

# Architecture of One Time Password with Voice Call from SIP to PSTN

Esa Fauzi<sup>1</sup>, Acep Edison<sup>2</sup>

**Abstract**---One Time Password (OTP) is a one way authentication mechanism that usually used to verify user. Generally the media used in OTP is SMS (Short Text Messages). But besides SMS, sound can be used as an alternative media.

SIP (Session Initiate Protocol) is a protocol that can send multimedia objects including sound. SIP can also be connected to a PSTN (Public Switched Telephone Network) through a media gateway. Therefore, this research focuses on building a system that can send OTP from SIP to PSTN.

The result of this research is an API (Application Programming Interface) developed with the REST (Representational State Transfer) architecture. This API can send OTP audio from SIP to PSTN. Besides it, This API has also been tested to send many OTP simultaneously at the same time.

**Keywords**---One-time password, SIP, PSTN.

---

## I. Introduction

One of the attacks in computer networks is the 'replay attack' [1]. Replay attack is eavesdropping attacks on network connections to obtain authentication information such as username and password. To avoid it, one way that can be done is by using a one-time password (OTP) mechanism. One-time password will provide a password that can be entered by the user to verify himself within a certain timeframe. One-time passwords can also be used only once so that even if they are stolen, they cannot be used anymore.

Generally the one-time password media is SMS. An application that uses a one time password will send an SMS to the user's phone. After receiving the password, the user can input the password into the application to verify himself. Besides SMS, voice calls over the telephone is an alternative media that can be used in a one-time password. This means that the password sent is not in the form of text as in SMS but in the form of voice calls.

To implement OTP with telephone calls, one technology that can be used is the Session Initiate Protocol (SIP) [2]. SIP is a signal protocol on the internet that can be connected to traditional telephone networks or public switched telephone networks (PSTN).

In this research, an API with an object oriented mechanism was developed (using the Java programming language) that can provide OTP via telephone calls using the SIP protocol. This API can also use own telephone numbers and audio that entered by the user to send an OTP. In addition, this API was developed so that it can send OTP simultaneously at the same time

---

<sup>1</sup>Department of Informatics Engineering, Faculty of Engineering

## II. Related Works

Research related to OTP using SMS media has been done a lot, some of them are research by Anusha that uses secure hash algorithm [3] and research by Nugroho with the AES 256 bits method [4]. Both studies focus on methods for generating OTP while this research only focuses on converting OTP to voice and then sending it over the PSTN network or telephone.

In addition, research related to OTP using SMS has several disadvantages, one of them is the stolen data using trojans as revealed by Mulliner [5]. Trojans that are installed on a mobile phone can intercept the SMS then forwards it to a certain number. This weakness is one of the reasons for this research to replace OTP SMS media with sound.

Currently, although there are many applications that send OTP by voice call, the author has not found a single paper that writes about it. However, there are several papers that write the relationship between SIP and PSTN including Zhang Yuan that writing a paper on how to interconnect between SIP and PSTN networks [[6]. Then there is also Gupta who wrote a paper about the gateway for PSTN on the asterisk framework that can be connected also with SIP [7].

In addition there is also research by Dorais who examines how to do simultaneous voice transmission using SIP [8]. Simultaneous transmission like this was done also in this study because this system must also to able handle many requests to sending OTP at the same time. But unlike Dorais, this study conducted simultaneous transmission of voice on the PSTN network.

## III. Analysis and Design

### Overview System

This research was designed to produce an API. The voice-otp API is divided into 3 sub-components. The first sub component is the settings sub component. In this sub-component, voice-otp API users can input their own audio. The input sound consists of the numbers 0-9 along with the opening sentence. In addition to this sub-component, users can also input their own phone number (user part) so that when an OTP is sent, the phone number that appears on the device (for example a mobile phone) is the phone number that has been set by the user.

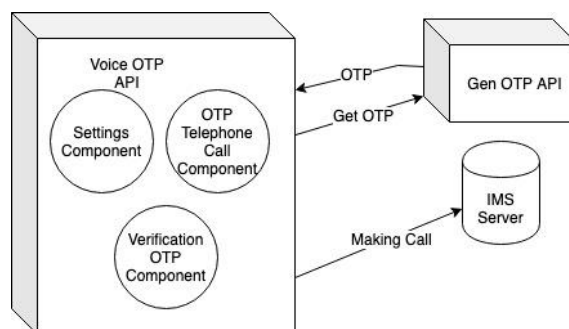


Fig. 1: Architecture API

Then the second sub component is telephone calling. In this sub component, basically the user will send an OTP with a telephone call. The user then enter data consist the target telephone number, key code for OTP, and the number of OTP digits (maximum 9 digits).

Then the last sub component is the OTP verification component. This sub component is used to verify the OTP code that has been sent. The data used for verification is the OTP itself, the key code, and the number of OTP digits.

In addition to this voice-otp API, the OTP code itself is not made by voice-otp API but is obtained through another API, namely the OTP Gen API. This OTP Gen API will generate an OTP code based on the key requested by the user with a time limit of 2 minutes. Then for the server used is the IMS server.

## 2) Use Case

Users who use the voice-otp API are called subscriber, because to use this API the user must subscribe to the API marketplace website where the voice-otp API is stored. There are 3 main features for the subscriber on this voice-otp API: the settings, sending OTP, and verifying OTP.

### a. The Settings

Based on the use case image 2, in the settings, the subscriber has 2 main features: input audio and input user part phone number. Audio is input if the subscriber wants to use its own voice to send the OTP. Whereas the user part to display the specific telephone number desired by the subscriber.

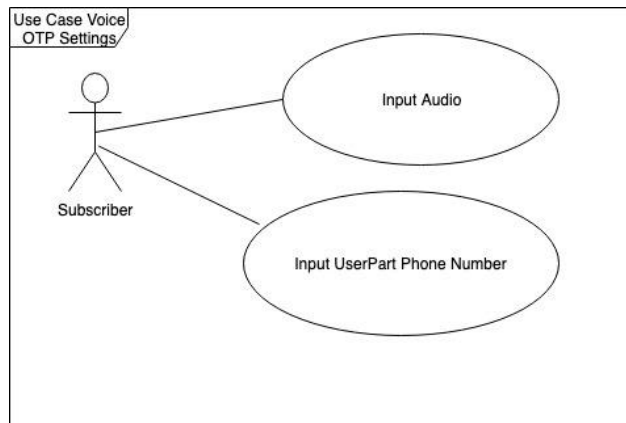


Fig. 2: Use case setting audio and user part

### b. Send the OTP

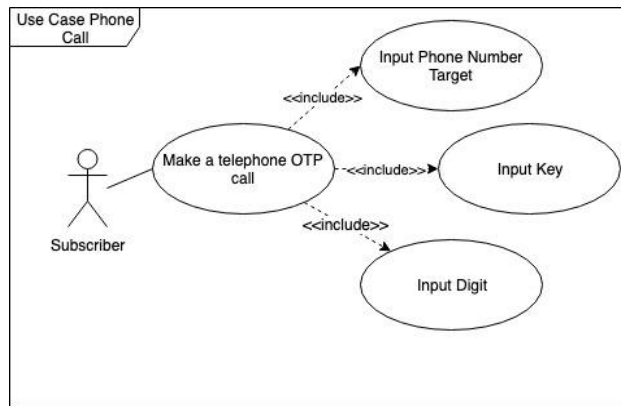


Fig. 3: Use case phone call

For the process of sending OTP with a telephone call there are 3 parameters that must be inputted by the user. First is the target phone number. The second is the key as an id which is used to generate and verify the OTP code. Third is the number of OTP digits.

c. Verified OTP

After the OTP is sent, the subscriber can verify the OTP. There are 3 parameters that must be sent, the OTP code itself, the key code, and the number of digits.

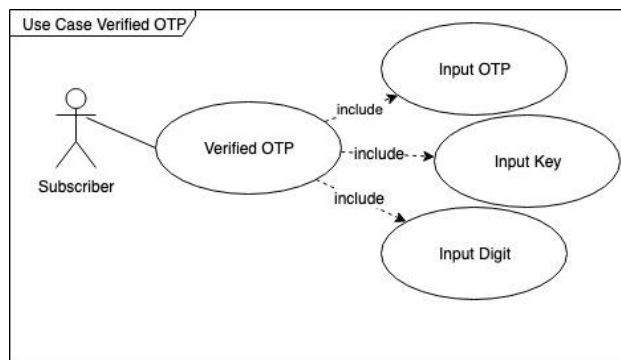


Fig. 4: Use case verification OTP

2) System Flowchart

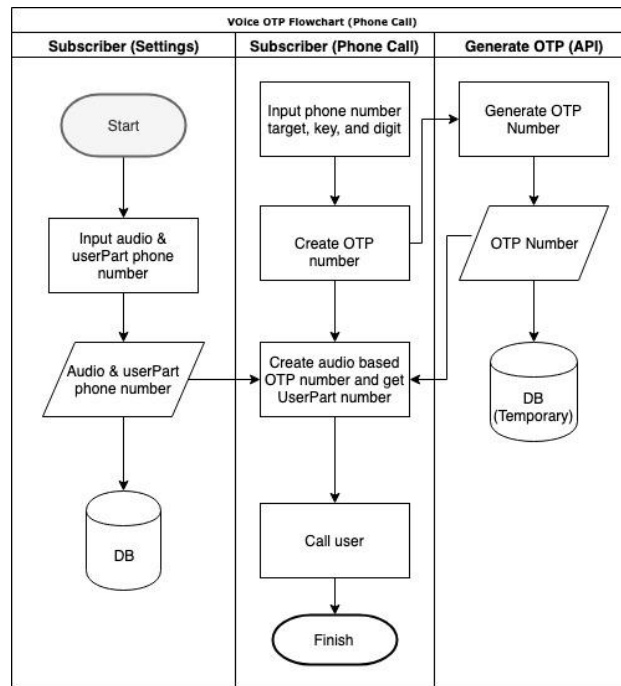


Fig. 5: Flow chart setting and send OTP

Figure 5 explains the process flow of the voice-otp API. Subscriber in the first process can input audio and user part phone number. The audio and user parts will later be saved into the My SQL database.

When there is an OTP sending request, the voice-otp API will generate a OTP code by calling the genOTP API. GenOTP API is an API used to generate API code. The voice-otp API does not create its own OTP but instead requests it from other API.

After getting the OTP code from the genOTP API, the voice-otp API will convert the OTP code into audio. Audio itself is created based on a sequence of numbers from the OTP code which is then matched with the sounds that have been saved. The audio that match with OTP code then will be combined into one. The voice-otp API also retrieves user part for identity phone numbers that will be displayed when the OTP code is sent.

If audio and user part already exist, the last step in the voice-otp API is to make a telephone call. But the request to make the telephone call must be queued up first. This is because race conditions can be occur [9]. The IMS server used by the voice-otp API can only handle 30 phones simultaneously. Therefore, in the algorithm, when there are multiple requests simultaneously, the first 30 requests will be executed immediately while the 31st request and then will be store into queue. To avoid race conditions in the queue, the queue must be handled and in the voice-otp API is using synchronization [10]. In addition, for the telephone calling process, voice-otp uses the library peers [11]

Algorithm 1. Call phone

```
public void callPhone(RequestOTP obj, QueueOTP list) {  
    synchronized (list) {  
        list.enqueue(obj); // add to queue  
        if (list.size < 31) {  
            RequestOTP callOTP = list.dequeue();  
            callOTP.getOTP();  
            callOTP.createAudio();  
            callOTP.getUserPart();  
            callOTP.loadProxyServerData();  
            callUser(otp); // make new thread  
        }  
    }  
}
```

When sending OTP is complete, before ending the process, an algorithm is added to check the queue first. If there is still a queue then the next queue will be executed. This can be seen in algorithm 2.

Algorithm 2. Check queue list when end call has finished

```
//Algorithm end call  
public void endCall(){  
    synchronized (list) {  
        if (list.size < 31) {  
            RequestOTP callOTP = list.dequeue();  
            callOTP.getOTP();  
            callOTP.createAudio();  
            callOTP.getUserPart();  
            callOTP.loadProxyServerData();  
            callUser(otp); // make new thread  
        }  
    }  
}
```

Algorithm 3. Invite Call

```
/* Transmanager Class to create singleton object of  
transportMangaer and transactionManager */  
Class TransManager {
```

```
private Static TransportManager transport;
private static TransactionManager transaction;

public TransManager(Config config) {
    transport = new TransportManager(Config.ip,
        config.sipPort, config userPart);
    transaction= new TransactionManager(Config.ip,
        config.sipPort, config userPart);
}
public getTransport(){
    return transport;
}
public getTransaction(){
    return transsaction;
}
}
}
.....
void callUser(otp){
    //create object userAgent as UAC
    UserAgent ua=new UserAgent(trans.getTransport(),
        trans.getTransaction());
    new Thread(new Runnable) {
        void run(){
            ua.invite(otp); //call user
        }
    }
}
```

For simultaneous transmission itself, the solution found in this study is to modify the transport and transaction objects in the library peers to singleton so the process of telephone call only use one object along with an IP address, SIP Port of the proxy server. This can be seen in algorithm 3.

Transport and transaction managers that handle audio transmission via UDP [12] will be designed as singleton in the transportManager class. So when creating a user agent (as a user agent client), the user agent will use the configuration of the transport and transaction manager that has been set as singleton. After creating the user agent, the user agent can process the process of invite or call the telephone to the destination phone number.

Finally, if an OTP has been sent, the subscriber can verify the OTP. Basically this OTP verification only matches the OTP with the database stored on the genOTP API (figure 6).

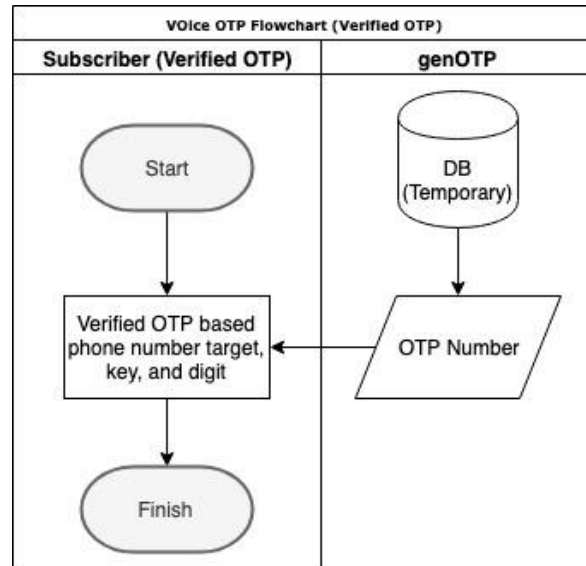


Fig. 6: Flow chart Verification OTP

#### IV. Implementation

The implementation of this research was made into an API with REST architecture [13]. From this API there are 3 main paths:

- a. `/subscriber/{subscriber}/config`

This path is used by the subscriber who will manage the audio and user parts that will be used when sending an OTP.

- b. `/otp/{key}`

This path is used to send OTP via voice call (see figure 7). Key parameters will be stored in the path while the intended telephone number and number of digits are stored in the body.



The screenshot shows an API configuration interface for the endpoint `PUT /otp/{key}` with the description "Generate OTP". A note states: "Use this end point to generate OTP and send it to your customer via phone call. Maximum time to play OTP audio is 30 seconds." The "Parameters" section includes a required `key` parameter of type `string` (path) with the value `abc`. The `body` parameter is of type `object` (body) and contains a JSON object: `{ "phoneNum": "08154666569", "digit": 6 }`. A "Cancel" button is visible in the top right.

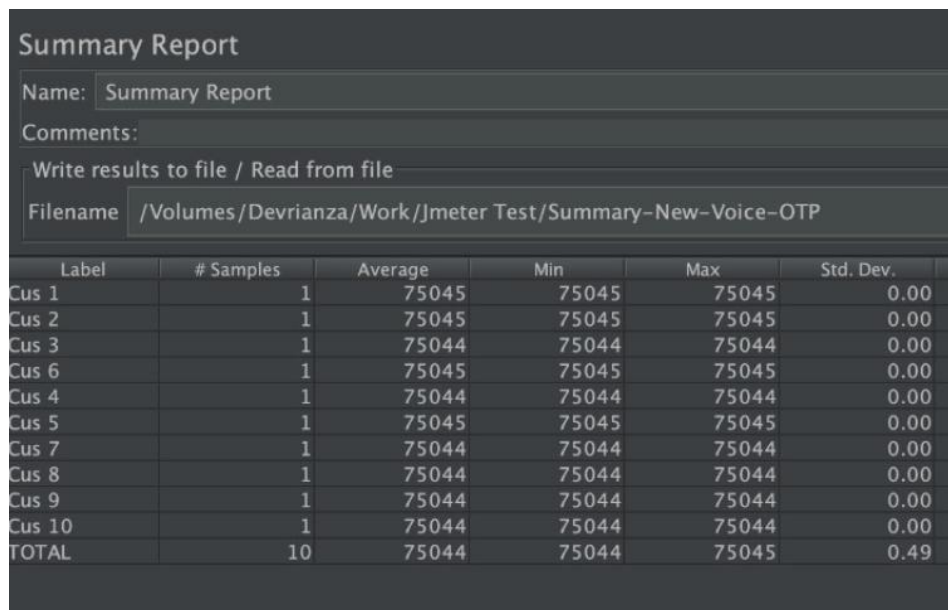
Fig. 7: Implementation voice-otp call in marketplace website

c. `/otp/{key}/verification`

This path is used to verify the OTP that has been sent by the subscriber. The parameters are similar to OTP sending, but include the OTP code that stored in the body (figure 8)

The screenshot shows an API configuration interface for the endpoint `POST /otp/{key}/verifications` with the description "Validate OTP". A note states: "Use this end point to validate OTP submitted by your customer". The "Parameters" section includes a required `key` parameter of type `string` (path) with the value `abc`. The `body` parameter is of type `object` (body) and contains a JSON object: `{ "otpStr": "08154666569", "digit": 6 }`. A "Cancel" button is visible in the top right.

Fig. 8: Implementation verification OTP in marketplace website



Label	# Samples	Average	Min	Max	Std. Dev.
Cus 1	1	75045	75045	75045	0.00
Cus 2	1	75045	75045	75045	0.00
Cus 3	1	75044	75044	75044	0.00
Cus 6	1	75045	75045	75045	0.00
Cus 4	1	75044	75044	75044	0.00
Cus 5	1	75045	75045	75045	0.00
Cus 7	1	75044	75044	75044	0.00
Cus 8	1	75044	75044	75044	0.00
Cus 9	1	75044	75044	75044	0.00
Cus 10	1	75044	75044	75044	0.00
TOTAL	10	75044	75044	75045	0.49

Fig. 9: Example result testing by jMeter

This Voice-otp API has also been tested by entering an incorrect parameter. The result is this voice-otp API will display an error message. Testing also includes sending OTP simultaneously. Testing is done by sending OTPs to 10, 20, until 30 different numbers together. The result is that OTP sending is proceeding properly, that all phones can receive right OTP. This test uses the jMeter application (example in Figure 9).

## V. Conclusion

Based on the analysis, design and implementation of the voice-otp API it can be concluded that:

- a. Sending OTP through voice media from SIP to PSTN can be done. To implement this, the OTP code in the form of text or numbers must first be converted into audio. Audio that has been created is then sent together with the invite command in SIP.
- b. Sending OTP simultaneously at the same time can be done as long as the object (ip server, sip port) proxy server must be the same.
- c. Sending OTP simultaneously must be monitored so that race conditions do not occur. This is because the proxy server can usually only handle requests at the same time with limited resources. To handle this race condition can be done by queuing and synchronization mechanisms.

## REFERENCES

- [1] N. Haller and R. Atkinson, "On Internet Authentication," Oct. 1994.
- [2] J. Rosenberg and H. Schulzrinne, "RFC 3261: Session Initiation Protocol (SIP)," *IETF RFC*, vol. 3261, pp. 1–269, 2002.
- [3] N. Anusha, A. D. Sai, and B. Srikar, "Locker security system using facial recognition and One Time Password (OTP)," *Proc. 2017 Int. Conf. Wirel. Commun. Signal Process. Networking, WiSPNET 2017*, vol. 2018-January, pp. 812–815, 2018.
- [4] E. P. Nugroho, R. R. Judhie Putra, and I. M. Ramadhan, "SMS authentication code generated by Advance Encryption Standard (AES) 256 bits modification algorithm and One time Password (OTP) to activate new applicant account," *Proceeding - 2016 2nd Int. Conf. Sci. Inf. Technol. ICSITech 2016 Inf. Sci. Green Soc. Environ.*, pp. 175–180, 2017.
- [5] M. Collin, B. Ravishankar, P. Stewin, and J. Seifert, "SMS-Based One-Time Passwords : Attacks and Defense," pp. 150–159, 2013.
- [6] Y. Zhang, "SIP-based VoIP network and its interworking with the PSTN," *Electron. Commun. Eng. J.*, vol. 14, no. 6, pp. 273–282, 2002.
- [7] P. Gupta, N. Agrawal, and M. A. Qadeer, "GSM and PSTN gateway for asterisk EPBX," *IFIP Int. Conf. Wirel. Opt. Commun. Networks, WOCN*, 2013.
- [8] A. Dorais-Joncas, W. Elleuch, and A. C. Houle, "A conference mechanism for the simultaneous transmission of voice and medical information using SIP," *Can. Conf. Electr. Comput. Eng.*, no. May, pp. 1834–1837, 2006.
- [9] D. A. Huffman, "The Synthesis of Sequential Switching Circuits," Research Laboratory of Electronics, Massachusetts Institute of Technology, 1954.
- [10] Prakash, G., Darbandi, M., Gafar, N., Jabarullah, N.H., & Jalali, M.R. (2019) A New Design of 2-Bit Universal Shift Register Using Rotated Majority Gate Based on Quantum-Dot Cellular Automata Technology, *International Journal of Theoretical Physics*, <https://doi.org/10.1007/s10773-019-04181-w>.
- [11] Y. Martineau, "Peers." 2014.
- [12] Harymawan, I., Nasih, M., Salsabilla, A., Putra, F.K.G. 2020. External assurance on sustainability report disclosure and firm value: evidence from Indonesia and Malaysia. *Entrepreneurship and Sustainability Issues*, 7(3), 1500-1512. [https://doi.org/10.9770/jesi.2020.7.3\(5\)](https://doi.org/10.9770/jesi.2020.7.3(5))
- [13] Mazurina, T.Y., Matkovskaya, Y.S., Neopulo, K.L., Rogulenko, T.M. 2020. Studying the impact of the depreciation policy on the development of innovation potential of industrial enterprises. *Entrepreneurship and Sustainability Issues*, 7(3), 1513-1526. [https://doi.org/10.9770/jesi.2020.7.3\(6\)](https://doi.org/10.9770/jesi.2020.7.3(6))