Cpts 515. 10/26/2020

Breaking MD5.

HW3: Hint.

Prob2. Use ILP.

$$X_1 := X_1 + X_2;$$

$\Rightarrow$ $X_1' = X_1 + X_2 \wedge X_2' = X_2 \wedge X_3' = X_3 \wedge \ldots$

Combine all states in the pogram into a bigger ILP instance (rename variables).

Prob 3. Bell number (wiki)

Probs 4 & 5: easy.

Breaking MD5 ⟶ Wang, Xiaoyun (2005) et. al.

MD5 digest alg, by Rivest, is a function

$$\underbrace{x\,x\,\cdots\,x}_{512\text{ bits}} \xrightarrow{f} \underbrace{x\,x\,\cdots\,x}_{128\text{ bits}}$$

It's a hash function without randomness (initial vals are fixed).

$$\boxed{\text{Book}} \xrightarrow{\text{MD5}} 128\text{ bits}.$$

I expect: ① MD5(Book1) ≠ MD5(Book2)

$\Rightarrow$ Book ≠ Book2

(*) MD5(Book1) = MD5(Book2)

⟹ Book1 = Book2 with high probability,

close to 1.

Collision: { Book1 ≠ Book2 but

MD5(Book1) = MD5(Book2).

"Breaking MD5" ⟹ within reasonable resources, one can find two instances Book1, Book2 s.t.

A message is a seq of <u>blocks</u> (each block is of 512 bits)

$M_0, M_1, M_2, \ldots, M_{t-1}$.

MD5 runs as a chain:

$$\begin{tikzcd}
 & H_0 \arrow[r, "f"'] & H_1 \arrow[r, "f"'] & H_2 & \cdots & H_{t-1} \arrow[r, "f"'] & H_t \\
\text{initial value} \arrow[ur, bend left] & M_0 \arrow[ur] & M_1 \arrow[ur] & & & M_{t-1} \arrow[ur] & \text{Hash value of the entire seq of blocks (message)} \arrow[u]
\end{tikzcd}$$

Each step

$$f(H_i, M_i) = H_{i+1}$$

where M_i = 512 bits

H_i = 128 bits

H_{i+1} = 128 bits.

Each M_i is cut into 16 32-bit words:

$$M_i = \underbrace{m_0}_{32 \text{ bits}} \underbrace{m_1}_{32 \text{ bits}} \cdots \underbrace{m_{15}}_{32 \text{ bits}}$$

f is designed to perform 4 rounds, each round has 16 operations.

Wiki (MD5) ⟶ for source code.
(simple code).

Each op ~~tra~~ involves op. on 4 varables, each wth 32 bits, a, b, c, d.

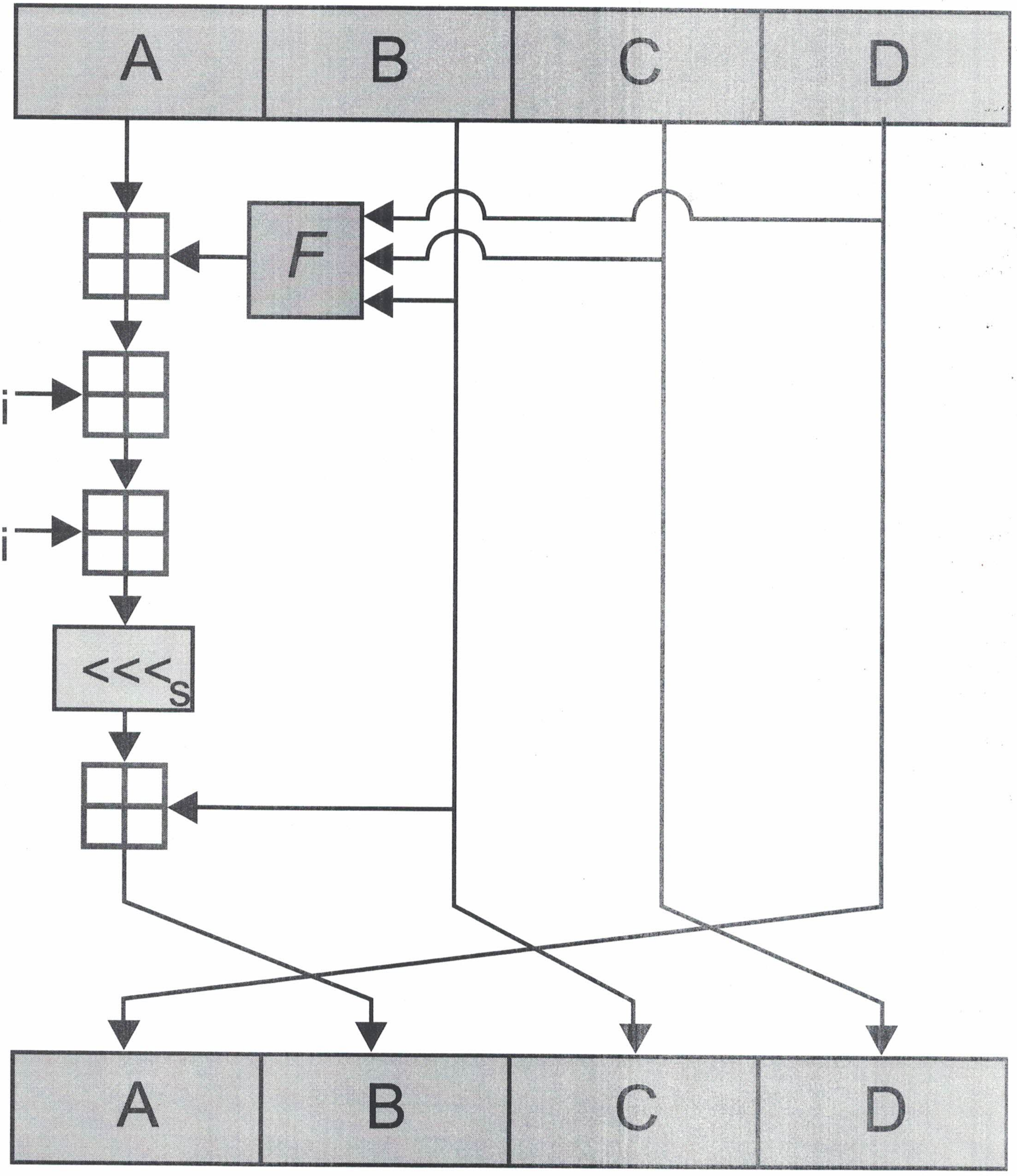
Each op will update these four varables using shift, m_j (from block M_i), and one of the following operations:

$(X, Y, Z) \Longrightarrow (X \wedge Y) \vee (\neg X \wedge Z)$

$(X, Y, Z) \Longrightarrow (X \wedge Z) \vee (Y \wedge \neg Z)$

$(X, Y, Z) \Longrightarrow (X \oplus Y \oplus Z)$

$(X, Y, Z) \Rightarrow (Y \oplus (X \vee \neg Z))$.



Source: Wiki(MD5)

Breaking MD5 ≡ find M_0, M_1 and M_0', M_1' such that

$$\begin{tikzcd}
{} \arrow[r, "\text{initial}"] & H_0 \arrow[r] & H_1 \arrow[r] & H_2 \\
& M_0 \arrow[ur, "f"] & M_1 \arrow[ur, "f"] &
\end{tikzcd}$$

$$\begin{tikzcd}
H_0 \arrow[r] & H_1 \arrow[r] & H_2 \\
M_0' \arrow[ur, "f"] & M_1' \arrow[ur, "f"] &
\end{tikzcd}$$

MD5, by design, is believed to be one-way.

However, There is a way to break:

Weakness:

① those "nolinear" functions.

↳ MD5 should have used "multiply", However, "multiply" is not efficient.

the chosen artificial functions are not perfectly nonlinear.

---

② apprach.

$$\Delta H_0 \xrightarrow{(M_0, M_0')} \Delta H_1 \xrightarrow{(M_1, M_1')} \Delta H_2 = 0.$$

$\Delta H_0 = 0$ (initial differences are the same)

collision found.

Wang's approach.

$$\begin{tikzcd}
\Delta H_0 \arrow[r, "P_1^1"] \arrow[d, equal] & \Delta R_{1,1} \arrow[r, "P_2^1"] & \Delta R_{1,2} \arrow[r, "P_3^1"] & \Delta R_{1,3} \arrow[r, "P_4^1"] & R_{1,4} \arrow[d, equal] \\
0 & & & & \Delta H_1 \arrow[dllll, bend right] \\
\Delta H_1 \arrow[r, "P_1^2"] & \Delta R_{2,1} \arrow[r, "P_2^2"] & \Delta R_{2;2} \arrow[r, "P_3^2"] & \Delta R_{2,3} \arrow[r, "P_4^2"] & {}
\end{tikzcd}$$

$$\Delta R_{2,4} = \Delta H_2 = 0 .$$

Each P is a Boolean condition on the 128 bit differences

(Hand-study).

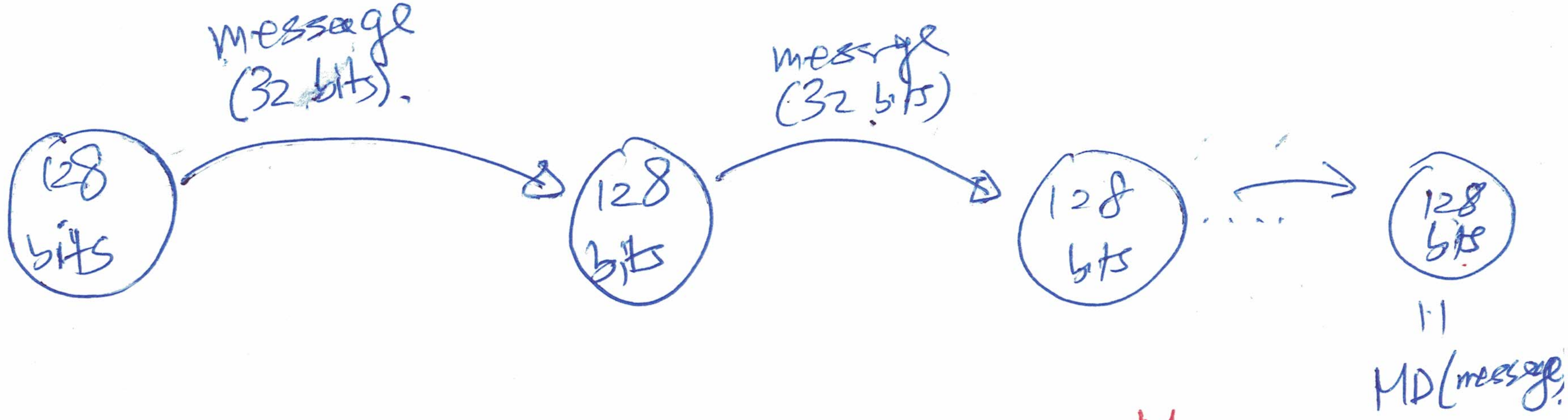
High-level:

① Those conditions are going to help us to brute-force on the messages but restricted over a "small" search space.

② Fundamental assumption: collision can be found with small Δ. (ΔM_1 is small). // bet.

①. We leaned BDD.

②. Another view of MD5:



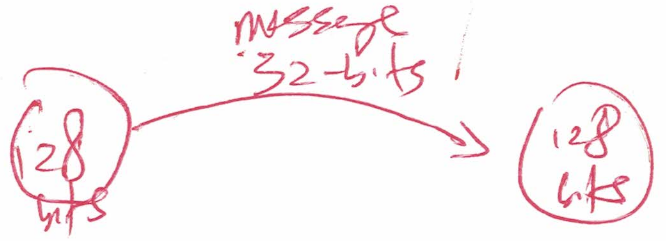
2.A. I treat The above as a "program" M running on 128 bits with 32-bit input on each step.

You are looking for:

Property: different messges have come to same hash val.

I want to check: M sat. the property.

2. B. use BDD to encode M as a Booleanformula over 128 + 32 + 128 = 256 Bool vars.



Modern processors should be able to handle these variables.

3. use SPIN (ask for education trial).

↳ it doesn't use BDD.

Can we use These BDD-like ideas for

512 bits key RSA ? NO, why ?

RSA uses p, q .

proven Can't be handled by BDD.