Power-Based Key Hopping (PBKH) and Associated Hardware Implementation

Rabie A. Mahmoud[1], Magdy Saeb[2]

1. *Department of Mathematics, Faculty of Science, Cairo University, Cairo, Egypt.*
2. *Computer Engineering Department, Arab Academy for Science, Tech. & Maritime Transport, Alexandria, Egypt,*

mail@magdysaeb.net

Abstract: *Power-Based Key Hopping (PBKH) is a process of key hopping that is used to interchange the user key of a cipher. Power based key hopping is founded on the idea of dynamic frequency power-based hopping to change the user key. This is achieved through computing the power of a previous cipher packet and comparing it with a standard value. In this work, we discuss various key hopping methods and suggest a procedure of power based key hopping. Moreover, we provide a Field Programmable Gate Array (FPGA) hardware implementation of the proposed key hopping technique.*

Keywords: *Power Based Key Hopping; Security; Hardware; FPGA*

I. INTRODUCTION

Power-Based Key Hopping (PBKH) is a key hopping method that is based on utilizing the idea of power based dynamic frequency hopping to provide a new procedure of key hopping. PBKH utilizes four keys; one acts as the authentication key where the power of cipher text packet will be the standard of changing the key. In the following sections, we provide a review of various key hopping methods, the structure of power based key hopping and the formal description of power based key hopping algorithm. Moreover, we provide the details of our hardware implementation, a discussion of the results of the FPGA implementation and finally a summary and our conclusions.

II. GENERAL KEY HOPPING TECHNIQUES

Various key hopping techniques were suggested in the literature. The major procedures are summarized as follows:

A. NextComm Key Hopping

"NextComm" [1] has proposed changing the keys frequently to overcome the weakness of Wired Equivalent Privacy (WEP) without incurring possible overheads and other complexities of the network. The key hopping process is designed as follows:

• The shared keys which are maintained in network devices are not used directly for encryption/decryption but the session keys are derived using a robust hash function for encryption and decryption purposes where the 128-bit MD5 one-way hash function is chosen.

• The session seeds which are the output of the large counter are changed frequently to ensure that the resulting session keys not to be reused where the size of the counter should also be large enough to ensure the seeds are not reprocessed during the lifetime of the secret shared keys.

• Session keys are used in the same way in encryption and decryption as the shared keys are used in the standard WEP process.

B. State Based Key Hop (SBKH)

State Based Key Hop (SBKH) protocol is created to provide a strong lightweight encryption scheme for battery operated 802.11 devices. SBKH includes base key pair, key duration, RC4 states, offset and an explicit SBKH sequence counter as the sharing parameters between communicating entities. Base key pair consists of two 128-bit keys which are shared by all entities in a network. The keys can be used as per-session keys that are agreed upon between the two communicating nodes. SBKH uses a 24-bit initialization vector IV within the original 802.11 WEP data frames as a SBKH sequence counter [2].

C. Dynamic Re-Keying with Key Hopping (DRKH)

Dynamic Re-keying with Key Hopping (DRKH) encryption protocol uses RC4 encryption technique where each two communicating entities share a secret list of the static keys. One of these keys is an authentication key that is used to encrypt and decrypt mutual authentication messages. The transmitted packets in DRKH between the two communicating nodes take place in several consecutive sessions where the access point (AP) in WLANs will be responsible for determining the sessions' start and end times. The access point maintains the session counter. A one-way hash function such as MD5 or SHA-1 is used in DRKH to hash the session counter with each of the four secret keys to generate four session keys and then these session keys are used instead of the secret keys to encrypt and decrypt packets during the lifetime of a session. Likewise, an initialization Vector (IV) is used in DRKH where each session key has a corresponding IV which value is incremented for every new packet to be

encrypted using that particular session key. Non-linear lookup-table based substitution technique to mix a session key with its corresponding IV value is used to reinitialize RC4 state instead of using the key scheduling algorithm where one of the four derived session keys is selected according to a previously agreed upon key hopping sequence for every packet to be encrypted. The Key Hopping Sequence determines the order in which session keys are used to encrypt and decrypt packets where the AP sends the key hopping sequence to the wireless station after the completion of a successful mutual authentication process [3].

D. Dynamic frequency hopping based power (DFHBP)

These methods can be generally divided into three categories:

- Full-replacement dynamic frequency hopping which assumes changing all current frequencies of poor quality after each physical layer frame,
- Worst dwell (DFH) where only one frequency (the lowest quality one) in a frequency-hop pattern is changed,
- Threshold (SIR)-based DFH where a subset of currently used frequencies is changed,

More recently, another version of dynamic frequency hopping is called Dynamic Frequency Hopping Based Power [4] which can be explained as follows:

A subset of currently used frequencies is changed depending on a criterion value of power. In each frame, the power is measured on the used frequencies and the current hopping pattern is changed if the measured power does not achieve the required threshold. Only the frequencies in with low powers are changed. Any frequency that meets the threshold can be used as a replacement frequency, and thus there is no need to scan all possible frequencies.

III. POWER BASED KEY HOPPING (PBKH)

Our proposed method uses a discrete form of signal analysis where the signal power is measured as the mean of the signal encountered. In this case, the power of discrete signal $x_{[i]}$ with length L is determined by the mean of $x_{[i]}$

$$\Pr = \frac{1}{L}\sum_{i=1}^{L}\left|x_{[i]}\right|^2 = \text{mean}(x^2) \qquad (1)$$

In other words, the power of a packet of binary sequence with length L can be determined by the number of ones in that packet compared with the number of zeros [5]. In this respect, our proposed power based key hopping (PBKH) has its roots in the above mentioned Dynamic frequency hopping based power (DFHBP). The proposed method utilizes four keys. Each two communicating entities maintain four secret keys, one of them will be an authentication key and the first key will be used as the default key at each time the communication session is started. The first key will be used to encrypt and decrypt the first plaintext packet then the sender and receiver compute the power of the ciphertext to determine which key will be used to encrypt and decrypt the next packet. When the power of ciphertext packet which is the number of ones equals a certain predefined value known before hand between the communicating entities, the same key will be used to encrypt and decrypt the next packet. If the power of ciphertext packet is larger than this value, then the next key from the set of secret keys is to be used. However, the previous key from secret key list is to be used to encrypt and decrypt the next packet when the power of ciphertext is less than this certain value. Initialization vector IV will be concatenated to the keys and incremented by one each clock independent of the used key where we use long IV consisting of 64 bits. In this process, the PBKH does not specify a key duration between the two communicating entities. Therefore, the process provides a higher degree of randomness increasing the entropy and thus improving the security.

IV. ALGORITHM

In this section, we provide the formal description of the power based key hopping algorithm as follows:

Algorithm: Power Based Key Hopping (PBKH)
INPUT: *Plain text message (P), User Authentication Key (K), User Key (K1), User Key (K2), User Key (K3), Initialization Vector (IV), predefined and known value between communicating entities (a).*
OUTPUT: *Cipher Text (C)*
Algorithm body:
Begin
Begin power based key hopping
1. Read user keys;
2. Authentication key with IV is used for authentication messages.
3. Encrypt first packet by calling the encrypt function using the user key K1 (default key) concatenated with the initial vector IV;
4. Compute the power of the encrypted packet;
*5. Compare the value of power of encrypted packet with certain value **a** defined and known between the communicating nodes;*
*5.1 If the power Pr = **a** then the same key is used to encrypt/decrypt the next packet.*
*5.2 If the power Pr > **a** then the next key is used to encrypt/decrypt the next packet.*
*5.3 If the power Pr < **a** then the previous key is used to encrypt/decrypt the next packet*
6. Increment the initial vector IV by one IV = IV + 1;
7. Encrypt the packet calling the encrypt function using the chosen key (in step 5) concatenating with new initial vector IV
8. Generate a new set of keys using a standard hash function;
9. Repeat steps 4 till 7 if message cache is not empty;
10. When the plain text messages are finished then halt;
End power based key hopping;
End Algorithm;

Function ENCRYPT
Begin
1. Read next message bit;
2. Read next key bit from user key;
3. Use any standard encryption algorithm to encrypt the plaintext; (In the very simple version of this protocol, one can use XOR operation, however generally this is not recommended)
6. Perform the encryption;
7. Store the resulting cipher;
End;

This procedure can be summarized as shown in Figure 1. In this Figure, we use the state diagram for a 64-bit IV and 192-bit user keys to encrypt and decrypt 256-bit plaintext packets with a given power threshold value equal to 128.

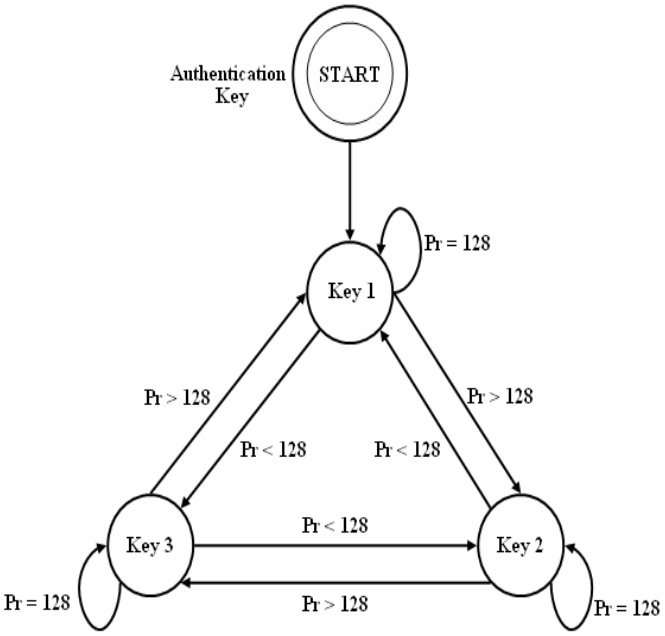


Figure 1. Power based key hopping scheme

V. FPGA IMPLEMENTATION

The concept of the power based key hopping applied to change cipher key leads to a relatively easy to design FPGA-based implementation. We have implemented the proposed technique applying VHDL hardware description language [6], [7], [8] and utilizing Altera design environment Quartus II 9.1 Service Pack 2 Web Edition [9]. The circuit is based on the idea of encrypting 256-bit plaintext blocks using 256-bit user keys that produce 256-bit ciphertext blocks. Four 192-bit keys are used and a 64-bit initialization vector IV which is incremented by one at each falling-edge of the clock trigger. Authentication key and three user keys will be interchanged depending on the power of previous encrypted text packet where the power threshold that is to be compared with is equal to 128. The schematic diagram for a demonstrative 256-bit power based key hopping module is shown in Figure 2. The design was implemented using an EP2C70F896C6, Cyclone II family device.

The worst case pin-to-pin delay was found to be equal to 15.592 ns from source pin "INPUT [110]" to destination pin "OUTPUT [110]". The longest pin-to-register delay was equal to 23.745 ns and the shortest pin-to-register delay was 9.485 ns. The longest register-to-pin delay was 9.206 ns and the longest register-to-register delay was 17.064 ns. The worst case clock-to-output delay was 12.359 ns from clock "CLK2" to destination pin "OUTPUT [69]" through register "KH_Power: inst|pre_temp_key [26]". Clock "CLK2" had an internal f_{max} of 57.91 MHz (period = 17.267 ns) between source register "KH_Power:inst|IV [60]" and the destination register "KH_Power:inst|pre_temp_key [160]". A series of screen-captures of the different design environment output are shown in Figures 3 to 8. Figures 3, 4, 5, and 6 provide the indication of a successful compilation and parts of RTL for power based key hopping respectively. These are shown as a series of repeated MUX's with different connections and a state machine block. Figure 7 displays the power based key hopping simulation showing the output encrypted bits. Figure 8 demonstrates the floor plan. The details of the analysis and synthesis report are shown in appendix A.

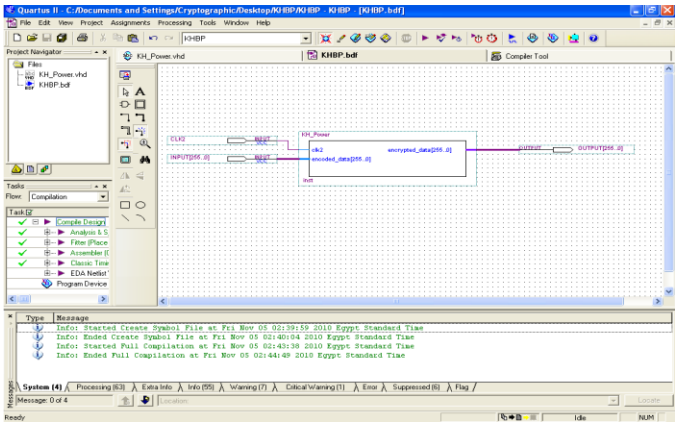


Figure 2. Schematic diagram of power based key hopping implementation

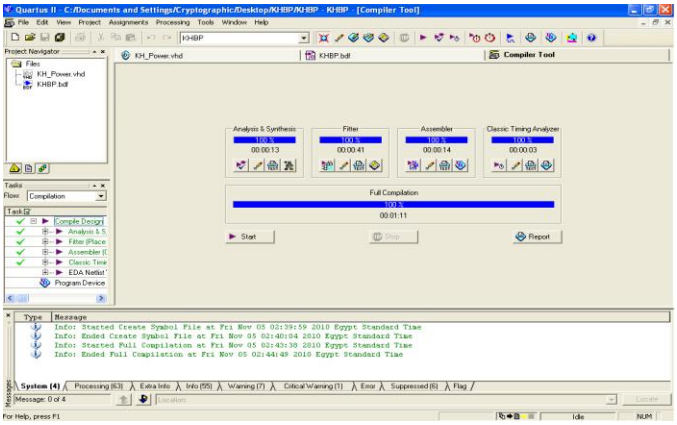


Figure 3. Compiler tool screen showing correct implementation

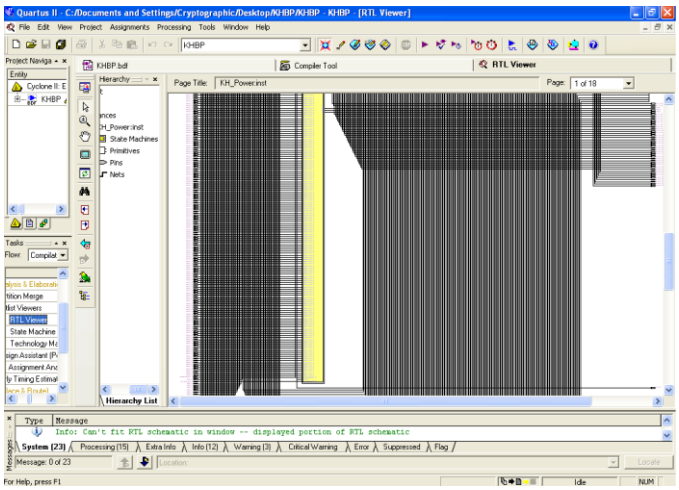


Figure 4. RTL screen for part of power based key hopping implementation

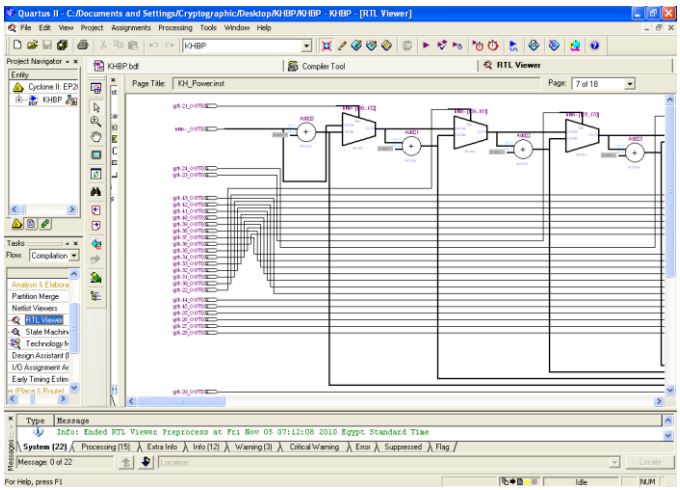


Figure 5. RTL screen for part of the resulting circuit diagram

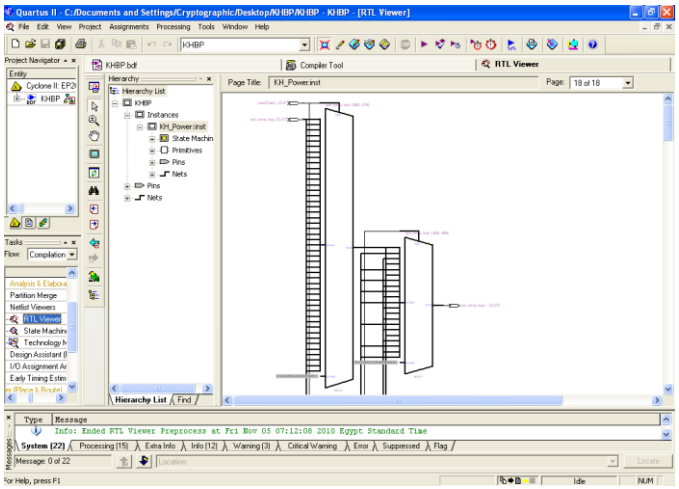


Figure 6. RTL screen for part of power based key hopping implementation

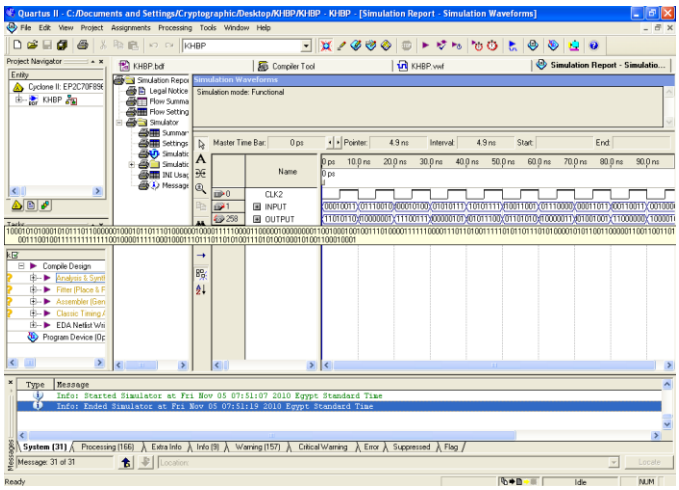


Figure 7. Simulator screen showing the output encrypted data depending on power based key hopping

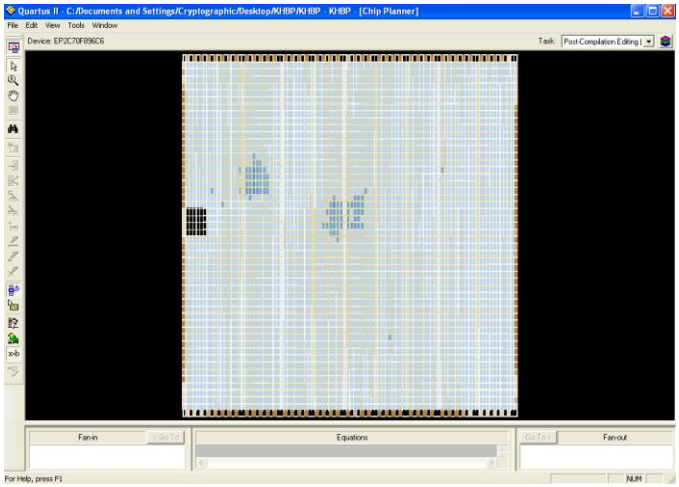


Figure 8. Floor-plan of power based key hopping implementation

VI. CONCLUSION

We have provided a brief discussion of the concept of power based key hopping and its hardware implementation. The details of the proposed procedure were discussed in sections iii and iv. The method uses a discrete form of signal analysis where the signal power is measured as the mean of the signal encountered. The power of a packet of binary sequence with a finite length can be determined by the number of ones in that packet. The proposed method utilizes four keys. Each two communicating entities maintain four secret keys, one of them will be an authentication key and the first key will be used as the default key at each time the communication session is started. When the power of ciphertext packet computed based on the number of ones equals a certain predefined value between the communicating entities, the same key will be

used to encrypt and decrypt the next packet. Otherwise the key is changed to the next key of the key set.

The hardware implementation uses a set of modules that were carried out applying VHDL and then joined together using the schematic editor. The resulting circuit provides a proof-of-concept FPGA implementation. It was shown that the worst case pin-to-pin delay is equal to 15.592 ns. Moreover, area and speed optimization were performed and it is shown that the worst case pin-to-pin delay is equal to 15.645 ns in the case of area optimization and 15.383 ns in speed optimization. Moreover, high fan-out reduces the usage of global interconnection resources. Therefore, the speed optimization decreases the use of global resources. This is clearly demonstrated in the synthesis report of our design. A comparison with other implementations is not relevant since this is the first time this power based key hopping is FPGA-implemented. This and other related effects will be dealt with in future development of the device.

REFERENCES

[1] W. Ying, "Key Hopping – A Security Enhancement Scheme for IEEE 802.11 WEP Standards," NextComm Inc, USA, 2002.

[2] K. Srinivasan, and S. Mitchell, "State Based Key Hop (SBKH) Protocol," Sixteenth International Conference on Wireless Communications Wireless 2004, Alberta, Canada, 2004.

[3] A. M. Kholaif, M. B. Fayek, H. S. Eissa, and H. A. Baraka, "DRKH: Dynamic Re-Keying with Key Hopping," McMaster University, Hamilton, Canada, 2005.

[4] H. Bli, and H. Salim, "Performance Enhancement of GSM Cellular Phone Network using Dynamic Frequency Hopping," Engineering & Technology Journal, Vol. 26, No.3 (2008), University of Technology, Baghdad, Iraq, 2008.

[5] V. Saxena, "K-Delta-1-Sigma Modulators for Wideband Analog-to-Digital Conversion," Electrical and Computer Engineering Department, Boise State University, Idaho, USA, 2010.

[6] P. P. Chu, RTL Hardware Design Using VHDL, John Wiley & Sons, Inc., New Jersey, 2006.

[7] E. O. Hwang, Digital Logic and Microprocessor Design with VHDL, La Sierra University, Riverside, California, USA, 2005.

[8] V. A. Pedroni, Circuit Design with VHDL, MIT Press, 2004.

[9] Altera's user-support site:
http://www.altera.com/support/examples/vhdl/vhdl.html

APPENDIX A

The analysis and synthesis report details

Family: Cyclone II
Device: EP2C70F896C6
Total logic elements: 887 out of 68,416 (1%)
Total combinational functions: 887
Logic element usage by number of LUT inputs
-- 4 input functions: 221
-- 3 input functions: 249
-- <=2 input functions: 417
Total registers: 80 out of 70,234 (< 1%)
Total pins: 513 out of 622 (82 %)
Total memory bits: 0 out of 1,152,000 (0 %)
Embedded Multiplier 9-bit elements: 0 out of 300 (0 %)
Total PLLs: 0 out of 4 (0 %)
Optimization Technique: Balanced
Maximum fan-out: 80
Total fan-out: 2721
Average fan-out: 1.83

Fitter Summary

Block interconnects: 1206 out of 197,592 (< 1%)
C16 interconnects: 387 out of 6,270 (6 %)
C4 interconnects: 1220 out of 123,120 (< 1%)
Direct links: 233 out of 197,592 (< 1%)
Global clocks: 1 out of 16 (6 %)
Local interconnects: 435 out of 68,416 (< 1%)
R24 interconnects: 511 out of 5,926 (9 %)
R4 interconnects: 1394 out of 167,484 (< 1%)
Nominal Core Voltage: 1.20 V
Low Junction Temperature: 0 °C
High Junction Temperature: 85 °C

The usage number of logic elements and their connections in the device can be changed depending on the optimization technique which is used for synthezing the power based key hopping. Table A1 and table A2 show the number of usage logic elements and the interconnections between them in Area, Speed, and Balanced optimization technique. We noticed that the total number of usage logic elements in the device increased in speed optimization technique comparing with balanced and area optimization techniques improving the rout-ability of the design through using more C4 interconnects, which are considered the main interconnects in the device, and applied 1.86 average fan-out. While area and balanced optimization techniques give less average fun-out 1.83.

Table A1. A synthesis comparison between optimization technique implementations of power based key hopping

	Balance	Area	Speed
Total logic elements	887	895	909
Total combinational functions	887	895	909
4 input functions	221	215	228
3 input functions	249	259	262
<=2 input functions	417	421	419
Total fan-out	2721	2735	2796
Average fan-out	1.83	1.83	1.86

Table A2. A fitter comparison between optimization technique implementations of power based key hopping

	Balance	Area	Speed
Block interconnects	1206	1201	1211
C16 interconnects	387	390	354
C4 interconnects	1220	1226	1291
Direct links	233	187	213
Global clocks	1	1	1
Local interconnects	435	451	451
R24 interconnects	511	615	550
R4 interconnects	1394	1439	1373

The delays comparison between optimization techniques was extracted from the timing reports of implementing area and speed optimization. Figure A.1 shows a comparison chart between various implementation delays.

- In area optimization
  - The worst case pin-to-pin delay was found to be equal to 15.645 ns from source pin "INPUT[23]" to destination pin "OUTPUT[23]".
  - The longest pin-to-register delay was 23.955 ns and the shortest pin-to-register delay was 8.878 ns. The longest register-to-pin delay was 9.705 ns and the longest register-to-register delay was 16.611 ns.
  - The worst case clock-to-output delay was 12.810 ns from clock "CLK2" to destination pin "OUTPUT[116]" through register "KH_Power:inst|pre_temp_key[26]".
  - Clock "CLK2" had internal fmax of 59.51 MHz (period = 16.803 ns) between source register "KH_Power:inst|IV[52]" and destination register "KH_Power:inst|pre_temp_key[161]".

- In speed optimization
  - The worst case pin-to-pin delay was found to be equal to 15.383 ns from source pin "INPUT[159]" to destination pin "OUTPUT[159]".
  - The longest pin-to-register delay was 24.002 ns and the shortest pin-to-register delay was 9.566 ns. The longest register-to-pin delay was 8.687 ns and the longest register-to-register delay was 17.476 ns.
  - The worst case clock-to-output delay was 11.770 ns from clock "CLK2" to destination pin "OUTPUT[104]" through register "KH_Power:inst|pre_temp_key[26]".
  - Clock "CLK2" had internal fmax of 56.37 MHz (period = 17.74 ns) between source register "KH_Power:inst|IV[38]" and destination register "KH_Power:inst|pre_temp_key[30]".

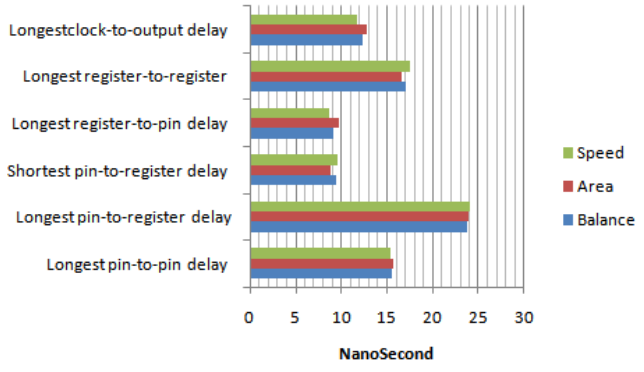


Figure A.1. Delays in our design of power based key hopping implementation

APPENDIX B

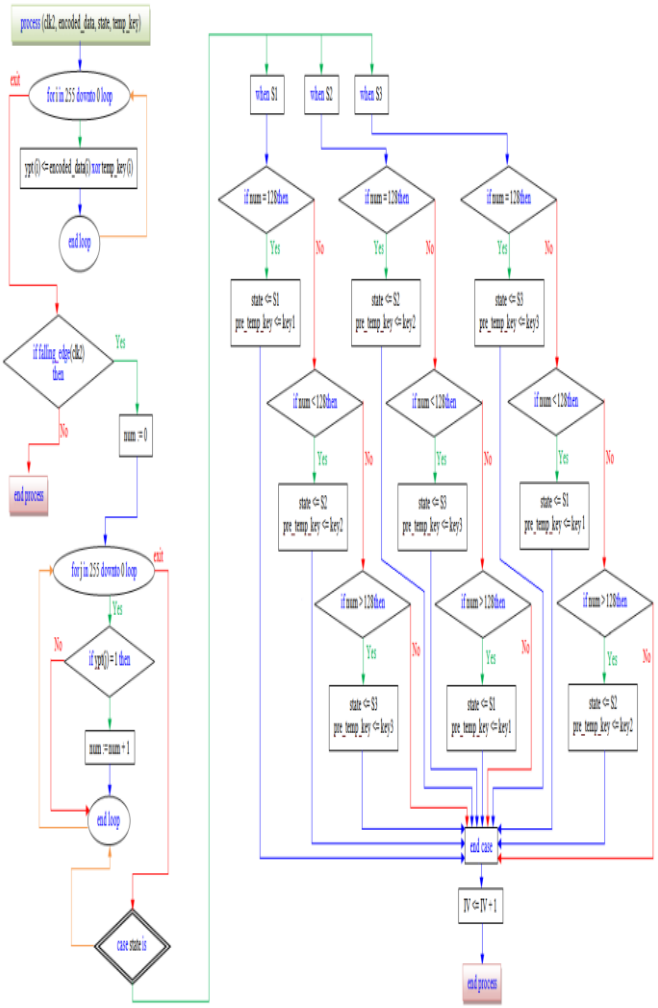
Sample VHDL code for 8-bit Power Based Key Hopping module

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;

ENTITY KH_Power IS
    port(clk2 : in std_logic;
         encoded_data : in std_logic_vector (7 downto 0);
         encrypted_data : out std_logic_vector (7 downto 0);
         key : out std_logic_vector (7 downto 0));
END KH_Power ;
ARCHITECTURE behavioral OF KH_Power IS
   signal IV   : std_logic_vector  (1 downto 0) := "00";
   signal key_authentication,  pre_temp_key :
                 std_logic_vector (5 downto 0) := "110011";
   signal key1 : std_logic_vector (5 downto 0) := "010101";
   signal key2 : std_logic_vector (5 downto 0) := "111000";
   signal key3 : std_logic_vector (5 downto 0) := "101010";
   signal temp_key : std_logic_vector (7 downto 0);
   signal ypt : std_logic_vector (7 downto 0);

   type state_type is (S1, S2, S3);
        signal state : state_type := S1;

BEGIN
   temp_key <= IV & pre_temp_key;
   process (clk2, encoded_data, state, temp_key)
      variable num : integer range 0 to 8;
   begin
              for i in 7 downto 0 loop
                   ypt (i) <= encoded_data(i) xor temp_key (i);
              end loop;

if falling_edge(clk2) then
             num := 0;
            for j in 7 downto 0 loop
                 if ypt(j) = '1' then num := num + 1;
                end if;
           end loop;

           case state is
              when S1 => if  num = 4    then
                           state <= S1; pre_temp_key <= key1;
                         elsif num < 4 then
                           state <= S2; pre_temp_key <= key2;
                         elsif num > 4 then
                           state <= S3; pre_temp_key <= key3;
                         end if;
              when S2 => if  num = 4    then
                           state <= S2; pre_temp_key <= key2;
                         elsif num < 4 then
                           state <= S3; pre_temp_key <= key3;
                         elsif num > 4 then
                           state <= S1; pre_temp_key <= key1;
```

```
                    end if;

        when S3 => if  num = 4    then
                      state <= S3; pre_temp_key <= key3;
                    elsif num < 4 then
                      state <= S1; pre_temp_key <= key1;
                    elsif num > 4 then
                      state <= S2; pre_temp_key <= key2;
                    end if;
      end case;
        IV <= IV + 1;
    end if;
  end process;
        temp_key <= IV & pre_temp_key;
        encrypted_data <= ypt;
        key <= temp_key;
END behavioral;
```

APPENDIX C

VHDL Flowchart for 256-bit Power Based Key Hopping module



AUTHORS' PROFILE

Magdy Saeb received the BSEE, School of Engineering, Cairo University, in 1974, the MSEE, and Ph.D. degrees in Electrical & Computer Engineering, University of California, Irvine, in 1981 and 1985, respectively. He was with Kaiser Aerospace and Electronics, Irvine California, and The Atomic Energy Establishment, Anshas, Egypt. Currently, he is a professor in the Department of Computer Engineering, Arab Academy for Science, Technology & Maritime Transport, Alexandria, Egypt; He was on-leave working as a principal researcher in the Malaysian Institute of Microelectronic Systems (MIMOS). His current research interests include Cryptography, FPGA Implementations of Cryptography and Steganography Data Security Techniques, Encryption Processors, Mobile Agent Security. www.magdysaeb.net.

Rabie Mahmoud received the BS. Degree, Faculty of Science, Tishreen University, Syria, in 2001, the MS. in Computational Science, Faculty of Science, Cairo University, Egypt in 2007. Currently, he is working on his Ph.D. Dissertation in the Department of Mathematics, Faculty of Science, Cairo University. His current interests include Image processing, Cryptography, FPGA Implementations of Cryptography and Data Security Techniques. rabiemah@yahoo.com