Extracting Variable-Size Secret Keys from Voice Key

Osama M. Amer, A. S. Obada, Emad Massamir and Tharwat O. Alhanafy*

Computer and System Engineering Department, Al-Azhar University, Cairo, Egypt
*s_ewiss@yahoo.com

Abstract: This paper presents a technique to produce secret keys with optional size from a voice key that can be obtained from the delta modulation. The technique that produces the secret key should be a function of all the bits in the voice key in to get unique keys for each voice key. To achieve this goal, the DES encryption algorithm in cipher block chaining (CBC) mode of operation is used after preprocessing the voice key. Throughout this paper, a step-by-step of the algorithm is introduced to achieve this goal. Therefore, a background about the cipher block chaining mode of operation is presented. Next, the discusses of the block diagram of the proposed technique is done. Then, the deploys of the entropy algorithm as a key evaluation metric is introduced. Finally, two examples are introduced to demonstrate the applicability of this technique.

[Osama M. Amer, A. S. Obada, Emad Massamir and Tharwat O. Alhanafy. **Extracting Variable-Size Secret Keys from Voice Key.** New York Science Journal 2011;4(1):49-60]. (ISSN: 1554-0200). http://www.sciencepub.net/newyork.

Keywords: Extracting; Variable-Size; Secret; Key

1. Introduction

The 'delta' refers to the difference value from one instant to the next, with the difference being limited to -1 or +1. This system maintains an accumulator that starts at zero. At every sample position, this accumulator either steps up by one or steps down by one, never maintaining the same value. Stepping up or down is in response to a comparison between the current accumulator value with the desired waveform amplitude. If the decision is too low, then it steps up by one. Otherwise it steps down. This is depicted in Figure 1 and it should be noted that this method would typically have a far higher sample rate than PCM audio – but then only require at each sample instant a 1 or 0 for up or down, respectively. The problem with delta modulation is that the quantization error depends on the step-size[1].

This means that in order to represent audio as accurately as possible, the step height should be small. However a small step means that more steps are needed to reach-up to larger waveform peaks. In Figure 1, when rising up to the first peak, and dropping down after it, there is a large gap between the waveform we desire to quantize and the actual step values. This is because delta modulation can only increase a single step at a time, but the gradient of the waveform has exceeded this. Such a limit on the gradient is termed the slew rate or sometimes slope overload [2].

The output bit-stream, representing the waveform in the delta modulation format, is also shown in Figure 2. In order to support a higher slew rate without increasing the step-size, it is necessary to sample more quickly [3]. This is a trade-off between bits per second used in the representation and the quantization noise introduced by it.

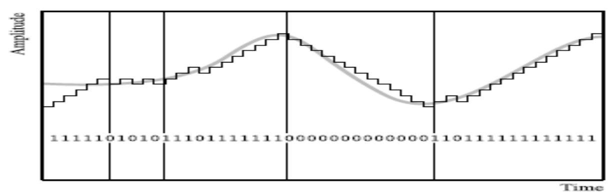


Figure 1: Illustration of an audio waveform being represented using delta modulation, where a 1 indicates a stepwise increase and a 0 indicates a stepwise decrease in amplitude at each sampling position [**8].**

The delta-modulation algorithm is implemented using Matlab and it takes two inputs. The first input is the spectrum associated with the speech of a person and the second input is the step size. The step size is the value by which the accumulator, a variable that is started with a value of zero, is increased or decreased based on the comparison of the value of accumulator and the spectrum magnitude at time t. If the value of the accumulator is greater than the spectrum magnitude, the accumulator is increased by the step size and a 1 is assigned to the corresponding bit in the bitstream. Otherwise, the accumulator is decreased by the step size and a 0 is assigned to the corresponding bit in the bitstream. The algorithm returns the generated bitstrem as an output.

The paper is organized as follows. Sections 1 are this introduction. In Section 2, we present an analysis of Adaptive delta modulations. Section 3 describes the Speech-key Generation Algorithm. Section 4 presents the Demonstration Examples. Section 5 evaluates the CBC Encryption Mode of Operation. And Entropy as Key Metric is introduced in section 6. Finally, our conclusions are drawn in Section 7.

```
function [cn] = deltaModulation (x, StepSize)
xlen = length(x);
accum(1) = 0;
for i =1: xlen
    if (x(i) >= accum(i))
        e_tilda_n (i) = 1;
        accum(i+1) = accum(i) + e_tilda_n (i) * StepSize;
    else
        e_tilda_n(i) = -1;
        accum(i+1) = accum(i) + e_tilda_n (i) * StepSize;
        tx(i)=0;
    end
end
for k = 1 : xlen
    if (e_tilda_n (k) == -1)
        e_tilda_n (k) = 0;
    end
end
cn = e_tilda_n;
```

Figure 2: Delta modulation algorithm.

2. Adaptive delta modulations

In an attempt to maintain the beneficial features and simplicity of delta modulation but to overcome the slew rate limitations, designers came up with several methods to vary the step height based on the past history of the audio [6]. One of those method is to adapt the quantization level so that it can be small when dealing with slowly changing waveforms, but coarse when dealing with rapid waveform changes. The technique is also known by the mouthful *continuously variable slope delta modulation*, abbreviated CVSDM or CSVD, and used as a speech compression method in some older radio systems. In the most basic form, such a system relies upon some rules to change step-size, such as the following artificial example: 'If the past *n* values were the same then double the step height, otherwise halve it.' There would, of course, be upper and lower limits to the step height changes [7]. In reality, some systems would themselves gradually adapt their step-size rules over time. Often step heights gradually increased, and gradually decreased (rather than the system mentioned which doubles it or halves it each time, considered a fairly large change).

The technique is illustrated in Figure 3, for n = 3, and thus the step height can be seen to change following three successive moves in the same direction. Several step height reductions are similarly illustrated. The bit-stream resulting from this process, which is a quantized representation of the original waveform, is shown across the bottom of the plot.

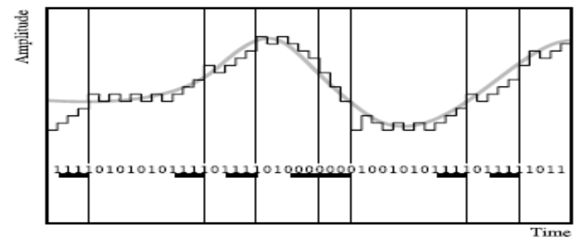


Figure 3: Illustration of an audio waveform being represented using adaptive delta modulation, where a 1 indicates a stepwise increase in signal amplitude and a 0 indicates a stepwise decrease in signal amplitude. Repetition of three like sample values triggers a doubling of step-size, and a halving of step-size is triggered by the neighboring samples which are unlike. The predominant step-size in the figure is the minimum limit, and there would likewise be a maximum step-size limits imposed [**8].**

3. 3 Speech-key Generation Algorithm

The algorithm the implements the speech-key generation model is introduced in this section. The algorithm is depicted in Figure 4. The algorithm is called speech-key-generation. It takes the speech signal as an input and produces the sequence of bits associated with the speech signal as an output. This sequence of bits represents the speech-key.

*Input***:** Speech-signal

*Output***:** Speech-key

Speech-key-generation-algorithm (speech-signal)

1. Choose window length in milliseconds
2. Filter to remove LF recording noise.
3. Compute LPC coefficient with model order n (e.g. n=10).
4. Compute prediction error with all zero filters.
5. Get spectrum from the (LPC) parameters.
6. Represent spectrum curve using delta modulation.

Figure 4: Speech-key generation algorithm.

This algorithm is implemented by using Matlab since it contains ready-made functions that accomplish this task. The Matlab implementation program is shown in Figure 5. Only the main lines in this algorithm are highlighted. The algorithm takes a wave file that has the person speech as an input (Line 1). Then it reads the sound signal from the specified file (Line 3). Next it filter the signal to remove noise (Line 6) and then compute the LPC coefficient (Line 7). After computing the LPC coefficients, it computes the prediction error with all zero filter (Line 8). Next, the spectrum associated with the LPC parameters is generated (Line 10). Finally the delta modulation algorithm in Figure 3.6 is called to generate the bitstream (Line 17).

```
1.  function [ ] = speech-key-generation (fileName )
2.  winlens = 50;  %PSD window length in milliseconds
3.   [y, fs] = wavread(fileName);   %  Read in wavefile
4.  winlen = winlens * fs/1000;
5.  [cb, ca] = butter (5,2 * 100/fs, 'high');
6.  yf = filtfilt (cb, ca, y);  %  Filter to remove LF recording noise
7.   [a, er] = lpc (yf, 10);  %  Compute LPC coefficent with model order 10
8.  predy = filter(a, 1, yf);  %  Compute prediction error with all zero filter
9.  figure(1) ;  plot(predy); title ('Prediction error'); xlabel ('Samples'); ylabel ('Amplitude')
10.  [hz, fz] = freqz (1, a, 1024, fs);   % Get spectrum from the AR (LPC) parameters
11. figure (2)
12. plot (fz, abs (hz))
13. title ('Spectrum Generated by LPCs')
14. xlabel ('Hertz')
15. ylabel ('Amplitude')
16. StepSize = 1/15;
17. cn = deltaModulation (hz, StepSize);  % Delta modulation-demodulation encoder
18. file = fopen('output.txt', 'w');  % write Matlab results to a text file
19. for i=1:1024
20.       fprintf(file, '%d \n', cn(i));
21. end;
22. fclose(file);
```

Figure 5: Implementation of speech-key generation using Matlab.

4. Demonstration Examples

The rest of this paper presents a case study using the Arabic latter ‘alif’ that is pronounced by two different persons: $person_1$ and $person_2$ to show the output of the speech-key generation algorithm for each person. Figure 6a and 6b shows the Prediction error associated with the sound signal of $person_1$ and $person_2$, respectively. Also Figure 7a and 7b shows the spectrum curve generated by LPCs script for $person_1$ and $person_2$, respectively. Finally, the bitstream or the speech key is generated for each person. This is depicted in Figure 8a and 8b for $person_1$ and $person_2$, respectively. This shows that the two keys are different.

5 CBC Encryption Mode of Operation

The *cipher-block chaining* (CBC) is an encryption mode of operation that is used with block-cipher encryption algorithms such as Data Encryption Standard (DES) and Advanced Encryption Standard (AES). In DES encryption In CBC mode, the plaintext which has *n* bits is divided into blocks of 64 bits each, where *n* should be multiple of 64. If not, the plaintext must be padded with the required number of bits to satisfy this condition. Each block is denoted by PTB_i and it is XORed with the previous ciphertext block, CTB_{i-1} before being encrypted. In this way, each CTB_i is dependent on all plaintext blocks processed up to that point (i.e., PTB_1 through PTB_{i-1}). Also, to make each message unique, an initialization vector, *IV*, must be used in the first block. The size of the *IV* is equal to the block size which is, in this case, equal to 64 bits. This process is described in Figure 9.

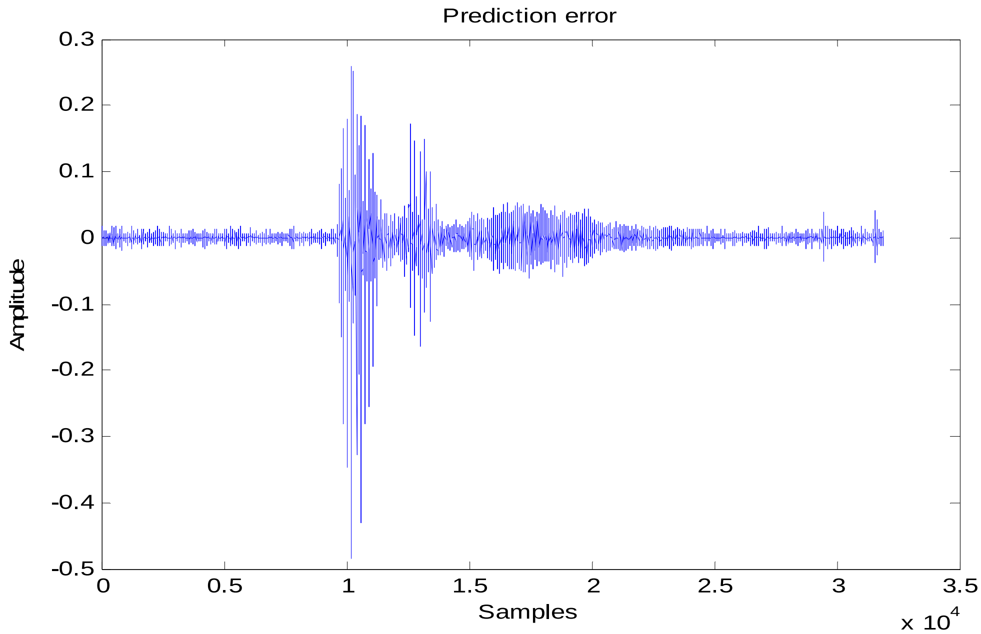
If the first block has index 1, the mathematical formula for CBC encryption is

$$CTB_i = E_k\,(PTB_i \text{ xor } CBT_{i-1}),\ \ CBT_0 = IV$$

while the mathematical formula for CBC decryption is

$$PTB_i = D_k\,(CTB_i) \text{ xor } CBT_{i-1},\ \ CBT_0 = IV$$

In this mode of encryption operation, if a one-bit is changed in a plaintext block PTB_i, all the following ciphertext blocks (i.e., CTB_{i+1} through CBT_N, where N is the number of blocks) will be affected.



(a) Prediction error associated with the sound signal of $person_1$.

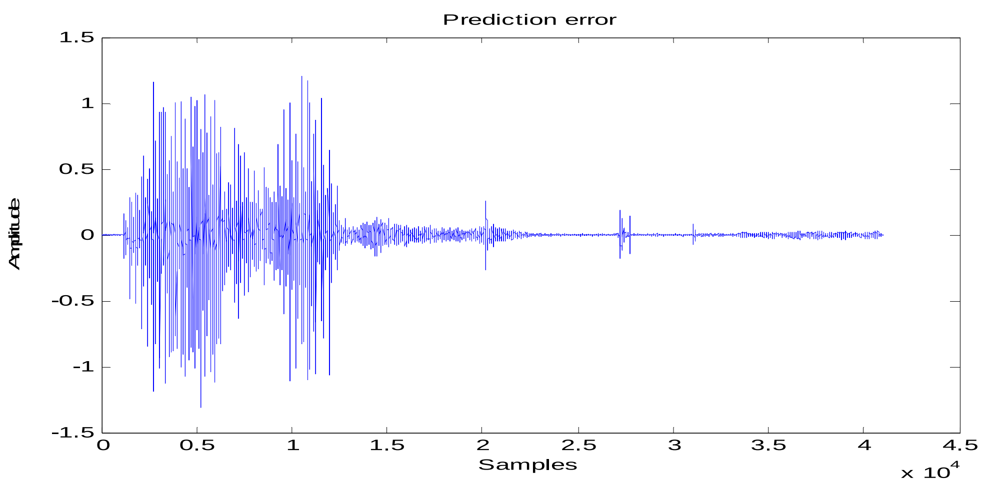
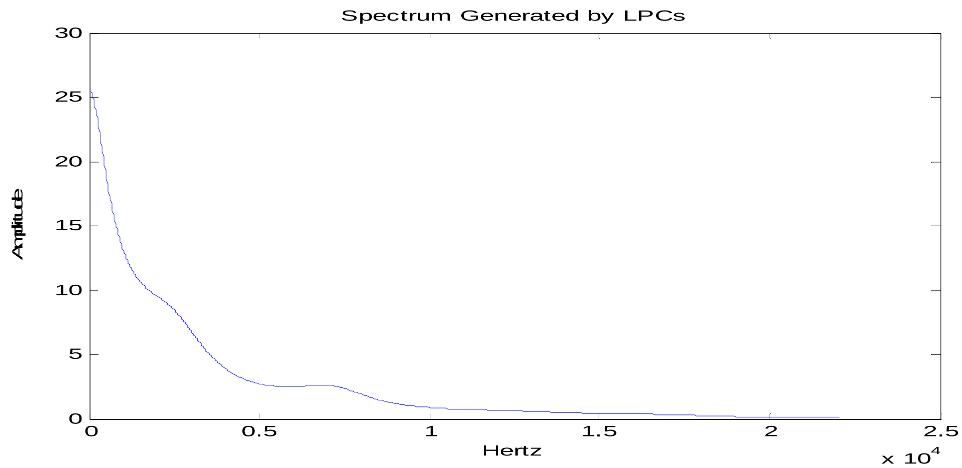


Figure 6: Prediction error associated with sound signals of the two persons.



(a) Spectrum curve corresponding to the sound of $person_1$.

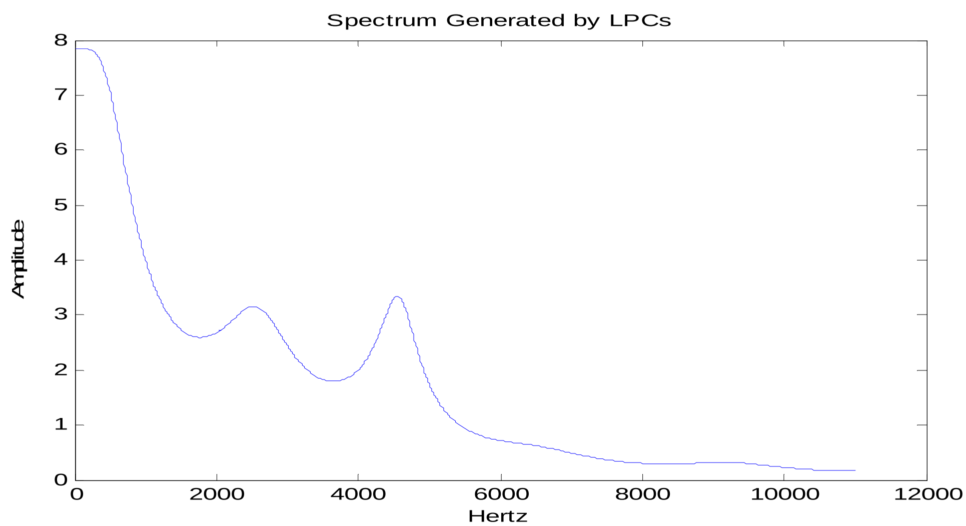


Figure 7: Spectrum curves associated with the sounds of the two persons.

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 1 0 1 0 0 1 0
0 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 0 1 0 1 0
1 1 0 1 0 1 0 1 1 0 1 0 1 1 0 1 0 1 1 0 1 0 1 1 0 1 0 1 1 0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0
1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 1 0 1 0 1 1 0 1 0 1 1 0 1 1 0 1 1
0 1 1 0 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 1 1 1 0 1 1 0 1 0 1 1 0 1 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0
0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 1 0 0 1 0 0 1 0 1 0 1 0 0 1 0 1 0 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
```

(a) Bitstream of $person_1$.

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 0 1 1 1 1 0 1 1 1 0 1 1 1 1 0 1 1 1 0 1 1 0 1 1
0 1 1 0 1 0 1 0 1 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
1 0 1 0 1 0 1 1 0 1 0 1 0 1 1 0 1 1 0 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 0 1 0 1 0 1 0 0 1 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0
1 0 1 0 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 1 0 1 0 1 1 0 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1 1
1 0 1 1 1 1 1 0 1 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 1
0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 0 1
0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 1 0 1 0 1 0 1 1 0 1 0 1 0 1 1 0 1 0 1 0 1
0 1 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1
0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
```

(a) Bitstream of $person_2$.

Figure 8: Generated bitstreams for the two persons based on delta modulation.

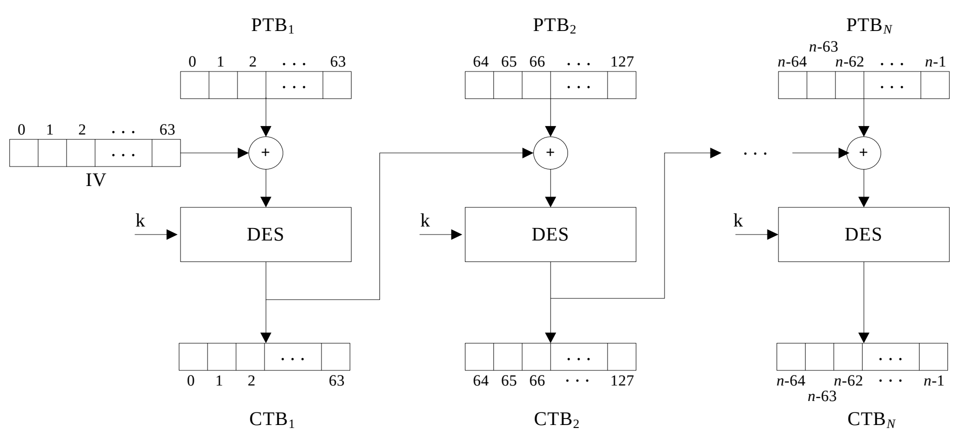


Figure 9: Cipher Block Chaining encryption mode of

5.1 Getting Variable-Size Keys Block Diagram

The block diagram of the proposed technique that is used to deduce variable-sized secret keys from a voice key is described in Figure 10. The block diagram accepts a voice key (*VoiceKey*) and the required key size (*KeySize*) as inputs while it produces the corresponding secret key with the required size as an output. In addition, this block diagram comprises six components to achieve the required secret key. These components are as follow:

1. Key Partition,
2. Padding 56 Bits,
3. Adding Parity,
4. Block Partition,
5. DES Encryption CBC Mode, and
6. Generating Corresponding Key.

The rest of this section describes all these components in more detail to show how the corresponding secret key is produced from the provided voice key.

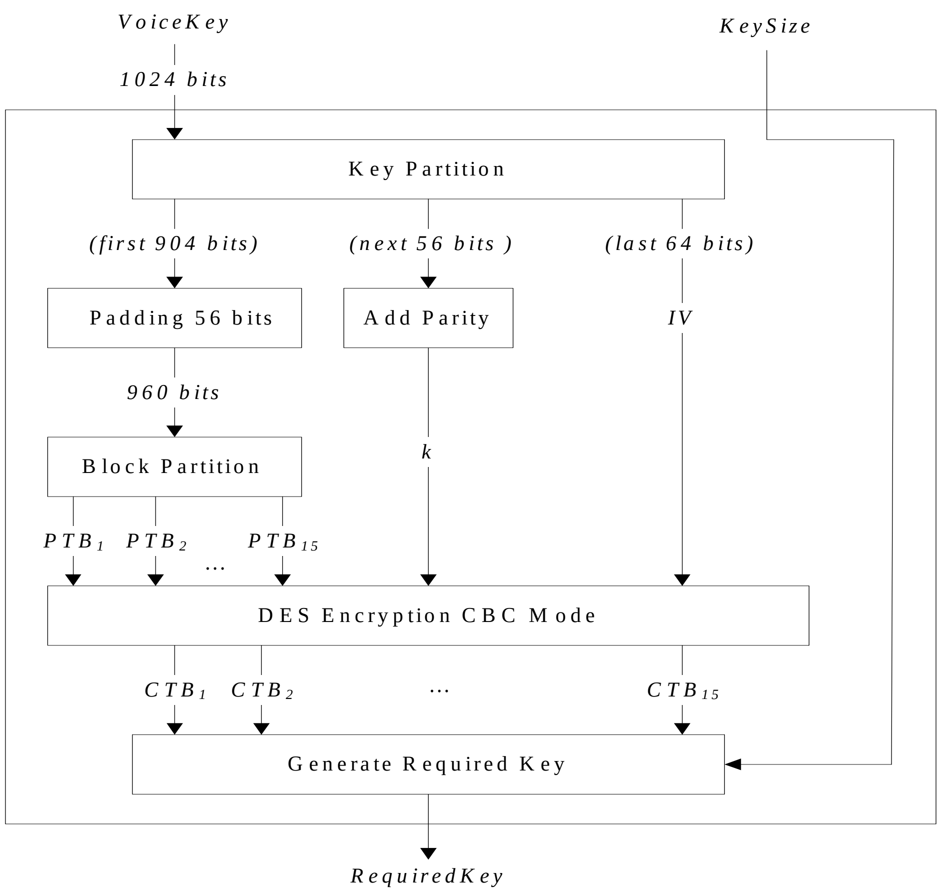


Figure 10: Block diagram of reducing voice-key to a key with optional size.

5.2 Key Partition

The *key partition* component simply divides the received voice key, *voiceKey*, which is 1024 bits into three parts (Figure 10). The first part is equal to the first 904 bits of the *voiceKey*. The second part is equal to the next 56 bits of the *voiceKey* while the third part is equal to the last 64 bits of the *voiceKey*. The total number of bits of the three parts is equal to 1024 bits which is the same size of the *voiceKey*. In other words, the first part includes the bits from 0 to 903, and the second part includes the bits from 904 to 959, and the last part comprises the bits from 960 to 1023 of the *voiceKey*.

5.3 Padding 56 Bits

The *padding 56 bits* component is used to pad the first part produced by the *key partition* component with a series of 56 bits in order to make this part multiple of 64 bits. Therefore, it becomes 960 bits instead of 904 bits as seen in Figure 10. These 960 bits will be presented as input to the *block partition* component.

5. 4 Adding Parity

The key size of the DES algorithm is 64 bits which are equal to eight bytes such that the eighth bit of each byte represents the parity bit. For example, if the first seven bits of a byte has even number of ones, then the eighth bit of that byte must be set with a one in case of odd parity. While in case of even parity, it will become zero. This is why we take only 56 bits from the *voiceKey* and then they are divided into eight group of seven bit each. Next, the parity bit is added to each group that becomes eight bits. Therefore, the DES key becomes 64 bits resulted from the eight bytes corresponding to eight groups. These parity bits are required to cope with the implementation of the DES encryption algorithm.

5.5. Block Partition

The *block partition* component receives a series of bits equal to 960 bits which is the first part produced by the *key partition* component. Since this series of bits is multiple of 64 bits, the underlying component split it into fifteen blocks each with 64 bits. Consequently, this component produces fifteen blocks. Where the first block has the bits from 0 to 63, the next block has the next 64 bits, and so on. Each block is denoted by PTB_i, where i = 1, 2, …, 15. This can be in Figure 10. These fifteen blocks will be used as inputs to the *DES Encryption CBC mode* component.

5.5 DES Encryption In CBC Mode

The *DES encryption CBC mode* component accepts seventeen inputs and produces fifteen outputs. The inputs include:

1. *IV* produced by the *key partition* component,
2. *k* resulted from the *adding parity* component, *and*
3. Fifteen blocks: PTB_1, PTB_2, …, *and* PTB_{15} produces by the *padding 56 bits* components while the outputs include fifteen blocks: CTB_1, CTB_2, …, *and* CTB_{15}. These blocks represent the encryption of the input blocks: PTB_1, PTB_2, *and* PTB_{15}. This can be shown in Figure 11.

Figure 10 comprises fifteen blocks. Each block, say block *i*, uses the DES encryption algorithm and it accepts the *k* and PTB_i XORed with CTB_{i-1} as inputs and produces the CTB_i as an output. This is applicable for all blocks except the first block which accepts k and PTB_1 XORed with *IV* as inputs. The encryption starts from the first block, then the second block, and so on until all the fifteen input blocks are encrypted. From Figure 11, it can be noted that any output block, say CTB_i, is a function in all input blocks from PTB_1 to PTB_i. This means that any change even with one bit will be reflected in the following subsequent blocks. After finishing the encryption of all input blocks, the output blocks are passed to the *generating corresponding* component.

5.6. Generating Corresponding Key

The *generating corresponding key* component takes the fifteen encrypted blocks: CTB_1, CTB_2,…, *and* CTB_{15} produced by the previous component as inputs. Also it accepts the required key size, *keySize*, as an input. It produces the key with the required size as an output. The fifteen encrypted blocks represent 960 bits (Figure 10). This component simply returns the last *keySize* bits of the 960 bits. For example, if the key is equal to 128, the component will return the last 128 bits of the 960 bits of the encrypted blocks.

6 Entropy as Key Metric

The entropy of two voice keys: $voiceKey_1$ and $voiceKey_2$ and their corresponding keys produced by the proposed algorithm are evaluated. The values of these keys are shown in Figure 12 and 13. The entropy of each key is evaluated according to the entropy's formula. Table 1 shows these results. For example, the entropy of the key with 64 bits that produced from $voiceKey_1$ is placed in the cell that crosses the row $voiceKey_1$ and the column "64-bit key." this value is equal to 66.

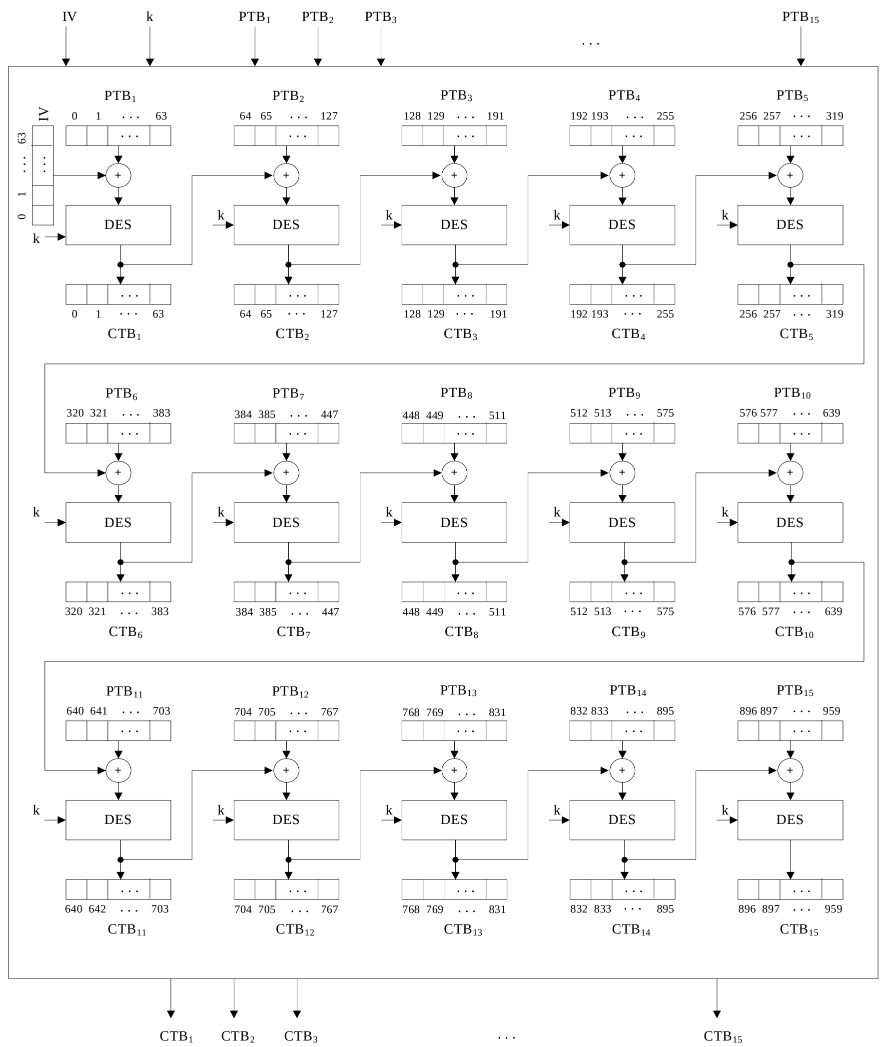


Figure 11: Encrypting fifteen blocks with 64 bits each using DES in CBC mode of operation.

6.1 Demonstration Examples

This section presents two examples to demonstrate the output of the proposed technique. The first example represents the voice key_1, $voiceKey_1$, which is 1024 bits as seen in Figure 12a. The algorithm is executed four times with *keySize* 64, 128, 256, and 512, respectively. In each time, it produces the corresponding key as shown in Figures 12b, 12c, 12d, and 12e, respectively. The $voiceKey_1$ is corresponding to the character "alef".

The second example represents the voice key_2, $voiceKey_2$, which is the same as the $voiceKey_1$ except that we replaced the first bit with "1" as shown in Figure 13a. The algorithm is executed again four times with the *keySize* 64, 128, 256, and 512, respectively. In each time, it produces the corresponding key as shown in Figures 13b, 13c, 13d, and 13e, respectively. It can be seen that even though the two voice keys: $voiceKey_1$ and $voiceKey_2$ are different only in one bit; the produced keys with the same size are different. For example, the keys in Figures 12b and 13b are different and they are resulted from $voiceKey_1$ and $voiceKey_2$, respectively. This means that a produced key is unique for its corresponding voice key.

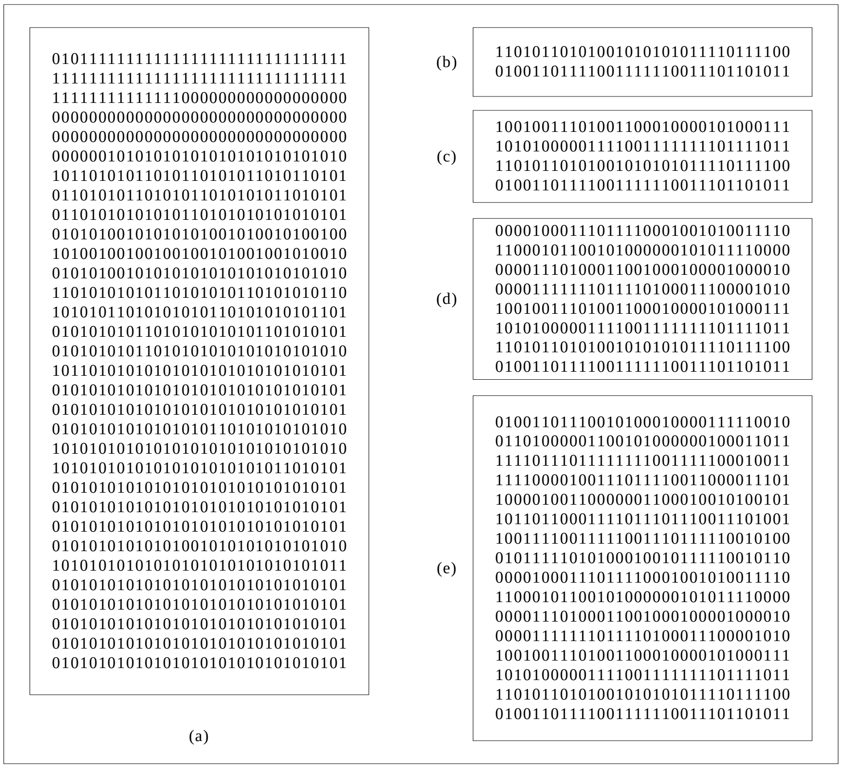


Figure 12: Different keys with different sizes are generated from *voicekey*$_1$ (a) *voicekey*$_1$, (b) 64-bit key, (c) 128-bit key, (d) 256-bit key, and (e) 512-bit key.

Figure 13: Different keys with different sizes are generated from *voicekey*$_2$ (a) *voicekey*$_2$, (b) 64-bit key, (c) 128-bit key, (d) 256-bit key, and (e) 512-bit key.

Table 1: Entropy of voice keys and their extracted keys.

	Entropy				
	Original key	64-bit key	128-bit key	256-bit key	512-bit key
$VoiceKey_1$	0.69	0.66	0.68	0.69	0.69
$VoiceKey_2$	0.69	0.69	0.68	0.69	0.69

7. Conclusions:

The technique to produce secret keys with optional size from a voice key that can be obtained from the delta modulation is presented. The technique that produces the secret key should be a function of all the bits in the voice key in to get unique keys for each voice key is introduced. To achieve this goal, the DES encryption algorithm in cipher block chaining (CBC) mode of operation is used after preprocessing the voice key. Throughout this paper, a step-by-step of the algorithm is introduced to achieve this goal. Therefore, a background about the cipher block chaining mode of operation is presented. Next, the discussion of the block diagram of the proposed technique is done. Then, the deploys of the entropy algorithm as a key evaluation metric is introduced. Finally, two examples are introduced to demonstrate the applicability of this technique.

Corresponding author
Emad Massamir and Tharwat O. Alhanafy
Computer and System Engineering Department, Al-Azhar University, Cairo, Egypt
s_ewiss@yahoo.com

References

1. B. Chen and V. Chandran, "Biometric Based Cryptographic Key Generation from Faces."*In Proceeding of the 9th Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications* (DICTA 2007), Pages 394-401,
2. Glenelg, South Australia, Australia, 3-5 Dec. 2007.Fabian Monrose, Michael K. Reiter, Qi Li, and Susanne Wetzel, "Cryptographic Key Generation from Voice," In Proceedings of the 2001 IEEE Symposium on Security and Privacy, Page 202, May 2001.
3. Christopher Costanzo, "Active Biometric Cryptography: Key Generation Using Feature and Parametric Aggregation," In Proceedings of the Second International Conference on Internet Monitoring and Protection 2007, Page 28, 1-5 July 2007.
4. D. Feldmeier and P. Karn. UNIX password security—Ten yearslater. In *dvancesin Cryptology—CRYPTO '89 Proceedings* (LectureNotes in Computer Science 435), 1990.
5. D. Klein. Foiling the cracker: A survey of, and improvements to,password security. In *Proceedings of the* 2nd *USENIX Security Workshop*,August 1990.
6. R. Morris and K. Thompson. Password security: A case history.*Communications of the ACM*, 22(11):594–597, November 1979.
7. E. Spafford. Observations on reusable password choices. In *Proceedingsof the* 3rd *USENIX Security Symposium*, September 1992.
8. T. Wu. A real-world analysis of Kerberos password security. In *Proceedings of the 1999 Network and Distributed System Security Symposium*,February 1999.
9. Q. Li, B.-H. Juang, C.-H.Lee, Q. Zhou, and F. K. Soong.Recentadvancements in automatic speaker authentication.*IEEE Robotics &Automation*, 6(1):24–34, March 1999.
10. F. Monrose, M. K. Reiter, and S. Wetzel. Password hardening basedon keystroke dynamics. In *Proceedings of the* 6th *ACM Conferenceon Computer and Communications Security*, pages 73–82, November1999.
11. DavideMaltoni, Dario Maio, Anil K. Jain, and SalilPrabhakar. Handbook of Fingerprint Recognition.Springer, 2nd edition, ISBN: 1848822537, May 2009.
12. NSTCsubcommittee on biometrics. Fingerprint Recognition, 2006, found at: http://www.biometrics.gov/Documents/FingerprintRec.pdf, last accessed on May 26, 2009.
13. P. Philips, A. Martin, C. L. Wilson, and M. Przybocki, "AnIntroduction to Evaluating Biometric Systems," (2000).<http://www.frvt.org/DLs/FERET7.pdf>.
14. NSTC subcommittee on biometrics. Biometrics "Foundation Documents" http://www.biometrics.gov/Documents/biofoundationdocs.pdf, last accessed on May 30, 2009.
15. Federal Bureau of Investigation (FBI), Criminal Justice Information Services (CJIS) Division. "Integrated Automated Fingerprint Identification System or IAFIS." http://www.fbi.gov/hq/cjisd/iafis.htm.

16. V. M. Jr and Z. Riha, “Biometric authentication systems,” Tech. Rep. FIMU-RS-2000-08, FI MU Report Series, http://www.fi.muni.cz/informatics/reports/files/older/FIMU-RS-2000-08.pdf, 2000.
17. NSTC subcommittee on biometrics, "Biometrics Overview," 2006. Found at: http://www.biometrics.gov/Documents/BioOverview.pdf.
18. See generally EPIC's Theme Parks and Privacy webpage http://www.epic.org/privacy/themepark/file.
19. James Wayman, ed., “National Biometric Test Center Collected Works,” San Jose State University, August 2000. http://www.engr.sjsu.edu/biometrics/nbtccw.pdf.
20. NSTC subcommittee on biometrics, "Dynamic Signature," 2006. http://www.biometrics.gov/Documents/DynamicSig.pdf
21. Sanchez-Reillo, R., Sanchez-Avila and C., Gonzales-Marcos A., “Biometric IdentificationThrough Hand Geometry Measurements”, IEEE Transactions on Pattern Analysis andMachine Intelligence, October 2000.
22. R. Sanchez-Reillo, C. Sanchez-Avila, A Gonzalez-Marcos, "Biometric identification through hand geometry measurements," IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 22, Pages 1168-1171, 2000.
23. R.Sanchez-Reillo, "Hand geometry pattern recognition through Gaussian mixturemodelling," In Proceedings of the 15thInternational Conference on Pattern Recognition, Volume 2, Pages 937-940, 2000.
24. NSTC subcommittee on biometrics, "Hand Geometry,"2006. http://www.biometrics.gov/Documents/HandGeometry.pdf.
25. D. P. Sidlauskas, “3D hand profile identification apparatus,” *US Patent No. 4736203*,1988.
26. NSTC subcommittee on biometrics, "Iris Recognition," Aug. 2006. http://www.biometrics.gov/Documents/IrisRec.pdf
27. JohnDaugman. "Biometric personal identification system based on iris analysis," U.S. Patent No. 5,291,560, March 1994.
28. Daugman, J. (1993) "High confidence visual recognition of persons by a test of statistical independence." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15(11), pp.1148-1161. http://www.cl.cam.ac.uk/users/jgd1000/iris_recognition.html.
29. LeonardFlom, AranSafir, Iris recognition system, U.S. Patent 4,641,349, 1987.
30. NSTC subcommittee on biometrics, "Face Recognition,"2006.http://www.biometrics.gov/Documents/FaceRec.pdf.
31. Jun Zhang, Yong Yan, M Lades. "Face recognition: eigenface, elastic matching, and neural nets," Proceedings of the IEEE, Volume 85, Issue 9, Pages 1423 – 1435, Sep 1997.
32. D. Bolme, R. Beveridge, M. Teixeira, and B. Draper, "The CSU Face Identification Evaluation System: Its Purpose, Features and Structure," International Conference on Vision Systems, Graz, Austria, April 1-3, 2003, Springer-Verlag, Pages 304-311.
33. W. Stallings, "Cryptography and network security principles and practices", Fourth Edition, Prentice Hall, 2005.
34. *R. Rivest, A. Shamir, L. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM 21 (2), Pages 120–126, 1978.*
35. W. Diffie and M.E. Hellman, New directions in cryptography, *IEEE Transactions on Information Theory*22 (1976), 644-654.
36. TaherElGamal, A public-key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Transactions on Information Theory*31 (1985), 469-472.
37. W. Diffie and M.E. Hellma, "Key exchange protocol,"http://www.adamsinfo.com/diffie-hellman-key-exchange
38. Ian Blake, GadielSeroussi, and Nigel Smart, *Eliptic curves in cryptography*, London MathematicalSociety, Lecture Note Series, no. 265, Cambridge University Press, 1999.

11/5/2010