STUDI MENGENAI ALGORITMA WHIRLPOOL DAN IMPLEMENTASINYA

oleh
Ade Gunawan 13504049

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Januari 2007

Abstrak

Makalah ini membahas tentang salah satu fungsi *hash* yang disebut WHIRLPOOL.Perlu diketahui bahwa fungsi *hash* adalah fungsi yang menerima masukan pesan berupa *string* yang panjangnya sembarang dan mengkonversinya menjadi *string* keluaran dengan panjang tetap (fixed) yang umumnya jauh lebih kecil daripada ukuran pesan. *String* keluaran itulah yang biasa disebut dengan pesan-ringkas (*message digest*), yang idealnya unik untuk setiap pesan.

WHIRLPOOL adalah fungsi *hash* kriptografi yang dirancang dan didesain oleh Vincent Rijmen dan Paulo S. L. M. Barreto. WHIRLPOOL menerima masukan suatu pesan berupa string dengan panjang maksimum 2^{256} bit, dan menghasilkan keluaran sebuah *message digest* dengan panjang tetap, yaitu 512 bit atau 128 angka dalam heksadesimal. Nama WHIRLPOOL diambil dari sebuah galaksi Whirlpool (pusaran) di Canes Venatici yang merupakan galaksi pertama yang ditemukan memiliki struktur spiral. Selanjutnya, dalam makalah akan terlihat bahwa fungsi yang digunakan dalam WHIRLPOOL mirip dengan sebuah pusaran.

WHIRLPOOL mempunyai tiga versi. Versi pertamanya disebut WHIRLPOOL-0 dibuat pada tahun 2000. Satu tahun kemudian, dibuatlah versi baru yang disebut dengan WHIRLPOOL-1 yang merupakan penyempurnaan dari versi pertamanya. Selanjutnya, pada tahun 2003 ditemukan kelemahan pada *diffusion matrix*-nya sehingga diluncurkanlah versi terbaru dan juga sekaligus versi terakhirnya sampai sekarang. Versi yang paling sempurna tersebut disebut WHIRLPOOL saja untuk memudahkan penyebutan.

Makalah ini juga akan membahas tentang pengimplementasian fungsi *hash* WHIRLPOOL dalam sebuah bahasa pemrograman tingkat tinggi. Hasil dari implementasi tersebut akan digunakan sebagai pengujian fungsi *hash* WHIRLPOOL pada berbagai macam kasus.

Berdasarkan hal-hal di atas, diharapkan makalah ini dapat membantu masalah otentikasi dan integritas pesan dengan mengunakan fungsi *hash* WHIRLPOOL sebagai “alat bantu”-nya.

1. Pendahuluan

1. 1. Latar Belakang

Dalam pengiriman dan penerimaan pesan, masalah otentikasi dan integritas pesan tetntunya menjadi masalah yang sangat penting. Salah satu contoh masalah yang sering terjadi adalah bagaimana meyakinkan penerima pesan bahwa pesan yang diterimanya benar-benar dikirim oleh nama orang yang tercantum sebagai pengirim di dalam pesan tersebut. Masalah yang juga sering terjadi lainnya adalah penerima tidak tahu bahwa pesan yang diterima itu sudah diubah oleh orang lain atau tidak. Untuk mengatasi masalah kritikal tersebut, dirancangkan suatu fungsi *hash* kriptografi (biasa disebut fungsi *hash* saja). Karena idealnya, hasil dari fungsi *hash* untuk setiap pesan adalah unik, akan dapat ditentukan apakah pesan tersebut benar-benar asli atau sudah diubah. Fungsi *hash* kemudian digunakan dalam digital *signature* untuk otentikasi pesan.

Berdasarkan hal ini, penulis tertarik untuk meneliti tentang salah satu fungsi *hash* kriptografi. Dari hasil observasi yang penulis lakukan, penulis tertarik pasa suatu fungsi *hash* yang disebut WHIRLPOOL. Penulis akan meneliti tentang algoritmanya dan fungsi yang

digunakannya serta hubungannya terhadap keamanan dari fungsi *hash* ini. Tidak hanya itu, penulis juga akan berusaha mengimplementasikan fungsi *hash* ini sehingga dapat dibuktikan keamanan dari fungsi *hash* WHIRLPOOL ini secara baik secara teori maupun praktik.

Tujuan dari penelitian ini bukan hanya untuk mengetahui algoritma dan kekuatan dari fungsi *hash* WHIRLPOOL, melainkan juga bagaimana implentasinya dalam suatu bahasa pemrograman tingkat tinggi. Diharapkan dari studi ini dapat diketahui bahwa WHIRLPOOL adalah fungsi *hash* dengan tingkat keamanan yang tinggi. Selain itu, implementasi fungsi *hash* ini diharapkan dapat berguna baik secara langsung maupun tidak langsung untuk pemakaian nyata dalam pengiriman pesan. Dari hal-hal tersebut, dapat dikatakan bahwa tujuan akhir dari penulis adalah untuk membantu mengatasi masalah otentikasi dan integritas pesan.

1. 2. Rumusan Masalah

Berdasarkan latar belakang di atas, rumusan masalah yang penulis kemukakan adalah:

1. Penjelasan rinci mengenai algoritma fungsi *hash* kriptografi WHIRLPOOL.
2. Implementasi dari fungsi *hash* kriptografi WHIRLPOOL.
3. Pengujian dan keamanan dari fungsi *hash* kriptografi WHIRLPOOL.

1. 3. Batasan Masalah

Untuk menjawab rumusan masalah yang dikemukakan, akan penulis kaji hal-hal berikut ini:

1. Semua hal secara rinci tentang fungsi *hash* kriptografi WHIRLPOOL (dalam hal ini WHIRLPOOL versi terbaru).
2. Implementasi dari fungsi *hash* kriptografi WHIRLPOOL ke dalam sebuah software dengan menggunakan sebuah bahasa pemrograman (C++).
3. Pengujian pada hasil implementasi seperti: pengujian untuk masukan pesan yang berbeda, pengujian untuk pesan yang telah diubah, dan lain-lain.

1. 4. Metode dan Teknik Pengumpulan Data

Metode yang digunakan dalam penelitian ini adalah deskriptif analitis. Tahap pertama dari deskriptif analitis adalah mendeskripsikan data dan informasi yang diperoleh. Kemudian dari data dan informasi yang diperoleh tersebut, akan penulis analisis sehingga menghasilkan suatu simpulan.
Untuk mendukung metode yang telah penulis kemukakan di atas, teknik pengumpulan data yang penulis gunakan adalah:

1. Studi literatur.
2. Studi hasil implementasi.
3. Pengujian implemementasi pada berbagai kasus.

2. Landasan Teori

2. 1. Kriptografi

2. 1. 1. Definisi

- Definisi lama: Kriptografi adalah ilmu dan seni untuk menjaga kerahasiaan pesan dengan cara menyandikannya ke dalam bentuk yang tidak dapat dimengerti lagi maksudnya.
- Definisi batu: Kriptografi adalah ilmu sekaligus seni untuk menjaga keamanan pesan (message).

2. 1. 2. Algoritma Kriptografi

Algoritma kriptografi adalah:

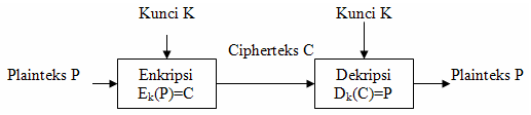
- Aturan untuk mengenkripsi dan mendekripsi.
- Fungsi matematika yang digunakan untuk enkripsi dan dekripsi.

Keterangan:

- Enkripsi: Proses transformasi plainteks menjadi cipherteks.
- Dekripsi: Proses transformasi cipherteks menjadi plainteks.
- Plainteks: Pesan yang dapat dibaca secara langsung.
- Cipherteks: Pesan yang tidak dapat dibaca secara langsung.

Baik proses enkripsi maupun dekripsi membutuhkan sebuah parameter yang disebut kunci. Algoritma kriptografi tidak perlu dirahasiakan, tapi kunci harus dirahasiakan sehingga hanya orang yang berhak saja yang mampu membaca pesan. Skema enkripsi dan

dekripsi dapat dilihat pada gambar 1 di bawah ini:



Gambar 1 Skema Enkripsi dan Dekripsi

2. 1. 3. Kriptografi Kunci Simetri / Nirsimetri

Berdasarkan kunci yang digunakan untuk enkripsi dan dekripsi, kriptografi dapat dibedakan menjadi dua, yaitu:

1. Kriptografi kunci simetri.
   Kriptografi kunci simetri adalah kriptografi yang kunci enkripsinya sama dengan kunci dekripsinya, sedangkan algoritma kriptografinya disebut algoritma simetri atau algoritma konvensional.
2. Kriptografi kunci nirsimetri.
   Kriptografi kunci nirsimetri adalah kriptografi yang kunci enkripsinya tidak sama / berbeda dengan kunci dekripsinya, sedangkan algoritmanya kriptografinya disebut algoritma nirsimetri atau algoritma kunci-publik.

2. 1. 4. Algoritma Kriptografi Modern

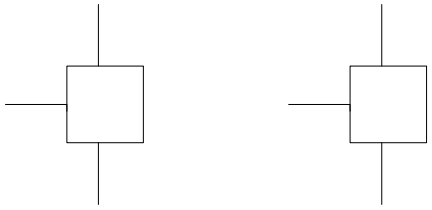
Penggunaan komputer digital yang merepresentasikan data dalam bentuk biner mendorong perkembangan algoritma kriptografi modern berbasis bit. Oleh karena itu, algoritma kriptografi modern umumnya beroperasi dalam mode bit daripada mode karakter. Operasi dalam mode bit memungkinkan enkripsi dan dekripsi dalam bentuk rangkaian (*string*) bit biner (0 dan 1).

2. 2. Tipe Algoritma Simetri

Algoritma simetri berbasis bit dapat dibagi menjadi dua macam, yaitu:

1. *Cipher* aliran (*stream cipher*).
   Stream cipher adalah algoritma yang beroperasi pada plainteks/cipherteks dalam bentuk bit tunggal, sehingga rangkaian bit dienkripsikan / didedripsikan bit per bit.

2. *Cipher* blok (*block cipher*).
   Block cipher adalah algoritma beroperasi pada plainteks/cipherteks dalam bentuk blok bit, sehingga rangkaian bit dibagi menjadi blok-blok sama panjang dan dienkripsikan / didedripsikan blok per blok. Skema *cipher* blok dapat dilihat pada gambar 2.



Gambar 2 Skema *cipher block*

Dari gambar di atas, dapat dilihat bahwa pada *cipher* blok, rangkaian bit-bit plainteks dibagi menjadi blok-blok bit dengan panjang sama (m). Enkripsi dilakukan terhadap blok bit plainteks menggunakan bit-bit kunci (yang ukurannya sama dengan blok plainteks). Algoritma enkripsi menghasilkan blok cipherteks yang berukuran sama dengan blok plainteks. Dekripsi dilakukan dengan cara yang serupa seperti enkripsi.

Misalkan blok plainteks (P) yang berukuran m bit dinyatakan sebagai vektor

$$P = (p_1, p_2, ..., p_m)$$

yang dalam hal ini p_i adalah bit 0 atau bit 1 untuk $i = 1, 2, \ldots, m$, dan blok cipherteks (C) adalah

$$C = (c_1, c_2, ..., c_m)$$

yang dalam hal ini c_i adalah bit 0 atau bit 1 untuk $i = 1, 2, \ldots, m$.

Bila plainteks dibagi menjadi n buah blok, barisan blok-blok plainteks dinyatakan sebagai

$$(P_1, P_2, \ldots, Pn)$$

Untuk setiap blok plainteks P_i, bit-bit penyusunnya dapat dinyatakan sebagai vektor

$$P_i = (p_{i1}, p_{i2}, ..., p_{im})$$

Enkripsi dengan kunci K dinyatakan dengan persamaan

$$E_k(P) = C,$$

sedangkan dekripsi dengan kunci K dinyatakan dengan persamaan

$$D_k(C) = P$$

2. 3 Fungsi Hash

2. 3. 1. Definisi dan Penjelasan Fungsi Hash

Di dalam kriptografi terdapat sebuah fungsi untuk aplikasi keamanan seperti otentikasi dan integritas pesan. Fungsi tersebut adalah fungsi *hash*. Fungsi *hash* adalah fungsi yang menerima masukan *string* yang panjangnya sembarang dan mengkonversinya menjadi *string* keluaran yang panjangnya tetap (*fixed*) yang umumnya berukuran jauh lebih kecil daripada ukurang *string* masukan semula. Fungsi *hash* dapat menerima masukan *string* apa saja.

Secara matematis fungsi *hash* dapat dirumuskan melalui persamaan:

$$h = H(M)$$

Dimana M adalah pesan (*message*) sembarang dengan ukuran bebas, H adalah fungsi *hash*, dan h adalah keluaran dari fungsi *hash* yang disebut juga nilai *hash* (*hash-value*) atau pesan-ringkas (*message digest*).

Dari penjelasan di atas, dapat disimpulkan bahwa fungsi *hash* mengkompresi sembarang pesan yang berukuran apa saja menjadi *message digest* yang ukurannya selalu tetap dan lebih pendek dari panjang pesan semula.

2. 3. 1. Fungsi Hash Satu-Arah

Fungsi *hash* satu-arah (*One-way Hash*) adalah fungsi *hash* yang bekerja dalam satu arah: pesan yang sudah diubah menjadi *message digest* tidak dapat lagi dikembalikan lagi menjadi pesan semula. Dua pesan yang berbeda akan selalu menghasilkan nilai *hash* yang berbeda pula. Sifat-sifat fungsi *hash* satu-arah adalah sebagai berikut:

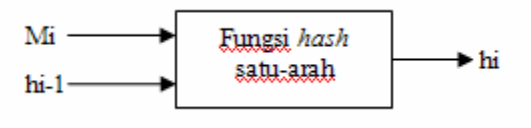
1. Fungsi H dapat diterapkan pada blok data berukuran berapa saja.
2. H menghasilkan nlai (h) dengan panjang tetap (*fixed-length output*).
3. H(x) mudah dihitung untuk setiap nilai x yang diberikan.
4. Untuk setiap h yang diberikan, tidak mungkin menemukan x sedemikian sehingga H(x)=h itulah sebabnya fungsi H dikatakan fungsi *hash* satu-arah (*one-way hash function*).
5. Untuk setiap x yang diberikan, tidak mungkin mencari y!=x sedemikian sehingga H(x)=H(y).
6. Tidak mungkin (secara komputasi) mencari pasangan x dan y sedemikian sehingga H(x)=H(y).

Keenam sifat di atas penting sebab sebuah fungsi *hash* seharusnya berlaku seperti fungsi acak. Sebuah fungsi *hash* dianggap tidak aman jika (i) secara komputasi dimungkinkan menemukan pesan yang bersesuaian dengan pesan ringkasnya, dan (ii) terjadi kolisi (*collision*), yaitu terdapat beberapa pesan berbeda yang mempunyai pesan ringkas yang sama.

Fungsi *hash* bekerja secara iteratif. Masukan fungsi *hash* adalah blok pesan (M) dan keluaran dari *hashing* blok pesan sebelumnya.

$$h_i = H(M_i, h_{i-1})$$

Skema fungsi *hash* ditunjukkan pada gambar berikut:



Gambar 3 Fungsi *hash* satu-arah

Fungsi *hash* adalah publik (tidak dirahasiakan), dan keamanannya terletak pada sifat satu arahnya itu.

2. 4 Kotak-S (S-*box*)

Kotak-S adalah matriks yang berisi subtitusi sederhana yang memetakan satu atau lebih bit dengan satu atau lebih bit lainya. Pada kebanyakan algoritma cipher blok, kotak-S memetakan m bit masukan menjadi n bit keluaran, sehingga kotak-S tersebut dinamakan kotak m x n S-*box*. Kotak-S merupakan satu-satunya langkah nirlanjar di dalam algoritma, karena operasinya adalah *look-up table*. Masukan dari operasi *look-up table* dijadikan sebagai indeks kotak-S dan keluarannya adalah *entry* di dalam kotak-S. Perancangan kotak-S

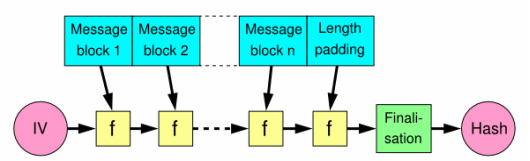
menjadi isu penting karena kotak-S harus dirancang sedemikian sehingga kekuatan kriptografinya bagus dan mudah diimplementasikan.

3. Fungsi Hash Whirlpool

3. 1 Gambaran Umum

Whirlpool adalah fungsi *hash* satu-arah yang dirancang oleh Vincent Rijmen dan Paulo S. L. M. Barreto. Whirlpool beroperasi pada *message* sembarang dengan panjang kurang dari 2^{256} bit, dan menghasilkan keluaran *message digest* dengan panjang tetap, yaitu 512 bit. *Message digest* yang dihasilkan adalah angka-angka dalam format heksadesimal. Karena satu angka heksadesimal mempunyai panjang 4 bit, panjang *message digest* yang dihasilkan 512 / 4 = 128 angka.

Whirlpool adalah fungsi *hash* yang berbasis pada *block cipher* (penjelasan tentang *block cipher* terdapat pada subbab 2. 2). Whirlpool menggunakan konstruksi Merkle-Damgard sehingga fungsi ini dapat mengubah pesan masukan dengan panjang sembarang dan menghasilkan *message digest* dengan panjang tetap.



Gambar 4 Konstruksi Merkle-Damgard

Dapat dilihat dari gambar di atas bahwa *message* masukan dengan panjang sembarang dipecah-pecah menjadi blok-blok dengan panjang sama. Panjang blok untuk Whirlpool adalah 512. Blok-blok *message* tersebut selanjutnya akan diproses secara sekuensial dengan fungsi kompresi f.

Dapat kita katakan bahwa secara garis besar, langkah-langkah pembuatan message digest adalah:

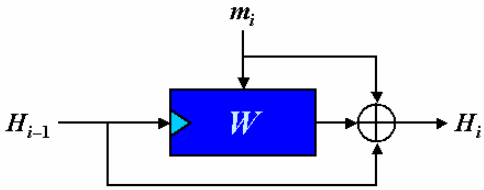
1. Inisialisasi IV (*initialization vector*). Dalam Whirlpool IV berukuran 512 bit dan semua bit-nya berisi bit "0".
2. Prosedur *finalization* dengan cara menambahkan bit-bit pengganjal (*padding bits*) dan penambahan nilai panjang *message*.
3. Pengolahan pesan dengan fungsi f

3. 2 Penambahan Bit-bit Pengganjal

Agar blok *message* yang menjadi masukan fungsi f selalu mempunyai panjang tetap (512 bit), dilakukan prosedur *padding*. *Message* akan ditambah dengan sejumlah bit-bit pengganjal sedemikian sehingga panjang pesan (dalam satuan bit) kongruen dengan 256 modulo 512. Bit-bit pengganjal yang ditambahkan tersebut terdiri dari sebuah bit "1" yang diikuti dengan bit-bit "0". 256 bit sisa dari blok *message* akan diisi dengan panjang *message* masukan. Dengan demikian, setelah dilakukan prosedur *finalization* ini, akan didapat *message* dengan panjang kelipatan 512 bit.

3. 3 Fungsi Kompresi

Fungsi f yang digunakan dalam Whirlpool adalah fungsi kompresi Miyaguchi-Preneel. Skema fungsi Miyaguchi-Preneel dapat dilihat di bawah ini:



Gambar 5 Fungsi Miyaguchi-Preneel

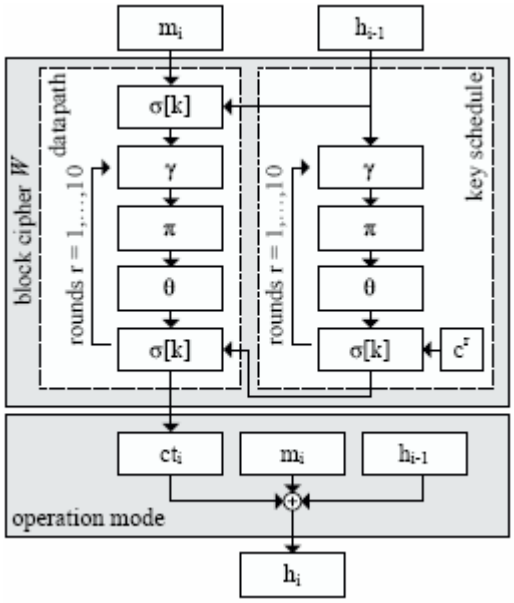
Fungsi ini menerima dua masukan: blok *message* yang *current* (m_i) dan hasil fungsi sebelumnya (H_{i-1}). Dua masukan in kemudian diproses berbasis *block cipher* 512 bit W. *Block cipher* W dapat dikatakan mengenkripsi m_i dengan kunci H_{i-1}. Hasil dari *block cipher* W kemudian di-XOR-kan kembali dengan dua masukan tadi. Hasil peng-XOR-an tersebutlah yang akan menjadi masukan fungsi selanjutnya.

3. 4 Block Cipher Internal W

Block cipher yang digunakan dalam Whirlpool sangat mirip dengan algoritma AES. W adalah *block cipher* 512 bit dan menggunakan kunci dengan panjang 512 bit pula. Yang berperan sebagai plainteks dalam hal ini adalah blok *message* yang ke-i (m_i), sedangkan yang

berperan sebagai kunci (*cipher key*) adalah hasil dari fungsi *hash* sebelumnya (H_{i-1}).

Block cipher W secara umum dapat dibagi menjadi dua bagian: *datapath* dan *key schedule*. Gambar di bawah ini akan memperjelas aliran data dalam *block cipher* W



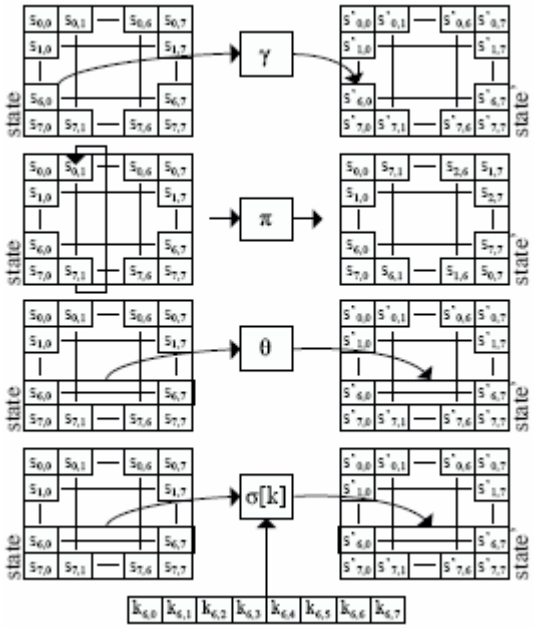
Gambar 6 Aliran Data Whirlpool

Datapath menerima masukan blok *message* m_i dengan panjang 512 bit dan memprosesnya secara iteratif dengan melakukan *round transformations* sebanyak 10 kali. Dapat dilihat pada gambar 6 bahwa setiap ronde pada *datapath* memerlukan *round key* yang dapat diturunkan dari proses *key schedule*. Setelah 10 ronde telah dilakukan, akan dihasilkan keluaran cipherteks ct_i.

Proses *key schedule* pada dasarnya menggunakan 10 ronde *round transformations* yang sama dengan *datapath*. Perbedaannya adalah *key schedule* menggunakan *round constants* sebagai masukannya.

3. 5 Round Transformation

Gambaran mengenai *round transformations* dapat dilihat pada gambar di bawah ini:



Gambar 7 Keempat Round Transformation

Penjelasan dari keempat *round transformations* tersbut sebagai berikut:

1. γ adalah langkah non linear dimana setiap *byte* dari masukan akan disubstitusikan ke dalam S-Box.
2. π adalah permutasi putaran (*cyclical permutation*) dimana *byte-byte* dari kolom j dirotasikan ke bawah sebanyak j kali.
3. θ adalah langkah difusi linear dimana masukan dilipatgandakan dengan matriks konstatnta.
4. σ[k] penambahan kunci (*key addition*) dimana pada *datapath*, kunci didapatkan dari hasil *key schedule*, sedangkan pada *key schedule* itu sendiri, kunci didapatkan dari *round constants*.

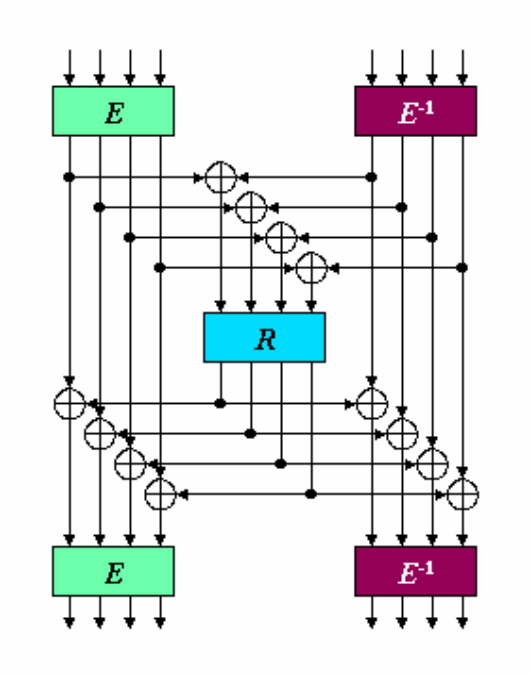
Dapat dilihat pula dari gambar 7 bahwa satu ronde transformasi ρ[k] dari *W* dapat dinyatakan secara matematis sebagai:

$$\rho[k] \equiv \sigma[k] \circ \theta \circ \pi \circ \gamma$$

3. 6 S-Box

Proses γ pada *round transformations* menggunakan S-Box non linear dengan ukuran 8x8. Pada versi pertama Whirlpool (disebut juga Whirlpool-0), S-Box dibangkitkan seluruhnya secara acak (*random*). S-Box tersebut

diperbaharui pada tahun 2001 menjadi S-Box berstruktur rekursif. Alasan pengubahan S-Box ini adalah karena ingin mempertinggi tingkat keamanan kriptografinya dan agar lebih mudah mengimplementasikannya dalam hardware.



Gambar 8 Struktur Rekursif S-Box

Dapat kita lihat bahwa sebenarnya S-Box berukuran 8x8 ini dibangun dari tiga "mini-box" yang berukuran 4x4 (E, E^{-1}, dan R).

Ketiga mini-box tersebut adalah:

1. Mini-box E yang disebut kotak eksponensial.

u	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$E(u)$	1	B	9	C	D	6	F	3	E	8	7	4	A	2	5	0

Gambar 9 Mini-box E

2. Mini-box E^{-1} yang merupakan invers dari E.

u	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$E^{-1}(u)$	F	0	D	7	B	E	5	A	9	2	C	1	3	4	8	6

Gambar 10 Mini-Box E^{-1}

3. Mini-box R yang dibangkitkan secara acak semu (*pseudo-random*).

u	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$R(u)$	7	C	B	D	E	4	9	F	6	3	8	A	2	5	1	0

Gambar 11 Mini-box R

3. 7 Round Constants

Round constants yang dipakai pada proses *round transformation* σ[k] adalah:

1. `0x1823c6e887b8014f`
2. `0x36a6d2f5796f9152`
3. `0x60bc9b8ea30c7b35`
4. `0x1de0d7c22e4bfe57`
5. `0x157737e59ff04ada`
6. `0x58c9290ab1a06b85`
7. `0xbd5d10f4cb3e0567`
8. `0xe427418ba77d95d8`
9. `0xfbee7c66dd17479e`
10. `0xca2dbf07ad5a8333`

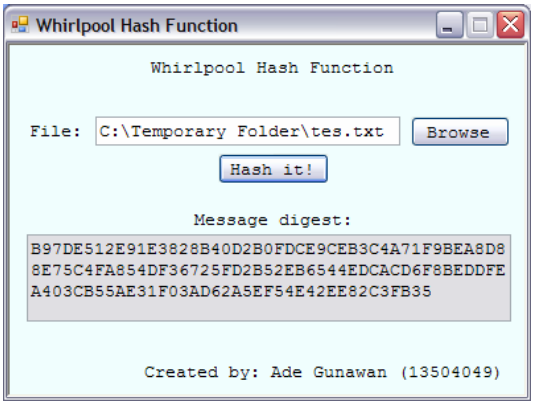
Dimana "0x" menyatakan angka dalam heksadesimal.

Konstanta-konstanta tersebut akan digunakan sesuai dengan urutan ronde pada *round transformation*. Jadi, konstanta pertama (`0x1823c6e887b8014f`) akan dipakai pada ronde pertama, dan seterusnya.

4. Implementasi Whirlpool

4. 1 Tampilan

Penulis melakukan implementasi *fungsi hash* satu-arah Whirlpool dengan kakas pemrograman Microsoft Visual Studio 2005 dengan bahasa C++. Tampilan GUI-nya dapat dilihat di bawah ini:



Gambar 12 Tampilan Implementasi

Software yang penulis buat ini menerima masukan file yang merupakan *message* yang akan di-*hash*, dan menghasilkan keluaran *message digest* berupa text seperti yang dapat dilihat pada gambar 12.

4. 2 Kelas Whirlpool

Dalam pengimplementasian ini, penulis merancang sebuah kelas yang diberi nama Whirlpool yang disimpan di dua file: file header Whirlpool.h dan file implementasi Whirlpool.cpp.

Kelas ini mempunyai beberapa atribut antara lain:

1. `array<System::UInt64>^ h;`
   h adalah *array* yang menyimpan *hashing value* ke-I, dimana h ke-0 adalah IV dan h terakhir adalah *message digest*.
2. `System::UInt64 LengthQ0, LengthQ1, LengthQ2, LengthQ3;`
   Variabel untuk menyimpan panjang *message* yang digunakan saat *finalization*. Panjang *message* disimpan dalam 4 x 64bit integer = 256 bit.
3. `array<System::Byte>^ m;`
   m adalah *array* untuk menyimpan blok message dengan panjang 64 *byte* = 512 bit.
4. `int mIndex;`
   mIndex digunakan sebagai iterator yang menandakan posisi bit m yang *current* pada saat pembagian input *message* menjadi blok *message*.
5. `static array<UInt64>^ rc`
   rc adalah *round constants* yang digunakan dalam *key schedule*.
6. `static array<UInt64>^ C0`
   `static array<UInt64>^ C1`
   `static array<UInt64>^ C2`
   `static array<UInt64>^ C3`
   `static array<UInt64>^ C4`
   `static array<UInt64>^ C5`
   `static array<UInt64>^ C6`
   `static array<UInt64>^ C7`
   Kedelapan tabel ini adalah tabel difusi yang digunakan pada saat *round transformations*.

Fungsi dan prosedur yang digunakan dalam kelas ini antara lain:

1. `Whirlpool();`
   Konstruktor yang berisi alokasi memori dan inisialisasi IV dan variabel lain.
2. `~Whirlpool();`
   Destruktor yang melakukan dealokasi memori.
3. `void Input(array<System::Byte>^ FileInput);`
   Prosedur yang menerima masukan *message* dan memproses blok-blok *message* setiap 512 bit / 64 *byte*.
4. `void ProcessMessageBlock();`
   Block cipher W (10 ronde *round transformations* dari *datapath* dan *key schedule*) dan fungsi kompresi Miyaguchi-Preneel.
5. `void Padding();`
   Finalisasi blok *message* dengan menambahkan bit-bit pengganjal dan panjang *message*.
6. `array<System::UInt64>^ GetDigest();`
   Fungsi yang menghasilkan *message digest* final yang telah di finalisasi diproses dengan algoritma fungsi *hash* Whirlpool.

Jadi secara umum, masukan berupa *array of byte* dari `FileInput` akan diproses dengan prosedur `Input(FileInput)`. Input akan melakukan konstruksi Merkle-Damgard dengan membagi masukan menjadi blok-blok dengan panjang 512 bit. Setiap blok *message* akan diproses dengan prosedur `ProcessMessageBlock()` yang melakukan 10 ronde *round* transformations dari *datapath* dan *key schedule*. Selanjutnya akan dipanggil fungsi `GetDigest()` yang akan mengembalikan *message digest* yang sebelumnya di finalisasi dengan prosedur `Padding()`.

5. Pengujian Implementasi

5. 1 Spesifikasi Hardware

Software hasil implementasi dijalankan pada komputer dengan spesifikasi:

1. Processor Intel Centrino Duo 1,66 GHz.
2. RAM 512 MB PC4200.
3. Virtual memory 768 MB dengan kecepatan haddisk 5400rpm.

5. 2 Performansi

Pengujian performa implementasi algoritma fungsi *hash* Whirlpool telah dilakukan pada berbagai macam file dengan berbagai macam ukuran.

No.	Ukuran File	Waktu Proses
1.	1, 04 MB	0, 21875 detik
2.	50, 6 MB	9, 078125 detik
3.	205 MB	53, 8125 detik

Tabel Pengujian Performa Implementasi

Perlu diingat bahwa waktu yang diperlukan pada tabel di atas bernilai sangat besar karena dipengaruhi oleh waktu I/O transfer yang sangat besar. Jika saja waktu I/O transfer pada pembacaan file masukan dapat dikurangi, waktu bersih yang diperlukan untuk menghasilkan *message digest* dapat berkurang sampai 70%.

5. 2 Modifikasi Message

Nilai *message digest* untuk masukan *message* "Jawa Timur akan tenggelam" adalah:

```
327FF4B64484C687968935BA04C72D845
9C8B87CFEC5F4F43A3BDE9902F3275818
6D2C6C5F10498849FB3D4C6657C92B6CC
D649ADECE4414C3A1F30CB627F7B3
```

Modifikasi pengubahan bit:
Nilai *message digest* untuk masukan *message* "Jawa Timur akan tenggelan" adalah:

```
B2428245A62E47A8386E2A1608DC49CEF
198DCE2A5EC4EE3AC9F6C62B955A9A9DF
93F6884A28B33D43040EB0875399D7161
823230CC50F5DFA8CAEA2E751F73E
```

Dapat dilihat bahwa pengubahan huruf "m" menjadi "n" pada kata "tenggelam" menghasilkan *message digest* yang jauh berbeda, padahal pengubahan huruf "m" menjadi "n" hanya mengubah satu bit saja dari *message* masukan

Nilai *message digest* untuk masukan *message* "Jawa Timur akan tenggelamo" adalah:

```
7259F14DACBB4E3F820C959757A8D7814
722B62A5D7CC61AE757FC411383EFA81F
5349EB8EAD3D823130B51B35891163D84
1F88FBED650AB79190E62FDFAC277
```

Modifikasi pengubahan satu *byte*:
Penambahan satu huruf "o" yang berarti menambah satu *byte message* juga menghasilkan *message digest* yang jauh berbeda

Nilai *message digest* untuk masukan *message* "Jawa Timur akan tenggela" adalah:

```
A74179E1B400F4D9A544CA0EFB97D8E5C
4DFC7769043BDA9351442C3A6EA18FA8B
1DE04EDFACC758EB75E1A7AFCFA868974
85FFAAA4DDAA7252EE33A6013F08C
```

Modifikasi pengurangan satu *byte*:
Pengurangan satu huruf "m" yang berarti mengurangi satu *byte message* menghasilkan *message digest* yang jauh berbeda pula.

6. Simpulan

1. Fungsi *hash* satu-arah Whirlpool menerima masukan *message* sembarang dengan panjang kurang dari 2^{256} bit dan menghasilkan keluaran *message digest* dengan panjang 512 bit.
2. Whirlpool menggunakan konstruksi Merkle-Damgard dengan fungsi kompresi Miyaguchi-Preneel.
3. Whirlpool berbasis *block cipher* 512 bit yang dapat dibagi menjadi *datapath* dan *key schedule*, dimana keduanya menggunakan 10 ronde *round transformations*.
4. Implementasi Whirlpool cukup mudah karena *round transformations* dapat diimplementasikan hanya dengan manipulasi dari delapan tabel difusi.
5. Dari pengujian implementasi dapat disimpulkan bahwa Whirlpool dapat menghasilkan *message digest* dengan waktu yang cukup cepat.
6. Modifikasi *message* sekecil apapun akan menghasilkan *message digest* yang jauh berbeda.
7. Sampai sekarang belum ditemukan adanya kolisi dan tidak ada satu pun serangan yang berhasil dilakukan pada fungsi *hash* Whirlpool.
8. Simpulan akhir: Whirlpool adalah fungsi *hash* yang memiliki tingkat keamanan sangat tinggi dan mudah diimplementasikan baik berupa software atau pun hardware.

Lampiran

Delapan Tabel Difusi Implementasi Whirlpool dalam bahasa C++:

```
static array<System::UInt64>^ C0={
        0x18186018c07830d8, 0x23238c2305af4626, 0xc6c63fc67ef991b8, 0xe8e887e8136fcdfb,
        0x878726874ca113cb, 0xb8b8dab8a9626d11, 0x0101040108050209, 0x4f4f214f426e9e0d,
        0x3636d836adee6c9b, 0xa6a6a2a6590451ff, 0xd2d26fd2debdb90c, 0xf5f5f3f5fb06f70e,
        0x7979f979ef80f296, 0x6f6fa16f5fcede30, 0x91917e91fcef3f6d, 0x52525552aa07a4f8,
        0x60609d6027fdc047, 0xbcbccabc89766535, 0x9b9b569baccd2b37, 0x8e8e028e048c018a,
        0xa3a3b6a371155bd2, 0x0c0c300c603c186c, 0x7b7bf17bff8af684, 0x3535d435b5e16a80,
        0x1d1d741de8693af5, 0xe0e0a7e05347ddb3, 0xd7d77bd7f6acb321, 0xc2c22fc25eed999c,
        0x2e2eb82e6d965c43, 0x4b4b314b627a9629, 0xfefedffea321e15d, 0x575741578216aed5,
        0x15155415a8412abd, 0x7777c1779fb6eee8, 0x3737dc37a5eb6e92, 0xe5e5b3e57b56d79e,
        0x9f9f469f8cd92313, 0xf0f0e7f0d317fd23, 0x4a4a354a6a7f9420, 0xdada4fda9e95a944,
        0x58587d58fa25b0a2, 0xc9c903c906ca8fcf, 0x2929a429558d527c, 0x0a0a280a5022145a,
        0xb1b1feb1e14f7f50, 0xa0a0baa0691a5dc9, 0x6b6bb16b7fdad614, 0x85852e855cab17d9,
        0xbdbdcebd8173673c, 0x5d5d695dd234ba8f, 0x1010401080502090, 0xf4f4f7f4f303f507,
        0xcbcb0bcb16c08bdd, 0x3e3ef83eedc67cd3, 0x0505140528110a2d, 0x676781671fe6ce78,
        0xe4e4b7e47353d597, 0x27279c2725bb4e02, 0x4141194132588273, 0x8b8b168b2c9d0ba7,
        0xa7a7a6a7510153f6, 0x7d7de97dcf94fab2, 0x95956e95dcfb3749, 0xd8d847d88e9fad56,
        0xfbfbcbfb8b30eb70, 0xeeee9fee2371c1cd, 0x7c7ced7cc791f8bb, 0x6666856617e3cc71,
        0xdddd53dda68ea77b, 0x17175c17b84b2eaf, 0x4747014702468e45, 0x9e9e429e84dc211a,
        0xcaca0fca1ec589d4, 0x2d2db42d75995a58, 0xbfbfc6bf9179632e, 0x07071c07381b0e3f,
        0xadad8ead012347ac, 0x5a5a755aea2fb4b0, 0x838336836cb51bef, 0x3333cc3385ff66b6,
        0x636391633ff2c65c, 0x02020802100a0412, 0xaaaa92aa39384993, 0x7171d971afa8e2de,
        0xc8c807c80ecf8dc6, 0x19196419c87d32d1, 0x494939497270923b, 0xd9d943d9869aaf5f,
        0xf2f2eff2c31df931, 0xe3e3abe34b48dba8, 0x5b5b715be22ab6b9, 0x88881a8834920dbc,
        0x9a9a529aa4c8293e, 0x262698262dbe4c0b, 0x3232c8328dfa64bf, 0xb0b0fab0e94a7d59,
        0xe9e983e91b6acff2, 0x0f0f3c0f78331e77, 0xd5d573d5e6a6b733, 0x80803a8074ba1df4,
        0xbebec2be997c6127, 0xcdcd13cd26de87eb, 0x3434d034bde46889, 0x48483d487a759032,
        0xffffdbffab24e354, 0x7a7af57af78ff48d, 0x90907a90f4ea3d64, 0x5f5f615fc23ebe9d,
        0x202080201da0403d, 0x6868bd6867d5d00f, 0x1a1a681ad07234ca, 0xaeae82ae192c41b7,
        0xb4b4eab4c95e757d, 0x54544d549a19a8ce, 0x93937693ece53b7f, 0x222288220daa442f,
        0x64648d6407e9c863, 0xf1f1e3f1db12ff2a, 0x7373d173bfa2e6cc, 0x12124812905a2482,
        0x40401d403a5d807a, 0x0808200840281048, 0xc3c32bc356e89b95, 0xecec97ec337bc5df,
        0xdbdb4bdb9690ab4d, 0xa1a1bea1611f5fc0, 0x8d8d0e8d1c830791, 0x3d3df43df5c97ac8,
        0x97976697ccf1335b, 0x0000000000000000, 0xcfcf1bcf36d483f9, 0x2b2bac2b4587566e,
        0x7676c57697b3ece1, 0x8282328264b019e6, 0xd6d67fd6fea9b128, 0x1b1b6c1bd87736c3,
        0xb5b5eeb5c15b7774, 0xafaf86af112943be, 0x6a6ab56a77dfd41d, 0x50505d50ba0da0ea,
        0x45450945124c8a57, 0xf3f3ebf3cb18fb38, 0x3030c0309df060ad, 0xefef9bef2b74c3c4,
        0x3f3ffc3fe5c37eda, 0x55554955921caac7, 0xa2a2b2a2791059db, 0xeaea8fea0365c9e9,
        0x656589650fecca6a, 0xbabad2bab9686903, 0x2f2fbc2f65935e4a, 0xc0c027c04ee79d8e,
        0xdede5fdebe81a160, 0x1c1c701ce06c38fc, 0xfdfdd3fdbb2ee746, 0x4d4d294d52649a1f,
        0x92927292e4e03976, 0x7575c9758fbceafa, 0x06061806301e0c36, 0x8a8a128a249809ae,
        0xb2b2f2b2f940794b, 0xe6e6bfe66359d185, 0x0e0e380e70361c7e, 0x1f1f7c1ff8633ee7,
        0x6262956237f7c455, 0xd4d477d4eea3b53a, 0xa8a89aa829324d81, 0x96966296c4f43152,
        0xf9f9c3f99b3aef62, 0xc5c533c566f697a3, 0x2525942535b14a10, 0x59597959f220b2ab,
        0x84842a8454ae15d0, 0x7272d572b7a7e4c5, 0x3939e439d5dd72ec, 0x4c4c2d4c5a619816,
        0x5e5e655eca3bbc94, 0x7878fd78e785f09f, 0x3838e038ddd870e5, 0x8c8c0a8c14860598,
        0xd1d163d1c6b2bf17, 0xa5a5aea5410b57e4, 0xe2e2afe2434dd9a1, 0x616199612ff8c24e,
        0xb3b3f6b3f1457b42, 0x2121842115a54234, 0x9c9c4a9c94d62508, 0x1e1e781ef0663cee,
        0x43431143225286 61, 0xc7c73bc776fc93b1, 0xfcfcd7fcb32be54f, 0x0404100420140824,
        0x51515951b208a2e3, 0x99995e99bcc72f25, 0x6d6da96d4fc4da22, 0x0d0d340d68391a65,
        0xfafacffa8335e979, 0xdfdf5bdfb684a369, 0x7e7ee57ed79bfca9, 0x242490243db44819,
        0x3b3bec3bc5d776fe, 0xabab96ab313d4b9a, 0xcece1fce3ed181f0, 0x1111441188552299,
        0x8f8f068f0c890383, 0x4e4e254e4a6b9c04, 0xb7b7e6b7d1517366, 0xebeb8beb0b60cbe0,
        0x3c3cf03cfdcc78c1, 0x81813e817cbf1ffd, 0x94946a94d4fe3540, 0xf7f7fbf7eb0cf31c,
        0xb9b9deb9a1676f18, 0x13134c13985f268b, 0x2c2cb02c7d9c5851, 0xd3d36bd3d6b8bb05,
        0xe7e7bbe76b5cd38c, 0x6e6ea56e57cbdc39, 0xc4c437c46ef395aa, 0x03030c03180f061b,
        0x565645568a13acdc, 0x44440d441a49885e, 0x7f7fe17fdf9efea0, 0xa9a99ea921374f88,
        0x2a2aa82a4d825467, 0xbbbbd6bbb16d6b0a, 0xc1c123c146e29f87, 0x53535153a202a6f1,
        0xdcdc57dcae8ba572, 0x0b0b2c0b58271653, 0x9d9d4e9d9cd32701, 0x6c6cad6c47c1d82b,
        0x3131c43195f562a4, 0x7474cd7487b9e8f3, 0xf6f6fff6e309f115, 0x464605460a438c4c,
        0xacac8aac092645a5, 0x89891e893c970fb5, 0x14145014a04428b4, 0xe1e1a3e15b42dfba,
        0x16165816b04e2ca6, 0x3a3ae83acdd274f7, 0x6969b9696fd0d206, 0x09092409482d1241,
        0x7070dd70a7ade0d7, 0xb6b6e2b6d954716f, 0xd0d067d0ceb7bd1e, 0xeded93ed3b7ec7d6,
        0xcccc17cc2edb85e2, 0x424215422a578468, 0x98985a98b4c22d2c, 0xa4a4aaa4490e55ed,
        0x2828a0285d885075, 0x5c5c6d5cda31b886, 0xf8f8c7f8933fed6b, 0x8686228644a411c2,
};
```

```
static array<System::UInt64>^ C1 = {
        0xd818186018c07830, 0x2623238c2305af46, 0xb8c6c63fc67ef991, 0xfbe8e887e8136fcd,
        0xcb878726874ca113, 0x11b8b8dab8a9626d, 0x0901010401080502, 0x0d4f4f214f426e9e,
        0x9b3636d836adee6c, 0xffa6a6a2a6590451, 0x0cd2d26fd2debdb9, 0x0ef5f5f3f5fb06f7,
        0x967979f979ef80f2, 0x306f6fa16f5fcede, 0x6d91917e91fcef3f, 0xf852525552aa07a4,
        0x4760609d6027fdc0, 0x35bcbccabc897665, 0x379b9b569baccd2b, 0x8a8e8e028e048c01,
        0xd2a3a3b6a371155b, 0x6c0c0c300c603c18, 0x847b7bf17bff8af6, 0x803535d435b5e16a,
        0xf51d1d741de8693a, 0xb3e0e0a7e05347dd, 0x21d7d77bd7f6acb3, 0x9cc2c22fc25eed99,
        0x432e2eb82e6d965c, 0x294b4b314b627a96, 0x5dfefedffea321e1, 0xd5575741578216ae,
        0xbd15155415a8412a, 0xe87777c1779fb6ee, 0x923737dc37a5eb6e, 0x9ee5e5b3e57b56d7,
        0x139f9f469f8cd923, 0x23f0f0e7f0d317fd, 0x204a4a354a6a7f94, 0x44dada4fda9e95a9,
        0xa258587d58fa25b0, 0xcfc9c903c906ca8f, 0x7c2929a429558d52, 0x5a0a0a280a502214,
        0x50b1b1feb1e14f7f, 0xc9a0a0baa0691a5d, 0x146b6bb16b7fdad6, 0xd985852e855cab17,
        0x3cbdbdcebd817367, 0x8f5d5d695dd234ba, 0x9010104010805020, 0x07f4f4f7f4f303f5,
        0xddcbcb0bcb16c08b, 0xd33e3ef83eedc67c, 0x2d0505140528110a, 0x78676781671fe6ce,
        0x97e4e4b7e47353d5, 0x0227279c2725bb4e, 0x7341411941325882, 0xa78b8b168b2c9d0b,
        0xf6a7a7a6a7510153, 0xb27d7de97dcf94fa, 0x4995956e95dcfb37, 0x56d8d847d88e9fad,
        0x70fbfbcbfb8b30eb, 0xcdeeee9fee2371c1, 0xbb7c7ced7cc791f8, 0x716666856617e3cc,
        0x7bdddd53dda68ea7, 0xaf17175c17b84b2e, 0x454747014702468e, 0x1a9e9e429e84dc21,
        0xd4caca0fca1ec589, 0x582d2db42d75995a, 0x2ebfbfc6bf917963, 0x3f07071c07381b0e,
        0xacadad8ead012347, 0xb05a5a755aea2fb4, 0xef838336836cb51b, 0xb63333cc3385ff66,
        0x5c636391633ff2c6, 0x1202020802100a04, 0x93aaaa92aa393849, 0xde7171d971afa8e2,
        0xc6c8c807c80ecf8d, 0xd119196419c87d32, 0x3b494939497270 92, 0x5fd9d943d9869aaf,
        0x31f2f2eff2c31df9, 0xa8e3e3abe34b48db, 0xb95b5b715be22ab6, 0xbc88881a8834920d,
        0x3e9a9a529aa4c829, 0x0b262698262dbe4c, 0xbf3232c8328dfa64, 0x59b0b0fab0e94a7d,
        0xf2e9e983e91b6acf, 0x770f0f3c0f78331e, 0x33d5d573d5e6a6b7, 0xf480803a8074ba1d,
        0x27bebec2be997c61, 0xebcdcd13cd26de87, 0x893434d034bde468, 0x3248483d487a7590,
        0x54ffffdbffab24e3, 0x8d7a7af57af78ff4, 0x6490907a90f4ea3d, 0x9d5f5f615fc23ebe,
        0x3d202080201da040, 0x0f6868bd6867d5d0, 0xca1a1a681ad07234, 0xb7aeae82ae192c41,
        0x7db4b4eab4c95e75, 0xce54544d549a19a8, 0x7f93937693ece53b, 0x2f222288220daa44,
        0x6364648d6407e9c8, 0x2af1f1e3f1db12ff, 0xcc7373d173bfa2e6, 0x8212124812905a24,
        0x7a40401d403a5d80, 0x4808082008402810, 0x95c3c32bc356e89b, 0xdfecec97ec337bc5,
        0x4ddbdb4bdb9690ab, 0xc0a1a1bea1611f5f, 0x918d8d0e8d1c8307, 0xc83d3df43df5c97a,
        0x5b97976697ccf133, 0x0000000000000000, 0xf9cfcf1bcf36d483, 0x6e2b2bac2b458756,
        0xe17676c57697b3ec, 0xe68282328264b019, 0x28d6d67fd6fea9b1, 0xc31b1b6c1bd87736,
        0x74b5b5eeb5c15b77, 0xbeafaf86af112943, 0x1d6a6ab56a77dfd4, 0xea50505d50ba0da0,
        0x5745450945124c8a, 0x38f3f3ebf3cb18fb, 0xad3030c0309df060, 0xc4efef9bef2b74c3,
        0xda3f3ffc3fe5c37e, 0xc755554955921caa, 0xdba2a2b2a2791059, 0xe9eaea8fea0365c9,
        0x6a656589650fecca, 0x03babad2bab96869, 0x4a2f2fbc2f65935e, 0x8ec0c027c04ee79d,
        0x60dede5fdebe81a1, 0xfc1c1c701ce06c38, 0x46fdfdd3fdbb2ee7, 0x1f4d4d294d52649a,
        0x7692927292e4e039, 0xfa7575c9758fbcea, 0x3606061806301e0c, 0xae8a8a128a249809,
        0x4bb2b2f2b2f94079, 0x85e6e6bfe66359d1, 0x7e0e0e380e70361c, 0xe71f1f7c1ff8633e,
        0x556262956237f7c4, 0x3ad4d477d4eea3b5, 0x81a8a89aa829324d, 0x5296966296c4f431,
        0x62f9f9c3f99b3aef, 0xa3c5c533c566f697, 0x102525942535b14a, 0xab59597959f220b2,
        0xd084842a8454ae15, 0xc57272d572b7a7e4, 0xec3939e439d5dd72, 0x164c4c2d4c5a6198,
        0x945e5e655eca3bbc, 0x9f7878fd78e785f0, 0xe53838e038ddd870, 0x988c8c0a8c148605,
        0x17d1d163d1c6b2bf, 0xe4a5a5aea5410b57, 0xa1e2e2afe2434dd9, 0x4e616199612ff8c2,
        0x42b3b3f6b3f1457b, 0x342121842115a542, 0x089c9c4a9c94d625, 0xee1e1e781ef0663c,
        0x6143431143225286, 0xb1c7c73bc776fc93, 0x4ffcfcd7fcb32be5, 0x2404041004201408,
        0xe351515951b208a2, 0x2599995e99bcc72f, 0x226d6da96d4fc4da, 0x650d0d340d68391a,
        0x79fafacffa8335e9, 0x69dfdf5bdfb684a3, 0xa97e7ee57ed79bfc, 0x19242490243db448,
        0xfe3b3bec3bc5d776, 0x9aabab96ab313d4b, 0xf0cece1fce3ed181, 0x9911114411885522,
        0x838f8f068f0c8903, 0x044e4e254e4a6b9c, 0x66b7b7e6b7d15173, 0xe0ebeb8beb0b60cb,
        0xc13c3cf03cfdcc78, 0xfd81813e817cbf1f, 0x4094946a94d4fe35, 0x1cf7f7fbf7eb0cf3,
        0x18b9b9deb9a1676f, 0x8b13134c13985f26, 0x512c2cb02c7d9c58, 0x05d3d36bd3d6b8bb,
        0x8ce7e7bbe76b5cd3, 0x396e6ea56e57cbdc, 0xaac4c437c46ef395, 0x1b03030c03180f06,
        0xdc565645568a13ac, 0x5e44440d441a4988, 0xa07f7fe17fdf9efe, 0x88a9a99ea921374f,
        0x672a2aa82a4d8254, 0x0abbbbd6bbb16d6b, 0x87c1c123c146e29f, 0xf153535153a202a6,
        0x72dcdc57dcae8ba5, 0x530b0b2c0b582716, 0x019d9d4e9d9cd327, 0x2b6c6cad6c47c1d8,
        0xa43131c43195f562, 0xf37474cd7487b9e8, 0x15f6f6fff6e309f1, 0x4c464605460a438c,
        0xa5acac8aac092645, 0xb589891e893c970f, 0xb414145014a04428, 0xbae1e1a3e15b42df,
        0xa616165816b04e2c, 0xf73a3ae83acdd274, 0x066969b9696fd0d2, 0x4109092409482d12,
        0xd77070dd70a7ade0, 0x6fb6b6e2b6d95471, 0x1ed0d067d0ceb7bd, 0xd6eded93ed3b7ec7,
        0xe2cccc17cc2edb85, 0x68424215422a5784, 0x2c98985a98b4c22d, 0xeda4a4aaa4490e55,
        0x752828a0285d8850, 0x865c5c6d5cda31b8, 0x6bf8f8c7f8933fed, 0xc28686228644a411,
};
```

```
static array<System::UInt64>^ C2 = {
        0x30d818186018c078, 0x462623238c2305af, 0x91b8c6c63fc67ef9, 0xcdfbe8e887e8136f,
        0x13cb878726874ca1, 0x6d11b8b8dab8a962, 0x0209010104010805, 0x9e0d4f4f214f426e,
        0x6c9b3636d836adee, 0x51ffa6a6a2a65904, 0xb90cd2d26fd2debd, 0xf70ef5f5f3f5fb06,
        0xf2967979f979ef80, 0xde306f6fa16f5fce, 0x3f6d91917e91fcef, 0xa4f852525552aa07,
        0xc04760609d6027fd, 0x6535bcbccabc8976, 0x2b379b9b569baccd, 0x018a8e8e028e048c,
        0x5bd2a3a3b6a37115, 0x186c0c0c300c603c, 0xf6847b7bf17bff8a, 0x6a803535d435b5e1,
        0x3af51d1d741de869, 0xddb3e0e0a7e05347, 0xb321d7d77bd7f6ac, 0x999cc2c22fc25eed,
        0x5c432e2eb82e6d96, 0x96294b4b314b627a, 0xe15dfefedffea321, 0xaed5575741578216,
        0x2abd15155415a841, 0xeee87777c1779fb6, 0x6e923737dc37a5eb, 0xd79ee5e5b3e57b56,
        0x23139f9f469f8cd9, 0xfd23f0f0e7f0d317, 0x94204a4a354a6a7f, 0xa944dada4fda9e95,
        0xb0a258587d58fa25, 0x8fcfc9c903c906ca, 0x527c2929a429558d, 0x145a0a0a280a5022,
        0x7f50b1b1feb1e14f, 0x5dc9a0a0baa0691a, 0xd6146b6bb16b7fda, 0x17d985852e855cab,
        0x673cbdbdcebd8173, 0xba8f5d5d695dd234, 0x2090101040108050, 0xf507f4f4f7f4f303,
        0x8bddcbcb0bcb16c0, 0x7cd33e3ef83eedc6, 0x0a2d050514052811, 0xce78676781671fe6,
        0xd597e4e4b7e47353, 0x4e0227279c2725bb, 0x8273414119413258, 0x0ba78b8b168b2c9d,
        0x53f6a7a7a6a75101, 0xfab27d7de97dcf94, 0x374995956e95dcfb, 0xad56d8d847d88e9f,
        0xeb70fbfbcbfb8b30, 0xc1cdeeee9fee2371, 0xf8bb7c7ced7cc791, 0xcc716666856617e3,
        0xa77bdddd53dda68e, 0x2eaf17175c17b84b, 0x8e45474701470246, 0x211a9e9e429e84dc,
        0x89d4caca0fca1ec5, 0x5a582d2db42d7599, 0x632ebfbfc6bf9179, 0x0e3f07071c07381b,
        0x47acadad8ead0123, 0xb4b05a5a755aea2f, 0x1bef838336836cb5, 0x66b63333cc3385ff,
        0xc65c636391633ff2, 0x041202020802100a, 0x4993aaaa92aa3938, 0xe2de7171d971afa8,
        0x8dc6c8c807c80ecf, 0x32d119196419c87d, 0x923b494939497270, 0xaf5fd9d943d9869a,
        0xf931f2f2eff2c31d, 0xdba8e3e3abe34b48, 0xb6b95b5b715be22a, 0x0dbc88881a883492,
        0x293e9a9a529aa4c8, 0x4c0b262698262dbe, 0x64bf3232c8328dfa, 0x7d59b0b0fab0e94a,
        0xcff2e9e983e91b6a, 0x1e770f0f3c0f7833, 0xb733d5d573d5e6a6, 0x1df480803a8074ba,
        0x6127bebec2be997c, 0x87ebcdcd13cd26de, 0x68893434d034bde4, 0x903248483d487a75,
        0xe354ffffdbffab24, 0xf48d7a7af57af78f, 0x3d6490907a90f4ea, 0xbe9d5f5f615fc23e,
        0x403d202080201da0, 0xd00f6868bd6867d5, 0x34ca1a1a681ad072, 0x41b7aeae82ae192c,
        0x757db4b4eab4c95e, 0xa8ce54544d549a19, 0x3b7f93937693ece5, 0x442f222288220daa,
        0xc86364648d6407e9, 0xff2af1f1e3f1db12, 0xe6cc7373d173bfa2, 0x2482121248129O5a,
        0x807a40401d403a5d, 0x1048080820084028, 0x9b95c3c32bc356e8, 0xc5dfecec97ec337b,
        0xab4ddbdb4bdb9690, 0x5fc0a1a1bea1611f, 0x07918d8d0e8d1c83, 0x7ac83d3df43df5c9,
        0x335b97976697ccf1, 0x0000000000000000, 0x83f9cfcf1bcf36d4, 0x566e2b2bac2b4587,
        0xece17676c57697b3, 0x19e68282328264b0, 0xb128d6d67fd6fea9, 0x36c31b1b6c1bd877,
        0x7774b5b5eeb5c15b, 0x43beafaf86af1129, 0xd41d6a6ab56a77df, 0xa0ea50505d50ba0d,
        0x8a5745450945124c, 0xfb38f3f3ebf3cb18, 0x60ad3030c0309df0, 0xc3c4efef9bef2b74,
        0x7eda3f3ffc3fe5c3, 0xaac755554955921c, 0x59dba2a2b2a27910, 0xc9e9eaea8fea0365,
        0xca6a656589650fec, 0x6903babad2bab968, 0x5e4a2f2fbc2f6593, 0x9d8ec0c027c04ee7,
        0xa160dede5fdebe81, 0x38fc1c1c701ce06c, 0xe746fdfdd3fdbb2e, 0x9a1f4d4d294d5264,
        0x397692927292e4e0, 0xeafa7575c9758fbc, 0x0c3606061806301e, 0x09ae8a8a128a2498,
        0x794bb2b2f2b2f940, 0xd185e6e6bfe66359, 0x1c7e0e0e380e7036, 0x3ee71f1f7c1ff863,
        0xc4556262956237f7, 0xb53ad4d477d4eea3, 0x4d81a8a89aa82932, 0x315296966296c4f4,
        0xef62f9f9c3f99b3a, 0x97a3c5c533c566f6, 0x4a102525942535b1, 0xb2ab59597959f220,
        0x15d084842a8454ae, 0xe4c57272d572b7a7, 0x72ec3939e439d5dd, 0x98164c4c2d4c5a61,
        0xbc945e5e655eca3b, 0xf09f7878fd78e785, 0x70e53838e038ddd8, 0x05988c8c0a8c1486,
        0xbf17d1d163d1c6b2, 0x57e4a5a5aea5410b, 0xd9a1e2e2afe2434d, 0xc24e616199612ff8,
        0x7b42b3b3f6b3f145, 0x42342121842115a5, 0x25089c9c4a9c94d6, 0x3cee1e1e781ef066,
        0x8661434311432252, 0x93b1c7c73bc776fc, 0xe54ffcfcd7fcb32b, 0x0824040410042014,
        0xa2e351515951b208, 0x2f2599995e99bcc7, 0xda226d6da96d4fc4, 0x1a650d0d340d6839,
        0xe979fafacffa8335, 0xa369dfdf5bdfb684, 0xfca97e7ee57ed79b, 0x4819242490243db4,
        0x76fe3b3bec3bc5d7, 0x4b9aabab96ab313d, 0x81f0cece1fce3ed1, 0x2299111144118855,
        0x03838f8f068f0c89, 0x9c044e4e254e4a6b, 0x7366b7b7e6b7d151, 0xcbe0ebeb8beb0b60,
        0x78c13c3cf03cfdcc, 0x1ffd81813e817cbf, 0x354094946a94d4fe, 0xf31cf7f7fbf7eb0c,
        0x6f18b9b9deb9a167, 0x268b13134c13985f, 0x58512c2cb02c7d9c, 0xbb05d3d36bd3d6b8,
        0xd38ce7e7bbe76b5c, 0xdc396e6ea56e57cb, 0x95aac4c437c46ef3, 0x061b03030c03180f,
        0xacdc565645568a13, 0x885e44440d441a49, 0xfea07f7fe17fdf9e, 0x4f88a9a99ea92137,
        0x54672a2aa82a4d82, 0x6b0abbbbd6bbb16d, 0x9f87c1c123c146e2, 0xa6f153535153a202,
        0xa572dcdc57dcae8b, 0x16530b0b2c0b5827, 0x27019d9d4e9d9cd3, 0xd82b6c6cad6c47c1,
        0x62a43131c43195f5, 0xe8f37474cd7487b9, 0xf115f6f6fff6e309, 0x8c4c464605460a43,
        0x45a5acac8aac0926, 0x0fb589891e893c97, 0x28b414145014a044, 0xdfbae1e1a3e15b42,
        0x2ca616165816b04e, 0x74f73a3ae83acdd2, 0xd2066969b9696fd0, 0x124109092409482d,
        0xe0d77070dd70a7ad, 0x716fb6b6e2b6d954, 0xbd1ed0d067d0ceb7, 0xc7d6eded93ed3b7e,
        0x85e2cccc17cc2edb, 0x8468424215422a57, 0x2d2c98985a98b4c2, 0x55eda4a4aaa4490e,
        0x50752828a0285d88, 0xb8865c5c6d5cda31, 0xed6bf8f8c7f8933f, 0x11c28686228644a4,
};
```

```
static array<System::UInt64>^ C3 = {
        0x7830d818186018c0, 0xaf462623238c2305, 0xf991b8c6c63fc67e, 0x6fcdfbe8e887e813,
        0xa113cb878726874c, 0x626d11b8b8dab8a9, 0x0502090101040108, 0x6e9e0d4f4f214f42,
        0xee6c9b3636d836ad, 0x0451ffa6a6a2a659, 0xbdb90cd2d26fd2de, 0x06f70ef5f5f3f5fb,
        0x80f2967979f979ef, 0xcede306f6fa16f5f, 0xef3f6d91917e91fc, 0x07a4f852525552aa,
        0xfdc04760609d6027, 0x766535bcbccabc89, 0xcd2b379b9b569bac, 0x8c018a8e8e028e04,
        0x155bd2a3a3b6a371, 0x3c186c0c0c300c60, 0x8af6847b7bf17bff, 0xe16a803535d435b5,
        0x693af51d1d741de8, 0x47ddb3e0e0a7e053, 0xacb321d7d77bd7f6, 0xed999cc2c22fc25e,
        0x965c432e2eb82e6d, 0x7a96294b4b314b62, 0x21e15dfefedffea3, 0x16aed55757415782,
        0x412abd15155415a8, 0xb6eee87777c1779f, 0xeb6e923737dc37a5, 0x56d79ee5e5b3e57b,
        0xd923139f9f469f8c, 0x17fd23f0f0e7f0d3, 0x7f94204a4a354a6a, 0x95a944dada4fda9e,
        0x25b0a258587d58fa, 0xca8fcfc9c903c906, 0x8d527c2929a42955, 0x22145a0a0a280a50,
        0x4f7f50b1b1feb1e1, 0x1a5dc9a0a0baa069, 0xdad6146b6bb16b7f, 0xab17d985852e855c,
        0x73673cbdbdcebd81, 0x34ba8f5d5d695dd2, 0x5020901010401080, 0x03f507f4f4f7f4f3,
        0xc08bddcbcb0bcb16, 0xc67cd33e3ef83eed, 0x110a2d0505140528, 0xe6ce78676781671f,
        0x53d597e4e4b7e473, 0xbb4e0227279c2725, 0x5882734141194132, 0x9d0ba78b8b168b2c,
        0x0153f6a7a7a6a751, 0x94fab27d7de97dcf, 0xfb374995956e95dc, 0x9fad56d8d847d88e,
        0x30eb70fbfbcbfb8b, 0x71c1cdeeee9fee23, 0x91f8bb7c7ced7cc7, 0xe3cc716666856617,
        0x8ea77bdddd53dda6, 0x4b2eaf17175c17b8, 0x468e454747014702, 0xdc211a9e9e429e84,
        0xc589d4caca0fca1e, 0x995a582d2db42d75, 0x79632ebfbfc6bf91, 0x1b0e3f07071c0738,
        0x2347acadad8ead01, 0x2fb4b05a5a755aea, 0xb51bef838336836c, 0xff66b63333cc3385,
        0xf2c65c636391633f, 0x0a04120202080210, 0x384993aaaa92aa39, 0xa8e2de7171d971af,
        0xcf8dc6c8c807c80e, 0x7d32d119196419c8, 0x70923b4949394972, 0x9aaf5fd9d943d986,
        0x1df931f2f2eff2c3, 0x48dba8e3e3abe34b, 0x2ab6b95b5b715be2, 0x920dbc88881a8834,
        0xc8293e9a9a529aa4, 0xbe4c0b262698262d, 0xfa64bf3232c8328d, 0x4a7d59b0b0fab0e9,
        0x6acff2e9e983e91b, 0x331e770f0f3c0f78, 0xa6b733d5d573d5e6, 0xba1df480803a8074,
        0x7c6127bebec2be99, 0xde87ebcdcd13cd26, 0xe468893434d034bd, 0x75903248483d487a,
        0x24e354ffffdbffab, 0x8ff48d7a7af57af7, 0xea3d6490907a90f4, 0x3ebe9d5f5f615fc2,
        0xa0403d202080201d, 0xd5d00f6868bd6867, 0x7234ca1a1a681ad0, 0x2c41b7aeae82ae19,
        0x5e757db4b4eab4c9, 0x19a8ce54544d549a, 0xe53b7f93937693ec, 0xaa442f222288220d,
        0xe9c86364648d6407, 0x12ff2af1f1e3f1db, 0xa2e6cc7373d173bf, 0x5a24821212481290,
        0x5d807a40401d403a, 0x2810480808200840, 0xe89b95c3c32bc356, 0x7bc5dfecec97ec33,
        0x90ab4ddbdb4bdb96, 0x1f5fc0a1a1bea161, 0x8307918d8d0e8d1c, 0xc97ac83d3df43df5,
        0xf1335b97976697cc, 0x0000000000000000, 0xd483f9cfcf1bcf36, 0x87566e2b2bac2b45,
        0xb3ece17676c57697, 0xb019e68282328264, 0xa9b128d6d67fd6fe, 0x7736c31b1b6c1bd8,
        0x5b7774b5b5eeb5c1, 0x2943beafaf86af11, 0xdfd41d6a6ab56a77, 0x0da0ea50505d50ba,
        0x4c8a574545094512, 0x18fb38f3f3ebf3cb, 0xf060ad3030c0309d, 0x74c3c4efef9bef2b,
        0xc37eda3f3ffc3fe5, 0x1caac75555495592, 0x1059dba2a2b2a279, 0x65c9e9eaea8fea03,
        0xecca6a656589650f, 0x686903babad2bab9, 0x935e4a2f2fbc2f65, 0xe79d8ec0c027c04e,
        0x81a160dede5fdebe, 0x6c38fc1c1c701ce0, 0x2ee746fdfdd3fdbb, 0x649a1f4d4d294d52,
        0xe0397692927292e4, 0xbceafa7575c9758f, 0x1e0c360606180630, 0x9809ae8a8a128a24,
        0x40794bb2b2f2b2f9, 0x59d185e6e6bfe663, 0x361c7e0e0e380e70, 0x633ee71f1f7c1ff8,
        0xf7c4556262956237, 0xa3b53ad4d477d4ee, 0x324d81a8a89aa829, 0xf4315296966296c4,
        0x3aef62f9f9c3f99b, 0xf697a3c5c533c566, 0xb14a102525942535, 0x20b2ab59597959f2,
        0xae15d084842a8454, 0xa7e4c57272d572b7, 0xdd72ec3939e439d5, 0x6198164c4c2d4c5a,
        0x3bbc945e5e655eca, 0x85f09f7878fd78e7, 0xd870e53838e038dd, 0x8605988c8c0a8c14,
        0xb2bf17d1d163d1c6, 0x0b57e4a5a5aea541, 0x4dd9a1e2e2afe243, 0xf8c24e616199612f,
        0x457b42b3b3f6b3f1, 0xa542342121842115, 0xd625089c9c4a9c94, 0x663cee1e1e781ef0,
        0x5286614343114322, 0xfc93b1c7c73bc776, 0x2be54ffcfcd7fcb3, 0x1408240404100420,
        0x08a2e351515951b2, 0xc72f2599995e99bc, 0xc4da226d6da96d4f, 0x391a650d0d340d68,
        0x35e979fafacffa83, 0x84a369dfdf5bdfb6, 0x9bfca97e7ee57ed7, 0xb448192424902 43d,
        0xd776fe3b3bec3bc5, 0x3d4b9aabab96ab31, 0xd181f0cece1fce3e, 0x5522991111441188,
        0x8903838f8f068f0c, 0x6b9c044e4e254e4a, 0x517366b7b7e6b7d1, 0x60cbe0ebeb8beb0b,
        0xcc78c13c3cf03cfd, 0xbf1ffd81813e817c, 0xfe354094946a94d4, 0x0cf31cf7f7fbf7eb,
        0x676f18b9b9deb9a1, 0x5f268b13134c1398, 0x9c58512c2cb02c7d, 0xb8bb05d3d36bd3d6,
        0x5cd38ce7e7bbe76b, 0xcbdc396e6ea56e57, 0xf395aac4c437c46e, 0x0f061b03030c0318,
        0x13acdc565645568a, 0x49885e44440d441a, 0x9efea07f7fe17fdf, 0x374f88a9a99ea921,
        0x8254672a2aa82a4d, 0x6d6b0abbbbd6bbb1, 0xe29f87c1c123c146, 0x02a6f153535153a2,
        0x8ba572dcdc57dcae, 0x2716530b0b2c0b58, 0xd327019d9d4e9d9c, 0xc1d82b6c6cad6c47,
        0xf562a43131c43195, 0xb9e8f37474cd7487, 0x09f115f6f6fff6e3, 0x438c4c464605460a,
        0x2645a5acac8aac09, 0x970fb589891e893c, 0x4428b414145014a0, 0x42dfbae1e1a3e15b,
        0x4e2ca616165816b0, 0xd274f73a3ae83acd, 0xd0d2066969b9696f, 0x2d12410909240948,
        0xade0d77070dd70a7, 0x54716fb6b6e2b6d9, 0xb7bd1ed0d067d0ce, 0x7ec7d6eded93ed3b,
        0xdb85e2cccc17cc2e, 0x578468424215422a, 0xc22d2c98985a98b4, 0x0e55eda4a4aaa449,
        0x8850752828a0285d, 0x31b8865c5c6d5cda, 0x3fed6bf8f8c7f893, 0xa411c28686228644,
};
```

```
static array<System::UInt64>^ C4 = {
        0xc07830d818186018, 0x05af462623238c23, 0x7ef991b8c6c63fc6, 0x136fcdfbe8e887e8,
        0x4ca113cb87872687, 0xa9626d11b8b8dab8, 0x0805020901010401, 0x426e9e0d4f4f214f,
        0xadee6c9b3636d836, 0x590451ffa6a6a2a6, 0xdebdb90cd2d26fd2, 0xfb06f70ef5f5f3f5,
        0xef80f2967979f979, 0x5fcede306f6fa16f, 0xfcef3f6d91917e91, 0xaa07a4f852525552,
        0x27fdc04760609d60, 0x89766535bcbccabc, 0xaccd2b379b9b569b, 0x048c018a8e8e028e,
        0x71155bd2a3a3b6a3, 0x603c186c0c0c300c, 0xff8af6847b7bf17b, 0xb5e16a803535d435,
        0xe8693af51d1d741d, 0x5347ddb3e0e0a7e0, 0xf6acb321d7d77bd7, 0x5eed999cc2c22fc2,
        0x6d965c432e2eb82e, 0x627a96294b4b314b, 0xa321e15dfefedffe, 0x8216aed557574157,
        0xa8412abd15155415, 0x9fb6eee87777c177, 0xa5eb6e923737dc37, 0x7b56d79ee5e5b3e5,
        0x8cd923139f9f469f, 0xd317fd23f0f0e7f0, 0x6a7f94204a4a354a, 0x9e95a944dada4fda,
        0xfa25b0a258587d58, 0x06ca8fcfc9c903c9, 0x558d527c2929a429, 0x5022145a0a0a280a,
        0xe14f7f50b1b1feb1, 0x691a5dc9a0a0baa0, 0x7fdad6146b6bb16b, 0x5cab17d985852e85,
        0x8173673cbdbdcebd, 0xd234ba8f5d5d695d, 0x8050209010104010, 0xf303f507f4f4f7f4,
        0x16c08bddcbcb0bcb, 0xedc67cd33e3ef83e, 0x28110a2d05051405, 0x1fe6ce7867678167,
        0x7353d597e4e4b7e4, 0x25bb4e0227279c27, 0x3258827341411941, 0x2c9d0ba78b8b168b,
        0x510153f6a7a7a6a7, 0xcf94fab27d7de97d, 0xdcfb374995956e95, 0x8e9fad56d8d847d8,
        0x8b30eb70fbfbcbfb, 0x2371c1cdeeee9fee, 0xc791f8bb7c7ced7c, 0x17e3cc7166668566,
        0xa68ea77bdddd53dd, 0xb84b2eaf17175c17, 0x02468e4547470147, 0x84dc211a9e9e429e,
        0x1ec589d4caca0fca, 0x75995a582d2db42d, 0x9179632ebfbfc6bf, 0x381b0e3f07071c07,
        0x012347acadad8ead, 0xea2fb4b05a5a755a, 0x6cb51bef83833683, 0x85ff66b63333cc33,
        0x3ff2c65c63639163, 0x100a041202020802, 0x39384993aaaa92aa, 0xafa8e2de7171d971,
        0x0ecf8dc6c8c807c8, 0xc87d32d119196419, 0x7270923b49493949, 0x869aaf5fd9d943d9,
        0xc31df931f2f2eff2, 0x4b48dba8e3e3abe3, 0xe22ab6b95b5b715b, 0x34920dbc88881a88,
        0xa4c8293e9a9a529a, 0x2dbe4c0b26269826, 0x8dfa64bf3232c832, 0xe94a7d59b0b0fab0,
        0x1b6acff2e9e983e9, 0x78331e770f0f3c0f, 0xe6a6b733d5d573d5, 0x74ba1df480803a80,
        0x997c6127bebec2be, 0x26de87ebcdcd13cd, 0xbde468893434d034, 0x7a75903248483d48,
        0xab24e354ffffdbff, 0xf78ff48d7a7af57a, 0xf4ea3d6490907a90, 0xc23ebe9d5f5f615f,
        0x1da0403d20208020, 0x67d5d00f6868bd68, 0xd07234ca1a1a681a, 0x192c41b7aeae82ae,
        0xc95e757db4b4eab4, 0x9a19a8ce54544d54, 0xece53b7f93937693, 0x0daa442f22228822,
        0x07e9c86364648d64, 0xdb12ff2af1f1e3f1, 0xbfa2e6cc7373d173, 0x905a248212124812,
        0x3a5d807a40401d40, 0x4028104808082008, 0x56e89b95c3c32bc3, 0x337bc5dfecec97ec,
        0x9690ab4ddbdb4bdb, 0x611f5fc0a1a1bea1, 0x1c8307918d8d0e8d, 0xf5c97ac83d3df43d,
        0xccf1335b97976697, 0x0000000000000000, 0x36d483f9cfcf1bcf, 0x4587566e2b2bac2b,
        0x97b3ece17676c576, 0x64b019e682823282, 0xfea9b128d6d67fd6, 0xd87736c31b1b6c1b,
        0xc15b7774b5b5eeb5, 0x112943beafaf86af, 0x77dfd41d6a6ab56a, 0xba0da0ea50505d50,
        0x124c8a5745450945, 0xcb18fb38f3f3ebf3, 0x9df060ad3030c030, 0x2b74c3c4efef9bef,
        0xe5c37eda3f3ffc3f, 0x921caac755554955, 0x791059dba2a2b2a2, 0x0365c9e9eaea8fea,
        0x0fecca6a65658965, 0xb9686903babad2ba, 0x65935e4a2f2fbc2f, 0x4ee79d8ec0c027c0,
        0xbe81a160dede5fde, 0xe06c38fc1c1c701c, 0xbb2ee746fdfdd3fd, 0x52649a1f4d4d294d,
        0xe4e0397692927292, 0x8fbceafa7575c975, 0x301e0c3606061806, 0x249809ae8a8a128a,
        0xf940794bb2b2f2b2, 0x6359d185e6e6bfe6, 0x70361c7e0e0e380e, 0xf8633ee71f1f7c1f,
        0x37f7c45562629562, 0xeea3b53ad4d477d4, 0x29324d81a8a89aa8, 0xc4f4315296966296,
        0x9b3aef62f9f9c3f9, 0x66f697a3c5c533c5, 0x35b14a1025259425, 0xf220b2ab59597959,
        0x54ae15d084842a84, 0xb7a7e4c57272d572, 0xd5dd72ec3939e439, 0x5a6198164c4c2d4c,
        0xca3bbc945e5e655e, 0xe785f09f7878fd78, 0xddd870e53838e038, 0x148605988c8c0a8c,
        0xc6b2bf17d1d163d1, 0x410b57e4a5a5aea5, 0x434dd9a1e2e2afe2, 0x2ff8c24e61619961,
        0xf1457b42b3b3f6b3, 0x15a5423421218421, 0x94d625089c9c4a9c, 0xf0663cee1e1e781e,
        0x2252866143431143, 0x76fc93b1c7c73bc7, 0xb32be54ffcfcd7fc, 0x2014082404041004,
        0xb208a2e351515951, 0xbcc72f2599995e99, 0x4fc4da226d6da96d, 0x68391a650d0d340d,
        0x8335e979fafacffa, 0xb684a369dfdf5bdf, 0xd79bfca97e7ee57e, 0x3db4481924249024,
        0xc5d776fe3b3bec3b, 0x313d4b9aabab96ab, 0x3ed181f0cece1fce, 0x8855229911114411,
        0x0c8903838f8f068f, 0x4a6b9c044e4e254e, 0xd1517366b7b7e6b7, 0x0b60cbe0ebeb8beb,
        0xfdcc78c13c3cf03c, 0x7cbf1ffd81813e81, 0xd4fe354094946a94, 0xeb0cf31cf7f7fbf7,
        0xa1676f18b9b9deb9, 0x985f268b13134c13, 0x7d9c58512c2cb02c, 0xd6b8bb05d3d36bd3,
        0x6b5cd38ce7e7bbe7, 0x57cbdc396e6ea56e, 0x6ef395aac4c437c4, 0x180f061b03030c03,
        0x8a13acdc56564556, 0x1a49885e44440d44, 0xdf9efea07f7fe17f, 0x21374f88a9a99ea9,
        0x4d8254672a2aa82a, 0xb16d6b0abbbbd6bb, 0x46e29f87c1c123c1, 0xa202a6f153535153,
        0xae8ba572dcdc57dc, 0x582716530b0b2c0b, 0x9cd327019d9d4e9d, 0x47c1d82b6c6cad6c,
        0x95f562a43131c431, 0x87b9e8f37474cd74, 0xe309f115f6f6fff6, 0x0a438c4c46460546,
        0x092645a5acac8aac, 0x3c970fb589891e89, 0xa04428b414145014, 0x5b42dfbae1e1a3e1,
        0xb04e2ca616165816, 0xcdd274f73a3ae83a, 0x6fd0d2066969b969, 0x482d124109092409,
        0xa7ade0d77070dd70, 0xd954716fb6b6e2b6, 0xceb7bd1ed0d067d0, 0x3b7ec7d6eded93ed,
        0x2edb85e2cccc17cc, 0x2a57846842421542, 0xb4c22d2c98985a98, 0x490e55eda4a4aaa4,
        0x5d8850752828a028, 0xda31b8865c5c6d5c, 0x933fed6bf8f8c7f8, 0x44a411c286862286,
};
```

```
static array<System::UInt64>^ C5 = {
        0x18c07830d8181860, 0x2305af462623238c, 0xc67ef991b8c6c63f, 0xe8136fcdfbe8e887,
        0x874ca113cb878726, 0xb8a9626d11b8b8da, 0x0108050209010104, 0x4f426e9e0d4f4f21,
        0x36adee6c9b3636d8, 0xa6590451ffa6a6a2, 0xd2debdb90cd2d26f, 0xf5fb06f70ef5f5f3,
        0x79ef80f2967979f9, 0x6f5fcede306f6fa1, 0x91fcef3f6d91917e, 0x52aa07a4f8525255,
        0x6027fdc04760609d, 0xbc89766535bcbcca, 0x9baccd2b379b9b56, 0x8e048c018a8e8e02,
        0xa371155bd2a3a3b6, 0x0c603c186c0c0c30, 0x7bff8af6847b7bf1, 0x35b5e16a803535d4,
        0x1de8693af51d1d74, 0xe05347ddb3e0e0a7, 0xd7f6acb321d7d77b, 0xc25eed999cc2c22f,
        0x2e6d965c432e2eb8, 0x4b627a96294b4b31, 0xfea321e15dfefedf, 0x578216aed5575741,
        0x15a8412abd151554, 0x779fb6eee87777c1, 0x37a5eb6e923737dc, 0xe57b56d79ee5e5b3,
        0x9f8cd923139f9f46, 0xf0d317fd23f0f0e7, 0x4a6a7f94204a4a35, 0xda9e95a944dada4f,
        0x58fa25b0a258587d, 0xc906ca8fcfc9c903, 0x29558d527c2929a4, 0x0a5022145a0a0a28,
        0xb1e14f7f50b1b1fe, 0xa0691a5dc9a0a0ba, 0x6b7fdad6146b6bb1, 0x855cab17d985852e,
        0xbd8173673cbdbdce, 0x5dd234ba8f5d5d69, 0x1080502090101040, 0xf4f303f507f4f4f7,
        0xcb16c08bddcbcb0b, 0x3eedc67cd33e3ef8, 0x0528110a2d050514, 0x671fe6ce78676781,
        0xe47353d597e4e4b7, 0x2725bb4e0227279c, 0x4132588273414119, 0x8b2c9d0ba78b8b16,
        0xa7510153f6a7a7a6, 0x7dcf94fab27d7de9, 0x95dcfb374995956e, 0xd88e9fad56d8d847,
        0xfb8b30eb70fbfbcb, 0xee2371c1cdeeee9f, 0x7cc791f8bb7c7ced, 0x6617e3cc71666685,
        0xdda68ea77bdddd53, 0x17b84b2eaf17175c, 0x4702468e45474701, 0x9e84dc211a9e9e42,
        0xca1ec589d4caca0f, 0x2d75995a582d2db4, 0xbf9179632ebfbfc6, 0x07381b0e3f07071c,
        0xad012347acadad8e, 0x5aea2fb4b05a5a75, 0x836cb51bef838336, 0x3385ff66b63333cc,
        0x633ff2c65c636391, 0x02100a0412020208, 0xaa39384993aaaa92, 0x71afa8e2de7171d9,
        0xc80ecf8dc6c8c807, 0x19c87d32d1191964, 0x497270923b494939, 0xd9869aaf5fd9d943,
        0xf2c31df931f2f2ef, 0xe34b48dba8e3e3ab, 0x5be22ab6b95b5b71, 0x8834920dbc88881a,
        0x9aa4c8293e9a9a52, 0x262dbe4c0b262698, 0x328dfa64bf3232c8, 0xb0e94a7d59b0b0fa,
        0xe91b6acff2e9e983, 0x0f78331e770f0f3c, 0xd5e6a6b733d5d573, 0x8074ba1df480803a,
        0xbe997c6127bebec2, 0xcd26de87ebcdcd13, 0x34bde468893434d0, 0x487a75903248483d,
        0xffab24e354ffffdb, 0x7af78ff48d7a7af5, 0x90f4ea3d6490907a, 0x5fc23ebe9d5f5f61,
        0x201da0403d202080, 0x6867d5d00f6868bd, 0x1ad07234ca1a1a68, 0xae192c41b7aeae82,
        0xb4c95e757db4b4ea, 0x549a19a8ce54544d, 0x93ece53b7f939376, 0x220daa442f222288,
        0x6407e9c86364648d, 0xf1db12ff2af1f1e3, 0x73bfa2e6cc7373d1, 0x12905a2482121248,
        0x403a5d807a40401d, 0x0840281048080820, 0xc356e89b95c3c32b, 0xec337bc5dfecec97,
        0xdb9690ab4ddbdb4b, 0xa1611f5fc0a1a1be, 0x8d1c8307918d8d0e, 0x3df5c97ac83d3df4,
        0x97ccf1335b979766, 0x0000000000000000, 0xcf36d483f9cfcf1b, 0x2b4587566e2b2bac,
        0x7697b3ece17676c5, 0x8264b019e6828232, 0xd6fea9b128d6d67f, 0x1bd87736c31b1b6c,
        0xb5c15b7774b5b5ee, 0xaf112943beafaf86, 0x6a77dfd41d6a6ab5, 0x50ba0da0ea50505d,
        0x45124c8a57454509, 0xf3cb18fb38f3f3eb, 0x309df060ad3030c0, 0xef2b74c3c4efef9b,
        0x3fe5c37eda3f3ffc, 0x55921caac7555549, 0xa2791059dba2a2b2, 0xea0365c9e9eaea8f,
        0x650fecca6a656589, 0xbab9686903babad2, 0x2f65935e4a2f2fbc, 0xc04ee79d8ec0c027,
        0xdebe81a160dede5f, 0x1ce06c38fc1c1c70, 0xfdbb2ee746fdfdd3, 0x4d52649a1f4d4d29,
        0x92e4e03976929272, 0x758fbceafa7575c9, 0x06301e0c36060618, 0x8a249809ae8a8a12,
        0xb2f940794bb2b2f2, 0xe66359d185e6e6bf, 0x0e70361c7e0e0e38, 0x1ff8633ee71f1f7c,
        0x6237f7c455626295, 0xd4eea3b53ad4d477, 0xa829324d81a8a89a, 0x96c4f43152969662,
        0xf99b3aef62f9f9c3, 0xc566f697a3c5c533, 0x2535b14a10252594, 0x59f220b2ab595979,
        0x8454ae15d084842a, 0x72b7a7e4c57272d5, 0x39d5dd72ec3939e4, 0x4c5a6198164c4c2d,
        0x5eca3bbc945e5e65, 0x78e785f09f7878fd, 0x38ddd870e53838e0, 0x8c148605988c8c0a,
        0xd1c6b2bf17d1d163, 0xa5410b57e4a5a5ae, 0xe2434dd9a1e2e2af, 0x612ff8c24e616199,
        0xb3f1457b42b3b3f6, 0x2115a54234212184, 0x9c94d625089c9c4a, 0x1ef0663cee1e1e78,
        0x4322528661434311, 0xc776fc93b1c7c73b, 0xfcb32be54ffcfcd7, 0x0420140824040410,
        0x51b208a2e3515159, 0x99bcc72f2599995e, 0x6d4fc4da226d6da9, 0x0d68391a650d0d34,
        0xfa8335e979fafacf, 0xdfb684a369dfdf5b, 0x7ed79bfca97e7ee5, 0x243db44819242490,
        0x3bc5d776fe3b3bec, 0xab313d4b9aabab96, 0xce3ed181f0cece1f, 0x1188552299111144,
        0x8f0c8903838f8f06, 0x4e4a6b9c044e4e25, 0xb7d1517366b7b7e6, 0xeb0b60cbe0ebeb8b,
        0x3cfdcc78c13c3cf0, 0x817cbf1ffd81813e, 0x94d4fe354094946a, 0xf7eb0cf31cf7f7fb,
        0xb9a1676f18b9b9de, 0x13985f268b13134c, 0x2c7d9c58512c2cb0, 0xd3d6b8bb05d3d36b,
        0xe76b5cd38ce7e7bb, 0x6e57cbdc396e6ea5, 0xc46ef395aac4c437, 0x03180f061b03030c,
        0x568a13acdc565645, 0x441a49885e44440d, 0x7fdf9efea07f7fe1, 0xa921374f88a9a99e,
        0x2a4d8254672a2aa8, 0xbbb16d6b0abbbbd6, 0xc146e29f87c1c123, 0x53a202a6f1535351,
        0xdcae8ba572dcdc57, 0x0b582716530b0b2c, 0x9d9cd327019d9d4e, 0x6c47c1d82b6c6cad,
        0x3195f562a43131c4, 0x7487b9e8f37474cd, 0xf6e309f115f6f6ff, 0x460a438c4c464605,
        0xac092645a5acac8a, 0x893c970fb589891e, 0x14a04428b4141450, 0xe15b42dfbae1e1a3,
        0x16b04e2ca6161658, 0x3acdd274f73a3ae8, 0x696fd0d2066969b9, 0x09482d1241090924,
        0x70a7ade0d77070dd, 0xb6d954716fb6b6e2, 0xd0ceb7bd1ed0d067, 0xed3b7ec7d6eded93,
        0xcc2edb85e2cccc17, 0x422a578468424215, 0x98b4c22d2c98985a, 0xa4490e55eda4a4aa,
        0x285d8850752828a0, 0x5cda31b8865c5c6d, 0xf8933fed6bf8f8c7, 0x8644a411c2868622,
};
```

```
static array<System::UInt64>^ C6 = {
        0x6018c07830d81818, 0x8c2305af46262323, 0x3fc67ef991b8c6c6, 0x87e8136fcdfbe8e8,
        0x26874ca113cb8787, 0xdab8a9626d11b8b8, 0x0401080502090101, 0x214f426e9e0d4f4f,
        0xd836adee6c9b3636, 0xa2a6590451ffa6a6, 0x6fd2debdb90cd2d2, 0xf3f5fb06f70ef5f5,
        0xf979ef80f2967979, 0xa16f5fcede306f6f, 0x7e91fcef3f6d9191, 0x5552aa07a4f85252,
        0x9d6027fdc0476060, 0xcabc89766535bcbc, 0x569baccd2b379b9b, 0x028e048c018a8e8e,
        0xb6a371155bd2a3a3, 0x300c603c186c0c0c, 0xf17bff8af6847b7b, 0xd435b5e16a803535,
        0x741de8693af51d1d, 0xa7e05347ddb3e0e0, 0x7bd7f6acb321d7d7, 0x2fc25eed999cc2c2,
        0xb82e6d965c432e2e, 0x314b627a96294b4b, 0xdffea321e15dfefe, 0x41578216aed55757,
        0x5415a8412abd1515, 0xc1779fb6eeee87777, 0xdc37a5eb6e923737, 0xb3e57b56d79ee5e5,
        0x469f8cd923139f9f, 0xe7f0d317fd23f0f0, 0x354a6a7f94204a4a, 0x4fda9e95a944dada,
        0x7d58fa25b0a25858, 0x03c906ca8fcfc9c9, 0xa429558d527c2929, 0x280a5022145a0a0a,
        0xfeb1e14f7f50b1b1, 0xbaa0691a5dc9a0a0, 0xb16b7fdad6146b6b, 0x2e855cab17d98585,
        0xcebd8173673cbdbd, 0x695dd234ba8f5d5d, 0x4010805020901010, 0xf7f4f303f507f4f4,
        0x0bcb16c08bddcbcb, 0xf83eedc67cd33e3e, 0x140528110a2d0505, 0x81671fe6ce786767,
        0xb7e47353d597e4e4, 0x9c2725bb4e022727, 0x1941325882734141, 0x168b2c9d0ba78b8b,
        0xa6a7510153f6a7a7, 0xe97dcf94fab27d7d, 0x6e95dcfb37499595, 0x47d88e9fad56d8d8,
        0xcbfb8b30eb70fbfb, 0x9fee2371c1cdeeee, 0xed7cc791f8bb7c7c, 0x856617e3cc716666,
        0x53dda68ea77bdddd, 0x5c17b84b2eaf1717, 0x014702468e454747, 0x429e84dc211a9e9e,
        0x0fca1ec589d4caca, 0xb42d75995a582d2d, 0xc6bf9179632ebfbf, 0x1c07381b0e3f0707,
        0x8ead012347acadad, 0x755aea2fb4b05a5a, 0x36836cb51bef8383, 0xcc3385ff66b63333,
        0x91633ff2c65c6363, 0x0802100a04120202, 0x92aa39384993aaaa, 0xd971afa8e2de7171,
        0x07c80ecf8dc6c8c8, 0x6419c87d32d11919, 0x39497270923b4949, 0x43d9869aaf5fd9d9,
        0xeff2c31df931f2f2, 0xabe34b48dba8e3e3, 0x715be22ab6b95b5b, 0x1a8834920dbc8888,
        0x529aa4c8293e9a9a, 0x98262dbe4c0b2626, 0xc8328dfa64bf3232, 0xfab0e94a7d59b0b0,
        0x83e91b6acff2e9e9, 0x3c0f78331e770f0f, 0x73d5e6a6b733d5d5, 0x3a8074ba1df48080,
        0xc2be997c6127bebe, 0x13cd26de87ebcdcd, 0xd034bde468893434, 0x3d487a7590324848,
        0xdbffab24e354ffff, 0xf57af78ff48d7a7a, 0x7a90f4ea3d649090, 0x615fc23ebe9d5f5f,
        0x80201da0403d2020, 0xbd6867d5d00f6868, 0x681ad07234ca1a1a, 0x82ae192c41b7aeae,
        0xeab4c95e757db4b4, 0x4d549a19a8ce5454, 0x7693ece53b7f9393, 0x88220daa442f2222,
        0x8d6407e9c8636464, 0xe3f1db12ff2af1f1, 0xd173bfa2e6cc7373, 0x4812905a24821212,
        0x1d403a5d807a4040, 0x2008402810480808, 0x2bc356e89b95c3c3, 0x97ec337bc5dfecec,
        0x4bdb9690ab4ddbdb, 0xbea1611f5fc0a1a1, 0x0e8d1c8307918d8d, 0xf43df5c97ac83d3d,
        0x6697ccf1335b9797, 0x0000000000000000, 0x1bcf36d483f9cfcf, 0xac2b4587566e2b2b,
        0xc57697b3ece17676, 0x328264b019e68282, 0x7fd6fea9b128d6d6, 0x6c1bd87736c31b1b,
        0xeeb5c15b7774b5b5, 0x86af112943beafaf, 0xb56a77dfd41d6a6a, 0x5d50ba0da0ea5050,
        0x0945124c8a574545, 0xebf3cb18fb38f3f3, 0xc0309df060ad3030, 0x9bef2b74c3c4efef,
        0xfc3fe5c37eda3f3f, 0x4955921caac75555, 0xb2a2791059dba2a2, 0x8fea0365c9e9eaea,
        0x89650fecca6a6565, 0xd2bab9686903baba, 0xbc2f65935e4a2f2f, 0x27c04ee79d8ec0c0,
        0x5fdebe81a160dede, 0x701ce06c38fc1c1c, 0xd3fdbb2ee746fdfd, 0x294d52649a1f4d4d,
        0x7292e4e039769292, 0xc9758fbceafa7575, 0x1806301e0c360606, 0x128a249809ae8a8a,
        0xf2b2f940794bb2b2, 0xbfe66359d185e6e6, 0x380e70361c7e0e0e, 0x7c1ff8633ee71f1f,
        0x956237f7c4556262, 0x77d4eea3b53ad4d4, 0x9aa829324d81a8a8, 0x6296c4f431529696,
        0xc3f99b3aef62f9f9, 0x33c566f697a3c5c5, 0x942535b14a102525, 0x7959f220b2ab5959,
        0x2a8454ae15d08484, 0xd572b7a7e4c57272, 0xe439d5dd72ec3939, 0x2d4c5a6198164c4c,
        0x655eca3bbc945e5e, 0xfd78e785f09f7878, 0xe038ddd870e53838, 0x0a8c148605988c8c,
        0x63d1c6b2bf17d1d1, 0xaea5410b57e4a5a5, 0xafe2434dd9a1e2e2, 0x99612ff8c24e6161,
        0xf6b3f1457b42b3b3, 0x842115a542342121, 0x4a9c94d625089c9c, 0x781ef0663cee1e1e,
        0x1143225286614343, 0x3bc776fc93b1c7c7, 0xd7fcb32be54ffcfc, 0x1004201408240404,
        0x5951b208a2e35151, 0x5e99bcc72f259999, 0xa96d4fc4da226d6d, 0x340d68391a650d0d,
        0xcffa8335e979fafa, 0x5bdfb684a369dfdf, 0xe57ed79bfca97e7e, 0x90243db448192424,
        0xec3bc5d776fe3b3b, 0x96ab313d4b9aabab, 0x1fce3ed181f0cece, 0x4411885522991111,
        0x068f0c8903838f8f, 0x254e4a6b9c044e4e, 0xe6b7d1517366b7b7, 0x8beb0b60cbe0ebeb,
        0xf03cfdcc78c13c3c, 0x3e817cbf1ffd8181, 0x6a94d4fe35409494, 0xfbf7eb0cf31cf7f7,
        0xdeb9a1676f18b9b9, 0x4c13985f268b1313, 0xb02c7d9c58512c2c, 0x6bd3d6b8bb05d3d3,
        0xbbe76b5cd38ce7e7, 0xa56e57cbdc396e6e, 0x37c46ef395aac4c4, 0x0c03180f061b0303,
        0x45568a13acdc5656, 0x0d441a49885e4444, 0xe17fdf9efea07f7f, 0x9ea921374f88a9a9,
        0xa82a4d8254672a2a, 0xd6bbb16d6b0abbbb, 0x23c146e29f87c1c1, 0x5153a202a6f15353,
        0x57dcae8ba572dcdc, 0x2c0b582716530b0b, 0x4e9d9cd327019d9d, 0xad6c47c1d82b6c6c,
        0xc43195f562a43131, 0xcd7487b9e8f37474, 0xfff6e309f115f6f6, 0x05460a438c4c4646,
        0x8aac092645a5acac, 0x1e893c970fb58989, 0x5014a04428b41414, 0xa3e15b42dfbae1e1,
        0x5816b04e2ca61616, 0xe83acdd274f73a3a, 0xb9696fd0d2066969, 0x2409482d12410909,
        0xdd70a7ade0d77070, 0xe2b6d954716fb6b6, 0x67d0ceb7bd1ed0d0, 0x93ed3b7ec7d6eded,
        0x17cc2edb85e2cccc, 0x15422a5784684242, 0x5a98b4c22d2c9898, 0xaaa4490e55eda4a4,
        0xa0285d8850752828, 0x6d5cda31b8865c5c, 0xc7f8933fed6bf8f8, 0x228644a411c28686,
};
```

```
static array<System::UInt64>^ C7 = {
        0x186018c07830d818, 0x238c2305af462623, 0xc63fc67ef991b8c6, 0xe887e8136fcdfbe8,
        0x8726874ca113cb87, 0xb8dab8a9626d11b8, 0x0104010805020901, 0x4f214f426e9e0d4f,
        0x36d836adee6c9b36, 0xa6a2a6590451ffa6, 0xd26fd2debdb90cd2, 0xf5f3f5fb06f70ef5,
        0x79f979ef80f29679, 0x6fa16f5fcede306f, 0x917e91fcef3f6d91, 0x525552aa07a4f852,
        0x609d6027fdc04760, 0xbccabc89766535bc, 0x9b569baccd2b379b, 0x8e028e048c018a8e,
        0xa3b6a371155bd2a3, 0x0c300c603c186c0c, 0x7bf17bff8af6847b, 0x35d435b5e16a8035,
        0x1d741de8693af51d, 0xe0a7e05347ddb3e0, 0xd77bd7f6acb321d7, 0xc22fc25eed999cc2,
        0x2eb82e6d965c432e, 0x4b314b627a96294b, 0xfedffea321e15dfe, 0x5741578216aed557,
        0x155415a8412abd15, 0x77c1779fb6eee877, 0x37dc37a5eb6e9237, 0xe5b3e57b56d79ee5,
        0x9f469f8cd923139f, 0xf0e7f0d317fd23f0, 0x4a354a6a7f94204a, 0xda4fda9e95a944da,
        0x587d58fa25b0a258, 0xc903c906ca8fcfc9, 0x29a429558d527c29, 0x0a280a5022145a0a,
        0xb1feb1e14f7f50b1, 0xa0baa0691a5dc9a0, 0x6bb16b7fdad6146b, 0x852e855cab17d985,
        0xbdcebd8173673cbd, 0x5d695dd234ba8f5d, 0x1040108050209010, 0xf4f7f4f303f507f4,
        0xcb0bcb16c08bddcb, 0x3ef83eedc67cd33e, 0x05140528110a2d05, 0x6781671fe6ce7867,
        0xe4b7e47353d597e4, 0x279c2725bb4e0227, 0x4119413258827341, 0x8b168b2c9d0ba78b,
        0xa7a6a7510153f6a7, 0x7de97dcf94fab27d, 0x956e95dcfb374995, 0xd847d88e9fad56d8,
        0xfbcbfb8b30eb70fb, 0xee9fee2371c1cdee, 0x7ced7cc791f8bb7c, 0x66856617e3cc7166,
        0xdd53dda68ea77bdd, 0x175c17b84b2eaf17, 0x47014702468e4547, 0x9e429e84dc211a9e,
        0xca0fca1ec589d4ca, 0x2db42d75995a582d, 0xbfc6bf9179632ebf, 0x071c07381b0e3f07,
        0xad8ead012347acad, 0x5a755aea2fb4b05a, 0x8336836cb51bef83, 0x33cc3385ff66b633,
        0x6391633ff2c65c63, 0x020802100a041202, 0xaa92aa39384993aa, 0x71d971afa8e2de71,
        0xc807c80ecf8dc6c8, 0x196419c87d32d119, 0x49394972709023b49, 0xd943d9869aaf5fd9,
        0xf2eff2c31df931f2, 0xe3abe34b48dba8e3, 0x5b715be22ab6b95b, 0x881a8834920dbc88,
        0x9a529aa4c8293e9a, 0x2698262dbe4c0b26, 0x32c8328dfa64bf32, 0xb0fab0e94a7d59b0,
        0xe983e91b6acff2e9, 0x0f3c0f78331e770f, 0xd573d5e6a6b733d5, 0x803a8074ba1df480,
        0xbec2be997c6127be, 0xcd13cd26de87ebcd, 0x34d034bde4688934, 0x483d487a75903248,
        0xffdbffab24e354ff, 0x7af57af78ff48d7a, 0x907a90f4ea3d6490, 0x5f615fc23ebe9d5f,
        0x2080201da0403d20, 0x68bd6867d5d00f68, 0x1a681ad07234ca1a, 0xae82ae192c41b7ae,
        0xb4eab4c95e757db4, 0x544d549a19a8ce54, 0x937693ece53b7f93, 0x2288220daa442f22,
        0x648d6407e9c86364, 0xf1e3f1db12ff2af1, 0x73d173bfa2e6cc73, 0x124812905a248212,
        0x401d403a5d807a40, 0x0820084028104808, 0xc32bc356e89b95c3, 0xec97ec337bc5dfec,
        0xdb4bdb9690ab4ddb, 0xa1bea1611f5fc0a1, 0x8d0e8d1c8307918d, 0x3df43df5c97ac83d,
        0x976697ccf1335b97, 0x0000000000000000, 0xcf1bcf36d483f9cf, 0x2bac2b4587566e2b,
        0x76c57697b3ece176, 0x82328264b019e682, 0xd67fd6fea9b128d6, 0x1b6c1bd87736c31b,
        0xb5eeb5c15b7774b5, 0xaf86af112943beaf, 0x6ab56a77dfd41d6a, 0x505d50ba0da0ea50,
        0x450945124c8a5745, 0xf3ebf3cb18fb38f3, 0x30c0309df060ad30, 0xef9bef2b74c3c4ef,
        0x3ffc3fe5c37eda3f, 0x554955921caac755, 0xa2b2a2791059dba2, 0xea8fea0365c9e9ea,
        0x6589650fecca6a65, 0xbad2bab9686903ba, 0x2fbc2f65935e4a2f, 0xc027c04ee79d8ec0,
        0xde5fdebe81a160de, 0x1c701ce06c38fc1c, 0xfdd3fdbb2ee746fd, 0x4d294d52649a1f4d,
        0x927292e4e0397692, 0x75c9758fbceafa75, 0x061806301e0c3606, 0x8a128a249809ae8a,
        0xb2f2b2f940794bb2, 0xe6bfe66359d185e6, 0x0e380e70361c7e0e, 0x1f7c1ff8633ee71f,
        0x62956237f7c45562, 0xd477d4eea3b53ad4, 0xa89aa829324d81a8, 0x966296c4f4315296,
        0xf9c3f99b3aef62f9, 0xc533c566f697a3c5, 0x25942535b14a1025, 0x597959f220b2ab59,
        0x842a8454ae15d084, 0x72d572b7a7e4c572, 0x39e439d5dd72ec39, 0x4c2d4c5a6198164c,
        0x5e655eca3bbc945e, 0x78fd78e785f09f78, 0x38e038ddd870e538, 0x8c0a8c148605988c,
        0xd163d1c6b2bf17d1, 0xa5aea5410b57e4a5, 0xe2afe2434dd9a1e2, 0x6199612ff8c24e61,
        0xb3f6b3f1457b42b3, 0x21842115a5423421, 0x9c4a9c94d625089c, 0x1e781ef0663cee1e,
        0x4311432252866143, 0xc73bc776fc93b1c7, 0xfcd7fcb32be54ffc, 0x0410042014082404,
        0x515951b208a2e351, 0x995e99bcc72f2599, 0x6da96d4fc4da226d, 0x0d340d68391a650d,
        0xfacffa8335e979fa, 0xdf5bdfb684a369df, 0x7ee57ed79bfca97e, 0x2490243db4481924,
        0x3bec3bc5d776fe3b, 0xab96ab313d4b9aab, 0xce1fce3ed181f0ce, 0x1144118855229911,
        0x8f068f0c8903838f, 0x4e254e4a6b9c044e, 0xb7e6b7d1517366b7, 0xeb8beb0b60cbe0eb,
        0x3cf03cfdcc78c13c, 0x813e817cbf1ffd81, 0x946a94d4fe354094, 0xf7fbf7eb0cf31cf7,
        0xb9deb9a1676f18b9, 0x134c13985f268b13, 0x2cb02c7d9c58512c, 0xd36bd3d6b8bb05d3,
        0xe7bbe76b5cd38ce7, 0x6ea56e57cbdc396e, 0xc437c46ef395aac4, 0x030c03180f061b03,
        0x5645568a13acdc56, 0x440d441a49885e44, 0x7fe17fdf9efea07f, 0xa99ea921374f88a9,
        0x2aa82a4d8254672a, 0xbbd6bbb16d6b0abb, 0xc123c146e29f87c1, 0x535153a202a6f153,
        0xdc57dcae8ba572dc, 0x0b2c0b582716530b, 0x9d4e9d9cd327019d, 0x6cad6c47c1d82b6c,
        0x31c43195f562a431, 0x74cd7487b9e8f374, 0xf6fff6e309f115f6, 0x4605460a438c4c46,
        0xac8aac092645a5ac, 0x891e893c970fb589, 0x145014a04428b414, 0xe1a3e15b42dfbae1,
        0x165816b04e2ca616, 0x3ae83acdd274f73a, 0x69b9696fd0d20669, 0x092409482d124109,
        0x70dd70a7ade0d770, 0xb6e2b6d954716fb6, 0xd067d0ceb7bd1ed0, 0xed93ed3b7ec7d6ed,
        0xcc17cc2edb85e2cc, 0x4215422a57846842, 0x985a98b4c22d2c98, 0xa4aaa4490e55eda4,
        0x28a0285d88507528, 0x5c6d5cda31b8865c, 0xf8c7f8933fed6bf8, 0x86228644a411c286,
};
```

Daftar Pustaka

Munir, Rinaldi. (Agustus, 2006). **Diktat Kuliah IF5054 Kriptografi**. Bandung: Institut Teknologi Bandung.

Barreto, Paulo S. L. M. (2006). **The Hash Function Lounge**. http://planeta.terra.com.br/informatica/paulobarreto/hflounge.html *last update: 6 November 2006*.

Barreto, Paulo S. L. M. (2006). **The WHIRLPOOL Hash Function**. http://paginas.terra.com.br/informatica/paulobarreto/WhirlpoolPage.html *last update: 11 October 2006*.

Pramstaller, Norbert, dkk. **A Compact FPGA Implementation of the Hash FunctionWhirlpool**. Austria: Graz University of Technology.

Wikipedia. http://en.wikipedia.org/wiki/Cryptographic_hash_function *last modified 10:20, 11 December 2006*.

Wikipedia. http://en.wikipedia.org/wiki/WHIRLPOOL *last modified 10:20, 11 December 2006*.