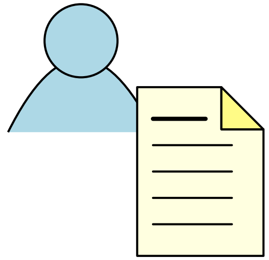
Secret Sharing

Imagine some sensitive information that is kept by a single agent

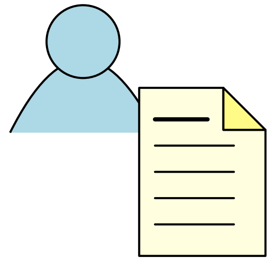
- A master encryption key
- Your bitcoin wallet
- Nuclear codes
- ...



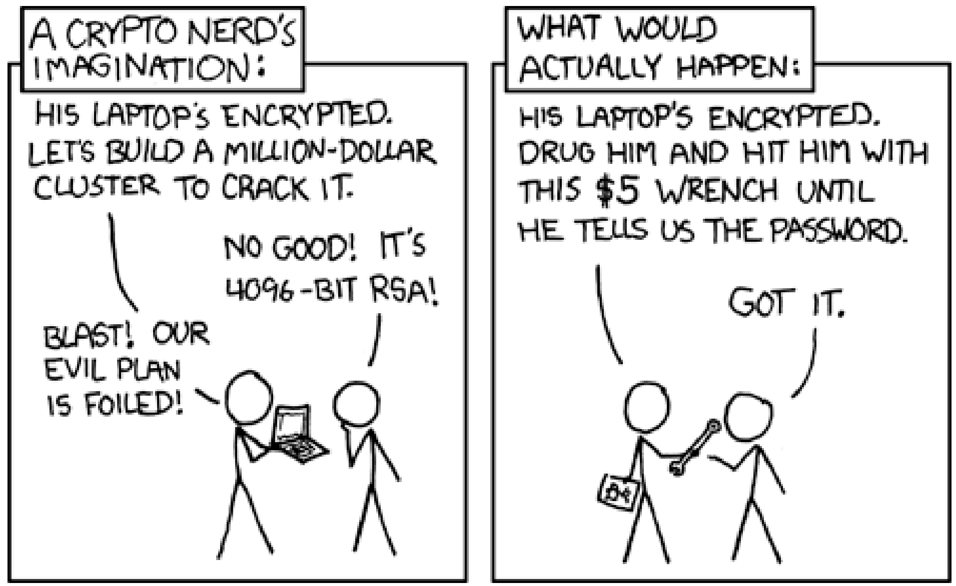
Secret Sharing

Imagine some sensitive information that is kept by a single agent

- A master encryption key 
- Your bitcoin wallet  
- Nuclear codes
- ...



Single point of failure!



Secret Sharing

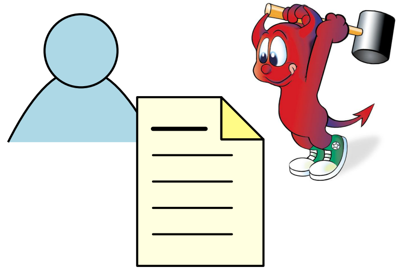
Imagine some sensitive information that is kept by a single agent

- A master encryption key

- Your bitcoin wallet

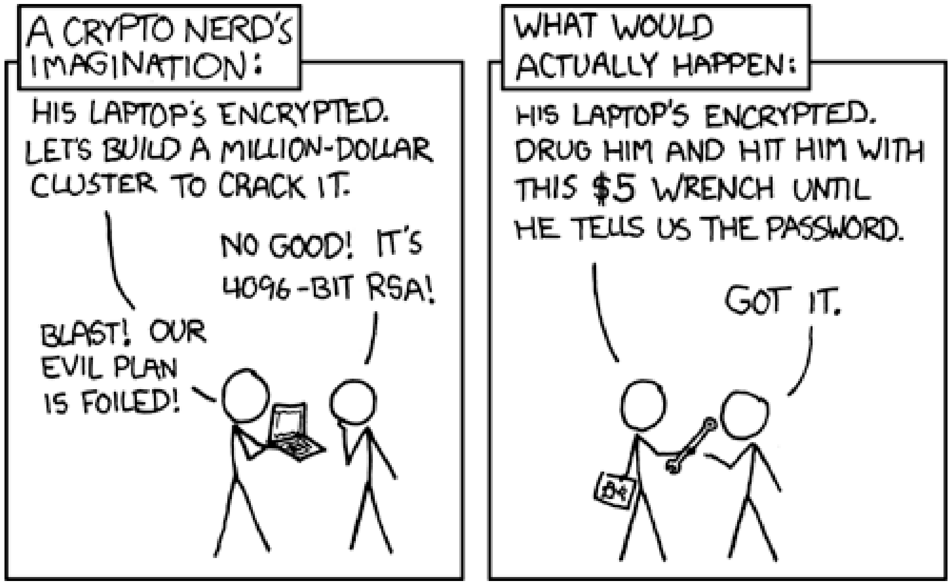
- Nuclear codes

- ...



Single point of failure!

An attacker can compromise one machine and steal the sensitive information



Secret Sharing

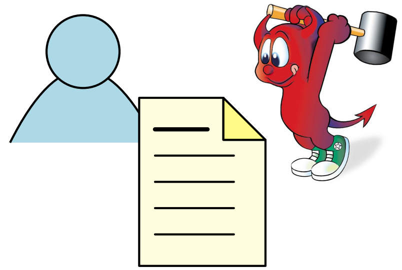
Imagine some sensitive information that is kept by a single agent

- A master encryption key

- Your bitcoin wallet

- Nuclear codes

- ...



Single point of failure!

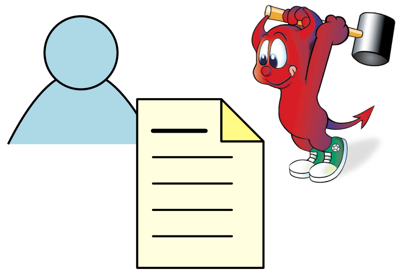
An attacker can compromise one machine and steal the sensitive information

Idea: Share the information across several agents

Secret Sharing

Imagine some sensitive information that is kept by a single agent

- A master encryption key
- Your bitcoin wallet
- Nuclear codes
- ...



Single point of failure!

An attacker can compromise one machine and steal the sensitive information

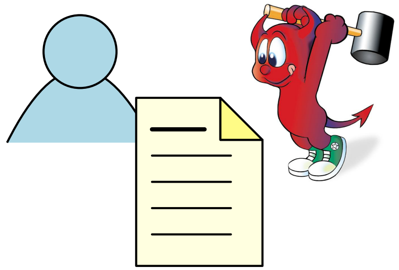
Idea: Share the information across several agents

"Magic box"

Secret Sharing

Imagine some sensitive information that is kept by a single agent

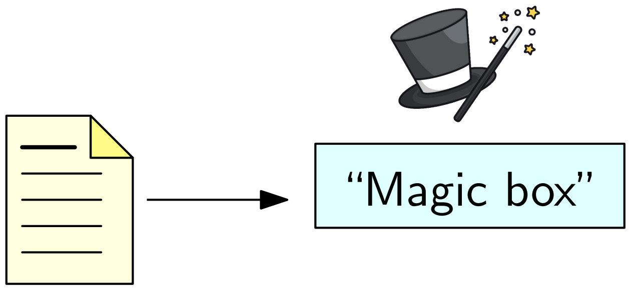
- A master encryption key 
- Your bitcoin wallet  
- Nuclear codes
- ...



Single point of failure!

An attacker can compromise one machine and steal the sensitive information

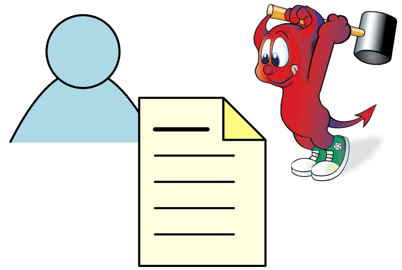
Idea: Share the information across several agents



Secret Sharing

Imagine some sensitive information that is kept by a single agent

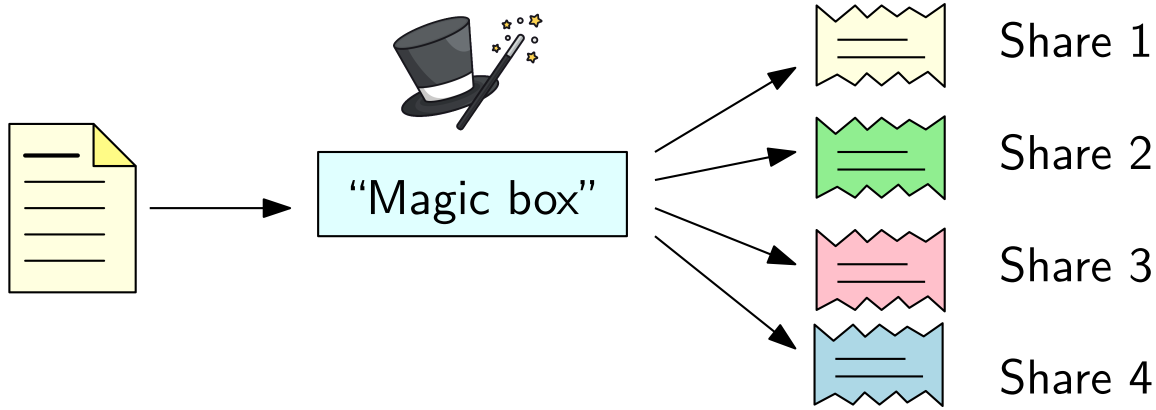
- A master encryption key
- Your bitcoin wallet
- Nuclear codes
- ...



Single point of failure!

An attacker can compromise one machine and steal the sensitive information

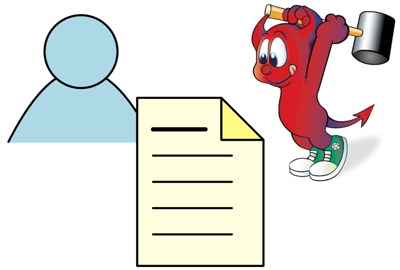
Idea: Share the information across several agents



Secret Sharing

Imagine some sensitive information that is kept by a single agent

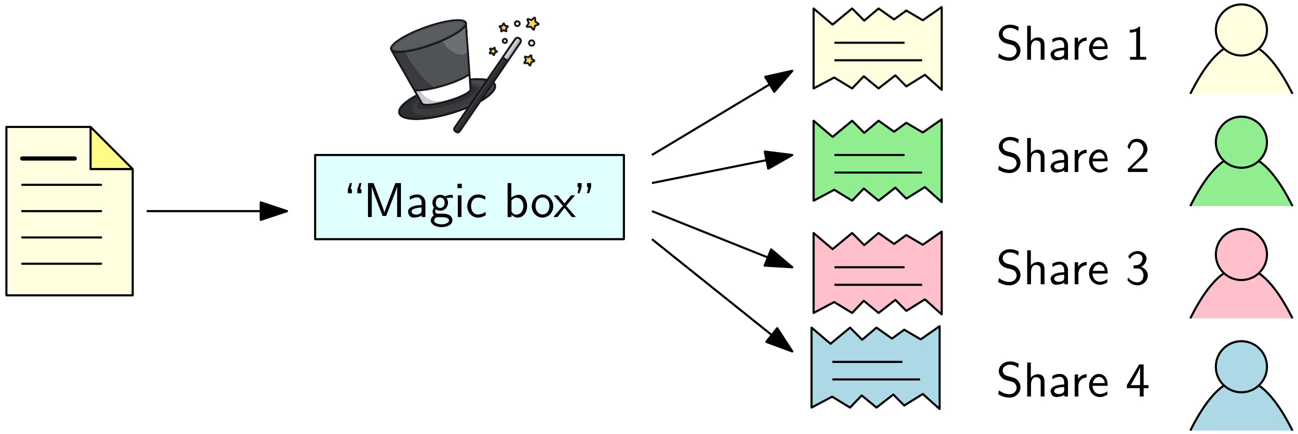
- A master encryption key
- Your bitcoin wallet
- Nuclear codes
- ...



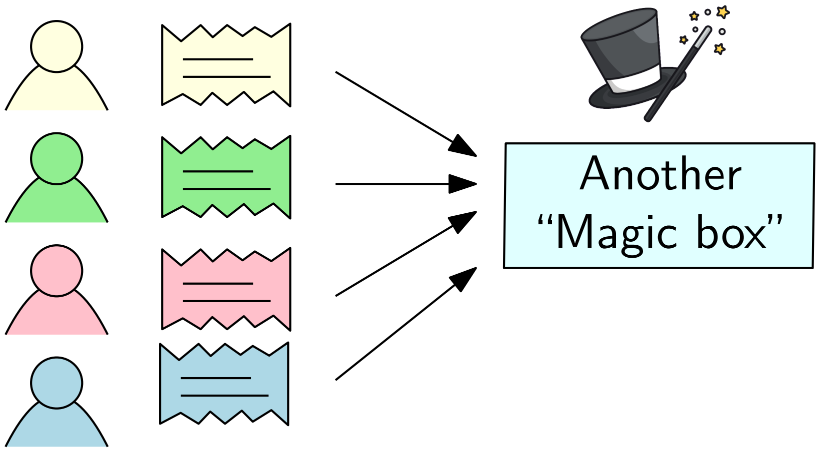
Single point of failure!

An attacker can compromise one machine and steal the sensitive information

Idea: Share the information across several agents



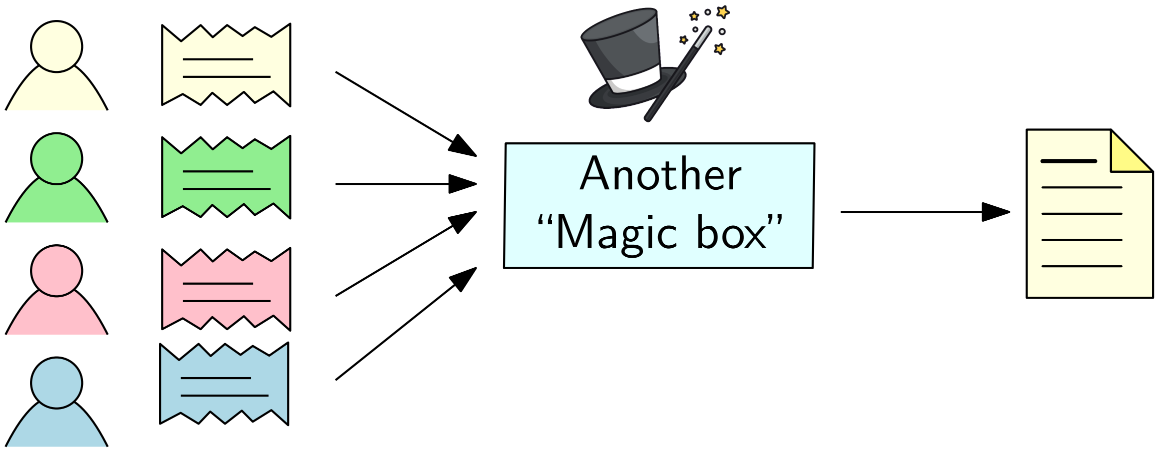
Secret Sharing



Idea:

- The shares of all agents can be used to reconstruct the secret

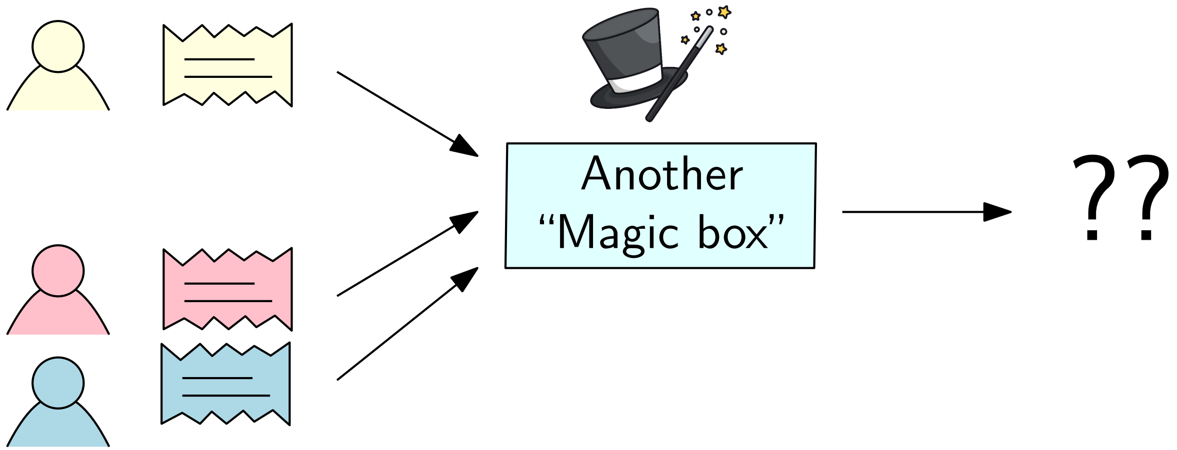
Secret Sharing



Idea:

- The shares of all agents can be used to reconstruct the secret

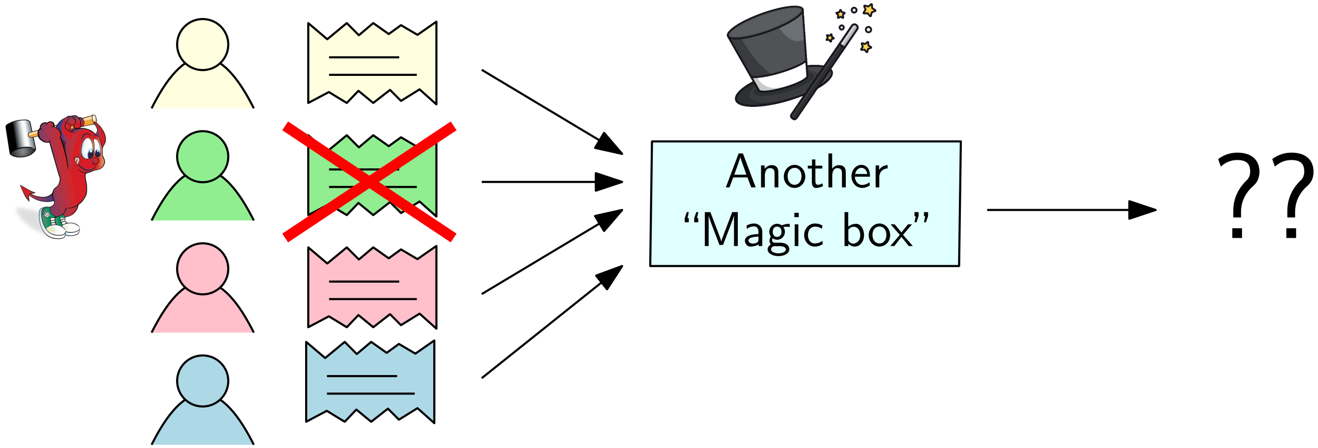
Secret Sharing



Idea:

- The shares of all agents can be used to reconstruct the secret
- The shares of any subset of agents look random and convey no information about the secret

Secret Sharing



Idea:

- The shares of all agents can be used to reconstruct the secret
- The shares of any subset of agents look random and convey no information about the secret

What if the adversary destroys a share?

Secret Sharing

What if the adversary destroys a share?

Idea:

- If there are n agents, any subset of at least k agents must be able to recover the secret.

Secret Sharing

What if the adversary destroys a share?

Idea:

- If there are n agents, any subset of at least k agents must be able to recover the secret.

- Any subset of $< t$ agent must not be able to gain any information about the secret

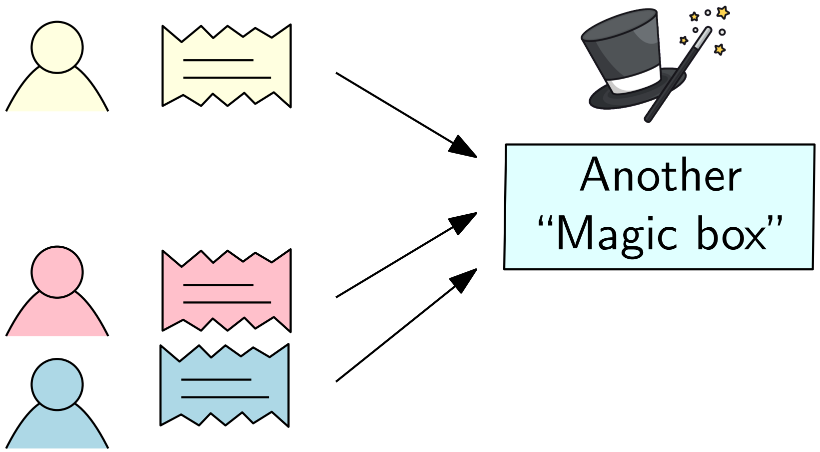
Secret Sharing

What if the adversary destroys a share?

Idea:

- If there are n agents, any subset of at least k agents must be able to recover the secret.
- Any subset of $< t$ agent must not be able to gain any information about the secret

Example: $n = 4$, $t = 3$



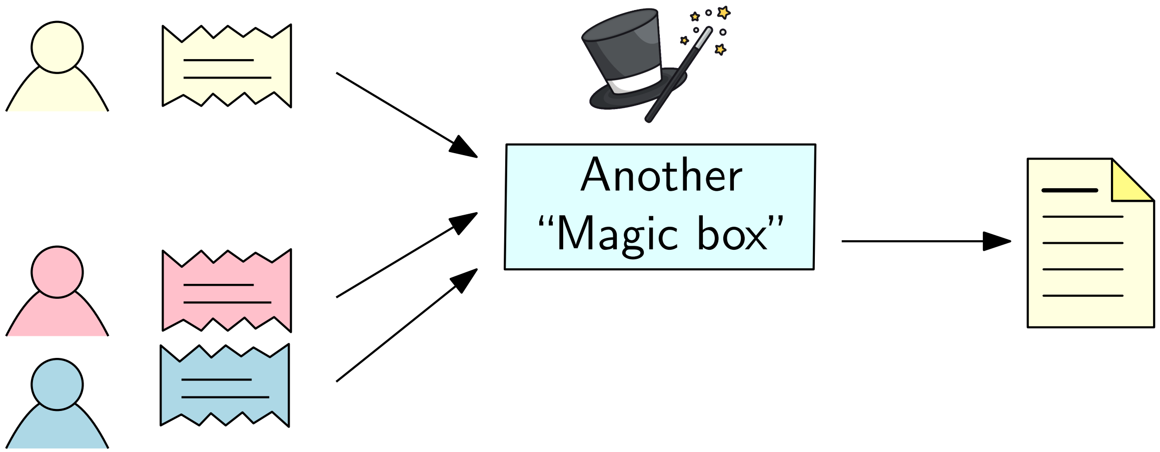
Secret Sharing

What if the adversary destroys a share?

Idea:

- If there are n agents, any subset of at least k agents must be able to recover the secret.
- Any subset of $< t$ agent must not be able to gain any information about the secret

Example: $n = 4$, $t = 3$



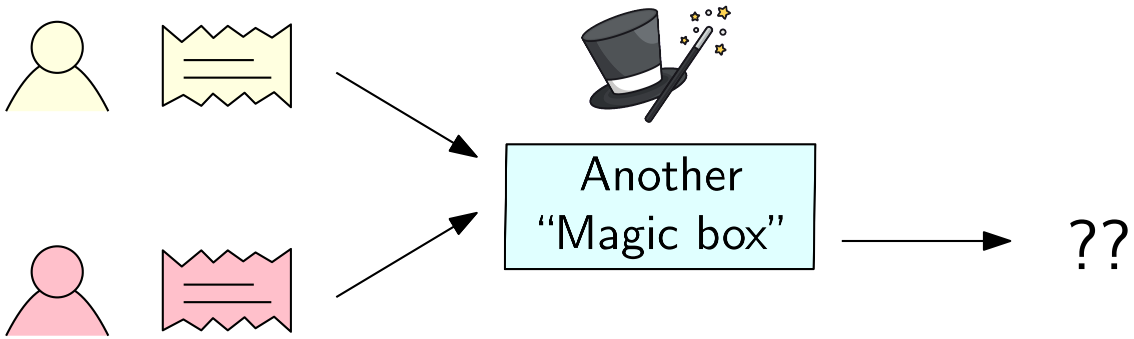
Secret Sharing

What if the adversary destroys a share?

Idea:

- If there are n agents, any subset of at least k agents must be able to recover the secret.
- Any subset of $< t$ agent must not be able to gain any information about the secret

Example: $n = 4$, $t = 3$



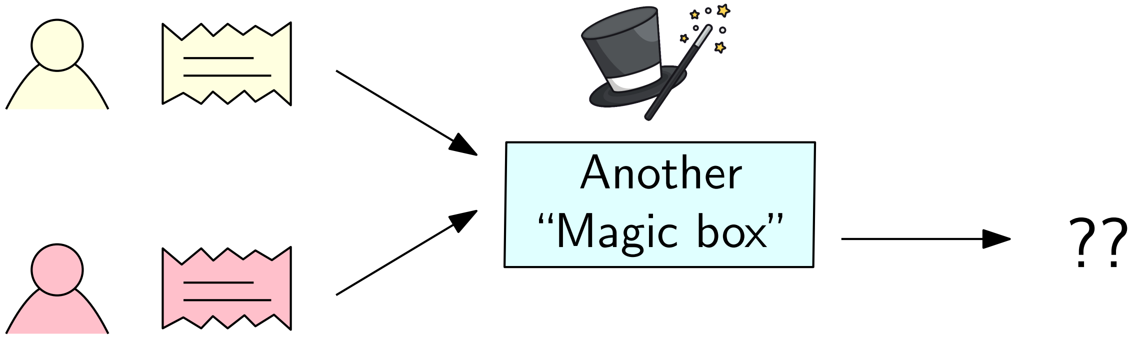
Secret Sharing

What if the adversary destroys a share?

Idea:

- If there are n agents, any subset of at least k agents must be able to recover the secret.
- Any subset of $< t$ agent must not be able to gain any information about the secret

Example: $n = 4$, $t = 3$



t**-out-of-**n **threshold secret-sharing scheme**

Access Structures and Qualifying Sets

Even more general:

- Let $\mathcal{A}$ be a set of n parties $a_1, \ldots, a_n$
- Let $\Gamma \subseteq 2^{\mathcal{A}}$ a collection of subsets of $\mathcal{A}$ such that:
  - — $\mathcal{A} \in \Gamma$
  - — Γ is an upward closed set w.r.t. set inclusion: if $A \in \Gamma$ and $A \subset B \subseteq \mathcal{A}$ then $B \in \Gamma$

Access Structures and Qualifying Sets

Even more general:

- Let $\mathcal{A}$ be a set of n parties $a_1, \ldots, a_n$
- Let $\Gamma \subseteq 2^{\mathcal{A}}$ a collection of subsets of $\mathcal{A}$ such that:
  - — $\mathcal{A} \in \Gamma$
  - — Γ is an upward closed set w.r.t. set inclusion: if $A \in \Gamma$ and $A \subset B \subseteq \mathcal{A}$ then $B \in \Gamma$
- The set Γ is a called a **(monotone) access structure**
- The sets $A \in \Gamma$ are called **qualifying sets**

Access Structures and Qualifying Sets

Even more general:

- Let $\mathcal{A}$ be a set of n parties $a_1, \ldots, a_n$
- Let $\Gamma \subseteq 2^{\mathcal{A}}$ a collection of subsets of $\mathcal{A}$ such that:
  - — $\mathcal{A} \in \Gamma$
  - — Γ is an upward closed set w.r.t. set inclusion: if $A \in \Gamma$ and $A \subset B \subseteq \mathcal{A}$ then $B \in \Gamma$
- The set Γ is a called a **(monotone) access structure**
- The sets $A \in \Gamma$ are called **qualifying sets**

Idea: A set $A \subseteq \mathcal{A}$ of parties should be able to recover the secret if and only if A is a qualifying set

Access Structures and Qualifying Sets

Even more general:

- Let $\mathcal{A}$ be a set of n parties $a_1, \ldots, a_n$
- Let $\Gamma \subseteq 2^{\mathcal{A}}$ a collection of subsets of $\mathcal{A}$ such that:
  - — $\mathcal{A} \in \Gamma$
  - — Γ is an upward closed set w.r.t. set inclusion: if $A \in \Gamma$ and $A \subset B \subseteq \mathcal{A}$ then $B \in \Gamma$

If A can recover the secret then any $B \supset A$ can recover the secret

- The set Γ is a called a **(monotone) access structure**
- The sets $A \in \Gamma$ are called **qualifying sets**

Idea: A set $A \subseteq \mathcal{A}$ of parties should be able to recover the secret if and only if A is a qualifying set

Access Structures and Qualifying Sets

Even more general:

- Let $\mathcal{A}$ be a set of n parties $a_1, \ldots, a_n$
- Let $\Gamma \subseteq 2^{\mathcal{A}}$ a collection of subsets of $\mathcal{A}$ such that:
  - — $\mathcal{A} \in \Gamma$
  - — Γ is an upward closed set w.r.t. set inclusion: if $A \in \Gamma$ and $A \subset B \subseteq \mathcal{A}$ then $B \in \Gamma$

If all parties come together, they must be able to recover the secret (otherwise $\Gamma = \emptyset$ and there is no point in sharing the secret)

If A can recover the secret then any $B \supset A$ can recover the secret

- The set Γ is a called a **(monotone) access structure**
- The sets $A \in \Gamma$ are called **qualifying sets**

Idea: A set $A \subseteq \mathcal{A}$ of parties should be able to recover the secret if and only if A is a qualifying set

Access Structures and Qualifying Sets

Even more general:

- Let $\mathcal{A}$ be a set of n parties $a_1, \ldots, a_n$
- Let $\Gamma \subseteq 2^{\mathcal{A}}$ a collection of subsets of $\mathcal{A}$ such that:
  - — $\mathcal{A} \in \Gamma$
  - — Γ is an upward closed set w.r.t. set inclusion: if $A \in \Gamma$ and $A \subset B \subseteq \mathcal{A}$ then $B \in \Gamma$

If all parties come together, they must be able to recover the secret (otherwise $\Gamma = \emptyset$ and there is no point in sharing the secret)

If A can recover the secret then any $B \supset A$ can recover the secret

- The set Γ is a called a **(monotone) access structure**
- The sets $A \in \Gamma$ are called **qualifying sets**

Idea: A set $A \subseteq \mathcal{A}$ of parties should be able to recover the secret if and only if A is a qualifying set

We can further assume that: $\forall a \in \mathcal{A}$ s,t, $\{a\} \notin \Gamma$ since otherwise we can simply send the secret to a and restrict ourselves to the access structure $\Gamma' = \{A \in \Gamma \mid a \notin A\}$

(this implies $\emptyset \notin \Gamma$)

Access Structures and Qualifying Sets

What's the access structure for a t-out-of-n threshold secret sharing scheme?

Access Structures and Qualifying Sets

What's the access structure for a t-out-of-n threshold secret sharing scheme?

$$\Gamma = \{A \in 2^{\mathcal{A}} \ : \ |A| \geq t\}$$

Access Structures and Qualifying Sets

What's the access structure for a t-out-of-n threshold secret sharing scheme?

$$\Gamma = \{A \in 2^{\mathcal{A}} \ : \ |A| \geq t\}$$

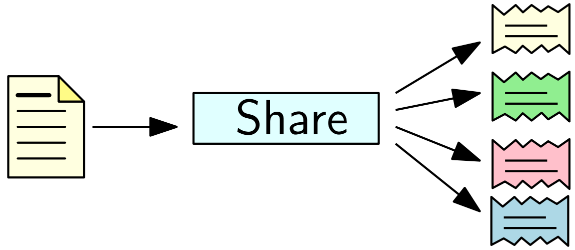
Example:

- $\mathcal{A} = \{$Alice, Bob, Charlie, Dan$\}$, $n = |\mathcal{A}| = 4$, $t = 2$
- $\Gamma = \{$ $\{$Alice, Bob$\}$, $\{$Alice, Charlie$\}$, $\{$Alice, Dan$\}$, $\{$Bob, Charlie$\}$, $\{$Bob, Dan$\}$, $\{$Charlie, Dan$\}$, $\{$Alice, Bob, Charlie$\}$, $\{$Alice, Bob, Dan$\}$, $\{$Alice, Charlie, Dan$\}$, $\{$Bob, Charlie, Dan$\}$, $\{$Alice, Bob, Charlie, Dan$\}$ $\}$

Access Structures and Qualifying Sets

Definition: A secret sharing scheme for a monotone access structure Γ over a set of parties $\mathcal{A}$ with respect to a space of secrets $\mathcal{S}$ is a pair of algorithms:

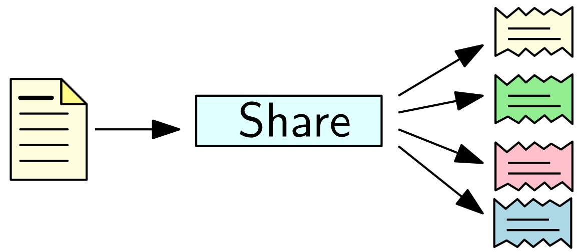
- Share(s, Γ): a (randomized) algorithm that takes a secret $s \in S$ and a monotone access structure Γ and outputs a value s_a for every $a \in \mathcal{A}$. The value s_a is called a's share of the secret.



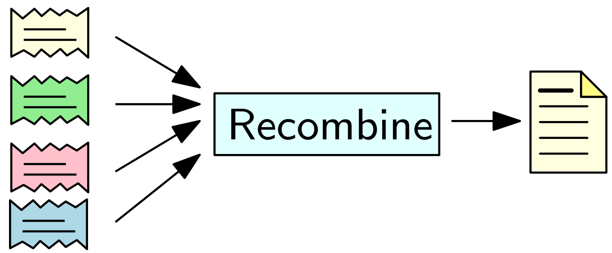
Access Structures and Qualifying Sets

Definition: A secret sharing scheme for a monotone access structure Γ over a set of parties $\mathcal{A}$ with respect to a space of secrets $\mathcal{S}$ is a pair of algorithms:

- Share(s, Γ): a (randomized) algorithm that takes a secret $s \in S$ and a monotone access structure Γ and outputs a value s_a for every $a \in \mathcal{A}$. The value s_a is called a's share of the secret.



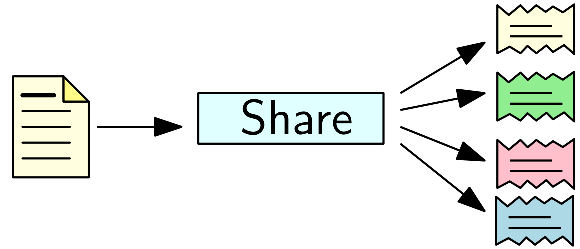
- Recombine(H): a deterministic algorithm that takes a set $H = \{s_a \mid a \in A\}$ containing a share for each party in some set $A \subseteq \mathcal{A}$ and outputs a secret $s \in \mathcal{S}$ if $A \in \Gamma$ and a failure symbol $\perp$ if $A \notin \Gamma$.



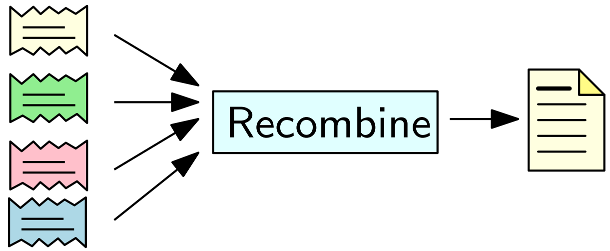
Access Structures and Qualifying Sets

Definition: A secret sharing scheme for a monotone access structure Γ over a set of parties $\mathcal{A}$ with respect to a space of secrets $\mathcal{S}$ is a pair of algorithms:

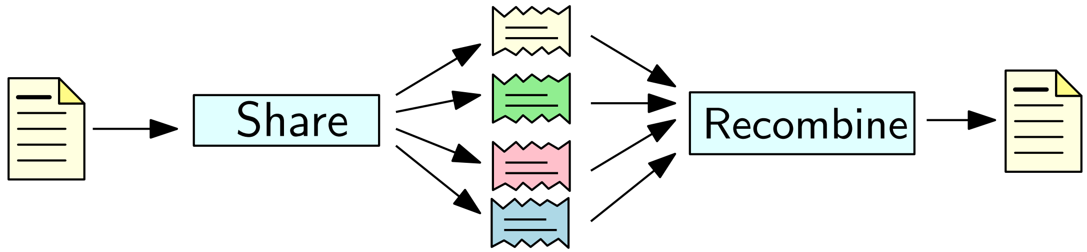
- Share(s, Γ): a (randomized) algorithm that takes a secret $s \in S$ and a monotone access structure Γ and outputs a value s_a for every $a \in \mathcal{A}$. The value s_a is called a's share of the secret.



- Recombine(H): a deterministic algorithm that takes a set $H = \{s_a \mid a \in A\}$ containing a share for each party in some set $A \subseteq \mathcal{A}$ and outputs a secret $s \in \mathcal{S}$ if $A \in \Gamma$ and a failure symbol $\perp$ if $A \notin \Gamma$.



Correctness: If $H = \{s_a \mid a \in A\}$ for a set $A \in \Gamma$ and all s_a were output by Share(s, Γ), then Recombine$(H) = s$.



Security Definition

A secret sharing scheme is **information theoretically secure** (or just secure) if no **(computationally unbounded) adversary** can learn anything about the underlying secret without having access to the shares of a qualifying set.

Security Definition

A secret sharing scheme is **information theoretically secure** (or just secure) if no **(computationally unbounded) adversary** can learn anything about the underlying secret without having access to the shares of a qualifying set.

Formalized similarly to perfect secrecy (there are multiple equivalent definitions):

A secret sharing scheme is secure if, for every $s, s' \in \mathcal{S}$, every access structure Γ, every $A \subset \mathcal{A}$ with $A \notin \Gamma$, and every vector of shares $\alpha = (\alpha_a)_{a \in A}$:

$$\Pr[(S_a)_{a \in A} = \alpha] = \Pr[(S'_a)_{a \in A} = \alpha],$$

where S_a (resp. S'_a) is a random variable representing the share given to the party $a \in A$ by Share(Γ, s) (resp. Share(Γ, s'))

Secret sharing with 2 parties

Consider $\mathcal{A} = \{a, b\}$. What are the possible access structures?

Secret sharing with 2 parties

Consider $\mathcal{A} = \{a, b\}$. What are the possible access structures?

$\Gamma = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$

$\Gamma = \{\{a\}, \{b\}, \{a, b\}\}$

$\Gamma = \{\{a\}, \{a, b\}\}$

$\Gamma = \{\{b\}, \{a, b\}\}$

$\Gamma = \{\{a, b\}\}$

Secret sharing with 2 parties

Consider $\mathcal{A} = \{a, b\}$. What are the possible access structures?

$\Gamma = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$ No secret sharing needed

$\Gamma = \{\{a\}, \{b\}, \{a, b\}\}$ No secret sharing needed

$\Gamma = \{\{a\}, \{a, b\}\}$

$\Gamma = \{\{b\}, \{a, b\}\}$

$\Gamma = \{\{a, b\}\}$

Secret sharing with 2 parties

Consider $\mathcal{A} = \{a, b\}$. What are the possible access structures?

$\Gamma = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$ No secret sharing needed

$\Gamma = \{\{a\}, \{b\}, \{a, b\}\}$ No secret sharing needed

$\Gamma = \{\{a\}, \{a, b\}\}$ Trivial: send secret to a and nothing to b

$\Gamma = \{\{b\}, \{a, b\}\}$ Trivial: send secret to b and nothing to a

$\Gamma = \{\{a, b\}\}$

Secret sharing with 2 parties

Consider $\mathcal{A} = \{a, b\}$. What are the possible access structures?

$\Gamma = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$ No secret sharing needed

$\Gamma = \{\{a\}, \{b\}, \{a, b\}\}$ No secret sharing needed

$\Gamma = \{\{a\}, \{a, b\}\}$ Trivial: send secret to a and nothing to b

$\Gamma = \{\{b\}, \{a, b\}\}$ Trivial: send secret to b and nothing to a

$\Gamma = \{\{a, b\}\}$ This is the only interesting case

2-out-of-2 threshold secret-sharing scheme

Secret sharing with 2 parties

Consider $\mathcal{A} = \{a, b\}$. What are the possible access structures?

$\Gamma = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$ No secret sharing needed

$\Gamma = \{\{a\}, \{b\}, \{a, b\}\}$ No secret sharing needed

$\Gamma = \{\{a\}, \{a, b\}\}$ Trivial: send secret to a and nothing to b

$\Gamma = \{\{b\}, \{a, b\}\}$ Trivial: send secret to b and nothing to a

$\Gamma = \{\{a, b\}\}$ This is the only interesting case

2-out-of-2 threshold secret-sharing scheme

Let the space of secrets be $\mathcal{S} = \{0, 1\}^{\ell}$

- Share(s, Γ): choose r u.a.r. from $\{0, 1\}^{\ell}$. Return $s_a = r$ and $s_b = r \oplus s$.
- Recombine(H): if $|H| < 2$ return $\perp$. Otherwise $H = \{s_a, s_b\}$, return $s_a \oplus s_b$.

Secret sharing with 2 parties

Consider $\mathcal{A} = \{a, b\}$. What are the possible access structures?

$\Gamma = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$ No secret sharing needed

$\Gamma = \{\{a\}, \{b\}, \{a, b\}\}$ No secret sharing needed

$\Gamma = \{\{a\}, \{a, b\}\}$ Trivial: send secret to a and nothing to b

$\Gamma = \{\{b\}, \{a, b\}\}$ Trivial: send secret to b and nothing to a

$\Gamma = \{\{a, b\}\}$ This is the only interesting case

2-out-of-2 threshold secret-sharing scheme

Let the space of secrets be $\mathcal{S} = \{0, 1\}^\ell$

- Share(s, Γ): choose r u.a.r. from $\{0, 1\}^\ell$. Return $s_a = r$ and $s_b = r \oplus s$.
- Recombine(H): if $|H| < 2$ return $\perp$. Otherwise $H = \{s_a, s_b\}$, return $s_a \oplus s_b$.

Correctness: $s_a \oplus s_b = r \oplus (r \oplus s) = s$.

2-out-of-2 threshold secret sharing: security

Let $s, s' \in \{0,1\}^\ell$ be two arbitrary secrets and consider S_a, S_b output by Share(s, Γ) (resp. S'_a, S'_b output by Share(s', Γ)).

2-out-of-2 threshold secret sharing: security

Let $s, s' \in \{0,1\}^\ell$ be two arbitrary secrets and consider S_a, S_b output by Share(s, Γ) (resp. S'_a, S'_b output by Share(s', Γ)).

- If $A = \{a\}$, then for an arbitrary $\alpha = (\alpha_a)$:

$$\Pr[S_a = \alpha_a] = \Pr[r = \alpha_a] = \frac{1}{2^\ell}$$

2-out-of-2 threshold secret sharing: security

Let $s, s' \in \{0,1\}^\ell$ be two arbitrary secrets and consider S_a, S_b output by Share(s, Γ) (resp. S'_a, S'_b output by Share(s', Γ)).

- If $A = \{a\}$, then for an arbitrary $\alpha = (\alpha_a)$:

$$\Pr[S_a = \alpha_a] = \Pr[r = \alpha_a] = \frac{1}{2^\ell}$$

$$||$$

$$\Pr[S'_a = \alpha_a] = \Pr[r = \alpha_a] = \frac{1}{2^\ell}$$

2-out-of-2 threshold secret sharing: security

Let $s, s' \in \{0,1\}^\ell$ be two arbitrary secrets and consider S_a, S_b output by Share(s, Γ) (resp. S'_a, S'_b output by Share(s', Γ)).

- If $A = \{a\}$, then for an arbitrary $\alpha = (\alpha_a)$:

$$\Pr[S_a = \alpha_a] = \Pr[r = \alpha_a] = \frac{1}{2^\ell}$$

$$\|$$

$$\Pr[S'_a = \alpha_a] = \Pr[r = \alpha_a] = \frac{1}{2^\ell}$$

- If $A = \{b\}$, then for an arbitrary $\alpha = (\alpha_b)$:

$$\Pr[S_b = \alpha_b] = \Pr[r \oplus s = \alpha_b]$$

2-out-of-2 threshold secret sharing: security

Let $s, s' \in \{0,1\}^\ell$ be two arbitrary secrets and consider S_a, S_b output by Share(s, Γ) (resp. S'_a, S'_b output by Share(s', Γ)).

- If $A = \{a\}$, then for an arbitrary $\alpha = (\alpha_a)$:

$$\Pr[S_a = \alpha_a] = \Pr[r = \alpha_a] = \tfrac{1}{2^\ell}$$

$$||$$

$$\Pr[S'_a = \alpha_a] = \Pr[r = \alpha_a] = \tfrac{1}{2^\ell}$$

- If $A = \{b\}$, then for an arbitrary $\alpha = (\alpha_b)$:

$$\Pr[S_b = \alpha_b] = \Pr[r \oplus s = \alpha_b] \; = \Pr[r = \alpha_b \oplus s]$$

2-out-of-2 threshold secret sharing: security

Let $s, s' \in \{0,1\}^\ell$ be two arbitrary secrets and consider S_a, S_b output by Share(s, Γ) (resp. S'_a, S'_b output by Share(s', Γ)).

- If $A = \{a\}$, then for an arbitrary $\alpha = (\alpha_a)$:

$$\Pr[S_a = \alpha_a] = \Pr[r = \alpha_a] = \frac{1}{2^\ell}$$

$$||$$

$$\Pr[S'_a = \alpha_a] = \Pr[r = \alpha_a] = \frac{1}{2^\ell}$$

- If $A = \{b\}$, then for an arbitrary $\alpha = (\alpha_b)$:

$$\Pr[S_b = \alpha_b] = \Pr[r \oplus s = \alpha_b] \;= \Pr[r = \alpha_b \oplus s] \;= \frac{1}{2^\ell}$$

2-out-of-2 threshold secret sharing: security

Let $s, s' \in \{0,1\}^\ell$ be two arbitrary secrets and consider S_a, S_b output by Share(s, Γ) (resp. S'_a, S'_b output by Share(s', Γ)).

- If $A = \{a\}$, then for an arbitrary $\alpha = (\alpha_a)$:

$$\Pr[S_a = \alpha_a] = \Pr[r = \alpha_a] = \frac{1}{2^\ell}$$

$$\|$$

$$\Pr[S'_a = \alpha_a] = \Pr[r = \alpha_a] = \frac{1}{2^\ell}$$

- If $A = \{b\}$, then for an arbitrary $\alpha = (\alpha_b)$:

$$\Pr[S_b = \alpha_b] = \Pr[r \oplus s = \alpha_b] = \Pr[r = \alpha_b \oplus s] = \frac{1}{2^\ell}$$

$$\|$$

$$\Pr[S'_b = \alpha_b] = \Pr[r \oplus s' = \alpha_b] = \Pr[r = \alpha_b \oplus s'] = \frac{1}{2^\ell}$$

2-out-of-2 threshold secret sharing: security

Let $s, s' \in \{0,1\}^\ell$ be two arbitrary secrets and consider S_a, S_b output by Share(s, Γ) (resp. S'_a, S'_b output by Share(s', Γ)).

- If $A = \{a\}$, then for an arbitrary $\alpha = (\alpha_a)$:

$$\begin{array}{c} \Pr[S_a = \alpha_a] = \Pr[r = \alpha_a] = \frac{1}{2^\ell} \\ \qquad\qquad\qquad\qquad\qquad\qquad\quad \| \\ \Pr[S'_a = \alpha_a] = \Pr[r = \alpha_a] = \frac{1}{2^\ell} \end{array}$$

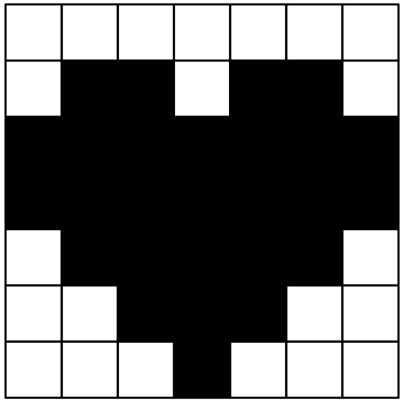
- If $A = \{b\}$, then for an arbitrary $\alpha = (\alpha_b)$:

$$\begin{array}{c} \Pr[S_b = \alpha_b] = \Pr[r \oplus s = \alpha_b] \;\; = \Pr[r = \alpha_b \oplus s] \;\; = \frac{1}{2^\ell} \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \| \\ \Pr[S'_b = \alpha_b] = \Pr[r \oplus s' = \alpha_b] = \Pr[r = \alpha_b \oplus s'] \;\; = \frac{1}{2^\ell} \end{array}$$

We have shown show that, regardless of s, $\Pr[S_a = \alpha]$ and $\Pr[S_b = \alpha]$ are constants

2-out-of-2 threshold secret sharing: a visual interpretation

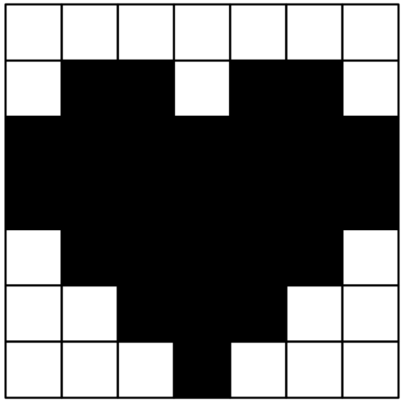
Imagine that the secret s is the following image:



s

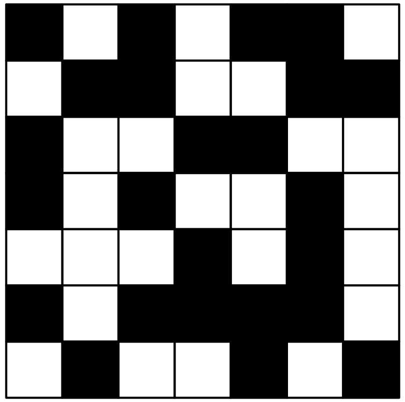
2-out-of-2 threshold secret sharing: a visual interpretation

Imagine that the secret s is the following image:



s

$=$

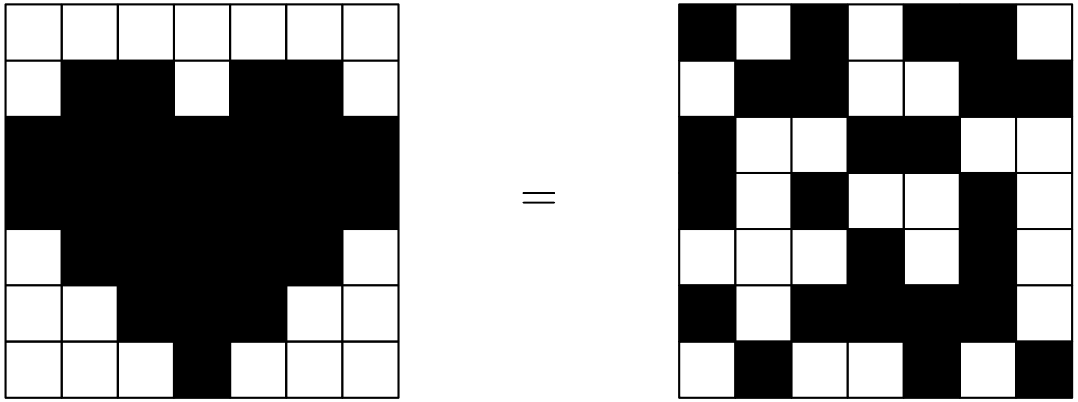


s_a

We generate the first share by coloring each pixel white or black u.a.r.

2-out-of-2 threshold secret sharing: a visual interpretation

Imagine that the secret s is the following image:



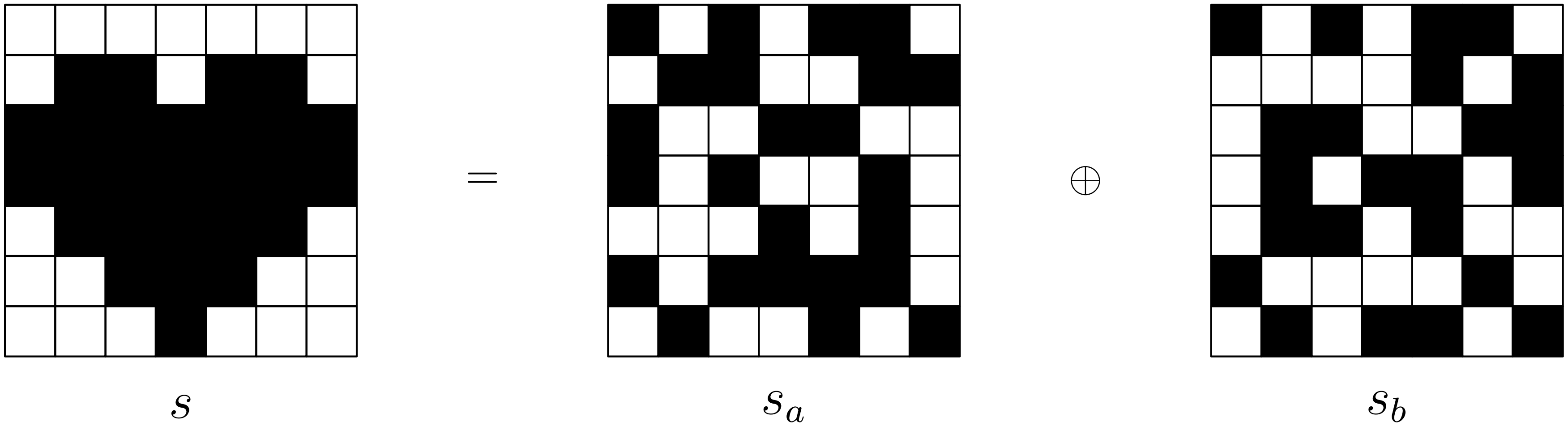
s $\qquad$ s_a

We generate the first share by coloring each pixel white or black u.a.r.

We generate the second share by XOR-ing each pixel of the secret with the corresponding pixel of the first share

2-out-of-2 threshold secret sharing: a visual interpretation

Imagine that the secret s is the following image:

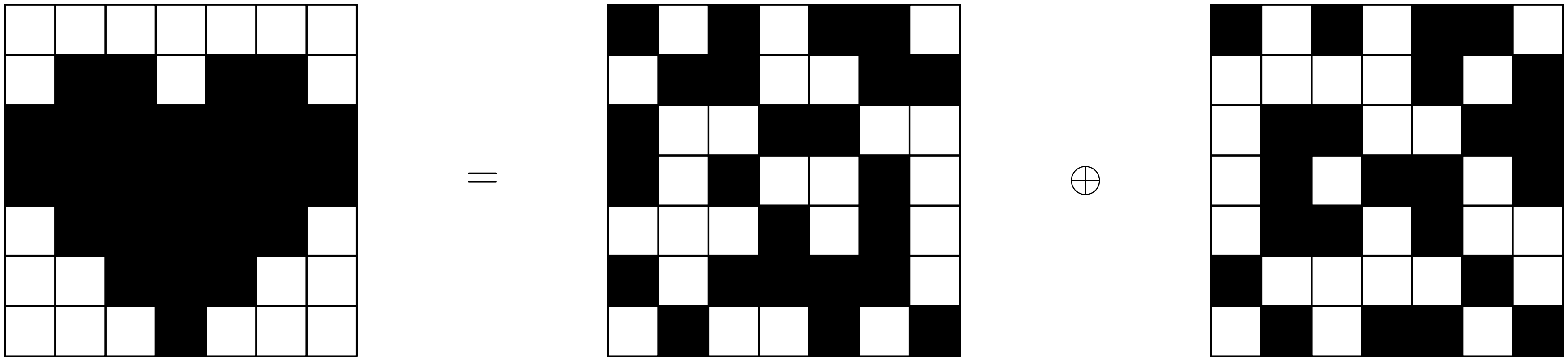


We generate the first share by coloring each pixel white or black u.a.r.

We generate the second share by XOR-ing each pixel of the secret with the corresponding pixel of the first share

2-out-of-2 threshold secret sharing: a visual interpretation

Imagine that the secret s is the following image:

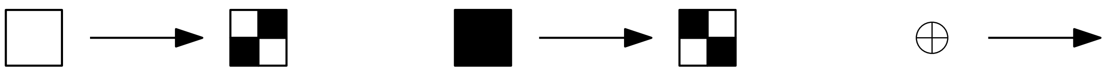


s $\qquad$ s_a $\qquad$ s_b

We generate the first share by coloring each pixel white or black u.a.r.

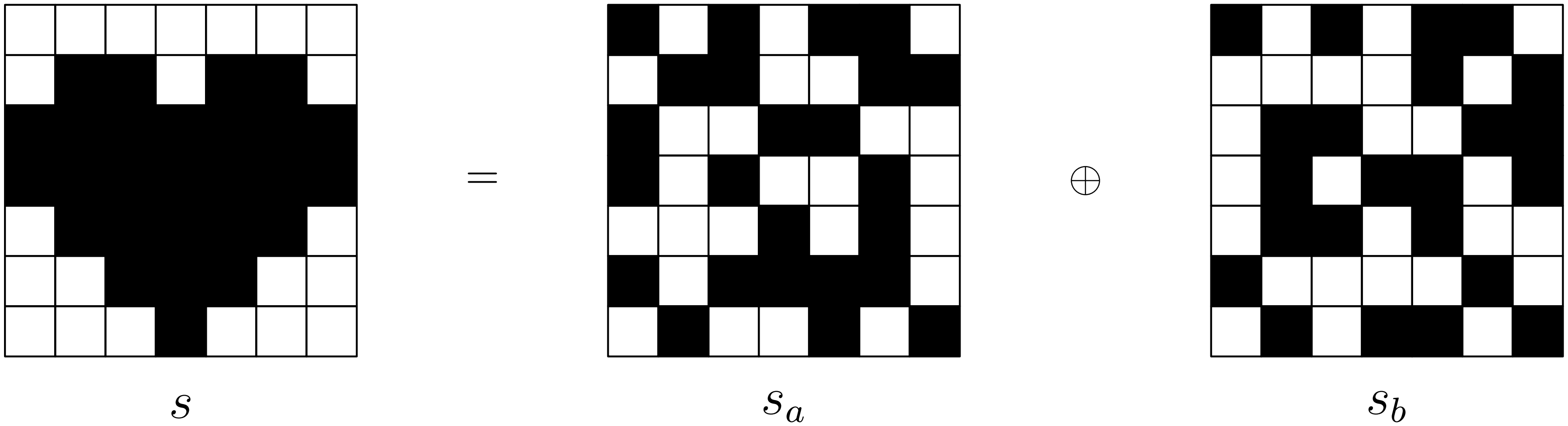
We generate the second share by XOR-ing each pixel of the secret with the corresponding pixel of the first share

Physical visual 2-out-of-2 threshold secret sharing scheme: subdivide each pixel in 4 subpixels

 overlay the two images

2-out-of-2 threshold secret sharing: a visual interpretation

Imagine that the secret s is the following image:



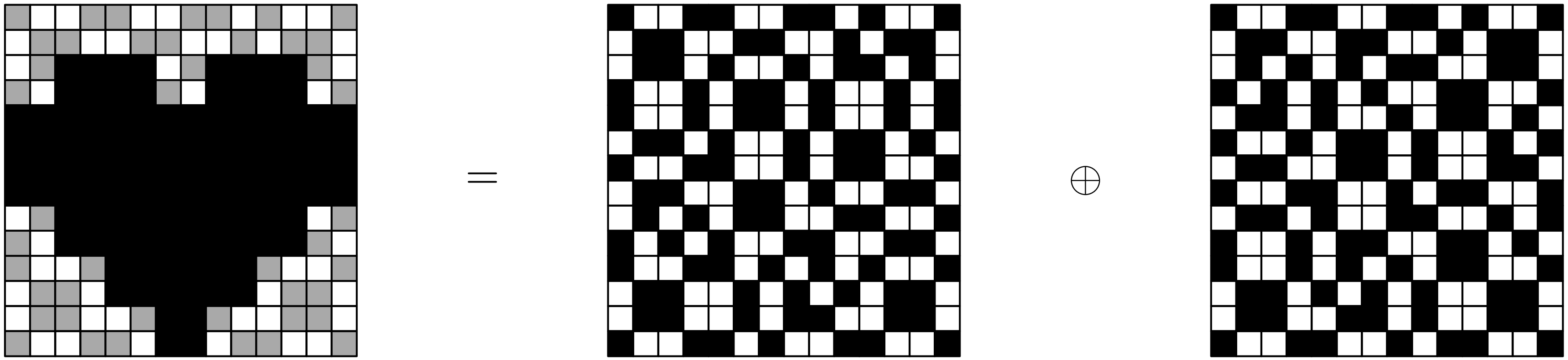
We generate the first share by coloring each pixel white or black u.a.r.

We generate the second share by XOR-ing each pixel of the secret with the corresponding pixel of the first share

Physical visual 2-out-of-2 threshold secret sharing scheme: subdivide each pixel in 4 subpixels

2-out-of-2 threshold secret sharing: a visual interpretation

Imagine that the secret s is the following image:



s s_a s_b

We generate the first share by coloring each pixel white or black u.a.r.

We generate the second share by XOR-ing each pixel of the secret with the corresponding pixel of the first share

Physical visual 2-out-of-2 threshold secret sharing scheme: subdivide each pixel in 4 subpixels

n-out-of-n threshold secret sharing

The above idea generalizes easily to $n \geq 2$ parties:

Consider any $\mathcal{A} = \{1, 2 \ldots, n\}$ with $|A| = n \geq 2$ and the access structure $\Gamma = \{\mathcal{A}\}$

Let the space of secrets be $\mathcal{S} = \{0, 1\}^\ell$

Index the parties with integers.
Makes notation easier.

n-out-of-n threshold secret sharing

The above idea generalizes easily to $n \geq 2$ parties:

Consider any $\mathcal{A} = \{1, 2 \ldots, n\}$ with $|A| = n \geq 2$ and the access structure $\Gamma = \{\mathcal{A}\}$

Let the space of secrets be $\mathcal{S} = \{0,1\}^\ell$

Index the parties with integers.
Makes notation easier.

Share(s, Γ):

- Let $r_1, \cdots r_{n-1}$ be $n-1$ strings chosen independent and u.a.r. from $\{0,1\}^\ell$.
- Return $(s_1, s_2, \ldots, s_n)$ where $s_i = r_i$ for $i < n$ and $s_n = r_1 \oplus r_2 \oplus \cdots \oplus r_{n-1} \oplus s$

n-out-of-n threshold secret sharing

The above idea generalizes easily to $n \geq 2$ parties:

Consider any $\mathcal{A} = \{1, 2 \ldots, n\}$ with $|A| = n \geq 2$ and the access structure $\Gamma = \{\mathcal{A}\}$

Let the space of secrets be $\mathcal{S} = \{0, 1\}^\ell$

Index the parties with integers.
Makes notation easier.

Share(s, Γ):

- Let $r_1, \cdots r_{n-1}$ be $n - 1$ strings chosen independent and u.a.r. from $\{0, 1\}^\ell$.
- Return $(s_1, s_2, \ldots, s_n)$ where $s_i = r_i$ for $i < n$ and $s_n = r_1 \oplus r_2 \oplus \cdots \oplus r_{n-1} \oplus s$

Recombine(H):

- If $|H| < n$ return $\perp$.
- Otherwise $H = \{s_1, s_2, \ldots, s_n\}$, return $s_1 \oplus s_2 \oplus \cdots \oplus s_n$.

n-out-of-n threshold secret sharing

The above idea generalizes easily to $n \geq 2$ parties:

Consider any $\mathcal{A} = \{1, 2 \ldots, n\}$ with $|A| = n \geq 2$ and the access structure $\Gamma = \{\mathcal{A}\}$

Let the space of secrets be $\mathcal{S} = \{0,1\}^\ell$

Index the parties with integers.
Makes notation easier.

Share(s, Γ):

- Let $r_1, \cdots r_{n-1}$ be $n-1$ strings chosen independent and u.a.r. from $\{0,1\}^\ell$.
- Return $(s_1, s_2, \ldots, s_n)$ where $s_i = r_i$ for $i < n$ and $s_n = r_1 \oplus r_2 \oplus \cdots \oplus r_{n-1} \oplus s$

Recombine(H):

- If $|H| < n$ return $\perp$.
- Otherwise $H = \{s_1, s_2, \ldots, s_n\}$, return $s_1 \oplus s_2 \oplus \cdots \oplus s_n$.

Correctness: $s_1 \oplus s_2 \oplus \cdots \oplus s_{n-1} \oplus s_n$

n-out-of-n threshold secret sharing

The above idea generalizes easily to $n \geq 2$ parties:

Consider any $\mathcal{A} = \{1, 2 \ldots, n\}$ with $|A| = n \geq 2$ and the access structure $\Gamma = \{\mathcal{A}\}$

Let the space of secrets be $\mathcal{S} = \{0, 1\}^\ell$

Index the parties with integers.
Makes notation easier.

Share(s, Γ):

- Let $r_1, \cdots r_{n-1}$ be $n - 1$ strings chosen independent and u.a.r. from $\{0, 1\}^\ell$.
- Return $(s_1, s_2, \ldots, s_n)$ where $s_i = r_i$ for $i < n$ and $s_n = r_1 \oplus r_2 \oplus \cdots \oplus r_{n-1} \oplus s$

Recombine(H):

- If $|H| < n$ return $\perp$.
- Otherwise $H = \{s_1, s_2, \ldots, s_n\}$, return $s_1 \oplus s_2 \oplus \cdots \oplus s_n$.

Correctness: $s_1 \oplus s_2 \oplus \cdots \oplus s_{n-1} \oplus s_n \quad = s_1 \oplus s_2 \oplus \cdots \oplus s_{n-1} \oplus (s_1 \oplus s_2 \oplus \ldots s_{n-1} \oplus s)$

n-out-of-n threshold secret sharing

The above idea generalizes easily to $n \geq 2$ parties:

Consider any $\mathcal{A} = \{1, 2 \ldots, n\}$ with $|A| = n \geq 2$ and the access structure $\Gamma = \{\mathcal{A}\}$

Let the space of secrets be $\mathcal{S} = \{0,1\}^\ell$

Index the parties with integers.
Makes notation easier.

Share(s, Γ):

- Let $r_1, \cdots r_{n-1}$ be $n-1$ strings chosen independent and u.a.r. from $\{0,1\}^\ell$.
- Return $(s_1, s_2, \ldots, s_n)$ where $s_i = r_i$ for $i < n$ and $s_n = r_1 \oplus r_2 \oplus \cdots \oplus r_{n-1} \oplus s$

Recombine(H):

- If $|H| < n$ return $\perp$.
- Otherwise $H = \{s_1, s_2, \ldots, s_n\}$, return $s_1 \oplus s_2 \oplus \cdots \oplus s_n$.

Correctness: $s_1 \oplus s_2 \oplus \cdots \oplus s_{n-1} \oplus s_n \;\; = s_1 \oplus s_2 \oplus \cdots \oplus s_{n-1} \oplus (s_1 \oplus s_2 \oplus \ldots s_{n-1} \oplus s) \;\; = s.$

Secret sharing with arbitrary access structures

Let Γ be an access structure (for an arbitrary number of parties n)

A qualifying set $B \in \Gamma$ is minimal if there is no qualifying set $B' \in \Gamma$ such that $B' \subset B$.

Let $m(\Gamma) = \{B_1, B_2, \ldots\}$ denote the set of all minimal qualifying sets in Γ

Secret sharing with arbitrary access structures

Let Γ be an access structure (for an arbitrary number of parties n)

A qualifying set $B \in \Gamma$ is minimal if there is no qualifying set $B' \in \Gamma$ such that $B' \subset B$.

Let $m(\Gamma) = \{B_1, B_2, \ldots\}$ denote the set of all minimal qualifying sets in Γ

Example:

- $\mathcal{A} = \{X, Y, W, Z\}$
- $\Gamma = \{\,\{X, Z\}, \{Y, W, Z\}, \{X, Y, Z\}, \{X, W, Z\}, \{X, Y, W, Z\}\,\}$
- $m(\Gamma) = \{\,\{X, Z\}, \{Y, W, Z\}\,\}$

Secret sharing with arbitrary access structures

Let Γ be an access structure (for an arbitrary number of parties n)

A qualifying set $B \in \Gamma$ is minimal if there is no qualifying set $B' \in \Gamma$ such that $B' \subset B$.

Let $m(\Gamma) = \{B_1, B_2, \dots\}$ denote the set of all minimal qualifying sets in Γ

Example:

- $\mathcal{A} = \{X, Y, W, Z\}$
- $\Gamma = \{\,\{X, Z\}, \{Y, W, Z\}, \{X, Y, Z\}, \{X, W, Z\}, \{X, Y, W, Z\}\,\}$
- $m(\Gamma) = \{\,\{X, Z\}, \{Y, W, Z\}\,\}$

If we think of a each party $a \in A$ as a Boolean variable, we can define the following Boolean formula in **disjunctive** normal form:

$$\bigvee_{B_i \in m(\Gamma)} \left(\bigwedge_{b \in B_i} b \right)$$

Each set B_i is a **clause** (conjunction of variables)

The formula is a disjunction of clauses

Secret sharing with arbitrary access structures

Let Γ be an access structure (for an arbitrary number of parties n)

A qualifying set $B \in \Gamma$ is minimal if there is no qualifying set $B' \in \Gamma$ such that $B' \subset B$.

Let $m(\Gamma) = \{B_1, B_2, \ldots\}$ denote the set of all minimal qualifying sets in Γ

Example:

- $\mathcal{A} = \{X, Y, W, Z\}$
- $\Gamma = \{\,\{X, Z\}, \{Y, W, Z\}, \{X, Y, Z\}, \{X, W, Z\}, \{X, Y, W, Z\}\,\}$
- $m(\Gamma) = \{\,\{X, Z\}, \{Y, W, Z\}\,\}$

$(X \wedge Z) \vee (Y \wedge W \wedge Z)$

If we think of a each party $a \in A$ as a Boolean variable, we can define the following Boolean formula in **disjunctive** normal form:

$$\bigvee_{B_i \in m(\Gamma)} \left(\bigwedge_{b \in B_i} b \right)$$

Each set B_i is a **clause** (conjunction of variables)

The formula is a disjunction of clauses

Secret sharing with arbitrary access structures

Let Γ be an access structure (for an arbitrary number of parties n)

A qualifying set $B \in \Gamma$ is minimal if there is no qualifying set $B' \in \Gamma$ such that $B' \subset B$.

Let $m(\Gamma) = \{B_1, B_2, \ldots\}$ denote the set of all minimal qualifying sets in Γ

Example:

- $\mathcal{A} = \{X, Y, W, Z\}$
- $\Gamma = \{\,\{X, Z\}, \{Y, W, Z\}, \{X, Y, Z\}, \{X, W, Z\}, \{X, Y, W, Z\}\,\}$
- $m(\Gamma) = \{\,\{X, Z\}, \{Y, W, Z\}\,\}$

$(X \wedge Z) \vee (Y \wedge W \wedge Z)$

If we think of a each party $a \in A$ as a Boolean variable, we can define the following Boolean formula in **disjunctive** normal form:

$$\bigvee_{B_i \in m(\Gamma)} \left(\bigwedge_{b \in B_i} b \right)$$

Each set B_i is a **clause** (conjunction of variables)

The formula is a disjunction of clauses

A set A of parties induces a truth assignment in which a is true iff $a \in A$

The truth assignment satisfies the formula if and only if A is a qualifying set

Ito–Nishizeki–Saito Secret Sharing

Share:

We can read the DNF formula as a set of instruction to build the shares s_a, $a \in \mathcal{A}$

- Each clause B_i corresponds to an "inner" $|B_i|$-out-of-$|B_i|$ threshold secret sharing scheme

  Each agent $b \in B_i$ gets a share $s_b^{(i)}$

Ito–Nishizeki–Saito Secret Sharing

Share:

We can read the DNF formula as a set of instruction to build the shares s_a, $a \in \mathcal{A}$

- Each clause B_i corresponds to an "inner" $|B_i|$-out-of-$|B_i|$ threshold secret sharing scheme

  Each agent $b \in B_i$ gets a share $s_b^{(i)}$

  E.g., for $B_1 = \{X, Z\}$ we pick a random string for $s_X^{(1)}$ and set $s_Z^{(1)} = s \oplus s_X^{(1)}$

  For $B_2 = \{Y, W, Z\}$ we pick random strings for $s_Y^{(2)}$ and $s_W^{(2)}$ and set $s_Z^{(2)} = s \oplus s_Y^{(2)} \oplus s_W^{(2)}$

Ito–Nishizeki–Saito Secret Sharing

Share:

We can read the DNF formula as a set of instruction to build the shares s_a, $a \in \mathcal{A}$

- Each clause B_i corresponds to an "inner" $|B_i|$-out-of-$|B_i|$ threshold secret sharing scheme

  Each agent $b \in B_i$ gets a share $s_b^{(i)}$

  E.g., for $B_1 = \{X, Z\}$ we pick a random string for $s_X^{(1)}$ and set $s_Z^{(1)} = s \oplus s_X^{(1)}$

  For $B_2 = \{Y, W, Z\}$ we pick random strings for $s_Y^{(2)}$ and $s_W^{(2)}$ and set $s_Z^{(2)} = s \oplus s_Y^{(2)} \oplus s_W^{(2)}$

- The "or" operators denote concatenation of the inner shares of each player

Ito–Nishizeki–Saito Secret Sharing

Share:

We can read the DNF formula as a set of instruction to build the shares s_a, $a \in \mathcal{A}$

- Each clause B_i corresponds to an "inner" $|B_i|$-out-of-$|B_i|$ threshold secret sharing scheme

  Each agent $b \in B_i$ gets a share $s_b^{(i)}$

  E.g., for $B_1 = \{X, Z\}$ we pick a random string for $s_X^{(1)}$ and set $s_Z^{(1)} = s \oplus s_X^{(1)}$

  For $B_2 = \{Y, W, Z\}$ we pick random strings for $s_Y^{(2)}$ and $s_W^{(2)}$ and set $s_Z^{(2)} = s \oplus s_Y^{(2)} \oplus s_W^{(2)}$

- The "or" operators denote concatenation of the inner shares of each player

  E.g., we combine the shares of the two clauses $(X \wedge Z) \vee (Y \wedge W \wedge Z)$ to obtain $s_X = s_X^{(1)}$, $s_Y = s_Y^{(2)}$, $s_W = s_W^{(2)}$, and $s_Z = s_Z^{(1)} \| s_Z^{(2)}$

Ito–Nishizeki–Saito Secret Sharing

Share:

We can read the DNF formula as a set of instruction to build the shares s_a, $a \in \mathcal{A}$

- Each clause B_i corresponds to an "inner" $|B_i|$-out-of-$|B_i|$ threshold secret sharing scheme

  Each agent $b \in B_i$ gets a share $s_b^{(i)}$

  E.g., for $B_1 = \{X, Z\}$ we pick a random string for $s_X^{(1)}$ and set $s_Z^{(1)} = s \oplus s_X^{(1)}$

  For $B_2 = \{Y, W, Z\}$ we pick random strings for $s_Y^{(2)}$ and $s_W^{(2)}$ and set $s_Z^{(2)} = s \oplus s_Y^{(2)} \oplus s_W^{(2)}$

- The "or" operators denote concatenation of the inner shares of each player

  E.g., we combine the shares of the two clauses $(X \wedge Z) \vee (Y \wedge W \wedge Z)$ to obtain $s_X = s_X^{(1)}$, $s_Y = s_Y^{(2)}$, $s_W = s_W^{(2)}$, and $s_Z = s_Z^{(1)} \| s_Z^{(2)}$

Recombine & Correctness:

If A is a qualifying set, then there is some clause consisting only of variables in A.

The parties involved in the clause can recover s using the Recombine step of the corresponding k-out-of-k threshold secret sharing scheme

Shamir Secret Sharing

The previous secret sharing scheme can produce shares that are much larger than the secret s

One notable example where this happens is the t-out-of-n case

- If $t = \frac{n}{2}$ there are $\binom{n}{n/2} = \Omega(2^n/\sqrt{n})$ minimal qualifying sets
- The shares are exponentially longer than the secret!

Shamir Secret Sharing

The previous secret sharing scheme can produce shares that are much larger than the secret s

One notable example where this happens is the t-out-of-n case

- If $t = \frac{n}{2}$ there are $\binom{n}{n/2} = \Omega(2^n/\sqrt{n})$ minimal qualifying sets
- The shares are exponentially longer than the secret!

Shamir proposed a secret t-out-of-n threshold secret-sharing scheme in which all the shares have (approximately) the same length as the secret

The scheme uses Lagrange interpolating polynomials

Lagrange interpolating polynomials

Consider a set $\{(x_1, y_1), \ldots, (x_k, y_k)\}$ of k points in $\mathbb{R}^2$ with distinct x_is.

We want to build a polynomial f that "passes through" all the points (i.e., $f(x_i) = y_i$ for $i = 1, \ldots, k$)

Lagrange interpolating polynomials

Consider a set $\{(x_1, y_1), \ldots, (x_k, y_k)\}$ of k points in $\mathbb{R}^2$ with distinct x_is.

We want to build a polynomial f that "passes through" all the points (i.e., $f(x_i) = y_i$ for $i = 1, \ldots, k$)

Consider the polynomial:

$$\ell_1(x) = (x - x_2)(x_1 - x_2)^{-1} \cdot (x - x_3)(x_1 - x_3)^{-1} \cdot \ldots \cdot (x - x_k)(x_1 - x_k)^{-1}$$

What happens when ℓ_1 is evaluated at the points $x_1, x_2, \ldots, x_k$?

Lagrange interpolating polynomials

Consider a set $\{(x_1, y_1), \ldots, (x_k, y_k)\}$ of k points in $\mathbb{R}^2$ with distinct x_is.

We want to build a polynomial f that "passes through" all the points (i.e., $f(x_i) = y_i$ for $i = 1, \ldots, k$)

Consider the polynomial:

$$\ell_1(x) = (x - x_2)(x_1 - x_2)^{-1} \cdot (x - x_3)(x_1 - x_3)^{-1} \cdot \ldots \cdot (x - x_k)(x_1 - x_k)^{-1}$$

What happens when ℓ_1 is evaluated at the points $x_1, x_2, \ldots, x_k$?

- If $x = x_1$ then each $(x - x_i)(x_1 - x_i)^{-1}$ evaluates to $1 \implies \ell_1(x_1) = 1$

Lagrange interpolating polynomials

Consider a set $\{(x_1, y_1), \dots, (x_k, y_k)\}$ of k points in $\mathbb{R}^2$ with distinct x_is.

We want to build a polynomial f that "passes through" all the points (i.e., $f(x_i) = y_i$ for $i = 1, \dots, k$)

Consider the polynomial:

$$\ell_1(x) = (x - x_2)(x_1 - x_2)^{-1} \cdot (x - x_3)(x_1 - x_3)^{-1} \cdot \dots \cdot (x - x_k)(x_1 - x_k)^{-1}$$

What happens when ℓ_1 is evaluated at the points $x_1, x_2, \dots, x_k$?

- If $x = x_1$ then each $(x - x_i)(x_1 - x_i)^{-1}$ evaluates to $1 \implies \ell_1(x_1) = 1$
- If $x = x_i$ for $i \neq 1$ then the product includes $(x - x_i) = 0 \implies \ell_1(x_i) = 0$

Lagrange interpolating polynomials

Consider a set $\{(x_1, y_1), \ldots, (x_k, y_k)\}$ of k points in $\mathbb{R}^2$ with distinct x_is.

We want to build a polynomial f that "passes through" all the points (i.e., $f(x_i) = y_i$ for $i = 1, \ldots, k$)

Consider the polynomial:

$$\ell_1(x) = (x - x_2)(x_1 - x_2)^{-1} \cdot (x - x_3)(x_1 - x_3)^{-1} \cdot \ldots \cdot (x - x_k)(x_1 - x_k)^{-1}$$

What happens when ℓ_1 is evaluated at the points $x_1, x_2, \ldots, x_k$?

- If $x = x_1$ then each $(x - x_i)(x_1 - x_i)^{-1}$ evaluates to $1 \implies \ell_1(x_1) = 1$
- If $x = x_i$ for $i \neq 1$ then the product includes $(x - x_i) = 0 \implies \ell_1(x_i) = 0$

We can generalize this to all j: $\quad \ell_j(x) = \prod_{\substack{i=1,\ldots,k \\ i \neq j}} (x - x_i)(x_j - x_i)^{-1}$

Lagrange interpolating polynomials

Consider a set $\{(x_1, y_1), \ldots, (x_k, y_k)\}$ of k points in $\mathbb{R}^2$ with distinct x_is.

We want to build a polynomial f that "passes through" all the points (i.e., $f(x_i) = y_i$ for $i = 1, \ldots, k$)

Consider the polynomial:

$$\ell_1(x) = (x - x_2)(x_1 - x_2)^{-1} \cdot (x - x_3)(x_1 - x_3)^{-1} \cdot \ldots \cdot (x - x_k)(x_1 - x_k)^{-1}$$

What happens when ℓ_1 is evaluated at the points $x_1, x_2, \ldots, x_k$?

- If $x = x_1$ then each $(x - x_i)(x_1 - x_i)^{-1}$ evaluates to $1 \implies \ell_1(x_1) = 1$
- If $x = x_i$ for $i \neq 1$ then the product includes $(x - x_i) = 0 \implies \ell_1(x_i) = 0$

We can generalize this to all j: $\ell_j(x) = \prod_{\substack{i=1,\ldots,k \\ i \neq j}} (x - x_i)(x_j - x_i)^{-1}$

$$\ell_j(x_i) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

Lagrange interpolating polynomials

The collection of polynomials $\ell_1(x), \ldots, \ell_k(x)$ is called a **Lagrange basis**

Lagrange interpolating polynomials

The collection of polynomials $\ell_1(x), \ldots, \ell_k(x)$ is called a **Lagrange basis**

Consider the polynomial:

$$f(x) = y_1\ell_1(x) + y_2\ell_2(x) + \cdots + y_k\ell_k(x)$$

What's the value of $f(x_i)$?

Lagrange interpolating polynomials

The collection of polynomials $\ell_1(x), \ldots, \ell_k(x)$ is called a **Lagrange basis**

Consider the polynomial:

$$f(x) = y_1\ell_1(x) + y_2\ell_2(x) + \cdots + y_k\ell_k(x)$$

What's the value of $f(x_i)$?

- If $j \neq i$ then $y_j\ell_j(x_i) = 0$
- For $i = j$ we have $y_i\ell_i(x_i) = y_i \cdot 1 = y_i$

Lagrange interpolating polynomials

The collection of polynomials $\ell_1(x), \ldots, \ell_k(x)$ is called a **Lagrange basis**

Consider the polynomial:

$$f(x) = y_1\ell_1(x) + y_2\ell_2(x) + \cdots + y_k\ell_k(x)$$

What's the value of $f(x_i)$?

$$\left.\begin{array}{l} \bullet \text{ If } j \neq i \text{ then } y_j\ell_j(x_i) = 0 \\ \\ \bullet \text{ For } i = j \text{ we have } y_i\ell_i(x_i) = y_i \cdot 1 = y_i \end{array}\right\} f(x_i) = y_i$$

Lagrange interpolating polynomials

The collection of polynomials $\ell_1(x), \dots, \ell_k(x)$ is called a **Lagrange basis**

Consider the polynomial:

$$f(x) = y_1\ell_1(x) + y_2\ell_2(x) + \cdots + y_k\ell_k(x)$$

What's the value of $f(x_i)$?

- If $j \neq i$ then $y_j\ell_j(x_i) = 0$
- For $i = j$ we have $y_i\ell_i(x_i) = y_i \cdot 1 = y_i$

$\Bigg\} \; f(x_i) = y_i$

$f(x)$ is called the **Lagrange interpolating polynomial**

Lagrange interpolating polynomials

The collection of polynomials $\ell_1(x), \ldots, \ell_k(x)$ is called a **Lagrange basis**

Consider the polynomial:

$$f(x) = y_1\ell_1(x) + y_2\ell_2(x) + \cdots + y_k\ell_k(x)$$

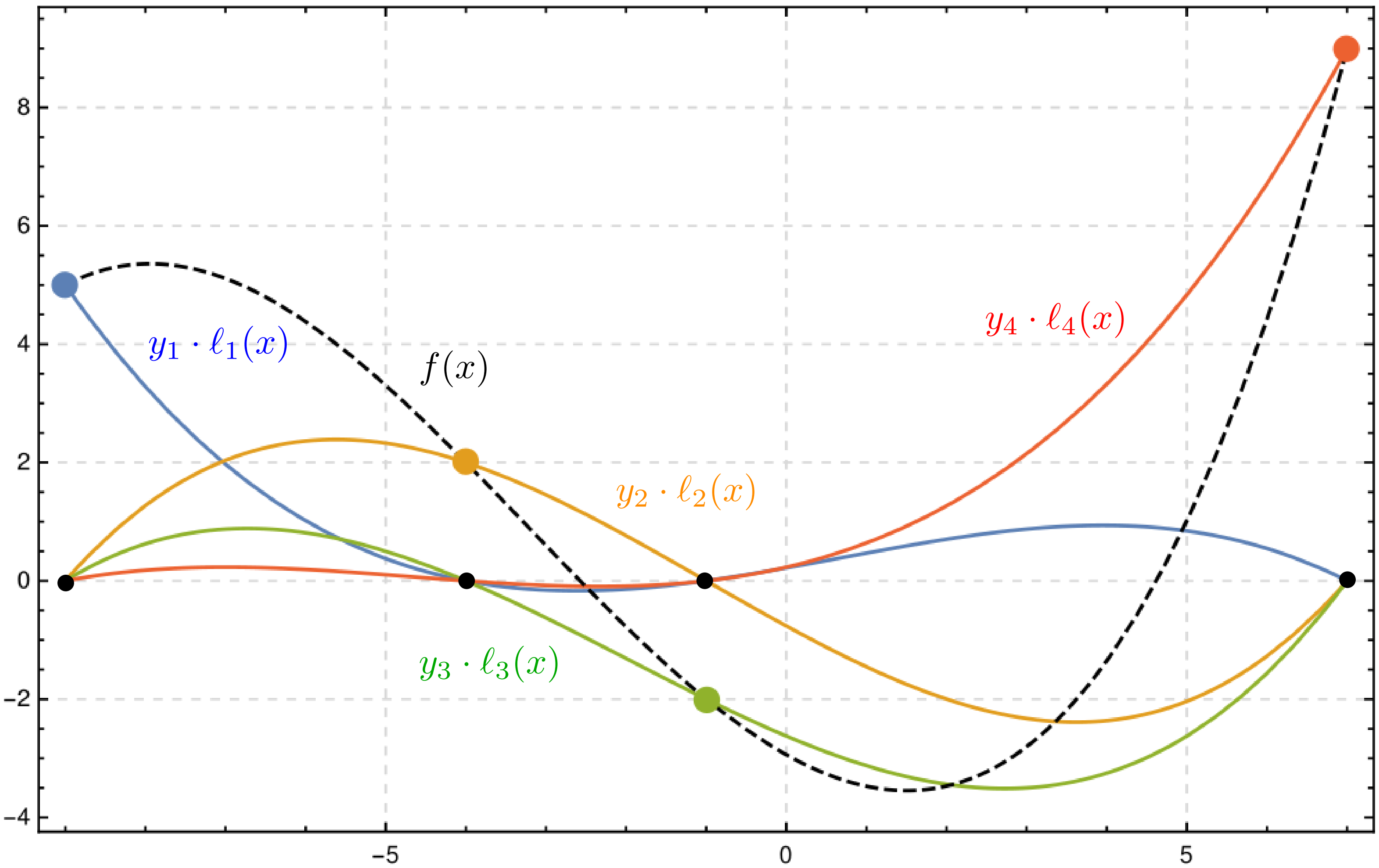
What's the value of $f(x_i)$?

$$\left.\begin{array}{l} \bullet \text{ If } j \neq i \text{ then } y_j\ell_j(x_i) = 0 \\ \\ \bullet \text{ For } i = j \text{ we have } y_i\ell_i(x_i) = y_i \cdot 1 = y_i \end{array}\right\} f(x_i) = y_i$$

$f(x)$ is called the **Lagrange interpolating polynomial**

- Each ℓ_j is the product of $k-1$ terms $(x - x_i)$ (and some constants), therefore ℓ_j has degree $k-1$
- $f(x)$ is a sum of polynomials of degree $k-1$, therefore $f(x)$ has degree $k-1$

Lagrange interpolating polynomials



Lagrange interpolating polynomials

Theorem: there is a unique polynomial $f(x)$ of degree at most $k-1$ with real coefficients such that $f(x_i) = y_i$ for all $i = 1, \ldots, k$.

Lagrange interpolating polynomials

Theorem: there is a unique polynomial $f(x)$ of degree at most $k-1$ with real coefficients such that $f(x_i) = y_i$ for all $i = 1, \ldots, k$.

Proof:

- We have already shown that a polynomial $f(x)$ of degree $k-1$ exists, we just need to argue that it is unique
- Let $g(x)$ be any polynomial of degree at most $k-1$ such that $g(x_i) = y_i$ for all i

Lagrange interpolating polynomials

Theorem: there is a unique polynomial $f(x)$ of degree at most $k-1$ with real coefficients such that $f(x_i) = y_i$ for all $i = 1, \ldots, k$.

Proof:

- We have already shown that a polynomial $f(x)$ of degree $k-1$ exists, we just need to argue that it is unique
- Let $g(x)$ be any polynomial of degree at most $k-1$ such that $g(x_i) = y_i$ for all i
- The polynomial $h(x) = g(x) - f(x)$ has degree at most $k-1$ and satisfies $h(x_i) = 0$ for all i

Lagrange interpolating polynomials

Theorem: there is a unique polynomial $f(x)$ of degree at most $k-1$ with real coefficients such that $f(x_i) = y_i$ for all $i = 1, \ldots, k$.

Proof:

- We have already shown that a polynomial $f(x)$ of degree $k-1$ exists, we just need to argue that it is unique
- Let $g(x)$ be any polynomial of degree at most $k-1$ such that $g(x_i) = y_i$ for all i
- The polynomial $h(x) = g(x) - f(x)$ has degree at most $k-1$ and satisfies $h(x_i) = 0$ for all i
- $h(x)$ has k roots and degree $k-1$

Lagrange interpolating polynomials

Theorem: there is a unique polynomial $f(x)$ of degree at most $k-1$ with real coefficients such that $f(x_i) = y_i$ for all $i = 1, \ldots, k$.

Proof:

- We have already shown that a polynomial $f(x)$ of degree $k-1$ exists, we just need to argue that it is unique
- Let $g(x)$ be any polynomial of degree at most $k-1$ such that $g(x_i) = y_i$ for all i
- The polynomial $h(x) = g(x) - f(x)$ has degree at most $k-1$ and satisfies $h(x_i) = 0$ for all i
- $h(x)$ has k roots and degree $k-1$

Fundamental theorem of algebra: every non-zero, single-variable, degree d polynomial with complex coefficients has, counted with multiplicity, exactly d complex roots

Lagrange interpolating polynomials

Theorem: there is a unique polynomial $f(x)$ of degree at most $k-1$ with real coefficients such that $f(x_i) = y_i$ for all $i = 1, \ldots, k$.

Proof:

- We have already shown that a polynomial $f(x)$ of degree $k-1$ exists, we just need to argue that it is unique
- Let $g(x)$ be any polynomial of degree at most $k-1$ such that $g(x_i) = y_i$ for all i
- The polynomial $h(x) = g(x) - f(x)$ has degree at most $k-1$ and satisfies $h(x_i) = 0$ for all i
- $h(x)$ has k roots and degree $k-1 \implies h(x) = 0$

Fundamental theorem of algebra: every non-zero, single-variable, degree d polynomial with complex coefficients has, counted with multiplicity, exactly d complex roots

Lagrange interpolating polynomials

Theorem: there is a unique polynomial $f(x)$ of degree at most $k-1$ with real coefficients such that $f(x_i) = y_i$ for all $i = 1, \ldots, k$.

Proof:

- We have already shown that a polynomial $f(x)$ of degree $k-1$ exists, we just need to argue that it is unique
- Let $g(x)$ be any polynomial of degree at most $k-1$ such that $g(x_i) = y_i$ for all i
- The polynomial $h(x) = g(x) - f(x)$ has degree at most $k-1$ and satisfies $h(x_i) = 0$ for all i
- $h(x)$ has k roots and degree $k-1 \implies h(x) = 0 \implies g(x) = f(x)$

Fundamental theorem of algebra: every non-zero, single-variable, degree d polynomial with complex coefficients has, counted with multiplicity, exactly d complex roots

$\square$

Lagrange interpolating polynomials with coefficient over $\mathbb{Z}_p$

We will need to choose an interpolating polynomial **uniformly at random** to obtain a secure secret-sharing scheme

- Unclear how to do that over the reals
- Unclear how to represent a real number on a computer

Lagrange interpolating polynomials with coefficient over $\mathbb{Z}_p$

We will need to choose an interpolating polynomial **uniformly at random** to obtain a secure secret-sharing scheme

- Unclear how to do that over the reals
- Unclear how to represent a real number on a computer

Idea: we restrict ourselves to polynomials with coefficients over $\mathbb{Z}_p$

Lagrange interpolating polynomials with coefficient over $\mathbb{Z}_p$

We will need to choose an interpolating polynomial **uniformly at random** to obtain a secure secret-sharing scheme

- Unclear how to do that over the reals
- Unclear how to represent a real number on a computer

Idea: we restrict ourselves to polynomials with coefficients over $\mathbb{Z}_p$

A **field** is a set of elements together with two binary operations $(F, \oplus, \otimes)$ such that:

- $(F, \oplus)$ is an Abelian group, we call its identity element 0
- $(F \setminus \{0\}, \otimes)$ is an Abelian group
- The $\otimes$ operation distributes over the $\oplus$ operation: i.e., $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$

Lagrange interpolating polynomials with coefficient over $\mathbb{Z}_p$

We will need to choose an interpolating polynomial **uniformly at random** to obtain a secure secret-sharing scheme

- Unclear how to do that over the reals
- Unclear how to represent a real number on a computer

Idea: we restrict ourselves to polynomials with coefficients over $\mathbb{Z}_p$

A **field** is a set of elements together with two binary operations $(F, \oplus, \otimes)$ such that:

- $(F, \oplus)$ is an Abelian group, we call its identity element 0
- $(F \setminus \{0\}, \otimes)$ is an Abelian group
- The $\otimes$ operation distributes over the $\oplus$ operation: i.e., $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$

Good news:

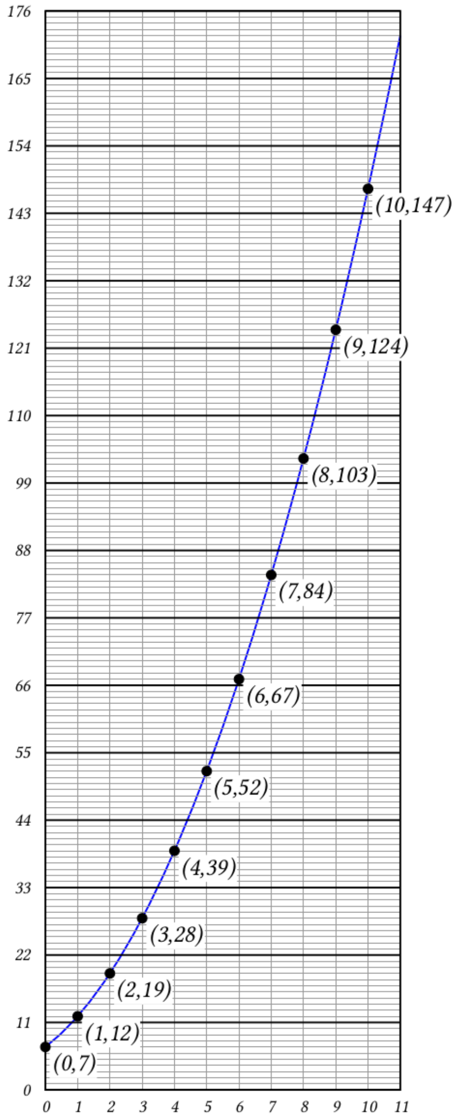
- The fundamental theorem of algebra can be extended to *univariate* polynomials *over a finite field*
- If p is prime then $(\mathbb{Z}_p, +, \cdot)$ is a finite filed

Lagrange interpolating polynomials with coefficient over $\mathbb{Z}_p$

Theorem: Let $\{(x_1, y_1), \ldots, (x_k, y_k)\}$ be a set of k points in $\mathbb{Z}_p \times \mathbb{Z}_p$ with distinct x_is. There is a unique polynomial $f(x)$ of degree at most $k-1$ with coefficients in $\mathbb{Z}_p$ such that $f(x_i) = y_i \pmod{p}$ for all $i = 1, \ldots, k$.

The construction and the proof of uniqueness are identical to the previous ones (where $-x$ and x^{-1} denote the additive and multiplicative inverses of x in $\mathbb{Z}_p$).

Lagrange interpolating polynomials with coefficient over $\mathbb{Z}_p$



Theorem: Let $\{(x_1, y_1), \ldots, (x_k, y_k)\}$ be a set of k points in $\mathbb{Z}_p \times \mathbb{Z}_p$ with distinct x_is. There is a unique polynomial $f(x)$ of degree at most $k-1$ with coefficients in $\mathbb{Z}_p$ such that $f(x_i) = y_i \pmod{p}$ for all $i = 1, \ldots, k$.

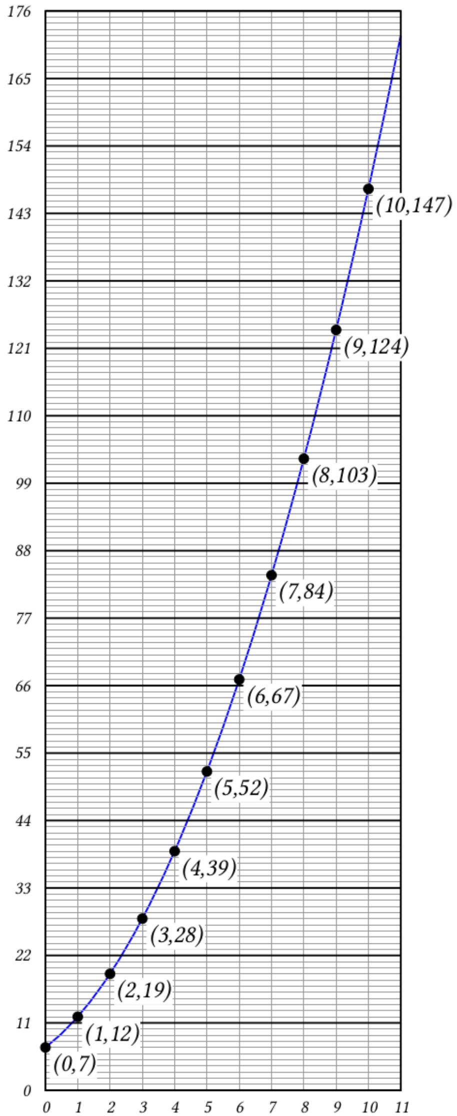
The construction and the proof of uniqueness are identical to the previous ones (where $-x$ and x^{-1} denote the additive and multiplicative inverses of x in $\mathbb{Z}_p$).

Example: $f(x) = x^2 + 4x + 7$

Source: Mike Rosulek, The Joy of Cryptography

Over the reals

Lagrange interpolating polynomials with coefficient over $\mathbb{Z}_p$



Theorem: Let $\{(x_1, y_1), \ldots, (x_k, y_k)\}$ be a set of k points in $\mathbb{Z}_p \times \mathbb{Z}_p$ with distinct x_is. There is a unique polynomial $f(x)$ of degree at most $k-1$ with coefficients in $\mathbb{Z}_p$ such that $f(x_i) = y_i \pmod{p}$ for all $i = 1, \ldots, k$.

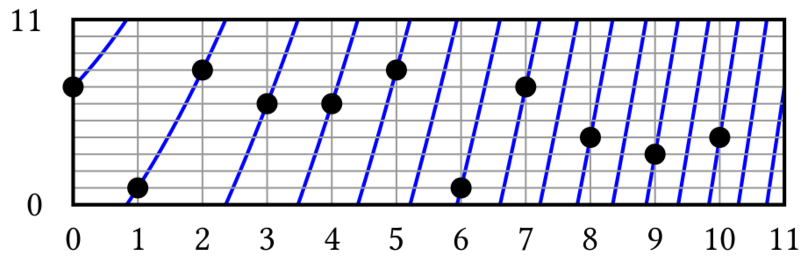
The construction and the proof of uniqueness are identical to the previous ones (where $-x$ and x^{-1} denote the additive and multiplicative inverses of x in $\mathbb{Z}_p$).

Example: $f(x) = x^2 + 4x + 7$

Source: Mike Rosulek, The Joy of Cryptography

Over the reals

Over $\mathbb{Z}_{11}$



Back to Shamir Secret Sharing

The set of parties is $\mathcal{A} = \{1, 2, \ldots, n\}$

The space of secrets $\mathcal{S}$ is $\mathbb{Z}_p$ for some prime number p

If the secret s is a binary number with t bits, we can pick a prime $p > \max\{s, n\}$ with $\Theta(t + \log n)$ bits.

The Shamir k-out-of-n threshold secret sharing scheme is as follows:

Share(s): (we omit the access structure, which is determined by k and n)

- Choose $k-1$ coefficients $\beta_1, \ldots, \beta_{k-1}$ independently and u.a.r. from $\mathbb{Z}_p$
- Define the polynomial: $f(x) = s + \sum_{i=1}^{k-1} \beta_i x^i$ (f is a random polynomial such that $f(0) = s$)
- For $i = 1, \ldots, n$:
    - Assign to party i the share $s_i = (i, f(i))$, where f is evaluated in $\mathbb{Z}_p$

Back to Shamir Secret Sharing

The set of parties is $\mathcal{A} = \{1, 2, \ldots, n\}$

The space of secrets $\mathcal{S}$ is $\mathbb{Z}_p$ for some prime number p

If the secret s is a binary number with t bits, we can pick a prime $p > \max\{s, n\}$ with $\Theta(t + \log n)$ bits.

The Shamir k-out-of-n threshold secret sharing scheme is as follows:

Share(s): (we omit the access structure, which is determined by k and n)

- Choose $k-1$ coefficients $\beta_1, \ldots, \beta_{k-1}$ independently and u.a.r. from $\mathbb{Z}_p$
- Define the polynomial: $f(x) = s + \sum_{i=1}^{k-1} \beta_i x^i$ (f is a random polynomial such that $f(0) = s$)
- For $i = 1, \ldots, n$:
  - Assign to party i the share $s_i = (i, f(i))$, where f is evaluated in $\mathbb{Z}_p$

Recombine($\{s_i \mid i \in A\}$) (A is a qualifying set)

- Compute the (unique) interpolating polynomial f (with coefficient in $\mathbb{Z}_p$) of degree $k-1$ such that $f(i) = s_i$
- Return $f(0)$

Shamir Secret Sharing: Example

Consider a set of $n = 5$ parties that want to share a secret $s = 8$ using Sharmir's 3-out-of-5 threshold secret sharing scheme

We will work in the field $\mathbb{Z}_{11}$

Shamir Secret Sharing: Example

Consider a set of $n = 5$ parties that want to share a secret $s = 8$ using Sharmir's 3-out-of-5 threshold secret sharing scheme

We will work in the field $\mathbb{Z}_{11}$

Sharing:

- We pick two random coefficients $\beta_1 = 4$, $\beta_2 = 7$
- The polynomial $f(x) = s + \beta_1 x + \beta_2 x^2$ is $8 + 4x + 7x^2$

Shamir Secret Sharing: Example

Consider a set of $n = 5$ parties that want to share a secret $s = 8$ using Sharmir's 3-out-of-5 threshold secret sharing scheme

We will work in the field $\mathbb{Z}_{11}$

Sharing:

- We pick two random coefficients $\beta_1 = 4$, $\beta_2 = 7$
- The polynomial $f(x) = s + \beta_1 x + \beta_2 x^2$ is $8 + 4x + 7x^2$
- The five shares are: $s_1 = (1, f(1)) = (1, 8)$ $s_2 = (2, f(2)) = (2, 0)$ $s_3 = (3, f(3)) = (3, 6)$ $s_4 = (4, f(4)) = (4, 4)$ $s_5 = (5, f(5)) = (5, 8)$

Shamir Secret Sharing: Example

Consider a set of $n = 5$ parties that want to share a secret $s = 8$ using Sharmir's 3-out-of-5 threshold secret sharing scheme

We will work in the field $\mathbb{Z}_{11}$

Sharing:

- We pick two random coefficients $\beta_1 = 4$, $\beta_2 = 7$
- The polynomial $f(x) = s + \beta_1 x + \beta_2 x^2$ is $8 + 4x + 7x^2$
- The five shares are: $s_1 = (1, f(1)) = (1, 8)$ $\quad s_2 = (2, f(2)) = (2, 0)$ $\quad s_3 = (3, f(3)) = (3, 6)$
  $s_4 = (4, f(4)) = (4, 4)$ $\quad s_5 = (5, f(5)) = (5, 8)$

Reconstructing the secret from the shares s_1, s_2, and s_4:

- $f(x) = 8 \cdot \ell_1(x) + 0 \cdot \ell_2(x) + 4 \cdot \ell_3(x)$

Shamir Secret Sharing: Example

Consider a set of $n = 5$ parties that want to share a secret $s = 8$ using Sharmir's 3-out-of-5 threshold secret sharing scheme

We will work in the field $\mathbb{Z}_{11}$

Sharing:

- We pick two random coefficients $\beta_1 = 4$, $\beta_2 = 7$
- The polynomial $f(x) = s + \beta_1 x + \beta_2 x^2$ is $8 + 4x + 7x^2$
- The five shares are: $s_1 = (1, f(1)) = (1, 8)$ $s_2 = (2, f(2)) = (2, 0)$ $s_3 = (3, f(3)) = (3, 6)$ $s_4 = (4, f(4)) = (4, 4)$ $s_5 = (5, f(5)) = (5, 8)$

Reconstructing the secret from the shares s_1, s_2, and s_4:

- $f(x) = 8 \cdot \ell_1(x) + 0 \cdot \ell_2(x) + 4 \cdot \ell_3(x)$

Shamir Secret Sharing: Example

Consider a set of $n = 5$ parties that want to share a secret $s = 8$ using Sharmir's 3-out-of-5 threshold secret sharing scheme

We will work in the field $\mathbb{Z}_{11}$

Sharing:

- We pick two random coefficients $\beta_1 = 4$, $\beta_2 = 7$
- The polynomial $f(x) = s + \beta_1 x + \beta_2 x^2$ is $8 + 4x + 7x^2$
- The five shares are: $s_1 = (1, f(1)) = (1, 8)$ $s_2 = (2, f(2)) = (2, 0)$ $s_3 = (3, f(3)) = (3, 6)$ $s_4 = (4, f(4)) = (4, 4)$ $s_5 = (5, f(5)) = (5, 8)$

Reconstructing the secret from the shares s_1, s_2, and s_4:

- $f(x) = 8 \cdot \ell_1(x) + 0 \cdot \ell_2(x) + 4 \cdot \ell_3(x)$
- $\ell_1(x) = (x-2)(1-2)^{-1} \cdot (x-4)(1-4)^{-1} = (x-2)10 \cdot (x-4)7 = 4x^2 + 9x + 10$
- $\ell_3(x) = (x-1)(4-1)^{-1} \cdot (x-2)(4-2)^{-1} = (x-1)4 \cdot (x-2)6 = 2x^2 + 5x + 4$

Shamir Secret Sharing: Example

Consider a set of $n = 5$ parties that want to share a secret $s = 8$ using Sharmir's 3-out-of-5 threshold secret sharing scheme

We will work in the field $\mathbb{Z}_{11}$

Sharing:

- We pick two random coefficients $\beta_1 = 4$, $\beta_2 = 7$
- The polynomial $f(x) = s + \beta_1 x + \beta_2 x^2$ is $8 + 4x + 7x^2$
- The five shares are: $s_1 = (1, f(1)) = (1, 8)$ $s_2 = (2, f(2)) = (2, 0)$ $s_3 = (3, f(3)) = (3, 6)$
  $s_4 = (4, f(4)) = (4, 4)$ $s_5 = (5, f(5)) = (5, 8)$

Reconstructing the secret from the shares s_1, s_2, and s_4:

- $f(x) = 8 \cdot \ell_1(x) + 0 \cdot \ell_2(x) + 4 \cdot \ell_3(x) = 8 \cdot (4x^2 + 9x + 10) + 4 \cdot (2x^2 + 5x + 4)$
- $\ell_1(x) = (x - 2)(1 - 2)^{-1} \cdot (x - 4)(1 - 4)^{-1} = (x-2)10 \cdot (x-4)7 = 4x^2 + 9x + 10$
- $\ell_3(x) = (x - 1)(4 - 1)^{-1} \cdot (x - 2)(4 - 2)^{-1} = (x - 1)4 \cdot (x - 2)6 = 2x^2 + 5x + 4$

Shamir Secret Sharing: Example

Consider a set of $n = 5$ parties that want to share a secret $s = 8$ using Sharmir's 3-out-of-5 threshold secret sharing scheme

We will work in the field $\mathbb{Z}_{11}$

Sharing:

- We pick two random coefficients $\beta_1 = 4$, $\beta_2 = 7$
- The polynomial $f(x) = s + \beta_1 x + \beta_2 x^2$ is $8 + 4x + 7x^2$
- The five shares are: $s_1 = (1, f(1)) = (1, 8)$ $s_2 = (2, f(2)) = (2, 0)$ $s_3 = (3, f(3)) = (3, 6)$ $s_4 = (4, f(4)) = (4, 4)$ $s_5 = (5, f(5)) = (5, 8)$

Reconstructing the secret from the shares s_1, s_2, and s_4:

- $f(x) = 8 \cdot \ell_1(x) + 0 \cdot \ell_2(x) + 4 \cdot \ell_3(x) = 8 \cdot (4x^2 + 9x + 10) + 4 \cdot (2x^2 + 5x + 4) = 7x^2 + 4x + 8$
- $\ell_1(x) = (x-2)(1-2)^{-1} \cdot (x-4)(1-4)^{-1} = (x-2)10 \cdot (x-4)7 = 4x^2 + 9x + 10$
- $\ell_3(x) = (x-1)(4-1)^{-1} \cdot (x-2)(4-2)^{-1} = (x-1)4 \cdot (x-2)6 = 2x^2 + 5x + 4$

Shamir Secret Sharing: Example

Consider a set of $n = 5$ parties that want to share a secret $s = 8$ using Sharmir's 3-out-of-5 threshold secret sharing scheme

We will work in the field $\mathbb{Z}_{11}$

Sharing:

- We pick two random coefficients $\beta_1 = 4$, $\beta_2 = 7$
- The polynomial $f(x) = s + \beta_1 x + \beta_2 x^2$ is $8 + 4x + 7x^2$
- The five shares are: $s_1 = (1, f(1)) = (1, 8)$ $s_2 = (2, f(2)) = (2, 0)$ $s_3 = (3, f(3)) = (3, 6)$
  $s_4 = (4, f(4)) = (4, 4)$ $s_5 = (5, f(5)) = (5, 8)$

Reconstructing the secret from the shares s_1, s_2, and s_4:

$f(0)$

- $f(x) = 8 \cdot \ell_1(x) + 0 \cdot \ell_2(x) + 4 \cdot \ell_3(x) = 8 \cdot (4x^2 + 9x + 10) + 4 \cdot (2x^2 + 5x + 4) = 7x^2 + 4x + 8$
- $\ell_1(x) = (x - 2)(1 - 2)^{-1} \cdot (x - 4)(1 - 4)^{-1} = (x - 2)10 \cdot (x - 4)7 = 4x^2 + 9x + 10$
- $\ell_3(x) = (x - 1)(4 - 1)^{-1} \cdot (x - 2)(4 - 2)^{-1} = (x - 1)4 \cdot (x - 2)6 = 2x^2 + 5x + 4$

Shamir Secret Sharing: Example

Consider a set of $n = 5$ parties that want to share a secret $s = 8$ using Sharmir's 3-out-of-5 threshold secret sharing scheme

We will work in the field $\mathbb{Z}_{11}$

Sharing:

- We pick two random coefficients $\beta_1 = 4$, $\beta_2 = 7$
- The polynomial $f(x) = s + \beta_1 x + \beta_2 x^2$ is $8 + 4x + 7x^2$
- The five shares are: $s_1 = (1, f(1)) = (1, 8)$ $s_2 = (2, f(2)) = (2, 0)$ $s_3 = (3, f(3)) = (3, 6)$
  $s_4 = (4, f(4)) = (4, 4)$ $s_5 = (5, f(5)) = (5, 8)$

Reconstructing the secret from the shares s_1, s_2, and s_4:

$f(0)$

- $f(x) = 8 \cdot \ell_1(x) + 0 \cdot \ell_2(x) + 4 \cdot \ell_3(x) = 8 \cdot (4x^2 + 9x + 10) + 4 \cdot (2x^2 + 5x + 4) = 7x^2 + 4x + 8$
- $\ell_1(x) = (x-2)(1-2)^{-1} \cdot (x-4)(1-4)^{-1} = (x-2)10 \cdot (x-4)7 = 4x^2 + 9x + 10$
- $\ell_3(x) = (x-1)(4-1)^{-1} \cdot (x-2)(4-2)^{-1} = (x-1)4 \cdot (x-2)6 = 2x^2 + 5x + 4$

Shamir Secret Sharing: Security

Let $A \subset \mathcal{A}$ be a non-qualifying set, and consider any vector $\alpha = (\alpha_i)_{i \in A}$.

Let $\eta(\alpha, s)$ be the number of polynomials (with coefficients in $\mathbb{Z}_p$) g of degree $k-1$ such that $g(i) = \alpha_i \pmod p$ for $i \in A$, and $g(0) = s \pmod p$.

Shamir Secret Sharing: Security

Let $A \subset \mathcal{A}$ be a non-qualifying set, and consider any vector $\alpha = (\alpha_i)_{i \in A}$.

Let $\eta(\alpha, s)$ be the number of polynomials (with coefficients in $\mathbb{Z}_p$) g of degree $k-1$ such that $g(i) = \alpha_i \pmod p$ for $i \in A$, and $g(0) = s \pmod p$.

The polynomial f is chosen u.a.r. among all the p^{k-1} polynomials (with coefficients in $\mathbb{Z}_p$) of degree $k-1$ such that $f(0) = s \pmod p$

Shamir Secret Sharing: Security

Let $A \subset \mathcal{A}$ be a non-qualifying set, and consider any vector $\alpha = (\alpha_i)_{i \in A}$.

Let $\eta(\alpha, s)$ be the number of polynomials (with coefficients in $\mathbb{Z}_p$) g of degree $k-1$ such that $g(i) = \alpha_i \pmod p$ for $i \in A$, and $g(0) = s \pmod p$.

The polynomial f is chosen u.a.r. among all the p^{k-1} polynomials (with coefficients in $\mathbb{Z}_p$) of degree $k-1$ such that $f(0) = s \pmod p$

$$\Pr[(S_i)_{i \in A} = \alpha] = \tfrac{\eta(\alpha,s)}{p^{k-1}} \qquad \stackrel{?}{=} \qquad \tfrac{\eta(\alpha,s')}{p^{k-1}} = \Pr[(S'_i)_{i \in A} = \alpha]$$

Shamir Secret Sharing: Security

Let $A \subset \mathcal{A}$ be a non-qualifying set, and consider any vector $\alpha = (\alpha_i)_{i \in A}$.

Let $\eta(\alpha, s)$ be the number of polynomials (with coefficients in $\mathbb{Z}_p$) g of degree $k-1$ such that $g(i) = \alpha_i \pmod p$ for $i \in A$, and $g(0) = s \pmod p$.

The polynomial f is chosen u.a.r. among all the p^{k-1} polynomials (with coefficients in $\mathbb{Z}_p$) of degree $k-1$ such that $f(0) = s \pmod p$

$$\Pr[(S_i)_{i \in A} = \alpha] = \frac{\eta(\alpha, s)}{p^{k-1}} \quad \stackrel{?}{=} \quad \frac{\eta(\alpha, s')}{p^{k-1}} = \Pr[(S'_i)_{i \in A} = \alpha]$$

We will show that these quantities do not depend on the secrets s and s'

Shamir Secret Sharing: Security

Let $A \subset \mathcal{A}$ be a non-qualifying set, and consider any vector $\alpha = (\alpha_i)_{i \in A}$.

Let $\eta(\alpha, s)$ be the number of polynomials (with coefficients in $\mathbb{Z}_p$) g of degree $k-1$ such that $g(i) = \alpha_i \pmod p$ for $i \in A$, and $g(0) = s \pmod p$.

The polynomial f is chosen u.a.r. among all the p^{k-1} polynomials (with coefficients in $\mathbb{Z}_p$) of degree $k-1$ such that $f(0) = s \pmod p$

$$\Pr[(S_i)_{i \in A} = \alpha] = \frac{\eta(\alpha,s)}{p^{k-1}} \stackrel{?}{=} \frac{\eta(\alpha,s')}{p^{k-1}} = \Pr[(S'_i)_{i \in A} = \alpha]$$

We will show that these quantities do not depend on the secrets s and s'

Theorem: Let $\{(x_1, y_1), \ldots, (x_h, y_h)\}$ be a set of $h \leq k$ points in $\mathbb{Z}_p \times \mathbb{Z}_p$ with distinct x_is, where $p \geq k$. The number of polynomials g of degree $k-1$ with coefficients in $\mathbb{Z}_p$ that such that $y_i = g(x_i) \pmod p$ for all $i = 1, \ldots, h$ is exactly p^{k-h}.

Shamir Secret Sharing: Security

Theorem: Let $\{(x_1, y_1), \ldots, (x_h, y_h)\}$ be a set of $h \leq k$ points in $\mathbb{Z}_p \times \mathbb{Z}_p$ with distinct x_is, where $p \geq k$. The number of polynomials g of degree $k-1$ with coefficients in $\mathbb{Z}_p$ that such that $y_i = g(x_i) \pmod{p}$ for all $i = 1, \ldots, h$ is exactly p^{k-h}.

Shamir Secret Sharing: Security

Theorem: Let $\{(x_1, y_1), \ldots, (x_h, y_h)\}$ be a set of $h \leq k$ points in $\mathbb{Z}_p \times \mathbb{Z}_p$ with distinct x_is, where $p \geq k$. The number of polynomials g of degree $k-1$ with coefficients in $\mathbb{Z}_p$ that such that $y_i = g(x_i) \pmod{p}$ for all $i = 1, \ldots, h$ is exactly p^{k-h}.

Proof: Let $j = k - h$ (i.e., $h = k - j$).

We show by induction on $j = 0, 1, \ldots, k$ that the number of such polynomials is p^j.

Shamir Secret Sharing: Security

Theorem: Let $\{(x_1, y_1), \ldots, (x_h, y_h)\}$ be a set of $h \leq k$ points in $\mathbb{Z}_p \times \mathbb{Z}_p$ with distinct x_is, where $p \geq k$. The number of polynomials g of degree $k-1$ with coefficients in $\mathbb{Z}_p$ that such that $y_i = g(x_i) \pmod{p}$ for all $i = 1, \ldots, h$ is exactly p^{k-h}.

Proof: Let $j = k - h$ (i.e., $h = k - j$).

We show by induction on $j = 0, 1, \ldots, k$ that the number of such polynomials is p^j.

Base case: $j = 0$, i.e., $h = k$:

- There exists a unique interpolating polynomial with coefficient in $\mathbb{Z}_p$ and $p^j = p^0 = 1$.

Shamir Secret Sharing: Security

Theorem: Let $\{(x_1, y_1), \ldots, (x_h, y_h)\}$ be a set of $h \leq k$ points in $\mathbb{Z}_p \times \mathbb{Z}_p$ with distinct x_is, where $p \geq k$. The number of polynomials g of degree $k-1$ with coefficients in $\mathbb{Z}_p$ that such that $y_i = g(x_i) \pmod{p}$ for all $i = 1, \ldots, h$ is exactly p^{k-h}.

Proof: Let $j = k - h$ (i.e., $h = k - j$).

We show by induction on $j = 0, 1, \ldots, k$ that the number of such polynomials is p^j.

Base case: $j = 0$, i.e., $h = k$:

- There exists a unique interpolating polynomial with coefficient in $\mathbb{Z}_p$ and $p^j = p^0 = 1$.

Inductive step:

- Consider $j \geq 1$ and assume that the claim holds for $j - 1$.

Shamir Secret Sharing: Security

Theorem: Let $\{(x_1, y_1), \ldots, (x_h, y_h)\}$ be a set of $h \leq k$ points in $\mathbb{Z}_p \times \mathbb{Z}_p$ with distinct x_is, where $p \geq k$. The number of polynomials g of degree $k-1$ with coefficients in $\mathbb{Z}_p$ that such that $y_i = g(x_i) \pmod{p}$ for all $i = 1, \ldots, h$ is exactly p^{k-h}.

Proof: Let $j = k - h$ (i.e., $h = k - j$).

We show by induction on $j = 0, 1, \ldots, k$ that the number of such polynomials is p^j.

Base case: $j = 0$, i.e., $h = k$:

- There exists a unique interpolating polynomial with coefficient in $\mathbb{Z}_p$ and $p^j = p^0 = 1$.

Inductive step:

- Consider $j \geq 1$ and assume that the claim holds for $j - 1$.
- Since $h = k - j < k \leq p$, there must be some $x^* \in \mathbb{Z}_p$ that is different from all x_is
- Let $N(y^*)$ be the number of polynomials g of degree $k-1$ such that $g(x_i) = y_i \, \forall i$ and $g(x^*) = y^*$

Shamir Secret Sharing: Security

Theorem: Let $\{(x_1, y_1), \ldots, (x_h, y_h)\}$ be a set of $h \leq k$ points in $\mathbb{Z}_p \times \mathbb{Z}_p$ with distinct x_is, where $p \geq k$. The number of polynomials g of degree $k-1$ with coefficients in $\mathbb{Z}_p$ that such that $y_i = g(x_i) \pmod{p}$ for all $i = 1, \ldots, h$ is exactly p^{k-h}.

Proof: Let $j = k - h$ (i.e., $h = k - j$).

We show by induction on $j = 0, 1, \ldots, k$ that the number of such polynomials is p^j.

Base case: $j = 0$, i.e., $h = k$:

- There exists a unique interpolating polynomial with coefficient in $\mathbb{Z}_p$ and $p^j = p^0 = 1$.

Inductive step:

- Consider $j \geq 1$ and assume that the claim holds for $j - 1$.
- Since $h = k - j < k \leq p$, there must be some $x^* \in \mathbb{Z}_p$ that is different from all x_is
- Let $N(y^*)$ be the number of polynomials g of degree $k-1$ such that $g(x_i) = y_i \, \forall i$ and $g(x^*) = y^*$

$$\sum_{y^* \in \mathbb{Z}_p} N(y^*)$$

Shamir Secret Sharing: Security

Theorem: Let $\{(x_1, y_1), \ldots, (x_h, y_h)\}$ be a set of $h \leq k$ points in $\mathbb{Z}_p \times \mathbb{Z}_p$ with distinct x_is, where $p \geq k$. The number of polynomials g of degree $k-1$ with coefficients in $\mathbb{Z}_p$ that such that $y_i = g(x_i) \pmod{p}$ for all $i = 1, \ldots, h$ is exactly p^{k-h}.

Proof: Let $j = k - h$ (i.e., $h = k - j$).

We show by induction on $j = 0, 1, \ldots, k$ that the number of such polynomials is p^j.

Base case: $j = 0$, i.e., $h = k$:

- There exists a unique interpolating polynomial with coefficient in $\mathbb{Z}_p$ and $p^j = p^0 = 1$.

Inductive step:

- Consider $j \geq 1$ and assume that the claim holds for $j - 1$.
- Since $h = k - j < k \leq p$, there must be some $x^* \in \mathbb{Z}_p$ that is different from all x_is
- Let $N(y^*)$ be the number of polynomials g of degree $k-1$ such that $g(x_i) = y_i \; \forall i$ and $g(x^*) = y^*$

$$\sum_{y^* \in \mathbb{Z}_p} N(y^*) \quad = \sum_{y^* \in \mathbb{Z}_p} p^{j-1}$$

Shamir Secret Sharing: Security

Theorem: Let $\{(x_1, y_1), \ldots, (x_h, y_h)\}$ be a set of $h \leq k$ points in $\mathbb{Z}_p \times \mathbb{Z}_p$ with distinct x_is, where $p \geq k$. The number of polynomials g of degree $k-1$ with coefficients in $\mathbb{Z}_p$ that such that $y_i = g(x_i) \pmod{p}$ for all $i = 1, \ldots, h$ is exactly p^{k-h}.

Proof: Let $j = k - h$ (i.e., $h = k - j$).

We show by induction on $j = 0, 1, \ldots, k$ that the number of such polynomials is p^j.

Base case: $j = 0$, i.e., $h = k$:

- There exists a unique interpolating polynomial with coefficient in $\mathbb{Z}_p$ and $p^j = p^0 = 1$.

Inductive step:

- Consider $j \geq 1$ and assume that the claim holds for $j - 1$.
- Since $h = k - j < k \leq p$, there must be some $x^* \in \mathbb{Z}_p$ that is different from all x_is
- Let $N(y^*)$ be the number of polynomials g of degree $k-1$ such that $g(x_i) = y_i \, \forall i$ and $g(x^*) = y^*$

$$\sum_{y^* \in \mathbb{Z}_p} N(y^*) = \sum_{y^* \in \mathbb{Z}_p} p^{j-1} = |\mathbb{Z}_p| \cdot p^{j-1}$$

Shamir Secret Sharing: Security

Theorem: Let $\{(x_1, y_1), \ldots, (x_h, y_h)\}$ be a set of $h \leq k$ points in $\mathbb{Z}_p \times \mathbb{Z}_p$ with distinct x_is, where $p \geq k$. The number of polynomials g of degree $k-1$ with coefficients in $\mathbb{Z}_p$ that such that $y_i = g(x_i) \pmod p$ for all $i = 1, \ldots, h$ is exactly p^{k-h}.

Proof: Let $j = k - h$ (i.e., $h = k - j$).

We show by induction on $j = 0, 1, \ldots, k$ that the number of such polynomials is p^j.

Base case: $j = 0$, i.e., $h = k$:

- There exists a unique interpolating polynomial with coefficient in $\mathbb{Z}_p$ and $p^j = p^0 = 1$.

Inductive step:

- Consider $j \geq 1$ and assume that the claim holds for $j - 1$.
- Since $h = k - j < k \leq p$, there must be some $x^* \in \mathbb{Z}_p$ that is different from all x_is
- Let $N(y^*)$ be the number of polynomials g of degree $k-1$ such that $g(x_i) = y_i \; \forall i$ and $g(x^*) = y^*$

$$\sum_{y^* \in \mathbb{Z}_p} N(y^*) = \sum_{y^* \in \mathbb{Z}_p} p^{j-1} = |\mathbb{Z}_p| \cdot p^{j-1} = p^j$$

$\square$

Two-Party Computation

Alice and Bob want to jointly compute a function $f(x_1, x_2, \ldots, x_m, y_1, y_2, \ldots, y_n)$

- Alice knows the inputs $x_1, \ldots, x_m$
- Bob knows the inputs $y_1, \ldots, y_n$

Two-Party Computation

Alice and Bob want to jointly compute a function $f(x_1, x_2, \ldots, x_m, y_1, y_2, \ldots, y_n)$

- Alice knows the inputs $x_1, \ldots, x_m$
- Bob knows the inputs $y_1, \ldots, y_n$

Each party wants the other party to learn nothing about their inputs*

*Except from what they can deduce from knowing the value $f(x_1, \ldots, x_m, y_1, \ldots, y_n)$ (this is inevitable)

Two-Party Computation

Alice and Bob want to jointly compute a function $f(x_1, x_2, \ldots, x_m, y_1, y_2, \ldots, y_n)$

- Alice knows the inputs $x_1, \ldots, x_m$
- Bob knows the inputs $y_1, \ldots, y_n$

Each party wants the other party to learn nothing about their inputs*

*Except from what they can deduce from knowing the value $f(x_1, \ldots, x_m, y_1, \ldots, y_n)$ (this is inevitable)

Example: In the "movie selection" scenario, Alice and Bob wanted to compute $f(x_1, y_1) = x_1 \wedge y_1$

Two-Party Computation

Alice and Bob want to jointly compute a function $f(x_1, x_2, \ldots, x_m, y_1, y_2, \ldots, y_n)$

- Alice knows the inputs $x_1, \ldots, x_m$
- Bob knows the inputs $y_1, \ldots, y_n$

Each party wants the other party to learn nothing about their inputs*

*Except from what they can deduce from knowing the value $f(x_1, \ldots, x_m, y_1, \ldots, y_n)$ (this is inevitable)

Example: In the "movie selection" scenario, Alice and Bob wanted to compute $f(x_1, y_1) = x_1 \wedge y_1$

We actually consider a stronger variant: Alice wants to learn $f(x_1, x_2, \ldots, x_m, y_1, y_2, \ldots, y_n)$ while Bob learns nothing

- If we can solve this variant, then we can solve the above case (Alice sends the final output Bob)
- This allows us to solve the more general case in which Alice learns $f_A(x_1, x_2, \ldots, x_m, y_1, y_2, \ldots, y_n)$ and Bob learns $f_B(x_1, x_2, \ldots, x_m, y_1, y_2, \ldots, y_n)$

Two-Party Computation: The Honest but Curious Model

We will design a **Two-Party computation protocol** that solves this problem for functions f that can be computed in polynomial-time in the **honest but curious model**.

Honest but curious model: Alice and Bob obey the protocol, but they try to gather as much information as they can (each of them wants to break the privacy of the other party)

Two-Party Computation: The Honest but Curious Model

We will design a **Two-Party computation protocol** that solves this problem for functions f that can be computed in polynomial-time in the **honest but curious model**.

Honest but curious model: Alice and Bob obey the protocol, but they try to gather as much information as they can (each of them wants to break the privacy of the other party)

This is the analogous of a passive eavesdropper in classic cryptography

Two-Party Computation: The Honest but Curious Model

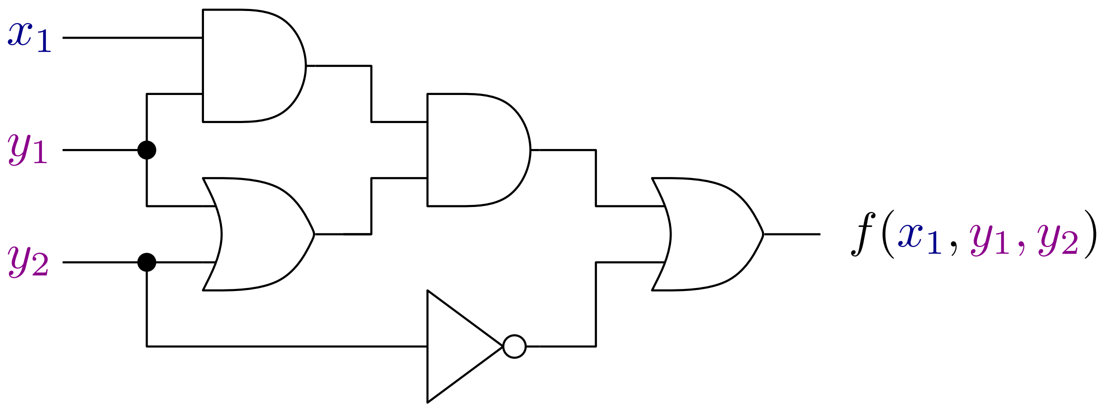
We will design a **Two-Party computation protocol** that solves this problem for functions f that can be computed in polynomial-time in the **honest but curious model**.

Honest but curious model: Alice and Bob obey the protocol, but they try to gather as much information as they can (each of them wants to break the privacy of the other party)

This is the analogous of a passive eavesdropper in classic cryptography

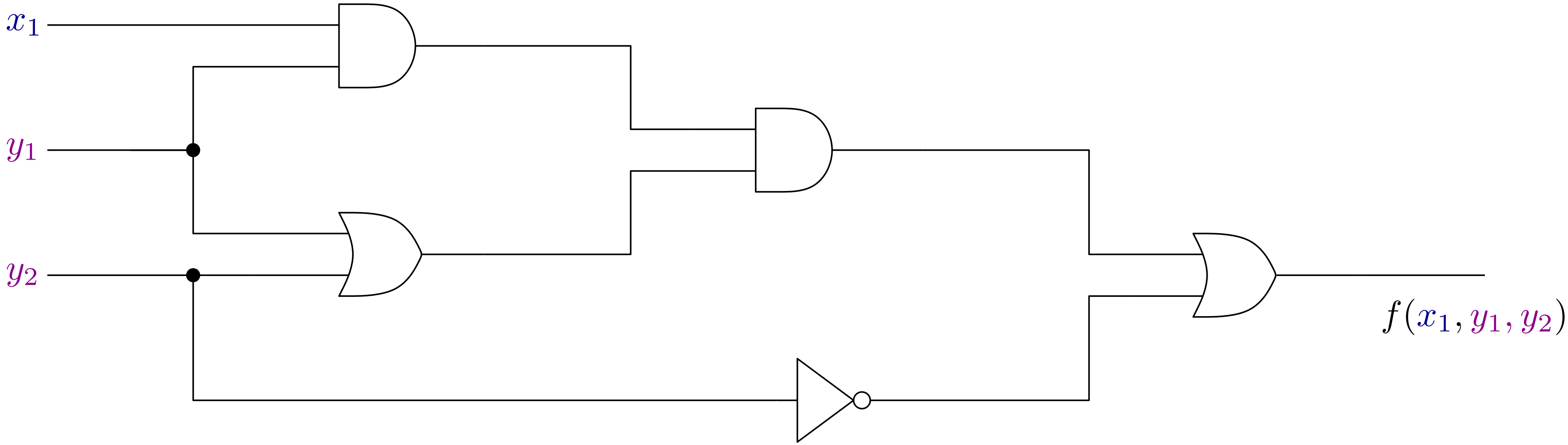
The protocol will be based on evaluating a (polynomial-size) **Boolean circuit** that computes f

For simplicity, think of Boolean circuits with a single output (the protocol extends to multiple outputs in a straightforward way)



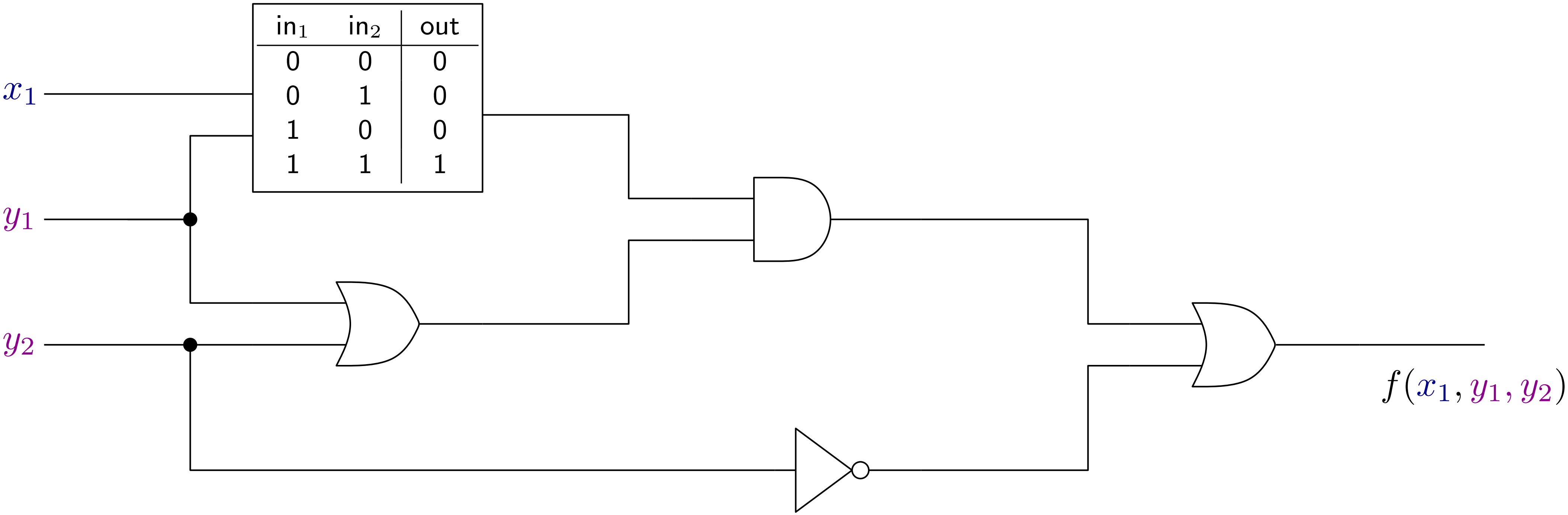
Yao's Garbled Circuits: Building the Circuit

Alice replaces each logic gate with an explicit description of its truth table



Yao's Garbled Circuits: Building the Circuit

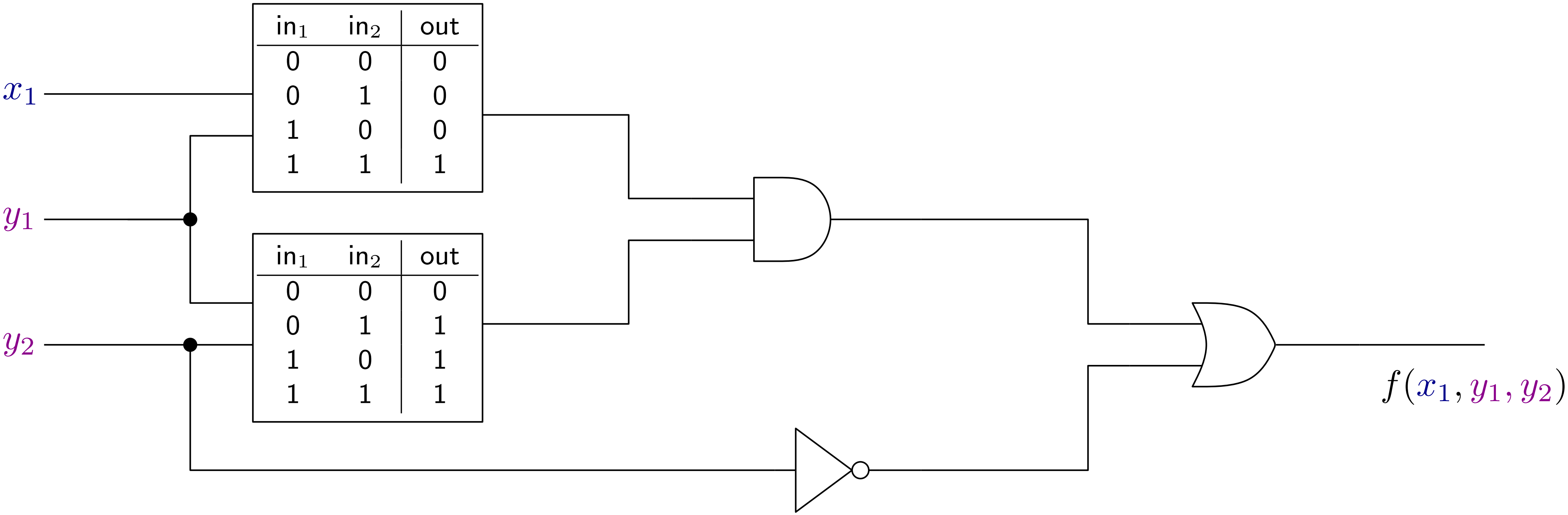
Alice replaces each logic gate with an explicit description of its truth table



in_1	in_2	out
0	0	0
0	1	0
1	0	0
1	1	1

Yao's Garbled Circuits: Building the Circuit

Alice replaces each logic gate with an explicit description of its truth table

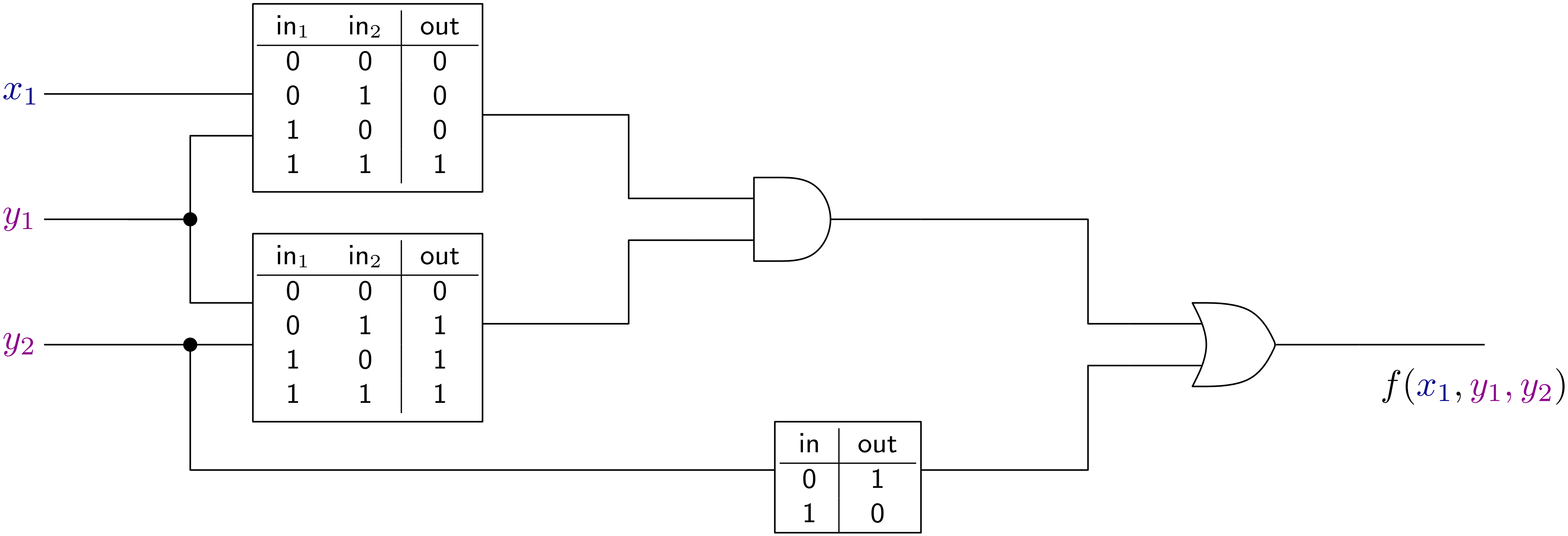


in_1	in_2	out
0	0	0
0	1	0
1	0	0
1	1	1

in_1	in_2	out
0	0	0
0	1	1
1	0	1
1	1	1

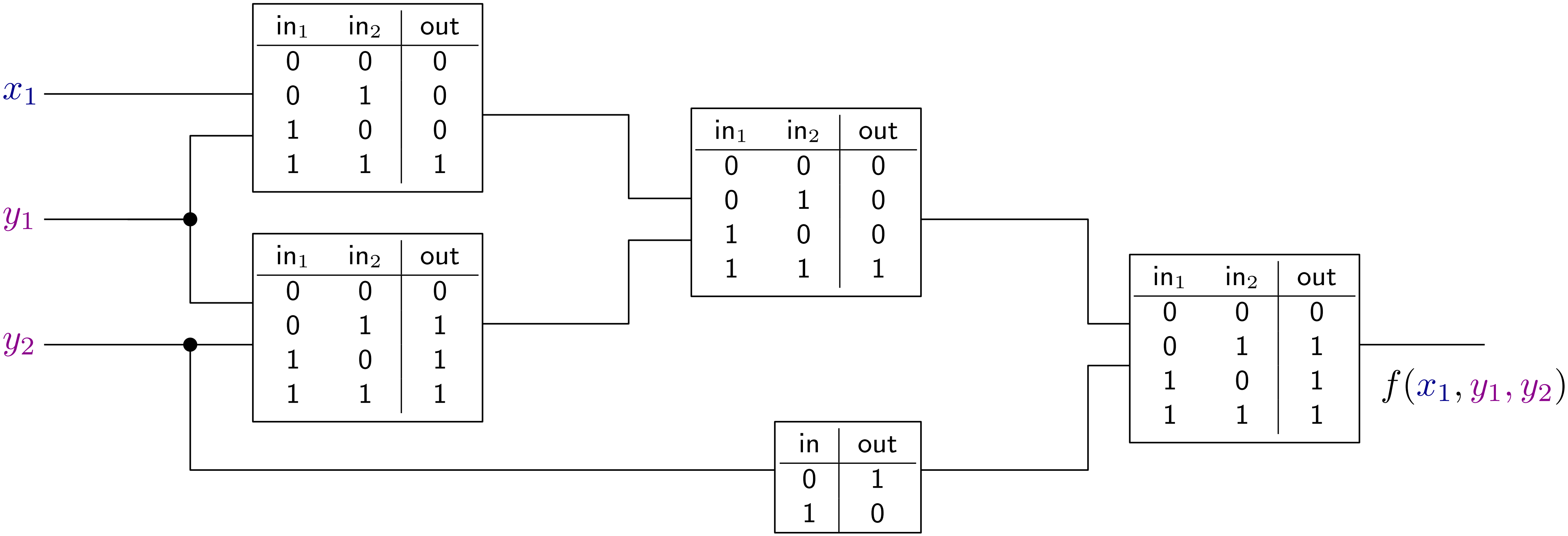
Yao's Garbled Circuits: Building the Circuit

Alice replaces each logic gate with an explicit description of its truth table



Yao's Garbled Circuits: Building the Circuit

Alice replaces each logic gate with an explicit description of its truth table



in$_1$	in$_2$	out
0	0	0
0	1	0
1	0	0
1	1	1

in$_1$	in$_2$	out
0	0	0
0	1	1
1	0	1
1	1	1

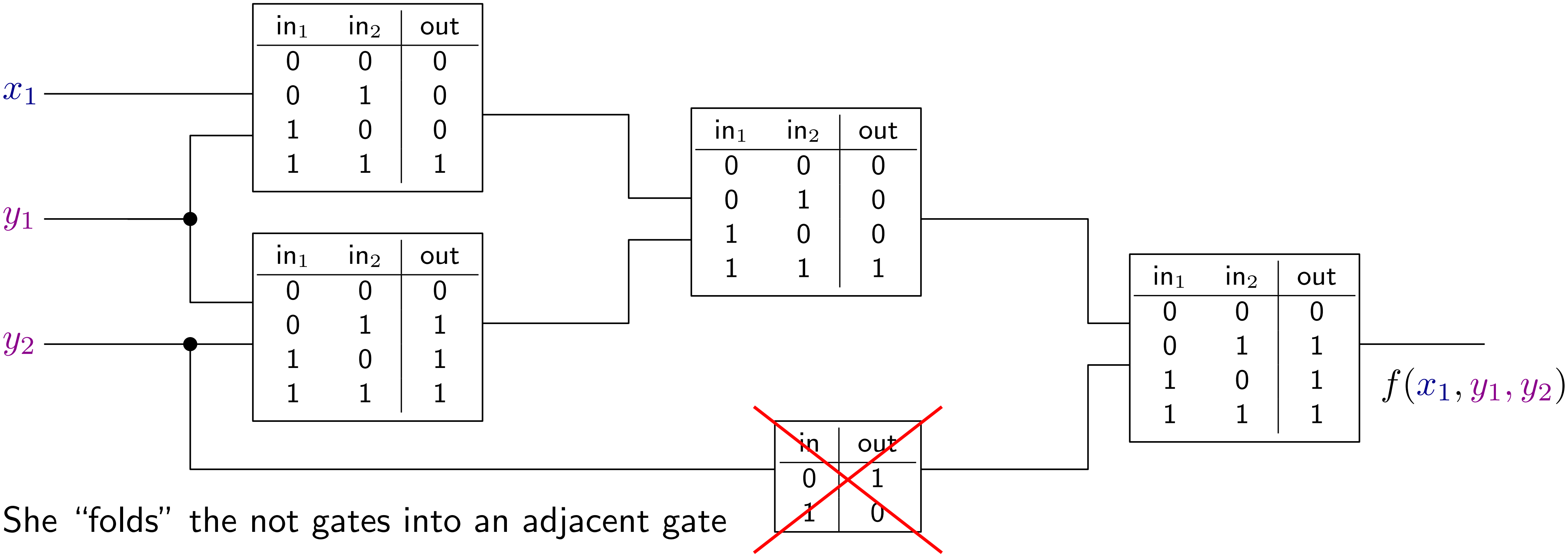
in$_1$	in$_2$	out
0	0	0
0	1	0
1	0	0
1	1	1

in	out
0	1
1	0

in$_1$	in$_2$	out
0	0	0
0	1	1
1	0	1
1	1	1

Yao's Garbled Circuits: Building the Circuit

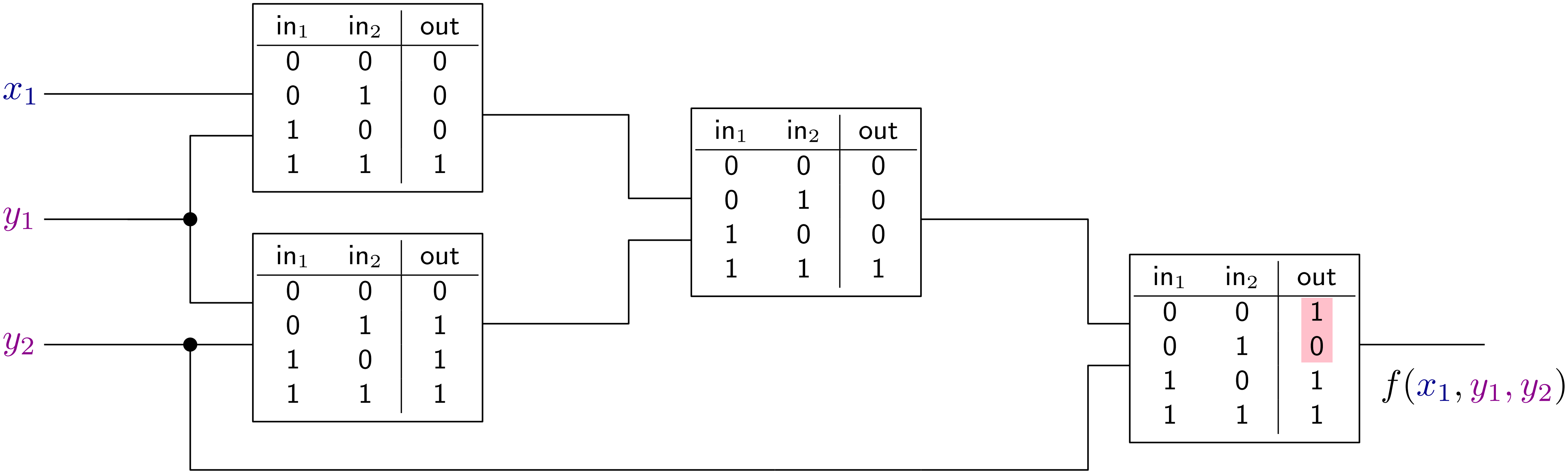
Alice replaces each logic gate with an explicit description of its truth table



She "folds" the not gates into an adjacent gate

Yao's Garbled Circuits: Building the Circuit

Alice replaces each logic gate with an explicit description of its truth table



in_1	in_2	out
0	0	0
0	1	0
1	0	0
1	1	1

in_1	in_2	out
0	0	0
0	1	1
1	0	1
1	1	1

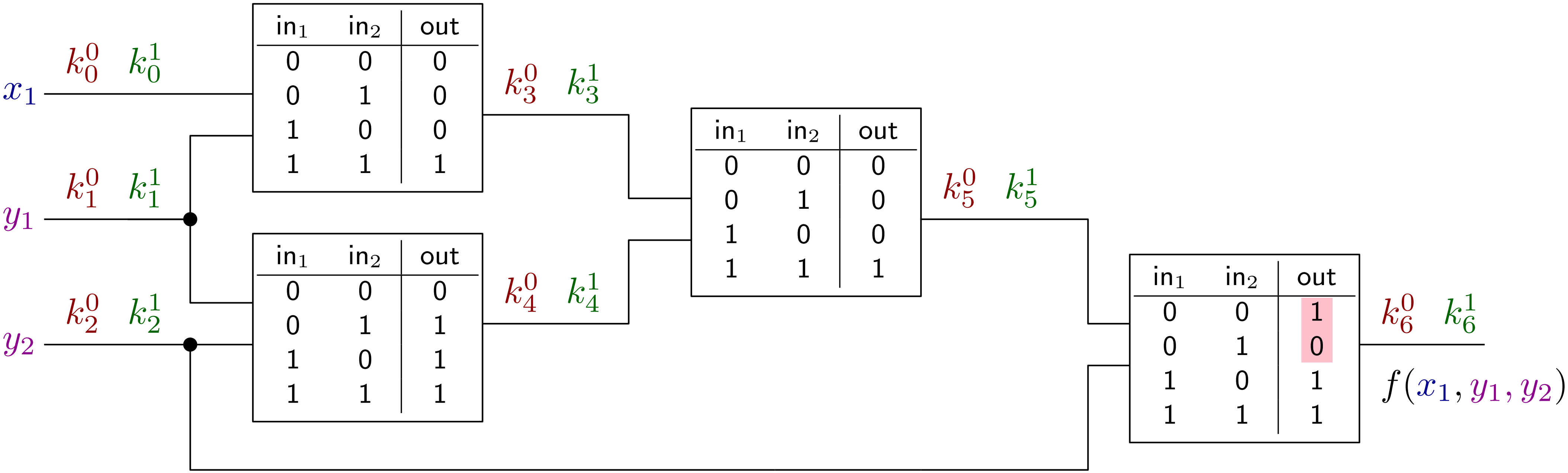
in_1	in_2	out
0	0	0
0	1	0
1	0	0
1	1	1

in_1	in_2	out
0	0	1
0	1	0
1	0	1
1	1	1

She "folds" the not gates into an adjacent gate

Yao's Garbled Circuits: Building the Circuit

Alice replaces each logic gate with an explicit description of its truth table



in$_1$	in$_2$	out
0	0	0
0	1	0
1	0	0
1	1	1

in$_1$	in$_2$	out
0	0	0
0	1	1
1	0	1
1	1	1

in$_1$	in$_2$	out
0	0	0
0	1	0
1	0	0
1	1	1

in$_1$	in$_2$	out
0	0	1
0	1	0
1	0	1
1	1	1

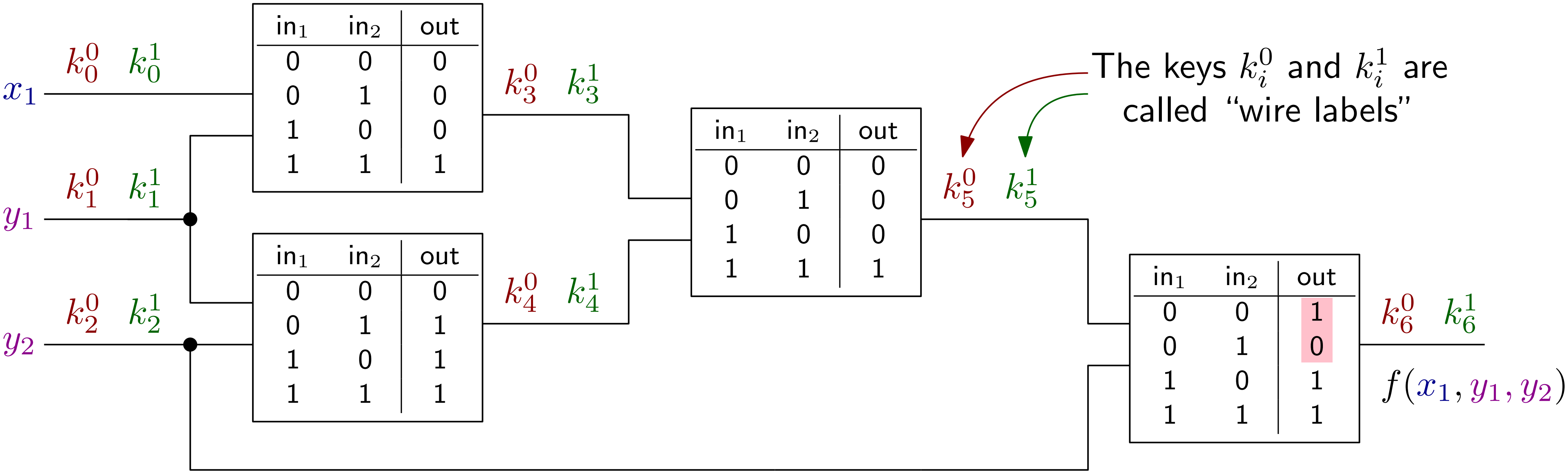
She "folds" the not gates into an adjacent gate

Each wire carries either a logic 0 or a logic 1

She assigns two random symmetric keys k_i^0 and k_i^1 to the generic i-th wire to represent 0 and 1, resp.

Yao's Garbled Circuits: Building the Circuit

Alice replaces each logic gate with an explicit description of its truth table



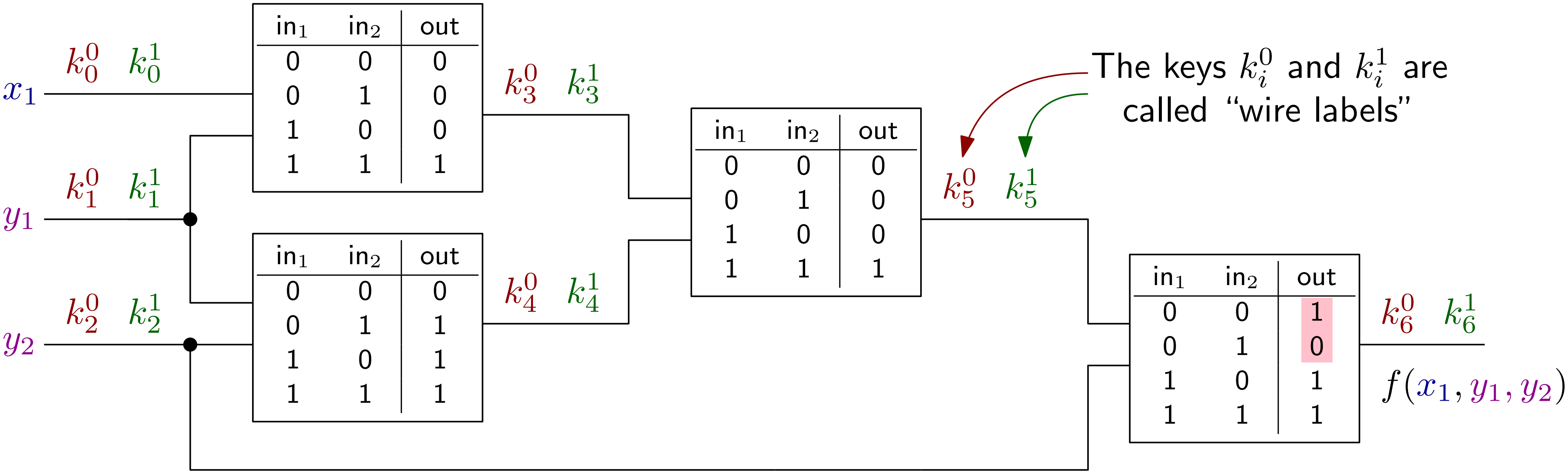
She "folds" the not gates into an adjacent gate

Each wire carries either a logic 0 or a logic 1

She assigns two random symmetric keys k_i^0 and k_i^1 to the generic i-th wire to represent 0 and 1, resp.

Yao's Garbled Circuits: Building the Circuit

Alice replaces each logic gate with an explicit description of its truth table



She "folds" the not gates into an adjacent gate

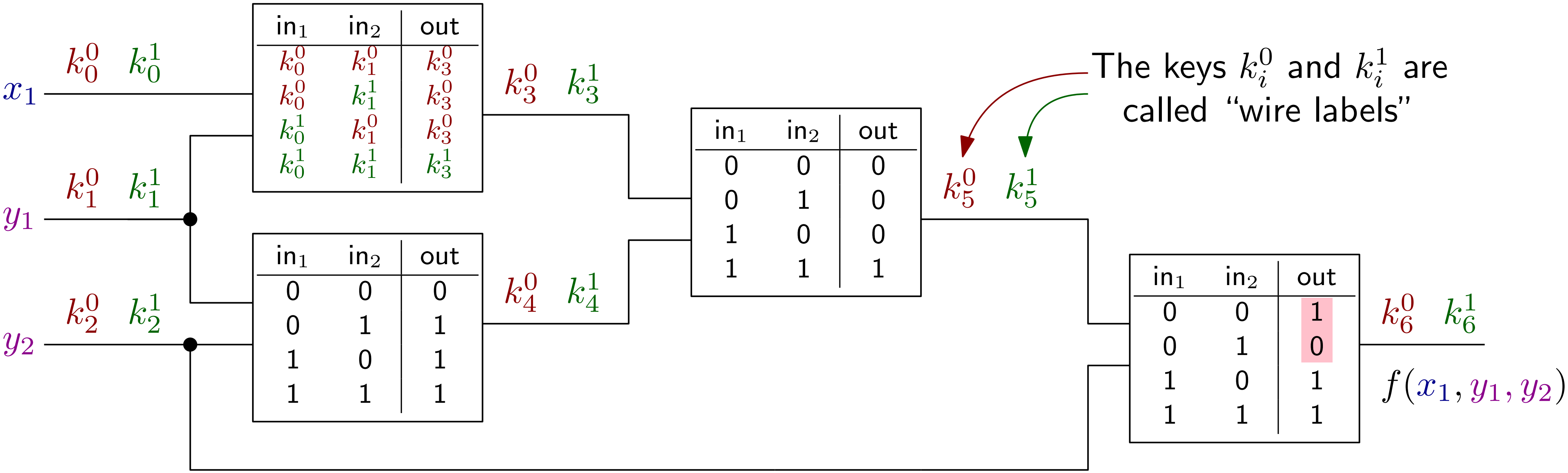
Each wire carries either a logic 0 or a logic 1

She assigns two random symmetric keys k_i^0 and k_i^1 to the generic i-th wire to represent 0 and 1, resp.

She rewrites the truth tables in terms of the input and output wire labels

Yao's Garbled Circuits: Building the Circuit

Alice replaces each logic gate with an explicit description of its truth table



She "folds" the not gates into an adjacent gate

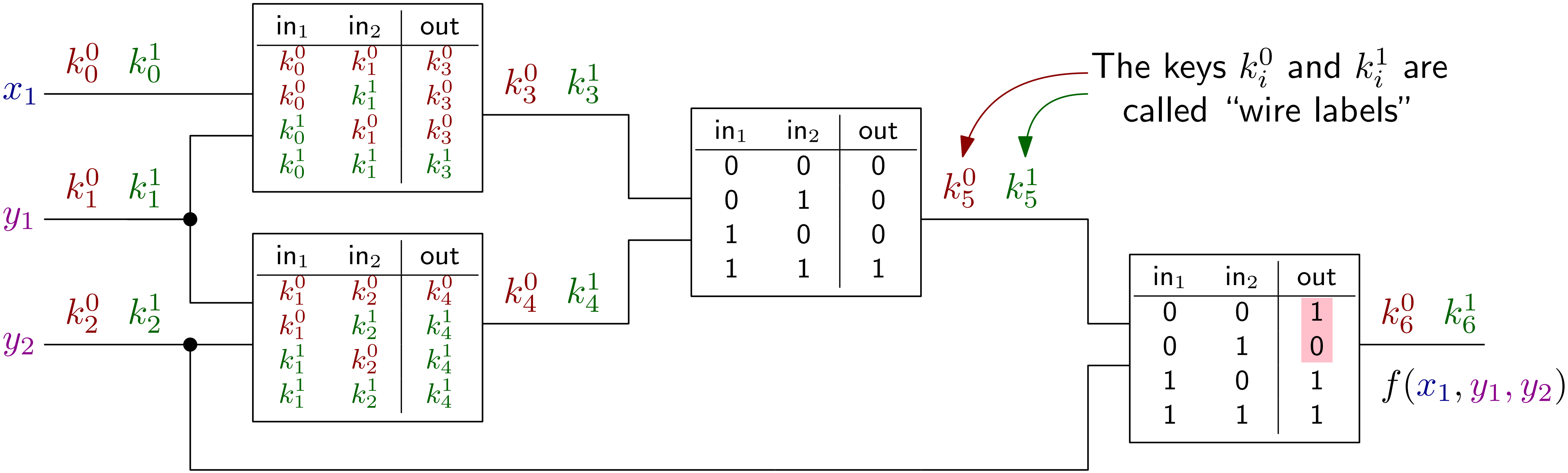
Each wire carries either a logic 0 or a logic 1

She assigns two random symmetric keys k_i^0 and k_i^1 to the generic i-th wire to represent 0 and 1, resp.

She rewrites the truth tables in terms of the input and output wire labels

Yao's Garbled Circuits: Building the Circuit

Alice replaces each logic gate with an explicit description of its truth table



She "folds" the not gates into an adjacent gate

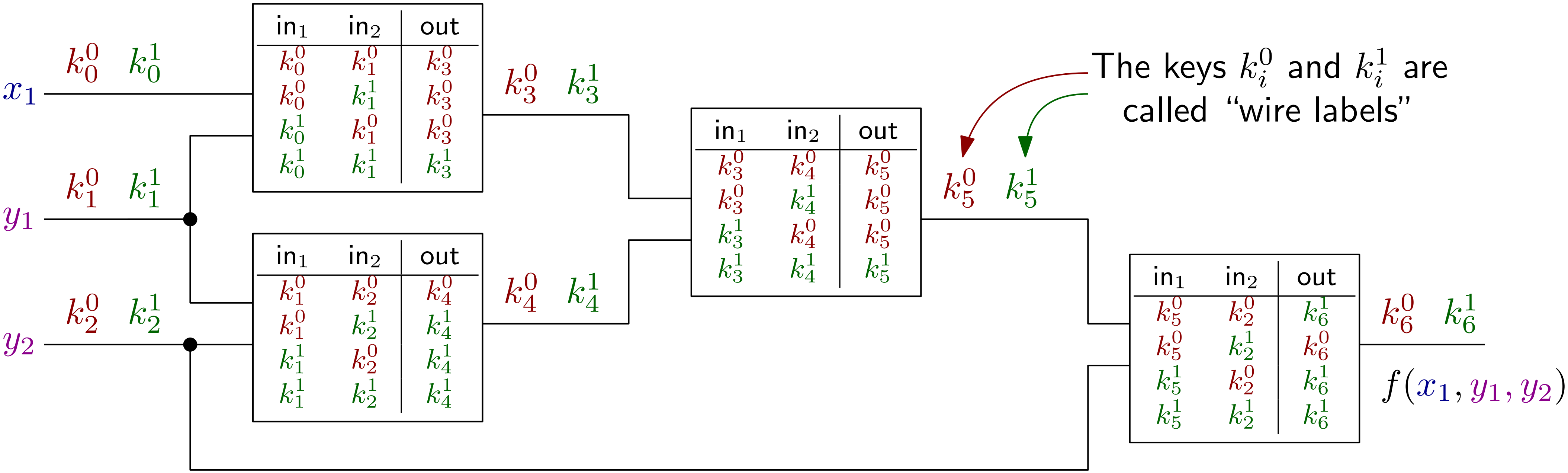
Each wire carries either a logic 0 or a logic 1

She assigns two random symmetric keys k_i^0 and k_i^1 to the generic i-th wire to represent 0 and 1, resp.

She rewrites the truth tables in terms of the input and output wire labels

Yao's Garbled Circuits: Building the Circuit

Alice replaces each logic gate with an explicit description of its truth table



She "folds" the not gates into an adjacent gate

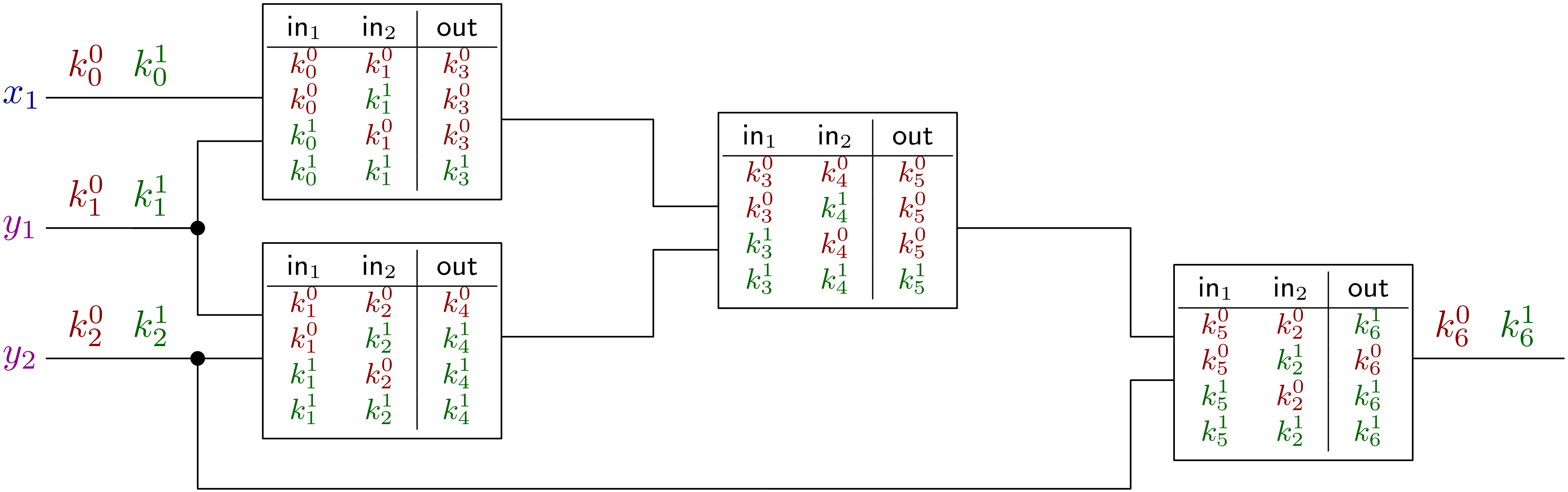
Each wire carries either a logic 0 or a logic 1

She assigns two random symmetric keys k_i^0 and k_i^1 to the generic i-th wire to represent 0 and 1, resp.

She rewrites the truth tables in terms of the input and output wire labels

Yao's Garbled Circuits: Building the Circuit

Alice now encrypts the outputs in each truth table using a **secure authenticated encryption scheme**



in_1	in_2	out
k_0^0	k_1^0	k_3^0
k_0^0	k_1^1	k_3^0
k_0^1	k_1^0	k_3^0
k_0^1	k_1^1	k_3^1

in_1	in_2	out
k_1^0	k_2^0	k_4^0
k_1^0	k_2^1	k_4^1
k_1^1	k_2^0	k_4^1
k_1^1	k_2^1	k_4^1

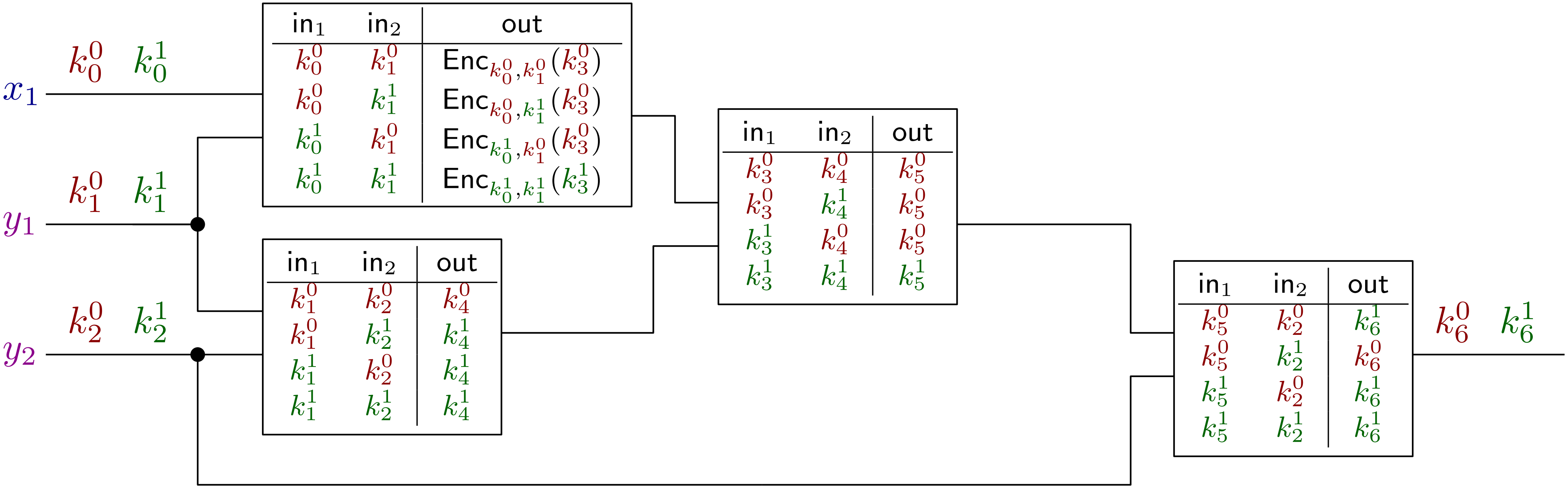
in_1	in_2	out
k_3^0	k_4^0	k_5^0
k_3^0	k_4^1	k_5^0
k_3^1	k_4^0	k_5^0
k_3^1	k_4^1	k_5^1

in_1	in_2	out
k_5^0	k_2^0	k_6^1
k_5^0	k_2^1	k_6^0
k_5^1	k_2^0	k_6^1
k_5^1	k_2^1	k_6^1

The key used to encrypt an output consists of the two corresponding input wire labels

Yao's Garbled Circuits: Building the Circuit

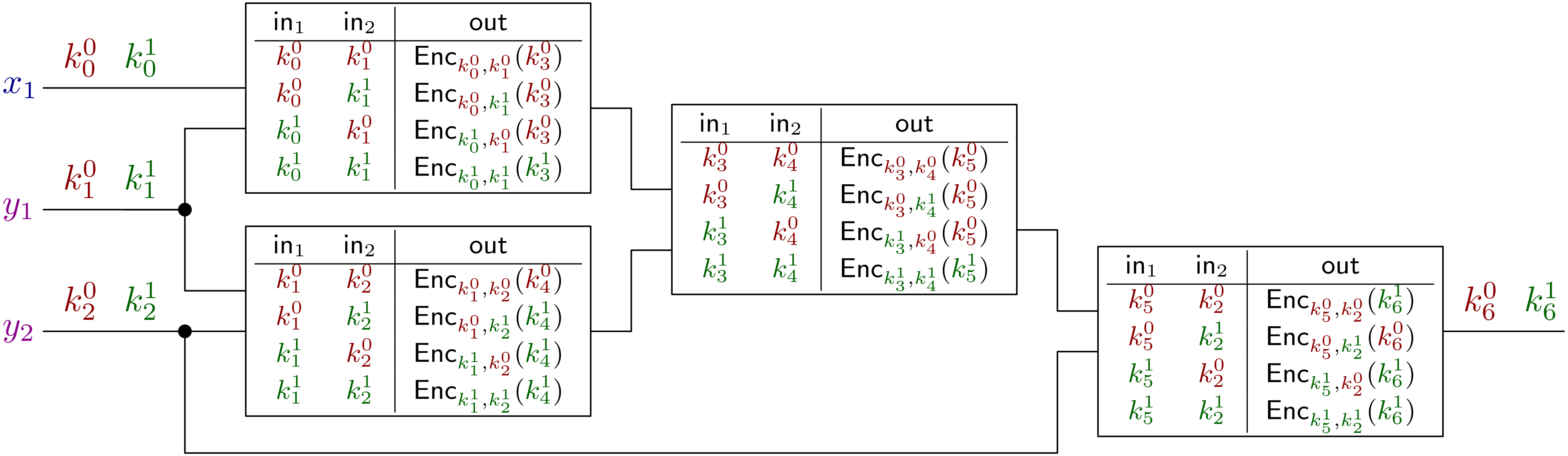
Alice now encrypts the outputs in each truth table using a **secure authenticated encryption scheme**



The key used to encrypt an output consists of the two corresponding input wire labels

Yao's Garbled Circuits: Building the Circuit

Alice now encrypts the outputs in each truth table using a **secure authenticated encryption scheme**



in_1	in_2	out
k_0^0	k_1^0	$\mathsf{Enc}_{k_0^0,k_1^0}(k_3^0)$
k_0^0	k_1^1	$\mathsf{Enc}_{k_0^0,k_1^1}(k_3^0)$
k_0^1	k_1^0	$\mathsf{Enc}_{k_0^1,k_1^0}(k_3^0)$
k_0^1	k_1^1	$\mathsf{Enc}_{k_0^1,k_1^1}(k_3^1)$

in_1	in_2	out
k_1^0	k_2^0	$\mathsf{Enc}_{k_1^0,k_2^0}(k_4^0)$
k_1^0	k_2^1	$\mathsf{Enc}_{k_1^0,k_2^1}(k_4^1)$
k_1^1	k_2^0	$\mathsf{Enc}_{k_1^1,k_2^0}(k_4^1)$
k_1^1	k_2^1	$\mathsf{Enc}_{k_1^1,k_2^1}(k_4^1)$

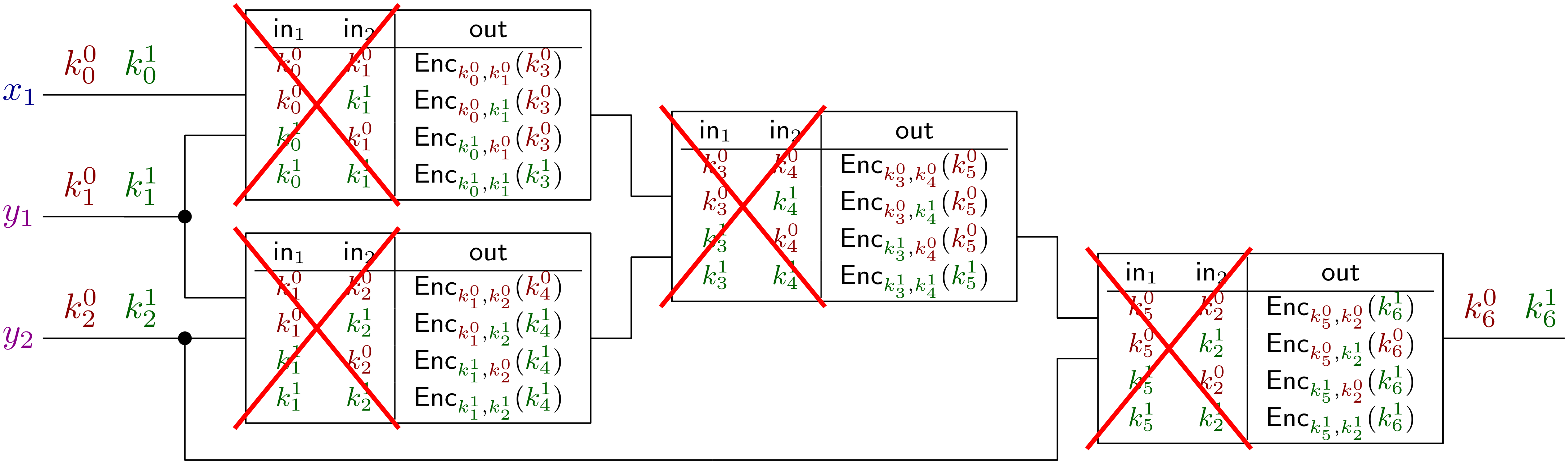
in_1	in_2	out
k_3^0	k_4^0	$\mathsf{Enc}_{k_3^0,k_4^0}(k_5^0)$
k_3^0	k_4^1	$\mathsf{Enc}_{k_3^0,k_4^1}(k_5^0)$
k_3^1	k_4^0	$\mathsf{Enc}_{k_3^1,k_4^0}(k_5^0)$
k_3^1	k_4^1	$\mathsf{Enc}_{k_3^1,k_4^1}(k_5^1)$

in_1	in_2	out
k_5^0	k_2^0	$\mathsf{Enc}_{k_5^0,k_2^0}(k_6^1)$
k_5^0	k_2^1	$\mathsf{Enc}_{k_5^0,k_2^1}(k_6^0)$
k_5^1	k_2^0	$\mathsf{Enc}_{k_5^1,k_2^0}(k_6^1)$
k_5^1	k_2^1	$\mathsf{Enc}_{k_5^1,k_2^1}(k_6^1)$

The key used to encrypt an output consists of the two corresponding input wire labels

Yao's Garbled Circuits: Building the Circuit

Alice now encrypts the outputs in each truth table using a **secure authenticated encryption scheme**

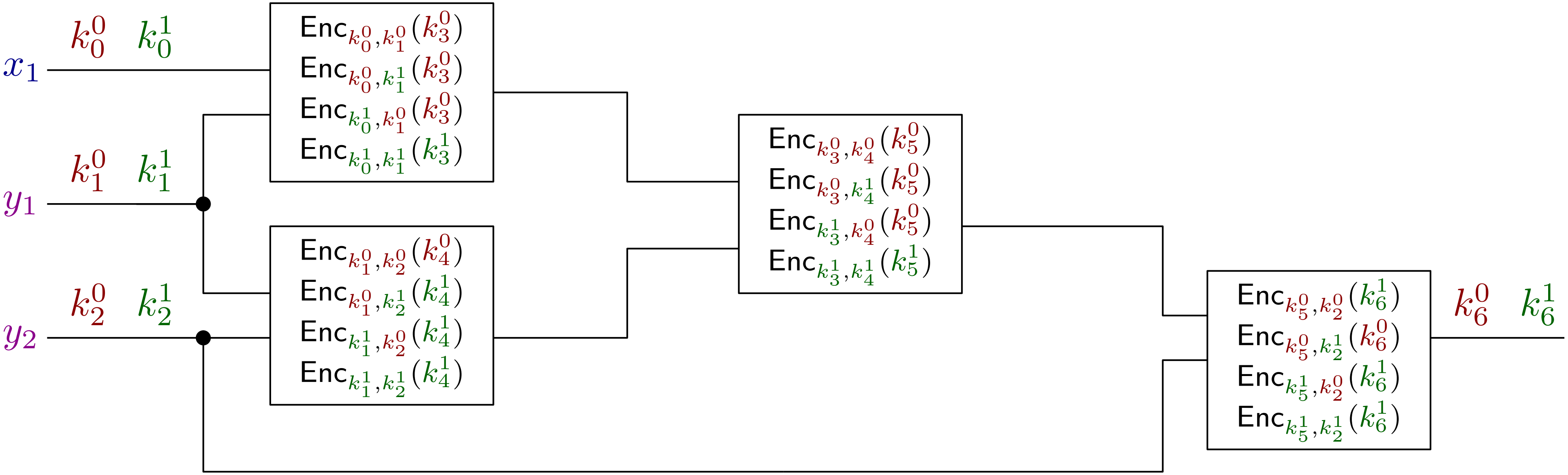


The key used to encrypt an output consists of the two corresponding input wire labels

She now drops the inputs from the truth tables...

Yao's Garbled Circuits: Building the Circuit

Alice now encrypts the outputs in each truth table using a **secure authenticated encryption scheme**

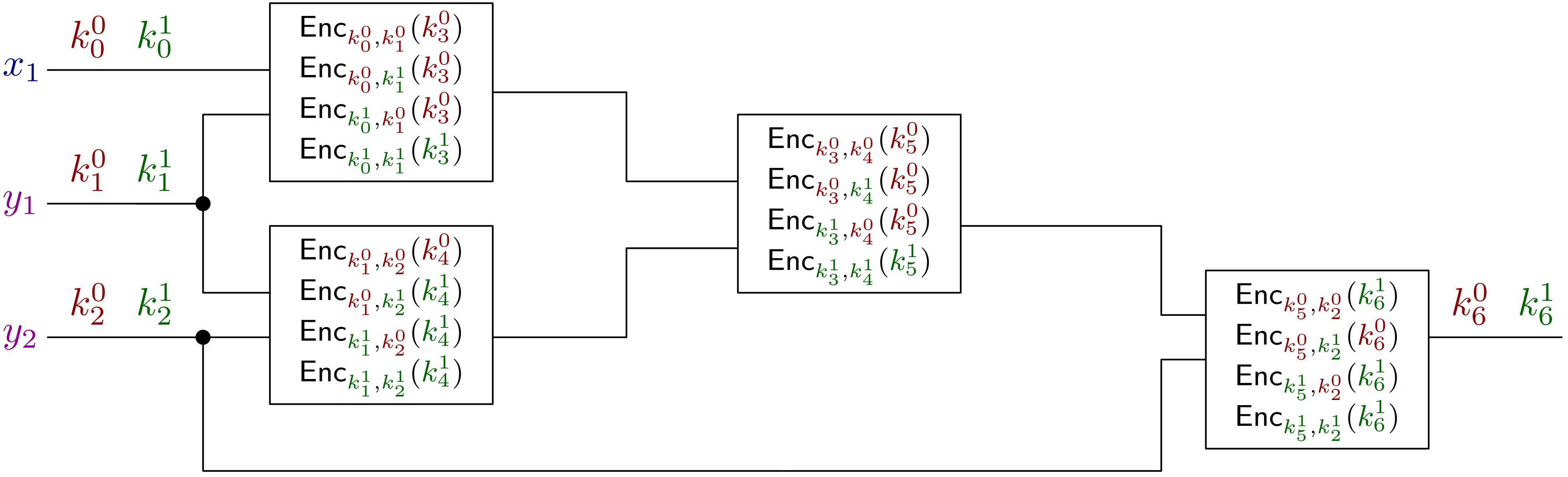


The key used to encrypt an output consists of the two corresponding input wire labels

She now drops the inputs from the truth tables. . .

Yao's Garbled Circuits: Building the Circuit

Alice now encrypts the outputs in each truth table using a **secure authenticated encryption scheme**

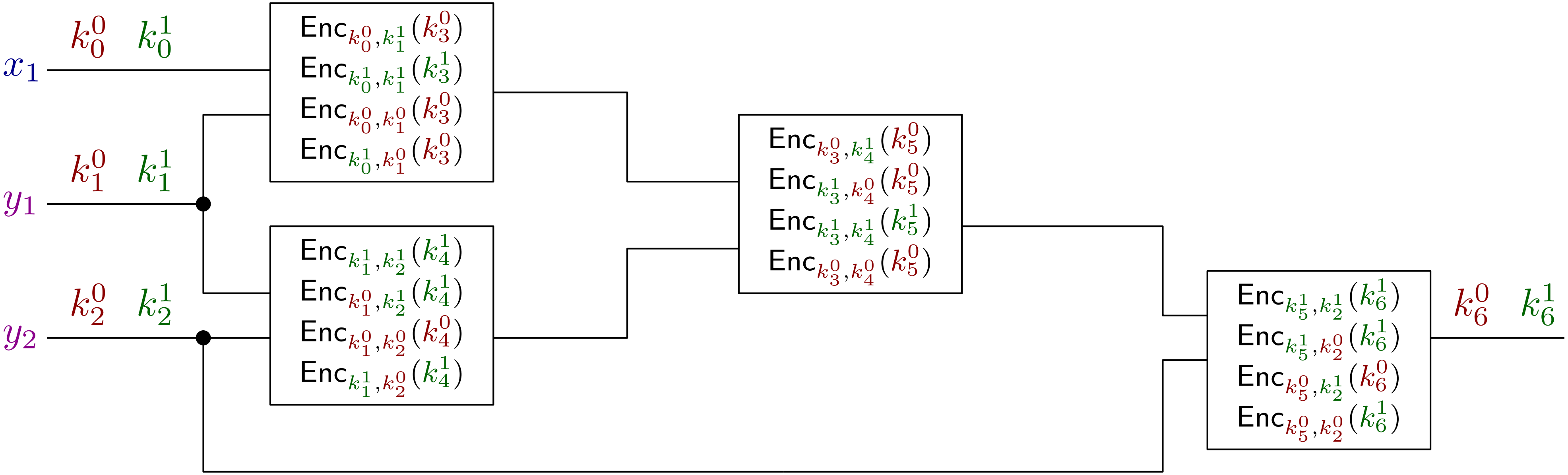


The key used to encrypt an output consists of the two corresponding input wire labels

She now drops the inputs from the truth tables. . . and randomly permutes the outputs of each table

Yao's Garbled Circuits: Building the Circuit

Alice now encrypts the outputs in each truth table using a **secure authenticated encryption scheme**

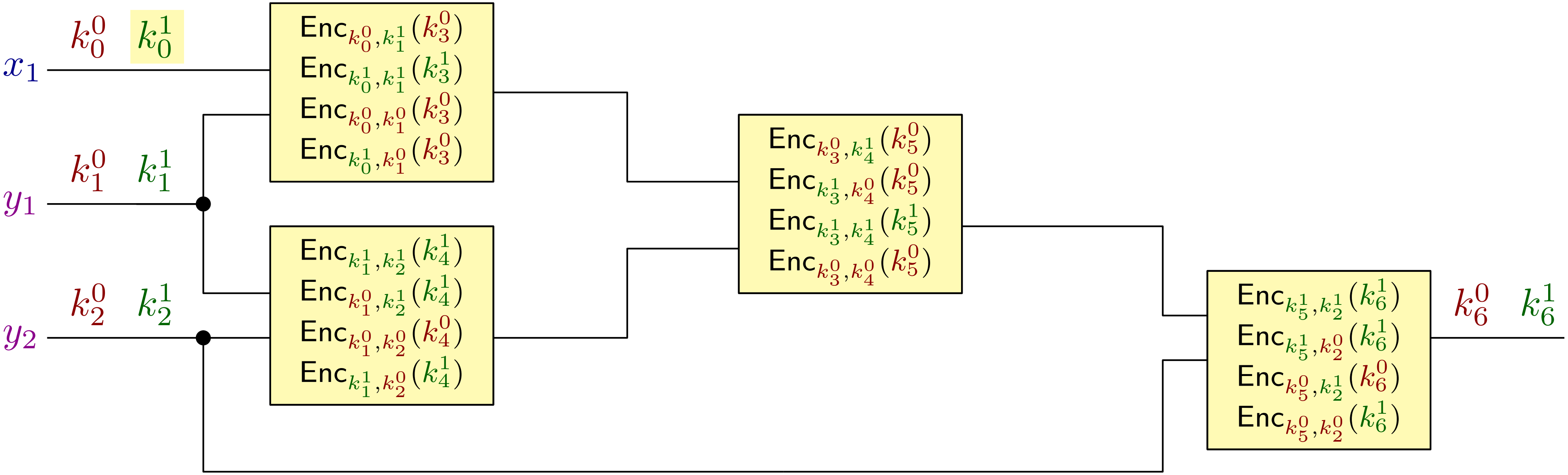


The key used to encrypt an output consists of the two corresponding input wire labels

She now drops the inputs from the truth tables. . . and randomly permutes the outputs of each table

Yao's Garbled Circuits: Building the Circuit

Alice now encrypts the outputs in each truth table using a **secure authenticated encryption scheme**



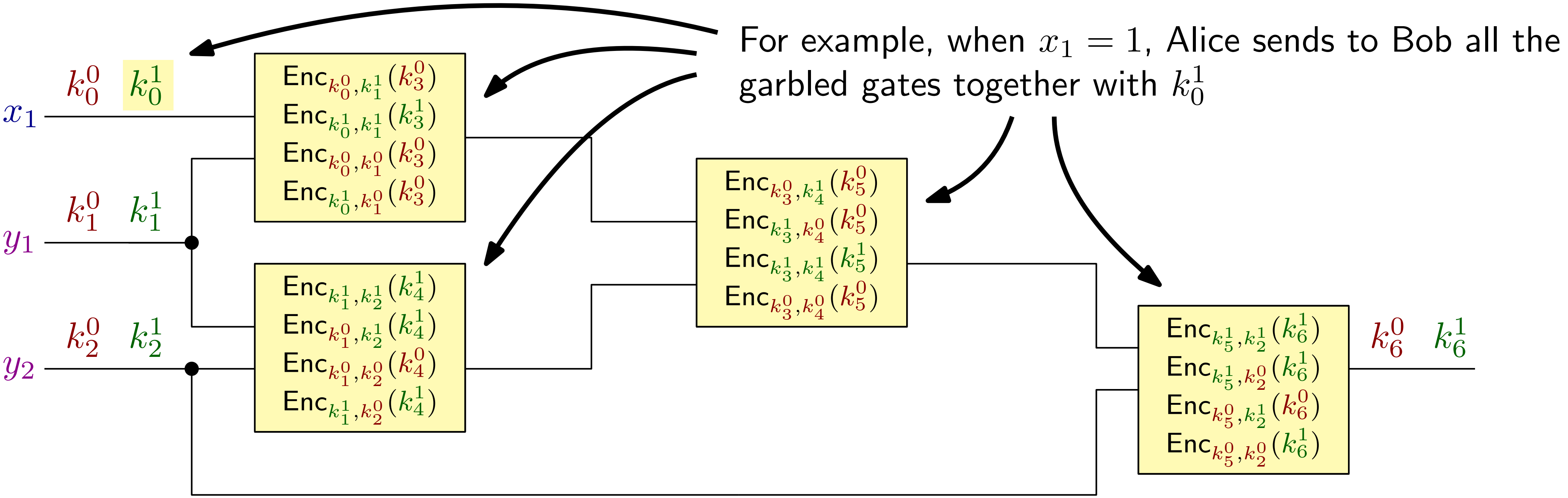
The key used to encrypt an output consists of the two corresponding input wire labels

She now drops the inputs from the truth tables… and randomly permutes the outputs of each table

Finally, Alice sends to Bob all the (now garbled) logic gates, the connections between them, and the wire-labels corresponding to her inputs

Yao's Garbled Circuits: Building the Circuit

Alice now encrypts the outputs in each truth table using a **secure authenticated encryption scheme**



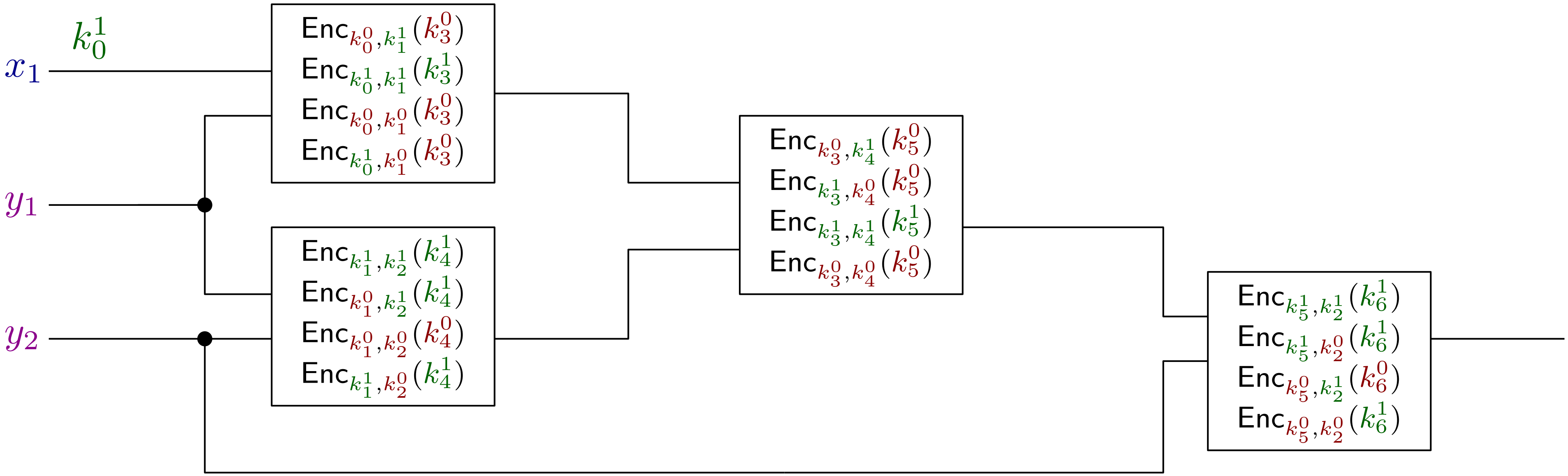
The key used to encrypt an output consists of the two corresponding input wire labels

She now drops the inputs from the truth tables... and randomly permutes the outputs of each table

Finally, Alice sends to Bob all the (now garbled) logic gates, the connections between them, and the wire-labels corresponding to her inputs

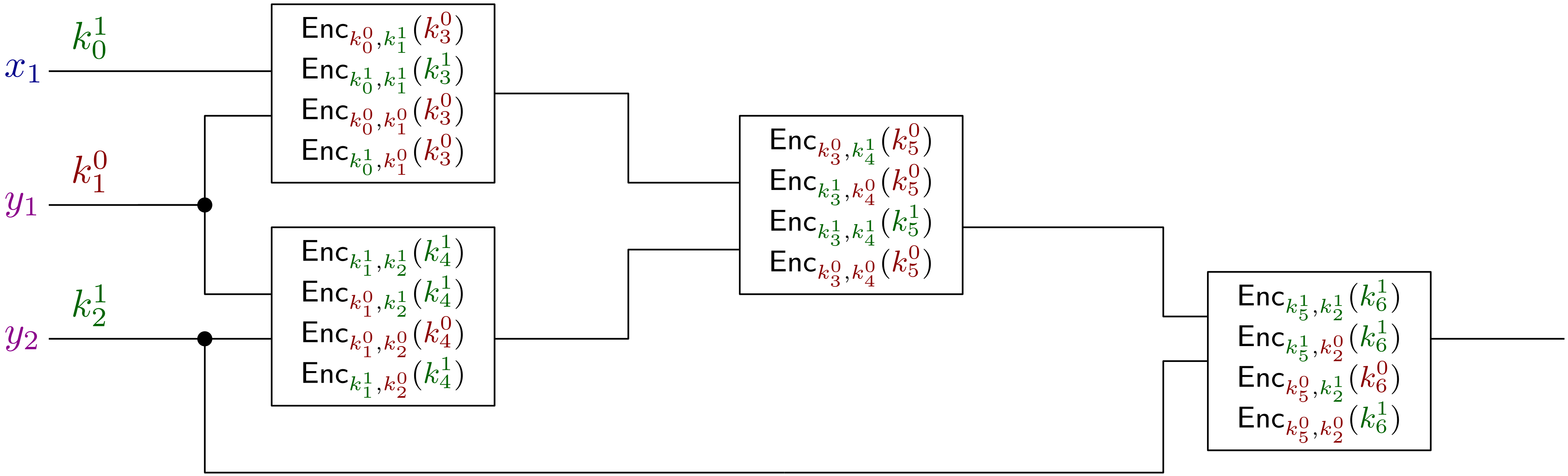
Yao's Garbled Circuits: Evaluating the Circuit

Bob receives the garbled circuit from Alice, together with the wire-labels of Alice's inputs



Yao's Garbled Circuits: Evaluating the Circuit

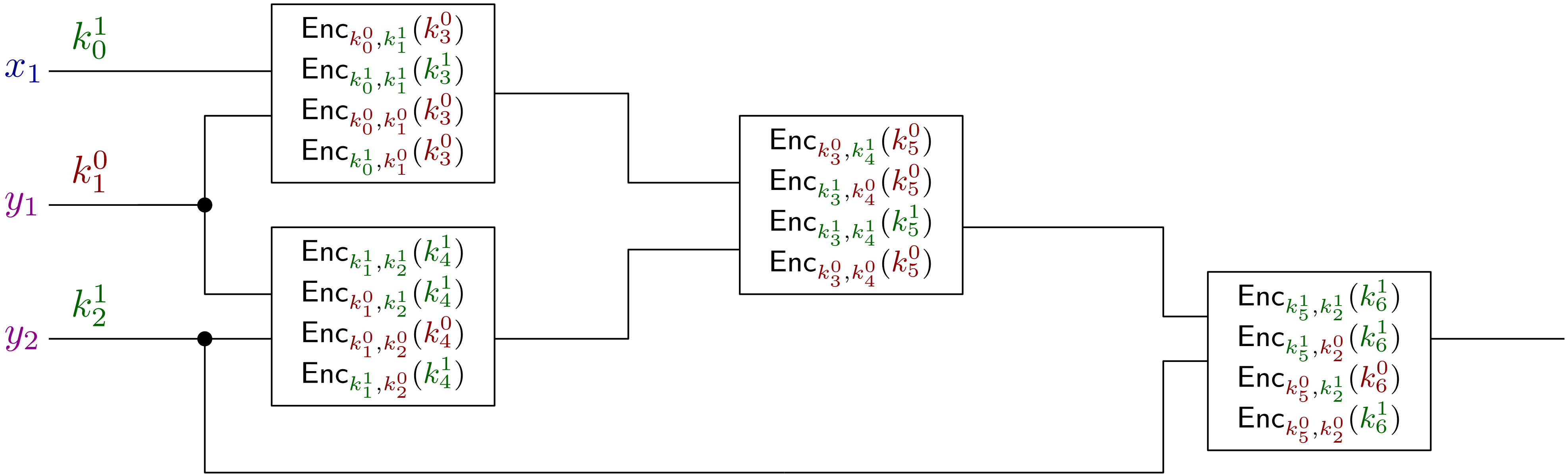
Bob receives the garbled circuit from Alice, together with the wire-labels of Alice's inputs



Suppose that Bob somehow knows the wire-labels corresponding to his input (we will handle this later)

Yao's Garbled Circuits: Evaluating the Circuit

Bob receives the garbled circuit from Alice, together with the wire-labels of Alice's inputs

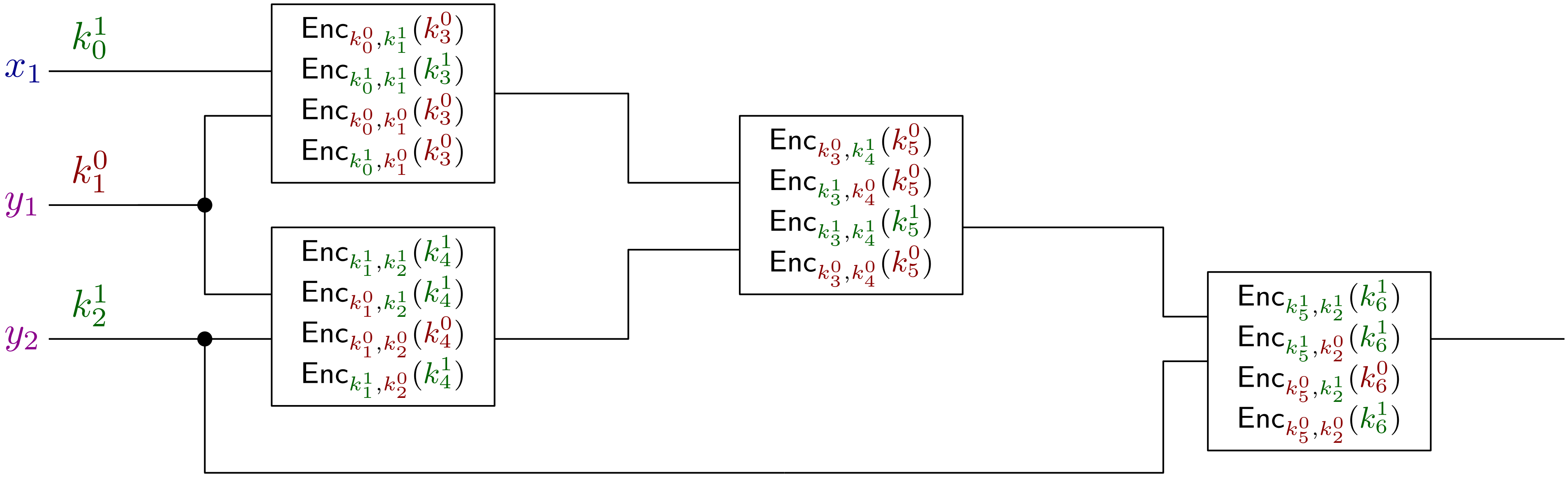


Suppose that Bob somehow knows the wire-labels corresponding to his input (we will handle this later)

Bob can evaluate the garbled circuit and recover the wire-label corresponding to the output wire

Yao's Garbled Circuits: Evaluating the Circuit

Bob receives the garbled circuit from Alice, together with the wire-labels of Alice's inputs



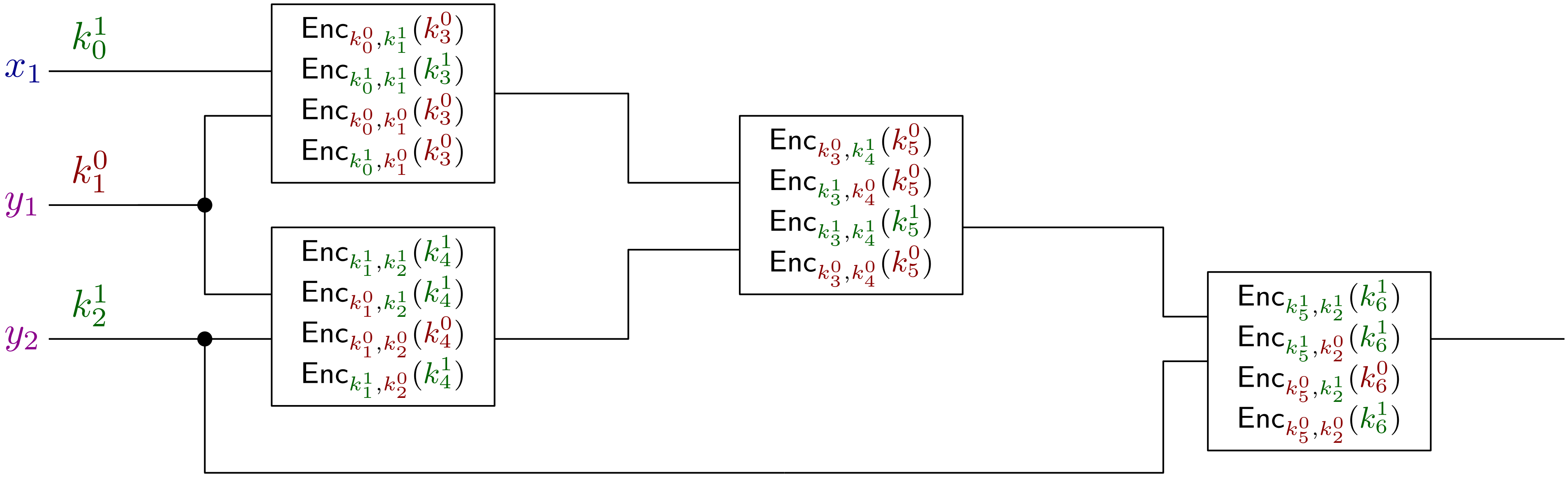
Suppose that Bob somehow knows the wire-labels corresponding to his input (we will handle this later)

Bob can evaluate the garbled circuit and recover the wire-label corresponding to the output wire

- To evaluate a logic gate, bob tries to Decrypt each of the four possible encrypted rows

Yao's Garbled Circuits: Evaluating the Circuit

Bob receives the garbled circuit from Alice, together with the wire-labels of Alice's inputs



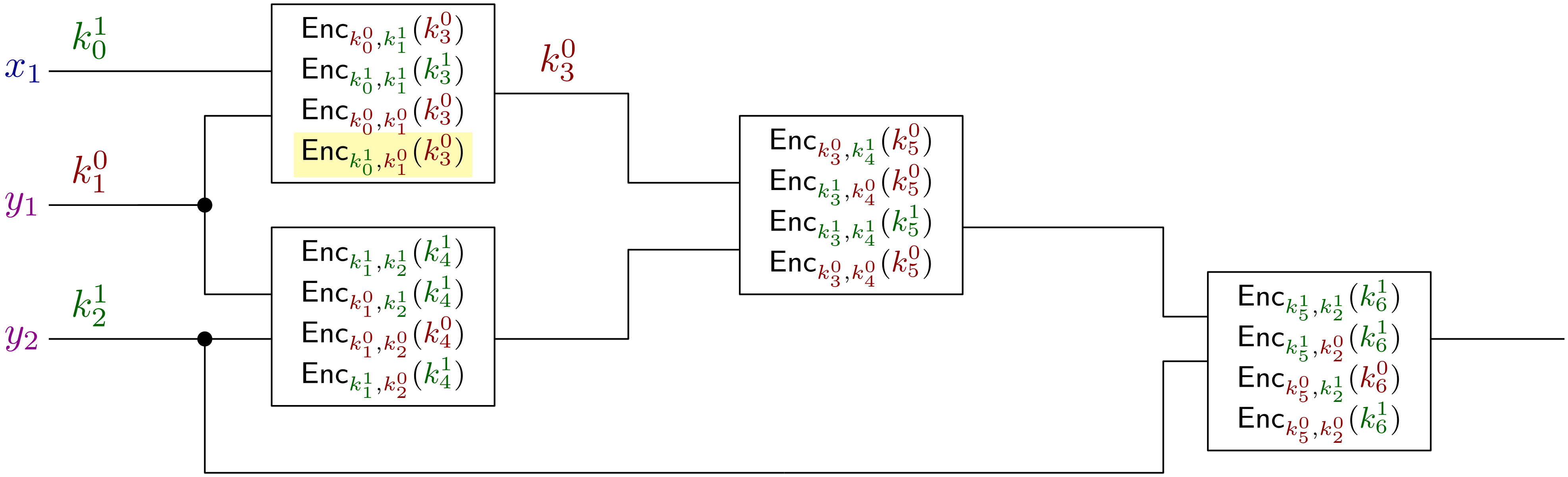
Suppose that Bob somehow knows the wire-labels corresponding to his input (we will handle this later)

Bob can evaluate the garbled circuit and recover the wire-label corresponding to the output wire

- To evaluate a logic gate, bob tries to Decrypt each of the four possible encrypted rows
- Since we are using a secure authenticated encryption scheme, exactly one of these rows will decrypt successfully (except for negligible probability)

Yao's Garbled Circuits: Evaluating the Circuit

Bob receives the garbled circuit from Alice, together with the wire-labels of Alice's inputs



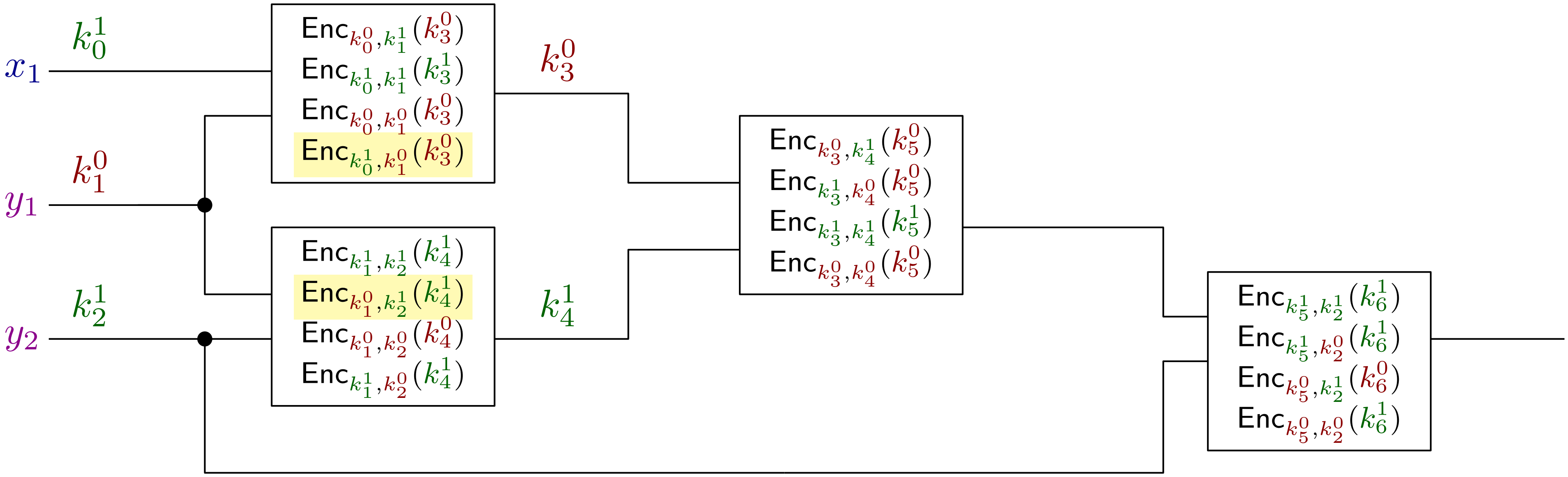
Suppose that Bob somehow knows the wire-labels corresponding to his input (we will handle this later)

Bob can evaluate the garbled circuit and recover the wire-label corresponding to the output wire

- To evaluate a logic gate, bob tries to Decrypt each of the four possible encrypted rows
- Since we are using a secure authenticated encryption scheme, exactly one of these rows will decrypt successfully (except for negligible probability)

Yao's Garbled Circuits: Evaluating the Circuit

Bob receives the garbled circuit from Alice, together with the wire-labels of Alice's inputs



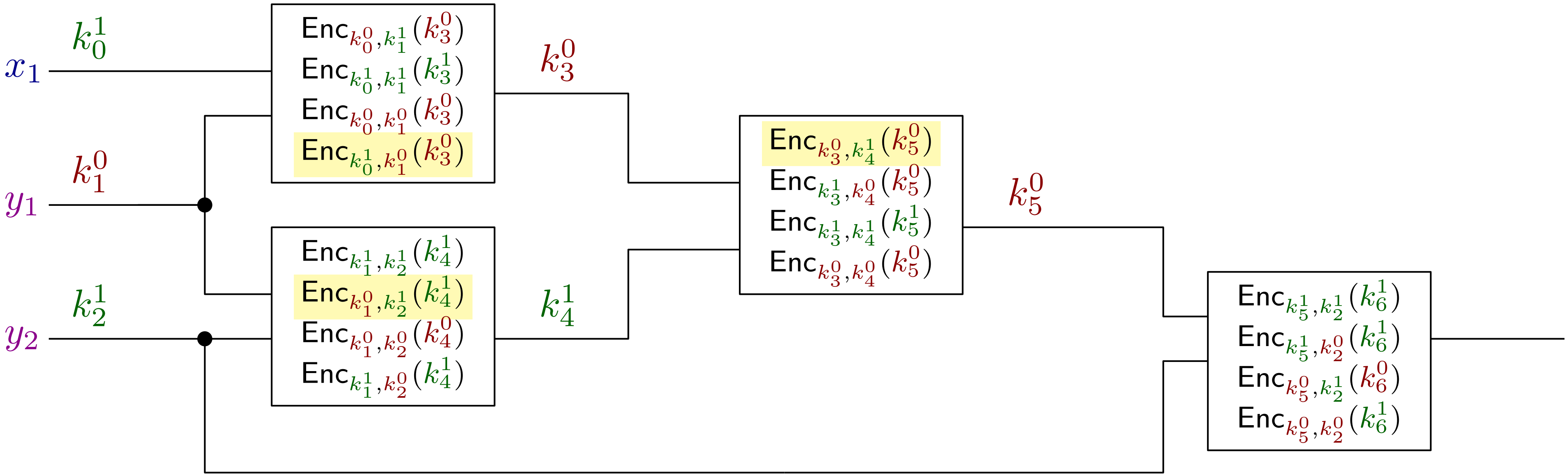
Suppose that Bob somehow knows the wire-labels corresponding to his input (we will handle this later)

Bob can evaluate the garbled circuit and recover the wire-label corresponding to the output wire

- To evaluate a logic gate, bob tries to Decrypt each of the four possible encrypted rows
- Since we are using a secure authenticated encryption scheme, exactly one of these rows will decrypt successfully (except for negligible probability)

Yao's Garbled Circuits: Evaluating the Circuit

Bob receives the garbled circuit from Alice, together with the wire-labels of Alice's inputs



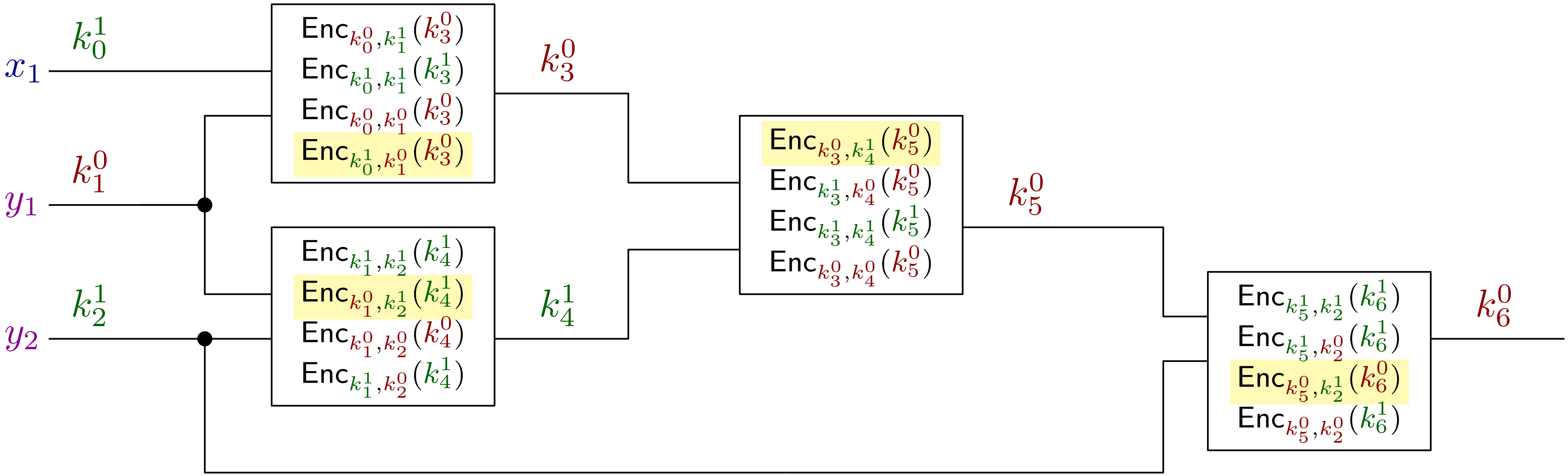
Suppose that Bob somehow knows the wire-labels corresponding to his input (we will handle this later)

Bob can evaluate the garbled circuit and recover the wire-label corresponding to the output wire

- To evaluate a logic gate, bob tries to Decrypt each of the four possible encrypted rows
- Since we are using a secure authenticated encryption scheme, exactly one of these rows will decrypt successfully (except for negligible probability)

Yao's Garbled Circuits: Evaluating the Circuit

Bob receives the garbled circuit from Alice, together with the wire-labels of Alice's inputs



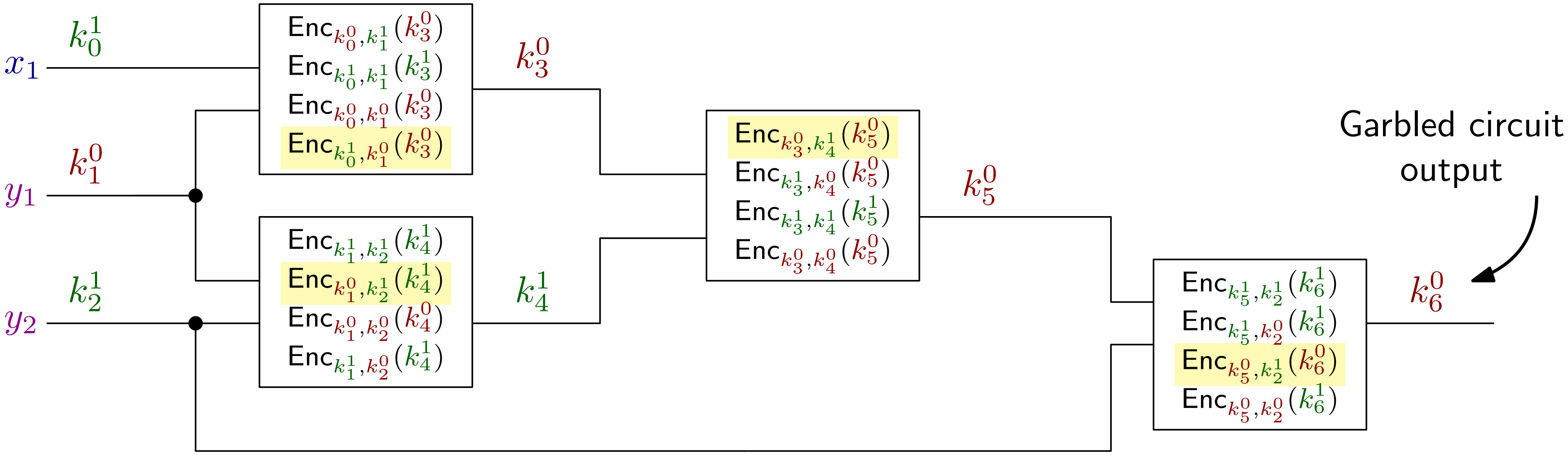
Suppose that Bob somehow knows the wire-labels corresponding to his input (we will handle this later)

Bob can evaluate the garbled circuit and recover the wire-label corresponding to the output wire

- To evaluate a logic gate, bob tries to Decrypt each of the four possible encrypted rows
- Since we are using a secure authenticated encryption scheme, exactly one of these rows will decrypt successfully (except for negligible probability)

Yao's Garbled Circuits: Evaluating the Circuit

Bob receives the garbled circuit from Alice, together with the wire-labels of Alice's inputs



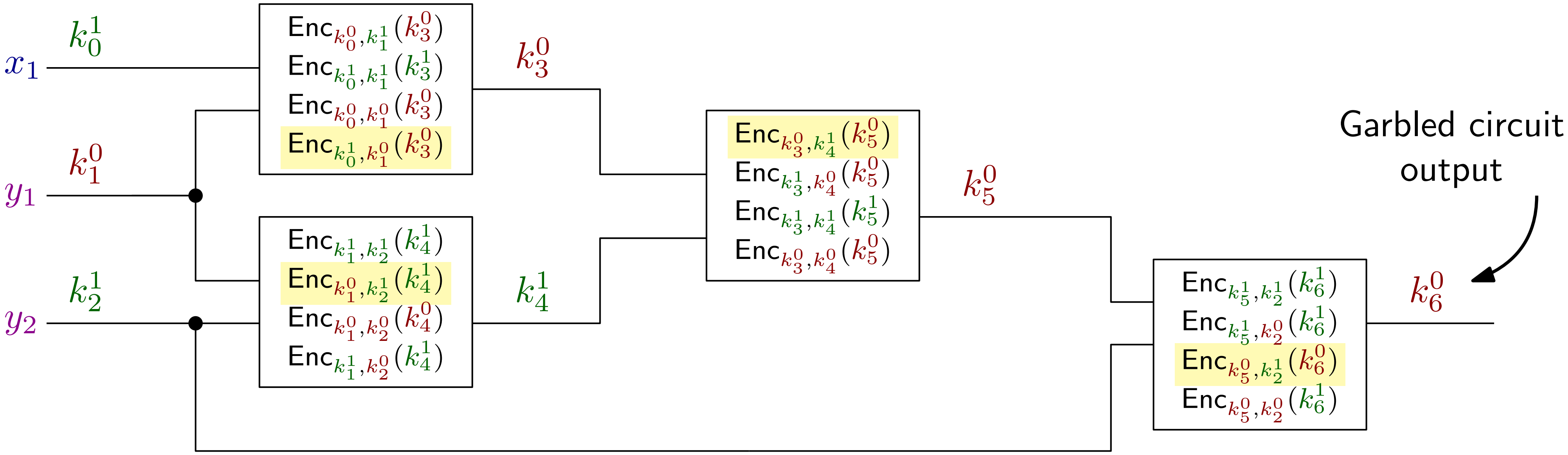
Suppose that Bob somehow knows the wire-labels corresponding to his input (we will handle this later)

Bob can evaluate the garbled circuit and recover the wire-label corresponding to the output wire

- To evaluate a logic gate, bob tries to Decrypt each of the four possible encrypted rows
- Since we are using a secure authenticated encryption scheme, exactly one of these rows will decrypt successfully (except for negligible probability)

Yao's Garbled Circuits: Evaluating the Circuit

Bob receives the garbled circuit from Alice, together with the wire-labels of Alice's inputs



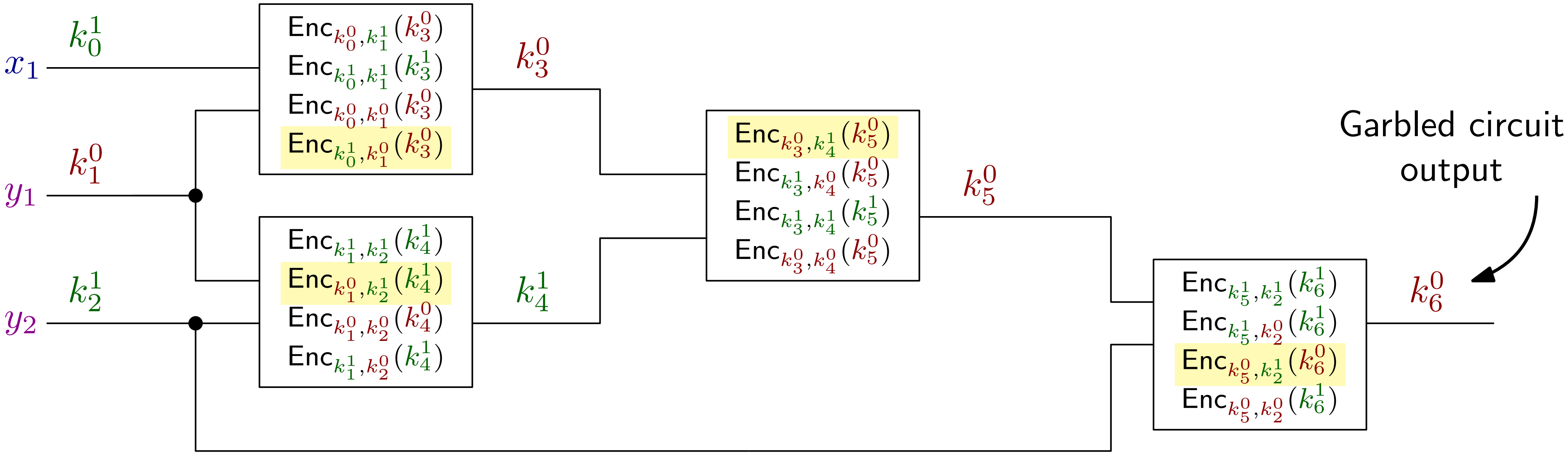
Suppose that Bob somehow knows the wire-labels corresponding to his input (we will handle this later)

Bob can evaluate the garbled circuit and recover the wire-label corresponding to the output wire

Once Bob knows the (garbled) circuit's output, he sends it to Alice

Yao's Garbled Circuits: Evaluating the Circuit

Bob receives the garbled circuit from Alice, together with the wire-labels of Alice's inputs



Suppose that Bob somehow knows the wire-labels corresponding to his input (we will handle this later)

Bob can evaluate the garbled circuit and recover the wire-label corresponding to the output wire

Once Bob knows the (garbled) circuit's output, he sends it to Alice

Alice knows whether the label she received corresponds to 0 or 1.
She learns $f(x_1, x_2, \ldots, x_m, y_1, y_2, \ldots, y_n)$