Improving the Round Complexity of Ideal-Cipher Constructions

Aishwarya Thiruvengadam

Block Ciphers

- Building block for many cryptographic constructions
  - Hash functions
  - Encryption schemes
  - Message authentication codes

Block Ciphers

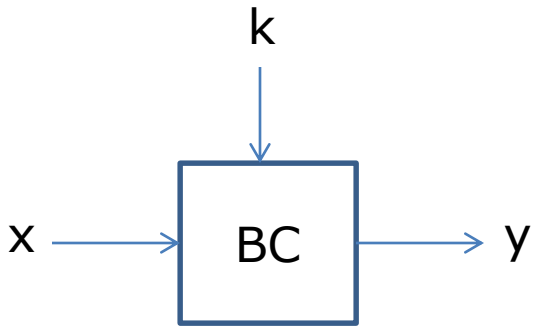
- Popular approaches to block cipher designs
  - Feistel Networks
    - DES
    - Applications of keyed round functions
  - Key-alternating ciphers
    - AES
    - Applications of public round permutations

Outline

- Security of Block Ciphers
  - Indifferentiability [MRH04]
- Security of Feistel Networks [DKT16]
- Security of Key-alternating Ciphers [DSST17]

Block Ciphers

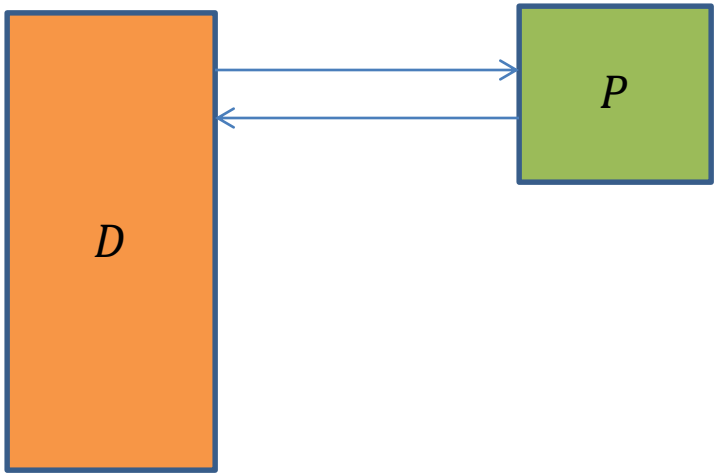
- Inputs: key k, input x
- Output: y
- Keyed permutations



- BC: $\{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$

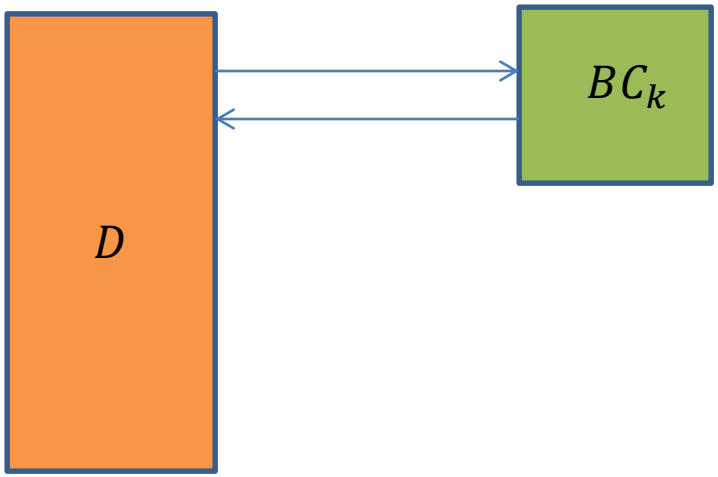
Security of Block Ciphers: Indistinguishability

- Ideal World



- P – random permutation

- Real World



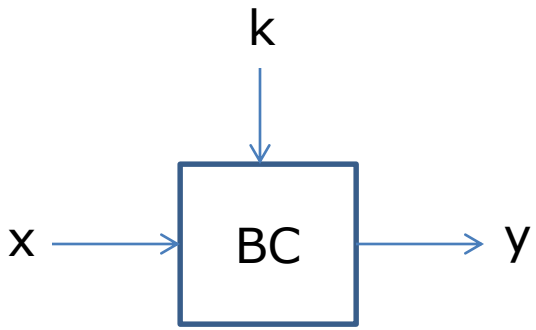
- BC_k - block cipher with key k

Security of Block Ciphers: Indifferentiability [MRH04]

- Is an r-round block cipher an ideal cipher?
  - Under appropriate assumptions on the underlying primitive O

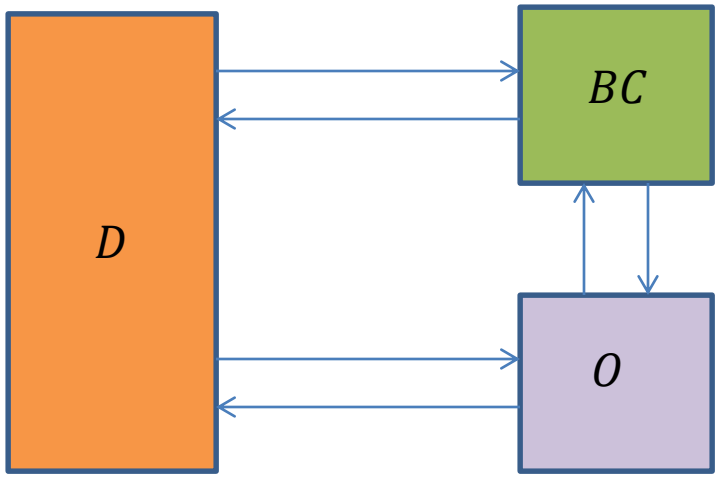
Ideal Cipher

- For each key k
  - $BC_k(\cdot)$ – uniform random permutation



Indifferentiability

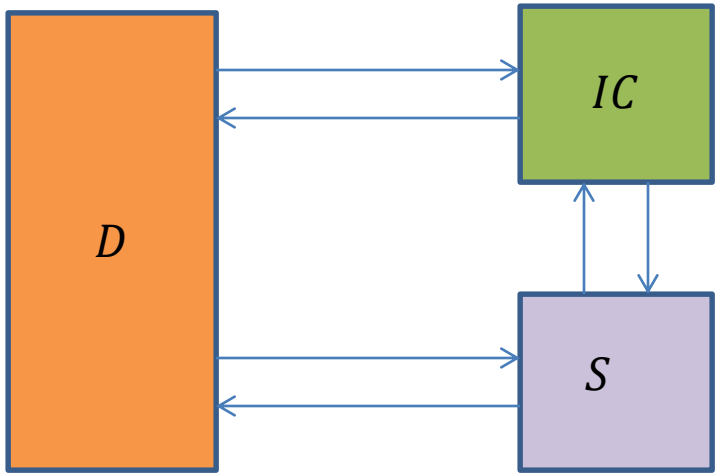
- Real World



- BC – block cipher construction
- $O = \{O_1, \ldots, O_r\}$, round functions

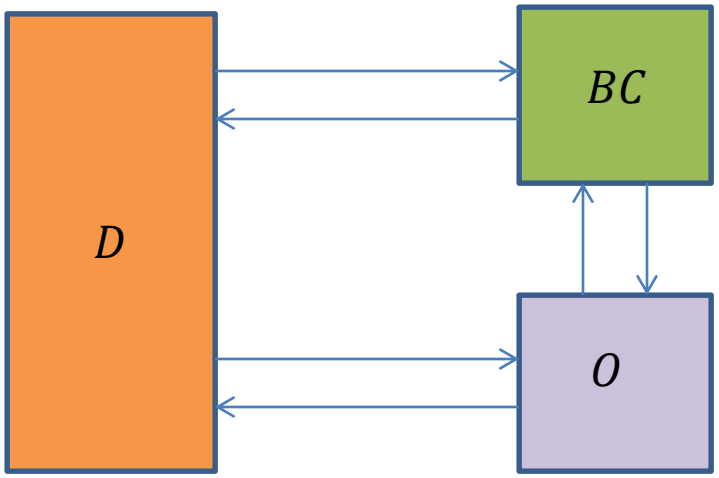
Indifferentiability

- Ideal World



- IC – random permutation
- S – alg. simulating round functions

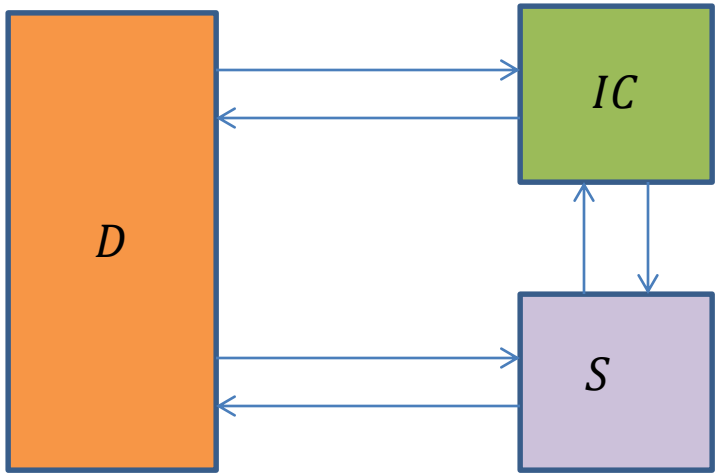
- Real World



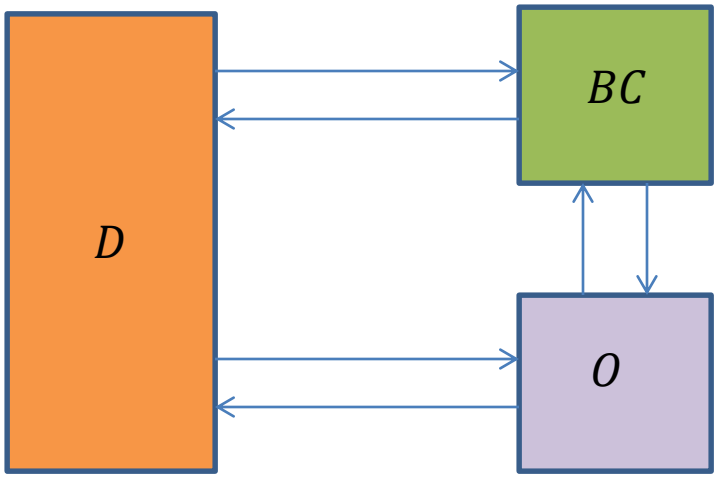
- BC – block cipher construction
- $O = \{O, \ldots, O_r\}$

Indifferentiability

- Ideal World



- Real World

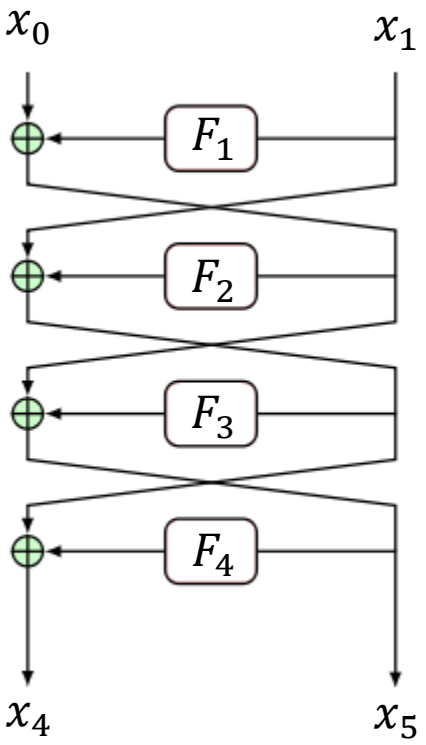


Block cipher construction BC indifferentiable from an ideal cipher IC if:
(efficient) S s.t.
No (efficient) D can distinguish between real and ideal w.h.p

Indifferentiability of Feistel Networks

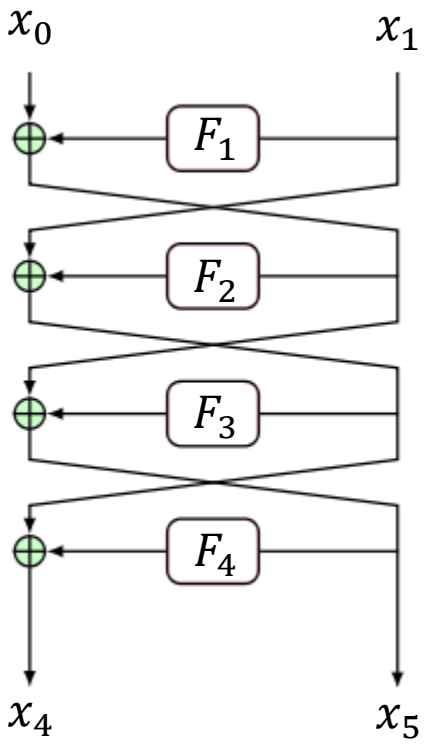
Feistel Network

- Iterated structure
- Repeated application of round functions
  - $F_1, \ldots, F_r : \{0,1\}^n \rightarrow \{0,1\}^n$
- Yields permutation



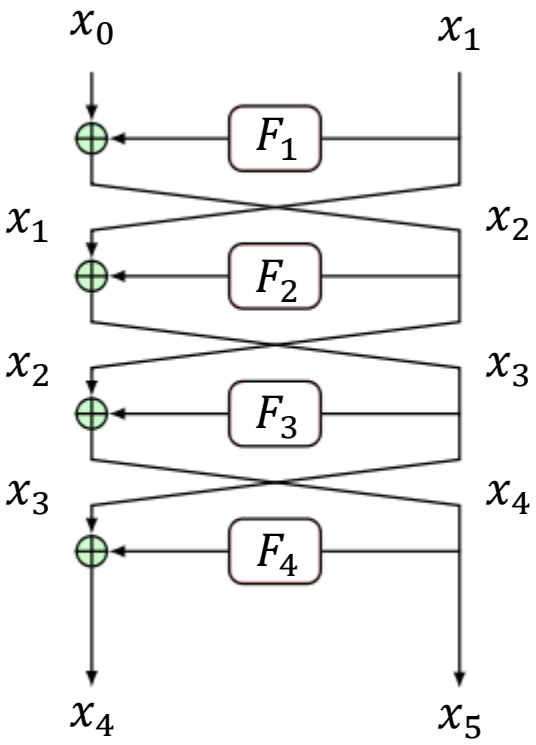
Feistel Network

- Input: $2n$-bit string x_0, x_1
- Output (after r rounds): $2n$-bit string x_r, x_{r+1}



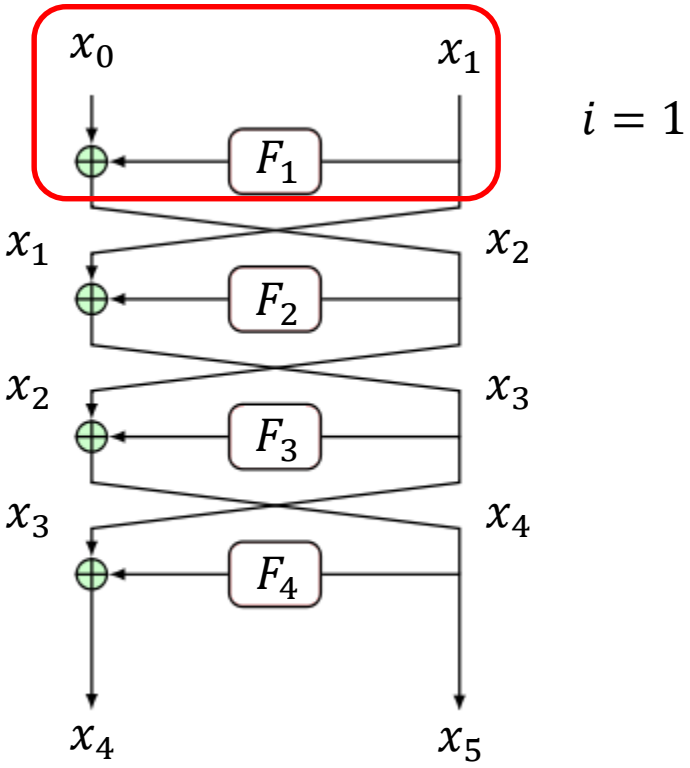
Feistel Network

- Input: x_0, x_1
- For $i = 1$ to r
  - Input: x_{i-1}, x_i
  - $x_{i+1} = F_i(x_i) \oplus x_{i-1}$
  - Output: x_i, x_{i+1}
- Output (after r rounds): x_r, x_{r+1}



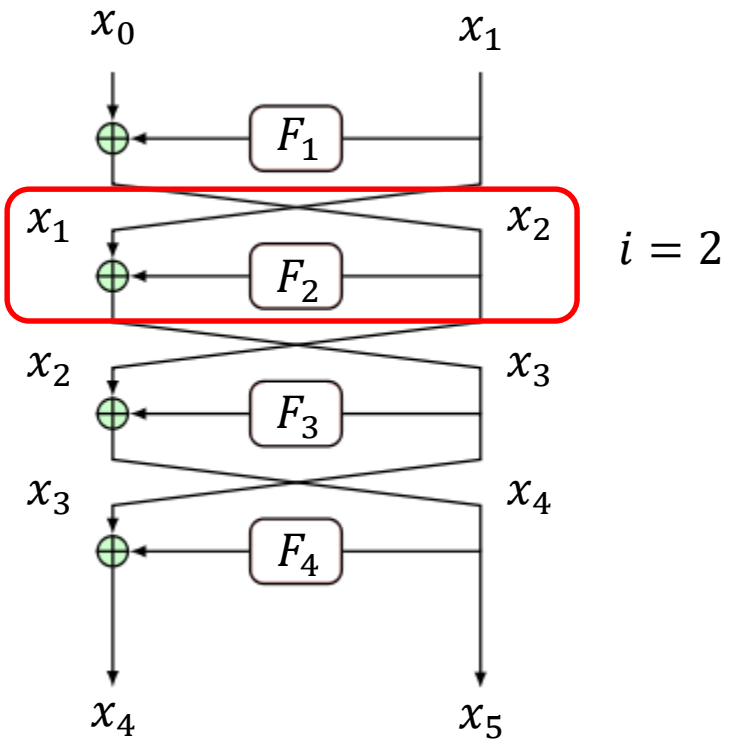
Feistel Network

- Input: x_0, x_1
- For $i = 1$ to r
  - Input: x_{i-1}, x_i
  - $x_{i+1} = F_i(x_i) \oplus x_{i-1}$
  - Output: x_i, x_{i+1}
- Output (after r rounds): x_r, x_{r+1}



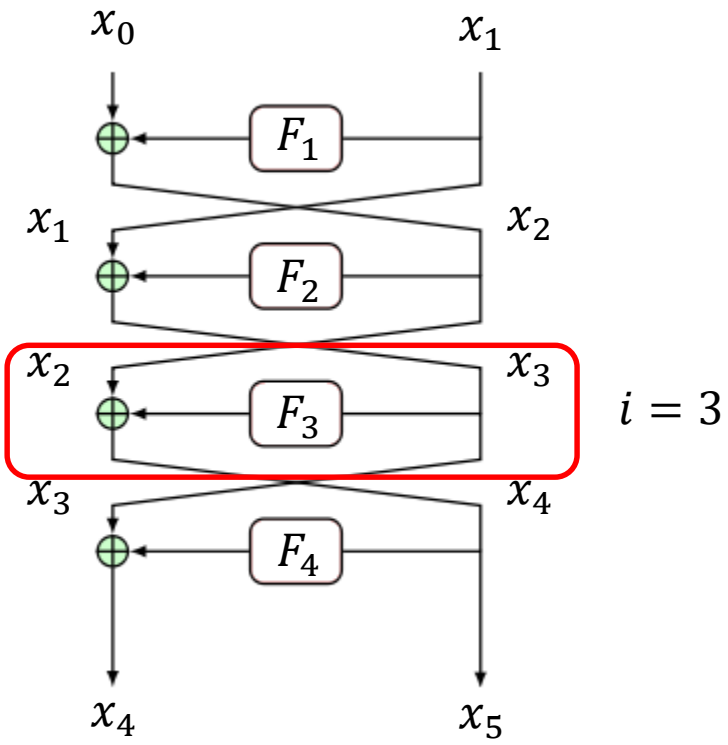
Feistel Network

- Input: x_0, x_1
- For $i = 1$ to r
  - Input: x_{i-1}, x_i
  - $x_{i+1} = F_i(x_i) \oplus x_{i-1}$
  - Output: x_i, x_{i+1}
- Output (after r rounds): x_r, x_{r+1}



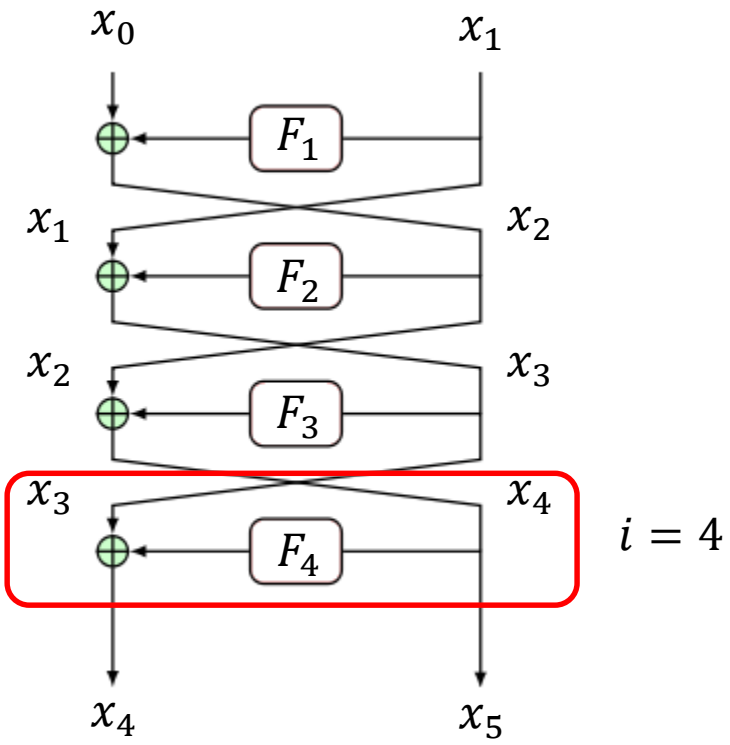
Feistel Network

- Input: x_0, x_1
- For $i = 1$ to r
  - Input: x_{i-1}, x_i
  - $x_{i+1} = F_i(x_i) \oplus x_{i-1}$
  - Output: x_i, x_{i+1}
- Output (after r rounds): x_r, x_{r+1}



Feistel Network

- Input: x_0, x_1
- For $i = 1$ to r
  - Input: x_{i-1}, x_i
  - $x_{i+1} = F_i(x_i) \oplus x_{i-1}$
  - Output: x_i, x_{i+1}
- Output (after r rounds): x_r, x_{r+1}



Security of Feistel Networks

- Is an r-round Feistel network an ideal cipher?
  - F independent, public random functions

Related Work

- 5 rounds are insufficient [CPS08]
- 14 rounds sufficient [HKT11,CHKPST14]
- This work :
  - 10 rounds are sufficient
- Further improvement : 8 rounds sufficient [DS16]

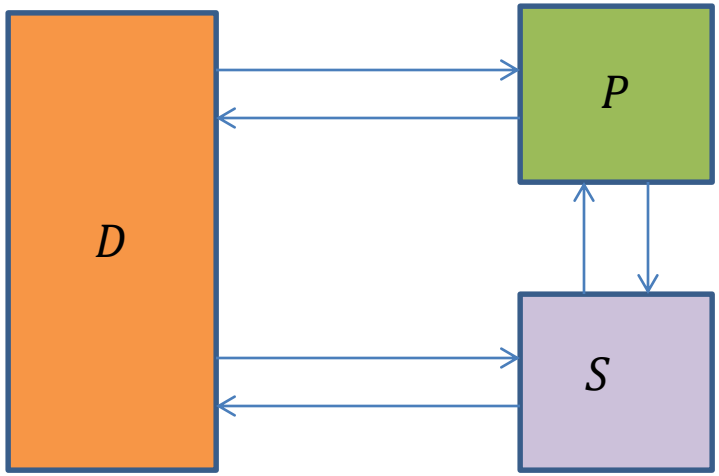
Our Result

10-round (keyed) Feistel network indifferentiable from an ideal cipher

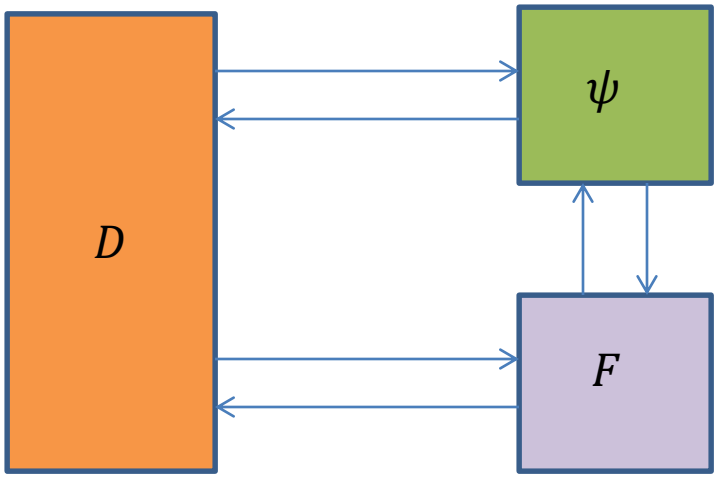
- Sufficient to show:
  - 10-round (unkeyed) Feistel network indifferentiable from a random permutation

Indifferentiability

- Ideal World



- Real World



Sufficient to show: (efficient) S s.t.

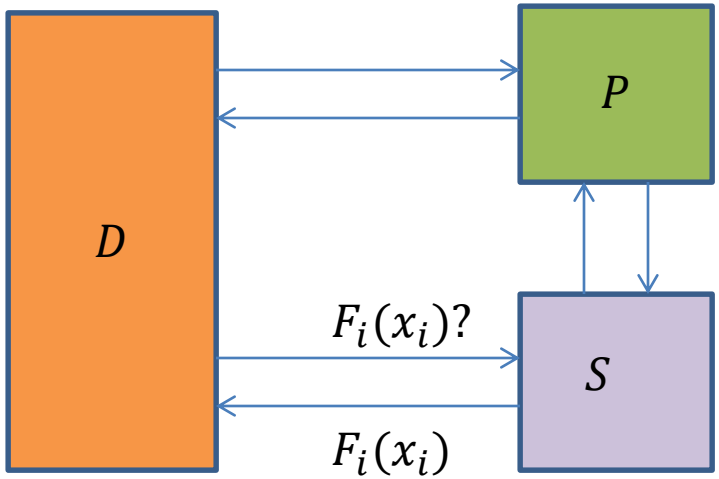
No (efficient) D can distinguish between real and ideal w.h.p

Naïve Simulator Strategy

- On query $F_i(x_i)$, return uniform value

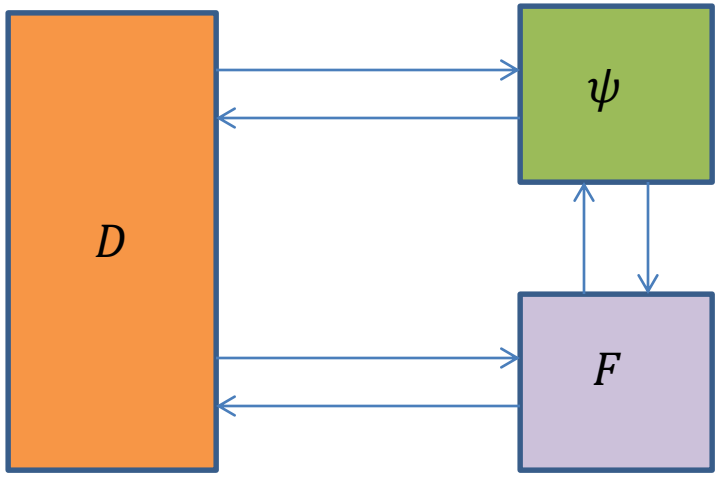
Naïve Simulator Strategy

- On query $F_i(x_i)$, return uniform value



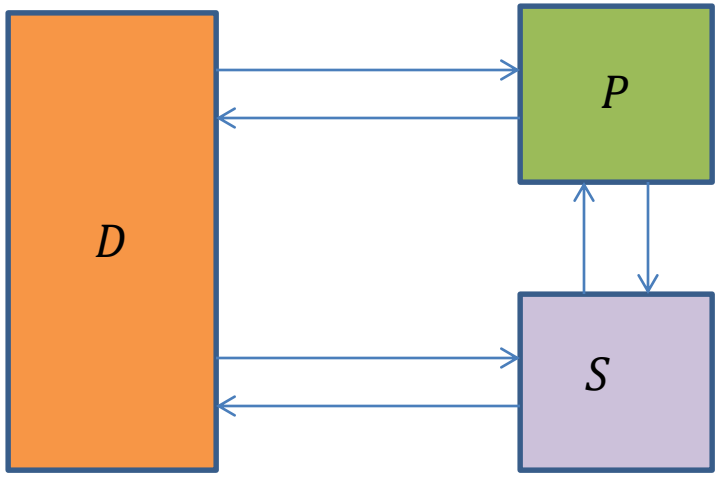
Distinguisher Strategy

- S: On query $F_i(x_i)$, return uniform value



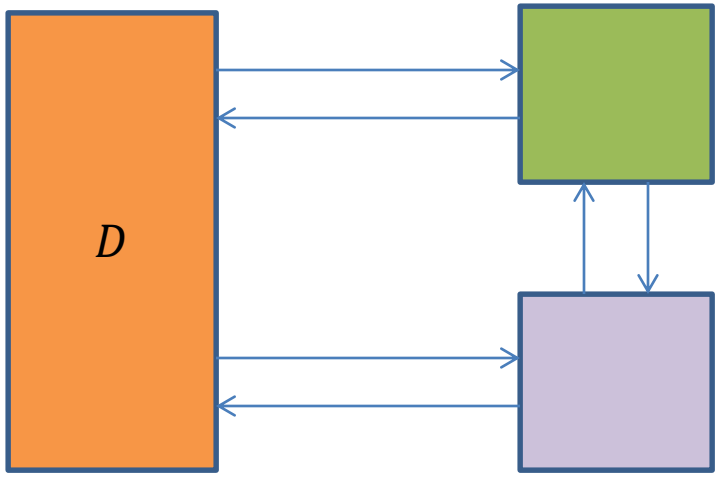
Distinguisher Strategy

- S: On query $F_i(x_i)$, return uniform value



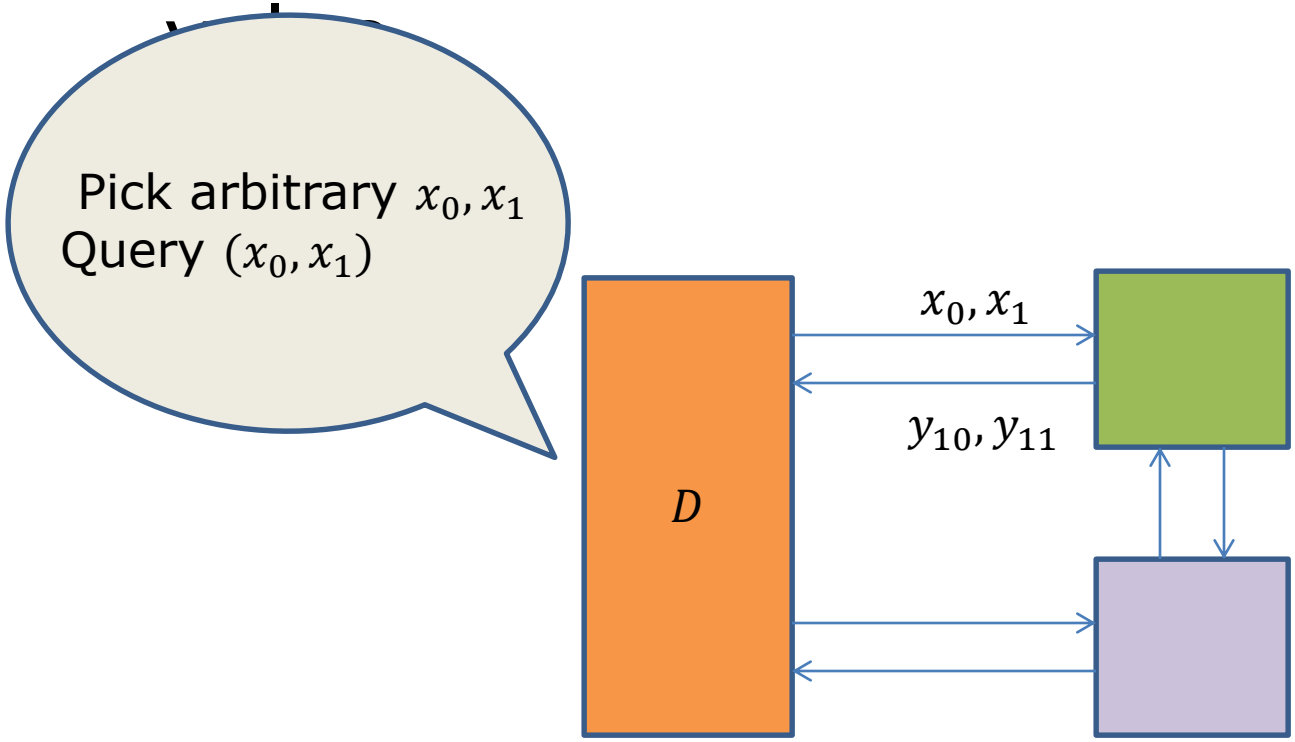
Distinguisher Strategy

- S: On query $F_i(x_i)$, return uniform value



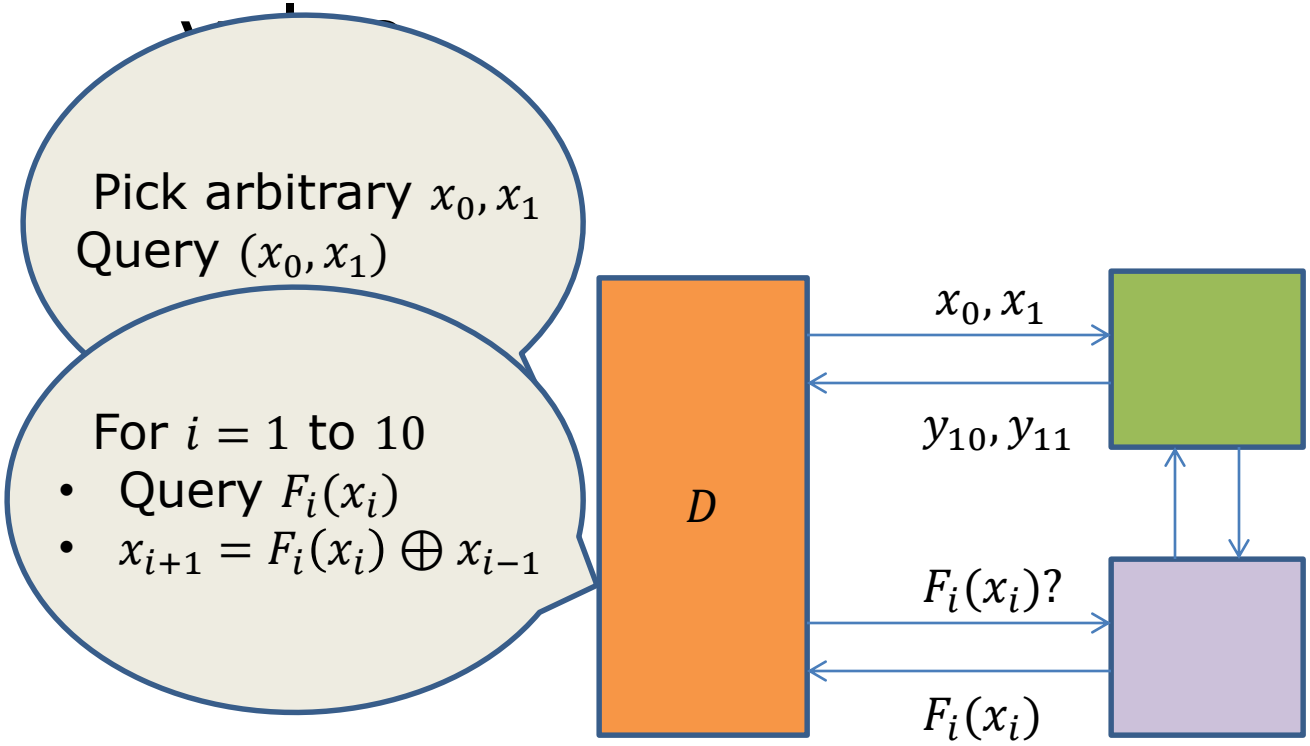
Distinguisher Strategy

- S: On query $F_i(x_i)$, return uniform



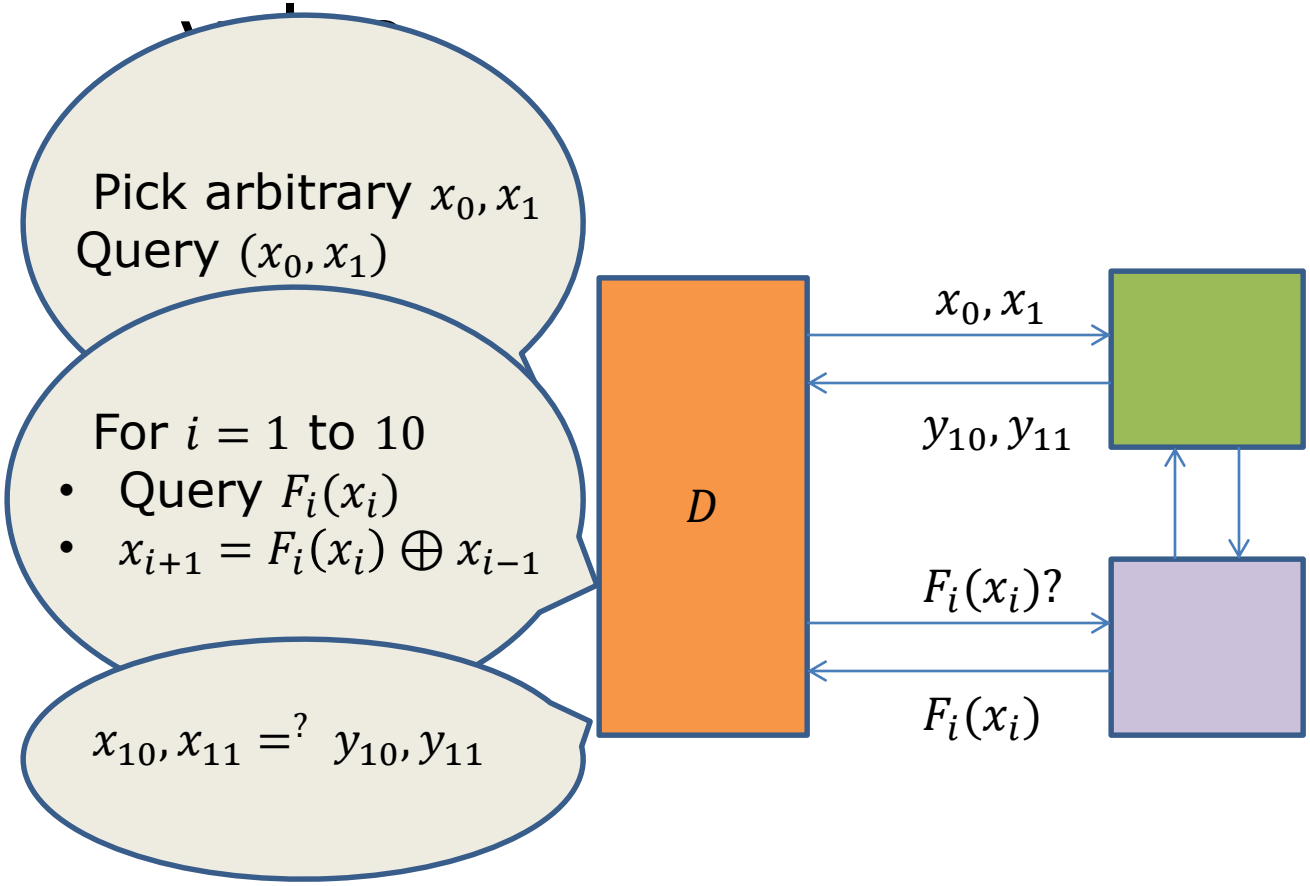
Distinguisher Strategy

- S: On query $F_i(x_i)$, return uniform



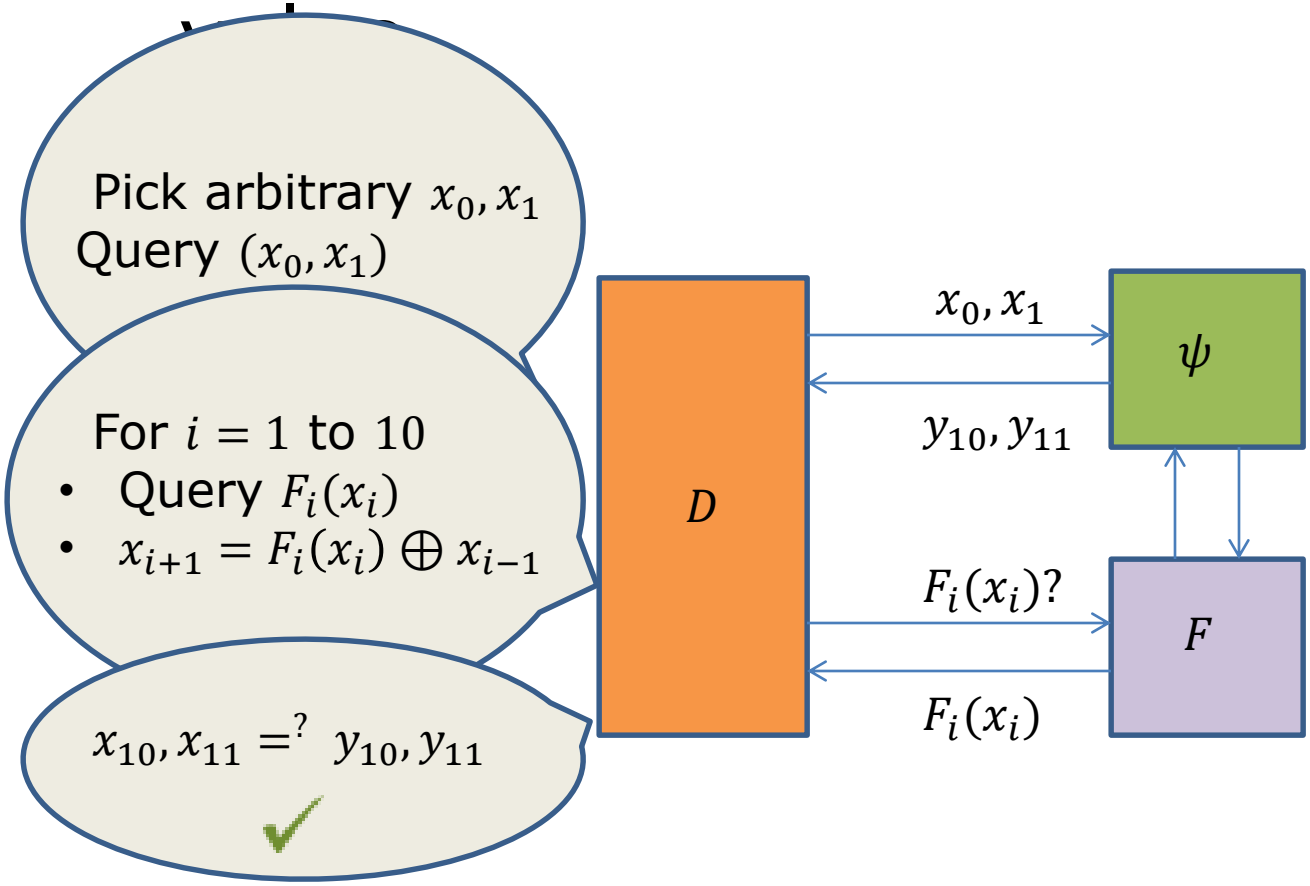
Distinguisher Strategy

- S: On query $F_i(x_i)$, return uniform



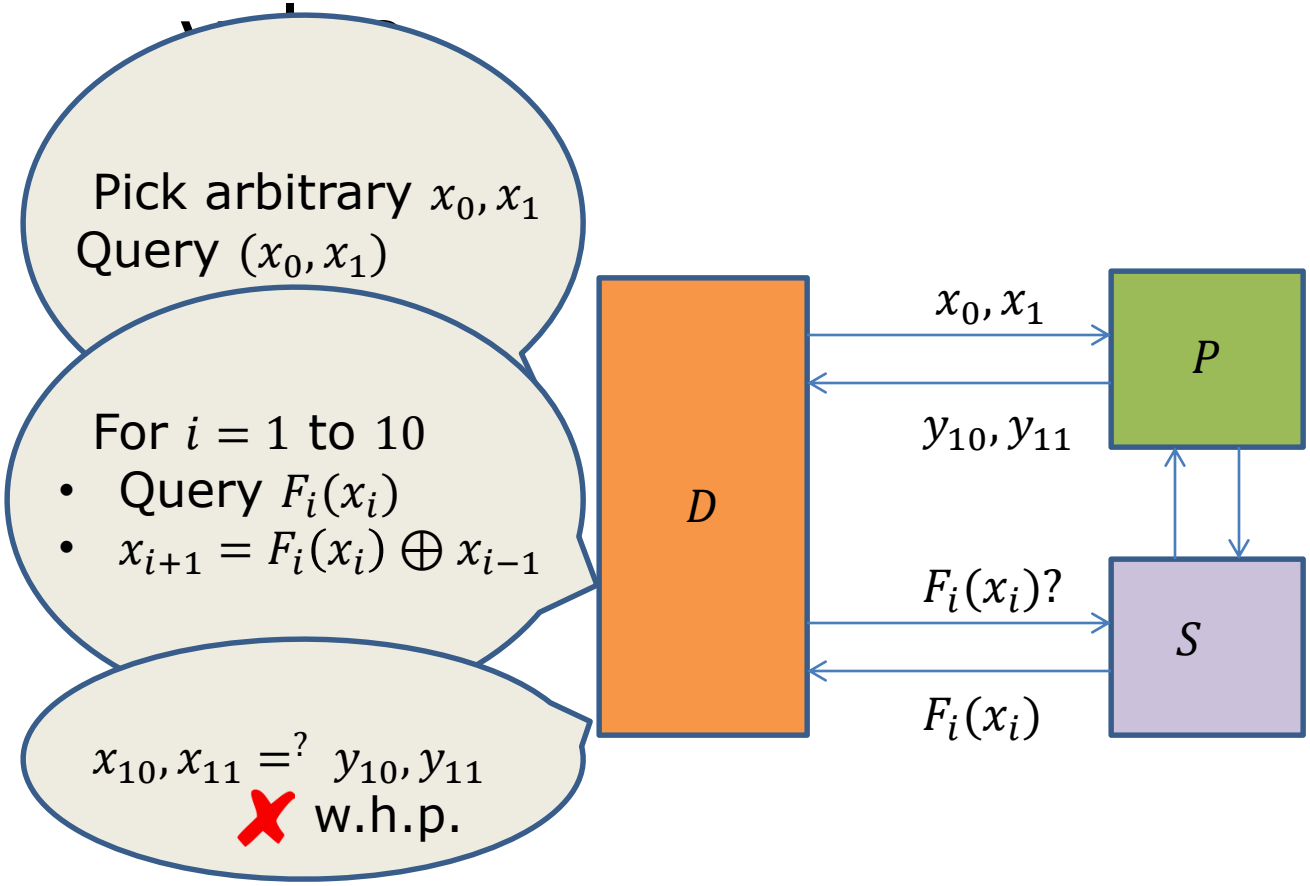
Distinguisher Strategy

- S: On query $F_i(x_i)$, return uniform

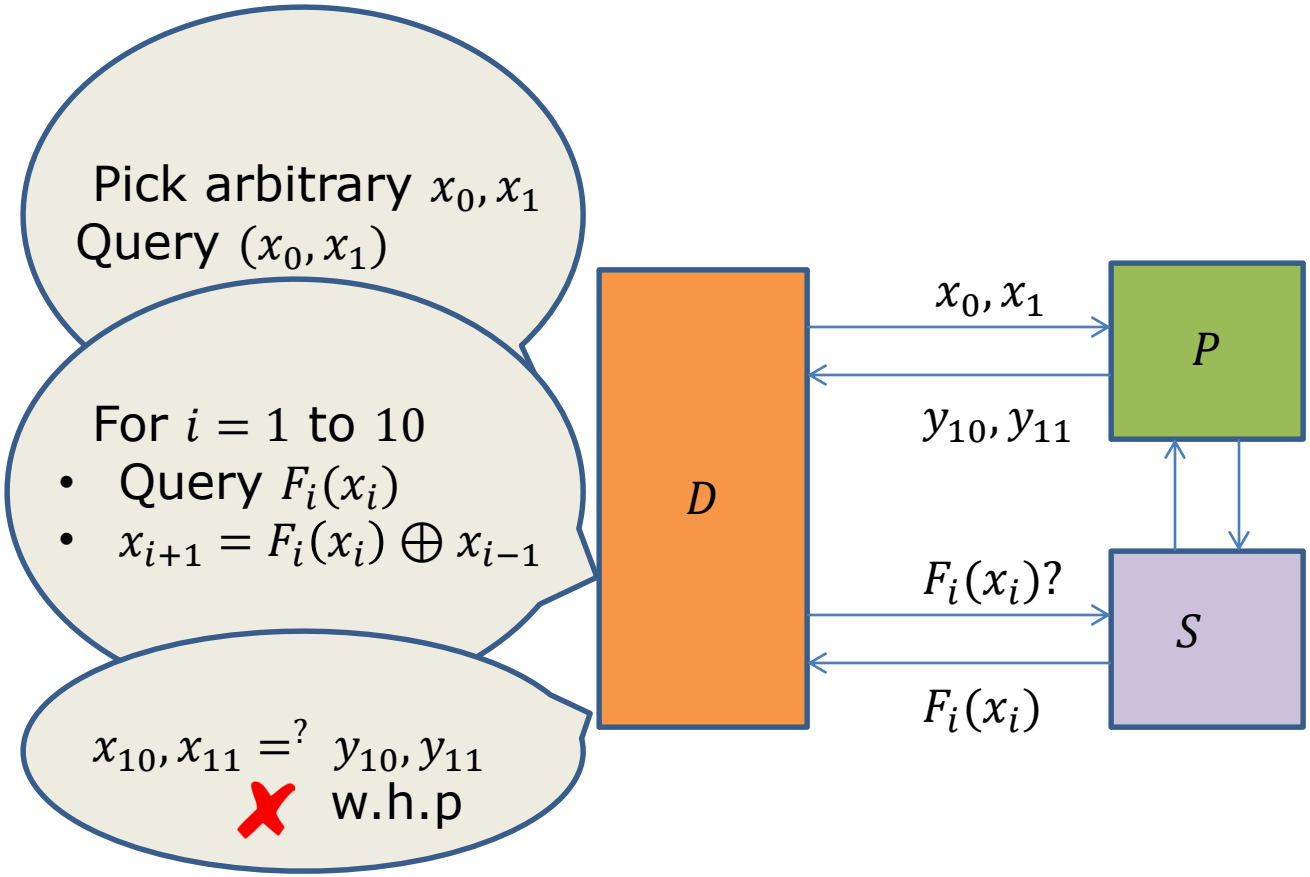


Distinguisher Strategy

- S: On query $F_i(x_i)$, return uniform

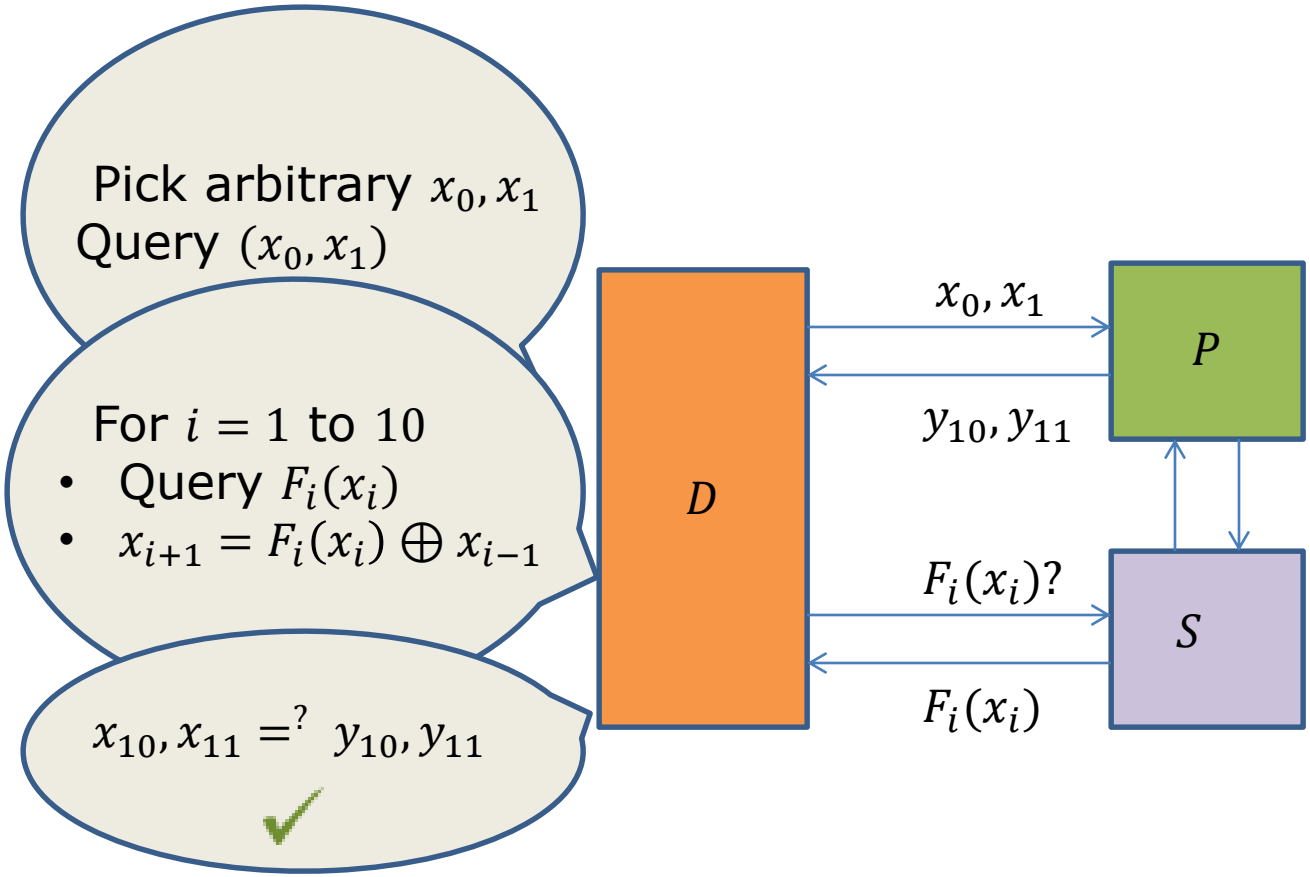


What should Simulator do?



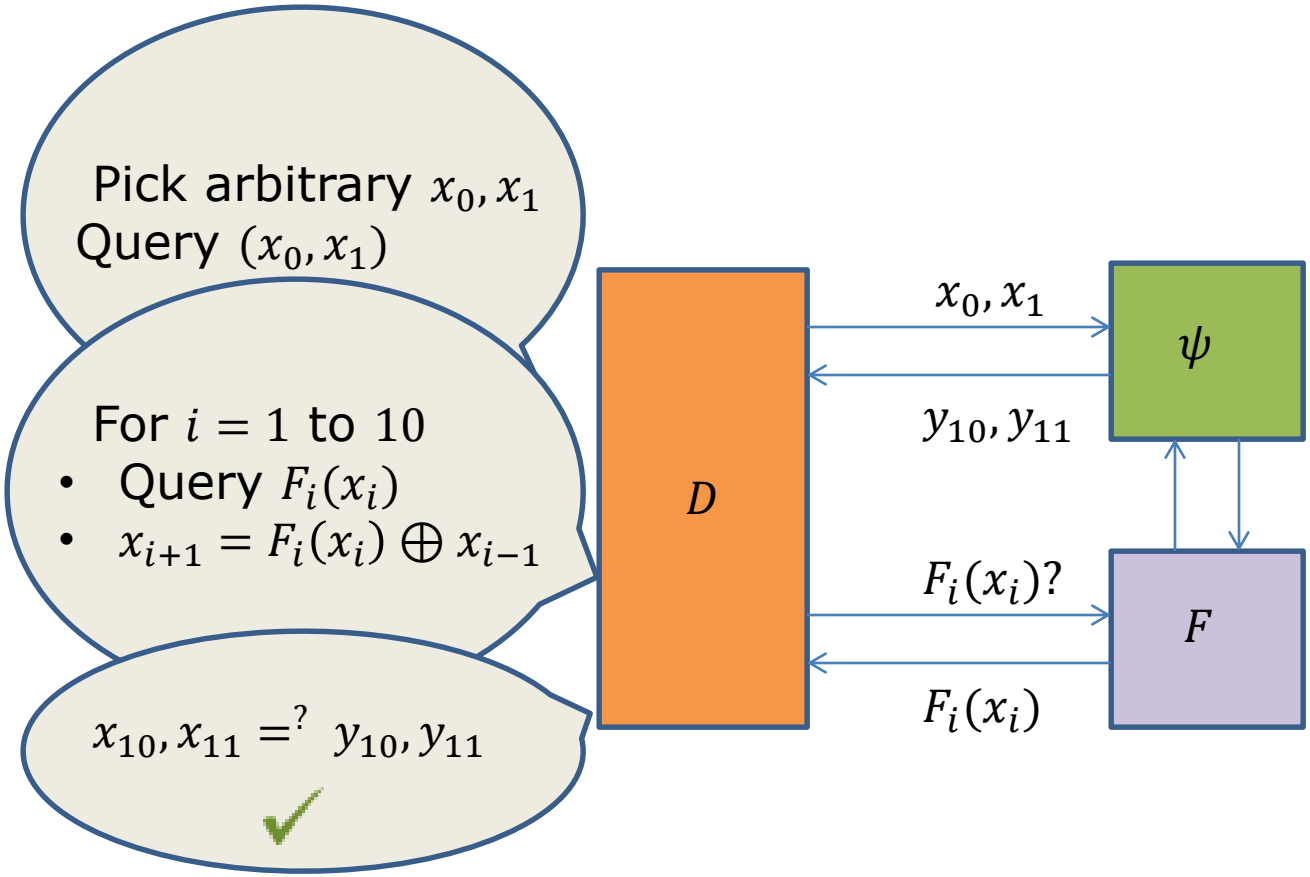
What should Simulator do?

- Make $x_{10}, x_{11} = y_{10}, y_{11}$



What should Simulator do?

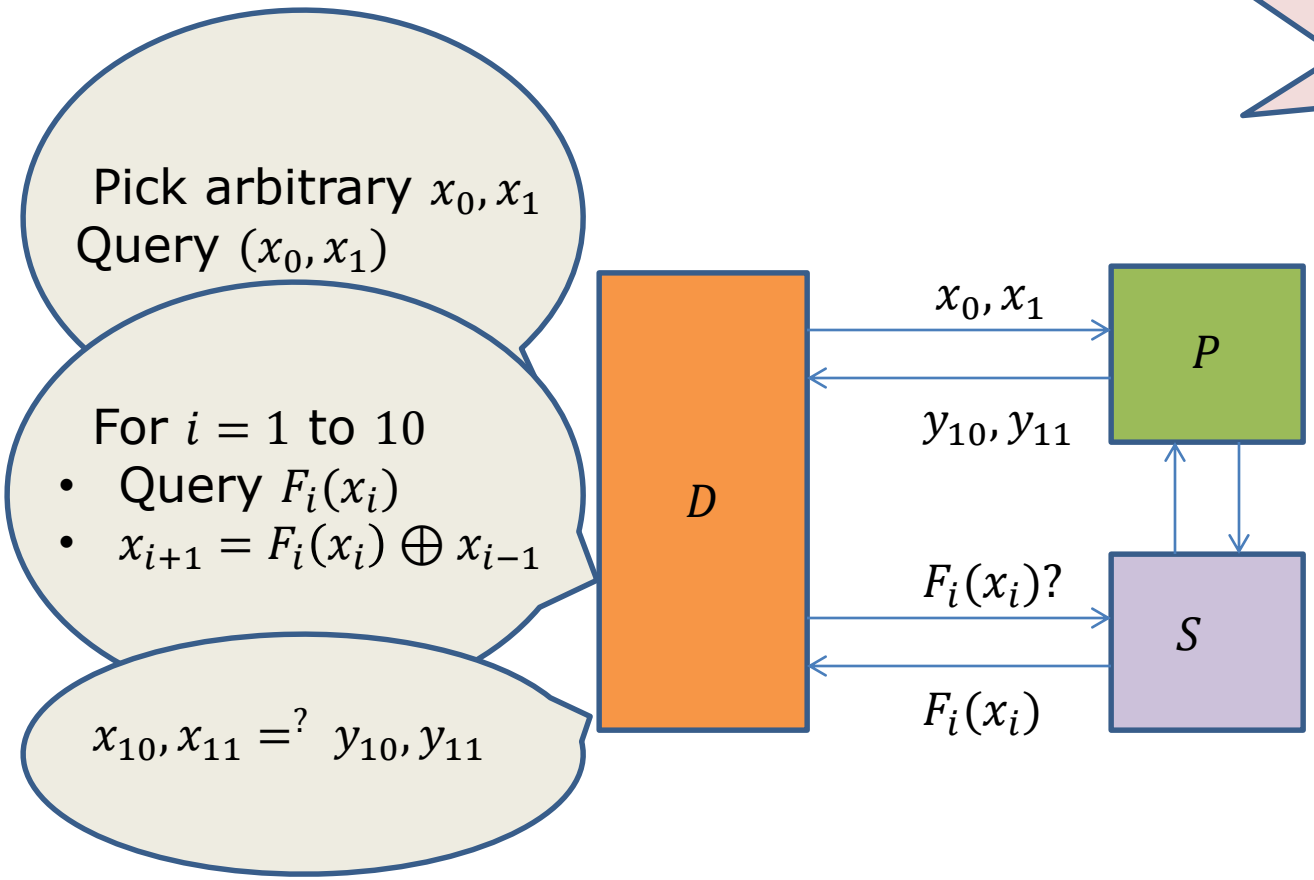
- Make $x_{10}, x_{11} = y_{10}, y_{11}$



What should Simulator do?

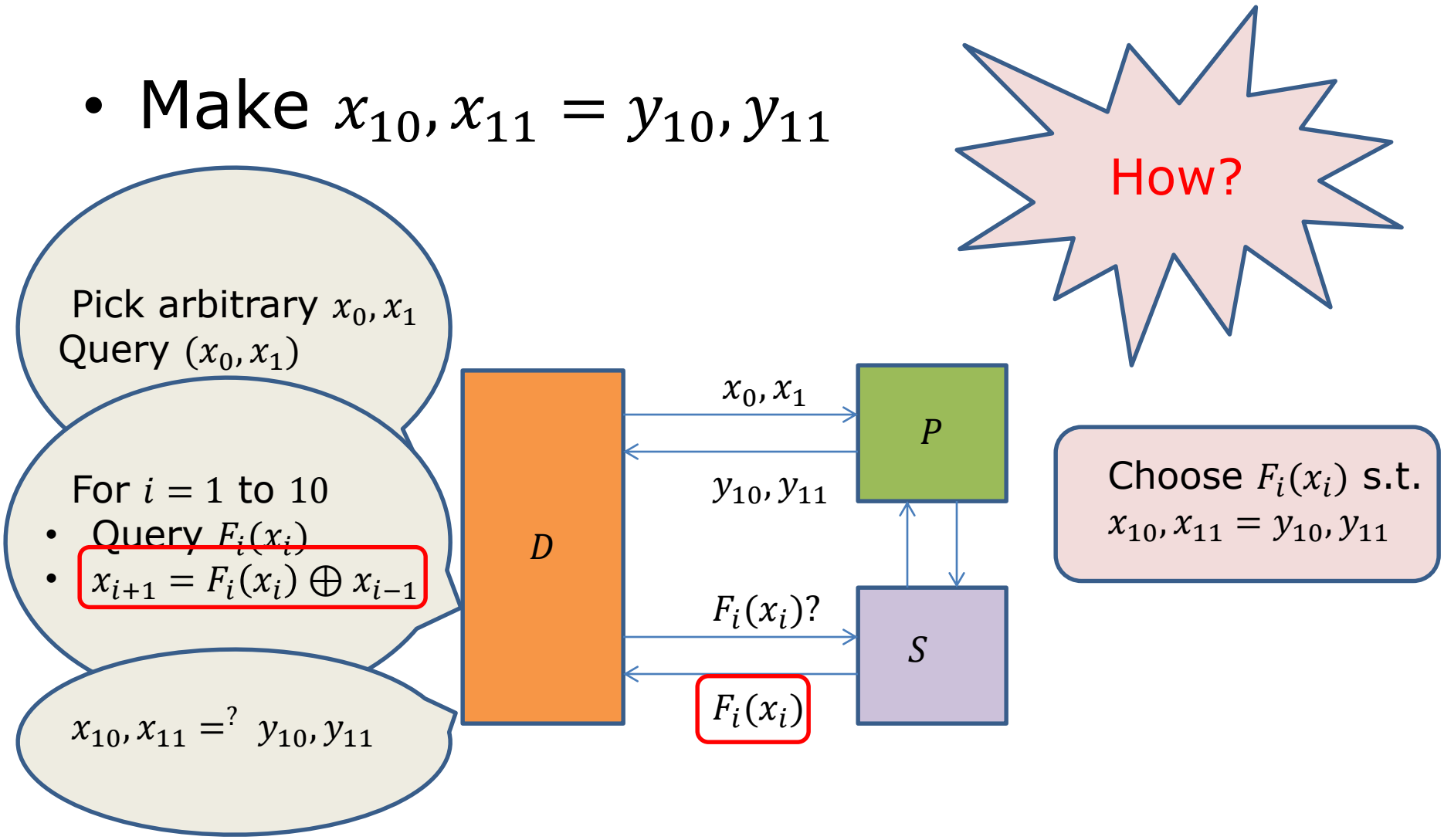
- Make $x_{10}, x_{11} = y_{10}, y_{11}$

How?



What should Simulator do?

- Make $x_{10}, x_{11} = y_{10}, y_{11}$



What should Simulator do?

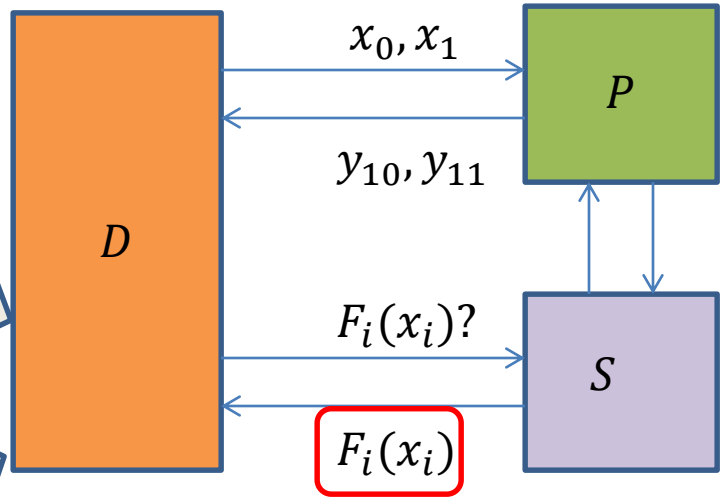
- Make $x_{10}, x_{11} = y_{10}, y_{11}$

How?

Pick arbitrary x_0, x_1
Query (x_0, x_1)

For $i = 1$ to 10
- Query $F_i(x_i)$
- $x_{i+1} = F_i(x_i) \oplus x_{i-1}$

$x_{10}, x_{11} =^{?} y_{10}, y_{11}$

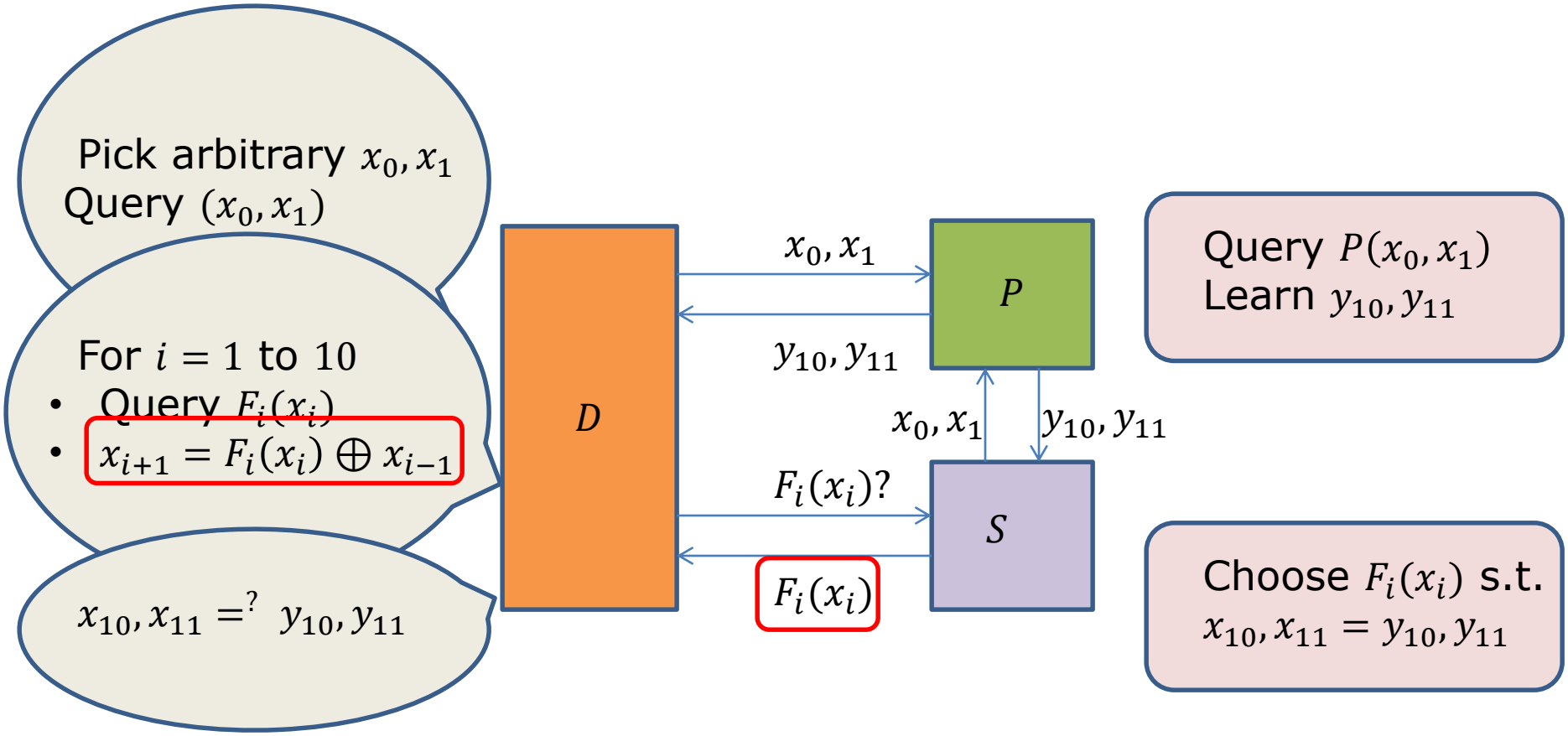


But S does not know y_{10}, y_{11}

Choose $F_i(x_i)$ s.t. $x_{10}, x_{11} = y_{10}, y_{11}$

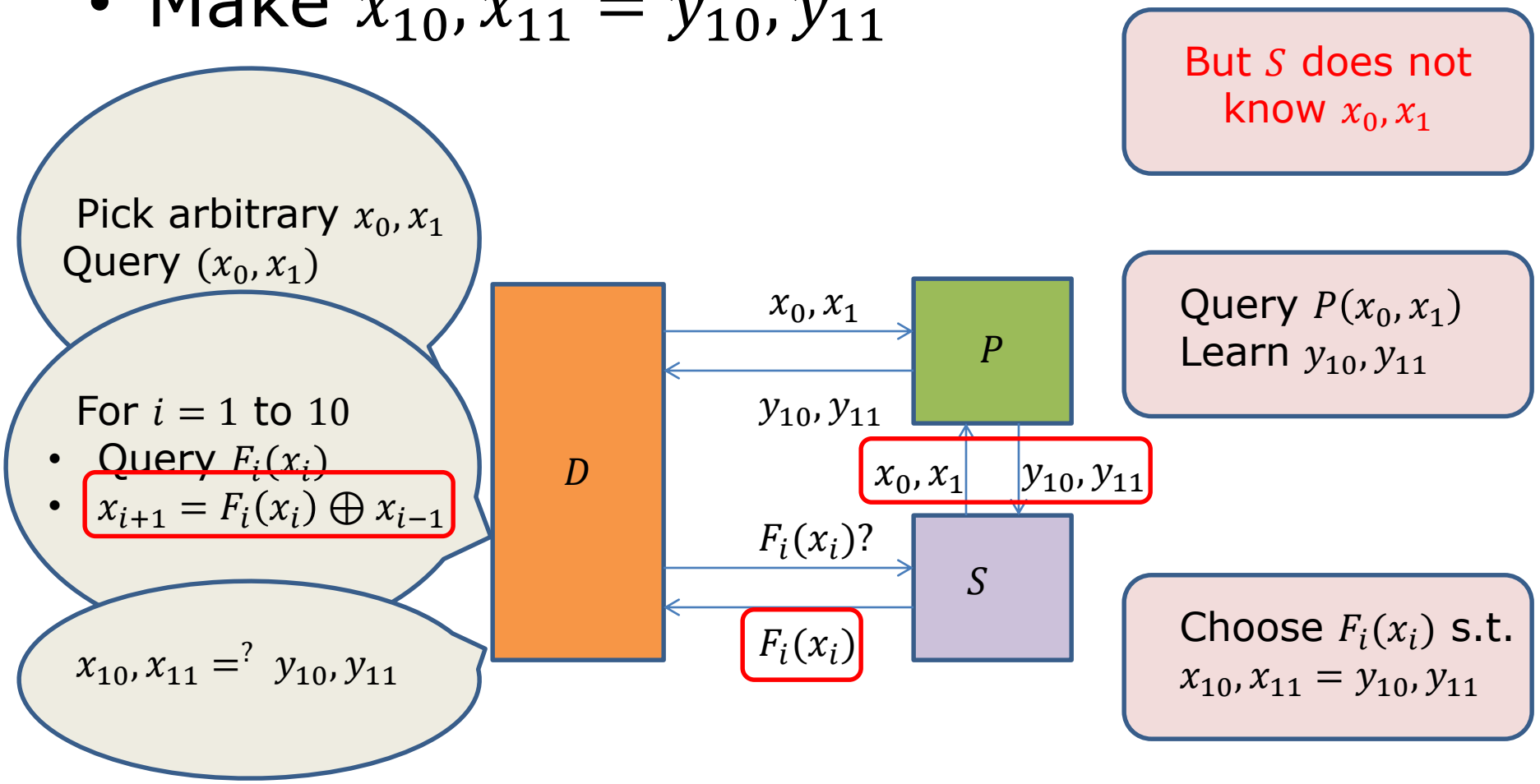
What should Simulator do?

- Make $x_{10}, x_{11} = y_{10}, y_{11}$

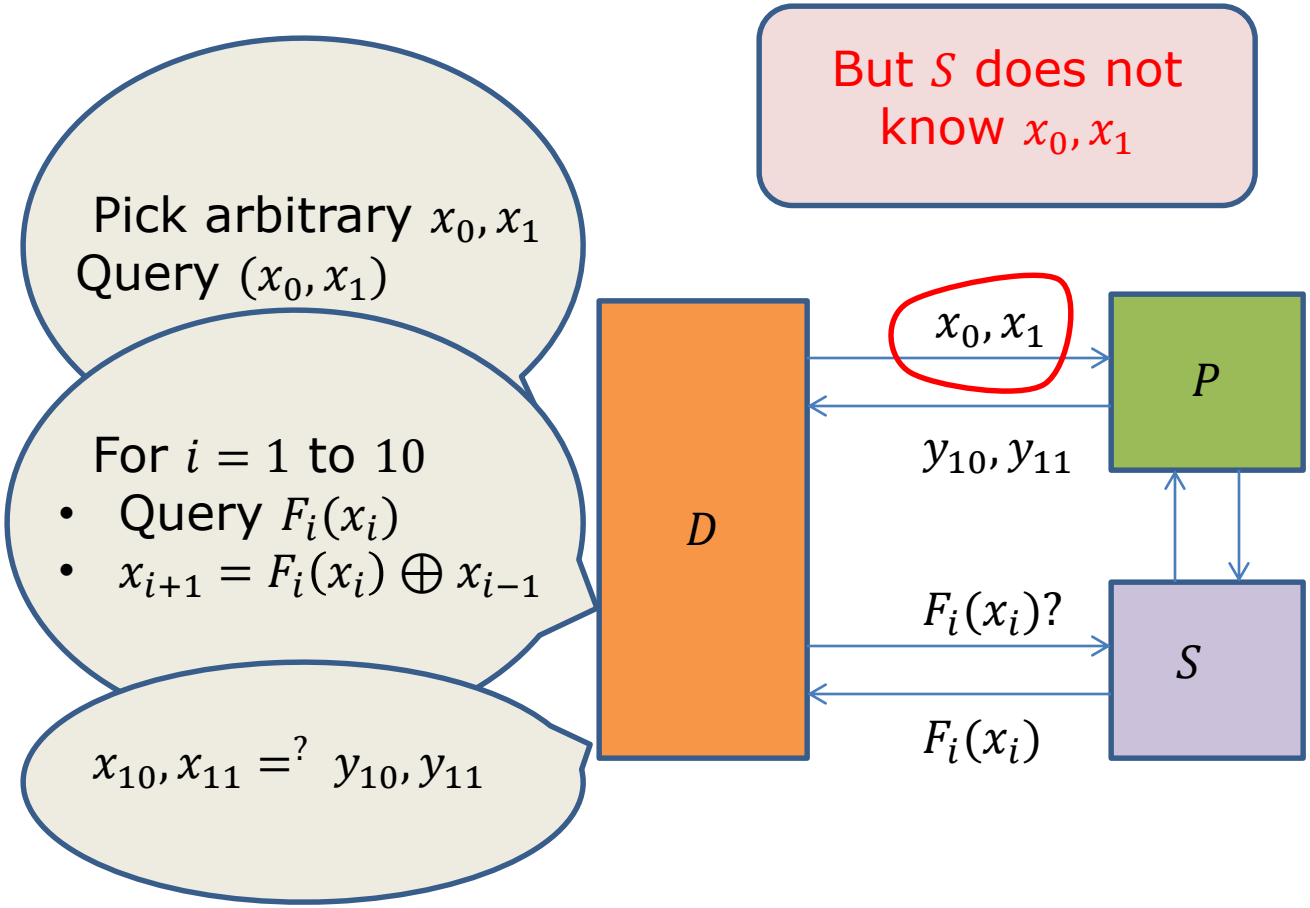


What should Simulator do?

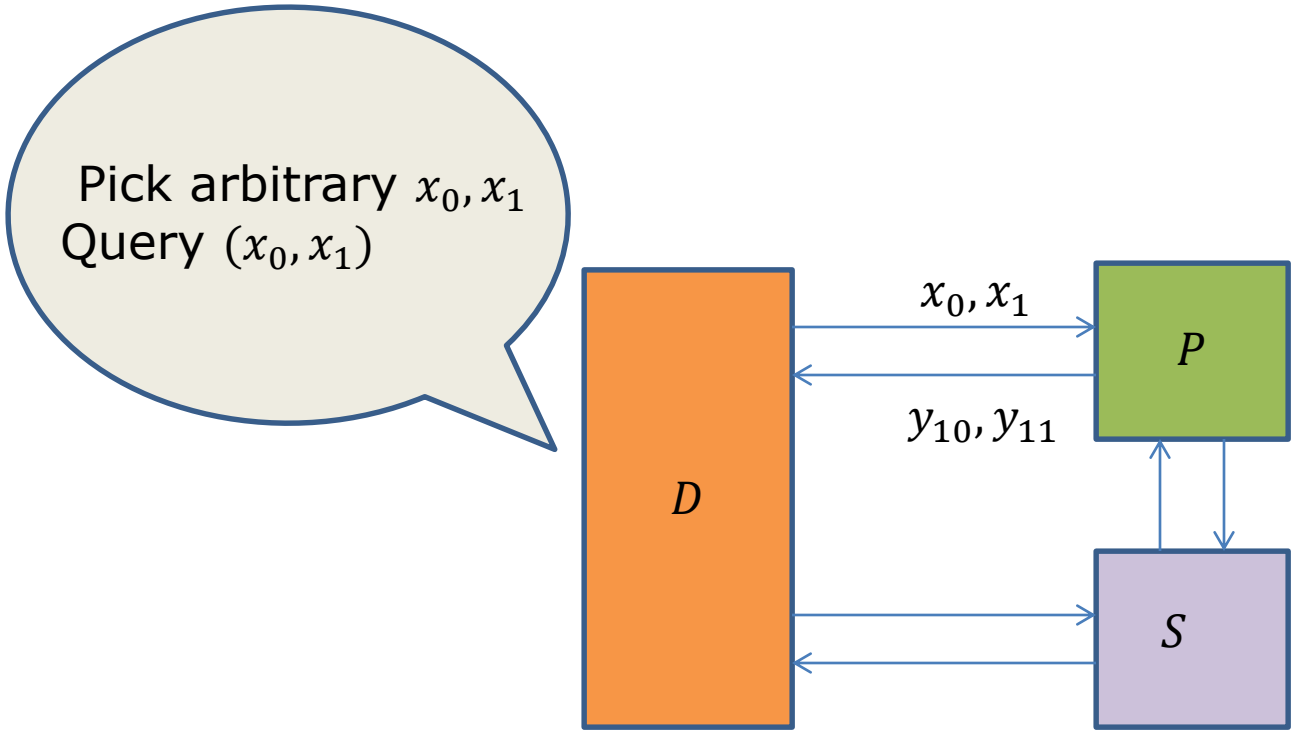
- Make $x_{10}, x_{11} = y_{10}, y_{11}$



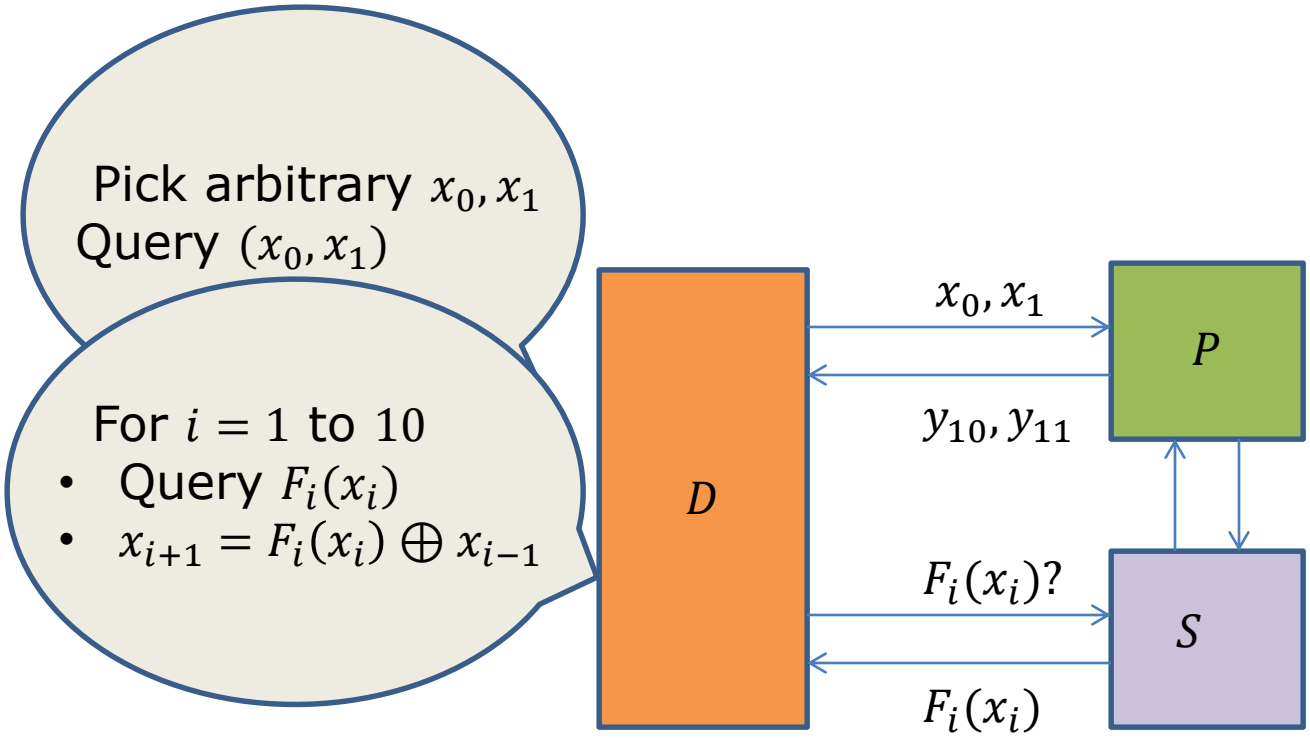
How to learn x_0, x_1?



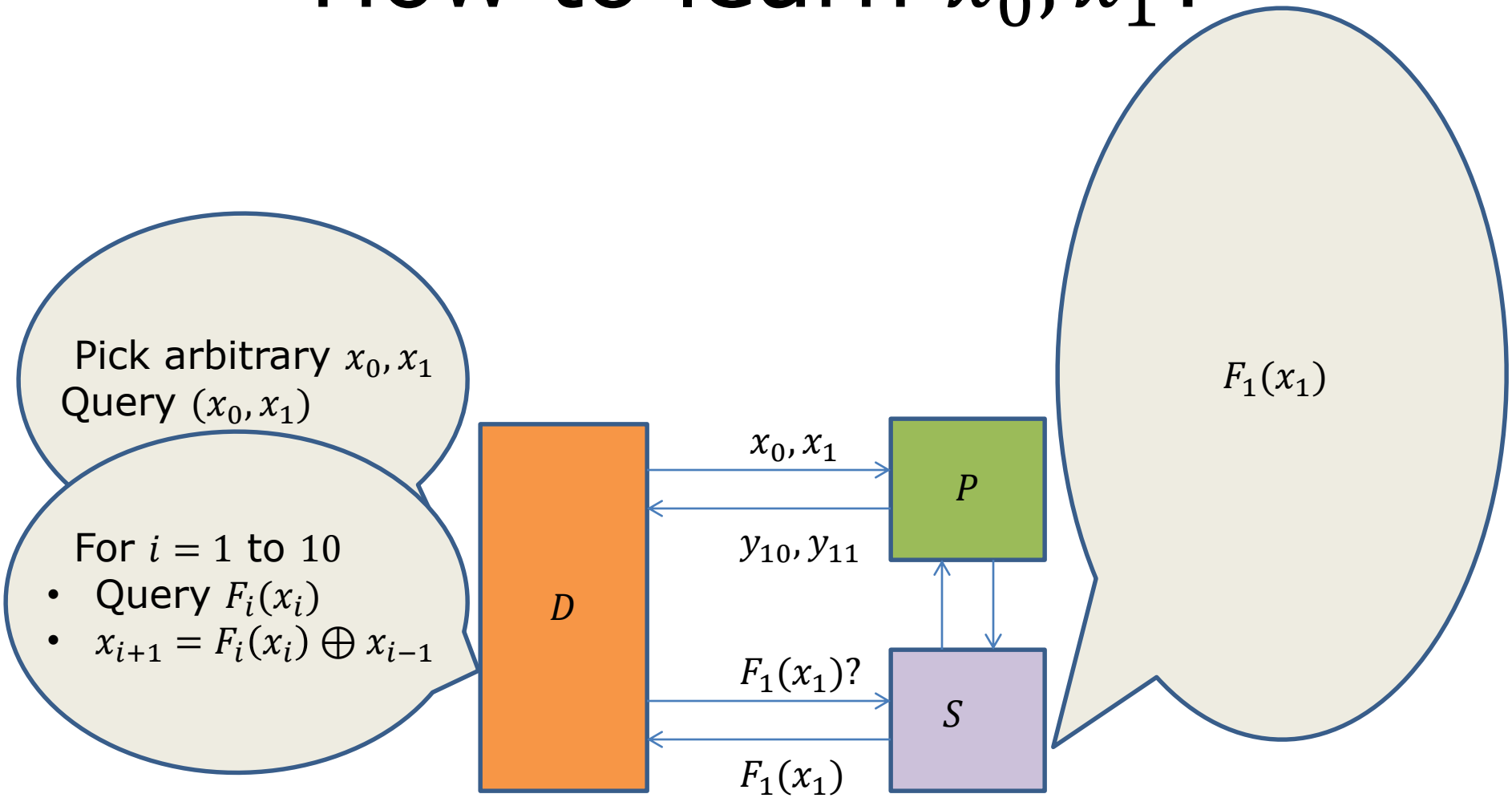
How to learn x_0, x_1?



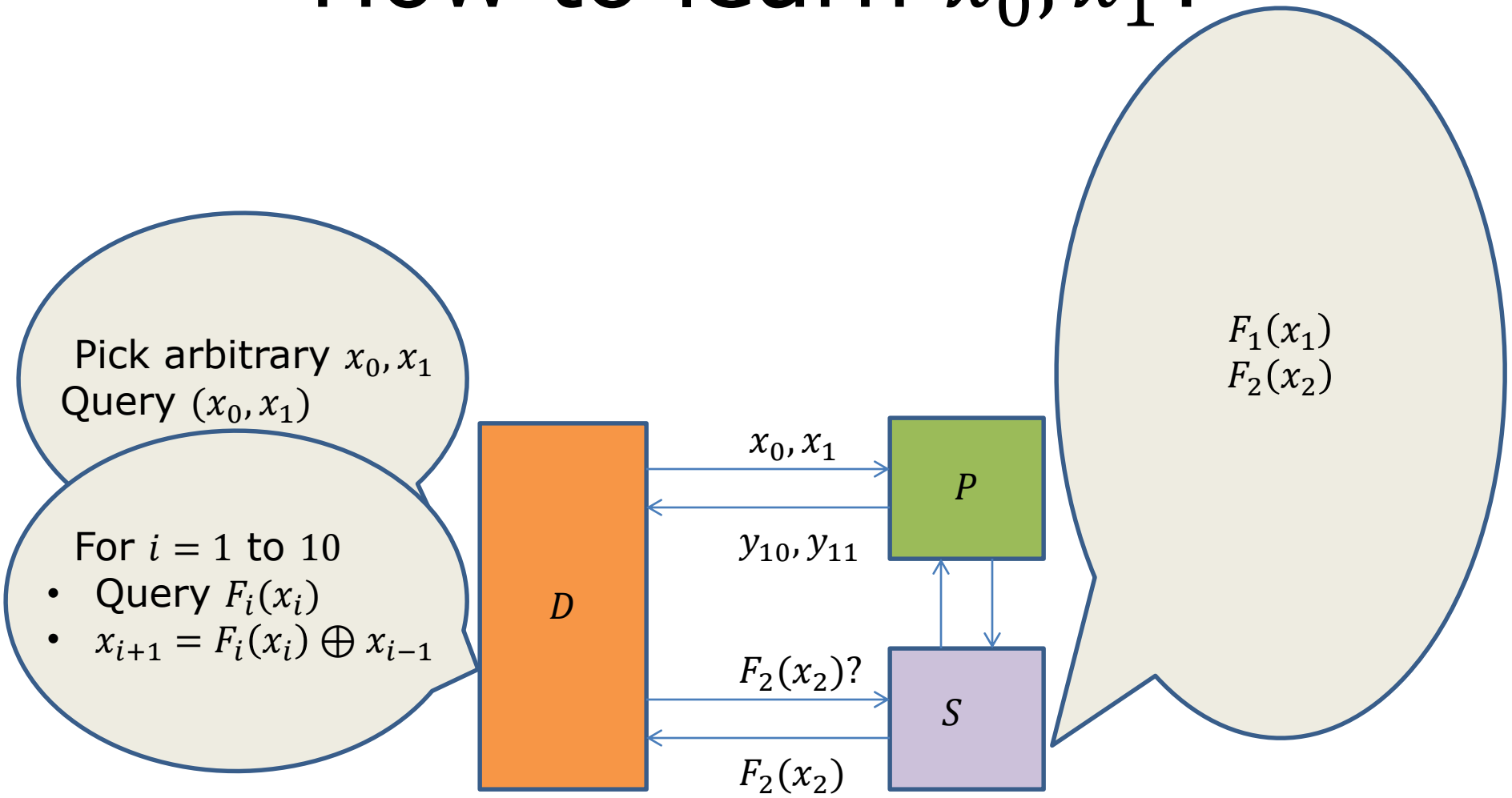
How to learn x_0, x_1?



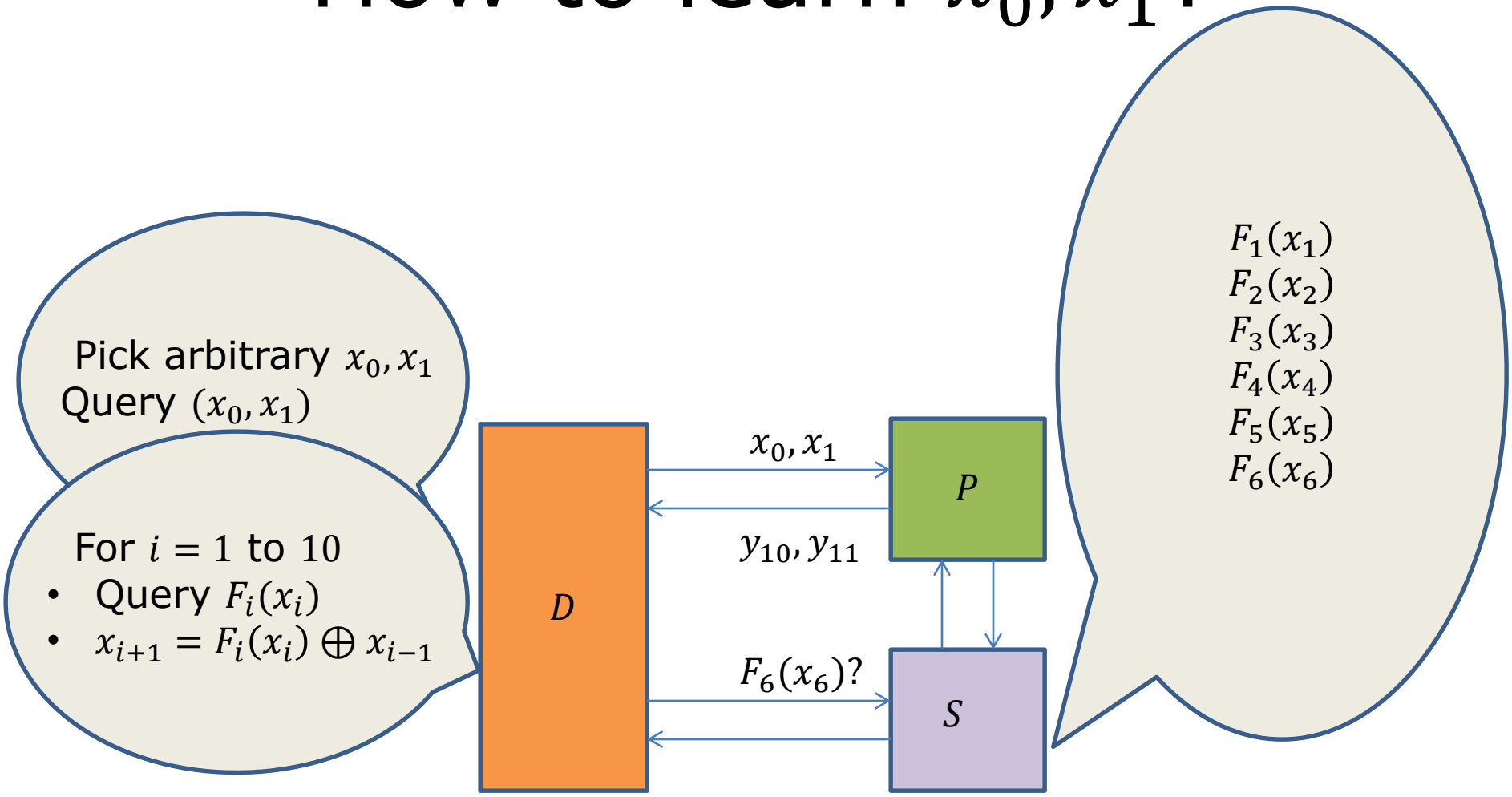
How to learn x_0, x_1?



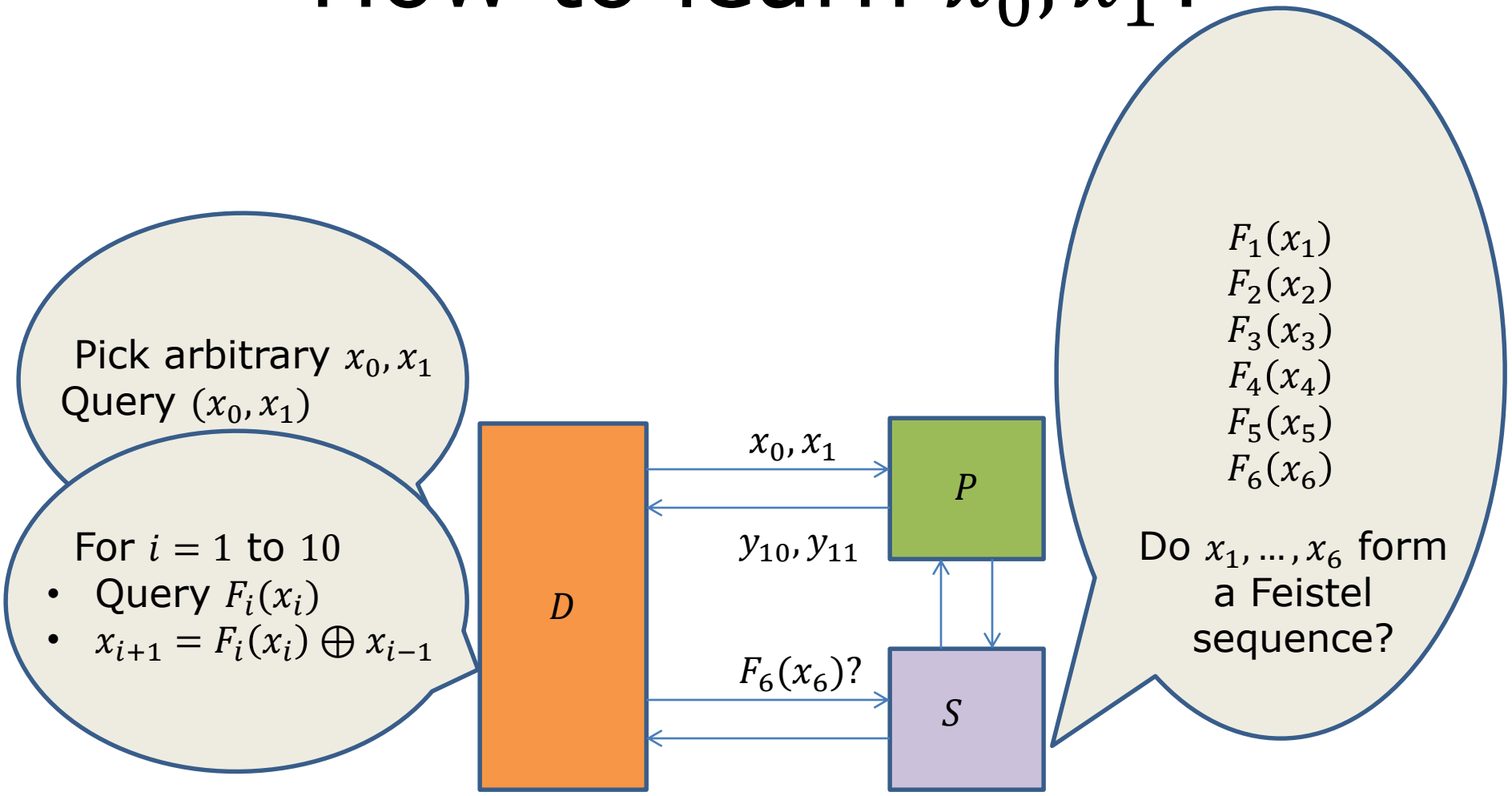
How to learn x_0, x_1?



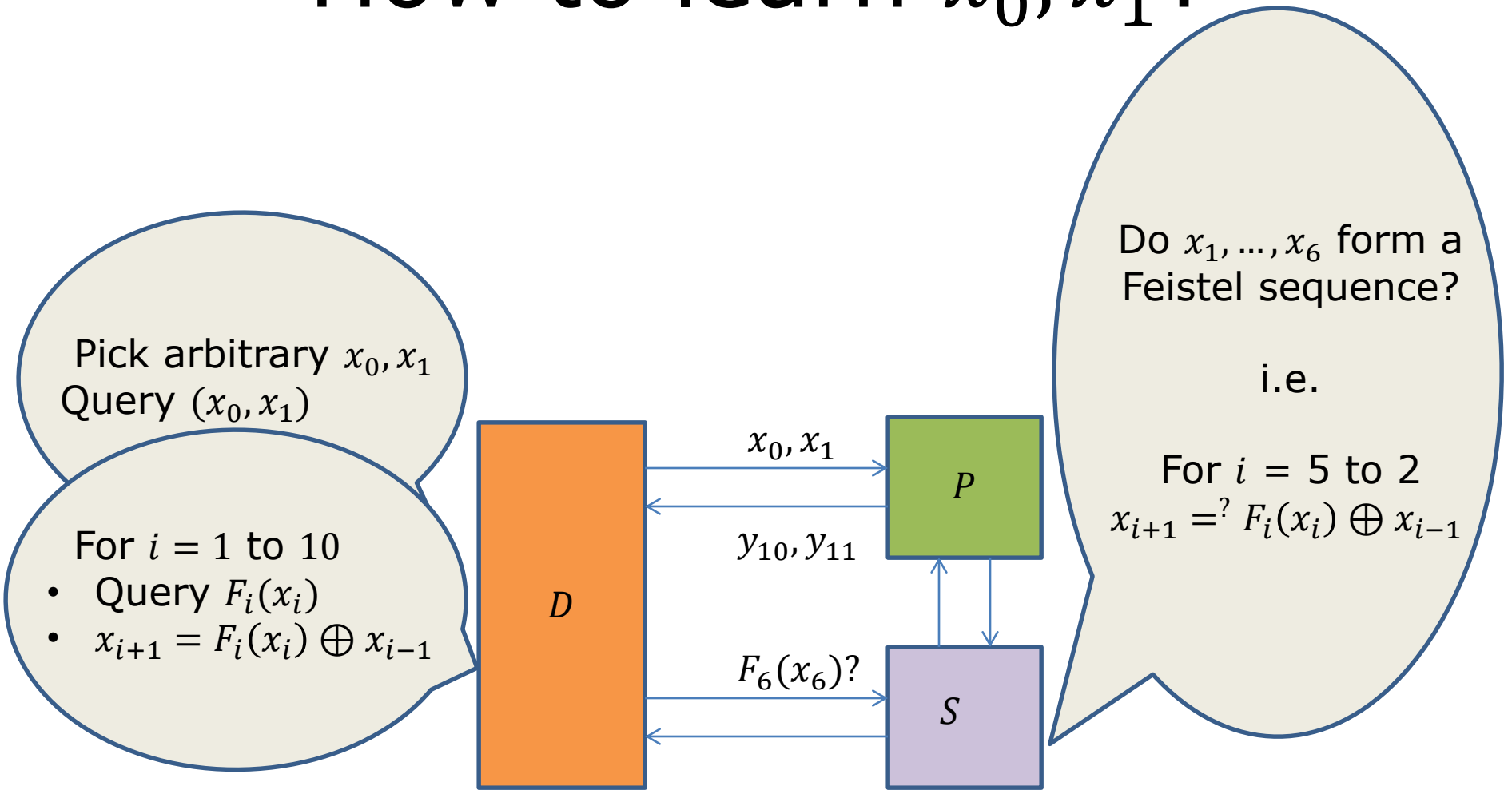
How to learn x_0, x_1?



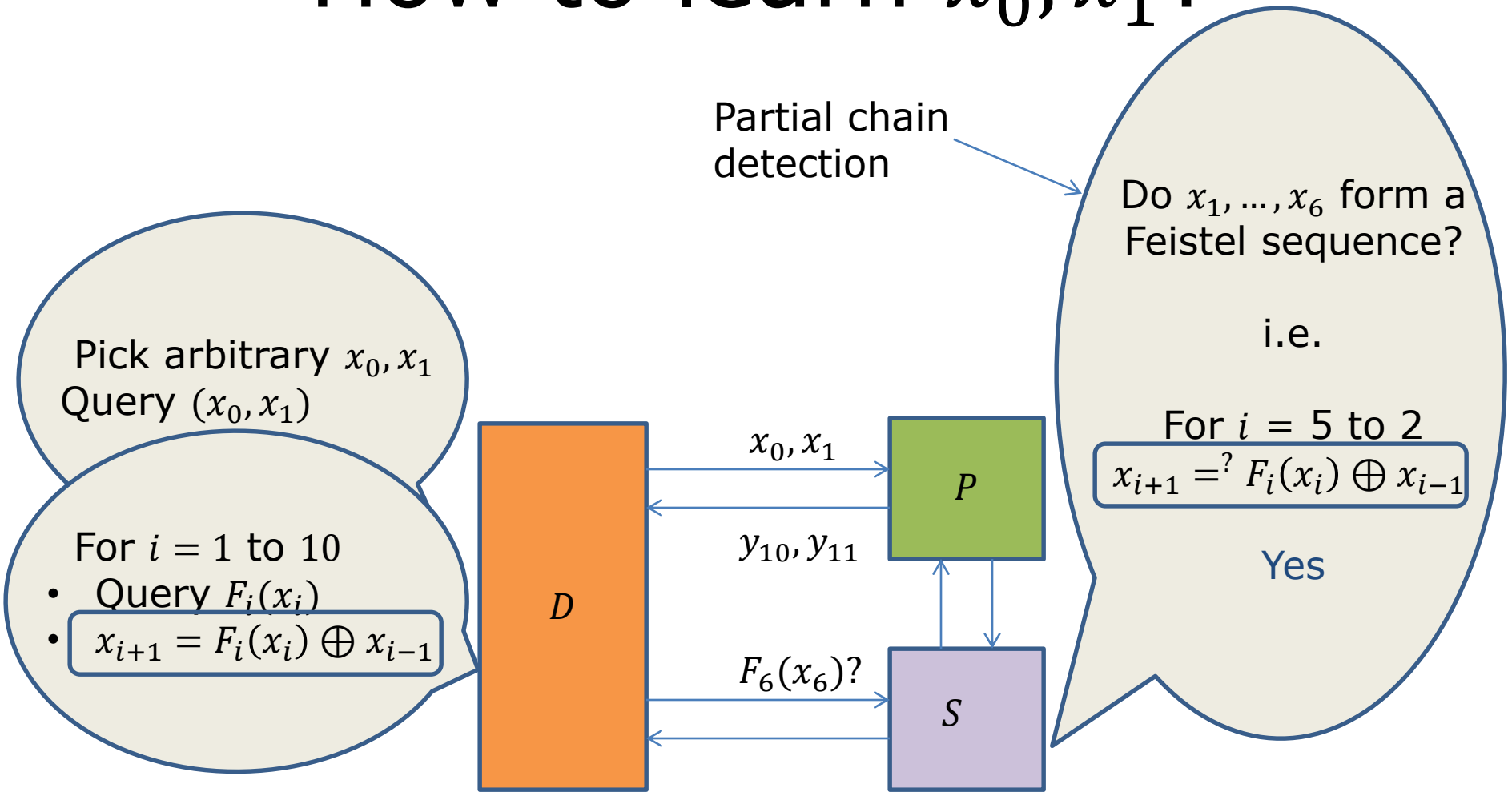
How to learn x_0, x_1?



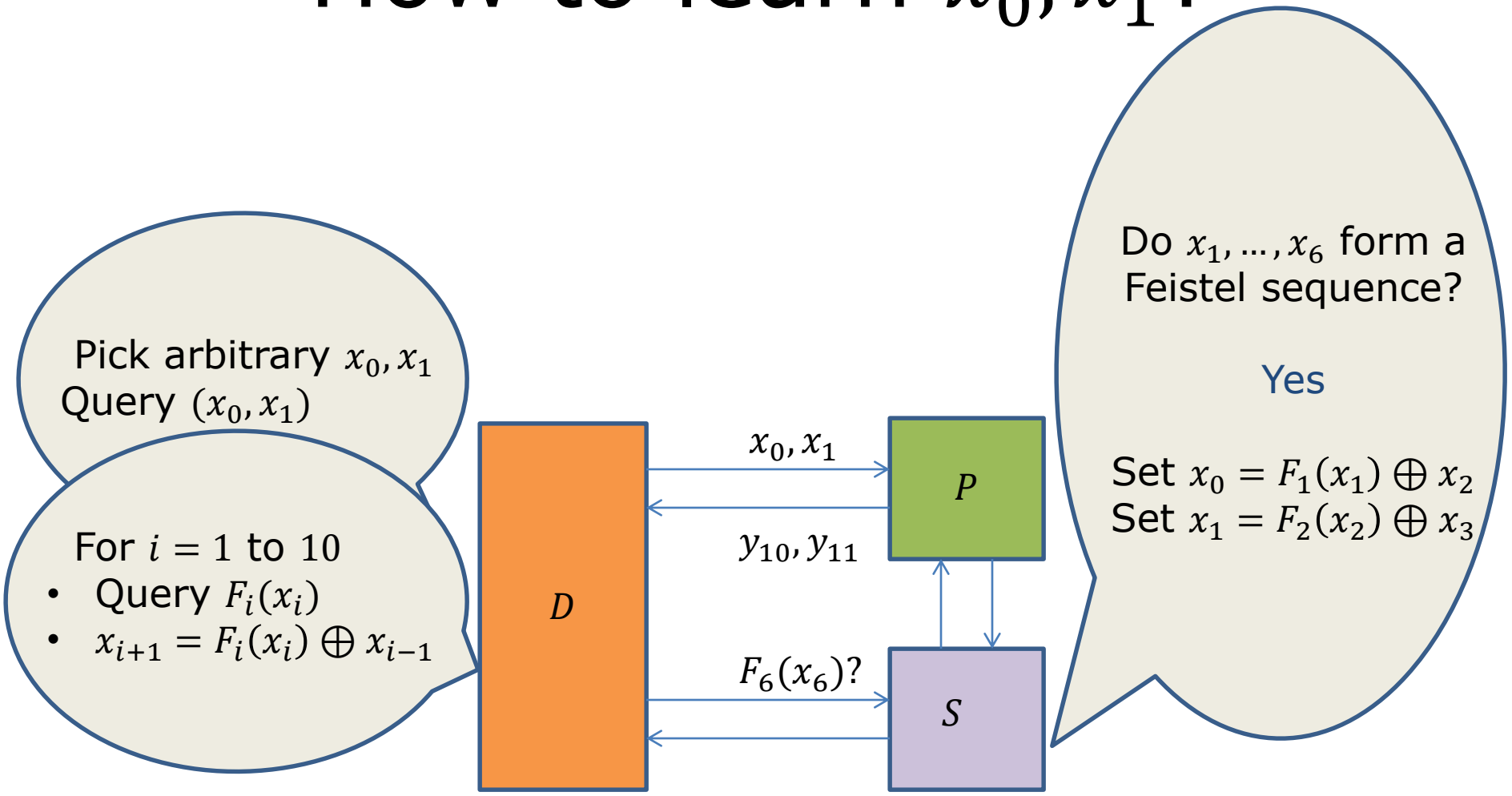
How to learn x_0, x_1?



How to learn x_0, x_1?

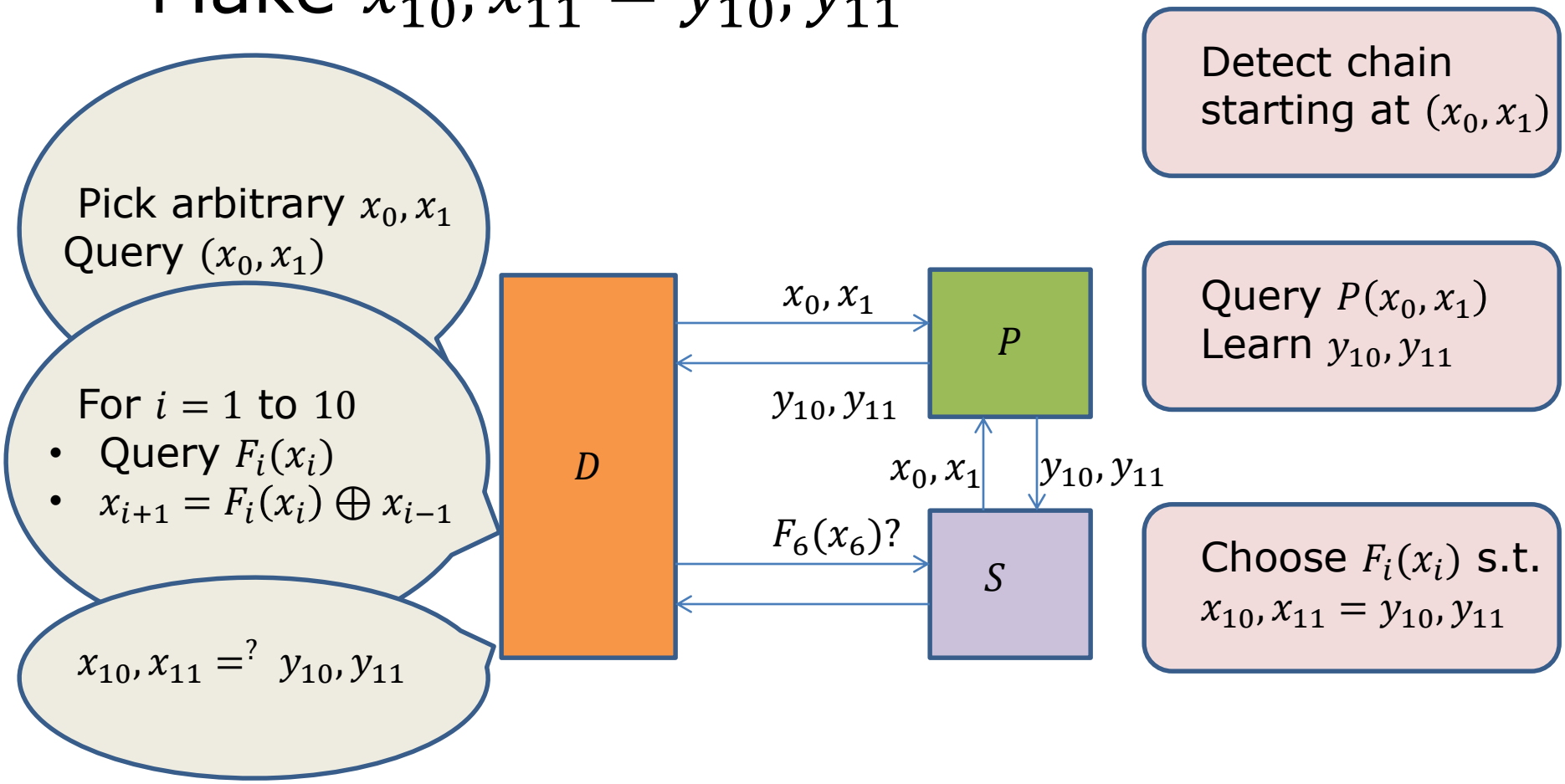


How to learn x_0, x_1?

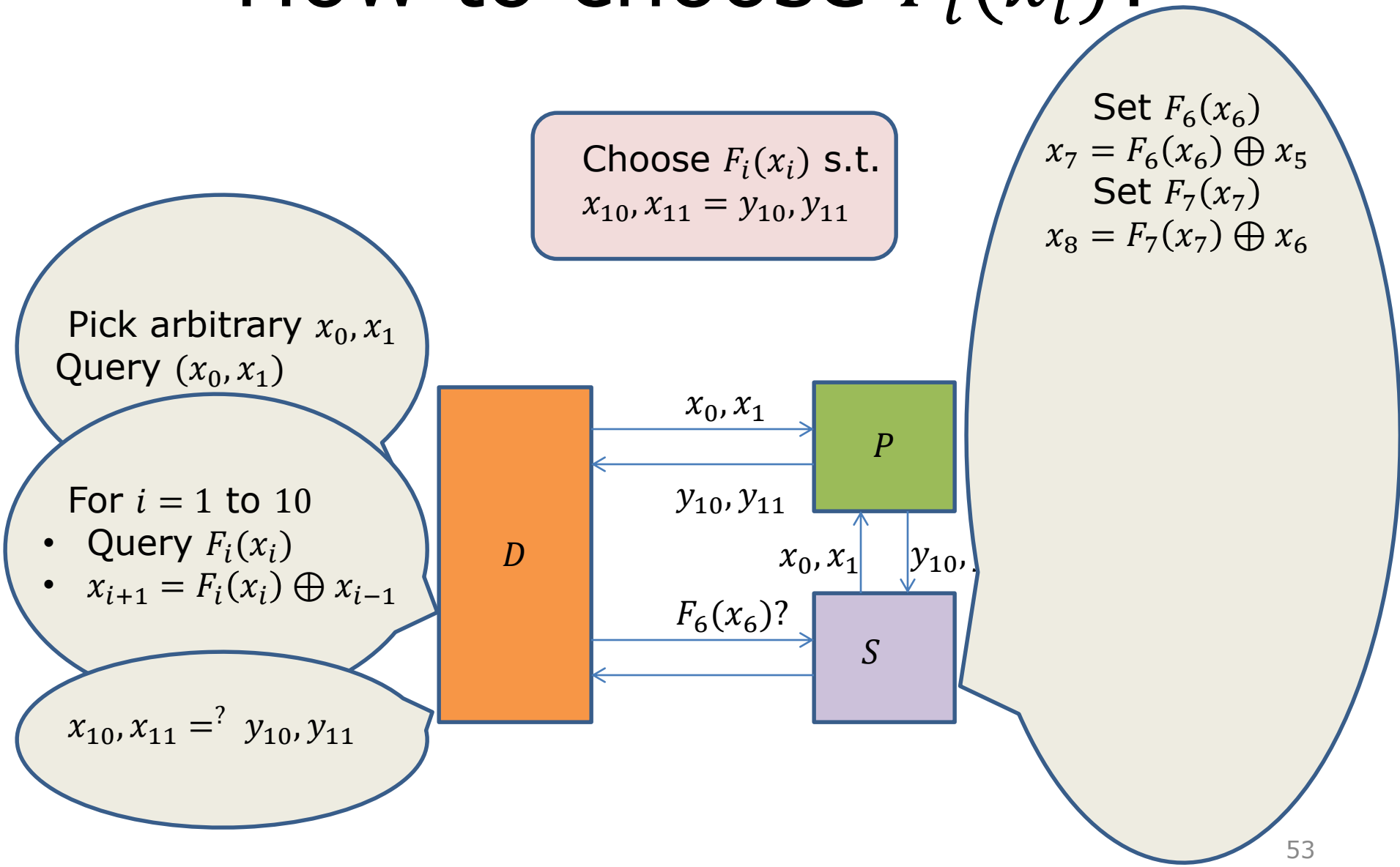


What should Simulator do?

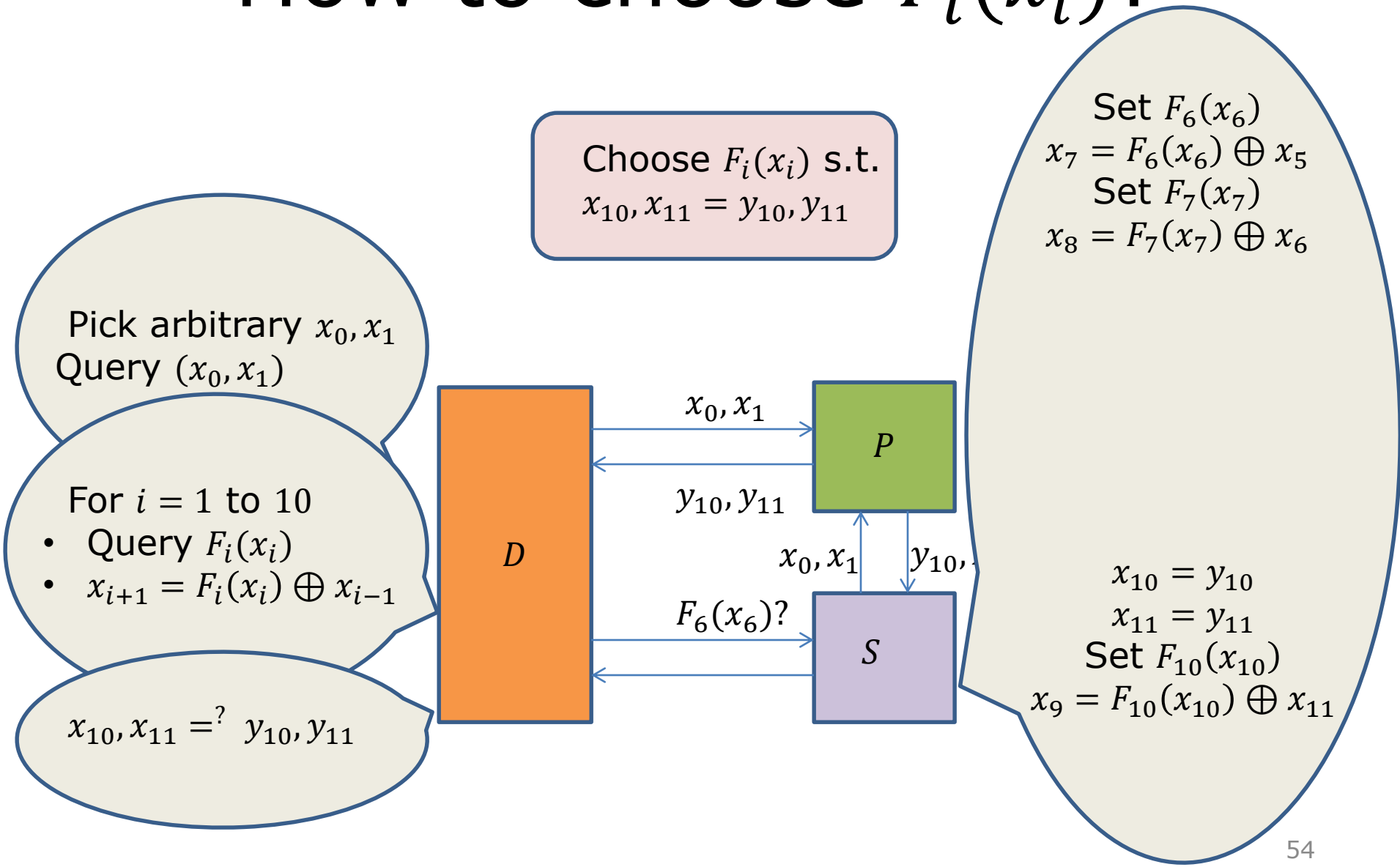
- Make $x_{10}, x_{11} = y_{10}, y_{11}$



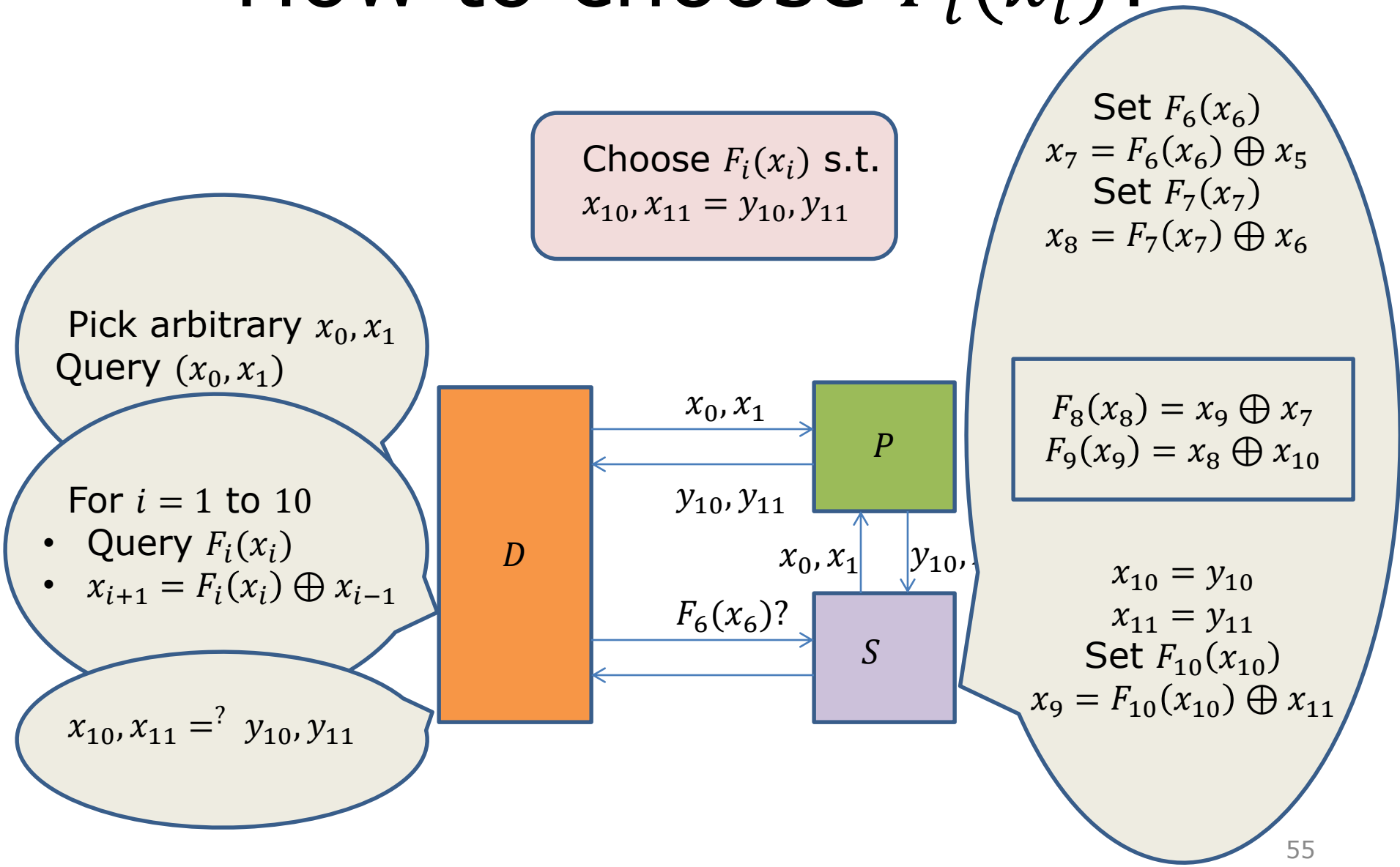
How to choose $F_i(x_i)$?



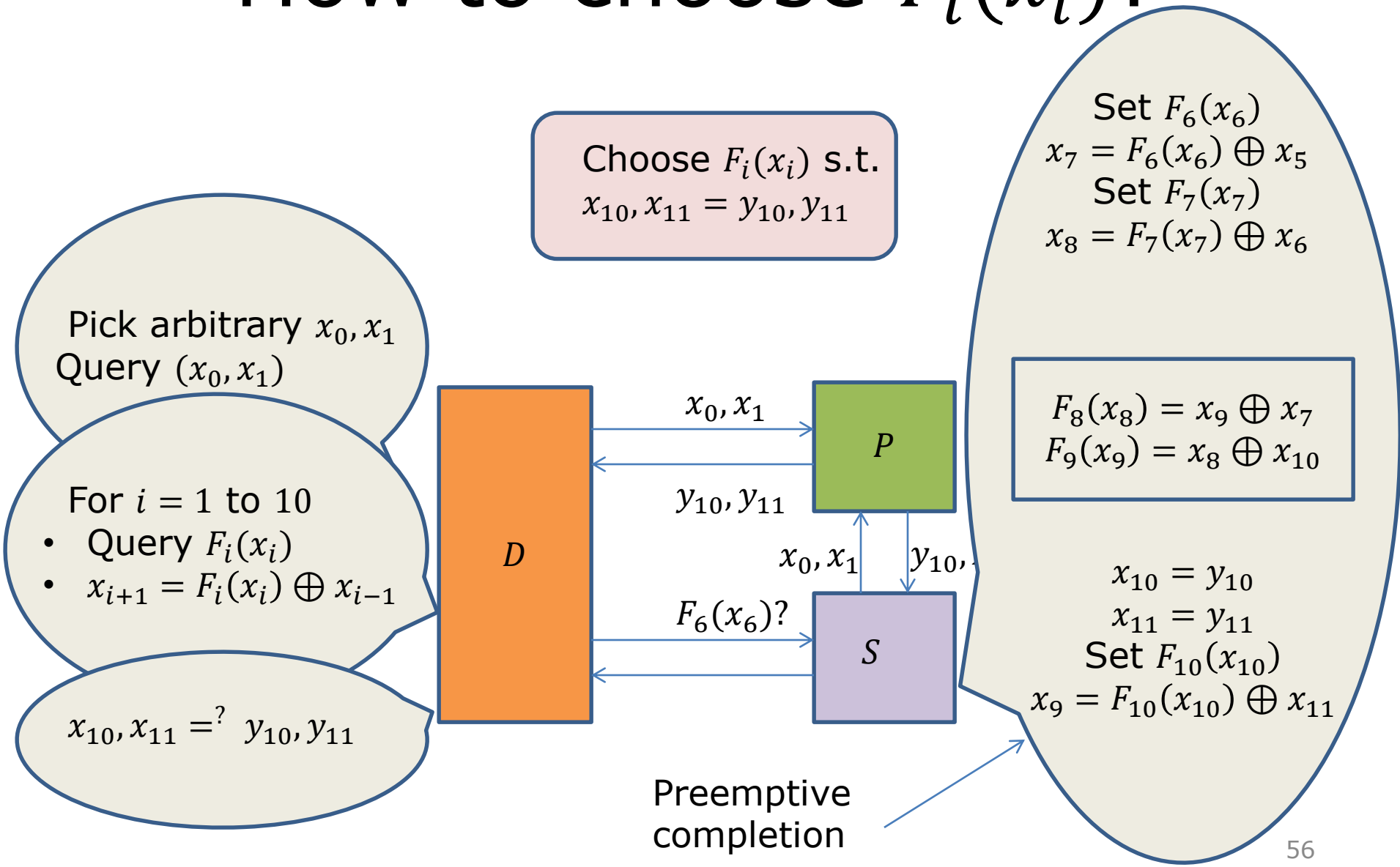
How to choose $F_i(x_i)$?



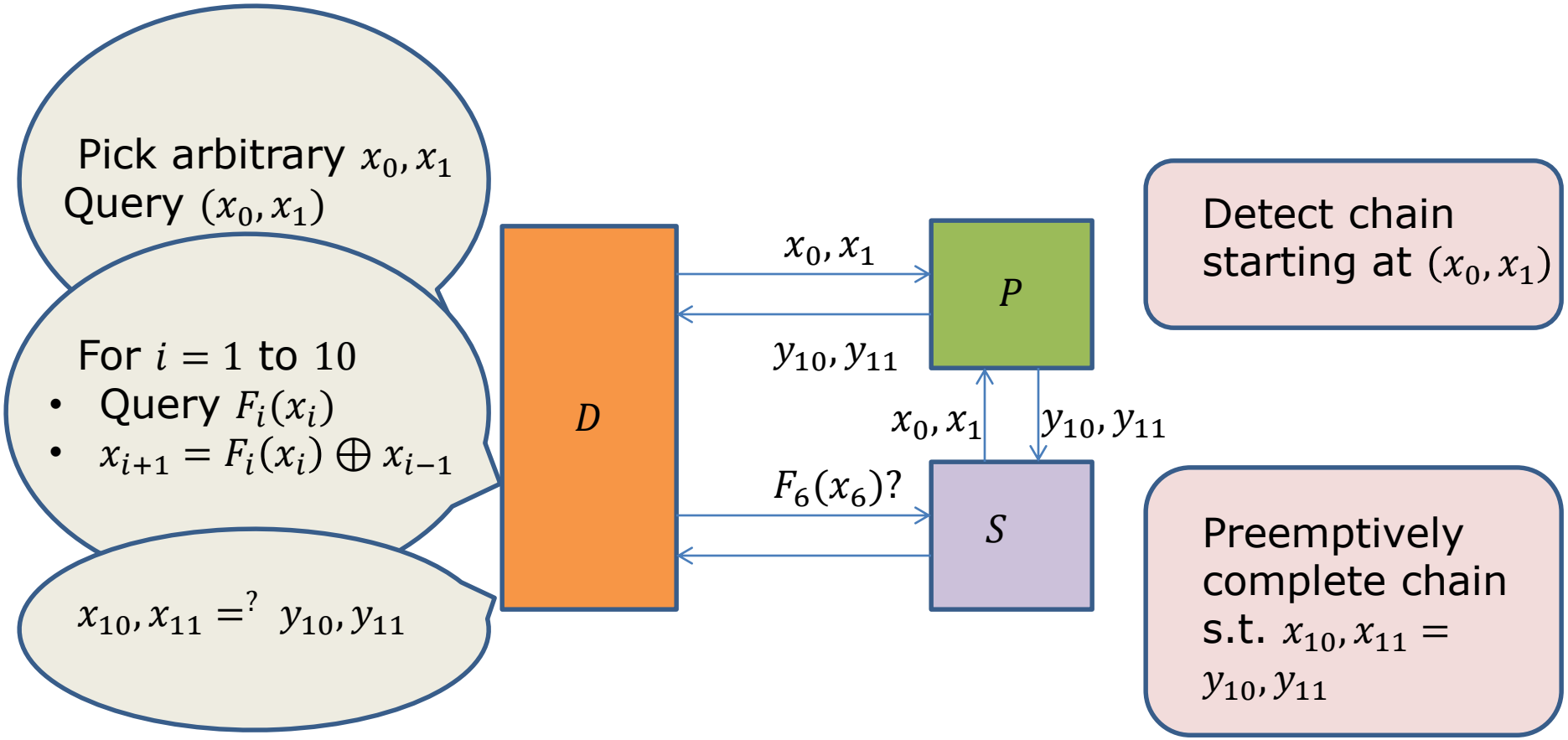
How to choose $F_i(x_i)$?



How to choose $F_i(x_i)$?



What should Simulator do?



Simulator Strategy

Partial chain detection

Preemptive completion

Simulator Strategy: Partial chain Detection

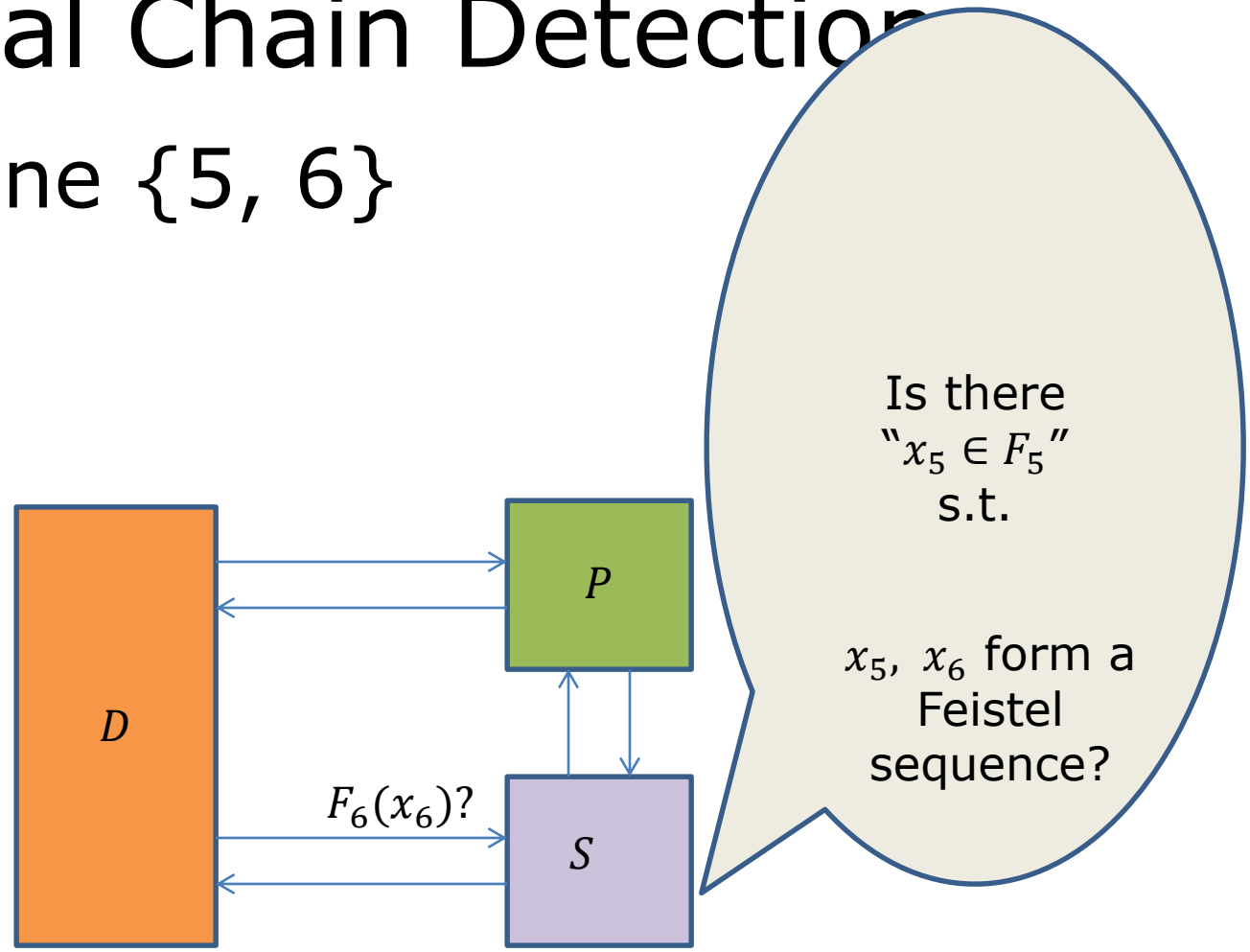
- In example,
  - D queried $F_1(x_1), \ldots, F_6(x_6)$
  - S checked if $x_1, \ldots, x_6$ formed a valid Feistel sub-sequence
- What if
  - D queried $F_6(x_6), \ldots, F_1(x_1)$?
  - D queried $F_1(x_1), F_1(x_1'), \ldots, F_6(x_6)$?

Simulator Strategy: Partial chain Detection

- Three detect zones
  - Spanning rounds {9, 10, 1}, {10, 1, 2} and {5, 6}

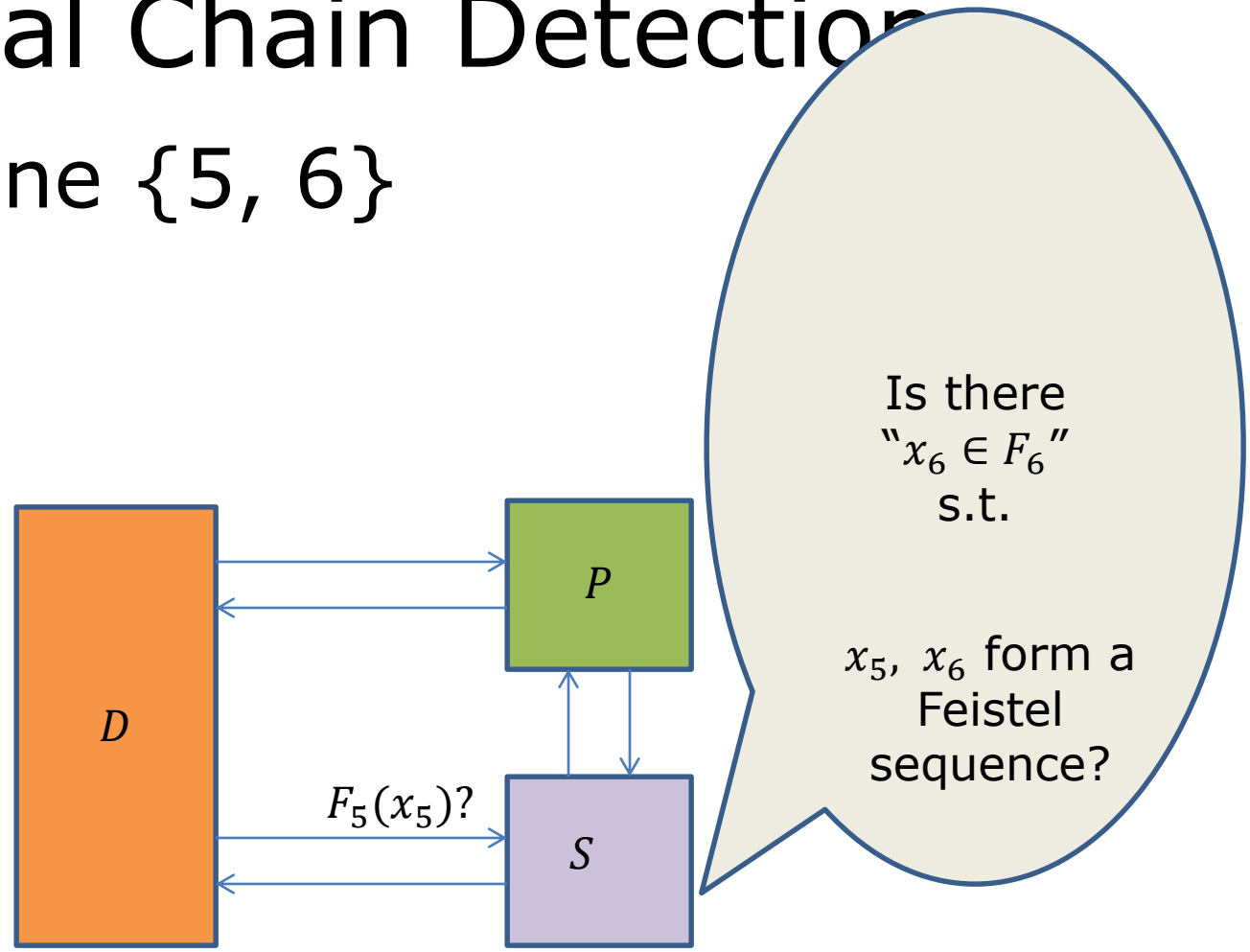
Simulator Strategy: Partial Chain Detection

- Detect zone {5, 6}



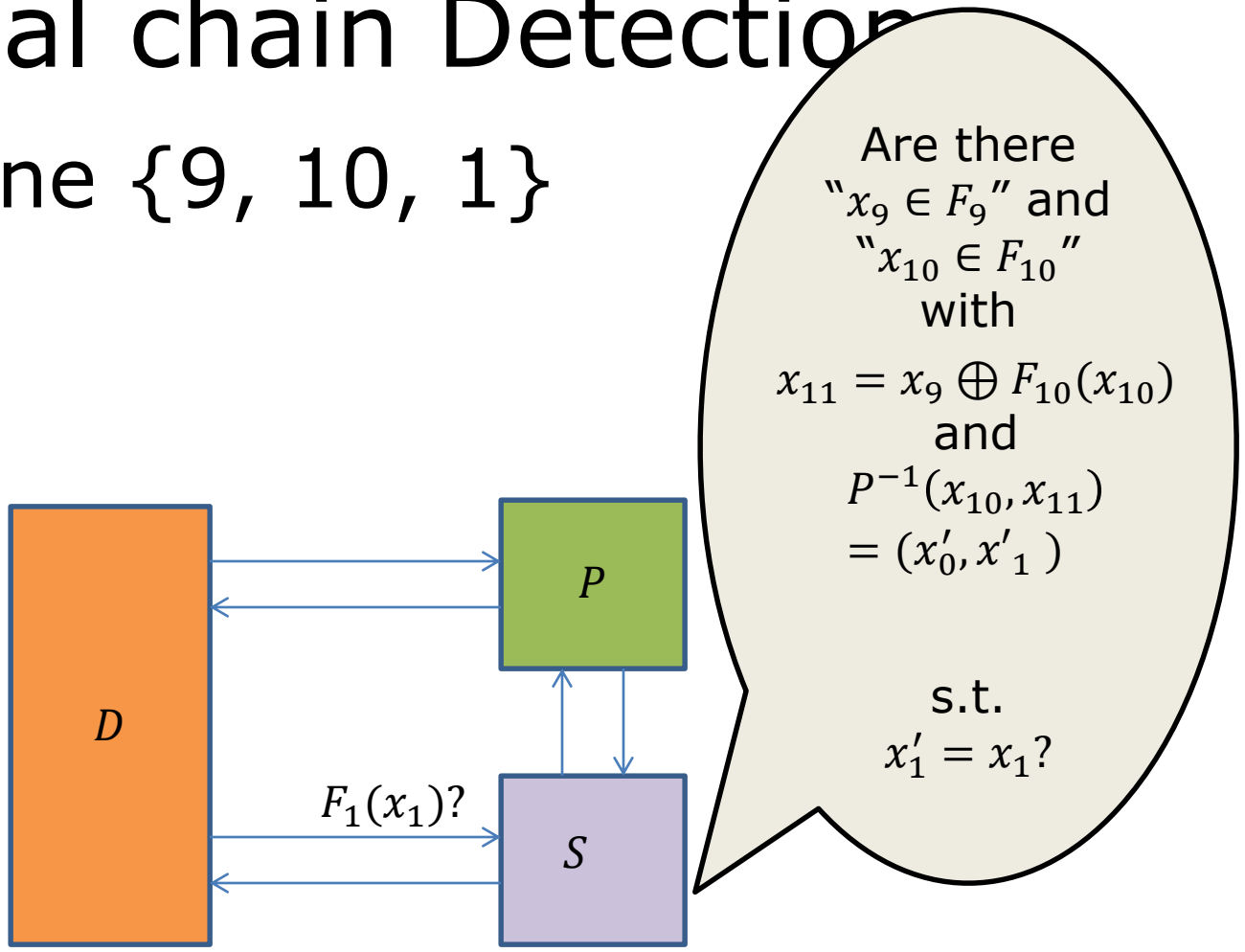
Simulator Strategy: Partial Chain Detection

- Detect zone {5, 6}



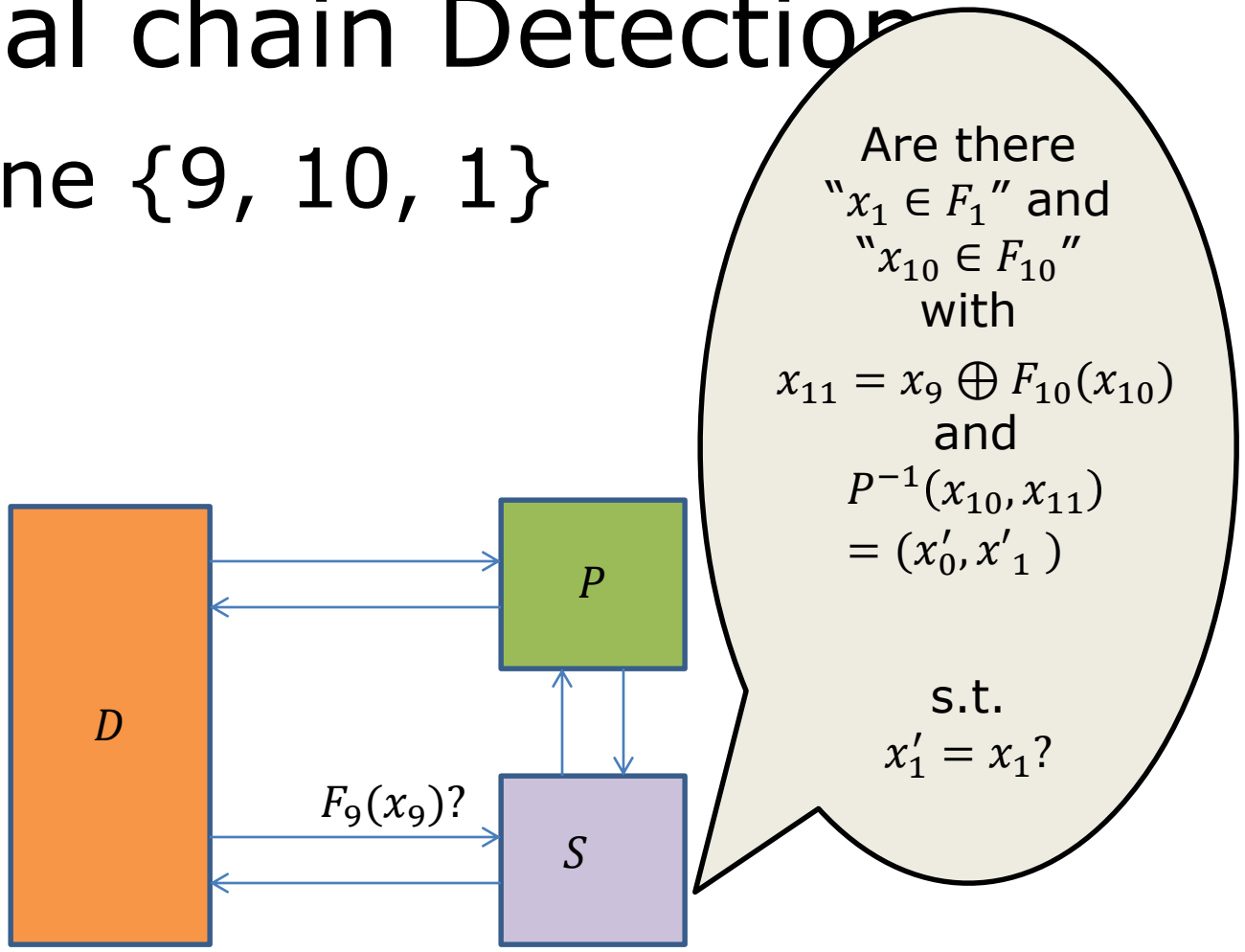
Simulator Strategy: Partial chain Detection

- Detect zone {9, 10, 1}



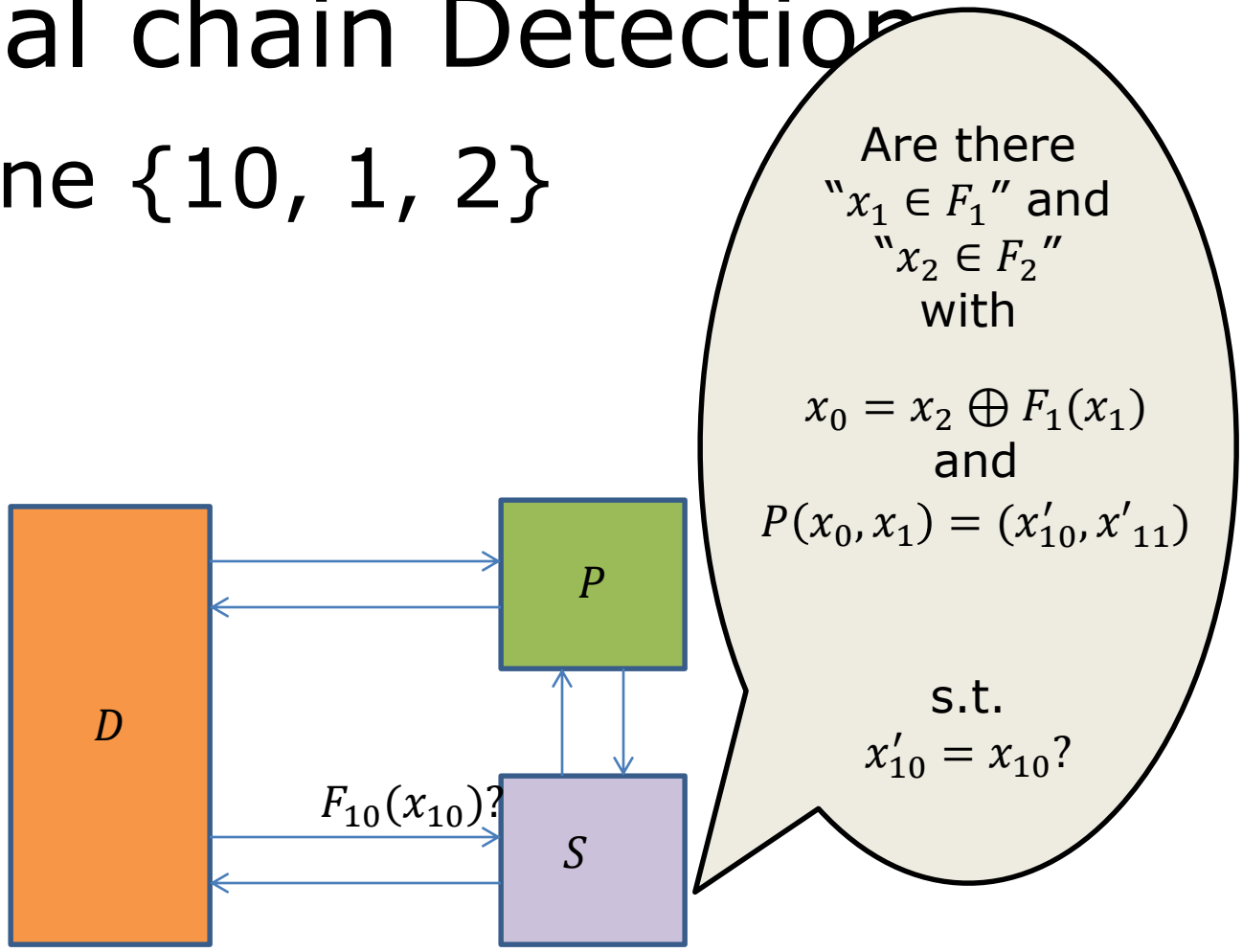
Simulator Strategy: Partial chain Detection

- Detect zone {9, 10, 1}



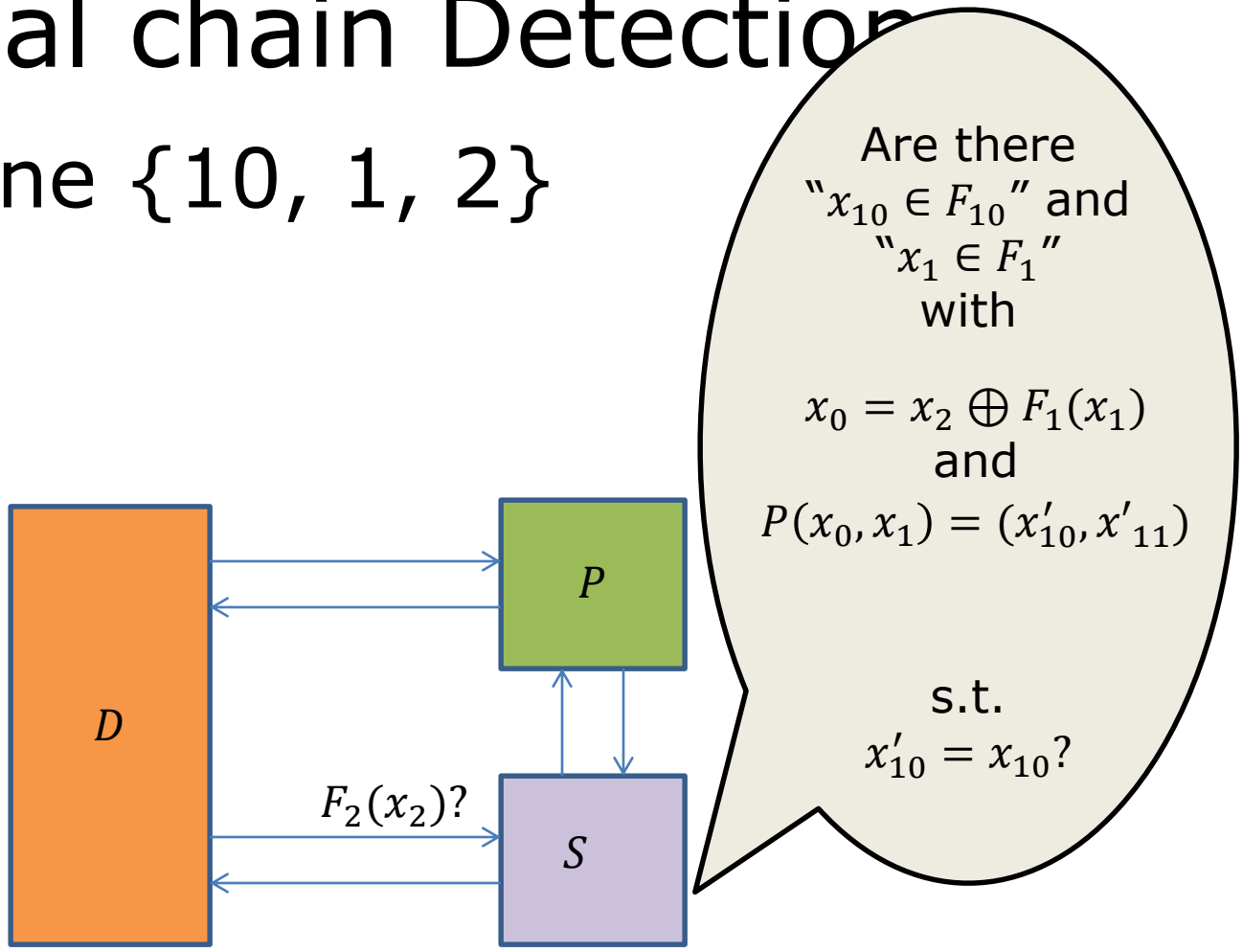
Simulator Strategy: Partial chain Detection

- Detect zone {10, 1, 2}



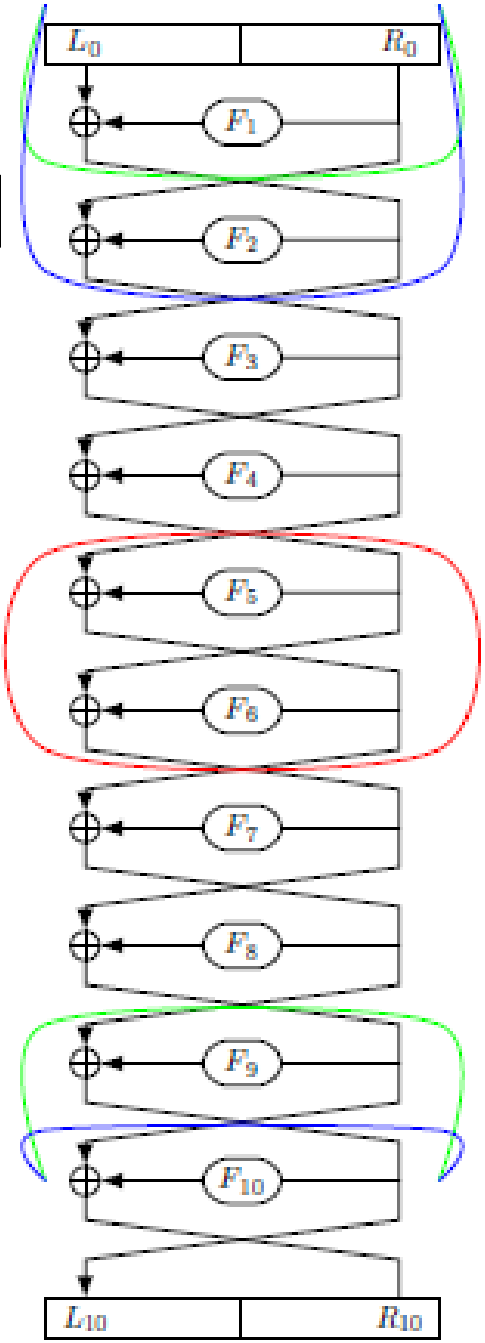
Simulator Strategy: Partial chain Detection

- Detect zone {10, 1, 2}



Simulator Strateg
Partial chain Detect

- Three detect zones
  - Spanning rounds {9, 10, 1}, {10, 1, 2} and {5, 6}

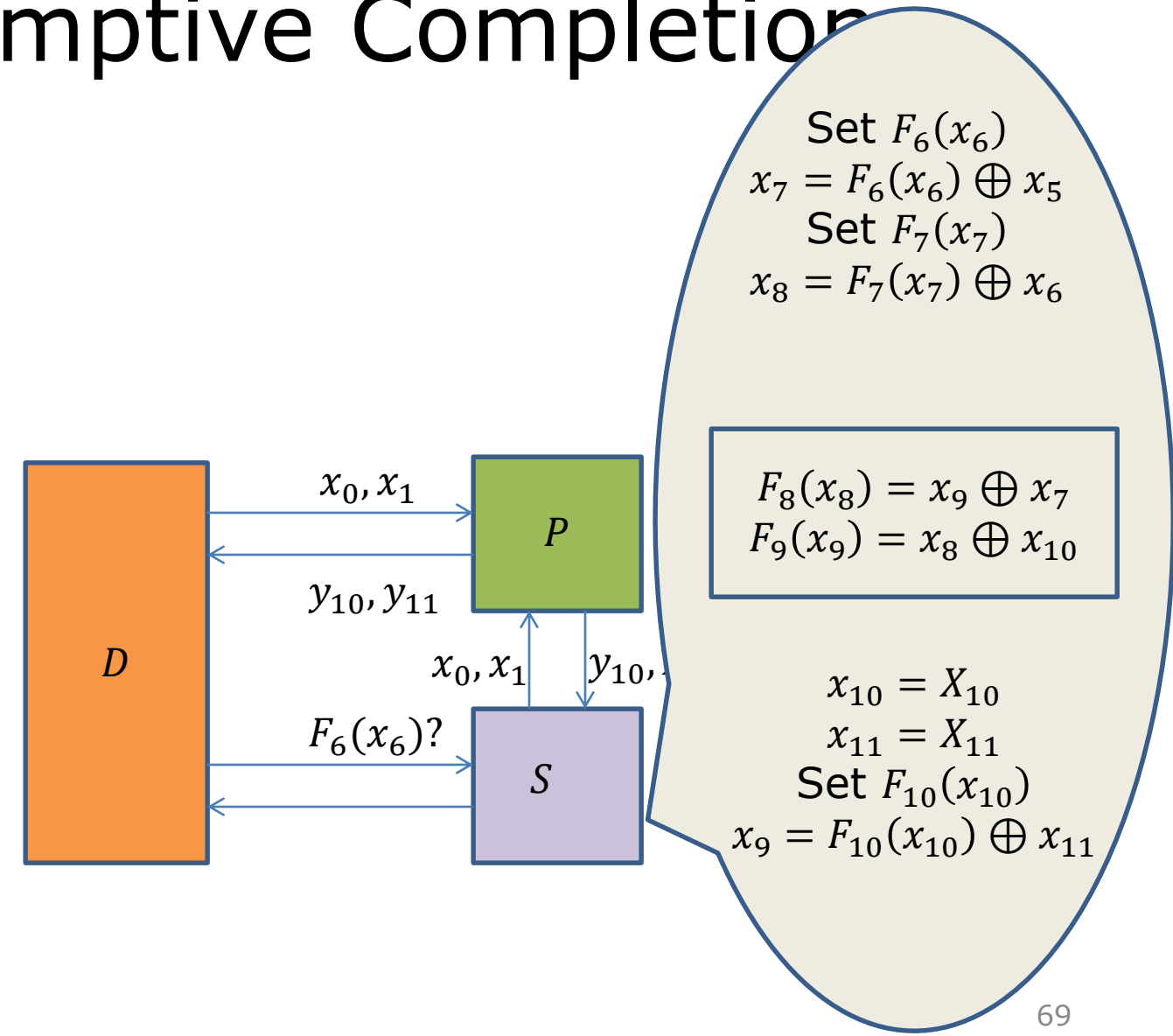


Simulator Strategy

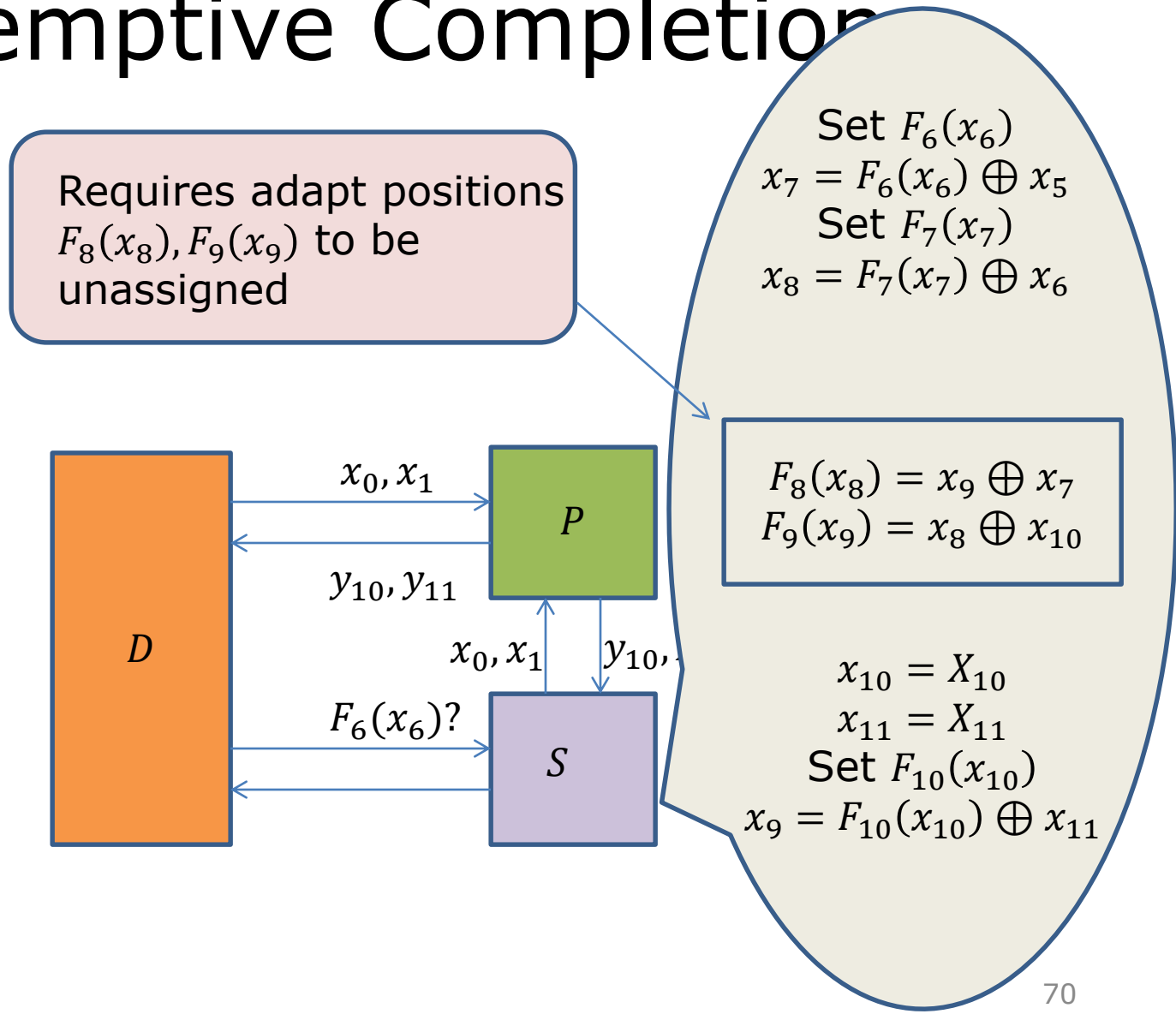
Partial chain detection

Preemptive completion

Simulator Strategy: Preemptive Completion



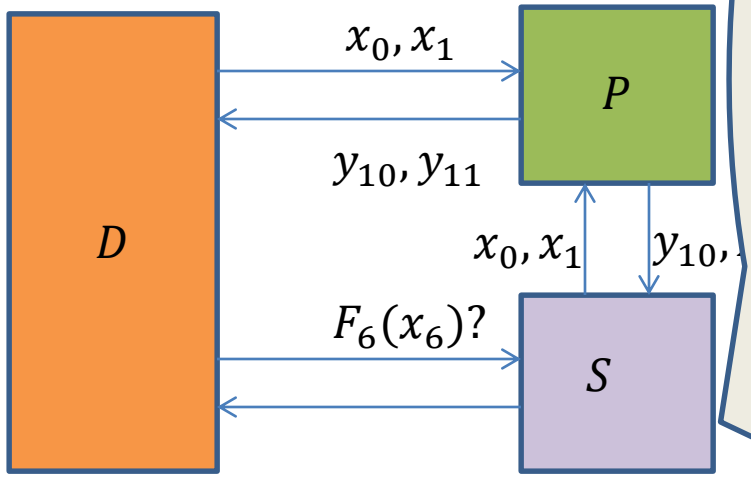
Simulator Strategy: Preemptive Completion



Simulator Strategy: Preemptive Completion

How?

Requires adapt positions $F_8(x_8), F_9(x_9)$ to be unassigned

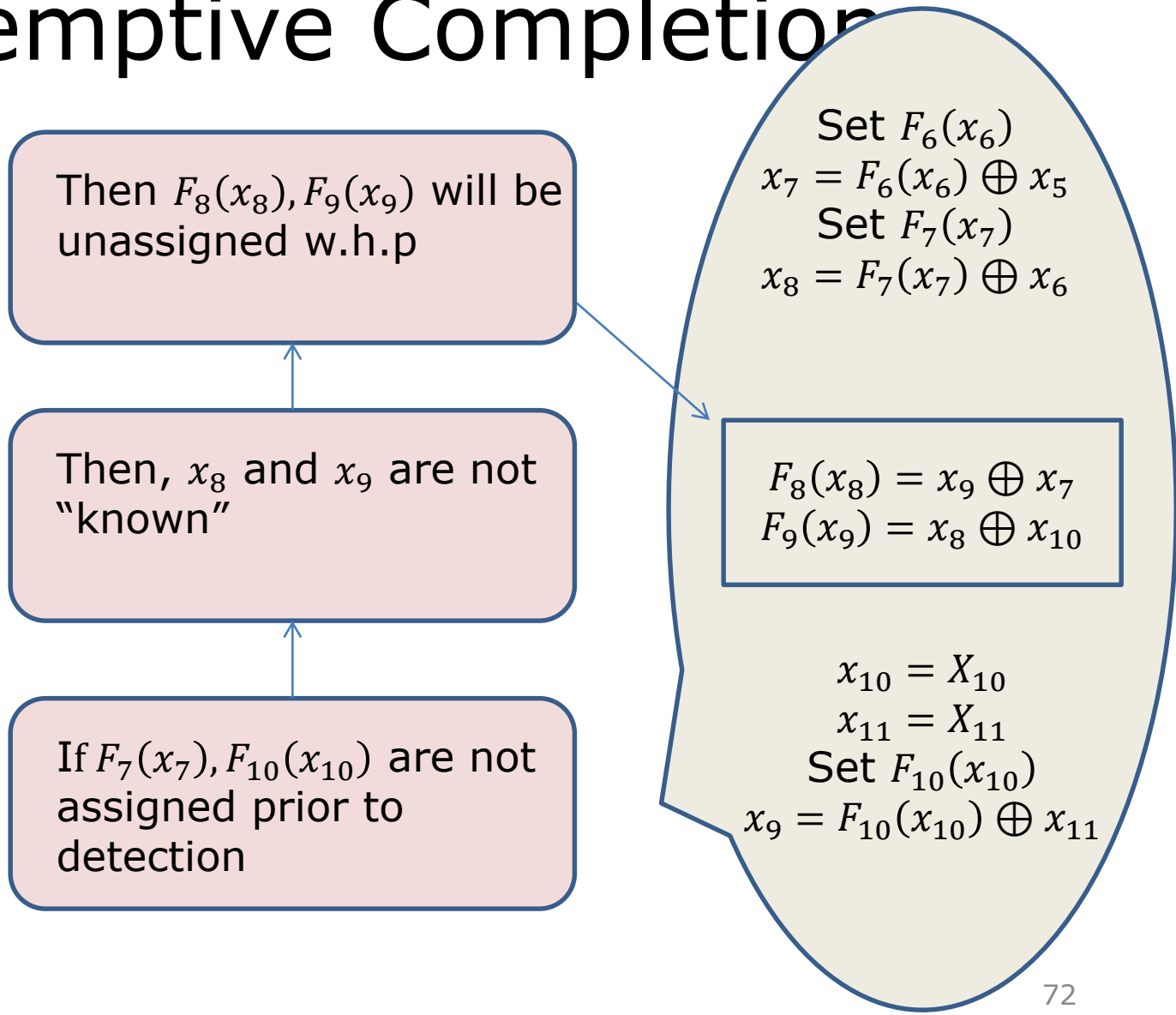


Set $F_6(x_6)$
$x_7 = F_6(x_6) \oplus x_5$
Set $F_7(x_7)$
$x_8 = F_7(x_7) \oplus x_6$

$F_8(x_8) = x_9 \oplus x_7$
$F_9(x_9) = x_8 \oplus x_{10}$

$x_{10} = X_{10}$
$x_{11} = X_{11}$
Set $F_{10}(x_{10})$
$x_9 = F_{10}(x_{10}) \oplus x_{11}$

Simulator Strategy: Preemptive Completion



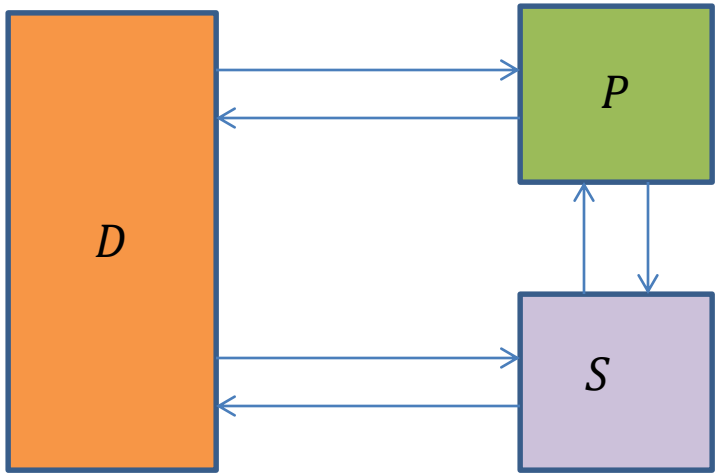
Simulator Strategy

Partial chain detection

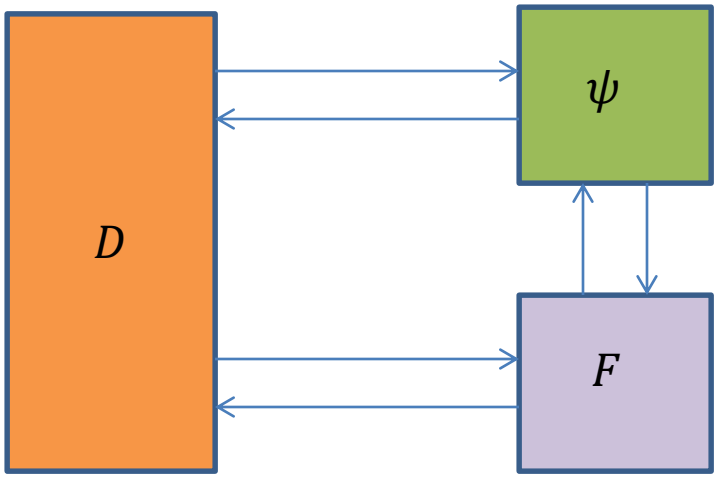
Preemptive completion

10-round ψ indifferentiable from P

- Ideal World



- Real World

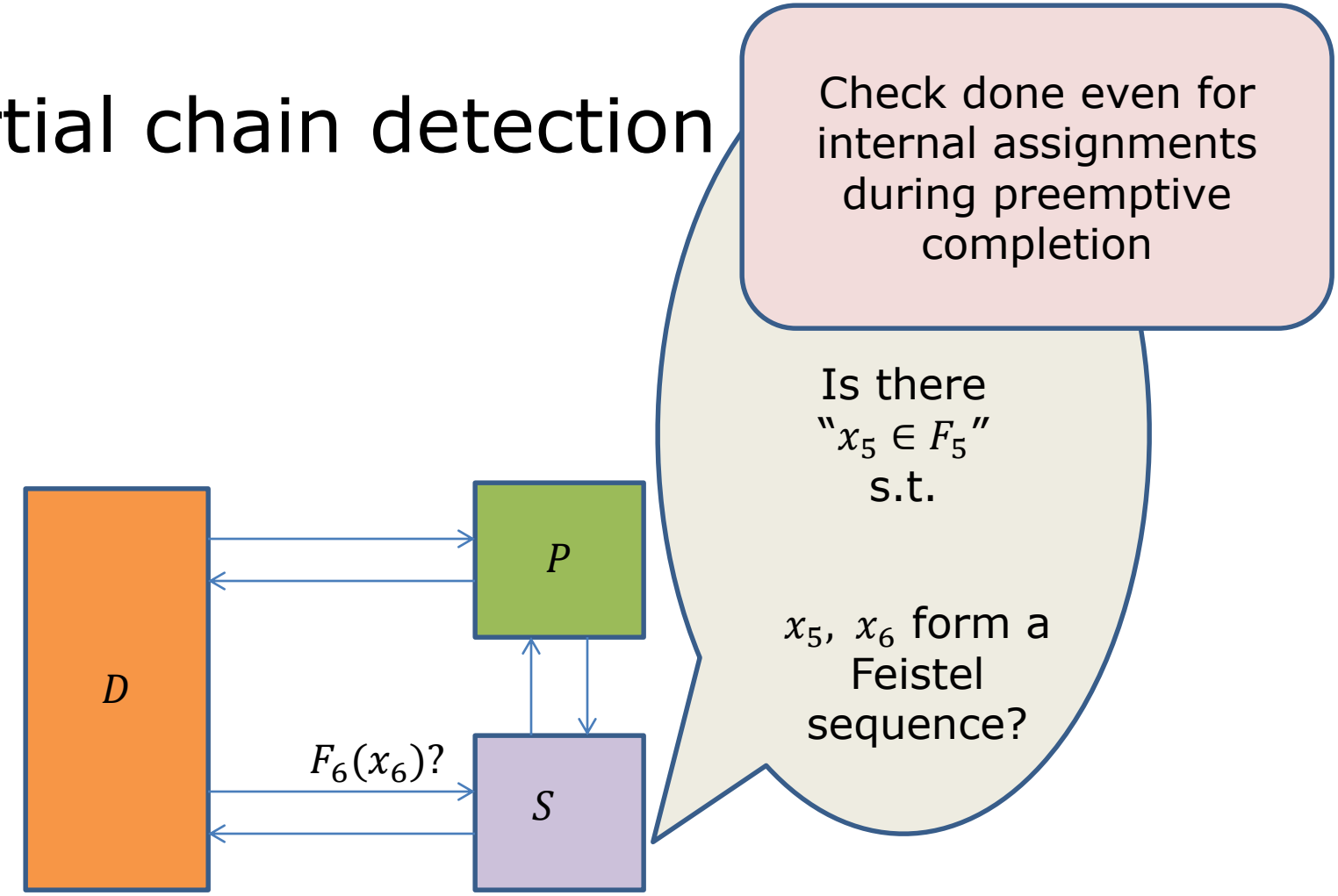


(1) Shown S s.t. no (efficient) D can distinguish w.h.p

(2) To show: S is efficient

Simulator Efficiency

- Partial chain detection



Simulator Efficiency

- No. of partial chains detected
  - Detect zone {5, 6}
  - Detect zone {9, 10, 1}
  - Detect zone {10, 1, 2}

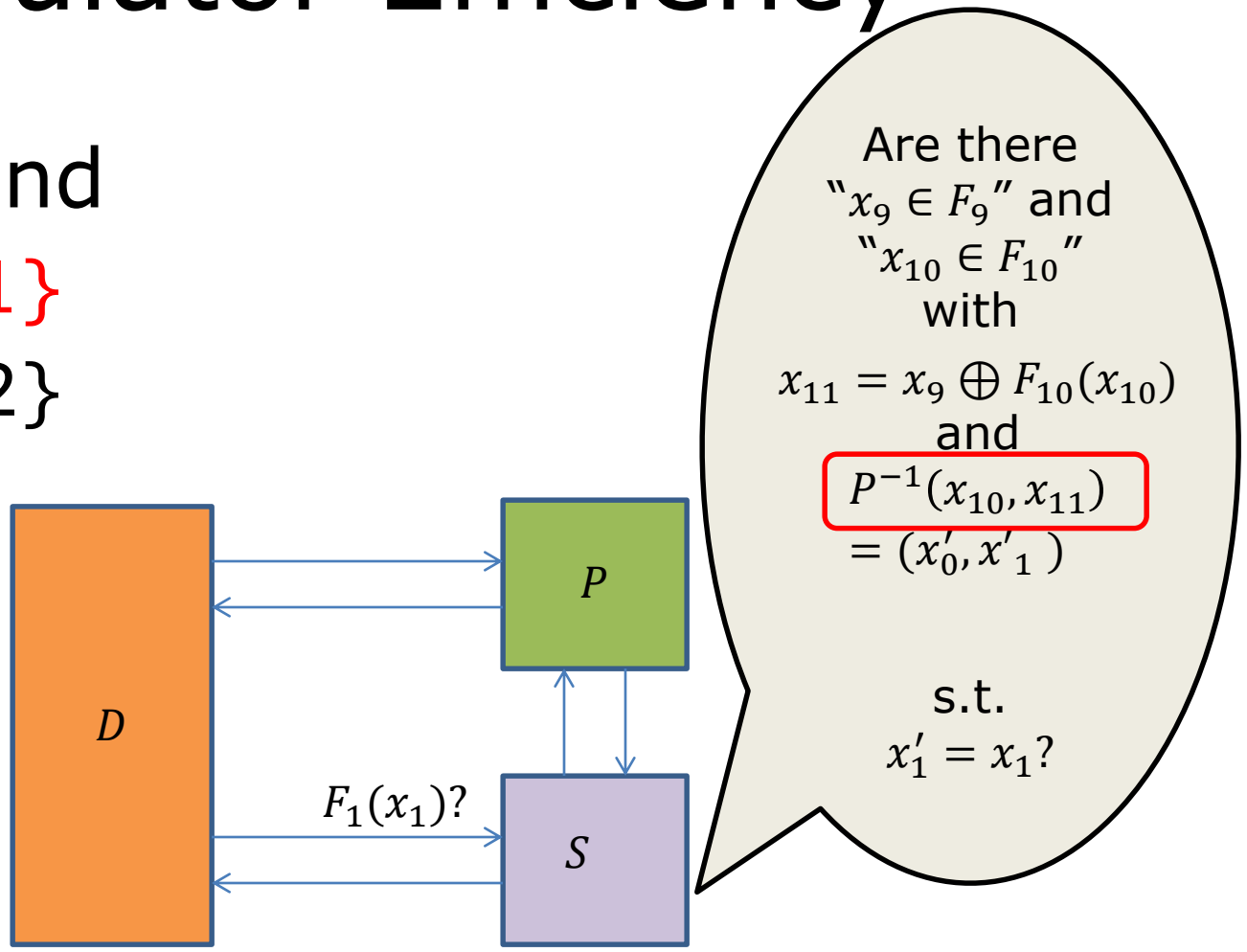
Simulator Efficiency

- No. of partial chains detected
  - Wrap-around detect zones
    - {9, 10, 1} and {10, 1, 2}
  - Inner detect zone
    - {5, 6}

Simulator Efficiency

Wrap-around

- {9, 10, 1}
- {10, 1, 2}



Simulator Efficiency

Wrap-around

- {9, 10, 1}
- {10, 1, 2}

- Involve a query to P
  - Charged to D
- At most q such chains detected

Simulator Efficiency

Wrap-around

- {9, 10, 1}
- {10, 1, 2}

Inner

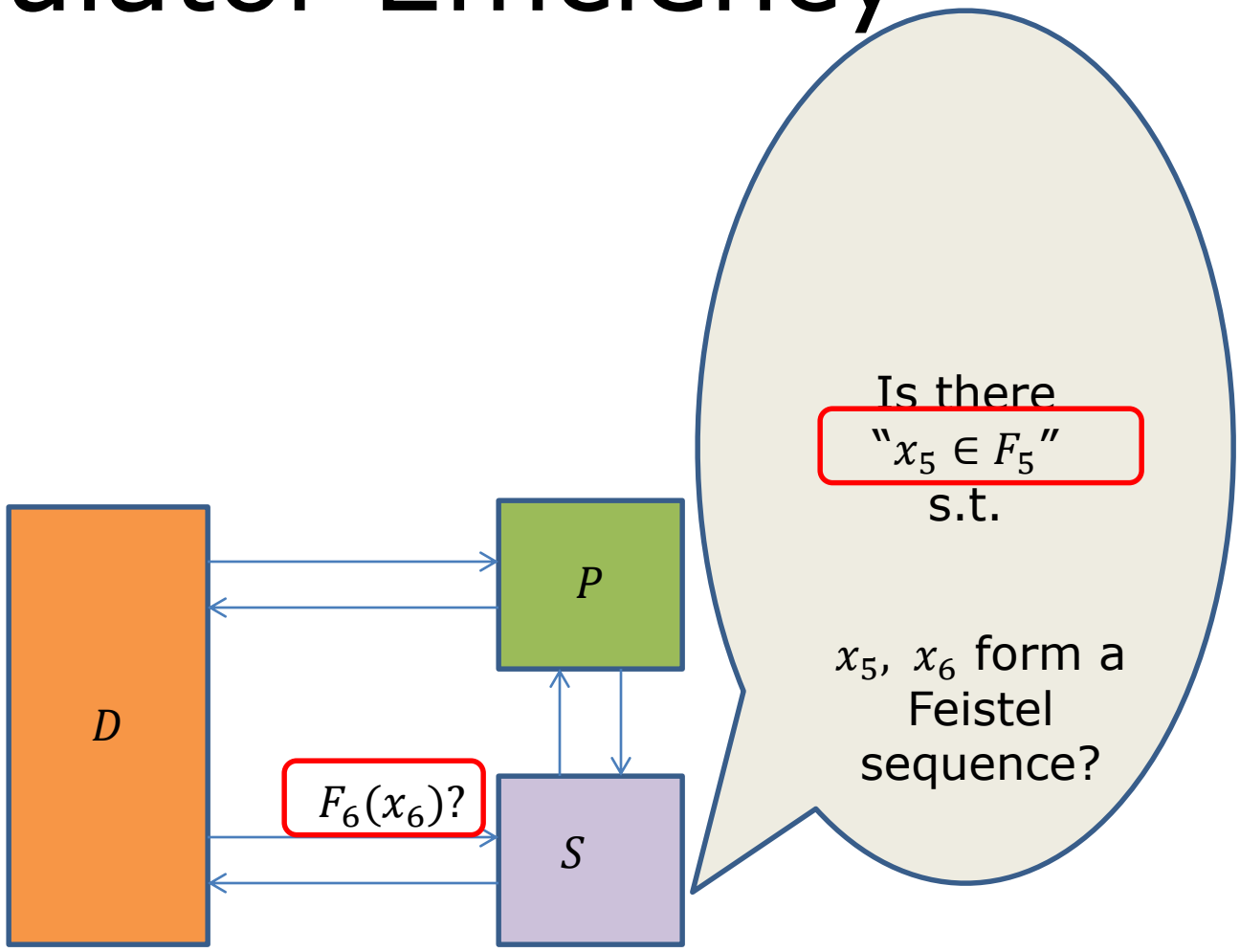
- {5, 6}

- Involve a query to P
  - Charged to D
- At most q such chains detected

Simulator Efficiency

Inner

- {5, 6}



Simulator Efficiency

Wrap-around
- {9, 10, 1}
- {10, 1, 2}

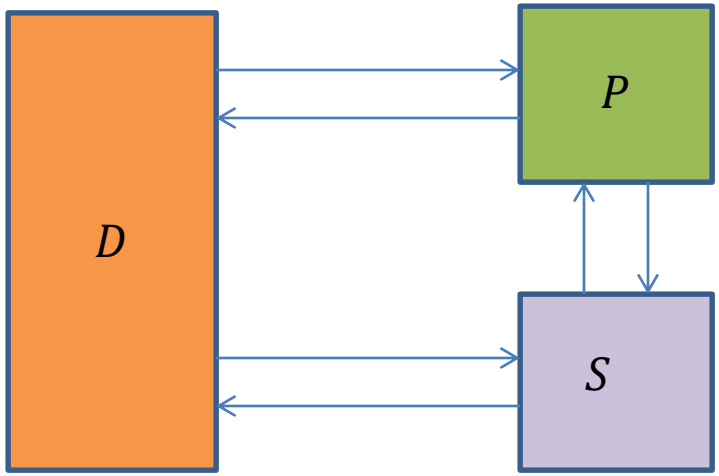
- Involve a query to P
  - Charged to D
- At most q such chains detected

Inner
- {5, 6}

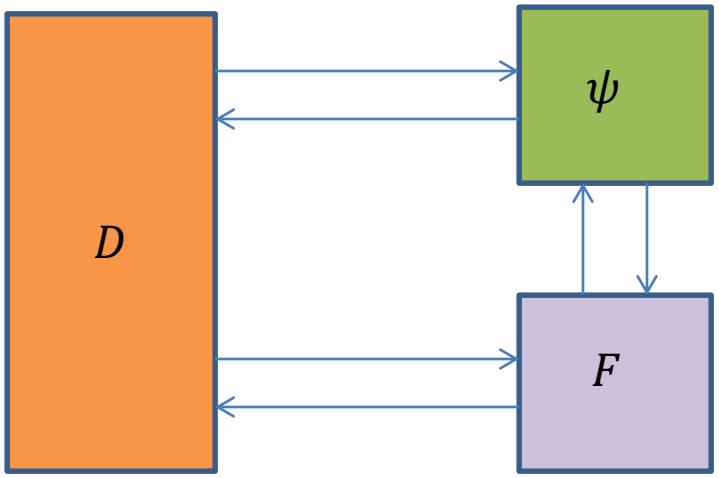
- Require F_5, F_6 queries to be defined
  - through D queries
  - Preemptive completion of wrap-around chains

10-round ψ indifferentiable from P

- Ideal World



- Real World

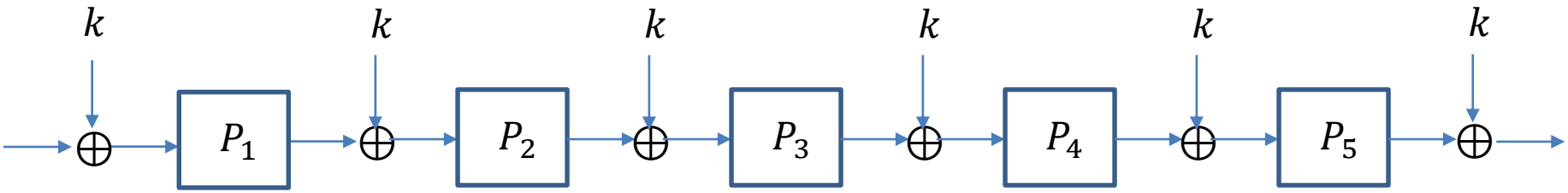


We show: (efficient) S s.t.

No (efficient) D can distinguish between real and ideal with prob. $O(q^{12}/2^n)$

Indifferentiability of Key-alternating Ciphers

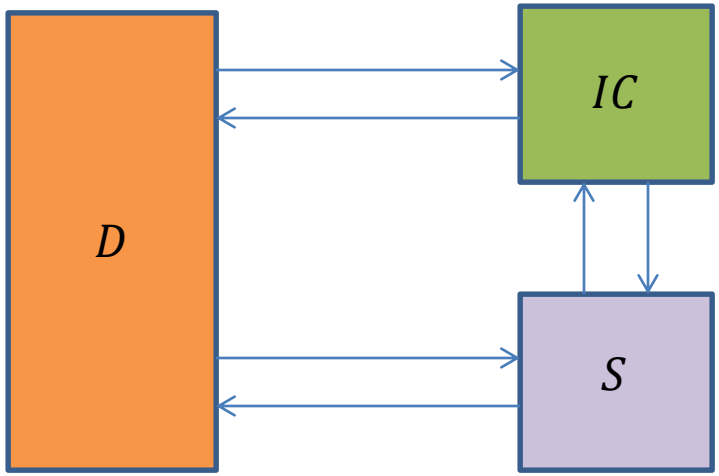
Key-alternating Ciphers



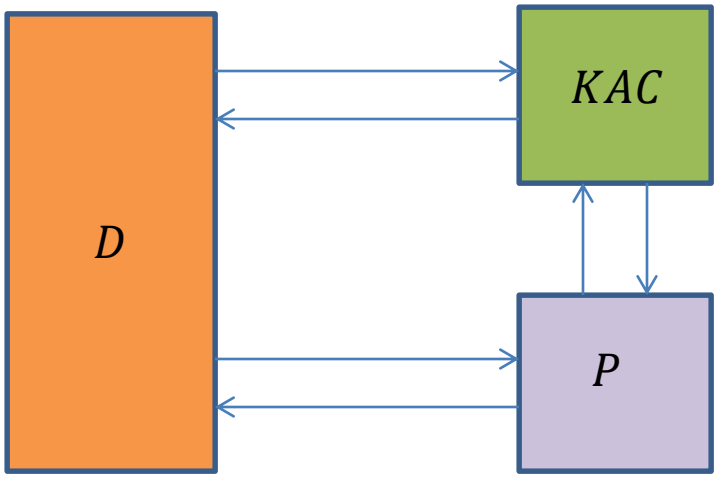
- Iterated structure
- Repeated application of (public) permutations
  - $P_1, \ldots, P_r : \{0,1\}^n \rightarrow \{0,1\}^n$

Indifferentiability

- Ideal World



- Real World



Sufficient to show: (efficient) S s.t.

No (efficient) D can distinguish between real and ideal w.h.p

Related Work

- 12 rounds sufficient [LS13]
  - 3 rounds insufficient
- Here: 5 rounds sufficient
  - 4 rounds insufficient [DSST17]

- Idealized key-derivation
  - 5 rounds sufficient [ABDMS13]
  - 3 rounds sufficient [GL16]

Simulator Strategy

Partial chain detection

Preemptive completion

Simulator Efficiency

- No. of partial chains detected
  - Detect zone {1, 2, 3}
  - Detect zone {2, 3, 4}
  - Detect zone {3, 4, 5}
  - Detect zone {4, 5, 1}
  - Detect zone {5, 1 ,2}

Simulator Efficiency

- No. of partial chains detected
  - Wrap-around detect zones
    - {4, 5, 1} and {5, 1, 2}
  - (multiple) Inner detect zones
    - {1, 2, 3}, {2, 3, 4}, {3, 4, 5}

Simulator Efficiency

Wrap-around

- $\{4, 5, 1\}$
- $\{5, 1, 2\}$

- Charged to D
- At most q such chains detected

Simulator Efficiency

Wrap-around
- {4, 5, 1}
- {5, 1, 2}

- Charged to D
- At most q such chains detected

Inner
- {1, 2, 3}
- {2, 3, 4}
- {3, 4, 5}

Simulator Efficiency

Wrap-around
- {4, 5, 1}
- {5, 1, 2}

- Charged to D
- At most q such chains detected

Inner
- {1, 2, 3}
- {2, 3, 4}
- {3, 4, 5}

- Require queries at 1, 2 and 3 to be defined
  - D queries
  - Preemptive completion of
    - wrap-around chains
    - {3, 4, 5} chains

Simulator Efficiency

Wrap-around
- {4, 5, 1}
- {5, 1, 2}

- Charged to D
- At most q such chains detected

Inner
- {1, 2, 3}
- {2, 3, 4}
- {3, 4, 5}

Simulator Efficiency

Wrap-around
- {4, 5, 1}
- {5, 1, 2}

- Charged to D
- At most q such chains detected

Inner
- {1, 2, 3}
- {2, 3, 4}
- {3, 4, 5}

- Require query at P_3 to be defined
  - ➢ through D queries
  - ➢ Preemptive completion of wrap-around chains

Simulator Efficiency

Inner

- {1, 2, 3}
- {2, 3, 4}
- {3, 4, 5}

- {3, 4, 5}

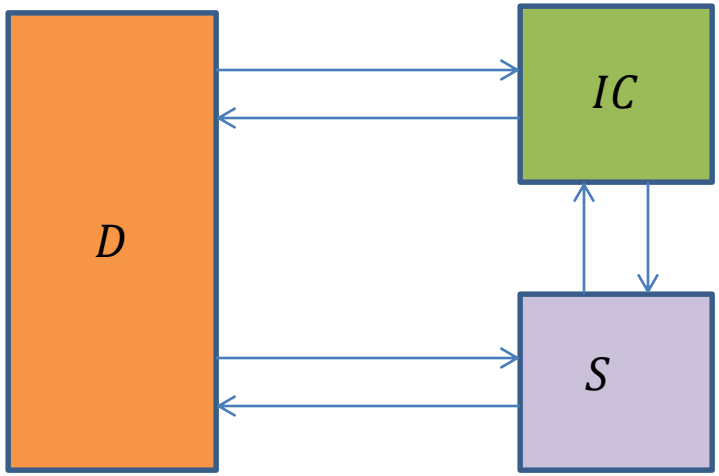
- Require query at P_3 to be defined
  - through D queries
  - Preemptive completion of wrap-around chains

Claim:

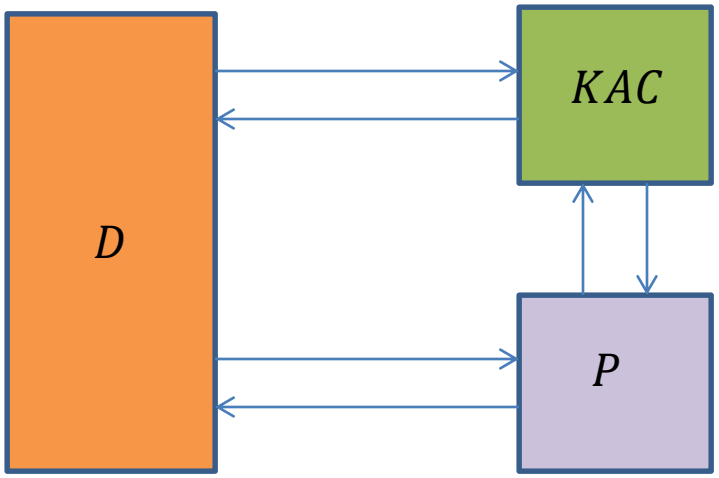
- A chain detected at {3, 4, 5} can be uniquely mapped to
  - A P_3 query and a D query
  - A pair of P_3 queries

5-round KAC indifferentiabe from an ideal cipher

- Ideal World



- Real World



We show: efficient S s.t.

No (efficient) D can distinguish between real and ideal with prob. $O(q^{38}/2^n)$

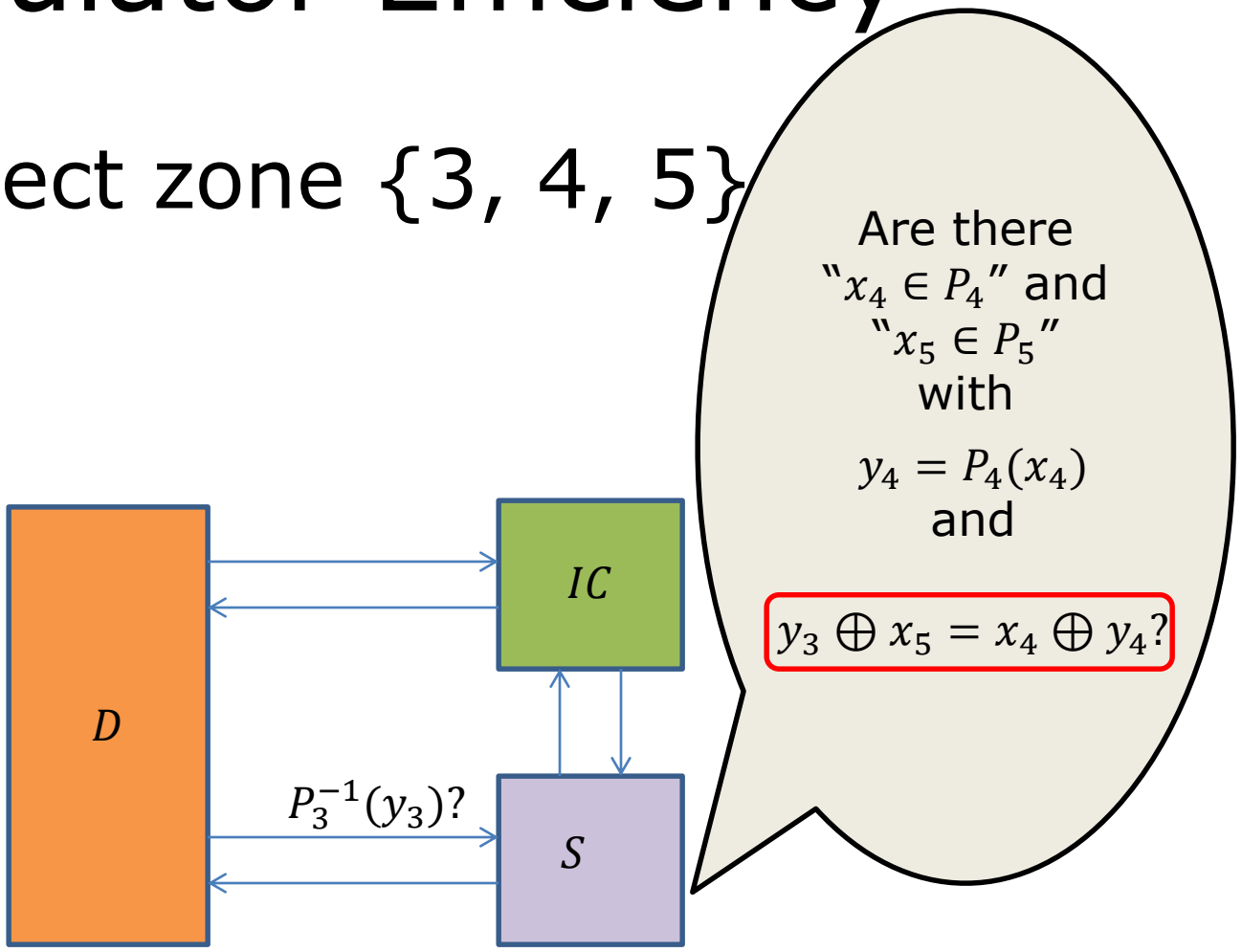
Conclusion

- Security of Block Ciphers
  - Indifferentiability [MRH04]
- Security of Feistel Networks [DKT16]
  - 10-round Feistel
- Security of Key-alternating Ciphers [DSST17]
  - 5-round KAC

Thank You

Simulator Efficiency

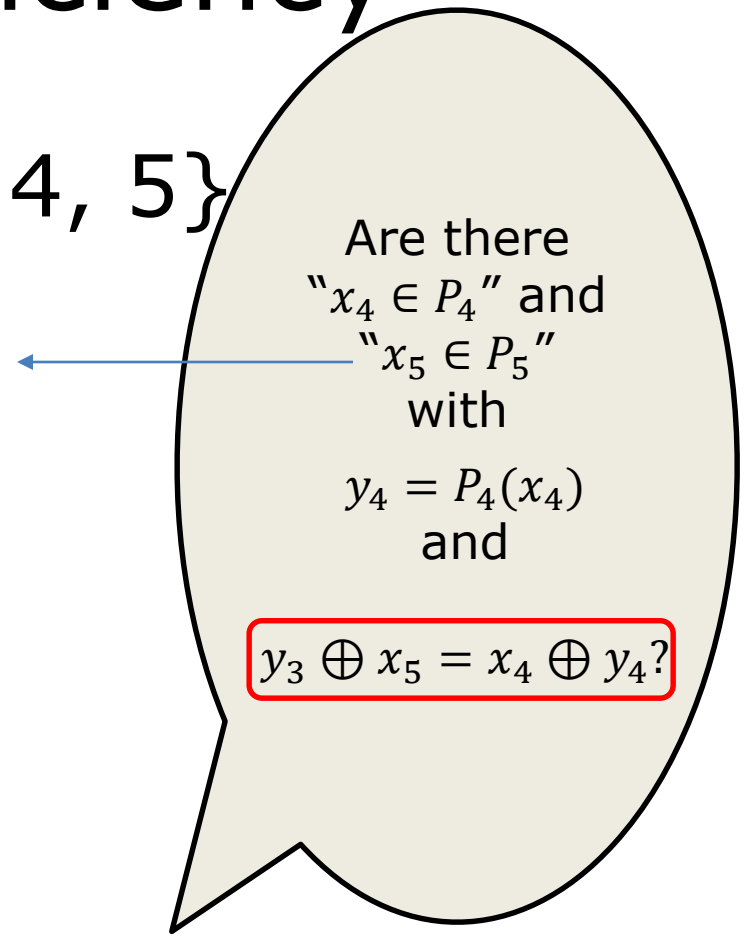
- Inner Detect zone {3, 4, 5}



Simulator Efficiency

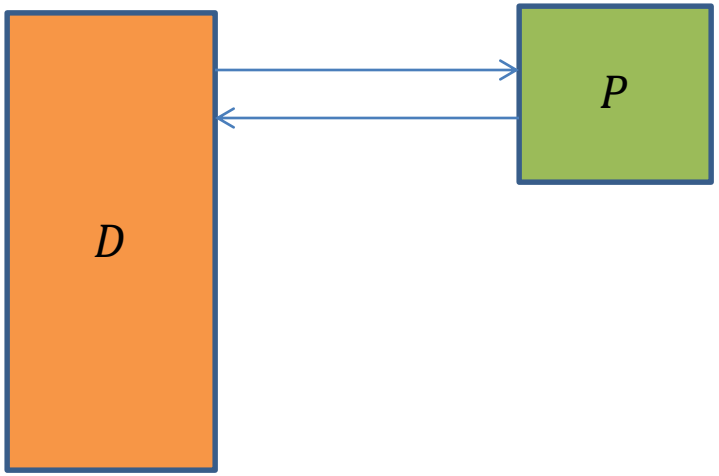
- Inner Detect zone {3, 4, 5}

If $x_5 \in P_5$ due to
- D query
  - $y_3 \oplus x_5 = x_4 \oplus y_4$
- Completion of another chain
  - $y_3 \oplus x_4 \oplus y_4 = x_5 = y_3' \oplus x_4' \oplus y_4'$
  - i.e., $y_3 \oplus y_3' = x_4 \oplus y_4 \oplus x_4' \oplus y_4'$



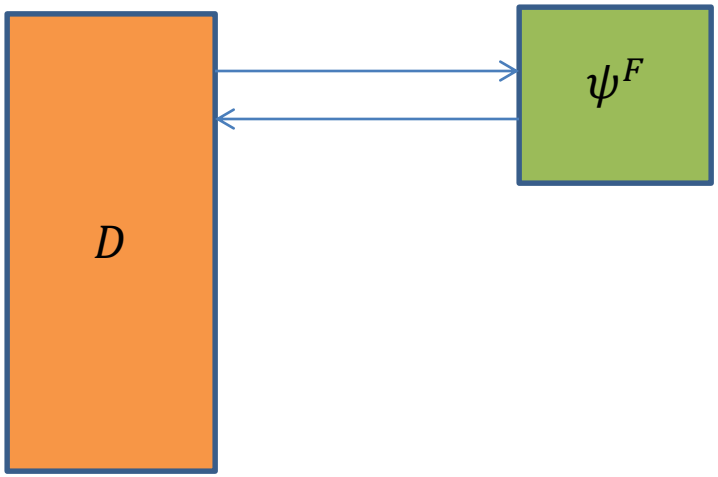
Security of Feistel Networks: Indistinguishability

- Ideal World



- P – random permutation

- Real World



- ψ – Feistel construction
- $F = \{F_1, \ldots, F_r\}$

Indistinguishability of Feistel Networks

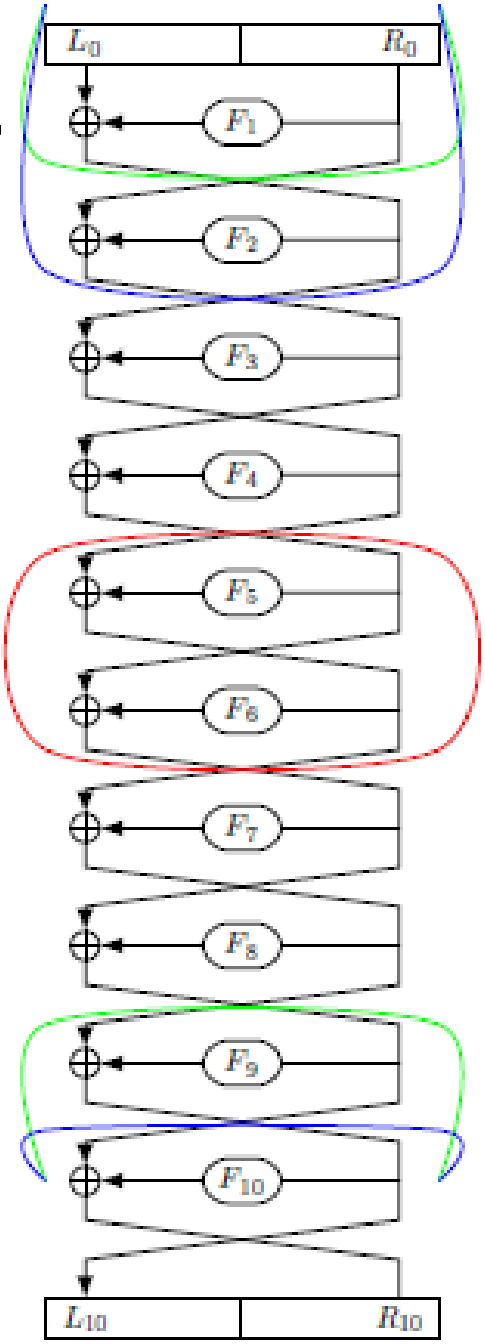
- [LR88] 4-round Feistel indistinguishable from random permutation
  - F independent, (secretly-keyed) random functions

Simulator Efficiency

- Related to size of tables F_i
- Size of F_i can increase only due to
  - D query to F_i
    - at most q such queries
  - Preemptive completion of a chain detected by the simulator

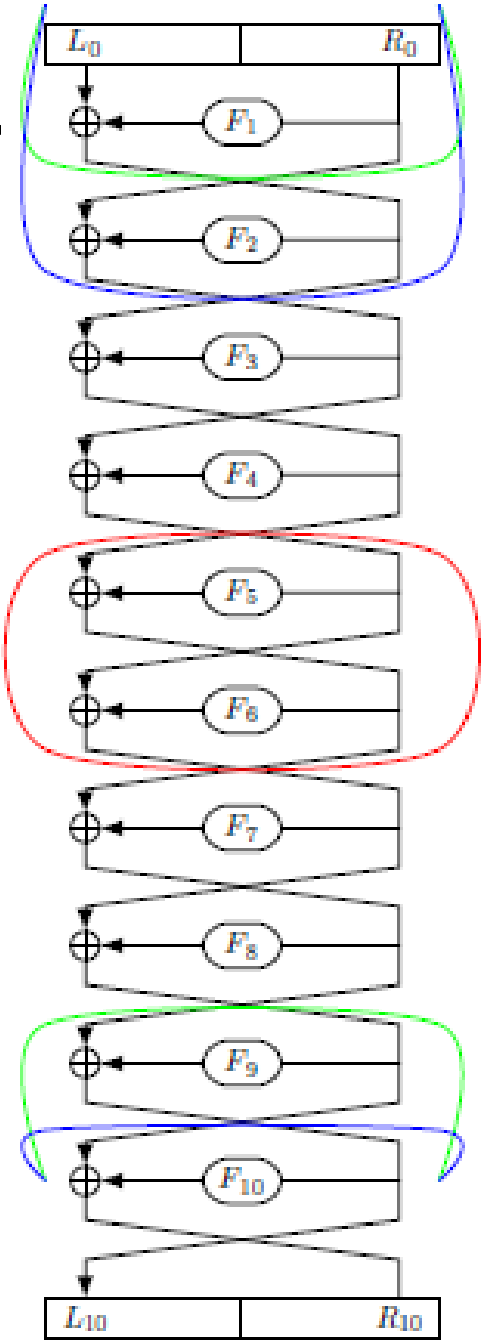
Simulator Efficier

- Three detect zones
  - Wrap-around: {9, 10, 1}, {10, 1, 2}
  - Middle: {5, 6}
- Wrap-around zones
  - Involve a query to P
    - Charged to distinguisher D
  - At most q such chains get detected



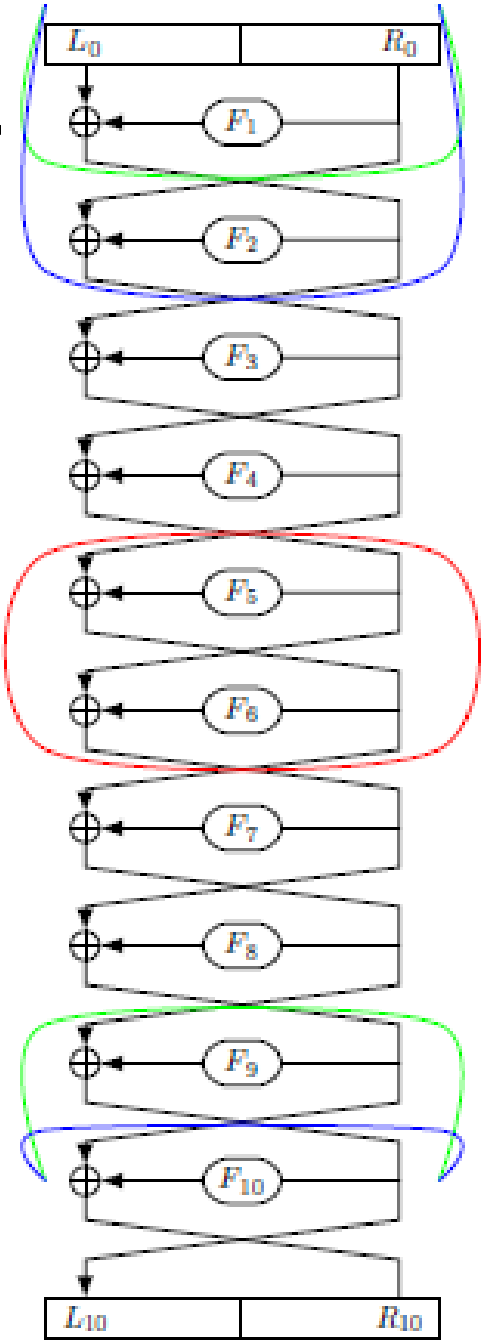
Simulator Efficier

- Three detect zones
  - Wrap-around: {9, 10, 1}, {10, 1, 2}
  - Middle: {5, 6}
- Middle zones with F_5 and F_6 filled due to
  - D query
  - Completion of wrap-around chains



Simulator Efficier

- Middle zones with F_5 and F_6 filled due to
  - D query
    - At most q such
  - Completion of wrap-around chains
    - At most q such



Simulator Efficiency

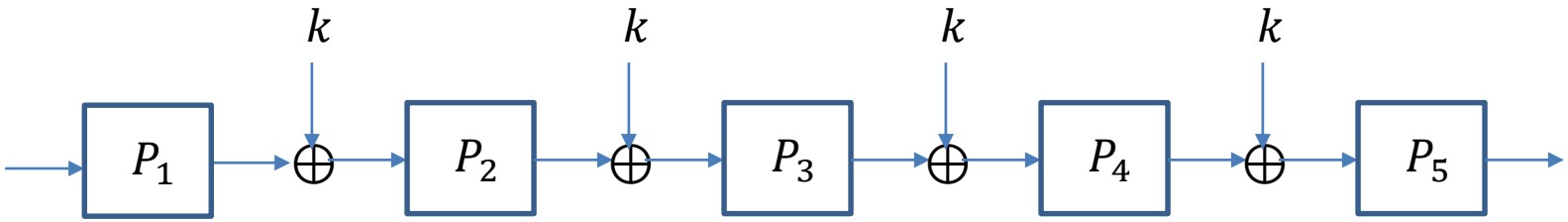
- Related to size of tables F_i
- Size of F_i can increase only due to
  - D query to F_i
    - at most q such queries
  - Preemptive completion of a chain detected by the simulator
    - q wrap-around chains
    - $O(q^2)$ middle chains

Simulator Efficiency

- Related to size of tables P_i
- Size of P_i can increase only due to
  - D query to P_i
    - at most q such queries
  - Preemptive completion of a chain detected by the simulator

Simulator Efficiency

- Five detect zones
  - Consecutive rounds of three
  - Wrap-around: {5, 1, 2}, {4, 5, 1}
  - Middle: {1, 2, 3}, {2, 3, 4}, {3, 4, 5}



Simulator Efficiency

- Five detect zones
  - Wrap-around: {5, 1, 2}, {4, 5, 1}
  - Middle: {1, 2, 3}, {2, 3, 4}, {3, 4, 5}
- Wrap-around zones
  - Involve a query to P
    - Charged to distinguisher D
  - At most q such chains get detected

Simulator Efficiency

- Five detect zones
  - Wrap-around: {5, 1, 2}, {4, 5, 1}
  - Middle: {1, 2, 3}, {2, 3, 4}, {3, 4, 5}
- Middle zones
  - D query
  - Completion of wrap-around chains
  - Completion of other Middle chains

Simulator Efficiency

- Five detect zones
  - Wrap-around: {5, 1, 2}, {4, 5, 1}
  - Middle: {1, 2, 3}, {2, 3, 4}, {3, 4, 5}
- ~~Middle zones~~ P_3
  - D query
  - Completion of wrap-around chains
  - ~~Completion of other Middle chains~~

Simulator Efficiency

- Size of P_2 can increase only due to
  - D query to P_i
    - at most q such queries
  - Preemptive completion of a chain detected by the simulator
    - Wrap-around: {4, 5, 1} – at most q such
    - Middle: {3, 4, 5}

Simulator Efficiency

- Size of P_2 can increase due to
  - D query to P_i
    - at most q such queries
  - Preemptive completion of middle chain at {3, 4, 5}
    - Claim: Detection of chain at {3, 4, 5} can be uniquely mapped to
      - A P_3 query and a distinguisher query
      - Pair of P_3 queries

Simulator Efficiency

- Detection of chain at {3, 4, 5}
  - $x_3 \oplus y_5 = x_4 \oplus y_4$
- Query at 5, y_5, is either due to
  - A distinguisher query
  - Completion of another chain
    - $x_3 \oplus x_4 \oplus y_4 = y_5 = x_3' \oplus x_4' \oplus y_4'$