Construction of Full State from Half State of HC-128

Goutam Paul

Department of Computer Science & Engineering
Jadavpur University, Kolkata, India
http://www.jadavpur.edu/admin1/Faculty/GKP.htm

March 10, 2010

Seminar at
Coding & Cryptography Research Group (CCRG)
Division of Mathematical Sciences
School of Physical & Mathematical Sciences
Nanyang Technological University (NTU), Singapore

Joint Work with

Subhamoy Maitra and Shashwat Raizada

Indian Statistical Institute, Kolkata, India

Roadmap

1. Introduction
   - Background
   - Description of HC-128
   - Contribution

2. Reconstructing One Array from Another
   - First Phase: Complete P_N from P
   - Second Phase: Part of Q from P_N
   - Third Phase: Tail of Q from its Parts
   - Fourth Phase: Complete Q_N from Tail of Q
   - Fifth Phase (Verification) and Total Complexity

3. Proposal for Design Modification

Roadmap

1. Introduction
   - Background
   - Description of HC-128
   - Contribution
2. Reconstructing One Array from Another
   - First Phase: Complete P_N from P
   - Second Phase: Part of Q from P_N
   - Third Phase: Tail of Q from its Parts
   - Fourth Phase: Complete Q_N from Tail of Q
   - Fifth Phase (Verification) and Total Complexity
3. Proposal for Design Modification

Basics of Stream Cipher

- Symmetric Key Cryptosystem, both sender and the receiver has the same key.
- Encryption: $C_i = M_i \oplus K_i$, Decryption: $M_i = C_i \oplus K_i$.
- The best possible scenario: the sender and receiver have a long common stream of bits that they have generated sitting in the same table and tossing an unbiased coin.
  - Pros: never used repeatedly (One Time Pad).
  - Cons: practically not possible.
- Solution: a Pseudorandom generator based on a seed (secret key).

Basics of Stream Cipher

- Symmetric Key Cryptosystem, both sender and the receiver has the same key.
- Encryption: $C_i = M_i \oplus K_i$, Decryption: $M_i = C_i \oplus K_i$.
- The best possible scenario: the sender and receiver have a long common stream of bits that they have generated sitting in the same table and tossing an unbiased coin.
    - Pros: never used repeatedly (One Time Pad).
    - Cons: practically not possible.
- Solution: a Pseudorandom generator based on a seed (secret key).

Basics of Stream Cipher

- Symmetric Key Cryptosystem, both sender and the receiver has the same key.
- Encryption: $C_i = M_i \oplus K_i$, Decryption: $M_i = C_i \oplus K_i$.
- The best possible scenario: the sender and receiver have a long common stream of bits that they have generated sitting in the same table and tossing an unbiased coin.
  - Pros: never used repeatedly (One Time Pad).
  - Cons: practically not possible.
- Solution: a Pseudorandom generator based on a seed (secret key).

Basics of Stream Cipher

- Symmetric Key Cryptosystem, both sender and the receiver has the same key.
- Encryption: $C_i = M_i \oplus K_i$, Decryption: $M_i = C_i \oplus K_i$.
- The best possible scenario: the sender and receiver have a long common stream of bits that they have generated sitting in the same table and tossing an unbiased coin.
  - Pros: never used repeatedly (One Time Pad).
  - Cons: practically not possible.
- Solution: a Pseudorandom generator based on a seed (secret key).

Basics of Stream Cipher

- Symmetric Key Cryptosystem, both sender and the receiver has the same key.
- Encryption: $C_i = M_i \oplus K_i$, Decryption: $M_i = C_i \oplus K_i$.
- The best possible scenario: the sender and receiver have a long common stream of bits that they have generated sitting in the same table and tossing an unbiased coin.
  - Pros: never used repeatedly (One Time Pad).
  - Cons: practically not possible.
- Solution: a Pseudorandom generator based on a seed (secret key).

Basics of Stream Cipher

- Symmetric Key Cryptosystem, both sender and the receiver has the same key.
- Encryption: $C_i = M_i \oplus K_i$, Decryption: $M_i = C_i \oplus K_i$.
- The best possible scenario: the sender and receiver have a long common stream of bits that they have generated sitting in the same table and tossing an unbiased coin.
  - Pros: never used repeatedly (One Time Pad).
  - Cons: practically not possible.
- Solution: a Pseudorandom generator based on a seed (secret key).

The eSTREAM Project

- A project of ECRYPT, a Network of Excellence within the Information Societies Technology (IST) Programme of the European Commission.
- An effort to get some secure stream ciphers satisfying the current requirements.
- This multi-year effort running from 2004 to 2008 has identified a portfolio of promising new stream ciphers.
- `http://www.ecrypt.eu.org/stream`

The eSTREAM Project

- A project of ECRYPT, a Network of Excellence within the Information Societies Technology (IST) Programme of the European Commission.
- An effort to get some secure stream ciphers satisfying the current requirements.
- This multi-year effort running from 2004 to 2008 has identified a portfolio of promising new stream ciphers.
- `http://www.ecrypt.eu.org/stream`

The eSTREAM Project

- A project of ECRYPT, a Network of Excellence within the Information Societies Technology (IST) Programme of the European Commission.
- An effort to get some secure stream ciphers satisfying the current requirements.
- This multi-year effort running from 2004 to 2008 has identified a portfolio of promising new stream ciphers.
- http://www.ecrypt.eu.org/stream

The eSTREAM Project

- A project of ECRYPT, a Network of Excellence within the Information Societies Technology (IST) Programme of the European Commission.
- An effort to get some secure stream ciphers satisfying the current requirements.
- This multi-year effort running from 2004 to 2008 has identified a portfolio of promising new stream ciphers.
- `http://www.ecrypt.eu.org/stream`

The eSTREAM Portfolio (Revision 1, September 2008)

Profile 1 (SW)	Profile 2 (HW)
HC-128	Grain v1
Rabbit	MICKEY v2
Salsa20/12	Trivium
SOSEMANUK	

Roadmap

HC-128

- Designed by Hongjun Wu.
- A scaled down version of HC-256 that has been presented in FSE 2004.
- A synchronous software stream cipher with 32-bit word output in each step.
- Intellectual Property: free for any use.
- Available at `http://www.ecrypt.eu.org/stream/hcp3.html`.
- 128-bit secret key.

Notations

- $+$: $x + y$ means $x + y \bmod 2^{32}$, where $0 \leq x < 2^{32}$ and $0 \leq y < 2^{32}$.
- $\boxminus$: $x \boxminus y$ means $x - y \bmod 512$.
- $\oplus$: bit-wise exclusive OR.
- $\|$: concatenation.
- $\gg$: right shift operator. $x \gg n$ means x being right shifted n bits.
- $\ll$: left shift operator. $x \ll n$ means x being left shifted n bits.
- $\ggg$: right rotation operator. $x \ggg n$ means $((x \gg n) \oplus (x \ll (32 - n))$, where $0 \leq n < 32$, $0 \leq x < 2^{32}$.
- $\lll$: left rotation operator. $x \lll n$ means $((x \ll n) \oplus (x \gg (32 - n))$, where $0 \leq n < 32$, $0 \leq x < 2^{32}$.

Data Structures

- Two tables P and Q, each with 512 many 32-bit elements are used as internal states of HC-128.
- A 128-bit key array $K[0, \ldots, 3]$ and a 128-bit initialization vector $IV[0, \ldots, 3]$ are used, where each entry of the array is a 32-bit element.
- s_t denotes the keystream word generated at the t-th step, $t = 0, 1, 2, \ldots$.

Data Structures

- Two tables P and Q, each with 512 many 32-bit elements are used as internal states of HC-128.
- A 128-bit key array $K[0, \ldots, 3]$ and a 128-bit initialization vector $IV[0, \ldots, 3]$ are used, where each entry of the array is a 32-bit element.
- s_t denotes the keystream word generated at the t-th step, $t = 0, 1, 2, \ldots$.

Data Structures

- Two tables P and Q, each with 512 many 32-bit elements are used as internal states of HC-128.
- A 128-bit key array $K[0, \ldots, 3]$ and a 128-bit initialization vector $IV[0, \ldots, 3]$ are used, where each entry of the array is a 32-bit element.
- s_t denotes the keystream word generated at the t-th step, $t = 0, 1, 2, \ldots$.

Functions

- $f_1(x) = (x \ggg 7) \oplus (x \ggg 18) \oplus (x \gg 3)$.
- $f_2(x) = (x \ggg 17) \oplus (x \ggg 19) \oplus (x \gg 10)$.
- $g_1(x, y, z) = ((x \ggg 10) \oplus (z \ggg 23)) + (y \ggg 8)$.
- $g_2(x, y, z) = ((x \lll 10) \oplus (z \lll 23)) + (y \lll 8)$.
- $h_1(x) = Q[x^{(0)}] + Q[256 + x^{(2)}]$.
- $h_2(x) = P[x^{(0)}] + P[256 + x^{(2)}]$

Here $x = x^{(3)} \| x^{(2)} \| x^{(1)} \| x^{(0)}$, x is a 32-bit word and $x^{(0)}$ (least significant byte) , $x^{(1)}$, $x^{(2)}$ and $x^{(3)}$ (most significant byte) are four bytes.

Functions

- $f_1(x) = (x \ggg 7) \oplus (x \ggg 18) \oplus (x \gg 3)$.
- $f_2(x) = (x \ggg 17) \oplus (x \ggg 19) \oplus (x \gg 10)$.
- $g_1(x, y, z) = ((x \ggg 10) \oplus (z \ggg 23)) + (y \ggg 8)$.
- $g_2(x, y, z) = ((x \lll 10) \oplus (z \lll 23)) + (y \lll 8)$.
- $h_1(x) = Q[x^{(0)}] + Q[256 + x^{(2)}]$.
- $h_2(x) = P[x^{(0)}] + P[256 + x^{(2)}]$

Here $x = x^{(3)} \| x^{(2)} \| x^{(1)} \| x^{(0)}$, x is a 32-bit word and $x^{(0)}$ (least significant byte) , $x^{(1)}$, $x^{(2)}$ and $x^{(3)}$ (most significant byte) are four bytes.

Functions

- $f_1(x) = (x \ggg 7) \oplus (x \ggg 18) \oplus (x \gg 3)$.
- $f_2(x) = (x \ggg 17) \oplus (x \ggg 19) \oplus (x \gg 10)$.
- $g_1(x, y, z) = ((x \ggg 10) \oplus (z \ggg 23)) + (y \ggg 8)$.
- $g_2(x, y, z) = ((x \lll 10) \oplus (z \lll 23)) + (y \lll 8)$.
- $h_1(x) = Q[x^{(0)}] + Q[256 + x^{(2)}]$.
- $h_2(x) = P[x^{(0)}] + P[256 + x^{(2)}]$

Here $x = x^{(3)} \| x^{(2)} \| x^{(1)} \| x^{(0)}$, x is a 32-bit word and $x^{(0)}$ (least significant byte) , $x^{(1)}, x^{(2)}$ and $x^{(3)}$ (most significant byte) are four bytes.

Functions

- $f_1(x) = (x \ggg 7) \oplus (x \ggg 18) \oplus (x \gg 3)$.
- $f_2(x) = (x \ggg 17) \oplus (x \ggg 19) \oplus (x \gg 10)$.
- $g_1(x, y, z) = ((x \ggg 10) \oplus (z \ggg 23)) + (y \ggg 8)$.
- $g_2(x, y, z) = ((x \lll 10) \oplus (z \lll 23)) + (y \lll 8)$.
- $h_1(x) = Q[x^{(0)}] + Q[256 + x^{(2)}]$.
- $h_2(x) = P[x^{(0)}] + P[256 + x^{(2)}]$

Here $x = x^{(3)} \| x^{(2)} \| x^{(1)} \| x^{(0)}$, x is a 32-bit word and $x^{(0)}$ (least significant byte) , $x^{(1)}, x^{(2)}$ and $x^{(3)}$ (most significant byte) are four bytes.

Key and IV Setup

Let $K[0, \ldots, 3]$ be the secret key and $IV[0, \ldots, 3]$ be the initialization vector. Let $K[i + 4] = K[i]$ and $IV[i + 4] = IV[i]$ for $0 \le i \le 3$.

The key and IV are expanded into an array $W[0, \ldots, 1279]$ as follows.

$$W[i] = \begin{cases} K[i] & 0 \le i \le 7; \\ IV[i-8] & 8 \le i \le 15; \\ f_2(W[i-2]) + W[i-7] + \\ f_1(W[i-15]) + W[i-16] + i & \\ & 16 \le i \le 1279. \end{cases}$$

Key and IV Setup

Let $K[0,...,3]$ be the secret key and $IV[0,...,3]$ be the initialization vector. Let $K[i+4] = K[i]$ and $IV[i+4] = IV[i]$ for $0 \leq i \leq 3$.
The key and IV are expanded into an array $W[0,...,1279]$ as follows.

$$W[i] = \begin{cases} K[i] & 0 \leq i \leq 7; \\ IV[i-8] & 8 \leq i \leq 15; \\ f_2(W[i-2]) + W[i-7] + \\ f_1(W[i-15]) + W[i-16] + i & \\ & 16 \leq i \leq 1279. \end{cases}$$

Key and IV setup (Contd.)

Update the tables P and Q with the array W as follows.

$$P[i] = W[i + 256], \text{ for } 0 \leq i \leq 511$$
$$Q[i] = W[i + 768], \text{ for } 0 \leq i \leq 511$$

Run the cipher 1024 steps and use the outputs to replace the table elements as follows.

For $i = 0$ to 511, do
$P[i] = (P[i] + g_1(P[i \boxminus 3], P[i \boxminus 10], P[i \boxminus 511])) \oplus h_1(P[i \boxminus 12]);$

For $i = 0$ to 511, do
$Q[i] = (Q[i] + g_2(Q[i \boxminus 3], Q[i \boxminus 10], Q[i \boxminus 511])) \oplus h_2(Q[i \boxminus 12]);$

Key and IV setup (Contd.)

Update the tables P and Q with the array W as follows.

$$P[i] = W[i + 256], \text{ for } 0 \leq i \leq 511$$
$$Q[i] = W[i + 768], \text{ for } 0 \leq i \leq 511$$

Run the cipher 1024 steps and use the outputs to replace the table elements as follows.

For $i = 0$ to 511, do
$P[i] = (P[i] + g_1(P[i \boxminus 3], P[i \boxminus 10], P[i \boxminus 511])) \oplus h_1(P[i \boxminus 12]);$

For $i = 0$ to 511, do
$Q[i] = (Q[i] + g_2(Q[i \boxminus 3], Q[i \boxminus 10], Q[i \boxminus 511])) \oplus h_2(Q[i \boxminus 12]);$

The Keystream Generation Algorithm

$i = 0;$
repeat until enough keystream bits are generated {
    $j = i \bmod 512;$
    if $(i \bmod 1024) < 512\{$
        $P[j] = P[j] + g_1(P[j \boxminus 3], P[j \boxminus 10], P[j \boxminus 511]);$
        $s_i = h_1(P[j \boxminus 12]) \oplus P[j];$
    }
    else {
        $Q[j] = Q[j] + g_2(Q[j \boxminus 3], Q[j \boxminus 10], Q[j \boxminus 511]);$
        $s_i = h_2(Q[j \boxminus 12]) \oplus Q[j];$
    }
    end-if
    $i = i + 1;$
} end-repeat

Roadmap

Existing Results

1. Wu, the designer of HC-128 himself, presented a distinguisher that requires 2^{156} keystream words.
2. Dunkelman in the eStream discussion forum: `http://www.ecrypt.eu.org/stream/phorum/read.php?1,1143` (dated November 14, 2007): $Prob(s_j \oplus s_{j+1} = P[j] \oplus P[j+1]) \approx 2^{-16}$.
3. *Some Observations on HC-128*, Subhamoy Maitra, Goutam Paul and Shashwat Raizada, Proc. WCC 2009, pages 527-539 (extended version to appear in *Designs, Codes and Cryptography* Journal).

Existing Results

1. Wu, the designer of HC-128 himself, presented a distinguisher that requires 2^{156} keystream words.
2. Dunkelman in the eStream discussion forum: `http://www.ecrypt.eu.org/stream/phorum/read.php?1,1143` (dated November 14, 2007): $Prob(s_j \oplus s_{j+1} = P[j] \oplus P[j+1]) \approx 2^{-16}$.
3. *Some Observations on HC-128*, Subhamoy Maitra, Goutam Paul and Shashwat Raizada, Proc. WCC 2009, pages 527-539 (extended version to appear in *Designs, Codes and Cryptography* Journal).

Existing Results

1. Wu, the designer of HC-128 himself, presented a distinguisher that requires 2^{156} keystream words.
2. Dunkelman in the eStream discussion forum: `http://www.ecrypt.eu.org/stream/phorum/read.php?1,1143` (dated November 14, 2007): $Prob(s_j \oplus s_{j+1} = P[j] \oplus P[j+1]) \approx 2^{-16}$.
3. *Some Observations on HC-128*, Subhamoy Maitra, Goutam Paul and Shashwat Raizada, Proc. WCC 2009, pages 527-539 (extended version to appear in *Designs, Codes and Cryptography* Journal).

Motivation for the Present Analysis

- Many attacks in the stream cipher domain assume knowledge of partial state information.
- State recovery attacks, on the other hand, assume knowledge of certain keystream bits and reconstruct the full internal state.
- Neither any partial state exposure attack nor any state recovery attack on HC-128 have been reported so far.

Motivation for the Present Analysis

- Many attacks in the stream cipher domain assume knowledge of partial state information.
- State recovery attacks, on the other hand, assume knowledge of certain keystream bits and reconstruct the full internal state.
- Neither any partial state exposure attack nor any state recovery attack on HC-128 have been reported so far.

Motivation for the Present Analysis

- Many attacks in the stream cipher domain assume knowledge of partial state information.
- State recovery attacks, on the other hand, assume knowledge of certain keystream bits and reconstruct the full internal state.
- Neither any partial state exposure attack nor any state recovery attack on HC-128 have been reported so far.

Main Idea

- Keystream is generated in *blocks* of 512 words
- Consider four consecutive blocks B_1, B_2, B_3, B_4.

Block B_1:	Block B_2:	Block B_3:
P unchanged, Q updated. (Q denotes the updated array)	P updated to P_N, Q unchanged.	P_N unchanged, Q updated to Q_N.

- Block B_4, that is not shown in the diagram, would only be used for verifying if our reconstruction is correct or not.
- Our algorithm, given the half state P, constructs the full state (P_N, Q_N).

(Note that we would use notation $s_{b,i}$ to denote the i-th keystream word generated in block B_b, $1 \le b \le 4$, $0 \le i \le 511$.)

Main Idea

- Keystream is generated in *blocks* of 512 words
- Consider four consecutive blocks B_1, B_2, B_3, B_4.

Block B_1: P unchanged, Q updated. (Q denotes the updated array)	**Block B_2:** P updated to P_N, Q unchanged.	**Block B_3:** P_N unchanged, Q updated to Q_N.

- Block B_4, that is not shown in the diagram, would only be used for verifying if our reconstruction is correct or not.
- Our algorithm, given the half state P, constructs the full state (P_N, Q_N).

(Note that we would use notation $s_{b,i}$ to denote the i-th keystream word generated in block B_b, $1 \leq b \leq 4$, $0 \leq i \leq 511$.)

Main Idea

- Keystream is generated in *blocks* of 512 words
- Consider four consecutive blocks B_1, B_2, B_3, B_4.

Block B_1: P unchanged, Q updated. (Q denotes the updated array)	**Block B_2:** P updated to P_N, Q unchanged.	**Block B_3:** P_N unchanged, Q updated to Q_N.

- Block B_4, that is not shown in the diagram, would only be used for verifying if our reconstruction is correct or not.
- Our algorithm, given the half state P, constructs the full state (P_N, Q_N).

(Note that we would use notation $s_{b,i}$ to denote the i-th keystream word generated in block B_b, $1 \leq b \leq 4$, $0 \leq i \leq 511$.)

Main Idea

- Keystream is generated in *blocks* of 512 words
- Consider four consecutive blocks B_1, B_2, B_3, B_4.

Block B_1: P unchanged, Q updated. (Q denotes the updated array)	**Block B_2:** P updated to P_N, Q unchanged.	**Block B_3:** P_N unchanged, Q updated to Q_N.

- Block B_4, that is not shown in the diagram, would only be used for verifying if our reconstruction is correct or not.
- Our algorithm, given the half state P, constructs the full state (P_N, Q_N).

(Note that we would use notation $s_{b,i}$ to denote the i-th keystream word generated in block B_b, $1 \leq b \leq 4$, $0 \leq i \leq 511$.)

Roadmap

Roadmap

Update of P in Block B_2

- Update of P (or Q) depends only on itself.

$$P_N[i] = \begin{cases} P[i] + g_1(P[509+i], P[502+i], P[i+1]), & \text{for } 0 \le i \le 2; \\ P[i] + g_1(P_N[i-3], P[502+i], P[i+1]), & \text{for } 3 \le i \le 9; \\ P[i] + g_1(P_N[i-3], P_N[i-10], P[i+1]), & \text{for } 10 \le i \le 510; \\ P[i] + g_1(P_N[i-3], P_N[i-10], P_N[i-511]), & \text{for } i = 511. \end{cases}$$

- If one knows the 512 words of P (or Q) corresponding to any one block, then one can easily derive the complete P (or Q) array corresponding to any subsequent block.

Update of P in Block B_2

- Update of P (or Q) depends only on itself.

$$P_N[i] = \begin{cases} P[i] + g_1(P[509+i], P[502+i], P[i+1]), & \text{for } 0 \le i \le 2; \\ P[i] + g_1(P_N[i-3], P[502+i], P[i+1]), & \text{for } 3 \le i \le 9; \\ P[i] + g_1(P_N[i-3], P_N[i-10], P[i+1]), & \text{for } 10 \le i \le 510; \\ P[i] + g_1(P_N[i-3], P_N[i-10], P_N[i-511]), & \text{for } i = 511. \end{cases}$$

- If one knows the 512 words of P (or Q) corresponding to any one block, then one can easily derive the complete P (or Q) array corresponding to any subsequent block.

Roadmap

Keystream Generation in Block B_2

$$s_{2,i} = \begin{cases} h_1(P[500+i]) \oplus P_N[i], & \text{for } 0 \le i \le 11; \\ h_1(P_N[i-12]) \oplus P_N[i], & \text{for } 12 \le i \le 511. \end{cases}$$

Since $h_1(x) = Q[x^{(0)}] + Q[256 + x^{(2)}]$, we can rewrite the above as

$$Q[l_i] + Q[u_i] = s_{2,i} \oplus P_N[i]$$

$$\left.\begin{array}{ll} \text{where for } 0 \le i \le 11, & l_i = (P[500+i])^{(0)} \\ \text{and} & u_i = 256 + (P[500+i])^{(2)} \end{array}\right\}$$

$$\left.\begin{array}{ll} \text{and for } 12 \le i \le 511, & l_i = (P_N[i-12])^{(0)} \\ \text{and} & u_i = 256 + (P_N[i-12])^{(2)}. \end{array}\right\}$$

- It can be proved that unique solution does not exist.

Keystream Generation in Block B_2

$$s_{2,i} = \begin{cases} h_1(P[500+i]) \oplus P_N[i], & \text{for } 0 \le i \le 11; \\ h_1(P_N[i-12]) \oplus P_N[i], & \text{for } 12 \le i \le 511. \end{cases}$$

Since $h_1(x) = Q[x^{(0)}] + Q[256 + x^{(2)}]$, we can rewrite the above as

$$Q[l_i] + Q[u_i] = s_{2,i} \oplus P_N[i]$$

$$\left.\begin{array}{ll} \text{where for } 0 \le i \le 11, & l_i = (P[500+i])^{(0)} \\ \text{and} & u_i = 256 + (P[500+i])^{(2)} \end{array}\right\}$$

$$\left.\begin{array}{ll} \text{and for } 12 \le i \le 511, & l_i = (P_N[i-12])^{(0)} \\ \text{and} & u_i = 256 + (P_N[i-12])^{(2)}. \end{array}\right\}$$

- It can be proved that unique solution does not exist.

Keystream Generation in Block B_2

$$s_{2,i} = \begin{cases} h_1(P[500+i]) \oplus P_N[i], & \text{for } 0 \le i \le 11; \\ h_1(P_N[i-12]) \oplus P_N[i], & \text{for } 12 \le i \le 511. \end{cases}$$

Since $h_1(x) = Q[x^{(0)}] + Q[256 + x^{(2)}]$, we can rewrite the above as

$$Q[l_i] + Q[u_i] = s_{2,i} \oplus P_N[i]$$

$$\left.\begin{array}{ll} \text{where for } 0 \le i \le 11, & l_i = (P[500+i])^{(0)} \\ \text{and} & u_i = 256 + (P[500+i])^{(2)} \end{array}\right\}$$

$$\left.\begin{array}{ll} \text{and for } 12 \le i \le 511, & l_i = (P_N[i-12])^{(0)} \\ \text{and} & u_i = 256 + (P_N[i-12])^{(2)}. \end{array}\right\}$$

- It can be proved that unique solution does not exist.

Solution Trick: Resort to Random Bipartite Graphs

The above system of 512 equations can be represented in the form of a bipartite graph $G = (V_1, V_2, E)$, where $V_1 = \{0, \ldots, 255\}$, $V_2 = \{256, \ldots, 511\}$ and for $l_i \in V_1$ and $u_i \in V_2$, $\exists$ an edge $\{l_i, u_i\} \in E$ if and only if the sum $Q[l_i] + Q[u_i]$ is known. Thus, $|E| = 512$ (counting repeated edges, if any).

We call such a graph G with the vertices as the indices of one internal array of HC-128 the *index graph* of the state of HC-128.

Lemma

Let M be the size of the largest connected component of the index graph G corresponding to block B_2. Then M out of 512 words of the array Q can be derived in 2^{32} search complexity.

Solution Trick: Resort to Random Bipartite Graphs

The above system of 512 equations can be represented in the form of a bipartite graph $G = (V_1, V_2, E)$, where $V_1 = \{0, \ldots, 255\}$, $V_2 = \{256, \ldots, 511\}$ and for $l_i \in V_1$ and $u_i \in V_2$, $\exists$ an edge $\{l_i, u_i\} \in E$ if and only if the sum $Q[l_i] + Q[u_i]$ is known. Thus, $|E| = 512$ (counting repeated edges, if any). We call such a graph G with the vertices as the indices of one internal array of HC-128 the *index graph* of the state of HC-128.

Lemma

Let M be the size of the largest connected component of the index graph G corresponding to block B_2. Then M out of 512 words of the array Q can be derived in 2^{32} search complexity.

Solution Trick: Resort to Random Bipartite Graphs

The above system of 512 equations can be represented in the form of a bipartite graph $G = (V_1, V_2, E)$, where $V_1 = \{0, \ldots, 255\}$, $V_2 = \{256, \ldots, 511\}$ and for $l_i \in V_1$ and $u_i \in V_2$, $\exists$ an edge $\{l_i, u_i\} \in E$ if and only if the sum $Q[l_i] + Q[u_i]$ is known. Thus, $|E| = 512$ (counting repeated edges, if any). We call such a graph G with the vertices as the indices of one internal array of HC-128 the *index graph* of the state of HC-128.

Lemma

Let M be the size of the largest connected component of the index graph G corresponding to block B_2. Then M out of 512 words of the array Q can be derived in 2^{32} search complexity.

The Giant Component

Consider a bipartite graph $G(n_1, n_2, T)$, formed by T independent trials, each of which joins two vertices chosen independently of each other from the distinct parts.

W.l.o.g., let $n_1 \geq n_2$, $\alpha = \frac{n_2}{n_1}$, $\beta = (1 - \alpha) \ln n_1$, $n = n_1 + n_2$. Let $\xi_{n_1,n_2,T}$ and $\chi_{n_1,n_2,T}$ respectively denote the number of isolated vertices and the number of connected components in $G(n_1, n_2, T)$.

The Giant Component

Consider a bipartite graph $G(n_1, n_2, T)$, formed by T independent trials, each of which joins two vertices chosen independently of each other from the distinct parts.

W.l.o.g., let $n_1 \geq n_2$, $\alpha = \frac{n_2}{n_1}$, $\beta = (1 - \alpha) \ln n_1$, $n = n_1 + n_2$. Let $\xi_{n_1,n_2,T}$ and $\chi_{n_1,n_2,T}$ respectively denote the number of isolated vertices and the number of connected components in $G(n_1, n_2, T)$.

The Giant Component (Contd.)

A. I. Saltykov. The number of components in a random bipartite graph. *Diskretnaya Matematika*, vol. 7, no. 4, 1995, pages 86-94.

Proposition

If $n \to \infty$ and $(1+\alpha)T = n \ln n + Xn + o(n)$, where X is a fixed number, then Prob $\left(\chi_{n_1,n_2,T} = \xi_{n_1,n_2,T} + 1\right) \to 1$ and for any $k = 0, 1, 2, \ldots,$ Prob $\left(\xi_{n_1,n_2,T} = k\right) - \frac{\lambda^k e^{-\lambda}}{k!} \to 0$, where $\lambda = \frac{e^{-X}(1+e^{-\beta})}{1+\alpha}$.

The Giant Component (Contd.)

A. I. Saltykov. The number of components in a random bipartite graph. *Diskretnaya Matematika*, vol. 7, no. 4, 1995, pages 86-94.

Proposition

If $n \to \infty$ *and* $(1+\alpha)T = n \ln n + Xn + o(n)$, *where* X *is a fixed number, then Prob* $\left(\chi_{n_1,n_2,T} = \xi_{n_1,n_2,T} + 1\right) \to 1$ *and for any* $k = 0, 1, 2, \ldots,$ *Prob* $\left(\xi_{n_1,n_2,T} = k\right) - \frac{\lambda^k e^{-\lambda}}{k!} \to 0$, *where* $\lambda = \frac{e^{-X}(1+e^{-\beta})}{1+\alpha}$.

Coming Back to Our Case

For our index graph model, $n_1 = |V_1|$, $n_2 = |V_2|$ and $T = |E|$.

Corollary

If M is the size of the largest component of the index graph G, then the mean and standard deviation of M is respectively given by $E(M) \approx 442.59$ and $sd(M) \approx 8.33$.

Simulations with 10 million trials, each time with 1024 consecutive words of keystream generation for the complete arrays P and Q, gives the average and standard deviation of M are found to be $407.91 \approx 408$ and 9.17 respectively.

Coming Back to Our Case

For our index graph model, $n_1 = |V_1|$, $n_2 = |V_2|$ and $T = |E|$.

Corollary

If M is the size of the largest component of the index graph G, then the mean and standard deviation of M is respectively given by $E(M) \approx 442.59$ and $sd(M) \approx 8.33$.

Simulations with 10 million trials, each time with 1024 consecutive words of keystream generation for the complete arrays P and Q, gives the average and standard deviation of M are found to be $407.91 \approx 408$ and 9.17 respectively.

Coming Back to Our Case

For our index graph model, $n_1 = |V_1|$, $n_2 = |V_2|$ and $T = |E|$.

Corollary

If M is the size of the largest component of the index graph G, then the mean and standard deviation of M is respectively given by $E(M) \approx 442.59$ and $sd(M) \approx 8.33$.

Simulations with 10 million trials, each time with 1024 consecutive words of keystream generation for the complete arrays P and Q, gives the average and standard deviation of M are found to be $407.91 \approx 408$ and 9.17 respectively.

Roadmap

1. Introduction
   - Background
   - Description of HC-128
   - Contribution
2. Reconstructing One Array from Another
   - First Phase: Complete P_N from P
   - Second Phase: Part of Q from P_N
   - Third Phase: Tail of Q from its Parts
   - Fourth Phase: Complete Q_N from Tail of Q
   - Fifth Phase (Verification) and Total Complexity
3. Proposal for Design Modification

Keystream Generation in Block B_1

$s_{i,1} = h_2(Q[i-12]) \oplus Q[i]$, for $12 \leq i \leq 511$.

Written in another way, it becomes

$Q[i] = s_{1,i} \oplus \left(P\left[(Q[i-12])^{(0)}\right] + P\left[256 + (Q[i-12])^{(2)}\right]\right)$.

Thus,

Theorem (Propagation Theorem)

If Q[y] is known for some y in $[0, 499]$, then $m = \lfloor \frac{511-y}{12} \rfloor$ more words of Q, namely, $Q[y+12], Q[y+24], \ldots, Q[y+12m]$, can all be determined from $Q[y]$ in a time complexity that is linear in the size of Q.

Keystream Generation in Block B_1

$s_{i,1} = h_2(Q[i-12]) \oplus Q[i]$, for $12 \le i \le 511$.
Written in another way, it becomes
$Q[i] = s_{1,i} \oplus \left(P\left[(Q[i-12])^{(0)}\right] + P\left[256 + (Q[i-12])^{(2)}\right]\right)$.
Thus,

Theorem (Propagation Theorem)

If Q[y] is known for some y in $[0, 499]$, then $m = \lfloor \frac{511-y}{12} \rfloor$ more words of Q, namely, $Q[y+12], Q[y+24], \ldots, Q[y+12m]$, can all be determined from $Q[y]$ in a time complexity that is linear in the size of Q.

Keystream Generation in Block B_1

$s_{i,1} = h_2(Q[i-12]) \oplus Q[i]$, for $12 \leq i \leq 511$.
Written in another way, it becomes
$Q[i] = s_{1,i} \oplus \left(P\left[(Q[i-12])^{(0)}\right] + P\left[256 + (Q[i-12])^{(2)}\right]\right)$.
Thus,

Theorem (Propagation Theorem)

If Q[y] is known for some y in $[0, 499]$*, then* $m = \lfloor \frac{511-y}{12} \rfloor$ *more words of Q, namely,* $Q[y+12], Q[y+24], \ldots, Q[y+12m]$*, can all be determined from* $Q[y]$ *in a time complexity that is linear in the size of Q.*

Propagation Exhausts 12 Words in Tail

Theorem

After the Third Phase, the expected number of unknown words amongst $Q[500], Q[501], \ldots, Q[511]$ is approximately $8 \cdot (1 - \frac{43}{512})^M + 4 \cdot (1 - \frac{42}{512})^M$, where M is the size of the largest component of the index graph G.

Substituting M by its theoretical mean estimate 443 as well as by its empirical mean estimate 408 yields $E(Y) \approx 0$.
In fact, for any $M > 200$, the expression $(1 - \frac{43}{512})^M + 4 \cdot (1 - \frac{42}{512})^M$ for $E(Y)$ becomes vanishingly small.
Our experimental data also supports that in every instance, none of the words $Q[500], Q[501], \ldots, Q[511]$ remains unknown.

Propagation Exhausts 12 Words in Tail

Theorem

After the Third Phase, the expected number of unknown words amongst $Q[500], Q[501], \ldots, Q[511]$ *is approximately* $8 \cdot (1 - \frac{43}{512})^M + 4 \cdot (1 - \frac{42}{512})^M$, *where M is the size of the largest component of the index graph G.*

Substituting M by its theoretical mean estimate 443 as well as by its empirical mean estimate 408 yields $E(Y) \approx 0$.

In fact, for any $M > 200$, the expression $(1 - \frac{43}{512})^M + 4 \cdot (1 - \frac{42}{512})^M$ for $E(Y)$ becomes vanishingly small.

Our experimental data also supports that in every instance, none of the words $Q[500], Q[501], \ldots, Q[511]$ remains unknown.

Propagation Exhausts 12 Words in Tail

Theorem

After the Third Phase, the expected number of unknown words amongst $Q[500], Q[501], \ldots, Q[511]$ *is approximately* $8 \cdot (1 - \frac{43}{512})^M + 4 \cdot (1 - \frac{42}{512})^M$*, where* M *is the size of the largest component of the index graph* G*.*

Substituting M by its theoretical mean estimate 443 as well as by its empirical mean estimate 408 yields $E(Y) \approx 0$.
In fact, for any $M > 200$, the expression $(1 - \frac{43}{512})^M + 4 \cdot (1 - \frac{42}{512})^M$ for $E(Y)$ becomes vanishingly small.
Our experimental data also supports that in every instance, none of the words $Q[500], Q[501], \ldots, Q[511]$ remains unknown.

Propagation Exhausts 12 Words in Tail

Theorem

After the Third Phase, the expected number of unknown words amongst $Q[500], Q[501], \ldots, Q[511]$ is approximately $8 \cdot (1 - \frac{43}{512})^M + 4 \cdot (1 - \frac{42}{512})^M$, where M is the size of the largest component of the index graph G.

Substituting M by its theoretical mean estimate 443 as well as by its empirical mean estimate 408 yields $E(Y) \approx 0$.
In fact, for any $M > 200$, the expression $(1 - \frac{43}{512})^M + 4 \cdot (1 - \frac{42}{512})^M$ for $E(Y)$ becomes vanishingly small.
Our experimental data also supports that in every instance, none of the words $Q[500], Q[501], \ldots, Q[511]$ remains unknown.

Roadmap

Keystream Generation in Block B_3

$s_{3,i} = h_2(Q[500+i]) \oplus Q_N[i]$, $0 \le i \le 11$.

Expanding $h_2(.)$, we get, for $0 \le i \le 11$,

$$Q_N[i] = s_{3,i} \oplus \left(P_N\left[(Q[500+i])^{(0)} \right] + P_N\left[256 + (Q[500+i])^{(2)} \right] \right).$$

Now apply Propagation Theorem. Thus, we have a new result.

Theorem

Suppose the complete array P_N and the 12 words $Q[500], Q[501], \ldots, Q[511]$ from the array Q are known. Then the entire Q_N array can be reconstructed in a time complexity linear in the size of Q.

Keystream Generation in Block B_3

$s_{3,i} = h_2(Q[500+i]) \oplus Q_N[i]$, $0 \le i \le 11$.
Expanding $h_2(.)$, we get, for $0 \le i \le 11$,

$$Q_N[i] = s_{3,i} \oplus \left(P_N\left[(Q[500+i])^{(0)}\right] + P_N\left[256 + (Q[500+i])^{(2)}\right] \right).$$

Now apply Propagation Theorem. Thus, we have a new result.

Theorem

Suppose the complete array P_N and the 12 words $Q[500], Q[501], \ldots, Q[511]$ from the array Q are known. Then the entire Q_N array can be reconstructed in a time complexity linear in the size of Q.

Keystream Generation in Block B_3

$s_{3,i} = h_2(Q[500+i]) \oplus Q_N[i]$, $0 \le i \le 11$.
Expanding $h_2(.)$, we get, for $0 \le i \le 11$,

$$Q_N[i] = s_{3,i} \oplus \left(P_N \left[(Q[500+i])^{(0)} \right] + P_N \left[256 + (Q[500+i])^{(2)} \right] \right).$$

Now apply Propagation Theorem. Thus, we have a new result.

Theorem

Suppose the complete array P_N and the 12 words $Q[500], Q[501], \ldots, Q[511]$ from the array Q are known. Then the entire Q_N array can be reconstructed in a time complexity linear in the size of Q.

Roadmap

Keystream Generation in Block B_4

- We update P_N as it would be updated in block B_4 and generate 512 keystream words with this P_N and the derived Q_N.
- If the generated keystream words entirely match with the observed keystream words $\{s_{4,0}, s_{4,1}, \ldots, s_{4,511}\}$ of block B_4, then our guess is correct.
- If we find a mismatch, then we repeat the procedure with the next guess, i.e., with another possible value in $[0, 2^{32} - 1]$ of the word of Q in the largest component of the index graph.

Keystream Generation in Block B_4

- We update P_N as it would be updated in block B_4 and generate 512 keystream words with this P_N and the derived Q_N.
- If the generated keystream words entirely match with the observed keystream words $\{s_{4,0}, s_{4,1}, \ldots, s_{4,511}\}$ of block B_4, then our guess is correct.
- If we find a mismatch, then we repeat the procedure with the next guess, i.e., with another possible value in $[0, 2^{32} - 1]$ of the word of Q in the largest component of the index graph.

Keystream Generation in Block B_4

- We update P_N as it would be updated in block B_4 and generate 512 keystream words with this P_N and the derived Q_N.
- If the generated keystream words entirely match with the observed keystream words $\{s_{4,0}, s_{4,1}, \ldots, s_{4,511}\}$ of block B_4, then our guess is correct.
- If we find a mismatch, then we repeat the procedure with the next guess, i.e., with another possible value in $[0, 2^{32} - 1]$ of the word of Q in the largest component of the index graph.

Data Complexity

- For the First Phase, we do not need any keystream word.
- For each of the Second, Third, Fourth and Fifth Phases, we need a separate block of 512 keystream words.
- Thus, the required amount of data is $4 \cdot 512 = 2^{11}$ no. of 32 $(= 2^5)$-bit keystream words, giving a data complexity 2^{16}.

Data Complexity

- For the First Phase, we do not need any keystream word.
- For each of the Second, Third, Fourth and Fifth Phases, we need a separate block of 512 keystream words.
- Thus, the required amount of data is $4 \cdot 512 = 2^{11}$ no. of 32 $(= 2^5)$-bit keystream words, giving a data complexity 2^{16}.

Data Complexity

- For the First Phase, we do not need any keystream word.
- For each of the Second, Third, Fourth and Fifth Phases, we need a separate block of 512 keystream words.
- Thus, the required amount of data is $4 \cdot 512 = 2^{11}$ no. of 32 $(= 2^5)$-bit keystream words, giving a data complexity 2^{16}.

Time Complexity

We prove the time complexity to be 2^{42}. This includes

- Time to find the largest component.
- Time for computing Phases 3, 4 and 5 for each of 2^{32} guesses of the selected node in the largest component.

Time Complexity

We prove the time complexity to be 2^{42}. This includes

- Time to find the largest component.
- Time for computing Phases 3, 4 and 5 for each of 2^{32} guesses of the selected node in the largest component.

Roadmap

1. Introduction
   - Background
   - Description of HC-128
   - Contribution
2. Reconstructing One Array from Another
   - First Phase: Complete P_N from P
   - Second Phase: Part of Q from P_N
   - Third Phase: Tail of Q from its Parts
   - Fourth Phase: Complete Q_N from Tail of Q
   - Fifth Phase (Verification) and Total Complexity
3. Proposal for Design Modification

Target

To guard against all known weaknesses:

- Weakness discovered in this work.
- Previously known weaknesses.

All known weaknesses exploit the fact that $h_1(.)$ as well as $h_2(.)$ makes use of only 16 bits from the 32-bit input.

Target

To guard against all known weaknesses:

- Weakness discovered in this work.
- Previously known weaknesses.

All known weaknesses exploit the fact that $h_1(.)$ as well as $h_2(.)$ makes use of only 16 bits from the 32-bit input.

Fix 1: Use All Input Bits in h_1, h_2

We replace h_1 and h_2 as follows.

$$\begin{aligned} h_{N1}(x) &= (P[x^{(0)}] + P[256 + x^{(2)}]) \oplus x. \\ h_{N2}(x) &= (Q[x^{(0)}] + Q[256 + x^{(2)}]) \oplus x. \end{aligned}$$

Fix 2: Make Updates of P and Q Interdependent

We include a randomly chosen word from the Q array in the update of P array elements and a randomly chosen word from the P array while updating the Q array elements.

$$
\begin{aligned}
g_{N1}(x, y, z) &= ((x \ggg 10) \oplus (z \ggg 23)) + Q[(y \gg 7) \wedge 1FF].\\
g_{N2}(x, y, z) &= ((x \lll 10) \oplus (z \lll 23)) + P[(y \gg 7) \wedge 1FF].
\end{aligned}
$$

The internal state would be preserved even if half the internal state elements are revealed and known distinguishers cannot be mounted.

Fix 2: Make Updates of P and Q Interdependent

We include a randomly chosen word from the Q array in the update of P array elements and a randomly chosen word from the P array while updating the Q array elements.

$$\begin{aligned} g_{N1}(x, y, z) &= \left((x \ggg 10) \oplus (z \ggg 23)\right) + Q[(y \gg 7) \wedge 1FF]. \\ g_{N2}(x, y, z) &= \left((x \lll 10) \oplus (z \lll 23)\right) + P[(y \gg 7) \wedge 1FF]. \end{aligned}$$

The internal state would be preserved even if half the internal state elements are revealed and known distinguishers cannot be mounted.

Fix 2: Make Updates of P and Q Interdependent

We include a randomly chosen word from the Q array in the update of P array elements and a randomly chosen word from the P array while updating the Q array elements.

$$\begin{aligned} g_{N1}(x, y, z) &= \big((x \ggg 10) \oplus (z \ggg 23)\big) + Q[(y \gg 7) \wedge 1FF]. \\ g_{N2}(x, y, z) &= \big((x \lll 10) \oplus (z \lll 23)\big) + P[(y \gg 7) \wedge 1FF]. \end{aligned}$$

The internal state would be preserved even if half the internal state elements are revealed and known distinguishers cannot be mounted.

Performance of the New Design

We evaluated the performance of our new design using the eSTREAM testing framework.

	HC-128	Our Proposal	HC-256
Stream Encryption (cycles/byte)	4.13	4.29	4.88

Results obtained in a machine with Intel(R) Pentium(R) D CPU, 2.8 GHz Processor Clock, 2048 KB Cache Size, 1 GB DDR RAM on Ubuntu 7.04 (Linux 2.6.20-17-generic) OS using the gcc-3.4_prescott_O3-ofp compiler.

Thank You

Questions?