FACULTY OF FUNDAMENTAL PROBLEMS OF TECHNOLOGY
WROCŁAW UNIVERSITY OF TECHNOLOGY

AUTHENTICATION AND COMMUNICATION PROTOCOL FOR MOBILE DEVICES IN SOUND CHANNEL

PAWEŁ KĘDZIA

In partial fulfillment of the requirements
for the degree of Master of Engineering

Thesis Supervisor
dr inz. Lukasz Krzywiecki

WROCLAW 2014

Contents

1 Introduction **3**
- 1.1 Problem . . . 4
- 1.2 Related work . . . 4

2 Analysis of the problem **5**
- 2.1 Authenticated Key Agreement Protocols . . . 5
  - 2.1.1 Security Model . . . 5
  - 2.1.2 Adversarial Model . . . 6
- 2.2 Anonymous Mutual Authentication Protocol . . . 7
- 2.3 Analysis of out-of-band channels . . . 10
- 2.4 Proposed usage of out-of-band channel . . . 11
- 2.5 Sound . . . 14
  - 2.5.1 Frequency . . . 14
  - 2.5.2 Sound intensity . . . 15
  - 2.5.3 Speed of sound . . . 15
  - 2.5.4 Ultrasounds . . . 15
  - 2.5.5 Fourier transform . . . 16
  - 2.5.6 Discrete Fourier Transform . . . 17
  - 2.5.7 Fast Fourier Transform . . . 19
  - 2.5.8 Hanning window . . . 19
  - 2.5.9 Pulse-Code Modulation . . . 20
- 2.6 Microphones and speakers in mobile devices . . . 21
- 2.7 Security analysis over the sound channel . . . 21
  - 2.7.1 No access to protocol transcript . . . 22
  - 2.7.2 Partial access to protocol transcript . . . 22
  - 2.7.3 Full access to protocol transcript . . . 22
  - 2.7.4 Possible vectors of attack on a sound channel . . . 23
  - 2.7.5 Speaker manipulating . . . 23
  - 2.7.6 Microphone manipulating . . . 23
  - 2.7.7 Eavesdropping . . . 23
  - 2.7.8 Malicious implementation . . . 24

3 Prototype Design **24**
- 3.1 Mobile Devices . . . 24
- 3.2 Operating System . . . 25
- 3.3 Security of Android . . . 25
- 3.4 Application architecture . . . 25
- 3.5 Cryptography library . . . 26
- 3.6 Design pattern . . . 26

3.7 NIST recommendation for security parameters 27
3.8 Shared cryptographic module 28
3.9 Java Native Interface - C++ Wrapper 29
3.10 Base64 29
3.11 Sound channel 30
3.11.1 Generating sound wave 30
3.11.2 Receiving sound 33

4 Tests **37**

5 Possible development of the project **39**

6 Summary **40**

1 Introduction

The goal of the dissertation is to analyze the usability of channels alternative to WiFi in Authenticated Key Establishment (AKE) protocols for mobile devices. Nowadays, mobile phones are more often used to send secret data. Each single device has embedded speaker and microphone therefore we want to check whether it is possible to establish session key using just sound waves.

Mobile devices use Internet mainly to transfer data. The disadvantage of this method is network accessibility. The alternative to transmitting data might be using other embedded components like IrDA, Bluetooth or NFC. Unfortunately, not every device has transmitting accessories implemented. However, speaker and microphone are embedded to handhelds as a standard equipment. Therefore this dissertation is focused on communication between devices by audio channel. The availability of audio appliance and usage variety of this solution are main advantages. Furthermore receiver and sender do not need to know each other (unlike Bluetooth) before starting the communication. Sender starts to transmit data and receiver at the same time listens to broadcast. Communication channels, such as WiFi, are using electromagnetic waves where sound channel is using mechanical waves. Therefore potentially adversary would have to use another type of devices, which would be able to eavesdropping protocol transcript, which is broadcasted by mechanical waves.

Anonymous Mutual Authentication (AMA) protocol was chosen for authentication and key agreement between mobile devices [15]. AMA is symmetric, which means participants execute the same operation but with a different order of operations. One of the advantage of this protocol is easy implementation on resource limited devices. It was originally designed for authenticating electronic documents, it is proved to be portable.

Application created on Android system serves as a proof of concept. There are applications which use sound channel to exchange data, but none of them use it to establish session keys. To use our application, we need a mobile phone/smartphone with Android software version 4.1.2 or higher.

The first section we focus on analysis of the problem. There are issues related to key exchange (KE) protocols. There is a description of security model on which these protocols are based on and an adversarial model. In the second part of this section we introduce the out-of-band channels. There is a description of communication channel which uses sound waves and its properties. At the end we analyse the security over the sound channel.

In the next chapter we describe design of prototype. There are shown, inter alia, the architecture of application, used libraries for cryptography module and sound channels. Then we describe algorithms of generating and receiving sound signals.

The next section describes results of application tests. At the end of the paper we describe possible future development and utilize of our application and the summary of our work.

1.1 Problem

The problem is secure exchange keys between two parties, which do not have knowledge of each other, in insecure, unreliable environment e.g. susceptible to eavesdropping.

The first known answer on this problem was published by Witfield Diffie and Martin Hellman. They have shown key exchange method which is called Diffie-Hellman. Its strength and immunity to eavesdropping is based on Discrete Logarithm Problem. Existing computers are not able to compute session key from eavesdropped protocol transcript in reasonable amount of time.

The method is safe from eavesdropping, however it does not authenticate both participants to each other and thus it does not guarantee security when adversary takes control over the channel in which the key establishment is being performed. Adversary by taking control over the channel performs Man-In-The-Middle-Attack (MITMA). MITMA means that, the adversary tries to make connection with participants of protocol without being detected. In this situation, communication might be manipulated by the adversary impersonating one of the participants, changing or sending his own messages, making them to believe that they are still talking to each other. The goal of these processes is to obtain secret key by attacker and thus decrypt and even modify all messages exchanged between parties.

To exchange data we mainly use electromagnetic radiation channels like Bluetooth, IrDA or Internet. We wanted to check other options to exchanging data between devices, which do not use electromagnetic radiation. We decided to study sound channel, because sound waves are mechanical waves which, in contrast to electromagnetic waves, require medium (e.g. water, air) to travel.

1.2 Related work

There are several ways to establish secure channel generating session key. There are solutions where shared secret password which prevents Man-In-The-Middle attacks is used as in [26, 25]. However, pre-shared passwords are not always practical in many scenarios. There are protocols such as KEA [21], CMQV [30], HMQV [19], Naxos, Naxos+ [13] and SIGMA [20] which are in the Authenticated Key Agreement family. The security requirements and security model definition in AKE protocols were proposed by Cenetti and Krawczyk in [31, 24]. To exchange data we used sound channel which is one of the considered out-of-band channels for key exchange protocols. This kind of channel is discussed in

[14, 18, 22, 28, 8, 12].

2 Analysis of the problem

In this section we focus on the problem of establishing secure connection in insecure channel between two parties. At first the parties need to authenticate each other and then create common session key, which is needed to symmetric encryption of the communication. We describe the adversarial model in which it is assumed that each party secretly holds a long-term private key that enables unique authentication of this party. There is also a trusted mechanism for binding identities with public keys. The private and public key with proper length are bounded in such way that obtaining private key from public key is difficult to accomplish. The communication channel might be completely controlled by an adversary who is able to intercept, delete, delay, modify or inject messages at will. There is a relation between the requirements of the security model and a well-defined attack scenario. In the second part of this section necessary issues to create a sound channel are described.

2.1 Authenticated Key Agreement Protocols

Authenticated key exchange (AKE) protocols allow two parties to establish a common session key which is used to symmetric encryption in further communication between parties. AKE also ensures authenticity of the parties. The protocols base on public key cryptography and trusted authorities which are the most common solutions being used. In public key cryptography (asymmetric cryptography) two separate keys are required, where the former one is private and the latter one is public. The keys are mathematically related to each other.

AKE protocols have to ensure that attacker cannot obtain any information about session key from the protocol transcript. They have to be resistant to exposed session transcript. Adversary, having this knowledge, should not be able to take control over the communication channel.

The design and analysis of secure AKE protocols is non-trival task. After many years of research, some important issues are still without satisfactory treatment. There are some proposed security models on the basis of create and analyses the key-exchange protocols. In below subsections we discuss about one of the security model on which security of AKE protocols underlies and adversarial model.

2.1.1 Security Model

We stress that there is no one ultimately defined security model for key exchange (KE). Security definitions can be different and it depends on the underlying mathematical methodology, the purposed application setting, etc. It is difficult to formalize correctly the notion

of "secure key-exchange protocol". However, we recourse to the requirements as the most fundamental for secure key exchange protocol which H. Krawczyk presents in [20].

- *Authentication* - each party participating in KE needs to uniquely verify identities of other authenticating party.
- *Consistency* - two parties participating in KE, after establishing of session key have to believe that common session key was established with party which participates in protocol. In other words, Alice has to believe that she established session key with Bob and Bob has to believe that he established session key with Alice.
- *Secrecy* - if two honest parties established session key then third party should not be able to obtain information about that session key. Third party by watching and interfering with the protocol run, should not be able to distinguish session key from a random key.

It is worth to notice that above requirements exist only with relation to well-defined attack model, which we describe in the next subsection.

2.1.2 Adversarial Model

In the adversarial model the attacker is able to listen the all transmitted information, delay, delete, change or inject messages at will. The attacker can also overtaking control the scheduling of all protocol events, such as initiation of protocols or message delivery. Additionally he is able to obtain the long term secrete key from the party. Having this key adversary is able to control this party completely and the party is considered as corrupted. This leads to the fundamental property of key exchange protocols:

- *Real-or-Random* - this is some kind of game which basic session key security is based on. When uncorrupted parties with the successful protocol end then it is determined bit b at random. If $b = 0$ the adversary gets the real session key, if $b = 1$ he receives a random value with the same length as the key. Now adversary has to output his own bit b'. If $b' = b$ then adversary wins the game. When the adversary wins the game with probability $P = 1/2 + \varepsilon$, where ε is negligible function then the basis session key security is achieved.
- *Perfect Forward Secrecy (PFS)* - it is the property which ensures that compromise of long-term keys does not compromise past session keys. This mean that if adversary obtain some secrete informations (e.g. long-term key) of corrupted party then he nothing learns about previous sessions where the party participated.

Foregoing criteria ensure key exchange protocol to be resistant to attacks such as known-key attacks and replay attacks (see [10]). For some KE protocols there are additional requirements needed e.g. Identity protection - identities of authenticating parties cannot be learn by an adversary.

2.2 Anonymous Mutual Authentication Protocol

To establish session keys we used in our prototype Anonymous Mutual Authentication (AMA) protocol which is based on Diffie-Hellman key exchange. AMA is symmetric which means that participants do the same code but with different order of operation. AMA originally was designed to anonymous authentication between smart cards, so it is easy to implement it on resource limited devices. The protocol scheme is shown in Fig. 1 [15]. In the scheme there are used the following notations:

- *Enc* is a symmetric encryption function. $Enc_K(M)$ means encryption of M using key K.
- H is a cryptographic hash function.
- For confirming public keys digital certificates and public key infrastructure (PKI) are used.

Alice		Bob
x_A - private key $y_A = g^{x_A}$ - public key $cert_A$ - certificate for y_A		x_B - private key $y_B = g^{x_B}$ - public key $cert_B$ - certificate for y_B
	MAIN PROCEDURE	
choose a at random $h_A := H(a)$ $c_A := g^{h_A}$	$\xrightarrow{c_A}$ $\xleftarrow{c_B}$	choose b at random $h_B := H(b)$ $c_B := g^{h_B}$
$K := c_B^{h_A}$ $K_A := H(K,1), K_B := H(K,2)$ $K'_A := H(K,3), K'_B := H(K,4)$ $r_A := H(c_B^{x_A}, K'_A)$		$K := c_A^{h_B}$ $K_A := H(K,1), K_B := H(K,2)$ $K'_A := H(K,3), K'_B := H(K,4)$
	$\xrightarrow{Enc_{K_A}(cert_A, r_A)}$	check $cert_A$, proceed with random values if $r_A \neq H(y_A^{h_B}, K'_A)$
	$\xleftarrow{Enc_{K_B}(cert_B, r_B)}$	$r_B := H(c_A^{x_B}, K'_B)$
check $cert_B$, proceed with random values if $r_B \neq H(y_B^{h_A}, K'_B)$		
$K_{session} := H(K,5)$		$K_{session} := H(K,5)$

Figure 1: Protocol description - Anonymous Mutual Authentication.

At the beginning, parties (Alice and Bob) generate their private key - x_A (respectively, x_B) and public key $y_A = g^{x_A}$ (respectively, y_B). Then parties generate their ephemeral keys. Private is $h_A := H(a)$ (where a is a random number) (respectively h_B, b) and public ephemeral key $c_A := g^{h_A}$ (respectively, c_B). Then participants exchange their ephemeral keys, based on

Diffie-Hellman key exchange. Note that in this phase parties do not exchange any identity information.

When parties obtain ephemeral public key, they start compute master key K. Four different one-time keys are established by hashing K and numeric parameter - different for each one-time key.

To authentication, parties encrypt their certificates $cert_A$ and $cert_B$ and the challenge values r_A and r_B which are the hash of two keys. Authenticated party is raising ephemeral public key c_B (respectively, c_A) to power of private key x_A (respectively, x_B). Verifier compute the same value without private key but with the discrete logarithm of c_A (respectively, c_B) which is y^{h_B} (respectively, y^{h_A}). If the compared values are equal then $K_{session}$ is the hash of master key K and new number parameter.

This protocol is proved to be secure under Random Oracle Model (ROM) for the used hash function H. In ROM, hash function H is modelled as an oracle O_H. For a query x, oracle responds y if in internal table T exist entry (x, y). If there is no such an entry for x, then O_H selects y at random, creates entry (x, y) is in T and answers y. In this model the only available operation concerning hash function H is asking for values for concrete arguments. Hence, having value y the only way to get $x = H^{-1}(y)$ is to remember it from a previous query or ask a query with new x and get y by accident.

Besides ROM, protocol's security bases on Computational Diffie-Hellman (CDH) assumption which is hard to solve.

Assumption 1. ***(CDH Assumption)*** *Given a cyclic group G with order q, and (g,g^a,g^b) where g is a random generator of G, and (a,b) were chosen randomly from the set 0,1,...,$q-1$ it is computationally intractable to compute the value g^{ab}.*

In some proofs of protocol security is used Decisional Diffie-Hellman assumption, which is stronger than CDH.

Assumption 2. ***(DDH Assumption)*** *Given a cyclic group G with order q, and two triples (g^a,g^b,g^{ab}), (g^a,g^b,g^c) where g is a random generator of G, and (a,b,c) were chosen randomly from the set 0,1,...,$q-1$ it is computationally intractable to distinguish the value g^{ab} from g^c.*

The protocol is characterized by properties:

- *Undetectability* - having protocol transcript one cannot confirm that the process really took place;
- *Simultability* - this property results from the above property. It is possible to create extended protocol transcript (protocol transcript appended with ephemeral a and b)

between parties without interacting with them, but with the same probability distribution as in case of real interactions. The proof consists of that it shows that third party Eve is able to prepare a fake protocol. She chooses a and b at random and she computes c_A, c_B, K, K_A, K'_A, K_B, K'_B. Challenge values are computed in such a way that: $r_A := H(y_A^{h_B}, K'_A)$, $r_B := H(y_B^{h_A}, K'_B)$, so the values are the same as in the real transcript. Then she prepares the encryptions according to the protocol. Eve using these values to creates fake protocol transcript;

- *Deniability* - in ROM model, no one of the participants can convince third party that he participate in protocol. It would be possible only if convincing party would resolve CDH which is hard (see more detail in [15]);

- *Privacy* - eavesdropping adversary cannot tell about identity of parties performing the protocol. It is proven by series of games in [15] for two cases. In the first case adversary is observing a single transmission. The adversary is not able to learn something about the parties if the following advantage is negligible: Let $T = (m_1, ..., m_2)$ be a tuple of all messages exchanged between A and B in an protocol execution which ended successfully. R denotes a tuple $(r_1, ..., r_2)$ of independent random values. The advantage of an adversary $\mathcal{A}^{bind}$, $\mathbf{Adv}(\mathcal{A}^{bind})$, is at least ε, if there is strategy for adversary such that difference between the probability that $\mathcal{A}^{bind}$ returns "0" for given (PK_A, PK_B, T) and the probability that $\mathcal{A}^{bind}$ returns "0" for given (PK_A, PK_B, R) is at least ε.

  Theorem 1. *For AMA, $\boldsymbol{Adv}(\mathcal{A}^{bind})$ is negligibly small, assuming hardness of DDH Problem and ROM model for H.*

  The second case considers that the adversary has access to some interactions between Alice and Bob (allegedly). For more details see [15].

- *Forward Security* - if long term keys of one of the authenticating parties is compromised then established session key does not reveal.

Foregoing properties are described and proved in [15] using mentioned assumptions and series of games, additionally there are security proofs of symmetric encryption for which in [15] following lemmas was described:

Lemat 2.1. *Assume that scheme E is deterministic, DDH is hard and a collision for H may be found with a negligible probability only. If B receives E_1 which equals $E_{K_A}(cert_A, r_A)$ as computed by B, then with overwhelming probability x_A has been used for computing E_1.*

Lemat 2.2. *In the ROM model for H, if B receives E_1 which equals $E_{K_A}(cert_A, r_A)$ as it would be computed by B, then with overwhelming probability A has received c_B not manipulated by the adversary.*

Lemat 2.3. *Assume ROM model for H and that Computational Diffie-Hellman Problem is hard. If adversary $\mathcal{A}$ sends E_1 which equals $E_{K_A}(cert_A, r_A)$ as it would be computed by B, then with overwhelming probability A participates in the interaction and c_A sent by A is delivered to B.*

The practical part of this paper was to implement AMA on mobile devices as to our best knowledge there is no practical implementation of AMA for smartphones.

2.3 Analysis of out-of-band channels

In this disseration, beyond focusing on safety of AMA protocol against active adversary, we want to analyse unusual communication channels, which prevent the adversary to overtaking control this channel.

During the secure exchange data two channels are needed for communication between two parties. First channel, to establish session keys, should be difficult to eavesdrop and taking over control for adversary. The second channel can be insecure because established session key should ensure that content of protected data intended for one of the parties can be encrypted only by the party participated in protocol.

There are many techniques to transfer secret data between devices. The most popular are e.g. Bluetooth, IrDA or WiFi. These techniques use electromagnetic waves. Devices which operates on electromagnetic radiation (e.g. bug in the phone) are needed for manipulating, changing or eavesdropping messages. So it is worth to consider out-of-band channel which is not so commonly used as electromagnetic devices.

Safety channel could be, for example, two devices connected by wire. It would be very difficult to eavesdrop the session for the third party. Attacker should be connected also to this wire but it would be visible for the other parties and easily to detect. Unfortunately, carrying additional cable every time when the parties want to establish session key is inconvenient.

The other possibility could be optical channel [23]. Sharing data using e.g. QR codes is also an interesting way to authenticate devices. Attacker would have a difficult task to manipulate shared data if e.g. two devices are covered by some walls. However, not every device has an appropriate quality camera that can read QR codes with appropriate big public keys.

Another way is to use vibrations [8]. New smartphones have installed sensor of vibration so it is possible to create such a communication channel. Attacker to manipulate this kind of channel, has to generate his own vibrations and it would be easily detected by authenticating parties.

Sound channel is also one of the options. Every mobile device has a speaker and a microphone, so we do not need to worry about unavailable Internet or that the other party has not installed Bluetooth or IrDA or even a camera. Using ultrasounds, the signal is not detected by humans ear. Additionally mobile phones cannot generate these waves with big intensity, so transferring data should be done within a small distance. Sound waves are mechanical waves which are characterized by different properties from electromagnetic waves (WiFi, Bluetooth etc.). Hence, adversary needs to have additional devices to be able to manipulate or eavesdrop sound channel. It is worth to consider the usage of kind of channel mixed with another channel. For example, ephemeral keys are sent by sound channel then encrypted values by Bluetooth or one party sends data by sound waves when the second party uses just electromagnetic channel. Then adversary would have hinder task to manipulate mechanical and electromagnetic channel at the same time.

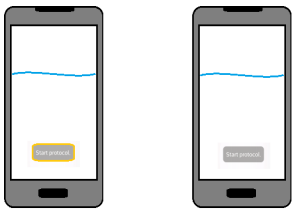
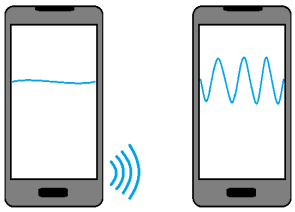
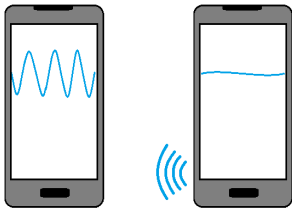
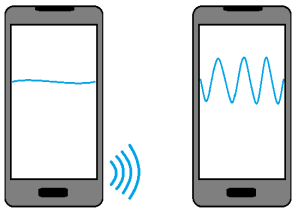
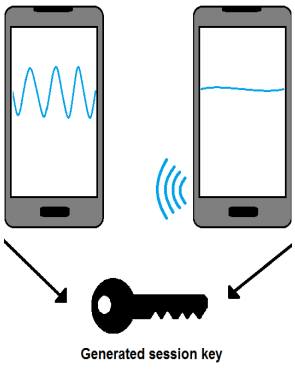
2.4 Proposed usage of out-of-band channel

Lets assumed that Alice and Bob want to establish a secure connection between their mobile phones using sound channel. To be sure that every high frequency will be detected, they try to put their phone in such a way a that microphone is as close as possible to a speaker of the second phone and vice versa. Usually the speaker and the microphone are set at the bottom of device, so mobile phones should be directed bottom to each other. To obtain session key is processed AKE protocol. After this process further exchange of data can be done by WiFi, because of the speed of transferring data which is higher.

- Alice and Bob put their devices into the lead box. Both mobile phone are listening for the first sign. The first one which starts generating sound wave is the initiator. For the best quality of transferring data mobiles should be directed bottom to each other no more than couple of millimetres.
- Alice decided to be the initiator of the protocol. She says "Start" to the little hole in the box and her phone, after recognition of her voice, first data is sent.
- Protocol is divided on couple of phases. First phase sends ephemeral key by initiator (in our case by Alice) to the receiver. Each phase takes couple of seconds. Every data is encoded in sound waves with appropriate frequencies.
- Bob's phone uses its microphone to obtain the data sent by Alice's device.
- When phase is over, the roles are reversed. Now Alice's phone listens and Bob's phone sends data.
- Alice's device receives and decodes data.

Anonymous Mutual Chip Authentication (described in detail in foregoing section) should ensure that even if some transferred data eavesdrop then adversary is not able to do anything with this data. Duration of the protocol is about 35 second - more information about duration and performance are described in the fourth section.

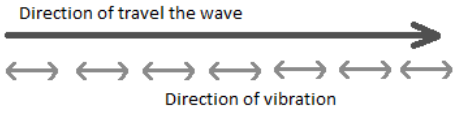
Table 1: Scheme of protocol used in devices.

Alice		Bob
Starting screen Alice starts the protocol as initializator		**Starting screen** Bob starts protocol as receiver.
Send ephemeral key Alice generates ephemeral key, encodes and start sending it by speaker.		**Receive ephemeral key** Bob receives ephemeral key using microphone.
Receive ephemeral key Alice receives ephemeral key using microphone.		**Send ephemeral key** Bob generates ephemeral key, encodes and start sending it by speaker.
Send encrypted authentication data Alice uses her and Bob's ephemeral keys to compute authentication data, encrypts it and sends by speaker.		**Receive and decrypt authentication data** Bob receives and decrypts Alices authentication data.
Receive and decrypt authentication data Alice receives and decrypts Bobs authentication data. If it's not valid, she aborts the protocol.		**Send encrypted authentication data** Bob verifies authentication data received from Alice. If it's valid, he generates his authentication data and send it by microphone. If data is not valid, Bob aborts the protocol.

2.5 Sound

Sound and its properties have one of the important role in our dissertation. All of the data are sent by using sound waves, so in this section necessary issues associated with it are explained.

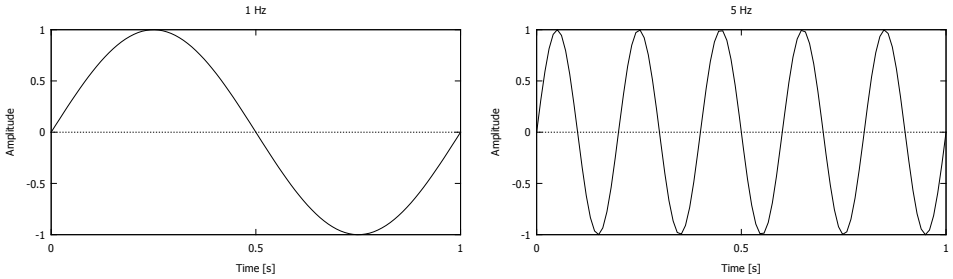
Sound is an acoustic wave propagating in various substances such as water, air (so called vibrating wire). There are waves which cause auditory sensation and at the same time, in appropriate amplitude and frequency, are not detected by human ear. Sound to spread, needs to have some medium, that is why it is not propagate in vacuum. Furthermore, sound is longitudinal waves, which means that particles of the medium is in the same direction as the travel direction of the wave.



2.5.1 Frequency

In our application, in sound channel part, differentiating of frequencies is a primary task. Data are sent by various frequencies which later are analysed and changed to adequate binary data. This section describes what the frequency of sound is.

Audio frequency is measured in hertz (Hz), where 1 Hz means one cycle per second. Below figures show graphs of audio frequency 1 Hz and 5 Hz.



There are four divisions of sound as to frequency:

- *Infrasound* - frequency is lower than 16 Hz. Too low frequency to be audible for human. Infrasounds have very large length of wave, more than 17 metres. They have influence on humans comfort. On high level we can feel pressure in ears, discomfort, excessive fatigue or somnolence;

- *Hearable sound* - frequency is greater than 16 Hz and lower than 20 kHz;
- *Ultrasound* - frequency is greater than 20 kHz. Too high to be audible for human. Ultrasounds are used inter alia in medicine (USG) or in sonars;
- *Hipersound* - frequency is greater than 10 GHz. They can spread only in crystals, because of such a high frequency. The wave is smaller than distance between molecules in the environment. They disappear in the gases.

Frequency specify also pitch of sound. A high frequency sound wave corresponds to a high pitch of sound and low frequency sound wave corresponds to a low pitch of sound.

2.5.2 Sound intensity

Loudness of the sound depends to its intensity. It is expressed by W/m^2 unit. This is ratio the power of the acoustic wave to surface area to which wave energy falls perpendicular during 1 second [4]. Human's ear works in such way that doubling intensity of sound, does not cause that human hear it two times louder. Human hears its logarithmized value. Logarithmized measure of sound intensity is called sound intensity level. It is measured in decibels and is given by [6]:

$$L = 10\log_{10}\left(\frac{I}{I_o}\right)[dB], \tag{1}$$

where:

- L - Sound intensity level,
- I - Sound intensity,
- I_o - Contractual value of the intensity which determines hearing range, which is $10^{-12}\frac{W}{m^2}$.

2.5.3 Speed of sound

Speed of sound depends on centre through which wave passes. Influence on speed has also temperature of the centre. It moves the most slowly in the flue. In the air with 20^o temperature reach in approximation 343 m/s (1230 km/h). In water with the same temperature speed equals 1482 m/s (5335 km/h). Sound spreads the fastest in solid. For example in steel it reaches about 5960 m/s (21460 km/h) [7].

2.5.4 Ultrasounds

The present application, uses largely ultrasound to send data. Ultrasounds are sound waves, which are inaudible for human, on the grounds of their too high frequency. Ultrasounds reception starts from 20 kHz, but it may differ depending on hearing ability. Children hear

the highest frequencies. Increasing age the upper limit of hearing descend [27]. Ultrasounds end at frequencies of several GHz, where hipersounds start. Contractual it is 10 GHz. Ultrasounds have many applications. Using them one can detect objects and specify distance to them. They are used in medicine for ultrasonography (USG) we can find unavailable for human eye anomalies, watching e.g. internal organs.

2.5.5 Fourier transform

In our prototype, microphone receives signal as a sinus wave. Because of that we need to know the frequency of this signal, we have to use some transform which turn sinus wave into the frequency spectrum. The solution is to use *Fourier transform*.

Using *Fourier transform* one can change function $f(x)$ in time domain to $F(s)$ in frequency domain. In other words, it distributes the function $f(x)$ to series of periodic functions, in such way that it shows how frequencies consist of the function $f(x)$. The formula for the *Fourier transform* looks as follows:

$$F(s) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi xs} dx. \tag{2}$$

Sum of three functions with frequencies 3 Hz, 7 Hz and 11 Hz on a graph:

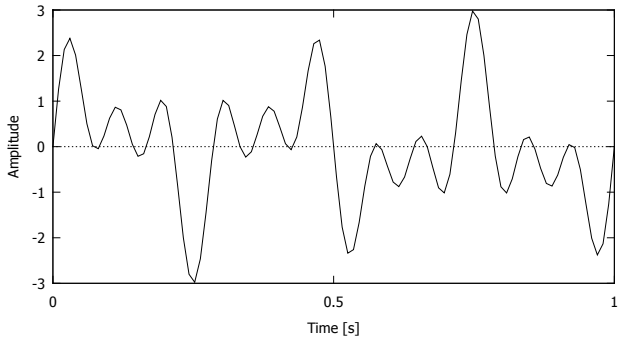


Figure 2: $f(x) = \sin(2\pi \cdot 3t) + \sin(2\pi \cdot 7t) + \sin(2\pi \cdot 11t)$, where t is current time.

Passing function $f(x)$ from fig. 2 through *Fourier transform* we will get the following graphs:

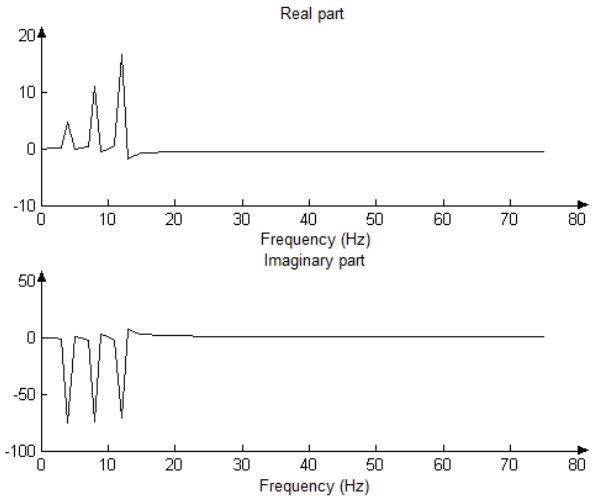


Figure 3: On the top real part results of *Fourier transform*, on the bottom imaginary part.

As you can see on the above figure, time domain was changed on frequency domain. On graphs there are bigger values for frequencies which had functions summed.

2.5.6 Discrete Fourier Transform

Application analyses periodically finite number of samples of sound. To converse to frequency domain *discrete Fourier transform* (*DFT*) is used. Inserted samples are in the form of complex numbers, but in practice we use only real numbers. Output samples are complex numbers. The following graphs show continuous signal and discretized signal [1]:

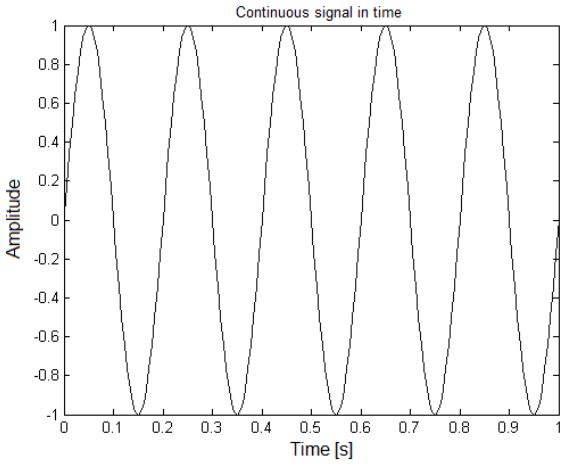


Figure 4: Continuous signal in duration 1 second.

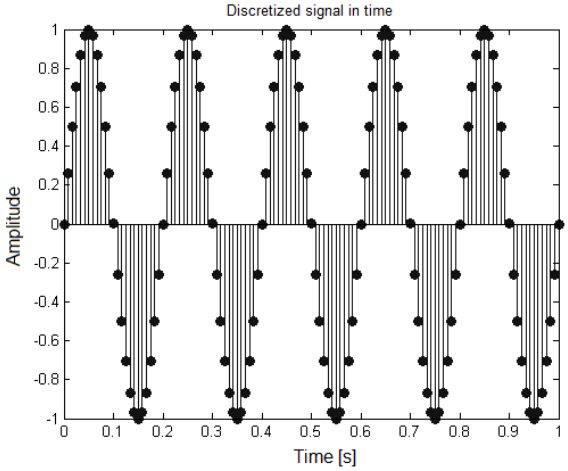


Figure 5: Discretized signal in constant time period. In this case in 1 second are 121 samples.

For N signal samples $(x_0, x_1, x_2, ..., x_{N-1})$, *discrete Fourier transform* gives series X_k for $k = 0, 1, 2, 3, ..., N-1$ using notation [1]:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi \frac{k}{N} n}. \tag{3}$$

Computational complexity of this notation is $O(N^2)$, because for N elements X_k series compute sum for N samples [17].

2.5.7 Fast Fourier Transform

Computational complexity of *discrete Fourier transform* is not enough for fast processing of more data. Therefore a lot of algorithms were developed, which improve its calculation. There are algorithms, which do *discrete Fourier transform* having computational complexity $O(NlogN)$, where for large number of data it is a significant improvement. They use design paradigm divide and conquer. Precursors of these solutions are James William Cooley and John Tukey, who elaborated first faster algorithm to compute *DFT*, known as *fast Fourier transform* (*FFT*). It is worth to mention that *FFT* is not approximate value of *DFT*. *FFT* returns the same values but performs it much faster.

2.5.8 Hanning window

Window functions are functions which are used with *DFT*. Often during process of *DFT* occur blur of spectrum, which means that on the spectrum values for frequencies different than the real frequency of signal are seen. To minimize this effect window functions are used. In time domain it multiplies every sample of signal by corresponding value of window sample. The simplest window is a rectangular window. Spectrum of this windowing is shown in 6

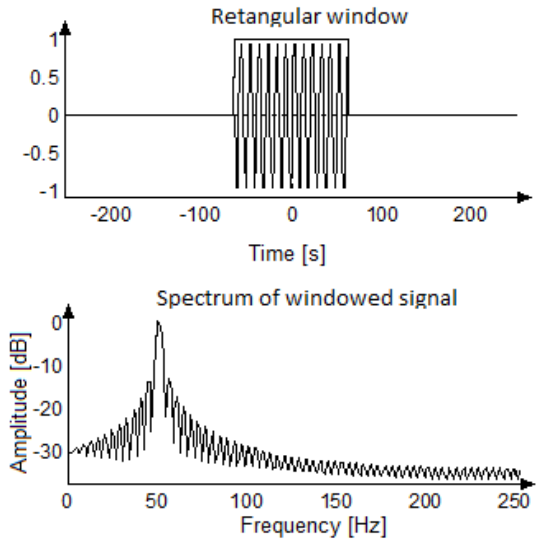


Figure 6: Rectangular window and its effect in spectrum of signal.

In our protocol is used Hanning window which is defined by:

$$w(n) = 0.5(1 - \cos\left(\frac{2\pi n}{N-1}\right)), \tag{4}$$

where N - amount of analysed samples, n - nth sample.

As one can see at fig. 7 the pitch is wider, hence is easier to determine the correct frequency.

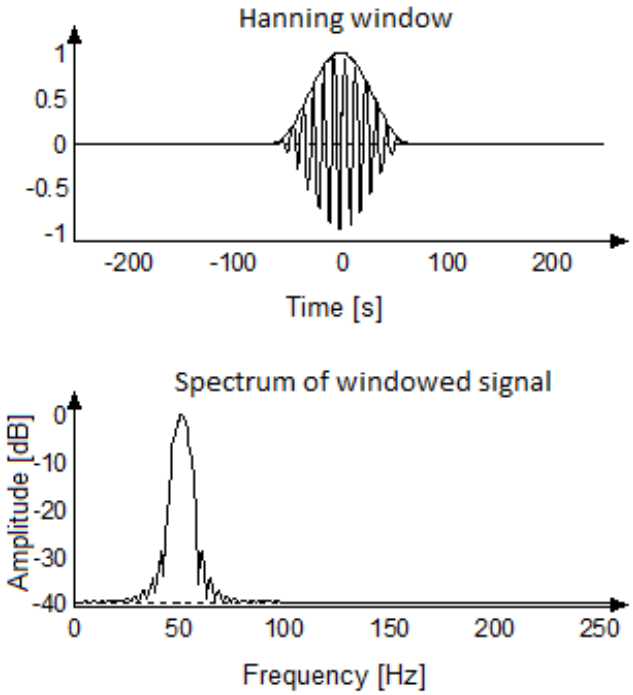


Figure 7: Hanning window and its effect in spectrum of signal.

2.5.9 Pulse-Code Modulation

The application uses *Pulse-Code Modulation* (*PCM*). This modulation is ensured by *AudioTrack* library.

Pulse-Code Modulation is a method used to represent digitally sampled analogue signals. *PCM* is based on representation of the instantaneous value of the signal (Sampling) at uniform intervals (sampling rate). The sampled analogue data are represented by binary data. During the demodulation, sample rate should be at least two times greater than frequency of receiving analogue signal (greater than the Nyquist frequency $f_s/2$). This ensures that sampled values represent analogue signal without errors. In our prototype sample rate is 44,1 kHz. The highest frequency which we use in application is about 20,5 kHz so 44,1

kHz is sufficient.

2.6 Microphones and speakers in mobile devices

Specifications of those appliances are very different and depend on devices. During the tests it could be noticed that microphone from more expensive SAMSUNG Galaxy Tab is better than microphone from LG Swift L9. SAMSUNG could detect higher frequency. However, microphone from LG Swift L9 was completely enough to registered sufficiently large frequencies. Speakers are also slightly different, however without difficulties in generating opportunely frequencies.

2.7 Security analysis over the sound channel

In this section we will analyse the security of chosen out-of-band channel taking into the account its properties. The most popular way to communicate between devices is a channel that uses electromagnetic radiation (WiFi, Bluetooth, IrDA etc.) which consists of propagated in space electromagnetic field disturbance. This way is being chosen because of, inter alia, its speed of transfer data and undetectable for human's senses. However, because of the widespread use, there are a lot of devices which are based on radiation waves and could be used by potential adversary to eavesdropping or manipulating of protocol transcript. In our channel we use mechanical waves, it means that wave propagates as an oscillation of matter, and therefore transfers energy through a medium. In our case the medium is air. So, there are needed different type of devices which use mechanical waves to eavesdropping or manipulating the channel. That is why sound channel was chosen. In terms of security, adversary would have to be prepared to gain different type of spectrum. Using mechanical waves in turns with electromagnetic waves would require from adversary greater involvement and works.

The key exchange protocol is considered to be secured on the assumptions that underlying problems are hard e.g. factorization, discrete logarithm, computational Diffie-Hellman etc. One interesting question is what happens if those assumptions are false. It could happened if implementation of cryptography module has some error or an adversary has super efficient quantum computer that is capable of solving the above problems in reasonable time.

The adversary can have different powers over the communication channel. In the following paragraphs we will discuss different powers of the adversary and their influence over the security of the protocol.

The security analysis below does not include security proofs of AMA but the different aspects of an adversary capturing the transmission if he was able to break the aforementioned assumptions.

2.7.1 No access to protocol transcript

At first the adversary might not be able to eavesdrop the transcript of the protocol at all e.g. in environment where capturing the communication requires sufficient distance to obtain sound signal from the authenticating devices. In that case there are not many requirements for the key exchange protocol being used.

This case is trivial as we can imagine a very simple game in which two parties are hidden from an adversary and he has to guess whether the parties performed a key exchange protocol. Without access to the messages exchanged between the parties, an adversary can only guess with probability $P = \frac{1}{2}$ if the parties performed the protocol. As a consequence he cannot win the "Real-or-Random" game concerning the session key in underlying CK or eCK models for the chosen AKE protocol [24, 32, 31]. Similarly, using the same reasoning, the adversary cannot guess the identities of parties mutually authenticating in the protocol.

However in practice it is possible for an adversary to partially eavesdrop the communication between devices.

2.7.2 Partial access to protocol transcript

Partial access to protocol transcript in sound channel depends on many factors. Participant's microphone, speakers and adversary's microphone have to have appropriate conditions which allow adversary to eavesdrop the transcript. For more details see section 2.9.7 about eavesdropping.

Factors described in 2.9.7 contribute to some probability $\bar{p}$ that adversary and participants will have this type of devices satisfying the conditions. Otherwise, adversary is not able to receive all frequencies. He will gain transcript but with meaningless data. So it would be similar situation when adversary has no access to the transcript.

Lets imagine what will happen when adversary will not gain one of the ephemeral key.

2.7.3 Full access to protocol transcript

Imagine that an adversary uses some kind of microphone with high sensitivity so he can capture all communication between authenticating devices. As mentioned in the previous section, in order to secure against eavesdropping, the key exchange protocol must use problems difficult to inverse like the discrete logarithm problem used in the Diffie-Hellman key exchange scheme.

2.7.4 Possible vectors of attack on a sound channel

There are two ways to attack the sound channel. The first one is trying to generate sound wave which can drown or change sending data by one of the authenticate parties. Additionally one can in some ways force microphone to gain frequencies in another bandwidth. Soundproofing box or case could make it more difficult. Good solution would be mixing of the channels. For example, the first phase (exchanging ephemeral keys) would be processed by out-of-band channel but the second phase (encrypted identity information) by commonly used radiation channel (WiFi, Bluetooth). Then adversary would have to posses devices which detect two types of waves - electromagnetic and mechanic.

2.7.5 Speaker manipulating

One of the vector attack is manipulating of the speaker. Assume that adversary wants to run a malicious programme which takes control over the speaker. He can generate whatever waves he wants, hence he can send malicious data and encryption due to which he can learn something about the protocol. System has to ensure that during the process of key exchange any kind of background application cannot interrupt this. Android ensures security in that way but does not forget about different types of viruses. Unfortunately we do not have influence on this. External manipulation is based on using additional speaker which generates sound in used bandwidth. This method could mislead the receiving microphone. It would be easy to disturb the signal but then recorder gains data without any sense. Attacker would have to start generating sound in perfect time with appropriate intense. Large distance could cause the necessity of speed of sound calculation. High intensity (which would be needed if microphone would have to receive high frequencies from distance) could cause that signal would be audible for human, hence attacker could be exposed. Inserting authenticated parties to some soundproofing box could minimize effectiveness of these methods of attacks.

2.7.6 Microphone manipulating

The manipulation of microphone would be possible only if attacker would have the access to the application. He could e.g. reduce or increase bandwidth of frequencies. Adversary could totally change the bandwidth, outside of the known range for authenticated parties and at the same time generate his sound waves in chosen bandwidth.

2.7.7 Eavesdropping

Eavesdropping in sound channel depends on many factors. At first, microphone has to have an appropriate sensitivity level. Sensitivity level is given by:

$$L_m = 20log_{10}\left(\frac{M}{M_0}\right)[dB], \tag{5}$$

where M is sensitivity [mV/Pa] and M_0 is the 1000 mV/Pa (1 V/Pa) reference output ratio. $M = \frac{U}{p_A}$ where U is the effective voltage microphone and p_A is the sound pressure. Microphone sensitivity M depends also on frequency and the angle of incidence. The important role of receiving signal have the speakers generating this signal. Sensitivity level of speaker is given by:

$$L_s = 10log_{10}\left(\frac{p_A}{p_0}\right)[dB], \tag{6}$$

where p_A is sound pressure generated by speaker powered $1V \cdot A$, p_0 is a reference pressure which has the value of $2 * 10^{-5}$ Pa. Frequency has also an influence on the sensitivity level of the speaker. The next thing which should be considered is the distance between the microphone and the speaker. Too great distance and too low sensitivity of microphone could cause that microphone is not able to gain sounds waves or at least some frequencies. Sound pressure decreases proportionally with increase of distance. So when sound level L_1 is measured at a distance r_1, the sound level L_2 at the distance r_2 is given by:

$$L_2 = L_1 + 20log_{10}\left(\frac{r_1}{r_2}\right)[dB]. \tag{7}$$

There are many devices used for eavesdropping. One of the example might be the microphone with high sensitivity which allows to eavesdrop the sounds even from a large distance e.g. parabolic or shotgun microphone. To gain ultrasounds for eavesdropper, an instrument for detecting ultrasonics from bats would be useful. The next one is using laser [29]. The special laser is a transmitter and the infrared measure of surface vibration. This method also allows to gain data from a large distance.

2.7.8 Malicious implementation

An adversary could try to swap a genuine application for the key exchange on the authenticating device with a malicious one and thus gain the session key. To ensure both parties that they use genuine software, it should be signed with a certificate of a trusted authority and it should be downloaded from a secure server. This case is mentioned in security assumptions in section 3.3.

3 Prototype Design

3.1 Mobile Devices

AMA does not require high-power computing. So it is ideal to implement it on chip cards. Due to the fact the chip cards are slowly replaced by mobile phones we decided to implement AMA on those devices. Android is the chosen operating system, because it is one of the most frequently used software on mobile phones. The lowest version, for which programme

should work without problems is Android 4.1.2. Core of the AMA was implemented in C++. Sound channel and layout was made by using Android's libraries.

3.2 Operating System

The chosen operating system is Android version 4.1.2 and higher. This is the newest version which was available for device which used in tests. The second device for testing involves Android 4.2.2. This software has all the required libraries whereby it is possible to create sound channel between two devices (*AudioTrack* and *AudioRecord*). Additionally, Android allows to use of modules written in C++.

3.3 Security of Android

The operating system should ensure that running application cannot interact with another one. Each application is a run in separate process having its own user ID. This sets up a kernel-level Application Sandbox, where by default application it cannot interact with each other and has limited access to OS.

Furthermore, every application has to be signed by developer. Applications without signature will be rejected by Google Play or the package installer on Android device. Signed certificate is verified by Package Manager after installation of application [11].

3.4 Application architecture

Application consists of three parts: cryptographic module, sound channel module and graphic interface. Cryptographic module is written in *C++* and to use it in Android, was needed to write a wrapper using *JNI* (*Java Native Interface*). *NDK* (*Native Development Kit*) is a toolset that allows to add native-code languages such as *C* and *C++* to Android's application. User interface consists of initial screen where user chooses which frequencies have to be used during sending data. In the next screen there is a dynamic graph with recording sound wave and button which starts the protocol. To show sound as a sinusoid wave we used external library *AChartEngine* [9], the rest of interface is created by standard Android's libraries.

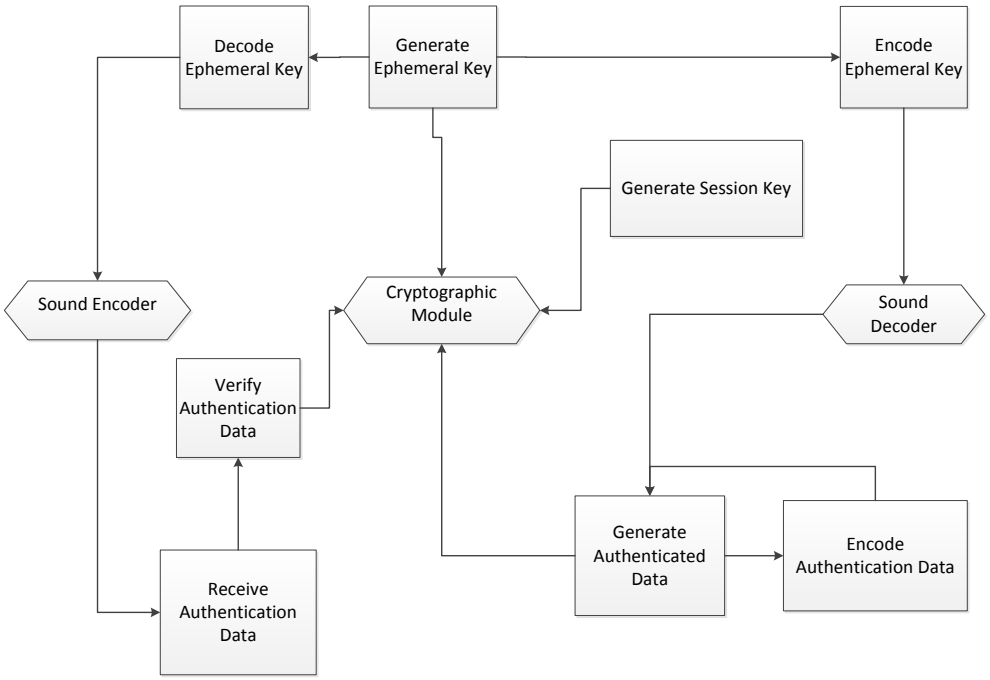


Figure 8: Application architecture.

3.5 Cryptography library

There are few of cryptography libraries which are written in Java (e.g. *javax.crypto*, *BouncyCastle*), but we wanted to create cryptography module which is independent from the platform. The choice fell on *CryptoPP*. It is free, open source cryptography library written in *C++*. The library is objectives and very easy to use. Library ensures symmetric and asymmetric cryptography along with signatures and secures hash function. It is used by such companies as Microsoft or Symantec.

3.6 Design pattern

Application architecture bases on Model-View-Controller pattern. Model includes whole cryptographic module, controller receives and sends data to appropriate places and the view is responsible for sending and receiving data associated with AMA protocol.

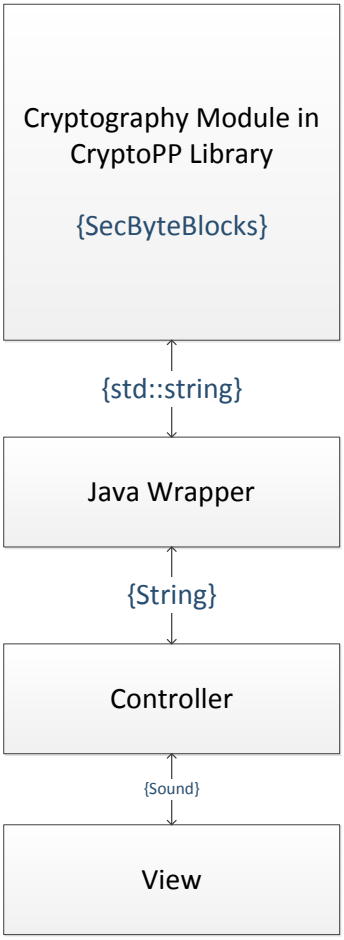


Figure 9: Design pattern.

3.7 NIST recommendation for security parameters

NIST recommends using 3072 bit length keys to ensure low probability of calculating discrete logarithm. This length is set as default in *CryptoPP*, however in our prototype we are using 1024 bit. It is caused by duration of sending data by sound (200b/s). At present, sending 1024 bit length keys takes some time and for the needs of presentation we decided to stay with this length. For more information please check [5].

3.8 Shared cryptographic module

Cryptographic module is designed in a such way that it will be easy to use it on different platforms. Application treats it as a black box where strings with data are exchanged. Implementation is made in C++ based on *Diffie-Hellman* and *Authenticated-Diffie-Hellman* classes from *CryptoPP* library. It performs following cryptographic operation:

- Modular exponentiation
- Long term and ephemeral key generation (Diffie-Helman)
- Certificate generation and verification
- Secure hashing (*SHA1*)
- Symmetric encryption (*AES*)

The core of this module is class *MututalAuthenticationChip*. It is responsible for inputting, preparing and outputting data to Application. All sensitive data are kept internally using secure byte blocks. Messages generated in the protocol are converted to strings. For group arithmetic operations and modular exponentiation we use build in Integer class provided by the library.

The overall implementation tightly follows the protocol specification and the only overhead is connected with conversions to provide compatibility with the rest of the application.

```
1  void KeyGenerator::GenerateEphemeralKeyPair(CryptoPP::RandomNumberGenerator &rng,
       byte *privateKey, byte *publicKey) const
2  {
3      Integer a = Integer(rng, keySize);
4      byte *aEncoded = new byte[keySize];
5      a.Encode(aEncoded, keySize);
6      privateKey = HashClass::getSHA1(aEncoded, keySize);
7      CryptoPP::Integer exponent(privateKey, keySize);
8      CryptoPP::Integer cA = a_exp_b_mod_c(g, exponent, p);   //ca = g^H(a)
9      cA.Encode(publicKey, keySize);
10 }
```

Listing 1: Example of CryptoPP usage for ephemeral key generation.

```
1  byte * HashClass::getSHA1(const byte * input, int length)
2  {
3    CryptoPP::SHA1 sha;
4    byte *digest = new byte [CryptoPP::SHA1::DIGESTSIZE];
5    sha.CalculateDigest(digest, input, length);
6    return digest;
7  }
```

Listing 2: Example of CryptoPP usage for hash function.

3.9 Java Native Interface - C++ Wrapper

To be able to use cryptographic part written in *C++* in Android, we had to create *C++* wrappers using *JNI* (*Java Native Interface*) [2]. Main class which communicates with *C++* code has native methods, for which, after appropriate compilation, *MACWrapper.h* was created where definition of methods looks like below.

```
/*
 * Class:     com_example_androidake_MutualAuthenticateChip
 * Method:    prepareMACCPP
 * Signature: (Z)V
 */
JNIEXPORT void JNICALL
    Java_com_example_androidake_MutualAuthenticateChip_prepareMACCPP
  (JNIEnv *, jobject, jboolean);
```

Listing 3: Example of JNI usage for method definition.

Method implementation is in *MACWrapper.cpp*. There are made conversions from *C++* data type to Java data type and vice versa.

```
JNIEXPORT void JNICALL
    Java_com_example_androidake_MutualAuthenticateChip_prepareMACCPP
(JNIEnv *env, jobject thisObj, jboolean jinit) {
  bool init = jinit;
  if(init == true) {
    mac = new MutualAuthenticationChip(init);
  } else {
    mac_B = new MutualAuthenticationChip(init);
  }
};
```

Listing 4: Example of JNI usage for method implementation.

3.10 Base64

One of the biggest problem was a decision in which representation data should be sent. The first choice was standard binary representation with 0 and 1. During creation of sound channel we found out that sending one ephemeral key, which has 1024 bits takes too long. So we decided to reduce sending data using *Base64* conversion. Six bits are transformed to one of the 64 defined signs. Hence, instead of sending 1024 sings we send just about 172. Data which come from cryptographic module are in Hexadecimal representation. To converse we have used library created by Robert Harder [16].

```
public static String fromHexToBase64(byte[] hex_byte) {
  String hex_str = ConverterJava.ByteToString(hex_byte);
  byte[] decodedHex = null;
  try {
    decodedHex = Hex.decodeHex(hex_str.substring(0,
        hex_str.length() - 1).toCharArray());
  } catch (DecoderException e) {
    e.printStackTrace();
  }
```

Character	Frequency [kHz]
A	10
B	10,15
C	10,3
⋮	⋮
/	19,25
,	19,4
.	19,55

Table 2: Mapping characters to frequencies.

```
10    String encodedHexB64 = Base64.encodeBytes(decodedHex);
11    return encodedHexB64;
12  }
```

Listing 5: Example of code converting from Hex String to Base64 String

3.11 Sound channel

Sound channel was one of the most difficult part during creation of application. One of the reasons was the inaccessibility of library whereby we could transform received sound data to cryptographic data. The whole process of analysis received data had to be created from the scratch.

3.11.1 Generating sound wave

Class responsible for generating sound wave gets data presented in Base64 (Check section with Base64). Every character A-Z, a-z, 0-9, + and / is mapped to one of the 64 frequencies.

Additional two characters "," and "." represent the beginning of the data and the end of the data. Sample rate is 44,1 kHz. Every character is sent by 30 ms (this is the lowest value which we achieve during optimization of prototype), which means that for each pitch 1323 samples are needed. To create sinus wave each sample for $i \in (0, 1322)$ is calculating by formula:

$$F(i) = \sin\left(\frac{f \cdot \pi \cdot 2 \cdot i}{44100}\right)$$

where f is the target frequency. These samples are processed by *AudioTrack* library to sound using *PCM* encoding. In the listing 6 is shown the piece of code preparing sinus waves in appropriate frequencies.

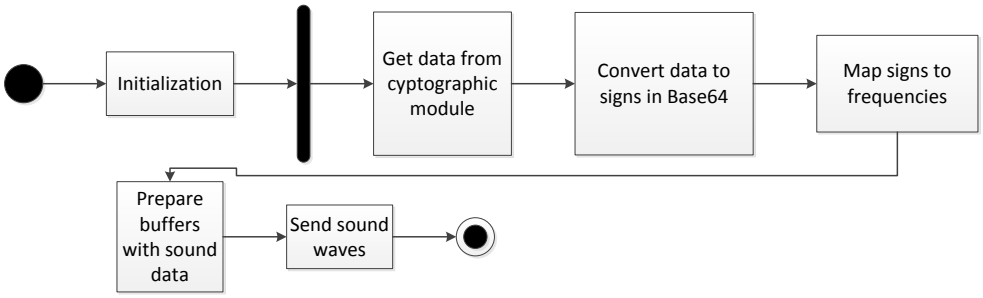


Figure 10: Activity diagram of sending data.

```
/* --------- Constants and variable used in code -------------------------*/
// Constants.DEFAULT_NUM_SAMPLES = 1323; - Number of samples needed to
//                        encode one character.
// Constants.FREQUENCIES - all 66 used frequencies;
// Constants.DEFAULT_BUFFER_SIZE = 8192;
// Constants.RAMP = 30; - on the edge of buffor with samples is needed
//              to decrease the amplitude to avoid scratching
// Input:
// List<Integer> inofsign; - signs which will be converted on frequencies
/*---------------------------------------------------------------------------*/

public ArrayList<Buffer> encodesDataToBuffers(List<Integer> inofsign) {

  ArrayList<Buffer> queue_with_data_AL = new ArrayList<Buffer>();
  for (int index : inofsign) {

    int totalCount = Constants.DEFAULT_NUM_SAMPLES;
    double per = (double) (( Constants.FREQUENCIES[index] * 2 * Math.PI )  /
        Constants.SAMPLING);
    double d = 0;

    int mFilledSize = 0;
    Buffer buffer = new Buffer();
    int ramp = totalCount / Constants.RAMP;
    short[] buffer_data = new short[Constants.DEFAULT_BUFFER_SIZE];

    for(int i = 0; i < totalCount ; ++i){

      if(mFilledSize >= Constants.DEFAULT_BUFFER_SIZE - 1){
        buffer.setBufferShort(buffer_data);
        buffer.setSize(mFilledSize);
        mFilledSize = 0;
        queue_with_data_AL.add(buffer);
        buffer = new Buffer();
        buffer_data = new short[Constants.DEFAULT_BUFFER_SIZE];
      }
      double out = (double) Math.sin(d);
      final short val;
      if(i < ramp){
        val = (short) ((out * Short.MAX_VALUE * i / ramp));
      }else if(i < totalCount - ramp){
        val = (short) ((out * Short.MAX_VALUE));
      }else{
        val = (short) ((out * Short.MAX_VALUE * (totalCount-i)/ramp));
      }

      buffer_data[mFilledSize++] = (short) ( val );
      d+=per;

    }


    buffer.setBufferShort(buffer_data);
    buffer.setSize(mFilledSize);
    queue_with_data_AL.add(buffer);

    mFilledSize = 0;

  }
  return queue_with_data_AL;
}
```

Listing 6: Method converting characters to sinus waves.

3.11.2 Receiving sound

Retrieving sound is more complicated than sending. In separate process, using *AudioRecord* library, in every 15 milliseconds (if sending of one character takes 30 ms), samples are taken and then, as one buffer, are appended to array of buffers. Separate process, at the same time, takes first buffer from array and analyses it.

At this moment, the samples are as short integer values which represent sound in time domain (sinusoid). To check in which frequency the sound was received there is a need to convert the sampled function from time domain to the frequency domain. For this action we used *discrete Fourier transform* . The library chosen to run *DFT* is [3].

Before the change of domain, we need to pass samples through windowing Hanning function. Thereby, chart in frequency domain is more readable and it is easier to find the place with the highest pitch. Windowed samples are put into *DFT* method and on the output we have result in frequency domain. The effect on the graph looks like at 11.

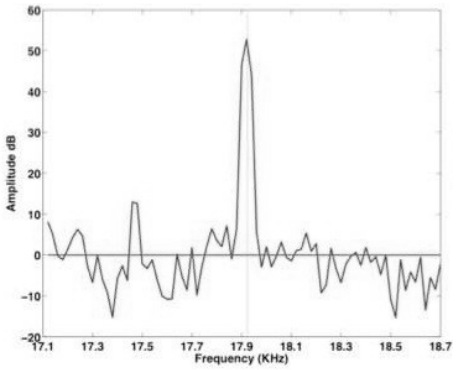


Figure 11: Sound wave in frequency domain.

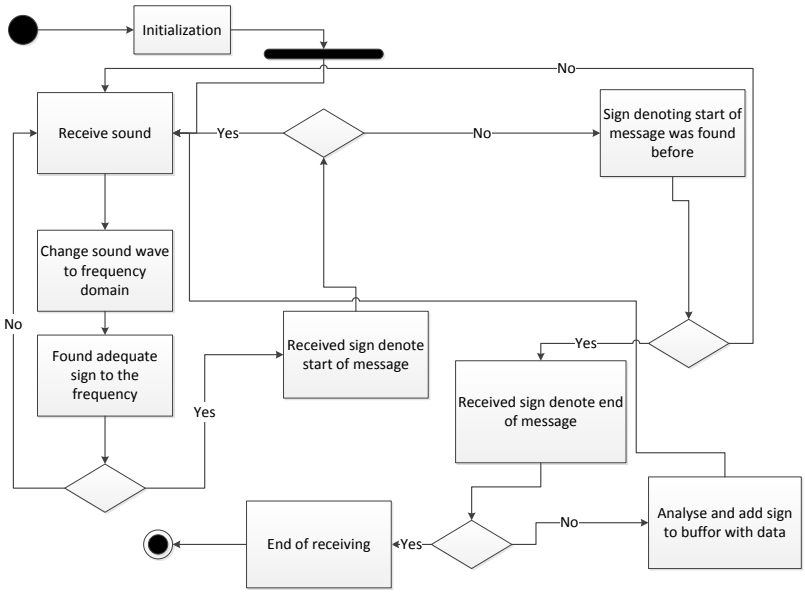


Figure 12: Activity diagram of receiving data.

Now, algorithm tries to find the highest place in the output. The reason of getting data within twice shorter time than sending one character is that, in one buffer there can be two frequencies. In the worst case, algorithm can find one frequency in three buffers in a row, even if in the first and the third buffer there was also another frequency, but because of lower sound intensity or less samples representing this frequency, it would not be found as the highest pitch. Dividing time of receiving by two ensures that always, at least once, expected frequency will be registered. If it finds pitch with amplitude higher than 50, for frequency lower than 14 kHz (this value was chosen after testing of prototype, these frequencies always exceed this limit), or 10 for higher than 14 kHz (this difference follows from the smaller volume for the higher frequency by hardware limitation) is checked whether caught frequency is in our dictionary (with margin of error + - 50 Hz).

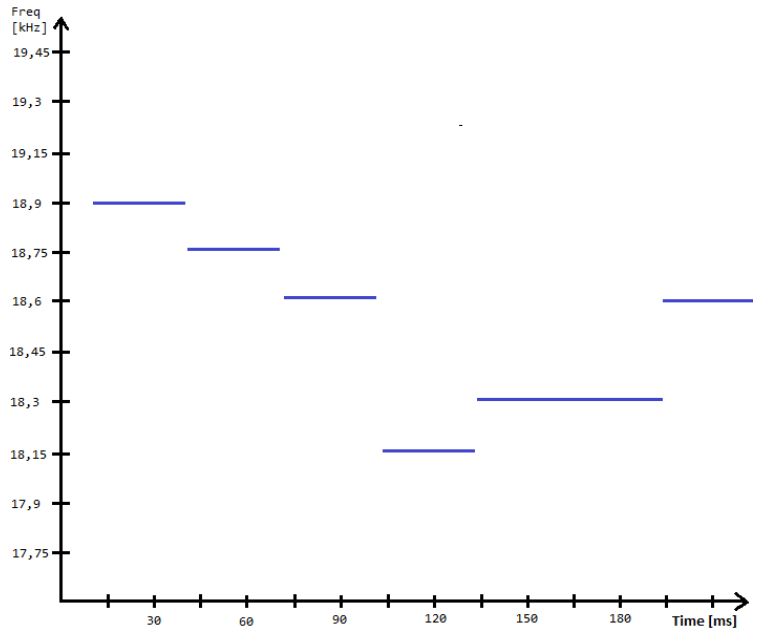


Figure 13: Figure shows received frequencies in time.

Caught value is put to the memory and waits for the next result. If current frequency is different than previously we change the previous frequency to one of the 64 signs. Otherwise we increment the counter. If counter gains the value of three (maximal number of occurrences on received character in three further buffers), counter is reset and returns the adequate character.

```
/* --------- Constants and variable used in code -------------------------*/
// Constants.AMPLITUDE_LT_14K = 50; - Limit of amplitude for frequencies
                        lower than 14 kHz
// Constants.AMPLITUDE_GT_14K = 10; -  Limit of amplitude for frequencies
                      greater than 14 kHz
// Constants.SAMPLING = 44,1 kHz - sampling rate of sound
// Constants.K14 = 2660 - place in fft array when starts frequencies greater than
    14 kHz.
// Constants.START_ST = 1800 - place from which algorithm must start to searching
    peak.
// Input:
// FFT fft; - data prepared by Discrete Fourier Transform.
/*-----------------------------------------------------------------------*/
private int findPitch(FFT fft) {
  float max_band = 0;
  int max_peak = 0;
  int k14 = Constants.K14;
  for (int i = Constants.START_ST; i < fft.specSize(); i++) {
    if (i < k14) {
      if (fft.getBand(i) > Constants.AMPLITUDE_LT_14K)
        if (fft.getBand(i) > max_band) {
          max_peak = i;
          max_band = fft.getBand(i);
        }
    } else {
      if (fft.getBand(i) > Constants.AMPLITUDE_GT_14K)
        if (fft.getBand(i) > max_band) {
          max_peak = i;
          max_band = fft.getBand(i);
        }
    }
  }
  int frequency = max_peak * (Constants.SAMPLING / 2) / fft.specSize();
  return frequency;

}
```

Listing 7: Method finding frequency with highest aplitude.

```
/* --------- Constants and variable used in code -------------------------*/
// Constants.FREQUENCIES - all 66 used frequencies;
// Constants.BORDER = 50 - error range
Input:
int freq; - Found frequncy by method findPitch(FFT)

/*-----------------------------------------------------------------------*/

private void checkPitch(int freq) {
  String sign = "";
  if (freq > Constants.FREQUENCIES[0] - 100) {
    if (current_freq != null) {
      if (current_freq.getFrequency() - Constants.BORDER < freq
      && current_freq.getFrequency() + Constants.BORDER > freq) {
        current_freq.increaseCount();
        sign = current_freq.foundAndReturnChar();
        if (!sign.equals(Constants.NOEND_STR)) {
          vrs.onRecognition(String.valueOf(sign));
        }
```

```
20        } else {
21          sign = current_freq.returnChar();
22          if (!sign.equals(Constants.NOEND_STR)) {
23            vrs.onRecognition(String.valueOf(sign));
24          }
25          current_freq = new FrequencyTime();
26          current_freq.setFrequency(freq);
27        }
28      } else {
29        current_freq = new FrequencyTime();
30        current_freq.setFrequency(freq);
31      }
32    } else {
33      if (current_freq != null) {
34        sign = current_freq.returnChar();
35      if (!sign.equals(Constants.NOEND_STR)) {
36        vrs.onRecognition(String.valueOf(sign));
37      }
38      current_freq = null;
39      }
40    }
41  }
```

Listing 8: Method changing founded freqency on appropriate character.

4 Tests

To all tests two devices was used: LG Swift L9 with system Android 4.1.2 and tablet Samsung GALAXY Tab 2 with Android 4.2.2.

During tests we have focused on below factors:

- Distance between devices,
- Position of devices in the terms of placement of speaker and microphone,
- Distance between chosen frequencies,
- Working during sounds in the background,
- Volume of sound.

Firstly, we tested the distance between chosen frequencies. Every character from *Base64* coding is mapped to appropriate frequency. With smaller distance it is possible to use higher frequencies. We choose three spaces: 100, 130 and 150 Hz.

Distance [Hz]	The lowest frequency [kHz]	The highest frequency [kHz]
150	10,5	20,25
130	11,8	20,25
100	13,8	20,3
100	14	20,5
100	14,3	20,8

Table 3: Mapping characters to frequencies.

We did not want to go below 10 kHz and higher frequency than 20,5 it is hard to detect for mobile phone (especially for LG). Distance 100 Hz is the smallest to test because of implemented algorithm (margin of error is + - 50 Hz). During the tests, it was proved to be for almost every test cases, protocol was successful in 80%. Except the last one which always ends by failure. Character which represents end of sending data has always the highest frequency. In the last case it was 20,8 kHz. Only LG Swift cannot registry that frequency - Samsung dealt with it flawlessly. Duration of protocol was about 34,5 seconds.

There are several reasons why the result was not 100%. Sometimes, if we send, for example, frequency 20 kHz then second device receives it as 19,9. It can be caused by some distortion on the microphone or even on the speaker. What happened was that the speaker made sound different than usual and then there was a need to restart the device. Sometimes it is sufficient that one of the devices froze on some milliseconds and then receiving device loses some characters.

Distance [cm]	Session ended with success
0	success
2	success
3	failure
4	failure

Table 4: Mapping characters to frequencies.

To measure the allowed distance between devices we started with stick microphone from LG with speaker from Samsung. That gave us the best results. Received amplitude was the biggest, therefore it was easier for algorithm to find the needed frequency. Then we increased the distance to 2 cm. In this case results were also good. Session keys were established almost every time. Problem started when we increased the distance to 6 cm. Higher frequencies were not registered by microphone. Placing the devices parallel the protocol always ended by failure. The LG was not able to gain all frequencies in such way. Samsung was still receiving data without problems.

Type of noises	Session ended with success
School at break	success
8 kHz, distance 5 cm	success
8 kHz, distance 0 cm	success
10,5-20 kHz, distance 5 cm	success
10,5-20 kHz, distance 2 cm	false

Table 5: Mapping characters to frequencies.

The next test relies on checking the influence of background sound. As we know, from the previous test, the best results were when devices were close to each other (maximal 2 cm from each other), so we have tested it in such position. At beginning we checked it at the University during the break between lectures. A lot of students were talking and laughing but even that, did not interfere with protocol. Almost every attempt ended with a success. Then we used third device which was generating constantly chosen frequency. We put this device very close to the participants. First disturbing frequency was 8 kHz. Because of that bandwidth is from 10,5 to 20,25 we suspected that 8 kHz does not cause the problem and we were right. Algorithm analysing data was correct. The problem was just after increasing the disturbing frequency above 10,5 kHz, but it was needed to stick speaker of that device to the microphone of participant protocol. Otherwise, the protocol always ends with a success.

Volume set on 70% on both devices was sufficient to run the protocol. Setting this value below 70% cause that character denoting start of transmission was not detected.

5 Possible development of the project

Lets imagine that Alice and Bob want to establish secure channel being far away from each other. They can authenticate each other using a telephone line. It is possible to run application during the calling but the microphone receiving the signal is unavailable in the application. So to this experiment the second pair of mobile phones would be needed to be used as intermediary. Each intermediary phone should be set as handsfree because of sound loudness. Authenticated phones should be as close as possible to intermediary phone. When intermediary phones establish the connection, the authenticated devices can start the protocol, the sound signal is transferred by telephone line. Device on the other side receives the signal. It is possible use computers as intermediary. There are a lot of voice communicators which might be useful.

Disadvantage in our protocol is the speed of transferring data. In every 15 milliseconds there are samples received which are put to the queue. Another process gets these samples and analyses them. For our test devices, receiving data in every 15 seconds was the most

optimal method. Algorithm analysing this data follows with computing, so in queue there is no accumulation of too many data. Decreasing period of receiving data to 10 millisecond causes that decoder of sound does not keep up and outcome is sometimes a couple of seconds after the last received character. To improve that it is worth to consider client-server model like in [14]. Mobile phone as a client gathers the data and sends to the external server. Server with much more computing power could decode the data definitely faster and returns the result to the client. Server could do much more complex computing, hence the results would be more precise.

For now the sound channel uses bandwidth from 10,5 kHz to 20,25 kHz. For the reason please check section with *Base64* conversion. These frequencies (especially below 19 kHz) could be audible for the most people (it sounds like squeaking). If we want to exchange data using sound which is inaudible, we could use just simple binary representation. Sending data using 0 and 1 there would be only two frequencies needed, so taking the highest as possible (for example 19,9 kHz and 20,10) would ensure that exchanging of data would be inaudible for the most people. Of course at the expense of length of sending data. It would be achievable also with hexadecimal representation. It would be needed to use bigger bandwidth than from binary representation but the length of data would be smaller.

6 Summary

Dissertation shows that is possible to use sound channel to establish session key. There is no need of Internet connection or another electromagnetic channels, so proposed protocol Anonymous Mutual Authenticated could be called even in the place without reach, using mobile devices with embedded speaker and microphone.

In the prototype we used the bandwidth of frequencies which are audible for human, however it is possible to use only these frequencies which are not detected by humans ears. Our mobile devices, which are medium priced products did not have any problems to send and receive frequency between 19-20,5 kHz which is inaudible for the most people.

We showed that using this type of channel could surprise the attacker who might not be prepared to eavesdrop/manipulate exchanging data in mechanical waves. Mixing the sound channel with other kinds of channels, for sure, could make it even more difficult. Duration of the protocol execution is not so fast as in radiation channels, however it is good enough to be applied in practical solutions. Establishing the authentication could be much faster if used client-server model. Computer as a server would analyse data faster and results would be better.

References

[1] The descrete transform fourier (dft): Definition and numerical examples. http://www.nbtwiki.net/doku.php?id=tutorial:the_discrete_fourier_transformation_dft#.UND9vuT6Jv8.

[2] Java programming tutorial - java native interface (jni). https://www3.ntu.edu.sg/home/ehchua/programming/java/JavaNativeInterface.html.

[3] Minim fft class. http://code.compartmental.net/minim/javadoc/ddf/minim/analysis/FFT.html.

[4] Nateżenie fali dźwiękowej. http://www.fizykon.org/akustyka/akustyka_natezenie_fali.htm.

[5] Nist sp 800-131 recommendation for key length. http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf.

[6] Poziom nateżenia dźwieku. http://www.fizykon.org/akustyka/akustyka_poziom_natezenia.htm.

[7] The speed of sound. http://www.sengpielaudio.com/calculator-speedsound.htm.

[8] Vibrations usage in out-of-band channel.

[9] AChartEngine. http://www.achartengine.org/.

[10] Scott A. Vanstone Alfred J. Menezes, Paul C. van Oorschot. *Handbook of Applied Cryptography*. CRC Press.

[11] Android. https://source.android.com/devices/tech/security/.

[12] Applidium. http://applidium.com/en/news/data_transfer_through_sound/.

[13] Anton Mityagin Brian LaMacchia, Kristin Lauter. *Stronger Security of Authenticated Key Exchange*. Microsoft Research, 1 Microsoft Way, Redmond, WA.

[14] Chirp. http://chirp.io/.

[15] Lucjan Hanzlik, Kamil Kluczniak, Łukasz Krzywiecki, and Mirosław Kutyłowski. *Mutual Chip Authentication*. Faculty of Fundamental Problems of Technology, Wrocław University of Technology.

[16] Robert Harder. Base64. http://iharder.sourceforge.net/current/java/base64/.

[17] Steve Haynal and Heidi Haynal. *Generating and Searching Families of FFT Algorithms*. http://jsat.ewi.tudelft.nl/content/volume7/JSAT7_13_Haynal.pdf.

[18] M.K. Reiter J.M McCune, A. Perrig. *Seeing-is-believing: using camera phones for human-verifiable authentication*. Carnegie Mellon Univ., Pittsburgh, PA, USA.

[19] Hugo Krawczyk. *HMQV: A High-Performance Secure Diffie-Hellman Protocol*. IBM T.J.Watson Research Center, Yorktown Heights, NY.

[20] Hugo Krawczyk. *SIGMA the 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and its Use in the IKE Protocols*. EE Department, Technion, Haifa, Israel, and IBM T.J. Watson Research Center.

[21] Anton Mityagin Kristin Lauter. *Security Analysis of KEA Authenticated Key Exchange Protocol*.

[22] Joshua D. Guttman Wenjing Lou Kui Ren Ming Li, Shucheng Yu. *Secure Ad Hoc Trust Initialization and Key Management in Wireless Body Area Networks*.

[23] Pawel Nuzka. *Authentication and communication protocol for mobile devices in optical channel*. Faculty of Fundamental Problems of Technology, Wroclaw University of Technology.

[24] Hugo Krawczyk Ran Canetti. *Analysis of key-exchange protocols and their use for building secure channels. Advances in Cryptology*. EUROCRYPT, 2001.

[25] M. Merrit S. Bellovin. *Encrypted key exchange: Password-based protocols secure against dictionary attacks.* In Proceedings of the IEEE Symposium on Security and Privacy, 1992.

[26] M. Merrit S. Bellovin. *Augmented encrypted key exchange: a password-based protocol secure against dictionary atttacks and password file compromise*. In Proceedings of the ACM Conference on Computer and Communications Security (CCS), 1993.

[27] Kazuhisa Miyashita Akeharu Okumura Yoshiaki Yoshida Shintaro Takeda, Ikuharu Morioka and Kenji Matsumoto. *Age variation in the upper limit of hearing*.

[28] Nisha Panwar Michael Segal Shlomi Dolev, Łukasz Krzywiecki. *Certificating Vehicle Public Key with Vehicle Attributes*.

[29] Mike Szczys. Laser mic makes eavesdropping remarkably simple. http://hackaday.com/2010/09/25/laser-mic-makes-eavesdropping-remarkably-simple/.

[30] Berkant Ustaoglu. *Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS*.

[31] SaanJae Moon Xinghua Li, Jianfeng Ma. *On the security of Cenetti-Krawczyk model*.

[32] Zheng Yang. *Efficient eCK-secure Authenticated Key Exchange Protocols in the Standard Model*.