

Card-based Cryptographic Protocols for Three-input Functions with a Standard Deck of Cards Using Private Operations

Naoki Kobayashi Yoshifumi Manabe

Kogakuin University

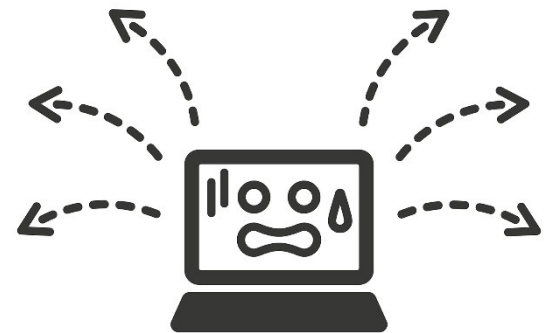
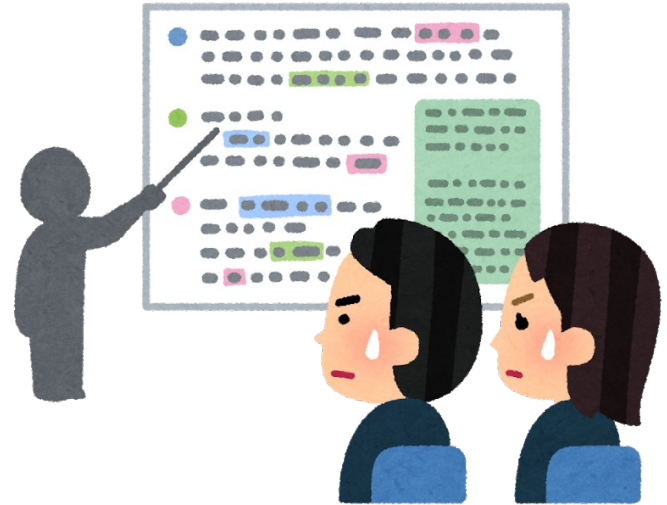
Shinjuku, Tokyo Japan

Outline of Talk

- What is “Card-based Cryptographic protocols?”
- Private operations
- Problem in using standard deck of cards
- Proposed protocols
- Conclusion

Cryptographic protocols using computers

- Difficult to understand the correctness for non-experts
- Cannot be executed when we cannot use computers
- Software might not be reliable: Is the private data really discarded?

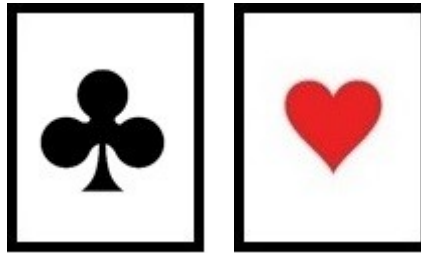


Card-based

Protocols

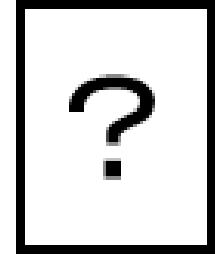
Cryptographic

- Using cards



- Can be executed when we cannot use computers
- Relatively easy to understand the protocol → can be used for teaching cryptography
- Private data are lost after the cards are shuffled

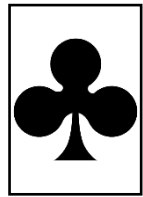
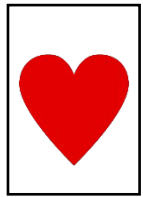
Back of the cards



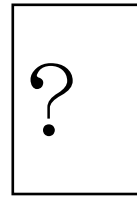
Overview of card-based cryptographic protocols

- Two kinds of protocols
 - Primitives to calculate any functions
 - AND, XOR, copy of inputs
 - Efficient protocols to solve specific problems
 - Voting/Grouping/Millionaires' Problem/etc.
- Executed by **semi-honest** Alice and Bob
 - Obey the rule but try to obtain private values
 - Private values must not be known to the players
- Most protocols use special cards
 - ⇒ Use standard deck of cards

Special cards used by most protocols

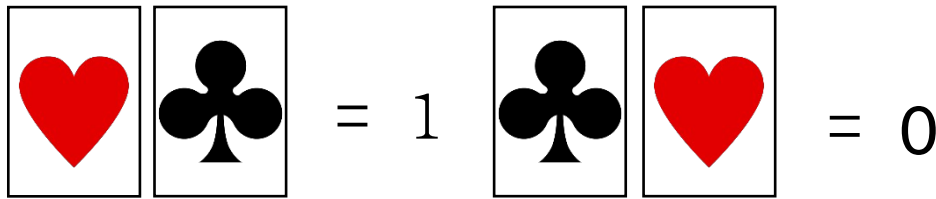


Back is

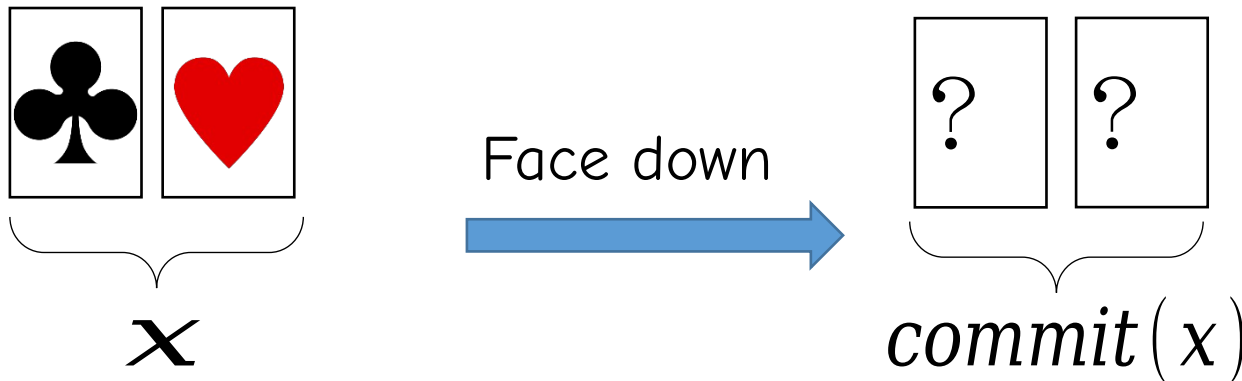


indistinguishable

- Encoding of one bit data



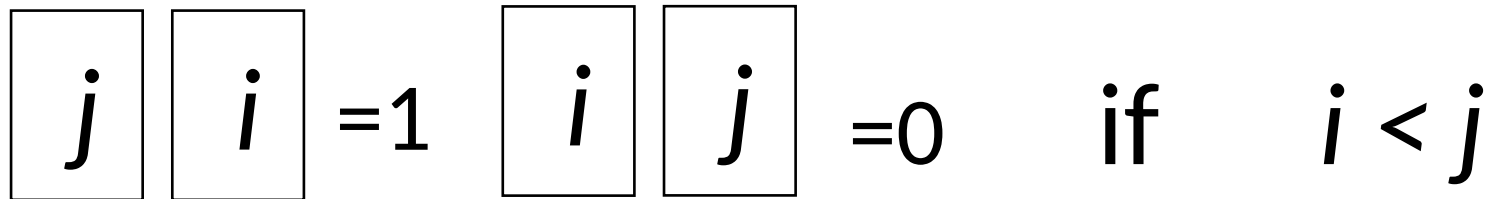
- Data and its commitment



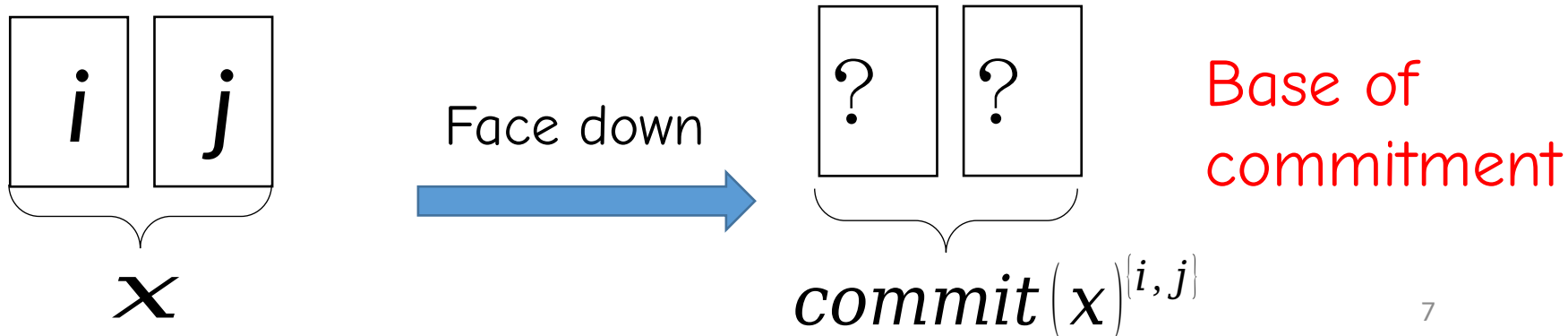
(This paper) Standard deck of cards



- Encoding of one bit data

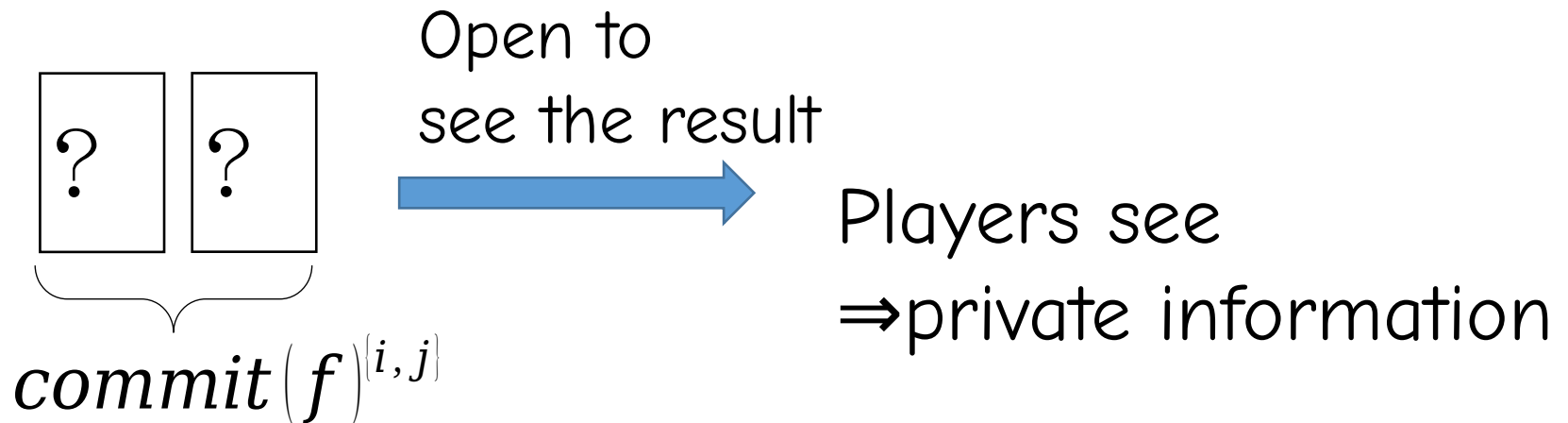


- Data and its commitment



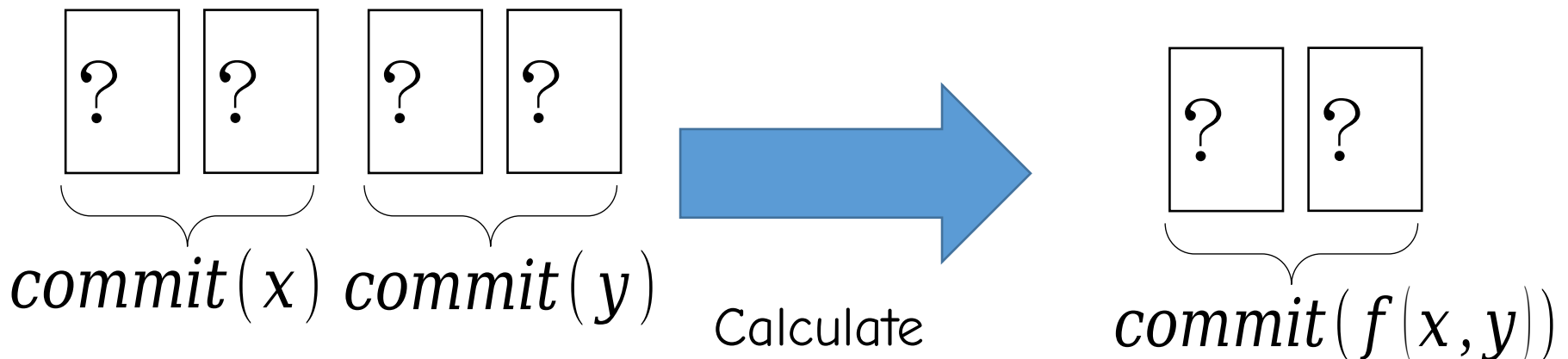
Difference between ♥♣ cards and standard deck of cards

- Information leakage from the bases of the cards







Requirements for inputs and outputs







- Input: protocols must allow committed inputs
 - Players can calculate using unknown values
- Output: protocols must output committed results
 - Can be used as inputs to further computation









private operations

- Some player executes operations where the other players cannot see (under the table, in the back): **private operations**
- [NTMIO16]   Millionaires' problem
- [OM18] :   Minimum number of cards for AND, XOR, Copy protocols

Number of additional cards (uniform closed shuffle, finite step)

	cards	Without Private operation	With Private operation
AND	 	2 [MS09]	0 [OM18]
	standard	4 [M16]	0 [MO24]
XOR	 	0 [MS09]	0 [OM18]
	standard	0 [M16]	0 [MO24]
Copy	 	2 [MS09]	0 [OM18]
	standard	2 [M16]	0 [MO24]

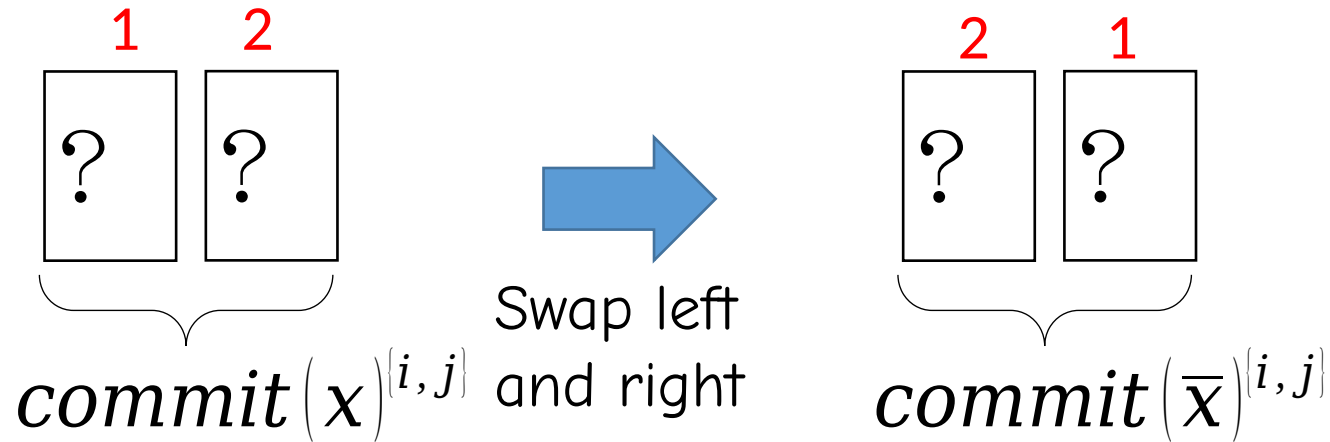
Number of additional cards (uniform closed shuffle, finite step)

	cards	Without Private operation	With Private operation
Three input fn.	 	2 [NHMS15]	0 [MO21]
	standard	4 [M16] ₊ ↑	0 [This]
Half/Full adder	 	2 [NHMS15]	0 [MO21]
	standard	4 [M16] ₊ ↑	0 [This]
Symmetric fn.	 	2 [NHMS15]	0 [MO21]
	standard	4 [M16] ₊ ↑	0 [This]

Primitives used in the protocols

- NOT
- AND with private operations
- (XOR, copy with private operations)
- XOR, AND with preserving an input
- Any Boolean function with 4 additional cards

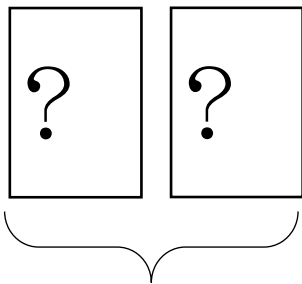
Primitive executable by anyone



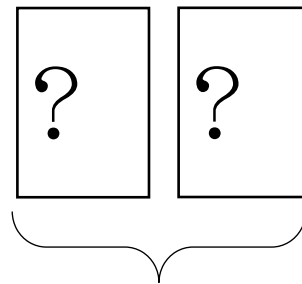
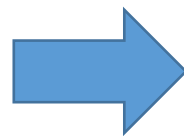
♥♣ AND protocol[OM18]

• Input: , Output:

(1) Alice(private):
Random swap



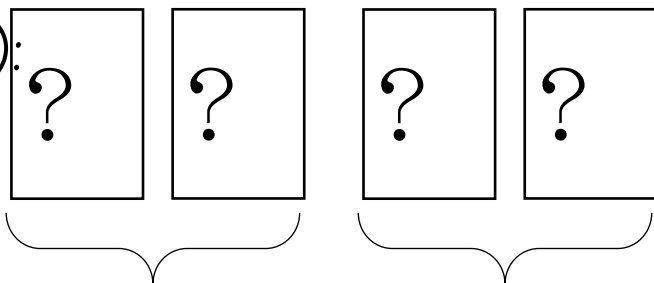
$commit(x)$



$commit(x \oplus a_1)$

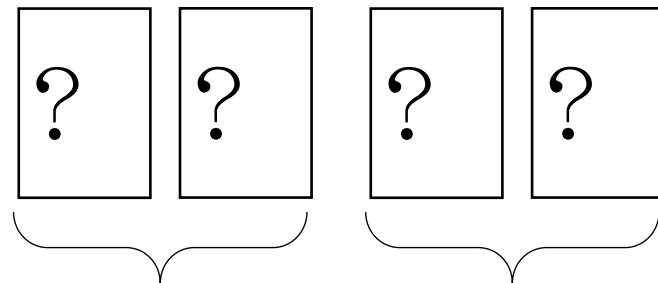
Randomly select

(2) Bob(private):
See
→ Set cards



$commit(y) \quad commit(0)$
 $x \oplus a_1 = 1$

OR



$commit(0) \quad commit(y)$
 $x \oplus a_1 = 0$

(3) Alice(private):

select

left pair if

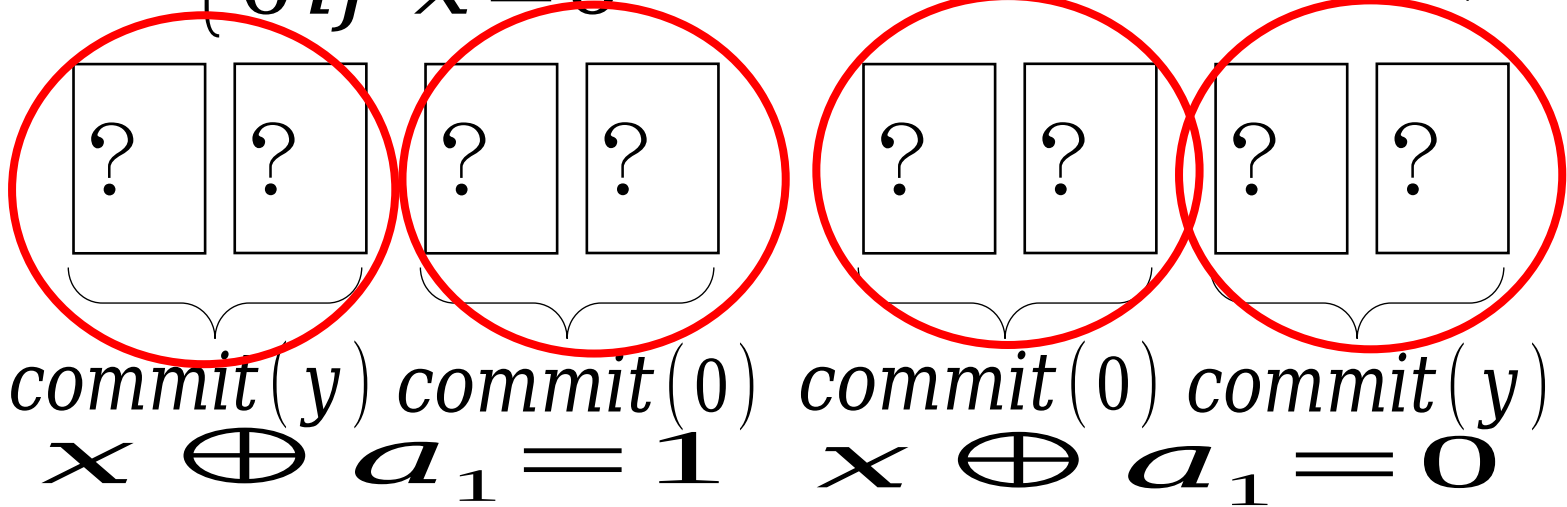
right pair if

of cards: 4 re-use
to set

Correctness of AND protocol

$$x \wedge y = \begin{cases} y & \text{if } x = 1 \\ 0 & \text{if } x = 0 \end{cases}$$

Alice selects left pair if
right pair if



Selects : and) or

(and) $\longleftrightarrow x = 1$

Selects : and) or

(and) $\longleftrightarrow x = 0$

AND: standard cards

$$x \wedge y = \begin{cases} y & \text{if } x = 1 \\ 0 & \text{if } x = 0 \end{cases}$$

- Input:

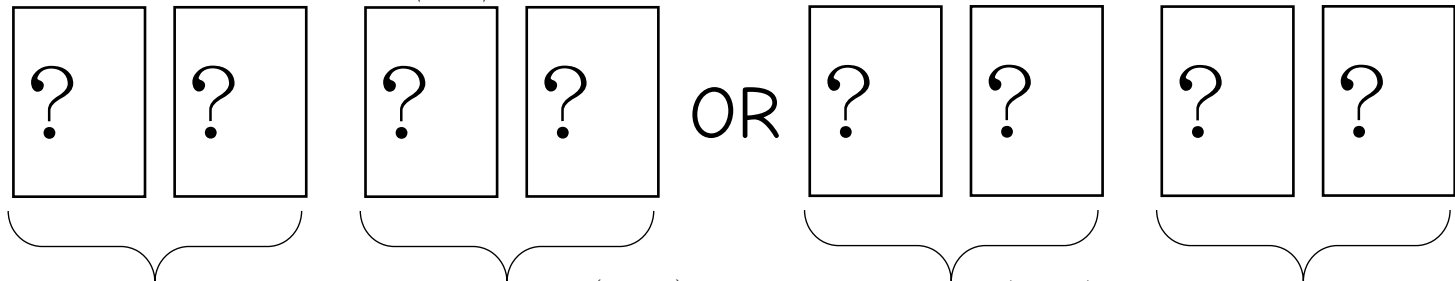
(1) Alice:
Random swap



$commit(x)^{(1,2)}$ Randomly select

$commit(x \oplus a_1)^{(1,2)}$

(2) Bob:
Reveal
→ Set cards



$commit(y)^{(3,4)}$

$commit(0)^{(1,2)}$

$commit(0)^{(1,2)}$

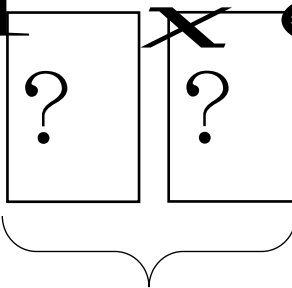
$commit(y)^{(3,4)}$

$x \oplus a_1 = 1$

$x \oplus a_1 = 0$

(3) Alice:
Select
left pair if
right pair if

Suppose
1



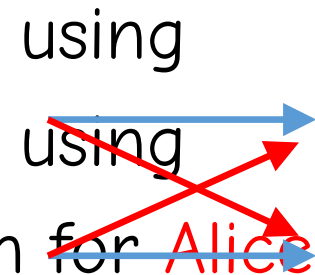
Open cards
→ sees

$commit(y)^{(3,4)}$

For execution using standard cards

- Prevent private input data leakage from the bases of cards
- (Definition) semi-opaque commitment pair

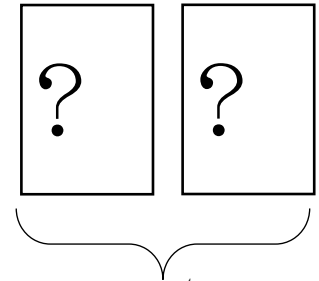
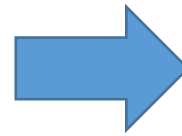
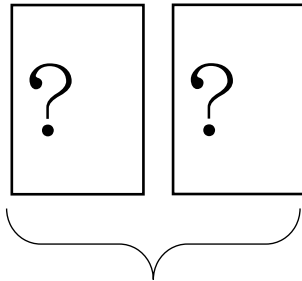
using
using
is unknown for Alice



AND protocol[M024]

- Input: , Output:

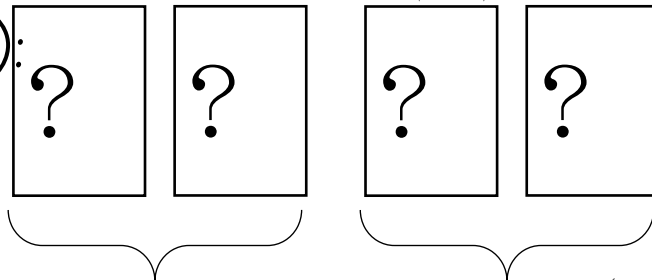
(1) Alice(private):
Random swap



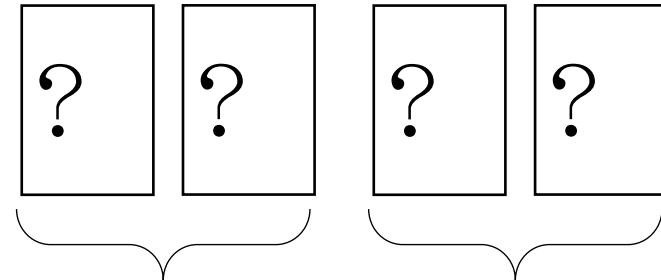
commit(x)^{1,2} Randomly select

commit($x \oplus a_1$)^{1,2}

(2) Bob(private):
See
→Set cards



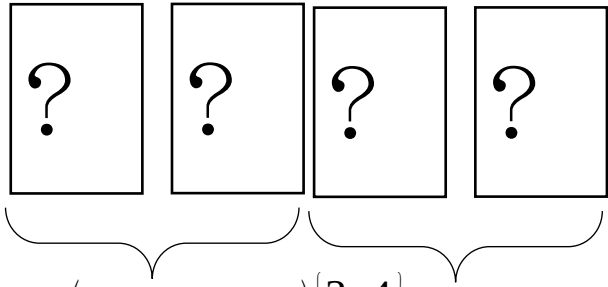
OR



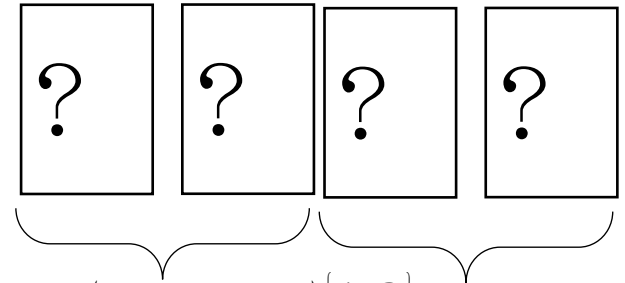
commit(y)^{3,4} *commit*(0)^{1,2} *commit*(0)^{1,2} *commit*(y)^{3,4}
 $x \oplus a_1 = 1$ $x \oplus a_1 = 0$

AND protocol(2)

(3) Alice:
random
swap



OR



$$\text{commit}(y \oplus a_2)^{\{3,4\}}$$

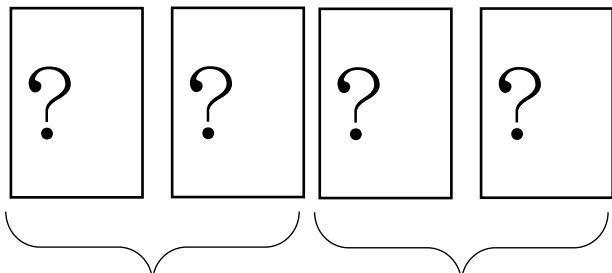
$$\text{commit}(0 \oplus a_3)^{\{1,2\}}$$

$$\text{commit}(0 \oplus a_2)^{\{1,2\}}$$

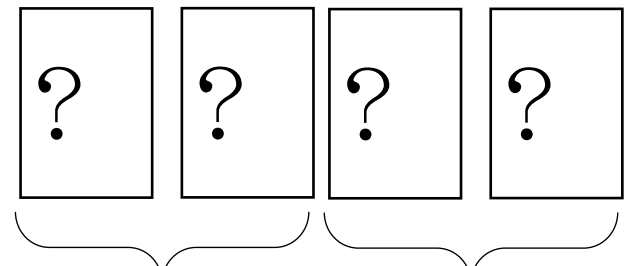
$$\text{commit}(y \oplus a_3)^{\{3,4\}}$$

(4) Bob: sees cards

Randomly select and swap $\{1,2\}\{3,4\}$ if



OR



$$\text{commit}(y \oplus a_2)^{\{1,2\},\{3,4\} \vee A}$$

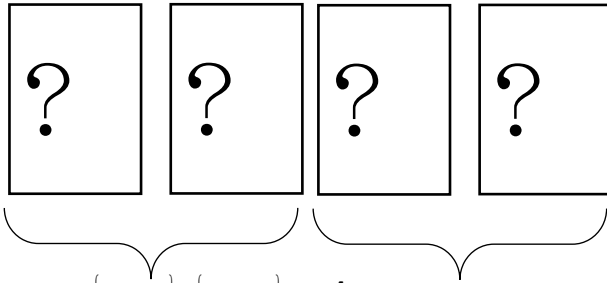
$$\text{commit}(0 \oplus a_3)^{\{1,2\},\{3,4\} \vee A}$$

$$\text{commit}(0 \oplus a_2)^{\{1,2\},\{3,4\} \vee A}$$

$$\text{commit}(y \oplus a_3)^{\{1,2\},\{3,4\} \vee A}$$

AND protocol(3)

(5) Alice:



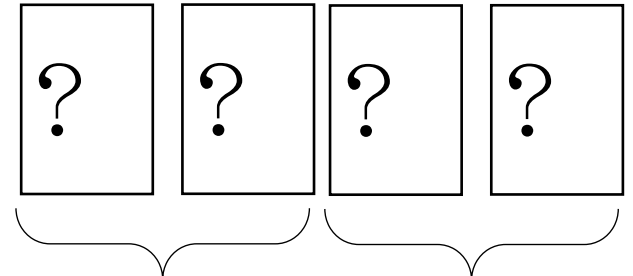
undo

swap

$$\text{commit}(y)^{[1,2],[3,4] \vee A}$$

$$\text{commit}(0)^{[1,2],[3,4] \vee A}$$

OR



$$\text{commit}(0)^{[1,2],[3,4] \vee A}$$

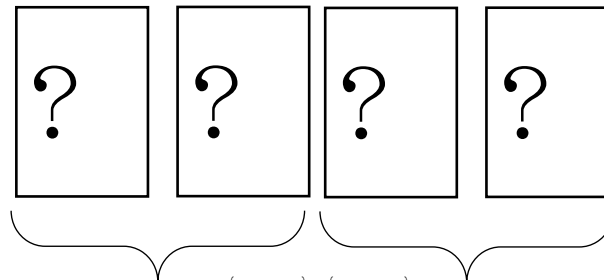
$$\text{commit}(y)^{[1,2],[3,4] \vee A}$$

Alice:

Select

left pair if

right pair if



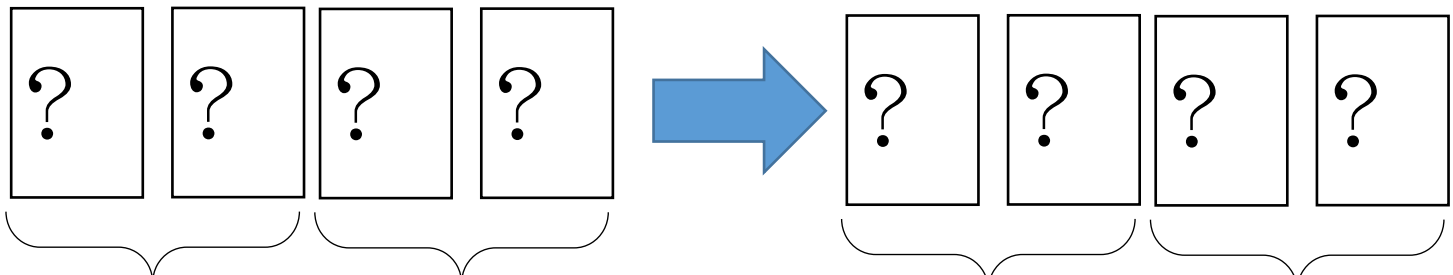
$$\text{commit}(x \wedge y)^{[1,2],[3,4] \vee A}$$

unselected pair:

AND protocol(4)

(6) Bob: For the selected pair and unused pair

Random swap using



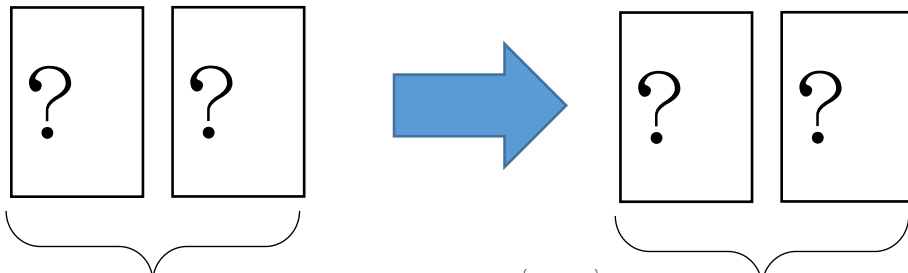
$$\text{commit}(x \wedge y)^{[1,2],[3,4] \vee A}$$

unselected pair:

$$\text{commit}(x \wedge y \oplus br_1)^{[1,2],[3,4] \vee A}$$

$$\text{commit}(\perp \oplus br_2)^{[1,2],[3,4] \vee A}$$

(7) Alice: see cards and set using {1,2}



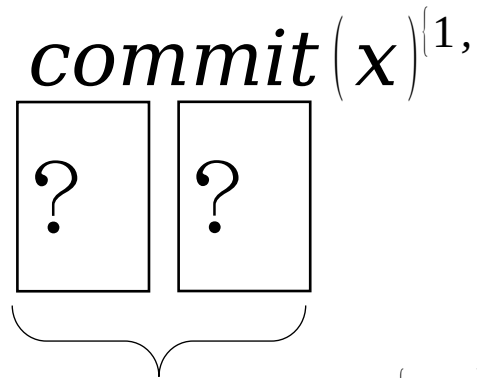
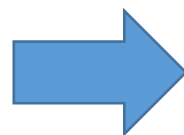
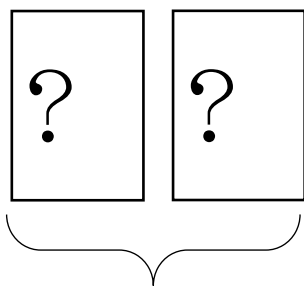
$$\text{commit}(x \wedge y \oplus br_1)^{[1,2]} \text{commit}(x \wedge y)^{[1,2]}$$

(8) Bob: undo randomization by \rightarrow obtain

XOR with preserving an input

• Input : Output :

(1) Alice:
Random swap



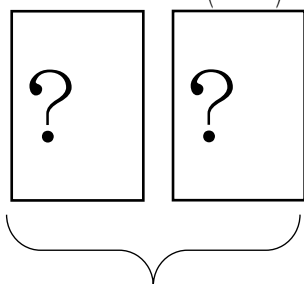
$commit(x)^{(1,2)}$ Randomly select

$commit(x \oplus b)^{(1,2)}$

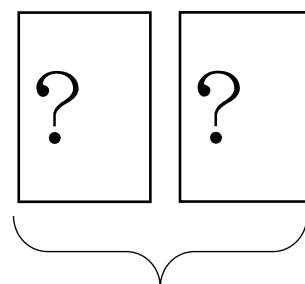
(2) Bob: (Private)
see :

→ set cards

returns



OR



$commit(\bar{y})^{(3,4)}$
 $x \oplus b = 1$

$commit(y)^{(3,4)}$
 $x \oplus b = 0$

(3) Alice:
Undo Swap if

$commit(y \oplus (x \oplus b))^{(3,4)}$

$commit(y \oplus (x \oplus b) \oplus b)^{(3,4)} = commit(x \oplus y)^{(3,4)}$

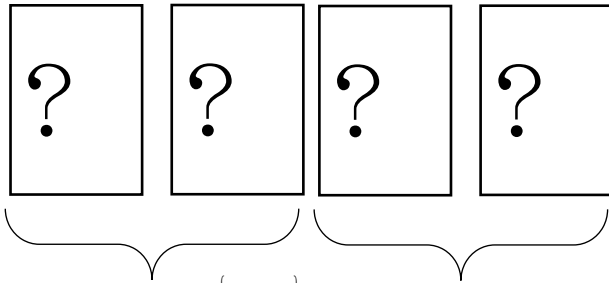
$commit(x)^{(1,2)}$

AND with preserving an input

- Input : Output :

$commit(y)^{(3,4)}$

For the selected pair and unused pair



Unselected pair:

$commit(x \wedge y)^{(1,2)}$

unselected pair:

XOR with preserving one input
between and

\Rightarrow

Any Boolean Function with 4 additional cards[M024]

- Two pairs(initial value 0)
: Store intermediate value
- Boolean function can be written
()

For to

copy

For to

(AND preserving input :)

Outline of the result

- Three-input Boolean function
- Half and full adder
- Any symmetric Boolean function

From the next page,
is written as

Three-input Boolean functions

Categorized to the following 14 functions by considering swapping input variables and negations[SB10]



Execute

AND/XOR primitives

$$NP N_7 = (x \wedge y) \vee (x \wedge z) = x \wedge (y \vee z)$$

$$NP N_{13} = (x \wedge y \wedge z) \vee (x \wedge \bar{y} \wedge \bar{z}) = x \wedge (\overline{y \oplus z})$$

$$NP N_6 = (x \wedge y \wedge z) \vee (\bar{x} \wedge \bar{y} \wedge \bar{z})$$

• Input

↓ XOR preserving NOT

↓ XOR, NOT

•

↓ AND

•

— :variables
 — current
 — operation is
 — applied

Red: newly obtained value

$$NP N_8 = (x \wedge y) \vee (\bar{x} \wedge \bar{y} \wedge z)$$

- Input



XOR with preserving ,NOT



OR(using AND)

- ,



AND

-

$$NP N_9 = (x \wedge y \wedge \bar{z}) v$$

- Input



 XOR with preserving input

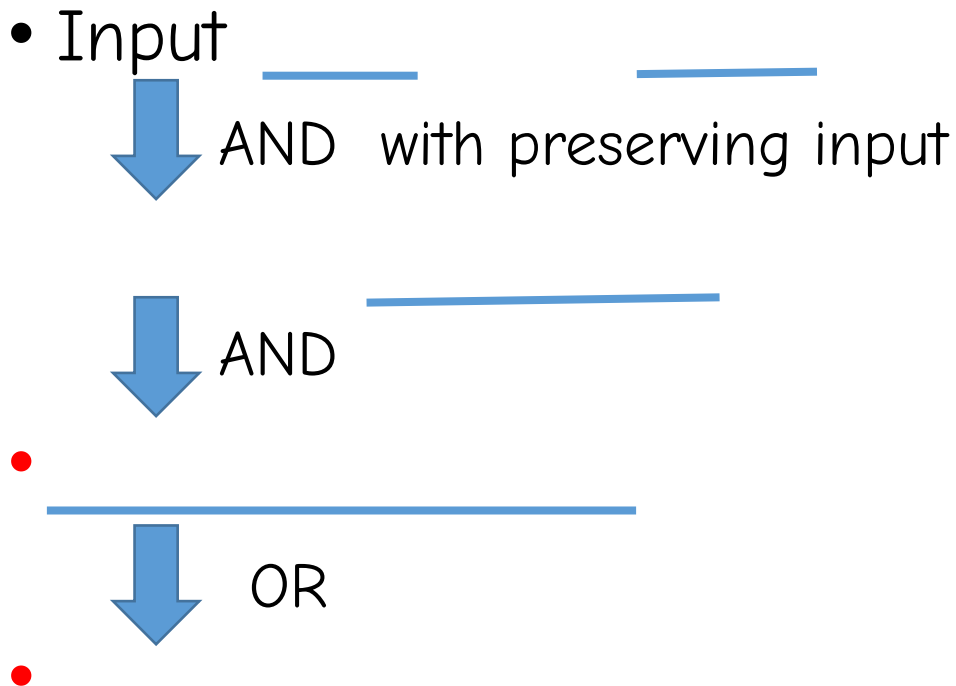

 XOR with preserving input NOT


 OR


 AND



$$NP N_{12} = (x \wedge \bar{z}) \vee (y \wedge z)$$



$$NP N_{11} = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$$

- Input



XOR with preserving input

Apply modified AND protocol

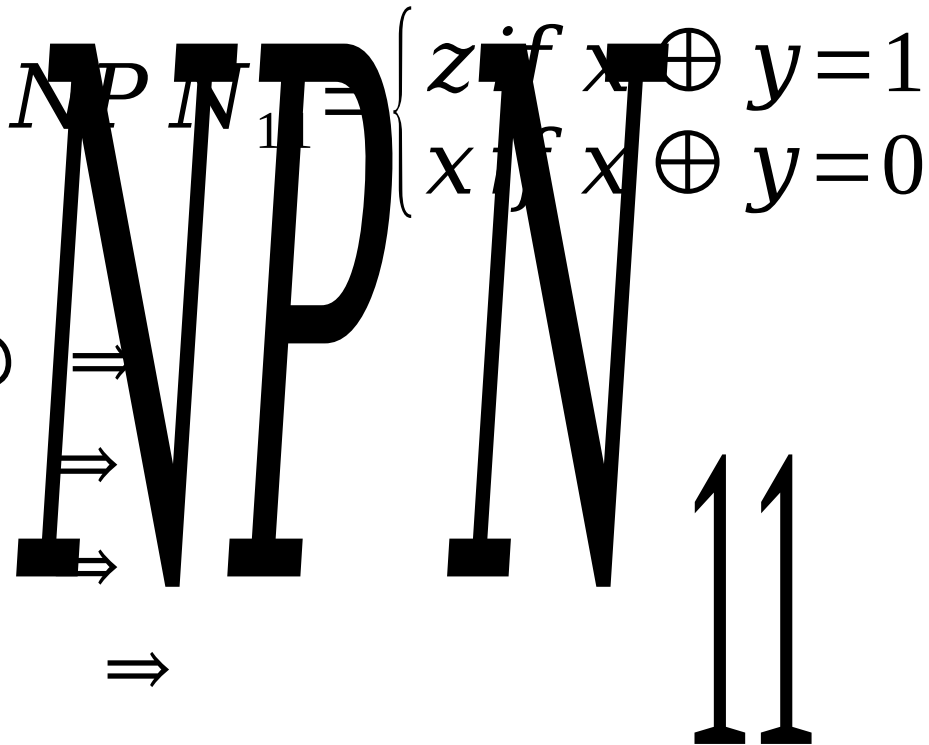


Modified AND protocol

- AND

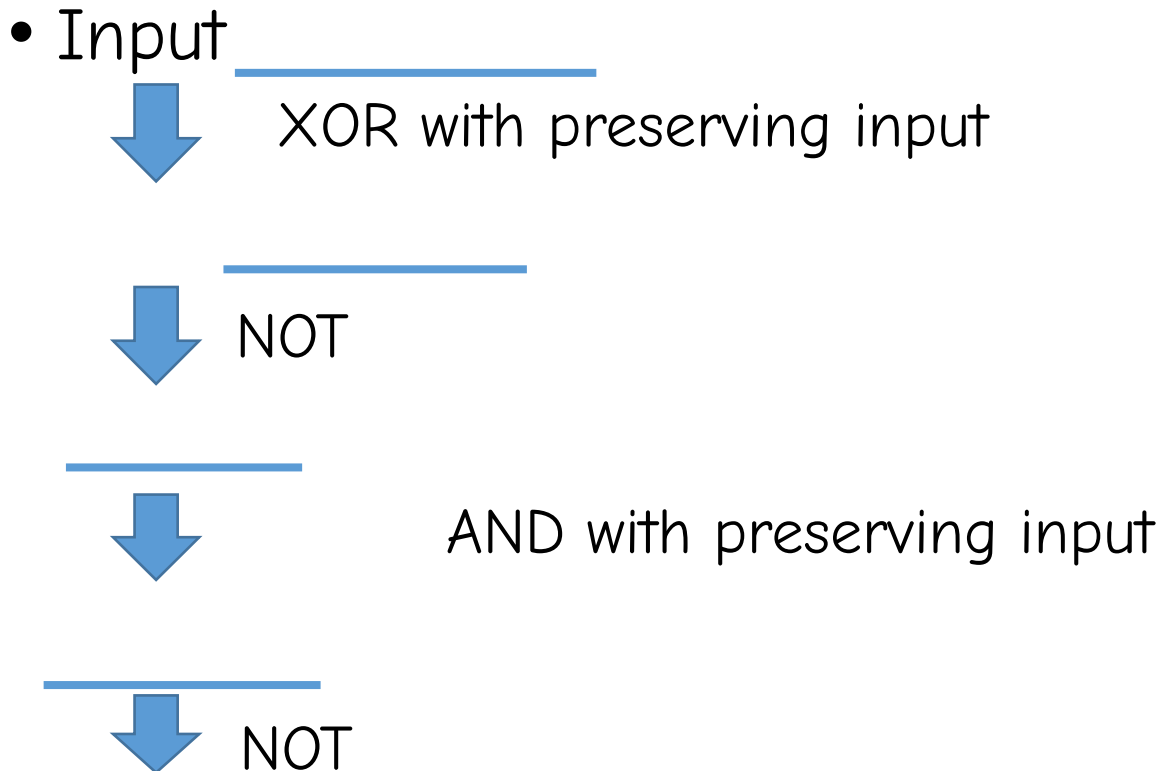
$$x \wedge y = \begin{cases} y & \text{if } x = 1 \\ 0 & \text{if } x = 0 \end{cases}$$

AND



(2) Half adder

Output:



Full adder

Output: ,

- Input



Half adder



Half adder



OR(using AND and NOT)

- ,

(3) Symmetric function for any

- is decided by the numbers of

$$k = \lfloor \log n \rfloor + 1$$

- be binary representation



Half adder, Full adder



Realization of any Boolean function

(1)

Result is 3 bits \Rightarrow Any three-input can be realized without additional cards \Rightarrow can be realized without additional cards

(2)

Result bits

\Rightarrow unused input cards

can be realized with 4 additional cards

\Rightarrow can be realized without additional cards

Conclusion

- Using private operations, minimum card protocol using standard deck of cards
 - Three-input Boolean function
 - Half and full adder
 - Symmetric Boolean function
- Open problems
 - n -variable Boolean function
 - Minimizing the number of steps

Backup slides

$$x \wedge y = \begin{cases} y & \text{if } x = 1 \\ 0 & \text{if } x = 0 \end{cases}$$

Security of AND protocol

- Bob: sees \rightarrow no information about x because of randomization by b_3
- Alice: sees y , b_3

$$b_3 = 0$$
~~$$b_3 = 1$$~~

Values are randomized by b_3

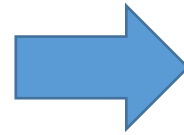
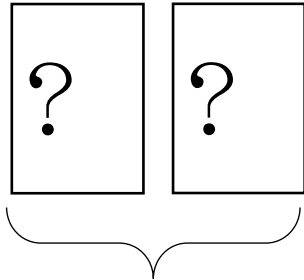
Sees $\{1,2\}$ or $\{3,4\}$ is randomized by b_3

\rightarrow No information from the cards

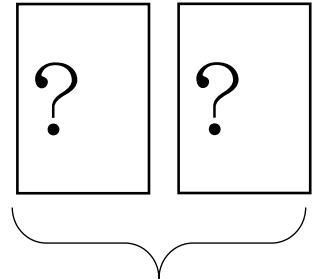
COPY protocol

• Input : , Output :

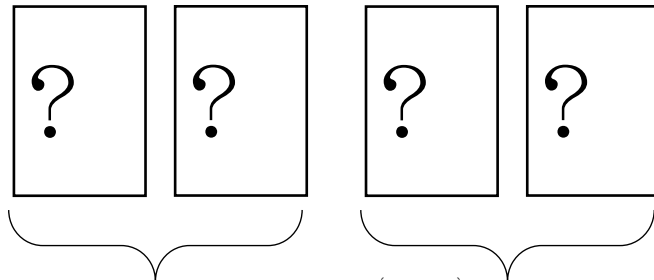
(1) Alice:
Random swap



Randomly select

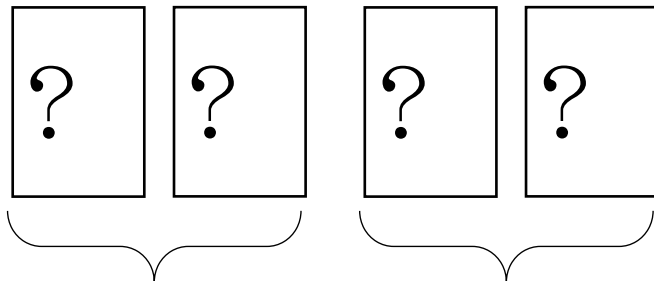


(2) Bob:
See
Copy the value



$commit(x \oplus b)^{1,2}$ $commit(x \oplus b)^{3,4}$

(3) Alice : Undo
randomization
for each pair

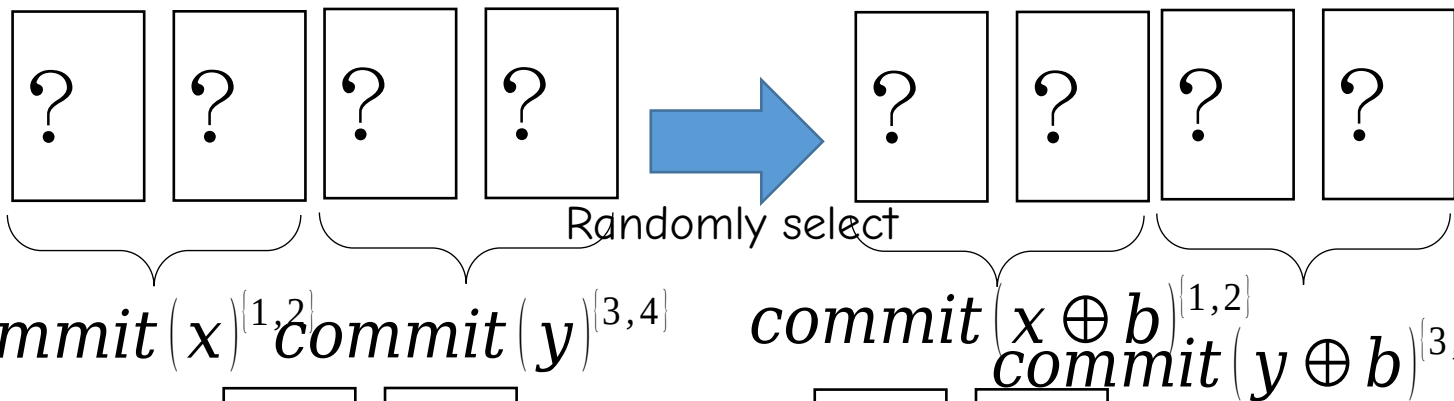


$commit(x)^{1,2}$ $commit(x)^{3,4}$

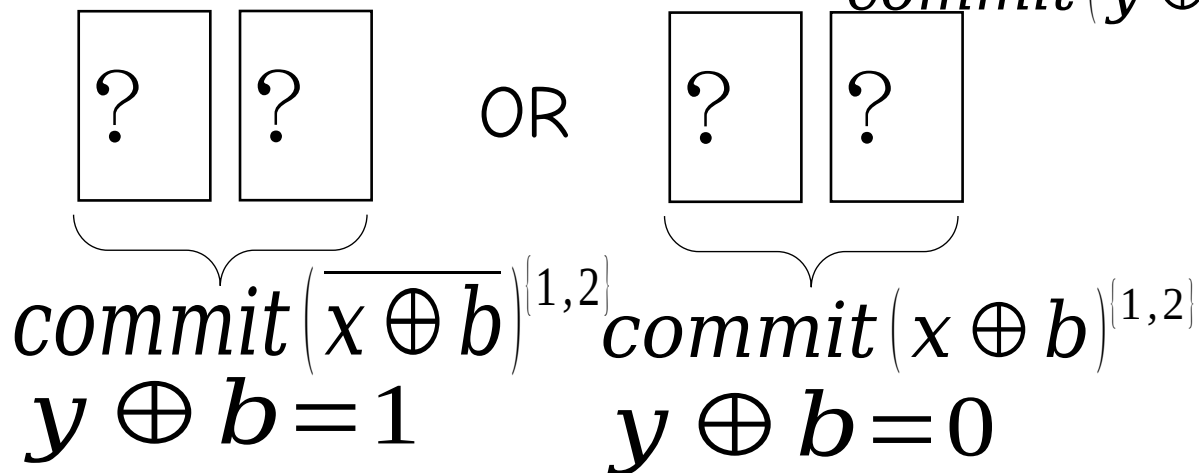
XOR protocol

• Input : Output :

(1) Alice:
random
swap



(2) Bob:
see and
swap
if the value is 1



Correctness:
Output is

Any n-variable Boolean function

Input:

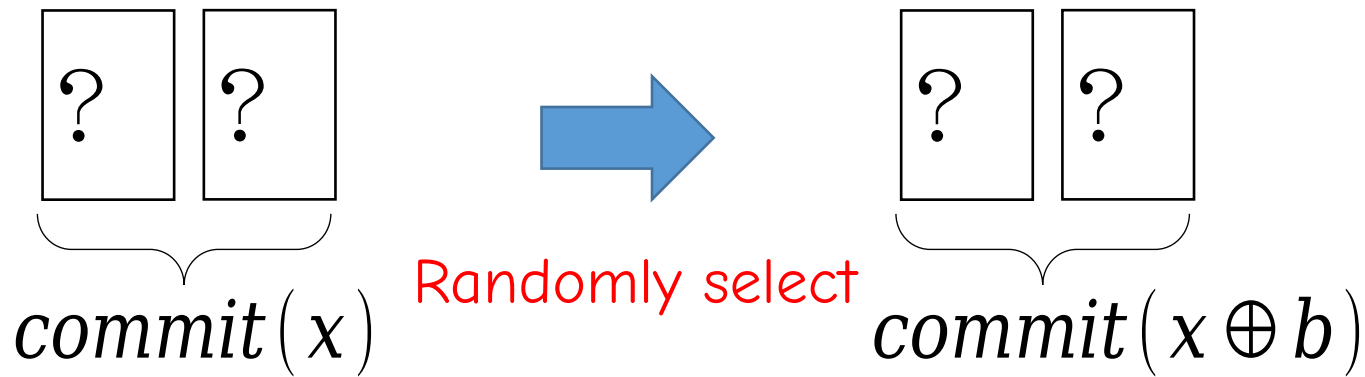
Output

(1) 4 rounds, # of cards

(2) rounds, # of cards

Private operation(1) [42]

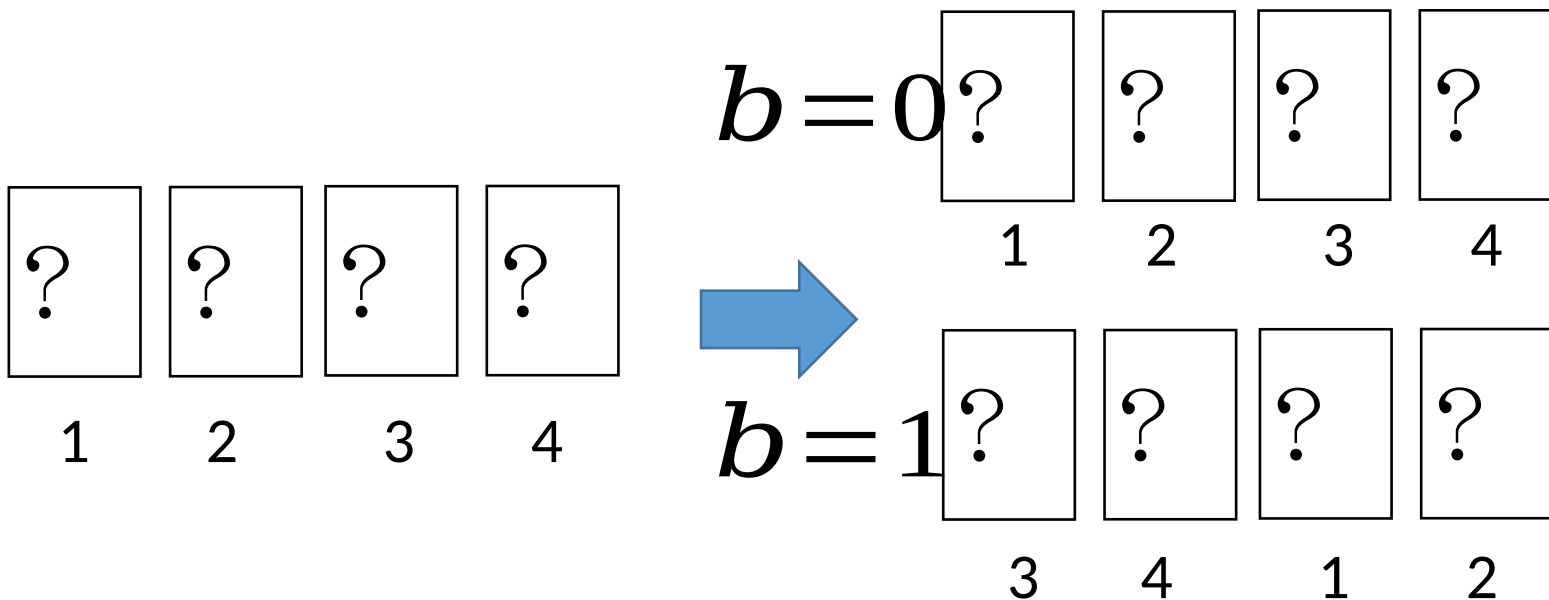
- Private random bisection cut: execute random bisection cut[31] in a hidden place



- Remember . Do not disclose to any other player

Private operation(2) [42]

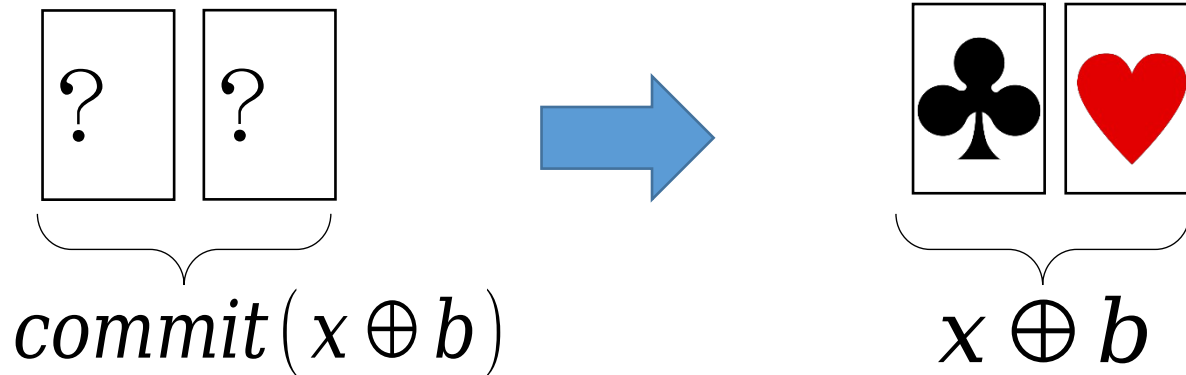
- Private reverse cut : swaps left and right of an even sequence using remembered



- (Private reverse selection :
select left if right if)







Private operation(3) [42]

- Private reveal: open cards and read the value



Executed by a player who does not know
→No information about




AND protocols

	Cards	# of cards	Note
[MS09]	 	6	
[RI18]	 	4	Las Vegas algorithm †
[OM18]	 	4	Use private operations
[NR99]	Standard	5	Las Vegas algorithm †
[KSK21]	Standard	4	Las Vegas algorithm †
[M16]	Standard	8	
[MO24]	Standard	4	Use private operations

† Bounded average execution time, no termination if random values are bad

Four cards are necessary for two inputs


Copy protocols

	Cards	# of cards	Note
[MS09]	 	6	
[OM18]	 	4	Use private operations
[NR99]	Standard	6	Las Vegas algorithm †
[M16]	Standard	6	
[MO24]	Standard	4	Use private operations

† Bounded average execution time, no termination if random values are bad

4 cards are necessary for copy

XOR protocols

	Cards	# of cards	Note
[MS09]	 	4	
[OM18]	 	4	Use private operations
[NR99]	Standard	4	Las Vegas algorithm \neq
[M16]	Standard	4	
[MO24]	Standard	4	Use private operations

\neq Bounded average execution time, no termination if random values are bad

Four cards are necessary for two inputs