 2137)

Deduction • Output of AEAD_ENC(pwd3, pwd, usnc1) obtained by reconstructing with pwd3, pwd, usnc1. (Analysis 2146)

Deduction • Output of AEAD_ENC(pwd3, pwd, usnc2) obtained by reconstructing with pwd3, pwd, usnc2. (Analysis 2152)

Deduction • Output of AEAD_ENC(pwd2, pwd2, tp2) obtained by reconstructing with pwd2, pwd2, tp2. (Analysis 2152)

Deduction • Output of AEAD_ENC(pwd2, usnc3, usnp) obtained by reconstructing with pwd2, usnc3, usnp. (Analysis 2152)

Deduction • Output of AEAD_ENC(usnp, nil, pwd) obtained by reconstructing with usnp, nil, pwd. (Analysis 2152)

Deduction • Output of AEAD_ENC(usnp, tp2, pwd) obtained by reconstructing with usnp, tp2, pwd. (Analysis 2153)

Deduction • Output of AEAD_ENC(pwd3, usnc2, pwd2) obtained by reconstructing with pwd3, usnc2, pwd2. (Analysis 2154)

Deduction • Output of AEAD_ENC(pwd3, usnc1, pwd) obtained by reconstructing with pwd3, usnc1, pwd. (Analysis 2154)

Deduction • Output of AEAD_ENC(pwd3, usnc1, tp1) obtained by reconstructing with pwd3, usnc1, tp1. (Analysis 2160)

Deduction • Output of AEAD_ENC(pwd3, usnc2, usnc2) obtained by reconstructing with pwd3, usnc2, usnc2. (Analysis 2160)

Deduction • Output of AEAD_ENC(usnp, tp2, usnp) obtained by reconstructing with usnp, tp2, usnp. (Analysis 2161)

Deduction • Output of AEAD_ENC(pwd2, pwd2, pwd) obtained by reconstructing with pwd2, pwd2, pwd. (Analysis 2166)

Deduction • Output of AEAD_ENC(pwd2, usnc3, usnc1) obtained by reconstructing with pwd2, usnc3, usnc1. (Analysis 2171)

Deduction • Output of AEAD_ENC(pwd3, pwd2, pwd2) obtained by reconstructing with pwd3, pwd2, pwd2. (Analysis 2174)

Deduction • Output of AEAD_ENC(usnp, pwd1, usnc1) obtained by reconstructing with usnp, pwd1, usnc1. (Analysis 2178)

Deduction • Output of AEAD_ENC(pwd2, pwd, usnc3) obtained by reconstructing with pwd2, pwd, usnc3. (Analysis 2183)

Deduction • Output of AEAD_ENC(pwd3, usnp, pwd3) obtained by reconstructing with pwd3, usnp, pwd3. (Analysis 2185)

Deduction • Output of AEAD_ENC(pwd3, usnp, usnp) obtained by reconstructing with pwd3, usnp, usnp. (Analysis 2185)

Deduction • Output of AEAD_ENC(usnp, pwd1, usnc3) obtained by reconstructing with usnp, pwd1, usnc3. (Analysis 2188)

Deduction • Output of AEAD_ENC(pwd2, usnp, usnp) obtained by reconstructing with pwd2, usnp, usnp. (Analysis 2190)

Deduction • Output of AEAD_ENC(usnp, pwd1, nil) obtained by reconstructing with usnp, pwd1, nil. (Analysis 2194)

Deduction • Output of AEAD_ENC(usnp, pwd2, usnc3) obtained by reconstructing with usnp, pwd2, usnc3. (Analysis 2198)

Deduction • Output of AEAD_ENC(pwd2, pwd, pwd1) obtained by reconstructing with pwd2, pwd, pwd1. (Analysis 2204)

Deduction • Output of AEAD_ENC(usnp, pwd3, pwd2) obtained by reconstructing with usnp, pwd3, pwd2. (Analysis 2204)

Deduction • Output of AEAD_ENC(usnp, pwd3, pwd3) obtained by reconstructing with usnp, pwd3, pwd3. (Analysis 2205)

Deduction • Output of AEAD_ENC(usnp, usnc3, usnc3) obtained by reconstructing with usnp, usnc3, usnc3. (Analysis 2208)

Deduction • Output of AEAD_ENC(usnp, tp1, usnc3) obtained by reconstructing with usnp, tp1, usnc3. (Analysis 2209)

Deduction • Output of AEAD_ENC(pwd, tp2, tp2) obtained by reconstructing with pwd, tp2, tp2. (Analysis 2211)

Deduction • Output of AEAD_ENC(usnp, pwd, pwd) obtained by reconstructing with usnp, pwd, pwd. (Analysis 2215)

Deduction • Output of AEAD_ENC(usnp, usnc2, pwd2) obtained by reconstructing with usnp, usnc2, pwd2. (Analysis 2215)

Deduction • Output of AEAD_ENC(pwd, tp2, usnc2) obtained by reconstructing with pwd, tp2, usnc2. (Analysis 2218)

Deduction • Output of AEAD_ENC(usnp, pwd, usnp) obtained by reconstructing with usnp, pwd, usnp. (Analysis 2218)

Deduction • Output of AEAD_ENC(usnc3, tp1, tp2) obtained by reconstructing with usnc3, tp1, tp2. (Analysis 2235)

Deduction • Output of AEAD_ENC(usnp, pwd, tp1) obtained by reconstructing with usnp, pwd, tp1. (Analysis 2236)

Deduction • Output of AEAD_ENC(usnp, usnp, usnc1) obtained by reconstructing with usnp, usnp, usnc1. (Analysis 2236)

Deduction • Output of AEAD_ENC(usnp, pwd, usnc1) obtained by reconstructing with usnp, pwd, usnc1. (Analysis 2241)

Deduction • Output of AEAD_ENC(usnp, pwd3, usnp) obtained by reconstructing with usnp, pwd3, usnp. (Analysis 2244)

Deduction • Output of AEAD_ENC(usnc3, pwd1, tp1) obtained by reconstructing with usnc3, pwd1, tp1. (Analysis 2246)

Deduction • Output of AEAD_ENC(pwd, nil, tp1) obtained by reconstructing with pwd, nil, tp1. (Analysis 2247)

Deduction • Output of AEAD_ENC(usnp, pwd, usnc3) obtained by reconstructing with usnp, pwd, usnc3. (Analysis 2249)

Deduction • Output of AEAD_ENC(usnp, usnp, pwd3) obtained by reconstructing with usnp, usnp, pwd3. (Analysis 2251)

Deduction • Output of AEAD_ENC(pwd, nil, nil) obtained by reconstructing with pwd, nil, nil. (Analysis 2252)

Deduction • Output of AEAD_ENC(usnp, usnp, usnc3) obtained by reconstructing with usnp, usnp, usnc3. (Analysis 2254)

Deduction • Output of AEAD_ENC(pwd, usnc3, usnp) obtained by reconstructing with pwd, usnc3, usnp. (Analysis 2259)

Deduction • Output of AEAD_ENC(pwd, usnc3, pwd) obtained by reconstructing with pwd, usnc3, pwd. (Analysis 2259)

Deduction • Output of AEAD_ENC(pwd, pwd3, nil) obtained by reconstructing with pwd, pwd3, nil. (Analysis 2262)

Deduction • Output of AEAD_ENC(usnc3, usnc2, tp2) obtained by reconstructing with usnc3, usnc2, tp2. (Analysis 2263)

Deduction • Output of AEAD_ENC(pwd, pwd, nil) obtained by reconstructing with pwd, pwd, nil. (Analysis 2263)

Deduction • Output of AEAD_ENC(usnc3, tp2, pwd1) obtained by reconstructing with usnc3, tp2, pwd1. (Analysis 2265)

Deduction • Output of AEAD_ENC(usnc3, tp1, pwd1) obtained by reconstructing with usnc3, tp1, pwd1. (Analysis 2266)

Deduction • Output of AEAD_ENC(pwd, tp2, tp1) obtained by reconstructing with pwd, tp2, tp1. (Analysis 2268)

Deduction • Output of AEAD_ENC(usnc3, tp1, usnc1) obtained by reconstructing with usnc3, tp1, usnc1. (Analysis 2269)

Deduction • Output of AEAD_ENC(pwd, pwd2, usnp) obtained by reconstructing with pwd, pwd2, usnp. (Analysis 2270)

Deduction • Output of AEAD_ENC(usnc3, tp1, pwd) obtained by reconstructing with usnc3, tp1, pwd. (Analysis 2276)

Deduction • Output of AEAD_ENC(pwd, nil, usnc3) obtained by reconstructing with pwd, nil, usnc3. (Analysis 2276)

Deduction • Output of AEAD_ENC(pwd3, nil, nil) obtained by reconstructing with pwd3, nil, nil. (Analysis 2276)

Deduction • Output of AEAD_ENC(pwd2, usnc2, usnc2) obtained by reconstructing with pwd2, usnc2, usnc2. (Analysis 2283)

Deduction • Output of AEAD_ENC(pwd, nil, usnp) obtained by reconstructing with pwd, nil, usnp. (Analysis 2286)

Deduction • Output of AEAD_ENC(pwd3, nil, tp2) obtained by reconstructing with pwd3, nil, tp2. (Analysis 2288)

Deduction • Output of AEAD_ENC(usnc3, usnp, tp2) obtained by reconstructing with usnc3, usnp, tp2. (Analysis 2288)

Deduction • Output of AEAD_ENC(pwd, usnc3, usnc1) obtained by reconstructing with pwd, usnc3, usnc1. (Analysis 2288)

Deduction • Output of AEAD_ENC(pwd2, pwd2, nil) obtained by reconstructing with pwd2, pwd2, nil. (Analysis 2288)

Deduction • Output of AEAD_ENC(usnp, usnc3, pwd) obtained by reconstructing with usnp, usnc3, pwd. (Analysis 2289)

Deduction • Output of AEAD_ENC(pwd3, tp2, usnc1) obtained by reconstructing with pwd3, tp2, usnc1. (Analysis 2290)

Deduction • Output of AEAD_ENC(usnc3, usnc2, pwd3) obtained by reconstructing with usnc3, usnc2, pwd3. (Analysis 2290)

Deduction • Output of AEAD_ENC(usnc3, usnc3, pwd) obtained by reconstructing with usnc3, usnc3, pwd. (Analysis 2299)

Deduction • Output of AEAD_ENC(usnc3, tp2, tp2) obtained by reconstructing with usnc3, tp2, tp2. (Analysis 2302)

Deduction • Output of AEAD_ENC(pwd, usnp, pwd1) obtained by reconstructing with pwd, usnp, pwd1. (Analysis 2307)

Deduction • Output of AEAD_ENC(pwd3, tp2, pwd1) obtained by reconstructing with pwd3, tp2, pwd1. (Analysis 2311)

Deduction • Output of AEAD_ENC(pwd2, usnc3, pwd3) obtained by reconstructing with pwd2, usnc3, pwd3. (Analysis 2311)

Deduction • Output of AEAD_ENC(usnc3, tp2, usnc2) obtained by reconstructing with usnc3, tp2, usnc2. (Analysis 2325)

Deduction • Output of AEAD_ENC(pwd3, tp2, nil) obtained by reconstructing with pwd3, tp2, nil. (Analysis 2328)

Deduction • Output of AEAD_ENC(usnc3, tp1, usnp) obtained by reconstructing with usnc3, tp1, usnp. (Analysis 2331)

Deduction • Output of AEAD_ENC(pwd3, tp2, pwd2) obtained by reconstructing with pwd3, tp2, pwd2. (Analysis 2335)

Deduction • Output of AEAD_ENC(usnc3, pwd1, usnc1) obtained by reconstructing with usnc3, pwd1, usnc1. (Analysis 2338)

Deduction • Output of AEAD_ENC(usnc3, usnc2, usnc2) obtained by reconstructing with usnc3, usnc2, usnc2. (Analysis 2340)

Deduction • Output of AEAD_ENC(pwd3, nil, pwd) obtained by reconstructing with pwd3, nil, pwd. (Analysis 2341)

Deduction • Output of AEAD_ENC(pwd, tp2, nil) obtained by reconstructing with pwd, tp2, nil. (Analysis 2311)

Deduction • Output of AEAD_ENC(usnc3, pwd1, usnc2) obtained by reconstructing with usnc3, pwd1, usnc2. (Analysis 2345)

Deduction • Output of AEAD_ENC(usnc3, usnc1, usnp) obtained by reconstructing with usnc3, usnc1, usnp. (Analysis 2352)

Deduction • Output of AEAD_ENC(pwd, tp1, pwd1) obtained by reconstructing with pwd, tp1, pwd1. (Analysis 2357)

Deduction • Output of AEAD_ENC(pwd, usnc3, pwd1) obtained by reconstructing with pwd, usnc3, pwd1. (Analysis 2363)

Deduction • Output of AEAD_ENC(pwd3, nil, pwd2) obtained by reconstructing with pwd3, nil, pwd2. (Analysis 2366)

Deduction • Output of AEAD_ENC(pwd, usnc2, usnc3) obtained by reconstructing with pwd, usnc2, usnc3. (Analysis 2369)

Deduction • Output of AEAD_ENC(usnc3, usnc2, pwd2) obtained by reconstructing with usnc3, usnc2, pwd2. (Analysis 2375)

Deduction • Output of AEAD_ENC(usnp, pwd3, nil) obtained by reconstructing with usnp, pwd3, nil. (Analysis 2375)

Deduction • Output of AEAD_ENC(pwd, pwd2, usnc1) obtained by reconstructing with pwd, pwd2, usnc1. (Analysis 2387)

Deduction • Output of AEAD_ENC(usnc3, pwd3, pwd2) obtained by reconstructing with usnc3, pwd3, pwd2. (Analysis 2388)

Deduction • Output of AEAD_ENC(usnc3, pwd3, pwd1) obtained by reconstructing with usnc3, pwd3, pwd1. (Analysis 2390)

Deduction • Output of AEAD_ENC(usnc3, usnc1, tp1) obtained by reconstructing with usnc3, usnc1, tp1. (Analysis 2392)

Deduction • Output of AEAD_ENC(pwd, nil, usnc2) obtained by reconstructing with pwd, nil, usnc2. (Analysis 2393)

Deduction • Output of AEAD_ENC(usnc3, usnc2, usnc3) obtained by reconstructing with usnc3, usnc2, usnc3. (Analysis 2397)

Deduction • Output of AEAD_ENC(pwd, usnc2, nil) obtained by reconstructing with pwd, usnc2, nil. (Analysis 2398)

Deduction • Output of AEAD_ENC(usnc3, usnc2, usnp) obtained by reconstructing with usnc3, usnc2, usnp. (Analysis 2398)

Deduction • Output of AEAD_ENC(pwd, usnc3, usnc2) obtained by reconstructing with pwd, usnc3, usnc2. (Analysis 2400)

Deduction • Output of AEAD_ENC(usnc3, usnp, pwd2) obtained by reconstructing with usnc3, usnp, pwd2. (Analysis 2401)

Deduction • Output of AEAD_ENC(pwd, tp1, usnc1) obtained by reconstructing with pwd, tp1, usnc1. (Analysis 2404)

Deduction • Output of AEAD_ENC(pwd3, tp1, pwd) obtained by reconstructing with pwd3, tp1, pwd. (Analysis 2404)

Deduction • Output of AEAD_ENC(pwd2, usnc3, pwd) obtained by reconstructing with pwd2, usnc3, pwd. (Analysis 2404)

Deduction • Output of AEAD_ENC(pwd3, tp1, usnc2) obtained by reconstructing with pwd3, tp1, usnc2. (Analysis 2404)

Deduction • Output of AEAD_ENC(pwd, usnc2, pwd) obtained by reconstructing with pwd, usnc2, pwd. (Analysis 2406)

Deduction • Output of AEAD_ENC(usnc3, pwd3, usnc1) obtained by reconstructing with usnc3, pwd3, usnc1. (Analysis 2409)

Deduction • Output of AEAD_ENC(pwd, nil, pwd1) obtained by reconstructing with pwd, nil, pwd1. (Analysis 2409)

Deduction • Output of AEAD_ENC(pwd, pwd, pwd1) obtained by reconstructing with pwd, pwd, pwd1. (Analysis 2411)

Deduction • Output of AEAD_ENC(pwd, usnp, usnp) obtained by reconstructing with pwd, usnp, usnp. (Analysis 2411)

Deduction • Output of AEAD_ENC(usnc3, pwd1, usnc3) obtained by reconstructing with usnc3, pwd1, usnc3. (Analysis 2414)

Deduction • Output of AEAD_ENC(usnc3, pwd3, usnc3) obtained by reconstructing with usnc3, pwd3, usnc3. (Analysis 2416)

Deduction • Output of AEAD_ENC(pwd3, pwd3, usnc3) obtained by reconstructing with pwd3, pwd3, usnc3. (Analysis 2418)

Deduction • Output of AEAD_ENC(pwd3, nil, usnc1) obtained by reconstructing with pwd3, nil, usnc1. (Analysis 2477)

Deduction • Output of AEAD_ENC(pwd3, usnc2, usnc3) obtained by reconstructing with pwd3, usnc2, usnc3. (Analysis 2481)

Deduction • Output of AEAD_ENC(pwd, pwd1, tp2) obtained by reconstructing with pwd, pwd1, tp2. (Analysis 2495)

Deduction • Output of AEAD_ENC(pwd3, tp2, tp2) obtained by reconstructing with pwd3, tp2, tp2. (Analysis 2509)

Deduction • Output of AEAD_ENC(pwd3, nil, pwd1) obtained by reconstructing with pwd3, nil, pwd1. (Analysis 2515)

Deduction • Output of AEAD_ENC(usnp, tp2, pwd3) obtained by reconstructing with usnp, tp2, pwd3. (Analysis 2518)

Deduction • Output of AEAD_ENC(pwd, pwd2, usnc2) obtained by reconstructing with pwd, pwd2, usnc2. (Analysis 2525)

Deduction • Output of AEAD_ENC(usnp, pwd2, nil) obtained by reconstructing with usnp, pwd2, nil. (Analysis 2531)

Deduction • Output of AEAD_ENC(usnp, pwd, pwd1) obtained by reconstructing with usnp, pwd, pwd1. (Analysis 2532)

Deduction • Output of AEAD_ENC(pwd, tp2, pwd1) obtained by reconstructing with pwd, tp2, pwd1. (Analysis 2544)

Deduction • Output of AEAD_ENC(usnp, usnc3, tp1) obtained by reconstructing with usnp, usnc3, tp1. (Analysis 2548)

Deduction • Output of AEAD_ENC(usnp, tp1, pwd1) obtained by reconstructing with usnp, tp1, pwd1. (Analysis 2559)

Deduction • Output of AEAD_ENC(usnp, usnc3, usnc2) obtained by reconstructing with usnp, usnc3, usnc2. (Analysis 2571)

Deduction • Output of AEAD_ENC(pwd, tp1, tp2) obtained by reconstructing with pwd, tp1, tp2. (Analysis 2575)

Deduction • Output of AEAD_ENC(pwd, usnc1, usnc3) obtained by reconstructing with pwd, usnc1, usnc3. (Analysis 2587)

Deduction • Output of AEAD_ENC(pwd, pwd3, pwd1) obtained by reconstructing with pwd, pwd3, pwd1. (Analysis 2587)

Deduction • Output of AEAD_ENC(usnp, usnp, tp2) obtained by reconstructing with usnp, usnp, tp2. (Analysis 2590)

Deduction • Output of AEAD_ENC(usnp, pwd, tp2) obtained by reconstructing with usnp, pwd, tp2. (Analysis 2599)

Deduction • Output of AEAD_ENC(pwd, pwd, usnc1) obtained by reconstructing with pwd, pwd, usnc1. (Analysis 2604)

Deduction • Output of AEAD_ENC(pwd3, pwd, pwd) obtained by reconstructing with pwd3, pwd, pwd. (Analysis 2614)

Deduction • Output of AEAD_ENC(pwd, pwd, tp2) obtained by reconstructing with pwd, pwd, tp2. (Analysis 2614)

Deduction • Output of AEAD_ENC(pwd, pwd3, pwd2) obtained by reconstructing with pwd, pwd3, pwd2. (Analysis 2617)

Deduction • Output of AEAD_ENC(usnp, tp1, tp2) obtained by reconstructing with usnp, tp1, tp2. (Analysis 2618)

Deduction • Output of AEAD_ENC(pwd3, usnc2, usnc1) obtained by reconstructing with pwd3, usnc2, usnc1. (Analysis 2623)

Deduction • Output of AEAD_ENC(pwd3, nil, tp1) obtained by reconstructing with pwd3, nil, tp1. (Analysis 2626)

Deduction • Output of AEAD_ENC(pwd, pwd2, pwd1) obtained by reconstructing with pwd, pwd2, pwd1. (Analysis 2626)

Deduction • Output of AEAD_ENC(pwd, nil, pwd2) obtained by reconstructing with pwd, nil, pwd2. (Analysis 2627)

Deduction • Output of AEAD_ENC(pwd, pwd3, tp2) obtained by reconstructing with pwd, pwd3, tp2. (Analysis 2628)

Deduction • Output of AEAD_ENC(usnc3, usnc3, tp1) obtained by reconstructing with usnc3, usnc3, tp1. (Analysis 2631)

Deduction • Output of AEAD_ENC(usnc3, usnp, usnc1) obtained by reconstructing with usnc3, usnp, usnc1. (Analysis 2631)

Deduction • Output of AEAD_ENC(pwd, nil, pwd3) obtained by reconstructing with pwd, nil, pwd3. (Analysis 2631)

Deduction • Output of AEAD_ENC(pwd, usnp, usnc1) obtained by reconstructing with pwd, usnp, usnc1. (Analysis 2631)

Deduction • Output of AEAD_ENC(pwd, usnp, pwd) obtained by reconstructing with pwd, usnp, pwd. (Analysis 2631)

Deduction • Output of AEAD_ENC(pwd, pwd, usnc3) obtained by reconstructing with pwd, pwd, usnc3. (Analysis 2632)

Deduction • Output of AEAD_ENC(pwd3, pwd, pwd3) obtained by reconstructing with pwd3, pwd, pwd3. (Analysis 2633)

Deduction • Output of AEAD_ENC(pwd3, tp1, pwd2) obtained by reconstructing with pwd3, tp1, pwd2. (Analysis 2633)

Deduction • Output of AEAD_ENC(pwd3, usnc2, pwd1) obtained by reconstructing with pwd3, usnc2, pwd1. (Analysis 2634)

Deduction • Output of AEAD_ENC(usnc3, nil, usnc2) obtained by reconstructing with usnc3, nil, usnc2. (Analysis 2634)

Deduction • Output of AEAD_ENC(usnc3, pwd, tp1) obtained by reconstructing with usnc3, pwd, tp1. (Analysis 2651)

Deduction • Output of AEAD_ENC(usnc3, tp1, pwd3) obtained by reconstructing with usnc3, tp1, pwd3. (Analysis 2659)

Deduction • Output of AEAD_ENC(pwd, pwd3, pwd) obtained by reconstructing with pwd, pwd3, pwd. (Analysis 2664)

Deduction • Output of AEAD_ENC(usnp, tp1, usnc1) obtained by reconstructing with usnp, tp1, usnc1. (Analysis 2668)

Deduction • Output of AEAD_ENC(pwd3, usnc1, usnp) obtained by reconstructing with pwd3, usnc1, usnp. (Analysis 2670)

Deduction • Output of AEAD_ENC(pwd, pwd3, tp1) obtained by reconstructing with pwd, pwd3, tp1. (Analysis 2670)

Deduction • Output of AEAD_ENC(pwd, tp1, pwd3) obtained by reconstructing with pwd, tp1, pwd3. (Analysis 2670)

Deduction • Output of AEAD_ENC(pwd, pwd, usnc2) obtained by reconstructing with pwd, pwd, usnc2. (Analysis 2677)

Deduction • Output of AEAD_ENC(pwd, pwd2, tp2) obtained by reconstructing with pwd, pwd2, tp2. (Analysis 2681)

Deduction • Output of AEAD_ENC(pwd, pwd3, usnc2) obtained by reconstructing with pwd, pwd3, usnc2. (Analysis 2687)

Deduction • Output of AEAD_ENC(usnc3, pwd3, usnp) obtained by reconstructing with usnc3, pwd3, usnp. (Analysis 2700)

Deduction • Output of AEAD_ENC(pwd, tp2, usnc1) obtained by reconstructing with pwd, tp2, usnc1. (Analysis 2701)

Deduction • Output of AEAD_ENC(pwd, pwd3, usnc1) obtained by reconstructing with pwd, pwd3, usnc1. (Analysis 2705)

Deduction • Output of AEAD_ENC(pwd, usnc2, usnp) obtained by reconstructing with pwd, usnc2, usnp. (Analysis 2705)

Deduction • Output of AEAD_ENC(pwd, tp1, pwd) obtained by reconstructing with pwd, tp1, pwd. (Analysis 2654)

Deduction • Output of AEAD_ENC(pwd3, tp2, tp1) obtained by reconstructing with pwd3, tp2, tp1. (Analysis 2718)

Deduction • Output of AEAD_ENC(usnc3, tp2, pwd2) obtained by reconstructing with usnc3, tp2, pwd2. (Analysis 2732)

Deduction • Output of AEAD_ENC(pwd, usnc3, pwd3) obtained by reconstructing with pwd, usnc3, pwd3. (Analysis 2739)

Deduction • Output of AEAD_ENC(pwd, usnp, tp2) obtained by reconstructing with pwd, usnp, tp2. (Analysis 2747)

Deduction • Output of AEAD_ENC(pwd, pwd, tp1) obtained by reconstructing with pwd, pwd, tp1. (Analysis 2747)

Deduction • Output of AEAD_ENC(usnc3, pwd2, pwd3) obtained by reconstructing with usnc3, pwd2, pwd3. (Analysis 2757)

Deduction • Output of AEAD_ENC(usnc3, pwd1, pwd1) obtained by reconstructing with usnc3, pwd1, pwd1. (Analysis 2779)

Deduction • Output of AEAD_ENC(pwd, usnc3, pwd2) obtained by reconstructing with pwd, usnc3, pwd2. (Analysis 2785)

Deduction • Output of AEAD_ENC(usnc3, pwd, pwd) obtained by reconstructing with usnc3, pwd, pwd. (Analysis 2793)

Analysis • Initializing Stage 2 mutation map for Client2...

Analysis • Initializing Stage 3 mutation map for Client2...

Analysis • Initializing Stage 2 mutation map for Client3...

Analysis • Initializing Stage 3 mutation map for Client3...

Analysis • Initializing Stage 5 mutation map for Broker...

Analysis • Initializing Stage 4 mutation map for Broker...

Analysis • Initializing Stage 4 mutation map for Publisher...

Analysis • Initializing Stage 4 mutation map for Client...

Analysis • Initializing Stage 5 mutation map for Publisher...

Analysis • Initializing Stage 5 mutation map for Client...

Analysis • Initializing Stage 5 mutation map for Client2...

Analysis • Initializing Stage 4 mutation map for Client2...

Analysis • Initializing Stage 4 mutation map for Client3...

Analysis • Initializing Stage 5 mutation map for Client3...

Analysis • Initializing Stage 6 mutation map for Broker...

Analysis • Initializing Stage 6 mutation map for Publisher...

Analysis • Initializing Stage 6 mutation map for Client...

Analysis • Initializing Stage 6 mutation map for Client2...

Analysis • Initializing Stage 6 mutation map for Client3...

Verifpal • Verification completed for 'MQTT_Model.vp' at 12:47:18 PM.

Verifpal • Summary of failed queries will follow.

Result • authentication? Publisher -> Broker: cnt1 — When:

cnt1 → CONCAT(nil, tp1) ← mutated by Attacker (originally CONCAT(usnp, pwd))

usnp_ → nil ← obtained by Attacker

cnt1 (CONCAT(nil, tp1)), sent by Attacker and not by Publisher, is successfully used in SPLIT(CONCAT(nil, tp1)) within Broker's state.

Result • confidentiality? cnt1 —

cnt1 (CONCAT(usnp, pwd)) is obtained by Attacker.

Result • equivalence? pwd, pwd_ — When:

cnt1 → tp1 ← mutated by Attacker (originally CONCAT(usnp, pwd))

usnp_ → SPLIT(tp1)

pwd, pwd_ are not equivalent.

Result • confidentiality? pwd — When:

usnp_ → usnp

pwd_ → pwd ← obtained by Attacker

h1 → HASH(pwd, salt1)

unnamed_0 → ASSERT(HASH(pwd, salt1), HASH(pwd, salt1))?

m_de → aa

m_de2 → aa

pwd (pwd) is obtained by Attacker.

Result • confidentiality? m —

m (AEAD_ENC(psk1, aa, tp2)) is obtained by Attacker.

Verifpal • Thank you for using Verifpal.