The nuts and bolts of SHA256

Bert Douglas
10 April 2017

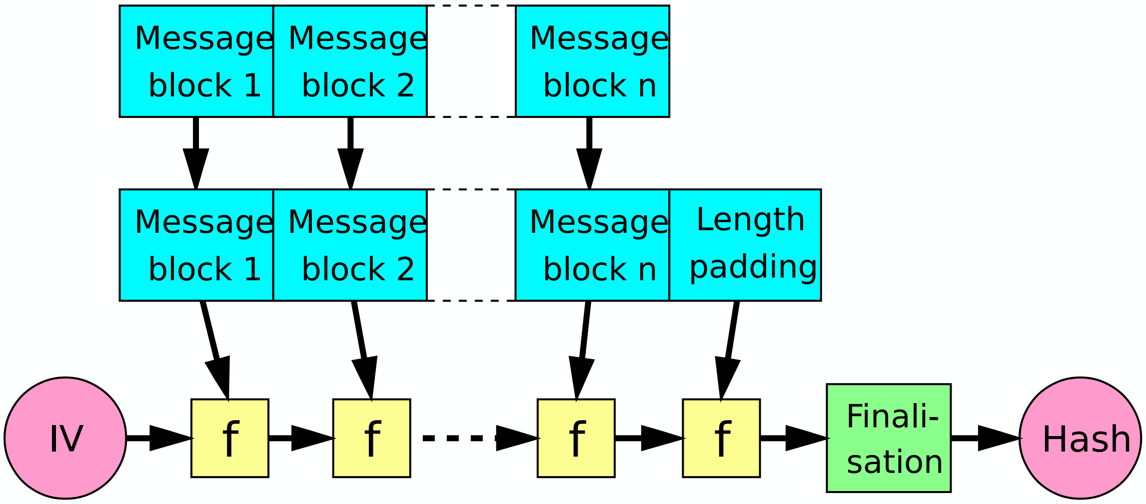
SHA2 Family

- Designed by NSA.
- Published in 2001.
- Only a few mystery constants.
- Merkle–Damgård construction
- Designed for general purpose computers and also for ASICs.

Output size (bits)	Internal state size (bits)	Block size (bits)	Max message size (bits)	Rounds	Operations
(SHA-256) 224 256	256 (8 × 32)	512	$2^{64} - 1$	64	And, Xor, Rot, Add (mod 2^{32}), Or, Shr
(SHA-512) 224 256 384 512	512 (8 × 64)	1024	$2^{128} - 1$	80	And, Xor, Rot, Add (mod 2^{64}), Or, Shr

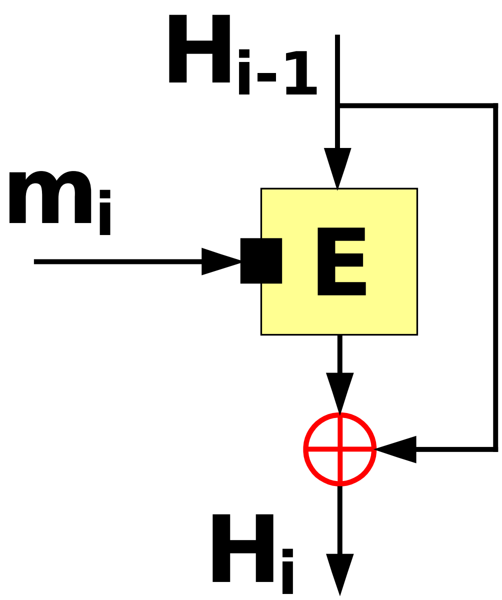
Merkle–Damgård construction

Merkle 1979 Damgård 1989



F is a one-way compression function. Merkle and Damgård proved that if F has good properties (collision resistance), then the chained construction also has good properties. So Merkle and Damgård reduced the problem of finding a good hash function to finding a good compression function.

In SHA-256, this function is of the Davies-Meyer type. This simply means that it iterates for a fixed number of “rounds” over a simpler function, typically based on a block-cypher.

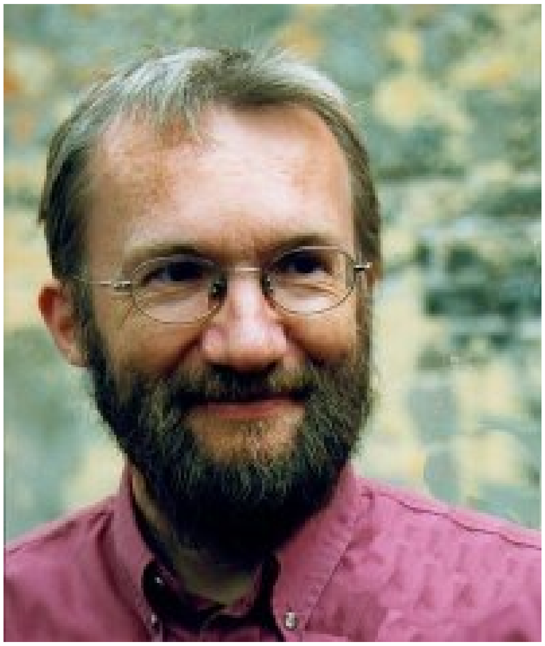


Davies-Meyer one way compression function.

Ralph Merkle

- Born 1952
- BA Computer Science, UC Berkeley, 1974
- PhD Electrical Engineering, Stanford, 1979, “Secrecy, authentication and public key systems”.
- Thesis adviser was Martin Hellman. Together with Whitfield Diffie, the three are considered the inventors of public key cryptography.
- Invented cryptographic hashing, with his PhD thesis. Now called the Merkle–Damgård construction.
- Invented the Merkle tree, which is notably used in Bitcoin. Patented 1979.

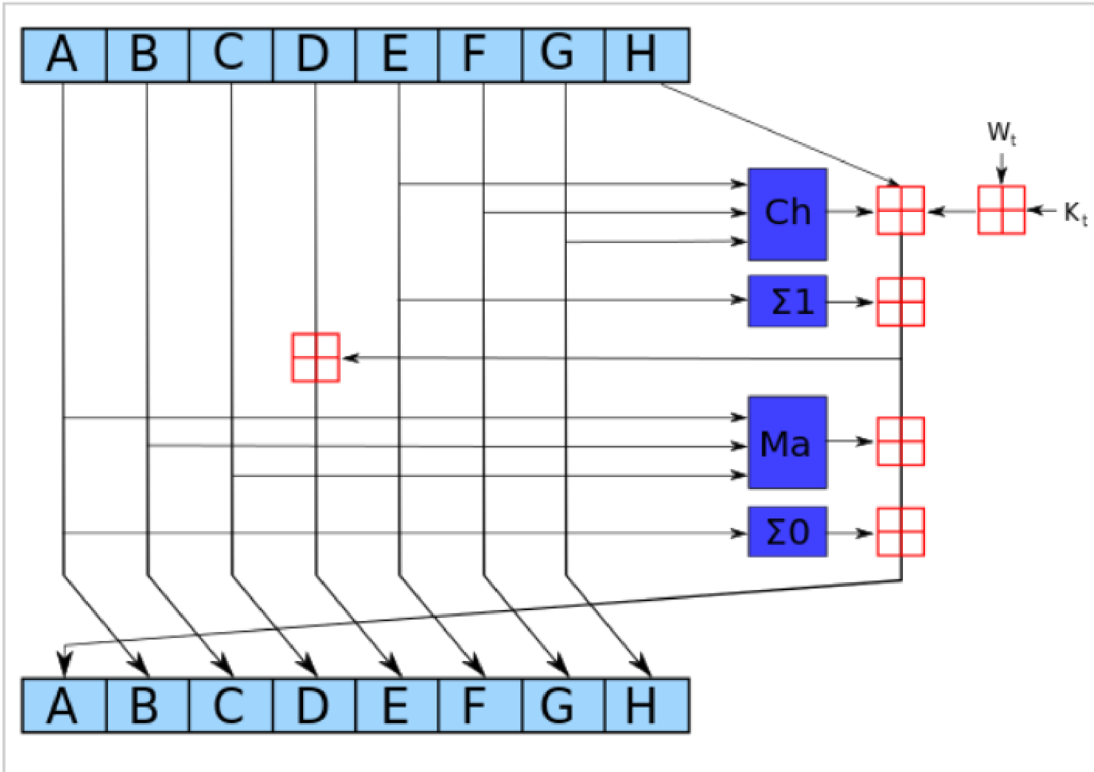
Ivan Bjerre Damgård



- Born 1956
- 1986 Founder Cryptomathic
- Phd 1988 “Unconditional protection in cryptographic protocols”
- 1989 Independently discovered and published M-D hash structure
- 2005 Full professor Department of Computer Science, Aarhus University, Denmark.
- 2010 IACR fellow

SHA-256 Compression Function (logical)

Block diagram from wikipedia



One iteration in a SHA-2 family compression function. The blue components perform the following operations:

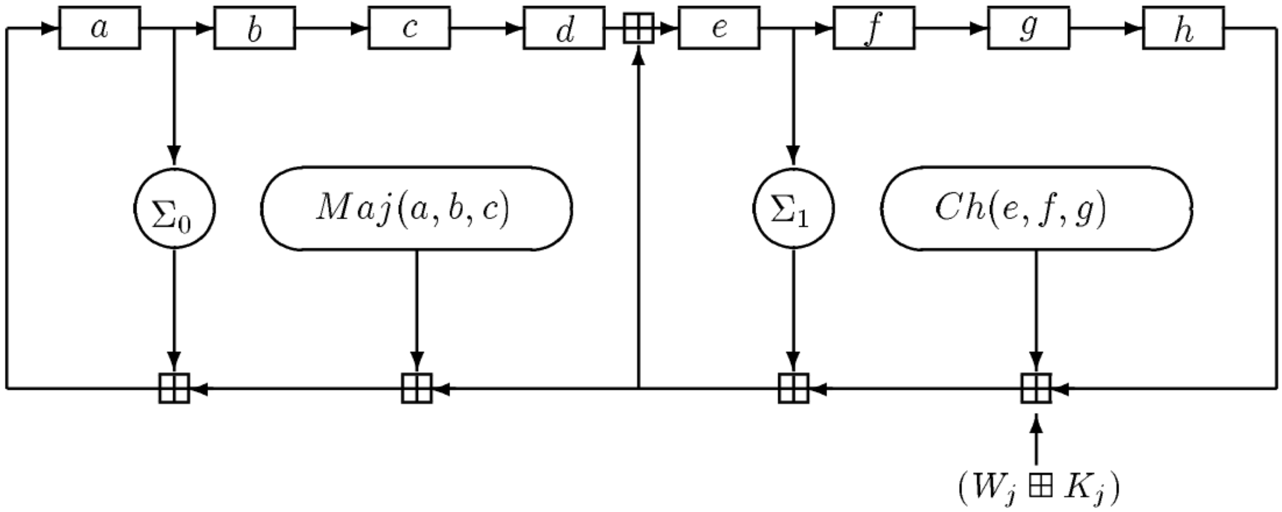
$$\mathrm{Ch}(E,F,G) = (E \land F) \oplus (\neg E \land G)$$
$$\mathrm{Ma}(A,B,C) = (A \land B) \oplus (A \land C) \oplus (B \land C)$$
$$\Sigma_0(A) = (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22)$$
$$\Sigma_1(E) = (E \ggg 6) \oplus (E \ggg 11) \oplus (E \ggg 25)$$

The bitwise rotation uses different constants for SHA-512. The given numbers are for SHA-256.

The red $\boxplus$ is addition modulo 2^{32} for SHA-256, or 2^{64} for SHA-512.

SHA-256 Compression Function (Physical)

Block diagram corresponding to physical layout in ASIC.



SHA-256 Compression function (pseudo-code)

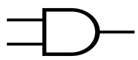
```
Initialize working variables to current hash value:
    a := h0
    b := h1
    c := h2
    d := h3
    e := h4
    f := h5
    g := h6
    h := h7

    Compression function main loop:
    for i from 0 to 63
        S1 := (e rightrotate 6) xor (e rightrotate 11) xor (e rightrotate 25)
        ch := (e and f) xor ((not e) and g)
        temp1 := h + S1 + ch + k[i] + w[i]
        S0 := (a rightrotate 2) xor (a rightrotate 13) xor (a rightrotate 22)
        maj := (a and b) xor (a and c) xor (b and c)
        temp2 := S0 + maj

        h := g
        g := f
        f := e
        e := d + temp1
        d := c
        c := b
        b := a
        a := temp1 + temp2

    Add the compressed chunk to the current hash value:
    h0 := h0 + a
    h1 := h1 + b
    h2 := h2 + c
    h3 := h3 + d
    h4 := h4 + e
    h5 := h5 + f
    h6 := h6 + g
    h7 := h7 + h
```

And Function

Mnemonic form

The output is 1 when all of the inputs are 1.

Also known as:

all	in haskell
	in electronic schematic diagrams
& &&	in C
conjunction ∧	in formal logic
* · ×	in Boolean algebra and electrical engineering equivalent to multiplication
min	in math equivalent to minimum function

Truth table form:

And(in1, in2) → out

in1	in2	out
0	0	0
0	1	0
1	0	0
1	1	1

Sum of products form in C:

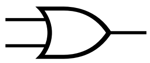
```
out = in1&in2;
```

Or Function

Mnemonic form

The output is 1 when any of the inputs are 1.

Also known as:

any	in haskell
	in electronic schematic diagrams
| ||	in C
disjunction ∨	in formal logic
+	in Boolean algebra and electrical engineering equivalent to addition for two inputs
max	in math equivalent to maximum function

Truth table form:

Or(in1, in2) → out

in1	in2	out
0	0	0
0	1	1
1	0	1
1	1	1

Sum of products form in C:

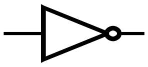
```
out = in1 | in2;
```

Not function

Mnemonic form

The output is the opposite of the input.

Also known as:

not	in scheme and haskell
	in electronic schematic diagrams inverter
~ !	in C
negation ¬	in formal logic
- / $\overline{x}$	in Boolean algebra and electrical engineering

Truth table form:

Not(in) → out

in1	out
0	1
1	0

Sum of products form in C:

```
out = ~in;
```

Exclusive Or Function

Mnemonic form:

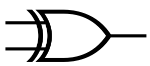
The output is 1 when there are an odd number of 1 inputs.

Truth table form:

Xor(in1, in2) → out

in1	**in2**	**out**
0	0	0
0	1	1
1	0	1
1	1	0

Also known as:

	in electronic schematic diagrams
^	in C
exclusive disjunction ⊻	in formal logic
⊕	in Boolean algebra and electrical engineering
modulo 2 sum	math

Sum of products form in C:

```
out = in1&(~in2) | (~in1)&in2;
```

Exclusive-or (unlike and/or) is linear, information preserving and reversible.

```
a ⊕ b  →  c
c ⊕ b  =  a
```

This is similar to ordinary addition and subtraction. However, exclusive-or acts as its own inverse.

```
a + b  →  c
c − b  =  a
```

Because of this property, exclusive-or is much used in symmetric key ciphers, where the same key is used for both encrypting and decrypting.

Boolean algebra summary

Boolean & DeMorgan's Theorems

1) $X \cdot 0 = 0$

2) $X \cdot 1 = X$

3) $X \cdot X = X$

4) $X \cdot \overline{X} = 0$

5) $X + 0 = X$

6) $X + 1 = 1$

7) $X + X = X$

8) $X + \overline{X} = 1$

9) $\overline{\overline{X}} = X$

$$\left.\begin{array}{ll} \text{10A)} & X \cdot Y = Y \cdot X \\ \text{10B)} & X + Y = Y + X \end{array}\right\} \text{Commutative Law}$$

$$\left.\begin{array}{ll} \text{11A)} & X(YZ) = (XY)Z \\ \text{11B)} & X + (Y + Z) = (X + Y) + Z \end{array}\right\} \text{Associative Law}$$

$$\left.\begin{array}{ll} \text{12A)} & X(Y + Z) = XY + XZ \\ \text{12B)} & (X + Y)(W + Z) = XW + XZ + YW + YZ \end{array}\right\} \text{Distributive Law}$$

$$\left.\begin{array}{ll} \text{13A)} & X + \overline{X}Y = X + Y \\ \text{13B)} & \overline{X} + XY = \overline{X} + Y \\ \text{13C)} & X + \overline{X}\,\overline{Y} = X + \overline{Y} \\ \text{13D)} & \overline{X} + X\overline{Y} = \overline{X} + \overline{Y} \end{array}\right\} \text{Consensus Theorem}$$

$$\left.\begin{array}{ll} \text{14A)} & \overline{X\,Y} = \overline{X} + \overline{Y} \\ \text{14B)} & \overline{X + Y} = \overline{X}\;\overline{Y} \end{array}\right\} \text{DeMorgan's}$$

$$\begin{aligned}
X &= \overline{A\overline{B} \cdot (A + C)} + \overline{A}B \cdot \overline{A + \overline{B} + \overline{C}} \\
&= \overline{A\overline{B}} + \overline{A + C} + \overline{A}B \cdot (\overline{A} \cdot \overline{\overline{B}} \cdot \overline{\overline{C}}) \\
&= (\overline{A} + \overline{\overline{B}}) + \overline{A} \cdot \overline{C} + \overline{A}\,\overline{A}BBC \\
&= \overline{A} + B + \overline{A}\,\overline{C} + \overline{A}BC \\
&= \overline{A}(1 + \overline{C}) + B + \overline{A}BC \\
&= \overline{A} + B + \overline{A}BC \\
&= \overline{A} + B(1 + \overline{A}C) \\
&= \overline{A} + B \quad \leftarrow \textit{simplified equation}
\end{aligned}$$

Majority Function

The output is 1 when the majority of the inputs are 1.

Truth table form:

Maj(in1, in2, in3) → out

in1	in2	in3	out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Sum of products form in C:

```
out = (in1&in2) | (in1&in3) | (in2&in3);
```

Product of sums form in C:

```
out = (in1|in2) & (in1|in3) & (in2|in3);
```

Form given in standards document:

Maj(A,B,C) = (A∧B) ⊕ (A∧C) ⊕ (B∧C)

Equivalent to above, but exclusive or is used instead of regular or. The result is the same, but arrived at by a different path.

Maj(A, B, C) → out

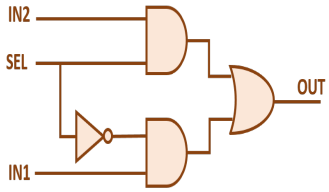
A	B	C	A∧B	A∧C	B∧C	out
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	0	0	0	0
1	0	1	0	1	0	1
1	1	0	1	0	0	1
1	1	1	1	1	1	1

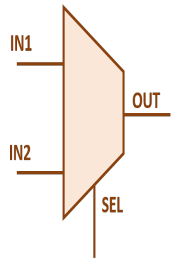
Choose Function

The first input chooses which other input becomes the output

Also known as:

2 input multiplexor





in electronic schematic diagrams

Truth table form:

Ch(sel, in1, in2) → out

sel	in1	in2	out
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Sum of products form in C:

```
out = (~sel&in1) | (sel&in2);
```

Form as given in the standards document:

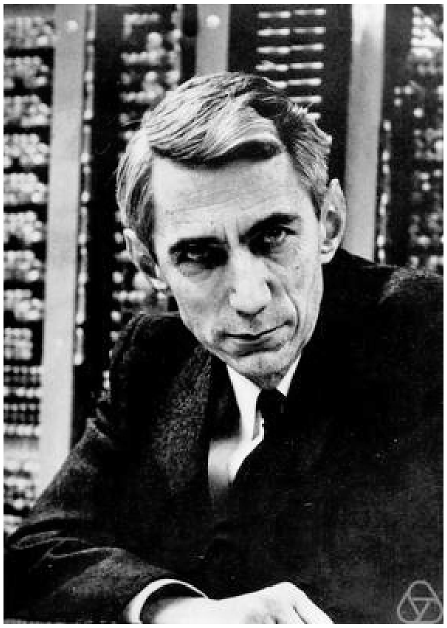
$Ch(E,F,G) = (E \wedge F) \oplus (\neg E \wedge G)$

Equivalent to above. The select input is inverted, and the or is replaced by exclusive or. One of the terms is always zero, so the exclusive or gives the same result as regular or.

Ch(**¬E**, F, G) → out

E	F	G	E∧F	¬E∧G	out
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	1	0	1
1	1	1	1	0	1
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	0	0
0	1	1	0	1	1

Claude Elwood Shannon



Electrical Engineer, Mathematician, Cryptographer
April 30, 1916 – February 24, 2001

Father of logic design
- Master's thesis MIT 1940
- "A symbolic analysis of relay and switching circuits"
- First to apply Boolean algebra to electrical logic circuits
- Put a theoretical foundation under field of logic design

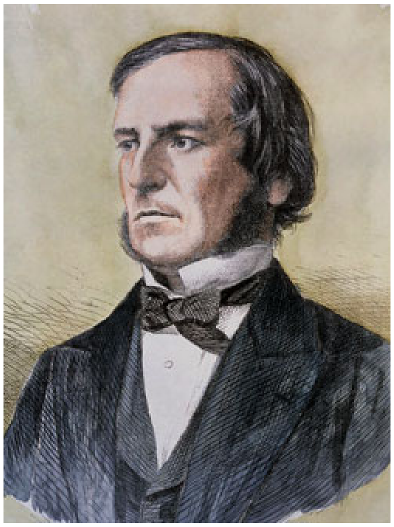
World War II code breaking and secure communications

Father of information theory
- Bell Labs 1948 "A Mathematical Theory of Communication"
- Defined the equivalence between information and thermodynamic entropy
- Invented and defined the term "bit" as a unit of entropy
- Shannon's Law:
  - maximum possible rate of communications in a noisy channel

$$C = B \log_2 \left(1 + \frac{S}{N}\right)$$

A very successful "card counter" in Las Vegas

George Boole



Mathematician, Educator, Philosopher and Logician
2 November 1815 – 8 December 1864

Algebra of Logic

- First to apply techniques of algebra to logic
- Followed the path of Descartes, who applied algebra to geometry
- Replaced the wordy syllogisms of Aristotle by concise symbols and equations
- 1847 “The Mathematical Analysis of Logic”
    - 2-element algebra
    - 0=false 1=true
    - **and** is multiplication denoted by juxtaposition
    - **or** is addition denoted by ‘+’
    - **not x** is denoted by (1-x)
- 1854 “An Investigation of the Laws of Thought”
- Unleashed development of logic as field of mathematics, philosophy, recreation.
- The original 2-element algebra was nearly forgotten by the time of Shannon
- First referred to as Boolean Algebra in 1913

Hexadecimal notation

Bit weight 8421	Decimal	Hex
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

Example 32-bit operations

Ref	Hex	Binary
a	DEADBEEF	1101 1110 1010 1101 1011 1110 1110 1111
~a	21524110	0010 0001 0101 0010 0100 0001 0001 0000
b	CAFEBABE	1100 1010 1111 1110 1011 1010 1011 1110
~b	35014541	0011 0101 0000 0001 0100 0101 0100 0001
c	8BADF00D	1000 1011 1010 1101 1111 0000 0000 1101
~c	74520FF2	0111 0100 0101 0010 0000 1111 1111 0010
d	23456789	0010 0011 0100 0101 0110 0111 1000 1001
~d	DCBA9876	1101 1100 1011 1010 1001 1000 0111 0110
a	DEADBEEF	1101 1110 1010 1101 1011 1110 1110 1111
b	CAFEBABE	1100 1010 1111 1110 1011 1010 1011 1110
a&b	CAACBAAE	1100 1010 1010 1100 1011 1010 1010 1110
c	8BADF00D	1000 1011 1010 1101 1111 0000 0000 1101
d	23456789	0010 0011 0100 0101 0110 0111 1000 1001
c|d	ABEDF78D	1010 1011 1110 1101 1111 0111 1000 1101
b	CAFEBABE	1100 1010 1111 1110 1011 1010 1011 1110
c	8BADF00D	1000 1011 1010 1101 1111 0000 0000 1101
b^c	41534AB3	0100 0001 0101 0011 0100 1010 1011 0011