Providing Identity Privacy in 5G Networks by Using Pseudonyms

UNIVERSITY OF TURKU

Department of Mathematics and Statistics

Master of Science Thesis

Cryptography and Data Security

May 2018

Gizem Akman

Supervisors:

Prof. Valtteri Niemi

Dr. Ville Junnila

UNIVERSITY OF TURKU
Department of Mathematics and Statistics

AKMAN GIZEM: Providing Identity Privacy in 5G Networks by Using Pseudonyms

Master of Science Thesis, 65 p., 39 app. p.
Information Security and Cryptography - Cryptography and Data Security
May 2018

The originality of this thesis has been checked in accordance with the University of Turku quality assurance system using the Turnitin Originality Check service

---

This thesis aims for presenting a solution for providing the identity privacy in mobile networks. The user is identified in mobile networks by an International Mobile Subscriber Identity (IMSI). An IMSI catcher is a device that acts like a fake base station and targets information such as identity and location. Location tracking is one of the most serious outcomes, in case attacker captures these details. Since building an IMSI catcher is now cheaper than before and detecting one is very hard, threat caused by this device has become a serious issue, especially while developing 5G.

Several solutions to protect against IMSI catchers are explained in this thesis, and one solution for defeating IMSI catchers is using pseudonyms instead of real identity. We claim that pseudonym can be an effective solution for providing identity privacy in 5G networks and can be also compatible with legacy networks. We have implemented a prototype that demonstrates how pseudonym can be imposed to an existing Authentication and Key Agreement (AKA) procedure. This prototype has been presented in two public demonstration sessions.

This thesis includes the history of the mobile networks including 5G. The changes between generations of networks show the requirements for better infrastructure, and also for improved security. We have also examined the development of AKA, since AKA is one of the most important procedures to provide secure service to valid users. Moreover, our prototype is about enhancing AKA for adapting pseudonym approach.

This thesis also mentions about a block cipher called KASUMI, which is used for encrypting and decrypting pseudonym during AKA in the prototype. Since KASUMI is designed specifically for 3GPP and cryptanalyses show it is still safe to use KASUMI, it was chosen to be used in the prototype.

Keywords: 5G, mobile networks, pseudonym, identity privacy, authentication and key agreement, KASUMI.

Acknowledgements

I want to thank to my supervisors, Prof. Valtteri Niemi and Dr. Ville Junnila, for their support throughout my studies and thesis.

I specially express my gratitude to Valtteri Niemi for accepting me for internship in my first year and making me learn about this topic. I am happy that I had a chance to study with such a valuable supervisor and a mentor. I sincerely thank him for being supportive and patient. I hope to continue working with him and learn more from him.

Finally, I want to thank to my mother, my father, and my sister for always being there for me. I would not succeed any of this without their support. They have been invaluable in my life.

Abbreviations

3GPP - *3rd Generation Partnership Project*

5G HE AV - *5G Home Environment Authentication Vector*

5G-AIR - *5G Authentication Initiation Request*

5G-GUTI - *5G Globally Unique Temporary Identifier*

AES - *Advanced Encryption Standard*

AIR - *Authentication Information Request*

AK - *Anonymity Key*

AKA - *Authentication and Key Agreement*

AMF - *Authentication Management Field*

also: - *Core Access and Management Function*

ARPF - *Authentication Credential Repository and Processing Function*

AS - *Access Stratum*

AuC - *Authentication Center*

AUSF - *Authentication Server Function*

AUTN - *Authentication Token*

AV - *Authentication Vector*

BTS - *Base Transceiver Station*

CA - *Certificate authority*

CK - *Cipher Key*

DN - *Data Network*

DoS - *Denial of Service*

ECC - *Elliptic Curve Cryptography*

EDGE - *Enhanced Data rates in GSM Environment*

eNodeB - *Evolved NodeB*

EPS - *Evolved Packet System*

E-UTRAN - *Evolved-Universal Terrestrial Radio Access Network*

GPRS - *General Packet Radio Services*

GSM - *Global System for Mobile Communications*

GUAMI - *Globally Unique AMF ID*

GUMMEI - *Globally Unique MME Identifier*

GUTI - *Globally Unique Temporary UE Identity*

HLR - *Home Location Register*

HN - *Home Network*

HSPA - *High Speed Packet Access*

HSS - *Home Subscriber Server*

IK - *Integrity Key*

IMSI - *International Mobile Subscriber Identity*

IMT - *International Mobile Communications*

KDF - *Key Derivation Functions*

LTE - *Long Term Evolution*

MAC - *Message Authentication Code*

MCC - *Mobile Country Code*

ME - *Mobile Equipment*

MME - *Mobility Management Entity*

MNC - *Mobile Network Code*

MSC - *Mobile Switching Center*

MSIN - *Mobile Subscriber Identification Number*

NAI - *Network Access Identifier*

OP - *Operator Variant Algorithm Configuration Field*

PIN - *Personal Identification Number*

PKI - *Public Key Infrastructure*

QoS - *Quality of Service*

RAN - *Radio Access Network*

SCMF - *Security Context Management Function*

SEAF - *Security Anchor Function*

S-GW - *Serving Gateway*

SIM - *Subscriber Identity Module*

SMF - *Session Management Function*

SMS - *Short Message System*

SN - *Serving Network*

SPCF - *Security Policy Control Function*

SQN - *Sequence Number*

SUCI - *Subscription Concealed Identifier*

SUPI - *Subscription Permanent Identifier*

TMSI - *Temporary Mobile Subscriber Identity*

UDM - *Unified Data Management*

UE - *User Equipment*

UPF – *User Plane Function*

USIM - *User Subscriber Identity Module*

VLR - *Visitor Location Register*

Wi-Fi - *Wireless Fidelity*

XRES - *Expected Response*

List of Figures

TABLE OF CONTENTS

Introduction

Throughout the history, mankind has been required to communicate with each other. As time passed, social conditions have evolved, and communication methods have changed from body language to speech, then to written materials. Along with many devices that were used in history, telephone was invented in 1876 [1]. The sound was transmitted across a wire from one telephone to another. In the beginning of 1900s, radio was invented and became popular in a short span of time [1]. Finally, in the end of 1970s, cell phone, which can be considered as composition of telephone and radio, came to existence. With this invention, the history of mobile networks begins and keeps growing continually.

Over the years, cell phones and mobile networks developed along with the improvement of technology. In 1990, the number of mobile subscribers was counted to be 11 million worldwide [2]. This number increased to 300 million by the end of 1998 and was expected to reach 500 million before 2000 [2]. The rapid growth in mobile networks never stopped and is still increasing. Figure 1 displays the number

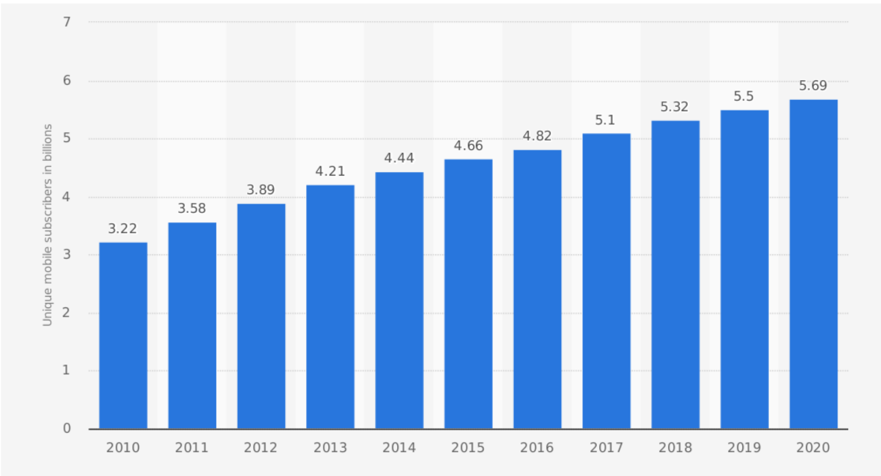


Figure 1: Number of unique mobile subscribers worldwide from 2010 to 2020 (in billions) [3]

of the mobile subscribers in the world between the years 2010 and 2020 [3]. Comparing the estimation in 2000 and the number of 2010 in the Figure 1, the number of subscribers increased for 3 billion. Numbers for the years after 2015 are the estimations of GSMA, made in 2015. So far, estimations for 2017 came true and the number of mobile subscribers reached 5 billion. According to GSMA, "the 5 billion milestone means that more than two-thirds of the global population is now connected to a mobile service" [4]. Gradually technology has become relatively cheaper and significantly more accessible, so it made and will make more people to benefit from this opportunity.

On the other hand, ever since the mankind managed to communicate, people tried to intercept communication of others. Especially in history, messages that are related to military issues were worth protecting. Therefore, cryptography was conceived more than 4000 years ago [5]. Cryptography can be defined as "the science or study of the techniques of secret writing, especially code and cipher systems" [6]. Invention of radio helped the improvement of cryptography, but it was still in use of military. Then, cell phones were invented, and mobile networks started to evolve. After 2G was introduced, digital communication era, which made encryption and decryption possible, started.

Next, we take a closer look at how cryptography is applied in mobile networks. Cryptography is first involved during the *Authentication and Key Agreement* (AKA) phase between home network and the subscriber in mobile networks. In this way, trusted subscriber can get service from a trusted network. However, attackers may aim for the beginning of authentication. Subscribers need to provide their identifiers in order to start authentication with the home network and attackers target for these identifiers. This attack can be performed through IMSI catchers, which are fake base stations and are explained in detail in Chapter 4. After the attacker gets the identifier of the subscriber, then attacker can track the location of the subscriber as well. Location tracking is only one of the consequences the IMSI catcher creates but is a great threat against identity privacy. Every person has right to have the identity privacy.

In this thesis, we discuss a method for avoiding the threat against identity privacy. Since cryptography cannot be used during the identification process, this method is necessary for protecting the identity privacy. This method, explained in the thesis, is using pseudonyms that only home network can relate to real identifiers. Moreover, in the thesis, a prototype is implemented to demonstrate one way of using pseudonyms during AKA procedure.

This thesis starts with the background information of mobile networks and continues with the recent developments along with an implemented prototype, which demonstrates one way of improving identity privacy in mobile networks. Chapter 1 explains identifiers and process of identification in mobile networks. Then, brief history of mobile networks is presented in Chapter 2. It becomes easy to see the progress between different generations, by the help of this chapter. Chapter 3 explains security issues in mobile networks. In this chapter, 4G network owns greater margin, because 4G is the latest network in use and improving the controversial circumstances in 4G would provide better service for 5G. The elements of mobile networks are explained in Chapter 4. Each generation is an improved version of the one before. Therefore, 4G network is explained briefly in this chapter. Chapter 5 shows the alteration and development of Authentication and Key Agreement procedures in all networks since 2G. In order to provide security and privacy, AKA has important place in mobile networks. Therefore, AKA needs to be improved and optimized for 5G network. Chapter 6 gives the details about the KASUMI cryptosystem. KASUMI is one of the block ciphers that can be used during AKA. Moreover, KASUMI is preferred in the prototype for encrypting the pseudonyms. In Chapter 7, developments in 5G, which are already accepted by 3GPP, are presented. Then, Chapter 8 displays the comparison of two methods for ensuring identity privacy in 5G. Chapter 9 includes the details of the implemented prototype for demonstrating pseudonym approach in 5G to provide identity privacy. The prototype is written in Java and the source codes can be found in Appendix A. Appendix B presents the output after the prototype is executed. Appendix C includes some screenshots from the demonstration. Finally, the demonstration is presented in demo sessions of two conferences and the details of the public demonstrations are given in Appendix D.

1. Identification in Mobile Networks

For mobile networks, identification of a user is an important process. With the help of identification, mobile networks provide proper service to the right user. Therefore, in order to understand this process, it is important to clarify some concepts. A subscriber is the person who registered for the *Subscriber Identity Module* (SIM). User can be anyone else who is given access to the phone. Hence, subscriber and user are not necessarily the same person. However, in this thesis, we simplify handling by not making a difference between subscriber and user. So, the term user refers to subscriber as well.

Subscriber Identity Module is a smart card that stores the credentials and necessary information of subscriber. However, the name of SIM changed into *Universal Subscriber Identity Module* (USIM), after 3G is established. The USIM is inserted in mobile devices, for example smartphones, and contributes in authentication and key agreement as well.

International Mobile Subscriber Identity (IMSI) is permanent identity number with a unique 15-digit number that corresponds to a USIM. The IMSI is composed of three parts, such that $IMSI = MCC \parallel MNC \parallel MSIN$. *Mobile Country Code*, MCC, has 3 digits that specifically identifies the home country of the USIM. Moreover, MNC, *Mobile Network Code*, is 2-digits and describes the home network, in other words, operator. Finally, rest of the 10 digits form MSIN, *Mobile Subscriber Identification Number*, which is the specific number that is assigned to the subscriber [7]. For example, 244 is MCC code for Finland and 12 is MNC code for DNA Oy [8], so 244121234567890 would be the IMSI, where 1234567890 is MSIN.

Each subscriber is assigned to a phone number as well as IMSI. The IMSI and phone number have almost similar structure. Both start with country code and operator code and continue with some amount of unique numbers. Next, we discuss differences between IMSI and the phone number. First of all, IMSI is permanent for the specific SIM card, it is not possible for user to change IMSI without changing the SIM card. On the other hand, phone number is assigned to SIM and IMSI by the operator. The phone number is used, e.g. by others to point to this particular user and call him/her. The phone number is included in the phone catalogues etc. and also is used for routing calls to right network. In practice, it is possible to change phone number without changing the SIM card. Moreover, it is also possible to change SIM card and IMSI, but to keep phone number same [9]. Another point relevant from the privacy point of view is that user knows the phone number and shares this number with necessary people, whereas IMSI is only known by the operator and the system behind the network. Therefore, it is harder for anyone to associate a phone number to corresponding IMSI number.

Temporary Mobile Subscriber Identity (TMSI) is temporary identity number, the shorter replacement of IMSI. The local operator assigns the TMSI for each IMSI that has arrived at their network. The local operator also sends the TMSI to the subscriber over encrypted channel. The main differences between two identities are that IMSI is global and permanent, whereas TMSI is local and temporary. The IMSI has to be unique all over the world. It follows that, two different SIM cards cannot have same IMSI. However, same TMSI can be used by different operators, even in the same country. Since different operators have different radio frequencies, potentially identical TMSIs from two different operators would not intercept each other.

In order to understand the functionality of MCC and MNC, let us assume that a user A has subscription from a Finnish operator and travels to another country, for example Turkey. User A tries to connect to a local operator. There should be an agreement between the local operator and the home operator of the user, which is called roaming agreement [10]. When the visited operator receives the IMSI number, then it immediately understands that the home operator is in Finland and informs the home operator that A is now in Turkey.

Figure 2 displays the relations between two users, home operators and visited operators of the users. In this figure, it is assumed that both users have subscriptions from operators in different countries and both users are visiting other countries. In other words, all visited operators and home operators are in different countries. User A connects to Visited Operator A', because Home Operator of User A has a roaming agreement with the Visited Operator A'. Likewise, User B connects to Visited

Operator B', because Home Operator of User B has a roaming agreement with the Visited Operator B'. In the Figure 2, Phone Number of User A and Phone Number of User B are abbreviated to respectively PN_A and PN_B.

Figure 2 also shows the procedure with dashed lines, when User A uses PN_B to initiate a call for User B. The process is explained in detail as:

1- User A sends a message containing the PN_B to the Visited Operator A'.

2- Visited Operator A' reaches to the Home Operator of User B by using the country and operator code in the phone number. Visited Operator A' also includes PN_A to inform who is trying to call User B.

3- Home Operator of User B knows that User B is in different country and connected to Visited Operator B'. Therefore, Home Operator of User B informs Visited Operator B' about the call by sending the IMSI of User B along with the PN_A.

4- When User B and Visited Operator B' connected, Visited Operator B' assigned TMSI for User B. So, Visited Operator B' sends the call request by sending TMSI of User B along with the PN_A.

User A and User B start talking after User B accept the call request from Visited Operator B'.

5- After the call ends, Visited Operator A' sends the charging information to the Home Operator of User A.

6- After the call ends, Visited Operator B' sends the charging information to the Home Operator of User B.

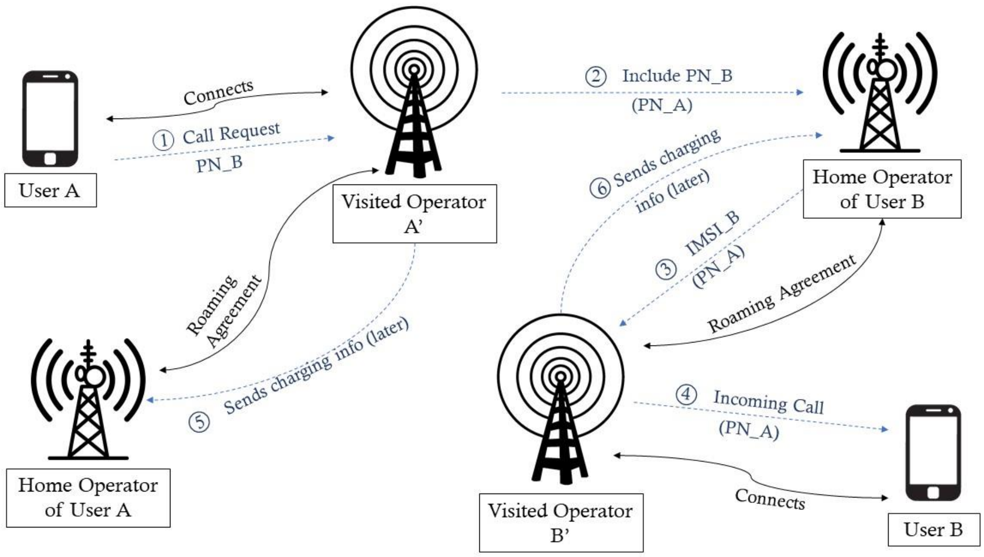


Figure 2: Relations of elements during international call from User A to User B.

Figure 2 displays the general case, in which each element is in different country. Many other special cases can also be derived from the Figure 2. For example, if the User B is not in different country, then Visited Operator B' would be same as Home Operator of User B. Therefore there would not be a roaming agreement, Step 3 would integrate with Step 4, and Step 6 would not exist. Another example can be given for User A being in the same country of the Home Operator of User B, whereas Visited Operator A' and Home Operator of User B would be the same. In this case, Step 1 unites with Step 2 and Home Operator of User B charges Home Operator of User A. There can be more examples for the special cases.

2. History of Mobile Networks

The history of mobile networks can be seen as an evolution story. The difference between the technologies of first generation and what we have today is massive. Ever since the first utilization of mobile services, it became so popular. When it is assumed that enough people are willing to pay for better services, the existing services are needed to be advanced. If the service level doesn't increase after some upgrades, then there is a necessity for changing the whole technology. When in fact the whole technology changes, then the security can also be improved and adapted to the new technology. On the way to enhance 5G networks, it is essential to understand the progress and weaknesses of prior mobile networks.

2.1. First Generation (1G)

First Generation was introduced in the beginning of 1980s [11] and 1G used analog techniques for speech services [13]. There were many complications in this system. First, establishing communication was not possible between the countries [13], which was not convenient. Then, capacity and service, provided by 1G, could not suffice the need of people. Moreover, security of 1G was falling short, since "voice calls were stored and played in radio towers" [12] and this situation gave opportunity for eavesdroppers.

2.2. Second Generation (2G)

Second Generation was introduced at the beginning of 1990s. Unlike 1G, 2G uses digital techniques, which means it was possible to start using cryptography for providing better security. In addition, 2G provided higher efficiency and improved data services [13]. *Global System for Mobile Communications* (GSM) was the first system of 2G, which helped to standardize the properties. GSM was used for speech services, *Short Message System* (SMS), and data rate up to 64 kbps [12]. Furthermore, GSM "enabled seamless services throughout Europe by means of international roaming" [13] and helped 2G to have precedence over 1G.

Thereafter, a new system, called *General Packet Radio Services* (GPRS), was developed for 2G, which was also known as 2.5G. The main idea behind GPRS was connecting to internet. Therefore, even though 2.5G had many properties same as 2G, GPRS had packet switching as an extra protocol. This new protocol speeded up the connection time by sending and receiving IP packets, so that data rate could go up to 144 kbps [12,13]. Along with the need of increasing the data rate more, *Enhanced Data rates in GSM Environment* (EDGE) was developed. Development of EDGE raised the data rate up to 384 kbps [13].

2.3. Third Generation (3G)

Towards the end of 1990s, around the same time when EDGE was founded, 3G was being developed. Moreover, throughout the world, there were various kinds of standards for developing network. Therefore, a decision "to have a network which provides services independent of the technology platform and whose network design standards are same globally" [13] was made. Thus, every country around the world would work collaboratively. For this aim, an organization with name *3rd Generation Partnership Project* (3GPP) was founded.

Thereupon, 3G extended the transmission rate to 2 Mbps with the opportunity of global roaming [12]. With 3G, voice quality was improved. In addition, several features were adopted in 3G, such as video calls and broadband wireless data [13]. Improvements did not end with 3G, new features were added to existing system. *High Speed Packet Access* (HSPA) and some other developments kept the data rate around 5-30 Mbps [12]. These new features built a bridge between 3G and 4G, which is why inclusion of HSPA was also called 3.5G.

2.4. Fourth Generation (4G)

Long Term Evolution (LTE) was the successor of 3G, designed by 3GPP [14]. One of the important outcomes of LTE was that LTE had only packet switching, not voice call. Therefore, LTE provided "better coverage with improved performance for less cost" [12]. This was indeed the aim since the very beginning of mobile networks, and yet LTE made it accessible.

After some number of upgrades of LTE, LTE-Advanced was meeting the requirements for 4G, which were determined by ITU [15]. Beside the escalated data rate, framework of 4G embodied differences compared to 3G. The object of this new framework is "to accomplish new levels of user experience and multi-service capacity by also integrating all the mobile technologies that exist such as GSM, GPRS, IMT-2000, Wi-Fi, and Bluetooth" [13]. Here IMT stands for *International Mobile Communications* and Wi-Fi stands for *Wireless Fidelity*. This unity of the services would make it easier to reach higher data rates with less expenses. Moreover, 4G is still developed and will be until 5G completely settles.

2.5. Fifth Generation (5G)

By the time of late 2017 and early 2018, 5G is not in use and still under construction. Developers have great expectations on 5G. 3GPP and ITU are planning to release the specification of 5G towards the end of 2019. However, by some commercial means, release date can be moved to earlier time, such as 2018 [16]. On the other hand, there are already test trials that are been conducted. For example, one of the trial was completed by Samsung and SK Telecom in Suwon, South Korea, in June of 2017. They have achieved "speeds over 1 Gbps and low latency of 1.2 millisecond" [17]. These are promising results, since 5G aims for higher data rate and lower end-to-end latency. Furthermore, faster broadband, higher capacity, higher responsive connectivity, and reduced cost are also goals of 5G [12, 16].

3. Security Issues in Mobile Networks

Early generations of mobile networks had serious security vulnerabilities. As stated in Chapter 2.1, First Generation was not only open to eavesdropping, but it was also possible to intercept the information and clone the mobile phones. In fact, 2G started using *Authentication and Key Agreement* (AKA), which was achieved by challenge and response technique and increased its security level comparing to 1G. However, 2G stayed secure only one-way, because *User Equipment* (UE) could not authenticate the *Serving Network* (SN), while SN could authenticate UE. Therefore, 2G was still vulnerable to false network attacks, in other words, fake networks that pretends to be real. Some of the false network attacks are eavesdropping, identity spoofing, man-in-the-middle. With 3G, AKA was changed into mutual AKA, where both UE and SN can authenticate each other. In addition, sequence number was introduced to make sure that *Home Network* (HN) and UE were synchronized, so that an attacker cannot try to attempt connecting with former information of UE. This solution was also risky, because with a possible *Denial of Service* (DoS) attack, the synchronization might be lost and disturb the connection [18]. These and some other vulnerabilities obliged developers to solve all the problems.

3.1. Security Issues in 4G

Expectations from 4G were comparatively high. Other than higher data rates with less cost, it should be unobstructed under attacks or meet *Quality of Service* (QoS) standards without a problem [18]. On the other hand, 3GPP required many security objectives for 4G. The main purpose of objectives is providing a secure channel for network elements to communicate with each other without any obstruction.

However, vulnerabilities in 4G were remarked either soon after launching it or were already known. Bikos and Sklavos listed some of the threats [20], one of the threats is against user identity and privacy. In this case, the attacker gains access to the UE, uses the services by his own purposes, and manipulates the identity information so that the real user becomes locked out of its own UE. If the attacker does not confiscate the UE, he can obtain the identity details such as IMSI. From the connection between IP address and IMSI, location tracking of the user can be an issue, which is a significant problem for privacy. Another threat is against SN. The attacks to SN can be done both physically and remotely [20]. UEs tend to connect to any base station around them with higher signaling frequency. Under these circumstances, UE would connect to compromised but stronger base station, thereby hand over its identity and security to attacker.

Denial of Service (DoS) attacks may create serious problems for both UE and SN. There are at least three types of DoS attacks. The first one aims UE, where the attacker sends a signal to UE with the name of SN. This may cause SN to become confused and UE to lose the service. Another type of DoS attack arises because of a feature of UE, gained with 4G, which is "in LTE, the UE is allowed to stay in active mode, but turn off its radio transceiver to save power consumption. During discontinuous reception period, the UE is still allowed to transmit packets because the UE may have urgent traffic to send" [21]. Hence, the attacker can trigger UE to send packets to the other UEs and cause a DoS attack. The third type imitates the real UE and sends fake buffer reports to SN. Consequently, SN assumes that it deals with enough amount of workload and rejects the connection requests of any new UEs [21]. There are many other threats that are not mentioned here, but they all have different methods with similar aims: defrauding the property, security, and privacy of the users.

3.2. IMSI Catchers

In 4G, UE sends its identity details, in other words IMSI, to SN via unencrypted channel. Exposing IMSI provides opportunity for eavesdropping and man-in-the-middle attacks [19], which would cause the attacker to capture IMSI of the user. This could create a threat against the user, because "IMSI is used by the mobile network to identify and locate subscribers to connect incoming calls and more" [22]. Therefore, captured IMSIs are great menace against identity privacy and may create a danger for location tracking.

The IMSI is valuable information for the attackers, therefore an attack device called 'IMSI catcher' has been developed already against 2G. IMSI catcher is the general name for a device that is used for eavesdropping and location tracking [22]. These devices aim to catch the IMSI from the wireless traffic between UE and SN [23]. Moreover, if there are more than one SN around the UE, UE tends to connect to the one with higher signal strength [24]. Especially in the beginning of AKA, there is no way for the UE to differentiate between the real SN and the fake ones. The UE has to share its IMSI with SN in order to start authentication. Therefore, IMSI catchers try to exploit this feature.

There are two types of IMSI catchers, passive and active. Passive IMSI catcher only gathers the information and identifies the IMSIs from the wireless traffic of the region. Passive one is only able to observe the specific neighborhood and detect IMSI if the UE tries to connect to SN [23]. Therefore, it is only possible to track the UE when the UE decides to send its IMSI. This typically happens only when the UE connects to the SN the first time. Another reason for UE to send its IMSI is when something has gone wrong in the network or in the UE. An active IMSI catcher is more compelling on getting IMSIs from the UEs. Active IMSI catcher is a "fake base station which acts as a preferred base station in terms of signal strength" [23]. Since there is not a chance for UE to authenticate the base station before it tries to connect, UE connects to the fake base station without a doubt. Moreover, when the IMSI catcher requests for identity, UE reveals its IMSI according to the standard process.

IMSI catchers are not newly developed devices that start to threaten security of people. The danger of IMSI catchers was already known by 3GPP during the development of 3G, because the history of IMSI catchers goes back to at least 1993 [25, 26]. This threat was not taken into consideration before, because it was difficult and expensive to build such device. One of the earlier IMSI catcher devices, called Stingray, was created in 2001 and was sold for $68,500, and the improved version of it came out six years later with a price of $135,000 [27]. Moreover, "only a few manufacturers existed, and the economic barrier limited the device's use mostly to governmental agencies" [26]. However, building IMSI catcher became cheaper recently. In 2010, an IMSI catcher was built for $1,500, then with the introduction of femtocells the cost of building a fake base station went even lower [23]. Obtaining cheap IMSI catchers enabled anyone, even other than government agencies, to use such devices for their own wills.

There are benefits of using active IMSI catchers as well as the harms. IMSI catchers can be used by a diversified range of people. Besides government and attackers, IMSI catchers are preferred by some companies for commercial issues [7]. By tracking movements of a person, a lot can be revealed about routines and preferences of people. Passive IMSI catchers help personalize advertisements for specific customers. This cannot be considered dangerous, but it is a serious violation of privacy. Benefits of location tracking are undeniable, if IMSI catchers are used correctly. For example, "law enforcement teams in the U.S. have used the technology to locate people of interest, to find equipment used in the commission of crimes" [28]. Thus, there is a chance to prevent terrorist attacks, or any kinds of physical assaults by using IMSI catchers. On the other hand, if the attackers aim for hurting people, they can wait for the target's arrival [23] or for the place to get crowded by observing through IMSI catcher and attack whenever the target area is full. In this case, the damages of IMSI catchers can be more crucial than the benefits, which makes it vital to look for readjustments of the current conditions.

4. Mobile Network Elements

Ever since the foundation of 1G, developments in mobile networks are sustained continuously and will continue developing. Despite the preservation of the basic overall structure, there have been some adjustments. Fourth Generation was using the *Evolved Packet System* (EPS) security architecture. Prior networks provided a basis for EPS, but some of the elements were improved or replaced. Necessary adjustments helped EPS to work with legacy networks, too. That is why, it is important to learn preceding networks very well, in order to break a new ground for new network. In this case, it is essential to learn about 4G and EPS so that 5G can be built on. In this chapter, only the elements of the network that take part in AKA will be explained. Figure 3 displays mentioned elements and their communication order.

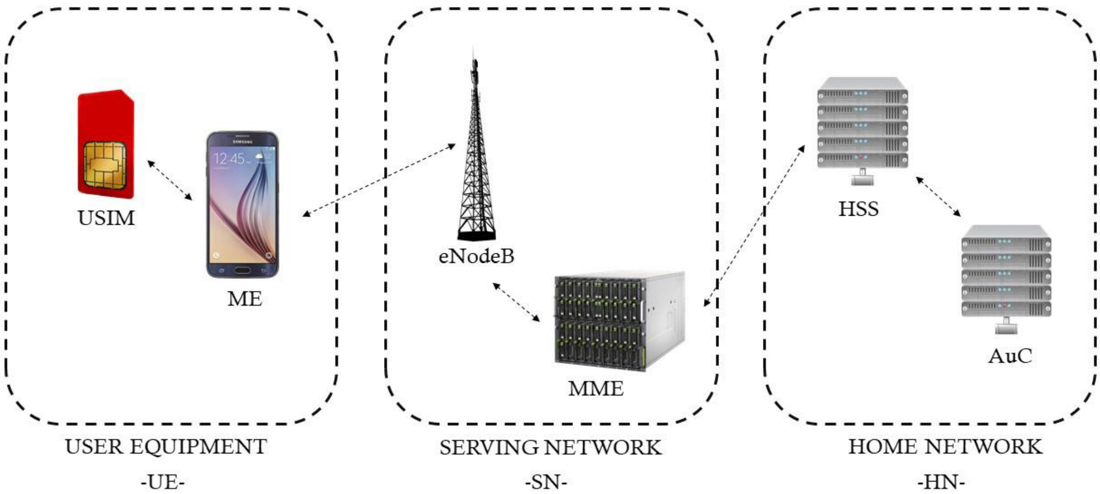


Figure 3: Mobile Network Security Architecture

4.1. Home Network (HN)

Home network is the operator, which provides service for user according to user's subscription. Authentication Center (AuC) and Home Subscriber Server (HSS) are main two components of HN that take part in AKA.

Authentication Center (AuC)

Authentication Center cooperates with HSS and generates necessary components for AKA. Later, HSS gathers these components and composes an *Authentication Vector* (AV). Authentication vector includes necessary information that is needed to be sent to UE, so that UE can successfully perform AKA. First, AuC begins with creating a sequence number (SQN) suitable for the UE. The main requirement for SQN is that it has not been used yet for this UE, but it should also be in some interval that helps UE and HSS to stay synchronized with each other. Then, AuC creates a random bit strings, called RAND, to use in the authentication challenges. After obtaining SQN and RAND, then AuC computes some values such as *Message Authentication Code* (MAC), *Expected Response* (XRES), *Cipher Key* (CK), *Integrity Key* (IK), *Anonymity Key* (AK), and *Authentication Token* (AUTN) by using SQN and RAND with secret key K [29]. These new computed values have particular tasks during AKA. For example, MAC helps UE to confirm that the message is sent from an authentic sender and not changed during communication. Then, XRES is for SN to authenticate UE, by comparing it to the parameter RES that UE computes and sends later in the protocol. This works because the correct RES can be calculated only by a correct UE that also has the same secret key K. Moreover, CK and IK are used by SN and UE for deriving further keys, starting from a key called K_{ASME}, so that they would not need to use a key more than once. Finally, AK is used for keeping SQN secret during the communication. The cryptographic MILENAGE functions are used for computing MAC, XRES, CK, IK, and AK [30]. Authentication Token includes necessary information that UE needs for participating and completing the authentication and is calculated as $AUTN = (SQN \oplus AK) \parallel AMF \parallel MAC$, where AMF is *Authentication Management Field* and used for revealing some specific information about other parts in AV or determining the time period of the key [29]. In the end, AuC forwards these parameters to HSS.

Home Subscriber Server (HSS)

Home Subscriber Server stores the subscription details of all subscribers in a database, such as "user identification, numbering and addressing information, security information, location information, and profile information" [31]. These details need to be preserved by HSS in order to ensure authentication and authorization. Moreover, HSS keeps track on *Mobility Management Entity* (MME) and makes sure that they are valid, while UEs are attaching them [32]. On the other hand, HSS trusts MME that MME would perform authentication honestly with short dated information, which comes in AV; but does not trust with the long-term credentials [23].

Home Subscriber Server is in interaction with AuC. When HSS needs to create an AV, AuC generates necessary components for HSS. Then, HSS computes K_{ASME} with the CK and IK, along with SQN [29]. Finally, HSS prepares $AV = RAND \parallel XRES \parallel K\_ASME \parallel AUTN$ and sends it to MME.

4.2. Serving Network (SN)

Serving network "provides the actual connectivity and mobility services" [23], by acting as a bridge between UE and HN. In roaming cases, SN can belong to different operator than the operator of the user. The two main components of SN, which take role in AKA, are Evolved NodeB (eNB) and Mobile Management Entity (MME).

Evolved NodeB (eNB)

Evolved NodeB is the name of base station in LTE [33]. Base station is a communication station, which receives and sends signals between the user and the rest of the network elements. The collection of eNBs is called *Evolved-Universal Terrestrial Radio Access Network* (E-UTRAN) and E-UTRAN manages the communication between UE and rest of the network. Both UE and MME send the requests and responses to eNB, then eNB forwards them back to MME and UE. On the other hand, eNBs in E-UTRAN have connections between each other, as well as to MME and to *Serving Gateway* (S-GW)[1]. The connection between eNBs with MME and S-GWs are shown in Figure 4.

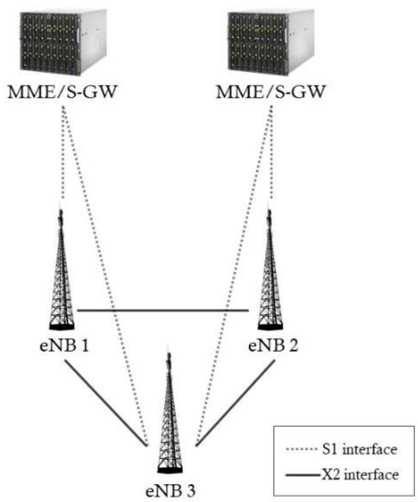


Figure 4: E-UTRAN architecture

Each eNB follows some protocols, which are collectively called *Access Stratum* (AS), during its communication with UE [33]. There are many functions that E-UTRAN is responsible for. First function is Radio Resource Management, which takes care of everything about radio bearers, such as "radio bearer control, radio admission control, radio mobility control, scheduling and dynamic allocation of resources to UEs in both uplink and downlink" [32]. Another function is Header Compression, and it compresses the IP packet header to increase efficiency of the network. The function that satisfies security requirements, sends all the data as encrypted [32]. The important point is that all these functions are embedded in eNBs, because each eNB can respond with the function that are restored in themselves. So, this gathering of the functions in eNB aims for decreasing latency, increasing

[1] S-GW tries to interwork with legacy networks, acts like the administrator of the visiting network in terms of billing the UE and supports lawful interception [32].

efficiency, providing high-availability, reducing the cost, and more importantly avoiding single point failures [32]. Because, all the eNBs possess the functions and can communicate with each other, they can share the information in case of a failure of one single eNB.

Mobility Management Entity (MME)

The MME takes care of authenticating the UE and supports providing service to the UE. When UE wants to connect to the network, MME requests authentication vector from HSS. Then, HSS returns with AV, which is prepared for this specific UE and this specific MME [29]. After obtaining AV, MME performs mutual authentication with UE by using the elements of AV. If the authentication succeeds, MME assigns TMSI to UE [32]. Later, UE uses TMSI instead of IMSI, when UE needs to connect to the network. By this way, MME can provide faster service, since MME already knows that UE is authenticated user.

As the main purpose, MME is responsible for tracking the location of UE on a large scale [12]. MME provides the location information to HSS and HSS keeps it in the database. If MME needs to check the location of UE, MME sends a message to trigger eNBs in the area, where UE is supposed to be. So, all the eNBs page UE, and UE replies to nearest one [32]. By the location of that base station, MME will be able to refresh the location information of the UE.

4.3. User Equipment (UE)

User equipment is the combination of Universal Subscriber Identity Module (USIM) and the Mobile Equipment (ME) together. User can connect to network through UE.

Universal Subscriber Identity Module (USIM)

Universal Subscriber Identity Module is included in a smart card, which is imbedded in a mobile device [23]. Important information that is necessary for authentication is stored in USIM. For example, IMSI and secret key K are stored in USIM. More generally, "the USIM contains all the operator-dependent data about the subscriber, including the permanent security information" [14]. Moreover, USIM also generates new keys from K by using *Key Derivation Functions* (KDF) and prepares responses for authentication protocol [33]. Universal Subscriber Identity Module takes an active role in generating new keys and responses, because secret information can be kept safer when it is not shared with anything else, even not with the ME.

Mobile Equipment (ME)

Mobile Equipment is the communication device that has "the radio functionality and all the protocols that are needed for communications with the network" [14], smartphone is an example of ME. In order to use the services, USIM is inserted in ME. Among other tasks, ME is responsible for sending and receiving necessary information between USIM and SN, as well as responding when an eNB is paging.

Apart from AKA, USIM has a separate authentication mechanism with ME. In the beginning, USIM requests for a Personal Identification Number (PIN), which only USIM and user knows. User needs to enter the PIN to ME to prove that the User is the correspondent to the USIM. In addition, there can actually be another PIN between the User and the ME. This PIN prevents anyone other than the authentic User to use ME.

5. Authentication and Key Agreement (AKA)

All the elements in a network interact with each other in many ways while providing and using service. During the interaction, they need to ensure that each element is valid and trustable. Verifying the identity is called authentication. In mobile networks, authentication consists of challenge response protocols [14].

5.1. GSM (2G) AKA

In GSM, UE consists of ME and SIM. *Base Transceiver Station* (BTS) and *Mobile Switching Center / Visitor Location Register* (MSC/VLR) are the components of SN. *Home Location Register* (HLR) and *Authentication Center* (AuC) form HN [14]. For GSM, only authentication of user is examined, SN and HN are trusted parties.

For each subscriber, there exists a master key K_i and this is located in the SIM of the user and in AuC. For providing security, K_i is never supposed to leave these locations.

Authentication is primarily based on checking if the user has possession of the specific K_i. Authentication process is summarized in Figure 5 and explained step by step:

- UE wants to connect to the network by sending its IMSI (or TMSI) to SN.
- SN forwards the IMSI to HN.
- HN assigns random RAND for the IMSI, calculates XRES and K_c by using the RAND and K_i. Then, HN returns (RAND, XRES, K_c) to SN.
- SN keeps XRES and K_c for itself and sends RAND to UE.
- UE calculates SRES and K_c. Then keeps K_c and sends SRES to SN.
- SN compares SRES and XRES, if they do not match, then connection request is rejected. Otherwise, the authentication is completed. Then, SN assigns TMSI to UE and sends it to UE after encrypting with K_c [14]. The K_c would be used for encrypting all messages until the authentication is redone.

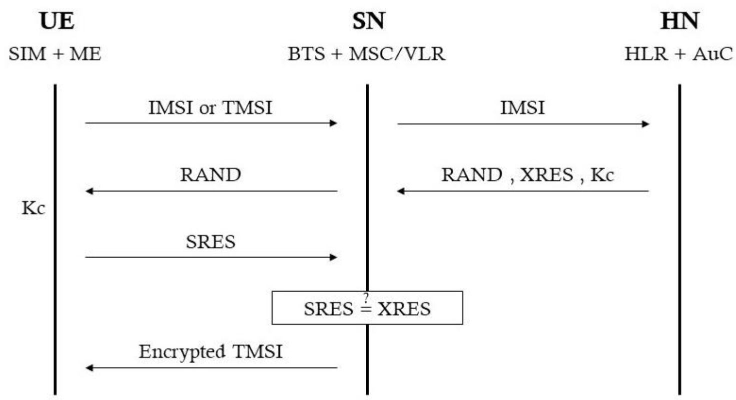


Figure 5: GSM AKA [14]

5.2. UMTS (3G) AKA

Principally, design of UMTS AKA relies on GSM AKA protocol, but with improvements. For example, GSM AKA is not meant to be secure against the active attacks from false base stations, because "such attacks, which would require the attacker to effectively have their own base station, would be too expensive compared to other methods of attacking GSM" [14]. As it is also mentioned in Chapter 3.2, it was thought that only governmental departments could afford such devices. However, 3G tried to reduce danger of false base stations and three new features were added to 3G UMTS AKA: authentication of the network (in addition to authenticating the user), generation of a key for integrity protection of signalling and prevention of replay of authentication messages [14]. These three are the biggest differences between GSM AKA and UMTS AKA.

Compared to GSM, in UE, SIM is replaced by USIM while ME remains under the same name. Then, SN consists of VLR/SGSN (*Serving GPRS Support Node*) and base stations, and HN is same as the HN in the GSM network.

As well as GSM AKA, there is also master key, K_i, which only USIM and AuC can possess. In UMTS AKA, mutual authentication is used, which means that while SN checks the identity of the user, user also checks if the SN is authorized by HN [14]. Even if the mutual authentication does not stop fake base stations completely, it would prevent serious outcomes.

Authentication process of UMTS is summarized in Figure 6 and explained step by step:

- UE sends its IMSI or TMSI to VLR/SGSN (SN).
- SN sends authentication request for related IMSI to AuC in HN.
- AuC prepares RAND, AUTN, XRES, CK (*Cipher Key*), and IK (*Integrity Key*) for requested IMSI, and sends it to SN.
- SN sends RAND and AUTN as authentication request to UE.
- USIM makes several calculations with K_i and RAND. First calculation, which is for verifying that AV is authentically produced in AuC, is compared with a value in AUTN. Then, USIM calculates RES, CK, and IK and sends RES back to SN.
- SN compares RES and XRES. If the results match, then authentication is successful [14]. Later, SN assigns TMSI for the user, encrypts it with a key CK and sends it to UE.
- After the authentication has been completed, all traffic between the UE and the network is encrypted by the key CK, and integrity of all control traffic is protected by the key IK.

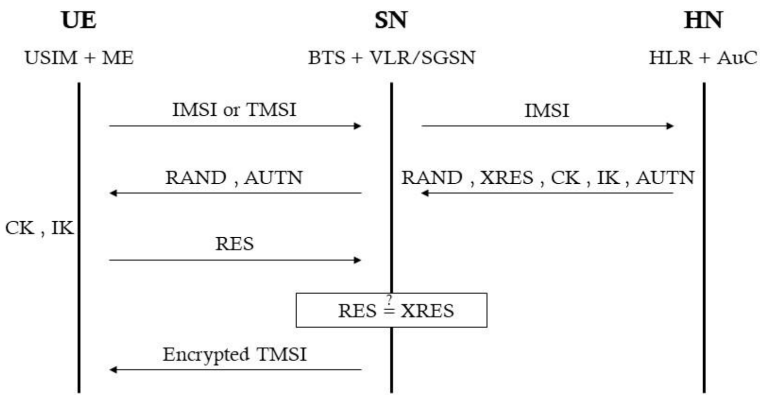


Figure 6: UMTS AKA [14]

5.3. EPS (4G) AKA

The EPS AKA is improved and reformed version of UMTS AKA. Therefore, some of the features are same in UMTS AKA and EPS AKA, but there are also differences. As network elements, MME in SN handles the roles of VLR/SGSN from UMTS [14] and a base station in EPS is called *eNodeB* (eNB). For HN, AuC is same as in UMTS, but HN has HSS instead of HLR. Moreover, UE does not have any new parts in EPS comparing to UMTS, it has still USIM and ME.

The structure of IMSI is also the same in EPS as in UMTS and GSM. It consists of MCC, MNC, and MSIN, which are explained in Chapter 1. Master key, K, is stored in USIM and AuC, and is not supposed to be transferred to anywhere else. The EPS names temporary user identities in a new way. Both GSM and UMTS were using TMSI, but now EPS uses *Globally Unique Temporary UE Identity* (GUTI). Globally Unique Temporary UE Identity is composed of two parts, *Globally Unique MME Identifier* (GUMMEI) and M-TMSI [14], where GUMMEI uniquely proclaims the MME that creates certain GUTI and GUMMEI consists of MCC, MNC, and MME Identifier, and M-TMSI is used to identify the UE that the GUTI is created for. Essentially, M-TMSI corresponds to the TMSI.

Authentication process of EPS AKA starts with the *Identity Request*, from MME to UE [34] and continues as:

- UE sends its IMSI or GUTI to MME. The UE captures SN_{id} of MME before sending its identifier to MME.
- MME sends an *Authentication Information Request* with IMSI and its SN_{id} to HN [35].
- AuC generates the elements of an authentication vector, RAND, XRES, CK, IK, and AUTN. Another difference of EPS AKA compared to UMTS AKA is with AMF, which is a component of AUTN. AMF is modified to store information about the AV. The reason for this change is that "it must be possible to use UMTS AKA and EPS AKA simultaneously in a single operator's network, and even in a single HLR/HSS and with the same AuC" [14]. So, by modifying a specific bit in AMF,

UE can understand if the AV is suitable for EPS or for legacy services. Then, for the EPS case, HSS obtains the components from AuC and computes K_{ASME} such as $K_{ASME} = KDF(CK, IK, SN_{id}, SQN \oplus AK)$. KDF is a key derivation function, which is explained in 3GPP TS 33.401 [35]. After K_{ASME} is ready, HSS sends authentication vector, $AV = RAND \parallel XRES \parallel K_{ASME} \parallel AUTN$, to MME as *Authentication Information Response*.

- MME keeps XRES and K_{ASME} for itself, then sends RAND and AUTN to UE as *User Authentication Request*.
- When UE receives AV, USIM immediately checks the freshness of the AV by controlling if the SQN is in acceptable range. To do this, USIM computes AK and reveals SQN. If the freshness is verified, then the authenticity of the sender is checked. USIM computes XMAC itself, and compares XMAC with MAC value in AV. If the authenticity is also verified, then USIM computes CK, IK, and RES. Then, ME sends RES to MME as *User Authentication Response* and computes K_{ASME} from CK, IK, and SN_{id}. The ME stores the new key.
- MME compares RES with XRES. If they match, then authentication is successful. MME creates GUTI for UE, encrypts it from a key, which is derived from K_{ASME} and sends it to UE.

Authentication and key agreement process in EPS is summarized and shown in Figure 7.

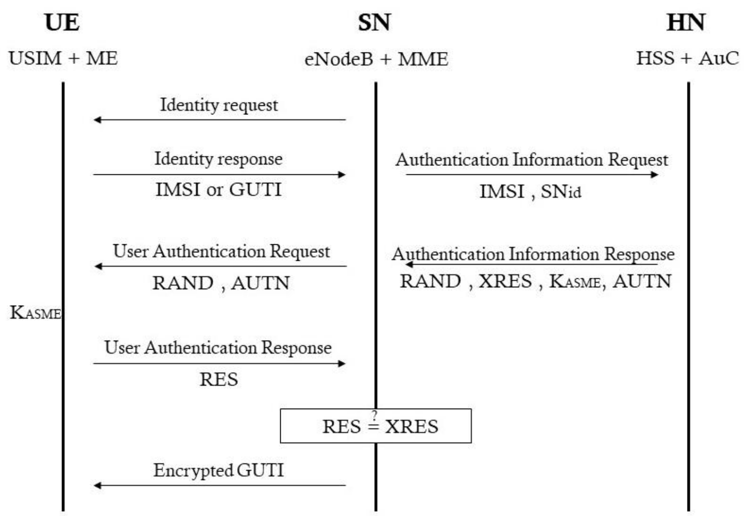


Figure 7: EPS AKA [14]

6. KASUMI

KASUMI is a symmetric key block cipher, which was designed for security architecture of 3GPP systems. KASUMI was accepted as a standard cipher in Europe for mobile phones in the beginning of 2000s [36]. Moreover, KASUMI was restricted to be used in encryption and integrity protection for the keys that are used in 3G and LTE.

KASUMI accepts 64-bit input and produces 64-bit output by using 128-bit key. This block cipher consists of 8 rounds. In each round, specific functions, which are defined for KASUMI, are executed.

Since KASUMI is the preference of 3GPP and each day users tend to use mobile technology more, this block cipher has liability for the security. There are many cryptanalyses for KASUMI, but until now there are no successful practical attacks. There are publications of attacks to 6 rounds of KASUMI, which would still leave 2 more rounds for security. Jia et al. performed impossible differential attack on the 7 rounds of 8 rounds. For this attack 2^{115} encryptions are required [37]. Even though the success of the attack is possible, it would require tremendous amount of time. On the other hand, Biham et al. tried another attack, called the related-key rectangle attack, on the full rounds of KASUMI. It requires 2^{76} encryptions [38]. This new attack is more compelling than the previous one, but still it is not fast enough.

6.1. Design of KASUMI

Before encryption, key scheduling is configured. In this phase, different keys are derived from the main key. Thereby, in each round of 8, different keys are used. After the key scheduling is completed, encryption starts. Both encryption and decryption are composed of various functions, which are explained below by following the rules of TS 35.202 [39].

6.2. Key schedule

For KASUMI block cipher, 128-bit key is used. In each round, each subfunction uses different keys. These keys are derived from the main 128-bit key, K.

First, 128-bit key is divided into 8 subkeys, each containing 16 bits:

$$K = K1 \parallel K2 \parallel K3 \parallel K4 \parallel K5 \parallel K6 \parallel K7 \parallel K8 .$$

Then, for each integer j, $1 \leq j \leq 8$, Kj' is computed such as $Kj' = Kj \oplus Cj$, where Cj is the constant value. These constant values are defined in the Table 2 in TS 35.202 [39]. For each integer j, $1 \leq j \leq 8$, Kj' is used during the derivation of round subkeys.

For the functions FL, FO, and FI, the keys KL_i, KO_i, and KI_i are derived respectively, where i represents the round of the cipher. The Table 1 in TS 35.202 [39] shows how to create subkeys for each round.

6.3. Functions

Function FL

Function FL takes 32-bit input I and produces 32-bit output O. The 32-bit subkey KL_i is divided into two pieces of 16 bits, such that $KL_i = KL_{i,1} \parallel KL_{i,2}$.

32-bit input is also divided into two pieces of 16 bits, such that $I = L \parallel R$.

Then, the computations are,

$$R' = R \oplus ROL(L \wedge KL_{i,1}) \text{ and}$$

$$L' = L \oplus ROL(R' \vee KL_{i,2}) ,$$

where ROL is circular left rotation by one bit. Finally, $O = L' \parallel R'$.

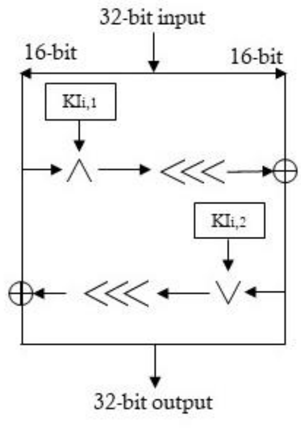


Figure 8: FL Function [39]

Function FO

Function FO accepts 32-bit input I and produces 32-bit output O. Two subkeys of 48 bits are used in this function, KO_i and KI_i. All of I, KO_i, and KI_i are divided into pieces of 16 bits such as, $I = L_0 \parallel R_0$, $KO_i = KO_{i,1} \parallel KO_{i,2} \parallel KO_{i,3}$, and $KI_i = KI_{i,1} \parallel KI_{i,2} \parallel KI_{i,3}$.

Then, for each integer j, $1 \leq j \leq 3$, R_j and L_j is calculated as,

$$R_j = FI(L_{j-1} \oplus KO_{i,j} , KI_{i,j}) \oplus R_{j-1}$$

$$L_j = R_{j-1} .$$

Finally, the output is $O = L_3 \parallel R_3$.

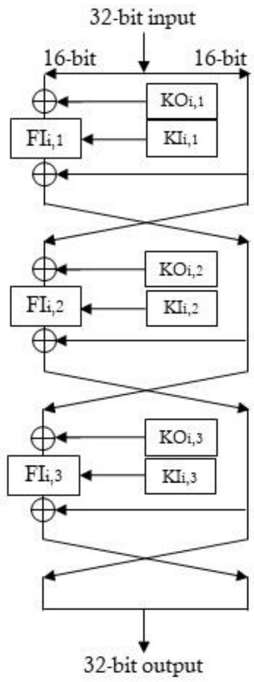


Figure 9: FO Function [39]

Function FI

Function FI takes 16-bit input I and gives 16-bit output O in the end. The subkey $K_{i,j}$ has 16 bits. Both I and $K_{i,j}$ are divided into two pieces of 9 bits and 7 bits:

$I = L_0 \parallel R_0$, where L_0 has 9 bits and R_0 has 7 bits, and $KI_{i,j} = KI_{i,j,1} \parallel KI_{i,j,2}$, where $KI_{i,j,1}$ has 7 bits and $KI_{i,j,2}$ has 9 bits.

In this function, there are two S-boxes, $S7$ and $S9$. TS 35.202 [39] explains the working principle of these boxes. Moreover, two other functions are also used for FI. One of the functions is ZE, which converts 7-bit string into 9-bit string by adding zeroes to the left. The other function is TR and it converts

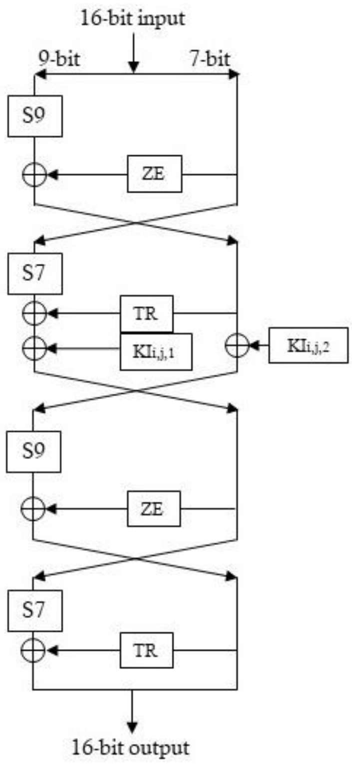


Figure 10: FI Function [39]

9-bit string to 7-bit string by deleting 2 values on the left end.

Then, the operations of function FI are,

$L_1 = R_0$ $R_1 = S9[L_0] \oplus ZE(R_0)$

$L_2 = R_1 \oplus KI_{i,j,2}$ $R_2 = S7[L_1] \oplus TR(R_1) \oplus KI_{i,j,1}$

$L_3 = R_2$ $R_3 = S9[L_2] \oplus ZE(R_2)$

$L_4 = S7(L_3) \oplus TR(R_3)$ $R_4 = R_3$

Therefore, the output becomes $O = L_4 \parallel R_4$.

Function fi

Finally, function f_i combines former functions and makes them ready for encryption. Function f_i accepts 32-bit input I and produces 32-bit output O, by using subkeys KL_i, KO_i, and KI_i.

When the round i is odd number, then

$$f_i(I, K_i) = FO(FL(I, KL_i), KO_i, KI_i)$$

When the round i is even number, then

$$f_i(I, K_i) = FL(FO(I, KO_i, KI_i), KL_i)$$

6.4. Encryption

For the encryption, input I of 64-bit and key K of 128-bit are required. In the end, the ciphertext C will be also 64-bit.

Before starting the encryption, I is divided into two pieces of 32-bit values, such as $I = L_0 \parallel R_0$. Moreover, K is also processed in key schedule, so a triplet $K_i = (KL_i, KO_i, KI_i)$, is obtained.

Encryption starts as,

$$R_i = L_{i-1}$$

$$L_i = R_{i-1} \oplus f_i(L_{i-1}, K_i) \ .$$

Finally, $KASUMI(I, K) = L_8 \parallel R_8$.

6.5. Decryption

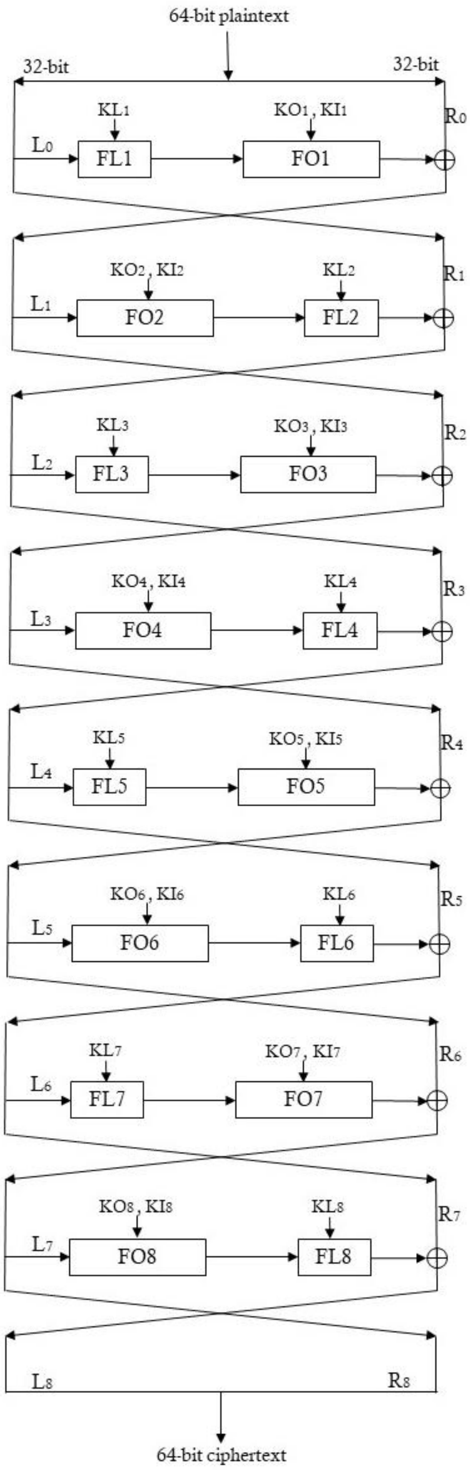
Decryption of KASUMI starts in a similar way like encryption. The 64-bit ciphertext C and key K are accepted as inputs. In the beginning, $C = L_8 \parallel R_8$ and KI_i are ready for decryption.

Decryptions starts as,

$$L_{i-1} = R_i$$

$$R_{i-1} = L_i \oplus f_i(L_{i-1}, K_i) \ .$$

In the end, $L_0 \parallel R_0$ is the plaintext.



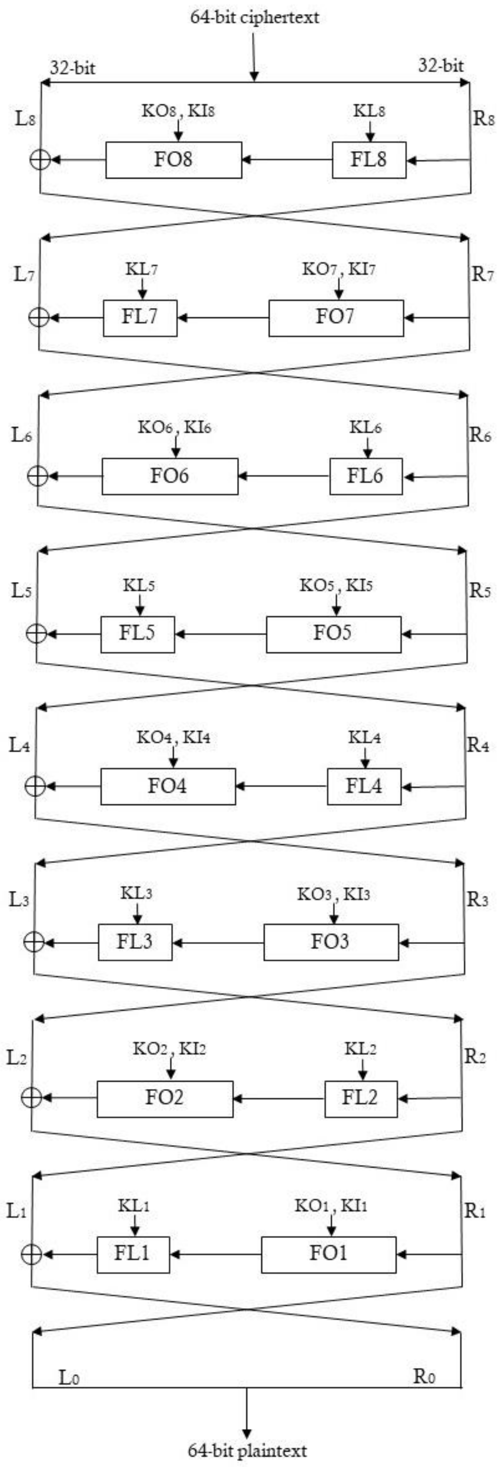


Figure 11, on the left, describes KASUMI encryption; whereas Figure 12, on the right, describes KASUMI decryption [39].

7. Structure of 5G

In spring 2018, Phase 1 of 5G development is coming to an end, but refinement process is still continuing. As well as the other protocols of 5G, 3GPP agreed on certain concepts of 5G AKA for Phase 1. Even though 5G AKA is open for improvements for further phases, the specifics about 5G AKA in Phase 1 are presented in 3GPP TS 33.501 [40]. It is important to learn about the architecture and AKA procedure in Phase 1 of 5G to continue improving the system.

7.1. 5G Architecture

There are many differences in 5G compared to the earlier generations and there are new elements introduced to the network. In the paper of Zhang et al., some of the changes in 5G architecture are explained, whereas in this section, only the separation of user plane from control plane is explained [41].

After user plane is taken apart from the control plane, UE lies in user plane along with base station, User Plane Function (UPF) and Data Network (DN). On the other hand, UE and base station are also in control plane where mobility and session management are divided into two functions. These are *Core Access and Management Function* (AMF) and *Session Management Function* (SMF). Other than AMF and SMF, there are new elements in 5G architecture, some of which are listed as follows:

- Security Anchor Function (SEAF)
- Authentication Server Function (AUSF)
- Authentication Credential Repository and Processing Function (ARPF)
- Security Context Management Function (SCMF)
- Security Policy Control Function (SPCF) [41].

First, SEAF is adjoined with AMF and used for creating key to provide security between UE and SN for the authentication. Another function that is adjoined with

AMF is SCMF, which extracts keys that are created in SEAF and derives into other keys to participate in different areas of network. Then, ARPF is adjoined with Unified Data Management (UDM) and keeps credentials related to security, like the key for AKA. Moreover, AUSF interacts between SEAF and ARPF, concludes the requests from SEAF and collaborates with ARPF. In the end, SPCF provides security policies for all the elements of the network [41]. All the network elements of 5G and the connection between them are displayed in Figure 13.

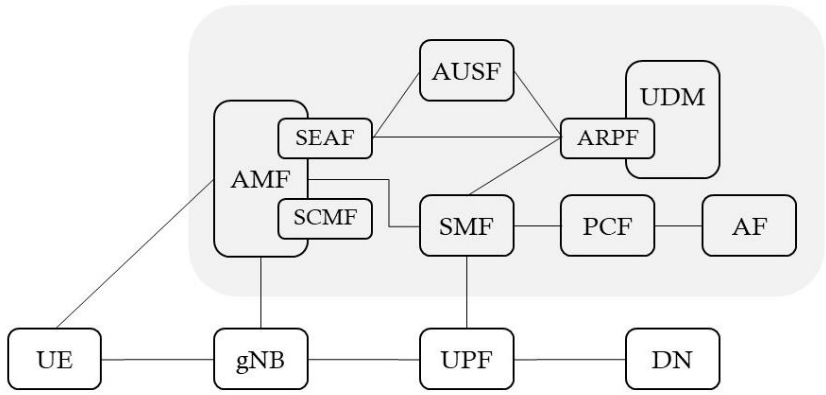


Figure 13: 5G network architecture [41].

7.2. 5G AKA

Even though the topic is open for further improvements, 3GPP presented the specifics about 5G AKA in Phase 1 in 3GPP TS 33.501 [40]. All the information in this Chapter (7.2) is adapted from this specification, unless stated otherwise.

In 5G, names of identifiers are different comparing to the earlier generations. One of the new identifier is *Subscription Permanent Identifier* (SUPI). The SUPI is the combination of IMSI and *Network Access Identifier* (NAI). Since IMSI is required in 3GPP legacy networks, SUPI is generally preferred to be same as IMSI for 3GPP networks. On the other hand, introducing NAI to SUPI will help SUPI to be used in non-3GPP networks as well, which do not require IMSI [43]. Another identifier is *Subscription Concealed Identifier* (SUCI) and SUCI is concealed version of IMSI-like SUPI. In other words, MSIN part of SUCI is concealed, while the other parts are in plaintext. Still another identifier is *5G Globally Unique Temporary Identifier* (5G-GUTI), which is assigned to UE by AMF and can be used for both 3GPP and non-3GPP access. Moreover, 5G-GUTI is composed of two components: GUAMI (Globally Unique AMF ID) and 5G-TMSI. Along with some codes, which defines the identity of AMF, GUAMI includes MCC and MNC, and 5G-TMSI is the same as TMSI, which identifies UE specifically to one AMF [43].

There are two types of AKA in 5G, one is EAP-AKA' and the other is 5G AKA (for the latter, also the term EPS-AKA* is used) [44]. Selection between the types of AKA is left up to the operators. Both AKA processes start with same initiation phase, then continue according to the selected type. The main idea of authentication and key agreement is same as earlier networks, like 3G and 4G. However, some improvements are applied to 5G AKA to provide more secure environment.

In the result of the authentication and key agreement procedure, the end-product is the key called K_{SEAF}. The importance of K_{SEAF} lies behind the fact that SN_{id} is used during the calculation of K_{SEAF}. In other words, K_{SEAF} specifically displays the SN that UE is connecting to. Thus, fake or unauthorized SNs would not be able to pretend as they are legitimate. Therefore, this feature gives UE a chance to authenticate SN.

Initiation:

Initiation of AKA is the same for both types. This process is summarized in Figure 14 and explained:

- UE starts authentication by sending its SUCI or 5G-GUTI to SEAF in SN. In some cases, SEAF can force UE to start the authentication.

- SEAF receives the identifier of UE. So, SEAF should send '*5G Authentication Initiation Request*' (5G-AIR) to AUSF. If the identifier is a valid 5G-GUTI, then it means that SEAF authenticated UE before. So, SEAF places SUPI as identifier in 5G-AIR. On the other hand, if the identifier is SUCI, then SEAF puts SUCI to 5G-AIR. In 5G-AIR, the identifier of UE, an indication that shows if the connection is for 3GPP or non-3GPP access[2], and the SN name are included. Moreover, SN name is determined with the concatenation of 5G and SN_{id}. Hence, SEAF sends 5G-AIR to AUSF.

- AUSF receives the '5G Authentication Initiation Request' and directly checks if SEAF is entitled to send authentication request. If SEAF is valid, then AUSF prepares '*Authentication Information Request*' (AIR) for UDM. Authentication Information Request includes SUCI or SUPI, depending on the 5G-AIR content, SN name, an indication that shows if the connection is for 3GPP or non-3GPP access, and the number of AVs that are requested. AUSF sends AIR to UDM.

- UDM receives AIR from AUSF. First of all, if the identifier is SUCI, then AUSF gets the SUPI out of concealed identity SUCI. After getting SUPI, the UDM decides which AKA type is going to be used. This choice is made "based on the subscription data and the access network type, 3GPP access or non-3GPP access" [40].

Then, AKA continues with either EAP-AKA' or EPS AKA*. While EAP-AKA' can be chosen for both 3GPP access and non-3GPP access, EPS AKA* can only be chosen for 3GPP access [40].

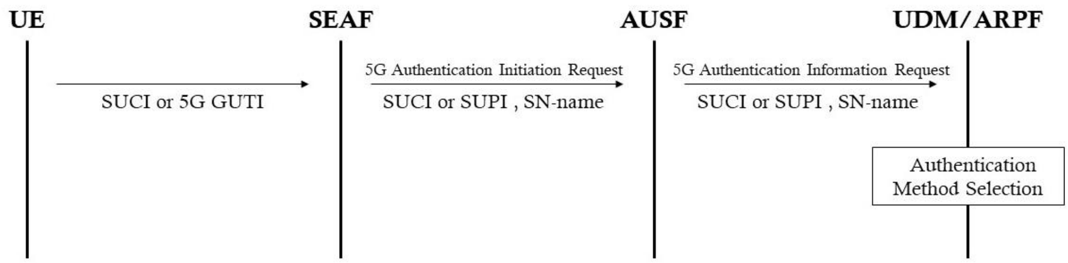


Figure 14: Initiation phase of 5G AKA [40]

[2] 3GPP access is when the protocols are determined by 3GPP, such as GSM, 3G, LTE, and 5G. Non-3GPP access is the other connections like WIFI, cable, ethernet.

EAP-AKA'

After the authentication method is specified and chosen as EAP-AKA':

- UDM generates AV. UDM modifies the separation bit in AMF according to their choice of AKA procedure and computes CK' and IK', as they are specified in TS 33.501 [40]. Then, authentication vector becomes ready as $AV = (RAND, AUTN, XRES, CK', IK')$. Finally, UDM sends AV in 'Authentication Information Response' to AUSF.

- AUSF receives the AV, forwards it to SEAF as EAP-Request/AKA'-Challenge in the message, '5G Authentication Initiation Answer'.

- SEAF is trusted to send the EAP-Request/AKA'-Challenge without intercepting the content. So SEAF sends it in 'Authentication Request' message to UE.

- UE receives 'Authentication Request' with EAP-Request/AKA'-Challenge. At this step, UE verifies the message and makes necessary calculations. Then, prepares and sends 'Authentication Response' with EAP-Response/AKA'-Challenge to SEAF.

- SEAF receives EAP-Response/AKA'-Challenge transfers it directly to AUSF without intercepting.

- AUSF receives EAP-Response/AKA'-Challenge and verifies it. If the verification is successful, AUSF creates K_{SEAF} from K_{AUSF}. Moreover, AUSF prepares EAP-Success message. Then, AUSF sends EAP-Success message and K_{SEAF} to SEAF. If SEAF sent SUCI in the initiation part, then AUSF also sends SUPI to SEAF.

- SEAF receives EAP-Success messages along with K_{SEAF} and, as occasion requires, SUPI. Then, SEAF forwards EAP-Success message to UE.

- After receiving EAP-Success message, UE computes K_{SEAF} after computing K_{AUSF}, similarly as AUSF computed these keys.

Figure 15 summarizes the communication between the elements during EAP-AKA'.

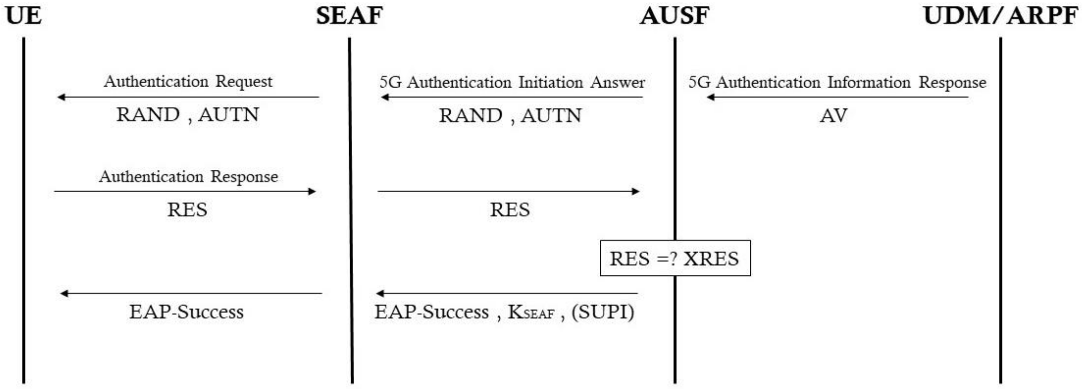


Figure 15: EAP-AKA' [40]

5G AKA (EPS-AKA)*

After the authentication method is specified and chosen as 5G AKA,

- UDM generates 5G HE AV (5G Home Environment Authentication Vector). To generate 5G HE AV, UDM first modifies AMF's separation bit as necessary. Then, UDM computes K_{AUSF} from CK, IK, $SQN \oplus AK$, and SN's name. Moreover, UDM prepares XRES* by using CK, IK, XRES, RAND, and SN's name. Thus, the 5G HE AV is composed with $RAND, AUTN, XRES*, K_{AUSF}$ and is sent to AUSF in 'Authentication Information Response' message.

- AUSF receives the 5G HE AV and prepares 5G AV from 5G HE AV. To do this, first AUSF calculates hash of XRES* to create HXRES*. Besides, AUSF should store XRES* until the time stamp expires. Then, AUSF computes K_{SEAF} from K_{AUSF}. Finally, AUSF gathers the components of $5G\ AV = RAND, AUTN, HXRES*, K_{SEAF}$ and sends 5G AV in '5G Authentication Initiation Answer' to SEAF. If SEAF sent SUCI in the initiation part, then AUSF sends also SUPI of UE to SEAF.

- SEAF receives 5G AV and sends RAND and AUTN in 'Authentication Request' message to UE.

- UE receives the message and USIM in UE computes RES, CK, and IK. Then, USIM sends them to ME and ME calculates RES* with respect to necessary functions. Later, ME sends RES* in 'Authentication Response' message to SEAF and UE calculates K_{AUSF} and K_{SEAF}, just the way UDM and AUSF calculated, respectively.

- After receiving RES*, SEAF calculates hash of RES*, which is called HRES*. Then, SEAF compares HRES* with HXRES*. If these two are the identical, then it means that authentication is successful. SEAF sends RES* to AUSF in '5G Authentication Confirmation' message.

- AUSF receives RES* and compares it with XRES*, which was stored earlier. If these two are identical same, AUSF understand that the authentication is done successfully.

Figure 16 summarizes the communication between the elements during 5G AKA (EPS-AKA*).

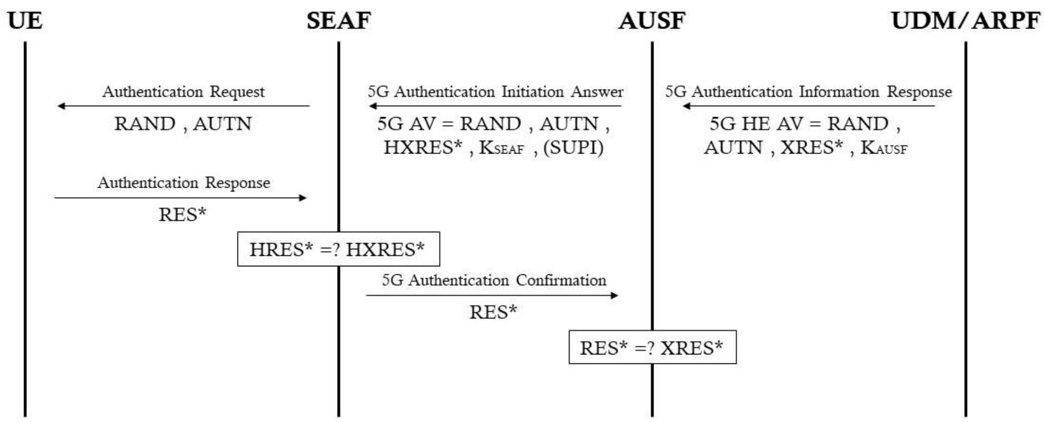


Figure 16: 5G AKA (EPS-AKA*) [40]

8. Identity Privacy in 5G

IMSI catchers are causing insecurity for the users and invading their identity privacy, as explained in Chapter 3.2. Therefore, providing identity privacy became one of the main issues for developing 5G. In order to provide identity privacy for the users, the important point is to avoid exposing IMSI to untrusted parties. Some different approaches are being discussed for executing AKA without endangering the identity privacy. In this section, we focus on the case where SUPI equals IMSI. The discussion could be generalized also to cover the case where SUPI equals NAI.

8.1. Public Key Approach

In public key approach, HN shares its public key with UEs, and keeps the private key safe. Then, UE encrypts only the MSIN part of its IMSI and keeps MCC and MNC as a plaintext. If MCC and MNC would also be encrypted, since none of the components other than UE and HN have access to the private key of HN, it would be impossible to transfer IMSI to a correct end. Therefore, UE identifies itself to the network with the $Encrypted\ IMSI = MCC \parallel MNC \parallel Encrypted\ MSIN$. Afterwards, AV is prepared with using plaintext IMSI [45].

It is important in public key approach is to end up with different ciphertexts each time when IMSI is encrypted. If the encrypted IMSI is the same at every turn, attackers can easily identify the same users without knowing their IMSIs. Therefore, anonymity and privacy would be damaged. To provide the security, a way should be found to randomize the encryption [25]. If attackers get the public key of HN, they can encrypt some random IMSIs and try connecting to network with someone else's account. Moreover, attackers can intercept the connection and provide UE with some wrong key, which would cause UE to lose connection. Therefore, to provide the confidentiality, one option is to install the public key in the SIM card, before delivering the SIM card to user. Otherwise, presenting valid certificate to UE becomes obligatory, so UE can be sure that the public key is trustable.

Root-key solution is the example of installing the public key to SIM card. In this solution, there is only one pair of public-private key pair for HN. Therefore, HN shares its public key with all the UEs, in other words with all of its subscribers. When UE wants to identify itself, UE encrypts MSIN part of IMSI with the public key of HN and sends the result to SN. After SN forwards the attach request to corresponding HN, HN decrypts and reveals the plaintext IMSI. Then, HN replies SN with cleartext IMSI and AV in a secure channel. At this point, AKA is executed between UE and SN, and SN assigns TMSI for UE. Therefore, there would not be a reason for using encrypted IMSI for the next session, because TMSI could be used instead.

In case of building certificate-based *Public Key Infrastructure* (PKI) for public key approach, then there are different types of solutions. To clarify the terms, the role of Certificate Authority (CA) in general can be explained as "a (digital) certificate is a signature by a trusted certificate authority (CA) that securely binds together several quantities. Typically, these quantities include at least the name of a user and its public key" [46]. Root CA is a trusted source, who can sign for its own certificate. In this sense, root certificate means self-signed certificate. First type of certificate-based PKI is choosing a trusted global entity for root CA. SN gives the public key and certificate, issued for the public key, to UE. If UE verifies the certificate, then UE encrypts its IMSI with the public key of SN and sends to SN.

In the second type, HN is the root CA. So, HN creates and signs the certificate for the public key and UE obtains the certificate beforehand. Moreover, HN creates a certificate for public key of SN, too. When UE wants to connect to the network, sends the public key of HN with corresponding network ID to SN. Then, SN presents its own certificate and signed public key to UE. If UE can verify the certificate of SN, UE encrypts IMSI with the public key of SN.

Third type has HN as the root CA as type two, but there is not any other CAs. In this type, UE have obtained the certificates of all possible SNs that UE can visit. Therefore, when UE wants to connect to SN, UE encrypts IMSI corresponding public key of SN [47]. Third type is more straightforward than the others, which eliminates the verification process and reduces calculation time. On the other hand, creating public-private key pairs for each authentication session, preparing certificate for the public key, and having an agreement between two parties would cause latency and workload.

One of the proposals about key agreement is based on Diffie-Hellman key exchange. Jimenez et al. suggest that the public-private key pair of HN always stays the same, but UE creates new pair of public-private key pair each time [48]. Therefore, same plaintext (IMSI) would be encrypted by different key, so the ciphertext would be different all the time. Then UE would send $Encrypted\ IMSI = MCC \parallel MNC \parallel Encrypted\ MSIN \parallel UE\ Public\ Key$ to SN.

Another issue about public key approach is about encryption. Since the public key belongs to HN, UE makes encryption, while HN makes decryption. There are some algorithms that are proven secure with required length of bits, such as RSA and *Elliptic Curve Cryptography* (ECC) [49], which can be chosen for the implementation of public key approach in 5G. According to Ginzboorg and Niemi, encryption is faster than decryption in RSA cryptosystem, but both encryption and decryption take approximately same time in ECC [25]. Moreover, the effect on bandwidth also differs between RSA and ECC. For example, "The European Union Agency for Network and Information Security (ENISA) recommends for RSA for the length of n 3072 bits for medium term, 15360 bits for long term security; for ECC for the greatest prime divisor of the group order 160 bits for medium term and 512 its for long term security" [49], where n is product of two large prime numbers. Summarizing, security with ECC can be provided with shorter keys, than with RSA.

One of the negative impacts of public key approach is computational load and bandwidth. In total, IMSI has 15 digits (60 bits) and MSIN has 10 digits (40 bits). However, after applying public key encryption (e.g. RSA) on 40 bits, the ciphertext would have more than 2000 bits [48]. Therefore, size of encrypted IMSI would create a huge bandwidth problem. The limit of computational load depends on the chosen cryptosystem and the chosen key.

Another negative side is that public key approach is not backward compatible. In the interview of Business Today, Joakim Sorelius from Ericsson claims "5G will be introduced across new spectrum bands that are not available today because it will not be backward compatible. So new devices will have to be developed. All device manufacturers are working on developing 5G and testing the same" [49]. This explanation means that each component of network needs to be changed or developed. Investments of the phone companies would be in high quantities, which would lead for expensive service for the subscribers. Other than service, financial effect would come to surface, when all the devices should be replaced with the ones with 5G compatibility.

8.2. Pseudonym Approach

Pseudonyms are temporary identifiers that are allocated for the UEs [47]. As a structure, a pseudonym looks like an IMSI and shares the same MCC and MNC with the IMSI. However, MSIN part of pseudonym differs from MSIN of IMSI. Only HN can correlate the pseudonym with the IMSI of the user. Therefore, when UE uses the pseudonym for identification, none of the attackers or SN would understand if it is real IMSI or not.

Creating pseudonym is an issue with some various proposals. The pseudonym replaces the MSIN part of IMSI, not the whole IMSI. The MCC and MNC would stay the same in order to make the destination HN clear. The most important point is that the new pseudonym should not match to any existing IMSI. One of the ways of creating pseudonym is choosing some random numbers [25]. After creating the random number, it can be compared to the existing IMSIs. If it does not have a match, then it is assigned to be the pseudonym of the UE. Another way is creating pseudonym from IMSI with a specific function by using K_{ASME} [23]. Therefore, HN can easily follow up the pseudonym from IMSI, in case the connection is lost. One more suggestion of creating pseudonym is encrypting IMSI with some random number and a session key [45]. Therefore, the new pseudonym would look random and be unknown if it is related to the real IMSI.

There are different approaches about the initial attach of UE with pseudonym-based approach. One suggestion is assigning UE a pseudonym in advance [25]. A pseudonym might be embedded to the SIM card along with IMSI, master key, and other necessary information. In this way, UE would start by using pseudonym instead of disclosing the IMSI. Another approach is encrypting IMSI before sending for attach request [23]. UE encrypts the MSIN part of the IMSI with the public key of HN. The difference of this approach with the public key approach is that encryption only occurs in initial attach, until HN assigns a pseudonym for UE.

An important point of pseudonym-based approach is the necessity of renewing pseudonym periodically [25]. Each time UE uses the pseudonym and HN needs to prepare AV for UE, where HN creates a new pseudonym and encrypts it with a key. This key can be the master key or some other key that is derived by the master key. Then, HN embeds the encrypted pseudonym in the AV as explained in Chapter 9.1.

One of the benefits of pseudonym is that this approach can be compatible with legacy networks. "The pseudonym mechanism could work even when the serving network is not aware of the existence of such mechanism" [25]. It is very important feature, because if the user travels to some places without 5G technology, he still can use pseudonym mechanism and preserve his/her identity privacy. Since pseudonyms have IMSI-like structure, no one in the middle of UE and HN would notice the difference.

The main problem of pseudonym shows up when the synchronization between UE and HN is lost. Attacker can force UE to reveal its IMSI by spoiling its connection with HN. For example, if UE uses encrypted IMSI mechanism and SN does not support 5G requirements, then UE is supposed to give plaintext IMSI for identification. The only way to avoid revealing IMSI as plaintext is if UE visits HN physically and share information in secure environment [47]. This is time-consuming and complex action to do in order to provide synchronization again securely.

8.3. Comparison of Public Key and Pseudonym Approaches

Both Public Key Approach and Pseudonym Approach have their own benefits. Even though it is agreed that public key approach is used for 5G Phase 1, applying only public key approach to 5G is not completely solving the problem. Public key approach does not have compatibility with legacy systems. All components should have ability to comply with legacy systems. For example, someone with 5G phone can travel to a foreign country with 4G or even earlier technologies. In this case, this user suddenly becomes vulnerable for identity privacy issues, and the dangers that 5G aims to discard. Moreover, in some areas in the country with 5G can have weaker connection. Then, phone automatically switches back to 4G or earlier networks, which makes the user vulnerable, again. Therefore, attackers can exploit this situation by forcing phones to fall back to legacy networks. In these cases, the public key approach loses its meaning. Public key approach on an individual basis can be securely applicable, when it becomes possible to abandon all the former networks.

On the other hand, pseudonym approach can work with legacy networks, because SN does not need to know whether UE sends IMSI or pseudonym. In this case, even when the user with 5G UE goes to another country with 4G, the user can give pseudonym as an identifier and HN will provide necessary AV. The problem here appears if the pseudonym synchronization is lost, because then cleartext IMSI should be revealed and attackers might exploit this situation. On the other hand, since pseudonyms look alike IMSI, there might be shortage for finding suitable pseudonym after a while. Before finding solution, it is important to decide how many pseudonyms should be stored related to a specific IMSI in HN or UE. Then, target number of the customers can be determined. If the number of the customers exceeds the limit, then additional MNC can be added, so same MSINs for IMSIs and pseudonyms can be used again.

Combination of public key approach and pseudonym approach can provide more secure environment, especially in case of identity privacy. Encrypting IMSI while sending to SN would avoid the risk of revealing the identity. The same way, pseudonyms can be encrypted like IMSI, too. However, if the user needs to be in a place without 5G, then he can use pseudonym to identify himself and keep his privacy intact.

9. Implemented Prototype

During the times that 5G development is in progress, we decided to make a pre-standard prototype for 5G security. We chose pseudonym approach for prototype implementation for this purpose. In this section, we describe this prototype. Due to the possiblity that pseudonym approach can be compatible with legacy networks, protection can be introduced immediately with pseudonym approach. In order to understand how this feature works, the implementation of the prototype is developed. A live demonstration can be done with the prototype and this helps in distinguishing the advantages and disadvantages of the pseudonym mechanism.

9.1. Illustration of Pseudonym Mechanism

The prototype is implemented for demonstrating identification, authentication and key agreement between UE and HN through SN. Before the actual demonstration starts, some preparations are needed in the prototype. Unique number IMSI, secret key K_{master}, OP (Operator Variant Algorithm Configuration Field), and SQN (Sequence Number) are derived.

User Equipment has its own database. In the database, IMSI, K_{master}, OP, and SQN are stored. Pseudonyms, P_{new} and P_{used}, are also stored in the database after they have been created. Home Network has also its own database, similar to the one that UE has. In the database, IMSI, K_{master}, OP, and SQN are stored. Pseudonyms, P_{new} and P_{used} are also stored in the database. Serving network has a database to store IMSI and XRES together.

There are 3 types of RANDs in the prototype and these are called R1-, R2-, and R3-type RAND. Each type has different tasks in AKA. R1-type and R3-type RANDs are randomly generated 128-bit arrays. The R1-type RAND is used for key creation, and this key will be used for encrypting and decrypting the new pseudonym. The R3-type RAND does not have any specific purposes in addition to what is specified for

AKA.When the R2-type RAND is in use, then there is a need for assigning a new pseudonym for UE. First, random 10-digit number is generated and crosschecked with all the numbers in the database to avoid overlapping with other IMSIs or pseudonyms. Then, the pseudonym is stored in the HN database as P_{new}. After that, pseudonym would be encrypted and embedded in RAND. Then, this RAND becomes R2-type RAND.

To make a request for attachment to the network, UE needs to send its IMSI or one of the stored pseudonyms to SN. As explained earlier, IMSI is composed of three parts, which are MCC, MNC, and MSIN. For example, 244 is MCC code for Finland and 12 is MNC code for DNA Oy [8], so 244 12 1234567890 is a representative example of IMSI, where MSIN is 1234567890. Pseudonym that corresponds to this IMSI would have exactly same structure, but different MSIN.

- In the beginning, only IMSI is stored in both the UE database and the HN database. So, the demonstration starts by UE sending its IMSI to SN.

- After UE has sent a request for attachment, SN receives IMSI of the user and stores it to its own database. According to the MCC and MNC codes, SN diverts the attachment request to the corresponding HN.

- HN receives IMSI from SN, starts a search in its database in order to check if IMSI belongs to a valid user. If the IMSI is valid, then HN prepares a R1-type RAND for key creation. After finalizing RAND generation, HN creates the key and stores it to the database for the next time. Moreover, AUTN is generated corresponding to the RAND as $SQN \oplus AK \parallel AMF \parallel MAC$. In the end, HN attaches XRES to the message to be sent to SN, then sends $AV = RAND \parallel AUTN \parallel XRES$ to SN.

- SN receives AV from HN for the corresponding IMSI. AV consists of $RAND \parallel AUTN \parallel XRES$. Next, SN takes XRES out from the AV, and sends the rest to UE. In the meantime, SN stores XRES to its database with IMSI.

- UE receives an AV from SN. This AV includes RAND and AUTN. Authentication token was prepared as $AUTN = SQN \oplus AK \parallel AMF \parallel MAC$. Here, AMF stores information about the type of RAND, so UE understand the purpose of RAND by checking AMF. At this point of the procedure, the AMF reveals that the type of RAND is R1-type and the purpose is then key creation for decrypting pseudonym later. Therefore, UE prepares the key by using this RAND and stores to the database. Then, MAC is used for authenticating HN and making sure that AV is not modified by someone else. First, UE computes MAC itself by using RAND and K_{master}. Then, UE compares computed MAC with the MAC from AUTN. If the comparison is successful, UE continues processing. Just as UE calculated MAC, UE can calculate AK by using similar functions. Therefore, UE can easily recover SQN by computing $AK \oplus (SQN \oplus AK)$ and check if SQN is in acceptable interval. If the check is successful, UE computes RES and sends it to SN.

- SN receives RES from UE and compares RES with XRES, because RES is a value that only an authentic UE can calculate. Then, SN notifies both UE and HN about the result. If the result is a match, authentication is successful, so that the UE can start using services through SN. Otherwise, connection drops and SN waits for further connection requests.

- HN receives the result of RES comparison from SN. Depending on the outcome, HN finalizes the procedure. If the outcome is positive, then HN knows that authentication is succeeded, and UE started using the services. However, if the comparison has failed, then HN understands something went wrong and services cannot be used.

- UE receives the result of RES comparison from SN. If the result is successful, authentication is succeeded. Otherwise, authentication fails, and UE needs to make another attempt for network attachment.

- When authentication has succeeded after sending IMSI for the first time, UE immediately sends IMSI to SN again and starts a new authentication automatically.

- SN receives the attach request and sends it directly to HN.

- HN receives IMSI along with the attach request. When HN receives IMSI for the second time, immediately after the first attempt, HN knows it should prepare R2-type RAND for assigning a pseudonym for UE. Then, HN prepares AUTN and XRES by using the generated RAND and sends $AV = RAND \parallel AUTN \parallel XRES$ to SN.

- SN receives AV from HN and keeps XRES in its database. Then, SN forwards RAND and AUTN to UE.

- UE receives a message from SN and extracts RAND and AUTN. First, UE checks MAC and SQN. If they both check out, UE checks AMF to understand the purpose of the RAND. Here, RAND is R2-type and there is an encrypted pseudonym in the RAND. Therefore, by using the key obtained from previous AKA, UE decrypts the pseudonym and stores to its database as P_{new} for further use. After that, UE computes RES and sends it to SN.

- SN receives RES from UE. Then, SN compares RES with XRES and notifies both UE and HN about the result.

- UE and HN receive result from SN. If the result is positive, then UE starts using the service. Otherwise, both UE and HN erase the new pseudonym from their databases.

- Next time when UE wants to attach, UE sends the new pseudonym P_{new} (instead of IMSI) to SN.

- SN receives the pseudonym from UE. However, SN would not understand that the pseudonym belongs to the previous UE, so SN assumes that the pseudonym belongs to a new UE. Therefore, SN stores the pseudonym to the database and forwards attach request to HN.

- HN receives the pseudonym from SN. Then, HN checks its database and understands that the pseudonym is the new pseudonym, earlier assigned to IMSI. Since the identifier is the pseudonym P_{new}, HN must assign another pseudonym to UE. Therefore, HN prepares R2-type RAND, computes corresponding AUTN and XRES. Finally, HN sends $AV = RAND \parallel AUTN \parallel XRES$ to SN.

- SN receives AV from HN and keeps XRES to the database. Then, SN forwards RAND and AUTN to UE.

- UE receives a message from SN and extracts RAND and AUTN. First, UE checks MAC and SQN. If they both check out, then UE checks AMF to understand the purpose of the RAND. Here, RAND is again R2-type and there is an encrypted pseudonym in the RAND. Before decrypting pseudonym, UE rearranges database by putting the stored pseudonym from P_{new} to the P_{used} slot. Next, UE decrypts the pseudonym and stores to the database as P_{new} for further use. Then, UE computes RES and sends it SN.

- SN receives RES from UE. Then, SN compares RES with XRES and notifies both UE and HN about the result.

- UE and HN receive result from SN. If the result is positive, then UE starts using the service. Otherwise, both UE and HN erase the new pseudonym from their databases. In this case, they need to put the pseudonym from P_{used} slot back to P_{new} slot.

- If UE wants to send used pseudonym to attach, UE sends P_{used} to SN.

- SN receives the attach request and sends it directly to HN.

- HN receives the pseudonym from SN. Then, HN checks the database and understands that the pseudonym is a pseudonym that has already been used at least once. Because the used pseudonym is sent for AKA, there is no need for assigning a new pseudonym or creating a new key. Therefore, HN prepares R3-type RAND, computes corresponding AUTN and XRES. Finally, HN sends $AV = RAND \parallel AUTN \parallel XRES$ to SN.

- SN receives AV from HN and keeps XRES to the database. Then, SN forwards RAND and AUTN to UE.

- UE receives a message from SN and extracts RAND and AUTN. First, UE checks MAC and SQN. If they both check out, UE checks AMF to understand the purpose of the RAND. Here, RAND is R3-type and does not serve for any specific purpose and it is just a random bit string. Then, UE computes RES and sends it SN.

- SN receives RES from UE. Then, SN compares RES with XRES and notifies both UE and HN about the result.

- UE and HN receive the result from SN. If the result is positive, UE starts using the services.

Next time UE wants to attach and start AKA, UE can choose between IMSI, P_{new}, and P_{used}. In this section, we have already explained how each choice affects the AKA session.

What happens after the authentication has succeeded is not implemented in the prototype. The prototype focuses on AKA procedure. That is why, in the demonstration, the UE can start from the beginning by sending identifier to SN, even after the authentication has just succeeded.

The demonstration could be stopped at any moment by the user, but the natural point to stop is when the authentication has succeeded.

Figure 17 summarizes how the prototype works.

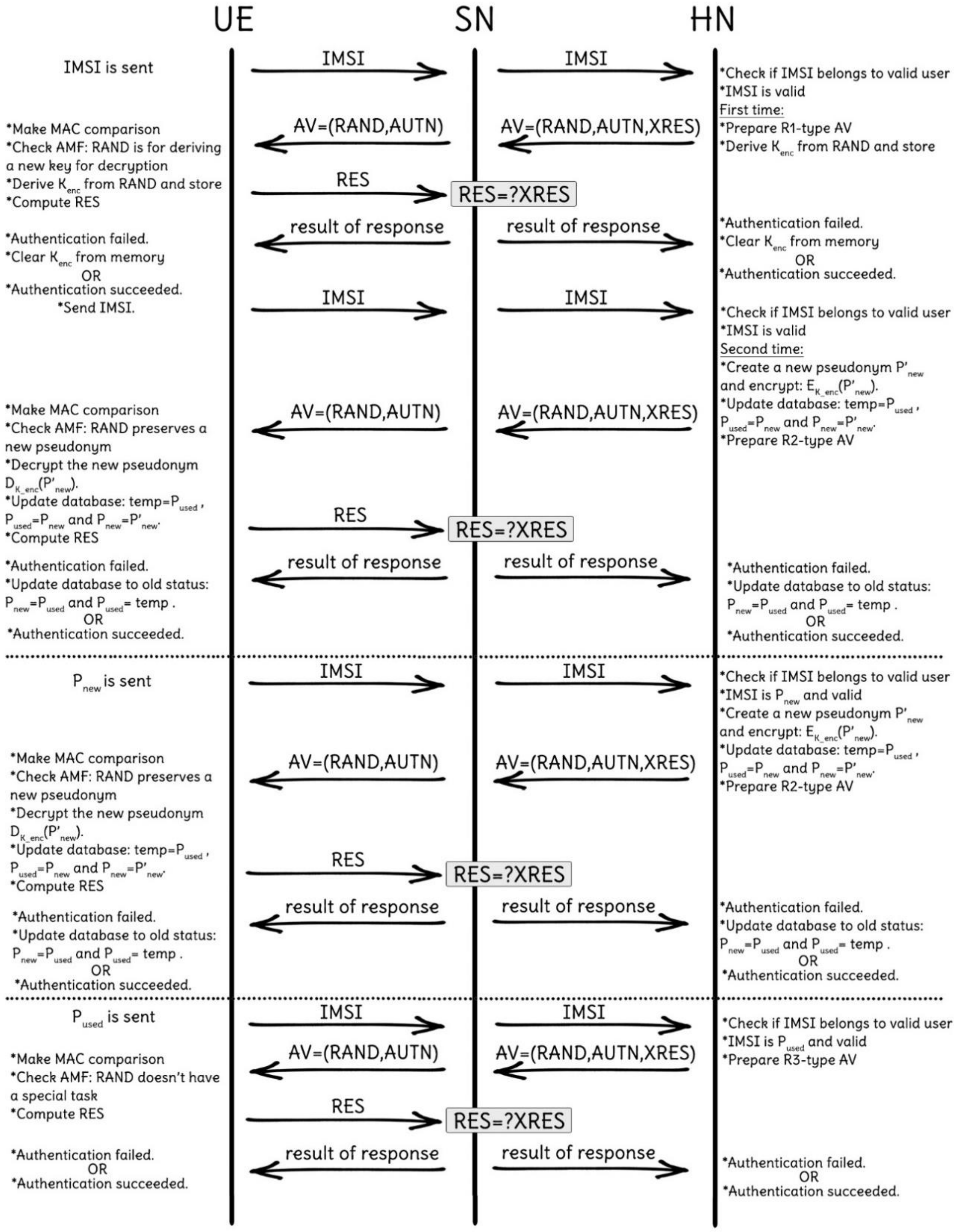


Figure 17: Authentication and Key Agreement stated in Prototype

9.2. User Interface

This implementation is designed for demonstrating authentication and key agreement with pseudonym-based approach for protection of identity privacy in 5G mobile networks. The communication occurs between the components, UE, SN, and HN. In the demonstrator, each AKA session starts with UE sending its identifier and ends with SN sending confirmation to both ends. We can call each AKA session a cycle, and the demonstrator allows performing more than one cycle. To be precise, the upper limit is 1000 cycles in the prototype, but this number does not represent anything specific in the real world.

The Demonstrator User, who runs the code, has some tasks to do during the demonstration. First of all, User should start by running the INPUT.java file to prepare the initial data (master key, IMSI, SQN, OP), that both UE and HN should possess from the beginning. Then, User should run UE.java, SN.java, and HN.java simultaneously. After that, User needs to jump between the windows and press enter to carry on the communication. However, this does not mean that User has the power to choose which component to go next. Each component knows if it is their turn or not. Therefore, if the User presses enter for the component whose turn did not come yet, then that component displays an error message and continues displaying this message until its turn comes. In the end, if User wants to leave the demonstrator, then User needs to write 'STOP' and press enter.

When the demonstration starts, UE is supposed to send its identifier to SN. The options for identifier are IMSI, new pseudonym, and used pseudonym. In this point, the User needs to act on behalf of UE and choose which identifier to send. Initially, UE has only IMSI recorded in the database. Therefore, if the User chooses something other than IMSI, UE displays error message and requests for new entry. Same error continues when the User chooses an identifier, which is not stored in the database yet.

9.3. Further Comments on Prototype

In the beginning of the demonstration, UE identifies itself with its IMSI in plaintext. It looks like it spoils the identity privacy. However, after UE and HN agrees on a pseudonym, UE will not need to use its IMSI again. Besides, since pseudonyms and IMSI look the same as a structure, an attacker would not be able to tell the difference between them and would not understand that IMSI and corresponding pseudonym represent same UE. However, if the synchronization gets lost between UE and HN, UE should use its IMSI as an identifier, when none of the pseudonyms are accepted by HN.

To avoid the connection being lost between UE and HN, both parties should be notified by SN in the end of AKA whether the authentication is successful or not. In 5G Phase 1, EPS AKA* includes informing HN about the result of authentication by sending RES back to HN. This might not be only way of letting HN know about the result, but a good start for informing both UE and HN. Moreover, if the confirmation does not arrive to both ends or it takes more time than usual (timer can be set), both UE and HN would not store the new pseudonym.

Despite informing both ends, there can still be problems against gaining the true pseudonym. There can be errors in one of the ends and pseudonym can be stored in database with a faulty bit. In this case, next time of attachment attempt, either UE will send false pseudonym or HN will not recognize true pseudonym. Therefore, UE will either try to send another recorded pseudonym, if such exists, or send IMSI for authentication. If the attacker realizes that the specific UE uses pseudonyms, then attacker would target UE until UE reveals its IMSI. However, it would not be so easy to realize if the identifier is IMSI or pseudonym, since they look alike. In this case, number of pseudonyms that are stored in database is an important issue. Eventually, having more than one pseudonym in the database would mitigate the problem.

In the prototype, only two consecutive pseudonyms are stored in the database. In real life, amount could change. This amount depends on the capacity of the database in HN. For each subscriber, HN should store at least 3 identifiers. Moreover, as the number of subscribers increase, the need of empty slot will increase, too. On the other hand, finding suitable pseudonym, which does not match with any existing IMSIs or pseudonyms, would get harder. Therefore, it is important for the operator to consider all advantages and disadvantages before deciding on the number of pseudonyms to store.

Pseudonyms are created randomly by random number generator in the prototype. Then, each number is checked through the database, to see if it is used before as an IMSI or as a pseudonym. Other than generating a random number, HN can come up with a proper function, which would create pseudonyms from previous pseudonyms, starting with IMSI. This helps keeping HN and UE synchronized, if either one of them loses synchronization, they can resynchronize again. However, there is a danger of generating pseudonym which belongs to another IMSI. Therefore, if some mechanism is designed for generating pseudonym, conflicts should be avoided by some mechanism.

Furthermore, SQN is the sequence number, which helps UE and HN to understand if they are synchronized. In the prototype, SQN is produced in INPUT class and kept as constant during whole demonstration. According to SQN's role in AKA, it was supposed to increase after each cycle. However, since the prototype includes only one user and forces all components to work, when it is their turn, there is no possibility for loss of synchronization. Moreover, SQN is necessary for most of the calculations. Summarizing, SQN is used in prototype as constant, unlike in the real life.

In the prototype, KASUMI cryptosystem is used for encrypting and decrypting the new pseudonym. KASUMI is preferred, because it is designed for 3GPP systems and is still considered as secure. However, in real life implementations or in any improvements of this prototype, any other block ciphers, other than KASUMI, can be used as well.

Procedure of encrypting the pseudonym and embedding it in RAND is not the only way to realize the pseudonym-based approach. The methods and paddings in the prototype can be changed for specific purposes. The procedure starts when MSIN of IMSI, which is 10-digit number, is converted into bits and becomes 40-bit array. Later, randomly generated 24 bits are padded to 40-bit pseudonym to have 64-bit array. This 64-bit array becomes the input for KASUMI encryption with the key K_{kasumi}. The ciphertext is again padded with two different and randomly generated 32-bit arrays from both left and right. So, the result is 128-bit of $random\ pad \parallel encrypted\ pseudonym \parallel random\ pad$ and this result becomes R2-type RAND.

Another issue with the prototype is about RAND. The purpose of using RAND, is to provide randomness in the calculations, so that attackers would not guess the following RAND and somehow use that in attacks. However, in the prototype, there are 3 types of RAND with different purposes. The first and third types of RAND are just random numbers, but the second type of RAND is not completely random. Encrypted pseudonym is embedded in the Type-2 RAND and concatenated with random numbers. Therefore, even though the RAND is not completely random, encrypted pseudonym looks random and does not reveal any information about the pseudonym itself. Thus, this situation does not harm the main purpose of RAND.

Finally, TMSI and K_{ASME} are not included in the prototype, because the aim of the prototype is to demonstrate pseudonym exchange during AKA. This prototype does not implement encryption and integrity protection. Therefore, after ending AKA, TMSI and K_{ASME} would be used in real life but not in the prototype. Moreover, AV that HN sends to SN normally includes CK and IK. Because of the same reasons stated here, these keys are not included in the prototype. SN does not need to use CK and IK during the prototype and UE can already compute CK and IK on its own.

9.4. Technical Details

All the program codes are written in JAVA language by using NetBeans IDE 8.2. The communication between networks (UE, SN, HN) are made by writing to and reading from .txt files.

All the codes are written by the author of this thesis. The code for one algorithm has been obtained from an existing library. This algorithm is HMAC, which can be found in METHODS.java. I have adapted it from a blog, see [51]. All the algorithms, other than HMAC, are implemented by the author according to the algorithm specifications from several sources, which will be specified in detail.

For generating key for KASUMI cryptosystem, I implemented the K_{ASME} derivation function by using the specifications from 3GPP TS 33.401 [35] and 3GPP TS 33.220 [52].

For implementing KASUMI cryptosystem, I used the specifications from 3GPP TS 35.202 [39]. I created specific functions in Java for the components and subfunctions of KASUMI. DivideFirst, DivideSecond, CircularLeftRotation, CircularRightRotation, ZE, TR, S7, S7_inv, S9, S9_inv, fi, FL, FL_inv, FI, FI_inv, FO, FO_inv, fi_odd, fi_odd_inv, fi_even, fi_even_inv are the functions that are created for implementing encryption and decryption of KASUMI cryptosystem. Moreover, KASUMI_enc is the code for KASUMI encryption and KASUMI_dec is for KASUMI decryption. All these listed functions can be found in METHODS.java. After writing the codes, I have tested the results by using test data from 3GPP TS 35.203 [53].

Advanced Encryption Standard 128-bit (AES-128) is preferred to be used in MILENAGE functions. Therefore, I implemented AES by using specifications from NIST (National Institute of Standards and Technology)'s publication [54] and explanations of algorithms from Kretzschmar's Application Report [55] about the implementation of AES-128. Some of the functions that are created for helping the implementation of AES are SBOX, ByteSubstitution, ShiftRow, T2, T3, MixColumn, AddRoundKey, and GenRoundKey. Finally, AES is the name of the function and all the listed functions can be found in METHODS.java file.

MILENAGE functions are used for generating certain elements for authentication and the key agreement. During the computation, MILENAGE uses a block cipher. This block cipher is chosen to be AES-128 by 3GPP [30]. The end products from MILENAGE functions are MAC, RES, CK, IK, and AK. Therefore, each of them has its own function in METHODS.java, so that they can be called any time it is necessary by HN or UE. I have implemented these functions by using the specifications from 3GPP TS 35.206 [30] and tested, whether they work or not, from 3GPP TS.35.207 [56].

There are some functions in METHODS.java file, which are already existed in java, such as AND, OR, XOR, CopyArray functions. However, I decided to rewrite them in more explicit way, which became more convenient and easy for me. Moreover, random() function creates array of 0 and 1, by calling SecureRandom() class. I aimed to use Java's random creator securely.

Conclusions

Mobile networks are in the center of people's lives through smart phones, tablets, and even computers. Therefore, these devices dominate a huge portion of the users' life. Besides the information that the user provides willingly, some details are needed to be kept away from the irrelevant companies. For example, the real identity and the location of the user are not supposed to be known by anyone other than home network, which needs this information in order to provide proper service according to the subscription. Identity privacy in mobile networks aims to keep sensitive information, such as real identity and location of the user, away from third parties. In this thesis, it is discussed how to provide identity privacy in 5G network.

This thesis starts with the explanation of the evolution of mobile networks. Then, Authentication and Key Agreement (AKA), which is a crucial process to provide authenticity and integrity, is described in all generations. A cryptosystem, which is called KASUMI, is explained in this thesis. KASUMI is designed to be used in encryption and integrity protection in mobile networks. Therefore, KASUMI can also be a part of 5G network. Then, existing ideas and approved decisions about the structure of 5G and 5G AKA are explained in detail. Thereupon, an idea for providing identity privacy is introduced. This idea involves using pseudonym instead of real identifier and could be adapted to the AKA in 5G. Even though, encrypting the real identifier with public key is accepted for 5G Phase 1, the two methods are compared and discussed in the thesis. In the end, a prototype is introduced, which presents pseudonym approach. The implementation of this prototype has been done by using Java. The prototype does not include all components of AKA. The main idea behind the prototype is presenting a possible way of creating pseudonyms and placing them in the components of AKA. If necessary, enhancements for the prototype can be done according to what are accepted for 5G AKA in the standardization.

For the future improvements of 5G, there is a need for an alternative or an additional method to public key approach for identity privacy. In case the user cannot connect to 5G network, the connection automatically falls to 4G, 3G, or 2G, in order to provide service to the user. However, this situation brings the privacy issues back to the surface. Since public key approach does not work for the networks other than 5G, then the user will need to use the real identifier and the identity privacy of the user would be put in risk. Thus, this situation creates an open door for attackers to exploit. IMSI catchers can convince the user that 5G is not available and force the user to fall back to other generations. Therefore, public key approach is a good solution for sustaining security and identity privacy, but not enough with older networks. In order to expect for high level of privacy, the other mobile networks should first be eliminated. However, this situation might take a long time. Therefore, until then, pseudonym approach can be introduced and be an efficient solution to protect identity privacy.

References

[1] "History of Communication from Cave Drawings to the Web", *Creative Displays Now*. [Online]. Available: https://www.creativedisplaysnow.com/articles/history-of-communication-from-cave-drawings-to-the-web/. [Accessed: 22- Mar- 2018].

[2] "World Communication Development Report 1999", *ITU*, 1999. [Online]. Available: https://www.itu.int/ITU-D/ict/publications/wtdr_99/material/wtdr99s.pdf. [Accessed: 03- Apr- 2018].

[3] "Unique Mobile Subscribers Worldwide 2010-2020", *Statista*, 2017. [Online]. Available: https://www.statista.com/statistics/371780/unique-mobile-subscribers-worldwide-from-2008/. [Accessed: 03- Apr- 2018].

[4] "Number of Mobile Subscribers Worldwide Hits 5 Billion", *GSMA*, 2017. [Online]. Available: https://www.gsma.com/newsroom/press-release/number-mobile-subscribers-worldwide-hits-5-billion/. [Accessed: 03- Apr- 2018].

[5] F. Cohen, "A Short History of Cryptography", *All.net*, 1995. [Online]. Available: http://www.all.net/books/IP/Chap2-1.html. [Accessed: 23- Mar- 2018].

[6] Dictionary.com, "cryptography," in *Dictionary.com*. Available: http://www.dictionary.com/. Accessed: March 23, 2018.

[7] F. van den Broek, R. Verdult and J. de Ruiter, "Defeating IMSI Catchers", in CCS '15 - Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, USA, 2015, pp. 340-351.

[8] "Mobile Network Codes (MNC) for the international identification plan for public networks and subscriptions", ITU, 2014. [Online]. Available: https://www.itu.int/dms_pub/itu-t/opb/sp/T-SP-E.212B-2014-PDF-E.pdf. [Accessed: 20- Aug- 2016].

[9] L. McLeary, "The Difference Between IMSI and MSISDN", *Techwalla.com*, 2015. [Online]. Available: https://www.techwalla.com/articles/the-difference-between-imsi-and-msisdn. [Accessed: 29- Mar- 2018].

[10] "International Roaming Explained", *Gsma.com*, 2012. [Online]. Available: https://www.gsma.com/latinamerica/wp-content/uploads/2012/08/GSMA-Mobile-roaming-web-English.pdf . [Accessed: 27- Mar- 2018].

[11] T. Damico, "A Brief History of Cryptography", *Inquiries Journal*, 2009. [Online]. Available: http://www.inquiriesjournal.com/articles/1698/a-brief-history-of-cryptography. [Accessed: 23- Mar- 2018].

[12] A. Gupta and R. Jha, "A Survey of 5G Network: Architecture and Emerging Technologies", IEEE Access, vol. 3, pp. 1206-1232, 2015.

[13] A. Kumar, Y. Liu and J. Sengupta, "Evolution of Mobile Wireless Communication Networks: 1G to 4G", IJECT, vol. 1, no. 1, pp. 68-72, 2010.

[14] D. Forsberg, W. Moeller, V. Niemi and G. Horn, LTE security. Wiley, 2010.

[15] I. Akyildiz, D. Gutierrez-Estevez, R. Balakrishnan and E. Chavarria-Reyes, "LTE-Advanced and the evolution to Beyond 4G (B4G) systems", Physical Communication, vol. 10, pp. 31-60, 2014.

[16] D. Hutton, "Five Things You Wanted to Know about 5G, But Never Dared to Ask - Future Networks", Future Networks, 2016. [Online]. Available: https://www.gsma.com/futurenetworks/digest/five-things-wanted-know-5g-never-dared-ask/. [Accessed: 20- Nov- 2017].

[17] "SK Telecom and Samsung Complete 5G End-to-End Network Trial Based on 3.5GHz 5G New Radio (NR) Technologies", Samsung Newsroom, 2017. [Online]. Available: https://news.samsung.com/global/sk-telecom-and-samsung-complete-5g-end-to-end-network-trial-based-on-3-5ghz-5g-new-radio-nr-technologies. [Accessed: 20- Nov- 2017].

[18] Y. Park and T. Park, "A Survey of Security Threats on 4G Networks", 2007 IEEE Globecom Workshops, 2007.

[19] D. Bhasker, "4G LTE Security for Mobile Network Operators", Journal of Cyber Security and Information Systems, pp. 20-29, 2013.

[20] A. Bikos and N. Sklavos, "LTE/SAE Security Issues on 4G Wireless Networks", IEEE Security & Privacy, vol. 11, no. 2, pp. 55-62, 2013.

[21] N. Seddigh, B. Nandy, R. Makkar and J. Beaumont, "Security Advances and Challenges in 4G Wireless Networks", in Eighth Annual International Conference on Privacy, Security and Trust, Ottawa, Canada, 2010, pp. 62-71.

[22] K. Norrman and P. Nakarmi, "Protecting 5G Against IMSI Catchers", Ericsson Research Blog, 2017. [Online]. Available: https://www.ericsson.com/research-blog/protecting-5g-imsi-catchers/. [Accessed: 14- Sep- 2017].

[23] K. Norrman, M. Näslund and E. Dubrovq, "Protecting IMSI and User Privacy in 5G Networks", in MobiMedia '16 - Proceedings of the 9th EAI International Conference on Mobile Multimedia Communications, Xi'an, China, 2016, pp. 159-166.

[24] D. Strobel, "IMSI Catcher", IT-Sicherheit Seminar 2007, 2007. [Online]. Available: https://www.emsec.rub.de/media/attachments/files/2011/11/imsi_catcher_update.pdf. [Accessed: 14- Sep- 2017].

[25] P. Ginzboorg and V. Niemi, "Privacy of the Long-term Identities in Cellular Networks", in MobiMedia '16 - Proceedings of the 9th EAI International Conference on Mobile Multimedia Communications, Xi'an, China, 2016, pp. 167-175.

[26] A. Dabrowski, N. Pianta, T. Klepp, M. Mulazzani and E. Weippl, "IMSI-catch me if you can: IMSI-catcher-catchers", in ACSAC '14 - Proceedings of the 30th Annual Computer Security Applications Conference, New Orleans, USA, 2014, pp. 246-255.

[27] R. Gallagher, "Meet the machines that steal your phone’s data", Ars Technica, 2013. [Online]. Available: https://arstechnica.com/tech-policy/2013/09/meet-the-machines-that-steal-your-phones-data/. [Accessed: 11-Oct- 2017].

[28] Langston, "Catching the IMSI-catchers: SeaGlass brings transparency to cell phone surveillance", UW News, 2017. [Online]. Available: http://www.washington.edu/news/2017/06/02/catching-the-imsi-catchers-seaglass-brings-transparency-to-cell-phone-surveillance/. [Accessed: 14- Sep- 2017].

[29] K. Prakash, "Authentication and Key Agreement in 3GPP Networks", in Fifth International Conference on Advances in Computing and Information Technology, Chennai, India, 2015, pp. 143-154.

[30] 3GPP TS 35.206 version 13.0.0 (2016): “Specification of the MILENAGE Algorithm Set: An example algorithm set for 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*; Document 2: Algorithm Specification”.

[31] 3GPP TS 23.002 version 14.1.0 (2017): “Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); LTE; Network architecture”.

[32] S. Palat and P. Godin, "Network Architecture", in LTE – The UMTS Long Term Evolution: From Theory to Practice, 1st ed., S. Sesia, I. Toufik and M. Baker, Ed. John Wiley & Sons, Ltd, 2009, pp. 21-50.

[33] A. Shaik, R. Borgaonkar, N. Asokan, V. Niemi and J. Seifert, "Practical Attacks Against Privacy and Availability in 4G/LTE Mobile Communication Systems", in Network and Distributed System Security Symposium, San Diego, United States, 2016.

[34] D. Lanzenberger, "Formal Analysis of 5G Protocols (Bachelor Thesis)", ETHZ - Swiss Federal Institute of Technology Zurich, 2017. [Online]. Available: https://www.ethz.ch/content/dam/ethz/special-interest/infk/inst-infsec/information-security-group-dam/research/software/5G_lanzenberger.pdf. [Accessed: 09- Dec- 2017].

[35] 3GPP TS 33.401 version 13.3.0 (2016): "3GPP System Architecture Evolution (SAE); Security architecture".

[36] M. Matsui and T. Tokita, "MISTY, KASUMI and Camellia Cipher Algorithm Development", Mitsubishi Electric ADVANCE, vol. 100, pp. 2-4, 2002.s

[37] K. Jia, L. Li, C. Rechberger, J. Chen and X. Wang, "Improved Cryptanalysis of the Block Cipher KASUMI", in SAC 2012 - International Conference on Selected Areas in Cryptography, Windsor, Canada, 2012, pp. 222-233.

[38] E. Biham, O. Dunkelman and N. Keller, "A Related-Key Rectangle Attack on the Full KASUMI", in ASIACRYPT 2005 - 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, 2005, pp. 443-461.

[39] 3GPP TS 35.202 version 13.0.0 (2016): "3G Security; Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 2: KASUMI Specification".

[40] 3GPP TS 33.501 version 0.6.0 (2017): "Security Architecture and Procedures for 5G System".

[41] X. Zang, A. Kunz and S. Schröder, "Overview of 5G security in 3GPP", in IEEE Conference on Standards for Communications & Networking, Helsinki, Finland, 2017, pp. 1-6.

[42] P. Rost, A. Banchs, I. Berberana, M. Breitbach, M. Doll, H. Droste, C. Mannweiler, M. Puente, K. Samdanis and B. Sayadi, "Mobile network architecture evolution toward 5G", IEEE Communications Magazine, vol. 54, no. 5, pp. 84-91, 2016.

[43] 3GPP TS 23.501 version 15.0.0 (2017): "System Architecture for the 5G System; Stage 2".

[44] 3GPP TS 33.899 version 1.3.0 (2017): "Study on the Security Aspects of the Next Generation System".

[45] 3GPP TS 33.821 version 9.0.0 (2009): "Rationale and track of security decisions in Long Term Evolved (LTE) RAN / 3GPP System Architecture Evolution (SAE)".

[46] C. Gentry, "Certificate-Based Encryption and the Certificate Revocation Problem", in *EUROCRYPT'03 Proceedings of the 22nd International Conference on Theory and Applications of Cryptographic Techniques*, Warsaw, Poland, 2003, pp. 272-293.

[47] M. Khan and V. Niemi, "Concealing IMSI in 5G Network Using Identity Based Encryption", in Network and System Security: 11th International Conference, NSS 2017, Helsinki, Finland, 2017, pp. 544-554.

[48] E. Jiménez, P. Nakarmi, M. Näslund and K. Norrman, "Subscription identifier privacy in 5G systems", in 2017 International Conference on Selected Topics in Mobile and Wireless Networking (MoWNeT), Avignon, France, 2017, pp. 1-8.

[49] R. Schulz, "RSA vs. ECC", 2015. [Online]. Available: http://page.mi.fu-berlin.de/rhschulz/Krypto/RSA_or_ECC.pdf. [Accessed: 12- Feb- 2018].

[50] M. Kaushik, "5G not backward compatible; we need new spectrum, devices: Ericsson's Joakim Sorelius", Businesstoday.in, 2017. [Online]. Available: http://www.businesstoday.in/opinion/interviews/5g-not-backward-compatible-we-need-new-spectrum-devices-ericssons-joakim-sorelius/story/251668.html. [Accessed: 01- Feb- 2018].

[51] K. Tan, "Generating HMAC MD5/SHA1/SHA256 etc in Java", Supermind.org, 2012. [Online]. Available: http://www.supermind.org/blog/1102/generating-hmac-md5-sha1-sha256-etc-in-java. [Accessed: 01- Aug- 2016].

[52] 3GPP TS 33.220 version 13.0.0 (2016): "Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture (GBA)".

[53] 3GPP TS 35.203 version 13.0.0 (2016): “3G Security; Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 3: Implementors’ Test Data”.

[54] National Institute of Standards and Technology, "Specification for the ADVANCED ENCRYPTION STANDARD (AES)", Federal Information Processing Standards Publication 197, 2001.

[55] U. Kretzschmar, "AES128 – A C Implementation for Encryption and Decryption", Ti.com, 2009. [Online]. Available: http://www.ti.com/lit/an/slaa397a/slaa397a.pdf. [Accessed: 05- Aug- 2016].

[56] 3GPP TS 35.207 version 13.0.0 (2016): “Specification of the MILENAGE Algorithm Set: An example algorithm set for 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*; Document 3: Implementors’ Test Data”.

APPENDIX A – Source Code

INPUT, UE, SN, and HN are presented here. METHODS file can be obtained separately.

A.1. INPUT.java

```
import java.io.IOException;

public class INPUT extends METHODS{

  public static void main(String[] args) throws IOException{

//Random and initial inputs for both UE and HN
//K, OP, IMSI
    int[] K_MASTER=random(128);
    int[] OP=random(128);
    int[] IMSI=rand_number(10);
    int[] IMSI_bit=PseudoToBits(IMSI);

      WriteFileHex("K_MASTER_hex.txt", K_MASTER);
      System.out.println("Key_hex is: "+ReadFileString("K_MASTER_hex.txt"));

      WriteFileHex("OP_hex.txt", OP);
      System.out.println("OP_hex is: "+ReadFileString("OP_hex.txt"));

      WriteFileHex("IMSI_hex.txt", IMSI_bit);
      System.out.println("IMSI is: "+ReadFileString("IMSI_hex.txt"));

//SQN should be synced between UE and HN
    int[] SQN=random(48);
      WriteFileHex("SQN_hex.txt", SQN);
      System.out.println("SQN is: "+ReadFileString("SQN_hex.txt"));

//Checkpoints tells each UE, SN, and HN, if it is their turn to continue.
//Always UE starts the process so checkpoint0 is 1 and the others are 0.
      WriteFileInt("checkpoint0.txt",1);
      WriteFileInt("checkpoint1.txt",0);
      WriteFileInt("checkpoint2.txt",0);
      WriteFileInt("checkpoint3.txt",0);
      WriteFileInt("checkpoint4.txt",0);
      WriteFileInt("checkpoint5.txt",0);
      WriteFileInt("checkpoint6.txt",0);
      WriteFileInt("checkpoint7.txt",0);
```

```java

        System.out.println("Checkpoints are ready.");
    }
}

```

A.2. UE.java

```java
import java.io.IOException;
import java.util.Scanner;


public class UE extends METHODS{

    public static void main(String[] args) throws IOException {
        Scanner scan = new Scanner (System.in);

//db[0]=IMSI
//db[1]=new pseudonym
//db[2]=used pseudonym
//db[3]=K_MASTER
//db[4]=OP
//db[5]=SQN

        String[] db=new String[6]; //database of UE

        db[0]=ReadFileString("IMSI_hex.txt");
        db[3]=ReadFileString("K_MASTER_hex.txt");
        db[4]=ReadFileString("OP_hex.txt");
        db[5]=ReadFileString("SQN_hex.txt");
        db[1]="";
        db[2]="";

        int x=0;
            System.out.println("----------|User Equipment|----------");
            System.out.println("---------------|UE|---------------\n");

        do{
            if(ReadFileInt("checkpoint0.txt")==1){
                proceed();

            while(ReadFileInt("checkpoint0.txt")==1){

//UE needs to choose between IMSI and pseudonym
//Each choice requires different calculations afterwards
                System.out.println("Choose what to send for ID:");
                System.out.println("Write 'P1' to send IMSI");
```

```
            System.out.println("Write 'P2' to send new pseudonym");
            System.out.println("Write 'P3' to send used pseudonym");

   //If the pseudonyms are not stored yet, it needs to ask again
            String msg=scan.nextLine();
              if(msg.equalsIgnoreCase("stop")||msg.equalsIgnoreCase("no")){
                System.exit(0);}

            if(msg.equalsIgnoreCase("p1")){
              WriteFileInt("checkpoint0.txt",0);
              WriteFileString("msg.txt",msg);}
            else if(msg.equalsIgnoreCase("p2")&& !(db[1].isEmpty())){
              WriteFileInt("checkpoint0.txt",0);
              WriteFileString("msg.txt",msg);}
            else if(msg.equalsIgnoreCase("p3")&& !(db[2].isEmpty())){
              WriteFileInt("checkpoint0.txt",0);
              WriteFileString("msg.txt",msg);}
            else if(msg.equalsIgnoreCase("p3")&& (db[2].isEmpty())){
              System.out.println("|UE| The pseudonym doesn't exist. Please try P1 or P2.\n");
              System.out.println("----------------------------------------"+"\n");}
            else if(msg.equalsIgnoreCase("p2")&& db[1].isEmpty()){
              System.out.println("|UE| The pseudonym doesn't exist. Please try P1.\n");
              System.out.println("----------------------------------------"+"\n");}
            else{
              System.out.println("|UE| Invalid choice. Please try again.\n");
              System.out.println("----------------------------------------"+"\n");}
          }}
          else if(ReadFileInt("checkpoint0.txt")==0){
            String msg=ReadFileString("msg.txt");
            System.out.println();

 //UE chooses to send IMSI
          if(msg.equalsIgnoreCase("p1")){
              part1(db,msg);

            System.out.println("|UE| Attachment request is sent to SN.");
            System.out.println("\n"+"----------------------------------------"+"\n");
              WriteFileInt("checkpoint1.txt",1);

            proceed();
            if(ReadFileInt("checkpoint4.txt")==0){
              do{
                System.out.println("|UE| There isn't a new file yet. Try again later.");
                System.out.println("\n"+"----------------------------------------"+"\n");
                proceed();
              } while(ReadFileInt("checkpoint4.txt")==0);}
            else if(ReadFileInt("checkpoint4.txt")==1){}
              WriteFileInt("checkpoint4.txt",0);

 //HN sends RAND and AUTN through SN, which are components of Authentication and Key Agreement
```

```java
            System.out.println("|UE| AV is received from SN. \n");

//UE makes necessary calculations with RAND and AUTN
              db=part2(db);

            System.out.println("|UE| RES is sent to SN.");
            System.out.println("\n"+"----------------------------------------"+"\n");
              WriteFileInt("checkpoint5.txt",1);

            proceed();
            if(ReadFileInt("checkpoint7.txt")==0){
              do{
                System.out.println("|UE| There isn't a new file yet. Try again later.");
                System.out.println("\n"+"----------------------------------------"+"\n");
                proceed();
              } while(ReadFileInt("checkpoint7.txt")==0);}
            else if(ReadFileInt("checkpoint7.txt")==1){}
              WriteFileInt("checkpoint7.txt",0);

//Calculates RES that is necessary for sending to SN
            res_respond(db);
            System.out.println("\n"+"----------------------------------------"+"\n");

//After UE recieves key from HN, also need a pseudonym
//Triggers automatically to start another authentication
              msg="p1";
              part1(db,msg);

            System.out.println("|UE| Attachment request is sent to SN.");
            System.out.println("\n"+"----------------------------------------"+"\n");
              WriteFileInt("checkpoint1.txt",1);

            proceed();
            if(ReadFileInt("checkpoint4.txt")==0){
              do{
                System.out.println("|UE| There isn't a new file yet. Try again later.");
                System.out.println("\n"+"----------------------------------------"+"\n");
                proceed();
              } while(ReadFileInt("checkpoint4.txt")==0);}
            else if(ReadFileInt("checkpoint4.txt")==1){}
              WriteFileInt("checkpoint4.txt",0);

//HN sends RAND and AUTN through SN, which are components of Authentication and Key Agreement
            System.out.println("|UE| AV is received from SN. \n");

//UE makes necessary calculations with RAND and AUTN
              db=part2(db);

            System.out.println("|UE| RES is sent to SN.");
            System.out.println("\n"+"----------------------------------------"+"\n");
              WriteFileInt("checkpoint5.txt",1);
```

```
            proceed();
            if(ReadFileInt("checkpoint7.txt")==0){
              do{
                System.out.println("|UE| There isn't a new file yet. Try again later.");
                System.out.println("\n"+"-----------------------------------------"+"\n");
                proceed();
              } while(ReadFileInt("checkpoint7.txt")==0);}
            else if(ReadFileInt("checkpoint7.txt")==1){}
              WriteFileInt("checkpoint7.txt",0);

//Calculates RES that is necessary for sending to SN
            res_respond(db);
            System.out.println("\n"+"-----------------------------------------"+"\n");
              WriteFileInt("checkpoint0.txt",1);
          }

//UE chooses to send used pseudonym
          else if(msg.equalsIgnoreCase("p2")){
              part1(db,msg);

            System.out.println("|UE| Attachment request is sent to SN.");
            System.out.println("\n"+"-----------------------------------------"+"\n");
              WriteFileInt("checkpoint1.txt",1);

            proceed();
            if(ReadFileInt("checkpoint4.txt")==0){
              do{
                System.out.println("|UE| There isn't a new file yet. Try again later.");
                System.out.println("\n"+"-----------------------------------------"+"\n");
                proceed();
              } while(ReadFileInt("checkpoint4.txt")==0);}
            else if(ReadFileInt("checkpoint4.txt")==1){}
              WriteFileInt("checkpoint4.txt",0);

//HN sends RAND and AUTN through SN, which are components of Authentication and Key Agreement
            System.out.println("|UE| AV is received from SN. \n");

//UE makes necessary calculations with RAND and AUTN
              db=part2(db);

            System.out.println("|UE| RES is sent to SN.");
            System.out.println("\n"+"-----------------------------------------"+"\n");
              WriteFileInt("checkpoint5.txt",1);

            proceed();
            if(ReadFileInt("checkpoint7.txt")==0){
              do{
                System.out.println("|UE| There isn't a new file yet. Try again later.");
                System.out.println("\n"+"-----------------------------------------"+"\n");
                proceed();
```

```
        } while(ReadFileInt("checkpoint7.txt")==0);}
      else if(ReadFileInt("checkpoint7.txt")==1){}
        WriteFileInt("checkpoint7.txt",0);

//Calculates RES that is necessary for sending to SN
      res_respond(db);
      System.out.println("\n"+"----------------------------------------"+"\n");
        WriteFileInt("checkpoint0.txt",1);
    }

//UE chooses to send new pseudonym
    else if(msg.equalsIgnoreCase("p3")){
        part1(db,msg);

      System.out.println("|UE| Attachment request is sent to SN.");
      System.out.println("\n"+"----------------------------------------"+"\n");
        WriteFileInt("checkpoint1.txt",1);

      proceed();
      if(ReadFileInt("checkpoint4.txt")==0){
        do{
          System.out.println("|UE| There isn't a new file yet. Try again later.");
          System.out.println("\n"+"----------------------------------------"+"\n");
          proceed();
        } while(ReadFileInt("checkpoint4.txt")==0);}
      else if(ReadFileInt("checkpoint4.txt")==1){}
        WriteFileInt("checkpoint4.txt",0);

//HN sends RAND and AUTN through SN, which are components of Authentication and Key Agreement
      System.out.println("|UE| AV is received from SN. \n");

//UE makes necessary calculations with RAND and AUTN
        db=part2(db);

      System.out.println("|UE| RES is sent to SN.");
      System.out.println("\n"+"----------------------------------------"+"\n");
        WriteFileInt("checkpoint5.txt",1);

      proceed();
      if(ReadFileInt("checkpoint7.txt")==0){
        do{
          System.out.println("|UE| There isn't a new file yet. Try again later.");
          System.out.println("\n"+"----------------------------------------"+"\n");
          proceed();
        } while(ReadFileInt("checkpoint7.txt")==0);}
      else if(ReadFileInt("checkpoint7.txt")==1){}
        WriteFileInt("checkpoint7.txt",0);

//Calculates RES that is necessary for sending to SN
      res_respond(db);
      System.out.println("\n"+"----------------------------------------"+"\n");
```

```
            WriteFileInt("checkpoint0.txt",1);
        }}
    }while (x<1000);
  }

//According to the input, proper element from database is written into the file
  public static void part1(String[] db, String message) throws IOException{
    String send="";
    if(message.equalsIgnoreCase("p1")){ //p1=IMSI
      send=db[0];}
    else if(message.equalsIgnoreCase("p2")){ //p2=new pseudonym
      send=db[1];}
    else if(message.equalsIgnoreCase("p3")){ //p3=used pseudonym
      send=db[2];}

    System.out.println("|UE| IMSI: DNA " + send);
    WriteFileString("UE_IMSI.txt",send);
  }

//UE performs Authentication and key agreement in this part. If exists, pseudonym is extracted
  public static String[] part2(String[] db) throws IOException{

    int[] AV_UE=HexToBinary(ReadFileString("AV_toUE.txt"));
    int[] RAND_UE=new int[128], AUTN_UE=new int[128];
    int[] AMF_UE=new int[16], MAC_UE_ext=new int[64];
    int[] RES_UE, MAC_UE;
    int[] K_MASTER,OP,SQN;
    K_MASTER=HexToBinary(db[3]);
    OP=HexToBinary(db[4]);
    SQN=HexToBinary(db[5]);

    System.out.println("|UE| Extracting RAND and AUTN..");

    CopyArray(AV_UE,RAND_UE,0,0,128);
    CopyArray(AV_UE,AUTN_UE,128,0,128);

    System.out.println("|UE| RAND and AUTN are extracted.\n");

    System.out.println("|UE| Extracting and calculating MAC.");
    System.out.println("|UE| Checking MAC..");

    CopyArray(AUTN_UE,MAC_UE_ext,64,0,64);
    CopyArray(AUTN_UE,AMF_UE,48,0,16);

    MAC_UE=MAC(RAND_UE,K_MASTER,OP,SQN,AMF_UE);
    String tmp=Compare(MAC_UE_ext,MAC_UE);
    if(tmp.equals("same"))
      System.out.println("|UE| MAC is verified."+"\n");
    else{
      System.out.println("|UE| MAC is not verified."+"\n");
      System.exit(0);}
```

```
    System.out.println("|UE| Extracting AMF..");
    System.out.println("|UE| AMF is extracted.\n");

    System.out.println("|UE| Checking AMF..");
    String temp;

//Process continues differently according to the value stored in AMF
    int amfue=CheckAMF(AMF_UE);
    WriteFileInt("AMF_result.txt",amfue);

//AMF shows that new pseudonym is stored in RAND.
    if(amfue==2){
      int[] RAND_KEY=ReadFile("RAND_key.txt",128);
      int[] AMF_KEY=ReadFile("AMF Key.txt",16);

      System.out.println("|UE| Extracting Pseudonym..");

//Encrypted pseudonym is extracted from RAND
      int[] PSEUDO_ENCRYPTED=new int[64],PSEUDO_DECRYPTED, PSEUDONYM_bit=new int[40];
      CopyArray(RAND_UE, PSEUDO_ENCRYPTED, 32, 0, 64);
      int[] KeyKasumi=KeyKasumi(RAND_KEY,K_MASTER,OP,SQN,AMF_KEY);
//Extracted and encrypted pseudonym is decrypted
      PSEUDO_DECRYPTED=KASUMI_dec(PSEUDO_ENCRYPTED,KeyKasumi);
      CopyArray(PSEUDO_DECRYPTED,PSEUDONYM_bit,0,0,40);
      String s=BinaryToHex(PSEUDONYM_bit);
      System.out.println("|UE| Pseudonym is extracted.");
      System.out.println("|UE| Pseudonym is "+s+"\n");
      int[] PSEUDONYM=PseudoHexToBit(s);
      WriteFile("Pseudo_extracted.txt",PSEUDONYM);
//New pseudonym is recorded to the database for further uses
      temp=db[2];
      db[2]=db[1];
      db[1]=BinaryToText(PSEUDONYM);
      WriteFileString("temp.txt",temp);}
//AMF shows that RAND will be used as K_kasumi for decrypting pseudonym
    else if(amfue==1){}
//AMF shows that RAND doesn't have a function in this round
    else if(amfue==3){}
//If AMF doesn't include the proper information, then it is not authentic.
    else
      System.exit(0);

    System.out.println("|UE| Preparing RES..");

//RES will let SN know that UE is valid user
    RES_UE=RES(RAND_UE,K_MASTER,OP,SQN,AMF_UE);
    WriteFileHex("RES_UE.txt",RES_UE);

    System.out.println("|UE| RES is prepared.\n");
```

```
    return db;
  }

//It prepares the RES that is needed to be sent to SN
  public static void res_respond(String[] db) throws IOException{
      System.out.println("|UE| Result for RES challenge is received from SN.\n");
      System.out.println("|UE| Checking result..");

    String snres=ReadFileString("SN_RES_result.txt");
    int amf_type=ReadFileInt("AMF_result.txt");
    if(snres.equalsIgnoreCase("valid")){
      System.out.println("|UE| Authentication succeeded.");}
    else if(snres.equalsIgnoreCase("invalid")){
      System.out.println("|UE| Authentication failed.");
      if (amf_type==2){
        String temp=ReadFileString("temp.txt");
        db[1]=db[2];
        db[2]=temp;
        System.out.println("|UE| Database is corrected.");
      }
    }
  }

//AMF includes information about the usage of RAND
  public static int CheckAMF(int[] AMF){
    int x=0;

    if(AMF[1]==1 & AMF[2]==0){
      System.out.println("|UE| This RAND is to be used for creating new key.\n");
      x=1;}
    else if(AMF[1]==0 & AMF[2]==1){
      System.out.println("|UE| This RAND includes pseudonym.\n");
      x=2;}
    else if(AMF[1]==0 & AMF[2]==0){
      System.out.println("|UE| This RAND doesn't include pseudonym and isn't to be used for creating new key.\n");
      x=3;}

    return x;
  }

//Creates Key for Kasumi encryption
  public static int[] KeyKasumi(int[] RAND, int[] K_MASTER, int[] OP, int[] SQN, int[] AMF) throws IOException{

    int[] CK, IK, AK;
    CK=CK(RAND, K_MASTER, OP, SQN, AMF);
    IK=IK(RAND, K_MASTER, OP, SQN, AMF);
    AK=AK(RAND, K_MASTER, OP, SQN, AMF);

    int[] K,S;
    int[] FC, P0, L0, P1, L1;
```

```
        int[] oct1, oct2, oct3;

        K=Concatenate(CK, IK);
        FC=HexToBinary("60");

//For MCC and MNC, I will use DNA Oy, which is 244 12.
        oct1=Concatenate(HexToBinary("4"),HexToBinary("2"));
        oct2=Concatenate(HexToBinary("f"),HexToBinary("4"));
        oct3=Concatenate(HexToBinary("2"),HexToBinary("1"));
        P0=Concatenate(Concatenate(oct1,oct2),oct3);
        L0=Concatenate(HexToBinary("000"),HexToBinary("3"));
        P1=XOR(SQN, AK);
        L1=Concatenate(HexToBinary("000"),HexToBinary("6"));

// S=FC || P0 || LO || P1 || L1.
        S=Concatenate(Concatenate(Concatenate(Concatenate(FC,P0),L0),P1),L1);

        int[] KEY_kasumi=HMAC(S,K);

        return KEY_kasumi;
    }

}
```

A.3. SN.java

```
import java.io.IOException;

public class SN extends METHODS{

  public static void main(String[] args) throws IOException {

//db[x][y]
//x=order of the user. for every new user, x+1
//y=0 == IMSI, y=1 == XRES
      String db[][]=new String[1000][2];
      db[0][0]="IMSI";
      db[0][1]="XRES";
   int x=1;

        System.out.println("----------|Serving Network|----------");
        System.out.println("----------------|SN|----------------\n");

      do{
        proceed();
        if(ReadFileInt("checkpoint1.txt")==0){
          do{
```

```
            System.out.println("|SN| There isn't a new file yet. Try again later.");
            System.out.println("\n"+"----------------------------------------"+"\n");
            proceed();
          } while(ReadFileInt("checkpoint1.txt")==0);}
        else if(ReadFileInt("checkpoint1.txt")==1){}
          WriteFileInt("checkpoint1.txt",0);

//Attach attempt from UE
          part1(db,x);

        System.out.println("|SN| Attachment request is sent to HN.");
        System.out.println("\n"+"----------------------------------------"+"\n");
          WriteFileInt("checkpoint2.txt",1);

        proceed();
        if(ReadFileInt("checkpoint3.txt")==0){
          do{
            System.out.println("|SN| There isn't a new file yet. Try again later.");
            System.out.println("\n"+"----------------------------------------"+"\n");
            proceed();
          } while(ReadFileInt("checkpoint3.txt")==0);}
        else if(ReadFileInt("checkpoint3.txt")==1){}
          WriteFileInt("checkpoint3.txt",0);

//XRES is extracted from AV
        System.out.println("|SN| Authentication Vector from HN.\n");

          part2(db,x);

        System.out.println("|SN| AV is sent to UE.");
        System.out.println("\n"+"----------------------------------------"+"\n");
          WriteFileInt("checkpoint4.txt",1);

        proceed();
        if(ReadFileInt("checkpoint5.txt")==0){
          do{
            System.out.println("|SN| There isn't a new file yet. Try again later.");
            System.out.println("\n"+"----------------------------------------"+"\n");
            proceed();
          }while(ReadFileInt("checkpoint5.txt")==0);}
        else if(ReadFileInt("checkpoint5.txt")==1){}
          WriteFileInt("checkpoint5.txt",0);

//Comparison of RES and XRES is done
          part3(db,x);

        System.out.println("|SN| Result of RES challenge is sent both to UE and HN.");
        System.out.println("\n"+"----------------------------------------"+"\n");
          WriteFileInt("checkpoint6.txt",1);
          WriteFileInt("checkpoint7.txt",1);

        x++;
```

```
    } while(x<1000);
  }

//Attach attempt from UE is received. It is recorded to database.
  public static String[][] part1(String[][] db, int x) throws IOException{

    String IMSI=ReadFileString("UE_IMSI.txt");
    System.out.println("|SN| Attach attempt from DNA " + IMSI+"\n");
    WriteFileString("SN_IMSI.txt",IMSI);
    db[x][0]=IMSI;

    return db;
  }

//XRES is extracted from the AV from HN
  public static String[][] part2(String[][] db, int x) throws IOException{

    Authenticate_UE(db[x][0]);
    int[] XRES=ReadFile("XRES_SN.txt",64);

    String XRES_str=BinaryToHex(XRES);
    db[x][1]=XRES_str;

    System.out.println("|SN| AV for UE is prepared.\n");

     return db;
  }

//RES comparison is done
  public static void part3(String[][] db, int x) throws IOException{

    System.out.println("|SN| RES is received from UE.\n");
    System.out.println("|SN| Checking if RES matches XRES..");

    int[] XRES=HexToBinary(db[x][1]);
    int[] RES_SN=HexToBinary(ReadFileString("RES_UE.txt"));

    String RES_result=Compare(XRES,RES_SN);

    if(RES_result.equalsIgnoreCase("same")){
       System.out.println("|SN| RES challenge succeeded.\n");
       WriteFileString("SN_RES_result.txt","VALID");}
    else {
       System.out.println("|SN| RES challenge failed.\n");
       WriteFileString("SN_RES_result.txt","INVALID");}
}

//SN eliminates the part for itself and for UE
  public static void Authenticate_UE(String IMSI) throws IOException {

    String AV_HN=ReadFileString("AV_HN.txt");
```

```
        int[] AV_SN=HexToBinary(AV_HN);

        System.out.println("|SN| Extracting XRES..");

        int[] XRES_SN=new int[64];
        int[] AV_toUE=new int[256];

        CopyArray(AV_SN,XRES_SN,256,0,64);

        System.out.println("|SN| XRES is extracted.\n");
        System.out.println("|SN| Preparing AV for UE..");

        CopyArray(AV_SN,AV_toUE,0,0,256);

        WriteFileHex("AV_toUE.txt",AV_SN);
        WriteFile("XRES_SN.txt",XRES_SN);
    }

}
```

A.4. HN.java

```
import java.io.IOException;

public class HN extends METHODS{

    public static void main(String[] args) throws IOException {

//db[0]=IMSI
//db[1]=new pseudonym
//db[2]=used pseudonym
//db[3]=K_MASTER
//db[4]=OP
//db[5]=SQN

        String[] db=new String[6];

        db[0]=ReadFileString("IMSI_hex.txt");
        db[3]=ReadFileString("K_MASTER_hex.txt");
        db[4]=ReadFileString("OP_hex.txt");
        db[5]=ReadFileString("SQN_hex.txt");
        db[1]="";
        db[2]="";
        WriteFileString("track.txt","0");

//AMF Creation
        int[] AMF_key=new int[16];
```

```
int[] AMF_pseudo=new int[16];
int[] AMF_empty=new int[16];
AMF_key[1]=1;
AMF_pseudo[2]=1;

WriteFile("AMF Key.txt",AMF_key);
WriteFile("AMF Pseudo.txt",AMF_pseudo);
WriteFile("AMF Empty.txt",AMF_empty);

int x=0;

  System.out.println("----------|Home Network|----------");
  System.out.println("--------------|HN|---------------\n");

do{
  proceed();
  if(ReadFileInt("checkpoint2.txt")==0){
    do{
      System.out.println("|HN| There isn't a new file yet. Try again later.");
      System.out.println("\n"+"-----------------------------------------"+"\n");
      proceed();
    } while(ReadFileInt("checkpoint2.txt")==0);}
  else if(ReadFileInt("checkpoint2.txt")==1){}
    WriteFileInt("checkpoint2.txt",0);

//Verifies if the UE is valid party
    part1(db);

  String type;
  type=ReadFileString("AV_type_req.txt");

//R1 type -- RAND will be used for creating K_kasumi
  if(type.equalsIgnoreCase("R1")){

//R1-type AV is created
      part2(db, type);

    System.out.println("|HN| " + type.toUpperCase() + "-type AV is sent to SN.");
    System.out.println("\n"+"-----------------------------------------"+"\n");
      WriteFileInt("checkpoint3.txt",1);

    proceed();
    if(ReadFileInt("checkpoint6.txt")==0){
      do{
        System.out.println("|HN| There isn't a new file yet. Try again later.");
        System.out.println("\n"+"-----------------------------------------"+"\n");
        proceed();
      } while(ReadFileInt("checkpoint6.txt")==0);}
    else if(ReadFileInt("checkpoint6.txt")==1){}
      WriteFileInt("checkpoint6.txt",0);

//Result of the RES comparison is received
```

```
        System.out.println("|HN| Result for RES challenge is received from SN.\n");
        System.out.println("|HN| Checking result..");

        String res=(ReadFileString("SN_RES_result.txt"));
        if(res.equalsIgnoreCase("valid")){
          System.out.println("|HN| Authentication succeeded.");
          System.out.println("\n"+"-----------------------------------------"+"\n");}
        else if(res.equalsIgnoreCase("invalid")){
          System.out.println("|HN| Authentication failed.");
          System.out.println("\n"+"-----------------------------------------"+"\n");}
        else{
          System.out.println("|HN| There is an error!");
          System.exit(0);}
      }

//R2 type -- RAND will include pseudonym
      else if(type.equalsIgnoreCase("R2")){

        System.out.println("|HN| Creating pseudonym..");
        PseudonymCreate(db);

        String temp=db[2];
        db[2]=db[1];
        db[1]=BinaryToText(ReadFile("Pseudonym.txt",10));

//R2-type AV is created
          part2(db, type);

        System.out.println("|HN| " + type.toUpperCase() + "-type AV is sent to SN.");
        System.out.println("\n"+"-----------------------------------------"+"\n");
          WriteFileInt("checkpoint3.txt",1);

        proceed();
        if(ReadFileInt("checkpoint6.txt")==0){
          do{
            System.out.println("|HN| There isn't a new file yet. Try again later.");
            System.out.println("\n"+"-----------------------------------------"+"\n");
            proceed();
          } while(ReadFileInt("checkpoint6.txt")==0);}
        else if(ReadFileInt("checkpoint6.txt")==1){}
          WriteFileInt("checkpoint6.txt",0);

//Result of the RES comparison is received
        System.out.println("|HN| Response result from SN.\n");
        System.out.println("|HN| Checking response..");

        System.out.println("|HN| Result for RES challenge is received from SN.\n");
        System.out.println("|HN| Checking result..");

        String res=(ReadFileString("SN_RES_result.txt"));

        if(res.equalsIgnoreCase("valid")){
```

```
            System.out.println("|HN| Authentication succeeded.");
            System.out.println("\n"+"---------------------------------------------"+"\n");}
        else if(res.equalsIgnoreCase("invalid")){
            System.out.println("|HN| Authentication failed.");
            db[1]=db[2];
            db[2]=temp;
            System.out.println("|UE| Database is corrected.");
            System.out.println("\n"+"---------------------------------------------"+"\n");}
        else{
            System.out.println("|HN| There is an error!");
            System.exit(0);}
    }

//R3 type -- RAND isn't used in specific task
    else if(type.equalsIgnoreCase("R3")){

//R3-type AV is created
            part2(db, type);

        System.out.println("|HN| " + type.toUpperCase() + "-type AV is sent to SN.");
        System.out.println("\n"+"---------------------------------------------"+"\n");
            WriteFileInt("checkpoint3.txt",1);

        proceed();
        if(ReadFileInt("checkpoint6.txt")==0){
            do{
                System.out.println("|HN| There isn't a new file yet. Try again later.");
                System.out.println("\n"+"---------------------------------------------"+"\n");
                proceed();
            } while(ReadFileInt("checkpoint6.txt")==0);}
        else if(ReadFileInt("checkpoint6.txt")==1){}
            WriteFileInt("checkpoint6.txt",0);

//Result of the RES comparison is received
        System.out.println("|HN| Result for RES challenge is received from SN.\n");
        System.out.println("|HN| Checking result..");

        String res=(ReadFileString("SN_RES_result.txt"));
        if(res.equalsIgnoreCase("valid")){
            System.out.println("|HN| Authentication succeeded.");
            System.out.println("\n"+"---------------------------------------------"+"\n");}
        else if(res.equalsIgnoreCase("invalid")){
            System.out.println("|HN| Authentication failed.");
            System.out.println("\n"+"---------------------------------------------"+"\n");}
        else{
            System.out.println("|HN| There is an error!");
            System.exit(0);}


    x++;
    }

```

```
    } while(x<1000);

    }
//Verification process in HN, when there is a connection attempt from UE
  public static void part1(String[] db) throws IOException{

    System.out.println("|HN| Attach attempt from DNA "+ReadFileString("SN_IMSI.txt")+"\n");
    System.out.println("|HN| Checking IMSI..");
    String track;

//IMSI is sent. HN receives IMSI twice. One for key and one for new pseudonym
    if(db[0].equals(ReadFileString("SN_IMSI.txt"))){
      System.out.println("|HN| IMSI is valid.\n");
      track=ReadFileString("track.txt");
      if(track.equals("0")){
        WriteFileString("AV_type_req.txt","R1");//generate R1-type AV
        WriteFileString("track.txt","1");}
      else if(track.equals("1")){
        WriteFileString("AV_type_req.txt","R2");//generate R2-type AV
        WriteFileString("track.txt","0");}}
//New pseudonym is sent. New pseudonym is required
    else if(db[1].equals(ReadFileString("SN_IMSI.txt"))){
      System.out.println("|HN| Pseudonym is valid.\n");
      WriteFileString("AV_type_req.txt","R2");}//generate R2-type AV
//Used pseudonym is sent.
    else if(db[2].equals(ReadFileString("SN_IMSI.txt"))){
      System.out.println("|HN| Pseudonym is valid.\n");
      WriteFileString("AV_type_req.txt","R3");}//generate R3-type AV
    else{
      System.out.println("|HN| IMSI is not valid.");
      System.exit(0);}

  }

//AV is created in order to send necessary information to UE
  public static void part2(String[] db, String type) throws IOException{

    System.out.println("|HN| " + type.toUpperCase() + "-type AV is required.\n");
    System.out.println("|HN| Creating "+ type.toUpperCase() + "-type AV..");

    int[] K_MASTER=HexToBinary(db[3]);
    int[] OP=HexToBinary(db[4]);
    int[] SQN=HexToBinary(db[5]);

    if(type.equalsIgnoreCase("r1")){
      RAND("key");
      int[] RAND_KEY=ReadFile("RAND_key.txt",128);
      int[] AMF_KEY=ReadFile("AMF Key.txt",16);
      int[] XRES_KEY=RES(RAND_KEY,K_MASTER,OP,SQN,AMF_KEY);
      int[] AK_KEY=AK(RAND_KEY,K_MASTER,OP,SQN,AMF_KEY);
      int[] MAC_KEY=MAC(RAND_KEY,K_MASTER,OP,SQN,AMF_KEY);
```

```
        int[] AUTN_KEY=AUTN(SQN,AK_KEY,AMF_KEY,MAC_KEY);
        int[] AV_KEY=AV(RAND_KEY,XRES_KEY,AUTN_KEY);

        WriteFileHex("KEY_KASUMI_HN.txt",KeyKasumi(RAND_KEY, K_MASTER, OP, SQN, AMF_KEY));

        System.out.println("|HN| R1-type AV is created.\n");
        WriteFileHex("AV_HN.txt",AV_KEY);
      }

      else if(type.equalsIgnoreCase("r2")){

        RAND("pseudonym");
        int[] RAND_PSEUDO=ReadFile("RAND_pseudo.txt",128);
        int[] AMF_PSEUDO=ReadFile("AMF Pseudo.txt",16);
        int[] XRES_PSEUDO=RES(RAND_PSEUDO,K_MASTER,OP,SQN,AMF_PSEUDO);
        int[] AK_PSEUDO=AK(RAND_PSEUDO,K_MASTER,OP,SQN,AMF_PSEUDO);
        int[] MAC_PSEUDO=MAC(RAND_PSEUDO,K_MASTER,OP,SQN,AMF_PSEUDO);
        int[] AUTN_PSEUDO=AUTN(SQN,AK_PSEUDO,AMF_PSEUDO,MAC_PSEUDO);
        int[] AV_PSEUDO=AV(RAND_PSEUDO,XRES_PSEUDO,AUTN_PSEUDO);

        System.out.println("|HN| R2-type AV is created.\n");
        WriteFileHex("AV_HN.txt",AV_PSEUDO);
      }

      else if(type.equalsIgnoreCase("r3")){
        RAND("empty");
        int[] RAND_EMPTY=ReadFile("RAND_emp.txt",128);
        int[] AMF_EMPTY=ReadFile("AMF Empty.txt",16);
        int[] XRES_EMPTY=RES(RAND_EMPTY,K_MASTER,OP,SQN,AMF_EMPTY);
        int[] AK_EMPTY=AK(RAND_EMPTY,K_MASTER,OP,SQN,AMF_EMPTY);
        int[] MAC_EMPTY=MAC(RAND_EMPTY,K_MASTER,OP,SQN,AMF_EMPTY);
        int[] AUTN_EMPTY=AUTN(SQN,AK_EMPTY,AMF_EMPTY,MAC_EMPTY);
        int[] AV_EMPTY=AV(RAND_EMPTY,XRES_EMPTY,AUTN_EMPTY);

        System.out.println("|HN| R3-type AV is created.\n");
        WriteFileHex("AV_HN.txt",AV_EMPTY);
      }
    }

//Creates RAND according to the input. Writes to the file
    public static void RAND(String reason)throws IOException{

      int[] RAND_key, RAND_pseudo, RAND_empty;

      if(reason.equalsIgnoreCase("key")){
        RAND_key=random(128);
        WriteFile("RAND_key.txt",RAND_key);}

      else if(reason.equalsIgnoreCase("empty")){
        RAND_empty=random(128);
        WriteFile("RAND_emp.txt",RAND_empty);}
```

```

     else if(reason.equalsIgnoreCase("pseudonym")){
        int[] Pseudo_IMSI, Pseudo_bit, Pseudo_pad;
        int[] pad, rnd1, rnd2;
        Pseudo_IMSI=ReadFile("Pseudo_IMSI.txt",10);
        Pseudo_bit=PseudoToBits(Pseudo_IMSI);
        pad=ReadFile("pad.txt",24);
        Pseudo_pad=Concatenate(Pseudo_bit,pad);

        int[] ciphertext, KEY_kasumi;
        KEY_kasumi=HexToBinary(ReadFileString("KEY_KASUMI_HN.txt"));
        ciphertext=KASUMI_enc(Pseudo_pad,KEY_kasumi);

        rnd1=ReadFile("rnd1.txt",32);
        rnd2=ReadFile("rnd2.txt",32);

        RAND_pseudo=Concatenate(Concatenate(rnd1,ciphertext),rnd2);
        WriteFile("RAND_pseudo.txt",RAND_pseudo);}
  }

//Creates AUTN with the inputs
  public static int[] AUTN(int[] SQN, int[] AK, int[] AMF, int[] MAC){
     int[] temp=XOR(SQN,AK);
     int[] output;
     output=Concatenate(Concatenate(temp,AMF),MAC);

     return output;
  }

//Creates AV-authentication vector
  public static int[] AV(int[] RAND, int[] XRES, int[] AUTN){
     int[] output = Concatenate(Concatenate(RAND,AUTN),XRES);

     return output;
  }

//Create Key for Kasumi encryption
  public static int[] KeyKasumi(int[] RAND, int[] K_MASTER, int[] OP, int[] SQN, int[] AMF) throws IOException{

     int[] CK, IK, AK;
     CK=CK(RAND, K_MASTER, OP, SQN, AMF);
     IK=IK(RAND, K_MASTER, OP, SQN, AMF);
     AK=AK(RAND, K_MASTER, OP, SQN, AMF);

     int[] K,S;
     int[] FC, P0, L0, P1, L1;
     int[] oct1, oct2, oct3;

     K=Concatenate(CK, IK);
     FC=HexToBinary("60");
```

```java
    //For MCC and MNC, I will use DNA Oy, which is 244 12.
        oct1=Concatenate(HexToBinary("4"),HexToBinary("2"));
        oct2=Concatenate(HexToBinary("f"),HexToBinary("4"));
        oct3=Concatenate(HexToBinary("2"),HexToBinary("1"));
        P0=Concatenate(Concatenate(oct1,oct2),oct3);
        L0=Concatenate(HexToBinary("000"),HexToBinary("3"));
        P1=XOR(SQN, AK);
        L1=Concatenate(HexToBinary("000"),HexToBinary("6"));

    // S=FC || P0 || LO || P1 || L1.
        S=Concatenate(Concatenate(Concatenate(Concatenate(FC,P0),L0),P1),L1);

        int[] KEY_kasumi=HMAC(S,K);

        return KEY_kasumi;
    }

//Creates random pseudonym with the necessary random inputs and writes to the file
    public static void CreatePseudonym() throws IOException{
        int[] Pseudo_IMSI;
        int[] pad, rnd1, rnd2;

        Pseudo_IMSI=rand_number(10);
        pad=random(24);
        rnd1=random(32);
        rnd2=random(32);

        WriteFile("Pseudo_IMSI.txt",Pseudo_IMSI);
        WriteFile("pad.txt",pad);
        WriteFile("rnd1.txt",rnd1);
        WriteFile("rnd2.txt",rnd2);
    }

//Makes sure that created pseudonym is not used before
    public static void PseudonymCreate(String[] db) throws IOException{

        CreatePseudonym();
        int[] pseudonym=ReadFile("Pseudo_IMSI.txt",10);
        int[] temp1=HexToBinary(db[0]);
        int[] temp2=HexToBinary(db[1]);
        int[] temp3=HexToBinary(db[2]);

        String t=Compare(temp1,pseudonym); //Check if the pseudonym is used before
        String tt=Compare(temp2,pseudonym); //Check if the pseudonym is used before
        String ttt=Compare(temp3,pseudonym); //Check if the pseudonym is used before

        if(t.equalsIgnoreCase("same") || tt.equalsIgnoreCase("same") || ttt.equalsIgnoreCase("same")){
            do{
                CreatePseudonym();
                pseudonym=ReadFile("Pseudo_IMSI.txt",10);
```

```java
            t=Compare(temp1,pseudonym);
            tt=Compare(temp2,pseudonym);
            ttt=Compare(temp3,pseudonym);
        }
        while(t.equalsIgnoreCase("same") || tt.equalsIgnoreCase("same") || ttt.equalsIgnoreCase("same"));
    }

    System.out.println("|HN| New Pseudonym is created.");
    System.out.println("|HN| New Pseudonym is "+BinaryToText(pseudonym)+"\n");
    WriteFile("Pseudonym.txt",pseudonym);
}
```

A.5. METHODS.java

```java
import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;
import static java.lang.Math.pow;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
import java.util.Arrays;
import java.util.Scanner;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;


class METHODS {


//---GENERAL FUNCTIONS---//


//creates random bits of 0 and 1. Input is length of the array. Output is an array of random bits.
    public static int[] random (int bitlength){
        int[] array;
        array = new int[bitlength];
        SecureRandom rnd = new SecureRandom();
        for(int i=0; i<bitlength; i++){
            array[i]=rnd.nextInt(2);}

        return array;
    }
```

```

//creates random bits from 0 to 9. Input is length of the array. Output is an array of random bits.
  public static int[] rand_number (int bitlength){
    int[] array;
    array = new int[bitlength];
    SecureRandom rnd = new SecureRandom();
    for(int i=0; i<bitlength; i++){
      array[i]=rnd.nextInt(10);}

    return array;
  }

//copies the elements from one array to another.
  public static int[] CopyArray (int[] arrayFrom, int[] arrayTo, int a, int b, int c){
    //a=initial element to start copy in arrayFrom
    //b=initial element to past in arrayTo
    //c=how many elements to copy

    System.arraycopy(arrayFrom, a, arrayTo, b, c);

    return arrayTo;
  }

//same as previous. Only the output is String.
  public static String CopyArrayString (int[] arrayFrom, int[] arrayTo, int a, int b, int c){
    //a=initial element to start copy in arrayFrom
    //b=initial element to past in arrayTo
    //c=how many elements to copy

    System.arraycopy(arrayFrom, a, arrayTo, b, c);
    String t = Arrays.toString(arrayTo);

    return t;
  }

//copies the elements from a String to another
  public static String CopyString (String from, int a, int b){
    String out="";
    for(int x=0; x<b; x++){
      out+=from.charAt(x+a);}

    return out;
  }

//takes two array and concatenates them, displays as a one array
  public static int[] Concatenate (int[] array1, int[] array2) {
    int al;
    al=array1.length+array2.length;
    int[] result = new int[al];
    CopyArray(array1,result,0,0,array1.length);
```

```
    CopyArray(array2,result,0,array1.length,array2.length);

    return result;
  }

//executes XOR operation. Input is two elements of operation as arrays. Output is array of integers as a result.
  public static int[] XOR (int[] input, int[] key) {
    int[] output;
    output = new int[input.length];
    for(int x=0; x<input.length; x++){
      output[x] = input[x] ^ key[x];}

    return output;
  }

//executes AND operation. Input is two elements of operation as arrays. Output is array of integers as a result.
  public static int[] AND (int[] input1, int[] input2) {
    int[] output;
    output = new int[input1.length];
    for(int x=0; x<input1.length; x++){
      output[x] = input1[x] & input2[x];}

    return output;
  }

//executes OR operation. Input is two elements of operation as arrays. Output is array of integers as a result.
  public static int[] OR (int[] input1, int[] input2) {
    int[] output;
    output = new int[input1.length];
    for(int x=0; x<input1.length; x++){
      output[x] = input1[x] | input2[x];}

    return output;
  }

//left circular rotation operation. Rotation is done by n bits.
  public static int[] CircularLeftRotation (int[] array, int n) {
    int al=array.length;
    int[] output;
    output=new int[al];
    CopyArray(array,output,0,0,al);
    for (int x=0; x<n; x++){
      int first = output[0];
      CopyArray(output,output,1,0,al-1);
      output[al - 1] = first;}

    return output;
  }

```

```
//right circular rotation operation. Rotation is done by n bits.
  public static int[] CircularRightRotation (int[] array, int n) {
    int al=array.length;
    int[] output;
    output=new int[al];
    CopyArray(array,output,0,0,al);
    for (int x=0; x<n; x++){
      int last = output[al-1];
      CopyArray(output,output,0,1,al-1);
      output[0] = last;}

    return output;
  }

//divides the array into two arrays and displays first part of n length.
  public static int[] DivideFirst (int[] array, int n) {
    int[] first = new int[n];
    CopyArray(array,first,0,0,n);

    return first;
  }

//divides the array into two arrays and displays second part of array.length-n length
  public static int[] DivideSecond (int[] array, int n) {
    int al;
    al=array.length;
    int[] second = new int[al-n];
    CopyArray(array,second,n,0,al-n);

    return second;
  }

//prints the array as String
  public static String ArrayToString (int[] input){
    String out="";
    for(int x=0; x<input.length; x++){
      out+=input[x];}

    return out;
  }

//Compares two arrays and gives as output if they are identical or not
  public static String Compare(int[] a1, int[] a2){
    String result;
    int t=0;

    if(a1.length==a2.length){
      for(int x=0; x<a1.length; x++){
        if(a1[x]==a2[x]){
          t+=1;}
      }
      if(t<a1.length){
```

```
            result="notsame";}
        else{
            result="same";}
    }
    else{
        result="notsame";}

    return result;
}

//Response process during communication
public static void response(String m){
    if(m.equalsIgnoreCase("YES")||m.equalsIgnoreCase("y")){}
    else if(m.equalsIgnoreCase("NO")){
        System.out.println("Operation is stopped!");
        System.exit(0);}
}

public static void proceed(){
    System.out.println("Press enter to proceed. (Write STOP to exit.)");
    Scanner scan = new Scanner (System.in);
    String t=scan.nextLine();
    if(t.equalsIgnoreCase("stop")||t.equalsIgnoreCase("no")){
        System.exit(0);}
}

//---CONVERSIONS---//

//converts hex to binary array. Input is String of hexadecimal digits. Output is the 4 times bigger as the length of the input.
public static int[] HexToBinaryArrayKey (String s){

    String digits = "0123456789ABCDEF";
    s = s.toUpperCase();
    int[] hex=new int[s.length()];

    for(int x=0; x<s.length(); x++){
            char c=s.charAt(x);
            int d=digits.indexOf(c);
            hex[x]=d;}

    int[] binary=new int[4*s.length()];
    for(int y=0; y<hex.length; y++){
        if(hex[y]==0){
            int[] temp={0,0,0,0};
            CopyArray(temp,binary,0,4*y,4);}
        else if(hex[y]==1){
            int[] temp={0,0,0,1};
            CopyArray(temp,binary,0,4*y,4);}
        else if(hex[y]==2){
            int[] temp={0,0,1,0};
            CopyArray(temp,binary,0,4*y,4);}
```

```
        else if(hex[y]==3){
          int[] temp={0,0,1,1};
          CopyArray(temp,binary,0,4*y,4);}
        else if(hex[y]==4){
          int[] temp={0,1,0,0};
          CopyArray(temp,binary,0,4*y,4);}
        else if(hex[y]==5){
          int[] temp={0,1,0,1};
          CopyArray(temp,binary,0,4*y,4);}
        else if(hex[y]==6){
          int[] temp={0,1,1,0};
          CopyArray(temp,binary,0,4*y,4);}
        else if(hex[y]==7){
          int[] temp={0,1,1,1};
          CopyArray(temp,binary,0,4*y,4);}
        else if(hex[y]==8){
          int[] temp={1,0,0,0};
          CopyArray(temp,binary,0,4*y,4);}
        else if(hex[y]==9){
          int[] temp={1,0,0,1};
          CopyArray(temp,binary,0,4*y,4);}
        else if(hex[y]==10){
          int[] temp={1,0,1,0};
          CopyArray(temp,binary,0,4*y,4);}
        else if(hex[y]==11){
          int[] temp={1,0,1,1};
          CopyArray(temp,binary,0,4*y,4);}
        else if(hex[y]==12){
          int[] temp={1,1,0,0};
          CopyArray(temp,binary,0,4*y,4);}
        else if(hex[y]==13){
          int[] temp={1,1,0,1};
          CopyArray(temp,binary,0,4*y,4);}
        else if(hex[y]==14){
          int[] temp={1,1,1,0};
          CopyArray(temp,binary,0,4*y,4);}
        else if(hex[y]==15){
          int[] temp={1,1,1,1};
          CopyArray(temp,binary,0,4*y,4);}
      }

    return binary;
  }

//converts binary array to hex. Input is array of binary digits. Output is 4 times smaller as the length of the array
  public static String BinaryToHex (int[] binary){
    int[] bin;
    if(binary.length%4!=0){
      int t=binary.length + (4 - (binary.length%4));
      bin=new int[t];
      CopyArray(binary,bin,0,t-binary.length,binary.length);}
```

```
    else{
      bin=new int[binary.length];
      CopyArray(binary,bin,0,0,binary.length);}

    String s="";
    for(int x=0; x<bin.length/4; x++){
      int h=8*bin[4*x]+4*bin[4*x+1]+2*bin[4*x+2]+bin[4*x+3];
      if(h==0){
        s+="0";}
      else if(h==1){
        s+="1";}
      else if(h==2){
        s+="2";}
      else if(h==3){
        s+="3";}
      else if(h==4){
        s+="4";}
      else if(h==5){
        s+="5";}
      else if(h==6){
        s+="6";}
      else if(h==7){
        s+="7";}
      else if(h==8){
        s+="8";}
      else if(h==9){
        s+="9";}
      else if(h==10){
        s+="A";}
      else if(h==11){
        s+="B";}
      else if(h==12){
        s+="C";}
      else if(h==13){
        s+="D";}
      else if(h==14){
        s+="E";}
      else if(h==15){
        s+="F";}
    }
    s=s.toLowerCase();

    return s;
  }

//converts hex to binary array. Input is String of hexadecimal digits. Output is the 4 times bigger as the length of the input.
  public static int[] HexToBinary (String s){

    String digits = "0123456789ABCDEF";
    s = s.toUpperCase();
    int[] hex=new int[s.length()];
```

```
for(int x=0; x<s.length(); x++){
  char c=s.charAt(x);
  int d=digits.indexOf(c);
  hex[x]=d;}

int[] binary=new int[4*s.length()];
for(int y=0; y<hex.length; y++){
  if(hex[y]==0){
    int[] temp={0,0,0,0};
    CopyArray(temp,binary,0,4*y,4);}
  else if(hex[y]==1){
    int[] temp={0,0,0,1};
    CopyArray(temp,binary,0,4*y,4);}
  else if(hex[y]==2){
    int[] temp={0,0,1,0};
    CopyArray(temp,binary,0,4*y,4);}
  else if(hex[y]==3){
    int[] temp={0,0,1,1};
    CopyArray(temp,binary,0,4*y,4);}
  else if(hex[y]==4){
    int[] temp={0,1,0,0};
    CopyArray(temp,binary,0,4*y,4);}
  else if(hex[y]==5){
    int[] temp={0,1,0,1};
    CopyArray(temp,binary,0,4*y,4);}
  else if(hex[y]==6){
    int[] temp={0,1,1,0};
    CopyArray(temp,binary,0,4*y,4);}
  else if(hex[y]==7){
    int[] temp={0,1,1,1};
    CopyArray(temp,binary,0,4*y,4);}
  else if(hex[y]==8){
    int[] temp={1,0,0,0};
    CopyArray(temp,binary,0,4*y,4);}
  else if(hex[y]==9){
    int[] temp={1,0,0,1};
    CopyArray(temp,binary,0,4*y,4);}
  else if(hex[y]==10){
    int[] temp={1,0,1,0};
    CopyArray(temp,binary,0,4*y,4);}
  else if(hex[y]==11){
    int[] temp={1,0,1,1};
    CopyArray(temp,binary,0,4*y,4);}
  else if(hex[y]==12){
    int[] temp={1,1,0,0};
    CopyArray(temp,binary,0,4*y,4);}
  else if(hex[y]==13){
    int[] temp={1,1,0,1};
    CopyArray(temp,binary,0,4*y,4);}
  else if(hex[y]==14){
    int[] temp={1,1,1,0};
```

```
            CopyArray(temp,binary,0,4*y,4);}
        else if(hex[y]==15){
            int[] temp={1,1,1,1};
            CopyArray(temp,binary,0,4*y,4);}
    }
    return binary;
  }

//Array to ASCII converter
  public static String BinaryToText(int[] message){
    String output="";
    for(int x=0; x<message.length; x++){
        output+=message[x];}

    return output;
  }

//Writes pseudonym in hex to an array. output 10-bits
  public static int[] PseudoHexToBit(String s){
    int[] ps=new int[10];
    String a="0123456789";
    for(int x=0; x<10; x++){
        if(s.charAt(x)==a.charAt(0)){
            ps[x]=0;}
        else if(s.charAt(x)==a.charAt(1)){
            ps[x]=1;}
        else if(s.charAt(x)==a.charAt(2)){
            ps[x]=2;}
        else if(s.charAt(x)==a.charAt(3)){
            ps[x]=3;}
        else if(s.charAt(x)==a.charAt(4)){
            ps[x]=4;}
        else if(s.charAt(x)==a.charAt(5)){
            ps[x]=5;}
        else if(s.charAt(x)==a.charAt(6)){
            ps[x]=6;}
        else if(s.charAt(x)==a.charAt(7)){
            ps[x]=7;}
        else if(s.charAt(x)==a.charAt(8)){
            ps[x]=8;}
        else if(s.charAt(x)==a.charAt(9)){
            ps[x]=9;}
    }
    return ps;
  }

//Converts IMSI-like number to bits. output 40-bits
  public static int[] PseudoToBits (int[] pseudonym){

    int[] pseudonymBinary;
    pseudonymBinary = new int[40];

```

```
for(int c=0; c<10; c++){
   int c1 = 0+4*c;
   int c2 = 1+4*c;
   int c3 = 2+4*c;
   int c4 = 3+4*c;

   if(pseudonym[c]==0){
      pseudonymBinary[c1]=0;
      pseudonymBinary[c2]=0;
      pseudonymBinary[c3]=0;
      pseudonymBinary[c4]=0;}

   else if(pseudonym[c]==1){
      pseudonymBinary[c1]=0;
      pseudonymBinary[c2]=0;
      pseudonymBinary[c3]=0;
      pseudonymBinary[c4]=1;}

   else if(pseudonym[c]==2){
      pseudonymBinary[c1]=0;
      pseudonymBinary[c2]=0;
      pseudonymBinary[c3]=1;
      pseudonymBinary[c4]=0;}

   else if(pseudonym[c]==3){
      pseudonymBinary[c1]=0;
      pseudonymBinary[c2]=0;
      pseudonymBinary[c3]=1;
      pseudonymBinary[c4]=1;}

   else if(pseudonym[c]==4){
      pseudonymBinary[c1]=0;
      pseudonymBinary[c2]=1;
      pseudonymBinary[c3]=0;
      pseudonymBinary[c4]=0;}

   else if(pseudonym[c]==5){
      pseudonymBinary[c1]=0;
      pseudonymBinary[c2]=1;
      pseudonymBinary[c3]=0;
      pseudonymBinary[c4]=1;}

   else if(pseudonym[c]==6){
      pseudonymBinary[c1]=0;
      pseudonymBinary[c2]=1;
      pseudonymBinary[c3]=1;
      pseudonymBinary[c4]=0;}

   else if(pseudonym[c]==7){
      pseudonymBinary[c1]=0;
      pseudonymBinary[c2]=1;
      pseudonymBinary[c3]=1;
```

```
            pseudonymBinary[c4]=1;}

        else if(pseudonym[c]==8){
            pseudonymBinary[c1]=1;
            pseudonymBinary[c2]=0;
            pseudonymBinary[c3]=0;
            pseudonymBinary[c4]=0;}

        else if(pseudonym[c]==9){
            pseudonymBinary[c1]=1;
            pseudonymBinary[c2]=0;
            pseudonymBinary[c3]=0;
            pseudonymBinary[c4]=1;}
    }
    return pseudonymBinary;
  }

//bit to byte conversion
  public static int BitToByteConv (int[] bit){
    int val=0;
    for(int x=0; x<8; x++){
        val=val+bit[x]*(int)pow(2,7-x);}

    return val;
  }

//byte to bit conversion
  public static int[] ByteToBitConv (int bayt){
    int[] bit=new int[8];

    for(int x=0; x<8; x++){
        if((bayt-pow(2,7-x))<0){
            bit[x]=0;}
        else{
            bit[x]=1;
            bayt=bayt-(int)pow(2,7-x);}}
    return bit;
  }

//change bits to bytes and create 4x4 matrices
  public static int[][] ByteMatrix (int[] bits){

    int[] arr0 = new int[8];
    int[] arr1 = new int[8];
    int[] arr2 = new int[8];
    int[] arr3 = new int[8];
    int[] arr4 = new int[8];
    int[] arr5 = new int[8];
    int[] arr6 = new int[8];
    int[] arr7 = new int[8];
    int[] arr8 = new int[8];
    int[] arr9 = new int[8];
```

```
        int[] arr10 = new int[8];
        int[] arr11 = new int[8];
        int[] arr12 = new int[8];
        int[] arr13 = new int[8];
        int[] arr14 = new int[8];
        int[] arr15 = new int[8];

        CopyArray(bits,arr0,0,0,8);
        CopyArray(bits,arr1,8,0,8);
        CopyArray(bits,arr2,16,0,8);
        CopyArray(bits,arr3,24,0,8);
        CopyArray(bits,arr4,32,0,8);
        CopyArray(bits,arr5,40,0,8);
        CopyArray(bits,arr6,48,0,8);
        CopyArray(bits,arr7,56,0,8);
        CopyArray(bits,arr8,64,0,8);
        CopyArray(bits,arr9,72,0,8);
        CopyArray(bits,arr10,80,0,8);
        CopyArray(bits,arr11,88,0,8);
        CopyArray(bits,arr12,96,0,8);
        CopyArray(bits,arr13,104,0,8);
        CopyArray(bits,arr14,112,0,8);
        CopyArray(bits,arr15,120,0,8);

        int a00=BitToByteConv(arr0);
        int a10=BitToByteConv(arr1);
        int a20=BitToByteConv(arr2);
        int a30=BitToByteConv(arr3);
        int a01=BitToByteConv(arr4);
        int a11=BitToByteConv(arr5);
        int a21=BitToByteConv(arr6);
        int a31=BitToByteConv(arr7);
        int a02=BitToByteConv(arr8);
        int a12=BitToByteConv(arr9);
        int a22=BitToByteConv(arr10);
        int a32=BitToByteConv(arr11);
        int a03=BitToByteConv(arr12);
        int a13=BitToByteConv(arr13);
        int a23=BitToByteConv(arr14);
        int a33=BitToByteConv(arr15);

        int[][] output={{a00,a01,a02,a03},{a10,a11,a12,a13},{a20,a21,a22,a23},{a30,a31,a32,a33}};

        return output;
    }

//change 4x4 matrices with bytes to bits
  public static int[] MatrixBit (int[][] matrix){
        int[] bits=new int[128];

        int[] a00=ByteToBitConv(matrix[0][0]);
```

```java
    int[] a10=ByteToBitConv(matrix[1][0]);
    int[] a20=ByteToBitConv(matrix[2][0]);
    int[] a30=ByteToBitConv(matrix[3][0]);
    int[] a01=ByteToBitConv(matrix[0][1]);
    int[] a11=ByteToBitConv(matrix[1][1]);
    int[] a21=ByteToBitConv(matrix[2][1]);
    int[] a31=ByteToBitConv(matrix[3][1]);
    int[] a02=ByteToBitConv(matrix[0][2]);
    int[] a12=ByteToBitConv(matrix[1][2]);
    int[] a22=ByteToBitConv(matrix[2][2]);
    int[] a32=ByteToBitConv(matrix[3][2]);
    int[] a03=ByteToBitConv(matrix[0][3]);
    int[] a13=ByteToBitConv(matrix[1][3]);
    int[] a23=ByteToBitConv(matrix[2][3]);
    int[] a33=ByteToBitConv(matrix[3][3]);

    CopyArray(a00,bits,0,0,8);
    CopyArray(a10,bits,0,8,8);
    CopyArray(a20,bits,0,16,8);
    CopyArray(a30,bits,0,24,8);
    CopyArray(a01,bits,0,32,8);
    CopyArray(a11,bits,0,40,8);
    CopyArray(a21,bits,0,48,8);
    CopyArray(a31,bits,0,56,8);
    CopyArray(a02,bits,0,64,8);
    CopyArray(a12,bits,0,72,8);
    CopyArray(a22,bits,0,80,8);
    CopyArray(a32,bits,0,88,8);
    CopyArray(a03,bits,0,96,8);
    CopyArray(a13,bits,0,104,8);
    CopyArray(a23,bits,0,112,8);
    CopyArray(a33,bits,0,120,8);

    return bits;
  }

//---WRITE & READ---//

//Reads file. Takes the name of the file and the length of the array. Returns an array.
  public static int[] ReadFile (String s, int limit) throws IOException{
    int[] rand=new int[limit];

    File file = new File(s);
    Scanner input = new Scanner(file);

    for(int x=0; x<limit; x++){
      rand[x]=input.nextInt();}

    input.close();

    return rand;
  }
```

```
//Reads file. Takes the name of the file and returns the integer
  public static int ReadFileInt (String s) throws IOException{
    int rand;

    File file = new File(s);
    Scanner input = new Scanner(file);

    rand=input.nextInt();

    input.close();

    return rand;
  }

//Reads file. Takes the name of the file. Returns string.
  public static String ReadFileString (String s) throws IOException{
    String output;

    File file = new File(s);
    Scanner input = new Scanner(file);

    output=input.nextLine();
    input.close();

    return output;
  }

//Writes to file. Take the name of the file and array to be written. Return message.
  public static String WriteFile (String s, int[] array) throws IOException{

    String FileName = s;
    PrintWriter outFile = new PrintWriter(FileName);

    for(int x=0; x<array.length; x++){
      outFile.println(array[x]);}

    outFile.close();

    String output="File " + s + " is created.";

    return output;
  }

//Writes to file. Take the name of the file and array to be written. Return message.
  public static String WriteFileInt (String s, int in) throws IOException{

    String FileName = s;
    PrintWriter outFile = new PrintWriter(FileName);

    outFile.println(in);

```

```
      outFile.close();

      String output="File " + s + " is created.";

      return output;
   }

//Stores the input array as converted to hex in the file
   public static String WriteFileHex(String s, int[] array) throws IOException{

      String FileName = s;
      PrintWriter outFile = new PrintWriter(FileName);

      String text;
      text=BinaryToHex(array);
      outFile.println(text);

      outFile.close();

      String output = "File " + s + " is created.";

      return output;
   }

//Writes the string to the file
   public static String WriteFileString(String s, String text) throws IOException{

      String FileName = s;
      PrintWriter outFile = new PrintWriter(FileName);

      outFile.println(text);
      outFile.close();

      String output = "File " + s + " is created.";

      return output;
   }

//write matrix
   public static String WriteMatrix (int[][] mat){
      String str="";
      for(int s=0; s<4; s++){
         for(int f=0; f<4; f++){
            str+=mat[s][f]+"\t";}
            str+="\n";}

      return str;
   }

//---AES---//

//S-BOX for AES
```

```
public static int SBOX (int input){
  int[] sbox = {99,124,119,123,242,107,111,197,48,1,103,43,254,215,171,
    118,202,130,201,125,250,89,71,240,173,212,162,175,156,164,114,192,
    183,253,147, 38, 54, 63,247,204, 52,165,229,241,113,216, 49, 21,
    4,199, 35,195, 24,150,  5,154,  7, 18,128,226,235, 39,178,117,9,
    131, 44, 26, 27,110, 90,160, 82, 59,214,179, 41,227, 47,132,83,209,
    0,237, 32,252,177, 91,106,203,190, 57, 74, 76, 88,207,208,239,170,
    251, 67, 77, 51,133, 69,249,  2,127, 80, 60,159,168,81,163, 64,143,
    146,157, 56,245,188,182,218, 33, 16,255,243,210,205, 12, 19,236,
    95,151, 68, 23,196,167,126, 61,100, 93, 25,115,96,129, 79,220,34,
    42,144,136, 70,238,184, 20,222, 94, 11,219,224, 50, 58, 10, 73,
    6, 36, 92,194,211,172, 98,145,149,228,121,231,200, 55,109,141,213,
    78,169,108, 86,244,234,101,122,174,  8,186,120, 37, 46, 28,166,180,
    198,232,221,116, 31, 75,189,139,138,112, 62,181,102, 72,  3,246,
    14, 97, 53, 87,185,134,193, 29,158,225,248,152, 17,105,217,142,148,
    155, 30,135,233,206, 85, 40,223,140,161,137, 13,191,230, 66,104,
    65,153, 45, 15,176, 84,187, 22};

  int output=sbox[input];

  return output;
}

//ByteSubstitution Transformation
public static int[][] ByteSubstitution (int[][] input){
  int[][] output=new int[4][4];

  for(int x=0; x<4; x++){
    for(int y=0; y<4; y++){
      output[x][y]=SBOX(input[x][y]);}
  }

  return output;
}

//ShiftRow Transformation
public static int[][] ShiftRow (int[][] input){
  int[][] output=new int[4][4];

  output[0][0]=input[0][0];
  output[0][1]=input[0][1];
  output[0][2]=input[0][2];
  output[0][3]=input[0][3];
  output[1][0]=input[1][1];
  output[1][1]=input[1][2];
  output[1][2]=input[1][3];
  output[1][3]=input[1][0];
  output[2][0]=input[2][2];
  output[2][1]=input[2][3];
  output[2][2]=input[2][0];
  output[2][3]=input[2][1];
  output[3][0]=input[3][3];
```

```
        output[3][1]=input[3][0];
        output[3][2]=input[3][1];
        output[3][3]=input[3][2];

        return output;
    }

//T2 function for MixColumn
    public static int T2 (int input){
        int output;

        if(input<128){
            output=2*input;}
        else{
            output=(2*input)^283;}

        return output;
    }

//T3 function for Mix Column
    public static int T3 (int input){
        int output;
        output=T2(input)^input;

        return output;
    }

//MixColumn Transformation
    public static int[][] MixColumn (int[][] input){
        int[][] output=new int[4][4];

        for(int x=0; x<4; x++){
            output[0][x]=T2(input[0][x])^T3(input[1][x])^input[2][x]^input[3][x];
            output[1][x]=input[0][x]^T2(input[1][x])^T3(input[2][x])^input[3][x];
            output[2][x]=input[0][x]^input[1][x]^T2(input[2][x])^T3(input[3][x]);
            output[3][x]=T3(input[0][x])^input[1][x]^input[2][x]^T2(input[3][x]);}

        return output;
    }

//round key addition process
    public static int[][] AddRoundKey (int[][] plaintext, int[][] key){
        int[][] ciphertext=new int[4][4];

        for(int x=0; x<4; x++){
            for(int y=0; y<4; y++){
                ciphertext[x][y]=plaintext[x][y]^key[x][y];}
        }

        return ciphertext;
    }
```

```java
//generate round keys
public static int[][] GenRoundKey (int[][] prev, int round){

    int round_const1=1;
    int round_const2=T2(round_const1);
    int round_const3=T2(round_const2);
    int round_const4=T2(round_const3);
    int round_const5=T2(round_const4);
    int round_const6=T2(round_const5);
    int round_const7=T2(round_const6);
    int round_const8=T2(round_const7);
    int round_const9=T2(round_const8);
    int round_const10=T2(round_const9);

    int[] round_const={round_const1, round_const2, round_const3, round_const4,
      round_const5, round_const6, round_const7, round_const8, round_const9, round_const10};

    int[][] RoundKey=new int[4][4];
    RoundKey[0][0]=prev[0][0]^SBOX(prev[1][3])^round_const[round-1];
    RoundKey[1][0]=prev[1][0]^SBOX(prev[2][3]);
    RoundKey[2][0]=prev[2][0]^SBOX(prev[3][3]);
    RoundKey[3][0]=prev[3][0]^SBOX(prev[0][3]);

    RoundKey[0][1]=prev[0][1]^RoundKey[0][0];
    RoundKey[1][1]=prev[1][1]^RoundKey[1][0];
    RoundKey[2][1]=prev[2][1]^RoundKey[2][0];
    RoundKey[3][1]=prev[3][1]^RoundKey[3][0];

    RoundKey[0][2]=prev[0][2]^RoundKey[0][1];
    RoundKey[1][2]=prev[1][2]^RoundKey[1][1];
    RoundKey[2][2]=prev[2][2]^RoundKey[2][1];
    RoundKey[3][2]=prev[3][2]^RoundKey[3][1];

    RoundKey[0][3]=prev[0][3]^RoundKey[0][2];
    RoundKey[1][3]=prev[1][3]^RoundKey[1][2];
    RoundKey[2][3]=prev[2][3]^RoundKey[2][2];
    RoundKey[3][3]=prev[3][3]^RoundKey[3][2];

    return RoundKey;
}

//AES128 encryption
public static int[] AES (int[] P, int[] K){

    int[][] Plaintext=ByteMatrix(P);
    int[][] Key=ByteMatrix(K);

//Zero'th round key is Key matrix

    int[][] RoundKey1, RoundKey2, RoundKey3, RoundKey4, RoundKey5, RoundKey6,
```

```
        RoundKey7, RoundKey8, RoundKey9, RoundKey10;

//Generate Round Key
    RoundKey1=GenRoundKey(Key,1);
    RoundKey2=GenRoundKey(RoundKey1,2);
    RoundKey3=GenRoundKey(RoundKey2,3);
    RoundKey4=GenRoundKey(RoundKey3,4);
    RoundKey5=GenRoundKey(RoundKey4,5);
    RoundKey6=GenRoundKey(RoundKey5,6);
    RoundKey7=GenRoundKey(RoundKey6,7);
    RoundKey8=GenRoundKey(RoundKey7,8);
    RoundKey9=GenRoundKey(RoundKey8,9);
    RoundKey10=GenRoundKey(RoundKey9,10);

//Encryption Starts
    int[][] Round0, Round1, Round2, Round3, Round4, Round5, Round6, Round7,
        Round8, Round9, Round10;
//Initial Key Addition
    Round0=AddRoundKey(Plaintext,Key);

//Round1
    Round1=ByteSubstitution(Round0);
    Round1=ShiftRow(Round1);
    Round1=MixColumn(Round1);
    Round1=AddRoundKey(Round1,RoundKey1);

//Round2
    Round2=ByteSubstitution(Round1);
    Round2=ShiftRow(Round2);
    Round2=MixColumn(Round2);
    Round2=AddRoundKey(Round2,RoundKey2);

//Round3
    Round3=ByteSubstitution(Round2);
    Round3=ShiftRow(Round3);
    Round3=MixColumn(Round3);
    Round3=AddRoundKey(Round3,RoundKey3);

//Round4
    Round4=ByteSubstitution(Round3);
    Round4=ShiftRow(Round4);
    Round4=MixColumn(Round4);
    Round4=AddRoundKey(Round4,RoundKey4);

//Round5
    Round5=ByteSubstitution(Round4);
    Round5=ShiftRow(Round5);
    Round5=MixColumn(Round5);
    Round5=AddRoundKey(Round5,RoundKey5);

//Round6
    Round6=ByteSubstitution(Round5);
```

```
    Round6=ShiftRow(Round6);
    Round6=MixColumn(Round6);
    Round6=AddRoundKey(Round6,RoundKey6);

//Round7
    Round7=ByteSubstitution(Round6);
    Round7=ShiftRow(Round7);
    Round7=MixColumn(Round7);
    Round7=AddRoundKey(Round7,RoundKey7);

//Round8
    Round8=ByteSubstitution(Round7);
    Round8=ShiftRow(Round8);
    Round8=MixColumn(Round8);
    Round8=AddRoundKey(Round8,RoundKey8);

//Round9
    Round9=ByteSubstitution(Round8);
    Round9=ShiftRow(Round9);
    Round9=MixColumn(Round9);
    Round9=AddRoundKey(Round9,RoundKey9);

//Round10
    Round10=ByteSubstitution(Round9);
    Round10=ShiftRow(Round10);
    Round10=AddRoundKey(Round10,RoundKey10);

    int[] ciphertext=MatrixBit(Round10);

    return ciphertext;
  }

//HMAC
  public static int[] HMAC(int[] message, int[] keys) {
    String msg=BinaryToText(message);
    String keyString=BinaryToText(keys);
    int[] output;
      String digest = null;
    try {
    SecretKeySpec key = new SecretKeySpec((keyString).getBytes("UTF-8"),
"HmacSHA256");
    Mac mac = Mac.getInstance("HmacSHA256");
    mac.init(key);

    byte[] bytes = mac.doFinal(msg.getBytes("ASCII"));

    StringBuffer hash = new StringBuffer();

    for (int i=0; i<bytes.length; i++){
      String hex = Integer.toHexString(0xFF & bytes[i]);
      if (hex.length() == 1){
      hash.append('0');}
```

```
        hash.append(hex);}
        digest = hash.toString();
        }catch (UnsupportedEncodingException e){
        }catch (InvalidKeyException e){
        }catch (NoSuchAlgorithmException e){}

        output=HexToBinary(digest);

        return output;
    }

//---KASUMI---//

//FL function. input: 32-bit, key: 32-bit, output: 32-bit
    public static int[] FL (int[] I, int[] KL){
        int[] output;

        int l=I.length;

        int[] L;
        int[] R;
        L=DivideFirst(I,l/2);
        R=DivideSecond(I,l/2);

        int[] KL1;
        int[] KL2;
        KL1=DivideFirst(KL,l/2);
        KL2=DivideSecond(KL,l/2);

        int[] LN;
        int[] RN;

        RN=XOR(R,(CircularLeftRotation(AND(L,KL1),1)));
        LN=XOR(L,(CircularLeftRotation(OR(RN,KL2),1)));

        output=Concatenate(LN,RN);
        return output;
    }

//FL function. input: 32-bit, key: 32-bit, output: 32-bit
    public static int[] FL_inv (int[] I, int[] KL){
        int[] output;

        int l=I.length;

        int[] L;
        int[] R;
        L=DivideFirst(I,l/2);
        R=DivideSecond(I,l/2);

        int[] KL1;
        int[] KL2;
```

```
    KL1=DivideFirst(KL,l/2);
    KL2=DivideSecond(KL,l/2);

    int[] LN;
    int[] RN;

    LN=XOR(L,(CircularLeftRotation(OR(R,KL2),1)));
    RN=XOR(R,(CircularLeftRotation(AND(LN,KL1),1)));

    output=Concatenate(LN,RN);
    return output;
  }

//ZE function. input: 7-bit. output: 9-bit
  public static int[] ZE (int[] I){
    int[] output=new int[9];
    output[0]=0;
    output[1]=0;
    CopyArray(I,output,0,2,7);

    return output;
  }

//TR function. input: 9-bit. output: 7-bit
  public static int[] TR (int[] I){
    int[] output=new int[7];
    CopyArray(I,output,2,0,7);

    return output;
  }

//S-BOX function - S7
  public static int[] S7 (int[] array) {
    int[] output = new int[7];
    int x0=array[6];
    int x1=array[5];
    int x2=array[4];
    int x3=array[3];
    int x4=array[2];
    int x5=array[1];
    int x6=array[0];

    int y0, y1, y2, y3, y4, y5, y6;

    int a=x1&x3;
    int b=x0&x1&x4;
    int c=x2&x5;
    int d=x3&x4&x5;
    int e=x0&x6;
    int f=x1&x6;
    int g=x3&x6;
    int h=x2&x4&x6;
```

```
int i=x1&x5&x6;
int j=x4&x5&x6;

y0=a^x4^b^x5^c^d^x6^e^f^g^h^i^j;
output[6]=y0;

a=x0&x1;
b=x0&x4;
c=x2&x4;
d=x1&x2&x5;
e=x0&x3&x5;
f=x0&x2&x6;
g=x3&x6;
h=x4&x5&x6;

y1=a^b^c^x5^d^e^x6^f^g^h^1;
output[5]=y1;

a=x0&x3;
b=x2&x3;
c=x1&x2&x4;
d=x0&x3&x4;
e=x1&x5;
f=x0&x2&x5;
g=x0&x6;
h=x0&x1&x6;
i=x2&x6;
j=x4&x6;

y2=x0^a^b^c^d^e^f^g^h^i^j^1;
output[4]=y2;

a=x0&x1&x2;
b=x1&x4;
c=x3&x4;
d=x0&x5;
e=x0&x1&x5;
f=x3&x2&x5;
g=x1&x4&x5;
h=x2&x6;
i=x1&x3&x6;

y3=x1^a^b^c^d^e^f^g^h^i;
output[3]=y3;

a=x0&x2;
b=x1&x3;
c=x1&x4;
d=x0&x1&x4;
e=x2&x3&x4;
f=x0&x5;
g=x1&x3&x5;
```

```
    h=x0&x4&x5;
    i=x1&x6;
    j=x3&x6;
    int k=x0&x3&x6;
    int l=x5&x6;

    y4=a^x3^b^c^d^e^f^g^h^i^j^k^l^1;
    output[2]=y4;

    a=x0&x2;
    b=x0&x3;
    c=x1&x2&x3;
    d=x0&x2&x4;
    e=x0&x5;
    f=x2&x5;
    g=x4&x5;
    h=x1&x6;
    i=x1&x2&x6;
    j=x0&x3&x6;
    k=x3&x4&x6;
    l=x2&x5&x6;

    y5=x2^a^b^c^d^e^f^g^h^i^j^k^l^1;
    output[1]=y5;

    a=x1&x2;
    b=x0&x1&x3;
    c=x0&x4;
    d=x1&x5;
    e=x3&x5;
    f=x0&x1&x6;
    g=x2&x3&x6;
    h=x1&x4&x6;
    i=x0&x5&x6;

    y6=a^b^c^d^e^x6^f^g^h^i;
    output[0]=y6;

    return output;
  }

//S7 inverse
  public static int[] S7_inv (int[] array) {
    int total = array[6]*1 + array[5]*2 + array[4]*4 + array[3]*8 + array[2]*16 + array[1]*32 + array[0]*64;
    int[] s7 = {54,50,62,56,22,34,94,96,38,6,63,93,2,18,123,33,55,113,39,114,21,
      67,65,12,47,73,46,27,25,111,124,81,53,9,121,79,52,60,58,48,101,127,40,120,
104,70,71,43,20,122,72,61,23,109,13,100,77,1,16,7,82,10,105,98,117,116,76,11,
89,106,0,125,118,99,86,69,30,57,126,87,112,51,17,5,95,14,90,84,91,8,35,103,32,
```

```java
97,28,66,102,31,26,45,75,4,85,92,37,74,80,49,68,29,115,44,64,107,108,24,110,83,36,78,42,19,15,41,88,119,59,3};

    int result=0;
    for(int x=0; x<128; x++){
      if(total != s7[x]){
      result++;}
      else{break;}
    }

    int[] output=new int[7];

    for(int a=0; a<7; a++){
      int b = result - (int)pow(2,6-a);
      if(b<0){
        output[a]=0;}
      else
        {output[a]=1;}
        result=result - ((int)pow(2,6-a))*output[a];
    }

    return output;
  }

//S-BOX function - S9
  public static int[] S9 (int[] array) {
    int[] output = new int[9];
    int x0=array[8];
    int x1=array[7];
    int x2=array[6];
    int x3=array[5];
    int x4=array[4];
    int x5=array[3];
    int x6=array[2];
    int x7=array[1];
    int x8=array[0];

    int y0;
    int y1;
    int y2;
    int y3;
    int y4;
    int y5;
    int y6;
    int y7;
    int y8;

    int a=x0&x2;
    int b=x2&x5;
    int c=x5&x6;
    int d=x0&x7;
```

```
int e=x1&x7;
int f=x2&x7;
int g=x4&x8;
int h=x5&x8;
int i=x7&x8;

y0=a^x3^b^c^d^e^f^g^h^i^1;
output[8]=y0;

a=x0&x1;
b=x2&x3;
c=x0&x4;
d=x1&x4;
e=x0&x5;
f=x3&x5;
g=x1&x7;
h=x2&x7;
i=x5&x8;

y1=x1^a^b^c^d^e^f^x6^g^h^i^1;
output[7]=y1;

a=x0&x3;
b=x3&x4;
c=x0&x5;
d=x2&x6;
e=x3&x6;
f=x5&x6;
g=x4&x7;
h=x5&x7;
i=x6&x7;
int j=x0&x8;

y2=x1^a^b^c^d^e^f^g^h^i^x8^j^1;
output[6]=y2;

a=x1&x2;
b=x0&x3;
c=x2&x4;
d=x0&x6;
e=x1&x6;
f=x4&x7;
g=x0&x8;
h=x1&x8;
i=x7&x8;

y3=x0^a^b^c^x5^d^e^f^g^h^i;
output[5]=y3;

a=x0&x1;
b=x1&x3;
c=x0&x5;
```

```
d=x3&x6;
e=x0&x7;
f=x6&x7;
g=x1&x8;
h=x2&x8;
i=x3&x8;

y4=a^b^x4^c^d^e^f^g^h^i;
output[4]=y4;

a=x1&x4;
b=x4&x5;
c=x0&x6;
d=x1&x6;
e=x3&x7;
f=x4&x7;
g=x6&x7;
h=x5&x8;
i=x6&x8;
j=x7&x8;

y5=x2^a^b^c^d^e^f^g^h^i^j^1;
output[3]=y5;

a=x2&x3;
b=x1&x5;
c=x2&x5;
d=x4&x5;
e=x3&x6;
f=x4&x6;
g=x5&x6;
h=x1&x8;
i=x3&x8;
j=x5&x8;
int k=x7&x8;

y6=x0^a^b^c^d^e^f^g^x7^h^i^j^k;
output[2]=y6;

a=x0&x1;
b=x0&x2;
c=x1&x2;
d=x0&x3;
e=x2&x3;
f=x4&x5;
g=x2&x6;
h=x3&x6;
i=x2&x7;
j=x5&x7;

y7=a^b^c^x3^d^e^f^g^h^i^j^x8^1;
output[1]=y7;
```

```
    a=x0&x1;
    b=x1&x2;
    c=x3&x4;
    d=x1&x5;
    e=x2&x5;
    f=x1&x6;
    g=x4&x6;
    h=x2&x8;
    i=x3&x8;

    y8=a^x2^b^c^d^e^f^g^x7^h^i;
    output[0]=y8;

    return output;
  }

//S9 inverse
  public static int[] S9_inv (int[] array) {
    int total = array[8]*1 + array[7]*2 + array[6]*4 + array[5]*8 + array[4]*16 + array[3]*32 + array[2]*64 + array[1]*128 + array[0]*256;
    int[] s9 = {167,239,161,379,391,334,9,338,38,226,48,358,452,385,90,
      397,183,253,147,331,415,340,51,362,306,500,262,82,216,159,356,177,
      175,241,489,37,206,17,0,333,44,254,378,58,143,220,81,400,95,3,315,
      245,54,235,218,405,472,264,172,494,371,290,399,76,165,197,395,121,
      257,480,423,212,240,28,462,176,406,507,288,223,501,407,249,265,89,
      186,221,428,164,74,440,196,458,421,350,163,232,158,134,354,13,250,
      491,142,191,69,193,425,152,227,366,135,344,300,276,242,437,320,113,
      278,11,243,87,317,36,93,496,27,487,446,482,41,68,156,457,131,326,
      403,339,20,39,115,442,124,475,384,508,53,112,170,479,151,126,169,
      73,268,279,321,168,364,363,292,46,499,393,327,324,24,456,267,157,
      460,488,426,309,229,439,506,208,271,349,401,434,236,16,209,359,52,
      56,120,199,277,465,416,252,287,246,6,83,305,420,345,153,502,65,61,
      244,282,173,222,418,67,386,368,261,101,476,291,195,430,49,79,166,
      330,280,383,373,128,382,408,155,495,367,388,274,107,459,417,62,454,
      132,225,203,316,234,14,301,91,503,286,424,211,347,307,140,374,35,
      103,125,427,19,214,453,146,498,314,444,230,256,329,198,285,50,116,
      78,410,10,205,510,171,231,45,139,467,29,86,505,32,72,26,342,150,313,
      490,431,238,411,325,149,473,40,119,174,355,185,233,389,71,448,273,
      372,55,110,178,322,12,469,392,369,190,1,109,375,137,181,88,75,308,
      260,484,98,272,370,275,412,111,336,318,4,504,492,259,304,77,337,
      435,21,357,303,332,483,18,47,85,25,497,474,289,100,269,296,478,270,
      106,31,104,433,84,414,486,394,96,99,154,511,148,413,361,409,255,
      162,215,302,201,266,351,343,144,441,365,108,298,251,34,182,509,138,
      210,335,133,311,352,328,141,396,346,123,319,450,281,429,228,443,
      481,92,404,485,422,248,297,23,213,130,466,22,217,283,70,294,360,
      419,127,312,377,7,468,194,2,117,295,463,258,224,447,247,187,80,398,
      284,353,105,390,299,471,470,184,57,200,348,63,204,188,33,451,97,
      30,310,219,94,160,129,493,64,179,263,102,189,207,114,402,438,477,
      387,122,192,42,381,5,145,118,180,449,293,323,136,380,43,66,60,455,
      341,445,202,432,8,237,15,376,436,464,59,461};

```

```
    int result=0;
    for(int x=0; x<512; x++){
      if(total != s9[x]){
        result++;}
      else{
        break;}
    }

    int[] output=new int[9];
    for(int a=0; a<9; a++){
      int b = result - (int)pow(2,8-a);
      if(b<0){
        output[a]=0;}
      else{
        output[a]=1;}
      result=result - ((int)pow(2,8-a))*output[a];
    }

    return output;
  }

//FI function. input: 16-bit, key: 16-bit, output: 16-bit
  public static int[] FI (int[] I, int[] KI){
    int[] output;

    int[] L0;
    int[] R0;
    L0=DivideFirst(I,9);
    R0=DivideSecond(I,9);

    int[] KI1;
    int[] KI2;
    KI1=DivideFirst(KI,7);
    KI2=DivideSecond(KI,7);

    int[] L1;
    int[] R1;
    int[] L2;
    int[] R2;
    int[] L3;
    int[] R3;
    int[] L4;
    int[] R4;

    L1=R0;
    R1=XOR(S9(L0),ZE(R0));
    L2=XOR(R1,KI2);
    R2=XOR(XOR(S7(L1),TR(R1)),KI1);
    L3=R2;
    R3=XOR(S9(L2),ZE(R2));
    L4=XOR(S7(L3),TR(R3));
    R4=R3;
```

```
        output=Concatenate(L4,R4);
        return output;
    }

//Inverse of FI function. input: 16-bit, key: 16-bit, output: 16-bit
    public static int[] FI_inv (int[] I, int[] KI){
        int[] output;

        int[] L4;
        int[] R4;
        L4=DivideFirst(I,7);
        R4=DivideSecond(I,7);

        int[] KI1;
        int[] KI2;
        KI1=DivideFirst(KI,7);
        KI2=DivideSecond(KI,7);

        int[] L1, R1, L2, R2, L3, R3, L0, R0;

        R3=R4;
        L3=S7_inv(XOR(L4,TR(R3)));
        R2=L3;
        L2=S9_inv(XOR(R3,ZE(R2)));
        R1=XOR(L2,KI2);
        L1=S7_inv(XOR(XOR(R2,TR(R1)),KI1));
        R0=L1;
        L0=S9_inv(XOR(R1,ZE(R0)));

        output=Concatenate(L0,R0);
        return output;
    }

//FO function. input: 32-bit, key: 48-bit two keys, output: 32-bit
    public static int[] FO (int[] I, int[] KO, int[] KI){
        int[] output;

        int l=I.length;
        int t=KO.length;

        int[] L0, R0, L1, R1, L2, R2, L3, R3;

        L0=DivideFirst(I,l/2);
        R0=DivideSecond(I,l/2);

        int[] KO1, KO2, KO2q, KO3;
        KO1=DivideFirst(KO,t/3);
        KO2q=DivideSecond(KO,t/3);
        KO2=DivideFirst(KO2q,t/3);
        KO3=DivideSecond(KO2q,t/3);
```

```
    int[] KI1, KI2, KI2q, KI3;
    KI1=DivideFirst(KI,t/3);
    KI2q=DivideSecond(KI,t/3);
    KI2=DivideFirst(KI2q,t/3);
    KI3=DivideSecond(KI2q,t/3);

    R1=XOR(FI(XOR(L0,KO1),KI1),R0);
    L1=R0;
    R2=XOR(FI(XOR(L1,KO2),KI2),R1);
    L2=R1;
    R3=XOR(FI(XOR(L2,KO3),KI3),R2);
    L3=R2;

    output=Concatenate(L3,R3);
    return output;
  }

//Inverse of FO function. input: 32-bit, key: 48-bit two keys, output: 32-bit
  public static int[] FO_inv (int[] I, int[] KO, int[] KI){
    int[] output;

    int l=I.length;
    int t=KO.length;

    int[] L0, R0, L1, R1, L2, R2, L3, R3;

    L3=DivideFirst(I,l/2);
    R3=DivideSecond(I,l/2);

    int[] KO1, KO2, KO2q, KO3;
    KO1=DivideFirst(KO,t/3);
    KO2q=DivideSecond(KO,t/3);
    KO2=DivideFirst(KO2q,t/3);
    KO3=DivideSecond(KO2q,t/3);

    int[] KI1, KI2, KI2q, KI3;
    KI1=DivideFirst(KI,t/3);
    KI2q=DivideSecond(KI,t/3);
    KI2=DivideFirst(KI2q,t/3);
    KI3=DivideSecond(KI2q,t/3);

    int[] temp;
    R2=L3;
    temp=XOR(R3,R2);
    L2=XOR(FI_inv(temp,KI3),KO3);
    R1=L2;
    temp=XOR(R2,R1);
    L1=XOR(FI_inv(temp,KI2),KO2);
    R0=L1;
    temp=XOR(R1,R0);
    L0=XOR(FI_inv(temp,KI1),KO1);

```

```
    output=Concatenate(L0,R0);
    return output;
  }

//fi function for odd rounds. input; 32-bit, keys: 32-bit, 48-bit, 48-bit.
  public static int[] fi_odd (int[] I, int[] KL, int[] KO, int[] KI){
    int[] output;
    output=FO(FL(I,KL),KO,KI);

    return output;
  }

//fi function for odd rounds. input; 32-bit, keys: 32-bit, 48-bit, 48-bit.
  public static int[] fi_odd_inv (int[] I, int[] KL, int[] KO, int[] KI){
    int[] output;
    int[] temp;

    temp=FO_inv(I,KO,KI);
    output=FL_inv(temp,KL);

    return output;
  }

//fi function for even rounds. input; 32-bit, keys: 32-bit, 48-bit, 48-bit.
  public static int[] fi_even (int[] I, int[] KL, int[] KO, int[] KI){
    int[] output;
    output=FL(FO(I,KO,KI),KL);

    return output;
  }

//fi function for even rounds. input; 32-bit, keys: 32-bit, 48-bit, 48-bit.
  public static int[] fi_even_inv (int[] I, int[] KL, int[] KO, int[] KI){
    int[] output;
    int[] temp;

    temp=FL_inv(I,KL);
    output=FO_inv(temp,KO,KI);

    return output;
  }

//Kasumi encryption. Input: 64-bit, key: 128-bit, output: 64-bit.
  public static int[] KASUMI_enc (int[] I, int[] K){

//Divide keys into 8 16-bit ki's
    int[] k1 = new int[16];
    CopyArrayString(K,k1,0,0,16);
    int[] k2 = new int[16];
    CopyArrayString(K,k2,16,0,16);
    int[] k3 = new int[16];
    CopyArrayString(K,k3,32,0,16);
```

```
	int[] k4 = new int[16];
	CopyArrayString(K,k4,48,0,16);
	int[] k5 = new int[16];
	CopyArrayString(K,k5,64,0,16);
	int[] k6 = new int[16];
	CopyArrayString(K,k6,80,0,16);
	int[] k7 = new int[16];
	CopyArrayString(K,k7,96,0,16);
	int[] k8 = new int[16];
	CopyArrayString(K,k8,112,0,16);

//Binary values of each constant ci's
	int[] c1=HexToBinaryArrayKey("0123");
	int[] c2=HexToBinaryArrayKey("4567");
	int[] c3=HexToBinaryArrayKey("89AB");
	int[] c4=HexToBinaryArrayKey("CDEF");
	int[] c5=HexToBinaryArrayKey("FEDC");
	int[] c6=HexToBinaryArrayKey("BA98");
	int[] c7=HexToBinaryArrayKey("7654");
	int[] c8=HexToBinaryArrayKey("3210");

//Round subkeys of KLi1
	int[] KL11=CircularLeftRotation(k1,1);
	int[] KL21=CircularLeftRotation(k2,1);
	int[] KL31=CircularLeftRotation(k3,1);
	int[] KL41=CircularLeftRotation(k4,1);
	int[] KL51=CircularLeftRotation(k5,1);
	int[] KL61=CircularLeftRotation(k6,1);
	int[] KL71=CircularLeftRotation(k7,1);
	int[] KL81=CircularLeftRotation(k8,1);

//Round subkeys of KLi2
	int[] KL12=XOR(k3,c3);
	int[] KL22=XOR(k4,c4);
	int[] KL32=XOR(k5,c5);
	int[] KL42=XOR(k6,c6);
	int[] KL52=XOR(k7,c7);
	int[] KL62=XOR(k8,c8);
	int[] KL72=XOR(k1,c1);
	int[] KL82=XOR(k2,c2);

//Round subkeys of KLi
	int[] KL1=Concatenate(KL11,KL12);
	int[] KL2=Concatenate(KL21,KL22);
	int[] KL3=Concatenate(KL31,KL32);
	int[] KL4=Concatenate(KL41,KL42);
	int[] KL5=Concatenate(KL51,KL52);
	int[] KL6=Concatenate(KL61,KL62);
	int[] KL7=Concatenate(KL71,KL72);
	int[] KL8=Concatenate(KL81,KL82);

//Round subkeys of KOi1
```

```
int[] KO11=CircularLeftRotation(k2,5);
int[] KO21=CircularLeftRotation(k3,5);
int[] KO31=CircularLeftRotation(k4,5);
int[] KO41=CircularLeftRotation(k5,5);
int[] KO51=CircularLeftRotation(k6,5);
int[] KO61=CircularLeftRotation(k7,5);
int[] KO71=CircularLeftRotation(k8,5);
int[] KO81=CircularLeftRotation(k1,5);

//Round subkeys of KOi2
int[] KO12=CircularLeftRotation(k6,8);
int[] KO22=CircularLeftRotation(k7,8);
int[] KO32=CircularLeftRotation(k8,8);
int[] KO42=CircularLeftRotation(k1,8);
int[] KO52=CircularLeftRotation(k2,8);
int[] KO62=CircularLeftRotation(k3,8);
int[] KO72=CircularLeftRotation(k4,8);
int[] KO82=CircularLeftRotation(k5,8);

//Round subkeys of KOi3
int[] KO13=CircularLeftRotation(k7,13);
int[] KO23=CircularLeftRotation(k8,13);
int[] KO33=CircularLeftRotation(k1,13);
int[] KO43=CircularLeftRotation(k2,13);
int[] KO53=CircularLeftRotation(k3,13);
int[] KO63=CircularLeftRotation(k4,13);
int[] KO73=CircularLeftRotation(k5,13);
int[] KO83=CircularLeftRotation(k6,13);

//Round subkeys of KOi
int[] KO1=Concatenate(Concatenate(KO11,KO12),KO13);
int[] KO2=Concatenate(Concatenate(KO21,KO22),KO23);
int[] KO3=Concatenate(Concatenate(KO31,KO32),KO33);
int[] KO4=Concatenate(Concatenate(KO41,KO42),KO43);
int[] KO5=Concatenate(Concatenate(KO51,KO52),KO53);
int[] KO6=Concatenate(Concatenate(KO61,KO62),KO63);
int[] KO7=Concatenate(Concatenate(KO71,KO72),KO73);
int[] KO8=Concatenate(Concatenate(KO81,KO82),KO83);

//Round subkeys of KIi1
int[] KI11=XOR(k5,c5);
int[] KI21=XOR(k6,c6);
int[] KI31=XOR(k7,c7);
int[] KI41=XOR(k8,c8);
int[] KI51=XOR(k1,c1);
int[] KI61=XOR(k2,c2);
int[] KI71=XOR(k3,c3);
int[] KI81=XOR(k4,c4);

//Round subkeys of KIi2
int[] KI12=XOR(k4,c4);
int[] KI22=XOR(k5,c5);
```

```
    int[] KI32=XOR(k6,c6);
    int[] KI42=XOR(k7,c7);
    int[] KI52=XOR(k8,c8);
    int[] KI62=XOR(k1,c1);
    int[] KI72=XOR(k2,c2);
    int[] KI82=XOR(k3,c3);

//Round subkeys of KIi3
    int[] KI13=XOR(k8,c8);
    int[] KI23=XOR(k1,c1);
    int[] KI33=XOR(k2,c2);
    int[] KI43=XOR(k3,c3);
    int[] KI53=XOR(k4,c4);
    int[] KI63=XOR(k5,c5);
    int[] KI73=XOR(k6,c6);
    int[] KI83=XOR(k7,c7);

//Round subkeys of KIi
    int[] KI1=Concatenate(Concatenate(KI11,KI12),KI13);
    int[] KI2=Concatenate(Concatenate(KI21,KI22),KI23);
    int[] KI3=Concatenate(Concatenate(KI31,KI32),KI33);
    int[] KI4=Concatenate(Concatenate(KI41,KI42),KI43);
    int[] KI5=Concatenate(Concatenate(KI51,KI52),KI53);
    int[] KI6=Concatenate(Concatenate(KI61,KI62),KI63);
    int[] KI7=Concatenate(Concatenate(KI71,KI72),KI73);
    int[] KI8=Concatenate(Concatenate(KI81,KI82),KI83);

    int[] L0, R0, L1, R1, L2, R2, L3, R3, L4, R4, L5, R5, L6, R6, L7, R7, L8, R8;

    L0=DivideFirst(I,32);
    R0=DivideSecond(I,32);

//Round 1:
    R1=L0;
    L1=XOR(R0,fi_odd(L0,KL1,KO1,KI1));

//Round 2:
    R2=L1;
    L2=XOR(R1,fi_even(L1,KL2,KO2,KI2));

//Round 3:
    R3=L2;
    L3=XOR(R2,fi_odd(L2,KL3,KO3,KI3));

//Round 4:
    R4=L3;
    L4=XOR(R3,fi_even(L3,KL4,KO4,KI4));

//Round 5:
    R5=L4;
    L5=XOR(R4,fi_odd(L4,KL5,KO5,KI5));
```

```
//Round 6:
        R6=L5;
        L6=XOR(R5,fi_even(L5,KL6,KO6,KI6));

//Round 7:
        R7=L6;
        L7=XOR(R6,fi_odd(L6,KL7,KO7,KI7));

//Round 8:
        R8=L7;
        L8=XOR(R7,fi_even(L7,KL8,KO8,KI8));

        int[] output;
        output=Concatenate(L8,R8);

        return output;
    }

//Kasumi decryption. Input: 64-bit, key: 128-bit, output: 64-bit.
    public static int[] KASUMI_dec (int[] I, int[] K){

//Divide keys into 8 16-bit ki's
        int[] k1 = new int[16];
        CopyArrayString(K,k1,0,0,16);
        int[] k2 = new int[16];
        CopyArrayString(K,k2,16,0,16);
        int[] k3 = new int[16];
        CopyArrayString(K,k3,32,0,16);
        int[] k4 = new int[16];
        CopyArrayString(K,k4,48,0,16);
        int[] k5 = new int[16];
        CopyArrayString(K,k5,64,0,16);
        int[] k6 = new int[16];
        CopyArrayString(K,k6,80,0,16);
        int[] k7 = new int[16];
        CopyArrayString(K,k7,96,0,16);
        int[] k8 = new int[16];
        CopyArrayString(K,k8,112,0,16);

//Binary values of each constant ci's
        int[] c1=HexToBinaryArrayKey("0123");
        int[] c2=HexToBinaryArrayKey("4567");
        int[] c3=HexToBinaryArrayKey("89AB");
        int[] c4=HexToBinaryArrayKey("CDEF");
        int[] c5=HexToBinaryArrayKey("FEDC");
        int[] c6=HexToBinaryArrayKey("BA98");
        int[] c7=HexToBinaryArrayKey("7654");
        int[] c8=HexToBinaryArrayKey("3210");

//Round subkeys of KLi1
        int[] KL11=CircularLeftRotation(k1,1);
        int[] KL21=CircularLeftRotation(k2,1);
```

```
    int[] KL31=CircularLeftRotation(k3,1);
    int[] KL41=CircularLeftRotation(k4,1);
    int[] KL51=CircularLeftRotation(k5,1);
    int[] KL61=CircularLeftRotation(k6,1);
    int[] KL71=CircularLeftRotation(k7,1);
    int[] KL81=CircularLeftRotation(k8,1);

//Round subkeys of KLi2
    int[] KL12=XOR(k3,c3);
    int[] KL22=XOR(k4,c4);
    int[] KL32=XOR(k5,c5);
    int[] KL42=XOR(k6,c6);
    int[] KL52=XOR(k7,c7);
    int[] KL62=XOR(k8,c8);
    int[] KL72=XOR(k1,c1);
    int[] KL82=XOR(k2,c2);

//Round subkeys of KLi
    int[] KL1=Concatenate(KL11,KL12);
    int[] KL2=Concatenate(KL21,KL22);
    int[] KL3=Concatenate(KL31,KL32);
    int[] KL4=Concatenate(KL41,KL42);
    int[] KL5=Concatenate(KL51,KL52);
    int[] KL6=Concatenate(KL61,KL62);
    int[] KL7=Concatenate(KL71,KL72);
    int[] KL8=Concatenate(KL81,KL82);

//Round subkeys of KOi1
    int[] KO11=CircularLeftRotation(k2,5);
    int[] KO21=CircularLeftRotation(k3,5);
    int[] KO31=CircularLeftRotation(k4,5);
    int[] KO41=CircularLeftRotation(k5,5);
    int[] KO51=CircularLeftRotation(k6,5);
    int[] KO61=CircularLeftRotation(k7,5);
    int[] KO71=CircularLeftRotation(k8,5);
    int[] KO81=CircularLeftRotation(k1,5);

//Round subkeys of KOi2
    int[] KO12=CircularLeftRotation(k6,8);
    int[] KO22=CircularLeftRotation(k7,8);
    int[] KO32=CircularLeftRotation(k8,8);
    int[] KO42=CircularLeftRotation(k1,8);
    int[] KO52=CircularLeftRotation(k2,8);
    int[] KO62=CircularLeftRotation(k3,8);
    int[] KO72=CircularLeftRotation(k4,8);
    int[] KO82=CircularLeftRotation(k5,8);

//Round subkeys of KOi3
    int[] KO13=CircularLeftRotation(k7,13);
    int[] KO23=CircularLeftRotation(k8,13);
    int[] KO33=CircularLeftRotation(k1,13);
    int[] KO43=CircularLeftRotation(k2,13);
```

```
    int[] KO53=CircularLeftRotation(k3,13);
    int[] KO63=CircularLeftRotation(k4,13);
    int[] KO73=CircularLeftRotation(k5,13);
    int[] KO83=CircularLeftRotation(k6,13);

//Round subkeys of KOi
    int[] KO1=Concatenate(Concatenate(KO11,KO12),KO13);
    int[] KO2=Concatenate(Concatenate(KO21,KO22),KO23);
    int[] KO3=Concatenate(Concatenate(KO31,KO32),KO33);
    int[] KO4=Concatenate(Concatenate(KO41,KO42),KO43);
    int[] KO5=Concatenate(Concatenate(KO51,KO52),KO53);
    int[] KO6=Concatenate(Concatenate(KO61,KO62),KO63);
    int[] KO7=Concatenate(Concatenate(KO71,KO72),KO73);
    int[] KO8=Concatenate(Concatenate(KO81,KO82),KO83);

//Round subkeys of KIi1
    int[] KI11=XOR(k5,c5);
    int[] KI21=XOR(k6,c6);
    int[] KI31=XOR(k7,c7);
    int[] KI41=XOR(k8,c8);
    int[] KI51=XOR(k1,c1);
    int[] KI61=XOR(k2,c2);
    int[] KI71=XOR(k3,c3);
    int[] KI81=XOR(k4,c4);

//Round subkeys of KIi2
    int[] KI12=XOR(k4,c4);
    int[] KI22=XOR(k5,c5);
    int[] KI32=XOR(k6,c6);
    int[] KI42=XOR(k7,c7);
    int[] KI52=XOR(k8,c8);
    int[] KI62=XOR(k1,c1);
    int[] KI72=XOR(k2,c2);
    int[] KI82=XOR(k3,c3);

//Round subkeys of KIi3
    int[] KI13=XOR(k8,c8);
    int[] KI23=XOR(k1,c1);
    int[] KI33=XOR(k2,c2);
    int[] KI43=XOR(k3,c3);
    int[] KI53=XOR(k4,c4);
    int[] KI63=XOR(k5,c5);
    int[] KI73=XOR(k6,c6);
    int[] KI83=XOR(k7,c7);

//Round subkeys of KIi
    int[] KI1=Concatenate(Concatenate(KI11,KI12),KI13);
    int[] KI2=Concatenate(Concatenate(KI21,KI22),KI23);
    int[] KI3=Concatenate(Concatenate(KI31,KI32),KI33);
    int[] KI4=Concatenate(Concatenate(KI41,KI42),KI43);
    int[] KI5=Concatenate(Concatenate(KI51,KI52),KI53);
    int[] KI6=Concatenate(Concatenate(KI61,KI62),KI63);
```

```
    int[] KI7=Concatenate(Concatenate(KI71,KI72),KI73);
    int[] KI8=Concatenate(Concatenate(KI81,KI82),KI83);

    int[] L0, R0, L1, R1, L2, R2, L3, R3, L4, R4, L5, R5, L6, R6, L7, R7, L8, R8;

    L8=DivideFirst(I,32);
    R8=DivideSecond(I,32);

//Round 1:
    L7=R8;
    R7=XOR(fi_even(L7,KL8,KO8,KI8),L8);

//Round 2:
    L6=R7;
    R6=XOR(fi_odd(L6,KL7,KO7,KI7),L7);

//Round 3:
    L5=R6;
    R5=XOR(fi_even(L5,KL6,KO6,KI6),L6);

//Round 4:
    L4=R5;
    R4=XOR(fi_odd(L4,KL5,KO5,KI5),L5);

//Round 5:
    L3=R4;
    R3=XOR(fi_even(L3,KL4,KO4,KI4),L4);

//Round 6:
    L2=R3;
    R2=XOR(fi_odd(L2,KL3,KO3,KI3),L3);

//Round 7:
    L1=R2;
    R1=XOR(fi_even(L1,KL2,KO2,KI2),L2);

//Round 8:
    L0=R1;
    R0=XOR(fi_odd(L0,KL1,KO1,KI1),L1);

    int[] output;
    output=Concatenate(L0,R0);

    return output;
  }


//---MILENAGE FUNCTIONS---//


//Milenage Functions - MAC
  public static int[] MAC (int[] RAND, int[] K, int[] OP, int[] SQN, int[] AMF){
```

```
    int[] MAC=new int[64];
    int[] OPc=XOR(OP,AES(OP,K));
    int[] TEMP=AES(XOR(RAND,OPc),K);

    int[] IN1=new int[128];
    CopyArray(SQN,IN1,0,0,48);
    CopyArray(AMF,IN1,0,48,16);
    CopyArray(SQN,IN1,0,64,48);
    CopyArray(AMF,IN1,0,112,16);

    int[] c1=new int[128];

    int r1;
    r1=64;

    int[] OUT1;

    int[]out10=CircularLeftRotation(XOR(IN1,OPc),r1);
    int[]out11=XOR(TEMP,out10);
    int[]out12=XOR(out11,c1);
    OUT1=XOR(AES(out12,K),OPc);

    CopyArray(OUT1,MAC,0,0,64);

    return MAC;
  }

//Milenage Functions - RES
  public static int[] RES (int[] RAND, int[] K, int[] OP, int[] SQN, int[] AMF){
    int[] RES=new int[64];
    int[] OPc=XOR(OP,AES(OP,K));
    int[] TEMP=AES(XOR(RAND,OPc),K);

    int[] IN1=new int[128];
    CopyArray(SQN,IN1,0,0,48);
    CopyArray(AMF,IN1,0,48,16);
    CopyArray(SQN,IN1,0,64,48);
    CopyArray(AMF,IN1,0,112,16);

    int[] c2=new int[128];
    c2[127]=1;
    int r2;
    r2=0;
    int[] OUT2;

    int[] tmp=XOR(TEMP,OPc);

    int[]out20=CircularLeftRotation(tmp,r2);
    int[]out21=XOR(out20,c2);
    OUT2=XOR(AES(out21,K),OPc);

    CopyArray(OUT2,RES,64,0,64);
```

```
    return RES;
}

//Milenage Functions - CK
public static int[] CK (int[] RAND, int[] K, int[] OP, int[] SQN, int[] AMF){
    int[] CK=new int[128];
    int[] OPc=XOR(OP,AES(OP,K));
    int[] TEMP=AES(XOR(RAND,OPc),K);

    int[] IN1=new int[128];
    CopyArray(SQN,IN1,0,0,48);
    CopyArray(AMF,IN1,0,48,16);
    CopyArray(SQN,IN1,0,64,48);
    CopyArray(AMF,IN1,0,112,16);

    int[] c3=new int[128];
    c3[126]=1;
    int r3;
    r3=32;
    int[] OUT3;

    int[] tmp=XOR(TEMP,OPc);

    int[] out30=CircularLeftRotation(tmp,r3);
    int[] out31=XOR(out30,c3);
    OUT3=XOR(AES(out31,K),OPc);

    CopyArray(OUT3,CK,0,0,128);

    return CK;
}

//Milenage Functions - IK
public static int[] IK (int[] RAND, int[] K, int[] OP, int[] SQN, int[] AMF){
    int[] IK=new int[128];
    int[] OPc=XOR(OP,AES(OP,K));
    int[] TEMP=AES(XOR(RAND,OPc),K);

    int[] IN1=new int[128];
    CopyArray(SQN,IN1,0,0,48);
    CopyArray(AMF,IN1,0,48,16);
    CopyArray(SQN,IN1,0,64,48);
    CopyArray(AMF,IN1,0,112,16);

    int[] c4=new int[128];

    c4[125]=1;

    int r4;
    r4=64;

```

```
    int[] OUT4;

    int[] tmp=XOR(TEMP,OPc);

    int[] out40=CircularLeftRotation(tmp,r4);
    int[] out41=XOR(out40,c4);
    OUT4=XOR(AES(out41,K),OPc);

    CopyArray(OUT4,IK,0,0,128);

    return IK;
  }

//Milenage Functions - AK
  public static int[] AK (int[] RAND, int[] K, int[] OP, int[] SQN, int[] AMF){
    int[] AK=new int[48];
    int[] OPc=XOR(OP,AES(OP,K));
    int[] TEMP=AES(XOR(RAND,OPc),K);

    int[] IN1=new int[128];
    CopyArray(SQN,IN1,0,0,48);
    CopyArray(AMF,IN1,0,48,16);
    CopyArray(SQN,IN1,0,64,48);
    CopyArray(AMF,IN1,0,112,16);

    int[] c2=new int[128];

    c2[127]=1;

    int r2;
    r2=0;

    int[] OUT2;

    int[] tmp=XOR(TEMP,OPc);

    int[]out20=CircularLeftRotation(tmp,r2);
    int[]out21=XOR(out20,c2);
    OUT2=XOR(AES(out21,K),OPc);

    CopyArray(OUT2,AK,0,0,48);

    return AK;
  }

}
```

APPENDIX B – Output of Demonstration

```
run: (INPUT.java)
Key_hex is: 18b7ac920d5bcef54a8107e976a4d3c8
OP_hex is: 0b8a475bc123d60177a29ac3615834aa
IMSI is: 5712919082
SQN is: 8e4c2be3b530
Checkpoints are ready.
BUILD SUCCESSFUL (total time: 1 second)
```

```
run: (UE.java)
----------|User Equipment|----------
----------------|UE|----------------

Press enter to proceed. (Write STOP to exit.)

Choose what to send for ID:
Write 'P1' to send IMSI
Write 'P2' to send new pseudonym
Write 'P3' to send used pseudonym
p1

|UE| IMSI: DNA 5712919082
|UE| Attachment request is sent to SN.

-------------------------------------------

Press enter to proceed. (Write STOP to exit.)

|UE| AV is received from SN.

|UE| Extracting RAND and AUTN..
|UE| RAND and AUTN are extracted.

|UE| Extracting and calculating MAC.
|UE| Checking MAC..
|UE| MAC is verified.

|UE| Extracting AMF..
|UE| AMF is extracted.

|UE| Checking AMF..
|UE| This RAND is to be used for creating new key.

|UE| Preparing RES..
|UE| RES is prepared.

|UE| RES is sent to SN.

-------------------------------------------

Press enter to proceed. (Write STOP to exit.)

|UE| Result for RES challenge is received from SN.
```

```
|UE| Checking result..
|UE| Authentication succeeded.
------------------------------------------

|UE| IMSI: DNA 5712919082
|UE| Attachment request is sent to SN.

------------------------------------------

Press enter to proceed. (Write STOP to exit.)

|UE| AV is received from SN.

|UE| Extracting RAND and AUTN..
|UE| RAND and AUTN are extracted.

|UE| Extracting and calculating MAC.
|UE| Checking MAC..
|UE| MAC is verified.

|UE| Extracting AMF..
|UE| AMF is extracted.

|UE| Checking AMF..
|UE| This RAND includes pseudonym.

|UE| Extracting Pseudonym..
|UE| Pseudonym is extracted.
|UE| Pseudonym is 3613856892

|UE| Preparing RES..
|UE| RES is prepared.

|UE| RES is sent to SN.

------------------------------------------

Press enter to proceed. (Write STOP to exit.)

|UE| Result for RES challenge is received from SN.

|UE| Checking result..
|UE| Authentication succeeded.

------------------------------------------

Press enter to proceed. (Write STOP to exit.)

Choose what to send for ID:
Write 'P1' to send IMSI
Write 'P2' to send new pseudonym
Write 'P3' to send used pseudonym
p2

|UE| IMSI: DNA 3613856892
|UE| Attachment request is sent to SN.

------------------------------------------
```

```
Press enter to proceed. (Write STOP to exit.)

|UE| AV is received from SN.

|UE| Extracting RAND and AUTN..
|UE| RAND and AUTN are extracted.

|UE| Extracting and calculating MAC.
|UE| Checking MAC..
|UE| MAC is verified.

|UE| Extracting AMF..
|UE| AMF is extracted.

|UE| Checking AMF..
|UE| This RAND includes pseudonym.

|UE| Extracting Pseudonym..
|UE| Pseudonym is extracted.
|UE| Pseudonym is 2890913730

|UE| Preparing RES..
|UE| RES is prepared.

|UE| RES is sent to SN.

------------------------------------------

Press enter to proceed. (Write STOP to exit.)

|UE| Result for RES challenge is received from SN.

|UE| Checking result..
|UE| Authentication succeeded.

------------------------------------------

Press enter to proceed. (Write STOP to exit.)

Choose what to send for ID:
Write 'P1' to send IMSI
Write 'P2' to send new pseudonym
Write 'P3' to send used pseudonym
p3

|UE| IMSI: DNA 3613856892
|UE| Attachment request is sent to SN.

------------------------------------------

Press enter to proceed. (Write STOP to exit.)

|UE| AV is received from SN.

|UE| Extracting RAND and AUTN..
|UE| RAND and AUTN are extracted.

|UE| Extracting and calculating MAC.
|UE| Checking MAC..
```

```
|UE| MAC is verified.

|UE| Extracting AMF..
|UE| AMF is extracted.

|UE| Checking AMF..
|UE| This RAND doesn't include pseudonym and isn't to be used for
creating new key.

|UE| Preparing RES..
|UE| RES is prepared.

|UE| RES is sent to SN.

------------------------------------------

Press enter to proceed. (Write STOP to exit.)

|UE| Result for RES challenge is received from SN.

|UE| Checking result..
|UE| Authentication succeeded.

------------------------------------------

Press enter to proceed. (Write STOP to exit.)
stop
BUILD SUCCESSFUL (total time: 3 minutes 31 seconds)
```

run: (SN.java)

```
----------|Serving Network|----------
----------------|SN|-----------------

Press enter to proceed. (Write STOP to exit.)

|SN| Attach attempt from DNA 5712919082

|SN| Attachment request is sent to HN.

------------------------------------------

Press enter to proceed. (Write STOP to exit.)

|SN| Authentication Vector from HN.

|SN| Extracting XRES..
|SN| XRES is extracted.

|SN| Preparing AV for UE..
|SN| AV for UE is prepared.

|SN| AV is sent to UE.

------------------------------------------

Press enter to proceed. (Write STOP to exit.)

|SN| RES is received from UE.
```

```
|SN| Checking if RES matches XRES..
|SN| RES challenge succeeded.

|SN| Result of RES challenge is sent both to UE and HN.

------------------------------------------

Press enter to proceed. (Write STOP to exit.)

|SN| Attach attempt from DNA 5712919082

|SN| Attachment request is sent to HN.

------------------------------------------

Press enter to proceed. (Write STOP to exit.)

|SN| Authentication Vector from HN.

|SN| Extracting XRES..
|SN| XRES is extracted.

|SN| Preparing AV for UE..
|SN| AV for UE is prepared.

|SN| AV is sent to UE.

------------------------------------------

Press enter to proceed. (Write STOP to exit.)

|SN| RES is received from UE.

|SN| Checking if RES matches XRES..
|SN| RES challenge succeeded.

|SN| Result of RES challenge is sent both to UE and HN.

------------------------------------------

Press enter to proceed. (Write STOP to exit.)

|SN| Attach attempt from DNA 3613856892

|SN| Attachment request is sent to HN.

------------------------------------------

Press enter to proceed. (Write STOP to exit.)

|SN| Authentication Vector from HN.

|SN| Extracting XRES..
|SN| XRES is extracted.

|SN| Preparing AV for UE..
|SN| AV for UE is prepared.
```

```
|SN| AV is sent to UE.

------------------------------------------

Press enter to proceed. (Write STOP to exit.)

|SN| RES is received from UE.

|SN| Checking if RES matches XRES..
|SN| RES challenge succeeded.

|SN| Result of RES challenge is sent both to UE and HN.

------------------------------------------

Press enter to proceed. (Write STOP to exit.)

|SN| Attach attempt from DNA 3613856892

|SN| Attachment request is sent to HN.

------------------------------------------

Press enter to proceed. (Write STOP to exit.)

|SN| Authentication Vector from HN.

|SN| Extracting XRES..
|SN| XRES is extracted.

|SN| Preparing AV for UE..
|SN| AV for UE is prepared.

|SN| AV is sent to UE.

------------------------------------------

Press enter to proceed. (Write STOP to exit.)

|SN| RES is received from UE.

|SN| Checking if RES matches XRES..
|SN| RES challenge succeeded.

|SN| Result of RES challenge is sent both to UE and HN.

------------------------------------------

Press enter to proceed. (Write STOP to exit.)
stop
BUILD SUCCESSFUL (total time: 3 minutes 33 seconds)
```

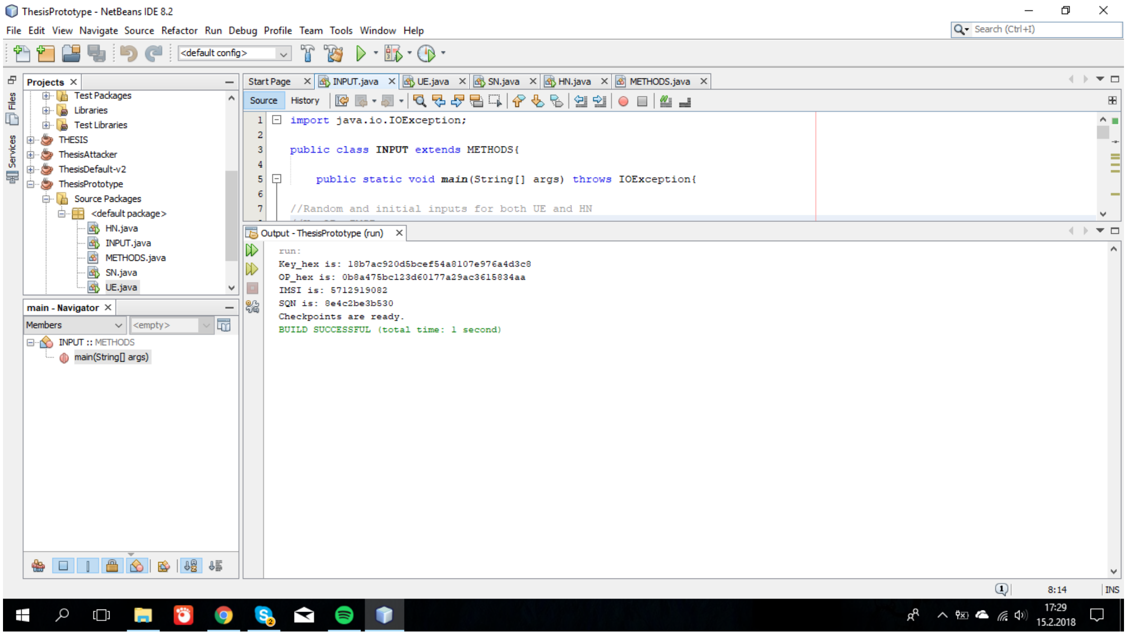
run: (HN.java)

```
----------|Home Network|----------
--------------|HN|----------------

Press enter to proceed. (Write STOP to exit.)

|HN| Attach attempt from DNA 5712919082

|HN| Checking IMSI..
|HN| IMSI is valid.

|HN| R1-type AV is required.

|HN| Creating R1-type AV..
|HN| R1-type AV is created.

|HN| R1-type AV is sent to SN.

------------------------------------------

Press enter to proceed. (Write STOP to exit.)

|HN| Result for RES challenge is received from SN.

|HN| Checking result..
|HN| Authentication succeeded.

------------------------------------------

Press enter to proceed. (Write STOP to exit.)

|HN| Attach attempt from DNA 5712919082

|HN| Checking IMSI..
|HN| IMSI is valid.

|HN| Creating pseudonym..
|HN| New Pseudonym is created.
|HN| New Pseudonym is 3613856892

|HN| R2-type AV is required.

|HN| Creating R2-type AV..
|HN| R2-type AV is created.

|HN| R2-type AV is sent to SN.

------------------------------------------

Press enter to proceed. (Write STOP to exit.)

|HN| Response result from SN.

|HN| Checking response..
|HN| Result for RES challenge is received from SN.

|HN| Checking result..
```

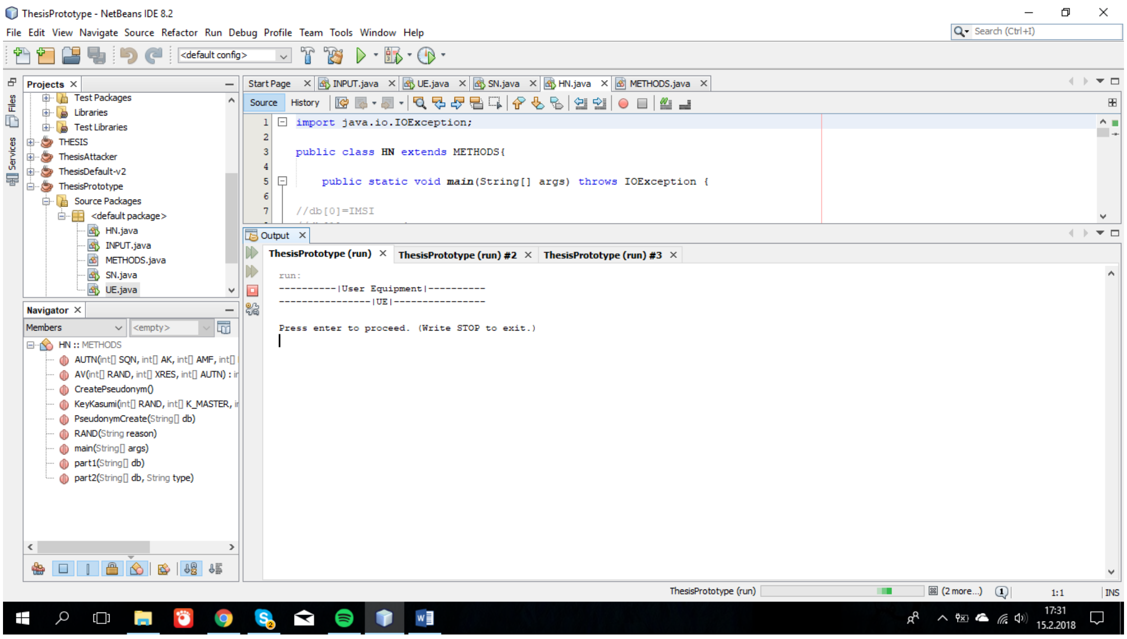
```
|HN| Authentication succeeded.
------------------------------------------
Press enter to proceed. (Write STOP to exit.)

|HN| Attach attempt from DNA 3613856892

|HN| Checking IMSI..
|HN| Pseudonym is valid.

|HN| Creating pseudonym..
|HN| New Pseudonym is created.
|HN| New Pseudonym is 2890913730

|HN| R2-type AV is required.

|HN| Creating R2-type AV..
|HN| R2-type AV is created.

|HN| R2-type AV is sent to SN.

------------------------------------------

Press enter to proceed. (Write STOP to exit.)

|HN| Response result from SN.

|HN| Checking response..
|HN| Result for RES challenge is received from SN.

|HN| Checking result..
|HN| Authentication succeeded.

------------------------------------------

Press enter to proceed. (Write STOP to exit.)

|HN| Attach attempt from DNA 3613856892

|HN| Checking IMSI..
|HN| Pseudonym is valid.

|HN| R3-type AV is required.

|HN| Creating R3-type AV..
|HN| R3-type AV is created.

|HN| R3-type AV is sent to SN.
------------------------------------------
Press enter to proceed. (Write STOP to exit.)

|HN| Result for RES challenge is received from SN.

|HN| Checking result..
|HN| Authentication succeeded.

------------------------------------------
Press enter to proceed. (Write STOP to exit.)
stop
BUILD SUCCESSFUL (total time: 3 minutes 35 seconds)
```

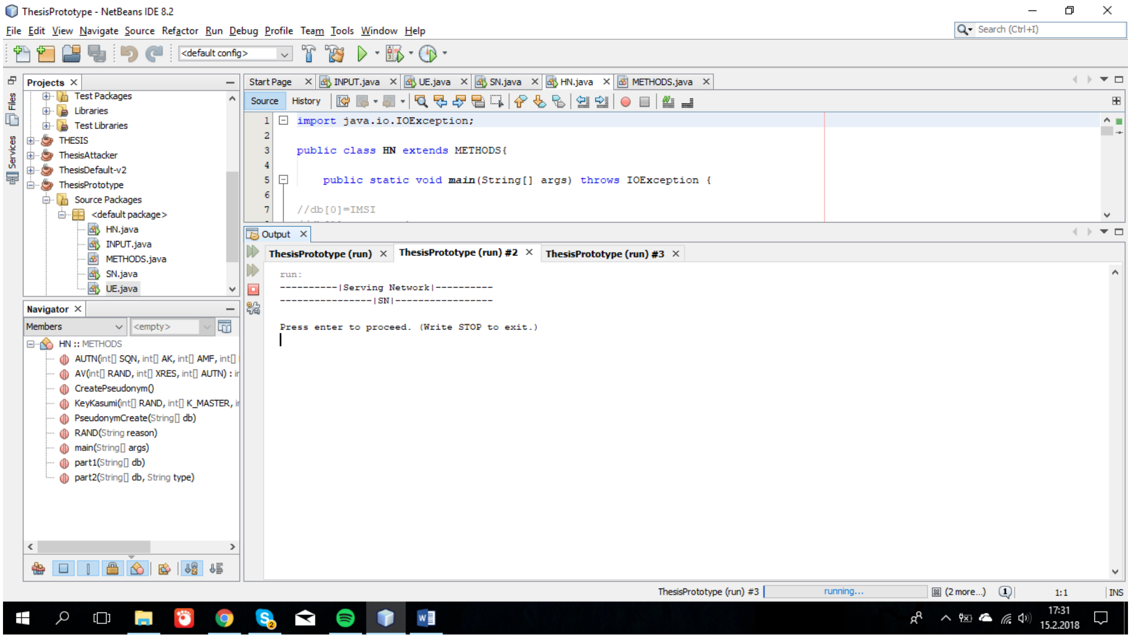
APPENDIX C – Screenshots

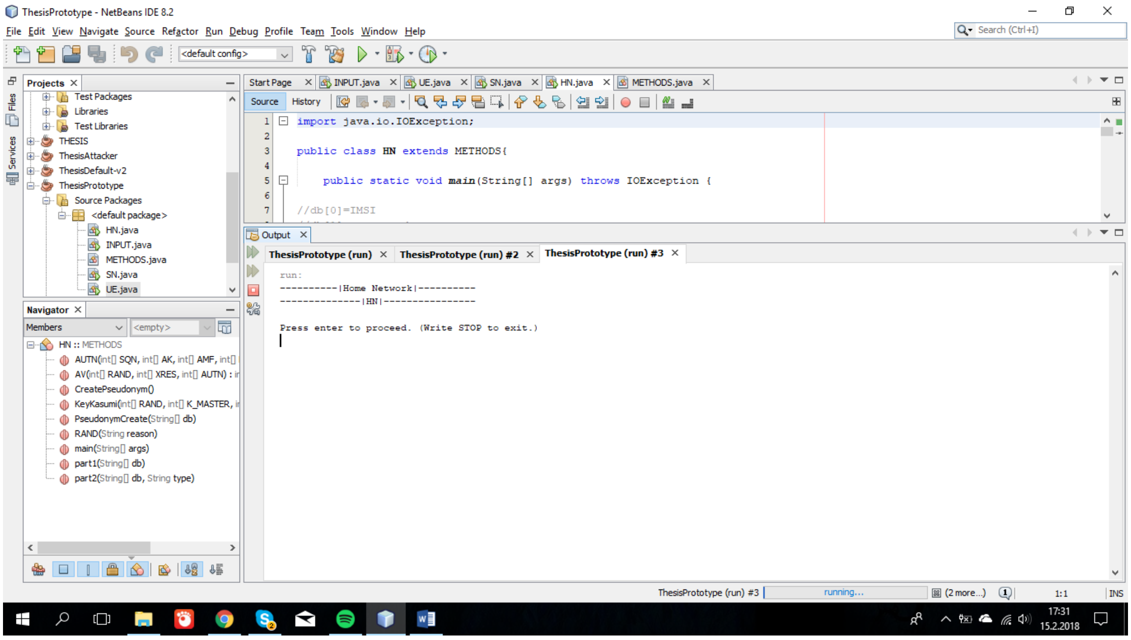
After running INPUT.java



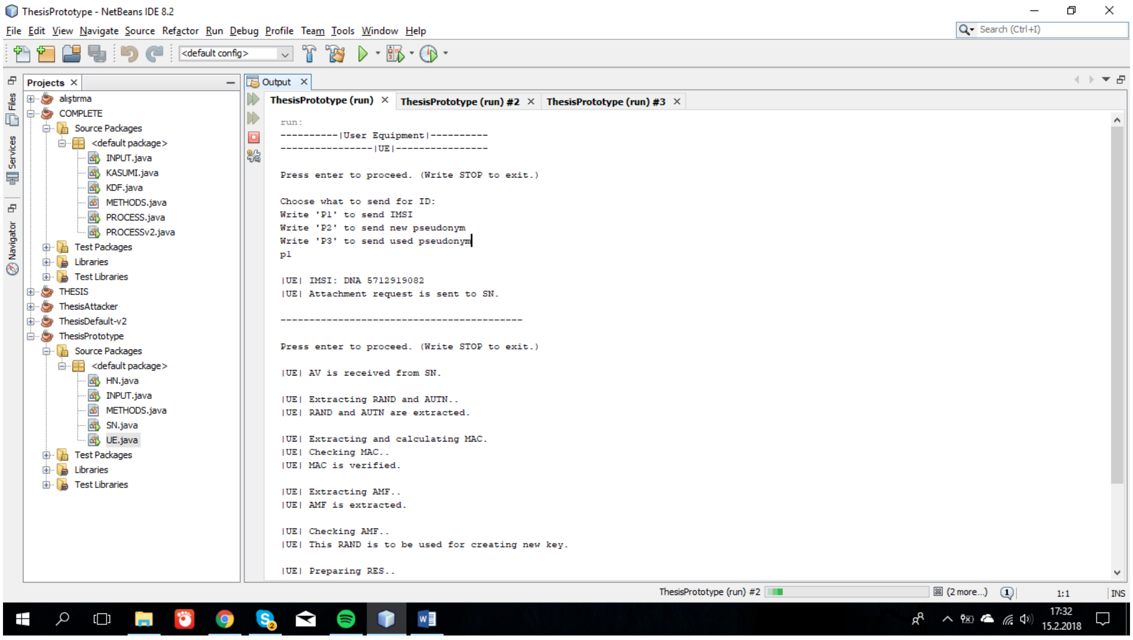
UE,SN,HN are run simultaneously



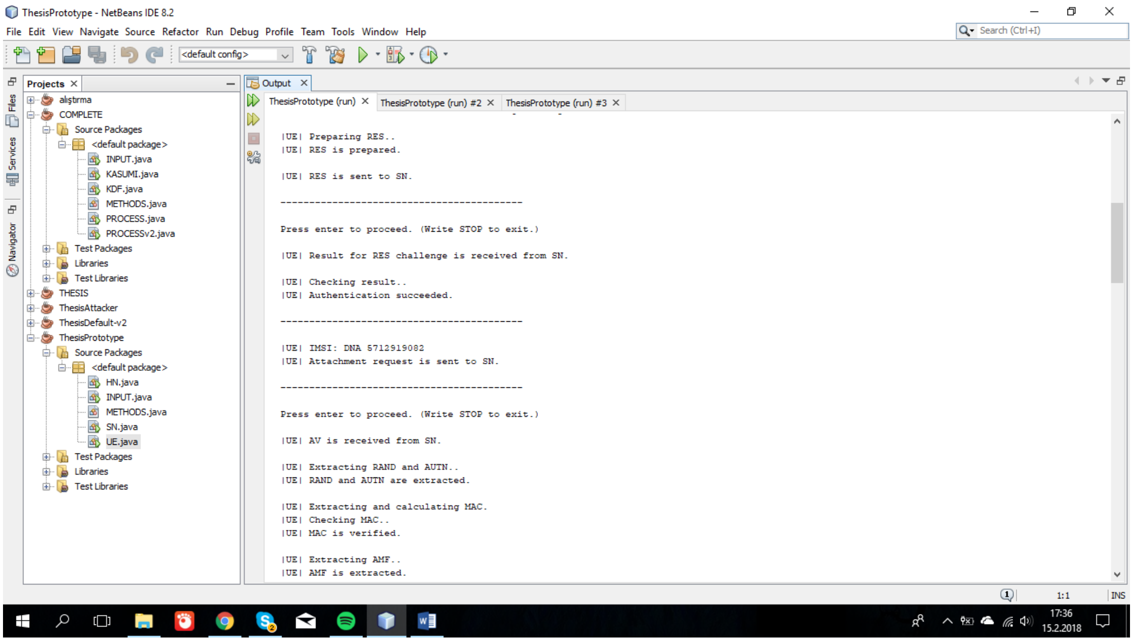

ThesisPrototype - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
Test Packages
Libraries
Test Libraries
THESIS
ThesisAttacker
ThesisDefault-v2
ThesisPrototype
Source Packages
<default package>
HN.java
INPUT.java
METHODS.java
SN.java
UE.java
Start Page
INPUT.java
UE.java
SN.java
HN.java
METHODS.java
Source
History
import java.io.IOException;
public class HN extends METHODS{
public static void main(String[] args) throws IOException {
//db[0]=IMSI
Output
ThesisPrototype (run)
ThesisPrototype (run) #2
ThesisPrototype (run) #3
run:
----------|Serving Network|----------
----------------|SN|-----------------
Press enter to proceed. (Write STOP to exit.)
Navigator
Members
HN :: METHODS
AUTN(int[] SQN, int[] AK, int[] AMF, int[]
AV(int[] RAND, int[] XRES, int[] AUTN) :
CreatePseudonym()
KeyKasumi(int[] RAND, int[] K_MASTER,
PseudonymCreate(String[] db)
RAND(String reason)
main(String[] args)
part1(String[] db)
part2(String[] db, String type)
ThesisPrototype (run) #3
running...
(2 more...)
1:1
INS
17:31
15.2.2018


ThesisPrototype - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
Test Packages
Libraries
Test Libraries
THESIS
ThesisAttacker
ThesisDefault-v2
ThesisPrototype
Source Packages
<default package>
HN.java
INPUT.java
METHODS.java
SN.java
UE.java
Start Page
INPUT.java
UE.java
SN.java
HN.java
METHODS.java
Source
History
import java.io.IOException;
public class HN extends METHODS{
public static void main(String[] args) throws IOException {
//db[0]=IMSI
Output
ThesisPrototype (run)
ThesisPrototype (run) #2
ThesisPrototype (run) #3
run:
----------|Home Network|----------
--------------|HN|----------------
Press enter to proceed. (Write STOP to exit.)
Navigator
Members
HN :: METHODS
AUTN(int[] SQN, int[] AK, int[] AMF, int[]
AV(int[] RAND, int[] XRES, int[] AUTN) :
CreatePseudonym()
KeyKasumi(int[] RAND, int[] K_MASTER,
PseudonymCreate(String[] db)
RAND(String reason)
main(String[] args)
part1(String[] db)
part2(String[] db, String type)
ThesisPrototype (run) #3
running...
(2 more...)
1:1
INS
17:31
15.2.2018

UE process:


ThesisPrototype - NetBeans IDE 8.2
Output
ThesisPrototype (run)
ThesisPrototype (run) #2
ThesisPrototype (run) #3
run:
----------|User Equipment|----------
----------------|UE|----------------
Press enter to proceed. (Write STOP to exit.)
Choose what to send for ID:
Write 'P1' to send IMSI
Write 'P2' to send new pseudonym
Write 'P3' to send used pseudonym
p1
|UE| IMSI: DNA 5712919082
|UE| Attachment request is sent to SN.

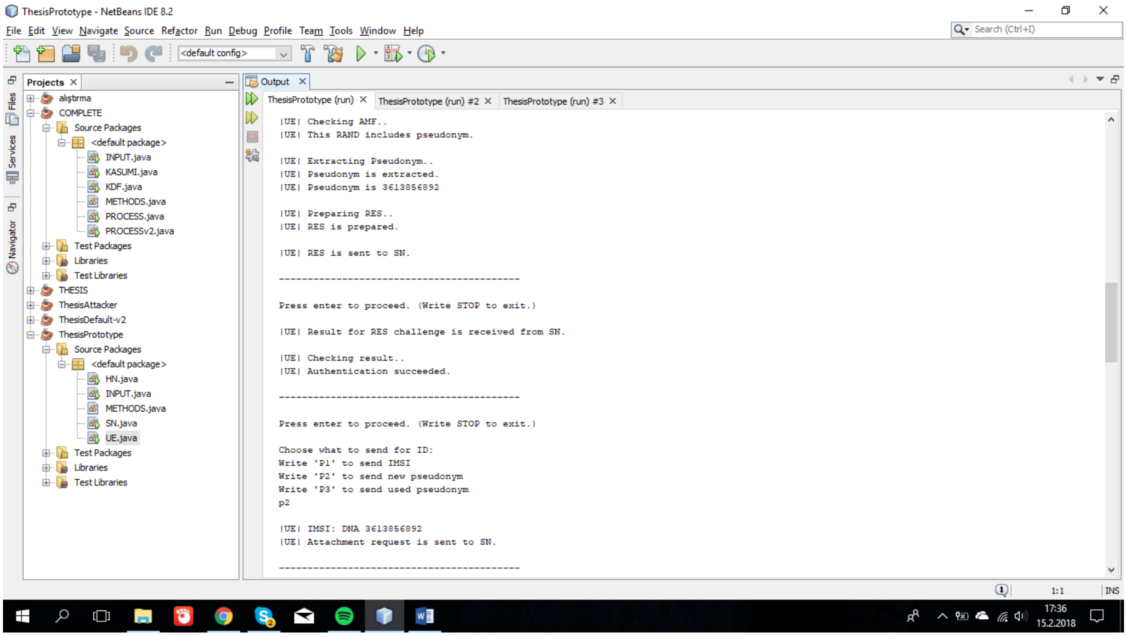
Press enter to proceed. (Write STOP to exit.)
|UE| AV is received from SN.
|UE| Extracting RAND and AUTN..
|UE| RAND and AUTN are extracted.
|UE| Extracting and calculating MAC.
|UE| Checking MAC..
|UE| MAC is verified.
|UE| Extracting AMF..
|UE| AMF is extracted.
|UE| Checking AMF..
|UE| This RAND is to be used for creating new key.
|UE| Preparing RES..


ThesisPrototype - NetBeans IDE 8.2
Output
ThesisPrototype (run)
ThesisPrototype (run) #2
ThesisPrototype (run) #3
|UE| Preparing RES..
|UE| RES is prepared.
|UE| RES is sent to SN.

Press enter to proceed. (Write STOP to exit.)
|UE| Result for RES challenge is received from SN.
|UE| Checking result..
|UE| Authentication succeeded.

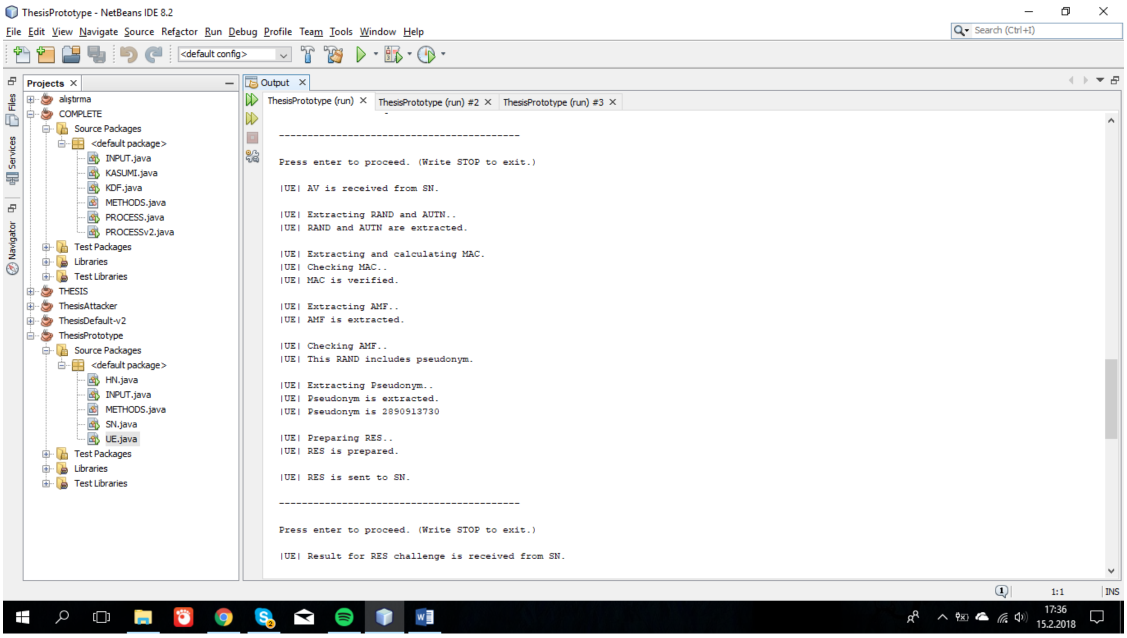
|UE| IMSI: DNA 5712919082
|UE| Attachment request is sent to SN.

Press enter to proceed. (Write STOP to exit.)
|UE| AV is received from SN.
|UE| Extracting RAND and AUTN..
|UE| RAND and AUTN are extracted.
|UE| Extracting and calculating MAC.
|UE| Checking MAC..
|UE| MAC is verified.
|UE| Extracting AMF..
|UE| AMF is extracted.


ThesisPrototype - NetBeans IDE 8.2
|UE| Checking AMF..
|UE| This RAND includes pseudonym.
|UE| Extracting Pseudonym..
|UE| Pseudonym is extracted.
|UE| Pseudonym is 3613856892
|UE| Preparing RES..
|UE| RES is prepared.
|UE| RES is sent to SN.

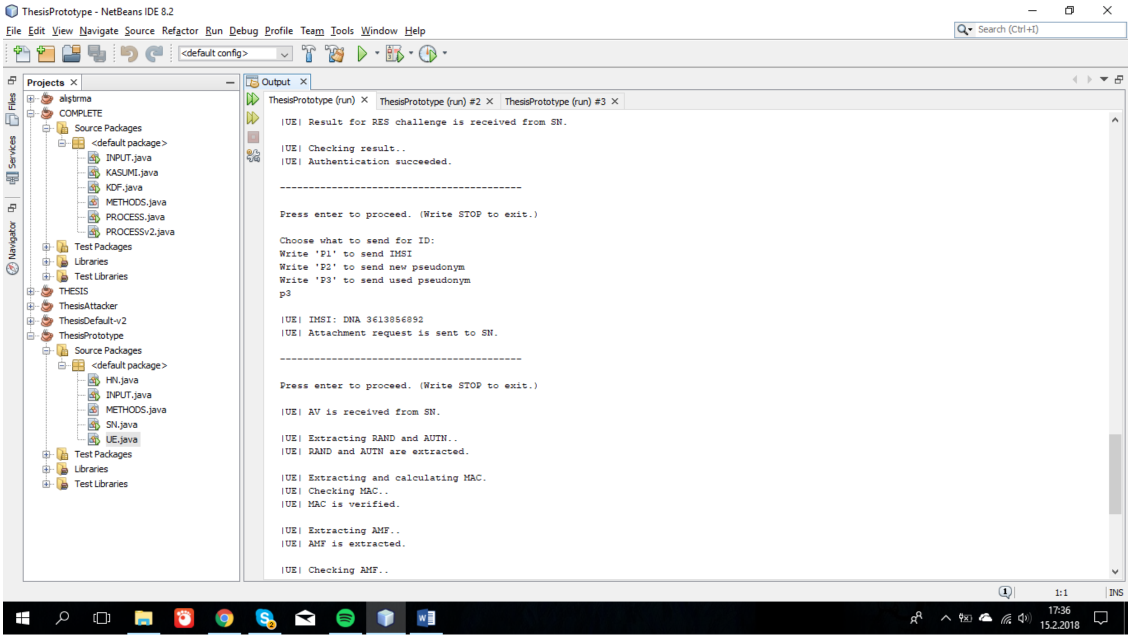
Press enter to proceed. (Write STOP to exit.)
|UE| Result for RES challenge is received from SN.
|UE| Checking result..
|UE| Authentication succeeded.

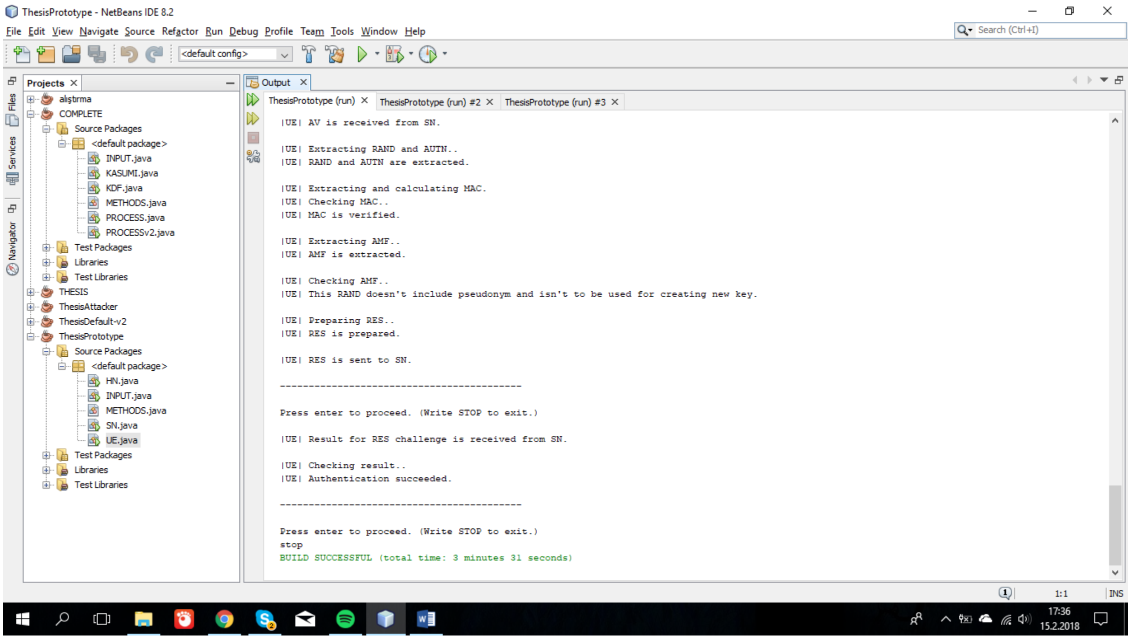
Press enter to proceed. (Write STOP to exit.)
Choose what to send for ID:
Write 'P1' to send IMSI
Write 'P2' to send new pseudonym
Write 'P3' to send used pseudonym
p2
|UE| IMSI: DNA 3613856892
|UE| Attachment request is sent to SN.


ThesisPrototype - NetBeans IDE 8.2

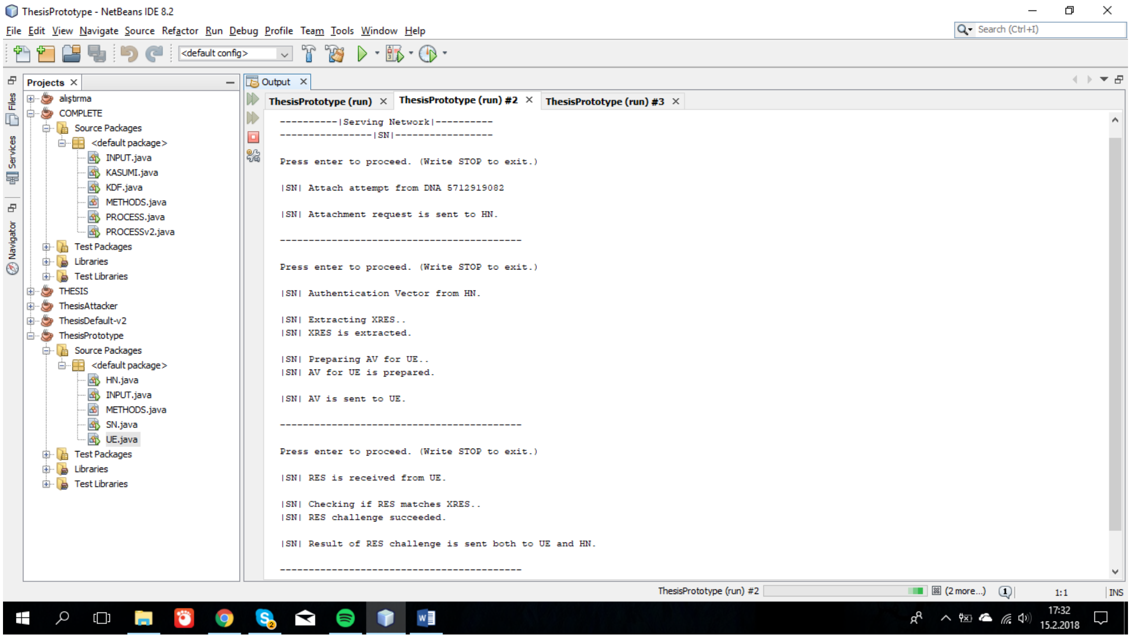
Press enter to proceed. (Write STOP to exit.)
|UE| AV is received from SN.
|UE| Extracting RAND and AUTN..
|UE| RAND and AUTN are extracted.
|UE| Extracting and calculating MAC.
|UE| Checking MAC..
|UE| MAC is verified.
|UE| Extracting AMF..
|UE| AMF is extracted.
|UE| Checking AMF..
|UE| This RAND includes pseudonym.
|UE| Extracting Pseudonym..
|UE| Pseudonym is extracted.
|UE| Pseudonym is 2890913730
|UE| Preparing RES..
|UE| RES is prepared.
|UE| RES is sent to SN.

Press enter to proceed. (Write STOP to exit.)
|UE| Result for RES challenge is received from SN.


ThesisPrototype - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Output
ThesisPrototype (run)
ThesisPrototype (run) #2
ThesisPrototype (run) #3
|UE| Result for RES challenge is received from SN.
|UE| Checking result..
|UE| Authentication succeeded.
Press enter to proceed. (Write STOP to exit.)
Choose what to send for ID:
Write 'P1' to send IMSI
Write 'P2' to send new pseudonym
Write 'P3' to send used pseudonym
p3
|UE| IMSI: DNA 3613856892
|UE| Attachment request is sent to SN.
Press enter to proceed. (Write STOP to exit.)
|UE| AV is received from SN.
|UE| Extracting RAND and AUTN..
|UE| RAND and AUTN are extracted.
|UE| Extracting and calculating MAC.
|UE| Checking MAC..
|UE| MAC is verified.
|UE| Extracting AMF..
|UE| AMF is extracted.
|UE| Checking AMF..

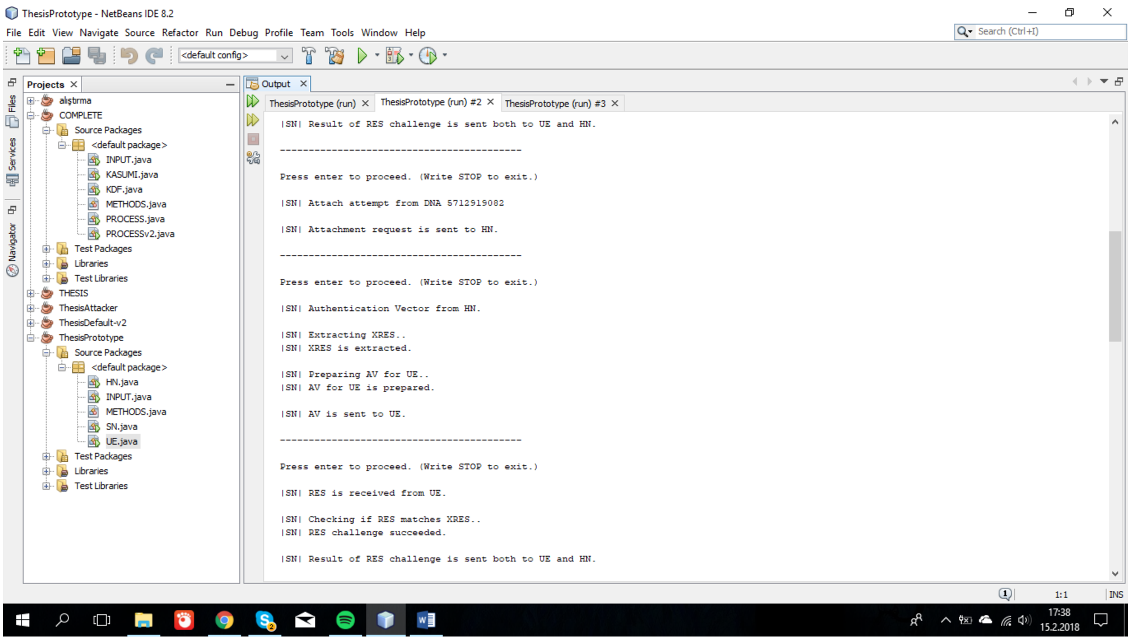

ThesisPrototype - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Output
ThesisPrototype (run)
ThesisPrototype (run) #2
ThesisPrototype (run) #3
|UE| AV is received from SN.
|UE| Extracting RAND and AUTN..
|UE| RAND and AUTN are extracted.
|UE| Extracting and calculating MAC.
|UE| Checking MAC..
|UE| MAC is verified.
|UE| Extracting AMF..
|UE| AMF is extracted.
|UE| Checking AMF..
|UE| This RAND doesn't include pseudonym and isn't to be used for creating new key.
|UE| Preparing RES..
|UE| RES is prepared.
|UE| RES is sent to SN.
Press enter to proceed. (Write STOP to exit.)
|UE| Result for RES challenge is received from SN.
|UE| Checking result..
|UE| Authentication succeeded.
Press enter to proceed. (Write STOP to exit.)
stop
BUILD SUCCESSFUL (total time: 3 minutes 31 seconds)

SN process:


TheisisPrototype - NetBeans IDE 8.2
----------|Serving Network|----------
----------------|SN|----------------
Press enter to proceed. (Write STOP to exit.)
|SN| Attach attempt from DNA 5712919082
|SN| Attachment request is sent to HN.

Press enter to proceed. (Write STOP to exit.)
|SN| Authentication Vector from HN.
|SN| Extracting XRES..
|SN| XRES is extracted.
|SN| Preparing AV for UE..
|SN| AV for UE is prepared.
|SN| AV is sent to UE.

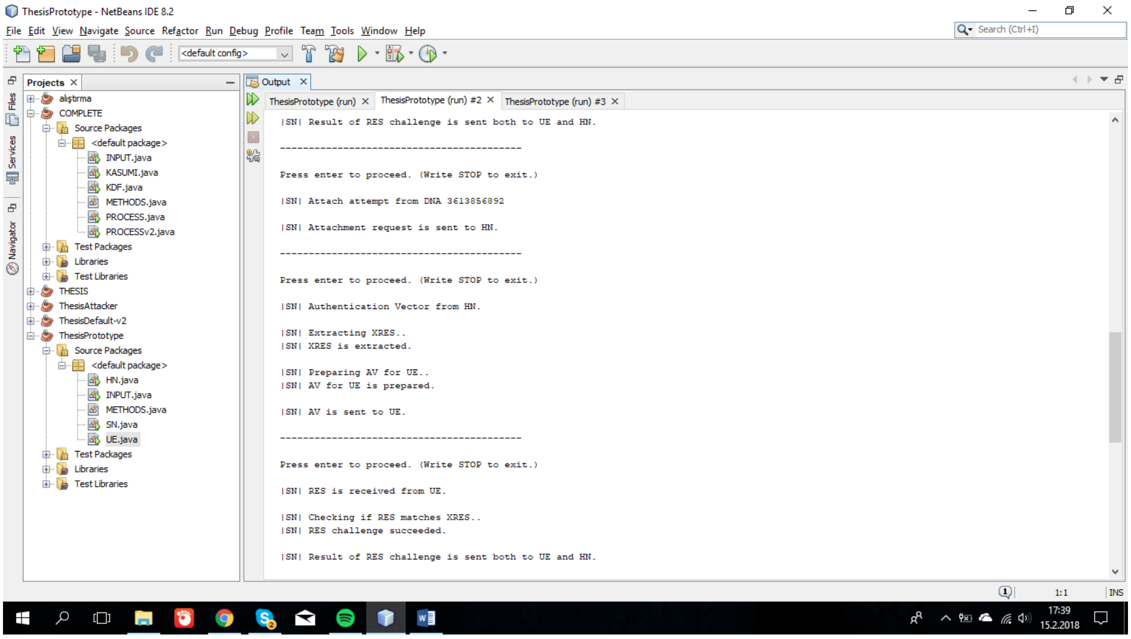
Press enter to proceed. (Write STOP to exit.)
|SN| RES is received from UE.
|SN| Checking if RES matches XRES..
|SN| RES challenge succeeded.
|SN| Result of RES challenge is sent both to UE and HN.


TheisisPrototype - NetBeans IDE 8.2
|SN| Result of RES challenge is sent both to UE and HN.

Press enter to proceed. (Write STOP to exit.)
|SN| Attach attempt from DNA 5712919082
|SN| Attachment request is sent to HN.

Press enter to proceed. (Write STOP to exit.)
|SN| Authentication Vector from HN.
|SN| Extracting XRES..
|SN| XRES is extracted.
|SN| Preparing AV for UE..
|SN| AV for UE is prepared.
|SN| AV is sent to UE.

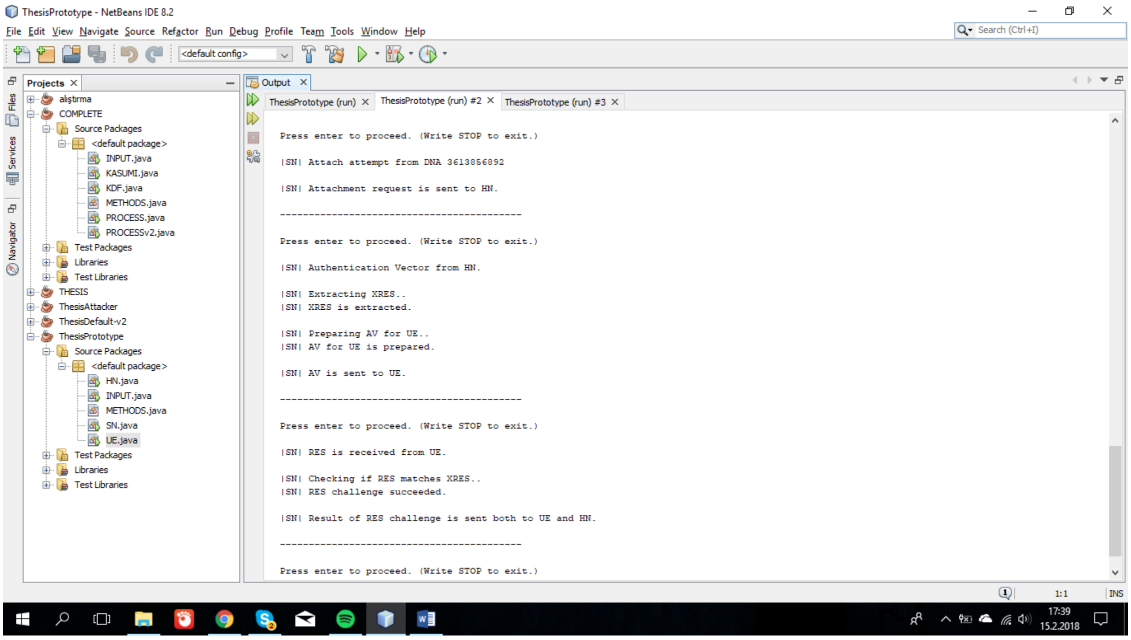
Press enter to proceed. (Write STOP to exit.)
|SN| RES is received from UE.
|SN| Checking if RES matches XRES..
|SN| RES challenge succeeded.
|SN| Result of RES challenge is sent both to UE and HN.


ThesisPrototype - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
Output
ThesisPrototype (run)
ThesisPrototype (run) #2
ThesisPrototype (run) #3
|SN| Result of RES challenge is sent both to UE and HN.

Press enter to proceed. (Write STOP to exit.)
|SN| Attach attempt from DNA 3613856892
|SN| Attachment request is sent to HN.

Press enter to proceed. (Write STOP to exit.)
|SN| Authentication Vector from HN.
|SN| Extracting XRES..
|SN| XRES is extracted.
|SN| Preparing AV for UE..
|SN| AV for UE is prepared.
|SN| AV is sent to UE.

Press enter to proceed. (Write STOP to exit.)
|SN| RES is received from UE.
|SN| Checking if RES matches XRES..
|SN| RES challenge succeeded.
|SN| Result of RES challenge is sent both to UE and HN.
17:39
15.2.2018

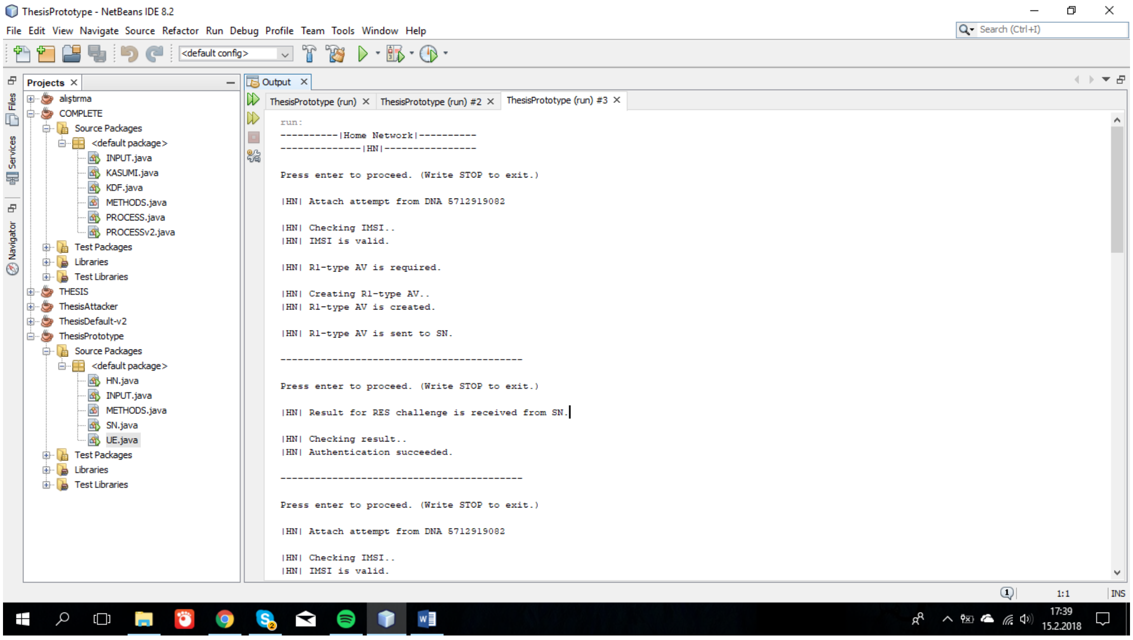

ThesisPrototype - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
Output
ThesisPrototype (run)
ThesisPrototype (run) #2
ThesisPrototype (run) #3
Press enter to proceed. (Write STOP to exit.)
|SN| Attach attempt from DNA 3613856892
|SN| Attachment request is sent to HN.

Press enter to proceed. (Write STOP to exit.)
|SN| Authentication Vector from HN.
|SN| Extracting XRES..
|SN| XRES is extracted.
|SN| Preparing AV for UE..
|SN| AV for UE is prepared.
|SN| AV is sent to UE.

Press enter to proceed. (Write STOP to exit.)
|SN| RES is received from UE.
|SN| Checking if RES matches XRES..
|SN| RES challenge succeeded.
|SN| Result of RES challenge is sent both to UE and HN.

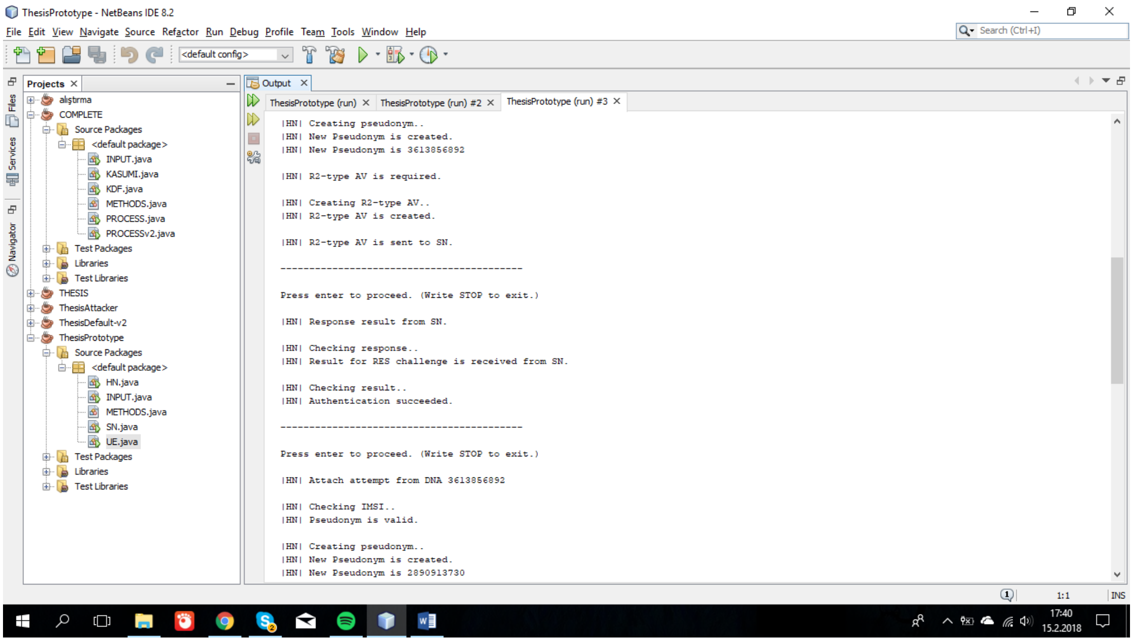
Press enter to proceed. (Write STOP to exit.)
17:39
15.2.2018

HN process:


ThesisPrototype - NetBeans IDE 8.2
Output
ThesisPrototype (run)
ThesisPrototype (run) #2
ThesisPrototype (run) #3
run:
----------|Home Network|----------
--------------|HN|----------------
Press enter to proceed. (Write STOP to exit.)
|HN| Attach attempt from DNA 5712919082
|HN| Checking IMSI..
|HN| IMSI is valid.
|HN| R1-type AV is required.
|HN| Creating R1-type AV..
|HN| R1-type AV is created.
|HN| R1-type AV is sent to SN.

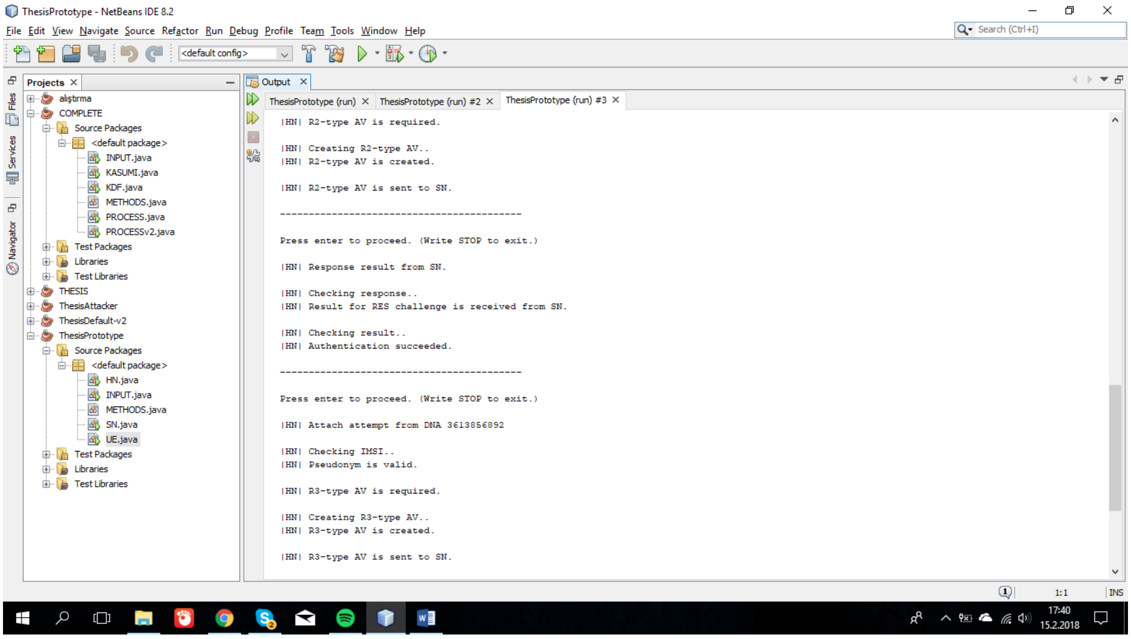
Press enter to proceed. (Write STOP to exit.)
|HN| Result for RES challenge is received from SN.
|HN| Checking result..
|HN| Authentication succeeded.

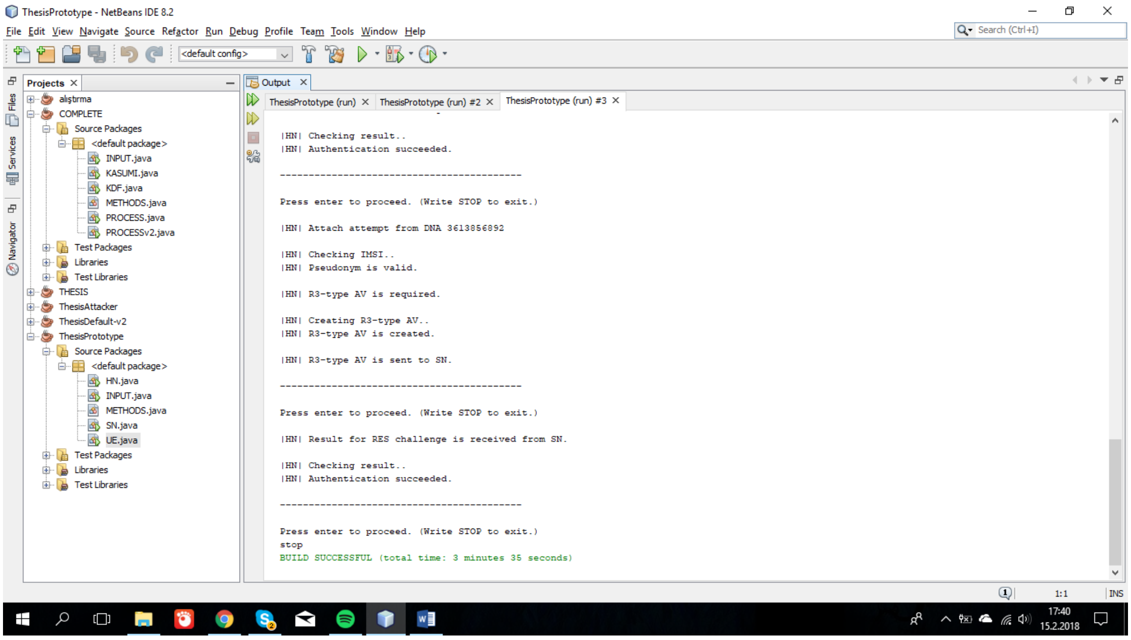
Press enter to proceed. (Write STOP to exit.)
|HN| Attach attempt from DNA 5712919082
|HN| Checking IMSI..
|HN| IMSI is valid.


ThesisPrototype - NetBeans IDE 8.2
Output
ThesisPrototype (run)
ThesisPrototype (run) #2
ThesisPrototype (run) #3
|HN| Creating pseudonym..
|HN| New Pseudonym is created.
|HN| New Pseudonym is 3613856892
|HN| R2-type AV is required.
|HN| Creating R2-type AV..
|HN| R2-type AV is created.
|HN| R2-type AV is sent to SN.

Press enter to proceed. (Write STOP to exit.)
|HN| Response result from SN.
|HN| Checking response..
|HN| Result for RES challenge is received from SN.
|HN| Checking result..
|HN| Authentication succeeded.

Press enter to proceed. (Write STOP to exit.)
|HN| Attach attempt from DNA 3613856892
|HN| Checking IMSI..
|HN| Pseudonym is valid.
|HN| Creating pseudonym..
|HN| New Pseudonym is created.
|HN| New Pseudonym is 2890913730


ThesisPrototype - NetBeans IDE 8.2
ThesisPrototype (run) #3
|HN| R2-type AV is required.
|HN| Creating R2-type AV..
|HN| R2-type AV is created.
|HN| R2-type AV is sent to SN.
Press enter to proceed. (Write STOP to exit.)
|HN| Response result from SN.
|HN| Checking response..
|HN| Result for RES challenge is received from SN.
|HN| Checking result..
|HN| Authentication succeeded.
Press enter to proceed. (Write STOP to exit.)
|HN| Attach attempt from DNA 3613856892
|HN| Checking IMSI..
|HN| Pseudonym is valid.
|HN| R3-type AV is required.
|HN| Creating R3-type AV..
|HN| R3-type AV is created.
|HN| R3-type AV is sent to SN.

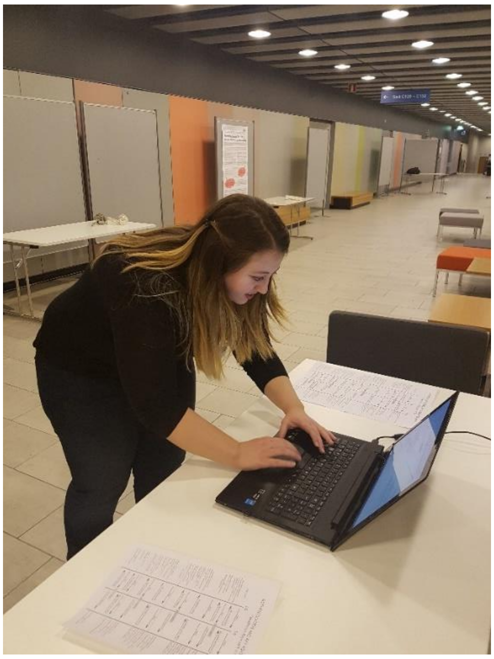

ThesisPrototype - NetBeans IDE 8.2
ThesisPrototype (run) #3
|HN| Checking result..
|HN| Authentication succeeded.
Press enter to proceed. (Write STOP to exit.)
|HN| Attach attempt from DNA 3613856892
|HN| Checking IMSI..
|HN| Pseudonym is valid.
|HN| R3-type AV is required.
|HN| Creating R3-type AV..
|HN| R3-type AV is created.
|HN| R3-type AV is sent to SN.
Press enter to proceed. (Write STOP to exit.)
|HN| Result for RES challenge is received from SN.
|HN| Checking result..
|HN| Authentication succeeded.
Press enter to proceed. (Write STOP to exit.)
stop
BUILD SUCCESSFUL (total time: 3 minutes 35 seconds)

APPENDIX D – Public demonstrations

1. The 21st Conference of Open Innovations Association FRUCT

   Helsinki, Finland

   6-10 November 2017



2. TAKE5 and 5G Test Network Finland workshop

   Espoo, Finland

   14 December 2017