Securing Telecommunication based on Speaker Voice as the Public Key

Monther Rateb Enayah and Azman Samsudin,

Universiti Sains Malaysia, Penang, Malaysia

Summary
This paper proposes a technique to generate a public cryptographic key from user's voice while speaking over a handheld device. Making use of the human intelligence to identify/authenticate the voice of the speaker and therefore use the voice as the public key. The generated public key is used to encrypt of the transferred data over the open communication channel. The implementation of such a system on mobile phones resist any eavesdrop on phone calls, even from the service provider itself. The proposed protocol also eliminates the need for a trusted third party. This work first analyzes the impact of using RSA and Diffie-Hellman as public key cryptographic methods with RC4 stream cipher as the proposed protocol. Then, the processing steps for the speaker's signal which is used to produce the public key. Finally the study proposed the use of RSA, Diffie-Hellman and RC4 algorithms in the proposed protocol to secure the communication between two mobile phone users.

Key words:
Cryptography, Secure Telecommunication, Voice Recognition.

1. Introduction

Most mobile phones were poor in security. One of the problems with these models is scanning, which means that third parties in the local area could intercept and eavesdrop in phone calls. This is especially true in analogue mobile phones were it is easy to eavesdrop by using radio scanners. More recent digital systems such as Global System for Mobile Communications (GSM) have tried to resolve these fundamental issues; however security problems still continue to persist. GSM uses various cryptographic algorithms for communication security such as A5/1 and A5/2. The A5/1 and A5/2 stream ciphers are used to provide over-the-air voice privacy in cellular telephone standard. A5/1 was developed first and used within Europe and the United States. A5/2 is weaker than A5/1 which is used in countries that may not be able to support the infrastructure necessary for A5/1. Both algorithms have been recently cracked [1].

This paper proposes a new solution to secure mobile phone conversation by doing the encryption and decryption process on the caller and the calle sides, regardless of the cryptographic algorithm used by the local network. By using this approach the user can avoid the attacker from listening or intercepting to the voice calls. In this paper RSA, Diffie-Hellman (DH) and RC4 cryptosystems will be used to ensure the security of the communication channel. The keys here are divided into public key and private key. Public key is generated from the speaker voice and the corresponding private key will be considered as the DH private key. A shared secret will be calculated to generate the input key for the RC4. RC4 algorithm will generate a key-stream to complete the encryption and decryption process.

Using speaker speech to generate the public key and further used in encryption is an area of great promise for security applications, the implementation of such a system on mobile phones presents its own unique challenges. These challenges can be identified as environmental conditions, microphone variability and mobile phones computational limitation.

While the task of securing the communication channels for the mobile phones has been a topic of substantial research, much of the work has been centered on securing the local network. This research departs from securing the channel between users by generating the cryptographic public key as they speak. This paper is organized as follows: In Section 2, we analyze the impact of RSA and DH as public key cryptographic methods and RC4 as stream cipher. Then, Section 3 provides an overview of a basic technique for speaker recognition. Next in Section 4, we describe the proposed framework from capturing the speaker utterance, generating the cryptographic keys and securing the communication channel. Finally, in Section 5 we show our results and draws together concluding remarks on the research project with the future work.

2. Cryptography

Public-Key cryptography is the science of using mathematics to encrypt and decrypt information. With cryptography storing or transmitting sensitive information becomes safer across insecure networks like the Internet. Cryptographic techniques are divided into two generic types: Symmetric key and Asymmetric key. Symmetric key is a conventional type of cryptography which is also

Manuscript received March 5, 2007
Manuscript revised March 25, 2007

known as secret key cryptography [2]. The same key is used for encryption and decryption process. Examples of the symmetric key cryptosystems are Data Encryption Standard (DES), Triple DES and Advanced Encryption Standard (AES). Speed of computations in the symmetric key algorithms is an advantage, compared to asymmetric key algorithms. Asymmetric key algorithms which give an alternative way of securing data require a huge amount of time to do the computation for encryption and decryption.

Public key cryptography was introduced by Whitfield Diffie and Martin Hellman in 1975 [3]. The term public key cryptography is a synonym for asymmetric key cryptography. Public keys have two separate keys that are mathematically connected, a public key which encrypts data, and a related private key for decryption. Public key is published to the public while the private key is being kept secretly. Some examples for public key cryptosystems are Elgamal, RSA, DH, and Elliptical Curve Cryptography [2].

2.1 RSA

This is probably the most recognizable asymmetric algorithm. RSA was created by Ron Rivest, Adi Shamir, and Leonard Adleman in 1977 [4]. To date, it is the only asymmetric algorithm in widespread use that is used for private/public key generation and encryption. The operation of the RSA is described with full details in [5].

Two mathematical problems play the important rule for the RSA cryptosystem [6]: the problem of factoring very large numbers, and the RSA problem. The integer factorization problem is the problem of finding a non-trivial factor of a composite almost prime number. When these numbers are very large, it becomes difficult to factorize and till now no efficient algorithm is known to factor these huge almost prime numbers. The RSA problem is simply the task of taking *e*-th roots modulo a composite *n*, trying to get the plaintext *m* such that *me=c mod n*, where the RSA public key is *e*, and *n*. An attacker needs to factor *n* into *p* and *q*, and computes *(p-1)(q-1)* which allows the determination of *d* from *e*.

Key distribution in RSA like other cipher algorithms needs to be secured against man-in-the-middle attack. The attacker can give a false identity to both sides, if the attacker intercepts the transmissions between the caller and the calle. None of the parties will be able to detect the attacker presence. Defenses against such attacks are often based on digital certificates or other components of a public key infrastructure.

2.2 Diffie-Hellman Key Exchange

The concept of DH key exchange is commonly known as DH. DH represents the last names of the inventors Whitfield Diffie and Martin Hellman. The method was introduced in 1976, and it was the first practical method for agreeing on a shared secret key based on a secure key-exchange protocol over an unsecured communications channel. DH is not an encryption method rather than it is a key exchange protocol. In [6] full details on how the DH concept works is presented.

DH generates a secret number just for one transaction. This is called a session key or a symmetric key. As mentioned before, all asymmetric key systems are considered slow. If little amount of data is being exchanged, the shared secret may be used to encrypt the actual data. But when a huge amount of data is to be passed between both sides, just like in case of phone conversation, encryption requires a stream cipher system such as A5/1, A5/2, FISH, SEAL or RC4. RC4 is the mostly used stream cipher in such applications.

The security of the DH cryptosystem depends on the discrete logarithm problem. The protocol assumes that it is computationally infeasible to calculate the shared secret key, $K = g^{X_A X_B} \bmod n$, iven the two public values $\left(g^{X_A} \bmod n\right)$ and $\left(g^{X_B} \bmod n\right)$ where n is a sufficiently large prime.

Breaking DH protocol is equivalent to calculate discrete logarithms under certain assumptions as what Maurer has showed in [7]. The DH key exchange is vulnerable to a man-in-the-middle attack [6]. In this attack, an attacker is placed between Alice and Bob. Alice and Bob will be used as a conventional terms referring to common characters used in cryptography field. The attacker fools Bob by sending his public key to Bob instead of Alice public key. Bob will transmit his public key. Whereby, the attacker will change it with his public key and sends it to Alice. At this stage, the attacker and Alice will agree on a secret key. On the other hand, the attacker and Bob will agree on another secret key. After this exchange, the attacker simply decrypts any messages sent out by Alice or Bob, and then the attacker is able to read, insert and modify them before encrypting again with the other party public key. This vulnerability is present because DH does not authenticate the participants.

2.3 RC4 Stream Cipher

RC4 is the most widely used stream cipher designed in 1987 by Ron Rivest for RSA Security [6]. It is a variable key size stream cipher with byte-oriented operations. The algorithm is based on the use of random permutations. RC4 is a very fast stream cipher and it is used in the SSL/TLS (Secure Socket Layer/Transport Layer Security) standards that used in the WEP (Wired Equivalent Privacy) protocol. Full detail on how RC4 works can be found in [6]. The encryption is done by applying XOR operation on a byte of the plaintext with one byte of the key-stream. Decryption process is the same

to the encryption process but the same key-stream byte is XORed with the ciphertext instead.

Many approaches tried to attack RC4 but none of these approaches is practical against RC4 with large key length such as 128 bits or more [8]. Therefore, up to now RC4 is considered as a secure stream cipher.

3. Speech Recognition

Biometrics uses biological information to verify the identity of a person [9]. Biometric recognition methods include: fingerprint scan, retina scan, face scan, and voice recognition. The choice of voice recognition against others is made because most of biometric techniques need complex equipments, and also some of these techniques need the physical presence of the person such as fingerprint and retina scanning. However, voice recognition can be done remotely like in the case of phones, giving more flexibility when dealing with the phone.

In order to understand how speaker recognition works, we need to understand how voice is produced [9]. Voice is simply created when air passes the larynx or other parts of the vocal tract. The vibration of the larynx creates an acoustic wave, especially the hum sound. This wave is modified by the motion of the palate, tongue and lips. There are also other sounds that are created by other parts of the vocal tract. The unique voice patterns which are produced by individuals depend on two factors: physiological and behavioral characteristics.

The digitizing process of the human voice begins from the voice produced by the human. The voice is an analogue signal. Analogue signal means continuous values within a time range. The analogue signal is converted into an electrical wave or digital signal, by using devices such as microphone. Next, the continuous digital signals are converted into a discrete voltage values. This process is called sampling. Sampling measures the voltage of the signal at regular time intervals.

At the highest level, all speaker recognition systems contain two main modules: Template Matching and Feature Extraction [10]. Template Matching is the simpler and the most accurate on some cases. It works by comparing the digitized voice with the digitized template based on the amplitude of the voice signal over many frequencies at various times over the entire period of the identification process. But Template Matching does not distinguish between the speech and the background noise. So if the registration is done with noise, the recognition process must be done with the same background noise again.

Feature Extraction does not really use any characteristics of the speech; it takes the digitized signal and applies it to mathematical techniques to produce the results. These results do not describe the voice in physical terms but they can be used to identify the speaker voice. Feature Extraction is much better to identify the speaker than Template Matching with weak signal strength and background noise, due to the mathematical techniques which isolates the mathematical features from the speech. Hence Feature Extraction is preferred more than Template Matching in comparing voiceprints, and it is implemented in the majority of voice identification systems.

3.1 Speech Feature Extraction

The aim of this module is to take the speech waveform and convert it to some type of parametric representation for further analysis and processing. This process is called the signal-processing front end. A variety of possibilities can be chosen to perform Feature Extraction on the speech signal to recognize a speaker, such as Linear Prediction Coding (LPC) [11], Mel-Frequency Cepstrum Coefficients (MFCC) [11], and others. MFCC is the best known and widely used.

3.2 Mel-frequency Cepstrum Coefficients

MFCC main purpose is to perform the same functionality of the human ears [11]. MFCC processor structure is illustrated in Fig. 1 using block diagram. Each step is discussed in [12]. Typically the speech recording is done at a sampling rate above 10000 Hz. This sampling frequency was chosen to reduce the effects of aliasing in the analog-to-digital conversion. Furthermore, these sampled signals have the ability to capture all frequencies up to 5 kHz, which cover most energy of sounds that are generated by humans.

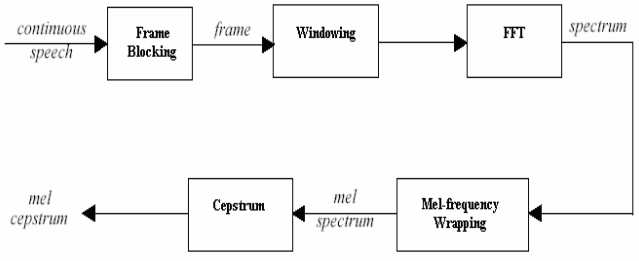

Fig. 1 Block Diagram for MFCC [12].

4. Software Based Voice Encryption Systems

There are many voice encryption systems available. Most of these cryptosystems make PC-to-PC phone calls and they are free for downloading like PGPFone [13], Nautilus [14] and Speak Freely [15]. PGPFone has a user friendly

interface and uses a selection of encryption schemes, such as 128 bit CAST key, 168 bit Triple-DES key or 192 bit Blowfish key. Nautilus depends mainly on DH Key Exchange. Speak Freely uses IDEA or DES. Other cryptosystems like Digital Voice Protection (DVP), STU III (Secure Telephone Unit, Generation III) and STE are hardware based encryption systems and they are not designed for PDA's or advanced cellular phones.

There are cryptosystems that secure cellular telecommunication over the GSM network, such as Snapshield [16] and CryptoPhone [17]. Snapshield has developed Snapcell, a plug-in cellular encryption unit to which it secures end-to-end GSM communications. The problem with Snapcell is that it uses hardware unit which must be installed on the mobile phone and above that, this hardware unit can support only some models of Ericsson and Sony mobile phones. Snapshield uses DH key exchange with AES to secure the channel. Snapshield did not publish the blue print of their design, in which there might be back doors for their algorithm. Furthermore, the attached unit also shortens the mobile phone battery life time.

On the other hand, CryptoPhone was designed by GSMK CryptoPhones Company [17]. CryptoPhone takes the advantage of the high processing performance of recently available PDAs and mobile phones to do real-time voice encryption. Unlike other products in the market CryptoPhone gave the details of their inner workings. The problem with this design is that it can be attacked by man-in-the-middle attack as mentioned previously in DH. Attacker can give to both sides his public key and wait them to send their public keys, which enables him to generate two shared secret keys. The attacker can now analyze the transmitted signal and may listen to the conversation. This problem occurs because both sides cannot authenticate each other before start transmitting. CryptoPhone solved this problem by showing six digits key on caller and calle mobile phones. Whereby, both sides need to speak three marked digits from the six digits shown. The caller will speak his three digits and the calle will check if they match with the digits shown on his mobile phone. The calle speak the other three digits and the caller check if they match with the shown digits in order to authenticate each other.

5. Methodology

Equipped with the background described in the previous sections, this section gives an overview for the methodology used to derive a cryptographic key from the user speech. The proposed methodology begins with capturing the speaker's utterance, dividing the voice samples of the utterance into overlapping windows, and deriving the user's key from the cepstrum coefficients using a feature descriptor. This procedure enables the users authenticate each other by hearing each other voice and generating each other public key.

The next goal after this processing is to construct a long enough cryptographic key for each speaker. This key is the speaker's public key. Our proposed method authenticates the user's public key values which makes this methodology immune to man-in-the-middle attack unlike today's available products as have been discussed before. This methodology further discuss the generation of the user's private key and the generation of the session key which is used as an input for producing a keystream to encrypt and decrypt information in real-time. Generating the private key is done iteratively in this proposed methodology until finding a suitable private key to be used. The following subsections cover in detail the description of these steps.

5.1 Speech Processing

Our proposed methodology begins with capturing the speaker utterance and turns it into a sequence of MFCC (acoustic vectors) using MFCC speech processing steps. Continuous speech signal is blocked into frames of N samples, with adjacent frames being separated by M ($M < N$). The number of samples taken per frame is 256 (N = 256) which is appropriate number to avoid aliasing. The distance between frames is 100 (M = 100). After cutting the speech signal into frames with overlap, the outcome is a matrix where each column is a frame of N samples from original speech signal. Next we employ Windowing and FFT processing steps to transform the signal from time domain into the frequency domain. The result is called the single's power spectrum. These processes together can be referred as Windowed Fourier Transform (WFT). Finally, the power spectrum resulted from WFT process is converted into mel-frequency cepstrum coefficients after using one filter bank for each desired mel-frequency component.

5.2 Mapping Frames to Features

Having derived the acoustic vectors with N frames, the main target now is to define features of these frames that are exactly the same when the same user speaks the same utterance. From these features, an m-bit feature descriptor is then derived. The approach introduced here isolates one feature in each data vector and generates one descriptor bit from each feature, so each data vector can be used as one bit, $N = m$, even though this may not be necessary; because multiple data vectors could be used to derive one bit, so that $N > m$.

The feature used to generate the feature descriptor b from the data vectors $V(1)$ ... $V(N)$ depends on the

amplitude values of the data vectors. In this approach the *i*-th feature $\phi_i = 0$ if the amplitude value is negative, $\phi_i = 1$ otherwise. To map these features to a feature descriptor, simply we need to test whether each feature is positive or negative. See Eq.(1)

$$b(i) = \begin{cases} 0 & if\ amplitude\ value < 0 \\ 1 & Otherwise \end{cases} \quad (1)$$

The value *b(i)* represents the position relative to the origin plane, so the value of 0 can be interpreted as the data vector value falls under the origin plane, while *b(i)* indicates that data vector falls on the plane itself or the positive side of the plane. At the end, the complete *b(i)* represents the public key in binary digits; it is recommended to have a very long key. If the public key was not a prime number the next prime number will be taken as the public key. This method has the flexibility to choose a variable key length. There are many several features that can be implemented to generate the feature descriptor.

5.3 Generating Keys, Encryption and Decryption Flow

At this stage we have derived the speaker's public key noted by *b(i)*. This public key is further used to generate the corresponding RSA private key. After that, the RSA private key will be used as the DH private key. The corresponding DH public key will be calculated and exchanged in a secure manner. The DH public key will be encrypted using RSA algorithm and transferred to the other side. At this step, each part has the other's DH public key after decrypting it using each one's RSA private key.

Now both users can compute the secret key by completing DH key exchange algorithm. The result is the same secret key for both sides; this secret key is used only for one communication session. Next, having the shared secret key in hand, the key will be used as the key for the RC4 stream cipher algorithm. RC4 stream cipher will start to produce a keystream; the keystream will be the same for both sides. This keystream will be used for doing two operations; encrypting the speaker voice and messages, and decrypting the received information from the other speaker. The process of generating these keys is next discussed in detail. Fig. 2 illustrates the encryption flow.

Two large random and distinct primes *p* and *q* needed to be generated for each session; compute *(n = pq)*, the Euler totient function *φ(n) = (p - 1)(q - 1)* and selecting a random integer *e, 1 < e < φ(n)*, such that *gcd(e, φ(n)) = 1*.

We notice that *e* which is the public key is already known and is generated in a separate manner from generating *φ(n)*. To generate the RSA private key the following steps are performed:

(i) Generate two large random and distinct primes *p* and *q*, each roughly the same size.
(ii) Compute *(n = pq)* and the totient function *φ(n) = (p - 1)(q - 1)*
(iii) Check the public key *e* with *φ(n)* such that *gcd(e, φ) = 1*. If this condition is achieved then move on to step 4, else go back to step 1.
(iv) Use the extended Euclidean algorithm to compute the unique integer *d, 1 < d < φ(n)*, such that *ed ≡ 1 (mod φ(n))*.
(v) Share *(n, e)*, and keep private key *d*.

This process is secure against any attacks on private keys; because the key space *φ(n)* and the private key *d* will be changed for each session, so it is hard to factorize *φ(n)* or even trying to figure out *d*.

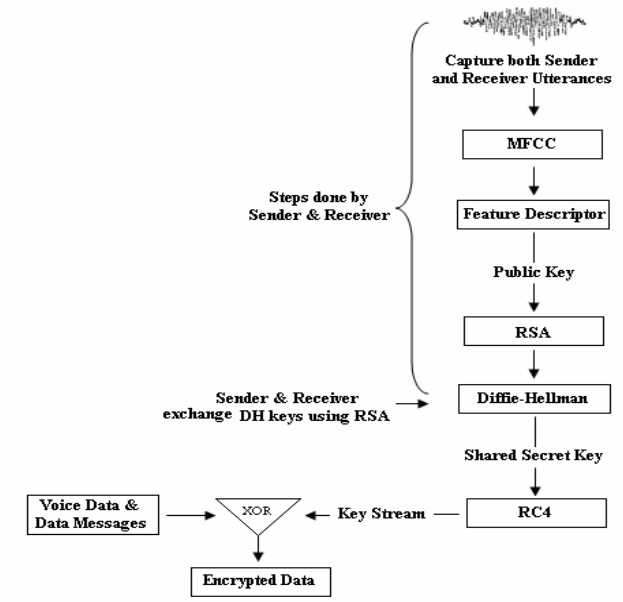

Fig. 2 Encryption Flow.

Next, the RSA private key obtained above will be considered as the DH private key and another process will start to generate a secret key for both users as follows:

(i) The two global integers *n* and *g* are fixed and known by each user.
(ii) The user's DH private key here is the RSA private key which has been already computed. So Alice's private key is $X_A = d_A$, while Bob's is $X_B = d_B$.
(iii) From *n*, *g* and *X*, each part computes his own DH

public key, $Y_A = g^{X_A} \bmod n$ for Alice and $Y_B = g^{X_B} \bmod n$ for Bob.

(iv) Both users exchange their DH public keys after encrypting them with RSA public keys.

(v) Inorder to get the secret key or session key, each users decrypt the DH public key. Here, Alice will compute $K = Y_B{}^{X_A} \bmod n$ to get the secret key and Bob will compute $K = Y_A{}^{X_B} \bmod n$ to get the same secret key.

Finally, after setting up the secret keys by both sides, each user uses the secret key as the RC4 input key and starts generating a keystream. This keystream is used to perform encryption by simply performing XOR operation for the out going data, and the same keystream will be used to decrypt the incoming data for each side by doing XOR operation. Decryption is simply done by XORing the keystream with the encrypted data stream to get the original data stream as illustrated in Fig. 3. XOR operation is not computationally expensive for today's mobile phones. Now the communication channel is secured, even if communication gets tapped, what an eavesdropper gets is nonsense.

The audio stream can be applied to pre-processing steps before doing encryption. That is encode the audio stream before doing encryption and decodes the audio stream after decryption to get the original audio stream. This process analogically like putting the audio streams in a secure envelops.

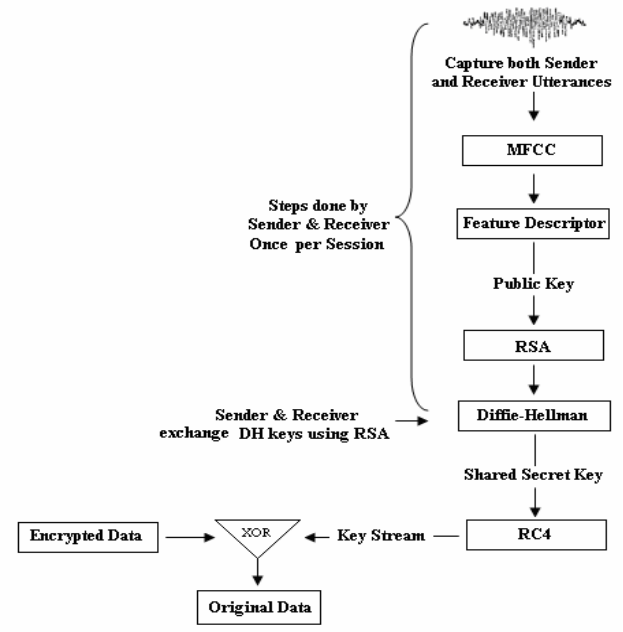

Fig. 3 Decryption Flow.

6. Results

This section presents the results for the proposed methodology adopted in Section 5. This section also discusses the obtained results from implementing the system. In order to implement such a system, one must go through several steps which were described in details in previous sections. The implementation for this simulated project is written using MATLAB. In addition, Maple 8 kernel access was used in programming.

Some tests have been done to measure in which degree the final result can work under personal computers and somewhat mobile phones (e.g. PC-to-PC, Mobile-to-Mobile and Mobile-to-PC). These tests were performed on the following specifications: Intel(R) Pentium(R) M processor 1.73GHz, 512 MB RAM of memory, Microsoft Windows XP Home Edition and programming language was MATLAB.

Testing was also done on the same PC but with different processor speed. The processor speed has been reduced to reach 412 MHz which is near to or less than recent mobile phones. Some of the available mobile phones are currently: Dell Axim X30 powered with Intel 624MHz XScale processor, with 64MB RAM and 64MB ROM. Samsung ARM9-based S3C2440, is clocked at 533MHz.

In this project, a standard microphone connected with the PC is used as the hardware for speech recording. The standard sound recorder program in Microsoft Windows is used to do speech recording. The audio format settings are PCM 44.100 kHz, 16 bit and stereo. This is even more than what GSM uses now which is 8000 kHz, 8 bit mono; that means less computation time needed than PCM 44.100 kHz, 16 bit and stereo. The speaker speech is recorded and saved in wave format.

The timing results for this cryptosystem are computed and presented in the same order of execution. First of all, Key setup starting from the MFCC processing. Table 1 shows the result of executing MFCC algorithm for different audio speech lengths to get 64 acoustic vectors. The key generated from these acoustic vectors is 64-bit length. Note that these audio files were stored on the hard drive and MFCC function needs the same execution time for a variable audio file length.

Table 1 also shows the storage required to save the audio files. On the other hand, Table II shows the timing results for executing MFCC algorithm for different acoustic vector sizes. A larger vector size needs more computation time. A 64 acoustic vector can generate from 64-bit public key up to 2048-bit public key.

Table 1: Time in seconds to generate 64 Acoustic Vectors from MFCC for different audio length

Audio length in seconds	Time for MFCC on P 1.7 GHz	Time for MFCC on P 412 MHz	Required Storage
1	0.0156	0.0313	180 KB
2	0.0156	0.0313	356 KB
3	0.0156	0.0313	527 KB

Table 2: Time in seconds to generate variable key sizes from MFCC

Key size	Time for MFCC on P 1.7 GHz	Time for MFCC on P 412 MHz
64-bit	0.0156	0.0313
128-bit	0.0311	0.1563
256-bit	0.3281	1.1094
512-bit	1.1250	3.5156

Table 2 shows different key sizes after applying the feature descriptor on the acoustic vectors resulted from MFCC process. The process for generating RSA private key and checking the primality of the public key is placed next.

Table 3: Average time needed to generate RSA keys in seconds

RSA key size	Time to prepare RSA keys on P 1.7 GHz	Time to prepare RSA keys on P 412 MHz
64-bit	0.21875	0.515625
128-bit	0.40625	1.40625
256-bit	0.53125	2.359375
512-bit	0.78125	2.765625
1024-bit	1.328125	6.09375

Table 3 shows average timing results for generating RSA keys for variable key sizes. The average time is done using ten tests for each key size.

From Table 3, it is obvious that using 64-bit long RSA keys is very fast. In the sense of security, a 64-bit key is secure enough to transfer public DH keys. After RSA keys have been computed, DH key exchange comes next. DH function first computes the corresponding private key. Secondly, it computes the second user public key. Finally, exchange the key using RSA and computes secret shared key. Table 4 shows the timing result to generate the secret key.

Tabele 4: Average time needed to generate DH secret key in seconds

DH Secret key size	Time to prepare DH key on P 1.7 GHz	Time to prepare DH key on P 412 MHz
64-bit	0.015625	0.21875
128-bit	0.0625	0.3125
256-bit	0.09375	0.4375
512-bit	0.15625	0.828125
1024-bit	0.28125	1.53437

When the secret key is known, the key is used as RC4 key. Table 5 shows the time required to initialize RC4.

Table 5: Time needed to initialize RC4

Time in seconds on P 1.7 GHz	Time in seconds on P 412 MHz
0.0006	0.0056

Finally, the encryption process will start. Encryption tests were done and the timing results were taken for either doing XOR operation alone, or encoding and then doing XOR operation over bytes of data. Timing tests includes the time needed to generate the proper keystream and perform an XOR operation. The encoder used here is AN643 Adaptive differential pulse code modulation (ADPCM). Table 6 shows the timing results for generating keystream and performing encryption or decryption process. Each byte of data needs one byte of keystream. Note that the time needed to decrypt one byte is the same as the time needed to encrypt one byte using either simple XOR operation or doing XOR operation and applying decoder after that.

Table 6: Time needed to initialize RC4

Operation	Time in seconds on P 1.7 GHz	Time in seconds on P 412 MHz
XOR (1-byte)	0.00001	0.00003
Encode & XOR (1-byte)	0.00007	0.00015
XOR (2-byte)	0.00002	0.00006
Encode & XOR (2-byte)	0.00015	0.0003

From these tests, the overall time needed to setup the system depends on the time needed to take the MFCC sample and the generation of keys, taking into consideration their lengths. Table 7 shows the total time needed to start the system. Note the MFCC audio length is not included.

Table 7: Time needed to Setup the System

Key size	Time needed on P 1.7 GHz	Time needed on P 412 MHz
64-bit	0.266175	0.802575
128-bit	0.51605	1.91195
256-bit	0.9693	3.943175
512-bit	2.0787	7.14625
1024-bit	2.950575	12.58062

These results show that a 64-bit system requires less than one second to prepare the keys after storing the MFCC audio sample. A 128-bit also requires less than one second for normal PC and less than two seconds on the mobile phone simulated processor. The drawback of this protocol is that it needs a few seconds to take the user's voice and setup the cryptographic keys, but this can be solved by encrypting the first seconds with a fixed key known between users until the system initializes. Note that the results obtained here using are 44,100 kHz PCM with 16 bit stereo wave format. Meanwhile, the GSM uses much less wave quality format which can reduce the system setup time to the half of these results, meaning that recent mobile phones can run the system in a much fast manner.

7. Conclusion

This paper discusses the development of a system combining audio Feature Extraction and public key cryptography. For Feature Extraction, Mel-Frequency Cepstrum Coefficients was selected due to the fast process, the quality of the results and it is applicable to any spoken word. MFCC performs these signal processing steps: Frame blocking, Windowing, Fast Fourier Transform, Cepstrum and Mel-frequency wrapping. In this paper, in order to perform authentication, MFCC was implemented to authenticate others by generating a key from the user's voice.

Both RSA and DH are used to generate the secret shared key. This key is further used as an RC4 key. RC4 is a fast algorithm to generate keystream and it is suitable for real-time applications. RSA algorithm generates public keys which are used to secure the exchange process of DH public keys. It is secure to exchange DH public keys without encryption but exchanging them this way gives additional security for the proposed method. Overall, the results for this project are as follows:

(i) Key authentication is done implicitly and automatically using MFCC feature extraction.
(ii) The generated key is secure against the man-in-the-middle attack.
(iii) The protocol is based on RSA, DH and RC4 cryptosystems.
(iv) Using pseudo-random numbers for the private keys and the session keys. Also using pseudorandom number generator for the key streams. These keys are non-deterministic.

The time required to setup the keys is fast enough for both PC and mobile phone usage. Also the protocol requires a very small storage.

References

[1] Sara Robinson. 2000. Cell phone flaw opens security hole. ZDnet News http://www.snapshield.com/Articles_helpful/Cell_phone_flaw.htm (Accessed March 13, 2006).
[2] Chey Cobb. 2004. Cryptography for Dummies. For Dummies: Hungry Minds Inc,U.S.
[3] W. Diffie and M. E. Hellman. 1979. Privacy and authentication: An introduction to cryptography. Proc. of the IEEE, 67(3):397--427.
[4] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. 1997. Handbook of applied cryptography. CRC Press.
[5] Wikipedia. 2006. RSA. Public-key encryption. Wikipedia. http://en.wikipedia.org/wiki/RSA (Accessed March 13, 2006).
[6] William Stallings. 2003. Cryptography and Network Security: Principles and Practice, Third edition. U.S. Prentice Hall.
[7] U. Maurer. 1994. Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms, Advances in Cryptology Crypto '94, 271-281.
[8] Wikipedia. 2006. RC4. RC4 stream cipher. Wikipedia. http://en.wikipedia.org/wiki/RC4 (Accessed March 13, 2006).
[9] Dualta Currie. 2003. Shedding some light on Voice Authentication.GSEC-V1.4b. http://www.sans.org/rr/whitepapers/authentication/847.php.
[10] Jim Baumann. 1993. Voice Recognition, Human Interface Technology Laboratory, University of Washington. http://www.hitl.washington.edu/scivw/EVE/I.D.2.d.VoiceRecognition.html (Accessed January 16, 2006).
[11] Davis, S.B Mermelstein.1980. Comparison of parametric representations for monosyllabic wired recognition in continuously spoken sentences. IEEE Transactions on Acoustic, Speech, Signal Processing, Vol. ASSP-28, No.4.
[12] Minh N. Do. 1998. An Automatic Speaker Recognition System. Swiss Federal Institute of Technology. http://lcavwww.epfl.ch/~minhdo/asr_project/asr_project.html (Accessed May 19, 2006).
[13] P. Zimmermann, 1996. PGPfone: Owner's Manual, http://www.pgpi.com
[14] B. Dorsey et al., 1996. Nautilus Documentation, http://www.lila.com/nautilus/
[15] J. Walker, B. C. Wiles, 1995. Speak Freely, http://www.fourmilab.ch
[16] Tadiran, 2006. Snap shield, www.snapshield.com

[17] Cryptophone. 2006. Cryptophone Encryption. Cryptophone. http://www.cryptophone.de/background/index.html (Accessed March 13, 2006).

Monther Enayah received the B.S. degree in Computer Science from Applied Science University in Jordan, in 2004, and M.S. degree in Computer science from Universiti Sains Malaysia, in Malaysia in 2006. Currently he is a Ph.D student at the Universiti Sains Malaysia in the area of Security and Cryptography.

Azman Samsudin received the B.S. degree in Interconnection Switching Networks from University of Rochester. He also received M.S. and Ph.D degrees in Parallel Distributed Computing and Cryptography from University of Denver respectively. Currently he is chairperson for Information Systems and deputy dean of Graduate Studies and Research