Practical Aspects of Modern Cryptography

Winter 2011

Josh Benaloh
Brian LaMacchia

Symmetric Cryptography

Agenda

- Symmetric key ciphers
  - Stream ciphers
  - Block ciphers
- Cryptographic hash functions

Agenda

- Symmetric key ciphers
  - Stream ciphers
  - Block ciphers
- Cryptographic hash functions

Public Key vs. Symmetric Key

- Recall from last time that in a public key cryptosystem, each participant has a key pair consisting of related keys

Public Key vs. Symmetric Key

- Recall from last time that in a public key cryptosystem, each participant has a key pair consisting of related keys
  - Alice (or anyone) encrypts to Bob using Bob's public key

Public Key vs. Symmetric Key

- Recall from last time that in a public key cryptosystem, each participant has a key pair consisting of related keys
  - Alice (or anyone) encrypts to Bob using Bob's public key
  - Bob decrypts with Bob's private key

Public Key vs. Symmetric Key

- Recall from last time that in a public key cryptosystem, each participant has a key pair consisting of related keys
  - Alice (or anyone) encrypts to Bob using Bob's public key
  - Bob decrypts with Bob's private key
- Public keys are public

Public Key vs. Symmetric Key

- Recall from last time that in a public key cryptosystem, each participant has a key pair consisting of related keys
  - Alice (or anyone) encrypts to Bob using Bob's public key
  - Bob decrypts with Bob's private key
- Public keys are public
  - They can be published in a directory, on a website, etc.

Public Key vs. Symmetric Key

- In contrast, in a symmetric key system, the same key is used for encryption and decryption

Public Key vs. Symmetric Key

- In contrast, in a symmetric key system, the same key is used for encryption and decryption
  - Such keys must be kept secret

Public Key vs. Symmetric Key

- In contrast, in a symmetric key system, the same key is used for encryption and decryption
  - Such keys must be kept secret
  - Alice and Bob must both know the same secret key to use a symmetric cryptosystem with that key

Public Key vs. Symmetric Key

- In contrast, in a symmetric key system, the same key is used for encryption and decryption
  - Such keys must be kept secret
  - Alice and Bob must both know the same secret key to use a symmetric cryptosystem with that key
- Common pattern we will come back to later:

Public Key vs. Symmetric Key

- In contrast, in a symmetric key system, the same key is used for encryption and decryption
  - Such keys must be kept secret
  - Alice and Bob must both know the same secret key to use a symmetric cryptosystem with that key
- Common pattern we will come back to later:
  - Use a public key cryptosystem to send/negotiate a randomly-generated secret key with the party to whom you wish to communicate

Public Key vs. Symmetric Key

- In contrast, in a symmetric key system, the same key is used for encryption and decryption
  - Such keys must be kept secret
  - Alice and Bob must both know the same secret key to use a symmetric cryptosystem with that key
- Common pattern we will come back to later:
  - Use a public key cryptosystem to send/negotiate a randomly-generated secret key with the party to whom you wish to communicate
  - Then use that secret key with a symmetric key cryptosystem

Symmetric Ciphers

Private-key (symmetric) ciphers are usually divided into two classes.

- Stream ciphers
- Block ciphers

Symmetric Ciphers

Private-key (symmetric) ciphers are usually divided into two classes.

- Stream ciphers
- Block ciphers

Stream Ciphers

- Examples of stream ciphers include RC4, A5/1, SEAL, etc.

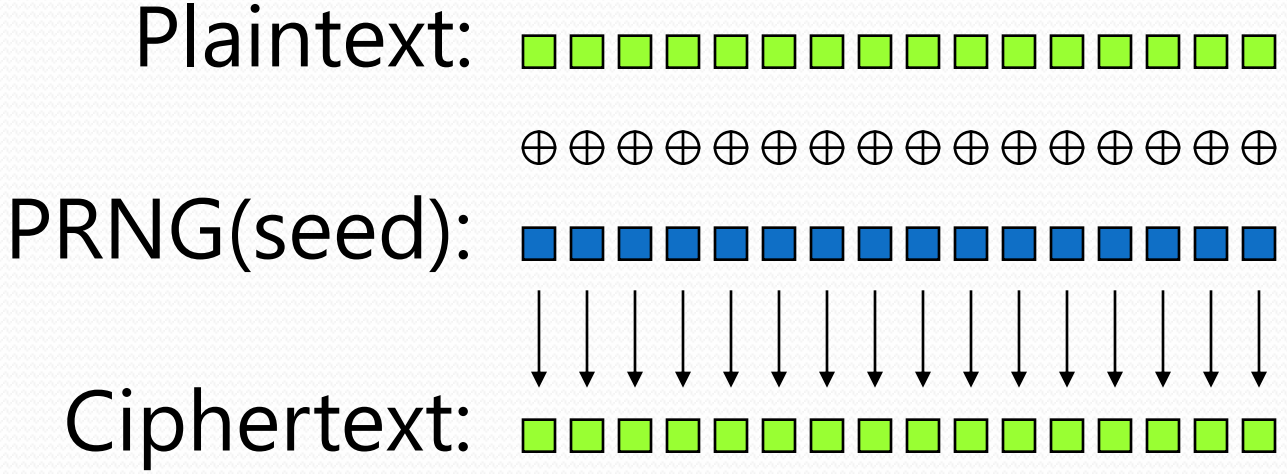
Stream Ciphers

- Examples of stream ciphers include RC4, A5/1, SEAL, etc.
- Use the key as a seed to a pseudo-random number-generator.

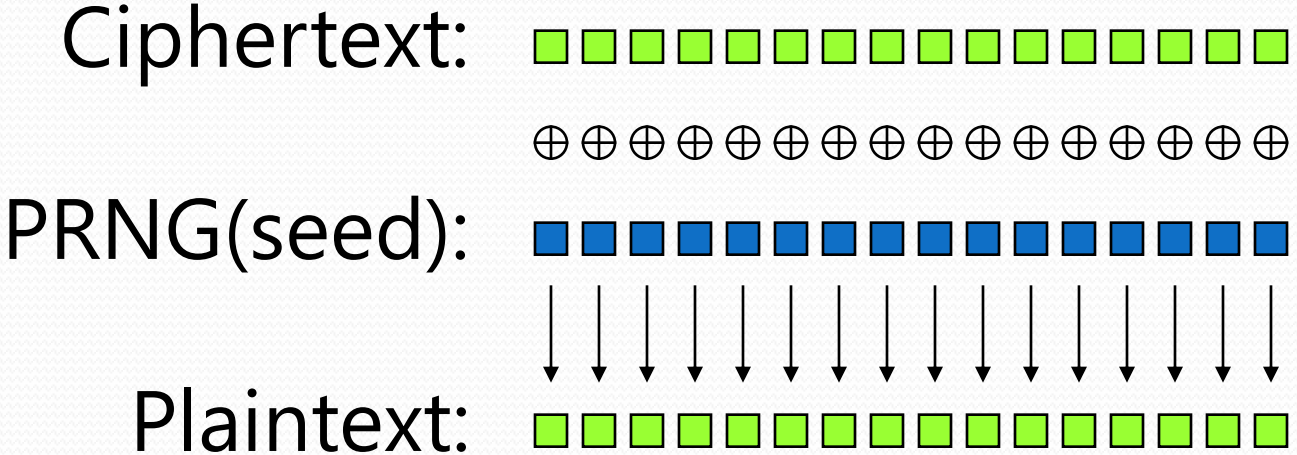
Stream Ciphers

- Examples of stream ciphers include RC4, A5/1, SEAL, etc.
- Use the key as a seed to a pseudo-random number-generator.
- Take the stream of output bits from the PRNG and XOR it with the plaintext to form the ciphertext.

Stream Cipher Encryption



Stream Cipher Decryption



Some Good Properties

- Stream ciphers are typically very fast.

Some Good Properties

- Stream ciphers are typically very fast.
- Stream ciphers can be very simple.

Some Good Properties

- Stream ciphers are typically very fast.
- Stream ciphers can be very simple.
- The same function is used for encryption and decryption.

A Sample PRNG: "Alleged RC4"

Initialization

S[0..255] = 0,1,…,255

K[0..255] = Key,Key,Key,…

for i = 0 to 255

    j = (j + S[i] + K[i]) mod 256

    swap S[i] and S[j]

A Sample PRNG: "Alleged RC4"

Iteration

i = (i + 1) mod 256

j = (j + S[i]) mod 256

swap S[i] and S[j]

t = (S[i] + S[j]) mod 256

Output S[t]

Stream Cipher Security

- If two plaintexts are *ever* encrypted with the same stream cipher and key

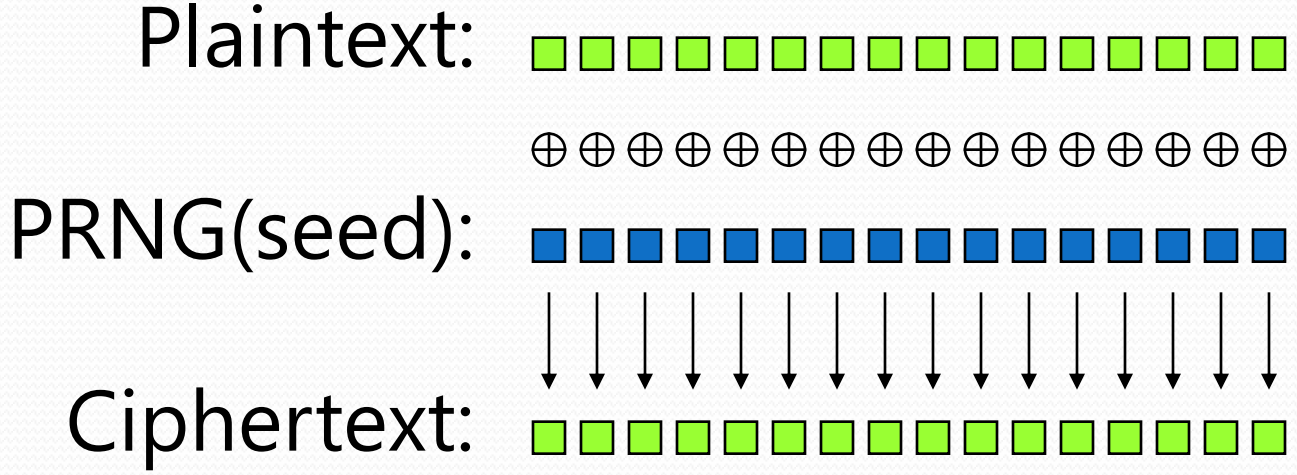
$$C_1 = K \oplus P_1$$
$$C_2 = K \oplus P_2$$

an attacker can easily compute

$$C_1 \oplus C_2 = P_1 \oplus P_2$$

from which P_1 and P_2 can usually be teased apart easily.

Stream Cipher Encryption



Stream Cipher Integrity

- It is easy for an adversary (even one who can't decrypt the ciphertext) to alter the plaintext in a known way.

Bob to Bob's Bank:

Please transfer $0,000,002.00 to the account of my good friend Alice.

Stream Cipher Integrity

- It is easy for an adversary (even one who can't decrypt the ciphertext) to alter the plaintext in a known way.

Bob to Bob's Bank:

Please transfer $1,000,002.00 to the account of my good friend Alice.

Stream Cipher Integrity

- It is easy for an adversary (even one who can't decrypt the ciphertext) to alter the plaintext in a known way.

Bob to Bob's Bank:
Please transfer $1,000,002.00 to the account of my good friend Alice.

- This can be protected against by the careful addition of appropriate redundancy.

Stream Ciphers are Fragile

- They are broken by key re-use.

Stream Ciphers are Fragile

- They are broken by key re-use.
- They require integrity checking.

Stream Ciphers are Fragile

- They are broken by key re-use.
- They require integrity checking.
- If you're going to use a stream cipher
  - Seriously consider other options. Make sure you understand the risks if you make a mistake in use

Symmetric Ciphers

Private-key (symmetric) ciphers are usually divided into two classes.

- Stream ciphers
- Block ciphers

Block Ciphers

- Why are they called “block” ciphers?

Block Ciphers

- Why are they called “block” ciphers?
  - Because the cipher is defined as a function on a fixed-size block of data.

Block Ciphers

- Why are they called “block” ciphers?
  - Because the cipher is defined as a function on a fixed-size block of data.
  - This is called the “block size” and is a fundamental parameter of the cipher.

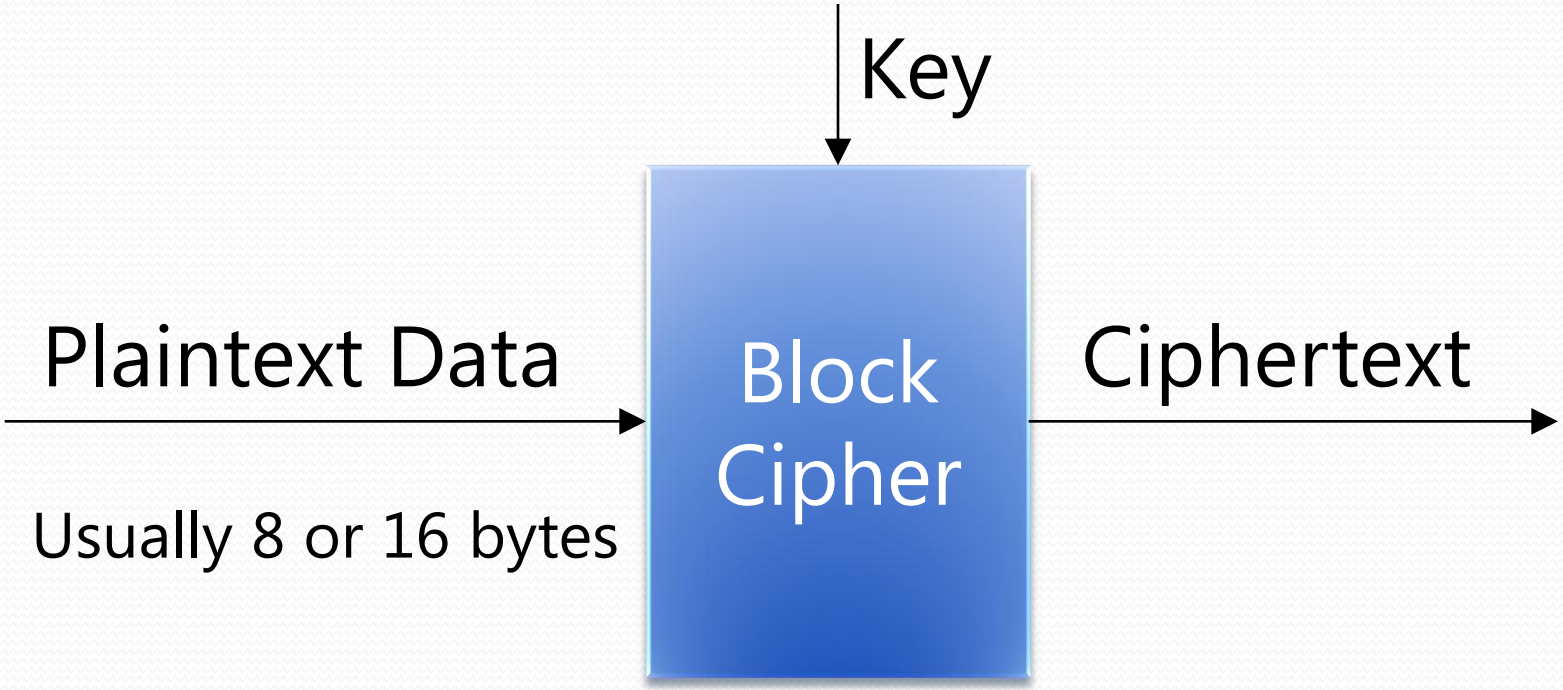
Block Ciphers

- Why are they called “block” ciphers?
  - Because the cipher is defined as a function on a fixed-size block of data.
  - This is called the “block size” and is a fundamental parameter of the cipher.
  - Today 8- or 16-byte blocks are common, but other sizes are possible.

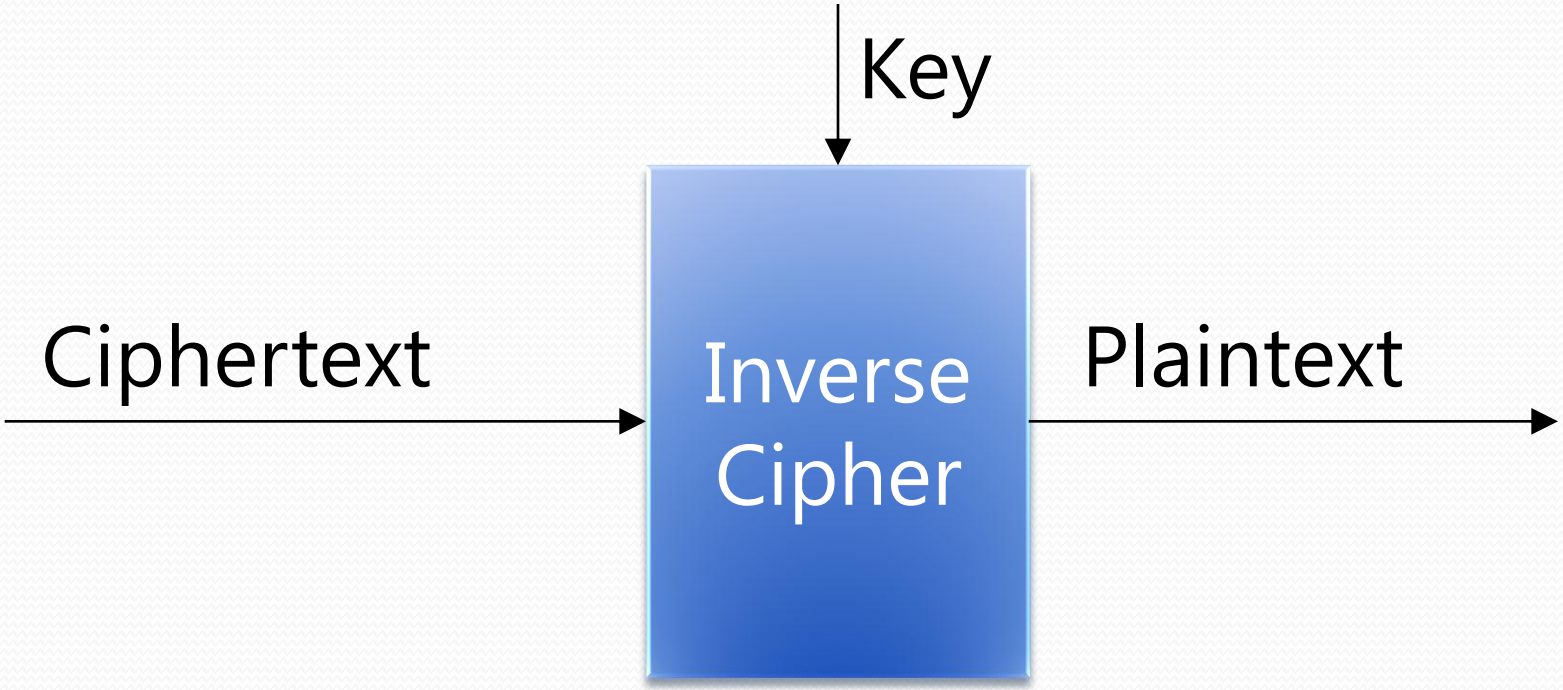
Block Ciphers

- Why are they called “block” ciphers?
  - Because the cipher is defined as a function on a fixed-size block of data.
  - This is called the “block size” and is a fundamental parameter of the cipher.
  - Today 8- or 16-byte blocks are common, but other sizes are possible.
- Question: What’s the “block size” of a stream cipher?

Block Ciphers -- Encryption



Block Ciphers -- Decryption



Block Ciphers

- Q: What do I do if I want to encrypt more that one block of data with a block cipher?

Block Ciphers

- Q: What do I do if I want to encrypt more that one block of data with a block cipher?

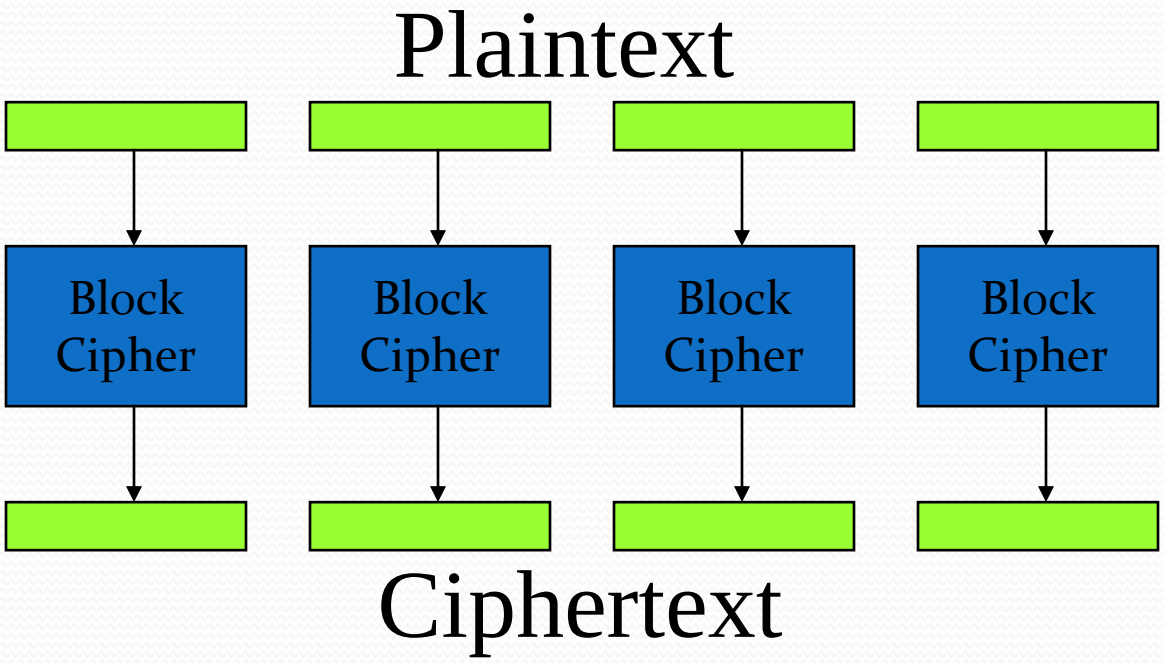
- Simple A: Divide the to-be-encrypted plaintext into block-size chunks, and then apply the cipher to each block.

Block Ciphers

- Q: What do I do if I want to encrypt more that one block of data with a block cipher?
- Simple A: Divide the to-be-encrypted plaintext into block-size chunks, and then apply the cipher to each block.
- Real A: Divide the to-be-encrypted plaintext into block-size chunks, and then apply the cipher to the sequence of blocks using a *mode of operation*.

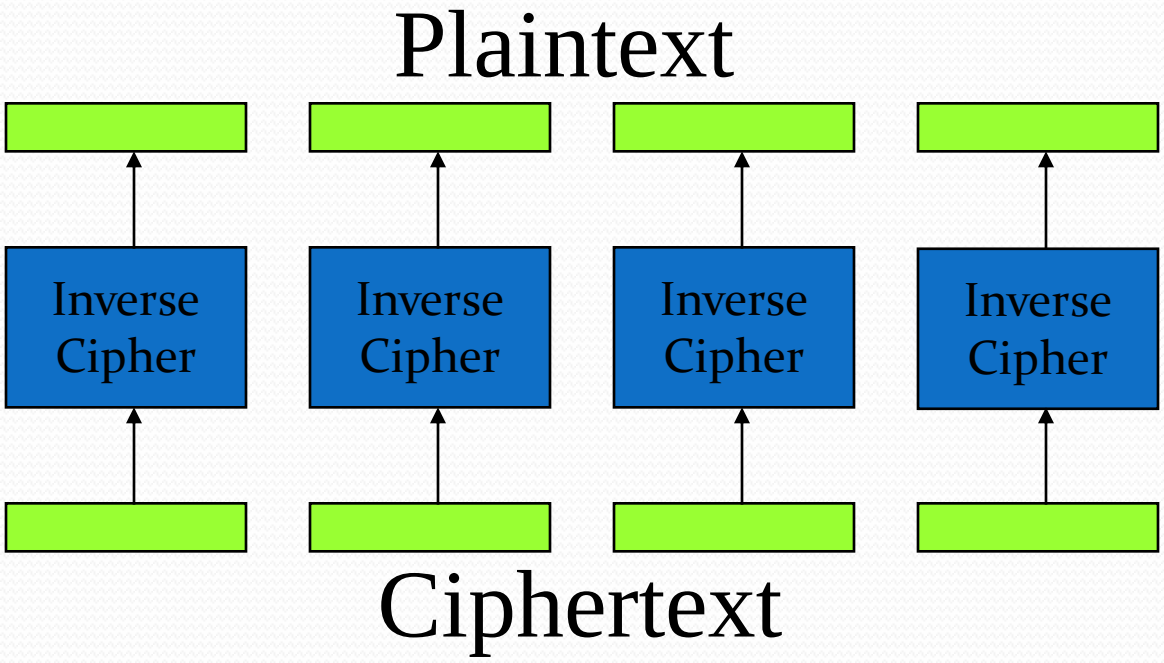
Block Cipher Modes

Electronic Code Book (ECB) Encryption:



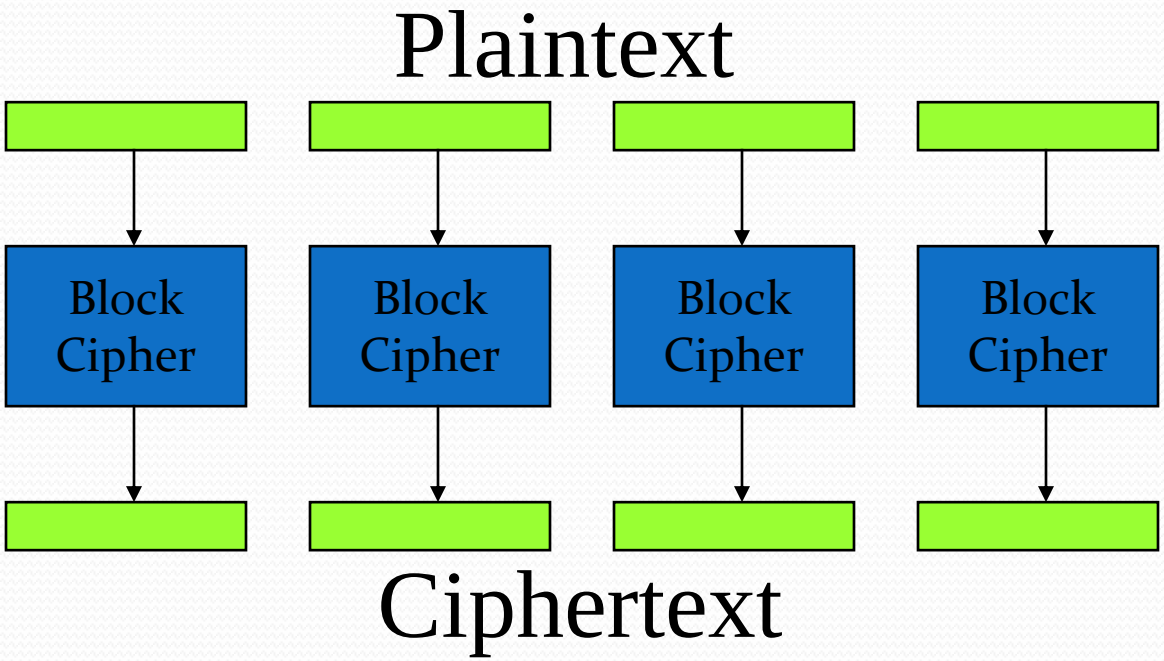
Block Cipher Modes

Electronic Code Book (ECB) Decryption:



Block Cipher Modes

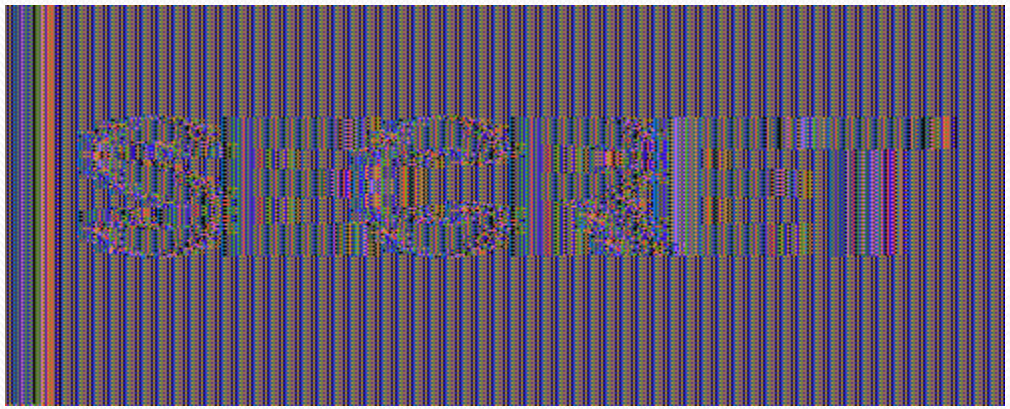
Electronic Code Book (ECB) Encryption:



Why is ECB of Concern?

Why is ECB of Concern?

ECB



Block Cipher Modes

Cipher Block Chaining (CBC) Encryption:

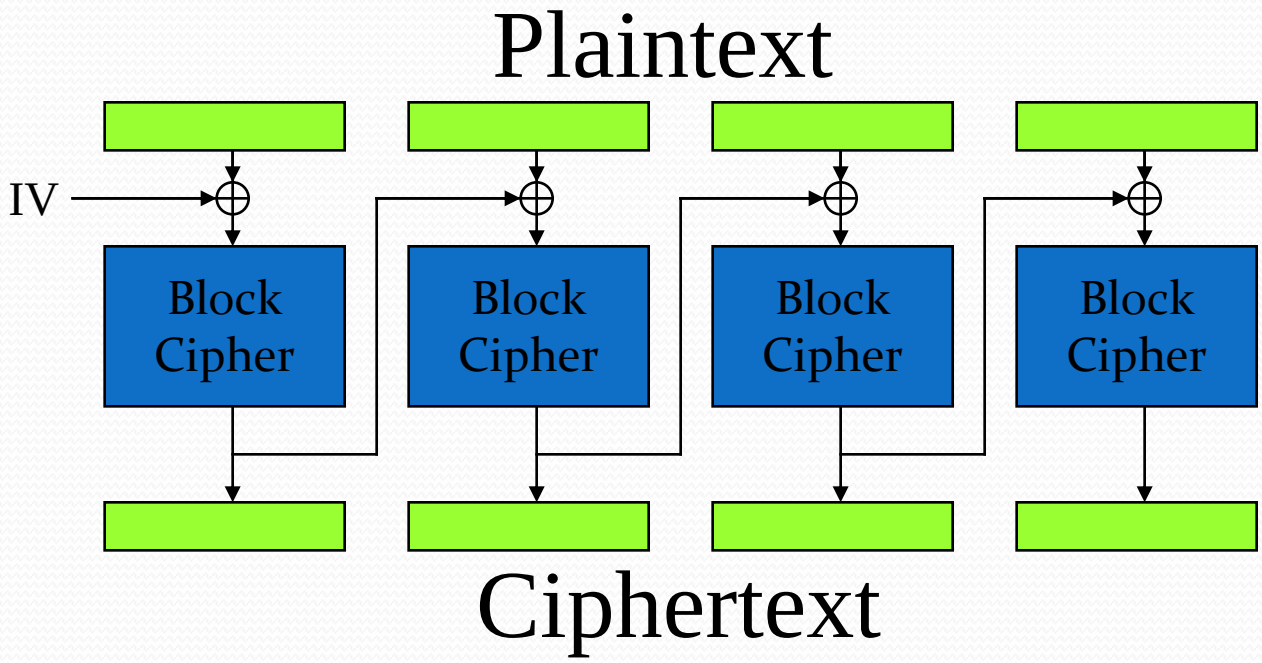
Incorporate an *Initial Value (IV)* which changes with each encryption.

The IV can be

- A counter
- A random value
- Openly known

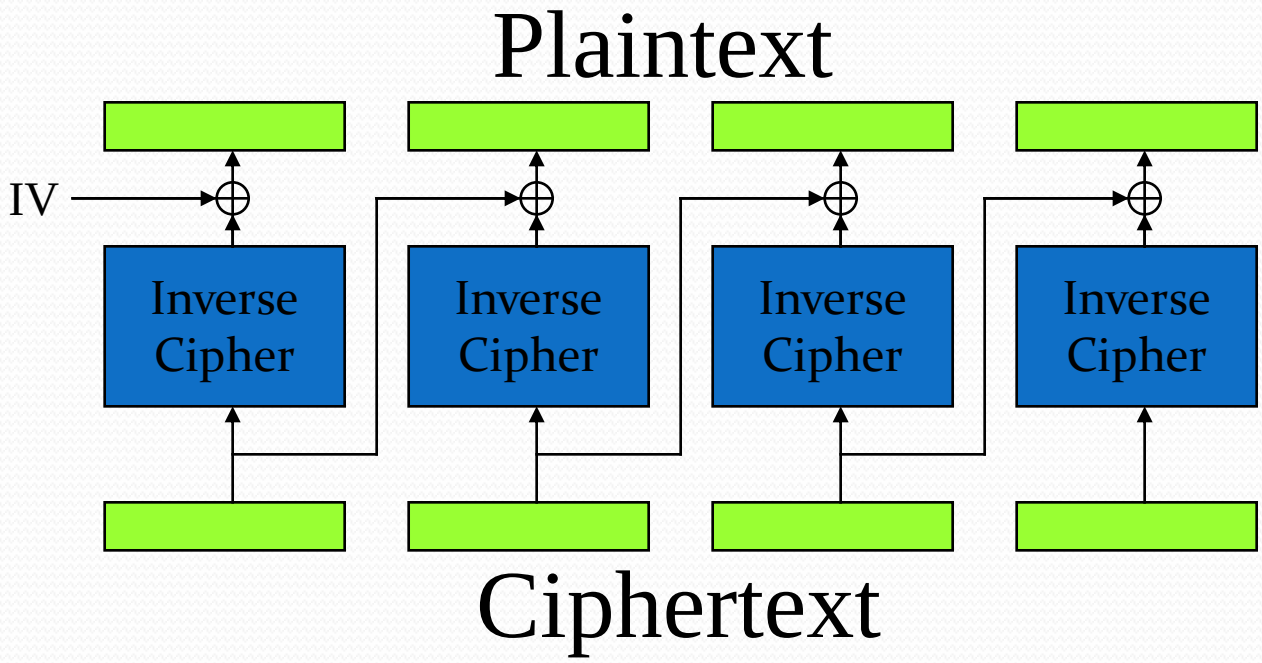
Block Cipher Modes

Cipher Block Chaining (CBC) Encryption:



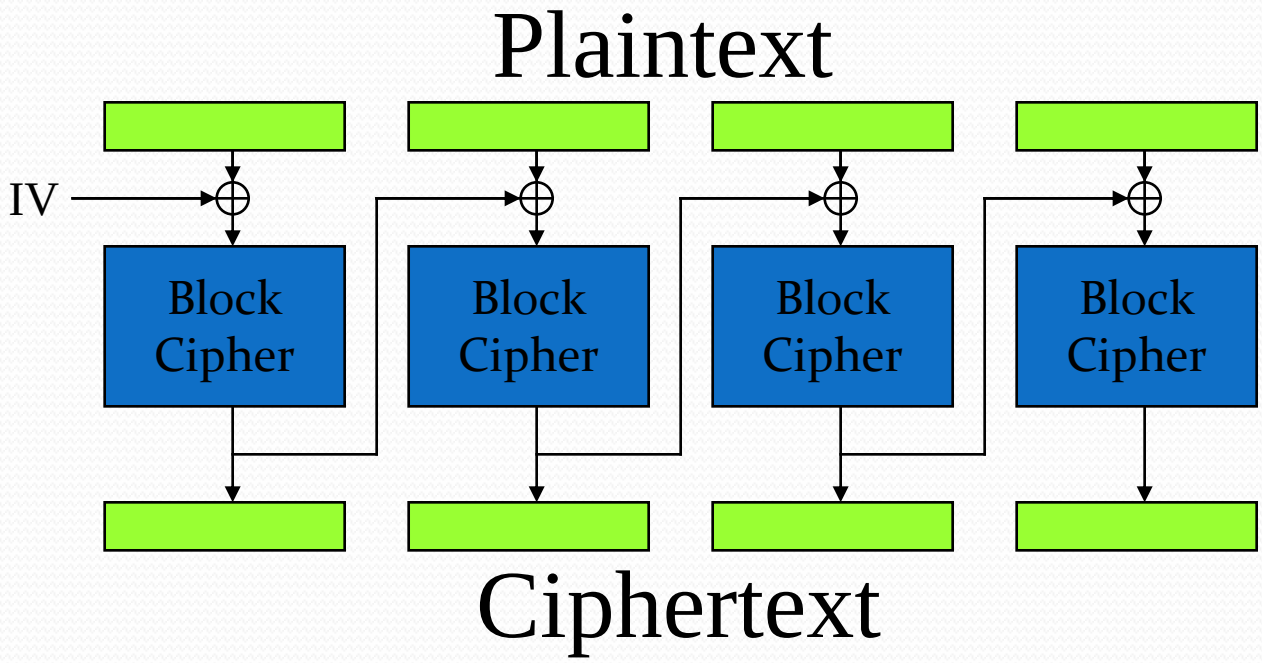
Block Cipher Modes

Cipher Block Chaining (CBC) Decryption:



Block Cipher Modes

Cipher Block Chaining (CBC) Encryption:



Example Block Ciphers

- DES
- 3DES
- AES
- RC2, RC5, TwoFish, Serpent, etc.

Example Block Ciphers

- DES
- 3DES
- AES
- RC2, RC5, TwoFish, Serpent, etc.

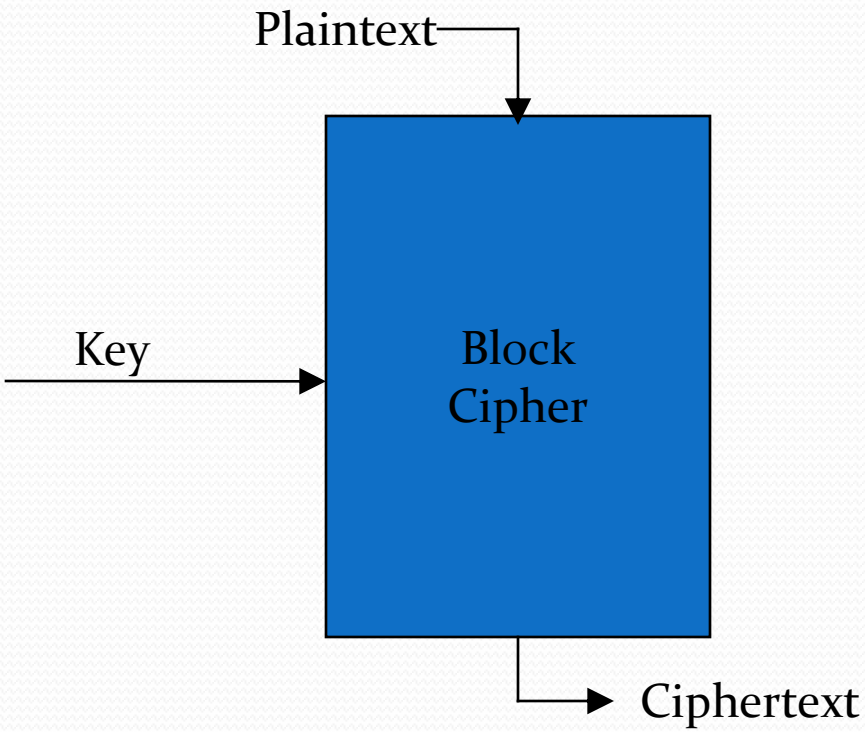
Example Block Ciphers

- DES
- 3DES
- AES
- RC2, RC5, TwoFish, Serpent, etc.

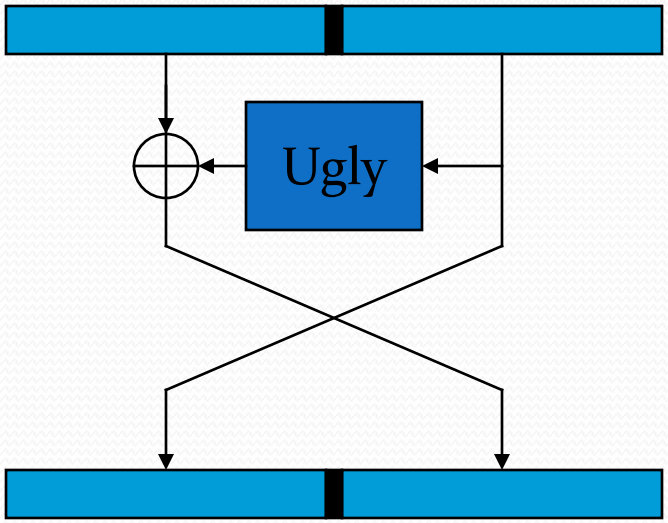
Example Block Ciphers

- DES
- 3DES
- AES
- RC2, RC5, TwoFish, Serpent, etc.

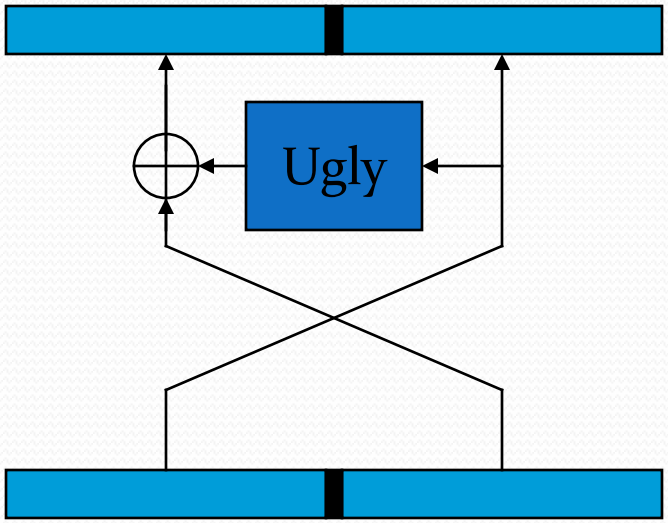
How to Build a Block Cipher



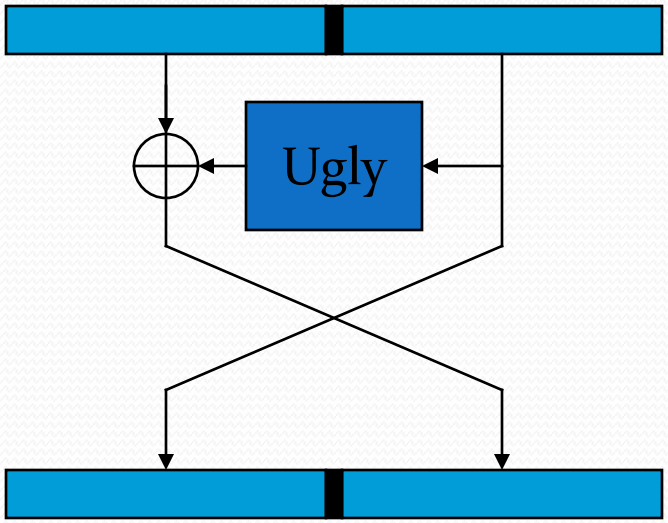
Feistel Ciphers



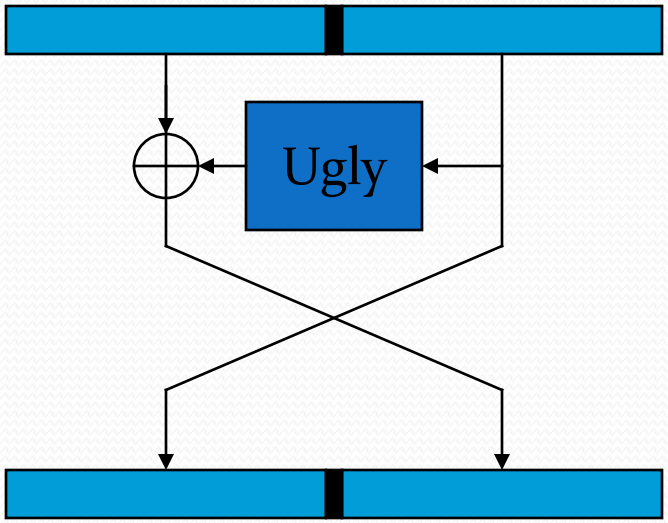
Feistel Ciphers



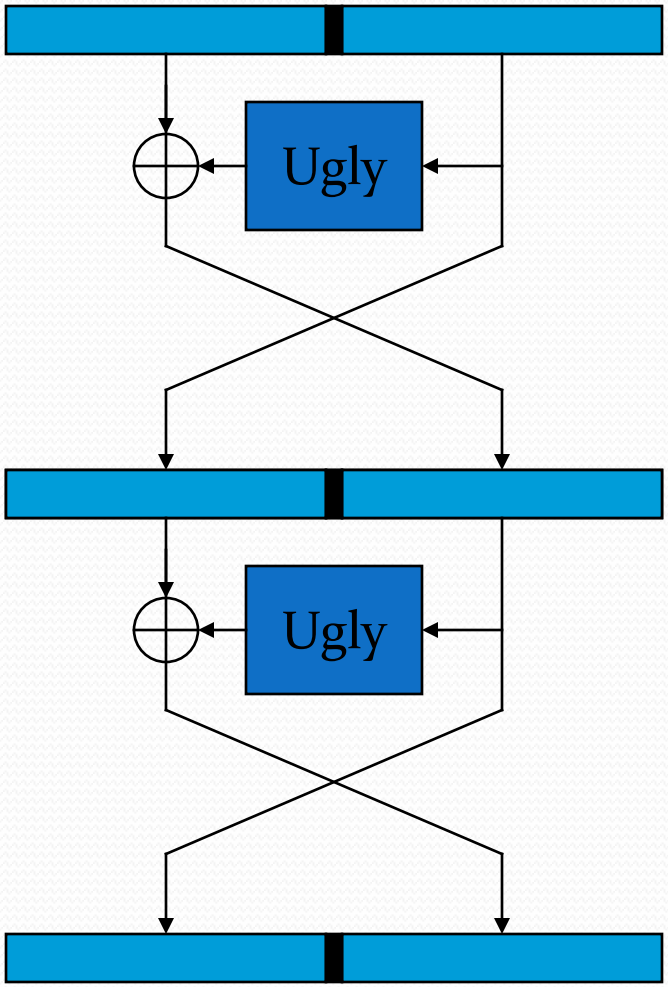
Feistel Ciphers



Feistel Ciphers



Feistel Ciphers

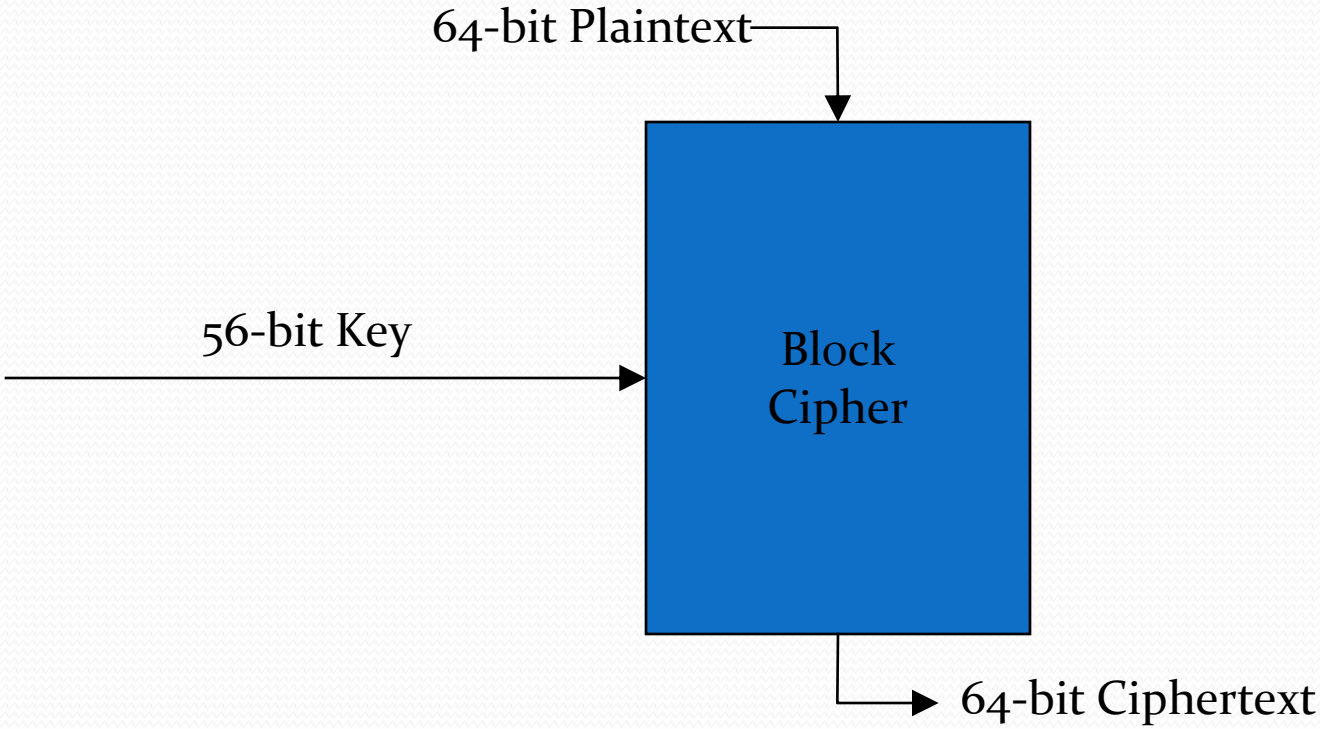


Feistel Ciphers

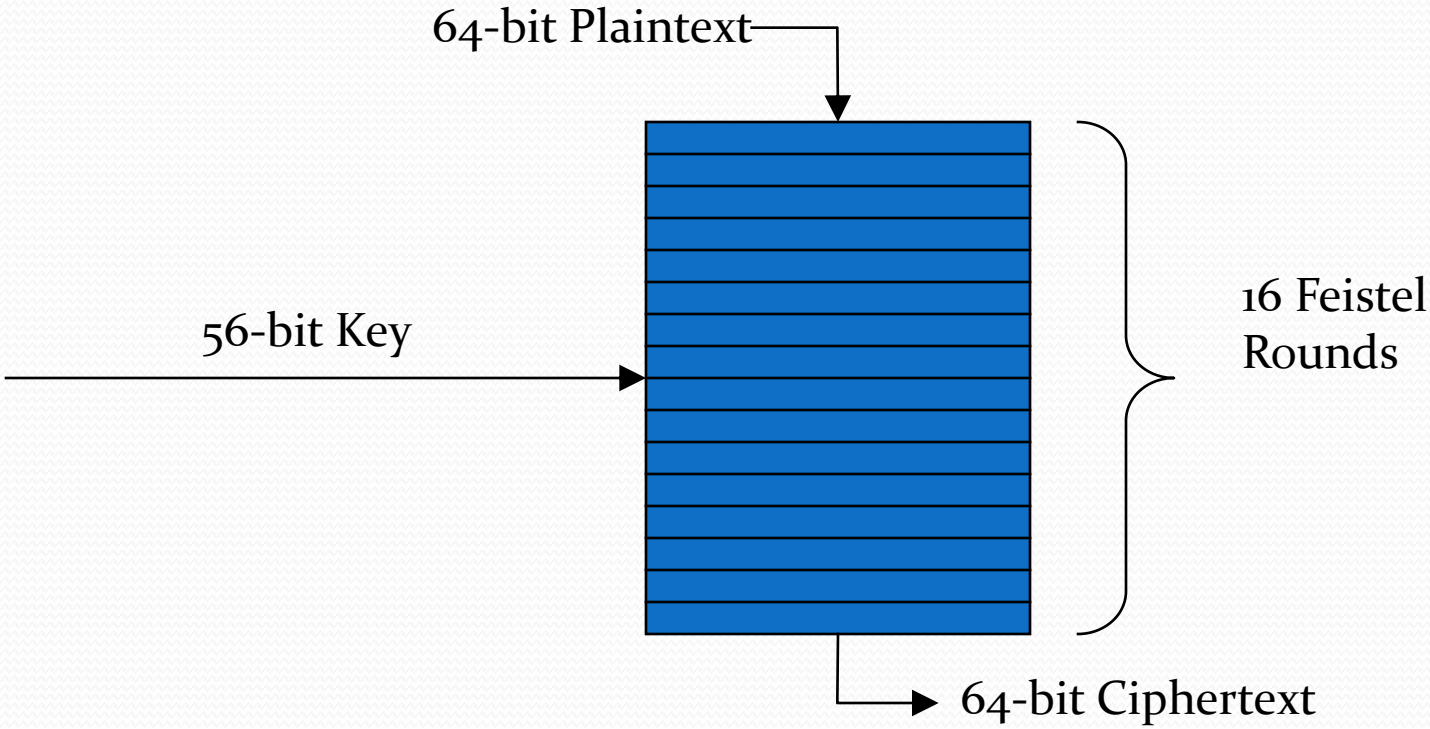
- Typically, most Feistel ciphers are iterated for about 16 rounds.
- Different “sub-keys” are used for each round.

- Even a weak round function can yield a strong Feistel cipher if iterated sufficiently.

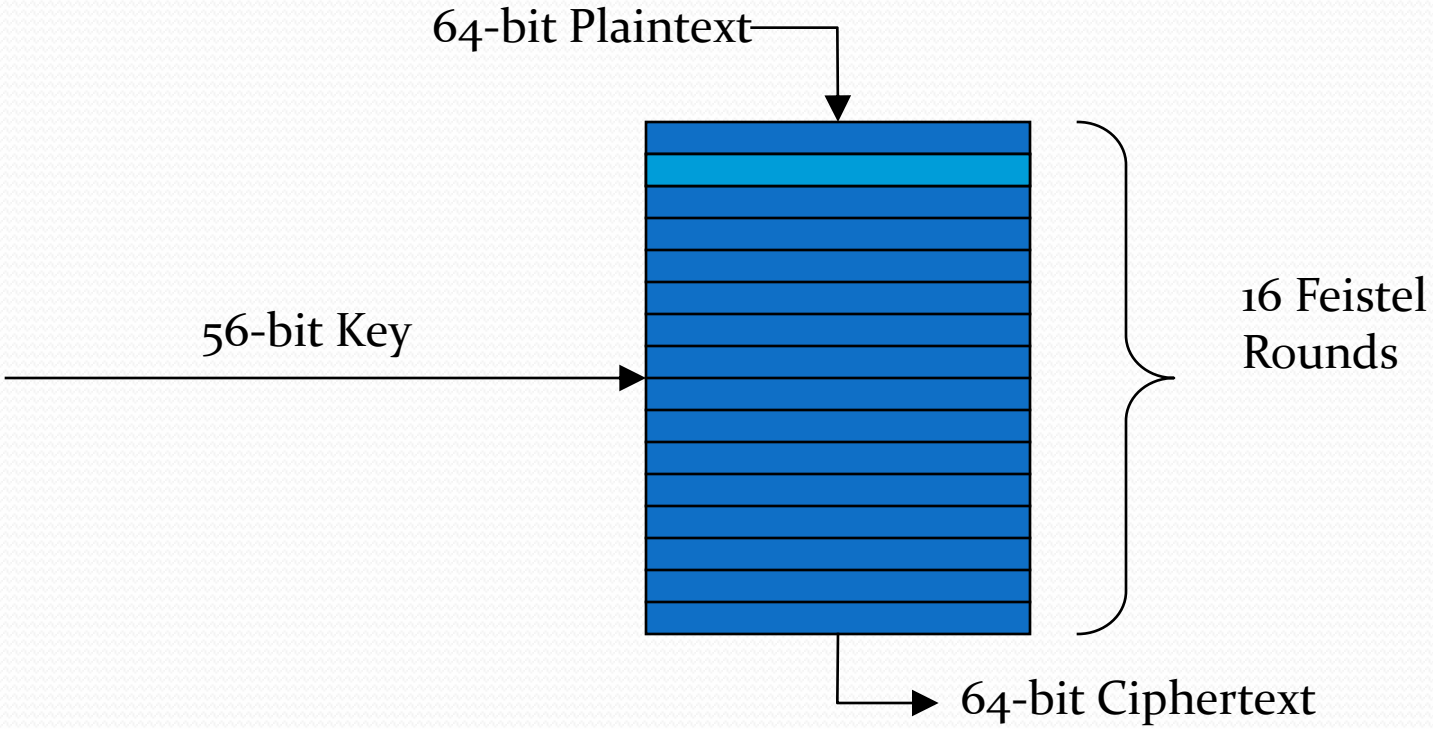
Data Encryption Standard (DES)



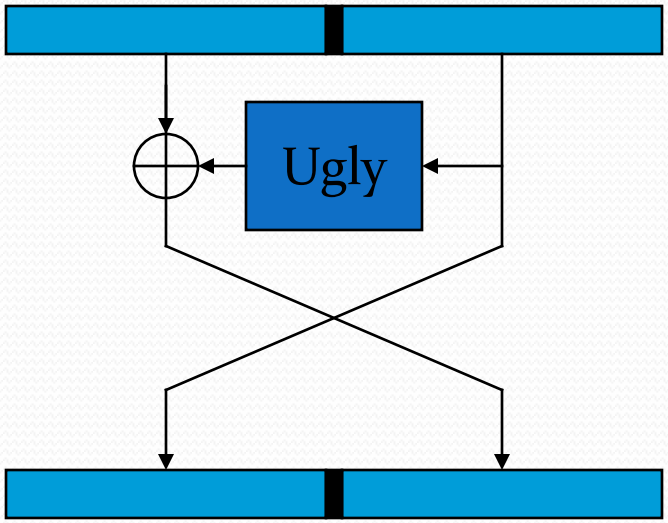
Data Encryption Standard (DES)



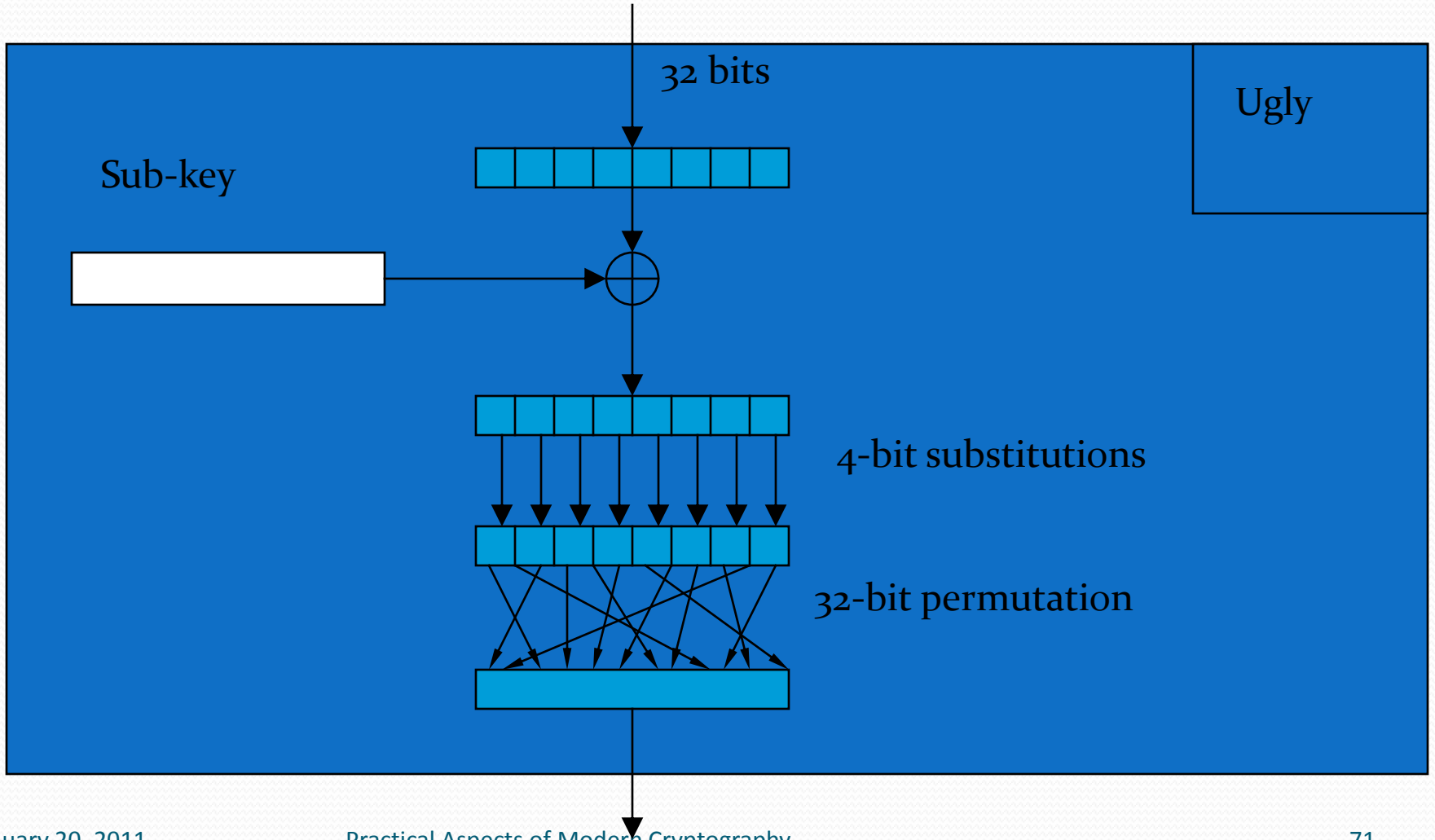
Data Encryption Standard (DES)



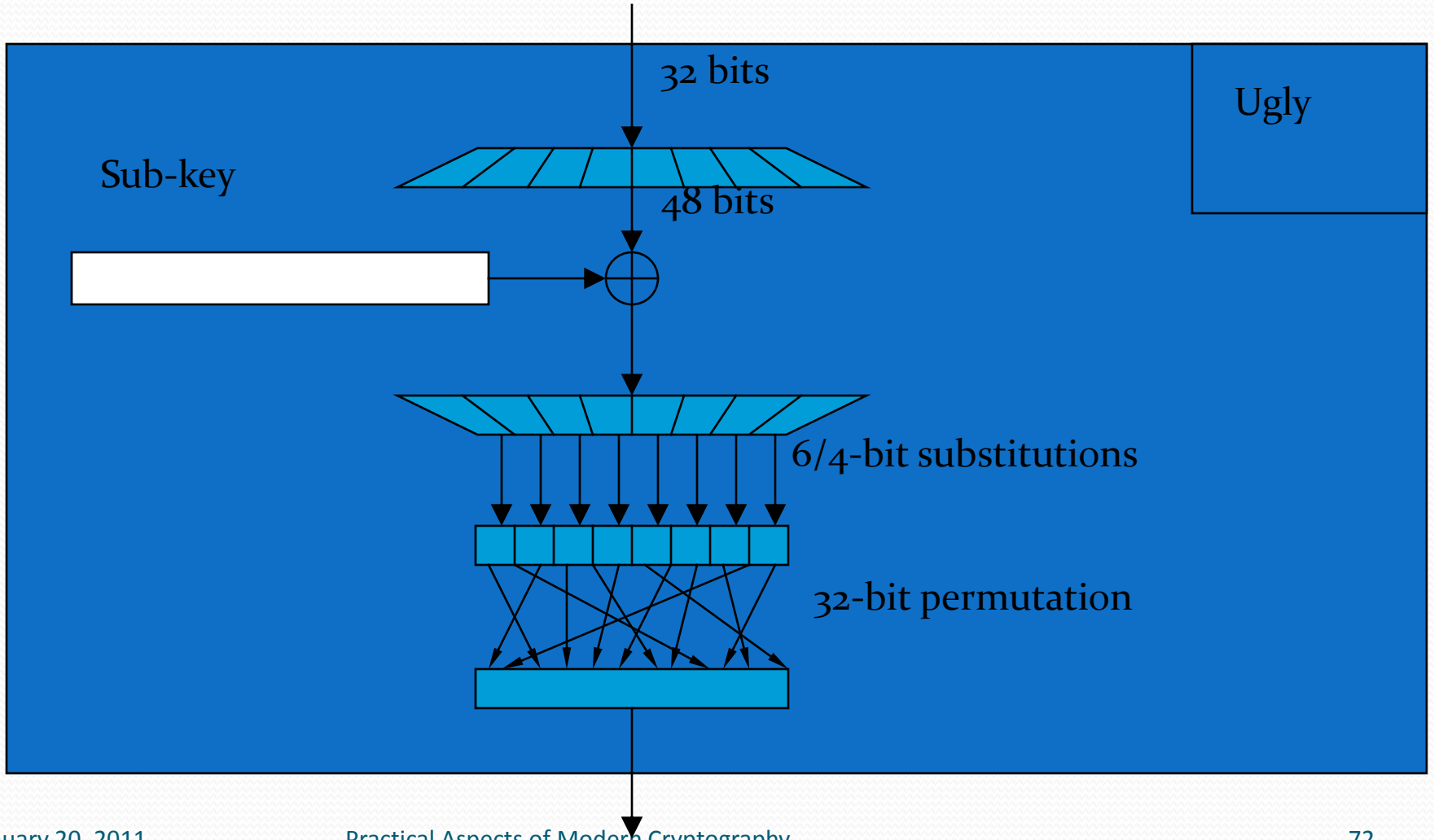
DES Round



Simplified DES Round Function



Actual DES Round Function



Advanced Encryption Standard

- Open competition run by NIST to replace DES
- 128-bit block size
- Key sizes of 128, 192, and 256 bits

- 15 ciphers were submitted
- 5 finalists were chosen

AES Finalists

- MARS (IBM submission)
- RC6 (RSA Labs submission)
- Rijndael (Joan Daemen and Vincent Rijmen)
- Serpent (Anderson, Biham, and Knudsen)
- Twofish (Schneier, et. al.)

AES Finalists

- MARS (IBM submission)
- RC6 (RSA Labs submission)
- **Rijndael (Joan Daemen and Vincent Rijmen)**
- Serpent (Anderson, Biham, and Knudsen)
- Twofish (Schneier, et. al.)

Rijndael

$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$	$k_{0,4}$	$k_{0,5}$	$k_{0,6}$	$k_{0,7}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$	$k_{1,4}$	$k_{1,5}$	$k_{1,6}$	$k_{1,7}$
$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$	$k_{2,4}$	$k_{2,5}$	$k_{2,6}$	$k_{2,7}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$	$k_{3,4}$	$k_{3,5}$	$k_{3,6}$	$k_{3,7}$

16, 24, or 32 bytes of key

16, 24, or 32 bytes of data

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$	$a_{0,6}$	$a_{0,7}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$	$a_{1,6}$	$a_{1,7}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$	$a_{2,6}$	$a_{2,7}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{3,5}$	$a_{3,6}$	$a_{3,7}$

Rijndael

4 transformations per round

- ByteSub: nonlinearity
- ShiftRow: inter-column diffusion
- MixColumn: inter-byte diffusion
- Round key addition

Rijndael ByteSub

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

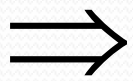
$\Rightarrow$

$b_{0,0}$	$b_{0,1}$	$b_{0,2}$	$b_{0,3}$
$b_{1,0}$	$b_{1,1}$	$b_{1,2}$	$b_{1,3}$
$b_{2,0}$	$b_{2,1}$	$b_{2,2}$	$b_{2,3}$
$b_{3,0}$	$b_{3,1}$	$b_{3,2}$	$b_{3,3}$

A single 8-bit to 8-bit (invertible) S-box is applied to each byte.

Rijndael MixColumn

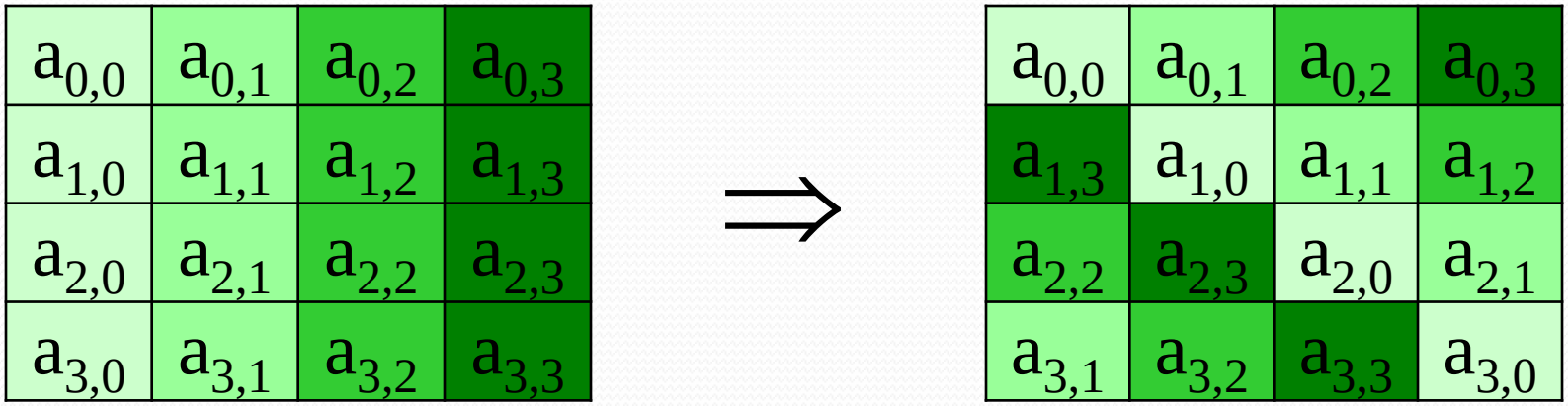
$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$



$b_{0,0}$	$b_{0,1}$	$b_{0,2}$	$b_{0,3}$
$b_{1,0}$	$b_{1,1}$	$b_{1,2}$	$b_{1,3}$
$b_{2,0}$	$b_{2,1}$	$b_{2,2}$	$b_{2,3}$
$b_{3,0}$	$b_{3,1}$	$b_{3,2}$	$b_{3,3}$

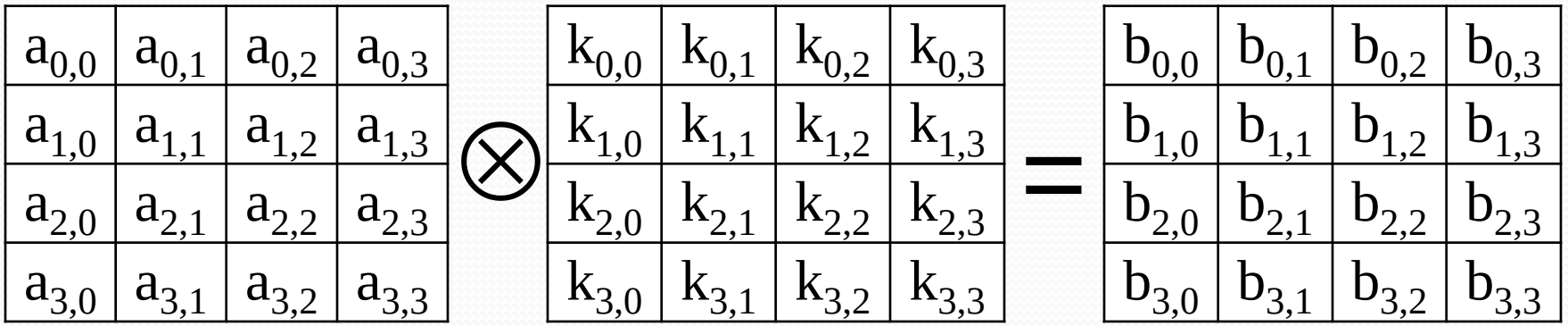
An (invertible) linear transform is applied to each column.

Rijndael ShiftRow



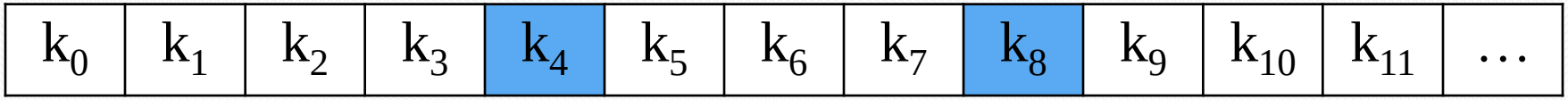
A different cyclic shift is applied to each row.

Rijndael Round key addition



The round key is XORed to complete the round.

Rijndael Key Schedule



The key schedule is defined on 4-byte words by

- $k_i = k_{i-4} \otimes k_{i-1}$ when i is not a multiple of 4
- $k_i = k_{i-4} \otimes f(k_{i-1})$ when i is a multiple of 4

Agenda

- Symmetric key ciphers
  - Stream ciphers
  - Block ciphers
- Cryptographic hash functions

One-Way Hash Functions

Generally, a *one-way hash function* is a function $H : \{0,1\}^* \rightarrow \{0,1\}^k$ such that given an input value x, one cannot find a value $x' \neq x$ such $H(x) = H(x')$.

One-Way Hash Functions

Generally, a *one-way hash function* is a function $H : \{0,1\}^* \rightarrow \{0,1\}^k$ such that given an input value x, one cannot find a value $x' \neq x$ such $H(x) = H(x')$.

- Typically k is 128, 160, 256, 384, or 512

One-Way Hash Functions

There are many properties of one-way hashes.

- Non-invertability: given y, it's difficult to find any x such that $H(x) = y$.

One-Way Hash Functions

There are many properties of one-way hashes.

- Non-invertability: given y, it's difficult to find any x such that $H(x) = y$.
- Second-preimage resistance: given x, it's difficult to find $x' \neq x$ such that $H(x) = H(x')$.

One-Way Hash Functions

There are many properties of one-way hashes.

- Non-invertability: given y, it's difficult to find any x such that $H(x) = y$.
- Second-preimage resistance: given x, it's difficult to find $x' \neq x$ such that $H(x) = H(x')$.
- Collision-intractability: one cannot find a pair of values $x' \neq x$ such that $H(x) = H(x')$.

Some Important Uses of One-Way Hash Functions

- When using a stream cipher, a hash of the message can be appended to ensure integrity. [Message Authentication Code]

Some Important Uses of One-Way Hash Functions

- When using a stream cipher, a hash of the message can be appended to ensure integrity. [Message Authentication Code]

- When forming a digital signature, the signature need only be applied to a hash of the message. [Message Digest]

What else are Hash Functions Good for?

Hash functions are useful in lots of situations; here are some additional examples

What else are Hash Functions Good for?

Hash functions are useful in lots of situations; here are some additional examples

- Uniquely and securely identify bit streams like programs. Hash is strong name for program.

What else are Hash Functions Good for?

Hash functions are useful in lots of situations; here are some additional examples

- Uniquely and securely identify bit streams like programs. Hash is strong name for program.
- Entropy mixing: Since cryptographic hashes are random functions into fixed size blocks with the properties of random functions, they are often used to "mix" biased input to produce a "seed" for a pseudo-random number generator.

What else are Hash Functions Good for?

Hash functions are useful in lots of situations; here are some additional examples

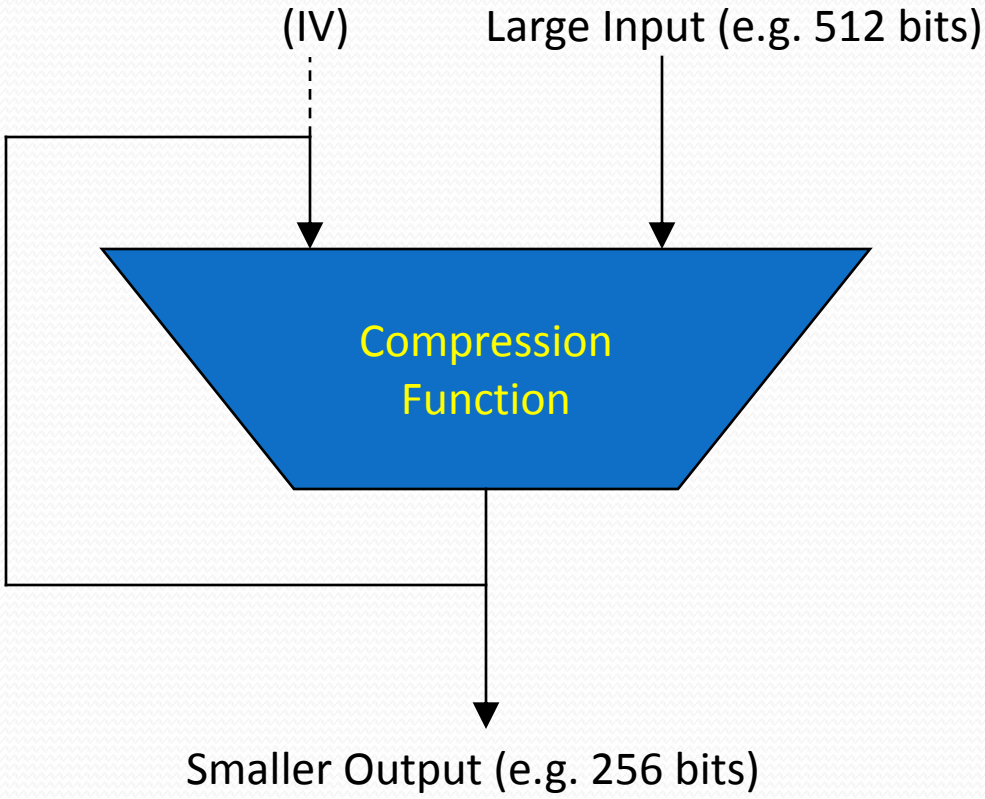
- Uniquely and securely identify bit streams like programs. Hash is strong name for program.
- Entropy mixing: Since cryptographic hashes are random functions into fixed size blocks with the properties of random functions, they are often used to "mix" biased input to produce a "seed" for a pseudo-random number generator.
- Password Protection: Store salted hash of password instead of password (Needham).

What else are Hash Functions Good for?

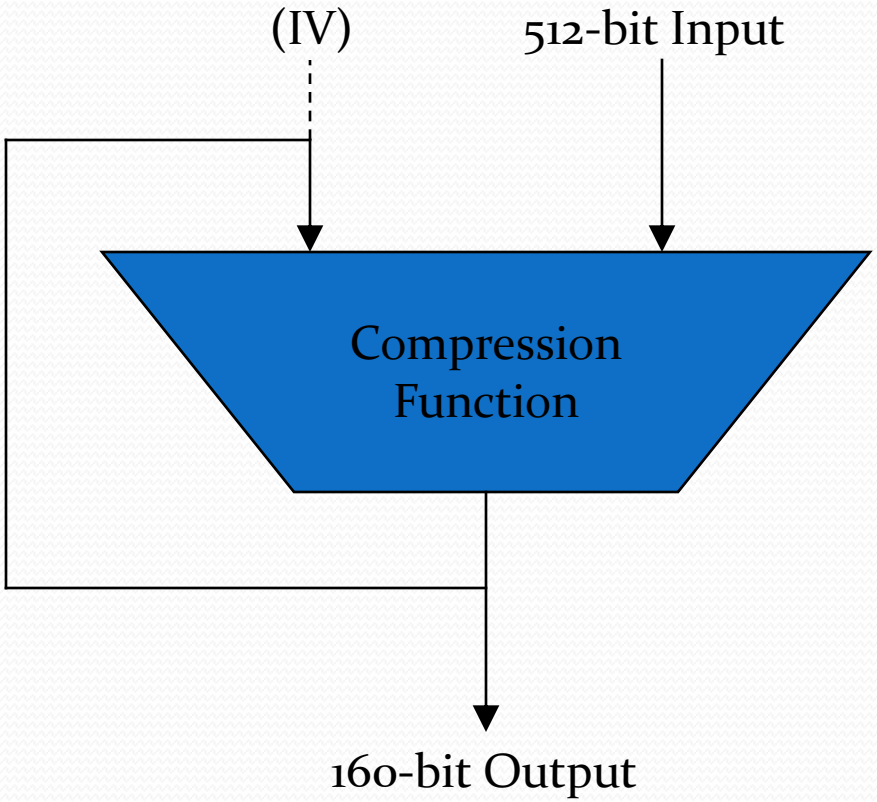
Hash functions are useful in lots of situations; here are some additional examples

- Uniquely and securely identify bit streams like programs. Hash is strong name for program.
- Entropy mixing: Since cryptographic hashes are random functions into fixed size blocks with the properties of random functions, they are often used to "mix" biased input to produce a "seed" for a pseudo-random number generator.
- Password Protection: Store salted hash of password instead of password (Needham).
- Bit Commitment

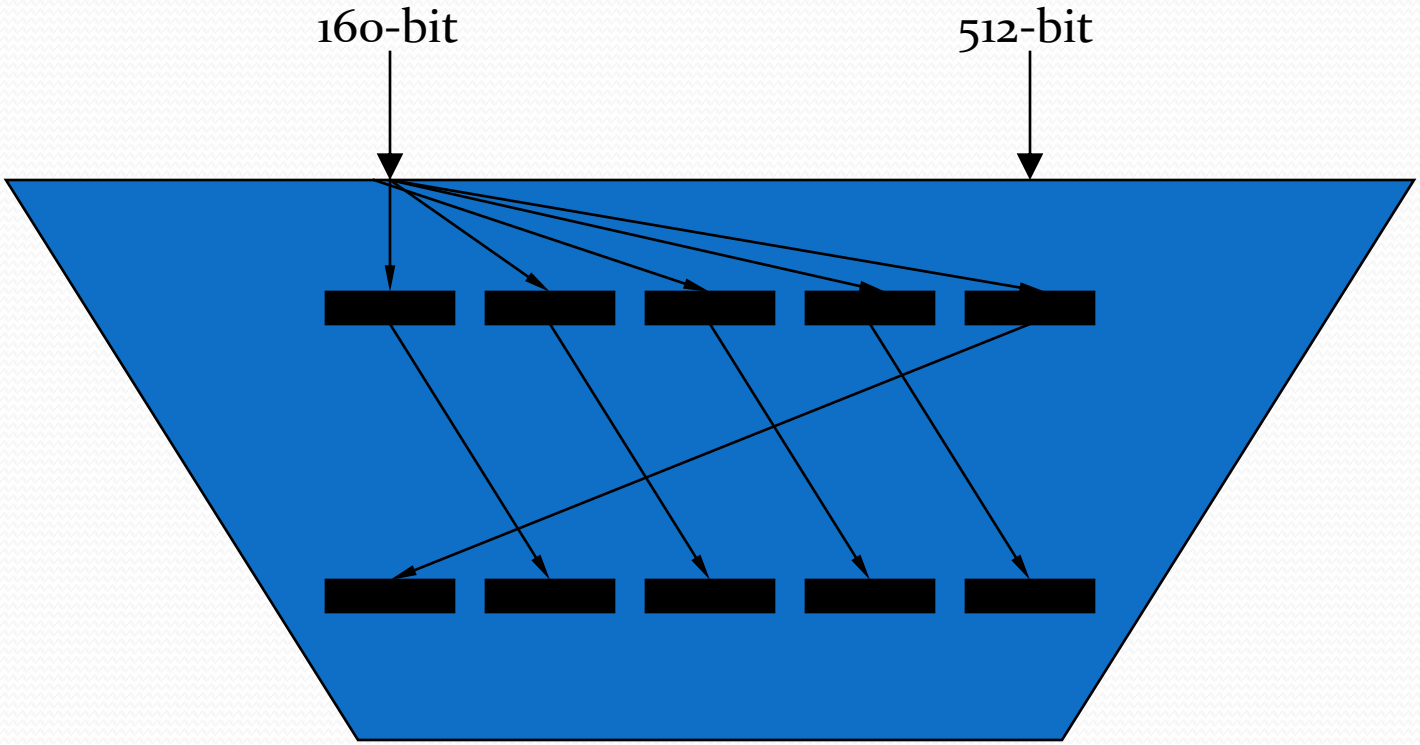
Merkle-Damgård Construction



A Cryptographic Hash: SHA-1

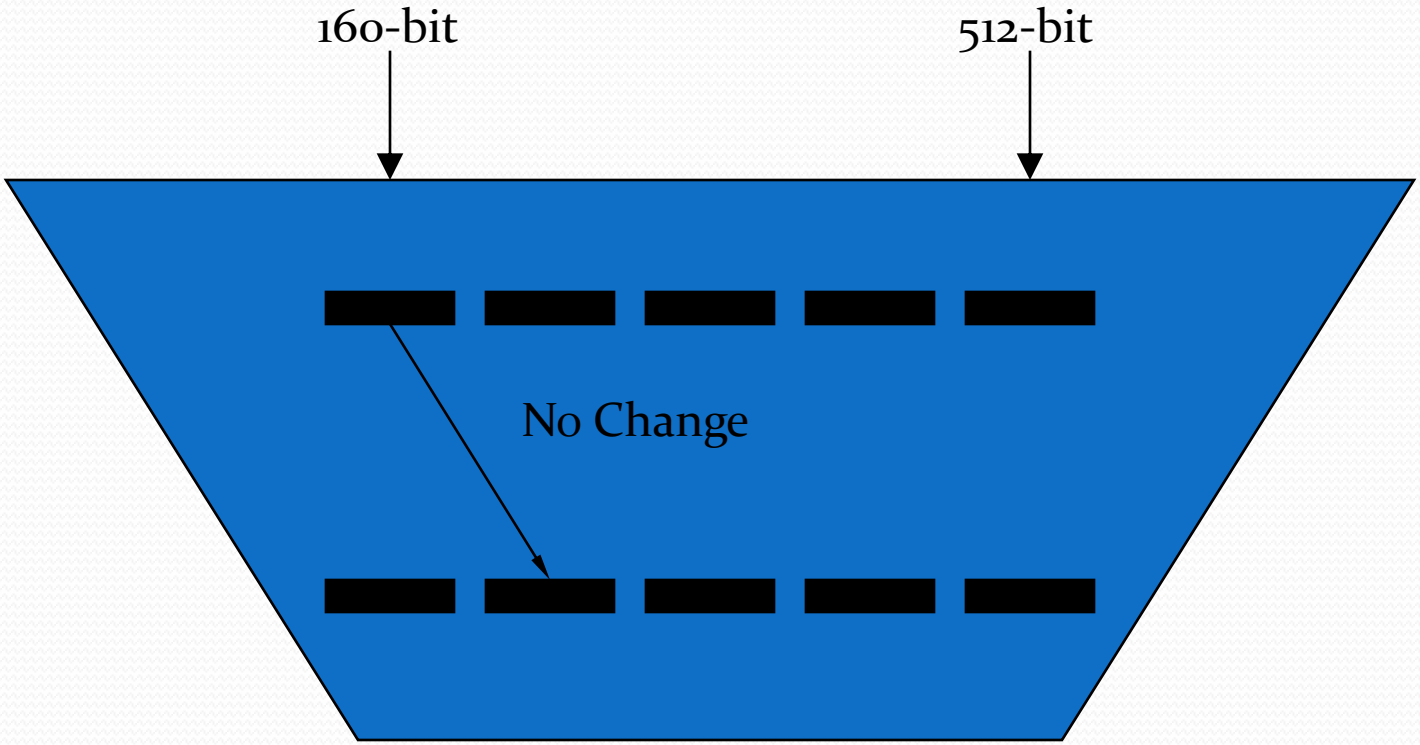


A Cryptographic Hash: SHA-1



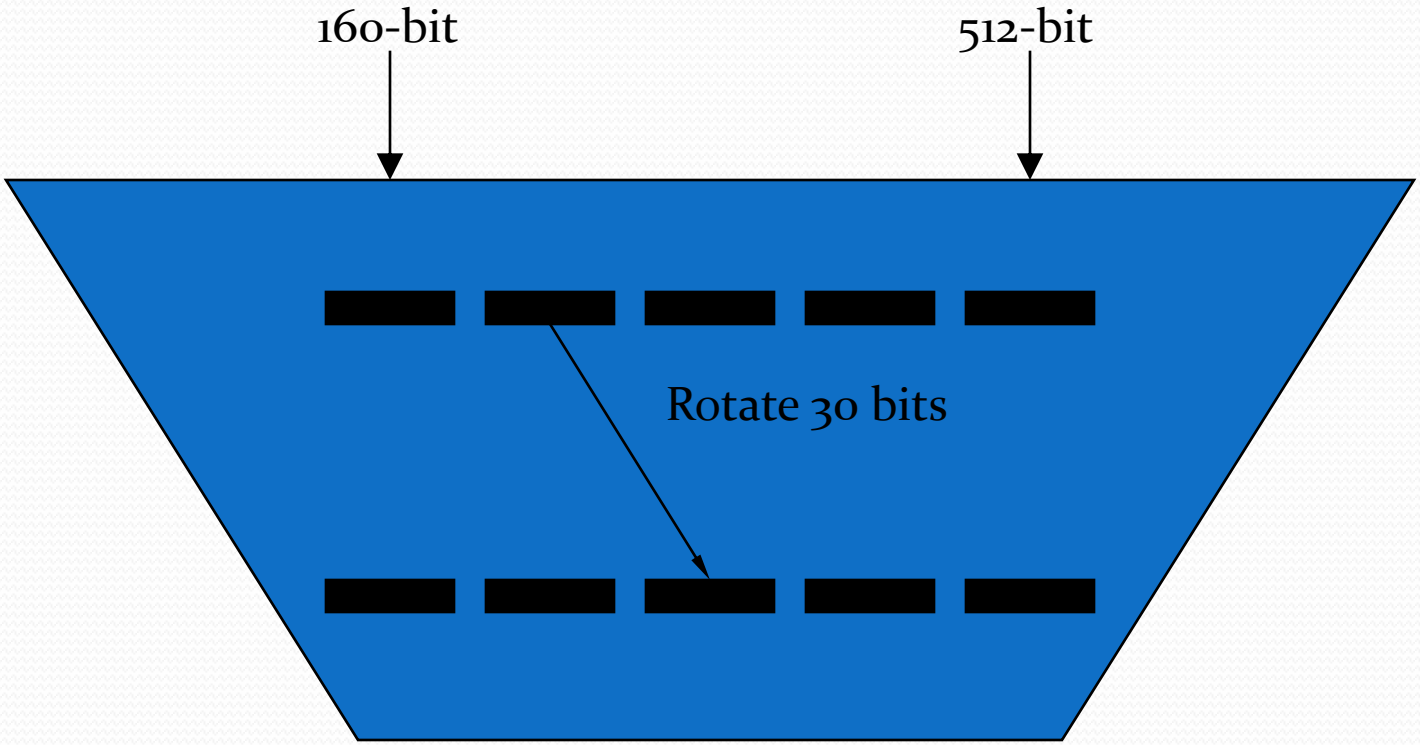
One of 80 rounds

A Cryptographic Hash: SHA-1



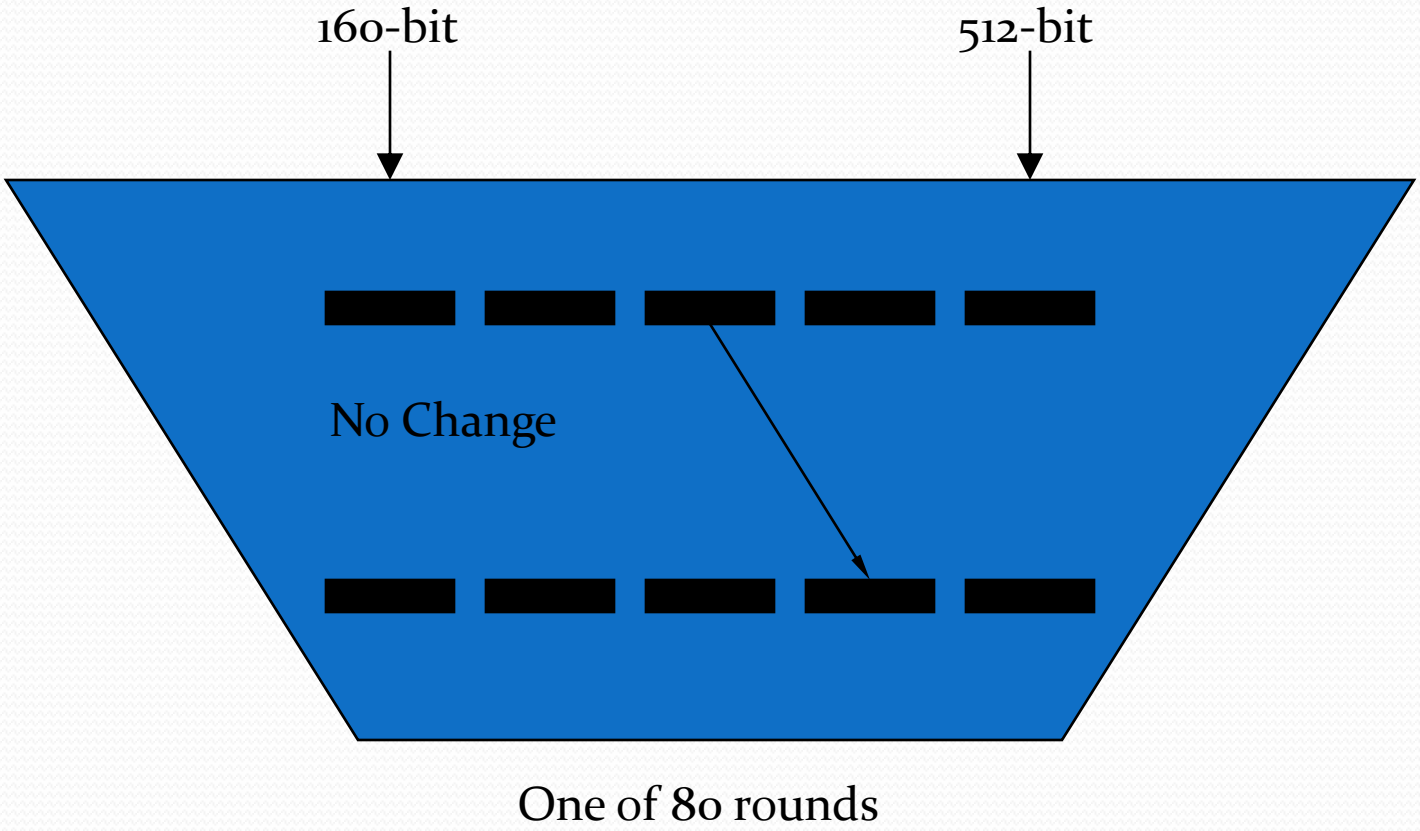
One of 80 rounds

A Cryptographic Hash: SHA-1

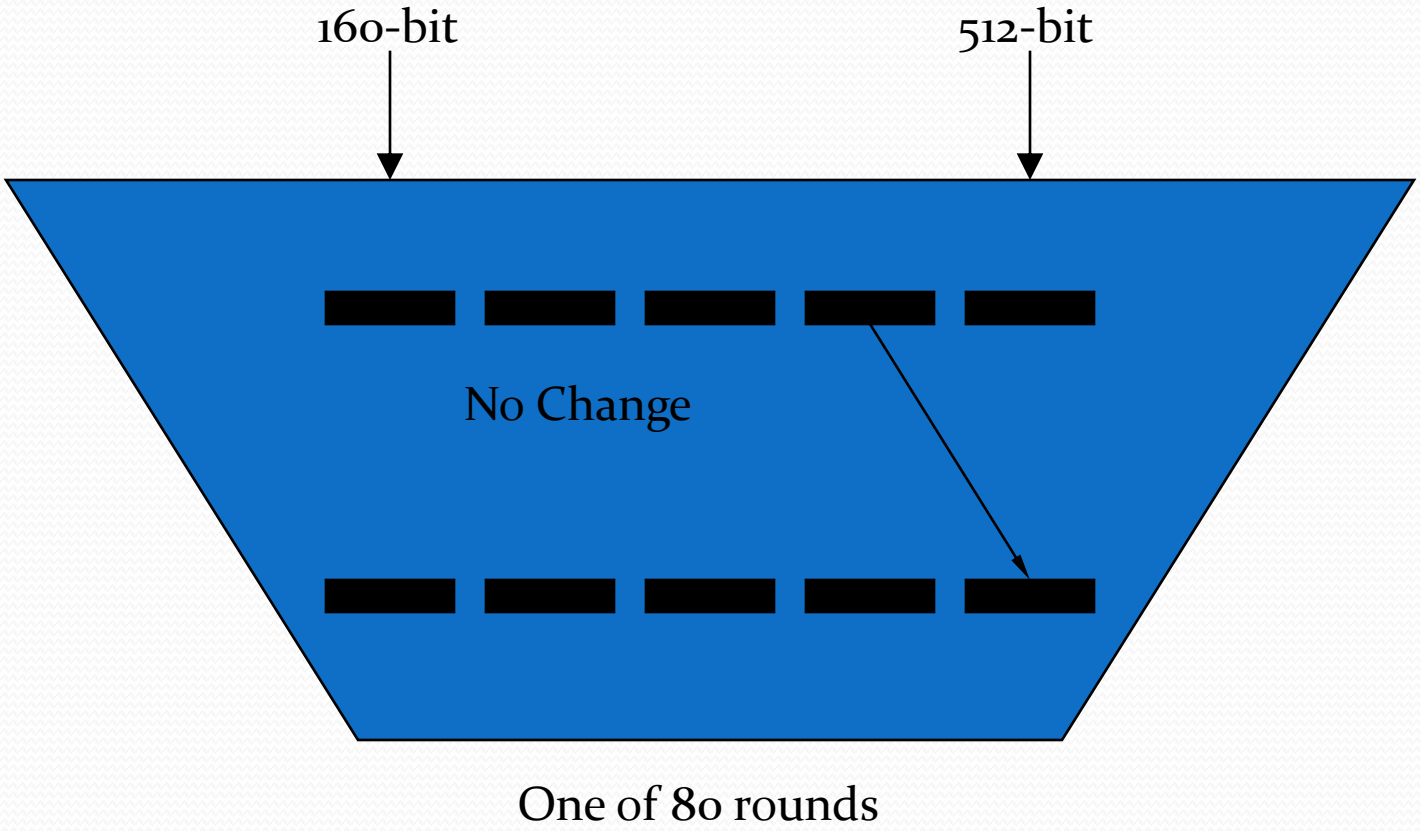


One of 80 rounds

A Cryptographic Hash: SHA-1

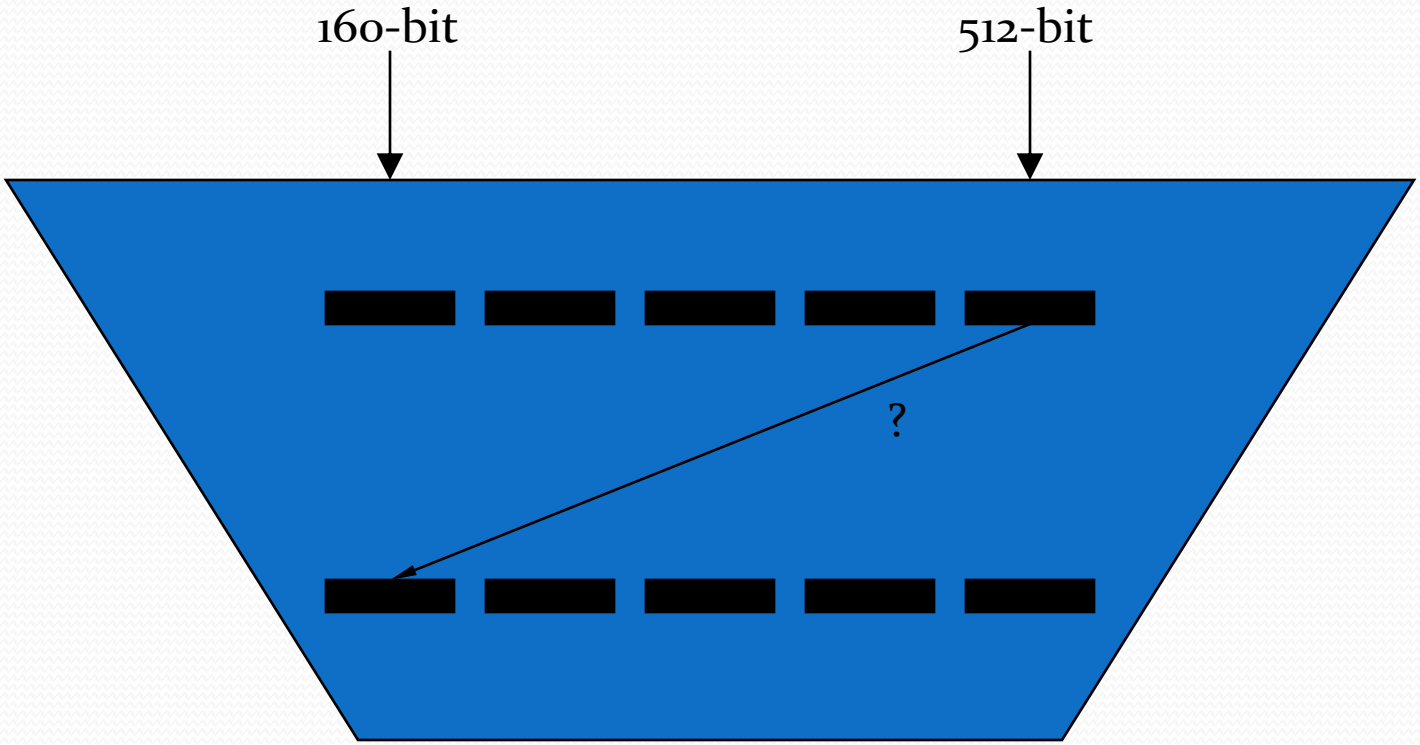


A Cryptographic Hash: SHA-1



One of 80 rounds

A Cryptographic Hash: SHA-1



One of 80 rounds

A Cryptographic Hash: SHA-1

What's in the final 32-bit transform?

- Take the rightmost word.

A Cryptographic Hash: SHA-1

What's in the final 32-bit transform?

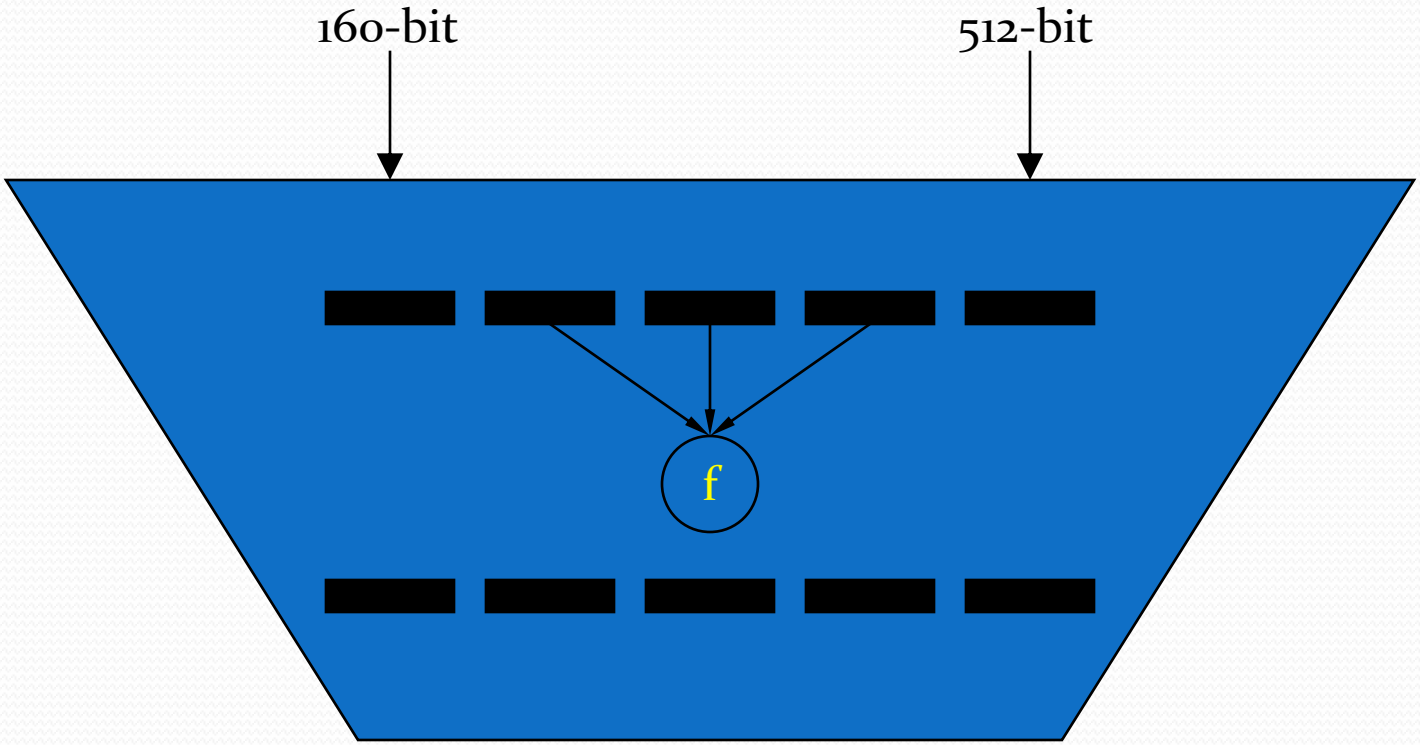
- Take the rightmost word.
- Add in the leftmost word rotated 5 bits.

A Cryptographic Hash: SHA-1

What's in the final 32-bit transform?

- Take the rightmost word.
- Add in the leftmost word rotated 5 bits.
- Add in a round-dependent function f of the middle three words.

A Cryptographic Hash: SHA-1



One of 80 rounds

A Cryptographic Hash: SHA-1

Depending on the round, the "non-linear" function f is one of the following.

$$f(X,Y,Z) = (X \wedge Y) \vee ((\neg X) \wedge Z)$$

$$f(X,Y,Z) = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$$

$$f(X,Y,Z) = X \oplus Y \oplus Z$$

A Cryptographic Hash: SHA-1

What's in the final 32-bit transform?

- Take the rightmost word.
- Add in the leftmost word rotated 5 bits.
- Add in a round-dependent function f of the middle three words.

A Cryptographic Hash: SHA-1

What's in the final 32-bit transform?

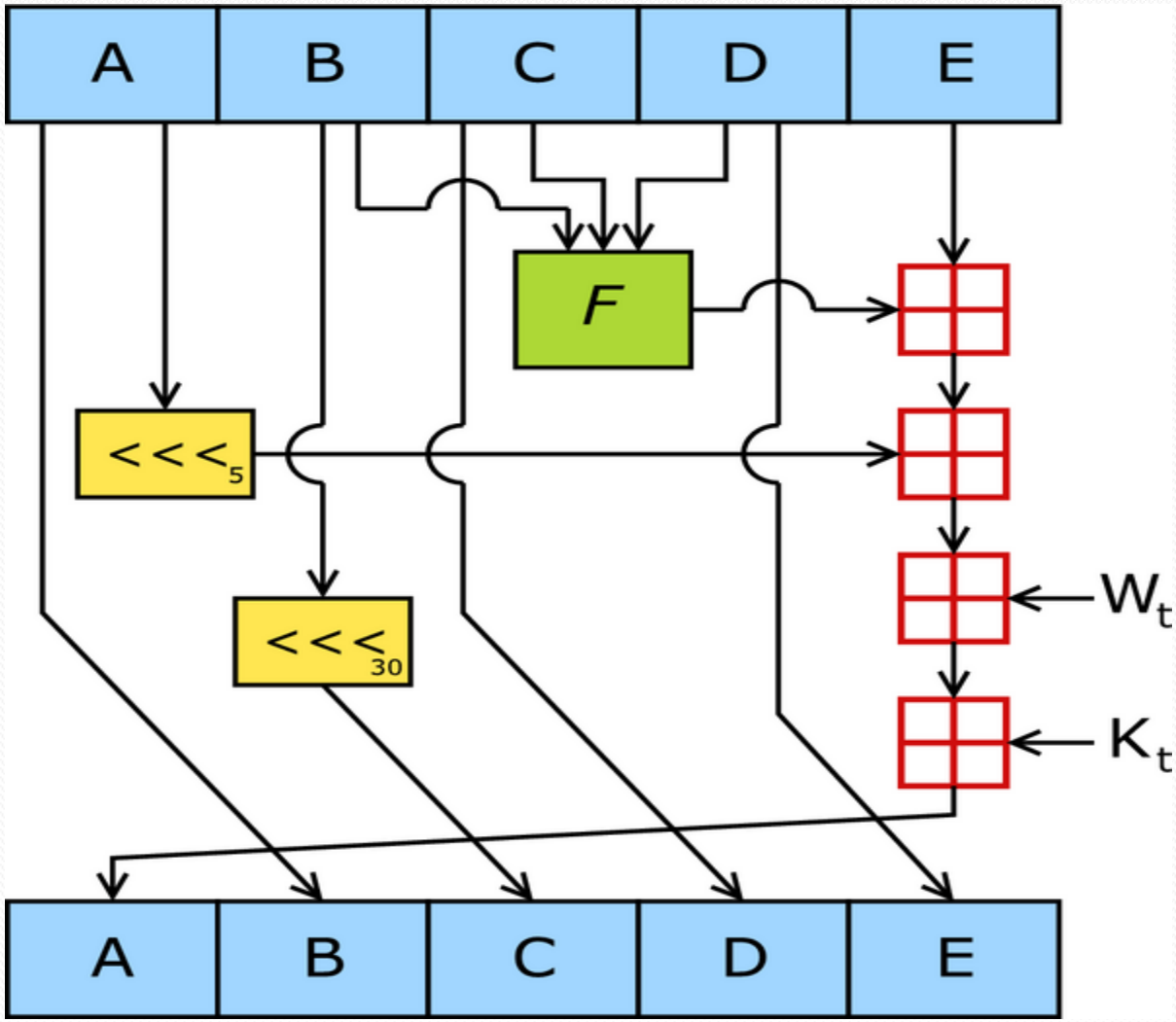
- Take the rightmost word.
- Add in the leftmost word rotated 5 bits.
- Add in a round-dependent function f of the middle three words.
- Add in a round-dependent constant.

A Cryptographic Hash: SHA-1

What's in the final 32-bit transform?

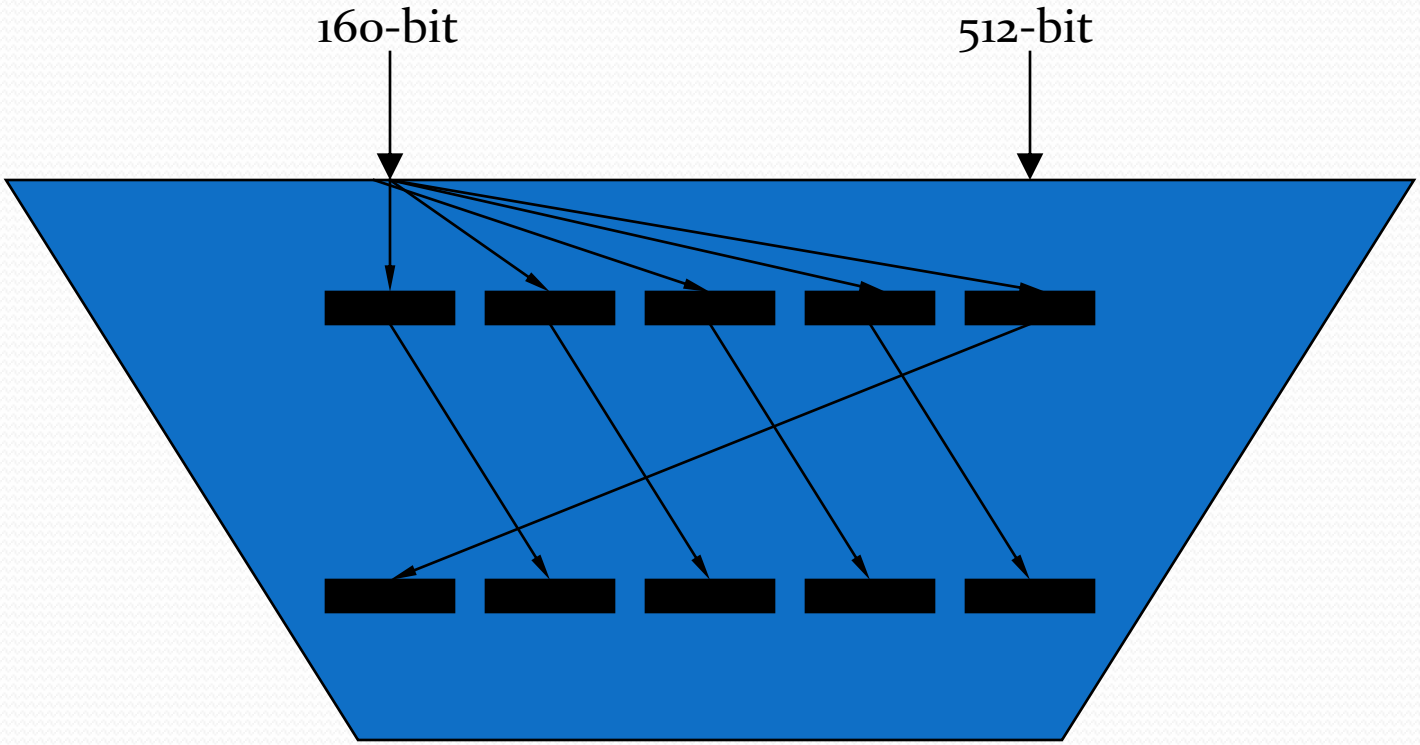
- Take the rightmost word.
- Add in the leftmost word rotated 5 bits.
- Add in a round-dependent function f of the middle three words.
- Add in a round-dependent constant.
- Add in a portion of the 512-bit message.

A Cryptographic Hash: SHA-1



Picture from Wikipedia

A Cryptographic Hash: SHA-1



One of 80 rounds

From SHA-1 to SHA-2

- Quick history: in 1993, NIST published SHA (Secure Hash Algorithm) as part of FIPS 180, *Secure Hash Standard*.

From SHA-1 to SHA-2

- Quick history: in 1993, NIST published SHA (Secure Hash Algorithm) as part of FIPS 180, *Secure Hash Standard*.
  - Designed by NSA

From SHA-1 to SHA-2

- Quick history: in 1993, NIST published SHA (Secure Hash Algorithm) as part of FIPS 180, *Secure Hash Standard*.
  - Designed by NSA
  - Withdrawn in 1995 and replaced with SHA-1

From SHA-1 to SHA-2

- Quick history: in 1993, NIST published SHA (Secure Hash Algorithm) as part of FIPS 180, *Secure Hash Standard*.
  - Designed by NSA
  - Withdrawn in 1995 and replaced with SHA-1
    - One modification: single bitwise rotation added

From SHA-1 to SHA-2

- Quick history: in 1993, NIST published SHA (Secure Hash Algorithm) as part of FIPS 180, *Secure Hash Standard*.
  - Designed by NSA
  - Withdrawn in 1995 and replaced with SHA-1
    - One modification: single bitwise rotation added
- In 2001, NIST revises FIPS 180 and adds SHA-2

From SHA-1 to SHA-2

- Quick history: in 1993, NIST published SHA (Secure Hash Algorithm) as part of FIPS 180, *Secure Hash Standard*.
  - Designed by NSA
  - Withdrawn in 1995 and replaced with SHA-1
    - One modification: single bitwise rotation added
- In 2001, NIST revises FIPS 180 and adds SHA-2
  - Also designed by NSA

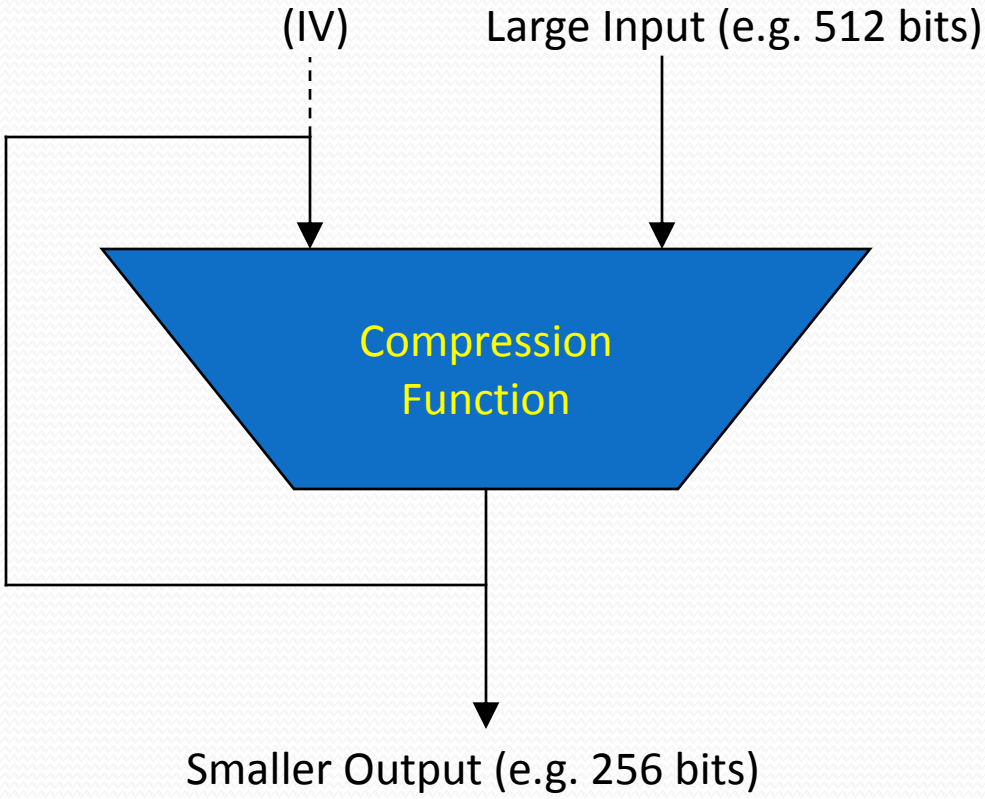
From SHA-1 to SHA-2

- Quick history: in 1993, NIST published SHA (Secure Hash Algorithm) as part of FIPS 180, *Secure Hash Standard*.
  - Designed by NSA
  - Withdrawn in 1995 and replaced with SHA-1
    - One modification: single bitwise rotation added
- In 2001, NIST revises FIPS 180 and adds SHA-2
  - Also designed by NSA
  - Originally 3 variants: SHA-256, SHA-384, SHA-512

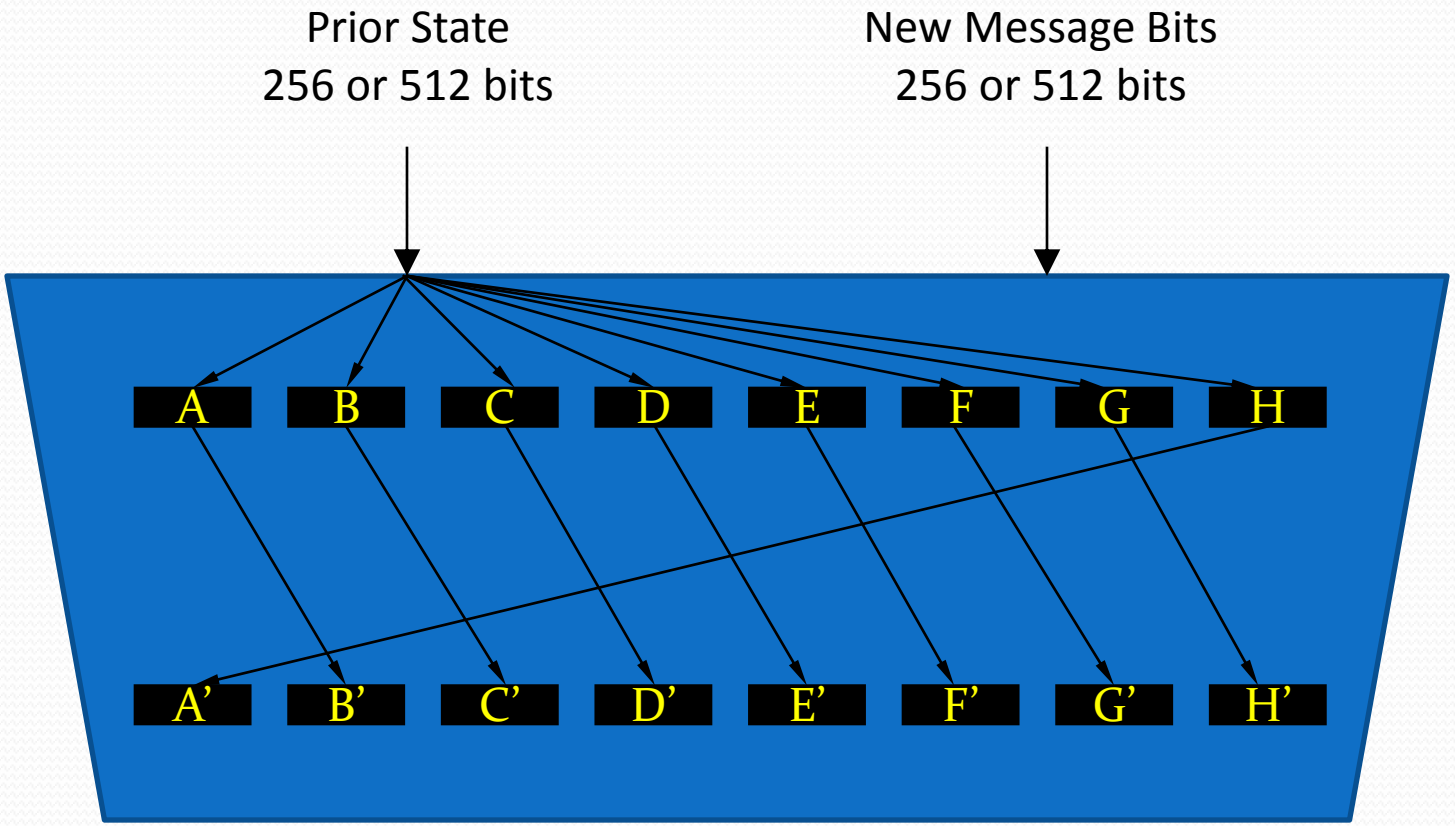
From SHA-1 to SHA-2

- Quick history: in 1993, NIST published SHA (Secure Hash Algorithm) as part of FIPS 180, *Secure Hash Standard*.
  - Designed by NSA
  - Withdrawn in 1995 and replaced with SHA-1
    - One modification: single bitwise rotation added
- In 2001, NIST revises FIPS 180 and adds SHA-2
  - Also designed by NSA
  - Originally 3 variants: SHA-256, SHA-384, SHA-512
    - SHA-224 added later

Merkle-Damgård Construction

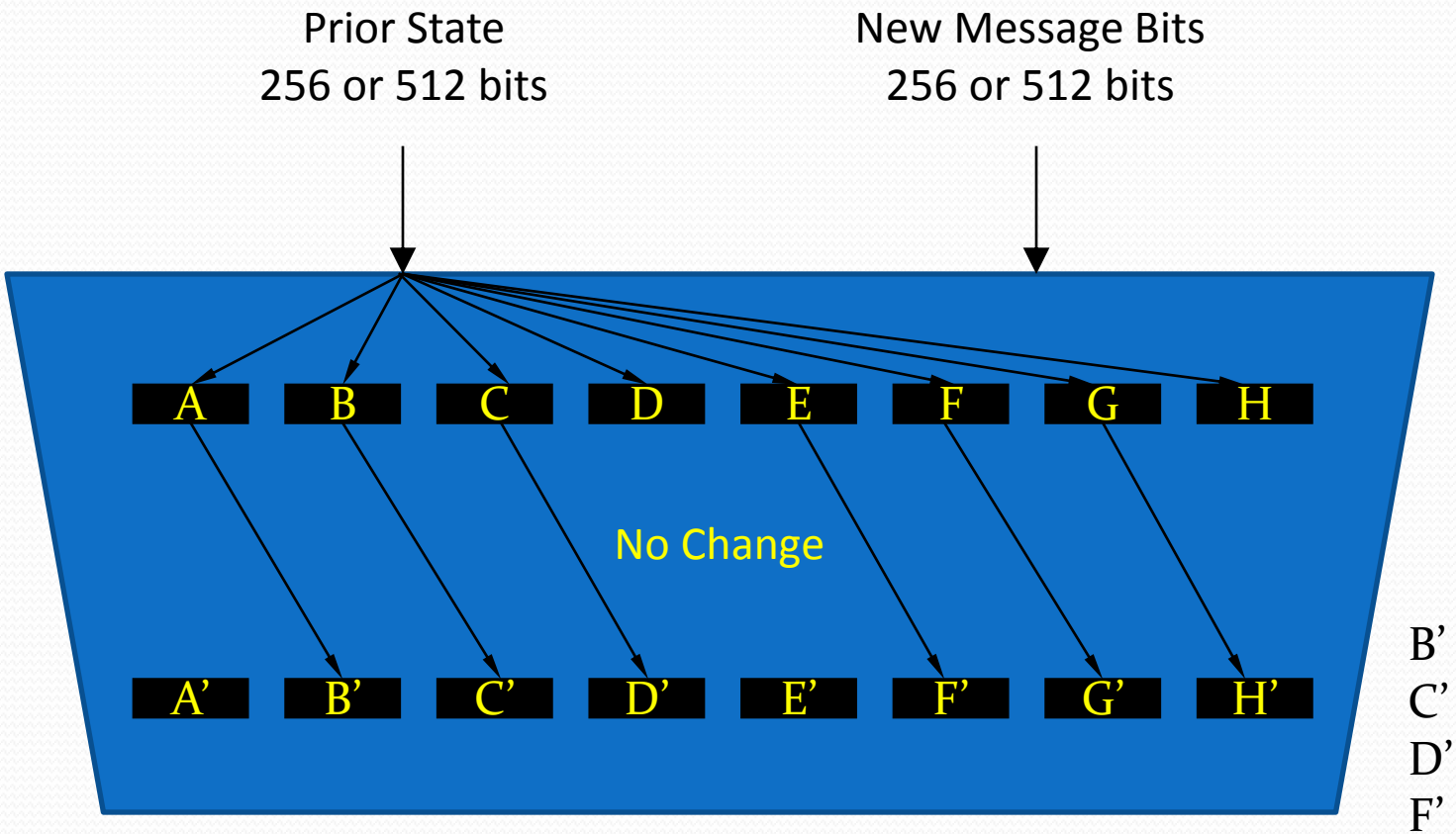


A Cryptographic Hash: SHA-2



One of 64 rounds

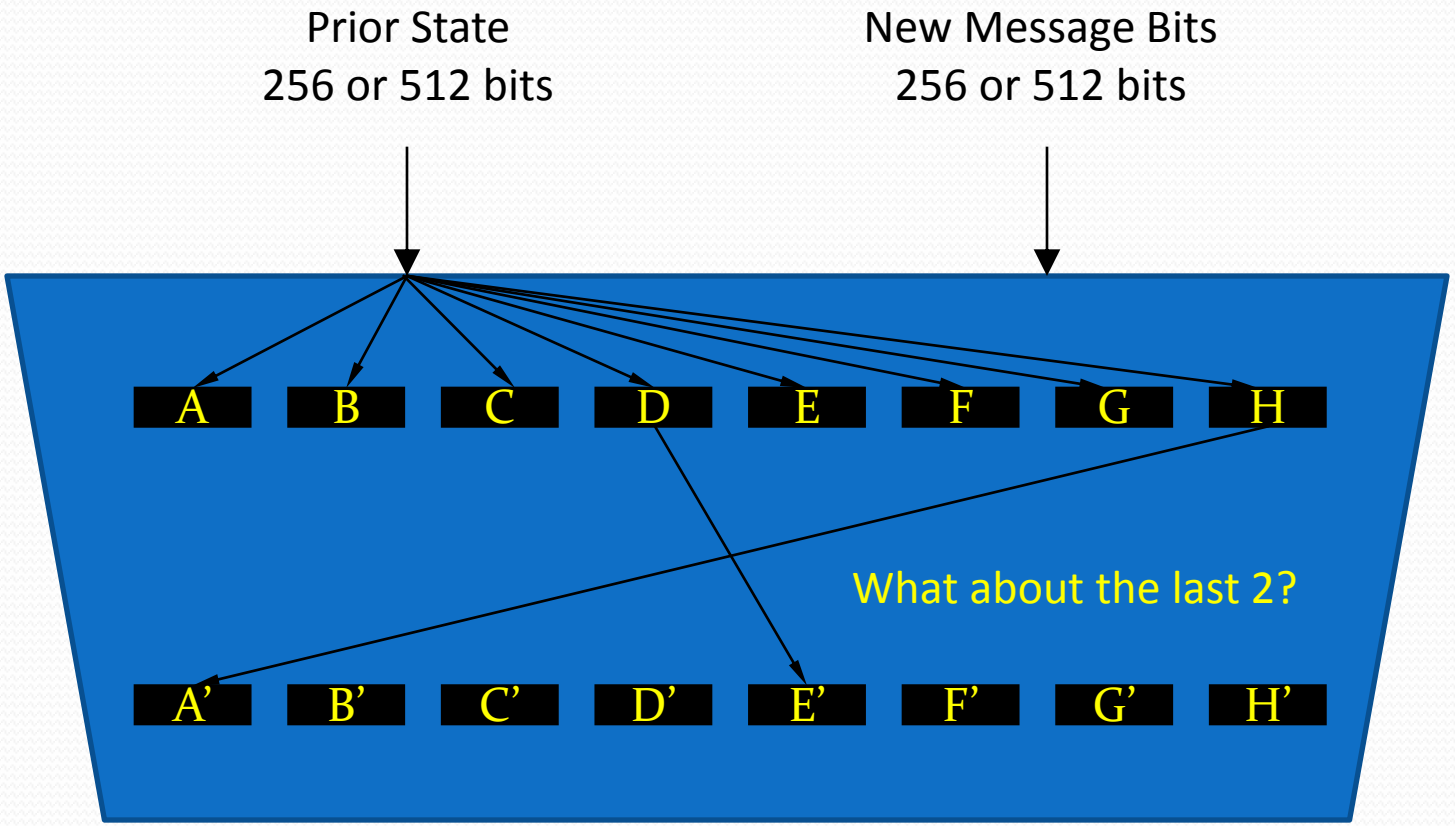
A Cryptographic Hash: SHA-2



One of 64 rounds

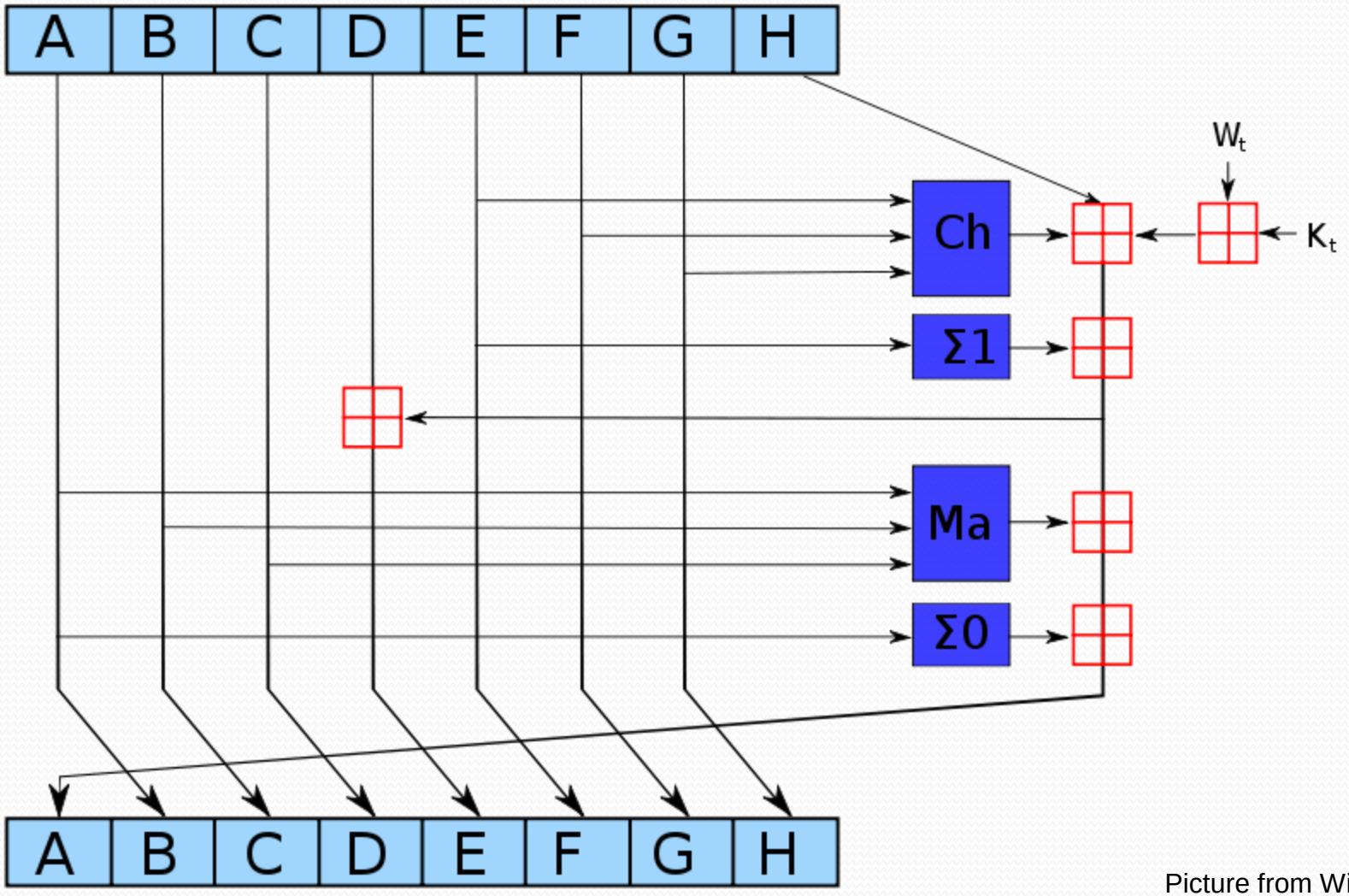
B' = A
C' = B
D' = C
F' = E
G' = F
H' = G

A Cryptographic Hash: SHA-2



One of 64 rounds

A Cryptographic Hash: SHA-2



Picture from Wikipedia

From SHA-2 to SHA-3

- In 2007, NIST announced that it would hold an open competition to pick a new “SHA-3” hash function

From SHA-2 to SHA-3

- In 2007, NIST announced that it would hold an open competition to pick a new “SHA-3” hash function
  - Similar to the competition held to pick AES.

From SHA-2 to SHA-3

- In 2007, NIST announced that it would hold an open competition to pick a new “SHA-3” hash function
  - Similar to the competition held to pick AES.
- Submissions were due 31 October 2008

From SHA-2 to SHA-3

- In 2007, NIST announced that it would hold an open competition to pick a new “SHA-3” hash function
  - Similar to the competition held to pick AES.
- Submissions were due 31 October 2008
  - 51 submissions accepted by NIST for Round One

From SHA-2 to SHA-3

- In 2007, NIST announced that it would hold an open competition to pick a new “SHA-3” hash function
  - Similar to the competition held to pick AES.
- Submissions were due 31 October 2008
  - 51 submissions accepted by NIST for Round One
  - 14 submissions advanced to Round 2

From SHA-2 to SHA-3

- In 2007, NIST announced that it would hold an open competition to pick a new "SHA-3" hash function
  - Similar to the competition held to pick AES.
- Submissions were due 31 October 2008
  - 51 submissions accepted by NIST for Round One
  - 14 submissions advanced to Round 2
  - 5 finalists (announced December 10, 2010)

From SHA-2 to SHA-3

- In 2007, NIST announced that it would hold an open competition to pick a new "SHA-3" hash function
  - Similar to the competition held to pick AES.
- Submissions were due 31 October 2008
  - 51 submissions accepted by NIST for Round One
  - 14 submissions advanced to Round 2
  - 5 finalists (announced December 10, 2010)
- SHA-3 expected to be formally selected in 2012

The SHA-3 Finalists

- BLAKE
- Grøstl (Knudsen et al.)
- JH
- Keccak (Keccak team, Daemen et al.)
- Skein (Schneier et al.)

State of the Art for Hashes

- Collisions have been demonstrated for MD4 and MD5 !
- The first SHA-1 collisions are likely to be found soon.

Integrity Checking

- Desirable for block ciphers
- Essential for stream ciphers

One-Way Hash Functions

- The idea of a checksum is great, but it is designed to prevent accidental changes in a message.
- For cryptographic integrity, we need an integrity check that is resilient against a smart and determined adversary.

Message Authentication Codes

A *Message Authentication Code* (MAC) is often constructed with a *keyed hash*.

If one hashes a secret key together with the correct message, an attacker who doesn't know the key will be unable to change the message without detection.

But how?

Cipher Integrity

- Original plaintext P.
- Encryption key K_1.
- MAC key K_2.
- Ciphertext $C=E_{K_1}(P)$.
- MAC $M=H_{K_2}(P)$ or $M=H_{K_2}(C)$.
- Transmit (C,M).

Message Authentication Codes

MAC key K, plaintext P, ciphertext $C=E(P)$.

$MAC=H(K,P)$? $MAC=H(P,K)$?
$MAC=H(K,C)$? $MAC=H(C,K)$?

There are weaknesses with *all* of the above.

$HMAC = H(K,H(K,P))$

Crypto Hygiene

Do I really need to use different keys for encryption and integrity?

It's always a good idea to use separate keys for separate functions, **but the keys can be derived from the same master**.

K_1=H("Key1",K) K_2=H("Key2",K)

Message Digests

A *message digest* is a short "unforgeable" fingerprint of a long message.

A simple hash can serve as a message digest.