462 – Intro to Cryptography
Topic 09
Hashing

Paar,Pelzl – Chapter 11

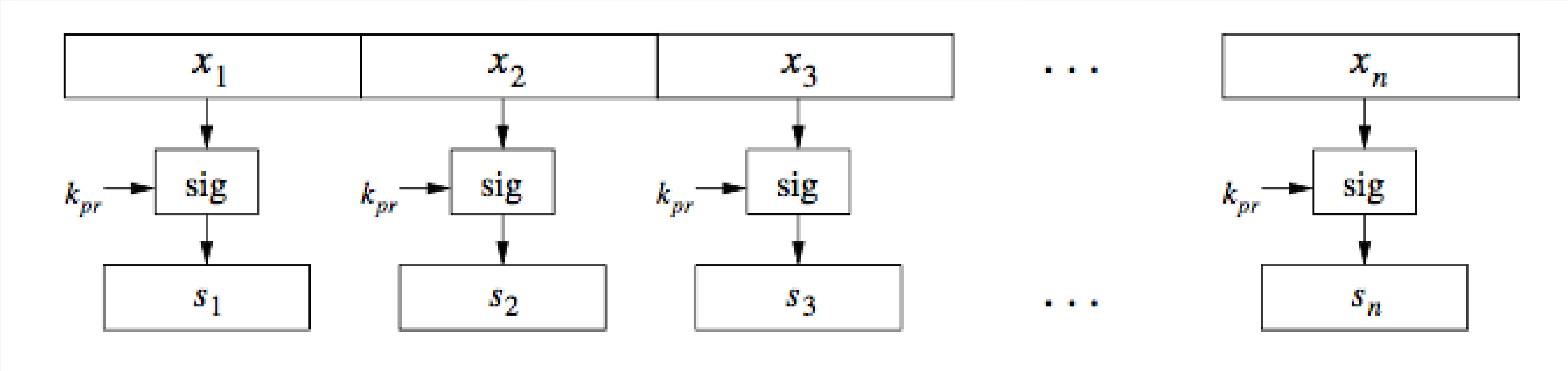
T.J. Borrelli

Hashing – main uses

- Hash and sign – digital signatures
- Message Authentication Codes (MACs)
- Digital fingerprint – data integrity
- Hashing can be used for (lossy) data compression
- PRNG

Motivation

Problem: Naive signing of long messages generates a signature of the same length!



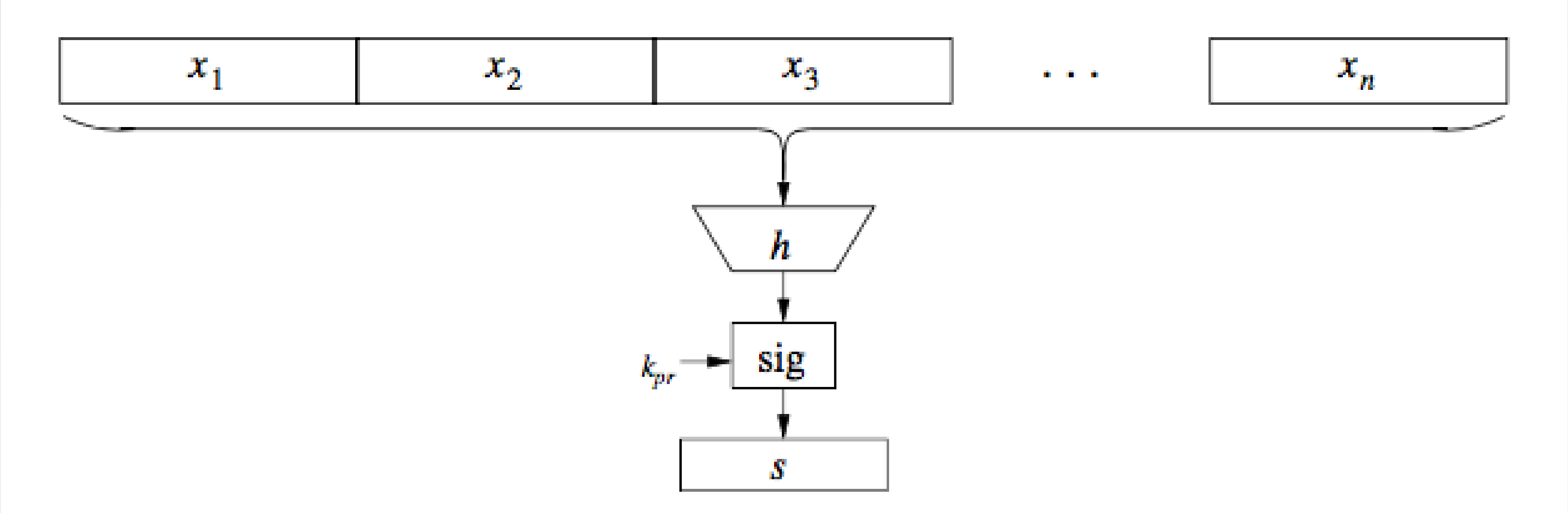
Three problems:

- Computational load/overhead – take a lot of effort if messages are long
- Message overhead – takes too much space (2x message)
- Security limitations – Oscar can remove/rearrange individual messages

Solution

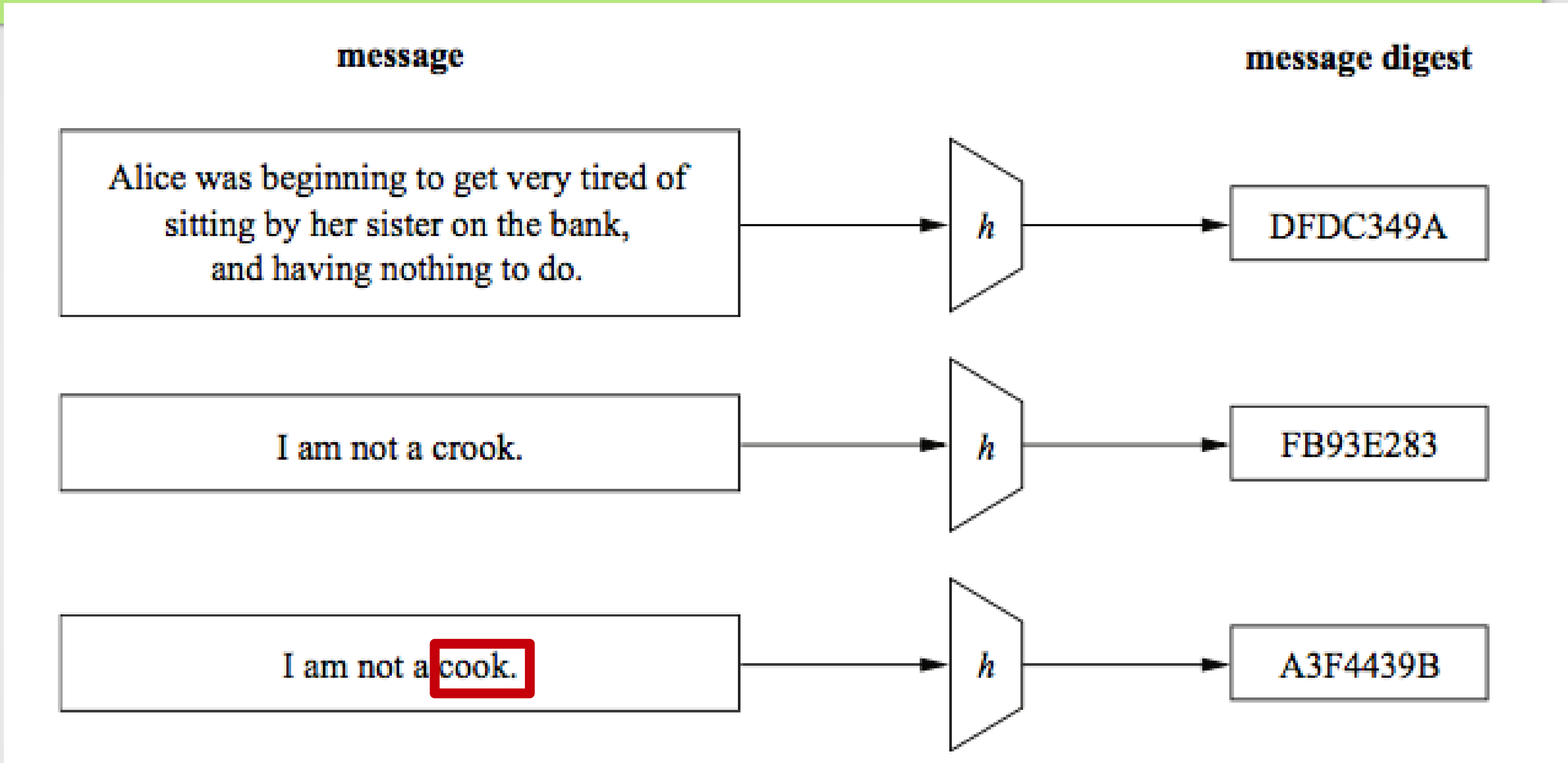
- Instead of signing the whole message, sign only the message digest – fingerprint of the message
- Secure and faster
- Need hash functions for this

Digital Signature with a Hash function



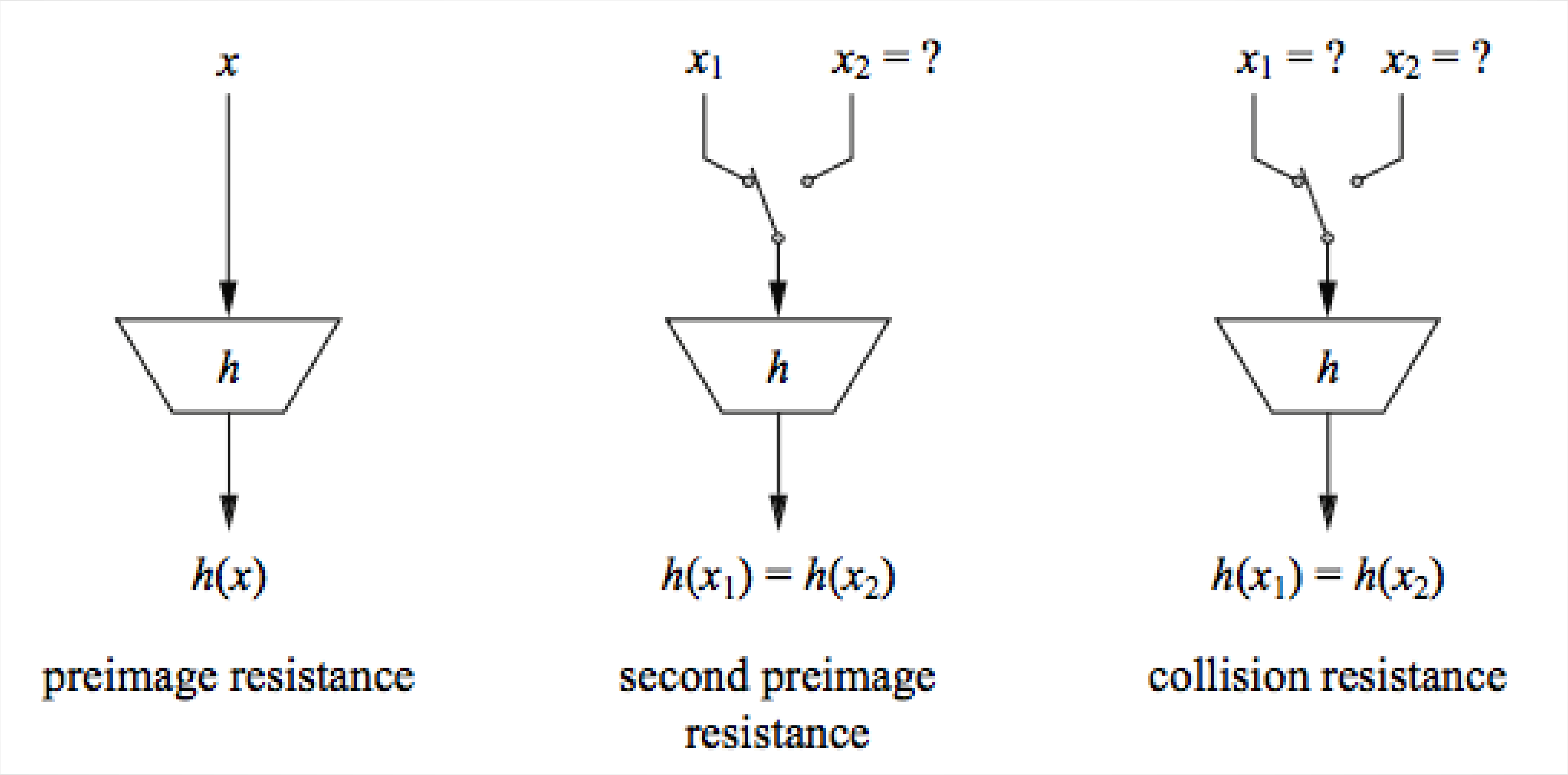
- x has variable length
- h has a fixed length
- Hash function h(x) does not require a key
- h(x) is public

Principal input-output behavior of hash function

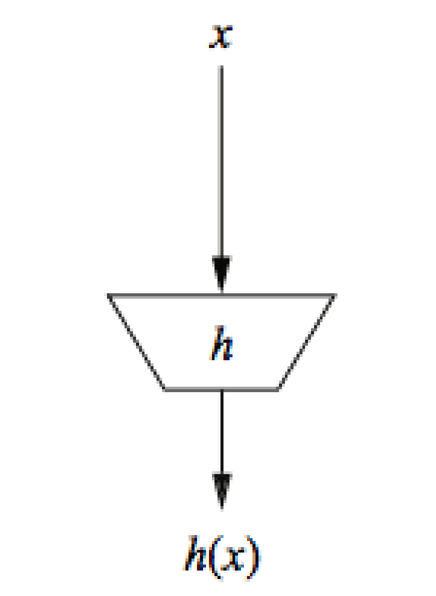


- Can apply hash function to message x of any size
- Hash function h(x) must be highly efficient
- Fixed length output
- The hash should be highly sensitive to all input bits

The three security properties of hash functions



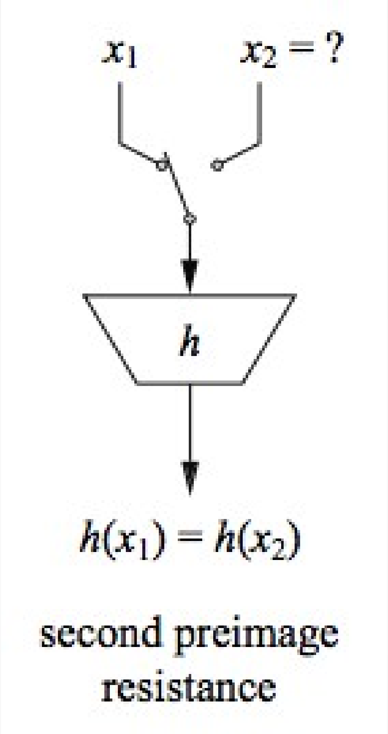
Hash function: security properties



preimage resistance

Preimage resistance (aka one-wayness) – for a given output z, it should be impossible to find any input x such that h(x) = z

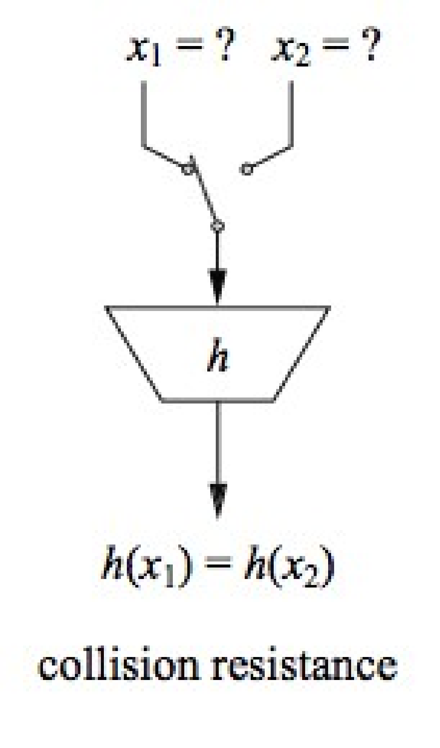
Hash function: security properties



Second preimage resistance (aka weak collision resistance)

– Given x_1, and computing $h(x_1)$ it is computationally infeasible to find any x_2 such that $h(x_1) = h(x_2)$.

Hash function: security properties



Collision resistance (aka *strong* collision resistance) – It is computationally infeasible to find any pairs $x_1 \neq x_2$ such that $h(x_1) = h(x_2)$.

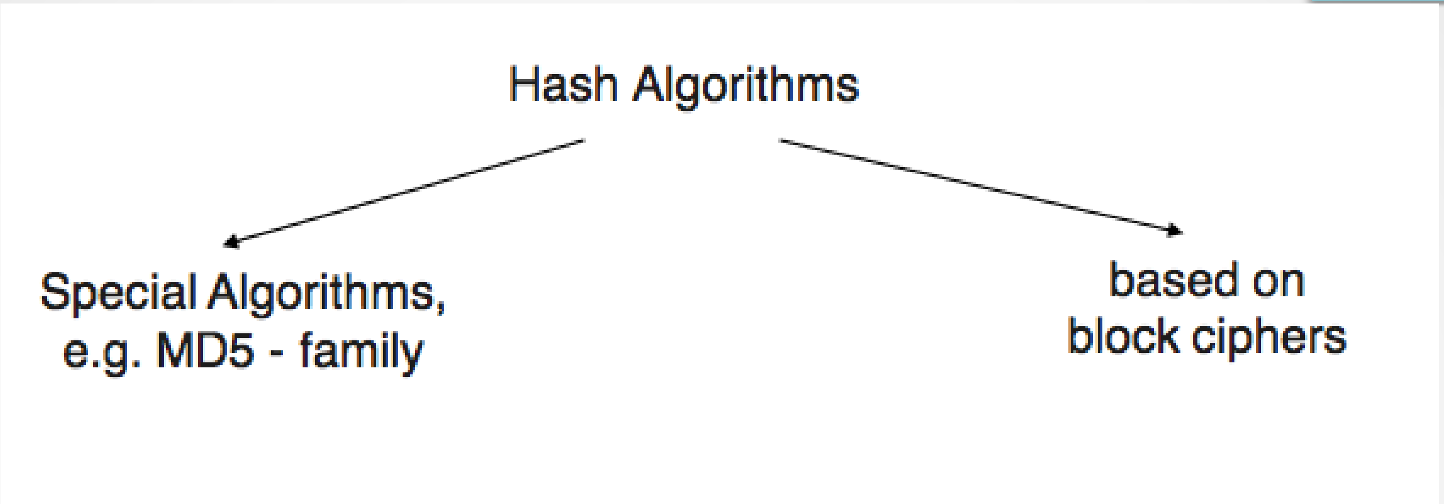
Hash function: security

- It turns out that collision resistance causes the most problems
- How hard is it to find a collision with probability .5?
- Related problem: how many people are needed such that two of them have the same birthday with probability of .5?

Hash function: security

- Many people think it’s about 365/2 = 183, but NO! It’s just 23 people!
- Since we are concerned not with the probability that someone shares a birthday with YOU but with the probability that someone shares a birthday with someone (anyone) else.
- This surprising result is called the “birthday paradox” – search takes approx $\sqrt{2^n} = 2^{n/2}$.
- To deal with this, hash functions need an output size of at least 160 bits.

Hash Functions: Algorithms



- MD4 family – includes MD5, SHA
- SHA-1 output – 160 bit; input 512 bit chucks of message x; operations – bitwise AND,OR, XOR, complement, cyclic shift

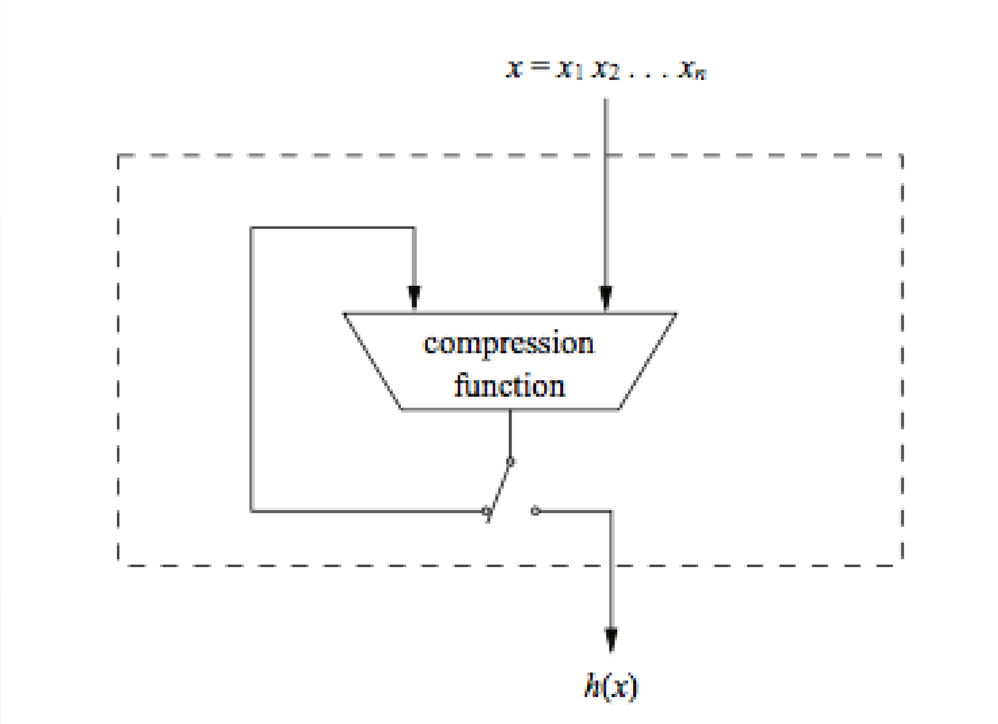
MD4 family of hash functions

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

Found in 2017!

SHA-1

- Part of the MD4 family
- Based on a Merkle-Damgård construction – hash value of the input message is defined as the output of the last iteration of the compression function

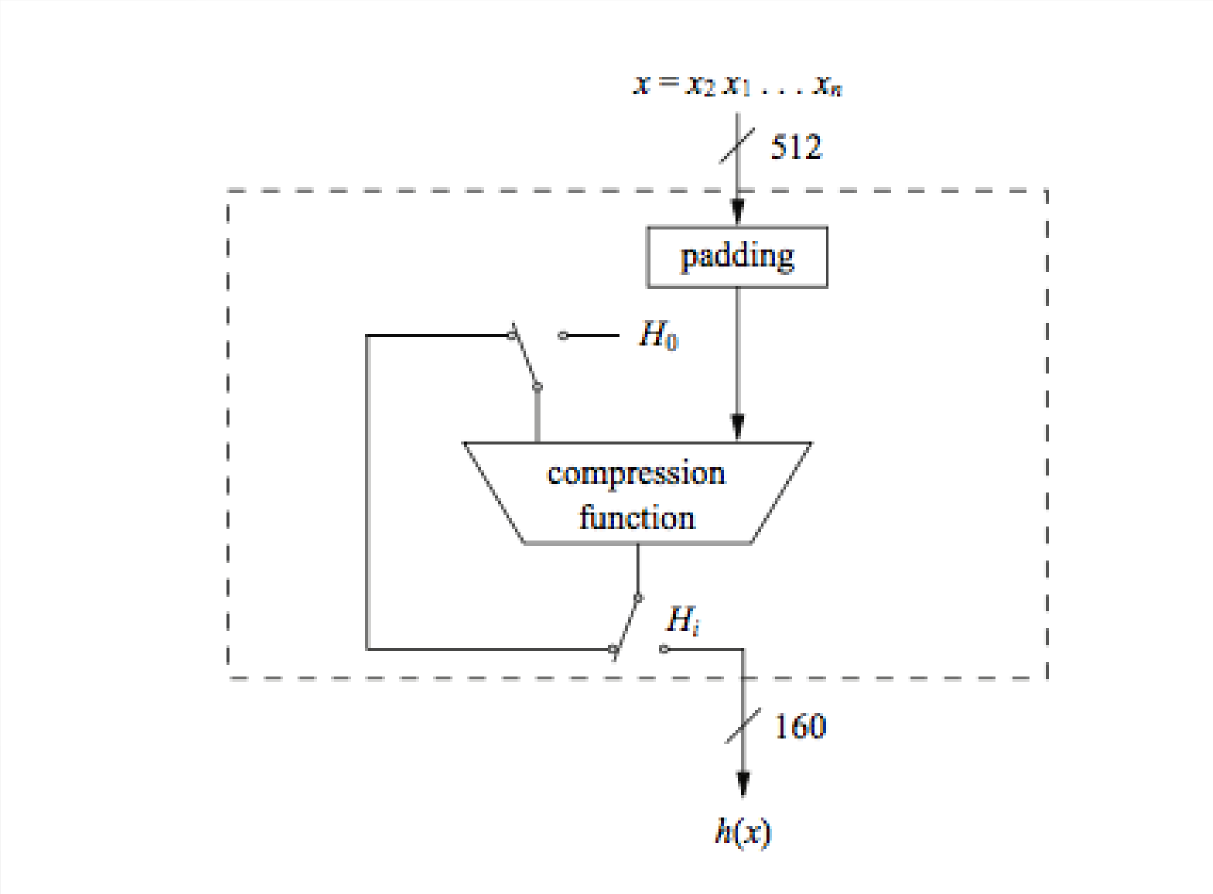


SHA-1

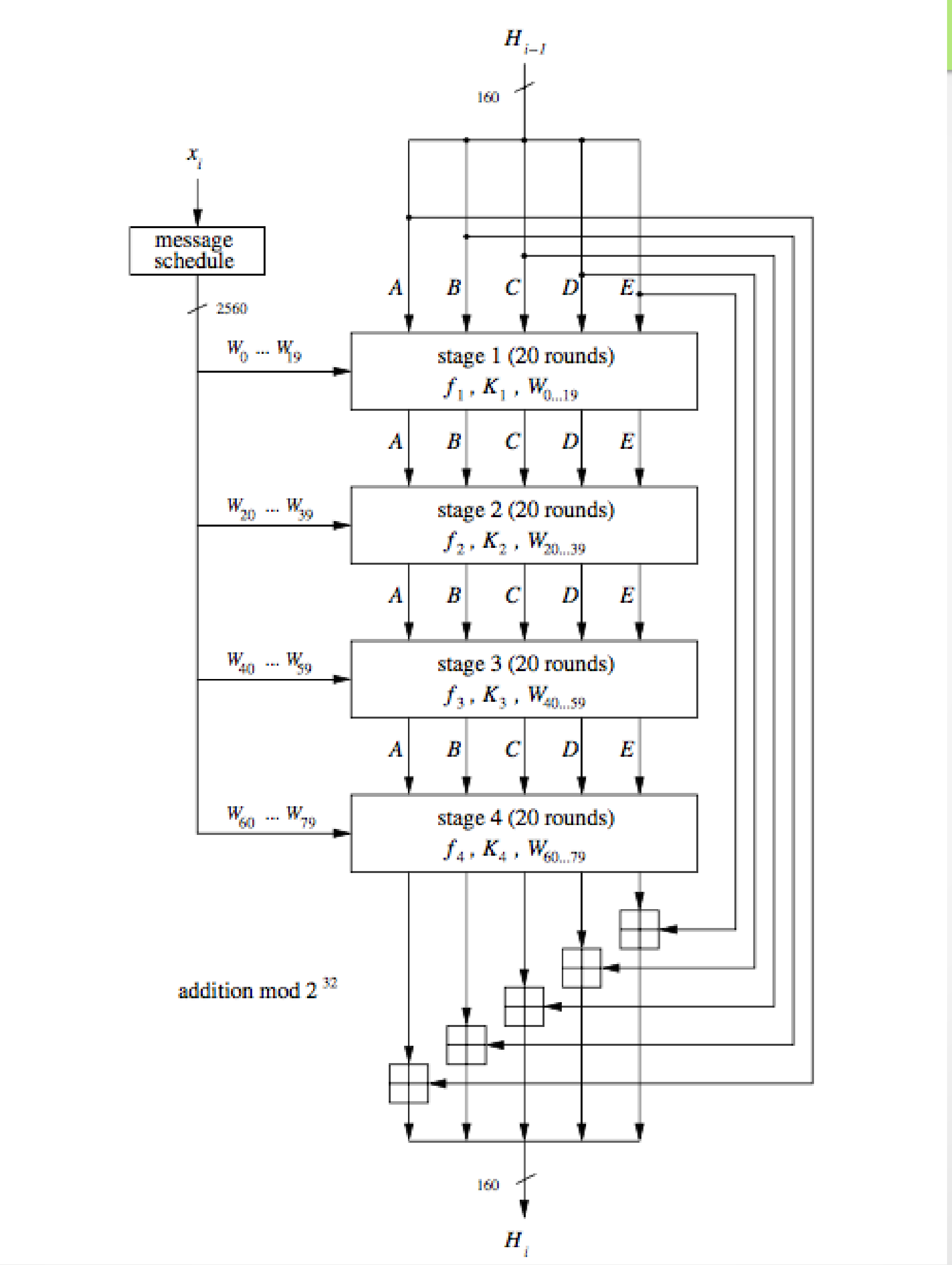
- 160-bit output from a message of maximum length 2^{64} bit
- Widely used, even though some weaknesses are known (collisions)
- Compression function consists of 80 rounds divided into 4 stages of 20 rounds each

High-level diagram of SHA-1

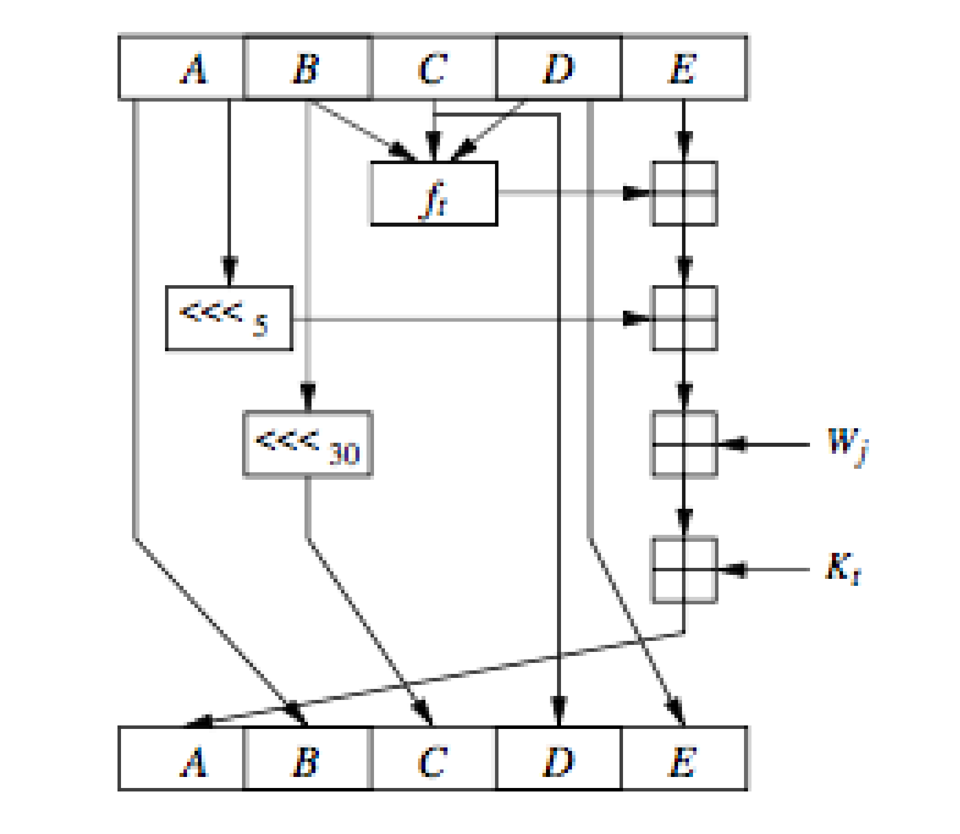
- Compression function consists of 80 rounds divided into 4 stages of 20 rounds each



Eighty-round compression function of SHA-1



Round j of stage t in SHA-1



The operation within round j of stage it is given by:

$$A,B,C,D,E = (E + f_t(B,C,D) + (A)_{\lll 5} + W_j + K_t), A, (B)_{\lll 30}, C, D$$

Summary of Lessons Learned

- Hash functions are keyless
- The two most important applications of has functions are their use in digital signatures and in message authentication
- The three security requirements for hash functions are preimage resistance (one-wayness), second preimage resistance (weak collision resistance) and (strong) collision resistance.
- Hash functions should have at least 160-bit output length in order to withstand collision attacks; 256 bit or more is desirable for long-term security
- MD5, once widely used, now insecure. Serious weaknesses found in SHA-1
- SHA-3 – Keccak is the new standard – release by NIST in 2015