Table 11.2 The MD4 family of hash functions

Algorithm		Output [bit]	Input [bit]	No. of rounds	Collisions found
MD5		128	512	64	yes
SHA-1		160	512	80	not yet
SHA-2	**SHA-224**	224	512	64	no
	SHA-256	256	512	64	no
	SHA-384	384	1024	80	no
	SHA-512	512	1024	80	no

preimage and second preimage resistance are required. For such applications, MD5 is still sufficient.

11.3.2 Hash Functions from Block Ciphers

Hash functions can also be constructed using block cipher chaining techniques. As in the case of dedicated hash functions like SHA-1, we divide the message x into blocks x_i of a fixed size. Figure 11.6 shows a construction of such a hash function: The message chunks x_i are encrypted with a block cipher e of block size b. As m-bit key input to the cipher, we use a mapping g from the previous output H_{i-1}, which is a b-to-m-bit mapping. In the case of $b = m$, which is, for instance, given if AES with a 128-bit key is being used, the function g can be the identity mapping. After the encryption of the message block x_i, we XOR the result to the original message block. The last output value computed is the hash of the whole message x_1,x_2,...,x_n, i.e., $H_n = h(x)$.

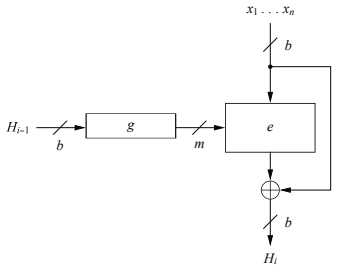

Fig. 11.6 The Matyas–Meyer–Oseas hash function construction from block ciphers

The function can be expressed as:

$$H_i = e_{g(H_{i-1})}(x_i) \oplus x_i$$

This construction, which is named after its inventors, is called the Matyas–Meyer–Oseas hash function.

There exist several other variants of block cipher based realizations of hash functions. Two popular ones are shown in Figure 11.7.

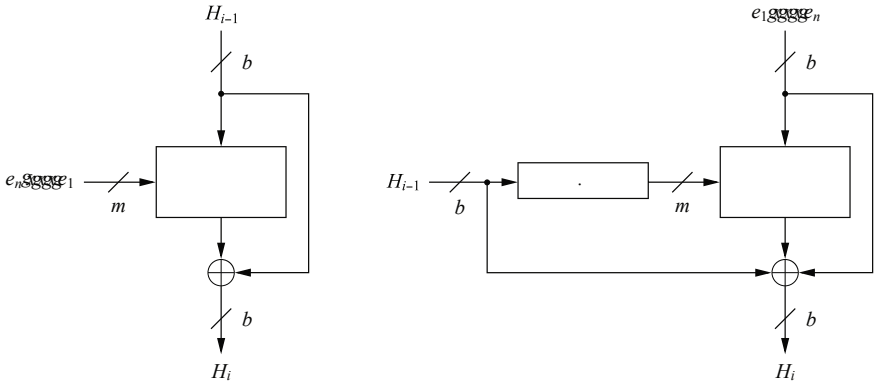

Fig. 11.7 Davies–Meyer (left) and Miyaguchi–Preneel hash function constructions from block ciphers

The expressions for the two hash functions are:

$$H_i = H_{i-1} \oplus e_{x_i}(H_{i-1}) \qquad \text{(Davies–Meyer)}$$
$$H_i = H_{i-1} \oplus x_i \oplus e_{g(H_{i-1})}(x_i) \qquad \text{(Miyaguchi–Preneel)}$$

All three hash functions need to have initial values assigned to H_0. These can be public values, e.g., the all-zero vector. All schemes have in common that the bit size of the hash output is equal to the block width of the cipher used. In situations where only preimage and second preimage resistance is required, block ciphers like AES with 128-bit block width can be used, because they provide a security level of 128 bit against those attacks. For application which require collision resistance, the 128-bit length provided by most modern block ciphers is not sufficient. The birthday attack reduces the security level to mere 64 bit, which is a computational complexity that is within reach of PC clusters and certainly is doable for attackers with large budgets.

One solution to this problem is to use Rijndael with a block width of 192 or 256 bit. These bit lengths provide a security level of 96 and 128 bit, respectively, against birthday attacks, which is sufficient for most applications. We recall from Section 4.1 that Rijndael is the cipher that became AES but allows block sizes of 128, 192 and 256 bit.

Another way of obtaining larger message digests is to use constructions which are composed of several instances of a block cipher and which yield twice the width of the block length b. Figure 11.8 shows such a construction for the case that a cipher e is being employed whose key length is twice the block length. This is in particular the case for AES with a 256-bit key. The message digest output are the $2b$ bit $(H_{n,L}||H_{n,R})$. If AES is being used, this output is $2b = 256$ bit long, which provides a high level of security against collision attacks. As can be seen from the figure, the previous output of the left cipher $H_{i-1,L}$ is fed back as input to both block

ciphers. The concatenation of the previous output of the right cipher, $H_{i-1,R}$, with the next message block x_i, forms the key for both ciphers. For security reasons a constant c has to be XORed to the input of the right block cipher. c can have any value other than the all-zero vector. As in the other three constructions described above, initial values have to be assigned to the first hash values ($H_{0,L}$ and $H_{0,R}$).

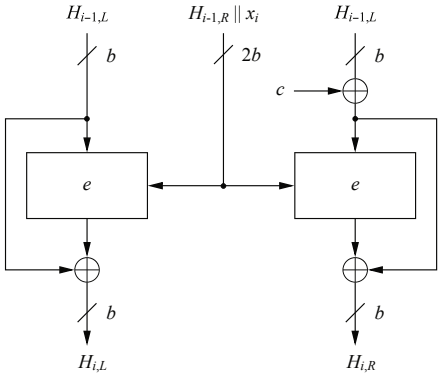

Fig. 11.8 Hirose construction for a hash function with twice the block width

We introduce here the Hirose construction for the case that the key length be twice the block width. There are many other ciphers that satisfy this condition in addition to AES, e.g., the block ciphers Blowfish, Mars, RC6 and Serpent. If a hash function for resource-constrained applications is needed, the lightweight block cipher PRESENT (cf. Section 3.7) allows an extremely compact hardware implementation. With a key size of 128-bit and a block size of 64 bit, the construction computes a 128-bit hash output. This message digest size resists preimage and second preimage attacks, but offers only marginal security against birthday attacks.

11.4 The Secure Hash Algorithm SHA-1

The Secure Hash Algorithm (SHA-1) is the most widely used message digest function of the MD4 family. Even though new attacks have been proposed against the algorithm, it is very instructive to look at its details because the stronger versions in the SHA-2 family show a very similar internal structure. SHA-1 is based on a Merkle–Damgård construction, as can be seen in Figure 11.9.

An interesting interpretation of the SHA-1 algorithm is that the compression function works like a block cipher, where the input is the previous hash value H_{i-1} and the key is formed by the message block x_i. As we will see below, the actual rounds of SHA-1 are in fact quite similar to a Feistel block cipher.

SHA-1 produces a 160-bit output of a message with a maximum length of 2^{64} bit. Before the hash computation, the algorithm has to preprocess the message. During the actual computation, the compression function processes the message in 512-bit