On recent attacks against Cryptographic Hash Functions

Martin Ekerå & Henrik Ygge

Outline

- **First part**
  - Preliminaries
  - Which cryptographic hash functions exist?
    - What degree of security do they offer?
  - An introduction to Wang's attack
- **Second part**
  - Wang's attack applied to MD5
  - Demo

Part I

Operators

Symbol	Meaning
$x \boxplus y$	Addition modulo 2^n
$x \boxminus y$	Subtraction modulo 2^n
$x \oplus y$	Exclusive OR
$x \wedge y$	Bitwise AND
$x \vee y$	Bitwise OR
$\neg x$	The negation of x.
$x \ll s$	Shifting of x by s bits to the left.
$x \lll s$	Rotation of x by s bits to the left.

Bitwise Functions

Function	
IF (x, y, z)	$(x \wedge y) \vee ((\neg x) \wedge z)$
XOR (x, y, z)	$x \oplus y \oplus z$
MAJ (x, y, z)	$(x \wedge y) \vee (y \wedge z) \vee (z \wedge x)$
XNO (x, y, z)	$y \oplus ((\neg z) \vee x)$

- The functions above are all bitwise.

Hash Functions

- A hash function maps elements from a finite or infinite domain, into elements of a fixed size domain.

Attacks on Hash Functions

- **Collision attack**
  Find m and m' ≠ m such that H(m) = H(m').

- **First pre-image attack**
  Given h find m such that h = H(m).

- **Second pre-image attack**
  Given m find m' ≠ m such that H(m) = H(m').

Attack Complexities

- **Collision attack**
  Naïve complexity $O(2^{n/2})$ due to the birthday paradox.

- **First pre-image attack**
  Naïve complexity $O(2^{n})$

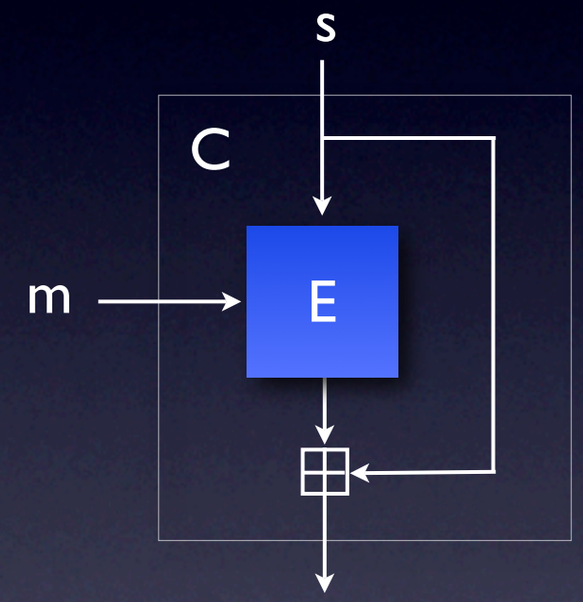
- **Second pre-image attack**
  Naïve complexity $O(2^{n})$

Cryptographic Hash Functions

- It is desirable for a **cryptographic** hash function to be collision resistant, first pre-image resistant and second pre-image resistant.

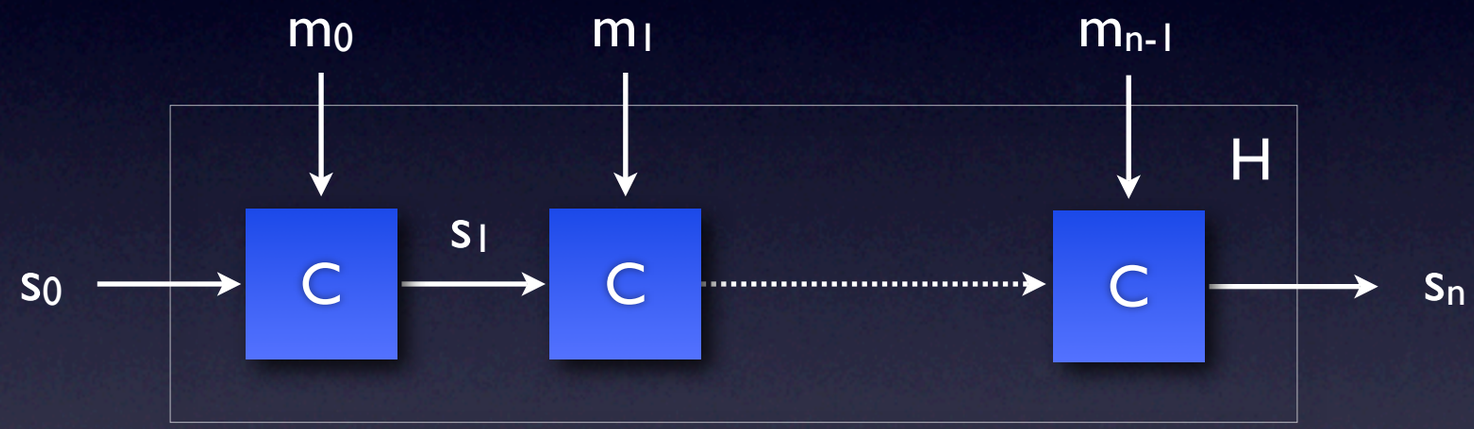
Construction Schemes

Davies-Meyer



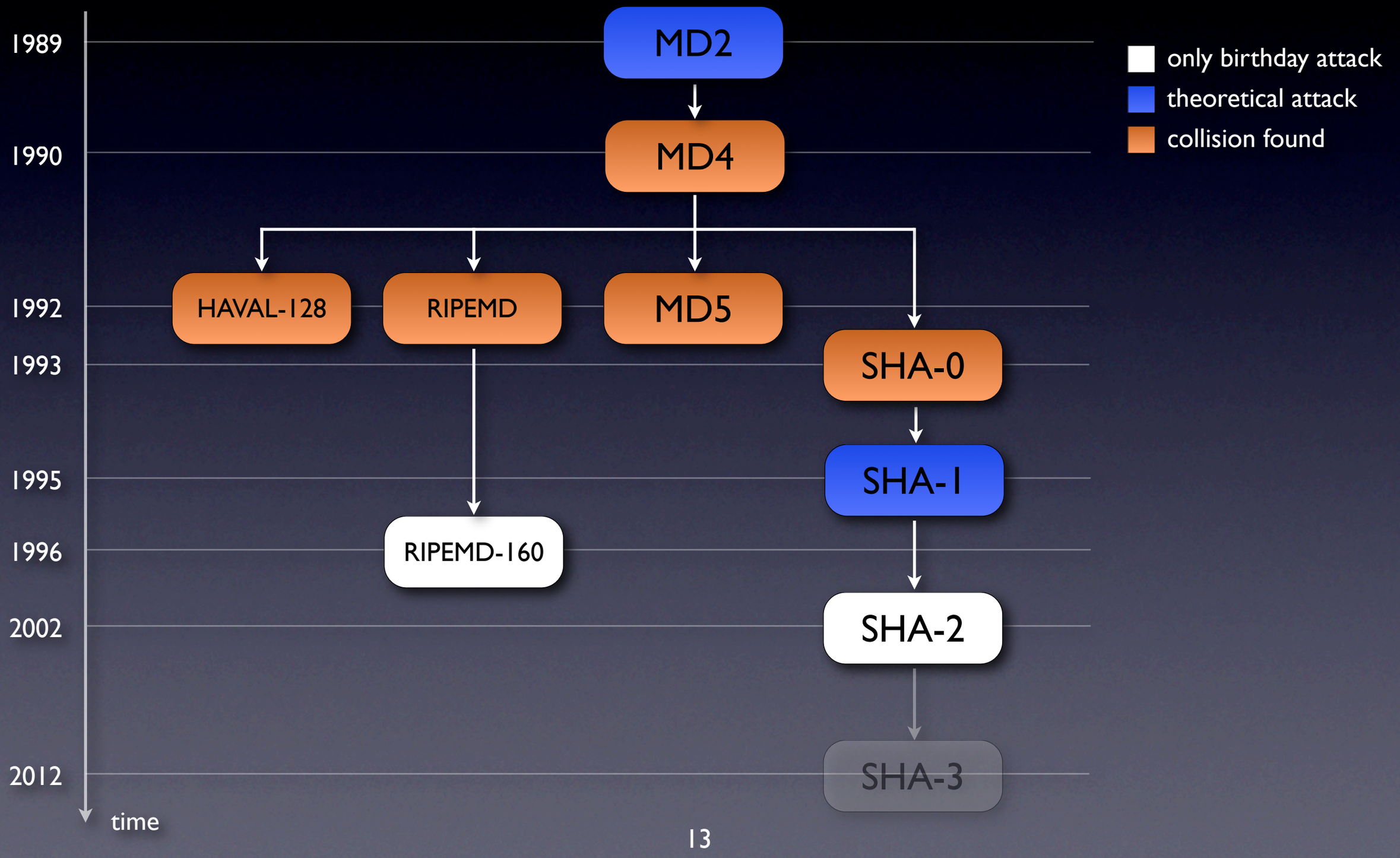
- The Davies-Meyer scheme builds a compression function C from an encryption function E.

Merkle-Damgård



- The Merkle-Damgård scheme builds a collision resistant hash function H from a collision resistant compression function C.

A Genealogy


1989
1990
1992
1993
1995
1996
2002
2012
time
MD2
MD4
HAVAL-128
RIPEMD
MD5
SHA-0
SHA-1
RIPEMD-160
SHA-2
SHA-3
only birthday attack
theoretical attack
collision found

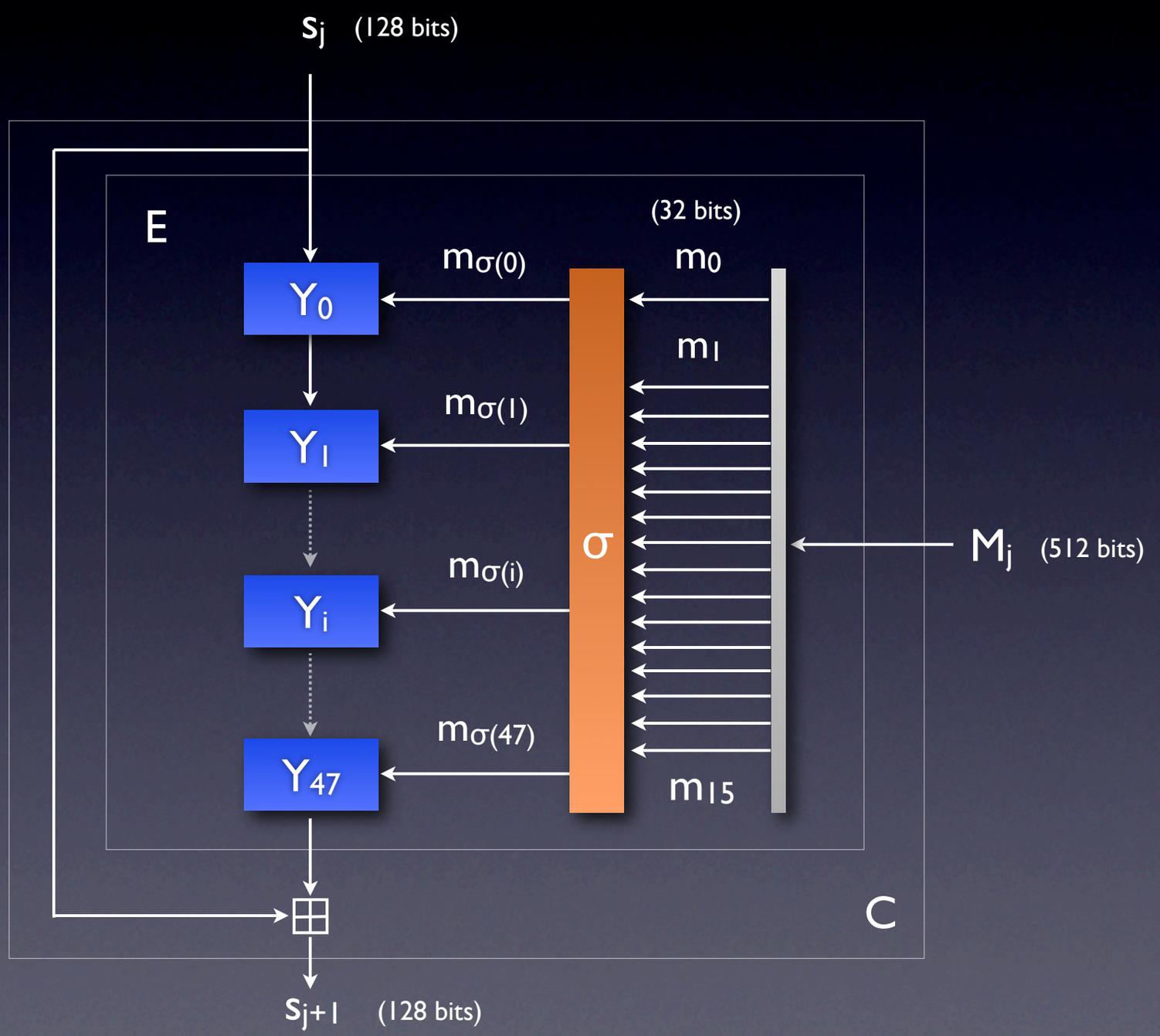
MD4

- Designed by Ron Rivest at MIT in 1990 as a successor to MD2.
- Established the basic structure of most hash functions in use today.

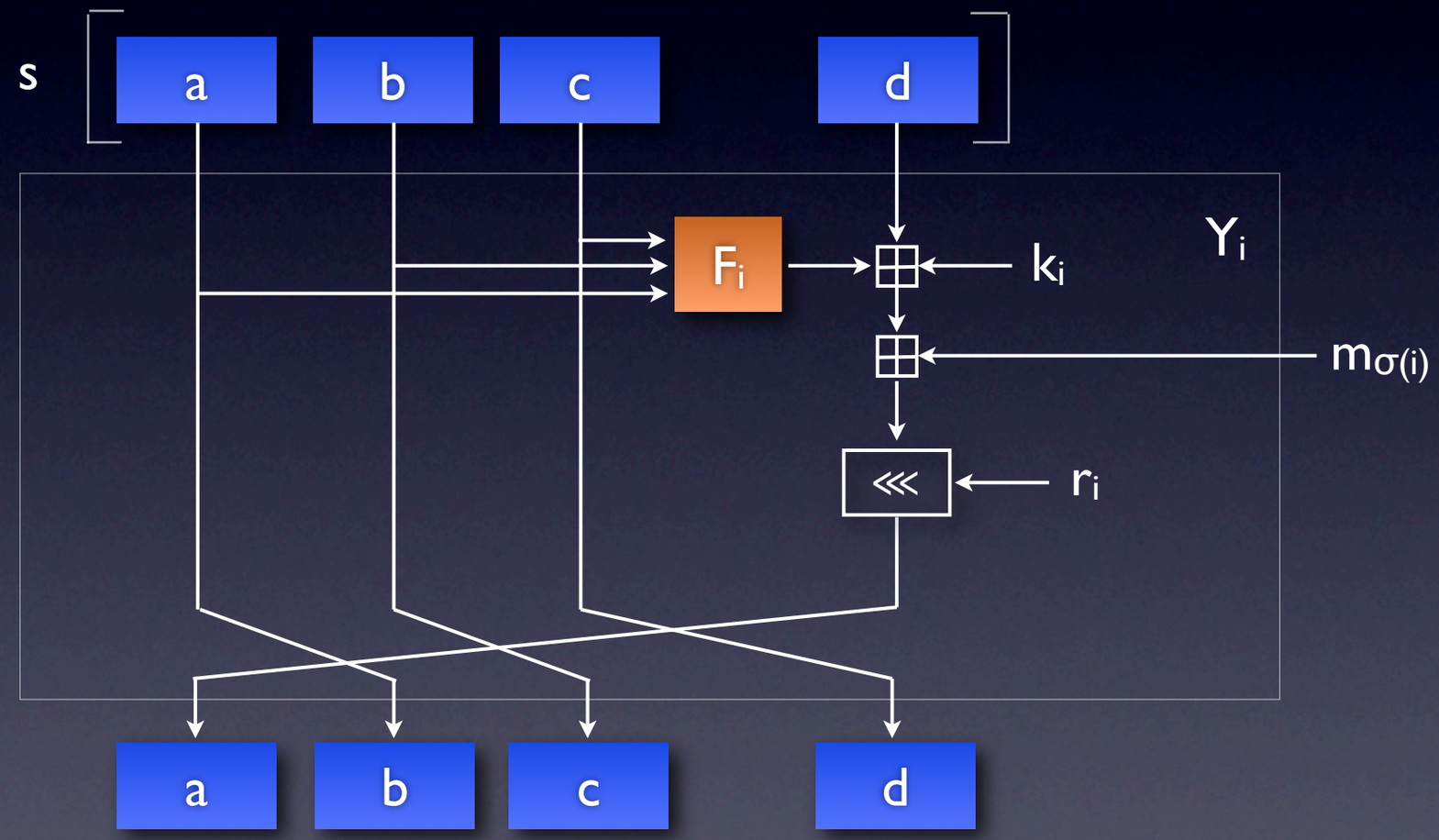
Structure of MD4

- Iterated encryption function
  - Three rounds with 16 steps in each round.
  - Encrypts the 128 bit input state under a 512 bit message block.
- Compression function created using the Davies-Meyer scheme.
- Hash function created using the Merkle-Damgård scheme.

MD4 Compression Function



MD4 Step Function



Round Functions & Constants

Round	Step	F_i	k_i
1	1 to 16	IF (a, b, c)	`0x00000000`
2	17 to 32	MAJ (a, b, c)	`0x5A827999`
3	33 to 48	XOR (a, b, c)	`0x6ED9EBA1`

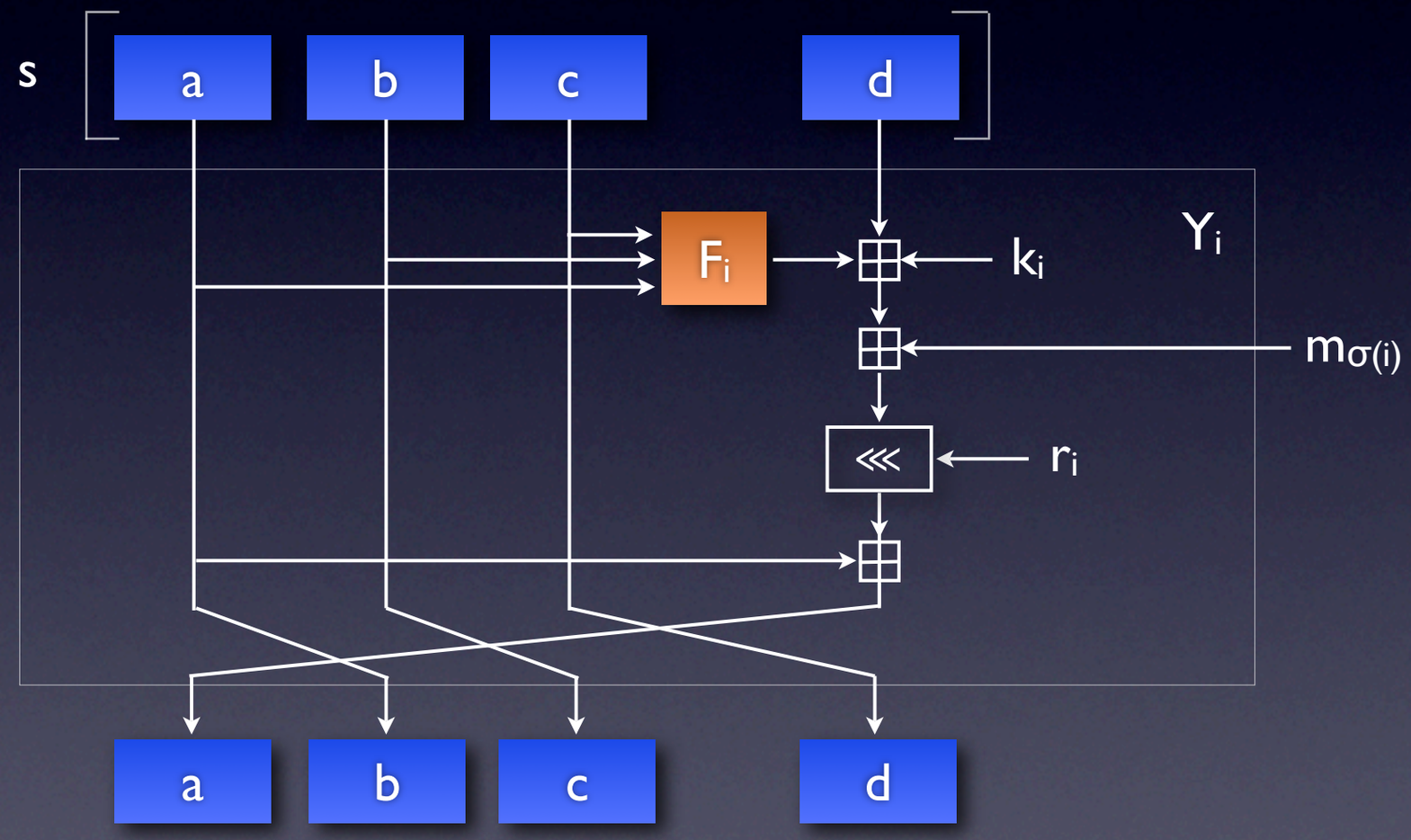
Attacks on MD4

- Last two rounds attacked in 1991 by den Boer and Bosselaers.
- Full collision with complexity $O(2^{22})$ by Dobbertin in 1996.
- Wang et al. presented an attack in 2004 using "hand calculation" $O(2^{8})$.
- The current complexity of finding a collision is less than the complexity of one pass through the compression function.
- MD4 should not be used anymore.

MD5

- Designed by Ron Rivest in 1992 as a successor to MD4.
  - A response to the analytic attacks of den Boer and Bosselaers on MD4.
- Standardized in RFC 1321 and widely used.
- Same overall structure as its predecessor.
  - One additional round. Different round functions.
  - Uses a new constant in each step.
  - Slightly modified step function.

MD5 Step Function



Round Functions & Constants

Round	Step	F_i
1	1 to 16	IF (a, b, c)
2	17 to 32	IF (c, a, b)
3	33 to 48	XOR (a, b, c)
4	49 to 64	XNO (a, b, c)

- The 64 steps are divided into 4 rounds with 16 steps each.
- A unique constant k_i is now used in each step.

Attacks on MD5

- Psuedo-collision $C(m, s_1) = C(m, s_2)$ by den Boer and Bosselaers in 1993.
- Psudo-collision $C(m_1, s_1) = C(m_2, s_2)$ by Dobbertin in 1996.
- Full collision by Wang et al. with complexity $O(2^{37})$ in 2004.
  - Wang's attack was optimized by Vlastimil Klíma in 2006.
- NIST recommends against using MD5.

SHA-0

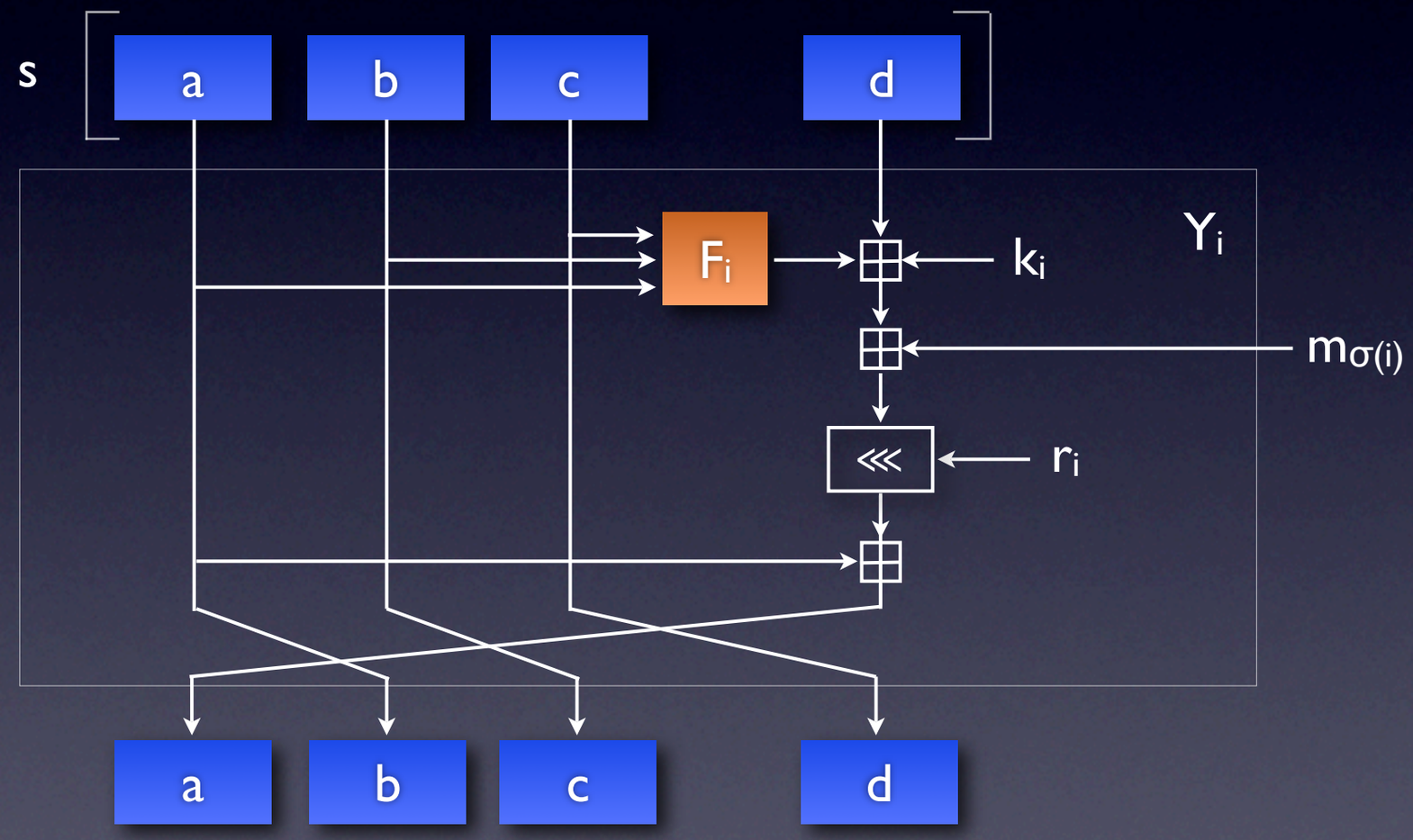
- Designed by the NSA and standardized by NIST in 1993.
- Was created out of the concerns that the hash digest size of MD5 was becoming too short.
- Hash digest length is 160 bits which gives a complexity of $O(2^{80})$ for a brute force attack.

SHA-0

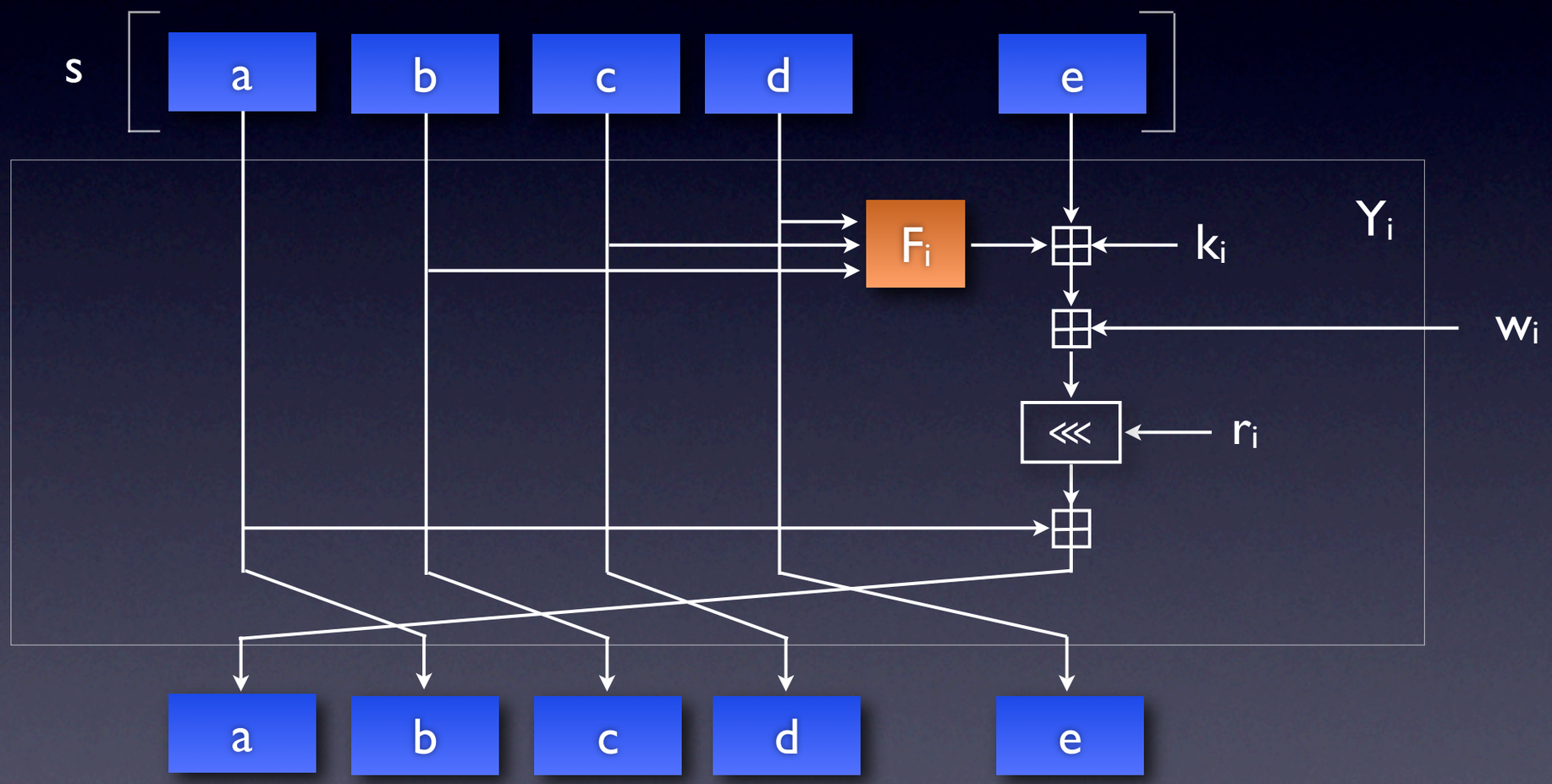
- Uses a more complex message expansion:

$$w_i = \begin{cases} m_i & i < 16 \\ w_{i-3} \oplus w_{i-8} \oplus w_{i-14} \oplus w_{i-16} & \text{otherwise} \end{cases}$$

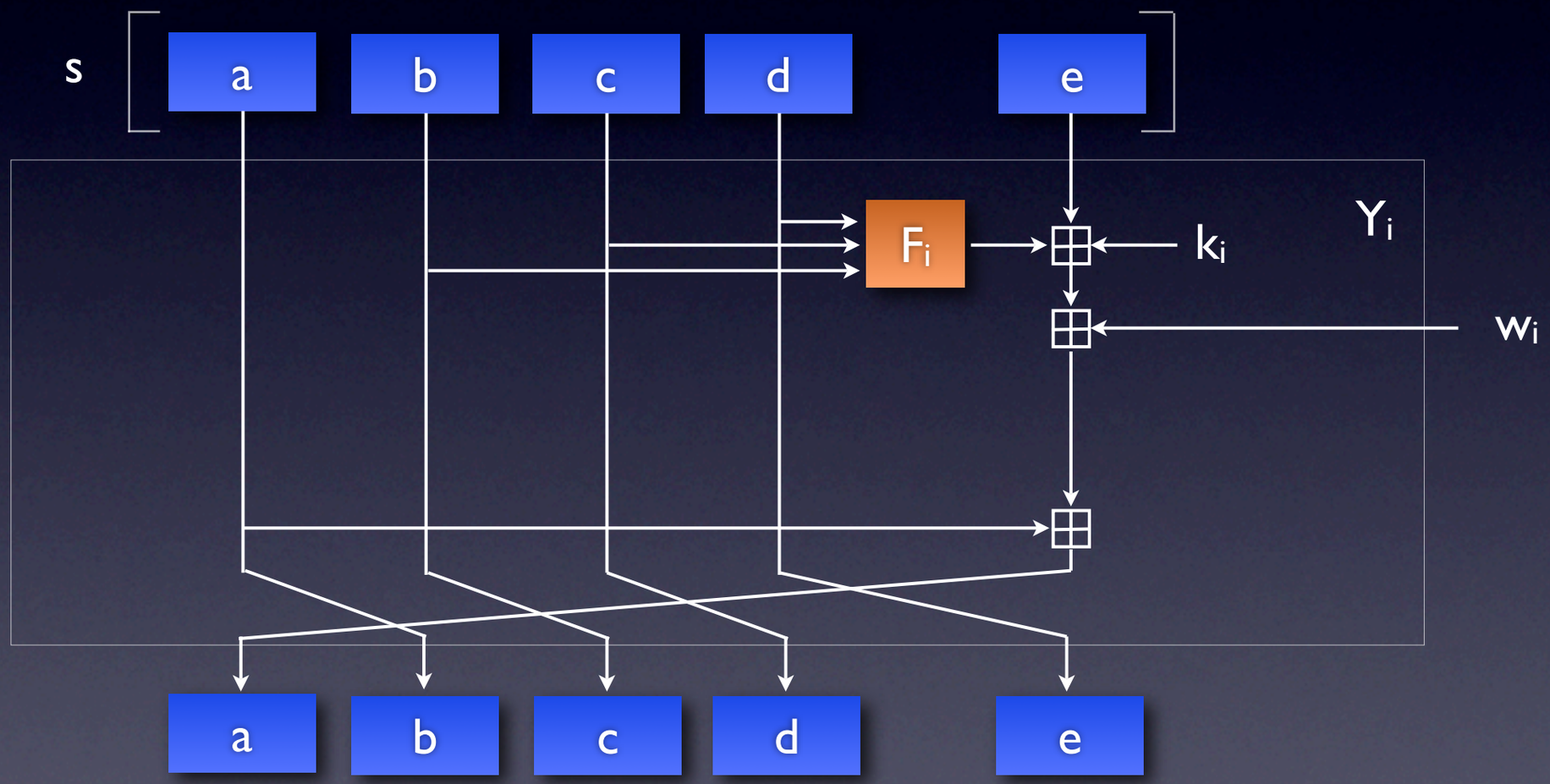
MD5 Step Function



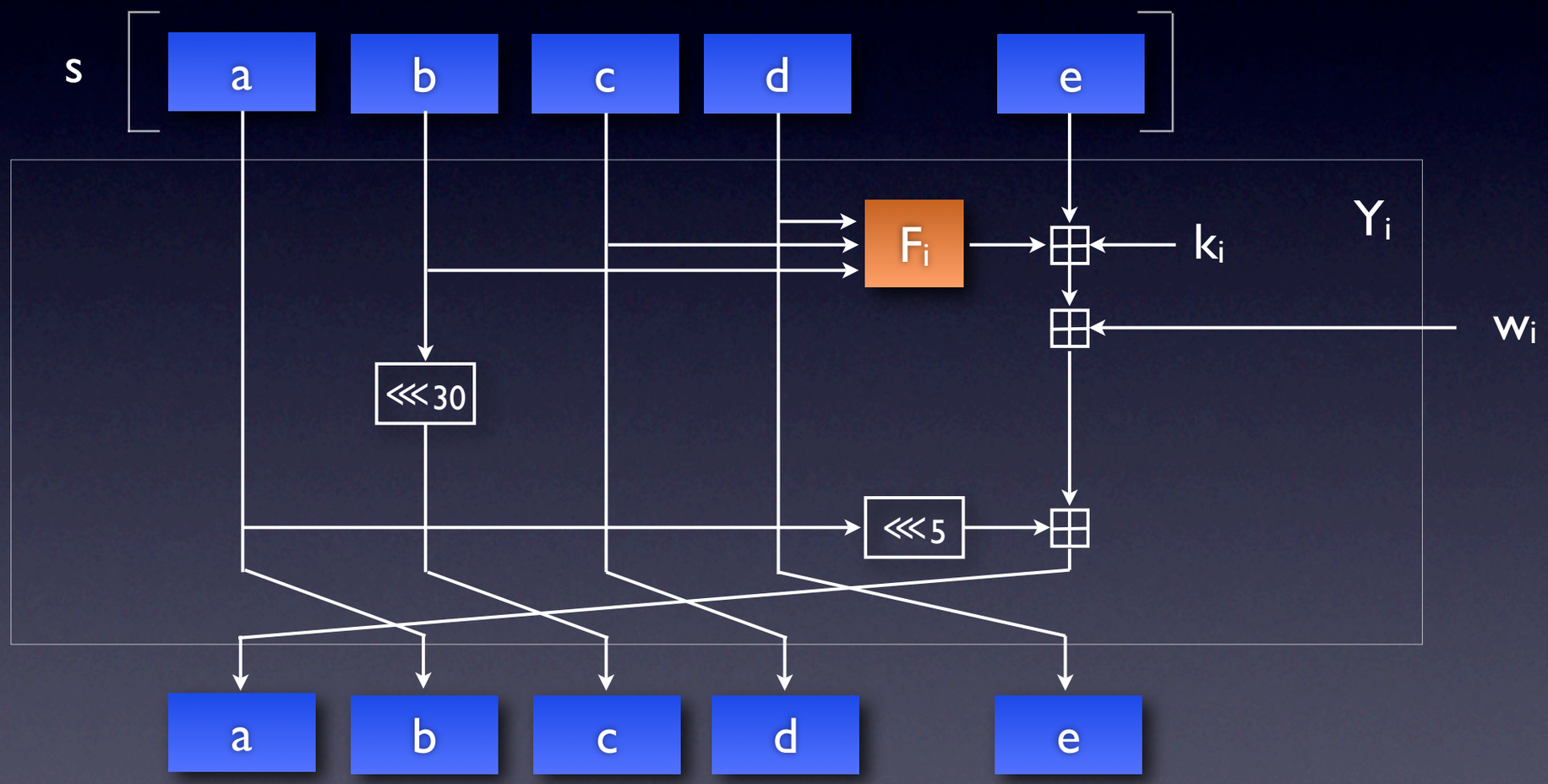
SHA-0 Step Function



SHA-0 Step Function



SHA-0 Step Function



Round Functions & Constants

Round	Step	F_i	k_i
1	1 to 20	IF (b, c, d)	`0x5A827999`
2	21 to 40	XOR (b, c, d)	`0x6ED9EBA1`
3	41 to 60	MAJ (b, c, d)	`0x8F1BBCDC`
4	61 to 80	XOR (b, c, d)	`0xCA62C1D6`

- The 80 steps are divided into 4 rounds with 20 steps each.

Attacks on SHA-0

- The first attack was published by Chabaud and Joux in 2002 with complexity $O(2^{61})$.
- Biham and Shamir improved upon the attack and reduced the complexity to $O(2^{51})$.
- The first collision was found by Joux in 2004 after 80 000 CPU hours on a 256 itanium processor cluster.
- Wang et al. published an attack in 2005 with complexity $O(2^{39})$.
- SHA-0 is not recommended for use by NIST anymore.

SHA-1

- Standardized by NIST in 1995 as a replacement for SHA-0, in response to concerns voiced by NSA over a weakness in the message schedule.
- NSA never officially explained the nature of the weakness.
- More recent studies have verified that this change has strengthened the hash function.

SHA-1

- Uses an even more complex message expansion:

$$w_i = \begin{cases} m_i & i < 16 \\ (w_{i-3} \oplus w_{i-8} \oplus w_{i-14} \oplus w_{i-16}) \lll 1 & \text{otherwise} \end{cases}$$

Attacks on SHA-1

- No collision has yet been found, but a theoretical attack with complexity $O(2^{63})$ was presented by Wang et al. in 2005.

- SHA-1 should not be used in new implementations and NIST recommends that the use of SHA-1 be discontinued by 2010 in favor of SHA-2.

SHA-2

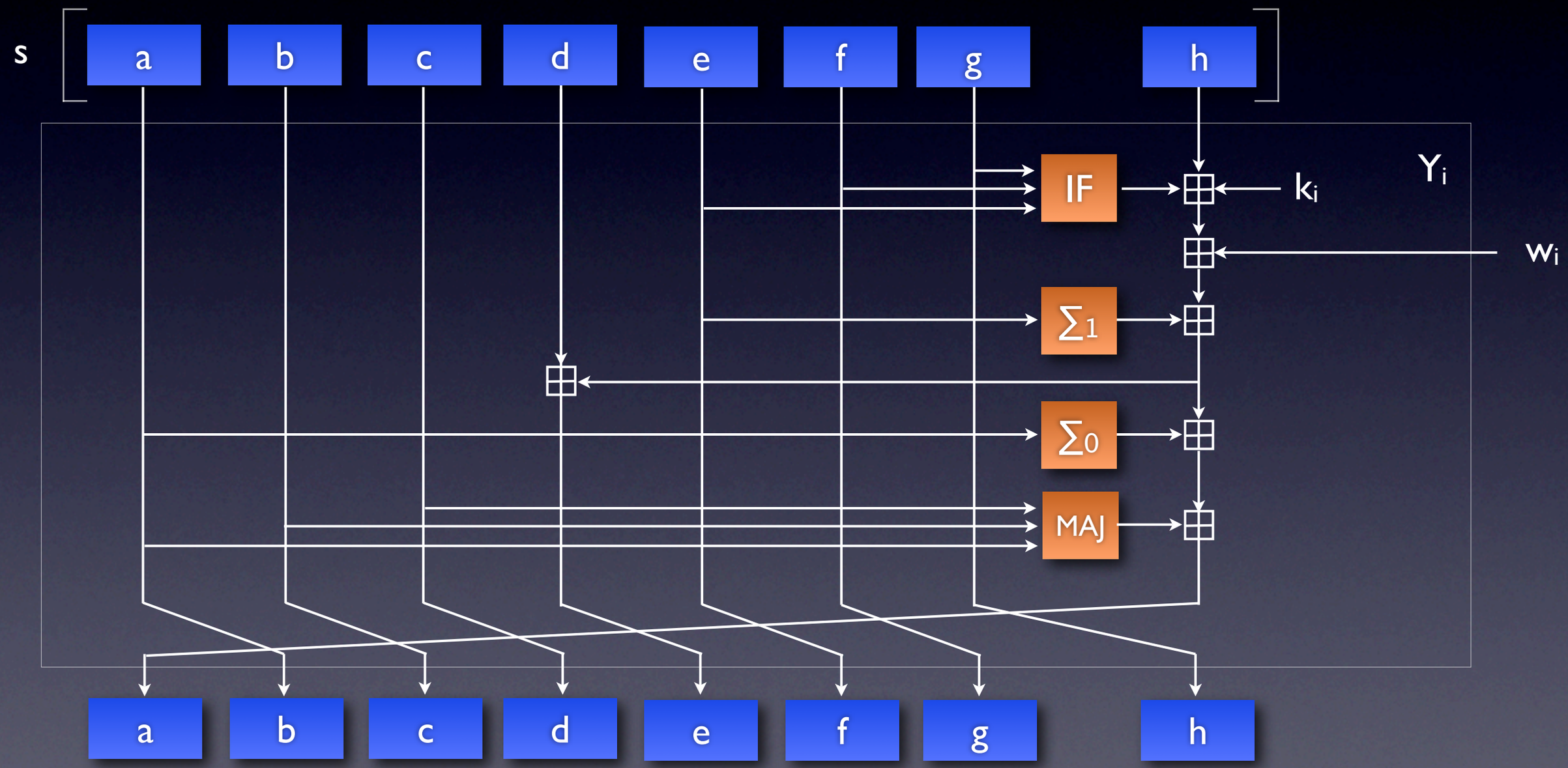
- Designed by the NSA and standardized by NIST in 2002.
- Consists of a family of hash functions
  - SHA-224
  - SHA-256
  - SHA-384
  - SHA-512

SHA-2

Function	Digest Length	Message Block Length	Steps	Word Length	Max Input Length
SHA-224	224 bits	512 bits	64	32 bit	$2^{64} - 1$ bits
SHA-256	256 bits	512 bits	64	32 bit	$2^{64} - 1$ bits
SHA-384	384 bits	1024 bits	80	64 bit	$2^{128} - 1$ bits
SHA-512	512 bits	1024 bits	80	64 bit	$2^{128} - 1$ bits

- More complex message expansion involving shift and rotate operations.
- No concept of rounds since the same step function is always used.

SHA-2 Step Function



SHA-2

- SHA-2 was developed...
- ...as a response to the attacks on SHA-0 and SHA-1.
- ...since 160 bit digests provide insufficient security.
- ...because a more flexible set of hash functions was needed.
- ...to provide 64-bit support.

Attacks on SHA-2

- No theoretical attack has been presented as of yet.

- SHA-2 is recommended by NIST and should be used in newly developed software until the release of SHA-3, which is planned in 2012.

SHA-3

- Specified as a drop-in replacement for SHA-2.
  - Same hash digest lengths as SHA-2.
- Public competition.
  - Announced after concerns that an attack would be found on SHA-2.
  - "Should be secure for several decades to come."

SHA-3 Timeline

2007	The 2nd of November 2007, NIST sent out a formal request for candidate algorithms for the standardization of SHA-3.
2008	Submissions accepted until the 31st of October 2008.
2009	First round candidates to be presented in 2009.
2010	Final candidates to be announced in 2010.
2012	Standard to be presented in 2012.

time

Wang's Attack

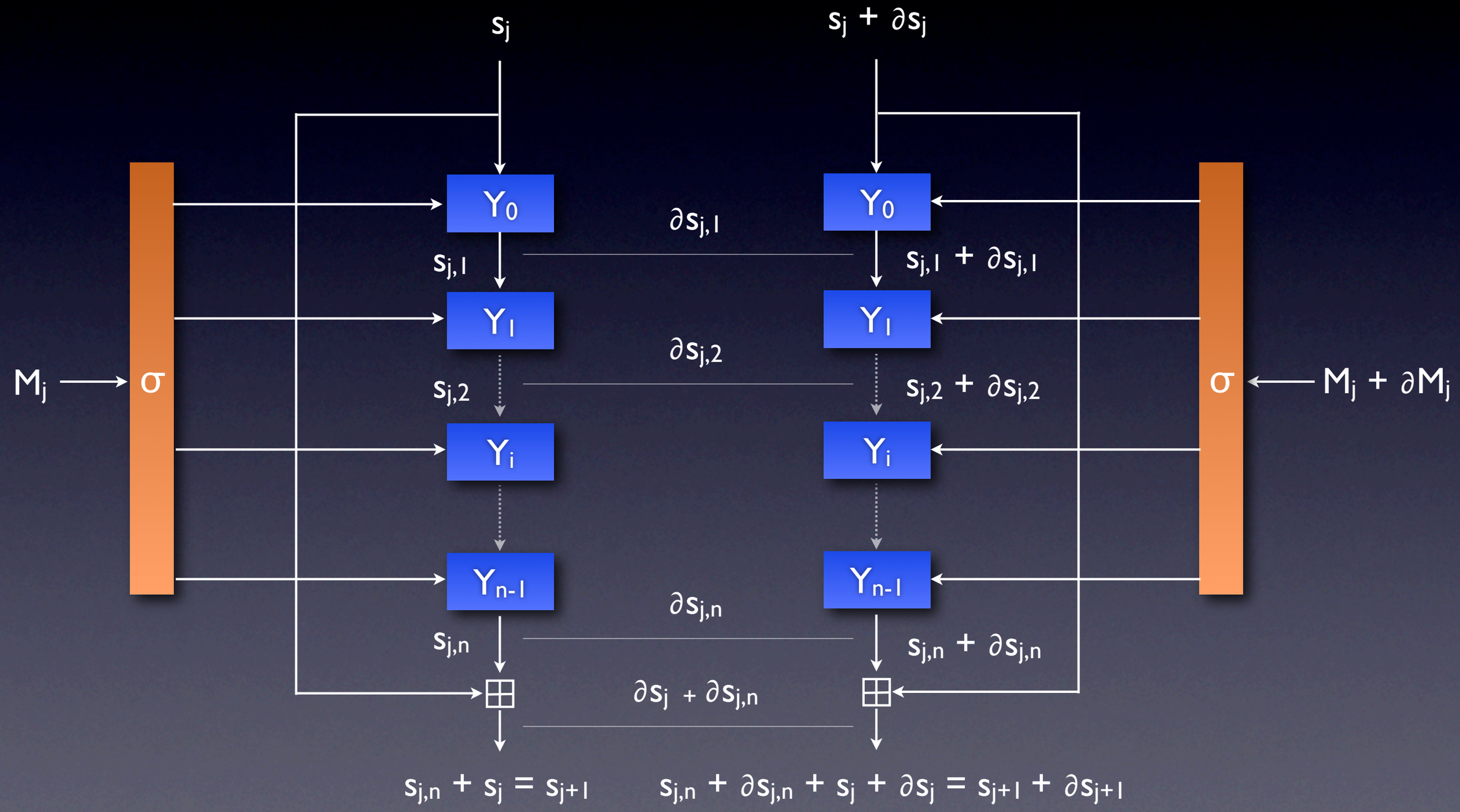
Wang's attack

- In 2004 Wang et al. presented the first collisions on MD5, RIPEMD and HAVAL-128, as well as a new collision on MD4.
- They used a differential attack to analyze how small message and input state differences propagate through the step functions.
- It is a generic attack which is applicable to most iterated hash functions.
- Used to find a collision on SHA-0 in 2004, as well as a collision on SHA-1 reduced to 58 steps.

Outline of Wang's Attack

- Select an appropriate message difference ∂M.
- Select an input state difference ∂s.
- Derive a differential path describing how differences propagate through the step function.

Outline of Wang's Attack

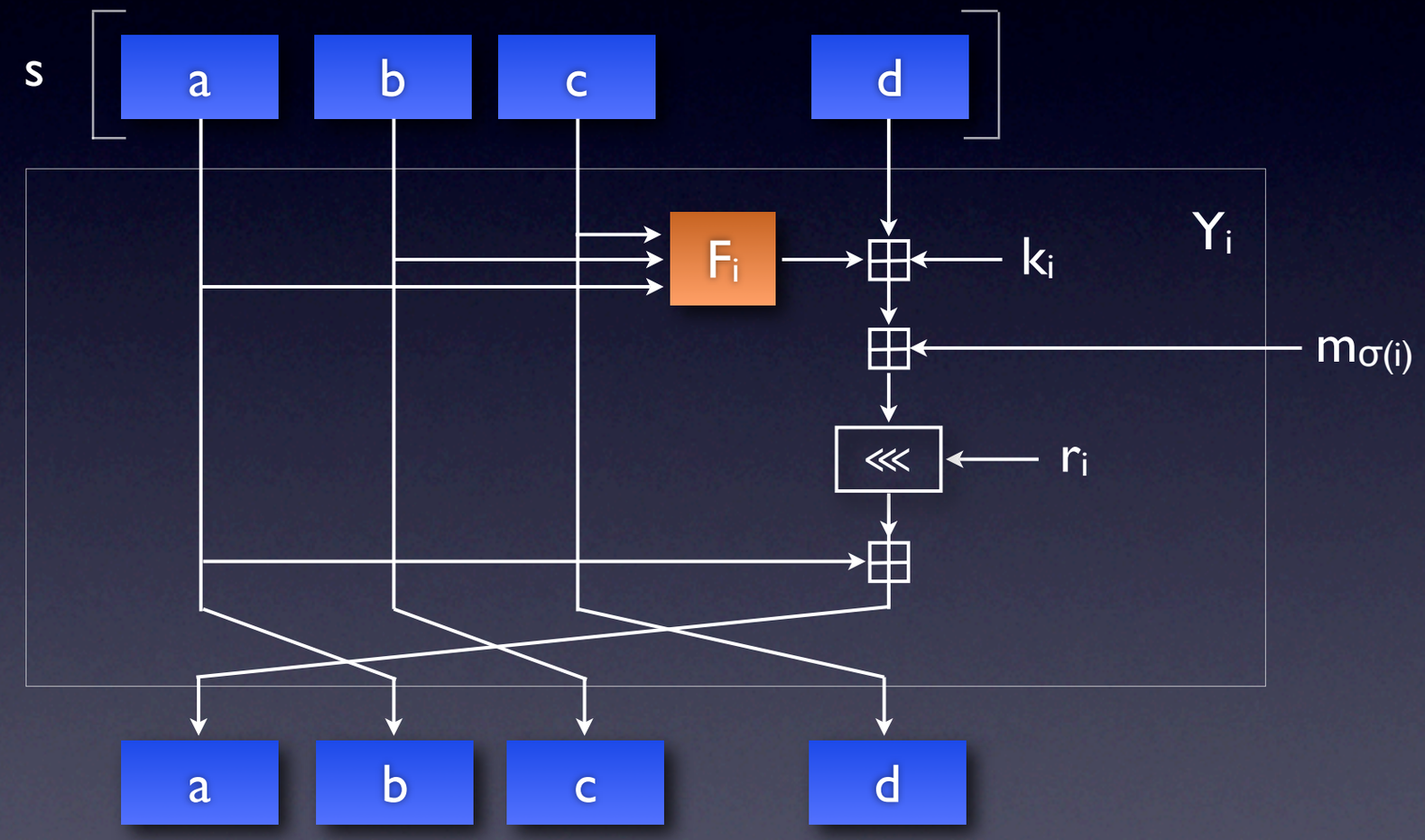


Outline of Wang's Attack

- Derive a set of pseudo-sufficient conditions on bit differences in the intermediary states, for the differential path to hold.
- Employ message modification techniques to efficiently search for a message M that fulfills the conditions for the unperturbed left branch.
- Then, the messages M and M + ∂M with input states s and s + ∂s give the desired output difference.
  - We can select differences to obtain collisions.

Part II

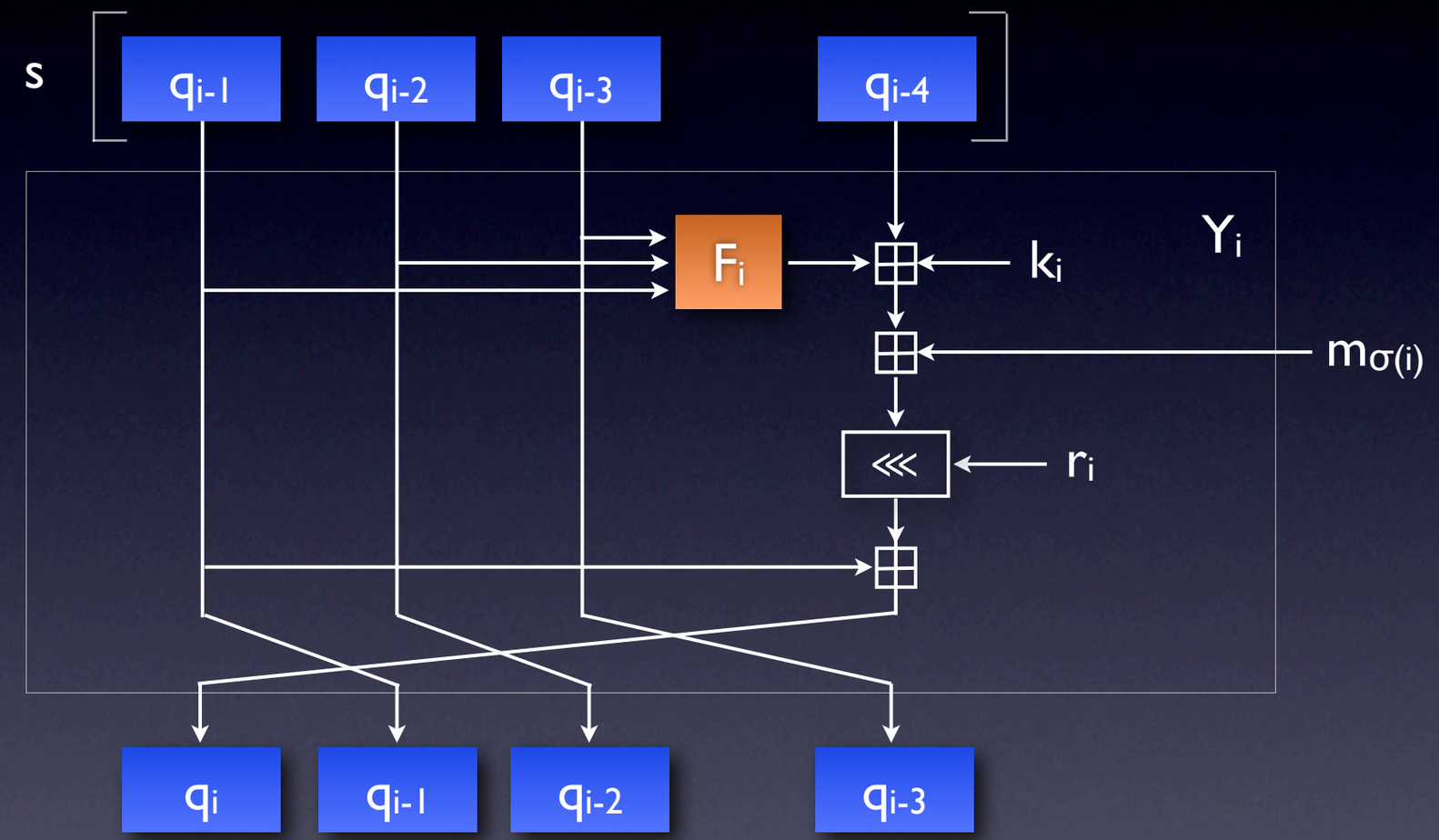
Another look at MD5



Another look at MD5

- Only the A register is updated in each step
- Create a vector Q = [q_{-3}, .. q_1, .., q_{64}] where
  - q_i is the value set in the A register in step i > 0
  - q_{-3} to q_0 are the IV values

MD5 Step Function



- Or on equation form

$$q_i = q_{i-1} \boxplus \underbrace{(q_{i-4} \boxplus F_i (q_{i-1}, q_{i-2}, q_{i-3}) \boxplus m_{\sigma(i)} \boxplus k_i)}_{T_i} \lll r_i$$

Round Functions & Constants

Round	Step	F_i
1	1 to 16	IF (q_{i-1}, q_{i-2}, q_{i-3})
2	17 to 32	IF (q_{i-3}, q_{i-1}, q_{i-2})
3	33 to 48	XOR (q_{i-1}, q_{i-2}, q_{i-3})
4	49 to 64	XNO (q_{i-1}, q_{i-2}, q_{i-3})

- The 64 steps are divided into 4 rounds with 16 steps each.
- A unique constant k_i is now used in each step.

The Permutation σ

- Sigma is a permutation of the message words, such that each message word is used exactly once in each round.

Round	Step i	σ(i)
1	1 to 16	i-1
2	17 to 32	(5(i-1) + 1) mod 16
3	33 to 48	(3(i-1) + 5) mod 16
4	49 to 64	7(i-1) mod 16

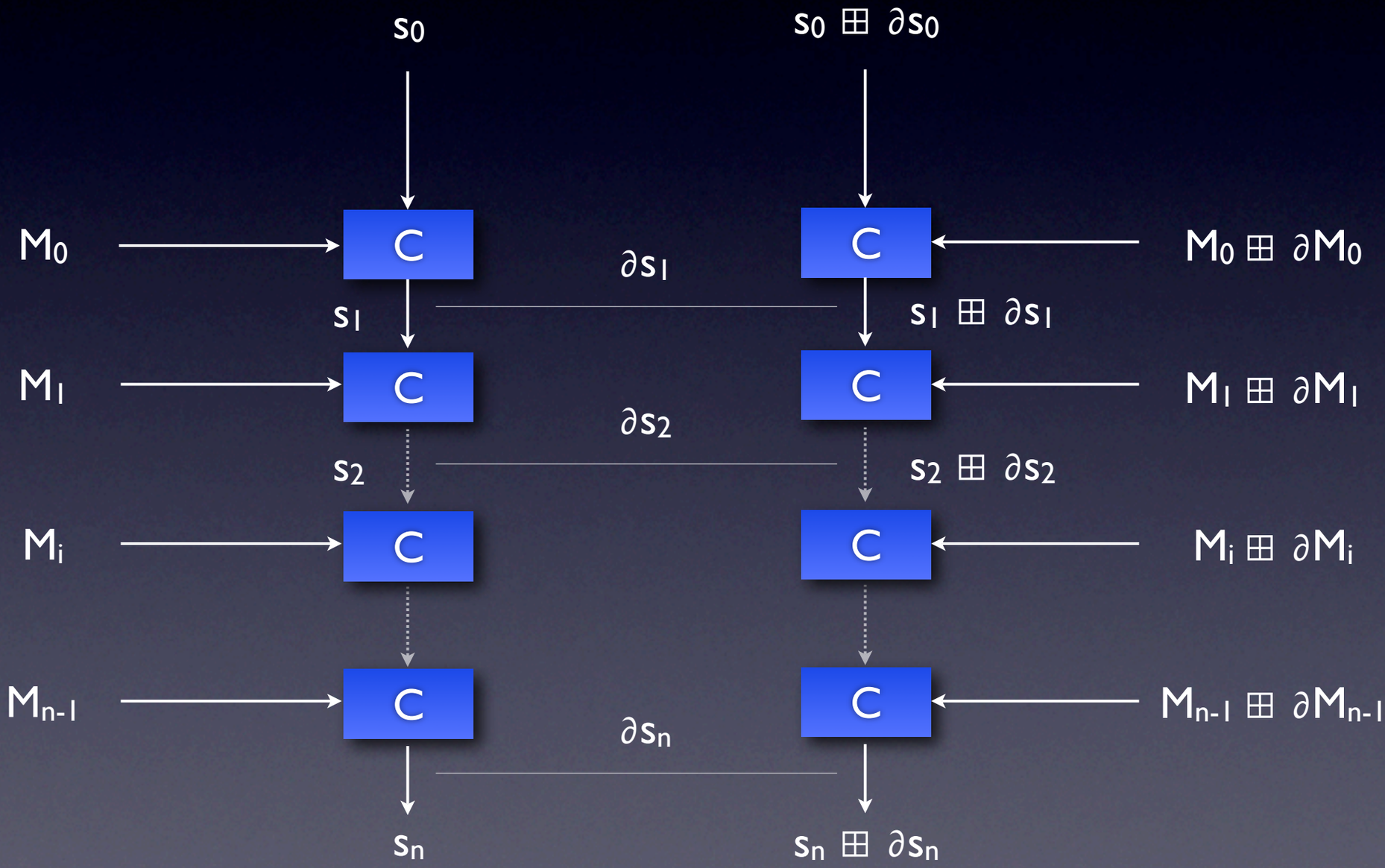
The Rotational Constants r_i

- Four rotational constants are used cyclically in each round.

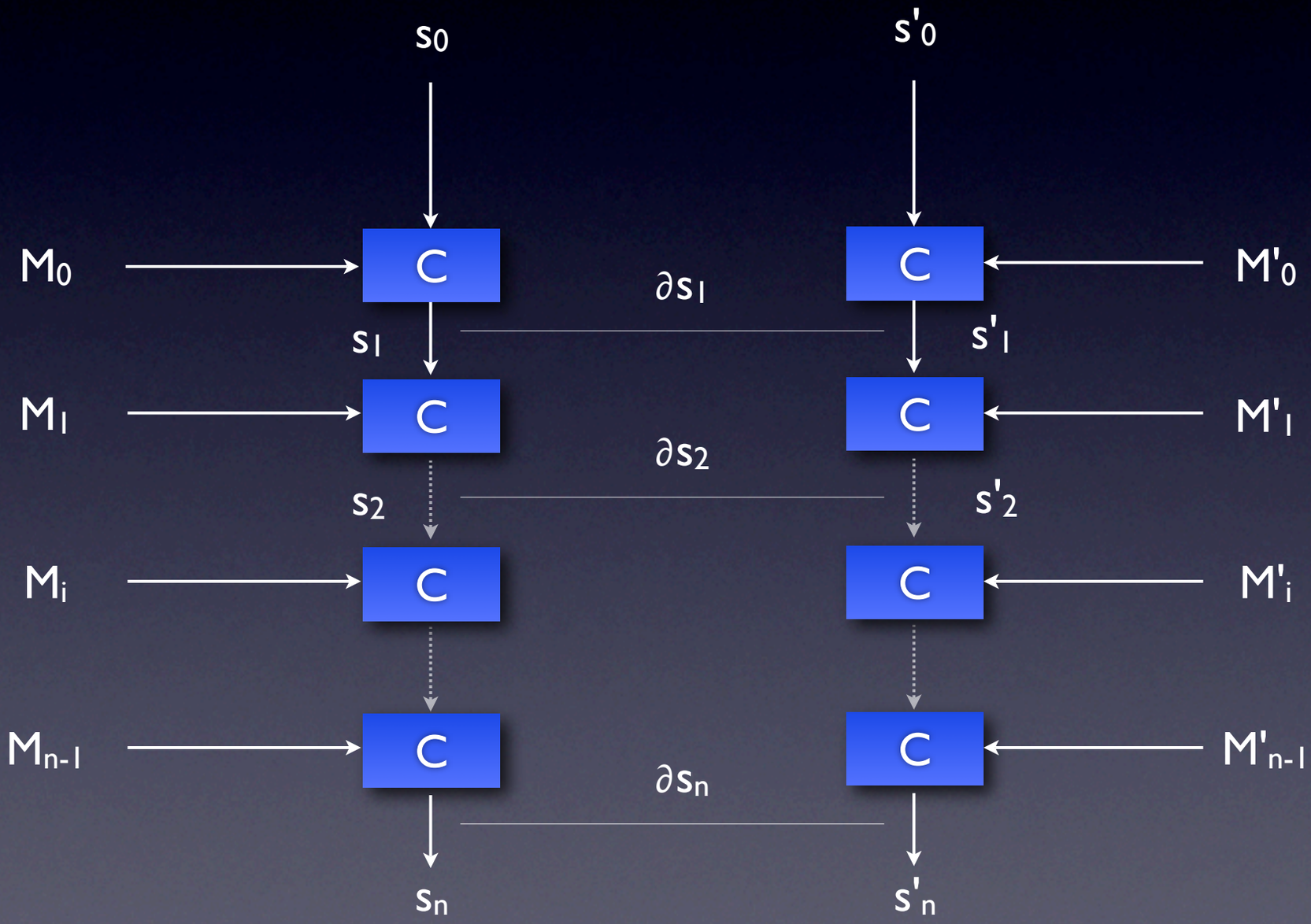
Round	Step i	r_i
1	1 to 16	7, 12, 17, 22, 7, 12, ...
2	17 to 32	5, 9, 14, 20, 5, 9, ...
3	33 to 48	4, 11, 16, 23, 4, 11, ...
4	49 to 64	6, 10, 15, 21, 6, 10, ...

Wang's Attack

Wang's Attack



Wang's Attack



- Let x' = x ⊞ ∂x.

Tracking Differences

- We seek to track how differences propagate between steps...
  - ...that is how bits differ between q_i and q'_i in each step.
- The differences are expressed additively as $\partial q_i = q'_i \boxminus q_i$.
- Also, the differences are expressed as binary signed digit representations Δq_i, which specify the bitwise difference.

$\Delta q_i [j]$	$q_i [j]$	$q'_i [j]$
.	0	0
.	1	1
+	0	1
-	1	0

Observations on the MD5 Round Functions

Observations on the MD5 Round Functions

Function	Absorbs	Flip 1	Flip 2	Flip 3
IF	Yes	Maybe	Maybe	Maybe
XOR	No	Yes	No	Yes
XNO	Yes	Maybe	Maybe	Maybe

- Flip N means that the function will change the output bit if N input bits are flipped.

Observations on the MD5 Round Functions

- The third round is critical, since one may not use the round function to absorb any single input bit differences.
- Select a message difference ∂M to handle the third round.

Selecting Message Differences

Selecting Message Differences

- If four consecutive ∂q values in the third round are set to 2^{31}
  - ... and no message words interfere further down ...
  - ... then all remaining ∂q values in the third round will be set to 2^{31}.

Proof by Induction

$\partial m_{\sigma(i)} = 0$ and $\partial q'_{i-1} = \partial q'_{i-2} = \partial q'_{i-3} = \partial q'_{i-4} = 2^{31}$ and $F_i = \mathrm{XOR}$

$$q_i = q_{i-1} \boxplus (q_{i-4} \boxplus F_i (q_{i-1}, q_{i-2}, q_{i-3}) \boxplus m_{\sigma(i)} \boxplus k_i) \lll r_i$$

$$q'_i = q'_{i-1} \boxplus (q'_{i-4} \boxplus F_i (q'_{i-1}, q'_{i-2}, q'_{i-3}) \boxplus m'_{\sigma(i)} \boxplus k_i) \lll r_i$$

$$q'_i = q'_{i-1} \boxplus (q'_{i-4} \boxplus \mathrm{XOR} (q_{i-1} \boxplus 2^{31}, q_{i-2} \boxplus 2^{31}, q_{i-3} \boxplus 2^{31}) \boxplus m_{\sigma(i)} \boxplus k_i) \lll r_i$$

$$q'_i = q'_{i-1} \boxplus (q'_{i-4} \boxplus (q_{i-1} \oplus 2^{31}) \oplus (q_{i-2} \oplus 2^{31}) \oplus (q_{i-3} \oplus 2^{31}) \boxplus m_{\sigma(i)} \boxplus k_i) \lll r_i$$

$$q'_i = q'_{i-1} \boxplus (q_{i-4} \boxplus 2^{31} \boxplus (q_{i-1} \oplus q_{i-2} \oplus q_{i-3}) \boxplus 2^{31} \boxplus m_{\sigma(i)} \boxplus k_i) \lll r_i$$

$$q'_i = q_{i-1} \boxplus 2^{31} \boxplus (q_{i-4} \boxplus (q_{i-1} \oplus q_{i-2} \oplus q_{i-3}) \boxplus m_{\sigma(i)} \boxplus k_i) \lll r_i$$

$$q'_i = q_i \boxplus 2^{31}$$

Selecting Message Differences

- **Assume** that the last four ∂q values in the second round are zero

$$\partial q_{29} = \partial q_{30} = \partial q_{31} = \partial q_{32} = 0$$

- **Pick** four message word differences so as to obtain

$$\partial q_i = \ldots = \partial q_{i+3} = 2^{31}$$

at some point in the third round.

Selecting Message Differences

$$\partial q_{i-1} = \partial q_{i-2} = \partial q_{i-3} = \partial q_{i-4} = 0$$

$$\partial q_i = \partial q_{i-1} \boxplus (\partial q_{i-4} \boxplus \partial F_i (q_{i-1}, q_{i-2}, q_{i-3}) \boxplus \partial m_{\sigma(i)}) \lll r_i$$

$$\partial q_i = \partial m_{\sigma(i)} \lll r_i = 2^{31} \qquad \Rightarrow \partial m_{\sigma(i)} = 2^{31 - r_i}$$

$$\partial q_{i+1} = \partial q_i \boxplus (\partial q_{i-3} \boxplus \partial F_i (q_i, q_{i-1}, q_{i-2}) \boxplus \partial m_{\sigma(i+1)}) \lll r_{i+1}$$

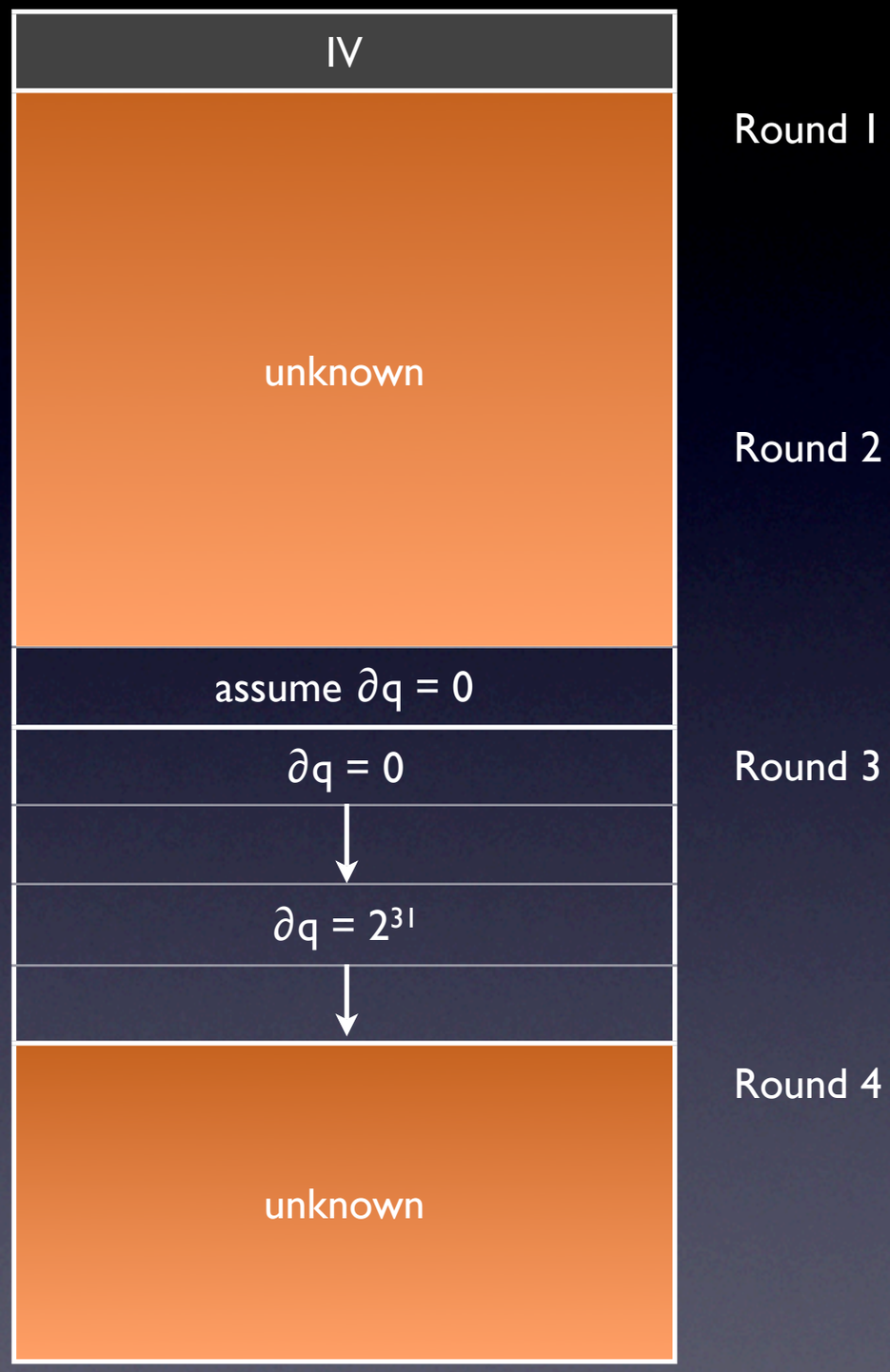
$$\partial q_{i+1} = 2^{31} \boxplus (2^{31} \boxplus \partial m_{\sigma(i+1)}) \lll r_{i+1} = 2^{31} \qquad \Rightarrow \partial m_{\sigma(i+1)} = 2^{31}$$

$$\partial q_{i+2} = \partial q_{i+1} \boxplus (\partial q_{i-2} \boxplus \partial F_i (q_{i+1}, q_i, q_{i-1}) \boxplus \partial m_{\sigma(i+2)}) \lll r_{i+2}$$

$$\partial q_{i+2} = 2^{31} \boxplus (\partial m_{\sigma(i+2)}) \lll r_{i+2} = 2^{31} \qquad \Rightarrow \partial m_{\sigma(i+2)} = 0$$

$$\partial q_{i+3} = \partial q_{i+2} \boxplus (\partial q_{i-1} \boxplus \partial F_i (q_{i+2}, q_{i+1}, q_i) \boxplus \partial m_{\sigma(i+3)}) \lll r_{i+3}$$

$$\partial q_{i+3} = 2^{31} \boxplus (2^{31} \boxplus \partial m_{\sigma(i+3)}) \lll r_{i+3} = 2^{31} \qquad \Rightarrow \partial m_{\sigma(i+3)} = 2^{31}$$


IV
unknown
assume ∂q = 0
∂q = 0
∂q = 2^31
unknown
Round 1
Round 2
Round 3
Round 4

Additional Constraints for the Fourth Round

- In the fourth round, it is often trivial to handle differences in the 31st bit but difficult to handle differences in other bits.
- Therefore, we need the difference that is in bit $31 - r_i$ to enter late into the fourth round.
  - This is the case for Wang's path.

Wang's Message Differences

Round 3

Step i	$\sigma(i)$	$\partial m_{\sigma(i)}$
33	5	0
34	8	0
35	11	2^{15}
36	14	2^{31}
37	1	0
38	4	2^{31}
39	7	0
40	10	0
41	13	0
42	0	0
43	3	0
44	6	0
45	9	0
46	12	0
47	15	0
48	2	0

Round 4

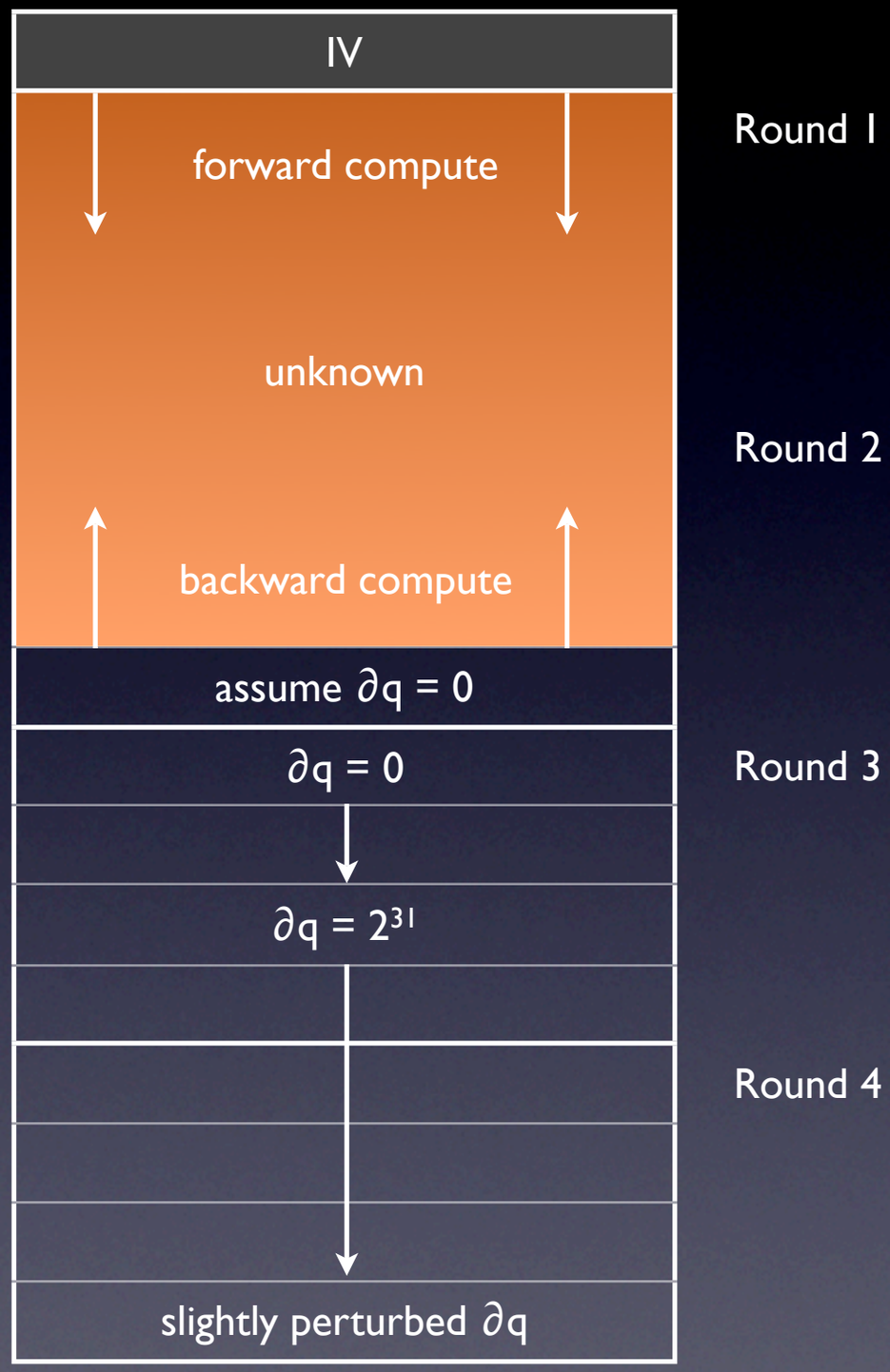
Step i	$\sigma(i)$	$\partial m_{\sigma(i)}$
49	0	0
50	7	0
51	14	2^{31}
52	5	0
53	12	0
54	3	0
55	10	0
56	1	0
57	8	0
58	15	0
59	6	0
60	13	0
61	4	2^{31}
62	11	2^{15}
63	2	0
64	9	0

Output Difference

- The output difference is given by

$$(2^{31},\ \ 2^{31} \boxplus 2^{25},\ \ 2^{31} \boxplus 2^{25},\ \ 2^{31} \boxplus 2^{25})$$

Forward & Backward Differential Derivation


IV
forward compute
unknown
backward compute
assume ∂q = 0
∂q = 0
∂q = 2^31
slightly perturbed ∂q
Round 1
Round 2
Round 3
Round 4

Forward Differential Derivation

- The round functions F_i are bitwise and depend on q_{i-1}, q_{i-2} and q_{i-3}.
  - If we know the binary signed digit representations Δq_{i-1}, Δq_{i-2} and Δq_{i-3}, then we know the possible values of ΔF_i.

∂F_i

- Consider an example in the first round where $F_i = IF(q_{i-1}, q_{i-2}, q_{i-3})$.

Δq_{i-3} = ...+. .....+. .....

Δq_{i-2} = ...−. .....+. .....

Δq_{i-1} = ...+. ....

ΔF_i = ...±. .....±. .....+. .....

variable difference fixed difference

∂F_i

- Consider an example in the first round where $F_i = IF(q_{i-1}, q_{i-2}, q_{i-3})$.

Δq_{i-3} = ...+. ......+. ....

Δq_{i-2} = ...−. ......+. ....

Δq_{i-1} = ...+. ....

ΔF_i = ...+. ......+. ....

variable difference fixed difference

∂F_i

- Consider an example in the first round where $F_i = IF(q_{i-1}, q_{i-2}, q_{i-3})$.

Δq_{i-3} = ...+. ......+. ....

Δq_{i-2} = ...^. ......-. ......+. ....

Δq_{i-1} = ...+. ......0. ....

ΔF_i = ...+. ......+. ....

variable difference fixed difference

- Select the desired ΔFi and impose additional conditions on the . bits.

Additional Differential Conditions

Symbol	Step i
.	$q_i = q'_i$
1	$q_i = q'_i = 1$
0	$q_i = q'_i = 0$
+	$q_i = 0$ and $q'_i = 1$
-	$q_i = 1$ and $q'_i = 0$
^	$q_i = q_{i-1}$ and $q'_i = q'_{i-1}$
!	$q_i \neq q_{i-1}$ and $q'_i \neq q'_{i-1}$

∂T_i and ∂q_i

- If we know ΔFi, we know which ∂T_i values are obtainable, since

$$\partial T_i = \partial q_{i-4} \boxplus \Delta F_i \boxplus \partial m_{\sigma(i)}$$

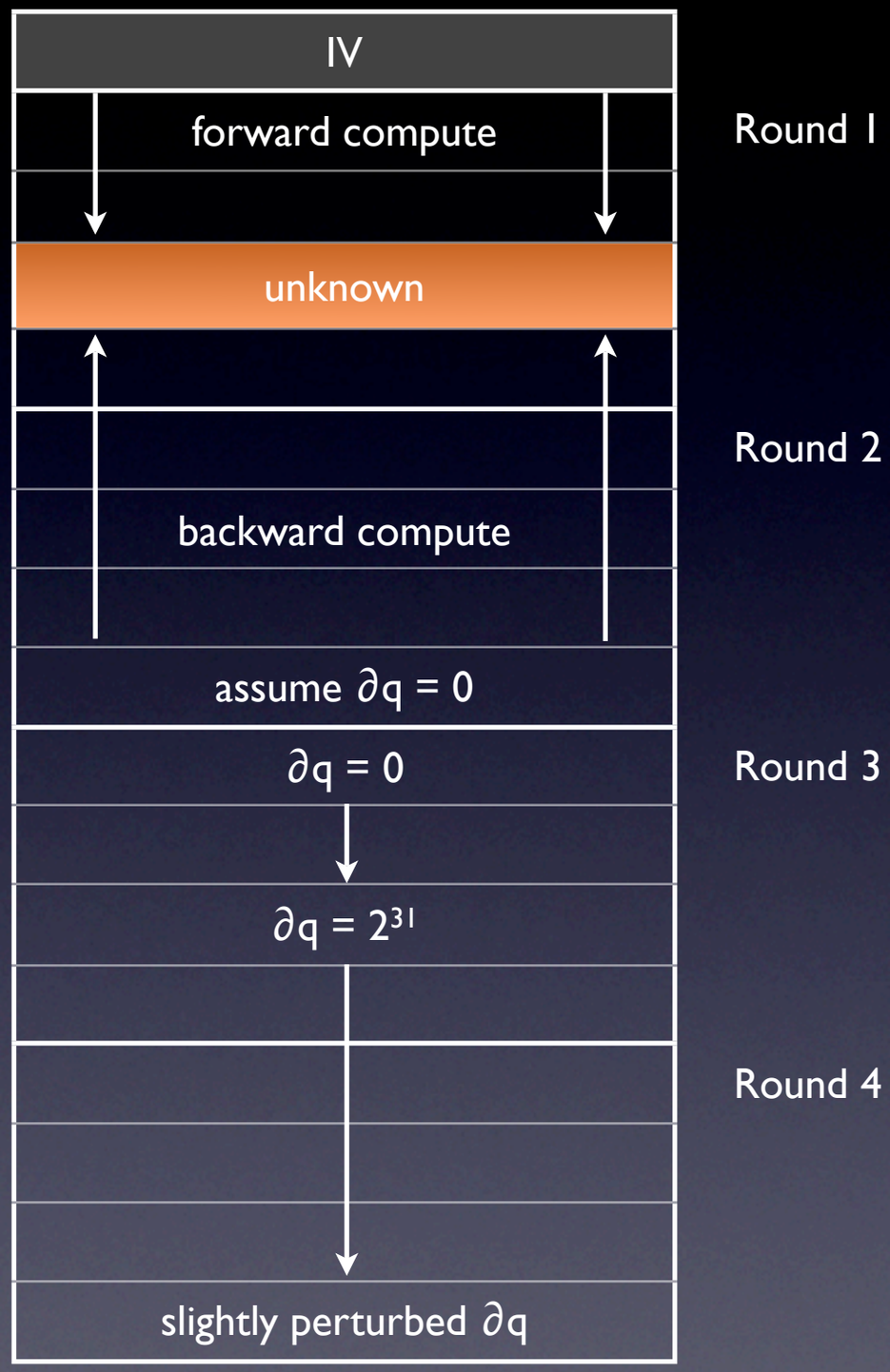
- Then we have at most four possible values of

$$\partial q_i = \partial qi{-}1 \boxplus (\partial T_i \lll r_i)$$

- Select a ∂q_i and select a BSDR representation Δq_i.
  - It is appropriate to minimize the number of set signed bits in Δq_i.

Backward Differential Derivation

- Analogous to forward differential derivation.


IV
forward compute
unknown
backward compute
assume ∂q = 0
∂q = 0
∂q = 2^31
slightly perturbed ∂q
Round 1
Round 2
Round 3
Round 4

Joining the Partial Paths

Joining the Paths

- We need to join the paths over four consecutive steps.
- That is, select a set of compatible BSDRs Δq_k, Δq_{k+1}, Δq_{k+2}, Δq_{k+3}
- These BSDRs may have a lot of set signed bits.
- It is trivial to respect conditions in the first round.
  - Select k such that $k + 3 \leq 16$

Joining Column-wise

- Estimate or "guess" values of ∂q_k to ∂q_{k+3}
- The paths may be joined column-wise from step k – 4 to k + 7.
  - The most computationally intense step.
  - May fail, in which case new partial paths must be selected.

The Second Block

The Second Block

- Negate the message word differences.
- Proceed in the same way as for the first block.
- In general, the output from encrypting the second block is then the negation of the encryption of the first block.
  - When these are added in the Davies-Meyer scheme, we obtain a two block collision!

Collision Search

Collision Search

- Find a message M which follows the differential path.
  - Use single-message modification in the first round.
  - Compute rounds 2 to 4 and verify each step.

Single Message Modification

- In the first round, there exists a bijection between $m_{\sigma(i)}$ and q_i provided that q_{i-4} to q_{i-1} have been fixed.

$$q_i = q_{i-1} \boxplus (q_{i-4} \boxplus F_i (q_{i-1}, q_{i-2}, q_{i-3}) \boxplus m_{\sigma(i)} \boxplus k_i) \lll r_i$$

$$\downarrow$$

$$m_{\sigma(i)} = ((q_i \boxminus q_{i-1}) \ggg r_i) \boxminus q_{i-4} \boxminus F_i (q_{i-1}, q_{i-2}, q_{i-3}) \boxminus k_i$$

- For i = 1, .., 16, simply select q_i by randomizing the . bits.
  - Set the other bits to respect conditions such as ^, !, 0, 1, etc.
  - Compute the message word m_{i-1} using the formula above.

Verification

- Compute the step function for steps 17 up to 64 and verify that the sought BSDRs are indeed obtained in each step.
- Every condition after step 16 increases the complexity.
  - If there are n conditions after step 16, complexity $O(2^n)$
  - Wang's first path has 37 such conditions, giving $O(2^{37})$.
  - This is why we joined the paths in the first round.

Complexity Analysis

- In the second blocks, there are differences in the IV.
- Some of the bits in the IV will be fixed by the F_i functions when the differential path for the second block is constructed.
- If there are m additional conditions on the IV then the complexity will increase with a factor 2^m since we need to find $\approx 2^m$ messages that pass the first path before we can start with the second block.
  - Total complexity $O(2^{m+n})$ for the first block.

Optimizations

Tunnels

- Vlastimil Klíma introduced the concept of tunnels in March of 2006.
- Tunnels provide a means of varying the message words slightly without recomputing all steps in rounds 2.
- Using tunnels reduces the search complexity.

Tunnels

- As an example, consider steps 9 to 13 in the first round.

$$q_9 = q_8 \boxplus (q_5 \boxplus IF(q_8, q_7, q_6) \boxplus m_8 \boxplus k_9) \lll r_9$$
$$q_{10} = q_9 \boxplus (q_6 \boxplus IF(q_9, q_8, q_7) \boxplus m_9 \boxplus k_{10}) \lll r_{10}$$
$$q_{11} = q_{10} \boxplus (q_7 \boxplus IF(q_{10}, q_9, q_8) \boxplus m_{10} \boxplus k_{11}) \lll r_{11}$$
$$q_{12} = q_{11} \boxplus (q_8 \boxplus IF(q_{11}, q_{10}, q_9) \boxplus m_{11} \boxplus k_{12}) \lll r_{12}$$
$$q_{13} = q_{12} \boxplus (q_9 \boxplus IF(q_{12}, q_{11}, q_{10}) \boxplus m_{12} \boxplus k_{13}) \lll r_{13}$$

Tunnels

- As an example, consider steps 9 to 13 in the first round.

$$q_9 = q_8 \boxplus (q_5 \boxplus IF(q_8, q_7, q_6) \boxplus m_8 \boxplus k_9) \lll r_9$$
$$q_{10} = q_9 \boxplus (q_6 \boxplus IF(q_9, q_8, q_7) \boxplus m_9 \boxplus k_{10}) \lll r_{10}$$
$$q_{11} = q_{10} \boxplus (q_7 \boxplus IF(q_{10}, q_9, q_8) \boxplus m_{10} \boxplus k_{11}) \lll r_{11}$$
$$q_{12} = q_{11} \boxplus (q_8 \boxplus IF(q_{11}, q_{10}, q_9) \boxplus m_{11} \boxplus k_{12}) \lll r_{12}$$
$$q_{13} = q_{12} \boxplus (q_9 \boxplus IF(q_{12}, q_{11}, q_{10}) \boxplus m_{12} \boxplus k_{13}) \lll r_{13}$$

- We seek to vary q_9 whilst keeping q_{10} to q_{13} constant.
  - Only vary bits in q_9 for which $q_{10} = 0$ and $q_{11} = 1$.

Tunnels

- As an example, consider steps 9 to 13 in the first round.

$$q_9 = q_8 \boxplus (q_5 \boxplus IF(q_8, q_7, q_6) \boxplus m_8 \boxplus k_9) \lll r_9$$
$$q_{10} = q_9 \boxplus (q_6 \boxplus IF(q_9, q_8, q_7) \boxplus m_9 \boxplus k_{10}) \lll r_{10}$$
$$q_{11} = q_{10} \boxplus (q_7 \boxplus q_8 \boxplus m_{10} \boxplus k_{11}) \lll r_{11}$$
$$q_{12} = q_{11} \boxplus (q_8 \boxplus q_{10} \boxplus m_{11} \boxplus k_{12}) \lll r_{12}$$
$$q_{13} = q_{12} \boxplus (q_9 \boxplus IF(q_{12}, q_{11}, q_{10}) \boxplus m_{12} \boxplus k_{13}) \lll r_{13}$$

- We seek to vary q_9 whilst keeping q_{10} to q_{13} constant.
  - Only vary bits in q_9 for which $q_{10} = 0$ and $q_{11} = 1$.
  - We then only have to recompute m_8, m_9 and m_{12}.

Tunnels

- In the second round, m_9 is first to appear, in step 25.

$$q_{17} = q_{16} \boxplus (q_{13} \boxplus IF(q_{14}, q_{16}, q_{15}) \boxplus m_1 \boxplus k_{17}) \lll r_{17}$$
$$q_{18} = q_{17} \boxplus (q_{14} \boxplus IF(q_{15}, q_{17}, q_{16}) \boxplus m_6 \boxplus k_{18}) \lll r_{18}$$
$$q_{19} = q_{18} \boxplus (q_{15} \boxplus IF(q_{16}, q_{18}, q_{17}) \boxplus m_{11} \boxplus k_{19}) \lll r_{19}$$
$$q_{20} = q_{19} \boxplus (q_{16} \boxplus IF(q_{17}, q_{19}, q_{18}) \boxplus m_0 \boxplus k_{20}) \lll r_{20}$$
$$q_{21} = q_{20} \boxplus (q_{17} \boxplus IF(q_{18}, q_{20}, q_{19}) \boxplus m_5 \boxplus k_{21}) \lll r_{21}$$
$$q_{22} = q_{21} \boxplus (q_{18} \boxplus IF(q_{19}, q_{21}, q_{20}) \boxplus m_{10} \boxplus k_{22}) \lll r_{22}$$
$$q_{23} = q_{22} \boxplus (q_{19} \boxplus IF(q_{20}, q_{22}, q_{21}) \boxplus m_{15} \boxplus k_{23}) \lll r_{23}$$
$$q_{24} = q_{23} \boxplus (q_{20} \boxplus IF(q_{21}, q_{23}, q_{22}) \boxplus m_4 \boxplus k_{24}) \lll r_{24}$$
$$q_{25} = q_{24} \boxplus (q_{21} \boxplus IF(q_{22}, q_{24}, q_{23}) \boxplus m_9 \boxplus k_{25}) \lll r_{25}$$

- If we had not used the tunnel, we would have had to begin at step 19.
- There are a lot of conditions in steps 17 to 24 that may be skipped.

Tunnels

- There are more tunnels in MD5 which may be combined.
  - This reduces the search complexity considerably.
- Tunnels exist in some other hash functions as well.

Demo & Questions

Thanks!