Reliable and Secure Voice Encryption over GSM Voice Channel

Hoon-Jae Lee, Won-Tae Jang and Tae Yong Kim, *Member, KIMICS*

***Abstract*—In this paper, we study and develope a special secure Dongle to be adapted in GSM SmartPhone for secure voice communication to the serial 20-pin connector in SmartPhone. We design and implement the Dongle module hardware, firmware, and software including cipher crypto-synchronization and cipher algorithm. Also we study and emulate the SmartPhone GUI software interface including communication software module to the Dongle. Finally, we analyze the performances of crypto-synchronization in some noisy environment and also we test the secure Dongle module.**

Index Terms*— *GSM phone security, Crypto dongle, PingPong-128, synchronization, stream cipher.

I. INTRODUCTION

GSM is the most widely used mobile communications system in the world. However the security of the GSM voice traffic is not guaranteed especially over the core network. If encryption is enabled by the GSM operator, then the voice is only encrypted between the GSM mobile phone handsets and the base station, the rest of the communication is send in clear. To achieve maximum security especially from eavesdropping, it is highly desirable to have end-to-end secure communications. In order to achieve end-to-end security, speech must be encrypted before it enters the GSM network.

Encrypting GSM voice communication is a real challenge. At the moment, encrypted GSM voice is sent through the circuit switch data channel (CSD). However, this approach suffers from a number of disadvantages; the GSM network channel gives priority to voice instead of data. Theoretically, it takes longer time to establish a connection between the GSM handset and network when using the CSD channel.

In this paper, we study and develope a special secure Dongle to be adapted in GSM SmartPhone for secure voice communication to the serial 20-pin connector in SmartPhone. We design and implement the Dongle module hardware, firmware, and software including cipher crypto-synchronization and cipher algorithm. Also we study and emulate the SmartPhone GUI software interface including communication software module to the Dongle. Finally, we analyze the performances of crypto-synchronization in some noisy environment and also we test the secure Dongle module.

II. DESIGN OF A CRYPTOGRAPHIC DONGLE

A. Operation Environment

The test environment is set up as shown in figure 1. In this set up, two smart phones are securely communicating with each other via the CDMA or a GSM network. A cryptographic dongle is physically connected to the 20 pin interface of the T-Omnia smartphone. The voice from the sender and the receiver are sent and received to the dongle through the use of an earpiece. A microprocessor unit resides inside of the dongle.

We design and implement the Dongle module hardware, firmware, and software including cipher cryptographic-synchronization and cipher algorithm. Also we study and emulate the Smartphone graphical user interface GUI software including communication software module to the Dongle.

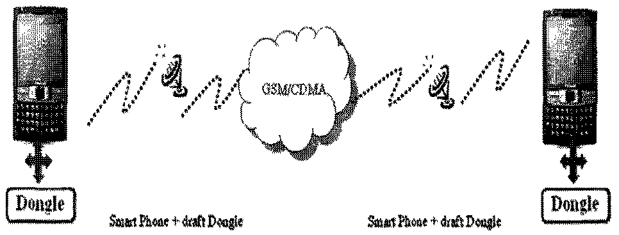


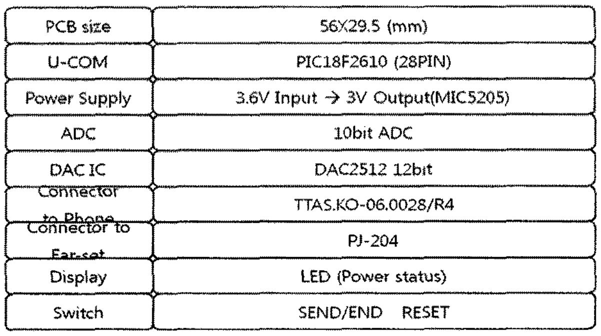
Fig. 1. Operation concept.

B. Design of a Secure Dongle

There are some requirements to perform a secure dongle like as summarized in figure 2. To reduce the size of a secure dongle, the microcontroller was chosen over the PIC series at the recommendation. They both contain serial interfaces and a sufficient number of pins. This MPU consumes small power and support 10 bit ADC for converting analog to digital signal. The size of double layered PCB (Printed Circuit Board) by using this device is 56 by 29.5 mm shown in figure 3.

Manuscript received September 23, 2009; revised October 9, 2009; accepted November 13, 2009.

Hoon-Jae Lee is with the Div. of Computer and Information Engineering, Dongseo University, Busan, 617-716, Korea (Email: hjlee @dongseo.ac.kr)



PCB size	56X29.5 (mm)
U-COM	PIC18F2610 (28PIN)
Power Supply	3.6V Input → 3V Output(MIC5205)
ADC	10bit ADC
DAC IC	DAC2512 12bit
Connector to Phone	TTAS.KO-06.0028/R4
Connector to Ear-set	PJ-204
Display	LED (Power status)
Switch	SEND/END RESET

Fig. 2. General/Electronic Specifications.

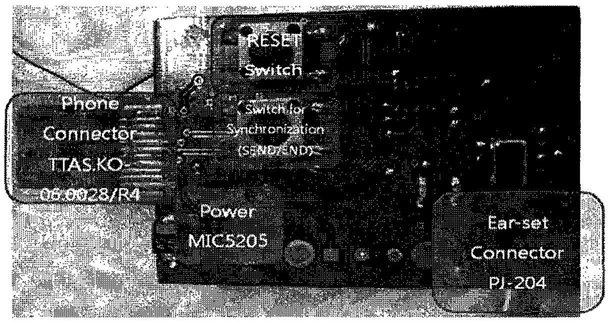

RESET Switch
Switch for Synchronization (SEND/END)
Phone Connector TTAS.KO-06.0028/R4
Power MIC5205
Ear-set Connector PJ-204

Fig. 3. Dongle PCB outline.

C. Proposed Cipher System

Symmetric cryptosystems can be classified as either block ciphers or stream ciphers. A block cipher (EBC mode) divides plaintext into blocks and then encrypts each block independently, whereas a stream cipher encrypts plaintext on a bit-by-bit (or byte-by-byte) basis. Since a stream cipher is based on an exclusive-OR (XOR) operation, the encryption/decryption of bit-stream data bit-by-bit (or byte-by-byte) using a stream cipher is very efficient. Therefore, generally, a stream cipher is much simpler and faster than a block cipher [1]. A block cipher is useful as regards software implement, however, its performance can be degraded in a noisy channel due to its channel error propagation properties. Conversely, a stream cipher is good for high-speed enciphering and it can be implemented in noisy (wireless) channels because it does not produce error propagation. The major problem of a stream cipher is the difficulty in generating a long unpredictable bit pattern (keystream). In the one-time pad in a stream cipher, the keystream is a sequence of randomly generated bits, and the probability that an individual bit will be 1, independent of the other bits, is equal to one half. An ideal keystream in a one-time pad is purely random with an infinite length. Such a keystream can neither be generated by the receiving end nor distributed to the receiving end. Currently, pseudorandom bit generators are widely used to construct keystreams by generating a fixed-length pseudorandom noise. The ability to increase the length of a keystream while maintaining its randomness, is crucial to the security of a stream cipher.

In general, cipher system is required to control automatically in various features like as encryption processing speeds, encryption methods, encryption algorithms, synchronization methods, network interfaces, user authentications, security-bit levels and the size of synchronization patterns.

In this paper, we design a secure GSM dongle for reliable cipher system using PingPong cipher [2]. The system contains a secure dongle for secure voice communication. Security-bit levels classified in 128-bit, and synchronization methods in continuous synchronization, initial synchronization and absolute synchronization. The size of synchronization patterns is classified in 32-bit SYNPAT.

Therefore, encryption algorithms with a PingPong-128 for data confidentiality, and a synchronization generator and detector with a SYNPAT-32 for system stabilization are all designed and analyzed.

1) Design of a Cipher System

Stream ciphers can be classified into self-synchronous stream ciphers and synchronous stream ciphers [3]. In a self-synchronous method, keystream synchronization is established autonomously based on the feedback of the ciphertext, however, one-bit errors can be propagated in the channel relative to the size of the shift register used. To eliminate this weakness, keystream synchronization in a synchronous stream cipher is re-established by sync protocols when out-of-synchronization occurs. In this case, no bit-errors will be propagated in the channel, which is why the latter cipher is more generally used [4]. In a stream cipher, the pseudorandom binary sequences must be equal at both the transmitter and the receiver. The output keystreams should always coincide with each other, which establishes keystream synchronization. In general, keystream synchronization methods can be classified into initial synchronization and continuous synchronization. In the initial synchronization, the two keystreams are only synchronized initially at the transmitter and at the receiver, whereas in the continuous synchronization the keystreams are synchronized both initially and periodically. Keystream synchronization exchanges the synchronization patterns (SYNPAT) at the transmitter and at the receiver to initialize the starting point of the keystream cycle. The reliability of the keystream synchronization is critical to the overall system performance and communication efficiency.

The proposed synchronous stream cipher system is shown in fig. 4 and fig. 5. In fig. 4, a secure voice communications (cipher system) for GSM links. The function of each block in fig.5 is as follows: Block (1) is a security agent in plain mode, block (3) is the main controller for monitoring the system and it contains the master CPU, block (4) is the synchronization pattern generator that matches the keystream sequences from the sender with those from the receiver for the initial synchronization, blocks (5) and (12) are the session-key buffers that initiate the keystream generators at the sender and receiver ends, blocks (6) and (11) are the session key constructors that distribute/ construct a secure session key through a public channel from the sender to the receiver, blocks (7) and (13) are the same keystream generators with high security both at the sender and the receiver ends, blocks (8) and (14) are enciphering process at the sender and deciphering process at the receiver (Zero Suppression is required if necessary), switch (9) is the data selector for the synchronization pattern, session key, or ciphertext data, block (2) is a security agent in cipher mode, and block (10) is the synchronization pattern detector that identifies the sync pattern in the received data.

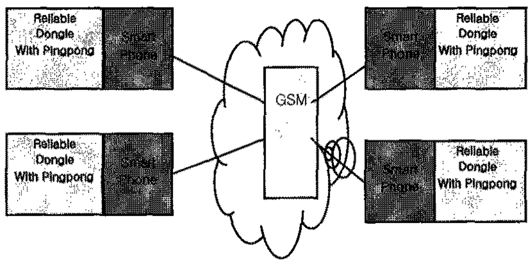

Reliable Dongle With Pingpong
Smart Phone
GSM
Smart Phone
Reliable Dongle With Pingpong
Reliable Dongle With Pingpong
Smart Phone
Smart Phone
Reliable Dongle With Pingpong

Fig. 4. GSM and secure voice system.

2) Basic Notions

In this paper, we propose a highly reliable initial keystream synchronization method with a generator and a receiver. A generator part (block 4) and a detector part (block 10) of the keystream synchronization, shown in fig. 5, are used to match the keystream sequences at both the sender and the receiver ends. In the proposed scheme, the keystream synchronization exchanges the synchronization patterns (SYNPAT) to initialize the starting point of the two keystream cycle sequences at the sender and receiver ends.

The statistical properties are selected based on the following decision criteria [5]:

- A good autocorrelation property is required.
- The same rate must be achieved on "0" and "1" in the pattern.
- A short run should be larger than a long run.

Accordingly, the pattern is determined based on the above criterion using a Gold sequence generator, as in fig. 6. In the figure, the primitive polynomials of a 31-stage LFSR1 and LFSR2 were selected, then the N-bit pattern (here, N= 32-bit SYNPAT) was generated and checked using the decision criteria. The two selected primitive polynomials and generated patterns are follows:

- $h_1(x) = x^{31} + x^{11} + x^2 + x + 1$ (31-stage LFSR$_1$),
- $h_2(x) = x^{31} + x^9 + x^3 + x + 1$ (31-stage LFSR$_2$)
- SYNPAT-32 = 8613 7D4B (hexadecimal) if channel bit error rate (BER) approached about 10^{-3}

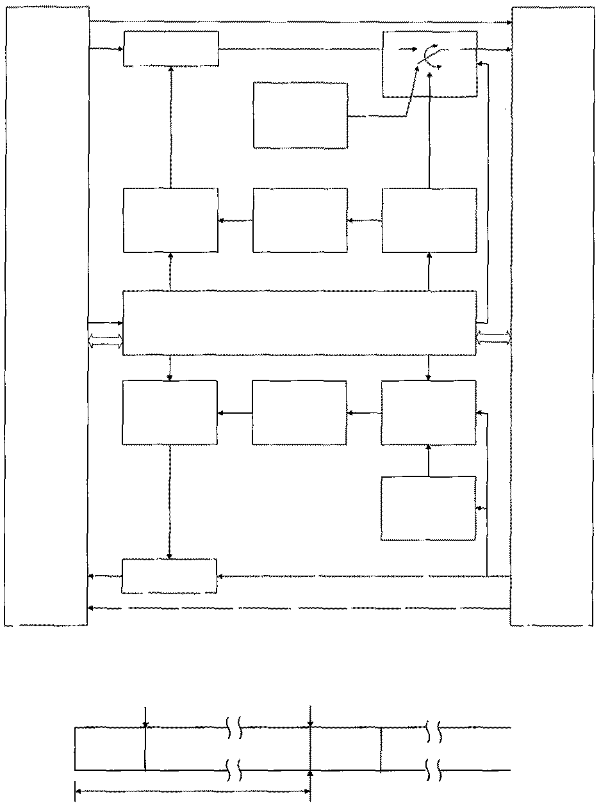


Fig. 5. Proposed cipher system with an absolute synchronization.

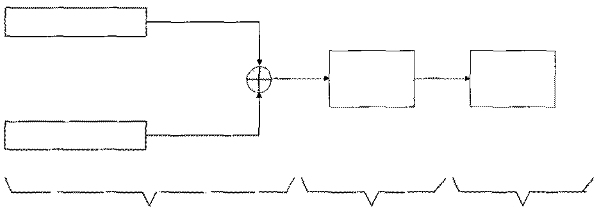


Fig. 6. Synchronization pattern generator in sender (N=32).

3) Design of Synchronization Method

In stream cipher system, cipher synchronization plays a transmission and reception synchronization role [3].

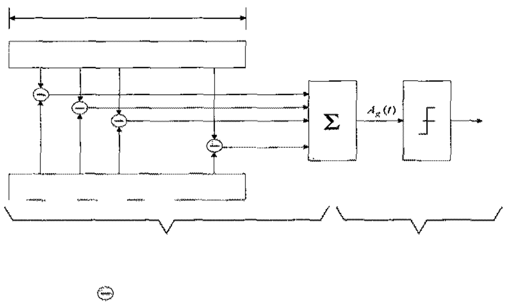


Fig. 7. Synchronization pattern detector in receiver (N=32).

Autocorrelation value of synchronization pattern is denoted as $A(t)$. Threshold is denoted as THR. The NT can be found by Beker et al. [6]. Autocorrelation, threshold, the number of agreements and the number of disagreements are follows:

$$A(t) = \frac{A_g(t) - D_g(t)}{N} \tag{1}$$

$$THR = N - N_T$$

Where, $A_g(t) = \sum_{i=1}^{N} X(i) \Theta R(i)$: the number of agreements in bits,

$D_g(t) = \sum_{i=1}^{N} X(i) \oplus R(i)$: the number of disagreements in bits,

$$A_g(t) + D_g(t) = N,$$

Θ denotes an exclusive-NOR operation, and
$\oplus$ denotes an exclusive-OR operation.

In fig. 7, the synchronization pattern detector is used to calculate the truth and false within one clock as adjusting summation part in system clock. It is difficult to embody this in hardware. The synchronization pattern detector at the receiver consists of a correlation part and decision part, as shown in fig. 7. The correlation part computes the number of agreements in bits, *Ag*(*t*), between an input pattern and the reference pattern (*SYNPAT*). The decisional part then compares the number of agreements in bits with a threshold, *THR*, and displays one of the following: If the number of agreements in bits is larger than the threshold, "synchronization detected (*SYNC_ DET*=1)", otherwise "synchronization failed (*SYNC_ DET*=0)".

We selected a 32-bit SYNPAT for reliability-level approaches as the following :

- 0x86137D4B(Hexadecimal) = 1000 0110 0001 0011 0111 1101 0100 1011 (Binary)

We have a rule for SYNDET with threshold=3;

- No error in SYNPAT (32-bit the same) → SYNPAT OK
- 1-bit error in SYNPAT (31-bit the same) → SYNPAT OK
- 2-bit error in SYNPAT (30-bit the same) → SYNPAT OK
- 3-bit error in SYNPAT (29-bit the same) → SYNPAT OK
- More than THR=3 (4-bit or more error) → No SYNPAT

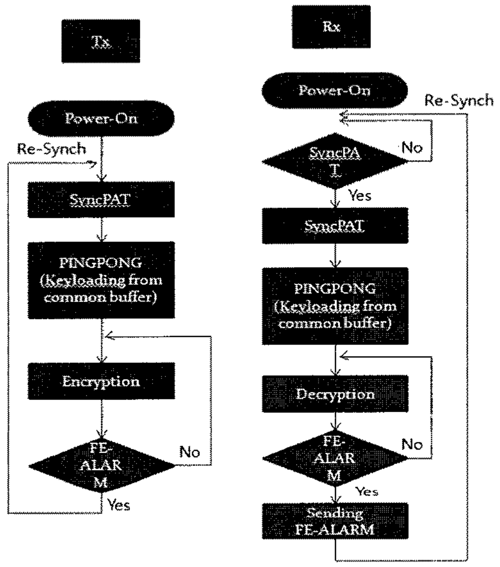


Fig. 8. Status Diagrams for SYNPAT generator at the Sender and detector at the Receiver.

D. PingPong Cipher Algorithm

1) Description on PingPong-128

PingPong family generators [2] are simple, easy to implement in hardware and in software, and high secure. PingPong family in fig. 9 is a hybrid generator, combining the LM generator (improved summation generator) with a high secure clock-controlled generator. LFSR A is clock-controlled by function f_b, it has a random integer output. And LFSR B is clock-controlled by function f_a, it also has a random output. Two clock-controlled functions give a multiple clock to the other LFSR. It makes that the output should be more unpredictable. Pingpong Family generator outputs z_j, c_j and c_j from each LFSR outputs a_j and b_j, previous carry c_{j-1} and previous memory d_{j-1} as in fig. 9.

$$c_j = f_c = a_j b_j \oplus (a_j \oplus b_j) c_{j-1} \tag{2}$$

$$y_j = f_y = a_j \oplus b_j \oplus c_{j-1} \tag{3}$$

$$z_j = f_z = y_j \oplus d_{j-1} \tag{4}$$

$$d_j = f_d = f(a_j, b_j, d_{j-1}) = b_j \oplus (a_j \oplus b_j) d_{j-1} \tag{5}$$

where (y) is the output sequence of summation generator, (a) the output sequence of LFSR 1, (b) the output sequence of LFSR 2, (c) carry sequence, c_{-1} = 0 carry initialization value, (d) memory sequences, d_{-1} =

0 memory initialization value.

PingPong-128 is a special design from the PingPong family of stream ciphers. It has two mutually clocking LFSRs and a single memory bit. The LFSRs are of lengths 127 bits and 129 bits. Together with the memory bit they give PingPong-128 an internal state of 257 bits. PingPong-128 takes a 128-bit key and a 128-bit initialization vector to fill the internal state.

Keystream Generation: The PingPong generator, depicted in fig. 9 produces the output keystream by combining the LFSR sequences and the memory sequence. PingPong-128 has two mutually clocking LFSRs L_a and L_b, and a single bit of memory c. Two primitive polynomials, $p_a(x)$ and $p_b(x)$ are following:

$$\begin{aligned} p_a(x) = {} & x^{127} \oplus x^{109} \oplus x^{91} \oplus x^{84} \oplus x^{73} \oplus x^{67} \oplus x^{66} \oplus x^{63} \oplus x^{56} \oplus \\ & x^{55} \oplus x^{52} \oplus x^{48} \oplus x^{45} \oplus x^{42} \oplus x^{41} \oplus x^{37} \oplus x^{34} \oplus x^{30} \oplus x^{27} \oplus \\ & x^{23} \oplus x^{21} \oplus x^{20} \oplus x^{19} \oplus x^{16} \oplus x^{13} \oplus x^{12} \oplus x^{7} \oplus x^{6} \oplus x^{2} \oplus \\ & x^{1} \oplus 1 \end{aligned} \tag{6}$$

$$\begin{aligned} p_b(x) = {} & x^{129} \oplus x^{125} \oplus x^{121} \oplus x^{117} \oplus x^{113} \oplus x^{109} \oplus x^{105} \oplus x^{101} \oplus x^{97} \oplus \\ & x^{93} \oplus x^{89} \oplus x^{85} \oplus x^{81} \oplus x^{77} \oplus x^{73} \oplus x^{69} \oplus x^{65} \oplus x^{61} \oplus x^{57} \oplus \\ & x^{53} \oplus x^{49} \oplus x^{45} \oplus x^{41} \oplus x^{37} \oplus x^{33} \oplus x^{29} \oplus x^{25} \oplus x^{21} \oplus x^{17} \oplus \\ & x^{13} \oplus x^{9} \oplus x^{5} \oplus 1 \end{aligned} \tag{7}$$

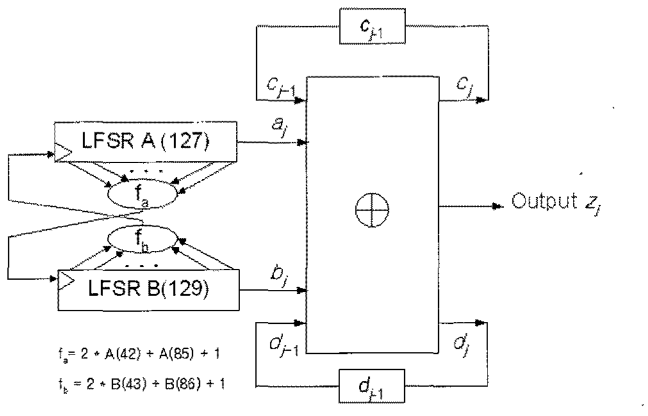


Fig. 9. PingPong-128 Keystream generator.

Two clock-control functions, $f_a(L_a)$ and $f_b(L_b)$, and the output keystream bit z_j and memory bit c_j at time j are defined to be identical to the summation generator:

$$f_a(L_a) = 2L_{a42}(t) + L_{a85}(t) + 1 \tag{8}$$

$$f_b(L_b) = 2L_{b43}(t) + L_{b86}(t) + 1 \tag{9}$$

Clock Control For PingPong-128, both LFSRs are irregularly clocked, with each register controlling the clocking of the other. Two taps are taken from L_a to calculate a value in the range of 1~4 clock(s), and L_b is clocked 1 to 4 times according to this value. Similarly, a value is calculated from two taps taken from L_b to clock L_a. The clock control is calculated by above two functions, f_A and f_B. This clocking scheme can be applied to the PingPong family of keystream generators with n underlying LFSRs, where L_i is used to clock L_{i+1} and L_1 is clocked by L_n.

In some communication systems, errors occur which require that the entire message be resent. When a synchronous stream cipher is used, then security requires that a different keystream sequence be used. To achieve this, the rekeying of a stream cipher should include a method for re-initialization using both the secret key and an additional initialization vector which is sent in the clear, or otherwise publicly known. We now describe a proposed method for the initial key loading and for the rekeying of PingPong-128. For PingPong-128, both *k* and *iv* have a length of 128 bits, and together they fill 257 bits of internal state. The initialization process can also be used for rekeying. The process to generate the initial state for the keystream generator uses the generator itself twice. The starting state of La is obtained simply by XORing the two 128-bit binary strings of the key, *k*, and *iv*, that is, $L_a = (k \oplus iv) \bmod 2^{127}$. The starting state of 129 bits for L_b is obtained by considering the 128-bit key, embedded in a 129-bit word and shifted 1 bit to the left, and XORing that with the initialization vector embedded in a 129-bit word with a leading zero, that is, $L_b = (k << 1) \oplus (0 \mid iv)$. Now the cipher is to run to produce an output string of length 257 bits. For the second iteration of the cipher, the first 128 bits of this output string are used to form the initial state of L_a, and the remaining 129 bits are used to form the initial state of L_b. The cipher is run a second time to produce an output string of length 257 bits. The output from this second application is used to form the initial state of the keystream generator when we begin keystream production. As previously, the first 128 bits form the initial state of L_a, and the remaining 129 bits form the initial state of L_b. It is very unlikely that either LFSR will be initialized with the all zero state. By employing the PingPong algorithm itself, we take advantage of both the known security properties of the algorithm and also its fast implementation. Due to the high security of PingPong we conclude that the best attack in the rekeying scenario is exhaustive key search.

Implementation Issue Both LFSRs in PingPong-128 use the Galois implementation rather than the Fibonacci implementation. This is a design decision based on the software performance of the implementation. It is observed that the Galois implementation is much more efficient in software than the Fibonacci, although both implementations are equally efficient in hardware. It is worth noting that these two implementations give different output sequences with the same initial LFSR states, therefore it is essential to specify the style of implementation.

III. TEST AND ANALYSIS

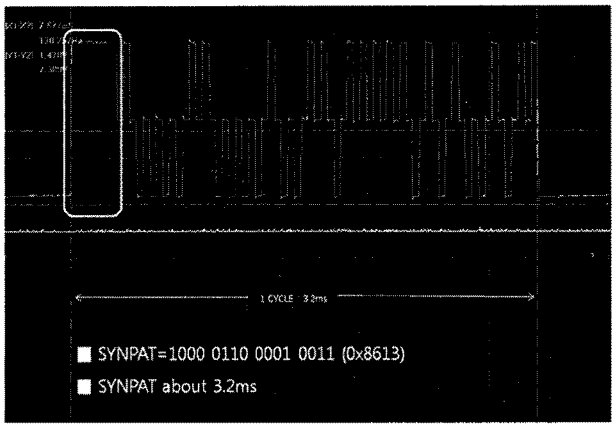
A. Synchronization method

We let the size (N) of synchronization pattern to 32, the channel error rate (BER) about 10^{-3}, and vary the synchronization threshold value (N_t) from the range around 0 ~32. Then we calculate/simulate the probability of false detection (P_{FA}), the probability of detection (P_D) and the probability of missing pattern (P_M) for channel BER at 10^{-3}. The result at N_t=3 is optimal for the complexity of detector and $P_D > 1-10^{-6}$ condition (see fig. 10).

SYNPAT= 0x86137D4B (Hexadicimal)
= 1000 0110 0001 0011 0111 1101 0100 1011 (Binary)
* DETECTOR N_SIZE (N) = 32
* CHANNEL BIT ERROR RATE (BER) = 10^{-3}

Nt=0 P_{FA}=.23283064365386960000D-09 P_D=.96849107428602590000D+00
P_M=.31508925713974120000D-01
Nt=1 P_{FA} =.76834112405776980000D-08 P_D=.99951381287526890000D+00
P_M=.48618712473114110000D-03
Nt=2 P_{FA} =.12316741049289700000D-06 P_D=.99999514668006910000D+00
P_M=.48533199309419570000D-05
Nt=3 P_{FA} =.12780074030160900000D-05 P_D=.99999996483650230000D+00
P_M=.35163497669188360000D-07
Nt=4 P_{FA} =.96505973488092420000D-05 P_D=.99999999980310470000D+00
P_M=.19689527785970990000D-09
Nt=5 P_{FA} =.56537101045250890000D-04 P_D=.99999999999911370000D+00
P_M=.88629104055826250000D-12
Nt=11 P_{FA} =.55092082591727380000D-01 P_D=.10000000000000000000D+01
P_M=.00000000000000000000D+00
Nt=12 P_{FA} =.10766357486136260000D+00 P_D=.10000000000000000000D+01
P_M=.00000000000000000000D+00

Fig. 10. Simulation Results (Probability of False-Alam, Detection and Missing).



■ SYNPAT=1000 0110 0001 0011 (0x8613)
■ SYNPAT about 3.2ms

Fig. 11. Measurement of SYNPAT cycle.

Another design aspect is to check if the two dongle are communicating together as well as be able to tell when to start encrypting/decrypting the signal. For this, a synchronization pattern was used. The waveform in the fig. 11 presents this pattern when the listed below operations are run

- Power ON
- Select One dongle & push reset button
- If the sync-pattern is matched, then communication each other
- If the sync-pattern d is not matched, then the dongle making the noise which similar buzzer
  - Push the reset button once more
  - If it does not work, several times repeat, then Power OFF and return to step 1)

B. Cryptographic Dongles

Two dongles are tested in the CDMA network.

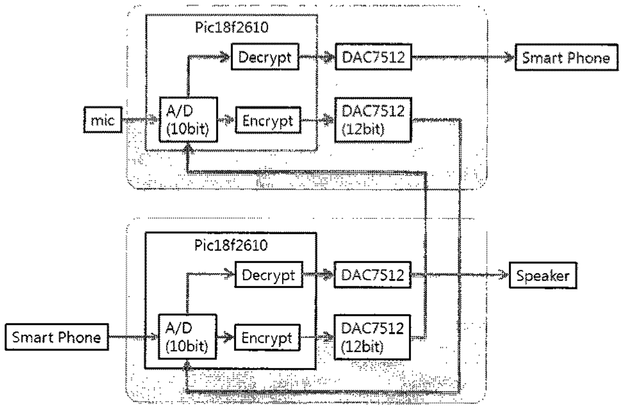


Fig. 12. Signal flow diagram for testing two Smartphone & dongle.

In this test, some noise occurred which made it difficult to hear the other party. Strangely, the noise level verifies from one keystreams to another keystreams. It is predicted that the noise level is low if the results of the XORing of the plain signal with the keystreams is close to the original plain signal. Therefore, some values of keystreams were selected in such a way they generate low noise when encrypted with the plain signal. The reason behind this is to test the quality of the encrypted voice over the voice channel without having added extra overheads such as the use of synchronization pattern. The synchronization problem was left beside until a satisfactory results obtained from the basic test. The synchronization problem needed more time to investigate and developed.

IV. CONCLUSION

In this paper, we investigated sending encrypted voice using the CDMA voice channel with the aim of testing this in the GSM channel. A cryptographic dongle with a simple encryption algorithm was used. The dongle used PIC18F2610 (28 pins) microcontroller for signals processing including analog to digital conversion and digital to analog conversion, encryption and communicating with a smart phone. T-Omnia smart phone was used to provide power to the dongle as well as passing the encrypted speech to the smart phone which then transmit the signal to the network.

In this paper, we studied and developed a special secure Dongle to be adapted in GSM SmartPhone for secure voice communication to the serial 20-pin connector in SmartPhone. We designed and implemented the Dongle module hardware, firmware, and software including cipher crypto-synchronization and cipher algorithm. Also we studied and emulated the SmartPhone GUI software interface including communication software module to the Dongle. Finally, we analyzed the performances of crypto-synchronization in some noisy environment and also we tested the secure Dongle module.

ACKNOWLEDGMENT

This work was supported in part by the Dongseo Frontier 2009 and the research foundation of Dongseo University in 2009.

REFERENCES

[1] Schneier, S. 1996. Applied Cryptography : Protocols, Algorithms, and Source Code in C (2nd Ed.). John Wiley and Sons, Inc., New York, USA.

[2] Lee, H., Chen, K. 2007. PingPong-128, An New Stream Cipher for Ubiquitous Application. IEEE CS (ICCIT 2007), pp.1893-1899.

[3] Lee, H. 1997. An Improved Synchronous Stream cipher system for a Link Encryption. PhD thesis in KyungPook National University, Daegu, Korea.

[4] Menezes, A., Oorschot, P. Van, Vanstone, S. 1997. Handbook of Applied Cryptography, CRC Press.

[5] Tilborg, H.C.A. 2000. Fundamentals of Cryptology. Kluwer Academic Publishers.

[6] Beker, H.J., and Piper, F.C. 1985. Secure Speech Communications. Academic Press, London.

[7] A.J. Menezes, P.C. Oorschot and S.A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1997.

[8] J. L. Massey, "Shift-Register Synthesis and BCH Decoding," IEEE Trans. on Infor. Theo., Vol. IT-15, No. 1, pp. 122-127, Jan. 1969.

[9] A. Biryikov and A. Shamir, "Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers", Advances in Cryptology, Proceedings of ASIACRYPT00, LNCS 1976, pp.1-13, 2000.

[10] Chen, K., Henrickson, M., Millan, W., Fuller, J., Simpson, A., Dawson, E., Lee, H., Moon, S. 2004. Dragon: A Fast Word Based Stream Cipher. LNCS Vol. 3506 (ICISC'2004), pp.414-431.

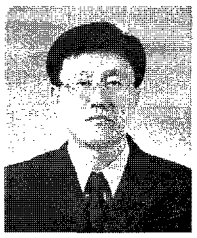
[11] Clark, A., Dawson, E., Fuller, J., Golic, J., Lee, H., Millan, W., Moon, S., Simpson, L. 2002. The LILI-II Keystream Generator. LNCS Vol. 2384(ACISP'2002), pp.25-39.

[12] Lee, H., Moon, S. 2002. Parallel Stream Cipher for Secure High-Speed Communications. Signal Processing Vol.82, No.2, pp.259-265.

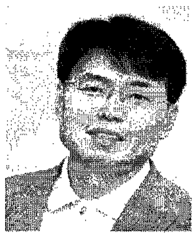
[13] Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J., Tokita, T. 2000. Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms — Design and Analysis. Selected Areas in Cryptography, pp. 39–56.

[14] ARIA. http://www.nsri.re.kr.

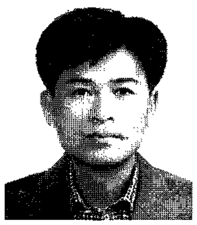
[15] NIST. 2001. Announcing the Advanced Encryption Standard (AES). FIPS-197



Hoon-Jae Lee received BS, MS, and PhD Degrees in electronic engineering from Kyungpook National University, Daegu, Korea in 1985, 1987, and 1998, respectively. He is currently an associate professor in the School of Computer and Information Engineering at Dongseo University. From 1987 to 1998, he was a research associate at the Agency for Defense Development (ADD). His current research interests include developing secure communication system, side-channel attack and USN/RFID security.



Won-Tae Jang received BS Degree in electronic engineering from SungKyunKwan University, Seoul, Korea in 1985, and received MS Degree in electronic engineering from University of Seoul, Korea in 1995. He is currently an assistant professor in the School of Computer and Information Engineering at Dongseo University. From 1989 to 2000, he was a research associate at the Korea Telecom International. His current research interests include developing mobile communication system, RFID system.



Tae Yong Kim was born in Korea, on November 14, 1965. He received a Ph.D. at Okayama University, Graduate School of Natural Science and Technology, in 2001 and is presently an assistant professor at Dongseo University, Korea. He was worked on analysis of antenna device, flow sensor system, numerical modelling, and simulation for various fields. Member of Institute of Control, Robotics and System and The Korea Institute of Maritime Information & Communication Sciences.