Diffie Hellman Key Exchange

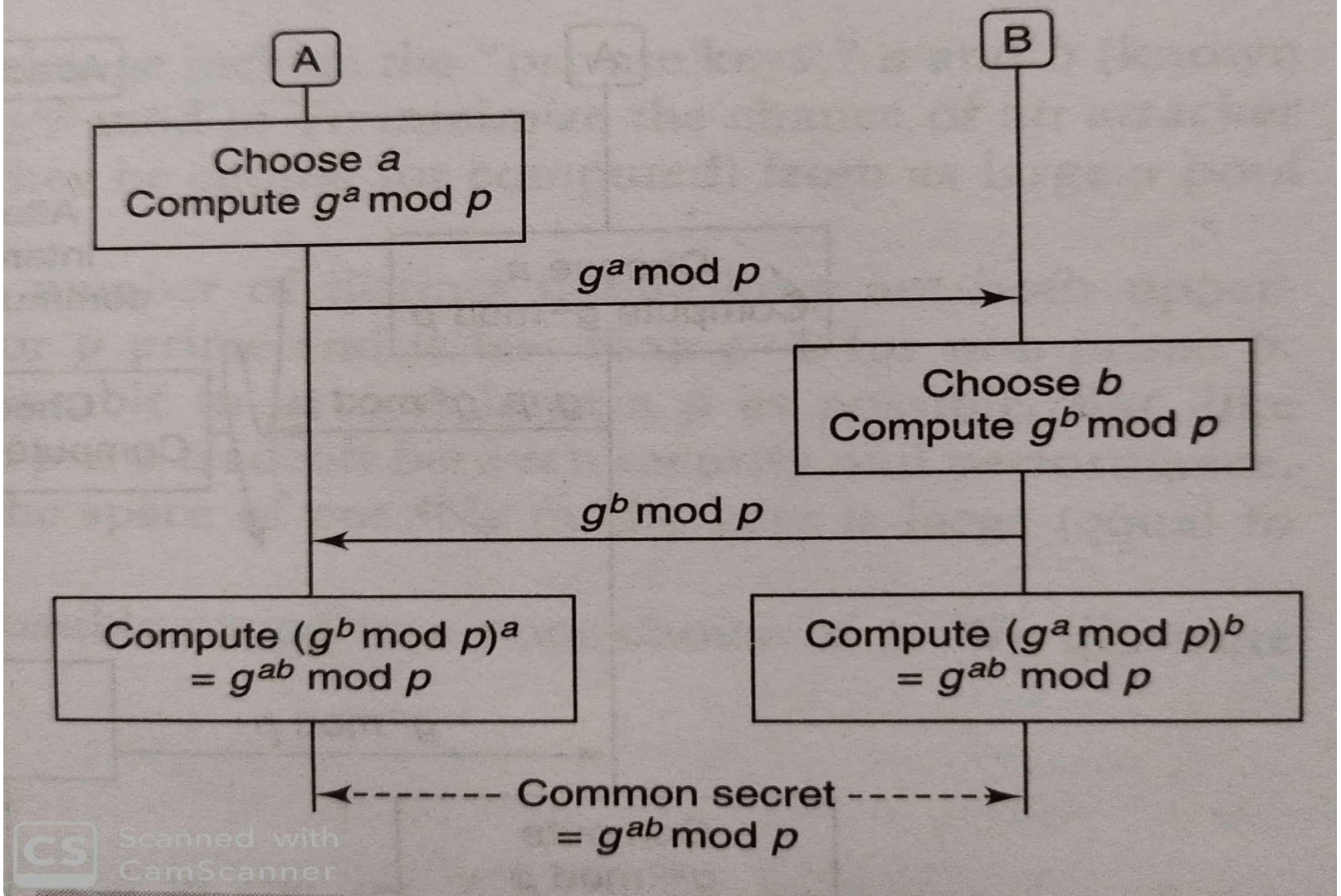
The **Diffie–Hellman key exchange** method allows two parties that have no prior knowledge of each other to jointly establish a shared secret **key** over an insecure channel (over Internet).This **key** can then be used to encrypt subsequent communications using a symmetric **key** cipher.

Working of Algorithm:

Consider two parties, ‘A’ and ‘B’ that need to agree upon a single shared key for the duration of their current session. Both ‘A’ and ‘B’ will be knowing about a common modulus ‘p’ and the generator ‘g’ of the selected modulus.

In order to exchange the shared secret, both will participate in the following sequence of steps

User A	User B
Selects Public Keys: **p, g**	Selects Public Keys: **p, g**
Private key selected is: **a**	Private key selected is: **b**
Public key generated: $x = g^a mod\ p$	Public key generated: $y = g^b mod\ p$
Exchange generated public keys	
Key received: y	Key received: x
Generates shared secret: $k_A = y^a mod\ p$	Generates shared secret: $k_B = x^b mod\ p$
Algebraically it can be shown that these 2 keys are equal and the same	
$(g^b mod\ p)^a mod\ p$	$(g^a mod\ p)^b mod\ p$
Above equation can also be written as $(g^a mod\ p)^b mod\ p$	$(g^a mod\ p)^b mod\ p$
Both the derived keys are found to be equal. Ie, $k_A = k_b$	
Now both the users 'A' and 'B' can use the shared secret key for encrypting the messages during the current session.	



Example Problem and solution

Let $p = 131$ and $g = 2$.

Let the random number chosen by A be 24. So, her partial key is $2^{24} \bmod 131 \equiv 46$.

Let the random number chosen by B be 17. So, his partial key is $2^{17} \bmod 131 \equiv 72$.

After receiving B's partial key, A computes the shared secret as $72^{24} \bmod 131 \equiv 13$.

After receiving A's partial key, B computes the shared secret as $46^{17} \bmod 131 \equiv 13$.

Questions:

1. **Explain Diffie Hellman key exchange protocol.**
2. **Compute shared secret between users A and B when, p=13, a=7, b=5, g=2**

Authentication

- Authentication techniques are used to verify identity. Message authentication verifies the authenticity of both the message content and the message sender.
- A common technique for authenticating a message is to implement a hash function, which is used to produce a "fingerprint" of a message.
- The hash value is added at the end of message before transmission.
- The receiver recomputes the hash value from the received message and compares it to the received hash value.
- If the two hash values are the same, the message was not altered during transmission.
- Once a hash function is applied on a message, m, the result is known as a message digest, or h(m).

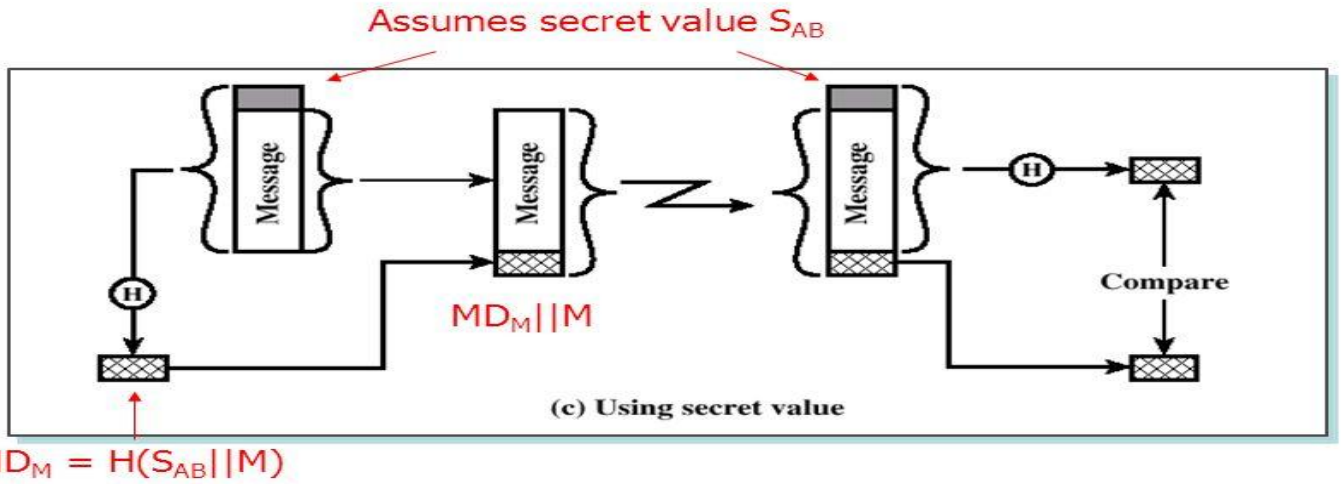
Hash function has the following properties.

- Unlike the encryption algorithm, the authentication algorithm is not required to be reversible.
- Given a message digest h(m), it is computationally infeasible to find m.
- It is computationally infeasible to find two different messages m1 and m2 such that h(m1) = h(m2).

- **Message authentication can be implemented by two methods.**
  1. Use of the hash function
  2. Hash combined with encryption

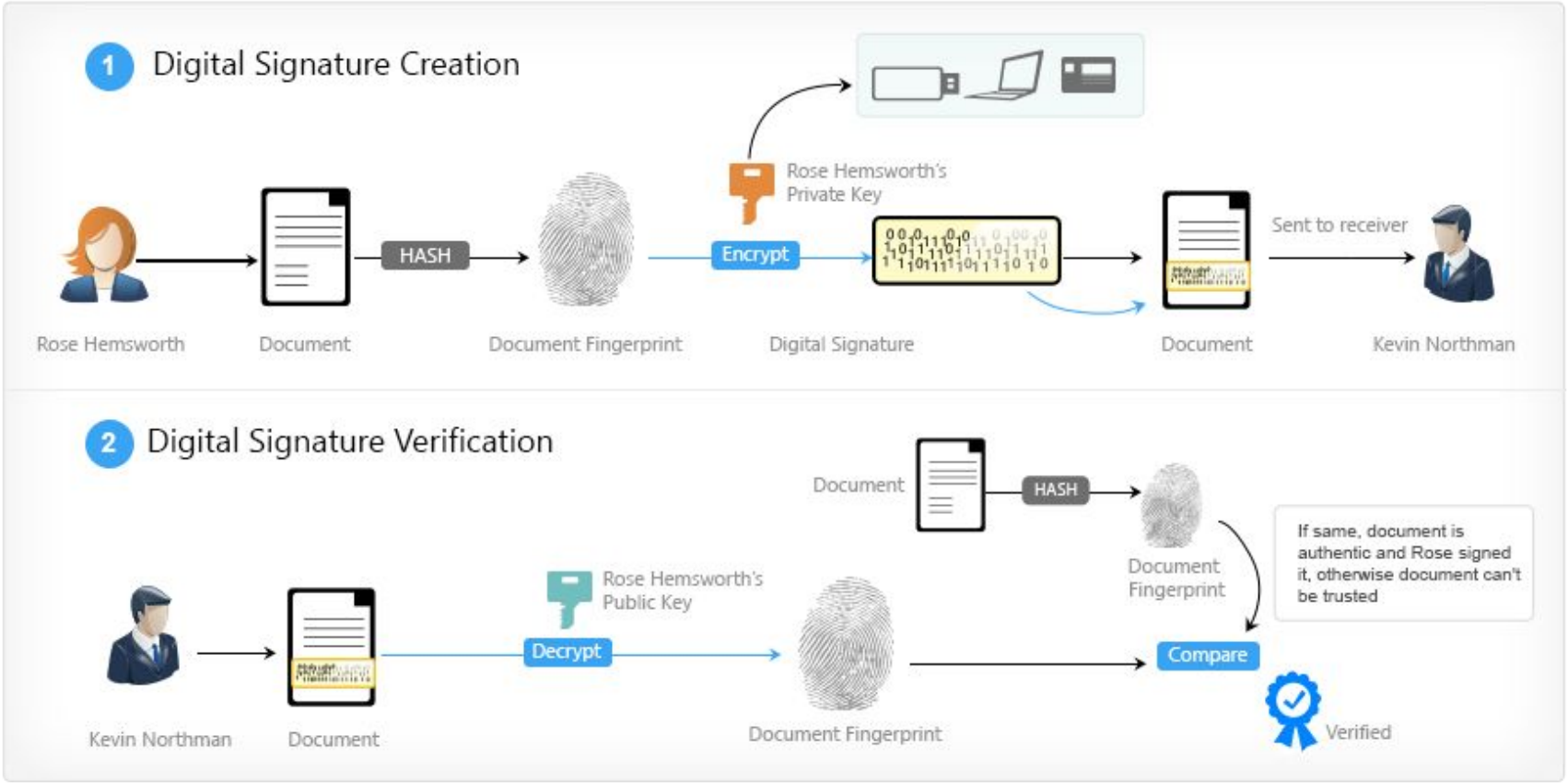
1. Use of the hash function

One Way Hash Function



No encryption for message authentication
Secret value never sent; can't modify the message
Important technique for **Digital Signatures**

2. Hash combined with encryption (Knows as Digital Signature)



Digital Signature

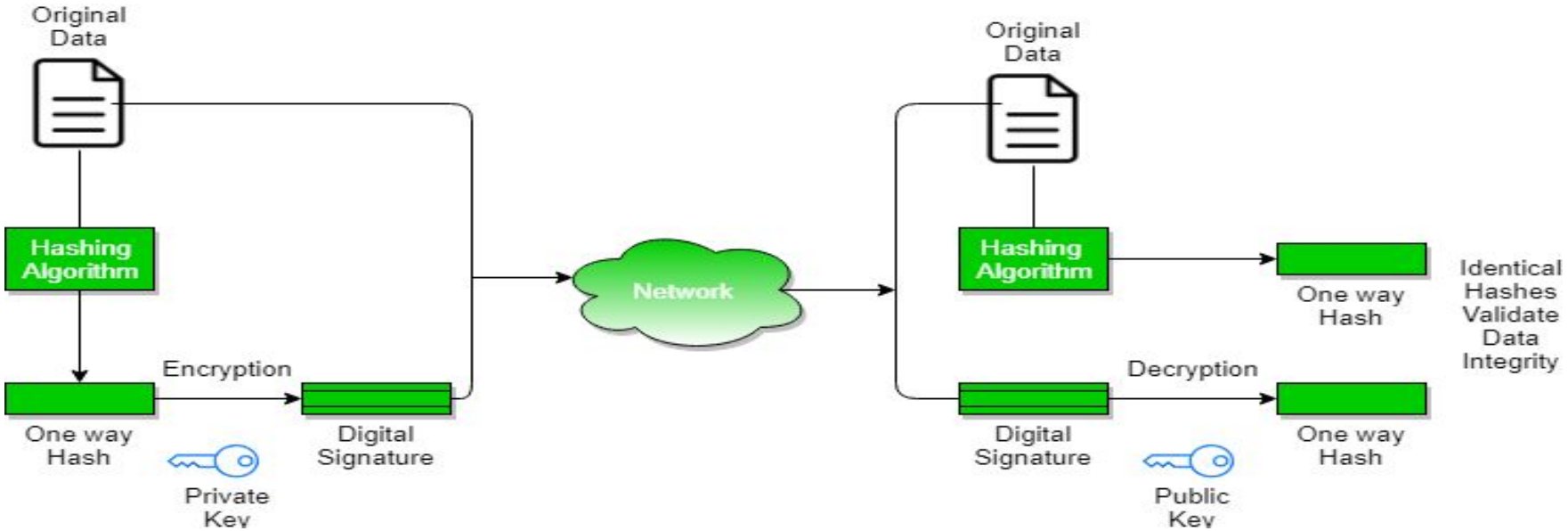
Question: Write a short note on Digital Signature

A digital signature is a mathematical technique used to validate the authenticity and integrity of a message, software or digital document.

The steps followed in creating digital signature are :

1. Message digest is computed by applying hash function on the message and then message digest is encrypted using private key of sender to form the digital signature.
2. Digital signature is then transmitted with the message.(message + digital signature is transmitted)
3. Receiver decrypts the digital signature using the public key of sender.(This assures authenticity,as only sender has his private key so only sender can encrypt using his private key which can thus be decrypted by sender's public key).
4. The receiver now has the message digest.

5. The receiver can compute the message digest from the message (actual message is sent with the digital signature).
6. The message digest computed by receiver and the message digest (got by decryption on digital signature) need to be same for ensuring integrity.



Steps involved in Digital Signature Generation

Secure Hash Function (SHA-1)

SHA is a cryptographic hash function. Uses a iterative hash construction.

Question: Explain the working of SHA-1 algorithm

- Input message is divided into smaller blocks of 512 bit size for processing. (ie, 512 bit block has part of the message + length of the message + required number of padded zero bits)
- Each block is processed in 80 rounds to get the hash code
- Computation of hash code (Hash digest) is carried out in two steps

1. Array Initialization
2. Computation of Hash code (Hash Digest)

Array Initialization:

- Each 512 bit block is further divided into 16 words of 32 bits.
- We need to generate total 80 words of 32 bits to process in 80 rounds. We have got 16 words from the message block (ie, 512/32=16). So, remaining words from 17th to 80th needs to be generated applying following operation on first 16 words of the message block

$$w_i = w_{i-3} \oplus w_{i-8} \oplus w_{i-14} \oplus w_{i-16} \; for \; 16 \le i \le 80$$

Hash Computation:

- A 160 bit block having 5 shift registers of 32 bit size and the shift registers, named as S1, S2, S3, S4 and S5, is used to store intermediate result
- These registers are initialized with a predetermined value at the beginning.
- Contents of the registers are processed in each round along the input message as given in the following figure.

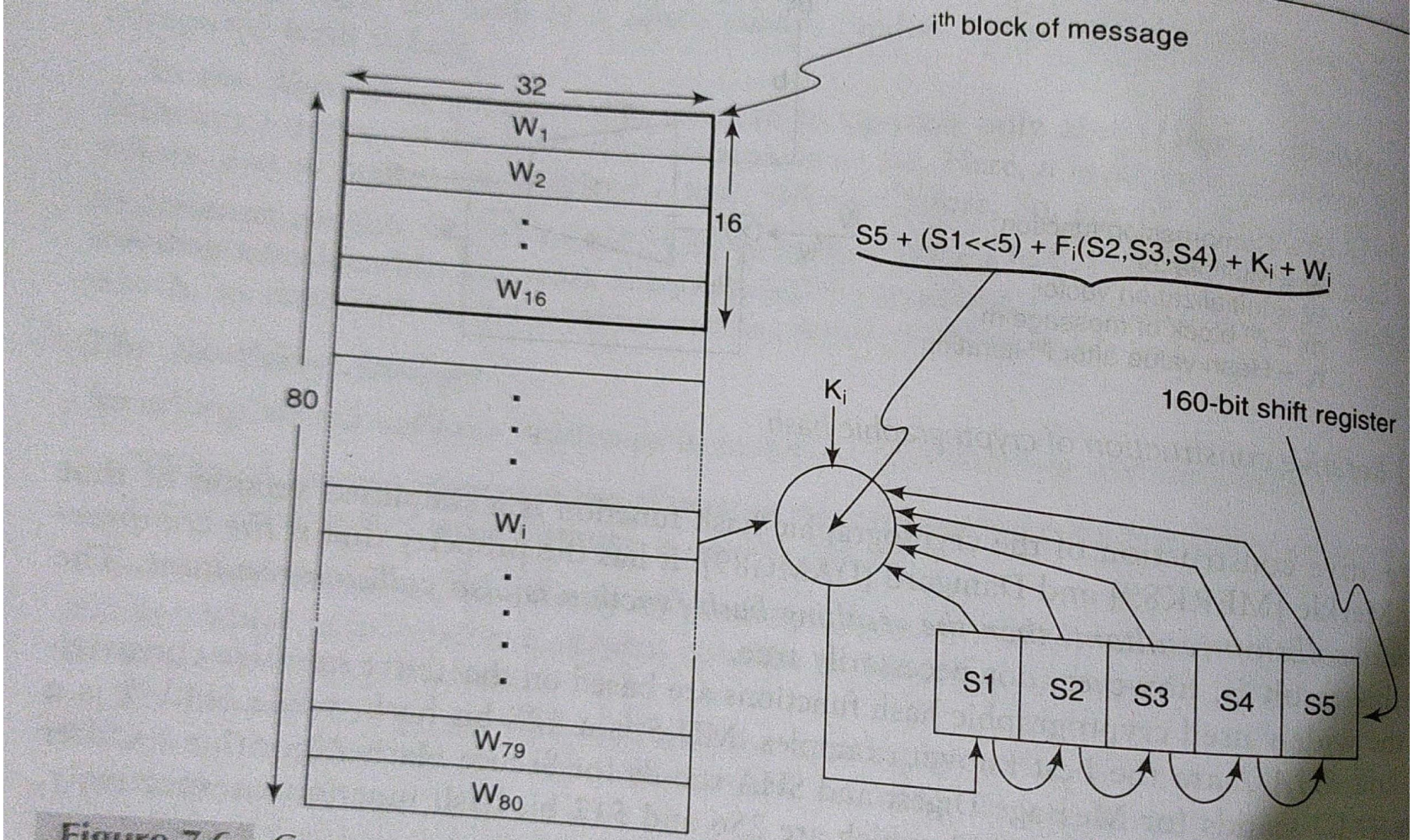


Figure 7.6 *Computation of SHA-1*

- Each word 0f 32 bit is processed applying following operation

$$s5+(s1 \ll 5)+F_i(s2,s3,s4)+k_i+w_i$$

- Processed data is stored in the shift registers as following

$$temp=s5+(s1 \ll 5)+F_i(s2,s3,s4)+k_i+w_i$$

s5=s4

s4=s3

s3=s2>>2

s2=s1

S1=temp

The function $F_i[s2,s3,s4]$ remains same for every 20 rounds and it works as follows.

$F_i[s2,s3,s4]=[s2 \wedge s3] \vee [s2 \wedge s4]; for 1 \leq i \leq 20$ (for first 20 rounds)

$F_i[s2,s3,s4]=s2 \oplus s3 \oplus s4; for 21 \leq i \leq 40$ (For rounds 21 to 40)

$F_i[s2,s3,s4]=[s2 \wedge s3] \vee [s2 \wedge s4] \vee [s3 \wedge s4]; for 41 \leq i \leq 60$ (for rounds 41 to 60)

$F_i[s2,s3,s4]=s2 \oplus s3 \oplus s4; for 61 \leq i \leq 80$ (For rounds 61 to 80)

At the end of 80th round, the contents stored in shift registers s1,s2,s3,s4 and s5 gives the 160 bits hash code.

Question:

1. Explain how to generate hash digest using SHA-1? [10M]***